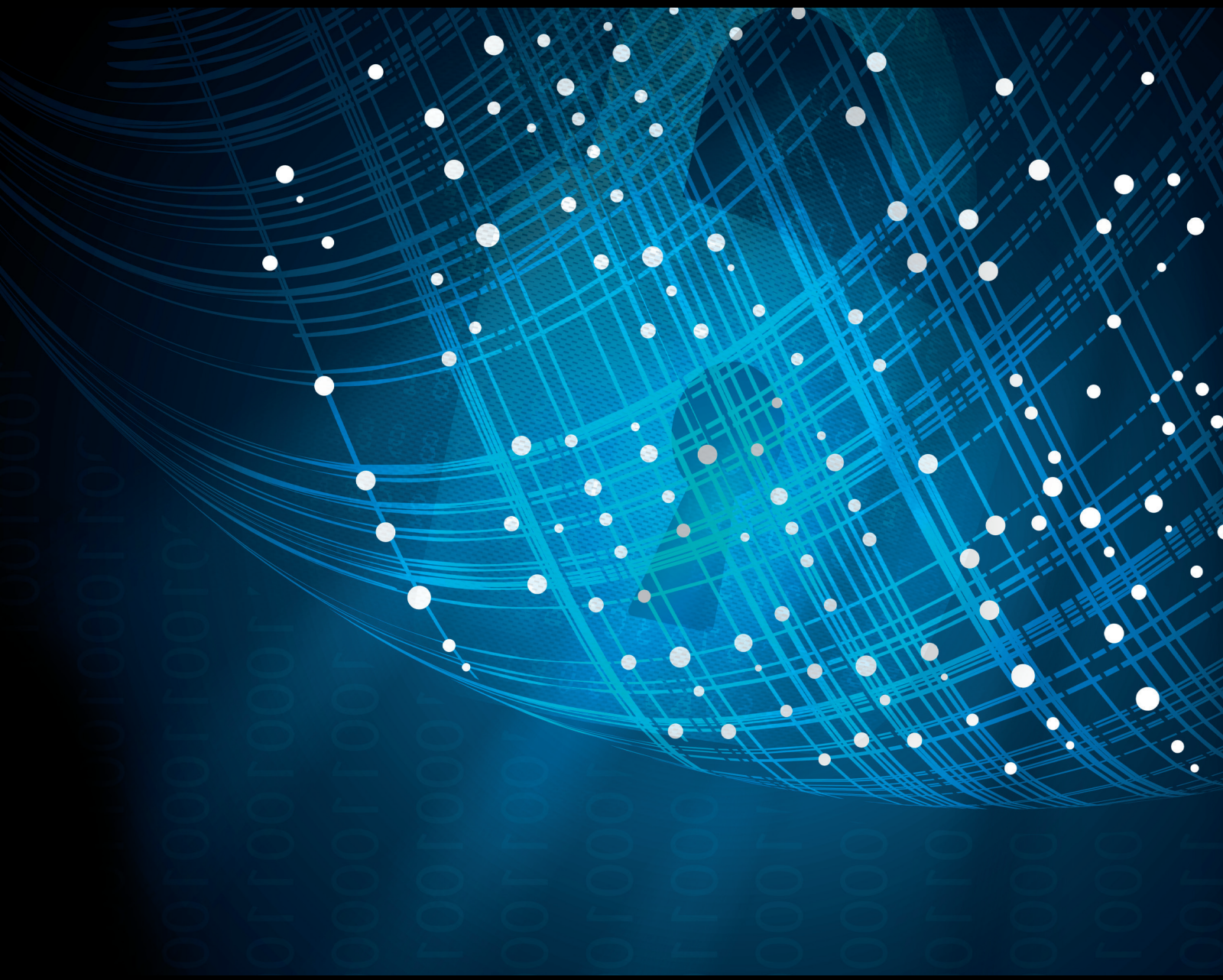
Security and Communication Networks

Security Threats to Artificial Intelligence-Driven Wireless Communication Systems

Lead Guest Editor: Huaming Wu
Guest Editors: Xiaolong Xu, Kaitai Liang, Yuan Yuan, and Junqing Zhang



Security Threats to Artificial Intelligence-Driven Wireless Communication Systems

Security and Communication Networks

Security Threats to Artificial Intelligence-Driven Wireless Communication Systems

Lead Guest Editor: Huaming Wu
Guest Editors: Xiaolong Xu, Kaitai Liang, Yuan Yuan, and Junqing Zhang

Contents

Hindawi
Security and Communication Networks
Volume 2021, Article ID 8872140, 13 pages
https://doi.org/10.1155/2021/8872140

Research Article

Hardware Sharing for Channel Interleavers in 5G NR Standard

Xiaokang Xiong, Yuhang Dai, Zhuhua Hu, Kejia Huo, Yong Bai, Hui Li, and Dake Liu

School of Information and Communication Engineering and State Key Laboratory of Marine Resource Utilization in South China Sea, Hainan University, Haikou 570228, China

Correspondence should be addressed to Zhuhua Hu; eagler_hu@hainu.edu.cn and Yong Bai; bai@hainanu.edu.cn

Received 9 April 2020; Revised 16 June 2020; Accepted 7 January 2021; Published 27 January 2021

Academic Editor: Huaming Wu

Interleaver module is an important part of modern mobile communication system. It plays an important role in reducing bit error rate and improving transmission efficiency over fading channels. In 5G NR (5th Generation New Radio) standards, LDPC (low-density parity-check) and polar channel codes are employed for data channels and control channels, respectively. If multiple interleavers are implemented separately for them, the cost increases significantly. To address this issue, a hardware multiplexing scheme for channel interleavers based on LDPC and polar codes is proposed in this paper. Firstly, the formulas for the processes of the control channel interleaving and data channel interleaving are derived with respect to 5G NR standard. Then, the hardware implementation structures of the two interleavers are given. Subsequently, hardware reuse is proposed by sharing the similar or identical parts between the two hardware structures. Simulation results verify the correctness of our proposed scheme and demonstrate that it can realize the hardware sharing of the two kinds of channel interleavers to reduce the cost of silicon.

1. Introduction

In the modern mobile communication, some important technologies are used, such as interleaving [1], offloading [2], spectrum sensing [3, 4], partitioning [5], hardware reusing, and resource sharing and allocation [6, 7]. Specifically, channel interleaving technology has been widely used. Channel interleaving aims to distribute transmitted bits in time to achieve desirable bit error distribution to counter the effects of fading channels. The interleaver can change the permutation of the signal bit stream to the utmost without changing the information content. Therefore, interleaver can maximize the dispersion of continuous error bits generated by bursts in the process of transmission. In this way, the error correction and error detection capabilities of the receiver can be improved. In the traditional LUT (lookup table) based interleaving and deinterleaving scheme, a large amount of silicon is used with high cost. Therefore, it is important to reuse the hardware for different types of interleavers to reduce the cost of silicon.

At present, the hardware equipment based on multimode and fast-switching has been studied for channel interleavers in WLAN (wireless local area network, which includes IEEE 802.11a/b/g and IEEE 802.11n standards), WiMAX (Worldwide Interoperability for Microwave Access, which includes IEEE 802.16e standard), 3GPP-WCDMA (3rd Generation Partnership Project-Wideband Code-Division Multiple Access), 3GPP-LTE (3GPP-long-term evolution), and DVB-T/H (Digital Video Broadcasting-Terrestrial/Handheld) standards [1]; multistandard hardware interleaver structure was proposed for HSPA (High Speed Packet Access) evolution, 3GPP-LTE, WiMAX, WLAN, and DVB-T/H in [8]. A parallel architecture for decoding reconfigurable interleavers was proposed to support HSPA evolution, DVB-SH (DVB-Satellite Services to Handhelds), 3GPP-LTE, and WiMAX standards [9]. The issues of address conflicts for hardware sharing were analyzed and resolved in [10]. Among these multistandard interleaver implementations, it is common to simplify and improve the interleaver algorithm of various standards such that the hardware implementation structure becomes simple and easy to reuse [11]. Then, the identical hardware structure is reused by careful comparison to reduce the cost of silicon for multistandards [12]. Although these works cover 2G, 3G, and even 4G standards [13], the latest 5G standard has not

been studied in them. Therefore, with respect to the 5G NR standard 3GPP TS 38.212 [14], this paper proposes a scheme of hardware reuse and cost-saving for polar-encoded channel interleaver [15] and LDPC-encoded channel interleaver [16]. We first derive the formulas for the interleaving schemes of data channel and control channel in 5G NR standard. Then, we design the corresponding hardware structure for them. Next, by comparative analysis, we obtain a multiplexing structure with a reused module to achieve the hardware sharing of two-channel interleavers.

The contributions of this paper are as follows:

(1) The interleaving schemes of data channel and control channel in 5G NR standard are formulized, and the corresponding hardware structures are given.

(2) The hardware structure diagrams of two kinds of channel interleaver are compared, and the hardware sharing structure is given to realize low-cost implementation.

The structure of the remaining parts of this paper is as follows: in Section 2, we introduce the interleaver schemes for LDPC and polar codes channel. In Section 3, we derive the interleaving formulas of two kinds of channels to facilitate the subsequent interleaver reuse. In addition, we give the hardware structure designs of two interleavers. Then, according to the derived formulas, we also give the hardware structure after hardware sharing. Subsequently, the feasibility verification of the final design is given. Finally, Section 4 summarizes the work of this paper.

2. Brief Introduction of Channel Interleavers in 5G NR Standard

In this paper, our work is mainly carried out in accordance with the final standard of 3GPP R15, which is the first version of the 5G standard and meets the part of IMT-2020 (International Mobile Telecommunications-2020) requirements of ITU (International Telecommunication Union). The interleaving method used in the 5G standard is the optimal conclusion after repeated discussion and demonstration [14, 17].

Channel interleaving mainly includes two modes: control channel interleaving and data channel interleaving. This paper focuses on the hardware sharing of these two interleaving methods in 5G NR uplink and downlink. The position of our work in the 5G NR standard is highlighted in Figure 1.

2.1. Interleaver for Data Channel. LDPC code is a new type of error correction code. Its performance in mobile channel is improved compared with turbo code. Even without interleaver, the error correction ability of irregular LDPC code is better than turbo code. Therefore, the LDPC code is listed as one of the candidate schemes in 5G communications. In addition, the simulation results show that LDPC has good performance in all block lengths and code rates, and the complexity is relatively low [18]. In the latest 5G standard, the construction, coding, and interleaving scheme of parity matrix H of LDPC code is specified. In 5G standard, QC-LDPC (quasi-cyclic-LDPC) code is adopted. QC-LDPC code belongs to a structured irregular LDPC code [19], which is composed of basic matrix H_b and lifting factor Z. In 5G standard, two basic matrices (i.e., BG1 and BG2) are determined. Two basic matrices have eight basic matrices, respectively, and they have different dimensions. The corresponding basic matrix [20] is selected according to the size and code rate of transmission block [21]. After the basic matrix is determined, the lifting factor is selected, and then, the basic matrix is modified according to the lifting factor to get the modified parity matrix H. Finally, according to the check matrix H, the encoded code word is directly obtained.

In essence, interleaver is a device which can change the information distribution structure without changing the information content. It is employed to make the burst errors generated in the process of channel transmission decentralized. The LDPC code interleaving scheme adopted in 5G standard is bit interleaving with block interleaver [22]. As shown in Figure 2, the interleaving method is to read the input sequence into a matrix by rows and then read out by columns. The process of deinterleaving is the opposite operation, i.e., read the interleaved sequence into the matrix by columns and then read it out by rows. The matrix is determined by the length and interleaving depth of the input sequence. The number of rows in the matrix is the interleaving depth, and the number of columns is the length of the input sequence divided by the interleaving depth. The interleaving depth is related to the modulation order. There are five modulation schemes specified in 5G NR standard, i.e., BPSK (binary phase shift keying), QPSK (quadrature phase shift keying), 16QAM (quadrature amplitude modulation), 64QAM, and 256QAM. The corresponding modulation orders are 1, 2, 4, 6, and 8, respectively. For example, if 16QAM modulation is used and the input sequence length is 8000 symbols, then the matrix size is 4×2000. After adding the interleaving function, the coding performance has a corresponding improvement, as shown in Figure 3.

2.2. Interleaver for Control Channel. Due to its low complexity of encoding and decoding, the polar code has become a research hotspot of error correction code. The core of polar code construction is related to the channel polarization. In the process of coding, each subchannel is made to show a different reliability [23, 24]. When the length of information code to be transmitted continues to increase, some channels tend to the perfect channel with capacity close to 1 (error-free code), and the other channels tend to the pure noise channel with capacity close to 0. On this basis, we can select those channels whose capacity is close to 1 to transmit information directly to approximate the channel capacity. In addition, the polar code is the only coding scheme that can be strictly proved to achieve the Shannon limit.

The construction of polar code is composed of error detection, code matrix generation, sequence, rate matching [25], and interleaving. In the interleaving part, we can also divide it into two steps, interleaving before coding and interleaving after coding. Interleaving before coding is applicable to 5G-NR DCI (downlink control information), and there is no upstream interleaving; the interleaving after coding is applicable to 5G-NR UCI (uplink control information), and there is no downstream interleaving. This paper discussed interleaving of UCI.

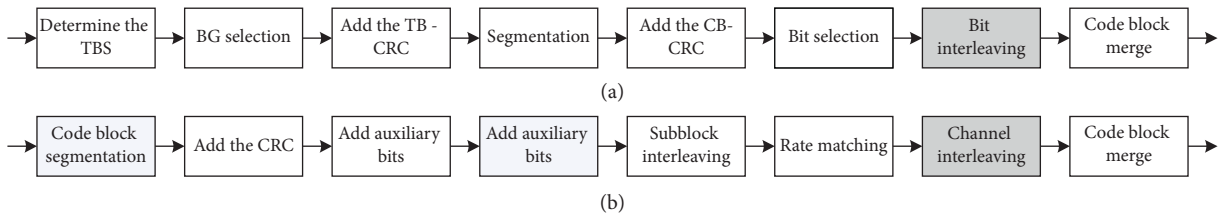


Figure 1: The position of our work in the 5G NR standard. (a) LDPC encoding process of PUSCH and PDSCH in 5G. (b) Polar encoding process for UCI in 5G.

In Introduction, we have briefly introduced that the interleaving is to disrupt the information structure without changing the information content and reduce the relevance between information bits to improve the resistance to burst interference. In the interleaving of UCI, the right triangle interleaving method is specified [26], as shown in Figure 4. In this method, we assume that the storage unit is an isosceles right triangle with a right angle side length of P, and the side length P is clearly defined in 3GPP, that is,

$$P \times \frac{P}{2} \leq 8192. \tag{1}$$

In 3GPP, the interleaver has a maximum of 8192 bits [27]. In this case, M is set as the number of bits after rate matching. At this time, it requires

$$P \times \frac{P+1}{2} \geq M. \tag{2}$$

When the equation takes the equal sign, we write the information into the interleaver line by line and then read it out in the order of columns. When the equation takes the greater than sign, there is still some unused space after all the information is loaded into the interleaver. At this time, we load dummy elements (nulls) into the interleaver and discard the dummy elements when reading out by columns. From the above process, we can see that this is similar to the interleaving process of block interleaver [28], but the rules of interleaving are not unitary because the number of rows in each column or the number of columns in each row is different. We can find that, after the right triangle interleaving, the spacing between each adjacent information data becomes P, P–1, and P–2, and they are not equidistant. With the right triangle interleaving theory, we use Matlab to simulate. We set up comparison groups; that is, one group contains isosceles right triangle interleaving method, while the other group does not. As shown in Figure 5, we can find that the performance for reducing the bit error rate is improved after using the interleaver. Among them, the red dotted line does not use the interleaving function, while the blue line uses the interleaving function.

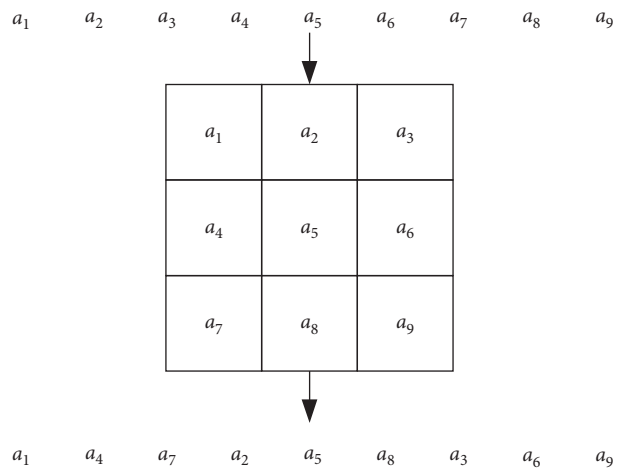


Figure 2: Interleaving process of row/column interleaver.

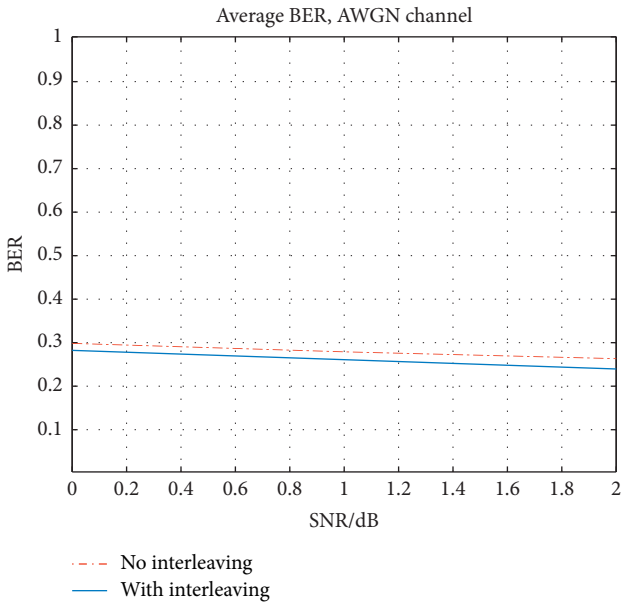


Figure 3: Comparison of BER before and after interleaving.

3. Multiplexing of Two Interleavers in 5G NR Standards

3.1. Formula Representation of Standardized Interleavers

3.1.1. Formula Representation of LDPC-Coded Data Channel Interleaver. The channel interleaving process based on LDPC encoding in 5G is given in Table 1.

In Table 1, E is the length of the input sequence, Q_m is the modulation order, e is the sequence before interleaving, and f is the sequence after interleaving.

From Table 1, the essence of the whole interleaving process is to write the input sequence $x(n)$ in rows and read the interleaving sequence as $f(n)$ in columns. Hence,

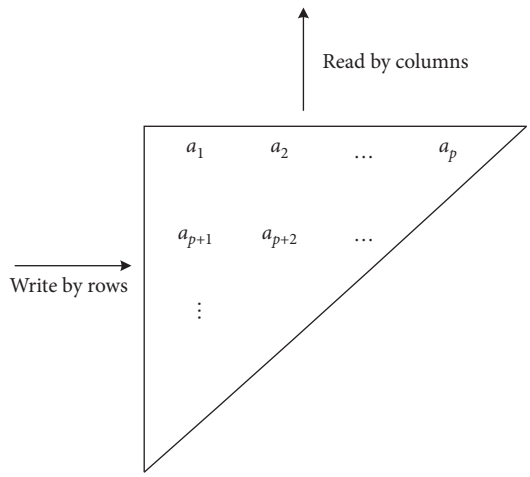


Figure 4: Right triangle interleaving process.

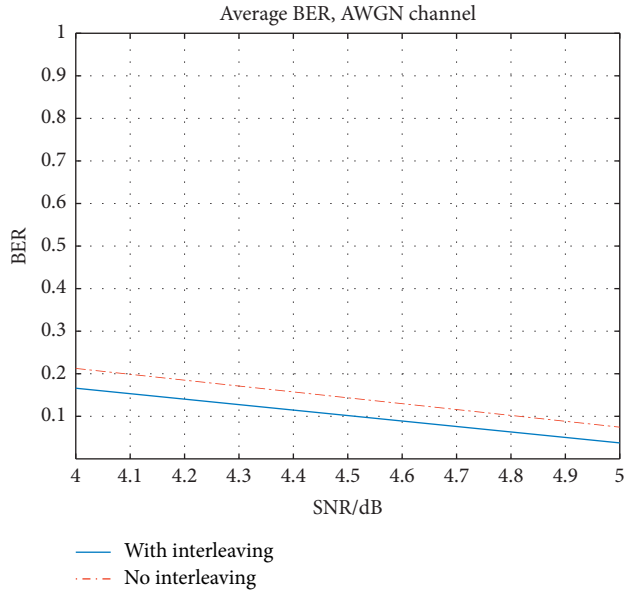


Figure 5: Comparison of BER before and after interleaving.

Table 1: The process of data channel interleaver.

```
for j = 0 to E/Q_m − 1
  for i = 0 to Q_m − 1
    f_{i+j·Q_m} = e_{i·(E/Q_m+j)}
  end for
end for
```

the realization of interleaver is to find the corresponding relationship between $f(n)$ and $x(n)$, that is, the interleaving address. Since the interleaving process can be equivalent to that in a matrix, the parameter i in the interleaving process can be equivalent to a row parameter, j can be equivalent to a column parameter, and the row and column correspond to the row and column in the matrix, respectively, where the range of i is $[0, Q_m - 1]$, and the range of j is $[0, E/Q_m - 1]$. Then, set the interleaving result as $J_{i,j}$, which is

$$f_{i+j\cdot Q_m} = e_{J_{i,j}}, \tag{3}$$

where j is the outer loop and its value increases from 0 to $E/Q_m - 1$ and i is the inner loop whose value increases from 0 to Q_m. Thus, we can get the value of $i + j \times Q_m$ is 0, 1, 2, 3, ..., $E - 1$. That is, with the increase in i and j, the value of $J_{i,j}$ is the position of the elements in the output sequence corresponding to the sequence before interleaving. For example, the calculated value of $J_{i,j}$ is written as [1–4] in order. If the input sequence is e, then the output sequence is [$e(4)$, $e(3)$, $e(2)$, $e(1)$]. In the original process,

$$J_{i,j} = \frac{i \cdot E}{Q_m + j}. \tag{4}$$

However, because the formula is not convenient for subsequent hardware reuse, this paper adopts a new method to achieve the result.

Let us first assume that the value of input sequence e is 0, 1, 2, 3, . . ., 19. In other words, the value of the element in the input sequence is equal to its position in the input sequence, i.e., $J_{i,j}$. It is convenient for the later observation. $Q_m = 4$ denotes 16QAM modulation, and the rectangle after the data in the modulation process is shown in Figure 6. When the value of j is 0, the data in the first column are readout. $J_{i,j}$ corresponds to the next data, and it is always 5 more than that of the previous data. For example, the first element in the first row corresponding $J_{i,j}$ is 0, the next element corresponding $J_{i,j}$ is 5, and the next element corresponds to 10. The law of the following columns is the same as that of the first column. Therefore, when the row parameter i is not equal to 0, the value of $J_{i,j}$ is the value of the last read-out data $J_{i,j}$ (can be set as $J_{i-1,j}$) and plus E/Q_m. Then, when i is equal to 0, it can be observed that $J_{i,j}$ is the value of the column parameter j, so the formula of $J_{i,j}$ can be derived as

$$J_{i,j} = \begin{cases} j, & i = 0, \\ J_{i-1,j} + \dfrac{E}{Q_m}, & i \neq 0, \end{cases} \tag{5}$$

where the value of E/Q_m (i.e., the number of columns of the rectangle) can be given in the precalculation stage.

3.1.2. Formula Representation of Polar-Coded Control Channel Interleaver. In the above part, we give a brief overview of the whole interleaving system. Here, we will refine the formula and implement the hardware diagram according to the interleaving process. First of all, we need to make it clear that the data entered the isosceles right triangle interleaver according to the order of rows but readout according to the order of columns. Therefore, we can think that the data are read in line order, and then, a transpose of rows and columns is carried out in the interleaver. Then, we read in line order, which is more convenient for us to derive the formula. Then, we introduce two variables, i and j, as row and column counter, respectively. Here, we make the following provisions for i and j

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19

Figure 6: Input sequence after filling in rectangle.

($i \le P-1, j \le P-1$), where P is the size of the right-angle side of an isosceles right triangle. Meanwhile, when i increases from 0 to $P-1$, j adds 1; j increases from 0 to $P-1$, then i adds 1. When the information data enter the interleaver and are transposed, it is easy to find the order of elements $a_{j,i}$ of row j and column i in the sequence after reading them out by row. We can obtain the formula as

$$a_{j,i} = \begin{cases} i, & j = 0, \\ \left(C_1 + C_j\right) \times (j+1) \times \frac{1}{2} - \left[C_j - (i+1)\right], & j \neq 0. \end{cases} \tag{6}$$

It should be noted that the above formula does not consider the existence of dummy elements, so it only describes the case when the information sequence completely fills the interleaver. In equation (6), C_1 is the length of the first line of the right triangle interleaver, which is P. C_j is defined by

$$C_j = \begin{cases} P, & j = 0, \\ C_{j-1} - 1, & j \neq 0, \end{cases} \tag{7}$$

where C_j is the number of columns in the $(j+1)$th row. Through the above two formulas, we have known the relationship between $a_{j,i}$ and the sequence after interleaving after the internal transposition of interleaver. However, we still do not know the corresponding relationship between $a_{j,i}$ after transposition and the information a_k before entering the interleaver. Through the observation of the internal data of the interleaver after transposition, it is found that when we read the data in the order of columns, it is exactly the order in which the data are stored in the interleaver. Therefore, we can get a corresponding relationship as

$$a_k = \begin{cases} j, & i = 0, \\ \left(C_1 + C_i\right) \times (i+1) \times \frac{1}{2} - \left[C_i - (j+1)\right], & i \neq 0, \end{cases} \tag{8}$$

where C_i is the number of rows in each column. The definition is similar to the above C_j. In the deinterleaving, we should subtract the number of dummy elements, and equation (6) becomes (9) for such a purpose:

$$aj, i = \begin{cases} i, & j = 0, \\ \left(C_1 + Cj\right) \times (j+1) \times \frac{1}{2} - [Cj - (i+1)] - C, & j \neq 0. \end{cases} \tag{9}$$

After deducing the interleaved address without dummy elements, we now address a more realistic situation, namely,

$$P \times \frac{P+1}{2} > M. \tag{10}$$

In this case, the isosceles right triangle interleaver is filled with information elements and many dummy elements. When the dummy elements are taken into account, formula (6) no longer holds. However, we can still use the above numbers to calculate the interleaving address of a certain information unit including several dummy elements, and then, we can calculate the dummy element number C before this information unit and then make a subtraction, and we can obtain the interleaved address of this information unit.

Let us explain in detail how to calculate the number of dummy elements, which needs to be discussed in several cases. Before that, we first define several variables: i_{d} denotes the number of columns of the information unit to be calculated, j_{d} denotes the number of rows of the information unit to be calculated, i_{s} denotes the number of columns of the first dummy element, and $j_{\max}$ denotes the number of columns in the last cell of the first dummy element. We have the following three situations:

(1) $j_{\max} < j_d$. All dummies should be considered at this time. That is,

$$\left[1 + P - i_s\right] \times (j \max + 1) \times \frac{1}{2} - j_s. \tag{11}$$

(2) $j_{\max} = j_d$. First calculate the number of all dummy elements and then subtract 1 to get the total number of dummy elements to be subtracted. That is,

$$\left[1 + P - i_s\right] \times \left(j_{\max} + 1\right) \times \frac{1}{2} - j_s - 1. \tag{12}$$

(3) $j_{\max} > j_d$, under this condition, and it can be further divided into the following two situations. The first case: $i_d = i_s$; at this time, all the dummy elements

included in the (i_s+1)th column and the (j_d-1)th row to be requested are subtracted. That is,

$$\left[P-(i_s+1)+Cj_{d-1}-(i_s+1)\right]\times j_d\times\frac{1}{2}. \tag{13}$$

The second case: $i_d<i_s$, can be divided into the following three scenarios:

(a) $j_d>j_s$. We first calculate the total number of the i_s-th column and the j_d−1th row to be calculated and then subtract the number of information units in this range. That is,

$$\left[P-i_s+C_{j_d-1}-i_s\right]\times j_d\times\frac{1}{2}-j_s. \tag{14}$$

(b) $j_d<j_s$. At this time, the number of dummy elements in the i_s-st column and the j_d-th row are calculated. That is,

$$\left[P-(i_s+1)+C_{j_d-1}-(i_s+1)\right]\times j_d\times\frac{1}{2}. \tag{15}$$

(c) $j_d=j_s$. This case is the same as scenario (b).

After we analyzed all the required formulas, we start to design the hardware implement scheme. The first is the implementation of C_j, and here, we can use a loop with a judgment to achieve it. According to the formula, we can design a hardware structure with the subtraction gate as the main structure. On this basis, we add a judgment on the position of the output. When $j=0$, the output is 4. If this condition is not satisfied, we set a delay through the register and then make a subtraction with 1 in turn. On this basis, we implement the hardware structure step by step according to the formula. We add a judgment to the final output to meet the requirements of the formula. For the implementation of dummy computing hardware, the formulas can be divided into three categories. Among them, case 1 and case 2 belong to one category, and the hardware implementation of equation (11) can be reused; then, a logical judgment is added, and if the second case is satisfied, one is subtracted. The case one of (3) and (b) can be reused, while (a) cannot be reused because they do not have the same structure.

Note: when the number of dummy rows to be considered is only 1, if $j_s=0$ at this time, we only need to consider the number of $P-i_s$. When j_s is not equal to 0, we only need to consider the number of $P-i_s-1$. If the information unit to be calculated is on the first line, there is no need to subtract the number of dummy elements.

3.2. Verification of Formula for Interleaved Addresses

3.2.1. Verification of Interleaved Address Formula for Data Channel. We use Matlab to simulate the LDPC interleaving formula. First assume that the input sequence is [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19] and the modulation order is 4. The corresponding figure of this sequence is shown in Figure 7.

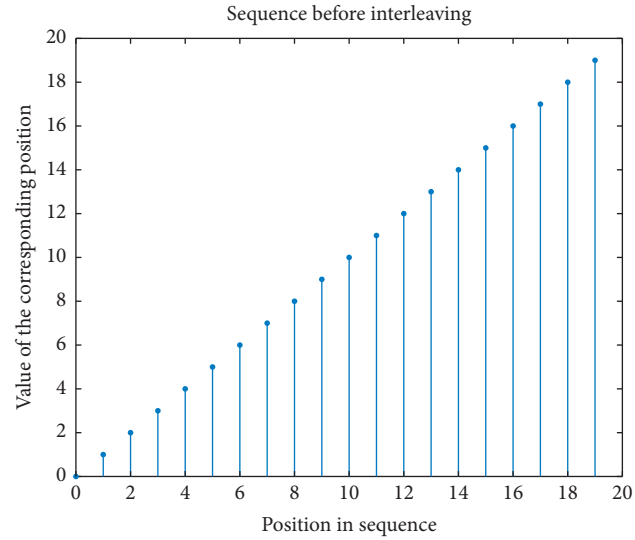


Figure 7: Sequence before interleaving.

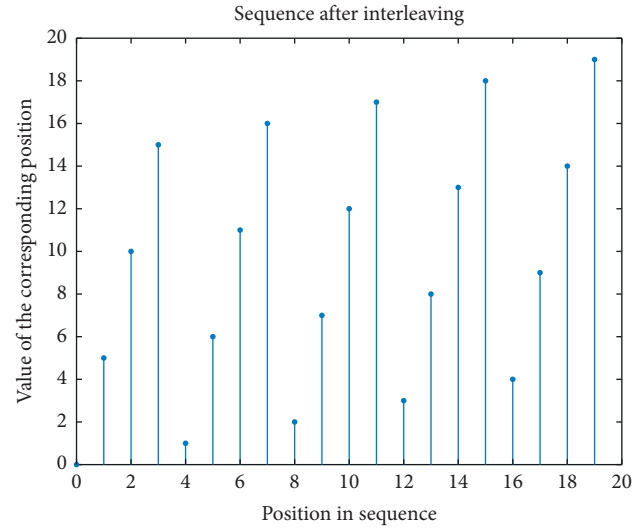


Figure 8: Interleaved sequence.

Using equation (5), $J_{i,j}$ is obtained. The corresponding position element is taken out from the input sequence according to the value of $J_{i,j}$, which is the value in the output sequence, as shown in Figure 8.

After verification, the interleaved sequence can be deinterleaved back to the original sequence. The formula is the same as the interleaving formula. We only need to exchange the ranges of i and j with each other and change the E/Q_m in the formula to Q_m. That is,

$$J_{i,j}=\begin{cases} j, & i=0,\\ J_{i-1,j}+Q_m, & i\neq 0. \end{cases} \tag{16}$$

Then, the interleaved sequence is deinterleaved in the same way as the interleaving process, and the deinterleaved sequence (i.e., the original input sequence) is obtained. The result of the deinterleaved sequence is shown in Figure 9.

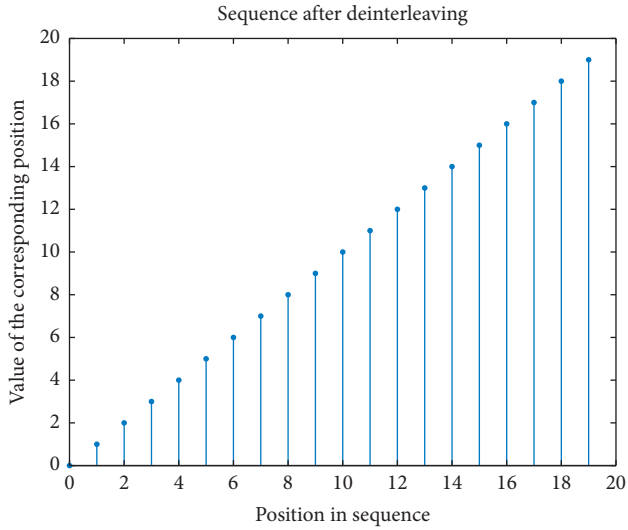


Figure 9: Deinterleaved sequence.

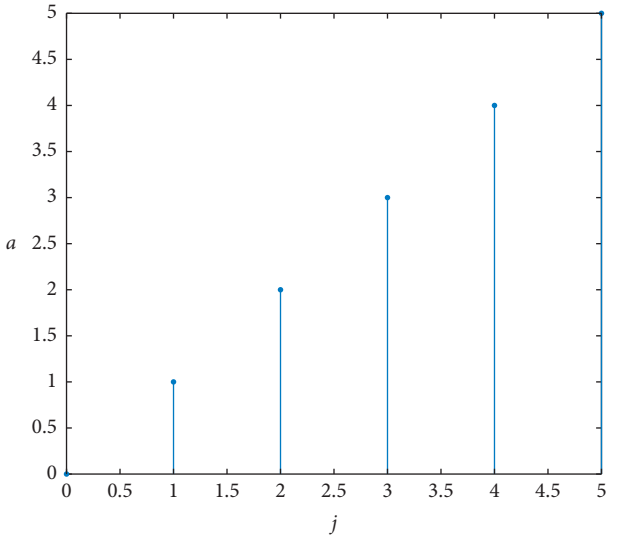


Figure 10: Verification 1 for formula (8).

From Figure 9, the deinterleaving method successfully restores the interleaved sequence to the original sequence. Hence, the interleaving formula and deinterleaving formula work correctly.

3.2.2. Verification of Interleaved Address Formula for Control Channel. In order to verify the formula, we define an isosceles right-angled triangle with $P = 6$ and load the data a_1 to a_{21} into the triangle interleaver in the order of the rows. The interleaving process is equivalent to the data in this triangle matrix. After transposing, we take them out in rows, so we make $a_{i,\ j}$ into $a_{j,\ i}$. First, we verify the correctness of formula (8). For the number of the first column in the interleaver, we can get it as shown in Figure 10.

For the second column number in the interleaver, the corresponding C_1 is 5 at this time, and then, the function formula we determined becomes

$$a_k = (6 + 5) \times 2 \times \frac{1}{2} - [5 - (j + 1)]. \tag{17}$$

Among them, the corresponding values of j are 0, 1, 2, 3, and 4, which can be obtained through Matlab. The result is shown in Figure 11.

For the third column number in the interleaver, the corresponding C_2 is 4 at this time, and then, the function formula becomes

$$a_k = (6 + 4) \times 3 \times \frac{1}{2} - [4 - (j + 1)], \tag{18}$$

where the corresponding values of j are 0, 1, 2, and 3. The result is shown in Figure 12.

After comparison, we find out that this corresponds to the actual serial number of the information after it is loaded into the interleaver and after transpose. Hence, the formula is theoretically feasible.

Next, we verify the formula of the interleaved address information given in equation (6). For $j = 0$, the first row of elements according to (6) is obtained and shown in Figure 13.

For $j = 1$, the second row of elements is obtained. According to equation (6), we get

$$a_{j,i} = (6 + 5) \times 2 \times \frac{1}{2} - [5 - (i + 1)]. \tag{19}$$

The simulation result is shown in Figure 14.

For $j = 2$, the third row of elements is obtained. According to equation (6), we get

$$a_{j,i} = (6 + 4) \times 3 \times \frac{1}{2} - [4 - (i + 1)]. \tag{20}$$

And the simulation result is shown in Figure 15.

After comparison, we find that this is consistent with the sequence corresponding to the actual information loaded into the interleaver and readout row by row after a transpose. From these, we conclude that the formula is theoretically feasible. In the following, we verify the corresponding C value for different situations, that is, the number of dummy elements to be subtracted.

For the first case, that is, $j_{\max} < j_d$ to be calculated, the number of dummy elements must be considered in calculating the address of the information at this time. This case corresponds to equation (11). Suppose we now find a_6, and we let the head of i_s in equation (11) be 1 and 2 to verify the correctness of the expression.

When $i_s = 1$, the corresponding $j_{\max} = 4$, and the value of j_s is in the range [0, 1, 2, 3, 4], that is, from a_6, a_7, a_8 to a_{11}, respectively, as the first dummy element. The parameters i_s and $j_{\max}$ are substituted into equation (11), and the result can be obtained and shown in Figure 16.

When $i_s = 2$, corresponding to $j_{\max} = 3$, the range of j_s is [0, 1, 2, and 3], that is, from a_{12}, a_{13}, a_{14} to a_{15}, respectively, as the first dummy element. The parameters i_s and $j_{\max}$ are substituted into equation (11), and the result can be obtained and shown in Figure 17.

After comparison, we verified the correctness of the formula. For the above second case, $j_{\max} = j_d$ is to be

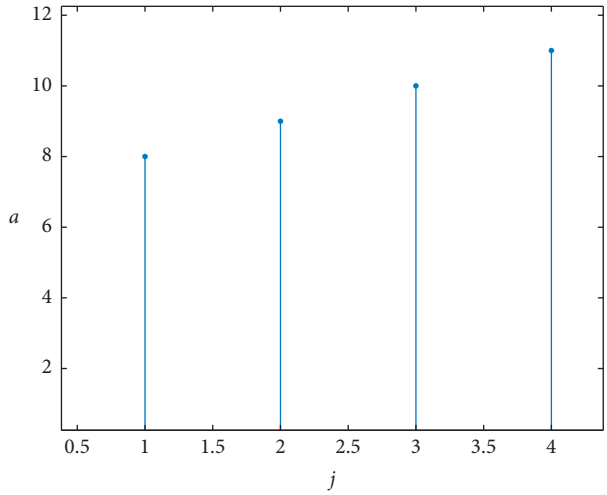


Figure 11: Verification 2 for formula (8).

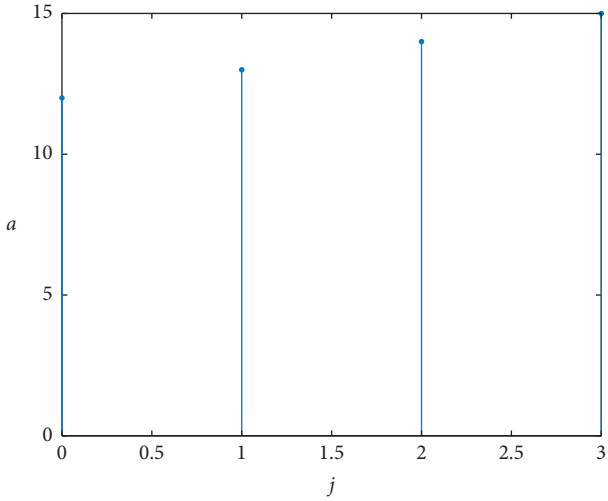


Figure 12: Verification 3 for formula (8).

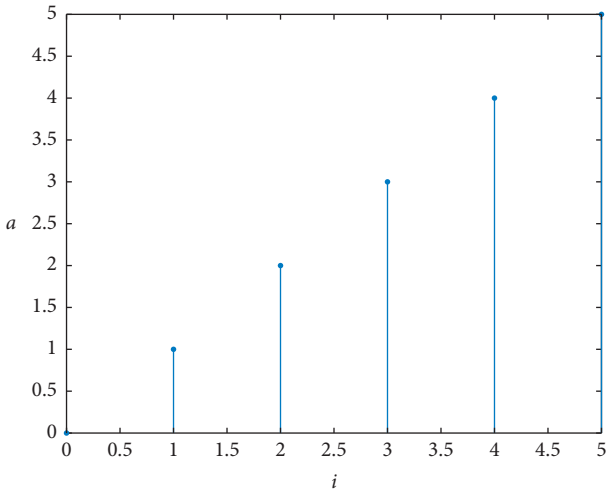


Figure 13: Verification 1 for formula (6).

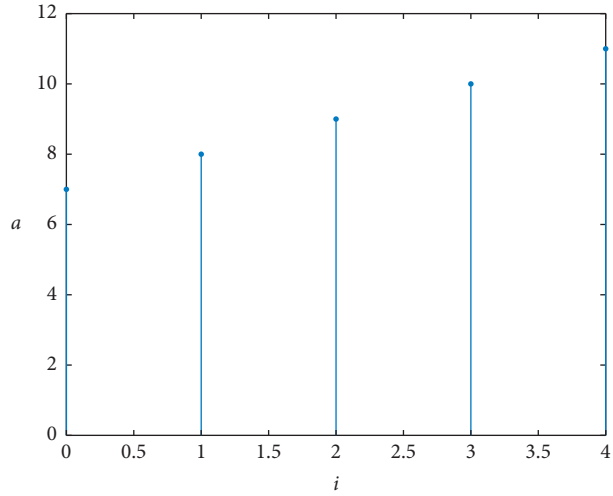


Figure 14: Verification 2 for formula (6).

calculated; that is, the information unit to be obtained is on the same line as the dummy with the largest number of rows. At this time, the number we need to consider is the total number of dummies minus 1 (only considering the number of dummy elements before this information unit). In view of this situation, we can subtract 1 if equation (11) is correct.

For the case 3, that is, $j_{\max} > j_d$ to be decided, first, we discuss $i_d = i_s$; that is, the information unit to be decided and the first dummy element are in the same column. At this time, we need to consider the number of all units contained in the $(i+1)th$ column and the $(j-1)th$ row based on the information unit. This case corresponds to equation (13). In equation (13), we choose $i_s = 1$ and $i_s = 2$ for verification. When the i_s is 1, j_d can be 2 and 3. When i_s is 2, the j_d can be 2 and 3 as well.

When $j_d = 2$, according to equation (13), for different i_s, the simulation result can be obtained in Figure 18.

When $j_d = 3$, according to equation (13), for different i_s, the simulation result is shown in Figure 19.

After verification, it is the same as the theoretical value. When $i_d < i_s$, first discuss $j_d > j_s$; that is, the number of rows of the information unit to be decided is greater than the first dummy element. At this time, the formula for calculating C differs from the above formula only in that we are considering the i_s column starts. Meanwhile, we can subtract the number of information units in the i_s row. When the first dummy element starts from 8 and 12, respectively, we use (14) to solve the address after a_3 interleaving. Because the corresponding j_s and t_s are different, the verification results are more general.

For the case where the first dummy starts at 8, the formula is

$$C = (6 - 1 + 5 - 1) \times 2 \times \frac{1}{2} - 1. \tag{21}$$

The calculated result is 8, which is the correct result.

For the case where the first dummy starts at 12, the formula is

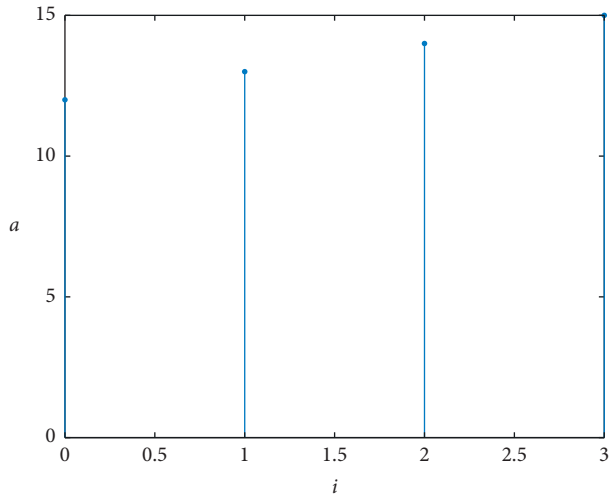


Figure 15: Verification 3 for formula (6).

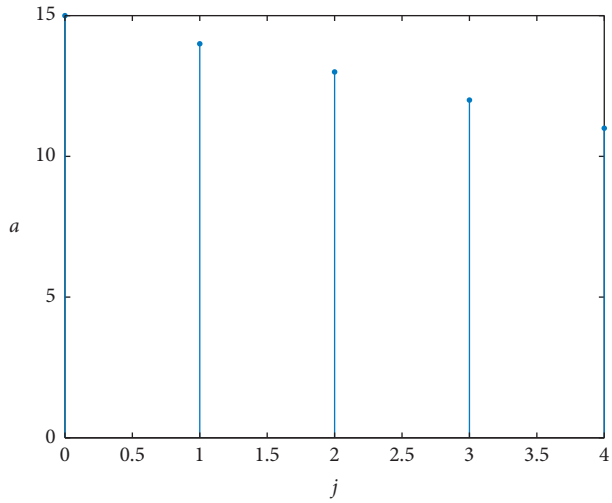


Figure 16: Verification 1 for formula (11).

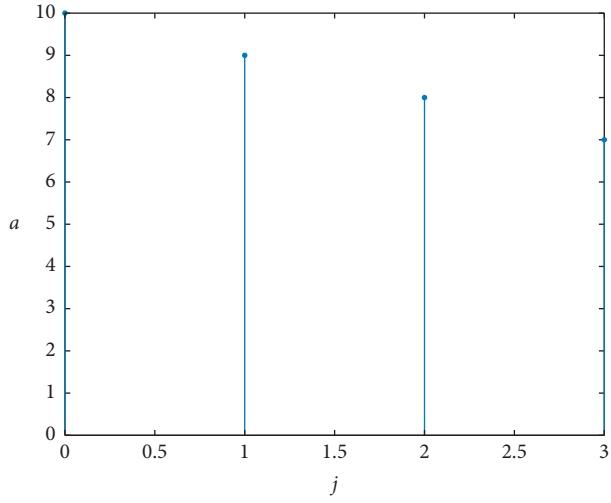


Figure 17: Verification 2 for formula (11).

$$C = (6 - 2 + 5 - 2) \times 2 \times \frac{1}{2} - 0. \tag{22}$$

The calculated result is 7, which is the correct result.

When $j_d \le j_s$, the number of dummy elements we need to calculate at this time is all the numbers in $i_s + 1$, $j_d - 1$. The calculation formula is equation (15).

3.3. Hardware Design and Reuse of Two Coding Interleavers

3.3.1. Interleaver Hardware Design for LDPC-Coded Data Channel. From the formula of $J_{i,j}$, it can be concluded that the hardware required for its implementation is an adder, a selector, and an address register, which can realize the interleaver of the data channel. It is shown in Figure 20.

3.3.2. Interleaver Hardware Design for Polar-Coded Control Channel. The interleaver hardware design for polar-coded control channel is shown in Figure 21. In the first part of the figure, we can get C_i and then pass through a few adders and subtractors. Before entering the second part, the output of the subtraction gate is

$$(C_i + C_1) \times (i + 1) \times \frac{1}{2} - [C_i - (i + 1)]. \tag{23}$$

When judged by a logic gate, if $j = 0$ (that is, the one in the first row after replacement), the output is i. If $j = 0$ is not satisfied, the output is equation (23).

3.3.3. Hardware Multiplexing of Two Interleavers. By observing and comparing the hardware implementation diagrams of two interleavers, we can find that the hardware structure of LDPC-coded data channel interleaver has also appeared in the polar-coded control channel interleaver. Thus, the hardware structure of the data channel interleaver can be set to a new module M, and its structure diagram is shown in Figure 22. It has a total of three input terminals (a, b, c) and one output terminal y. The input and output parameters can be determined according to the selection of the interleaving scheme. If it is selected for the data channel interleaving, the input parameters a, b, and c are E/Q_m, j, and i; the output is the interleaved address $J_{i,j}$. If it is selected for the control channel interleaving, the input parameters are 1, 4, and j, respectively. The output is C_j. Therefore, the final design of the multiplexing structure can be obtained as shown in Figure 23.

3.3.4. Flow Charts of Precalculation Stage and Execution Stage. The flow charts of precalculation stages and execution stages are shown in Figures 24 and 25, respectively.

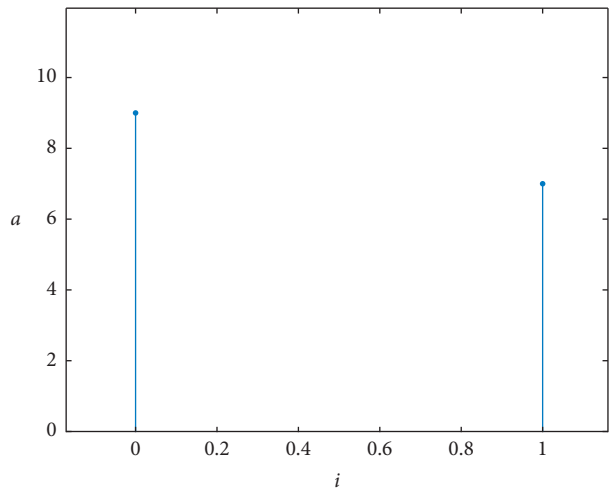


Figure 18: Verification 1 for formula (13).

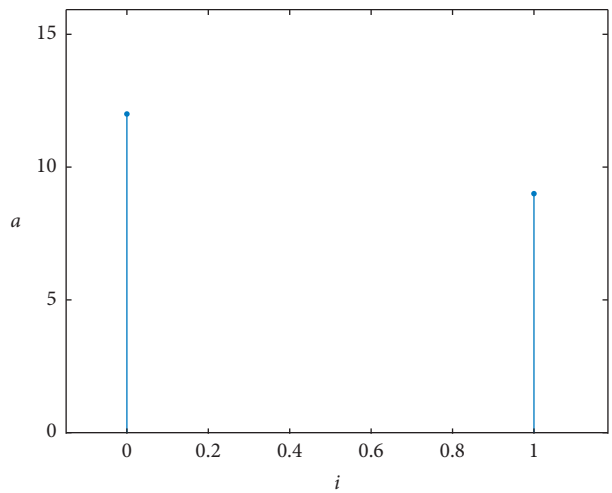


Figure 19: Verification 2 for formula (13).

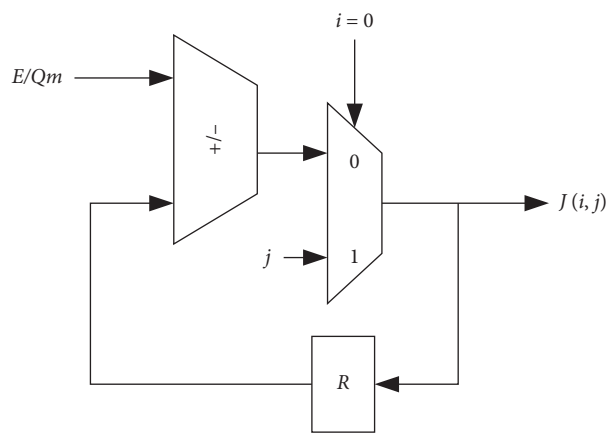


Figure 20: Hardware diagram of data channel interleaver.

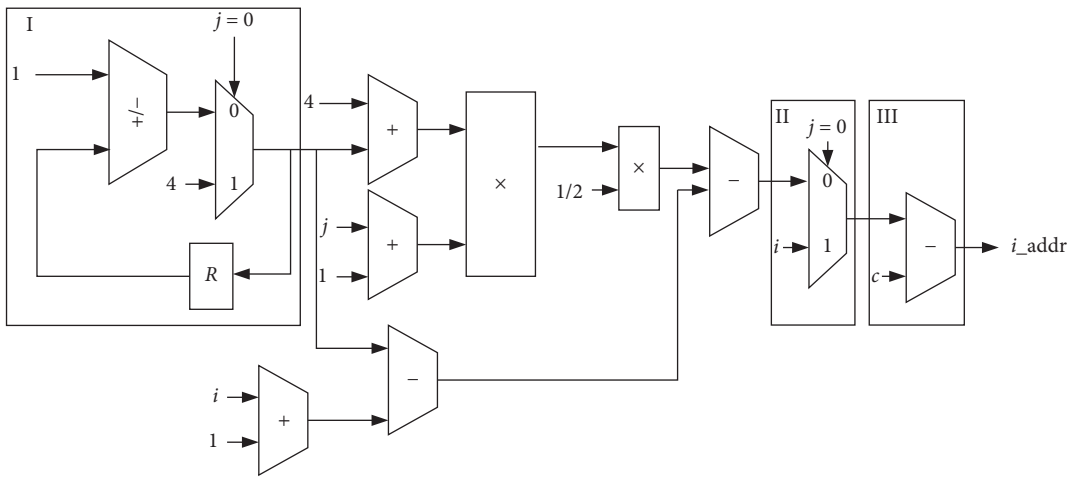


Figure 21: Hardware diagram of control channel interleaver.

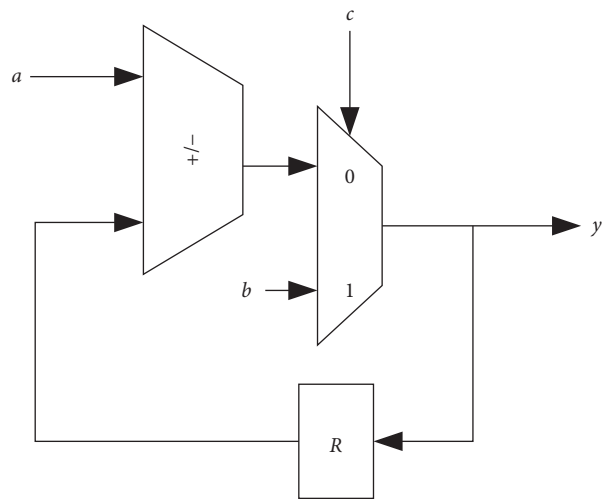


Figure 22: Multiplexing module *M*.

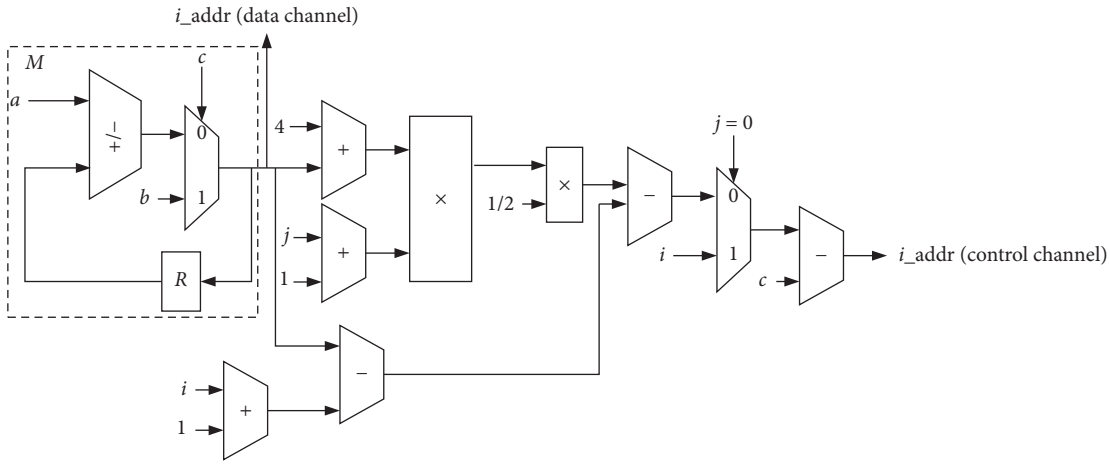


Figure 23: Hardware sharing structure after reuse.

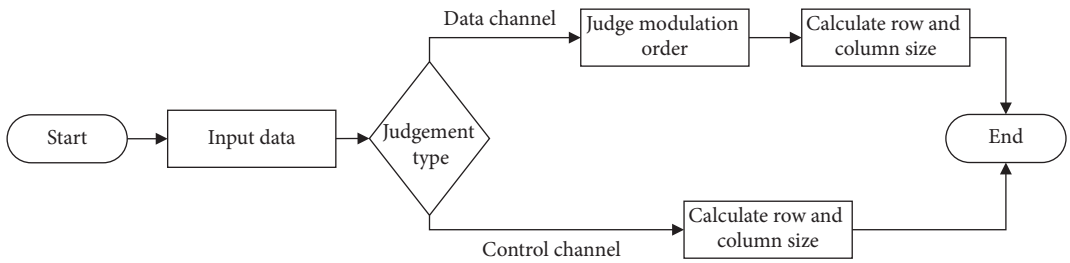


Figure 24: Flow chart of precalculation stage.

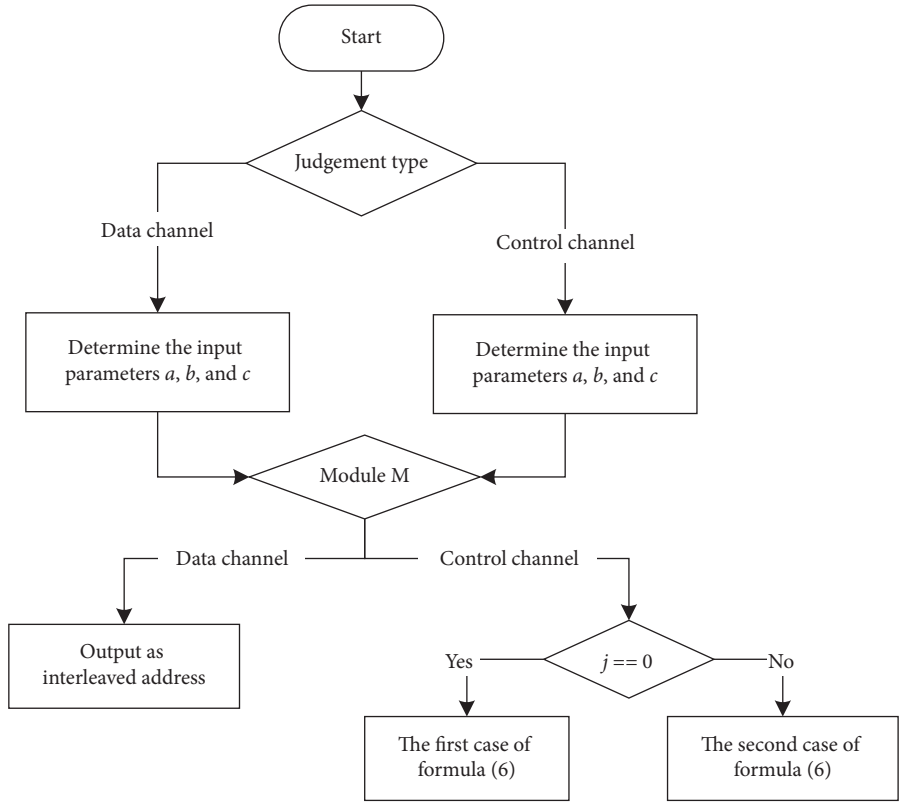


Figure 25: Flow chart of execution stage.

4. Conclusions and Future Work

This paper presents an interleaver multiplexing scheme for the LDPC and polar encoding channel which are specified in 5G NR standards. First, we analyze the two interleaving methods and then refine and improve the formulas according to the interleaving process to achieve the hardware reuse. Then, according to the formulas, the hardware realization of interleaving address is derived. Finally, the hardware implementation of the two-channel interleavers is reused as much as possible to achieve the purpose of reducing the hardware cost. However, there are still some issues to be improved in our research work. For example, the formula for generating interleaving address extracted is complicated; especially the formulas for refining interleaving process of control channel need to be further simplified. In the future work, we will also consider the parallelization processing under a variety of channel encoding standards in combination with rate matching.

Data Availability

The code and data of ".m" and ".mat" format files used to support the findings of this study have been deposited in the GitHub repository (https://github.com/huzhuhua/Data-and-Code-for-Security-and-Communication-Networks).

Conflicts of Interest

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

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61963012, 61961014, and 61661018), Natural Science Foundation of Hainan Province, China (no. 619QN195), and Key R&D Project of Hainan Province, China (no. ZDYF2018015).

References

[1] R. Asghar, *Flexible Interleaving Sub–systems for FEC in Baseband Processors*, Linköping University Electronic Press, Linköping, Sweden, 2010.

[2] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 461–474, 2017.

[3] Z. Hu, Y. Bai, Y. Zhao, and M. Xie, "Adaptive and blind wideband spectrum sensing scheme using singular value decomposition," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 3279452, 14 pages, 2017.

[4] Z. Hu, Y. Bai, M. Huang, M. Xie, and Y. Zhao, "A self-adaptive progressive support selection scheme for collaborative wideband spectrum sensing," *Sensors*, vol. 18, no. 9, p. 3011, 2018.

[5] H. Wu, W. J. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.

[6] X. Liu, M. Jia, X. Zhang, and W. Lu, "A novel multichannel Internet of things based on dynamic spectrum sharing in 5G communication," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 5962–5970, 2019.

[7] X. Liu and X. Zhang, "NOMA-based resource allocation for cluster-based cognitive industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5379–5388, 2020.

[8] R. Asghar and D. Liu, "Multimode flex-interleaver core for baseband processor platform," *Journal of Computer Networks and Communications*, vol. 2010, p. 2010.

[9] Y. Sun, Y. Zhu, M. Goel, and J. R. Cavallaro, "Configurable and scalable high throughput turbo decoder architecture for multiple 4G wireless standards," in *Proceeding of the IEEE International Conference on Application-specific Systems, Architectures and Processors*, pp. 209–214, IEEE, Leuven, Belgium, July 2008.

[10] R. Asghar, D. Wu, J. Eilert, and D. Liu, "Memory conflict analysis and implementation of a re-configurable interleaver architecture supporting unified parallel turbo decoding," *Journal of Signal Processing Systems*, vol. 60, no. 1, pp. 15–29, 2010.

[11] Z. Zhang, B. Wu, Y. Zhou, and X. Zhang, "Low-complexity hardware interleaver/deinterleaver for IEEE 802.11a/g/n WLAN," *VLSI Design*, vol. 2012, Article ID 948957, 7 pages, 2012.

[12] P. Benoit, L. Torres, G. Sassatelli et al., "Dynamic hardware multiplexing: improving adaptability with a run time reconfiguration manager," in *Proceeding of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, March 2006.

[13] R. Asghar and D. Liu, "Low complexity hardware interleaver for MIMO-OFDM based wireless LAN," in *Proceeding of the IEEE International Symposium on Circuits and Systems*, pp. 1747–1750, IEEE, Taipei, China, May 2009.

[14] 3GPP TS 38.212. NR, "Multiplexing and Channel Coding," 2017.

[15] E. Arikan, "Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[16] W. E. Ryan, "An introduction to LDPC codes," *CRC Handbook for Coding and Signal Processing for Recording Systems*, pp. 1–23, CRC Press, Boca Raton, FL, USA, 2004.

[17] R1-1713474. "Design and Evaluation of Interleaver for Polar Codes," Qualcomm Inc., 3GPP TSG RANWG1 #90 Meeting, Prague, Czechia, 2017.

[18] H. Gamage, N. Rajatheva, and M. Latva-Aho, "Channel coding for enhanced mobile broadband communication in 5G systems," in *Proceeding of the European Conference on Networks and Communications (EuCNC)*, IEEE, Oulu, Finland, 2017.

[19] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, 1997.

[20] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.

[21] J. Xu and J. Xu, "Structured LDPC applied in IMT-advanced system," in *Proceeding of the 4th IEEE International Conference on Wireless Communication*, IEEE, Dalian, China, 2008.

[22] R. Asghar and D. Liu, "Multimode flex-interleaver core for baseband processor platform," *Journal of Computer Networks and Communications*, vol. 2010, Article ID 793807, 16 pages, 2010.

[23] R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution," *IEEE Communications Letters*, vol. 13, no. 7, pp. 519–521, 2009.

[24] 3GPP R1-167209, "Polar Code Design and Rate Matching," Huawei and HiSilicon, 3GPP TSG RAN WG1 #86 Meeting, Gothenburg, Sweden, 2016.

[25] 3GPP, R1-1713705, "Polar rate-matching design and performance," MediaTek, WG1#90, 2017.

[26] 3GPP, R1-1708649, "Interleaver design for polar codes," qualcomm, RAN#89, 2017.

[27] 3GPP, "Draft_Minutes_report_RAN#91_v020," 2017, http://www.3gpp.org/ftp/tsg_ran/WG1_RL1/a) TSGR1_91/Report/Draft_Minutes_report_RAN1%2391_v020.zip.

[28] E. Tell and D. Liu, "A hardware architecture for a multi-mode block interleaver," in *Proceeding of the International Conference on Circuits and Systems for Communications (ICCSC)*, Moscow, Russia, 2004.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8876317, 12 pages
https://doi.org/10.1155/2020/8876317

Research Article

A Study on the Optimization of Blockchain Hashing Algorithm Based on PRCA

Jinhua Fu,[1,2] **Sihai Qiao**,[2] **Yongzhong Huang**,[1,3] **Xueming Si**,[1,4] **Bin Li**,[1,5] **and Chao Yuan**[1]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China*
[2]*School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450001, China*
[3]*School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China*
[4]*School of Computer Science, Fudan University, Shanghai 201203, China*
[5]*Zhengzhou University, Zhengzhou 450001, China*

Correspondence should be addressed to Jinhua Fu; jinhua@zzuli.edu.cn

Received 12 March 2020; Revised 14 April 2020; Accepted 23 May 2020; Published 14 September 2020

Academic Editor: Yuan Yuan

Blockchain is widely used in encrypted currency, Internet of Things (IoT), supply chain finance, data sharing, and other fields. However, there are security problems in blockchains to varying degrees. As an important component of blockchain, hash function has relatively low computational efficiency. Therefore, this paper proposes a new scheme to optimize the blockchain hashing algorithm based on PRCA (Proactive Reconfigurable Computing Architecture). In order to improve the calculation performance of hashing function, the paper realizes the pipeline hashing algorithm and optimizes the efficiency of communication facilities and network data transmission by combining blockchains with mimic computers. Meanwhile, to ensure the security of data information, this paper chooses lightweight hashing algorithm to do multiple hashing and transforms the hash algorithm structure as well. The experimental results show that the scheme given in the paper not only improves the security of blockchains but also improves the efficiency of data processing.

1. Introduction

Blockchain is a kind of distributed general ledger technology, originated from the literature [1]. Initially, it was mainly used in the field of cryptocurrency, the most representative of which were Bitcoin and Litecoin [2], Monroe [3], and Zcash [4]. Amid its rapid development, blockchain technology can effectively guarantee the authenticity, security, and reliability of data. It also has been widely used in medical data [5], personal data protection [6], and data allocation scheme [7]. As the basic unit of blockchain, block consists of partition header including original data and block body including transaction data. Among them, block data are used to connect the previous block and index the data from the hash value of range block. Each blockchain transaction is conducted by using hash function interaction. It guarantees the security of blockchain.

However, with the continuous development of blockchain, its security issues become increasingly prominent. The lightweight hash function SHA1 in the blockchain is no longer regarded as an attacker that can withstand sufficient funds and computing resources. SHA256 can replace SHA1 for information exchange with good anticollision ability, while it cannot be changed at will. To avoid chain breakage, it is necessary to modify the hash values of all blocks behind the block at the same time. As a result, a large computational complexity is needed and the security of the blockchain is not guaranteed.

In the process of executing operations, the PRCA (Proactive Reconfigurable Computing Architecture) generates the optimal computation structure set by self-perception and dynamic selection. All the software and hardware variants are dynamically variable. Therefore, in the process of application processing, they can select optimal solutions according to the independent variables in the program to get the variable optimal solution sets with equivalent function and different computing efficiency [8]. Combining with blockchain, it can improve the performance

of the algorithm, improve the transmission efficiency, and enhance the security of hash algorithm.

This paper proposes an optimization scheme of blockchain hashing algorithm based on PRCA. Aiming at the blockchain hash algorithm structure, a reconfigurable hash algorithm with high performance is implemented in a full pipeline way. At the same time, 10,000 Mbp communication is realized by mimic computer to reduce data transmission delay, and data is read from memory by DMA, which improves transmission efficiency. In each transaction, the hash algorithm is negotiated and the mimic computer is reconstructed, which aims to transform the hash algorithm structure through using lightweight hash algorithm for many times. This scheme not only improves the efficiency of processing data for blockchain but also increases its security.

2. Proactive Reconfigurable Computing Architecture

2.1. Definition of Proactive Reconfigurable Computation. PRCA is an operation mechanism based on multidimensional reconstructed functional structure and dynamic multibody. When proactive reconfigurable computation is processing data, execution structures, such as computing, storage, and interconnection, are changing dynamically with the efficiency of transaction processing, instead of improving the algorithm to improve the operation performance without changing the basic hardware. There are many functional equivalents in PRCA, but they are accomplished by combining different hardware structures with this algorithm. The purpose is to achieve the high performance of computing, that is, how to automatically perceive variables to generate the optimal computing set and autonomously reconstruct the computing in the processing algorithm [9].

PRCA has variable infrastructure and algorithm, which makes it possible to obtain optimal solutions to different problems. It pursues different services and comprehensive high performance under different loads or other conditions, builds the most appropriate processing components, and forms the most appropriate architecture. Proactive reconfigurable computation combines the advantages of general computing and special computing to achieve the goal of solving problems efficiently. In terms of the general computing structure, it is characterized by its determined structures and variable algorithm and may calculate any computable problems with high efficiency. Its principle is shown in Figure 1.

2.2. Proactive Reconfigurable Computer. Proactive reconfigurable computer is a new type of computer developed according to the principle of mimetic computing to achieve the high performance of computing. The computational structure can be regarded as a high-order function. In the analysis of the calculation, the computational structure will generate the most efficient set of settlement structures by selecting the perceptual independent variables. The essence of proactive reconfigurable computer is the functionalization of computational structure. Its high performance and efficiency are very suitable for the processing and analysis of big data nowadays. Compared with the traditional computer, the energy efficiency of proactive reconfigurable computer has been improved more than 10 times. The structure of the principle prototype of the proactive reconfigurable computer is shown in Figure 2.

The purpose of proactive reconfigurable computer is to deal with intensive computing. It consists of an ATOM general microprocessor, four high-order reconfigurable large-scale reconfigurable FPGAs, and DDR3 memory, which connects LVDS bus FULL-MESH through floor GTX, and is controlled by the control unit BMC and synchronized by clock synchronization unit. The prototype supports multiple interfaces and storage media and reconstructs FPGA processing core, I/O interface, and on-chip interconnection network according to the application requirements, so as to achieve the purpose of high-efficiency computing [10].

Proactive reconfigurable computers use dynamic randomicity to build an asymmetric defense system, which expands the attack surface to weaken intrinsic attacks of feature sniffing and state transition [11]. Based on such a characteristic, 10,000 Mbp communication is realized by using FPGA to reduce data transmission delay, build a simulated hash structure, and improve the speed of hash value calculation of blockchain data. A Merkle tree is formed to match the algorithm, which makes it difficult for attackers to distinguish the complexity of the target and improves the security performance of the system [12]. The protection function of computer hardware is used to expand the area of attack, increase the difficulty of blockchain attack, and improve the antiattack ability.

3. Optimization of Blockchain Hash Algorithms Based on PRCA

3.1. System Framework and Block Structure. The proactive reconfigurable computer is configured as a node in the blockchain network. Users and proactive reconfigurable computers establish a connection. The proactive reconfigurable computer catches the data in the DDR memory and realizes the direct connection high-speed transmission from network to the memory data by the asynchronous FIFO, reducing the intermediate transmission level. In blockchain, a high-performance hash algorithm is implemented by means of pipelines and the key segment calculation data hash is extracted from memory [13]. After calculating the hash value, the result is encapsulated and transferred to the storage server to complete the storage of the blockchain. The specific system framework is shown in Figure 3.

The block stores all the information about transactions, including the generation time of transaction, the record index number of transaction, the hash value of transactions, bitcoin's expenditure address and its amount of expenditures, and other types of transaction. A Merkle value will be generated in the transaction. The hash node value in the transaction determines that each address cannot be repeatedly traded and forged. To further improve the security of transactions, a proactive reconfigurable hash is added to the blockchain, which is composed of various types and

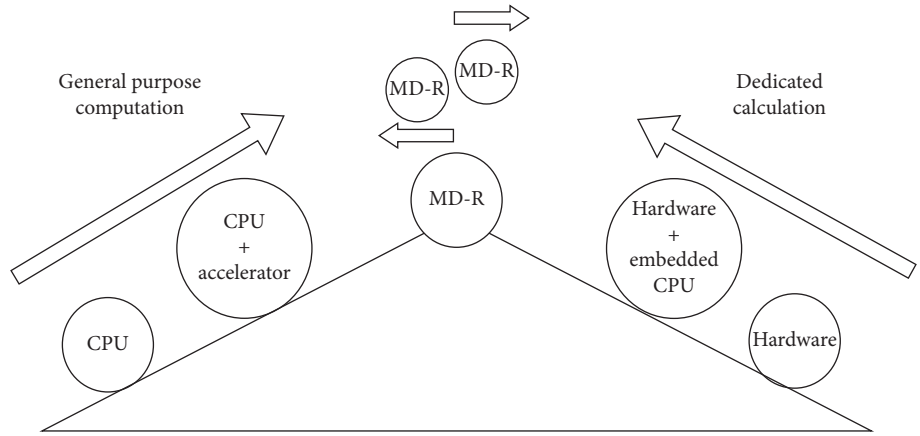


Figure 1: The basic concept of PRCA.

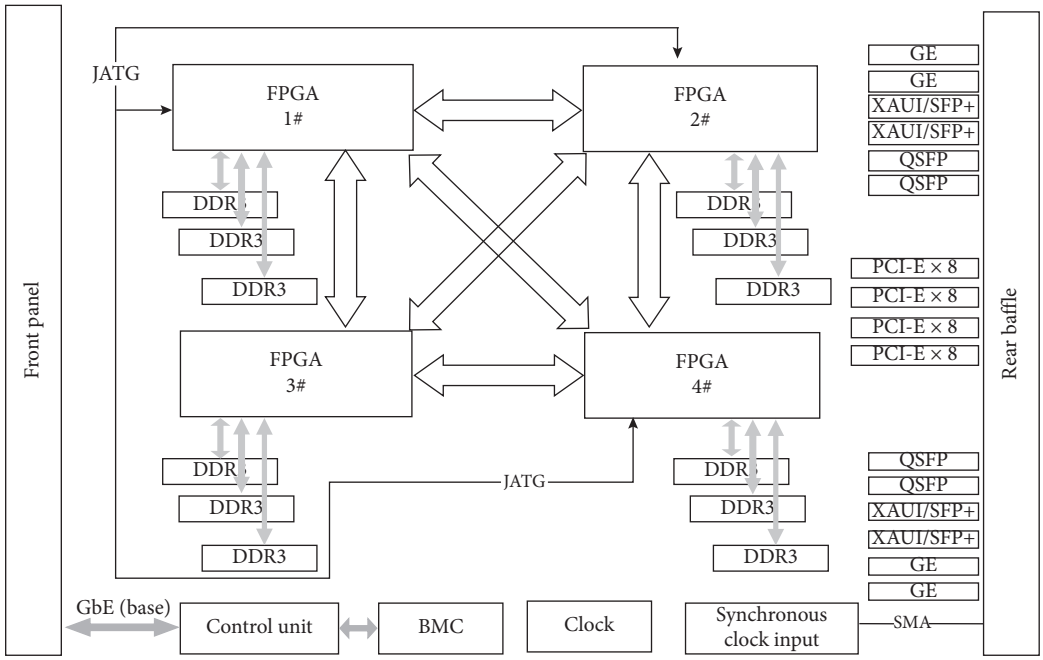


Figure 2: Principle prototype structure of the proactive reconfigurable computer.

structures of hash algorithms and can be used separately or in series. The concrete structure model is shown in Figure 4.

Unit nodes in blockchains monitor network traffic to calculate transaction volume [14]. Before the transaction is generated, the hash algorithm selection step will be added, and then the appropriate hash function will be selected from the hash list. The unit node uses the selected hash function to compete to find the hash value. Once the hash value is found, the block will be propagated to another node in the blockchain for verification.

In the interaction, the sensor layer on the spot collects data. The sensor transmits data to unit nodes and requests the transaction to store the data. If unit nodes successfully complete the transaction mining, the blockchain network will update the block. After that, the blockchain network returns the field layer data to the control layer. Then block mining will be started. After the block mining is finished, the blockchain network receives the node of transaction and broadcasts the block and validation request to other nodes. Other nodes using hash algorithm confirmed from the block header for verification. After the successful verification, they will update the block and store nodes and blocks. If the contents of transactions are transferring data or commands, the requested node will transfer the data or command to the other layers. The specific block mining and updating are shown in Figure 5.

At the same time, the random number generator randomly chooses the new hash algorithm at intervals, and the two sides negotiate again and update for new hash algorithm to improve security.

3.2. Hash Algorithm Optimization. Hash function is an important part of many cryptographic algorithms. An important component of blockchain technology is to apply

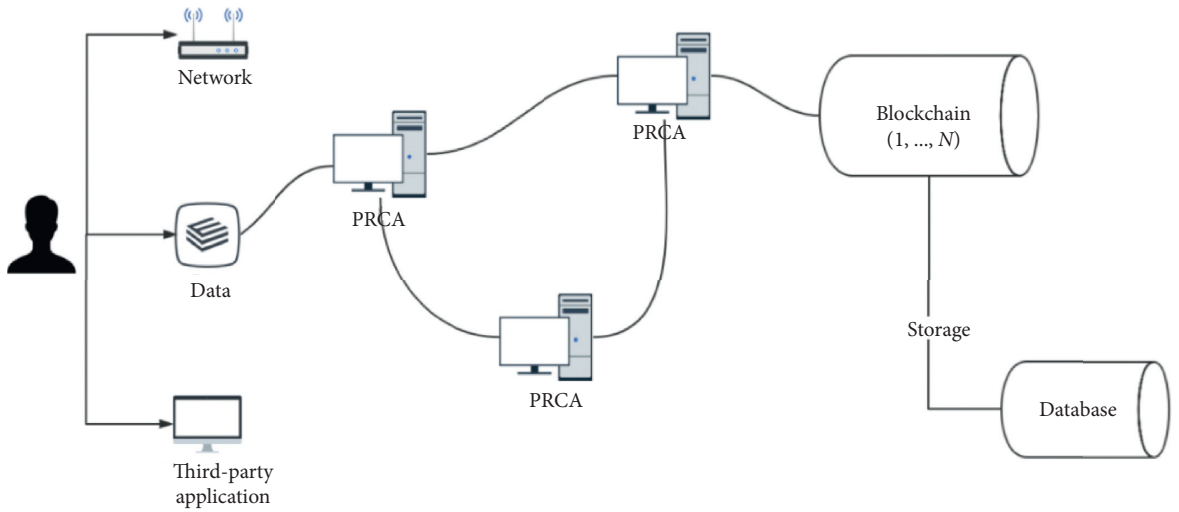


Figure 3: Blockchain system architecture based on PRCA.

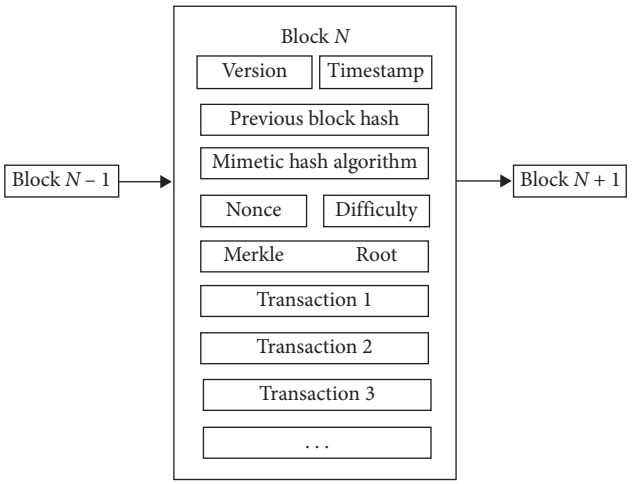


Figure 4: Proactive reconfigurable hash structure in blockchain.

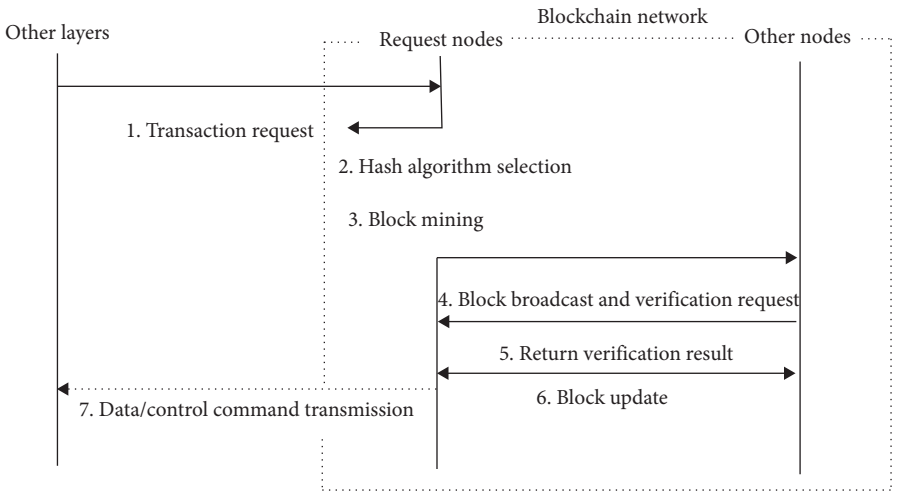


Figure 5: Mining and updating of block.

hash function for many operations. Hashing is a method of applying hash function to data that computes a relatively unique output for almost any size of input. It allows individuals to independently obtain input data and hash data and produce the same results, proving that the data has not changed. Take SHA256 as an example to illustrate the optimization and implementation of hash algorithm on proactive reconfigurable computers.

The throughput of the algorithm solves the computational performance of the algorithm. The specific implementation formula is as follows:

$$T = \frac{B \times f_{\max} \times N}{d}. \tag{1}$$

In equation (1), T is the throughput, B denotes the data block size, f is the maximum clock frequency, N is the pipeline series, and d denotes the calculation delay. The number of pipeline series is proportional to frequency and throughput. In order to improve the throughput of the algorithm, we can use prediction and CSA strategies to reduce the delay of critical paths and use full-pipeline SHA1 and SHA256 algorithms.

The following is an introduction to the optimization of SHA256, which can be extended to SHA1.

3.2.1. SHA256. For messages with a length no more than 2^{64} bits, the hash algorithm SHA256 will produce a hash value with a length of 256 bits, which is called a message digest. The digest is a 32-byte array that can be represented by a hexadecimal string of length 64. The processing of the SHA256 algorithm is divided into five steps:

(i) Add great many 0 bits to the input data until 448 bits. Then add 64-bit length to the input data until 512 bits.

(ii) Divide the spliced 512-bit data into 16 groups: M_0–M_{15}.

(iii) Initialize the vectors K_0–K_{63} and h_0–h_7, and let the initial values of A, B, C, D, E, F, G, and H be h_0–h_7.

(iv) Set the variable t to loop from 0 to 63 and then update as follows: $B_{t+1} = A_t, C_{t+1} = B_t, D_{t+1} = C_t, F_{t+1} = E_t, G_{t+1} = F_t, H_{t+1} = G_t$,

$$\begin{aligned} A_{t+1} &= H_t + \sum_1 (E_t) + \mathrm{Ch}(E_t, F_t, G_t) + K_t + W_t \\ &\quad + \sum_0 (A_t) + \mathrm{Maj}(A_t, B_t, C_t), \\ E_{t+1} &= H_t + \sum_1 (E_t) + \mathrm{Ch}(E_t, F_t, G_t) + K_t + W_t + D_t. \end{aligned} \tag{2}$$

(v) Let $h_0 = h_0 + A_{63}, h_1 = h_1 + B_{63}, h_2 = h_2 + C_{63}, h_3 = h_3 + D_{63}, h_4 = h_4 + E_{63}$, $h_5 = h_5 + F_{63}, h_6 = h_6 + G_{63}, h_7 = h_7 + H_{63}$. Output $h_0 - h_7$.

In the above algorithm, $\sum_1(E_t)$, $\sum_0(A_t)$, $\mathrm{Maj}(A_t, B_t, C_t)$, and $\mathrm{Ch}(E_t, F_t, G_t)$ are logical functions, and W_t is updated according to

$$W_t = \begin{cases} M_t, & 0 \le t \le 15, \\ \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}, & 16 \le t \le 63. \end{cases} \tag{3}$$

From the processing of the SHA256 algorithm, it can be seen that the key is to update the values of A and E, which requires multiple addition operations and 64 cycles of iteration. Therefore, the optimization of these two operands will play an important role in reducing the time consumption of the algorithm.

3.2.2. Critical Path Segmentation Optimization. The time consumption of the SHA256 operation is mainly in the iteration part of Step 4, and the most time-consuming part is the calculation of A and E values. Therefore, adopting the method of critical path segmentation and combining with the parallel characteristics of FPGA computing resources can effectively shorten the time consumption.

H_t, K_t, and W_t in the critical path do not need additional logical operations or do not depend on other operands of the current round. Therefore, the critical path of the algorithm is divided into the following formulas:

$$S_t = H_t + K_t + W_t, \tag{4}$$

$$\begin{aligned} A_{t+1} &= \sum_1 (E) + \mathrm{Ch}(E_t, F_t, G_t) + S_t \\ &\quad + \sum_0 (A) + \mathrm{Maj}(A_t, B_t, C_t), \end{aligned} \tag{5}$$

$$E_{t+1} = \sum_1 (E) + \mathrm{Ch}(E_t, F_t, G_t) + S_t + D_t. \tag{6}$$

In this way, A and E values will be updated and shortened from the original $6t_{\mathrm{ADD}}$ and $5t_{\mathrm{ADD}}$ to $4t_{\mathrm{ADD}}$ and $3t_{\mathrm{ADD}}$, where t_{ADD} denotes the time consumption of addition operations.

3.2.3. Minimum Addition Optimization. FPGA is suitable for bit operation. Carry-Save Adders (CSA) strategy can reduce addition operation, minimize critical path length, and ensure pipeline throughput. For n-bit binary numbers a, b, and c, the CAS operations are as follows:

$$\begin{aligned} S(a, b, c) &= a \wedge b \wedge c, \\ \mathrm{Ca}(a, b, c) &= [(ab) \mid (bc) \mid (ac)] \ll, \\ \mathrm{CSA}(a, b, c) &= S(a, b, c) + \mathrm{Ca}(a, b, c) = a + b + c. \end{aligned} \tag{7}$$

By dividing the critical paths, it takes $2t_{\mathrm{ADD}}$, $4t_{\mathrm{ADD}}$, and $3t_{\mathrm{ADD}}$ to calculate S_t, A_{t+1}, and E_{t+1}, respectively. Since the addition operation consumes a lot of time on the FPGA, the CSA method should be used to increase bit operation and reduce the addition operation, in order that the total time consumption can be reduced. By using the critical path

partitioning method and CSA strategy, formulas (4)~(6) are replaced by CSA operation in the following formulas:

$$S_t = \mathrm{CSA}\left(H_t, K_t, W_t\right), \tag{8}$$

$$A_{t+1} = \mathrm{CSA}\left(\mathrm{CSA}\left(\sum_1(E), \mathrm{Ch}\left(E_t, F_t, G_t\right), S_t\sum_0(A), \mathrm{Maj}\left(A_t, B_t, C_t\right)\right)\right), \tag{9}$$

$$E_{t+1} = \mathrm{CSA}\left(\sum_1(E), \mathrm{Ch}\left(E_t, F_t, G_t\right), S_t\right) + D_t. \tag{10}$$

The critical path segmentation method and the CSA strategy reduce the operation of A_{t+1} and E_{t+1} to only $2t_{\mathrm{ADD}}$, thus improving the efficiency of the algorithm.

3.2.4. Pipeline Optimization. After the optimization of critical path partition, the time consumption of the longest path is reduced. For serial computing, the total time consumption does not decrease. Therefore, it is necessary to use the parallel characteristics of FPGA and pipeline method for optimization, so as to truly reduce the total time consumption of computing.

According to the characteristics of the SHA256 algorithm and the optimization of critical path, the core processing of the algorithm is divided into three modules: W module, split S module, and update module $A-H$. The pipelining technology reduces time consumption by increasing resource utilization. Therefore, each module needs 64 computing units and a total of 192 computing units.

While data are being calculated, in the first clock cycle, the first data are input to the W_0 computing unit for processing in the first clock cycle. In the second clock cycle, the output of W_0 is taken as the input of $S0$, and W_1 is calculated. At the same time, the second data are input to W_0. In the third clock cycle, three computing units are processed in parallel, and so on. Until the 66th clock cycle, when all 192 units are running, the output of the first data is completed. When there is a large amount of data to be computed, one type of data is computed in a clock cycle, which reduces the time consumed by 64 iterations in the algorithm. Therefore, the throughput and resource utilization of the algorithm are greatly improved. The pipeline structure of the SHA256 algorithm is shown in Figure 6.

3.3. Communication and Network Optimization

3.3.1. Communication Optimization. For adapting to the calculation of blockchain hash, the concrete structure of proactive reconfigurable computer is shown in Figure 7, which mainly includes Hash_Core, I_10G, CTL_DDR3_0/1, State_U, Ctl_Core, and I_1G modules.

The functions of each module are as follows:

(i) *Hash_Core module*. The core processing module of hash computing is mainly responsible for hash calculation of blockchain data, which is implemented in full-pipeline mode and supports hash calculation of SHA1, SHA256, and so forth.

(ii) *I_10G module*. The data communication interface circuit based on 10,000 Mega mainly includes 10,000 Mega MAC interface, data buffer, and interface of module on the same chip. The module is mainly responsible for the input of data to be processed and the recovery of calculation results.

(iii) *CTL_DDR3_0 module*. The data communication interface circuit based on DDR3 mainly includes DDR3 interface, data buffer, and interface of on-chip module. This module is mainly responsible for data memory reading.

(iv) *CTL_DDR3_1 module*. The data communication interface circuit based on DDR3 mainly includes DDR3 interface, data buffer, and interface of on-chip module. This module is mainly responsible for data memory writing.

(v) *State_U module*. Acquire the on-chip state of each module, and then output it to Ctl_Core.

(vi) *Ctl_Core module*. The processor-based on-chip processing control core is mainly responsible for reporting the running state of the mimic computer and processing the control information.

(vii) *I_1G module*. Data communication interface based on Gigabit Ethernet interface is mainly used for communication of control information.

Block data are cached to CTL_DDR3_0 via I_10G network interface, hash values are read and calculated by Hash_Core, and results are cached into CTL_DDR3_1 and finally sent to the network by I_10G. The host computer controls the proactive reconfigurable computer in real time through I_1G Gigabit interface and Ctl_Core according to the information reported by State_U.

3.3.2. 10G Network. 10G network is implemented based on IP protocol, and the content of data transmission is controlled by external users. It uses FIFO interface to communicate with external devices [15]. In the process of transmitting control messages, if the receiver does not have an ARP response, the system will issue a timeout error because ARP does not respond; if there is a timeout transmission, the system will show the number of times of timeout transmission. If the transmission succeeds, the successful message will be returned; if the transmission fails, the error message which is retransmitted overtime will be returned. If there is a timeout and no information is received, the system will send out the wrong signal of communication channel, according to which the user will take appropriate action accordingly. The whole structure is shown in Figure 8.

In Figure 8, the sending port includes two FIFOs: the sending data FIFO (ip_snd_fifo) and the sending status FIFO (ip_snd_status_fifo). The sending data FIFO's depth is 65 bits and low 64 bits are data interface. The highest bit

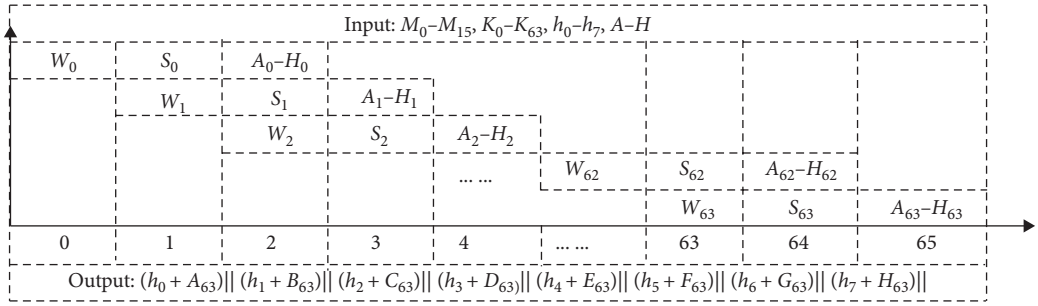


Figure 6: Pipeline structure of the SHA256 algorithm.

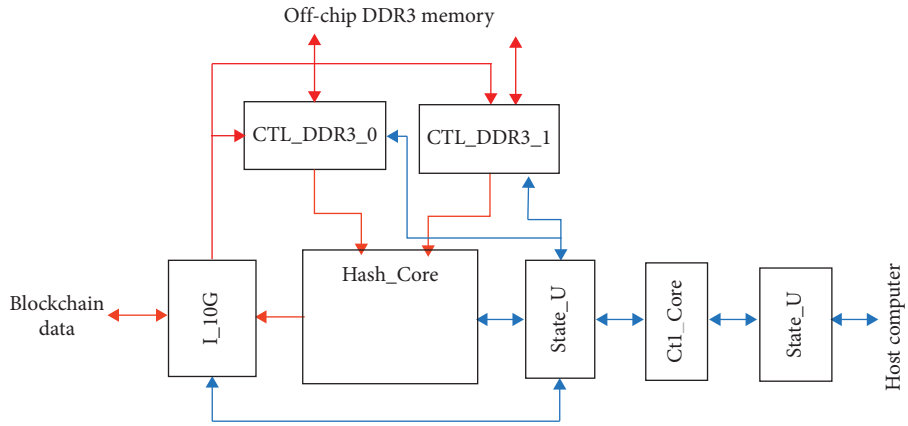


Figure 7: On-chip architecture of proactive reconfigurable computer.

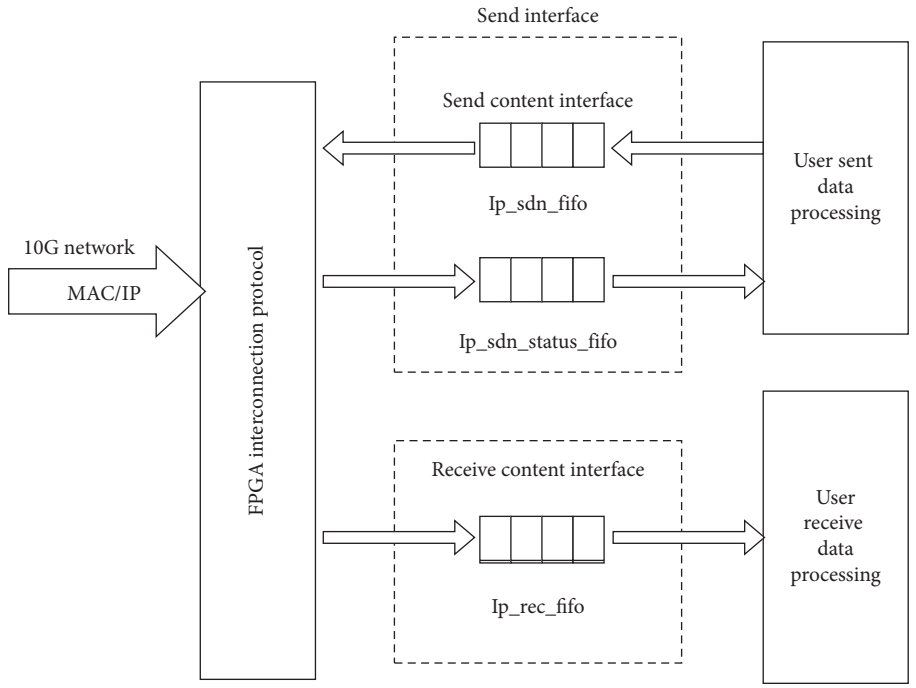


Figure 8: The whole structure of 10G network.

indicates whether the data transmission is the last one. If more than 1440 bytes of data are to be transmitted, multiple transfers are required. The sending status FIFO is used to identify whether there is an error in the data transmission. If there is an error like the timeout in the process of data transmission, all subsequent contents will be read out until the last one. Each data transmission corresponds to a state FIFO write. The receiving port has only one FIFO, that is, the

receiving data FIFO (ip_rec_fifo), which has a depth of 65 bits and low 64 bits as the data interface. The highest bit indicates whether the data transmission is the last frame of data, and the data received is identified by index number.

3.3.3. Memory Management. Read-write memory is implemented by four groups of FIFOs in burst mode. Every time before it reads and writes memory, it will calculate the memory address range according to the length of the data and store it in wrrdinfo_fifo. At the same time, the data will be cached in wfifo_fifo, and according to the information of wrrdinfo_fifo, the read-write arbitration module determines whether it is a reading operation or a writing one. If it is a writing operation, the data will be written to memory through the DDR write module. The process of reading memory data is similar to that of writing. The read information and data will be cached in out_rdinfo_fifo and rififo_fifo, respectively. The read-write structure of memory is shown in Figure 9, where the size of request information wrrdinfo_fifo and out_rdinfo_fifo is $16 * 64$ bits, and the size of reading and writing wfifo_fifo and rififo_fifo is $4096 * 64$ bits.

When the initialization of memory is completed, that is, phy_init_done is set to 1, the CTL_DDR3_0 and CTL_DDR3_1 modules are in the read-write state, and the read-write state jump will be completed according to the wrrdinfo_q[0] identifier bit, as shown in Figure 10. When it begins reading and writing memory, the address of memory will be counted according to the length of writing, and the reading and writing of the whole data will be completed. After the reading and writing operation is completed, it will jump to the idle state and wait for the next operation.

3.4. Application of PRCA Blockchain. Public and private keys in blockchains are a pair of keys obtained by a kind of algorithm. It will be encrypted with public key and decrypted with corresponding private key. After three times of SHA256 computation and one time of RIPEMD160 computation for the public key, a public key hash can be obtained, and the address can finally be obtained through base58 encoding [16]. Merkle tree is a kind of tree structure. In trading with blockchains, every transaction is hashed, and the final root is Merkle root [17]. Proof-of-work (PoW) is called mining in blockchains. CPU calculation uses the complexity of hash operation to determine PoW, and it will produce a value smaller than the specified target [18]. Block filter proposed in the blockchain is a fast search based on hash function, which can quickly determine whether a retrieved value exists in the searched set [19]. The application of hash algorithm in blockchain is shown in Figure 11.

In this paper, the communication equipment and network are optimized. In a relatively safe environment, a relatively simple and lightweight hash algorithm is chosen to replace the complex hash algorithm, so as to improve the running speed of the system and reduce the energy consumption of the system. Meanwhile, multiple hash algorithm is used to reduce the attack of length expansion and ensure the integrity and tamper-proofing of information, which reflects the security performance of blockchain.

4. Experimental Analysis

In this paper, proactive reconfigurable computer is used for experiments. The software platform is ISE software integrating design, simulation, integration, wiring, and generation. First, the comparison of CPU running speed and resource utilization is given by optimizing the hash algorithm deeply. Second, the collision resistance of proactive reconfigurable hashes is analyzed. Finally, the security of this scheme is analyzed from many aspects.

The configuration information of each computing unit used in the experiment is shown in Table 1.

4.1. Performance Analysis. On the proactive reconfigurable computer, the SHA256 and SHA1 algorithms are implemented, respectively. Their resource occupation, frequency, and throughput are shown in Table 2.

As seen from Table 2 and Figure 12, SHA256 and SHA1 implemented in a pipelined manner occupy less than 10% of the resources but with high throughput.

Next is the performance comparison of SHA256 and SHA1 between the proactive reconfigurable computer and CPU, as is shown in Table 3.

From Table 3, it can be seen that the proactive reconfigurable computer can realize the parallelism of multiple modules and can fully meet the application requirements of hash computing in blockchain. Taking Bitcoin three hash as an example, three SHA256 combinations are connected in series to form a cascade pipeline. The data can be directly input into the pipeline without waiting, and the results are output sequentially by the end, which is very efficient. Contrastively, CPU can only rely on multithreaded concurrency to improve computing performance, and its essence is still serial execution, which will not be competent for blockchain applications requiring large amounts of computing.

Meanwhile, the proactive reconfigurable computer is equipped with a 10-gigabit network, whose data transmission peak is about 10 Gbps, which can meet the communication requirements of blockchain high-frequency transactions. As each clock cycle can transmit 8 bytes of data, the clock frequency is 156.25 MHz; while the FIFO interface and frequency of DDR are 8 bytes and 156.25 MHz, the data transmitted by 10G network can be synchronized through FIFO cache and written into memory with 64 bytes and 300 MHz. Two memory modules are configured: one is responsible for writing operation of 10G network and reading operation of hash module, and the other is responsible for writing operation of hash module and reading operation of 10G network. The two memory modules work independently, which improves the efficiency of data transmission.

4.2. Antiattack Analysis. Hash operation is irreversible and gets different values for different contents. Any change of input information will lead to significant changes in hash

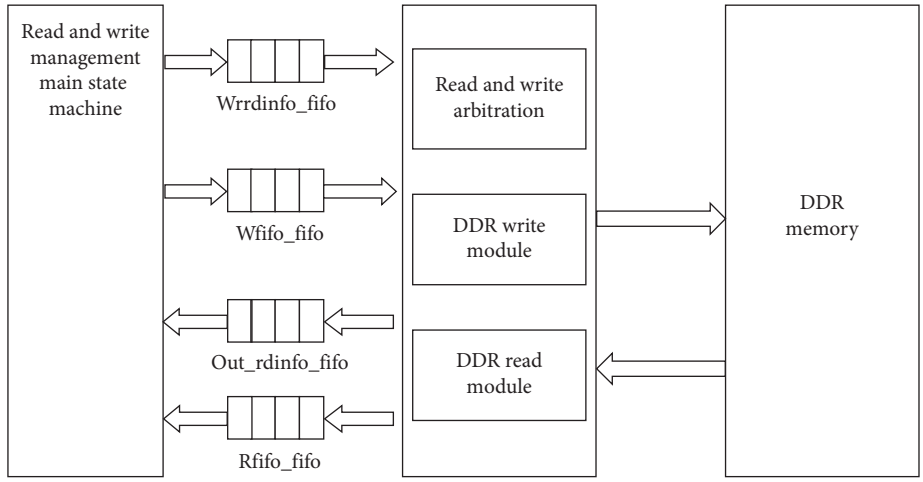


Figure 9: The read-write structure of memory.

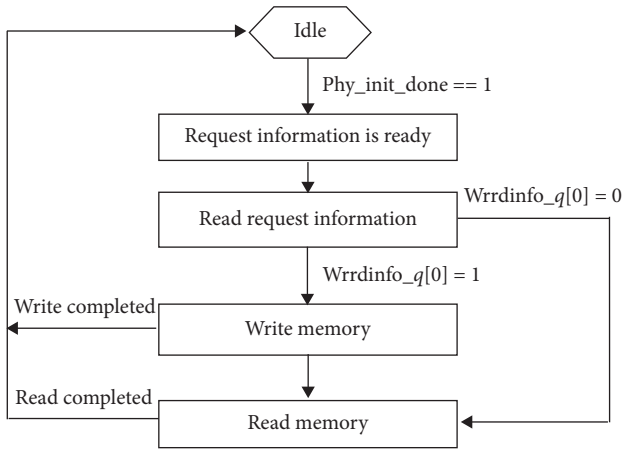


Figure 10: Memory state management mechanism.

results. Moreover, hash operation is also anticollision; that is, two pieces of information with the same hash result cannot be found, which can effectively prevent differential attack [20].

Assuming that the output value of hash function is uniformly distributed and the message digest has m bits, the hash value has $n = 2^m$ possible outputs. For any k ($k \le n$) random input, the probability of at least one collision is

$$\begin{aligned} p(n,k) &= 1 - \frac{n!}{(n-k)! \times n^k} \\ &= 1 - \left[\left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right) \cdots \left(1 - \frac{k-1}{n}\right)\right] \\ &= 1 - \prod_{i=1}^{k-1}\left(1 - \frac{i}{n}\right) \\ &\approx 1 - \coprod \\ &= 1 - e^{-(k(k-1)/2n)}. \end{aligned} \tag{11}$$

If $p(n,k) > 0.5$, that is, $1 - e^{-(k(k-1)/2n)} = 1/2$, then $\ln 2 \approx (k^2/2n)$; this means $k \approx \sqrt{n}$.

According to the above calculation, if the hash function has an output digest of m bits, then only $k = 2^{m/2}$ attempts will result in a collision with a probability of at least 50%. SHA1 and SHA256 are operations of 2^{160} and 2^{256} orders of magnitude, respectively. Table 4 gives the threshold of hash function conflict.

Bitcoin obtains hash data through the SHA256 algorithm and runs two iterations in block trading to mitigate the length expansion attack. PRCA blockchain system can be described by a triple tuple as $\Omega = \{\text{Block}, \text{Hash}, \text{Num}\}$, where "Block" represents block data, "Hash" represents hash algorithm, and "Num" represents iteration times. The multiple phases of Ω have many different hash combination schemes and can be represented by $\Omega = \{\text{Block}(t), \text{Hash}(t), \text{Num}(t)\}$ at time t, which is dynamic, diverse, and random.

The hash algorithm of PRCA blockchain system Ω is dynamically reconfigurable. After negotiation, the hash algorithm can be reconstructed dynamically and partially to complete the switching of different algorithms. In addition, "Block" is changing constantly, and the content of each transaction is unpredictable and completely different. Finally, "Num" can be negotiated by both sides to improve its security by increasing the number of iterations without significantly increasing the amount of computation. Obviously, the blockchain based on PRCA not only improves the complexity of internal hash operation but also combines the hash to increase the length of output, which greatly hinders the attackers from extending the blockchain and reduces the probability of collision.

4.3. Security Performance Analysis. Encryption of information is the key link of blockchain, which mainly includes hash function and asymmetric encryption algorithms [21]. Asymmetric encryption uses private key to prove the ownership of the node and is implemented by digital signature. Hash algorithm is used to transform the input of any

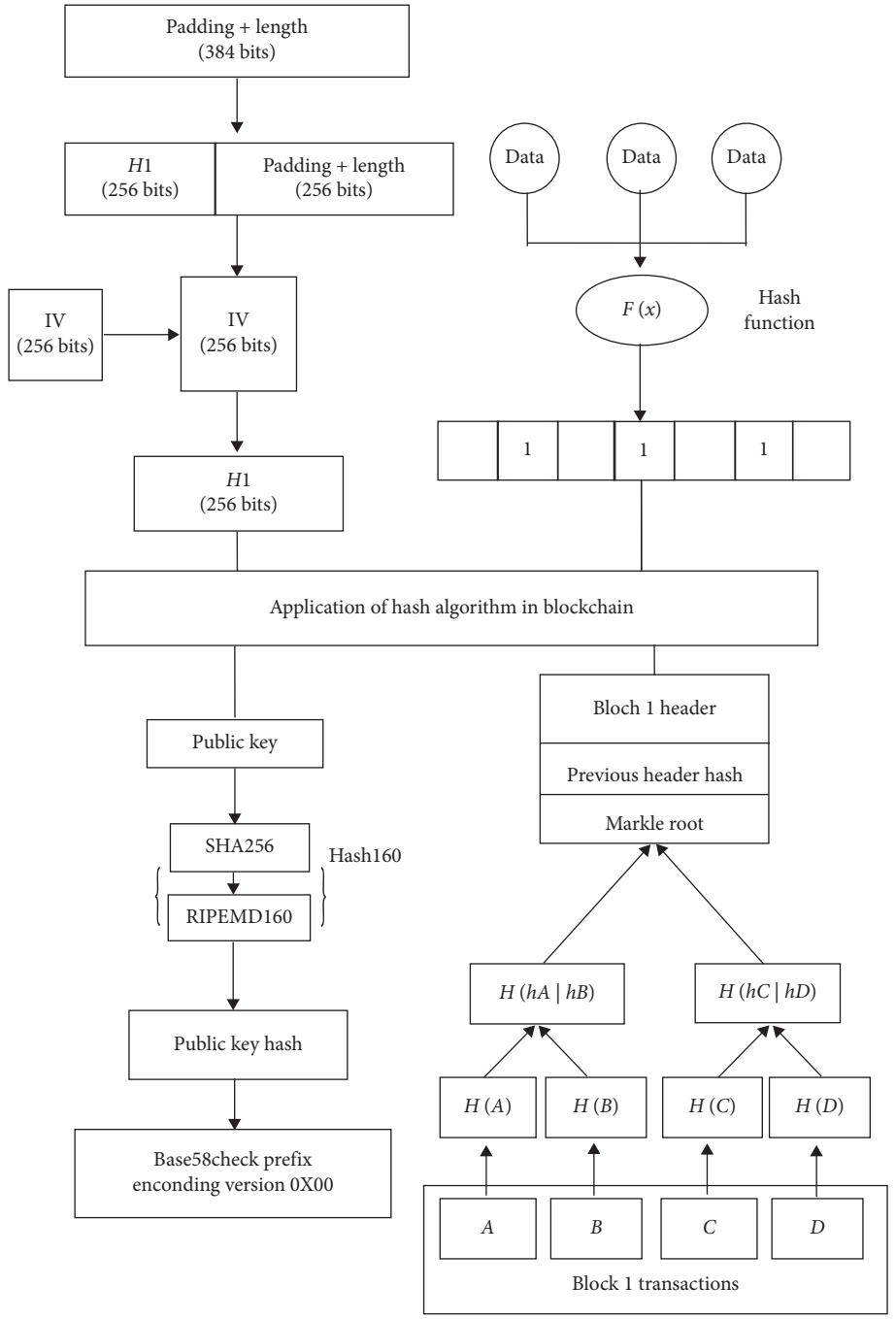


Figure 11: The application of hash algorithm in blockchain.

Table 1: The configuration information of each computing unit.

Calculation component	Configuration information
CPU server	4-core CPU; model: i5-7500; main frequency: 3.40 GHz; memory: 24 GB
PRCA	4 FPGA cards; on-chip resources slices: 85920; memory: 24 GB
10G switch	24 1/10G SFP + ports; 4 10/100/1000 m electrical interface

Table 2: The actual operation of SHA256 and SHA1.

	Regs (687, 360)	LUTs (343, 680)	Slices (85, 920)	Frequency (MHz)	Throughput (Mbps)
SHA1	24,703	18,899	6106	243.8	124825.6
SHA256	27669	25648	7745	172.0	88064

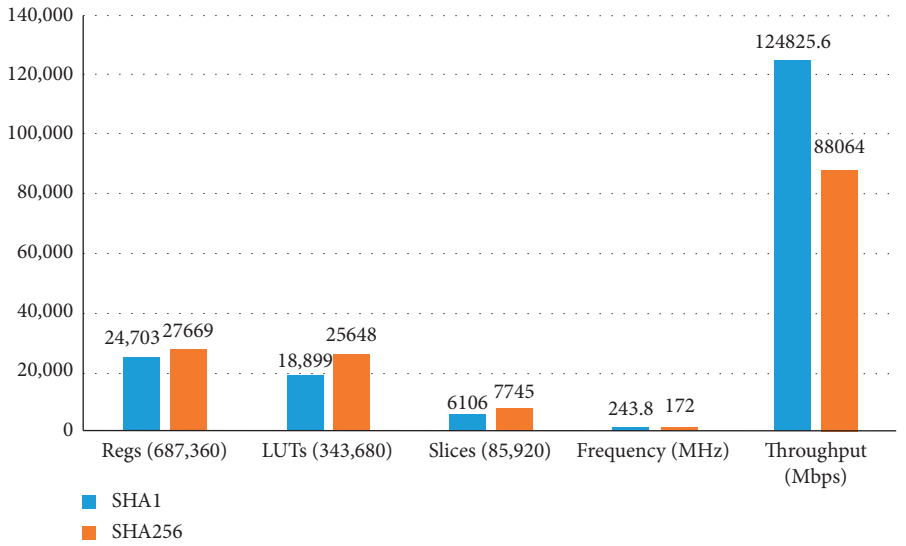


Figure 12: The actual operation of SHA256 and SHA1.

Table 3: The performance comparison of SHA256 and SHA1 between the proactive reconfigurable computer and CPU.

Calculation component	Number of parallel modules		Frequency (MHz)	Running speed (m)
PRCA	SHA1	40	200	8000
	SHA256	24	150	4800
CPU	SHA1	—	—	270.6
	SHA256	—	—	119.3

Table 4: The threshold of hash function conflict.

Hash function	Function collision threshold
SHA1	$2^{80} \approx 1.2 \times 10^{24}$
SHA256	$2^{128} \approx 3.4 \times 10^{38}$

length into an output of fixed length consisting of letters and numbers, which is irreversible and tamper-proofing.

From the perspective of information security, the main advantages of this scheme are as follows:

(i) Multiple hash algorithms are jointly used to ensure the integrity and nontampering of information

(ii) There is a pseudorandom dynamic selection and the hash algorithm is updated to increase the difficulty of attack in time dimension

(iii) By using the hardware implementation of proactive reconfigurable computer, the attack surface is expanded and the attack threshold is raised

Obviously, the blockchain based on PRCA enhances the confidentiality, authenticity, and integrity of data and enhances the overall security of blockchain transactions with its reliability, security, and tamper-resistance.

5. Conclusions

In order to improve the efficiency and security of blockchain hash algorithm, a scheme of blockchain hash algorithm optimization based on PRCA is proposed in this paper. This scheme combines blockchain with proactive reconfigurable computer to improve the performance of blockchain hash function. In terms of security performance, several lightweight hash algorithms are used to exchange information to ensure the integrity and tamper-proofing of information. The proactive reconfigurable computer hardware is used to expand the attack surface, improve the attack threshold, and ensure the security of blockchain.

Blockchain security is the most important part of the system, which includes data, intelligent contract, privacy protection, and application risk. Meanwhile, the data of blockchain is unique. Under the condition of its own security, data writing cannot be changed. Based on the security problem of data immutability, the data structure,

cryptography technology, and communication network at the bottom of blockchain are improved to promote the healthy development of blockchain application.

Data Availability

The data used support the findings of the study are available from the corresponding authors upon request.

Additional Points

Highlights. In this paper, proactive reconfigurable computer is used for experiments. The software platform is ISE software integrating design, simulation, integration, wiring, and generation. First, the comparison of CPU running speed and resource utilization is given by optimizing the hash algorithm deeply. Second, the collision resistance of proactive reconfigurable hashes is analyzed. Finally, the security of this scheme is analyzed from many aspects.

Conflicts of Interest

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

Acknowledgments

This research work was supported by the Innovative Research Groups of the National Natural Science Foundation of China (61521003), Intergovernmental Special Programme of National Key Research and Development Programme (2016YFE0100300 and 2016YFE0100600), National Scientific Fund Programme for Young Scholar (61672470), and Science and Technology Project of Henan Province (182102210617).

References

[1] Q. Lu and X. Xu, "Adaptable blockchain-based systems: a case study for product traceability," *IEEE Software*, vol. 34, no. 6, pp. 21–27, 2017.

[2] M. Padmavathi and R. M. Suresh, "Secure P2P intelligent network transaction using Litecoin," *Mobile Networks and Applications*, vol. 24, no. 2, pp. 318–326, 2018.

[3] I. Bentov and R. Kumaresan, "How to use Bitcoin to design fair protocols," *Lecture Notes in Computer Science*, vol. 8617, pp. 421–439, 2017.

[4] P. Katsiampa, "Volatility estimation for Bitcoin: a comparison of GARCH models," *Economics Letters*, vol. 158, pp. 3–6, 2017.

[5] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "MeDShare: trust-less medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. 5, no. 99, pp. 14757–14767, 2017.

[6] G. Liang, S. R. Weller, F. Luo, J. Zhao, and Z. Dong, "Distributed blockchain-based data protection framework for modern power systems against cyber attacks," *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 162–173, 2019.

[7] W. Pennington and J. Evans, "Blockchain-enabled, subscriber-based capital markets index data distribution," *The Journal of Index Investing*, vol. 7, no. 4, pp. 83–87, 2017.

[8] S. X. Xi, W. N. Zhang, Q. L. Zhou, S. XueMing, and B. Li, "High-throughput implementation of SHA512 algorithm based on mimetic computer," *Computer Engineering and Science*, vol. 40, no. 8, pp. 1344–1350, 2018.

[9] S. X. Chen, X. Y. Jiang, J. J. Cai, J. Y. Liu, and W. ChunMing, "Research on mimic security gateway technology based on attack transfer," *Journal of Communications*, vol. 39, no. S2, pp. 76–82, 2018.

[10] J. Steckert and A. Skoczen, "Design of FPGA-based radiation tolerant quench detectors for LHC," *Journal of Instrumentation*, vol. 12, no. 4, p. T04005, 2017.

[11] H. Xu, X. Chen, J. Zhou, Z. Wang, and H. Xu, "Research on basic problems of cognitive network intrusion prevention," in *Proceedings of the 2013 Ninth International Conference on Computational Intelligence and Security*, pp. 514–517, Leshan, China, December 2013.

[12] H. Li, R. Lu, L. Zhou, B. Yang, and X. Chen, "An efficient merkle-tree-based authentication scheme for smart grid," *IEEE Systems Journal*, vol. 8, no. 2, pp. 655–663, 2014.

[13] Q. Wen, D. Wang, S. Feng, Y. Zhang, and G. Yu, "A novel cross-modal hashing algorithm based on multimodal deep learning," *Science China (Information Sciences)*, vol. 60, no. 9, pp. 50–63, 2017.

[14] Y. Kano and T. Nakajima, "A novel approach to solve a mining work centralization problem in blockchain technologies," *International Journal of Pervasive Computing and Communications*, vol. 14, no. 1, pp. 15–32, 2018.

[15] L. Xue, L. Yi, H. Ji, P. Li, and W. Hu, "Symmetric 100-Gb/s TWDM-PON based on 10g-class optical devices enabled by dispersion-supported equalization," *Journal of Lightwave Technology*, vol. 36, no. 2, pp. 580–586, 2018.

[16] A. S. Konoplev, A. G. Busygin, and D. P. Zegzhda, "A blockchain decentralized public key infrastructure model," *Automatic Control and Computer Sciences*, vol. 52, no. 8, pp. 1017–1021, 2018.

[17] H. Liu, A. Kadir, X. Sun, and Y. Li, "Chaos based adaptive double-image encryption scheme using hash function and *S*-boxes," *Multimedia Tools and Applications*, vol. 77, no. 1, pp. 1391–1407, 2018.

[18] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and C. Yang, "The blockchain as a decentralized security framework (future directions)," *IEEE Consumer Electronics Magazine*, vol. 7, no. 2, pp. 18–21, 2018.

[19] W. Liu, W. Qu, J. Gong, and K. Li, "Detection of superpoints using a vector bloom filter," *IEEE Transactions on Information Forensics & Security*, vol. 11, no. 3, pp. 514–527, 2017.

[20] W. H. Zhou, N. D. Jiang, and C. C. Yan, "Research on anti-collision algorithm of RFID tags in logistics system," *Procedia Computer Science*, vol. 154, pp. 460–467, 2019.

[21] D. Yaseen Khudhur, S. Saad Hameed, and S. M. Al-Barzinji, "Enhancing e-banking security: using whirlpool hash function for card number encryption," *International Journal of Engineering and Technology*, vol. 7, no. 2, pp. 281–286, 2018.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8824163, 10 pages
https://doi.org/10.1155/2020/8824163

Research Article

Game Theoretical Method for Anomaly-Based Intrusion Detection

Zhiyong Wang,[1] Shengwei Xu,[2] Guoai Xu,[1] Yongfeng Yin,[3] Miao Zhang,[1] and Dawei Sun[4]

[1]*National Engineering Laboratory of Mobile Network Security, School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, China*
[2]*Deputy Director of Information Security Research Institute, Beijing Institute of Electronic Science and Technology, Beijing, China*
[3]*School of Reliability and Systems Engineering, Beihang University, Beijing, China*
[4]*R&D Department, Beijing Softsec Technologies Co., Ltd., Beijing, China*

Correspondence should be addressed to Yongfeng Yin; yyf@buaa.edu.cn

Received 8 May 2020; Revised 24 June 2020; Accepted 19 August 2020; Published 4 September 2020

Academic Editor: Xiaolong Xu

In this paper, the game theoretical analysis method is presented to provide optimal strategies for anomaly-based intrusion detection systems (A-IDS). A two-stage game model is established to represent the interactions between the attackers and defenders. In the first stage, the players decide to do actions or keep silence, and in the second stage, attack intensity and detection threshold are considered as two important strategic variables for the attackers and defenders, respectively. The existence, uniqueness, and explicit computation of the Nash equilibrium are analyzed and obtained by considering six different scenarios, from which the optimal detection and attack actions are provided. Numerical examples are provided to validate our theoretical results.

1. Introduction

Nowadays, network devices and communication services are vulnerable to various kinds of intrusion attacks, such as DoS/DDoS, false data injection, and botnet attacks. The intrusion attacks tend to be more intelligent and the unexpected attack modes arise frequently. Consequently, great challenges are brought into network security control and management. As one of the most important techniques to tackle with various attacks, anomaly-based intrusion detection system (A-IDS) has been widely adopted in almost all kinds of network environments [1, 2]. An anomaly-based intrusion detector attempts to estimate the normal behavior with a profile and generates an anomaly alarm once the profile collected from real-time observation exceeds a predefined threshold [2].

In an intrusion detection system, the attacker and defender can naturally be regarded as two players who try to maximize their payoffs, respectively, by executing certain optimal strategies. Thus, the game theoretical method is an effective tool which enables a defender to earn the maximum payoff (or the minimum loss) while fighting with the attacks. A number of results on game theory-based intrusion detection methods have been reported for different network environments and security requirements. Excellent surveys about this topic can be found in [3–6]. In [7], two-player noncooperative strategic game models are established for some general intrusion detection problems and Nash equilibriums are analyzed explicitly. In [8–11], game theoretical intrusion detection methods are investigated to solve the security resource allocation problems of large-scale heterogeneous networks. Note that, in [8–10], it is assumed that the defender scan always correctly identify the malicious behaviors of the attackers without any errors, while such an assumption may not be satisfied in some cases. For example, for intelligent APT attacks, the attackers often disguise themselves as no attack happens, which may make the detector to not always preciously identify the malicious actions. To handle these uncertainties, Bayesian games are

considered in intrusion detection by updating the defender's belief to her/his opponent based on the past behaviors [12–15]. The main idea of Bayesian game-based intrusion detection is to use probability to represent the uncertainties and further use Bayesian iteration to update the dynamics. For self-organizing ad hoc networks, some nodes may be malicious and how to detect the malicious actions is an important work. Some strategic games are presented to stimulate the cooperation among distinct regular nodes, based on which the hidden malicious nodes can be detected [16–21]. In [22], a two-player Stackelberg stochastic game is analyzed for achieving the best response against the intrusion. In [23, 24], game theory-based analysis methods for distributed intrusion detection are proposed, where consensus-based distributed detection method is presented and then game analysis is provided for the optimal defense and attack strategies. In [25], the privacy defense problem is also considered in the collaborative security scheme design problem by using the game theoretical analysis method. In [26], a differential game model is established to analyze the dynamic process of the attack and defense.

In a game between an attacker and a defender, the rational attacker will not launch an attack otherwise she/he can get a positive payoff. Moreover, the attack intensity needs be chosen to maximize her/his positive payoff. On the contrary, the defender will perform a defense action to resist the attack according to a similar rule. In an A-IDS, a predefined detection threshold needs be cautiously determined. In general, a higher threshold with a larger normal coverage area will result in a smaller false alarm rate but a larger missing alarm rate. Note that the missing alarm rate is also closely related to the attack intensity. More specifically, larger attack intensity will cause a lower missing alarm rate. Though attack intensity and detection threshold are two important factors affecting the false and missing alarm rates, which correspond to the payoffs of attackers and defenders in an intrusion detection game, they are seldom considered in the aforementioned results. In most of the aforementioned works, the false and missing alarm rates are assumed to be known constants and only binary actions "do" or "not do" are considered in their game models. In [11], the detection threshold and attack intensity are considered, while the focus is mainly related to distributed resource allocation of the heterogeneous networks.

Motivated by the limitations mentioned above in the literature, a more realistic two-stage form of the intrusion detection game model is presented in this paper. The attack intensity and detection threshold are considered as two strategic variables. In the first stage, the attackers and defenders make decisions on whether the attack and defense actions should be executed, respectively. Once the attack/monitoring actions are decided to be executed, optimal attack intensity and detection threshold are determined to maximize their utilities in the second stage. The existence and uniqueness of the Nash equilibrium are discussed for the first stage of our presented game model under different scenarios, when the strategic variables of the second stage are restricted to certain regions. Then, the optimal attack intensity and detection threshold are derived for each scenario, correspondingly.

The contributions of this paper can be summarized as follows:

(1) A two-stage game model is presented for anomaly-based intrusion detection confrontation. In contrast to the existing work, where only binary actions "do" or "not do" are considered in the game model, the attack intensity and the detection threshold are considered as two key strategic variables, and the false and missing alarm rates are the functions of the attack intensity and the detection threshold, instead of being assumed to be constant. The two stages of the game model are tightly coupled with each other and thus the game model is more complex.

(2) The existence, uniqueness, and calculation of Nash equilibriums are discussed. Based on the results, optimal selections of attack intensity and detection threshold for achieving the maximum payoffs of the attackers and defenders are provided. The results provide a new method to determine the detection threshold in the defense, from the perspectives of the optimization and confrontation. So, the presented game model and Nash equilibrium solution give a more realistic theoretical analysis framework for the anomaly-based security detection.

The rest of this paper is organized as follows. In Section 2, some definitions are introduced and a two-stage game model of the A-IDS is presented. In Section 3, the Nash equilibrium of the proposed game model is analyzed. Simulation results are given to show the effectiveness of our game theoretical analysis methods in Section 4, followed by the conclusions of the paper summarized in Section 5.

2. A Two-Stage Intrusion Detection Game Model

Suppose that there is a network unit vulnerable to intrusion attacks. Typical examples for such a unit include a software system, network equipment, and a communication channel. Here, we adopt similar attack and A-IDS detection models as that in [11]. The strategic form of two-player noncooperative game is given in Table 1. U_A and U_D denote the payoffs of the attacker and the defender, respectively.

In the following, we give the physical meanings of the corresponding variables in Table 1. The variable x denotes the attack intensity, for example, the number of attack packets in a DoS/DDoS attack, or the number of bogus packets in a DNS cache poisoning attack or jamming strength in a communication attack, or the magnitude of false data injection. It is assumed that $x \in [\underline{x}, \overline{x}]$, where $1 \geq \overline{x} > \underline{x} > 0$. The function $s(x) \in \mathfrak{R}$ is used to represent the extent of damage to the security of the unit, when it is suffered from an attack with intensity x. It is natural to consider $s(x)$ as a strictly increasing function such that $(\partial s(x)/\partial x) > 0$ and $s(x) \in [\underline{s}, \overline{s}]$ with $1 \geq \overline{s} > \underline{s} > 0$. The term $c_1 W + u(x)W$, where $c_1 \in (0, 1)$ is a constant, W is the security asset of the unit, and $u(x)$ is a strictly increasing function, denotes the cost of launching the attack. The

Table 1: Strategic form of the local game.

	Monitor	Not monitor
Attack	$U_A(x,y) = q(x,y)s(x)W - c_1W - u(x)W$, $U_D(x,y) = -q(x,y)s(x)W - c_2W$	$U_A(x,y) = s(x)W - c_1W - u(x)W$, $U_D(x,y) = -s(x)W$
Not attack	$U_A(x,y) = 0$, $U_D(x,y) = -p(y)c_3W - c_2W$	$U_A(x,y) = 0$, $U_D(x,y) = 0$

variable y denotes the detection threshold. It is assumed that $y \in (\underline{y}, \overline{y})$ with $\overline{y} > \underline{y} > 0$ and a larger y corresponds to a larger coverage area for normal behavior. The function p denotes the false alarm rate, i.e., it represents the probability that an alarm is generated though no attack is activated. Obviously, p is determined completely by the threshold y and $p(y)$ is a strictly decreasing function in this paper. The function q denotes the missing alarm rate, i.e., it represents the probability that no alarm is generated though an attack is executed. The function q is determined by both attack intensity x and threshold y. It can be easily derived that q is strictly decreasing and increasing with respect to x and y, respectively. The parameters $c_2 \in (0,1)$ and $c_3 \in (0,1)$ are two constants.

Clearly, the game model described in Table 1 contains the following two stages. In the first stage, the optimal strategy set "Attack/Not attack" and "Monitor/Not monitor" needs be determined by the attacker and defender. Then, both players proceed to the next stage to select optimal attack intensity x and detection threshold y. For better understanding, the two-stage pure-strategic intrusion detection game model with one attacker and one defender is described in Table 2 in a more rigorous way.

Remark 1. The attack and detection models are similar to that in [11], while the results of [11] mainly consider the attack and defense resource allocation problem for heterogeneous distributed networks. In this paper, we consider the confrontation problem for one network unit, as expressed by the game model in Table 2. Thus, it is essentially different from the work in [11]. Besides, we establish a two-stage game model by considering the attack intensity and detection threshold as the key strategic variables, which is also different from the existing works.

3. Nash Equilibrium Analysis of the Game

As mentioned in Section 2, the attacker/defender needs to decide whether to launch an attack/to monitor the unit or keep silence in the first stage of the presented game model. For simplicity, an extra assumption is imposed that if the payoffs of a player choosing to perform the action and to keep silence are the same, she/he will keep silence. In other words, the attacker/defender tends to do nothing if she/he cannot earn larger payoffs by launching an attack/monitoring. Note that the value of W has no impact on the analysis of Nash equilibrium (*hereinafter referred to as NE*) of the game from Table 1. Thus, without loss of generality, we set $W = 1$.

Denote the feasible set of x and y by π with $\pi \in [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$. For convenience in later analysis, π is divided into the following subsets:

$$\begin{aligned}
\pi_1 &= \{(x,y) \in \pi:\ s(x) - c_1 - u(x) \le 0\},\\
\pi_2 &= \{(x,y) \in \pi:\ s(x) - c_1 - u(x) \ge 0, -q(x,y)s(x) - c_2 \le -s(x)\},\\
\pi_3 &= \{(x,y) \in \pi:\ q(x,y)s(x) - c_1 - u(x) \ge 0, -q(x,y)s(x) - c_2 \ge -s(x)\},\\
\pi_4 &= \{(x,y) \in \pi:\ q(x,y)s(x) - c_1 - u(x) < 0, s(x) - c_1 - u(x) > 0, -q(x,y)s(x) - c_2 > -s(x)\}.
\end{aligned} \tag{1}$$

It can be readily shown that $\pi_1 \cup \pi_2 \cup \pi_3 \cup \pi_4 = \pi$ and $(\pi_1 \cup \pi_2 \cup \pi_3) \cap \pi_4 = \varnothing$. The results of NE for the game as described in Tables 1 and 2 will be obtained from the following scenarios. In Scenario L.1, only one subset of π_1, π_2, π_3, and π_4 is nonempty. In Scenarios L.2~L.5, π_4 is empty while at least two subsets of π_1, π_2, and π_3 are nonempty. In Scenario L.6, π_4 and at least one subset of π_1, π_2, and π_3 are nonempty. Clearly, there is no overlap between any two scenarios and the six scenarios include all the possibilities. In the following, the sufficient and necessary conditions on x and y for the existence and uniqueness of NE are first derived for Scenarios L.1~L.6, respectively. Then, the optimal values of x and y, denoted by x^* and y^*, are further provided.

For convenient expression in what follows, two variables x' and x'' are first defined, i.e.,

$$x' = \arg\max_x q\left(x, \underline{y}\right)s(x) - u(x) - c_1, \quad \text{s.t. } x \in \left[\underline{x}, \overline{x}\right], \tag{2}$$

$$x'' = \arg\max_x s(x) - c_1 - u(x), \quad \text{s.t. } x \in [\underline{x}, \overline{x}]. \tag{3}$$

The optimization problems presented by (2) and (3) can be solved by classical optimization methods such as the gradient method and Lagrangian multiplier method [27].

Table 2: Two-stage pure-strategic intrusion detection game.

Players	Attacker, defender
Strategy sets	Attacker: Attack, not attack, attack intensity x Defender: Monitor, not monitor, detection threshold y
Constraints	$x \in [\underline{x}, \overline{x}]$, $\overline{x} > \underline{x} > 0$, $y \in [\underline{y}, \overline{y}]$, $\overline{y} > \underline{y} > 0$, $s(x) \in [\underline{s}, \overline{s}]$, $1 \geq \overline{s} > \underline{s} > 0$, $(\partial s(x)/\partial x) > 0$, $(\partial u(x)/\partial x) > 0$, $(\partial \overline{p}(y)/\partial y) < 0$, $(\partial q(x, y)/\partial x) < 0$, $(\partial q(x, y)/\partial y) > 0$, $p \in [p, \overline{p}]$, $q \in [q, \overline{q}]$, $1 > \overline{p} > p > 0$, $1 > \overline{q} > q > 0$, $c_1, c_2, c_3 \in (0, 1)$
Payoffs	U_A, U_D (see Table 1)
Game target	The players choose their strategies to maximize their payoffs U_A, U_D

Scenario L.1. Only one of the subsets π_1, π_2, π_3, and π_4 is nonempty.

The following conclusions can be drawn.

Theorem 1. *In Scenario L.1, the NE of the game, as described in Table 1, is derived as follows:*

(1) If only the subset $\pi_1 \neq \emptyset$, "not attack, not monitor" is the unique NE

(2) If only the subset $\pi_2 \neq \emptyset$, "attack, not monitor" is the unique NE and $x^* = x''$

(3) If only the subset $\pi_3 \neq \emptyset$, "attack, monitor" is the unique NE and $x^* = x'$, $y^* = \underline{y}$

(4) If only the subset $\pi_4 \neq \emptyset$, no NE exists

Proof. Firstly, the strategy combination "attack, not monitor" will not be the NE. This is because, $-p(y)c_3 - c_2 < 0$, the defender tends to "not monitor" the unit to earn zero payoff:

(1) If only $\pi_1 \neq \emptyset$, we have $q(x, y)s(x) - c_1 - u(x) < s(x) - c_1 - u(x) \leq 0$. This indicates that the attacker has no incentive to launch an attack either. Therefore, "not attack, not monitor" is the unique NE.

(2) If only $\pi_2 \neq \emptyset$, as the payoff of the attacker $s(x) - c_1 - u(x)$ is positive for any attack intensity x, the attacker will select "attack." Besides, the defender will never get more payoffs when she/he selects "monitor" as $-q(x, y)s(x) - c_2 \leq -s(x)$ for an arbitrary threshold y. Thus, the defender will select "monitor." The optimal attack intensity x^* should be derived by maximizing the payoff of the attack; therefore, $x^* = x''$ based on (3).

(3) If only $\pi_3 \neq \emptyset$, the attacker will always select "attack." This is because for any attack intensity x and detection threshold y the payoff of the attacker satisfies $s(x) - c_1 - u(x) > q(x, y)s(x) - c_1 - u(x) \geq 0$. Since the payoff of the defender satisfies $-q(x, y)s(x) - c_2 \geq -s(x)$ for an arbitrary y, the defender will select "monitor." Then, for the defender, the optimal threshold is computed by

$$y^* = \arg\max_y -q(x, y)s(x) - c_2, \quad \text{s.t. } x \in [\underline{x}, \overline{x}], y \in [\underline{y}, \overline{y}]. \tag{4}$$

Based on the property that $(\partial q(x, y)/\partial y) > 0$ in Table 2, we have $y^* = \underline{y}$. Then, the optimal attack intensity is given by $x^* = x'$ based on (2).

(4) If only $\pi_4 \neq \emptyset$, "attack, monitor" cannot be the NE since $q(x, y)s(x) - c_1 - u(x) < 0$. Meanwhile, "attack, not monitor" is not the NE because $s(x) - q(x, y)s(x) > c_2$ indicates that the defender will selects "monitor".Moreover,since $s(x) - c_1 - u(x) > 0$, "not attack, not monitor" cannot be the NE, either. Combining with the result derived in the beginning that "not attack, monitor" cannot be the NE, it is concluded that no NE exists.

Remark 2. From Theorem 1, the payoffs of the two players are, respectively, expressed as $U_A = s(x^*) - c_1 - u(x^*)$ and $U_D = -s(x^*)$ in (2) in Scenario L.1. It implies that the attacker obtains positive payoff while the defender loses certain security asset in this scenario. On the contrary, the payoffs of two players are, respectively, expressed as $U_A = q(x^*, \underline{y})s(x^*) - u(x^*) - c_1$ and $U_D = -q(x^*, \underline{y})s(x^*) - c_2$ in (3) in Scenario L.1. Similar to (2) in Scenario L.1, the attacker earns positive payoff while the defender loses certain security asset. Nevertheless, different from (2) in Scenario L.1, the defender compensates for part of the loss by executing monitoring action in this scenario as $q < 1$. Thus, the payoff earned by the attacker decreases.

As discussed previously, Scenarios L.2~L.5 cover the possibilities that π_4 is empty while at least two subsets of π_1, π_2, and π_3 are nonempty. Details are given as below.

Scenario L.2. $\pi_1 \neq \emptyset$, $\pi_2 \neq \emptyset$, and $\pi_3 = \pi_4 = \emptyset$.

The following results about the NE for this scenario can be shown.

Theorem 2. *In Scenario L.2, the strategy combination "attack, not monitor" is the unique NE and $x^* = x''$.*

Proof. The subset $\pi_2 \neq \emptyset$ indicates that there exists an x such that the payoff of the attacker $s(x) - u(x) - c_1$ is positive. Thus, the attacker will select the strategy "attack." Besides, the payoff of the defender satisfies $-q(x, y)s(x) - c_2 \leq -s(x)$ for any threshold y, so the defender will select "not monitor." Besides, the optimal attack intensity is given by $x^* = x''$. □

Scenario L.3. $\pi_1 \neq \varnothing$, $\pi_3 \neq \varnothing$, and $\pi_2 = \pi_4 = \varnothing$.

Main results for this scenario are formally stated in the following theorem.

Theorem 3. *In Scenario L.3, the strategy combination "attack, monitor" is the unique NE if and only if* $q(x', \underline{y})s(x') - c_1 - u(x') > 0$. *The optimal attack intensity and detection threshold are* $x^* = x'$ *and* $y^* = \underline{y}$.

Proof. Necessity: if "attack, monitor" is the unique NE, then from (2) and (4), there are $x^* = x'$ and $y^* = \underline{y}$. The payoff of the attacker with x^* and y^* must be positive; thus, $q(x', \underline{y})s(x') - c_1 - u(x') > 0$.

Sufficiency: since $q(x', \underline{y})s(x') - c_1 - u(x') > 0$, the attacker can earn a positive maximum payoff if the defender selects the strategy "monitor" and $y^* = \underline{y}$. Thus, the attacker will select to "attack" and $y^* = \underline{y}$. As $q < 1$ and $(\partial q(x, y)/\partial y) > 0$, there is $s(x') - c_1 - u(x') > q(x', y)s(x') - c_1 - u(x') \geq q(x', \underline{y})s(x') - c_1 - u(x') > 0$ for $y \in [\underline{y}, \overline{y}]$. It follows that $x' \notin \pi_1$ and $(x', y) \in \pi_3$ for $y \in [\underline{y}, \overline{y}]$. From the definition of π_3, it can be concluded that $-q(x', y)s(x') - c_2 \geq -s(x')$ for $y \in [\underline{y}, \overline{y}]$. This indicates that no matter how the threshold is selected, the defender will earn larger payoff when she/he selects the strategy "monitor" rather than "not monitor." Clearly, the defender will select "monitor" and the optimal threshold is set as $y^* = \underline{y}$ from (4). Therefore, the strategy combination "attack, monitor" is the unique NE and $x^* = x'$ and $y^* = \underline{y}$. □

Scenario L.4. $\pi_2 \neq \varnothing$, $\pi_3 \neq \varnothing$, and $\pi_1 = \pi_4 = \varnothing$.

The following conclusions can be drawn for this scenario.

Theorem 4. *In Scenario L.4,*

(1) If and only if $-q(x'', \underline{y})s(x'') - c_2 \leq -s(x'')$, *"attack, not monitor" is the NE and* $x^* = x''$

(2) If and only if $-q(x', \underline{y})s(x') - c_2 > -s(x')$, *"attack, monitor" is the NE and* $x^* = x'$ *and* $y^* = \underline{y}$

Proof

(1) Necessity: under the strategy combination "attack, not monitor", the attacker will select x'' as the optimal attack intensity. If $-q(x'', \underline{y})s(x'') - c_2 > -s(x'')$, the defender will select "monitor" to earn larger payoffs, which is a contradiction to the premise that "attack, not monitor" is the NE. Thus, the necessity is shown.

Sufficiency: from the definitions of π_2 and π_3, the attacker can always earn positive maximum payoff when s/he selects "attack." As $\partial q/\partial y > 0$, there is

$$\begin{aligned} &-q(x'', y)s(x'') - c_2 \\ &\quad \leq -q(x'', \underline{y})s(x'') - c_2 \\ &\quad \leq -s(x''). \end{aligned} \tag{5}$$

This means when the attacker selects $x^* = x''$, the defender never earn larger payoffs than she/he does nothing no matter how the threshold is set. Thus, "attack, not monitor" is the NE and $x^* = x''$. The sufficiency is shown.

(2) Necessity: under the strategy combination "attack, monitor," the defender and attacker will select $\underline{y}$ and x' as the optimal detection threshold and attack intensity from (4) and (2). If $-q(x'', \underline{y})s(x'') - c_2 \leq -s(x'')$, then similar to (5), there is

$$\begin{aligned} &-q(x', y)s(x') - c_2 \\ &\quad \leq -q(x', \underline{y})s(x') - c_2 \\ &\quad \leq -s(x'). \end{aligned} \tag{6}$$

This means the defender never earns larger payoffs than she/he does nothing, which is a contradiction to the premise that "attack, monitor" is the NE. Thus, the necessity is shown.

Sufficiency: the attacker always selects "attack" from the definitions of π_2 and π_3. If the attacker selects $x^* = x'$, since $-q(x', \underline{y})s(x') - c_2 > -s(x')$, the defender will select "monitor" to obtain larger payoffs than "not monitor" and the optimal detection threshold is $\underline{y}$ from (4). Meanwhile, when the defender selects "monitor" and $y^* = \underline{y}$, from (2), the attack will select "attack" and $x^* = x'$ to earn the maximum positive payoff. Thus, the sufficiency is shown.

Based on Theorem 4, the uniqueness of the NE for Scenario L.4 can also be concluded.

Corollary 1. *In Scenario L.4,*

(1) If and only if $-q(x'', \underline{y})s(x'') - c_2 \leq -s(x'')$ *and* $-q(x', \underline{y})s(x') - c_2 \leq -s(x')$, *"attack, not monitor" is the unique NE and* $x^* = x''$

(2) If and only if $-q(x'', \underline{y})s(x'') - c_2 > -s(x'')$ *and* $-q(x', \underline{y})s(x') - c_2 > -s(x')$, *"attack, monitor" is the unique NE and* $x^* = x'$ *and* $y^* = \underline{y}$

Proof. From Theorem 4, "attack, not monitor" and "attack, monitor" are the only two possible NEs. Clearly, "attack, not monitor" is the unique NE if an extra condition holds, i.e., $-q(x'', \underline{y})s(x'') - c_2 \leq -s(x'')$. Then, "attack, monitor" will not be the NE. Similarly, "attack, monitor" is the unique NE if the extra condition $-q(x', \underline{y})s(x') - c_2 > -s(x')$ holds. Then, "attack, not monitor" is not the NE. Therefore, Corollary 1 can be concluded. □

Scenario L.5. $\pi_1 \neq \varnothing$, $\pi_2 \neq \varnothing$, $\pi_3 \neq \varnothing$, and $\pi_4 = \varnothing$.

Different from Scenario L.4, there exists $x \in [\underline{x}, \overline{x}]$ belonging to π_1 such that $s(x) - c_1 - u(x) \leq 0$. Since the attacker can always find an x such that she/he earns a positive payoff, the strategy combination "not attack, not monitor"

cannot be the NE in this scenario. The main results about the NE in this scenario can be formally stated in the following theorem.

Theorem 5. *In Scenario L.5,*

(1) If and only if $-q(x'', \underline{y})s(x'') - c_2 \leq -s(x'')$, *"attack, not monitor" is the NE and* $x^* = x''$

(2) If and only if $q(x', \underline{y})s(x') - c_1 - u(x') > 0$ *and* $-q(x', \underline{y})s(x') - c_2 > -s(x')$, *"attack, monitor" is the NE and* $x^* = x'$ *and* $y^* = \underline{y}$

(3) If and only if $-q(x'', \underline{y})s(x'') - c_2 \leq -s(x'')$ *and* $\{q(x', \underline{y})s(x') - c_1 - u(x') \leq 0$ *or* $-q(x', \underline{y})s(x') - c_2 \leq -s(x')\}$, *"attack, not monitor" is the unique NE and* $x^* = x''$

(4) If and only if $q(x', \underline{y})s(x') - c_1 - u(x') > 0$, $-q(x', \underline{y})s(x') - c_2 > -s(x')$, *and* $-q(x'', \underline{y})s(x'') - c_2 > -s(x'')$, *"attack, monitor" is the unique NE and* $x^* = x'$ *and* $y^* = \underline{y}$

Proof

(1) The proof is similar to that of (1) in Theorem 4 and is omitted here.

(2) Different from Scenario L.4, there exists $x \in [\underline{x}, \overline{x}]$ belonging to π_1 such that

$$q(x, y)s(x) - c_1 - u(x) < s(x) - c_1 - u(x) \leq 0, \quad (7)$$

as $q(x, y) < 1$. Thus, compared to (2) in Theorem 4, an extra condition $q(x', \underline{y})s(x') - c_1 - u(x') > 0$ needs be added to ensure that "attack, monitor" still be the NE. The remaining proof is similar to that of (2) in Theorem 4 and is omitted here.

(3) and (4) By following similar analysis in the proof of Corollary 1, the uniqueness of the NE in this case can also be concluded.

In contrast to previous scenarios, π_4 and at least one subset of π_1, π_2, and π_3 are nonempty in Scenario L.6 as described below. □

Scenario L.6. $\pi_1 \cup \pi_2 \cup \pi_3 \neq \varnothing$, and $\pi_4 \neq \varnothing$.

From (4) in Theorem 1, there is no NE if only $\pi_4 \neq \varnothing$. Besides, if $\pi_4 = \varnothing$ is replaced by $\pi_4 \neq \varnothing$ for (1)–(3) in Scenario L.1 and Scenarios L.2–L.5, the NEs will never belong to π_4. This is because all the strategy combinations driven by x and y within π_4 are inconsistent with the obtained NE in Theorems 1–5. Hence, (x^*, y^*) of the NE for Scenario L.6 will belong to π_1, π_2, or π_3. Moreover, the conditions for the derived NEs in Theorems 1–5 are still necessary. Therefore, to analyze the NE in Scenario L.6, we only need to verify whether the results in Theorems 1–5 are still correct if the subset π_4 is changed to be nonempty. The following conclusions will be shown.

Theorem 6. *In Scenario L.6, the NE for the game as described in Table 1 is derived as follows:*

(1) If $\pi_1 \neq \varnothing$ *and* $\pi_2 = \pi_3 = \varnothing$, *no NE exists*

(2) If $\{\pi_2 \neq \varnothing, \pi_1 = \pi_3 = \varnothing\}$ *or* $\{\pi_1 \neq \varnothing, \pi_2 \neq \varnothing, \pi_3 = \varnothing\}$, *the results in (1) in Theorem 4 hold true and "attack, not monitor" is the unique NE*

(3) If $\{\pi_3 \neq \varnothing, \pi_1 = \pi_2 = \varnothing\}$ *or* $\{\pi_1 \neq \varnothing, \pi_3 \neq \varnothing, \pi_2 = \varnothing\}$, *the results in Theorem 3 hold true*

(4) If $\{\pi_2 \neq \varnothing, \pi_3 \neq \varnothing, \pi_1 = \varnothing\}$ *or* $\{\pi_1 \neq \varnothing, \pi_2 \neq \varnothing, \pi_3 \neq \varnothing\}$, *the results in Theorem 5 hold true*

Proof

(1) As there exists an x such that the payoff of the attacker $s(x) - c_1 - u(x)$ is positive, "not attack, not monitor" is no longer the NE if $\pi_4 = \varnothing$ is replaced by $\pi_4 \neq \varnothing$ for (1) in Scenario L.1, i.e., $\pi_1 \neq \varnothing$, $\pi_2 = \pi_3 = \varnothing$, and $\pi_4 \neq \varnothing$. It can be easily shown that other strategy combinations cannot be the NE either.

(2) If $\pi_4 = \varnothing$ is replaced by $\pi_4 \neq \varnothing$ for (2) in Scenario L.1, there exists feasible x and y such that $-q(x, y)s(x) - c_2 > -s(x)$. Thus, an extra condition $-q(x'', \underline{y})s(x'') - c_2 \leq -s(x'')$ is required with comparison to (2) in Theorem 1 to ensure that "attack, not monitor" still be the NE. If $\pi_4 = \varnothing$ is replaced by $\pi_4 \neq \varnothing$ for Scenario L.2, by following similar analysis in the proofs of Theorem 2 and (1) in Theorem 4, we can show that the results in (1) in Theorem 4 are true.

(3) When $\pi_3 \neq \varnothing$, $\pi_1 = \pi_2 = \varnothing$, and $\pi_4 \neq \varnothing$, there exist feasible x and y such that $q(x, y)s(x) - c_1 - u(x) < 0$. Thus, an extra condition $q(x', \underline{y})s(x') - c_1 - u(x') > 0$ is required with comparison to (3) in Theorem 1 to ensure that "attack, monitor" still be the NE. When $\pi_1 \neq \varnothing$, $\pi_3 \neq \varnothing$, $\pi_2 = \varnothing$, and $\pi_4 \neq \varnothing$, based on the proof of Theorem 3 and the definitions of π_3 and π_4, it can be shown that the results of Theorem 3 are still true.

(4) Firstly, if π_4 is changed to be nonempty in Scenario L.4, x and y belonging to π_4 will have no influence on the results of (1) in Theorem 4. As the results of (1) in Theorem 5 are the same as that of (1) in Theorem 4, (1) in Theorem 5 holds true in this case. Besides, an extra condition $q(x', \underline{y})s(x') - c_1 - u(x') > 0$ is required with comparison to (2) in Theorem 4 to ensure "attack, monitor" be the NE since there exist x and y such that $q(x, y)s(x) - c_1 - u(x) < 0$ from the definition of π_4. Thus, the results in (2) in Theorem 5 are true. The uniqueness of the NE can also be verified from (3) and (4) in Theorem 5. Secondly, if all the subsets are nonempty, i.e., $\pi_1 \neq \varnothing$, $\pi_2 \neq \varnothing$, $\pi_3 \neq \varnothing$, and $\pi_4 \neq \varnothing$, it can be easily shown that the feasible values of x and y belonging to π_4 have no influence on the results of Theorem 5. □

Remark 3. It can be seen from (3) in Theorem 1, Theorem 3, (2) in Theorem 4, and (2) in Theorem 5 that once the defender decides to monitor in (3) in Scenario L.1, Scenario L.3, (2) in Scenario L.4, (2) in Scenario L.5, and (4) and (5) in

Scenario L.6, she/he will always select $\underline{y}$ as the optimal threshold y^*.

Remark 4. In this paper, we assume that the attackers are completely rational, while this assumption may not be satisfied in some scenarios. However, based on our method, we present an optimal defense strategy for the worst case. That is, we can guarantee that the maximum damage in the worst case can be minimized by our method.

4. Simulation Studies

In this section, simulation results are provided to validate the theoretical results as presented above. In A-IDS, a profile is generally selected to cause distinctions between normal and abnormal states. Such a profile is normally described by a random variable in many cases. Here, we assume it follows a Gaussian distribution with zero mean under normal states. Similar assumptions can be seen in many intrusion detection application areas such as network traffic detection and Kalman filtering-based anomaly detection. Let the intensity of the attack be denoted as x. Other parameters in simulation are chosen as $\underline{x} = \underline{y} = 0.1$, $\overline{x} = \overline{y} = 2$, $s = 0.5x$, $u = 0.1x$, and $c_3 = 0.2$. The false alarm rate and missing alarm rate can be expressed by

$$p = \left(\int_{y}^{\infty} e^{-z^2/8} \mathrm{d}z \right) / 2\sqrt{2\pi},$$
$$q = \left(\int_{-\infty}^{y} e^{-(z-x)^2/8} \mathrm{d}z \right) / \sqrt{2\pi}, \tag{8}$$

respectively. Parameters c_1 and c_2 are used to represent the costs of the attacker and the defender, respectively.

Case 1. We first select $c_1 \in [0, 0.2]$ and $c_2 = 0.2$. Then, it can be calculated by (1) that

(a) If $c_1 \in [0, 0.04]$, there are $\pi_2 \neq \varnothing$, $\pi_3 \neq \varnothing$, and $\pi_1 = \pi_4 = \varnothing$, which corresponds to Scenario L.4

(b) If $c_1 \in [0.04, 0.08]$, there are $\pi_1 \neq \varnothing$, $\pi_2 \neq \varnothing$, $\pi_3 \neq \varnothing$, and $\pi_4 = \varnothing$, which corresponds to Scenario L.5

(c) If $c_1 \in [0.08, 0.2]$, all the four subsets are nonempty, which corresponds to Scenario L.6

Then, it can be checked whether the inequality conditions in Theorems 4 and 5 and (4) in Theorem 6 are satisfied for the above three scenarios, as given in Table 3. 'IC 4.1', 'IC 4.2', 'IC 5.1', and 'IC 5.2' refer to the inequality conditions in (1) and (2) in Theorem 4 and (1) and (2) Theorem 5, respectively. It is worth noting that the inequality conditions in (4) in Theorem 6 are the same as those in Theorem 5. From the theoretical analysis given in Section 2, the following conclusions on the NEs can be drawn:

(a) Based on (2) in Theorem 4, "attack, monitor" is the unique NE if $c_1 \in [0, 0.04]$ and $c_2 = 0.2$.

(b) Based on (2) in Theorem 5, "attack, monitor" is the unique NE if $c_1 \in [0.04, 0.08]$ and $c_2 = 0.2$.

(c) Based on (4) in Theorem 6 and (2) in Theorem 5, "attack, monitor" is still the unique NE if $c_1 \in [0.08, 0.2]$ and $c_2 = 0.2$. However, no NE exists if $c_1 \in (0.12, 0.2]$, $c_2 = 0.2$. This result can be verified by observing the payoff of the attacker (U_A) with respect to c_1, as shown in Figure 1. U_A decreases as c_1 increases. Besides, U_A will approach zero when c_1 tends to 0.12, which indicates that the NE is broken.

Case 2. In this case, we fix c_1 as $c_1 = 0.1$, while let c_2 vary within the interval $[0, 0.2]$. It can be calculated that

(a) If $c_2 \in [0, 0.04]$, there are $\pi_1 \neq \varnothing$, $\pi_3 \neq \varnothing$, $\pi_4 \neq \varnothing$, and $\pi_2 = \varnothing$, which corresponds to Scenario L.6

(b) If $c_2 \in [0.04, 0.2]$, all the four subsets are nonempty, which also corresponds to Scenario L.6

Similarly, Table 4 is given to show whether the inequality conditions in Theorems 3 and 5 are satisfied, where 'IC 3' refers to the inequality condition in Theorem 3. Then, the following conclusions on the NEs can be drawn:

(a) Based on Theorem 3 and (3) in Theorem 6, "attack, monitor" is the unique NE if $c_1 = 0.1$ and $c_2 \in [0, 0.04]$

(b) Based on (2) in Theorem 5 and (4) in Theorem 6, "attack, monitor" is the unique NE if $c_1 = 0.1$ and $c_2 \in (0.04, 0.2]$

Therefore, "attack, monitor" is always the unique NE if $c_1 = 0.1$, $c_2 \in [0, 0.2]$. Besides, from Theorem 3 and (2) Theorem 5, it can be calculated that the payoff of the attacker (U_A) is equal to 0.024 if $c_2 \in [0, 0.2]$. It indicates that the attacker has the motivation to launch the attack. The performance of the defender's payoff (U_D) with respect to c_2 is shown in Figure 2. Clearly, the defender loses some security asset as $U_D < 0$. Moreover, the lost security asset will increase as the defense cost c_2 increases.

At last, we make some comparisons with the existing methods in [7–15], where attack intensity and detection threshold are scarcely considered and majority of them assume that the false and missing alarm rates, and the game model of detection problem can be modelled as Table 5.

It can be seen that, without considering the attack intensity and detection threshold, the payoffs of the game model will be reduced to be constant and the Nash equilibrium analysis can be easily done. From the definition of the Nash equilibrium, it can be calculated that if $q + c_2 > 1$, (Attack, Monitor) will be the unique NE. Though the existing analysis methods in [7–15] can determine the optimal action strategies, while our results can further determine the optimal explicit attack intensity and detection threshold, different results can be obtained. First, the existing work considers only the strategy do or not do; thus, the one-stage game model, as expressed in Table 3, is established to help analyze the optimal actions, while we further consider the attack intensity and detection threshold in the game model, as these two parameters are two key strategies used for the defender and the attacker. Moreover, we establish a more detailed two-stage game model to consider both the action do or not do and the attack intensity

Table 3: The results showing whether the inequality conditions in Theorems 4 and 5 and (4) in Theorem 6 are satisfied when $c_1 \in [0, 0.2]$ and $c_2 = 0.2$.

$c_1 \in [0, 0.04]$ Scenario L.4		$c_1 \in [0.04, 0.08]$ Scenario L.5		$c_1 \in [0.08, 0.12]$ Scenario L.6		$c_1 \in (0.12, 0.2]$ Scenario L.6	
IC 4.1	×	IC 5.1	×	IC 5.1	×	IC 5.1	×
IC 4.2	√	IC 5.2	√	IC 5.2	√	IC 5.2	×

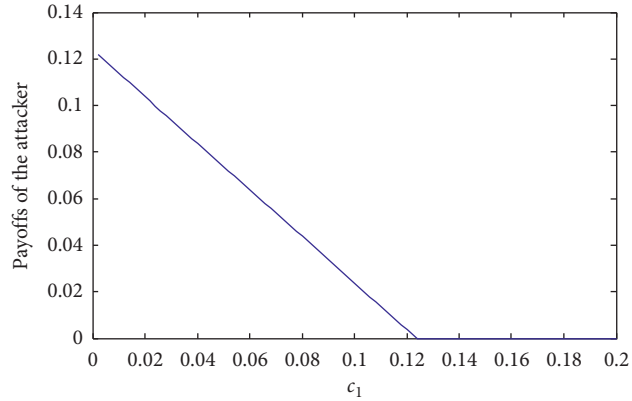


Figure 1: Payoff of the attacker U_A with respect to c_1 if c_2 is fixed as $c_2 = 0.2$.

Table 4: The results about the inequality conditions in Theorems 3 and 5 with $c_1 = 0.1$ and $c_2 \in [0, 0.2]$.

$c_2 \in [0, 0.04]$ Scenario L.4		$c_2 \in (0.04, 0.2]$ Scenario L.5	
IC 3	√	IC 5.1	×
		IC 5.2	√

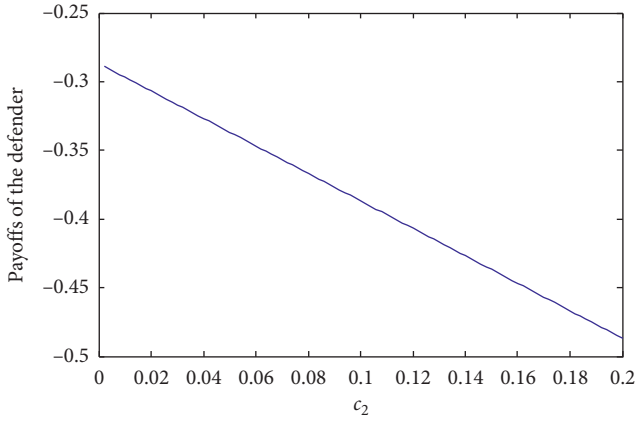


Figure 2: Payoff of the defender U_D with respect to c_2 if c_1 is fixed as $c_1 = 0.1$.

Table 5: Strategic form of the game in existing work.

	Monitor	Not monitor
Attack	$U_A = qW - c_1W$ $U_D = -qW - c_2W$	$U_A = W - c_1W$ $U_D = -W$
Not attack	$U_A = 0$ $U_D = -pc_3W - c_2W$	$U_A = 0$ $U_D = 0$

and detection threshold. Based on the experimental results, we can see that the attack intensity and detection threshold play an important role in the determination of the Nash equilibrium. Intuitively, for the game in Table 3, the NE are completely determined by the parameter x and y; however, this conclusion seems not to make sense as the false alarm rate and other parameters have no any effect on the Nash equilibrium. Alternatively, for our game model, we can see that all parameters will jointly determine the Nash equilibriumthus, our analysis results are more realistic. In practical, the false and missing alarm rates are not constant, as the attacks are always dynamically changing. In A-IDS methods, the false and missing alarm rates are commonly determined by the attack intensity and detection threshold. Our method just considers this real scenario and establishes a more explicit game model, based on which the optimal strategies are completely determined.

5. Conclusion

For anomaly-based intrusion detection system, we present a game theoretical analysis method to provide the optimal strategies. We first establish a more realistic game model by considering the attack intensity and detection threshold as two strategies for the players. The necessary and sufficient conditions, for which strategies are the Nash equilibriums, are presented. Simulation studies are provided to validate our theoretical results. The results provide a new method to determine the detection threshold in the security defense. In the future, some more research work could be considered, for example, the game theoretical analysis method for

specific scenarios such as Internet of Things and DoS/DDoS attacks. Besides, dynamic game analysis is also an interesting topic for dynamic security confrontation process, for example, Stackelberg game analysis can be adopted to solve the sequential problem of the attack and defense actions.

Data Availability

The manuscripts of game theory algorithm in this article are from the databases of Cambridge University and Columbia University. Copies of these data can be obtained from https://dl.acm.org/doi/book/10.5555/1951874 and https://doi.org/10.1016/j.ins.2018.04.051.

Conflicts of Interest

The authors declared that they have no conflicts of interest.

Acknowledgments

This work was supported by the Basic Scientific Research Projects of National Defense Science, Technology and Industry Technology under Grant no. JSZL2017601C-1 and in part by the National Natural Science Foundation of China under Grant nos. 61897069 and 61831003, National Key Research and Development Program of China under Grant no. 2017YFB0801903, and National Key Program for Basic Research of China under Grant no. 2017-JSJQ-ZD-043.

References

[1] H. J. Liao, C. H. R. Lin, Y. C. Lin, and K. Y. Tung, "Intrusion detection system: a comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[2] P. G. Teodoro, J. D. Verdejo, G. M. Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[3] M. Manshaei, Q. Zhu, T. Alpcan, T. Basar, and J. P. Hubaux, "Game theory meets network security and privacy," *ACM Computing Surveys*, vol. 45, no. 3, pp. 1–39, 2013.

[4] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu, "A survey of game theory as applied to network security," in *Proceedings of the 43rd Hawaii International Conference on System Sciences*, IEEE, Honolulu, HI, USA, January 2010.

[5] X. Liang and Y. Xiao, "Game theory for network security," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 472–486, 2013.

[6] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: a statistical anomaly approach," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 76–82, 2002.

[7] T. Alpcan and T. Basar, *Network Security: A Decision and Game-Theoretic Approach*, Cambridge University Press, Cambridge, UK, 2011.

[8] L. Chen and J. Leneutre, "A game theoretical framework on intrusion detection in heterogeneous networks," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 2, pp. 165–178, 2009.

[9] Z. Ismail and J. Leneutre, "A game theoretical analysis of data confidentiality attacks on smart-grid AMI," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 7, pp. 1486–1499, 2014.

[10] Q. Zhu, C. Fung, R. Boutaba, and T. Basar, "GUIDEX: A game-theoretic incentive-based mechanism for intrusion detection networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 11, pp. 2220–2230, 2012.

[11] H. Wu, W. Wang, C. Wen, and Z. Li, "Game theoretical security detection strategy for networked systems," *Information Sciences*, vol. 453, pp. 346–363, 2018.

[12] Y. Liu, C. Comaniciu, and H. Man, "A Bayesian game approach for intrusion detection in wireless ad hoc networks," *ACM International Conference Proceeding Series*, vol. 199, 2006.

[13] K. C. Nguyen, T. Alpcan, and T. Basar, "Security games with incomplete information," in *Proceedings of the of 2009 IEEE International Conference on Communications*, IEEE, Dresden, Germany, June 2009.

[14] W. Wang, M. Chatterjee, and K. Kwiat, "Attacker detection game in wireless networks with channel uncertainty," in *Proceedings of the 2010 IEEE International Conference on Communications*, IEEE, Cape Town, South Africa, May 2010.

[15] Y. E. Sagduyu, R. Berry, and A. Ephremides, "MAC games for distributed wireless network security with incomplete information of selfish and malicious user types," in *Proceedings of the 2009 International Conference on Game Theory for Networks*, IEEE, Istanbul, Turkey, May 2009.

[16] A. Bradai and H. Afifi, "Game theoretic framework for reputation-based distributed intrusion detection," in *Proceedings of the 2013 International Conference on Social Computing*, IEEE, Alexandria, VA, USA, September 2013.

[17] W. Wang, M. Chatterjee, K. Kwiat, and Q. Li, "A game theoretic approach to detect and co-exist with malicious nodes in wireless networks," *Computer Networks*, vol. 71, pp. 63–83, 2014.

[18] W. Yu and K. J. R. Liu, "Secure cooperation in autonomous mobile ad-hoc networks under noise and imperfect monitoring: a game-theoretic approach," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 2, pp. 317–330, 2008.

[19] F. Li, Y. Yang, and J. Wu, "Attack and flee: game-theory-based analysis on interactions among nodes in MANETs," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 40, no. 3, pp. 612–622, 2010.

[20] L. Xiao, Y. Chen, W. S. Lin, and K. J. R. Liu, "Indirect reciprocity security game for large-scale wireless networks," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1368–1380, 2012.

[21] H. Moosavi and F. M. Bui, "A game-theoretic framework for robust optimal intrusion detection in wireless sensor networks," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 9, pp. 1367–1379, 2014.

[22] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley, "RRE: A game-theoretic intrusion response and recovery engine," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 395–406, 2014.

[23] H. Wu and W. Wang, "A game theory based collaborative security detection method for Internet of Things systems," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 6, pp. 1432–1445, 2018.

[24] H. Wu and Z. Wang, "Multi-source fusion-based security detection method for heterogeneous networks," *Computers & Security*, vol. 74, pp. 55–70, 2018.

[25] R. Jin, X. He, and H. Dai, "On the security-privacy tradeoff in collaborative security: a quantitative information flow game perspective," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 12, pp. 3273–3286, 2019.
[26] H. Zhang, L. Jiang, S. Huang, J. Wang, and Y. Zhang, "Attack-defense differential game model for network defense strategy selection," *IEEE Access*, vol. 7, pp. 50618–50629, 2018.
[27] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8890306, 11 pages
https://doi.org/10.1155/2020/8890306

Research Article

DL-IDS: Extracting Features Using CNN-LSTM Hybrid Network for Intrusion Detection System

Pengfei Sun,[1] Pengju Liu,[2] Qi Li,[3] Chenxi Liu,[2] Xiangling Lu,[2] Ruochen Hao,[2] and Jinpeng Chen [1]

[1]*School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[3]*Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China*

Correspondence should be addressed to Jinpeng Chen; chenjinpeng@nlsde.buaa.edu.cn

Received 6 April 2020; Revised 17 July 2020; Accepted 25 July 2020; Published 28 August 2020

Academic Editor: Huaming Wu

Many studies utilized machine learning schemes to improve network intrusion detection systems recently. Most of the research is based on manually extracted features, but this approach not only requires a lot of labor costs but also loses a lot of information in the original data, resulting in low judgment accuracy and cannot be deployed in actual situations. This paper develops a DL-IDS (deep learning-based intrusion detection system), which uses the hybrid network of Convolutional Neural Network (CNN) and Long Short-Term Memory Network (LSTM) to extract the spatial and temporal features of network traffic data and to provide a better intrusion detection system. To reduce the influence of an unbalanced number of samples of different attack types in model training samples on model performance, DL-IDS used a category weight optimization method to improve the robustness. Finally, DL-IDS is tested on CICIDS2017, a reliable intrusion detection dataset that covers all the common, updated intrusions and cyberattacks. In the multiclassification test, DL-IDS reached 98.67% in overall accuracy, and the accuracy of each attack type was above 99.50%.

1. Introduction

1.1. Background. In recent years, with the rapid development of emerging communications and information technologies such as 5G communications, mobile Internet, Internet of Things, "cloud computing," and big data, network security has become increasingly important. As an important research content of network security, intrusion detection has been paid attention by experts and scholars. Problems that are common under traditional anomaly-based detection methods include the inaccurate feature extraction of network traffic and difficulty in building attack detection models, which leads to high false alarm rate when judging attack traffic. It is difficult for network security personnel to find unknown threats, which makes the defense inherently passive. In other words, traditional methods are no longer applicable to today's Internet as per its massive data scale.

In recent years, many scholars have explored how to use artificial intelligence (AI) to detect and analyze network traffic for intrusion detection and defense systems. Hassan et al. [1] proposed an ensemble-learning model based on the combination of a random subspace (RS) learning method with random tree (RT), which detected cyberattacks of SCADA by using the network traffic from the SCADA-based IoT platform. Khan and Gumaei [2] compared the most popular machine learning methods for intrusion detection in terms of accuracy, precision, recall, and training time cost. Alqahtani et al. [3] proposed GXGBoost model to detect intrusion attacks based on a genetic algorithm and an extreme gradient boosting (XGBoost) classifier. Derhab et al. [4] proposed a security architecture that integrates the Blockchain and the software-defined network (SDN) technologies, which focuses on the security of commands in industrial IoT against forged commands and misrouting of

commands. The current mainstream methods are intrusion detection systems based on machine learning (ML) or deep learning (DL). Among them, the ML-based system mainly classifies and detects network traffic by analyzing the manually extracted features of network traffic, while the DL-based system can not only analyze the manually extracted features but also automatically extract the features from the original traffic. Therefore, DL-based systems can circumvent the manual feature extraction problem and enhance the detection accuracy compared to general ML-based systems.

To achieve higher accuracy, DL-based intrusion detection methods require a large amount of data for training, especially different types of attack traffic data. In the actual environment and the existing datasets [KDD99, NSL-KDD, and CICIDS2017], the attack traffic is always less compared with normal traffic. Moreover, because some types of attack traffic are difficult to capture and simulate, the amount of data available for model training is particularly small. These problems greatly restrict the accuracy of the DL-based method, making it difficult to judge certain types of attacks.

1.2. Key Contributions. This paper proposes a DL-based intrusion detection system, DL-IDS, which uses the hybrid network of Convolutional Neural Network (CNN) and Long Short-Term Memory Network (LSTM) to extract the temporal and spatial features of network traffic data to improve the accuracy of intrusion detection. In the model training phase, DL-IDS uses category weights to optimize the model. This method reduces the effect of the number of unbalanced samples of several attack types in model training samples on model performance and improves the robustness of training and prediction. Finally, we test DL-IDS to classify multiple types of network traffic on the CICIDS2017 dataset and compare it with the CNN-only model, LSTM-only model, and other machine learning models because CICIDS2017 is a recent original network traffic dataset simulating real situations. The results show that DL-IDS reached 98.67% in overall accuracy, and the accuracy of each attack type was above 99.50%, which achieved the best results in all models.

1.3. Paper Organization. The remainder of this paper is organized as follows. Section 2 discusses the classification of abnormal traffic as per previously published studies. Section 3 describes in detail the datasets and data preprocessing methods we used in this study. The classifier structure and classification methods used for traffic classification under the proposed model are described in Section 4. Section 5 presents the results of our model evaluation with various hyperparameters. Section 6 summarizes the paper and discusses potential future development trends.

2. Related Work

With the continual expansion of the Internet, network security has become a problem that cannot be ignored. Malicious network behaviors such as DDos and brute force attacks tend to be "mixed into" malicious traffic. Security researchers seek to effectively analyze the malicious traffic in a given network so as to identify potential attacks and quickly stop them [5–8].

2.1. Traditional Intrusion Detection System. Traditional methods of intrusion detection mainly include statistical analysis methods [9], threshold analysis methods [10], and signature analysis methods [11]. These methods do reveal malicious traffic behavior; however, they require security researchers to input data related to their personal experience; to this effect, their various rules and set parameters are very inefficient. Said "experience" is also only a summary of the malicious traffic behavior found in the past and is typically difficult to quantify, so these methods cannot be readily adapted to the huge amount of network data and volatile network attacks of today's Internet.

2.2. Intrusion Detection System Based on ML. Advancements in machine learning have produced models that effectively classify and cluster traffic for the purposes of network security. Early researchers attempted simple machine learning algorithms for classification-clustering problems in other fields, such as the k-Nearest Neighbor (KNN) [12], support vector machine (SVM) [13], and self-organizing maps (SOM) [14], with good results on KDD99, NSL-KDD, DARPA, and other datasets. These datasets are out of date, unfortunately, and contain not only normal data but also attack data that are overly simple. It is difficult to use these datasets to simulate today's highly complex network environment. It is also difficult to achieve the expected effect using these algorithms to analyze malicious traffic in a relatively new dataset, as evidenced by our work in this study.

2.3. Intrusion Detection System Based on Deep Neural Network. The success of machine learning algorithms generally depends on data representation [15]. Representation learning, also called feature learning, is a technique in deep neural network, which can be used to learn the explanatory factors of variation behind the data. Ma et al. combine spectral clustering and deep neural network algorithms to detect intrusion behaviors [16]. Niyaz et al. used deep belief networks for developing an efficient and flexible intrusion detection system [17]. But these research methods construct their models to learn representations from manually designed traffic features, not taking full advantage of the ability of deep neural networks. Eesa et al. showed that higher detection rate and accuracy rate with lower false alarm rate can be obtained by using improved traffic feature set [18]. Learning features directly from traffic raw data should be feasible, such as in the fields of computer vision and natural language processing [19].

Two most widely used deep neural network models are CNN and RNN. The CNN uses original data as the direct input to the network, does not necessitate feature extraction or image reconstruction, has relatively few parameters, and requires relatively little data in process. CNNs have been proven to be highly effective in the field of image recognition

[20]. For certain network traffic of protocols, CNNs can perform well through fast training. Fan and Ling-zhi [21] extracted very accurate features by using a multilayer CNN, wherein the convolution layer connects to the sampling layer below; this model outperformed classical detection algorithms (e.g., SVM) on the KDD99 dataset. However, the CNN can only analyze a single input package—it cannot analyze timing information in a given traffic scenario. In reality, a single packet in an attack traffic scenario is normal data. When a large number of packets are sent at the same time or in a short period, this packet becomes malicious traffic. The CNN does not apply in this situation, which in practice may lead to a large number of missed alerts.

The recurrent neural network (RNN) is also often used to analyze sequential information. LSTM, a branch of RNNs, performs well in sequence information analysis applications such as natural language processing. Kim et al. [22] compared the LSTM-RNN network against Generalized Regression Neural Network (GRNN), Product-based Neural Network (PNN), *k*-Nearest Neighbor (KNN), SVM, Bayesian, and other algorithms on the KDD99 dataset to find that it was superior in every aspect they tested. The LSTM network alone, however, centers on a direct relationship between sequences rather than the analysis of a single packet, so it cannot readily replace the CNN in this regard.

Wu and Guo [23] proposed a hierarchical CNN + RNN neural network and carried out experiments on NSL-KDD and UNSW-NB15 datasets. Hsu et al. [24] and Ahsan and Nygard [25] used another CNN + LSTM model to perform multiclassification experiments on the NSL-KDD dataset. Hassan et al. [26] proposed a hybrid deep learning model to detect network intrusions based on CNN network and a weight-dropped, long short-term memory (WDLSTM) network. This paper mainly conducted experiments on UNSW-NB15 dataset. However, these studies are still based on extracted features in advance.

Abdulhammed et al. [27] used Autoencoder and Principle Component Analysis to reduce the CICIDS2017 dataset's feature dimensions. The resulting low-dimensional features from both techniques are then used to build various classifiers to detect malicious attacks. Musafer et al. [28] proposed a novel architecture of IDS based on advanced Sparse Autoencoder and Random Forest to classify the patterns of the normal packets from those of the network attacks and got good results.

In this study, we adopted a malicious traffic analysis method based on CNN and LSTM to extract and analyze network traffic information of network raw dataset from both spatial and temporal dimensions. We conducted training and testing based on the CICIDS2017 dataset that well simulates the real network environment. We ran a series of experiments to show that the proposed model facilitates very effective malicious flow analysis.

3. Datasets and Preprocessing

3.1. Dataset. The IDS is the most important defense tool against complex and large-scale network attacks, but the lack of available public dataset yet hinders its further development. Many researchers have used private data within a single company or conducted manual data collection to test IDS applications, which affects the credibility of their results to some extent. Public datasets such as KDD99 and NSL-KDD [29] are comprised of data encompassing manually selected stream characteristics rather than original network traffic. The timing of the data collection is also outdated compared to modern attack methods.

In this study, in an effort to best reflect real traffic scenarios in real networks as well as newer means of attack, we chose the CICIDS2017 dataset (Canadian Institute for Cybersecurity) [30] which contains benign traffic and up-to-date common attack traffic representative of a real network environment. This dataset constructs the abstract behavior of 25 users based on HTTP, HTTPS, FTP, SSH, and e-mail protocols to accurately simulate a real network environment. The data capture period was from 9 a.m. on July 3, 2017, to 5 p.m. on July 7, 2017; a total of 51.1 g data flow was generated over this five-day period. The attack traffic collected includes eight types of attack: FTP-Patator, SSH-Patator, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS. As shown in Table 1, the attacks were carried out on Tuesday, Wednesday, Thursday, and Friday morning and afternoon. Normal traffic was generated throughout the day on Monday and during the nonaggressive period from Tuesday to Friday. The data type for this dataset is a pcap file.

After acquiring the dataset, we analyzed the original data and selected seven types of data for subsequent assessment according to the amount of data and its noise rate. They are Normal, FTP-Patator, SSH-Patator, DoS, Heartbleed, Infiltration, and PortScan.

3.2. Network Traffic Segmentation Method. The format of the CICIDS2017 dataset is one pcap file per day. These pcap files contain a great deal of information, which is not conducive to training the machine. Therefore, the primary task of traffic classification based on machine learning is to divide continuous pcap files into several discrete units according to a certain granularity. There are six ways to slice network traffic: by TCP, by connection, by network flow, by session, by service class, and by host. When the original traffic data is segmented according to different methods, it splits into quite different forms, so the selected network traffic segmentation method markedly influences the subsequent analysis.

We adopted a session sharding method in this study. A session is any packet that consists of a bidirectional flow, that is, any packet that has the same quad (source IP, source port, destination IP, destination port, and transport layer protocol) and interchangeable source and destination addresses and ports.

3.3. Data Preprocessing. Data preprocessing begins with the original flow, namely, the data in pcap format, for formatting the model input data. The CICIDS2017 dataset provides an original dataset in pcap format and a CSV file detailing some of the traffic. To transform the original data into the model input format, we conducted time division, traffic

Table 1: CICIDS2017 dataset.

Date	Type	Size
Monday	Normal	11.0GB
Tuesday	Normal + Force + SFTP + SSH	11.0GB
Wednesday	Normal + Dos + Heartbleed Attacks	13GB
Thursday	Normal + XSS + Web Attack + Infiltration	7.8GB
Friday	Normal + Botnet + PortScan + DDoS	8.3GB

segmentation, PKL file generation, PKL file labeling, matrix generation, and one_hot encoding. A flow chart of this process is given in Figure 1.

Step 1 (time division). Time division refers to intercepting the pcap file of the corresponding period from the original pcap file according to the attack time and type [30]. The input format is, again, a pcap file; the output format is still a pcap file. The time periods corresponding to the specific type and the size of the file are shown in Table 2.

Step 2 (traffic segmentation). Traffic segmentation refers to dividing the pcap file obtained in Step 1 into corresponding sessions by sharding according to the IP of attack host and victim host corresponding to each time period [31]. The specific process is shown in Figure 2. This step involves shredding the pcap file of Step 1 into the corresponding flow using pkt2flow [31], which can split the pcap package into the flow format (i.e., the original pcap package is divided into different pcap packages according to the flow with different five-tuples). Next, the pcap package is merged under the premise that the source and destination are interchangeable. Finally, the pcap package of Step 1 is divided into sessions.

Step 3 (generate the PKL file) and Step 4 (tag the PKL file). As shown in Table 1, the pcap file is still large in size after extraction; this creates a serious challenge for data reading in the model. To accelerate the data reading process, we packaged the traffic using the pickle tool in Python. We use PortScan type traffic as an example of the packaging process here. In this class, many sessions are generated after Step 2, each of which is saved in a pcap file. We generated a label of the corresponding attack type for each session. Each session contains several data flows, and each data flow contains an n_1 packet. We then saved the n_2 sessions in a PKL file to speed up the process of reading the data. n_1 can be changed as needed; according to the experimental results, we finally selected the best value, that is, $n_1 = 8$. The value of n_2 can be calculated by formula (1). In this case, we packaged each type of sessions into a PKL file. The structure of the entire PKL file is shown in Figure 3.

$$n_2 = \frac{\text{total number of packets of this type}}{n_1}. \quad (1)$$

Step 5 (matrix generation). The input of the model must have a fixed length, so the next step is to unify the length of each session. The difference between each attack is mainly in the header, so we dealt with the packet according to the uniform length of PACKET_LEN bytes; that is, if the packet length was greater than PACKET_LEN, then bytes were intercepted, and if the packet length was less than PACKET_LEN, bytes were filled with -1. Each session is then divided into a matrix MAX_PACKET_NUM_PER_SESSION∗PACKET_LEN. According to the results of our experiment, we finally chose MAX_PACKET_NUM_PER_SESSION as 8 and PACKET_LEN as 40.

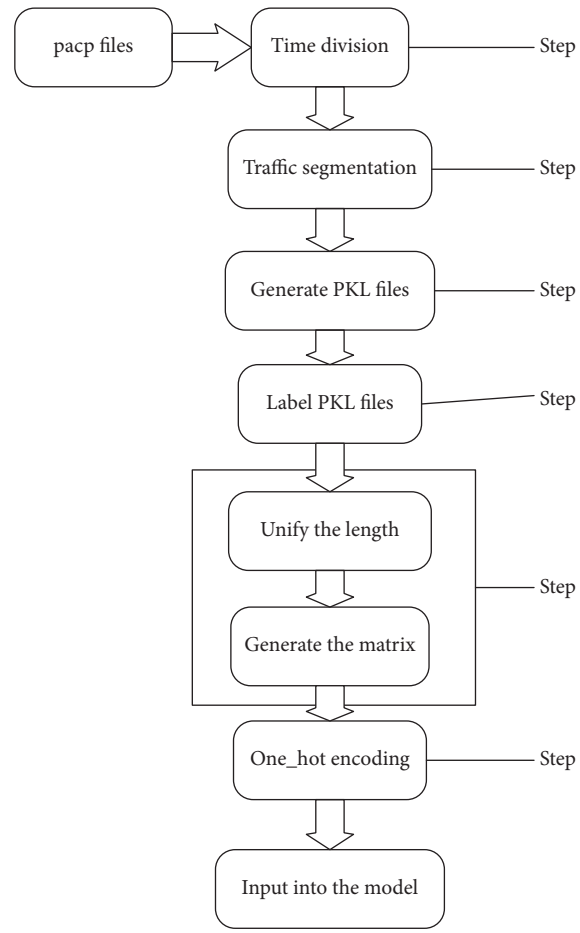


Figure 1: Data preprocessing process.

Table 2: Distribution of network traffic periods in CICIDS2017 dataset.

Attack	Time	Size
Normal	Monday	11 GB
FTP-Patator	Tuesday (9 : 20–10 : 20)	12 MB
SSH-Patator	Tuesday (14 : 00–15 : 00)	25 MB
DoS	Wednesday (9 : 47–11 : 23)	2.3 GB
Heartbleed	Wednesday (15 : 12–15 : 32)	79 MB
Infiltration	Thursday (14 : 19–15 : 45)	21 MB
PortScan	Friday (13 : 55–14 : 35)	31 MB

Step 6 (one_hot encoding). To effectively learn and classify the model, the data from Step 5 are processed by one_hot encoding to convert qualitative features into quantitative features:

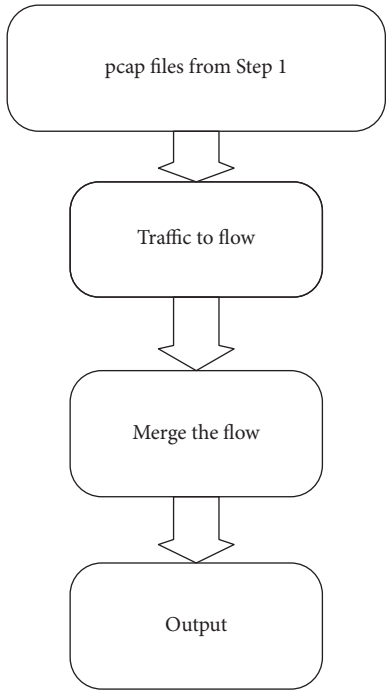


Figure 2: Traffic segmentation process.

$$ohe_i(B, num) = \begin{cases} num, & B = i, \\ 0, & B \neq i, \end{cases}$$
$$OHE_j\left(ohe_{j1}, \ldots, ohe_{j(\text{Length})}\right) = ohe_1 \oplus \ldots \oplus ohe_{\text{Length}},$$
$$\text{Input}_k = \begin{pmatrix} OHE_1 \\ OHE_2 \\ \ldots \\ OHE_N \end{pmatrix}, \tag{2}$$

where B is a byte in the data packet; num is the number used for encoding in one_hot encoding. In the model implementation, $num = 1$, ohe_i is a bit in the one_hot encoding of a byte, $\oplus$ is the series notation, and OHE_j is the one_hot encoding of a byte.

4. DL-IDS Architecture

This section introduces the traffic classifier we built into DL-IDS, which uses a combination of CNN and LSTM to learn and classify traffic packets in both time and space. According to the different characteristics of different types of traffic, we also used the weight of classes to improve the stability of the model.

The overall architecture of the classifier is shown in Figure 4. The classifier is composed of CNN and LSTM. The CNN section is composed of an input and embedded layer, convolution layer 1, pooling layer 2, convolution layer 3, pooling layer 4, and full connection layer 5. Upon receiving a preprocessed PKL file, the CNN section processes it and returns a high-dimensional package vector to the LSTM section. The LSTM section is composed of the LSTM layer 6, LSTM layer 7, full connection layer 8, and the OUTPUT layer. It can process a series of high-dimensional package vectors and output a vector that represents the probability that the session belongs to each class. The Softmax layer outputs the final result of the classification according to the vector of probability.

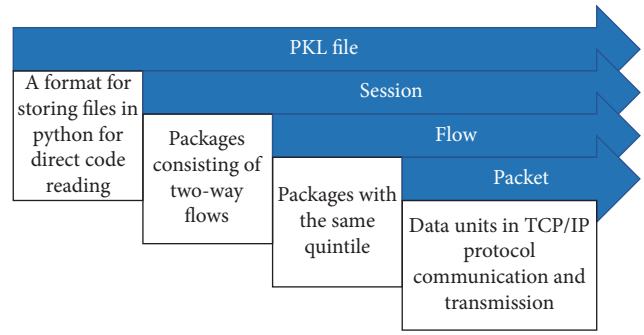


Figure 3: Structure of entire PKL file.

4.1. CNN in DL-IDS. We converted the data packets obtained from the preprocessed data into a traffic image. The so-called traffic image is a combination of all or part of the bit values of a network traffic packet into a two-dimensional matrix. The data in the network traffic packet is composed of bytes. The value range of the bytes is 0–255, which is the same as the value range of the bytes in images. We took the x byte in the header of a packet and the y byte in the payload and composed them into a traffic image for subsequent processing, as discussed below.

As mentioned above, the CNN section is composed of input and embedded layers, convolution layer 1, pooling layer 2, convolution layer 3, pooling layer 4, and full connection layer 5. We combined convolution layer 1 and pooling layer 2 into Combination I and convolution layer 3 and pooling layer 4 into Combination II. Each Combination allows for the analysis of input layer characteristics from different prospective. In Combination I, a convolution layer with a small convolution kernel is used to extract local features of traffic image details (e.g., IP and Port). Clear-cut features and stable results can be obtained in the pooling layer. In Combination II, a large convolution kernel is used to analyze the relationship between two bits that are far apart, such as information in the traffic payload.

After preprocessing and one_hot coding, the network traffic constitutes the input vector of the input layer. In the input layer, length information is intercepted from the ith packet $Pkg_i = (B_1, B_2, \ldots, B_{\text{Length}})$ followed by synthesis S of n Pkgs information set $S = (Pkg_1, Pkg_2, \ldots, Pkg_n)$.

Formula 3 shows a convolution layer, where f is the side length of the convolution kernel. In the two convolution layers, $f = 7$ and $f = 5$. s is stride, p is padding, b is bias, w is weight, c is channel, l is layer, L_l is the size of Z_l, and $Z(i, j)$ is the pixel of the corresponding feature map. Additionally $(i, j) \in \{0, 1, 2, \ldots, L_{l+1}\} L_{l+1} = ((L_l + 2p - f)/s) + 1$.

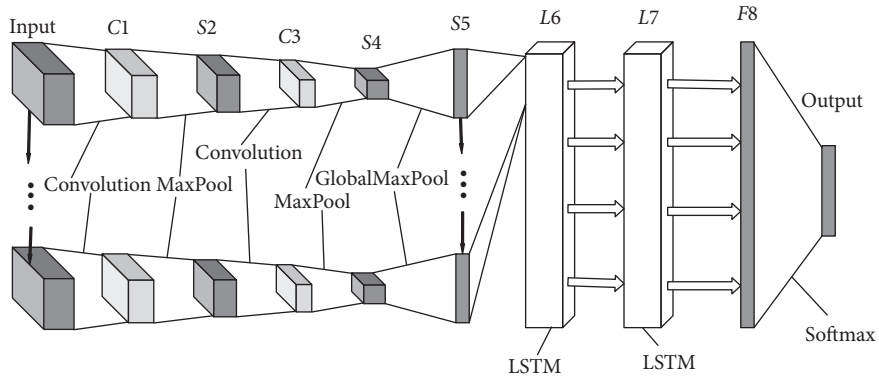


Figure 4: Architecture of DL-IDS.

$$Z^{l+1}(i,j) = \sum_{k=1}^{c} \sum_{x=1}^{f} \sum_{y=1}^{f} \left[Z_k^1 (s*i+x, s*j+y) * w_k^{l+1}(x,y) \right] + b. \tag{3}$$

The convolution layer contains an activation function (formula 4) that assists in the expression of complex features. K is the number of channels in the characteristic graph and A represents the output vector of the Z vector through the activation function. We used sigmoid and ReLU, respectively, after two convolution layers.

$$A_{i,j,k}^1 = f\left(Z_{i,j,k}^1\right). \tag{4}$$

After feature extraction in the convolution layer, the output image is transferred to the pooling layer for feature selection and information filtering. The pooling layer contains a preset pooling function that replaces the result of a single point in the feature map with the feature graph statistics of its adjacent region. The pooling layer is calculated by formula 5, where p is the prespecified parameter. We applied the maximum pooling in this study, that is, $p \longrightarrow \infty$.

$$A_k^l(i,j) = \left[\sum_{x=1}^{f} \sum_{y=1}^{f} A_k^l (s*i+x, s*j+y)^p \right]^{1/p}. \tag{5}$$

We also used a back-propagation algorithm to adjust the model parameters. In the weight adjustment algorithm (formula 6), δ is delta error of loss function to the layer, and α is the learning rate.

$$w^l = w^l - \alpha \sum \delta^l * A^{l-1}. \tag{6}$$

We used the categorical cross-entropy algorithm in the loss function. In order to reduce training time and enhance the gradient descent accuracy, we used the RmsProp optimization function to adjust the learning rate.

After two convolution and pooling operations, we extracted the entire traffic image into a smaller feature block, which represents the feature information of the whole traffic packet. The block can then be fed into the RNN system as an input to the RNN layer.

4.2. LSTM in DL-IDS. Normal network communications and network attacks both are carried out according to a certain network protocol. This means that attack packets must be ensconced in traffic alongside packets containing fixed parts of the network protocol, such as normal connection establishments, key exchanges, connections, and disconnections. In the normal portion of the attack traffic, no data can be used to determine whether the packet is intended to cause an attack. Using a CNN alone to train the characteristics of a single packet as the basis for the system to judge the nature of the traffic makes the data difficult to mark, leaves too much "dirty" data in the traffic, and produces altogether poor training results. In this study, we remedied this by introducing the LSTM, which takes the data of a single connection (from initiation to disconnection) as a group and judges the characteristics of all data packets in said group and the relations among them as a basis to judge the nature of the traffic. The natural language processing model performs well in traffic information processing [32] under a similar methodology as the grouping judgment method proposed here.

The LSTM section is composed of LSTM layer 6, LSTM layer 7, full connection layer 8, and Softmax and output layers. The main functions are realized by two LSTM layers. The LSTM is a special RNN designed to resolve gradient disappearance and gradient explosion problems in the process of long sequence training. General RNN networks only have one tanh layer, while LSTM networks perform better processing timing prediction through their unique forgetting and selective memory gates. Here, we call the LSTM node a cell (C_t), the input and output of which are x_t and h_t, respectively.

The first step in the LSTM layer is to determine what information the model will discard from the cell state. This decision is made through the forgetting gate (formula 7). The gate reads h_{t-1} and x_t and outputs a value between 0 and 1 to each number in the C_{t-1} cell state; 1 means "completely retained" and 0 means "completely discarded." W and b are weight and bias in the neural network, respectively.

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right). \tag{7}$$

The next step is to decide how much new information to add to the cell state. First, a sigmoid layer determines which information needs to be updated (formula 8). A tanh layer generates a vector as an alternative for updating (formula 9). The two parts are then combined to make an update to the state of the cell (formula 10).

$$i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right), \tag{8}$$

$$\widetilde{C}_t = \tan h\left(W_c \cdot \left[h_{t-1}, x_t\right] + b_c\right), \tag{9}$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t. \tag{10}$$

The output gate determines the output of the cell. First, a sigmoid layer determines which parts of the cell state are exported (formula 11). Next, the cell state is processed through a tanh function to obtain a value between −1 and 1 and then multiplied by the output of the sigmoid gate. The output is determined accordingly (formula 12).

$$o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right), \tag{11}$$

$$h_t = o_t * \tan h\left(C_t\right). \tag{12}$$

In the proposed model, the feature maps of n data packets in a group of traffic images in a connection serve as the input of the LSTM section. The feature relations between these n data packets were analyzed through the two LSTM layers. The first few packets may be used to establish connections; such packets may exist in the normal data streams,but they may occur in the attack data streams too. The next few packets may contain long payloads as well as attack data. The LSTM finds the groups containing attack data and marks all packets of those whole groups as attack groups.

LSTM layer 6 in DL-IDS has a linear activation function designed to minimize the training time. LSTM layer 7 is nonlinearly activated through the ReLU function. The flow comprises a multiclassification system, so the model is trained to minimize multiclass cross-entropy. We did not update the ownership weight at every step but instead only needed to add the initial weight according to the volume of various types of data.

4.3. Weight of Classes. The data obtained after preprocessing is shown in Table 3, where, clearly, the quantities ("numbers") of different data types are uneven. The number of type 0 is the highest, while those of types 2 and 4 are the lowest. This may affect the final learning outcome of the classification. For example, if the machine were to judge all the traffic as type 0, the accuracy of the model would seem to be relatively high. We introduced the weights of classes to resolve this problem: classes with different sample numbers in the classification were given different weights, class_weight is set according to the number of samples, and class_weight[i] is used instead of 1 to punish the errors in the class [i] samples. A higher class_weight means a greater emphasis on the class. Compared with the case without considering the weight, more samples are classified into high-weight classes.

Table 3: Quantity of data per category in CICIDS2017.

Label	Attack	Num
0	Normal	477584
1	FTP-Patator	11870
2	SSH-Patator	7428
3	DoS	63240
4	Heartbleed	4755
5	Infiltration	64558
6	PortScan	160002

The class weight is calculated via formula 13, where w_i represents the class weight of class i and n_i represents the amount of traffic of class i.

$$w_i = \frac{\sum_{i=0}^{K-1} n_i}{n_i}. \tag{13}$$

When training the model, the weighted loss function in formula 14 makes the model focus more on samples from underrepresented classes. K is the number of categories, y is the label (if the sample category is i, then $y_i = 1$; otherwise $y_i = 0$) and p is the output of the neural network, which is the probability that the model predicts that the category is i and is calculated by Softmax in this model. Loss function J is defined as follows:

$$J = -\sum_{i=0}^{K-1} w_i y_i \log\left(p_i\right). \tag{14}$$

5. Experimental Results and Analysis

We evaluated the performance of the proposed model on the CICIDS2017 dataset using a series of selected parameters: (1) the impact of the length of data packets involved in training; (2) the influence of the number of packets in each flow; (3) the impact of the selected batch size; (4) the effect of the number of units in LSTM; and (5) the influence of the weight of classes. We optimized the DL-IDS parameters accordingly and then compared them against a sole CNN and a sole LSTM. The ratio of Train set, Validation set, and Test set is 18 : 1 : 1.

5.1. Metrics. We adopted four commonly used parameters for evaluating intrusion detection systems: accuracy (ACC), true positive rate (TPR), false positive rate (FPR), and F1-score. ACC represents the overall performance of the model, TPR represents the ratio of the real positive sample in the current positive sample to all positive samples, and FPR represents the ratio of the real negative sample wrongly assigned to the positive sample type to the total number of all negative samples. Recall represents the number of True Positives divided by the number of True Positives and the number of False Negatives. Precision represents the number of positive predictions divided by the total number of positive class values predicted. F1-score is a score of a classifier's accuracy and is defined as the weighted harmonic mean of the Precision and Recall measures of the classifier.

$$\begin{aligned}
\mathrm{ACC} &= \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}+\mathrm{TN}} * 100\%, \\
\mathrm{TPR} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} * 100\%, \\
\mathrm{FPR} &= \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}} * 100\%, \\
\mathrm{Precision} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} * 100\%, \\
\mathrm{Recall} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} * 100\%, \\
\mathrm{F1-score} &= \frac{2 * \mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}} * 100\%.
\end{aligned} \tag{15}$$

For each type of attack, TP is the number of samples correctly classified as this type, TN is the number of samples correctly classified as not this type, FP is the number of samples incorrectly classified as this type, and FN is the number of samples incorrectly classified as not this type. The definitions of TP, TN, FP, and FN are given in Figure 5.

5.2. Experimental Environment. The experimental configuration we used to evaluate the model parameters is described in Table 4.

5.3. LSTM Unit Quantity Effects on Model Performance. The number of units in the LSTM represents the model's output dimension. In our experiments, we found that model performance is first enhanced and then begins to decline as the number of LSTM units continually increases. We ultimately selected 85 as the optimal number of LSTM units.

5.4. Training Packet Length Effects on Model Performance. Figure 6 shows the changes in ACC, TPR, and FPR with increase in the length of packets extracted during training. As per the training results, model performance significantly declines when the package length exceeds 70. It is possible that excessively long training data packets increase the proportion of data packets smaller than the current packet length, leading to an increase in the proportion of units with a median value of −1 and thus reducing the accuracy of the model. However, the data packet must exceed a certain length to ensure that effective, credible, and scientific content is put into training. This also prevents overfitting effects and provides more accurate classification ability for data packets with partial header similarity. We found that a length of 40 is optimal.

Under the condition that the packet length is 40, the efficiency and performance of the DL-IDS intrusion detection system in identifying various kinds of traffic are shown in Table 5.

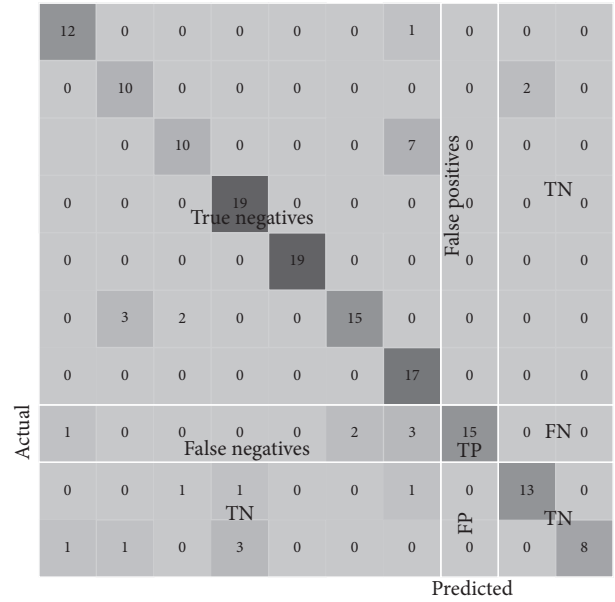


Figure 5: TP, TN, FP, and FN in multiple classifications.

Table 4: Experimental environment.

OS	CentOS Linux release 7.5.1804
CPU	Intel (R) Xeon (R) CPU E5-2620 v3 @ 2.40 GHz
RAM	126 GB
Anaconda	4.5.11
Keras	2.2.2
Python	3.6.5

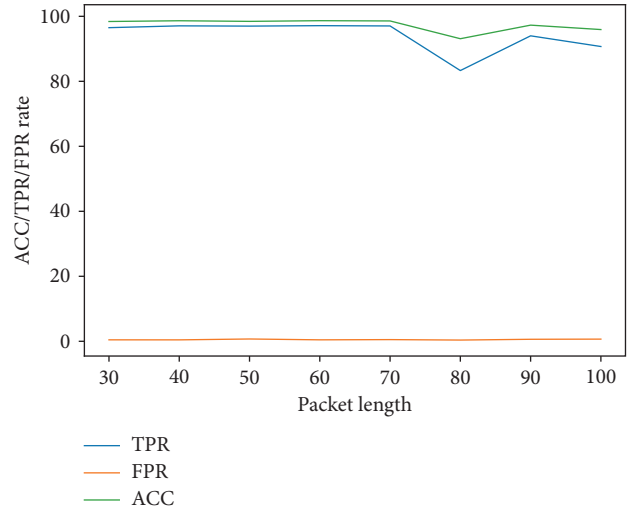


Figure 6: Impact of training packet length on model performance.

5.5. Per-Flow Packet Quantity Effects on Model Performance. As the number of data packets in each flow involved in the training process increases, the features extracted by the model become more obvious and the recognition accuracy

Table 5: Model performance with training packet length of 40.

Label	ACC (%)	TPR (%)	FPR (%)	F1-score (%)
Normal	99.54	99.52	0.00	99.61
FTP-Patator	99.62	92.29	0.27	91.45
SSH-Patator	99.63	87.04	0.00	86.87
Dos	99.61	98.25	0.00	97.87
Heartbleed	99.66	81.12	0.00	81.20
Infiltration	99.55	98.07	0.18	97.54
PortScan	99.54	98.42	0.00	98.71
Overall	98.67	97.21	0.47	93.32

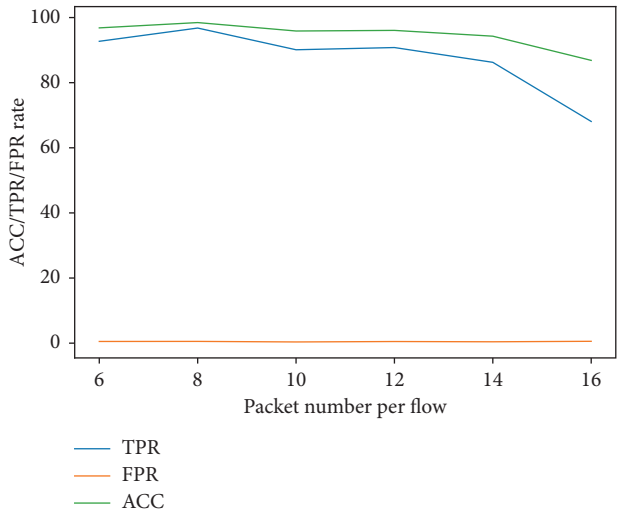


Figure 7: Impact of per-flow packet quantity on model performance.

of the model is enhanced. If this number is too high, however, the proportion of filling data packets increases, thus affecting the model's ability to extract features. Figure 7 shows the impact of the number of packets per flow on model performance. We found that when the number of packets in each flow exceeds 8, the performance of the model declines significantly. We chose 8 as the optimal value of per-flow packet quantity in the network.

5.6. Batch Size Effects on Model Performance. Batch size is an important parameter in the model training process. Within a reasonable range, increasing the batch size can improve the memory utilization and speed up the data processing. If increased to an inappropriate extent, however, it can significantly slow down the process. As shown in Figure 8, we found that a batch size of 20 is optimal.

5.7. Class Weight Effects on Model Performance. Table 6 shows a comparison of two groups of experimental results with and without class weights. Introducing the class weight does appear to reduce the impact of the imbalance of the number of data of various types in the CICIDS2017 dataset on model performance.

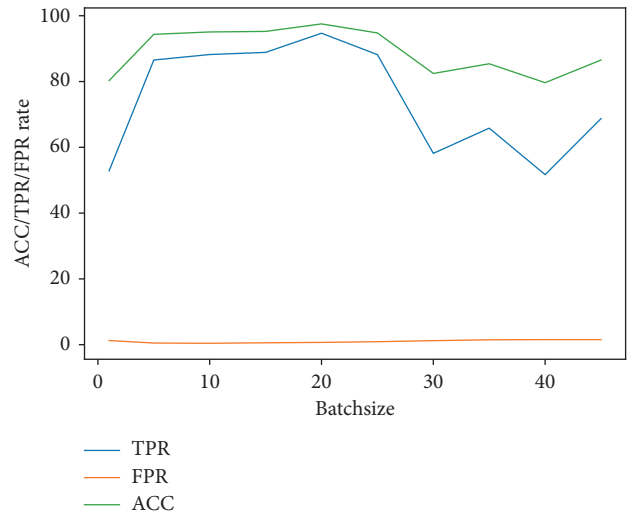


Figure 8: Selected batch size affecting model performance.

Table 6: Effects of applying class weight on model performance.

	Without class weights (%)	With class weights (%)
ACC	97.16	98.58
TPR	93.33	97.04
FPR	0.38	0.52

5.8. Model Evaluation. The LSTM unit can effectively extract the temporal relationship between packets. Table 7 shows a comparison of accuracy between the DL-IDS model and models with the CNN or LSTM alone. The LSTM unit appears to effectively improve the identification efficiency of SSH-Patator, Infiltration, PortScan, and other attack traffic for enhanced model performance, possibly due to the apparent timing of these attacks. Compared to the LSTM model alone, however, adding a CNN further improves the identification efficiency of most attack traffic. As shown in Table 7, the proposed DL-IDS intrusion detection model has very low false alarm rate and can classify network traffic more accurately than the CNN or LSTM alone.

Table 8 shows a comparison of models using CNN and LSTM with traditional machine learning algorithms [33]. The DL-IDS model achieves the best performance among them, with the largest ACC value and the lowest FPR value.

The data input to DL-IDS is raw network traffic. There is no special feature extraction in the model; the training and testing time include the feature extraction time. The traditional machine learning algorithm does not consider data extraction or processing time, so we could not directly compare the time consumption of the various algorithms in Table 8. The training time and testing time of the model were under 600 s and 100 s, respectively, so we believe that the DL-IDS achieves optimal detection effects in the same time frame as the traditional algorithm.

Table 7: CNN, LSTM, and CNN + LSTM results.

Label	CNN (%)				LSTM (%)				CNN + LSTM (%)			
	ACC	TPR	FPR	F1-score	ACC	TPR	FPR	F1-score	ACC	TPR	FPR	F1-score
Normal	99.57	99.56	0.00	99.64	99.56	99.55	0.00	99.63	99.54	99.52	0.00	99.61
FTP-Patator	99.45	91.95	0.27	89.95	99.34	89.27	0.15	91.72	99.62	92.29	0.27	91.45
SSH-Patator	99.53	80.90	0.00	87.22	98.91	79.91	0.00	86.66	99.63	87.04	0.00	86.87
Dos	99.53	98.04	0.00	97.86	98.55	94.85	0.13	89.89	99.61	98.25	0.00	97.87
Heartbleed	99.64	80.08	0.00	80.88	99.63	81.12	0.07	77.69	99.66	81.12	0.00	81.20
Infiltration	99.59	97.91	0.07	97.75	98.84	96.80	1.16	97.17	99.55	98.07	0.18	97.54
PortScan	99.35	97.48	0.00	98.51	99.42	97.99	0.00	94.04	99.54	98.42	0.00	98.71
Overall	98.44	96.46	**0.36**	93.11	96.83	94.21	1.54	90.97	**98.67**	**97.21**	0.47	**93.32**

Table 8: CNN and LSTM models versus traditional machine learning algorithms.

	ACC (%)	TPR (%)	FPR (%)	F1-score (%)
MultinomialNB	72.52	78.20	33.16	52.06
Random Forest	96.08	95.47	3.30	76.71
J48	97.32	96.80	2.17	91.43
Logistic Regression	97.68	94.96	1.47	90.55
DL-IDS	**98.67**	**97.21**	**0.47**	**93.32**

6. Conclusions and Future Research Directions

In this study, we proposed a DL-based intrusion detection system named DL-IDS, which utilized a hybrid of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) to extract features from the network data flow to analyze the network traffic. In DL-IDS, CNN and LSTM, respectively, extract the spatial features of a single packet and the temporal feature of the data stream and finally fuse them, which improve the performance of intrusion detection system. Moreover, DL-IDS uses category weights for optimization in the training phase. This optimization method reduced the adverse of the number of unbalanced samples of attack types in Train set and improved the robustness of the model.

To evaluate the proposed system, we experimented on the CICIDS2017 dataset, which is often used by researchers for the benchmark. Normal traffic data and some attack data of six typical types of FTP-Patator, SSH-Patator, Dos, Heartbleed, Infiltration, and PortScan were selected to test the ability of DL-IDS to detect attack data. Besides, we also used the same data to test the CNN-only model, the LSTM-only model, and some commonly used machine learning models.

The results show that DL-IDS reached 98.67% and 93.32% in overall accuracy and F1-score, respectively, which performed better than all machine learning models. Also, compared with the CNN-only model and the LSTM-only model, DL-IDS reached over 99.50% in the accuracy of all attack types and achieved the best performance among these three models.

There are yet certain drawbacks to the proposed model, including low detection accuracy on Heartbleed and SSH-Patator attacks due to data lack. Generative Adversarial Networks (GAN) may be considered to overcome the drawback to some degree. Further, combining with some traditional traffic features may enhance the overall model performance. We plan to resolve these problems through further research.

Data Availability

Data will be made available upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research was funded by Beijing Natural Science Foundation under Grant no. 4194086, the National Natural Science Foundation of China under Grant no. 61702043, and the Key R&D Program of Heibei Province under Grant no. 201313701D.

References

[1] M. M. Hassan, A. Gumaei, S. Huda, and A. Ahmad, "Increasing the trustworthiness in the industrial IoT networks through a reliable cyberattack detection model," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, 2020.

[2] F. A. Khan and A. Gumaei, "A comparative study of machine learning classifiers for network intrusion detection," in *Proceedings of the International Conference on Artificial Intelligence and Security*, pp. 75–86, New York, NY, USA, July 2019.

[3] M. Alqahtani, A. Gumaei, M. Mathkour, and M. Maher Ben Ismail, "A genetic-based extreme gradient boosting model for detecting intrusions in wireless sensor networks," *Sensors*, vol. 19, no. 20, p. 4383, 2019.

[4] A. Derhab, M. Guerroumi, A. Gumaei et al., "Blockchain and random subspace learning-based IDS for SDN-enabled industrial IoT security," *Sensors*, vol. 19, no. 14, p. 3119, 2019.

[5] T. Yaqoob, H. Abbas, and M. Atiquzzaman, "Security vulnerabilities, attacks, countermeasures, and regulations of networked medical devices-a review," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3723–3768, 2019.

[6] X. Jing, Z. Yan, X. Jiang, and W. Pedrycz, "Network traffic fusion and analysis against DDoS flooding attacks with a novel reversible sketch," *Information Fusion*, vol. 51, no. 51, pp. 100–113, 2019.

[7] Z. A. Baig, S. Sanguanpong, S. N. Firdous et al., "Averaged dependence estimators for DoS attack detection in IoT networks," *Future Generation Computer Systems*, vol. 102, pp. 198–209, 2020.

[8] Y. Yuan, H. Yuan, D. W. C. Ho, and L. Guo, "Resilient control of wireless networked control system under denial-of-service attacks: a cross-layer design approach," *IEEE Transactions on Cybernetics*, vol. 50, no. 1, pp. 48–60, 2020.

[9] A. Verma and V. Ranga, "Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning," *Procedia Computer Science*, vol. 125, pp. 709–716, 2018.

[10] H. Xu, F. Mueller, M. Acharya et al., "Machine learning enhanced real-time intrusion detection using timing information," in *Proceedings of the International Workshop on Trustworthy & Real-Time Edge Computing for Cyber-Physical Systems*, Nashville, TN, USA, 2018.

[11] Y. Wang, W. Meng, W. Li, J. Li, W.-X. Liu, and Y. Xiang, "A fog-based privacy-preserving approach for distributed signature-based intrusion detection," *Journal of Parallel and Distributed Computing*, vol. 122, pp. 26–35, 2018.

[12] H. Xu, C. Fang, Q. Cao et al., "Application of a distance-weighted KNN algorithm improved by moth-flame optimization in network intrusion detection," in *Proceedings of the 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS)*, pp. 166–170, IEEE, Lviv, Ukraine, September 2018.

[13] S. Teng, N. Wu, H. Zhu, L. Teng, and W. Zhang, "SVM-DT-based adaptive and collaborative intrusion detection," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 108–118, 2018.

[14] J. Liu and L. Xu, "Improvement of SOM classification algorithm and application effect analysis in intrusion detection," *Recent Developments in Intelligent Computing, Communication and Devices*, pp. 559–565, Springer, Berlin, Germany, 2019.

[15] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[16] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, 2016.

[17] Q. Niyaz, W. Sun, A. Y. Javaid, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th International Conference on Bio-inspired Information and Communications Technologies*, pp. 21–26, Columbia, NY, USA, December 2015.

[18] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems," *Expert Systems with Applications*, vol. 42, no. 5, pp. 2670–2679, 2015.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, USA, 2016.

[20] Z. Wang, *The Applications of Deep Learning on Traffic Identification*, pp. 21–26, BlackHat, Las Vegas, NV, USA, 2015.

[21] J. Fan and K. Ling-zhi, *Intrusion Detection Algorithm Based on Convolutional Neural Network*, Beijing Institude of Technology, Beijing, China, 2017.

[22] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proceedings of the 2016 International Conference on Platform Technology and Service (PlatCon)*, February 2016.

[23] P. Wu and H. Guo, "LuNET: a deep neural network for network intrusion detection," in *Proceedings of the Symposium Series on Computational Intelligence (SSCI)*, IEEE, Xiamen, ChinaIEEE, Xiamen, China, December 2019.

[24] C. M. Hsu, H. Y. Hsieh, S. W. Prakosa, M. Z. Azhari, and J. S. Leu, "Using long-short-term memory based convolutional neural networks for network intrusion detection," in *Proceedings of the International Wireless Internet Conference*, Springer, Taipei, Taiwan, pp. 86–94, October 2018.

[25] M. Ahsan and K. Nygard, "Convolutional neural networks with LSTM for intrusion detection," in *Proceedings of the 35th International Conference*, vol. 69, pp. 69–79, Seville, Spain, May 2020.

[26] M. M. Hassan, A. Gumaei, A. Ahmed, M. Alrubaian, and G. Fortino, "A hybrid deep learning model for efficient intrusion detection in big data environment," *Information Sciences*, vol. 513, pp. 386–396, 2020.

[27] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, p. 322, 2019.

[28] H. Musafer, A. Abuzneid, M. Faezipour, and A. Mahmood, "An enhanced design of Sparse autoencoder for latent features extraction based on trigonometric simplexes for network intrusion detection systems," *Electronics*, vol. 9, no. 2, p. 259, 2020.

[29] V. Ramos and A. Abraham, "ANTIDS: self organized ant based clustering model for intrusion detection system," in *Proceedings of the Fourth IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST'05)*, Springer, Muroran, JapanSpringer, Muroran, Japan, May 2005.

[30] I. Sharafaldin, A. H. Lashkari, and A. Ali, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the The Fourth International Conference on Information Systems Security and Privacy (ICISSP)*, Madeira, Portugal, January 2018.

[31] X. Chen, "A simple utility to classify packets into flows," https://github.com/caesar0301/pkt2flow.

[32] B. J. Radford and B. D. Richardson, "Sequence aggregation rules for anomaly detection in computer network traffic," 2018, https://arxiv.org/abs/1805.03735v2.

[33] A. Ahmim, M. A. Ferrag, L. Maglaras, M. Derdour, and H. Janicke, "A detailed analysis of using supervised machine learning for intrusion detection," 2019, https://www.researchgate.net/publication/331673991_A_Detailed_Analysis_of_Using_Supervised_Machine_Learning_for_Intrusion_Detection.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8872923, 17 pages
https://doi.org/10.1155/2020/8872923

Review Article

Network Attacks Detection Methods Based on Deep Learning Techniques: A Survey

Yirui Wu, Dabao Wei, and Jun Feng

College of Computer and Information, Hohai University, Nanjing, China

Correspondence should be addressed to Jun Feng; fengjun@hhu.edu.cn

Received 7 May 2020; Revised 26 June 2020; Accepted 20 July 2020; Published 28 August 2020

Academic Editor: Xiaolong Xu

With the development of the fifth-generation networks and artificial intelligence technologies, new threats and challenges have emerged to wireless communication system, especially in cybersecurity. In this paper, we offer a review on attack detection methods involving strength of deep learning techniques. Specifically, we firstly summarize fundamental problems of network security and attack detection and introduce several successful related applications using deep learning structure. On the basis of categorization on deep learning methods, we pay special attention to attack detection methods built on different kinds of architectures, such as autoencoders, generative adversarial network, recurrent neural network, and convolutional neural network. Afterwards, we present some benchmark datasets with descriptions and compare the performance of representing approaches to show the current working state of attack detection methods with deep learning structures. Finally, we summarize this paper and discuss some ways to improve the performance of attack detection under thoughts of utilizing deep learning structures.

1. Introduction

The continuous development and extensive usage of Internet benefit numerous network users from a quantity of aspects. Meanwhile, network security becomes much more important with wide usage of network. Network security is closely related to computers, networks, programs, various data, and so forth, where the purpose of defense is to prevent unauthorized access and modification [1]. However, the growing number of internet-connected systems in finance, E-commerce, and military makes them become targets of network attacks, resulting in large quantity of risk and damage. Essentially, it is necessary to provide effective strategies to detect and defend attacks and maintain network security. Furthermore, different kinds of attacks are usually required to be processed in different ways. How to identify different kinds of network attacks thus becomes the main challenge in domain of network security to be solved, especially those attacks never seen before.

Over the past several years, researchers have used various kinds of machine learning methods to classify network attacks without prior knowledge of their detailed characteristics. However, traditional machine learning methods are not capable of providing distinctive feature descriptors to describe the problem of attack detection, due to their limitations in model complexity. Recently, machine learning has made a great breakthrough by simulating human brain with structure of neural networks, which are named deep learning methods for their general architecture of deep layers to solve complicated problems. Among these successful applications, Google's AlphaGo is one of the most outstanding trials for the game of "go," involving the strength of a typical kind of deep learning structure, that is, convolutional neural networks.

Since deep learning is complex in its original structures and domain-oriented applications, this paper is written to explain so for those who aim to study in the field of network security by utilizing deep learning methods. Essentially, there exists a quantity of previous work focusing on attack detection using deep learning techniques. Among them, several literature reviews [2–8] have been conducted to get ideas from applying deep learning on attack detection, which

is the foundation of our paper. For example, Berman et al. [5] provide a quantity of reading resources to describe the basic knowledge and development history of deep learning methods and their corresponding applications in attack detection. Different from a complete view on this specific domain brought by Berman et al. [5], Apruzzese et al. [4] focus on explaining attack detection methods related to intrusion detection, malware analysis, and spam detection. In work of Wickramasinghe et al. [7], they mainly review deep learning methods on securing under the usage of Internet of Things technologies, which offers a clear view on variant kinds of cyberattacks and the corresponding techniques used in detection. Afterwards, Aleesa et al. [3] review and analyze the research status of intrusion detection system based on deep learning technology among four major databases. Meanwhile, they offer a systematic literature review of the relevant articles using the keywords "deep learning", "invasion", and "attack" selection, which provide a wide range of resource background for the researchers. By regarding dataset as significantly important to intrusion detection, Ferrag et al. [6] describe 35 well-known network datasets and divide them into seven categories. They introduce seven presentative models for each category, where they evaluate and compare the efficiency via accuracy and false alarm rate based on real traffic datasets, that is, CSE-CIC-IDS2018 and Bot-IoT.

In fact, all the above review papers have their own emphases, such as security applications, attacks type, datasets, or databases. Unlike former methods, we intend to build our paper on the basis of deep learning models, thus paying special attention to attack detection methods built on different kinds of deep learning architectures. Furthermore, we offer a fair comparison and our own specified analysis on performance of representing approaches based on benchmark datasets. We believe our paper could offer a more understandable reading resource for readers, who are interested in how different deep learning architectures affect the area of attack detection.

In the paper, we attempt to build up basics for future research through a thorough literature review of deep learning related approaches in the field of attack detection. More specially, firstly, we summarize the fundamental problems, classify the previous methods, and review the useful methods for beginners. Then, we briefly introduce the great progress on deep learning techniques in cybersecurity. By replacing traditional machine learning methods with deep learning structures, researchers have proposed a quantity of novel algorithms to greatly improve the performance referring to higher detection rate and lower false alarm rate. Afterwards, we compare and analyze the performance of some representative deep learning approaches on benchmark dataset. Finally, we make a summary of the problems to be solved and future direction of deep learning method to improve attack detection.

We organize the rest of our work as follows. Section 2 focuses on concepts of attack detection and cyber applications via research background introduction. Section 3 offers overviews on different deep learning methods for attack detection, which are categorized as unsupervised and supervised methods with different structures. Section 4 presents datasets and analyzes the performance comparisons of a quantity of deep learning methods. Section 5 provides discussion and conclusion based on the current foundations and presents several ideas for future research.

2. Brief Introduction to Attack Detection

In order to provide an overview of effective attack detection based on deep learning techniques, it is essential to introduce background knowledge. We thus first give a brief introduction to the concepts of attack detection, which could offer a basic recognition for new learners. Afterwards, we make a brief representation of successful applications for cybersecurity.

2.1. Developing Process of Attack Detection. Attacks could be recognized as the attempts to bypass security policies of the system, which gives attackers easier access to obtain or modify information, even destroying the system. With technologies developing on wireless communication systems, serious threats to network security, especially security of wireless communication systems, have been proposed by more frequent network attack activities, due to openness characteristics of wireless channels. Since we are now in machine learning and big data epoch [9], cybersecurity in wireless communication systems is important for users to protect network, computer, and data from attacks. There exist variant kinds of attacks for cyber systems, such as flooding, distributed denial of service, abnormal packet attack, and spoofing.

To deal with such attack threads to cybersecurity, researchers have proposed many solutions [10]. Among the solutions, attack detection is one of the most effective ways, which offers a complete and dynamic security mechanism to monitor, prevent, and resist attacks. Specifically, attack detection would collect information by monitoring network, system status, behavior, and the usage of system, which could automatically detect unauthorized usage of system users and attacks of external attackers on the system.

In recent years, machine learning is developing with incredible speed. Among different machine learning methods, deep learning structures construct artificial neural networks to simulate interconnecting neurons of human brains, which brings distinctive power to solve complicated problems. Researchers thus adopt various deep learning methods to operate attack detection, resulting in significant achievements. However, there are still many unsolved problems due to the limitation of deep learning methods. It is essential to make a summarization of how former methods use deep learning methods to detect attacks, which could bring new ideas for future developments.

2.2. Applications of Attack Detection Using Deep Learning Structures. Since deep learning shows great potential in constructing security applications, it has been widely used in cybersecurity [11]. There are numerous related applications such as malware, intrusion, phishing, spam detection, and

traffic analysis [12]. We believe these successful application examples could help analyze users' requirements with the innovation brought by deep learning structures. Thus, we provide some typical applications to present practicability of deep learning method, where we believe these applications can be implemented in domains of multimedia handling [13], signal processing [14], and so on [15].

2.2.1. Intrusion Detection. Intrusion detection system could detect malicious activities by collecting and analyzing network behavior, security log, and other information available on the network and among connected computers [16]. Essentially, intrusion detection system checks existence of abnormal behaviors against system security policy and signs of being attacked in the system, which is capable of protecting the system with real-time responses. Under traditional system settings, intrusion detection system works as a reasonable, active, and efficient supplement to firewall, which actually acts as a passive defense mean to attacks.

Traditional intrusion detection system is firstly built on misuse of intrusion detection technology, which mainly extracts characteristics or rules of intrusion behavior. After appearance of abnormal behavior detection technology with traditional machine learning models, intrusion detection system evolutes to carry out probability statistical modeling for normal behaviors, which could analyze and alarm abnormal behaviors with large deviation. However, such system may have unsatisfactory results, due to low capability in problem space defining and complexity in modeling malicious activities.

To further overcome shortcomings brought by traditional machine learning methods, deep learning technology is performed to analyze network packets, which progressively changes the mainstream idea of intrusion detection from blacklist to white model. A new NIDS deep learning model is proposed by Shone et al. [17], which is helpful to analyze the network traffic under the symmetrical deep autoencoder technology. On the basis of LSTM algorithm, Vinayakumar et al. [18] design a system call modeling approach with integration method for anomaly intrusion detection system. System call modeling helps capture the semantics of each call and relationship on the network. The integration method mainly focuses on the false alarm rate in accordance with IDS design. Currently, a mature intrusion detection system could detect many kinds of attacks with the strength of deep learning structures.

2.2.2. Malware Detection. Malware is designed to reduce performance and vulnerability of a computer, server, or computer network. Under extreme situations, Malware will result in destruction of the entire system. Malware requires to be implanted into the target computer at first. Afterwards, it could execute code, script, active content, and other software automatically or following orders from planters. It is noted that such software or codes could be categorized in forms of computer viruses, worms, Trojans, spyware, advertising software, and malicious codes.

We divide the malware detection methods into two categories, that is, signature-based and anomaly-based detection. Traditional antivirus software can be included in the first category, which detects malicious files based on file signature. However, slightly deformed malicious codes could be bypassed, leading to a large number of false positives. Later, technologies of sandbox and virtual machine appear to detect dynamic behaviors of virus, which can be regarded as big progress from static detection to dynamic analysis, greatly improving the ability to detect unknown malicious code.

For example, in [19], Saxe discusses the deep learning of a four-layer network application. In order to get appropriate computing feature text extraction technology, PE Metadata Features can be used. The author proposes eXpose neural network, where their network takes the original short strings as input and extracts features to classify with character-level embeddings. Because of the feature design of self-extraction, the method of express is better than the baseline method based on manual feature extraction. Pascanu et al.'s [20] echo state network is helpful to extract all information by random time projection technology, the max pooling is used for nonlinear sampling of data, and the logistic regression is used for final classification of data.

2.2.3. Domain Generate Algorithms. DGAs are popular to be used as malware tools to create a great quantity of domain names for tracking communication with C2 server. Different domain names make it difficult to use standard technologies like blacklist or sink-holing to prevent malicious domain names. DGAs are often used in various network attacks, such as spam, personal data theft, and DDoS attacks.

By applying deep learning technologies, DGA is capable of detecting domain names from the perspective of syntax analysis. Specifically, such novel algorithms could not only compare word frequency with normal domain names by n-gram methods but also compare the probabilities of character combination with normal domain names by HMM method. Moreover, it is capable of analyzing the entropy, consonant letter, and other characteristics of domain names, which are utilized in LSTM for abnormal classification. Due to the slow speed and poor performance of traditional technology, Feng et al. [21] provide a deep learning method which helps to distinguish DGA domains from non-DGA domains. In [22], the advantages of featureless extraction of raw domain names as an input in LSTM network are also discussed.

3. Deep Learning Methods for Attack Detection

Considering the current deep learning methods for attack detection [23] and following the categorization of the previous works [24, 25], we roughly divide them into three categories as well, that is, unsupervised (e.g., autoencoder (AE), deep belief network (DBN), and generative adversarial network (GAN)), supervised (e.g., deep neural network (DNN), convolutional neural network (CNN), and recurrent neural network (RNN)), and other hybrid methods; we show

the details of categorization in Figure 1. Essentially, there exist other classification criterions. For example, Berman et al. [5] review the related deep learning methods according to attacks type and focus on how deep learning is used for various attacks. Moreover, Al-Garadi et al. [2] offer a comprehensive view of deep learning methods based on the applications of cybersecurity.

Adopting different kinds of deep learning algorithms could bring variant advantages for attack detection methods. Supervised learning based methods often result in high accuracy, due to quantity of information provided by manually labeled samples. Without sufficient knowledge from labeled data, unsupervised learning based methods are generally low in performance. However, manually labelling is a time-consuming task, especially for complex attacks. There even exist cases that cannot be described by a simple label, due to the inherent complexity of real-world network attacks. Therefore, unsupervised learning based methods could perform well without prior knowledge of attacks, which is an obvious advantage. Hybrid methods decrease the number of training samples and maintain a relatively high performance, which is suitable to deal with variant attack situations. However, it is generally complex in structure and high in computing time, which prevents its wide usage.

3.1. Unsupervised Learning for Attack Detection

3.1.1. Autoencoder Based Methods for Attack Detection. Let us first introduce the architecture of AE, which can be regarded as a data compression algorithm with neural network structure. In fact, it is capable of firstly compressing the input into feature space representation and then reconstructing representation into the output. Since AE can be regarded as a typical representing learning algorithm, it is widely used for dimension reduction and outlier detection. Researchers in cybersecurity also adopt AE to represent abnormal behaviors in its compressed feature space, which brings the advantage of dynamical representation for unknown category of attacks.

To extract informative feature descriptors from original network traffic data, Zhang et al. [26] propose to detect network intrusion by stacking dilated convolutional AE (DCAEs), which is a successful combination of self-taught and representation learning. Specifically, original network traffic data is firstly transformed into a vector through the preprocessing step. During unsupervised training, DCAEs learn the hierarchical structure of feature representation from a large number of unlabeled samples. Afterwards, use backpropagation algorithm and a few labeled instances to fine-tune and improve feature description ability learned from the unlabeled instances. In fact, using original network traffic and unsupervised pretraining makes their model more adaptive and flexible to deal with complicated raw data.

Following the idea to facilitate intrusion detection with AE models, Shone et al. [17] propose nonsymmetric deep AE (NDAE) for unsupervised feature blearing, which successfully reduces computations cost of analysis by combining AE with shallow learning. Specifically, NDAE has an additional coding stage comparing with typical AE, which could reduce complexity and improve the accuracy of the model. We show such structure in Figure 2, where we can observe its hierarchical feature extractor. At the end of their proposed NDAE, they apply the structure of random forest to recognize abnormal situations with the help of feature representation learned from NDAEs. To evaluate their model, the authors have implemented their codes in GPU and evaluated with KDDCup 99 and NSL-KDD, achieving promising results comparing with others.

Since AE is capable of learning potential representation of unknown attacks, Yousefi-Azar et al. [27] propose to learn feature representation with AE structure for different cybersecurity applications, which consists of two training stages, that is, pretraining and fine-tuning. The former stage is designed to search for an appropriate starting point for the fine-tuning stage. After determining the parameters in the pretraining stage, fine-tune stage will coverage offering feature description for input data. Their proposed feature learning scheme can greatly reduce feature dimensions, thus significantly minimizing storage requirements. Experiment results show their feature representation can be used in many domains and could achieve remarkable results comparing with previous works.

Since collected network raw data can be unbalanced in distribution, Farahnakian et al. [28] utilize deep stacked autoencoder to focus on important and informative feature representations, thus constructing classification models to detect abnormal behaviors. Specifically, their proposed network consists of 4 AEs in sequential order, which will be trained in a greedy layerwise fashion. Experimental results on KDDCup 99 dataset show it could achieve high accuracy for abnormal detection, that is, 94.71%, even under the situation of unbalanced data.

In order to construct a flexible system for detecting intrusion attacks, Javaid et al. [29] utilize sparse AE and softmax-regression layer for construction and self-taught learning (STL) for the training process. Specifically, their proposed STL could be divided into two steps, where sparse AE is used for unsupervised feature learning at first and softmax-regression is used for classification after feature extraction. In fact, usage of STL could largely improve the learning ability of constructed network facing unknown attacks, where new categories of attacks can be incrementally analyzed during runtime without troubles of training from scratch.

Following such idea, Papamartzivanos et al. [30] present a more powerful approach with MAPE-K framework, which could construct a misuse intrusion detection system with scalable, self-adaptive, and autonomous characteristics. It could extract generalized features for problem reconstruction, even facing unknown environment and using unlabeled data. They believe their proposed method could work well by grasping the nature of variant attacks, where they further design experiments to show that their method could deal with new situations without updating the training set manually.

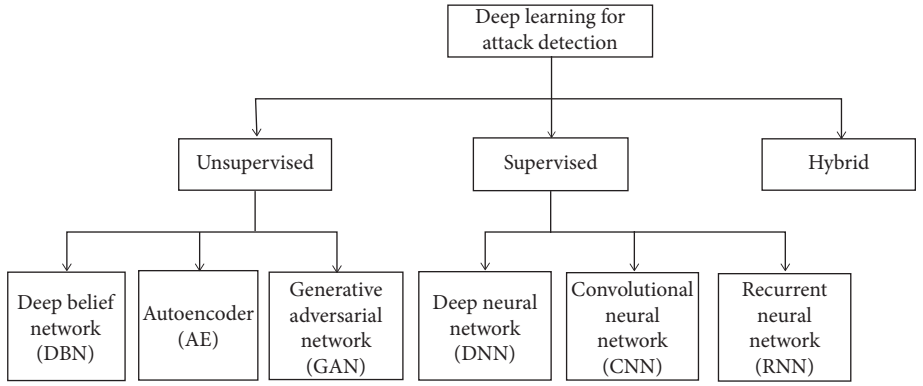


Figure 1: Categorization of the current deep learning methods for attack detection.

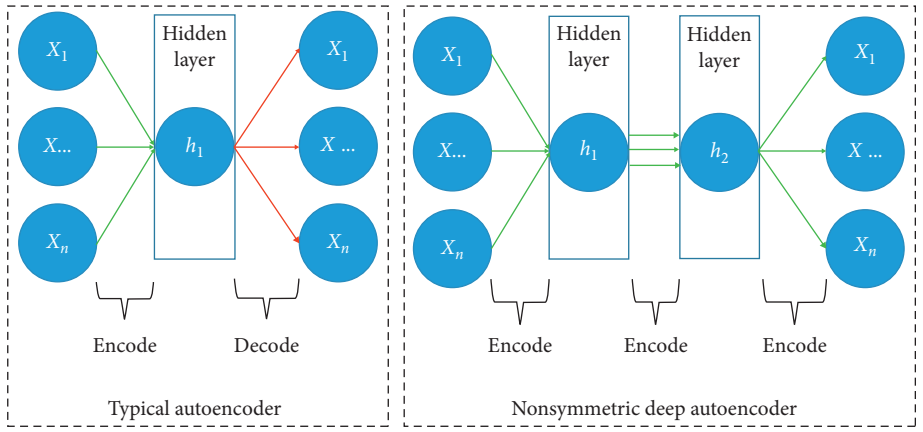


Figure 2: Network structure of Shone et al. [17], which is a novel structure of AE designed with nonsymmetrical multiple hidden layers.

Feature extraction is one of the major issues to address for attack detection. Regarding AE as a structure for information compressing and feature generation, utilizing AE brings advantages of automatical and dynamical feature construction, resulting in high accuracy for detecting predefined attacks existing in datasets. Facing variant and unknown attacks which are the main characteristics in cybersecurity, researchers have emphasized self-learning strategies to make AE more powerful.

3.1.2. Deep Belief Network Based Methods for Attack Detection. Deep belief network (DBN) could be divided into two categories, that is, restricted Boltzmann machines (RBM) with several layers of unsupervised learning networks and backpropagation neural network (BPNN or BP) with one such layer. Essentially, RBM is a random structure of generating neural network, which is undirected graph model composed of different layers constructed by visible neurons and hidden neurons. Due to the natural characteristics of RBM, it is effective for DBN to train layer by layer.

Early, Gao et al. [31] focus on dealing with big raw data and apply deep belief network to construct such intrusion detection system. In their paper, they try different DBN models by adjusting parameters like number of layers and hidden layers. They find the best parameter settings for DBN is a four-layer DBN model, which could achieve better performance than other machine learning methods on KDDCup 99 dataset.

Afterwards, Ding et al. [32] represent malware as opcode sequences and use DBN to detect malware, where they use unsupervised learning to pretrain a multilayer generative model to help DBNs solve the overfitting problem. We show its structure in Figure 3, where we can observe DBN works as classier in the whole workflow with steps of RBM training and BP fine-tuning. With the help of additional unlabeled data, their proposed DBN could achieve accuracy as high as 96%, which outperforms three other traditional artificial intelligence models, that is, SVM, kNN, and decision tree. However, their methods are not justified by other metrics.

Since behavioral characteristics of ad hoc networks have brought great challenges to network security, Tan et al. [33] propose a deep belief network based on ad hoc network intrusion detection structure. Their proposed DBN model contains 6 modules: wireless monitoring node for data fetching, data fusion module to fuse useful data and remove redundancy, DBN training module and DBN intrusion module to train and identify whether there is intrusion, respectively, and response module that expresses results of the proposed model to users. Experimental results show their proposed method can reach 97.6% in accuracy, leading

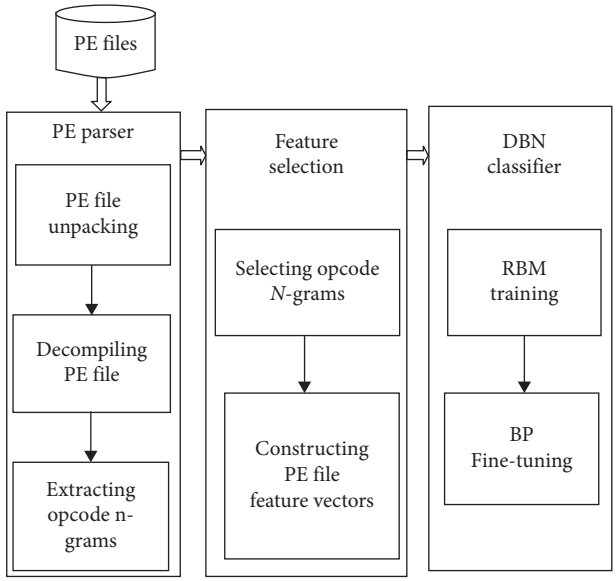


Figure 3: Workflow of opcode malware detection approach proposed by Ding et al. [32], which consists of three major components: PE (Portable Executable) parser, feature extractor, and malware detection module. It is noted that DBN is the core classier of malware detection module.

it to be fit with implementation in intrusion detection applications.

To explore the capabilities of DBN for detecting intrusion attacks, Alom et al. [34] propose an effective platform to explain intrusion attempts in network traffics. Their constructed system firstly uses digital encoding and standardized method to select features and then uses DBN to classify network intrusion by assigning class label to each feature vector. According to their experiments and analysis, their constructed system can not only detect attacks, but also accurately identify and classify network activities according to limited, incomplete, and nonlinear data sources.

Many trials have been applied in using DBN for intrusion detection. However, there still exist many unsolved problems, such as redundant information, easy to trap into local maximal. To solve these problems, Zhao et al. [35] propose to detect intrusion attacks by involving strength of DBN and probabilistic neural network (PNN). Firstly, they rescale original input data to low-dimensional by utilizing nonlinear describing capability of DBN. Meanwhile, DBN could maintain basic characteristics of original data in representation. Secondly, particle swarm optimization algorithm is used to reduce the size of hidden nodes of every layer. Thirdly, PNN is introduced to classify low-dimensional information. Their experiments on KDDCup 99 dataset show they have solved the above problem to a certain extent.

Regarding real-time attacks detection as the biggest challenge of intrusion detection, Alrawashdeh and Purdy [36] propose an anomaly detection method based on DBN, which only consists of one-hidden layer RBM and a fine-tuning layer constructed by logistic regression classifier. Their simplest design of DBN achieves instant running and best performance (reported as 97.7% in accuracy and 8s CPU time for each instance) when testing with KDDCup 99 dataset. Their method offers possibility to implement deep learning methods for attack detection on low computation resource platforms like drones, cell phones, and personal computers, which greatly expands usage scenarios of such methods.

Because the traditional intrusion detection approaches face difficulties dealing with high-speed network data and cannot detect the unknown attacks at present, Zhang et al. [37] propose a network attack detection model integrating flow calculation and deep learning, which comprises two parts: real-time detection algorithm based on frequent patterns and a classification algorithm based on the DBN and SVM. Sliding window stream data processing can realize real-time detection, and the DBN-SVM algorithm can improve classification accuracy. Based on the CICIDS2017 dataset, several groups of comparative experiments are carried out. The method's real-time detection efficiency is higher than that of traditional algorithms.

3.1.3. Generative Adversarial Network Based Methods for Attack Detection. Due to property of discovering inherent pattern of data to generate new samples, generative adversarial network (GAN) is one of the most promising unsupervised learning methods proposed in recent years. The main inspiration of GAN comes from the idea of zero-sum game. When it is applied to deep neural network, it keeps playing games between generator G and discriminator D, and finally G is capable of learning distribution representation of actual data. G is to imitate, model, and learn distribution characteristics of real data as much as possible, while the task of D is to distinguish whether an input data comes from real data or output of G. Through the continuous competition between these internal models, the generation ability and discrimination ability of both G and D can be greatly improved.

Even though GAN is new in conception and hard in the training process, researchers successfully build several attack detection applications by regarding it as basic structure. For instance, Erpek et al. [38] propose a GAN-based approach to detect jamming attacks on wireless communications and defend it based on collected information of attacks. Specifically, their model consists of a transmitter, a receiver, and a jammer. A pretrained classifier is adopted by the transmitter to predict the current channel state and decide whether to send based on the latest sensing results, while the jammer collects the channel state and ACKs to construct a classifier, which could predict next transmission and block it successfully. The jammer uses classification score to control the power under the average power constraint. Afterward, a GAN is designed to perform as a jammer, which can cut down collection time by adding synthetic samples.

Utilizing machine learning technology to perform phishing detection, that is, URL of fake web address, is popular, due to its high effectiveness and real-time response. However, adversary may bypass URL classification algorithm by modifying components. To solve this problem,

AlEroud and Karabatis [39] propose to generate URL-based phishing examples by using generator of GAN, which are then shipped to discriminator, that is, black-box phishing detector. In their proposed GAN model where its structure is shown in Figure 4, generator network could generate disturbed versions of real phishing examples and convert them into adversary examples. Discriminator network learns to classify both generated examples and real ones working as a phishing detector, where the generator parameters and weights are updated with information passing from the discriminator. After testing with a public phishing dataset, their experimental results show that their proposed GAN is successful by avoiding a large number of unknown phishing examples.

GAN is not often used for attack detection field. In fact, GAN is in fast developing in terms of structures, algorithms, and so forth. At present, GAN have shown promising results in many domains, which lead us to believe this proposing new technique to synthesize attempts is quite significant in creating a defensive mechanism. Such novel defensive mechanism can further complete quantity of tasks, such as preventing zero-day phishing attempts, performing opinion spam, and detecting intrusion attacks. Therefore, we think there exists a broad research space to connect GAN structure with attack detection filed.

3.2. Supervised Learning for Attack Detection

3.2.1. Deep Neural Network Based Methods for Attack Detection. DNN is recognized as multilayer perceptron due to characteristic of multiple hidden layers. Such multilayer feature brings advantage to express complex functions with fewer parameters, which makes DNN capable of facilitating tasks of feature extraction and representation learning. Essentially, there exist three categories of layers in DNN. Generally speaking, we regard the first layer as input layer, the last layer as output layer, and middle layers as hidden layers.

To provide a solution to network security problem, Tang et al. [40] propose a DNN model to perform flow based anomaly detection. Their first attempt in applying DNN for network security results in a relatively simple DNN, which is composed of one input layer, three hidden layers, and one output layer. Some experiments are carried out on NSL-KDD dataset, where the proposed DNN model is proven to detect zero-day attack and behaves better than the other machine learning methods.

To enhance ability of DNN, Li et al. [41] propose a novel network structure called HashTran-DNN to classify Android malware. We show its architecture design in Figure 5, where we can observe their most innovation point lies in transforming input samples by using hash functions to preserve locality characteristics. After transforming input data, HashTran-DNN uses AE to perform denoising task, so that DNN classifier can obtain locality information in the potential space for better performance. After analyzing the experimental results, we can observe that HashTran-DNN can effectively defend against four special testing attacks, where standard DNN fail to detect all these attacks.

Challenges arise motivated by the fact that malicious attacks are constantly varying and occur on very large volumes which require scalable solutions. To meet this challenge, a DNN structure with a scalable and hybrid design is proposed by Vinayakumar et al. [18], which can watch network traffic and host level events in real-time, actively warning possible network attacks. Specifically, their proposed framework adopts scalable computing architecture, text representation method, and DNNs to meet the requirement to process big data, where DNN could help improve the performance of their model with functions of nonlinear activation.

For network administrator, it is an urgent task to prevent the invasion of malicious network hackers and keep the network system and computer in a safe and normal operation state. Peng et al. [42] propose a network intrusion detection method based on deep learning, which uses deep neural network to extract features of network monitoring data, and BP neural network is used to classify intrusion types. The method is evaluated by KDDCup 99 dataset. The results show that the method achieves the accuracy of 95.45%, and it has a significant improvement while compared with the traditional machine learning method.

3.2.2. Convolutional Neural Network Based Methods for Attack Detection. CNN involves convolution computation and depth structure, which is a representative and commonly used techniques in deep learning domain. Specifically, CNN uses multilayer perception variant design requiring minimal preprocessing. The basic structure of CNN is composed of input and output layers and multiple hidden layers which include convolution, pooling, and full connection layer. Compared with other classification algorithms, CNN uses relatively less preprocessing and is independent of feature design containing prior knowledge, which are its main advantages.

Convolutional neural network has been applied to network security field with much promising progress. For example, Kolosnjaji et al. [43] attempt to construct a neural network with convolutional and recursive network layers, which obtains classification features to model malware detection system. Through their proposed method, they obtain a hierarchical feature extraction architecture, which combines advantages of convolution operation from convolutional layer and sequence modeling from recursive network layer. Afterwards, Kolosnjaji et al. [44] further develop it to involve with feature derived from headers of Portable Executable files, which achieves quite remarkable accuracy and recall rate under cases of fusing data.

To detect attack indicators in advance, Saxe and Berlin [19] propose eXpose neural network, where their network takes the original short strings as input and extracts features to classify with character-level embeddings. It is noted that their original inputs are a wide and complicated range for algorithms to deal with. Owing to the self-extracted feature design, eXpose is superior to baseline methods based on

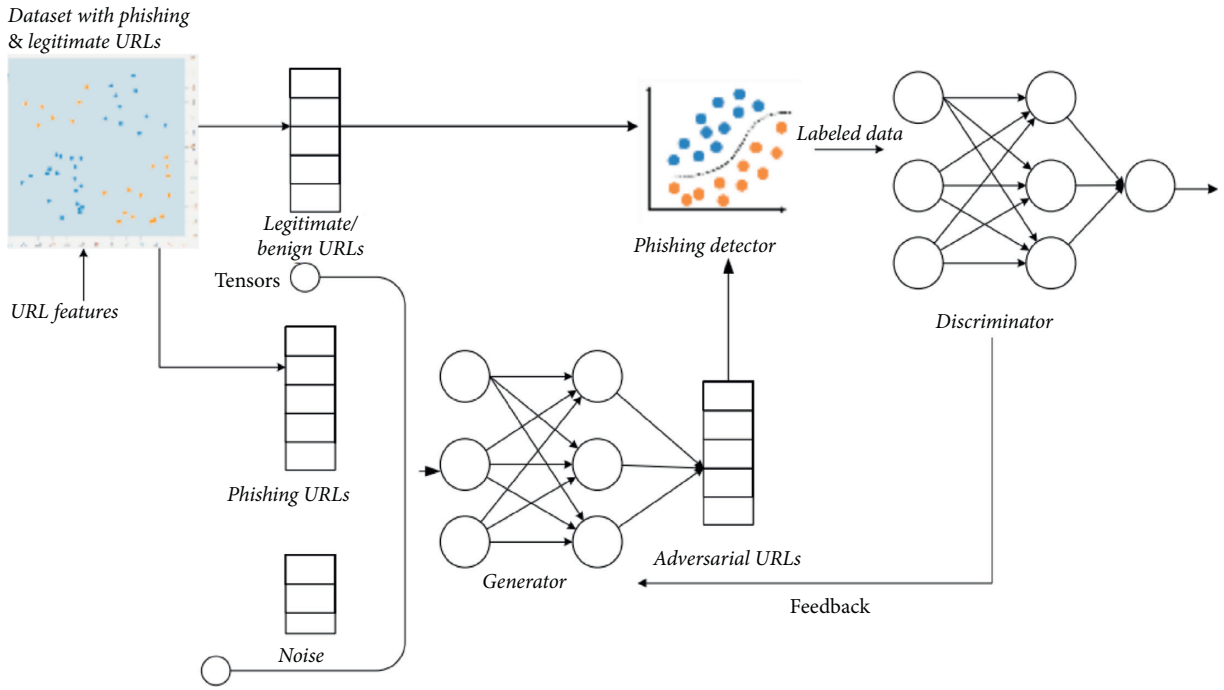


Figure 4: Overview of steps for the GAN model proposed by AlEroud and Karabatis [39].

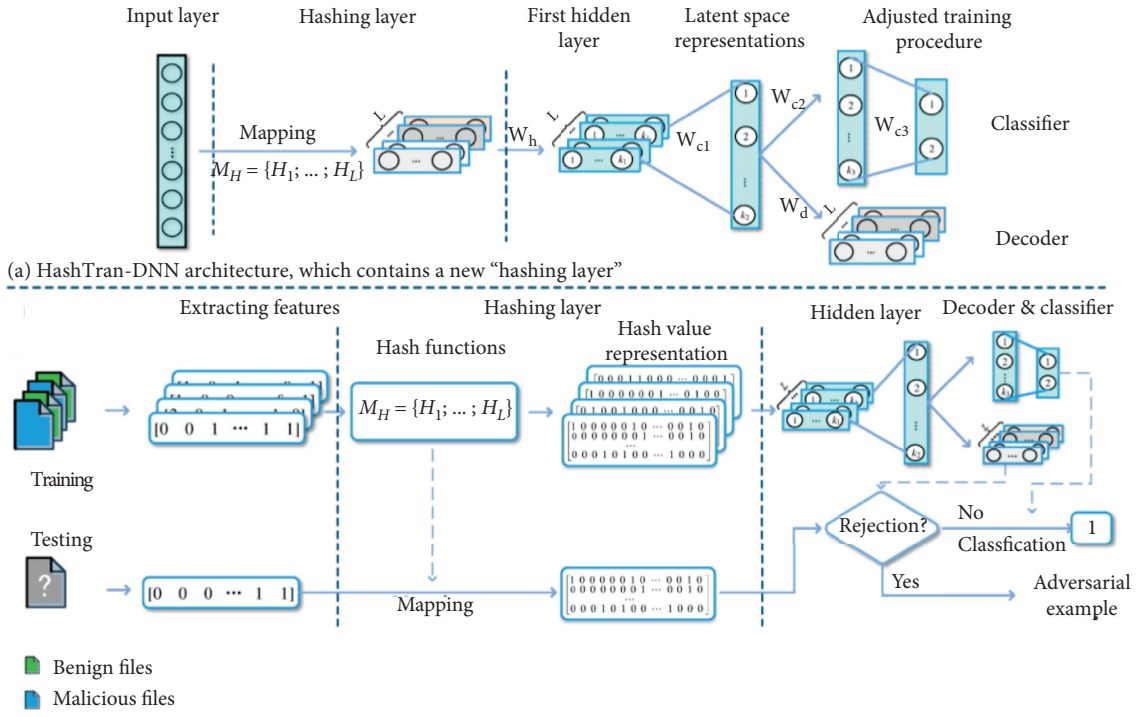


Figure 5: Architecture design of HashTran-DNN model proposed by Li et al. [41].

manual feature extraction. However, it achieves a decrease in false alarm rate compared with these baselines, which proves automatically feature extraction process in CNN is not robust and reasonable enough with introducing extra or even noise information from original inputs.

Malicious web shell detection is an important means to protect network security. Aiming at analysis of HTTP requests, Zhang et al. [46] propose a word2vec representing and CNN-based malicious detection approach, which is the first attempt to combine "word2vec" and CNN in malicious detection domain. Specifically, they first introduce the "word2vec" tool to represent each word obtained from HTTP by features. Then, they represent the web request as a fixed-size matrix by concatenating features. Finally, they

build up the shell classification model based on CNN structure. Several groups of experiment are carried out, and the proposed method performs the best when comparing with relevant classical classifiers.

To achieve robust performance in attack detection with CNN structure, an end-to-end encrypted traffic classification method based on one-dimensional CNN is presented by Wang et al. [45], in which feature extraction, selection, and classifier are integrated into an end-to-end framework. We show its detailed network design in Figure 6, where their proposed 1D-CNN as learning algorithm directly learns relationship between automatical extracted features and outputs with predicted labels in training phase. In their experiment, they adopt ISCX VPN-nonVPN traffic dataset for verification, where they achieve better performance than the latest methods in 11 of 12 evaluation measurements. Such promising results are remarkable, due to robust and informative traffic data representation and fine-tuning steps to improve model ability. Regarding network traffic data as two-dimensional image, a new traffic analysis approach based on CNN is further proposed by Wang et al. [47]. They test their algorithm on USTC-TRC2016 flow dataset to show average classification accuracy is as high as 99%.

To solve the diversity attack of wireless network traffic and improve the detection ability of malicious intrusion in wireless network, an intrusion detection method based on improved convolutional neural network is proposed by Yang and Wang [48], namely, ICNN-Based Wireless Network Intrusion Detection Model. Preprocess the network traffic data, and then model the data using CNN. CNN abstracts low-level intrusion traffic data into high-level features, automatically extracts sample features and optimizes network parameters through random gradient descent algorithm to converge the model. The results on the KDDTest + show that the detection accuracy is 8.82% and 0.51% higher than that of LeNet-5 and DBN, while the false positive rate is also lower. It also has a big advantage compared to the previous methods.

Low rate denial of service (LDOS) attacks reduce the performance of network services, and it is difficult to distinguish the attack behavior from the normal traffic. Thus, a new detection method of LDOS attack based on multifeature fusion and convolutional neural network (CNN) is proposed by Tang et al. [49]. They calculate features and fuse them into a feature map to describe the state of the network. The CNN model is used to distinguish and detect feature maps including LDOS attacks. Experiments are carried out on NS2 simulation platform and test-bed and results show that the proposed method can effectively detect LDOS attacks with accuracy of 97.1%.

3.2.3. Recurrent Neural Network Based Methods for Attack Detection. Since the output of DNN and CNN only considers the influence of the current input without considering information from the previous and future time, they could achieve significant performance on the classification or recognition tasks without time-varying characteristics. Involving time-dependent data, RNN is proposed as a special

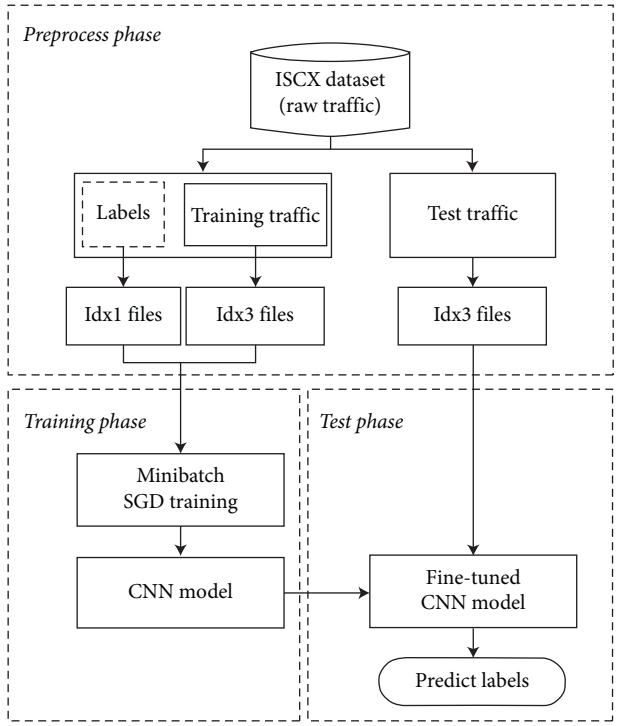


Figure 6: Workflow of the traffic analysis approach proposed by Wang et al. [45], which consists of three parts: preprocess, training, and testing phase.

category of neural network structures, which is designed with "memory" function to maintain previous content. In fact, such design feature coincides with the idea that "human cognition is based on the past experience and memory." RNN is thus good at dealing with time-series information. However, there are still some problems in structure design of RNN like gradient disappearance or gradient explosion, which leads failure to remember or model long-time dependence. Therefore, researchers develop LSTM and GRU with gates design and memory cell, which successfully keep long-time relationship unforgotten by passing through important components of information flow.

Early, Staudemeyer [50] proposes to consider time-series characteristics of known malicious behavior and network traffic, which may improve accuracy performance of attacks detection algorithms. To confirm this, they implement LSTM for intrusion detection based on excellent property of LSTM to model long-time dependant relationship. They design a four-memory blocks network, each of which contains two cells. The network is capable of keeping balance in both computational cost and detection performance. Their experimental results indicate that the proposed LSTM model is better than previously published methods since LSTM could learn to backtrack and correlate continuous connection records in a time-varying manner.

Later, Krishnan and Raajan [51] apply RNN to perform task of attack classification, where their anticipated model is constructed as a sawy self-erudition based Intrusion Detection System by RNN structure. During the experiments, their proposed intrusion detection system could filter

attacks, but fail to identify false positives. Comparing with the baseline methods, their proposed method has improved in measurements, such as classification accuracy and time-consuming.

Similarly, Yin et al. [52] explore utilizing RNN for intrusion detection named RNN-IDS, where they evaluate RNN-IDS with forms of binary classification and multiclass classification. In fact, RNN model has one-way information flow from the first units to the hidden, also from the previous hidden unit to the current one, where the hidden units could be regarded as storage units to store end-to-end and useful information for classification. They have tested whether the parameters, such as number of the neurons, have impact on the RNN-IDS using NSL-KDD dataset. When comparing with previous works such as ANN, random forest, and SVM, RNN-IDS has an advantage in classification performance with high accuracy.

Since LSTM solves the long-term dependency problem and overcomes the vanishing gradient drop during training, Kim et al. [54] apply LSTM architecture for intrusion detection, where the size of hidden layer and the learning rate are settled as 80 and 0.01 after experiments. Comparing with Staudemeyer [50], the constructed LSTM model has a higher false detection rate when training with the KDDCup 99 dataset. Following the trend of applying LSTM on attack detection, Le et al. [55] build a LSTM classifier to detect intrusion as well. They aim is to find the most suitable optimizer for gradient descent optimization of LSTM, where they compare six widely used optimization methods, that is, Adagrad, Adadelta, RMSprop, Adam, Adamax, and Nadam, and find the most effective one is LSTM with Nadam optimizer.

To reduce high false alarm rate achieved by the former methods, a system-call analysis method is proposed by Kim et al. [53], which is developed for anomaly-based host intrusion detection system. As shown in Figure 7, their method consists of two modules: the front-end module, that is, system call language models, which is used to model time-varying characteristics of system calls with LSTM structure in various environments, and the back-end module which is used to predict exceptions based on information passing from the front-end module by a set of ensemble and threshold-based classifiers.

GRU is a variant of LSTM, in which softmax function is used as the final output layer. Moreover, GRU uses cross-entropy function to calculate its losses. Based on GRU structure, Agarap [56] proposes a novel network for binary classification in the attack detection field, which regards a total of 21 features as model inputs. Specifically, linear support vector machine (SVM) is introduced to replace softmax function of the proposed GRU model, which could achieve relatively better effects than the traditional GRU-softmax network on public datasets, due to fast convergence and better ability in classification.

3.3. Other Deep Learning Methods for Attack Detection. In this subsection, we aim to emphasize on the hybrid category of methods on attack detection, which are designed with the idea of integrating advantages of different deep learning structures.

Early in 2015, Li et al. [57] apply a AE and DNN based hybrid deep learning method for malicious code detection. Specifically, they adopt AE to reduce dimensions of original data and focus on the main and important features. Afterward, they use a DBN-based learning model to do the detection of malicious code, which consists of multilayer RBM and a layer of BPNN. Defining each layer of RBM as unsupervised trained and BP as supervised trained, their optimal hybrid model is finally obtained by fine-tuning the whole network. Experiments show that detection accuracy of their hybrid network is higher than other previous DBN-based networks.

Later in 2017, Ludwig [58] employs an ensemble network to classify various types of attacks. In fact, the neural network learning classifies targets with multiple classifiers and merges their results to form robust outputs. To distinguish between normal and abnormal behaviors, their proposed method fuses AE, BNN, DNN, and extreme learning machine for better performance. Their proposed ensemble method brings promising results, which achieve more accurate performance than utilizing single classifier for detection task.

Following the idea of fusing classifiers to obtain better results, Li et al. [59] propose an ensemble structure to enhance the robustness of neural networks for malware detection in 2018; the network is shown in Figure 8. More specifically, a group of neural networks are trained in the training stage and each classifier keeps its counter such as input conversion and semantic preservation. In the test stage, the labeled samples are determined by voting of different classifiers. Their proposed ensemble framework is applied to the challenge of AICS 2019 and has received a good performance in both accuracy and recall.

In order to detect network attacks effectively, Liu et al. [60] propose an end-to-end detection method in 2019. Based on the deep learning model, the author proposes two payload classification models: PL-CNN and PL-RNN. The model learns feature representation from the original payload without feature engineering and end-to-end detection. At the same time, they design a data preprocessing method, which can keep enough information while keeping efficiency. The accuracy of the proposed methods is 99.36% and 99.98%, respectively, when applied to DARPA 1998 dataset. The proposed methods support the use of network data flow for effective end-to-end attack detection, so as to solve the practical problem.

Most recently in 2019, Zhang et al. [61] do not design the characteristics of the flow but directly extract the original data information for analysis. At the same time of learning the temporal and spatial characteristics of flow, a new network intrusion detection model called deep network is proposed, which integrates the improved leNet-5 and LSTM neural network structure. The CICIDS2017 dataset and the CTU dataset are used to evaluate the performance of the network. The amount of traffic is large, and the type of attack is relatively new. The experimental results show that the performance of the network model is better than other

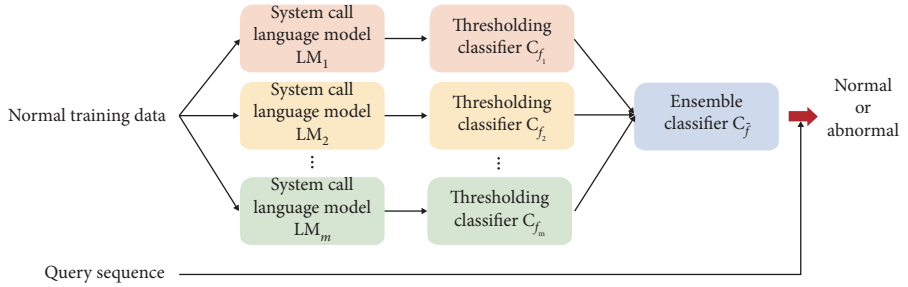


Figure 7: Structure design of Kim et al. [53] for intrusion detection system.

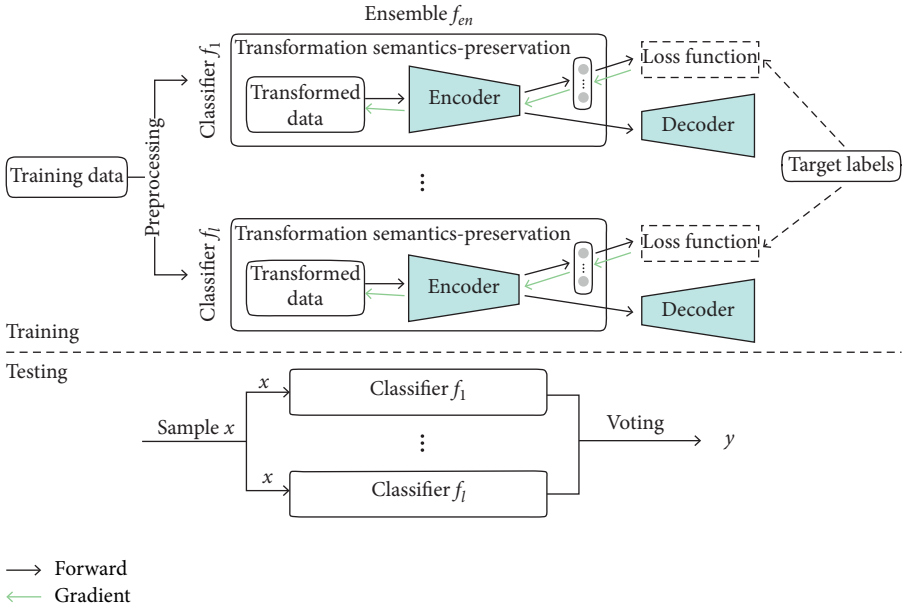


Figure 8: Workflow of the hybrid model proposed by [59].

network intrusion detection models, and it can achieve the best detection accuracy.

4. Comparisons and Analysis

4.1. Public Datasets. Many public datasets are popular to prove and compare efficiency and effectiveness among different attack detection methods. Among them, we list two famous benchmark datasets, that is, KDDCup 99 and NSL-KDD, which are widely used in the academic research to evaluate the ability to detect attacks.

4.1.1. KDDCup 99 Dataset. Despite the fact that there exist some drawbacks like containing a great deal of redundant training and testing data, KDDCup 99 dataset is famous in the field of cybersecurity. It includes both labeled training data and unlabeled test data, which correspond to seven and two weeks of data originated from DARPA′98 IDS evaluation program [62].

Five categories of labels are contained in the dataset which are normal, DoS, Probe, R2L and U2R, that is, short for DoS, Probe, R2L, and U2R, where normal refers to normal traffic instances, Dos is an attack in which the attacker tries to make the target machine stop providing service or resource access to system, Probe represents surveillance and probing, and R2L refers to the unauthorized access while there is an illegal access from the remote machine to local one and represents that there is an unauthorized access to local superuser privileges by local unprivileged user. In Table 1, we display 22 different attacks in training and test data, which could be categorized into these four attack types.

In KDDCup 99 dataset, each record has 41 features in total including basic features, content features, and traffic features as shown in Table 2, where the basic features are obtained from TCP/IP connections including basic characteristics of connection. The content features are extracted from data content, which can be used in the detection of U2R and R2L attacks, which are usually hidden in the packets data without abnormal appearance in single packet and normal connection. Meanwhile, traffic features refer to accumulated values in a time window with 100 connections. It is noted that 7 features and 34 features are symbolic and continuous in data type, respectively.

4.1.2. NSL-KDD Dataset. NSL-KDD is famous as a new development of KDDCup 99 dataset, which comes out to

Table 1: Category of 22 different attacks contained by KDDCup 99.

Class label	Attack name
DoS	back, land, neptune, pod, smurf, teardrop.
Probe	ipsweep, nmap, portsweep, satan.
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster.
U2R	buffer_overflow, loadmodule, perl, rootkit.

Table 2: Feature set for each instance in KDDCup 99 dataset.

No.	Features	Types
1	Duration	Continuous
2	protocol_type	Symbolic
3	Service	Symbolic
4	Flag	Symbolic
5	src_bytes	Continuous
6	dst_bytes	Continuous
7	Land	Symbolic
8	wrong_fragment	Continuous
9	Urgent	Continuous
10	Hot	Continuous
11	num_failed_logins	Continuous
12	logged_in	Symbolic
13	num_compromised	Continuous
14	root_shell	Continuous
15	su_attempted	Continuous
16	num_root	Continuous
17	num_file_creations	Continuous
18	num_shells	Continuous
19	num_access_files	Continuous
20	num_outbound_cmds	Continuous
21	is_hosts_login	Symbolic
22	is_guest_login	Symbolic
23	Count	Continuous
24	srv_count	Continuous
25	serror_rate	Continuous
26	srv_serror_rate	Continuous
27	rerror_rate	Continuous
28	srv_rerror_rate	Continuous
29	same_srv_rate	Continuous
30	diff_srv_rate	Continuous
31	drv_diff_host_rate	Continuous
32	dst_host_count	Continuous
33	dst_host_srv_count	Continuous
34	dst_host_same_srv_count	Continuous
35	dst_host_diff_srv_rate	Continuous
36	dst_host_same_src_port_count	Continuous
37	dst_host_srv_diff_host_rate	Continuous
38	dst_host_serror_rate	Continuous
39	dst_host_srv_serror_rate	Continuous
40	dst_host_serror_rate	Continuous
41	dst_host_srv_rerror_rate	Continuous

reduce shortcomings of the previous dataset. Specifically, it not only removes redundant data from the training and test data to achieve more accurate detection rate but also officially sets the number of records in both training and test data. Moreover, different difficulty level group has different number of records, which is inversely proportional to the percentage of that in the primary KDD dataset. Hence, evaluations and comparisons among different learning technologies become more effective and obvious.

NSL-KDD and KDDCup 99 dataset are similar in structure, where both of them are divided into four attack types as mentioned before. NSL-KDD dataset is divided into two parts: KDDTrain+ and KDDTest+, where we show the specific numbers corresponding to each attack type in Table 3. It is noted that there are 17 attack types in KDDTest+, which do not appear in KDDTrain+. This interesting setting makes NSL-KDD more challenging than KDDCup 99 dataset, which imitates real-life network environment with unknown attacks. We believe only these learning methods built on realistic theoretical basis, that is, analyzing inherent characteristics of attack behaviors, would achieve promising results on NSL-KDD.

4.2. Measurements. In this subsection, we describe 7 measurements including accuracy (ACC), precision (PR), true positive rate (TPR), recall (RE), false positive rate (FPR), true negative rate (TNR), and F1-score. Firstly, we define several items, where true positive (TP) and false negative (FN) refer to attack data correctly classified or not, respectively, and false positive (FP) and true negative (TN) are normal data which are classified as normal or attack, respectively. Afterwards, we define measurements as follows:

$$\begin{aligned}
\mathrm{ACC} &= \frac{(\mathrm{TP}+\mathrm{TN})}{(\mathrm{TP}+\mathrm{FN}+\mathrm{TN}+\mathrm{FP})},\\
\mathrm{PR} &= \frac{\mathrm{TP}}{(\mathrm{TP}+\mathrm{FP})},\\
\mathrm{RE} &= \frac{\mathrm{TP}}{(\mathrm{TP}+\mathrm{FN})},\\
\mathrm{FNR} &= \frac{\mathrm{FN}}{(\mathrm{TP}+\mathrm{FN})},\\
\mathrm{FPR} &= \frac{\mathrm{FP}}{(\mathrm{FP}+\mathrm{TN})},\\
\mathrm{TNR} &= \frac{\mathrm{TN}}{(\mathrm{TN}+\mathrm{FP})},\\
\mathrm{FS} &= \frac{(2*\mathrm{PR}*\mathrm{RE})}{(\mathrm{PR}+\mathrm{RE})},
\end{aligned} \tag{1}$$

where ACC shows the proportion of the amount of data that are correctly classified to whole data, PR calculates the proportion of the amount of attack data that are correctly classified to all attack data representing how many attacks predicted are actual attacks, TPR or RE shows the proportion of predicted attacks to all attacks, FNR estimates the percentage of the number of misclassified normal data to all

Table 3: Records distribution in training and test data [63].

Class	KDDTrain+	KDDTest+
Dos	45927	74588
Probe	11656	2421
R2L	995	2754
U2R	52	200

normal data, FPR called FAR measures the proportion of the benign events that are incorrectly classified as attack, TNR is recognized as the proportion of attack data that are correctly classified to the whole attack data, and F1-score is the weighted average of PR and RE and representing balance performance in both precision and recall.

4.3. Comparisons and Performance Analysis. In Table 4, we offer detailed statics on attack detection results achieved by various methods listed in Section 3, where most of the listed deep learning methods are designed to perform network intrusion detection and malware detection. Among quantity of measurements, we select accuracy, precision, F1-score, and FPR as evaluations, since most of the listed methods use these measurements for experiments. We must emphasize that there exist imbalances in performance comparisons since different authors adopt different datasets, measurements, and settings. However, Table 4 can still provide much information by roughly comparing different deep learning methods for attack detection.

From Table 4, we could notice that the mean performances of different categories of attack detection methods are variant. In the authors' opinion, DBN, LSTM, CNN, and AE achieve the detection performance in descending order. Meanwhile, hybrid methods are inconsistent, since their performances are highly related with ensemble classifiers. DBN is the highest in performance, due to its inherent property of multiple layers in dealing with quantity of unlabeled data. LSTM may achieve higher performance than CNN by involving temporal property for more precise modeling. AE may suffer from large unlabeled data without enough prior knowledge or enough layers to describe the complexity embedded.

Essentially, it is interesting to point out that RBMs and AEs are popular in intrusion detection because we can pretrain the RBMs and AEs with unlabeled data and fine-tune with only a small number of labeled data. Regarding ACC values achieve by listed methods as the first evaluation index due to its completeness, we can find the best performance achieved by attack detection methods on KDDCup 99 dataset, that is, 99.8% achieved by Kim et al. [53], is larger than that on NSL-KDD dataset, that is, 98.3% achieved by Javaid et al. [29], which proves that NSL-KDD dataset is much more difficult than KDDCup 99 dataset due to settings of unknown instances in testing dataset. Another interesting point is that all CNN-based methods abandon the usage of KDDCup 99 and NSL-KDD datasets since their small number of samples could not support showing distinguished power of CNN for generating feature descriptors with abundant information. Meanwhile, other deep learning methods, especially unsupervised learning methods, could bare the shortage of sufficient training samples.

We can observe that performance of AE-based methods is uneven, where most of the improved AE-based methods obviously perform better than traditional AE-based methods. This is due to the fact that the structure of AE might lose important information during compression process. Meanwhile, improved AE could better capture important and informative parts of input data with additional designs. Similarly, LSTM-based and GRU-based methods outperform RNN-based methods, due to their features in structure design of gates and memory cells. In fact, such intelligent designs bring advantage of capability of maintaining long-term information, thus better modeling long-time relationship.

Due to the large number of DBN- and RNN-based (e.g., LSTM and GRU) methods for attack detection proposed by researchers, we would like to regard DBN- and RNN-based methods as typical unsupervised and supervised algorithms, respectively, where we further compare them to show the advantages and disadvantages of both groups.

Essentially, RNN could remember information of the last several moments and then apply it in the calculation for the current unit, which introduces temporal information to help more accurate classification. However, RNN can be powerful structure with sufficient training instances, where attack data especially those unknown attacks are hard to be achieved. Meanwhile, DBN is capable of automatically discovering feature pattern from input data. Moreover, the unsupervised DBN network is less likely to be overfitting than those supervised methods due to its pretraining procedure, where DBN could learn inherent descriptions on abnormal behaviors or attacks by learning from unlabeled data. This feature of generated ability makes DBN, that is, a typical unsupervised learning method, fit with real environment of network security. Last but not least, DBN is easy to be trained, fast to be converged, and low in running time, due to less hidden layers compared with deep structures of CNN or so. Therefore, we think unsupervised learning methods could produce better classification results than supervised learning methods, especially when facing small, imbalanced, or redundant dataset.

5. Summary

Deep learning uses cascaded layers in a hierarchy structure to perform data processing, which results in significant results in domains of unsupervised feature learning and pattern recognition. Inspired by performance of deep learning methods, we believe deep learning is important for field of network security, so as to review the current deep learning methods for attack detection. We analyze recent methods, classify them according to different deep learning techniques, and compress the performance of the most representative methods.

Over the past few years, research on how to apply deep learning methods on attack detection has made a great progress. However, many problems still exist. Firstly, it is challenging to modify deep learning methods as real-time

Table 4: Quantitative evaluation of listed attack detection methods using different deep learning structures, where ID, MD, and TI represent network intrusion detection, malware detection, and traffic identification, respectively.

DL	Method	Usage	Dataset	ACC (%)	PR (%)	FPR (%)	FS
Convolutional AE	Yu et al. [26]	ID	CTU-UNB	—	98.44	—	0.980
Sparse AE	Javaid et al. [29]	ID	NSL-KDD	98.30	—	—	0.990
AE	Pamartzivanos et al. [30]	ID	KDDCup 99	77.99	80.00	—	—
SAE	Farahnakian and Heikkonen [28]	ID	KDDCup 99	94.71	94.53	0.42	—
AE	Shone et al. [17]	ID	NSL-KDD	89.22	92.97	10.78	0.910
Sparse AE	Shone et al. [17]	ID	KDDCup 99	97.85	99.99	2.15	0.980
AE	Aygun and Yavuz [64]	ID	NSL-KDD	93.62	91.39	—	0.938
Denoising AE	Aygun and Yavuz [64]	ID	NSL-KDD	94.35	94.26	—	0.940
Sparse AE	Gharic et al. [65]	ID	NSL-KDD	96.45	95.56	—	0.965
AE	Yousefi-Azar et al. [27]	ID, MD	NSL-KDD	83.34	—	—	—
DBN	Gao et al. [31]	ID	KDDCup 99	93.49	92.33	0.76	—
DBN	Ding et al. [32]	MD	Netflow	96.10	—	—	—
DBN	Qu et al. [66]	ID	NSL-KDD	95.25	—	—	—
DBN	Tan et al. [33]	ID	Netflow	97.60	—	0.90	—
DBN	Alom et al. [34]	ID	40% NSL-KDD	97.50	—	—	—
DBN	Zhao et al. [35]	ID	KDDCup 99	99.14	93.25	0.62	—
DBN	Alrawashdeh and Purdy [36]	ID	10% KDDCup 99	97.90	97.81	2.10	0.975
DNN	Tang et al. [40]	ID	NSL-KDD	91.70	83.00	—	—
DNN	Vinayakumar et al. [18]	ID, MD	KDDCup 99	93.00	99.00	0.95	
DNN	Wang et al. [42]	ID	KDDCup 99	95.45	—	—	—
CNN	Kolosnjaji et al. [43]	MD	Netflow	—	93.00	—	0.920
CNN	Saxe and Berlin [19]	MD	Netflow	92.00	—	0.10	—
CNN	Wang et al. [45]	ID	ISCX	—	97.30	—	0.960
CNN	Wang et al. [47]	TI	Netflow	99.41	—	—	—
CNN	Tang et al. [49]	ID	NS2 simulation	97.1	—	—	—
CNN	Yang and Wang [48]	ID	KDDCup 99	95.36	95.55	0.76	0.930
LSTM	Staudemyer [50]	ID	10% KDDCup 99	93.85	—	1.62	—
RNN	Krishnan and Raajan [51]	ID	KDDCup 99	77.55	84.60	—	0.730
RNN	Yin et al. [52]	ID	NSL-KDD	83.28	—	—	—
LSTM	Kim et al. [54]	ID	10% KDDCup 99	96.93	98.80	10.00	—
LSTM	Le et al. [55]	ID	KDDCup 99	97.54	98.95	9.98	—
LSTM	Kim et al. [53]	ID	KDDCup 99	99.80	—	5.50	—
GRU	Agarap [56]	ID	Netflow	84.15	—	—	—
Ensemble	Ludwig [58]	ID	NSL-KDD	92.50	93.00	0.92	—
AE, DBN	Li et al. [57]	ID	KDDCup 99	92.10	—	1.58	—
DCNN	Naseer et al. [67]	ID	NSL-KDD	85.00	—	—	0.980
PL-CNN	Liu et al. [60]	ID	DARPA1998	99.36	90.56	—	0.910
PL-RNN	Liu et al. [60]	ID	DARPA1998	99.98	99.98	—	0.990

classifiers for attack detection. In most of the previous works, they only reduce feature dimension for less computation cost during phase of feature extraction. Secondly, most of the deep learning techniques are appropriate for analysis of image and pattern recognition. Thus, how to conduct the classification of network traffic reasonably with deep learning techniques will be an interesting issue. Thirdly, with more data involving the experiments, the classification results will be better [68]. However, most of the attack detection problems are short of sufficient data. Therefore, combining supervised and unsupervised learning may provide better performance, which has been proved by many trials. Moreover, with the development of IoT [69], fog, cloud [70], and big data technologies, how to involve them to help improve effectiveness of attack detection methods using deep learning remains an open and interesting question. According to the above analysis, we hold a belief that this overview is a benefit for those who have ideas to improve the performance of attack detection in terms of accuracy; our review will provide guidance and dictionaries for further research in this field.

Data Availability

The data used to support the findings of this study were supplied by Dabao Wei under license and so cannot be made freely available. Requests for access to these data should be made to Yirui Wu (wuyirui@hhu.edu.cn).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by National Key R&D Program of China under Grant 2018YFC0407901, the Fundamental Research Funds for the Central Universities under Grant

B200202177, the Natural Science Foundation of China under Grant 61702160, and the Natural Science Foundation of Jiangsu Province under Grant BK20170892.

References

[1] S. Aftergood, "Cybersecurity: the cold war online," *Nature*, vol. 547, no. 7661, pp. 30-31, 2017.

[2] M. A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, and M. Guizani, "A survey of machine and deep learning methods for internet of things (iot) security," 2018, http://arxiv.org/abs/ 11023.

[3] A. Aleesa, B. Zaidan, 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 and Applications*, pp. 1–32, Springer, Berlin, Germany, 2019.

[4] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cyber security," in *Proceedings of 2018 10th International Conference on Cyber Conflict (CyCon)*, IEEE, Tallinn, Estonia, pp. 371–390, June 2018.

[5] D. Berman, A. Buczak, J. Chavis, and C. Corbett, "A survey of deep learning methods for cyber security," *Information*, vol. 10, no. 4, p. 122, 2019.

[6] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[7] C. S. Wickramasinghe, D. L. Marino, K. Amarasinghe, and M. Manic, "Generalization of deep learning for cyber-physical system security: a survey," in *Proceedings of IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, IEEE, Washington, DC, USA, pp. 745–751, October 2018.

[8] Y. Xin, L. Kong, Z. Liu et al., "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.

[9] X. Xu, C. He, Z. Xu, L. Qi, S. Wan, and M. Z. A. Bhuiyan, "Joint optimization of offloading utility and privacy for edge computing enabled iot," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2622–2629, 2020.

[10] X. Xu, Q. Liu, X. Zhang, J. Zhang, L. Qi, and W. Dou, "A blockchain-powered crowdsourcing method with privacy preservation in mobile environment," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1407–1419, 2019.

[11] X. Xu, X. Liu, Z. Xu, F. Dai, X. Zhang, and L. Qi, "Trust-oriented iot service placement for smart cities in edge computing," *IEEE Internet of Things Journal*, vol. 7, 2019.

[12] X. Xu, Y. Chen, X. Zhang, Q. Liu, X. Liu, and L. Qi, *A Blockchain-Based Computation Offloading Method for Edge Computing in 5g Networks*, John and Wiley, Hoboken, NJ, USA, 2019.

[13] C. Wang, Z. Chen, K. Shang, and H. Wu, "Label-removed generative adversarial networks incorporating with k-means," *Neurocomputing*, vol. 361, pp. 126–136, 2019.

[14] T. Meng, K. Wolter, H. Wu, and Q. Wang, "A secure and cost-efficient offloading policy for mobile cloud computing against timing attacks," *Pervasive and Mobile Computing*, vol. 45, pp. 4–18, 2018.

[15] X. Li and H. Wu, "Spatio-emporal representation with deepneural recurrent network in MIMO CSI feedback," CoRRabs/1908.07934, 2019.

[16] R. Vinayakumar, K. Soman, and P. Poornachandran, "Evaluating effectiveness of shallow and deep networks to intrusion detection system," in *Proceedings of 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, Udupi, India, pp. 1282–1289, September 2017.

[17] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[18] 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.

[19] J. Saxe and K. Berlin, "expose: a character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys," 2017, http://arxiv.org/abs/ 1702.08568.

[20] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in *Proceedings of 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1916–1920, Queensland, Australia, April 2015.

[21] Z. Feng, C. Shuo, and W. Xiaochuan, "Classification for dga-based malicious domain names with deep learning architectures," in *Proceedings of 2017 Second International Conference on Applied Mathematics and Information Technology*, London, UK, January 2017.

[22] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," 2016, http://arxiv.org/abs/ 1611.00791.

[23] M. Z. Alom, T. M. Taha, C. Yakopcic et al., "The history began from alexnet: a comprehensive survey on deep learning approaches,"2018 pages, CoRR abs/1803.01164.

[24] E. Aminanto and K. Kim, "Deep learning in intrusion detection system: an overview," in *Proceedings of 2016 International Research Conference on Engineering and Technology (2016 IRCET)*, Higher Education Forum, Seoul, South Korea, January 2016.

[25] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing*, vol. 3, 2014.

[26] Y. Yu, J. Long, and Z. Cai, "Network intrusion detection through stacking dilated convolutional autoencoders," *Security and Communication Networks*, vol. 2017, Article ID 4184196, 10 pages, 2017.

[27] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proceedings of 2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, San Diego, CA, USA, pp. 3854–3861, June 2017.

[28] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *Proceedings of 2018 20th International Conference on Advanced Communication Technology (ICACT)*, IEEE, Chuncheon, South Korea, pp. 178–183, July 2018.

[29] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (formerly BIONETICS)*, pp. 21–26, New York, NY, USA, December 2016.

[30] D. Papamartzivanos, F. Gomez Marmol, and G. Kambourakis, *Introducing Deep Learning Self-Adaptive Misuse Network*

Intrusion Detection Systems, IEEE Access, Piscataway, NJ, USA, 2019.

[31] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," in *Proceedings of 2014 Second International Conference on Advanced Cloud and Big Data*, IEEE, Huangshan, China, pp. 247–252, November 2014.

[32] Y. Ding, S. Chen, and J. Xu, "Application of deep belief networks for opcode based malware detection," in *Proceedings of 2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3901–3908, Vancouver, British, July 2016.

[33] Q. . s. Tan, W. Huang, and Q. Li, "An intrusion detection method based on dbn in ad hoc networks," in *Proceedings of Wireless Communication and Sensor Network: International Conference on Wireless Communication and Sensor Network (WCSN*, World Scientific, Wuhan, China, pp. 477–485, December 2015.

[34] M. Z. Alom, V. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *Proceedings of 2015 National Aerospace and Electronics Conference (NAECON)*, pp. 339–344, Dayton, OH, USA, June 2015.

[35] G. Zhao, C. Zhang, and L. Zheng, "Intrusion detection using deep belief network and probabilistic neural network," in *Proceedings of 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, Taipei, Taiwan, December 2017.

[36] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proceedings of 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 195–200, Anaheim, CA, USA, December 2016.

[37] H. Zhang, Y. Li, Z. Lv, A. K. Sangaiah, and T. Huang, "A real-time and ubiquitous network attack detection based on deep belief network and support vector machine," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 790–799, 2020.

[38] T. Erpek, Y. E. Sagduyu, and Y. Shi, "Deep learning for launching and mitigating wireless jamming attacks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 1, pp. 2–14, 2018.

[39] A. AlEroud and G. Karabatis, "Bypassing detection of url-based phishing attacks using generative adversarial deep neural networks," in *Proceedings of the Sixth International Workshop on Security and Privacy Analytics*, pp. 53–60, New Orleans, LA, USA, March 2020.

[40] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proceedings of 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, IEEE, Reims, France, pp. 258–263, October 2016.

[41] D. Li, R. Baral, T. Li, H. Wang, Q. Li, and S. Xu, "Hashtran-dnn: a framework for enhancing robustness of deep neural networks against adversarial malware samples," 2018, http://arxiv.org/abs/ 1809.06498.

[42] W. Peng, X. Kong, G. Peng, X. Li, and Z. Wang, "Network intrusion detection based on deep learning," in *Proceedings of 2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pp. 431–435, Haikou, China, July 2019.

[43] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Proceedings of Australasian Joint Conference on Artificial Intelligence*, pp. 137–149, Springer, Hobart, Australia, December 2016.

[44] B. Kolosnjaji, G. Eraisha, G. Webster, A. Zarras, and C. Eckert, "Empowering convolutional networks for malware classification and analysis," in *Proceedings of 2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 3838–3845, San Diego, CA, USA, June 2017.

[45] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proceedings of 2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 43–48, Taipei, Taiwan, June 2017.

[46] M. Zhang, B. Xu, S. Bai, S. Lu, and Z. Lin, "A deep learning method to detect web attacks using a specially designed CNN," in *Proceedings of 24th International Conference on Neural Information Processing*, pp. 828–836, Guangzhou, China, November 2017.

[47] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proceedings of 2017 International Conference on Information Networking*, ICOIN), Da Nang, Vietnam, January 2017.

[48] H. Yang and F. Wang, "Wireless network intrusion detection based on improved convolutional neural network," *IEEE Access*, vol. 7, pp. 64366–64374, 2019.

[49] D. Tang, L. Tang, W. Shi, S. Zhan, and Q. Yang, *Mf-cnn: A New Approach for Ldos Attack Detection Based on Multi-Feature Fusion and Cnn. Mobile Networks and Applications*, pp. 1–18, Springer, Berlin, Germany, 2020.

[50] R. C. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," *South African Computer Journal*, vol. 56, no. 1, pp. 136–154, 2015.

[51] R. B. Krishnan and N. Raajan, "An intellectual intrusion detection system model for attacks classification using rnn," *International Journal of Pharmaceutical Technology and Biotechnology*, vol. 8, no. 4, pp. 23157–23164, 2016.

[52] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[53] G. Kim, H. Yi, J. Lee, Y. Paek, and S. Yoon, "Lstm-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems," 2016, http://arxiv.org/abs/ 1611.01726.

[54] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proceedings of 2016 International Conference on Platform Technology and Service (PlatCon)*, pp. 1–5, Jeju, Korea, February 2016.

[55] T. Le, J. Kim, and H. Kim, "An effective intrusion detection classifier using long short-term memory with gradient descent optimization," in *Proceedings of 2017 International Conference on Platform Technology and Service (PlatCon)*, pp. 1–6, Jeju, Korea, February 2017.

[56] A. F. M. Agarap, "A neural network architecture combining gated recurrent unit (gru) and support vector machine (svm) for intrusion detection in network traffic data," in *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, ACM, Macau, China, February 2018.

[57] Y. Li, R. Ma, and R. Jiao, "A hybrid malicious code detection method based on deep learning," *International Journal of Security and Its Applications*, vol. 9, no. 5, pp. 205–216, 2015.

[58] S. A. Ludwig, "Intrusion detection of multiple attack classes using a deep neural net ensemble," in *Proceedings of 2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, Honolulu,HI, USA, November 2017.

[59] D. Li, Q. Li, Y. Ye, and S. Xu, "Enhancing robustness of deep neural networks against adversarial malware samples: principles, framework, and aics'2019 challenge," 2018, http://arxiv.org/abs/ 1812.08108.

[60] H. Liu, B. Lang, M. Liu, and H. Yan, "Cnn and rnn based payload classification methods for attack detection," *Knowledge-Based Systems*, vol. 163, pp. 332–341, 2019.

[61] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, "Network intrusion detection: based on deep hierarchical network and original flow data," *IEEE Access*, vol. 7, pp. 37004–37016, 2019.

[62] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Computer Networks*, vol. 34, no. 4, pp. 579–595, 2000.

[63] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, IEEE, Ottawa, Canada, pp. 1–6, July 2009.

[64] R. C. Aygun and A. G. Yavuz, "Network anomaly detection with stochastically improved autoencoder based models," in *Proceedings of 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 193–198, New York, NY, USA, June 2017.

[65] M. Gharib, B. Mohammadi, S. H. Dastgerdi, and M. Sabokrou, "Autoids: auto-encoder based method for intrusion detection system," 2019, http://arxiv.org/abs/ 1911.03306.

[66] F. Qu, J. Zhang, Z. Shao, and S. Qi, "An intrusion detection model based on deep belief network," in *Proceedings of the 2017 VI International Conference on Network, Communication and Computing*, pp. 97–101, Kunming, China, December 2017.

[67] S. Naseer, Y. Saleem, S. Khalid et al., "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[68] N. Jones, "Computer science: the learning machines," *Nature*, vol. 505, no. 7482, pp. 146–148, 2014.

[69] X. Xu, X. Zhang, H. Gao, Y. Xue, L. Qi, and W. Dou, "Become: blockchain-enabled computation offloading for iot in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4187–4195, 2020.

[70] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou, "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Transactions on Industrial Informatics*, vol. 16, 2019.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 3932584, 12 pages
https://doi.org/10.1155/2020/3932584

Research Article

The Defense of Adversarial Example with Conditional Generative Adversarial Networks

Fangchao Yu,[1] Li Wang,[1] Xianjin Fang,[1] and Youwen Zhang[2]

[1]*School of Computer Science and Engineering, Anhui University of Science and Technology, Huainan 232000, China*
[2]*School of Computer Science and Engineering, Anhui University, Hefei 230601, China*

Correspondence should be addressed to Li Wang; liwang@aust.edu.cn

Received 9 February 2020; Accepted 6 May 2020; Published 25 August 2020

Academic Editor: Xiaolong Xu

Deep neural network approaches have made remarkable progress in many machine learning tasks. However, the latest research indicates that they are vulnerable to adversarial perturbations. An adversary can easily mislead the network models by adding well-designed perturbations to the input. The cause of the adversarial examples is unclear. Therefore, it is challenging to build a defense mechanism. In this paper, we propose an image-to-image translation model to defend against adversarial examples. The proposed model is based on a conditional generative adversarial network, which consists of a generator and a discriminator. The generator is used to eliminate adversarial perturbations in the input. The discriminator is used to distinguish generated data from original clean data to improve the training process. In other words, our approach can map the adversarial images to the clean images, which are then fed to the target deep learning model. The defense mechanism is independent of the target model, and the structure of the framework is universal. A series of experiments conducted on MNIST and CIFAR10 show that the proposed method can defend against multiple types of attacks while maintaining good performance.

1. Introduction

Deep learning [1–5] is a hierarchical machine learning method involving multilevel nonlinear transformations and is good at mining abstract and distributed feature representations from raw data. Deep learning can solve many problems that are considered challenging in machine learning. Recently, driven by the emergence of big data and hardware acceleration, deep learning has made significant progress in numerous machine learning domains, such as computer vision, natural language processing, edge computing [6–10], and services computing [11–13], and promotes the large-scale application of artificial intelligence technology in the real world. While deep learning has achieved great success, its performance and applications are also questioned due to the lack of interpretability [14], which means that we cannot reasonably explain the decisions made by deep learning models. This exposes deep learning-based artificial intelligence applications to potential security risks.

Many types of research have shown that deep learning is threatened by multiple attacks, such as membership inference attack [15, 16] and attribute inference attack [17]. The most serious security threat to deep learning is the adversarial example [18] proposed by Szegedy in 2013. An adversary can add small-magnitude perturbations to inputs, which can easily fool a well-performed deep learning model with few perturbations imperceptible to humans [19]. The disturbed inputs are called adversarial examples, and they make the target model report high confidence in incorrect predictions. Moreover, recent research shows that artificial intelligence applications in the real world can be exposed to adversarial samples [20], for example, attacks in the face recognition system [21] and vision system in autonomous cars [22].

With the in-depth study of adversarial examples, the development of this field mainly presents the following main trends. (1) A growing number of methods for constructing adversarial examples are proposed. According to adversarial

specificity, we can divide these attack methods into targeted attacks and nontargeted attacks. For targeted attacks, the adversary can submit well-designed inputs to the target model and cause maliciously chosen target outputs, such as $R+\mathrm{LLC}$ [23], JSMA [24], EAD [25], and C&W [26]. For nontargeted attacks, the adversary can cause the target model to misclassify well-designed inputs into classes that are different from the ground truth, such as FGSM [27], BIM [20], PGD [28], and DeepFool [29]. Even worse, the robustness of adversarial examples constantly increases, and detection and defense are challenging. (2) The cost of constructing adversarial examples is decreasing. Due to the transferability [30] of the adversarial example, the adversary can successfully launch an attack without background knowledge about the target model. (3) The range of attacks is also expanding. Adversarial examples can also successfully attack different deep learning models such as reinforcement learning models and recurrent neural network models. Moreover, attack scenarios are not limited to the computer vision. The same security risks exist in text [31] and speech [32]. Therefore, building an effective defense mechanism against adversarial examples is crucial in deep learning.

There is no uniform conclusion on the cause of the adversarial examples; thus, building a defense mechanism is challenging. In general, there are two classes of approaches to defend against adversarial examples: (1) making deep neural networks more robust by adjusting learning strategies, such as adversarial training [27, 33] and defensive distillation [34]; (2) detecting adversarial examples or eliminating adversarial noise after deep neural networks are built, such as LID [35], Defense-GAN [36], MagNet [37], and ComDefend [38]. There are some bottlenecks in these defense mechanisms. First, some defense mechanisms are only effective against specific attacks. For example, defensive distillation is effective for gradient-based attacks, and it is defeated by C&W attacks. Second, some methods require large samples and high computational costs, which limit the application scenarios for these defense mechanisms. Third, the difference between the adversarial example and the clean example is small; thus, it is difficult for current detection methods to distinguish them with high confidence. In summary, we hope to find a defense mechanism with good performance on most attacks and low computational cost.

Our work has made some progress toward building a better defense mechanism against adversarial examples in computer vision. The main reason for adversarial examples to mislead the target model is that the added noise changes the characteristics of the original inputs; thus, an intuitive approach is to remove the noise from the adversarial examples and generate a mapping of the adversarial examples to the clean examples. In computer vision, this can be posed as "translating" an input image (adversarial example) into a corresponding output image (clean example). In this paper, we use the framework proposed by Isola et al. [39] as a defense mechanism. Based on conditional adversarial networks (conditional GANs) [40], the framework consists of a generator network to translate the adversarial images to the clean images and a discriminator network to ensure that the generated images are realistic. Our method can effectively eliminate adversarial perturbations and restore the characteristics of the original clean images. The overview of the defense model is shown in Figure 1. The advantages of our method are listed as follows:

(1) The proposed method is a general-purpose defense framework. On the one hand, the defense mechanism processes the input and is model independent, which means that the target model does not need to be retrained. On the other hand, the network structure of the defense framework is based on a general-purpose solution of image-to-image, and we can apply the framework for different scenarios with only a few adjustments.

(2) Our method is simple and easy to use, and it is effective against most commonly considered attack strategies, such as FGSM, DeepFool, JSMA, and CW. Moreover, this defense mechanism shows certain transferability, which means the defense mechanism built for the specific target model is still effective for other models.

The remainder of the paper is as follows. We introduce some related works about adversarial example in Section 2. In Section 3, we review the necessary theories and concepts about adversarial example and conditional GANs. We give a detailed technical development about the framework of the generation and defense of adversarial example in Section 4. Section 5 describes the experimental results, and Section 6 concludes the paper.

2. Related Works

In this section, we introduce the application of GANs in the field of adversarial examples: generating adversarial examples with GANs and defending adversarial examples with GANs.

2.1. Generating Adversarial Examples with GANs. Xiao [41] proposed AdvGAN to generate adversarial examples. AdvGAN takes a clean image x as the input of the generator G and obtains the adversarial images as $x+G(x)$. The adversarial examples generated by AdvGAN perform high attack success rates in both semiwhite box and black-box attacks. Song et al. [42] designed an unrestricted approach to generate adversarial examples with an auxiliary classifier generative adversarial network (AC-GAN) [43]. Different from perturbation-based attacks, this approach constructs adversarial examples entirely from scratch instead of perturbing an existing data point. In addition, the adversary can specify the style of the generated adversarial examples and labels that are misclassified on the target model. Zhao et al. [44] noticed that the adversarial perturbations are often unnatural and not semantically meaningful. He proposed a framework consisting of a WGAN [45] and an inverter. The inverter maps a clean image to random dense vectors z. The generator of the WGAN obtains the $\widetilde{z}$ (perturbing z) as the input. The goal of the generator is to synthesize an image that is as close to the original image as possible. This method can

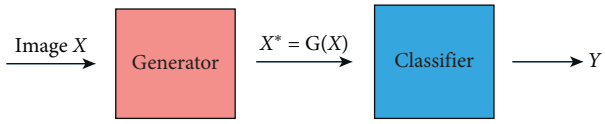


Figure 1: The overview of defense model for the adversarial example proposed in this paper. Between the input images and the classifier, we add a generator, which can eliminate the adversarial perturbations in the input images and map the adversarial examples into clean images.

generate natural and legible adversarial examples that lie on the data manifold. Hu and Tan [46] focused on adversarial examples in traditional security scenarios. They proposed the MalGAN to generate adversarial malware examples, which are able to bypass black-box machine learning-based detection models.

2.2. Defense Adversarial Examples with GANs. Lee et al. [47] introduced a novel adversarial training framework named generative adversarial trainer (GAT). The framework consists of a generator and a classifier. The generator attempts to generate adversarial perturbations that can easily fool the classifier and the classifier attempts to correctly classify both original and generated adversarial images. This approach can improve the robustness of the model and outperforms other adversarial training methods using a fast gradient method. Santhanam and Grnarova [48] proposed *cowboy*, an approach to defend against adversarial attacks with GANs. This work shows that adversarial samples lie outside of the data manifold learned by a GAN that has been trained on the same dataset. They used the discriminator (GAN) to detect adversarial examples and the generator (GAN) to eliminate adversarial perturbations. Samangouei et al. [36] proposed a new framework named Defense-GAN, which leverages the expressive capability of generative models (WGAN) to defend against adversarial examples. Defense-GAN finds a close input to the adversarial examples and sends the input to the generator of WGAN. Then, the generated images are fed to the target model.

3. Background

In this section, we introduce four methods of generating adversarial examples. In addition, GAN and its connection to our method will be discussed.

3.1. Generating Adversarial Example. The main idea of generating adversarial samples is to add appropriate perturbations to the input samples to make the noisy samples as similar to the original input as possible, but mislead the target model. We can briefly describe this process: for a given input image x, the adversary needs to find a minimal perturbation η and craft the noisy example as $x^* = x + \eta$. In recent years, many methods of generating adversarial examples have been proposed. Here, we introduce some of the most well-known attacks.

3.1.1. Fast Gradient Sign Method (FGSM) [27]. Szegedy et al. first introduced adversarial examples against deep neural networks and proposed the method named L-BFGS [18] to generate adversarial examples; however, it was time-consuming and impractical. In 2014, Goodfellow et al. argued that the primary cause of neural networks' vulnerability to adversarial perturbations is their linear nature. Based on this explanation, they proposed a simple and fast method to generate adversarial samples, named fast gradient sign method (FGSM). Let θ be the parameters of a target model, x is the input to the model, y is the label associated with x, and $J(\theta, x, y)$ is the cost function used to train the model. The adversarial sample is generated as

$$x^* = x + \varepsilon sign\left(\nabla_x J(\theta, x, y)\right), \tag{1}$$

where ε is a parameter that determines the perturbation size.

3.1.2. DeepFool [29]. FGSM is simple and effective; however, it causes a large degree of perturbations to inputs. Moosavi-Dezfooli et al. observed that adding noise along the vertical direction of the closest decision boundary to the inputs can ensure that the added perturbation is optimal. They used an iterative method to approximate the perturbation by considering that f is linearized around x_i at each iteration. The minimal perturbation is computed as

$$\begin{aligned} &\text{argmin}_{\eta_i} \|\eta_i\|_2, \\ &\text{s.t.} f(x_i) + \nabla f(x_i)^T \eta_i = 0, \end{aligned} \tag{2}$$

where η_i is the distance to the decision boundary.

3.1.3. Jacobian-Based Saliency Map Attack (JSMA) [24]. The previous two attack methods are both nontargeted attacks. Papernot et al. observed that different input features have different degrees of influence on the output of the target model. If we find that some features correspond to a specific output in the target model, we can make the target model produce a specified type of output by enhancing these features in the inputs. Based on this idea, they proposed a simple iterative method for targeted attack named the Jacobian-based saliency map attack (JSMA). First, the JSMA requires the calculation of the forward derivative, which shows the influence of each input feature on the output. Then, it can generate the adversarial saliency map and use the adversarial saliency map to find the input features that have the greatest impact on the specific output of the target model. Finally, a small perturbation added to the features can fool the neural network.

3.1.4. Carlini and Wagner (C&W) [26]. Carlini et al. proposed a method of generating a more robust adversarial example that can bypass many advanced defense mechanisms. This method treats the adversarial example as a variable, and two conditions need to be met for the attack to succeed. First, the difference between the adversarial example and the corresponding clean sample should be as small as possible. Second, the adversarial example should

make the model classification error rate as high as possible. There are three attacks for the L_0, L_1, and L_2 distance metrics, and we provide a brief description of the L_2 attack:

$$\min_w \left|\frac{1}{2}(\tanh(w)+1)\right|_2 + c \cdot g\left(\frac{1}{2}\tanh(w)+1\right). \tag{3}$$

The loss function g is defined as

$$g(x) = \max\left(\max_{i\neq l'}(Z(x)_i) - Z(x)_{t'} - \kappa\right), \tag{4}$$

where Z denotes the SoftMax function, κ is a constant used to control the confidence (as κ increases, the adversarial examples become more powerful), t is the target label of misclassification, and the constant c can be chosen with binary search.

3.2. Generative Adversarial Networks. Generative adversarial networks (GANs) [49] are a successful framework for generative models and are widely used in many fields [50–52]. A GAN framework forces two networks to compete with each other: a generator G, which attempts to map a sample z (noise distribution $z \sim p_z(z)$ to the data distribution ($x \sim p_{\text{data}}(x)$), and a discriminative model D, which estimates the probability that a sample came from the training data rather than G. The goal of a generator G is to maximize the probability of D making a mistake. Thus, this framework plays a two-player minimax game via the following value function $V(G, D)$:

$$\begin{aligned}\min_G \max_D V(D,G) = {} & E_{x\sim p_{\text{dt}}(x)}[\log D(x)] \\ & + E_{z\sim p_z(z)}[\log(1 - D(G(z)))].\end{aligned} \tag{5}$$

In the competition, both the generator and discriminator will be improved until the discriminator cannot distinguish a generated sample from a data sample.

Mirza and Osindero [40] introduced the conditional version of generative adversarial networks (conditional GAN), and the conditional GAN can be expressed as a mapping from an observed input x and random noise z to y, G: $\{x, z\} \longrightarrow y$. The value function $V(G, D)$ in conditional GAN is as follows:

$$\begin{aligned}\min_G \max_D V(D,G) = {} & E_{x,y}[\log D(x,y)] \\ & + E_{\mathbf{x},\mathbf{z}}[log(1 - D(\mathbf{x}, \mathbf{G}(\mathbf{x},\mathbf{z})))].\end{aligned} \tag{6}$$

With the conditional GAN, it is possible to direct the data generation process and obtain the specified result.

4. Proposed Method

In this section, we introduce the defense mechanism against adversarial examples in detail.

4.1. Motivation. In computer vision, we can consider the attack and defense of adversarial examples as an image-to-image translation process. For the adversary, the goal is to perturb clean images to generate adversarial images. For the defender, the usual idea is to transform the input adversarial images and eliminate the perturbation to restore them to clean images. According to this idea, we can apply some image conversion methods to the field of adversarial examples. In 2018, Isola et al. [39] proposed a generic approach named *pix2pix* to solve image-to-image translation problems and is based on the conditional GAN. They demonstrated that *pix2pix* is effective at reconstructing objects from edge maps and colorizing images, among other tasks. In this paper, we use the same network framework as *pix2pix* to solve the problems in adversarial examples. We use the framework as a defense mechanism to generate a mapping of adversarial images to clean images.

4.2. Framework. The framework of *pix2pix* is based on the conditional GAN. This means that the structure of this framework mainly consists of two parts: a generator and a discriminator. As shown in Figure 2, we introduce the structure of our framework from two aspects.

4.2.1. Generator. We use the structure of U-Net [53] as a generator, which adds skip connections based on the encoder-decoder network. Although there are some minor distinctions in surface appearance between the inputs (adversarial images) and outputs (clean images), the underlying structures of both are the same. Therefore, in the task of image-to-image (adversarial images to clean images), both of them should share the same underlying information. The traditional encoder-decoder generator model lacks the transmission of low-level information, which causes some distortion of the outputs. Therefore, we add skip connections to share underlying information between the inputs and outputs based on the encoder-decoder network, which can ensure that the quality of the converted images is closer to the expected result. Each skip connection simply concatenates all channels at layer i with those at layer $n - i$, where n is the total number of layers.

4.2.2. Discriminator. We use the structure of PatchGAN [39] as a discriminator. The traditional GAN discriminator judges the output as a whole, and it restricts the discriminator to model the high-frequency structure. The PatchGAN maps each input image into $N \times N$ patches via a convolutional network and attempts to determine whether each $N \times N$ patch in an image is real or fake. Then, it averages all responses to provide the ultimate output of the discriminator. In this way, the local features of the generated images can be well constructed.

4.3. Defense Adversarial Example. Figure 2 illustrates the overall architecture of the defense mechanism for the adversarial example. We use paired data (x, y) for training, and each pair of data contains a clean image y and its adversarial image x. Here, the generator G takes the adversarial example x as its input and generates the images $G(x)$. Then, $(x, G(x))$ and (x, y) are sent to the discriminator D, which is used to distinguish the generated data and

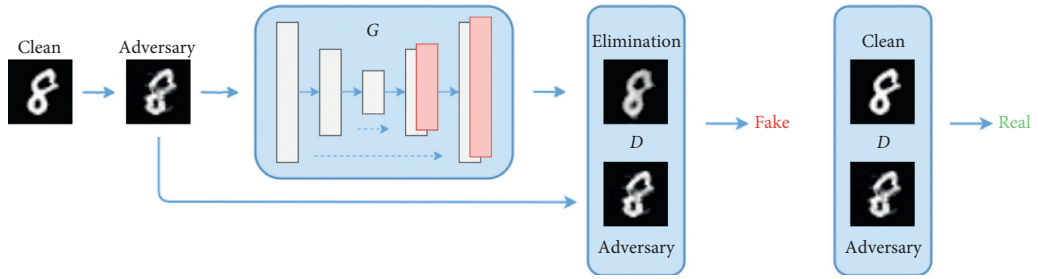


FIGURE 2: The training framework of defense mechanism with the conditional GAN. This framework consists of a generator and a discriminator. The generator takes adversarial images x as input and eliminates perturbations in x. Then, we obtain $G(x)$. The discriminator is used to distinguish the generated data $(x, G(x))$ and the original instance (x, y), where y denotes the clean images.

the original instance. The adversarial loss can be written as follows:

$$\mathscr{L}_{cGAN} = E_{x,y}[\log D(x, y)] + E_x[\log(1 - D(\mathbf{x}, \mathbf{G}(\mathbf{x})))]. \tag{7}$$

The goal of G is to not only fool the discriminator but also be near the ground truth output. Therefore, we add the loss $\mathscr{L}L_1(G)$, which encourages the generated instances $G(x)$ to be close to the clean images y:

$$\mathscr{L}L_1(G) = E_{x,y}[|y - G(x)|_1]. \tag{8}$$

The current objective function is

$$G^* = \arg\min_G \max_D \mathscr{L}_{cGAN}(G, D) + \lambda \mathscr{L}L_1(G), \tag{9}$$

where λ controls the relative importance of $\mathscr{L}L_1(G)$.

As shown in Figures 3 and 4, our defense mechanism can eliminate adversarial perturbations in the images. However, for some complex datasets (such as CIFAR10), although the generated images are close to the original clean images, their performance in the target model f is not satisfactory. To solve this problem, we adjust the objective function. Our core goal is to eliminate the adversarial perturbations in x and make the prediction results of the generated images $G(x)$ close to the prediction results of y in the target model. Therefore, we add the loss function as follows:

$$\mathscr{L}^f_{\text{adv}} = E_x L_f G(x, y). \tag{10}$$

The final objective function is

$$G^* = \arg\min_G \max_D \mathscr{L}_{cGAN}(G, D) + \lambda \mathscr{L}L_1(G) + \mu \mathscr{L}^f_{\text{adv}}, \tag{11}$$

where μ controls the relative importance of $\mathscr{L}^f_{\text{adv}}$.

In general, the loss functions $\mathscr{L}_{cGAN}(G, D)$ and $\lambda \mathscr{L}L_1(G)$ encourage the adversarial data to appear similar to the clean data, while the loss function $\mathscr{L}^f_{\text{adv}}$ improves the prediction accuracy of the generated images on the target model.

5. Experiment

In this section, we evaluate the defense mechanism against adversarial examples. All experiments are based on two datasets: MNIST and CIFAR10.

MNIST (the MNIST used to support the findings of the study is public, and one can find it in http://yann.lecun.com/exdb/mnist/) is a dataset of handwritten digits and consists of 60000 training examples and 10000 testing examples. Each sample consists of 28 × 28 pixels, where each pixel is a grayscale value. For MNIST, we trained two classifiers **Anet** and **Bnet** and used these classifiers as target models to generate adversarial examples and test our approach. The network structure is shown in Table 1. The prediction accuracies of **Anet** and **Bnet** on the test set are 98.96% and 99.74%, respectively.

The CIFAR10 (the CIFAR10 used to support the findings of the study is public, and one can find it in https://www.cs.toronto.edu/~kriz/cifar.html) dataset consists of 60000 32 × 32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. For CIFAR10, we trained two classifiers Resnet (**Rnet**) [54] and DenseNet (**Dnet**) [55] and used these classifiers as target models to generate adversarial examples and test our approach. The prediction accuracy of **Rnet** and **Dnet** on the test set is 93.63% and 95.04%, respectively.

5.1. Implementation Details. We used the adversarial examples generated by the training data and the clean images in the training data as the training set for our framework. All attacks (FGSM, DeepFool, JSMA, and CW) were implemented in *advbox* [56], which is a toolbox used to benchmark deep learning systems' vulnerabilities to adversarial examples. We used the interface provided by *advbox* to generate the adversarial examples. We experimented with $\varepsilon = 0.15$ on MNIST, $\varepsilon = 0.1$ on CIFAR10, and L_2 attacks for CW. For the targeted attacks

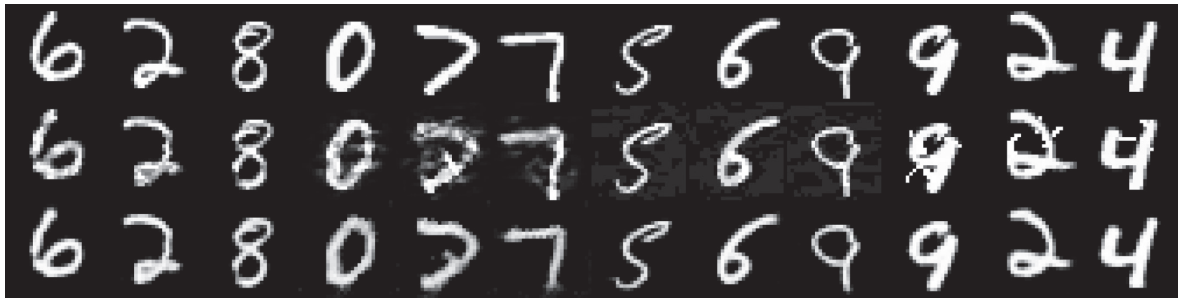


Figure 3: Experimental results of the defense adversarial example in the MNIST dataset. The first line shows clean images. The second line shows an adversarial example. Here, we use four different approaches to generate adversarial (CW, DeepFool, FGSM, and JSMA) images and three images for each approach. The third line shows the images generated by the defense framework.

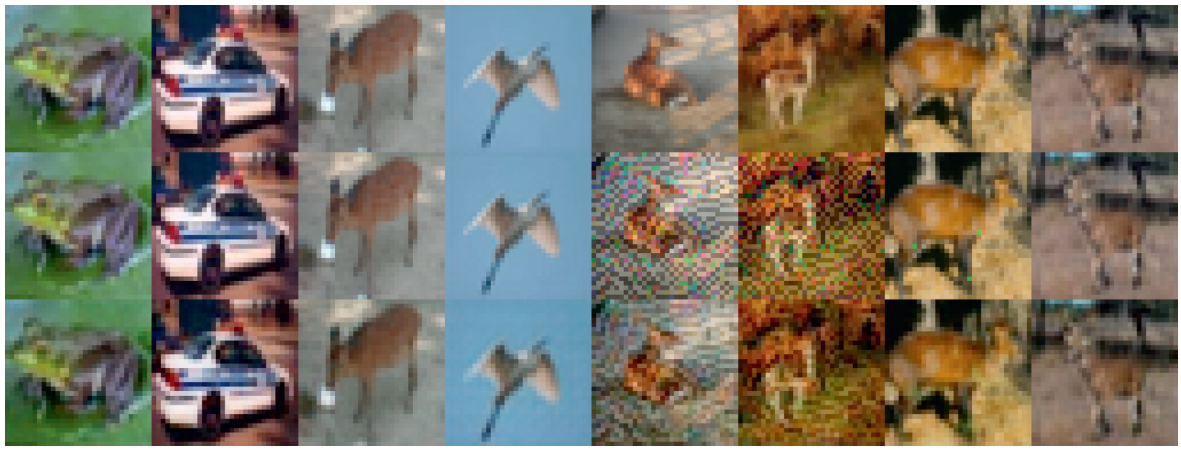


Figure 4: Experimental results of defense adversarial example in the CIFAR10 dataset. The first line shows clean images. The second line shows an adversarial example. Here, we use four different approaches to generate adversarial (CW, DeepFool, FGSM, and JSMA) images and two images for each approach. The third line shows the images generated by the defense framework.

Table 1: The structure of target model.

Anet	Bnet
Conv(64, 5×5, 1) + ReLU	Conv(64, 3×3, 1) + ReLU
Conv(64, 3×3, 2) + ReLU	Conv(64, 3×3, 1) + ReLU
Dropout(0.5)	Maxpooling
FC(128) + ReLU	Conv(64, 3×3, 1) + ReLU
Dropout(0.5)	Conv(64, 3×3, 1) + ReLU
FC(10) + ReLU	Maxpooling
SoftMax	FC(200) + ReLU
	Dropout(0.5)
	FC(200) + ReLU
	Dropout(0.5)
	FC(10) + ReLU
	SoftMax

JSMA and CW, we set a random target label for each sample. The network structure of our framework (include the generator and discriminator) is the same as *pix2pix* [39].

5.2. Defense Adversarial Example. To verify the effectiveness of the defense mechanism, we tested it on two datasets MNIST and CIFAR10. For each dataset, we trained two defense frameworks for different target models. We generated adversarial examples on test data and selected the adversarial examples that successfully attacked in the target model as members of the test set. Therefore, the prediction accuracy of the target model on the test set is 0%. In our defense mechanism, we sent the adversarial examples to a generator that had previously been trained. Then, we took the generated data as input to the target model. Figures 5 and 6 show the prediction accuracy of the target model on the adversarial example under the defense mechanism, where epoch means the number of training iterations. The result indicates that our defense framework can quickly converge during training. For the MNIST dataset, we take epoch = 20 as the final result, as shown in Table 2. Our defense mechanism is effective against different types of attacks. It improves the prediction accuracy of the target models (**Anet** and **Bnet**) on the adversarial sample from 0 to almost 98%. For the CIFAR10 dataset, we take epoch = 40 as the final result, as shown in Table 3. Since the CIFAR10 dataset is much more complicated than the MNIST dataset, it can cause some losses in the denoising process. Therefore, the defensive performance on CIFAR10 is reduced compared to that on MNIST. CW attacks are more robust than other attacks, which means that defending against such attack is more challenging. Our defense mechanism still achieves good performance on CW attacks.

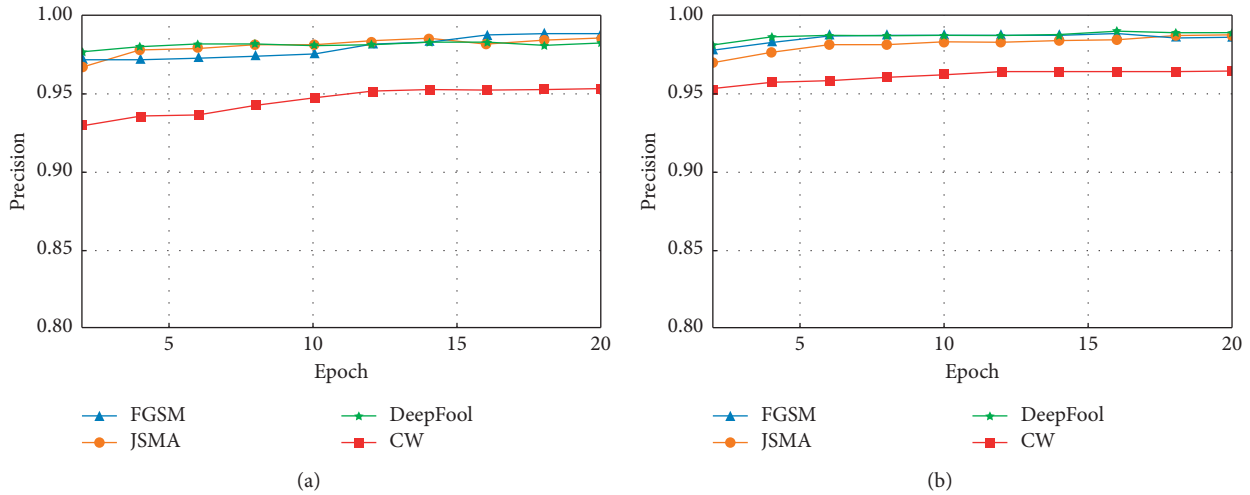


Figure 5: The prediction accuracy of the defense mechanism for the MNIST dataset (the range of epoch is from 2 to 20): the experimental results of the target models (a) Anet and (b) Bnet.

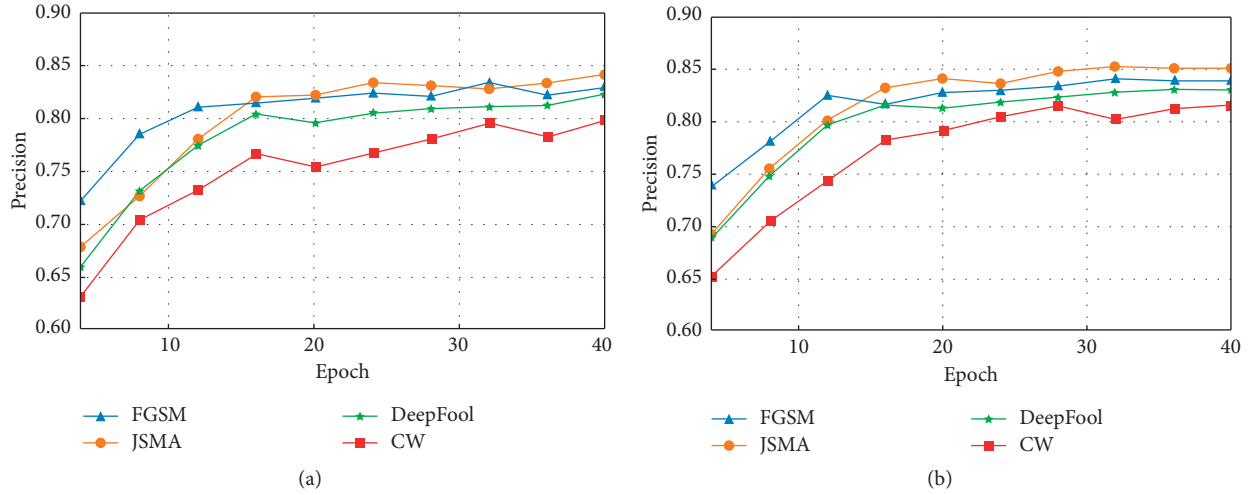


Figure 6: The prediction accuracy of the defense mechanism for CIFAR10 dataset (the range of epoch is from 4 to 40): the experimental results of target models (a) Rnet and (b) Dnet.

Table 2: Defense performance on MNIST (epoch = 20).

Target model	FGSM	DeepFool	JSMA	CW
Anet	98.73	98.44	98.25	95.39
Bnet	98.70	98.68	98.88	96.51

Table 3: Defense performance on CIFAR (epoch = 40).

Target model	FGSM	DeepFool	JSMA	CW
Rnet	82.82	84.15	82.17	79.84
Dnet	84.19	85.10	83.30	81.63

In addition, we compare the adversarial perturbation and defense loss for both the MNIST dataset (epoch = 20) and CIFAR10 dataset (epoch = 40). An adversarial perturbation means average L_1 norm loss between adversarial images and clean images, and the defense loss means an average L_1 norm loss between the generated images and

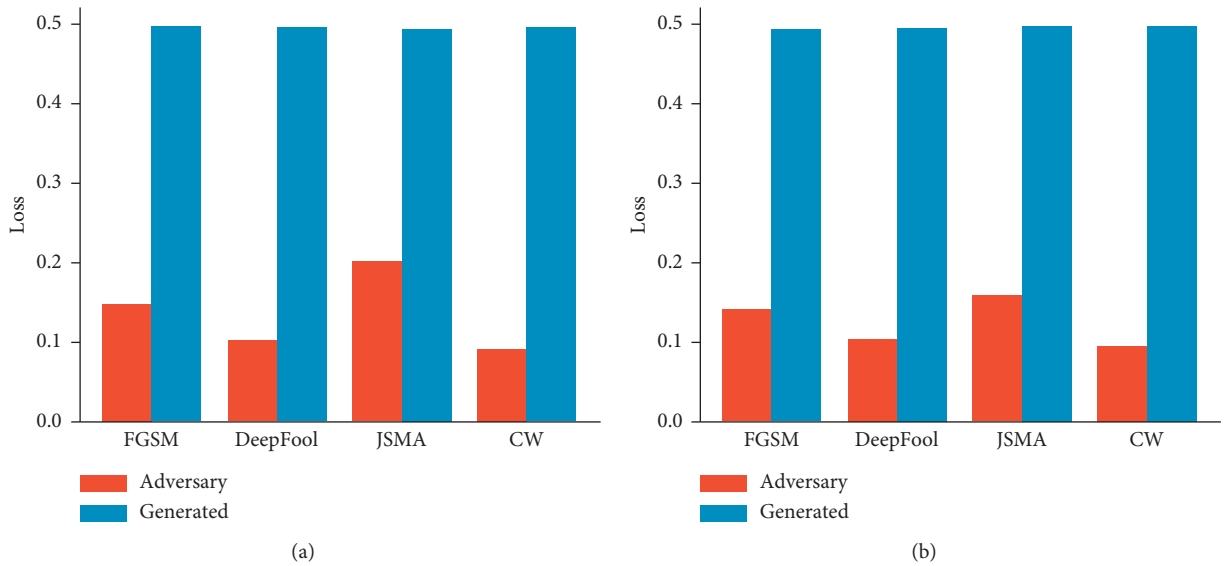


Figure 7: The loss of the defense mechanism on the MNIST dataset: experimental results of target models (a) Anet and (b) Bnet.

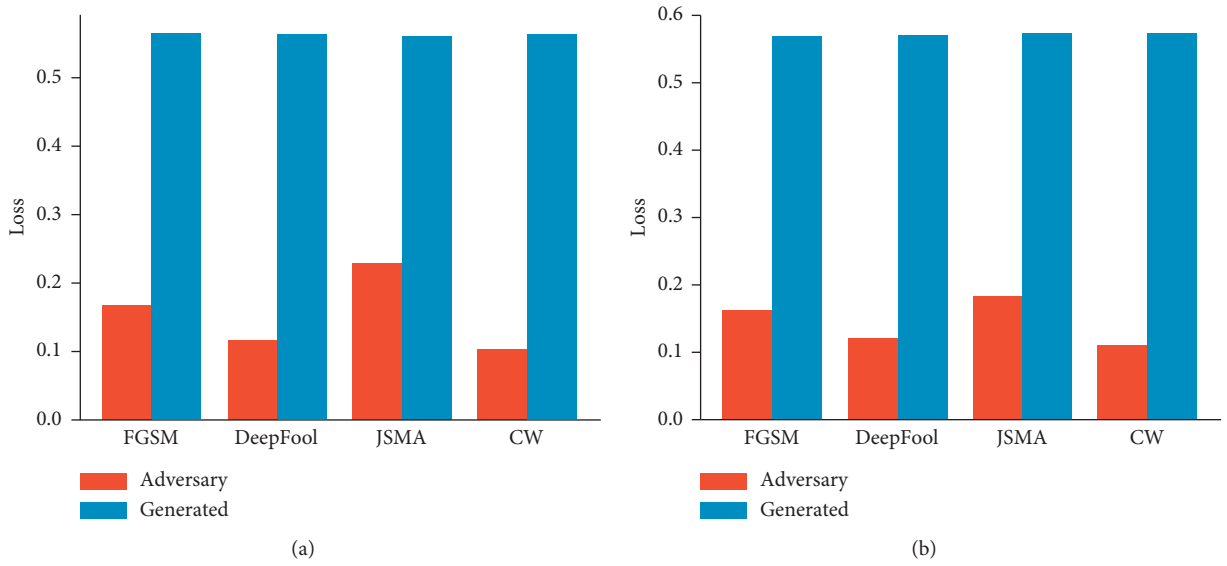


Figure 8: The loss of the defense mechanism on the CIFAR10 dataset: experimental results of target models (a) Rnet and (b) Dnet.

clean images. Since our defense framework consists of U-Net and PatchGAN, their combination enables the generator to restore the details of the original clean data. As shown in Figures 7 and 8, our defense mechanism can control defense losses within a certain range. This ensures the high quality of the generated images and the similarity to the clean images.

5.3. Defense Transferability. In this experiment, we tested the transferability of our defense mechanism. We used the adversarial examples generated by other target models to test the framework trained for the specific target model. Figures 9 and 10 show the trend of the transferability of the prediction accuracy during training. Similar to previous results, in this case, our defense mechanism still achieves a high convergence speed.

For the MNIST dataset and CIFAR10 dataset, we separately took epoch = 20 and epoch = 40 as the final result, and the result is shown in Tables 4 and 5 (**Anet**/**Bnet** means that we use the adversarial examples generated by the target model **Anet** to test the framework trained for the target model **Bnet**). The purpose of our defense mechanism is to restore the original characteristics of the adversarial examples and eliminate their adversarial perturbations. Therefore, our defense framework focuses on adversarial examples, not the target model. The experimental results prove that our method is universal. It can transfer the capabilities learned from the specific target model to other models.

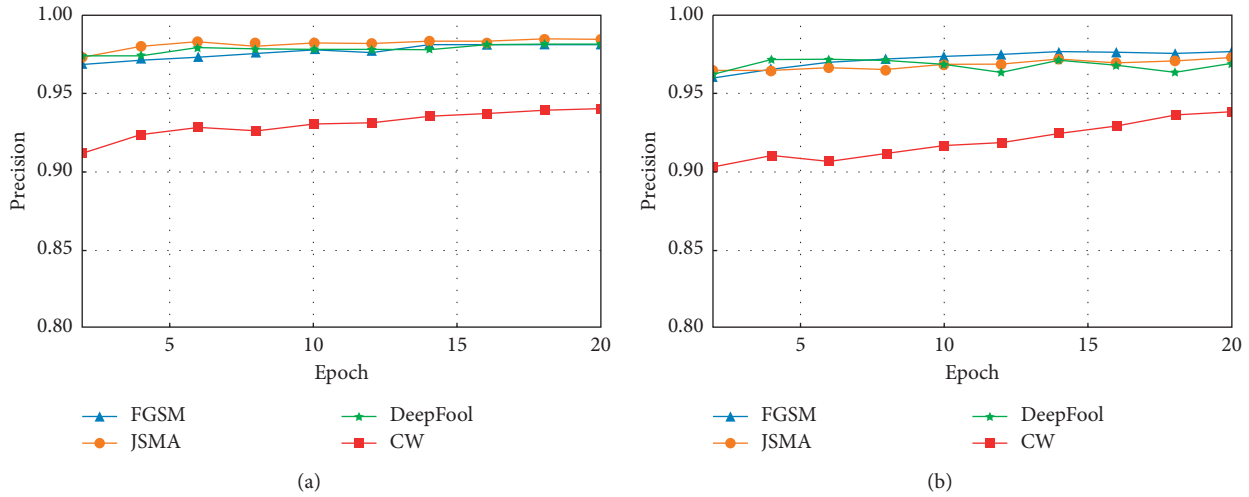


Figure 9: The transferability of the defense mechanism on the MNIST dataset (the range of epoch is from 2 to 20): experimental results of (a) Anet/Bnet and (b) Bnet/Anet.

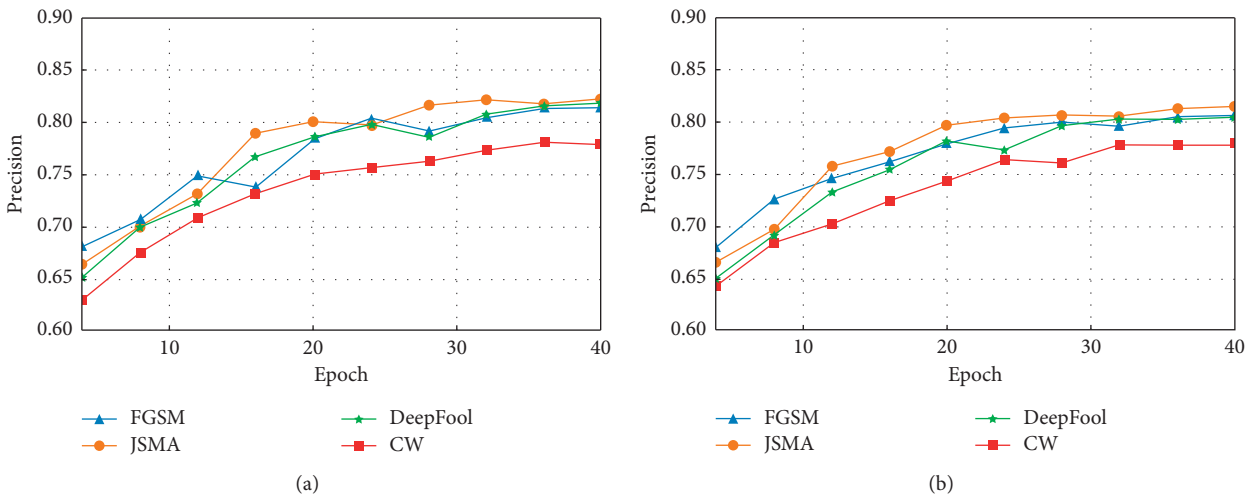


Figure 10: The transferability of the defense mechanism on the CIFAR10 dataset (the range of epoch is from 4 to 40): experimental results of (a) Rnet/Dnet and (b) Dnet/Rnet.

Table 4: Transferability on MNIST (epoch = 20).

Target model	FGSM	DeepFool	JSMA	CW
Anet/Bnet	98.13	98.36	98.06	94.11
Bnet/Anet	97.56	97.24	96.89	93.82

Table 5: Transferability on CIFAR (epoch = 40).

Target model	FGSM (%)	DeepFool (%)	JSMA (%)	CW (%)
Rnet/Dnet	81.45	82.71	81.79	78.16
Dnet/Rnet	80.36	81.36	80.35	78.01

5.4. Comparison with Other Defense Methods. Following the experimental setup in Defense-GAN [36], we compared the proposed method with other defense mechanisms such as Defense-GAN, MagNet [37], and adversarial training [27]. The adversarial training uses the adversarial example as part of the training set to build a more robust model. The magnet consists of a detector and a reformer. The detector is used to detect adversarial examples, and reformer is used to transform adversarial examples into clean examples. Since Defense-GAN is not argued secure on

Table 6: Comparisons with other defense methods.

Attack	Target model	No attack	No defense	Defense-GAN	MagNet	Adv.Tr	Our method
FGSM	A	0.997	0.217	0.988	0.191	0.651	0.986
	B	0.962	0.022	0.956	0.082	0.060	0.958
	C	0.996	0.331	0.989	0.163	0.786	0.987
	D	0.992	0.038	0.980	0.094	0.732	0.983
CW	A	0.997	0.141	0.989	0.038	0.077	0.965
	B	0.962	0.032	0.916	0.034	0.280	0.924
	C	0.996	0.126	0.989	0.025	0.031	0.968
	D	0.992	0.032	0.983	0.021	0.010	0.966

CIFAR10, we only use MNIST and experiment with $\varepsilon = 0.3$ for FGSM and the L_2 attack for CW. There are four target models A, B, C, and D, whose structures are the same as the settings in Defense-GAN. The experiment results are shown in Table 6.

The proposed method is better than MagNet and adversarial training. Although our method is slightly inferior to Defense-GAN in some tests, our method also has certain advantages. (1) Our method is simpler than Defense-GAN. Simultaneously, Defense-GAN requires two steps before feeding the input to the classifier: minimizing the reconstruction error and generating. However, our method only requires generating. (2) Our defense mechanism is a general-purpose defense framework, which means that we can adapt the defense mechanism to different datasets or scenarios with a few adjustments.

6. Conclusions

In this paper, we propose a novel defense strategy utilizing conditional GANs to enhance the robustness of classification models against adversarial examples. Our method is a universal defense framework. We tested it on different datasets and target models, and the experimental results proved that our method is effective against most commonly considered attack strategies. In addition, compared to the state-of-the-art defense methods, the proposed method also has many advantages.

It is worth mentioning that although our method is a feasible and simple defense mechanism, there are still some practical difficulties in implementing and deploying this method. For example, our experimental performance will be reduced on complex datasets. In the future, we will focus on adjusting the network structure of the defense framework to improve the performance on complex scenarios.

Data Availability

The MNIST dataset used to support the findings of the study is public and available at http://yann.lecun.com/exdb/mnist/. The CIFAR10 dataset used to support the findings of the study is public and available at https://www.cs.toronto.edu/~kriz/cifar.html.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61572034) and Major Science and Technology Projects in Anhui Province (18030901025).

References

[1] I. G. Goodfellow, Y. Bengio, and A. C. Courville, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[2] B. Wu, Z. Chen, J. Wang, and H. Wu, "Exponential discriminative metric embedding in deep learning," *Neurocomputing*, vol. 290, pp. 108–120, 2018.

[3] M. M. Y. Zhang, K. Shang, and H. Wu, "Learning deep discriminative face features by customized weighted constraint," *Neurocomputing*, vol. 332, pp. 71–79, 2019.

[4] C. Liu and H. Wu, "Channel pruning based on mean gradient for accelerating convolutional neural networks," *Signal Processing*, vol. 156, pp. 84–91, 2019.

[5] X. Li and H. Wu, "Spatio-temporal representation with deep neural recurrent network in mimo csi feedback," *IEEE Wireless Communications Letters*, vol. 9, no. 5, pp. 653–657, 2020.

[6] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou, "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6172–6181, 2019.

[7] X. Xu, C. He, Z. Xu, L. Qi, S. Wan, and M. Z. A. Bhuiyan, "Joint optimization of offloading utility and privacy for edge computing enabled IoT," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2629–2622, 2019.

[8] X. Xu, X. Zhang, H. Gao, Y. Xue, L. Qi, and W. Dou, "Become: blockchain-enabled computation offloading for IoT in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4187–4195, 2019.

[9] X. Xu, X. Liu, Z. Xu, F. Dai, X. Zhang, and L. Qi, "Trust-oriented iot service placement for smart cities in edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4084–4091, 2019.

[10] X. Xu, X. Zhang, X. Liu, J. Jiang, L. Qi, and M. Z. A. Bhuiyan, "Adaptive computation offloading with edge for 5g-envisioned internet of connected vehicles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2020.

[11] Y. Zhang, C. Yin, Q. Wu, Q. He, and H. Zhu, "Location-aware deep collaborative filtering for service recommendation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–12, 2019.

[12] Y. Zhang, G. Cui, S. Deng, F. Chen, Y. Wang, and Q. He, "Efficient query of quality correlation for service composition," *IEEE Transactions on Services Computing*, 2018.

[13] Y. Zhang, K. Wang, Q. He et al., "Covering-based web service quality prediction via neighborhood-aware matrix factorization," *IEEE Transactions on Services Computing*, 2019.

[14] Q.-s. Zhang and S.-C. Zhu, "Visual interpretability for deep learning: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 1, pp. 27–39, 2018.

[15] R. Shokri, M. Stronati, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy*, pp. 3–18, May 2017, San Jose, CA, USA.

[16] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: analyzing the connection to overfitting," in *Proceedings of the 2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, IEEE, San Jose, CA, USA, pp. 268–282, May 2017.

[17] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic counter measures," in *Proceedings of the Computer and Communication Security*, Denver, CO, USA, October 2015.

[18] C. Szegedy, W. Zaremba, I. Sutskever et al., "Intriguing properties of neural networks," 2013, https://arxiv.org/abs/1312.6199.

[19] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, pp. 2805–2824, 2017.

[20] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, https://arxiv.org/abs/1607.02533.

[21] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Adversarial generative nets: neural network attacks on state-of-the-art face recognition," 2018, https://arxiv.org/abs/1801.00349.

[22] I. Evtimov, K. Eykholt, E. Fernandes et al., "Robust physical-world attacks on deep learning models," 2017, https://arxiv.org/abs/1707.08945.

[23] F. Trame`r, A. Kurakin, N. Papernot, D. Boneh, and P. D. McDaniel, "Ensemble adversarial training: attacks and defenses," 2017, https://arxiv.org/abs/1705.07204.

[24] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings of the 2016 IEEE European Symposium on Security and Privacy*, IEEE, Saarbrucken, Germany, pp. 372–387, March 2015.

[25] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "EAD: elastic-net attacks to deep neural networks via adversarial examples," 2017, https://arxiv.org/abs/1709.04114.

[26] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy*, IEEE, San Jose, CA, USA, pp. 39–57, May 2017.

[27] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, https://arxiv.org/abs/1412.6572.

[28] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, https://arxiv.org/abs/1706.06083.

[29] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Las Vegas, NV, USA, pp. 2574–2582, June 2015.

[30] F. Trame`r, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," 2017, https://arxiv.org/abs/1704.03453.

[31] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Sri- vastava, and K.-W. Chang, "Generating natural language adversarial examples," 2018, https://arxiv.org/abs/1804.07998.

[32] Y. Qin, N. Carlini, I. Goodfellow, G. Cottrell, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," 2019, https://arxiv.org/abs/1903.10346.

[33] F. Trame`r, A. Kurakin, N. Papernot, I. Goodfel-low, D. Boneh, and P. McDaniel, "Ensemble adversarial training: attacks and defenses," 2017, https://arxiv.org/abs/1705.07204.

[34] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proceedings of the 2016 IEEE Symposium on Security and Privacy*, IEEE, San Jose, CA, USA, pp. 582–597, May 2016.

[35] X. Ma, B. Li, Y. Wang et al., "Characterizing adversarial subspaces using local intrinsic dimensionality," 2018, https://arxiv.org/abs/1801.02613.

[36] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: protecting classifiers against adversarial attacks using generative models," 2018, https://arxiv.org/abs/1805.06605.

[37] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ACM, Dallas, TX, USA, pp. 135–147, October 2017.

[38] X. Jia, X. Wei, X. Cao, and H. Foroosh, "Comdefend: An efficient image compression model to defend adversarial examples," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Long Beach, CA, USA, pp. 6084–6092, June 2019.

[39] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Honolulu, HI, USA, pp. 5967–5976, July 2016.

[40] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, https://arxiv.org/abs/1411.1784.

[41] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. X. Song, "Generating adversarial examples with adversarial networks," in *Proceedings of the International Conference on Artificial Intelligence*, Stockholm, Sweden, July 2018.

[42] Y. Song, R. Shu, N. Kushman, and S. Ermon, "Constructing unrestricted adversarial examples with generative models," in *Proceedings of the Neural Information Processing Systems*, Montreal, Canada, December 2018.

[43] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, pp. 2642–2651, Sydney, Australia, August 2017.

[44] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," 2017, https://arxiv.org/abs/1710.11342.

[45] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017, https://arxiv.org/abs/1701.07875.

[46] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," 2017, https://arxiv.org/abs/1702.05983.

[47] H. Lee, S. Han, and J. Lee, "Generative adversarial trainer: defense to adversarial perturbations with GAN," 2017, https://arxiv.org/abs/1705.03387.

[48] G. K. Santhanam and P. Grnarova, "Defending against adversarial attacks by leveraging an entire GAN," 2018, https://arxiv.org/abs/1805.10652.

[49] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 2672–2680, Montreal, Canada, December 2014.

[50] H. Huang, P. S. Yu, and C. Wang, "An introduction to image synthesis with generative adversarial nets," 2018, https://arxiv.org/abs/1803.04469.

[51] C. Wang, Z. Chen, K. Shang, and H. Wu, "Label-removed generative adversarial networks incorporating with K-Means," *Neurocomputing*, vol. 361, pp. 126–136, 2019.

[52] Z. Chen, C. Wang, H. Wu, K. Shang, and J. Wang, "DMGAN: discriminative metric-based generative adversarial networks," *Knowledge-Based Systems*, vol. 192, p. 105370, 2020.

[53] O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, Springer, Munich, Germany, October 2015.

[54] 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*, IEEE, Las Vegas, NV, USA, pp. 770–778, June 2016.

[55] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Wein-berger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Honolulu, HI, USA, pp. 4700–4708, July 2017.

[56] B. X-lab, "Advbox:a toolbox to generate adversarial examples that fool neural networks," 2019, https://github.com/baidu/AdvBox.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8876056, 15 pages
https://doi.org/10.1155/2020/8876056

Research Article

Anomaly Event Detection in Security Surveillance Using Two-Stream Based Model

Wangli Hao,[1] Ruixian Zhang,[1] Shancang Li,[2] Junyu Li,[1] Fuzhong Li,[1] Shanshan Zhao,[2] and Wuping Zhang[1]

[1]*School of Software, Shanxi Agricultural University, Taigu District, Jinzhong, Shanxi 030801, China*
[2]*University of the West of England, Bristol BS16 1QY, UK*

Correspondence should be addressed to Fuzhong Li; lifuzhong@sxau.edu.cn

Received 7 March 2020; Revised 10 May 2020; Accepted 23 June 2020; Published 3 August 2020

Academic Editor: Xiaolong Xu

Anomaly event detection has been extensively researched in computer vision in recent years. Most conventional anomaly event detection methods can only leverage the single-modal cues and not deal with the complementary information underlying other modalities in videos. To address this issue, in this work, we propose a novel two-stream convolutional networks model for anomaly detection in surveillance videos. Specifically, the proposed model consists of RGB and Flow two-stream networks, in which the final anomaly event detection score is the fusion of those of two networks. Furthermore, we consider two fusion situations, including the fusion of two streams with the same or different number of layers respectively. The design insight is to leverage the information underlying each stream and the complementary cues of RGB and Flow two-stream sufficiently. Two datasets (UCF-Crime and ShanghaiTech) are used to validate the effectiveness of proposed solution.

1. Introduction

Security surveillance is increasingly utilized at public places such as streets, hospitals, intersections, shopping malls, and banks, to guarantee public safety. However, the law enforcement agencies and monitoring abilities have not been matched. Consequently, the result is that there are obvious defects in the use of surveillance cameras. Anomaly event detection in surveillance videos is an important research topic in computer vision, which has been widely used in many security related scenarios, including traffic accidents investigation, crimes or illegal activities surveillance, forensics investigation, and violence alerting [1]. Because anomalous events rarely appear in real life, behavioral or appearance patterns deviating from normal patterns are often defined as anomalies [1–3].

Anomaly event detection has been effectively performed on the basis of several prevalent theories in the past decade, such as dictionary learning [4–7], probabilistic models [8, 9], and deep learning [10–12]. However, anomaly event detection is still facing a number challenges.

Most existing researches in anomaly event detection mainly focus on the RGB modality when extracting video features in anomaly event detection. In this work, we propose a two-stream-based model to handle the anomaly event detection problem using the RGB and Flow two convolutional neural network (ConvNets) to extract video features. The RGB stream performs anomaly event detection from video frames, whilst the Flow stream is trained to detect anomalies from motion-based on dense optical flow. Moreover, the proposed framework is able to utilize all frames in the video, while almost no additional calculation is introduced in inference when compared to [13]. The main reason is that the final number of features utilized for training the anomaly model is the same. Specifically, one video is divided into several clips, features of all frames in one clip are averaged to obtain the video clip-level feature.

There are noticeable advantages of our two-stream-based anomaly event detection. Instead of only considering RGB features for MIL models, in this work, we propose TAEDM that can leverage information of both RGB and Flow modalities. Specifically, the information from RGB modality is

the static features underlying still images, such as the color, shape, and appearance of objects or people in the event. The information from the Flow modality is the motion features of the event. As a result, TAEDM can capture the complementary information on RGB stream from still images and motion between images in one video sufficiently. We evaluate the developed approach on two different-scale benchmark datasets, including UCF-Crime [13] and ShanghaiTech [14]. The extensive ablation experimental studies demonstrate that our model obtains the state-of-the-art performance.

The main contributions of this work are summarized as follows:

(i) A novel two-stream-based anomaly event detection model is proposed for anomaly detection in surveillance videos. Furthermore, a dense feature extraction method is proposed to obtain video-level feature.

(ii) Proposed models are tested using benchmark datasets UCF-Crime [13] and ShanghaiTech [14], and results from both datasets show good performances than existing works.

The rest of this paper is organized as follows: Section 2 reviews the state-of-the-art research in anomaly event detection. Section 3 proposes a two-stream-based anomaly event detection model. Experimental results are elaborated in Section 4 and further discussion is presented in Section 5. Section 6 concludes this paper.

2. Related Work

In this section, we will discuss the most recent research results in anomaly event detection, and details about anomaly detection, ranking, and two-stream action recognition will be discussed.

2.1. Anomaly Detection. In computer vision, anomaly event detection is one of the most challenging problems and has attracted lots of research efforts in the past decades [15–21], where the commonly used detection methods can be roughly categorized into following three groups.

The first category of anomaly detection methods focuses on the hypothesis that anomalies are rare, and behaviors different from normal patterns seriously are seen as anomalous. In these methods, the regular patterns are encoded through various statistic models, such as Gaussian process based models [22, 23], the model of social force [24], Hidden Markov-based models [15, 25], the spatial-temporal Markov random field based models [26, 27], the combination of dynamic models [21], and treat anomalies as outliers.

The second category of anomaly detection approaches is sparse reconstruction [3, 14, 16, 28], which is utilized for usual pattern learning. Specifically, a dictionary is constructed by employing sparse representation for normal behavior, and the ones with high error are detected as anomalies. Recently, with the promising breakthrough of deep learning, some researchers construct deep neural networks for anomaly detection, including video prediction learning [29], and abstraction feature learning [6, 30, 31].

The third group is the hybrid methods of normal and anomaly behavior for modelling [13, 32, 33], in which, under weakly supervised setting, multi instance learning (MIL) is utilized to model motion patterns [13, 33], e.g., Sultani et al. developed an MIL-based classifier [13], which is employed to detect anomalies. Meanwhile, a deep ranking model is utilized to predict anomaly scores.

Aimed at leveraging the superiority of Sultani's work that considering both normal and anomalous videos, in this work, we rebuilt the model using a weak labelled supervised learning.

2.2. Ranking. Learning to rank is a popular research problem in machine learning and many research efforts have been conducted, including [7, 11, 34–38]. These approaches aimed at boosting relative scores of the pieces rather than individual scores. Rank-SVM [7] was proposed to enhance the retrieval performance of search engines.

The detection algorithm proposed in [34] can solve multiple-instance ranking problems through gradual linear programming. This method has been utilized in computational chemistry to solve hydrogen abstraction problem. More recently, researchers have proposed deep ranking networks for computer vision-related applications and achieved promising success, such as highlight detection [35], person reidentification [11], feature learning [36], Graphics Interchange Format (GIF) generation [37], face detection and verification [38], and metric learning and image retrieval [39]. All the above deep ranking approaches need extensive annotations of both positive and negative samples. Unlikely, in this work, a ranking model is proposed by reformulating anomaly detection problem as a regression problem under the ranking framework based on both normal and anomalous samples. The proposed model utilizes MIL depending on weakly supervised data to train the anomaly model and located anomaly with video segment level during testing. Unlike the conventional multiple instance learning (MIL) setting, the proposed ranking component forces ranking only includes two segments with the highest anomaly score in the negative and positive bags.

2.3. Two-Stream Action Recognition. Video-based action recognition has been extensively researched and achieved comparable attention recently. Among them, the two-stream-based action recognition is superior [40–42]. Inspired by neuroscience, one kind of action recognition methods introduced two-stream neural network architecture [40–42], to perform RGB and Flow feature extraction in parallel. The final score of action classification can be achieved by fusing the results of two paths. In order to further enhance the action recognition performance, Wang et al. developed a novel Temporal Segment Network (TSN) [41], which focuses on modelling the long-range temporal structure in videos. Further, various extensions of two-stream model [40] that explore convolutional fusion [42]

and residual connections [43, 44] were developed. The model in [43] established the residual connections between RGB and Flow streams. The STDDCN [44] integrated the multiscale information into residual connections via dense-connectivity interaction and contained a new knowledge distillation module.

Two-stream-based methods have been widely employed on some other task of video, such as action recognition [40–43, 45, 46]. However, two-stream-based methods are rarely applied to anomaly event detection. Inspired by the two-stream-based action recognition architectures leveraging the complementary information of RGB and Flow modalities underlying actions, we first design a novel two-stream anomaly event detection model. Compared with action recognition, the anomaly event detection can identify the kind of behavior (normal or abnormal) and locate the time range of an exception. That leads this problem more difficult to solve than the others.

3. Two-Stream Anomaly Event Detection Network

This section will detail the proposed two-stream-based anomaly event detection model as shown in Figure 1. We first will introduce the abnormal video and the normal video, and then divide them into multiple time video clips for extracting the two-stream features (RGB stream and Flow stream) of the video clips. A fully connected neural network will be trained using a ranking loss function, which calculates the highest-scoring instance (shown in blue) and the fusion operation then will be performed.

Video clip can be naturally split into synchronous spatial and temporal parts. The spatial component underlying the individual frame image consists of scenes and object information in the video. The temporal component hidden in the motion across the images carries the movement between the objects and the observer. We designed our anomaly detection model accordingly and decomposed it into two streams, as is illustrated in Figure 1. Each stream is realized via a deep convolutional network (ConvNet), anomaly detection scores of which are fused in the late.

In the proposed model, video segments that obtained high anomaly scores will be marked as anomaly event. Each video will be split into equal number of nonoverlapping segments. The video containing anomaly segment is labelled as positive and a video without any anomaly segment is labelled as negative. A positive/negative video is treated as a positive/negative bag and the segments as instances in the multiple instance learning. Through ranking method, anomaly scores for each video segment can be obtained and the video segments obtained high anomaly scores is seen as anomaly event.

First, given the abnormal video and the normal video, we divided them into multiple time video clips. Secondly, we extracted the two-stream features (RGB stream and Flow stream) of the video clips and then trained a fully connected neural network using a ranking loss function, which calculates the highest-scoring instance (shown in blue) and performed the fusion operation in the last step.

3.1. Problem Formulation. In the past decade, a number of pattern learning methods have been developed [10, 15, 19, 25], most of them assuming that any pattern that violates this common pattern should be abnormal. In fact, it is impossible to propose a method to define a full set of normal patterns, because the normal pattern may contain too many different events and behaviors. To define anomaly events is another challenge, since anomaly events may also contain many similar events and behaviors.

To handle the above issues, the proposed method formulates each anomaly detection task (RGB branch and Flow branch) as a regression problem, which is realized under the ranking framework by leveraging both normal and anomalous data. To achieve more precise segment-level labels, a weakly supervised deep multiple instance learning (MIL) ranking is employed. Specifically, weakly supervised rank indicates that the model only knows whether there is an abnormal event in a video rather than the category of the anomaly event and the corresponding occurrence time during training.

The differences of the proposed pattern learning method from those in [10, 15, 19, 25] is that our model utilizes both normal and anomalous data rather than normal data in previous studies (e.g., [10, 15, 19, 25]). Furthermore, our model is formulated as a regression problem, which means that we consider a certain segment as an abnormal event based on regression prediction score rather than the probability less than a certain threshold.

3.2. Data Formulation. To align the data for deep MIL setting in anomaly detection, the source video is first split into equal number of nonoverlapping segments during training. All segments in the same video are denoted as a bag, and each segment is acted as an instance. All videos formed two different bags, positive bags and negative bags, respectively. The segments of anonymous video are treated as positive bag and those of normal video negative bag. Moreover, as our insight is based on leveraging the complementary information of RGB and Flow streams, video clips in each bag are all decomposed into RGB and Flow components. Each kind of component is fed into the corresponding branch networks separately.

3.3. Network Architecture. The deep MIL framework includes two main branch deep MIL ranking networks: RGB and Flow, as shown in Figure 1. Each branch contains feature extraction and instance scoring parts. Concerning the feature extraction, ResNet [47] is chosen as backbone because of its superiority in both efficiency and effectiveness.

3.3.1. Spatial Branch ConvNet. Spatial Branch ConvNet focuses on single video frame, effectively conducting anomaly event detection from still images. The static RGB stream by itself contains useful information, since some anomaly events are closely associated with specific objects. Actually, as will be reported in the section of experiments,

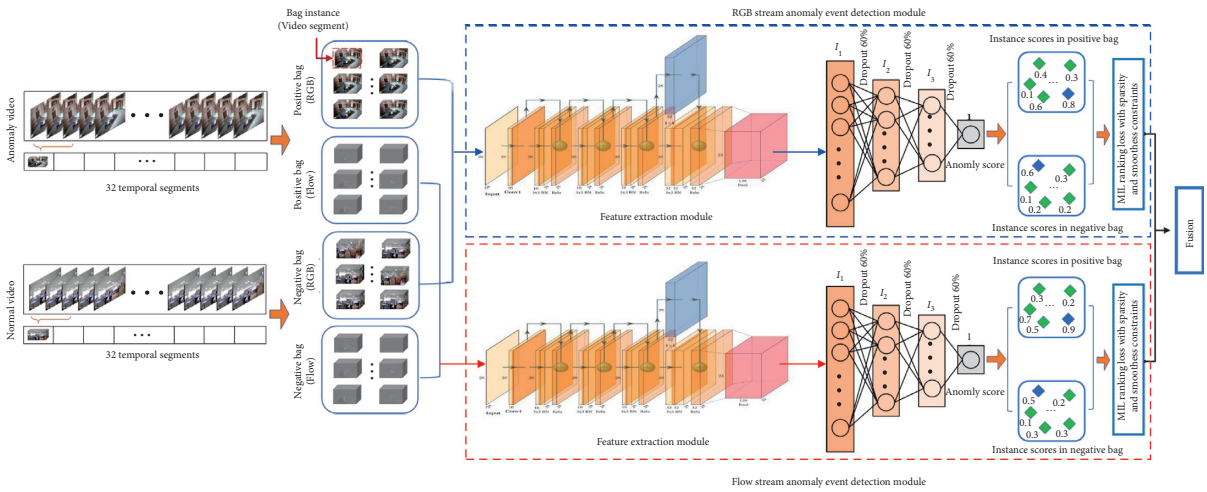


Figure 1: A flowchart of the proposed two-stream based anomaly event detection framework.

anomaly event detection from still images (the RGB anomaly event detection stream) is quite competitive by itself.

3.3.2. Temporal Branch ConvNet. Unlike the conventional ConvNet models, the input of proposed temporal anomaly event detection stream is the stacked optical flow displacement fields among several adjacent frames. This input explicitly models the motion between video clip images, which makes the anomaly event detection easier, as no implicit motion estimation is required.

The dense optical flow is formed using a group of displacement vector fields $\mathbf{v}_t$ between adjacent t and $t+1$ frames. Further, $\mathbf{v}_t(m,n)$ indicates the displacement vector at the corresponding point (m,n) in frame t, which represents the movement of point (m,n) from frame t to frame $t+1$. Moreover, the displacement vector $\mathbf{v}_t$ contains two components, including horizontal and vertical ones, which dubbed as $\mathbf{v}_t^x$ and $\mathbf{v}_t^y$, respectively. $\mathbf{v}_t^x$ and $\mathbf{v}_t^y$ are seen as image channels (as shown in Figure 2) and can be fed into the temporal anomaly event detection stream network.

Figures 2 (a1), (b1) and (a2), (b2) indicate the pair of adjacent video frames with the highlighted moving area outlined with a red rectangle. Figures 2(c1)/(c2) denote the horizontal part $\mathbf{v}_t^x$ of displacement vector field (higher/lower intensity relates to positive/negative values). Figures 2 (d1) and (d2) illustrate the vertical part $\mathbf{v}_t^y$ of displacement vector field.

3.3.3. Loss Function. To pursue better performance, we employ the following loss function (referred from [13]) to train each branch network:

$$\mathscr{L} = l\left(S_p, S_n\right) + \|\mathscr{W}\|_F, \tag{1}$$

where S_p and S_n denote positive and negative bags, respectively. $l(S_p, S_n)$ indicates the loss over these two kind of bags. $\|\mathscr{W}\|_F$ denotes the $F-$ norm regularization on weights of the model, for boosting its generalization. Among them, $l(S_p, S_n)$ is defined as

$$l\left(S_p, S_n\right) = l_{\text{rank}} + \lambda_a l_{\text{smooth}} + \lambda_b l_{\text{sparsity}}, \tag{2}$$

in which l_{rank} denotes the ranking loss, l_{smooth} denotes the temporal smooth restrict and l_{sparsity} represents the sparsity constraint. λ_a and λ_b are two hyper-parameters which balance the strengths of corresponding terms. Among them, l_{rank} is formulated as

$$l_{\text{rank}} = \max\left(0, 1 - \max_{i \in S_p} f\left(C_a^i\right) + \max_{i \in S_n} f\left(C_n^i\right)\right), \tag{3}$$

where S_p and S_n share the same meanings with those of equation (1). C_n and C_a indicate normal and anomalous video instances. $f(C_a^i)$ and $f(C_n^i)$ denote the predicted scores for the corresponding video instances.

The l_{rank} forces rank only on two segments with the highest anomaly score in the negative and positive bags separately, rather than every segment of the bag. Thus, the max operation is performed over all instances in each bag. The reason for this different setting is the absence of video segment-level annotations in anomaly event detection task.

The l_{rank} loss here is superior for anomaly detection task due to several appealing reasons. First, it can enforce the anomalous video segments to achieve higher anomaly scores compared to normal ones. Furthermore, it can separate the positive instances and negative instances based on anomaly score.

On the other hand, l_{smooth} and l_{sparsity} are defined as

$$\begin{aligned} l_{\text{smooth}} &= \sum_{i=1}^{n-1} \left(f\left(C_a^i\right) - f\left(C_a^{i+1}\right)\right)^2, \\ l_{\text{sparsity}} &= \sum_{i=1}^{n} f\left(C_a^i\right), \end{aligned} \tag{4}$$

where n is the number of instances in the specific bag. l_{smooth} is utilized to guarantee the temporal smoothness via minimizing the difference of anomaly scores between neighboring video instances in a bag. l_{sparsity} is employed to enforce the sparsity of scores in the anomalous bag. The

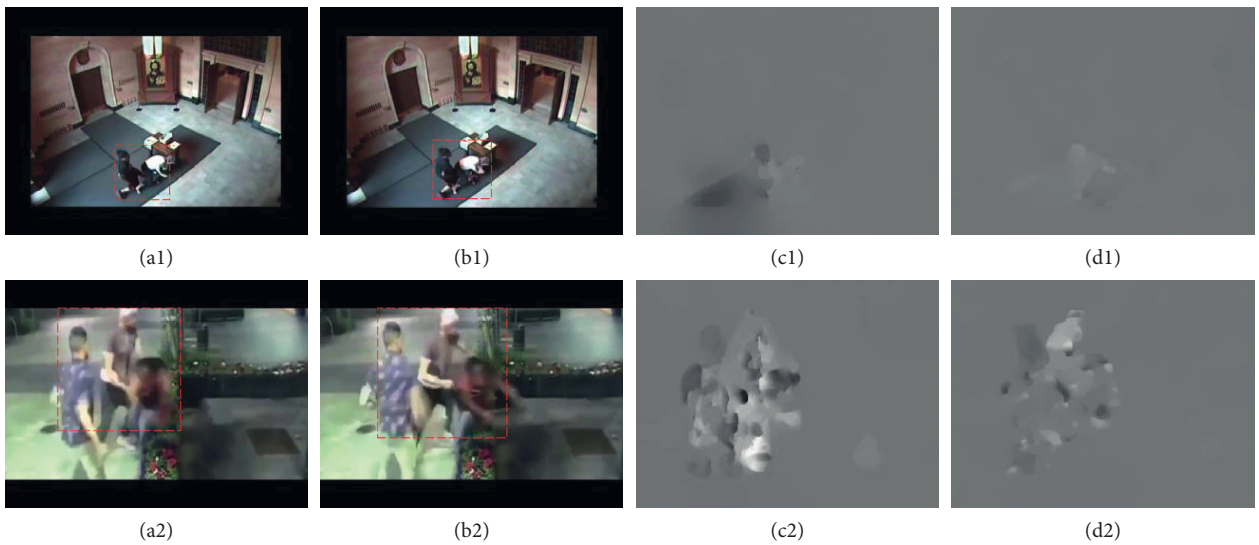


Figure 2: Optical flow examples. (a1, b1) and (a2, b2): the pair of adjacent video frames with the highlighted moving area outlined with a red rectangle. (c1, c2) horizontal part $\mathbf{v}_t^x$ of displacement vector field (higher/lower intensity relates to positive/negative values). (d1, d2): vertical part $\mathbf{v}_t^y$ of displacement vector field.

reason for introducing l_{sparsity} loss function is that few segments may involve the anomaly event.

4. Experiments

In this section, we will illustrate our experiments in detail from aspects including datasets, implementation details, evaluation metric, and sufficient quantitative and qualitative experiments, respectively.

4.1. Datasets. UCF-Crime [13] and ShanghaiTech [14] are two popular benchmark datasets commonly used in anomaly detection task. In this work, we use both datasets to validate the superiority of our proposed anomaly event detection model. With following steps, the proposed model can also support other datasets: (1) extract the RGB image and optical flow images of each video in the dataset; (2) extract their corresponding features; and (3) feed both the RGB and Flow stream features into corresponding branch subnetworks in the proposed model for training and obtaining expected test results. Before introducing the details, we first briefly introduce the two benchmark datasets as follows.

4.1.1. UFC-Crime. Reference [13] is a large-scale dataset that contains a total of 13 anomaly events and 1900 real-word surveillance videos. Among them, 950 videos include clear anomalies and the other videos are treated as normal video. Further, concerning the dataset partitioning, the training set contains 1610 videos (800 normal videos, 810 anomalous videos), and the test set contains 290 videos (150 normal, 140 anomalous videos).

4.1.2. ShanghaiTech. Reference [14] is a medium-scale dataset with a total of 437 videos, which contains 130 abnormal events of 13 scenes. This dataset cannot be utilized directly to perform anomaly event detection because the training set has no abnormal video. To tackle this problem, Zhong et al. [48] rebuilt the dataset via randomly choosing abnormal test videos and putting them into the training data and vice versa. Simultaneously, both training and test dataset contain 13 scenes. This new organization of dataset made it suitable for anomaly event detection task. Thus, we perform the same operation as that in [48], before executing the experiments.

4.2. Implementation Details and Evaluation Metrics. To implement the proposed model, we first extract features of RGB and Flow images from the last fully connected (FC) layer of the ResNet network [47]. Concerning the RGB stream, ResNet features for every frame are computed. The video segment-level feature can be obtained by averaging all frame features in the corresponding video segment. Similarly, for the Flow stream, features can be extracted using the same way of RGB stream. The only difference between these two streams is that each frame in Flow stream contains two directional flow images, namely, vertical ($\mathbf{v}_t^x$) and and horizontal ($\mathbf{v}_t^y$) images as stated above, which makes the ResNet infeasible to extract their features. Specifically, $\mathbf{v}_t^x$ and $\mathbf{v}_t^y$ are all grayscale images with only one channel (the concatenation of them only has two channels), while the input sample of feature extraction network (ResNet) needs three ones. To handle this problem, we concatenate the two directional flow images and their average variant to form the input flow sample with three channels for the feature extraction network.

After obtaining segment-level RGB and Flow ResNet features, we feed them (2048D) into a three-layer FC neural network as that of [13]. Further, the Adagrad optimizer is utilized, which initial learning rate is 0.001. To perform a fair comparison, the smoothness constraint, the sparsity restriction, and the segment number of each video are the

same with those of [13]. We stop our training at 20, 000 iterations.

The following commonly used evaluation metrics are adopted to validate the performance our model. They are receiver operating characteristic (ROC) curve and the area under the curve (AUC), respectively. The reason we utilize ROC and AUC is that they are two popular metrics for anomaly event detection tasks [13, 21, 48]. For fair comparison with other works and to verify the effectiveness of our model, ROC and AUC are employed.

4.3. Experimental Results

4.3.1. Evaluation of the Proposed Model. To validate the performance of the proposed method, we compare the results with those of state-of-the-art models, based on UCF-Crime [13] and ShanghaiTech [14]. Comparison ROC curves are shown in Figure 3. In Figure 3, RGB, Flow and Two denote the anomaly event detection results of different models based on RGB stream network, Flow stream network and the fusion of them separately.

Figure 3 illustrates that RGB, Flow, and Two obtain better results than the other models, validating the dense feature extraction is effective. Further, Two yields better results than those of RGB and Flow, which verifies the superiority of the proposed model.

The AUC results from different models on UCF-Crime [13] and ShanghaiTech [14] are displayed in Tables 1 and 2, respectively. It can be seen that the results are the same with those of Figure 3, which further validates the effectiveness of our model.

4.3.2. Ablation Studies. In this section, several ablation studies are designed to demonstrate the effectiveness of the proposed model.

(1) Evaluation of the Generalization Capacity of the Model. To validate the generalization of the proposed method, we present the results of proposed method based on models with different depths and architectures, including ResNet50, ResNet100, ResNet150, and VGG16, respectively, as shown in Tables 3 and 4. The results in Tables 3 and 4 illustrate that model Two achieves better results than those of the corresponding RGB and Flow models in all cases, which verifies the generalization capacity of the proposed model in terms of model depth and architecture.

Additionally, the ROC curves of ResNet50, ResNet100, ResNet150, and VGG16 are exhibited in Figures 4 and 5, respectively. Among them, Figures 4(a)–4(c) and 5(a)–5(c) report the ROC curves of RGB, Flow, and Two networks from ResNet50, ResNet100, ResNet150, and VGG16 models, respectively. Figures 4 and 5 further validate the generalization capacity of our method on model depth and architecture.

(2) Evaluation of the Fusion of Two Streams. As stated above, different backbone feature extraction models are employed to assess the proposed method. This naturally raises the following evaluations, including the fusion of two streams with the same number of layers and the fusion of two streams with different number of layers separately:

(1) Fusion of two streams with the same number of layers: To validate the effectiveness of the fusion of two streams with the same number of layers, we utilize the identical network (including ResNet50, ResNet100, ResNet150, and VGG16, respectively) to perform both RGB and Flow stream feature extractions. Comparison results are presented in Tables 1 and 2. Tables 1 and 2 show model Two obtains uniformly better results than those of the corresponding RGB and Flow models, which illustrates the effectiveness of the proposed method under the same layer fusion setting (dubbes as $\text{Fusion}_{\text{same}}$ setting).

(2) Fusion of two streams with different number of layers: To verify the effectiveness of the fusion of two streams with different number of layers, we employ different networks to perform RGB and Flow stream feature extractions, respectively.

Tables 5 and 6 illustrate that the performance of model Two is consistently superior to those of corresponding RGB and Flow models, which validates the superiority of the proposed method under the different layers fusion setting (dubbed as $\text{Fusion}_{\text{dif}}$ setting). Further, an appealing conclusion can be drawn that $\text{Fusion}_{\text{dif}}$ surpasses $\text{Fusion}_{\text{same}}$ in most cases. In addition, the case that RGB stream with ResNet50 and Flow stream with VGG16 yields our best anomaly event detection results, which again verifies the effectiveness of fusion under different model architectures and depths.

(3) Evaluation of Fusion Proportion. This paper obtains the final anomaly detection scores via fusing two streams via the following equation: $Score = \beta * Score_{Flow} + (1-\beta) * Score_{RGB}$. To validate the effects of fusion proportion β between two streams, we perform anomaly event detection with various fusion proportion ranges from 0.1 to 0.9 with step size 0.1.

(1) Results of fusion proportion with same number of layers: Figure 6 reports the results of $\text{Fusion}_{\text{same}}$ with different fusion proportions. From Figure 6, we can see that each ResNet backbone (including RenNet50, RenNet100, and RenNet150) obtains similar fusion anomaly detection results respectively under different fusion proportions, with differences range from 0.2 to 1.23 (UCF-Crime) and 0.05 to 0.6 (ShanghaiTech). Further, the best fusion results are seemed obtained at $\beta = 0.7$ in most ResNet backbone cases. On the other hand, VGG backbone's fusion varies a lot, about 4 points (UCF-Crime) and 1 point (ShanghaiTech). Moreover, the best fusion result is achieved at $\beta = 0.9$.

(2) Results of fusion proportion with different number of layers: Figure 7 shows the results of $\text{Fusion}_{\text{dif}}$ with different fusion proportions on UCF-Crime. Figures 7(a)–7(d) denote the results of ResNet50 (Flow), ResNet100 (Flow), ResNet150 (Flow), and VGG16 (Flow) fusing with different RGB networks,

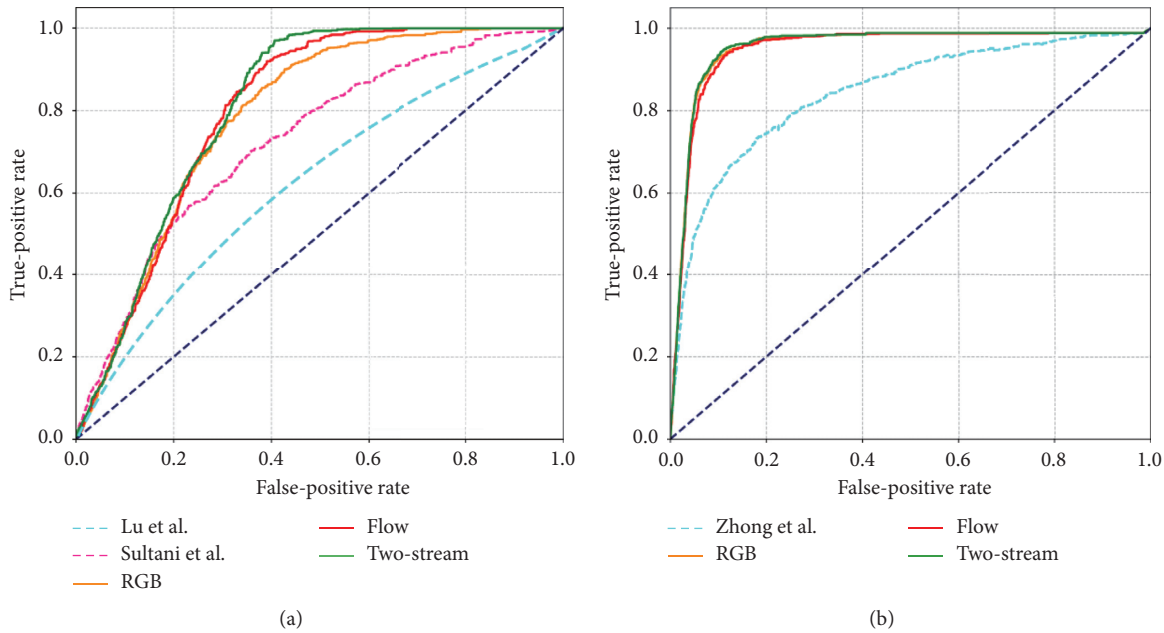


Figure 3: ROC curves of different models on UCF-Crime [13] and ShanghaiTech [14]. (a) Results of UCF-Crime [13] and (b) that of ShanghaiTech [14].

Table 1: Quantitative comparison on UCF-Crime.

Method	AUC (%)
Hasan et al. [6]	50.6
Lu et al. [28]	65.51
Sultani et al. [13] with w/o constraints	74.44
Sultani et al. [13] with w constraints	75.41
Ours (RGB)	78.51
Ours (flow)	80.29
Ours (two)	81.22

Table 2: Quantitative comparison on ShanghaiTech.

Method	AUC (%)
Zhong et al. [48]	84.44
Ours (RGB)	94.53
Ours (flow)	95.42
Ours (two)	96.74

respectively. Figure 7 shows that as fusion proportion value β increases, the trends of the AUC curves of Figures 7(a) and 7(d) are monotonically increasing and those of Figures 7(b) and 7(c) are monotonically decreasing. Reasons are that anomaly scores of flow streams of Figures 7(a) and 7(d) are superior to those of their corresponding fused RGB streams, and anomaly scores of flow streams of Figures 7(a) and 7(d) are worse than or comparable to those of their corresponding fused RGB streams. In other words, which stream has better performance, the fusion result will be better when its proportion is higher. The general proportion value for that stream with better results is 0.8 in most cases.

Table 3: Comparison results of different models on UCF-Crime.

Model	ResNet50	ResNet100	ResNet150	VGG16
RGB AUC (%)	78.51	76.84	77.44	71.21
Flow AUC (%)	79.92	77.51	77.93	80.29
Two AUC (%)	80.87	79.54	79.19	80.58

Table 4: Comparison results of different models on ShanghaiTech.

Model	ResNet50	ResNet100	ResNet150	VGG16
RGB AUC (%)	94.53	95.11	95.28	83.07
Flow AUC (%)	95.42	94.81	95.28	95.49
Two AUC (%)	95.53	95.23	95.38	95.85

Figure 8 presents the results of $\text{Fusion}_{\text{dif}}$ with different fusion proportions on ShanghaiTech. Figures 8(a)–8(d) show the results of ResNet50 (Flow), ResNet100 (Flow), ResNet150 (Flow), and VGG16 (Flow) fusing with different RGB networks respectively. Figure 8 shows that as fusion proportion value β increases, the trends of the almost all AUC curves of four subfigures are monotonically increasing, and also three curves are almost monotonically decreasing. Reason is that flow streams of these curves have higher anomaly scores than those of their corresponding fused RGB streams, and anomaly scores of flow streams are worse than or comparable to those of their corresponding fused RGB streams. In this dataset, we also obtain the similar conclusion that for the stream with a better result, the fusion result will be better when its proportion is higher. The general proportion value for that stream with better results is 0.9 in most cases.

4.3.3. Qualitative Results. To provide a more intuitive perception of the proposed model, we introduce the scores of anomalies per segment in a video obtained via our

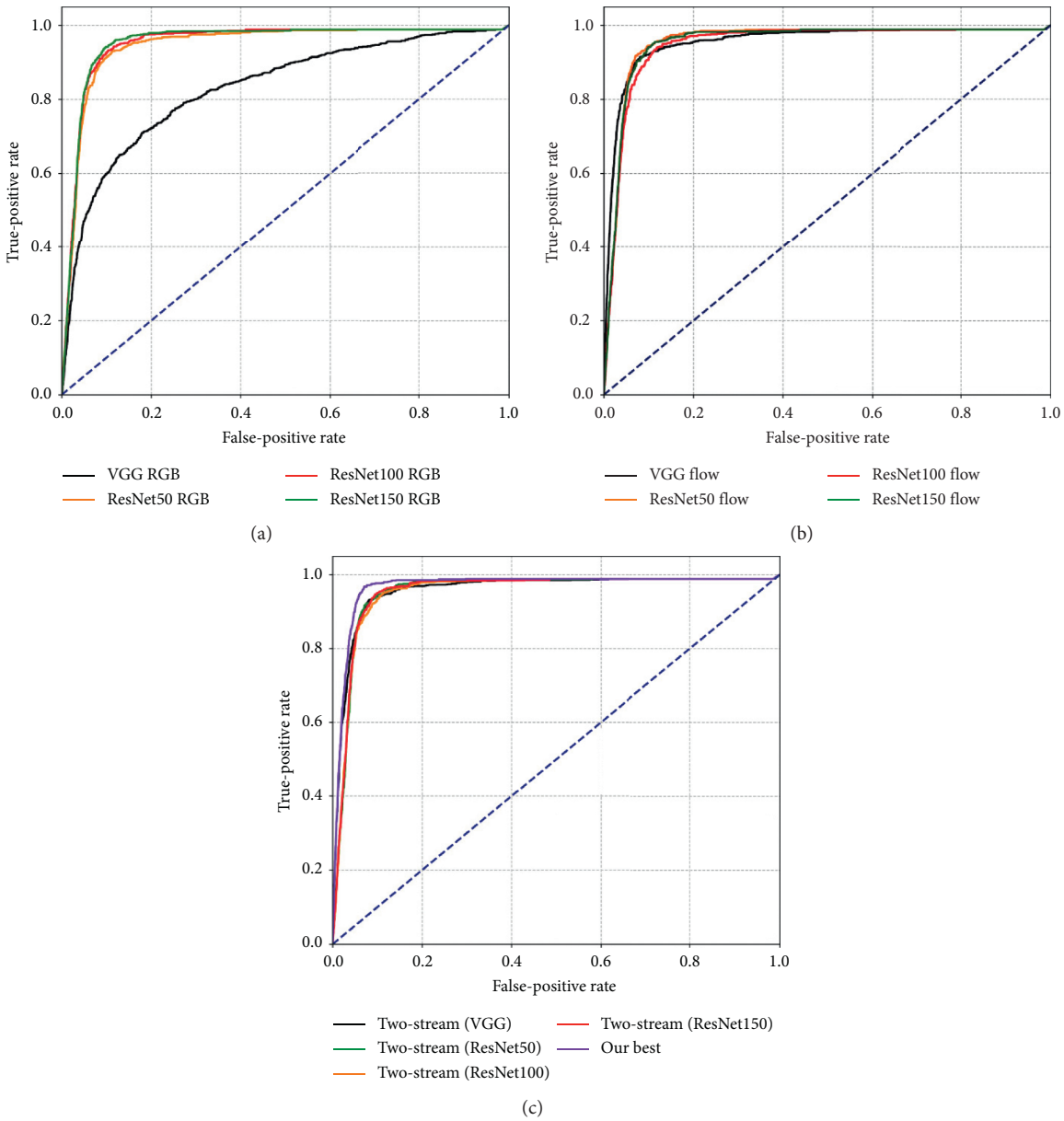


Figure 4: ROC curves of different models on ShanghaiTech. (a) Results of RGB stream. (b) Results of Flow stream. (c) Results of the fusion of RGB and Flow two streams.

method. Meanwhile, corresponding events with the highest or lower abnormal event scores in the video are also presented, with results presented in Figure 9 for UCF-Crime and Figure 10 for ShanghaiTech. Specifically, three example events are displayed in Figures 9 and 10, respectively. The first row of Figures 9 and 10 show the visualization results obtained by our best model variant, and the green blocks in the gray rectangle in Figure 9 or purple rectangles in Figure 10 represent the ground truth time period in which the anomaly event occurred. The second row of Figures 9 and 10 present the visualization results of different variants of our model, including results of ResNet50, ResNet50, ResNet150, and the best model variants, respectively. Simultaneously, several frames at the corresponding time are exhibited, including corresponding frames with the highest or lower abnormal event scores in the video. The area marked by the red circle in the image is the corresponding abnormal event. From Figures 9 and 10, we can see that our model can effectively predict the time period of anomalous events.

5. Discussion

It is noted that the RGB stream focuses on the appearance information and Flow stream concentrates on motion clues underlying a certain video. The fusion of these two streams with the same number of layers boosts the anomaly event detection performance effectively, as Table 1 and Table 2 show. Reason is that this fusion can

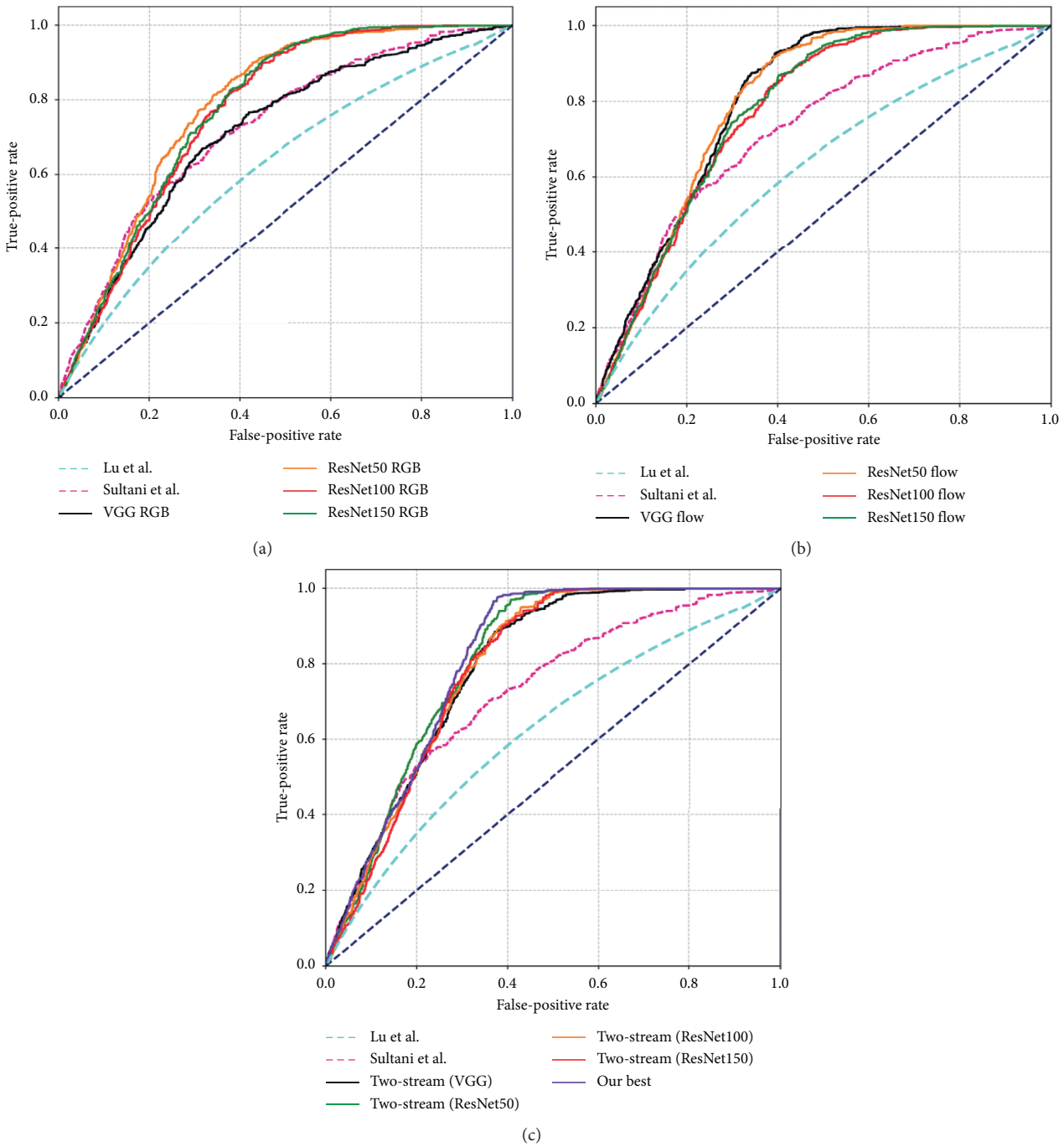


Figure 5: ROC curves of different models on ShanghaiTech. (a) Results of RGB stream. (b) Results of Flow stream. (c) Results of the fusion of RGB and Flow two streams.

Table 5: Comparison results of different models of two streams with different number of layers on UCF-Crime.

Flow	RGB			
	ResNet50	ResNet100	ResNet150	VGG16
ResNet50	80.87	80.48	80.19	80.08
ResNet100	80.22	79.54	79.16	78.28
ResNet150	80.11	79.66	79.19	78.75
VGG16	81.21	80.8	80.54	80.58

Table 6: Comparison results of different models of two streams with different number of layers on ShanghaiTech.

Flow	RGB			
	ResNet50	ResNet100	ResNet150	VGG16
ResNet50	95.53	95.54	95.55	95.85
ResNet100	94.93	95.23	95.13	95.33
ResNet150	94.93	95.13	95.38	95.33
VGG16	96.66	96.74	96.61	95.85

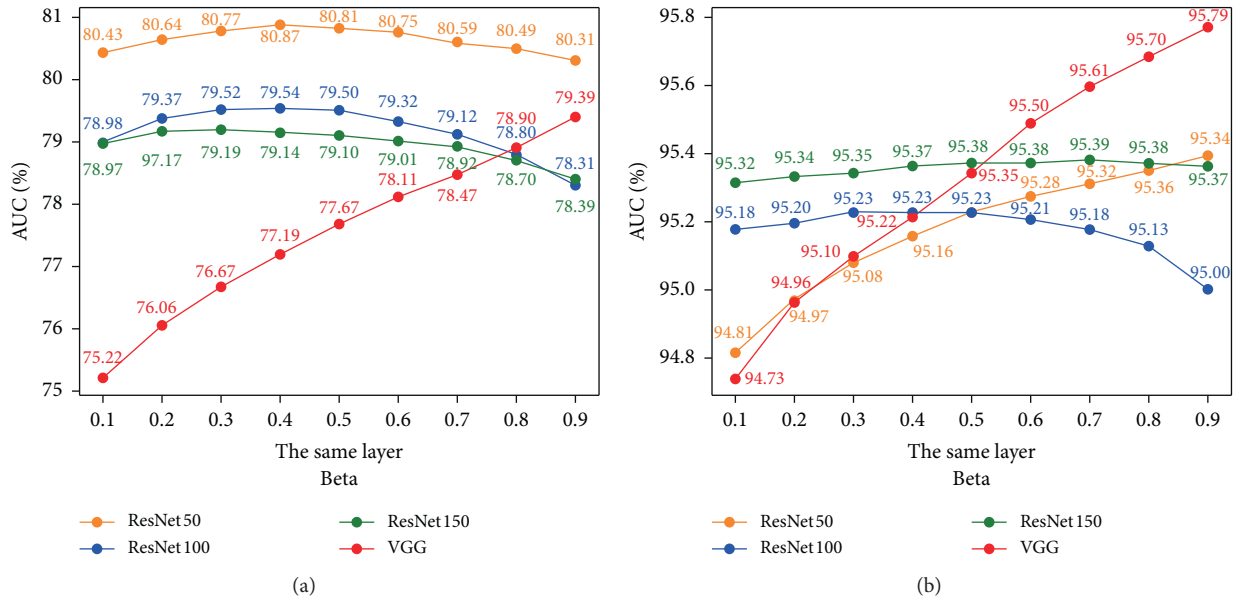


Figure 6: Results of different fusion proportions under $\text{Fusion}_{\text{same}}$ setting on UCF-Crime and ShanghaiTech. (a) Results of UCF-Crime. (b) Results of ShanghaiTech.

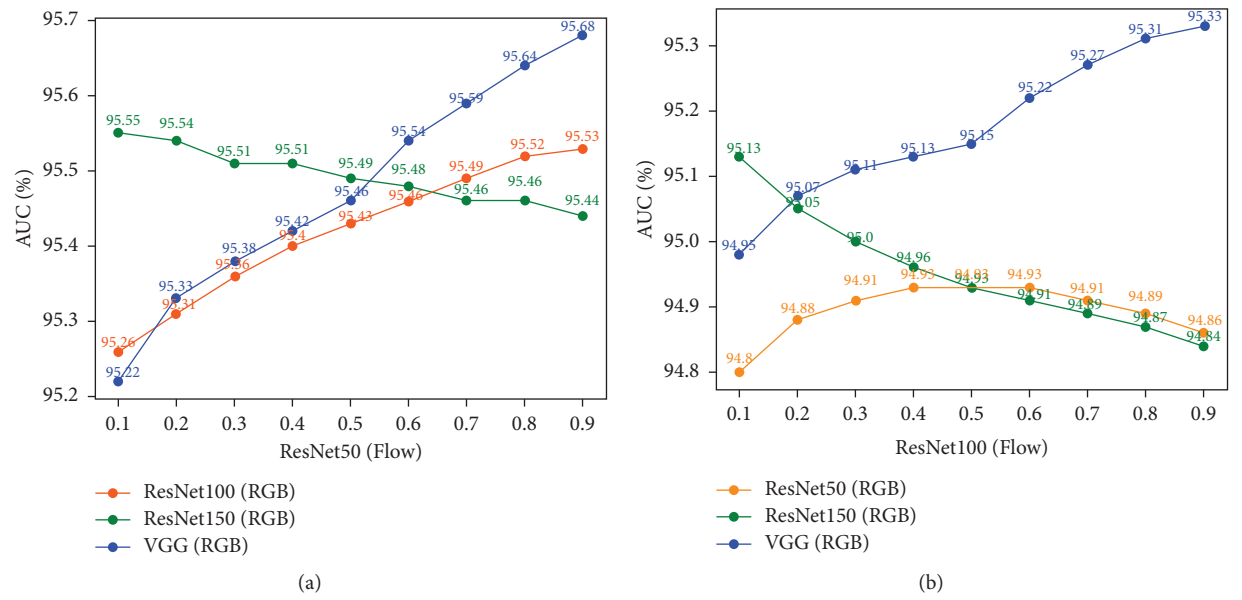


Figure 7: Continued.

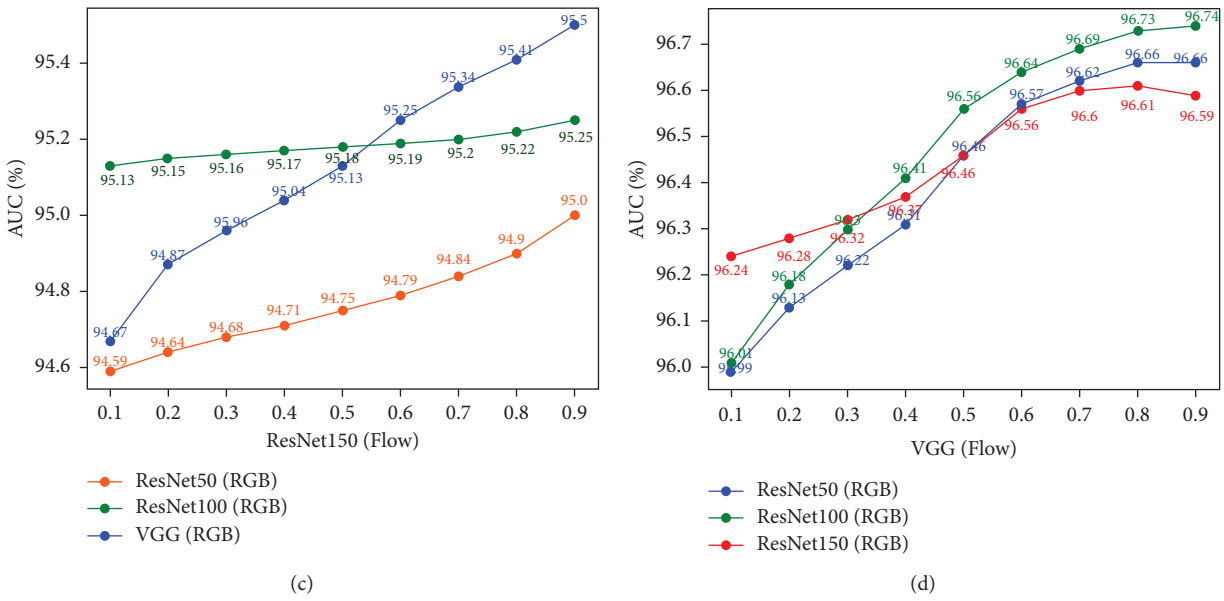


Figure 7: Results of different fusion proportions under $\text{Fusion}_{\text{dif}}$ setting on UCF-Crime.

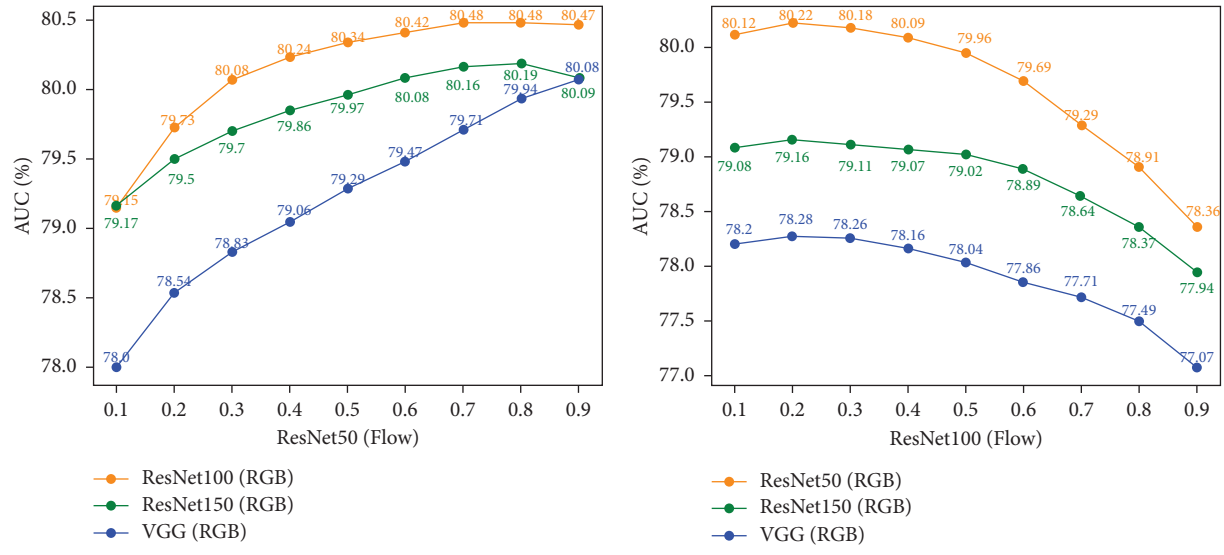


Figure 8: Continued.

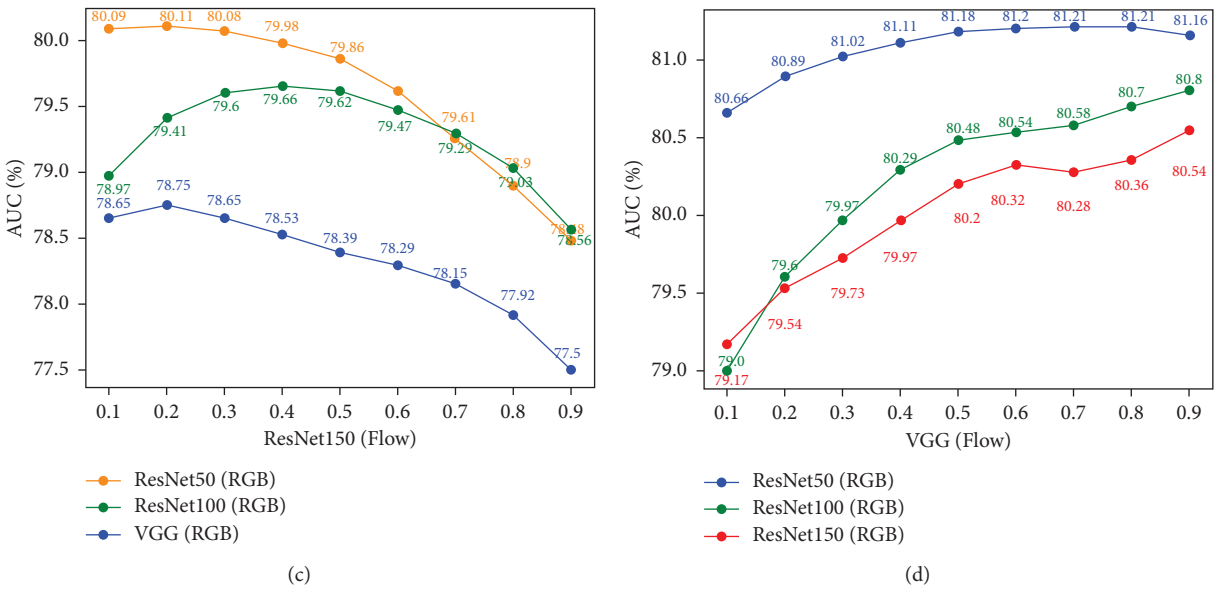


Figure 8: Results of different fusion proportions under $\text{Fusion}_{\text{dif}}$ setting on ShanghaiTech.

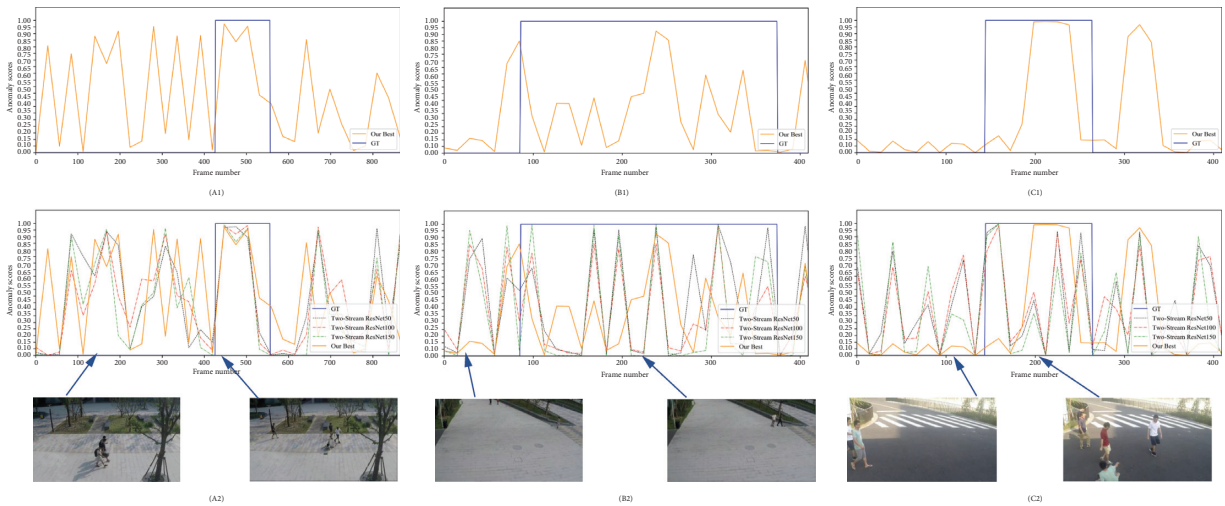


Figure 9: The visualization results of our method on testing videos on UCF-Crime. The first row shows the visualization results obtained by our best model variant, and the green block in the gray rectangle represent the ground truth time period in which the anomaly event occurred. The second row presents the visualization results of different variants of our model, including results of ResNet50, ResNet50, ResNet150, and the best model variants, respectively.

leverage the complementary spatiotemporal information on the same scale underlying videos. In addition, the fusion of two streams with different number of layers achieves better results than those of the same layer fusion. Reason is this different layer fusion not only utilizes the complementary information between two streams, but also leverages the multiscale information at different layers, as Tables 5 and 6 show. Thus, fusion of RGB and Flow two streams is optimal in anomaly event detection task.

The benefits of our proposed solution are that it can further improve the performance of anomaly event detection significantly by leveraging the complementary information of RGB.

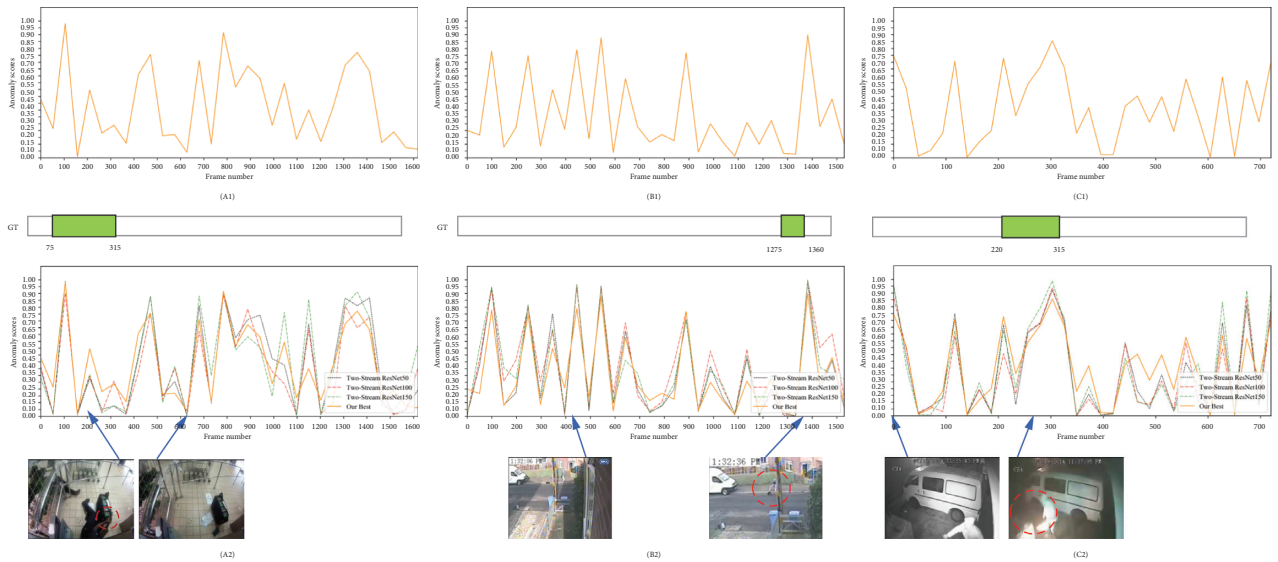


FIGURE 10: The visualization results of our method on testing videos on ShanghaiTech. The first row exhibits the visualization results obtained by our best model variant, and the purple rectangles represent the ground truth time period in which the anomaly event occurred. The second row shows the visualization results of different variants of our model, including results of ResNet50, ResNet50, ResNet150, and the best model variants, respectively, and Flow modalities in the video. Moreover, our proposed solution can provide inspiration for other video-related tasks, including video classification, video segmentation, video tracking and video detection, through bistream setting to obtain the improved.

6. Conclusion

This paper proposes a novel two-stream-based model for anomaly event detection. Specifically, this model consists of RGB and Flow two branch networks, and the final anomaly detection score is the fusion of two networks. Meanwhile, we consider two fusion strategies, including the fusion of two streams with the same of different number of layers, respectively. The proposed model can utilize the complementary information of the two streams hidden in the video, which can improve the performance of anomaly event detection. Ablative studies based on two benchmark datasets UCF-Crime and ShanghaiTech have validated the effectiveness of the proposed model. Future work should focus more on effective feature extraction methods for improved anomaly event detection using new inputs [49] in edge computing environment [50–52].

Data Availability

The datasets used to support the findings of this study are available at https://webpages.uncc.edu/cchen62/dataset.html (UFC-Crime) and https://svip-lab.github.io/datasets.html (ShanghaiTech).

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was partially funded by Shanxi Agricultural University Young Science and Technology Innovation Programme (41257914) and Shanxi Key Research and Development Program (201703D221033-3).

References

[1] Y. Benezeth, P. Jodoin, V. Saligrama, and C. Rosenberger, "Abnormal events detection based on spatio-temporal co-occurences," in *Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami, FL, USA, June 2009.

[2] A. Adam, E. Rivlin, I. Shimshoni, D. Reinitz, and M. Intelligence, "Robust real-time unusual event detection using multiple fixed-location monitors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 555–560, 2008.

[3] Y. Cong, J. Yuan, and J. Liu, "Sparse reconstruction cost for abnormal event detection," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, June 2011.

[4] A. Basharat, A. Gritai, and M. Shah, "Learning object motion patterns for anomaly detection and improved object detection," in *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, June 2008.

[5] J. C. Duchi and E. Hazan, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[6] M. Hasan, J. Choi, J. Neumann, A. K. Roychowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," 2016, https://arxiv.org/abs/1604.04574.

[7] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Edmonton*, Alberta, Canada, July 2002.

[8] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence*, vol. 89, no. 1-2, pp. 31–71, 1997.

[9] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27 th International Conference on Machine Learning*, Haifa, Israel, 2010.

[10] X. Cui, Q. Liu, M. Gao, and D. N. Metaxas, "Abnormal detection using interaction energy potentials," in *Proceedings of the Computer Vision and Pattern Recognition*, Providence, RI, USA, June 2011.

[11] S. Ding, L. Lin, G. Wang, and H. Chao, "Deep feature learning with relative distance comparison for person re-identification," *Pattern Recognition*, vol. 48, no. 10, pp. 2993–3003, 2015.

[12] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, "Traffic monitoring and accident detection at intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 2, pp. 108–118, 2000.

[13] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," 2018, https://arxiv.org/abs/1801.04264.

[14] W. Luo, W. Liu, and S. Gao, "A revisit of sparse coding based anomaly detection in stacked RNN framework," in *Proceedings of the International Conference on Computer Vision*, Venice, Italy, October 2017.

[15] L. Kratz and K. Nishino, "Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, June 2009.

[16] B. Zhao, L. Feifei, and E. P. Xing, "Online detection of unusual events in videos via dynamic sparse coding," in *Proceedings of the Computer Vision and Pattern Recognition*, Providence, RI, USA, June 2011.

[17] S. Wu, B. E. Moore, and M. Shah, "Chaotic invariants of Lagrangian particle trajectories for anomaly detection in crowded scenes," in *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, June 2010.

[18] N. Li, H. Guo, D. Xu, and X. Wu, "Multi-scale analysis of contextual information within spatio-temporal video volumes for anomaly detection," in *Proceedings of the International Conference on Image Processing*, Paris, France, October 2014.

[19] B. Antic and B. Ommer, "Video parsing for abnormality detection," in *Proceedings of the International Conference on Computer Vision*, Barcelona, Spain, November 2011.

[20] H. Mobahi, R. Collobert, and J. Weston, "Deep learning from temporal coherence in video," in *Proceedings of the 26 th International Conference on Machine Learning*, Montreal, Canada, 2009.

[21] W. Li, V. Mahadevan, V. NJIToPA, and M. Intelligence, "Anomaly detection and localization in crowded scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 18–32, 2014.

[22] N. Li, X. Wu, H. Guo et al., "Anomaly detection in video surveillance via Gaussian process," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 29, no. 6, Article ID 1555011, 2015.

[23] K. Cheng, Y. Chen, and W. Fang, "Video anomaly detection and localization using hierarchical feature representation and Gaussian process regression," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, June 2015.

[24] R. Mehran, A. Oyama, and M. Shah, "Abnormal crowd behavior detection using social force model," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, June 2009.

[25] T. M. Hospedales, S. Gong, and T. Xiang, "A Markov clustering topic model for mining behaviour in video," in *Proceedings of the International Conference on Computer Vision*, Kyoto, Japan, October 2009.

[26] C. Wang, Z. Chen, K. Shang, and H. Wu, "Label-removed generative adversarial networks incorporating with *K*-Means," *Neurocomputing*, vol. 361, pp. 126–136, 2019.

[27] T. Meng, K. Wolter, H. Wu, Q. Wang, and M. Computing, "A secure and cost-efficient offloading policy for Mobile Cloud Computing against timing attacks," *Pervasive and Mobile Computing*, vol. 45, pp. 4–18, 2018.

[28] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 FPS in MATLAB," in *Proceedings of the International Conference on Computer Vision*, Sydney, Australia, December 2013.

[29] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection—a new baseline," 2018, https://arxiv.org/abs/1712.09867.

[30] Y. S. Chong and Y. H. Tay, "Abnormal event detection in videos using spatiotemporal autoencoder," 2017, https://arxiv.org/abs/1701.01546.

[31] W. Luo, W. Liu, and S. Gao, "Remembering history with convolutional LSTM for anomaly detection," in *Proceedings of the International Conference on Multimedia and Expo*, Hong Kong, China, July 2017.

[32] K. P. Adhiya, S. R. Kolhe, and S. S. Patil, "Tracking and identification of suspicious and abnormal behaviors using supervised machine learning technique," in *Proceedings of the International Conference on Advances in Computing, Communication and Control*, Mumbai India, January 2009.

[33] C. He, J. Shao, and J. Sun, "An anomaly-introduced learning method for abnormal event detection," *Multimedia Tools and Applications*, vol. 77, no. 22, pp. 29573–29588, 2018.

[34] C. Bergeron, J. Zaretzki, C. M. Breneman, and K. Bennett, "Multiple instance ranking," in *Proceedings of the 25 th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[35] T. Yao, T. Mei, and Y. Rui, "Highlight detection with pairwise deep ranking for first-person video summarization," in *Proceedings of the 016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016.

[36] J. Wang, Y. Song, T. Leung et al., "Learning fine-grained image similarity with deep ranking," 2014, https://arxiv.org/abs/1404.4661.

[37] M. Gygli, Y. Song, and L. Cao, "Video2GIF: automatic generation of animated GIFs from video," 2016, https://arxiv.org/abs/1605.04850.

[38] S. Sankaranarayanan, A. Alavi, and R Chellappa, *Triplet Similarity Embedding for Face Verification*, https://arxiv.org/abs/1602.03418, 2016.

[39] A. Gordo, J. Almazan, J. Revaud, and D. Larlus, "Deep image retrieval: learning global representations for image search," 2016, https://arxiv.org/abs/1604.01325.

[40] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," 2014, https://arxiv.org/abs/1406.2199.

[41] L. Wang, Y. Xiong, Z. Wang et al., "Temporal segment networks: towards good practices for deep action recognition," 2016, https://arxiv.org/abs/1608.00859.

[42] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," 2016, https://arxiv.org/abs/1604.06573.

[43] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal residual networks for video action recognition," 2016, https://arxiv.org/abs/1611.02155.

[44] H. Kwon, Y. Kim, J. S. Lee, and M. Cho, "First person action recognition via two-stream ConvNet with long-term fusion pooling," *Pattern Recognition Letters*, vol. 112, pp. 161–167, 2018.

[45] L. Sevillalara, Y. Liao, F. Guney, V. Jampani, A. Geiger, and M. J. Black, "On the integration of optical flow and action recognition," 2018, https://arxiv.org/abs/1712.08416.

[46] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3D residual networks," 2017, https://arxiv.org/abs/1711.10305.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016, https://arxiv.org/abs/1512.03385.

[48] J. Zhong, N. Li, W. Kong, S. Liu, T. H. Li, and G. Li, "Graph convolutional label noise cleaner: train a plug-and-play action classifier for anomaly detection," 2019, https://arxiv.org/abs/1903.07256.

[49] X. Xu, X. Liu, Z. Xu, F. Dai, X. Zhang, and L. Qi, "JIIoTJ. Trust-oriented IoT service placement for smart cities in edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4084–4091, 2019.

[50] X. Xu, X. Zhang, H. Gao, Y. Xue, L. Qi, and W. Dou, "BeCome: blockchain-enabled computation offloading for IoT in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4187–4195, 2020.

[51] X. Xu, C. He, Z. Xu, L. Qi, S. Wan, and Z. A. Bhuiyan, "JIIoTJ. Joint optimization of offloading utility and privacy for edge computing enabled IoT," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2622–2629, 2019.

[52] S. Li, "Zero trust based internet of things," *EAI Endorsed Transactions on Internet of Things*, vol. 5, no. 20, p. 6, 2020.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 2132138, 12 pages
https://doi.org/10.1155/2020/2132138

Research Article

Wearable Sensor-Based Human Activity Recognition Using Hybrid Deep Learning Techniques

Huaijun Wang,[1,2] Jing Zhao,[1] Junhuai Li,[1,2] Ling Tian,[1] Pengjia Tu,[1] Ting Cao,[1,2] Yang An,[1] Kan Wang,[1,2] and Shancang Li[3]

[1]*School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China*
[2]*Shaanxi Key Laboratory for Network Computing and Security Technology, Xi'an 710048, China*
[3]*Department of Computer Science and Creative Technologies, UWE Bristol, Bristol BS16 1QY, UK*

Correspondence should be addressed to Junhuai Li; lijunhuai@xaut.edu.cn

Received 16 February 2020; Revised 8 June 2020; Accepted 6 July 2020; Published 27 July 2020

Academic Editor: Xiaolong Xu

Human activity recognition (HAR) can be exploited to great benefits in many applications, including elder care, health care, rehabilitation, entertainment, and monitoring. Many existing techniques, such as deep learning, have been developed for specific activity recognition, but little for the recognition of the transitions between activities. This work proposes a deep learning based scheme that can recognize both specific activities and the transitions between two different activities of short duration and low frequency for health care applications. In this work, we first build a deep convolutional neural network (CNN) for extracting features from the data collected by sensors. Then, the long short-term memory (LTSM) network is used to capture long-term dependencies between two actions to further improve the HAR identification rate. By combing CNN and LSTM, a wearable sensor based model is proposed that can accurately recognize activities and their transitions. The experimental results show that the proposed approach can help improve the recognition rate up to 95.87% and the recognition rate for transitions higher than 80%, which are better than those of most existing similar models over the open HAPT dataset.

1. Introduction

Human behavior recognition (HAR) is the detection, interpretation, and recognition of human behaviors, which can use smart heath care to actively assist users according to their needs. Human behavior recognition has wide application prospects, such as monitoring in smart homes, sports, game controls, health care, elderly patients care, bad habits detection, and identification. It plays a significant role in depth study [1] and can make our daily life become smarter, safer, and more convenient.

Currently, human behavior data can be acquired in two ways: one is based on computer vision and the other is based on sensors [2]. Behavior recognition based on computer vision has been studied for a long time and has a mature theoretical basis. However, the vision-based approaches have many limitations in practice. For example, the use of a camera is limited by various factors, such as light, position, angle, potential obstacles, and privacy invasion issues, which make it difficult to be restricted in practical application. Although the research time of sensor-based behavior recognition is relatively short, with the development and maturity of microelectronics and sensor technology, there are various types of sensors, such as accelerometers, gyroscopes, magnetometers, and barometers. These sensors can be integrated into mobile phones and wearable devices such as watches, bracelets, and clothes. Furthermore, state-of-the-art wearable sensors have solved the issue of antimagnetic field interference, such as [3], which can accurately estimate the current acceleration and angular velocity of motion sensors in real time in the presence of magnetic field interference. So these wearable sensors are usually small in size, high in sensitivity, and strong in anti-interference ability, so the sensor-based identification method is more suitable for practical situations. Moreover, sensor-based behavior recognition is not limited by scene or time, which

can better reflect the nature of human activities. Therefore, the research and application of human behavior recognition based on sensors are more and more valuable and significant.

Besides, the HAR includes two types: basic actions and transition actions. Due to the low incidence and short duration of transition movement, there are relatively few studies on the transition movement from standing to sitting, walking to standing, and so on in the research of human behavior recognition [4]. However, the study of transitional movement is a very important part of human behavior recognition. In order to improve the behavior recognition rate, transition action recognition is not negligible. The transition action is the distinction of a variety of basic actions in frequent alternations. The accurate division of the transition action can accurately segment the streaming data to a certain extent and ultimately improve the recognition rate. In addition, the behavior recognition methods based on traditional patterns have shortcomings such as manual feature extraction. With the application and development of deep learning in different fields, the deep learning model also shows great advantages in the field of behavior recognition.

The main contributions of this work are summarized as follows:

(1) We presented a deep learning model composed of convolutional and Long Short-Term Memory recurrent layers, which can automatically learn local features and model the time dependence between features.

(2) We discussed the influence of key parameters in deep learning model on performance and finally determined the best parameters in the model.

(3) We analyzed and compared the experimental results with other models that adopt the same common data set. The results show that the proposed method is superior to the other advanced methods.

In this work, we use both acceleration sensor and a gyroscope sensor of smart phones to acquire data and proposed a CNN-LSTM hybrid model to recognize the transition motion. Convolution neural network (CNN) [5] is a type of depth neural network used as a feature extractor. It is characterized by local dependence, so it has good performance in extracting local features. However, human activity information belongs to long instance, which is composed of complex movements and changes with time. So the CNN model does not work well in extracting the relationship between time and features. The Long Short-Term Memory (LSTM) [6] neural network is a kind of recursion network that contains a memory to simulate a time dependent sequence problem. Therefore, the mixture of CNN-LSTM can accurately identify the basic and transitional features of activities.

The remainder of the paper is organized as follows: Section 2 reviews the literature on human activity identification based on deep learning and existing problems; Section 3 presents the mixed deep learning framework proposed in this paper for existing problems; Section 4 discusses and analyzes the experimental results based on experimental data. Finally, Section 5 concludes this paper.

2. Related Works

Due to the extensive application of human-computer interaction, behavior detection, and other technologies, human behavior recognition has become a hot field [7]. Human behavior recognition can be regarded as a representative pattern recognition problem. The traditional pattern of behavior recognition research using decision tree, support vector machine (SVM), and other machine learning algorithms can obtain much satisfactory results, in premise of some controlled experimental environments and a small number of labeled data. However, the accuracy of these methods depends on the effectiveness and comprehensiveness of manual feature extraction. In addition, these methods can only extract shallow features. Because of these limitations, the behavior recognition methods based on traditional pattern recognition are limited in classification accuracy and model generalization.

In recent years, deep learning has developed rapidly and attracted many research efforts, especially in image, processing time series, natural language, logical reasoning, and other complex data processing aspects and has achieved unparalleled achievements [8]. Different from the traditional behavior recognition method, deep learning could reduce the workload of feature design. In addition, the higher-level and more meaningful features can be learned via the end-to-end neural network. Furthermore, the deep network structure is more suitable for unsupervised incremental learning. Moreover, deep networks created by superimposing several layers of features can model data with complex structures. In a word, the deep learning is an ideal method for HAR.

Since deep learning has made outstanding achievements in image feature extraction, many researchers first try to apply it to behavior recognition based on video. In early periods, Taylor et al. [9] used convolution threshold *Boltzmann* machine to identify video behavior data and extract sensitive features. Ji et al. [10] proposed a three-dimensional CNN model to capture more action information from space and time. Liu et al. [11] proposed that CNN and conditional random domains (CRFs) be combined for action segmentation and recognition. The CNN can automatically learn space-time characteristics, while CRF is able to capture the dependency between outputs. Other common deep learning methods are also widely used, such as recursive neural network [12] and long short-term memory network. On one hand, it is successful on application of deep learning in video behavior recognition. On the other hand, it is also widely used in human behavior recognition based on sensors.

Zeng et al. [13] proposed treating the single-axis sensor data as one-dimensional data of images and then sending them to CNN for identification. Jiang and Yin [14] combined the signal sequences of accelerometer and gyroscope into an active image, enabling deep convolutional neural network (DCNN) to automatically learn the optimal features

from the active image. Chen and Xue [15] modified the CNN convolution kernel to adapt to the characteristics of triaxial acceleration signals. Ronao and Cho [16] proposed a convNet, which realized efficient and data adaptive human behavior recognition with smart phone sensors. ConvNets not only utilize the inherent time-local dependence of sensor signal sequences but also provide an adaptive method for extracting robust features. Experimental results show that this method can recognize similar actions, which are difficult to be processed by traditional machine learning. Murad and Pyun [17] and Zhou et al. [18] proposed three deep recursive neural network structures based on LSTM to establish recognition models to capture time relations in input sequences and could achieve more accurate recognition. Due to the superior performance of LSTM in behavior recognition application, Guan and Plötz [19] and Qi et al. [20] improved the LSTM and proposed an integration model, integrating different LSTM learners into an integrated classifier. Through the experimental evaluation in the standard data set, it is proved that the integrated system composed of LSTM learners is superior to a single LSTM network. Ignatov [21] combined the manually extracted statistical features with the features automatically extracted by neural network and realized a human behavior recognition method based on user autonomous deep learning. Among them, CNN extracted local features, while statistical features preserved the information about the global form of time series. Experiments on open data sets show that the model has the advantages of small computation, short running time, and good performance. Nweke et al. [22] and Wang et al. [23], respectively, summarized the application of deep learning method in sensor-based behavior recognition and not only put forward detailed views on the existing work, but also pointed out the challenges and improvement directions of future research.

This work demonstrated the potential of deep neural network to learn the potential features and time series features. Nevertheless, existing works on action recognition mainly focus on the aspect of basic behavior recognition, while the transition between actions is usually ignored because the transition action has a short duration. However, it is necessary to study the transition action in depth in order to improve the robustness of the model. The precise division of the transition action can accurately segment the streaming data to a certain extent and ultimately improve the recognition rate. In this paper, CNN combined with LSTM hybrid model is adopted to extract deep and advanced features, and elaborate description is made of basic and transition action, so as to realize accurate identification.

3. Proposed Method

The overall architecture diagram of the method proposed in this paper is shown in Figure 1, which contains three parts. The first part is the preprocessing and transformation of the original data, which combines the original data such as acceleration and gyroscope into an image-like two-dimensional array. The second part is to input the composite image into a three-layer CNN network that can automatically extract the motion features from the activity image and abstract the features, then map them into the feature map. The third part is to input the feature vector into the LSTM model, establish a relationship between time and action sequence, and finally introduce the full connection layer to achieve the fusion of multiple features. In addition, Batch Normalization (BN) is introduced [24], in which BN can normalize the data in each layer and finally send it to the S*oftmax* layer for action classification.

3.1. Data Preprocessing. Due to the large amount of behavioral data collected by the sensor, it is impossible to input all the data into the depth model at one time. Therefore, sliding window segmentation should be carried out before data input into the model. The behavior recognition method proposed in this paper can recognize both the basic action and the transition action at the same time. The transition action lasts for a short time; it is necessary to choose the appropriate window size. If the window is too large, important information will be lost. Otherwise, the computational costs will be increased. After data segmentation, the behavioral data collected by sensors are one-dimensional time series different from image data. Therefore, before applying the deep learning model to these input data, it is necessary to input and adapt them. Dimension transformation is carried out on the data after window segmentation. The method of transformation is to splice the sensor data of all axes into a two-dimensional matrix. The advantage of this approach to data processing is that it preserves the correlation between sensors' axes. Finally, samples similar to pictures are formed and input into the deep learning model. Figure 2 shows the model structure of data preprocessing.

3.2. Feature Learning Based 1D-CNN. The original uniaxial acceleration and gyroscope data are equivalent to two-dimensional array of images after dimensional transformation. The feature image is input into the convolution neural network, and its structure is generally composed of convolution layer and pooling layer. The convolution layer carries out convolution operation on the input image through convolution kernel to obtain feature mapping. The pooling layer extracts local features from the feature map of the convolution layer through sampling operation to lessen the size of neurons and the number of parameters. The convolution layer and pooling layer are stacked to form a deep structure, which can automatically extract the action feature information from the original action data [5].

The CNN model structure designed in this paper is shown in Figure 3. The CNN network model consists of three convolution layers and three pooling layers (each convolution layer is followed by one pooling layer) and finally outputs a number of feature map images with action features. Table 1 illustrates the settings of different parameters for each convolution and pooling layer. Convolution is achieved by the convolution of two-dimensional convolution kernel with images superimposed by multiple adjacent frames. The convolution kernel number of the three convolution layers is 18, 36, and 72, respectively. The

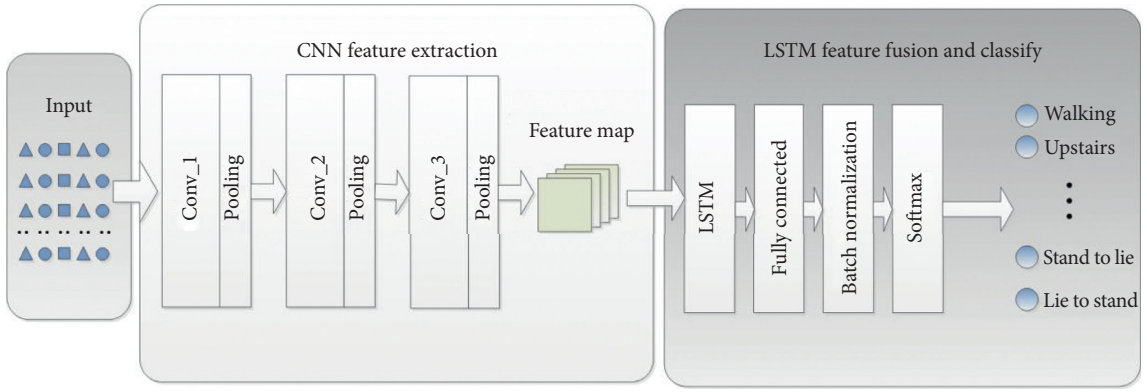


Figure 1: Human activity recognition framework based on CNN-LSTM.

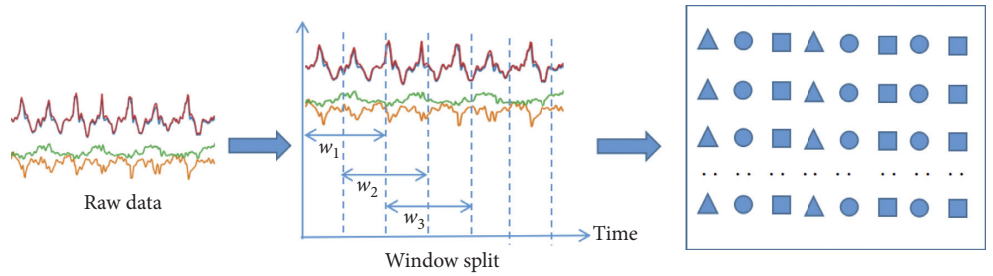


Figure 2: Structure of data preprocessing model.

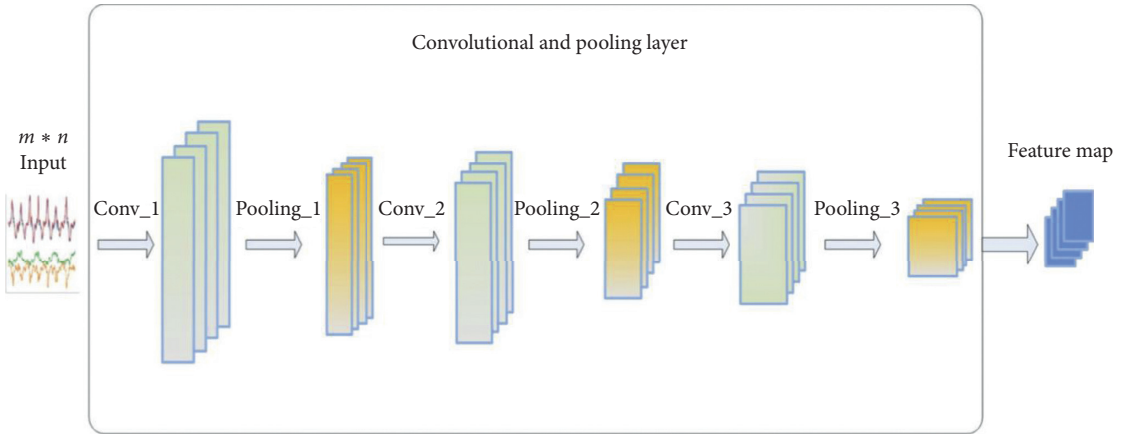


Figure 3: CNN model architecture.

Table 1: Activity label corresponding to the original data.

Id	Exp	Label	Start	End
1	1	5	250	1232
1	1	7	1233	1392
1	1	4	1393	2194
1	1	8	2195	2359
1	1	5	2360	3374
1	1	11	3375	3662
1	1	5	3663	4538
1	1	11	4539	4735
1	1	5	4736	5667
1	1	11	5668	5859
1	1	5	5860	6786
1	1	11	6787	6977
1	1	5	6978	8078

convolution kernel size is 2×8, 2×18, and 2×36, and the step size is 1. Since the filter may not be able to process the data in a certain direction in the operation of convolution, to avoid reducing data of the image edge, the padding parameter is introduced and set to "SAME" and 0 is added to the edge of the input image matrix. After the convolution operation in the convolution layer, the output will usually pass through a nonlinear activation function and then form the output of the convolution layer. The popular activation functions include *Sigmoid* function, *ReLU* function, and Tanh function. Among them, *ReLU* function can change the negative value of the data extracted by CNN into 0, and the positive value of the data greater than 0 remains unchanged. After nonlinear processing operation, the positive value

greater than 0 can be more clearly expressed by the extracted features. Therefore, *ReLU* activation function is used in the convolution layer of CNN:

$$f(x) = \max(0, x) = \begin{cases} 0, & x < 0, \\ x, & x \geq 0. \end{cases} \tag{1}$$

Further, we have

$$f'(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0. \end{cases} \tag{2}$$

Pooling layer is regarded as reducing the number of feature mappings and parameters. The popular pooling techniques include maximum pooling and average pooling. In recent years, relevant theoretical analysis and performance evaluation have shown the superior performance of the maximum pooling strategy, which is widely used in deep learning [25, 26]. Moreover, some studies show that the maximum pooling technology is very suitable for sensor-based human behavior recognition [27]. Therefore, all pooling layers of CNN in this paper utilized the maximum pooling technique. Specific convolution and pooling process parameters are set as shown in Table 2.

3.3. Feature Fusion and Action Classification. To improve the recognition rate of transition actions, we build a LSTM after the CNN network $\{f_1, f_2 \ldots\ldots f_n\}$ is the feature sequence converted from the feature map calculated by CNN from the images composed of original data. Therefore, the sequence $\{f_1, f_2 \ldots\ldots f_n\}$ input LSTM and the storage unit of LSTM will produce a sequence of characters $\{m_1, m_2 \ldots\ldots m_n\}$.

Since LSTM has different gating units, memory units such as input gate, forgetting gate, and output gate are combined with learning weights to solve the problem of gradient disappearance in the process of back propagation of ordinary circular neural network. Meanwhile, LSTM can model time-dependent actions and fully capture global features, so as to improve the recognition accuracy [28]. LSTM cell controls the inward flowing information of neurons, which is composed of forgetting gate, input gate, and output gate. Furthermore, the predicted value of LSTM cell is obtained using Tanh function.

Firstly, the forgetting gate determines how much information from the previous moment can be accumulated to the current cell. As shown in equation (3), the probability value is calculated to determine the amount of information that can pass through the gate:

$$\Gamma_f = \sigma\left(w_f * \left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_f\right), \tag{3}$$

where w_f represents the weight corresponding to the input vector, b represents the bias, $a^{\langle t-1\rangle}$ presents the output of the neuron at the last moment, and $x^{\langle t\rangle}$ represents the current input of the neuron.

Secondly, the input gate consists of update gate and Tanh layer, which controls how much information can flow into the current cell. The calculation process is shown in equations (4)–(6). The input of the input gate and the output of the forgetting gate update the cell at the same time, discarding unwanted information. Then, the predicted value of the current unit is determined by the output gate, and the output of the model is obtained, as shown in equations (7) and (8):

$$\Gamma_u = \sigma\left(w_u * \left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_u\right), \tag{4}$$

$$\widetilde{C} = \tanh\left(w_c * \left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_c\right), \tag{5}$$

$$C_t = \Gamma_u * \widetilde{C}^{\langle t\rangle} + \Gamma_f * C^{\langle t-1\rangle}, \tag{6}$$

$$\Gamma_o = \sigma\left(w_o * \left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_o\right), \tag{7}$$

$$a^{\langle t\rangle} = \Gamma_o * \quad \tanh\left(C^{\langle t\rangle}\right). \tag{8}$$

After the processing of LSTM layer, the final output is a set of vectors containing time and action sequence correlation, which are input into the full connection layer for the fusion of global action features. The training process of neural network model becomes complicated since the statistical distribution of input of each layer changes with the parameters of the previous layer. To keep the distribution of output data from changing too much, a lower learning rate will be used, which could reduce the training speed. To solve this issue, this paper introduces the BN to standardize the values of each layer in LSTM (the output of neurons at the last moment and the input at the current moment), so that the mean and variance of sum will not change with the change of the distribution of the underlying parameters and effectively separate the parameters of each layer from other layers. In this way, the gradient disappearance or explosion can be prevented and the training speed of the network can be accelerated. The BN algorithm is shown in Algorithm 1.

In Algorithm 1, μ_x and ς_x^2 are the mean and variance of x_i obtained through minibatch. The mean and variance were used to normalize x_i to make the sample follow normal distribution. However, the positive distribution is not able to reflect the characteristic distribution of the training samples, and thus it is necessary to introduce the scaling factor γ and the shift factor β. As training progresses, γ and β are also learned by back propagation to improve accuracy.

After BN operation, the features are more obvious, so input them to *Softmax* layer to extract the action features and classify them in time series. In this model, the output layer uses *Softmax* normalized exponential function to calculate the posterior probabilities of different actions to

Table 2: The convolution and pooling layers of the CNN architecture.

Layers	Conv1d_1	Conv1d_2	Conv1d_2
Size	$1 \times 2 \times 8$	$1 \times 2 \times 18$	$1 \times 2 \times 36$
Stride	$1 \times 1 \times 1$	$1 \times 1 \times 1$	$1 \times 1 \times 1$
Channel	18	36	72
Layers	Pooling_1	Pooling_2	Pooling_3
Size	$1 \times 2 \times 18$	$1 \times 2 \times 36$	$1 \times 2 \times 72$
Stride	$1 \times 1 \times 1$	$1 \times 1 \times 1$	$1 \times 1 \times 1$
Channel	18	36	72

Input: data set: $\chi = \{x_1 \ldots x_n\}$
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$
(1) Calculate the mean of data set: $\mu_x \leftarrow (1/n)\sum_{i=1}^{n} x_i$
(2) Calculate the variance of data set: $\varsigma_\chi^2 \leftarrow (1/n)\sum_{i=1}^{n}(x_i - \mu_\chi)^2$
(3) Normalize data: $\widehat{x}_i \leftarrow (x_i - \mu_x/\sqrt{\varsigma_x^2 + \varepsilon})$
(4) Scale change and deviation: $y_i \leftarrow \gamma\widehat{x}_i + \beta = \mathrm{BN}_{\gamma,\beta}(x_i)$
(5) Return learning parameter γ and β

Algorithm 1: Algorithm of batch normalization.

realize classification. It maps the output values of neurons between (0, 1), which can be regarded as the prediction probability of actions, and the largest one is the result of classification. Then the *Softmax* output layer outputs a category vector such as [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0], indicating that the classification result is an action numbered 5.

3.4. Model Implementation and Training. The neural network described here is implemented in TensorFlow [29]. It is a lightweight library for building and training neural networks. Model training and classification runs on a conventional computer with a 2.4 GHz CPU and 16 GB memory.

The model is trained in a fully supervised manner to backpropagate the gradient from the *Softmax* layer to the convolution layer. Network parameters are optimized by using minibatch gradient descent method and Adam optimizer through minimizing cross-loss function [13]. Adam is widely used due to its advantages in simple implementation, efficient calculation, and low memory demand. Compared with other kinds of random optimization algorithms, Adam has great advantages. In this paper, to better train the model, after the training data are input into the network. Adam optimizer and backpropagation algorithm are used to learn and optimize the network parameters. Meanwhile, the cross-entropy loss function is used to calculate the total error, as shown in the following equation:

$$C = -\frac{1}{N}\sum_{X}[y\mathrm{In}a + (1-y)\mathrm{In}(1-a)], \tag{9}$$

where y is the true tag and a is the predicted value.

To improve efficiency, small batches of data segment size are segmented during training and testing. With these configurations, the cumulative gradient of the parameters is calculated after each small batch. The weights are randomly and orthogonally initialized. As a form of regularization, we introduce a dropout operator on each dense layer of input. This operator sets the activation of a randomly selected unit to zero during training. Dropout technology proposed by Hinton et al. [30] is based on the principle of randomly deleting some nodes in the network while maintaining the integrity of input and output neurons, which is equivalent to training many different networks. Different networks may overfit in different ways, but their average results can effectively reduce overfitting. In addition, dropout allows neurons to learn stronger features by not relying on other specific neurons. The number of parameters to be optimized in a deep neural network varies depending on the type of layer it contains. And it has a great impact on the time and computer skills required to train the network. The specific model training parameters will reflect the best choices in the experiment.

4. Activity Recognition

4.1. Experiment Data. In addition to common basic actions, this paper also studies transition actions. Actually, a few existing public data sets contain transition actions. Therefore, this paper adopts the international standard Data Set, Smart phone Based Recognition of Human Activities and Postural Transitions Data Set [31, 32] to conduct an experiment, which is abbreviated as HAPT Data Set. The data set is an updated version of the UCI Human Activity Recognition Using popularity Data set [8]. It provides raw data from smart phone sensors rather than preprocessed data. In addition, the action category has been expanded to include transition actions. The HAPT data set contains twelve types of actions. Firstly, it has six basic actions that include three types of static actions, such as standing, sitting, and lying, and three types of walking activities such as walking, going downstairs, and upstairs; Secondly, it has six possible transitions between any two static movements: standing to sitting, sitting to standing, standing to lying, lying to sitting, sitting to lying, and lying to standing.

The HAPT data collection process is shown in Figure 4. The experiment involved 30 volunteers, whose ages range from 19 to 48, each wearing a smart phone on their waist. Data collection is carried out with the built-in acceleration sensor and gyroscope, and the sampling frequency is 50 Hz. Meanwhile, video records of the experimental process are made for the convenience of subsequent data marking.

The collected data is saved in the form of .txt, and the acceleration and gyroscope data are stored independently, with 60 groups, respectively. As shown in Table 1, it is the label information corresponding to the original data of the experiment. Among them, the first column is the experiment ID, the second column is the experimenter number, the third column is the action label, and the fourth and fifth columns are the start and end row labels of the corresponding sensor data. The label ranges from 1 to 12, representing 12 types of actions. It can be seen from the figure that the collected data contains invalid data, and the first 250 pieces of data are unlabeled and belong to invalid data.

Figure 4: Data collection of the physical activities.

Table 3: The data amount of various activities in the HAPT.

Type	ID	Number
Walk	*A*1	122,091
Upstairs	*A*2	116,707
Downstairs	*A*3	107,961
Sit down	*A*4	126,677
Stand	*A*5	138,105
Lie	*A*6	136,865
Stand to sit	*A*7	10,316
Sit to stand	*A*8	8,029
Sit to lie	*A*9	12,428
Lie to sit	*A*10	11,150
Stand to lie	*A*11	14,418
Lie to stop	*A*12	10,867

After preliminary processing of the original data, all the data without labels were deleted. Finally, 815,614 valid pieces of data were obtained. Due to the low frequency and short duration of transition action, as well as the high frequency and long duration of basic action, there is a considerable difference in data volume between transition action and basic action. The data volume of the six transition actions is much lower than that of the other basic actions, accounting for only about 8% of the total data. Table 3 lists the amount of data for different actions. The original data is divided into three parts, training set, verification set, and test set, in which the training set is used for model training, and verification set is used to adjust parameters, and test set is used to measure the quality of the final model.

4.2. Parameters Setting. In the deep learning network, the model parameters greatly affect its recognition rate. Therefore, the experimental analysis of the number of neurons, learning rate, BN, Batch size, and other parameters in LSTM layer would be conducted in the following sections.

4.2.1. Number of Neurons in LSTM Layer. In order to verify the influence of the number of neurons in LSTM layer on the recognition results, the following experiments are carried out in this paper, as shown in Figure 5. It shows that the recognition rate is the lowest when each LSTM layer contains only 8 neurons. This is because, given less neurons, the network lacks the necessary learning ability and information processing ability, resulting in the low recognition rate. As

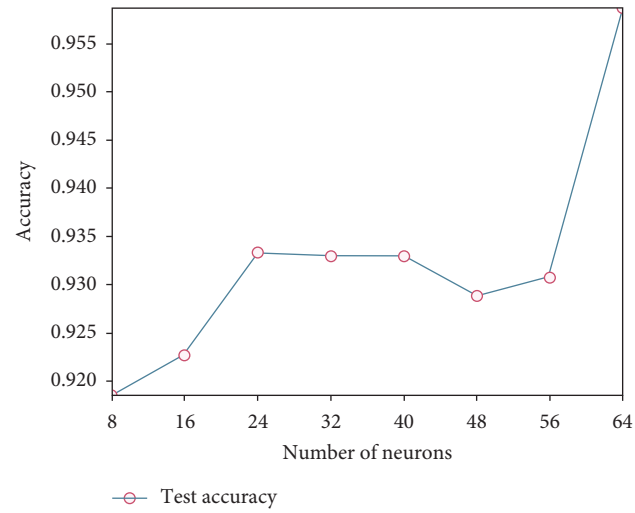


Figure 5: Accuracy of different numbers of neurons on test sets.

the number of neurons increases, the recognition rate tends to increase. When the number of neurons is 64, the recognition rate reaches 95.87%. If the number of neurons is too large, the complexity of network structure will increase and the learning speed of network will slow down. Therefore, considering the training time of the network, the number of LSTM layer neurons in this paper is tentatively 64.

4.2.2. The Learning Rates. Experiments are carried out at different learning rates in this paper. As shown in Table 4, it can be seen that the recognition rate of the model reaches a maximum of 95.87% when the learning rate is 0.002. Therefore, the learning rate of 0.002 is adopted.

4.2.3. BN Operation. To verify the improvement of the BN operation on the network model, a comparative experiment is carried out first with and without BN layer. The epoch is set to 400, and other parameters remain unchanged. The recognition rates of both methods on the test set are shown in Table 5. Obviously, the recognition rate on the test set is improved by about 4.24% after the BN layer is added.

4.2.4. Batch Size. Batch size refers to the Batch sample size, whose maximum value is the total number of samples in the

Table 4: Accuracy of different learning rates on test sets.

Learning rate	Recognition rate (%)
0.001	93.57
0.0015	94.21
0.002	95.87
0.0025	92.39
0.003	93.34
0.0035	92.12
0.004	92.84
0.0045	92.01

Table 5: Accuracy and loss rate on test sets with or without BN layer.

	Recognition rate (%)
Without BN layer	91.63
With BN layer	95.87

Table 6: Accuracy of different batch size on test sets.

Batch size	Recognition rate (%)
25	91.74
50	92.88
75	92.92
100	93.10
125	94.33
150	95.87
175	93.45
200	93.37
225	93.72
250	93.45
275	92.84
300	93.35
325	94.06
350	93.34
375	92.96
400	93.53

training set. When the amount of data is small, the batch data is the whole data set, so that it can approach the extreme value direction more accurately. However, in practical applications, the amount of data used by deep learning is relatively large, and the principle of small batch processing is generally adopted. Using small batch processing requires relatively little memory and faster training time. Within an appropriate range, increasing the batch size can more accurately determine the direction of gradient descent and cause less training shock. However, when the batch size increases to a certain value, the determined downward direction will not change and the correction of parameters will slow down significantly. The identification results of different batch sizes are shown in Table 6. It can be seen that when the batch size is 150, the maximum identification rate reaches 95.87%. Therefore, 150 is selected as the best batch size in this paper.

The parameters of the CNN-LSTM model proposed in this paper are shown in Table 7.

Table 7: Experimental parameters of CNN-LSTM model.

Parameters	Value
Input vector size	150
Input channel number	8
Convolution kernel size	3
Pool size	2
Activation function	ReLu
LSTM layer	1
Neurons number	64
Dropout	0.5
Learning rate	0.002
Batch size	150
Epoch	400

5. Experimental Results and Analysis

For human movement recognition, Wang and Liu [33] proposed to use the F-measure standard measurement method to verify the performance of the deep-rooted LSTM network model in human activity recognition. Lu et al. [34] demonstrated the superiority of the model in behavior recognition by using accuracy, prediction rate, and recall rate in the experiment. Therefore, to evaluate the performance of the motion recognition method proposed in this paper, we also used the measurement method of accuracy, recall rate, loss rate, and F-measure in the experiment.

According to the above parameters, the recognition confusion matrix of 12 different actions is shown in Table 8. Accuracy curve of CNN-LSTM model is shown in Figure 6. It can be seen from Table 9 that the overall recognition rate of CNN-LSTM is high, and the CNN-LSTM has a better recognition effect on the transition action.

Table 8: Confusion matrix of various actions.

Actual	Predict											
	*A*1	*A*2	*A*3	*A*4	*A*5	*A*6	*A*7	*A*8	*A*9	*A*10	*A*11	*A*12
*A*1	410	1	3	0	0	0	0	0	0	0	0	0
*A*2	5	388	3	0	0	0	0	0	0	0	0	0
*A*3	1	3	346	0	0	0	0	0	0	0	0	0
*A*4	1	0	0	383	32	3	1	0	1	1	0	0
*A*5	0	0	1	31	431	0	0	0	0	0	0	0
*A*6	0	0	0	1	0	457	0	0	0	0	0	0
*A*7	0	0	0	1	0	0	17	0	0	0	0	0
*A*8	0	0	0	0	0	0	0	4	0	0	1	0
*A*9	0	0	0	0	0	0	0	0	19	1	4	1
*A*10	0	0	0	0	0	0	0	0	1	14	0	2
*A*11	0	1	0	1	0	0	1	0	2	1	32	1
*A*12	0	0	0	0	0	0	0	0	0	1	1	16

6. Case Study

In the non-deep-learning method, the random forest classification method (RF) and K-nearest neighbor (KNN) classification perform well in action classification recognition. Therefore, the CNN-LSTM model proposed is compared with the RF and KNN methods. First of all, input the HAPT data set into RF and KNN. Then, segment the original

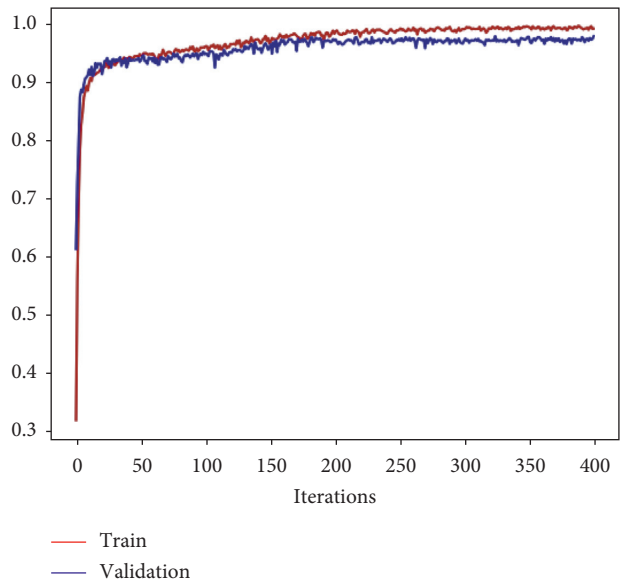


Figure 6: Accuracy curve of CNN-LSTM Model.

Table 9: The recognition accuracy, recall rate, and F value of various actions.

ID	Accuracy (%)	Recall (%)	F-measure (%)
$A1$	99.03	98.32	98.68
$A2$	97.78	98.73	98.35
$A3$	98.86	98.02	98.44
$A4$	90.76	91.85	91.30
$A5$	93.09	93.09	93.09
$A6$	99.78	99.56	99.56
$A7$	94.44	89.47	91.89
$A8$	100	100	100
$A9$	76.00	82.61	79.17
$A10$	82.35	77.78	80.00
$A11$	82.05	86.49	84.21
$A12$	88.89	80.00	84.21

Table 10: Average accuracy of various actions in CNN-LSTM, RF, and KNN models.

ID	RF (%)	KNN (%)	CNN-LSTM (%)
$A1$	99.90	88.10	99.03
$A2$	92.50	97.80	97.78
$A3$	90.20	99.40	98.86
$A4$	91.90	83.80	90.76
$A5$	90.80	87.50	93.09
$A6$	97.10	100	99.78
$A7$	71.30	66.70	94.44
$A8$	72.00	68.00	100
$A9$	51.30	38.60	76.00
$A10$	74.90	36.30	82.35
$A11$	59.20	33.70	82.05
$A12$	61.10	57.90	88.89

sensor data and calculate the mean value, variance, covariance, and 15 features. Finally, classify the basic actions and transition actions according to the clustering results. The classification results are shown in Table 10. It can be seen that the recognition rate of CNN-LSTM model is higher than that of RF and KNN methods for both basic actions and transition actions.

In addition to the comparison with RF and KNN classifier, our proposed model is also compared with a single CNN, a single LSTM, CNN-GRU, and CNN-BLSTM deep

Table 11: Average accuracy of different activities with five deep learning models.

ID	CNN (%)	LSTM (%)	CNN-BLSTM (%)	CNN-GRU (%)	CNN-LSTM (%)
*A*1	97.50	97.70	97.41	99.75	99.03
*A*2	97.25	97.10	95.65	98.99	97.78
*A*3	95.60	97.15	100	96.57	98.86
*A*4	91.26	90.26	91.96	81.99	90.76
*A*5	90.80	90.80	84.74	92.48	93.09
*A*6	99.67	98.58	100	99.78	99.78
*A*7	76.47	64.86	44.44	77.78	94.44
*A*8	100	66.67	66.67	50.00	100
*A*9	63.83	69.39	62.07	48.00	76.00
*A*10	84.85	70.27	80.00	52.94	82.35
*A*11	72.50	69.33	65.00	71.79	82.05
*A*12	83.30	70.27	70.59	55.56	88.89

Table 12: Average accuracy of the five models in this paper.

Method	Average recognition rate (%)
CNN	94.29
LSTM	93.22
CNN-BLSTM	92.73
CNN-GRU	93.34
CNN-LSTM	95.87

Table 13: Average accuracy of different methods on test set in the paper [35, 36].

Method	Average recognition rate
BLSTM [35]	87.5
DBN [36]	89.6
CNN-LSTM	95.8

learning models. Table 11 shows the average accuracy of various actions in five different depth models. As can be seen from Table 11, CNN-LSTM not only has a slightly higher recognition of basic movements than the other five models, but also has a significantly better recognition of transition movements, especially standing to sitting, sitting to lying, and standing to lying. Table 12 shows the recognition rates of different models on the test set. It can be seen from the table that the average recognition rate of the three models is higher than 90%, but the recognition effect of CNN-LSTM model is slightly better than that of CNN, LSTM, CNN-GRU, and CNN-BLSTM.

To prove the effectiveness of the CNN-LSTM deep learning model, it is also compared with other deep learning methods using the same dataset. Kuang [35] applied BLSTM to construct the behavior recognition model. Hassan et al. [36] used deep belief network (DBN) for human behavior recognition. We compared the performance with the approaches in [35, 36], with the result shown in Table 13. It follows that the proposed CNN-LSTM can achieve highest average recognition rate.

7. Conclusion

This paper explored the recognition method based on deep learning and designed the behavior recognition model based on CNN-LSTM. CNN learns local features from the original sensor data, and LSTM extracts time-dependent relationships from local features and realizes the fusion of local features and global features, fine description of basic and transition movements, and accurate identification of the two motion patterns.

The actions identified in this paper only include common basic actions and individual transition actions. In the next step, more kinds of actions can be collected and more complex actions can be added, such as eating and driving. And the individual recognition can be realized by considering the behavior differences of different users. Meanwhile, the deep learning model still needs to be optimized and improved. Studies show that the combination of depth model and shallow model can achieve better performance. Deep learning model has strong learning ability, while shallow learning model has higher learning efficiency. The collaboration between the two can achieve more accurate and lightweight recognition.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

The authors would like to thank the support of the laboratory, university, and government. This research was funded by the National Key Research and Development Plan (No. 2017YFB1402103), the National Natural Science Foundation of China (No. 61971347), Scientific Research Program of Shaanxi Province (2018HJCG-05), and Project of Xi'an Science and Technology Planning Foundation (201805037YD15CG214).

References

[1] I. H. Lopez-Nava and M. M. Angelica, "Wearable inertial sensors for human motion analysis: a review," *IEEE Sensors Journal.*vol. 16, no. 15, 2016.

[2] Y. Liu, L. Nie, L. Liu, and D. S. Rosenblum, "From action to activity: sensor-based activity recognition," *Neurocomputing*, vol. 181, pp. 108–115, 2016.

[3] T. Liu, F. Bingfei, and L. Qingguo, "The invention relates to a wearable motion sensor and a method for resisting magnetic field interference," 2017.

[4] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[5] F. J. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors (Switzerland)*, vol. 16, p. 1, 2016.

[6] X. Du, R. Vasudevan, and M. Johnson-Roberson, "Bio-LSTM: a biomechanically inspired recurrent neural network for 3-d pedestrian pose and gait prediction," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1501–1508, 2019.

[7] Y. Huang, C. Wan, and H. Feng, "Multi-feature fusion human behavior recognition algorithm based on convolutional neural network and long short term memory neural network," *Laser Optoelectron. Prog*, vol. 56, p. 7, 2019.

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

[9] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer, Berlin, Germany, 2010.

[10] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[11] C. Liu, J. Liu, Z. He, Y. Zhai, Q. Hu, and Y. Huang, "Convolutional neural random fields for action recognition," *Pattern Recognition*, vol. 59, pp. 213–224, 2016.

[12] K. Cho, M. Bart van, G. Caglar et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, Doha, Qatar, October 2014.

[13] M. Zeng, T. N. Le, Y. Bo et al., "Convolutional Neural Networks for human activity recognition using mobile sensors," in *Proceedings Of the 2014 6th International Conference On Mobile Computing, Applications And Services*, pp. 197–205, Austin, TX, USA, November 2015.

[14] W. Jiang and Z. Yin, "Human activity recognition using wearable sensors by deep convolutional neural networks," in *Proceedings Of the 2015 ACM Multimedia Conference MM 2015*, pp. 1307–1310, Brisbane, Australia, October 2015.

[15] Y. Chen and Y. Xue, "A deep learning approach to human activity recognition based on single accelerometer," in *Proceedings of the 2015 IEEE International Conference On Systems, Man, and Cybernetics, SMC 2015*, pp. 1488–1492, Hong Kong, China, October 2016.

[16] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Systems with Applications*, vol. 59, pp. 235–244, 2016.

[17] A. Murad and J. Y. Pyun, "Deep recurrent neural networks for human activity recognition," *Sensors (Switzerland)*, vol. 17, p. 11, 2017.

[18] J. Zhou, J. Sun, P. Cong et al., "Security-critical energy-aware task scheduling for heterogeneous real-time MPSoCs in IoT," *IEEE Transactions On Services Computing (TSC)*, vol. 12, p. 99, 2019.

[19] Y. Guan and T. Plötz, "Ensembles of deep LSTM learners for activity recognition using wearables," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 2, pp. 1–28, 2017.

[20] L. Qi, X. Zhang, W. Dou, C. Hu, C. Yang, and J. Chen, "A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment," *Future Generation Computer Systems*, vol. 88, pp. 636–643, 2018.

[21] A. Ignatov, "Real-time human activity recognition from accelerometer data using Convolutional Neural Networks," *Applied Soft Computing*, vol. 62, pp. 915–922, 2018.

[22] H. F. Nweke, Y. W. Teh, M. A. Al-garadi, and U. R. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: state of the art and research challenges," *Expert Systems with Applications*, vol. 105, pp. 233–261, 2018.

[23] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: a survey," *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.

[24] S. Wu, G. Li, L. Deng et al., "$L1$-norm batch normalization for efficient training of deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 7, pp. 2043–2051, 2019.

[25] B. Almaslukh, J. Al Muhtadi, and A. M. Artoli, "A robust convolutional neural network for online smartphone-based human activity recognition," *Journal of Intelligent & Fuzzy Systems*, vol. 35, no. 2, pp. 1609–1620, 2018.

[26] R. Yao, G. Lin, Q. Shi, and D. C. Ranasinghe, "Efficient dense labelling of human activity sequences from wearables using fully convolutional networks," *Pattern Recognition*, vol. 78, pp. 252–266, 2018.

[27] T. Kautz, B. H. Groh, J. Hannink, U. Jensen, H. Strubberg, and B. M. Eskofier, "Activity recognition in beach volleyball using a deep convolutional neural network," *Data Mining and Knowledge Discovery*, vol. 31, no. 6, pp. 1678–1705, 2017.

[28] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd international Conference on machine learning, ICML 2015*, vol. 3, pp. 2332–2340, Lille, France, July 2015.

[29] S. Li, S. Zhao, P. Yang, P. Andriotis, L. Xu, and Q. Sun, "Distributed consensus algorithm for events detection in cyber-physical systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2299–2308, 2019.

[30] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, *Improving Neural Networks by Preventing Co-adaptation of Feature Detectors*, arXiv preparation, Geneva, Switzerland, 2012.

[31] B. M. h. Abidine, L. Fergani, B. Fergani, and M. Oussalah, "The joint use of sequence features combination and modified weighted SVM for improving daily activity recognition," *Pattern Analysis and Applications*, vol. 21, no. 1, pp. 119–138, 2018.

[32] G. M. Weiss, J. W. Lockhart, T. T. Pulickal et al., "A smartphone-based activity recognition system for improving health and well-being," in *Proceedings of the 3rd IEEE International Conference On Data Science And Advanced Analytics, DSAA 2016*, pp. 682–688, Montreal, QC, Canada, October 2016.

[33] L. Wang and R. Liu, "Human activity recognition based on wearable sensor using hierarchical deep LSTM networks," *Circuits, Systems, and Signal Processing*, vol. 39, no. 2, pp. 837–856, 2019.

[34] W. Lu, F. Fan, J. Chu, P. Jing, and S. Yuting, "Wearable computing for internet of things: a discriminant approach for human activity recognition," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2749–2759, 2019.

[35] X. Kuang, *Human Behavior Recognition Based on Deep Learning and Wearable Sensor*, Nanjing University of Information Engineering, Nanjing, China, 2018.

[36] M. M. Hassan, M. Z. Uddin, A. Mohamed, and A. Almogren, "A robust human activity recognition system using smartphone sensors and deep learning," *Future Generation Computer Systems*, vol. 81, pp. 307–313, 2018.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 6371814, 13 pages
https://doi.org/10.1155/2020/6371814

Research Article

Warehouse-Oriented Optimal Path Planning for Autonomous Mobile Fire-Fighting Robots

Yong-tao Liu,[1,2] Rui-zhi Sun,[1,3] Tian-yi Zhang,[4,5] Xiang-nan Zhang,[1] Li Li,[1] and Guo-qing Shi[1]

[1]*College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China*
[2]*North China Institute of Science and Technology, East Yanjiao, Beijing 065201, China*
[3]*Scientific Research Base for Integrated Technologies of Precision Agriculture (Animal Husbandry), The Ministry of Agriculture, Beijing 100083, China*
[4]*Graduate School of Advanced Integration Science, Chiba University, Chiba 263-8522, Japan*
[5]*Wedo Electric Solutions Technology Co. Ltd., Beijing 100095, China*

Correspondence should be addressed to Rui-zhi Sun; sunruizhi@cau.edu.cn

Received 17 December 2019; Revised 30 January 2020; Accepted 7 February 2020; Published 20 June 2020

Guest Editor: Xiaolong Xu

In order to achieve the fastest fire-fighting purpose, warehouse autonomous mobile fire-fighting robots need to make an overall optimal planning based on the principle of the shortest time for their traveling path. A∗ algorithm is considered as a very ideal shortest path planning algorithm, but the shortest path is not necessarily the optimal path for robots. Furthermore, the conventional A∗ algorithm is affected by the search neighborhood restriction and the theoretical characteristics, so there are many problems, which are closing to obstacles, more inflection points, more redundant points, larger total turning angle, etc. Therefore, A∗ algorithm is improved in eight ways, and the inflection point prior strategy is adopted to compromise Floyd algorithm and A∗ algorithm in this paper. According to the criterion of the inflection point in this paper, the path inflection point arrays are constructed and traveling all path nodes are replaced by traveling path inflection points for the conventional Floyd algorithm backtracking, so it greatly reduces the backtracking time of the smooth path. In addition, this paper adopts the method of the extended grid map obstacle space in path planning safety distance. According to the relationship between the actual scale of the warehouse grid map and the size of the robot body, the different safe distance between the planning path and the obstacles is obtained, so that the algorithm can be applied to the safe path planning of the different size robots in any map environments. Finally, compared with the conventional A∗ algorithm, the improved algorithm reduces by 7.846% for the path length, reduces by 71.429% for the number of the cumulative turns, and reduces by 75% for the cumulative turning angle through the experiment. The proposed method can ensure robots to move fast on the planning path and ultimately achieve the goal of reducing the number of inflection points, reducing the cumulative turning angle, and reducing the path planning time.

1. Introduction

One of the core technologies of autonomous mobile robots is the ability of real time path planning. Path planning refers to, under the premise of following an optimal index (such as the shortest time, the optimal path, and the lowest energy consumption), the optimal path without collision from origin to termination is planned in the usage scenario [1], and the second path planning can be planned in the process of the original path planning in real time.

The premise of path planning needs to build environment map. There are generally two ways to build a map: one is that the robot automatically builds the map through SLAM; the other one is that the site CAD is manually imported into the robot system according to the specified format. Either way, the environmental layout is required to be as fixed as possible.

The object of this paper, warehouse-oriented autonomous mobile fire-fighting robot, is just suitable for this scenario. In the environment, multiple flame detection

probes with address codes are installed on the top. When a flame is detected by a certain probe, the probe code and alarm signal are sent to the robot through edge computing processing and wireless networking technology [2–7]. Here, we need to ensure the security and stability of wireless sensor network and reduce the network delay [8–11]. The robot completes autonomous path planning and reaches the ignition point in the shortest time, and it extinguishes fires early in the fire. The structural space in the scene is relatively stable. In order to adapt to the path planning of the fire-fighting robot, we should take the best rather than the shortest as the principle according to the optimization index path. Because the robot is limited by its own space structure, it is difficult to ensure high-speed travel when it is in a small space and the loss time is much longer than that of path planning. The optimal path is based on the principle of the shortest time, i.e., the robot can achieve the fastest traveling path, and the fire-fighting robot can reach the fire point in the shortest time to put out the fire. This requires that, under the premise of ensuring the shortest planning path, the path and obstacles (such as shelves) have enough safe distance, reserved robot turning action space, etc. And, the number of turning points is less; the total turning angle is lower.

After years of development, the path planning algorithm has achieved good results, but there are still some problems in specific application scenarios. The proposed method of the artificial potential field by Khatib [12] in 1994 falls into local minimum and oscillate near obstacles easily. In genetic algorithm (GA) [13], the range of coding length is large and the convergence is bad. In the complex map mode, the convergence speed of the algorithm is slow and the computation is large. In artificial neural network (ANN) [14], when the initial feedback information of the algorithm is insufficient, it needs to adjust the weight in a long time to achieve the optimization effect of the project approval. The search speed of the ant colony algorithm is slow and falls into local optimality easily [15]. The Dijkstra algorithm is a classical breadth prior algorithm [16], and it can always find the optimal path and conduct the overall search directly. However, it does not consider the target point information, and it has a long search time and low search efficiency because it cannot meet the need of rapid path planning. BFS optimal prior algorithm can quickly guide the search to the target node and greatly improve the search efficiency but cannot often get the shortest path.

A∗ algorithm, first proposed in 1968 by Hart et al. [17], is a heuristic search algorithm [18–20] combined with the advantages of the above algorithms. It uses heuristic information to guide the search direction, so as to reduce the search scope and improve the search efficiency. It is a typical heuristic path planning algorithm [21, 22], which has been successfully applied and verified in mobile robot path planning [23].

However, due to the computational characteristic of A∗ algorithm, the planned path points generally have many problems, such as many broken lines and large cumulative turning angle, and easily discard some points to generate suboptimal problems [24]. Therefore, many researchers have proposed improved algorithms in computational time [25, 26]. Gao et al. [27] proposed a bidirectional time-effective A∗ algorithm to find the path and adopted a multineighborhood grid distance computational scheme to achieve the effect of improving efficiency and smoothing the path, but it is not suitable for large-scale complex maps. In [28], an improved search A∗ algorithm combined with skip points is proposed, the speed of the algorithm has been greatly improved, but there are still many inflection points and close to obstacles. In [29], the size of the robot is considered, and a neighborhood matrix is proposed to improve the path safety of obstacle search. However, the path smoothing and length are not improved, so there are still many problems including more turning angles and large total turning angle. In [30, 31], proposed expand search neighborhood and dispose of quadratic smoothing of quintic polynomial reduce the length of search path and turning angle and greatly improve path smoothing. However, there are still some problems, such as closing to obstacles, and safety distance is not considered.

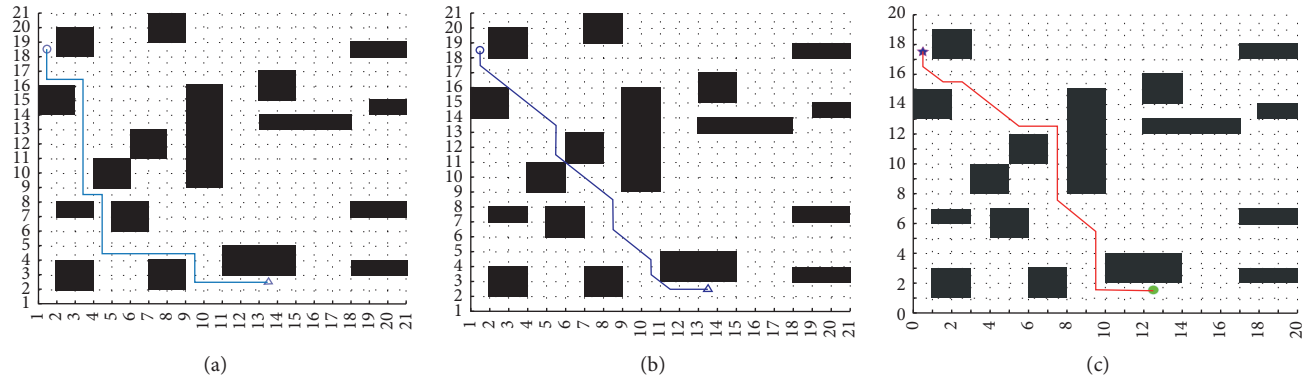
Aiming at the problems of usage scenarios of warehouse-oriented fire-fighting robots and the existing various optimizational A∗ algorithm, this paper proposes a comprehensive improved A∗ algorithm to achieve path smoothing and reduces broken lines and cumulative turning angle. And, according to the bulk of the robot and the scale of map, we set up safe distance with the obstacles to meet the path planning application of the robots.

The paper is organized as follows. In Section 2, we propose the improved A∗ algorithm and smoothen the optimal path. Then, we introduce the constraint conditions including robot's safety traveling distance and compare with the effect of the current latest algorithm. Section 3 introduces the practical application effect of the complete algorithm in this paper and verifies a great impact of safety distance on the path. Section 4 concludes the paper.

2. Improved A∗ Algorithm

The planning effect of the conventional four-way search A∗ algorithm is shown in Figure 1(a). There are a lot of right-angle points in the route, and the length of the route is not the shortest. On this basis, eight-way search A∗ algorithms are proposed; the planning effect is shown in Figure 1(b). It can be seen that the eight-way search A∗ algorithm overcomes the problem of right-angle inflection point in the path and plans the shortest path length under the algorithm. However, the algorithm has the problems of closing to the obstacles and crossing the diagonal vertex of the obstacles (Figure 1(a)); this path cannot be used as the traveling path of the robot obviously. Therefore, the conventional eight-way search A∗ algorithm is not suitable for robot path planning and it needs to improve the basic safety distance and the constraint.

2.1. Improved Eight-Way A∗ Algorithm. In view of the existing problems of conventional eight-way search A∗ algorithm, we make a basic improvement, i.e., when the eight-way search area is expanded and if the path forward direction is a slash, the constraint term is added, for example,



(a) (b) (c)

Figure 1: The comparisons of A∗ algorithm comparison. (a) Conventional four-way A∗ algorithm. (b) Conventional eight-way A∗ algorithm. (c) Improved eight-way A∗ algorithm.

in Figure 1(c), when the path is planned according to a lower right slash, then we judge whether there is any obstacle on the right and bottom sides of the parent-node. If there is, the path is not feasible and the subnode needs to be selected again. When the path is planned according to upper left slash, then we judge whether there is any obstacle on the left and upper sides of the parent-node. If there is, the path is not feasible, the subnode needs to be selected again. The planning path according to this method is shown in Figure 1(c).

The performance comparisons of the three algorithms in Figure 1 are shown in Table 1. It can be seen that the conventional four-way search A∗ algorithm has the longest path length and the slowest operation time. The conventional eight-way search A∗ algorithm has the shortest path length and the fastest operation time, but the path has the problems of closing to the obstacle vertex and crossing the obstacle vertex angle, so the path cannot be used by robots. On this basis the eight-way search, the A∗ algorithm is improved in this paper and the shortest path is realized under the algorithm. Compared with the conventional four-way search A∗ algorithm, the path length reduces by 12.53%, the operation time reduces by 46.9%, and the situation closing to the obstacle vertex and crossing the corner is effectively avoided.

2.2. Improved Floyd Path Smoothing Algorithm. In [30], traversing nodes are used for the smoothing process. In [32], Floyd algorithm is used for the smoothing process. These two methods need to trace back all path nodes one by one from the origin or termination to determine whether they intersect obstacles. If they do not intersect, the intermediate node will be discarded, and if they intersect, the previous node will be retained. These two methods can optimize the original path, but traversing all the path nodes significantly reduces the execution speed, and the straight path nodes in the original path can avoid traversing judgment completely. Therefore, this paper proposes the criterion of the prior inflection point, and Floyd algorithm is used to connect the backtracking method of the inflection point.

2.2.1. Criterion of the Planning Route Inflection Point. Before path smoothing is completed, the eight-way A∗ algorithm path planning has to be improved. Inflection criteria are added after planning; path inflection array is generated. For example, let point D in Figure 2 be the current node n, according to the A∗ algorithm formula:

$$f(n) = g(n) + h(n), \tag{1}$$

where $g(n)$ indicates the length of the actual path from the origin to the current node and $h(n)$ is the distance of estimate cost function from the state node n to termination.

According to the eight-way extended A∗ algorithm, there are only two path lengths of adjacent nodes, i.e., L and $\sqrt{2}L$. If the distance from the current node n to its parent-node $(n-1)$ and sub-node $(n+1)$ is not equal, then the node in the planning path is the inflection point. The criteria are as follows:

$$L(X_{n-1}, X_n) \neq L(X_n, X_{n+1}). \tag{2}$$

For example, if $L(C, D) \neq L(D, E)$, then the planning path node D is the inflection point; if $L(D, E) = L(E, F)$, then the planning path node E is a normal node. In the judgment, all the noninflection nodes in the planning path are expressed as (x, y, 0); the inflection nodes are expressed as (x, y, 1); the origin of the robot path is defined as a common node; the termination is defined as an inflection. All the nodes are saved in the file-list array. In the last path backtracking, simply priorly connecting the inflection points can get the path smooth optimization.

2.2.2. Design of Floyd Algorithm. We combine Floyd algorithm and the improved A∗ algorithm; the length of the original planning path can be shorter; the number of inflection points can be less; the cumulative turning angle can be less; the path can be smoother.

In the smooth process, firstly, we need to get the file list of the inflection points of the planning path and directly connect the second inflection point in the backtracking path when backtracking from the termination. If it does not intersect the obstacle, then the third inflection point should be connected. If the connection encounters the obstacle, all nodes between the third inflection point and the second inflection point are traced back in turn until the node that

Table 1: The comparisons of performance efficiency for A∗ algorithm.

Origin termination	Algorithm	The length of the path	Turn times
(18, 1) (2, 13)	Four-way A∗	28.00	0.0678
	Eight-way A∗	22.14	0.0196
	Improved eight-way A∗	24.49	0.036

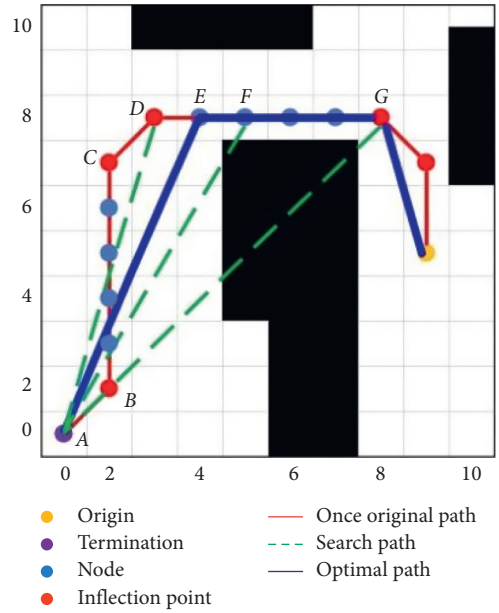


Figure 2: The principle of improved A∗ algorithm.

does not intersect the obstacle is found, and the intermediate node is discarded.

In Figure 2, if points A and B are adjacent inflection points, then they can be connected directly. The distance is written as $L(A, B)$. Judge $A \longrightarrow C$ and $A \longrightarrow D$ in turn; if they do not intersect obstacles when they are connecting, the original path can be changed to $A \longrightarrow D$. But, when $A \longrightarrow G$, they intersect obstacles, so the path length $L(A, G) = +\infty$:

$$L(A, C) < L(A, B) + L(B, C). \tag{3}$$

Then, the array of points A and C are retained, and the array of point B is deleted:

$$L(A, D) < L(A, C) + L(C, D). \tag{4}$$

Then, the array of points A and D are retained, and the array of point C is deleted.

Because $L(A, G) = +\infty$,

$$L(A, G) = L(A, D) + L(D, G). \tag{5}$$

At this time, assume the coordinate of point G is $G(i, j, 1)$, then the original path node between D and G will be traced back from point G, $G_1(i-1, j, 0)$, $G_2(i-2, j, 0)$, $G_3(i-3, j, 0)$, and $G_4(i-4, j, 0)$, where $G_3 = F$ and $G_4 = E$. When connected with point A, $L(A, G_1) = +\infty$, $L(A, G_2) = +\infty$, and $L(A, F) = +\infty$ are abandoned:

$$L(A, E) + L(E, G) < L(A, D) + L(D, G). \tag{6}$$

Then, the array of points A and E are retained, and the array of point D is deleted. The original path $A \longrightarrow B \longrightarrow C \longrightarrow D \longrightarrow$ G is optimized as $A \longrightarrow E \longrightarrow G$, so far, this section of path optimization is completed, and the next path optimization is carried out accordingly.

The specific implementation steps of the path smoothing algorithm are as follows:

(i) Step 1: take out the processed nodes of the above improved eight-way A∗ algorithm and carry out the secondary backtracking analysis from the termination.

(ii) Step 2: start the termination to connect all the inflection points one by one, $B \longrightarrow C \longrightarrow D$, and then the nodes of noninflection points, such as the points between $B \longrightarrow C$, are ignored during the connection process.

(iii) Step 3: when points A and D are directly connected, continue to search backward inflection point G. At this time, there will be a conflict between search path $A \longrightarrow G$ and obstacles. Then, search the nodes of noninflection point between $G \longrightarrow D$.

(iv) Step 4: if there is still a conflict with the obstacle when searching to node F, then continue to Step 3

until node E, that is, no conflict with the obstacle is found.

(v) Step 5: at this time, take the direct connection path $A \longrightarrow E$ as the fixed path and repeat step $2 \longrightarrow 3 \longrightarrow 4$ from node E until reaching the origin.

The flowchart of the algorithm is given in Figure 3.

Compared with [33], the method only makes redundant judgments for inflection points; it adds the backtracking function of common nodes between obstacle inflection nodes and makes the path distance shorter. Compared with [30, 32], the traversal connection judgment time of the smoothing path algorithm is significantly reduced. In the method, all node connection judgments are no longer performed, and only the inflection nodes in the eight-way A∗ algorithm planning nodes are preferentially connected twice. There is a conflict with obstacles, and then node by node from the inflection node back is connected; the number of connection judgments is reduced greatly; the path smoothing processing time becomes shorter.

An array of once planned path node *int a* [3] [6] is shown as

$$\left\{\begin{array}{l} 0,0,0, \\ 2,2,1, \\ 2,3,0, \\ 2,4,0, \\ 2,5,0, \\ 2,6,0, \\ 2,7,1, \\ 3,8,1, \\ 4,8,0, \\ 5,8,0, \\ 6,8,0, \\ 7,8,0, \\ 8,8,1, \\ 9,7,1, \\ 9,6,0, \\ 9,5,1 \end{array}\right\}. \tag{7}$$

It can be seen that there are five inflection nodes besides origin [0, 0, 0] and termination [1, 5, 9] in once planned path, and the following five inflection nodes are used as the prior traversal point:

$$\left\{\begin{array}{l} 2,7,1, \\ 3,8,1, \\ 8,8,1, \\ 9,7,1, \\ 9,5,1 \end{array}\right\}. \tag{8}$$

After executing the optimized algorithm based on Floyd algorithm, the final path contains only 3 nodes and the noninflection points in the once path. The optimal path node array *int a* [3] [3] is shown as follows:

$$\left\{\begin{array}{l} 4,8,0, \\ 8,8,1, \\ 9,5,1. \end{array}\right\}. \tag{9}$$

2.2.3. Simulation Analysis of the Confluent Improved Algorithm. To verify the effect of the proposed confluent improved Floyd and improved A∗ algorithms in this paper, comparisons are made between the basic improved A∗ algorithm and the proposed prior moving route algorithm in [34]. The optimized path in [34] is shown in the red path in Figure 4(a). The length of the path is actually 21.4852 grids, but the path has the problem of closing to the apexes of the obstacles and the possibility of crossing the top angle of the obstacles, so the path is not suitable as a robot planning path.

This paper basically improves the path planning of the A∗ algorithm and adds grid constraints to avoid closing to obstacle paths. The planned path is shown in Figure 4(b).

Finally, path smoothing process is added based on the path planning of the improved A∗ algorithm, and the processing effect is shown in the blue path in Figure 4(c).

In the case that the initial orientation of the robot is not considered in the three paths in Figure 4, the obtained data graph by the experimental simulation is shown in Table 2. Compared with the improved eight-way A∗ algorithm, in this paper, the path length of the confluent algorithm reduces by 7.846%, the number of cumulative turns reduces by 71.429%, and the cumulative turning angle reduces by 75%. Compared with the improved A∗ algorithm in [34], in this paper, the path length of the smoothing algorithm reduces by 0.02%; the length optimization is not obvious. But the number of the cumulative turns reduces by 66.67%, and the cumulative turning angle reduces by 57.91%. The advantage of the part is obvious, the path is more conducive to the execution of the robot.

So far, although the proposed confluent algorithm in this paper has obvious advantages in cumulative turning points and cumulative turning angles, there are still cases crossing the vertices of obstacles, so it needs to further optimize the path safety distance considering the actual situation of the robot body space.

2.3. Optimization Algorithm of the Path Planning Safety Distance. For the reasonable traveling path of the robot, we should fully consider its own space structure and a sufficient safe distance from the obstacles should be reserved in the path planning. In this way, the robots can reduce the detection and processing time required to avoid obstacles when it actually follows the path in the later stage. Although this paper achieves the goal of the smoothing path by improving the Floyd algorithm, the optimized path still has the situation closing to the obstacles. Therefore, this paper continues to lead into the concept of path safety space in the design for algorithm optimization.

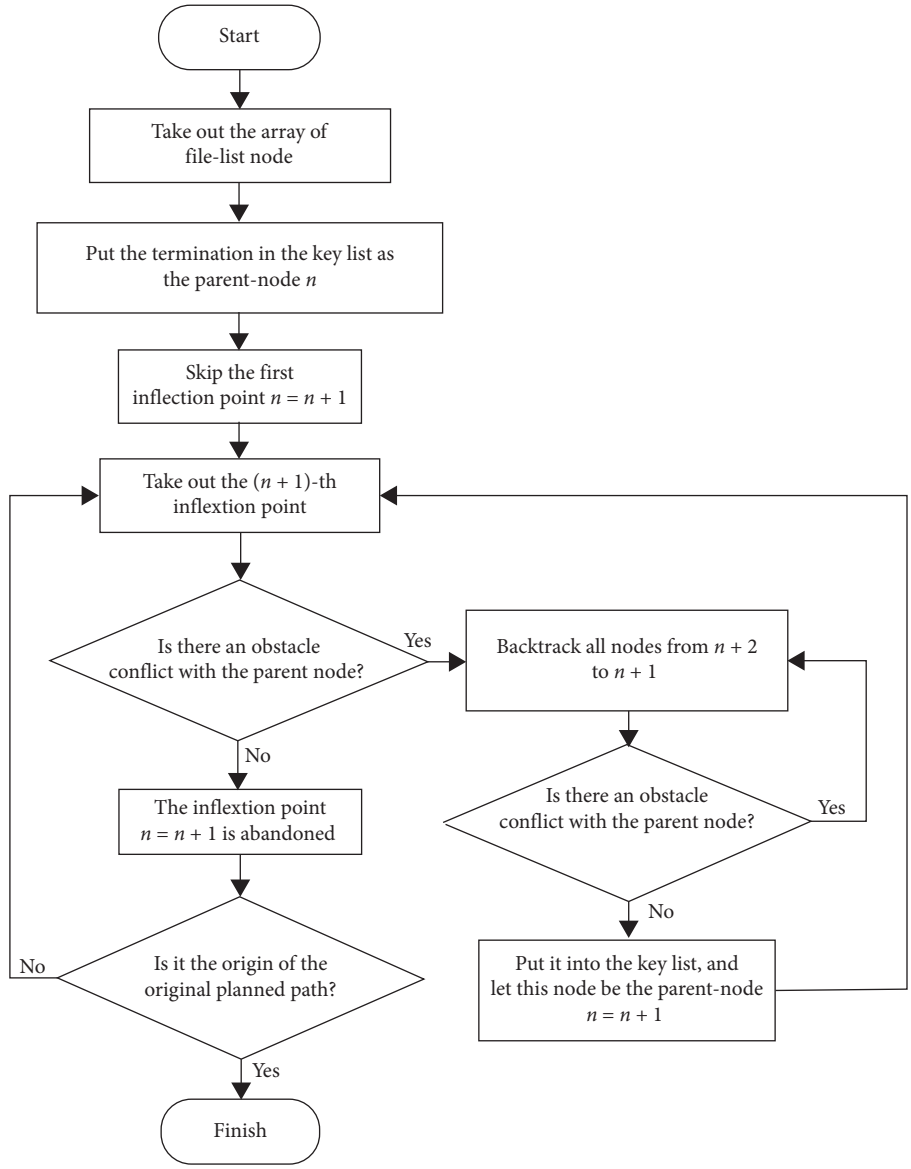


Figure 3: Algorithm flowchart.

2.3.1. Optimization Algorithm of the Safety Distance Based on Extended Obstacle Boundary. In order to ensure the space safety distance of the path, there are generally two methods. One is to expand the pixels occupied by the robot body within the map pixels and make the robot body space consistent with the actual map pixel ratio. The cross-obstacle search algorithm and the *m*-shaped obstacle search algorithm are given in [29]. The algorithm obtains different safety distances by changing the size of the search matrix to ensure the safety of the different robots in different map environments. The other one is the method that is to extend the obstacle boundary. This method combining with the size of the robot can set the safety distance arbitrarily and intuitively without increasing the difficulty of the algorithm, and it can effectively reduce the number of traversal grids in the search area and the search time.

Note: in the following path planning diagrams, the red path is the planned path of the basic improved eight-way A∗ algorithm; the blue path is the smoothed planned path, which is the final path; the colored grid is all the traversal grids in the path planning process.

The planned route is the planned route obtained in [29] using the 12-neighborhood search algorithm in Figure 5(a). The method can improve the search efficiency, and it also leads to the increase of the path length. The algorithm is designed with the path length as the primary priority and the search time as the secondary priority. In Figure 5(b), the red path is the improved eight-way A∗ algorithm planned path in this paper. The path is linked by the grid center point. The planned path can also achieve a safety distance to a certain degree. However, it is not flexible enough and cannot be set arbitrarily according to the robot map ratio.

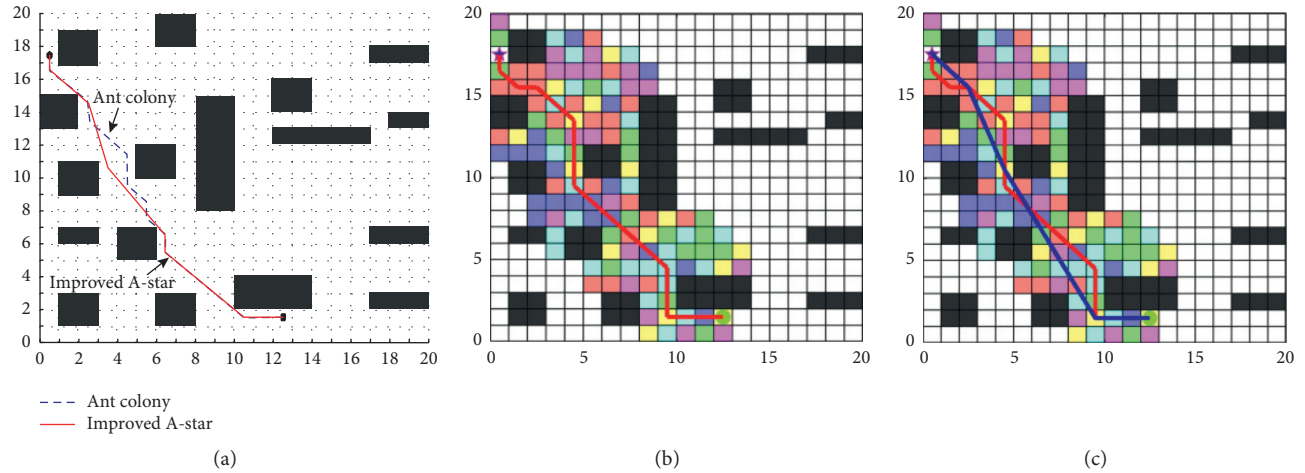


Figure 4: The improved path smoothing algorithm process. (a) The improved algorithm in [34]. (b) The improved algorithm in this paper. (c) The confluent path smoothing algorithm.

Table 2: Comparisons of key parameters for the algorithmic effectiveness.

Algorithm	The length of path	Inflection points	Cumulative turning point (°)
Improved eight-way A∗	23.31	7	360.00
Algorithms in [25]	21.4852	6	213.84
Smoothing algorithm in this paper	21.4809	2	90.00

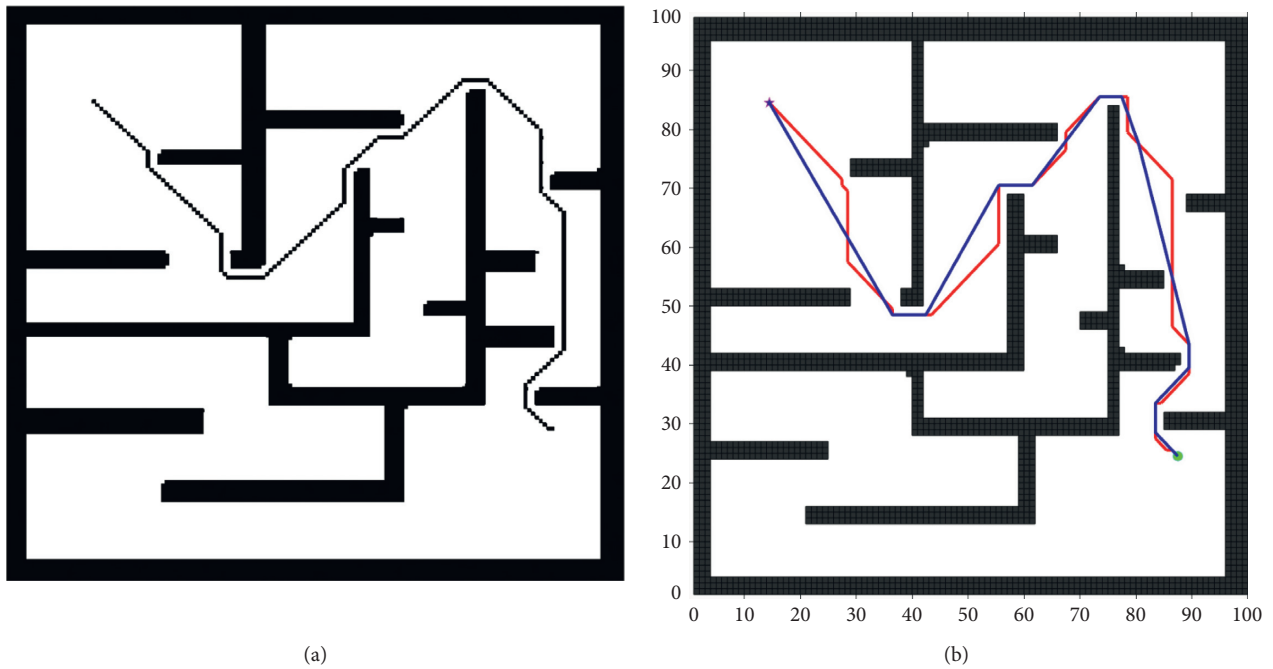


Figure 5: The comparison of the safety distance solutions. (a) Cross barrier search in [29]. (b) Extended obstacle boundary method in this paper.

It can be seen in Table 3 that the improved A∗ smoothing algorithm adopts the search method of expanding obstacle boundary in this paper; compared with [29], the path length is shorter and the number of traversed grids is smaller. The number of the path inflection points that is a key indicator for evaluating the robot's traveling path has also decreased 44.4%; the corresponding cumulative turning angle has decreased significantly; the path smoothness advantage is very obvious.

Table 3: The comparisons of two safety distance search methods.

Algorithm	The length of path	Inflection points	Ergodic grid number
12 neighborhood cross search algorithms in [29]	224	18	7957
Extended barrier boundary method	169.81	10	3739

2.3.2. Impact Analysis of the Path Safety Distance. Although the constraint condition of safety distance is required in path planning, the design path is more suitable for robot walking. On the surface, the path designed by the algorithm should be as far away from the obstacles as possible, but when this safety distance is set too large, the path cannot be planned. As shown in Figure 6, this paper has tested and analyzed the situations of the different safety distances.

In Figure 6, the path planning test was performed on a map of 25×25 grid pixels for each without safety distance constraint, 0.6 grid safety distance, and 1.0 grid safety distance.

According to the analysis in Table 4, it can be found that when there is no safety distance, the planned path is the shortest; the number of inflection points is centered; the number of traversed grids is the largest; and the planning time is the longest. At 0.6 grid safety distance, the path length is 1.2% longer than the shortest path; the number of inflection points reduces by 50%; the number of traversal grids reduces by 62.6%; and the planned time reduces by 37.79%. 1.0 grid safety distance has the longest path length, the largest number of inflection points, and the shortest planned time. If the safety distance grid is set to 2.0, path planning cannot be completed.

Under the condition that the robot path planning guarantees a safe distance, the time spent in path planning can be ignored compared with the time saved by robot turning and fast passing. Therefore, it can be ensured that the second shortest path with the least number of inflection points and the smallest cumulative turning angle of the robot is the optimal path of the robot.

3. Autonomous Mobile Robot Warehouse Center Path Planning

After the algorithm design is completed, it needs to be verified in the actual map environment. Because the robot application environment is a structural layout space, the verification uses the warehouse center as the verification object, and the warehouse map is derived from Baidu Gallery. Robot path planning must first establish an accurate two-dimensional map of the environment space and then perform path planning.

3.1. Construction of Environmental Map for Warehouse Center. The map construction in the normal A$*$ algorithm uses the method that directly creates a two-dimensional array in the program, as shown in Figure 7(a), and the produced grid map is shown in Figure 7(b). "0" in the array indicates that the open grid is in white, and "1" indicates that the obstacle is in black. Each "0" or "1" represents a grid, and the number of rows and columns of the two-dimensional array corresponds to the number of rows and columns of the grid map.

This representation method is flexible to change the map and is suitable for low-resolution map construction. For large-space and high-resolution maps, the two-dimensional array is huge and it is difficult to accurately match the actual space. For institutional spaces such as warehouse centers, accurate CAD drawings are generally available during construction, as shown in Figure 8 (The Figure 8 from the Internet, the website is: http://www.51w2c.com/details_id_1347.html). The actual positions and absolute coordinates of various items are accurately marked on the drawings, and the positions will not basically change easily. Low-resolution grid maps in this scenario will cause large errors and cannot accurately reflect the precise coordinate positions of obstacles and feasible paths, and it is easy to make the planned path deviates from the actual path, causing the robot to excessively rely on obstacle avoidance function when it is traveling.

Aiming at the above problems, in order to improve the practicability of the system, this paper designs a drawing program by using Matlab's composition conversion function. When the software is imported, we must first draw the warehouse center (Figure 8) into one of the three formats bmp, jpg, and png according to the size proportion and position, as shown in Figure 9. Select the imported image in the GUI interface designed by Matlab, set the required horizontal axis resolution of the grid image, and complete the conversion according to the horizontal and vertical proportions of the original image.

According to this step, the actual warehouse center shown in Figure 8 is converted into the binary map of Figure 9 for the application scenario and imported into Matlab R2017a for experimental verification.

In the scenario, the actual space of the CAD drawing of the warehouse center is 4476×4000 cm. The two-dimensional space of the robot is 60×70 cm. The two-dimensional size of a single shelf is 60×200 cm. The road width between the shelves is 150 cm.

The higher the resolution during the construction of the grid map, the smaller the error will be, but the time consumption with the system path planning will increase greatly. Therefore, there are three preliminary methods for setting the resolution:

(1) Take the two-dimensional space of the robot as the grid point size. The robot adopted a two-wheel differential turning method, and the width of the forward direction needs to be 60 cm. Taking this as the minimum resolution of the grid map, the resolution of the warehouse center grid map is 43×48.

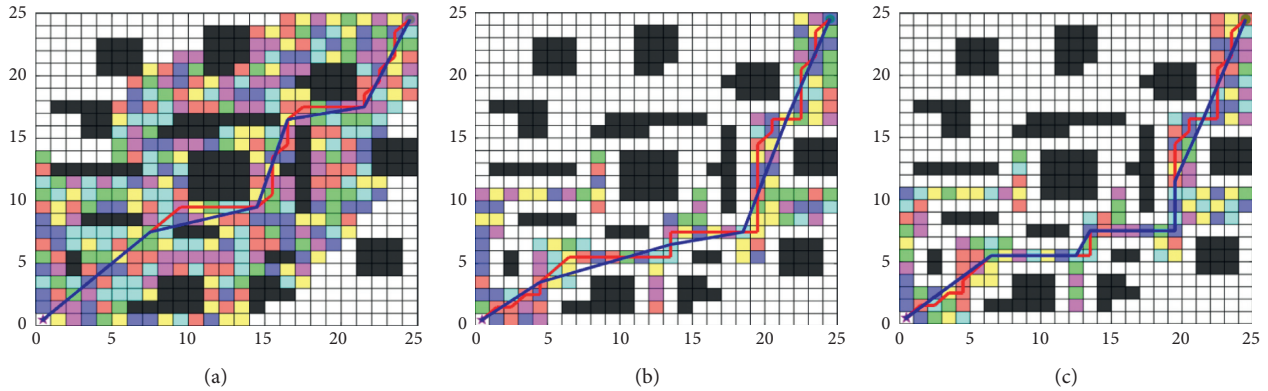


Figure 6: The effect of path planning at different safety distances. (a) No safety distance. (b) Safety distance 0.6 grid. (c) Safety distance 1 grid.

Table 4: Effect analysis of path planning for different safe distances.

Algorithm	The length of path	Discount points	Ergodic grid number	Planning time (S)
No safe distance	37.17	4	361	0.4424
0.6 grid safe distance	37.62	2	135	0.2749
1.0 grid safe distance	39.98	5	135	0.2272

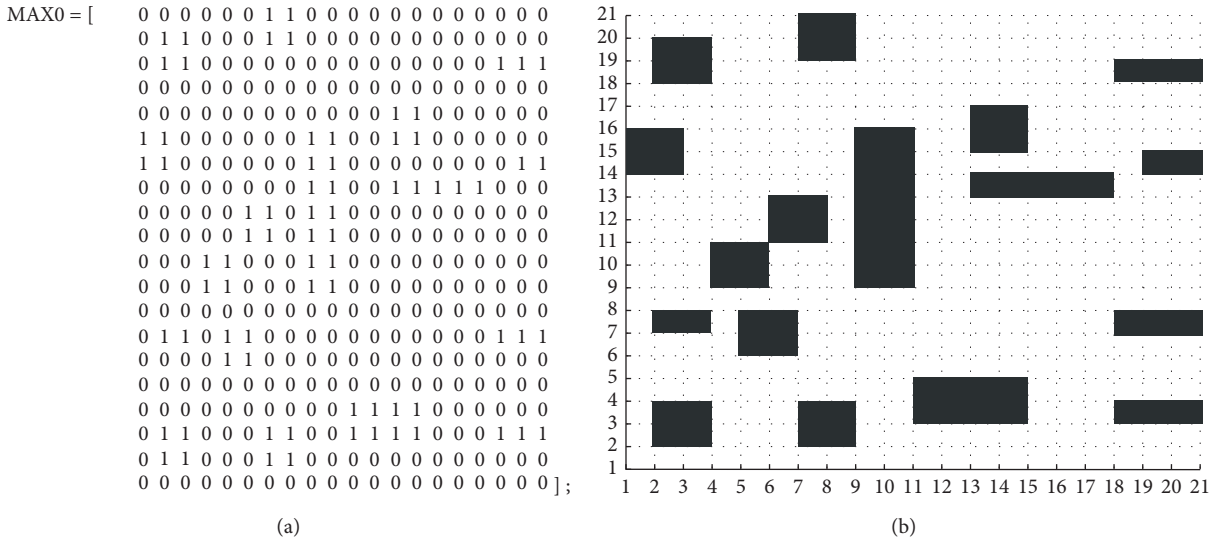


Figure 7: The grid map creation. (a) Map array. (b) Corresponding grid map.

(2) Use shelf width as a reference and consider most path widths. It can be seen from the CAD drawing that the width of most of the auxiliary passages between the shelves is 150 cm, so the grid size of the storage center is considered to be 30 cm, and the resolution of the warehouse center grid map is 132×149.

(3) According to the CAD map, the grid is drawn according to a certain proportion and the grid size is determined by the path planning time.

The Matlab image conversion program design process is shown in Figure 10.

3.2. Path Planning Verification

3.2.1. Impact of Different Resolutions on Planning Time. Path planning experiments were performed on the grid maps designed to be constructed at multiple resolutions. The obtained data are shown in Table 5.

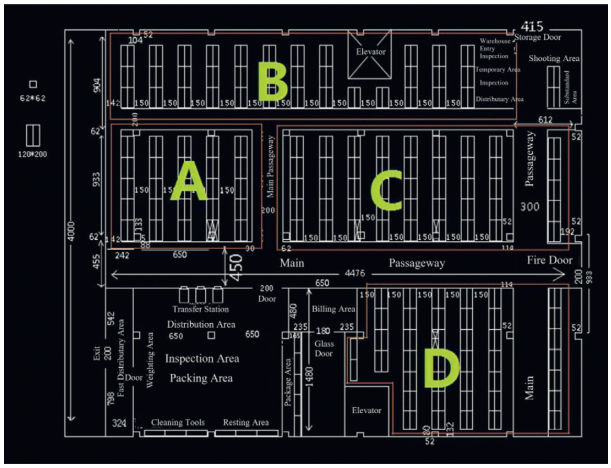


Figure 8: Actual layout of a warehouse center.

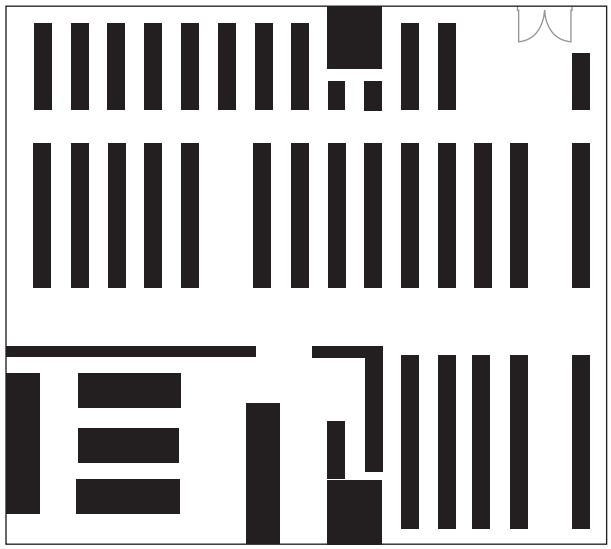


Figure 9: Binary map for warehouse center CAD.

It can be seen that the larger the number of grids on the same map, the higher the resolution, the smaller the spatial error of the map expression, but the longer the path planning time. When the number of grids is 132, the grid map can basically and accurately express the spatial layout of shelves and aisles in the warehouse center, and the planning time can be completed within 0.5S.

3.2.2. Impact of Different Safety Distances on Planned Paths. At the same grid resolution, different safety distance settings will have a greater impact on path planning.

The warehouse maps are set at different security distances and at the same 132 grid resolution, the obtained path planning results are shown in Figures 11(a) and 11(b), and the specific experimental data are shown in Table 6.

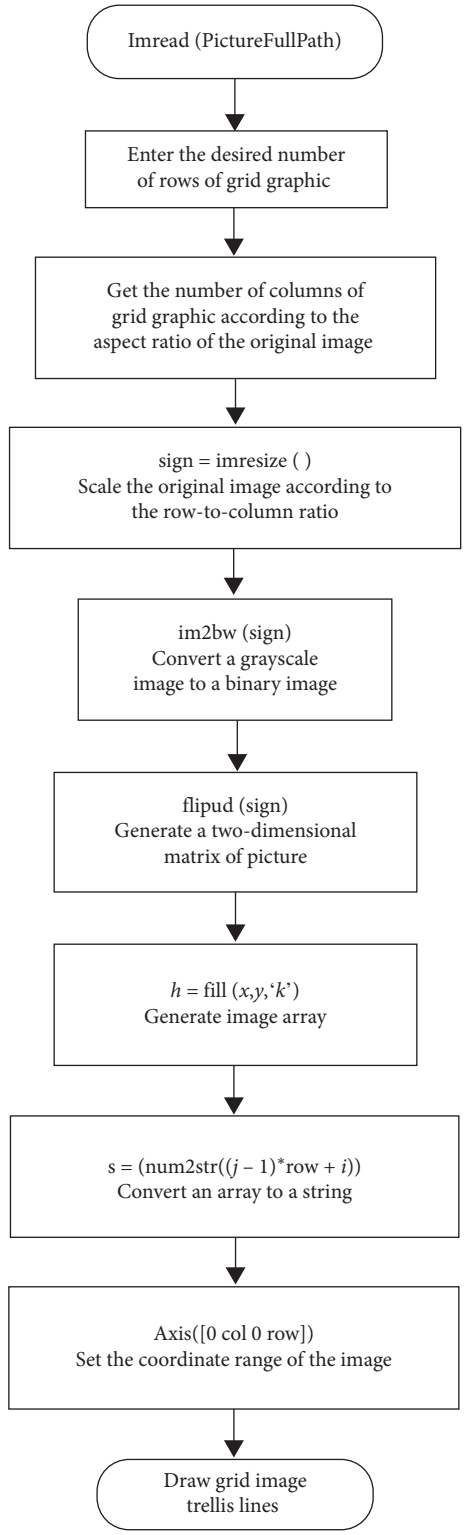


Figure 10: Grid map creation process.

Figure 11(a) shows the path planning without the safety distance. There are many cases where the path is close to the obstacles. The path planning passed the narrow section. The

Table 5: Path planning time at different grid resolutions.

Safe distance	Planning time/(S)	The length of path	Discount points	Ergodic grid number
Body width	0.4976	194.445	7	7807
15 cm	0.2711	205.5228	4	6034

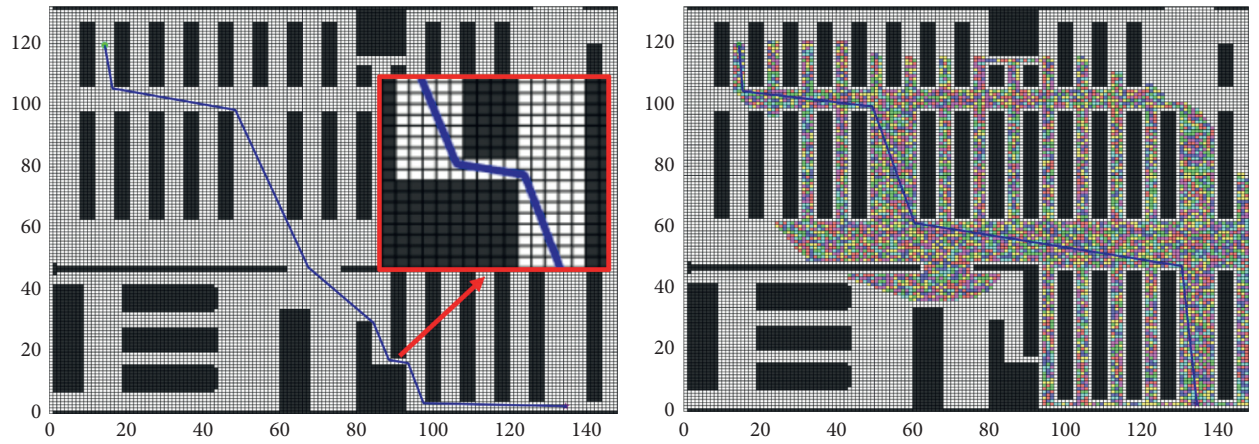


Figure 11: 132 grid maps with different safety distance paths. (a) No safety space path. (b) 10 cm safety space path.

Table 6: Effects of different safety distances on grid path planning.

Algorithm	Grid size (cm)	Safe distance	Planning time (S)
66	60.0	Body width	0.0975
132	30.0	10 cm	0.3421
200	20.0	10 cm	0.6429
400	10.0	10 cm	8.1772
800	5.0	10 cm	103.3154

road width is only 2 grids, and the actual width is 60 cm; it is shown in the enlarged path of Figure 11(a); it is exactly the same as the width of the robot body. Obviously, the section of the robot is very difficult to pass or even unable to pass. The path planning has 7 inflection points, and the robot needs to perform steering actions multiple times. Both of these problems greatly affect the traveling time of robot.

In Figure 11(b), a grid closing to the obstacle is placed in the closed-list by expanding the obstacle area, and it is set to prohibit traversal. The method reserves a 15 cm safety distance between the robot path and the obstacles. Due to the introduction of the safety distance, the planned path has changed greatly compared with that in Figure 11(a). Because it is difficult for the robot to travel at high speeds in narrow sections, increasing the safety distance ensures that the robot travels on the secondary shortest path, but it ensures that the robot can travel at full speed and reduces turns. Compared with the path without safety distance, the number of turns of the path reduces by 42.8% and the planning time reduces by 44.91%.

It can be seen that the correct and reasonable safety distance setting not only determines whether the algorithm can plan the correct path, but also can greatly improves the path optimization. Therefore, the path planning safe distance must be considered in practical applications.

4. Conclusions

The shortest path cannot be as the judgment that the path is optimal for warehouse autonomous mobile fire-fighting robots in indoor structure space. It should also be based on the safety distance between the path and the obstacle to make the robot move at high speed, and the executive time is the shortest. Therefore, this paper improves the problems of the conventional A∗ algorithm in the path planning of autonomous mobile robots.

In the design, Floyd algorithm and A∗ algorithm are fused by the inflection point priority strategy. Experiments show that the method can reduce the path optimization time and significantly reduces the total number of the path of the inflection points and the cumulative turning angle and thus shortens the path length and increases the smoothness of the path. Finally, the problem of safe path of all kinds of robots in different space is solved by expanded obstacles; the time of the path planning is reduced greatly; the path is optimized in many aspects; the planning efficiency and the algorithm practicability are improved.

Although the method of the path planning has achieved good experimental results, there is still a long planning time in the process of high-resolution map path planning. In the next research, we will consider how to add new search heuristic functions and constraints, so as to improve the search efficiency of high-resolution grid map in complex environment and ensure the algorithm has better applicability.

In the future, the fire-fighting ability of a single robot will not meet the demand for a large indoor space or a flammable environment. Therefore, we will also consider the cooperation and game of multirobots in this environment [35] and study the wireless mobile networking and anticollision theory of multirobots [36].

Data Availability

The MATLAB path planning data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the Fundamental Research Funds for the Central Universities (3142018047).

References

[1] E. Frazzoli, A. Munther, and M. A. Dahleh, "Real-time motion planning for agile autonomous vehicles," in *Proceedings of the 2001 American Control Conference*, vol. 25, no. 1, pp. 43–49, IEEE, Arlington, VA, USA, June 2001.

[2] X. Xu, Y. Li, T. Huang et al., "An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks," *Journal of Network and Computer Applications*, vol. 133, pp. 75–85, 2019.

[3] H. Wu, W. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.

[4] L. Qi, Y. Chen, Y. Yuan, S. Fu, X. Zhang, and X. Xu, "A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web*, vol. 23, no. 2, pp. 1275–1297, 2019.

[5] X. Xu, Y. Xue, L. Qi et al., "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, pp. 89–100, 2019.

[6] H. Wu, "Performance modeling of delayed offloading in mobile wireless environments with failures," *IEEE Communications Letters*, vol. 22, no. 11, pp. 2334–2337, 2018.

[7] J.. Zhao, X.. Guan, and X.. Li, "Power allocation based on genetic simulated annealing algorithm in cognitive radio networks," *Chinese Journal of Electronics*, vol. 22, no. 1, pp. 177–180, 2013.

[8] X. Xu, S. Fu, L. Qi et al., "An IoT-Oriented data placement method with privacy preservation in cloud environment," *Journal of Network and Computer Applications*, vol. 124, pp. 148–157, 2018.

[9] H. Wu, Z. Han, K. Wolter, Y. Zhao, and H. Ko, "Deep learning driven wireless communications and mobile computing," *Wireless Communications and Mobile Computing*, vol. 20192 pages, 2019.

[10] L. Qi, Q. He, F. Chen et al., "Finding all you need: web APIs recommendation in web of things through keywords search," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 1063–1072, 2019.

[11] X. Xu, Q. Liu, Y. Luo et al., "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, pp. 522–533, 2019.

[12] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings of the 1994 IEEE*, Munich, Germany, 1994.

[13] Q. Bu, Z. Wang, and X. Tong, "An improved genetic algorithm for searching for pollution sources," *Water Science and Engineering*, vol. 6, no. 4, pp. 392–401, 2013.

[14] J. Ye, "Tracking control of a nonholonomic mobile robot using compound cosine function neural networks," *Intelligent Service Robotics*, vol. 6, no. 4, pp. 191–198, 2013.

[15] M. M. Mohamad, M. W. Dunnigan, and N. K. Taylor, "Ant colony robot motion planning," in *Proceedings of the EUROCON 2005-The International Conference on "Computer as a Tool"*, November 2006.

[16] F. Zhang, J. Liu, and Q. Li, "A new way of network analysis based on dijkstra," *Remote Sensing Information*, vol. 2, pp. 38–41, 2004.

[17] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[18] F. Duchoň, A. Babinec, M. Kajan et al., "Path planning with modified a star algorithm for a mobile robot," *Procardia Engineering*, vol. 96, no. 1, pp. 159–169, 2014.

[19] S. Uttendorf, B. Eilert, and L. Overmeyer, "Combining a fuzzy inference system with an A∗ algorithm for the automated generation of roadmaps for automated guided vehicles," *At-Automatisierungstechnik*, vol. 65, no. 3, pp. 189–197, 2017.

[20] M. Wodziński and A. Krzyżanowska, "Sequential classification of palm gestures based on A∗ algorithm and MLP neural network for quadrocopter control," *Metrology and Measurement Systems*, vol. 24, no. 2, pp. 265–276, 2017.

[21] S. G. Cui, H. Wang, and L. Yang, "A simulation study of a-star algorithm for robot path planning," *16th Int Conf on Materials. Beijing*, vol. 282, pp. 33–38, 2013.

[22] A. R. Soltani, H. Tawfik, J.Y. Goulermas, and T. Fernando, "Path planning in construction sites: performance evaluation of the Dijkstra, A∗, and GA search algorithms," *Advanced Engineering Informatics*, vol. 16, no. 4, pp. 291–303, 2002.

[23] F. Duchoň, D. Huňady, M. Dekan, and A. Babinec, "Optimal navigation for mobile robot in known environment," *Applied Mechanics and Materials*, vol. 282, no. 1, pp. 33–88, 2013.

[24] Y. Qin, H. Wang, and C. Du, "Mobile robot path planning based on double-layer A∗ algorithms," *Manufacturing Automation*, vol. 24, pp. 21–25, 2014.

[25] M. Hawa, "Light-assisted A∗ path planning," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 2, pp. 888–898, 2013.

[26] D. František, B. Andrej, and K. Martin, "Path planning with modified A star algorithm for a mobilerobot," *Procedia Engineering*, vol. 96, no. 96, pp. 59–69, 2014.

[27] M. Gao, Y. Zhang, and L. Zhu, "Bidirectional time- efficient A∗ algorithm for robot path planning," *Application Research of Computers*, vol. 36, no. 4, pp. 1–6, 2019.

[28] X. Zhao, Z. Wang, and C. Huang, "Mobile robot path planning based on an improved A∗ algorithm," *Robot*, vol. 40, no. 6, pp. 903–910, 2018.

[29] R. Chen, C. Wen, and L. Peng, "Improve A∗ algorithm and apply to indoor path planning for mobile robots," *Journal of Computer Applications*, vol. 39, no. 4, pp. 1006–1011, 2019.

[30] Y. Ren, F. U. Lixia, and Y. Zhang, "Smoothing A∗ algorithm extended search neighborhood for robot path planning," *Electronic Science and Technology*, vol. 31, no. 5, pp. 33–43, 2018.

[31] W. Wang and Z. Feng, "The shortest path planning for mobile robots using improved A∗ algorithm," *Journal of Computer Applications*, vol. 38, no. 5, pp. 1523–1526, 2018.

[32] W. Lu, J. Lei, and Y. Shao, "Path planning for mobile robot based on an improved A∗ algorithm," *Journal of Ordnance Equipment Engineering*, vol. 40, no. 4, pp. 197–201, 2019.

[33] C. Cheng, X. Hao, J. Li et al., "Global dynamic path planning based on fusion of improved A∗ algorithm and dynamic window approach," *Journal of Xi'an Jiaotong University*, vol. 51, no. 11, pp. 137–143, 2017.

[34] Y. Zhang, L.-L. Lin, and H.-C. Lin, "Development of path planning approach using improved A-star algorithm in AGV system," *Journal of Internet Technology*, vol. 20, no. 3, pp. 915–924, 2019.

[35] J. Zhao, Y. Tao, G. Yi et al., "Power control algorithm of cognitive radio based on non-cooperative game theory," *China Communications*, vol. 10, no. 11, pp. 143–154, 2013.

[36] X. Xu, X. Liu, Z. Xu et al., "Trust-oriented IoT service placement for smart cities in edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4084–4091, 2019.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 5218612, 8 pages
https://doi.org/10.1155/2020/5218612

Research Article

Exploiting the Relationship between Pruning Ratio and Compression Effect for Neural Network Model Based on TensorFlow

Bo Liu,[1] **Qilin Wu**,[1] **Yiwen Zhang**,[2] **and Qian Cao**[1]

[1]*College of Information Engineering, Chaohu College, Chaohu 238000, China*
[2]*School of Computer Science and Technology, Anhui University, Hefei 230000, China*

Correspondence should be addressed to Qian Cao; 19875069@qq.com

Received 12 December 2019; Revised 4 February 2020; Accepted 10 February 2020; Published 30 April 2020

Guest Editor: Xiaolong Xu

Pruning is a method of compressing the size of a neural network model, which affects the accuracy and computing time when the model makes a prediction. In this paper, the hypothesis that the pruning proportion is positively correlated with the compression scale of the model but not with the prediction accuracy and calculation time is put forward. For testing the hypothesis, a group of experiments are designed, and MNIST is used as the data set to train a neural network model based on TensorFlow. Based on this model, pruning experiments are carried out to investigate the relationship between pruning proportion and compression effect. For comparison, six different pruning proportions are set, and the experimental results confirm the above hypothesis.

1. Introduction

Model compression is a common method to transplant artificial intelligence from the cloud to the embedded terminal. Network pruning is a particularly effective compression solution for models [1, 2]. In [1, 3], Han et al. proposed a method of compression based on pruning but did not investigate the relationship between pruning proportion and compression effect. At the same time, He et al. [2] studied channel pruning for accelerating very deep neural networks, yet the pruning rate on the prediction effect is not stated. In fact, some studies of pruning methods have been carried out in recent years. However, to the best of our knowledge, there are very few studies on the relationship between the pruning proportion and the size, accuracy, and computing time which is used to make predictions. It is also the motivation of our research.

In a trained neural network model, pruning sets all parameters with values less than a specific threshold to zero. After pruning, retraining and sparsification are normally conducted, where sparsification can delete connections with the zero values to compress the size of the model [4, 5]. As an example, the two figures show the comparison before and after pruning, where Figure 1 shows the original structural diagram, and Figure 2 shows the structural diagram after pruning.

Here, based on TensorFlow, we will use MNIST as the data set to train a neural network model. TensorFlow is an open-source machine learning framework. Specifically, it is software, and users need to build mathematical models by programming in Python and other languages. These models are used in the application of artificial intelligence. MNIST data set is a handwritten data set with 60,000 handwritten digital images in the training library and 10,000 in the test library. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

In this paper, we make the hypothesis that the pruning proportion is positively correlated with the compression scale of the model but not with the prediction accuracy and calculation time. So, our research object is the preliminary relationship between pruning proportion and compression effect in the neural network model. Specifically, this paper studies the relationship from three aspects: first, the

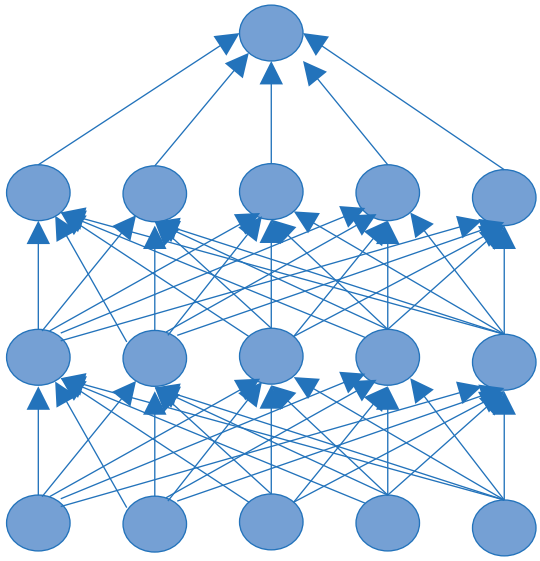


Figure 1: Original structural diagram.

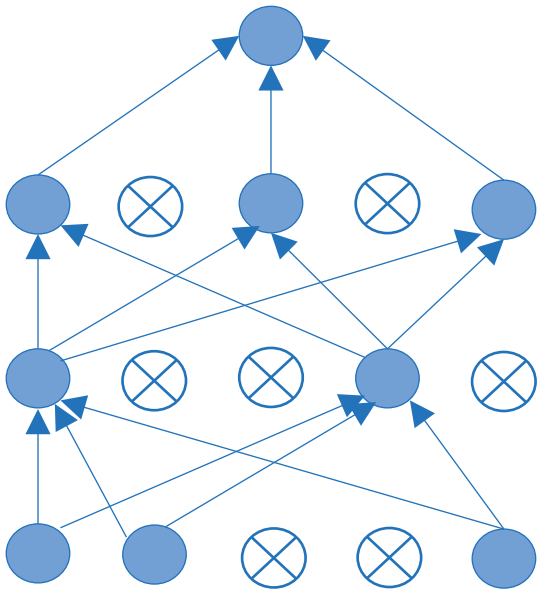


Figure 2: Structural diagram after pruning.

relationship between pruning proportion and model size; second, the relationship between pruning proportion and model prediction accuracy; lastly, the relationship between pruning proportion and computing time for model predictions. For the above objective, a great number of experiments are carried out to investigate the relationship between pruning proportion and compression effect, and the above hypothesis is confirmed, which is our main contribution in this paper.

The rest of this paper is organized as follows. In Section 2, the neural network model is proposed first. To test the hypothesis, an original model and an experimental plan are introduced in Section 3. Section 4 gives the experimental procedures, and Section 5 gives the experimental results and analysis. Finally, Section 6 concludes this paper.

2. Neural Network Model

A neural network is constituted by one input layer, one or several hidden layers, and one output layer, and every layer is constituted by a certain number of neurons. These neurons are interconnected, just like the nerve cells of humans. Figure 3 shows the structure of the neural network.

We assume that $X_i(t) = [x_{i,1}(t), x_{i,2}(t), \ldots, x_{i,D}(t)]$ is the ith individual (solution) in the population. The mutation operator aims to generate mutant solutions. For each solution X_i, a mutant solution V_i is created by the corresponding mutation scheme. There are some classical mutation schemes listed as follows:

(1) DE/rand/1:

$$v_{i,j}(t) = x_{i1,j}(t) + F \cdot \left(x_{i2,j}(t) - x_{i3,j}(t)\right). \tag{1}$$

(2) DE/rand/2:

$$v_{i,j}(t) = x_{i1,j}(t) + F \cdot \left(x_{i2,j}(t) - x_{i3,j}(t)\right) + F \cdot \left(x_{i4,j}(t) - x_{i5,j}(t)\right). \tag{2}$$

(3) DE/best/1:

$$v_{i,j}(t) = x_{\text{best},j}(t) + F \cdot \left(x_{i1,j}(t) - x_{i2,j}(t)\right). \tag{3}$$

(4) DE/best/2:

$$v_{i,j}(t) = x_{\text{best},j}(t) + F \cdot \left(x_{i1,j}(t) - x_{i2,j}(t)\right) + F \cdot \left(x_{i3,j}(t) - x_{i4,j}(t)\right), \tag{4}$$

where $i1$, $i2$, $i3$, $i4$, and $i5$ are five randomly selected individual indices between 1 and N, and $i1 \neq i2 \neq i3 \neq i4 \neq i5$. $F \in [0, 1]$ is usually used. X_{best} is the global best individual (solution).

The crossover operator focuses on recombining two different individuals and creates a new one. In DE, a trial solution U_i is created based on the following crossover operation:

$$\text{swap}(X_{i1}, X_{i2}), \quad \text{if } f(X_{i2}) < f(X_{i1}), \tag{5}$$

where CR is called the crossover rate, the random value rand_j is in the range [0, 1], and j_r is a randomly selected dimension index. As seen, U_i inherits from V_i and X_i based on the value of CR. For a large CR, most dimensions of U_i are taken from V_i. For a small CR, most dimensions of U_i are taken from X_i. For the latter case, U_i is similar to its parent X_i.

3. Design of the Experiment

3.1. Structure of the Original Model. The basic neural network structure consists of the following layers in sequence: convolutional layer, pooling layer, convolutional layer, pooling layer, and two fully connected layers [6, 7], which is shown in Figure 4. In the experiment plan, pruning is performed by default on the weight parameters w of the two fully connected layers. Alternative pruning is performed on

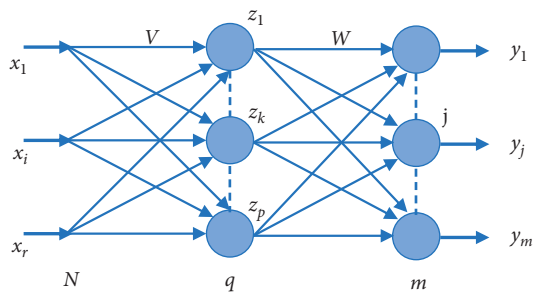


Figure 3: Structure of the neural network model.

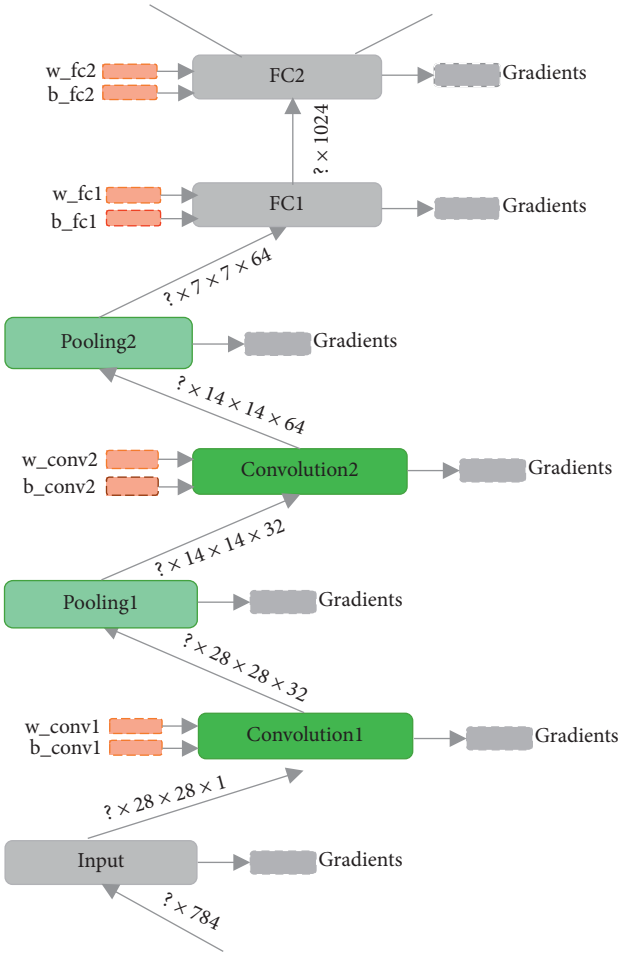


Figure 4: Basic structure of the neural network of the experiment.

all network parameters, and the specific operations are executed by changing the command line parameters [8, 9].

3.2. Experiment Plan. The experiment is based on the TensorFlow framework and used MNIST as the dataset. An original model is trained in the beginning, and then six pruning practices with different pruning proportions are employed [10, 11]. For each pruning, retraining and sparsification are subsequently performed. When all three operations are completed on the original model, the task of pruning compression is also finished [12, 13]. Then, the data are collected and analysed for comparison (size, accuracy, and computing time for making predictions).

4. Experimental Procedures

4.1. Run Command of the Pruning Experiment. Model pruning is executed by the following command: python train.py −1 −2 −3 --train_data_dir /tmp/mnist_data --train_dir /tmp/mnist_train --variables_dir /tmp/mnist_-variables --max_steps 10000 --batch_size 32 --sparse_ratio 0.9 --pruning_variable_names w_fc1, w_fc2. Table 1 specifies the parameters in this command [14–16].

4.2. Pruning Effect View Command. The effects of the −1 or −2 parameters can be viewed through eval_-predict_with_dense_network.py. The specific command is python eval_predict_with_dense_network.py --test_data_-dir /tmp/mnist_data --checkpoint_dir /tmp/mnist_train/step_2_2 --batch_size 32 --max_steps 10. Table 2 specifies the parameters in this command [17–19].

The effect of −3 sparsification can be viewed through eval_predict_with_sparse_network.py. The specific command is python eval_predict_with_sparse_network. py --test_data_dir /tmp/mnist_data --checkpoint_dir /tmp/mnist_train/step_3 --batch_size 32 --max_steps 10. Table 3 specifies the parameters in the command.

4.3. Hardware and Software Configuration of the Experiment. Pruning experiments are based on the following hardware and software parameters and versions [20, 21]:

Operating system: Windows 10

GPU: NVIDIA GeForce GTX 1080 Ti 11.0 G

CPU: Intel(R) Core(TM) i3-4160 CPU @ 3.60 GHz

Memory: 16.0 GB DDR3

Disk: Lenovo SSD SL700 240G

Software: TensorFlow-GPU 1.5.0, Python 3.6

4.4. Construction of Experimental Environment. The experiment is based on the MNIST dataset and the TensorFlow framework. The experimental environment was constructed by the following three steps [22–24]:

Step 1: constructing the Python environment. Directly following Anaconda and then directly adding and running Anaconda3-4.3.1-Windows-x86_64.exe.

Step 2: constructing the plug-ins of NVIDIA GPU. Directly running the cuda_9.0.103_win10.exe for installation. Unzipping cudnn-9.0-windows10-x64-v7.zip and copying its contents to the folder C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0.

Step 3: constructing the TensorFlow environment. Executing the installation command on the CMD commands: pip install tensorflow-gpu = = 1.5.0.

Table 1: Descriptions of parameters of the pruning command.

Parameter	Description	Default value
−1	Most basic model training	
−2	Pruning is performed on the most basic model	
−3	Pruning is performed, and network parameters of sparsification are stored	
--train_data_dir	Folder path containing training data	/tmp/mnist_data
--train_dir	Log data during training and folder path containing model data	/tmp/mnist_train
--variables_dir	Folder containing data of the last trained model	/tmp/ mnist_variables
--max_steps	Number of iterations of the basic model	10,000
--batch_size	Sample data size of each batch during model training	32
--sparse_ratio	Percentage of compression parameter of pruning/volume of parameters set to 0	0.9
--pruning_variable_names	Parameters on which pruning could be performed. Optional parameters include w_conv1, w_conv2, w_fc1, and w_fc2	w_fc1, w_fc2

Table 2: Descriptions of parameters of the pruning effect view command (view effects of −1 or −2 training).

Parameter	Description	Default value
--test_data_dir	Folder path containing test data	/tmp/mnist_data
--checkpoint_dir	Folder path of the trained model	/tmp/mnist_train/step_2_2
--variables_dir	Folder containing data of the last trained model	/tmp/mnist_variables
--max_steps	Number of iterations during model testing	10
--batch_size	Size of sample data of each batch used for calculation during model testing	32

Table 3: Descriptions of parameters of the pruning effect view command (view effects of −3 sparsification).

Parameter	Description	Default value
--test_data_dir	Folder path containing test data	/tmp/mnist_data
--checkpoint_dir	Folder path of the trained model	/tmp/mnist_train/step_3
--variables_dir	Folder containing data of the last trained model	/tmp/mnist_variables
--max_steps	Number of iterations during model testing	10
--batch_size	Size of sample data of each batch used for calculation during model testing	32

5. Experimental Results and Analysis

In this section, six different pruning proportions are employed in this experiment. The six groups of tables show the specific data of pruning proportion, model size, accuracy, and computing time for predictions.

5.1. 10% Pruning Proportion. First, the pruning proportion is set to 10%; Table 4 shows the parameters of pruning effect in the first scene. In this group of experiments, the parameter threshold values of the two fully connected layers are set to 0.012034996 and 0.013038448. In this way, the valid parameter numbers are reduced from 3,211,264 and 10,240 to 2,890,137 and 9,215, respectively, making exactly 10% of the parameter values of the two fully connected layers equal to 0. However, the model size after the pruning, retraining, and sparsification is 66.5 M, which is larger than the size (37.5 M) of the original model. Hence, no compression effect is achieved. In addition, compared with the original model, the accuracy does not change, and the computing time for predictions slightly increases [25, 26].

Table 4: Descriptions of parameters of the pruning effect in the first scene.

	Original number of parameters	Threshold value	Valid parameter number
w_conv1	800	0.0	800
w_conv2	51,200	0.0	51,200
w_fc1	3,211,264	0.012034996	2,890,137
w_fc2	10,240	0.013038448	9,215

5.2. 30% Pruning Proportion. Second, the pruning proportion is set to 30%; Table 5 shows the parameters of pruning effect in the first scene. In this group of experiments, the parameter threshold values of the two fully connected layers are set to 0.036936015 and 0.039559085. In this way, the valid parameter numbers are reduced from 3,211,264 and 10,240 to 2,247,884 and 7,167, respectively, making exactly 30% of the parameter values of the two fully connected layers equal to 0. However, the model size after the pruning, retraining, and sparsification is 51.8 M, which is larger than the size (37.5 M) of the original model. Hence, no compression effect was achieved. Again, compared with the

Table 5: Descriptions of parameters of the pruning effect in the second scene.

	Original number of parameters	Threshold value	Valid parameter number
w_conv1	800	0.0	800
w_conv2	51,200	0.0	51,200
w_fc1	3,211,264	0.036936015	2,247,884
w_fc2	10,240	0.039559085	7,167

Table 6: Descriptions of parameters of the pruning effect in the third scene.

	Original number of parameters	Threshold value	Valid parameter number
w_conv1	800	0.0	800
w_conv2	51,200	0.0	51,200
w_fc1	3,211,264	0.06429165	1,605,631
w_fc2	10,240	0.068891354	5,119

original model, the accuracy does not change, and the computing time for predictions slightly increases.

5.3. 50% Pruning Proportion. Third, the pruning proportion is set to 50%; Table 6 shows the parameters of pruning effect in the first scene. In this group of experiments, the parameter threshold values of the two fully connected layers are set to 0.06429165 and 0.068891354. In this way, the valid parameter numbers are reduced from 3,211,264 and 10,240 to 1,605,631 and 5,119, respectively, making exactly 50% of the parameter values of the two fully connected layers equal to 0. The model size after pruning, retraining, and sparsification is 37.1 M, which is slightly smaller than the size (37.5 M) of the original model. Here, compression takes effect. Besides, both accuracy and computing time for predictions slightly decrease as compared with those of the original model.

5.4. 70% Pruning Proportion. Fourth, the pruning proportion is set to 70%; Table 7 shows the parameters of pruning effect in the fourth scene. In this group of experiments, the parameter threshold values of the two fully connected layers are set to 0.09749276 and 0.10360378. In this way, the valid parameter numbers are reduced from 3,211,264 and 10,240 to 963,379 and 3,071, respectively, making exactly 70% of the parameter values of the two fully connected layers equal to 0. The model size after pruning, retraining, and sparsification is 22.3 M, which is smaller than the size (37.5 M) of the original model. The compression effect is obvious. Moreover, both accuracy and computing time for predictions slightly decrease as compared with those of the original model.

5.5. 80% Pruning Proportion. Fifth, the pruning proportion is set to 80%; Table 8 shows the parameters of pruning effect in the fifth scene. In this group of experiments, the parameter threshold values of the two fully connected layers are set to 0.11903707 and 0.12662686. In this way, the valid parameter numbers are reduced from 3,211,264 and 10,240 to 642,252 and 2,047, respectively, making exactly 80% of the parameter values of the two fully connected layers equal to 0. The model size after pruning, retraining, and sparsification is 14.9 M, which is smaller than the size (37.5 M) of the original model, and compression is 60%. Additionally, as compared with the original model, the accuracy slightly decreases and the computing time for predictions slightly increases.

Table 7: Descriptions of parameters of the pruning effect in the fourth scene.

	Original number of parameters	Threshold value	Valid parameter number
w_conv1	800	0.0	800
w_conv2	51,200	0.0	51,200
w_fc1	3,211,264	0.09749276	963,379
w_fc2	10,240	0.10360378	3,071

5.6. 90% Pruning Proportion. Lastly, the pruning proportion is set to 90%; Table 9 shows the parameters of pruning effect in the sixth scene. In this group of experiments, the parameter threshold values of the two fully connected layers are set to 0.14814831 and 0.15710811. In this way, the valid parameter numbers are reduced from 3,211,264 and 10,240 to 321,126 and 1,023, respectively, making exactly 90% of the parameter values of the two fully connected layers equal to 0. The model size after pruning, retraining, and sparsification is 7.6 M, which is compressed by 80%. Furthermore, both accuracy and computing time for predictions slightly decreased as compared with those of the original model.

5.7. Comparison Results. Figure 5 shows the comparison results for persistence model size of the four networks, with the pruning ratio increases, and the model size represented by the red columns decreases gradually. Apparently, the pruning proportion is positively correlated with the model size.

Figure 6 shows the comparison results for testing accuracy of the four networks, with the pruning ratio increases, and the testing accuracy represented by the red columns has no obvious changes. This means that there is no positive relationship between pruning proportion and accuracy.

Table 8: Descriptions of parameters of the pruning effect in the fifth scene.

	Original number of parameters	Threshold value	Valid parameter number
w_conv1	800	0.0	800
w_conv2	51,200	0.0	51,200
w_fc1	3,211,264	0.11903707	642,252
w_fc2	10,240	0.12662686	2,047

Table 9: Descriptions of parameters of the pruning effect in the sixth scene.

	Original number of parameters	Threshold value	Valid parameter number
w_conv1	800	0.0	800
w_conv2	51,200	0.0	51,200
w_fc1	3,211,264	0.14814831	321,126
w_fc2	10,240	0.15710811	1,023

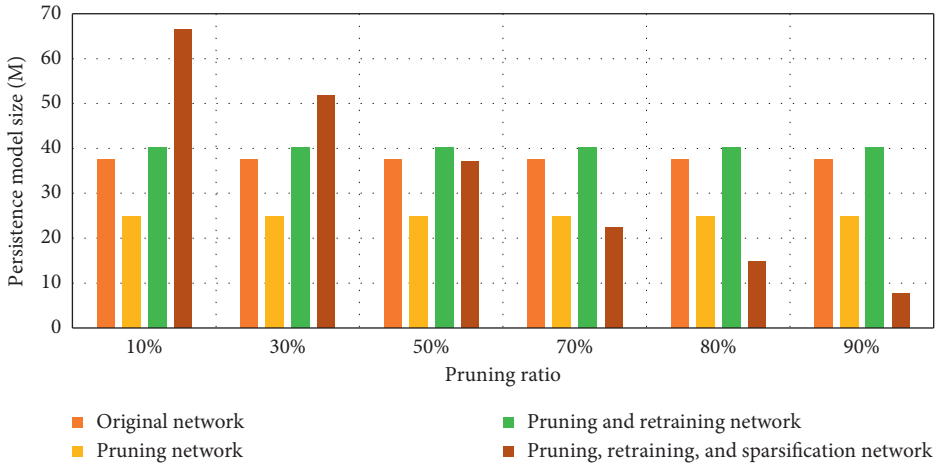


Figure 5: Comparison results for persistence model size of the four networks.

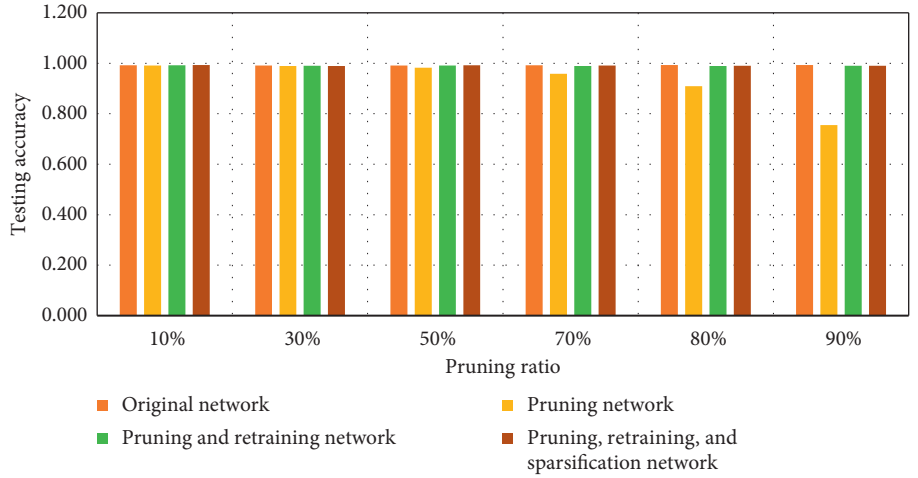


Figure 6: Comparison results for testing accuracy of the four networks.

Figure 7 shows the comparison results for computing time of the four networks. With the pruning ratio increases, the computing time for prediction represented by the red columns changes irregularly. Also, there is no positive relationship between pruning ratio and computing time for predictions.

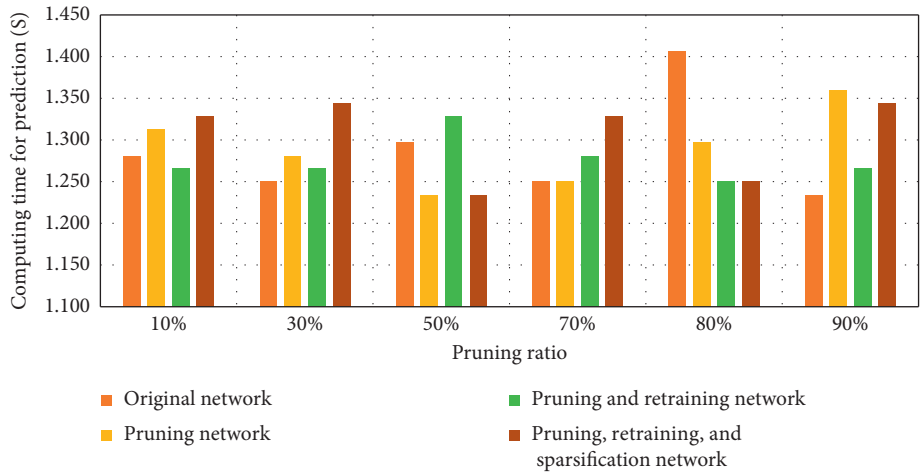


Figure 7: Comparison results for computing time of the four networks.

6. Conclusions

By comparing the experimental data of six different pruning proportions, it is found that pruning does not necessarily compress the size of the model. Compression takes effect only when the pruning proportion reaches 50% or more. Furthermore, we found a positive relationship between the pruning proportion and the model size. However, there was no positive relationship between pruning proportion and accuracy and between pruning proportion and computing time for predictions.

Since there is no specific experimental verification for other models, the conclusion does not apply to other models. Additionally, the experimental is based on the pruning method, pruning is only one of the compression methods of various models; thus, the conclusion of this study is not applicable to other compression methods [27–29].

Data Availability

The data used to support the findings of this study can be accessed publicly in the website http://yann.lecun.com/exdb/mnist/.

Conflicts of Interest

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

Acknowledgments

This work was supported by the key project of the Natural Science Research of Higher Education Institutions in Anhui Province (grant no. KJ2018A0461); Anhui Province Key Research and Development Program Project (grant no. 201904a05020091); and a provincial quality engineering project from Department of Education Anhui Province (grant no. 2019mooc283).

References

[1] S. Han, J. Pool, S. Narang, H. Mao et al., "DSD: dense-sparse-dense training for deep neural networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1–13, Toulon, France, April 2017.

[2] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[3] S. Han, J. Kang, H. Mao et al., "ESE," in *Proceedings of the Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays—FPGA '17*, pp. 75–84, Monterey, California, USA, February 2017.

[4] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," 2016, https://arxiv.org/abs/1608.08710.

[5] Y. Zhang, G. Cui, S. Deng et al., "Efficient query of quality correlation for service composition," *IEEE Transactions on Services Computing*, 2018.

[6] C. Wan, X. Yan, D. Zhang, Z. Qu et al., "An advanced fuzzy Bayesian-based FMEA approach for assessing maritime supply chain risks," *Transportation Research Part E: Logistics and Transportation Review*, vol. 125, pp. 222–240, 2019.

[7] L. Qi, Y. Chen, Y. Yuan, S. Fu et al., "A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web*, vol. 23, no. 2, pp. 1275–1297, 2019.

[8] Z. Huang, G. Shan, J. Cheng, and J. Sun, "TRec: an efficient recommendation system for hunting passengers with deep neural networks," *Neural Computing and Applications*, vol. 31, no. 1, pp. 209–222, 2019.

[9] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, p. 32, 2017.

[10] S. Han, H. Mao, and W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, https://arxiv.org/abs/1510.00149.

[11] B. Wu, X. Yan, Y. Wang, and C. Guedes Soares, "An evidential reasoning-based CREAM to human reliability analysis in maritime accident process," *Risk Analysis*, vol. 37, no. 10, pp. 1936–1957, 2017.

[12] B. Wu, L. Zong, X. Yan, and C. Guedes Soares, "Incorporating evidential reasoning and TOPSIS into group decision-making under uncertainty for handling ship without command," *Ocean Engineering*, vol. 164, pp. 590–603, 2018.

[13] Y. Wang, E. Zio, X. Wei, D. Zhang, and B. Wu, "A resilience perspective on water transport systems: the case of Eastern Star," *International Journal of Disaster Risk Reduction*, vol. 33, pp. 343–354, 2019.

[14] L. Qi, X. Zhang, S. Li, S. Wan et al., "Spatial-temporal data-driven service recommendation with privacy-preservation," *Information Sciences*, vol. 515, pp. 91–102, 2020.

[15] R. Zhu, Z. Sun, T. Ristaniemi, and J. Hu, "Special issue on green telecommunications," *Telecommunication Systems*, vol. 52, no. 2, pp. 1233-1234, 2013.

[16] R. Zhu, W. Shu, T. Mao, and T. Deng, "Enhanced MAC protocol to support multimedia traffic in cognitive wireless mesh networks," *Multimedia Tools and Applications*, vol. 67, no. 1, pp. 269–288, 2013.

[17] D. Zhang, R. Zhu, S. Men, and V. Raychoudhury, "Query representation with global consistency on user click graph," *Journal of Internet Technology*, vol. 14, no. 5, pp. 759–769, 2013.

[18] J. Chang, H. Chao, C. Lai, and R. Zhu, "An efficient geographic routing protocol design in vehicular ad-hoc network," *Computing*, vol. 96, no. 2, pp. 119–131, 2014.

[19] K. Zhu, R. Zhu, H. Nii, H. Samani et al., "PaperIO: a 3D interface towards the internet of embedded paper-craft," *IEICE Transactions on Information and Systems*, vol. E97.D, no. 10, pp. 2597–2605, 2014.

[20] Y. Jalaeian, C. Yin, Q. Wu et al., "Location-aware deep collaborative filtering for service recommendation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems (TSMC)*, pp. 1–12, 2019.

[21] Y. Ma, W. Cho, J. Chen, Y. Huang, and R. Zhu, "RFID-based Mobility for seamless personal communication system in cloud computing," *Telecommunication Systems*, vol. 58, no. 3, pp. 233–241, 2015.

[22] L. Qi, Q. He, F. Chen et al., "Finding all you need: web APIs recommendation in web of Things through keywords search," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 1063–1072, 2019.

[23] X. Xu, Y. Xue, L. Qi, Y. Yuan, X. Zhang et al., "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, pp. 89–100, 2019.

[24] K. Guo, S. Han, S. Yao, Y. Wang et al., "Software-hardware co-design for efficient neural network acceleration," *IEEE Micro*, vol. 37, no. 2, pp. 8–25, 2017.

[25] X. Xu, Y. Li, T. Huang, Y. Xue et al., "An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks," *Journal of Network and Computer Applications*, vol. 133, pp. 75–85, 2019.

[26] Y. Peng, K. Wang, Q. He et al., "Covering-based web service quality prediction via neighborhood-aware matrix factorization," *IEEE Transactions on Services Computing*, 2019.

[27] X. Xu, Q. Liu, Y. Luo, K. Peng et al., "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, pp. 522–533, 2019.

[28] B. Jalaeian, R. Zhu, H. Samani, and M. Motani, "An optimal cross-layer framework for cognitive radio network under Interference Temperature model," *IEEE Systems Journal*, vol. 10, no. 1, pp. 293–301, 2014.

[29] T. Zhou, C. Wu, J. Zhang, and D. Zhang, "Incorporating CREAM and MCS into fault tree analysis of LNG carrier spill accidents," *Safety Science*, vol. 96, pp. 183–191, 2019.