Hindawi
Security and Communication Networks
Volume 2022, Article ID 4375232, 16 pages
https://doi.org/10.1155/2022/4375232

Research Article

Pattern Mathematical Model for Fingerprint Security Using Bifurcation Minutiae Extraction and Neural Network Feature Selection

Nesreen Alsharman,[1] **Adeeb Saaidah**,[2] **Omar Almomani**,[2] **Ibrahim Jawarneh**,[3] **and Laila Al-Qaisi**[2]

[1]*Computer Science Department, The World Islamic Sciences Education University, Amman, Jordan*
[2]*Computer Network and Information Systems Department, The World Islamic Sciences Education University, Amman, Jordan*
[3]*Mathematics Department, Al-Hussein Bin Talal University, Ma'an, Jordan*

Correspondence should be addressed to Nesreen Alsharman; nesreen.alsharman@wise.edu.jo

Received 11 November 2021; Revised 3 February 2022; Accepted 25 February 2022; Published 16 April 2022

Academic Editor: Luigi Catuogno

Biometric based access control is becoming increasingly popular in the current era because of its simplicity and user-friendliness. This eliminates identity recognition manual work and enables automated processing. The fingerprint is one of the most important biometrics that can be easily captured in an uncontrolled environment without human cooperation. It is important to reduce the time consumption during the comparison process in automated fingerprint identification systems when dealing with a large database. Fingerprint classification enables this objective to be accomplished by splitting fingerprints into several categories, but it still poses some difficulties because of the wide intraclass variations and the limited interclass variations since most fingerprint datasets are not categories. In this paper, we propose a classification and matching fingerprint model, and the classification classifies fingerprints into three main categories (arch, loop, and whorl) based on a pattern mathematical model using GoogleNet, AlexNet, and ResNet Convolutional Neural Network (CNN) architecture and matching techniques based on bifurcation minutiae extraction. The proposed model was implemented and tested using MATLAB based on the FVC2004 dataset. The obtained result shows that the accuracy for classification is 100%, 75%, and 43.75% for GoogleNet, ResNet, and AlexNet, respectively. The time required to build a model is 262, 55, and 28 seconds for GoogleNet, ResNet, and AlexNet, respectively.

1. Introduction

Biometrics science is used to identify people using their physical characteristics. These characteristics are fingerprint, iris, palm, face, DNA, and voice [1]. Among these characteristics, the fingerprint is one the most accurate and reliable for identifying a person [2] since fingerprints are the unique biometric characteristics of any person; therefore, it is used in forensic divisions worldwide for criminal investigations where even the twins have nonidentical fingerprints.

Therefore, fingerprints have been confirmed to be good and secure biometrics. The process of fingerprint identification is to confirm or refuse if a scanned fingerprint belongs to a specific person or not. The increasing commercial applications and number of civilians that depend on fingerprint-based identification lead to a huge fingerprint database. Matching specific fingerprints stored in the database is computationally time-consuming. The subject of automatic fingerprint identification has received intensive attention among researchers.

To solve automatic fingerprint identification, fingerprints can be stored in databases based on the characteristics of their ridge and furrow patterns. In general, fingerprints can be divided into three major classes known as whorl (W), loop (L), and arch (A) according to Galton [3]. The Galton classification scheme is shown in Figure 1. The database of fingerprints can now be indexed based on one of these three classes [4].

Determining the classes that fingerprint belongs to allows fingerprint matching on the portion or index of the database corresponding to that particular class. By doing this, the time required for fingerprint identification is reduced. Such an indexing mechanism for fingerprint forms is the basis of fingerprint classification.

Edward Henry improved the classification of Galton by increasing the number of classes to five [5]. The Edward Henry classification scheme's five classes are arches, tented arches, left loop, right loop, and whorl, which are shown in Figure 2. Several approaches have been proposed for automatic fingerprint classification. These approaches are categorized based on rule-based, structural-based, frequency-based, and syntactic [6]. Other approaches are statistical-based, neural network-based, and multiclassifier-based [7]. After fingerprint classification is completed, fingerprint matching is required. Figure 3 shows the general fingerprint matching processes.

Preprocessing stage aims to improve and enhance the quality of the image. The preprocessing stage has two functions: ridge enhancement restoration and segmentation of fingerprint images. In the classification stage, the input image is commonly classified into three or five main classes, as shown in Figures 1 and 2. In the stage of feature extraction, the block of the relevant information is extracted that will be applied for identification with the template fingerprint. In the verification stage, the decision is determined based on the percentages or matching results of similarity. Several techniques for fingerprint matching have been proposed. These techniques are minutiae-based [8], correlation-based [9], and pattern-based [10].

The rest of the paper is organized as follows: Section 2 explains the related work. Section 3 shows the proposed method. Section 4 elaborates the used dataset and the mathematical model used to classify the dataset. Section 5 presents results and discussion. Finally, Section 6 concludes the paper.

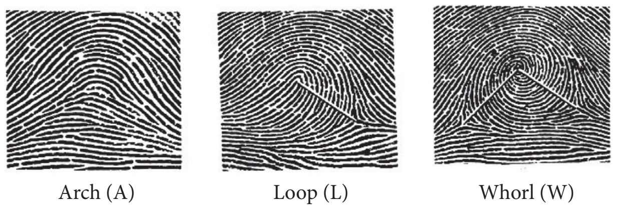


Figure 1: Galton–Henry classification of fingerprints classes.

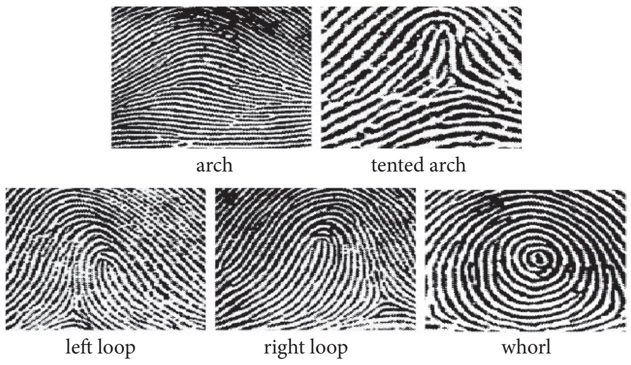


Figure 2: Edward Henry classification of fingerprints classes.

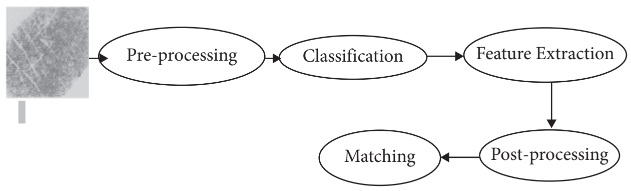


Figure 3: Matching process.

2. Related Work

The classifications of fingerprints shall be made on the basis of following characteristics: ridges, ends, bifurcations, delta, and cores. Fingerprints are classified into whorl, right loop, double loop, left loop, and arch classes based on these characteristics. Most of the fingerprints datasets have a large size. To find a match-out fingerprint of such a large dataset, a correct classification of the images is required. Machine learning (ML) is one of the applications that is attracting the growth of categorization of fingerprints in impractical application domains. As a result, various research studies employing machine learning to classify fingerprints have been done. Therefore, several studies have been conducted for fingerprint classification using ML.

The study by [11] implemented a fingerprint classification system using a fuzzy neural network classifier and its output in the recognition method. The classification scheme is based on the extraction of the fingerprint feature, which involves encoding the singular points along with their relative positions and directions from a fingerprint image of the binaries. Analysis of images is carried out in four steps, namely, segmentation, estimation of directional image, extraction of singular points, and encoding of features. This involves the encoding of the singular points (core and delta) with direction and location.

Zhang and Yan [12] presented ridge tracing analysis and curves features for fingerprint classification. SVM is an ML algorithm that adopts a robust approach to fingerprint classification. The presented approach provided a system of classification that was highly accurate. The algorithm's benefit is seen when classifying fingerprints into different classes.

Moreover, Hong and Wang et al. [13] proposed a combination of the SVM and the naive Bayes to classify the fingerprints based on the number of fingerprint core and delta points.

While Wang et al. [14] proposed a fingerprint classification algorithm based on a depth neural network to improve classification accuracy, they adopted the softmax regression for fuzzy classification. They gave the "suspicious" fingerprints a secondary class.

Furthermore, Kouamo and Tangha [15] proposed a fingerprint authentication model using a neural network with a

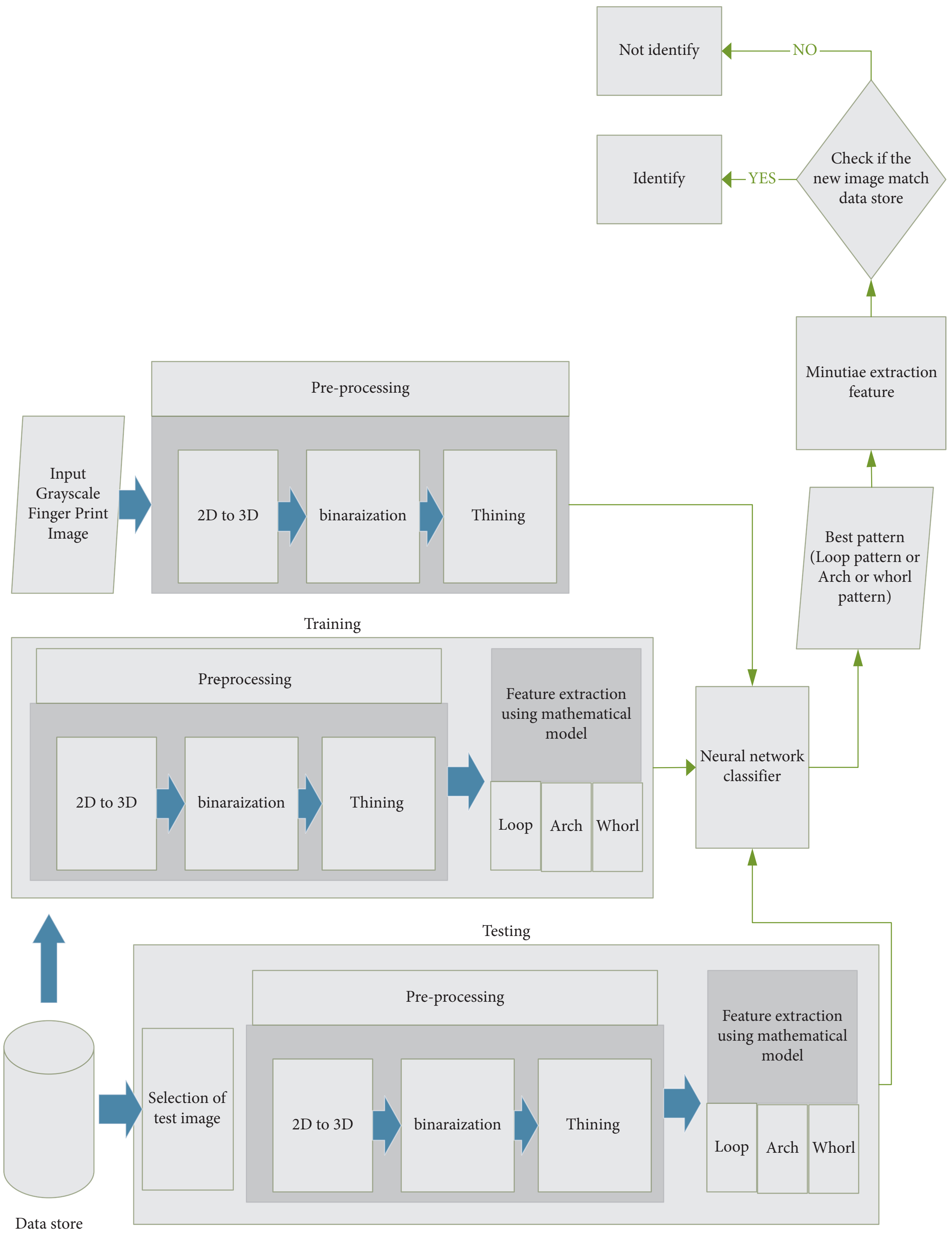

Not identify
NO
Identify
YES
Check if the new image match data store
Minutiae extraction feature
Best pattern (Loop pattern or Arch or whorl pattern)
Pre-processing
Input Grayscale Finger Print Image
2D to 3D
binaraization
Thining
Training
Preprocessing
2D to 3D
binaraization
Thining
Feature extraction using mathematical model
Loop
Arch
Whorl
Neural network classifier
Testing
Pre-processing
Selection of test image
2D to 3D
binaraization
Thining
Feature extraction using mathematical model
Loop
Arch
Whorl
Data store

Figure 4: Architecture of proposed model.

multilayer perceptron structure and extraction algorithm. They used probability calculations to identify the subblocks of the input image. More recently neural network fingerprint classification method is [16] where the proposed method is retrained over AlexNet, GoogleNet, and ResNet with an average precision of 95.55%, 92.51, and 94, 88 respectively.

Matching fingerprints is the mechanism by which the similarity scores between the two fingerprints match. Due to its intraclass correlation diversities from the fingerprint images of the same finger and its correlation similarities from the fingerprint images of different fingers, fingerprint matching is a challenging pattern-recognition problem. This form of diversity occurs particularly due to the pressure of the finger, the placement-rotation of the finger, the dryness of the skin and finger cuts, and so on. For the form of similarity similarities, this happens primarily when the method only describes the print for three types of fingerprint patterns (arch, loop, and whorl). Several studies have been proposed for fingerprint matching; here is a review of some of them.

Peralta et al. [17] proposed a general technique of decomposition for the matching algorithm based on minutiae. It breaks up the matching scores into very comprehensive processes. Any minute algorithm can be adapted to frameworks like MapReduce or Apache Spark by decomposition.

In another study conducted by Lee et al. [18], they proposed a new partial fingerprint matching for all sensors in mobile devices using minutiae and ridge-form features (RSFs). RSFs are the small ridge segments that observe unique edge shapes. There have been numerous algorithms of fingerprint classification developed [12, 19–26]. Among them, the generally used features are orientation and singularities image information. Combining these characteristics is a common occurrence. Unfortunately, singularity points are not always present in a fingerprint image: either the acquisition process was not perfect, resulting in a fractured fingerprint, or the fingerprint belongs to the arch class. Pseudosingularity points will be discovered and extracted in the circumstances mentioned above, allowing for fingerprint categorization and matching [27].

The geometric properties of major ridge curves in a fingerprint image called orientation field flow curves were used to achieve a manual fingerprint categorization (OFFCs) [28]. Recently, a method for detecting a fingerprint's reference point was proposed, which involved examining the curvatures of the fingerprint ridges, more information [29], which had a total execution time of 143 milliseconds for the most important stages. Distinctive Ridge Point (DRP), a recent fingerprint feature, has been developed [30], along with an enhancement triangle-based representation that includes minutiae. In this method, to achieve better outcomes, this strategy must reduce the dependence of ridge points on minutiae. Recently, a receiver operating characteristic (ROC) [31] curve model was suggested that used a weighted empirical approach to account for both the order constraint and the within-cluster correlation structure. As a result, the additional time complexity is required for statistical assessment of performance fingerprint matching data. Most recently, a novel technique [32] has been offered for fingerprint reconstruction that takes into account orientation field direction and minutiae density, although the suggested method for reconstruction of orientation field simply takes into account the local orientation pattern.

In the next section, we present the proposed model for fingerprint classification based on neural networks and matching based on bifurcation minutiae extraction.

3. Proposed Model

Identification of fingerprints is the oldest forensic science known to humans. Over time, fingerprints have proved to be the fastest, most accurate, and most cost-effective means of identifying unknown deceased persons, especially in a mass disaster setting. In addition, the fingerprint is one of the most accurate and discriminating biometrics that has been investigated and used to identify human beings for hundreds of years [33]. In order to address the limitations of existing contact-based fingerprint identification systems, improve recognition accuracy, and reduce time analysis function, a fingerprint database with categories and prepossessing using neural network classification technique has attracted growing attention in order to improve accuracy and reduce the time for fingerprint classification and matching. Figure 4 shows the architecture of the proposed model.

Algorithm 1 for the proposed model is given in the following box.

System 1 shows the mathematical model for the concentric whorl pattern, which is illustrated in Figure 5; see [4].

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= -x\left(x^2-1\right)^2. \end{aligned} \tag{1}$$

System 2 represents the mathematical model for the upper right-lower left (UR-LL) spiral whorl pattern, which is shown in Figure 6; see [4].

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= -x\left(x^2-1\right)^2 + 0.2y\left(x^2-1\right)^2. \end{aligned} \tag{2}$$

System 3 describes the mathematical model for the lower right-upper left (LR-UL) spiral whorl pattern, which is explained in Figure 7; see [4].

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= -x\left(x^2-1\right)^2 - 0.2y\left(x^2-1\right)^2. \end{aligned} \tag{3}$$

System 4 clarifies the mathematical model for the composite whorl with the S core pattern, which is shown in Figure 8; see [4].

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= -x\left(x^2-1\right)^2 + 0.9y\left(x^2-1\right)^2. \end{aligned} \tag{4}$$

System 5 represents the mathematical model for the plain arch pattern, which is appeared in Figure 9; see [35].

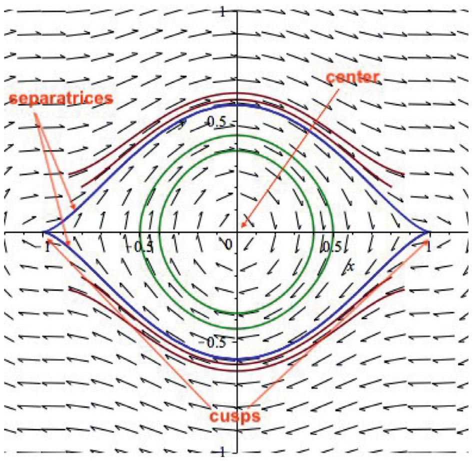
$$\begin{aligned} \dot{x} &= y^2, \\ \dot{y} &= -0.001x. \end{aligned} \tag{5}$$

System 6 shows the mathematical model for the tented arch pattern, which is shown in Figure 10; see [35].

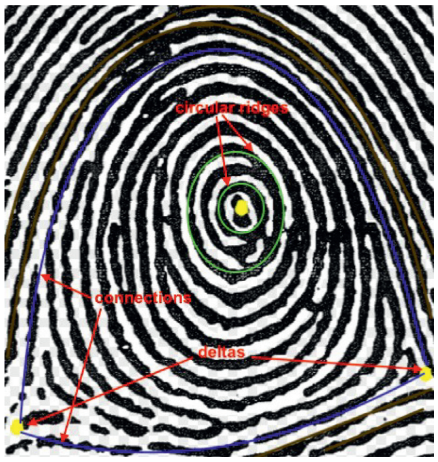
$$\begin{aligned} \dot{x} &= y^2, \\ \dot{y} &= -0.5x. \end{aligned} \tag{6}$$

```
Step 1: Preprocessing the dataset (image categorization using categorization method, 2D grayscale to 3D color, binarization, and
        thinning) and a test set of fingerprint images.
Step 2: Training the dataset using CNNs (GoogleNet, AlexNet, and ResNet).
        Where 70% of data are for training and 30 for testing.
Step 3: Input the image and preprocess the image (2D grayscale to 3D color, binarization, and thinning).
Step 4: Classify the input image using neural network classifiers.
Step 5: After the input fingerprint is classified into a specific pattern, matching using minutiae extraction will be done.
        Categorization method:
 (1) For ($i = 1$, $i =$ length of data set, $i$++)
 (2) {
 (3) X = read image
 (4) if $X$ satisfies $\{\dot{x} = y, \dot{y} = -x(x^2 - 1)^2\}$
 (5) OR $\{\dot{x} = y, \dot{y} = -x(x^2 - 1)^2 + 0.2y(x^2 - 1)^2\}$ OR $\{\dot{x} = y, \dot{y} = -x(x^2 - 1)^2 - 0.2y(x^2 - 1)^2\}$ OR $\{\dot{x} = y, \dot{y} = -x(x^2 - 1)^2 + 0.9y$
     $(x^2 - 1)^2\}$
 (6) THEN
 (7) $X$ belong to Whorl patterns
 (8) else if $X$ satisfies $\dot{x} = y^2$, $\dot{y} = 0.001x$ OR $\dot{x} = y^2$, $\dot{y} = 0.5x$
 (9) { – }
(10) { – }
(11) OR x&dot; = $y^2$, y&dot; = $5x$ THEN
(12) { – }
(13) $X$ belong to Arch patterns; else
(14) $X$ belong to loop patterns
(15) }
```

Algorithm 1: The algorithm for the proposed model.



(a)



(b)

Figure 5: Simulation of the concentric whorl fingerprint: (a) phase portrait of the model and (b) image of the concentric whorl fingerprint.

System 7 represents the mathematical model for the strong arch pattern, which is explained in Figure 11; see [35].

$$\begin{aligned}\dot{x} &= y^2,\\ \dot{y} &= -5x.\end{aligned} \tag{7}$$

3.1. Preprocessing. A fingerprint image has a lot of redundant information when it is captured. To get an acceptable and accurate image, the problems such as images with scars, too dry or too moist fingers, or incorrect pressure must be overcome. Hence, the input fingerprint images need to be preprocessed. The preprocess for fingerprint images can be done using some processes such as image enhancement, normalization, filtering, noise reduction, binarization, and thinning [36]. In this research, binarization and thinning are applied since they are widely used before fingerprint classification and matching. In the proposed architecture, preprocessing consists of three phases: 2D grayscale to 3D color, binarization, and thinning.

(i) 2D grayscale to 3D color: the main idea to convert from 2D grayscale to 3D color is that the CNNs just accept the image with 3D color so that the following MATLAB function is used for converting:

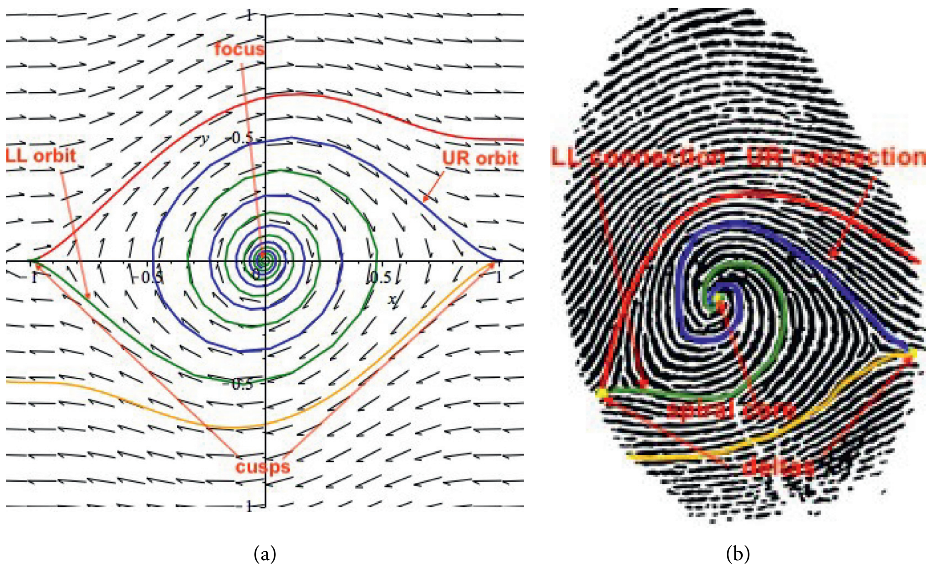


Figure 6: Simulation of the upper right-lower left (UR-LL) spiral whorl fingerprint: (a) phase portrait of the model and (b) image of the upper right-lower left (UR-LL) spiral whorl fingerprint.

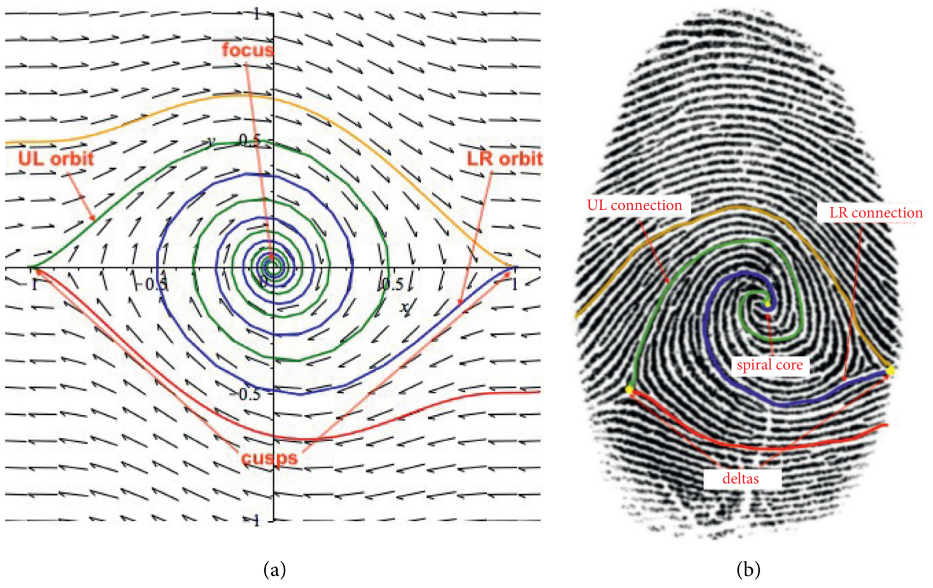
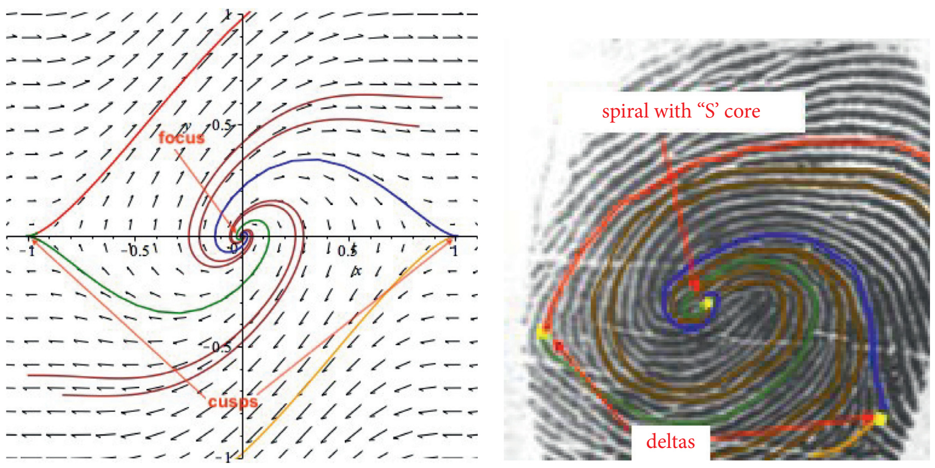


Figure 7: Simulation of the lower right-upper left (LR-UL) spiral whorl fingerprint: (a) phase portrait of the model and (b) image of the lower right-upper left (LR-UL) spiral whorl fingerprint.

AugmentedTrainingSet = augmentedImageDatastore (imageSize, trainingSet, ‘ColorPreprocessing’, ‘gray2rgb’); augmentedTestSet = augmentedImage-Datastore (imageSize, testSet, ‘ColorPreprocessing’, ‘gray2rgb’);

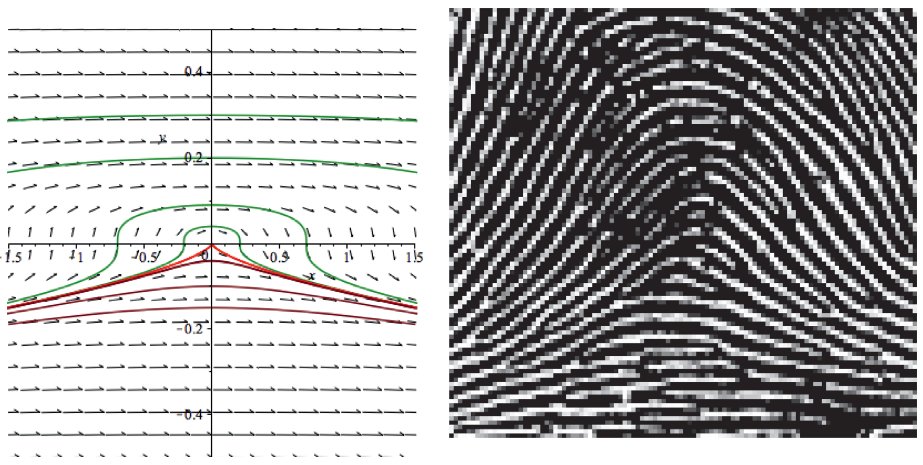
(ii) The separation of the object and background is known as binarization. The applied imbinarize function (I) using MATLAB generates a binary image from 2D grayscale or 3D color image by replacing all values above a globally determined threshold with 1s and setting all other values to 0s. By default, imbinarize uses Otsu’s method, which chooses the threshold value to minimize the intraclass variance of the threshold black and white pixel [37]. Imbinarize uses a 256-bin image histogram to compute Otsu’s threshold.

(iii) One way to make a skeleton is through thinning algorithms. The technique takes a binary image of a fingerprint and makes the ridges that appear in print just one pixel wide without changing the overall pattern and leaving gaps in the ridges creating a sort of “skeleton” of the image. Thinning makes it easier to find minutiae and removes a lot of redundant data that would have resulted in longer process time and sometimes different results [38].



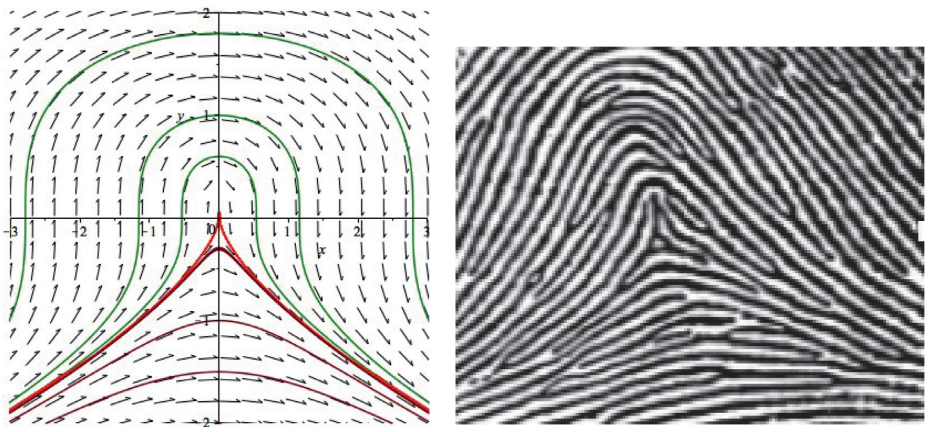
(a) (b)

FIGURE 8: Simulation of the composite whorl with S core fingerprint: (a) phase portrait of the model and (b) image of the composite whorl with "S" core.



(a) (b)

FIGURE 9: Simulation of the plain fingerprint: (a) phase portrait of the model and (b) image of the plain fingerprint [34].



(a) (b)

FIGURE 10: Simulation of the tented arch fingerprint: (a) phase portrait of the model and (b) image of the tented fingerprint.

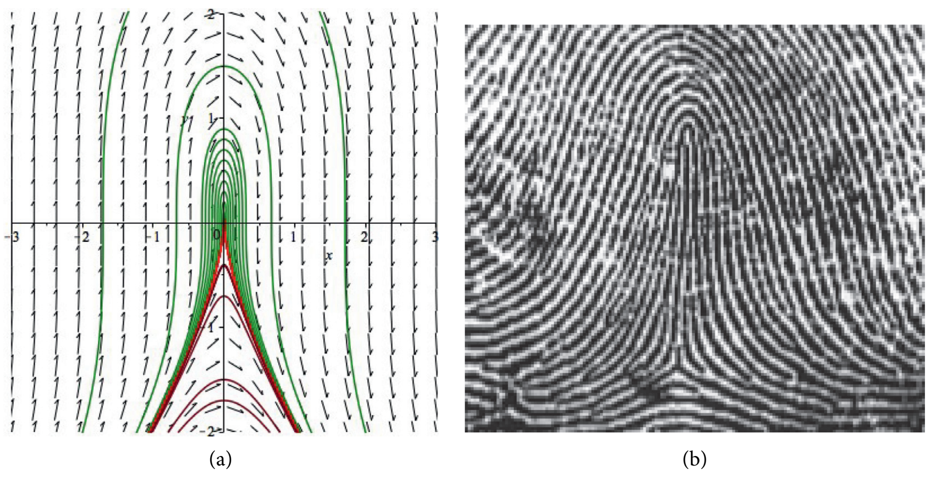


Figure 11: Simulation of the strong arch fingerprint: (a) phase portrait of the model and (b) image of the strong arch fingerprint.

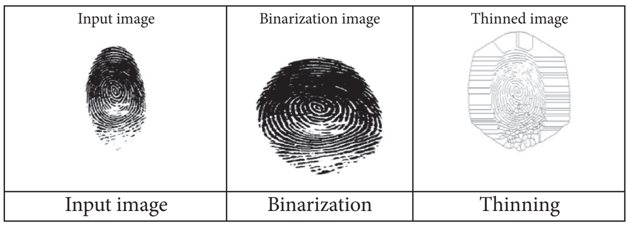


Figure 12: Preprocessing steps.

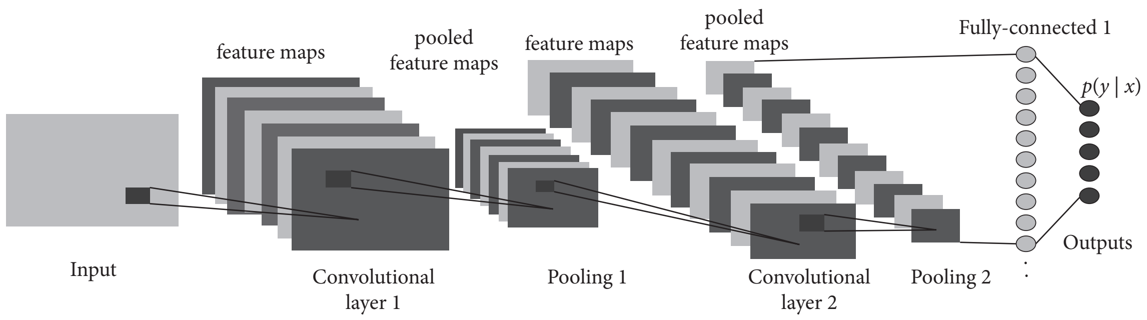


Figure 13: The structure of a CNN [51].

Figure 12 shows the preprocessing image after applying binarization and thinning.

3.2. Neural Network. ML algorithms are a field of Artificial Intelligence (AI) that provides computers with intelligence by studying the underlying relationships between the data and making decisions without explicit programming. Since the late 1990s, several and numerous ML algorithms have been implemented to mimic sensory human responses such as speech and vision but have generally failed to achieve satisfaction at the human level [39, 40]. The challenging nature of Machine Vision (MV) tasks produces a specific class of neural networks called CNN [41].

CNN is considered one of the best strategies for learning image content and shows state-of-the-art results related to image recognition, segmentation, detection, and retrieval-related tasks [42–44]. CNN's success has attracted attention outside academia, industry, and companies including Google, Microsoft, AT&T, NEC, and Facebook that have formed active study groups to explore CNN's new architectures [45]. At present, deep CNN-based models are employed by most of the frontrunners in image processing competitions. CNN is also a special one-size-fits-all multilayer neural network developed to recognize visual patterns directly from pixel images with minimal preprocessing [46]. Figure 13 shows the structure of a CNN. This research uses the AlexNet, GoogleNet, and ResNet CNN architecture classifier [47] for training fingerprint datasets.

3.2.1. AlexNet. The usage of AlexNet [47, 48] since the beginning of deep CNNs was limited to hand digit recognition tasks and did not scale well to all classes of images. AlexNet [49] is considered as the first deep CNN architecture, which showed groundbreaking results for image classification and recognition tasks. AlexNet was proposed by Lee et al. [18]. They improved the learning capacity of the CNN by doing it deeper and by applying several parameter optimizations strategies [49]. Technology constraints in the early 2000s curtailed the learning ability of deep CNN architecture by limiting it to limited dimensions. To gain from CNN's representational ability, AlexNet was simultaneously trained on two NVIDIA GTX 580 GPUs to address hardware deficiencies. With AlexNet, feature extraction stages have been expanded from 5 (LeNet) to 8 to render CNN accessible for different image categories [47].

3.2.2. GoogleNet. GoogleNet is called Inception-V1. The key goal of GoogleNet's architecture was to achieve high precision and reduce computational costs [50]. It presented the latest definition of the initiation block in CNN, where it integrates multiscale convolutionary transformations through the notion of separating, transforming, and merging. This block encapsulates filters of different sizes (1×1, 3×3, and 5×5) to capture various scales of spatial information (at both fine and coarse grain levels). In GoogleNet, convolutionary layers are substituted in small blocks, as suggested in the Network in Network (NIN) architecture, such as replacing each layer with micro-NN [51]. GoogleNet's exploitation of the concept of splitting, transforming, and merging has helped resolve a problem related to understanding various types of variations found in the same category of various pictures. GoogleNet's emphasis was on making the CNN parameter effective in addition to increasing learning efficiency.

3.2.3. ResNet (2015). Residual Neural Network (ResNet) was introduced by He et al. [52] as a novel architecture featuring "skip connections" and fast batch normalization. These skip connections are also known as gated units or gated recurrent units and have a clear resemblance to recent effective elements introduced in RNNs [53]. This technique is able to train a NN with 152 layers while still having lower complexity than VGGNet [54]. It achieves a top-5 error rate of 3.57%, which beats human-level performance on this dataset.

3.3. Classification Method. In general, the study of fingerprints for matching purposes involves a comparison of several print pattern features. These include patterns that are aggregate features of ridges, as well as minutia points that are unique features contained within patterns. The three main classifications for fingerprints include the loop arch and whorl. Therefore, this research categorized the dataset into three main fingerprint patterns (arches, loops, and whorls) that make CNN works efficiently. To categorize the dataset into three main fingerprint patterns, mathematical systems for each pattern were used to identify pattern types.

3.3.1. Whorls. We have considered three mathematical systems of the whorl patterns as mentioned above in the categorization method: concentric, spiral, and composite with "S" core [4]. If the image satisfies one of these systems, then it belongs to whorl patterns. Figure 5 shows the phase portrait of the concentric whorl, Figures 6 and 7 show the phase portrait for the spiral whorl, and Figure 8 shows the phase portrait for the composite with the "S" core. The pattern of the concentric has three equilibrium points, the origin is center, and the points (1, 0) and (−1, 0) are cusps; also, there are two orbits between the endpoints from the above side and below side. On the other hand, the pattern of the spiral has three equilibrium points, the origin is spiral out, and the points (1, 0) and (−1, 0) are cusps. In addition, the image has connection orbits between the origin and the other two points on the left and right sides. In the pattern of the composite with the "S" core, the spiral core is twisted more as "S" with the existence of the cusps.

3.4. Arches. To complete the categorization method on the dataset for the three main patterns in fingerprint (whorls, arches, and loops), we have to state the three mathematical models for arch patterns which are plain, tented, and strong arch; see [55]. The phase portrait of all classes of arch fingerprint has only one singular point at the origin, that is, cusp with varying in the length of its vertical ridges in the middle. The plain pattern is represented by equation (5).

3.5. Matching. After preprocessing, the image is taken as an input to binarization and thinning to be performed. Then, the pattern of the input image is determined. Minutiae is extracted from the thinning image, and matching is conducted in a specific part in the database that is determined during classify input image stage to reduce time complexity function. The type of minutiae can also be classified into ridge bifurcation and ridge ending. Figure 14 shows an example of a ridge ending and bifurcation. A ridge bifurcation minutia is a point where a ridge splits from a single path to two paths, while a ridge ending minutia is a point where a ridge terminates.

Ultimately, if the image does not belong to the previous systems, then the image belongs to the form of loops pattern.

4. Dataset Description

In order to evaluate the proposed model, we use the FVC2004 [56] competitions dataset. These datasets are commonly used as benchmarks for evaluating fingerprint matchers in the context of fingerprint verification. Figure 15 shows a sample of fingerprints of the dataset. FVC2004 was also revealed via mailing lists and online magazines affiliated with biometrics. The creation of four new databases was performed using three commercially available scanners and a synthetic fingerprint generator [57]. The subset of each database consists of 80 fingerprints made available to the participants from 10 fingers.

5. Experimental Results

5.1. Experiments Setting. In this section, we describe a number of experiments conducted to test the proposed

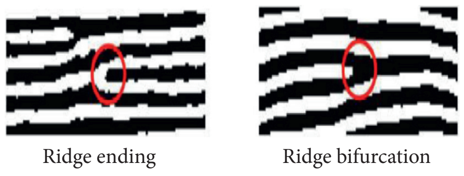


Figure 14: Ridge bifurcation and ridge ending.

Figure 15: Sample of used dataset.

Parameters	GoogleNet	AlexNet	ResNet
Number of convolutional layers	22	8	18
Mini Batch Size	7	7	7
Max Epochs	12	12	12
Initial Learn Rate	1.0000e-04	1.0000e-04	1.0000e-04
Validation Frequency	3	3	3
Training Set	70%	70%	70%
Test Set	30%	30%	30%

Figure 16: Parameters setting for CNN architecture classifier.

model for fingerprint classification and matching. The proposed model implementation runs using a MATLAB environment with a PC containing 4 GB of RAM and 4 Intel cores i5 (2.0 GHz each). In our experiments, we used the FVC2004 dataset to test the proposed model. The performance evaluation of the proposed model is done under three different CNN architectures, ResNet, AlexNet, and GoogleNet. The data have trained with various convolutional layers to find the best architecture of CNN. The CNN-based classifier has been implemented with varying layer numbers, and the GoogleNet has been implemented with 22 layers, where ResNet-18 has been implemented with 18 layers; finally, AlexNet has been implemented with 8 layers. Figure 16 shows the list of parameters setting and their candidate values for different CNN models. In order to make a fair experiment, different CNN models run under the same options.

5.2. Results and Discussion. To analyze the effect of the proposed model, different CNN architecture classifiers were applied. CNN architecture classifiers used in this paper are GoogleNet, AlexNet, and ResNet. The following presents results and a discussion of each CNN architecture classifier.

5.2.1. ResNet Results. The first experiments are performed by applying ResNet-18 to the proposed model. Table 1 shows the results that are obtained from the experiment. The training accuracy used to report during training corresponds to the accuracy of the particular training at the defined iteration. Figure 17 shows training and testing accuracy and training and testing loss with respect to iteration. From the results, we can observe that training accuracy rises to 100% in iterations 15, 69, 87, 99, and 102, which means that there is overfitting in training. The testing accuracy rises to 75% in iterations 99, 102, and 108. The overall validation accuracy is 75%. The time required to build a model for training and testing for all iterations is 55 seconds. Loss is used to optimize a deep learning algorithm. The loss is measured on training and testing, and its meaning is dependent on how well the model in these two sets is doing. It is observable that when the accuracy is high, the loss is low.

5.2.2. AlexNet Results. From the results in Table 2, we can observe that training accuracy rises to 100% in iterations 87, 90, and 102. The testing accuracy rises to 81.25% in iteration 96. The overall validation accuracy is 43.75%. The time required to build a model for training and testing for all iterations is 28 seconds. It is observable that when the accuracy is high, the loss is low. Figure 18 shows accuracy and loss for AlexNet-8, showing "training and testing accuracy" and "training and testing loss" with respect to iteration.

5.2.3. GoogleNet Results. In the experiment of the GoogleNet-22 that is applied to the proposed model, Table 3 shows the results that are obtained from training of the GoogleNet-22 architecture to the proposed model. Figure 19 obtained results from the experiment.

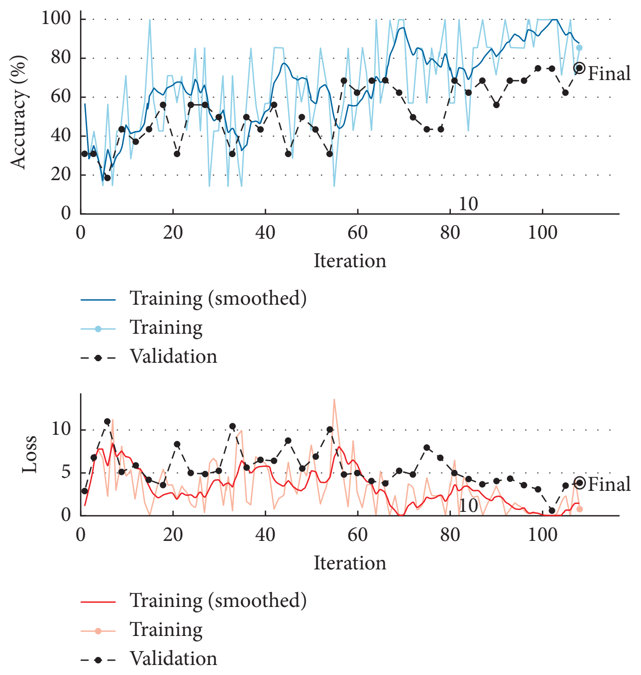
From the results, we can observe that training accuracy rises to 100% in iterations 48, 57, 60, 63, 66, 69, 78, 81, 84, 87, 90, 93, 96, 99, 102, 105, and 108.

The testing accuracy rises to 100% in iterations 60, 63, 66, 75, 78, 90, 93, 96, 99, 102, 105, and 108. The overall validation accuracy is 100%. The time required to build a model for training and testing for all iterations is 262 seconds. It is observable that when the accuracy is high, the loss is low.

5.2.4. Comparison of CNN Architecture Classifier. In addition, it has three main categories of fingerprints (whorls, arches, and loop). After applying three CNNs (ResNet, AlexNet, and GoogleNet) mentioned in the previous section using the MATLAB tool, the accuracy results are 100%, 75%, and 43.75% for GoogleNet, ResNet, and AlexNet, respectively. The training time results are 262, 55, and 28 for GoogleNet, ResNet, and AlexNet, respectively. Figures 20 and 21 show the accuracy and time results, respectively. According to Figure 20, GoogleNet is the accurate one. According to Figure 21, the AlexNet is the faster one. Furthermore, the fingerprint could be utilized to solve one of the most difficult problems in the system and network security: user authentication. For user authentication, time is critical. The embedded access points for trusted data and resources access in HPC systems [58] are one of interesting related work that discussed one possible solution for user

Table 1: ResNet-18 architecture classifier results.

Epoch	Iteration	Time elapsed	Training accuracy (%)	Testing accuracy (%)	Training loss	Testing loss
1	1	00 : 00 : 01	57.14	31.25	1.0842	2.8786
1	3	00 : 00 : 02	42.86	31.25	6.2243	6.7976
1	6	00 : 00 : 03	57.14	18.75	2.1485	11.0027
1	9	00 : 00 : 05	42.86	43.75	8.1363	5.0982
2	12	00 : 00 : 06	42.86	37.50	1.8785	5.8701
2	15	00 : 00 : 08	100.00	43.75	0.0096	4.1785
2	18	00 : 00 : 09	42.86	56.25	5.4363	3.5828
3	21	00 : 00 : 11	57.14	31.25	1.8393	8.3207
3	24	00 : 00 : 13	57.14	56.25	1.2059	4.9842
3	27	00 : 00 : 14	85.71	56.25	0.3168	4.8677
4	30	00 : 00 : 16	42.86	50.00	5.3774	5.2108
4	33	00 : 00 : 17	71.43	31.25	0.4266	10.4761
4	36	00 : 00 : 19	57.14	50.00	1.5931	5.6201
5	39	00 : 00 : 20	42.86	43.75	6.6447	6.5483
5	42	00 : 00 : 22	85.71	56.25	0.6081	6.3882
5	45	00 : 00 : 23	57.14	31.25	6.2282	8.7916
6	48	00 : 00 : 25	57.14	50.00	5.8847	5.5018
6	51	00 : 00 : 26	57.14	43.75	4.9419	6.9097
6	54	00 : 00 : 28	57.14	31.25	4.9069	10.0431
7	57	00 : 00 : 29	57.14	68.75	4.1653	4.778
7	60	00 : 00 : 31	57.14	62.50	2.622	4.9376
7	63	00 : 00 : 32	42.86	68.75	3.5718	4.0243
8	66	00 : 00 : 34	71.43	68.75	2.525	3.7462
8	69	00 : 00 : 35	100.00	62.50	2.06E−06	5.2429
8	72	00 : 00 : 37	57.14	50.00	3.3527	4.7472
9	75	00 : 00 : 38	57.14	43.75	4.4912	7.9152
9	78	00 : 00 : 40	85.71	43.75	2.2775	6.7274
9	81	00 : 00 : 41	57.14	68.75	6.5187	4.9828
10	84	00 : 00 : 43	42.86	62.50	3.2893	4.2506
10	87	00 : 00 : 44	100.0	68.75	4.18E−05	3.6644
10	90	00 : 00 : 46	71.43	56.25	3.4216	4.0034
11	93	00 : 00 : 47	85.71	68.75	0.6548	4.3335
11	96	00 : 00 : 49	85.71	68.75	0.5733	3.5268
11	99	00 : 00 : 50	100.00	75.00	0.006	3.0587
12	102	00 : 00 : 52	100.00	75.00	0.0946	0.5617
12	105	00 : 00 : 54	85.71	62.50	2.2777	3.4835
12	108	00 : 00 : 55	85.71	75.00	0.6971	3.7687

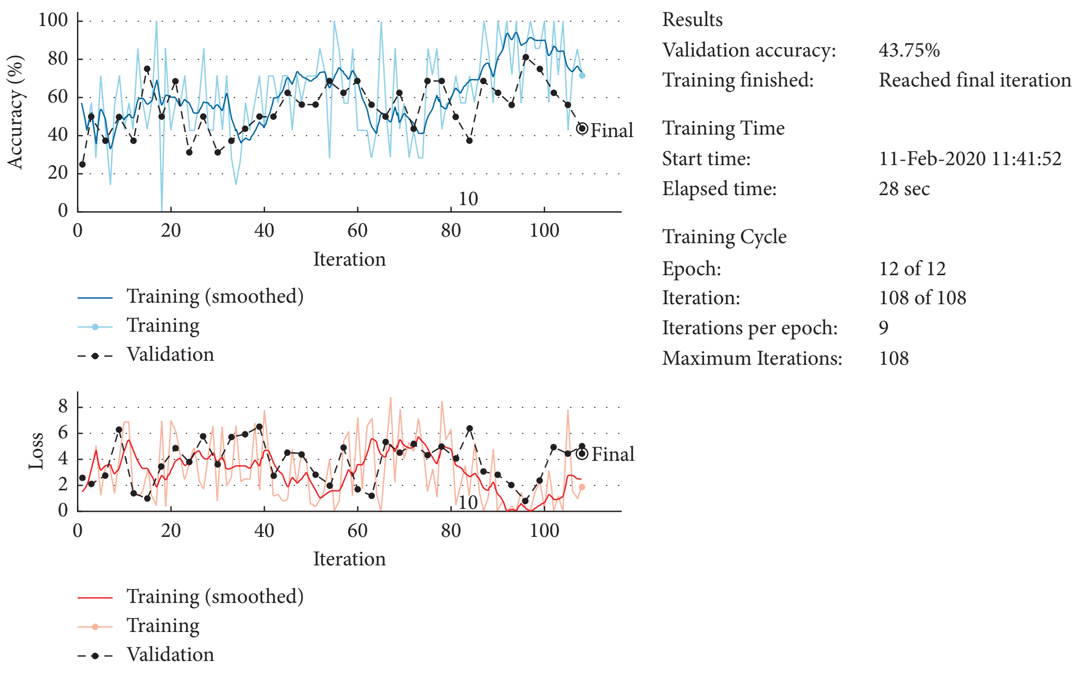


Results	
Validation accuracy:	75.00%
Training finished:	Reached final iteration
Training Time	
Start time:	11-Feb-2020 11:37:52
Elapsed time:	55 sec
Training Cycle	
Epoch:	12 of 12
Iteration:	108 of 108
Iterations per epoch:	9
Maximum Iterations:	108

Figure 17: Accuracy and loss for ResNet-18.

Table 2: AlexNet-8 architecture classifier results.

Epoch	Iteration	Time elapsed	Training accuracy (%)	Testing accuracy (%)	Training loss	Testing loss
1	1	00 : 00 : 01	57.14	25.00	1.4978	2.5875
1	3	00 : 00 : 01	57.14	50.00	3.2016	2.103
1	6	00 : 00 : 02	42.86	37.50	3.628	2.7452
1	9	00 : 00 : 03	71.43	50.00	4.1279	6.3035
2	12	00 : 00 : 04	42.86	37.50	1.2736	1.3775
2	15	00 : 00 : 04	42.86	75.00	3.2486	0.9802
2	18	00 : 00 : 05	0.00	50.00	6.9613	3.4558
3	21	00 : 00 : 06	57.14	68.75	6.2496	4.8568
3	24	00 : 00 : 07	42.86	31.25	3.8846	3.7702
3	27	00 : 00 : 07	85.71	50.00	1.6855	5.8037
4	30	00 : 00 : 08	71.43	31.25	0.809	3.6344
4	33	00 : 00 : 09	28.57	37.50	2.6885	5.7263
4	36	00 : 00 : 09	57.14	43.75	2.5091	5.9148
5	39	00 : 00 : 10	71.43	50.00	1.6331	6.5207
5	42	00 : 00 : 11	71.43	50.00	1.151	2.7441
5	45	00 : 00 : 12	71.43	62.50	0.9064	4.4997
6	48	00 : 00 : 12	57.14	56.25	4.6229	4.368
6	51	00 : 00 : 13	71.43	56.25	0.3475	2.8
6	54	00 : 00 : 14	28.57	68.75	3.9778	1.9498
7	57	00 : 00 : 15	57.14	62.50	5.3242	4.9202
7	60	00 : 00 : 15	42.86	68.75	7.2394	1.6834
7	63	00 : 00 : 16	28.57	56.25	7.191	1.1742
8	66	00 : 00 : 17	57.14	50.00	3.7439	5.3485
8	69	00 : 00 : 18	42.86	62.50	7.8516	4.5162
8	72	00 : 00 : 19	42.86	43.75	4.5165	5.194
9	75	00 : 00 : 19	85.71	68.75	1.4853	4.3325
9	78	00 : 00 : 20	42.86	68.75	8.5128	4.982
9	81	00 : 00 : 21	71.43	50.00	0.4735	4.0692
10	84	00 : 00 : 22	71.43	37.50	1.1465	6.3691
10	87	00 : 00 : 23	100.00	68.75	0.0377	3.0489
10	90	00 : 00 : 24	100.00	62.50	0.0272	2.7935
11	93	00 : 00 : 24	71.43	56.25	0.4595	2.0301
11	96	00 : 00 : 25	85.71	81.25	0.4496	0.7833
11	99	00 : 00 : 26	85.71	75.00	2.3075	2.355
12	102	00 : 00 : 27	100.00	62.50	0.0001	4.9323
12	105	00 : 00 : 28	42.86	56.25	7.8079	4.4485
12	108	00 : 00 : 28	71.43	43.75	1.8536	5.0001

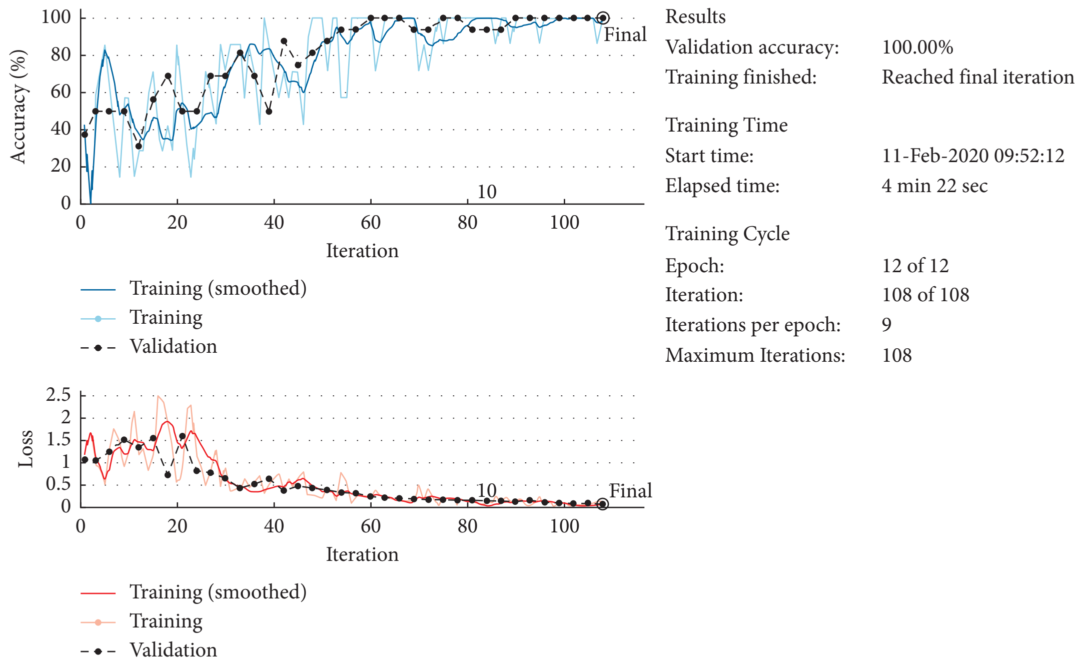


Results	
Validation accuracy:	43.75%
Training finished:	Reached final iteration
Training Time	
Start time:	11-Feb-2020 11:41:52
Elapsed time:	28 sec
Training Cycle	
Epoch:	12 of 12
Iteration:	108 of 108
Iterations per epoch:	9
Maximum Iterations:	108

Figure 18: Accuracy and loss for AlexNet-8.

TABLE 3: GoogleNet-22 architecture classifier results.

Epoch	Iteration	Time elapsed	Training accuracy (%)	Testing accuracy	Training loss	Testing loss
1	1	00 : 00 : 03	42.86	37.50%	1.1722	1.0717
1	3	00 : 00 : 08	57.14	50.00%	0.9451	1.0425
1	6	00 : 00 : 16	57.14	50.00%	1.3488	1.2365
1	9	00 : 00 : 23	57.14	50.00%	0.8974	1.4963
2	12	00 : 00 : 30	28.57	31.25%	1.179	1.332
2	15	00 : 00 : 38	71.43	56.25%	1.1686	1.5411
2	18	00 : 00 : 45	42.86	68.75%	1.7907	0.7173
3	21	00 : 00 : 52	57.14	50.00%	0.7581	1.5879
3	24	00 : 00 : 59	42.86	50.00%	1.1363	0.8127
3	27	00 : 01 : 06	57.14	68.75%	0.8824	0.7707
4	30	00 : 01 : 13	71.43	68.75%	0.8792	0.6418
4	33	00 : 01 : 20	85.71	81.25%	0.402	0.432
4	36	00 : 01 : 27	71.43	b 68.75%	0.5519	0.5142
5	39	00 : 01 : 34	85.71	50.00%	0.5002	0.6351
5	42	00 : 01 : 42	71.43	87.50%	0.3781	0.3745
5	45	00 : 01 : 49	71.43	75.00%	0.6498	0.4651
6	48	00 : 01 : 56	100.00	81.25%	0.2742	0.4266
6	51	00 : 02 : 03	71.43	87.50%	0.4637	0.3808
6	54	00 : 02 : 10	57.14	93.75%	0.7768	0.3296
7	57	00 : 02 : 18	100.00	93.75%	0.2694	0.313
7	60	00 : 02 : 26	100.00	100.00%	0.1386	0.2526
7	63	00 : 02 : 33	100.00	100.00%	0.2667	0.2256
8	66	00 : 02 : 40	100.00	100.00%	0.2094	0.2087
8	69	00 : 02 : 47	100.00	93.75%	0.0676	0.1964
8	72	00 : 02 : 54	71.43	93.75%	0.4086	0.1807
9	75	00 : 03 : 01	85.71	100.00%	0.2515	0.1776
9	78	00 : 03 : 10	100.00	100.00%	0.0403	0.1587
9	81	00 : 03 : 17	100.00	93.75%	0.2288	0.1578
10	84	00 : 03 : 25	100.00	93.75%	0.0995	0.1556
10	87	00 : 03 : 32	100.00	93.75%	0.0399	0.1481
10	90	00 : 03 : 40	100.00	100.00%	0.1824	0.1303
11	93	00 : 03 : 47	100.00	100.00%	0.0628	0.1631
11	96	00 : 03 : 54	100.00	100.00%	0.0656	0.0656
11	99	00 : 04 : 01	100.00	100.00%	0.1219	0.0998
12	102	00 : 04 : 08	100.00	100.00%	0.0187	0.0962
12	105	00 : 04 : 15	100.00	100.00%	0.0499	0.1034
12	108	00 : 04 : 22	100.00	100.00%	0.1435	0.0834



Results	
Validation accuracy:	100.00%
Training finished:	Reached final iteration
Training Time	
Start time:	11-Feb-2020 09:52:12
Elapsed time:	4 min 22 sec
Training Cycle	
Epoch:	12 of 12
Iteration:	108 of 108
Iterations per epoch:	9
Maximum Iterations:	108

FIGURE 19: Accuracy and loss for GoogleNet-22.

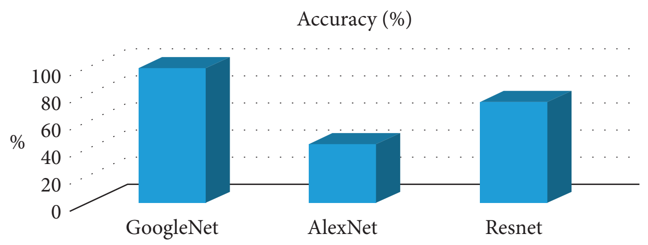


Figure 20: Validation accuracy of three CNN architecture classifiers.

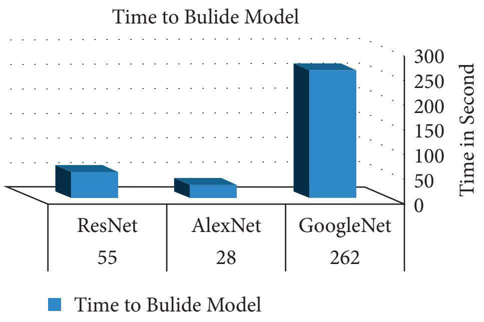


Figure 21: Model building time for three CNN architecture classifiers.

authentication in network security and it is a hardware implementation in hight-performance computing field. A free database biometric authentication system is presented, with a tamper-resistant smartcard serving as the storage device. Furthermore, fingerprint processing units have been incorporated in hardware, resulting in embedded access points capable of hiding various biometric authentication system attack points. Its access point prototype, which was created with FPGA technology, a smartcard read/write device, and the AES algorithm to encrypt the biometric template, yielded intriguing results in terms of recognition rates. This interesting related work could be integrated with this proposed method as future work and other many network [59, 60] problems to produce a new efficient method.

In CNNs, adding more layers leads to extracting more features that indicate that highly accurate results could be achieved with more computation time. The main goal of the GoogleNet architecture was to get high accuracy so that the motivation for the GoogleNet is creating Inception CNN module to make a deeper CNN by adding 22 layers and almost 12 times fewer parameters than AlexNet. Thus, the highly accurate results could be achieved with more computation time. Second, the ResNet in this research had 18 layers, so it is between AlexNet and GoogleNet in the accuracy and time computation. Finally, AlexNet had 8 layers, so it is faster with less accurate.

6. Conclusion

In this paper, we have proposed a fingerprint classification and matching model based on a mathematical model using different CNN architectures. To the best of our knowledge, this is the first such attempt to tackle complex fingerprint classification issues using CNN. In our proposed model, the fingerprint image is classified into three main categories arch, loop, and whorl, and matching is performed based on bifurcation minutiae extraction. Binarization and thinning model has been used in order to improve image quality. In this paper, we have implemented and tested the proposed model with three CNN architectures, namely, GoogleNet, AlexNet, and ResNet. From the obtained results, GoogleNet provides better results in terms of accuracy. Moreover, AlexNet provides better results in terms of time training. On the other hand, the usage of reconfigurable hardware devices is a viable solution to some of the issues that plague software-based solutions. Indeed, they enable the creation of embedded and tamper-resistant devices, which are particularly helpful in contexts where security is critical. Novel hardware implementation in the field of high-performance computing is presented in [58] that could be integrated with the proposed method to produce a more effective and efficient algorithm.

Data Availability

The data used to support the findings of this paper are available online and FVC2004 free Database fingerprint dataset. They are at "http://bias.csr.unibo.it/fvc2004/" website [6].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] C. O. Folorunso, O. S. Asaolu, and O. P. Popoola, "A review of voice-base person identification:state-of-the-art," *Covenant Journal of Engineering Technology (CJET)*, vol. 3, no. 1, 2019.
[2] C. Lin and A. Kumar, "A CNN-based framework for comparison of contactless to contact-based fingerprints," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 662–676, 2019.
[3] F. Galton, *Finger Prints*, McMillan & Co., London, UK, 1892.
[4] I. Jawarneh and N. Alsharman, "The mathematical model and deep learning features selection for whorl fingerprint classifications," *InterNational Journal of Computational Intelligence Systems*, vol. 14, pp. 1208–1216, 2021.
[5] E. R. Henry, *Classification and Uses of finger Prints*, HM Stationery Office, Richmond, UK, 1905.
[6] K. Anil, "Jain, salil prabhakar, student member, and lin Hong. A multichannel approach to fingerprint classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 348–359, 1999.
[7] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*, Springer Science & Business Media, Berlin, Germany, 2009.
[8] A. K. Jain, H. Lin Hong, S. Pankanti, and R. Bolle, "An identity-authentication system using fingerprints," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1365–1388, 1997.
[9] A. Lindoso, L. Entrena, J. Liu-Jimenez, and E. San Millan, "Correlation- based fingerprint matching with orientation field alignmen," *Lecture Notes in Computer Science*, vol. 4642, 2007.

[10] N. K. Ratha, K. Karu, S. Chen, and A. K. Jain, "A real-time matching system for large fingerprint databases," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 799–813, 1996.

[11] S. M. Mohamed and H. Nyongesa, "Automatic fingerprint classification system using fuzzy neural techniques," in *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE'02. Proceedings (Cat. No. 02CH37291)*, Honolulu, USA, May 2002.

[12] Q. Zhang and H. Yan, "Fingerprint classification based on extraction and analysis of singularities and pseudo ridges," *Pattern Recognition*, vol. 37, no. 11, pp. 2233–2243, 2004.

[13] J.-H. Hong, J.-K. Min, U.-K. Cho, and S.-B. Cho, "Fingerprint classification using one-vs-all support vector machines dynamically ordered with naı¨ve Bayes classifiers," *Pattern Recognition*, vol. 41, no. 2, pp. 662–671, 2008.

[14] R. Wang, C. Han, Y. Wu, and T. Guo, "Fingerprint classification based on depth neural network," 2014, https://arxiv.org/abs/1409.5188.

[15] S. Kouamo and C. Tangha, "Fingerprint recognition with artificial neural networks: application to e-learning," *Journal of Intelligent Learning Systems and Applications*, vol. 08, no. 02, pp. 39–49, 2016.

[16] C. Militello, L. Rundo, S. Vitabile, and V. Conti, "Fingerprint classification based on deep learning approaches: experimental findings and comparisons," *Symmetry Plus*, vol. 13, no. 5, 2021.

[17] D. Peralta, S. Garc´ıa, J. M. Benitez, and F. Herrera, "Minutiae-based fingerprint matching decomposition: methodology for big data frameworks," *Information Sciences*, vol. 408, pp. 198–212, 2017.

[18] W. Lee, S. Cho, H. Choi, and J. Kim, "Partial fingerprint matching using minutiae and ridge shape features for small fingerprint scanners," *Expert Systems with Applications*, vol. 87, pp. 183–198, 2017.

[19] G. T. Candela, P. Grother, C. Watson, R. A. Wilkinson, and C. Wilson, *Pcasys- a Pattern-Level Classification Automation System for Fingerprints — Nist*, NIST, Maryland, USA, 1995.

[20] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni, "Fingerprint classification by directional image partitioning," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, pp. 402–421, 1999.

[21] J. Li, Y. Wei-Yun, and H. Wang, "Combining singular points and orientation image information for fingerprint classification," *Pattern Recognition*, vol. 41, pp. 353–366, 2008.

[22] K. Karu and A. K. Jain, "Fingerprint classification," *Pattern Recognition*, vol. 29, no. 3, pp. 389–404, 1996.

[23] H. O. Nyongesa, S. Al-Khayatt, S. M. Mohamed, and M. Mahmoud, "Fast robust fingerprint feature extraction and classification," *Journal of Intelligent and Robotic Systems*, vol. 40, no. 1, pp. 103–112, 2004.

[24] N. K. Ratha, K. Karu, S. Shaoyun Chen, and A. K. Jain, "A real-time matching system for large fingerprint databases," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 799–813, 1996.

[25] S. Shah and P. Sastry, "Fingerprint classification using a feedback-based line detector," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 34, no. 85–94, 2004.

[26] Y. Yao, G. Marcialis, M. Pontil, P. Frasconi, and F. Roli, "Combining flat and structured representations for fingerprint classification with recursive neural networks and support vector machines," *Pattern Recognition*, vol. 36, pp. 397–406, 2002.

[27] V. Conti, C. Militello, F. Sorbello, and S. Vitabile, "Introducing pseudo- singularity points for efficient fingerprints classification and recognition," in *Proceedings of the The 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2010)*, Krakow, Poland, February 2010.

[28] S Dass and A Jain, "Fingerprint classification using orientation field flow curves," in *Proceedings of the ICVGIP 2004, Proceedings of the Fourth Indian Conference on Computer Vision, Graphics & Image Processing*, Kolkata, India, December 2004.

[29] R. Doroz, K. Wrobel, and P. Porwik, "An accurate fingerprint reference point determination method based on curvature estimation of separated ridges," *International Journal of Applied Mathematics and Computer Science*, vol. 28, no. 1, pp. 209–225, 2018.

[30] K. Castillo-Rosado and J. Hernández-Palancar, "Latent fingerprint matching using distinctive ridge points," *Informatica*, vol. 30, no. 3, pp. 431–454, 2019.

[31] W. Zhang, L. L. Tang, Q. Li, A. Liu, and M. L. T. Lee, "Order-restricted inference for clustered ROC data with application to fingerprint matching accuracy," *Biometrics*, vol. 76, no. 3, pp. 863–873, 2020.

[32] R. Gupta, M. Khari, D. Gupta, and R. Crespo, "Fingerprint image enhancement and reconstruction using the orientation and phase reconstruction," *Informing Science*, vol. 530, pp. 201–218, 2020.

[33] N. Kaushal and P. Kaushal, "Human identification and fingerprints: a review," *Journal of Biometrics & Biostatistics*, vol. 2, no. 4, 2011.

[34] A. Saleh and A. Mahmood, "A framework for designing the architectures of deep convolutional neural networks," *Entropy*, vol. 19, no. 6, 2017.

[35] I. Jawarneh and N. Alsharman, "The classification of arch fingerprint using mathematical model and deep learning features selection," *International Journal of Mathematics and Computer Science*, vol. 17, pp. 289–307, 2022.

[36] E. Erwin, N. N. B. Karo, A. Y. Sari, and N. Aziza, "The enhancement of fingerprint images using gabor filter," *Journal of Physics: Conference Series*, vol. 1196, no. 1, 2019.

[37] *Nobuyuki Otsu*, vol. 9, 1979.

[38] T. H. Nguyen, Y. Wang, and R. Li, "An improved ridge features extraction algorithm for distorted fingerprints matching," *Journal of Information Security and Applications*, vol. 18, no. 4, pp. 206–214, 2013.

[39] N. Alsharman and I. Jawarneh, "Googlenet cnn neural network towards chest CT-coronavirus medical image classification," *Journal of Computer Science*, vol. 16, no. 5, pp. 620–625, 2020.

[40] M. Heikkil¨a, M. Pietik¨ainen, and C. Schmid, "Description of interest regions with local binary patterns," *Pattern Recognition*, vol. 42, no. 3, pp. 425–436, 2009.

[41] S. Minaee, E. Azimi, and A A Abdolrashidi, "Fingernet: pushing the limits of fingerprint recognition using convolutional neural network," *CoRR, abs/*, 2019.

[42] B. Herbert, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[43] D. G. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157, 1999.

[44] M. Tzelepi and A. Tefas, "Deep convolutional learning for content based image retrieval," *Neurocomputing*, vol. 275, pp. 2467–2478, 2018.

[45] L. Deng and Yu Dong, "Deep learning: methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3-4, pp. 197–387, 2014.

[46] W. Wang, Y. Yang, X. Wang, W. Wang, and J. Li, "Development of convolutional neural network and its application in image classification: a survey," *Optical Engineering*, vol. 58, 2019.

[47] A. Khan, A. Sohail, U. Zahoora, and A. S. Saeed, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455–5516, Apr 2020.

[48] Y. Lecun, L. D. Jackel, B. Leon, C. Cartes, J. S. Denker, and H. Drucker, "Learning algorithms for classification: a comparison on handwritten digit recognition," in *Neural Networks: the Statistical Mechanics Perspective*, pp. 261–276, World Scientific, Singapore, 1995.

[49] A. Krizhevsky, I. Sutskever, and E. Hinton Geoffrey, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 60, no. 6, pp. 1097–1105, 2017.

[50] S. Christian, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, June 2015, https://doi.org/10.1109/CVPR.2015.7298594.

[51] M. Lin, Q. Chen, and S. Yan, *Network in Network*, 2014.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, June 2016.

[53] A. L. Caterini and D. E. Chang, *Recurrent Neural Networks*, 2018.

[54] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," International Conference on Learning Representations, San Diego, CA, USA, 2015, https://arxiv.org/abs/1409.1556.

[55] I. Jawarneh and N. Alsharman, "A mathematical model for arch fingerprint," 2020, https://arxiv.org/abs/2003.00308.

[56] R. Cappelli, D. Maio, D. Maltoni, J. L. Wayman, and A. K. Jain, "Performance evaluation of fingerprint verification systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 1, pp. 3–18, 2006.

[57] R. Cappelli, D. Maio, and D. Maltoni, "Synthetic fingerprint-database generation," *International Conference on Pattern Recognition*, vol. 3, pp. 744–747, 2002.

[58] C. Militello, V. Conti, S. Vitabile, and F. Sorbello, "Embedded access points for trusted data and resources access in hpc systems," *The Journal of Supercomputing*, Springer Netherlands, vol. 55, no. 1, pp. 4–27, 2011.

[59] A. Nagar, K. Nandakumar, and A. K. Jain, "Securing fingerprint template: fuzzy vault with minutiae descriptors," in *Proceedings of the 2008 International Conference for Pattern Recognition*, Tampa, USA, December 2008.

[60] A. Saaidah, A. Omar, L. Al-Qaisi, and M. Mohammed Kamel, "An efficient design of rpl objective function for routing in internet of things using fuzzy logic," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 8, 2019.