Hindawi
Security and Communication Networks
Volume 2019, Article ID 3898951, 12 pages
https://doi.org/10.1155/2019/3898951

Research Article

An Insider Threat Detection Approach Based on Mouse Dynamics and Deep Learning

Teng Hu,[1,2] **Weina Niu**,[3] **Xiaosong Zhang**,[1] **Xiaolei Liu**,[4] **Jiazhong Lu**,[1] **and Yuan Liu**[2]

[1]*Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, China*
[2]*Institute of Computer Application, China Academy of Engineering Physics, Mianyang, Sichuan, 621900, China*
[3]*College of Cybersecurity, Sichuan University, Chengdu, Sichuan, China*
[4]*School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China*

Correspondence should be addressed to Weina Niu; niuweina1@126.com

Received 25 September 2018; Revised 21 November 2018; Accepted 16 January 2019; Published 17 February 2019

Guest Editor: Jiageng Chen

In the current intranet environment, information is becoming more readily accessed and replicated across a wide range of interconnected systems. Anyone using the intranet computer may access content that he does not have permission to access. For an insider attacker, it is relatively easy to steal a colleague's password or use an unattended computer to launch an attack. A common one-time user authentication method may not work in this situation. In this paper, we propose a user authentication method based on mouse biobehavioral characteristics and deep learning, which can accurately and efficiently perform continuous identity authentication on current computer users, thus to address insider threats. We used an open-source dataset with ten users to carry out experiments, and the experimental results demonstrated the effectiveness of the approach. This approach can complete a user authentication task approximately every 7 seconds, with a false acceptance rate of 2.94% and a false rejection rate of 2.28%.

1. Introduction

Insider threats have always been one of the most severe challenges for intranets with security requirements [1], because they can cause system destruction, information exfiltration, etc. In recent years, with the frequent occurrence of insider threat events, intranet security has aroused increasing attention [2, 3].

Internal personnel have access to use internal proprietary systems, and they know internal security policies and protection techniques and review regulations from safety facilities [4], e.g., firewalls and IDS. Hence, internal personnel can bypass existing security facilities. Even worse, a malicious insider may also be the one who configures security measures [5]. Moreover, a cyberattack from insiders is prevalent within the organization [6]; according to the survey, 27% of all cybercrime incidents were suspected to be committed by insiders [2]. There are two reasons for these insider threats; on the one hand, employees with malicious intentions modify or steal organization's confidential information, trade secrets, or customer data for personal interests [7]. For example, insiders make use of sensitive information to obtain commercial interests or sell them to foreign organizations. On the other hand, internal employees with inadvertent behavior expose the organization's key assets and sensitive information to external opponents [8]. Furthermore, in some confidential organizations, the insider threat attacks may even be spy activities at the national level [8, 9]. Thus, the effective detection methods are worth studying.

The organizations in physical isolation high-security networks environment, such as confidential research institutes and military enterprises, are less likely to suffer external attacks. Thus, the internal attacks are the main reason for the leakage of sensitive information [10]. Fraudulent use of privileged users' terminals is the primary attack method for such internal threats [11].

There are two main reasons for this attack:

(1) Due to the negligence of internal employees, they left the workstation without logging out of the terminal equipment. As a result, malicious people use their terminals to unauthorized access and copy sensitive information;

(2) Others forge a privileged user's identity, such as password leakage or USB-key loss, and thus access privileged user's terminal for espionage activities.

Traditional authentication methods, such as passwords, USB keys, and fingerprints, determine the user identity when logging in. Thus, they cannot effectively discover end users with identity theft [12]. However, some approaches conduct continued authentication using human-computer interaction (HCI) behaviors [13] to solve the problem. HCI behaviors are unique biometric features from input devices like keyboards and mouses. For example, some researchers used interventional scenarios to capture HCI behaviors, like recording the user's responses during equipment failures. However, the interventional scheme has directly affected the convenience of using the devices, which may be identified by malicious users.

This paper proposed a continuous identity authentication method based on mouse dynamic behavior and deep learning to solve the insider threat attack detection problem. We verified the effectiveness of our proposed method on an open-source mouse dynamic dataset which contains the mouse dynamic data from ten users. The experimental results showed that our proposed method could identify the user's identities in seconds and has a lower false accept rate (FAR) and false reject rate (FRR). Specifically, the contributions of the work are as follows:

(i) We propose a novel continuous identity authentication method using mouse dynamic behavior and deep learning. It achieves better accuracy and lower verification time than existing methods.

(ii) Instead of manually extracting features from raw operations to characterize a user's unique mouse behavior characteristics, such as movement speed curves, we map the mouse dynamic behavior into pictures. Hence, the whole details of the mouse behavior can be preserved.

(iii) We construct a 7-layer CNN network to train the mouse behavior pictures datasets. The network converges with a small amount of data (about 18000 pictures). Moreover, the network can be used to train other mouse behavior datasets and implement identity authentication easily.

The remainder of this paper is organized as follows: Section 2 reviews the background and related work in the area of insider threat and mouse dynamics. Section 3 describes the method of preprocessing the dataset and the CNN network architecture and specific parameters. Section 4 presents our experimental design and results, and Section 5 concludes.

2. Background and Related Work

As early as 2006, the American Institute of Computer Security (CSI) issued a report that the insider threat caused by the malicious abuse of authority has exceeded the traditional Trojan attacks and has become the main threat to organizations [14]. The 2012 Annual Global Fraud Survey revealed that 60% of fraud cases were initiated by insiders [15]. The Edward Snowden incident in June 2013, known as "PRISM," caused widespread concern about insider threats all over the world. The 2014 US State of Cybercrime Survey released by CERT showed that 28% of insider attacks resulted in a loss of 46% [16]. In the same year, insider threats caused incredible damage to many well-known companies: for example, the Korean Credit Bureau has had 27 million accounts of credit card information stolen because of abusing access rights by its computer contractor [17]; the dismissed employee retaliated against US oil and gas company EnerVest, and all the web servers are restored to factory settings, thus leading to 30 days of disruption in overall communications and business operations and recovery costs of hundreds of thousands of dollars [18]. The 2016 US State Cybercrime Survey found that insiders caused 27% of e-criminals. The investigation report also revealed that 30% of the respondents believe that the insider attacks caused more serious losses than outsider attacks [19]. According to the 2017 CSO Cybercrime Survey [20], about 50 percent of organizations experienced at least one malicious insider incident in the previous year [21]. The above examples fully proved that insider threats had become the main cyberspace security threat faced by organizations.

The current research on insider threats is becoming more systematic and specific. Since 2001, the United States Secret Service (USSS) and Carnegie Mellon University have jointly established the CERT Insider Threats Center. The center collected more than 700 insider threat cases of fraud, theft, and destruction and thus solved the problem of insufficient data in insider threat research [22]. In 2011, the United States DARPA proposed building up a military insider threat detection system named ADAMS (Anomaly Detection at Multiple Scales) [23]. In the third year of the ADAMS project implementation, the SAIC company and four universities in the United States, including Carnegie Mellon University, jointly developed a realistic version of the ADAMS called PRODIGAL system and run tests on the actual enterprise data achieving good results [24].

The most common attack on insider threat is identity fraud. Since the user's identity cannot be continuously verified during the process of using the terminal, the malicious user has the opportunity to masquerade as a legitimate user. At present, research on the use of human-computer interaction data for insider threat attack detection has achieved good results in practical applications. The human-computer interaction data detection mainly studies the behavior pattern of the user using the computer input device, in which the mouse and the keyboard are mainly used as the data source.

In [12], the user's mouse operation when using IE browser was collected and based on these operations; the features such as moving coordinates, moving distance, angle, and moving time were constructed. The literature [25] focuses on

the three basic types of mouse movements, clicks, and drags and uses them to establish characteristics such as coordinates and speed. The deficiency of the above work is that the experimental data is too small, and malicious data adopts simulation data, which cannot truly reflect the actual attack conditions.

Research using keyboard data includes static text analysis and dynamic text analysis. For example, in [26], user input passwords are used to analyze the changes in user input methods. This study of user input of the same text is a static text analysis method. Another dynamic analysis is to study the user's arbitrary input of text. For example, the paper [27] analyzes the user's mail information and mainly records the keystroke behavior and time stamp. The insufficiency of this kind of research is that the habit of using the keyboard even by the same user may constantly be changing. It needs a large amount of input data to truly reflect a user's actual behavior characteristics, which is unrealistic in practical application.

Among different input devices, the mouse is a suitable choice. In some existing methods based on mouse dynamics, papers [12, 28, 29] have achieved low false reject rate (FRR) and false accept rate (FAR). For example, in the paper [28] FRR and FAR are approximately 2%. However, these methods have a common limitation. They require an average of several minutes (even more than 10 minutes) to collect enough mouse behavior for verification [30]. In the actual situation, attacks from insiders may only take tens of seconds or even seconds to complete the attack, such as copying files with sensitive information, sending emails with viruses, or just implanting one trojan in the machine. The methods commonly used in existing research are rough as follows: they collect data from users' valid mouse movements, extract selected features from them, and use these features to train models for classification. For example, the literature [28] uses the backpropagation neural network, and the literature [12] uses the C5.0 decision tree. There are two obvious disadvantages to these methods. One is the behavioral features of the mouse are artificial selected, which will inevitably lead to the loss of some details behavior; the other is that these methods need to collect a large number of valid movements when verifying the user's identity, so the verification takes longer.

The paper [13] used an interventional scenario method. In these scenarios, the user's mouse was disabled for a short period and collects the behavioral data generated by the user due to the loss of the mouse cursor during this period. The data for this period was used to verify the user's identity, achieving 5 seconds of verification and having 2.86% of FRR and 4% of FAR. However, this method also has obvious shortcomings. Regularly adopting an interventional scenario to verify user identity will inevitably seriously affect the convenience of using the terminal, and the attacker will easily perceive it. If the interval between two verifications is too long, the attacker will probably be able to complete the attack within the interval.

There are also some other approaches to detect insider threats. Reference [31] presents an ontological framework and a methodology for improving physical security and insider threat detection (called PS0). PS0 can facilitate forensic data analysis and proactively mitigate insider threats by leveraging rule-based anomaly detection. Reference [32] uses machine learning algorithms and extracts the frequently used words from the topic modeling technique and then verifies the analysis results by matching them to the information security compliance elements, to find the possible malicious insider from social media. Reference [33] implements a fuzzy classifier along with genetic algorithm (GA) to enhance the efficiency of a fuzzy classifier and the functionality of all other modules, to achieve better results in terms of false alarms. Reference [34] applies Hidden Markov method on a CERT dataset and analyses a number of distance vector methods in order to detect changes of behavior, which are shown to have success in determining different insider threats.

3. Overview of Our Approach

From previous research on insider threat detection based on mouse behavior, researchers usually extract some mouse features based on the basic mouse movements (which we call raw data, including clicks, moves, and drags), such as direction, velocity, Angle of Curvature, Curvature Distance, and Pause-and-Click [35], and then go through these features to determine human behavior characteristics, to conduct user authentication [36].

This kind of method has achieved outstanding results, but all have a common shortcoming. Researchers are all extracting features based on their own experience and understanding. This method has certain limitations; during the process of extracting features, some researchers use only mouse click actions; some researchers use the moving distance combined with clicks to generate features, and others consider other basic mouse actions. But there is still much raw data that can reflect the individual's unique being ignored and therefore affects the accuracy of recognition.

We propose a method that can completely retain all basic mouse operations and use deep learning for user authentication. First of all, we map all actions generated by the mouse to images through a particular method. Then we train the image datasets through the CNN network to create classification models. In the process of authentication, the user's mouse operation is mapped according to the same method, and then classified by the trained model, so as to achieve the purpose of user identification. Our approach makes full use of the advantages of deep learning. First, in the process of mapping mouse behaviors into images, we retain all basic mouse operations. Neither need to manual extract features to train these image datasets when using the CNN network, nor need the feature extraction algorithm in traditional machine learning. The convolutional neural network can automatically complete feature extraction and abstraction in the training. Secondly, there are many successful and efficient solutions to the problem of how to use CNN to classify images. In order to take this advantage, we map the mouse actions to the behavior trajectory on the picture, which based on mouse behavior. This turns the problem of the user authentication based on mouse behavior into a classic image classification problem.

3.1. Data Preprocessing

3.1.1. Dataset. Many of the previous studies collect the raw mouse data by themselves and use for analysis and

experimentation. Most of the datasets have not been disclosed. Even if some researchers open source their own datasets, they are currently unavailable for download. To prove the effectiveness of our method, we choose to use the open-source mouse dynamic dataset, which published on GitHub [37], for our experiments.

The dataset stores the mouse dynamic data from ten users. The dataset consists of two parts, one part is stored in the "training_files" folder and contains data session files from ten users who normally operate the mouse, and store them separately in their respective named folders. There are 5-7 long session data files in each user folder; the other part stores them in the "test_files" folder, which also contains ten folders named after ten users. Each user folder stores many short session data files. Some of these session files are actually not generated by current users. The dataset also gives a "public_labels.csv" file that marks whether the session data in the folder is legal or not.

Each session is stored in the following format:

$$[\text{record timestamp}, \text{client timestamp}, \text{button}, \text{state}, x, y] \quad (1)$$

The specific meanings are as follows [37]:

record timestamp: elapsed time (in a sec) since the start of the session as recorded by the network monitoring device;

client timestamp: elapsed time (in a sec) since the start of the session as recorded by the RDP client;

button: the current condition of the mouse buttons;

state: additional information about the current state of the mouse;

x: the x coordinate of the cursor on the screen;

y: the y coordinate of the cursor on the screen;

The data in "training_files" is quite complete and large enough compared to the data in "test_files". We divided "training_files" into two parts, one to train our model and the other to verify whether our model is effective or not. In addition, we use "test_files" to further verify that our model is still accurate enough for insider threat detection.

3.1.2. Mouse Dynamic Mapping Method. All the basic operations that the mouse can produce are move, Click, Drag, Scroll, and Stay. In previous studies, the researchers usually extract features from these five basic actions based on their experience. Such as moving distance, moving speed, click frequency, etc., which can reflect the behavior of individuals using the mouse to a certain extent. However, there are also two obvious shortcomings. First, the use of a single feature (or a combination of multiple feature vectors) does not fully or accurately reflect the individual's unique behavioral characteristics; the second is that more time is required in the process of acquiring features, because the feature vectors (some researchers call them effective operations) are extracted from the basic operations. Undoubtedly, relatively more basic actions are needed to generate enough effective operations.

In order to completely preserve the features generated by people using the mouse, we map all the basic actions generated by users to the image. Because the data generated by the mouse is a one-dimensional dataset, we map these one-dimensional datasets to two-dimensional tensors according to the following mapping method. The specific mapping method is as follows:

Figure 1: The position of the mouse is represented by a two-dimensional coordinate system (x, y), and the distance between two mouse positions is expressed as a movement feature of the mouse. The movement features can reflect the personality characteristics of users such as moving speed, moving angle, moving range, and average moving distance.

Require 1. m, the number of mouse basic actions;

Require 2. xMax, the maximum x coordinate in a session; yMax, the maximum y coordinate in a session;

Step 1. Take m mouse operations according to the data sequence of the session;

Step 2. Construct a coordinate system D based on xMax and yMax;

Step 3. Extract all the "Move" operations in this m, record all the coordinates of "Move" $(x, y) \in [\text{xMove}, \text{yMove}]$, and map a red color line on D according to the coordinate distance of two actions operations, as shown in Figure 1;

Step 4. Extract all the "Pressed" operations in m, the pressed operation $(x, y) \in [\text{xLeft\_P}, \text{yLeft\_P}]$ for the left button, and $(x, y) \in [\text{xRight\_P}, \text{Right\_P}]$ for the right button. Similarly, all "Released" operations' coordinates of the left and right button are recorded in [xLeft_R, yLeft_R] and [xRight_R, Right_R], respectively. To distinguish between the left and the right buttons and the difference between "Pressed" and "Released" in the image, the left button "Pressed" is represented by a blue "circle" on D, the left button "Released" is represented by a green "x", the right button "Pressed" is represented by a black ".", and the right button "Released" is represented by a black "x", as shown in Figure 2;

Step 5. Extract all the "Drag" operations and $(x, y) \in [\text{xDrag}, \text{yDrag}]$ is represented by a thick yellow line on D, as shown in Figure 3;

Step 6. In the same way, all "Scrolls" are extracted. Scroll's "Up" operations are recorded in [scroll_up_x, scroll_up_y] and "Down" operations are recorded in [scroll_down_x, scroll_down_y]. The "Scroll Up" is represented by a cyan upward triangle"△" and the "Scroll Down" is represented by "▽", as shown in Figure 4;

Step 7. In this m, if there are two consecutive actions on the same coordinate and stay for a while, we believe that this

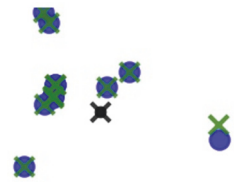


Figure 2: The click action is actually made up of two actions: Pressed and Released. So a "click" action is actually done by a Pressed and a Released. Two consecutive times on the same coordinate (x, y) becomes a "double click" action. In addition, the mapping method also needs to distinguish the left and right buttons of the mouse, that is, "click" and "double-click" of the left mouse button and "click" and "double-click" of the right mouse button. It can reflect the features of the number of clicks, frequencies, and other features.

Figure 3: The mouse did not release immediately after pressing "Pressed" but "drag" a distance and then "Released," which is a drag operation. Drag operations are also common in actual mouse use and are often not noticed as features of the user.

period can also reflect the personal operating habits and call it a "Stay" operation. (x, y, s) ∈ [stay_x, stay_y, stay_s] represents all "Stay" operations in m, where x and y are the coordinates of the "Stay" and s represents the relative time of operation. It is represented by light red translucent squares on D, and the square size is linearly related to s, as shown in Figure 5;

Step 8. Save the mapped D as a picture in JPG format, so we get a track diagram of the mouse's behavior in units of m basic operations.

Step 9. Repeat Steps 2-8 until the number of remaining operations in a session is less than m, and get n pictures of a user in a session, as shown in Figure 6.

According to the mapping method, all sessions of all users in "training_files" store them in the folders named by each user, and we get a dataset that can be trained by the CNN network. Each username is the label of each sub-dataset. We generated image sets for training CNN models in units of m=25, 50, 100, 500, and 1000, respectively, as shown in Figure 7.

3.1.3. Data Augmentation. Although the open-source dataset was used to generate 10 sets of tagged datasets, as shown in table x, the amount of generated images is insufficient. Therefore, we use three methods for data augmentation. One is to flip the image, including horizontal flip and vertical flip; The other is to rotate the image by 90 degrees, 180 degrees, and 270 degrees, respectively; The third is to randomly rotate the image by 25 degrees. Each picture will be judged according to the probability of 50% on whether to perform the above operation. As long as the dataset reaches our preset target, the dataset stops to augment. In this paper, our default augmentation goal is 18,000, which is to augment each user's dataset to 18,000 images.

Figure 4: Mouse "Scroll" operation is also often ignored by researchers, but it can also reflect people's habitual operating characteristics. The "Scroll" operation is divided into two Up and Down scroll actions.

Figure 5: Usually, researchers think of mouse movements and clicks as mouse operations. They do not pay attention to the interval between two mouse operations. In fact, this interval is also an "operation" of the mouse, and we call it the "Stay" operation. When there is no other operation on the mouse in one coordinate or the interval between two operations of the mouse, it can be defined as the "operation," which is represented by a semitransparent square on the two-dimensional image. The size of the square is used to indicate the length of stay.

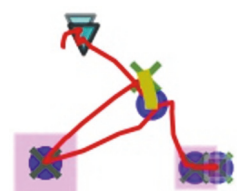
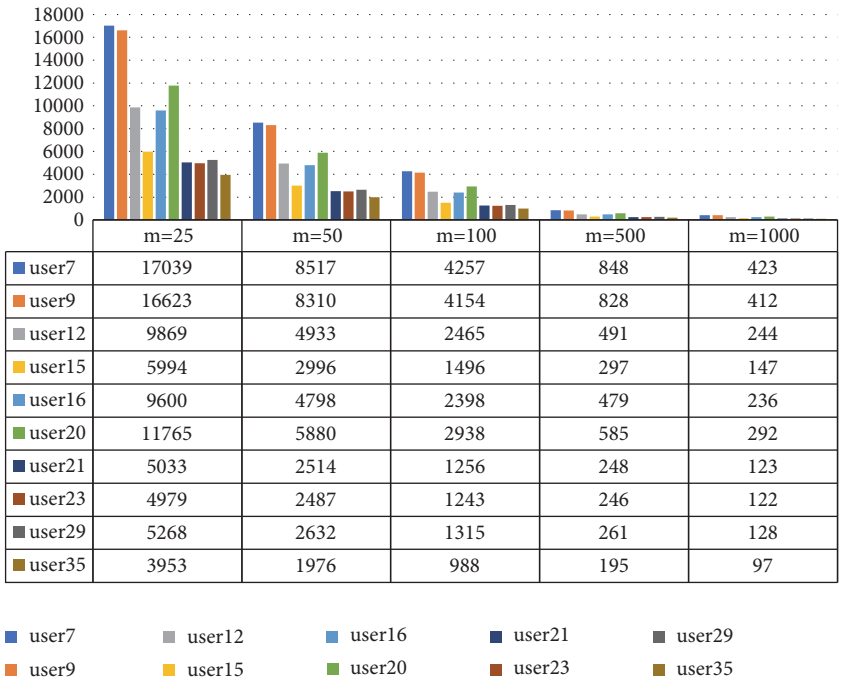


Figure 6: A picture of a user's mouse behavior (when m=100).

3.2. Overall Architecture of CNN. We refer to the networks of Alexnet [38], VGG [39], and GoogleNet [40]. We do not have such a large training set, so we do not choose to use such deep networks as VGG and GoogleNet. In fact, an 8-layer network constructed entirely in accordance with Alexnet parameters is not very applicable in our experiments. Therefore, we have constructed a 7-layer CNN network as shown in Figure 8.

The first four layers are convolution layers, and the remaining three are fully connected layers. A max-pooling layer follows each convolutional layer. The output of the first two full-connected layers is processed by Dropout. The output of the last layer is sent to Softmax and obtained the probability distribution of the classified labels. Apply the ReLU nonlinearity to the output of all convolutional and all fully connected layer except the last one.

Details of Parameters. The first convolutional layer filters the 100∗100∗3 input image with 32 kernels of size 3∗3∗3 with a stride of 1 pixel. The second convolutional layer takes as input the output of the first convolutional layer and filters it with 64 kernels of size 3∗3∗32. The third convolutional layer has 128 kernels of size 3∗3∗64 connected to the outputs of the second convolutional layer. The fourth convolutional layer has 128 kernels of size 3∗3∗128. The first fully connected layers have 1024 neurons and the second have 512, and both have passed the Dropout layer processing with a probability of 0.5.



	m=25	m=50	m=100	m=500	m=1000
user7	17039	8517	4257	848	423
user9	16623	8310	4154	828	412
user12	9869	4933	2465	491	244
user15	5994	2996	1496	297	147
user16	9600	4798	2398	479	236
user20	11765	5880	2938	585	292
user21	5033	2514	1256	248	123
user23	4979	2487	1243	246	122
user29	5268	2632	1315	261	128
user35	3953	1976	988	195	97

Figure 7: Image dataset generated from "training_files" in a unit of m. As the value of m increases, the dataset of user pictures becomes smaller.

(a) Activation Function. The CNN with Rectified Linear Units is much faster than the CNN with tanh [41], so we use the nonlinear activation function ReLU. The function form of ReLU is $f(x) = x^+ = \max(0, x)$. This nonlinear function means that when x is less than 0, the output is always 0, and when x is greater than 0, the output is the input value.

(b) LRN Layer. Although the concept of LRN (Local Response Normalisation) was mentioned for the first time in Alexnet and it also shows that LRN can reduce the error rate of its model, in [39] it is mentioned that LRN has little impact on the network and increased time-consuming. In our actual training, we also found that adding the LRN layer will increase the training time, but the improvement of training results is not very obvious. Therefore, the LRN layer is not added after convolution.

(c) Pooling Function. We used the max-pooling function [42], which is a commonly used pooling function in current CNN networks. The max-pooling can be expressed as taking the maximum value of this area within a certain rectangular area (z∗z) instead of the output of the network at this location and performing pooling every S pixels.

(d) Dropout. To prevent overfitting of the model, we added the Dropout layer to the first two layers fully connected layers. In machine learning, the output of multiple models is usually integrated to reduce the generalization error, such as Bagging [43], but when each model is a large neural network, the training and evaluation of such networks requires a lot of time and memory, as [40] integrates six neural networks, beyond which it will become difficult to handle. While Dropout provides an approximate Bagging integration method [44, 45], Dropout randomly drops some of the nodes' outputs with a certain probability, so that the dropped output does not participate in the propagation. Each round of training randomly drops some nodes, which is equivalent to a part of the network forming a subnetwork or submodel. This can effectively reduce the complex coadaptation between neurons; because of random inactivation, the neuron cannot be overly dependent on a previous neuron with an output. In training our model, we set the Dropout probability to 0.5.

(e) Gradient Optimization Algorithm. We used the Adam [45] optimization algorithm provided by Tensorflow, which is an optimization algorithm with learning rate adaptation and introduces quadratic gradient correction. The specific implementation algorithm in Tensorflow is as in Algorithm 1.

Where learning_rate is stepsize, which is the learning rate, Beta1 and beta2 are the exponential decay rates of first-order moment estimation and second-order moment estimation, respectively. Momentum in other optimization algorithms is directly incorporated into first-order moment estimation in Adam. The range of moment estimation is [0, 1); Epsilon is a small constant used for numerical stability. The default value of this constant is 1e-08, but it is necessary to test the best choice based on experiments. The variable is updated according to the gradient g. In our model training, we set learning_rate=0.01; beta1=0.9; beta2=0.999; epsilon=1e-08.

4. Experiments and Results

In this section, we will conduct three experiments. We completed the entire model training and experiments in

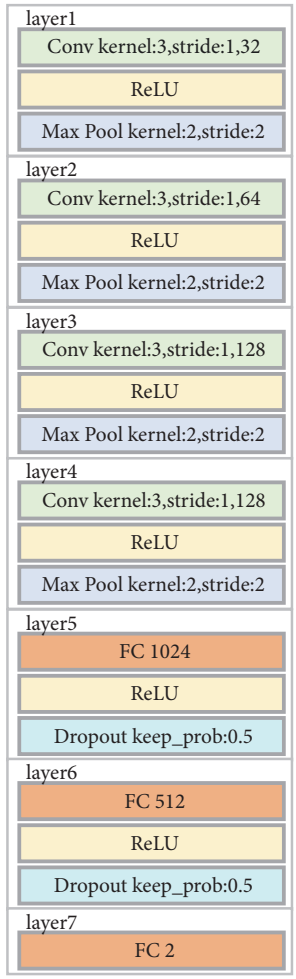


Figure 8: The architecture of our CNN.

the following experimental environment: Python 3.5.2, Tensorflow r1.4, CUDA Version 8.0.61, cudnn-8.0-windows10-x64-v6.0, windows10, and NVIDIA GTX 1060 6GB GPU. Experiment A is to verify the effectiveness of our method. Through the experiment, we can confirm that the use of the CNN network is effective for identity authentication based on mouse features and achieves good FAR and FRR values. Experiment B is to illustrate that our method requires very little time for authentication and can perform continuous user identity authentication after the user logs in to the terminal. Experiment C is to experiment with the "test-files" data provided by the dataset. Our experiment is designed to be a scene that needs to be faced with a real insider threat attack and takes measures to reduce FAR as much as possible. Experiments can show that our method can be applied in practical situations.

4.1. Experiment A: Identity Authentication. We believe that, in the insider threat detection scenario, the problem to be solved by the identity authentication should be judging whether the person currently using the terminal is consistent with the currently logged-in user. Therefore, we designate one user as a legal user and nine other users as illegal users (we total

```
t <- t + 1
lr_t <- learning_rate * sqrt(1 - beta2^t) / (1 - beta1^t)
m_t <- beta1 * m_{t-1} + (1 - beta1) * g
v_t <- beta2 * v_{t-1} + (1 - beta2) * g * g
variable <- variable - lr_t * m_t / (sqrt(v_t) + epsilon)
```

Algorithm 1: Adam optimization algorithm.

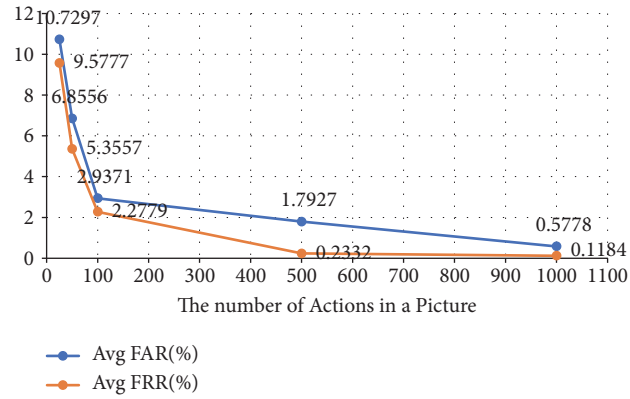


Figure 9: Trend chart of average values of FAR and FRR.

have ten users' mouse dynamic data). This is a typical binary classification problem. To verify the effect of our method, we design the experiment as follows.

Step 1. Use the image dataset that was generated in Section 3.1.2. Appoint a legal user (such as user 12), and extend the subdataset D as described in Section 3.1.3. Then, divide the dataset D into a training set T0 and a test set T0′ according to the ratio of 85% and 15%;

Step 2. According to the T0 and T0′, randomly extract the same amount from the other nine users' subsets, to construct an illegal user training set T1 and an illegal test set T1′. And ensure that there is no intersection between T1 and T1′.

Step 3. Take T0 and T1 as input, and use the CNN network constructed in Section 3.2 to train the model. Then, test T0′ and T1′ with the generated model and calculate FAR and FRR.

Step 4. Appoint one of the ten users as the legal user, and the remaining nine are considered as illegal users. Repeat the above experiment steps and calculate the average FAR and FRR.

In this experiment, we made T0 + T0′ = 18000 and T1 + T1′ = 18000. Hence, the size of training set is T0 + T1 = 30600(85%), and the size of test set is T0′ + T1′ = 5400(15%). The above experiment was conducted for different mouse operation datasets (m=25, 50, 100, 500, or 1000), and the experimental results were shown in Table 1. Figure 9 shows the average values of FAR and FRR vary with m, and they decrease as m increases (the number of features on each image increases).

Table 1: The results of experiment A.

User	m=25		m=50		m=100		m=500		m=1000	
	FAR(%)	FRR(%)	FAR(%)	FRR(%)	FAR(%)	FRR(%)	FAR(%)	FRR(%)	FAR(%)	FRR(%)
7	16.556	10.333	7.37	6.37	4.889	2.444	4.519	0.185	1.778	0.259
9	9.519	10.815	6.222	3.889	2.704	2.556	2.444	0.037	0.519	0.037
12	18.37	12.259	8.148	9.63	4.222	2.963	1.259	0.444	0.852	0.296
15	9.407	7.037	11.593	7.63	4.963	1	0.519	0.148	0	0.259
16	10.667	9.185	7.556	4.148	5.111	1.852	1.037	0.074	0.444	0
20	6.074	8.185	9.63	1.556	2.444	1.112	0.852	0.37	0.519	0.037
21	7.704	8.704	4.407	4	1	2.519	0.778	0.519	0.222	0.037
23	7.259	9.222	3	5.852	0.63	3.481	1.704	0.37	0.259	0
29	12.481	10.333	6.963	5.593	1.704	2.667	1.037	0.148	0.481	0.185
35	9.26	9.704	3.667	4.889	1.704	2.185	3.778	0.037	0.704	0.074
Avg	*10.7297*	*9.5777*	*6.8556*	*5.3557*	*2.9371*	*2.2779*	*1.7927*	*0.2332*	*0.5778*	*0.1184*

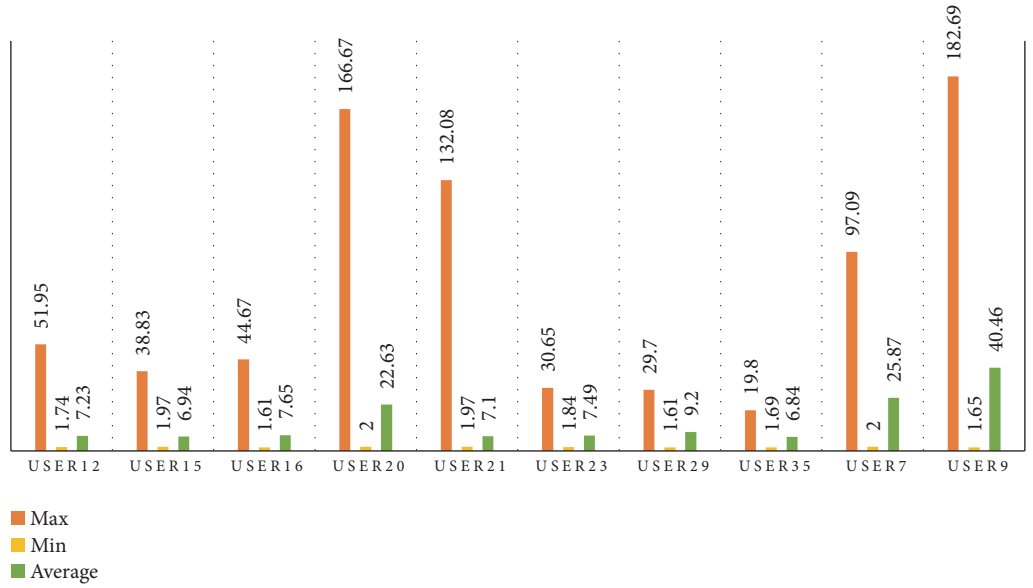


Figure 10: Number of mouse actions per second generated by users.

4.2. Experiment B: The Time Cost of Identity Authentication. In our opinion, the authentication time is composed of the time needed to collect the mouse features and the time required for classification. Compared with the time of collecting mouse features, the time of classifying mouse features using the trained model is almost negligible. Hence, our primary concern is the time required to obtain enough features. It can be seen from the data set analysis that the number of operations per second generated by the user when using the mouse normally has individual differences. The detailed data is shown in Figure 10.

As can be seen from Figure 10, the user (user 9) generated up to about 182 operation actions in one second. That is to say, in the fastest case, it only takes less than 1 second to complete 100 operations collection and authentication (when m=100). In the slowest case, the user (user 16 and user 29) only generates about 1.61 operations per second, which takes about 1 minute to complete the operation collection and authentication (when m=100). On average, ten users can generate 14.141 mouse operations per second, and about 7 seconds can complete acquisition and authentication (when m=100). The mouse operation data we use is the raw data of the mouse (as we mentioned in Section 3). We analyzed the average time and the minimum time, respectively, under different values of m (when m=25, 50,100,500,1000), the result shown in Table 2.

4.3. Experiment C: Insider Threat Detection. In an insider threat attack scenario, the insider is familiar with the internal system; it is possible to copy sensitive information in a very short time. Therefore, we believe that the authentication time should be within 10 seconds, and the time intervals between two authentications should also be controlled within 10 seconds. According to the above experimental results, we select the third model (m = 100) for the next experiment. The model was able to complete authentication in about 7 seconds

Table 2: Average time and the minimum time required to acquire mouse actions.

m	Avg Time(s)	Min Time(s)
25	1.768	0.315
50	3.536	0.63
100	7.072	1.26
500	35.358	6.296
1000	70.716	12.592

Table 3: The test data set generated according to the mapping method (when m=100), in which each user folder contains legal sessions and illegal sessions.

test_files	legal	illegal	total_picutres
user7	36	37	1659
user9	23	43	1585
user12	56	49	1306
user15	45	70	1238
user16	68	38	1173
user20	30	20	1089
user21	37	22	605
user23	38	33	765
user29	43	20	684
user35	35	73	1076
Total	411	405	11180

Note. During the generation process, we found that some sessions do not have label information in "public_labels.cvs". Because we could not confirm these sessions are legitimate data or not, we removed this data information.

on average and reached 2.94% FAR and 2.28% FRR. We think this can basically meet the needs of such insider threat attack detection.

We will use the test set ("test_files") provided by the dataset for the experiment and then determine whether a session is legal data based on the labels ("public_labels.csv") provided by the dataset. The user data in "test_files" is mapped according to the mapping method in Section 3.1.2 to generate a test data set. The results are shown in Table 3.

It would be fair to say that an insider threat detection system with low false reject rates may be tolerated, but an insider threat detection system with low false accept rates cannot be allowed. That is because if there is a false reject event, the results of the false reject event report can be assisted by various measures, such as on-site inspection, video surveillance, IDS, firewall, and SOC, and the false reject event can be verified. But if a malicious behavior of espionage is not be detected, that means the attacker has successfully achieved the goal and will incur an incalculable loss to the organization. That is to say, the system designing is to minimize FAR but with FRR-tolerant in the actual application scenario.

Therefore, we design this experiment according to the purpose of reducing the FAR as much as possible. A session represents the beginning of a mouse session, in which mouse actions are generated in chronological order. In the actual authentication scenario, authentication is performed every time sufficient actions are generated (we generate pictures according to the setting of m=100). We do not consider the current user to be a legitimate user until three consecutive authentications are legal. In other words, each authentication result is compared with the previous two authentication results, and a warning is issued as long as one of the authentication results is illegal. In this way, the actual authentication requires three pictures (m=100∗3), which can effectively reduce the FAR. The experimental results are shown in Table 4.

Table 4: The results of experiment C.

User	FAR(%)	FRR(%)
7	0	3.223
9	0	2.365
12	2.5	7.537
15	0	3.704
16	0	12.776
20	0	12.296
21	0	23.958
23	0	22.52
29	0	11.556
35	0	15.73
Average	0.25	11.5665

4.4. Comparison with Previous Work. In this section, we show a comparison of our experimental results against the results of previous works, which are shown in Table 5. The verification time is highly dependent on the number of mouse actions. As described in Section 3.1.2, the type of mouse actions can be divided into Click, Move, Drag, Scroll, and Stay. We choose to use all these five raw mouse actions to construct the authentication model, but the previous works do not use all the basic actions and try to extract features from the basic actions. Reference [28] uses move, drag, click, and stay for authentication, but they need about 2000 mouse actions and 1033 seconds. Reference [35] needs 20 mouse clicks, but sometimes they need a very long time to collect enough clicks (average of 37.73 minutes). If they also use mouse move actions, the verification time can reduce to 3.03 minutes. Reference [36] chooses to use mouse click and mouse movement to construct the features.

Generally speaking, as the number of mouse actions increases, both FAR and FRR show a downward trend, but the verification time increases accordingly.

5. Conclusion

Many previous studies have shown that mouse biobehavioral features can authenticate users. In this paper, we focus on the challenges of using mouse behavioral features for insider threat detection and propose a method that combines deep learning with mouse biobehavioral features for insider threat detection. This method can complete a user authentication task in a very short time while maintaining high accuracy. In the previous studies, one or several basic actions were selected

Table 5: Compare with previous works.

Source	FAR	FRR	Data required	Authentication time
[28]	2.46%	2.46%	2000 mouse actions (click/move/drag/stay)	1033 seconds
[35]	1.30%	1.30%	20 mouse clicks (click or click/move)	37.73 minutes(click) or 3.03 minutes(click/move)
[36]	8.74%	7.69%	32 mouse operations (click/move)	11.80 seconds
	4.69%	4.46%	160 mouse operations (click/move)	59.49 seconds
	3.33%	2.12%	320 mouse operations (click/move)	118.14 seconds
ours	10.73%	9.58%	25 mouse actions (click/move/drag/stay/scroll)	1.768 seconds
	6.86%	5.36%	50 mouse actions (click/move/drag/stay/scroll)	3.536 seconds
	2.94%	2.28%	100 mouse actions (click/move/drag/stay/scroll)	7.072 seconds
	1.79%	0.23%	500 mouse actions (click/move/drag/stay/scroll)	35.358 seconds
	0.58%	0.12%	1000 mouse actions (click/move/drag/stay/scroll)	70.716 seconds

from mouse five basic actions, and these actions were used to extract features to describe the unique behavioral characteristics of the user and then classified by using methods such as SVM, to realize user authentication. We use all five basic mouse actions to prevent the user's unique behavior characteristics from being omitted. Then, we map the user's mouse actions into pictures and automatically extract and model the user behavior pictures through the CNN network of deep learning. We use an open-source mouse behavior dataset that contains mouse action data from 10 users. The experiments have demonstrated the effectiveness of the proposed approach, with a false acceptance rate of 2.94%, a false rejection rate of 2.28%, and the authentication time of 7.072 seconds (when m = 100). These results show that this approach can be applied to detect insider threat attacks in specific scenarios.

Data Availability

The mouse dynamic data used to support the findings of this study were supplied by Balabit Mouse Dynamics Challenge dataset and available at https://github.com/balabit/Mouse-DynamicsChallenge.

Conflicts of Interest

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

Acknowledgments

This work was supported in part by Defense Industrial Technology Development Program JCKY2018212C020 and JCKY2016212C005 and in part by the National Natural Science Foundation (NSFC) under Grant CNS 61572115.

References

[1] Q. Yaseen and B. Panda, "Insider threat mitigation: preventing unauthorized knowledge acquisition," *International Journal of Information Security*, vol. 11, no. 4, pp. 269–280, 2012.

[2] R. Trzeciak and D. Costa, *Model-Driven Insider Threat Control*, Carnegie Mellon University, 2017, https://resources.sei.cmu.edu/asset_files/Presentation/2017_017_001_509187.pdf.

[3] B. Bose, B. Avasarala, S. Tirthapura, Y.-Y. Chung, and D. Steiner, "Detecting insider threats using RADISH: a system for real-time anomaly detection in heterogeneous data streams," *IEEE Systems Journal*, vol. 11, no. 2, pp. 471–482, 2017.

[4] D. Liu, X. Wang, and J. Camp, "Game-theoretic modeling and analysis of insider threats," *International Journal of Critical Infrastructure Protection*, vol. 1, pp. 75–80, 2008.

[5] http://www.cert.org/insider-threat/index.cfm.

[6] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, "Insight into insiders: A survey of insider threat taxonomies, analysis, modeling, and countermeasures," 2018, https://arxiv.org/abs/1805.01612.

[7] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, "Detecting and preventing cyber insider threats: a survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1397–1418, 2018.

[8] M. Collins, *Common Sense Guide to Mitigating Insider Threats*, Carnegie Mellon University, Pittsburgh, PA, USA, 2016.

[9] S. L. Pfleeger, J. B. Predd, J. Hunker, and C. Bulford, "Insiders behaving badly: addressing bad actors and their actions," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 1, pp. 169–179, 2010.

[10] M. Watkins and K. Wallace, *Understanding Network Security Principles*, Chapter 1, NetworkWorld, 2009, https://www.networkworld.com/article/2268110/lan-wan/chapter-1–understanding-network-security-principles.html.

[11] S. Eberz, K. Rasmussen, V. Lenders, and I. Martinovic, *Preventing Lunchtime Attacks: Fighting Insider Threats with Eye Movement Biometrics*, 2015.

[12] M. Pusara and C. E. Brodley, "User re-authentication via mouse movements," in *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pp. 1–8, ACM, USA, October 2004.

[13] X. J. Chen, F. Xu, R. Xu, S. M. Yiu, and J. Q. Shi, "A practical real-time authentication system with Identity Tracking based on mouse dynamics," in *Proceedings of the Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference*, pp. 121-122, IEEE, 2014.

[14] L. A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Richardson, *CSI/FBI Computer Crime and Security Survey*, 2006.

[15] Kroll and Economist Intelligence Unit, *Annual Global Fraud Survey, 2011/2012*, 2012.

[16] *2014 US State of Cybercrime Survey*, US CERT, Camegie Mel-lon University, 2014.

[17] *Credit Card Details on 20 Million South Koreans Stolen*, BBC News, 2014, https://www.Bbc.Com/News/Technology-25808189.

[18] *Former Engineer Sentenced to Prison for Destroying Company's Computer System*, WSAZ News, 2014, http://www.wsaz.com/home/headlines/Former-Engineer-Sentenced-to-Prison-for-Destroying-Companys-Computer-System-259992681.html.

[19] *2016 State of Cybercrime Survey*, 2016, https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=499782.

[20] *2017 U.S. State of Cybercrime*, CSO, 2017, https://images.idgesg.net/assets/2017/08/idg_presentation_cybercrime_07202017_final_compressed.pdf.

[21] *5 Best Practices for Preventing and Responding to Insider ... (n.d.)*, 2017, https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=509914.

[22] *The CERT Guide to Insider Threats*, US CERT, 2012, http://www.cert.org/insider-threat/.

[23] *Anomaly Detection at Multiple Scales (ADAMS) Broad Agency Announcement DARPA-BAA-11-04 (PDF)*, General Services Administration, 2011.

[24] E. Ted, H. G. Goldberg, A. Memory et al., "Detecting insider threats in a real corporate database of computer usage activity," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery And Data Mining*, pp. 1393–1401, ACM, August, 2013.

[25] A. Weiss, A. Ramapanicker, P. Shah, S. Noble, and L. Immohr, "Mouse movements biometric identification: a feasibility study," in *Proceedings of Student/Faculty Research Day*, CSIS, Pace University, White Plains, New York, NY, USA, 2007.

[26] K. Killourhy and R. Maxion, "Why did my detector do that?!," in *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*, pp. 256–276, Springer, Berlin, Heidelberg, 2010.

[27] A. Messerman, T. Mustafić, S. A. Camtepe, and S. Albayrak, "Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics," in *Proceedings of the Biometrics (IJCB), 2011 International Joint Conference*, pp. 1–8, IEEE, USA, October 2011.

[28] A. A. E. Ahmed and I. Traore, "A new biometric technology based on mouse dynamics," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 3, pp. 165–179, 2007.

[29] H. Gamboa and A. Fred, "A behavioral biometric system based on human-computer interaction," in *Biometric Technology for Human Identification*, vol. 5404, pp. 381–393, International Society for Optics and Photonics, August, 2004.

[30] Z. Jorgensen and T. Yu, "On mouse dynamics as a behavioral biometric for authentication," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pp. 476–482, ACM, March, 2011.

[31] V. Mavroeidis, K. Vishi, and A. Jøsang, "A framework for data-driven physical security and insider threat detection," in *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 1108–1115, IEEE, August, 2018.

[32] W. Park, Y. You, and K. Lee, "Detecting potential insider threat: analyzing insiders' sentiment exposed in social media," *Security and Communication Networks*, vol. 2018, Article ID 7243296, 8 pages, 2018.

[33] M. Bin Ahmad, A. Akram, M. Asif, and S. Ur-Rehman, "Using genetic algorithm to minimize false alarms in insider threats detection of information misuse in windows environment," *Mathematical Problems in Engineering*, vol. 2014, Article ID 179109, 12 pages, 2014.

[34] O. Lo, W. J. Buchanan, P. Griffiths, and R. Macfarlane, "Distance measurement methods for improved insider threat detection," *Security and Communication Networks*, vol. 2018, Article ID 5906368, 18 pages, 2018.

[35] N. Zheng, A. Paloski, and H. Wang, "An efficient user verification system via mouse movements," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pp. 139–150, ACM, October 2011.

[36] C. Shen, Z. Cai, X. Guan, Y. Du, and R. A. Maxion, "User authentication through mouse dynamics," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 16–30, 2013.

[37] Á. Fülöp, L. Kovács, T. Kurics, and E. Windhager-Pokol, *Balabit Mouse Dynamics Challenge Data Set*, 2016, https://github.com/balabit/Mouse-Dynamics-Challenge.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[39] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2014, https://arxiv.org/abs/1409.1556.

[40] C. Szegedy, W. Liu, Y. Jia et al., *Going Deeper with Convolutions*, 2014, https://arxiv.org/abs/1409.4842.

[41] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann," in *Proceedings of the 27th International Conference on Machine Learning (ICML '10)*, pp. 807–814, Haifa, Israel, June 2010.

[42] Y. T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins, "Image restoration using a neural network," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 7, pp. 1141–1151, 1988.

[43] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[45] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2014.