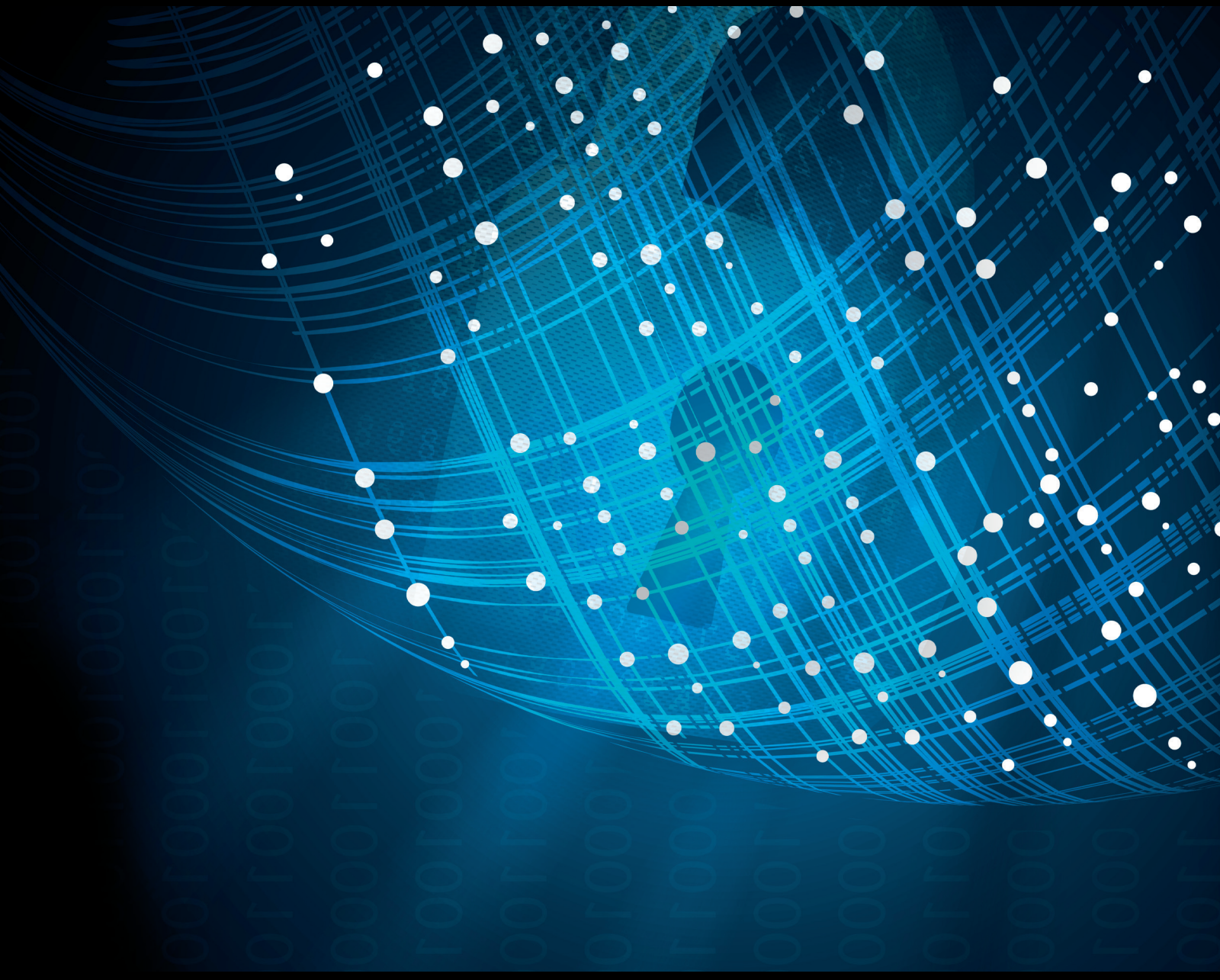
Security and Communication Networks

Recent Advancements in Digital Forensics in Recent Emerging Technologies

Lead Guest Editor: Farhan Ullah
Guest Editors: Sohail Jabbar and Mehmet Emin Aydin



Recent Advancements in Digital Forensics in Recent Emerging Technologies

Security and Communication Networks

Recent Advancements in Digital Forensics in Recent Emerging Technologies

Lead Guest Editor: Farhan Ullah
Guest Editors: Sohail Jabbar and Mehmet Emin Aydin

Contents

SESCon: Secure Ethereum Smart Contracts by Vulnerable Patterns' Detection
Amir Ali, Zain Ul Abideen, and Kalim Ullah
Research Article (14 pages), Article ID 2897565, Volume 2021 (2021)

Real-Time Surveillance Using Deep Learning
Muhammad Javed Iqbal, Muhammad Munwar Iqbal, Iftikhar Ahmad, Madini O. Alassafi, Ahmed S. Alfakeeh, and Ahmed Alhomoud
Research Article (17 pages), Article ID 6184756, Volume 2021 (2021)

Cell Traffic Prediction Based on Convolutional Neural Network for Software-Defined Ultra-Dense Visible Light Communication Networks
Shanjun Zhan, Lisu Yu, Zhen Wang, Yichen Du, Yan Yu, Qinghua Cao, Shuping Dang, and Zahid Khan
Research Article (10 pages), Article ID 9223965, Volume 2021 (2021)

Hindawi
Security and Communication Networks
Volume 2022, Article ID 5680357, 16 pages
https://doi.org/10.1155/2022/5680357

Research Article

An Efficient Lightweight Image Encryption Scheme Using Multichaos

Asad Ullah,[1] Awais Aziz Shah,[1,2] Jan Sher Khan,[3] Mazhar Sajjad,[4] Wadii Boulila,[5,6] Akif Akgul,[7] Junaid Masood,[8] Fuad A. Ghaleb,[9,10] Syed Aziz Shah,[10] and Jawad Ahmad[11]

[1]*School of Computing, Engineering and Physical Sciences, University of the West of Scotland, Paisely PA12BE, UK*
[2]*Department of Electrical and Informational Engineering (DEI), Polytechnic University of Bari, Bari, Italy*
[3]*Department of Electrical and Electronics Engineering, University of Gaziantep, Gaziantep, Türkiye*
[4]*Department of Computer Science, Comsats University, Islamabad 45550, Pakistan*
[5]*Robotics and Internet of Things Laboratory, Prince Sultan University, Riyadh 12435, Saudi Arabia*
[6]*RIADI Laboratory, University of Manouba, Manouba, Tunisia*
[7]*Department of Computer Engineering, Faculty of Engineering, Hitit University, Corum 19030, Türkiye*
[8]*Department of Computer Science, IQRA National University, Peshawar, Pakistan*
[9]*Department of Computer and Electronic Engineering, Sana'a Community College, Sana'a, Yemen*
[10]*School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia*
[11]*Research Centre for Intelligent Healthcare, Coventry University, Coventry, UK*

Correspondence should be addressed to Fuad A. Ghaleb; fuadeng@gmail.com

Received 28 July 2021; Revised 18 August 2021; Accepted 27 July 2022; Published 14 October 2022

Academic Editor: Farhan Ullah

With an immense increase in Internet multimedia applications over the past few years, digital content such as digital images are stored and shared over global networks, the probability for information leakage and illegal modifications to the digital content is at high risk. These digital images are transferred using the network bandwidth; therefore, secure encryption schemes facilitate both information security and bandwidth issues. Hence, a state-of-the-art lightweight information security methodology is required to address this challenge. The main objective of this work is to develop a lightweight nonlinear mechanism for digital image security using chaos theory. The proposed scheme starts by changing a plain image into an encrypted image to improve its security. A block cipher, using lightweight chaos, has been added to achieve this objective for digital image security. We utilized multiple chaotic maps to generate random keys for each channel. Also, Arnold cat map and chaotic gingerbread map are used to add confusion and diffusion. During the permutation stage, image pixels are permuted, while in diffusion stage, pixels are distorted utilizing gingerbread map to add more security. The proposed scheme has been validated using different security parameter tests such as correlation coefficient tests (CC), whose results have been observed closer to zero and information entropy (IE) value is 7.99, respectively, which is almost equal to the ideal value of 8. Moreover, number of pixels changing rate (NPCR) obtained value is higher than 99.50%, while the unified average changing intensity (UACI) is 33.33. Other parameters such as mean absolute error (MAE), mean square error (MSE), lower value of peak to signal noise ratio (PSNR), structural content (SC), maximum difference (MD), average difference (AD), normalized cross-correlation (NCC), and histogram analysis (HA) is tested. The computed values of the proposed scheme are better. The achieved results after comparison with existing schemes highlight that the proposed scheme is highly secure, lightweight, and feasible for real-time communications.

1. Introduction

There has been a rapid growth in the computers, Internet, social media, and communications during the last decade. The information revolution consists of digital media transmitted from billions of users and devices globally using network bandwidth. This increase in digital media transmission poses numerous challenges in terms of privacy and security as hacking incidents have occurred in the last decade that consists of hacking the content for information leakage and modifications [1–4]. More recently, numerous cryptographic methods have been proposed to enable secure image encryption. However, these methods often have an ordinary structure that performs the permutation and diffusion stages of decryption. Most of these algorithms deal with issues such as lack of robustness and security. Additionally, these encryption techniques were unsuitable for images with certain features, i.e., huge capacity, high redundancy, and high correlation among pixels. In addition to these characteristics, images also tend to be much larger than other media that might require decryption. These encryption plans require additional operations on compressed image data, demanding long computational time and high computing power. This translates to low encryption and decryption speeds in real-time communications, and images may present significant latency [5, 6].

With recent advancements in social media platforms, many images are transmitted over the web daily. Securing such images against unauthorized access becomes considerably more critical for authorized people and various fields to confront their unique security issues. Biology, military sciences, banking, and online shopping are just a few fields to name. In these cases, information transmitted as or via images is very different from information transmitted as or via text [7]. Images deal with greater data redundancy and more excellent information coverage while having a higher correlation between adjacent pixels. They also require real-time and robust security for communication purpose. Thus, numerous techniques are used to secure confidential image data transmitted over an insecure channel [8–12]. Data sharing is increased using social networks that are expanding in size and traffic, which has altered the underlying infrastructure of communication.

Today, much effort and research have created a complicated and robust cryptosystem providing the best possible security. Such a system uses several highly desired cryptographic properties, such as randomness, ergodicity, and sensitivity to original values [8].

As technology's scope ascends, new methods are developed, including special techniques such as deoxyribonucleic acid (DNA) coding [13], quantum theory, compression, and chaos theory. Though all of these have their advantages, chaos theory, in particular, has played a critical role in encryption research over the last 20 years [14–17]. By definition, the chaos theory uses mathematics and physics to deal with the behavior of NLDSs, i.e., nonlinear dynamical systems. The typical characteristics of chaotic systems are ergodicity, unpredictability, and initial value sensitivity. One advantage of employing chaos-based image encryption is that it offers higher security even though the mathematical complexity involved is much lower [18]. However, such systems' narrow and discontinuous range causes massive differences in chaotic properties due to minor disturbances such as noise factors. Two major factors that controls the performance of chaos-based encryption algorithms for imaging. The first such factor is based upon the robustness of the encryption algorithm against cryptanalysis. The second factor is based upon the real-time performance of chaotic maps. If the obtained performance is lower, this means that the achieved security robustness and efficiency of chaotic maps is lower. When both these factors are combined to attempt better results, many new challenges and difficulties also arise. So, their balance is essential in this regard.

This paper presents an efficient and lightweight image encryption technique based on multiple chaos algorithms. The work includes designing a novel and efficient information safety scheme based on various chaotic maps. Numerous encryption techniques are proposed and analyzed for their computational speed, reliability, robustness, and nonlinearity, which has been found exceptional. Our primary research objective is to achieve a high level of security. The security of a given system depends upon ensuring its cryptographic services. Several services of protection are provided by the International Telecommunication and Union's Telecommunication Standardization. We investigate the chaotic properties, including topology mixing, strange attractor, ergodicity, randomness, and dependence on its initial condition. After that, we evaluated the proposed cryptosystem using different tests, such as histogram analysis (HA), correlation coefficient (CC), the mean absolute error (MAE), differential attacks analysis (DAA), number of pixels changing rate (NPCR), unified average changing intensity (UACI), the mean square error (MSE), peak to signal noise ratio (PSNR), information entropy (IE), structural content (SC), normalized cross-correlation (NCC), maximum difference (MD), and average difference (AD). The comparison of proposed algorithms results in existing algorithms for all tests. The results highlight that calculated CC value is closer to zero, and IE value is 7.99, respectively, which is almost equal to the ideal value of 8. Moreover, NPCR is higher than 99.50%, while the UACI is 33.33. Other parameters such as MAE, MSE, lower value of PSNR, SC, MD, AD, and NCC showed better result in the proposed scheme. The offered scheme is compared to the existing schemes. It is concluded that the proposed scheme is highly secure, lightweight, and feasible for real-time communications.

The rest of the article has been organized as follows: Section 2 gives an overview and relation between chaos and cryptography, Section 3 deeply describes the proposed methodology of the lightweight image encryption scheme. The statistical analysis and experimental results are discussed in Section 4 and finally, Section 5 concludes the paper.

2. Chaos and Cryptography

It has been made clear that data transmitted over any public network is susceptible to malicious attacks and might become a target to be a break. For the sole purpose of protecting communications in general, several encryption techniques have been suggested. This has emphasized dynamic cryptosystems, such as chaos systems, that transmit plain data into unintelligible cipher data. Sensitivity is one of the unique features of such systems, i.e., any minor change in the original condition will result in drastically random output formulas such as the Lyapunov exponent help calculate the parameters of a map. Randomness, initial sensitivity, and periodicity are more beneficial properties that cryptographer use to design well-built cryptographic algorithms that provide security against unauthorized users. Among these systems, chaotic systems stand out due to their random output and other unique features [19]. As chaotic systems can perform without any dependence on parameters [19], they demonstrate in-determinism, thus helping other creators to design better algorithms. A chaotic output can be determined using its initial values for the system, making the chaotic more deterministic. When combined, these features help to diversify cryptosystems meticulously. When equipped with diffusion and confusion, a nonlinear system can effectively encounter cryptanalysis. Due to this robust infrastructure involved in chaotic cryptosystems, these processes are now being employed in other fields such as biological sciences, chemistry, and physics to build a new set of hybrid systems to achieve better security. Figure 1 illustrates the basic schematic chart of chaos-based encryption schemes.

In an NLDS, i.e., a nonlinear dynamical system, initial conditions significantly impact chaos, which seems to work upon and show pseudo-random behavior. Consider the stability of the system as an essential parameter when considering Lyapunov exponents. When the observer is aware of the system's initial conditions, the observer seems to understand the system output. However, if the preliminary requirements are unknown, the overall production seems to be highly erratic and random. When the source owner knows the pseudo-random order of the data, then the plain text can be easily substituted and diffused to be protected against malicious attacks and invasion. Several data formats can also be utilized in telecommunication channels that require protection.

2.1. Fundamental Properties of Chaotic System. Many technical and industrial areas within natural structures witness the chaos. These areas often exhibit well-defined possessions marking them as complicated and highly volatile. The chaos theory is concerned with situations that move through time toward a specific type of dynamic action. Multiple authors across various fields have discussed the chaos theory's mathematical background due to its wide variety of applications. In broad terms, some specific procedures are followed by these schemes for improvement. Individual nonlinear deterministic systems are usually potential sources of chaos. A long-term progression that is continuous, erratic, and fulfills mathematical benchmarks will display chaos phenomena. The features of a chaotic system can be described as a set of properties that measure mathematical principles that are used to describe chaos. The most notable of these characteristics include the following:

(1) Nonlinearity: The output is changed unproportionally concerning input.

(2) Deterministic: Every state of the system must follow some deterministic fundamental rules.

(3) Sensitivity to initial conditions: Slight deviation in its early state can lead to a dissimilar performance in its last state.

(4) Continued irregularity in the system's actions: The framework of chaotic systems is based upon a secret order combined with an infinite number of unstable random designs. This unseen direction forms the structure of irregular, chaotic systems.

(5) Long-term prediction: It can only be comprehended with limited accuracy as chaos is sensitive to the initial state.

2.2. Elementary Possessions of Chaotic Systems. Chaos has been measured in various laboratories, and natural systems [20, 21], covering many engineering and scientific areas, including fields such as biology, physics, ecology, meteorology, economics, and computer science, among others. All the techniques above provide valuable assets, which create certain unpredictability and a complex system. Systems that display such erratic and dynamic behavior over time are explained by chaos theory. For a broader overview, the concerned reader is directed to (Robinson, 1995). Generally, these systems are deterministic, and they follow a specific set of laws of evolution. Unfortunately, it must be said that chaos only occurs in a certain deterministic NLDS (nonregular linear dynamic systems). The most relevant of these criteria are dynamic instability, topological mixing, aperiodicity, and ergodicity:

Now let us examine the interaction and connection of chaos with cryptography properties of diffusion and confusion (Shannon, 1949). Suggesting to review chaotic systems features, like ergodicity, topology mixing property, and auto-similarity that are directly connected to the confusion phase of chaotic systems. The dynamics, or dynamical behavior in the chaotic output attractor, is concerned with nonperiodic orbits that produce identical patterns. The output patterns can be utilized to mask clear messages using substitution-like techniques. On the other hand, diffusion is directly connected to the system's sensitive parameters and initial conditions. Small changes in control parameters give rise to avalanche effects and produce random outputs.

As a conclusion, it would be best to list the benefits of chaotic cryptosystems that are being used and to offer strong security in cryptography. Firstly, chaotic systems seem to happen on their own and can be used for lightweight security in cryptography. These maps may show nonlinear behaviours that are used to make communication more secure. These systems have the advantage of having simple,

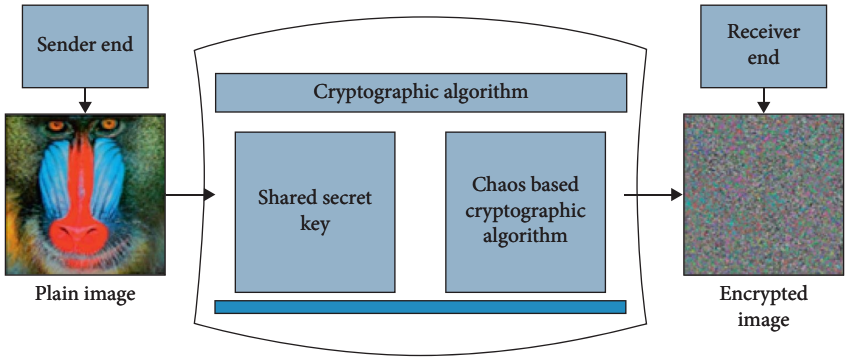


Figure 1: Basic schematic chart of chaos-based encryption process.

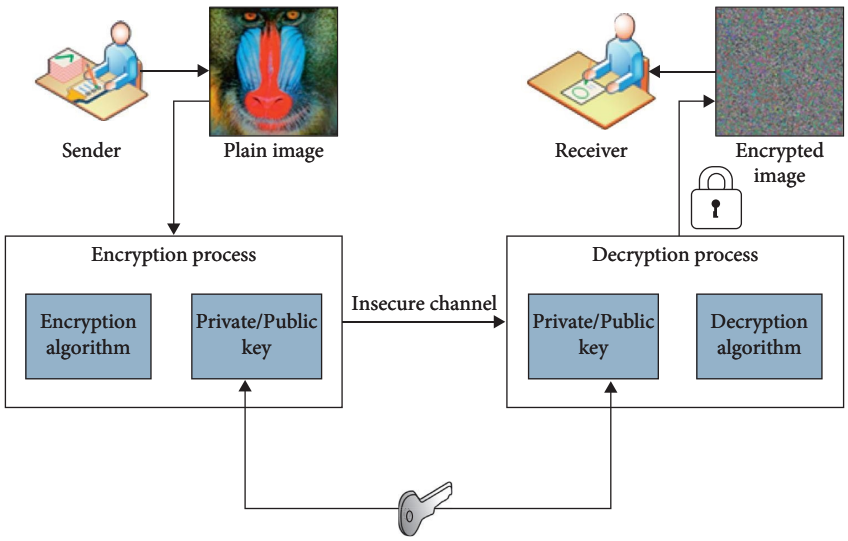


Figure 2: The process of encryption and decryption.

deterministic algorithms that are easy to use. Three basic ideas can be used to build a security infrastructure: confidentiality, integrity, and authentication. Confidentiality means that only people who are allowed to can see the data and people who are not should not. Integrity makes sure that the information available has not been changed. Availability means that there are resources that can be used.

3. The Proposed Methodology of Lightweight Image Encryption Scheme

Figure 2 shows the general process of encryption and decryption. Lightweight cryptosystems that operate based on chaos principles are designed by utilizing confusion and diffusion characteristics noted by Claude Shannon [22], as discussed earlier in Section 2. Such lightweight encryption systems work by using several chaotic maps to build hybrid systems that bring together several initial systems' best characteristics. The suggested method is then subjected to robust testing parameters compared with other existing techniques to determine its efficiency and efficacy.

In this section, the design of such a cryptosystem is discussed. This discussion will include all requirements that are mandatory for creating a truly secure cryptosystem. The proposed system is then investigated using several tests to secure the image and its transmission remains. During this process, we evaluate the performance of the proposed algorithm. We have conducted statistical tests including HA, CC tests between pixels along three primary axes, i.e., horizontal, diagonal, and vertical for each channel. MSE, PSNR, IE against each channel, sensitivity analysis which interpolates NPCR, and UACI. Other tests such as an AD, MD, SC, and NCC are performed. Our experiments enabled us to instantly examine output bit pixel, input bit pixel, and their behavior after investigation. Our primary goal is to create algorithms that include confusion and diffusion (randomness) since these are two essential properties for optimal security service.

3.1. Utilized Chaotic Maps. To help develop and implement encryption algorithms, general chaotic maps will be defined and discussed.

3.1.1. Arnold Scrambling Cat Map. Arnold's method is a standard technique for scrambling an image. This method is secure because of its high computational rate and quick processing time. Moreover, this method facilitates data scrambling in an array as a data stream using a chaotic Arnold transformation. Arnold's transformation chaotic map can be defined as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \mathrm{mod} 1, \tag{1}$$

where x and $y \in 0, 1$. The aforementioned formula of the Arnold chaotic map is for unit square and can be extended to multiple rows and columns depends upon the pixels of the image to be encrypted having a size of $N_{1(i,j)} \times N_{2(i,j)}$ In this case, then the above matrix can be extended to the finite number of pixels.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \mathrm{mod} N, \tag{2}$$

where x and $y \in \{0, 1, 2, 3, \ldots, N-1\}$ are the original images pixels scrambled and transformed to new positions, i.e., $\{x', y'\}$.

3.1.2. Gingerbread Man Chaotic Map. The gingerbread man is a 2D chaotic map that is one of the most widely utilized choices in chaotic, random sequencing. However, it is also essential to have a robust cryptosystem to satisfy all the designed algorithm's security needs when using this option. The formula for this 2D piecewise chaotic linear map is as follows:

$$\begin{aligned} x_{n+1} &= 1 - y_n + |x_n|, \\ y_{n+1} &= x_n. \end{aligned} \tag{3}$$

3.2. Image Encryption Process

3.2.1. Steps Involved in the Process of Image Encryption. The step wise flowchart of the proposed scheme is illustrated in Figure 3. The pseudo code of the image encryption process is illustrated in Algorithm 1. Let us consider a baboon test image measuring $256 \times 256 \times 3$ pixels that will be encrypted in the following process:

(1) First, this test image is fragmented into three channels (red, green, and blue) with an image size of 256×256 pixels.

(2) To make the scheme plaintext dependent, each layer is passed through SHA-512. Then, the utilized chaotic maps such as logistic map, Gaussian map, and Chebyshev chaotic map initial conditions (keys) are computed using the calculated hash values.

(3) In this phase, the original image is permuted (confused) using 2D Arnold scrambling. The plaintext channels are permuted row- and column-wise from their initial position to the maximum iterative state.

(4) Next, two random and arbitrary chaotic matrices are generated using a gingerbread man chaotic map. These two random matrices are bitwise XORed with each other to develop diffused channels, and the resultant matrix is bitwise XORed with the red permuted channel.

(5) To make the scheme ciphertext dependent, the resultant red channel diffused channel is bitwise XORed with a green permuted channel. Their result matrix is bitwise XORed with a blue permuted channel.

(6) The multi-layered cryptosystem then produced three sets of encrypted images by applying different chaotic maps.

(7) For the final step, the cat command module is employed to combine the three encrypted channels and generate the final ciphertext image.

3.2.2. Steps Involved for Image Decryption

(1) Initially, the encrypted image is divided into its respective three parts (R, *G*, B).

(2) The green and blue cipher channels are bitwise XORed to get a blue permuted channel. Then the red and blue cipher channels are bitwise XORed to obtain a blue permuted channel.

(3) Two random matrices are generated using a gingerbread man chaotic map. These two random matrices are bitwise XORed with each other, and the resultant matrix is bitwise XORed with the red cipher channel to produce a red permuted layer.

(4) The 2D Arnold chaotic map is then used to reiterate the three different layers row- and column-wise from maximum permuted (iterated) state to the original condition.

(5) Three plain channels are recovered after employing Arnold cat scrambling for the reiteration process.

(6) The plain grey channels are combined using cat command

(7) The decrypted full-colored image is retrieved.

4. Statistical Analysis and Experimental Results

Experiments are conducted utilizing multiple plain images to demonstrate the effectiveness of the proposed scheme. A good encryption strategy should be robust to address the needs engendered by cryptanalysis, statistics, and brute force attacks, respectively. Along with these considerations, the goal is to analyze security investigations to demonstrate the effectiveness of the proposed approach against widely-recognized attacks. To illustrate the strength of our developed methodology, a statistical investigation has been conducted by creating a schematic histogram and correlating the nearby pixels in both plain and cipher images.

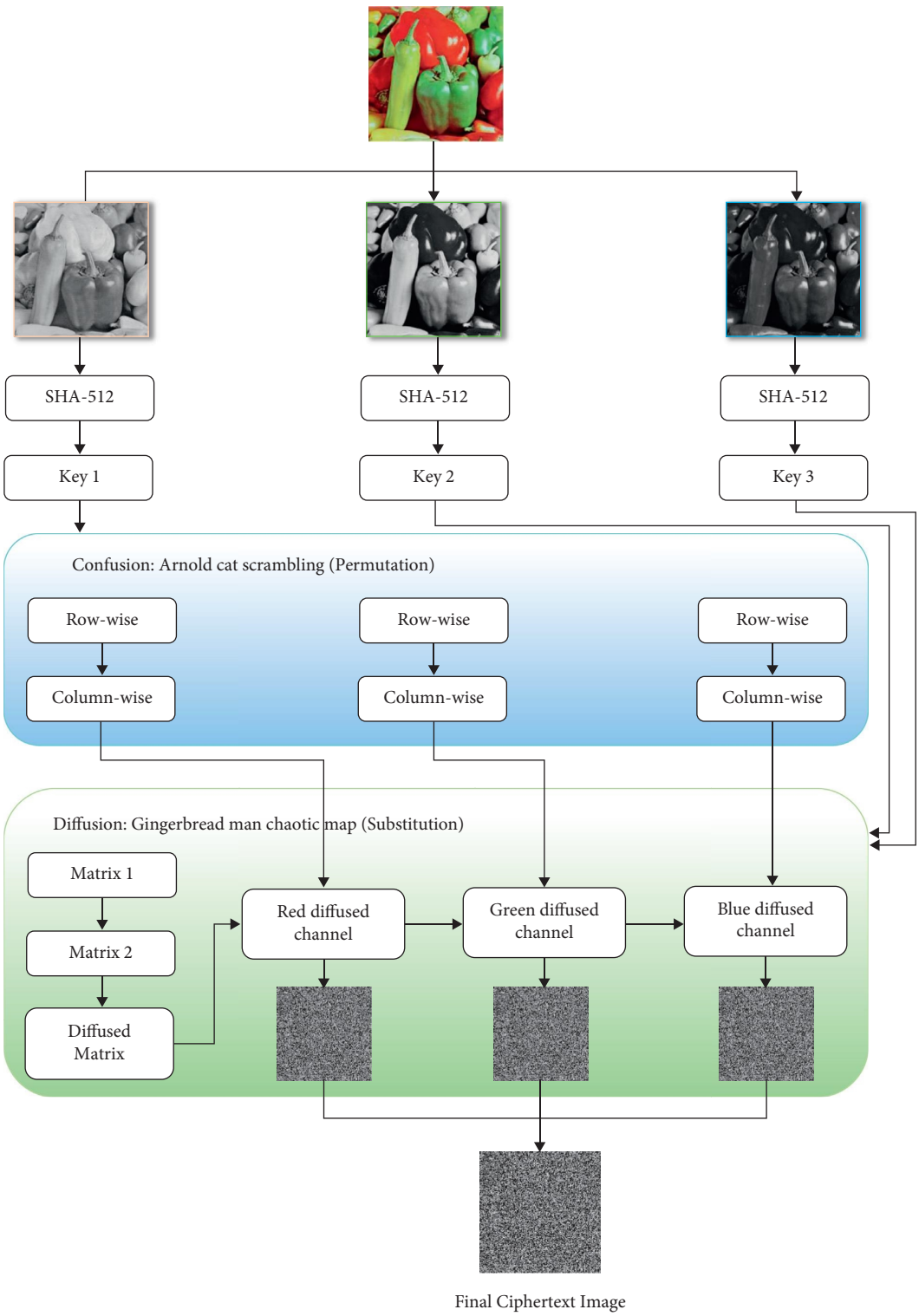


Figure 3: Flow chart of the proposed scheme.

4.1. Histogram Analysis (HA). It is widely used as a tool for analyzing the security of encrypted images. The original content in the plain image must be secure. This test explains that the grey-level pixel value should be equally distributed in its defined range depending on the image. In the cipher image, the distribution of pixels is always uniform, while in the case of a plain image, these pixels are bumpy. The bumpy pixels demonstrate that anyone can access the pixel information easily. High peaks of pixels reveal that maximum information lies at those points while low peaks indicate that minimum information lies at this point. A secure image histogram

Inputs: Plaintext image P, its row and column numbers row, col,
Outputs: Ciphertext image C
(1) for $i = 1$ *to* row
(2) for $j = 1$ to col
(3) $R, G, B \leftarrow P$
(4) Key1, Key2, Key3$\leftarrow SHA256(R, G, B)$
(5) $R, G, B \leftarrow$Row- and column −wisepermutation
(6) Permuted$(R, G, B) \leftarrow$bit − wiseXORed
(7) **Endfor**
(8) **Endfor**
(9) $C \leftarrow$Diffused(R, G, B)

Algorithm 1: Image encryption.

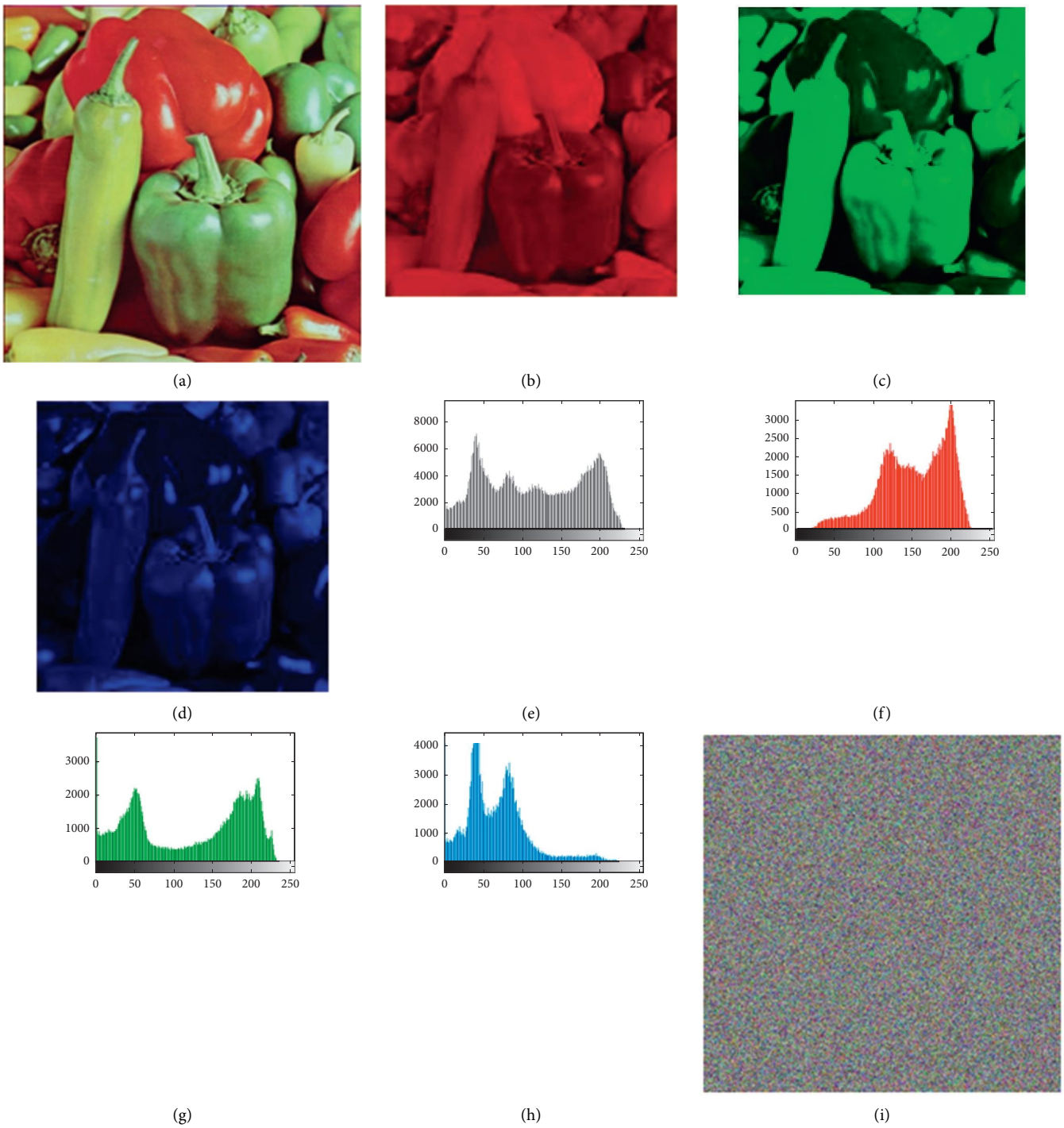


Figure 4: Continued.

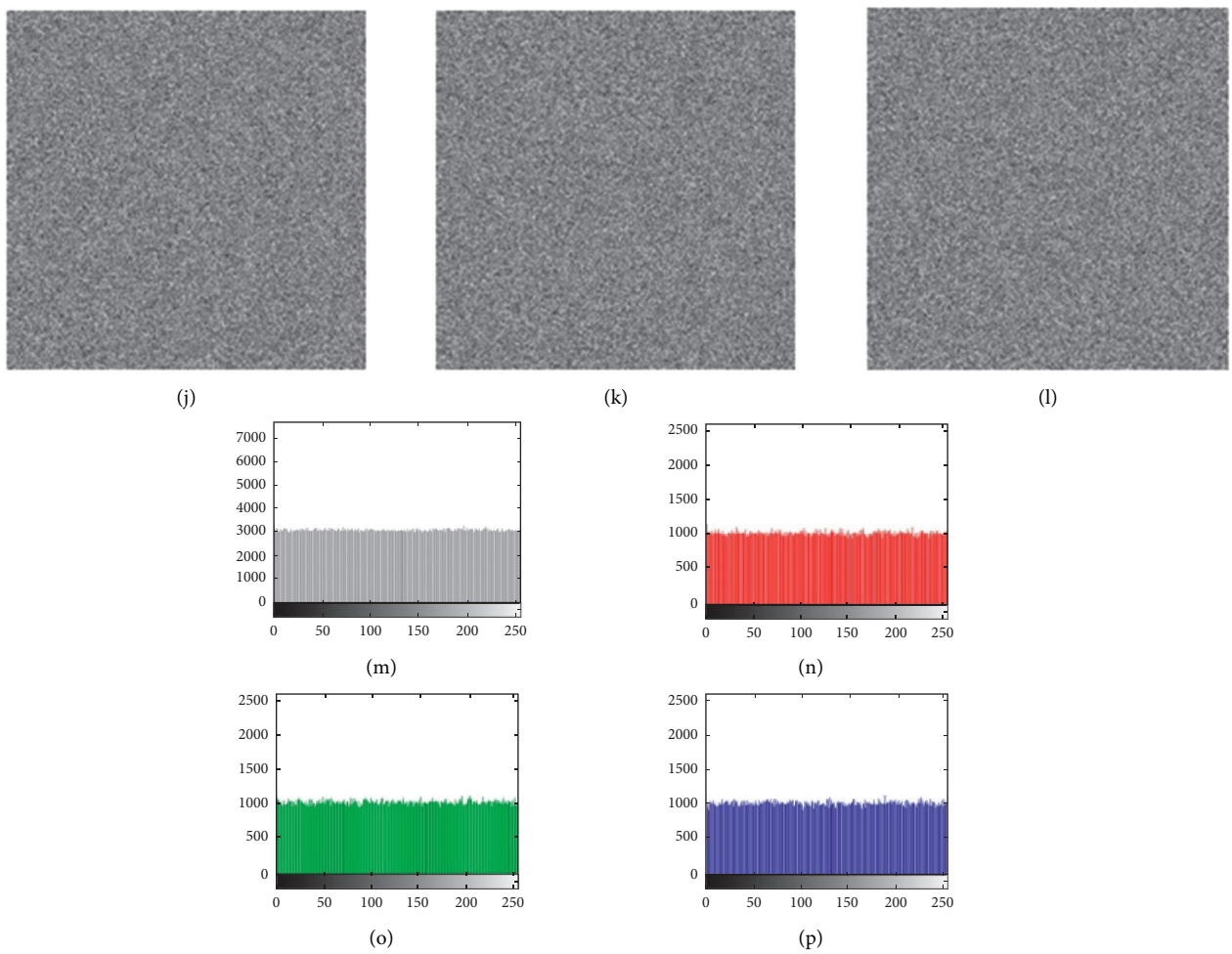


Figure 4: Encryption, decryption, and histogram results of the proposed scheme. (a, b, c, and d) Plaintext Pepper, plaintext red layer Pepper, plaintext green layer Pepper, and plaintext blue layer Pepper images, respectively; (e, f, g, and h) corresponding histograms; (i, j, k, and l) corresponding ciphertext images; (m, n, o, and p) ciphertext histograms.

creates a uniform pattern with minimal peaks and valleys, making it difficult for hackers to read the actual content due to pixel uniformity. The histogram shows the statistical quality of various images. It also discloses how arbitrary numbers created from chaotic maps, such as white noise, are uniformly dispersed. The histograms of plain images and encrypted images are given in Figure 4.

4.2. Correlation Coefficient Analysis (CC). It is used to determine the relationship between two variables. To discover the relationship between two variables, in our case, CC analysis is used. CC finds the quality of encrypted image by analyzing pixels distribution [23]. How effective a cryptosystem is can be measured by the encryption algorithm's ability to mask all plain image characteristics. Furthermore, it relies on an image's randomness and how uncorrelated the cipher image is [23–25]. If the CC for the two images (i.e., plain and encrypted) is low or close to zero, the proposed system is secure. In comparison, in the case of highly similar pixels between the two images, this demonstrates that the system is insecure. The defined range of correlation falls within [−1 1]. The high correlation of pixels between the two images reveals that the system is susceptible to many types of attacks and that malicious third parties can easily detect an image's original confidential content. In the case of encrypted images, the pixels are scattered in its defined range, depicting that the proposed system is secure. Figure 5 demonstrates the correlation plots for Pepper image. The test is applied on color and a channel-wise image with an image size of $512 \times 512 \times 3$ and 512×512 for three grey channels, respectively. Thus, one can confirm that in case of plaintext, the correlation among the adjacent pixels is very much strong and the plots are diagonally condensed. While in case of ciphertext, the plots are highly scattered which confirms high dissimilarity and randomness generated using the presented

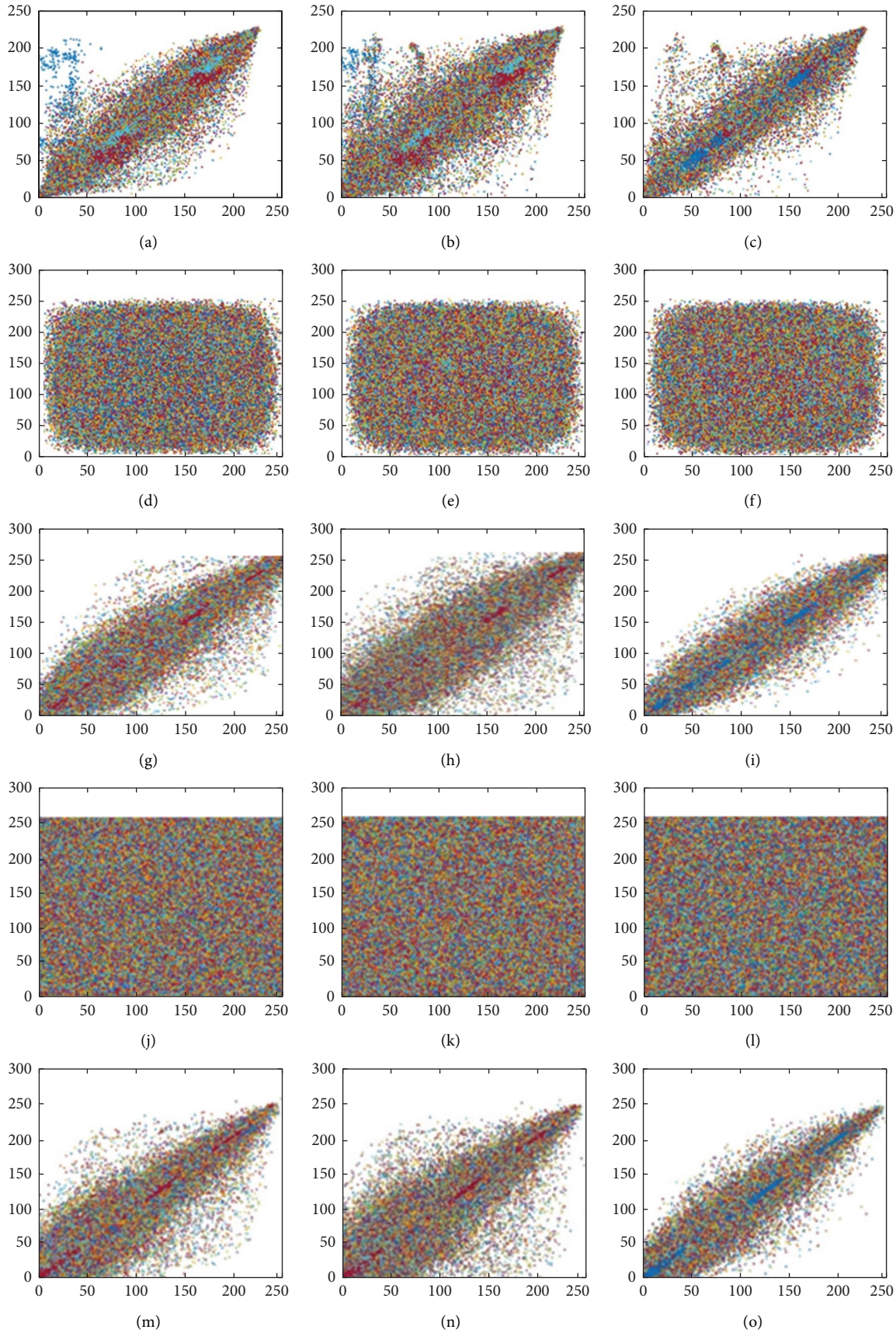


Figure 5: Continued.

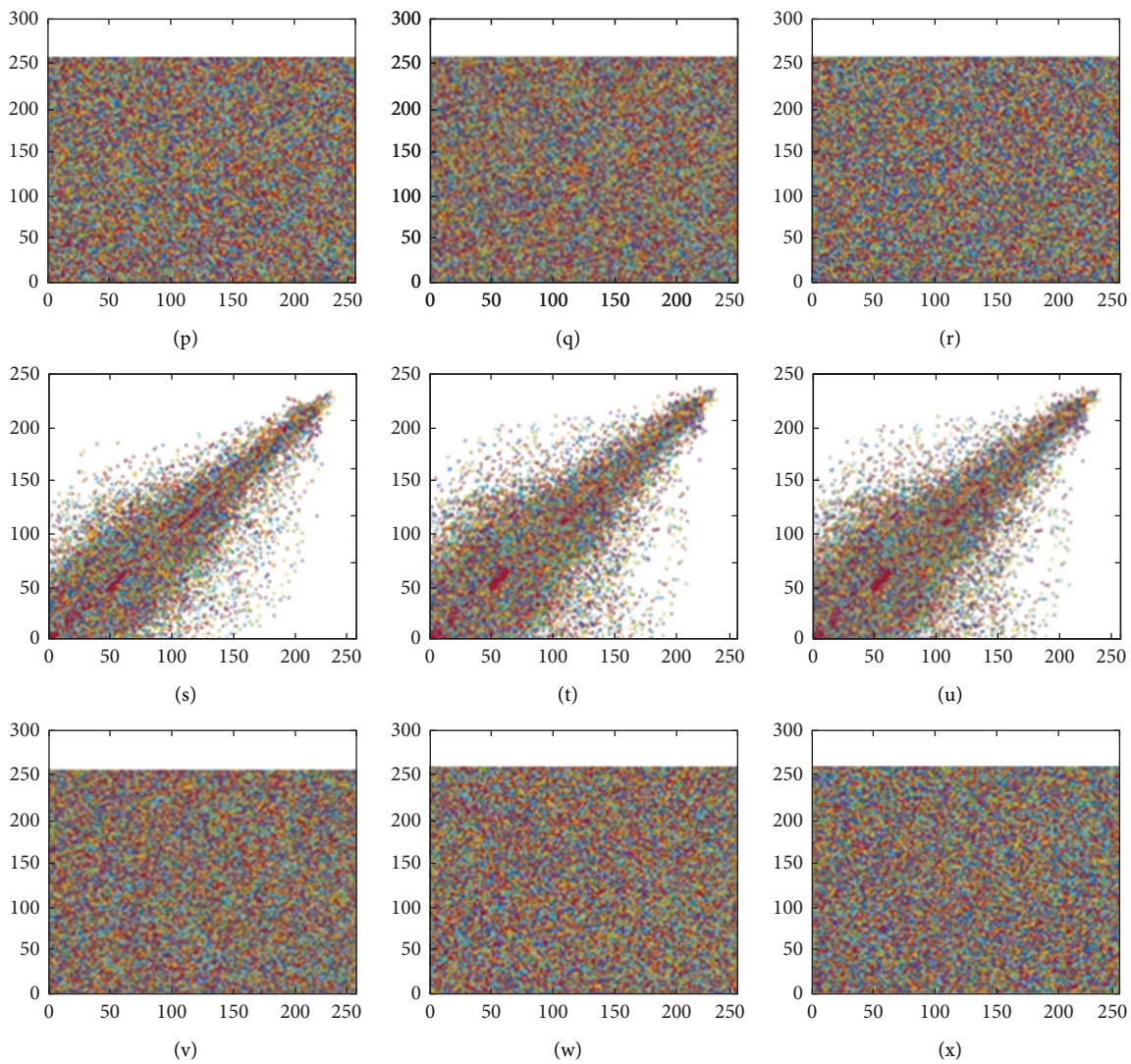


Figure 5: Pepper image correlation plots: (a, b, and c) plaintext Pepper correlation plots in horizontal, diagonal and vertical direction while row (1) (d, e, and f) corresponding ciphertext plots; (g, h, and i) plaintext Pepper red layer correlation plots in horizontal, diagonal and vertical direction; (j, k, and l) corresponding ciphertext plots; (m, n, and o) plaintext Pepper green layer correlation plots in horizontal, diagonal and vertical direction; (p, q, and r) corresponding ciphertext plots; (s, t, and u) plaintext Pepper blue layer correlation plots in horizontal, diagonal and vertical direction; (v, w, and x) corresponding ciphertext plots.

Table 1: Correlation coefficient values for Pepper image of size 512×512.

	Plaintext image Directions			Encrypted image Directions		
	HC	DC	VC	HC	DC	VC
Ideal	1	1	1	−1	−1	−1
Pepper	0.9635	0.9564	0.9663	−0.0033	0.0011	0.007
Ref. [28]	—	—	—	0.0075	0.0012	0.0049
Ref. [29]	—	—	—	0.0005	0.0008	0.0011
Ref. [30]	—	—	—	0.0117	0.0026	0.0010
Ref. [31]	—	—	—	0.0043	0.0054	0.0072
Ref. [32]	—	—	—	0.0108	0.0181	0.0061
Ref. [33]	—	—	—	0.0032	0.0042	0.0018
Ref. [34]	—	—	—	0.0204	−0.0174	0.0231
Ref. [35]	—	—	—	0.0053	−0.0027	0.0016

HC : Horizontally correlated; DC : Diagonally correlated; VC : Vertically correlated.

scheme. These scattered plots verify the scheme resistant to an attack. Correlation can be mathematically described as [26, 27]

$$r = \frac{S_{xy}}{S_x S_y}. \tag{4}$$

where the system's covariance is S_{xy}, while standard deviations of random variables of S_x, S_y, are x, and y, respectively. We applied the CC on the grey images of the same size as having a length of subsequently combined all the encrypted grey channels for the colored image. We also applied the CC test for the second time on colored images having a size of respectively. The computed values of the CC are shown in Table 1.

4.3. The Mean Absolute Error (MAE). MAE is one of the most important criteria required to determine an image's quality. It is also used to test how resistant a method is toward differential attacks. To analyze, let's consider the image being the total size of a test image. Let the C denotes the ciphered image, and P denotes the grey pixels of a plain image at the i th row and J th column. The equation used to find the mean absolute error is

$$MAE = \frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left| C_{(i,j)} - P_{(i,j)} \right|. \tag{5}$$

It is essential to obtain a higher value of MAE to validate the cryptosystem's robustness. If the evaluated MAE value is not higher than the existing scheme's average values, then the system has not resisted the differential attack. In other words, the higher the MAE, the stronger the security level of the designed cryptosystem. However, the resistant differential attack may still be fulfilled if the MAE computed value is not higher than the existing cryptosystems' values. The minimum value should be 75, which means that value higher than 75 demonstrates a high-security level. Our findings show that the proposed algorithm's calculated values are higher than 75: specifically, the estimated MAE value for the pepper image is 80, while the MAE for the Airplane image is 92, both of which are significantly higher than 75. Meanwhile, the MAE value of the "Tiffany" image over is two times higher than either of those, as shown in Table 2.

4.4. Differential Attacks Analysis. The randomness of the proposed encryption scheme can be evaluated using permutation method. Permutation's ability is to highlight even small changes in plain images. To check the proposed system's sensitivity level, we change a single pixel in the plain image and then calculate its respective outcome. As discussed in previous sections, small changes in the plain image can lead to drastic changes in the encrypted image, making the connection between original and cipher images very difficult and makes it more difficult for unauthorized third parties to decode them. Thus, the alteration of pixels in the plain image allows us to expand the encryption algorithm.

Table 2: MAE test values.

Image	Size	MAE
Pepper	512 × 512	80
Tiffany	512 × 512	182
Airplane	512 × 512	92
Ref. [36] pepper	256 × 256	74.93
Ref. [36] tiffany	256 × 256	94.36
Ref. [36] airplane	256 × 256	82.88

Table 3: AND UACI test values for different image (512 × 512) channels.

Image	Test type	R–C	G–C	B–C
Pepper	UACI	99.613%	99.613%	99.613%
Pepper		33.01	33.39	33.75
Tiffany	UACI	99.626%	99.626%	99.626%
Tiffany		36.13	36.13	36.13

Table 4: Computed and UACI values comparison with other schemes.

Image	Test type	Combined value
Pepper	UACI	99.613%
Pepper		33.38
Tiffany		99.626%
Tiffany		36.13
Ref. [37] pepper	UACI	99.589%
Ref. [37] pepper		33.373
Ref. [38] pepper	UACI	99.01%
Ref. [38] pepper		33.51
Ref. [39] pepper	UACI	99.22%
Ref. [39] pepper		33.12
Ref. [34] pepper	UACI	99.61%
Ref. [34] pepper		33.48
Ref. [40] pepper	UACI	99.60%
Ref. [40] pepper		33.41

The techniques of UACI and NPCR are usually employed to measure the effectiveness of pixel alteration.

4.4.1. Number of Pixels Changing Rate (NPCR). Several pixels' change rate examines how many pixels transmute when a single pixel alters in the plaintext image. This value reaches 99.60% in NPCR, showing us that the system has approached cautiously as it can easily defend itself against plain text attack. Ideally, the value for NPCR should be 100%. Tables 3 and 4 indicate that our proposed system achieves a calculated value of 99.613% for the encrypted pepper image across all three channels. In contrast, the Tiffany image's calculated; value is 99.626% for its all respective channels, as shown in Table 3. The proposed values and preexisting algorithms values are subsequently shown in Table 4, which compares the values we achieve to the values achievable by other existing systems. Table 4 reveals that both images achieve a higher proposed calculated value than those achieved by existing methods.

Two cipher images we consider for this evaluation are C1 and C2. However, their source image differs from a single

pixel. The equation used to calculate the value is as follows [41]:

$$= \frac{\sum_{i,j} D_{i,j}}{W \times H} \times 100, \tag{6}$$

where $W \times H$ denotes the total dimension of an image and can be illustrated as follows:

Case 1:

$$\begin{aligned} D_{(i,j)} &= 0, \\ C_{1(i,j)} &= C_{2(i,j)}. \end{aligned} \tag{7}$$

Case 2:

$$\begin{aligned} D_{(i,j)} &= 1, \\ C_{1(i,j)} &\neq C_{2(i,j)}. \end{aligned} \tag{8}$$

4.4.2. Unified Average Changing Intensity (UCAI). UACI is used to test and compare the intermediate intensities of both plain and encrypted images. UACI can be defined as [42] follows:

$$\text{UACI} = \frac{1}{W \times H} \sum_{i,j} \left[\frac{C_{1(i,j)} - C_{2(i,j)}}{255} \right] 100\%. \tag{9}$$

Here, $C_{1(i,j)}$ and $C_{2(i,j)}$ are the two ciphered images, while $W \times H$ denotes the total dimension of an image. We have considered a colored image that is divided into three layers of the same size. The evaluated results for all the images are higher than the average value of 33%, which indicates that the system has added a great deal of randomness into the proposed cryptosystem. For the pepper image, the algorithmic proposed value for those three channels (R, G, B) is 33.01, 33.39, and 33.75, respectively, as tabulated in Table 3. The Tiffany image's evaluated average values, and its three channels (R, G, B) are 36 for all three channels. Table 4 compares the values shown with preexisting values from [34, 37–40], which are reported as 33.373, 33.51, 33.12, 33.48, and 33.41, respectively. These results indicate that the presented scheme has produced a higher security level against any type of external attack.

4.5. The Mean Square Error (MSE). MSE can be evaluated using the difference between the two images. When the pixels of the plain and encrypted images are squared and then averaged, we get MSE that is calculated as follows [34, 37–40]:

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(P_{(i,j)} - C_{(i,j)} \right)^2, \tag{10}$$

where $M \times N$ represents the cumulative size of an image, while i and j represent image rows and columns, p and C are the plain and ciphered images at the ith row and jth column, respectively. A higher value of MSE shows that our suggested scheme has a greater level of robustness. Table 5 contains several values for MSE and PSNR, including an average layer value of 10,000. Thus, we conclude that the proposed algorithm has excellent potential in the field of secure communications.

Table 5: MSE and PSNR values for different channels of Pepper and Tiffany image.

Image	Image channel	Size	MSE	PSNR
Pepper	Red	512 × 512	11117.33	7.70
	Green	512 × 512	11222.95	7.66
	Blue	512 × 512	7988.03	9.14
Tiffany	Red	256 × 256	177701.01	5.67
	Green	256 × 256	13143.97	6.89
	Blue	256 × 256	7347.84	9.50

4.6. Peak to Signal Noise Ratio (PSNR). The image quality can be measured by utilizing the PSNR test. PSNR can be calculated as follows [34, 37–40]:

$$\text{PSNR} = 10 \log_2 \left(\frac{I_{\max^2}}{MSE} \right). \tag{11}$$

Here I_max denotes the maximum number of pixels of the test image. of pixels of the test image. PSNR is usually calculated in terms of decibels (dB). While MSE achieved value is better to ensure robustness of the cryptosystem's security, PSNR should have a lower value for better data protection. The aforementioned Table 5 shows different PSNR numbers for a set of additional test images. The average comes out to be 7, which satisfies the requirements of a robust cryptosystem.

4.7. Information Entropy (IE). This test provides key points regarding self-information. A report of any encryption channel requires entropy, and uncertainty itself relies upon entropy that is commonly known as randomness. IE can explain the degree to which uncertainties are present in communication channels [43]. According to Claude Shannon, as early as 1949, information theory is based on the mathematics of data transfer and storage [28]. Today, information theory is linked to communication systems, cryptology, data compression, and many other similar topics [44, 45]. Thus, entropy is the most significant characteristic of uncertainty and unpredictability since it demonstrates irregularity in the behavior of both plain and encrypted data. For the best-encrypted result, the cryptosystem must achieve a value of entropy as near eight as possible. IE of any message can be calculated as follows [46]:

$$H(m) = \sum_{j=0}^{N-1} p(x_j) \log_b p(x_j). \tag{12}$$

This explains the probability of any symbol to occur. Suppose there is a truly random source generating about 28 symbols with an equal probability m will be $= m_1 \ldots\ldots\ldots m_{28}$, where every symbol is shown by 8 bits. Upon applying this for Eq. (10), an IE value of $H(m) = 8$ bits will be attained corresponding to a uniform random source. As a whole, the ideal value is always greater than the IE of the actual source. This is because actual data rarely sends out randomized information. In this case, as mentioned above, a

Table 6: Entropy values for different channels of Pepper and Tiffany image.

Image	Image channel	Size	Computed entropy
Pepper	Red	512 × 512	7.999
	Green	512 × 512	7.999
	Blue	512 × 512	7.999
dTiffany	Red	256 × 256	7.999
	Green	256 × 256	7.999
	Blue	256 × 256	7.999
Splash	Red	512 × 512	7.999
	Green	512 × 512	7.999
	Blue	512 × 512	7.999
Tiffany	Red	256 × 256	7.997
	Green	256 × 256	7.997
	Blue	256 × 256	7.997
Airplane	Red	512 × 512	7.999
	Green	512 × 512	7.999
	Blue	512 × 512	7.999

Table 7: Comparison of computed entropy values and pre-existing algorithms.

Image	Dimension	Entropy
Proposed	512 × 512	7.999
Reference [47]-lena	512 × 512	7.996
Reference [48]-lena	512 × 512	7.997
Reference [40] (Sun's algorithm)	512 × 512	7.9965
Reference [40] (Baptista's algorithm)	256 × 256	7.9690
Reference [40] (Xiang's algorithm)	256 × 256	7.9950

Table 8: SC NCC, MD, and AD computed values.

Image	SC	NCC	MD	AD
Proposed	0.8004	1	223	34.1219
Tiffany	2.0954	1	255	89.1378

certain level of predictability arises when the IE comes out to be less than 8 bits [44, 45]. Entropy remaining close to the ideal value avoids IE attacks [44, 45]. Table 6 and 7 demonstrates the computed IE values for the proposed cryptosystem and its comparison with pre-existing schemes, respectively.

4.8. Structural Content (SC). SC is involved in the provision of aggregate weight regarding a specific plain signal to a coded or already present signal. A value of 1 for SC implicates that the image's quality is enhanced, while a more considerable value indicates that the image quality will be very low. Mathematically, SC is written as follows:

$$SC = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(y_{(i,j)}\right)^2}{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(x_{(i,j)}\right)^2}, \tag{13}$$

where, on account of the plain and encrypted images, SC estimation is not close to unity because the encryption scheme includes substitution, permutation-like noise, and commotion in the plain image. SC cannot be approximated as one if all of the shading images are advanced. SC calculated values for the proposed image encryption are illustrated in Table 8.

4.9. Normalized Cross-Correlation (NCC). NCC is a substantial test used to find the sets of similarity in image sets, particularly a plain image and its respective encryption of the same size. This test is widely utilized in image processing to find the quality of an image as well. The range of NCC falls between −1 and 1. Here −1 shows a strong correlation between the plain and encrypted images. Also, 1 shows that the correlation between two images is not strong, also known as a perfectly inverse correlation. Mathematically, NCC is defined as

$$NCC = \frac{1}{N} \times \frac{\mathrm{sum}\left(\left[x_n - \mathrm{mean}_x\right] \times \left[y_n - \mathrm{mean}_y\right]\right)}{\sqrt{\mathrm{var}_x \times var_y}}, \tag{14}$$

where $M \times N$ is the total size of an image and var shows variance of image between $x_{i,j}$ and $y_{i,j}$. Meanwhile, i and j show the actual position (row and column) of the pixels, and mean_x and mean_y show the mean level of the image x and y. The computed values for our two test images "pepper" and "baboon" are shown in Table 8. The value for both the test images is 1, which clearly demonstrates that the proposed work ensures higher security.

4.10. Maximum Difference (MD). MD is another criterion widely adapted for image security. This test is used to find the actual difference that has been created between the plain and encrypted images. The higher the value of MD depicts the fundamental difference between the plain and encrypted images. Mathematically, maximum distance is defined as:

$$MD = \mathrm{MAX}\left|x_{(i,j)} - y_{(i,j)}\right|, \tag{15}$$

where MAX show the actual maximum difference between plain image $x_{(i,j)}$ at ith row and jth column and encrypted image $y_{(i,j)}$ at ith row and jth column. It is important to achieve maximum value for a secure system. The evaluated values for our test images "Pepper" and "Tiffany" are recounted in subsequent Table 8. The values of 223 for the pepper image and 255 for the Tiffany image (Table 8) validates the security of the proposed scheme.

4.11. Average Difference (AD). AD is another important test utilized by several research types in image processing, object detection and recognition, and security application. The test utilizes two images, i.e., the plain image and its encrypted image counterpart, in the formula that functions as follows:

$$AD = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(x_{(i,j)} - y_{(i,j)}\right), \tag{16}$$

where $M \times N$ is the cumulative size of an image while $x_{(i,j)}$ and $y_{(i,j)}$ are two images at the ith row and jth

column, respectively. Table 8 shows the evaluated values of the average difference of Pepper and Tiffany images, which are 34.1219 and 89.1378, respectively. These results demonstrate that a high level of security is generated using the proposed system.

5. Ablation Study

The designed image encryption algorithm is based on chaotic maps. If there is a small change in key, it would not be possible to decrypt the original image. To reproduce the exact results obtained in this work is to use exact key. Moreover, if the size of the images is different than it would also be not possible to get exact same encryption results.

6. Conclusion and Future Work

This article discusses several security parameters that can be utilized to implement a robust encryption scheme and further perform evaluations using permutations to demonstrate how such methods would prevent various types of attacks and malicious interference. In the proposed work, we have also added multiple security layers, particularly multiple-dimensional chaotic iterative maps. Furthermore, in the permutation process, the actual value of image pixels was changed using the 2D gingerbread chaotic map to strengthen the security feature. The aforementioned chaotic maps generated a highly secure ciphertext image. The proposed scheme has been tested using various security parameters, i.e., histogram analysis, correlation coefficient, the mean absolute error, differential attacks analysis, number of pixels changing rate, unified average changing intensity, the mean square error, the peak to signal noise ratio, information entropy, structural content, normalized cross-correlation, maximum difference, and average difference. The results demonstrate the effectiveness of the proposed scheme against cryptographic and differential analysis attacks. The scheme presented in this paper uses various lightweight chaos maps, and hence it could also be tested in real-time applications. The proposed scheme is tested in MATLAB software; however, real-time implementation of FPGA should also be tested. Moreover, the proposed scheme should be tested for audio and video applications as well. [49–52] This work is developed for symmetric key encryption only. We aim to enhance the proposed scheme for asymmetric key encryption. In future, we will explore using our suggested cryptographic techniques to secure other kinds of data, such as diabetes classifications and biological data [53].

Data Availability

Data will be available upon request to the corresponding author.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Authors' Contributions

A. U., A. A. S., J. S. K., M. S., W. B., A. A., J. M., and F. A. G. performed formal analysis and original draft preparation. S. A. S. and J. A. proposed the main ideas and validated analysis. A. U., J. S. K., N. R., and J.A. crystallized framework and also revised the manuscript. All authors have read and agreed to the published version of the manuscript.

References

[1] A. Ghaleb, F. Saeed, M. Al-Sarem et al., "Misbehavior-aware on-demand collaborative intrusion detection system using distributed ensemble learning for VANET," *Electronics*, vol. 9, p. 1411, 2020.

[2] M. Alkhelaiwi, W. Boulila, J. Ahmad, A. Koubaa, and M. Driss, "An efficient approach based on privacy-preserving deep learning for satellite image classification," *Remote Sensing*, vol. 13, p. 2221, 2021.

[3] M. Driss, D. Hasan, W. Boulila, and J. Ahmad, "Microservices in IoT security: current solutions, research challenges, and future directions," *Procedia Computer Science*, vol. 192, pp. 2385–2395, 2021.

[4] M. A. Khan, M. A. Khan Khattk, S. Latif et al., "Voting classifier-based intrusion detection for iot networks," *Advances On Smart And Soft Computing*, vol. 1399, pp. 313–328, 2022.

[5] F. Masood, J. Ahmad, S. A. Shah, S. S. Jamal, and I. Hussain, "A novel hybrid secure image encryption based on julia set of fractals and 3D lorenz chaotic map," *Entropy*, vol. 22, no. 3, pp. 274–709, 2020.

[6] F. Masood, M. Driss, W. Boulila et al., "A Lightweight Chaos-Based Medical Image Encryption Scheme Using Random Shuffling and XOR Operations," *Wireless Personal Communications*, pp. 1–28, 2021.

[7] W. Stallings, *Cryptography and Network Security*, Vol. 4, E. Pearson Education India, Noida, India, 2006.

[8] H C. Cheng, Y Y. Zhi, L. GuoShiang, and H. ZengWei, "A virtual optical encryption software system for image security," *Journal of Convergence Information Technology*, vol. 6, no. 2, pp. 357–364, 2011.

[9] H. M. Al-Najjar, "Digital image encryption algorithm based on multi-dimensional chaotic system and pixels location," *International Journal of Computer Theory and Engineering*, vol. 4, no. 3, pp. 357–354, 2012.

[10] A. KrBanthia and N. Tiwari, "Image encryption using pseudo random number generators," *International Journal of Computer Application*, vol. 67, no. 20, pp. 1–8, 2013.

[11] R. L. Rivest, *Cryptography. In Algorithms and Complexity*, pp. 717–755, 1990.

[12] Z. Kartit, A. Azougaghe, H. K. Idrissi et al., "Applying encryption algorithm for data security in cloud storage," in *Advances in Ubiquitous Networking*, pp. 141–154, Springer, Singapore, 2016, [8].

[13] F. Masood, W. Boulila, J. Ahmad, S. Sankar, S. Rubaiee, and W. J. Buchanan, "A novel privacy approach of digital aerial images based on mersenne twister method with DNA genetic encoding and chaos," *Remote Sensing*, vol. 12, p. 1893, 2020.

[14] A. Akhavan, A. Samsudin, and A. Akhshani, "Cryptanalysis of an image encryption algorithm based on DNA encoding," *Optics & Laser Technology*, vol. 95, pp. 94–99, 2017.

[15] M. S. Baptista, "Cryptography with chaos," *Physics Letters A*, vol. 240, no. 1-2, pp. 50–54, 1998.

[16] R. Parvaz and M. Zarebnia, "A combination chaotic system and application in color image encryption," *Optics & Laser Technology*, vol. 101, pp. 30–41, 2018.

[17] E. Solak, R. Rhouma, and S. Belghith, "Cryptanalysis of a multi-chaotic systems based image cryptosystem," *Optics Communications*, vol. 283, no. 2, pp. 232–236, 2010.

[18] R. Matthews, "On the derivation of a "chaotic" encryption algorithm," *Cryptologia*, vol. 13, no. 1, pp. 29–42, 1989.

[19] M. Khan and F. Masood, "A novel chaotic image encryption technique based on multiple discrete dynamical maps," *Multimedia Tools and Applications*, vol. 78, Article ID 26203, 2019.

[20] R. A. Thiétart and B. Forgues, "Chaos theory and organization," *Organization Science*, vol. 6, pp. 19–31, 1995.

[21] C. M. Reigeluth, "Chaos theory and the sciences of complexity: foundations for transforming education," in *Annual Meeting of the American Educational Research Association,* San Diego, CA, 2004.

[22] C. E. Shannon, "Communication theory of secrecy systems," *Bell system technical journal*, vol. 28, pp. 656–715, 1949.

[23] S. Bone and M. Castro, *A Brief History of Quantum Computing*, Imperial College in London, London, UK, 1997.

[24] F. Belkhouche and U. Qidwai, "Binary image encoding using one-dimensional chaotic map," *IEEE Annual Technical Conference IEEE Region*, vol. 5, pp. 39–43, 2003.

[25] G. Chen, Y. Mao, and C. K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps," *Chaos, Solitons & Fractals*, vol. 21, no. 3, pp. 749–761, 2004.

[26] Y. Lu, W. Meier, and S. Vaudenay, "The conditional correlation attack: a practical attack on bluetooth encryption," in *Annual International Cryptology Conference*, pp. 97–117, Springer, Berlin, Heidelberg, 2005.

[27] L. Ballard, M. Green, B. De Medeiros, and F. Monrose, *Correlation-Resistant Storage via Keyword-Searchable Encryption*, p. 417, IACR Cryptology ePrint Archive, 2005.

[28] S. Mazloom and A. M. Eftekhari-Moghadam, "Color image encryption based on coupled nonlinear chaotic map," *Chaos, Solitons & Fractals*, vol. 42, no. 3, pp. 1745–1754, 2009.

[29] S. Mohammad Seyedzadeh and S. Mirzakuchaki, "A fast color image encryption algorithm based on coupled two-dimensional piecewise chaotic map," *Signal Processing*, vol. 92, no. 5, pp. 1202–1215, 2012.

[30] S. Liu, J. Sun, and Z. Xu, "An improved image encryption algorithm based on chaotic system," *Journal of Computers*, vol. 4, pp. 1091–1100, 2009.

[31] A. Akhshani, A. Akhavan, S. C. Lim, and Z. Hassan, "An image encryption scheme based on quantum logistic map," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, pp. 4653–4661, 2012.

[32] C. Zhu and K. Sun, "Cryptanalyzing and improving a novel color image encryption algorithm using RT-enhanced chaotic tent maps," *IEEE Access*, vol. 6, Article ID 18759, 2018.

[33] A. A. Abd El-Latif, L. Li, N. Wang, Q. Han, and X. Niu, "A new approach to chaotic image encryption based on quantum chaotic system, exploiting color spaces," *Signal Processing*, vol. 93, pp. 2986–3000, 2013.

[34] X. Wang and L. Yang, "A novel chaotic image encryption algorithm based on water wave motion and water drop diffusion models," *Optics Communications*, vol. 285, pp. 4033–4042, 2012.

[35] Y. Wu, Y. Zhou, J. P. Noonan, and S. Agaian, "Design of image cipher using Latin squares," *Information Sciences*, vol. 264, pp. 317–339, 2014.

[36] B. Norouzi, S. Mirzakuchaki, S. M. Seyedzadeh, and M. R. Mosavi, "A simple, sensitive and secure image encryption algorithm based on hyper-chaotic system with only one round diffusion process," *Multimedia Tools and Applications*, vol. 71, no. 3, pp. 1469–1497, 2014.

[37] S. M. Seyedzadeh, B. Norouzi, M. R. Mosavi, and S. Mirzakuchaki, "A novel color image encryption algorithm based on spatial permutation and quantum chaotic map," *Nonlinear Dynamics*, vol. 81, no. 1-2, pp. 511–529, 2015.

[38] R. Hamza and F. Titouna, "A novel sensitive image encryption algorithm based on the Zaslavsky chaotic map," *Information Security Journal: A Global Perspective*, vol. 25, no. 4-6, pp. 162–179, 2016.

[39] R. E. Boriga, A. C. Dascalescu, and A. V. Diaconu, "A new fast image encryption scheme based on 2D chaotic maps," *IAENG International Journal of Computer Science*, vol. 41, no. 4, pp. 249–258, 2014.

[40] G. Zhang and Q. Liu, "A novel image encryption method based on total shuffling scheme," *Optics Communications*, vol. 284, pp. 2775–2780, 2011.

[41] J. S. Khan, W. Boulila, J. Ahmad et al., "DNA and plaintext dependent chaotic visual selective image encryption," *IEEE Access*, vol. 8, Article ID 159732, 2020.

[42] J. S. Khan and J. Ahmad, "Chaos based efficient selective image encryption," *Multidimensional Systems and Signal Processing*, vol. 30, no. 2, pp. 943–961, 2019.

[43] Y. Mao, G. Chen, and S. Lian, "A novel fast image encryption scheme based on the three-dimensional chaotic baker Map," *International Journal of Bifurcation and Chaos*, vol. 14, pp. 3613–3624, 2004.

[44] F. Sun, S. Liu, Z. Li, and Z. Lu, "A novel image encryption scheme based on spatial chaos map," *Chaos, Solitons & Fractals*, vol. 38, no. 3, pp. 631–640, 2008.

[45] J. C. Yen and J. I. Guo, "A New chaotic key-based design for image encryption and decryption," *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, pp. 49–52, 2000.

[46] J. S. Khan, J. Ahmad, S. F. Abbasi, and S. K. Kayhan, "DNA Sequence Based Medical Image Encryption Scheme," in *Proceedings of the 2018 10th IEEE Computer Science and Electronic Engineering (CEEC)*, pp. 24–29, IEEE, Colchester, UK, September 2018.

[47] A. Belazi, A. A. Abd El-Latif, and S. Belghith, "A novel image encryption scheme based on substitution-permutation network and chaos," *Signal Processing*, vol. 128, pp. 155–170, 2016.

[48] X. Huang and G. Ye, "An image encryption algorithm based on hyper-chaos and DNA sequence," *Multimedia Tools and Applications*, vol. 72, no. 1, pp. 57–70, 2014.

[49] J. Ahmad, A. Tahir, J. S. Khan, M. A. Khan, F. A. Khan, and Z. Habib, "A Partial Ligt-Weight Image Encryption Scheme," in *Proceedings of the 2019 IEEE UK/China Emerging Technologies (UCET)*, pp. 1–3, Glasgow, UK, August 2019.

[50] B. A. Forouzan, *Cryptography & Network Security*, McGraw-Hill, New York, NY, USA, 2007.

[51] Y. Dodis, M. Stam, J. Steinberger, and T. Liu, "Indifferentiability of confusion-diffusion networks," in *Proceedings of the Annual International Conference on the Theory and*

Applications of Cryptographic Techniques, pp. 679–704, Springer, Berlin, Heidelberg, September 2016.

[52] J. x. Chen, Zl Zhu, C. Fu, Lb Zhang, and Y. Zhang, "An efficient image encryption scheme using lookup table-based confusion and diffusion," *Nonlinear Dynamics*, vol. 81, no. 3, pp. 1151–1166, 2015.

[53] J. Masood, M. Shahzad, Z. A. Khan et al., "Effective Classification Algorithms and Feature Selection for Bio-Medical Data Using IoT," in *Proceedings of the 2020 Seventh International Conference on Information Technology Trends (ITT)*, pp. 42–47, Abu Dhabi, UAE, November 2020.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 2959222, 31 pages
https://doi.org/10.1155/2022/2959222

Review Article

Deep Learning Methods for Malware and Intrusion Detection: A Systematic Literature Review

Rahman Ali,[1] Asmat Ali,[2] Farkhund Iqbal,[3] Mohammed Hussain,[4] and Farhan Ullah[5]

[1]*QACC, University of Peshawar, Peshawar, Pakistan*
[2]*Department of Computer Science, University of Peshawar, Peshawar, Pakistan*
[3]*College of Technological Innovation, Zayed University, Dubai, UAE*
[4]*College of Technological Innovation, Zayed University, Dubai, UAE*
[5]*School of Software, Northwestern Polytechnical University, 127 West Youyi Road, Beilin District, Xi'an 710072, China*

Correspondence should be addressed to Farhan Ullah; farhan@nwpu.edu.cn

Received 4 August 2021; Revised 10 August 2022; Accepted 12 September 2022; Published 10 October 2022

Academic Editor: Andrea Michienzi

Android and Windows are the predominant operating systems used in mobile environment and personal computers and it is expected that their use will rise during the next decade. Malware is one of the main threats faced by these platforms as well as Internet of Things (IoT) environment and the web. With time, these threats are becoming more and more sophisticated and detecting them using traditional machine learning techniques is a hard task. Several research studies have shown that deep learning methods achieve better accuracy comparatively and can learn to efficiently detect and classify new malware samples. In this paper, we present a systematic literature review of the recent studies that focused on intrusion and malware detection and their classification in various environments using deep learning techniques. We searched five well-known digital libraries and collected a total of 107 papers that were published in scholarly journals or preprints. We carefully read the selected literature and critically analyze it to find out which types of threats and what platform the researchers are targeting and how accurately the deep learning-based systems can detect new security threats. This survey will have a positive impact on the learning capabilities of beginners who are interested in starting their research in the area of malware detection using deep learning methods. From the detailed critical analysis, it is identified that CNN, LSTM, DBN, and autoencoders are the most frequently used deep learning methods that have effectively been used in various application scenarios.

1. Introduction

Deep learning (DL) is a representation learning approach having multiple levels of representation, each of which transforms the representation at one level to a representation at a higher level allowing very complex functions to be learned. Deep learning methods may help in solving the problems that have been a challenge for the machine learning community. It has been found very useful in discovering complex structures in high-dimensional data and is applied in various domains of government, business, and science. Representation learning methods allow us to feed a machine with raw data and discover the representation we need for classification or detection [1, 2]. Machine learning is used for enhancing the functionality of computer systems and data processing systems in various fields, like medicine, research, robotics, web search engines, content filtering, and recommendation systems on social networks and e-commerce platforms, and in products, like cameras and smartphones. However, traditional machine learning techniques have certain limitations in processing data in raw form [2].

Malware is a piece of code written with the intent to cause harm to, or subvert, the functionality of a computer system. It is a general term that describes viruses, spyware, Trojans, adware, and other types of malicious code [1]. A malware detecting system is a system that tries to determine whether or

not a specific program or piece of code is malicious [1, 3]. Malware developers use obfuscation methods to change the form of malware and deceive virus scanners that use pattern matching techniques because they are purely semantic and ignore the instructions' semantics and the pattern matching techniques are not resilient to variations [3]. Malware designed today are polymorphic and metamorphic, having the ability to change their code with propagation. Malware variants share various behavioral patterns that could be exploited to detect unknown malware using machine learning techniques that could not be achieved by using the traditional malware detection techniques [4]. Deep learning techniques have been widely used in various fields, including computer vision, speech recognition, pattern recognition, NLP, and malware detection and classification, and have become an active area of research. It is a challenging task to develop systems that can detect any kind of malicious code accurately in a short time. Due to the increasing number and variants of malware, a malware detection system should automatically perform without or with minimal human intervention. Moreover, signature-based malware detection techniques are not sufficient to fight against malware as they could be easily deceived in an intelligent manner [5]. Using deep learning methods for malware detection and classification enables us to build scalable models that may handle any measure of data and improve their accuracy as they can identify more features than traditional machine learning methods. After the training phase, DL models can acquire a new pattern of malware easily [6].

From the extensive literature survey of the deep learning methods in the area of malware and intrusion detection, one can understand that researchers have worked in this direction; however, the majority of studies focus on deep learning-based methods or they are related to a specific type of malware (e.g., android malware detection or Windows-based malware detection or network anomaly detection, etc.). Very few surveys can be found, such as [7–9], discussing the subject area; however, they reviewed a very limited number of research works and techniques.

A separate study for each type of malware could be conducted, such as Windows-based malware, Android-based malware, intrusion, network anomalies, and other threats to security. Moreover, it is also a good idea to conduct an SLR for different types of threats, such as ransomware; however, we feel the need to present a broad picture and provide a broad-scoped vision to the new researchers in the area, separating the different platforms by discussing them in different sections.

A large amount of data is being produced online and in various organizations, which makes this era the era of big data. Traditional data processing and traditional machine learning methods may not be able to process this data and extract insights from it. Deep learning is thought to be more capable of processing large volumes of data and enable the researchers to reach better solutions. During the recent years, researchers have been focusing on deep learning methods for malware and threats detection and classification. Keeping in view the effectiveness and importance of deep learning techniques and the recent trends in research, we conducted a systematic literature review (SLR) of DL techniques used in malware and intrusion detection systems in the last six years, 2015 to 2022.

The aim of this study is to identify the scope, trends, and methods of deep learning algorithms exploited in the area of malware detection systems for various platforms and develop a better understanding of the new researchers in this area. To achieve the said aim, the following objectives are set as research questions.

(i) RQ1: Which platforms are affected the most by various types of malware attacks?

(ii) RQ2: What are different malware analytics (hot era of the research for malware detection, most vulnerable platform and most popular journals publishing malware detection literature)?

(iii) RQ3: Which methods are used by the research community for malware detection?

(iv) RQ4: What are the major DL algorithms used in the domain of malware detection?

(v) RQ5: What are the main challenges that we face in using deep learning methods for malware and intrusion detection?

(vi) RQ6: Do the proposed android malware detection systems, approaches, and studies support the characteristics of sustainability, evolvability, and automatically picking the most appropriate malware detection DL algorithm?

(vii) RQ7: Do the Android-based surveyed methods have the ability of identifying any newly introduced malware and what type of feature analysis technique (s) the researchers have used in their research?

(viii) RQ8: What are the most commonly used datasets used for malware detection in the Android-based and Windows-based environments?

The rest of the paper is structured as follows. In Section 2, a summary of the surveys done in the subject area is made. In Section 3, the research methodology is elaborated from different perspectives. In this section, we summarize the selected studies and present a complete picture of how DL methods are used for malware detection in Windows and Android platforms. In Section 4, experiments are performed to show how DL methods supersede traditional ML methods. Section 5 critically discusses different aspects of the study. Section 6 recommends key points to the readers and those who are new to the area. Finally, Section 7 provides a conclusion of the study.

2. Existing Surveys on Deep Learning Methods for Malware Detection

There have been a number of relevant studies in the domain of malware and intrusion detection systems using machine learning, deep learning, or other methods. Some of these studies are summarized in Table 1.

Table 1: State-of-the-art research in deep learning for malware detection.

Ref.	Year	Description	Limitations
[10]	2019	Survey of approaches that detect permissions demanded by apps that might be used for malicious activities	(i) Limited to android malware detection (ii) Permission-based malware detection only
[11]	2017	Deep learning techniques to detect network anomalies	(i) Only network anomaly and intrusion detection systems
[12]	2019	Survey of approaches to network anomaly detection	(i) Only network anomaly detection (ii) Traditional learning based
[13]	2018	Different malware detection techniques, like signature- and behavior-based detection	(i) Not limited to deep learning
[14]	2018	A survey of intrusion detection techniques in vehicular ad hoc networks	(i) Limited to intrusion detection systems in vehicular networks
[6]	2018	A survey of android malware analysis using deep learning with static analysis, dynamic analysis, and hybrid analysis	(i) Limited to android malware detection (ii) Reviewed very few publications
[15]	2019	A survey of machine learning techniques used in cyber security, like spam detection, phishing detection, and malware detection	(i) Traditional learning-based systems (ii) Limited to cyber security
[7]	2019	Review of deep learning-based android malware detection techniques	(i) Limited to android malware detection (ii) Reviewed very few publications
[16]	2019	A review of different intrusion detection systems in IoT, including anomaly based, specification based, signature based, and hybrid IDS's	(i) IoT-based systems only (ii) Not limited to DL- and ML-based methods
[17]	2018	A survey of IDS's and defense systems in IoT	(i) IoT-based systems only (ii) Not limited to DL and ML
[5]	2018	A survey of applications of deep learning techniques in malware detection	(i) Reviewed only the different DL algorithms used for malware detection
[18]	2020	A survey of deep learning techniques in defense against phishing	(i) Limited to phishing attacks only
[19]	2019	A survey of android malware detection systems	(i) Limited to Android malware (ii) Not DL or ML based
[20]	2019	A survey of techniques for security in IoT	(i) Only IoT-based systems (ii) Not limited to DL
[21]	2019	A survey of intrusion detection systems using deep learning	(i) Only intrusion detection systems
[8]	2020	A review of malware detection systems using deep learning	(i) Reviewed only about 35 publications
[22]	2020	An extensive review of malware detection systems based on deep learning	(i) Limited to Android malware
[9]	2020	A review of DL algorithms used for malware detection and some relevant literature	(i) Reviewed very few publications
Proposed study		Deep learning methods for malware and intrusion detection—a systematic literature review	Key contributions (i) One of the extensive surveys covering a large number of research articles (94) in Windows-, Android-, and IoT-based environments for malware and intrusion detection using deep learning approaches. (ii) Extraction of deeper malware analytics (iii) Extraction of useful information about deep learning methods applied in the domain of malware and intrusion detection (iv) Identification of the most effective deep learning algorithms for malware and intrusion detection (v) Highlighting the key challenges faced during the use of deep learning methods for malware and intrusion detection

It is evident from the last column of Table 1 that these surveys are related to malware or intrusion detection systems; however, most of them are not deep learning-based or related to a specific type of malware (e.g., android malware detection or network anomaly detection). Very few surveys were found that reviewed malware detection systems using deep learning, such as [7–9]; however, they reviewed a very limited number of research works and techniques. In the proposed study, we have tried our level best to cover the use of deep learning algorithms applied to Windows-based, Android-based, and IoT-based environments for the detection and classification of deep learning algorithms. This will provide a base to the beginners in the area to start their research work and easily find the gap for further improvements and start of new research.

Key contributions of the survey are enlisted as follows:

(i) One of the extensive surveys covering a large number of research articles (94) in Windows-, Android-, and IoT-based environments for malware and intrusion detection using deep learning approaches

(ii) Extraction and formulation of malware analytics from the relevant literature on DL methods used for malware and intrusion detection

(iii) An extensive study of the relevant literature to extract useful information about deep learning methods used in the domain of malware and intrusion detection systems

(iv) An analysis of which deep learning algorithms are used in malware and intrusion detection systems

(v) Highlighting the key challenges faced by the research community in using deep learning methods for malware and intrusion detection

3. Research Methodology

To assess the applications and impact of deep learning methods for malware and intrusion detection systems, a systematic literature review (SLR) is conducted. In an SRL, a systematic approach is followed to identify and analyze relevant studies regarding a specific area of interest [23]. There are a number of guidelines defined by experts for conducting a comprehensive and effective SLR. We studied and followed the guidelines provided in [24, 25].

3.1. Research Design. Firstly, we do need an assessment to conduct the literature review. We studied a number of papers from various sources, including journals and conferences. We observed that deep learning has been used by a large number of researchers in malware detection, intrusion detection, and other security-related systems. Based on the need assessment, we formulated a number of research questions, mentioned in the introduction section, to show the effectiveness and outcomes of the proposed study.

We then selected a number of search libraries to make sure the selected literature is from an authentic source and is reliable. We selected five different libraries, including Google Scholar, Science Direct, IEEE Explore, ACM, and Springer Link. Next, we formulated a search query to retrieve only the relevant studies focusing on deep learning for malware and intrusion detection. We identified keywords, based on the research questions that represent and reflect the RQs to objectively search the relevant publications. It is not effective to search the libraries using individual words; instead, the search keywords are combined by using "AND" and "OR" operators to generate queries that may return only relevant results. The search query formulated was finally fed into the searching mechanism of each library to retrieve the relevant studies from the last six years, 2015 to 2022. We searched these libraries for the previous years as well; however, we found very few, or no relevant studies during these years; thus, we limited our search to the recent years only. Moreover, we intend to focus on the latest state-of-the-art research in our study. We selected the primary studies by analyzing the title and abstract of the publications. We created an EndNote library and downloaded all the selected papers which were then read in full to exclude the papers which were not relevant to the theme of the current study and generated a final set of the relevant studies. This set of publications was then studied to answer the research questions (RQs) and to achieve the objectives of the study.

A tracer list for the findings and insights of each RQ is given below to easily guide the readers towards the respective section of the paper for better comprehension. Section 3.2.1 describes different platforms and extracts the insights regarding the severely affected platform so that to find answer to RQ1. Section 3.2.1 extracts different malware analytics and tries to find answer to RQ2. Column 3 of Tables 2–6 of Sections 4–8, respectively, extracts insights regarding the DL-based malware detection methods and tries to find answer to RQ3, insights for RQ4, which are the major DL algorithms used so far and have been discussed in Section 9. The key research challenges faced during implementation of DL-based malware detection systems are focused in RQ5 and described in Section 11.3. RQ6 and RQ7 describe the issues of sustainability and evolvability which are also discussed in Section 11.3 and RQ8 extracts insights of the most commonly used datasets for malware detection in Section 11.4.

3.1.1. Criteria for the Survey. As deep learning-based security systems and other intelligent software tools are getting more and more popular in the field of computer science, it is important to conduct research about how deep learning technology is performing better than the traditional approach for threat hunting and traditional machine learning-based systems. In this survey, we are interested in analyzing which deep learning algorithms are used in malware detection systems and how they can perform better when compared with traditional machine learning-based systems. We carefully read the selected literature to find out which types of threats and what platforms the researchers are targeting and how accurately the deep learning-based systems can detect new security threats when trained with a training dataset. Moreover, in the case of Android-based malware detection research, the focus is also given to know whether the proposed methods support the characteristics of sustainability, evolvability, automatically picking the most appropriate malware detection DL algorithm, ability of identifying new malware (s), and diverse feature analysis methods, such as static, dynamic, or hybrid. Apart from the above, the survey also discusses the issue of data quality for deep learning-based methods in the form of publicly available malware datasets for research purpose.

3.1.2. Query Formulation. As we discussed earlier, it is not the correct way to search each library using a single keyword each time as it is a very tedious and lengthy process. We need to formulate a search query containing all the important keywords including names, synonyms, and abbreviations,

Table 2: Summary of the metadata extracted from the literature on Windows-based malware detection.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Library/ framework used	Platform	Dataset used	Accuracy/F_1 score
[26]	Malware classification by extracting static features and converting to gray images	CNN	Not stated	Windows	Kaggle by Microsoft	98.86%
[27]	Malware classification by converting malware binary file to gray image through code mapping, texture partitioning, and texture extraction	CNN	Not stated	Windows	BIG 2015	99%
[28]	Malware classification by extracting series of system calls having malicious behavior	Not stated	Not stated	Windows	Self-generated	95.6%
[29]	Malware detection and classification by using the op-code and API calls data of malware and benign-ware	CNN, BPNN	Not stated	Windows	Self-generated	95%
[30]	Multilevel deep learning system for malware detection using different static and dynamic features	Proposed MLDLS	Not stated	Windows	Self-generated	Not stated
[31]	Ransomware detection system based on *n*-gram op-code with deep learning	CNN	Keras, TensorFlow	Windows	Self-generated	89.5%
[32]	Malware detection by transforming PE file to op-code sequences and representing the op-code as *n*-gram vector	DBN	Not stated	Windows	Self-generated	About 98%
[33]	Malware detection by visualizing the malware binary file as gray image	CNN	MatConvNet in MATLAB	Windows	Self-generated	—
[34]	Malware detection using API calls of Windows' executable files	DAE, RBM	Not stated	Windows	Comodo Cloud Security Center's dataset	Around 98%
[35]	Malware detection based on API calls sequence and statistical features	LSTM, RNN	TensorFlow	Windows	Self-generated	95.7%
[36]	Identifying executable files as malware or benign using static and dynamic analysis and categorizing the malware to the corresponding family	CNN, LSTM	TensorFlow, Keras	Windows	Malimg, EMBER, self-generated	98.8%
[37]	Hybrid image-based technique for malware detection by converting malware binaries to gray images	CNN, LSTM	TensorFlow, Keras	Windows	BIG 2015	96–97%
[38]	Malware detection by extracting API call sequences of malware using dynamic analysis and generating feature images	CNN	Not stated	Windows	VirusShare dataset	About 99%
[39]	Predicting malicious behavior of executable program based on small amount of behavioral data within the first few seconds of execution	RNN	Keras, scikit-learn	Windows	Self-generated	96%
[40]	DLMD: malware detection technique based on static features using byte and ASM files	CNN	PyTorch	Windows	BIG 2015	97.5%
[41]	Malware detection extracting control flow graph of the sample by lazy binding and transforming it into an image	CNN	Not stated	Windows	MALICIA, VirusShare, VXHeaven	92%–97.7%
[42]	Deep learning system with two hidden layers for malware detection using dependency of malware sequence and avoiding back-propagation	TELM	Not stated	Windows	Kaggle, VXHeaven	Above 99%
[43]	Malware classification by transforming malware binary file to grayscale images	CNN, LSTM	TensorFlow, Keras	Windows	BIG 2015	98.2%
[44]	Zero-day malware detection by generating fake malware and learning to distinguish it from the real malware	DAE, DCGAN	Keras	Windows	Kaggle	About 99%
[45]	Malware detection by visualizing the malware as grayscale image	CNN	TensorFlow	Windows	Malimg, Microsoft dataset	99.97%

Table 2: Continued.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Library/ framework used	Platform	Dataset used	Accuracy/F_1 score
[46]	Detecting threats in the cloud-assisted Internet of things by extracting API calls data from malware	DBN	Not stated	Windows	VXHeaven	Up to 99.78%
[47]	Malware classification by visualizing the malware as grayscale image	CNN	Not stated	Windows	Malimg, BIG 2015	97.5%
[48]	Malware variants detection by visualizing malware samples as grayscale images	CNN	Caffe NN framework	Windows	Dataset by Vision Research Lab	94.5%
[49]	Malware detection by converting malware executable to grayscale image and using NSGA-II algorithm to deal with data imbalance	CNN	TensorFlow	Windows	Dataset by Vision Research Lab	97.6%
[50]	Malware detection by visualizing the malware sample as a grayscale image	Deep transfer learning	Not stated	Windows	Not stated	99.25%
[51]	Malware detection by using static analysis to extract features of the malware samples	LSTM	Keras, TensorFlow	Windows	Self-generated dataset named MC-dataset-multiclass	90.63%
[52]	Malware detection by visualizing the malware sample as a grayscale image	CNN	TensorFlow	Windows	Malimg	80.5%
[53]	Malware detection by extracting features, like file activity, registry activity, service activity, processes, runtime DLLs and network activities, etc., and applying big data analytics techniques	Not stated	Keras	Windows	Self-generated	97%
[54]	Malware classification by converting malware binaries to Markov images	CNN	Keras, TensorFlow	Windows	Microsoft dataset, Drebin dataset	97.3% for Drebin, 99.3% for Microsoft
[55]	A comparative study of CNN and ELM-based detection systems using malware represented as grayscale images	CNN	Keras	Windows	Malimg	96.3% for CNN, 97.7% for ELM
[56]	Metamorphic malware detection using API calls made on the operating system	LSTM	Keras	Windows	Self-generated API sequence dataset	Up to 98.5%
[57]	Malware detection by extracting features of PE files, including import functions feature, general information feature, and bytes entropy feature	Not stated	Not stated	Windows	Self-generated	AUC up to 0.989
[58]	Cryptomining malware detection by static and dynamic analysis of the op-code sequences of PE files	CNN, LSTM, ATT-LSTM	Not stated	Windows	Self-generated	97% on average
[59]	Malware classification using malware samples represented as grayscale images	CNN	Keras, TensorFlow	Windows	Malimg	99.72%
[60]	Malware classification by extracting features including API calls, sequence of assembly language instructions, and malware's binary contents	CNN	TensorFlow	Windows	Kaggle	99.7%
[61]	Image-based malware classification system using an ensemble of CNN	CNN	TensorFlow, Keras, scikit-learn	Windows	Malimg	99.5%
[62]	Malware detection by black-and-white embedding of malware images rather than grayscale to avoid bit loss in byte	CNN	Keras, TensorFlow	Windows	KISA dataset	92.8%
[63]	Malware classification by generating a low-dimensional vector and using op-codes and API function calls to train model	Bi-LSTM	Not stated	Windows	Microsoft dataset	96.8%
[64]	Malware detection by extracting the API calls sequence and generating the API pixel vector and finally visualizing the malware	CNN	Not stated	Windows	Self-generated	94.7%

Table 3: Summary of the metadata extracted from the literature on android-based malware detection.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Automatic DL algorithm selection (yes/no)	Ability of identifying new malware? (yes/no)	Features engineering method (static/dynamic/hybrid analysis)	DL model needs updating? (yes/no)	Sustainability/resilience against evolution? (yes/no)	Library/framework used	Platform	Dataset used	Accuracy/F_1 score
[65]	Malware detection using neural networks and k-means clustering	Not stated	No	No	Static analysis	Yes	No	Not stated	Android	Self-generated	88.0%
[66]	Malware detection based on API method calls sequence mining	CNN	No	No	Static analysis	Yes	No	TensorFlow	Android	Malgenome, Drebin, MalDozer	About 99%
[67]	Malware detection by analyzing the permission wanted by app	Deep eigenspace learning	No	No	Static analysis	Yes	No	Not stated	Android	Self-generated	Not stated
[68]	Malware detection by extracting and analyzing several features	Multimodal neural networks	No	No	Static analysis	Yes	No	Keras, TensorFlow, scikit-learn	Android	VirusShare, malgenome	94 – 98%
[69]	Dynamic malware detection system based on CPU, memory, and battery usage	LSTM RNN, encoder-decoder	No	No	Static analysis	Yes	No	Not stated	Android	M0Droid	About 80%
[70]	Malware detection by associating the features from static analysis with the features from dynamic analysis	DBN	No	No	Hybrid analysis	Yes	No	Not stated	Android	Self-generated, malgenome	96.76%
[71]	Malware detection using several static and dynamic features	DBN	No	No	Hybrid analysis	Yes	No	Not stated	Android	Self-generated	96.5%
[72]	Malware detection by using the importance of words from the apk file of applications	CNN	No	No	Static + renaming variables and prioritizing	Yes	No	Not stated	Android	Self-generated	92.67%
[73]	Malware detection by extracting several features for model training	CNN	No	No	Static analysis for static features	Yes	No	Keras	Android	Self-generated	99.25%
[74]	Malware detection using seven different features of android applications	DAE, CNN	No	No	Static analysis for 7 categories of static features	Yes	No	Keras, TensorFlow, scikit-learn	Android	Self-generated	99.82%

Table 3: Continued.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Automatic DL algorithm selection (yes/no)	Ability of identifying new malware? (yes/no)	Features engineering method (static/dynamic/hybrid analysis)	DL model needs updating? (yes/no)	Sustainability/resilience against evolution? (yes/no)	Library/framework used	Platform	Dataset used	Accuracy/F_1 score
[75]	ITMF, (image texture median filter) for analyzing and detecting malware on Derbin dataset	DBN	No	Potential of dynamic activity of malware	Static analysis	Yes	No	Keras, TensorFlow, scikit-learn	Android	Drebin	95.43%
[76]	Malware detection using static analysis, dynamic analysis, and system calls	DBN	No	No	Hybrid analysis	Yes	No	Not stated	Android	Not stated	99.1%
[77]	Malware detection by extracting the API calls graph of applications and generating graph embedding	CNN, RNN	No	No	Pseudodynamic analysis	Yes	No	Keras, TensorFlow	Android	AMD dataset, AndroZoo, Drebin, ISCX	98.86%
[78]	Malware detection by examining all execution paths and detecting malicious and benign paths	LSTM RNN	No	No	Pseudodynamic analysis	Yes	No	TensorFlow, Keras, scikit-learn	Android	AndroZoo,	91.42%
[79]	Malware detection using features extracted from manifest file and through static analysis and various deep learning methods	CNN, DBN, LSTM, DAE	No	No	Static analysis	Yes	No	TensorFlow, Keras, theano	Android	Drebin, VirusShare	Up to 93.6%
[80]	Malware detection by extracting features through dynamic analysis and generating Markov chains	RNN, CNN, LSTM	No	No	Dynamic analysis	Yes	No	TensorFlow, Keras, scikit-learn	Android	Drebin	Around 81%
[81]	Malware detection by extracting texture fingerprint features and mapping malicious code to grayscale image	DBN	No	No	Static analysis	Yes	No	Theano, GDBN, TensorFlow, Keras, scikit-learn	Android	Drebin	95.9%

Table 3: Continued.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Automatic DL algorithm selection (yes/no)	Ability of identifying new malware? (yes/no)	Features engineering method (static/dynamic/hybrid analysis)	DL model needs updating? (yes/no)	Sustainability/resilience against evolution? (yes/no)	Library/framework used	Platform	Dataset used	Accuracy/F_1 score
[82]	Malware detection by using various features, including hardware components, permissions, filtered intents, API calls, and network addresses	LSTM, CNN	No	No	Static analysis	Yes	No	Not stated	Android	Drebin	Up to 98.53%
[83]	Malware detection using LASSO feature shrinkage and selection technique and deep belief networks	DBN	No	No	Static analysis	Yes	No	Not stated	Android	DroidWare	85.22%
[84]	Malware detection by generating API images from the sequence of API calls of applications	DAE	No	No	Dynamic analysis	Yes	No	Not stated	Android	Malgenome, contagio minidump	98%
[85]	Malware detection by using features like permissions and API calls and generating a feature vector	DBN	No	No	Static analysis	Yes	No	TensorFlow	Android	Drebin, VirusTotal, contagio, self-generated	99.04%
[86]	Malware detection by extracting byte code from the.dex file of the android package	DAE, DBN, RNN, LSTM, BiLSTM,	No	No	Static analysis	Yes	No	Keras, theano	Android	Drebin, AMD, VirusShare	Up to 99.9%
[87]	Malware detection by extracting features like API calls, events and permissions by dynamic analysis	Not stated	No	No	Dynamic analysis	Yes	No	Not stated	Android	Self-generated	98.5%
[88]	Malware detection by converting static features like permissions API calls and components into a set of semantic features	GCN	No	No	Static analysis	Yes	No	Not stated	Android	Drebin, AMD, lab-built, AndroZoo, PRAGuard	Up to 99.7%

Table 3: Continued.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Automatic DL algorithm selection (yes/no)	Ability of identifying new malware? (yes/no)	Features engineering method (static/dynamic/hybrid analysis)	DL model needs updating? (yes/no)	Sustainability/resilience against evolution? (yes/no)	Library/framework used	Platform	Dataset used	Accuracy/F_1 score
[89]	Hybrid deep learning for android malware detection using various static and dynamic feature of the application	DBN	No	No	Hybrid analysis	Yes	No	TensorFlow, Keras	Android	Self-generated	96.8%
[90]	Malware detection by using dataset comprising of intent features and permission features extracted from benign and malicious applications	Not stated	No	No	Hybrid analysis	Yes	No	Not stated	Android	Omnidroid	91%
[91]	Malware detection by converting the application binary to gray-scale image	Not stated	No	No	Static analysis	Yes	No	Not stated	Android, iOS	AMD, self-generated	96.6% for Android, 95.8% for iOS
[92]	Malware detection by extracting byte code of the application and generating embedding	LSTM	No	No	Hybrid analysis	Yes	No	Not stated	Android, IoT	Self-generated	98% for Android, 99% for IoT malware
[93]	Malware detection by extracting 11 static behavioral features and transforming them to a multidimensional vector	DBN	No	No	Static analysis	Yes	No	TensorFlow	Android	Self-generated, Drebin, etc.	Up to 99.5%
[94]	Malware detection by extracting API sequence and the methods from the DEX file of the application and generating the hot vector of the API sequence	Bi-LSTM	No	No	Static analysis	Yes	No	Not stated	Android	AMD	97.2%

Table 3: Continued.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Automatic DL algorithm selection (yes/no)	Ability of identifying new malware? (yes/no)	Features engineering method (static/dynamic/hybrid analysis)	DL model needs updating? (yes/no)	Sustainability/resilience against evolution? (yes/no)	Library/framework used	Platform	Dataset used	Accuracy/F_1 score
[95]	Malware detection in the IoT devices by reading the DEX file of the application as an unsigned vector and converting it to a fixed size by image resampling technique	CNN, RNN, GRU, LSTM, Bi-LSTM	No	No	Static analysis	Yes	No	Keras	Android IoT devices	Self-generated	Up to 95.8%
[96]	Malware detection by generating the function call graph from the DEX file of the application and the op-code-level FCG features	LSTM	No	No	Static analysis	Yes	No	Keras, TensorFlow	Android	Self-generated	97%
[97]	Malware detection by extracting and vectorizing the manifest features and API calls from the binary file of the app	CNN, GRU, LSTM	No	No	Static analysis	Yes	No	Keras, TensorFlow	Android	Drebin, genome, contagio, pwnzen, VirusShare	96.8%
[98]	Malware detection by extracting features like permissions, system events, APIs and data flow from the manifest, DEX and layout xml files	MLP	No	No	Static analysis	Yes	No	TensorFlow	Android	Self-generated	94.9%
[99]	Malware detection by extracting static features (permissions) from the manifest file and then generating feature vector	CNN, DAE	No	No	Static analysis	Yes	No	Keras, TensorFlow	Android	CIC and Mal2017, self-generated	98.2%

Table 4: Summary of the metadata extracted from the literature on IoT-based malware detection.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Library/ framework used	Targeted platform	Dataset used	Accuracy/F_1 score
[100]	IoT and IoBT malware detection through op-code analysis	CNN	Not stated	IoT, IoBT	Self-generated	99.68%
[101]	Behavior-based deep learning framework for detecting malware in IoT environment	SAE	Keras	IoT	Self-generated	About 98.6%
[102]	Detecting pirated software and security threats in internet of things environment	CNN	TensorFlow, Keras	IoT	Not stated	96% for piracy detection, 97.46% for malware detection
[103]	Malware detection in the Internet of things environment by decompiling and extracting op-codes from application samples	LSTM	TensorFlow, Keras, scikit-learn	IoT	Self-generated	98.18%
[104]	Attack detection in industrial Internet of things environment	DAE, DFFNN	Not stated	IoT	NSL-KDD, UNSW-NB15	Up to 98.6%
[105]	Classification of malicious applications in the Internet of things environment by using graph embedding	CNN	Not stated	IoT	Self-generated	88.5%
[106]	Malware detection in industrial IoT devices by extracting DEX file and converting to java class file and extracting the bytecode from the class file	CNN	TensorFlow	IoT	Leopard mobile dataset	98.7%

which are connected by the logical operators "OR" and "AND." For example, different studies may use any of "Convolutional Neural Networks" and "CNN" in their title, so we need to combine these by the "OR" operator. Similarly, we need to have both "Deep Learning" and "Malware Detection" discussed in the abstract or the contents of the paper, so we need to combine these keywords by the "AND" operator. We included different keywords that may appear in the relevant studies, including Deep Learning, Deep Learning algorithms, and malware detection. The final search query we formulated is as follows.

("Deep Learning" OR "Convolutional neural network" OR "Deep belief network" OR "recurrent neural network" OR "CNN" OR "RNN" OR "DBN" OR "LSTM") AND ("malware") AND ("detection" OR "detect" OR "identification" OR "identify" OR "classification")

3.1.3. Searched Libraries. For searching the relevant studies, we select five popular libraries, as follows.

(i) Google Scholar: https://scholar.google.com/ (accessed 02-07-2022)

(ii) Science Direct: https://www.sciencedirect.com/ (accessed 02-07-2022)

(iii) IEEE Explore: https://ieeexplore.ieee.org/(accessed 03-07-2022)

(iv) ACM: https://dl.acm.org/(accessed 03-01-2021)

(v) Springer Link: https://link.springer.com/(accessed 03-07-2022)

The selected libraries were searched using the formulated search query, filtering the search results to include only the relevant papers published during 2015–2022. The search query returned a total of 935 publications. The detailed search results for each library are shown in Table 7.

3.1.4. Inclusion/Exclusion Criteria. Not all of the returned results could be relevant to the domain of the study. We need to further refine the results to get a final list of the most relevant publications. The inclusion and exclusion rules we applied are summarized in Table 8.

Our search query returned a total of 935 papers on all five libraries. Firstly, we excluded the papers that did not appear relevant based on the title or abstract, the survey papers, book chapters, and the gray literature. Table 7 summarizes the results after applying inclusion/exclusion criteria.

We downloaded the references of the 290 papers and created an endnote library. There, we further refined the results by removing duplicates, non-journal papers, papers written in a language other than English, and irrelevant papers based on full text. Table 9 summarizes the finally selected papers for critical analysis.

Figure 1 presents a summary of the research methodology used for research articles retrieval.

3.2. Metalevel Critical Analysis of the Literature. Once the relevant literature is retrieved and filtered, metalevel analysis is performed from different perspectives. In the first phase, to let the readers know of the importance of the proposed study, statistical analysis of the research work done so far is performed. In the second and third phase, respectively, Windows and Mobile platforms-wise metalevel analysis of the malware detection research is performed. These analyses are summarized in the subsequent sections.

Table 5: Summary of the metalevel analysis of the literature on malware detection in platforms other than Windows, Android, and IoT or multiple platforms.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Library/ framework used	Targeted platform	Dataset Used	Accuracy/$F1$ score
[107]	Malware dynamic behavior classification and family clustering algorithm	CNN, GAN	Not stated	Windows, android	DataCon, GreekPwn	Not stated
[108]	Detecting domain generation algorithms (DGAs) and automatically labelling domain names in real traffic	LSTM RNN	Keras, scikit-learn	Not stated	ALexaBamb, Retro	About 98%
[109]	Deep learning-based intrusion detection system for detecting cyber-attacks	Not stated	TensorFlow, Keras, scikit-learn	Internet	KDDCup99, NSL KDD, UNSW-NB15, WSN-DS, CICIDS 2017, Kyoto	85 – 99%
[110]	An intrusion detection system to protect in-vehicle network, the controller area network (CAN) bus	CNN	Not stated	—	Self-generated	99%
[111]	Source-based distributed denial-of-service defense system in fog and cloud computing systems	LSTM	Keras, TensorFlow	Cloud computing	Hogzilla	98.88%
[112]	A hybrid deep learning-based system for detecting botnet	CNN, RNN	Keras, TensorFlow, scikit-learn	Internet	CTU-13, ISOT	99.3% (CUT-13) 99.5% (ISOT)
[113]	Using robust software modeling tool (RSMT) to monitor and characterize the behavior of web based applications	SAE	Keras, TensorFlow, scikit-learn	Internet	Not stated	About 92%.
[114]	Deep multilayer perceptron and RNN-based deep learning system for detecting cloud-based intrusion	RNN, LSTM	Keras, TensorFlow, Theano	Cloud computing	Not stated	86.9%
[115]	Malware detection in PDF files	CNN	Not stated	Multiple	Self-generated	Up to 98.92%
[116]	Ransomware detection and classification by extracting event sequences during a program execution	LSTM, CNN	Keras, TensorFlow	Not stated	Self-generated	99.6%
[117]	Malware detection in cloud platforms by extracting several features of each process, like CPU usage, memory usage, and disk usage	CNN	Not stated	Cloud IaaS	Self-generated	Up to 93%
[118]	Using ML and DL techniques to distinguish normal traffic from cryptomining traffic by extracting the data flow features	Fully connected CNN	Keras, TensorFlow, scikit-learn	Internet	Self-generated mining traffic	99.98%
[119]	Malware detection on various platforms, including Windows, Android, IoT, IoBT, and the Internet by vector embedding	LSTM	Not stated	Windows, Android, IoT, IoBT, Internet	VXHeaven, Drebin, Kaggle	94.1% on average

3.2.1. Statistical Analysis. To answer to RQ2 and know about the peak era of research for malware detection using deep learning methods, the most appropriate place for publishing the work and the most affected platform out of the Windows and Mobile platforms, the literature is statistically summarized and presented in the following sections.

(1) Year-Wise Distribution of the Papers. We observed that, in recent years, there are a lot of research articles published discussing malware and intrusion detection systems using deep learning. We have only one paper in the years 2015 and 2016 while the number increases as we go upward. This shows that malware detection using deep learning algorithms is an active area of research in computer science. Figure 2 shows the year-wise distribution of the research papers published.

(2) Platform-Wise Distribution of the Papers (RQ1). As malware is not limited to a specific platform, different researchers have focused on different platforms for malware

Table 6: Summary of the metalevel analysis of the literature on malware detection during the recent years.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Library/ framework used	Targeted platform	Dataset used	Accuracy/F_1 score
[120]	Visualizing malware binaries as two-dimensional images and feeding to classifier that uses reweighted class-balanced loss function	Densely connected CNN with ReLU	Keras	Windows	Malimg, BIG 2015, MaleVis	98.46%
[121]	Two-stage hybrid malware detection by extracting op-code by static analysis and then performing dynamic analysis to classify benign files	Bi-LSTM, CNN	Not stated	IoT	KISA 2019	Up to 95%
[122]	Malware detection by representing the application as image, extracting the dex file, and grouping the sequence of bytes into grayscale pixel	CNN	CUDA, TensorFlow	Android	Argus Cyber Security Lab	97%
[123]	Malware detection by using text classification method, using the text sequence of APPs analysis and exploring information	CNN	Keras	Android	Various datasets	96.6%
[124]	Malware detection using dynamic analysis by generating dynamic analysis logs for an APK and transforming the features into a feature vector	CNN with leaky ReLU	Not stated	Android	Self-generated	98%
[64]	Malware detection by visualizing malware as RGB color images using both static and dynamic as well as hybrid analysis	CNN (VGG16)	Not stated	Windows	Dataset by VirusSign	94.7%
[125]	Detection of Java bytecode malware using static analysis of the Java program and extracting interprocedural control flow graph from bytecode file	CNN	Not stated	Platforms capable of running Java programs	Self-generated	98.4%
[126]	Analysis of behavior of malicious programs based on API call graphs. The detection is based on analyzed patterns of the API calls	CNN (used only for discovering common features)	Not stated	Android	Apps from playstore and VirusShare	93.2%
[127]	Classification and detection of malware using executable and linkable format (ELF) binary file, making use of static, dynamic, and hybrid analysis	Bi-GRU-CNN	Keras, TensorFlow, scikit-learn	IoT	Collected from various sources	98% (detect) 100% (classify)
[128]	Malware classification by converting the bytecode of methods of the malware into grayscale feature image and analyzing its feasibility based on reconstruction error of AE	AE based on CNN	TensorFlow	Android	Apps from playstore and VirusShare	96.2%
[129]	Distributed deep learning-based model for malware detection using both static and dynamic analysis	CNN-BiLSTM	Not stated	Windows	Apps from various sources	97%
[130]	Using DL and model-checking to detect malware by converting source code to format of the model-checker, using both static and dynamic analysis	CNN	PyTorch	IoT	Not stated	95%
[131]	Malware detection using static analysis, emphasizing on features extraction from PE files	Not stated	Keras	Windows	EMBER	97.5%

detection. To answer the RQ1 (which platforms are affected the most?), analytics have been calculated from the selected papers and the insights are summarized in Figure 3, stating that android and windows are the most widely used platforms on mobile devices and personal computers, respectively, and hence affected the most. Most of the researchers

Table 7: Summary of the literature extracted from the five scholarly libraries and the process of filtration.

Library	Library URL	Returned papers	SLR/book/gray literature	Irrelevant papers	Total
Google Scholar	https://www.scholar.google.com/	333	74	70	180
Science Direct	https://www.sciencedirect.com/	306	114	146	46
IEEE Explore	https://ieeeexplore.ieee.org/	144	12	85	47
ACM	https://dl.acm.org/	9	2	1	6
Springer	https://link.springer.com/	143	45	87	11
	Total	**935**	**247**	**398**	**290**

Table 8: The inclusion/exclusion criteria for the deep learning methods.

	Inclusion criteria
1	The paper must discuss a deep learning-based system for malware or intrusion detection.
2	The paper must be published in a scholarly journal or be a preprint.
3	The paper is published from January 2015 till 2022.
	Exclusion criteria
1	Papers focusing on economic, business, or legal impacts of malware detection and intrusion detection systems
2	Gray literature, such as blogs or reports
3	Papers written in a language other than English
4	Review papers
5	Duplicate papers
6	Papers not published in any scholarly journal, such as conference preceding
7	The studies that do not focus on deep learning for malware detection

Table 9: Summary of the selected literature for review.

Total papers	Nonjournal papers	Duplicates	Non-English	Full text exclusion	Final total
290	101	64	7	11	107

focused on these two platforms. However, the Internet of things environment and web-based malware received less attention despite the fact that malware developers target these platforms frequently. Web-based malicious programs are especially a big threat to the Internet and data security. Figure 3 shows the platform-wise distribution of the papers. Based on the findings of RQ1, Sections 4 and 5 have been given more attention, and hence in-depth critical analysis is performed.

(3) Journal-Wise Distribution of the Papers. We retrieved relevant publications from a large number of journals using the selected libraries. The number of papers from individual journals varies from 1 to 13, including arXiv preprints and journals from other publishers like Hindawi, IEEE, ACM, and more. Table 10 contains the journal-wise distribution data of the papers.

4. Windows Malware Detection

In the Windows platform, researchers have extensively worked on the subject matter to protect personal computers (PC) against cyber-attacks. In this section, we analyze the research work that focuses on Windows malware detection and present a summary in Table 2.

Ni et al. [26] proposed an algorithm "Malware Classification using SimHash and CNN" (MCSC) based on convolutional neural networks. They disassemble the code of the malware and convert it to gray images to identify its family. They apply locality-sensitive hashing (LSH) to convert similar malware code into similar hash values. These hash values are converted to gray images to train neural networks. They claim about 98% accuracy.

Zhao et al. [27] proposed MalDeep, a deep learning-based malware detection system that uses the binary file of the malware. They convert the binary file to gray image and use convolutional neural networks to classify the malware. The strength of their system has a high accuracy of over 99%.

Zhang et al. [28] proposed a deep learning system that uses sensitive system calls for malware detection. The application is monitored in Cuckoo sandbox to retrieve system calls data and train the neural networks. Their system achieves an accuracy of over 95%.

Zhang et al. [29] proposed a convolutional neural network-based malware detection system that includes unpacking the application to retrieve its op-codes and API calls, generating structured data to represent each binary and obtaining PCA-initialized op-code bi-gram matrix and PCA-initialized API frequency vector which are then fed to CNN and BPNN to train a feature embedding model. Their proposed system achieved an accuracy of 95%.

Zhong and Gu [30] proposed a multilevel deep learning system that selects important features from the dynamic and static features set, partitioned the set into many one-level clusters using K-means algorithm, generated cluster subtrees

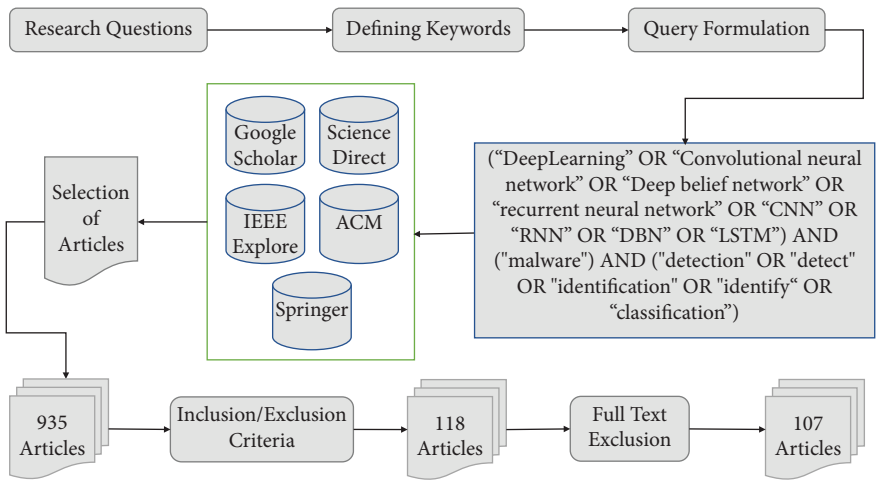


Figure 1: Research articles retrieval methodology of the systematic literature review.

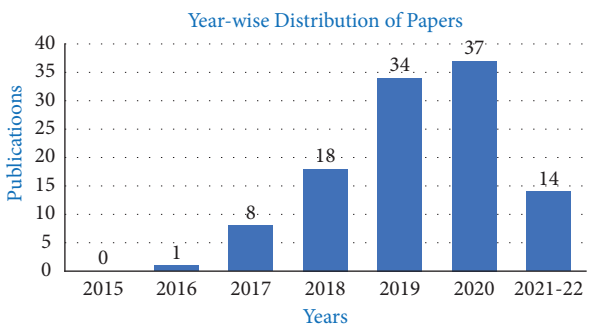


Figure 2: Year-wise distribution of the research work on the subject area malware detection using deep learning.

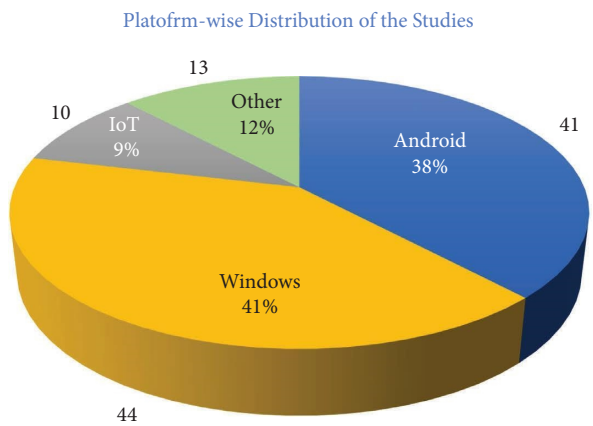


Figure 3: Platform-wise distribution of the research papers on the subject area and malware detection using deep learning.

and combined decision values of deep learning models in the tree for classification of the application as malware or benign.

Zhang et al. [31] proposed a ransomware detection system that transforms the op-code data and ransomware family label to numeric tensors to be used as input to the neural network. They use self-attention powered convolutional neural networks (SA-CNN) in their proposed method. The weakness of their system is the comparatively lower accuracy of about 90%.

Yuxin and Siyi [32] developed a deep belief network-based system that extracts the op-code of malware and used the neural network to detect it. Their system consists of a PE parser that transforms the PE file to op-code sequences, a feature extractor that selects n-grams that have strong classification power and to represent a PE file as n-grams vector, and a malware detection module. Their proposed model achieved about 98% accuracy.

Yue [33] proposes a weighted softmax loss (combination of softmax regression and entropy loss) for deep convolutional networks on malware image classification. It is claimed that this would resolve the issues that are caused by the imbalance of malware families.

Ye et al. [34] proposed a malware detection system that performs directly on Windows PE file. Their proposed system consists of a feature extractor that decompresses the file and parses PE code to extract the API calls from the file. Then they use unsupervised heterogeneous, autoencoder, and RBM-based deep learning model for malware detection.

Xiaofeng et al. [35] combined machine learning and deep learning and proposed an LSTM RNN-based malware detection system that uses API calls sequence and statistical features of malware. They run the malware in a sandbox to get the API calls and use random forest model to classify the system call sequence, which is then processed to get a feature vector that is used as input to the deep learning model for classification.

Vinayakumar et al. [36] proposed a distributed system ScalMalNet, which collects malware samples from different sources and processes the malware samples in real time or on demand basis in a distributed manner. They proposed an image processing framework for malware detection and classification using static and dynamic analysis. They applied various shallow learning and deep learning techniques for malware detection and experimentally shown that deep learning-based malware detection systems work much better than traditional ML-based systems.

Table 10: Journal-wise distribution of the papers on malware detection using deep learning.

S. no.	Journal	No. of papers	References
1	ArXiv preprints	11	[33, 40, 50, 52, 59, 66, 69, 94, 105, 107, 117]
2	Academia	1	[65]
3	ACM SIGCOMM computer communication review	1	[71]
4	Ad hoc networks, 2020—Elsevier	2	[95, 106]
5	Alexandria Engineering Journal	1	[76]
6	Cluster Computing	1	[79]
7	Computer Communications	1	[63]
8	Computers and Security	13	[26, 29, 39, 41, 42, 54, 60, 61, 87, 88, 122, 125, 127]
9	Cybersecurity	1	[28]
10	Digital Investigation	1	[43]
11	Engineering Applications of Artificial Intelligence	1	[80]
12	Expert Systems with Applications	1	[30]
13	Future Generation Computer Systems	4	[31, 86, 103, 116]
14	IEEE Access	8	[36, 99, 102, 108, 109, 114, 118, 128]
15	IEEE Transactions on Industrial Informatics	1	[48]
16	IEEE Transactions on Emerging Topics in Computational Intelligence	1	[119]
17	IEEE Transactions on Network Science and Engineering	1	[98]
18	IEEE Transactions on Sustainable Computing	1	[100]
19	IEEE Transactions on Information Forensics and Security	2	[68, 97]
20	IEICE Transactions on Information and Systems	1	[62]
21	IET Information Security	1	[38]
22	Information Sciences	1	[44]
23	International Journal of Advance Soft Computing	1	[82]
24	International Journal of Digital Crimes and Forensics	1	[45]
25	International Journal of Emerging Technology and Computer Science	1	[67]
26	International Journal of Performability Engineering	1	[73]
27	Journal of Ambient Intelligence and Humanized Computing	1	[74]
28	Journal of computer Virology and Hacking Techniques	2	[47, 91]
29	Journal of Computers	1	[81]
30	Journal of Grid Computing	1	[58]
31	Journal of Information Security and Applications	2	[37, 104]
32	Journal of Internet Services and Applications	1	[113]
33	Journal of King Saud University—Computer and Information Sciences	1	[111]
34	Journal of Parallel and Distributed Computing	3	[46, 49, 84]
35	Journal of Signal Processing Systems, 2021—Springer	1	[64]
36	Knowledge and Information Systems	1	[34]
37	KSII Transactions on Internet and Information Systems	2	[75, 85]
38	Mathematical Problems in Engineering	1	[101]
39	Microelectronics Reliability	1	[72]
40	Neural Computing and Applications	3	[32, 92, 112]
41	Neurocomputing	1	[78]
42	PeerJ Computer Science, 2020	1	[56]
43	Pertanika Journal of Science and Technology	1	[83]
44	PloS One, 2020	1	[57]
45	Procedia Computer Science	2	[35, 51]
46	Security and Communication Networks	3	[27, 89, 115]
47	Sensors	2	[96, 130]
48	Soft Computing	2	[77, 93]
49	TechRxiv	1	[90]
50	Tsinghua Science and Technology	1	[70]
51	Vehicular Communications	1	[110]
52	Entropy	1	[120]
53	Human-Centric Computing and Information Sciences	1	[121]
54	Applied Soft Computing	1	[123]
55	Journal of Internet Services and Information Security	1	[124]
56	Signal Processing Systems	1	[64]

Table 10: Continued.

S. no.	Journal	No. of papers	References
57	International Journal of Information Security	1	[126]
58	Cybernetics and Systems	1	[129]
59	International Journal of Computer Network & Information Security	1	[131]

Venkatraman et al. [37] investigated the use of image-based techniques for detecting suspicious behavior and proposed their own image-based malware detection technique by transforming the binary code of malware samples to grayscale images. They use both CNN and LSTM and develop a self-learning system that is capable of detecting the known malware as well as unknown malware.

Tang and Qian [38] proposed a CNN-based malware classification system that extracts API calls sequence of the application using dynamic analysis and generating feature image. They run the malware sample in a sandbox and extract the API call sequences and generate feature images using color mapping rules, category of the API and the number of times the category occurs in unit time. These images are used to train the convolutional neural network for detecting unknown malware. The strength of their system is the claimed accuracy of over 99% in most cases.

Rhode et al. [39] proposed an early detection system based on a recurrent neural network that works within the first five seconds of execution of a program and detects malicious behavior based on behavioral data. They generate machine activity data metrics based on the initial dynamic data and use them as feature input to the model. The features they used were CPU usage, packets sent and received, bytes sent and received, swap use, memory usage, number of current processes, and the maximum process ID assigned. They compared their RNN model with traditional machine learning algorithms and showed that deep learning performed much better, achieving an accuracy of 96%.

Rafique et al. [40] proposed a malware detection technique that uses the byte and ASM files to extract static features for classification. They use a convolutional neural network to extract features from the byte files, while for extracting features from ASM files, a wrapper-based technique is used. Then, they use feature space to train multilayer perceptron, which classifies the different malware categories of the BIG 2015 dataset.

Nguyen et al. [41] proposed a CNN-based deep learning system for malware detection that uses a modified form of control flow graph called lazy-binding CFG. They generate CFG from the binary code of the malware by using lazy-binding instead of early-binding for more precise results and convert the CFG to pixel image by transforming the CFG to adjacency matrix. By this way, variants of a specific malware are represented by closely similar objects, which are then inputted to the deep learning model for malware detection.

Namavar Jahromi et al. [42] proposed two-hidden-layer-based extreme learning machine (ELM) for malware detection. The system uses dependencies between malware features, like op-codes and API, calls to train the deep learning model. The extreme learning model has a different connection of input to the first hidden layer and is partially connected. They compared their proposed method with various deep learning methods and showed that their method achieved a better accuracy of above 99% in most cases.

Le et al. [43] proposed a malware classification system that is based on transforming the malware binaries to grayscale images. They use a convolutional neural network based deep learning model to classify malware, training their model using Microsoft Kaggle dataset. They developed three different models, one with CNN, another one with CNN and LSTM, and the third one with CNN and biLSTM achieving the highest accuracy of 98.2% with the third model.

Kim et al. [44] used transferred deep convolutional generative adversarial network (tDCGAN) to detect zero-day (a type of malware) by creating fake malware and feeding it to the modal to learn to distinguish a real malware.

Kalash et al. [45] proposed a CNN-based data-independent system for malware detection that uses the grayscale representation of the malware sample. The system reads the malware binary file in a vector of 8-bit integers and converts the binary value to the decimal equivalent and a new decimal vector representation is generated. Then, they represent this decimal vector as a two-dimensional matrix and transforms it to a grayscale image. The strength of their system is the high accuracy rate of 99.97%.

Huda et al. [46] proposed a deep belief networks-based system for detecting threats in the cloud-assisted Internet of things environment. To collect data, they execute the malware in a virtual sandbox, observe the change of states and collect any operations performed by it, and generate a report that includes the API calls and their parameters, which are used to prepare a frequency list of the APIs. Next, they train the deep belief networks-based model for malware detection.

Gibert et al. [47] proposed a convolutional neural networks-based system for malware classification that visualizes the malware as a grayscale image to extract features. They interpret each byte of the malware sample as a pixel and visualize the resulting two-dimensional array as a grayscale image. Then, they use a convolutional neural network-based system for extracting features from the images. Next, they use CNN with a softmax layer to classify the malware samples.

Cui et al. [48] proposed a CNN-based malware variants detection system. They first split the malware binary bit string into 8-bit substrings and consider each of the substrings as a pixel to visualize the image as a grayscale image. Then, they use a convolutional neural network based deep

learning system that consists of an input layer, a convolutional layer, a subsampling layer, and several fully connected layers to classify malware.

Cui et al. [49] proposed a CNN-based approach for malware detection that uses grayscale images generated from the malware executable. They also split the malware binary bit string into 8-bit substrings each of which is considered as a pixel. This way they visualize the malware as a grayscale image. Then, they use a CNN-based deep learning framework that consists of two convolution layers, one pooling layer, and two dense layers to classify malware samples.

Chen [50] proposed a deep transfer learning based method for malware detection. The malware binary was mapped into an integer in the range 0 to 255 to generate pixels and visualize the malware. Next, deep transfer learning-based model was used to classify malware samples. This approach was compared with several shallow learning approaches, and it was shown that deep learning achieved much better results.

Andrade et al. [51] developed an LSTM-based system for detecting five different families of malware, including rootkit, virus, Trojan, worm, and backdoor. They rely only on static analysis (or code analysis) of the malware to extract features of the malware file. Their model consists of an input layer, an LSTM layer, a dropout layer, and a dense layer. A weakness of their system is the average accuracy of 90%.

Agarap [52] proposed an SVM and CNN-based system for detecting malware. First, the binary string of the malware sample was transformed to 8-bit vectors, which are further processed and transformed to a grayscale image. Both multilayer perceptron and CNN were used for experiments and achieved better results with MLP; however, a below-average accuracy rate of about 80% was achieved.

Other papers that focus on Windows-based malware detection include [53–64].

5. Android Malware Detection

Like Windows platform, in android platform, researchers have worked on malware and intrusion detection using deep learning. This section analyzes the research work performed over android platform, extracting meta-information highlighted in research questions (RQ7 and RQ8), in addition to the information considered in Windows-based malware detection literature. Summary of android-based malware detection techniques is shown in Table 3.

Devi [65] proposed a permission based android malware detection system. They extracted the manifest files and permissions from the android packages and generated feature vectors and trained their model using neural networks and k-means clustering algorithm. The weakness of this approach is comparatively lesser accuracy of 88%.

Karbab et al. [66] proposed MalDozer, an android malware detection system based on the API method call sequence of the applications. MalDozer disassembles the classes.dex file of an android package to extract API method calls and discretize them by replacing each API method by an identifier and generates the semantic vectors. Then, they train the neural networks to predict android malware. The strength of their system is the high F_1 score achieved on various datasets.

Khedkar et al. [67] also proposed a permission-based android malware detection system. The FAST algorithm they designed use graph clustering method to cluster the features of the application and construct a trained dataset to classify new malware.

Kim et al. [68] used multiple features for malware detection including API methods, op-code features, permission features, share library function op-code features, component features, and environment features to generate feature vectors for each feature. These vectors are then fed to the classification model to predict malware. The strength of their approach is the usage of multiple features unlike most of the others who used a single or two features.

Milosevic and Huang [69] proposed a deep learning-based malware prediction system that uses CPU, memory, and battery usage to predict malware. Their unsupervised method is based on encoder-decoder and LSTM networks, using different applications to retrieve data, like CPU and battery usage. The weakness of their system is comparatively lower F_1 score of about 80%.

Yuan et al. [70] developed an online android malware detection system. The proposed system extracts three features, required permissions, sensitive API calls, and dynamic behavior and then use deep belief networks to detect malware in an application. Their deep learning model has two phases, an unsupervised learning phase and a supervised back propagation phase.

Yuan et al. [71] proposed a deep learning-based method that includes extracting features like permissions, sensitive API calls, and dynamic behavior. They used more than 200 different features in their proposed framework and used deep belief networks for malware detection. Their claimed accuracy is over 96%.

Yen and Sun [72] proposed a system that use the importance of words in apk file for malware detection. They extract the classes from apk file and convert them to java files and find the importance value of each word in the code. Then, they generate images by using the words importance from code using Term Frequency-Inverse Document Frequency (TF-IDF), a text mining and information retrieval method. These images were used to train and text their CNN based model.

Xie et al. [73] proposed a CNN-based approach, which includes extracting seven different malware features: API calls, hardware features, filtered intent, requested permissions, used permission, and restricted API calls. The framework consists of Dataset Construction, which includes collecting samples, labelling and features extraction, and classification process in which feature vectors are transformed to matrices and the dataset is divided to training set and validation set. The strength of their system is the claimed accuracy of 99.25%.

Wang et al. [74] combined deep autoencoder with a modified model of convolutional neural network they called CNN-S and proposed an android malware detection system that uses seven different features of applications to train their

model. The features include restricted API calls, suspicious API calls, permissions, requested permissions, hardware features, filtered intents, and code-related patterns. The strength of their system is the claimed high accuracy rate of 99.82%.

Luo et al. [75] proposed ITMF (image texture median filter) to analyze and detect android malware. Median filter is a filtering technique for removing or reducing noise from images and signals to improve processing and results. They obtain the malware binary file and convert it to a vector which is then transformed to grayscale image, which is then inputted to the ITMF. They extract features including API calls, used permissions, URL, and activity and train deep belief network for malware detection. They compared their model with shallow learning techniques and achieved better results with deep learning.

Saif et al. [76] proposed a deep belief network-based android malware detection system. They used both static analysis and dynamic analysis of android application and extracted features like manifest components, API calls, dynamic behavior of the application, and system calls and generated feature vector. They applied relief feature selection by using relief algorithm, which outputs another vector with the quality measurement of features. This vector is inputted to the deep neural network.

Pektaş and Acarman [77] proposed an android malware detection system that uses API calls graph. They build an API call graph for each execution path; the API call number is selected to generate graph embedding if it is equal to or greater than a threshold value and the graph embedding features are processed to be interpreted numerically. Then, the embedding vectors are inputted to the CNN-based deep learning model to classify malware.

Pektaş and Acarman [78] built an android malware detection system that examines all the execution paths and detect malware by using features extracted from instruction call graph. Their method consists of pseudodynamic analysis of the application in which call graphs and execution paths are extracted in terms of op-codes. Then, they construct a flow graph for each execution path and process the graphs to be interpreted numerically and to generate vectors. The vectors are then inputted to the LSTM RNN-based deep learning model to determine the probability of being benign or malware. They compared their approach with traditional machine learning approaches and showed that deep learning achieves better accuracy rate.

Nauman et al. [79] used different deep learning methods, including CNN, DBN, LSTM, and autoencoders on large-scale dataset for detecting android malware. They used the features from manifest file and those extracted through static analysis including requested permissions, components, filtered intents and restricted API calls, and so on as input to the deep learning model and evaluated the performance of different deep learning methods.

Martín et al. [80] proposed a deep learning-based system CANDYMAN, which classifies malware by combining dynamic analysis and Markov chains. They use DroidBox tool to run the application and extract dynamic behavior. The information gathered include network data, read/write operations, services, loaded classes, file, and SMS services and permissions, which is reported in a JSON file. In the next step, the data from the JSON file is represented in terms of Markov chains. Finally, the Markov chains are transformed into feature vectors that are then fed into deep learning networks for malware classification. They performed experiments using different machine learning and deep learning algorithms; however, they achieved a lower accuracy of around 81%.

Shiqi et al. [81] presented an attention-CNN-LSTM-based deep learning system for android malware detection. They use deep belief networks to extract texture fingerprint features and the malware activity embedding in vector space and then the malicious code is converted to grayscale image. The malware texture fingerprint features and the activity embedding in vector space are fed to the attention-CNN-LSTM-based deep learning model for malware classification. They compared their model to traditional machine learning algorithms and showed that they achieved a better accuracy with deep learning.

Halim et al. [82] proposed an android malware detection system that uses Bag of Words (BOW) model to extract various features of the application, including hardware components, used permissions, requested permissions, application components, filtered intents, restricted API calls, suspicious API calls, and network addresses. They used two different deep learning models for malware detection, a CNN-LSTM model in which CNN is stacked over LSTM and an LSTM-CNN model in which LSTM is stacked on top of CNN. They achieved an accuracy of 96.76% with the CNN-LSTM model while 98.53% with LSTM-CNN model.

Elsersy and Anuar [83] proposed a deep belief network-based deep learning system for android malware detection. They used Lasso features shrinkage and selection technique, which is used for features selection by means of absolute regularization penalty and evaluated traditional machine learning technique (K-NN classifier) and deep learning technique (DBN) for malware detection. They achieved better accuracy with deep learning technique; however, the accuracy rate they achieved was below average, 85.22%.

D'Angelo et al. [84] generated sparse matrices from the sequence of the API calls to be used for malware detection. Their autoencoders based system represents the temporal behavior of the application by using the sequence of sparse matrices and extract features from the sparse matrices, which are then used to classify the application as malware or benign-ware.

Chen et al. [85] proposed an android malware detection system that uses features like permissions and sensitive API calls extracted from the APK file. They model the features as a document and generate k-dimensional word vectors using word2vec. Finally, they use deep belief networks based deep learning system for malware classification.

Amin et al. [86] proposed an android malware detection system based on various deep learning methods. They extract the .dex file (dalvik executable file) from the APK file and further use it to extract the byte code. This byte code is given as input to the deep learning model for training,

feature engineering, and classification of the sample as malware or benign. They used different deep learning methods, including DAE, DBN, LSTM, BiLSTM, CNN, and RNN, and claimed to have achieved an accuracy of up to 99.9%.

Alzaylaee et al. [87] proposed a malware detection system for android platform that runs the application to extract its features. They use DynaLog, a platform that runs a large number of android applications in sequence to log and extract dynamic features, such as API calls, actions, events, and permissions. They extract 178 features and rank them using InfoGain to select the top 120 of them for experiments. Other research articles that focused on android malware detection include [88–99].

6. IoT/IoBT Malware Detection

A number of studies focused on malware detection in the Internet of Things and Internet of Battlefield (Military) Things (IoBT/IoMT) environments. These studies are analyzed and summarized in the following section.

Azmoodeh et al. [100] proposed a two-phase method for malware detection. They first generate the Op-code sequence graph by using the selected features and then use deep eigenspace learning to classify Internet-of-things and Internet of (battlefield) things (IoBT) malware. The strength of their system is the claimed accuracy of over 99%.

Xiao et al. [101] also combined machine learning and deep learning, combining DT, NB, SVM, and KNN with autoencoders. They proposed a behavior-based deep learning framework for malware detection in the Internet of Things environment. Their proposed model consists of IoT environment, which includes local computers and smart devices and a cloud platform module (CP module). CP provides storage space, constructs behavior graph, and transforms the API call graphs to binary vectors. These vectors are used as input to the stacked autoencoders-based deep learning model.

Ullah et al. [102] proposed a convolutional neural network-based system for detecting pirated software applications and files infected by malware in the IoT environment. Their system consists of a preprocessing module that transforms the malware binary file to grayscale image, a convolutional neural network to which the training images are inputted so that the classifier identifies the respective malware families using the images, a convolution layer that is used to extract meaningful features, and a pooling layer that is used to minimize the consequences of image distortion and increase CNN functioning. They claimed to have a better accuracy rate of 96% (piracy detection) and 97.46% (malware detection) as compared to other traditional machine learning techniques.

Haddadpajouh et al. [103] proposed an LSTM-based system for detecting malware in the Internet of things environment. They first collected samples of malware and benignware and decompiled them using object-dump tool. They used a Linux hash script to extract op-codes from the samples and used text mining techniques to generate features from the op-codes. Next, they used LSTM network with two hidden networks for malware detection. They compared their approach with several traditional learning methods and showed that they achieved much better accuracy rate with deep learning.

Al-Hawawreh et al. [104] proposed a deep autoencoder and deep feed-forward neural network-based system for detecting malicious activities in IoT environment. The deep autoencoder-based model learns to use normal network observations and creates initialization parameters and learns the representation of normal behavior. The parameters created at this stage are used as input to the DFFNN-based model for detecting new attacks.

Abusnaina et al. [105] used graph embedding to classify malicious programs in the IoT environment.

Naeem et al. [106] extracted bytecode from the java class file of the malicious software and used this bytecode for detecting malware in the Internet of things environment.

7. Other Platforms

Some of the studies focused on malware in the cloud environment, web applications or did not even mention what operating system or platform they targeted, or targeted multiple platforms at a time. These studies are analyzed in the following section and summarized in Table 5.

Lu et al. [107] constructed their own deep neural network for malware classification, which they named MalDeepNet. They also used several features, like API features, PID features, RET features, EXINFO features, and reboot features, and implemented TB-MalNet (Text-based MalNet) and IB-MalNet (Image-based MalNet) for malware prediction.

Yu et al. [108] investigated the use of deep learning algorithms for Domain Generation Algorithms (DGAs) and developed an LSTM-based deep learning model for DGA detection trained with weakly labelled data obtained from real traffic. They achieved an accuracy of around 98% on different datasets.

Vinayakumar et al. [109] proposed a deep learning-based distributed system for detecting cyber-attacks. The proposed system was built using big data processing frameworks Apache Hadoop YARN and Apache Spark. They used various shallow learning algorithms, like NB, RF, SVM, LR, KNN, and so on, and deep neural networks with different layers and evaluated their performance on different datasets and experimentally shown that deep neural networks perform better than traditional machine learning algorithms.

Song et al. [110] proposed an intrusion detection system for the controller area network in vehicles to protect the CAN bus of the vehicle. Their CNN-based system retrieves CAN IDs from the logged CAN and assembles data frames, each consisting of 29 sequential IDs. The data frames are then processed and classified as attack or nonattack. They also compared their proposed system with traditional machine learning techniques and experimentally showed that DL performed better. The strength of their system is the claimed F_1 score of above 99%.

Priyadarshini and Barik [111] proposed a DDoS defense system that is capable of detecting and mitigating denial of

service attacks in fog and cloud computing environment. They use Hogzilla dataset to train their LSTM-based deep learning model for DDoS defense and achieve a high accuracy rate of 98.88%.

Pektaş and Acarman [112] presented a hybrid, RNN- and CNN-based, deep learning system for detecting botnet. They extract flow features from the network traffic and transform them to multidimensional feature vector. The feature vector is inputted to the classification model for detecting whether the element is normal or botnet. They use the connection patterns created due to the data transmission between botnets and servers and split network traffic between endpoints and represent them as graph to extract features. The strength of their system is the high accuracy rate of nearly 99.4% on average.

Pan et al. [113] proposed a deep learning-based system for detecting attack in the web traffic by analyzing web applications. Their system uses robust software modeling tool (RSMT), which is a tool that targets languages that run on JVM, and extracts traces of program execution and generates models of behavior of the running application. RSMT captures features that represent program behavior, which are used as input to the Stacked Autoencoders-based deep learning model for detecting anomalies in web applications.

Loukas et al. [114] used deep multilayer perceptron and recurrent neural networks and built an intrusion detection system in the cloud environment. They used a robotic vehicle to evaluate their system by detecting various type attacks, including denial-of-service attack, command injection attack, and malware attack. They tested various traditional learning models and achieved much better average accuracy of 87% with deep learning.

Jeong et al. [115] proposed a system for detecting malware in PDF files. Their CNN-based model consists of one embedding layer, two convolutional layers, one pooling layer, one fully connected layer, and one output layer. The first layer is used to represent contextual meaning of the byte values and generate E-dimensional vectors, which are then given as input to the convolutional layers.

Homayoun et al. [116] proposed an LSTM and CNN-based deep learning approach for ransomware detection and classification. They use a deep feature extractor and a one-class classifier. It records the executed events when an application is started and transforms the sequence of the events to a numerical form and combines the input datasets to a single dataset. They use two different deep learning tasks for ransomware detection and classification, respectively. Other literature focusing on different types of malware and vulnerabilities detection include [117–119].

8. Research during the Recent Years

As obvious from Figure 2, the use of deep learning methods for malware and intrusion detection system is on the rise and has been increasing each year. In this section, we have selected a few important and most cited studies from the recent years (i.e., 2021–22) that the readers and researchers would be more interested in. Table 6 summarizes these studies.

9. Major Deep Learning Algorithms Used in Malware Detection

In order to answer RQ4 (What are the major DL algorithms used in the domain of malware detection?), we collected information about the usage of different DL algorithms in any form by the researchers. From the summary results shown in Figure 4, it is evident that Convolutional Neural Networks were used in most of the studies by the researchers which represents more than 50% of the publications surveyed, while LSTM-based neural networks in different forms were used by 25 researchers, which make up 25%. Similarly, DBN-based and AE-based algorithms were used in 13 and 11 publications that form 12% and 10.3%, respectively, of the total publications reviewed. Like many other domains, in malware detection and classification, most of the researchers have preferred convolutional neural networks when choosing a deep learning model. CNN is one of the most popular deep learning networks, which is capable of detecting the significant features without supervision. It is widely used for classification tasks, such as plant diseases, object detection, medical image analysis and computer vision, and so on. It is especially reported effective in image classification and image/object detection.

10. Discussion

10.1. Effectiveness of Deep Learning in Malware Detection. Deep learning produces best results with unstructured data. As most of the data produced by various systems is unstructured and in various formats, we either need to structure the data or have systems that have the capability to process unstructured data. Deep learning enables us to develop malware detection systems that can produce better results with unstructured and unlabeled data as well. Moreover, a deep learning algorithm can perform thousands of complex and repetitive tasks in very short time when trained once and produces accurate results as long as the raw data provided represents the problem.

Traditional malware detection techniques do not use machine learning or deep learning algorithms, and their performance is quite limited when it comes to detecting new types of malware. They rely on regularly updating their "malware definitions," which are used to detect threats. On the other hand, machine learning and deep learning-based algorithms can discover complex structures in structured and unstructured data when once trained and are very useful in developing effective malware detection systems. Hackers are developing malware that can change their code when propagated and thus hard to detect with the traditional pattern matching techniques. These malware can also deceive the traditional pattern matching-based systems easily in an intelligent manner. The different behavioral patterns that malware share could be used to detect unknown malware using ML and DL techniques.

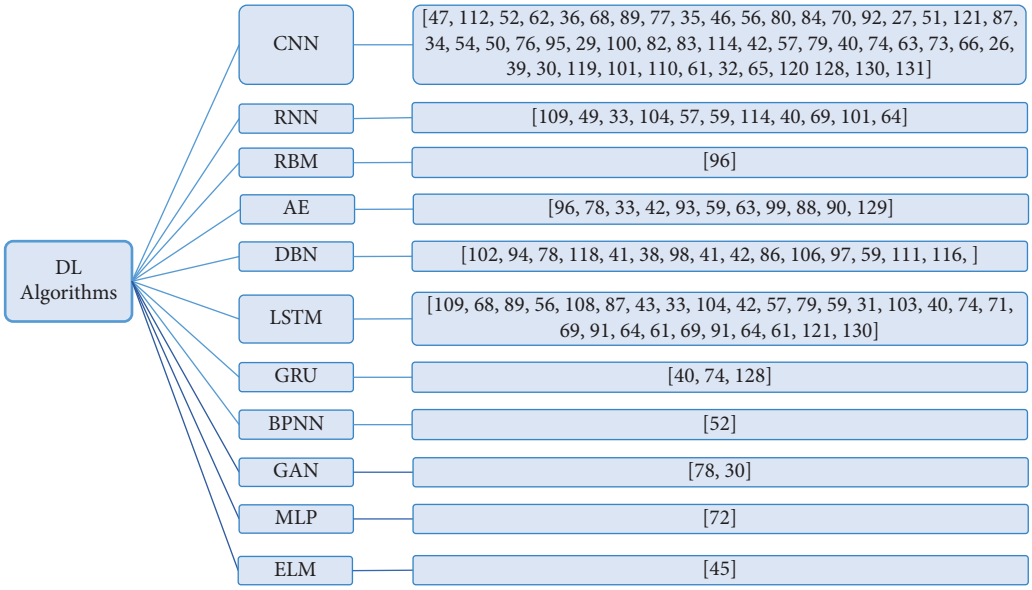


Figure 4: Major deep learning algorithms used in the domain of malware detection.

10.2. Performance of DL Compared with Traditional Learning. Deep learning algorithms have become popular as they can deliver more accurate results when trained with large amounts of data as compared to traditional learning algorithms. These algorithms can learn high-level features from data and mostly do not need domain expertise and hard-core feature extraction. The authors of some of the papers we reviewed used both deep learning and traditional learning algorithms for malware detection and experimentally showed that deep learning algorithms performed better than machine learning algorithms. Rhode et al. [39] proposed an early-stage malware detection system that is intended to detect malware within a few seconds of execution of a program. They used RNN and compared it with traditional learning algorithms, such as SVM. SVM achieved considerable accuracy of 80%, but RNN outperformed it after 1 second of execution and achieved an accuracy of 96% at 19 seconds into execution. Random Forest classifier achieved accuracy of 92% while Decision Trees achieved 92.6% accuracy. Haddadpajouh et al. [103] achieved an accuracy of 98.18% using RNN-LSTM for threat hunting in the Internet of things environment. They also used traditional machine learning algorithms and achieved the highest accuracy of 94% using KNN. Loukas et al. [114] developed an intrusion detection system for detecting cyber-attacks against vehicles, which achieved an accuracy of 86.9% with RNN. They also used several machine learning algorithms, including Logistic Regression, DT, RF, and SVM, achieving accuracy of 73.3%, 74%, 77.3%, and 79.9%, respectively. The cyber security threats detection system developed by Ullah et al. [102] performed better compared to other systems that used traditional learning algorithms and achieved much higher accuracy of 96%. Vinayakumar et al. [109] developed a deep neural network-based intrusion detection system and achieved an accuracy of up to 99.2%. They achieved a much lower average accuracy of around 80% with classical machine learning algorithms. Luo et al. [81] worked on detecting android malware and used Attention-CNN-LSTM in their system. They compared their deep learning-based model with SVM-based and KNN-based models and achieved a higher average accuracy of 96% compared to 95% with KNN and 94% with SVM. Similarly, Pektaş and Acarman [78] proposed an android malware detection system using instruction calls graphs and achieved 91.4% accuracy. They compared the proposed method with traditional learning algorithms, including KNN, Logistic Regression, SVN, and RF, and achieved accuracy of 80%, 70%, 79%, and 89%, respectively. Schranko de Oliveira and Sassi [90] acheived an accuracy of 91% with their deep neural network-based android malware detection system outperforming several ML algorithms, including SVM, RF, Logistic Regression, Extra Trees, and KNN. On the contrary, Jain et al. [55] achieved better accuracy of 97.7% using ELM with just one hidden layer as compared to 96.3% with CNN based architecture. Similarly, Pastor et al. [118] tested various traditional learning algorithms and CNN for detecting cryptomining traffic and achieved equal or better results with traditional learning algorithms.

In most of the cases, deep learning models performed much better than traditional learning methods. All these statistics show that deep learning algorithms are more capable of detecting and hunting malware or other threats and using shallow learning techniques may not lead us to a scalable solution with significant accuracy. However, it is not guaranteed that DL algorithms will always perform better than ML algorithms as some of the studies and our experiment's results depict. In our case, we compared the performance of Deep Autoencoders with different ML algorithms and achieved a higher accuracy rate with traditional learning models. Table 11 summarizes the results of our experiments. Our AE-based model could outperform only the Naïve Bayes ML algorithm.

Table 11: Analysis of the sustainability of android-based malware detection techniques.

Ref.	Evolvability/ability of identifying new malware? (yes/no)	DL model need updating/ retraining? (yes/no)	Sustainability/ resilience against evolution? (yes/no)	Sustainable up to years/accuracy	Initial accuracy/ F_1 score
Towards sustainable android malware detection [132]	Yes	After 5 years	Yes	5 (82%)	Above 93%
MaMaDroid [133]	Yes	After 2 years	Yes	2 (87%)	99%
Frequency analysis model (FAM)—a variant of MaMaDroid [134]	Yes	After several years	Yes	Several (76%)	81%
DroidSpan [135]	Yes	After 7 years	Yes	1–7 (21–37% superior than competitor)	6–32% superior than competitor
RevealDroid [136]	Yes	After 3 years	Yes	3 (87%)	98%
DroidCat [137]	Yes	After 9 years	Yes	9 (97%)	97%
G-Droid [138]	Yes	—	Yes	—	98.99%

10.3. Challenges in Malware Detection Using Deep Learning

10.3.1. The Issue of Data Unavailability. Machine learning and deep learning algorithms need a huge amount of training data to start with. In malware and threat detection, one of the biggest challenges is to provide a sufficient amount of malware and benign samples to the algorithm. Many datasets are available for use in research, but they need to be updated frequently so that the most recent malware samples could be used for training models.

10.3.2. The Issue of Data Noise and Model Overfitting. Another big challenge is the risk of wrongly labelled and noisy data, which may result in "overtraining the model" that leads to incorrect results.

10.3.3. The Issue of Model Validation and Response to New Threats. Many of the studies achieved high accuracy rates. However, they did not provide experimental results demonstrating how their systems would perform if a new type of malware attacked. New types of malware are being created across the world with passage of time and the malware detection systems should not only be able to detect variants of the malware samples that are used for training but also new types of malware to actually get the advantage of deep learning over the traditional threat detection systems.

10.3.4. The Issue of Evolution. Since a key issue in the android ecosystem is its fast evolution and various problems caused by the evolution [139, 140], the development of a flexible model that could be used at all times for the detection of new types of threats in the future is required. In the case the model does not need frequent retraining to cope with the situation of arising new malware but needs to update the model after a few years with very low degradation in the model performance, an evolvable model leads to sustainability in the model.

10.3.5. The Issue of Automatic Selection of DL Algorithm. In literature, a large number of DL methods can be found, which deal with complex problems, like malware classification over the big data. However, like ML algorithm (s) selection problem [141], the researchers always find it difficult to decide which method to pick for their problems in hand without frequently training, testing, and adopting the model.

10.3.6. The Issue of Sustainability. The frequent changes and continuous evolution of android malware demands for frequent retaining of the supervised malware detection models, which is a challenging job [142–144]. This requires building a sustainable malware detection model to update itself over the time in an effective and scalable manner. In case of declaring a model as sustainable, the frequency of retaining, duration for which the model is sustainable, and degradation in performance after the declared period are a few characteristics that need to be considered. For further details, Table 11 summarizes a few sustainable models with their key characteristics.

10.3.7. The Issue of Automatic Features Engineering and Analysis for Robust Modeling. One of the key challenges is how to pick or automatically learn, in the case of using deep learning for automatic feature engineering, features that stand the best of the time and future without frequent retraining. In literature, static, dynamic, and hybrid analysis methods have been used which automatically extract features for learning the DL model [145–147]. Column 2, "Automatic DL algorithm selection (yes/no)," of Table 3 summarizes these methods.

10.4. Data Quality for Malware Detection Using Deep Learning. Data quality is an essential component for machine and deep learning tools used for malware detection. Hence, along with technical approaches, availability of a sizable and informative dataset plays an essential role in the predictive accuracy of such systems. Therefore, the

Table 12: Android-based malware detection datasets for research community.

Reference	Dataset: description, size, type
[150]	15,451 benign apps and 15,183 malware
AndroZoo [151]	More than three million apps
AndroCT [152]	A large-scale dataset on the run-time traces of function calls in 35,974 benign and malicious android apps from ten historical years (2010 through 2019)
Rmvdroid [153]	Malware dataset containing 9,133 samples that belong to 56 malware families over the four years of 2014–18
[154]	17,664 apps sampled from the apps developed in each of the past eight years (2012–21)
AndroZooOpen [155]	AndroZooOpen, currently contains over 45,000 app artefacts, a representative picture of Github-hosted android apps
Deep ground [156]	Dataset (containing 24,650 malware apps)
DREBIN	In an evaluation with 123,453 applications and 5,560 malware samples DREBIN
Malgenome [157]	1,200 malware samples that cover the majority of existing android malware families, ranging from their debut in August 2010 to recent ones in October 2011

Table 13: Windows-based malware datasets.

Reference	Dataset: description, size, type
API Call dataset [149]	7107 different malicious software belonging to various families, such as virus, backdoor, trojan, and so on, have been analyzed, categorized into its different families, and made available for researchers to work on.
EMBER [158]	A labelled benchmark dataset for training machine learning models to statically detect malicious Windows portable executable files. This dataset includes features extracted from 1.1 M binary files.
SOREL-20 M [159]	A large-scale dataset consisting of nearly 20 million files with preextracted features and metadata, high-quality labels derived from multiple sources, information about vendor detections of the malware samples at the time of collection, and additional "tags" related to each malware sample to serve as additional targets.

quality of dataset should be carefully considered when creating and developing predictive models and tools for malware detection [148]. Such datasets are created by the research community to serve as a source of research for empirical analysis and extracting new insights about apps. In case of malware detection, data quality may be read as the number and types of apps used in android-based malware detection and operating system's application programming interfaces (API) calls in case of Windows-based malware detection [149]. The details are explained in the subsequent sections.

10.5. Android-Based Malware Detection Datasets. In literature, a number of android-based malware datasets have been curated and made publicly available for researchers to perform their research activities. The details of a few most important datasets are made available here, in Table 12, with brief description of their characteristics.

10.6. Windows-Based Malware Datasets. Like android-based malware detections datasets, different malicious programs belong to different families, such as backdoor, adware, downloader, dropper, spyware, trojan, virus and worms, and so on, have been studied by researchers in windows environment in the form of API calls. These are collected into different datasets, which can be used by analysts to build intelligent automatic malware detection systems in windows environment. A few examples of such datasets have been provided in Table 13.

11. Recommendations and Future Perspective

In this section, on the basis of our study and observations, we make some recommendations for the readers and the future researchers in the domain of security and malware threat detection using ML and DL techniques. A large number of researchers have focused on developing intelligent systems for malware threat detection and classification; however, very few of them have considered using big data analytics tools.

(i) With the growth of the Internet, the enormous amount of data being generated could not be handled using traditional data processing techniques. Big data analytics frameworks, such as Apache Hadoop and Apache Spark, are being adopted by organizations and websites to handle the big data generated in an efficient manner in relatively lesser time, which would not be possible otherwise.

(ii) The same is the case with security systems and threat hunting software, which need to deal with huge amounts of data within the machines and across the web. DL- and ML-based systems, integrated with big data processing tools, may be much more efficient and cost effective, especially in the domain of Internet security.

(iii) It is a big question whether the different DL-based security systems proposed in these studies would perform as good with big data as they perform theoretically.

(iv) We also observe that the deep learning-based security systems have mostly focused on windows and android platforms as compared to web-based security systems. Internet security is not less important than securing a computer offline or a smartphone.

(v) The results of this study lead us to the conclusion that developing deep learning-based intelligent Internet security systems is one of the areas in DL we need to focus on.

(vi) As obvious from the results of this study, many researchers have achieved a very high accuracy rate of up to 99.9% in malware detection; however, we need to have these malware hunting systems operating in real world performing as perfectly as these statistics show. Future researchers should focus on how to enable the users to easily and effectively use deep learning for protection against malware.

(vii) The scholars are recommended to work on sustainable and self-evolvable models that do not require frequent retraining.

12. Conclusion

In this review paper, we extensively studied the recent research publications that aimed at using deep learning for malware detection on various platforms, like Windows, smartphones, IoT, and the Internet. We searched five different libraries, including Google Scholar, Springer, Science Direct, IEEE Explore, and ACM Digital Library, to retrieve the relevant literature published during the last six years. We collected a total of 290 studies and then carefully studied all of them to select the studies to include in this survey. We excluded duplicates, non-English literature, book chapters, SLRs, and conference papers and finally selected a total of 107 publications to review. A lot of work has been done in the field of DL techniques for malware detection and classification, and various systems have been developed that have achieved accuracy as high as 99.9%. The statistics show that CNN, AE, RNN, and LSTM are used by most of the researchers. Python and Python-based libraries, like TensorFlow, Keras, and scikit-learn, are widely used to implement the DL models. However, there is less or no information about how these DL-based security systems would perform when applied in real-world scenarios and the question remains unanswered whether these systems would be able to handle the huge amounts of data being produced in organizations and online.

The current study presents a big picture of the deep learning-based models used for threats classification and detection on a number of platforms, including Windows, Android, IoT, cloud computing, and the Internet. In the future, we intend to conduct extensive reviews for each of these platforms that will be useful for the researchers focusing on a specific platform.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported with research funds from Research Incentive Funds (R20090), Zayed University, United Arab Emirates.

References

[1] N. Idika and A. P. Mathur, "A survey of malware detection techniques," *Purdue University*, vol. 48, no. 2, 2007.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] M. Christodorescu, S. Jha, S. A. Seshia, D. Song, and R. E. Bryant, "Semantics-aware malware detection," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P'05)*, pp. 32–46, IEEE, Oakland, CA, USA, 2005 May.

[4] E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: a survey," *Journal of Information Security*, vol. 05, no. 02, pp. 56–64, 2014.

[5] A. S. Bist, "A survey of deep learning algorithms for malware detection," *International Journal of Computer Science and Information Security*, vol. 16, no. 3, 2018.

[6] A. Naway and Y. Li, "A review on the use of deep learning in android malware detection," 2018, https://arxiv.org/abs/1812.10360.

[7] T. Sonali Kothari and V. Khedkar, "Analysis of recent trends in malware attacks on android phone: a survey using scopus database," *Library Philosophy and Practice*, pp. 1–20, 2021.

[8] B. Yadav and S. Tokekar, "Recent innovations and comparison of deep learning techniques in malware classification: a review," *International Journal of Information Security Science*, vol. 9, no. 4, pp. 230–247, 2021.

[9] M. V. R. Kumar, S. Anand Kumar, A. Bando, S. R. Gs, H. Shah, and S. C. Reddy, "A survey of deep learning techniques for malware analysis," *International Journal of Advanced Science and Technology*, vol. 29, no. 4, pp. 6031–6042, 2020.

[10] H. Lubuva, Q. Huang, and G. C. Msonde, "A review of static malware detection for Android apps permission based on deep learning," *International Journal of Computer Networks and Applications*, vol. 6, no. 5, pp. 80–91, 2019.

[11] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, no. S1, pp. 949–961, 2019.

[12] R. A. Ariyaluran Habeeb, F. Nasaruddin, A. Gani, I. A. Targio Hashem, E. Ahmed, and M. Imran, "Real-time big data processing for anomaly detection: a survey," *International Journal of Information Management*, vol. 45, pp. 289–307, 2019.

[13] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Computing and Information Sciences*, vol. 8, no. 1, pp. 3–22, 2018.

[14] S. Sharma and A. Kaul, "A survey on Intrusion Detection Systems and Honeypot based proactive security mechanisms in VANETs and VANET Cloud," *Vehicular communications*, vol. 12, pp. 138–164, 2018.

[15] J. Martínez Torres, C. Iglesias Comesaña, and P. J. Garcia-Nieto, "Review: machine learning techniques applied to cybersecurity," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 10, pp. 2823–2836, 2019.

[16] S. Hajiheidari, K. Wakil, M. Badri, and N. J. Navimipour, "Intrusion detection systems in the Internet of things: a

comprehensive investigation," *Computer Networks*, vol. 160, pp. 165–191, 2019.

[17] R. Coulter and L. Pan, "Intelligent agents defending for an IoT world: a review," *Computers & Security*, vol. 73, pp. 439–458, 2018.

[18] E. Benavides, W. Fuertes, S. Sanchez, and M. Sanchez, "Classification of phishing attack solutions by employing deep learning techniques: a systematic literature review," *Developments and advances in defense and security*, pp. 51–64, 2020.

[19] K. Bakour, H. M. Ünver, and R. Ghanem, "The Android malware detection systems between hope and reality," *SN Applied Sciences*, vol. 1, no. 9, pp. 1120–1142, 2019.

[20] M. Aly, F. Khomh, M. Haoues, A. Quintero, and S. Yacout, "Enforcing security in Internet of Things frameworks: a systematic literature review," *Internet of Things*, vol. 6, Article ID 100050, 2019.

[21] A. M. Aleesa, B. B. Zaidan, A. A. Zaidan, and N. M. Sahar, "Review of intrusion detection systems based on deep learning techniques: coherent taxonomy, challenges, motivations, recommendations, substantial analysis and future directions," *Neural Computing & Applications*, vol. 32, no. 14, pp. 9827–9858, 2020.

[22] Z. Wang, Q. Liu, and Y. Chi, "Review of android malware detection based on deep learning," *IEEE Access*, vol. 8, pp. 181102–181126, 2020.

[23] B. Kitchenham, "Procedures for Performing Systematic Reviews," *Keele UK Keele University*, vol. 33, pp. 1–26, 2004.

[24] B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering," 2007.

[25] S. Keele, "Guidelines for performing systematic literature reviews in software engineering," vol. 5, EBSE, Mumbai, India, 2007, Technical report Ver. 2.3 EBSE Technical Report.

[26] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," *Computers & Security*, vol. 77, pp. 871–885, 2018.

[27] Y. Zhao, C. Xu, B. Bo, and Y. Feng, "Maldeep: a deep learning classification framework against malware variants based on texture visualization," *Security and Communication Networks*, vol. 2019, Article ID 4895984, 11 pages, 2019.

[28] J. Zhang, K. Zhang, Z. Qin, H. Yin, and Q. Wu, "Sensitive system calls based packed malware variants detection using principal component initialized MultiLayers neural networks," *Cybersecurity*, vol. 1, no. 1, pp. 10–13, 2018.

[29] J. Zhang, Z. Qin, H. Yin, L. Ou, and K. Zhang, "A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding," *Computers & Security*, vol. 84, pp. 376–392, 2019.

[30] W. Zhong and F. Gu, "A multi-level deep learning system for malware detection," *Expert Systems with Applications*, vol. 133, pp. 151–162, 2019.

[31] B. Zhang, W. Xiao, Xi Xiao, A. K. Sangaiah, W. Zhang, and J. Zhang, "Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes," *Future Generation Computer Systems*, vol. 110, pp. 708–720, 2020.

[32] D. Yuxin and Z. Siyi, "Malware detection based on deep learning algorithm," *Neural Computing and Applications*, vol. 31, pp. 461–472, 2017.

[33] S. Yue, "Imbalanced malware images classification: a cnn based approach," 2017, https://arxiv.org/abs/1708.08042.

[34] Y. Ye, L. Chen, S. Hou, W. Hardy, and X. Li, "DeepAM: a heterogeneous deep learning framework for intelligent malware detection," *Knowledge and Information Systems*, vol. 54, no. 2, pp. 265–285, 2018.

[35] L. Xiaofeng, Z. Xiao, J. Fangshuo, Y. Shengwei, and S. Jing, "ASSCA: API based sequence and statistics features combined malware detection architecture," *Procedia Computer Science*, vol. 129, pp. 248–256, 2018.

[36] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019.

[37] S. Venkatraman, M. Alazab, and R. Vinayakumar, "A hybrid deep learning image-based analysis for effective malware detection," *Journal of Information Security and Applications*, vol. 47, pp. 377–389, 2019.

[38] M. Tang and Q. Qian, "Dynamic API call sequence visualisation for malware classification," *IET Information Security*, vol. 13, no. 4, pp. 367–377, 2019.

[39] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," *Computers & Security*, vol. 77, pp. 578–594, 2018.

[40] M. F. Rafique, M. Ali, A. S. Qureshi, A. Khan, and A. M. Mirza, "Malware classification using deep learning based feature extraction and wrapper based feature selection technique," 2019, https://arxiv.org/abs/1910.10958.

[41] M. H. Nguyen, D. L. Nguyen, X. M. Nguyen, and T. T. Quan, "Auto-detection of sophisticated malware using lazy-binding control flow graph and deep learning," *Computers & Security*, vol. 76, pp. 128–155, 2018.

[42] A. Namavar Jahromi, S. Hashemi, A. Dehghantanha et al., "An improved two-hidden-layer extreme learning machine for malware hunting," *Computers & Security*, vol. 89, Article ID 101655, 2020.

[43] Q. Le, O. Boydell, B. Mac Namee, and M. Scanlon, "Deep learning at the shallow end: malware classification for non-domain experts," *Digital Investigation*, vol. 26, pp. S118–S126, 2018.

[44] J. Y. Kim, S. J. Bu, and S. B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Information Sciences*, vol. 460-461, pp. 83–102, 2018.

[45] M. Kalash, M. Rochan, N. Mohammed, N. Bruce, Y. Wang, and F. Iqbal, "A deep learning framework for malware classification," *International Journal of Digital Crime and Forensics*, vol. 12, no. 1, pp. 90–108, 2020.

[46] S. Huda, S. Miah, J. Yearwood, S. Alyahya, H. Al-Dossari, and R. Doss, "A malicious threat detection model for cloud assisted internet of things (CoT) based industrial control system (ICS) networks using deep belief network," *Journal of Parallel and Distributed Computing*, vol. 120, pp. 23–31, 2018.

[47] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using convolutional neural networks for classification of malware represented as images," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 15–28, 2019, \.

[48] Z. Cui, F. Xue, X. Cai, Y. Cao, G. G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187–3196, 2018.

[49] Z. Cui, L. Du, P. Wang, X. Cai, and W. Zhang, "Malicious code detection based on CNNs and multi-objective algorithm," *Journal of Parallel and Distributed Computing*, vol. 129, pp. 50–58, 2019.

[50] L. Chen, "Deep transfer learning for static malware classification," 2018, https://arxiv.org/pdf/1812.07606.

[51] E. D. O. Andrade, J. Viterbo, C. N. Vasconcelos, J. Guérin, and F. C. Bernardini, "A model based on lstm neural networks to identify five different types of malware," *Procedia Computer Science*, vol. 159, pp. 182–191, 2019.

[52] A. F. Agarap, "Towards building an intelligent anti-malware system: a deep learning approach using support vector machine (SVM) for malware classification," 2017, https://arxiv.org/abs/1801.00318.

[53] E. Masabo, K. S. Kaawaase, and J. Sansa-Otim, "Big data: deep learning for detecting malware," in *Proceedings of the 2018 IEEE/ACM Symposium on Software Engineering in Africa (SEiA)*, pp. 20–26, IEEE, Gothenburg, Sweden, 2018 May.

[54] B. Yuan, J. Wang, D. Liu, W. Guo, P. Wu, and X. Bao, "Byte-level malware classification based on Markov images and deep learning," *Computers & Security*, vol. 92, Article ID 101740, 2020.

[55] M. Jain, W. Andreopoulos, and M. Stamp, "CNN vs ELM for Image-Based Malware Classification," 2020, http://arXiv.org/2103.13820.

[56] F. O. Catak, A. F. Yazı, O. Elezaj, and J. Ahmed, "Deep learning based Sequential model for malware analysis using Windows exe API Calls," *PeerJ Computer Science*, vol. 6, Article ID e285, 2020.

[57] Y. Fang, Y. Zeng, B. Li, L. Liu, and L. Zhang, "DeepDetectNet vs RLAttackNet: an adversarial method to improve deep learning-based static malware detection model," *PLoS One*, vol. 15, no. 4, Article ID e0231626, 2020.

[58] H. Darabian, S. Homayounoot, A. Dehghantanha et al., "Detecting cryptomining malware: a deep learning approach for static and dynamic analysis," *Journal of Grid Computing*, vol. 18, no. 2, pp. 293–303, 2020.

[59] R. Mitsuhashi and T. Shinagawa, "High-accuracy malware classification with a malware-optimized deep learning model," 2020, https://arxiv.org/abs/2004.05258.

[60] D. Gibert, C. Mateu, and J. Planes, "HYDRA: a multimodal deep learning framework for malware classification," *Computers & Security*, vol. 95, Article ID 101873, 2020.

[61] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-Based malware classification using ensemble of CNN architectures (IMCEC)," *Computers & Security*, vol. 92, Article ID 101748, 2020.

[62] M. Cho, J. S. Kim, J. Shin, and I. Shin, "Mal2d: 2d based deep learning model for malware detection using black and white binary image," *IEICE - Transactions on Info and Systems*, vol. 103, no. 4, pp. 896–900, 2020.

[63] Y. Sung, S. Jang, Y.-S. Jeong, and J. H. J. J. Park, "Malware classification algorithm using advanced Word2vec-based Bi-LSTM for ground control stations," *Computer Communications*, vol. 153, pp. 342–348, 2020.

[64] X. Huang, L. Ma, W. Yang, and Y. Zhong, "A method for windows malware detection based on deep learning," *Journal of Signal Processing Systems*, vol. 93, no. 2-3, pp. 265–273, 2021.

[65] K. Devi, "Android malware detection using deep learning," *International Research Journal of Engineering and Technology (IRJET)*, Academia, vol. 06, no. 05, 2019.

[66] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "Android malware detection using deep learning on api method sequences," 2017, https://arxiv.org/abs/1712.08996.

[67] Y. Suleiman, S. Sezer, and I. Muttik, "Android malware detection: An eigenspace analysis approach," in *Proceedings of the 2015 Science and Information Conference (SAI)*, pp. 1236–1242, 2015.

[68] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection using various features," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 773–788, 2019.

[69] N. Milosevic and J. Huang, "Deep learning guided Android malware and anomaly detection," 2019, https://arxiv.org/abs/1910.10660.

[70] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 114–123, 2016.

[71] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-sec: deep learning in android malware detection," in *Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 371-372, Chicago, IL, USA, 2014 August.

[72] Y. S. Yen and H. M. Sun, "An Android mutation malware detection based on deep learning using visualization of importance from codes," *Microelectronics Reliability*, vol. 93, pp. 109–114, 2019.

[73] N. Xie, X. Di, X. Wang, and J. Zhao, "Andro_MD: android malware detection based on convolutional neural networks," *International Journal of Performability Engineering*, vol. 14, no. 3, p. 547, 2018.

[74] W. Wang, M. Zhao, and J. Wang, "Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 8, pp. 3035–3043, 2019.

[75] S. Q. Luo, B. Ni, P. Jiang, S. W. Tian, L. Yu, and R. J. Wang, "Deep learning in Drebin: android malware image texture median filter analysis and detection," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 13, no. 7, pp. 3654–3670, 2019.

[76] D. Saif, S. Elgokhy, and E. A. Sallam, "Deep Belief Networks-based framework for malware detection in Android systems," *Alexandria Engineering Journal*, vol. 57, no. 4, pp. 4049–4057, 2018.

[77] A. Pektaş and T. Acarman, "Deep learning for effective Android malware detection using API call graph embeddings," *Soft Computing*, vol. 24, no. 2, pp. 1027–1043, 2020.

[78] A. Pektaş and T. Acarman, "Learning to detect Android malware via opcode sequences," *Neurocomputing*, vol. 396, pp. 599–608, 2020.

[79] M. Nauman, T. Ali, S. Khan, and T. A. Syed, "Deep Neural Architectures for Large Scale Android Malware Analysis," *Cluster Computing*, vol. 21, pp. 1–20, 2018.

[80] A. Martín, V. Rodríguez-Fernández, and D. Camacho, "CANDYMAN: classifying Android malware families by modelling dynamic traces with Markov chains," *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 121–133, 2018.

[81] L. Shiqi, Z. Liu, B. Ni, H. Wang, H. Sun, and Y. Yuan, "Android malware analysis and detection based on attention-CNN-LSTM," *Journal of Computers*, vol. 14, no. 1, pp. 31–43, 2019.

[82] M. A. Halim, A. Abdullah, and K. A. Z. Ariffin, "Recurrent neural network for malware detection," *Int. J. Advance Softwarw Computing. Appl*, vol. 11, no. 1, pp. 43–63, 2019.

[83] W. F. Elsersy and N. B. Anuar, "Android malware detection using deep belief network," *PERTANIKA JOURNAL OF SCIENCE AND TECHNOLOGY*, vol. 25, pp. 143–150, 2017.

[84] G. D'Angelo, M. Ficco, and F. Palmieri, "Malware detection in mobile environments based on Autoencoders and API-

images," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 26–33, 2020.

[85] T. Chen, Q. Mao, M. Lv, H. Cheng, and Y. Li, "Droidvecdeep: android malware detection based on Word2Vec and deep belief network," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 13, no. 4, pp. 2180–2197, 2019.

[86] M. Amin, T. A. Tanveer, M. Tehseen, M. Khan, F. A. Khan, and S. Anwar, "Static malware detection and attribution in android byte-code through an end-to-end deep system," *Future Generation Computer Systems*, vol. 102, pp. 112–126, 2020.

[87] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, Article ID 101663, 2020.

[88] X. Pei, L. Yu, and S. Tian, "AMalNet: a deep learning framework based on graph convolutional networks for malware detection," *Computers & Security*, vol. 93, Article ID 101792, 2020.

[89] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android malware detection based on a hybrid deep learning model," *Security and Communication Networks*, vol. 2020, Article ID 8863617, pp. 1–11, 2020.

[90] A. Schranko de Oliveira and R. J. Sassi, "Chimera: an android malware detection method based on multimodal deep learning and hybrid analysis," 2020.

[91] F. Mercaldo and A. Santone, "Deep learning for image-based mobile malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 16, no. 2, pp. 157–171, 2020.

[92] M. Amin, D. Shehwar, A. Ullah, T. Guarda, T. A. Tanveer, and S. Anwar, "A deep learning system for health care IoT and smartphone malware detection," *Neural Computing & Applications*, vol. 34, no. 14, pp. 11283–11294, 2020.

[93] X. Su, W. Shi, X. Qu, Y. Zheng, and X. Liu, "DroidDeep: using Deep Belief Network to characterize and detect android malware," *Soft Computing*, vol. 24, no. 8, pp. 6017–6030, 2020.

[94] Z. Ma, H. Ge, Z. Wang, Y. Liu, and X. Liu, "Droidetec: android malware detection and malicious code localization through deep learning," 2020.

[95] Z. Ren, H. Wu, Q. Ning, I. Hussain, and B. Chen, "End-to-end malware detection for android IoT devices using deep learning," *Ad Hoc Networks*, vol. 101, Article ID 102098, 2020.

[96] W. Niu, R. Cao, X. Zhang, K. Ding, K. Zhang, and T. Li, "OpCode-level function call graph based android malware classification using deep learning," *Sensors*, vol. 20, no. 13, p. 3645, 2020.

[97] R. Feng, S. Chen, X. Xie, G. Meng, S. W. Lin, and Y. Liu, "A performance-sensitive malware detection system using deep learning on mobile devices," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1563–1578, 2021.

[98] H. Zhu, Y. Li, R. Li, J. Li, Z. H. You, and H. Song, "SEDMDroid: an enhanced stacking ensemble framework for android malware detection," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 984–994, 2021.

[99] J. Feng, L. Shen, Z. Chen, Y. Wang, and H. Li, "A two-layer deep learning method for android malware detection using network traffic," *IEEE Access*, vol. 8, pp. 125786–125796, 2020.

[100] A. Azmoodeh, A. Dehghantanha, and K. K. R. Choo, "Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning," *IEEE transactions on sustainable computing*, vol. 4, no. 1, pp. 88–95, 2019.

[101] F. Xiao, Z. Lin, Y. Sun, and Y. Ma, "Malware detection based on deep learning of behavior graphs," *Mathematical Problems in Engineering*, vol. 2019, Article ID 8195395, pp. 1–10, 2019.

[102] F. Ullah, H. Naeem, S. Jabbar et al., "Cyber security threats detection in internet of things using deep learning approach," *IEEE Access*, vol. 7, pp. 124379–124389, 2019.

[103] H. Haddadpajouh, A. Dehghantanha, R. Khayami, and K. K. R. Choo, "A deep recurrent neural network based approach for internet of things malware threat hunting," *Future Generation Computer Systems*, vol. 85, pp. 88–96, 2018.

[104] M. N. Al-Hawawreh, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information Security and Applications*, vol. 41, pp. 1–11, 2018.

[105] A. Abusnaina, M. Abuhamad, H. Alasmary et al., "A deep learning-based fine-grained hierarchical learning approach for robust malware classification," 2020.

[106] H. Naeem, F. Ullah, M. R. Naeem et al., "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model," *Ad Hoc Networks*, vol. 105, Article ID 102154, 2020.

[107] S. Lu, L. Ying, W. Lin et al., "New era of deeplearning-based malware intrusion detection: the malware detection and prediction based on deep learning," 2019, https://arxiv.org/abs/1907.08356.

[108] B. Yu, J. Pan, D. Gray et al., "Weakly supervised deep learning for the detection of domain generation algorithms," *IEEE Access*, vol. 7, pp. 51542–51556, 2019.

[109] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[110] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, Article ID 100198, 2020.

[111] R. Priyadarshini and R. K. Barik, "A deep Learning Based Intelligent Framework to Mitigate DDoS Attack in Fog Environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 3, pp. 825–831, 2019.

[112] A. Pektaş and T. Acarman, "Deep learning to detect botnet via network flow summaries," *Neural Computing & Applications*, vol. 31, no. 11, pp. 8021–8033, 2019.

[113] Y. Pan, F. Sun, Z. Teng et al., "Detecting web attacks with end-to-end deep learning," *Journal of Internet Services and Applications*, vol. 10, no. 1, pp. 16–22, 2019.

[114] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D Gan, "Cloud-based cyber-physical intrusion detection for vehicles using deep learning," *IEEE Access*, vol. 6, pp. 3491–3508, 2018.

[115] Y. S. Jeong, J. Woo, and A. R. Kang, "Malware detection on byte streams of pdf files using convolutional neural networks," *Security and Communication Networks*, vol. 2019, pp. 1–9, 2019.

[116] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh et al., "DRTHIS: deep ransomware threat hunting and intelligence system at the fog layer," *Future Generation Computer Systems*, vol. 90, pp. 94–104, 2019.

[117] A. McDole, M. Abdelsalam, M. Gupta, and S. Mittal, "Analyzing cnn based behavioural malware detection techniques

on cloud iaas," in *Proceedings of the International Conference on Cloud Computing*, pp. 64–79, Honolulu, HI, USA, 2020, September.

[118] A. Pastor, A. Mozo, S. Vakaruk et al., "Detection of encrypted cryptomining malware connections with machine and deep learning," *IEEE Access*, vol. 8, pp. 158036–158055, 2020.

[119] A. N. Jahromi, S. Hashemi, A. Dehghantanha, R. M. Parizi, and K. K. R. Choo, "An enhanced stacked LSTM method with no random initialization for malware threat hunting in safety and time-critical systems," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 5, pp. 630–640, 2020.

[120] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An efficient densenet-based deep learning model for malware detection," *Entropy*, vol. 23, no. 3, p. 344, 2021.

[121] S. Baek, J. Jeon, B. Jeong, and Y. S. Jeong, "Two-stage hybrid malware detection using deep learning," *Human-centric Computing and Information Sciences*, vol. 11, no. 27, pp. 10–22967, 2021.

[122] G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, "Towards an interpretable deep learning model for mobile malware detection and family identification," *Computers & Security*, vol. 105, Article ID 102198, 2021.

[123] N. Zhang, Y. A. Tan, C. Yang, and Y. Li, "Deep learning feature exploration for android malware detection," *Applied Soft Computing*, vol. 102, Article ID 107069, 2021.

[124] V. Sihag, M. Vardhan, P. Singh, G. Choudhary, and S. Son, "De-LADY: deep learning based Android malware detection using Dynamic features," *J. Internet Server. Information. Security*.vol. 11, no. 2, pp. 34–45, 2021.

[125] I. Obaidat, M. Sridhar, K. M. Pham, and P. H. Phung, "Jadeite: a novel image-behavior-based approach for java malware detection using deep learning," *Computers & Security*, vol. 113, Article ID 102547, 2022.

[126] J. Kim, Y. Ban, E. Ko, H. Cho, and J. H. Yi, "MAPAS: a practical deep learning-based android malware detection system," *International Journal of Information Security*, vol. 21, no. 4, pp. 725–738, 2022.

[127] R. Chaganti, V. Ravi, and T. D. Pham, "Deep learning based cross architecture internet of things malware detection and classification," *Computers & Security*, vol. 120, Article ID 102779, 2022.

[128] X. Xing, X. Jin, H. Elahi, H. Jiang, and G. Wang, "A malware detection approach using autoencoder in deep learning," *IEEE Access*, vol. 10, pp. 25696–25706, 2022.

[129] M. Kumar, "Scalable malware detection system using distributed deep learning," *Cybernetics & Systems*, pp. 1–29, 2022.

[130] A. A. Hamza, I. T. Abdel Halim, M. A. Sobh, and A. M. Bahaa-Eldin, "HSAS-MD analyzer: a hybrid security analysis system using model-checking technique and deep learning for malware detection in IoT apps," *Sensors*, vol. 22, no. 3, p. 1079, 2022.

[131] S. S Lad and A. C. Adamuthe, "Improved deep learning model for static PE files malware detection and classification," *International Journal of Computer Network and Information Security*, vol. 14, no. 2, pp. 14–26, 2022.

[132] H. Cai and J. Jenkins, "Towards sustainable android malware detection," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, pp. 350-351, Gothenburg Sweden, 2018, May.

[133] E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, "Mamadroid: detecting android malware by building Markov chains of behavioral models," 2016.

[134] L. Onwuzurike, E. Mariconti, P. Andriotis, E. D. Cristofaro, G. Ross, and G. Stringhini, "Mamadroid: detecting android malware by building Markov chains of behavioral models (extended version)," *ACM Transactions on Privacy and Security (TOPS)*, vol. 22, no. 2, pp. 1–34, 2019.

[135] W. Li, X. Fu, and H. Cai, "Androct: ten years of app call traces in android," in *Proceedings of the 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pp. 570–574, IEEE, Madrid, Spain, 2021 May.

[136] J. Garcia, M. Hammad, and S. Malek, "Lightweight, obfuscation-resilient detection and family identification of android malware," *ACM Transactions on Software Engineering and Methodology*, vol. 26, no. 3, pp. 1–29, 2018.

[137] H. Cai, N. Meng, B. Ryder, and D. Yao, "Droidcat: effective android malware detection and categorization via app-level profiling," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 6, pp. 1455–1470, 2019.

[138] H. Gao, S. Cheng, and W. Zhang, "GDroid: android malware detection and classification with graph convolutional network," *Computers & Security*, vol. 106, Article ID 102264, 2021.

[139] H. Cai, "Embracing mobile app evolution via continuous ecosystem mining and characterization," in *Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems*, pp. 31–35, 2020, July.

[140] G. Suarez-Tangil and G. Stringhini, "Eight years of rider measurement in the android malware ecosystem: evolution and lessons learned," 2018, https://arxiv.org/abs/1801.08115.

[141] R. Ali, S. Lee, and T. C. Chung, "Accurate multi-criteria decision making methodology for recommending machine learning algorithm," *Expert Systems with Applications*, vol. 71, pp. 257–278, 2017.

[142] X. Fu and H. Cai, "On the deterioration of learning-based malware detectors for Android," in *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pp. 272-273, IEEE, Montreal, Canada, 2019 May.

[143] K. Xu, Y. Li, R. Deng, K. Chen, and J. Xu, "DroidEvolver: self-evolving Android malware detection system," in *Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 47–62, IEEE, Stockholm, Sweden, 2019 June.

[144] H. Cai, "Assessing and improving malware detection sustainability through app evolution studies," *ACM Transactions on Software Engineering and Methodology*, vol. 29, no. 2, pp. 1–28, 2020.

[145] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Computing Surveys*, vol. 44, no. 2, pp. 1–42, 2012.

[146] M. Akour, I. Alsmadi, and M. Alazab, "The malware detection challenge of accuracy," in *Proceedings of the 2016 2nd International Conference on Open Source Software Computing (OSSCOM)*, pp. 1–6, IEEE, Beirut, Lebanon, 2016 December.

[147] S. D. Nikolopoulos and I. Polenakis, "A graph-based model for malware detection and classification using system-call groups," *Journal of Computer Virology and Hacking Techniques*, vol. 13, no. 1, pp. 29–46, 2017.

[148] V. Sessions and M. Valtorta, "The effects of data quality on machine learning algorithms," *ICIQ*, vol. 6, pp. 485–498, 2006.

[149] F. O. Catak and A. F. Yazı, "A benchmark API call dataset for windows PE malware classification," 2019.

[150] H. Cai, X. Fu, and A. Hamou-Lhadj, "A study of run-time behavioral evolution of benign versus malicious apps in android," *Information and Software Technology*, vol. 122, Article ID 106291, 2020.

[151] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: collecting millions of android apps for the research community," in *Proceedings of the 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pp. 468–471, IEEE, Austin, TX, USA, 2016 May.

[152] W. Li, X. Fu, and H. Cai, "AndroCT: ten years of app call traces in android," in *Proceedings of the 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pp. 570–574, IEEE, Madrid, Spain, 2021 May.

[153] H. Wang, J. Si, H. Li, and Y. Guo, "Rmvdroid: towards a reliable android malware dataset with app metadata," in *Proceedings of the 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pp. 404–408, IEEE, Montreal, QC, Canada, 2019 May.

[154] H. Cai and B. G. Ryder, "A longitudinal study of application structure and behaviors in android," *IEEE Transactions on Software Engineering*, vol. 47, no. 12, pp. 2934–2955, 2021.

[155] P. Liu, L. Li, Y. Zhao, X. Sun, and J. Grundy, "Androzooopen: collecting large-scale open source android apps for the research community," in *Proceedings of the 17th International Conference on Mining Software Repositories*, pp. 548–552, Republic of Korea, 2020 June.

[156] F. Wei, Y. Li, S. Roy, X. Ou, and W. Zhou, "Deep ground truth analysis of current android malware," in *Proceedings of the International Conference on Detection Of Intrusions And Malware, and Vulnerability Assessment*, pp. 252–276.

[157] Y. Zhou and X. Jiang, "Dissecting android malware: characterization and evolution," in *Proceedings of the 2012 IEEE symposium on security and privacy*, pp. 95–109, IEEE, San Francisco, CA, USA, 2012 May.

[158] H. S. Anderson and P. Roth, "Ember: an open dataset for training static pe malware machine learning models," 2018, https://arxiv.org/abs/1804.04637.

[159] R. Harang and E. M. Rudd, "SOREL-20M: a large scale benchmark dataset for malicious PE detection," 2020, https://arxiv.org/abs/2012.07634.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 1424638, 16 pages
https://doi.org/10.1155/2022/1424638

Research Article

Digital Forensics as Advanced Ransomware Pre-Attack Detection Algorithm for Endpoint Data Protection

Jian Du,[1] Sajid Hussain Raza,[2] Mudassar Ahmad,[2] Iqbal Alam,[3] Saadat Hanif Dar,[4] and Muhammad Asif Habib[2]

[1]*Transport Information Security Center Co. Ltd, Transport Telecommunications & Information Center, Beijing, China*
[2]*Department of Computer Science, National Textile University, (NTU), Faisalabad, Pakistan*
[3]*Academic Department, Nan Yang Academy of Sciences (NASS), Beijing, China*
[4]*Department of Electrical Engineering, University of Azad Jammu & Kashmir, Muzaffarabad 13100, Pakistan*

Correspondence should be addressed to Muhammad Asif Habib; drasif@ntu.edu.pk

Received 19 November 2021; Accepted 22 May 2022; Published 6 July 2022

Academic Editor: Farhan Ullah

Ransomware is a malicious software that takes files hostage and demands ransomware to release them. It targets individuals, corporations, organizations, and public services such as hospitals and police stations. It is a growing industry that affected more than three million users from 2019 to 2020. The ransom payments totaled 25 billion-plus dollars in the year 2019. The latest version of ransomware was developed using undetectable and nonanalysis techniques. This paper represents an intelligent KNN and density-based machine learning algorithm to detect ransomware pre-attacks on an endpoint system. The data preprocessing and feature engineering techniques are augmented with the KNN algorithm for finding the solution. This helps the anti-malware developer, vendors, endpoint security provider companies, or researchers work on malware detection using advanced machine learning algorithms to develop the more effective ransomware defensive solutions to detect and prevent ransomware pre-attack execution. The proposed KNN and density-based algorithm will predict ransomware detection with higher accuracy than other machine learning algorithms. The anti-malware and anti-ransomware solution provider companies can use this algorithm to improve their existing ransomware detection solutions for endpoint users.

1. Introduction

Malware is a computer program used to access a computer system without the user's permission, usually for the benefit of a third-party known as a hacker. The malware contains malicious and harmful pieces of code, like ransomware, spyware, and other malware applications. In this paper, the only focus is on ransomware-based malware. Ransomware is a unique branch of malicious software that takes files hostage and demands ransomware to release them. It targets individuals, corporations, organizations, and public services such as hospitals and police stations. It is a growing industry that affects more than 3 million users from 2018 to 2019 [1]. The ransom payments totaled 25 billion-plus dollars [2]. The more advanced versions of ransomware contain anti-analysis techniques that are undetectable. The best guarantee against a ransomware attack is a good data backup. Ensure that any device comprising backups is safe and well protected. It needs to test the archived data from time to time. Ensure that any device containing backups is safe and well protected. Test the archived data from time to time.

A well-designed antivirus gets rid of ransomware in seconds, but ransomware designers are brilliant. They always work very hard to circumvent old-designed signature-based ransomware detection. Finally, it takes a receipt from an antivirus for a newly generated, zero-day ransomware attack to fetch your files. Even though the antivirus received a message that removed the ransomware, the files could not be recovered permanently. New generation antivirus programs complement signature-based detection and modify programs' monitoring behaviors. A few rely heavily on malicious behavior other than looking for threats, and behavior-

based malware detection, specifically looking towards ransomware behavior, is becoming more specific.

A schematic diagram of the working ransomware process is shown in Figure 1.

More than 300 ransomware types will exist until the end of 2022. However, this research discusses only a few types of ransomware that are the most harmful and dangerous for endpoint users. When examining the currently monitored ransomware types, our focus was only on the top five crypto-ransomware types/families: Cerber, CryptoWall, Locky, TeslaCrypt, and TorrentLocker.

Ransomware usually hunts files stored in public places such as desktops and documents folders in windows. A few antivirus toolkits and security packages prevent ransomware attacks by restricting unauthorized user access to these locations. Usually, some good programs, like Microsoft Word processing and spreadsheets, are already authorized. Every time they try to access it using a unique software program, you, as the user, can ask if you want to allow file access to the program. If this alert was unexpected and not generated by you, just report and block it. Using an online backup toolkit to keep your backup files up to date is undoubtedly the best possible protection against a ransomware attack. You get rid of the malware first, perhaps with the help of technical support from antivirus companies.

Once this task is complete, just restore all of your backup files in the previous state. Be aware that a ransomware attacker may try to encrypt your archive's backup files. A backup system that displays your backup files on a virtual drive can be highly vulnerable to a ransomware attack. Contact your backup provider to determine what protection the product offers against ransomware. Top five ransomware families trend in 2019 is shown in Figure 2.

1.1. Research Motivation. Machine Learning (ML) is a branch of Artificial Intelligence (AI) that studies systems' automatic learning without feeding any hard-coded program. ML is also a data analysis technique that computers use to analyze what humans and animals take from nature, like learning from their experiences. We proposed a unique and intelligent KNN and density-based adaptive clustering algorithm to detect ransomware pre-attacks on an endpoint system. Hence, Machine learning has become the most demanding topic. ML algorithms use computational methods to train information directly from given data without depending on predefined calculations as models.

With colossal data evolution every day, ML has become an essential technique for solving many information security problems. The general machine learning diagram process is shown in Figure 3. We used both supervised and unsupervised machine learning techniques to improve the detection of ransomware attacks for an endpoint.

1.1.1. Supervised Machine Learning. Supervised machine learning is used to build a data model for evidence-based prediction in uncertain situations. Supervised machine learning uses regression and classification techniques to develop a prediction model. Clustering is very helpful due to its maximum capacity to discover the previously unknown groups in the dataset. Five basic categories are the most common in cluster methods. One of them is the hierarchical method, which generates a hierarchical tree to divide into specific predefined datasets. Supervised machine learning uses regression and classification techniques to develop a prediction model.

Regression: In supervised machine learning, regression is used to predict continuous responses like changes in encryption methods such as hash or changes in the level of encryption keys. There is a relationship between independent and dependent sets of variables in regression analysis. There are some popular machine learning–based regression algorithms like linear regression, generalized linear model (GLM), support vector regression (SVR), Gaussian process regression (GPR), ensemble methods, decision trees, and neural networks.

Classification: In supervised machine learning, classification is used to predict the discrete response. If there is a set of mixed ransomware attack data, we can find the categories of good ware and ransomware, respectively, by applying classification techniques. These classified data can be categorized, tagged, and classified into different classes or groups. There are some popular machine learning-based classification algorithms like support vector machine, discriminant analysis, naive Bayes, and nearest neighbor.

1.1.2. Unsupervised Machine Learning. It deals with studying inherent structures and fetching hidden patterns of data. Unsupervised machine learning contains unlabeled data. Unsupervised machine learning uses clustering techniques for the development of data models.

Clustering: In unsupervised machine learning, clustering is a technique used for an exploratory data analysis to fetch or recognize hidden groupings or patterns in data. Market research, object recognition, and sequence analysis are the most common clustering applications. Some popular machine learning–based regression algorithms are K-Means, K-medoids, fuzzy, c-means, Hierarchical, Gaussian mixture, and hidden Markov model.

1.1.3. Research Challenges. Many researchers, professionals, and security experts have developed good ransomware analysis and detection systems. Still, they only focused on either network-based preventive systems or OS-based prevention systems. Some only focus on static analysis or dynamic solutions to analyze ransomware-affected strategies.

1.1.4. Main Contribution. This paper proposes a unique and intelligent KNN and density-based adaptive clustering algorithm to detect ransomware pre-attacks on an endpoint system. The implemented algorithm is better than the previously developed ransomware analysis and detection algorithms. The first step is to collect ransomware data of the most relevant multi-class ransomware families that only affect Windows-based operating systems in the form of exe files. The data is collected or provided by a data engineer,

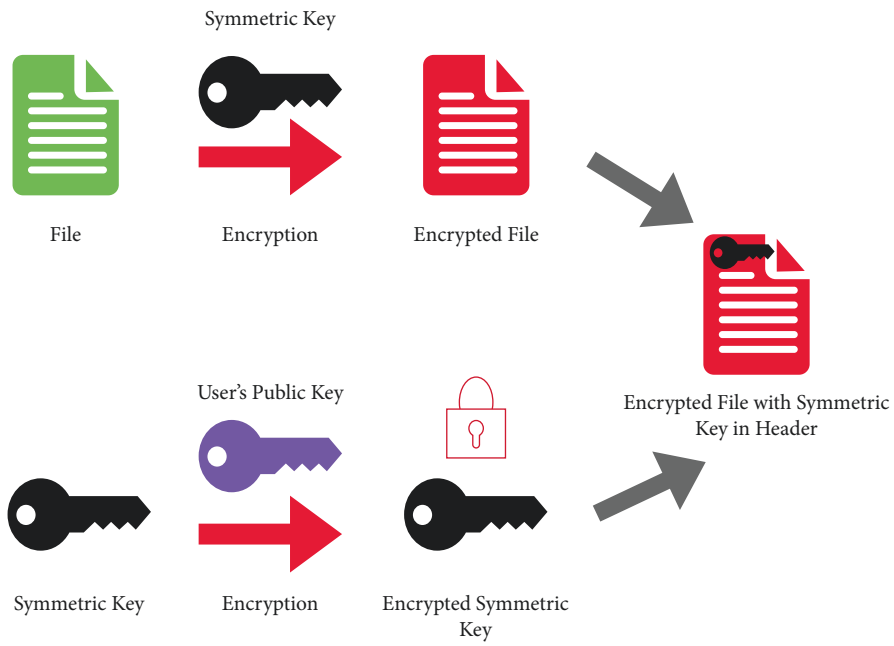


Figure 1: Ransomware generic working process.

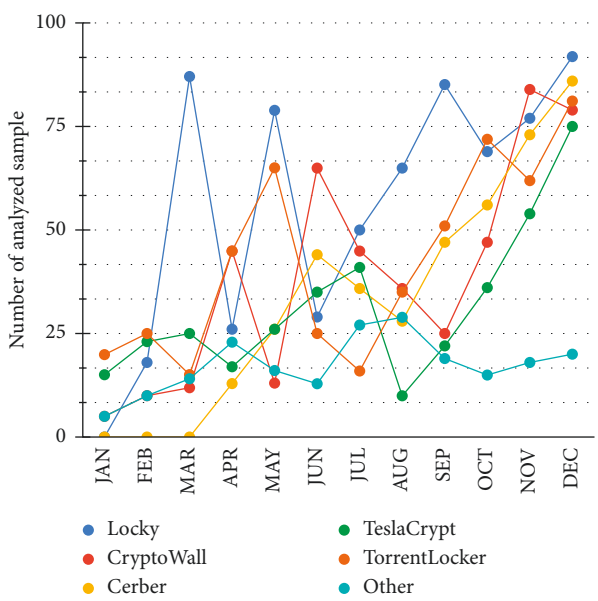


Figure 2: Top five ransomware families' trend samples in 2019.

respectively. We use a dataset of ransomware from the Github repository [3]. Then, we check for the outliers by initializing threshold values like measuring the distance of the closest data point, which is greater than its nearest cluster identifier and is identified as an outlier in our dataset. The next step is to remove outliers and clean up the ransomware dataset.

Data cleaning may generate some missing values, and we must fix them. The second step, wrangle a ransomware dataset to create new variables, identify duplicates, and filter variables. The next step is ransomware data's feature engineering. It features extraction using recursive feature elimination (RFE) and principal component analysis (PCA), which use orthogonal transformation to convert data into lower dimensional space like in between a specific number ranges to maximize the variance of the dataset. Then shuffle the extracted feature dataset to apply machine learning–based KNN and density algorithms and get trained in the dataset with the ratio of 70:30 as training and testing data. The proposed KNN and density-based algorithm predicted ransomware detection with higher accuracy than other machine learning algorithms. Finally, the anti-malware and anti-ransomware solution provider companies can use this algorithm to improve their existing ransomware detection solutions for endpoint users.

1.1.5. Use of Clustering. In clustering, the number of data objects is divided into subsets. Every individual subset is known as a cluster, so objects in a cluster are similar to each other but not different from other clusters. Separation is done using a clustering algorithm [4]. Clustering is very helpful due to its maximum capacity to discover the previously unknown groups in the dataset. Five basic categories are most common in cluster methods. One of them is the hierarchical method, which generates a hierarchical tree to divide into specific predefined datasets. These methods can also be categorized into two more types of operating modes. 1- Decomposition (top-down), 2- Concatenation (bottom-up) and contain BIRCH (CURE)-ROCK & CHEMALOEN. Second-one is the division method. The first k-partition is generated, and then the objects are shifted from one substance to other substances (partitions) to increase the substance quality with cycle positioning techniques that contain K means- CLARINS and fuzzy c-means algorithms.

These methods can also be categorized into two more types of operating modes; decomposition (top-down) and Concatenation (bottom-up), which contain BIRCH

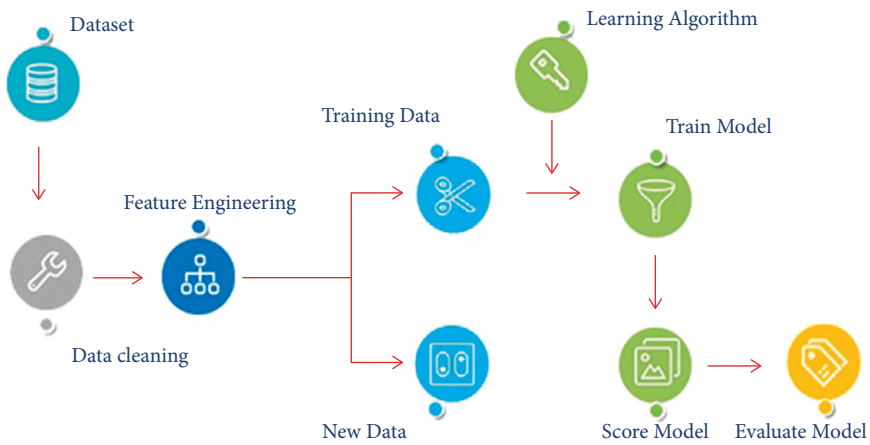


Figure 3: General machine learning process diagram.

(CURE)-ROCK & CHEMALOEN. Second mode is the division method. The first K- partition is generated. Then, the objects are shifted from one substance to another to increase the substance quality with cycle positioning techniques containing K means- CLARINS and Fuzzy C-means algorithms.

1.2. Density-Based Method. Density-Based Method (DBM) groups objects under Object Density (OD). e.g., DBSCAN describes the clusters as areas of point-to-point density. The clustering algorithm also enters data by constantly developing areas of high density. It can detect groups of random numbers in the space as clusters in a database by using noise. The Density-Based Clustering (DBC) algorithm can process clusters of any kind because the time complexity is less than $O(n^2)$ and is acceptable for processing large amounts of data. For e.g., the DBSCAN algorithm. Figure 4 shows the types of points in DBSCAN.

The density-based clustering (DBC) algorithm can process clusters of any kind because the time complexity is more minor than O(N2) and is acceptable for processing large amounts of data. For e.g., the DBSCAN algorithm. This differentiates the main points from less essential points. Still, there is a drawback to specifying two parameters; the radius of the surrounding area (RSA) and the lowest amount of points in a given surrounding area. Moreover, DBCLASD uses the sample distance to its nearest neighbor as a random variable, and the sample density is calculated from the probability distribution of this distance.

The best advantage is that the parameters do not have to be specified, and the disadvantage is that the uptime is about twice as long as the DBSCAN algorithm. The fundamental principle attracts much attention concerning the algorithm based on more than three similar algorithms. The only methods assume that neighbors surround the cluster center. Density-based clustering algorithms (DBCA) are not only widely used at any point with a higher density [5] but are also being further developed [6]. In particular, KNN and density-based methods are integrated to increase Overall Cluster Efficiency (OCE).

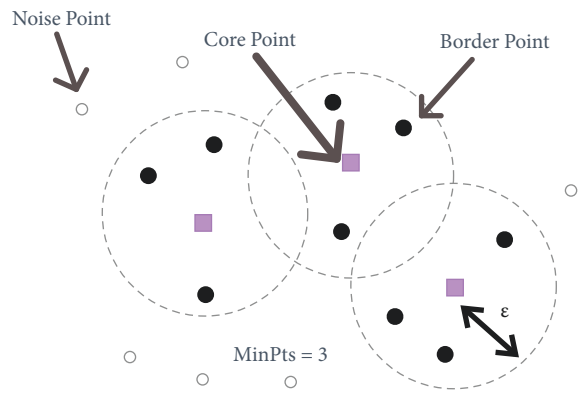


Figure 4: Types of points in DBSCAN.

Analysis of clusters is highly used in many science fields, such as social science, bio, stat, pattern recognition, info fetching, machine-learning (ML), and Artificial Intelligence (AI). For e.g., analysis of clusters is used to categorize related search documents, find genes and proteins with the same functions, and classify earthquake-prone locations.

Moreover, the analysis of clusters is used as a preprocessing step in multiple applications such as data compression and efficient search of the closest points. The already defined clustering algorithms face several challenges today due to their growing size and uneven data distribution. The clustered data are probably a challenging task complicated by the random shapes of the clusters, their varying sizes, different densities, and ambiguous separation.

Many researchers have studied clustering by density under a promising approach to addressing these challenges. They can find randomly formed clusters and handle noise without first knowing the number of clusters. The cluster is generally considered in high-density areas separate from low-density regions. This hypothesis allows groups to be characterized by nonconvex shapes and zones. The effectiveness of existing density-based clustering algorithms relies heavily on intensive user arbitration to determine density thresholds, which determines the differentiation of all density levels available in the data. This density limit is

often harder to calculate for large amounts of anonymous data. Additionally, using a global density threshold can lead to nonclassified clusters with lower densities.

Decision-tree works on the entire dataset by using all of its variables and features, but the random forest only selects elements and variables randomly. The random forest has to build multi-level decision trees from the dataset and then show the results in averages.

2. Literature Review

A literature review of published work and development on Machine Learning Algorithm based ransomware detection is discussed in this section. A few years ago, when the Harvard-trained evolutionary biologist Joseph Pop developed the first ransomware virus of its kind in 1989, it was called trojan AIDS, also known as PC Cyborg. Pop sent 20,000 introductory discs with information on AIDS infection to the World Health Organization (WHO)'s International AIDS Conference attendees. AIDS trojan horse is the first generation of ransomware malware and is relatively easy to remove. The trojan uses simple symmetric cryptography, and a tool to decrypt file names is available soon, but the trojan horse AIDS created the conditions for what lies ahead. Top 10 malware categories are shown in Figure 5.

In the last few years, ransomware turned pro. It is usually installed on a user's workstation (PC or Mac) using psychological manipulation attacks that trick users into clicking a link or opening an attachment. Once on a computer, the malware begins encrypting any data files it can find in the system itself and any networks that come under computer access. Then, suppose the user requests access to one of these files. In that case, it is blocked, and the system administrator, who is receiving a warning from the user, finds two files in the directory indicating that the file was used as a ransom and how to pay the ransom decrypted files. The techniques used by cybercriminals are constantly evolving to bypass traditional protections.

Some significant ransomware are WAnnaCry, Petya/NotPetya, Bad Rabbit, and Locky. Ransomware is a highly successful criminal business model. According to a report from Cybersecurity Ventures, the annual cost of ransomware is projected to exceed $11.5 billion by 2019. Finding and replying to copies of phishing emails to 3.8 billion people worldwide, papered to reach 6 billion by 2022, online is the next best thing to vaccinating them with ransomware. Cybersecurity Ventures predicts that cybercrime will cost the world more than $6 trillion per year by 2021, up from $3 trillion in 2019.

Ransomware is expected to worsen and cause a more significant share of total cybercrime by 2021. Employees are an essential variable and a notable potential winner in reducing the cost of ransomware damage. The report from Cybersecurity Ventures, due for release in 2018, includes the estimated cost of ransomware for the five years from 2017 to 2021 [7, 8]. Table. 1 describes the top five ransomware families trend samples in 2019 concerning crime type in a loss.

The ML techniques can work more effectively and efficiently to detect malware [? 8]. They used dynamic analytics; the reports collected by some online analytics services could not evaluate various ML classifiers. The best performing J48 classifier from the decision tree achieved 97.3% accuracy and a percentage of False Positives (FPR) of 2.4%. They proposed a framework for automated malware detection behavior using ML [9, 10]. They embed observed behavior in a vector space and applied clustering algorithms to improve previous work in this area significantly.

ML was highly needed to overcome existing limitation techniques for more effective detection of malware [11]. Authors in Ref. [4] introduced a statically and dynamically integrated classification method to overcome the limitations associated with each technique [4]. They demonstrate the importance of combining old and new malware patterns to overcome malware authors' avoidance techniques. Their accuracy was at least 5% lower when only new samples were used in all classifiers assessed. Similarly, one more detailed review of the methods and technical tools used to analyze and classify malware has been conducted [5]. They include processes for acquisition, static or dynamic analysis or feature extraction, and ML classification. The flow control method achieved an accuracy increase of 2% compared to the text-based method using ANN classification [12].

A similar technique was being used with additional filtering features to achieve 97% accuracy with random forest classifiers [13]. The ransomware classification system has approximate values of the control flow graph adjustment. The distance metrics are based on the distance between the string-based signature feature vectors. It is being done to achieve the best accuracy rates [14]. Additional studies on size reduction/screening were included in which a two-step approach was proposed to the dimensional reduction that significantly combines feature selection and extraction to reduce training feature size and classification [15].

They used the reduction function to identify the top 10 malware detection codes and reduced the controlled learning algorithm's training time by 91% without losing accuracy [16]. The feature reduction identifies nine features that differentiate malware from good software. With the random forest classifier, an accuracy of 99.60% was achieved [17]. A ransomware detection framework is based on 20 extracted file systems and registry events [18].

With the Bayesian network model it achieves an F-measure of 93.3%. Most recently, in 2017, the use of a sequential sample file system, registry, and DLL event is being fetched to achieve 97% accuracy when differentiating between crypto-ransomware and valuable software and 95.5% when distinguishing between three different ransomware groups [6, 12]. They introduce a new method (Adaptive-DP) that estimates density using the heat diffusion method and the adaptive approach to select the exact number of cluster centers.

The limitations of DP, the difficulty of choosing an appropriate density estimation method, the selection of boundary distances, and the human interpretation required to select the number of cluster centers have been improved

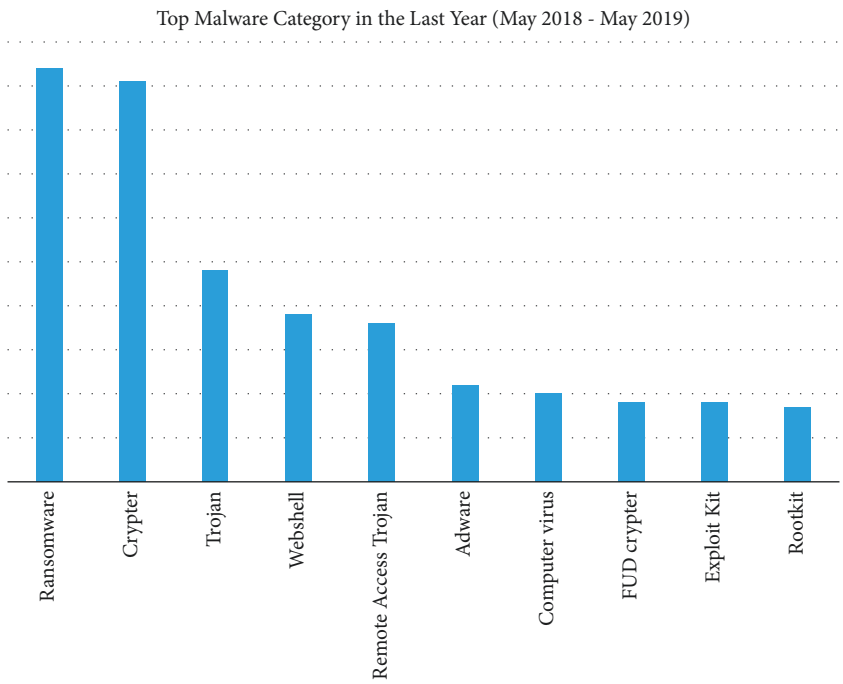


Figure 5: Top 10 malware category mentions overall [2].

Table 1: Top five ransomware families' trend samples in 2019.

Crime Type	Loss
BEC/EAC	$676,151, 185
Confidence fraud/Romance	$211,382,989
Nonpayment/Nondelivery	$141,110,441
Investment	$96,844,144
Personal data breach	$77,134,865
Identity theft	$66,815, 298
Corporate data breach	$60,942,306
Advanced fee	$57,861,324
Credit card fraud	$57,207,248
Real estate/Rental	$56,231,333
Overpayment	$53,450,830
Employment	$38,883,616
Phishing/Vishing/Smishing/Pharming	$29,703,421
Other	$23,853, 704
Lottery/Sweepstakes	$16,835,001
Extortion	$15,302,792
Tech support	$14,810,080
Misrepresentation	$14,580,907
Harassment/Threats of violence	$12,569,185
Government impersonation	$12,467,380
Civil matter	$5,766,550
IPR/Copyright and counterfeit	$5,536,912
Malware/Scareware/Virus	$5,003,434
Ransomware	$2,344,365
Denial of service/TDoS	$1,466,195
Charity	$1,405,460
Health care related	$925,849
Re-shipping	$809,746
Gambling	$598,853
Crimes against children	$46,411
Hacktivist	$20,147
Terrorism	$18,926
No lead value	$0

in Adaptive-DP [19, 20]. They have developed a RE framework for detecting ransomware infected data recovery by using machine learning and features generation engines [21].

This RE framework can analyze malware files at a multi-level by using binary codes and different libraries. The author used a portable executable parser with object code dump Linux-based tools to decode binary-level assembly instructions and DLL, respectively [19]. Their experiments show the performance of detecting ransomware samples accurately from 76% to 97% using eight machine learning techniques where seven results out of eight showed 90% accuracy for detection rate [22]. This study is purely based on the static level of analysis to differentiate between ASM and DLL levels as compared to regular binaries [3].

The research gap was founded after careful study of the literature. We evaluate shallow and deep networks for ransomware detection and classification. To characterize and differentiate the ransomware from different types of other ransomware families, some researchers use API. Some previous researchers used nonmachine learning techniques to achieve more accurate results in detecting ransomware. Unfortunately, over time, due to the more advanced use of machine learning techniques used by the hackers to develop the ransomware, the endpoint use required more accurate solutions against ransomware.

Based on the literature review, we analyze that many researchers, professionals, and security experts have developed good ransomware analysis and prevention systems. Still, they only focused on either network-based prevention systems or only on OS-based prevention systems [23]. Some only focus on static analysis or dynamic solutions to analyze ransomware-affected strategies. That is why we proposed a

unique and intelligent KNN and density-based Adaptive clustering algorithm to detect ransomware pre-attacks on an endpoint system. So this suggested algorithm shows better results than previously developed ransomware analysis and detection algorithms. There are various applications of machine learning and deep learning in different applied domains of artificial intelligence [24, 25].

2.1. Problem Statement. There is a need for possible detection solutions against zero-day attacks and random file system detection. There is also a need to use a more advanced and more accurate machine learning detection technique like KNN and a density-based adaptive clustering algorithm for improving ransomware pre-attacks detection. Proposed solution: Our proposed KNN and density-based machine learning algorithm efficiently detects and analyzes ransomware pre-attack on an endpoint's system. This proposed solution also offers the detection of zero-day attacks as well.

2.1.1. Objective. To detect zero-day attacks and random file system attacks by using portable executables (.exe). To provide an intelligent KNN and density-based adaptive clustering algorithm for detecting ransomware pre-attacks on an endpoint's system.

2.1.2. Aim of Our Research. This project aims to provide a KNN and density-based adaptive clustering algorithm to detect ransomware pre-attacks on an endpoint's system. This proposed solution also offers the detection of zero-day attacks as well.

2.1.3. Research Questions. Q.1. How to detect and analyze ransomware using KNN and density-based algorithms? Q.2. How to perform detection of zero-day and random file name attacks using KNN and density-based algorithms?

2.1.4. Scope of Research. The scope of the suggested algorithms is very high for the detection and analysis of ransomware. To get live ransomware detection and zero-day attack updates, one can embed or deploy these algorithms in the Microsoft BI dashboard. It also has a limitation in scope because these algorithms are only trained for analyzing and detecting portable executable files (.exe format).

Table 2 shows a comparison of methods for the analysis and protection of ransomware attacks used by many researchers in literature.

There is a need for possible detection solutions against zero-day attacks and random file system detection. There is also a need to use a more advanced and more accurate machine learning detection technique like KNN and a density-based Adaptive clustering algorithm for improving ransomware pre-attacks detection to detect zero-day attacks and random file system attacks by using portable ex-executable in.exe format and to offer an intelligent KNN and density-based Adaptive clustering algorithm for detecting ransomware pre-attacks on an endpoint's system. To get live ransomware detection and zero-day attack updates, one can embed or deploy these algorithms in the Microsoft BI dashboard. It also has a limitation in scope because these algorithms are only trained for analyzing and detecting portable executable files in.exe format.

3. Research Methodology

Many windows based ransomware detection techniques and algorithms are available. These detection algorithms are working very effectively. But with the continued growth of zero-day attacks and the new generation of ransomware families' attacks on endpoint systems, there is a need to improve the malware detection techniques and algorithms. In this research, for the detection of ransomware in supervised ML, we used KNN and density-based algorithm and random forest with outstanding accuracy results. On the other hand, to detect zero-day attacks that lie under unsupervised ML, we implemented k-means and DBSCAN clustering algorithms to detect zero-day attacks. The methodology used in this research is shown in Figure 6.

For this purpose, we need to fetch the number of clusters by using the elbow method to identify the best optimal value of k. DBSCAN randomly placed k-centroids, and there is no need to specify the number of clusters. We used the Python programming language under the Jupyter platform in this research. These steps and their workings are briefly defined below step by step, respectively, in Figure 6. The first step is to collect ransomware data of the most relevant multi-class ransomware families that only affect Windows-based operating systems in the form of.exe files. The data collection is done by third-party sources, as mentioned in the below sections, provided by the data engineer in CSV file format. We converted the CSV file into.xlsx format for ease of implementation in Python. Then, we check for the outliers by initializing threshold values like measuring the distance of the closest data point, which is greater than its nearest cluster identifier and is identified as an outlier in our dataset. The next step is to remove outliers and clean up the ransomware dataset.

Data cleaning may generate some missing values, and we have to fix them. Afterward, wrangling a ransomware dataset to create new variables, duplicate values are identified, and the variables are filtered. The next step is ransomware data's feature engineering. It features extraction using RFE and PCA, which use orthogonal transformation for the conversion of data into lower dimensional space like in between specific number range to maximize the variance of the dataset. Then shuffle the extracted feature dataset to apply machine learning-based KNN and density algorithm and get trained in the dataset with 70:30 as training and testing data. For zero-day attack detection under unsupervised ML, k-means will be used.

Data collection is the most critical step in solving controlled machine learning problems. Machine learning models often give outstanding results when asked to recall objects from their training, but sometimes they show inferior results when taken out of their comfort zone.

Table 2: Comparison of State-of-the-art Methods for analysis and protection of Ransomware Attack.

Technique/Methodology	Today Attacks Protection	Min. File Loss	Min. False Positive Rate	Inputs are Enough	Min. Latency and Max. Performance	Invul.	Offer Counter Measure
Cloud-based sandbox environmental method	Unknown	Unknown	Unknown	No	No	Yes	Yes
Cloud-based detection method	Unknown	Unknown	Unknown	No	No	Yes	No
Monitoring abnormal registry and file system activities	Unknown	No	Unknown	Yes	No	No	Yes
Monitoring process and IO events	Yes	No	No	No	Yes	No	No
Machine learned behavior-based method	Yes	Yes	No	Yes	No	No	No
Software defined networking-based method	Unknown	Yes	Yes	Yes	Unknown	No	Yes
Connection monitor and connection breaker method	Yes	Yes	Yes	Yes	No	Unknown	No
A large scale automated approach	Yes	Yes	Yes	Yes	Unknown	No	No
Automated dynamic analysis method	Yes	Yes	Yes	Yes	Unknown	Yes	No

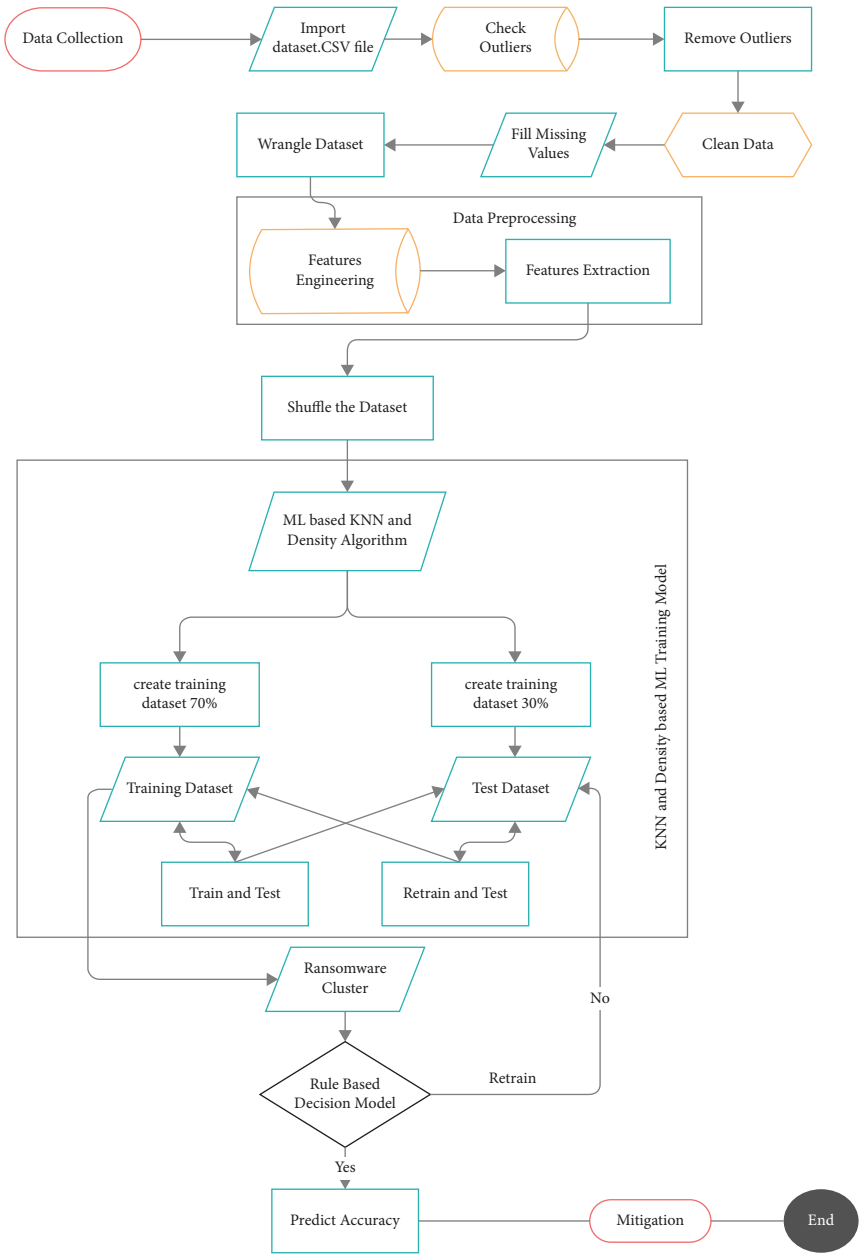


Figure 6: Advanced ransomware pre-attack detection algorithm for endpoint data protection.

ML algorithms require significant amounts of data to function. When they deal with millions and billions of pictures or text records, it is much more challenging to identify what causes an algorithm to perform poorly. Therefore, capturing a massive amount of information is not enough; by treating it with an ML model, the competitive results can be expected

Classifying ransomware into particular families is a challenging task due to the massive number of ransomware and multiple ransomware families. The ransom families' data used in this research experiment were identified on the ransomware Tracker website [21], the online ransomware data providing resource mainly used by the Internet-Service-Providers (ISPs), Computer Emergency Response Teams (CERT), and Law Enforcement-Agencies (LEA). When examining the currently being monitored ransomware families, our focus was only on the five-crypto families like Cerber, CryptoWall, Locky, TeslaCrypt, and TorrentLocker, including the available historical examples to confirm the variant history and malware changes in displayed code [26].

The ransomware tracking website (RTW) lists popular hosts for all ransomware families' distributions, payments, administrations, and management. The md5 value for all collected samples is uploaded to the VirusTotal Intelligence platform [36] to retrieve malware details rapidly. The download list is checked to ensure that only the Portable Executable PE (.exe) file's format is downloaded, with some other forms like DLL (Dynamic Link Library) being removed. This allows accurate comparisons with good software samples with a similar PE (.exe) format.

The dataset used to demonstrate the given algorithm is a making-dataset of malware images: visualizations and automatic classification documents [19]. This dataset contains 25 clusters of ransomware with several different family variants and 56 columns with a 65535 number of rows [6]. The deep convolutional neural network is used to detect malicious infections in IoT network through color image visualization [27].

3.1. Dataset Arrangement. The way datasets are organized plays a crucial role in controlled (supervised) classification. As we can see in Figure 7, there are 25 ransomware (malware) families, and each family has a different number of samples. Our classification method does not require malware debugging or code execution to related works. Besides, the texture of the image used for classification offers a lot more sustainable property in ambiguous technologies, especially for encryption. Finally, we apply our algorithm to a larger dataset that consists of 25 families in the corpus of ransomware malware than 9458 ransomware. After using a newly suggested algorithm, the final results show that our method offers better accuracy at lower computational costs. They encrypt various files on the victim's hard drives before requesting a ransom to get the files decrypted. Security-related media and some antivirus vendors quickly brandished this "new" type of virii.

3.1.1. Import Datasetcpsec. After Collecting and summarizing ransomware data, we convert them into two dataset

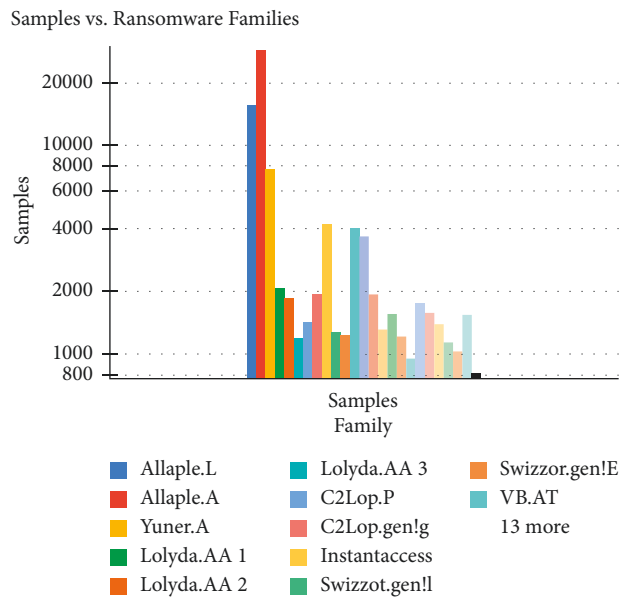


Figure 7: Ransomware (malware) dataset of 25 families [20].

formats of comma-separated file (CSV) and Microsoft Excel file format.xls or.xlsx. We require the dataset in three stages for the training of the machine learning model. One of them is training, the second is validation, and the third one is testing.

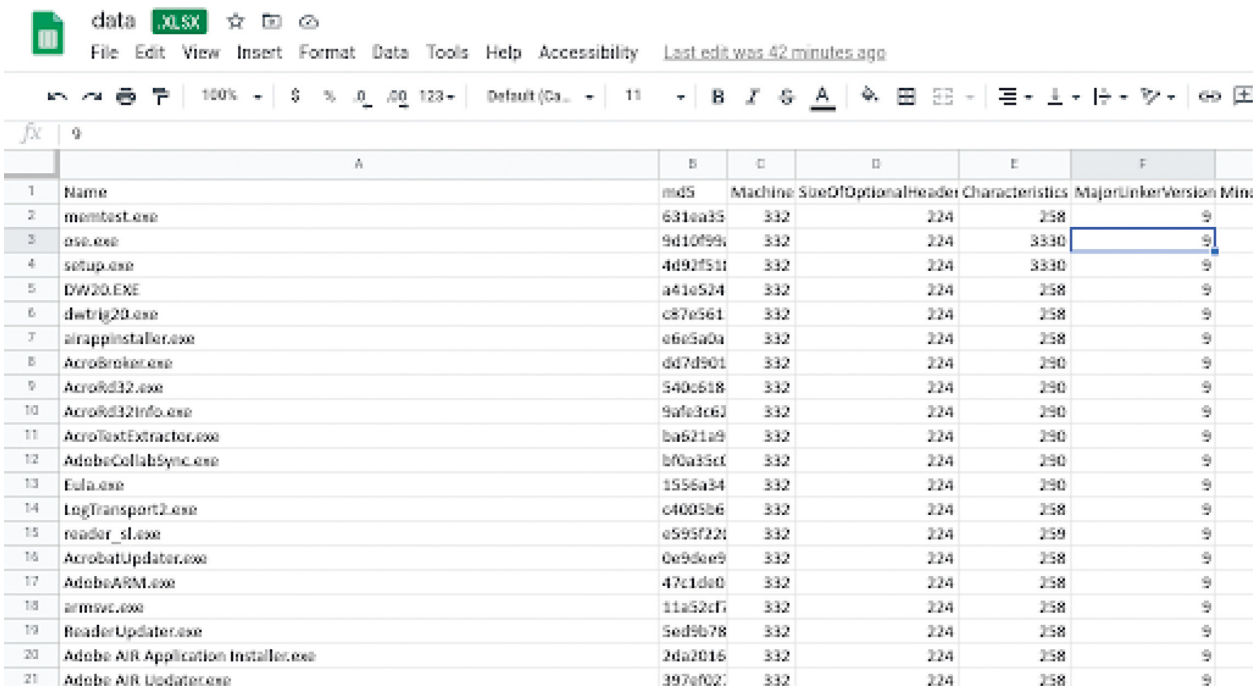
3.1.2. Excel File Data Reading. We need to import data from the excel file to pandas first. For this purpose, first, import the pandas' library using the commands. Then, we use the read_excel panda's method to read data from an excel file as shown in Figure 8.

The simplest way to call this method is to enter a filename. If the sheet name is not specified, the first sheet in the index is read.

Now, the read_excel method reads data from the excel file to the pandas DataFrame object. By default, pandas store data in DataFrames (df). Then, we store the DataFrame in a variable named df. Pandas have a built-in DataFrame.head () method, which we can use to return a few rows of our DataFrame quickly. If no arguments are given, the first five lines are displayed by default. We can verify that this aggregation exists by checking the number of rows in the aggregated DataFrame by calling the "shape" method, which displays the number of rows & columns.

The data.xls dataset file contains 65535 records and 57 columns. It can be helpful when counting the number of columns and rows compared with source records of datasets. Another tail method is also beneficial for seeing the last rows of a given dataset.

3.1.3. Training a Model. Data collection and modeling training is an iterative process, so we may need to retake the decisions taken while collecting data. The method includes data preprocessing, model training, and parameter setting.



	A	B	C	D	E	F
1	Name	md5	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion
2	memtest.exe	631ea35	332	224	258	9
3	osk.exe	9d10f99	332	224	3330	9
4	setup.exe	4d92f51	332	224	3330	9
5	DW20.EXE	a41e524	332	224	258	9
6	dwtrig20.exe	c87e561	332	224	258	9
7	airappinstaller.exe	e6e5a0a	332	224	258	9
8	AcroBroker.exe	dd7d901	332	224	290	9
9	AcroRd32.exe	540c618	332	224	290	9
10	AcroRd32Info.exe	9afe3c63	332	224	290	9
11	AcroTextExtractor.exe	ba621a9	332	224	290	9
12	AdobeCollabSync.exe	bf0a35c	332	224	290	9
13	Eula.exe	1556a34	332	224	290	9
14	LogTransport2.exe	c4005b6	332	224	258	9
15	reader_sl.exe	e595f22	332	224	259	9
16	AcrobatUpdater.exe	0e9dee9	332	224	258	9
17	AdobeARM.exe	47c1de0	332	224	258	9
18	armsvc.exe	11a52cf7	332	224	258	9
19	ReaderUpdater.exe	5ed9b78	332	224	258	9
20	Adobe AIR Application Installer.exe	2da2016	332	224	258	9
21	Adobe AIR Updater.exe	397ef02	332	224	258	9

Figure 8: Ransomware dataset in Microsoft excel sheet.

3.1.4. Data Preprocessing. In any machine learning process, the preprocessing of data is the step where the information is changed or encoded to bring it into a state such that machines can quickly analyze it. On the other hand, an algorithm can now easily interpret the data features. Cleaning data: is the most crucial step in any ML project. There are multiple and different statistical analyses and data visualization techniques in spreadsheet data can be used to examine a dataset to identify data cleaning operations that need to be performed. The primary focus of data cleaning is to identify, correct errors, and check redundant data to develop reliable datasets. These steps improve the quality of analytical training data and enable the best decision-making.

3.1.5. Fill Missing Values. In standard practice, data values are often missing from your dataset. This could have occurred during data collection or due to data validation rules but must be considered regardless of missing values.

MinMaxScaler: We have applied Min-Max-Scaler directly to the ransomware dataset for the normalization of input variables. We have used the default configuration and the scaling values, which ranged from 0 to 1. The MinMaxScaler instance was defined by applying standard hyper-parameters, then calling the fit_transform () to pass it to the ransomware dataset to update and upgrade it.

From sklearn. Preprocessing import MinMaxScaler = transforms the features by scaling each element to a limited range. Each feature estimator scales individually and translates so that it falls within the scope of a given training set, between [0, 1]. The transformation can be written as:

$$\begin{aligned} X\text{std} &= X - X \cdot \text{Min axis} \\ &= 0X \cdot \text{Max axis} \\ &= 0 - X \cdot \text{Min axis} \\ &= 0, \end{aligned} \tag{1}$$

$$X\_\text{scaled} = X\_\text{std} * (\text{Max} - \text{Min}) + \text{Min}, \tag{2}$$

where, Min, Max = feature_range.

Missing data handling by elimination rows: The elimination of rows is simple and may be the most effective technique. It fails if many objects lose value. If most of the features have missing values, these features themselves can be eliminated.

Evaluate missing values: If only a fair % of the weight is missing, we can also use a simple-interpolation method (SIM) to fill this value.

Duplicate values in the dataset: Datasets can contain data objects that are duplicates of one another. This can happen if you say the same person inserts the same values multiple times.

The data frame.nunique () is a Pandas function that returns a series with the number of individual values over the requested axis. If the axis value is 0, all numbers of unique observations above the index axis have been found. Setting the axis value to 1 finds the number of individual comments from the column axis. Excluded NaN values can be provided as a feature from the number of unique numbers.

3.1.6. Wrangle Dataset. Wrangling dataset is a significant part of any data science project. Data wrangling is the process by which a data scientist transforms "raw data" to

make it more fruitful for analysis, and this improves the quality of data to be analyzed. There are several preprocessing techniques to cover the entire process.

Zero-value: Count zero values, and decide what to do with these values.

Data-exploration: Search data types of features, unique values, and data descriptions.

Feature engineering and reshaping: Converts the raw data into more useful formats. Examples of engineering functions are one-time coding, aggregation, and clustering.

3.1.7. Exploratory Data Analysis. There is only one column and only one missing value. This research project loads the row with the lost value and sees how to handle them. Let us use pandas' df to check and confirm that our data matches the original information. The output of the describe() function summarizes the statistical data of all numerical columns. Types of statistical data are shown in Figure 9.

3.1.8. Feature Engineering and Feature Extraction. A feature is a property that can be individually measured or a characteristic of an observed phenomenon. A dataset can be seen as a collection of data objects, often referred to as a set of data, points, vectors, patterns, events, samples, observations, or things. Data objects are structured by several characteristics that capture the basic properties of an object.Features are commonly known as variables, fields, attributes, and dimensions. Various types of features can occur while handling data.

3.1.9. Principal Component Analysis. PCA is an unsupervised algorithm that generates linear combinations of original features. PCA is classified in the order of explained variance. PCA is used to reduce the dimensions of an extensive dataset, as we used 57 features in our ransomware dataset. The first principal component, PC1, describes the most significant deviation in your dataset, PC2 describes the second largest variation, etc.

Refer to Figure 10; principal component analysis of a data matrix extracts the dominant patterns in the matrix in terms of a complementary set of score and loading plots. It is the responsibility of the data analyst to formulate the scientific issue at hand in terms of PC projections, PLS regressions, etc. Ask yourself or the investigator why the data matrix was collected and for what purpose the experiments and measurements were made. Specify before the analysis what kinds of patterns you would expect and what you would find exciting. In the initial study, look for outliers and strong groupings in the plots, indicating that the data matrix perhaps should be "polished" or whether disjoint modeling is the proper course. Use the resulting principal components to guide your continued investigation or chemical experimentation, not as an end.Hence, we reduce the dimensions by limiting the number of principal components that must be retained based on the accumulated variance. We choose only critical components as necessary to keep achieving a cumulative described variant of 88.

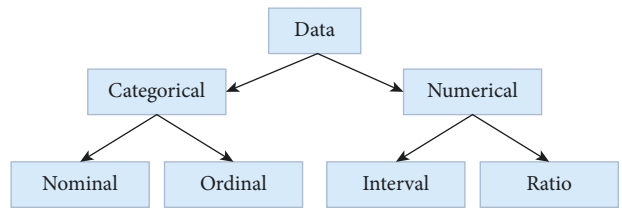


Figure 9: Types of statistical data.

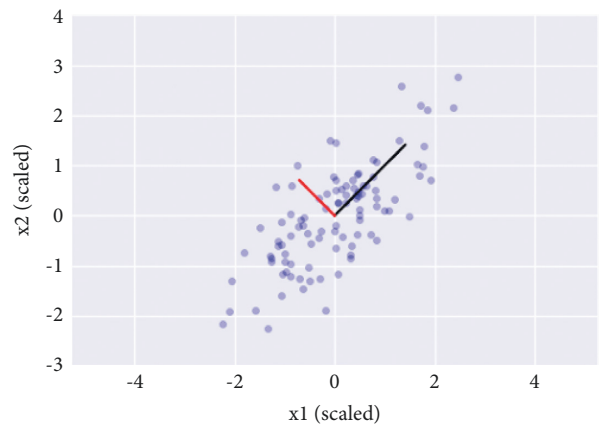


Figure 10: Principal component analysis.

3.1.10. Shuffle Data. In machine learning, we need to shuffle our learning data. The major problem we are facing is that, let us suppose, we have to train 70% of the data. But the given data frames contain the string, char, and integer values. During the testing of data, we have to test deals starting with either from A-R and then test S-Z values or start from 1–25 and then test the 25–100 values, respectively. This representative test does not look nice, or it is too necessary to perform this way. The main limitation is that we cannot train and test the same data, but given our current algorithm, there is nothing wrong with training and testing the same data values. There are multiple methods available for shuffling dataset values, but we have tried the given technique for data shuffling.

1st 25% - train 2nd 25% - train 3rd 25% - train 4th 25% - test Then, 1st 25% - test 2nd 25% - train 3rd 25% - train 4th 25% - train Then: 1st 25% - train 2nd 25% - test 3rd 25% - train 4th 25% - train Finally, 1st 25% - train 2nd 25% - train 3rd 25% - test 4th 25% - train.

We have done training and testing of all available data and then take the average of all values instead of simply shuffling the rows. For this purpose, we have created a Randomizing() function.

Another method we have used is the keyword "frac" argument which specifies the fragment of rows that should be returned to the random sample. "Frac = 1" means that all the rows should be returned in the random order. By using reset_index function with the parameter drop=true, the index can be reset with shuffling the data frame. Drop = True specifies the prevention of reset_index from generating an old column that contains an old index.

3.1.11. ML Algorithms. Many ML algorithms are used for the accuracy, classification, and prediction of ransomware over good ware. For this research project, Random-forest, DBSCAN, and KNN algorithms are being utilized to detect ransomware and zero-day attacks.

Random forest is a multi-level estimator that matches multiple decision tree classifiers in different subsets of the dataset and uses averages to improve forecast accuracy and control for over-fitting. The sample size is controlled with the max_samples parameter when bootstrap = True by default, and else the entire dataset is used to build all trees.

Random forest is not only an effective classifier but also useful for column selection. In this research project, we create a large, carefully constructed tree-set to predict target classes and use the usage statistics for each column to find the most informative subsets of columns. We made a large set (65535) of many flat trees (tree levels), and each tree was trained on a fraction (10 columns) of the total number of columns. The most predictable column is the column with the highest score.

Random-forest algorithm implementation. To discuss why we implement the random forest algorithm, let me tell you the following benefits of the random forest algorithm. Random forest and random forest classifiers are both algorithms that are often used to perform classification or regression problems. The random forest classifier can handle the missing values. If we have to use many trees in a forest, random-forest classifiers help avoid model overfitting problems. Random-forest creation pseudocode: 1. Select k features randomly from the "m" number of total features, where k << m. 2. Calculate the "d" number of nodes by using the best split-point concerning "k" features. 3. Use the best split of the given nodes into daughter node. 4. Repeat Steps 1–3 until you reach the "I" number of nodes. 5. Repeat Steps 1–4 by building a forest to generate "n" number of trees.

Random-forest prediction pseudocode: The trained random forest algorithm will be used to perform random-forest prediction 1-import test features and implements the rules of randomly generated decision trees to perform prediction as output and save it. 2-Calculate votes for all predicted outputs. A random forest algorithm will select the 3-Final prediction based on maximum votes. To make predictions using a trained random forest algorithm, we have to run the test feature through the rules of individual random trees.

Scikit-learn Python-based library calculates the importance of each node by using Gini importance for each decision tree to consider two child nodes only like a binary tree, as follows:

$$\begin{aligned} \text{Gini} &= 1 - i \\ &= Cpi2. \end{aligned} \tag{3}$$

The Gini formula uses class and probability to calculate the gini of each tuple on each node and find which branch is more likely to occur. In this formula, pi is used for the relative frequency of the class we are looking at in the dataset, and c is used for the number of classes in it. We also calculate entropy to determine how many nodes branch in a decision tree.

$$\begin{aligned} \text{Entropy} &= i \\ &= 1C - pi * \log 2(pi). \end{aligned} \tag{4}$$

Entropy is used for calculating the probability of a specific outcome to decide to check how to adjust nodes as a branch. Compared to the Gini index, entropy is more complex in the mathematical calculation because of its logarithmic function. Hence, we may say that random forest is the most practical algorithm with multiple types of datasets regarding regression or classification data. Random-forest is very easy to use and very fast to train data and find more accurate representations of decision trees.

(1) Training Dataset. At this stage, ML algorithms are trained to build the model. The model tries to learn the dataset and its multiple properties, which raises overfitting and underfitting problems.

Split training and testing dataset: Now, a dataset is used to test the hypotheses of our model. It remains unused and invisible until decisions are made about models and hyper-parameters. After that, the model is applied to the test data to accurately measure how the model is performed when applied to real-world data. The training set defines a subset of the given dataset used for the ML model training. On the other hand, a subgroup is part of a dataset used for testing an ML model. The machine learning model uses a series of test-set to predict outcomes. It is a common practice that the dataset is divided into 70:30 ratios — 80:20 ratio of total datasets. You can take 70% or 80% of the model's training data and leave the remaining 30% or 20% for testing data as shown in Figure 11.

We need to use the python library; the first row divides the array from the dataset into random subsets of train and test datasets. The second line contains four variables:

X_train - training data features.

X_test - testing data features.

Y_train - training data's dependent variables.

Y_test - the independent variable used for test data.

The train_test_split () function has four parameters, the first and second used for dataset arrays. The test_size function shows the test-set size. The test size can be 0.5 or 0.3, or 0.2.

These values define the distributed ratio between the training and test set. In the end, random_state parameters always fix input for a ransom generator to get the same output.

Collecting statistical information of data Pandas' python library offers several very convenient statistical display methods for our datasets. We can use the technique "describe" to get a statistical summary of a dataset. The method described shows the information given below for each column.

Total value or count Mean (concerning all features of the dataset).

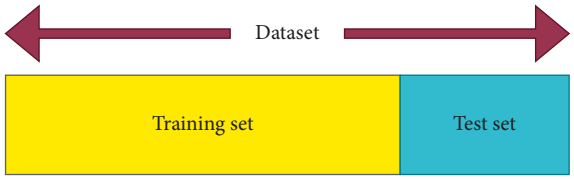


Figure 11: Dataset division into 70% training set and 30% test set.

Standard deviation (concerning all features of the dataset).

Min-max values (concerning all features of the dataset) 75%, 50%, 25% of quantity (concerning all features of the dataset).

This information is only calculated for numeric values. Prediction accuracy for ransomware detection. To check prediction accuracy, we apply data analysis techniques.

Analysis of data: there are two experimental stages: (1) the training stage and (2) the testing stage. All the machine learning algorithms described in the above section were trained and tested in the ransomware dataset [3]. The dataset is divided as follows: 70% for the training stage and 30% for the testing stage. The variables considered in the experiment are as follows:

Accuracy test (invisible data accuracy estimate): Accuracy is an indicator for the evaluation of classification models. Accuracy is the fraction of the predictions that our model gets right. Now we may define accuracy as follows:

$$\text{Accuracy} = \text{number of correct predictions total number of predictions.} \tag{5}$$

For binary classification, accuracy can also be calculated using true [T], false [F], positive [P], and negative [N] as follows:

$$\text{Accuracy} = TP + TNTP + TN + FP + FN. \tag{6}$$

(2) F1-score (harmonic mean for recall and precision, as given below).

The F1 score is a harmonious mean between recall & precision. The range for the F1 score is [0–1]. High-precision but lower recall gives you good accuracy but misses many instances that are not easy to classify. The higher the F1 score, the betterness of our model concerning performing. Mathematically, we may express as

$$F = 2PPVTPRPPV + TPR. \tag{7}$$

(3) Recall (True-positive-rate, as given below) True-positive-rate can be calculated as TP/(FN + TP). A TPR corresponds to the proportion of positive data points (PDP) correctly considered positive for all positive data points.

$$\begin{aligned} TPR &= \text{True Positive True Positive} \\ &= \text{False Negative.} \end{aligned} \tag{8}$$

(4) Precision (Positive-predictive-value, as given below). Precision can be calculated by dividing the number of true-positive results by the true-positive results predicted by the classifier.

$$\begin{aligned} PPV &= \text{True Positive True Positive} \\ &= \text{False Negative.} \end{aligned} \tag{9}$$

The results, accuracy, and fetching of the F1 classification measure are calculated with the Classification_report () function using the python library sklearn metrics library.

4. KNN and Density

K-Nearest Neighbors and Density-Based Algorithm (KNN) is a branch of supervised machine learning. KNN is very user-friendly for implementation. KNN also can perform complex classification tasks efficiently. The KNN is a non-parametric technique used for data classification & regression. The source contains the next k nearest training data for learning from feature space. The results depend on whether the classification is used in KNN or regression. It is a lazy learning algorithm because there is no specific training phase. In KNN, all training data are used when a new data point or instance is classified. KNN is a nonparametric training algorithm that does not require a dany population being analyzed by meeting specific parameters or assumptions; that is why it is considered a handy feature as most real-world data do not follow only theoretical assumptions like linear separation and vice versa. In this research, KNN is implemented with Python's popular library scikit-learn and uses some of its utilities. Figure 12 shows the closest points with the shortest distance.

The basic purpose of the KNN algorithm is to provide the easiest supervised machine learning algorithms. KNN measures the distance from the new data point to all other training data points. Any type of distance can be applied, like Euclidean, Manhattan, and so on. Then choose the closest data point, K, and K must be an integer. At last, KNN allocates the data points to the class with the most K points. Let us apply the KNN algorithm using a simple example. The task is to choose the class of new data points by considering X in the Pink color and the green color as a class. The coordinates of the data points are $x = 45$, $y = 50$.

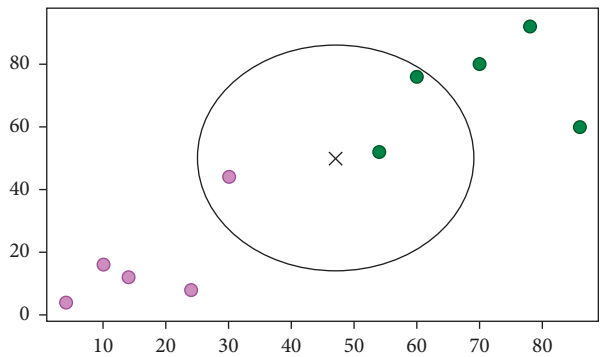


Figure 12: Closest points with the shortest distance.

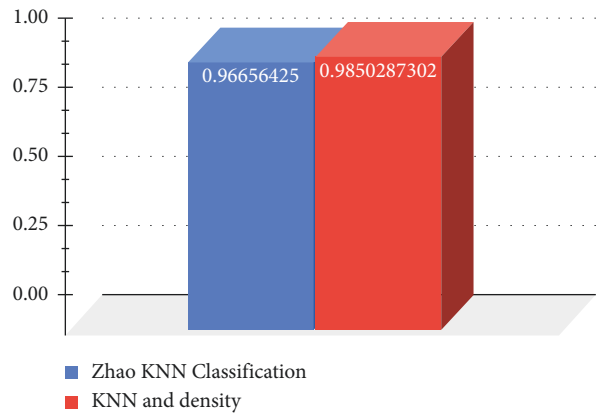


Figure 13: Zhao KNN classification vs. our KNN and density classification results.

Table 3: Classification report.

	Precision	Recall	F1-Score	Support
0	0.99	0.98	0.98	4805
1	0.99	0.99	0.99	8302
Accuracy			0.99	13107
Macro avg.	0.99	0.98	0.99	13107
Weighted avg.	0.99	0.99	0.99	13107
Final accuracy: 0.9846060814051052				

Let us suppose the k value is 3. KNN first measures the distance of point X from all other data points, then adopts the three closest points with the smallest distance concerning point X, as shown below. The three nearest points are surrounded. Now KNN will allocate a new point to a class that has a majority of the three closest points. As shown in Figure 13, there are two of the three most relative dots in the green color's class, and one belongs to the Blue color's class. As a result, the new data points are allocated to the green color's class.

5. Results

After successfully implementing KNN and density, we got the following fruitful results, as shown in Table. 3. The classification report shows the average performance as 0.99 w.r.t f1 sources, precision, support, and recall. This report also shows the accuracy by values, near about 99%. By using hyper-parameters-tuning, we have improved the performance by about 15% as compared to previously calculated adoptive clustering KNN and density-based algorithm's accuracy, as described in previous papers [26, 28]. Zhao also used KNN classification and random forest techniques with some additional features to achieve more accuracy of 97%.

Table 4: ROC score.

Precision	Recall
Best Leaf_Size	5
Best p	7
Best n_neighbors	1

The KNN and density algorithm recognize the classifiers of different shapes, sizes, densities, internal variants, and a few other features and are useful to remove noise and outliers automatically. The effectiveness of this algorithm is tested on multi-dimensional ransomware data. The key advantage of this algorithm is that it performs the classification process w.r.t K nearest neighbor. This algorithm is more suitable for incremental processes because this model is simple and can be efficiently working for large amounts of data processing. The Receiver Operating Characteristics (ROC) score shows the best results, as described in Table 3 that is, 0.9846060814051052.

According to these results, using the Grid Search method, the best leaf size = 5, where $p = 1$ shows the optimal distance technique used is Manhattan, and the best value for the K nearest neighbor is number 7, as described in Table. 4. The overall accuracy after testing the ROC score is about 0.98962542, which is about 99% as shown in Figure 13.

6. Conclusion

This research demonstrates the analysis and implementation of windows-based ransomware pre-attack detection using PE (.exe) files format. Many windows based on ransomware detection techniques and algorithms are available. These detection algorithms are working very effectively. But with the continued growth of zero-day attacks and the new generation of ransomware families' attacks on endpoint systems, there is a need to improve the malware detection techniques and algorithms. In this research, for the detection of ransomware in supervised ML, we used KNN and density-based algorithm and random forest with outstanding accuracy results of 98% and 99%, respectively. On the other hand, to detect zero-day attacks that lie under unsupervised ML, we implemented k-means and DBSCAN clustering algorithms to detect zero-day attacks. For this purpose, we need to fetch the number of clusters by using the elbow method to identify the best optimal value of k. DBSCAN randomly placed k-centroids, and there is no need to specify the number of clusters. Still, after analysis and implementation, the results show that K-means clustering leads to more efficient results than DBSCAN for many ransomware datasets. In unsupervised ML and zero-day attack detection,

k-means show better detection results. In supervised ML and ransomware detection, the random forest algorithm shows more accuracy in results than KNN and density-based algorithms. There are still more improvements required in ML algorithms to detect ransomware and zero-day attacks for multi-dimensional features.

7. Future Work

In the future, there are many opportunities for data scientists and researchers to get more improvement in this research concerning increasing the ransomware detection accuracy by using more multi-dimensional data features. Results can be improved by using different advanced ML algorithms with the latest ransomware datasets to gain highly accurate detection.

Data Availability

No datasets have been used or refereed in thise article. However, preliminary data can be found at Malwarebytes (2017): Cybercrime Tactics and Techniques, Report: https://www.malwarebytes.com/resources/webinars/cybercrime-tactics-techniques-q1-2017.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] C. Moore, "Detecting ransomware with honeypot techniques," in *Proceedings of the 2016 Cybersecurity and Cyberforensics Conference (CCC)*, pp. 77–81, IEEE, Amman, Jordan, August 2016.

[2] A. Kharraz and E. Kirda, "Redemption: real-time protection against ransomware at end-hosts," in *International Symposium on Research in Attacks, Intrusions, and Defenses,* Springer Cham, Switzerland., Europe, 2017.

[3] L. Karthikeyan, G. Jacob, and B. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, no. 4, Pittsburgh, PA, USA, July 2011.

[4] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 646–656, 2013.

[5] E. Gandotra, D. Bansal, and S. Sofat, "Tools techniques for malware analysis and classification," *International Journal of Next-Generation Computing*, vol. 7, no. 3, 2016.

[6] A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K. R. Choo, "Detecting crypto-ransomware in IoT networks based on energy consumption footprint," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 4, pp. 1141–1152, 2018.

[7] M. A. Habib, M. Ahmad, S. Jabbar, S. H. Ahmed, and J. J. P. C. Rodrigues, "Speeding up the internet of things: Leaiot: A lightweight encryption algorithm toward low-latency communication for the internet of things," *IEEE Consumer Electronics Magazine*, vol. 7, no. 6, pp. 31–37, 2018.

[8] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, 2019.

[9] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.

[10] S. M. Muzammal, M. A. Shah, H. A. Khattak, and S. G. S. S. K. Jabbar, "Counter measuring Conceivable security threats on Smart Healthcare devices," *IEEE Access*, vol. 6, Article ID 20722, 2018.

[11] J. Landage and M. P. Wankhade, "Malware and malware detection techniques: a survey," *International Journal of Engineering Research and Technology*, vol. 2, no. 12, pp. 2278–0181, 2013.

[12] M. Conti, T. Dargahi, and A. Dehghantanha, "Cyber threat intelligence: challenges and opportunities," in *Cyber Threat Intelligence*, pp. 1–6, Springer Cham, Switzerland., Europe, 2018.

[13] Z. Zhao, J. Wang, and J. Bai, "Malware detection method based on the control-flow construct feature of software," *IET Information Security*, vol. 8, no. 1, pp. 18–24, 2014.

[14] S. Cesare, Y. Xiang, and W. Zhou, "Control flow-based malware VariantDetection," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 4, pp. 307–317, 2014.

[15] C. T. Lin, N. J. Wang, H. Xiao, and C. Eckert, "Feature selection and extraction for malware classification," *Journal of Information Science and Engineering*, vol. 31, no. 3, pp. 965–992, 2015.

[16] J. B. Park, K. S. Han, T. G. Kim, and E. G. Im, "A study on selecting key Opcodes for malware classification and its Usefulness," *Journal of KIISE*, vol. 42, no. 5, pp. 558–565, 2015.

[17] M. M. Ahmadian and H. R. Shahriari, "2entFOX: a framework for high survivable ransomware detection," in *Proceedings of the 2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, pp. 79–84, IEEE, Tehran, Iran, September 2016.

[18] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, 2017.

[19] S. Ruan, R. Mehmood, A. Daud, H. Dawood, and J. S. Alowibdi, "An adaptive method for clustering by fast search-and-find of density peaks: adaptive-dp," in *Proceedings of the 26th International Conference on World Wide Web Companion*, pp. 119–127, Perth, Australia, April 2017.

[20] M. Ahmad and S. A. F. G. Jabbar, "A sustainable solution to support data security in high Bandwidth Healthcare Remote locations by using TCP CUBIC Mechanism," *IEEE Transactions on Sustainable Computing*, vol. 5, no. 2, pp. 249–259, 2020.

[21] B. Jethva, "A New Ransomware Detection Scheme Based on Tracking File Signature and File Entropy," (Doctoral Dissertation), B.Eng, Gujarat Technological University, Gujarat, India, 2019.

[22] M. M. Hasan and M. M. Rahman, "RansHunt: a support vector machines based ransomware analysis framework with integrated feature set," in *Proceedings of the 2017 20th International Conference of Computer and Information Technology (ICCIT)*, pp. 1–7, IEEE, Dhaka, Bangladesh, December 2017.

[23] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, pp. 1–7, Pittsburgh, PA, USA, July 2011.

[24] J. Liu, Z. Liu, C. Sun, and J. Zhuang, "A data transmission approach based on ant colony optimization and threshold proxy re-encryption in wsns," *Journal of Artificial Intelligence and Technology*, vol. 2, no. 1, pp. 23–31, 2022.

[25] X. Zhang and G. Wang, "Stud pose detection based on photometric stereo and lightweight YOLOv4," *Journal of Artificial Intelligence and Technology*, vol. 2, no. 1, pp. 32–37, 2022.

[26] M. A. Mirza, M. Ahmad, M. A. Habib, N. Mahmood, C. M. N. Faisal, and U. Ahmad, "CDCSS: cluster-based distributed cooperative spectrum sensing model against primary user emulation (PUE) cyber attacks," *The Journal of Supercomputing*, vol. 74, no. 10, pp. 5082–5098, 2018.

[27] F. Ullah, H. Naeem, S. Jabbar et al., "Cyber security threats detection in internet of things using deep learning approach," *IEEE Access*, vol. 7, Article ID 124379, 2019.

[28] B. Shi, L. Han, and H. Yan, "Adaptive clustering algorithm based on kNN and density," *Pattern Recognition Letters*, vol. 104, pp. 37–44, 2018.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 9291971, 27 pages
https://doi.org/10.1155/2022/9291971

Review Article

Overview of Image Inpainting and Forensic Technology

Kai Liu,[1,2] **Junke Li**,[2,3,4] **and Syed Sabahat Hussain Bukhari**[5]

[1]*College of Educational Science, Qiannan Normal University for Nationalities, Duyun 558000, Guizhou, China*
[2]*Key Laboratory of Machine Learning and Unstructured Data Processing of Qiannan, Duyun 558000, Guizhou, China*
[3]*School of Computer and Information, Qiannan Normal University for Nationalities, Duyun 558000, Guizhou, China*
[4]*Key Laboratory of Complex Systems and Intelligent Optimization of Guizhou, Duyun 558000, Guizhou, China*
[5]*College of Computer Science, Neijiang Normal University, Neijiang 641100, China*

Correspondence should be addressed to Junke Li; ljk2006ljk@163.com

Received 23 September 2021; Revised 27 October 2021; Accepted 30 March 2022; Published 20 May 2022

Academic Editor: Farhan Ullah

At present, digital image is the most significant information carrier on the Internet, which provides convenience for people to exchange information. With the progress of image processing theory, digital image inpainting technology has also developed rapidly. It not only enhances the image expression but also provides tools for image forgery. Misusing the tools may trigger a crisis of trust in the image. To eliminate the negative impact of image forgery, many scholars have conducted in-depth research on it and proposed a series of detection methods called image forensics for institutions or communities to identify such forged images. Current forensic technology still has many limitations, such as relying on specific features and data distribution which makes forensic technology lag behind image inpainting technology. So far, academia still lacks a unified understanding of image forgery and detection technology, and the research architecture of inpainting and forensic technology is not clear. This article reviews the development of image inpainting and forensic technology and systematically summarizes and scientifically classifies the current research work. Finally, the forensic technology is summarized and the future research direction is prospected to guide subsequent scholars to further promote the progress of image forensic technology.

1. Introduction

As early as decades after the birth of photos, image processing technology has emerged [1]. In the film age, it is time-consuming and hard to modify images. However, with the development of society and the progress of science and technology, the popularization of digitization and informatization, the rapidity of digital images have brought many conveniences to various industries, including news photography. While using images, people often need to modify the images appropriately to make up for the defects leftover from the camera. For example, photo processing such as removing red eyes, making up for insufficient exposure, color scale adjustment, and sharpening enhancement is used to enhance the expressiveness of images and make digital images clearer and beautiful. Today, with the highly developed Internet, digital images have become the significant carrier of information on the Internet. People can publish the captured digital images at any time by using social network platforms. These published digital images convey important information to people. At the same time, news image is no longer the patent of news photographers but has become people's daily skill.

Limited by the carrier of news image, it cannot be released according to the original size at the time of collection on some occasions, so the image needs to be compressed. With the development of digital image processing technology and the continuous emergence of more and more image processing and editing software (such as Photoshop, AutoCAD), the modification, editing compression, and storage of digital images become very simple and imperceptible. While people enjoy the convenience of the digital images from networks, it also takes opportunities to some individuals and institutions with ulterior motives. For various purposes, they unintentionally or deliberately, or even maliciously, release digital images that are beneficial to

themselves to deceive and confuse the public. Therefore, people also begin to face the great test of image authenticity judgment especially in the field of news photography. Even in the news reports of some well-known media at home and abroad, there are many fake "news images," which often become the fuse of network rumors. For example, the faked image of Indian Prime Minister Narendra Modi inspecting the flood disaster in southern India by plane in the official news released by the Indian information and Information Bureau in 2015 makes this fraud widely questioned by the public [2]. Chinese Sichuan officials publicly apologized for publishing synthetic images [3]. In recent years, face-changing technology represented by Deepfakes [4] and Zao app [5] has begun to rise on the Internet and provide mass face-changing entertainment services, which has posed a significant challenge to the judicial system. The problem of image tampering has widely existed in various fields, such as news media, fashion magazines, social media, online auction websites, and academic research journals. [6]. When the forged image is applied in the process of the monitoring system, publication, crime investigation, and court proof as evidence, its authenticity identification is very significant.

The authenticity of the image plays a more significant role in the quality of the event. For example, whether the authenticity of the moon landing photos in the U.S. Lunar Landing Plan is real or not [7]. For another example, seek the suspect who assassinated U.S. President Kennedy [8]. To make digital images truly reflect objective facts and news reports more authentic, how to distinguish the authenticity of digital images has become a key problem to be solved. The digital image processing technology is a "double-edged sword" which is originally produced to facilitate the use of images. Under normal circumstances, image processing not only does not affect the information of image exchange but also makes the information contained in the image clearer and more accurate. This "modification" is acceptable to people. For counterfeiters who maliciously tamper with images, forge scenes, and distort facts, this "tampering" is unacceptable. The reasons for forging images are complex, including political reasons, interest-driven, expressing opinions, and simple entertainment. In general, modifying the image can convey a certain intention or attract more people's attention, and finally achieve a certain purpose. Some image tampering behavior will bring adverse effects, introduce unstable factors to society, and cause immeasurable serious consequences. Therefore, how to identify the authenticity of image content which is also called digital image forensics is an urgent issue related to people's production and life, national and social stability, and so on. Malicious image inpainting causes digital image forensics. To avoid image forensic recognition, malicious attackers have developed anti-forensic-related technologies. Their relationship between them is listed in Figure 1, and the structure of this context is based on it. At present, some summaries of image inpainting [9, 10] and forensics [11–14] are still lacking in this field. There is only an urgent need to systematically sort out and scientifically summarize and classify the existing works to promote the research in this field.

The rest of this paper is organized as follows: Section 2 introduces digital image inpainting technology. Section 3 reviews digital image forensic technology. Section 4 discusses the game of image forgery and detection technology. Section 5 summarizes the current datasets for forgery research. Finally, Section 6 provides the summary and prospects of the work.

2. Digital Image Inpainting

With the advent of the Internet era, multimedia presents a colorful virtual world for people, in which digital images carry a huge amount of information and become an indispensable role. Therefore, it has attracted many scholars in companies and academia to study digital images. With the increasing demand for the repair of old and damaged photos and the coloring of black-and-white photos, there is also research on image inpainting in the field of image research. At present, according to the evolution of image inpainting technology, it can be divided into image editing inpainting, image synthesis inpainting, and deep learning-based image inpainting, as shown in Figure 2.

2.1. Image Editing Inpainting. There are many kinds of image editing operations, such as splicing, deletion, copy-move, blurring, contrast enhancement, compression, scaling, median filtering, and adding noise. Splicing refers to cutting an object from another image and inserting it into the image to be edited. Deleting operation refers to deleting one or more objects in the image to be edited. Copy-move is a common image inpainting method. It hides important information or forges false scenes by copying an area in the image, scaling, rotating, and pasting it to other locations of the same or other images. Usually, to achieve the purpose of attack, attackers will perform preprocessing operations such as rotation and scaling on the editing area, or smooth the edge of the editing area. These operations will change the image content to varying degrees, among which splicing, deletion, and copy-move are the most common image editing operations. Through image editing operation, the tension of image performance can be enhanced, black-and-white photos can be colored, and the original appearance of damaged photos can be restored.

2.2. Image Synthesis Inpainting. Before the development of deep learning technology, image inpainting was mostly realized by synthetic inpainting algorithm, which was deeply studied by many scholars. Image synthesis inpainting refers to the process of visually filling the damaged area of the image to restore the integrity of the image, and it is difficult for the observer to detect the damaged area afterward. The main idea of image synthesis inpainting is to use the information of multiple reference areas of the image to fill or synthesize the area to be restored and to keep the structure and texture of the image before and after the restoration as consistent as possible. Criminisi et al. [15] proposed an inpainting method based on image block texture synthesis, which used a block-based sampling process to achieve the filling of texture and

Figure 1: The structure of context.

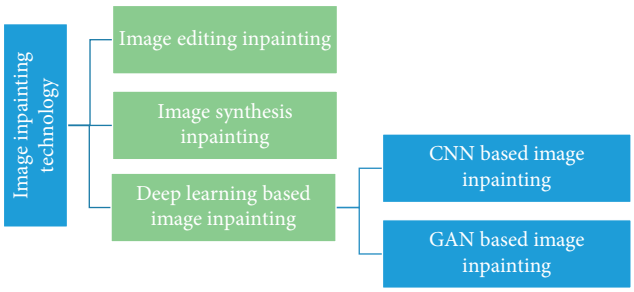


Figure 2: Classification of image inpainting technology.

structure information. Its process to repair the image is shown in Figure 3. Let the repaired area be O and the reference area be F as shown in Figure 3(a). It first calculates the priority of each point on the boundary ($\delta\Omega$) of the repaired area in the image to obtain the highest priority point p and then selects the image block ψp centered on p as the current block to be repaired as shown in Figure 3(b). Second, the image block Ψ_q' that most matches ψp is searched in the reference area of the image as shown in Figure 3(c). Again, the part to be filled in ψ_p is replaced with the pixel at the corresponding position in the best matching block ψ_q' as shown in Figure 3(d). Finally, the edge is updated and the above process is repeated until the O is repaired. At present, most image inpainting algorithms improve the algorithm in the [15]. For example, the algorithm in [16] has improved the fast priority dropping and visual continuity of [15]. The algorithm in [17] has improved the structural discontinuity and texture inconsistency of [15]. Jiao et al. [18] and Su et al. [19] have improved computing efficiency and the visual effect of repair respectively.

2.3. Image Inpainting Based on Deep Learning. At present, image synthesis inpainting methods mainly fill the missing part according to the statistical information of the residual image content, that is, each time a pixel of the missing part is constructed using the similarity principle to maintain its consistency with the surrounding pixels. However, when the missing part is large or complex, the traditional image inpainting methods often fail because they cannot obtain higher level semantic, texture, and other deep features from the original image. In 2006, Hinton [20] first put forward the relevant viewpoints and concepts of deep learning to learn the deep feature expression of things by creating a multilayer network. With the rapid progress of machine learning technology, various neural networks emerge, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs). Among them, CNNs and GANs have been shown their effectiveness for image-related tasks. CNN is a kind of feedforward neural network with convolution operation and deep structure. Typical CNN architectures include AlexNet [21], ResNet [22], VGGNet [23], Xception [24], GoogleNet [25], DenseNet [26], and SENet [27]. CNN is one of the most typical architectures of deep learning and has achieved certain results in image inpainting [28, 29].

The current advanced image inpainting methods mainly include two categories: (1) the methods based on deep CNN proposed by [30, 31] of NVIDIA company; and (2) the method based on GAN proposed by [32].

2.3.1. CNN-Based Image Inpainting. In the early days when deep learning was applied to image inpainting, Pathak et al. [30] combined the encoder and decoder structure with CNN to design a context coder in 2016 to solve the problem that CNN depended on a large number of labeled data and the semantic understanding problem contained in the image to be completed. It used multiple convolution layers in the codec structure, which can complement the semantic-sensitive content of the image scene in a parametric way. Therefore, it can synthesize high-level features in a large spatial range and provide better features for the inpainting method based on the nearest neighbor. This method realizes the application of CNN in image inpainting for the first time.

In order to meet the visual and semantic rationality of the repaired image, Zeng et al. [33] proposed a pyramid-context encoder network called PEN-NET for image inpainting. Based on U-Net structure, they used deep generative models to restore an image by encoding contextual semantics from full-resolution input and by decoding the learned semantic features back into images. To solve the above problem, Liu et al. [31] proposed to apply the partial convolution algorithm to the image inpainting problem and used some convolution layers to replace all convolution layers in the U-Net structure to complete the missing part of

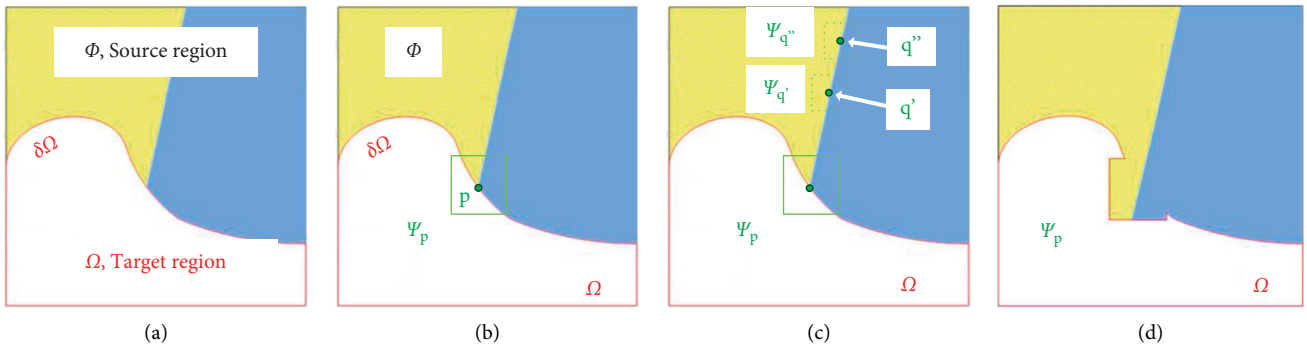


Figure 3: The process of algorithm of [15].

the image. This method only inputted the effective pixels of the unmissed area in the picture and added a mask update process after each layer, which realized the image inpainting that was independent of the size of the initial missing part and did not require any postprocessing. This is the first time CNN has been used to complete the missing irregular shapes in images. Yu et al. [34] optimized partial convolution by using gated convolution. They abandoned the hard mask of partial convolution and optimized it into a flexible mask which was automatic learning from data to better complete the images with irregular shapes of incomplete parts. Gated convolution network accelerated the training speed and improved flexibility through flexible mask and dynamic feature selection mechanism. It can improve the repair effect under the condition of free-form masks and user-guided input.

To repair the wrong texture generated by content-aware filling and apply image inpainting to higher resolution images, Yang et al. [35] proposed a high-resolution image inpainting method of multiscale neural patch synthesis. There are two networks in its structure: one is a content generation network and the other is a texture generation network. The former network directly generates images and infers the possible contents of missing parts. The latter network enhances the output texture of the content network. The obvious disadvantage of this algorithm is that its execution process consumes a lot of computing resources and takes a long time to generate the results. In addition, this method does not extend the application to the image inpainting with irregular missing areas.

For the image inpainting problem with poor processing edge position effect of the missing part of the image, DFNet (deep fusion network) proposed by Hong et al. [36] can harmoniously integrate the completed image area with the existing incomplete image, especially in terms of pixel transition and semantic structure consistency of the boundary area, so as to better realize image inpainting.

Both Yan et al. [37] and Wang et al. [38] are dedicated to repairing the central part, without considering the continuity between pixels. From the semantics perspective, they did not consider the continuity of features, resulting in color fault or line fault. For solving the above problem, Liu et al. [39] proposed a coherent semantic attention (CSA) model, which has a rough repair step and a fine repair step. The network of rough repair used a U-net structure. The repair speed is fast and the effect is good.

To solve the problem that the mean and variance shifts caused by full-spatial feature normalization (FN) limit the image inpainting network training, Yu et al. [40] proposed a spatial region-wise normalization named region normalization (RN). It divides spatial pixels into different regions according to the input mask and computes the mean and variance in each region for normalization. Two kinds of RN for image inpainting network are introduced: (1) Basic RN (RN-B), which normalizes pixels from the corrupted and uncorrupted regions separately based on the original inpainting mask to solve the mean and variance shift problem. (2) Learnable RN (RN-L), which automatically detects potentially corrupted and uncorrupted regions for separate normalization and performs global affine transformation to enhance their fusion.

In summary, Table 1 lists the categories, models, structures, advantages/disadvantages, and the best performance in the article of the above deep CNN-based image inpainting methods.

2.3.2. GAN-Based Image Inpainting. GAN is a deep learning method proposed by Goodfellow et al. [41], which uses the mutual game of generative models and discriminant methods to produce better output results. Its network structure is shown in Figure 4, and it includes two parts: generator G and discriminator D. G takes random noise as input and generates some fake samples similar to real ones, whereas the D has to learn to determine whether samples are real or fake. If D judges correctly, it is necessary to adjust the parameters of G to make the generated false data more realistic. If the judgment of D is wrong, the parameters of D are adjusted to avoid making a wrong judgment next time. GAN directly performs feature extraction and image generation on samples by considering global information. Therefore, the generation time of the target is shorter, and the speed is relatively fast, and the generated images are more realistic. These make GAN an extremely high-performance generation method, which has become the basis of many image inpainting algorithms.

With the continuous development of GAN, a series of GAN models has been developed, which correspond to

Table 1: Comparison of deep CNN-based image inpainting methods.

Category	Model	Structure	Advantage/disadvantage	Perf.
CNN	Traditional CNN	Convolution layer; subsampling layer; full link layer	Efficient processing; automatic feature selection; good classification effect.	
Deep CNN	Context encoder	Encoder and decoder structure combined with convolutional neural network	Unmarked image, irregular shape of missing area; can obtain the semantic information of the image; good repair effect, high processing speed.	17.58 dB (PSNR)
	PEN-NET	U-net structure with deep generative models	Visual and semantic coherence for image inpainting; fast convergence in training	9.94 (L1 loss)
	Partial convolution	Applying the partial convolution to replace all convolutions in the U-Net structure.	Stable performance; suitable for image inpainting with irregular shape.	19.04 dB (PSNR)
	Gated convolution	Using gated convolution to optimize partial convolution	Improved the flexibility; with mask and guiding input, the repair effect can be improved.	1.6% (mean L1 error)
	High-resolution	Content network and texture network	Consume a lot of computing resources and take a long time; only the patch in the picture is used instead of the data in the whole dataset.	18.00 dB (PSNR)
	Shift-net	U-net-based architecture	Image center restoration; continuity between pixels is not considered.	26.51 dB (PSNR)
	CSA	U-net architecture	Fast speed; good repair effect	26.54 dB (PSNR)
	RN	Encoder and decoder structure	Good repair effect	28.16 dB (PSNR)

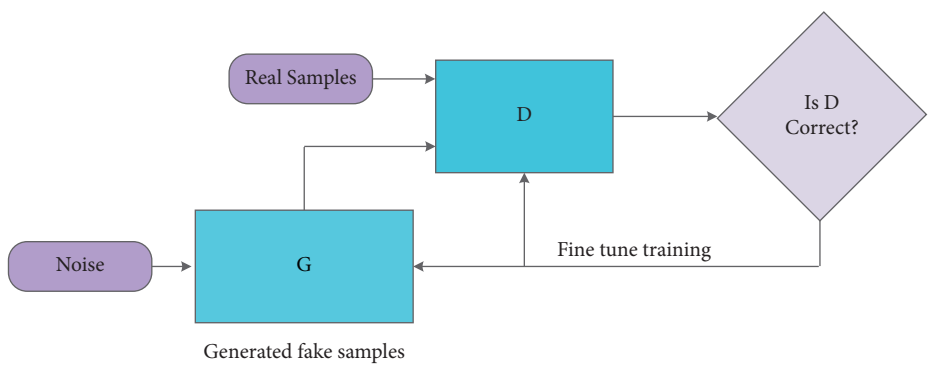


Figure 4: The structure of GAN.

different computing performance and application requirements. Generally, GANs can be divided into two categories: structure change-based GAN and loss function-based GAN [42]. Among them, structure change-based GAN mainly covers several categories: conditional GAN, deep convolution GAN, combined GAN, self-attention GAN, style-based GAN, etc. The loss function-based GAN mainly covers several categories, such as least-squares GAN and Wasserstein GAN.

(1) Conditional GAN. In order to solve the problem that the original GAN has almost no constraints on the generator, Mehdi et al. [43] imposed constraints on it and proposed conditional GAN (CGAN) to make the network generate samples in a given direction. In the training process of CGAN, constraints were added to both generator and discriminator to guide the generation of data. But CGAN did not solve the problem of unstable training. CGAN is mainly used in image enhancement, such as image generation and image inpainting [44].

In the original GAN, the input of the generator G is generally a continuous single random noise Z, and the generator G will perform highly entangled processing on Z. In this case, we cannot specify the samples generated by G. For this, Chen et al. [45] proposed interpretable representation learning by information maximizing GAN (info-GAN). It added a latent code c based on GAN to control the attributes of the generated image by controlling the variables of the corresponding dimensions. Info-GAN can make the network learn interpretable features and generate specified images, but the mutual information between data needs to be calculated in the training process, which increases the training complexity.

(2) Deep Convolution GAN. The image inpainting method based on context encoder predicts the missing area through context, and the effect of the generated image is not ideal. Therefore, the deep convolution GAN model (DCGAN) in Yu et al. [46] is proposed by combining traditional CNN with GAN and applying traditional CNN to generators and

discriminators. It used convolution layer instead of full connection layer and fractional step convolution instead of the upper sampling layer, and its generator and discriminator were both using batch normalization layer. This method has no artifact in the inpainted area, the image edge is clearer and has achieved good results in solving the problem of inpainting large missing areas.

In 2017, the semantic image inpainting method based on DCGAN proposed by Yeh et al. [32] gave a better solution. It used DCGAN as the basic model structure and combined the semantic repair method. Then, it adjusted and improved the loss function of DCGAN in a targeted manner. Similar to the context encoder, this method used unlabeled data during training and testing. But compared to the context encoder, one of the main advantages of this method is that no mask is required for training. Because the training process does not rely on masks, this method is more suitable for missing regions of arbitrary structure than context encoders.

Although DCGAN is used for missing areas of arbitrary structure, it is difficult to cope with image inpainting in many different scenes. To improve the coordination among the inpainting area, the surrounding area, and the global and make the inpainting model meet the image generation tasks of high-resolution, irregular missing areas and multiple scenes at the same time, Iizuka et al. [47] proposed a globally and locally consistent image inpainting (GL) algorithm by combining GAN and CNN. This method took an inpainting network and two discriminator (a global discriminator and a local discriminator) networks as the main architecture, which greatly improved the quality of image inpainting. Although the GL model can complete a variety of scenes, the effect of this method will be poor if the mask of the image contains a large area of structural objects, such as people and animals.

(3) Combined GAN. In order to solve some complex problems, a method of splitting complex problems into small parts and then training them one by one is proposed. This method achieves better results by translating, stacking, or combining multiple GANs.

Synthesizing high-quality images through text description is a difficult problem in computer vision. For this reason, Zhang et al. [48] proposed a stacked GAN (StackGAN) model, which was essentially a combination of two layers of CGAN. It used a two-stage generation method: the first stage drew the basic outline and color of the image according to the given text and generated a low-resolution image; and the second stage used the low-resolution image and text description generated in the first stage as input, corrected the defects in the results of the first stage, and added some details on this basis to generate high-resolution images. The StackGAN model stabilizes the training of CGAN very well and has made significant improvements in generating realistic images based on text descriptions. StackGAN++ in Zhang et al. [49] is also used to generate high-resolution images for text.

Zhu et al. [50] proposed cycle consistent generative network (CycleGAN) in 2017. CycleGAN realized the conversion of images from source domain x to target domain y and did not need pairs of images as training data. The model of CycleGAN is a ring structure, which is composed of two generators and two discriminators. CycleGAN can learn the mapping relationship and transformation method between artworks and real photos. Its disadvantage is that although its cyclic mechanism can ensure that the imaging will not deviate too much, some information will be lost in the cyclic conversion, resulting in the low quality of the generated image.

In order to extract information from areas far from the image, expand the receptive field of computer vision, and improve the stability of network training, Yu et al. [51] proposed a deep generative model-based approach (DGM) based on the GL method. The method has global and local discriminators, and its generator is composed of a feed-forward generation network with two stages. A narrow and deep network is used in the generator. In this method, the coarse network and fine network used for image completion are "connected in series," and the two-generation networks with different functions in the front and back stages are combined into a generator to make the model structure "narrow." The two networks are respectively responsible for the completion tasks of the first and second stages and are composed of deep expansion convolution layers, so the structure is "deep." Similar to the GL method, this method uses a deeper extended convolution layer in both stages of image completion from coarse to fine, but the number of parameters is significantly less than that of the GL method.

In order to maintain higher sample diversity, obtain multiple inpainting results, and improve the completion effect of existing methods in large missing areas, Zheng et al. [52] proposed a pluralistic image completion (PICNet) method with two parallel paths based on probability principle in 2019 and used short and long-term attention layer in the model. PICNet can generate multiple different completion schemes with trusted content for a single masked input. The use of short-term and long-term perception layers improves the reliability of completion. PICNet has high quality in various datasets, especially for large missing areas.

(4) StyleGAN. When the resolution of the generated image is very high, the discriminator can easily recognize that the image generated by the generator is fake but this makes the generator difficult to train. The described above problem makes DCGAN can only generate 64×64 images. When DCGAN generated a larger resolution image, the output seriously lost more details. To solve this problem, Karras et al. [53] proposed progressive growing of GAN (PGGAN) by introducing a progressive training method through gradually generating images from 4×4 to 1024×1024. The generator and discriminator in PGGAN grow incrementally by gradually adding new layers to make the network model more complicated to learn better-detailed features. This method can not only accelerate the training but also make the training more stable. The progressive method of PGGAN allows the training to first discover the distribution of images with large-scale structures and then gradually shifts the focus to better scale details, instead of having to learn all scales at the same time. Moreover, the generator and the

discriminator compete with each other to promote the training of both. With the gradual growth of the GAN network, most of the iterations are completed at a lower resolution, which reduces the training time.

StyleGAN proposed by the well-known technology company NVIDIA uses the idea of style conversion to design a new generator architecture in Karras et al. [54], which has a better effect when generating a single object. The network structure of StyleGAN consists of two parts: the first is the mapping network, which is used to control the style of the generated image; and the second is the synthesis network, which is used to generate the image and is used to control the style of the generated image. The entire network structure of Style-GAN still maintains the structure of PGGAN. However, the images generated by StyleGAN sometimes contain speckle-like artifacts. For this reason, StyleGAN v2 is proposed in Karras et al. [55].

(5) Self-Attention GAN. Deep CNN can improve the details of GANs to generate high-resolution images, but due to the limitations of the local receptive field of convolutional networks, if you want to generate long-range dependency regions, CNN will have problems. For this reason, the researchers proposed to apply the attention mechanism to make the designated and refined area has a larger field of vision.

Zhang et al. [49] built a variant of GAN that is attention driven and process dependent, called self-attention GAN (SAGAN). It combined the self-attention in the attention model with the original GAN, thus realized the application of all feature points in the low-resolution image to generate high-resolution detail features and used spectral normalization to improve the training effect of the generator in the training process. Both generative network and discriminant network of SAGAN adopt attention mechanisms. SAGAN can well model the attention-driven process dependency and coordinate the relationship between each position and each end detail when generating the image. The discriminator of SAGAN can also more accurately realize the complex geometric constraints on the global image structure.

In order to solve the problem of successfully generating high-resolution and diverse samples from complex datasets, BigGAN in Brock [56] was proposed in 2018. Based on SAGAN, BigGAN uses a set of TPUs to improve accuracy by leaps and bounds. BigGAN uses a larger number of channels in the design of the convolutional layer and uses a large batch size, which can maximize the performance of GAN. BigGAN can improve the model performance and make the training more stable by using the truncation trick, but it needs to balance the sample diversity and fidelity. BigGAN can ensure the stability of training through the collection of existing and other novel technologies, but the accuracy will also decline. It is necessary to balance the performance and training stability. BigGAN also has the disadvantages of a large model, many parameters, and high training cost.

In summary, given the shortcomings of the original GAN, different scholars have changed its structure to obtain different types of GANs and achieved good results, but there are still shortcomings. Table 2 shows the comparison of different types of GAN models based on structural change. The performance mentioned in Table 2 is the best in each model.

2.3.3. Loss Function-Based GAN

(1) Least-Squares GAN. In order to further solve the problem of gradient disappearance during GAN training, Mao et al. [57] proposed least-squares GAN (LSGAN), which replaced the cross-entropy loss function in the GAN discriminant network with the least square loss function. The advantage of least-squares loss is that it allows the samples generated by the generator to pull the generated image to the decision boundary on the premise of cheating the discriminator. Compared with the original GAN, LSGAN generates higher quality samples and its training process is more stable. The defect of LSGAN is that it does not solve the problem of gradient dispersion when the discriminator is good enough, because the least square function does not meet the Lipschitz continuity condition, and its derivative has no upper bound.

(2) Wasserstein GAN. In order to solve the problem of the disappearance of the training gradient of the original GAN, Arjovsky et al. [58] proposed Wasserstein GAN (WGAN) in 2017, which analyzed the reasons for the instability of the GAN in the training process, but also theoretically solved the mode collapse. It proposed to use Wasserstein distance to replace the JS divergence used in the loss functions of various GANs before. This method does not need generator and discriminator to maintain balance in the training process. It can generate more kinds of images to ensure the diversity of generated samples. WGAN promotes the development of GANs. However, in the experiment, it is found that WGAN still has the problems of difficult training and slow convergence. In the process of limiting the weight, if the design is inappropriate, the gradient explosion or gradient disappearance of the network will occur.

Given the problems existing in WGAN, Gulrajani et al. [59] introduced a gradient penalty on the basis of WGAN called WGAN-GP. It discards the weight clipping in WGAN and directly adds the gradient of the discriminator as a regular term to the loss function of the discriminator, that is, the gradient penalty replaces the weight clipping operation. WGAN-GP solves the problem of vanishing/exploding gradients while generating higher quality samples. However, the experiment found that the method has a slow convergence speed. Under the same dataset, WGAN-GP requires more training times to converge.

(3) Context-Aware Semantic Inpainting. In 2018, Li et al. [60] proposed a context-aware semantic inpainting method (CASI) based on the GAN framework. Its generator used a complete convolution network to replace the full connection layer to better preserve the spatial structure and combined the improved joint loss function to capture the high-level semantics in the context. The complete convolution network can save the spatial information of the image so that the network can accept the input of any dimension, and the

Table 2: Comparison of different types of structural change-based GAN.

Category	Model	Structural features	Advantage/disadvantage	Perf.
Initial model	GAN	Generator and discriminator	High-definition images; unstable training; mode collapses; disappearing gradient.	225 (mean log likelihood)
Conditional GAN	CGAN	Adding constraints to the input layer of GAN to guide generation	High requirements for the dataset; tagged dataset; low quality image.	—
	Info-GAN	Use generator and discriminator and add hidden code in the generator	Extending the theory and increasing the interpretability of GAN model; large amount of computation; and poor diversity.	—
Deep convolutional GAN	DCGAN	Symmetrical discriminator and the generator; using fractional-strided convolution instead of upsampling.	Stability training; No artifacts; clearer edges; data distribution affect the effect; unsuitable for complex scenarios.	85.95 (accuracy)
	GL	A inpainting network and two discriminators network	The result is locally and globally consistent; can complete any scene; and unsuitable for heavily structured objects.	96.5% (accuracy)
Combined GAN	DGM	Global and local discriminators; two stages feedforward generation network.	Expanding the receptive field and improving the stability of network training; less parameters.	18.91 (PSNR)
	StackGAN	Superimpose two CGANs	High-resolution images; stable training.	8.45 (IS)
	Cycle-GAN	Circular mechanism to achieve two domain conversions and constraints	Low data requirements; no one-to-one paired images; low resolution is not high.	0.58 (accuracy)
	PICnet	Parallel generating paths and reconstructing paths; variation encoder structure of the generator.	The reliability of complementary content; high-quality images ; suitable for the images with arbitrary of incomplete parts.	20.10 (PSNR)
Pattern-based GAN	PGGAN	Gradually grow generators and discriminators.	Reducing the training time; generating better high-resolution images.	0.2838 (MS-SSIM)
	Style-GAN	Including mapping network and synthesis network; style-based structure of generator.	Can control the visual features from coarse features to fine details.	4.40 (FID)
	Style-GAN2	Replace the gradient network structure of style GAN with fixed network training.	Focus on repairing artifacts and further improving the quality of generated images	2.32 (FID)
Self-attention GAN	SAGAN	Both generative network and discriminant network adopt attention mechanism.	Getting good balance between improving the receptive field and reducing the amount of parameters.	18.65 (FID)
	BigGAN	More channels in convolutional layer and use truncation and orthogonal regularization	Model training is stable and can generate ultra-clear images; large amount of parameters; difficult to train	7.4 (FID)

spatial information of the image can be retained without dimensionality reduction. In addition, without the full connection layer, the complete convolution network occupies less memory and can learn and predict more effectively. The CASI model can successfully reduce semantic errors, infer semantically valid content from the image context, and generate images that conform to the context of the image. Compared with the normal model using the fully connected layer, it can better grasp the spatial information of the image and make the structure of the completed image more reasonable.

To sum up, good results have been achieved in changing the original GAN loss function, but there are still deficiencies. Table 3 shows the comparison of loss function-based GAN. The performance in Table 3 is the best in each paper. Table 4 lists various image inpainting methods and their advantages and disadvantages. The use of various image inpainting techniques under normal morals and ethics reproduces the classics of the past, makes up for the defects of the pictures, and brings more and more laughter to people. However, when the repaired image goes beyond the original intention of the image, it becomes image forgery and image malicious attack, which also provides technical conditions for attackers to use the inpainting image for malicious purposes. The multicategory and realistic high-resolution color images generated by forged technologies [41, 48, 49, 53, 56, 61] not only destroy the authenticity of the images, but also convey false information to the public and may even cause serious social harm. As a technology, how to use it, whether it brings happiness or disaster to people depends entirely on people themselves, and not on tools. Therefore, it is necessary to detect the images inpainting by these technologies.

3. Digital Image Forensics

How to identify the forgery and malicious attacks brought by image inpainting has attracted the attention of a large number of people in government, associations, and academia. Then, they put forward many solutions, resulting in digital image forensic technology [62–65].

Table 3: Comparison of GAN model based on variant of loss function.

Category	Model	Structural features	Advantage/disadvantage	Perf.
Initial model	GAN	Generator and discriminator	High-definition images; unstable training; mode collapses; disappearing gradient.	225 (mean log likelihood)
GAN model based on loss function variation	LSGAN	Replacing the cross-entropy loss function with the least square loss function	Limit the unlimited modeling capabilities of GAN; unsolved the problem of gradient dispersion in the generator.	6.47 (IS)
	WGAN	Using Wasserstein distance instead of JS divergence	Stable training; theoretically solves the model collapse; gradients disappear and explode; slow convergence speed.	—
	WGAN-GP	Using gradient penalty mechanism instead of weight interception operation	Stable training; solves the problem of gradient disappearance and explosion; higher quality samples; slow convergence speed.	7.86 (IS)
	CASI	Fully convolutional network; introduced perceptual loss; joint loss function.	Reduces semantic errors; generates images that conform to the contextual content.	20.37 dB (PSNR)

Table 4: Comparison of various image inpainting methods.

Method	Advantage	Disadvantage
Image editing inpainting	Widely popularized; easy to operate; easy to learn	The effect only stays at the visual level and depends on the experience of the operator; the repair speed of multiple pictures is slow.
Image synthesis inpainting	The inpainting speed is faster and the algorithm is relatively stable. The inpainting can be completed under the condition of a small number of images.	The abstraction process of information and data only stays at a shallow level; higher level features cannot be obtained. The resolution is low.
Image inpainting based on deep learning	High-level features of the image can be obtained; the method is independent and anti-interference; the image inpainting rate is higher and the effect is better.	Rely on a large amount of labeled data. The inpainting speed depends on the configuration of the device. The accuracy of the calculation results and the robustness of the method cannot be obtained at the same time. The gradient tends to be unstable.

In the government and associations, the Defense Advanced Research Projects Agency (DARPA) launched a research project called media forensics to develop an end-to-end digital media forensic platform that automatically evaluates the integrity of images and videos [66]. IEEE launched the digital image forensic competition in 2013 to promote technological progress in this field [67]. Recent forensic tools such as FotoForensics [68] and MMC Image Forensic Tool [69] are created to identify the image.

In academia, digital image forensic technology realizes the objectives of identifying the source, detecting the processing operation of the image, and judging the authenticity of content by analyzing the inconsistency and tampering traces of image content. Digital image forensics approximately describes the process of applying image processing operation to a digital image in the form of a mathematical model. Through the analysis and understanding of the mathematical model, it is proposed that the features of the above operation process can be characterized. Finally, the corresponding feature extraction algorithm is designed to detect whether the operation process exists in the image. This research process requires theoretical knowledge such as digital image processing, signal processing, mathematical-statistical analysis, computer vision, pattern recognition, machine learning, etc. In addition, the research on image forensics is helpful to further understand the change law of statistical features of images before and after the operation, thus improving the performance of image processing operation. The research of digital image forensics is based on "digital fingerprint," which can be the information actively added by the forensics, or the unique traces introduced by image processing or tampering. These two types of digital fingerprints correspond to active forensics and passive forensics, respectively. The classification of forensics is shown in Figure 5. The active forensic method is to actively add a digital watermark [70, 71] or digital signature [72] to the original image. When copyright and authentication problems occur, the extraction algorithm is used to extract this information to provide copyright proof or content authenticity and integrity authentication. Passive forensics, also known as blind forensics, means to verify its authenticity and source only by analyzing the image without adding any information in advance. The methods of image verification can use image generation features, image operation features, and deep learning technology. At present, passive forensics has a wide range of applications and is the focus of scholars' attention, so this article mainly focuses on passive forensics.

Understanding the image generation process is helpful to better study passive forensics. Therefore, Figure 6 shows the generation process of a digital image. In a specific scene in the real world, the camera image is generated by the lens, optical filtering, CFA interpolation, and compression coding in the camera, and then the final image is obtained after

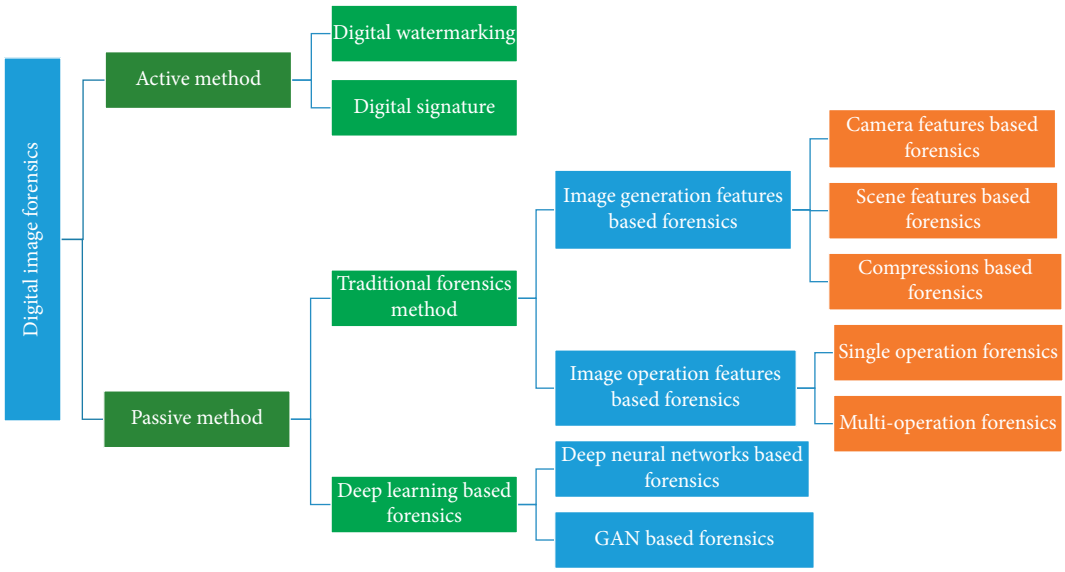


Figure 5: Classification of digital image forensic technology.

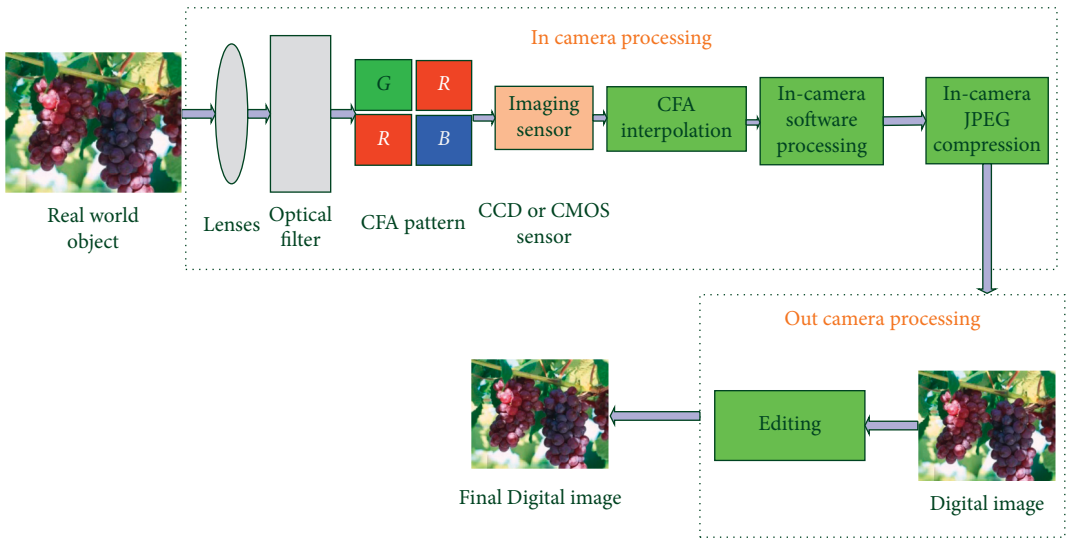


Figure 6: Generation process of a digital image.

editing and processing out of the camera. For each of the above stages, scholars have carried out a large number of forensic research. Forensic efforts are devoted to finding traces left in each step of the above image generation process. These traces may come from light and shadow in the scene, an imaging component of the camera, image editing operations, or anti-forensic operations added to cover up image editing operations. These traces can be regarded as the digital fingerprint of the operation, which uniquely corresponds to the operation performed so that the fingerprints are different from each other. These digital fingerprints are the design basis of the forensic detection algorithm. According to the digital fingerprints generated in different stages, digital image forensic technology can be divided into the following categories: traditional forensics and deep learning-based forensics. Traditional forensic method mainly includes image generation feature-based forensics and image operation feature-based forensics. Deep learning-based forensics mainly includes deep neural network-based forensics and GAN-based forensics. These forensics methods will be introduced as follows.

3.1. Traditional Forensics

3.1.1. Image Generation Feature-Based Forensics. According to the process of image generation, the specific scene is imaged by the camera, and then the captured images are compressed and stored or published. Since the camera, specific scene, and generated compressed image all have their features, obtaining the common features of the generated image can detect whether the image has undergone the operation. Based on detecting features, the image generation feature-based forensic technology has developed into

scene feature-based forensics, camera feature-based forensics, and compression-based forensics.

The scene feature-based forensics is often based on fact that the characteristics of illumination, shadow, reflection, geometric characteristics, and perspective in the same scene are the same, while the lighting and shadow generated in different scenes are different, and even the geometric characteristics of objects from different scenes do not match each other.

Therefore, based on the inconsistency of the above characteristics in the image, many forensic technologies have been developed. Johnson et al. [73] conducted forensic research on illumination features, [74] is on shadow features, [75, 76] are on texture features, [76, 77] are on vector methods in image objects, [78] is on the inconsistency of objects, and height ratio features in images.

The camera feature-based forensics mainly adopts inconsistent characteristics such as the distribution or relationship of camera color [75, 79], camera pattern noise [80, 81], photo response non-uniformity (PRNU) [82–84], and color filter array (CFA) [85, 86] to realize image forensics.

For facilitating image storage and transmission, images are usually undergoing compression encoding. If the image has undergone tampering, it is inevitable to save the image in a compressed format again. Therefore, the detection of single and multiple compression encoding operations has become an important forensic problem. For compression detection, DCT transformation is one of the steps. The blocking execution of JPEG compression will lead to a blocking effect, and the higher the compression intensity, the more significant the blocking effect. Luo et al. [87] counted the integral of the DCT coefficient histogram in the region and calculated the ratio between them. If the ratio is less than a certain threshold, it is judged that the image has undergone JPEG compression. Because the image tampering operation must save the image again, this will lead to two or more JPEG compression. Huang et al. [88] considered that two compression operations adopt the same quantization step size and detect two JPEG compression operations by analyzing the number of changed DCT coefficients. Milani et al. [89] used the distribution of the first number of DCT coefficients to detect double JPEG compression.

3.1.2. Image Operation Feature-Based Forensics. Whether it is to achieve image modification or to cover up the traces of image tampering, people often use various image processing operations. The study of these operations can deeply explore the impact of various operations on the statistical law of the original signal. The detection of these operations can effectively expose the fact of image tampering and provide more detailed evidence. According to the types of operations to modify or tamper with images, it can be divided into single operation forensics and multioperation forensics. Single operation forensics can be divided into image enhancement, median filtering, resampling, and content modification. Multioperation forensics is a combination of the above single forensic technologies.

(1) Single Operation Forensics. The image enhancement operation can improve the image quality or mask the trace of image modification by modifying the brightness distribution, smoothing or sharpening, noise removal, etc. The main image enhancement operations include contrast enhancement, histogram equalization, median filtering, and so on. Contrast enhancement operation will produce "peak-valley" effect in the histogram. Stamm et al. [90] detected this feature by Fourier transform of image gray histogram. Cao et al. [91] further used the zero-height gray-level features in the histogram to detect homology contrast-enhanced image synthesis. Zou et al. [92] exploited GAN to process the contrast-enhanced image and make it indistinguishable from the unaltered one in the pixel domain. Then, a designed histogram-based loss to enhance the attack effectiveness in the histogram domain and the gray-level co-occurrence matrix domain and pixel-wise loss to keep the visual enhancement effect of the processed image are introduced.

Median filtering can smooth the image while maintaining the edges of the image through nonlinear operations. It is widely used for image denoising and smoothing, and even for anti-forensic operations [93]. Kirchner et al. [94] calculated the ratio of the number of zeros to the number of ones in the difference graph and detected the median filter for the probability of occurrence of zeros. To resist the postprocessing of JPEG compression, Liu et al. [90] used the SPAM feature in steganalysis. In order to avoid the influence of image content on the statistical characteristics of image difference, Popescu et al. [95] proposed the concept of median filter residual (MFR) and adopted an autoregressive model to capture the statistical characteristics of MFR.

Resampling is an indispensable process after the image is scaled, rotated, and affine transformed. Therefore, it has attracted the attention of many forensic researchers. Resampling operation will introduce correlation between local samples, and this correlation can be used as an important basis for testing resampling [95]. Mahdian et al. [96] designed detection algorithms based on the mean and variance periodicity of the second-order difference of the image after image resampling to realize the judgment. Vázquez-Padín et al. [97] performed singular value decomposition on the image matrix to extract eigenvalues and distinguish whether the image has undergone resampling operation according to the distribution characteristics of eigenvalues.

Modifying the image content is the ultimate goal of the tamper, for the most commonly used copy-move technique, so the forensics of image copy-move is the focus of image forensics. An algorithm for identifying copy-move tampering in images was proposed by Avidan et al. [98] for the first time. It divided the image into partially overlapping blocks and traversed all the blocks for block matching search. If there are two identical blocks, it is judged that there is a copy-move operation. The idea points out the direction for related research work in this field in the future. Based on this idea, many forensic methods that extract features from image blocks and then conduct block matching search are proposed, mainly including forensic methods based on scale-invariant feature transform [99], DCT coefficient

[99, 100], residual image [101], JPEG compression [102], and Fourier-Mellin transform [103].

The content-aware image resizing (CAIR) technology proposed by Avidan et al. [98] is another content modification operation technology. To detect the CAIR operation, Chen et al. [104] designed an improved method based on calibrated adjacency density (CNJD) to identify seam carving for JPEG images. Liu et al. [105] proposed a detection method based on large-scale feature mining, which used an integrated learning classifier when dealing with high-dimensional problems to avoid overfitting of traditional classifiers in detection. Liu et al. [98] proposed a hybrid detection method based on large-scale feature mining. Taspinar et al. [106] detected the tampered image with seam carving using the PRNU feature of the camera. Liu et al. [107] extracted high-dimensional features in the spatial domain and frequency domain. Then, they used an integrated learning algorithm to detect whether the JPEG image has undergone the seam carving operation. Ye et al. [108] proposed a detection method based on mixed features, including subtracted pixel adjacency model, Markov transition probability, and local derivative mode. Zhang et al. [109] proposed a blind detection method for image seam carving for low zoom ratios. Zhang et al. [110, 111] proposed a method for detecting image seam carving based on the Unified Local Binary Pattern. The CAIR operation will affect the local texture of the image, resulting in changes in the amplitude distribution of the pixel intensity difference in the local neighborhood, and the correlation of the pixels in the local neighborhood will be destroyed [112, 113].

(2) Multioperation Forensics. For forgers, when a single operation is difficult to achieve purpose, they will often consider using multiple operations. The concept of multiple operations was put forward earlier in Comesaña et al. [114]. Thereafter, different methods are proposed. Chen et al. [115] researched multioperation forensics for the image undergoing JPEG compression, then zooming, rotating, or both operations on the image, and finally saved it as a JPEG format. It proposed a deformed block-effect grid feature extraction algorithm and DCTR feature to detect multioperation forgery. Chen et al. [116] proposed a random feature selection strategy for multiple operations and proved the effectiveness of the strategy theoretically. Conotter et al. [117] constructed model for multiple operations of JPEG compression and linear filtering to describe the DCT coefficients in JPEG images after linear filtering of the entire image. Ferrara et al. [118] proposed two methods to detect the multiple operations of double JPEG compression with linear contrast enhancement between them. Its detectors used the "peak-valley" effect of the DCT coefficients and the distribution characteristics of the first digits. Bianchi et al. [119] used the idea of reverse engineering to detect multiple operations of double JPEG compression with scaling between them. Chu et al. [120] proposed a method to define the order detection problem as a multi-hypothesis test problem. Baracchi et al. [121] studied the anti-forensic method and pointed out that the image single operation (e.g., JPEG compression, sensor pattern noise, histogram statistics) could negatively affect other traces. Based on this observation, they could generate a pristine image with native camera metadata, JPEG structure, quantization tables, preview, thumbnails, their corresponding quantization tables, single compression statistics, and any in-camera (even proprietary) processing under a given tampered picture. They called this method camera obscura.

3.2. Deep Learning-Based Forensics. Deep learning has the features of powerful learning ability, strong adaptability, and good expression ability, which has caused evidence collectors to introduce it into the field of forgery detection and forensics. According to the deep learning technologies applied to forensics, it can be divided into deep neural network-based forensics and GAN-based forensics.

3.2.1. Deep Neural Network-Based Forensics. According to the features and structure of the forensic method used, the deep neural network-based forensics can be divided into three categories, namely image generating feature-based method, operation feature-based method, and CNN structure or loss function changed method. Table 5 lists the classification, method, detection type, and the best performance in each article of deep neural network-based forensics. Their specific descriptions are as follows:

(1) Image Generating Feature-Based Method. For the double JPEG compression operation, Wang et al. [122] designed a deep convolution neural network by extracting the histogram of DCT coefficients. On the premise of no manual careful design of features, the deep network is used to automatically learn the most essential changes of histogram features before and after tampering. Finally, tamper detection based on deep learning is demonstrated. Bunk et al. [123] proposed two methods to detect and localize image manipulations based on a combination of resampling features and deep learning. The first method used the resampling features, deep learning classifiers, a Gaussian conditional random field model, and random walker. The second method used resampling features and LSTM-based network. Experimental results show that both are effective in detecting and localizing digital image forgeries.

The intrinsic model features of the camera can also be used for image forensics in combination with deep learning technology. Bondi et al. [124] used CNN to extract the intrinsic model features of the camera from the image patch. Then, these features were analyzed by clustering technology to detect image tampering and locate tampering. In Chen et al. [125], the camera response function (CRF) was used for splicing and copy-move detection and positioning. Local descriptors based on image noise residuals are widely used in image forensics, such as tamper detection and location. Cozzolino et al. [126] showed a residual-based descriptor, which can actually be considered as a simple constrained CNN. Zhou et al. [127] proposed a double stream faster R-CNN network to detect tampered images. One branch of the two streams is the RGB stream, which is used to learn strong contrast differences. Another stream is used for noise

Table 5: Comparison of deep neural network-based forensics.

Category	Source	Method	Detection type	Perf.
Image generating feature-based method	[122]	Histograms of discrete cosine transform (DCT) coefficients	Double JPEG compression	0.98 (AUC)
	[123]	Resampling features, deep learning, and LSTM	Image manipulation	0.9138 (AUC)
	[124]	Camera models	Image authenticity.	0.908 (AUC)
	[125]	Camera response functions	Splicing and copy-move operations	0.97 (accuracy)
	[126]	Residual-based local descriptors	Image manipulation	100.00% (accuracy)
	[127]	Tampered artifacts and noise features	Image manipulation	0.95 (AUC)
	[128]	LSTM and CNN; similar patches	Image object removal	97.89% (accuracy)
Operation feature-based method	[129]	Median filtering residual; Modified CNN;	Compressed image	96.84% (accuracy)
	[130]	Hybrid CNN-LSTM; resampling features	Copy-clone, object splicing, and removal	94.8% (accuracy)
	[131]	Block-like features; self-correlations between different blocks,	Copy-move	0.7572 (F-score)
	[132]	BusterNet; visual artifacts; visual similarities	Copy-move	80.48% (accuracy)
	[133]	Similarity between feature vectors of image overlapped subblocks	Copy-move	0.93 (F-measure)
	[134]	Dense-InceptionNet	Copy-move	0.7058 (precision)
	[135]	Adaptive attention and residual refinement Network ; Atrous spatial pyramid pooling ; correlation maps	Copy-move	0.8488 (AUC)
CNN structure or loss function changed method	[136]	CNN with weighted cross-entropy loss function	Patch-based operation.	98.3% (AP)
	[137]	Decoder network and feature pyramid network	Patch-based operation	98.99% (TPR)
	[138]	Encoder-decoder network structure; using a label matrix and the weighted cross-entropy as the loss function	Recapture image	99.74% (accuracy)
	[139]	CNNs and the segmentation-based multiscale analysis	Splicing and copy-move	0.4063 (F1-score)
	[140]	Segmentation-based key point distribution strategy and adaptive over segmentation method	Copy-move	0.7627 (F1-measure)
	[141]	CNN-based framework; full-resolution information	Splicing, copy-move	0.886 (AUC)
	[142]	Modified CNN by demosaicing algorithms; mosaic inconsistencies	Splicing, inpainting, or copy-move	0.926 (AUC)
	[143]	Constrained R-CNN	Splicing, copy-move, and removal.	0.992 (AUC)
	[144]	Dense fully convolutional encoder-decoder architecture with dense connections and dilated convolutions	Commonly used editing tools and operations in photoshop	0.99 (AUC)
	[145]	ResNet; image residuals	Remove; CNN inpainting	97.97 (precision)
	[146]	Multibranch CNN architecture	Copy-move	0.920 (F1-score)
	[147]	RGB-N, MSCNNs, DCNNs	Splicing, copy-move	0.7328 (precision)
	[148]	Feature pyramid network, stagewise-weighted cross-entropy Loss function	JPEG compression, scaling	0.9967 (F1-score)
	[149]	CNN with CRF-based attention model	Splicing, copy-move	0.804 (F1-score)
	[150]	Dense self-attention encoders	Copy-move	0.883 (AUC)

extraction to find the noise inconsistency between the real and tampered regions. Then, the bilinear fusion of the features from the two streams is conducted to enhance the tamper identification ability. The algorithm achieved better detection performance than each stream and the comparison method. For detecting patch-based image inpainting, Wang

et al. [151] proposed a mask regional convolutional neural network (Mask R-CNN) approach by adjusting the sizes of the anchor scales due to the inpainting images and then by replacing the original non-maximum suppression single threshold with an improved non-maximum suppression (NMS) to reduce the missed detection areas and improve detection accuracy. Lu et al. [128] proposed an image inpainting forgery detection method based on LSTM-CNN. This method used the strong learning ability of CNN to search abnormal similar blocks and used the LSTM network to eliminate the influence of texture consistent region on the detection results.

(2) Operation Feature-Based Method. Chen et al. [129] made the first layer be the filter layer that received the input of images and outputs them as media filtering residual to improve the effect of CNN. The author proposed to use deep learning technology to solve the problem of median filter forensics for the first time, which is the first work of deep learning in the field of forensics, and also played a guiding role in the combination of follow-up deep learning and image forensics.

To efficiently segment various types of manipulations including copy-move, object removal, and splicing, Bappy et al. [130] proposed a high-confidence manipulation localization architecture, which included CNN architecture to provide spatial feature maps of manipulated objects and LSTM to observe the transition between manipulated and nonmanipulated patches by using resampling features of the patches, and a decoder network to learn the mapping from low-resolution feature maps to pixel-wise predictions for image tamper localization. Aiming at the problem of copy-move forgery location, Wu et al. [131] used CNN to extract block features from the image. Then, the feature points were matched according to the autocorrelation between different blocks. Finally, the forgery location results were generated by a deconvolution neural network. Wu et al. [132] added another branch structure based on the similarity detection network [122] to detect the forged target area of the image. Then, they proposed a two-way fusion BusterNet model, which can preliminarily distinguish the original area of the image, the forged source area, and the forged target area, but there is still a problem of low detection accuracy. Muzaffer et al. [133] used AlexNet to extract the feature vectors of images and studied the similarity between them for copy-move forgery detection and location. Zhong et al. [134] proposed using a dense network structure to solve the problem of copy-move forgery detection and location. First, the pyramid feature extractor was used to extract multidimensional and multiscale features. Second, the feature correlation matching module was used to independently learn the correlation of features. Finally, the forgery location results were obtained through the postprocessing module. Zhu et al. [135] introduced the adaptive attention mechanism and the neural network structure of residual optimization into the copy-move forgery location task. Based on obtaining the coarse mask, the result was optimized through the residual subdivision module to obtain the final forgery location result.

(3) CNN Structure or Loss Function Changed Method. Zhu et al. [136] proposed a CNN-based approach which was built following the encoder-decoder network structure to detect patch-based inpainting operation by predicting the inpainting probability for each pixel in an image. By assigning a class label for each pixel of an image can guide the CNN to automatically learn the inpainting features. Using the weighted cross-entropy as the loss function can supervise the CNN to capture the manipulation information. Zhu et al. [137] proposed an image inpainting forensic method based on deep CNN. This method used an encoder network to extract image features and a decoder network to restore the feature map extracted by the encoder network to the original image size. Then, the feature pyramid network is used to supplement the information of the feature map in the upsampling process. Yang et al. [138] proposed a Laplacian CNN algorithm to detect and reacquire images which put signal enhancement layer into CNN structure, and Laplacian filter was used in the signal enhancement layer. Liu et al. [139] proposed a CNN and segmentation-based multiscale localization to analyze the tampered area in a digital image. The author designed the unified CNN architecture to get a set of tampering detectors for different scales to generate a series of complementary tampering possibility maps. Then, the proposed segmentation-based method fused these maps and generated the final decision map. Liu et al. [140] used convolution kernel network for feature extraction in copy-move forgery detection and location tasks and used adaptive segmentation methods to obtain the forgery location results. In order to solve the problem that the current deep learning model adjusts the size of the input image to destroy the valuable high-frequency details and seriously affect the forensic effect, Marra et al. [141] proposed a CNN-based image forgery detection framework based on full-resolution information gathered from the whole image. Using gradient check pointing, the framework was trainable end-to-end with limited memory resources and weak supervision, allowing for the joint optimization of all parameters. Bammey et al. [142] designed a CNN structure based on demosaicing algorithms that can train directly on unlabeled and potentially forged images to point out local mosaic inconsistencies and then classify image blocks by their position in the image. To address the deep learning-based models lacking poor universality of handcrafted features and only focusing on manipulation localization and overlooking manipulation classification, Yang et al. [143] proposed constrained R-CNN for complete and accurate image forensics. It included a learnable manipulation feature extractor to create a unified feature representation of various content manipulation directly from data and a two-stage architecture to simulate the coarse-to-fine forensic process in the real world.

To well capture tampering traces left by Photoshop, Li et al. [144] proposed a fully convolutional encoder-decoder architecture with dense connections and dilated convolutions for achieving better localization performance. For effectively training a model in the case of insufficient tampered images, it designed a training data generation strategy by resorting to Photoshop scripting, which can

imitate human manipulations and generate large-scale training samples. Li et al. [145] found that in the residual region of the image, the transfer probability value of adjacent pixels in the region repaired by deep learning is much lower than that of adjacent pixels in the unrepaired region, indicating that the repaired image contains fewer high-frequency components. According to this law, the author first set up a learnable prefiltering module to extract the image residual, then used four consecutive ResNet V2 modules for feature extraction, and finally carried out upsampling to realize the accurate positioning of the repaired area. Barni et al. [146] designed a multibranch CNN architecture that solved the hypothesis testing problem by learning a set of features capable to reveal the presence of interpolation artifacts and boundary inconsistencies in the copy-moved area. Zhang et al. [147] proposed a hybrid architecture called Pixel-level Image Tampering Localization Architecture (PITLArc) by integrating the advantages of top-down detection-based methods and bottom-up segmentation-based methods. To evaluate the effectiveness, they used one outstanding detection-based method (dual-stream faster region-based convolutional neural network (RGB-N)) and two segmentation-based methods (multiscale convolution neural networks (MSCNNs) and dual-domain convolutional neural networks (DCNNs)) to implement the PITLArc. Zhang et al. [148] improved U-shaped net to migrate feature pyramid network (FPN) for multiscale inpainting feature extraction and designed a stagewise-weighted cross-entropy loss function to take advantage of both the general loss and the weighted loss to improve the prediction rate of inpainted regions of all sizes. Rao et al. [149] found the key observation that the boundary transition artifacts arising from the blending operations are ubiquitous in various image forgery manipulations, which is well characterized with CRF (conditional random field)-based attention model. Therefore, an image forgery detection and localization scheme is proposed based on a deep convolutional neural network (CNN) and CRF-based attention model. Hao et al. [150] proposed a novel image localization method inspired by transformers called TransForensics. It had two major components which were dense self-attention encoders and dense correction modules. The dense self-attention encoders were used to model global context and all pairwise interactions between local patches at different scales. The dense correction modules were used for improving the transparency of the hidden layers and correcting the outputs from different branches. Experiments showed that TransForensics could not only capture discriminative representations and obtain high-quality mask predictions but were also not limited by tampering types and patch sequence orders.

3.2.2. GAN-Based Forensics. According to the different purposes of GAN-based forensics, it can be divided into forensics for classification and forensics for localization. Table 6 lists the classification, source, detection type, methods, and the best performance in each article of GAN-based forensics. Their specific descriptions are as follows.

(3) GAN-Based Forensics for Classification. Li et al. [152] statistically analyzed each component of the original pixel domain and residual domain of the image in three color spaces (RGB, HSV, and YCbCr). Then, they found the difference between the GAN tampered image and the real image in the residual domain of different color spaces. After analyzing, they got the component with the most significant difference (i.e., H, S, Cb, and Cr) from the statistical results. According to the above statistical results, it first selected the required channels from different color spaces. Second, the residual characteristics of color components and the co-occurrence matrix were calculated. Third, connected them into feature vectors and finally trained classifiers to predict whether the image is real or generated by the deep network. Nataraj et al. [153] proposed a method to detect GAN-tampered images using a three-channel co-occurrence matrix. In this method, the author did not calculate the co-occurrence matrix on the residual domain. Instead, they directly used the three channels of the input image to calculate the co-occurrence matrix, then inputted it into the convolution network to extract features, and finally used the fully connected layer for classification. McCloskey et al. [154] extracted features from color and saturation space to detect GAN-generated images. For the color feature, the INH network is used to classify GAN-generated images by using the bivariate histograms of r, g components in the standard rg chromaticity space. For the saturation feature, the SVM is used to classify GAN-generated images by extracting two groups of saturation. According to the report, saturation statistics provided better performance. From the perspective of artificial fingerprints, Marra et al. [155] explored a PRNU-based scheme for Cycle-GAN, ProGAN, and Star-GAN-generated image detection. The results demonstrated that those GAN schemes would leave artificial fingerprints into the generated images. Ning et al. [156] proposed a method to realize multiclassification through fingerprint matching. The author believed that each GAN network model will have its unique fingerprint affected by training data, network structure, loss function, parameter setting and other factors. The fingerprint of the GAN model will affect its generated image and leave a unique image fingerprint. Therefore, they designed a forensic method to extract GAN model fingerprint and image fingerprint and then matched them to realize image classification. No matter what types of images are generated by various GAN and what network structure is used, the synthetic false images have the same defects. Wang et al. [157] explored how to use a single GAN model to identify other GAN-generated images. First, 11 GAN models were used to construct a large-scale synthetic image authentication database. Thereafter, only using a single ProGAN model to train can show good generalization performance on the above database. Experiments show that data enhancement as a postprocessing method and the diversity of training data is the key factor to success. Mo et al. [158] expected that the main difference between the original and GAN-generated images would be reflected on the residual domain. Therefore, they presented a three-layer CNN with a Laplacian filter preprocessing to identify fake face images generated by PGGAN. Marra et al.

Table 6: Comparison of GAN-based forensics.

Category	Source	Detection type	Methods	Performance
Classification	[152]	Deep network-generated images	Residual domain of chrominance components	100% (ACC)
	[153]	Cycle-GAN Star-GAN	Co-occurrence matrices	93.42 (accuracy) 99.49 (accuracy)
	[154]	GAN-generated images	INHNet Saturation feature	0.56 (AUC) 0.7 (AUC)
	[155]	Cycle-GAN, ProGAN, Star-GAN	GAN fingerprints	0.999 (AUC)
	[156]	GAN-generated images	GAN fingerprints	99.93% (accuracy)
	[157]	11 GAN models	Single ProGAN	93.0 (mAP)
	[158]	PGGAN	Lap-CNN	96.3%
	[159]	Cycle-GAN	Eight methods	>83.58%
	[160]	GAN-generated human face images	CNN with self-attention mechanism	99.3 (accuracy)
	[161]	Five GAN models	EM; fingerprint	99.65 (accuracy)
	[162]	PGGAN	Shallow CNN architecture	99.99% (AUC)
	[163]	PGGAN	Xception	99.99% (accuracy)
	[164]	PGGAN, WGAN, Style-GAN, LSGAN, DCGAN	Global and local feature, ArcFace loss, CNN	99.99% (accuracy)
	[165]	BigGAN, Style-GAN2, PGGAN	R, G, and B components, DWT, SVM	98.45% (accuracy)
Localization	[166]	Criminisi, SN-PatchGAN	Multitask deep learning network based on mask R-CNN	97.8 (mAP)
	[167]	GAN-based face manipulation	Gray-scale fakeness prediction map; encoder-decoder architecture with attention mechanism	99.95 (ACC)

[159] evaluated the performance of several image forensic detectors and popular computer vision CNN architectures on GAN-generated images detection. More specifically, the authors used four image forensic detectors and four CNN architectures. Experimental results showed that Xception-Net has the highest average detection accuracy. Mi et al. [160] found that the upsampling process conducted by the Transposed Convolution operation was the most vulnerable link in GAN generators. The transposed convolution in the process will cause the lack of global information in the generated images. Then, the authors proposed a detection method with CNN and self-attention to improve detection accuracy. Guarnera et al. [146] found that a sort of fingerprint left in the image generation process is hidden in images. The authors extracted forensic traces through the EM algorithm and then used naive classifiers to get the classification.

In recent years, different approaches have been revealed for detecting GAN-generated images. Lee et al. [162] proposed the face manipulation detection pipeline which included facial image preprocessing and the proposed Shallow-FakeFaceNet (SFFN) detection model. The detection pipeline had two stages. The first stage was to perform facial image preprocessing to (1) crop the face region, (2) filter cropped faces, (3) apply upscaling methods, and (4) augment the data. The second stage is to pass the preprocessed faces to the SFFN models to train and detect facial manipulations. This method only used RGB channel information from the image to distinguish facial manipulated images. Based on the discovery that both the luminance and chrominance components play an important role, and the RGB and YCbCr color spaces achieve better performance than the HSV and Lab color spaces, Chen et al. [163] designed the dual-stream network to detect image by integrating both the luminance component and chrominance components of dual color spaces (RGB and YCbCr). Then, they improved Xception model by introducing the convolutional block attention module and multilayer feature aggregation module to enhance its feature representation power and aggregate multilayer features, respectively. Chen et al. [164] pointed out that the local features had played significant roles in the field of face recognition and face synthesis. For improving the generalization capability, Chen et al. [164] strengthened the learning on important local areas and combined the global and local features. Then, they proposed the method including the feature learning step and classification learning step. In the learning step, important local areas containing many face key points are learned by combining the global and local features. In the classification learning step, the metric based on the ArcFace loss is applied to extract common and discriminative features. Finally, the extracted features are fed into the classification module to detect GAN-generated faces. Tang et al. [165] proposed a detecting GAN-synthesized fake image method named fake image discriminator (FID) which used the strong spectral correlation in the imaging process of natural color images. FID includes the feature extraction stage and classification stage. In the feature extraction stage, the color image is converted into three color components of *R*, G, and B, then discrete wavelet transform (DWT) is applied to RGB components separately. In the classification stage, the correlation coefficient between the subband images was used as an input feature vector to SVM for authenticity classification.

(4) GAN-Based Forensics for Localization. Wang et al. [166] tried to use an improved mask R-CNN to detect the images generated by [15] and GAN-generated networks. Based on

the traditional mask R-CNN, the author added the local binary pattern channels to the network input, and combined feature pyramid networks and back connections to extract more features. Experiments show that the proposed method has good performance.

Using the imperfection of the upsampling procedure in all the GAN-based partial face manipulation methods and full-face synthesis methods, Huang et al. [167] proposed the FakeLocator method. First, ground truth fakeness maps are produced. Second, the real images or fake images are inputted to the encoder-decoder architecture network by introducing the attention mechanism to output gray-scale fakeness prediction maps. Last, the gray-scale fakeness prediction maps and the ground truth fakeness maps are combined to calculate the loss.

In summary, with the continuous emergence of various new technologies, forensic technology has made considerable progress and achieved good results. As the conclusion from Gragnaniello et al. [168] shows that we are still very far from having reliable tools for GAN image detection after analyzing and comparing seven detectors for GAN image detection. Table 7 shows the advantages and disadvantages of various forensic methods.

4. Game between Image Inpainting and Forgery

While forensic technology is making progress, forgers have not waited to die. After knowing the forensics algorithm, they will try their best to cover up the traces of image forgery or image operation and carry out the anti-forensic operation on the forensics so that the forensic algorithm can get wrong judgment results. At present, anti-forensic algorithms can be divided into orientation methods and general methods [169]. The orientation method is to remove a specific trace to invalidate the method of relying on this trace to obtain evidence. The general method modifies the features that the forensic method depends on to make the features consistent with the features of the image without operation, to make the forensics fail.

For the orientation method, Wang et al. [157] found that different GAN-generated fake images all leave specific fingerprint features. Although the generalization ability of detectors that rely on fingerprint feature training is not good, preprocessing the training data, such as adding JPEG compression, blur, and other operations, greatly improves the generalization performance of the model. At the same time, postprocessing the image during detection can increase the robustness of the model. Neves et al. [170] designed an automatic encoder, which can remove the fingerprint and other information from the synthetic forged image, making the existing forgery detection system invalid. Zhang et al. [171] looked for common traces of GAN to improve the robustness of the detector. The existing detectors have a strong dependence on data and lack generalization. Du et al. [172] used the automatic encoder with local sensitivity to realize the detection, making the model focus on the tampered area and have stronger universality. Cozzolino et al. [173] proposed the SpoC method which used a GAN-based approach that injected camera traces into synthetic images to deceive forensic detectors. To conceal or eliminate the traces left by multiple manipulating operations such as JPEG compression and median filtering successively, Wu et al. [174] investigated multioperation anti-forensics with GANs. Its generator network is trained to automatically learn the visual and statistical features of the original images by applying appropriate loss functions in the process of optimization. Then two strategies are given to validate the effectiveness.

In the general method, Marra et al. [159] simulated the detection of tampered pictures in the social network scene. The results showed that the existing detectors perform poorly in the real network confrontation environment. Brockschmidt et al. [175] evaluated the antagonism of deep forgery detectors (Xception [24] and Mesonet [159]). The author used six forgery datasets to detect the reliability of the detector. The results showed that the detector can achieve a very high detection rate on the same distributed datasets. However, in the dataset of unknown tamper type, only the datasets with high feature coincidence had good mobility, otherwise, the detection effect was very poor. Huang et al. [176] drew on the idea of adversarial samples, carried out adversarial attacks on these neural network-based detectors, and designed two methods—a single adversarial attack and a general adversarial attack—to make the tampering classification and positioning of the detector invalid. Most of the deep forgery detection algorithms used neural network technology, and the neural network itself had anti-sample attacks [177, 178]. The anti-sample attack was a technology that disturbed the model input and made the model misjudge, which made the deep forgery technology can hide some of its features to bypass the detection. Although many detectors performed well on some datasets, attackers could still improve the generation method and hid some symbolic features to bypass the detector, which was a long-term attack and defense game process.

When the anti-forensics is aware of the existence of forensics, they have devised different methods to achieve their goals. Chen [116] detected multiple operations with anti-forensic operations. It is assumed that the anti-forensics can obtain the forensic decision plane, modify the target image to reach the decision plane, or even surpass the decision plane, thereby deceiving the detector. It proposed a strategy of random feature selection and theoretically proved the effectiveness of the strategy. Considering the problem of camera source identification based on device fingerprints, Zeng et al. [179] studied the confrontation between analysts and fingerprint copy attackers in the case of incomplete information. The author established a Bayesian game model through the noise-level estimation algorithm. The results showed that the gains obtained by the forensics when they have incomplete information are lower than those obtained when they have complete information. A stronger confrontation is considered in the game theory framework of [180, 181], that is, anti-forensics can destroy training samples to interfere with forensic analysis. Evidence collectors and anti-forensics were both working towards their side, making both parties enter a ring with no end. Ding et al. [182] proposed an anti-forensic method based on the GAN

Table 7: Advantages and disadvantages of various detection methods.

Methods			Advantages	Disadvantages
Traditional forensic method	Image generation feature-based forensics		Mature technology and interpretable features	Dependence on specific Features; weak universality
	Image operation feature-based forensics	Single operation forensics	Pay attention to the local information of the image	Dependent on specific features
		Multioperation forensics	Learning local information of image	Weak universality and insufficient robustness
Deep learning-based forensics		Deep neural network-based forensics	Large amount of data; more learning information and high accuracy	Dependence on distributed datasets and unknown types have a great impact on performance
		GAN-based forensics	Focus on GAN fingerprint information; high accuracy	Strong data dependence; poor interpretability

model with two input channels. One is for the real frames and the other is for the corresponding Deepfake ones. Paired images are inputs to the GAN model for adversarial training. In order to a better result, anti-forensic abilities are defined and a loss function is designed. Xie et al. [183] proposed an anti-forensic framework called dual-domain generative adversarial network (DDGAN). It consisted of a generator and two discriminators working on the operation-specific feature domain which helps to conceal the artifacts from the perspective of forensic analysis for the target task, and the spatial domain which facilitates taking advantage of machine-learned features from the scratch as a supplementary.

5. Dataset for Image Forensics

To study the feasibility, effectiveness, and robustness of forensic algorithms, different researchers, scientific research institutions and companies have released datasets for forensics, covering all categories. According to the purpose of the dataset, it can be divided into splicing, camera-based forensics, double JPEG compression, copy-move, GAN-generated image forensics, and various manipulations. Their detailed list is shown in Table 8 and its introduction is as follows.

5.1. Dataset for Splicing. The Columbia dataset is one of the first ones made available to the forensic community. The Columbia dataset contains the gray version of Columbia [187] released in 2004 and the color version of Columbia [188] released in 2006. The gray version of the dataset contains 1845 image blocks (128×128 pixels) with different contents extracted from the images on the CalPhotos website, including 933 real image blocks and 912 mosaic image blocks as well as a small group of 10 images taken by the authors. The number of real images and splicing images in the dataset is roughly the same. The real image and the splicing image are separate local blocks of fixed size (128×128 pixels). The blocks are kept at a reasonable size to ensure that the empirical data of each block can be used to estimate sufficiently accurate statistical characteristics. Color version of Columbia dataset comprises a total of uncompressed 363 images. 183 of them are authentic images and 180 are spliced ones. For the splicing image, there was not any type of postprocessing put on the spliced region and it can be identified easily by our eyes. Based on the above reason, fine-tuning, training, and testing are not recommended to perform.

For this reason, CASIA [189] and DSO-1 [190] datasets are released. For the CASIA dataset, it has 1.0 and 2.0 versions. The 1.0 version provided splicing with sharp boundaries and is easily detectable. To meet practical needs, the 2.0 version is released. The tampered image in CASIA 2.0 preprocesses the tampered part before tampering: for example, scaling, deformation, rotation, and other operations are performed on the tampered part before pasting to the target image; some blur operations are carried out on the edge of the tampered area or inside the tampered area. Therefore, the dataset can better represent most of the spliced tampered images in daily life. DSO-1 [190] realistic dataset is a subset taken from the IEEE Image Forensics Challenge. It provided the images with uncompressed PNG format, but most of them had been JPEG compressed before. Unfortunately, DSO-1 only provided fixed image resolution and missed information such as how to create a dataset and how many cameras are used.

5.2. Dataset for Camera-Based Forensics. Dataset for camera-based forensics mainly includes the dataset proposed in [184], Dresden [191], and IMD2020 [192]. For identifying the camera from sensor fingerprints, [184] utilized Flickr to form the dataset which contained 1053580 pictures taken by 6896 cameras covering 150 camera models to evaluate the camera identification based on sensor fingerprint. Dresden database in [191] is specially used for the development and benchmarking of camera-based digital forensic technology. A total of 73 digital cameras collected more than 14000 images of various indoor and outdoor scenes. The camera models involve 25 to ensure that device-specific and model-specific characteristics can be separated and studied separately. In addition, auxiliary images for estimating device-specific sensor noise patterns are collected for each camera. To study the model-specific JPEG compression algorithm, another set of images has been compiled for each model. In Novoz´amsk´y et al. [192], the authors identified the majority of camera brands and models on the market, which resulted in 2,322 camera models. Then, they collected a dataset named IMD2020, which included 35,000 real images captured by these

Table 8: The detailed list of datasets.

Forensic type	Dataset name		Started year	No. original/forged	Image size	Format
Splicing	Columbia	gray	2004	933/912	128×128	BMP
		Color	2006	183/180	757 × 568~1152 × 768	Raw, BMP
	DSO-1		2013	100/100	2048 × 1536	PNG
	CASIA	1.0	2013	800/921	384 × 256	JPG
		2.0	2013	7491/5123	240 × 160 ~ 800 × 600	BMP, TIF, JPG
Camera-based forensics	[184]		2009	1053580/—	1600 × 1200 ~ 3072 × 2048	JPEG
	Dresden		2010	Over 14000/—	2592 × 1944 ~ 4032 × 3024	Raw, JPEG
	IMD2020		2020	37000/—	Various	—
Double JPEG compression	[185]		2012	100/110	256 × 256 ~ 1024 × 1024	TIFF
	[186]		2018	18946	256 × 256	RAW
Copy-move	MICC F220		2011	110/110	722 × 48 ~800 × 600	JPG
	MICC F2000		2011	1,300/700	2048 × 1536	JPG
	FAU		2012	48/48	2362 × 1581~ 3888 × 2592	PNG, JPG
	CoMoFoD		2013	260/160	512 × 512 ~ 3 000 × 2 000	PNG, JPG
	GRIP		2015	80/80	768 × 1024	PNG
	COVERAGE		2016	100/100	400 × 486	TIF
GAN-generated forensics	[159]		2018	More than 18,432/18,432	256 × 256	—
	FFHQ		2019	70000/-	1024 × 1024	PNG
	[160]		2020	3,000/3,000	256 × 256	JPG
	IMD2020		2020	37,000/37,000	Various	—
Various manipulations	NC2017		2017	2667/1410	160 × 120~8000 × 5320	Raw, PNG, BMP, JPG
	PS-battles		2018	11,142/102,028	130 × 60~ 10,000 × 8558	PNG, JPG
	MFC2018		2018	14,156/3,265	128 × 104~ 7952 × 5304	Raw, PNG, BMP, JPG, TIF
	MFC2019		2019	10,279/5,750	160 × 120~2624 × 19680	Raw, PNG, BMP, JPG, TIF
	DEFACTO		2019	–/229,000	240 × 320~ 640 × 640	TIF

camera models. The authors also created a dataset of 2,000 real-life manipulated images and corresponding real images.

5.3. Dataset for Double JPEG Compression. For testing double JPEG compression, Bianchi et al. [185] provided a realistic dataset that was captured by three different digital cameras. In 2018, for training deep networks, Yang et al. [138] released a dataset by collecting 1120 different quantization tables from actual requested images which were generated by mixing all quality factors.

5.4. Dataset for Copy-Move. For copy-move, two ground truth databases for CMFD algorithms, called MICC F220 and MICC F2000 consisting of 200 and 2000 images, respectively, were released by Park et al. [186].In each of these datasets, half of the images have been tampered with. The image size is 2048 × 1536 pixels. The type of processing on the copy-move forgeries is limited to rotation and scaling. Additionally, the source files are not available. Thus, adding noise or other artifacts to the copied region is not feasible. To address these issues, many datasets have been released by [99, 193–196]. Some of them simply copy one object from one position to paste it to another position in the image, and some of them copy an object and then use rotation, resizing, and change of illumination operation on it to express the actual operation as truthfully as possible.

5.5. Dataset for GAN-Generated Image Forensics. For the detection of GAN-generated images, the current datasets mainly include FFHQ, CelebA-HQ, IMD2020, and other GAN-generated datasets [159]. The following mainly introduces FFHQ, CelebA-HQ, and IMD2020.

FFHQ (Flickr-faces-high quality), originally created as the benchmark of GAN, is also used in the training dataset of StyleGAN [54, 55] and is open-sourced by NVIDIA in 2019. FFHQ is a high-quality face dataset containing 70,000 high-definition face images in PNG format with a resolution of 1024 × 1024. They are rich and diverse in age, race (many face attributes), and image background and have obvious differences.

The datasets in Mi et al. [160] focused on human faces which consisted of the real face set with 30,000 images from the CelebA-HQ dataset [197], and the fake face set (GAN-generated human face images) with 30,000 images that NVIDIA open-sourced as the exemplary images generated by PGGAN [180]. In order to control variables and rule out possible interference, the 1024 × 1024 images from both sets are resized to 256 × 256 using bilinear interpolation.

Marra et al. [159] built a large dataset of samples of different categories by image-to-image translation using the code available online (https://github.com/junyanz/CycleGAN). The dataset included more than 36K 256 × 256 color images; and half of them are real images, and the other half are GAN generated.

In the study by Novoz´amsk´y et al. [192], the authors also created the same number of digitally manipulated images by using a large variety of main image manipulation methods as well as advanced ones such as GAN or inpainting resulting in a dataset of 70,000 images. The real versions of these images are also provided. The authors also manually created binary masks localizing the exact manipulated areas of these images.

5.6. Dataset for Various Manipulations. To test the robustness of the algorithms, NC2016, NC2017, MFC2018, and MFC2019 [197] are released by the U.S. National Institute of Standards and Technology (NIST). For the NC2016 has some redundancies, for example, each spliced photo is presented four times, JPEG compressed at low and high quality, and with and without postprocessing on the splicing boundaries. Therefore, NIST published NC2017, MFC2018, and MFC2019 datasets in 2017, 2018, and 2019 without the above redundancies. These datasets presented the various types of manipulations, resolutions, formats, compression levels, and acquisition devices. Moreover, multiple manipulations are often carried out on the same image and even on the same objects. Overall, they represent a very challenging and reliable testbed for new proposals. Heller et al. [198] presented the PS-Battles dataset which was gathered from a large community of image manipulation enthusiasts and provided a basis for media derivation and manipulation detection in the visual domain. It consisted of 102,028 images grouped into 11,142 subsets, each containing the original image as well as a varying number of manipulated derivatives.

Mahfoudi et al. [199] presented a dataset named DEFACTO for image and face manipulation detection and localization. The DEFACTO was automatically generated using Microsoft common object in context database (MSCOCO) to produce semantically meaningful forgeries. It generated Splicing forgeries, copy-move forgeries, object removal forgeries, and morphing forgeries. Over 200000 images have been generated, and each image is accompanied by several annotations allowing precise localization of the forgery and information about the tampering process.

6. Summary and Prospect

6.1. Summary. With the development of image inpainting technology, image forgery technology has become more and more sophisticated. This brought great negative impact to the country and society. At the national level, the negative impact of forged pictures will have an irreversible impact on the government or politicians. If the evidence of forged pictures is accepted by the court, it will doubt the credibility of the government. At the social level, forged pictures of individuals seriously damage personal reputation. The negative of forged pictures make citizens lose confidence in the stock market. Forged pictures are certified by equipment, which has a huge security risk to personal credit and assets. These risks also hide deeper social issues such as national security and stability, ethics, economic development, and trust crises, and more effective countermeasures are urgently needed.

6.2. Prospect. At present, the forensic method based on specific features has formed the forensic theory and achieved fruitful research results, which has gotten rid of the trap of forgery to a certain extent. However, we should also be aware that there are still many key problems to be solved in this field, especially the rapid development of image inpainting technology, which makes identification and forensics in a passive and disadvantageous position. To avoid this dilemma and look forward to the future, we can consider exploring the feasible direction of detection and forensics in the future from multiple angles and levels. At the national level, relevant laws and regulations are promulgated to clearly stipulate the responsibility for forging and tampering with images and to clarify the behavioral norms of forgers. At the social level, for researchers, the universal and robust detection algorithms are studied, as many deep forgery types as possible are explored, and the common features are found. By learning the common features, the detection model can be applied to more forgery types. Second, scattered data resources are concentrated and a unified detection and forensic group is established, so that researchers can make better use of existing resources and achievements, regularly hold academic seminars and competitions, and increase researchers' attention to the field of deep forgery detection. Finally, with the development of existing traceability technology and blockchain technology, forensics can gain some inspiration and combining them with current forensic technology can achieve complementary advantages.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant no. 61862051; the Science and Technology Foundation of Guizhou Province under Grant no. [2019]1447; the Youth Philosophy and Social Science Foundation of Guizhou Province under Grant no. 18GZQN36; the Nature Science Foundation of Educational Department under Grant nos. [2022]094 and [2022]100; and the Nature Science Foundation of Qiannan Normal University for Nationalities under Grant no. QNSYRC201714.

References

[1] H. Farid, "Digital image forensics," *Scientific American*, vol. 298, no. 6, pp. 66–71, 2008.
[2] wwwchinadailycom, "www.chinadaily.com," 2015, http://www.chinadaily.com.cn/interface/yidian/1120783/2015-12-04/cd.22630382.html.
[3] politics.people.com, "politics.people.com," 2015, http://politics.people.com.cn/n/2015/0303/c70731-26629561.html.
[4] Deepfakes, "Deepfakes," 2019, https://github.com/deepfakes/faceswap.
[5] ZaoApp, "Zao App," 2019, https://zao-app.com/.

[6] H. Farid, "How to detect faked photos," *American Scientist*, vol. 105, no. 2, pp. 77–81, 2017.

[7] E. Kee and J. F. O, Brien and H. Farid, "Exposing photo manipulation from shading and shadows," *ACM Transactions on Graphics*, vol. 33, no. 5, pp. 1–21, 2014.

[8] S. Pittala, E. Whiting, and H. Farid, "A 3-d stability analysis of lee Harvey Oswald in the backyard photo," *Journal of Digital Forensics, Security and Law*, vol. 10, no. 3, 2015.

[9] H. F. Tang and Y. F. Dong, "Survey of image inpainting algorithms based on deep learning," *Computer Science*, vol. 47, no. S2, pp. 151–164, 2020.

[10] M. H. Yap, N. Batool, and C. Ng, "A Survey on Facial Wrinkles Detection and Inpainting: Datasets, Methods, and Challenges," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, 2021.

[11] M. C. Stamm, M. Min Wu, and K. J. R. Liu, "Information forensics: an overview of the first decade," *IEEE Access*, vol. 1, pp. 167–200, 2013.

[12] E. Nowroozi, A. Dehghantanha, R. M. Parizi, and K.-K. R. Choo, "A survey of machine learning techniques in adversarial image forensics," *Computers & Security*, vol. 100, Article ID 102092, 2021.

[13] Z. P. Chen, *Research on Digital Image Forensics of Processing History*, Beijing Jiaotong University, Beijing, China, 2019.

[14] Y. Liang, *Research on Key Technologies of Digital Image Forensics Based on Deep Neural Network*, Xidian University, Xi'an, China, 2019.

[15] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1200–1212, 2004.

[16] H. Wang, L. Jiang, R. Liang, and X.-X. Li, "Exemplar-based image inpainting using structure consistent patch matching," *Neurocomputing*, vol. 269, pp. 90–96, 2017.

[17] Z. Chen, C. Dai, L. Jiang et al., "Structure-aware image inpainting using patch scale optimization," *Journal of Visual Communication and Image Representation*, vol. 40, pp. 312–323, 2016.

[18] A. S. M. Jiao, P. W. M. Tsang, and T.-C. Poon, "Restoration of digital off-axis Fresnel hologram by exemplar and search based image inpainting with enhanced computing speed," *Computer Physics Communications*, vol. 193, pp. 30–37, 2015.

[19] C. Su, T. Fu, X. Zhang, J. Ren, and L. Jin, "Adaptively-weighted blind image restoration algorithm based on energy constraint," *Acta Optica Sinica*, vol. 38, no. 2, Article ID 0210001, 2018.

[20] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," *Lecture Notes in Computer Science*, vol. 9, no. 1, pp. 599–619, 2012.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Advances in Neural Information Processing Systems*, pp. 1097–1105, MIT Press, Cambridge, MA, USA, 2012.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[23] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014, https://arxiv.org/abs/1409.1556.

[24] F. Chollet, "Xception: deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition (CVPR)*, pp. 1251–1258, Honolulu, HI, USA, July 2017.

[25] C. Szegedy, W. Liu, and Y. Jia, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, Boston, MA, USA, June 2015.

[26] G. Huang, Z. Liu, and L. Van Der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition (CVPR)*, pp. 4700–4708, Honolulu, HI, USA, July 2017.

[27] H. Jie, S. Li, and S. Albanie, "Squeeze-and-Excitation Networks," 2017, https://arxiv.org/abs/1709.01507.

[28] J. Gu, Z. Wang, J. Kuen et al., "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[29] K. Sasaki and S. Iizuka, "Joint gap detection and inpainting of line drawings," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5725–5733, IEEE, Honolulu, HI, USA, June 2017.

[30] D. Pathak, P. Krahenbuhl, and J. Donahue, "Context encoders: feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2536–2544, IEEE, Las Vegas, NV, USA, June 2016.

[31] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European Conferenceon Computer Vision (ECCV)*, pp. 89–105, Glasgow, UK, August 2018.

[32] R. A. Yeh, C. Chen, and T. Y. Lim, "Image inpainting with deep generative models," in *Proceedings of the 2017 IEEE-Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5485–5489, IEEE, New York, NY, USA, June 2017.

[33] Y. Zeng, J. Fu, H. Chao, and B. Guo, "Learning pyramid-context encoder network for high- quality image inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1486–1494, IEEE, Long Beach, CA, USA, June 2019.

[34] J. Yu, Z. Lin, and J. Yang, "Free-form image inpainting with gated convolution," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4471–4480, Long Beach, CA, USA, June 2019.

[35] C. Yang, X. Lu, and Z. Lin, "High-resolution image inpainting using multi-scale neural patch synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6721–6729, IEEE, New York, NY, USA, June 2017.

[36] X. Hong, P. Xiong, and R. Ji, "Deep fusion network for image completion," in *Proceedings of the 27th ACM International Conference on Multimedia*, pp. 2033–2042, France, October 2019.

[37] Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan, "Shift-net: image inpainting via deep feature rearrangement," 2018, https://arxiv.org/abs/1801.09392.

[38] N. Wang and J. Li, "MUSICAL: Multi-Scale Image Contextual Attention Learning for Inpainting," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 1–19, Yokohama, August 2019.

[39] H. Liu, "Coherent Semantic Attention for Image Inpainting," in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)* IEEE, Seoul, Korea (South), June 2019.

[40] T. Yu, Z. Guo, X. Jin et al., "Region normalization for image inpainting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, Article ID 12733, 2020.

[41] I. Goodfellow, J. Pouget-Abadie, and M. Mirza, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[42] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks in computer vision," *ACM Computing Surveys*, vol. 54, no. 2, pp. 1–38, 2022.

[43] M. Mehdi and O. Simon, "Conditional generative adversarial nets," 2014, https://arxiv.org/abs/1411.1784.

[44] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proceedings of the International Conference on Machine Learning (PMLR)*, pp. 2642–2651, Paris, France, July 2017.

[45] X. Chen and Y. Duan, "Infogan: interpretable representation learning by information maximizing generative adversarial nets," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NeuralIPS)*, pp. 2180–2188, Seoul, Korea (South), December 2016.

[46] Y. Yu, Z. Gong, P. Zhong, and J. Shan, "Unsupervised representation learning with deep convolutional neural network for remote sensing images," in *Proceedings of the International Conference on Image and Graphics*, pp. 97–108, Springer, China, September 2017.

[47] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–14, 2017.

[48] H. Zhang, T. Xu, and H. Li, "Stackgan: text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer vision (ICCV)*, pp. 5907–5915, IEEE, Venice, Italy, October 2017.

[49] H. Zhang, T. Xu, H. Li et al., "Stackgan++: realistic image synthesis with stacked generative adversarial networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1947–1962, 2019.

[50] J. Y. Zhu, T. Park, and P. Isola, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer vision (ICCV)*, pp. 2242–2251, IEEE, Venice, Italy, August 2017.

[51] J. Yu, Z. Lin, and J. Yang, "Generative image inpainting with contextual attention," in *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition (CVPR)*, pp. 5505–5514, Salt Lake City, UT, USA, October 2018.

[52] C. Zheng, T. J. Cham, and J. Cai, "Pluralistic image completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1438–1447, Long Beach, CA, USA, June 2019.

[53] T. Karras, T. Aila, and S. Laine, "Progressive Growing of gans for Improved Quality, Stability, and Variation," 2017, https://arxiv.org/abs/1710.10196.

[54] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4401–4410, Long Beach, CA, USA, June 2019.

[55] T. Karras, S. Laine, and M. Aittala, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8110–8119, Seattle, WA, USA, June 2020.

[56] A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," 2018, https://arxiv.org/abs/1809.11096.

[57] X. Mao, Q. Li, and H. Xie, "Least squares generative adversarial networks," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2813–2821, IEEE, Istanbul, Turkey, January 2018.

[58] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017, https://arxiv.org/abs/1701.07875.

[59] I. Gulrajani, F. Ahmed, and M. Arjovsky, "Improved Training of Wasserstein GANs," 2017, https://arxiv.org/abs/1704.00028.

[60] H. Li, G. Li, L. Lin, H. Yu, and Y. Yu, "Context-aware semantic inpainting," *IEEE Transactions on Cybernetics*, vol. 49, no. 12, pp. 4398–4411, 2019.

[61] H. Zhang, I. Goodfellow, and D. Metaxas, "Self-attention generative adversarial networks," in *Proceedings of the International Conference on Machine Learning (PMLR)*, pp. 7354–7363, Paris, France, May 2019.

[62] A. R¨ossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Faceforensics++: learning to detect manipulated facial images," in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1–11, Seoul, Korea (South), November 2019.

[63] H. Naeem, F. Ullah, M. R. Naeem et al., "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model," *Ad Hoc Networks*, vol. 105, Article ID 102154, 2020.

[64] L. Zhu, J. Wang, X. Luo, and Y. Zhang, "PRUDA: A Novel Measurement Attribute Set towards Robust Steganography in Social Networks," *Security and Communication Networks*, vol. 2021, Article ID 9864833, 13 pages, 2021.

[65] H. Naeem, "Detection of malicious activities in internet of things environment based on binary visualization and machine intelligence," *Wireless Personal Communications*, vol. 108, no. 4, pp. 2609–2629, 2019.

[66] wwwdarpamil, "www.darpa.mil," 2019, https://www.darpa.mil/program/media-forensics.

[67] T. C. Ifs, "The 1st IEEE IFS-tc image forensics challenge," 2013, http://ifc.recod.ic. unicamp.br/fc.website/index.py.

[68] N. Krawetz, "FotoForensics, Neal Krawetz," 2020, http://fotoforensics.com.

[69] Multimedia Computing Laboratory, "MMC Image Forensic Tool image Watermarking Tool," 2015, http://rtlab.kaist.ac.kr.

[70] S. Duan, H. Wang, and Y. Liu, "A Novel Comprehensive Watermarking Scheme for Color Images," *Security and Communication Networks*, vol. 2020, Article ID 8840779, 15 pages, 2020.

[71] W. Qi, Y. Liu, S. Guo, and X. Wang, "An Adaptive Visible Watermark Embedding Method Based on Region Selection," *Security and Communication Networks*, vol. 2021, Article ID 6693343, 10 pages, 2021.

[72] P. Zhang, L. Wang, and W. Wang, "A Blockchain System Based on Quantum-Resistant Digital Signature," *Security and Communication Networks*, vol. 2021, Article ID 6671648, 13 pages, 2021.

[73] M. K. Johnson and H. Farid, "Exposing digital forgeries by detecting inconsistencies in lighting," in *Proceedings of the 7th Workshop on Multimedia and Security*, pp. 1–10, New York, NY, USA, August 2005.

[74] Q. Qiguang Liu, X. Xiaochun Cao, C. Chao Deng, and Xiaojie Guo, "Identifying image composites through shadow matte consistency," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1111–1122, 2011.

[75] H. Farid and J. Kosecka, "Estimating planar surface orientation using bispectral analysis," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2154–2160, 2007.

[76] Y. Zhang, T. Liu, C. Cattani, Q. Cui, and S. Liu, "Diffusion-based Image Inpainting Forensics via Weighted Least Squares Filtering Enhancement," *Multimedia Tools and Applications*, vol. 80, pp. 1–15, 2021.

[77] U. A. Ciftci, I. Demir, and L. Yin, "Fakecatcher: Detection of Synthetic Portrait Videos Using Biological Signals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, 2020.

[78] H. Yao, S. Wang, Y. Zhao, and X. Zhang, "Detecting image forgery using perspective constraints," *IEEE Signal Processing Letters*, vol. 19, no. 3, pp. 123–126, 2012.

[79] V. Itier, O. Strauss, L. Morel, and W. Puech, "Color noise correlation-based splicing detection for image forensics," *Multimedia Tools and Applications*, vol. 80, no. 9, Article ID 13215, 2021.

[80] J. Luka, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006.

[81] A. K. Jaiswal and R. Srivastava, "Forensic image analysis using inconsistent noise pattern," *Pattern Analysis & Applications*, vol. 24, no. 2, pp. 655–667, 2021.

[82] F. Bellavia, M. Fanfani, C. Colombo, and A. Piva, "Experiencing with electronic image stabilization and PRNU through scene content image registration," *Pattern Recognition Letters*, vol. 145, pp. 8–15, 2021.

[83] D. Cozzolino, F. Marra, D. Gragnaniello, G. Poggi, and L. Verdoliva, "Combining PRNU and noiseprint for robust and efficient device source identification," *EURASIP Journal on Information Security*, vol. 2020, no. 1, 2020.

[84] X. Lin and C. T. Li, "On constructing A better correlation predictor for PRNU-based image forgery localization," in *Proceedings of the 2021 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, IEEE, Shenzhen, China, July 2021.

[85] A. C. Popescu and H. Farid, "Exposing digital forgeries in color filter array interpolated images," *IEEE Transactions on Signal Processing*, vol. 53, no. 10, pp. 3948–3959, 2005.

[86] P. Ferrara, T. Bianchi, A. De Rosa, and A. Piva, "Image forgery localization via fine-grained analysis of CFA artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1566–1577, 2012.

[87] W. Luo, J. Huang, and G. Qiu, "JPEG error analysis and its applications to digital image forensics," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 3, pp. 480–491, 2010.

[88] F. Huang, J. Huang, and Y. Q. Shi, "Detecting double JPEG compression with the same quantization matrix," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 848–856, 2010.

[89] S. Milani, M. Tagliasacchi, and S. Tubaro, "Discriminating multiple JPEG compressions using first digit features," *APSIPA Transactions on Signal and Information Processing*, vol. 3, no. 1, 2014.

[90] M. C. Stamm and K. J. R. Liu, "Forensic detection of image manipulation using statistical intrinsic fingerprints," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 3, pp. 492–506, 2010.

[91] G. Cao, Y. Zhao, R. Ni, and X. Li, "Contrast enhancement-based forensics in digital images," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 3, pp. 515–525, 2014.

[92] H. Zou, P. Yang, R. Ni, and Y. Zhao, "Anti-Forensics of Image Contrast Enhancement Based on Generative Adversarial Network," *Security and Communication Networks*, vol. 2021, Article ID 6663486, 8 pages, 2021.

[93] J. Wu, T. Tong, and Y. Chen, "An adversarial learning framework with cross-domain loss for median filtered image restoration and anti-forensics," *Computers & Security*, vol. 112, Article ID 102497, 2021.

[94] M. Kirchner and J. Fridrich, "On detection of median filtering in digital images," in *Media forensics and security II* vol. 7541, International Society for Optics and Photonics, Article ID 754110, 2010.

[95] A. C. Popescu and H. Farid, "Exposing digital forgeries by detecting traces of resampling," *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 758–767, 2005.

[96] B. Mahdian and S. Saic, "Blind authentication using periodic properties of interpolation," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 529–538, 2008.

[97] D. Vazquez-Padin, F. Perez-Gonzalez, and P. Comesana-Alfaro, "A random matrix approach to the forensic analysis of upscaled images," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 9, pp. 2115–2130, 2017.

[98] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," *ACM SIGGRAPH 2007 papers on - SIGGRAPH'07*, vol. 26, no. 3, pp. 10–16, 2007.

[99] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, "A sift-based forensic method for copy-move attack detection and transformation recovery," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1099–1110, 2011.

[100] Z. Lin, J. He, X. Tang, and C.-K. Tang, "Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis," *Pattern Recognition*, vol. 42, no. 11, pp. 2492–2501, 2009.

[101] Z. Zhou, Y. Li, Y. Zhang, Z. Yin, L. Qi, and R. Ma, "Residual visualization-guided explainable copy-relationship learning for image copy detection in social networks," *Knowledge-Based Systems*, vol. 228, Article ID 107287, 2021.

[102] T. Bianchi and A. Piva, "Detection of nonaligned double JPEG compression based on integer periodicity maps," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 842–848, 2012.

[103] S. Bayram, H. T. Sencar, and N. Memon, "An efficient and robust method for detecting copy-move forgery," in *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1053–1056, IEEE, Taipei, Taiwan, April 2009.

[104] Q. Liu and Z. Chen, "Improved approaches with calibrated neighboring joint density to steganalysis and seam-carved forgery detection in JPEG images," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 4, pp. 1–30, 2014.

[105] Q. Liu, "An approach to detecting JPEG down-recompression and seam carving forgery under recompression anti-forensics," *Pattern Recognition*, vol. 65, pp. 35–46, 2017.

[106] S. Taspinar, M. Mohanty, and N. Memon, "PRNU-based camera attribution from multiple seam-carved images," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 12, pp. 3065–3080, 2017.

[107] Q. Liu, "An improved approach to exposing JPEG seam carving under recompression," *IEEE Transactions on Circuits*

and Systems for Video Technology, vol. 29, no. 7, pp. 1907–1918, 2019.

[108] J. Ye and Y. Shi, "A local derivative pattern based image forensic framework for seam carving detection," in *Proceedings of the International Workshop on Digital Watermarking*, pp. 172–184, Springer, Guildford UK, September 2016.

[109] D. Zhang, T. Yin, G. Yang, M. Xia, L. Li, and X. Sun, "Detecting image seam carving with low scaling ratio using multi-scale spatial and spectral entropies," *Journal of Visual Communication and Image Representation*, vol. 48, pp. 281–291, 2017.

[110] D. Zhang, G. Yang, F. Li, J. Wang, and A. K. Sangaiah, "Detecting seam carved images using uniform local binary patterns," *Multimedia Tools and Applications*, vol. 79, no. 13-14, pp. 8415–8430, 2020.

[111] D. Zhang, X. Chen, and F. Li, "Seam-carved Image Tampering Detection Based on the Cooccurrence of Adjacent LBPs," *Security and Communication Networks*, vol. 2020, Article ID 8830310, 12 pages, 2020.

[112] M. Lu and S. Niu, "Detection of Image Seam Carving Using a Novel Pattern," *Computational Intelligence and Neuroscience*, vol. 2019, Article ID 9492358, 15 pages, 2019.

[113] M. Lu, S. Niu, and Z. Gao, "An efficient detection approach of content aware image resizing," *Computers, Materials & Continua*, vol. 64, no. 2, pp. 887–907, 2020.

[114] P. Comesaña, "Detection and information theoretic measures for quantifying the distinguishability between multimedia operator chains," in *Proceedings of the 2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 211–216, IEEE, Costa Adeje, Spain, December 2012.

[115] Z. Chen, Y. Zhao, and R. Ni, "Detection of operation chain: JPEG-Resampling-JPEG," *Signal Processing: Image Communication*, vol. 57, pp. 8–20, 2017.

[116] Z. Chen, B. Tondi, X. Li, R. Ni, Y. Zhao, and M. Barni, "Secure detection of image manipulation by means of random feature selection," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2454–2469, 2019.

[117] V. Conotter, P. Comesana, and F. Perez-Gonzalez, "Forensic detection of processing operator chains: recovering the history of filtered JPEG images," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 11, pp. 2257–2269, 2015.

[118] P. Ferrara, T. Bianchi, and A. De Rosa, "Reverse engineering of double compressed images in the presence of contrast enhancement," in *Proceedings of the 2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, pp. 141–146, IEEE, Pula, Italy, September 2013.

[119] T. Bianchi and A. Piva, "Reverse engineering of double JPEG compression in the presence of image resizing," in *Proceedings of the 2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 127–132, IEEE, Costa Adeje, Spain, December 2012.

[120] X. Chu, Y. Chen, and K. J. R. Liu, "Detectability of the order of operations: an information theoretic approach," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 823–836, 2016.

[121] D. Baracchi, D. Shullani, M. Iuliani, D. Giani, and A. Piva, "Camera Obscura: exploiting in-camera processing for image counter forensics," *Forensic Science International: Digital Investigation*, vol. 38, Article ID 301213, 2021.

[122] Q. Wang and R. Zhang, "Double JPEG compression forensics based on a convolutional neural network," *EURASIP Journal on Information Security*, vol. 2016, no. 1, pp. 23–12, 2016.

[123] J. Bunk, J. H. Bappy, and T. M. Mohammed, "Detection and localization of image forgeries using resampling features and deep learning," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1881–1889, Costa Adeje, Spain, July 2017.

[124] L. Bondi, S. Lameri, and D. Guera, "Tampering detection and localization through clustering of camera-based CNN features," in *Proceedings of the Computer Vision & Patterm Recognition Workshops*, pp. 1855–1864, Hawaii, USA, July 2017.

[125] C. Chen, S. McCloskey, and J. Yu, "Image splicing detection via camera response function analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition (CVPR)*, pp. 5087–5096, Honolulu, USA, June 2017.

[126] D. Cozzolino, G. Poggi, and L. Verdoliva, "Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection," in *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, pp. 159–164, New York, USA, August 2017.

[127] P. Zhou, X. Han, and V. I. Morariu, "Learning rich features for image manipulation detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1053–1061, Salt Lake City, UT, USA, May 2018.

[128] M. Lu and S. Niu, "A detection approach using LSTM-CNN for object removal caused by exemplar-based image inpainting," *Electronics*, vol. 9, no. 5, pp. 858–922, 2020.

[129] J. Chen, X. Kang, Y. Liu, and Z. J. Wang, "Median filtering forensics based on convolutional neural networks," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1849–1853, 2015.

[130] J. H. Bappy, C. Simons, L. Nataraj, B. S. Manjunath, and A. K. Roy-Chowdhury, "Hybrid LSTM and encoder-decoder architecture for detection of image forgeries," *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3286–3300, 2019.

[131] Y. Wu, W. Abd-Almageed, and P. Natarajan, "Image copy-move forgery detection via an end-to-end deep neural network," in *Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1907–1915, IEEE, Lake Tahoe, NV, USA, March 2018.

[132] Y. Wu, W. Abd-Almageed, and P. Natarajan, "BusterNet: detecting copy-move image forgery with source/target localization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 168–184, Glasgow, UK, August 2018.

[133] G. Muzaffer and G. Ulutas, "A new deep learning-based method to detection of copy-move forgery in digital images," in *Proceedings of the 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*, pp. 1–4, IEEE, Istanbul, Turkey, April 2019.

[134] J.-L. Zhong and C.-M. Pun, "An end-to-end dense-InceptionNet for image copy-move forgery detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2134–2146, 2020.

[135] Y. Zhu, C. Chen, G. Yan, Y. Guo, Y. Dong, and Ar-Net, "AR-net: adaptive attention and residual refinement network for copy-move forgery detection," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6714–6723, 2020.

[136] X. Zhu, Y. Qian, X. Zhao, B. Sun, and Y. Sun, "A deep learning approach to patch-based image inpainting

forensics," *Signal Processing: Image Communication*, vol. 67, pp. 90–99, 2018.

[137] X. S. Zhu, Y. J. Qian, B. Sun et al., "Image inpainting forensics algorithm based on deep neural network," *Acta Optica Sinica*, vol. 38, no. 11, pp. 1110005–1110113, 2018.

[138] P. Yang, R. Ni, and Y. Zhao, "Recapture image forensics based on Laplacian convolutional neural networks," in *Proceedings of the International Workshop on Digital Watermarking*, pp. 119–128, Springer, Guangzhou China, September 2016.

[139] Y. Liu and Q. Guan, "Image forgery localization based on multi-scale convolutional neural networks," in *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec)*, pp. 85–90, Innsbruck, Austria, June 2018.

[140] Y. Liu, Q. Guan, and X. Zhao, "Copy-move forgery detection based on convolutional kernel network," *Multimedia Tools and Applications*, vol. 77, no. 14, Article ID 18269, 2018.

[141] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi, "A full-image full-resolution end-to-end-trainable CNN framework for image forgery detection," *IEEE Access*, vol. 8, Article ID 133488, 2020.

[142] Q. Bammey, R. G. V. Gioi, and J. M. Morel, "An adaptive neural network for unsupervised mosaic consistency analysis in image forensics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14194–14204, Seattle, WA, US, June 2020.

[143] C. Yang, H. Li, and F. Lin, "Constrained R-CNN: a general image manipulation detection model," in *Proceedings of the 2020 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, IEEE, London, United Kingdom, July 2020.

[144] P. Zhuang, H. Li, S. Tan, B. Li, and J. Huang, "Image tampering localization using a dense fully convolutional network," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2986–2999, 2021.

[145] H. Li and J. Huang, "Localization of deep inpainting using high-pass fully convolutional network," in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8300–8309, Seoul, Korea (South), November 2019.

[146] M. Barni, Q.-T. Phan, and B. Tondi, "Copy move source-target disambiguation through multi-branch CNNs," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1825–1840, 2021.

[147] Y. Zhang, J. Zhang, and S. Xu, "A hybrid convolutional architecture for accurate image manipulation localization at the pixel-level," *Multimedia Tools and Applications*, vol. 80, pp. 1–16, 2021.

[148] Y. Zhang, F. Ding, S. Kwong, and G. Zhu, "Feature pyramid network for diffusion-based image inpainting detection," *Information Sciences*, vol. 572, no. 2021, pp. 29–42, 2021.

[149] Y. Rao, J. Ni, and H. Xie, "Multi-semantic CRF-based attention model for image forgery detection and localization," *Signal Processing*, vol. 183, Article ID 108051, 2021.

[150] J. Hao, Z. Zhang, S. Yang, D. Xie, and S. Pu, "TransForensics: image forgery localization with dense self-attention," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, Article ID 15055, Nashville, TN, USA, June 2021.

[151] X. Wang, H. Wang, and S. Niu, "An intelligent forensics approach for detecting patch-based image inpainting," *Mathematical Problems in Engineering*, vol. 2020, Article ID 8892989, 10 pages, 2020.

[152] H. Li, B. Li, S. Tan, and J. Huang, "Identification of deep network generated images using disparities in color components," *Signal Processing*, vol. 174, Article ID 107616, 2020.

[153] L. Nataraj, T. M. Mohammed, B. S. Manjunath et al., "Detecting GAN generated fake images using Co-occurrence matrices," *Electronic Imaging*, vol. 31, no. 5, pp. 532–541, 2019.

[154] S. McCloskey and M. Albright, "Detecting GAN-generated imagery using color cues," in *Proceedings of the 2019 IEEE International Conference on Image Processing*, pp. 4584–4588, IEEE, Piscataway, January 2019.

[155] F. Marra and D. Gragnaniello, "Do GANs leave artificial fingerprints?" in *Proceedings of the 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 506–511, San Jose, CA, USA, March 2019.

[156] Y. Ning, S. D. Larry, and F. Mario, "Attributing fake images to GANs: learning and analyzing GAN fingerprints," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7556–7566, Seoul, Korea (South), December 2019.

[157] S. Y. Wang and O. Wang, "CNN-generated images are surprisingly easy to spot for now," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8692–8701, Seattle, WA, USA, June 2020.

[158] H. Mo, B. Chen, and W. Luo, "Fake faces identification via convolutional neural network," in *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, pp. 43–47, Innsbruck, Austria, June 2018.

[159] F. Marra, D. Gragnaniello, D. Cozzolino, and L. Verdoliva, "Detection of GAN-generated fake images over social networks," in *Proceedings of the 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 384–389, Miami, FL, USA, February 2018.

[160] Z. Mi, X. Jiang, T. Sun, and K. Xu, "GAN-generated image detection with self-attention mechanism against GAN generator defect," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 969–981, 2020.

[161] L. Guarnera and O. Giudice, "DeepFake detection by analyzing convolutional traces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 666-667, Seattle, WA, USA, June 2020.

[162] S. Lee, S. Tariq, Y. Shin, and S. S. Woo, "Detecting handcrafted facial image manipulations and GAN-generated facial images using Shallow-FakeFaceNet," *Applied Soft Computing*, vol. 105, Article ID 107256, 2021.

[163] B. Chen, X. Liu, and Y. Zheng, "A robust GAN-generated face detection method based on dual-color spaces and an improved Xception," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[164] B. Chen, W. Tan, Y. Wang, and G. Zhao, "Distinguishing between natural and GAN-generated face images by combining global and local features," *Chinese Journal of Electronics*, vol. 31, no. 1, pp. 59–67, 2022.

[165] G. Tang, L. Sun, and X. Mao, "Detection of GAN-Synthesized Image Based on Discrete Wavelet Transform," *Security and Communication Networks*, vol. 2021, Article ID 5511435, 10 pages, 2021.

[166] X. Wang, S. Niu, and H. Wang, "Image inpainting detection based on multi-task deep learning network," *IETE Technical Review*, vol. 38, no. 1, pp. 149–157, 2021.

[167] Y. Huang, F. Juefei-Xu, and R. Wang, "FakeLocator: Robust Localization of GAN-based Face Manipulations," 2020, https://arxiv.org/abs/2001.09598#:~:text=FakeLocator%3A%20Robust%20Localization%20of%20GAN-Based%20Face%20Manipulations,-Yihao%20Huang%2C%20Felix&text=F.

[168] D. Gragnaniello, D. Cozzolino, and F. Marra, "Are GAN generated images easy to detect? A critical analysis of the state-of-the-art," in *Proceedings of the 2021 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, IEEE, Shenzhen, China, July 2021.

[169] M. Barni, M. C. Stamm, and B. Tondi, "Adversarial multimedia forensics: overview and challenges ahead," in *Proceedings of the 2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 962–966, IEEE, Rome, Italy, September 2018.

[170] J. C. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, H. Proenca, and J. Fierrez, "GANprintR: improved fakes and evaluation of the state of the art in face manipulation detection," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 1038–1048, 2020.

[171] X. Zhang, S. Karaman, and S. F. Chang, "Detecting and simulating artifacts in gan fake images," in *Proceedings of the 2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, IEEE, Delft, Netherlands, December 2019.

[172] M. Du, S. Pentyala, and Y. Li, "Towards Generalizable Forgery Detection with Locality-Aware Autoencoder," 2019, https://arxiv.org/abs/1909.05999.

[173] D. Cozzolino, J. Thies, and A. Rossler, "Spoc: spoofing camera fingerprints," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 990–1000, Nashville, TN, USA, June 2021.

[174] J. Wu and W. Sun, "Towards multi-operation image anti-forensics with generative adversarial networks," *Computers & Security*, vol. 100, Article ID 102083, 2021.

[175] J. Brockschmidt, J. Shang, and J. Wu, "On the generality of facial forgery detection," in *Proceedings of the 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*, pp. 43–47, IEEE, Monterey, CA, USA, November 2019.

[176] R. Huang, F. Fang, and H. Nguyen, "Security of facial forensics models against adversarial attacks," in *Proceedings of the 2020 IEEE International Conference on Image Processing (ICIP)*, pp. 2236–2240, IEEE, Abu Dhabi, United Arab Emirates, October 2020.

[177] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of the 3rd International Conference on Leaning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[178] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proceedings of the 5th International Conference on Leaning Representations Workshop (ICLRW)*, Toulon, France, April 2017.

[179] H. Zeng, J. Liu, J. Yu, X. Kang, Y. Q. Shi, and Z. J. Wang, "A framework of camera source identification Bayesian game," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1757–1768, 2017.

[180] M. Barni and B. Tondi, "Adversarial source identification game with corrupted training," *IEEE Transactions on Information Theory*, vol. 64, no. 5, pp. 3894–3915, 2018.

[181] B. Tondi, N. Merhav, and M. Barni, "Detection games under fully active adversaries," *Entropy*, vol. 21, no. 1, p. 23, 2018.

[182] F. Ding, G. Zhu, Y. Li, and X. Zhang, "Anti-Forensics for Face Swapping Videos via Adversarial Training," *IEEE Transactions on Multimedia*, 2021.

[183] H. Xie, J. Ni, and Y. Q. Shi, "Dual-Domain Generative Adversarial Network for Digital Image Operation Anti-forensics," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, 2021.

[184] M. Goljan, J. Fridrich, and T. Filler, "Large scale test of sensor fingerprint camera identification," *Proceedings SPIE Media Forensics and Security*, vol. 7254, 2009.

[185] T. Bianchi and A. Piva, "Image forgery localization via block-grained analysis of JPEG artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1003–1017, 2012.

[186] J. Park and D. Cho, "Double JPEG detection in mixed JPEG quality factors using deep convolutional neural network," in *Proceedings of the European Conference on Computer Vision*, Munich, Germany, September 2018.

[187] T. T. Ng, S. F. Chang, and Q. Sun, "A Data Set of Authentic and Spliced Image Blocks," Columbia University, ADVENT, Technical Report, , pp. 203–2004, 2004.

[188] Y. F. Hsu and S. F. Chang, "Detecting image splicing using geometry invariants and camera characteristics consistency," in *Proceedings of the 2006 IEEE International Conference on Multimedia and Expo*, pp. 549–552, IEEE, Ontario, Canada, July 2006.

[189] J. Dong, W. Wang, and T. Tan, "Casia image tampering detection evaluation database," in *Proceedings of the IEEE China Summit and International Conference on Signal and Information Processing*, pp. 422–426, Chengdu, China, July 2013.

[190] T. J. de Carvalho, C. Riess, E. Angelopoulou, H. Pedrini, and A. de Rezende Rocha, "Exposing digital image forgeries by illumination color classification," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 7, pp. 1182–1194, 2013.

[191] T. Gloe and R. Böhme, "The' Dresden Image Database' for benchmarking digital image forensics," *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC'10*, vol. 3, pp. 150–159, 2010.

[192] A. Novoz´amsk´y, B. Mahdian, and S. Saic, "IMD2020: a large-scale annotated dataset tailored for detecting manipulated images," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (CVPRW)*, pp. 71–80, Long Beach, CA, USA, June 2020.

[193] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou, "An evaluation of popular copy-move forgery detection approaches," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1841–1854, 2012.

[194] D. Cozzolino, G. Poggi, and L. Verdoliva, "Efficient dense-field copy-move forgery detection," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 11, pp. 2284–2297, 2015.

[195] D. Tralic, I. Zupancic, and M. Grgic, "CoMoFoD - new database for copy-move forgery detection," in *Proceedings of the 55th International Symposium ELMAR*, pp. 49–54, Zadar, Croatia, September 2013.

[196] B. Wen, Y. Zhu, and R. Subramanian, "COVERAGE-a novel database for copy-move forgery detection," in *Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP)*, pp. 161–165, IEEE, Phoenix, Arizona, September 2016.

[197] H. Guan, M. Kozak, E. Robertson et al., "MFC datasets: large-scale benchmark datasets for media forensic challenge evaluation," in *Proceedings of the 2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pp. 63–72, HI, USA, January 2019.

[198] S. Heller, L. Rossetto, and H. Schuldt, "The Ps-Battles Dataset-An Image Collection for Image Manipulation Detection," 2018, https://arxiv.org/abs/1804.04866.

[199] G. Mahfoudi, B. Tajini, and F. Retraint, "DEFACTO: image and face manipulation dataset," in *Proceedings of the 2019 27th European Signal Processing Conference (EUSIPCO)*, pp. 1–5, IEEE, Coruña, Spain, September 2019.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 1322264, 17 pages
https://doi.org/10.1155/2022/1322264

Research Article

Proving Reliability of Image Processing Techniques in Digital Forensics Applications

Saima Iqbal,[1] Wilayat Khan,[1] Abdulrahman Alothaim,[2] Aamir Qamar,[1] Adi Alhudhaif,[3] and Shtwai Alsubai[3]

[1]*Department of Electrical and Computer Engineering, COMSATS University Islamabad, Wah Campus, Islamabad, Pakistan*
[2]*Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh 11451, Saudi Arabia*
[3]*Department of Computer Science, College of Computer Engineering and Sciences in Al-Kharj, Prince Sattam Bin Abdulaziz University, Al-Kharj, Saudi Arabia*

Correspondence should be addressed to Wilayat Khan; wilayat@ciitwah.edu.pk

Received 17 November 2021; Revised 28 January 2022; Accepted 5 February 2022; Published 31 March 2022

Academic Editor: Farhan Ullah

Binary images have found its place in many applications, such as digital forensics involving legal documents, authentication of images, digital books, contracts, and text recognition. Modern digital forensics applications involve binary image processing as part of data hiding techniques for ownership protection, copyright control, and authentication of digital media. Whether in image forensics, health, or other fields, such transformations are often implemented in high-level languages without formal foundations. The lack of formal foundation questions the reliability of the image processing techniques and hence the forensic results loose their legal significance. Furthermore, counter-forensics can impede or mislead the forensic analysis of the digital images. To ensure that any image transformation meet high standards of safety and reliability, more rigorous methods should be applied to image processing applications. To verify the reliability of these transformations, we propose to use formal methods based on theorem proving that can fulfil high standards of safety. To formally investigate binary image processing, in this paper, a reversible formal model of the binary images is defined in the Proof Assistant Coq. Multiple image transformation methods are formalized and their reliability properties are proved. To analyse real-life RGB images, a prototype translator is developed that reads RGB images and translate them to Coq definitions. As the formal definitions and proof scripts can be validated automatically by the computer, this raises the reliability and legal significance of the image forensic applications.

1. Introduction

Image processing in general is widely used in the computer-based tools and techniques in different applications including digital forensics, health, crack detection in concrete, obstacle detection, and astronomy. The modern quality digital cameras and the sophisticated photo editing tools have made image-based applications very attractive for people, in particular, in social media and mobile applications. However, the same powerful photo editing software have made digital image fabrication and forgery very easy [1]. Such forgeries appear authentic to human eye but are unacceptable to the law enforcement agencies [2]. Image processing techniques are used for identifying fabricated documents using digital forensics analysis [3]. The reliability of the processes implemented by the digital forensics practitioners, however, must be ensured [4]. In a typical digital forensics application, the raw binary data is normally derived as a binary image. During the forensic examination of a hard drive, for example, a binary image of the entire hard drive is created and analysed. Image forensic has been used in forensic document and algorithmic handwriting analysis [5]. The applications of the binary and grayscale images are so versatile that they create the need of their forensic analysis. The digital forensics analysis is used to find malicious manipulations in the image [6]. In fact, digital

image forensic has emerged as a new research field aimed at validating the origin authenticity of the images [7]. Nevertheless, the counter-forensics has also risen up to impede or mislead forensic analysis of the digital images [8].

In digital forensics applications or otherwise, binary images are often quantized to two extreme tones/intensity values, where black (0 intensity value) is usually represented with a 1, whilst white (255 intensity value) is represented as 0. Binary images are usually created by an information abstraction process from a gray channel after applying either thresholding or any other segmentation method. A standard RGB image is comprised of three gray channels, packed in a predefined sequence of the red, green, and blue colors [9]. A binary conversion leads to a significant loss of information; however, if this conversion process is handled with care and an appropriate segmentation algorithm is applied, salient/key information can be preserved. Formally, the binary operations are carried out using a sliding window of variable sizes (e.g., 3×3 or 5×5). Conventionally, a window center plays a pivotal role and the standard operations including median, mean, mode, and so on, are carried out to update the center location of the sliding window. Amongst several, a few operations on a binary image are run-length encoding, dilation, erosion, close, open-close, close-open, and skeletonization to name but a few.

The RGB and gray-scale images have more internal details, but they require more storage and bandwidth to store and transmit. Furthermore, applications based on the color or gray-scale images are computationally extensive and require expensive equipment and computer capabilities to process such images [10]. Binary images, on the other hand, have only two gray values, 1 (black) or 0 (white), and require only one bit per pixel storage (regardless of the bit assignment to the black and white colors, the formal definitions and proofs are still valid). Binary images are created in a number of ways: they may be delivered directly by sensors or indirectly, and more often, they are created from the gray-scale images. Among the major objectives of using the binary images are simplified processing, transmission, storage, and printing.

To achieve high standards of security and reliability of applications based on image transformations, they need to be formally verified. To enable formal reasoning, such transformations need to be specified and reasoned about using mathematical tools and techniques. The conventional methods based on simulation and testing can be used for systems verification; however, these methods can only show presence of faults but cannot show their absence. There are tools based on model checking, but they might face state or memory explosion problems. Formal verification methods based on interactive proof assistants, such as Coq and Isabelle/HOL, are more powerful and expressive and do not face state explosion issues. A proof assistant is a computer-based tool that is guided by a proof engineer in the step-by-step proof process. The interactively generated proof script can be checked by the computer.

To the best of our knowledge, there is no mechanized proof of properties of the binary image processing algorithms described in this paper. To formally prove the correctness of the transformations carried over the binary images, the images and the transformations must be defined in a formal language with a proof facility. In this paper, we are using Coq for the formal specification and verification of the binary image processing. In this approach, as depicted in Figure 1, an RGB image is first read and then converted to a binary image. The binary image is translated to formal representation (model) of binary images as defined using the Coq notations. The Coq definition of the binary image is reversible: a binary image matrix can be generated back from the Coq definition of the binary image. In addition to the binary image, a number of transformations on the binary images and their properties are also formally defined. Using the Coq proof commands (tactics), interactive proofs of properties are carried out and then checked using the Coq proof checker facility. At the end, the binary image defined in the Coq is translated back to the binary image matrix of 0's and 1's (see Section 6).

The following are the major contributions of this paper.

(i) Light-weight formal models of the gray-scale and binary images are built in the Coq tool (described in Section 4.1).

(ii) Several image transformations and an image compression technique are formally defined (discussed in Sections 4.2–4.5).

(iii) A number of properties of the binary images and operations over them, such as involution, area size, and distance measures, are stated and proved in Section 5.

(iv) A prototype translator is developed to translate an RGB image to a binary image and generate other Coq definitions. The translator automatically generates a proof that the translation to binary image and other definitions is sound. Furthermore, the translator converts the Coq binary image back to a binary image matrix. The translator and their experimental analysis are given in Section 6.

The formal developments in this paper formalize binary image operations and carry proof of soundness of these operations. The formal proofs of the binary image operations guarantee the soundness of any application based on these operations, including but not limited to forensics, health, and engineering. Complete Coq script including formal definitions of the binary/grayscale images, image transformations, proof of soundness of image transformations, and source code of the translator is available at our GitHub repository at https://github.com/wilstef/binaryimagescoq. The rest of the paper is organized as follows. A review of the research contributions in the binary image processing, formal hardware, and software verification is included in the next section. In Section 3, the Coq Proof Assistant and the binary image processing are introduced. In Section 4, the gray-scale and binary images are formally specified and a number of operations on them are defined. Several properties of the image processing and encoding are defined and proved in Section 5. An experimental analysis of the translator and formal definitions is carried out in Section 6. The paper is concluded in Section 7.

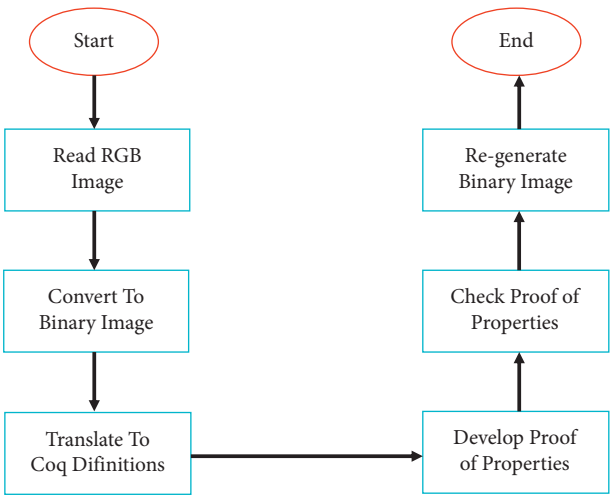


Figure 1: Formal image processing verification approach.

2. Literature Review

In this section, a summary of the research contributions in the binary image processing and formal hardware and software systems' verification is given.

2.1. Binary Image Processing. Binary image processing has found its place in a number of applications including but not limited to agriculture and engineering. Hernández et al. [10] used binary image analysis in their open-source methodology to measure water elevations in two-dimensional hydrodynamic experiments. To detect winding deformation faults in power transformers, frequency response analysis has been widely used as a diagnostic tool. Zhao et al. [11] introduced fault detection method based on the analysis of binary images obtained from frequency response analysis signatures. To detect cracks in concrete using unmanned aerial vehicles, Kim et al. [12] used binary image analysis to transform the crack and background into binary images. They identified the crack objects by categorizing the pixels as black whose gray values are less than a threshold value.

Formal verification has been applied in a number of domains and applications, including image processing tools. Narendran and Stillman formally verified the description of an image processing chip Sobel [13] using theorem proving approach. During their verification attempt, they found several design errors in the circuit description of the design under test. Bourbakis and Alexopoulos [14] defined a special-purpose context-free grammar for the language SCAN. The language SCAN was devoted to describing and generating a range of two-dimensional array accessing algorithms used for image processing. They defined a formal definition of SCAN and described the underlying method for spatial access.

Ehlers [15] presented an approach for formally verifying piecewise linear feed-forward neural networks. Their formally verified approach was evaluated on critical applications, such as obstacle detection and handwritten digit recognition. To protect against adversarial attacks on convolutional neural networks (CNNs), Kouvaros and Lomuscio [16] formally verified CNN-based perception systems. The authors formally verified the robustness of the local image transformations. Sun et al. [17] formally verified the safety of autonomous robot controlled by a neural network. The neural network processes images to produce control actions for avoiding obstacles. None of these researchers have applied proof assistants in the formal verification of these systems.

2.2. Formal Hardware Verification. Formal verification techniques have been applied to study computer hardware systems. Meredith et al. [18] defined an executable semantics by embedding hardware description language (HDL) Verilog in the theorem prover [19]. Inspired from [18], Khan et al. [20, 21] defined an HDL, dubbed as VeriFormal, in the more powerful and expressive higher-order logic of proof assistant Isabelle/HOL. VeriFormal is available with an executable simulator and a prototype translator to translate existing design descriptions in Verilog to VeriFormal. The HDL VeriFormal can be used to design hardware circuits, simulate them, and formally verify their properties using Isabelle/HOL. Later on, the HDL VeriFormal and its simulator were used to formally prove functional equivalence of several logic circuits [22].

Braibant and Chlipala [23] defined a simplified version of the high-level language Bluespec, called Fe-Si. Their language Fe-Si has been deeply embedded in the Proof Assistant Coq. In a similar contribution, Choi et al. [24] developed a formalized version of Bluespec in Coq, called Kami. The objective of platform Kami was to develop high-level parametric hardware specifications. Both, Fe-Si [23] and that of Choi et al. [24], are different from the HDL VeriFormal in the level of hardware specification. The high-level language Bluespec is used for high-level hardware specification, while the HDL VeriFormal is a low-level language at the register-transfer logic level.

2.3. Formal Software Verification. Software systems, in particular those used in business or life critical systems, must be ensured to behave as desired using formal tools. Bugliesi et al. used formal techniques to study web session integrity [25, 26] and web session confidentiality [27, 28]. In addition to the formal analysis, they also developed prototype browser extension as the proof of concept. Khan et al. defined a formal model, dubbed as CrashSafe, of inter-component communication within Android systems in Coq proof assistant. The authors formally showed that their model can be used to capture faults in the Android applications. They later on formally analysed Android systems' security using language-based security techniques [22].

Alturki et al. [29] developed a state-transition formal model of the Algorand consensus protocol and verified its asynchronous safety using Coq proof assistant. The authors claim that their model is general and can be adopted to prove other properties (e.g., liveness) of the protocol. Anand et al. [30] developed and mechanically verified an optimized compiler, called CertiCoq, for Coq programs in Coq proof

assistant. Leroy et al. [31] developed a formally verified compiler, CompCert, for the C programming language. The main advantage of this compiler is that the executable code produced by CompCert is proved to behave as specified by the C semantics.

3. Background

The formal definitions and proofs of binary image processing are carried out in the Proof Assistant Coq. In this section, the basics of Coq and binary image processing are introduced.

3.1. Coq Proof Assistant. To formally verify systems' properties, the system and the properties of interest should be specified in the logic of a proof assistant (e.g., Coq [32, 33] and Isabelle/HOL [34]). The proof assistant can be used to build a proof that the (model of the) system satisfy the properties. The proof checker facility of the proof assistant is then used to mechanically check if the proof is valid. To describe the formal developments and proofs using a proof assistant, a simple system of natural numbers is defined and reasoned about using the proof assistant Coq. To begin with, numbers are inductively defined as data type nat using the keyword Inductive with two constructors for generating elements of the type nat (lines 1–3, Listing 1). The definition of nat states that O (for 0) is nat and if n is nat, then S n is also nat. The term S (S (S (S O))), for example, is a natural number 4 in nat.

Next, we define a recursive function add (lines 5–9) on the numbers just defined. The function returns the second argument m if the first argument is O and it returns S (add n′ m) if the first argument is of the form S n′. A lemma add_n_o, that is add n O = nm holds for any value of n is stated and proved in Listing 1 (lines 11–18). A proof begins with the proof command Proof and ends with Qed. Each proof command, also called tactic, ends with a dot (.), and multiple tactics can be combined into a sequence using semicolon (;). Comments can be introduced anywhere within the script using the syntax (∗ a comment ∗). The lemma add_n_o is proved using induction on the construction of the first argument n. During the proof process, the proof assistant is guided interactively by providing tactics (lines 12–18). The correctness of the proof script just created (lines 12–18 in Listing 1) can be mechanically checked using the Coq proof checker program.

3.2. Binary Images. Binary images consist of picture elements (pixels) with two gray values black and white, denoted as 1 and 0, respectively. The color depth of the binary images is just one bit per pixel. It plays a significant role in the reduced size of such images. Binary image representation is used in a multitude of image applications, including but not limited to digital forensics, optical character recognition, edge detection, optical measurement applications, mathematical morphology, and object tracking and orientations.

The upper binary image in Figure 2 represents the letter N. The resolution of (number of pixels represented) this image is 16 and the depth is one bit per pixel. In the lower image of Figure 2, each pixel is designated with either 1 or 0 depending on its color (black is 1 and white is 0). The bits matrix on the right side of the figure represents the binary image in bits.

4. Formalization of Binary Images

To reason about binary image processing, the concepts of the pixels, binary images, and grayscale images are first defined in the Coq Proof Assistant. In a binary image, every pixel has only two values or colors: black and white. On the contrary, in grayscale images, the pixels' color value ranges over 0–255 different values.

4.1. Binary and Gray-Scale Images. To begin with, first, the data type color is inductively defined in the code Listing 2. The data type color has two possible values or constructors white and black.

A pixel of the binary image is defined as a type pixel as shown in Listing 3. The type pixel is defined as a tuple of the row, column, and a color. It has one constructor pixel that takes three arguments of type nat and color. The first two arguments of the type nat define the row and column position while the third argument defines the color of a pixel. On line 4, the shorthand notation B{r,c,col} is defined to represent a pixel at coordinates (r,c) with color intensity col. Finally, a binary image is defined as a list of binary pixels (Listing 4). To facilitate the reuse of the formal definitions, Coq allows notations. As described in the introduction, the black and white colors are assigned digits 1 and 0, respectively (Listing 5).

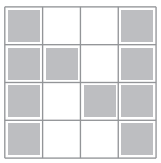
To highlight the application of the formal definitions in Listings 2–4, the binary image in Figure 2 is encoded using the formal definitions developed (lines 1–4, Listing 6). Using the notations for the colors black and white from Listing 5, the same image is encoded more compactly (lines 6–9, Listing 6).

Similarly, a grayscale pixel is defined as a gspixel type in Listing 7. It has a single constructor GSPixel with three arguments of type nat for row, column, and grayscale color of the pixel. The grayscale image gsimage is defined as a list of grayscale pixels (line 4, Listing 7). The notations for binary and grayscale pixels are defined as B{row,column, color} and G{row,column,gs-color}, respectively, where row, column and gs-color are natural numbers, while color is either white or black value (Listings 8 and 9).

4.2. Basic Operations on the Pixels and Images. To state and prove interesting properties of binary and grayscale images, different operations over pixels and images are first defined in Listing 10. These operations include equality of colors eqbcol (lines 1–6), negation of a color negcolor (lines 8–12), negation of a pixel color negpix (lines 14–17), negation of a binary image negimage (lines 19–23), equality of pixels eqpixel (lines 25–29), and negation of a specific pixel in an image negpiximg (lines 31–37), respectively. The first function eqbcol defines when two colors are equal. It returns

```
Inductive nat: Type:=
   |O: nat
   |S: nat ⟶ nat.
Fixpoint add (n m: nat): nat:=
   match n with
   |O ⇒ m
   |S n′ ⇒ S (add n′ m)
   end.
Lemma add_n_o: ∀ n, add n O = n.
Proof.
   induction n.
+ (∗ CASE 1: n is O ∗)
reflexivity.
   + (∗ CASE 2: n is (S n) ∗)
   simpl. rewrite IHn. auto.
Qed.
```

Listing 1: Interactive formal proof in Coq.



1	0	0	1	1	0	0	1
1	1	0	1	1	1	0	1
1	0	1	1	1	0	1	1
1	0	0	1	1	0	0	1

Figure 2: Binary image of the letter N.

```
Inductive color: Type:=
   |white: color
   |black: color.
```

Listing 2: Data type color.

```
Inductive pixel: Type :=
 |Pixel (r c: nat) (col: color).
Notation "B r,c, col" (Pixel r c col).
```

Listing 3: Data type pixel.

```
Definition image: = list pixel.
```

Listing 4: Definition of the binary image.

```
Notation "1": = (black).
Notation "0": = (white).
```

Listing 5: Shorthand notations representing black and white colors.

```
[B{0, 0, black}; B{0, 1, white}; B{0, 2, white; B0, 3, black};
B{1, 0, black}; B{1, 1, black}; B{1, 2, white}; B{1, 3, black}; B{2, 0, black}; B{2, 1, white}; B{2, 2,
black}; B{2, 3, black};
B{3, 0, black}; B{3, 1, white}; B{3, 2, white}; B{3, 3, black}].
[B{0, 0, 1}; B{0, 1, 0}; B{0, 2, 0}; B{0, 3, 1};
B{1, 0, 1}; B{1, 1, 1}; B{1, 2, 0}; B{1, 3, 1};
B{2, 0, 1}; B{2, 1, 0}; B{2, 2, 1}; B{2, 3, 1};
B{3, 0, 1}; B{3, 1,0}; B{3, 2, 0}; B{3, 3, 1}].
```

Listing 6: Encoding binary image of Figure 2.

```
Inductive gspixel: Type :=
  |GSPixel (r c v: nat).
Definition gsimage: = list gspixel.
```

Listing 7: Definition of grayscale pixel and image.

```
Notation "B r, c, col": = (Pixel r c col).
Notation "G r, c, v": = (GSPixel r c v).
```

Listing 8: Shorthand notations representing the binary and gray-scale images.

true (of type bool) if both the colors are either white or black; otherwise, it returns false.

Given one color, the function negcolor (lines 8–12) returns the other color. It uses pattern matching (match ... with) that looks for the two patterns, black and white, of the color. The function negpix just inverts the color of the given pixel while keeping the row and column unchanged. Using negpix, the function negimage inverts the color of each pixel in the image. The next function eqpixel defines equality of two pixels. Two pixels are equal if their rows, columns, and colors are the same. This function is used in the function negpiximg, which inverts a specific pixel in an image.

4.3. Thresholding. In the binary image processing, thresholding is a technique used for converting a gray-scale image to a binary image. It is used to separate subimages that represent objects from the background. If a camera is designed to give gray-scale images, then thresholding is used to create binary image from the gray-scale image. A binary image B is retrieved by applying a threshold transformation T to a gray-scale image G. Informally, a fixed type of thresholding operation is defined (in equations (1) and (2)) below.

$$B[i, j] = G_T[i, j], \tag{1}$$

where

$$G_T[i, j] = \begin{cases} 1, & \text{if } G[i, j] \leq T, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

This kind of thresholding operation over a gray-scale image is defined in the Coq as a recursive function thresholding as shown in Listing 11. It takes an image and a fixed threshold value (of type nat) and returns threshold binary image. This function replaces the pixel gray-scale color value with 1 (black) if the gray-scale value of the pixel is less than or equal to the threshold provided; otherwise, it turns the color of the pixel to 0 (white).

In some applications, if the intensity values of the object of interest are in a range, then a range of thresholding values T_1 and T_2 is used. In this thresholding technique, the gray-scale pixel is turned black (represented with 1) if the gray-scale value of a pixel is in the range of T_1 and T_2. In other

```
Definition eqbcol (c1 c2: color): bool :=
match c1, c2 with
  |white, white ⇒ true
  |black, black ⇒ true
  |_, _ ⇒ false
end.
Definition negcolor (c: color): color:=
match c with
  |white ⇒ black
  |black ⇒ white
end.
Definition negpix (p: pixel): pixel:=
match p with
  |Br,c,col ⇒ Br,c, negcolor col
end.
Fixpoint negimage (pic: image): image:=
match pic with
  |nil ⇒ nil
  |cons p tl ⇒ cons (negpix p) (negimage tl)
end.
Fixpoint eqpixel (p1 p2: pixel): bool :=
match p1, p2 with
  |Br,c,col, Br′,c′,col′ ⇒
    andb (andb (r = ? r′) (c = ? c′)) (eqbcol col col′)
end.
Fixpoint negpiximg (p: pixel) (img: image): image:=
match img with
  |nil ⇒ nil
  |cons p′ ⇒ if eqpixel p p′ then
    cons (negpix p′) (negpiximg p tl)
    else cons p′ (negpiximg p tl)
end.
```

Listing 9: Definitions of operations over pixels and images.

```
Compute (thresholding gsimage 170).
Compute gsimagecoq.
Definition binaryimage := (thresholdrange gsimage 100 200).
Compute (areasize binaryimage).
Compute (runlength binaryimage).
Compute (sumrunlen (runlength binaryimage)).
```

Listing 10: Operations over the image.

```
Fixpoint thresholding (img: gsimage) (thresh: nat): image :=
match img with
|nil ⇒ nil
|cons Gr,c,gsv tl ⇒
  if (gsv < = ? thresh)
  then cons Br,c,black (thresholding tl thresh)
  else cons Br,c,white (thresholding tl thresh)
end.
```

Listing 11: Definition of fixed thresholding.

```
Fixpoint thresholdrange (img: gsimage) (T1 T2: nat): image :=
match img with
|nil ⇒ nil
|Gr,c,gsv:tl ⇒
   if andb (T1 <=? gsv) (gsv <=? T2)
   then Br,c,black::(thresholdrange tl T1 T2).
   else Br,c,white::(thresholdrange tl T2 T2)
end.
```

Listing 12: Definition of range thresholding.

words, the binary color of the pixel is black if the gray-scale value of the corresponding pixel is less than or equal to T_2 and greater than or equal to T_1. Thresholding based on a range of threshold values is defined as follows.

$$G_T[i, j] = \begin{cases} 1, & \text{if } T_1 \leq G[i, j] \leq T_2, \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

Thresholding operation in equation (3) is defined as a function thresholdrange in Coq as shown in Listing 12. The function thresholdrange takes a gray-scale image and two threshold values and returns a threshold binary image.

4.4. Area Size. The area of all objects in a binary image is the sum of all 1's (black) intensity values. Informally, the area A of all objects in a binary image B is defined in the equation below.

$$A = \sum_{i=1}^{n} \sum_{j=1}^{m} B[i, j]. \quad (4)$$

In Coq, the area of objects is defined by the recursive function areasize in Listing 13. It takes an image and returns the total area of the objects as a natural number.

4.5. Run-Length Encoding. Run-length encoding is a compact representation of a binary image normally used for image transmission. In this encoding, the lengths of the runs of 1 (black) pixels in the image compactly represent the image. Run-length of an image may be calculated using two methods. In one method, the start position and length (repetitions) of 1's in a row are used. In the second one, the lengths of 1 and 0 runs are used; however, it must begin with 1 run. The later representation is more compact and is formalized in this section.

In the first step, run-length of the 0's (white pixels) is calculated using the recursive function runlenwhite (Listing 14). The function runlenwhite takes an image img and a list of values l and returns a list of values, where each value represents length of 0 runs. It adds the run-lengths to the provided list initialized to nil (second argument of runlenwhite). When the pixel at the head of the image is white (lines 5–6), it appends 1 at head if the list l is nil (line 5); otherwise, it increments the value at head (line 6). The run-length value h preceded with constructor S increments h by one. When the color of the pixel is black (lines 7–8), it just moves on to the next pixel with the same list l if 0 is at the head of list l (line 7); otherwise, it adds 0 at the head of the list and repeats the cycle (line 8). At the end, the order of the list of values, created for the binary image, is reversed (lines 2–3) using the function rev.

Similarly, in the second step, the run-length of the 1's (black pixels) is calculated using the recursive function runlenblack (Listing 15). It returns the list of values (of type nat), where each value represents the run-length of 1's. Next, we define a recursive function alternate (Listing 16) that appends two lists by alternatively taking one value from each list.

Finally, the run-length of a binary image is defined in the function runlength (Listing 17). This function calculates the run-length of the image with either 0 or 1 run at the head of the returned list depending on which run occurs first. Alternatively, the run-length of the image with 1 run first is defined in the function runlength′ in Listing 18.

To state interesting theorems, we need a summation function to add all the run-lengths of a binary image. This is defined in the recursive function sumrunlen in Listing 19. The function sumrunlen takes black or white run-lengths of an image (as a list of nat) and returns the sum of all runs.

5. Proof of Properties

After binary and gray-scale images are formally defined, we are now ready to state and prove different properties of the binary images and image transformations.

5.1. Involution. The involution property states that double negation of a binary image makes no change to the original image. The involution property is defined as a Lemma negimg_involution in Listing 20. This lemma is proved by induction on the argument img of type image. It additionally applies an axillary lemma negpix_involution (Listing 21) using the rewrite tactic. The negpix_involution lemma states that double negation does not change the pixel.

5.2. Run-Length and Area. Run-length encoding can be used in computing horizontal projection, the area of all the objects, and vertical and diagonal projections. In this section, we prove that the area of all the objects in an image is equal

```
Fixpoint areasize (img: image): nat:=
match img with
|nil ⇒ 0
|Br,c,col:tl ⇒ match col with
   |black ⇒ S (areasize tl)
   |white ⇒ areasize tl
   end
end.
```

Listing 13: Calculating object area.

```
Fixpoint runlenwhite (img: image) (l: list nat): list nat \coloneq
match img, l with
| nil, 0:tl″ ⇒ rev tl″
| nil, _ ⇒ rev l
| Br,c,white:tl′, nil ⇒ runlenwhite tl′ (1:nil)
| Br,c,white:tl′, h:tl″ ⇒ runlenwhite tl′ (S h:tl″)
| Br,c,black:tl′, 0:tl″ ⇒ runlenwhite tl′ l
| Br,c,black:tl′, _ ⇒ runlenwhite tl′ (0:l)
end.
```

Listing 14: Run-length of 0's.

```
Fixpoint runlenblack (img: image) (l: list nat): list nat:=
match img, l with
|nil, 0:tl″ ⇒ rev tl″
|nil, _ ⇒ rev l
|Br,c,black:tl′, nil ⇒ runlenblack tl′ (1:nil)
|Br,c,black:tl′, h:tl″ ⇒ runlenblack tl′ (S h:tl″)
|Br,c,white:tl′, 0:tl″ ⇒ runlenblack tl′ l
|Br,c,white:tl′, _⇒ runlenblack tl′ (0:l)
end.
```

Listing 15: Run-length of 1's.

```
Fixpoint alternate (l1 l2: list nat): list nat:=
match l1, l2 with
|nil, _ ⇒ l2
|_, nil ⇒ l1
|h1:tl1, h2:tl2 ⇒ h1:h2::(alternate tl1 tl2)
end.
```

Listing 16: Intermixing two lists.

to the sum of the lengths of all 1 runs. Formally, this is stated as a lemma runlenblack_eq_area_all in Listing 22.

The lemma runlenblack_eq_area_all states that the sum of the 1 (black) runs of a binary image img is equal to the area of all the objects in the image. This lemma is proved using induction on the argument img, followed by case analysis over the pixels p. In addition, the proof of this lemma includes the application of few other lemmas such as runlen_0_nil and sumrunlen_l. Complete proof script of these additional lemmas is not included here due to limited space and is available at our GitHub repository.

5.3. Distance Measures. Many applications [35, 36] require to measure the distance between any two pixels of a binary image. There are several methods to find the distance between any two pixels of the image. Among them are Euclidean, City-block, and Chessboard. The City-block

```
Definition runlength (img: image): list nat :=
match img with
|nil ⇒ nil
|Br,c,black:tl′ ⇒
   alternate (runlenblack img nil) (runlenwhite img nil)
|Br,c,white:tl′ ⇒
   alternate (runlenwhite img nil) (runlenblack img nil)
end.
```

Listing 17: Run-length of binary image.

```
Definition runlength′ (img: image): list nat :=
   alternate (runlenblack img nil) (runlenwhite img nil).
```

Listing 18: Run-length of binary image (1 run first).

```
Fixpoint sumrunlen (runl: list nat): nat :=
match runl with
| nil ⇒ 0
|r:tl ⇒ r + (sumrunlen tl)
end.
```

Listing 19: Adding the run-lengths of an image.

```
Lemma negimg_involution: forall img: image,
   negimage (negimage img) = img.
Proof.
   intros.
   induction img.
   + simpl. auto.
   + simpl. rewrite IHimg.
      rewrite negpix_involution.
      auto.
Qed.
```

Listing 20: Involution property of images.

```
Lemma negpix_involution: forall pix: pixel,
   negpix (negpix pix) = pix.
Proof.
   intro.
   destruct pix.
   simpl.
   induction col.
   + simpl. auto.
   + simpl. auto.
Qed.
```

Listing 21: Involution property of pixels.

```
Lemma runlenblack_eq_area_all: forall img:image
sumrunlen (runlenblack img nil) = areasize img.
Proof.
  intro.
  induction img as [ |p pl].
  + simpl. auto.
  + destruct p.
    destruct col.
    −simpl.
      rewrite < −IHpl.
      rewrite runlen_0_nil.
      auto.
    −simpl.
      rewrite < −IHpl.
      simpl.
      rewrite sumrunlen_l.
      simpl.
auto.
Qed.
```

Listing 22: Run-length and area.

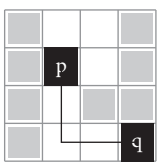


Figure 3: City-block distance measure.

distance measure between two pixels p and q is shown in Figure 3. The two pixels, p and q, are located at the positions (1,1) and (3,3), respectively. The City-block method for distance measure is informally defined as $D_{\text{city}} = |i_1 - i_2| + |j_1 - j_2|$, where (i_1, j_1) and (i_2, j_2) represent positions of the two pixels. The City-block distance between pixels p and q in the example above is $D_{\text{city\_pq}} = |1-3| + |1-3| = 2+2=4$.

Regardless of the distance measure method, all pixels of the image must satisfy the following properties. For any three pixels p, q, and r, we must prove the following:

(1) $d(p,q) \geq 0$ and $d(p,q) = 0$ iff $p = q$.

(2) $d(p,q) = d(q,p)$.

(3) $d(p,r) <= d(p,q) + d(q,r)$.

In this section, we formally define City-block method for computing distance between any two pixels and then state and prove the above three properties of City-block distance. To begin with, first, the operation $|m - n|$, for any m and n, is formally defined as a Coq definition modminus in Listing 23. The shorthand notation used for the function modminus is {m − −n}.

The City-block method of distance measure is defined in the function citydist in Listing 24. This function gets two pixels as argument and returns the distance between them. After defining the City-block distance measure method, we are now ready to reason about it. In the lemma distcb_min in the Listing 25, we state and prove one of the two parts of the first property. The first part of this property is $d(p,q) \geq 0$, where p and q are any two pixels of the image. This lemma is proved using case analysis on the arguments p and q.

The second part of the first property of the City-block distance measure is $d(p,q) = 0$ iff $p = q$. This property is stated as a lemma distcb_eq_0 and proved in Listing 26. The proof of this lemma proceeds using case analysis of arguments p and q using tactic destruct and then applying other Coq tactics such as rewrite, simpl, inversion, and lia.

The reverse ∀ p q, citydist p q = 0->eqpixel p q = true. of lemma distcb_eq_0 (Listing 26) is not true. The reason is that the definition of citydist does not take the color of the pixels into consideration. So even if citydist p q = 0 holds, it does not mean their colors are also equal; hence, the equality of pixels cannot be proved.

The second property states that the distance measure is commutative. That is, for any two pixels p and q, $d(p,q) = d(q,p)$. This is formalized in Coq as a lemma distcb_comm in Listing 27. Similarly, the third property, $d(p,r) <= d(p,q) + d(q,r)$, is defined as a lemma citydist_trans in Listing 28 (due to space limitations, complete proofs of these two lemmas are not listed here and can be accessed from our repository).

6. Experimental Analyses

The binary image in Figure 2 is a small 16-pixel grayscale image. To test that our formal definitions can be used for the real life images, we performed experiments by running the algorithms over RGB images. For this purpose, we designed and developed a prototype tool in C++ programming language. When run, the tool asks to enter an RGB image name and a threshold value for binary conversion. It then generates Coq proof script for the image (file proofofgsi.v), binary image matrix (text file "binaryimage.txt"), and Coq

```
Definition modminus (n m: nat): nat \coloneq
if n < ? m then m – n else n – m.
Notation "m – n" \coloneq (modminus m n).
```

Listing 23: Definition and notation of modminus operation.

```
Definition citydist (p1 p2: pixel): nat \coloneq
match p1, p2 with
| Br1,c1,_, Br2,c2,_ =>  r1–r2 + c1–c2
end.
```

Listing 24: Calculating distance using the city-block method.

```
Lemma distcb_min: forall p q, citydist p q >= 0.
Proof.
intros.
destruct p as [r1 c1 col1].
destruct q as [r2 c2 col2].
simpl.
lia.
Qed.
```

Listing 25: Proof of lemma distcb_min.

```
Lemma distcb_eq_0: for all p q, eqpixel p q = true->citydist p q = 0.
Proof.
intros.
destruct p as [r1 c1 col1].
destruct q as [r2 c2 col2].
simpl.
unfold eqpixel in *.
do 2 rewrite andb_true_iff in H.
inversion H.
inversion H0.
assert (r1 = r2) as Hreq.
apply beq_nat_true; auto.
assert (c1 = c2) as Hceq.
apply beq_nat_true; auto.
rewrite Hreq. rewrite Hceq.
unfold modminus.
destruct ltb.
destruct ltb.
lia.
lia.
destruct ltb.
lia.
lia.
Qed.
```

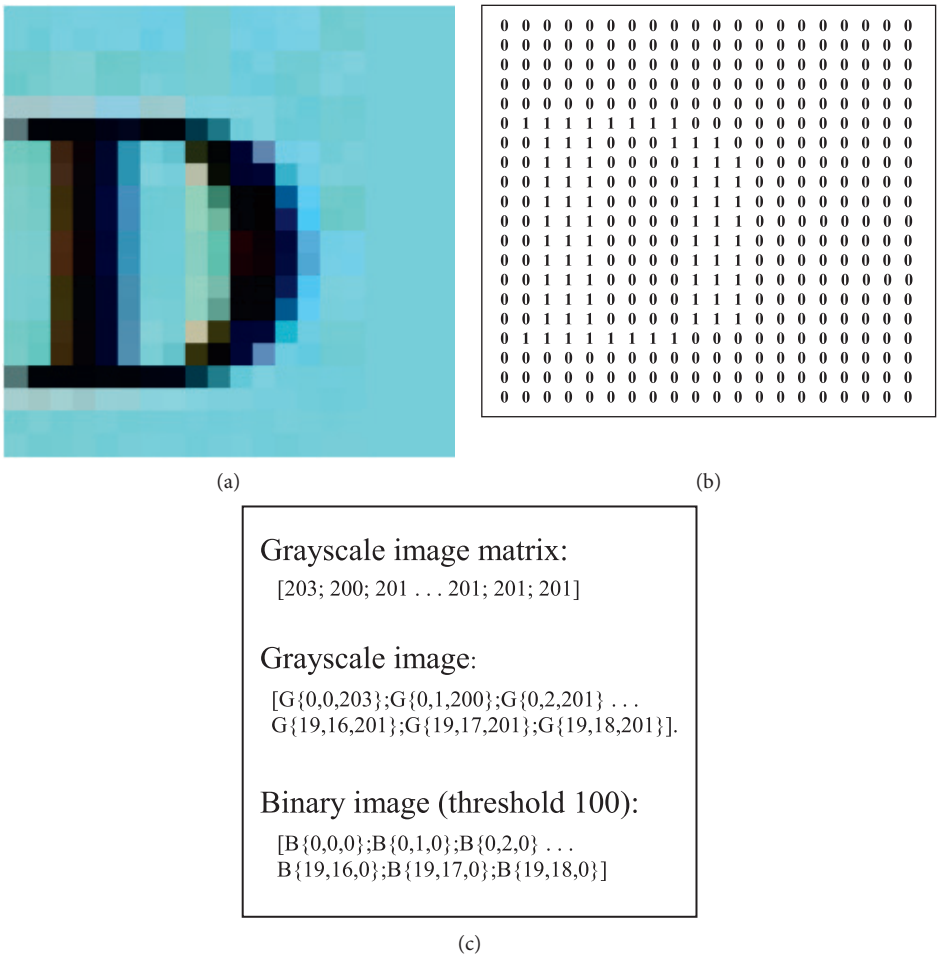
Listing 26: Proof of lemma distcb_eq_0.

```
Lemma distcb_comm: forall p q, citydist p q = citydist q p.
```

Listing 27: Proof of lemma distcb_comm.

```
Lemma citydist_trans: forall p q r
citydist p r <= citydist p q + citydist q r.
```

Listing 28: Proof of lemma citydist_trans.



(a)

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

(b)

Grayscale image matrix:

[203; 200; 201 . . . 201; 201; 201]

Grayscale image:

[G{0,0,203};G{0,1,200};G{0,2,201} . . .
G{19,16,201};G{19,17,201};G{19,18,201}].

Binary image (threshold 100):

[B{0,0,0};B{0,1,0};B{0,2,0} . . .
B{19,16,0};B{19,17,0};B{19,18,0}]

(c)

Figure 4: Example 1: translating the RGB image to the matrix and Coq definition gsimage. (a) RGB image. (b) Binary image matrix. (c) Coq definitions of the image.

definition of the binary image (text file "coqbinaryimage.txt"). The Coq script file includes the following:

(1) Definitions of the grayscale image matrix, grayscale image and binary image (all are tool generated)

(2) Definitions of the grayscale and binary images (Coq generated)

(3) Proof of equivalence of the two versions of the grayscale and binary images

(4) Other operations over the image

When the RGB image is read, it is first translated to an RGB matrix where every group of three consecutive gray values corresponds to the three colors red, green, and blue of the pixel. The gray value of a pixel in the grayscale matrix is the average of the red, green, and blue colors' values of the corresponding pixel in the RGB image. The grayscale image matrix is converted to a formal definition of the grayscale image, both by the tool and by the Coq

(a)

1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	1	0	1	0
1	1	1	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b)

Grayscale image matrix:

[97;93;84; . . . 136;137;138]

Grayscale image:

[G{0,0,97};G{0,1,93};G{0,2,84} . . .
;G{19,16,136};G{19,17,137};G{19,18,138}]

Binary image (threshold 120):

B{0,0,1};B{0,1,1};B{0,2,1} . . .
B{19,16,0};B{19,17,0};B{19,18,0}]

(c)

Figure 5: Example 2: translating the RGB image to matrix and Coq definition gsimage. (a) RGB image. (b) Binary image matrix. (c) Coq definitions of the image.

```
Fixpoint createimage (matrix: list nat) (row: nat) (col: nat) (maxcol: nat): gsimage \coloneq
match matrix with
  |nil ⇒ nil
  |cons gray m ⇒
    match row, col with
    |O, O ⇒ [Grow, col, gray]
    |S r, O ⇒  createimage m r maxcol maxcol) ++ [Grow, col, gray]
    |O, S c ⇒ (createimage m row c maxcol) ++ [Grow, col, gray]
    |S r, S c ⇒ (createimage m row c maxcol) ++ [Grow, col, gray]
    end
end.
```

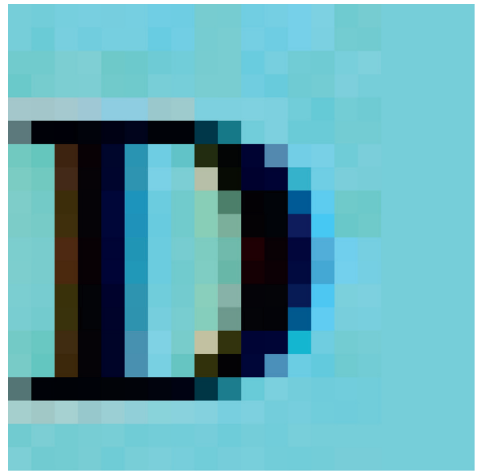
Listing 29: Definition of function createimage.

```
Theorem gsimage_eq: coqgsimage = gsimagecoq.
Proof.
  unfold coqgsimage.
  unfold gsimagecoq.
  simpl.
  reflexivity.
Qed.
```

Listing 30: Proof of equivalence of grayscale images.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

(a)



(b)

Figure 6: Binary image matrix comparison with the corresponding RGB image (Example 1). (a) Binary image matrix. (b) RGB image.

```
1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1 0
1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

(a)

(b)

Figure 7: Binary image matrix comparison with the corresponding RGB image (Example 2). (a) Binary image matrix. (b) RGB image.

function createimage. In either case, the grayscale image conforms to the Coq formal definition gsimage (Listing 7). To verify that this translation is sound, we defined a function createimage (Listing 29) to generate a grayscale image from the matrix. The function createimage gets a matrix (list of gray values) and its dimensions (number of rows and columns) and generates a grayscale image of the form given in Figures 4 and 5. In the theorem gsimage_eq (Listing 30), we prove that the grayscale image coqgsimage generated by the tool is exactly the same as the image gsimagecoq generated by our formal function createimage.

The grayscale matrix is then converted to the formal definition of a binary image using a threshold value or a pair (range) of threshold values. Again, this translation is carried out by both the tool and the Coq function. The formal definition of the binary image (Listing 4) is a list of pixels where each pixel is a tuple of row, column, and binary value. The tuple, defined as a record type in Coq, has member rows, cols, and pixels as a list of nat. A recursive function imagetorealimage is defined to generate the tuple from the Coq binary image. To show that the translation to binary image is sound, a proof (Lemma binimage_eq) is automatically generated. This lemma states that the tool generated binary image coqbinaryimage and the binary image binimagecoq generated by the formal definition thresholding are the same.

In addition, the file proofofgsi.v has some fixed code for different operations over the image chosen for the analysis. These operations include thresholding (to generate binary image from the grayscale image), computing area size and run-length, and summing the run-lengths of the binary image. Currently, the images chosen are small (20×20 pixels). Larger images can also be translated, but the resulting binary images in the Coq would take longer (in order of minutes) to compile. The tool, though, can be easily

modified to split the image in slices and then perform the analysis for each slice separately.

We used our tool and converted multiple 20×20 RGB images to Coq script. Two 20×20 (400 pixels) translated RGB images with their Coq definitions of the matrix, grayscale, and binary forms are given in Figures 4 and 5 (the image matrix, Coq grayscale, and binary images are too long to be included here and are available at our repository). These grayscale images were created by the tool from their corresponding image matrices and then translated to Coq notations.

As mentioned earlier, the tool can generate binary image from the RGB image. The binary image matrix of the RGB image 4 is shown in Figure 6 (the RGB image from Figure 4 is resketched). Similarly, the binary image matrix of the RGB image 5 is shown in Figure 7. These binary images are retrieved using the threshold values 100 and 120, respectively. For each pixel of the binary image, the value, whether 0 or 1, is calculated by comparing the corresponding pixel's gray value with the threshold. The binary pixel is 0 if the pixel gray value is greater than the threshold; otherwise, it is 1. Note that the binary image matrix would change for different values of the threshold.

7. Conclusion

Binary images include pixels of just two gray values, black and white, and require only one bit per pixel storage. Binary image processing involves different transformations and encoding in a wide range of applications, including digital forensics and image integrity. To investigate binary image processing using formal techniques, a lightweight formal model of the binary images and multiple image transformations were defined in the Coq proof assistant. Using the interactive proof facility of Coq, several properties of the transformations were stated and proved. The formal system was tested against real RGB images though a prototype translator. The binary image defined as Coq notations is converted back to binary matrix.

Currently, the formal development can be tested against light (20×20 pixels) RGB images; however, high-resolution images would take longer (in order of minutes) to check. In the future, this issue may be resolved by splitting the high-resolution RGB images into slices. Each slice of the RGB image may be treated as a lightweight RGB image and processed separately. Furthermore, we formally specified and verified a number of interesting properties of the binary images and image transformations, but many are still missing. For example, the current formal system could be extended in the future to include proof of the distance properties of Euclidean and Chessboard distance functions.

Abbreviations

RGB: Red, green, and blue
HOL: Higher-order logic
CNN: Convolutional neural network
HDL: Hardware description language.

Data Availability

This research paper includes Coq formal developments and C++ code. All the source codes are available online at our repository at https://github.com/wilstef/binaryimagescoq.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Authors' Contributions

W.K. and S.I. carried out the formal specification and verification in the Proof Assistant Coq and wrote the first draft of the paper. W.K., A.R.A., and A.Q. contributed to the design and development of the C++ code and carried out the experiments. A.A., A.H., and S.A. reviewed the paper, contributed to the formal proof checking, and arranged the funding for this research.

Acknowledgments

This research was funded by the Deanship of Scientific Research at Prince Sattam Bin Abdulaziz University, Al-Kharj, Saudi Arabia.

References

[1] R. Chauhan, P. Mishra, and R. C. Joshi, "An overview of digital image forensics: image morphing and forgery detection algorithms," in *Information Security and Optimization*, pp. 107–120, Chapman and Hall/CRC, London, UK, 2020.

[2] S. Lai and R. Böhme, "Countering counter-forensics: the case of JPEG compression," in *International Workshop on Information Hiding*, pp. 285–298, Springer, 2011.

[3] P. S. Othman, R. R. Ihsan, R. B. Marqas, and S. M. Almufti, "Image processing techniques for identifying impostor documents through digital forensic examination," *Image Process. Tech*, vol. 62, pp. 1781–1794, 2020.

[4] G. Horsman and J. R. Lyle, "Dataset construction challenges for digital forensics," *Forensic Science International: Digital Investigation*, vol. 38, Article ID 301264, 2021.

[5] A. Shaus, Y. Gerber, S. Faigenbaum-Golovin, B. Sober, E. Piasetzky, and I. Finkelstein, "Forensic document examination and algorithmic handwriting analysis of Judahite biblical period inscriptions reveal significant literacy level," *PLoS One*, vol. 15, no. 9, Article ID e0237962, 2020.

[6] S. Agarwal, D. Sharma, and K. H. Jung, "Forensic analysis of colorized grayscale images using local binary pattern," in *Proceedings of the 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 507–510, Noida, India, 2020.

[7] J. A. Redi, W. Taktak, and J.-L. Dugelay, "Digital image forensics: a booklet for beginners," *Multimedia Tools and Applications*, vol. 51, no. 1, pp. 133–162, 2011.

[8] R. Böhme and M. Kirchner, "Counter-forensics: attacking image forensics," in *Digital Image Forensics*, pp. 327–366, Springer, Berlin, Germany, 2013.

[9] Q. Duan, T. Akram, P. Duan, and X. Wang, "Visual saliency detection using information contents weighting," *Optik*, vol. 127, no. 19, pp. 7418–7430, 2016.

[10] I. D. Hernández, J. V. Hernández-Fontes, M. A. Vitola, M. C. Silva, and P. T. Esperança, "Water elevation measurements using binary image analysis for 2D hydrodynamic experiments," *Ocean Engineering*, vol. 157, pp. 325–338, 2018.

[11] Z. Zhao, C. Yao, C. Tang, C. Li, F. Yan, and S. Islam, "Diagnosing transformer winding deformation faults based on the analysis of binary image obtained from FRA signature," *IEEE Access*, vol. 7, pp. 40463–40474, 2019.

[12] H. Kim, J. Lee, E. Ahn, S. Cho, M. Shin, and S.-H. Sim, "Concrete crack identification using a UAV incorporating hybrid image processing," *Sensors*, vol. 17, no. 9, Article ID 2052, 2017.

[13] P. Narendran and J. Stillman, "Formal verification of the Sobel image processing chip," in *Current Trends in Hardware Verification and Automated Theorem Proving*, pp. 92–127, Springer, Berlin, Germany, 1989.

[14] N. G. Bourbakis and C. Alexopoulos, "A fractal-based image processing language: formal modeling," *Pattern Recognition*, vol. 32, no. 2, pp. 317–338, 1999.

[15] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks, Automated Technology for Verification and Analysis," in *Proceedings of the International Symposium on Automated Technology for Verification and Analysis*, pp. 269–286, Pune, India, 2017.

[16] P. Kouvaros and A. Lomuscio, "Formal verification of CNN-based perception systems," 2018, https://arxiv.org/abs/1811.11373.

[17] X. Sun, H. Khedr, and Y. Shoukry, "Formal verification of neural network controlled autonomous systems," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pp. 147–156, New York, NY, USA, 2019.

[18] P. Meredith, M. Katelman, J. Meseguer, and G. Roşu, "A formal executable semantics of Verilog," in *Proceedings of the Eighth ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE 2010)*, pp. 179–188, Grenoble, France, 2010.

[19] M. Clavel, F. Durán, J. Hendrix, S. Lucas, J. Meseguer, and P. Ölveczky, "The Maude formal tool environment," in *Proceedings of the International Conference on Algebra and Coalgebra in Computer Science*, pp. 173–178, Bergen, Norway, 2007.

[20] W. Khan, D. Sanan, Z. Hou, and L. Yang, "On embedding a hardware description language in Isabelle/HOL," *Design Automation for Embedded Systems*, vol. 23, no. 3-4, pp. 123–151, 2019.

[21] W. Khan, A. Tiu, and D. V.F. Sanán, "An executable formal model of a hardware description language," in *Proceedings of the 2nd Singapore Cyber-Security R&D Conference*, pp. 19–36, Singapore, February 2017.

[22] W. Khan, M. Kamran, A. Ahmad, F. A. Khan, and A. Derhab, "Formal analysis of language-based android security using theorem proving approach," *IEEE Access*, vol. 7, pp. 16550–16560, 2019.

[23] T. Braibant and A. Chlipala, "Formal verification of hardware synthesis," in *Proceedings of the International Conference on Computer Aided Verification*, pp. 213–228, Saint Petersburg, Russia, 2013.

[24] J. Choi, M. Vijayaraghavan, B. Sherman, and A. Chlipala, "Kami: a platform for high-level parametric hardware specification and its modular verification," in *Proceedings of the ACM on Programming Languages*, pp. 1–30, New York, NY, USA, 2017.

[25] M. Bugliesi, S. Calzavara, R. Focardi, W. Khan, and M. Tempesta, "Provably sound browser-based enforcement of web session integrity," in *Proceedings of the 2014 IEEE 27th Computer Security Foundations Symposium*, pp. 366–380, Vienna, Austria, 2014.

[26] W. Khan, S. Calzavara, M. Bugliesi, W. De Groef, and F. Piessens, "Client side web session integrity as a non-interference property," in *Proceedings of the International Conference on Information Systems Security*, pp. 89–108, Hyderabad, India, 2014.

[27] M. Bugliesi, S. Calzavara, R. Focardi, and W. Khan, "Automatic and robust client-side protection for cookie-based sessions," in *Proceedings of the International Symposium on Engineering Secure Software and Systems*, pp. 161–178, Munich, Germany, 2014.

[28] M. Bugliesi, S. Calzavara, R. Focardi, and W. Khan, "CookiExt: patching the browser against session hijacking attacks," *Journal of Computer Security*, vol. 23, no. 4, pp. 509–537, 2015.

[29] M. A. Alturki, J. Chen, V. Luchangco et al., "Towards a verified model of the Algorand consensus protocol in Coq," 2019, https://arxiv.org/abs/1907.05523.

[30] A. Anand, A. Appel, G. Morrisett et al., "CertiCoq: a verified compiler for Coq," in *Proceedings of theThird International Workshop on Coq for Programming Languages (CoqPL)*, Paris, France, 2017.

[31] X. Leroy, S. Blazy, D. Kästner, B. Schommer, M. Pister, and C. Ferdinand, "CompCert-a Formally Verified Optimizing Compiler," in *Proceedings of the 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, Toulouse, France, 2016.

[32] A. Chlipala, *Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant*, MIT Press, Cambridge, MA, USA, 2013.

[33] B. C. Pierce, C. Casinghino, M. Gaboardi et al., "Software foundations," 2010, https://www.cis.upenn.edu/bcpierce/sf/current/index.html.

[34] T. Nipkow, L. C. Paulson, and M. Wenzel, *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, Springer Science & Business Media, Berlin, Germany, 2002.

[35] Y. D. Mistry, "Textural and color descriptor fusion for efficient content-based image retrieval algorithm," *Iran Journal of Computer Science*, vol. 3, no. 3, pp. 169–183, 2020.

[36] J. Wang and Y. Tan, "Efficient Euclidean distance transform algorithm of binary images in arbitrary dimensions," *Pattern Recognition*, vol. 46, no. 1, pp. 230–242, 2013.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 2109024, 13 pages
https://doi.org/10.1155/2022/2109024

Research Article

Profile Aware ObScure Logging (PaOSLo): A Web Search Privacy-Preserving Protocol to Mitigate Digital Traces

Mohib Ullah,[1] **Rafi Ullah Khan**,[1] **Irfan Ullah Khan**,[2] **Nida Aslam**,[3] **Sumayh S. Aljameel**,[2] **Muhammad Inam Ul Haq**,[4] **and Muhammad Arshad Islam**[5]

[1]*Institute of Computer Science and Information Technology, The University of Agriculture, Peshawar, Pakistan*
[2]*Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia*
[3]*SAUDI ARAMCO Cybersecurity Chair, Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia*
[4]*Department of Computer Science and Bioinformatics, Khushal Khan Khattak University Karak, Pakistan*
[5]*National University of Computer and Emerging Sciences, Islamabad, Pakistan*

Correspondence should be addressed to Mohib Ullah; mrmohibkhan@gmail.com

Received 27 August 2021; Accepted 23 November 2021; Published 3 February 2022

Academic Editor: Farhan Ullah

Web search querying is an inevitable activity of any Internet user. The web search engine (WSE) is the easiest way to search and retrieve data from the Internet. The WSE stores the user's search queries to retrieve the personalized search result in a form of query log. A user often leaves digital traces and sensitive information in the query log. WSE is known to sell the query log to a third party to generate revenue. However, the release of the query log can compromise the security and privacy of a user. In this work, we propose a Profile Aware ObScure Logging (PaOSLo) Web search privacy-preserving protocol that mitigates the digital traces a user leaves in Web searching. PaOSLo systematically groups users based on profile similarity. The primary objective of this work is to evaluate the impact of the systematic group compared to random grouping. We first computed the similarity between the users' profiles and then clustered them using the K-mean algorithm to group the users systematically. Unlikability and indistinguishability are the two dimensions in which we have measured the privacy of a user. To compute the impact of systematic grouping on a user's privacy, we have experimented with and compared the performance of PaOSLo with modern distributed protocols like OSLo and UUP(e). Results show that, at the top degree of the ODP hierarchy, PaOSLo preserved 10% and 3% better profile privacy than the modern distributed protocols mentioned above. In addition, the PaOSLo has less profile exposure for any group size and at each degree of the ODP hierarchy.

1. Introduction

Web search engines (WSEs) like Google, Ask, Bing, AOL, Baidu, and others provide the easiest way to search and retrieve information from the Web. The WSE stores the users' submitted queries in a query log. The WSE regularly builds and updates a user profile from the query log to provide personalized results [1]. The WSE generates revenue by analyzing the query log coupled with a user profile to provide relevant advertisements [2, 3]. Research shows that 35% of products people buy on Amazon and 75% of videos they watch on Netflix result from personalized recommendations [4]. The user query log often contains sensitive information, and the release of such information poses a risk to user's security and privacy [2, 5]. In today's Internet life, preserving web search privacy is the real perturb of a user. Existing techniques hide the identity through unlinkability and obfuscate the profile through indistinguishability to succeed the Web search privacy of a person [2, 6]. Internet users often use proxy services (like scoogle.com, anonymizer.com, and others) and TOR (the onion routing) network to attain unlinkability [7], whereas users utilise TrackMeNot [7], GooPIR, and DisPA [8] to achieve indistinguishability by sending fictitious but real

queries to obfuscate the profile maintained by the WSE [2]. However, the WSE can recognise TOR users' queries from the cookies and application layer. Similarly, the unlinkability achieved through proxy services can be more precarious to users as the privacy policies of the proxy servers are not as regulated as the WSE.

The distributed privacy-preserving protocol is another approach that provides both unlinkability and indistinguishability. It works on the cooperation of multiple users. The distributed protocols create a group of "*n*" users who need to query the WSE secretly. Group users send each other queries to WSE and broadcast the results received in the group. A user achieves unlinkability in distributed protocols as a user's query is forwarded by another group user. Similarly, a user attains indistinguishability by forwarding queries of the other group of users. Such measures obscure the users' profiles with group member queries. Hence, the user's profile kept at WSE includes queries of other group users.

The grouping of users is considered a primary step in distributed protocols. In the existing approaches, a Core Server (CS) accepts a connection request from individuals who want to perform a Web search secretly. Upon receiving "*n*" number of connection requests, the CS creates a group of users on a first-come first-serve basis, also called random grouping. The major shortcoming of a random grouping is that a user may be grouped with those users who have similar interests, as there is no prior information about the users' preferences [9]. Consider a situation where a user Mr. X has a medical-related query. He is in a group with other users having a similar interest; in such a case, although Mr. X forwards a query of another user, his profile will not be significantly obfuscated. The profile maintained by the WSE for Mr. X will contain the same type of categories. Profile obfuscation is the fundamental objective of web search privacy; however, a user can be grouped with similar interests with an existing randomized grouping. Such grouping will not expressively obfuscate the profiles of the users. Therefore, a mechanism is required to systematically group users based on their interests.

This work proposes a novel distributed privacy-preserving protocol, PaOSLo (Profile Aware ObScure Logging), to significantly obfuscate a user's profile by systematically grouping users instead of random grouping. The PaOSLo executes in two steps: (i) cluster the user according to their profile similarity. (ii) The CS creates a group of users from each distinct cluster instead of random grouping. PaOSLo aims to achieve the following objectives:

(1) To notably obfuscate the profile of a user by creating a profile-aware grouping mechanism

(2) An experiment is performed to estimate the extent of profile obfuscation by utilizing profile-aware grouping versus randomized grouping

A K-mean algorithm is employed to set users into three clusters, four clusters, and five clusters. The similarity between the users' profiles is computed using the cosine similarity metric.

The rest of this article is represented as existing work which is discussed in Section 2. PaOSLo and its execution are explained in Section 3. Section 4 describes the dataset and simulation details. Section 5 presents PaOSLo privacy evaluation. Section 6 details results and discussion. The last section of the article demonstrates the conclusion and future work.

2. Existing Work

As discussed above, unlinkability and indistinguishability are the advantages of distributed protocols compared to other privacy-preserving schemes. Crowds was the first distributed protocol proposed to protect web search privacy [10]. However, it was vulnerable to an active and passive adversary [11]. User private information retrieval (UPIR) presented by Domingo-Ferrer et al. used memory spaces as drop boxes to achieve indistinguishability [12]. However, the query remained visible to the users associated with the same memory location, and UPIR was vulnerable to intersection attacks. Swanson and Stinson extended the model of UPIR and added standard terminologies in the field of privacy [13]. They calculated the probabilistic advantage to a user connected with the same dropbox in linking a query with the originator. Swanson and Stinson again extended the UPIR, and they figured the privacy relative to peer users when they make a coalition to link a query with the originator [14]. However, this extended technique suffered the same shortcomings. First, the search query remained visible to group users; second, group user collaboration could reveal the user's identity. Castella Roca et al. proposed a Useless User Profile (UUP) protocol. It employed a central server (CS) to create a group of "*n*" users; the queries were shuffled among the users before forwarding to the WSE [15]. UUP results were broadcast in clear text, letting everyone know the intention of other group users. Romero-Tris et al. integrated the ElGamal key encipherment to attain concealment and the optimized Benes network to shuffle the queries in UUP to achieve privacy in the existence of an untrusted partner [16]. However, the extended UUP (UUP (e)) was still unprotected as the researchers compromised a user's privacy through machine learning attacks [5]. Ullah et al. presented ObScure logging (OSLo) to lessen the deficiencies of previous techniques and preserve a user's privacy. The central server (CS) in OSLo exercised a single dynamic group of "*n*" random users [11]. OSLo encrypted both the query and results to achieve confidentiality and shuffled the queries with the flip of a coin to attain unlinkability. However, there was no system for group creation; any user can be grouped with another user. Due to random grouping, the chances of being grouped with users having the same interests were higher. Domingo-Ferrer et al., to support social welfare, introduced a novel concept of a self-enforcing protocol called coutile [17]. To send a query, a user must check the query if the query would obfuscate the user's profile or expose the profile. In the former case, the user sends the search keyword directly to WSE, while in the latter case, the user asks the peer member to deliver the query on their behalf. When a user asks a peer for query forwarding, he/she follows the same steps. The peer only forwards if the query obscures the profile and denies the

request otherwise. Delay is a primary issue in coutile protocol, a user waits for a longer time to get a query answered, but the user's request gets denied for not obfuscating the profile of a peer user. Such a situation can cause a significant delay in getting results for the query. Authors have presented MG-OSLo to obfuscate the profile of users by employing multiple groups [2]. MG-OSLo achieved both local privacy and profile privacy through unlinkability and indistinguishability. However, the same random grouping is used in MG-OSLo, and the chance of being grouped with users having similar interests remains the same. Kaaniche et al. proposed a decentralized solution CoWSA that empowers end-users to have control over personal data, mitigates single-point failure, ensures the security of the queries, and provides anonymity to a user [1]. User, client, WSE, third parties (TP), and trusted authorities are the five entities of CoWSA. It is a proxy solution to retrieve data from the WSE based on the Sys_Init, Query_Submit, and Query_Resp procedures. The Sys_Init procedure involves interest-based group creation. Query_submit corresponds to the process of sending queries to the WSE by setting a random path through multiple relay users. The Query_Resp occurs when the WSE receives the query, aggregates the profile of the user, and returns the answer to the user through TPs. However, the CoWSA does not explain how aggregated profiles are computed and what level of obfuscation a client achieves.

3. Profile Aware ObScure Logging (PaOSLo) Protocol

PaOSLo creates a single dynamic group of users with diverse interests to obscure a person's profile significantly. The PaOSLo consists of entities like users, a central server (CS), a query forwarding node (QFN), and a web search engine (WSE). A user is an individual who wishes to perform a web search secretly. A CS is a dedicated machine that oversees the PaOSLo group creation and execution process. QFN is one of the group users selected by the CS for sending queries of group users to WSE, retrieving and broadcasting the query results to group members.

3.1. PaOSLo Execution Process. The following are the steps necessary in the accomplishment of PaOSLo:

(i) The similarity between the users' profiles is computed in the first step of the PaOSLo execution. The user's profile construction is detailed in Section 3.

(ii) In the second step, the users are clustered using the K-mean algorithm based on their profile similarity.

(iii) In the following step, PaOSLo creates a group of users by selecting one from each cluster. This group will have users of different interests.

Figure 1 shows the activity diagram of computing the similarities between the users' profiles and grouping them into k clusters using the K-mean algorithm [8]. A user's profile is compared with the profiles of all other users using the cosine similarity metric [18]. An $n \times n$ matrix is obtained

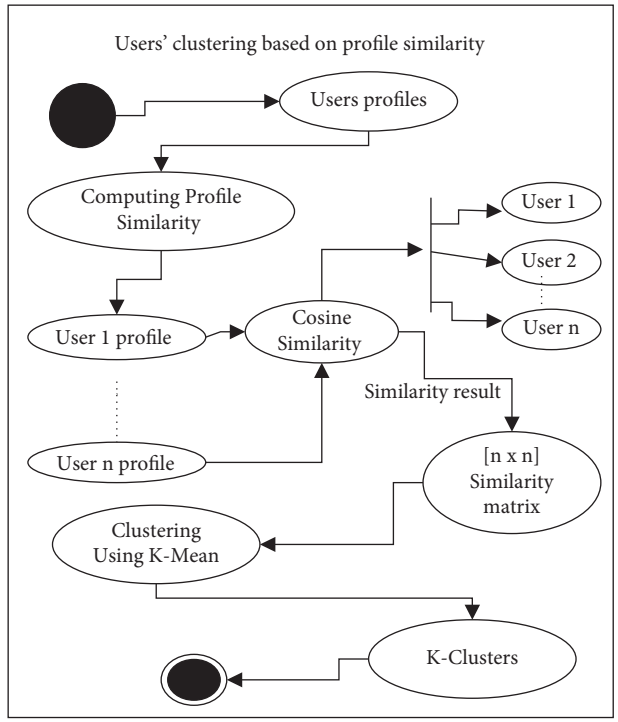


Figure 1: Activity diagram of computing similarity and users' clustering.

representing the similarities between the users' profiles. The matrix is uploaded into the Weka tool for clustering users using the K-mean algorithm.

3.2. Group Making and Query Forwarding Node Selection Process. In this work, three group sizes have been chosen for experimentation, i.e., three users, four users, and five users. Figure 2 shows the activity diagram of group making and the QFN designation process. After clustering, one user from each cluster is selected by CS to create a group. The CS maintains a user_list[], containing the IP address and port number of the users selected from each cluster. Once the required group size is reached, the CS selects one user from the group as QFN. The CS forwards a "get_QFN_info" message to the user specified as QFN. When the user receives a get_QFN_info message, it generates a public-private key pair and selects a port number for communication. The QFN generates a detail_QFN message containing information about the encryption keys and port number for communication and forwards a detail_QFN message to the CS. The CS then broadcasts a user_list[] and detail_QFN in the created group. The role of QFN is to send queries of other group peers to the WSE, collect the queries' results from WSE, and broadcast them back to the group. The CS selects each user of the group as QFN in round-robin fashion. When all users play the role of QFN, the CS concludes the group. If there are "n" users in the group, the QFN forwards "$n-1$" queries to the WSE; the QFN never forwards their queries to the WSE.

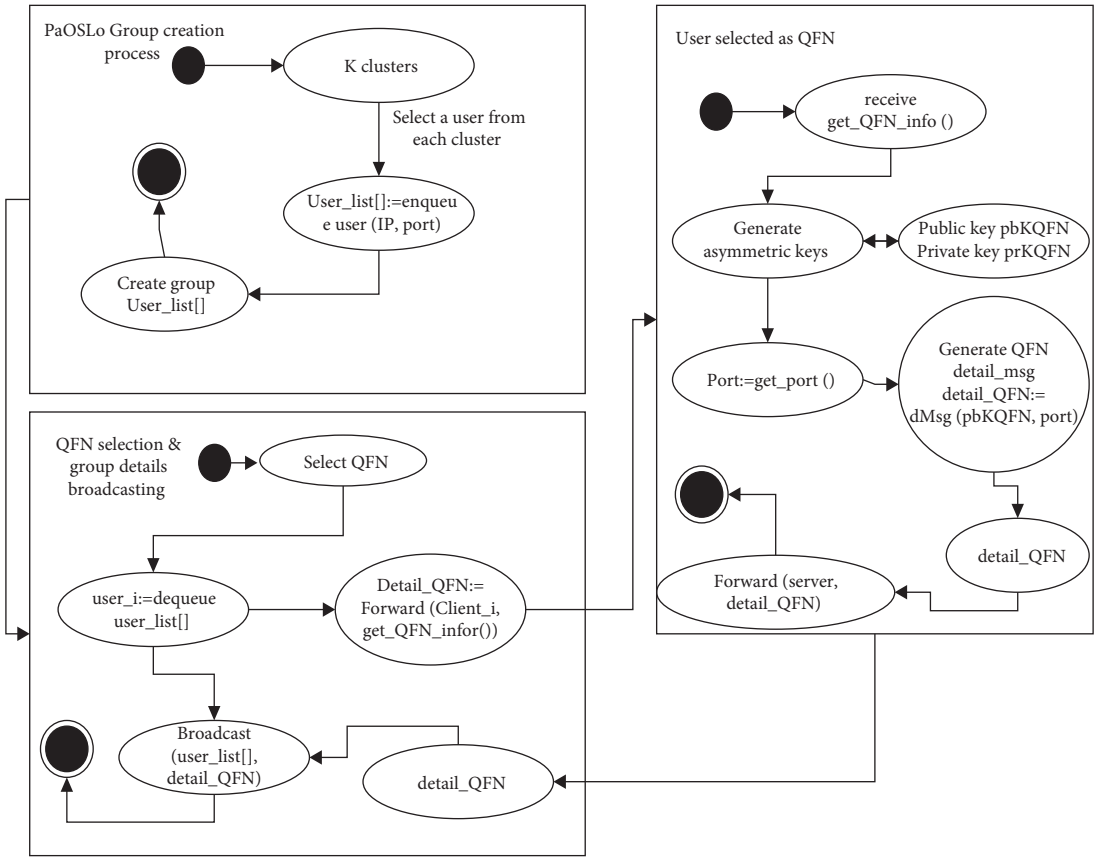


Figure 2: PaOSLo: activity diagram of group making and query forwarding node selection process.

3.3. Query Sending and Results Retrieval Procedure. Once the group making and QFN designation process concludes, the CS shares the user_list and detail_QFN message in the group. All users of the group receive this information. Figure 3 depicts the activity diagram of the query sending and result retrieval process. To send a query, a user (U_i) first gets information about the group users and the selected QFN. The user (U_i) creates a query and a cryptographic key (K_U_i). The Ui concatenates q and K_U_i making a QMsg. The U_i then encrypts QMsg with the public key of QFN, producing an encrypted query message (eQ). Afterwards, the U_i generates a q_ID, which the U_i will use for result identification. The U_i then generates the encrypted message (eMsg) by concatenating the encrypted message (eQ) and q_ID. The query encryption ensures the confidentiality of the query contents.

Once the query encryption process concludes, the eMsg is shuffled among the group users. The U_i flips a coin to decide where to forward the eMsg. If the coin produces a head, the U_i forwards the eMsg to QFN; if the coin produces a tail, the U_i forwards the eMsg to another random user U_j. The U_j on receiving eMsg does the same coin tossing and takes the same action on the results of head or tail. The shuffling of eMsg ensures that the query is unlinkable with the user. After a few passes, the eMsg reaches the QFN. The QFN in the first step separates the eMsg. The QFN gets all three parts distinctly (eQ, q_ID, and K_U_i). Afterwards, the QFN fetches the original query by decrypting the eQ with the private key. Afterwards, the QFN fetches the original query by decrypting the eQ with the private key. In the following step, QFN dispatches the query to WSE, which searches the relevant data over the Internet and delivers back the search query results (r). The QFN enciphers the results (r) with the cryptographic key (K_U_i) of the user. The QFN then appends q_ID with the encrypted results making an encrypted answer message (eAnsMsg). The QFN broadcasts the eAnsMsg to all group users. All users in the group receive the eAnsMsg. Each tries to match the q_ID to find out if the results are for their query. If the q_ID matches, the user decrypts the results (r), and the query sending process completes. Otherwise, the user drops the eAnsMsg.

4. Methodology

4.1. Dataset. America Online (AOL) released a three-month query log of 650 thousand users in 2006 for research [3, 19, 20]. It consisted of twenty million queries, but AOL did not inform the users about the release of the query log and that it would be freely obtainable [21]. To achieve the unlinkability between the users and queries, AOL had removed some information such as IP address, e-mail address, name, and other personal data from the query log. The query log is comprised of 5 features: "AnonID" describes the anonymous ID given to a user. Query, the actual search word of a user. Query Time, indicating the temporal information, ItemRank, and ClickURL, the URL clicked by the user after

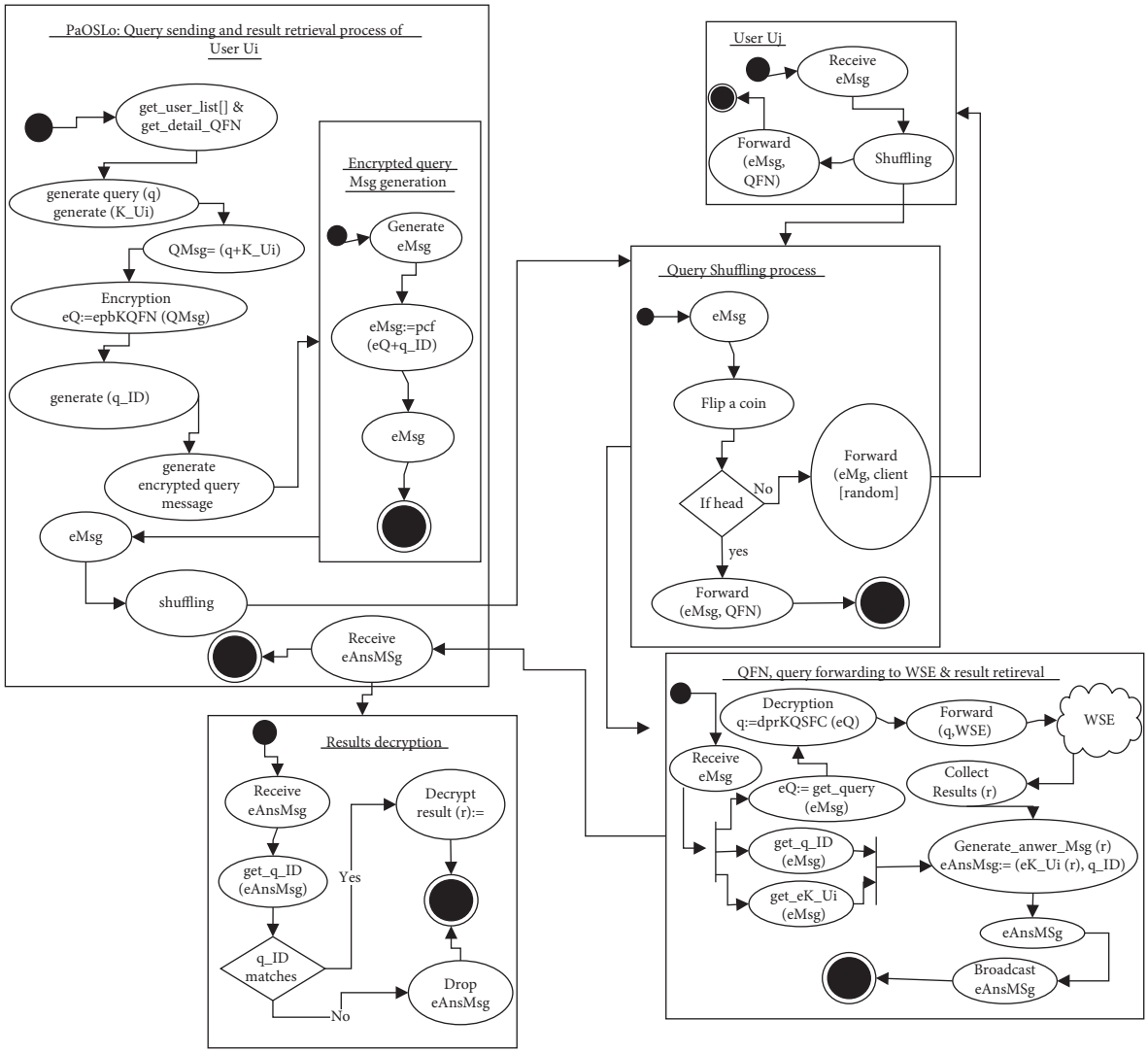


Figure 3: PaOSLo: activity diagram of sending and result retrieval process.

obtaining the results from AOL [19]. This query log is considered as a principal data source for evaluating web search privacy [2]. Piddinti and Saxena statistically examined various features of the AOL query log. The study shows that 98.72% of users have sent fewer than 100 queries over three months [2, 22]. 70% of people had fewer than thirty searches. In this work, we have selected a subset of the AOL query log dataset consisting of 1000 users ranging from highly active to minor active users for experimentation. By highly active, we mean those users who have sent more queries to the WSE, e.g., more than 200 queries. Similarly, minor active users are those who have sent less than 200 queries to the WSE. The selected users had sent a minimum of 25 queries to a maximum of 1514 queries. We have chosen the same dataset used by Ullah et al. [2] for the experiment. Table 1 shows the statistics of the users selected for the PaOSLo experimentation.

4.2. User Profile Construction. WSE builds the profile of the user from the queries it collects from the user. The WSE uses this profile to provide personalized results. Authors in [2, 3, 5, 11, 16, 19] have described the steps to build the user profile from the search queries. We have followed the same steps to create the user profile. Figure 4 shows the activity diagram of user profile construction. There are two significant steps taken in this process. This first step involves morphosyntactic analysis such as sentence detection, tokenization, part-of-speech tagging, and stop word removal for getting the query's actual topic defined by Cohen and Dolbey [23]. Semantic analysis is the second step of user profile construction. The words obtained by morphosynthetic are forwarded to Dmoz.org for user profile construction. Dmoz.org commonly referred to as the Open Directory Project (ODP) is an extensive Web directory managed by an alliance of volunteers [16, 24]. ODP classifies

Table 1: Dataset description.

Number of users in the dataset	1000
Total number of queries sent by the users	103644
Minimum-maximum queries by user	25–1512
Average queries by a user	103

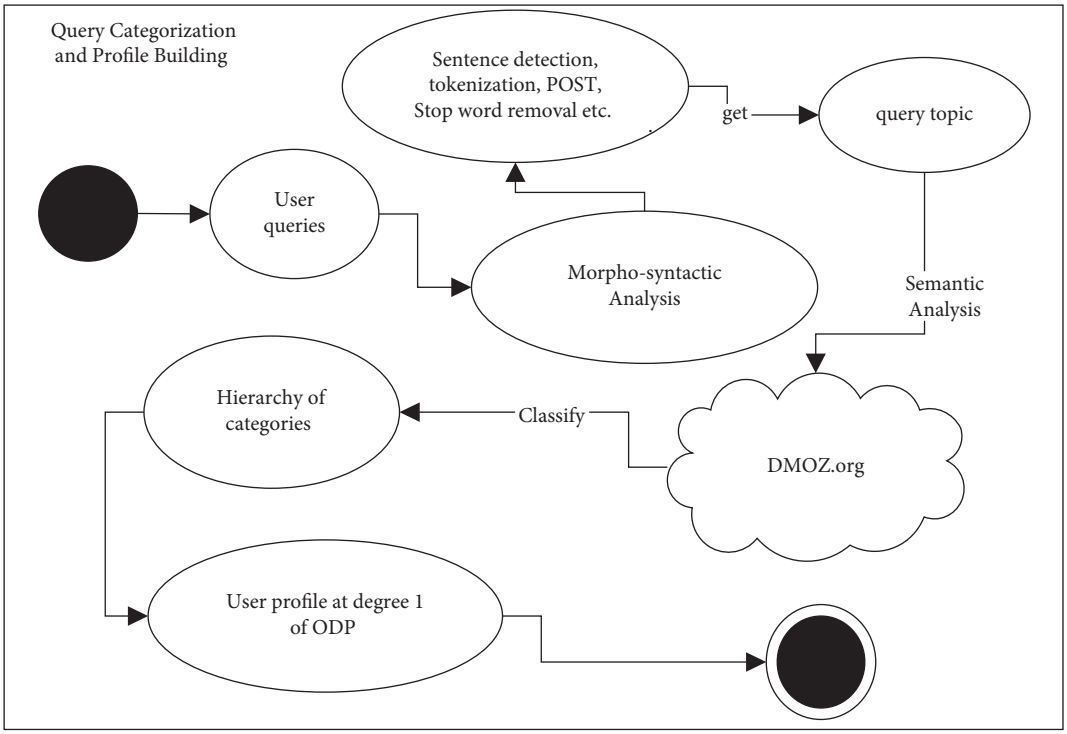


Figure 4: Activity diagram of user profile construction.

the search keywords into a hierarchy of levels. At the top level or first degree there are sixteen various classes. Each class has further subclasses, and so on and so forth. It has around 1 million distinctive classes [2, 16, 24]. Figure 5 shows the categories at first degree or top level of the ODP hierarchy. ODP categories any query into one of these 16 categories at degree 1. Table 2 shows a sample of query categorization by ODP. For example, a query "Valley National Banker" is categorized as "Business: Financial Services: Banking Services: Credit Unions: Regional: United States:" by the ODP. At degree 1, the query is represented as "business," at degree 2 as "Financial service," banking services at the third degree, and so on. By following the abovementioned steps, the profiles of all users are built.

4.3. Similarity Computation. After building the users' profiles, the categories/terms at degree 1 of a user profile are collected to compute the similarity between the users' profiles. Table 3 represents the term count at degree 1 of the ODP hierarchy of two sample users with AnonID "3978802" and AnonID "280617." The cosine similarity metric computes the likeness between two vectors. It is a function of the angle between the vectors of two users' profiles in the term vector space [26]. Equation (1) calculates the similarity between vectors A and B [6, 18]. The similarity function returns a value ranging from 0 to 1. The 0 represents distinct profiles, and the 1 donates the same matching profile. Based on the values of Tables 3 and 4, the similarity between user AnonID "3978802" and AnonID "280617" is 0.335. Following the same steps, the similarity between 1000 users is computed, and a 1000×1000 profile similarity matrix is made.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{B_i^2}}. \quad (1)$$

4.4. Users Clustering. Once the similarity calculation process is completed, profile clustering, the second step of PaOSLo, is performed. The 1000×1000 profile similarity matrix is converted into Attribute-Relation File Format (ARFF) and loaded into the Weka tool for clustering. Using the K-mean algorithm, the Weka tool clusters the users based on profile similarity into three clusters, four clusters, and five clusters [8]. The distance between the objects (users' profiles) is determined by the Euclidean distance. If the values of Euclidean distance between the users' profiles are low, the users are assigned to the same cluster, but they are different in other cases.

Table 4 shows the count of users allotted to each cluster. It is calculated over the term count of degree 1 categories of the ODP hierarchy.

Figure 6 shows a term count in each category of a first degree in ODP hierarchy. The X-axis of the graph's x-axis shows the top level of 16 categories of ODP, whereas the y-axis shows the count of the terms in each cluster. Cluster 1 contains those users who have frequently sent queries from the "Regional" category. Cluster 2 has users who have sent maximum queries from the "Business" category. Furthermore, cluster 3 has users who mostly sent queries from the "Arts" category. Similarly, Figure 7 shows that when users are grouped into four clusters, cluster 1 contains users who have forwarded queries from the "Arts" category. Cluster 2 contains users interested in "Business" queries; cluster 3 contains

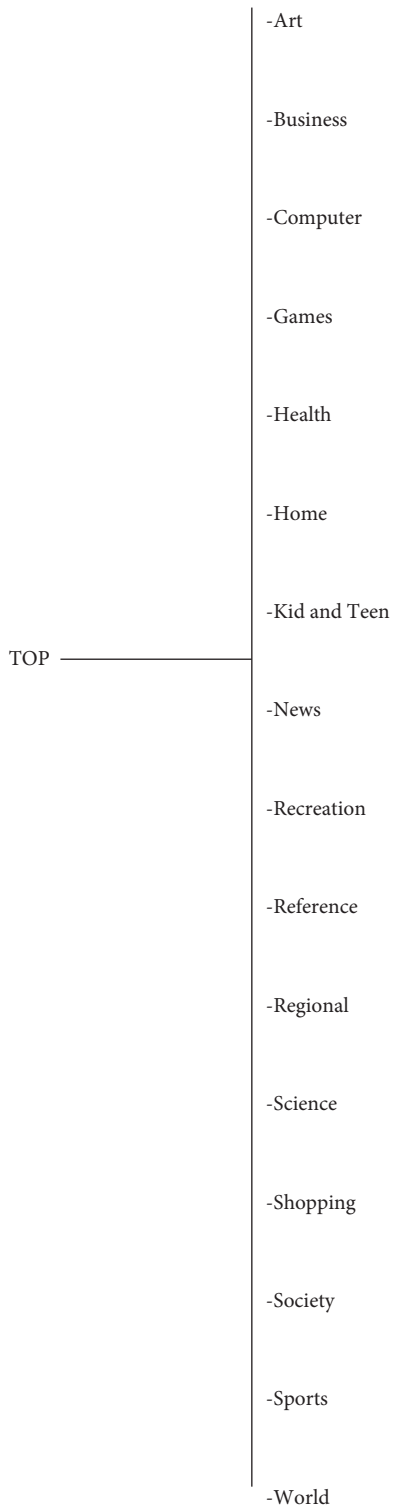


Figure 5: Top level categories of the ODP hierarchy [25].

users of "Regional" interests; and cluster 4 of the "Computer" category.

Figure 8 shows when users are grouped into five clusters based on the term count of degree 1 of the ODP hierarchy. Cluster 1 contains users who have sent maximum queries from the "Arts" category. "Computer" related queries dominate cluster 2, cluster 3 with "Regional" interests, and cluster 4 with "Recreation," while cluster 5 users are interested in "Business" queries.

5. PaOSLo Privacy Evaluation

The primary goal of a user simulating a distributed privacy-preserving protocol is to accomplish the following objectives:

(i) A user aims to attain confidentiality, i.e., the query content and result to query remains hidden from the group entities

(ii) A user intends to achieve unlinkability such that a query cannot be linked back to the user

(iii) A user endeavors to accomplish indistinguishability such that the WSE shall not create the actual profile

Local privacy and profile privacy are the aspects that researchers have described to assess the privacy of distributed protocols [2, 11, 16, 20]. Local privacy computes the probability of associating a query with the user by a curious entity of PaOSLo. Furthermore, profile privacy estimates the extent of user profile obfuscation.

A user executing PaOSLo achieves local privacy through confidentiality, query shuffling, and profile privacy by forwarding other users' queries. A user attains the confidentiality of a query and results as both the query and results are encrypted. The query encryption is performed using the RSA encryption algorithm. As mentioned earlier, the user encrypts the query (q) with the public key of QFN. No group user will be able to see the query content. Similarly, the QFN encrypts the results with the user's cryptographic key (K_Ui). We have used the AES encryption method for result (r) encryption. In such a case, the query and results will remain concealed from users of the group peers. Furthermore, to break the connection concerning a user and a query, it is shuffled among the peers. Even when the QFN receives a query from the user, it will not be able to determine if the query they received from the user is the query's originator or just a forwarder.

Additionally, as a user forwards the queries of group peers, their profile retained by WSE will be obfuscated with the diverse interests of the group peers. In such a case, the user achieves indistinguishability. The following section presents a thorough assessment of local privacy and profile privacy.

5.1. Local Privacy. To compute the local privacy, we must calculate the probability of associating a query by a curious entity with the originator. Let two random variables, Sr and Py, where Sr denotes the source of the query, and Py represents the proxy (a user in the group) that passes the query to the QFN. Suppose there are "*n*" users in the group. If the QFN wants to find the originating user of a suspicious query (q), the chances of associating a (q) to "Ui" are stated as follows:

Table 2: Example of query classification by ODP.

Query	ODP classification at different degrees
Valley National Banker	Business: financial services: banking services: credit unions: regional: United States
Photography Studios	Arts: photography: techniques and styles: documentary: photographers
mac.com	Computers: software: operating systems: MacOS: Internet

Table 3: Terms extracted for degree 1 of users AnonID "3978802" and AnonID "280617."

Query class	User 3978802	User 280617
Arts	2	0
Business	2	6
Computers	2	5
Games	0	1
Health	7	0
Home	0	3
Kids and teen	1	0
News	0	2
Recreation	1	0
Reference	1	1
Regional	1	4
Science	0	0
Shopping	0	0
Society	1	1
Sports	0	0
World	3	0

Table 4: Number of users in each cluster after K-mean clustering.

Cluster counts	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Three clusters	390	306	304	—	—
Four clusters	266	188	311	235	—
Five clusters	241	122	222	236	179

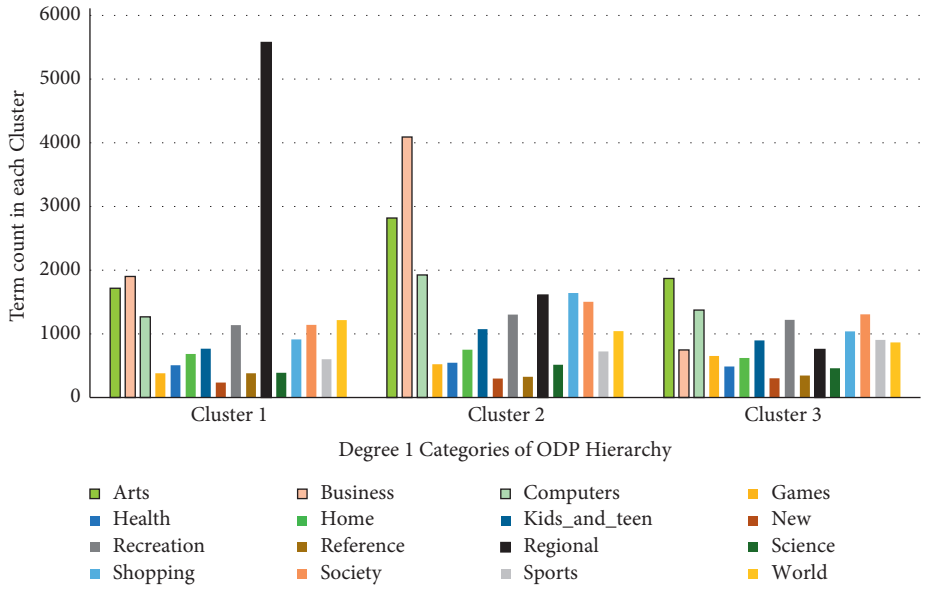


Figure 6: Term count in three clusters.

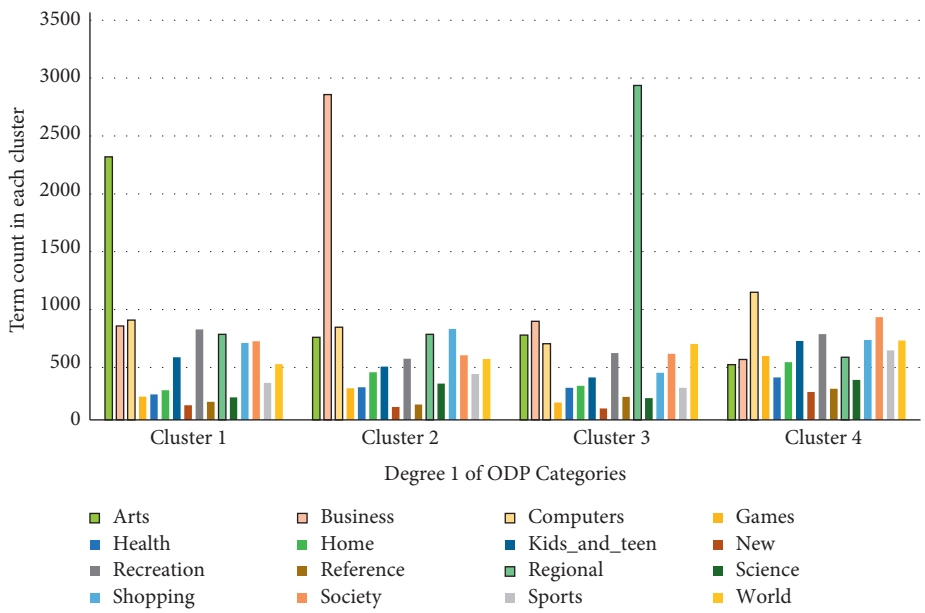


Figure 7: Term count in four clusters.

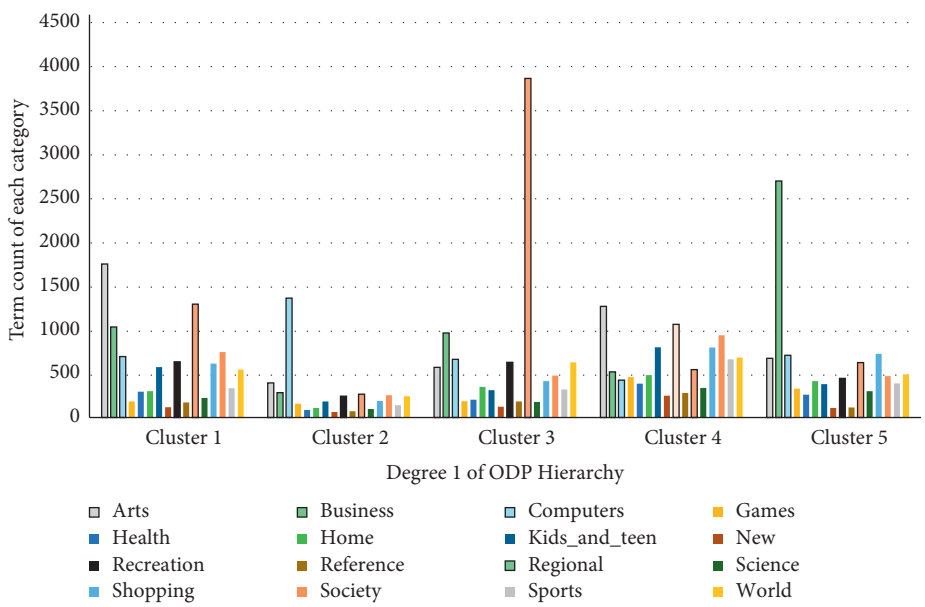


Figure 8: Term count in five clusters.

$$\Pr[\mathrm{Sr} = \mathrm{Ui} \mid \mathrm{Py} = \mathrm{Uj}] = \frac{\Pr[\mathrm{Py} = \mathrm{Uj} \mid \mathrm{Sr} = \mathrm{Ui}] \cdot \Pr[\mathrm{Sr} = \mathrm{Ui}]}{\Pr[\mathrm{Py} = \mathrm{Uj}]}. \tag{2}$$

Equation (2) represents the probability of source when the proxy is given to curious entity.

$$\Pr[\mathrm{Py} = \mathrm{Uj} \mid \mathrm{Sr} = \mathrm{Ui}] = \Pr[\mathrm{Py} = \mathrm{Uj}]. \tag{3}$$

$$\Pr(\mathrm{Sr} = \mathrm{Ui}) = \frac{1}{n-1}. \tag{4}$$

Equation (4) represents the probability of source Ui where "n" represents the number of users in a group and i, $j \in (1, \ldots, n)$, as QFN is not the query source, QFN excludes itself.

$$\Pr(\mathrm{Sr} = \mathrm{Ui} \mid \mathrm{Py} = \mathrm{Uj}) = \frac{1}{n-1}. \tag{5}$$

Equation (5) shows that the probability of associating a query (q) with Ui by QFN depending on the count of users in a group. However, if QFN and C users collaborate to identify the query (q) originator, then the chance of associating (q) is given in.

$$\Pr(\mathrm{Sr} = \mathrm{Ui} \mid \mathrm{Py} = \mathrm{Uj}) = \frac{1}{n-C}. \tag{6}$$

Table 5: Average PEL comparison of UUP(e), OSLo, and PaOSLo.

Number of users	Protocol	Degree1	Degree 2	Degree 3	Degree 4
3 users	UUP(e)	51.86	13.39	7.3	7.22
	OSLo	47.7	12.79	6.95	7.17
	PaOSLo	46.65	11.14	5.5	6.01
4 users	UUP(e)	51.16	13.14	7.07	7.2
	OSLo	48.56	12.76	6.85	6.95
	PaOSLo	46.32	10.6	5.44	5.75
5 users	UUP(e)	51.55	13.47	7.37	7.26
	OSLo	49.18	12.86	6.99	6.85
	PaOSLo	46.29	10.58	5.39	5.92

Equation (6) shows that if QFN makes a coalition with C users, the chance of associating (q) with an initiator $1/(n-C)$ which means all compromised C users will be excluded from the list. The value of n-C must be greater than 1.

5.2. Profile Privacy. Profile privacy measures the user's privacy in comparison to the WSE. Equation (7) shows a profile exposure level (PEL). It is an estimation metric utilized to evaluate the profile privacy. PEL computes the variation between an original profile and an obscured profile of an individual. The difference between the user's profiles has been calculated using entropy and mutual information [2, 11, 16]. It is necessary to mention that the original profile is made from the queries sent directly to WSE. In contrast, the obfuscated profile is built from the queries after the execution of PaOSLo.

$$\mathrm{PEL} = \left(\frac{I(P,Q)}{H(P)} \times 100\right). \tag{7}$$

$H(P)$ represents the entropy, whereas $I(P, Q)$ denotes the mutual information. The value of PEL is between 0 and 100, where 100 means fully exposed and 0 means no exposure.

6. Results and Discussion

This segment provides a comprehensive explanation of simulations conducted for the estimation of the profile privacy of users by executing PaOSLo. To perform experiments, we have developed a Java-based program to simulate PaOSLo using java socket programming. In the first step, the users get connected to the CS, and then the CS creates groups by following the steps mentioned in Section 3. We have adopted a CryptoUtil key pair generator method to generate RSA public-private key pairs and AES encryption key. The simulation of PaOSLo is performed over Intel i5-11400F CPU with 8 Gb RAM over Windows 10. After the simulation process, each user's query log is obtained and developed their profile from it. The profile privacy a user succeeds by executing PaOSLo has been compared with modern distributed privacy-preserving protocols such as OSLo [11] and UUP(e) [16].

Table 5 shows the average PEL a user achieves by executing PaOSLo. The average PEL at ODP's first degree for the group consisting of three users is 46.56%. Likewise, at degree 2, the average PEL drops to 11.14%, the average PEL further drops to 5.5% at degree 3, and so on. Similarly, for a group count of four, the average PEL is 46.32% at degree 1, 10.6% at degree 2, 5.44% at degree 3, and 5.75% at the fourth degree ODP. Likewise, the results at degree 1 for the group count of five is 46.29%, 10.58%, 5.83%, and 5.92% at the second, third, and fourth degrees of the ODP hierarchy, respectively. On the other hand, the results show that the OSLo provided 47.7% average PEL, and UUP (e) depicted 51.86% average PEL for the group size of three users at degree 1 of the ODP hierarchy. The average PEL values of UUP (e) dropped 51.16%, whereas the OSLo value reached 48.56% for the group size of four users. Similarly, when the group size is increased to five users, the UUP (e) showed 51.55% and OSLo showed 49.19% average PEL at degree 1 of the ODP hierarchy. The results show that the average PEL of a user simulating PaOSLo decreases when the group size is increased. A similar pattern is observed at all degrees of the ODP hierarchy. Such a decrease is because the user's profile is obfuscated by multiple users having different interests. However, in a random grouping, the chances of getting grouped with users having similar interests are higher, resulting in lower-profile obfuscation or higher profile exposure.

In this study, the profile privacy a user achieves after executing PaOSLo is compared with the modern privacy-preserving protocol UUP(e) and OSLo. This comparison is between a random grouping of users and a profile-aware grouping. Figure 9 shows that PaOSLo has less average PEL at all degrees of the ODP hierarchy than UUP(e) and OSLo for any group size. The PaOSLo has 10% less average PEL than UUP(e) and 2.5% less than OSLo at first degree of ODP for the group count of three members. The PaOSLo has 9.6% and 4.7% smaller average PEL than UUP(e) and OSLo at first degree of the ODP for the group of four members. Similarly, for five users group, the PaOSLo has 8.7% and 4.36% improved profile privacy compared to the abovementioned protocols. A similar effect is observed at a higher degree of ODP hierarchy, the PaOSLo outperform their counterpart OSLo and UUP(e). Results show that PaOSLo preserved better profile privacy than UUP(e) and OSLo at all degrees of ODP for any group size. The reason for the PaOSLo better results is the fact that it first clusters users based on their profile similarities compared to the random grouping. In such a case, the group created by CS from clusters contains users of different interests. Each user forwards a query of users from the other clusters; hence, a user's profile is

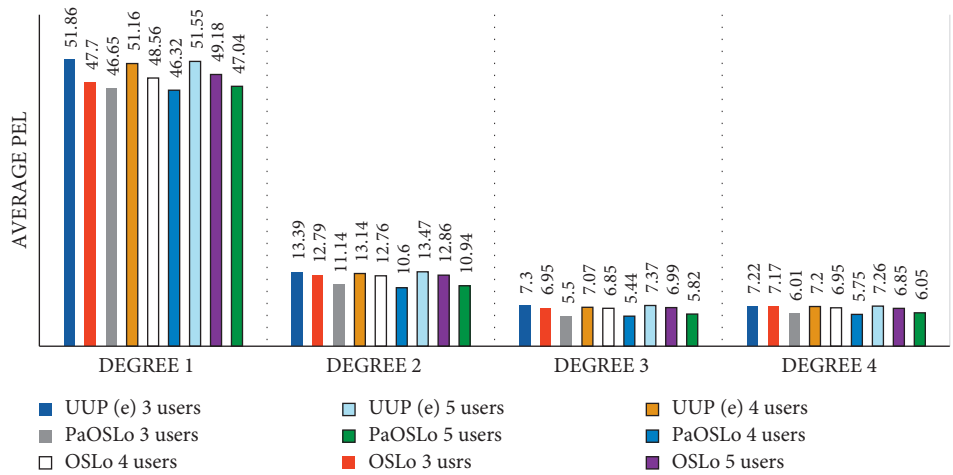


Figure 9: Average PEL of PaOSLo vs. OSLo vs. UUP(e).

Table 6: List of abbreviations and acronyms used in this article.

Abbreviation	Explanations	Abbreviation	Explanations
WSE	Web search engine	QMsg	Query message
PaOSLo	Profile Aware ObScure Logging	eQ	Encrypted query message
ODP	Online directory project	eMsg	Encrypted message
TOR	The onion routing	r	Results
CS	Core Server	eAnsMsg	Encrypted answer message
UPIR	User private information retrieval	AOL	America Online
TP	Third party	AnonID	Anonymous ID
QFN	Query forwarding node	Pr	Probability
Q	Query	S	Source
Ui	User i	P	Proxy
Uj	User j	PEL	Profile exposure level
K_Ui	Encryption key of Ui	QMsg	Query message

obfuscated with diverse interests' multiple peoples, and thus, the user achieves maximum obfuscation. Table 6 shows the list of abbreviations and acronyms used in this article.

7. Conclusion and Future Work

The existing distributed privacy-preserving protocols create a group of "*n*" users on a first-come, first-serve basis. Users are randomly grouped to forward each other's queries to the WSE without any prior knowledge of their interests. In random grouping, there is a greater chance that users having similar interests may be group together. Such random grouping has a trivial effect on profile obfuscation. To overcome the limitation of random grouping on web search privacy, this study recommends a novel distributed privacy-preservation protocol, PaOSLo. It obfuscates a person's profile with queries of those who have different interests. Systematic grouping is the primary objective of this work. Profile building and finding the similarity between users' profiles are the fundamental steps of PaOSLo. The measures the resemblance between the users' profiles cosine similarity is used. The K-mean algorithm clusters the users into three, four, and five clusters based on the similarity computed in the previous step. We experimented with and evaluated the impact of systematic grouping and compared random grouping with three members, four members, and five members. Results show that the systematic group depicts better profile privacy compared to the random group.

In the future, we are interested in investigating the impact of systematic grouping on the quality of the results. The impact of profile-aware grouping on the search results and the delay caused by clustering needs to be measured in the future. The profile privacy shall also be examined with other privacy metrics such as entropy, standard deviation, and KL divergence. Furthermore, maintaining privacy and personalization at the same time will be the future direction of this work. [27].

Abbreviations

WSE: Web search engine
PaOSLo: Profile Aware ObScure Logging
ODP: Online directory project
TOR: The onion routing
CS: Core Server
UPIR: User private information retrieval
TP: Third party
QFN: Query forwarding node
q: Query
U_i: User i

U_j: User j
K_Ui: Encryption key of Ui
QMsg: Query message
eQ: Encrypted query message
eMsg: Encrypted message
r: Results
eAnsMsg: Encrypted answer message
AOL: America Online
AnonID: Anonymous ID
Pr: Probability
S: Source
P: Proxy
PEL: Profile exposure level
QMsg: Query message.

Data Availability

The datasets used for the experimentation in this work are accessible at https://github.com/mrmohibkhan/Profile-aware-OSLo.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank SAUDI ARAMCO Cybersecurity Chair at Imam Abdulrahman Bin Faisal University (IAU) for supporting and funding this work.

References

[1] N. Kaaniche, S. Masmoudi, S. Znina, M. Laurent, and L. Demir, "Privacy preserving cooperative computation for personalized web search applications," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pp. 250–258, ACM, Brno Czech Republic, March 2020.

[2] M. Ullah, R. Khan, M. Inam Ul Haq et al., "Multi-group ObScure logging (MG-OSLo) A privacy-preserving protocol for private web search," *IEEE Access*, vol. 9, pp. 79005–79020, 2021.

[3] M. Rodriguez-Garcia, M. Batet, D. Sánchez, and A. Viejo, "Privacy protection of user profiles in online search via semantic randomization," *Knowledge and Information Systems*, vol. 63, pp. 1–23, 2021.

[4] A. Kumar and K. Hosanagar, "Measuring the value of recommendation links on product demand," *Information Systems Research*, vol. 30, no. 3, pp. 819–838, 2019.

[5] R. Khan, M. Ullah, A. Khan, M. I. Uddin, and M. Al-Yahya, "NN-QuPiD attack: neural network-based privacy quantification model for private information retrieval protocols," *Complexity*, vol. 2021, Article ID 6651662, 8 pages, 2021.

[6] R. Khan, M. A. Islam, M. Ullah, M. Aleem, and M. A. Iqbal, "Privacy exposure measure: a privacy-preserving technique for health-related web search," *Journal of Medical Imaging and Health Informatics*, vol. 9, no. 6, pp. 1196–1204, 2019.

[7] A. Raza, K. Han, and S. O. Hwang, "A framework for privacy preserving, distributed search engine using topology of DLT and onion routing," *IEEE Access*, vol. 8, pp. 43001–43012, 2020.

[8] J. Wu, *Advances in K-Means Clustering: A Data Mining Thinking*, Springer Science & Business Media, Berlin, Germany, 2012.

[9] J. Parra-Arnau, D. Rebollo-Monedero, and J. Forné, "Measuring the privacy of user profiles in personalized information systems," *Future Generation Computer Systems*, vol. 33, pp. 53–63, 2014.

[10] M. K. Reiter and A. D. Rubin, "Crowds: anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998.

[11] M. Ullah, M. A. Islam, R. Khan, M. Aleem, and M. A. Iqbal, "ObSecure Logging (OSLo): a framework to protect and evaluate the web search privacy in health care domain," *Journal of Medical Imaging and Health Informatics*, vol. 9, no. 6, pp. 1181–1190, 2019.

[12] J. Domingo-Ferrer, M. Bras-Amo'ros, Q. Wu, and J. Man'jon, "Userprivate information retrieval based on a peer-to-peer community," *Data & Knowledge Engineering*, vol. 68, no. 11, pp. 1237–1252, 2009.

[13] C. M. Swanson and D. R. Stinson, "Extended combinatorial constructions for peer-to-peer user-private information retrieval," *Advances in Mathematics of Communications*, vol. 6, 2011.

[14] C. M. Swanson and D. R. Stinson, "Extended results on privacy against coalitions of users in userprivate information retrieval protocols," *Cryptography and Communications*, vol. 7, no. 4, pp. 415–437, 2015.

[15] J. Castellá-Roca, A. Viejo, and J. Herrera-Joancomartì, "Preserving users privacy in web search engines," *Computer Communications*, vol. 32, no. 13-14, pp. 1541–1551, 2009.

[16] C. Romero-Tris, J. Castella-Roca, and A. Viejo, "Distributed system for private web search with untrusted partners," *Computer Networks*, vol. 67, pp. 26–42, 2014.

[17] J. Domingo-Ferrer, S. Martínez, D. SÆnchez, and J. Soria-Comas, "Coutility: self-enforcing protocols for the mutual benefit of participants," *Engineering Applications of Artificial Intelligence*, vol. 59, pp. 148–158, 2017.

[18] A. Huang, "Similarity measures for text document clustering," in *Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008)*, vol. 4, pp. 9–56, Christchurch, New Zealand, 2008.

[19] R. Khan, A. Ahmad, A. O. Alsayed, M. Binsawad, M. A. Islam, and M. Ullah, "Qupid Attack: machine learning-based privacy quantification mechanism for PIR protocols in health-related web search," *Scientific Programming*, vol. 2020, Article ID 8868686, 11 pages, 2020.

[20] R. Masood, D. Vatsalan, M. Ikram, and M. A. Kaafar, "Incognito: a method for obfuscating web data," in *Proceedings of the 2018 World Wide Web Conference*, pp. 267–276, ACM, Lyon, France, April 2018.

[21] D. Pàmies-Estrems, J. Castellà-Roca, and J. Garcia-Alfaro, "A real-time query log protection method for web search engines," *IEEE Access*, vol. 8, pp. 87393–87413, 2020.

[22] S. T. Peddinti and N. Saxena, "On the privacy of Web search basedon query obfuscation: a case study of trackmenot," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, pp. 19–37, Springer, Berlin, Germany, 2010.

[23] K. B. Cohen and A. Dolbey, "Foundations of statistical natural language processing," *Language*, vol. 78, no. 3, p. 599, 2002.

[24] N. C. Senthilkumar and Ch Pradeep Reddy, "Prediction of user interest fluctuation using fuzzy neural networks in web search," *International Journal of Intelligent Unmanned Systems*, vol. 8, no. 4, pp. 307–319, 2020.

[25] A. Viejo, J. Castella-Roca, O. Bernadó, and J. M. Mateo-Sanz, "Single-party private web search," in *Proceedings of the 2012 Tenth Annual International Conference on Privacy, Security and Trust*, pp. 1–8, IEEE, Paris, France, July 2012.

[26] A. Huang, "Similarity measures for text document clustering," in *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC 2008)*, vol. 4, pp. 9–56, Universities and Research Institutions, Christchurch, New Zealand, April 2008.

[27] M. Juarez and V. Torra, "Dispa: an intelligent agent for private web search," in *Advanced Research in Data Privacy*, pp. 389–405, Springer, New York, NY, USA, 2015.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 9692892, 15 pages
https://doi.org/10.1155/2022/9692892

Research Article

Radio Network Forensic with mmWave Using the Dominant Path Algorithm

Usman Rauf Kamboh,[1] **Muhammad Rehman Shahid**,[1] **Hamza Aldabbas**,[2] **Ammar Rafiq**,[3] **Bader Alouffi**,[4] **Muhammad Asif Habib**,[5] **and Ubaid Ullah**[1]

[1]*Department of Computational Sciences, The University of Faisalabad, Faisalabad 38000, Pakistan*
[2]*Prince Abdullah Bin Ghazi Faculty of Information and Communication Technology, Al-Balqa Applied University, Al Salt, Jordan*
[3]*NFC Institute of Engineering and Fertilizer Research, Faisalabad, Pakistan*
[4]*Department of Computer Science, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia*
[5]*Department of Computer Science, National Textile University, Faisalabad, Pakistan*

Correspondence should be addressed to Muhammad Asif Habib; drasif@ntu.edu.pk

Received 27 August 2021; Accepted 4 December 2021; Published 12 January 2022

Academic Editor: Mamoun Alazab

For the last two decades, cybercrimes are growing on a daily basis. To track down cybercrimes and radio network crimes, digital forensic for radio networks provides foundations. The data transfer rate for the next-generation wireless networks would be much greater than today's network in the coming years. The fifth-generation wireless systems are considering bands beyond 6 GHz. The network design of the next-generation wireless systems depends on propagation characteristics, frequency reuse, and bandwidth variation. This article declares the channel's propagation characteristics of both line of sight (LoS) and non-LOS (NLoS) to construct and detect the path of rays coming from anomalies. The simulations were carried out to investigate the diffraction loss (DL) and frequency drop (FD). Indoor and outdoor measurements were taken with the omnidirectional circular dipole antenna with a transmitting frequency of 28 GHz and 60 GHz to compare the two bands of the 5th generation. Millimeter-wave communication comes with a higher constraint for implementing and deploying higher losses, low diffractions, and low signal penetrations for the mentioned two bands. For outdoor, a MATLAB built-in 3D ray tracing algorithm is used while for an indoor office environment, an in-house algorithmic simulator built using MATLAB is used to analyze the channel characteristics.

1. Introduction

Present research work in the field of network forensic is on traditional networks. Because of the emergence of cellular or radio networks, radio network forensic provides the platform to investigate, capture, and detect faults, illegal activities, and cybercrimes [1].

Figure 1 illustrates the investigation process for digital forensic. The DFRWS model begins from identifying the crime. This identification is further divided to different tasks, for example, signature resolving, system monitoring, analysis, profile detection, complaints, and anomalous detection. The next step is the preservations. In preservations, we need to set up case management, management of the technology, insurance of custody chain, and synchronizing the time element. In the third step, data are collected by following the approved methodology. Examination is the fourth stage in DFWS. In examination, data are examined. Then, the data are analyzed, and tracing of evidences, validation of data, and filtering of data begin. In the presentation stage, we document the evidence, and final decision is taken. Frequencies greater than sub-6 GHz are not properly utilized in digital forensics due to which mmWave provides much greater bandwidth than the existing ones. On the one hand,

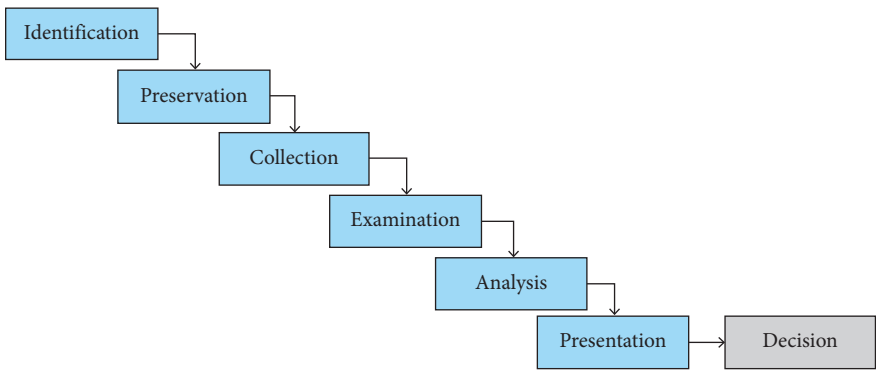


Figure 1: Investigative process for digital forensic science (DFRWS).

higher-frequency bands (e.g., mmWave bands) are not heavily utilized and, thus, offer larger bandwidths for wireless communication systems. Hence, determinations are focused upon the discovery of advanced frequencies as an alternate to the sub-6 GHz band [2]. On the other hand, the performance of the wireless communication system (WCS) is affected by multipath fading. As a matter of fact, the mobile device and the router for WCS have to cope up multipath manifestations which in other words demand deep knowledge of the propagation channel [3]. The designers of radio coverage make use of path loss prediction models and other channel parameters to deploy and install access points in an environment for the sake of better coverage and through put [4]. This work deals with extensive measurements, modeling, ray tracing, and simulations of channel characteristics in open outdoor as well as in narrow corridor and institution room environment. The frequency bands considered for the WCS modeling are 28 GHz and 60 GHz. We rationally premeditated received signal strength (RSS) by the principle of the "five rays" and "two rays" receiving power model.

To associate proposed model systematic results, simulations and measurements were performed at The University of Faisalabad, Faisalabad, Pakistan. The experimental campaign is shown in Figure 2(a). The outdoor scenario is shown in Figure 2(b). The transmitter indicated in red is roughly 250 m apart from the receiver shown in blue. The remaining part of the article is described as follows. A summary of the recent related work and the contribution are discussed in Section 2. Section 3 deals with the path loss modeling for both outdoor as well as indoor environment. Section 4 deals with the ray tracing modeling. Section 5 analyzes the detailed simulations and algorithm for the smart ray tracing algorithm. Finally in Section 6, we draw conclusion.

2. Recent Studies and Contributions

Numerous tactics are suggested in the literature to control the extraordinary attenuation issue on mmWave frequency bands [6, 7]. In [8, 9], the authors indicated that increment in the directivity or gain of the concerned antennas can be beneficial in high attenuation problems. Table 1 illustrates the recent findings in the field of mmWave. The model used, geometry, scenario, carrier frequency, and separation between the transmitter and receiver are mentioned against each reference. Another aspect is the material characteristics of the environment [32]. The need for ray tracing comes here as the indoor and outdoor environment difference in geometry so is the LoS and NLoS for numerous frequency bands. Ray tracing is used to model the propagation characteristics [33–36].

A separate ray source is presented in this article for end-to-end penetration. We considered five rays for indoor and five for outdoor. In indoor, we analyzed one ray for LoS and the rest of four for the wall, ground, and ceiling of the room. All the five rays considered to be contributing in received and illustrated with different colors. These remain because of the small separation between transmitter and receiver antennas. The reflected rays are excluded by Rx because of the directivity pattern. The high fresnel reflection coefficients are organised with reference to the radio links. An escalation in the RSS was spotted as compared to outdoors. The path loss for the outdoor open environment has greater slope than that for indoor.

Normally, two ray models are deliberated for outdoors because of their nature of modeling P_r in an unavailability scenario of 1st order reflections. Thus, no concrete solid evidence of the difference is found between PL slopes. At last, the calculated modeling is compared with experimental parameters and RT simulations. We also prepared a reasonable investigation of the measurements with 5-ray and 2-ray investigative models, and RT simulations with 5 rays are delivered for the indoor office environment.

In Figure 3(a), the probability of dipole antenna for the proposed model is marginal with the theoretical values. This indicates that the probability of dipole decreases with the increase in path loss. Additionally, the probability for close-in and 3GPP models deviates from the theoretical model. Figure 3(b) indicates the reactance and resistance of the dipole antenna. It is evident from the graph that the

(a) (b)

Figure 2: Measurement environment. (a) Indoor setup [5]. (b) Aerial view.

Table 1: Recent studies.

Reference	Model used	Geometry	Scenario	Antenna pattern	Frequency (GHz)	Tx-Rx (m)
[10]	Emp.	NLoS	Suburban	Direct	28	Unspecified
[11]	RT	NLoS	Opened Squared	Unspecified	Unspecified	Unspecified
[12]	Emp.	LoS/NLoS	Indoor	Direct	19, 28, 38	18
[13]	Emp.	LoS/NLoS	Indoor	Omni	3.5	20
[14]	Emp.	LoS	Outdoor	Direct./Omni	3.5	450
[15]	Emp.	LoS/NLoS	Outdoor Campus	Horn/Omni	28, 38	30
[16]	Emp.	LoS/NLoS	Indoor	Omni	40	3
[17]	RT	LoS/NLoS	Indoor	Horn	60	Unspecified
[18]	Emp.	NLoS	I2O	Dipole	19, 28, 38	13.5
[19]	Emp.	LoS/NLoS	Indoor	Horn	3.5, 28	20
[20]	Emp.	LoS/NLoS	Indoor	Horn	3.5	20
[21]	Emp.	LoS/NLoS	Indoor	Omni	310, 350, 390	0.7
[22]	Emp.	LoS/NLoS	Outdoor	Horn	38	50
[23]	Emp.	LoS	Indoor	Omni	25.5, 38,37.5, 39.5	8.5
[24]	Emp.	LoS	Indoor	Omni	2.4, 3.52,5.8	4
[25]	Emp.	LoS	Outdoor	Direct	Unspecified	28
[26]	RT	LoS/NLoS	Opened Squared	Unspecified	Unspecified	Unspecified
[27]	RT	LoS	Indoor	Isotropic	28	Unspecified
[28]	7 RT	LoS/NLoS	Indoor	Omni	28	Unspecified
[29]	RT	LoS/NLoS	Outdoor	Omni	28	Unspecified
[30]	Emp. and RT	LoS	Indoor	Emp. Horn 3D Direct.	28	22.7
[31]	3D RT	LoS/NLoS	Outdoor	Dipole	28	488
This study	3D RT	LoS/NLoS	Outdoor Campus	Omni/Horn	28, 60	See Tables 3 and 4

reactance and resistance are opposite to each other. The above recent literature when compared with our proposed work points out the following differences:

(i) Five dominant rays for indoor and two dominant rays for outdoor were adequate considering this work.

(ii) The distance of rays was considered to be resolvable when compared with LoS. Smaller is the distance of resolution and the rays superimposed on one another.

(iii) At 28 GHz and 60 GHz, a polarized dependent variable was utilized.

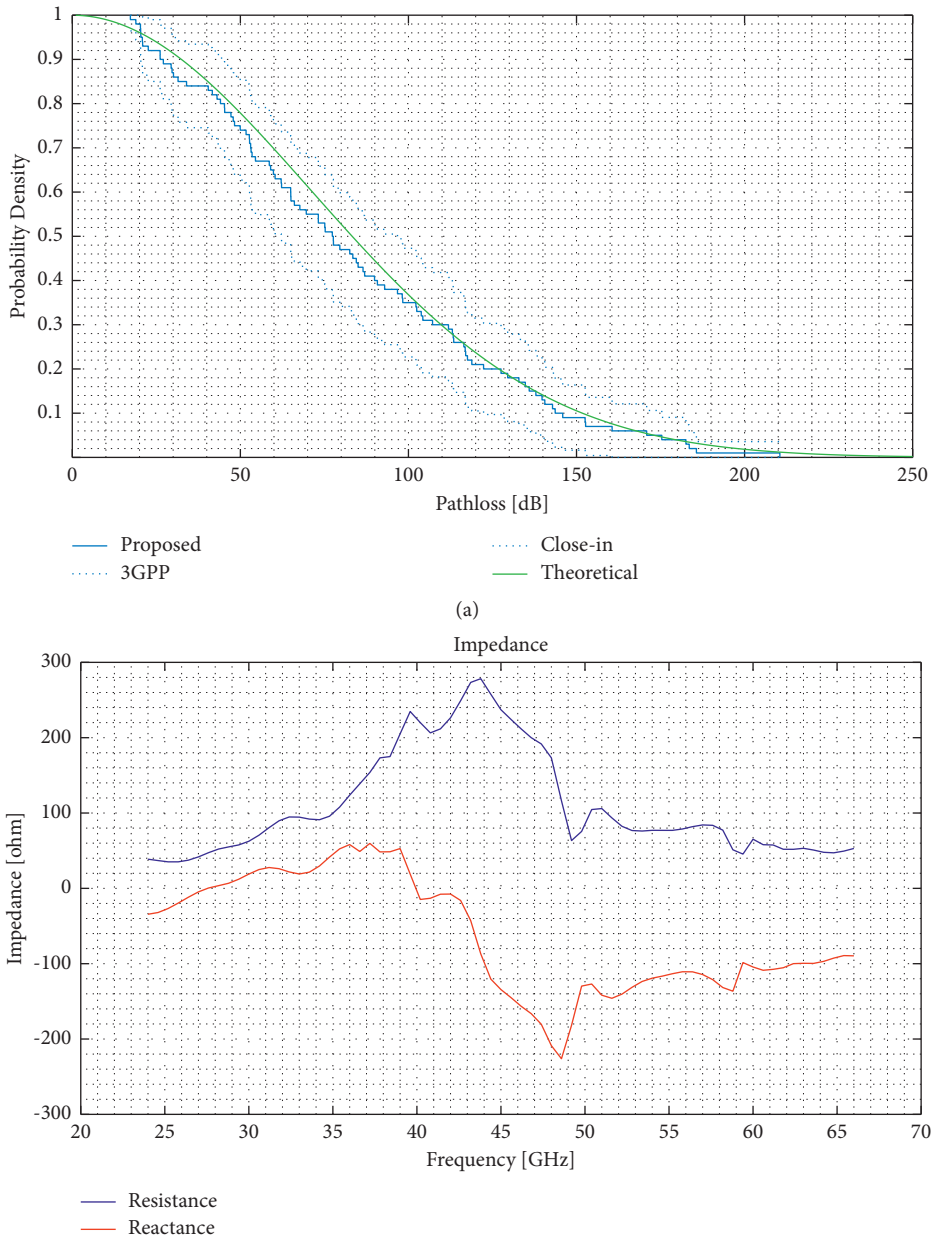


Figure 3: Circular dipole antenna characteristics. (a)Probability density of dipole antenna for different models; (b)resistance and reactance of dipole antenna.

(iv) The close deployment of Tx and Rx is very common in indoor scenarios nowadays. This scenario with a typical classroom environment is focused here.

(v) Antenna gains for Tx and Rx are closely monitored with the help of MATLAB simulations using antenna radiation patterns.

3. Path Loss Modeling

Figure 4 is taken from the recent studies conducted by authors to ray trace the path loss in a different environment with heatmap techniques using MATLAB Simulink. The close-in (CI) free-space path loss model is utilized to adjust the path loss from the result of simulations. The supremacy of CI is that its parameters are dependent on the operating frequency. The relationship of PL with frequency for CI is as follows [37]:

$$\mathrm{PL}^{\mathrm{CI}}(d)[\mathrm{d}B] = \mathrm{FSPL}(f, d_0) + 10n.\ \log\left(\frac{d}{d_0}\right) + X^{\mathrm{CI}}_{\sigma_{\mathrm{SF}}}. \quad (1)$$

$X^{\mathrm{CI}}_{\sigma_{\mathrm{SF}}}$ denotes the standard deviation. $\mathrm{FSPL}(f, d_0)$ represents the free-space path loss in dB and is given by

$$\mathrm{FSPL}(f, d_0) = 20\ \log\ 10\left(\frac{(4\pi d_0 f)}{c}\right), \quad (2)$$

where d_0 is the Tx-Rx separation at 1 m reference; f stands for the frequency of the carrier radio wave; c denotes the light speed, and n is the path loss exponent (PLE). In [38], WINNER II and 3GPP are illustrated with the help of floating intercept (FI). According to [38], FI is expressed as follows:

$$\mathrm{PL}^{\mathrm{FI}}(d)[dB] = \alpha + 10\beta\ \log_{10}(d) + Y^{\mathrm{FI}}_{\sigma_{\mathrm{SF}}}, \quad (3)$$

where σ_{SF} of $Y^{\mathrm{FI}}_{\sigma_{\mathrm{SF}}}$ is the Gaussian SF. Multifrequency path loss models are used for a variety of mmWaves. For this aspect, the famous alpha-beta-gamma (ABG) model with a 1 m reference distance and 1 GHz referred frequency is considered here as follows [39]:

$$\mathrm{PL}^{\mathrm{ABG}}(f, d)\ \ (\mathrm{dB}) = 10\alpha.\ \log\left(\frac{d}{d_0}\right) + 10\gamma.\ \log\left(\frac{f}{f_{\mathrm{ref}}}\right) + \beta + X^{\mathrm{ABG}}_{\sigma}. \quad (4)$$

In the above equation, α and γ denote constant coefficients. β denotes the offset, and $X^{\mathrm{ABG}}_{\sigma}$ is a Gaussian random variable. In literature [40], ABG is solved with the help of MMSE to minimize error.

Another derived model from CI and FSPL which is widely used for path loss analysis is CIF (Close in free space) [38]:

$$\mathrm{PL}^{\mathrm{CIF}}(f, d)[\mathrm{d}B] = 10\,m.\ \log_{10}(d_0).\left(1 + g.\frac{f - f_0}{f_0}\right) + X^{\mathrm{CIF}}_{\sigma}, \quad (5)$$

where m and g are the distance dependent and linear frequency dependent factors, respectively.

4. RT Modeling

Ray tracing involves two patterns. (a) ray launch and (b) the image theory. In the first line of attack, rays commencing Tx are launched with even angular departure and then traced. After tracing the rays, the received field will be calculated to find the power [41–43]. This approach has faster computational time but has less accuracy when the separation increases. An alternate approach as mentioned is the image method, in which the route is first defined of image points from Tx or Rx [41, 43]. This method of images has a drawback of high computational time and involves complex algorithms. The interaction mechanism implements propagation's theory; nonetheless, the deviation through different RT tools occurs due to the dissimilar method of modeling the similar phenomena (e.g., diffraction, diffusion, scattering, and reflection). Other deviations follow in the antenna design, pattern, sources, frequencies, and models. The constitutive material aspects, e.g., the magnetic permeability (μ), electrical permittivity (ϵ), and the conductivity (ρ), are essentials for accomplishing accurate channel estimations in RT [44]. Undeniably, Maxwell's electromagnetic wave equations for diverse edges are fighting fit for constitutive parameters. This directly affects the wave's phase and amplitude, therefore, eventually governs the influence of discrete multipath constituents. The influence of "material properties" on the RT estimations precision depends on the dominant propagation.

Accordingly, in situations wherever the LoS dominates, material properties can affect very little on the accuracy of the prediction algorithm. The modeling of diffused scatterings is usually carried out with the help of severe roughness models as proposed in literature [45]. Because the model is not dependent on the Maxwell equation, it solves the issue of diffused component's power. Besides, the roughness model is not dependent on the material properties; it can predict a diffuse scattered field. Fresnel's equation works on smooth surfaces for the analysis of reflection coefficients and transmission coefficients. The reflection coefficient for the smooth surface is given by [32]

$$\tau_{\perp} = \frac{e^r_{\perp}}{e^i_{\perp}} = \frac{\eta_1 \mathrm{Sec}\theta_t - \eta_0 \mathrm{Sec}\theta_t}{\eta_1 \mathrm{Sec}\theta_t + \eta_0 \mathrm{Sec}\theta_t}, \quad (6)$$

In the above equation, $e^i_{\perp}$ denotes the incident electric fields, θ_i represents the incident angles, $e^r_{\perp}$ expresses the reflected electric fields, η is the wave impedance, and θ_r declares the transmission's angle. The transmission coefficients are denoted by

$$t_{\perp} = \frac{e^t_{\perp}}{e^i_{\perp}} = \frac{2\eta_1\ \ \sec\ \theta_t}{\eta_1 \sec\ \theta_i + \eta_0 \sec\ \theta_t},$$
$$t_{\parallel} = \frac{e^t_{\parallel}}{e^i_{\parallel}} = \frac{2\eta_1 \sec\ \theta_t}{\eta_0 \sec\ \theta_i + \eta_1 \sec\ \theta_t}. \quad (7)$$

The constitutive parameters of the used materials will determine the impedance of the wave (η) and the also the angle of transmission θ_t:

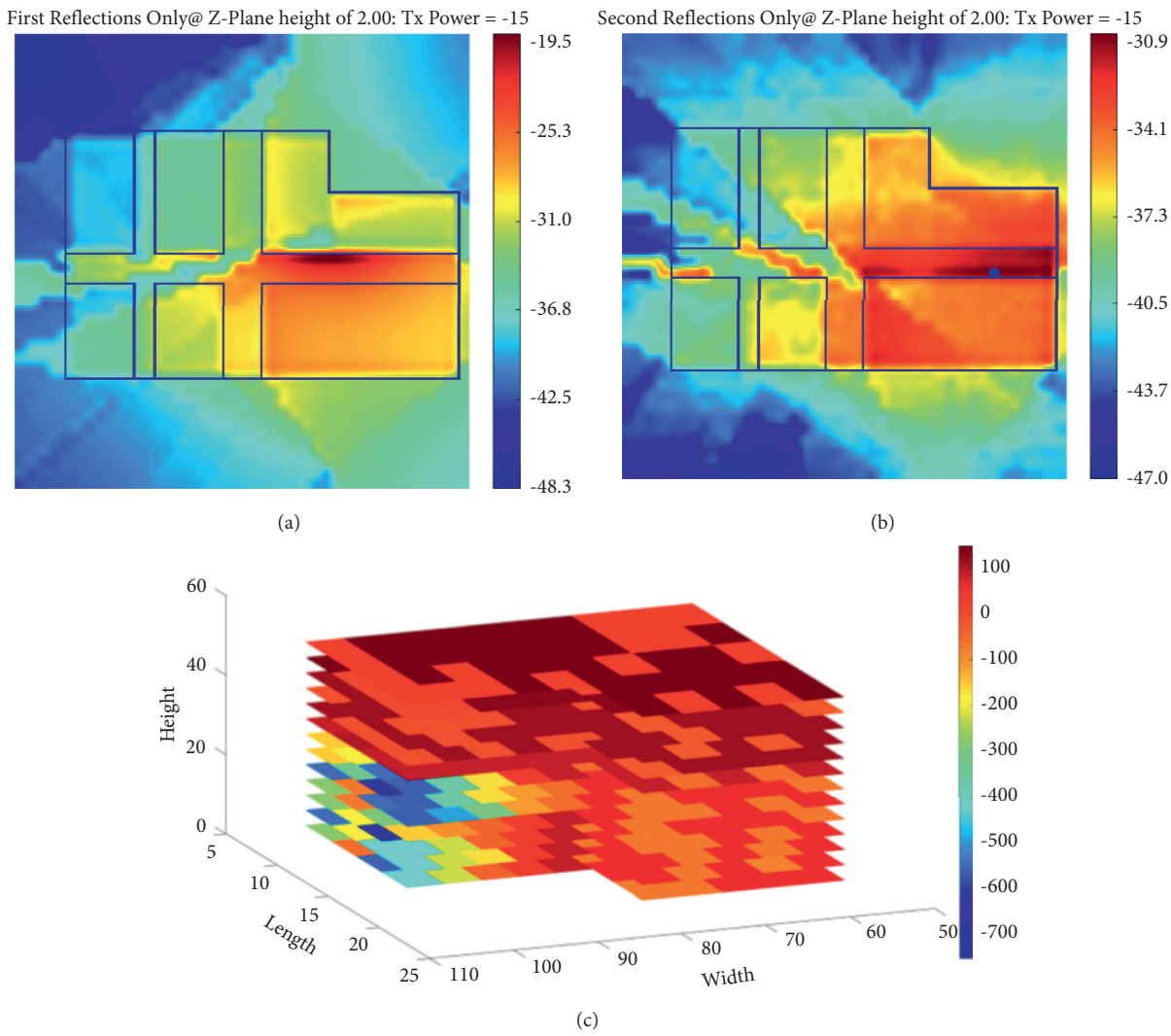


Figure 4: Recent related work [32]. (a) 1st reflection; (b) 2nd reflection; (c) heap map of outdoor simulations.

$$\eta = \sqrt{\frac{\iota\Omega\mu}{\rho + \iota\Omega\epsilon}},$$
$$\theta_r = \cos^{-1}\sqrt{1 - \left(\frac{p_0}{p_1}\right)^2 (\sin\theta_t)^2}, \tag{8}$$

where Ω stands for the angular frequency and p denotes the wave number for that frequency.

$$\Omega = 2\pi f,$$
$$p = \Omega\sqrt{\mu\epsilon - \frac{\iota\mu\rho}{\Omega}}. \tag{9}$$

More accurate results for diffracted wedges can be obtained by using the "Uniform Theory of Diffraction" (UTD). For UTD, the details of material constitutive parameter are necessary to be determined. Considering Figure 5, the diffraction coefficient can be obtained by [46]

$$D_{\parallel}^{\perp} = \frac{-e^{-\iota\pi/4}}{2 * p\sqrt{2\pi p}} \times \left\{D_{i_1} + D_{i_2} + t_{\parallel,0}^{\perp} D_{i_3} + t_{\parallel,p}^{\perp} D_{i_4}\right\}, \tag{10}$$

where

$$D_{i_1} = \frac{1}{\tan\left(\pi + (\Psi - \Psi')/2p.Y\left(vLb^{+}(\Psi - \Psi')\right)\right)},$$
$$D_{i_2} = \frac{1}{\tan\left(\pi - (\Psi - \Psi')/2p.Y\left(vLb^{-}(\Psi - \Psi')\right)\right)},$$
$$D_{i_3} = \frac{1}{\tan\left(\pi - (\Psi - \Psi')/2p.Y\left(vLb^{-}(\Psi - \Psi')\right)\right)}, \tag{11}$$
$$D_{i_4} = \frac{1}{\tan\left(\pi - (\Psi - \Psi')/2p.Y\left(vLb^{+}(\Psi - \Psi')\right)\right)},$$

where Ψ ' is the incidence angle and Ψ is the diffraction angle; the Fresnel integral $Y(v)$ can be expressed as follows:

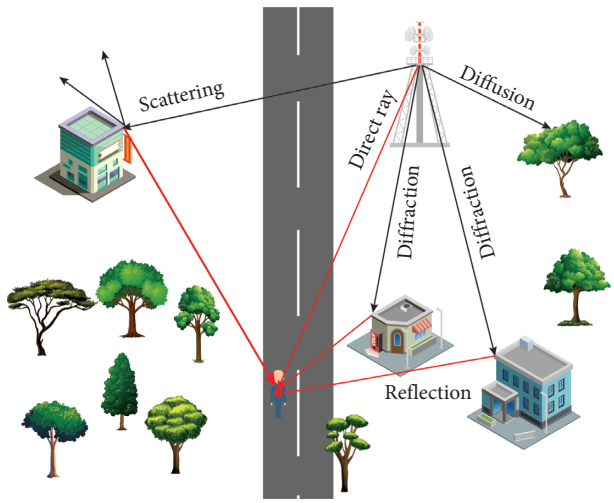


Figure 5: Dominant ray tracing concept [32].

$$Y(v) = 2j\sqrt{(v)e^{iv}}\int_{\sqrt{v}}^{\infty} e^{-it^2}. \tag{12}$$

L is defined as follows:

$$L = \frac{ss'}{s+s'}. \tag{13}$$

Figure 5 explains that n represents the wedge's factor, and s and s' are the distances. $a^{\pm}$ is defined as follows:

$$a^{\pm}(\Psi \pm \Psi') = 2\times\left(1 - \sin^2\left(\frac{2n\Psi M^{\pm} - (\Psi \pm \Psi')}{2}\right)\right). \tag{14}$$

Finally, $M^{\pm}$ are the integers of the following equations:

$$\begin{aligned} 2n\pi M^{+} - (\Psi \pm \Psi') &= \pi, \\ 2n\pi M^{-} - (\Psi \pm \Psi') &= -\pi. \end{aligned} \tag{15}$$

The information of material properties is hence crucial to precisely model the diffraction.

Figure 6 indicates the properties of different materials and diffraction loss occuring due to different material constitutes.

5. Simulations with the Dominant 3D Ray Tracing

In mmWaves (high frequencies), the concept of ray optics is considered. Maxwell equations are solved asymptotically. The wave length of the electromagnetic wave plays a key role in obtaining attractive and accurate results. Greater the size of the wavelength, the more accurate the results from an RT stimulation [47, 48]. Reliant on the electrogeometrical appearances available in digital maps, a ray undergoes diffuse scattering, diffraction, penetration, reflection, or attenuation. The penetration of rays joints upon accurate geometric and materialistic characteristics.

The simulation setup is shown in Table 2. We have considered a three-floored building. The path loss model we used for simulations is Close in. Two mmWave frequencies (28 GHz and 60 GHz are discussed for the complex layout of LoS and NLoS. The length and width of the wall are 2.8 m and 0.1 m, respectively. The rest of the parameters of the Base station and User terminal are mentioned in Table 2.

RT tool complications increase with the order of interaction mechanism. Nevertheless, with high order interaction, the expansions in relation to prediction precision are frequently negligible but even so acquire major computations overhead [49].

5.1. Simulations for Outdoor. The simulations performed in this campaign are based on the intelligent 3D ray tracer built-in MATLAB. This tool is different from other 3D RT tools as it is capable of performing full 3D ray tracing in terms of path loss and received power. Other ray-tracing tools can find many paths, but the mentioned tool is used to find the dominant paths. The 3D map of the concerned scenario is shown in Figure 7.

In this section, received signal strength (RSS) is the main parameter in designing the simulator. The antenna designed for outdoor simulations is an omni-directional circular dipole antenna with a transmitting frequency of 28 GHz and 60 GHz.

Figure 8(a) shows the current distribution of the antenna. The current at the center (circular) is greater than that at the bottom and top of the antenna. This is because of the feed which is at the center of the antenna. Figure 8(b) shows the 3D radiation pattern. The maximum radiation is 1.71 dBi and minimum is 13.2 dBi (Algorithm 1).

The height of the receiver (user with mobile phone) is 1.6 m. The user is standing still on the location mentioned in Table 2. Figure 7 indicates the RSS in dBm. Figure 7(a) shows RSS for 28 GHz. It is evident that the fig that the received power at Rx is about −50 dBm. Figure 7(b) shows RSS for 28 GHz. It is evident from the figure that the received power at Rx is about −40 dBm. The comparison of Figures 7(a) and 7(b) reveal that the RSS level has a mean difference of around 8 dBm–10 dBm between 28 GHz and 60 GHz. This is due to the fact that the huge gap between mmWave frequencies does not affect much power of the signal in small cell environments. On the other hand, this is the sign of low penetration, low signal diffraction, and very high free-space losses.

5.2. Simulations for Indoor. The indoor floor map of the simulation environment is shown in Figure 9. There are three transmitters in the facility. To understand the complex scenario for ray tracing, we choose the nearby transmitter, (in the corridor) (i.e., Tx2). Rx is in the office. Figure 10 illustrates the typical classroom where measurements were performed, and simulations were carried out. It consists of plywood, concrete tiles, plastic, brick work, plaster, and metal. Figure 6 shows the path loss of different material obstacles in the path of rays against a range of frequencies. It is evident that the loss due to bricks is constant throughout the range of mmWave frequencies (Algorithm 2).

Figures 11(a) and Figure 11(b) show the results of indoor simulations at 28 GHz and 60 GHz, respectively. Five

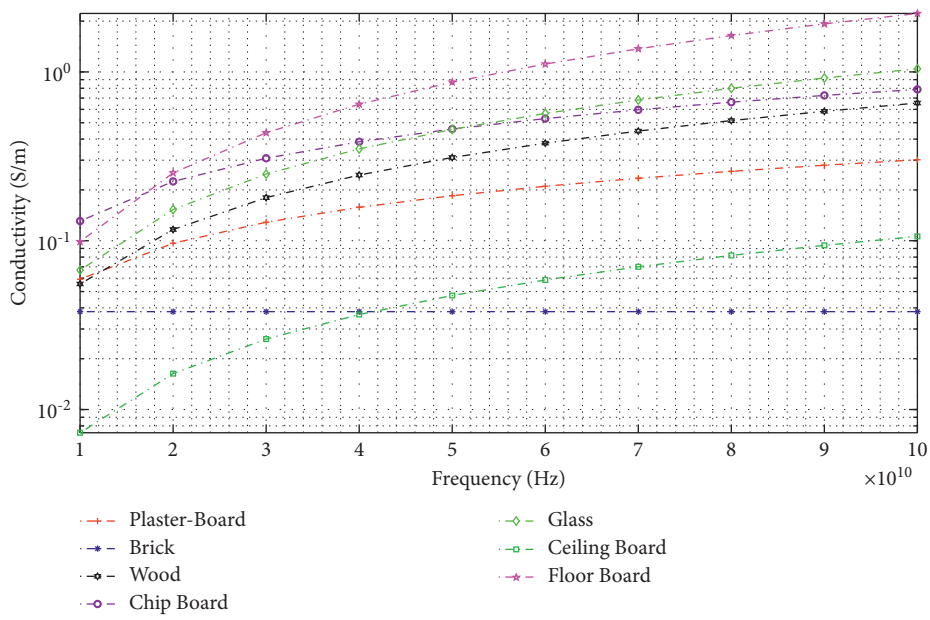


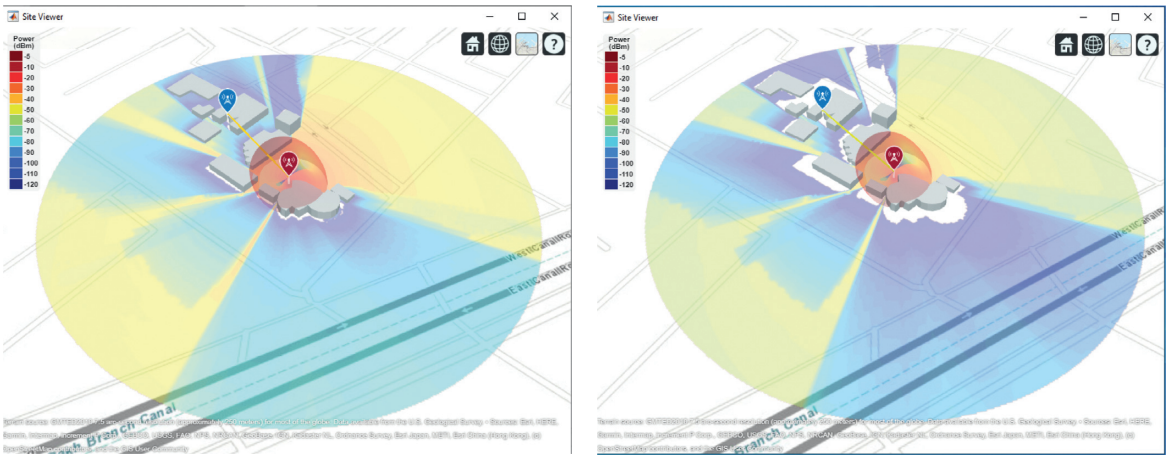
Figure 6: Permittivity of different materials.

Table 2: Simulation setup.

Parameter	Description	Unit value
Number of floors	3	
Cable losses (dBm)	3	
Ray tracing method	Close-in	
Propagation model	Full 3D	
Coupling losses (dB)	70	
Operating frequency	mmWave	28 and 60 GHz
Layout	Complex	LoS/NLoS
Fading	Model	Rayleigh
Cell structure	Hex (grid)	1
Building's wall	Length (m)	2.8
	Width (m)	0.1
Base station	Transmitting power (dBm)	21.0
	Height (m)	25.1
	Pattern	Dipole
	Bandwidth (GHz)	0.1
	Obstacle	Enabled
	Polarization	Latitude
User terminal	Receiving power loss (dBm)	2
	Longitude (m)	2.7
	Pattern	Omni
	Bandwidth (GHz)	0.1
	Obstacle	Enabled
	Polarization	Longitude/latitude

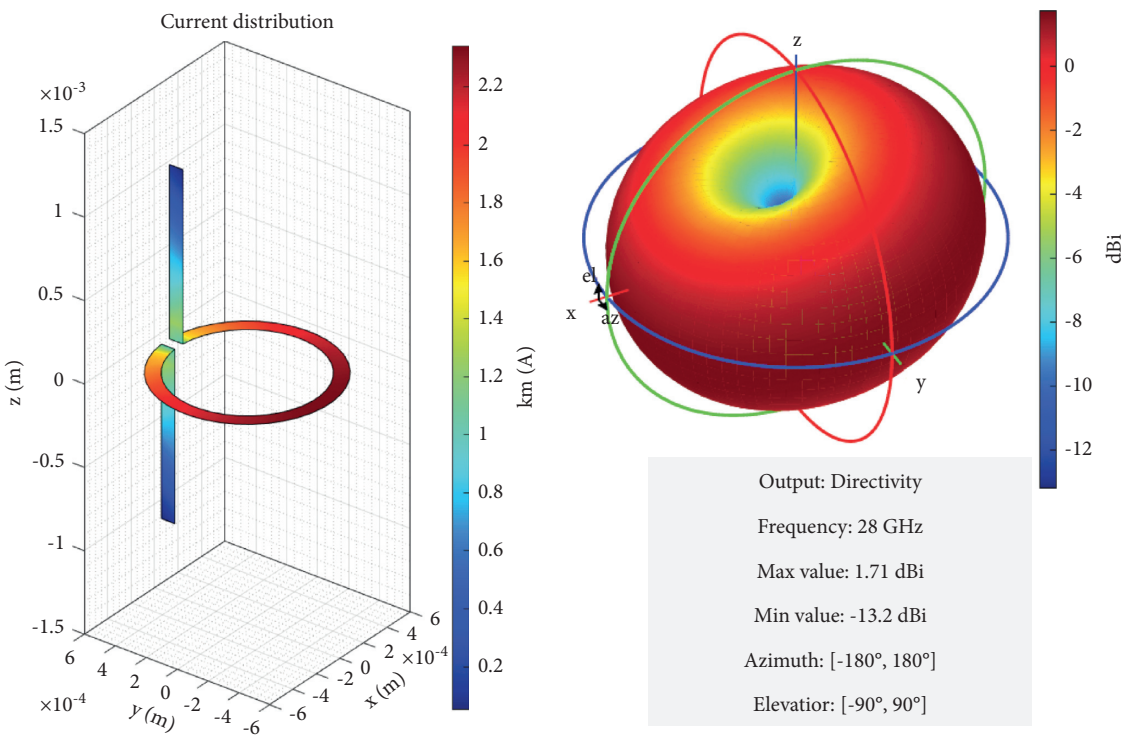
dominant rays are considered here to decrease computational time and make the algorithm less complex. In Figure 11(a), the color of two LoS rays is dark blue showing that the path loss is near to 80 dB. The other rays colored red and yellow are indicating path loss of 90 and 95 + respectively. This is due to the NLoS, reflection and diffraction of metal and brick walls. More detailed analyses of the rays and pattern of the path loss are explained in Table 3.

In Figure 11(b), the color of two LoS rays is light blue showing that path loss is near to 90 dB. The other rays colored green and yellow indicate the path loss of 95 and



(a) (b)

Figure 7: Coverage map. (a) At 28 GHz; (b) at 60 GHz.



(a) (b)

Figure 8: Antenna characteristics. (a) Current distribution; (b) antenna radiation pattern.

100 +, respectively. This is due to NLoS, reflection, and diffraction of metal and brick walls. More detailed analyses of the rays and pattern of the path loss are explained in Table 4. The difference between the path loss due to the two different transmitting frequencies is promising in a way that increasing the frequency in indoor environment results in rising the path loss by 5 dB for every 10 GHz. This is also clear from CI model calculations.

```
//defining 5 dominant rays
for ray_out = 1: 5 do
    //defining horizontal angles
    for hori = −180: 180° do
        //defining vertical angles
        for vert = −90: 90° do
            //developing rays and increment
            ray_out[i] = 1
            Develop-ray = i+1
            hori == > verti
            //finding intersection points
            while ray_out = intersect do
              for j = 1: 2 do
                //finding diffracted rays
                if difr_out = 1 then
                  ray_dfr = ray_dfr + 1
                  else
                  end
                  //finding refracted rays
                  rays = ray_refr
                  ray_refr = ray_refr + 1
                end
              end
            end
            //applying the proposed model
            PL^CI(d)[dB] = FSPL(f, d_0) + 10n. log(d/d_0) + X^CI_σSF
        end
    end
end
```

Algorithm 1: Outdoor ray-tracing algorithm pseudocode.

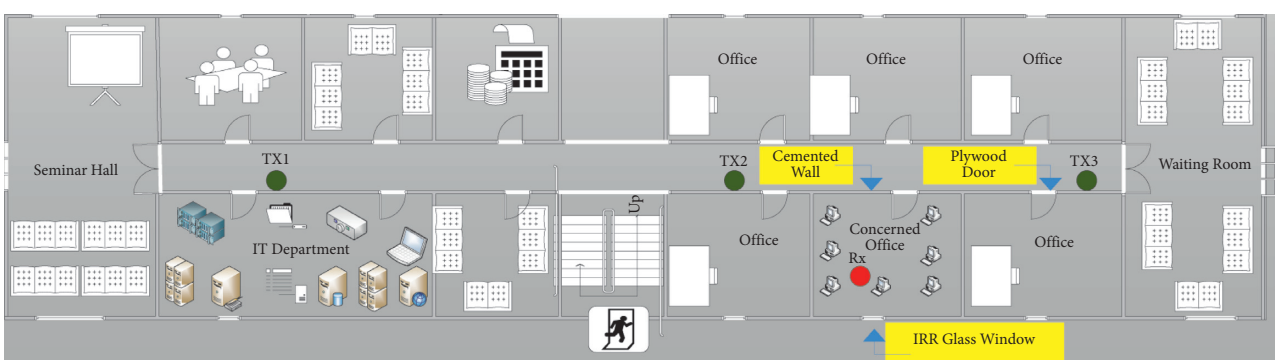
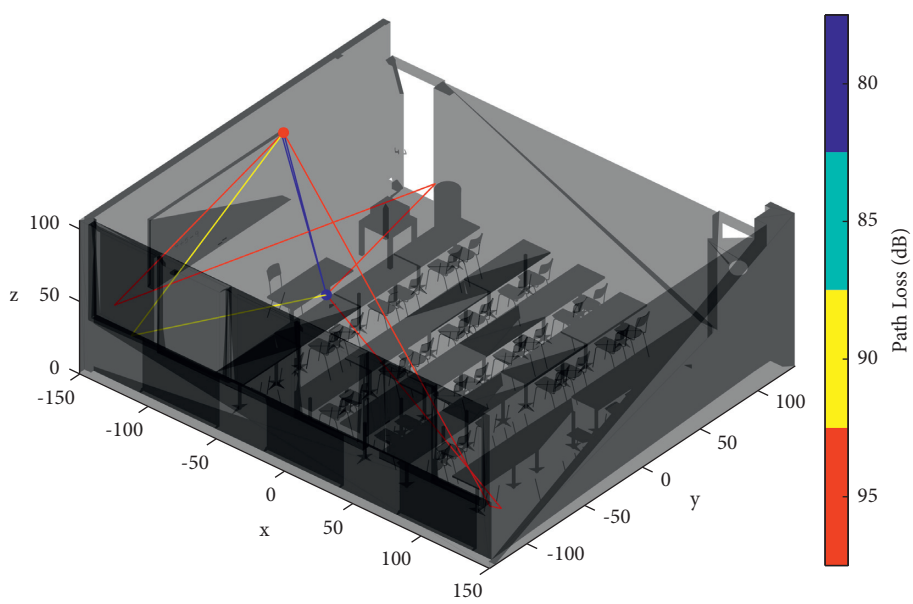


Figure 9: Floor plan for indoor simulations [5].

Figure 10: Typical classroom under consideration.

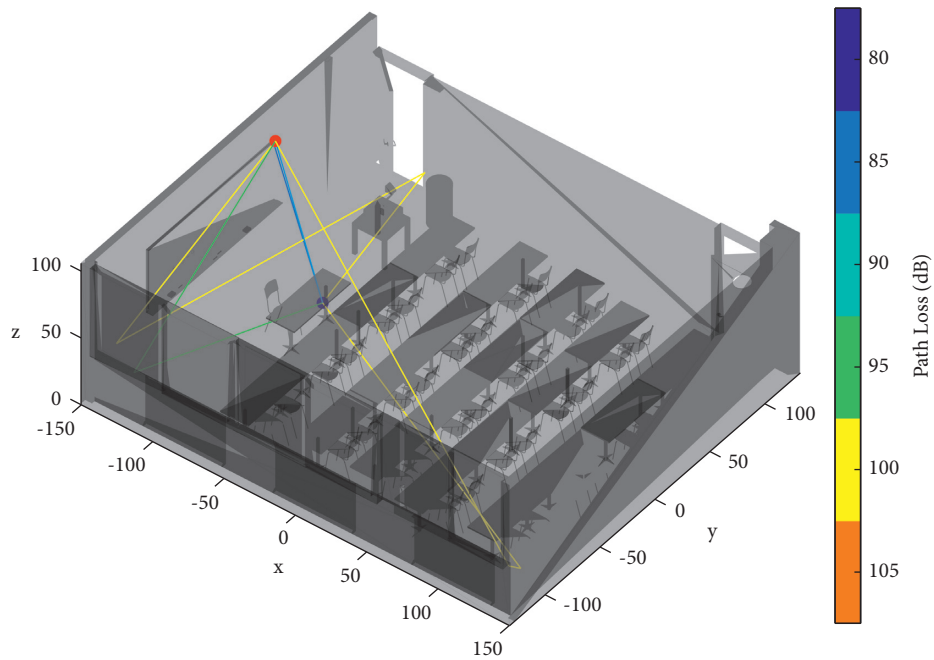
```
//defining 5 dominant rays
for ray_in = 1:5 do
    //defining horizontal angles
    for hori = −180: 180° do
        //defining vertical angles
        for vert = −90: 90° do
            //developing rays and increment
            ray_out [i] = 1
            Develop-ray = i+1
            hori == > verti
            //finding intersection points
            while ray_in = intersect do
                for j = 1: 2 do
                    //finding diffracted rays
                    if difr_in = 1 then
                        ray_dfr = ray_dfr + 1
                        else
                        end
                        //finding refracted rays
                        rays = ray_refr
                        ray_refr = ray_refr + 1
                    end
                end
            end
            //applying proposed model
            PL^CI (d) [dB] = FSPL (f, d_0) + 10n. log (d/d_0) + X^CI_σSF
        end
    end
end
```

Algorithm 2: Indoor ray-tracing algorithm pseudocode.



(a)

Figure 11: Continued.



(b)

Figure 11: Indoor intelligent ray tracing. (a) At 28 GHz; (b) at 60 GHz.

Table 3: Rays attributes for indoor ray tracing at 28 GHz.

Ray number	1	2	3	4	5
Frequency (GHz)	28				
Coordinate system	Cartesian				
Transmitter location	[−144.414; 37.593; 79.0609]				
Receiver location	[−94.898; 16.208; 0.763]				
Propagation path	LoS	LoS	NLoS	NLoS	NLoS
Reflection location	[10; 10; 0]	[−117.91; −125.3; 37.161]	[−145.41, −110.15; 37.17, 22.54; 77.54, 23.94]	[−128.74, −104.99; −127.15, 122.38; 54.27, 16.73]	[144.41, 150.21; −125.31, −122.04; 37.16, 36.32]
Propagation delay (μ s)	0.3171	1.06	0.206	1.76	2.0835
Propagation distance (m)	9.5	31.8	9.6	52.86	62.46
Angle of departure	[−23.35; −55.43]	[−80.76; −14.24]	[−157.45; −54.53]	[−84.56; −8.51]	[−29.42; −7.20]
Angle of arrival	[156.64; 55.43]	[−99.238; 14.244]	[157.45; 54.53]	[95.43; 8.51]	[−29.42; 7.20]
Number of reflections	0	1	2	2	2
Path loss (dB)	80.95	91.44	81.05	95.85	97.31
Phase shift (°)	0.0105	0.1073	5.490	2.0279	4.348

Table 4: Rays attributes for indoor ray tracing at 60 GHz.

Ray number	1	2	3	4	5
Frequency (GHz)	60				
Coordinate system	Cartesian				
Transmitter location	(−144.414; 37.593; 79.060 9)				
Receiver location	(−94.898; 16.208; 0.763)				
Propagation path	LoS	LoS	NLoS	NLoS	NLoS
Reflection location	(10; 10; 0)	(−117.91; −125.3;37.161)	(−145.41, −110.15; 37.17, 22.54; 77.54, 23.94)	(−128.74, −104.99; −127.15, 122.38; 54.27, 16.73)	(144.41, 150.21; −125.31, −122.04; 37.16, 36.32)
Propagation delay (μ s)	0.3171	1.06	0.206	1.76	2.0835
Propagation distance (m)	9.5	31.8	9.6	52.86	62.46
Angle of departure	(−23.35; −55.43)	(−80.76; −14.24)	(−157.45; −54.53)	(−84.56; −8.51)	(−29.42; −7.20)
Angle of arrival	(156.64; 55.43)	(−99.238; 14.244)	(157.45; 54.53)	(95.43; 8.51)	(−29.42; 7.20)
Number of reflections	0	1	2	2	2
Path loss (dB)	87.57	98.067	87.67	102.48	103.93
Phase shift (°)	5.401	3.82	0.096 3	6.14	4.83

6. Conclusion

In this paper, the outdoor to outdoor and indoor to indoor characteristics of the path loss with increasing the mmWave frequencies considering the network forensic with the help of the dominant path algorithm are studied. In the first approach, a conventional approach of the empirical and statistical analysis of path loss is adopted. Three different models including ABG, CI, and FI are discussed. The conclusion of the models is done that close in is better for indoor as well as outdoor scenarios. In the second approach, a software-based method is applied. Ray-tracing simulations indicate that the outdoor RSS level has a mean difference of around 8 dBm–10 dBm between 28 GHz and 60 GHz. This is due to the fact that, on the one hand, the huge gap between mmWave frequencies does not affect the power of the signal in small cell environments. On the other hand, this is the sign of low penetration, low signal diffraction, and very high free-space losses. Similarly, in indoor environment, the difference between the path loss is due to the two different transmitting frequencies which is promising in a way that increasing the frequency in indoor environment results in an increase in the path loss by 5 dB for every 10 GHz. In the future, we would like to investigate radio network forensic for artificial intelligence-aided wireless systems.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by Taif University Researchers Supporting Project number (TURSP–2020/314), Taif University, Taif, Saudi Arabia.

References

[1] S. Chen, K. Zeng, and P. Mohapatra, "Efficient data capturing for network forensics in cognitive radio networks," *IEEE/ACM Transactions on Networking*, vol. 22, no. 6, pp. 1988–2000, 2013.

[2] C. Meisch, "FCC adopts rules to facilitate next generation wireless technologies," 2021, https://www.fcc.gov/document/fcc-adopts-rules-facilitate-next-generation-wireless-technologies.

[3] U. R. Kamboh, Q. Yang, M. Qin, and S. Rauf, "Uncertainty cost analysis of heterogeneous wireless network based on loan repayment approach," in *Proceedings of the 2017 Nineth International Conference on Advanced Infocomm Technology (ICAIT)*, pp. 170–175, Chengdu, China, November 2017.

[4] J. Lloret, J. J. López, C. Turró, and S. Flores, "Afast Design Model for Indoor Radio Coverage in the2.4 GHz Wireless LAN," in *Proceedings of the Wireless CommunicationSystems, 2004, Firstst International Symposium on, Mauritius*, pp. 408–412, Mauritius, September 2004.

[5] U. R. Kamboh, U. Ullah, S. Khalid, U. Raza, C. Chakraborty, and F. Al-Turjman, "Path Loss Modelling at 60 GHz mmWave Based on Cognitive 3D ray Tracing Algorithm in 5G," *Peer-To-Peer Networking and Applications*, vol. 14, no. 8, 2021.

[6] W. Khawaja, O. Ozdemir, Y. Yapici, F. Erden, M. Ezuma, and I. Guvenc, "Coverage Enhancement for NLOS Mmwave Links Using Passive Reflectors," 2019, https://arxiv.org/pdf/1905.04794.pdf.

[7] W. Khawaja, O. Ozdemir, Y. Yapici, I. Guvenc, and Y. Kakishima, "Coverage enhancement for mmWave communications using passive reflectors," in *Proceedings of the IEEE Global Symp. Millimeter Waves (GSMM)*, pp. 1–6, Boulder, CO, USA, May 2018.

[8] M. Kim, J. Liang, J. Lee, J. Park, B. Park, and H. K. Chung, "Investigating the effect of antenna beamwidth on millimeter-wave channel characterization," in *Proceedings of the IEEE URSI Asia-Pacific Radio Sci. Conf. (URSI AP-RASC)*, pp. 1–4, Seoul, Republic of Korea, August 2016.

[9] D. M. Tuan, Y. Cheon, Y. Aoki, and Y. Kim, "Performance comparison of millimeter-wave communications system with

different antenna beamwidth," in *Proceedings of the IEEE European Conf. Antennas Propag. (EuCAP)*, pp. 1–5, Davos, Switzerland, April 2016.

[10] H. Cheng, S. Ma, and H. Lee, "CNN-based mmWave path loss modeling for fixed wireless access in suburban scenarios," *IEEE Antennas and Wireless Propagation Letters*, vol. 19, no. 10, pp. 1694–1698, 2020.

[11] N. Kuno, W. Yamada, M. Sasaki, and Y. Takatori, "Convolution neural network for prediction method of path loss characteristics considering diffraction and reflection in an open-square environment," in *Proceedings of the . URSI Asia-Pacifc Radio Sci. Conf. (AP-RASC)*, pp. 1–3, New Delhi, India, March 2019.

[12] E. I. Adegoke, E. Kampert, and M. D. Higgins, "Empirical indoor path loss models at 3.5GHz for 5G communications network planning," in *Proceedings of the . Int.Conf. UK-China Emerg. Technol. (UCET)*, pp. 1–4, Glasgow, UK, August 2020.

[13] M. C. Lawton and J. P. McGeehan, "The application of a deterministic ray launching algorithm for the prediction of radio channel characteristics in small-cell environments," *IEEE Transactions on Vehicular Technology*, vol. 43, no. 4, pp. 955–969, 1994.

[14] A. Al-Samman, T. Rahman, M. Hindia, A. Daho, and E. Hana, "Path loss model for outdoor parking environments at 28 GHz and 38 GHz for 5G wireless networks," *Symmetry*, vol. 10, no. 12, p. 672, 2018.

[15] S. Li, Y. Liu, L. Lin, D. Sun, S. Yang, and X. Sun, "Simulation and modeling of millimeter-wave channel at GHz in indoor environment for 5G wireless communication system," in *Proceedings of the 2018 IEEE International Conference on Computational Electromagnetics (ICCEM)*, pp. 1–3, March 2018.

[16] G. R. Maccartney, T. S. Rappaport, M. K. Samimi, and S. Sun, "Millimeter-wave omnidirectional path loss data for small cell 5G channel modeling," *IEEE Access*, vol. 3, pp. 1573–1580, 2015.

[17] A. M. Al-Samman, M. H. Azmi, Y. A. Al-Gumaei et al., "Millimeter wave propagation measurements and characteristics for 5G system," *Applied Sciences*, vol. 10, no. 1, p. 335, 2020.

[18] A. M. Al-Samman, T. A. Rahman, T. Al-Hadhrami et al., "Comparative study of indoor propagation model below and above 6 GHz for 5G wireless networks," *Electronics*, vol. 8, no. 1, p. 44, 2019.

[19] I. Cuinas, M. G. Sanchez, A. Feys, W. Debaenst, and J. Verhaevert, "Indoor path loss variations with frequency and visibility conditions at 3.5 GHz band," in *Proceedings of the 2019 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*, pp. 2069-2070, Atlanta, GA, USA, July 2019.

[20] N. Khalid, N. A. Abbasi, and O. B. Akan, "Statistical characterization and analysis of low-THz communication channel for 5G Internet of Things," *Nano Commun. Netw.*vol. 22, Article ID 100258, 2019.

[21] F. Qamar, M. N. Hindia, K. Dimyati et al., "Investigation of future 5G-IoT millimeter-wave network performance at 38 GHz for urban microcell outdoor environment," *Electronics*, vol. 8, no. 5, p. 495, 2019.

[22] Y. Zhou, X. Sun, P. Zhang, H. Wang, Z. Hu, and H. Wang, "Multi-frequency millimeter-wave large-scale path loss characterization for indoor environment," in *Proceedings of the 2019 International Symposium on Antennas and Propagation (ISAP)*, pp. 1–3, Xi'an, China, October 2019.

[23] J. Wen, Y. Zhang, G. Yang, Z. He, and W. Zhang, "Path loss prediction based on machine learning methods for aircraft cabin environments," *IEEE Access*, vol. 7, Article ID 159251, 2019.

[24] H. Cheng, S. Ma, H. Lee, and M. Cho, "Millimeter wave path loss modeling for 5G communications using deep learning with dilated convolution and attention," *IEEE Access*, vol. 9, Article ID 62867, 2021.

[25] N. Kuno and Y. Takatori, "Prediction method by deep-learning for path loss characteristics in an open-square environment," *In Proc. Int. Symp. Antennas Propag. (ISAP)*, pp. 1-2, 2018.

[26] A.-Y. Hsiao, C.-F. Yang, T.-S. Wang, I. Lin, and W.-J. Liao, "Ray tracing simulations for millimeter wave propagation in 5G wireless communications," in *Proceedings of the 2017 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, pp. 1901-1902, San Diego, CA, USA, July 2017.

[27] C. Ge, Y. Zhang, and X. Jiang, "Simulation and analysis of 28 GHz millimeter-wave propagation characteristics in typical residential house environment," in *Proceedings of the 2018 11th UK-Europe-China Workshop on Millimeter Waves and Terahertz Technologies (UCMMT)*, vol. 1, pp. 1–3, HangZhou, China, September 2018.

[28] K. Kitao, A. Benjebbour, T. Imai, Y. Kishiyama, M. Inomata, and Y. Okumura, "5G system evaluation tool," in *Proceedings of the 2018 IEEE International Workshop on Electromagnetics: Applications and Student Innovation Competition (iWEM)*, pp. 1-2, Nagoya, Japan, August 2018.

[29] F. Hossain, T. Geok, T. Rahman et al., "An efficient 3-D ray tracing method: prediction of indoor radio propagation at 28 GHz in 5G network," *Electronics*, vol. 8, no. 3, p. 286, 2019.

[30] C.-F. Chang-Fa Yang, B.-C. Boau-Cheng Wu, and C.-J. Chuen-Jyi Ko, "A ray-tracing method for modeling indoor wave propagation and penetration," *IEEE Transactions on Antennas and Propagation*, vol. 46, no. 6, pp. 907–919, 1998.

[31] A. E. Shaikh, F. Majeed, M. Zeeshan, T. Rabbani, and I. Sheikh, "Efficient implementation of deterministic 3-D ray tracing model to predict propagation losses in indoor environments," in *Proceedings of the 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1208–1212, Lisboa, Portugal, September 2002.

[32] U. Ullah, U. R. Kamboh, F. Hossain, and M. Danish, "Outdoor-to-Indoor and indoor-to-indoor propagation path loss modeling using smart 3D ray tracing algorithm at 28 GHz mmWave," *Arabian Journal for Science and Engineering*, vol. 45, pp. 1–10, 2020.

[33] D. Murugesan and T. R. Rao, "Indoor corridor radio propagation characteristics at 60 GHz for wireless communications," in *Proceedings of the 2012 3rd International Conference on Computing, Communication and Networking Technologies (ICCCNT'12)*, pp. 1–5, Coimbatore, India, July 2012.

[34] A. Karstensen, W. Fan, I. Carton, and G. F. Pedersen, "Comparison of ray tracing simulations and channel measurements at mmwave bands for indoor scenarios," in *Proceedings of the 2016 10th European Conference on Antennas and Propagation (EuCAP)*, pp. 1–5, Davos, Switzerland, April 2016.

[35] S. Helhel, "Comparison of 900 and 1800 MHz indoor propagation deterioration," *IEEE Transactions on Antennas and Propagation*, vol. 54, no. 12, pp. 3921–3924, 2006.

[36] J. J. P. C. Rodrigues, S. Jabbar, M. Abdallah, C. Verikoukis, and M. Guizani, "Future communication trends toward

internet of things services and applications," *IEEE Wireless Communications*, vol. 26, no. 6, pp. 6–8, 2019.

[37] F. A. Rodríguez-Corbo, L. Azpilicueta, M. Celaya-Echarri et al., "Deterministic 3D ray-launching millimeter wave channel characterization for vehicular communications in urban environments," *Sensors*, vol. 20, no. 18, p. 5284, 2020.

[38] G. R. Maccartney, T. S. Rappaport, S. Sun, and S. Deng, "Indoor office wideband millimeter-wave propagation measurements and channel models at 28 and 73 GHz for ultra-dense 5G wireless networks," *IEEE Access*, vol. 3, pp. 2388–2424, 2015.

[39] S. Sun, T. A. Thomas, T. S. Rappaport, H. Nguyen, I. Z. Kovacs, and I. Rodriguez, "Path Loss, Shadow Fading, and Line-Of-Sight Probability Models for 5G Urban Macro-Cellular Scenarios," in *Proceedings of the 2015 IEEE Globecom Workshops (GC Wkshps)*, San Diego, CA, USA, December 2015.

[40] A. M. Al-Samman, T. A. Rahman, M. H. Azmi, M. N. Hindia, I. Khan, and E. Hanafi, "Statistical modelling and characterization of experimental mm-wave indoor channels for future 5G wireless communication networks," *PLoS One*, vol. 11, no. 9, Article ID e0163034, 2016.

[41] Z. Yun and M. F. Iskander, "Ray tracing for radio propagation modeling: principles and applications," *IEEE Access*, vol. 3, pp. 1089–1100, 2015.

[42] M. F. Iskander and Z. Zhengqing Yun, "Propagation prediction models for wireless communication systems," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 3, pp. 662–673, 2002.

[43] G. Durgin, N. Patwari, and T. S. Rappaport, "An advanced 3D ray launching method for wireless propagation prediction," in *Proceedings of the 1997 IEEE 47th Vehicular Technology Conference. Technology in Motion*, vol. 2, pp. 785–789, Phoenix, AZ, USA, May 1997.

[44] D. He, B. Ai, K. Guan, L. Wang, Z. Zhong, and T. Kurner, "The design and applications of high-performance ray-tracing simulation platform for 5G and beyond wireless communications: a tutorial," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 10–27, 2019.

[45] V. Degli-Esposti, D. Guiducci, A. de'Marsi, P. Azzi, and F. Fuschini, "An advanced field prediction model including diffuse scattering," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 7, pp. 1717–1728, 2004.

[46] R. Luebbers, "Finite conductivity uniform GTD versus knife edge diffraction in prediction of propagation path loss," *IEEE Transactions on Antennas and Propagation*, vol. 32, no. 1, pp. 70–76, 1984.

[47] O. Franek, S. Zhang, K. Olesen, P. C. F. Eggers, C. Byskov, and G. F. Pedersen, "Numerical modeling of ultrawideband propagation along a wind turbine blade," *IEEE Transactions on Antennas and Propagation*, vol. 66, no. 12, pp. 6570–6579, 2018.

[48] M. S. L. Mocker, M. Schiller, R. Brem et al., "Combination of a full-wave method and ray tracing for radiation pattern simulations of antennas on vehicle roofs," in *Proceedings of the Nineth Eur. Conf. Antennas Propag. (EuCAP)*, pp. 1–5, Lisbon, Portugal, April 2015.

[49] S. Jabbar, H. Lloyd, M. Hammoudeh, B. Adebisi, and U. Raza, "Blockchain-enabled supply chain: analysis, challenges, and future directions," *Multimedia Systems*, vol. 27, no. 4, pp. 787–806, 2021.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 4494447, 13 pages
https://doi.org/10.1155/2021/4494447

Research Article

Digital Forensics Use Case for Glaucoma Detection Using Transfer Learning Based on Deep Convolutional Neural Networks

Jahanzaib Latif,[1] Shanshan Tu,[1] Chuangbai Xiao,[1] Sadaqat Ur Rehman,[2] Mazhar Sadiq,[3] and Muhammad Farhan[3]

[1]*Engineering Research Center of Intelligent Perception and Autonomous Control, Faculty of Information Technology, Beijing University of Technology, 100124 Beijing, China*
[2]*Department of Computer Science, Namal Institute, Mianwali 42250, Pakistan*
[3]*Department of Computer Science, COMSATS University Islamabad, Sahiwal Campus, Islamabad 57000, Pakistan*

Correspondence should be addressed to Shanshan Tu; sstu@bjut.edu.cn

Received 25 August 2021; Accepted 6 November 2021; Published 29 November 2021

Academic Editor: Farhan Ullah

In parallel with the development of various emerging fields such as computer vision and related technologies, e.g., iris identification and glaucoma detection, criminals are developing their methods. It is the foremost reason for the blindness of human beings that affects the eye's optic nerve. Fundus photography is carried out to examine this eye disease. Medical experts evaluate fundus photographs, which is a time-consuming visual inspection. Most current systems for automated glaucoma detection in fundus images rely on segmentation-based features nuanced by the underlying segmentation methods. Convolutional neural networks (CNNs) are powerful tools for solving image classification tasks, as they can learn highly discriminative features from raw pixel intensities. However, their applicability to medical image analysis is limited by the nonavailability of large sets of annotated data required for training. In this work, we aim to accelerate this process using a computer-aided diagnosis of this severe disease with the help of transfer learning based on deep convolutional neural networks. We have suggested the Inception V-3 approach for image classification based on convolution neural networks. Our developed model has the potential to address this CNN model's problem of classification accuracy, and with imaging data, our proposed method outperforms recent state-of-the-art approaches. The case study for digital forensics is an essential component of emerging technologies, and hence glaucoma detection plays a vital role in it.

1. Introduction

Glaucoma is a set of eye situations that hurt the optic nerve, the health of which is vigorous for sound vision. This harm is often produced by oddly high pressure in the eyes. Glaucoma is the leading reason of blindness, and it may occur at any age but is more common in older people having more than 60 years of age. Several types of glaucoma have no cautioning signs, but few warning signs indicate the possibility of this disease. If this disease is recognized earlier, vision loss can be slowed or prevented completely [1].

Deep learning is a modern technique with much promise in the area of ophthalmology. Deep learning tools have been used to analyze multimedia images, optical coherence tomography, and visual fields, among other diagnostic modalities. These methods effectively determine cataracts, glaucoma, age-related macular degeneration, and diabetic retinopathy, among other diseases. Deep learning methods are quickly emerging and can increasingly be used in ophthalmic treatment [2].

Convolutional neural networks (CNNs) have been a popular method for solving computer vision problems, including image recognition and semantic segmentation. These methods are robust enough to learn deep image properties that are usually neglected but are essential for the target challenge, avoiding intermediate steps including segmentation or function design and selection [3]. However, this benefit comes at the expense of learning from incredibly

broad, annotated training sets [4]. Ting et al. [5] released a research report on eye disorders like diabetic retinopathy, glaucoma, and age-related macular degeneration. In their research, they compiled a list of papers published between 2016 and 2018. They collected a collection of publications that used fundus and optical coherence tomography pictures and TL methods. They did not include recent publications that used TL approaches, and they did not have eye cataract disease diagnosis in their research scope.

Similarly, Hogarty et al. [6] reviewed existing state publications in ophthalmology that used AI, but their emphasis missed systematic AI methodologies. Hagiwara et al. [7] looked at an essay on utilizing fundus pictures to detect glaucoma efficiently. They concentrated on computer-assisted systems and optical disc segmentation systems. A range of research works used DL and TL approaches to diagnose glaucoma.

The most popular approach to studying the human eye in ophthalmology is to take and evaluate an eye-fundus image. A fundus camera is used to take a snapshot of the eye's surroundings via the pupil during this type of eye test. Visual analysis is a popular method of examining these images. In medical screening, this procedure will take many hours in front of a computer screen. We aim to speed up the diagnosis by utilizing computational algorithms to examine the photos and find and highlight the most relevant information. We would want to be able to diagnose anomalies and illnesses without the need for human involvement. The popular image processing methods built and validated utilizing low-resolution images have demonstrated disadvantages in clinical usage due to fundus images' increasingly growing spatial resolution. A new generation of approaches must be created for this reason. These techniques must be able to work on high-resolution images while being computationally simple. We present a novel vessel segmentation approach with low computational criteria and a publicly accessible high-resolution fundus database with manually developed gold standards for testing retinal structure segmentation methods.

Retinal vessel segmentation is a complicated process that has drawn researchers from all around the world for years. Many different algorithms were used during this period. Unsupervised and supervised approaches are the two critical types of segmentation algorithms. Unsupervised methods use heuristics to identify boats, while supervised methods immediately learn a criteria scheme using prelabelled data as the gold standard. Since supervised methods need a broad training set for each camera configuration, we concentrate on heuristic methods. On the other hand, heuristic techniques necessitate a series of criteria tailored to the camera configuration. As a result, they are far less reliant on the test dataset during production—a more in-depth look at the segmentation and other algorithms used in retinal image processing.

Model-based approaches are another popular segmentation technique. Active contour-based techniques, stage sets, and other related approaches are the most well known and frequently employed. Early snake-based algorithms begin with a rough object contour that is iteratively optimized by multiple powers. The forces achieve their equilibrium on the object boundaries in the ideal situation. These approaches are vulnerable to parameterization, and they may encounter difficulties when segmenting thick and thin vessels simultaneously. As a result, the consumer must manually set and optimize the parameters.

The authors of [8] proposed an automated solution and classification of this chronic condition using the Cup to Disk Ratio in another report (CDR). Consequently, we may infer that classifying the image's retina can be complicated if we are not going for the retina areas after extracting the functionality. After identifying the picture features, we do not need any experience to pick the discriminative features. Consequently, we may infer that identifying glaucoma using CADx devices for ophthalmologists is a challenging job.

The matched-filter method was one of the first and most popular methods for fundus photographs. It enhances vessels by applying predefined vessel profiles of various sizes and orientations to the picture. To obtain vessel segmentation, early implementations of these methods depended on an essential thresholding phase. These techniques were occasionally mixed with other interventions. The matched filters yield high-quality performance, but their only downside is that they need vessel profiles and large-region comparisons for each pixel in the image, which takes a long time to compute. The consistency and scale of the used vessel profile database have a direct effect on the segmentation performance. It is subject to race, camera configuration, and even eye or vascular disorders, restricting its applicability. Some algorithms are designed to segment only one or more items identified by a consumer or preprocessed. These approaches typically do not look at the whole picture but rather the area around the already segmented regions. Based on parallels and other parameters, region-growing methods attempt to increase the segmented area of nearby pixels. These techniques are most accessible, but they could have problems in some parts of the picture where the vessels have more inadequate contrast than the underlying tissues, such as vessel ends or small vessels. In this situation, the expanding zone can segment vast unwelcomed regions. In these cases, vessel monitoring algorithms are more reliable. They monitor the provided vessels by looking for a vessel-like structure in the already segmented area. These algorithms are robust at identifying vessel endings, but they may fail at bifurcations and vessel crossings, where the local structures no longer resemble traditional vessels.

Categorization is a process in which things (objects, ideas, or people) are classified according to their similarities or standard criteria (classes, types, and index). It allows people to simplify and explain their understanding of the essence of the things, activities, and ideas around them. Humans do categorize it as something that many animals do: "do the utmost with the right things." Categorization depends on characteristics ranging from non-member to party representative. Categorization is essential in the study, assessment, inference, decision-making, vocabulary, and other ways the animals communicate with their ecosystems [9].

ANN cannot fulfill general roles. Until patterns as entrants are presented, they are thoroughly eligible with

datasets named testing datasets. These emerging trends can be an accurate definition or just a prognosis after a training session. ANN may detect trends in real-system results, computer programs, and physical models, or other sources. The design of the appropriate algorithm, called hidden layers for answers, can handle as much of the details. ANN concentrates on our brain [10].

The machine learning subset depends on input and output rates, which are representations, variables, functions, and the basics [11]. It differentiates between meanings from the lower to the higher standard. It is designed to contain data, including text, photographs, and audio. It is an ANN sector with complex perceptron secret layers. Furthermore, backspreading is used in ANN for defining the mistake that the actual output would take. There are two weights and the input meaning for both coming ties. The production of each relation is a function of the summed unit values [12]. The abbreviations and acronyms used in this work are shown in Table 1.

1.1. Motivation. As already stated, DL and TL techniques have several advantages, and in recent years, many researchers have applied these methods. Overall, relatively few analysis articles are conducted in clinical databases that concurrently discuss all glaucoma detection forms. This study is also an essential and needed work towards the disease diagnosis of glaucoma detection.

1.2. Digital Forensics and Computer Vision. Nowadays, digital content is widely available and easily redistributed, whether legally or illegally [13]. For instance, after sharing photographs on the Internet, other web users can alter them and then publish modified versions, resulting in identical images [14]. The presence of near duplicates has a significant impact on the performance of search engines. Computer vision is the process of extracting, analyzing, and understanding usable information from digital images automatically. Computer vision's primary use is picture understanding. Image comprehension encompasses various tasks, including feature extraction, object identification, object recognition, image cleaning, and image transformation. CNNs have consistently performed well in various computer vision tasks that previously needed a visual inspection [15, 16]. As a result of these astounding results, academics have begun to use CNNs to solve picture forensic challenges. However, image classification approaches in computer vision typically rely on visually discernible cues. On the other hand, forensic solutions rely on imperceptible traces that are often highly delicate and hidden in the excellent details of the image under investigation. As a result, training a CNN to tackle a forensic assignment requires extra caution, as standard processing procedures can significantly impair forensic traces.

1.3. Organization of the Paper. This paper is organized as follows: Section 1 provides the introduction of the research. Section 2 provides a literature review of related work. We have provided concise and clear past work related to our study. The methodology is discussed in Section 3 with the help of the proposed solutions and implementation of research. Section 4 is composed of a discussion and results section. In the discussion section, we discussed our overall research with results tables and performance graphs. The conclusion section concluded our investigation with conclusory remarks and provided some valuable hints for future research directions.

Table 1: Abbreviations and acronyms used throughout the manuscript.

Abbreviations	Full forms
CNN	Convolutional neural network
DCNNs	Deep convolutional neural networks
AI	Artificial intelligence
DL	Deep learning
TL	Transfer learning
CDR	Cup to disc ratio
CADx	Computer-aided diagnostics
ANN	Artificial neural network
RF	Random forests
MAs	Microaneurysms
DME	Diabetic macular edema

2. Literature Review

In many applications, the advances achieved in information technology and the demands of image processing are spreading. There are many techniques in image processing, and image classification is also its subject. Researchers have used different methods for image classification, but there is no specific solution for image classification. The target of image classification is the foreground of an image, as we can say the problem is the background of an image. Classification is the way to separate a picture into various sections or stages that may take an image from our goal region [17]. Thus, when we take our attention from it, we avoid classification at that stage. The usage of the conditional random field in sketches and classifications is to boost the state-of-the-art. The selective object recognition search method used in a single image in multilocation has provided the OverFeat system for location, identification, and classifying boundaries in convolution networks [18] using the skin-color chart for the color analysis of the face position by area grading. The FaceNet was used to learn a mapping from face pictures to calculate facial similarities.

Ophthalmologists can conduct the eye screening procedure more effectively, and a more decisive outcome can be achieved, resulting in a more cost-effective approach for patients. We will save time and money by taking an in-depth research approach to glaucoma research and study. Since ophthalmologists need to distinguish between the glaucoma retinal picture and the healthy image, the computer-aided diagnostics (CADx) method should be introduced in clinics and hospitals to discern between the glaucoma retinal image and the healthy image.

Decision tree classifier (C4.5), Naive Bayes classifier, and random forest are examples of the current machine learning

methods. These tests are insufficient to diagnose glaucoma disorder with greater precision. The author has developed a CNN architecture focused on the deep learning methodology in this report. The authors determined which network is the best for identification after analyzing performance, precision, recall, and f-score. Deep learning methods may be used to distinguish between nonglaucoma and glaucoma patterns for diagnosis decisions, resulting in higher precision.

Reference [19] published a comprehensive analysis of automated approaches to diabetic eye disease identification from several viewpoints, including (i) accessible databases, (ii) picture preprocessing methods, (iii) deep learning models, and (iv) output assessment indicators. As seen in Figure 1, the survey offered a detailed review of eye disease identification strategies, including state-of-the-art field approaches.

Instead of utilizing the segmentation technique, image characteristics across the retina area may diagnose glaucoma eye disease. Consequently, there is a need to correctly recognize cup and disc boundaries using a procedure that suggests the disorder induces morphological adjustments in the retina. Without requiring deep domain awareness in image processing, the essential features may be extracted in detail automatically.

Acharya et al. [13] used a sequence of radon, discrete wavelet, and discrete cosine transforms to derive features from retinal images. Messidor data (1200 fundus photos divided into 400 sets) discovered a decision tree graded DME with the best results. The package is split into four 100-image subsets, each with a corresponding excel file corresponding to the diagnosis.

Yu et al. [20] concentrated on offline classifiers to increase numerical performance when classifying pixels, while Chudzik et al. [21] highlighted the significance of layer freezing and transfer learning. CNNs can acquire lower-level functionality from public databases using transfer learning, which compensates for data scarcity. Hatanaka et al. [22] used a two-step DCNN to detect MAs while still filtering false positives. Natural language processing was used by Dai et al. [23] to compensate for videos that were poorly monitored (NLP). During preparation, the multisieving system was used. CNN connects the supervised evidence in clinical reports to the positions of lesions in retinal photographs. The CNN then employs these newly acquired comparisons to detect lesions in research photos without the assistance of clinical documents.

The authors of [24] wrote a review paper on fundus image classification methods to detect glaucoma using multiple ML approaches and techniques. The authors of [25] provided a systematic review of the articles using CNNs for medical image analysis, published in the medical literature before 2019. Articles were screened based on the following items: type of image analysis approach (detection or classification), algorithm architecture, the dataset used, training phase, test, comparison method (with specialists or other), results (accuracy, sensibility, and specificity), and conclusion. The writers of [26] have used five open-access databases, which are open to the public: retinal fundus images; OCT images; skin lesion images, and pediatric and adult chest X-ray images, to feed them into the search system for neural architecture, hosted by Google Cloud autoML, which has automatically built a deep learning architecture to identify standard information. The authors of [27] used medical small-sized image data for classification using the RF algorithm and deep ensembles. The authors of [28] worked on a deep learning convolutional network based on Keras and Tensor Flow using python for image classification. They used several different medical images as a dataset to diagnose eye diseases, which contain four types of diseases, diabetic retinopathy, glaucoma, myopia, and normal. CNN, VGG16, and InceptionV3 neural network structures were compared singly and together using a bagging ensemble to diagnose eye diseases. All experiments were applied, and the result was obtained. The authors of [29] worked on eye disease classification using backpropagation with parabola learning rate. They achieved 89.83% classification accuracy in their experiment. Table 2 summarizes the related works.

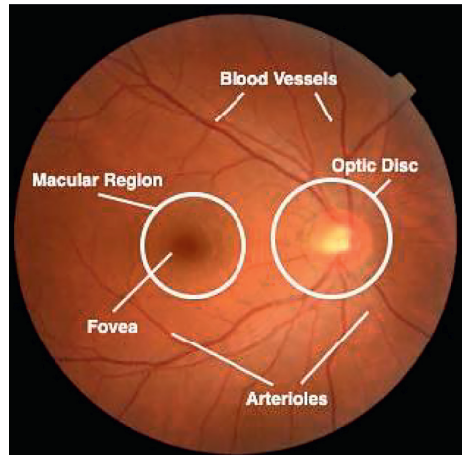


Figure 1: Anatomical structures of the retina for describing health and unhealthy eye [19].

3. Problem Formulation

State-of-the-art literature has shown glaucoma detection using various datasets of fundus images. Various authors have used different datasets for multiple purposes, but a comprehensive collection of fundus images for classifying healthy and unhealthy eyes is missing. After selecting these images, applying deep learning models as a transfer learning factor can give us good results in terms of classification and reasonable statistical measures. The critical contributions in the literature and their approaches are summarized in Table 2.

4. Materials and Methods

We looked at many machine learning methods used by various researchers on fundus photos in the literature. C 4.5, Naive Bayes classifier, and random forest are examples of the current machine learning methods. These tests are insufficient to diagnose glaucoma disorder with greater precision.

Table 2: Summary of related work on image classification using ML and DL approaches publishing year, primary contribution attributes, and approach name.

Ref no.	Key contribution	Approach
Sarki et al. [19]	Fundus images are used to detect eye disease of the diabetic patient using a deep learning approach	Deep learning models
Hameed et al. [30]	DME classification	Decision tree
Yu and Xiao 2017 [20]	Retinopathy of diabetic patients for detecting exudate	CNN
Chudzik et al. [21]	Interleaved freezing of deep learning method for microaneurysm detection	Transfer learning and layer freezing
Hatanaka et al. [22]	Retinal images are used to automatic microaneurysms detect using deep convolution neural network	DCNN
Dai et al. [23]	• Multi-sieving deep learning is used to detect retinal microaneurysm for clinical report	CNN
Saba et al. [24]	Glaucoma detection using fundus image	A mixture of ML methods
Fourcade et al. [25]	Image analysis for medical purposes using deep learning	CNN
Faes et al. [26]	Medical image classification using deep learning model design	Google cloud AutoML
Katzmann et al. [27]	Medical small-sized image data classification using RF algorithm	Random forest classifiers and deep ensembles
Smaida and Yaroshchak [28]	Deep learning convolutional network based on keras and tensor flow using python for image classification	DCNN
Hameed et al. [29]	Eye diseases classification	Back propagation with parabola learning rate

In this analysis, we developed a CNN for glaucoma disease diagnosis utilizing a deep learning framework. We determined which network is better for identification after examining, depending on accuracy, as shown in Figure 2.

Convolution layers often might not be connected to input data, like raw pixel value, but the output of the other layers. Convolution layer initialization allows for the hierarchical redirection of the results. For filters directly operating on raw pixel value, regarding the weighted value like a line characteristic and those filters attempting to extract weaves that combine a function, these filters are assisted by first line layer production, which became multiple lines to display the shape. It can begin until the deeper layers delete faces such as houses, animals, or people. It is precise how the breadth of this network can be improved if we choose to see high and low functionality abstracted by ANN [31]. Moreover, we practice this in a timely fashion. Thus, we get the message that the layers closest to the input layers are more pertinent to concert components than chairs and tables, such as lines and types, and the thresholds are relative to the output layers.

The core building blocks in convolutional neural networks are classified as revolutionary layers. The heart of the Convolutional neural network is the convolutional layer, which gives the network its name. This sheet carries out a convolution mechanism. CNN cannot be comprehended due to the single filter. Besides, several structures are being examined alongside a single collection of results. The current layers aim to keep the exact dimensions as the input. Local features tend to be gained through convolutional layers [32]. Compared to models 32 or 512 for collecting features from an entity or various ways of seeing and observing guidance, those layers have many essential methods of monitoring input data. This versatility allows for flexibility in the training specifics, not only lines but also specific lines.

There are many channels for colorful images, typically for each color channel like red, green, and blue. From a data point of view, it means that if the input of a single image to a model is given, it is three images. The number of channels, also called the width, must also be close to the stream. When an image has 3 channels for input, 3 channels should be used in a filter applied to that image, e.g., a depth of 3. A 3-3 filter can be $3 \times 3 \times 3 \times 3$ or 3 [3,3,3] sides, second columns, and third columns. The filter is used for a point product process to achieve a single input and filter depth value. Since 32 and 32 filters are in a convolution sheet, they are twin and tri-dimensional, with correct filter weights for all three channels [27]. However, each filter gives upshots of 32 of the 32-convergence filters for the 32 feature maps generated in a single function chart.

Resizing is a method in which the number of pixels is decreased or expanded. The scaling of images is referred to as resizing. When an image is resized, a new picture may be produced with a higher or lower number of pixels. There are two different types of resizing: zooming and shrinking. The process by which an image is broadened and visible in detail is called zooming. The image should be scaled in such a way that the image's precision is not affected. Reducing the image means deleting pixels from columns and lines that have little effect on the image details. By running a python script via CMD with the dataset route, we resized Inception V-3 input size: [224 $*$ 224] and VGG16 input size: [224 $*$ 224]. We divided the 1563 photos into eleven datasets, each labeled with a folder name and held the same picture in each group.

We have used this technique in the Inception of V-3 and VGG16 beginnings. It allows the data availability of the training model to be explicitly increased. It expends the dataset of training by using the same dataset to create modified picture versions. In a deep learning neural network, Keras provides this capability to fit models through data enhancement. Training deep neural networks on

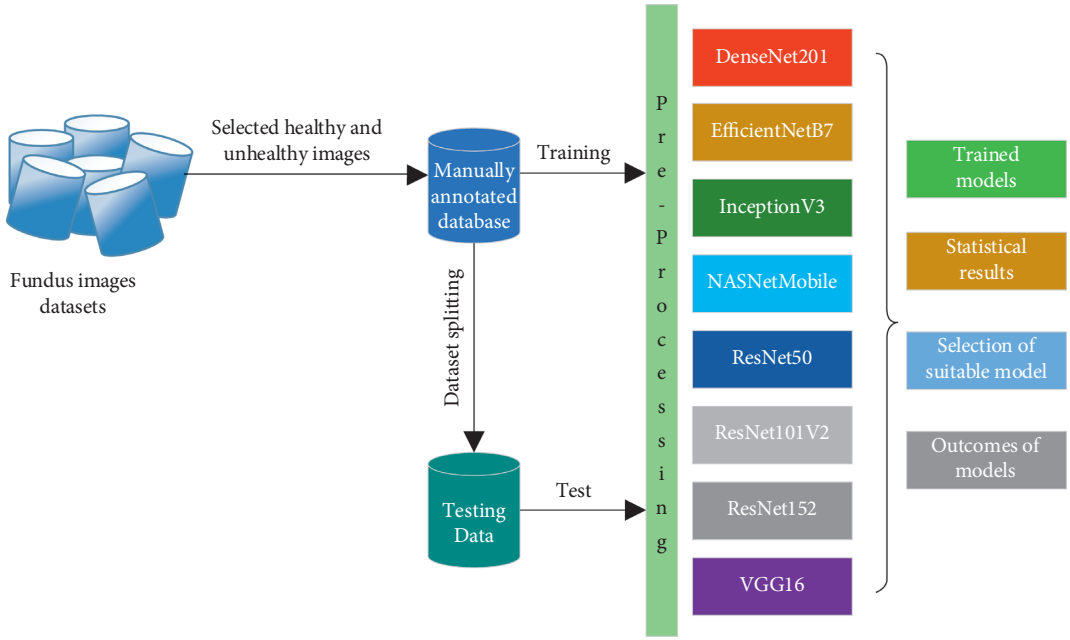


Figure 2: A framework for glaucoma detection using transfer learning.

additional data can best provide skilled models. The variations of the pictures can be created with this technology, improving the possibility of fitting in models to simplify what they have learned about the new pics. Kera's profound learning library can fit models with image data generator using the increase of image data. This technique trains a large number of neural networks by padding, cutting, and horizontal flipping. In 2D images, most of the data increase is used. This technique can be used to supply transformed image forms. Random samples are extracted from the dataset, labeled as healthy and unhealthy, as shown in Figure 3.

5. Results and Discussion

The next trait is a plane sheet; the neuron's weight is the same in the plane. This map uses the sigmoid functionality; in short, the activation function can be named CNN. In the mapping plane, neurons exchange the weights in the same plane. It can be used in 2D image creation due to being multilayer. The first convolution and the second sampling method rely on CNN. The first method utilizes the filter to render it the coconvolution layer [33]. After this scalar weighting and resulting prejudice, the second process uses pixels from the neighbourhoods to generate a new attribute.

Pictures are more of the day in red, grey, and 96 by 96 measurements, implying 32 by 32. The convolutional layer that brings the network its name is at the core of the convolutional neural network. This layer executes a convolution process. Current layers try to maintain the same size as the input. Convolutional layers appear to acquire local characteristics [32]. Such layers provide several specific methods of tracking input data compared to models 32 or 512 to collect features from an object or multiple forms of seeing and observing instruction. This variety enables variation, for example, not just lines but precise lines in the specific training details. The filter is used for a point product process to achieve a single input and filter depth value. It means that since 32 and 32 filters are in a convolution sheet, they are not only twin but also tridimensional, with correct filter weights for all three channels. Convolution layers often might not be connected to input data, like raw pixel value, but the output of the other layers. Convolution layer initialization allows for the hierarchical redirection of the results. For filters directly operating on raw pixel value, regarding the weighted value like a line characteristic and those filters attempting to extract weaves that combine a function, these filters are assisted by first line layer production, which became multiple lines to display the shape as shown in Figures 4 and 5.

By pool size arguments, the average pooling layers determined the number of rectangular areas. If the magnitude of the pond is the height of 2 and the width of 3, the layer returns to the average region value. We also analyzed the previous pooling layers where the input feature map is sampled, but the global pooling samples the whole feature map in a single value. As FC layers, we call these layers; they flatten the matrix into vectors and nourish it like a neural network into completely interconnected layers. In detecting 3D systems, layers are not better than ultimately linked structures. It is also necessary to learn independently of the characteristics of a convolution layer and a linked layer. By keeping this straightforward, we have a square form picture with a scale of one with c-channels. It requires an illustration of a square convolution layer of size k, as shown in Figure 6.

The graphical representation of the architecture of the model is presented in Figure 6. This graphical representation consists of layers of the model, including dense, pool, and convolutional layers. The representation of the model's architecture is implemented using the visual-Keras library. It

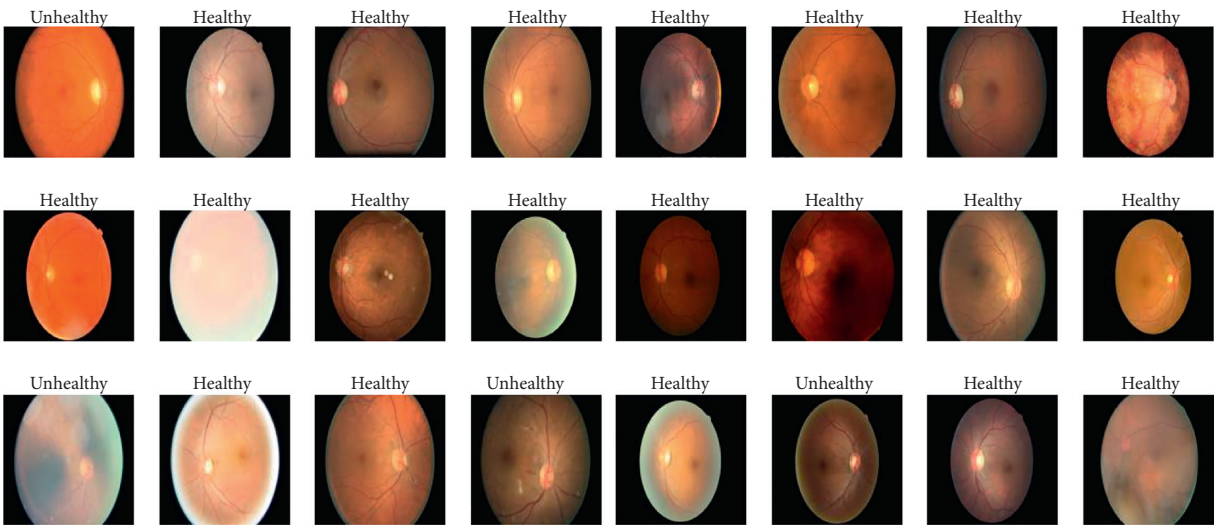


Figure 3: Random samples of healthy and unhealthy eyes' images.

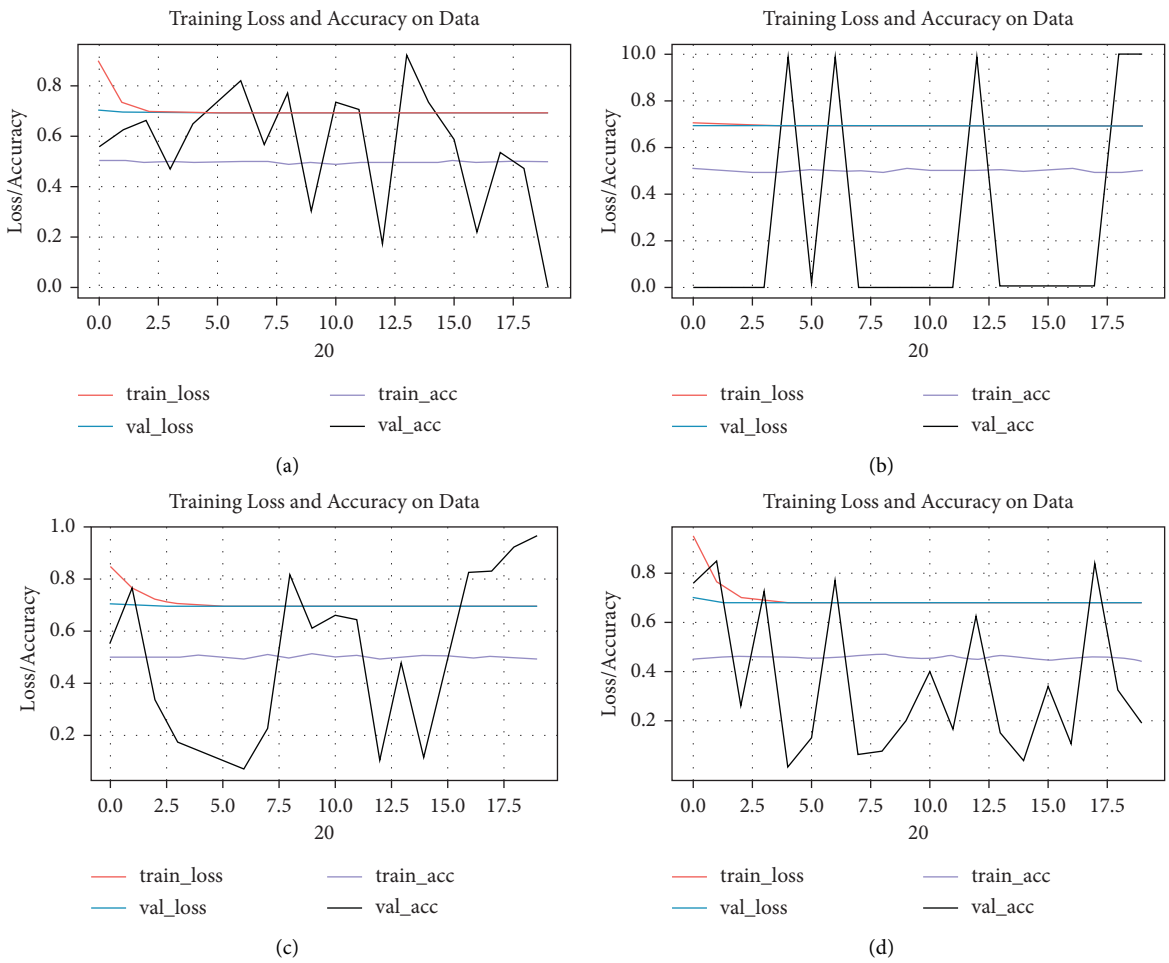
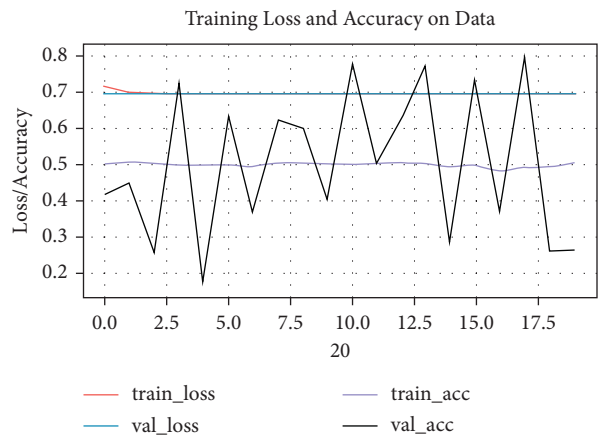


Figure 4: Continued.

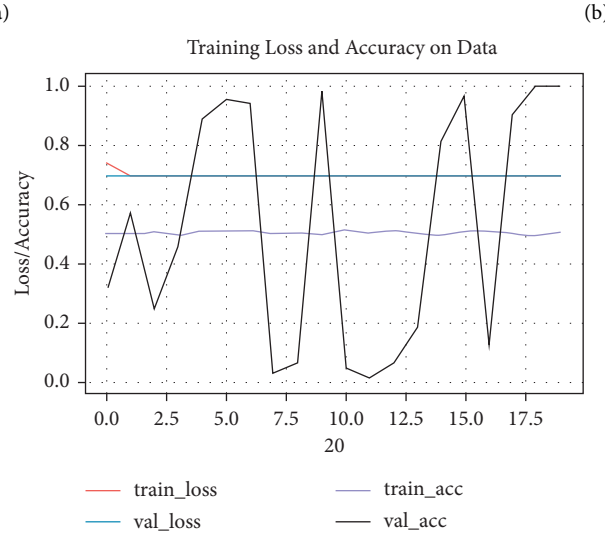


(e)

Figure 4: Training, validation, and loss of deep neural networks on 20 epochs. (a) DenseNet201. (b) EfficientNetB7. (c) InceptionV3. (d) NASNetMobile. (e) VGG16.

(a)

(b)



(c)

Figure 5: Training, validation, and loss of ResNet based deep neural networks on 20 epochs. (a) ResNet50. (b) ResNet101V2. (c) ResNet152.

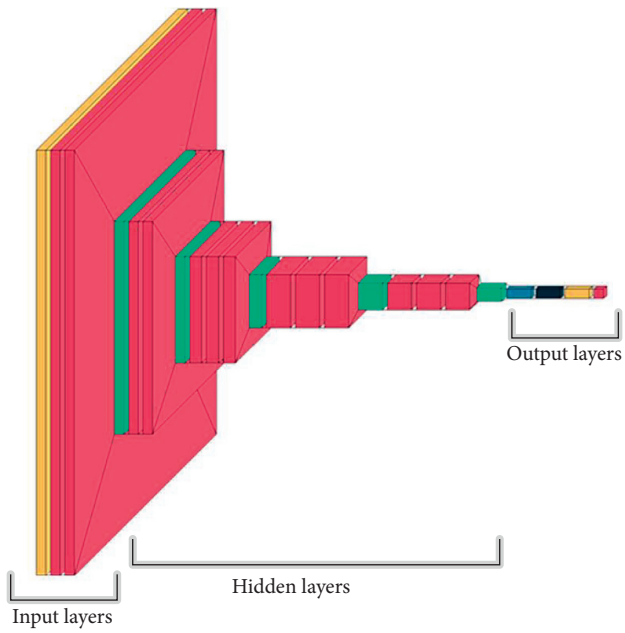


Figure 6: Network visualization with the layer size and connectivity. The left-most layer represents the input, the left-most layer represents output, and the middle are the hidden layers.

is a python package that allows plotting the architecture of the model's architecture. It supports the layered architectural view of the model, which is extremely significant for the convolutional neural network.

In Figure 7 there is a brief comparison in terms of training accuracy, training loss, and validation accuracy of different deep learning networks. The circular diagrams show the graphical comparisons of multiple deep learning models that are implemented in this paper. The graphical representation shows that EfficientNetB7, ResNet152, and InceptionNetV3 have validation accuracies of 100.0%, 99.65%, and 96.47%, respectively. The validation loss is 0.6931 in all the networks. This graphical comparison is extremely significant to visually analyze the models' training and their respective validation loss. The graphical comparison allows the reader to quickly get an idea about the statistics of each model, which helps build a narrative on the selection of deep learning neural networks when used for transfer learning for medical images dataset or any other images dataset. Each deep learning neural network is graphically represented in different color codes to make them different. The validation accuracy, validation loss, and training accuracy are also graphically presented in different color codes. The overall comparison can be visually seen in Figure 7.

Eight different deep learning neural networks are implemented on the glaucoma dataset. The results of various statistics of different models in terms of validation accuracy, validation loss, and training accuracy are presented in Table 3. The table shows that the EfficientNetB7 deep learning model has the highest validation accuracy with 100.0%. ResNet152 has the second-highest validation accuracy with 99.65%, and InceptionNetV3 has the third-highest validation accuracy with 96.47%. The remaining five deep learning models, including FenseNet201, NASNetMobile, ResNet50, ResNet101V2, and VGG16, have less validation accuracy than the other three models. The validation accuracy is the same in all the models with 0.69315.

The evaluation matrix of implemented deep neural networks precision, recall, f1-score, and support is presented in the bar graph in Figure 8. The values of precision, recall, f1-score, and support are graphically described in the bar graph. The graph showed the values in different colors for different evaluation parameters. The bar chart is a very comprehensive way of representing categorical data by summarizing. Data is displayed using different bars, with each bar representing a particular category. Each category is represented with a different color. The height of each bar represents the degree of occurrence of the data. The overall evaluation matrix, including precision, recall, f1-score, and support of the deep neural models implemented in this paper, are represented in the bar chart in Figure 8.

The distribution of values of evaluating matrix is represented in the box plot shown in Figure 9. Box plot is a comprehensive way of describing the distribution of data generally in a five-number summary. The five-number summary is minimum, first quartile, median, third quartile, and maximum. Box plot is extremely significant to get the information about the values that are outliers. The box plot also describes if your data is symmetrical, how tightly the data is grouped, and the skewness. The box plot shown in Figure 8 represents the distribution of the evaluating matrix values of deep neural networks. The values of evaluating matrices such as precision, recall, f1-score, and support are presented in a five-number summary by grouping the values and finding the minimum, first quartile, median, third quartile, and maximum. This five-number summary is

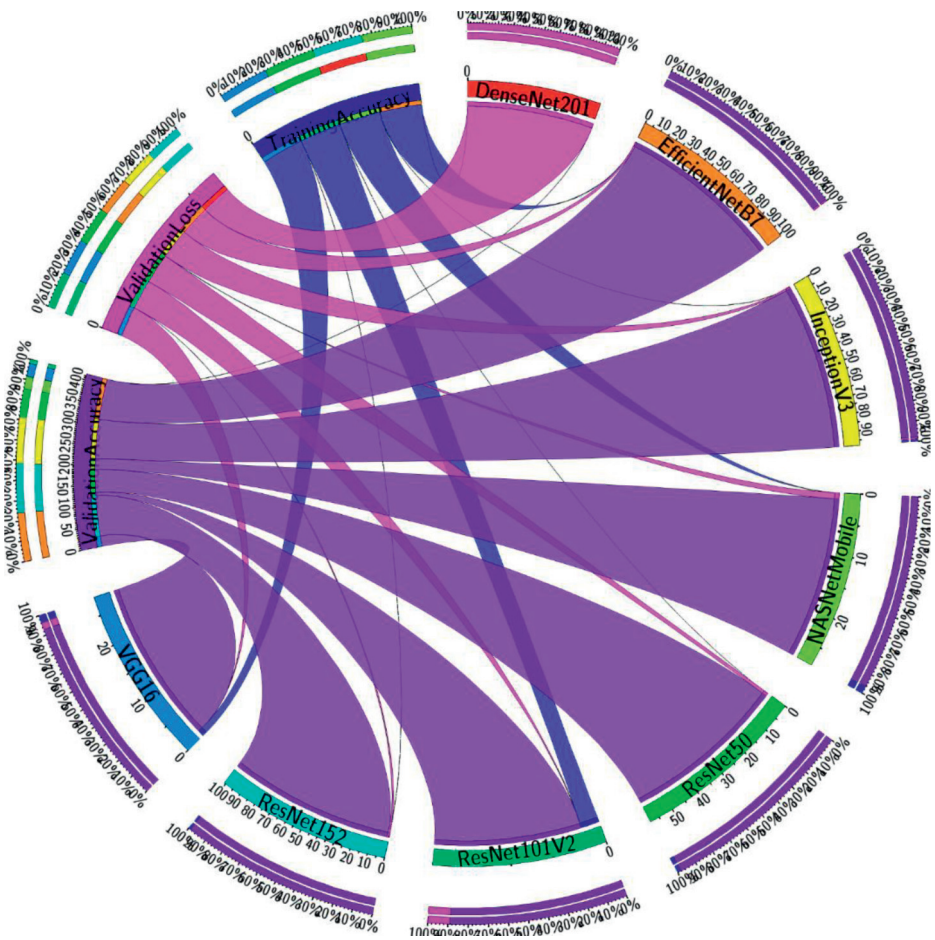


Figure 7: Comparison of different neural networks based on validation accuracy, validation loss, and training accuracy.

Table 3: Validation accuracy, validation loss, and training accuracy of different deep learning neural networks.

	DenseNet201	EfficientNetB7	InceptionV3	NASNetMobile	ResNet50	ResNet101V2	ResNet152	VGG16
Validation accuracy (%)	0.12	100.00	96.47	26.56	56.64	7.87	99.65	26.32
Validation loss	0.69315	0.6931	0.6931	0.6931	0.6931	0.6932	0.6931	0.6931
Training accuracy (%)	0.50	0.50	0.49	0.49	0.50	0.51	0.50	0.51

represented in the form of a box, which is present in the box plot graph shown in Figure 9.

A brief graphical representation of support, f1-score, recall, and precision is presented in Figure 10. On the x-axis, there are deep neural networks that are implemented in this research paper. On the y-axis, three different scales represent different evaluation matrices. The scale with green color represents the precision value of various deep neural networks ranging from 0.1 to 0.8. The plotted curves in green color represent the precision values of deep neural networks placed on the x-axis. The scale with orange color represents the recall, and it ranges from 0.0 to 1.0. The plotted curves with orange color are representing the recall values of each deep neural network concerning each class. The scale with purple color is representing the f1-score ranging from 0.0 to 0.8. The curves with purple color are representing the f1-score values of each deep neural network concerning each class. The blue square boxes in the plot are the support values of each deep neural model concerning each class. This graphical representation of evaluating matrix is extremely significant in terms of describing the model's performance.

A comparison between different deep learning neural networks in terms of precision, recall, f1-score, and support is presented in Table 4 concerning both classes healthy and unhealthy. The precision, also called the positive predicted value, is the fraction between true positive instances and the number of true positive and false positive instances. Eight experiments are carried out with eight different deep learning neural networks, and precision is calculated for each deep neural network for each class. The results of the precision values of each deep neural model are presented. The recall value of each deep learning model for each class is also calculated. The F1-score is the weighted average of precision and recall, and this is also calculated for every deep

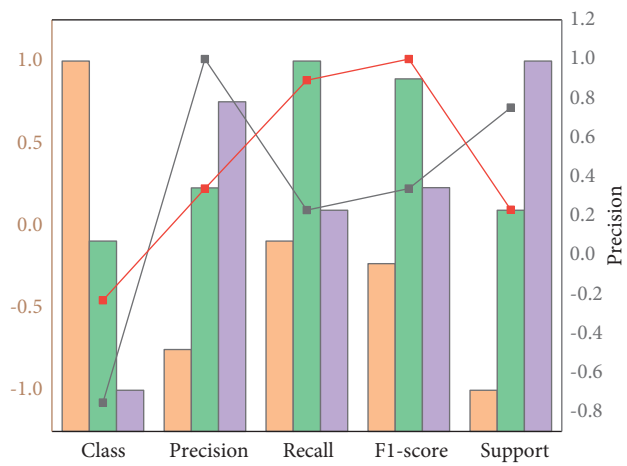


Figure 8: Class, precision, recall, F1-score, and support for implemented deep neural networks.

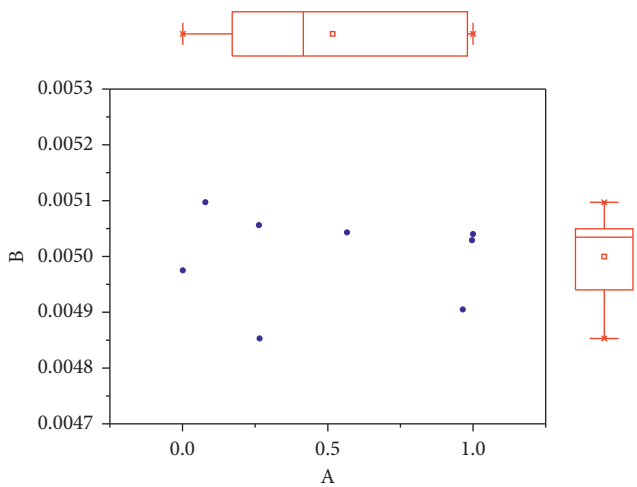


Figure 9: Boxplot for accuracy on the X-axis and other measures on the Y-axis of implemented deep neural networks.

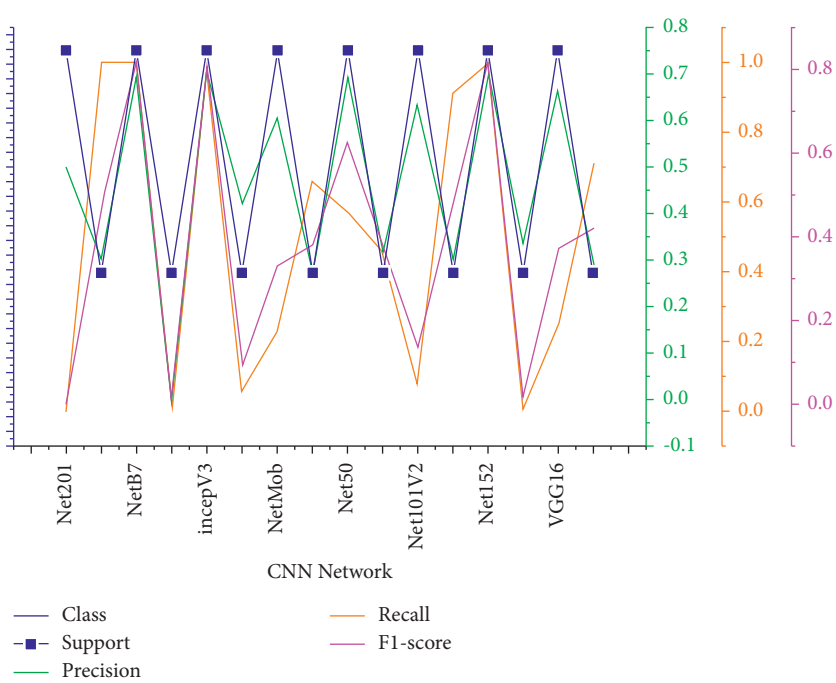


Figure 10: Support, precision, recall, and f1 measure of different CNN networks used in the experiment.

Table 4: A comparison among different deep learning neural networks in terms of precision, recall, F1-score, and support.

CNN network	Class	Precision	Recall	F1-score	Support
DenseNet201	Healthy	0.50	0.00	0.00	1184.00
	Unhealthy	0.30	1	0.47	518
EfficientNetB7	Healthy	0.7	1	0.82	1184
	Unhealthy	0	0	0	518
InceptionV3	Healthy	0.7	0.97	0.81	1184
	Unhealthy	0.42	0.05	0.09	518
NASNetMobile	Healthy	0.61	0.23	0.33	1184
	Unhealthy	0.27	0.66	0.38	518
ResNet50	Healthy	0.7	0.57	0.63	1184
	Unhealthy	0.31	0.45	0.37	518
ResNet101V2	Healthy	0.64	0.07	0.13	1184
	Unhealthy	0.3	0.91	0.45	518
ResNet152	Healthy	0.7	1	0.82	1184
	Unhealthy	0.33	0	0.01	518
VGG16	Healthy	0.67	0.25	0.37	1184
	Unhealthy	0.29	0.71	0.42	518

neural network for each class. The support is all true instances of the class. The support value is also calculated for all deep neural networks to each class. All these values are calculated to form the confusion matrix of the models. The comparison table of these matrices is extremely significant to get information about the performance of each deep learning model. A detailed comparison among the values of all evaluation matrices is presented in Table 4.

6. Conclusions

Before that, too many models of neural network segmentation were planned. Many computer scientists have done some research to boost the classification performance, but the precision of such artificial neural networks is still missing. That is why we have suggested the Inception V-3 approach for image classification based on convolution neural networks. This model we have developed has the potential to address this CNN model's problem of classification accuracy. DenseNet201 achieved recall 1 for unhealthy class, EfficientNetB7 recall is 1 for healthy class, ResNet152 recall is 1 for healthy class, and InceptionV3 achieved recall 0.97 for healthy class. When we take F1-measure, then EfficientNetB7 and ResNet152 beat the other algorithms with 0.82 values. InceptionV3 has 0.81 value for f-measure.

We will use different latest neural networks' parameter tuning to compare the old one we have already used. Further research and improvement in the current work are outlined in future work to achieve this goal entirely. An increase in the number of images in the dataset gives high precision than before. Assessing the output of other neural networks for the dataset defined is planned further. We are developing an application to track the past database glaucoma.

Data Availability

The data supporting this study's findings are available from the corresponding author or Jahanzaib Latif (zaib.chauhan@hotmail.com) upon reasonable request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported in part by the Beijing Natural Science Foundation (No. 4212015), Natural Science Foundation of China (No. 61801008), China Ministry of Education - China Mobile Scientific Research Foundation (No. MCM20200102), China Postdoctoral Science Foundation (No. 2020M670074), Beijing Municipal Commission of Education Foundation (No. KM201910005025).

References

[1] M. C. Staff, "Glaucoma," 2021, https://www.mayoclinic.org/diseases-conditions/glaucoma/symptoms-causes/syc-20372839.

[2] P. S. Grewal, F. Oloumi, U. Rubin, and M. T. S. Tennant, "Deep learning in ophthalmology: a review," *Canadian Journal of Ophthalmology*, vol. 53, no. 4, pp. 309–313, 2018.

[3] M. Khan, T. Liu, and F. Ullah, "A new hybrid approach to forecast wind power for large scale wind turbine data using deep learning with TensorFlow framework and principal component analysis," *Energies*, vol. 12, no. 12, p. 2229, 2019.

[4] H. Naeem, B. Guo, M. R. Naeem, F. Ullah, H. Aldabbas, and M. S. Javed, "Identification of malicious code variants based on image visualization," *Computers & Electrical Engineering*, vol. 76, pp. 225–237, 2019.

[5] D. S. W. Ting, L. R. Pasquale, L. Peng et al., "Artificial intelligence and deep learning in ophthalmology," *British Journal of Ophthalmology*, vol. 103, no. 2, pp. 167–175, 2019.

[6] D. T. Hogarty, D. A. Mackey, A. W. Hewitt, and e. ophthalmology, "Current state and future prospects of artificial intelligence in ophthalmology: a review," *Clinical and Experimental Ophthalmology*, vol. 47, no. 1, pp. 128–139, 2019.

[7] Y. Hagiwara, J. E. W. Koh, J. H. Tan et al., "Computer-aided diagnosis of glaucoma using fundus images: a review," *Computer Methods and Programs in Biomedicine*, vol. 165, pp. 1–12, 2018.

[8] S. M. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and M. K. Khan, "Medical image analysis using convolutional neural networks: a review," *Journal of Medical Systems*, vol. 42, no. 11, p. 226, 2018.

[9] Q. J. I. J. A. C. S. A. Abbas, "Glaucoma-deep: detection of glaucoma eye disease on retinal fundus images using deep learning," vol. 8, no. 6, pp. 41–45, 2017.

[10] J. Gao, Q. Jiang, Q. Jiang, B. Zhou, and D. Chen, "Convolutional neural networks for computer-aided detection or diagnosis in medical image analysis: an overview," *Mathematical Biosciences and Engineering*, vol. 16, no. 6, pp. 6536–6561, 2019.

[11] M. Farhan, S. Jabbar, M. Aslam et al., "A real-time data mining approach for interaction analytics assessment: IoT based student interaction framework," *International Journal of Parallel Programming*, vol. 46, no. 5, pp. 886–903, 2018.

[12] S. Liu, S. L. Graham, A. Schulz et al., "A deep learning-based algorithm identifies glaucomatous discs using monoscopic fundus photographs," *Ophthalmology Glaucoma*, vol. 1, no. 1, pp. 15–22, 2018.

[13] G. Liu, J. He, and X. Xuan, "A data preservation method based on blockchain and multidimensional hash for digital forensics," *Complexity*, vol. 2021, Article ID 5536326, 12 pages, 2021.

[14] A. Piva, "An overview on image forensics," *ISRN Signal Processing*, vol. 2013, Article ID 496701, 22 pages, 2013.

[15] H. Zou, P. Yang, R. Ni, and Y. Zhao, "Anti-forensics of image contrast enhancement based on generative adversarial network," *Security and Communication Networks*, vol. 2021, Article ID 6663486, 8 pages, 2021.

[16] X. Wang, H. Wang, and S. Niu, "An intelligent forensics approach for detecting patch-based image inpainting," *Mathematical Problems in Engineering*, vol. 2020, Article ID 8892989, 10 pages, 2020.

[17] M. Vinícius dos Santos Ferreira, A. Oseas de Carvalho Filho, A. Dalília de Sousa, A. Corrêa Silva, and M. Gattass, "Convolutional neural network and texture descriptor-based automatic detection and diagnosis of glaucoma," *Expert Systems with Applications*, vol. 110, pp. 250–263, 2018.

[18] J. M. Ahn, S. Kim, K.-S. Ahn, S.-H. Cho, K. B. Lee, and U. S. Kim, "A deep learning model for the detection of both advanced and early glaucoma using fundus photography," *PLoS One*, vol. 13, no. 11, Article ID e0207982, 2018.

[19] R. Sarki, K. Ahmed, H. Wang, and Y. Zhang, "Automatic detection of diabetic eye disease through deep learning using fundus images: a survey," *IEEE Access*, vol. 8, Article ID 151149, 2020.

[20] S. Yu, D. Xiao, and Y. Kanagasingam, "Exudate detection for diabetic retinopathy with convolutional neural networks," in *Proceedings of the 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 1744–1747, IEEE, Jeju, Korea (South), July 2017.

[21] P. Chudzik, S. Majumdar, F. Caliva, B. Al-Diri, and A. Hunter, "Microaneurysm detection using deep learning and interleaved freezing," in *Medical Imaging 2018: Image Processing*vol. 10574, International Society for Optics and Photonics, Article ID 105741I, 2018.

[22] Y. Hatanaka, K. Ogohara, W. Sunayama, M. Miyashita, C. Muramatsu, and H. Fujita, "Automatic microaneurysms detection on retinal images using deep convolution neural network," in *Proceedings of the 2018 International Workshop on Advanced Image Technology (IWAIT)*, pp. 1-2, IEEE, Chiang Mai, Thailand, January 2018.

[23] L. Dai, R. Fang, H. Li et al., "Clinical report guided retinal microaneurysm detection with multi-sieving deep learning," *IEEE Transactions on Medical Imaging*, vol. 37, no. 5, pp. 1149–1161, 2018.

[24] T. Saba, S. T. F. Bokhari, M. Sharif, M. Yasmin, and M. Raza, "Fundus image classification methods for the detection of glaucoma: a review," *Microscopy Research and Technique*, vol. 81, no. 10, pp. 1105–1121, 2018.

[25] A. Fourcade and R. H. Khonsari, "Deep learning in medical image analysis: a third eye for doctors," *Journal of Stomatology, Oral and Maxillofacial Surgery*, vol. 120, no. 4, pp. 279–288, 2019.

[26] L. Faes, S. K. Wagner, D. J. Fu et al., "Automated deep learning design for medical image classification by health-care professionals with no coding experience: a feasibility study," *The Lancet Digital Health*, vol. 1, no. 5, pp. e232–e242, 2019.

[27] A. Katzmann, A. Muehlberg, M. Suehling, D. Nörenberg, J. W. Holch, and H.-M. Gross, "Deep random forests for small sample size prediction with medical imaging data," in *Proceedings of the 2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pp. 1543–1547, IEEE, Iowa City, IA, USA, April 2020.

[28] M. Smaida and S. Yaroshchak, "Bagging of convolutional neural networks for diagnostic of eye diseases," in *Proceedings of the COLINS*, pp. 715–729, Lviv, Ukraine, April 2020.

[29] S. R. Hameed and H. M. J. A.-Q. J. O. P. S. Ahmed, "Eye diseases classification using back propagation with parabola learning rate," *Tikrit Journal of Pure Science*, vol. 26, no. 1, pp. 1–9, 2021.

[30] U. R. Acharya, M. R. K. Mookiah, J. E. W. Koh et al., "Automated diabetic macular edema (DME) grading system using DWT, DCT features and maculopathy index," *Computers in Biology and Medicine*, vol. 84, pp. 59–68, 2017.

[31] K. Hu, B. Shen, Y. Zhang, C. Cao, F. Xiao, and X. Gao, "Automatic segmentation of retinal layer boundaries in OCT images using multiscale convolutional neural network and graph search," *Neurocomputing*, vol. 365, pp. 302–313, 2019.

[32] H. Harkat, A. E. Ruano, M. G. Ruano, and S. D. Bennani, "GPR target detection using a neural network classifier designed by a multi-objective genetic algorithm," *Applied Soft Computing*, vol. 79, pp. 310–325, 2019.

[33] L. Zou, S. Yu, T. Meng, Z. Zhang, X. Liang, and Y. Xie, "A technical review of convolutional neural network-based mammographic breast cancer diagnosis," vol. 2019, Article ID 6509357, 16 pages, 2019.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 3320436, 9 pages
https://doi.org/10.1155/2021/3320436

Research Article

Privacy-Aware Data Forensics of VRUs Using Machine Learning and Big Data Analytics

Muhammad Babar,[1] **Muhammad Usman Tariq**,[2] **Ahmed S. Almasoud**,[3] **and Mohammad Dahman Alshehri**[4]

[1]*Department of Computer Science, Allama Iqbal Open University, Islamabad, Pakistan*
[2]*Abu Dhabi School of Management, Abu Dhabi, UAE*
[3]*College of Computer and Information Sciences, Prince Sultan University, Riyadh, Saudi Arabia*
[4]*Department of Computer Science, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia*

Correspondence should be addressed to Muhammad Babar; muhammad.babar@aiou.edu.pk

Received 27 September 2021; Revised 4 November 2021; Accepted 12 November 2021; Published 28 November 2021

Academic Editor: Farhan Ullah

The present spreading out of big data found the realization of AI and machine learning. With the rise of big data and machine learning, the idea of improving accuracy and enhancing the efficacy of AI applications is also gaining prominence. Machine learning solutions provide improved guard safety in hazardous traffic circumstances in the context of traffic applications. The existing architectures have various challenges, where data privacy is the foremost challenge for vulnerable road users (VRUs). The key reason for failure in traffic control for pedestrians is flawed in the privacy handling of the users. The user data are at risk and are prone to several privacy and security gaps. If an invader succeeds to infiltrate the setup, exposed data can be malevolently influenced, contrived, and misrepresented for illegitimate drives. In this study, an architecture is proposed based on machine learning to analyze and process big data efficiently in a secure environment. The proposed model considers the privacy of users during big data processing. The proposed architecture is a layered framework with a parallel and distributed module using machine learning on big data to achieve secure big data analytics. The proposed architecture designs a distinct unit for privacy management using a machine learning classifier. A stream processing unit is also integrated with the architecture to process the information. The proposed system is apprehended using real-time datasets from various sources and experimentally tested with reliable datasets that disclose the effectiveness of the proposed architecture. The data ingestion results are also highlighted along with training and validation results.

1. Introduction

In a recent technological globe, data are mounting rapidly, and humans are mostly relying on data. Besides the pace at which the data rise, it is becoming impracticable to stock up the data into any specific server. Today the planet holds an enormous quantity of data that persists to grow exponentially at very high speed and is insecure [1]. Moreover, the entire globe has gone online with the invention of the web, and every single action we do puts down a digital map out that is prone to vulnerability [2]. With the rise of big data and machine learning, the notion of improving accuracy and enhancing the efficacy of AI projects is also gaining importance and is largely recognized [3]. Some of these factors of the evolution of data are the enhancement of technology, social media, and Internet of Things (IoT). IoT is one of the latest concepts in the current age that is mostly applicable in traffic controlling and monitoring applications. The future of this globe is secure IoT that will be going to alter today's world objects into intelligent and smart objects [4]. Smart systems include IoT devices, such as sensors and actuators, process input connectivity, and people. Sensors and

actuators are acting as a backbone for any emerging system. The interactions among all these components create a new type of smart application and service. With the rise of IoT devices, the idea of edge computing is also gaining prominence and is broadly recognized. Machine learning solutions provide improved guard safety in hazardous traffic circumstances in the context of traffic applications [5–7].

As several new-fangled and ground-breaking technologies pledge benefits through enhanced optimization of traffic community systems, "Smart" traffic system development chooses the best of these techniques and services to resolve traffic most imperative confronts [8–10]. Hence, the smart traffic trend going towards the higher side. There are many aspects of urban from transportation management to building blueprint and community safety, which are examined as grown for reinvention. Besides, some cutting-edge and imperative technologies such as cloud computing, robotics, artificial intelligence, machine learning, big data, and particularly, machine learning seems progressively more within the reach [3, 11]. The overall big data analytics process goes through several stages to serve the purpose [12]. These stages include identification of the problem, designing the data requirements, preprocessing, data loading, performing processing and analytics, and data visualization. Firstly, all the problems are needed to be identified accurately which are required to be addressed using big data analytics. Then, all the data requirements are designed to provide a logical solution to be executed in the later stages. Big data are usually very chaotic, messy, incoherent, incomplete, and inconsistent [13].

Therefore, proper preprocessing is required to be done before processing the big data. Consequently, the next phase is the data loading before the data processing and analytics of big data. The smart traffic environment is based on IoT devices and objects generating gigantic data (Big Data) which requires efficient aggregation, processing, and analysis to achieve optimal results for decision-making [14, 15]. Efficient exhaustive analysis of such data is not possible through traditional data analytics techniques. On the contrary, some big data analytics methods are also found in the past other than traditional methods; however, there is no all-inclusive, common, and effective resolution proposed to aggregate and process the big data produced in an IoT-based smart traffic environment [16–18]. The existing solutions are based on traditional or classical Hadoop framework. Moreover, the data ingestion or data loading performance of big data files into Hadoop is overlooked in the existing solutions, which is one of the major factors affecting the overall processing [19, 20]. Big data analytic involves smart management of the data to give real-time monitoring of the data population of the VRUs which has drastically expanded everywhere throughout the world. The solutions using IoT big data are proposed for the VRUs' information management along with traffic management. This research prefers the customization of the YARN parallel and distributed framework. However, to comprehend the reliability of the smart traffic, many challenges are required to be addressed where privacy is one of the most brutal between the imperative challenges.

2. Literature Review

A malevolent hit on the services of users can be extremely costly in the context of the trustworthiness of edge computing [21, 22]. Hence, this article presents a secure architecture for data supervision to deal with data security challenges in smart traffic applications. The work related to the proposed architecture about data analytics and machine learning for smart traffic data management is very significant. The key problems decorated in the architecture are the use of traditional MR cluster, inadequate data piling, intangible structure, and only specific dataset [10, 23]. A scheme was discussed in detail in the context of V2X connections [24]. The bog data analytics approaches are also taken into consideration including Tiers that are accountable for various steps and activities of the data analytics. Though it is a complete four-tier architecture consisting tiers from data collection to data analysis usage, it causes processing delay [25, 26], and a classical map-reduce framework is used that slows down the performance. Moreover, data aggregation before data loading is focused while data loading competence is overlooked. The data aggregation of results is preferred and data loading before analysis is overlooked in this architecture.

An approach is proposed for reducing the conflict between VRUs and automated vehicles [27]. This proposal is only focusing on automated vehicles. It does not support big data processing in general. The key issue is the data ingestion performance in this model. It takes a lot of time to insert the big data into the system for processing. Some researchers proposed a model based on data analysis that promotes the notion of smart traffic and utilizes the big data to be processed but overlooks the data loading efficiency. A framework is presented to overcome the VRUs' issues, but these researchers also overlook the data loading and ingestion into a distributed environment. There are some models proposed to deal with the same problem of Big Data analysis in the smart environment [14, 28], but this solution is the utilization of the conventional cluster resource management scheme and insufficient data loading to the Hadoop server. Moreover, architecture is proposed to investigate the data in a transport environment that is more accessible and efficient [29]. However, it causes an additional delay in processing, and the said scheme is only tested for transportation datasets, and data load efficiency is overlooked while loading Big Data to Hadoop server as well. The additional delay affected the overall performance of the big data analytics.

On the contrary, a scheme is proposed using a parallel processing approach. Though a YARN-based solution is offered, the data loading efficiency is still overlooked in this architecture. The standard practice of traditional data analytics techniques is to analyze the limited data only, which generates an open area of errors and biases in the Big Data scenario. Another challenge that needs to be addressed is insufficient data loading into the traditional cluster management framework, e.g., Hadoop. The traditional data loading challenges are time-consuming, more storage required, commands are difficult, no append, and no partial ingestion. Similarly, Hadoop processing based on traditional

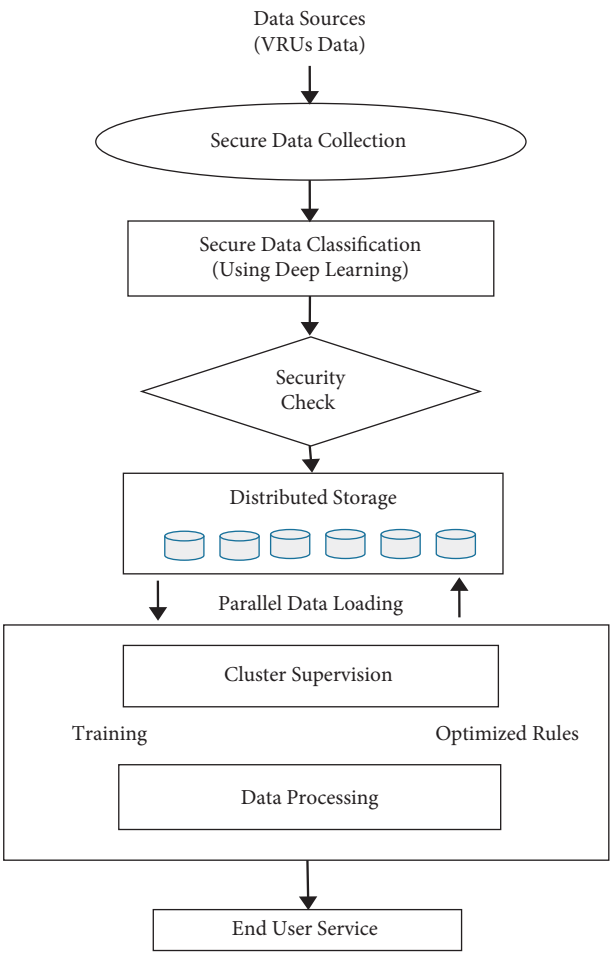


Figure 1: Overview of the proposed framework.

cluster management challenges includes scheduling issues, inefficient load balancing, scalability issues, NameNode availability, and responsibility unification. The objective of this research is to propose a framework based on edge intelligence to process enormous data efficiently and overcome the data loading and processing issue. IoT gathers data and directs the driver to follow the free lanes. A specific proposal is designed to realize the map-reduce paradigm integrated with Apache Spark for real-time data processing comprehension of big data. Spark deals with hasty computation and allows reusability. Effective data ingestion into the distributed storage mechanism is missing in the loading and storage process efficiently.

The current work has deficiencies in the big data storage and processing for IoT-enabled intelligent transportation. Furthermore, model parallelism is also missing for effective extrapolation and decision-making. The proposed research will propose a framework to overcome the existing challenges. Trust and privacy in the smart traffic application, particularly considering the VRUs, is a prejudiced experience that brings complexity in recognizing the attacks. The insecure VRUs in the smart traffic applications could cause a breakdown in the transportation monitoring and controlling services. Therefore, to enhance security, we need to evaluate the level of insecurity in an application first. This study proposed a secure architecture based on machine learning in the smart traffic domain that evaluates the privacy level of the VRUs.

3. Proposed Architecture

The proposed architecture based on machine learning connects the smart community departments (e.g., traffic monitoring and control department). The data sources are comprised of traffic monitoring and controlling big data. The workflow of the proposed parallel and the distributed scheme is depicted in Figure 1. Data gathering is done by the respective units collected from various traffic control sources (e.g., sensors and cameras). To devise effective parallel and distributed architecture, the data must be scrutinized before computation. The data are generated by different devices such as environmental sensors, security monitoring sensors, traffic cameras, and transportation monitoring sensors. The data are properly collected by the various departments such as the traffic-controlling authorities. This process is known as secure data collection. The data are classified using the machine learning approach.

The data are given to the proposed parallel and distributed architecture to process using proposed modules. The number of parallel changes is balanced using the fixed block size of the chunk. The default block size of the utility is time-consuming and has less parallelism. The default size is optimized and modified to improve the data loading efficiency. This data collection is a part of a distributed system. It involves overall data management that includes aggregation, collection, and storage. The data are also preprocessed before injecting into the proposed scheme to remove noise and anomalies for speeding up the processing activities. Afterward, the data are divided into different chunks for parallel processing at the edge level. The distributed storage mechanism is also taken into consideration to assist the parallel processing. The YARN parallel and distributed platform for big data analytics is preferred because the cluster management is dealt with separately by the resource manager that is a part of the YARN. Premediated algorithms are applied while processing the data in the cluster.

The processed results are sent for decision-making to the concerned smart society services' providers that are finally forwarded to the users. Following filtration, the Hadoop processing unit is used to process the data which are stored in the distributed storage mechanism. Lastly, the analyzed data are operated for community planning. The data are collected from the departments, and the decisions are sent back to the community development departments. The objective is to realize a smart traffic scheme to perform processing and keep the data private. The said-community departments are the data sources for the proposed system and a mediator between the system and the user. Architecturally, the anticipated solution consists of 3 modules that are data security, organization, and processing, which are shown in Figure 2.

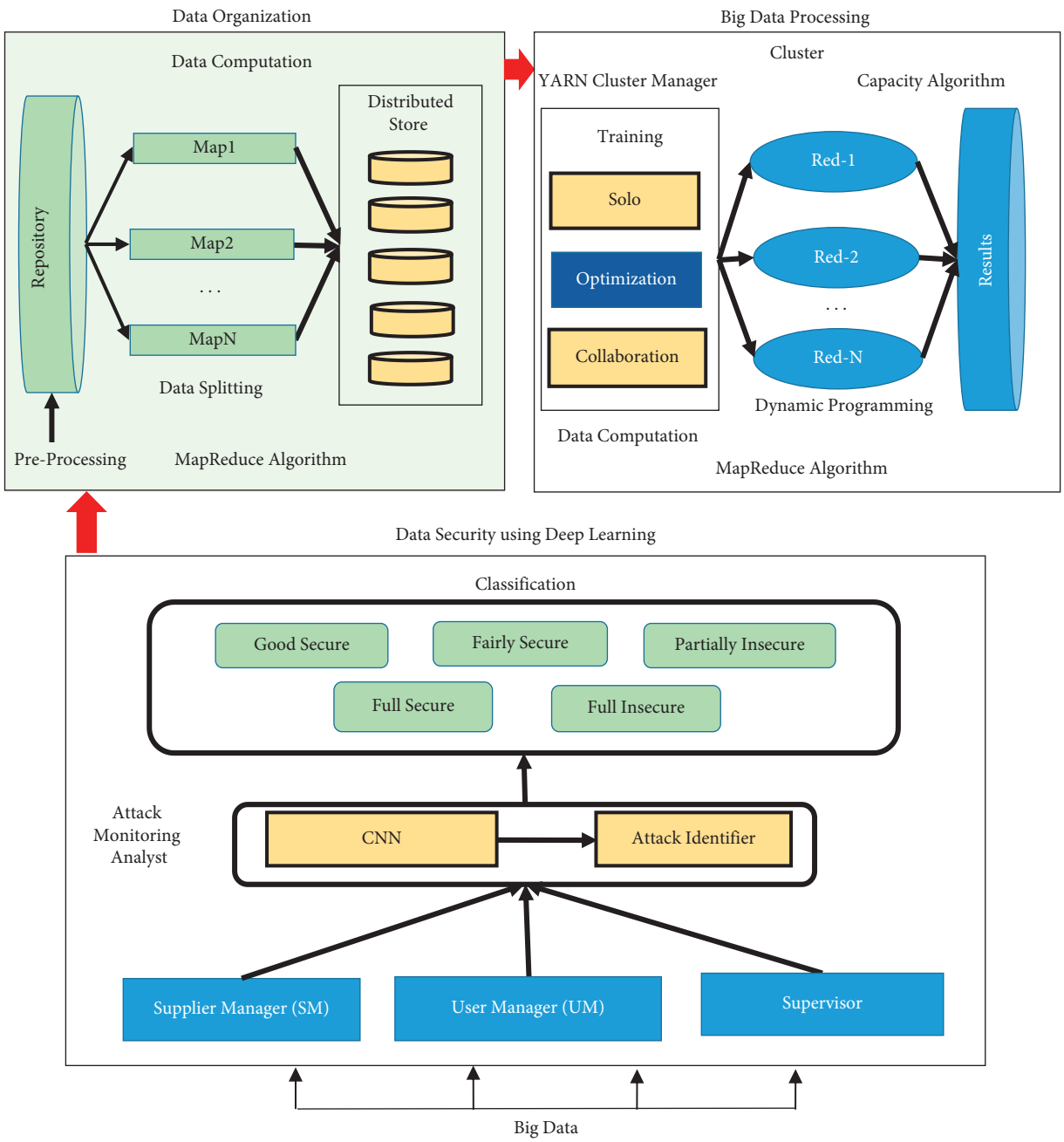


Figure 2: Proposed architecture.

3.1. Data Security Layer. The proposed structural design has a security layer for keeping secure the VRUs' data from attacks. It is a part of smart traffic architecture. It recommends flexibility in opposition to the attacks. The supplier manager (SM), user manager (UM), and superviser are the components' security layer. The SM and UM pay attention to the supplier and the user, while the supervisor applies the algorithm of machine learning. The CNN DL technique is integrated that classifies secure or insecure data. The SM is accountable for the profile maintenance of every supplier, and the UM is accountable for the profile maintenance of users. The supervisor is trained using the classifier. The level of security is predicted using special classes that are highly secure (HS), fine secure (FS), moderately secure (MS), highly insecure (HIS), and partly insecure (PIS). Equation (1) is used to compute the security score:

$$\text{Lev\_of\_Security} = \begin{cases} \text{HS}, & \text{if Sec} = 1, \\ \text{FS}, & \text{if Sec} \geq 0.76 \text{ and } S < 1, \\ \text{MS}, & \text{if Sec} \geq 0.51 \text{ and } S < 0.76, \\ \text{HIS}, & \text{if Sec} \geq 0.26 \text{ and } S < 0.51, \\ \text{PIS}, & \text{if Sec} = 0 \text{ and } S < 0.26, \end{cases} \tag{1}$$

where Sec is computed using

$$\text{Sec} = \frac{1}{n}\sum_{k=0}^{n} Fk. \quad (2)$$

Equation (1) is used to calculate the various levels of security. The purpose of the different security levels is to give the particular score to the candidate user for the prospect. The major purpose of the multiclassification is to identify the watch list of the risks in the future. It helps identify the intruders with less score to be analyzed further for future investigation.

3.2. Big Data Organization. The big data organization system involves the overall data management including aggregation, collection, and storage. The data are distributed across various nodes for computation to get a load from the central server or cloud. Intelligent applications are supported by acquiring data via the Internet from various local devices. Several devices that include sensors, cameras, and object-mounted devices record the information of the environment in the different domains. This data are later utilized for analysis to get insights and produce intelligent decisions. It is the first layer that is accountable for assembling the data from different community departments that are used to manage the smart community development services. A practical community does not only hold large data only but also includes versatile and wide-ranging processing areas. The smart community implementation is dependent on all forms of data processing due to their heterogeneous nature. Data collection is used to transform signals that are assessed in practical circumstances and converts outcomes to the digital form for processing. The collection is done by a special system that converts data from analog to digital form. The smart traffic centers pull out the data using various sensors in the community to gather real-time data. The data organization layer further contains the data aggregation, where the data are grouped based on the identification of the connected devices. This aggregation process is implemented due to the data size because the data are very massive and required to be assembled for efficient processing. The aggregation improves the modularity and processing.

3.3. Big Data Processing. This unit is the main processing part that preprocesses the raw data initially including the irrational data combination, missing values, and values beyond the range which are integrated before processing. If the data are not inspected for such problems, there could be misleading results during decision-making. Hence, the transformation is also done to scale the data to a specific scale. Then, the data are taken by a parallel processing unit that is the backbone of the proposed architecture. The proposed architecture is based on a parallel computing model called MapReduce that is utilized. MapReduce is introduced to realize big data analytics. This programming paradigm is composed of Map and Reduce functions. It is a useful model that exploits huge datasets and processes them in parallel. It executes processes in a distributed manner and offers high availability. The underlying system also manages machine failures, performance issues, and efficient communications. Task distribution in the cluster is carried out using the YARN distributed cluster management framework. The YARN is equipped with dynamic programming for task distribution and cluster management. The previous platforms such as MapReduce paradigm are only responsible for the processing. The YARN is preferred because the cluster management is dealt with separately by the resource manager that is a part of the YARN. The fair algorithm is integrated with YARN to perform scheduling. Besides, interleaving is possible between map and reduce phases; therefore, the reduced phase might begin before the map phase finishes.

4. Results and Discussion

The proposed scheme is implemented using the parallel and distributed platform of Hadoop version 3.0. The Hadoop is equipped with Apache Spark module. The reliable datasets are utilized. The pyspark library is utilized in Python 3.8. Similarly, the resilient agent evaluation is carried out using a detailed setting with a machine learning classification module. The machine learning library is also utilized and implemented in Python 3.8. The comparative analysis of the proposed design is provided with current proposals. The experimental results and comparison disclose the effectiveness of the proposed design. The discussion about the results is provided in this section. Results are produced using various reliable datasets to assess the proposed architecture based on parallel and distributed paradigms using premeditated algorithms. We performed a noise and anomalies removal process on data on top of our proposed architecture. The anomalies are removed using the min-max normalizations and Kilman algorithm. The data ingestion is achieved using the map-only algorithm.

The traditional YARN cluster management framework is customized with improved capacity and a fair algorithm of scheduling. We applied the dynamic algorithm to set the parameters of the YARN framework dynamically. The processing is performed using MapReduce algorithms. We also optimize the MapReduce algorithm for edge computing to utilize at every edge. Thus, notable efficiency is achieved in the processing time. The proposed architecture implemented using the Hadoop parallel and distributed framework along with optimized algorithms. These datasets are preferred due to the utilization of this dataset in the literature. We deliberately executed almost the same queries to compare the processing time and throughput of proposed edge-enabled IoT architecture using customized MapReduce and YARN for parallel processing.

4.1. Data Security Results. The results and experiments of the security layer include the required training of the dataset using an ML classifier. The model is trained using secure and insecure interaction with the proposed architecture. The assessment of the security layer is performed in a specific setting. Initially, the model was trained on $365 * 925$ matrices. The training process of the Naïve Bayes classifier is shown in Figure 3.

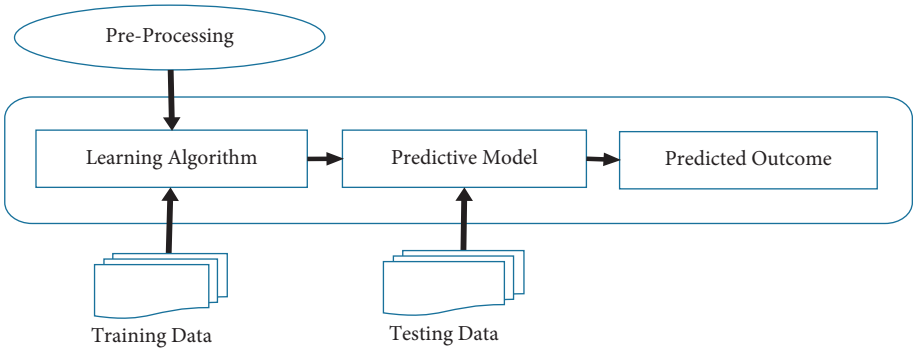


Figure 3: Model training.

Table 1: Confusion matrix.

		Predicted	
		Secure	Insecure
Labeled	Secure	TP	FN
	Insecure	FP	TN

Table 2: Proposed model results using the confusion matrix.

		Predicted	
		Secure	Insecure
Actual	Secure	96.3%	3.7%
	Insecure	4.1%	95.9%

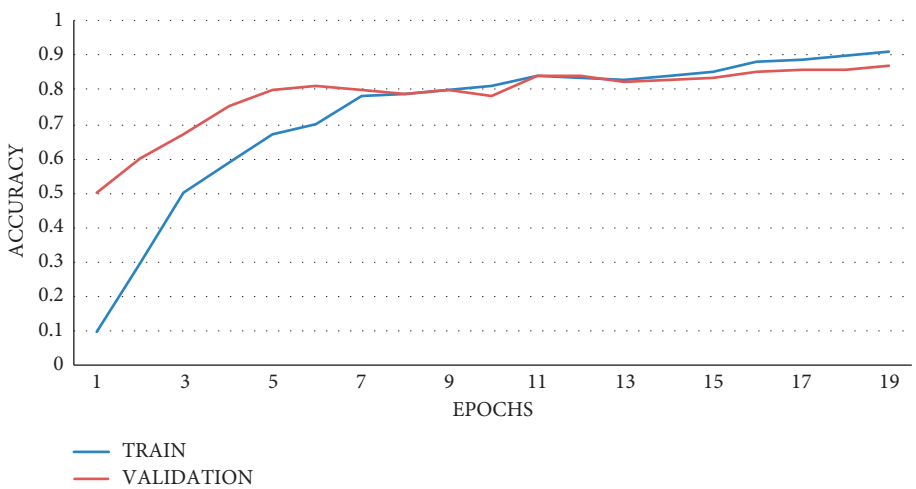


Figure 4: Accuracy.

The proposed resilient agent evaluation is also performed by setting a specific environment where the proposed model is trained using the proposed model. To assess the proficiency of the proposed model, the confusion matrix is exploited, as depicted in Table 1. To measure the effectiveness of the classifier, the confusion matrix is utilized concerning two classes (e.g., secure and insecure), as shown in Table 2. The value is considered secure if it is greater than 0.5; otherwise, it is considered insecure. The performance measures are applied to the ML technique utilized for a resilient agent. The accuracy of the technique is expressed in the form of percentages in Table 2. The specific value of percentage of each confusion matrix value is also highlighted in Table 2.

4.2. Training and Validation Results. Figure 4 is the confirmation of the enhanced accuracy of the validation and training. The enhanced level of accuracy in training and validation is a result of the enlarged number of epochs (e.g., 200 epochs). Likewise, Figure 5 reveals the proposed model's validation and training loss that is the indication of minimal loss. The reduction in the loss is a result of the enlarged number of epochs (e.g., 200 epochs).

4.3. Data Ingestion Results. It gets nearly no time to load the dataset into Hadoop when the dataset size is small. There are not quite time differences of data loading either manually or

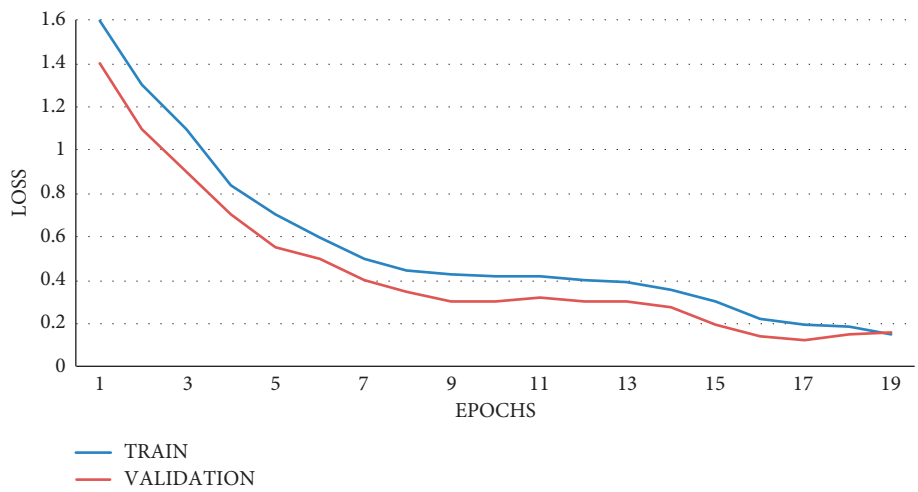


Figure 5: Loss.

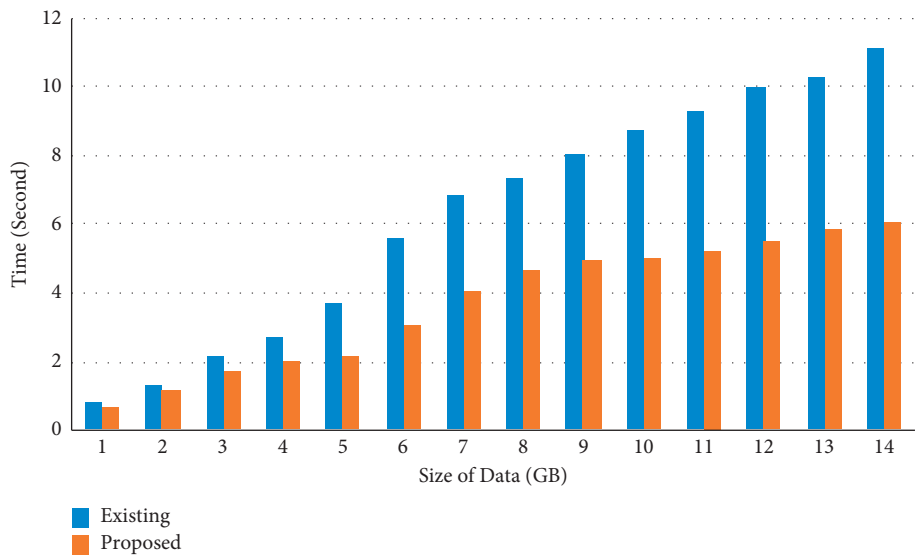


Figure 6: Overall data loading efficiency.

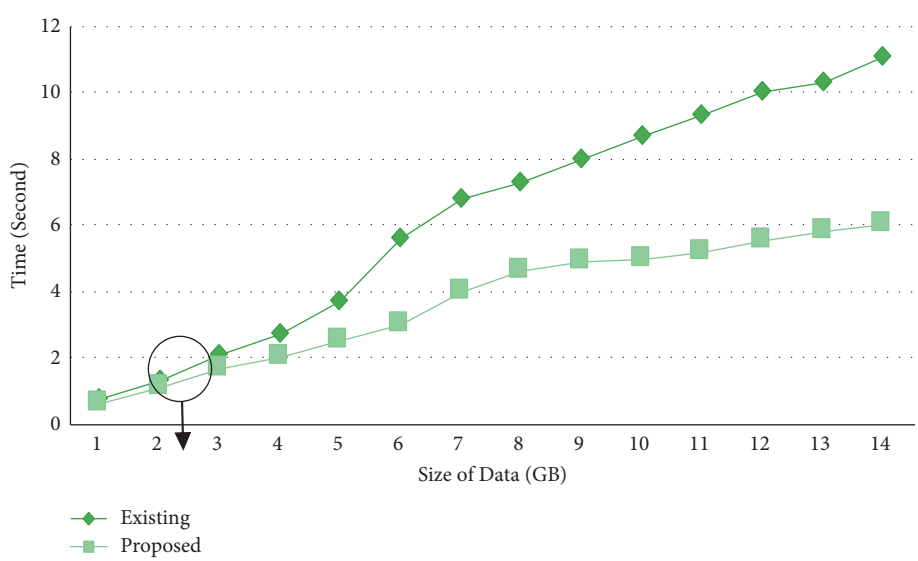


Figure 7: Dataset size threshold (overall).

using the specific utility. It has been experimentally proved that it gets nearly no time to load the dataset into Hadoop when the dataset size is small.

Overall, the proposed system efficiency including all the parameters' modification of data loading is shown in Figure 6. In the same way, Figure 7 demonstrates the threshold for all the parameters' modification of data loading using the proposed system. The threshold is the alarming set value that highlights the focal point where the difference between existing and proposed schemes starts. The proposed scheme is manual in the context of data ingestion and automated in the context of classification and processing.

5. Conclusion

A smart traffic application is considered by the extensive expansion of IoT-connected devices particularly with the rise of Big Data and machine learning. Machine learning solutions provide efficient results in the context of efficiency and accuracy of the machine learning models. However, it becomes challenging to tackle the privacy of the users in the smart traffic management and surveillance of the users because that produces an enormous amount of big data to be processed and analyzed efficiently. In this study, an architecture is proposed based on machine learning to process big data efficiently in a secure environment considering user privacy. The proposed architecture is a layered framework with a parallel and distributed module using machine learning on big data to achieve secure big data analytics. A specific privacy layer is proposed that classifies the dishonest entities using machine learning. The proposed system is apprehended using real-time datasets from various sources and experimentally tested with reliable datasets that disclose the effectiveness of the proposed architecture. The data ingestion results are also highlighted along with training and validation results. This study proposes an architecture based on machine learning to process big data efficiently in a secure environment considering user privacy. The proposed design is the optimization of the existing parallel and distributed framework to achieve efficient processing. The current proposals lack efficient parallel data ingestion and efficient mechanism for communication overhead. Therefore, the security challenges using machine learning are explored in this paper. This paper proposes a separate secure and resilient module to overcome the privacy issue of the users. The proposed architecture is equipped with a resilient agent using an ML classifier. A stream processing unit is also integrated with the architecture to process the information produced by edge devices.

Data Availability

The data used to support the findings of the study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Taif University Researchers Supporting (project no. TURSP-2020/126), Taif University, Taif, Saudi Arabia.

References

[1] M. I. Razzak, M. Imran, G. Xu, and G. Xu, "Big data analytics for preventive medicine," *Neural Computing & Applications*, vol. 32, no. 9, pp. 4417–4451, 2020.

[2] N. Paltrinieri, L. Comfort, and G. Reniers, "Learning about risk: machine learning for risk assessment," *Safety Science*, vol. 118, pp. 475–486, 2019.

[3] S. K. Maurya and A. Choudhary, "Deep learning based vulnerable road user detection and collision avoidance," in *Proceedings of the 2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 1–6, IEEE, Madrid, Spain, 2018 September.

[4] M. Ahmad, T. Younis, M. A. Habib, R. Ashraf, and S. H. Ahmed, "A review of current security issues in internet of things," *Recent Trends and Advances in Wireless and IoT-enabled Networks*, Springer, Singapore, pp. 11–23, 2019.

[5] M. Garcia-Venegas, D. A. Mercado-Ravell, and C. A. Carballo-Monsivais, "On the safety of vulnerable road users by cyclist orientation detection using Deep Learning," 2020, https://arxiv.org/abs/2004.11909.

[6] M. Goldhammer, S. Köhler, S. Zernetsch, K. Doll, B. Sick, and K. Dietmayer, "Intentions of vulnerable road users—detection and forecasting by means of machine learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 7, pp. 3035–3045, 2019.

[7] Z. Ahmed and R. Iniyavan, "Enhanced vulnerable pedestrian detection using deep learning," in *Proceedings of the 2019 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0971–0974, IEEE, Chennai, India, April 2019.

[8] M. Babar and F. Arif, "Real-time data processing scheme using big data analytics in internet of things based smart transportation environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 10, pp. 4167–4177, 2019.

[9] E. Carter, P. Adam, D. Tsakis, S. Shaw, R. Watson, and P. Ryan, "Enhancing pedestrian mobility in smart cities using big data," *Journal of Management Analytics*, vol. 7, pp. 1–16, 2020.

[10] W. Tabone, J. de Winter, C. Ackermann et al., "Vulnerable road users and the coming wave of automated vehicles: expert perspectives," *Transportation Research Interdisciplinary Perspectives*, vol. 9, Article ID 100293, 2021.

[11] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2014.

[12] A. Sharma, G. Singh, and S. Rehman, "A review of big data challenges and preserving privacy in big data," in *Advances in Data and Information Sciences*, pp. 57–65, Springer, Singapore, 2020.

[13] S. Boubiche, D. E. Boubiche, A. Bilami, and H. Toral-Cruz, "Big data challenges and data aggregation strategies in wireless sensor networks," *IEEE Access*, vol. 6, pp. 20558–20571, 2018.

[14] D. Nallaperuma, R. Nawaratne, T. Bandaragoda et al., "Online incremental machine learning platform for big data-driven

smart traffic management," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4679–4690, 2019.

[15] A. Ahmad, M. Babar, S. Din et al., "Socio-cyber network: the potential of cyber-physical system to define human behaviors using big data analytics," *Future Generation Computer Systems*, vol. 92, pp. 868–878, 2019.

[16] M. Elkhodr, B. Alsinglawi, and M. Alshehri, "Data provenance in the internet of things," in *Proceedings of the 2018 32nd international conference on advanced information networking and applications workshops (WAINA)*, pp. 727–731, IEEE, Krakow, Poland, May 2018.

[17] S. G. Farrag, N. Sahli, Y. El-Hansali, E. M. Shakshuki, A. Yasar, and H. Malik, "STIMF: a smart traffic incident management framework," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 85–101, 2021.

[18] A. Ahmad, M. Khan, A. Paul et al., "Toward modeling and optimization of features selection in big data based social internet of things," *Future Generation Computer Systems*, vol. 82, pp. 715–726, 2018.

[19] J. Yang, Y. Han, Y. Wang, B. Jiang, Z. Lv, and H. Song, "Optimization of real-time traffic network assignment based on IoT data using DBN and clustering model in smart city," *Future Generation Computer Systems*, vol. 108, pp. 976–986, 2020.

[20] N. Cárdenas-Benítez, R. Aquino-Santos, P. Magaña-Espinoza, J. Aguilar-Velazco, A. Edwards-Block, and A. Medina Cass, "Traffic congestion detection system through connected vehicles and big data," *Sensors*, vol. 16, no. 5, p. 599, 2016.

[21] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: survey and open issues," *IEEE Access*, vol. 6, pp. 18209–18237, 2018.

[22] C. Johnsson, A. Laureshyn, and T. De Ceunynck, "In search of surrogate safety indicators for vulnerable road users: a review of surrogate safety indicators," *Transport Reviews*, vol. 38, no. 6, pp. 765–785, 2018.

[23] A. Siulagi, J. F. Antin, and L. J. Molnar, S. Bai, S. Reynolds, O. carsten, and R. greene-roesel, Vulnerable road users: how can automated vehicle systems help to keep them safe and mobile?" in *Road Vehicle Automation*, vol. 3, pp. 277–286, Springer, Cham, 2016.

[24] N. Dasanayaka, K. F. Hasan, C. Wang, and Y. Feng, "Enhancing vulnerable road user safety: a survey of existing practices and consideration for using mobile devices for V2X connections," 2020, https://arxiv.org/abs/2010.15502.

[25] H. S. A. J. J. Sun and R. Bie, "Internet of Things and big data analytics for smart and connected communities," *IEEE Access*, vol. 4, pp. 766–773, Mar. 2016.

[26] R. Tönjes, P. Barnaghi, M. Ali et al., "Real time iot stream processing and large-scale data analytics for smart city applications," in *Proceedings of the European Conference on Networks and Communications (poster session)*, Bologna, Italy, 2014.

[27] J. M. Owens, R. Greene-Roesel, A. Habibovic, L. Head, and A. Apricio, "Reducing conflict between vulnerable road users and automated vehicles," in *Road Vehicle Automation*, vol. 4, pp. 69–75, Springer, Cham, 2018.

[28] A. Ahmad, A. Paul, M. M. Rathore, and H. Chang, "Smart cyber society: integration of capillary devices with high usability based on Cyber–Physical System," *Future Generation Computer Systems*, vol. 56, pp. 493–503, 2016.

[29] S. Shukla, K. Balachandran, and V. S. Sumitha, "A framework for smart transportation using Big Data," in *Proceedings of the 2016 International Conference on ICT in Business Industry & Government (ICTBIG)*, pp. 1–3, IEEE, Indore, India, November 2016.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 2897565, 14 pages
https://doi.org/10.1155/2021/2897565

Research Article

SESCon: Secure Ethereum Smart Contracts by Vulnerable Patterns' Detection

Amir Ali,[1] **Zain Ul Abideen**,[2] **and Kalim Ullah**[3]

[1]*School of Cyber Science and Engineering, Xian Jiaotong University Xian, Shaanxi 710049, China*
[2]*Department of Computer Science, Xian Jiaotong University Xian, Shaanxi 710049, China*
[3]*Department of Computer Science, CECOS University of IT and Emerging Sciences, Peshawar, Pakistan*

Correspondence should be addressed to Amir Ali; amir.ali@stu.xjtu.edu.cn

Received 11 July 2021; Revised 12 August 2021; Accepted 20 August 2021; Published 21 September 2021

Academic Editor: Farhan Ullah

Ethereum smart contracts have been gaining popularity toward the automation of so many domains, i.e., FinTech, IoT, and supply chain, which are based on blockchain technology. The most critical domain, e.g., FinTech, has been targeted by so many successful attacks due to its financial worth of billions of dollars. In all attacks, the vulnerability in the source code of smart contracts is being exploited and causes the steal of millions of dollars. To find the vulnerability in the source code of smart contracts written in Solidity language, a state-of-the-art work provides a lot of solutions based on dynamic or static analysis. However, these tools have shown a lot of false positives/negatives against the smart contracts having complex logic. Furthermore, the output of these tools is not reported in a standard way with their actual vulnerability names as per standards defined by the Ethereum community. To solve these problems, we have introduced a static analysis tool, SESCon (secure Ethereum smart contract), applying the taint analysis techniques with XPath queries. Our tool outperforms other analyzers and detected up to 90% of the known vulnerability patterns. SESCon also reports the detected vulnerabilities with their titles, descriptions, and remediations as per defined standards by the Ethereum community. SESCon will serve as a foundation for the standardization of vulnerability detection.

1. Introduction

E-commerce has now prevailed on the Internet to purchase everything using digital cash, where a trusted third party can process and verify the transactions made on the Internet across the globe. However, the security concerns in online transactions are prevailing everywhere due to their centralized nature. For example, users can spend the same digital cash multiple times, hence causing the double-spend attack. Similarly, tampering of digital records through hacking or even by insiders has no 100% secured solution, yet. Therefore, to overcome this problem, blockchain was introduced in the bitcoin application [1] and provides a strong solution for the double-spending amount and immutability without any trusted party.

The main strong aspect of blockchain is to provide irreversible transparent transaction history on the distributed ledger. Anyone can see the complete history of user transactions. No one can change or alter the contents of this distributed ledger (blockchain). After a few years from the inception of bitcoin blockchain, researchers had realized the strong potentials of its Distributed Ledger Technology (DLT), which can be applied to other domains in parallel to its cryptocurrency applications. Vitalik Buterin provided a concept of smart contract (SC) to extend the blockchain applicability to other domains and introduced the Ethereum blockchain [2] and also introduced their own cryptocurrency, i.e., Ether. Smart contract has Turing complete language, i.e., Solidity, where any business logic can be defined and programmed. A smart contract is basically a piece of code that is deployed and stored permanently on blockchain with a unique address and executed automatically upon receiving some transactions.

There are two types of accounts in Ethereum blockchain, namely, Externally Own Account (EOA) and Smart Contract Account (SCA). An EOA is normally controlled through a private key that Ethereum users save in a secured place (Wallet). Anyone can create this account without paying any cost. This account can initiate different types of transactions in the Ethereum blockchain network, i.e., transferring/receiving Ethers (Ethereum cryptocurrency) from one account to another and signing transactions by its private key to prove its authorization. On the other hand, SCA is a little bit different in its nature. For example, to create this account we need some fractions of Ether and it has some associated code (written in Solidity/other languages), which will be deployed and stored on blockchain network permanently. We cannot remove/delete such an account from the blockchain network except to disable it. SCA can only initiate transactions when it receives transactions and do some action upon fulfilling the predefined criteria, hence providing some sort of automation. Both types of accounts have balance in the form of Ethers.

In the Ethereum network, millions of smart contracts have been deployed with a balance of billions of US dollars, which shows the potentials and trust of a lot of investors and business ventures. However, due to the irreversibility nature of its distributed ledger, we cannot change/alter SC's code if there would be some logical errors. If any vulnerabilities are found in SC after deploying it on blockchain, then it is very hard to fix its bugs by simply patching its source code like in conventional applications. Therefore, if there is any vulnerability found in SC, then we have to fix it and deploy it on the blockchain with a new address and also deactivate the previously deployed one.

SC holds billions of dollars and has been mostly utilized in the financial sector and related activities, hence attracting the attacker to steal its money. Therefore, it demands a strong security guarantee to prevent its breach. Unfortunately, it is very hard to produce bugs-free SC, since the developers are not trained and do not have so much experience in the blockchain environment. As a result, a large number of the critical vulnerabilities had been found in most famous smart contracts, i.e., the DAO [3], Frozen Ethers [4], and King of Ether [5], and caused loss of millions of dollars.

To secure the SC and detect its security vulnerabilities before deployment to blockchain, many researchers have contributed to the static and dynamic analysis of SC. Some has provided fuzzing solution (ContractFuzzer [6]), symbolic execution (Oyente [7], ZUES [8], and Securify [9]), fuzzing + symbolic execution (ILF [10]), taint analysis (Slither [11], NPChecker [12]), XPath of vulnerabilities patterns (SmartCheck [13]), etc. The main problems with these static analysis solutions are the high percentage (up to 70%) of false positives and false negatives. Furthermore, reporting of their detection is based on their own defined taxonomy of patterns, which may not be compatible with Ethereum defined standards patterns. Therefore, it is very difficult to evaluate such tools and measure their true positives/negatives against standard defined patterns.

Well-known vulnerabilities patterns in smart contracts have been defined by experts from the Ethereum community [14]. They [14] give each vulnerability with its title, description, relationship with CWE (Common Weakness Enumeration), remediation, samples of vulnerable contracts, and their fixed versions. If we consider these patterns and obey their guidelines, we can develop and deploy smart contracts with confidence and provide bug-free smart contracts. Our research work is based on these standards, and we have followed their guidelines during the detection of vulnerabilities. However, most of the state-of-the-art tools mentioned above could not detect such well-known critical vulnerabilities in smart contracts, or/and their output is not reported in a standard way [14]. They just detect the existence of some keywords with parameter values and give alarms (false positives/negatives) without checking them intelligently. For example, the most well-known attack of DAO on a smart contract even in its simple form [14] could not be detected by the SmartCheck [13] and/or other tools.

In this paper, we are going to provide an intelligent solution toward a static analysis of smart contracts. Our solution is based on taint analysis [15] and XPath of vulnerabilities patterns technique [13]. We first take the source code of the smart contract and convert it to an XML parse tree in the form of Abstract Syntax Tree (AST) where each node represents different constructs (statements, expressions, loops, conditions, variables, etc.). From AST, we become able to detect all its variables, statements, and vulnerable keywords/patterns through XPath queries. Then we pass it to our taint analysis module, where its dependent and nondependent variables, local and state variables, and their flows are detected. We then make vulnerable patterns as defined in standard vulnerability patterns [14]. As we follow the guidelines provided by Ethereum experts [14], therefore, our tool SESCon is aware of all vulnerabilities patterns and thus provides very high accurate vulnerability detection. Our main contributions in this research work are as follows:

(i) We have reduced false positives during vulnerabilities detections by providing an intelligent tool, i.e., SESCon, based on taint analysis which statistically analyzes the smart contract's source code written in Solidity language.

(ii) Since the state-of-the-art tools have defined their own taxonomy of vulnerabilities and detected source code against them, which could not be compared with the standards defined by the Ethereum community, and it became hard to evaluate their efficiency, we are the first, as per our knowledge, to detect vulnerabilities against these standards [14], hence providing a foundation for standardization toward detection and reporting vulnerabilities.

(iii) We have also applied our tool to detect vulnerabilities in real contracts. Our results show that a large number of vulnerabilities still exist in almost all contracts (i.e., 99%).

The rest of the paper is organized as follows: the related work is presented in Section 2; Section 3 gives some background to understand the domain of our research work. The detailed solution of our architecture is described in Section 4; in Section 5, experimental results, evaluation, limitation, and implementation are illustrated; and we finally concluded our research work with future extension in Section 6.

2. Related Work

In this section, we have reviewed some state-of-the-art works, which are related to static analysis directly or indirectly.

ContractFuzzer [6] has detected some vulnerabilities by instrumenting the EVM and defined some test oracles against such vulnerabilities. It has generated a lot of fuzzing inputs from contract ABI specs and invoked the contracts and analyzed their execution log. Oynte [7], on the other hand, by using symbolic execution techniques with Z3 constraint solver on contract bytecode, has detected three types of bugs, TOD, Timestamp, mishandle exception, and some level of reentrancy, whereas ZUES [8] used abstract interpretation and symbolic execution techniques to provide a formal solution for the correctness and fairness of SC in XACML style. It performed static analysis on IR to determine verification predicate points for assertion checking. Similarly, Securify [9] uses the symbolic execution, and also its algorithm has learned the semantics of code and then detected vulnerabilities patterns against violation/compliance patterns. In the first attempt by ILF [10] to fuzz the contract after symbolic execution, it has used the imitation learning framework by using a neural network to model its fuzzing policy to generate effective inputs for the sequence of transactions. However, ILF needs constructor valid parameters to deploy which causes hindrance toward automatically fuzzing the contract under test. Slither [11] converts Solidity contract to Intermediate Representation (SlithIR) using SSA (Static Single Assignment) form and applied data flow and taint tracking techniques to detect vulnerabilities. However, we cannot use Slither on a large dataset of contracts if they have a different version of the Solidity compiler.

NPChecker [12] provides the solution for some payment call bugs by employing the nondeterminism behavior of Ethereum (using bytecode to LLVM IR), without any already know pattern-based vulnerabilities, but has more false positives which need some human interaction to verify. It has adopted taint analysis for local and global variables of SC to explore information flow. However, it has not provided the solution for common arithmetic issues (i.e., integer overflow). Meanwhile, other works [16–22] also addressed the DoS, reentrancy, and Transaction Origin vulnerabilities on EVM bytecode patterns by using the symbolic execution and machine learning techniques without considering the standard patterns of vulnerabilities. EtherTrust [23] provided a static solution through formally defining security properties (reachability) of the semantic of SC (bytecode) into Horn clauses and queries were solved by the Z3 SMT solver. They focused on single reentrancy (SE) and independent miner transaction (MI) bug-related vulnerabilities. VerX [24] introduced a new symbolic execution engine in which delayed abstraction is applied to verify the functional safety properties of SC. They first formalized the temporal safety properties of SC by extending Solidity language (i.e., always and once). Then SC is instrumented to achieve the reachability property and, at the end, the symbolic execution engine has performed its function. Vandal [25] worked on EVM bytecode to abstract interpretation using declarative language, i.e., Souffle, and then generate logic relation. They used a logic-driven approach for defining security vulnerabilities. However, defining the new vulnerabilities needs expertise, which is not user-friendly. There were also some other works [26, 27], which are presented to provide semantic and formal correctness of smart contract at bytecode and smart contract functional level.

To summarize and conclude the related works, most tools (ContractFuzzer, Oynte, ZUES, ILF, Securify, Vandal, etc.) that have used symbolic execution or fuzzing or formally verified the security of smart contracts are operated on bytecode and/or source code. However, their solutions have a lot of overheads during the conversion of source code to some Intermediate Representations (IR), generating test cases for fuzzing, or dynamic running for symbolic executions to detect vulnerabilities. These solutions have not achieved 100% code coverage and hence left some well-known vulnerabilities undetected. Although Slither [11] used taint techniques to construct their Intermediate Representations constructs (SlitherIR), their solution is not directly applied to the source code of smart contract. Also, their solution is time-consuming due to the construction of its IR and SSA (Static Single Assignment) form. NPChecker [12] also uses taint techniques, but it is only related to payments activities and its payments bugs, whereas our SESCon tool is used to find all the security vulnerabilities as defined by the Ethereum community [14]. The most related to our work is SmartCheck [13], which uses static analysis through AST of Solidity source code by translating its grammar in ANTLR and detecting patterns of vulnerabilities by XPath technology. However, it just detects some keyword or some simple pattern; however, its false positive rate is very high, i.e., 69%. Further, for complex patterns, SmartCheck fails. We have extended SmartCheck with taint analysis so that we can detect vulnerabilities patterns intelligently. Another most related work [28] also uses XPath techniques to detect vulnerabilities in smart contracts, but their work is limited for integer overflow.

3. Background

Blockchain can be described as an append-only distributed database (called ledger) where data (transactions) are stored in chronological order. A group of transactions on this ledger are packed into a block and each block is cryptographically linked to the previous block, hence making a chain of blocks (called blockchain). The most important aspect of this technology is that its architecture is decentralized, which means there is no single server that would be

responsible to process and manage/store all transactions [1]. Therefore, blockchain technology is inherently secured from DoS/DDoS, since there is no central point of failure. Anyone can join this peer-to-peer network just by installing its open-source software and participate without any permission, which is called permissionless blockchain, i.e., bitcoin [1] and Ethereum [2]. There are also some permissioned blockchains, where access control layers have been introduced to provide additional security in the system [29–33] so that certain actions must be performed by only authorized participants.

In general, when a user wants to interact with the blockchain, he/she just sends a transaction (i.e., Alice sends 1 bitcoin/Ether to Bob), and this transaction is passed to a pool of transactions (mining pool), where special nodes (miners) select transactions randomly as per their priority (mostly have high fees) and put them into a logical block and produce a hash of the whole block with some restriction of its contents (leading number of zeros) with a puzzled algorithm. When a miner solves such a puzzle of leading zero to produce the hash of the block, he broadcasts this newly generated block to a P2P blockchain network (bitcoin, Ethereum, etc.). All the other nodes update their locally stored blockchain ledger, after the verification and validation of this new block. This puzzle is often called a consensus algorithm which is normally described as Proof of Work (PoW, in case of bitcoin blockchain) or Proof of Stack (PoS, in case of Ethereum 2.0 blockchain) [2]. All the transactions in any block are stored in Merkle Tree form [34], to achieve fast processing time and low storage space. A simple logical illustration of the blockchain is depicted in Figure 1.

However, to extend the blockchain functionality toward automation, the smart contract was provided [2]. Basically, a smart contract is a piece of program code written in any computer programming language (i.e., Solidity [35], Vyper [36]). This code is permanently deployed on the blockchain after compilation into bytecode and gets executed in Ethereum Virtual Machine (EVM) when certain conditions (written in the smart contract) are met. So, there is no third party to control its execution and hence provide a kind of automation (see details in Figure 2).

However, the main problem with smart contracts is that, when it is deployed on the blockchain, we cannot update it or patch it, if some vulnerabilities are found. However, there are some types of actions like Suicide and SelfDestruct which can disable your contract after transferring its balance to a newly deployed contract with the new address. Smart contract development is in its infancy stage, and that is why developers are not well aware of its complexities in terms of the blockchain environment, which leads to a lot of vulnerabilities.

4. SESCon Architecture

In this section, we have presented our SESCon (Secure Ethereum Contracts) solution architecture. First, we described its overview, then we have explained its modules, namely, Vulnerable Patterns module, XPath module, and the Taint module, and, at the end, we present our solution in algorithm form.

4.1. Overview. First of all, in the Vulnerable Patterns module, we have generated vulnerable patterns from the dataset of samples of smart contracts [14]. Details of this module are given in this section. In XPath module, we feed smart contract's source code to get its Intermediate Representation (IR) and then find some vulnerable patterns by utilizing the XPath queries. However, we do not totally depend on XPath, because it leads to false positives in the case of complex rules [13]. Therefore, we have introduced the Taint module on the output of the XPath module. In this module, we first identify its local and global variables, the most important being the state variables which can change the status of SC. We then captured the variables flow and its dependency graph [38] as per our defined algorithm. When we have captured all the information about source code, variable flow, and dependency graph, we then compare the SC under test with the vulnerability patterns defined by the Ethereum community [14]. Finally, SESCon also generates a report about the vulnerabilities patterns found in the source code of the Solidity file with its location, line number, and its potential solutions (please see Figure 3).

4.2. Vulnerable Patterns' Module. The purpose of this module is to generate vulnerable patterns of smart contracts in light of samples of vulnerable smart contracts as provided by the Ethereum community [14]. Since we have known vulnerable patterns of smart contracts in advance, therefore, we just use them to construct vulnerable patterns. This module consists of five tasks as shown in Figure 4.

4.2.1. Sample Contracts. Vulnerable smart contract samples are collected along with their fixed solutions [14]. These fixed solutions help the developer to correct the source code and make the smart contract fixed from vulnerability.

4.2.2. Preprocessing. The aforementioned vulnerable samples have gone through preprocessing steps, where we manually convert those samples into proper standard smart contract form so that we can fully capture its Intermediate Representation (IR).

4.2.3. Representation Learning. The IR consists of an Abstract Syntax Tree (AST), where we have identified the vulnerable keywords and labeled them. After the parsing of AST, we get the dependency graphs and their tokens to learn the potentials vulnerabilities [39].

4.2.4. Classification. In the classification phase, we have assigned the severity of each vulnerable pattern to high, medium, or low. For example, the Outdated Compiler Version vulnerability is considered as low, whereas the reentrancy vulnerability is considered high and so on.

4.2.5. Vulnerable Patterns. In the end, all patterns have been stored in our database, which has been used as a standard

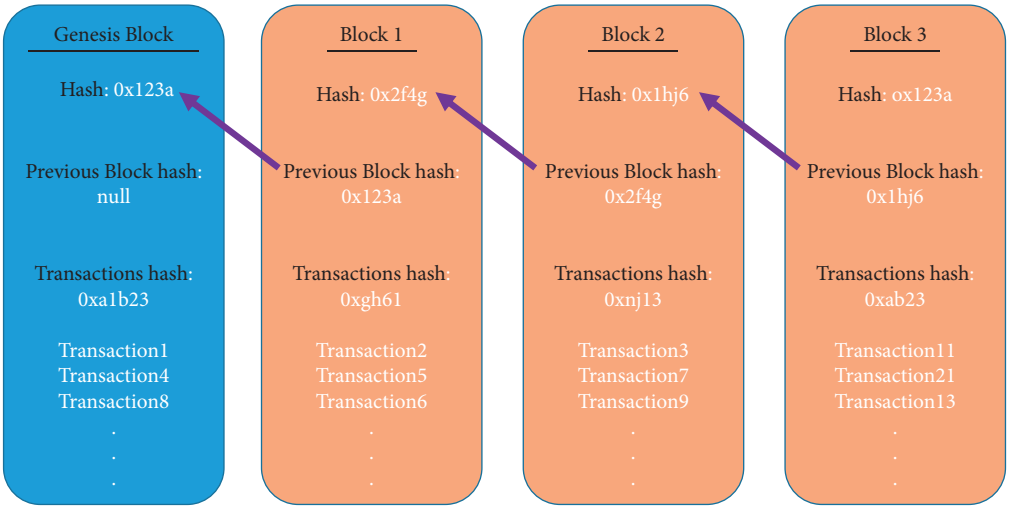


Figure 1: Overview of the blockchain [1].

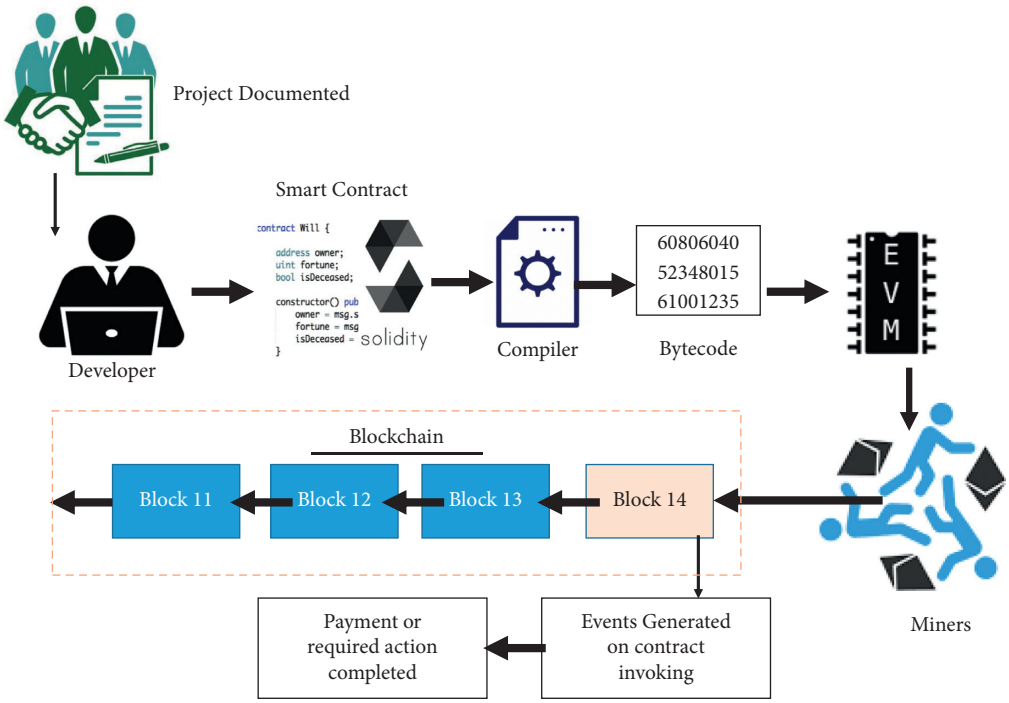


Figure 2: Smart contract execution cycle [37].

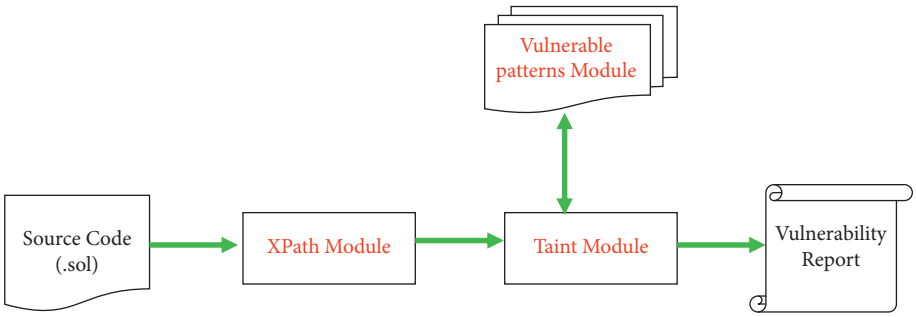


Figure 3: Overview of the SESCon architecture.

Figure 4: Vulnerable patterns' module.

reference to detect vulnerability in real-world smart contracts.

4.3. XPath Module. In this module, first, we take the source code of SC written in a Solidity language (.sol file) and convert it to its XML parse tree in the form of Abstract Syntax Tree (AST). After getting the AST of the source code, now we have all its keywords, i.e., statements, expressions, and variables. Using XPath queries, we have detected vulnerabilities as in [13]. This module takes the Solidity (.sol) source code file as input. We first achieved its AST as shown in Figure 5.

Then using Solidity grammar and ANTLR [40], we have constructed the XML parse tree. This tree was then analyzed through XPath [41] to detect some vulnerable keywords and patterns. By parsing source code as an XML tree, we have achieved 100% code coverage and all elements are matched through XPath queries. It also provides the vulnerabilities locations and line numbers as XML attributes. Therefore, we can easily indicate the vulnerabilities in our source code of smart contract under test. XPath module and the following taint module are given in Figure 6.

4.4. Taint Analysis Module. This module takes the output of the XPath module as input and extracts Control Flow Graph, Protected Functions, local variables and state variables, and Data Dependency Graph. Here we see which variables can/cannot read/write and especially cause a change in the status of smart contract by modifying the state variables. This module provided us with all the internal and external flow of contracts, i.e., which function can change the state variables, and which functions are dependent. We have provided here a sample contract with all the information which can be visualized as given in Figure 7.

Figure 7 shows that the smart contract is comprised of the main contract (apexSolid.sol), especially its purchase function, which is interacting with functions and other contracts, i.e., SafeMath and ERC721. The data flow from one function to another function is shown by an arrow. The green arrow is for the internal flow of data, whereas the orange arrow indicates the external flow of data. We have also extracted the protected, private, and public methods and the variables that can change the value of Smart Contract's state variable and which method is payable. In the end, we make patterns and label them as SmartContractPatterns, and these patterns will go through test. The same procedure is also applied to sample standard vulnerable contracts to extract their patterns. Finally, we compare these two patterns and show vulnerabilities, if comparing results was true. Most of the tools could not detect even the known critical

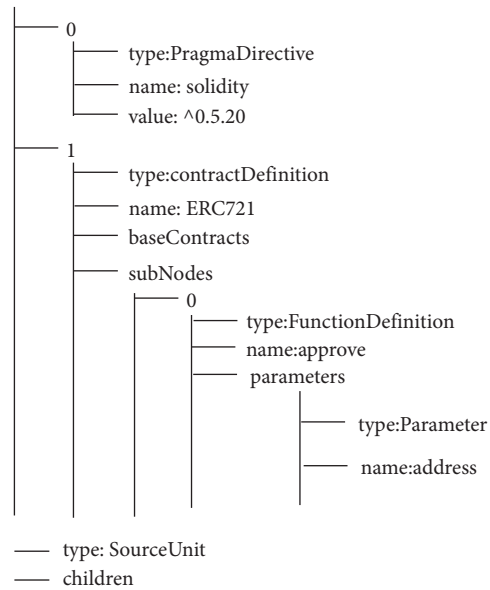


Figure 5: Abstract syntax tree (AST).

vulnerabilities in a smart contract. They just detect the existence of keywords with some parameter values and give their alarms (false positives/negatives) without checking intelligently. For example, the most well-known attack of DAO on a smart contract [14], even in its simple form (given in Figure 8), could not be detected by [13].

First, we give sample standard DAO code (Figure 8) to different tools and check whether they detect DAO vulnerability or not. If smart contracts have this kind of vulnerability, most state-of-the-art tools have detected this vulnerability. However, when we feed the corrected fixed version of the DAO contract (Figure 9), suggested by the Ethereum community [14], to tools, they almost failed and gave us false positives about the reentrancy attack. To provide a solution for these kinds of issues, there is a need for some intelligent algorithms, which must be aware of data flow before and after the execution of each statement. For example, if we feed a vulnerable standard sample pattern of a contract to the tool, it must report it as vulnerable. However, when we feed the fixed version (Figure 9) of the vulnerable contract (after removing vulnerable code from the contract), then the tool must show this new fixed contract (Figure 9) as a secure contract. For this purpose, we have introduced an intelligent algorithm and implemented it in the form of the SESCon tool.

Therefore, our algorithm is not dependent only on just keywords and simple patterns. These patterns are a well-known sequence of instructions that lead to some vulnerabilities in smart contract, for example, calling external

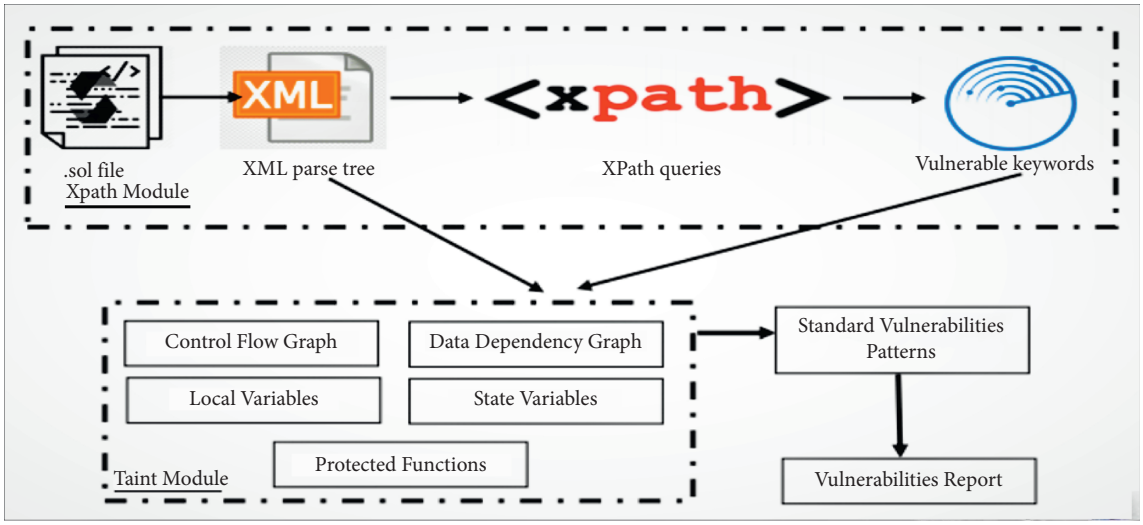


Figure 6: Detailed components of SECon.

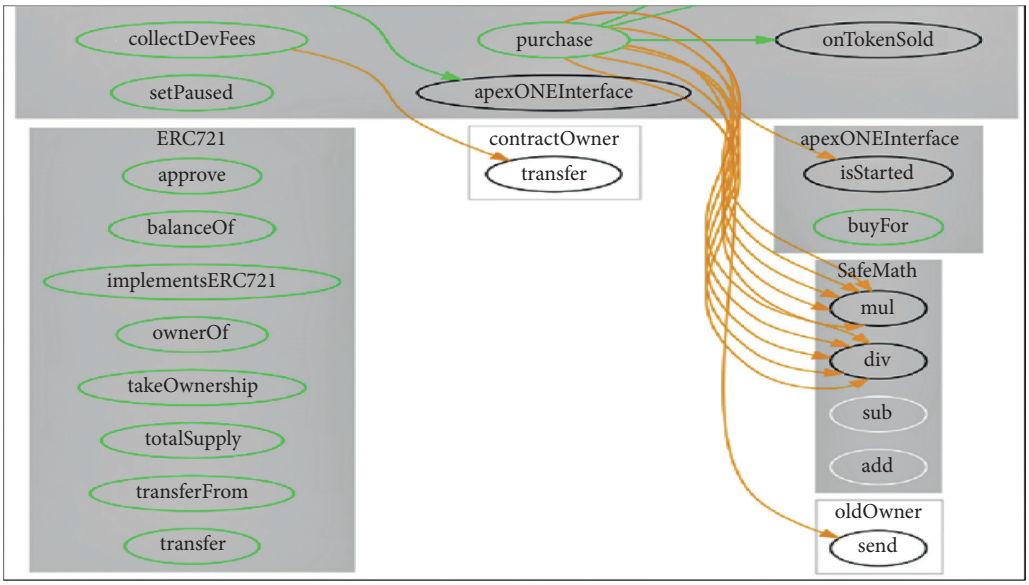


Figure 7: Information flow between different functions of the smart contract.

```
function withdraw(uint amount) public{
    if (credit[msg.sender]>= amount) {
      require(msg.sender.call.value(amount)());
      credit[msg.sender]-=amount;
    }
  }
```

Figure 8: Simple DAO withdraw function.

```
function withdraw(uint amount) public {
    if (credit[msg.sender]>= amount) {
      credit[msg.sender]-=amount;
      require(msg.sender.call.value(amount)());
    }
  }
```

Figure 9: Simple fixed DAO withdraw function.

contracts, insecure usage of SelfDestruct instruction, and checking the return value of message call. We have provided a context-aware solution to detect the vulnerable smart contract variables. Hence, we checked and compared the vulnerable keywords and patterns from its Control Flow Graph and Data Dependency Graph and see that whether it is exploiting the best practices of smart contract or these keywords/patterns are the actual requirement of business logic of state variables. This gives us accurate true positives about detected vulnerabilities as per standards defined by the Ethereum community.

In Algorithm 1, first, we give the .sol file (smart contract) to the SESCon tool, which extracts its Abstract Syntax Tree (AST) and then converts it to the XML parse tree. From lines 5 to 10, required parameters/variables and functions are extracted which may or may not affect the state of a smart

contract. In line 11, the standard sample patterns of vulnerable contracts are collected from the local repository. Then (at line 12) each pattern is compared with the target contract under test. If the pattern is found (line 13), it is added in detectedVulList (line 14) and its occurrence is stored in the source code of the contract (line 15). At the end (line 18), a standard report is generated which shows the title of vulnerability, its description, and its solution to correct.

With the help of this algorithm, we can check not only the vulnerable keywords but also the ability to detect vulnerable patterns, intelligently. Our algorithm is aware of all contract flows graphs, i.e., money flow graph (using payable functions), dependency graph of state variables, and local variables along with their tainted values. This makes our algorithm more accurate than the state-of-the-art tools.

5. Experimental Results

In this section, we have presented our experimental results. First, we present some statistics of our dataset of contracts, then we show the results that we have achieved on testing samples provided by the Ethereum community for vulnerability detection. In the end, we have shown our results against real contracts.

5.1. Statistics of the Dataset of Smart Contracts. Dataset of our contracts under static analysis can be categorized as standard samples defined by the Ethereum community [14] and real contracts. However, here we only focused on real contracts to describe their statistics. So, these real contracts can be viewed as balance of contract and line of source code. Regarding balance, we have observed that most of the contracts have zero balance. The remaining nonzero contracts and their balances (Ethers) distribution can be seen in Figure 10. Only one contract has more than one million Ethers. In the case of lines of source codes, smart contracts range from 20 lines to 3500 lines. We have also seen that most (70%) of the contracts have compiler version 4.x, 25% contracts are compiled with 5.x version, and a very few number of contract (5%) were with compiler version of 6.x (see Figure 11). For some tools (Slither, Securify, etc.), if the contract has version 0.4.x and we have installed 0.5x compiler, then their tool fails at an initial stage. Therefore, they are dependent on the compiler version. However, our tool is not dependent on the compiler version; it just takes the source code of the smart contract and then starts analyzing for vulnerabilities.

5.2. Testing against Standard Vulnerability Samples. Vulnerabilities patterns in smart contracts have been defined [14] by experts from the Ethereum community. We have tested the state-of-the-art tools (SmartCheck, Solhint, Securify, Slither) on these provided standard samples and compared our results with these state-of-the-art tools. The summary is presented in Table 1, where "yes" means vulnerabilities are detected accurately, "no" means they are not detected, and "partially" means they are detected with some false positive. Overall SESCon has shown outstanding performance among the state-of-the-art tools, and almost 90% of defined vulnerability patterns are detected, while other tools could not reach more than 55% (see Figure 12).

To evaluate the performance of our proposed approach toward detection of vulnerabilities in sample standard contracts, we have used three evaluation metrics, namely, precision, recall, and $F1$-score, which are normally used in pattern recognition [43]. So in our case, they are defined as follows: precision is the proportion of sample standard smart contracts that are correctly detected as vulnerable among all the sample vulnerable smart contracts.

$$\text{Precision} = \frac{\text{smart contracts correctly detected as vulnerable}}{\text{all smart contracts detected as vulnerable}}. \tag{1}$$

Recall, in our case, is the proportion of sample standard smart contracts that are correctly detected as vulnerable among all the sample really vulnerable smart contracts.

$$\text{Recall} = \frac{\text{smart contracts correctly detected as vulnerable}}{\text{all really vulnerable smart contracts}}. \tag{2}$$

There is another method with which we can test our accuracy of results, that is, the $F1$-Score (F-score or F measure). This is a harmonic means of our precision results and recall results and measures by the following formula:

$$F1 - \text{score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{3}$$

Here, we summarized our evaluation of the accuracy of results against standard vulnerabilities (Table 2).

The reason behind the same values for precision, recall, and F1-Score against each tool is that, in our dataset, we have only standard sample smart contracts that are already defined as "really" vulnerable by the Ethereum community. In this dataset, there is no smart contract that is not "really" vulnerable. To achieve realistic results, we have tested our tool against real contracts and we measured the accuracy against real contracts. Then, we obtained the understandable results as presented in Table 3.

5.3. Testing against Real Contracts. We have also tested our tool against more than 8000 real contracts which we have downloaded from Etherscan [44] using JSoup API [45]. Most of the downloaded contracts (85%) have a balance of zero, and the remaining contracts have a balance of more than 1 Ethers and up to 1 million Ethers. SESCon has analyzed 8125 contracts, and most of the contracts have shown vulnerabilities, which are shown in Figure 13.

The main reason behind the exceptional results is that we have designed our solution in light of standard vulnerability patterns [14] and followed their suggested solutions to generate standard vulnerable patterns. This will help us to detect the vulnerability patterns in real-world smart contracts with great accuracy, while in other solutions, they have devised their own patterns and their tools detect them accordingly without considering the standard vulnerable patterns.

```
(1) Read the smart contract
(2) Extract the abstract syntax tree
(3) Convert AST to XML path using XPath queries
(4) Store locations of each statement which L1, L2, ... Ln
(5) Get control flow graphs (cf1, cf2)
(6) Get dependency graph, dg1, dg2
(7) Get local variable (lv1, lv2)
(8) Get state variable (sv1, sv2, \enleadertwodots svn)
(9) Get payable function (pf1, pf2)
(10) Get nonpayable function (npf1, npf2, ...\enleadertwodots npfn)
(11) Load standard patterns of vulnerabilities p1, p2, ..., pn
(12) for each (pi) compare dgi in given smart contract do
(13)     if foundPattern then
(14)         detectVulList.add (pi)
(15)         locationsList.add (Li)
(16)     end if
(17) end for
(18) Generate report
```

Algorithm 1: Detecting vulnerabilities in the smart contract using SESCon.

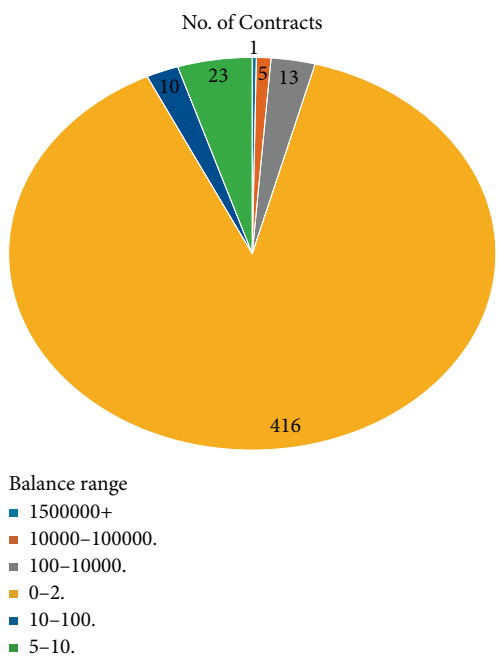


Figure 10: Balance in Ethers distribution in contracts.

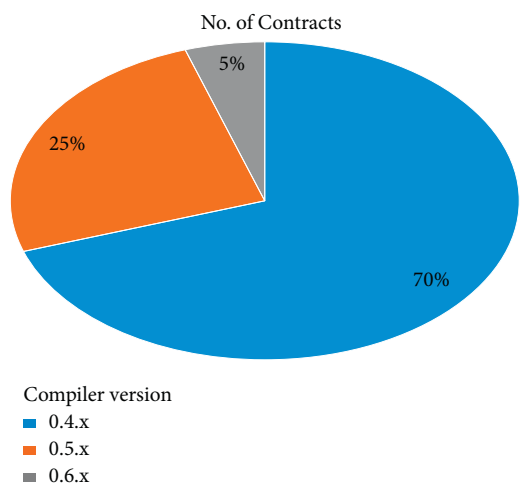


Figure 11: Compiler versions' distribution.

Another reason is that some vulnerability needs an in-depth analysis of smart contract code and its context. This means that only vulnerable patterns or keywords detections are not enough to report vulnerabilities, but we have to consider where the vulnerable code is present. This is because, sometimes, a vulnerable pattern is not causing vulnerability, if the said pattern existed in source code along with some security precautions; hence, it should not be reported.

5.4. Evaluation and Limitation. To validate our tool, we give a sample vulnerable smart contract from standard patterns [14] and check whether the SESCon can detect it or not, and our tool gave 100% accurate true positive detection. Then we give a fixed version of the vulnerable smart contract to SESCon, which shows 100% accurate true negative detection. After that, we have tested our tool on real smart contracts. During testing our tool against real contracts, it is revealed that still a lot of contracts have well-known vulnerabilities, namely, DAO, Transaction Order Dependency (TOD), trx.origin, and block.timestamp, and uses some assembly code. Among them, DAO type vulnerabilities were at the top and tx.original was the second most discovered vulnerability.

The main limitation with our tool is that it performs static analysis against standard vulnerabilities patterns defined by the Ethereum community; therefore, we cannot detect zero-day exploits. Due to the unavailability of samples of contracts for certain standard vulnerabilities patterns, we have also some false positives, which we are trying to overcome in our future work. Since our tool is

Table 1: Comparison of vulnerability detection.

Vul. ID [14]	Solhint [42]	Securify [9]	Slither [11]	SmartCheck [13]	SESCon
SWC-100	Yes	Yes	Yes	Yes	Yes
SWC-101	No	Yes	Yes	No	Yes
SWC-102	Yes	No	Yes	No	Yes
SWC-103	Partially	Yes	Yes	Yes	Yes
SWC-104	Yes	Yes	Yes	Yes	Yes
SWC-105	Yes	Yes	Yes	No	Yes
SWC-106	Yes	Yes	Yes	No	Yes
SWC-107	Partially	Yes	Yes	Partially	Yes
SWC-108	Yes	Yes	Partially	Yes	Yes
SWC-109	No	Yes	No	No	Yes
SWC-110	No	No	Yes	No	Yes
SWC-111	Yes	No	Yes	Yes	Yes
SWC-112	Partially	Yes	Yes	Partially	Yes
SWC-113	Yes	Partially	No	Yes	Yes
SWC-114	Partially	Yes	Yes	Partially	Yes
SWC-115	No	Yes	No	Yes	Yes
SWC-116	Yes	Yes	Partially	No	Yes
SWC-117	No	No	No	No	Yes
SWC-118	No	No	Yes	No	Yes
SWC-119	No	Partially	Yes	No	Yes
SWC-120	Yes	No	No	No	Yes
SWC-121	No	Partially	Yes	No	Yes
SWC-122	No	No	Yes	No	Yes
SWC-123	No	No	Yes	No	Yes
SWC-124	No	Yes	No	Yes	Yes
SWC-125	No	Partially	No	No	Yes
SWC-126	No	Partially	Yes	No	Yes
SWC-127	Partially	No	Yes	Partially	Yes
SWC-128	No	No	Partially	Partially	Yes
SWC-129	Partially	Partially	No	No	Yes
SWC-130	No	Yes	Partially	No	Yes
SWC-131	Partially	No	Yes	No	Yes
SWC-132	Partially	Yes	Yes	No	Yes
SWC-133	No	No	Partially	No	Partially
SWC-134	Partially	No	Partially	No	Partially
SWC-135	No	Partially	No	No	Partially
SWC-136	No	No	No	No	Partially

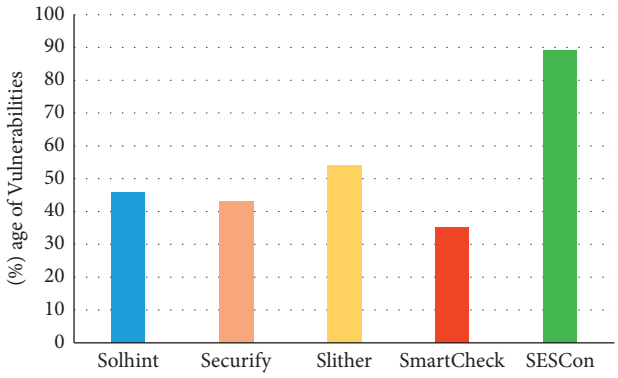


Figure 12: Testing against standard vulnerability samples.

dependent on the source code of the smart contract, therefore, it cannot be valid and applied to the bytecode of the smart contract. There are also some works [46, 47], which claimed that their tools can be used to evaluate the existing static analyzer tools for Ethereum smart contracts. We have tried to incorporate our tool into their work and evaluate our tool, but due to some technical errors during building their work, we failed to integrate

Table 2: Accuracy results against standard vulnerabilities.

Models	Precision	Recall	$F1$-score
Solhint [42]	0.5263	0.5263	0.5263
Securify [9]	0.6956	0.6956	0.6956
SmartCheck [13]	0.6153	0.6153	0.6153
Slither [11]	0.7777	0.7777	0.7777
SESCon	0.8918	0.8918	0.8918

Table 3: Accuracy results against standard vulnerabilities.

Vul. ID	Precision	Recall	$F1$-score
SWC-100	0.880	0.961	0.919
SWC-101	0.880	0.961	0.919
SWC-102	0.880	0.961	0.919
SWC-103	0.880	0.961	0.919
SWC-104	0.880	0.961	0.919
SWC-105	0.880	0.961	0.919
SWC-106	0.880	0.961	0.919
SWC-107	0.880	0.961	0.919
SWC-108	0.880	0.961	0.919
SWC-109	0.880	0.961	0.919
SWC-110	0.880	0.961	0.919
SWC-111	0.880	0.961	0.919
SWC-112	0.880	0.961	0.919
SWC-113	0.880	0.961	0.919
SWC-114	0.880	0.961	0.919
SWC-115	0.880	0.961	0.919
SWC-116	0.880	0.961	0.919
SWC-117	0.880	0.961	0.919
SWC-118	0.880	0.961	0.919
SWC-119	0.880	0.961	0.919
SWC-120	0.880	0.961	0.919
SWC-121	0.880	0.961	0.919
SWC-122	0.880	0.961	0.919
SWC-123	0.880	0.961	0.919
SWC-124	0.880	0.961	0.919
SWC-125	0.880	0.961	0.919
SWC-126	0.880	0.961	0.919
SWC-127	0.880	0.961	0.919
SWC-128	0.880	0.961	0.919
SWC-129	0.880	0.961	0.919
SWC-130	0.880	0.961	0.919
SWC-131	0.880	0.961	0.919
SWC-132	0.880	0.961	0.919
SWC-133	0.880	0.961	0.919
SWC-134	0.880	0.961	0.919
SWC-135	0.880	0.961	0.919
SWC-136	0.880	0.961	0.919

them. Further, their evaluation criteria are different from ours: we have made patterns to detect vulnerability while they just injected bugs and then detect them. We will consider them [46, 47] to investigate and evaluate our tool in the future.

5.5. Implementation. We have implemented SESCon in Java 8 and for lexical and syntactical analysis we have used ANTLR 4.8 and Solidity grammar 5.6 on Windows 10 64-bit platform on the core of i7 with RAM 4 GB. To find simple vulnerabilities patterns and keywords, we have used XPath 2.0 queries.

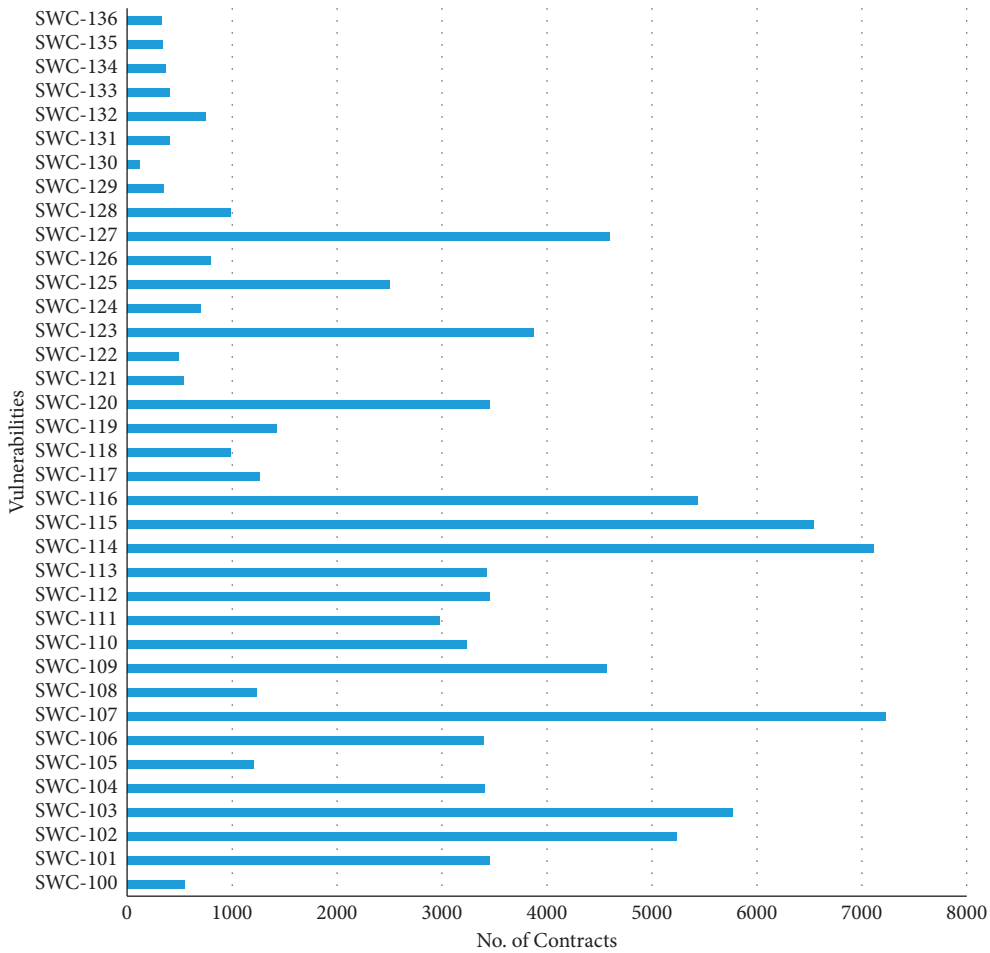


Figure 13: Testing against real contracts.

6. Conclusion and Future Work

In this paper, we have presented a solution to detect smart contract vulnerability through static analysis. Our solution is based on XPath and taint analysis. Most of the static analysis tools for smart contract give a large number of false positives. We have reduced such alarms with a hybrid approach of a combination of XPath and taint analysis. First, we convert a .sol file to its equivalent AST XML parse tree and apply the XPath query to find some simple vulnerabilities patterns. After that, we go through deep analysis by taint analysis techniques, where we have extracted state variable, local variable, their control flow, graph dependency of function, and payable and nonpayable functions to make some vulnerable patterns. Then we compare smart contract under test against the standard vulnerabilities patterns defined by the Ethereum community. Our tool outperforms other analyzers and can detect up to 90% of the known vulnerability patterns, accurately. We have also analyzed more than 8000 real contracts through our tool SESCon and found that a large number of vulnerable smart contracts still existed, which could be corrected by our tool. Our solution will provide a foundation toward the standardization in comparing and evaluating tools with standards vulnerabilities patterns.

Our work can be extended in many directions. The natural extension of our work is to reduce some false positives against the last four vulnerabilities (SWC-133 to 136) patterns. As we have performed static analysis on the Solidity source file, our next work would be to detect vulnerabilities analysis from the bytecode of smart contract. We are also planning how we could detect some zero-day exploits by some machine learning approaches. In the end, after consolidating the tool, we will provide our solution as an open source after some enhancements, so that early-stage researchers can be facilitated.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," *Decentralized Business Review*, 2008, https://bitcoin.org/bitcoin.pdf.

[2] V. Buterin, "A next-generation smart contract and decentralized application platform," 2014.

[3] The Dao –2016: https://etherscan.io/address/%200xbb9bc244d798123fde783fcc1c72d3bb8c189413.

[4] Parity attack: https://www.cnbc.com/2017/11/08/accidental-bug-may-have-frozen-280-worth-of-ether-on-parity-wallet.html>.

[5] The King of Ether - https://www.kingoftheether.com/postmortem.html>.

[6] B. Jiang, Y. Liu, and W. K. Chan, "Contractfuzzer: fuzzing smart contracts for vulnerability detection," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, New York, NY, US, September 2018.

[7] L. Luu, D. H Chu, and H. Olickel, P. Saxena and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, Vienna, Austria, October 2016.

[8] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, "ZEUS: analyzing safety of smart contracts," in *Proceedings of the Network and Distributed System Security Symposium NDSS*, San Digeo, CA, USA, February 2018.

[9] P. Tsankov, A. Dan, D. D. Cohen, A. Gervais, F. Bunzli, and M. Vechev, "Securify: practical security analysis of smart contracts," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Toronto, Canada, October 2018.

[10] J. He, M. Balunović, N. Ambroladze, P. Tsankov, and M. Vechev, "Learning to fuzz from symbolic execution with application to smart contracts," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, London, UK, November 2019.

[11] J. Feist, G. Grieco, and A. Groce, "Slither: a static analysis framework for smart contracts," in *IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, May 2019.

[12] S. Wang, C. Zhang, and Z. Su, "Detecting nondeterministic payment bugs in Ethereum smart contracts," *Proceedings of the ACM on Programming Languages*, vol. 3, pp. 1–29, 2019.

[13] S. Tikhomirov, E. Voskresenskaya, and I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov, "Smartcheck: static analysis of ethereum smart contracts," in *Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*, Gothenburg, Sweden, May 2018.

[14] Vulnerabilities Patterns in Smart Contracts- https://swcregistry.io/docs/SWC-100.

[15] J. Newsome and D. X. Song, "Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software," *NDSS*, vol. 5, pp. 3-4, 2005.

[16] Y. Chinen, N. Yanai, J. P. Cruz, and S. Okamura, "RA: Hunting for re-entrancy attacks in ethereum smart contracts via static analysis," in *Proceedings of the 2020 IEEE International Conference on Blockchain (Blockchain)*, IEEE, Rhodes Island, Greece, November 2020.

[17] Y. Xue, M. Ma, Y. Lin, Y. Sui, J. Ye, and T. Peng, "Cross-contract static analysis for detecting practical reentrancy vulnerabilities in smart contracts," in *Proceedings of the 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, September 2020.

[18] S. Schmeelk, B. Rosado, and P. E. Black, "Blockchain smart contracts static analysis for software assurance," *Lecture Notes in Networks and Systems*, vol. 284, pp. 881–890, 2021.

[19] N. He, R. Zhang, H. Wang et al., "EOSAFE: security analysis of EOSIO smart contracts," in *Proceedings of the 30th USENIX Security Symposium*, vol. 21, USENIX Security, Vancouver, Canada, August 2021.

[20] N. Ashizawa, N. Yanai, J. P Cruz, and S. Okamura, "Eth2Vec: learning contract-wide code representations for vulnerability detection on ethereum smart contracts," in *Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure*, Hong Kong, China, May 2021.

[21] N. F. Samreen and M. H. Alalfi, "Smartscan: an approach to detect denial of service vulnerability in ethereum smart contracts," 2021, https://arxiv.org/abs/2105.02852.

[22] K. L. Narayana and K. Sathiyamurthy, "Automation and smart materials in detecting smart contracts vulnerabilities in blockchain using deep learning," *Materials Today: Proceedings*, 2021.

[23] I. Grishchenko, M. Maffei, and C. Schneidewind, "EtherTrust: sound static analysis of ethereum bytecode," Technical Report D, Technische Universität Wien, Vienna, Austria, 2018.

[24] A. Permenev, D. Dimitrov, P. Tsankov, D. D. Cohen, and M. Vechev, "Verx: safety verification of smart contracts." in *Proceedings of the 2020 IEEE Symposium on Security and Privacy, SP*, San Francisco, CA, USA, May 2020.

[25] L. Brent, A. Jurisevic, M. Kong et al., "Vandal: a scalable security analysis framework for smart contracts," 2018, https://arxiv.org/abs/1809.03981>.

[26] C. Schneidewind, I. Grishchenko, M. Scherer, and M. Maffei, "Ethor: practical and provably sound static analysis of ethereum smart contracts," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, November 2020.

[27] J. Schiffl, M. Grundmann, M. Leinweber, O. Stengele, S. Friebe, and B. Beckert, "Towards correct smart contracts: a case study on formal verification of access control," in *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, New York, NY, USA, June 2021.

[28] E. Lai and W. Luo, "Static analysis of integer overflow of smart contracts in ethereum," in *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*, Nanjing China, January 2020.

[29] 2021 https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html> Accessed on: 11.08.2021.

[30] 2021 https://docs.chainstack.com/blockchains/quorum> Accessed on: 11.08.2021.

[31] 2021 https://www.multichain.com/developers/permissions-consensus/> Accessed on: 11.08.2021.

[32] 2021 https://docs.corda.net/docs/corda-os/4.8.html Accessed on: 11.08.2021.

[33] J. Polge, J. Robert, and Y. Le Traon, "Permissioned blockchain frameworks in the industry: a comparison," *Ict Express*, vol. 7, no. 2, pp. 229–233, 2021.

[34] M. Szydlo, "Merkle tree traversal in log space and time," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, May 2004.

[35] Solidity -https://solidity.readthedocs.io/en/latest/>.

[36] Vyper -https://vyper.readthedocs.io/en/latest/>.

[37] S. Sayeed, H. Marco-Gisbert, and T. Caira, "Smart contract: attacks and protections," *IEEE Access*, vol. 8, pp. 24416–24427, 2020.

[38] A. Johnson, L. Waye, S. Moore, and S. Chong, "Exploring and enforcing security guarantees via program dependence graphs," *ACM SIGPLAN Notices*, vol. 50, no. 6, pp. 291–302, 2015.

[39] X. Li, L. Wang, Y. Xin, Y. Yang, and Y. Chen, "Automated vulnerability detection in source code using minimum

intermediate representation learning," *Applied Sciences*, vol. 10, no. 5, p. 1692, 2020.
[40] 2020 Antlr–https://www.antlr.org/> Accessed on 20-06-2020.
[41] 2020 XPath-https://www.w3.org/TR/xpath20/> Accessed on 20-06-2020.
[42] 2020 Solhint: A solidity linting tool - https://github.com/protofire/solhint> Accessed on 20-06-2020.
[43] Precision and recall-https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/>.
[44] 2020 EtherScan APIs-https://etherscan.io/apis> Accessed on 20-06-2020.
[45] 2020 JavaSoup Library-https://jsoup.org/%20Accessed on 20-06-2020.
[46] A. Ghaleb and K. Pattabiraman, "How effective are smart contract analysis tools? evaluating smart contract static analysis tools using bug injection," in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, New york, NY, USA, July 2020.
[47] A. L. Vivar, A. L. Sandoval Orozco, and L. Javier García Villalba, "A security framework for Ethereum smart contracts," *Computer Communications*, vol. 172, pp. 119–129, 2021.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 6184756, 17 pages
https://doi.org/10.1155/2021/6184756

Research Article

Real-Time Surveillance Using Deep Learning

Muhammad Javed Iqbal,[1] Muhammad Munwar Iqbal,[1] Iftikhar Ahmad,[2] Madini O. Alassafi,[2] Ahmed S. Alfakeeh,[2] and Ahmed Alhomoud[3]

[1]*Department of Computer Science, University of Engineering and Technology, Taxila 47050, Pakistan*
[2]*Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia*
[3]*Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia*

Correspondence should be addressed to Muhammad Munwar Iqbal; munwariq@gmail.com

Received 4 July 2021; Accepted 10 August 2021; Published 17 September 2021

Academic Editor: Farhan Ullah

It is crucial to ensure proper surveillance for the safety and security of people and their assets. The development of an aerial surveillance system might be very effective in catering to the challenges in surveillance systems. Current systems are expensive and complex. A cost-effective and efficient solution is required, which is easily accessible to anyone with a moderate budget. In aerial surveillance, quadcopters are equipped with state-of-the-art image processing technology that captures detailed photographs of every object underneath. A quadcopter-based solution is proposed to monitor desired premises for any unusual activities, like the movement of persons with weapons and face detection to achieve the desired surveillance. After detection of any unusual activity, the proposed system generates an alert for security personals. The proposed solution is based on quadcopter surveillance and video streaming for anomaly detection in the received video streams through deep learning models. A well-known FasterRCNN algorithm is modified for fast learning with feature reduction in the initial feature extraction step. Five different kinds of CNNs were evaluated for their ability to identify objects of interest in surveillance images. ResNet-50–based FasterRCNN with the highest average precision performed as an excellent solution for threat detection. The average precision of the system is 79% across all categories achieved.

1. Introduction

These days security is the critical concern of everyone around the globe. Every citizen wants to cease the possibility of any upcoming threat to himself or his possessions. This tremendous escalation in the requirement of security needs for individuals and an organization needs a proper and up-to-date surveillance system. The existing systems are expensive and complex to operate. Hence, a system that can account for insecurities regarding security and of less complexity is the need of time. A cost-effective, simple, and efficient surveillance system is necessary for every organization, either in the public sector or private sector, for internal monitoring and the analysis of post events later onwards [1]. The drone is globally recognized as an aerial vehicle without any pilot on board. They are known for their use in numerous diverse fields, namely, Security, Aerial Photography, Surveying, and Surveillance. This drone-based security surveillance system provides the user ultimate control, providing surveillance updates directly to the user's owned systems. Using a quadcopter can conveniently start the surveillance process in areas where we cannot place security cameras. Also, with the help of the quadcopter, we can deliver goods where human access is beyond the bounds of possibility, as in explosions, volcano eruptions, and numerous other disasters, thereby minimizing the risk to human life.

Doing remote surveillance is one of the major problems that Pakistan is confronting these days due to the unsettling political situation and banned outfits. The basic architecture of the working drone system is given in Figure 1. Adversaries had targeted our country since 1990. The target killing and bomb blasts seized the growth of many business sectors, leaving inhabitants to face devastation and inflation. A

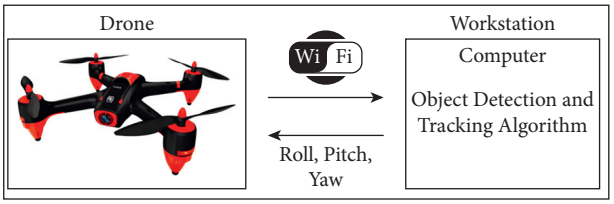


Figure 1: Basic architecture of drone surveillance system.

stranded economy is resultant due to losing thousands of precious lives and property worth millions of dollars. Unfortunately, remote surveillance has remained unaddressed so far. To facilitate the country in attaining proper security level requires deploying a number of security personnel, which is quite undesirable. This deployment will lay hold of the country's major resources, including human resources and finances.

Moreover, even by incorporating human resources, the surveillance issue still is alarming as staffing is unlikely to perform impregnable duties. We cannot incorporate individuals to fight against unenviable situations, including harsh, barren deserts, unbeatable oceanic waves, and blazing hot regions. Therefore, the idea here is to perform surveillance conveniently using drones because drones can undergo continuous and spot-on surveillance. In a nutshell, drones are the comprehensive and cheaper solution to all surveillance and security-related issues. One of the primary objectives behind developing this quadcopter security system was to assist security personnel [2]. However, the deployment of security personnel is not the ultimate solution to deal with security threats. An aerial surveillance system aids in advancing security procedures as real-time data are directly transmitted to user systems, including smartphones and computer servers. In addition to that, it continuously performs surveillance in open areas, especially in areas that are undesirable to human existence. This system helps the user to keep track of different angles of the spotted location. A dimensional view of the spotted area helps figure out minute details that lead to a more secure and hazard-free surrounding. We aim to provide the most appropriate and inexpensive solution while simultaneously deploying drones rather than personnel for surveillance, which is the most inexpensive solution to security-related issues.

Security is essential in every domain, be it public or private. For example, the increase in crime rate in a crowded event, be they concerts, airports, marriages, or other public gatherings or lonely areas where there are a lot of blind areas that cannot be monitored or are general blind spots in terms of security and public safety. Many solutions have been presented regarding the problems mentioned above. But none of them provides a clear and definitive solution to the issues regarding security and safety. Weapon detection or anomaly detection is one of the major problems most security institutions are trying to tackle every day. Weapon or anomaly detection system identifies abnormal events differently from existing patterns because any anomaly is a pattern that is different from a set of standard patterns. Weapon detection can be related to a mass shooting incident, a bank robbery, or any other more public disaster, or it can be in the form of concealed weapon detection in airports, luggage, or other forms of searches. Millions of dollars have been put into the weapon detection field, and many solutions have been created or implemented. One aim of the study was to provide computationally less expensive algorithmic arrangements for detection purposes. A precise ablation can chalk out the weak areas, but in our case, our intuition was based on the fact that researchers like SSD, YoloV3 etc., have proposed many other fast detectors and that reduction in the number of input features can reduce the processing time. The search for an ideal method is still going on. The proposed system provides a security framework to monitor the defined premises to alert the security team after identifying any unusual activity, including any suspicious individual or item in a locality. Afterwards, subsequent operations undergo ensuring effective crisis management with proficiency without any dilemma or restriction. The system under discussion provides feasibility in case of emergencies with the efficiency of overcoming crises in the first place. Consequently, providing users with means of dealing with disasters effectively and proficiently.

The main contributions of our study can be described as follows:

(i) Construction of data set that addresses four types of entities concerning drone surveillance

(ii) A less computationally expensive FasterRCNN discussed

(iii) A near Real-time performance for object detection

(iv) Four existing deep neural networks are also used and compared for their capability to categorize

This article is arranged as follows. Section 2 briefly discusses the background information and related literature review. Section 3 provides the material and methods used to implement this research work. Results and discussion are given in Section 4. The final Section 5 presents the conclusion of this work.

2. Literature Review

Currently, many robotic applications are being developed for doing tasks autonomously without any commands from a human. A system enabling a robot to surveillance, like detecting and tracking an object in motion, will carry us to a more advanced task. For taking the role of UAV (Unmanned Aerial Vehicle) AR [3], the drone comes in place as a flying robotic platform. As far as the implementation part is concerned, they developed an algorithm to detect and track an object by analyzing its shape and color. Unmanned Aerial Aircraft Systems (UAS) are being divided into different transportation engineering areas. Traffic analysis and work on the grounds are highly compatible with the systems. Such studies are helpful for understanding and working UAS in transportation. In the study by Berrahal et al. [2], the authors proposed a cooperative border surveillance solution that relies on WSN (Wireless Sensor Network), implemented to detect and track transfer and UAVs in the form of

quadcopters. A heuristic-based scheduling algorithm is used by increasing the rate of trespassers spotted by the quadcopters. Also, techniques to localize sensors using RFID technology and optimal computing positions and relay data between isolated islands of nodes were used for evaluating the performance of the mentioned WSN-based surveillance system. In another work [3], the authors started their studies by defining the capabilities of aerial surveillance systems and suggested several technologies that can be used in it. They also discussed how drones could be used in aerial surveillance. The work also found that surveillance can be utilized in peace retaining activities and time monitoring of a place at any time of the day. The intention was to provide speedy and green surveillance at a low-cost charge so that it could be used extensively at the nonpublic, institutional, and governmental stage.

Gadda and Patil [4] stated that security issues of borders are increasing with rapid tensions across the countries. Terrorist attacks at the border are risking the lives of common people living near the borders. Monitoring of terrorism activities and security lapses becomes quite challenging and challenging in changing weather conditions. Designing an UAV for monitoring the border area from long distance conveyed to the observer. In this study [5], the IR remote is used for controlling the quadcopter. The Audio-Visual will be transferred to the computer through a wireless camera. The work presented in this article focused on trends of image processing for UAV (Unmanned Aerial Vehicles) and quadcopter data. Examples for totalling the total number of vehicles, edge detection, and evasion algorithms were discussed. SIFT (scale-invariant features transform) matching and the median filter are mainly used in UAVs image processing and visualization of new urban buildings and mechanisms that control the quadcopter. In this work, a simple camera-based navigation system for an autonomous quadcopter was proposed. Authors observed that additional infrastructure like artificial landmarks and GPS is not required in navigation methods and algorithms [6]. Factors affecting the computational complexity were independent of the environment size and factors like sensing only one landmark at a time. FPGA-based embedded realization of the methods was discussed with the most computationally demanding phase. In the study by Haque et al. [7], they proposed quadcopter as a low-weight and low-cost autonomous flight Unmanned Aerial Vehicle (UAV) for delivering parcels ordered online. Usage of Google maps to navigate to destination are the significant functions of the quadcopter. The authors also discussed the capability of delivering orders through quadcopter using an online process and coming back to the original point or starting place.

The work by Ding et al. [8] presented that those earlier systems were expensive and limited to outdoor environments with access to Global Positioning Systems (GPS). This work aimed to develop a design for a quadcopter to create a map of an unknown space using commercial off-the-shelf (COTS) to reduce cost and for mapping SLAM (Simultaneous Localization and Mapping) for creating an automatic map. Lastly, it was suggested that the SLAM algorithm might run efficiently on the Android phone. This work by Ding et al. [8] stated that applications in communications, photography, agriculture, and mainly surveillance make drones as mini-UAVs a bit more attractive. State-of-the-art studies for that amateur drone with a vision named Dragnet, tailoring the recent Cognitive Internet of Things framework. At the same time, the detection and classification of authorized and unauthorized drones are done, which eventually allow only authorized ones to fly over.

Amato et al. [9] discussed that Public security and safety are ensured through wireless communication technologies and support mobility. In the study by Alford et al. [10], the authors stated that the primary cause of security and critical information could be transmitted among the entities involved, dealing with cyber issues and mainly risking privacy of UAV equipped with communication hardware and directed to specific positions for privacy and safety. For the past few decades, Iraq and Syria among the most affected countries due to terrorism by ISIS were liberated, and internationally displaced persons (IDP) were going back to their homes. Infrastructure loss in the form of destroyed buildings and operators are exposed to undefined situations. Situations like this make innovations in the search and clearance of operational technology for ensuring safety and peace. In the study by Bangare et al. [11] focussed on ways to save homes from fire and smoke. The primary concern was home monitoring, appliances controlling, and door latches control from faraway regions.

With the use of their device, people will be able to manage our home appropriately from remote places. In the study of Munagekar [12], researchers focused on detecting places where criminal talk happens and tracing the robber quickly and efficiently. The purpose of this work was to minimize the efforts and finances of the security forces. Canny edge detection algorithm was used to develop this system, which was further used to detect and catch the robbers and stop unusual happenings. Usage of canny edge filter reduces the postprocessing size of objects, which saves much memory of hard disk. The focus of Alshammari and Rawat work was on a multicamera video system for the betterment of security [13]. The multicamera video system can make a widespread impact within the security industry. The technical aspect of this may help many people to counter the protection challenges of their everyday lives.

The proposed technique may locate and understand human behaviour from videos taken from cameras mounted at the wall. The proposed method consists of the detection and monitoring of any targets. In this article [14], a live surveillance system was proposed for weapon and abnormal events detection. The system was composed of three modules. The detection was done using the convolutional neural networks in the first processing module, whereas the second module manages guns and tracking. The alarm operations were done in the last processing module. In their work, shape detection and object detection techniques were experimented to find detection accuracy. The authors claimed that the proposed method substantially reduces the crime rate and the time required to trap the criminal. In this

article, Mahalanobis et al. [15] presented autodetection and monitoring strategies based on three significant aspects. Three different aspects of this work were the automatic goal detection and reputation strategies with monitoring, the handover, and smooth tracking of objects within a community and the improvement of actual-time communication and messaging protocols using COTS networking components.

The comprehensive review of the various security and surveillance techniques shows that this area has the potential to work. The mechanism of surveillance can be enhanced using modern deep learning approaches. This work proposed a technique for the detection of arms and other objects using advanced deep learning models.

3. Materials and Methods

This work proposed a methodology to detect various entities in a video. Figure 2 shows the block diagram of the methodology and phases involved in the proposed security and surveillance system. The proposed system consists of the following main phases: Data acquisition/collection of the data set containing the instances of entities to be detected. Next, the preprocessing phase transforms the images into reduced features for efficient processing and the Training Phase for model building. The fourth phase consists of Query Image entity prediction, in which query image is processed and presented to a trained model for the prediction of potential. Following is a detailed description of each phase.

3.1. Data Acquisition. The data acquisition phase is responsible for collecting and acquiring videos from surveillance drones to construct the required data set. The videos in the data set are obtained from different environments through a surveillance drone camera. The data set consists of videos of different persons with different poses, with and without weapons, etc. The persons include guards and common people. Later the captured videos are transformed into image frames, which are then taken from the data set for further processing. A good data set plays an important core role in the development of the proposed surveillance system. A machine learning model for the detection will be trained by giving them image frames from the developed data set. In the next phase, necessary preprocessing steps are applied to the image frames. The detail of the preprocessing step is as follows.

3.2. Preprocessing. In this preprocessing phase, several steps are taken to facilitate the model training process. The key steps include resizing the data set image frames, extracting image features using pretrained VGG16 CNN, and later PCA is applied on features extracted from the desired input image. The detail of each step within preprocessing is discussed next.

3.2.1. Resizing of Image Frames and Data set Construction. The original size of the captured videos/image frames was $2980 * 2021$ pixels is shown in Figure 3 and is captured from a hovering quadcopter. Image with such a large number of pixels is computationally expensive to be modelled to learn the presence of entities in images. So, the next natural step is to reduce the size of an image to make it computationally tractable. So, all the extracted images in the data set are resized to $600 * 800$ dimensions as shown in Figure 4. The presence of a data set is a key step for judging the quality of algorithms. For that purpose, we also built a data set consisting of 1805, 1700, 1600, and 1550 training images for objects like Face, Weapon, Guard, and Intruder, respectively. Similarly, 695, 550, 500, and 500 testing images are selected for Face, Weapon, Guard, and Intruder, respectively.

The images used for feature extraction should be in grayscale format. This step converts the images into grayscale. For example, Figure 5 shows the color image, whereas Figure 6 shows the same transformed image in the grayscale form.

3.2.2. Features Extraction from the Image Frames and PCA. After resizing step, we need to extract dense VGG16 features from the images [16]. The VGG-16 is a well-known pretrained CNN. VGG-16 model is trained on a subset of the ImageNet database of over one million images to classify images into 1 k categories. The data set images and their respective annotation files are given for features extraction. PCA is a feature or dimensionality reduction technique that transforms original features and produces a new set of variables, namely, principal components (PC) [17]. Each PCs is a linear combination of the original variables, which are orthogonal to each other. So finally, extracted features are reduced in size using the PCA method to decrease the computational burden on the training model [18].

Reduced features from the image frames and their corresponding entity's locations and category identification are needed to be learned by some ML or CNN-based algorithm. Traditionally, Viola-jones [19] algorithm is used by researchers for object detection. However, for our purpose, we have chosen the FasterRCNN object detector [20], which is better and well known for its applications in different fields like cars/pedestrians detection [21], cancerous cell type identification [22], and even Urdu-text detection [22]. The basic working of FasterRCNN is shown in Figure 7. They are a little different from standard types of shallow networks because they have two outputs. The two kinds of outputs are classification for a given entity, and the other is the bounding box (location coordinates) prediction. Figures 8 and 9 show the example of bounding boxes created in the MATLAB application for our surveillance data set.

Therefore, for training, extracted reduced features and corresponding are given to the Faster-RCNNmodel. The learning rate was tuned to 0.001, with "Adam" as learning optimizer, and the value of minibatch size was 1. The model is trained for a different number of epochs to see its learning capability for different entities. The model is trained and tested from 50 to 200 epochs. The key categories are identified and located in our proposed methodology based on Four (4) classes/categories. These four categories are

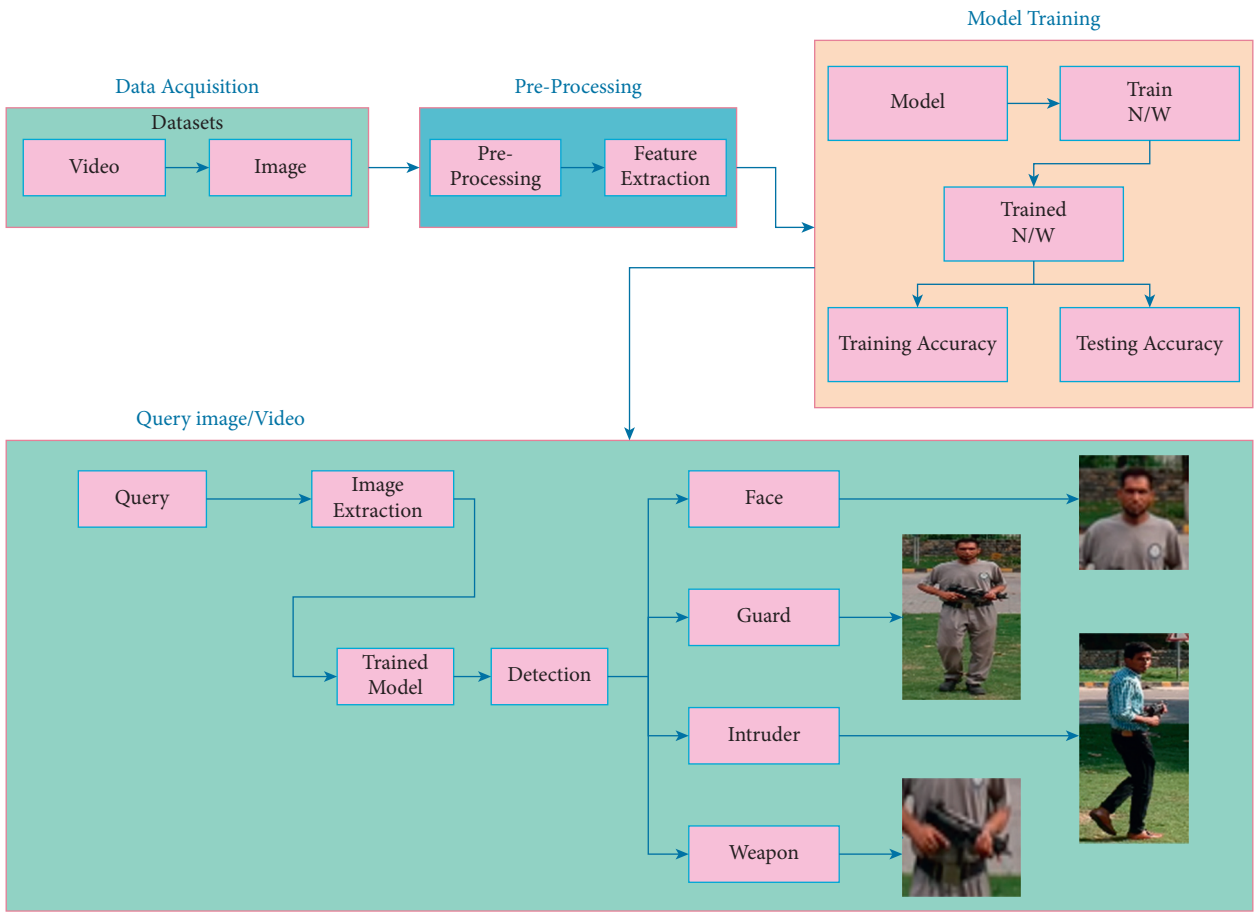


Figure 2: Block diagram and various phases of proposed methodology.

Face, Weapon, Intruder, and guard. Figure 10 shows how the 4 classes or categories are identified from a given image. The detection of weapons was at top priority, along with intruders. Based on our experiments, as shown in Table 1, we were successful regarding weapons but not best for an intruder. This shows that we further need to enhance our training and perhaps need to expand the data set. ConvNets are extensively used in deep learning [23] to solve various types of similar problems.

The biological structure inspires convolutional neural networks (ConvNet). A ConvNet comprises several layers, such as convolutional, max-pooling or average-pooling layers, and fully-connected layers. Ultimately, the trained model is used to enquire about the training and testing accuracy. Figure 11 presents the architecture of RCNN-based image region classification model. A trained detector was able to detect the Civilian, Guard, Person Face, and intruder.

For detection of the image, query videos and trained models are given for processing. It detects the images and shows the information of that image.

Figure 8 shows the label or ground truth of a civilian. Similarly, Figures 9 and 12 show the labels of guard and threat, respectively. These videos are labelled with the help of a ground truth labeller. FasterRCNN [24] model extracts potential regions for learning to detect whether a person is a civilian, guard, or a threat at later stages with this labelled data.

Generally, the Object training algorithm can be subdivided into two subparts. One is a feature extractor, whereas the other is a classification and regressor (i.e., bounding box predictor), as shown in Figure 7. Although objects are initially detected through a trained object detector, the final decision regarding the presence of objects is finalized using Algorithm 1.

3.3. Pretrained Neural Networks. The other 4 pretrained neural networks used as a feature extracted in our experiments are discussed, and explanations are given to understand their role in object categorization with key information about them.

3.3.1. SqeezeNet. SqueezeNet is a deep neural network that uses the concept of squeeze block and expands blocks [25]. It has 80.3% accuracy for ImageNet. SqueezeNet objective was to build a smaller neural network with a reduced number of parameters compared with other architectures. The SqueezeNet layer involves layers of fire modules and several max-pooling layers as shown in Figure 13. The network takes an input of 227×227 pixels and can classify 1 k categories.

Figure 3: Image with original size (2980 ∗ 2021).

Figure 4: Resized image (600 ∗ 800).

3.3.2. GoogleNet. GoogleNet is 144-layer pretrained CNN and takes in an image with the dimension of 224 × 224 pixels [26]. It introduced the concept of inception blocks in its architecture and is shown in Figure 14. It is the winner of the ILSVRC 2014 competition with a top-5 error rate of 6.67%. Nowadays, we can have 2 types of pretrained GoogleNet, one is trained on the ImageNet data set, and the other is trained on the Places365 data set.

3.3.3. ResNet-18. ResNet is residual network, and it works based on the concept of skip module [27]. ResNet-18 is a deep neural network (DNN) trained on more than 1 million images [25]. ResNet-18 originally has 18 layers, and its detailed architecture is described in Figure 15. It can categorize images into 1 k classes, enabling this network to learn rich feature representations of various objects. ResNet-18 expects input images of size 224 × 224 pixels.

3.3.4. ResNet-50. ResNet-50 is an extension of ResNet-18 and is trained on the ImageNet data set of more than 1 million images [28]. This network has 50 layers and can classify images into 1 k classes. ResNet-50 processes an input image with a dimension of 224-by-224 pixels.

In its architecture, as shown in Figure 16, builders of this network have further connected the layers by bypassing 2 layers, and this step continues till fully-connected layer (F).

Figure 5: Color image representation.

Thus, bilayer jumping and connection between layers can be considered a Network-In-Network model and a key intuition that improves accuracy.

4. Results and Discussion

We trained and tested our built model using 4-different type object detectors trained on 50, 100, 150, and 200 epochs to know how well our proposed model object detector methodology works. For each model corresponding to epochs, the average precision for each kind of entity is determined. Table 1 succinctly describes the experimental outcome. Although Figure 17 graphically describes the results, increasing epochs improve the detection precision of each entity. As a result, our model best detects weapons and faces.

On the other hand, a guard is the third most effectively identified entity, whereas the lowest results were far intruder. As intruders can be a person, but on the other hand, a guard is also the person. The only difference is the presence or absence of a weapon, which in our case seems tricky for the learning algorithm to pick. There is a subtle difference between guard and intruder, which can hopefully be overcome with more training examples and extending methodology.

The results of the study by Angelov et al. [29] were compared based on speed, accuracy, and memory usage. Figure 17 shows the results for Detection Average Precision for four categories. We have compared the proposed method with previous work [30] and Table 2 and Figure 18 show that the method mentioned earlier [28, 29] run faster to some extent with fewer memory requirements, but these produce false positives. Both methods proposed in earlier studies [28, 29] approximate the background as a plane, so they are too oversimplified, and the AP for various object categories is shown in Figure 19. The proposed method uses less resources, and its performance is comparable with others.

The number of objects per frame vs the percentage of frames in the training, validation, and testing sets for object detection in videos is described in Figure 20.

4.1. Impact of Different Convolutional Neural Networks Features. Finally, detailed experiments were carried out for each kind of intended object shown in Table 3. The experiments were carried out using reduced vgg16 features for training the FasterRCNN. Later combinations of different CNN features are used for identifying the impact of various models on the detection of various intended entities and the time taken to train each model.

The models are compared using hold out ratios and again 4-different well-established models like SqueezeNet, GoogleNet, ResNet-18, and ResNet-50. The comparison shows the consistency of the data set for different target categories.

4.2. Detailed Results of Training and Testing for Various Object Detectors. Object entities were divided into two subcategories of train and test sets. Train and test set images contained a different number of objects per category. Four main types of object detectors were trained and tested with different parameters as given in Table 3. Furthermore, the results have been graphically visualized in Figure 21. Different models show consistent APs across different categories except for SqueezeNet. The SqueezeNet model has a small parameter size and is the most efficient, as checked in the existing literature. In fact, there is a tradeoff between model size and efficiency.

The image issues of the blur, jaggedness, and other problems, as we are using deep learning models. It implicitly accepted that deep learning models efficiently address occlusion, blur, jaggedness, and other problems. That is why, we have not discussed those tasks that come directly under

Figure 6: Grayscale image representation.

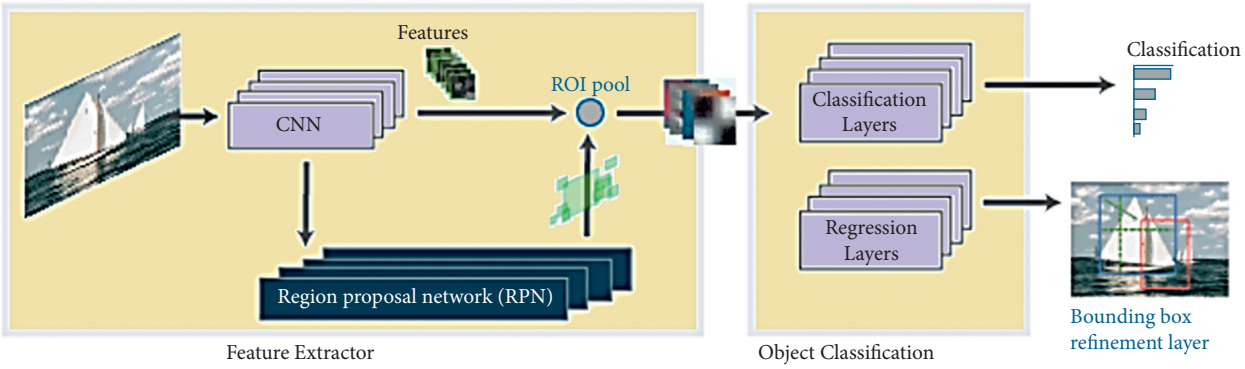


Figure 7: Fast R-CNN structure.

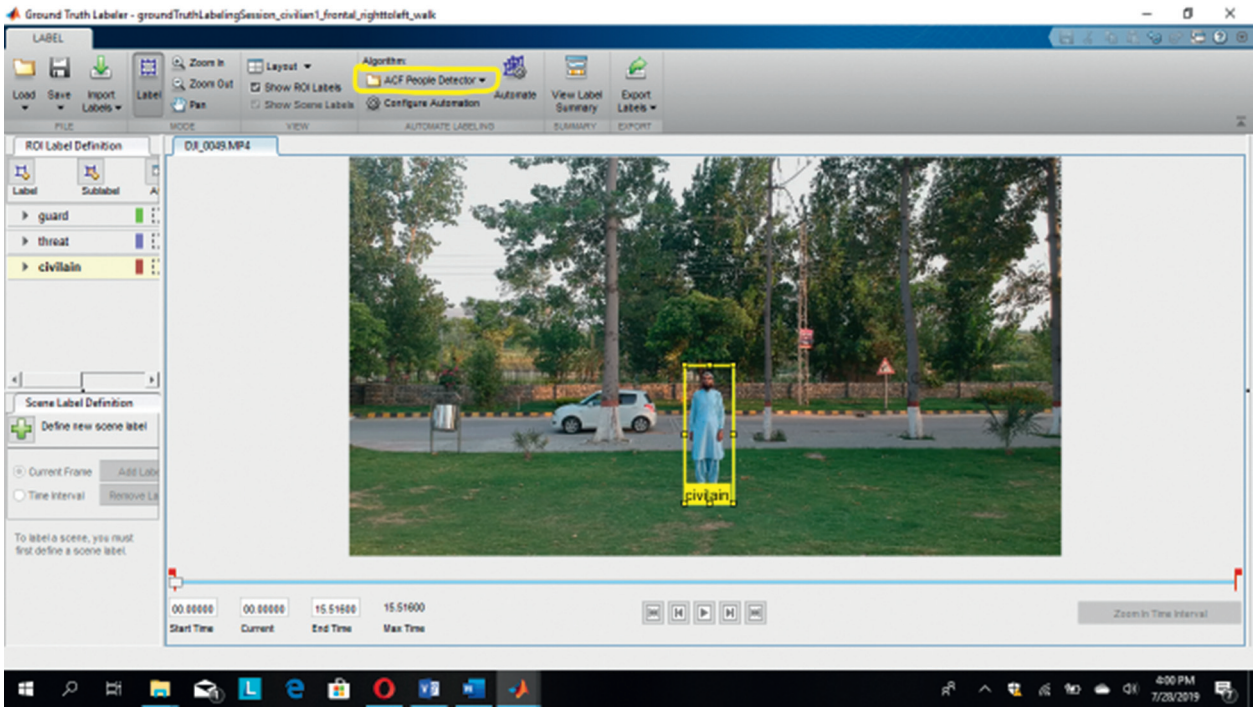


Figure 8: Ground truth labeller annotation of civilian.

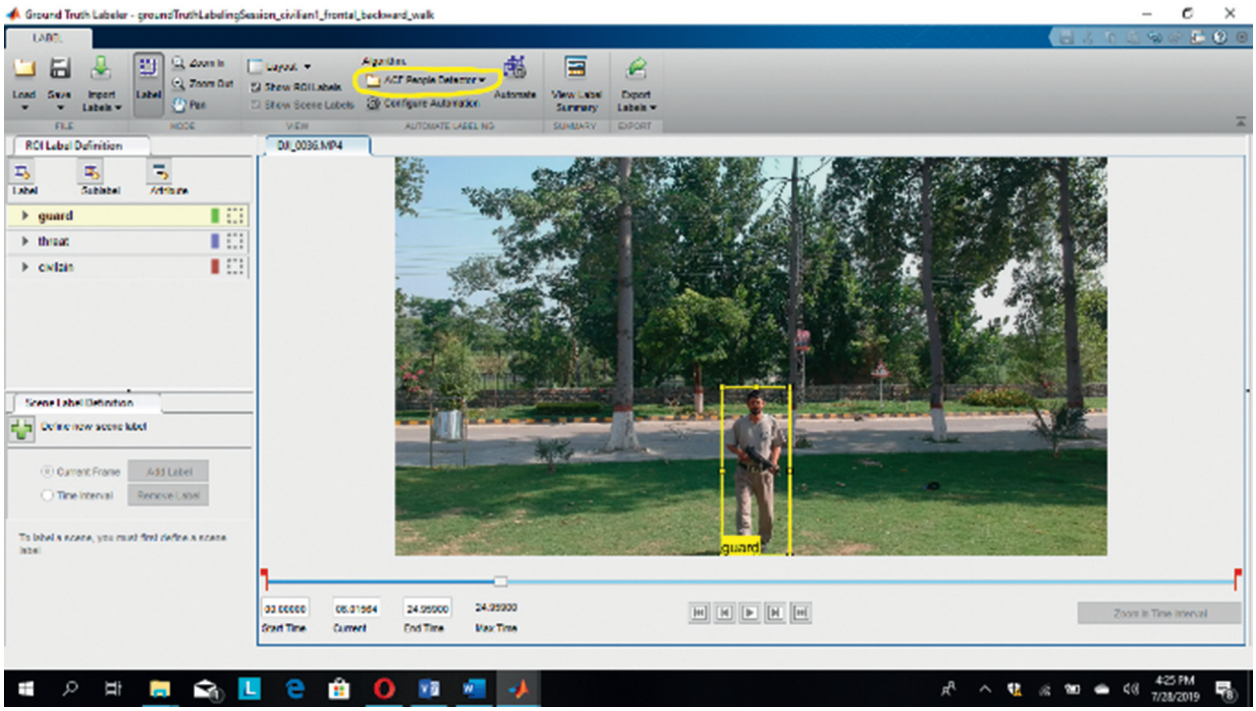


Figure 9: Ground truth labeller annotation of guard.

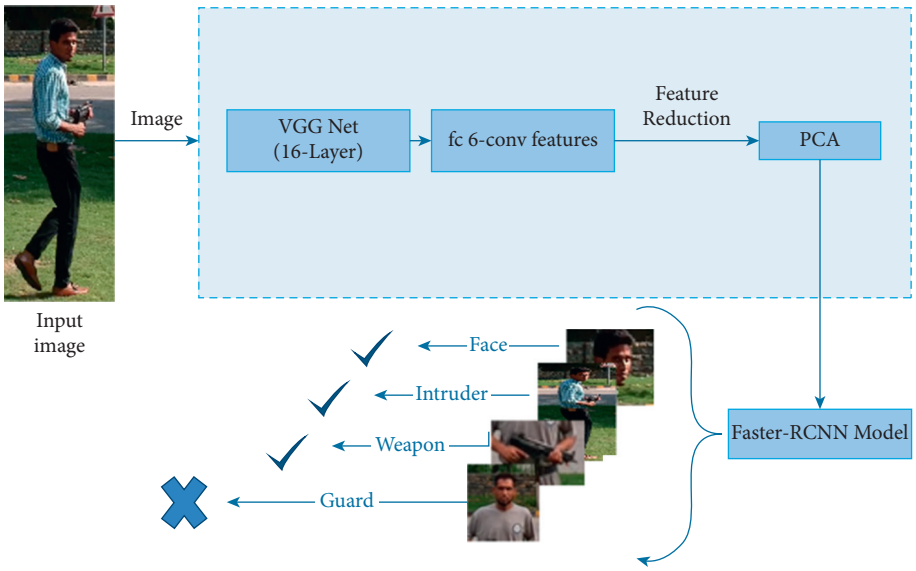


Figure 10: Reduced VGG16 features and detector model.

the umbrella of image processing. The experiments and results depict that the detection of the security surveillance using the drone is effective. The security surveillance is performed through deep learning architecture. CNN's Deep Learning architecture SqueezeNet, GoogleNet, ResNet-18, and ResNet-50 are used. ResNet-50, based on FasterRCNN, is performed excellent at the data set and surpassed the results generated.

Table 1: Detection average precision (A) for given entities.

Object type	AP @ 50 epochs	AP @ 100 epochs	AP @ 150 epochs	AP @ 200 epochs
Face	0.60	0.73	0.78	0.83
Weapon	0.40	0.50	0.58	0.90
Guard	0.66	0.69	0.71	0.76
Intruder	0.3	0.44	0.50	0.67

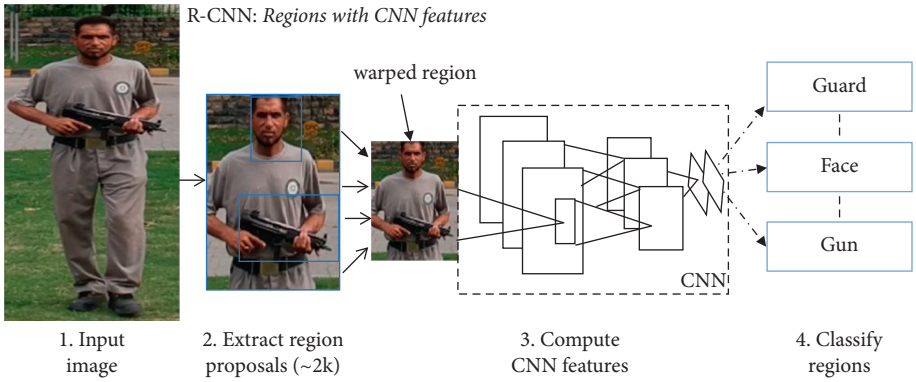


Figure 11: FasterRCNN-based image recognition model.

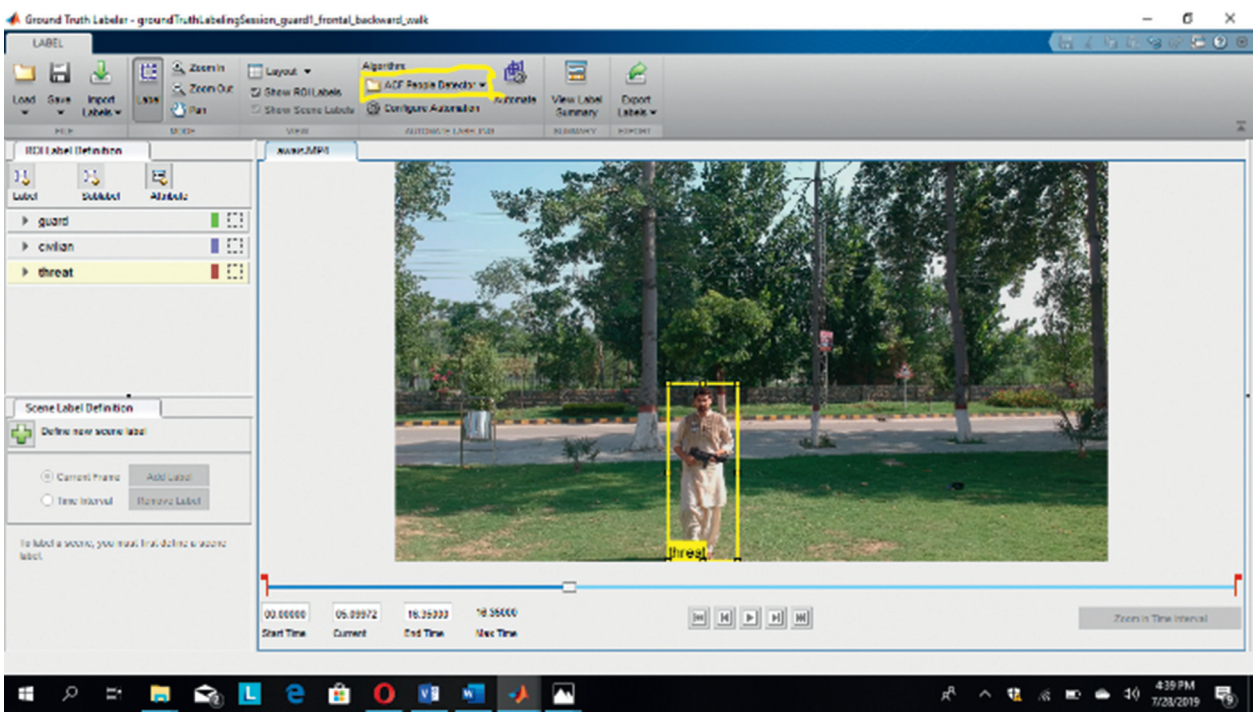


Figure 12: Ground truth labeller annotation of threat.

```
INPUT Image
OUTPUT Intruder Detected (Yes/No)
procedure INTRUDER
  for y = 1 to images_n do
    Predict candidate objects in y_n ⟶ [F, iND, W, G]
    if F == true && W == true && G! = true then
      iNDx_y = true, highlight area(s) in y_n
      interrupt for y_n to main system
    end if
    if W == true then
      Generate alert and keep-track of location in y_n
    end if
  end for
end procedure
```

Algorithm 1: Intruder detection algorithm.

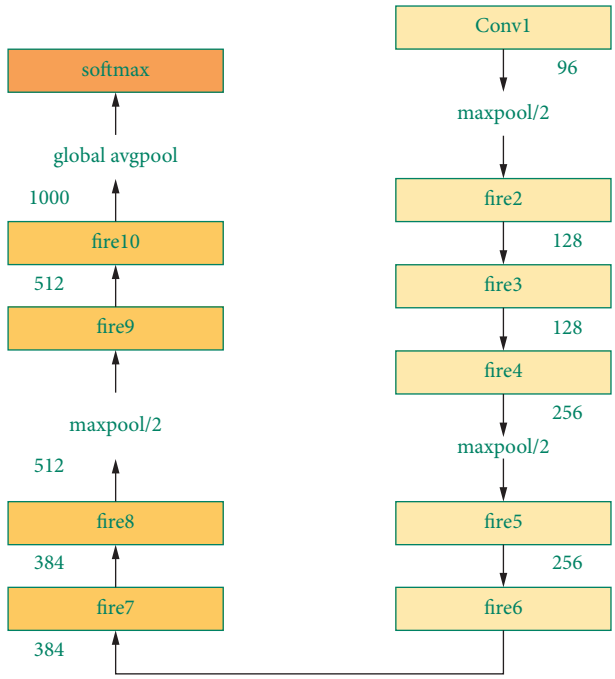


Figure 13: SqueezeNet architecture [25].

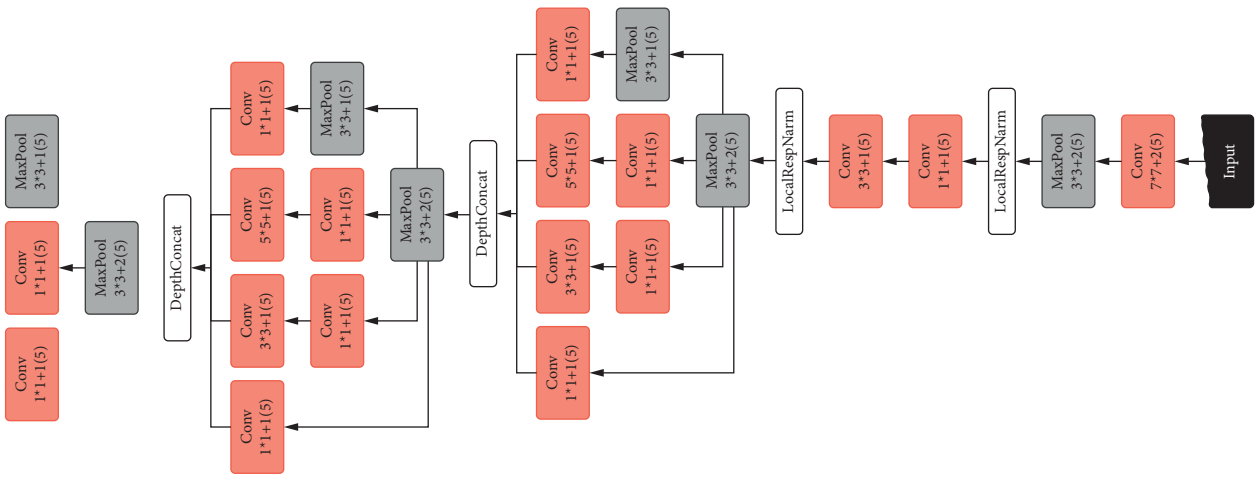


Figure 14: GoogleNet architecture [26].

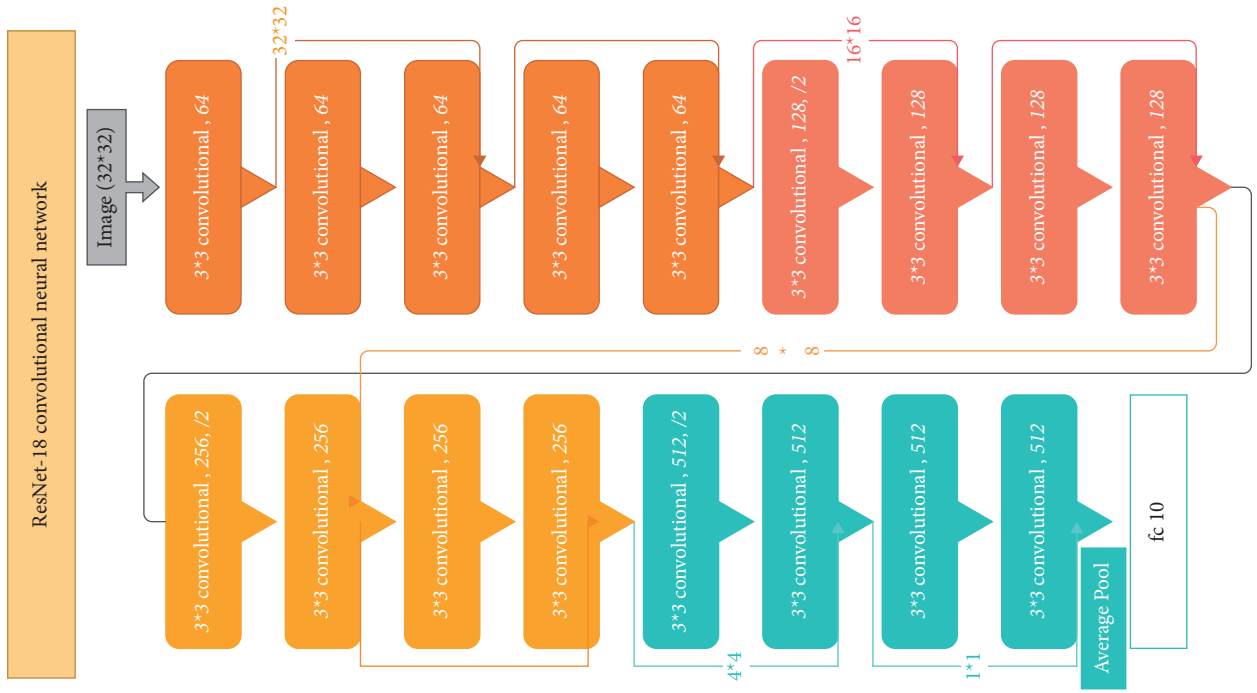


Figure 15: ResNet-18 architecture [27].

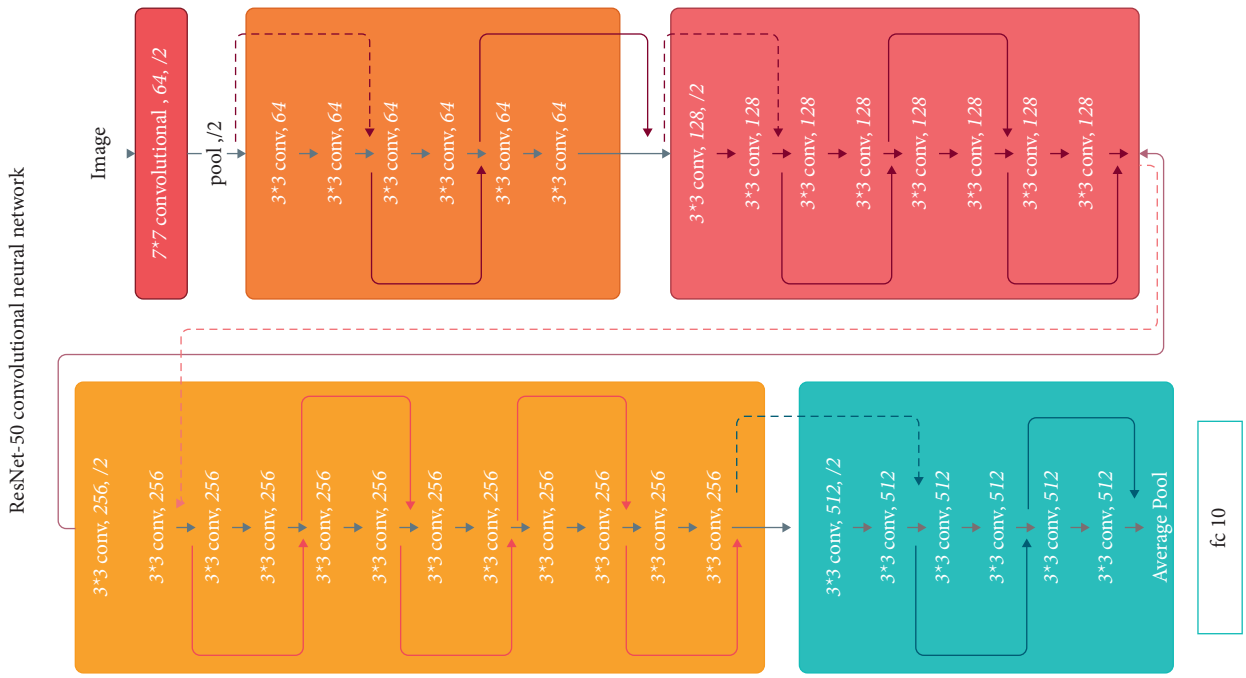


Figure 16: ResNet-50 architecture [28].

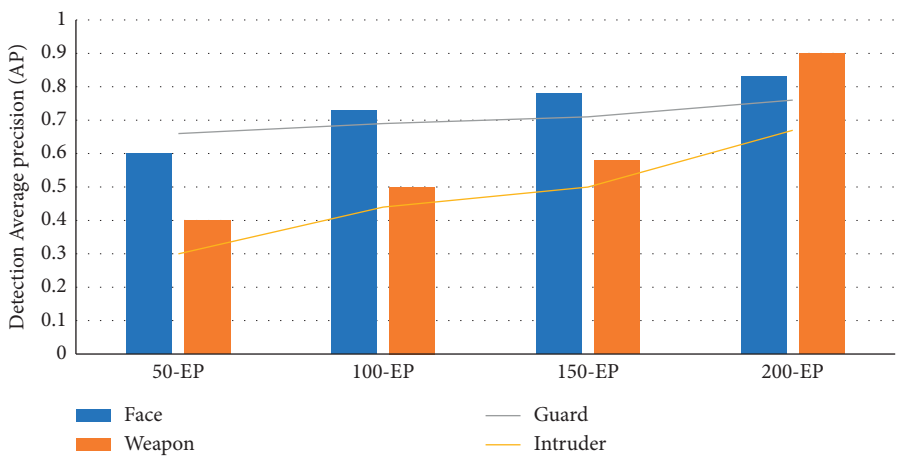


Figure 17: Detection average precision.

Table 2: Results comparison with existing work.

Method/ref no	Framerate (fps)	Error @ 50%	Memory (MB)
Ref [30]	20.6	0.442	8.5
Ref [29]	19.1	0.422	9.1
Ref [31]	16.2	0.341	10.5
Proposed	15.1	0.311	10.8

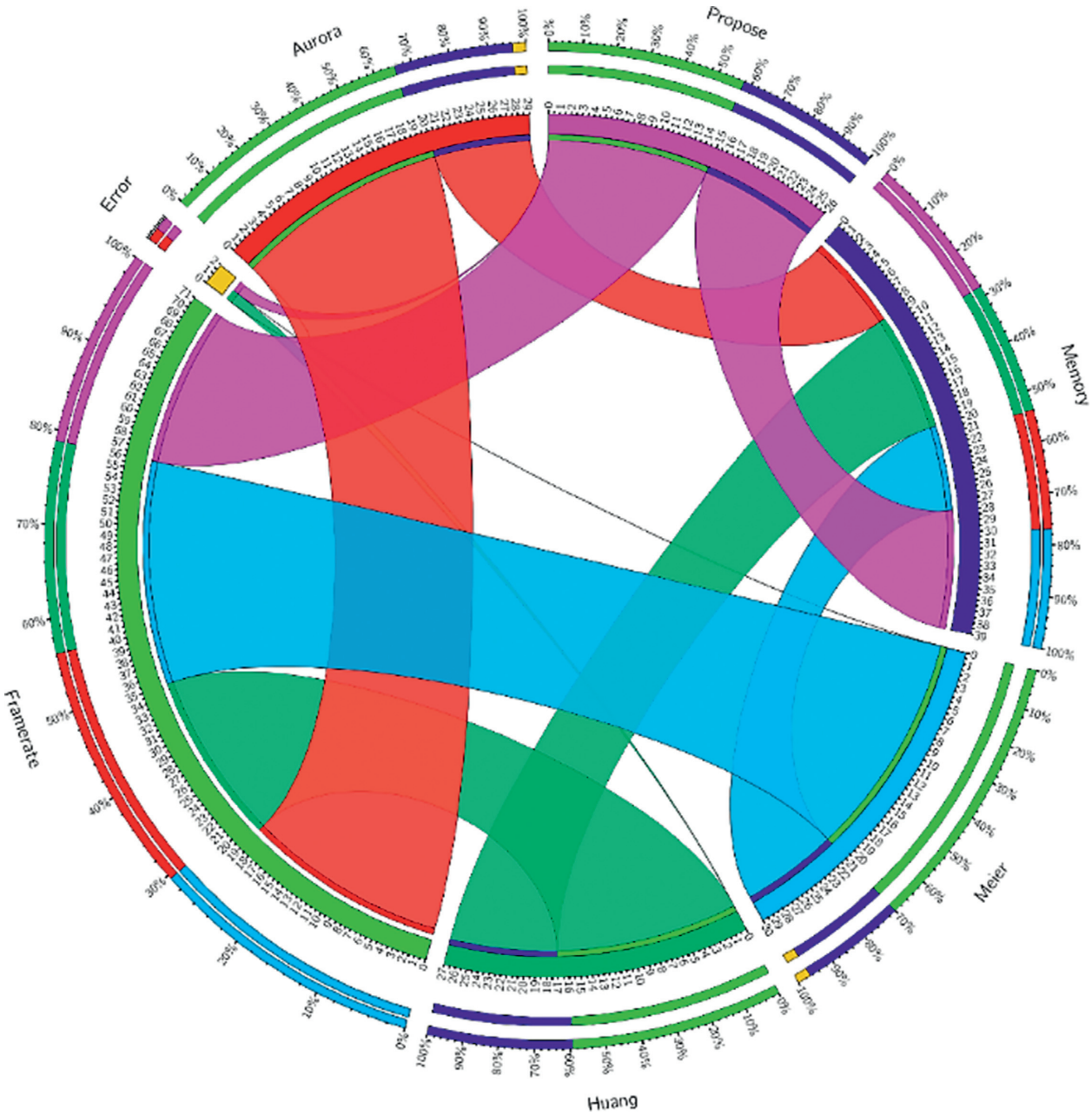


Figure 18: Circos graph results comparison.

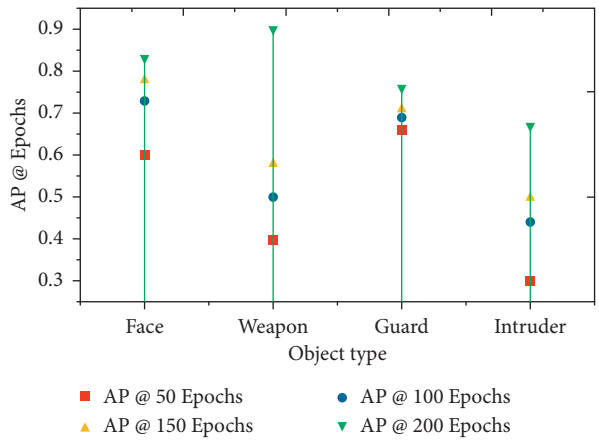


Figure 19: Average precision detection accuracy.

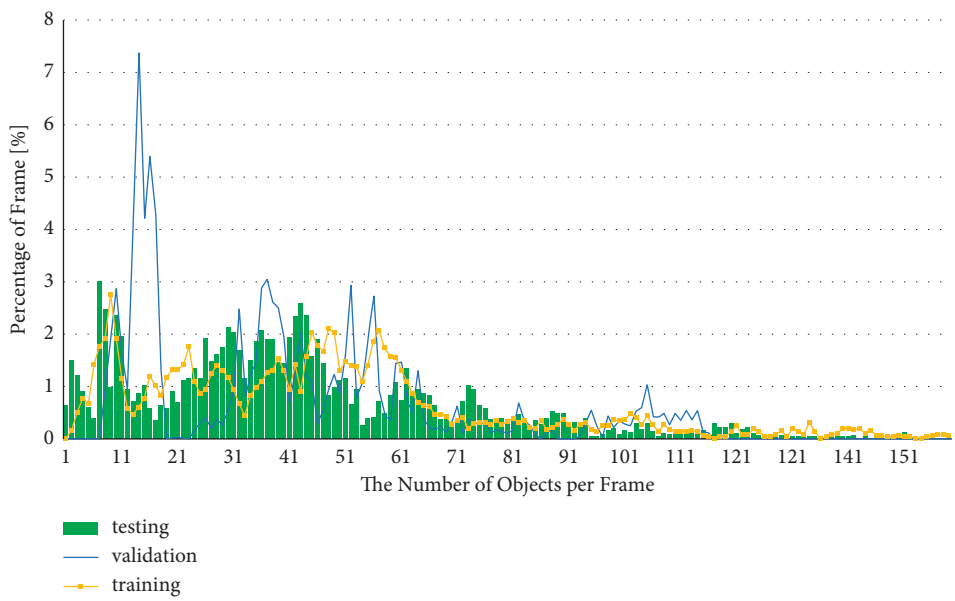


Figure 20: Number of objects per frame vs percentage of frames.

Table 3: Detailed results of training and testing for various object detectors.

Object detector	CNN feature extractors	No. of training images	Train-set AP	Time to train in hours	No. of testing images	Test-set AP
SqueezeNet-based FasterRCNN	Face	1805	0.22	13.86	695	0.23
	Weapon	1700	0.21	12.95	550	0.20
	Guard	1600	0.28	11.80	500	0.25
	Intruder	1550	0.20	11.56	500	0.18
GoogleNet-based FasterRCNN	Face	1805	0.96	25.08	695	0.97
	Weapon	1700	0.94	23.33	550	0.96
	Guard	1600	0.97	22.01	500	0.95
	Intruder	1550	0.80	21.64	500	0.79
ResNet-18-based FasterRCNN	Face	1805	0.97	17.01	695	0.97
	Weapon	1700	0.95	15.23	550	0.94
	Guard	1600	0.96	14.85	500	0.93
	Intruder	1550	0.79	14.66	500	0.83
ResNet-50-based FasterRCNN	Face	1805	0.97	47.01	695	0.98
	Weapon	1700	0.96	45.20	550	0.97
	Guard	1600	0.97	44.09	500	0.96
	Intruder	1550	0.81	42.77	500	0.85

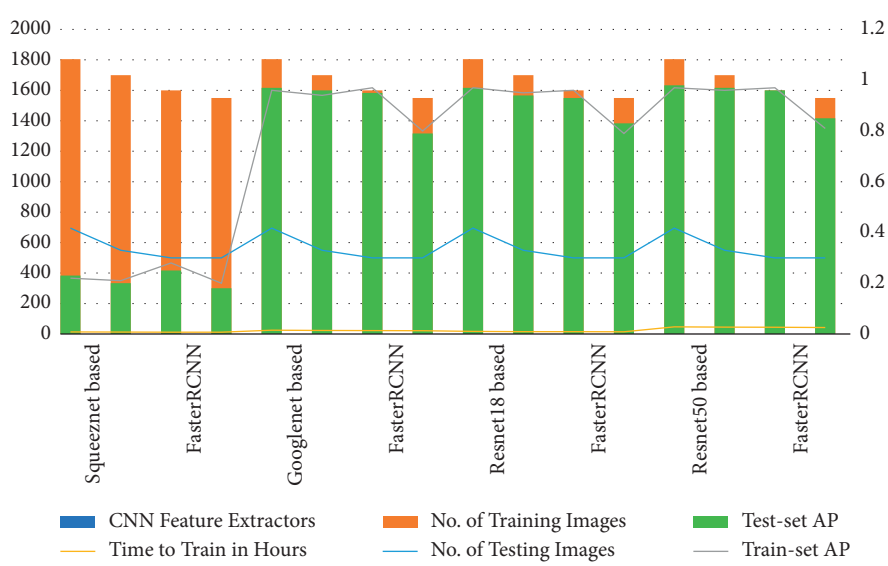


Figure 21: Train and test set average precisions (APs).

5. Conclusions

This research investigates a deep learning-based security and surveillance method for an organization, institution, or any other official procession. The surveillance system directly synchronizes the real-time data with the control room. The air surveillance system validates individuals and the community's protection by notifying the concerned departments about vulnerabilities to lay the foundation. Multiple experiments are carried out to validate the working system. Multiple experiments contained different kinds of CNNs feature to know their capabilities to detect various objects. Four well-known CNNs, namely, SqueezeNet, GoogleNet, ResNet-18, and ResNet-50, which are feature-based FasterRCNNs, were evaluated and reported. There are manifold advantages of an air surveillance system over the existing surveillance system used in our country. Firstly, the air surveillance system's pace will be optimum as they are aerial vehicles capable of roaming at any desired location. Secondly, air surveillance will be abandoned as these (UAV's) undergo surveillance irrespective of getting overtired or exhausted, contrary to those humans who need rest. Thirdly, this latest system will be systematic and least prone to errors, having not as many defaults as human calculations. Human beings are more liable to cause errors than machines. Finally, the air surveillance system is much cheaper than existing surveillance systems because the cost to deploy this system is comparatively less, and it does not involve assigning a task to human resources during surveillance because they are incorporated with sensors and are made intelligent. This system can also be utilized in monitoring the disaster-stricken area to assist while taking management initiatives afterwards. In the future, this work can be extended using the ensemble of other deep learning models, such as VGG19, InceptionV3, and Xception, along with the extension of the data set with fire and smoke categories.

Data Availability

The data used in the experiments will be available upon request.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors thank the University of Engineering and Technology Taxila and Higher Education Commission Pakistan for supporting this work under NRPU Project no. 6338.

References

[1] D. He, S. Chan, and M. Guizani, "Drone-assisted public safety networks: the security aspect," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 218–223, 2017.

[2] S. Berrahal, J. H. Kim, S. Rekhis, N. Boudriga, D. Wilkins, and J. Acevedo, "Border surveillance monitoring using quadcopter UAV-aided wireless sensor networks," *Journal of Communications Software and Systems*, vol. 12, no. 1, pp. 67–82, 2016.

[3] Z. Zaheer, A. Usmani, E. Khan, and M. A. Qadeer, "Aerial surveillance system using UAV," in *Proceedings of the 2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN)*, pp. 1–7.

[4] J. S. Gadda and R. D. Patil, "Quad copter (UAVs) for border security with GUI system," *International Journal of Engineering Research and Technology*, vol. 2, no. 12, pp. 620–624, 2013.

[5] J.-N. Lee and K.-C. Kwak, "A trends analysis of image processing in unmanned aerial vehicle," *World Academy of Science, Engineering and Technology*, vol. 8, no. 2, 2014.

[6] T. Krajník, M. Nitsche, S. Pedre, L. Přeučil, and M. E. Mejail, "A simple visual navigation system for an UAV," in *Proceedings of the International Multi-Conference on Systems, Signals & Devices*, pp, 1-6, Chemnitz, Germany..

[7] M. R. Haque, M. Muhammad, D. Swarnaker, and M. Arifuzzaman, "Autonomous quadcopter for product home delivery," in *Proceedings of the 2014 International Conference on Electrical Engineering and Information & Communication Technology*, pp. 1–5, 2014.

[8] G. Ding, Q. Wu, L. Zhang, Y. Lin, T. A. Tsiftsis, and Y.-D. Yao, "An amateur drone surveillance system based on the cognitive Internet of Things," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 29–35, 2018.

[9] G. Amato, P. Barsocchi, F. Falchi et al., "Towards multimodal surveillance for smart building security," *Proceedings*, vol. 2, no. 2, 2018.

[10] B. Alford, E. Curran, and S. Cole, "Determining the value of UAVs in Iraq," *The Journal of Conventional Weapons Destruction*, vol. 22, no. 1, 2018.

[11] P. S. Bangare, A. Pote, S. L. Bangare, P. Kurhekar, and D. Patil, "The online home security system: ways to protect home from intruders & thefts," *International Journal of Innovative Technology and Exploring Engineering*, vol. 2, no. 3, pp. 2278–3075, 2013.

[12] M. S. Munagekar, "Smart Surveillance system for theft detection using image processing," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 8, pp. 232–234, 2018.

[13] A. Alshammari and D. B. Rawat, "Intelligent multicamera video surveillance system for smart city applications," in *Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0317–0323, IEEE, uary, 2019.

[14] P. Bhagyalakshmi, P. Indhumathi, and L. R. Bhavadharini, "Real time video surveillance for automated weapon detection," *International Journal of Trend in Scientific Research and Development (IJTSRD)*, vol. 3, no. 3, 2019.

[15] A. Mahalanobis, J. L. Cannon, S. R. Stanfill, R. R. Muise, and M. A. Shah, "Network video image processing for security, surveillance, and situational awareness," in *Proceedings of the SPIE - The International Society for Optical Engineering*, pp. 1–8, FL, USA, August 2004.

[16] A. Reghunath, S. V. Nair, and J. Shah, "Deep learning based customized model for features extraction," in *Proceedings of the 2019 International Conference on Communication and Electronics Systems (ICCES)*, pp. 1406–1411, 2019.

[17] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37–52, 1987.

[18] S. C. Ng, "Principal component analysis to reduce dimension on digital image," *Procedia Computer Science*, vol. 111, pp. 113–119, 2017.

[19] W. Y. Lu and M. Yang, "Face detection based on viola-jones algorithm applying composite features," in *Proceedings of the 2019 International Conference on Robots & Intelligent System (ICRIS)*, pp. 82–85, Haikou, China, 2019.

[20] S. Ren, K. He, R. Girshick, J. Sun, and R.-C. N. N. Faster, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[21] T. Liu and T. Stathaki, "Faster R-CNN for robust pedestrian detection using semantic segmentation network," *Frontiers in Neurorobotics*, vol. 12, no. 64, 2018, in English.

[22] R. Ezhilarasi and P. Varalakshmi, "Tumor detection in the brain using faster R-CNN," in *Proceedings of the 2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 388–392, Palladam, India, ust, 2018.

[23] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[24] Matlab, "Getting started with R-CNN, fast R-CNN, and faster R-CNN," 2021, https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html.

[25] O. Russakovsky, J. Deng, H. Su et al., "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[26] C. Szegedy, Y. Jia, W. Liu et al., "Going deeper with convolutions," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

[27] A. S. Das, "CNN architectures: LeNet, VGG, GoogLeNet,, R. A. More," 2019, Accessed: Mar. 20, https://medium.com/sidereal/cnns-architectures-lenet-alexnet-vgggooglenet-resnet-and-more-666091488df5.

[28] O. A. Resnet50, https://www.mathworks.com/help/deeplearning/ref/resnet50.html.

[29] P. Angelov, P. Sadeghi-Tehran, and C. Clarke, "AURORA: autonomous real-time on-board video analytics," *Neural Computing & Applications*, vol. 28, no. 5, pp. 855–865, 2017.

[30] D. Meier, R. Brockers, L. Matthies, R. Siegwart, and S. Weiss, "Detection and characterization of moving objects with aerial vehicles using inertial-optical flow," in *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2473–2480, Hamburg, Germany, ember.

[31] C. Huang, P. Chen, X. Yang, and K. T. Cheng, "REDBEE: a visual-inertial drone system for real-time moving object detection," in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1725–1731, ember.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 9223965, 10 pages
https://doi.org/10.1155/2021/9223965

Research Article

Cell Traffic Prediction Based on Convolutional Neural Network for Software-Defined Ultra-Dense Visible Light Communication Networks

Shanjun Zhan,[1] **Lisu Yu**,[1] **Zhen Wang**,[1,2] **Yichen Du**,[1] **Yan Yu**,[3] **Qinghua Cao**,[1] **Shuping Dang**,[4] **and Zahid Khan**[5]

[1]*School of Information Engineering, Nanchang University, Nanchang 330031, China*
[2]*School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[3]*School of Information Engineering, Jingdezhen Ceramic University, Jingdezhen 333403, China*
[4]*Department of Electrical and Electronic Engineering, University of Bristol, Bristol BS8 1UB, UK*
[5]*College of Computer and Information Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia*

Correspondence should be addressed to Lisu Yu; lisuyu@ncu.edu.cn

Received 20 June 2021; Accepted 9 August 2021; Published 19 August 2021

Academic Editor: Farhan Ullah

With the explosive growth of ubiquitous mobile services and the advent of the 5G era, ultra-dense wireless network (UDN) architectures have entered daily production and life. However, the massive access capacity provided by 5G networks and the dense deployment of micro base stations also bring challenges such as high energy consumption, high maintenance costs, and inflexibility. Fiber-based visible light communication (FVLC) has the advantages of large bandwidth and high speed, which provides an efficient connection option for UDN. Thus, in order to make up for the poor flexibility of UDN, we propose a new FVLC-UDN architecture based on software-defined networks (SDNs). Specifically, SDN decouples the data plane and the control plane of the device and centralizes the control of the LED in the cell through a unified control plane, which can not only improve the resource allocation ability of the network but also transmit the data only as the data plane, reducing the manufacturing and implementation costs of the LED. In order to get a better resource allocation scheme, this paper proposes a model for predicting cell traffic based on convolutional neural networks. By predicting the traffic of each cell in the control domain, the traffic trend and cells' status in the future period of time in the control domain can be obtained, so that a much more efficient resource allocation scheme can be formulated proactively to reduce energy consumption and balance communication loads. The experimental results show that on the real cell traffic dataset, this method is better than the existing prediction methods when the size of training dataset is limited.

1. Introduction

With the rapid development of communication networks and the continuous expansion of network scales, great challenges are confronting wireless communication systems due to the need for a large number of concurrent services with high connection density and heterogeneous service demands. This is especially the case in dense urban areas, where there are large-scale distributed wireless communication systems [1]. As a result, mobile communication traffic has witnessed explosive growth. Cisco predicts that the IP traffic will triple from 2017 to 2022 [2]. In view of this challenge, a small cell-based ultra-dense wireless network (UDN) design is proposed for 5G and higher versions [3]. However, the traditional communication equipment cannot meet the high requirements of 5G or 6G communication systems, such as high capacity, high data rate, high spectral efficiency, high energy efficiency, and low delay. The latest research shows that the UDN based on fiber-based visible light communication (FVLC) is an effective solution [4]. Based on this, this paper focuses on the traffic prediction of FVLC networks in massive access scenarios.

However, the traditional UDN architecture lacks flexibility in adapting to the data traffic demand with rapid time,

space, and spectrum changes, and the redundant deployment of equipment will lead to high maintenance and energy consumption costs. To this end, this paper proposes a new network framework based on SDN, UDN, and FVLC, which is called the software-defined ultra-dense FVLC (SDUD-FVLC) network. Specifically, the architecture uses the idea of SDN to decouple the control plane and data plane of the device [5–7], making the network architecture more reasonable and orderly. It is applied to the FVLC network to optimize its structure in an ultra-dense network.

Many researchers have noticed that the control scheme proposed by the control plane is of paramount importance for resource allocation, and accordingly, they have proposed different solutions [8, 9]. However, these solutions are all planning for the state of the existing traffic and have a certain delay for dynamic traffic changes. Therefore, to deal with the problem of delay, we believe that it is necessary to proactively evaluate the state of traffic changes and formulate a control plane based on the evaluation results. To this end, we propose a traffic prediction model based on convolutional neural networks. The use of convolutional neural networks can well capture the characteristics of dynamic traffic and trends and predict the network traffic for a period of time in the future. On this basis, the control plane is predictable and thereby well managed. To achieve this goal, this paper focuses on building a better traffic prediction model and proposes a 2D convolutional neural network (2D-CNN) model.

Specifically, the traffic predict for the cell is a time series analysis problem, and the research on the time series problem has become extensive. For example, Chen et al. [10] investigated the temporal and spatial characteristics of cellular network traffic, in which the researchers proposed a model of traffic changes over time and space on weekly cellular network traffic data. There are also some other studies that make accurate predictions based on the nonstationarity and seasonality of cellular networks [11]. The prediction of cellular traffic in these studies can be regarded as a time series analysis problem whose performance depends on its linear statistical models, such as autoregressive integrated moving average (ARIMA), α stability [12], and entropy theory [13]. Artificial intelligence and machine learning technologies are also introduced to help solve traffic prediction problems, like linear regression (LR) and support vector regression (SVR) methods [14, 15].

However, with the development and popularity of 5G networks, the formation of ultra-dense network architecture, resulting in the mode of cellular network, is highly complex. It is clear that classical methods based on linear models and simple neural networks are unable to provide precise predicting results of network traffic in the 5G era [16]. In order to enable high-quality traffic prediction for a selected cell, some researchers have also used deep learning methods, like recursive neural computing [17], deep transfer learning [18], and deep meta-learning [19]. On the other hand, most of the aforementioned works can only handle grid-based traffic, which is of little help for cell traffic prediction of micro base stations in UDN.

Differently, the authors in [20] employ a convolutional neural network to predict cell traffic and use the traffic of a selected cell and 80 adjacent cells to form a 9×9 2D image. In this way, they can expand time to form a 3D dataset. At the same time, the authors divide the time correlation into two parts: daily cycle and hourly cycle, and then use a two-channel convolutional neural network for traffic prediction. Similarly, the authors in [21] leverage the same strategy to restructure the cell traffic data to be 3D data according to spatial and temporal characteristics. In order to get a better correlation of time series, data are sorted into three temporal components: hourly cycle, daily cycle, and weekly cycle. A three-channel convolutional neural network framework is the input to predict the cell traffic.

However, both of the above studies focus on the prediction of cellular traffic data of large base stations in the whole city. At the same time, the above two models have higher requirements for input data, requiring the selection of historical traffic data of 80 cells adjacent to the selected cell. The transmission and storage of historical data of 80 cells will undoubtedly render considerable consumption of network resources and memory for a micro base station. In UDN, the number of micro base stations is large, and their locations are densely distributed. Therefore, the above two models are obviously inappropriate for the prediction of cell traffic of densely distributed micro base stations in the UDN environment.

Considering the above constraints and challenges, our proposed 2D-CNN model in this paper is suitable for applying in SDUD-FVLC networks. First, it does not require too much historical data from the data plane to precisely predict cell traffic. Second, the control plane can formulate a reasonable control plane based on the predicted cell state and traffic trend to reduce energy consumption and balance communication load. To summarize, the major contributions of this paper are as follows:

(i) This paper proposes a new network framework to meet the requirements in the future networks.

(ii) Through traffic prediction, the future state of a cell can be known in advance, so that the control plane can be formulated proactively to avoid unnecessary delay.

(iii) Using CNN for traffic prediction can better capture the temporal characteristics of cell traffic.

(iv) Only a small amount of historical data from the cell is needed to precisely predict the traffic direction, which can save network resources and reduce the load on the control plane.

The specific content of the paper is as follows. Section 2 elaborates on the new network architecture of SDUD-FVLC. Section 3 discusses the source of the dataset used in the experiment and introduces 2D-CNN. In Section 4, the proposed 2D-CNN model is elaborated, and the related algorithm is given. Section 5 shows the comparison model and evaluation parameters. Furthermore, the experimental results of different models are compared and analysed. Finally, Section 6 concludes the paper.

2. Software-Defined Ultra-Dense Visible Light Communication Networks

In order to meet the challenges brought by UDN, a novel UDN architecture based on a software-defined network (SDN) is proposed in [22–24]. SDN is programmable, scalable, and flexible, which can effectively mitigate the problems of high energy consumption and large redundancy of UDN. Using the concept of SDN, the communication equipment is decoupled, the control plane and the data plane are separated, and the data plane is controlled through a unified control plane, thereby simplifying the management and control of the UDN network.

However, the arrival of 5G networks and the 6G era, which is not far away from us, have put forward higher requirements and challenges for the transmission speed of traffic and the quality of network services. In order to improve the quality of network service, the VLC network architecture based on SDN has been proposed [25, 26]. Nevertheless, most of the indoor VLC communication environment is not very complicated, SDN cannot play its maximum role and is limited by the communication range. Thus, the existing software-defined VLC schemes cannot efficiently solve the problems in the ultra-dense network environment. By combining optical fiber and VLC to form the structure of FVLC, the communication range of VLC can be expanded, so that it can be applied to large and ultra-dense areas, improve network service quality, and simplify network management and control.

Therefore, in this paper, we look forward to the future network architecture. A novel network architecture based on UDN, SDN, and FVLC is proposed. In this architecture, we divide the ultra-dense network area into many small areas. In these small areas, we can apply VLC communication to them and connect the devices in each small area through an optical fiber to form the FVLC structure in the ultra-dense network, which can improve the quality of service and transmission rate of the network. But this will certainly form a complex network environment, and it is difficult to control and deploy the ultra-dense network as a whole. So, we use SDN to decouple the equipment in each small area and control and allocate the data plane of the small area in the super-dense area through a single control plane. In this way, the LED device in FVLC only needs to be responsible for data transmission, which greatly simplifies the function of LED and greatly reduces its production and use costs. Finally, a new network architecture is formed as shown in Figure 1.

3. Data Processing and Observation Analysis

3.1. Wireless Large Traffic Dataset. The data used in this paper is obtained from the dataset provided by Telecom Italia [27], a large European telephone service provider, which mainly includes the communication records of telephone services, SMS services, and Internet activities (62 days, 500 million records) in Milan and Trentino. In the spatial dimension, the dataset divides the entire city into $H \times W$ grids, with each square in the grid being divided into a "unit"; each unit covers 0.05 square kilometers. The data used in this paper come from the dataset of Milan. In the dataset of Milan, $H = W = 100$ means that the city area is divided into 100×100 units. In the time dimension, the dataset recorded traffic from 00:00 on November 1, 2013, to 00:00 on January 1, 2014, at a 10-minute interval.

3.2. Temporal and Spatial Characteristics. In this paper, we focus on the time characteristics of cell traffic. As shown in Figure 2, we show the time distribution of five different services (phone call, phone out, SMS call, SMS out, and network traffic) in randomly selected units in Milan for a period of 336 hours over a two-week period. It can be seen that the time activities of the five services follow the characteristics of daily and weekly cycles. Obvious features like a significant reduction in data during the weekend can be observed, suggesting that there is almost no population in the area on weekends, e.g., factories, business districts, and campus areas. In addition, the number of activities for the five services is slightly different. The amount of Internet services is always greater than that of the other four services. The traffic dialed in by phone is almost the same as that dialed out by phone, while the number of dialed in by SMS is obviously excessive. Traffic prediction depends on these important temporal characteristics of mobile traffic, and convolutional neural networks can well extract these temporal characteristics.

In addition, the prediction of the traffic of the cell in the FVLC-UDN environment is closely related to the time series. In other words, the characteristics of the time series play a decisive role in the prediction of the traffic of the cell. In the experimental process, by comparing the prediction results of traffic prediction based on temporal and spatial characteristics and the prediction results of traffic prediction based only on time series features [20, 21], it can be found that the temporal correlation is more important than spatial dimension correlation. The essence of spatial characteristics is the similarity of time series between adjacent cells. It is thereby not very helpful for traffic prediction but requires a large amount of external data from nonselected base stations. In terms of resource utilization, adding spatial features to traffic prediction indicates a large consumption of network resources. Taking [20] as an example, for the traffic prediction of a selected cell, the model requires several days of historical data from 80 cells adjacent to the target cell. The sorting and transmission of the input data consume tremendous resources.

In the era of 5G, UDN consuming a huge amount of network resources emerges, and therefore, more attention should be paid to the management and reasonable use of network resources. Consequently, in the model proposed in this paper, we discard the spatial characteristics of cell traffic and focus on its temporal characteristics only, which can also achieve excellent performance. In this way, the model in this paper requires less data and can reduce resource consumption for data transmission and processing.

Figure 1: A new network framework based on SDN, UDN, and FVLC.

3.3. 2D Convolutional Neural Network. In machine learning, the convolutional neural network (CNN) architecture has been perfected and successfully used in a variety of tasks, such as face recognition, scene marking, image classification, and so on [28]. With the development and maturity of the convolutional neural network framework, the data it can process have expanded from 1D data to 2D data and from 2D data to 3D data. The emergence of 3D-CNN architecture makes it easier for us to manipulate volumetric data such as video and extract the spatial and temporal characteristics of 3D data [29].

In the model proposed in this paper, we use the 2D-CNN architecture to use the time characteristics of the cellular network to predict the traffic of the cell in FVLC-UDNs. The structure of 2D-CNN is shown in Figure 3. In 2D-CNN, the convolution kernel slides in 2D. In Figure 3, f_h and f_w are the height and width of the convolution core, and i and j are the height and width of the 2D image. Through the sliding of the convolution kernel with a stride of 1, the entire time series data can be effectively covered. At the same time, when the convolution kernel slides to the edge of the data, zero padding is used to avoid ignoring the feature extraction of the edge value of the 2D image. The 2D-CNN model described above can extract the data characteristics of 2D images well, thus making the traffic prediction for cellular networks more accurate.

4. 2D Convolutional Neural Network Model

4.1. Time Series Modeling. In this paper, to fully utilize the influence of the historical traffic data of the base station on the prediction accuracy, the daily period and the weekly period are taken as two features of the 2D image, and the traffic data in this specific period can be regarded as another feature. On this basis, the three features are expanded in an hourly cycle to form the data image as shown in Figure 4. When using a convolutional neural network for training, the model can extract the characteristics of the hourly cycle, daily cycle, and weekly cycle of traffic well, thus improving the accuracy of prediction.

As shown in Figure 4, the three features are continuously extended to form a 2D data image of $L \times K$. In this paper, we select 12 hours of continuous time, using the daily cycle, weekly cycle, and traffic data as three characteristics, so $L = 12$ and $K = 3$, and we build a 2D data image except 12×3. We construct the original dataset into a continuous two-dimensional image in the above way, that is, we construct the original dataset into $[X_{t-p}, X_{t-(p-1)}, \ldots, X_{t-1}]$ continuous dataset. The value in the upper left corner of the 2D image is the continuous time in hours; that is, the 2D image is arranged in continuous time. At this point, we have finished modeling the input data (training set and test set). The original dataset has a period of 62 days. We select the last 7 days of data as the

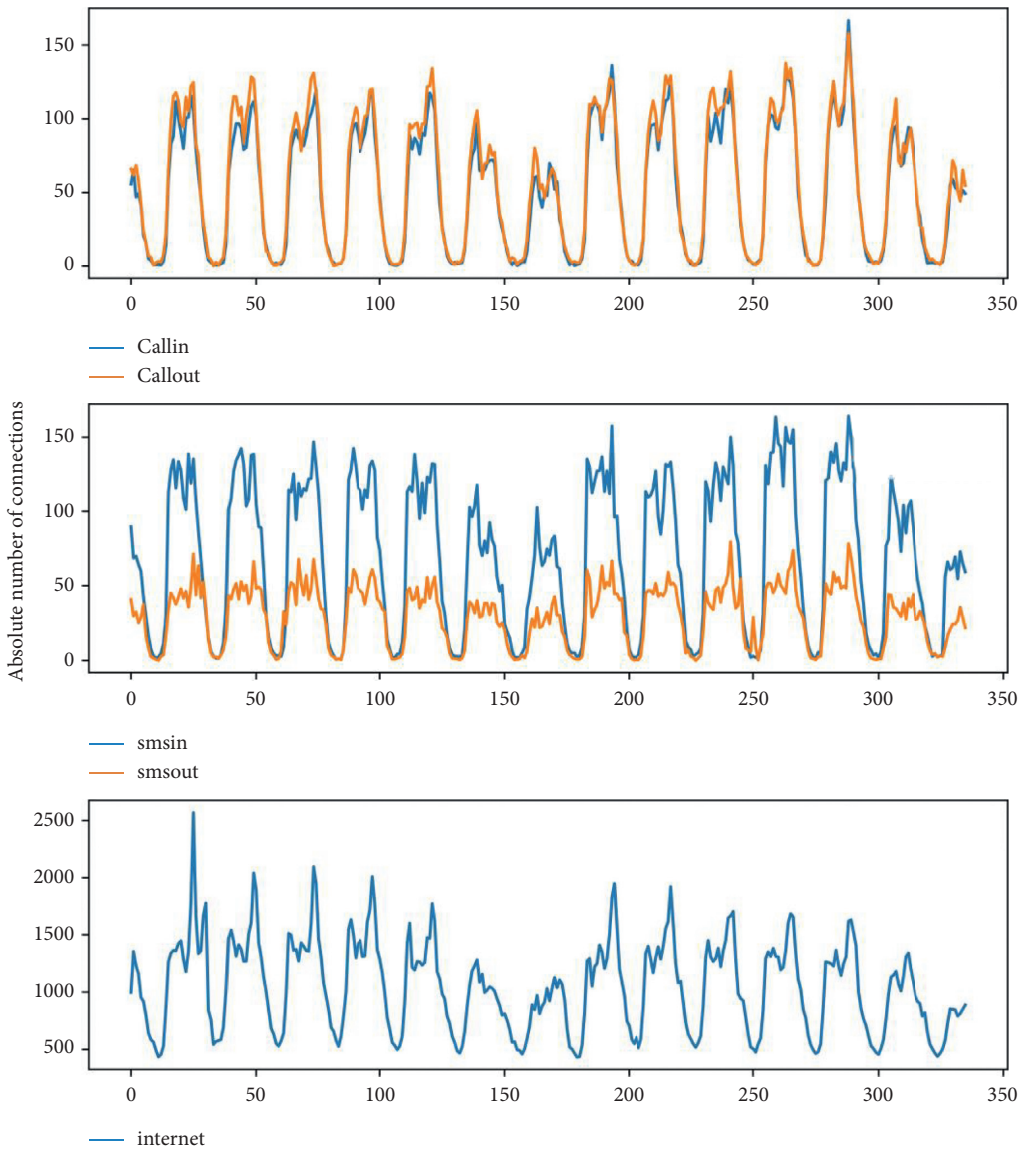


Figure 2: Traffic characteristics.

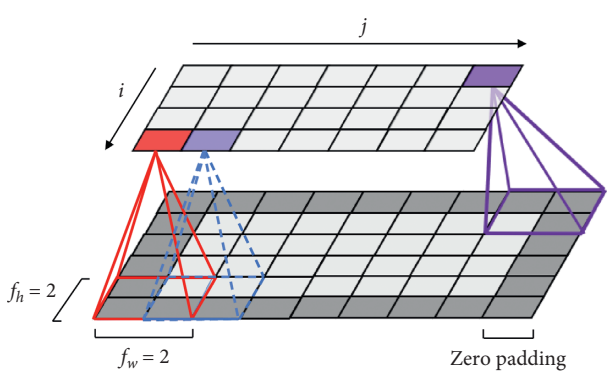


Figure 3: : 2D-CNN architecture.

test set, and all previous traffic is set to the training set and the validation set (allocated at a ratio of 7 : 2), thus completing all processing of the data.

4.2. Convolutional Neural Network Model. Figure 5 shows the overall framework of the 2D-CNN model proposed in this paper. It consists of several 2D convolution layers, max-

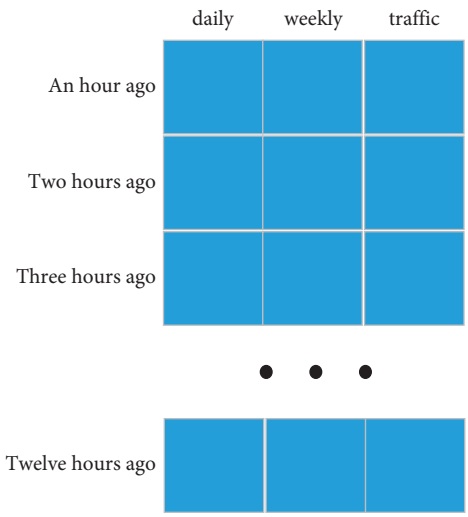


Figure 4: 2D data image.

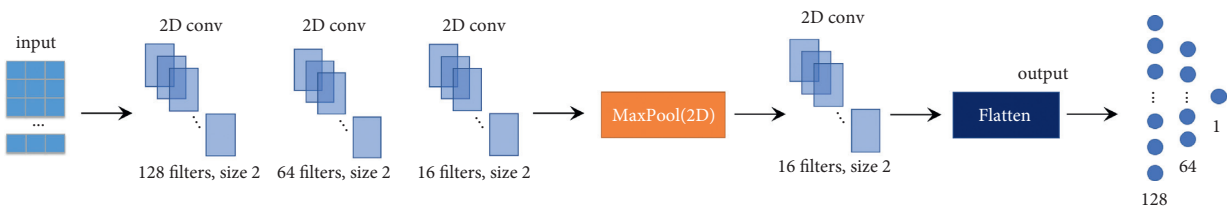


Figure 5: : 2D-CNN model.

pooling layers, Flatten layers, and several fully connected layers. The 2D convolutional layer is directly connected to the 2D image built in Section 3.1. Through a series of connected layers, the final output of the prediction results is obtained. By integrating the historical data of the selected cell into the 2D-CNN model, the next hour's cell traffic is predicted accurately, so that the control plane can provide an efficient control scheme based on the cell status in the next hour, which can help attain the goal of energy saving, load balancing, and delay reducing.

4.2.1. Convolution Layer. Using a 2D convolution layer, the 2D image dataset modeled in Section 3.1 can be extracted well. After the size of the convolution kernel is selected, the convolution kernel is slid on the surface of the 2D image, and when the edge of the image is encountered, the method of zero padding is used to avoid ignoring the data characteristics of the image edge. After the data framed by the convolution kernel are subjected to a convolution operation, they are connected to the next network layer as the data of the next network layer for use.

4.2.2. Max-Pooling Layer. Due to a large amount of trained 2D image data, the max-pooling layer can effectively reduce the amount of calculation, memory usage, and the number of parameters (so as to reduce the risk of overfitting). After determining the size and stride of the receiving field in the pooling layer, only the maximum input value of each receiving field can enter the next layer network, while other inputs are discarded. Besides reducing the amount of computation, memory usage, and the number of parameters, the max-pooling layer also introduces a certain degree of invariance for small changes. At the same time, the max-pooling layer has some disadvantages. First of all, it is obviously destructive; then, in some applications, invariance is not desirable, and the pooling layer cannot be used for these applications. However, in this model, the application of the maximum pooling layer can well reduce the amount of calculation, memory usage, and the number of parameters, so that the model runs faster and can get the prediction of the selected base station more quickly and timely.

4.2.3. Flatten Layer. The output data of the max-pooling layer are 2D data, while the input data of the fully connected layer are 1D data. Through Flatten layer, the input of the max-pooling layer can be "flattened"; that is, multidimensional input can be 1D to meet the input requirements of the fully connected layer.

4.2.4. Fully Connected Layer. After the 2D input of the max-pooling layer is converted into 1D data through the Flatten layer, it is input to the dense layer composed of three fully connected layers. The input and hidden layer neurons use ReLU as the activation function, and it (ReLU) also acts as an activation function for the output neurons.

We can train our 2D-CNN model to predict traffic by minimizing the mean absolute error (MAE) between the estimated value $\hat{y}_n$ and the actual value y_n. The loss function can be defined as

$$\mathrm{MAE} = \frac{1}{N}\sum_{n=1}^{N}\left|y_n - \hat{y}_n\right|, \tag{1}$$

where y_n is the actual value, $\hat{y}_n$ is the predicted value, and N is the size of the dataset.

The complete model is shown in Figure 5. It shows the connection mode of each network layer and the trend of fitting data in the model.

4.3. Model Algorithm. We trained the 2D-CNN model through the process of Algorithm 1, which consists of two parts: training instance construction and model training. The input of the 2D image is obtained from the integration of the original traffic dataset and then input into the model for training. After constructing the data and initializing the model as shown in Figure 5, a small number of training instances are randomly selected. In this model, we use the widely used optimization technology to train and randomly select a small batch to train. Afterwards, the model can be trained by the loss function defined in Section 4.2.4. In the process of training, in order to prevent overfitting or loss function wandering around the minimum, we use an early stopping method to stop training in time, so as to optimize the performance of the model.

5. Prediction Results and Performance Analysis

5.1. Evaluation Index. In this section, we will describe the quantitative criteria of evaluation indicators. For this task, we refer to the application and case of time series prediction. For this model, we will use root mean square error (RMSE) as the evaluation index. RMSE is defined as

$$\mathrm{RMSE} = \left(\frac{1}{N}\sum_{n=1}^{N}\left(y_n - \hat{y}_n\right)^2\right)^{(1/2)}, \tag{2}$$

where y_n is the real data of the datasets, $\hat{y}_n$ is the predicted value of the model, and N is the number of datasets to be tested. The performance of the smaller RMSE model is better.

Mean absolute error (MAE) is the average value of the difference between two time series. The definition of MAE has been introduced in Section 4.2.4. We also use the loss value of the trained model as an index to evaluate the performance of these models. A smaller MAE means that the model has higher performance.

We also use the mean absolute percentage error (MAPE), which is defined as

$$\mathrm{MAPE} = \frac{1}{N}\sum_{n=1}^{N}\left|\frac{y_n - \hat{y}_n}{y_n}\right|. \tag{3}$$

In this index, y_n, $\hat{y}_n$, and N are the same as above, and smaller MAPE means higher performance.

At the same time, we use the direction accuracy (DirAcc) to compare the difference between the predicted direction and the actual number sequence direction, which is defined as follows:

$$\mathrm{DirAcc} = \sum_{n=1}^{N} d_n, \tag{4}$$

and

$$d_n = \begin{cases} 1, & \left(y_n - y_{n-1}\right)\left(\hat{y}_n - \hat{y}_{n-1}\right) \geq 0, \\ 0, & \text{others.} \end{cases} \tag{5}$$

For this index, y_n and y_{n-1} represent the actual data of T and T–1 time, respectively. Similarly, we can get $\hat{y}_n$ and $\hat{y}_{n-1}$.The higher the value of the indicator, the better the prediction. From the definition of the indicator, we can see that DirAcc $\leq M$ (M is the size of the test data).

RMSE, MAE, MAPE, and DirAcc will be used to evaluate the performance of these models. In the next section, we will show some common and widely used time series prediction models. By comparing the evaluation indexes with the model proposed in this paper, we can verify the superior performance of the model in cellular network traffic prediction.

5.2. Comparison Model. In order to get a more intuitive evaluation effect and better understand the performance of the model proposed in this paper, three models which have achieved good results in time series prediction and which have been applied to production and life are selected, which are HA, ARIMA, and SVG. The evaluation indexes of the above three models and the 2D convolutional neural network model proposed in this paper are calculated, respectively, namely, the four evaluation indexes shown in Section 5.1. Through the intuitive comparison of data, we can clearly see the excellent performance of the 2D model and the accuracy of prediction results.

HA model adopts the strategy of the historical average, which predicts the value of the next week by getting the average value of the last week. This is a relatively simple probability strategy to predict cellular network traffic through the historical average. It is suitable for the historical data comparison rule and unified sequence. For the cellular network traffic which is irregular and interfered by the external environment, HA model performs poorly, as can be seen from Table 1.

By combining the autoregressive (AR) model, moving average (MA) model, and differential method, we can get the ARIMA model, which is usually used for time series predication. The AR model extracts the relationship between the current value and the history and uses the historical data of the variable itself to predict. The MA model focuses on the accumulation of error terms in the autoregressive model and then integrates AR and MA into one model by the difference method. ARIMA is widely used in time series prediction, which is a mature prediction model.

In order to obtain a better prediction effect in the field of regression, we use the idea of support vector machine (SVM)

Input: training set historical data: $[X_{t-p}, X_{t-(p-1)}, \ldots, X_{t-1}]$
Output: trained 2D convolutional neural network model
//Construct training examples
$D \leftarrow \varphi$
While all available time interval T $(1 \leq T \leq N)$
$Y_t = [X_{t-p}, X_{t-(p-1)}, \ldots, X_{t-1}]$
Put the training instance Y_t into D
//Y_t is the actual value at time t
end
//Training model
Initialization of all trainable parameters θ
Repeat
Randomly select a batch of instances D_b from D
Use D_b and Adam optimization to find the best θ (the loss value defined in Section 4.2.4))
until meet the stop condition (early stopping or completed the training batch)

Algorithm 1: 2D convolutional neural network model training process.

Table 1: Results of forecasting performance

Model		2D model	HA	ARIMA	SVG
RMSE	Call in	**9.72**	44.00	12.92	20.57
	Call out	**10.95**	45.43	13.67	19.37
	SMS in	**17.58**	51.56	21.77	26.35
	SMS out	**8.40**	19.93	10.39	11.08
MAE	Call in	**6.68**	38.86	9.10	13.65
	Call out	**7.68**	40.55	9.76	13.57
	SMS in	**12.33**	45.18	14.87	16.50
	SMS out	**6.14**	16.87	7.41	7.59
MAPE	Call in	**43.06**	649.54	116.43	59.72
	Call out	**47.24**	713.40	83.04	57.52
	SMS in	**64.02**	455.01	71.13	83.31
	SMS out	**68.32**	421.39	86.37	86.42
DirAcc	Call in	**0.83**	0.62	0.71	0.67
	Call out	**0.78**	0.67	0.69	0.71
	SMS in	**0.64**	0.57	0.63	0.62
	SMS out	**0.60**	0.57	0.62	0.56

to construct a model besides support vector regression (SVR). According to the distribution of training set, SVR divides different types of samples in the specified interval region and does not calculate the loss for the samples falling in the interval region but calculates the loss for the samples falling outside the interval region, so as to improve the prediction accuracy of the model.

5.3. Performance Analysis of 2D Convolutional Neural Network Model. Table 1 shows the evaluation index values of the 2D convolution model and three comparison models (HA, ARIMA, and SVR). It can be seen intuitively that the evaluation index of the 2D convolutional neural network model proposed in this paper is better than that of other methods. Compared with the other three models, evaluation metrics (RMSE, MAE, MAPE, and DirAcc) for 2D-CNN have better results.

During the experiment, we randomly select a certain area in Milan, and the traffic prediction results of four different services (phone in, phone out, SMS in, and SMS out) are shown in Figure 6. We can observe that the prediction results can accurately capture the dynamic trend of real traffic in 2D images. The traditional time series has a general performance in dealing with complex and nonstationary traffic data and has a higher RMSE and lower accuracy in all four services.

In addition, from the prediction results in Figure 6, the 2D model can also effectively predict sudden changes of traffic trend. By taking the daily period, weekly period, and traffic as the features of 2D data image and then extending the three features on the hourly period, the 2D convolutional neural network model proposed in this paper achieves better performance on all evaluation indexes of four kinds of services.

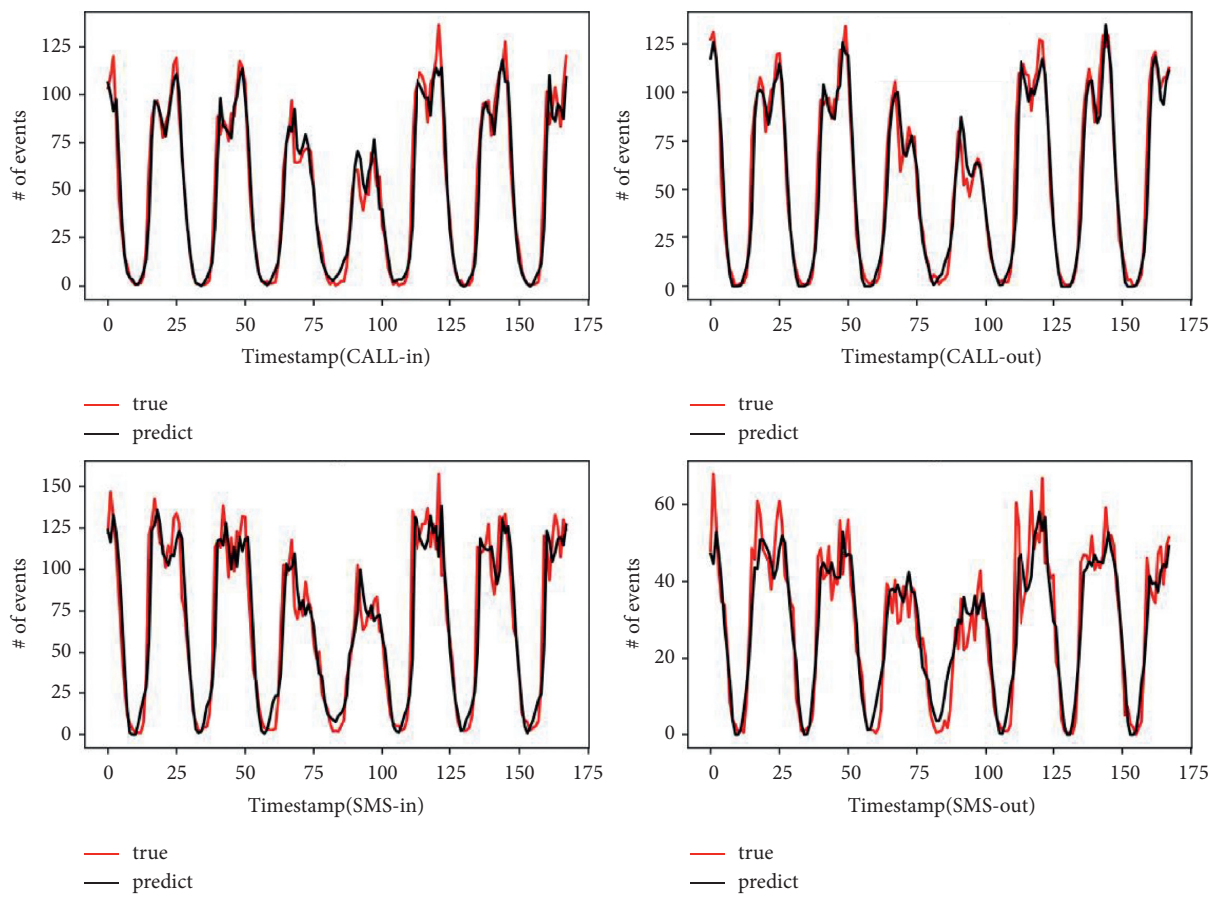


Figure 6: Prediction results of randomly selected units.

6. Conclusions

Compared with traditional UDN, the newly proposed UDN architecture based on SDN and FVLC can better adapt to the high requirements of the future networks. In this paper, a 2D-CNN deep learning model is proposed to predict the traffic of cells in SDUD-FVLC architecture, so that the network state can be perceived from a global perspective and used to solve the delay of the current control scheme. Compared with the traditional time series prediction model, 2D-CNN can capture the dynamic performance of traffic well and requires less data than 3D models, which reduces the load of the control plane. Also, the model presents a better prediction effect and higher performance, which provides a feasible solution to solve the delay problem of the SDUD-FVLC network control plane.

Data Availability

The data used to support the findings of this study are included within the article. The data presented in this study are also available at https://doi.org/10.1038/sdata.2015.55.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

SZ was responsible for investigation and methodology. ZW and SD contributed to formal analysis. QC and YD were responsible for software. SZ, QC, and LY validated the study. SZ and YD were responsible for resources. SZ and LY conceptualized the study and wrote the original draft. SZ and YY were responsible for visualization and data curation. SD and ZK reviewed and edited the manuscript. LY supervised the study and acquired funding. ZW and LY were responsible for project administration. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

This research was supported in part by the State Key Laboratory of Computer Architecture (ICT, CAS) Open Project under grant no. CARCHB202019, China Postdoctoral Science Foundation under grant no. 2021TQ0136, Training Program of Innovation and Entrepreneurship for Undergraduates in Nanchang University under grant nos. 2020CX234 and 2020CX236, and Student Research Training Program (SRTP) in Nanchang University under grant nos. 5258 and 5259. ZK was supported in part by Prince Sultan University, Saudi Arabia.

References

[1] L. Yu, J. Wu, A. Zhou, E. G. Larsson, and P. Fan, "Massively distributed antenna systems with nonideal optical fiber fronthauls: a promising technology for 6G wireless communication systems," *IEEE Vehicular Technology Magazine*, vol. 15, no. 4, pp. 43–51, 2020.

[2] V. Cisco, "Cisco visual networking index: forecast and methodology (2017–2022)," *White paper*, vol. 1, 2018.

[3] S. Feng, R. Zhang, W. Xu, and L. Hanzo, "Multiple access design for ultra-dense VLC networks: orthogonal vs non-orthogonal," *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 2218–2232, 2019.

[4] L. Yu, Z. Liu, M. Wen et al., "Sparse code multiple access for 6G wireless communication networks: recent advances and future directions," *IEEE Communications Standards Magazine*, vol. 5, no. 2, pp. 92–99, 2021.

[5] A. Filali, Z. Mlika, S. Cherkaoui, and A. Kobbane, "Preemptive SDN load balancing with machine learning for delay sensitive applications," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15947–15963, 2020.

[6] K. Phemius, M. Bouet, and J. Leguay, "DISCO: distributed multi-domain SDN controllers," in *Proceedings of 2014 IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–4, Krakow, Poland, May 2014.

[7] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN networks: a survey of existing approaches," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3259–3306, 2018.

[8] Y. Zhao, Y. Chen, R. Jian, and L. Yang, "A resource allocation scheme for SDN-based 5G ultra-dense heterogeneous networks," in *Proceedings of 2017 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, Singapore, December 2017.

[9] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[10] X. Chen, Y. Jin, S. Qiang, W. Hu, and K. Jiang, "Analyzing and modeling spatio-temporal dependence of cellular traffic at city scale," in *Proceedings of IEEE International Conference on Communications (ICC)*, pp. 3585–3591, London, UK, June 2015.

[11] A. D'Alconzo, A. Coluccia, F. Ricciato, and P. Romirer-Maierhofer, "A distribution-based approach to anomaly detection and application to 3G mobile traffic," in *Proceedings of GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*, pp. 1–8, Honolulu, HI, USA, December 2009.

[12] R. Li, Z. Zhao, J. Zheng, C. Mei, Y. Cai, and H. Zhang, "The learning and prediction of application-level traffic data in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3899–3912, 2017.

[13] R. Li, Z. Zhao, X. Zhou, J. Palicot, and H. Zhang, "The prediction analysis of cellular radio access network traffic: from entropy theory to networking practice," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 234–240, 2014.

[14] H. Sun, H. X. Liu, H. Xiao, R. R. He, and B. Ran, "Use of local linear regression model for short-term traffic forecasting," *Transportation Research Record: Journal of the Transportation Research Board* in *Proceedings of 82nd Annual Meeting of the Transportation-Research-Board*, vol. 1836, no. 1, pp. 143–150, Washington, DC, USA, January 2003.

[15] N. Sapankevych and R. Sankar, "Time series prediction using support vector machines: a survey," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 24–38, 2009.

[16] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International Journal of Forecasting*, vol. 22, no. 3, pp. 443–473, 2006.

[17] Z. Qin, F. Cao, Y. Yang et al., "CellPred," in *Proceedings of the ACM on Interactive Mobile Wearable and Ubiquitous Technologies*, vol. 4, no. 1, pp. 1–24, New York, NY, USA, May 2020.

[18] C. Zhang, H. Zhang, J. Qiao, D. Yuan, and M. Zhang, "Deep transfer learning for intelligent cellular traffic prediction based on cross-domain big data," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1389–1401, 2019.

[19] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng et al., "Urban traffic prediction from spatio-temporal data using deep meta learning," in *Proceedings of 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pp. 1720–1730, Anchorage, AK, USA, August 2019.

[20] C. Zhang, H. Zhang, D. Yuan, and M. Zhang, "Citywide cellular traffic prediction based on densely connected convolutional neural networks," *IEEE Communications Letters*, vol. 22, no. 8, pp. 1656–1659, 2018.

[21] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Spatial-temporal attention-convolution network for citywide cellular traffic prediction," *IEEE Communications Letters*, vol. 24, no. 11, pp. 2532–2536, 2020.

[22] Z. Zhou, J. Feng, C. Zhang, Z. Chang, Y. Zhang, and K. M. S. Huq, "SAGECELL: software-defined space-air-ground integrated moving cells," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 92–99, 2018.

[23] D. Wu, J. Yan, H. Wang, and R. Wang, "User-centric edge sharing mechanism in software-defined ultra-dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1531–1541, 2020.

[24] Q. Liu, G. Chuai, J. Wang, J. Pan, W. Gao, and X. Liu, "Proactive mobility management based on virtual cells in SDN-enabled ultra-dense networks," in *Proceedings of 2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, Shanghai, China, May 2019.

[25] H. Koumaras, D. Makris, A. Foteas, G. Xilouris, K. Ma et al., "A SDN-based WiFi-VLC coupled system for optimised service provision in 5G networks,," in *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pp. 14–17, IEEE, Chania, Greece, June 2018.

[26] J. Cosmas, B. Meunier, K. Ali, N. Jawad, M. Salih et al., "A 5G radio-light SDN architecture for wireless and mobile network access in buildings," in *Proceedings of 2018 IEEE 5G World Forum (5GWF)*, pp. 135–140, Silicon Valley, CA, USA, July 2018.

[27] G. Barlacchi, M. De Nadai, R. Larcher et al., "A multi-source dataset of urban life in the city of Milan and the Province of Trentino," *Scientific Data*, vol. 2, no. 1, Article ID 150055, 2015.

[28] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, no. 12, pp. 1273–1278, 2019.

[29] H. Yang, C. Yuan, B. Li et al., "Asymmetric 3D convolutional neural networks for action recognition," *Pattern Recognition*, vol. 85, pp. 1–12, 2019.