IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON DEEP LEARNING: SECURITY AND FORENSICS RESEARCH ADVANCES AND CHALLENGES

Received September 14, 2019, accepted October 6, 2019, date of publication October 18, 2019, date of current version November 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2948358

How to Make Attention Mechanisms More Practical in Malware Classification

XIN MA, SHIZE GUO, HAIYING LI, ZHISONG PAN, JUNYANG QIU, YU DING, AND FEIQIONG CHEN

Command and Control Engineering College, Army Engineering University of PLA, Nanjing 210007, China

Corresponding author: Feiqiong Chen (feiqongchen@sina.cn)

This work was supported by the National Key Research and Development Program of China under Project 2018YFB0805000.

ABSTRACT Malware and its variants continue to pose a threat to network security. Machine learning has been widely used in the field of malware classification, but some emerging studies, such as attention mechanisms, are rarely applied in this field. In this paper, we analyze the correspondence between bytecode and disassembly of malware, and propose a new feature extraction method based on multi-dimensional sequence. Also, we construct a new classification framework based on attention mechanism and Convolutional Neural Networks mechanism. Furthermore, we also compare the different architectures based on the attention mechanisms. Experiments on open datasets show that our feature extraction method and our framework have a good classification effect, and the accuracy rate is 0.9609.

INDEX TERMS Attention mechanisms, multi-dimensional sequence, disassembly code.

I. INTRODUCTION

Malware and its variants usually perform unauthorized operations to gain illegal benefits, which have caused great damage to end-users. According to 360 Security company's announcement in 2018, they [1] intercepted 270 million new malicious program samples on the PC side, an average of 752,000 new malicious program samples on the PC side every day. With the emergence of malware and its variants, network security continues to be seriously threatened. It has become an important task to effectively detect malware and its variants. Machine learning has shown good application prospects in malware detection and classification. Traditional machine learning methods use Support Vector Machine (SVM) [12], Random Forest [3] and other models to detect malware, which is mainly relied on extracting feature vectors manually. For example, in the work of [7], the authors make a deep permission usage analysis and propose a malware detection system based on the SVM model, which shows a good classification result. As deep learning has a more powerful representation ability, it has achieved great success in the field of image and natural language. Therefore, deep learning may have more potential ability in the field of malware detection and classification.

The associate editor coordinating the review of this manuscript and approving it for publication was Kim-Kwang Raymond Choo.

From now on, malware detection methods based on deep learning mainly focus on image [13], signal [14], and Application Programming Interface (API) sequence [10]. It may not stimulate the potential ability of a deep learning model if we just simply transform malware into the input vector. How to effectively use expert knowledge to process data, and transform it into the input needed by deep learning model, and design a specific deep learning model are the key to improve the effectiveness of deep learning models in detecting malware area.

In this paper, we try to start with the analysis of the corresponding relationship between assembly code and binary code, and further excavate the relevant data characteristics. At the same time, we also try to migrate the attention mechanism from the common natural language processing model to the field of malware classification by analyzing the principle of attention mechanism. Because the same kind of malware will also show some similar characteristics in binary, it can achieve the classification of malware by discovering binary features for malware.

The main contributions of this paper are as follows:

Firstly, combined with expert knowledge, a multi-dimensional input sequence with a strong correlation is extracted, which effectively adds the characteristics information of malware.

Secondly, a malware classification framework (**ACNN**) based on attention mechanism and Convolutional Neural

Networks (CNN) mechanism are designed by using deep learning, which improves the ability of deep learning to classify malware.

Thirdly, we deeply analyze the essence of the attention mechanism with different architectures and successfully apply the attention mechanism derived from the Encoder-Decoder framework to the malware classification framework.

Experiment on open Microsoft's 2015 kaggle database shows that our method has high accuracy and achieves good classification results.

II. RELATED WORK

Machine learning mainly includes traditional machine learning methods and deep learning methods in the field of malware classification. Traditional methods we refer to are methods that do not use deep learning, such as Random Forest, SVM, etc. We analyze the difference between the two from the perspectives of feature extraction and model construction.

A. MALWARE CLASSIFICATION BASED ON TRADITIONAL MACHINE LEARNING

Compared with deep learning, it is easier for traditional machine learning methods to construct effective feature vectors, because they can use feature extraction methods more flexibly by using for reference in the fields of natural language processing and image processing, such as grayscale, N-Gram, operator statistics, etc.

Liu [10] uses the shared nearest neighbor clustering algorithm to construct the input feature vector by extracting image textures, operation code features, and API features.

Nataraj [13] uses image processing techniques to extract the GIST features of images by converting malware into images, and then uses Gaussian filters for classification.

Huang [6] uses N-Gram to process API call sequences to form feature vectors, and then uses k-Nearest Neighbor (kNN), Decision Tree, and SVM for classification.

Carlin [4] constructs a virtual execution environment to capture the assembly code of malware running, and then uses the hidden Markov model for classification.

Nataraj [14] first converts malware into a signal, and then uses the characteristics of the signal to classify the malware.

B. MALWARE CLASSIFICATION BASED ON DEEP LEARNING

Deep learning mainly refers to models such as CNN and Recurrent Neural Network (RNN) that derive from the neural network. They are more sensitive to data with image features and time-series features. Therefore, the key point is how to extract data with image features or time-series features more effectively.

Lu [18] first extracts the API call sequence, which merges the same API. Then they build a Long Short-Term Memory (LSTM) model with 2 hidden layers and extract the hidden layer features using the Max-Pooling layer.

Qian [15] divides the API characteristics into 16 categories based on experience, and then constructs an image as input according to the category and number of occurrences of the API, and then applies the CNN model.

Gibert [5] uses the CNN model to learn image features by directly converting the disassembled ASM files into pixel maps.

Kolosnjaji [9] uses the sandbox to extract the dynamic API execution sequence and then removes the recurring API calls to form the input API sequence. The deep learning detection model is then constructed by using CNN and LSTM models.

In the work of [12] and [11], they all use Principal Component Analysis (PCA) method to extract its features after transforming the malware into the picture.

In the work of [8], the author uses autoencoder to learn the malware figure features, which achieves a good classification on the Microsoft Kaggle database. The difference is that they only use 8 categories on the database, instead of the original 9 categories.

In summary, we analyze the approach taken by machine learning in the field of malware classification and detection. Traditional machine learning methods and deep learning methods have similarities in feature extraction. They often process malicious code data by using some common methods such as natural language processing, image processing method, etc. The difference is that traditional machine learning methods can extract targeted features by using expert knowledge. However, deep learning is mostly difficult to extract targeted features by using expert knowledge, because input data is mostly limited to pictures or sequences. However, deep learning has shown strong learning ability in the fields of image and natural language, so it is worth exploring how to apply it in the field of malware classification. Therefore, on the one hand, to dig the potential ability of deep learning, it is necessary to use expert knowledge to extract more favorable input features. On the other hand, it is necessary to design a targeted deep learning model based on features.

At present, there are few studies on malware classification based on attention mechanisms, so we will try to design a targeted attention mechanism framework based on the characteristics of data and attention mechanism.

III. OUR RESEARCH ABOUT MALWARE AND ATTENTION MECHANISM

A. MALWARE ANALYSIS

For a binary form of malware, when human experts analyze its maliciousness, they mainly focus on a series of malicious operations taken by the code. Usually, we will study its disassembly code, which reflects the execution logic of the program to some extent. We detect malware by studying the disassembly code. It can be said that analyzing malware from the perspectives of binary and disassembly code is an important part of the field of malware detection. A considerable number of scholars have engaged in such research and

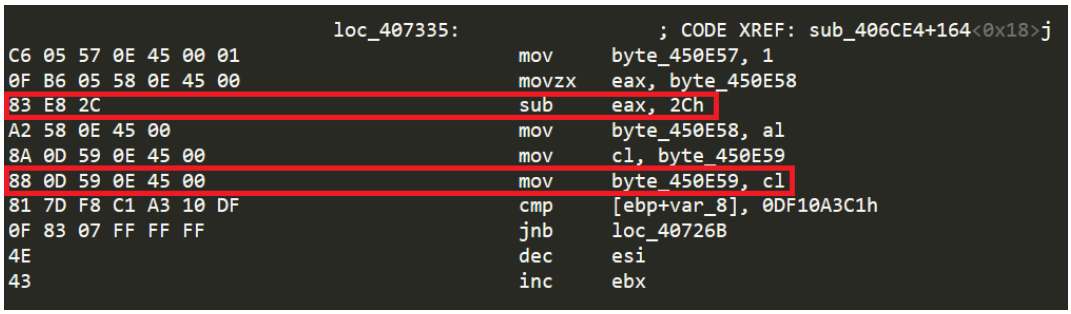


FIGURE 1. Disassembly analysis . This is a screenshot of the ASM file generated by IDA Pro.

have given good feature extraction and model construction methods.

However, they [13], [14] may not make effective use of the information when processing binary and disassembly code. For example, some scholars directly convert binary code into a picture with a fixed length. There is no evidence for setting this number as a fixed length. Others directly extract the opcodes in the disassembly code to form a sequence, which preserves the sequence information to some extent but loses considerable disassembly code semantic information. We know that there still may be a big difference in the assembly statement meaning, even if its opcodes are the same but the operands are different. For example, 'mov eax, 0xC' and 'mov ebx, 0x11' are different. If we just focus on opcodes, then they both get the same opcode 'mov'. Therefore, it is not enough to just extract the opcode.

In reality, there is a close correlation between binary and disassembly code. An opcode must be a hexadecimal number. At the same time, an assembly statement must correspond to several hexadecimal numbers. If we associate binary and assembly statements, we can extract the characteristics of both the sequence information in the vertical direction and the associated information in the horizontal direction as shown in Figure 1. From Figure 1 we can see that the left hexadecimal code and the right assembler code in the two red rectangles are related. However, the hexadecimal length corresponding to each assembly statement may be different.

Due to this corresponding relationship, it will be better to retain the characteristics of malware and apply deep learning.

B. ATTENTION MECHANISM ANALYSIS

The attention mechanism is a deep learning model that emerged in 2014. It uses the human brain signal processing mechanism for reference and aims to focus on more critical information from lots of information for the current mission. By using the attention mechanism, it filters and extracts information more effectively. In recent years, attention mechanisms have been widely used in image recognition, natural language processing, and speech recognition. Especially in Encoder-Decoder framework in natural language processing, attention mechanism is mostly used. By entering a sentence into the frame, it can output another sentence. By using the attention mechanism, we can get the weight values for different words in a sentence. And the key words always output more big weight value. It works similarly in the human brain way in which we always notice some keywords in a sentence. This is why attention mechanism is more successful.

Essentially, the attention mechanism [16] can be described as giving a query vector Q and a set of key-value pairs (K, V), mapping the set of queries and key-value pairs into an output vector through a mapping function. The query, key-value pairs, and output are all vectors. The elements of the output vector are weighted and averaged by the key-value pairs of all vector values in the vector set. Wherein, the weight corresponding to each value vector is got by the calculation function F of the query vector and the key vector, that is,

$$Attention(Q, K, V) = F(Q, K)V.$$

We further analyze this formula.
Let $Q = (q_1, q_2, \ldots, q_n)$
Let $AV = Attention\,(Q, K, V)$
Let $AV = (av_1, av_2, \ldots, av_n)$

In this formula, we can consider av_i as the global attention value of q_i. Then by applying the attention mechanism, we get the attention score of each query element q_i on the whole.

At present, most attention models are applied in the Encoder-Decoder framework, and there are few applications for classification and detection frameworks.

In the Encoder-Decoder framework, the av_{i-1} generated by each word q_{i-1} is the current intermediate state, which participates in the output of the next word q_i. This becomes very beneficial. Therefore, the attention mechanism is very effective for the Encoder-Decoder framework in translation models.

In the classification and detection framework, if the attention mechanism is used, it specifically needs to combine with the mission. How to make good use of each part's attention global score is the key to the successful application of the model. If we just want to transfer their codes for execution, obviously there will be no practical meaning and no good result.

Further, we will compare and analyze the various implementation architectures of the attention mechanism in the malware classification experiment.

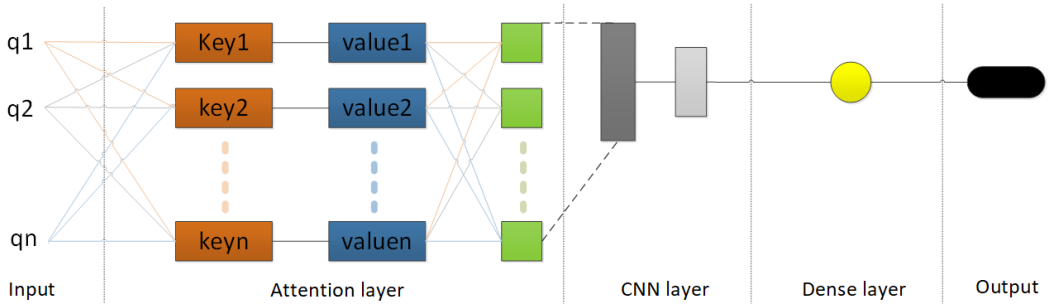


FIGURE 2. Our ACNN framework. This model consists of 5 layers: Input, attention layer, CNN layer, dense layer and output.

IV. OUR ACNN CLASSIFICATION FRAMEWORK AND OTHER DESIGN FRAMEWORKS

According to our analysis of malware and attention mechanisms in the previous section, we will construct our feature extraction method and design a unique classification model **(ACNN)** based on attention mechanism and CNN mechanism. The whole process is divided into 4 steps: malware parsing, feature extraction, modeling, and model prediction.

A. MALWARE PARSING

We employ IDA Pro [2] as our parser to build an automated parsing module. Each executable code will generate two files, that is, Byte file and ASM file. The byte file is a hexadecimal representation of the executable code, and the ASM file is the assembly code representation of the disassembled executable code.

B. FEATURE EXTRACTION

We perform feature extraction based on the code segment with sequential features. According to the previous section of malware analysis, each assembly statement will correspond to a set of hexadecimal numbers. First, we extract the assembly instructions in the disassembly ASM file and then extract the corresponding hexadecimal numbers from the byte file. Since the length of the hexadecimal code corresponding to each assembly instruction is different, we set the length of the hexadecimal code for each assembly instruction to 20. It will be filled with 0 if less than 20 and it will be cut off if more than 20. This forms a two-dimensional vector, which will be entered into our model.

C. MODELING

Here, we will explore how to apply attention mechanisms to the classification framework. According to the previous section on the study of attention mechanisms, we can obtain the scores of each word on the whole by using the attention mechanism. In our data set, we will get the importance score of each assembly instruction in the whole.

1) ATTENTION MECHANISMS + CNN MECHANISMS (ACNN)

In fact, several adjacent assembly statements usually constitute a set of assembly operations with special meaning, which is very beneficial for judging the malware category. Therefore, many scholars construct the classification model by using the 3-gram slice of the code. For the same category of malware, they must have the same slice on the fixed gram, so it can effectively extract its high-level features by setting a reasonable slice size. Therefore, we use the combination of attention mechanism and CNN mechanism to build our model, as shown in Figure 2.

From the Figure 2, it can be seen that our **ACNN** framework is divided into 5 parts: Input layer, Attention layer, CNN layer, Dense layer, and Output layer. In the Input layer, the two-dimension input vector (2000*20) is entered. Then, the vectors are input into the Attention layer, which can output different score values for different input values according to the global context. That is, the Attention layer is used to capture the key assembly instructions information, which is represented as hexadecimal code in the input vector. After that, the score values are input into the CNN layer. The CNN layer is used to capture the characteristics of these adjacent assembly statements because these adjacent assembly statements usually form a set of operations, which is also an important basis for distinguishing the malware category. Next, the Dense layer is used to capture the high-level features from the CNN layer. Finally, the Output layer outputs the final result.

In our model, the Attention layer uses a multi-head attention mechanism. In essence, it is consistent with attention mechanisms. The difference is that it uses multiple sets of attention and concatenates the results of the attentions. By using multiple sets of attentions, it can extract more features, so we use it in our **ACNN** framework.

The CNN layer extracts the high-level features based on the segment by setting the convolution kernel of the window to 3, which draws on the same 3-gram slice thought. The third part outputs the classification result through the fully connected layer.

2) LOCAL + GLOBAL ATTENTION MECHANISMS

Here, we consider the idea of drawing a hierarchical attention mechanism [19] to design the model. By analyzing the data we input, the horizontal row is the hexadecimal representation of the assembly statement, and the vertical column is the order of the assembly statements.

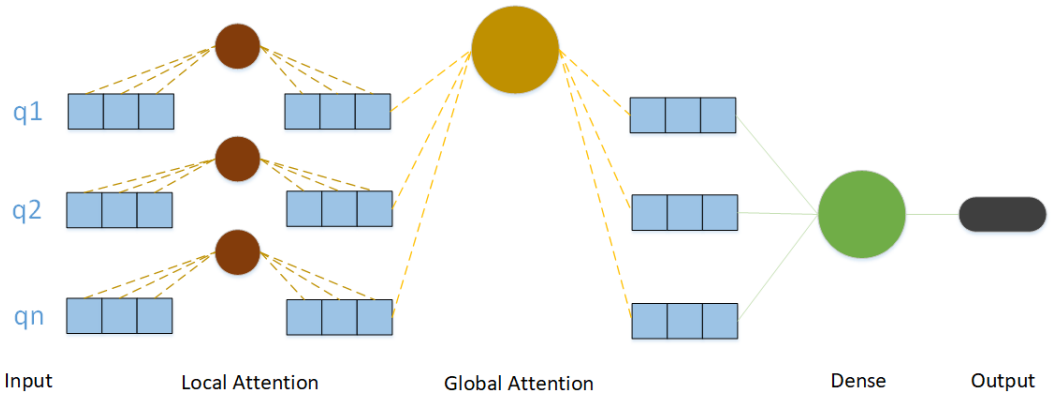


FIGURE 3. Local + global attention mechanisms. This model consists of 5 layers: Input, local attention layer, global attention layer, dense layer and output.

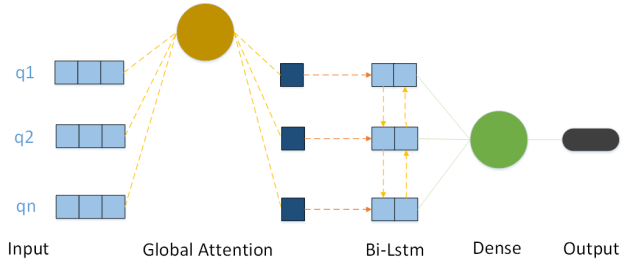


FIGURE 4. Attention mechanisms + Bi-Lstm mechanisms. This model consists of 5 layers: Input, global attention layer, Bi-Lstm layer, dense layer and output

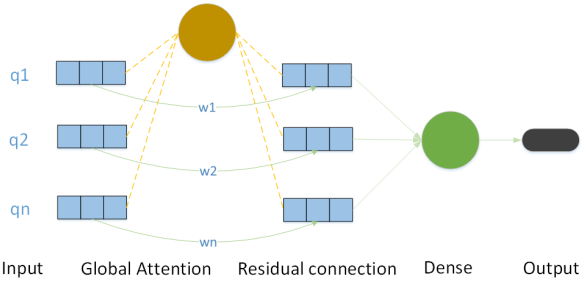


FIGURE 5. Attention mechanisms + Residual mechanisms 1. This model consists of 4 layers: Input, global attention + residual connection layer, dense layer and output. the residual connection is from input layer to value layer in attention mechanism.

Therefore, we design local attention for the assembly statements, trying to obtain different dimensional weight values in the horizontal 20 dimensions. Furthermore, we take global attention in the vertical direction, trying to extract the sequence characteristics of the data in the vertical sequence relationship. The overall structure is shown in Figure 3.

From Figure 3 we can see that this architecture is divided into 5 parts: Input layer, Local Attention layer, Global Attention layer, Dense layer, and Output layer. The Local Attention layer is used to capture the key information in the horizontal 20 dimensions. For example, we may always focus on the 'call' opcode in 'call eax' statement. Then the score values generated will be sent into the Global Attention layer.

3) ATTENTION MECHANISMS + BI-LSTM MECHANISMS

We try to draw on the idea of local attention and extract horizontal weight values by using the attention mechanism in the horizontal direction.

Due to the superior performance of the Bi-Lstm model [17] in processing sequence data, we used the Bi-Lstm model in the vertical direction after using the attention mechanism in the horizontal direction. The overall structure is shown in Figure 4.

From Figure 4 we can see that this architecture is divided into 5 parts: Input layer, Global Attention layer, Bi-Lstm layer, Dense layer, and Output layer. It uses the Bi-Lstm layer to learn the sequential features after the Global Attention layer generating the global score.

4) ATTENTION MECHANISMS + RESIDUAL MECHANISMS 1

The idea of the residual network is that after the multi-layer network is built, the network is easily degraded and the original information is lost. Therefore, the information is effectively transmitted through a cross-layer connection. Here, we draw on the idea of the residual network. After applying the attention mechanism, we add a new layer of cross-layer connection. The mathematical expression is as follows:

$$R\_Attention\_1(Q, K, V, G) = Concat(Attention(Q, K, V),\ G) \quad (1)$$

$$G = QW \quad (2)$$

Here, W is a weight matrix, which represents a linear mapping of Q. The overall structure is shown in Figure 5.

From Figure 5 we can see that it completes a residual connection from the Attention layer's input to the Attention layer's output so that it can keep the original information for the input vector.

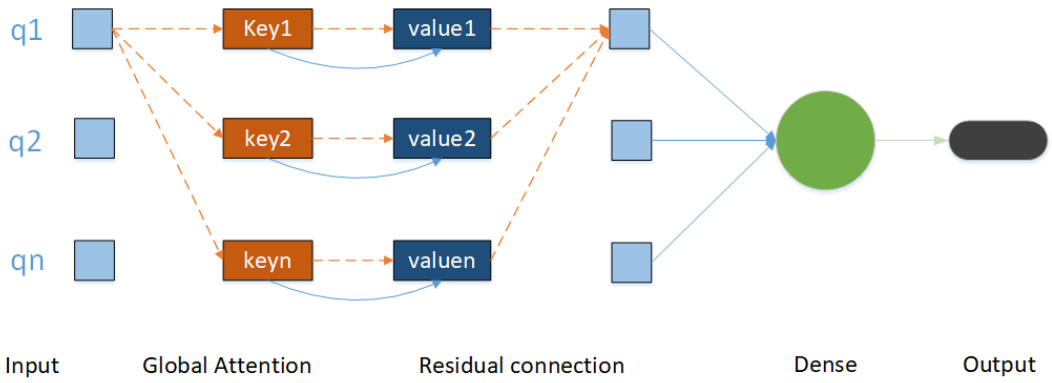


FIGURE 6. Attention mechanisms + Residual mechanisms 2. This model consists of 4 layers: Input, global attention + residual connection layer, dense layer and output. the residual connection is from key layer to value layer in attention mechanism.

5) ATTENTION MECHANISMS + RESIDUAL MECHANISMS 2

As in the previous section, we construct a network of attention mechanisms and residual mechanisms from the perspective of retaining the original information. The difference is that we have a cross-layer connection from $F(Q, K)$. The mathematical expression is as follows:

$$Attention(Q, K, V) = F(Q, K)V$$
$$R\_Attention\_2(Q, K, V) = F(Q, K)(E + V)$$

The overall structure is shown in Figure 6.

From Figure 6 we can see that it completes a residual connection from the Attention layer's key values to the Attention layer's output so that it can keep the original key information.

D. MODEL PREDICTION

We first divide the data into a training set and test set. After training the model, we input the test set into the model for judgment and obtain the classification result.

V. EXPERIMENT

A. DATA SET

We choose the data from Microsoft's 2015 kaggle competition as our data set. Their data is the malware of the window platform. After IDA Pro parsing, Byte and ASM files are generated. They total 10086 samples and are divided into 9 different types of malware, that is Ramnit, Lollipop, Kelihos_ver3, Vundo, Simda, Tracur, Kelihos_ver1, Obfuscator.ACY and Gatak. The number of each category is shown in Figure 7.

B. EXPERIMENT PARAMETERS

Our **ACNN** framework based on Attention mechanism and CNN mechanism are set as follows: The attention mechanism layer takes a multi-head attention model in which the number is set to 2 and the head size is set to 8. The convolution layer is set to a 3-layer convolution. Each layer has a convolution kernel size of 3 and a number of 100, connected to a pooling layer of size 3.

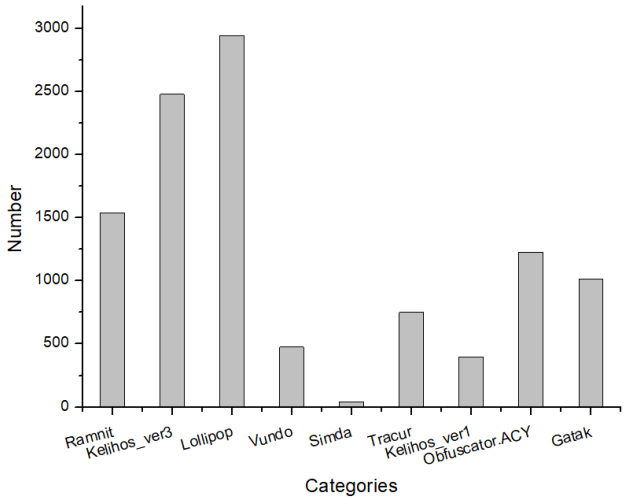


FIGURE 7. Distribution of data sets. In the picture, the x-axis represents the categories of malware, and the y-axis represents the number.

C. CLASSIFICATION RESULT

Figure 8 shows the accuracy value changes with each round. After 200 rounds of training, the accuracy reaches the maximum.

Figure 9 shows the loss value changes with each round. After 200 rounds of training, the loss reaches the minimum. As can be seen from Figures 8 and 9, our model converges quickly, has high accuracy and low loss.

From Table 1, we can find that the micro avg of the F1-score is 0.96. It proves that our **ACNN** framework has a good performance in classification results.

From Figure 10, we can find that although the total number of each category is different, the classification accuracy of each category is well.

The final classification accuracy achieved by our **ACNN** framework is 0.9609.

It shows that the **ACNN** classification framework we design is very suitable for the data we extracted, and achieves a good classification effect.

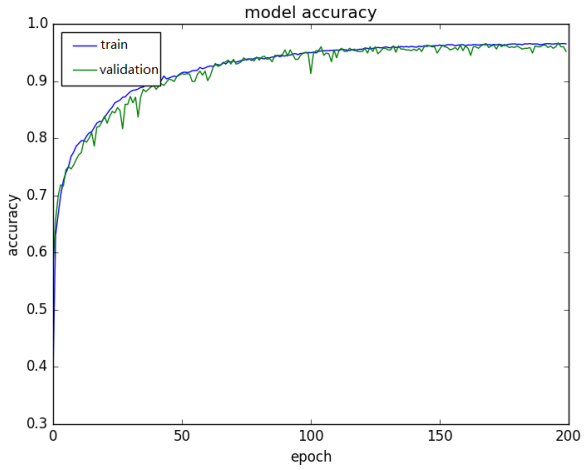


FIGURE 8. The accuracy of training and validation varies with epoch.

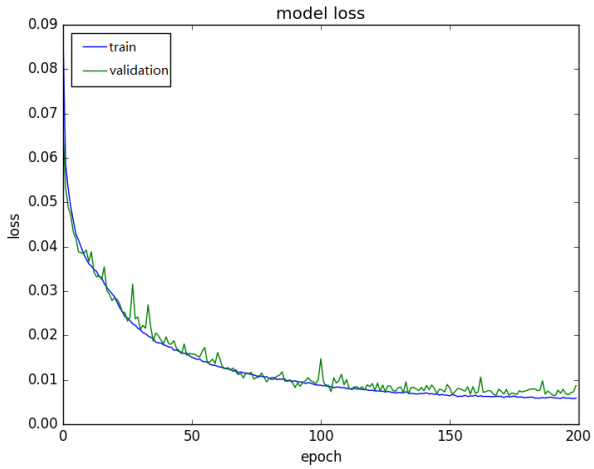


FIGURE 9. The loss of training and validation varies with epoch.

TABLE 1. The classification report of ACNN framework.

Category	Precision	Recall	F1-score	Support
Ramnit	0.93	0.91	0.92	503
Kelihos_ver3	0.97	0.98	0.98	833
Lollipop	1.00	1.00	1.00	970
Vundo	0.97	0.95	0.96	155
Simda	1.00	0.32	0.48	19
Tracur	0.80	0.96	0.87	253
Kelihos_ver1	0.98	0.95	0.96	146
Obfuscator.ACY	0.96	0.90	0.93	415
Gatak	0.98	0.98	0.98	329
micro avg	0.96	0.96	**0.96**	3623

VI. EVALUATION

To verify the effectiveness of our feature extraction method and **ACNN** framework, we set up several sets of comparison experiments. Then, we take the 10-fold cross-validation method and divide the database into a training set and testing set in each fold. Then, we count the average classification accuracy from these testing set.

A. COMPARE FEATURE EXTRACTION METHOD AND ANALYSIS

The data we enter into the model is a two-dimensional vector of which horizontal row representation represents the hexadecimal representation of each assembly statement, and vertical column representation represents the sequential execution order of the assembly statements. To verify this two-dimensional vector form is more efficient, we have designed two contrast forms. The one is commonly used one-dimensional data generated by flattening the two-dimensional vector into a one-dimensional vector. The data shape is (6000, 1). The other is also two-dimensional data, just less than the horizontal dimension we extract. We extract 20 dimensions. The data shape is (2000, 20), and the contrast shape is (2000, 3).

The model was evaluated using the LSTM model and the Random Forest model. In this experiment, we first flatten the two-dimensional vector into a one-dimensional vector as the input of the Random Forest model. The test results are shown in Table 2.

It can be seen from Table 2 that for the traditional Random Forest model, the classification accuracy under different feature extraction methods is 0.71, and the various feature extraction methods have little effect.

It proves that our feature extraction method has little effect on the traditional Random Forest model.

For the LSTM model, the larger is the dimension extracted in the horizontal direction, the better is the classification effect. The best classification accuracy reaches 0.8725 under shape (2000,20).

It proves that for the deep learning model, our feature extraction method is beneficial to enhance the horizontal correlation and improves the classification accuracy.

B. COMPARE MODEL CONSTRUCTION METHOD AND ANALYSIS

Our **ACNN** classification framework is constructed by using the attention mechanism and the CNN mechanism. For comparison, we set up four baseline models, that is, the Random Forest model, the bidirectional LSTM model, the Multi-head attention mechanism model, and the Deep Neural Networks (DNN) model.

We also compare different design architectures, mentioned in Section IV.C, namely **'Local + global attention mechanisms', 'Attention mechanisms + Bi-Lstm mechanisms', 'Attention mechanisms + residual mechanisms 1'** and **'Attention mechanisms + residual mechanisms 2'**.

1) BASELINE 1

We choose a more representative traditional machine learning method, namely the Random Forest [3], as a baseline model, which was used by the champion team in the Kaggle contest. Its parameters are set as follows:

$$n\_estimators = 500, \quad n_jobs = -1$$

TABLE 2. Comparison for different feature extraction methods.

Method	shape(6000,1)	shape(2000,3)	shape(2000,20)
LSTM	0.8123	0.8456	**0.8725**
Random Forest	0.7217	0.7177	0.7127

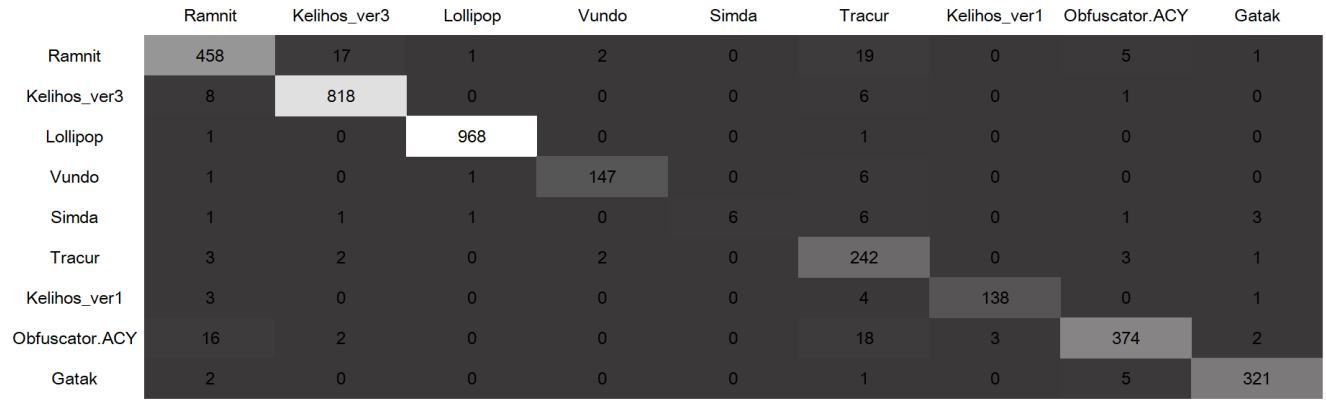


FIGURE 10. The confusion matrix of ACNN framework.

TABLE 3. Comparison for different models.

Model	Epochs	Batch-size	Optimizer	Accuracy	Loss
Random Forest [3](baseline)				0.7127	
Bi-Lstm(baseline)	200	512	adam	0.8738	0.0225
Multi-head attention [16](baseline)	200	8	adam	0.9159	0.0147
DNN(baseline)	200	512	adam	0.7353	0.0451
Local+Global Attention	200	8	adam	0.7738	0.0435
Attention + Bi-Lstm	200	8	adam	0.7453	0.0424
Attention + Dense	200	8	adam	0.8363	0.0319
Attention + CNN	200	8	adam	**0.9609**	0.0106
Attention + Residual 1	200	8	adam	0.9287	0.0135
Attention + Residual 2	200	8	adam	0.8456	0.0287

2) BASELINE 2

Recycle neural networks are effective for processing data with timing relationships. Among them, LSTM is a long and short time memory network, and Bi-Lstm is a two-way LSTM model built based on LSTM, which sometimes is more powerful than LSTM. So we choose Bi-Lstm as our baseline model. Its parameters are set as follows: We have adopted 2 layers of Bi-Lstm with 32 cells per layer.

3) BASELINE 3

In the work of [16], the authors use the attention mechanism to replace the RNN structure and the CNN structure, demonstrating the powerful ability of the attention mechanism for learning time series data. Therefore, we use the multi-head attention mechanism proposed by the authors as our baseline model.

4) BASELINE 4

Since DNN is the most basic deep learning model, we choose DNN as one of our baseline models. Its parameters are set as follows: We use five full-connection layers, of which the first four full-connection layers are set up 64 units, using **tanh** as the activation function. The last full connection layer outputs the classification results, using **sigmoid** function as the activation function. The dropout connection is used between each layer, and the setting parameter is 0.5.

5) COMPARISON AND ANALYSIS

The results of the comparison test are as shown in Table 3.

Also, we list their parameters in the Table 4.

As can be seen from Table 3, our **ACNN** framework based on the combination of attention mechanism and CNN mechanism achieves the best classification accuracy of 0.9609.

TABLE 4. Parameters for different models.

Random Forest	Bi-Lstm	Multi-Attention	DNN	Local+ Global Attention	Attention+ Bi-Lstm	Attention + Dense	Attention+CNN	Attention +Residual 1	Attention +Residual 2
n_estimators=500, n_jobs=-1	2 layers of Bi-Lstm with 32 cells per layer	number_head=2,size_head=8	5 Dense layers with per 64 units	local:20 dimension;global: 2000 dimension	global attention layer; 2 layers of Bi-Lstm with 32 cells	global attention layers; 3 layers with 32 cells per layer	multi-attention layer:number_head=2,size_head=8;3 layer convolution and each layer has a convolution kernel size of 3 and a number of 100, and each layer is connected to a pooling layer of size 3	attention layer: the input layer is connected to attention's output layer	attention layer: the key layer is connected to attention's output layer

Our **ACNN** framework is nearly 5 percentage points higher than the baseline model multi-head, nearly 9 percentage points higher than the baseline model Bi-Lstm model, and nearly 25 percentage points higher than the traditional random forest model. At the same time, **ACNN** is far ahead of other models based on attention-based mechanisms. The reason our **ACNN** framework works well is that our model is closely designed based on data characteristics. Our data is specially designed with two-dimensional data. The horizontal representation of the data is assembly statements, and its vertical representation is the executive orders of the assembly statements. And from the vertical perspective, several consecutive assembly statements have the meaning of the actual execution order, although their size may not be fixed.

Therefore, our **ACNN** framework first extracts the weight of each assembly statement in the whole through the attention mechanism and then employs a convolution kernel of size 3 to extract features in the vertical direction, thereby learning higher-dimensional features.

Other models based on attention-based mechanisms do not perform well in classification, and some even do not exceed the baseline model. The main reason for this is that they do not make full use of the characteristics of the data.

It proves that it is hard to achieve good results if we construct a model only paying attention to the reference of different design ideas that are not suitable for the input data.

C. COMPARE OTHER ALGORITHMS ON THE SAME DATASET AND ANALYSIS

As far as we know, there are a few open malware databases. Therefore, many researchers use Microsoft's kaggle data set as a benchmark. In the following 3 references, they achieved a high accuracy of 96.6% [12], 99.9% [8] and 99.1% [11] respectively on the same Microsoft's kaggle dataset. Our framework could achieve an accuracy of 0.9606, which is approximately 96.1%.

This data set contains not only its disassembly in the ASM file but also binary in the Byte file, which is represented as hexadecimal code. But, we only use part of the data set. We use the hexadecimal code in the byte file according to the assembly code of the code segment in the ASM file. However, they use the whole data, not only code segments, but also other resource segments, so they could get more information than us.

Further, compared with the work [12], their methods are 0.5 percentage points higher than ours. But, we only use part contents in the Byte file. Therefore, this result could be considered as reasonable.

Compared with the work [8] and [11], their methods are 3 percentage points higher than ours. In fact, this data set contains 9 categories. But the authors exclude 'Simda' class because it only contains 42 samples. Therefore, they do 8 class classification. However, our framework and Narayanan's method [12] do 9 class classification. From our and Narayanan's classification results, the classification result of the 'Simda' class is very low. This is one of the most probable reasons that affect the accuracy of our and Narayanan's classification results.

Overall, the accuracies of our framework and Narayanan's method [12] are both nearly 96%, and we only use part content on the data set. This shows that our framework and Narayanan's method are credible, although their classification results [8] and [11] are higher.

In fact, we have only studied the characteristics of assembly code in-depth and designed the attention mechanism. We can combine with their work and study the characteristics of malware from various perspectives to form a more comprehensive integrated detection framework, which is meaningful for the next work.

VII. CONCLUSION

In this paper, we propose a feature extraction method based on correspondence between assembly code and binary code and then construct a multi-dimensional feature vector as input. Based on the attention mechanism, we design a malware classification framework **(ACNN)**, which achieves a good classification result. We also make a meaningful exploration of how to apply the attention mechanism to the field of malware classification and prove that the design of the model should be closely combined with the characteristics of the data.

The inadequacy is that the application of the attention mechanism we design may not contain all of them. Other characteristics for malware, such as other resource segments, API information, binary structure, etc, are not considered in our classification model.

VIII. CONFLICTS OF INTERESTS

The authors declare that they have no competing interests.

IX. DATA AVAILABILITY

Research Data Related to this Submission

Title: Microsoft 2015 contest dataset

Repository: malware classification dataset

url: https://www.kaggle.com/c/malware-classification/

REFERENCES

[1] (2019). *360 Internet Security Center*. [Online]. Available: https://zt.360.cn/1101061855.php?dtid=1101062370&did=610142397

[2] (2019). *IDA Pro*. [Online]. Available: https://www.hex-rays.com/products/ida/

[3] (2019). *Microsoft Kaggle 2015 Malware Contest Champion Team Algorithms*. [Online]. Available: https://github.com/bindog/ToyMalwareClassification

[4] D. Carlin, A. Cowan, P. O'Kane, and S. Sezer, "The effects of traditional anti-virus labels on malware detection using dynamic runtime opcodes," *IEEE Access*, vol. 5, pp. 17742–17752, 2017.

[5] D. Gibert, "Convolutional neural networks for malware classification," Univ. Rovira i Virgili, Tarragona, Spain, 2016.

[6] H. Quanwei, "Malicious executables detection based on N-gram system call sequences," Harbin Inst. Technol., Harbin, China, 2009.

[7] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, "Significant permission identification for machine-learning-based Android malware detection," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3216–3225, Jul. 2018.

[8] T. M. Kebede, O. Djaneye-Boundjou, B. N. Narayanan, A. Ralescu, and D. Kapp, "Classification of malware programs using autoencoders based deep learning architecture and its application to the microsoft malware classification challenge (big 2015) dataset," in *Proc. IEEE Nat. Aerosp. Electron. Conf. (NAECON)*, Jun. 2017, pp. 70–75.

[9] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Proc. Australas. Joint Conf. Artif. Intell.*, 2016, pp. 137–149.

[10] L. Liu, B.-S. Wang, B. Yu, and Q.-X. Zhong, "Automatic malware classification and new malware detection using machine learning," *Frontiers Inf. Technol. Electron. Eng.*, vol. 18, no. 9, pp. 1336–1347, Sep. 2017.

[11] T. Messay-Kebede, B. N. Narayanan, and O. Djaneye-Boundjou, "Combination of traditional and deep learning based architectures to overcome class imbalance and its application to malware classification," in *Proc. IEEE Nat. Aerosp. Electron. Conf. (NAECON)*, Jul. 2018, pp. 73–77.

[12] B. N. Narayanan, O. Djaneye-Boundjou, and T. M. Kebede, "Performance analysis of machine learning and pattern recognition algorithms for malware classification," in *Proc. IEEE Nat. Aerosp. Electron. Conf. (NAECON) Ohio Innov. Summit (OIS)*, Jul. 2016, pp. 338–342.

[13] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. 8th Int. Symp. Vis. Cyber Secur.*, Jul. 2011, Art. no. 4.

[14] L. Nataraj, "A signal processing approach to malware analysis," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. California, Santa Barbara, Santa Barbara, CA, USA, 2015.

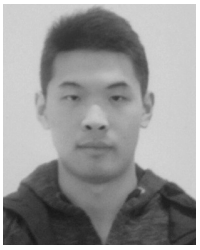
[15] Q. Qian and M. Tang, "Dynamic API call sequence visualisation for malware classification," *IET Inf. Secur.*, vol. 13, no. 4, pp. 367–377, Jul. 2019.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS*, 2017, pp. 1–11.

[17] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based lstm for aspect-level sentiment classification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 606–615.

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

[19] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2016, pp. 1480–1489.



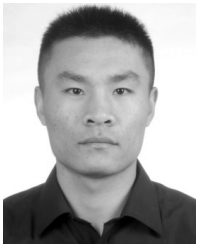
XIN MA is currently pursuing the master's degree in network security with the Army Engineering University of PLA.

SHIZE GUO was born in 1969. He is currently a Professor with the Army Engineering University of PLA, Nanjing, China. He is also with the Command and Control Engineering College, Army Engineering University of PLA. His main research interests include information technology and information security.

HAIYING LI is currently pursuing the Ph.D. degree in network security. His main research interests include information technology and information security.

ZHISONG PAN was born in 1973. He is currently a Professor with the Army Engineering University of PLA, Nanjing, China. He is also with the Command and Control Engineering College, Army Engineering University of PLA. His main research interests include artificial intelligence and network security.

YU DING is currently pursuing the Ph.D. degree with the Army Engineering University of PLA, Nanjing, China. He is also with the Command and Control Engineering College, Army Engineering University of PLA. His main research interests include artificial intelligence and network security.



JUNYANG QIU is currently pursuing the Ph.D. degree with the School of Information Technology, Deakin University, Geelong, VIC, Australia. He is also with the School of Information Technology, Deakin University. His research interests include cyber security and machine learning.

FEIQIONG CHEN received the M.S. degree in computer software and theory from the PLA University of Science and Technology, Nanjing, China, in 2006. She is currently an Assistant Professor with the Army Engineering University of PLA, Nanjing. Her research interests include artificial intelligence and information retrieval.