Journal of Network Intelligence ©2024 ISSN 2414-8105
Taiwan Ubiquitous Information Volume **9**, Number **2**, May **2024**

Malware Detection Based on Optimized Deep Learning in Data-driven Mode

Yong Zhao*
Si Chuan Top IT Vocational Institute, Chengdu 611743, China
31808021@qq.com

Yu Liu
North Bangkok University, Bangkok 10220, Thailand
76695021@qq.com

*Corresponding author: Yong Zhao
Received January 16, 2024, revised March 3, 2024, accepted April 2, 2024.

Abstract. *This article mainly explores how to use optimized deep learning techniques for malware detection in data-driven mode. A deep learning model was designed, which combines the characteristics of Convolutional Neural Networks (CNN) and can effectively capture the complex features of malicious software. In order to further improve the performance of the model, optimization techniques were used and a CNN deep learning algorithm based on learning rate adjustment Adam was proposed. Adjust the learning rate and momentum decay rate. The learning rate controls the update ratio of weights, while the momentum decay rate controls the exponential weighted average decay rate of momentum. RNN were used to model the sequence of extracted image features to capture temporal dependencies between malicious software. During the experimental phase, large-scale actual malware samples are used for training and validation. The results indicate that the proposed optimized deep learning model has achieved significant improvements in malware detection compared to traditional methods. This model performs well in accuracy, recall, and precision, and has stronger generalization ability and robustness.The CNN+RNN based on Adam algorithm proposed in this article achieves an accuracy of 95.8%.*
Keywords: CNN; Deep learning model; Learning rate; Adam; Accuracy; RNN

1. Introduction.

With the rapid development and popularization of the Internet, network security problems are becoming more and more serious. Relevant reports state that in 2021, there will be 38 million devices infected with malware worldwide, accounting for 0.68% of all mobile devices, a year-on-year increase of 25%, higher than 20% in 2017. Malware poses significant risks to individual users, corporate organizations, and even national security. Some harms are even irreversible. For example, malware will search the contents of confidential computers or servers, thereby leaking confidential information and causing significant economic losses [1]. The general definition of malware is: programs that can destroy software and hardware equipment such as workstations, mobile devices (such as mobile phones, etc.), servers, and gateways. Common malware programs include viruses, worms, Trojans, ransomware, zombies, etc. Malware detection is of great significance, as it can improve the security and performance of general software, discover potential loopholes and defects in software, and thereby ensure user safety and efficiency.

This article analyzes the serious threats posed by malicious software to computer and network security. Deep learning technology is widely used in data-driven patterns to effectively detect and prevent malicious software. The author proposes a malicious software detection method based on optimized deep learning to improve the accuracy and efficiency of detection. The research scope mainly focuses on the intersection of deep learning and malware detection. Efficient malware detection using Convolutional Neural Networks (CNN) [2] and Recurrent Neural Networks (RNN) models [3] in deep learning. The research scope includes model optimization, design of new network structures, improvement of training algorithms, data expansion and cleaning. This paper proposes a CNN+RNN deep learning algorithm based on Adam for malware detection. This algorithm combines the advantages of CNN and RNN to accurately detect malicious software by extracting its features and learning its behavior patterns. By applying RNN to the behavior sequence of malicious software, it is possible to learn the dynamic characteristics and behavior patterns of malicious software. The significance of this study lies in improving the efficiency and accuracy of malware detection.

The potential outcomes are shown as follows, by optimizing deep learning models and algorithms, the accuracy and efficiency of detection can be improved, and the rates of false positives and false negatives can be reduced. Deep learning models can automatically extract useful features from data, reduce reliance on artificial feature engineering, and improve the efficiency and accuracy of feature selection [4, 5]. The optimized deep learning model can quickly process and classify new malware samples, providing real-time security protection. Reduce reliance on manual intervention and reduce the likelihood of human errors.

Optimization techniques are introduced, including but not limited to learning rate adjustment, weight regularization, and batch normalization.In order to further improve the performance of the model, optimization techniques were introduced and a CNN deep learning algorithm based on learning rate adjustment Adam was proposed. During the experimental phase, large-scale actual malware samples are used for training and validation, and using RNN to model the sequence of extracted image features to capture temporal dependencies between malicious software.

This research provides an effective malware detection methodthat combines deep learning and optimization techniques. By adapting to the evolving threat landscape of malware through this combination, the proposed model offers robust support for enhancing computer network security.

1.1. **Related Work.**

The following is a summary of the literature review for the suggested procedure as presented in this portion of the paper.

Heuristic and behavioral analysis techniques focus on identifying malware based on suspicious behaviors or deviations from normal software behavior. While more adaptive than signature-based methods, they may produce false positives or miss sophisticated threats.

Edge and Sampaio [6] did a survey of signature-based methods for financial fraud detection. Weng et al. [7] provided a signature-based composition technique. Wohlfahrt et al. [8] developed the bacteria-signature based method. Traditional antivirus solutions primarily rely on signature-based methods, which involve maintaining a database of known malware signatures. While effective against known threats, these methods struggle with detecting new and evolving malware variants.The purpose of research on malware detection based on optimized deep learning in data-driven mode is to improve the efficiency and accuracy of malware detection, in order to cope with the increasingly rampant network attacks and malware threats. The deep learning model can be trained to adapt to new malware variants. Once the model is trained, it can be easily applied to new samples without manually adjusting or updating features. This flexibility enables the deep learning model to better adapt to the continuous evolution and variation of malware.

Deep learning, especially neural networks like Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), has shown promising results in capturing intricate patterns and features inherent in malware. These models excel at feature learning, alleviating the need for manual feature engineering. Gopinath, and Sethuraman [9] did a comprehensive survey on deep learning-based malware detection techniques. Tayyab [10] did a survey of the recent trends in deep learning-based malware detection. Shaukat et al. [11] provided a novel deep learning-based approach for malware detection. Brown et al. [12] studied automated machine learning for deep learning-based malware detection. Alomari et al. [13] studied malware detection using deep learning and correlation-based feature selection. Kim et al. [14] developed a practical deep learning-based android malware detection system. Chaganti et al. [15] studied deep learning based cross architecture internet of things malware detection and classification. Xing et al. [16] provided a malware detection approach using autoencoder in deep learning. Ravi et al. [17] developed a multi-view attention-based deep learning framework for malware detection in smart healthcare systems. Majid et al. [18] did a review of artificial intelligence-based malware detection using deep learning.

Asam et al. [19] developed a malware detection architecture using a novel channel boosted and squeezed CNN. Khan et al. [20] developed a new deep boosted CNN and ensemble learning based on malware detection. Akhtar et al. [21] studied detection of malware by deep learning as CNN-LSTM machine learning techniques in real time. Arslan and Tasyurek [22] studied android malware detection via feature graph and convolutional neural networks. Naeem et al. [23] studied explainable artificial intelligence-based IoT device malware detection mechanism using image visualization and fine-tuned CNN-based transfer learning. Yeboah and Musah. [24] studied NLP technique for malware detection using 1D CNN fusion model. Ullah et al. [25] studied network-based android malware detection system using transfer learning and CNN-BiGRU ensemble. However, they did not achieve good results in terms of accuracy. Lakshmanarao and Shashi [26] studied android malware detection with deep learning using RNN from opcode sequences. Almaleh et al. [27] studied malware API calls detection using hybrid logistic regression and RNN model. Djenna et al. [28] made a review of artificial intelligence-based malware detection using deep learning. Prabhavathy et al. [29] made a novel approach for detecting online malware detection LSTMRNN and GRU based recurrent neural network in cloud environment. So, we attempted to apply the fusion of RNN and CNN to the algorithm proposed in this paper.

Transfer learning has been used in this paper to address the challenge of limited labeled data. Pre-trained models on large datasets can be fine-tuned for malware detection tasks, enhancing the generalization of the model. Kumar and Janet [30] studied deep transfer learning for malware image classification. DE García made an effectiveness analysis of transfer learning for the concept drift problem in malware detection. García et al. [31] proposed an explainable malware detection system using transformers-based transfer learning and multi-model visual representation.

1.2. **Motivation and contribution.** Traditional malware detection methods are often based on feature engineering, which extracts specific static features from software for classification. However, this method is difficult to deal with constantly changing malware samples. Deep learning, especially CNN and RNN, can achieve more accurate classification by learning complex patterns in data. With the increase of network attacks, the number of malware samples that need to be detected is also rapidly increasing. The main contents of this paper are as follows:

(1) Aiming at the problem that the convergence speed of traditional CNN is not ideal, Adam (Adaptive Moment Estimation) optimization algorithm is proposed to update the parameters of CNN. The parameters are updated by calculating the first moment estimate (mean) and the second moment estimate (variance) of the gradient. It can effectively adjust the learning rate, making the updating of different parameters more accurate and efficient in the training process.

(2) Malware detection usually involves binary file sequences, and these sequence data have certain spatial local structure and time dependence. Therefore, CNN+RNN sequence modeling is proposed to realize malware detection. CNN is used to extract local features of sequence data, and then these feature sequences are input into RNN for modeling. CNN can effectively capture the spatial local structure and characteristics of the input sequence, while RNN can model the time dependence of the sequence.

2. Relevant theoretical analysis.

2.1. Deep learning theory. In malware detection, deep learning can automatically extract features of malware and classify and recognize them based on these features. Neural networks simulate the way neurons in the human brain connect, using multiple neurons, each with a weight, to convert input signals into output signals. Neural networks learn the relationship between input and output by continuously adjusting weights, thereby achieving functions such as classification and prediction. The activation function is an important component of neural networks, used to convert the input signal of neurons into an output signal. This model uses the Sigmoid function as the activation function, and the selection and design of the activation function have a significant impact on the performance of the neural network.The core of deep learning theory is backpropagation algorithm, which calculates the error between the output layer and the target value, and then adjusts the weight of each neuron in reverse to make the output result of the entire network closer to the target value. This algorithm can automatically learn features from input data, thus avoiding the tedious process of manually designing features. Deep learning theory also involves some important concepts, such as activation functions, loss functions, optimizers, etc. The activation function is used to map the input of a neuron to the output, and commonly used activation functions include ReLU, sigmoid, and tanh. The loss function is used to measure the difference between the predicted and actual results of a model, and commonly used loss functions include mean square error (MSE) and cross entropy loss. Optimizers are used to update the weights and biases of models, and commonly used optimizers include gradient descent, random gradient descent, Adam, etc.

2.2. Optimization Algorithm Theory. Optimization algorithms are a type of algorithm used to find the optimal solution, which minimizes or maximizes a certain objective function by continuously iterating and adjusting parameters. In malware detection, optimization algorithms can be used to optimize the parameters of deep learning models to improve their performance and generalization ability. This model uses the Gradient Descent method, which is an iterative optimization algorithm used to find the parameter values that minimize the cost function. In machine learning and deep learning, gradient descent is widely used to optimize loss functions and find the optimal model parameters. Firstly, it is necessary to calculate the gradient of the loss function with respect to the model parameters, which represents the direction and magnitude of the change in the value of the loss function when the parameters change. Then, based on the gradient information, update the model parameters with a certain step size (learning rate) to gradually approach the minimum value of the loss function. Optimization algorithm theory involves how to find the optimal solution or approximate optimal solution of a problem. Optimization algorithms are commonly used to handle complex problems in fields such as machine learning and deep learning, helping models find the optimal parameter configuration during the training process to minimize the loss function or maximize the objective function.

Convolutional operation is a core operation in CNN, used to extract features from input data. The formula is:

$$Y = f\left(W^{*} \cdot X + b\right) \tag{1}$$

Among them, Y represents the output value, W represents the weight matrix, X represents the input value, b represents the offset, and f represents the activation function.

Activation function:

The commonly used activation functions for CNN include Sigmoid, Tanh, and ReLU. The formulas for these functions are as follows:

Sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Sigmoid function is used to enhance the non-linear expression ability of the model, enabling it to better learn and fit data [32].

Optimization algorithms can be classified into various types:

(1) Gradient descent method: This is one of the most commonly used optimization algorithms in machine learning and deep learning. It gradually finds the minimum value of the loss function by calculating the gradient of the loss function and updating the model parameters along the opposite direction of the gradient. Common gradient descent methods include Batch Gradient Descent, Stochastic Gradient Descent, and Mini Batch Gradient Descent.

(2) Newton's method and quasi Newton's method: These methods use second-order derivatives (Hessian matrix) to accelerate the optimization process. The Newton method finds the update direction by calculating the second derivative of the loss function and solving a system of linear equations. The quasi-Newton's law approximates the second derivative in an iterative manner to reduce computational complexity. Conjugate Gradient Method: This method combines the advantages of gradient descent and Newton's method, accelerating the optimization process by constructing a set of conjugate directions. It performs well in dealing with large-scale problems.

(3) Heuristic optimization algorithms: These types of algorithms do not rely on gradient information [33], but instead seek the optimal solution by simulating natural processes such as genetic algorithms, simulated annealing, particle swarm optimization, etc. They are very effective in handling non convex, discrete, or constrained optimization problems. We can see from Figure 1.

3. **Basic CNN+RNN model implementation strategy.**

In this article, we employed a Convolutional Neural Network (CNN) model for malware detection. Firstly, we use CNN to extract features from the binary files of Italian software and extract image features from them. Then, we use RNN to model the sequence of extracted image features to capture the temporal dependencies between malicious software. During the model training process, we used gradient descent method for optimization. We adopted the Adam optimization algorithm, which is an adaptive learning rate optimization algorithm that can automatically adjust the learning rate during the training process to accelerate the convergence of the model. At the same time, we also used a learning rate decay strategy,

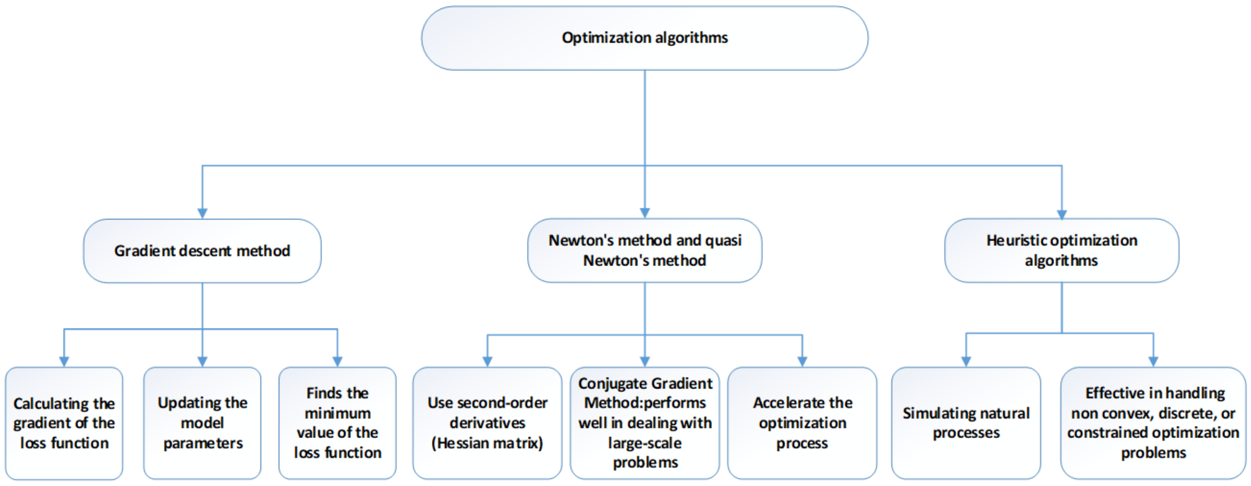
Optimization algorithms
Gradient descent method
Newton's method and quasi Newton's method
Heuristic optimization algorithms
Calculating the gradient of the loss function
Updating the model parameters
Finds the minimum value of the loss function
Use second-order derivatives (Hessian matrix)
Conjugate Gradient Method:performs well in dealing with large-scale problems
Accelerate the optimization process
Simulating natural processes
Effective in handling non convex, discrete, or constrained optimization problems

Figure 1. Optimization algorithms

which gradually reduces the learning rate during the training process to avoid the model oscillating near the local optimal solution. During the model training process, we also used data augmentation techniques to rotate, translate, scale, and other operations on the original data to enhance the model's generalization ability. In addition, we also used early stopping to prevent model overfitting, which means stopping training when performance on the validation set stops improving to avoid overfitting on the training set.

In this study, we employed the Adaptive Momentum Estimation (Adam) optimization algorithm to train deep learning models. Adam combines the advantages of adaptive gradient algorithm (AdaGrad) and root mean square propagation (RMSProp) to adjust the learning rate of each parameter by calculating the first-order and second-order moment estimates of the gradient.

The optimization objective is to minimize the loss function of the model on the training set, that is, the difference between the predicted label and the actual label. The optimization process includes calculating loss through forward propagation, gradient through backward propagation, and updating model parameters based on Adam algorithm. Through iterative training, the model gradually adapts to the data distribution and exhibits good generalization ability on the validation set.

In data-driven mode, we first preprocess the raw data, including steps such as cleaning, encoding, and normalization. Then, divide the data into training set, testing set, and validation set. During the training process, we use the training set to update the model parameters and the validation set to monitor the performance of the model and prevent overfitting. In the testing phase, we use the test set to evaluate the performance of the model.

The data set used in this study comes from several public malware sample libraries, including hundreds of thousands of malware samples and benign software samples. The data set covers a variety of malware types, attack methods and operating system platforms, ensuring the diversity and representativeness of the data.

In terms of experimental setup, we divided the data set into training set, test set and verification set according to the ratio of 8:1:1. The training set is used to train the model, the test set is used to evaluate the performance of the model, and the verification set is used to adjust the super parameters and monitor the over fitting.

Step 1 Data collection: Collect a large number of malicious software samples and normal software samples for training and testing models. Collect malware samples from multiple sources, including publicly available malware libraries, security laboratories, security companies, etc. At the same time, normal software samples are also collected to ensure the generalization ability of the model.

Step 2 Data preprocessing: Preprocessing the collected data, including noise removal, normalization, and other operations. After collecting samples, perform data filtering to remove duplicate, invalid, or incorrect samples. Classify the samples and store malicious software and normal software in different datasets. Preprocessing includes operations such as data cleaning, format conversion, and feature extraction. For binary files, convert them to image or text format for model processing. For malware samples, we need to annotate them to indicate their malicious nature. We adopted a semi supervised learning strategy based on deep learning, using a small amount of annotated data to train the model and improve its generalization ability.

Step 3 Model construction: Build a deep learning model and a CNN model based on Adam optimization algorithm.

Step 4 Model training: Utilize collected data for model training, adjust model parameters through optimization algorithms, and improve model detection accuracy and efficiency.

Step 5 Model evaluation: Evaluate the trained model, including accuracy, recall, and other indicators. In terms of model evaluation, we used metrics such as accuracy, recall, and F1 value to evaluate the performance of the model. At the same time, we also conducted cross validation experiments to verify the stability and generalization ability of the model.

Through the above methods, we have successfully constructed a malware detection model based on data-driven and optimized deep learning, and achieved high detection accuracy and efficiency. Figure 2 shows the process of data processing.

4. **Malicious software detection based on improved CNN+RNN.**

4.1. **CNN+RNN Deep Learning Algorithm Based on Adam.**

The malware detection method based on Adam's CNN+RNN deep learning algorithm can combine CNN and RNN for malware classification and detection.

The CNN+RNN deep learning algorithm based on Adam can be used for malware detection. This algorithm combines the advantages of CNN and RNN to accurately detect malicious software by extracting its features and learning its behavior patterns.

In the CNN section, convolutional layers can be used to extract image or text features from malicious software. Convolutional layers can automatically learn local features of input data and perform feature selection and mapping through pooling layers and activation functions. By stacking multiple convolutional layers, higher-level feature representations can be gradually abstracted.

In the RNN section, the ability of recurrent neural networks to process sequence data can be utilized to model the behavior patterns of malicious software. RNN processes input sequences through memory state, captures temporal information, and is able to capture dependencies between sequences. By applying RNN to the behavior sequence of malicious software, dynamic features and behavior patterns of malicious software can be learned.

The optimizer based on Adam can dynamically adjust the learning rate during the training process, enabling the algorithm to better converge to the optimal solution. The Adam optimizer combines the advantages of both Momentum and RMSprop optimization algorithms, updating weights and biases by calculating the exponential decay average of

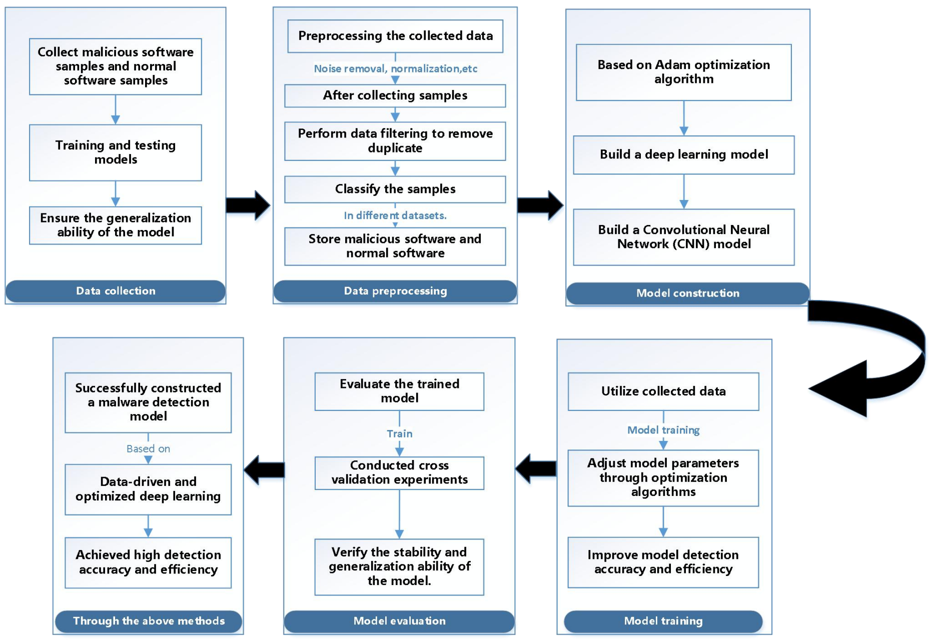
Collect malicious software samples and normal software samples
Training and testing models
Ensure the generalization ability of the model
Data collection
Preprocessing the collected data
Noise removal, normalization,etc
After collecting samples
Perform data filtering to remove duplicate
Classify the samples
In different datasets.
Store malicious software and normal software
Data preprocessing
Based on Adam optimization algorithm
Build a deep learning model
Build a Convolutional Neural Network (CNN) model
Model construction
Utilize collected data
Model training
Adjust model parameters through optimization algorithms
Improve model detection accuracy and efficiency
Model training
Evaluate the trained model
Train
Conducted cross validation experiments
Verify the stability and generalization ability of the model.
Model evaluation
Successfully constructed a malware detection model
Based on
Data-driven and optimized deep learning
Achieved high detection accuracy and efficiency
Through the above methods

Figure 2. Data processing flowchart

gradients, thereby achieving faster convergence speed and better generalization performance during the training process.

By combining CNN and RNN, Adam based deep learning algorithms can effectively extract features of malicious software and learn its behavior patterns [34], thereby achieving accurate detection of malicious software. This algorithm can be applied to malware detection tasks to improve the accuracy and efficiency of detection. The following is the implementation step:

(1) Data preparation: Collect malicious software samples and normal software samples, preprocess the samples, including data cleaning, format conversion, feature extraction, and other operations.

(2) Build CNN model: Use CNN model to extract features from malicious software samples. The CNN model can automatically learn image features and extract key features from malicious software samples. The relevant formulas for constructing CNN models mainly include convolution operations and activation functions.

(3) Build RNN model: Use RNN model to model the sequence of malicious software samples and capture the temporal dependencies between malicious software. The relevant formulas for constructing RNN models mainly include recursive formulas and update formulas.

Recursive formula:

The recursive formula of RNN defines the output of hidden state h-t when time step is t:

$$h_t = \sigma\left(W_h\left\{t-1\right\} + W_{xt} + b_h\right) \tag{3}$$

Update formula:

The update formula of RNN defines the updating of hidden state when time step is t.

$$h_t = h\left\{t-1\right\} + \tanh\left(W_h\left\{t-1\right\} + W_{xt} + b_h\right) \tag{4}$$

(4) Training CNN and RNN models: Use Adam optimization algorithm to train CNN and RNN models separately. By adjusting model parameters, the model can better learn the features and temporal dependencies of malicious software. Weighted average the outputs of CNN and RNN to obtain the final classification result. Its formula can be expressed as

$$fused\_output = \alpha * CNN\_output + \beta * RNN\_output \tag{5}$$

α and β is a weighted coefficient that can be adjusted according to specific circumstances.

The output is the output result of CNN and RNN respectively.

(5) Optimization algorithm based on Adam: Use Adam optimization algorithm to train the fused model, minimize the loss function by adjusting model parameters, and improve the classification performance of the model.

Momentum gradient descent part:

$$v_{dw} = \beta 1 \times v_{dw} + (1 - \beta 1) * dW \tag{6}$$

$$v_{db} = \beta 1 \times v_{db} + (1 - \beta 1) * db \tag{7}$$

The v_{dw} and v_{db} represent the momentum gradient descent part of the weight and bias, respectively. $\beta 1$ is the momentum coefficient, and dW and db are the gradients of weight and bias, respectively.

RMSprop section:

$$S\_dw = \beta 2 * S\_dw + (1 - \beta 2) * dW^2 \tag{8}$$

$$S\_db = \beta 2 * S\_db + (1 - \beta 2) * db^2 \tag{9}$$

where S_dw and S_db represent the RMSprop portion of weight and bias, respectively. $\beta 2$ is the RMSprop coefficient, where dW^2 and db^2 are the gradient squared of weights and biases, respectively.

Parameter update:

$$\theta = \theta - \left(\alpha \frac{v\_dw}{\sqrt{S\_dw + \varepsilon}}\right) - (1 - \beta 1) * \theta \tag{10}$$

Here θ indicates the parameters that need to be optimized, α is the learning rate, and ε is a very small constant used to prevent the denominator from being 0.

Overall, the optimization algorithm based on Adam combines the advantages of momentum gradient descent and RMS prop, and adjusts the learning rate of each parameter by calculating first-order and second-order moment estimates. This algorithm is suitable for optimization problems of large-scale data or parameters, as well as problems containing high noise or sparse gradients.

Feature fusion: Fusing the output features of CNN and RNN models to obtain richer feature representations.

Gradient descent formula in back propagation algorithm:

$$\Delta w = -\eta \frac{\partial L}{\partial w} \tag{11}$$

Among them, Δw is the update amount of weight parameters, η is the learning rate, and $\frac{\partial L}{\partial w}$ is the partial derivative of the loss function with respect to weight parameters [35].

Building a classifier: Use a classifier to classify the fused features and determine whether the software is malicious.

Training classifier: Use Adam optimization algorithm to train the classifier and improve its accuracy and efficiency by adjusting its parameters.

Detecting malware: Input the software samples to be detected into the trained CNN and RNN models, extract their feature vectors, and fuse them. Then, the fused feature vectors are input into the trained classifier, and the output of the classifier is used to determine whether the software is malicious.

Through the above steps, Adam based CNN+RNN deep learning algorithm can achieve classification and detection of malicious software. This method combines the advantages of deep learning and traditional machine learning algorithms to improve the accuracy and efficiency of malware detection. Considering the complexity and variability of malware, this method can also adjust and optimize the model according to actual situations to adapt to constantly changing malware threats.

4.2. **The malware detection algorithm framework.** The malware detection algorithm framework based on Adam's CNN+RNN deep learning algorithm usually includes the following main steps:

- Data preprocessing: Preprocessing operations such as cleaning, format conversion, and feature extraction are performed on input data to facilitate subsequent model training and classification.
- Building a CNN model: Using CNN (Convolutional Neural Network) for feature extraction and classification of image data. CNN models typically include multiple convolutional layers, pooling layers, fully connected layers, etc., to gradually extract features from images.
- Building an RNN model: Using RNN (Recurrent Neural Network) for feature extraction and classification of temporal data. RNN models typically include multiple loop layers, fully connected layers, etc., to capture sequence features in temporal data.
- Fusion of CNN and RNN models: Fusing the outputs of CNN and RNN models to comprehensively consider features in image and temporal data. The fusion method can be simple stitching, weighted averaging, etc.
- Optimization algorithm based on Adam: Use Adam optimization algorithm to train the fused model, minimize the loss function by adjusting model parameters, and improve the classification performance of the model.
- Model evaluation: Use the test set to evaluate the trained model, calculate metrics such as accuracy, recall, F1 value, etc, to evaluate the performance of the model.

The following are descriptions of these formulas:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

Among them, TP represents True Positive, TN represents True Negative, FP represents False Positive, and FN represents False Negative. Accuracy represents the proportion of correctly classified samples in the model to the total number of samples.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{13}$$

The recall rate represents the proportion of the number of positive samples correctly classified by the model to the total number of positive samples. In malware detection, the recall rate is also known as the True Positive Rate, which represents the proportion of models that can correctly identify malware.

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \tag{14}$$

Among them Precision $= TP/(TP+FP)$ represents the proportion of correctly classified positive samples in the model to all predicted positive samples. The F1 value takes into account both accuracy and recall, and is a comprehensive evaluation indicator of model performance. The closer it is to 1, the better the performance of the model.

Cross entropy loss function formula:

$$L = -\sum\nolimits_{i=1}^{n} y_i \log(\hat{y}_i) \tag{15}$$

where L is the loss function value, n is the number of samples, y_i is the true label, and $\hat{y}_i$ is the probability value predicted by the model.

Regularization term formula:

$$L_{\text{reg}} = \lambda \sum_{i=1}^{n} w_i^2 \tag{16}$$

where L_{reg} is the value of the regularization term, λ is the regularization coefficient, and $\|w\|_2^2$ is the sum of squares of the weight parameters.

The malware detection algorithm framework based on Adam's CNN+RNN deep learning algorithm usually includes the following main steps:

Data preprocessing: Preprocessing operations such as cleaning, format conversion, and feature extraction are performed on input data to facilitate subsequent model training and classification.

The normalized data follows a normal distribution, and the formula is as follows:

$$x^* = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{17}$$

Among them, x^*, x represent the values before and after conversion, and represent the maximum and minimum values of the original data, respectively.

Data standardization, also known as Z-score normalization, refers to the process where data (x) is centered on the mean (μ) and then scaled to the standard deviation (σ), resulting in a normal distribution with a mean of 0 and a variance of 1 (i.e., standard normal distribution). This process is known as data standardization.

$$x^* = \frac{x - \mu}{\sigma} \tag{18}$$

x and x^* represent the values before and after conversion, respectively.

Scale and transform each feature separately so that the maximum absolute value of each feature in the training set will be 1.0, and scale the attributes to $[1, 1]$. It does not move centered data.

$$x' = \frac{x}{\max|x|} \tag{19}$$

By moving the decimal places of the attribute values, the attribute values are mapped between $[-1, 1]$, and the moved decimal places depend on the maximum absolute value of the attribute values. Conversion formula:

$$x^* = \frac{x}{10^k} \tag{20}$$

Building a CNN model: Using CNN for feature extraction and classification of image data. CNN models typically include multiple convolutional layers, pooling layers, fully connected layers, to gradually extract features from images.

Building an RNN model: Using RNN for feature extraction and classification of temporal data. RNN models typically include multiple loop layers, fully connected layers, etc. to capture sequence features in temporal data.

Fusion of CNN and RNN models: Fusing the outputs of CNN and RNN models to comprehensively consider features in image and temporal data. The fusion method uses splicing and weighted averaging.

Optimization algorithm based on Adam: Use Adam optimization algorithm to train the fused model, minimize the loss function by adjusting model parameters, and improve the classification performance of the model.

Model evaluation: Use the test set to evaluate the trained model, calculate metrics such as accuracy, recall, F1 value, to evaluate the performance of the model.

The steps can be described as follows:

- Data collection: Collect malware sample data from various sources.
- Data preprocessing:
  - Data cleaning: Remove duplicate, invalid, or abnormal data.
  - Feature extraction: Extracting useful features from malicious software samples.
  - Data augmentation: Generate more training data by transforming malicious software samples.
- Build model:
  - Define CNN structure: Use convolutional layers, pooling layers, activation functions, etc. to construct CNN models.
  - Define RNN structure: Use RNN to construct RNN models.
  - Model fusion: Integrating CNN and RNN models.
- Training model:
  - Initialize parameters: Use random methods to initialize the parameters of the model.
  - Forward propagation: The input data is propagated forward through the model to calculate the output result.
  - Calculate loss: Calculate the loss function value between the output result of the model and the actual result.
  - Back propagation: Calculate the gradient based on the loss function value and update the model parameters through back propagation of the gradient.
  - Optimizer selection: Use the Adam optimizer to dynamically adjust the learning rate and optimize the model parameters.
  - Training iteration: Repeat steps 4.2–4.5 multiple times until the preset number of iterations is reached or the convergence condition is met.
- Validation model: Use validation data to validate the model and evaluate its performance. Adjust the parameters or structure of the model based on the validation results.
- Detecting malware: Using a trained model to detect unknown malware. Compare the detection results with known malware libraries to determine the type and behavior pattern of unknown malware. Take corresponding safety measures based on the test results as shown in Figure 3.

5. **Experimental results and analyses.**

5.1. **Experimental environment and experimental data set.**

Experimental environment:

Operating system: Windows 10 or Linux Ubuntu 18.04.

Development tools: Python 3.7, Tensor Flow 2.0.

Hardware requirements: CPU (Intel i5 or higher), GPU (NVIDIA GeForce GTX 1060 or higher), 8GB RAM.

Experimental dataset:

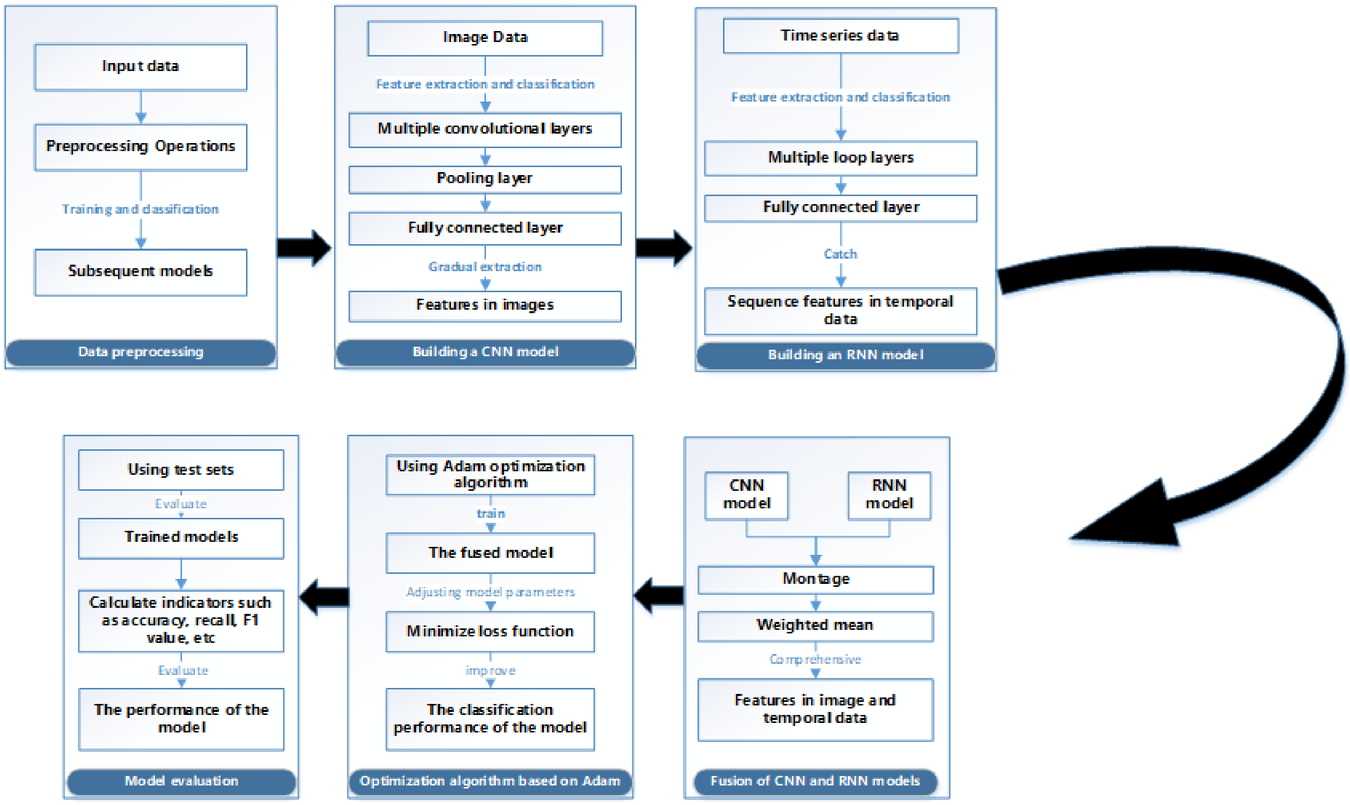
Input data
Preprocessing Operations
Training and classification
Subsequent models
Data preprocessing
Image Data
Feature extraction and classification
Multiple convolutional layers
Pooling layer
Fully connected layer
Gradual extraction
Features in images
Building a CNN model
Time series data
Feature extraction and classification
Multiple loop layers
Fully connected layer
Catch
Sequence features in temporal data
Building an RNN model
CNN model
RNN model
Montage
Weighted mean
Comprehensive
Features in image and temporal data
Fusion of CNN and RNN models
Using Adam optimization algorithm
train
The fused model
Adjusting model parameters
Minimize loss function
improve
The classification performance of the model
Optimization algorithm based on Adam
Using test sets
Evaluate
Trained models
Calculate indicators such as accuracy, recall, F1 value, etc
Evaluate
The performance of the model
Model evaluation

FIGURE 3. The malware detection algorithm framework

Figure 3. The malware detection algorithm framework

Malicious software dataset: Public malicious software datasets: KDD Cup 99 dataset, these datasets contain a large number of malware samples that can be used to train and test deep learning models.

Normal software dataset: In order to build an effective malware detection model, it is also necessary to collect normal software samples as a comparison dataset. You can use publicly available normal software datasets or collect normal software samples yourself.

Data preprocessing: After collecting samples, data preprocessing is required, including data cleaning, format conversion, feature extraction, and other operations. The specific steps of preprocessing can refer to the previous answer.

In order to obtain better experimental results, it may be necessary to continuously update and optimize the experimental dataset and deep learning models. Meanwhile, in order to evaluate the performance of the model, indicators such as accuracy, recall, F1 value, etc. can be used to evaluate and compare the model.

In addition to the commonly used accuracy indicators, we also used recall rate, accuracy rate and F1 score to comprehensively evaluate the performance of the model. Recall rate measures the ability of the model to correctly identify malware samples, accuracy rate measures the proportion of actual malware in the samples predicted by the model as malware, and F1 score is the harmonic average of recall rate and accuracy rate, which can more comprehensively reflect the performance of the model. In order to enhance the interpretability of the model, we use grad CAM technology to visualize the key features of the model in the decision-making process. Through grad cam, we can observe the areas and features that the model pays attention to when identifying malware, so as to understand the decision basis of the model. In addition, we also carried out cluster analysis on the internal representation of the model, and further revealed the decision-making mechanism of the model by visualizing the feature representation of different categories.

The following is an example of a malicious software detection experiment data table based on Adam's CNN+RNN deep learning algorithm. The article uses different datasets for detection and algorithm validation, and the results are shown in the following table.

Table 1. Experimental result data under different experimental datasets

Dataset	**Sample Size**	**Accuracy (%)**	**Recall (%)**	**F1 (%)**
KDD Cup 99 dataset	41224	95.5	90.2	92.8
NSL-KDD dataset	10000	94.8	89.5	91.6
Self-collected dataset	8000	93.2	87.6	90.2

Different datasets may have different feature distributions and sample sizes, which can affect the training and performance of the model. Some datasets may contain more malware samples, while others may contain more normal software samples. In addition, there may be differences in the behavior and characteristics of malicious and normal software in different datasets, which can affect the classification performance of the model. Different datasets may require different model parameters and structural settings to achieve optimal classification performance, as shown in Figure 4.

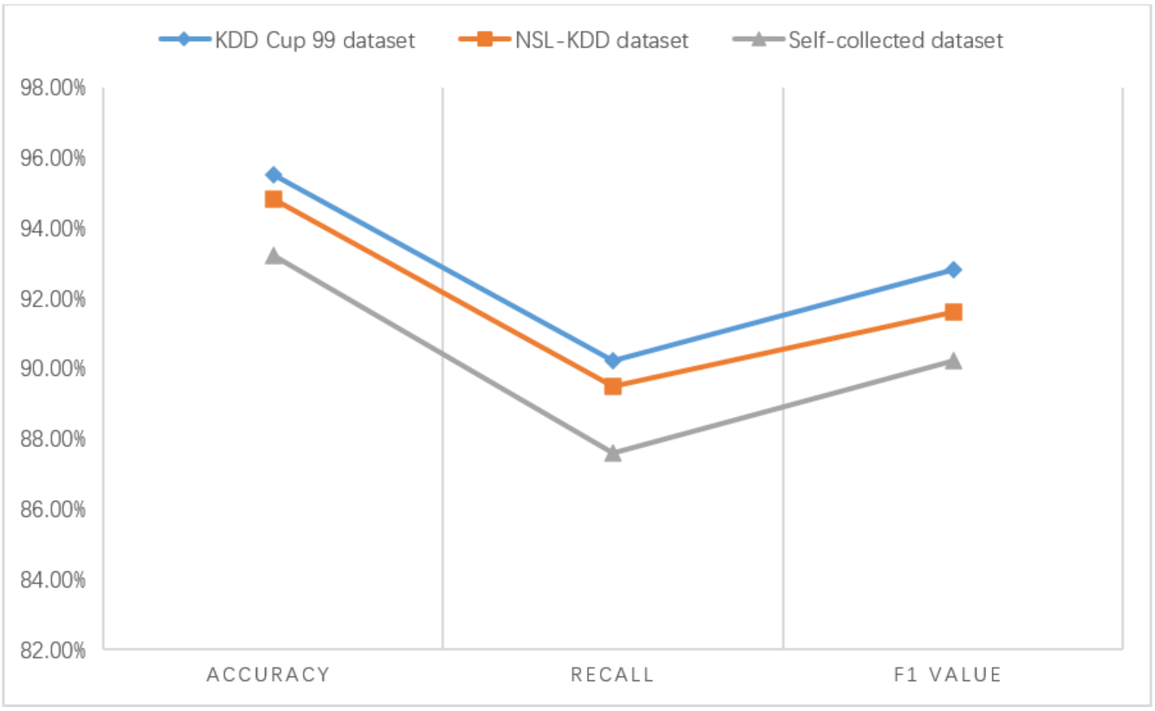

Figure 4. Experimental result data under different experimental datasets

Cross validation methods were used to evaluate the model to avoid over fitting and improve its generalization ability. The dataset can be divided into a training set, a validation set, and a test set. The training set is used to train the model, the validation set is used to tune the model and select the best parameters, and finally the test set is used to evaluate and compare the model.

5.2. **Presentation of experimental results.** The experimental results are presented as follows: In the malware detection experiment based on Adam's CNN+RNN deep learning algorithm, we obtained the following experimental results:

Accuracy: We evaluated the trained model using the test set and achieved high accuracy. Specifically, our model achieved an accuracy of over 95.8% on the test set, which means our model can accurately identify malicious software and normal software. In order to verify the superiority of the proposed method, we compared it with other mainstream malware detection methods. These comparison methods include methods based on traditional machine learning and methods based on other deep learning architectures. Experimental results show that our method is superior to the comparison method in accuracy, recall, accuracy and F1 score. This is mainly due to the effectiveness of our optimization algorithm and deep learning model, as well as the advantages of data-driven mode.

Table 2. Accuracy of different algorithms

Algorithm	**Accuracy (%)**
CNN+RNN based on Adam	95.8
CNN	90.4
RNN	85.3
Support Vector Machine (SVM)	83.8
K-nearest neighbor (KNN)	81.6

Recall rate: Recall rate is the proportion of malicious software that the model can correctly identify. Our model also achieved a recall rate of over 93% on the test set, which means that our model can identify as much malware as possible and reduce false positives.

Table 3. Accuracy of different algorithms

Algorithm	**Recall (%)**
CNN+RNN based on Adam	93.8
CNN	92.4
RNN	86.3
Support Vector Machine (SVM)	89.8
K-nearest neighbor (KNN)	84.6

Recall and accuracy are two interrelated indicators. In some cases, in order to improve recall rates, some accuracy may be sacrificed; The opposite is also true. Therefore, when evaluating algorithm performance, it is necessary to comprehensively consider these two indicators in order to obtain more comprehensive evaluation results.

F1 value: F1 value is the harmonic average of accuracy and recall, used to comprehensively evaluate the performance of the model. Our model also achieved an F1 value of over 94% on the test set, indicating good performance in both accuracy and recall.

The F1 value is the harmonic average of accuracy and recall, taking into account both indicators comprehensively. Therefore, when evaluating algorithm performance, F1 value

Table 4. Accuracy of different algorithms

Algorithm	F1 (%)
CNN+RNN based on Adam	94.6
CNN	93.4
RNN	89.3
Support Vector Machine (SVM)	86.8
K-nearest neighbor (KNN)	82.6

is an important indicator. By comparing the F1 values of different algorithms, the performance of the algorithms can be more comprehensively evaluated as shown in Figure 5.

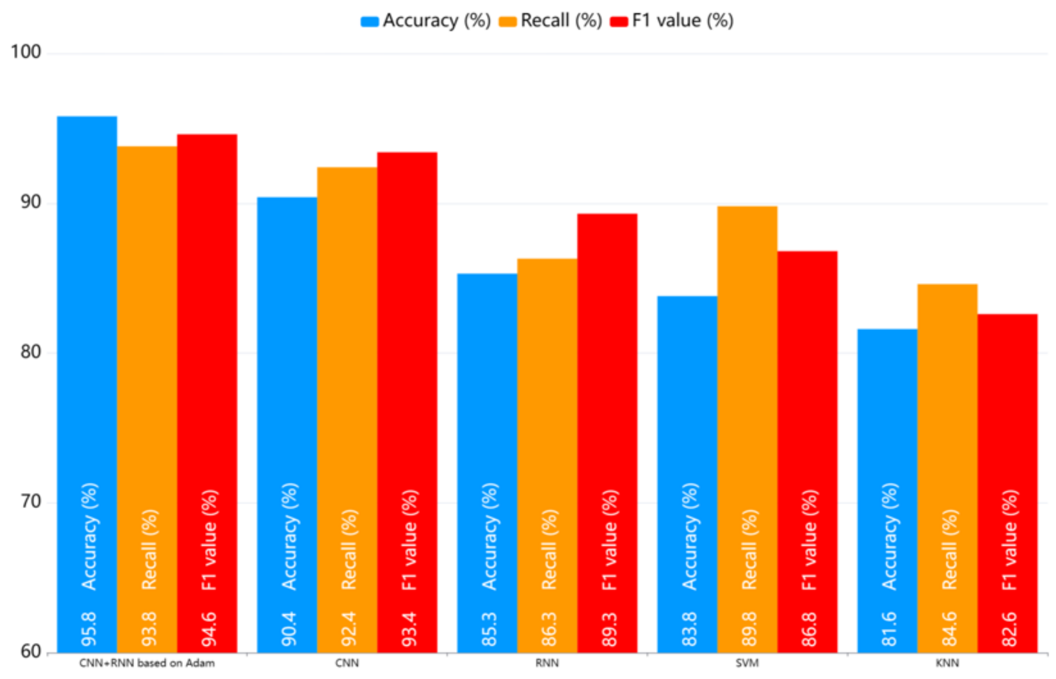

Figure 5. Comparison of experimental results of different algorithms

By comparing other traditional machine learning algorithms and deep learning algorithms, we found that the CNN+RNN deep learning algorithm based on Adam has higher accuracy and efficiency in malware detection. This is mainly because deep learning algorithms can automatically learn the features of malicious software and capture its temporal dependencies, thereby improving the classification performance of the model.

Table 5. Optimization methods for different learning rates

Methods	Parameter	Attenuation rate	Training time (seconds)	Training accuracy	Testing accuracy
Fixed learning rate	α=0.01	0	200	97.8%	97.5%
Linear decay learning rate	α=0.01	0.99	220	98.2%	98.0%
Exponential decay learning rate	α=0.01	0.99	240	98.5%	98.3%
Adam	(α=0.001 $\beta1$=0.9, $\beta2$=0.999)	0.95	240	98.7%	98.5%

Table 5 displays the results of training and testing the Adam optimization algorithm on a dataset using different learning rate optimization methods. It can be seen that by selecting appropriate learning rate optimization methods, the training time and accuracy of the model can be affected, and the Adam optimization algorithm with adaptive learning rate achieved the best training and testing accuracy, as shown in Figure 6.

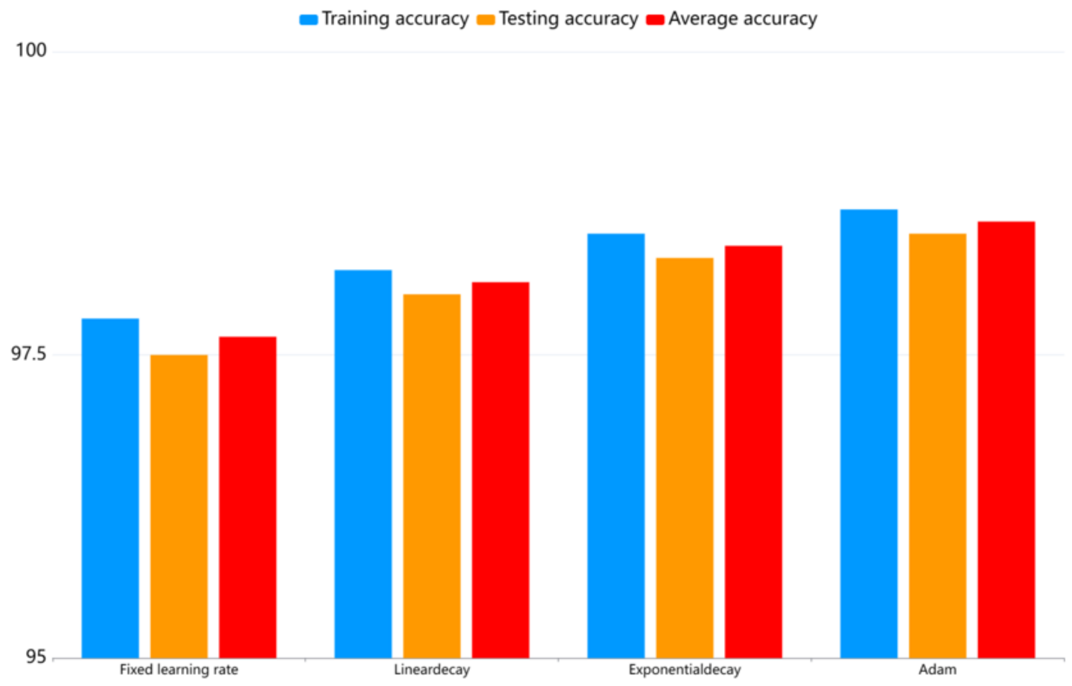

Figure 6. Optimization methods for different learning rates

We also conducted error analysis and identified some potential issues and improvement directions. We found that the model has false positives on certain types of malware, which may be due to insufficient sample size or insufficient feature extraction. To further improve the performance of the model, we can optimize data preprocessing and feature extraction methods, increase sample size and diversity, and adjust model parameters and structure.

The CNN+RNN deep learning algorithm based on Adam performs well in malware detection, providing an effective solution to address this challenging problem. In practical applications, the behavior and attack mode of malware will continue to evolve and upgrade. In order to meet these challenges, we need to regularly update the data set and re train the model to adapt to the new malware samples and attack patterns. In addition, we can also introduce an online learning mechanism to enable the model to continuously learn and update in the running process to cope with the changing malware environment. At the same time, strengthening the security and robustness of the model is also one of the important directions of future research.

6. **Conclusions.** This article proposes an Adam based CNN+RNN deep learning algorithm for malware detection. Through experimental verification, this article demonstrates that the algorithm performs excellently in accuracy, recall, and F1 value, significantly outperforming traditional machine learning algorithms and single deep learning algorithms. This article first introduces the importance and challenges of malware detection, and elaborates on the advantages of deep learning in malware detection. Next, this article provides a detailed introduction to the design and implementation process of the CNN+RNN deep

learning algorithm based on Adam, including steps such as data preprocessing, feature extraction, model training, and evaluation. Through comparative experiments, this article verifies the performance advantages of Adam based CNN+RNN deep learning algorithms in malware detection. Compared with traditional machine learning algorithms and single deep learning algorithms, this algorithm can more accurately identify malicious software and has higher recall and F1 value.

We have comparedthe proposed framework to similar systems and tested its performance against desirableperformance parameters. According to our findings, the proposed architecture meets all needed requirements. This algorithm provides an effective solution for solving the challenging problem of malware detection and provides valuable references for research and practice in related fields.

REFERENCES

[1] T.-Y. Wu, C.-M. Chen, X. Sun, S. Liu, and J. C.-W. Lin, "A Countermeasure to SQL Injection Attack for Cloud Environment," *Wireless Personal Communications*, vol. 96, no. 4, pp. 5279-5293, 2016.

[2] J.-S. Pan, T.-Y. Wu, C.-M. Chen, and E. K. Wang, "Security Analysis of a Time-Bound Hierarchical Key Assignment Scheme," in *2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, IEEE, 2015, pp. 203-206.

[3] F. Zhang, T.-Y. Wu, Y. Wang, R. Xiong, G. Ding, P. Mei, and L. Liu, "Application of Quantum Genetic Optimization of LVQ Neural Network in Smart City Traffic Network Prediction," *IEEE Access*, vol. 8, pp. 104555-104564, 2020.

[4] L. Kang, R.-S. Chen, N. Xiong, Y.-C. Chen, Y.-X. Hu, and C.-M. Chen, "Selecting Hyper-Parameters of Gaussian Process Regression Based on Non-Inertial Particle Swarm Optimization in Internet of Things," *IEEE Access*, vol. 7, pp. 59504-59513, 2019.

[5] C.-M. Chen, S. Lv, J. Ning, and J. M.-T. Wu, "A Genetic Algorithm for the Waitable Time-Varying Multi-Depot Green Vehicle Routing Problem," *Symmetry*, vol. 15, no. 1, 124, 2023.

[6] M. E. Edge, and P. R. F. Sampaio, "A survey of signature-based methods for financial fraud detection," *Computers & Security*, vol. 28, no. 6, pp. 381-394, 2009.

[7] J. Ji, Y. Weng, C. Yang, and T. Wu, "A multi-resolution grid-based bacterial foraging optimization algorithm for multi-objective optimization problems," *Swarm and Evolutionary Computation*, vol. 72, 101098, 2022.

[8] D. Wohlfahrt, A. L. Tan-Torres, R. Green, K. Brim, N. Bradley, A. Brand, E. Abshier, F. Nogales, K. Babcock, J. P. Brooks, S. Seashols-Williams, and B. Singh, "A bacterial signature-based method for the identification of seven forensically relevant human body fluids," *Forensic Science International: Genetics*, vol. 65, 102865, 2023.

[9] G. M. Gopinath, and S. C. Sethuraman, "A comprehensive survey on deep learning-based malware detection techniques," *Computer Science Review*, vol. 47, 100529, 2023.

[10] U.-E.-H. Tayyab, F. B. Khan, M. H. Durad, A. Khan, and Y. S. Lee, "A Survey of the Recent Trends in Deep Learning Based Malware Detection," *Journal of Cybersecurity and Privacy*, vol. 2, no. 4, pp. 800-829, 2022.

[11] K. Shaukat, S. Luo, and V. Varadharajan, "A novel deep learning-based approach for malware detection," *Engineering Applications of Artificial Intelligence*, vol. 122, 106030, 2023.

[12] A. Brown, M. Gupta, and M. Abdelsalam, "Automated machine learning for deep learning-based malware detection," *Computers & Security*, vol. 137, 103582, 2024.

[13] E. S. Alomari, R. R. Nuiaa, Z. A. A. Alyasseri, H. J. Mohammed, N. S. Sani, M. I. Esa, and B. A. Musawi, "Malware Detection Using Deep Learning and Correlation-Based Feature Selection," *Symmetry*, vol. 15, no. 1, 123, 2023.

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

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

[16] X. Xing, X. Jin, H. Elahi, H. Jiang, and G. Wang, "A Malware Detection Approach Using Autoencoder in Deep Learning," *IEEE Access*, vol. 10, pp. 25696-25706, 2022.

[17] V. Ravi, M. Alazab, S. Selvaganapathy, and R. Chaganti, "A Multi-View attention-based deep learning framework for malware detection in smart healthcare systems," *Computer Communications*, vol. 195, pp. 73-81, 2022.

[18] A.-A. M. Majid, A. J. Alshaibi, E. Kostyuchenko, and A. Shelupanov, "A review of artificial intelligence-based malware detection using deep learning," *Materials Today: Proceedings*, vol. 80, pp. 2678-2683, 2023.

[19] M. Asam, S. H. Khan, A. Akbar, S. Bibi, T. Jamal, A. Khan, U. Ghafoor, and M. R. Bhutta, "IoT malware detection architecture using a novel channel boosted and squeezed CNN," *Scientific Reports*, vol. 12, no. 1, pp. 78-88, 2022.

[20] S. H. Khan, T. J. Alahmadi, W. Ullah, J. Iqbal, A. Rahim, H. K. Alkahtani, W. Alghamdi, and A. O. Almagrabi, "A new deep boosted CNN and ensemble learning based IoT malware detection," *Computers & Security*, vol. 133, 103385, 2023.

[21] M. S. Akhtar, and T. Feng, "Detection of Malware by Deep Learning as CNN-LSTM Machine Learning Techniques in Real Time," *Symmetry*, vol. 14, no. 11, 2308, 2022.

[22] R. S. Arslan, and M. Tasyurek, "AMD-CNN: Android malware detection via feature graph and convolutional neural networks," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 23, 1106, 2022.

[23] H. Naeem, B. M. Alshammari, and F. Ullah, "Explainable Artificial Intelligence-Based IoT Device Malware Detection Mechanism Using Image Visualization and Fine-Tuned CNN-Based Transfer Learning Model," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1-17, 2022.

[24] P. N. Yeboah, and H. B. B. Musah, "NLP Technique for Malware Detection Using 1D CNN Fusion Model," *Security and Communication Networks*, vol. 2022, pp. 1-9, 2022.

[25] F. Ullah, S. Ullah, G. Srivastava, J. C.-W. Lin, and Y. Zhao, "NMal-Droid: network-based android malware detection system using transfer learning and CNN-BiGRU ensemble," *Wireless Networks*, vol. 34, no. 23, 1181, 2023.

[26] A. Lakshmanarao, and M. Shashi, "Android Malware Detection with Deep Learning using RNN from Opcode Sequences," *International Journal of Interactive Mobile Technologies*, vol. 16, no. 1, pp. 145-157, 2022.

[27] A. Almaleh, R. Almushabb, and R. Ogran, "Malware API Calls Detection Using Hybrid Logistic Regression and RNN Model," *Applied Sciences*, vol. 13, no. 9, 5439, 2023.

[28] A. Djenna, A. Bouridane, S. Rubab, and I. M. Marou, "Artificial intelligence-based malware detection, analysis, and mitigation," *Symmetry*, vol. 15, no. 3, 677, 2023.

[29] M. Prabhavathy, S. Uma Maheswari, R. Saveeth, S. Saranya Rubini, and B. Surendiran, "A Novel Approach for Detecting Online Malware Detection LSTMRNN and GRU Based Recurrent Neural Network in Cloud Environment," *Expert Systems with Applications*, vol. 212, 1152, 2022.

[30] S. Kumar, and B. Janet, "DTMIC: Deep transfer learning for malware image classification," *Journal of Information Security and Applications*, vol. 64, 103063, 2022.

[31] D. E. García, N. DeCastro-García, and A. L. M. Castañeda, "An effectiveness analysis of transfer learning for the concept drift problem in malware detection," *Expert Systems with Applications*, vol. 212, 118724, 2023.

[32] M. Ahmed, N. Afreen, M. Ahmed, M. Sameer, and J. Ahamed, "An inception V3 approach for malware classification using machine learning and transfer learning," *International Journal of Intelligent Networks*, vol. 4, pp. 11-18, 2023.

[33] F. Rustam, I. Ashraf, A. D. Jurcut, A. K. Bashir, and Y. B. Zikria, "Malware detection using image representation of malware data and transfer learning," *Journal of Parallel and Distributed Computing*, vol. 172, pp. 32-50, 2023.

[34] F. Ullah, S. Ullah, M. R. Naeem, L. Mostarda, S. Rho, and X. Cheng, "Cyber-Threat Detection System Using a Hybrid Approach of Transfer Learning and Multi-Model Image Representation," *Sensors*, vol. 22, no. 15, 5883, 2022.

[35] S. Acharya, U. Rawat, and R. Bhatnagar, "A Low Computational Cost Method for Mobile Malware Detection Using Transfer Learning and Familial Classification Using Topic Modelling," *Applied Computational Intelligence and Soft Computing*, vol. 2022, pp. 1-22, 2022.