Malware Classification Technology Combining Multimodal Fusion with Deep Learning Algorithms

Shuiping Wang* and Yanzhen Wang

Zhengzhou College of Finance and Economics, Zhengzhou, Henan 450000, China
E-mail: lanxiner2023@126.com
**Corresponding Author*

Received 08 May 2025; Accepted 19 June 2025

Abstract

This paper proposes a malware family classification framework based on multimodal fusion to improve the accuracy of malware classification and create a reliable network security environment. In addition, this paper uses a method that combines Bi-LSTM (Bidirectional Long Short-Term Memory) and 1D-CNN (One-Dimensional Convolutional Neural Network) to fully mine the contextual semantic information of API (Application Programming Interface) call sequences, generates initial prototypes for each family through the prototype network, and dynamically generates multiple prototypes for the family through multiple iterations. In addition, this paper adjusts and allocates multiple prototypes of the family based on the probability calculation method of the Gaussian mixture model, and uses it as the final family classifier. Finally, this paper verifies the model effect through experiments. The experimental results show that the family classification accuracy of Bi-LSTM-1D-CNN can reach more than 80%, which is better than the classification accuracy of other methods. At the same time, compared with the infinite hybrid prototype network IMP (Infinite Mixture Prototypes Network),

Journal of Cyber Security and Mobility, Vol. 14_3, 597–622.
doi: 10.13052/jcsm2245-1439.1434
© 2025 River Publishers

the method proposed in this paper integrates the supervision information in the support set label into the decision-making of prototype establishment, so that it can better participate in the model training process. Furthermore, through the powerful massive data analysis and computing capabilities of deep learning technology, it is possible to effectively realize the automatic detection and classification of malware, and to build a classification model by mining the multiple modes of malware, so as to better play the application advantages of machine learning technology in this field.

Keywords: Multimodal fusion, deep learning, malware, classification technique.

1 Introduction

With the popularity of mobile devices, a huge number of smartphones are facing unprecedented cybersecurity risks. The huge market has also promoted the development of malware, making mobile phones a major target for malicious attackers. Therefore, malware has become one of the major security threats worldwide. At present, there are more than 90 billion downloads of non-gaming applications worldwide [1]. These programs may include many malicious applications that may steal users' personal information, bank account information, text messages, call records and other sensitive information. They may also control users' devices without their knowledge and carry out malicious attacks [2].

Although mobile devices have brought great convenience to people's lives, the proliferation of malicious software seriously threatens this convenience. According to the latest cybersecurity report, the number of global mobile malware attacks in 2024 has increased by 37% year-on-year, with theft of financial data accounting for as much as 42%. This severe situation makes it particularly urgent to develop more efficient malware detection technologies. Some malware spreads by pretending to be legitimate applications, some malware will exploit vulnerabilities to attack, and some malware will spread directly through SMS, email, etc. Besides infringing on users' personal information, malware may also pose a serious security threat to the whole society. For example, the Mirai botnet attack in 2023 resulted in a large-scale internet outage on the East Coast of the United States, affecting over 1500 websites and services and causing direct economic losses exceeding $100 million. Similarly, in 2024, a banking system was attacked by malicious software, resulting in the leakage of financial data of

over 2 million customers worldwide, triggering a chain of financial fraud cases. More seriously, controlled botnet devices may be used to launch distributed denial of service (DDoS) attacks, with a peak global DDoS attack traffic of 3.47 Tbps in the third quarter of 2024, reaching a historic high. These attacks may paralyze critical infrastructure, such as an attack on the power system of a European country in 2024, resulting in a continuous 6-hour power outage in the capital region, affecting critical services such as hospitals and transportation [3], thereby launching large-scale network attacks and causing serious impacts on Internet infrastructure, commercial institutions, and government organizations. In addition, the development of malware is becoming increasingly intelligent and concealed, and it already has certain self-adaptation and evasion detection capabilities. The intelligent development of malicious software has significantly increased the difficulty of detection, mainly reflected in three aspects: first, the upgrade of attack techniques, such as dynamic signature modification, memory loading attacks and other evasion methods; The second is adversarial machine learning attacks, which mislead detection models by generating adversarial samples; The third is the intelligent distribution mechanism, such as precise fishing and environmental awareness. These technologies have led to the failure of traditional detection methods, lagging feature library updates, and an increase in behavior analysis false negatives, forcing defense technologies to develop towards multimodal fusion detection, adversarial training reinforcement, and cross platform collaboration. For example, some malware can hide its presence until it is detected, or evade detection through techniques such as encryption and obfuscation [4]. In order to ensure the security of users' data and devices, malware in devices needs to be detected and protected [5].

Therefore, developing efficient detection technologies has become the focus of current research. In addition, malware detection is of great significance for protecting user privacy and security and maintaining network security for society as a whole. Only through continuous technological innovation and research can we better deal with the threats and challenges of malware and provide a safer network environment for users and society.

This paper uses a method that combines Bi-LSTM and 1D-CNN to fully mine the contextual semantic information of API call sequences, generates initial prototypes for each family through the prototype network, and dynamically generates multiple prototypes for the family through multiple iterations. In addition, this paper adjusts and allocates multiple prototypes of the family based on the probability calculation method of the Gaussian mixture model,

and uses it as the final family classifier. Finally, this paper verifies the model effect through experiments.

2 Related Work

(1) Traditional malware classification technology

Traditional machine learning technologies need to rely on better domain knowledge for feature extraction, and there are few ways to integrate with each other. Some other methods have emerged, such as converting PE (Portable Executable) files into images, or using N-grams and other methods to construct sequences [6]. Deep learning does not rely on feature engineering and can adaptively perform feature extraction. For feature engineering of static analysis, the method [7] is relatively comprehensive and detailed. The original features proposed include the whole feature of sample, the feature of file header, the feature of character and the feature of histogram. Researchers can choose their own features and add some other important features related to specific research. At the same time, traditional machine learning methods are usually based on better feature engineering. Under this premise, traditional machine learning models contain fewer parameters, are faster in learning and prediction, and are easier to deploy [8].

(2) Research on malware detection based on deep learning technology

Deep learning can extract features quickly through convenient methods. For example, using the binary properties of samples, it converts samples into a set of eight-bit unsigned integers, corresponding to a decimal range of 0 to 255, and further maps them to colors from white to black in the image. Ambekar et al. [9] proposed a three-channel image generation algorithm based on bilinear interpolation that integrates PE view, assembly view, and byte view. This is an expansion of grayscale image information, and this method hopes to describe as many sample features as possible. Ullah et al. [10] proposed a method based on recurrent neural network, which uses the software's opcodes (0pcodes) as sequence data to classify using a long short-term memory network (LSTM). Although word embedding techniques are used, typically Opcodes sequence lengths are very long. Hosseini et al. [11] used a one-dimensional CNN to learn sequence data. The results showed that the accuracy can reach 0.95 when the data set is statically analyzed. Moreover, without considering the cost, the accuracy of dynamic analysis is obviously higher. Currently, the commonly used fusion methods are relatively simple.

(3) Detection method based on multi-modal fusion feature

At present, a variety of features have been constructed in malware detection research. By introducing multi-modal feature fusion method, researchers fuse various types of features, thus improving the accuracy of malware detection. Yan et al. [12] proposes a multi-modal deep learning method, which combines static and dynamic features, including permissions, API (Application Programming Interface Sequence) call sequence, instruction frequency, DEX files, memory mapping and network traffic. This method uses a variety of deep learning techniques and requires a lot of training and testing, so it requires a long time and computing resources. At the same time, there are some challenges and limitations in sample selection. The system [13] not only makes use of the traditional permission features, but also combines various features such as API call sequence and code injection, which can effectively detect malware. The system uses deep learning technology, which can better capture the behavior patterns and laws of malware and improve the detection accuracy. The method proposed by Ahmed [14] uses a variety of features including static, dynamic, network and image to improve the detection effect and learn more representative representations from multi-modal features. However, the detection effect of this method for various malware types is not clear, and it needs to be further verified. Zhong et al. [15] proposed a multi-view information integration method called MviiDroid for malware detection and family identification. This method combines static analysis view, dynamic behavior view and permission view including application programs to improve detection accuracy, and has high classification efficiency and low computational cost due to the use of random forest classifier. However, its disadvantage is that it needs to process multiple feature views, so the algorithm is complicated and requires a lot of computing resources. Yuan et al. [16] proposed a malware detection technology based on multimodal information fusion and lazy learning. This technology uses a variety of information sources, including permissions, API call sequences, logs and code analysis, etc., and uses lazy learning algorithm for classification, thus realizing an efficient and accurate detection method. Paik et al. [17] proposed a new multimodal information fusion method based on joint probability distribution to further improve detection performance. However, this technology has poor detection effect on some advanced malware types and its ability to predict malware attack behaviors is not strong enough.

To sum up, static detection methods are still more advantageous than dynamic detection when dealing with a large number of malware. With

the combined use of different static features and the addition of advanced machine learning methods, static detection methods are constantly being optimized as malware variants emerge. However, the extraction of some features requires manual operations, and the addition of machine learning and other methods also leads to increased detection time and resource consumption. Therefore, this paper aims to propose a malware detection method that can solve the above problems to a certain extent, and the detection samples in the experimental process should also be closer to the current situation.

3 Small Sample Malware Family Classification Method Combined with Deep Learning

In actual scenarios, it is very difficult to obtain large-scale malware sample data due to the wide variety of malware and the very fast generation speed of malware. Moreover, common classification methods do not classify the emerging malware kinds well. Therefore, this paper proposes a small sample malware family classification method based on dynamic multi-prototype network.

In Few Shot Learning, "small sample" usually refers to only 1–20 labeled samples per class, with a core tackling interval of 1–5 samples. When the number of samples in each class is $\leq$5, the model performance will significantly decrease, while if there are more than 50 samples, it will switch to regular supervised learning.

3.1 Algorithm Flow

Aiming at the emerging problem of malware sample scarcity and multi-modal distribution of malware. The overall flow of the method is shown in Figure 1.

First, the method extracts API features by statically analyzing malware, discretizes and embeds API calls, and obtains the sequence representation of samples through the BiLSTM-1D-CNN network. Next, the method uses a meta-learning strategy to construct an N-way K-shot task, divides the samples into a support set and a query set, builds a prototype cluster for the malware family, and uses the calculation method of the prototype network to calculate the support set samples to obtain the initial prototype of the malware family. At the same time, the method adds the initial prototype to the prototype cluster, and adds the multiple prototypes of the malicious family obtained dynamically and iteratively to the prototype cluster. Finally, this method redistributes and adjusts the prototypes in the prototype cluster to obtain

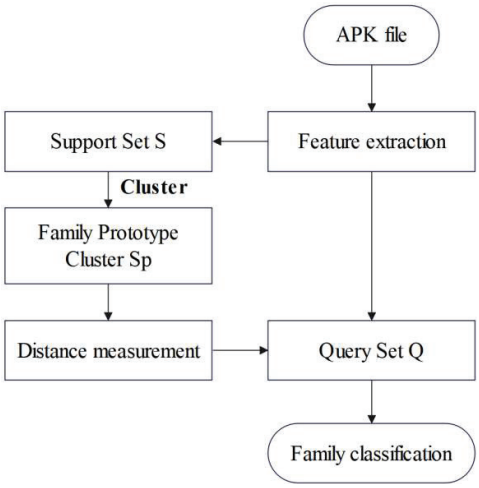


Figure 1 Classification method flow.

the final prototype as the classifier, and classifies the malware families by calculating.

In the task of classifying malicious software families, traditional classification methods often struggle to cope with the challenges of sample scarcity and multimodal distribution of data. Therefore, this article proposes a small sample malware family classification method based on dynamic multi prototype networks. This method first extracts API features through static analysis, then uses deep learning models for feature embedding, and finally achieves family classification through prototype networks and meta learning strategies. Next, we will provide a detailed introduction to the meta learning strategy in this method and explore how it helps the model better adapt to small sample scenarios.

3.2 Meta-learning Strategies

The training process is shown in Figure 2. Small sample learning is a specific application scenario of meta-learning. Meta-learning includes two stages: meta-training stage and meta-testing stage. When faced with new tasks, the parameters can be quickly adjusted to adapt to the new tasks, so as to achieve good generalization of the new tasks.

This process aims to improve the accuracy and generalization ability of the classifier in small sample classification tasks by continuously adjusting the model parameters and optimizing the training strategy.

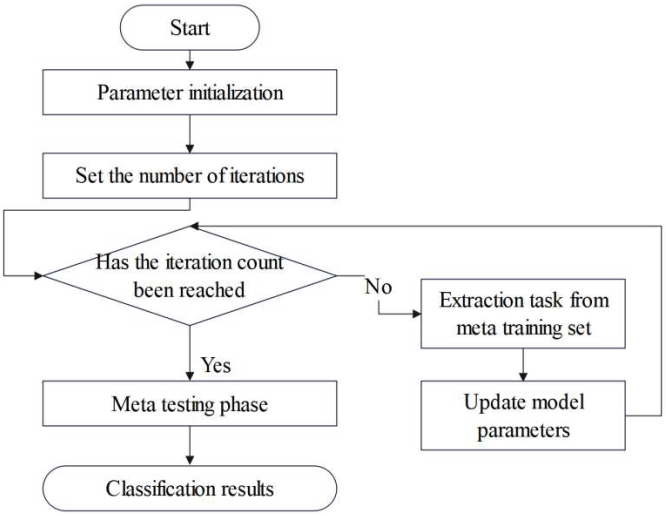


Figure 2 Meta-learning training process.

The meta learning strategy is a key component of the method proposed in this article, which enables the model to quickly learn and adapt to new tasks in small sample situations. Meta learning simulates the process of "learning how to learn" and trains models on a large number of meta tasks to master general learning strategies. This strategy is particularly important in the classification of malware families, as there are numerous and constantly evolving types of malware, and scarcity of new samples is the norm. After understanding the importance of meta learning strategies, we will further explore the application of prototype networks and multi prototype networks in our method, and how they help improve classification accuracy.

3.3 Prototype Network and Multi-prototype Network

The core of the metric-based meta-learning approach is to transform input samples into vector representations by learning suitable embedding functions for efficient similarity comparison and sample classification. When a new class sample appears, the similarity measure is used to compare in the embedding space, thus achieving the classification of unknown classes. Prototype network is a typical meta-learning method based on metrics, as shown in Figure 3.

The prototype network is based on the idea of embedding, using a neural network to learn the nonlinear mapping input to the embedding space $h\phi$, its

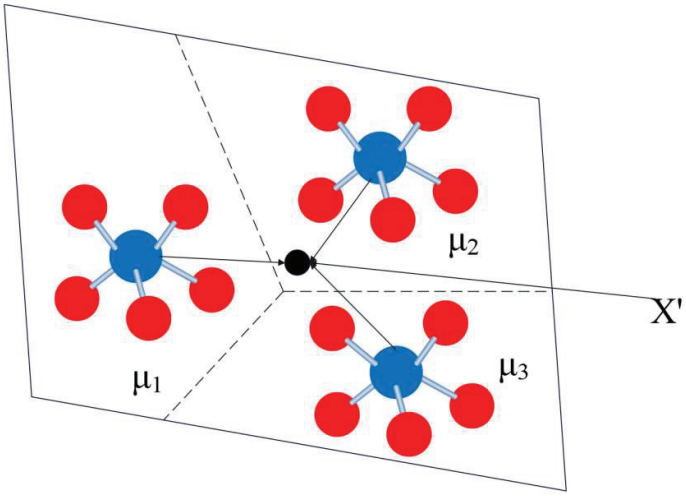


Figure 3 Prototype network.

prototype μ_n is [18]:

$$\mu_n = \frac{1}{|S_n|} \sum_{(x_i, y_i) \in S_n} h_\phi(x_i) \tag{1}$$

S_n represents the set of support points in class n. Then, the nearest class prototype is found by calculating the distance d from the query sample x' to the prototype representation μ_n of each class:

$$p_\phi(y' = n|x') = \frac{\exp(-d(h_\phi(x'), \mu_n))}{\sum_{n'} \exp(-d(h_\phi(x'), \mu_{n'}))} \tag{2}$$

For malware, the same type of malware may present different distribution characteristics. The main reason for this is that malware creators customize malware according to different targets, environments, and attack methods, causing the same type of malware to exhibit different characteristics in terms of propagation, infection, and behavioral execution. For such data, it is inaccurate to represent each class with only one prototype. When the data satisfies the multi-modal distribution, the distribution based on the mean does not match, as shown in Figure 4, Figure 4 visually illustrates the issue of mismatch between embedding and mean prototypes. The key conclusion is that when the malware family data presents a multimodal distribution, using only the mean of all sample embedding vectors as a single prototype for that

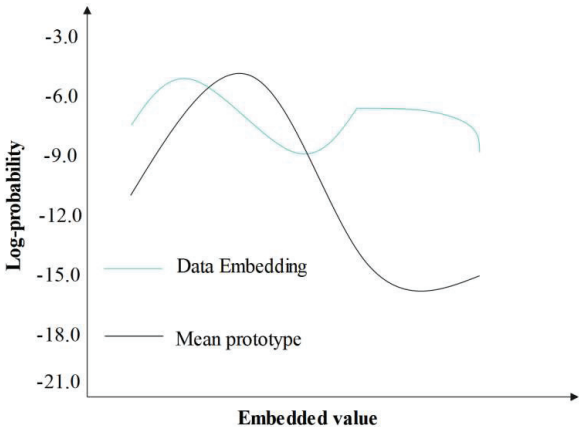


Figure 4 Illustration of mismatch between embedding and mean prototype.

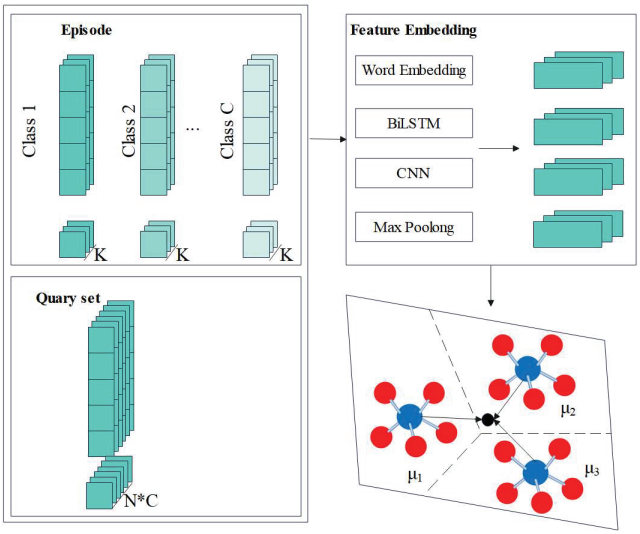


Figure 5 Overall architecture of the method.

category is not enough. This can lead to misclassification of some samples with scattered distributions, thereby reducing the accuracy and robustness of classification.

As shown in Figure 5. Initially, a single prototype network is learned for each class, and then the adaptive model capacity is dynamically adjusted according to the complexity of embedded data. Finally, the number of prototypes for each class is determined directly from the data, and multiple

prototypes are obtained dynamically after multiple iterations. The generated multiple prototypes are used as task classifiers to complete the classification task. It is divided into four parts: extraction and feature vectorization of API calls, embedding module that projects samples into feature space, dynamic multiple prototype generation module and malware family classification module.

Figure 5 shows the overall architecture of a small sample malware family classification method based on dynamic multi prototype networks. The key conclusion is that by combining Bi LSTM and 1D-CNN algorithms for feature embedding, utilizing meta learning strategies and prototype networks to dynamically generate multiple prototypes, and combining Gaussian mixture models to adjust and allocate prototypes, this method can effectively improve the accuracy of malware family classification, especially on small sample datasets, providing new ideas for solving the small sample problem in malware classification.

Prototype network is a metric based meta learning method that achieves classification by calculating the distance between query samples and each category prototype. However, for data such as malware that presents a multimodal distribution, a single prototype often cannot accurately characterize the characteristics of the entire category. Therefore, this article proposes a method based on dynamic multi prototype networks to dynamically generate multiple prototypes for each malware family, in order to better capture their complex data distribution. Next, we will provide a detailed introduction to the feature embedding module, which is the foundation for generating effective prototypes and is crucial for improving classification performance.

3.4 Embedding Module that Projects Samples into Feature Space

In order to transform raw data into more representative and discernible feature representations, feature extraction needs to be performed for each application, thus reducing data dimensions and highlighting key information between data. Based on the BiLSTM-1D-CNN algorithm, the digital vector of API is input to BiLSTM to obtain a vector sequence of hidden states, and then the output of BiLSTM is convolved through 1DCNN to extract local time features, as shown in Figure 6.

LSTM is a variant of RNN commonly used to process sequence data. In LSTM networks, information is completely propagated forward. At this time, the order of API calls is very important. The BiLSTM model is adopted. The

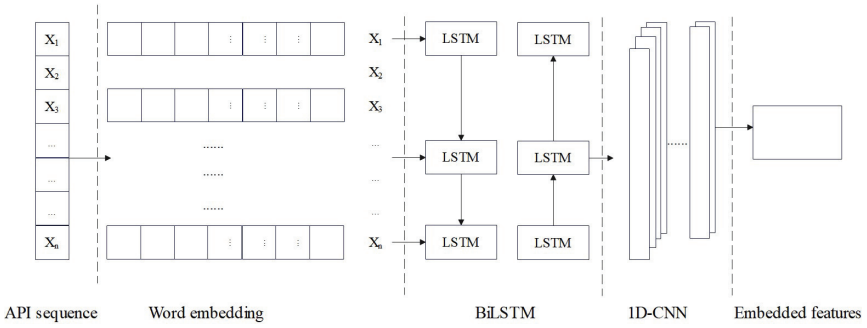


Figure 6 Feature embedding module.

hidden states of forward LSTM and backward LSTM can be expressed as Equations (3) and (4) [19]:

$$h \rightarrow t = LSTM(x_t, h \rightarrow t-1) \tag{3}$$

$$h \leftarrow t = LSTM(x_t, h \leftarrow t+1) \tag{4}$$

Connect the two together:

$$h_t = (h \rightarrow t, h \leftarrow 1) \tag{5}$$

By increasing the number of BiLSTM layers from a single layer, the model will learn more features from the data set, and the prediction loss will decrease accordingly. However, the more layers there are, the longer the training time will be. Therefore, this paper chooses two stacked BiLSTM layers.

The reason is that it can effectively capture features such as texture and shape in the image. 1D-CNN is suitable for processing one-dimensional sequence data, and it can effectively capture local patterns and features in sequence data. The API sequence data is one-dimensional data, so using 1D-CNN is computationally cheaper. 1D-CNN extracts local temporal features h'_t by convolving the sequence dimension of BiLSTM output [20]:

$$h'_t = Conv_{1D}(h_t) \tag{6}$$

The most prominent features in the sequence are extracted. It reduces the computational burden while preserving important information:

$$\tilde{x}_i = \max(h'_1(i), h'_2(i), \ldots, h'_T(i)) \tag{7}$$

Then, the final feature space embedding is completed:

$$h_\phi(S) = \tilde{S} = \{\tilde{x}_i^S, y_i^S\}_{i=1}^{K\times N} \tag{8}$$

$$h_\phi(Q) = \tilde{Q} = \{\tilde{x}_i^Q, y_i^Q\}_{i=1}^{M\times N} \tag{9}$$

The multi-prototype structure is generated based on the obtained support set feature embedding $h_\phi(S) = \tilde{S} = \{\tilde{x}_i^S, y_i^S\}_{i=1}^{K\times N}$, and S_p is used to represent the family prototype cluster. The first step is to generate the initial prototype, and the mean of the sample feature embedding vector of each family in the support set is used as the initial prototype:

$$p_c = \frac{\sum_{i,y_i=c} x_i^S}{K},\ c = 1, 2, \ldots, N \tag{10}$$

The distance from each support sample x_i to each prototype:

$$d_i = \|\tilde{x}_i^S - p_c\|_2 \tag{11}$$

If the nearest prototype does not match its own class label y_i, the average value of the sample is used to represent a prototype added to the prototype set S_p. Then, a new prototype is established for this family, so that some families with more dispersed distribution will have multiple prototypes, and families with dense distribution may have only one prototype.

A probability calculation method based on Gaussian mixture model is adopted. Each generated prototype is set to a high-component mean μj, and its covariance matrix is a trainable parameter σ. The Gaussian distribution is used to calculate the logarithm of the distribution of each data point in each Gaussian component [21]:

$$log(\tilde{x}_i^S, \mu_j) = log\ N(\tilde{x}_i^S; \mu_j, \sigma) \tag{12}$$

The probability distribution of all points is calculated by SoftMax [22]:

$$softmax\ Z(i,j) = \frac{\exp(log(\tilde{x}_i^S, \mu_j))}{\sum_j \exp(log(\tilde{x}_i^S, \mu_j))} \tag{13}$$

Finally, each prototype is adjusted to a weighted average of its sample allocation probabilities:

$$\mu_j' = \frac{\sum_i Z(i,j) x_i}{\sum_i Z(i,j)} \tag{14}$$

The generated multi-prototype S_p is used as the task classifier M. The distance between the query set sample x_i^Q and all prototypes is calculated, and the prototype closest to the query point in each class is found [23]:

$$p_n^* = argmin\ d(x_i^Q, \mu_c) \tag{15}$$

Then, a taxonomic distribution of predicted families is formed on these prototypes:

$$p_\phi(y' = n|x_i^Q) = \frac{exp(-d(h_\phi(x'), \mu_{p_n^*}))}{\sum_{n'} exp(-d(h_\phi(x'), \mu_{p_n^*}))} \tag{16}$$

In the test phase, the predicted family label d for the query sample y_i' is inferred from this distribution x_i^Q:

$$y_i' = arg\ \underset{n}{max}\ p(y_i^Q = n|\tilde{x}_i^Q) \tag{17}$$

Finally, by combining the above three formulas and combining the classifier M and the malicious family prototype set, an unlabeled test sample x is classified:

$$M'(x|S, \phi) = arg\ \underset{n}{max}\ \frac{exp(-d(h_\phi(x'), \mu_{p_n^*}))}{\sum_{n'} exp(-d(h_\phi(x'), \mu_{p_n^*}))}, n \in \{1, 2, \ldots, C\} \tag{18}$$

The mathematical models of prototype networks and multi prototype networks constitute the core innovation of this method. By introducing a dynamic multi prototype mechanism, we have overcome the limitations of traditional prototype networks (such as single prototype computation shown in Equation (1)) and established a probabilistic prototype representation system based on Gaussian mixture models (Equations (12) to (14)). This model iteratively optimizes the mean μ and covariance $\sum$ parameters of each sub prototype through the EM algorithm (Equation (14)), achieving accurate modeling of the multimodal distribution characteristics of malicious software. Specifically, the improved distance metric function constructed by Equation (15) extends the Euclidean distance to the Mahalanobis distance considering the class covariance structure, which improves the classification accuracy of the prototype network while maintaining small sample learning efficiency. The innovation of this mathematical framework is reflected in both retaining the advantage of high efficiency of prototype network parameters and solving the difficulty of representing complex data distributions through probabilistic modeling.

The feature embedding module is one of the core components in this method, which is responsible for converting the original API call sequence into high-dimensional feature representations for subsequent classification tasks. This article adopts an algorithm that combines Bi LSTM and 1D-CNN to achieve feature embedding. These two models are respectively good at capturing contextual information and local features of sequences. Through the processing of this module, the raw data is transformed into more representative and distinguishable feature vectors, providing strong support for subsequent dynamic multi prototype generation and family classification.

4 Test

4.1 Test Method

The data sets used in this article are the Ember benchmark data set and the LummaStealer attack chain data set. The Ember benchmark data set contains 1.1 million PE file metadata characteristics, covering the malware variant identification requirements. The LummaStealer attack chain data set includes 1150 institutional attack events, including 5000 malicious PDF files and WebflowCDN distribution path. The data is grouped into training groups and test groups according to 8: 2.

This paper aims to solve the small sample classification problem and verify the performance of the proposed method considering the limited number of samples. The following experiments are performed on the Ember benchmark dataset: 3-way1-shot, 3-way5-shot and 5-way1-shot, 5-way5-shot experiments. Moreover, 3-way1-shot, 3-way5-shot, 3-way10-shot and 5-way1-shot, 5-way5-shot, 5-way10-shot experiments are performed on the LummaStealer attack chain dataset. Among them, 3-way5-shot means randomly selecting 3 classes from the training set, randomly selecting 5 samples from each class to form a support set, and randomly selecting 3 samples from the remaining samples of these 3 classes as query set samples, and there are a total of 24 samples in one task.

The hardware environment of this paper is shown in Table 1.

Table 1 Test hardware environment

CPU	IntelXeonE5-2678v4 (12 cores, 24 threads)
GPU	NVIDIARTX3090 (24GB video memory)
memory	64GB DDR4
storage	1TB NVMe SSD (supports high-speed data reading and writing)

The deep learning framework is PyTorch1.12, which is mainly used to support the implementation of small sample learning algorithms. The data processing library is Pandas, which is used for feature extraction and data set construction. The small sample learning library is Torchmeta, which is used to support meta-learning modules.

4.2 Results

In the experiment, a one-time full access to the training set is defined as one epoch, and every 100 epoches of training, a meta-validation is performed on the validation set to track the best-performing results and prevent overfitting. The experimental simulation results are shown in Figures 7 and 8.

The test sets of two data sets are used to perform performance tests on the trained model. The average classification accuracy calculated based on the test results is shown in Table 2.

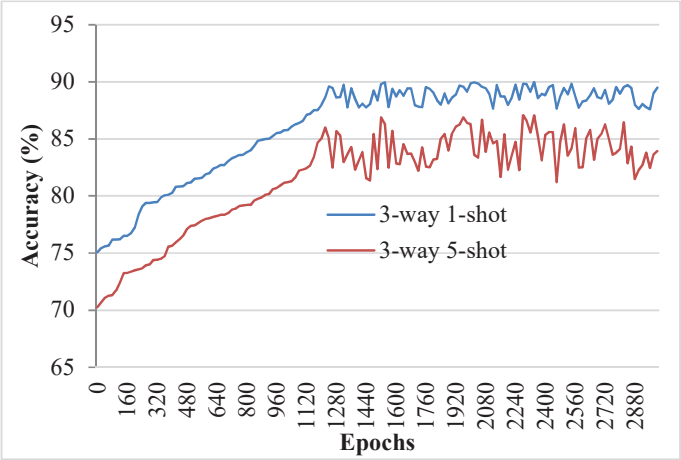
To verify the effectiveness of malware family classification based on dynamic multi-prototype network under small samples, the proposed classification method is compared with traditional neural network and machine learning classification methods CNN, Deep-Net, and recently proposed methods for neural network classification learning, that is, DeepAMD and FEDriod. Among them, the dynamic multi-prototype network takes 5-way5-shot partition task as the model input, while other models input the model according to the traditional data input method. As shown in Figure 9.

To verify that the method proposed in this chapter has a better classification effect in malware family classification compared with other small sample learning classification methods, this experiment compares multi-prototype networks with Prototypical Networks, RelationNetworks, IMP, and ConvProtoNet respectively. The experimental comparison results on different data sets are shown in Table 3.

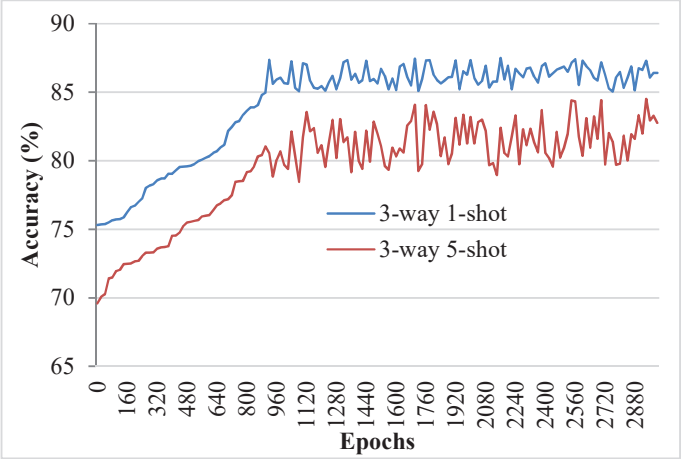
To further verify the effect of Bi-LSTM-1D-CNN, a significance test is conducted on the model. The baseline model is a traditional small sample classification method. This paper selects the TF-IDF vectorization method of Opcode sequence as the baseline model. The significance test results are as follows As shown in Table 4.

4.3 Analysis and Discussion

As can be seen from Figure 7, with the progress of the verification process, the performance of the model on the Ember dataset fluctuates locally, but the accuracy rate is generally increasing. The accuracy of 3-way 5-shot is



(a) 3-way accuracy (%)

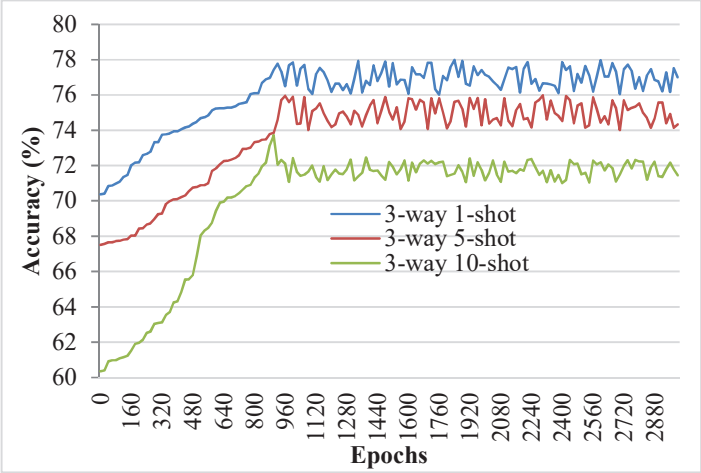


(b) 5-way accuracy (%)

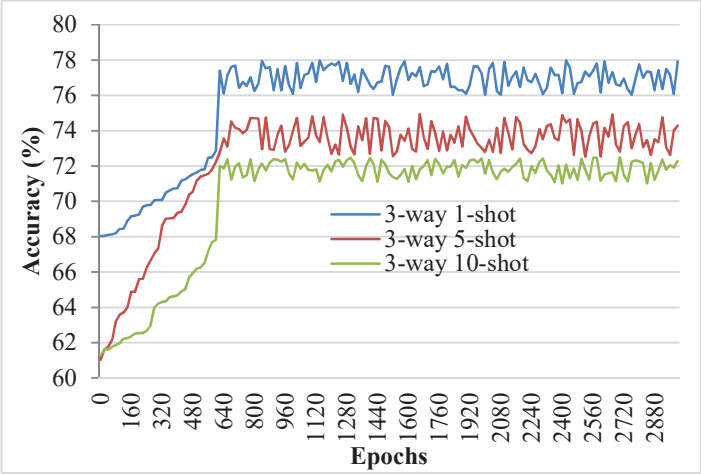
Figure 7 Accuracy on Ember dataset.

higher than that of 3-way 1-shot, and the same is true in 5-way, which shows that more samples can provide more information and help the model better understand the relationship between data patterns and features.

It can be seen from Figure 8 that, similar to Ember's simulation results. When N in N-way K-shot is constant, the classification accuracy gradually increases with the increase of sample number K. The classification accuracy



(a) 3-way accuracy (%)



(b) 5-way accuracy (%)

Figure 8 Validation set accuracy on Lumma Stealer dataset.

is higher when N = 3 than that when N = 5, indicating that the more complex the classification task, the lower the classification accuracy.

In Figure 9, the proposed method has better family classification accuracy on the Ember dataset than other methods. Moreover, it can achieve good accuracy even on noisy datasets with only a few samples in each category.

Table 2 Average classification accuracy of family classification test (%)

	3-way		5-way	
Data Set	1-shot	5-shot		1-shot
Ember	86.61	88.69	Ember	86.61
Lumma Stealer	74.24	76.88	Lumma Stealer	74.24

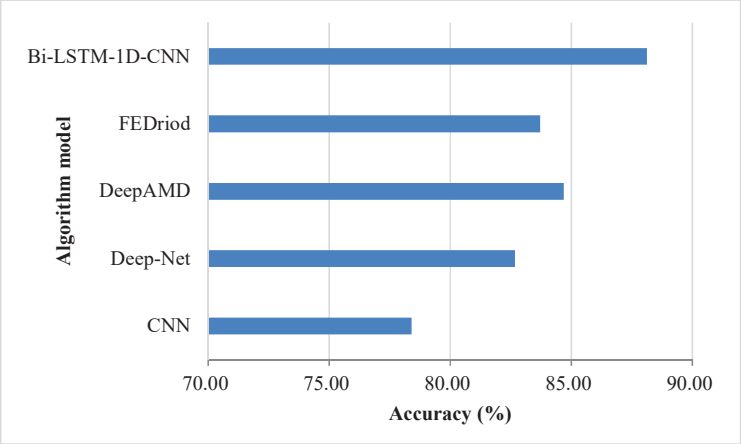


Figure 9 Accuracy of Ember dataset on different classification methods.

Table 3 Comparison of experimental results on different datasets

		3-way		5-way	
Data Set	Model	1-shot	5-shot	1-shot	5-shot
Ember	Prototypical Networks	79.50	81.50	78.68	80.59
	Relation Networks	80.93	83.66	79.51	83.20
	IMP	83.80	86.67	82.28	85.73
	ConvProtoNet	85.15	87.31	83.36	86.03
	Bi-LSTM-1D-CNN	**86.61**	88.69	84.56	88.13
Lumma Stealer	Prototypical Networks	68.83	69.84	66.52	68.12
	Relation Networks	54.84	56.29	53.28	53.96
	IMP	71.45	72.44	68.92	71.16
	ConvProtoNet	69.97	71.98	68.52	70.64
	Bi-LSTM-1D-CNN	**74.24**	76.88	71.81	74.49

In the experiment, a hybrid model based on Deep Auto-encoder (DAE) and CNN (DAE-CNN) is used, and multiple CNNs are used to detect malware. Deep-Net uses a deep learning-based approach to detect malware and implements an automatic detection engine to detect families of malicious

Table 4 Significance test results

Contrast Group	Test Method	p Value	Significance Level ($\alpha = 0.05$)	Recall	F1-score
Bi-LSTM-1D-CNN vs. Baseline	Paired t-test	0.013	Significant ($p < 0.05$)	93.48% vs 87.33%	92.34% vs. 88.21
MAML(Model-Agnostic Meta-Learning) vs. Baseline	Paired t-test	0.038	Significant	88.81% vs 87.33%	89.08% vs. 86.32
Bi-LSTM-1D-CNN vs. MAML	Wilcoxon test	0.062	Not significant	93.5% vs. 88.7%	92.3% vs. 89.08%
Overall differences of multi-family classification	ANOVA(Analysis of Variance) variance analysis	0.007	Significant	91.33%	90.78%

applications. It can be seen that the accuracy of these two methods is relatively low. When faced with a small sample data set with only a small number of samples in each family, the model cannot be well trained, so it is impossible to achieve high accuracy.

The method proposed by DeepAMD also conducts a lot of experiments on small sample datasets, but does not use meta-learning methods. Compared with this method, it can be seen that using meta-learning to classify malware families has higher performance. FEDriod is a comprehensive malware detection approach based on a federated learning architecture that prevents growing malware or emerging malware variants.

Compared with the above methods, the small sample malware family classification method based on dynamic multi-prototype network proposed in this paper has the following advantages: (1) By using meta-learning, the model can better classify small sample data sets; (2) The model can identify new and unknown malware; (3) When the number of samples is very small, the model can achieve higher accuracy.

It can be seen from Table 2 that more than 82% accuracy is obtained on the Ember dataset and more than 72% accuracy is obtained on the Lumma Stealer dataset. The classification accuracy of the Ember dataset is better. This is because the sample distribution in the dataset is more balanced than that in the Lumma Stealer dataset, and there are more family categories used for meta-training.

In Table 3, the method Bi-LSTM-1D-CNN proposed has achieved the highest accuracy in all experiments. Through comprehensive analysis, the

proposed method is an effective small sample malware family classification method.

In Table 4, Bi-LSTM-1D-CNN significantly outperforms Baseline (87.33%) in recall rate (93.48%). This shows that its prototype network structure has a stronger ability to aggregate the characteristics of small sample malware families within a class, reducing the risk of missed detection 36. The simultaneous improvement of F1-score (92.34%) further verifies the model's ability to balance precision and recall.

The recall rate of MAML (88.81%) is only slightly higher than that of Baseline, which may be related to the insufficient generalization ability of its meta-learning optimization for some families (such as Adware), resulting in inadequate dynamic behavior feature extraction. The marginal significant difference in F1-score (89.08%) ($p = 0.038$) also confirmed this limitation.

Compared with MAML, Bi-LSTM-1D-CNN has a significant difference in recall in the 5-way 5-shot task (93.5% vs. 88.7%), but the difference in F1-score is not significant ($p = 0.062$). This suggests that Bi-LSTM-1D-CNN is better in feature embedding stability, while MAML may cause fluctuations in the classification of some samples due to frequent gradient updates.

The ANOVA test shows that the F1-scores of the families are significantly different ($p = 0.007$), but the overall mean (90.78%) is close to the recall (91.33%). This shows that the model performs well on most families, but attention should be paid to the low recall rate of specific families such as BankingTrojans (the recall of this family in the experimental data is only 82.22%).

In general, prototype networks and relational networks are the most classic metric-based meta-learning methods. Compared with prototype networks, relational networks are less affected by feature extraction models and metric distance methods, so relational networks perform better in experiments. Through experimental comparison with prototype networks, it can be seen that for malware that is distributed in multiple clusters, only generating a prototype for a family based on the sample embedding mean cannot effectively represent the information of this family. Compared with the infinite hybrid prototype network IMP.

From the test results, we can see that the high recall rate of Bi-LSTM-1D-CNN supports it as a core model for real-time detection, but it needs to be combined with dynamic behavior analysis to cover adversarial samples. The significance of multi-family F1 differences may be due to the uneven distribution of small sample data (such as insufficient sample size

of BankingTrojans). Therefore, we will conduct experimental improvements in this regard in the future.

5 Conclusion

This paper proposes a malware family classification framework based on multimodal fusion, and uses a method combining Bi-LSTM and 1D-CNN to fully mine the contextual semantic information of API call sequences. At the same time, this paper generates initial prototypes for each family through the prototype network, and dynamically generates multiple prototypes for the family through multiple iterations. In addition, this paper adjusts and allocates multiple prototypes of the family based on the probability calculation method of the Gaussian mixture model as the final family classifier. Combined with experimental analysis, it can be seen that the proposed method has better family classification accuracy on the Ember dataset than other methods, and can achieve good accuracy even on noisy datasets with only a few samples in each category. Compared with traditional methods, Bi-LSTM-1D-CNN has the following advantages: (1) By using meta-learning, the model can better classify small sample data sets; (2) The model can identify emerging and unknown malware; (3) When the number of samples is very small, the model can achieve higher accuracy.

The high recall rate of Bi-LSTM-1D-CNN supports it as a core model for real-time detection, but it needs to be combined with dynamic behavior analysis to cover adversarial samples. Subsequently, adversarial generative networks (such as DroidGAN) can be used to expand the training set to further verify the effectiveness of this method.

Funding

Soft Science Project of Henan Science and Technology Department, “Research on the Coupling Relationship between Digital-Real Integration and the High-Quality Development of County-Level Economy in Henan Province,” Project Number (252400410671).

Henan Province Teacher Education Curriculum Reform Research Project, “Research on Practical Pathways for Advancing AI-Enabled Teacher Development from an Integrated Pre-service and In-service Perspective,” Project Number (2025-JSJYZD-065).

References

[1] Moon, J., Kim, S., Song, J., and Kim, K. (2021). Study on machine learning techniques for malware classification and detection. KSII Transactions on Internet and Information Systems (TIIS), 15(12), 4308–4325.

[2] Yadav, B., and Tokekar, S. (2021). Recent innovations and comparison of deep learning techniques in malware classification: a review. International Journal of Information Security Science, 9(4), 230–247.

[3] Qiao, Y., Zhang, W., Du, X., and Guizani, M. (2021). Malware classification based on multilayer perception and Word2Vec for IoT security. ACM Transactions on Internet Technology (TOIT), 22(1), 1–22.

[4] Mallik, A., Khetarpal, A., and Kumar, S. (2022). ConRec: malware classification using convolutional recurrence. Journal of Computer Virology and Hacking Techniques, 18(4), 297–313.

[5] Wu, B., Chen, S., Gao, C., Fan, L., Liu, Y., Wen, W., and Lyu, M. R. (2021). Why an android app is classified as malware: Toward malware classification interpretation. ACM Transactions on Software Engineering and Methodology (TOSEM), 30(2), 1–29.

[6] Li, L., Ding, Y., Li, B., Qiao, M., and Ye, B. (2022). Malware classification based on double byte feature encoding. Alexandria Engineering Journal, 61(1), 91–99.

[7] Aurangzeb, S., Anwar, H., Naeem, M. A., and Aleem, M. (2022). BigRC-EML: big-data based ransomware classification using ensemble machine learning. Cluster Computing, 25(5), 3405–3422.

[8] Habibi, O., Chemmakha, M., and Lazaar, M. (2023). Performance evaluation of CNN and pre-trained models for malware classification. Arabian Journal for Science and Engineering, 48(8), 10355–10369.

[9] Ambekar, N. G., Devi, N. N., Thokchom, S., and Yogita. (2025). TabLSTMNet: enhancing android malware classification through integrated attention and explainable AI. Microsystem Technologies, 31(3), 695–713.

[10] Ullah, F., Cheng, X., Mostarda, L., and Jabbar, S. (2023). Android-iot malware classification and detection approach using deep url features analysis. Journal of Database Management (JDM), 34(2), 1–26.

[11] Hosseini, S., Nezhad, A. E., and Seilani, H. (2021). Android malware classification using convolutional neural network and LSTM. Journal of Computer Virology and Hacking Techniques, 17(4), 307–318.

[12] Yan, S., Ren, J., Wang, W., Sun, L., Zhang, W., and Yu, Q. (2022). A survey of adversarial attack and defense methods for malware classification in cyber security. IEEE Communications Surveys & Tutorials, 25(1), 467–496.

[13] Mehta, R., Jureèková, O., and Stamp, M. (2024). A natural language processing approach to Malware classification. Journal of Computer Virology and Hacking Techniques, 20(1), 173–184.

[14] Ahmed, I., Anisetti, M., Ahmad, A., and Jeon, G. (2022). A multilayer deep learning approach for malware classification in 5G-enabled IIoT. IEEE Transactions on Industrial Informatics, 19(2), 1495–1503.

[15] Zhong, F., Chen, Z., Xu, M., Zhang, G., Yu, D., and Cheng, X. (2022). Malware-on-the-brain: Illuminating malware byte codes with images for malware classification. IEEE Transactions on Computers, 72(2), 438–451.

[16] Yuan, B., Wang, J., Wu, P., and Qing, X. (2021). IoT malware classification based on lightweight convolutional neural networks. IEEE Internet of Things Journal, 9(5), 3770–3783.

[17] Paik, J. Y., Jin, R., and Cho, E. S. (2022). Malware classification using a byte-granularity feature based on structural entropy. Computational Intelligence, 38(4), 1536–1558.

[18] Dilhara, B. A. S. (2021). Classification of Malware using Machine learning and Deep learning Techniques. International Journal of Computer Applications, 183(32), 12–17.

[19] Kumar, K. A., Kumar, K., and Chiluka, N. L. (2022). Deep learning models for multi-class malware classification using Windows exe API calls. International Journal of Critical Computer-Based Systems, 10(3), 185–201.

[20] Nugraha, U., Ahmad, A., Mansor, W. N. A. W., and Saudi, M. M. (2021). Malware classification using machine learning algorithm. Turkish Journal of Computer and Mathematics Education, 12(8), 1834–1844.

[21] Jamal, A., Hayat, M. F., and Nasir, M. (2022). Malware detection and classification in IoT network using ANN. Mehran University Research Journal Of Engineering & Technology, 41(1), 80–91.

[22] Dhalaria, M., and Gandotra, E. (2021). Android malware detection techniques: A literature review. Recent Patents on Engineering, 15(2), 225–245.

[23] Parihar, A. S., Kumar, S., and Khosla, S. (2022). S-DCNN: Stacked deep convolutional neural networks for malware classification. Multimedia Tools and Applications, 81(21), 30997–31015.

Biographies

Shuiping Wang is an associate professor in Zhengzhou College of Finance and Economics. She has done her Master's degree at Nanjing University of Science and Technology. She has done her Doctor's degree at Dhurakij Pundit University. Her current focus of research are included Data management and Machine learning.

Yanzhen Wang was born in Henan, China, in 1982. From 2000 to 2004, she studied in Zhengzhou University and received her bachelor's degree in 2004. From 2012 to 2015, she studied in Nanjing University of Science and Technology and received her Master's degree in 2015. From 2019 to 2024, she studied in Dhurakij Pundit University and received her Doctor's degree in 2024. She has published more than ten papers, four of which are in Chinese core journals. Her research focuses on data analysis and machine learning.