Malware Classification Using Attention-Based Transductive Learning Network

Liting Deng[1,2], Hui Wen[1,2(✉)], Mingfeng Xin[1,2], Yue Sun[1,2], Limin Sun[1,2], and Hongsong Zhu[1,2]

[1] Beijing Key Laboratory of IOT Information Security Technology, Institute of Information Engineering, CAS, Beijing, China
{dengliting,wenhui,xinmingfeng,sunyue0205,sunlimin,zhuhongsong}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. Malware has now grown up to be one of the most important threats in the internet security. As the number of malware families has increased rapidly, a malware classification model needs to classify the samples from emerging malware families. In real-world environment, the number of malware samples varies greatly with each family and some malware families only have a few samples. Therefore, it is a challenge task to obtain a malware classification model with strong generalization ability by using only a few labeled malware samples in each family. In this paper, we propose an attention-based transductive learning approach to tackle this problem. To extract features from raw malware binaries, our approach first converts them into gray-scale images. After visualization, an embedding function is used to encode the images into feature maps. Then we build an attention-based Gaussian similarity graph to help transduct the label information from well-labeled instances to unknown instances. With end-to-end training, we validate our attention-based transductive learning network on a malware database of 11,236 samples with 30 different malware families. Comparing with state-of-the-art approaches, the experimental results show that our approach achieves a better performance.

Keywords: Malware classification · Tranductive learning · Attention mechanism · Deep learning

1 Introduction

Malware has now become one of the most important threats in the internet security. The cyber threat report from SonicWall shows that there were 10.52 billion malware attacks in 2018, an increasing of 22% from 2017 [1]. This emphasizes the importance of developing robust and efficient approaches to detect as

N. Park et al. (Eds.): SecureComm 2020, LNICST 336, pp. 403–418, 2020.
https://doi.org/10.1007/978-3-030-63095-9_26

well as classify malicious samples. Traditionally, malware analysis methods can be divided into two main categories including static approaches and dynamic approaches. In static approaches, malware is analyzed by signatures without executing. However, the attackers can easily tweak the available malware and create their own versions to bypass the detection. As even a small change of code can cause the change of signature, static approaches would fail to detect these malware variants. In dynamic approaches, malware is analyzed in a controlled environment such as a virtual environment, simulator, and sandbox. Although dynamic approaches can solve the code obfuscation problem by executing, they are time consuming and need significant efforts from security experts with proper experiences.

As the above hand-crafted malware analysis approaches need a lot of effort to extract features, new methods are adopted to improve the efficiency. Since Krizhevsky et al. [2] achieved the first place in ImageNet competition using deep Convolutional Neural Network (CNN), researchers have introduced deep learning methods into almost every field including malware classification. Benefited from deep learning technology, features can be learned automatically and malware classification model can be built without security experts. Raff et al. (2018) [3] embedded the malware raw bytes into fixed length and performed malware classification with both CNN and Recurrent Neural Networks (RNNs). Quan et al. (2018) [4] proposed a Convolutional Neural Network-Bi Long Short-Term Memory (CNN-BiLSTM) architecture. Taking the raw binaries as input, their approach achieved a high accuracy of 98.2% in classifying nine malware families.

Although deep learning-based approaches can efficiently obtain ideal classification results, they need large amounts of samples for training. However, when setting up the malware database, we found the number of samples in different malware families varies greatly. With the rapidly emerging of malware families, it is almost impossible to collect malware samples from all the existed families. Therefore, we introduce a few-shot learning method to learn the classification model with strong generalization ability from few malware samples. In few-shot learning, the training instances are randomly divided into support set and query set. If each of N unique classes contains K labeled instances as the support set, the target few-shot problem is called N-way K-shot [5]. In each episode, the few-shot learning network learns from the labeled instances which are in the support set and predicts the malware families for the unlabeled instances which are in the query set. The concept of few-shot learning (FSL) was proposed by Fei-Fei et al. (2006) [6]. And FSL is also called one-shot learning if there is only one labeled instance in each unique class of the support set. By using three known categories to derive a priority, they developed a Bayesian learning framework and achieved a detection performance of around 70–95% on images from 101 categories. Vinyals et al. (2016) [7] proposed matching networks, which used a cosine similarity matrix following the embedding function to predict classes for the unlabeled points. Snell et al. (2017) [8] proposed prototypical networks, which computed the mean of the support instances of each unique class in the embedding space as class prototype. Then they predicted the labels of query set by finding the nearest prototype for embedded query instances. Sung et al. (2018)

[5] proposed Relation Network (RN) which can label the instances of query set by computing relation scores between the query images and each instance in the support set. Liu et al. (2018) [9] produced Transductive Propagation Network (TPN) to propagate labels from labeled instances to unlabeled instances. Their experiments showed the state-of-the-art results on $miniImageNet$ [10] and $tieredImageNet$ [11].

Considering similar malware samples have similar image textures, Nataraj et al. (2011) [12] transferred the raw malware binaries into gray-scale images and used GIST [13,14] to do a wavelet decomposition. By choosing k-Nearest Neighbors (kNN) with Euclidean distance for classification, their method achieved an accuracy of 98% with 25 malware families, a total of 9,458 samples. Kalash et al. (2018) [15] obtained 98.52% and 99.97% accuracy on Malimg and Microsoft malware datasets by using a CNN based on VGG-16 [16]. Ding et al. (2018) [17] converted the bytecodes exacted from each Android APK file into an image. Of the total 3,962 malware samples in fourteen families, they obtained an accuracy of 94% with four convolutional layers. Inspired by these results, we also visualize the malware binaries into gray-scale images as data preprocessing.

To the best of our knowledge, few researchers have attempted to use few-shot learning methods on malware classification task. Trung et al. (2018) [18] converted the API calls into vectors via word2vec. They proposed a Memory Augmented Neural Network (MANN) [19] with Least Recently Used Access (LRUA) to classify the unlabeled malware. However, their test dataset had only 430 malicious samples in 5 ransom families. Besides, they compared the experimental results with traditional machine learning methods but not with the few-shot learning methods. In this paper, we propose an attention-based transductive learning network, which can learn the label information from the few samples in each malware family. We evaluate the effectiveness of our method with traditional machine learning methods, deep learning methods and some state-of-the-art few-shot learning methods. Our main contributions are:

- We develop an attention-based transductive learning network for malware classification which can propagate information from labeled instances to unlabeled instances through an attention-based Gaussian similarity graph.
- We set up a malware database with 11,236 samples in total 30 malware families.
- We conduct our approach with traditional machine learning, deep learning and few-shot learning experiments on our collected malware database. As our proposed method has a strong generalization ability, the classification accuracies are increased by more than 60% compared with the traditional machine learning methods and 50% compared with the deep learning methods. In addition, comparing with several state-of-the-art few-shot learning methods, we can also achieve the best performance in 5-shot and 10-shot learning experiments.

The rest of this paper is organized as follows. In Sect. 2, we describe our malware classification approach called attention-based transductive learning network in detail. The experiments are presented in Sect. 3. We conclude the whole paper in Sect. 4.

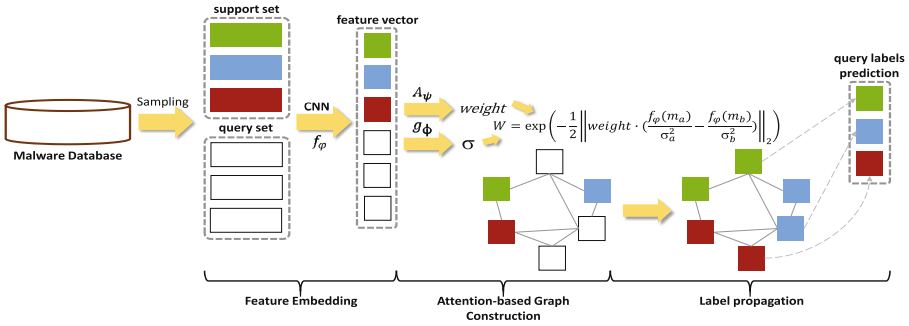

Fig. 1. The overview of the proposed attention-based transductive learning network.

2 Proposed Method

In this section, we will introduce the proposed attention-based transductive learning network, which can be divided into two parts: malware visualization and model learning. With this approach, we can efficiently achieve the malware classification task by just learning a few malicious instances in each family. The overview of the proposed method is shown in Fig. 1.

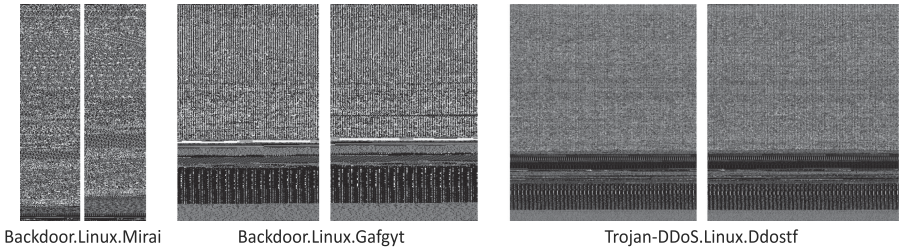

Fig. 2. We can see that different malware samples appear visually similar from a given family and distinct from those belonging to different families. The images are rescaled for better visualization.

2.1 Malware Visualization

We consider this part as the data preprocessing module. Inspired by Nataraj et al. (2011) [12], gray-scale images of different malware samples appear visually similar from a given family and distinct from those belonging to different families. We provide some examples extracted from various families that support this observation in Fig. 2. Therefore, as shown in Fig. 3, we transfer the collected raw malicious binaries into gray-scale images. We first read the binary file in each 8-bits as unsigned integers which can exactly represent the gray value from 0 to 255. As the input of the CNN need to be a fixed size, we then resize every gray-scale images into 64×64. After this data preprocessing module, our raw malware database can be converted into the database with gray-scale images.

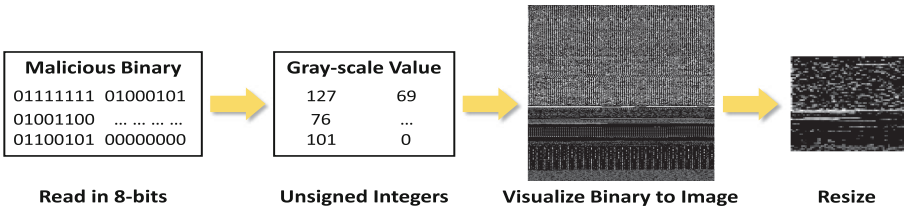

Fig. 3. The process of malware visualization module.

2.2 Model Learning

We illustrate our attention-based transductive learning model in Fig. 1, which consists of three components: feature embedding function with CNNs; Gaussian similarity graph based on attention mechanism; and label propagation algorithm which flows the label information from support set to the query set.

In each episode, we randomly select N malware families with K labeled instances from each of N as the support set $S = \{(m_i, y_i)\}_{i=1}^{l} (l = N \cdot K)$. Similarly, we select T labeled instances in each of these N families for the query set $Q = \{(m_j, y_j)\}_{j=1}^{p} (p = N \cdot T)$. The aim of our model is to learn information from the support set S and minimize the loss between prediction and the query set Q.

Feature Embedding. After malware visualization, we put the 64×64 gray-scale images into feature embedding function f_φ to extract features from the input, where φ represents the parameters of the function. The f_φ includes four convolutional blocks where each block starts with a 2D-convolutional layer with the filter size of 64 and a 3×3 kernel. The convolutional layer is followed by a batch-normalization layer, a ReLU function and a 2×2 2D-Max-Pooling layer (see Fig. 4). Remarkably, we use f_φ to the instances in both support set and query set. So we get $f_\varphi(m_s)$ and $f_\varphi(m_q)$ as the output of the feature embedding function where m_s and m_q represent the images from support set and query set.

Fig. 4. Detailed architecture of each convolutional block in feature embedding function.

Attention-Based Graph Construction. The output $f_\varphi(\cdot)$ of the feature embedding function is a 64-channel 4×4 image. We then flatten $f_\varphi(\cdot)$ and concatenate the feature vectors of support set and query set. In this way, if we set $N = 5$ for selected malware family number, $K = 5$ and $T = 15$ as the number of support and query instances in each of the unique families, we can get the

feature matrix of 100 × 1024. To represent the similarity between each vector of the matrix, we choose a Gaussian similarity formulation for graph construction:

$$W_{ij} = exp\left(-\frac{E(f_\varphi(m_a), f_\varphi(m_b))}{2\sigma^2}\right) \tag{1}$$

where $f_\varphi(m_a)$, $f_\varphi(m_b)$ $(a, b \in [1, (N \cdot K + N \cdot T)])$ represent each vector in the feature matrix, σ is a scale parameter and $E(\cdot, \cdot)$ is defined as a distance computation function. As W_{ij} deeply depends on σ, we need for optimal this parameter to achieve the best performance. Therefore, we use a scale select function g_ϕ to get σ for each feature vector, where ϕ represents the parameters of the function.

With learning an independent σ for each feature vector, the Gaussian similarity formulation can be converted as follows:

$$W_{ij} = exp\left(-\frac{1}{2}E\left(\frac{f_\varphi(m_a)}{\sigma_a^2}, \frac{f_\varphi(m_b)}{\sigma_b^2}\right)\right) \tag{2}$$

For function $E(\cdot, \cdot)$, we simply choose the Euclidean distance formula. Inspired by Woo et al. (2018) [20], different pixels contribute differently to the final classification accuracy. So we need to show the importance of each feature value after extracting them from each malicious instance by $f_\varphi(\cdot)$. Therefore, we introduce an attention mechanism A_ψ to our Gaussian similarity graph, where ψ contains the parameters of the function. The output of A_ψ is a *weight* vector which shapes the same as the feature vector. By using the Euclidean distance and adding the attention mechanism, similarity formulation can finally be defined as follows:

$$W_{ij} = exp\left(-\frac{1}{2}\left\|\boldsymbol{weight} \times \left(\frac{f_\varphi(m_a)}{\sigma_a^2} - \frac{f_\varphi(m_b)}{\sigma_b^2}\right)\right\|_2\right) \tag{3}$$

The detailed structure of the attention mechanism based graph construction module is shown in Fig. 5. In general, the module consists of two main functions which are the scale select function g_ϕ and the attention mechanism function A_ψ. For g_ϕ, it contains a convolutional block, a fully-connected layer, a ReLU non-linear function and a fully-connected layer. And the convolutional block is composed of a 3 × 3 convolutional layer with 1 padding, a batch-normalization

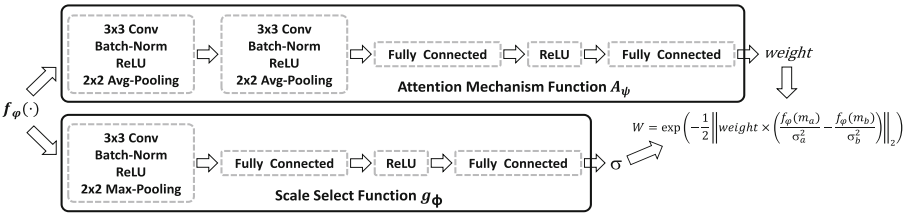

Fig. 5. Detailed architecture of the attention-based Gaussian similarity graph construction module.

layer and a ReLU activation function followed by a 2×2 max-pooling layer. For A_ψ, it contains two convolutional blocks, a fully-connected layer, a ReLU non-linear function and a fully-connected layer. And each of the convolutional blocks is composed of a 3×3 convolutional layer with 1 padding, a batch-normalization layer and a ReLU activation function followed by a 2×2 max-pooling layer. Specially, inspired by kNN that each instance can be represented by its nearest k neighbors, we choose the top k of similarity scores for each instance in W_{ij}.

To accelerate the convergence of our network, we normalize the similarity matrix W_{ij} symmetrically. The formulation of symmetric normalized Laplacian is $L = D^{-1/2}WD^{-1/2}$, where D is defined as the diagonal matrix $D = diag(d_1, d_2, \cdots, d_i, \cdots, d_n)$. Indicating the sum of the i-th row of W, d_i is called the degree of vertex i. Finally, we obtain our attention-based Gaussian similarity graph L, which is a $(N \cdot K + N \cdot T) \times (N \cdot K + N \cdot T)$ symmetric matrix.

Label Propagation Algorithm. In this part, we describe how to get labels of the query set from the support set. The detailed process is shown in Fig. 6.

First, we build a label matrix $Y^L \in \mathbb{R}^{(N \cdot K) \times N}$ for the support set, where N is the number of selected malware families and K represents the instances in each family of the support set. We define $i \in (N \cdot K)$ as the instance number and $j \in N$ as the family number. By using one-hot encoding, $Y^L_{ij} = 1$ if the label of m_i is j and $Y_{ij} = 0$ otherwise, where m_i represents the instance of the support set. Accordingly, we build a label matrix $Y^U \in \mathbb{R}^{(N \cdot T) \times N}$ for the query set, where T represents the instances in each malware family of the query set. As all the instances of the query set are initially unlabeled, we define Y^U as a zero matrix. For convenience, we concatenate the matrix Y^L and Y^U into a new matrix $Y^P \in \mathbb{R}^{(N \cdot K + N \cdot T) \times N}$.

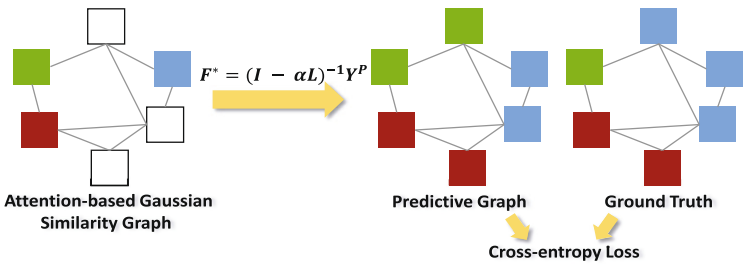

Fig. 6. The detailed process of label propagation algorithm.

According to Zhou et al. (2004) [21], each unlabeled instance in the query set can be labeled in a convergent sequence $\{F(t)\}$ iteratively. $F(0)$ is supposed to be Y^P. We use the formulation as follow to iterate the matrix Y^P:

$$F(t+1) = \alpha L F(t) + (1 - \alpha) Y^P \tag{4}$$

where α is a parameter in the range of 0 to 1, L is the symmetric normalized Laplacian matrix as mentioned above. And $F(t)$ indicates the predictive label matrix in the t-th iteration. Through the Eq. (4), we can get the general formula:

$$F(t) = (\alpha L)^{t-1} Y^P + (1-\alpha) \sum_{n=0}^{t-1} (\alpha L)^i Y^P \tag{5}$$

Considering the ranges of the variables in the formulation, we can get the limit values when t tends to be the positive infinite:

$$\begin{cases} \lim\limits_{t \to +\infty} (\alpha L)^{t-1} = 0, \\ \lim\limits_{t \to +\infty} \sum_{n=0}^{t-1} (\alpha L)^i = (I - \alpha L)^{-1} \end{cases} \tag{6}$$

Therefore, the Eq. (5) can be simplified as follows:

$$F^* = \lim_{t \to +\infty} F(t) = (1-\alpha)(I - \alpha L)^{-1} Y^P \tag{7}$$

where I is the identity matrix. Because $(1-\alpha)$ denotes to a constant value, we use $F^* = (I - \alpha L)^{-1} Y^P$ as the approximate equation. For this reason, we can now directly obtain the predictive label matrix F^* without iterations. And this equation is used to do the label prediction in the actual experimental studies.

After obtaining the predictive label matrix F^*, we choose the position of the max value in each vector as the label of each query instance. Then we use these predictive labels to calculate the accuracy of our network comparing with the ground truth. In addition, the classification loss is computed between the ground truth labels (in both support set and query set) and F^*. The cross-entropy method is chosen as the loss function. By using softmax followed by a logarithmic operator, we transfer the predictive matrix F^* into a probabilistic score matrix.

$$P_{ij} = log \left(\frac{exp(F^*_{ij})}{\sum_{j=1}^{N} exp(F^*_{ij})} \right) \tag{8}$$

Then the whole loss function is computed as follows:

$$J(\varphi, \phi, \psi) = -\frac{1}{N \cdot (K+T)} \sum_{i=1}^{N \cdot (K+T)} \sum_{j=1}^{N} D(gt, j) P_{ij} \tag{9}$$

where gt is the label value of ground truth and $D(\cdot, \cdot)$ is a position function defined in Eq. (10):

$$D(gt, j) = \begin{cases} 1 \ gt = j \\ 0 \ gt \neq j \end{cases} \tag{10}$$

Remarkably, the loss of our learning network is dependent on the parameters φ, ϕ and ψ.

3 Experiments

In this section, we first describe the collected malware database in detail. Then we represent the hyper parameters used in the experiments of our attention-based transductive learning network. Finally, we evaluate our approach with traditional machine learning methods, deep learning methods and state-of-the-art few-shot learning methods on our malware database.

3.1 Malware Dataset

We crawl 11,236 well-labeled malicious samples with 30 categories from several malware information sharing platforms such as MalShare [22], Hybrid-Analysis [23] and VirusSign [24]. After visualizing all the malware binaries, we can obtain

Table 1. Detail information of the malware database.

Malware family	Train samples	Test samples
AdWare.Win32.iBryte	607	261
Downloader.Win32.LMN	240	103
AdWare.Win32.MultiPlug	144	63
Trojan-DDoS.Linux.Ddostf	65	28
Backdoor.Linux.Gafgyt	905	388
Linux.Lightaidra	137	52
Backdoor.Linux.Mirai	108	41
Trojan.Kryptik	158	68
Trojan.MSIL.Crypt	237	102
Trojan.MSIL.Inject	91	40
Trojan.Win32.Generic	147	63
Backdoor.Win32.Androm	55	25
Trojan.JS.Agent	196	84
Trojan.Script.Generic	47	28
Trojan.SuspectCRC	218	94
Trojan.VB.Crypt	79	34
Trojan.VBS.Agent	84	37
AdWare.Win32.Generic	1740	747
Trojan.Win32.Agentb	125	54
Trojan.Win32.Crypt	393	169
Trojan.Win32.Lethic	64	28
Trojan.Win32.Emotet	45	26
Trojan.Win32.Filecoder	60	26
Trojan.Win32.Agent	68	30
Trojan.Win32.MicroFake	58	25
RiskTool.Win32.Generic	60	26
Backdoor.Win32.Hlux	72	31
Trojan.Win32.Ransom	1295	556
Backdoor.Linux.Tsunami	300	129
Trojan-DDoS.Linux	53	27

a database of 11,236 well-sorted gray-scale images. The detail of the database is listed in Table 1. We load our malware database and split it into two sets: 7,851 samples for training and 3,385 samples for testing.

3.2 Experiment Configuration

To make a better comparison with other few-shot learning approaches, we use the same feature embedding function f_φ with the same parameters as other few-shot learning methods [5,8]. In addition, we randomly select N ($N = 5, 10, 20, 30$) families in the total 30 malware families. Refer to the conclusion mentioned by Zhou et al. (2004) [21], the parameter k of top k values for each instance in W_{ij} is set to 20. And α is set to 0.99 [21], which denotes the hyper-parameter of label propagation information. All the instances are randomly generated from the malware database. With end-to-end training, we initialize the learning rate to 10^{-3} and cut in half every 20 epochs while each epoch has 100 batches. Adam optimization is applied for training.

The experiments are based on a 64-bit Ubuntu 16.04 system. We use PyTorch (version 1.2.0) framework to implement our model and NVIDIA Tesla V100 to accelerate the computation. The training model for 5-shot and 10-shot cost 1010.13 s and 1201.37 s while 1-shot costs 847.46 s. Due to the less training time, we can update our model efficiently with new malware families and samples.

3.3 Experimental Results

In this part, we evaluate and verify the effectiveness of our attention-based transductive network with three groups of comparison experiments. We compare the proposed approach with traditional machine learning methods, deep learning methods and some state-of-the-art few-shot learning methods. We treat the gray-scale images as input in all groups of experiments. The best performance in each experimental setting is depicted in bold.

Compare with Traditional Machine Learning Methods. In the first group of experiments, we compare our approach with several traditional machine learning methods. In our approach, we treat the support set as training data and query set as the test data in each episode. We predict the accuracies of query set after training 100 epochs while each epoch has 100 episodes. In each traditional machine learning method, we choose the same number of malware samples as our approach for training and testing. Therefore, we randomly select 5 instances in each unique family for training and 15 instances for testing. To stabilize the accuracies for each of the traditional machine learning methods, we train for 10000 episodes. By choosing N ($N = 5, 10, 20, 30$) in the all 30 malware families, Table 2 describes the results comparing the proposed approach with SVM, kNN, Random Forest (RF) and Decision Tree (DT).

As shown Table 2, our approach reaches the classification accuracies of more than 90% in all the experiments. In addition, even compared with the second

best score in 5-way experiment, the accuracy of our approach is 60% better. This group of experiment shows that our approach has a strong feature extraction ability in multi-class malware classification problem. Comparing with the traditional learning methods, our proposed network can achieve the best performance by learning only a few malware samples.

Table 2. The classification accuracies compared with traditional machine learning methods.

Model	5-way	10-way	20-way	30-way
SVM	59.50%	47.71%	38.64%	34.34%
kNN	50.49%	37.05%	27.47%	23.70%
RF	49.61%	37.81%	29.83%	26.66%
DT	44.92%	34.26%	27.20%	24.55%
Our approach	**95.33%**	**94.63%**	**93.33%**	**92.56%**

Compare with Deep Learning Methods. In the second group of experiments, we compare our approach with two malware classification methods in recent researches using the deep Convolutional Neural Network. Espoir et al. (2017) [25] designed a network with three-layers CNN and two fully connected layers while Gurumayum et al. (2020) [26] built a network with two CNN layers and a fully connected layer. They both got over 97% classification accuracies on their own malware datasets. Therefore, we compare the performance of malware classification between our approach and theirs when training only a few samples. In our approach, we use the same experimental settings as the first group of experiments. For each deep learning method, we also choose the same number of malware samples as our approach for training and testing. In each epoch, we randomly select 5 instances in each unique family for training and 15 instances for testing. And we train the model for 100 episodes in each training step. To stabilize the accuracy, we test 1000 epochs and average the results. By choosing N (N=5,10,20,30) in the all 30 malware families, Table 3 shows the results comparing our approach with two deep learning malware classification methods.

Compared with results in the first group of experiments (Table 2), we can see that the deep learning methods get better accuracies than the traditional machine learning methods. It indicates that the deep learning methods have stronger ability of feature selection. With the increasement of selected malware families N (N=5,10,20,30), our approach shows more robust in multi-class malware classification while the accuracies of deep learning methods decrease obviously. In addition, even compared with the second best score in 5-way experiment, the accuracy of our approach is 50% better. Therefore, our approach can learn the family features more efficiently from a few numbers of malware samples.

Table 3. The classification accuracies compared with deep learning methods.

Model	5-way	10-way	20-way	30-way
Espoir et al. [25]	63.42%	52.28%	41.55%	37.20%
Gurumayum et al. [26]	60.81%	49.71%	38.74%	34.49%
Our approach	**95.33%**	**94.63%**	**93.33%**	**92.56%**

Compare with State-of-the-art Few-Shot Learning Methods. In the third group of experiments, we compare our approach with some state-of-the-art few-shot learning methods in malware classification task. By choosing $N(N = 5, 10, 20, 30)$ categories in the all 30 malware families, Table 4, Table 5 and Table 6 describe the classification accuracies comparing our approach with Matching Networks [7], Relation Networks [5] and Prototypical Networks [8]. We use the common training and testing settings in few-shot classification task. In testing phase, we randomly generated $K + T$ samples in random N families from the testing set. The model parameters are finetuned by K instances and evaluated by the rest T instances in each unique family for in each testing episode.

To make a better comparison with other few-shot learning approaches, we use the same parameters as the other few-shot learning methods [5,8] in Table 4. So the number of support set in each unique malware family is $K = 5$ while the number of query set in each unique family is $T = 15$. From Table 4, We can notice that the proposed method achieves the best results in all the N-way 5-shot experiments. This experiment shows that the proposed attention-based transductive learning network achieves the highest classification accuracies and is more suitable for malware classification task.

Table 4. The classification accuracies compared with state-of-the-art few-shot learning methods ($K = 5$).

Model	5-way	10-way	20-way	30-way
Matching networks [7]	66.98%	59.54%	47.46%	45.07%
Relation networks [5]	67.83%	56.46%	48.41%	44.28%
Prototypical networks [8]	73.46%	61.69%	50.44%	45.76%
Our approach	**75.41%**	**63.15%**	**52.48%**	**50.41%**

Moreover, we also try to set $K = 10$ as the number of support set in each unique malware family while the number of query set in each unique family is still 15. By using more 5 instances for training in each malware family, our approach can still achieve the best performance in all experiments (Table 5). Compared with the results in the N-way 5-shot experiments above (Table 4), we can find that the accuracies of our approach are 2–3% higher. This experiment indicates that our network can achieve a stronger generalization ability by learning from a few more instances in each malware family.

Finally, we also try to extend our approach to N-way 1-shot experiments. By just learning one labeled instance from each selected malware family, we still use $T = 15$ as the number of query set in each unique malware family. As seen from Table 6, our approach can still achieve the best performance in 5-way and 10-way experiments. However, Matching Network [7] performs better than our approach in 20-way and 30-way experiments. These results are explicable because Matching Network [7] is specifically designed for one-shot learning.

Table 5. The classification accuracies compared with state-of-the-art few-shot learning methods ($K = 10$).

Model	5-way	10-way	20-way	30-way
Matching networks [7]	68.94%	60.52%	48.80%	45.41%
Relation networks [5]	73.84%	59.92%	50.68%	44.49%
Prototypical networks [8]	76.57%	64.06%	53.74%	48.14%
Our approach	**77.95%**	**66.73%**	**55.53%**	**51.39%**

Table 6. The classification accuracies compared with state-of-the-art few-shot learning methods ($K = 1$).

Model	5-way	10-way	20-way	30-way
Matching networks [7]	60.22%	52.64%	**50.61%**	**49.28%**
Relation networks [5]	64.43%	51.84%	42.38%	36.55%
Prototypical networks [8]	63.97%	55.26%	43.83%	38.63%
Our approach	**66.89%**	**55.67%**	45.14%	39.83%

With the increasing of epochs, we show the curve of classification accuracy (Fig. 7) and loss value (Fig. 8) for each approach in 5-way 5-shot. As shown in Fig. 7, our approach can finally achieve the best performance after 100 epochs while each epoch contains 100 batches. For Fig. 8, we choose cross-entropy loss function for the proposed approach and Prototypical Network [8] while MSE (Mean-Square Error) and NLL (Negative Log Likelihood) loss for Relation Network [5] and Matching Network [7].

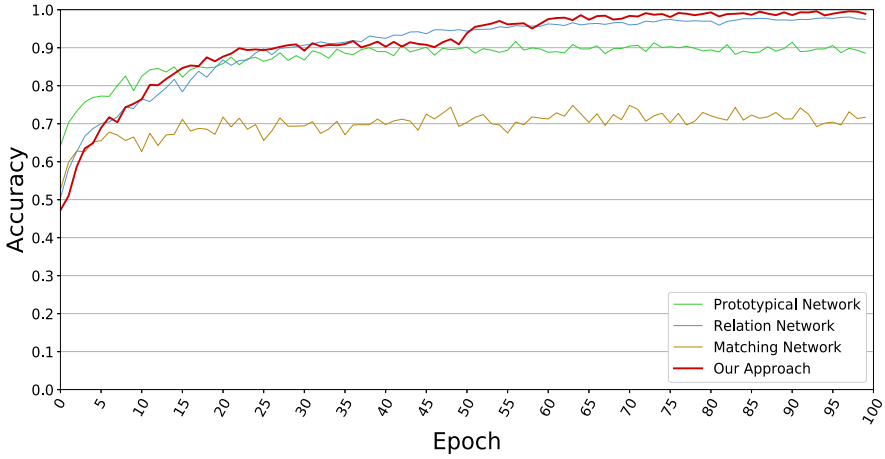

Fig. 7. Training accuracy curve with the increasing of epochs in 5-way 5-shot.

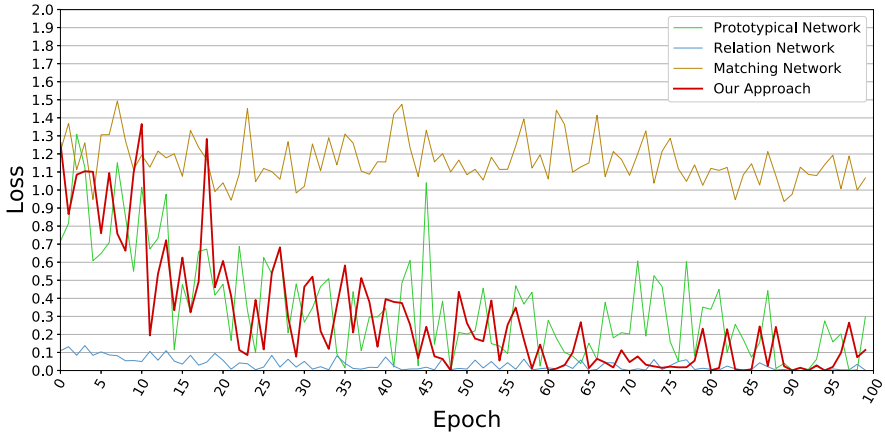

Fig. 8. Training loss curve with the increasing of epochs in 5-way 5-shot.

4 Conclusion

In this paper, we collect 11,236 malware samples from the real-world environment and build a malware database. To solve the problem of few labeled samples of malware classification and improve the generalization ability, we propose a novel malware classification approach called attention-based transductive learning network. By training with few malware samples, our approach performs the classification accuracies of 60% more than traditional machine learning methods and 50% more than the recent deep learning malware classification methods. Comparing with some few-shot learning approaches, we also achieve the highest

classification accuracies in few-shot experiments and a considerable performance in the one-shot experiment.

Acknowledgement. This work was supported by the National Key R&D Program of China(Grant No. 2018YFC1201102, Grant No. 2017YFB0802804) and Key Program of National Natural Science Foundation of China (Grant No. U1766215).

References

1. Sonicwall cyber threat report. https://www.sonicwall.com/resources/white-papers/2019-sonicwall-cyber-threat-report/
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
3. Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., Nicholas, C.K.: Malware detection by eating a whole exe. In: Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence (2018)
4. Le, Q., Boydell, O., Mac Namee, B., Scanlon, M.: Deep learning at the shallow end: malware classification for non-domain experts. Digital Invest. **26**, S118–S126 (2018)
5. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H., Hospedales, T.M.: Learning to compare: relation network for few-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1199–1208 (2018)
6. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Trans. Pattern Anal. Mach. Intell. **28**(4), 594–611 (2006)
7. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D. et al.: Matching networks for one shot learning. In: Advances in Neural Information processing systems, pp. 3630–3638 (2016)
8. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems, pp. 4077–4087 (2017)
9. Liu, Y., et al.: Learning to propagate labels: Transductive propagation network for few-shot learning (2018). arXiv preprint arXiv:1805.10002
10. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning (2016)
11. Ren, M., et al.: Meta-learning for semi-supervised few-shot classification (2018). arXiv preprint arXiv:1803.00676
12. Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.: Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security, ACM, p. 4 (2011)
13. Torralba, A., Murphy, K.P., Freeman, W.T., Rubin, M.A.: Context-based vision system for place and object recognition (2003)
14. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. Int. J. Comput. Vision **42**(3), 145–175 (2001)
15. Kalash, M., Rochan, M., Mohammed, N., Bruce, N.D., Wang, Y., Iqbal, F.: Malware classification with deep convolutional neural networks. In: 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), IEEE pp. 1–5 (2018)
16. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014). arXiv preprint arXiv:1409.1556

17. Ding, Y., Wu, R., Xue, F.: Detecting android malware using bytecode image. In: Xiao, J., Mao, Z.-H., Suzumura, T., Zhang, L.-J. (eds.) ICCC 2018. LNCS, vol. 10971, pp. 164–169. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94307-7_13
18. Tran, T.K., Sato, H., Kubo, M.: One-shot learning approach for unknown malware classification. In: 2018 5th Asian Conference on Defense Technology (ACDT), IEEE pp. 8–13 (2018)
19. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., Lillicrap, T.: Meta-learning with memory-augmented neural networks. In: International Conference on Machine Learning, pp. 1842–1850 (2016)
20. Woo, S., Park, J., Lee, J.-Y., Kweon, I.S:.Cbam: Convolutional block attention module. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 3–19 (2018)
21. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing Systems, pp. 321–328 (2004)
22. MalShare. https://www.malshare.com
23. Hybrid-Analysis. https://www.hybrid-analysis.com
24. VirusSign. https://www.virussign.com
25. Kabanga, E.K., Kim, C.H.: Malware images classification using convolutional neural network. J. Comput. Commun. **6**(1), 153–158 (2017)
26. Sharma, G.A., Singh, K.J., Singh, M.D.: A deep learning approach to image-based malware analysis. In: Das, H., Pattnaik, P.K., Rautaray, S.S., Li, K.-C. (eds.) Progress in Computing, Analytics and Networking. AISC, vol. 1119, pp. 327–339. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-2414-1_33