IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON INTERNET-OF-THINGS ATTACKS AND DEFENSES: RECENT ADVANCES AND CHALLENGES

Received August 17, 2020, accepted September 5, 2020, date of publication September 18, 2020, date of current version September 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3024991

Multi-Loss Siamese Neural Network With Batch Normalization Layer for Malware Detection

JINTING ZHU[1], JULIAN JANG-JACCARD[1], AND PAUL A. WATTERS[2]
[1]Cyber Security Laboratory, Comp Sci/Info Tech, Massey University-Auckland, Auckland 0632, New Zealand
[2]Security Studies and Criminology, Macquarie University, Sydney, NSW 2109, Australia

Corresponding author: Jinting Zhu (j.zhu3@ massey.ac.nz)

This work was supported by the Cyber Security Research Programme—Artificial Intelligence for Automating Response to Threats from the Ministry of Business, Innovation, and Employment (MBIE) of New Zealand as a part of the Catalyst Strategy Funds under Grant MAUX1912.

ABSTRACT Malware detection is an essential task in cyber security. As the trend of malicious attacks grows, unknown malware detection with high accuracy becomes more and more challenging. The current deep learning-based approaches for malware detection are typically trained with large amounts of samples using labeled and existing malware families in the training set, thus, their capability to detect new unseen malware (such as a zero-day attack) is limited. To address this issue, we propose a new one-shot model called ''Multi-Loss Siamese Neural Network with Batch Normalization Layer'' that can work with fewer samples while providing high detection accuracy. Our model utilizes the Siamese Neural Network to detect new variants of malware that is trained with only a few samples. Our model is equipped with batch normalization and multiple loss functions to address the overfitting issue, due to the use of small samples, that can create the vanishing gradient problem as a result of binary cross-entropy loss, and feature embedding space to improve the detection accuracy. In addition, we illustrate a way to convert raw binary files into malware gray scale images, to work with the popular Siamese Neural Network by generating the positive and negative pairs for training. Our experimental results show that our model outperforms existing similar methods.

INDEX TERMS Siamese neural network (SNN), malware detection, vanishing gradient problem, feature embedding space, zero-day attack.

I. INTRODUCTION

Malware is an urgent and pressing problem for the cybersecurity industry. Malware attacks cause an enormous amount of social and economic damage. Modern security software, widely deployed, utilises machine learning to facilitate the accurate and timely identification of malware. However, employing these algorithms requires training a machine learning model with thousands or millions of known malware samples that are labelled and representative of future samples [1]. In most real-world scenarios, it is often difficult to obtain that kind of large number of malware samples with proper labels, especially if they are new variants of polymorphic code, and possibly not seen before (e.g., a zero-day attack).

Malware features are generally extracted by using two different types of analysis techniques: static and dynamic respectively. Signature-based static features are extracted by disassembling the code and analyzing the execution trace to identify any malicious patterns. On the other hand, dynamic features are extracted from executing the code in a virtual environment and by generating a behavioral report based on the execution trace. The behavioral report is analyzed to capture the behavior of malicious code. Dynamic analysis is typically regarded to have more advantages than static features, because it provides better information for detecting accurate malware. This includes dynamic code loading and system calls. Despite the advantages, executing the malicious code in a virtual environment comes with a several challenges. For example, it may not trigger the conditions that are critical in detecting malware in a real environment. Moreover, it generally demands more significant time, resources and professional knowledge. For example, dynamic analysis may need to activate the Command & Control (C&C) server [2]. Hence, a simpler way to extract dynamic features is needed.

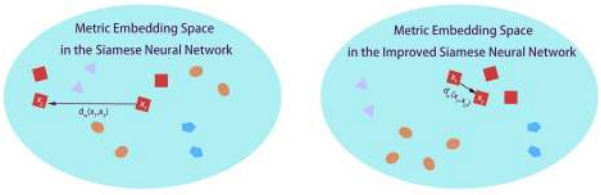
The associate editor coordinating the review of this manuscript and approving it for publication was Weizhi Meng.

The analysis of raw byte sequences has been regarded to be most promising [3]–[5], thus, a new method based on the raw executable technique known as ''malware visualization'' has been proposed. Nataraj *et al.* [6] utilizes a technique to visualize malware binaries resulted in the dynamic analysis as gray-scale images then classify them based on the similar image layout and texture. Though promising, another issue involved with effective malware detection is with the size of malware samples, especially if deep learning-based technique is utilized. In the majority of cases, a deep learning model typically require tens of thousands (or even millions) of data points to train to achieve accurate classification, such as seen in Malconv [4]. Unfortunately, it is often difficult to find such a large number of samples for training. To address this issue, data augmentation technology has been used to increase the sample size and diversity of samples (e.g., different malware families) [7].

Extending further from the data argumentation, a new type of neural network model called N-shot learning (or also called as one-shot or few-shot) have been proposed which can make full use of prior information to improve the performance of the model under more constraint input samples. As name implies, the N-shot learning aims at building a task from one or very few training examples. These latest techniques have been applied in malware detection with some degree of successes. Li *et al.* [8] proposed a methodology for detecting malware with binary classifier. Though claimed to produce better accuracy with smaller sample, their proposal does not take account into the details of malware classes and the intra-class features. Kalash *et al.* [9] and Pascanu *et al.* [10] proposed a network for classifying malware based on malware samples only without getting benign samples involved in detection. As these existing methods do not contain the comparisons among different samples involved (e.g., benign vs malware, cross referencing among the same or different families of samples), the binary cross-entropy loss tends to show slow convergence and unstable performance [11], and the distance between the pairs generated from same family is not considered. Furthermore, as these models tend to ignore the embedding space of inter-class, the performance of accuracy may be limited, since the same malware image may have large distances in the feature embedding space due to the changed part of executable file. However, these malware pairs are supposed to share similar labels and contribute to a loss term via the same label (as seen in Figure 1(a)).

To address these issues associated with the existing methods, we propose Multi-Loss Siamese Neural Network with Batch-Normalization. With added Batch-Normalization and the combination of two loss functions, hence the name multi-loss, our model provides a better similarity distance measure resulting in achieving high detection accuracy (as seen in Figure 1(b)). The main contributions of our proposal are as follows:

- We introduce a strategy to convert a binary file, such as Andro-Dumpsys dataset, into the image files that can feed into an N-shot based neutral network model such as a Siamese Network.
- Our proposed model is tuned such as way that it can work well on a small amount of training datasets. Specifically, this is done by appropriately setting the network parameters for the data augmentation and adding batch normalization to avoid overfitting that could have caused due to the use of small datasets.
- The multiple loss function is useful to improve the feature embedding space in the Siamese Neural Network for binary classification. In the feature embedding space, the distance of each positive pair that belongs to the same class become small.
- Our experimental results illustrate that our proposed model achieves higher accuracy than the baseline methods in one-shot classification tasks in malware detection.



(a) Feature embedding space (b) Improved embedding space

FIGURE 1. **Improving the similarity distance in an embedding space.**

II. RELATE WORK

In this section, we discuss the state of the art in the use of deep leaning techniques for malware detection. In one of the earliest adoptions of a deep learning technique for malware analysis, Kalash *et al.* [9] used a Convolutional Neural Network (CNN) to detect malicious code and showed detection results on the gray images of the Malimg Dataset [6]. From there, Wang *et al.* [12] employed multiple CNNs to detect Android malware including the architecture for the use of the activation function based on Rectified Linear Unit (ReLU) to increase sparseness and dropout to prevent over-fitting. In many cases, deep learning researchers generally use the combined convolutional and pooling layer with the full-connection layer to enhance feature extraction capability. This approach is further defined to combine CNN and deep auto-encoder to learn more flexible model and reduce the training time. A Recurrent Neural Network (RNN) [13] is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence which allows to exhibit temporal dynamic behavior. Unlike other feed forward based neural network models, RNN can use their internal states, kept in the memory, to process sequences of inputs. This makes it more applicable to tasks such as sentence classification [14] and speech recognition [15]. Pascanu *et al.* [10] employed a RNN technique to detect malware by constructing an API call language model. Their proposal used the hidden state of the

model which encodes the history of previous event as the fixed-length feature vector that is given to loss function of logistic regres- sion. Xiao *et al.* [16] considered that there exists some semantic information in system call sequences, much like the structure of natural language by treating one system call sequence as a sentence in the language and then constructing a classifier based on the Long Short-Term Memory (LSTM) language model. In this model, two LSTM models are trained respectively by the system call sequences from malware and those from benign samples. According to these models, two similarity scores are computed. Finally, the classifier determines whether the application under analysis is malicious or trusted by examining the score. Tobiyama *et al.* [17] investigated the application of Deep Neural Networks (DNN) to classify malware process. In their project, the authors trained a Recurrent Neural Network (RNN) to extract the features of malware behavior followed by further training the Convolutional Neural Network (CNN) to classify the feature images which are generated by the extracted features from the trained RNN. However, these models are trained on the large scale of samples.

III. OUR MODEL

In this section, we introduce our model along with the provision of the structure and step by step guide to construct the training samples.

A. MALWARE IMAGES

In our approach, we take malware detection as a visualization task by converting malware binary code into malware images then running a classification task using a deep learning approach. To feed into our proposed model, the malware binary code needs to be converted into an appropriate input format as seen in Figure 2.

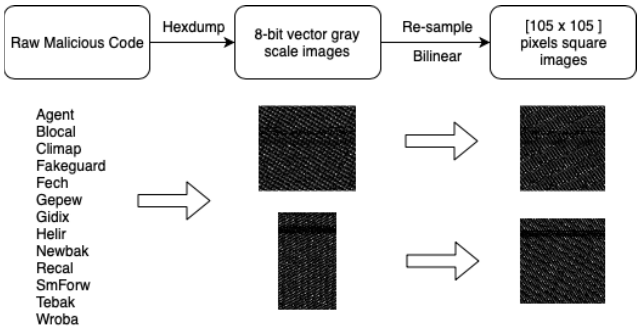


FIGURE 2. Input image conversion.

Firstly, the binary malware code is read as a vector of 8 bit unsigned integers. This 1D vector of 8 bit unsigned integers is converted into 2D vectors with the fixed width for various file sizes (see Table 1) and the height corresponding to the actual size of the original file, as proposed by Nataraj *et al.* [6]. Further, we convert the 2D vector to form the gray images in the range [0, 255]. At this stage, the converted gray images are in various dimension (e.g., vary in size with different height and width) as shown in Table 1.

TABLE 1. Image width for various file sizes.

File size	Image width	File size	Image width
<100 KB	256	4M∼8M	1280
100KB∼500KB	512	8M∼16M	1536
500 KB∼1M	768	16M∼32M	1792
2M∼4M	1024	>32M	2048

These various image dimensions would result in the fully connected layer in a neural network to occur errors due to inconsistency in the dimensions across different input samples. To avoid such problems, we use a bi-linear interpolation method [18] to make the images uniform to 105×105.

In addition, the malware images come from different malware families. The understanding of the origin of the malware family can be captured by examining the texture of malware images. For example, the texture of two malware images from the same family would be similar but slightly different in texture (as seen in Figure 3 (b) and (e)). The similarity in the image texture is because many variants of malware in the same family are usually developed based on the original malware and share many similar characteristics (i.e., malware signatures). Figure 3 illustrates the different texture of different malware families.

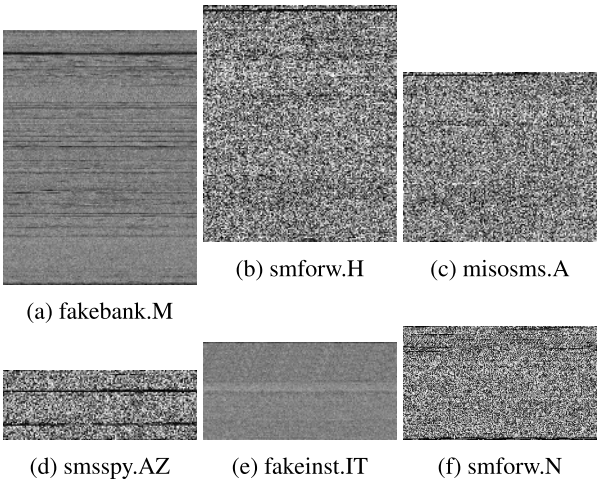


FIGURE 3. The samples of different malware family.

B. THE OVERVIEW

We utilize a Siamese Neural Network (SNN) as a key component. Our model is mainly composed of two main parts; the shared neural network and multi-loss structure respectively. The shared neural network is further composed of a convolutional neural network, a batch normalization layer, pooling layer whereas the multi-loss structure is composed of the two loss functions and fully connected layers. This general overview of our approach is shown in Figure 4. Our proposed model goes through three phases; malware image pre-processing, training, and testing respectively.

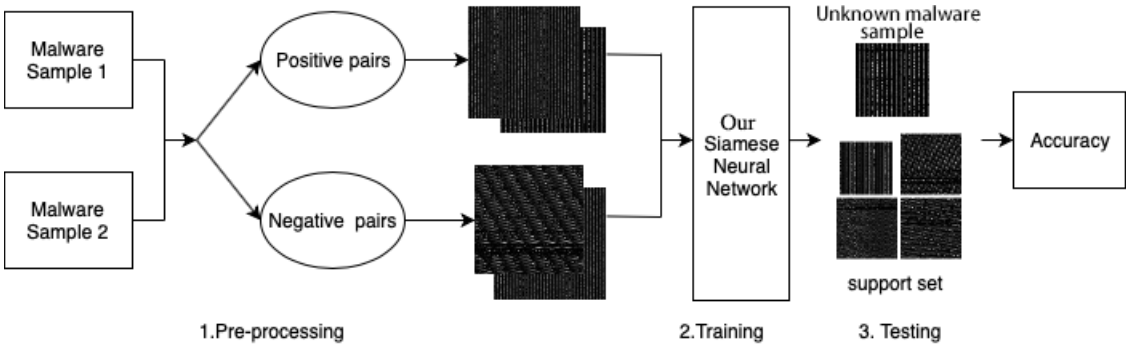


FIGURE 4. Flowchart.

- The pre-processing phase: the input images are prepared from the binary format (of Android malware) into gray images (as seen in Section 3.1). The images are divided into two different groups: positive pairs and negative pairs and feed into two separate CNN neural networks (i.e., branches of the Siamese Neural network).
- The training phase: we create the pairs of images which are concatenated with samples selected from the same or different classes. The dimension of N-image batch size is n, x, and y. Overall of the batches are separated into two aspects equally and labeled as 1, or 0 respectively, where N is the batch size, n is the randomly selected sample in the categories, x is the width of image and y is the height of image.

Put more formally, our approach can be written as follows. Note that we define the each malware images as E_i, where dimension is 150 × 150 pixels. Table 2 lists the notation and their descriptions.

TABLE 2. The used notations in this paper.

Notation	Description
E_i^n	the number of sample
E_i^x	the width of image
E_i^y	the height of image
E_i^z	the channel of image
x_i	the feature vectors of image
y_i	the label of sample
d_w^i	the distance feature of a pairs of images $E_i^+ and E_i^-$
y_d	the label of pairs of images
F_w	the convolutional filter with parameters w
λ	the hyper-parameters

1) INPUTTED MATRIX

We use $< E_i^+, E_j^+ >$ and $< E_i^+, E_j^- >$ to denote the positive pairs and negative pairs respectively. The form of the inputted matrix E_i are concatenated as:

$$E_i = [E_i^n; E_i^x; E_i^y : E_i^z] \tag{1}$$

where E_i^n is the number of sample, and E_i^x is the width pixels, E_i^y is the height pixels. the E_i^z is the RGB channels for the image. For instance, if the image is RGB the E_i^z will be 3. In this proposal, our processed images are grayscale, therefore the value of E_i^z is 1.

2) NETWORK STRUCTURE

As shown in Figure 5, our model has a pair of convolutional networks which share the parameters, the weights $W \in R^d$. The main advantage of shared weights, is that we can substantially lower the degrees of parameters to optimize and avoid overfitting as the weight are shared with some other neurons. The pair of convolutional networks is inputted with a unit matrix. This can be represented as:

$$F_w(E_i) =< F_w(E_i^+), F_w(E_i^-) > \tag{2}$$

where the F_w is the feature representation of inputted matrix of E_i which is generated by the model. Generally, this model g_w: $R^n \mapsto R^d$ is parameterized by weights w;

3) MULTI-LOSS TRAINING

Given a Siamese Neural Network structure g_w and the inputted matrix E_t. We aim to learn a model which predict the probability of malware with the multi-loss function which jointly optimizes identification loss and detection loss for malware detection, which is defined as follows:

$$L(d_w, y_d, x_i, y_i) = L_1(d_w, y_d) + \lambda L_2(x_i, y_i) \tag{3}$$

where λ is a hyper-parameter to weight the relative and importance of each loss.

C. MAIN COMPONENTS

The main components of our improved Siamese Neural Network includes: 1) the batch normalization layer is added for normalizing the output of the previous activation; 2) multi-loss structure is added for improving the feature embedding space, as seen in Figure 5.

In a Siamese network g_w, there are typically two symmetrical branches which shares the same learned weights. A pair of images is entered as a representation of certain features into each branch. Within a branch, the pair of images go through a series of convolutional layers, pooling layers, and fully connected layers. Generally, the top layer in each branch

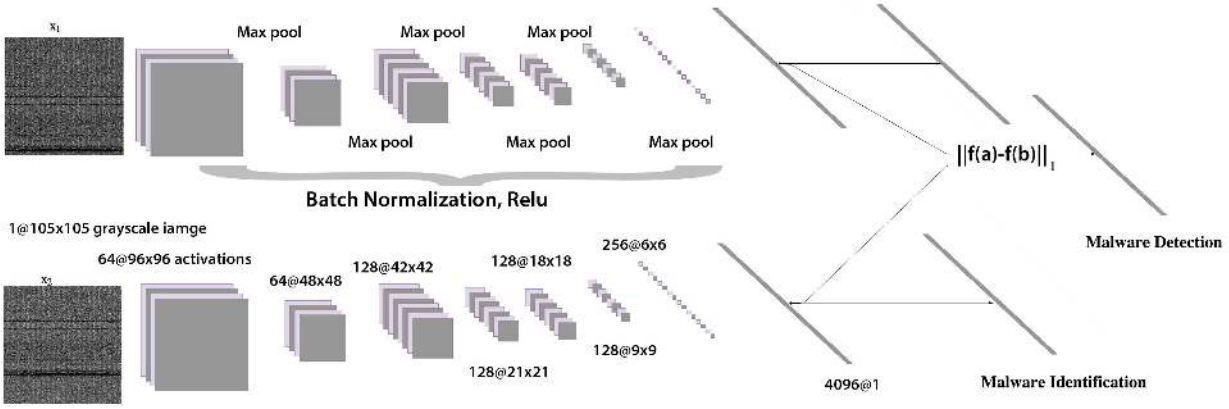


FIGURE 5. The overall network structure.

Algorithm 1 Pseudo-Code of Our Proposed Algorithm

Pairs generator:$E = G(x_1, x_2, \ldots, x_n)$;
Initialization weights and biases as [19];
function Euclid $d_w(x_1, x_2)$;
Binary_crossentropy L_1; Softmax L_2;
Input : Training set E_i, support set E_s, Target label L_t, Hyper-parameters λ
Output: [Predicted PairLabel]
1) Training Stage with forward and backprogation ;
while *not reach to iterations* **do**
  $F_w(x_1)$ = One branch of Siamese with batch normalization layer ;
  $F_w(x_2)$ = One branch of Siamese with batch normalization layer ;
  $d_w(x_1, x_2) = F_w(x_1) - F_w(x_2)$;
  Loss of PairLabel = $L_1(d_w)$;
  Loss of ClassLabel = $L_2((F_w(x_1))$;
  Loss of ClassLabel = $L_2((F_w(x_2))$;
  Total loss = $L_1 + \lambda L_2$
2) Testing Stage;
while *not reach to iterations* **do**
  if $E_s[index].Predict\_binarylabel = L_t[index]$ **then**
    return $correct + 1$
Accuracy = 100× correct/iterations;

uses softmax activation function to classify the pairs of the image to see whether the pair belongs to the same category.

Mathematically, the similarity between a pair of images (x_1, x_2) within Euclidean distance can be computed by a Siamese Neural Network (SNN) and it can be described as

$$d_w(x_1, x_2) = \|F_w(x_1) - F_w(x_2)\|_2 \tag{4}$$

In this equation, if x_1 and x_2 are similar, the $d_w(x_1, x_2)$ will be close to zero, otherwise they are dissimilar. However, previous research [20] only employed the logistic loss for malware detection, as shown in Eq.(5), to measure the similarity between the inputted images.

$$L(w) = \sum_{i=1}^{p}(1 - y_i)f_p(d_w^i) + y_i f_q(d_w^i) \tag{5}$$

where y_i is the label for the input pair of images, if the images (x_1, x_2) are similar, the $y_i = 1$, otherwise, $y_i = 0$ denote that they are dissimilar.

D. BATCH NORMALIZATION(BN) LAYER

However, in a typical SNN, the vanishing gradient problem would occur mainly because the distribution of training data points have changed or shifted as the distribution gradually approaches the upper and lower limits of the interval of the nonlinear function value. This happens since the parameters of the preceding layers change at the training stage. Accordingly, the distribution changes at the current layer such that the current layer needs to constantly readjust to new distributions. This problem is especially severe for deep networks, because small changes in shallower hidden layers would be amplified as they propagate within the network resulting in significant shift in deeper hidden layers. Our model adds Batch Normalization (BN) [21] to reduce these unwanted shifts to speed up training and to produce more reliable models. With this additional layer, the network can use higher learning rate without vanishing or exploding gradients. Furthermore, it also regularizes the network such that it is easier to generalize, and avoids the use dropout to mitigate overfitting. The network also becomes more robust to different initialization schemes and learning rates. As in [21], the BN framework is considered primarily for convolutional neural networks. Both the input and output of a BN layer are four dimensional tensors, which are denoted as $I_{b,c,x,y}$ and $O_{b,c,x,y}$, respectively. These dimensions correspond to examples within a batch b, channel c, and two spatial dimensions x, y respectively. For input images, the channels correspond to the RGB channels. BN applies the same normalization for

all activation in a given channel,

$$O_{b,c,x,y} \leftarrow \gamma_c \frac{I_{b,c,x,y} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} + \beta_c \quad \forall b, c, x, y \tag{6}$$

where μ_c denotes $\frac{1}{|\omega|}\sum_{b,x,y} I_{b,c,x,y}$ from all input activations in channel c, where ω means all activations in channel c across all features b in the entire mini-batch and spatial x, y locations. Subsequently, BN divides the centered activation by the standard deviation μ_c (plus ϵ for numerical stability) which is calculated analogously. During testing, running averages of the mean and variances are used. Normalization is followed by a channel-wise affine transformation parameterized through γ_c, β_c, which are learned during training.

E. MULTI-LOSS STRUCTURE

In our model, we introduce two loss functions: the softmax loss (as the identification loss function) and logistic loss (as the detection loss function), respectively. The softmax loss is used to optimizing the feature embedding space which is enhanced by the logistic loss that issued to calculate the distance between pair of images. We provide that this style of combined use of multi-loss functions allow the sharing of the information better across different tasks thus contributing to achieve better detection accuracy. In addition, the sharing of the information by these two multi-loss functions also allows our model to suffer less from overfitting when compare to other similar models that use only a single loss function.

1) IDENTIFICATION LOSS

Generally, the identification loss plays the same role as the general classification task. A softmax regression is usually used for the multi-class classification. In this proposal, we employ softmax loss to enlarge the inter-class distinction to optimize the feature embedding space to force the feature representation corresponding to the data point of the same class. This can be written as follows:

$$L_1 = -\frac{1}{M}\sum_{i=1}^{M} log \frac{e^{W_{y_i}^T x_i} + b_{y_i}}{\sum_{j=1}^{N} e^{W_j^T x_i + b_j}} \tag{7}$$

where M is the number of batch size, y_i denotes the label (i.e., the number of malware class) and x_i which denotes the deep feature of malware. $W_j \in \mathbb{R}^{d \times n}$ in the identification fully connected layer and $b \in \mathbb{R}^n$ is the bias term. Finally, the Eq.(7) is the part of two losses, which are optimized jointly during the training.

2) DETECTION LOSS

In our approach, the malware samples belong to the same class, which are positive pairs (labelled as 1) while a pair of a malware sample and benign sample, or the malware from different class are constructed to negative pairs (labelled as 0). The distance of two images of positive pairs is expected to closest when they are matched with each other. With these considerations, the most suitable loss function for our model is the logistic regression. Thus, the logistic regression for our model can be given as follows:

$$L_2 = -\frac{1}{M}\sum_{i=1}^{M}[y_d^i f_p(d_w^{(i)}) + (1 - y_d^i) f_q(d_w^{(i)})] \tag{8}$$

where y_d is the label of image pairs, d_w is the Euclidean Distance between two images. Theoretically, the most similar image is supposed to be the closest in the feature embedding space.

3) MULTI-LOSS FUNCTION

Our model is trained by softmax loss and logistic loss and finally both of them are converged. The identification loss of a softmax function, which shares the fully connected layer with detection loss, considers the feature of shared fully connected layer as the input. Each pair of feature vectors independently generates a loss term and its specific embedding space. It is possible that the manifold feature is changed by the behavior of malware code, because the same malware family not sharing the specific class label, is only marked by the indicator, y_d of a pair. Furthermore, the features of all branches will be feed into their final loss to be optimized. During the training stage, the combined loss function is formulated as follows:

$$L = L_1 + \lambda L_2 \tag{9}$$

where λ is a hyper-parameter to balance the weight of two loss function, which is set to 0.4. From the Eq.(7) we can observe that each loss function is responsible for performing a specific classification task, and this loss function assists the prediction ability of the one-vs-one classifier with its discriminative features.

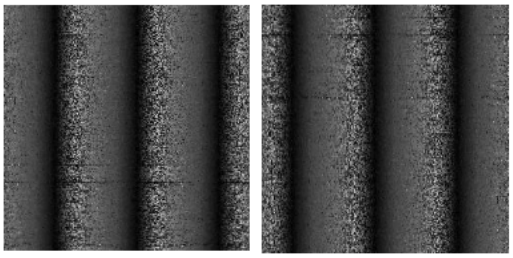
IV. EXPERIMENTS AND RESULTS

In this section, we evaluate the performance of our proposal. The experimental results have been obtained by running the tests on the desktop with the 32GM RAM, Nvidia Quadro P2000(5GB), and Xeon W-2133 CPU@3.6GHz.

A. DATASET AND SETUP

In this experiment, we employ the dataset from our collaborator [22] that has been widely used. The dataset consists of 906 malicious binary files from 13 malware families. The number of the malware samples from different malware classes varied. Almost half of the classes have no more than 25 malware samples, and some have only one as they are new malware detected lately(e.g., Blocal and Newbak). We increased the data size for every malware class to have at least 30 samples using a manual data argumentation technique for malware diversity [1] using the parameters shown in Table 3. A separate dataset containing 1,776 benign samples was randomly used as input pairs in the support set.

To train our model is conducted per mini-batch. Our model randomly selects image pairs as half positive pairs and half negative pairs. We randomly select anchor image pairs as mini-batch to feed into the model. The batch size includes



(a) Agent(1).a: the original image (b) Agent(1).b: an augmented image

FIGURE 6. The examples of data augmentation.

TABLE 3. Data augmentation parameters.

Methods	Values	Description
rescale	1./255	Resizing an image by a given scaling factor.
zca_epsilon	1e-06	Epsilon for ZCA whitening.
fill_mode	wrap	Points outside the boundaries of the input are filled according to the given mode.
rotation_range	0.1	Setting degree of range for random rotations.
height_shift_range	0.5	Setting range for random vertical shifts.
horizontal_flip	True	Randomly flips inputs horizontally.

the numbers of different randomly selected image pairs. The initial learning rate was set to 0.00002. The maximum number of iterations was 15,000, and the batch size was 35. After training, we save the parameter weights for testing. The details of the parameters we set of training are shown in Table 4.

TABLE 4. Training parameters.

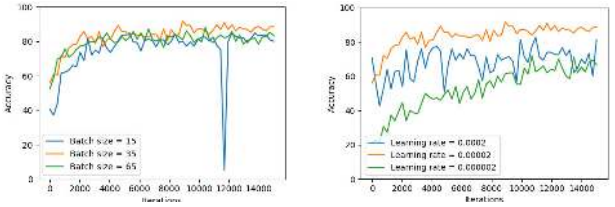
Parameters	Values	Description
Mini-Batch	35	The number of training examples in one forward/backward pass
Learning rate	0.00002	Learning rate is used in the training of neural networks- range between 0.0 and 1.0.
N-iterations	15000	Total numbers of iterations in the training process
N-way	3 to 15	To compare a test image with N different images from different classes
N-tasks	250	The number of one-shot tasks to validate on

B. N-WAY ONE-SHOT ACCURACY

The testing process conducts M times of N-way one-shot learning tasks, where Q times of correct predictions contribute to the accuracy calculated by the following formula:

$$Accuracy = (100 * Q/m)\% \tag{10}$$

The evaluation of N-way one-shot learning at each test state, we choose an anchor image from one class of test, and then randomly selects N classes of images to form the support set $X = \{x_i\}_{i=1}^{N}$, where $x_1, \forall x \in X$, where selected image's class is the same as the anchor image $\hat{x}$, the other images in support set are from different classes. The similarity score between $\hat{x}$



(a) Parameters of batch size (b) Parameters of learning rate

FIGURE 7. Parameters comparison.

and other image is calculated through Siamese network. To be specific, if the similarity score of feature vector of x_1, which can be represented as $S = \{s_i\}_{i=1}^{N}$, that score is the maximum of S the task can be labeled as a correct prediction. Otherwise, it is regarded as the incorrect prediction. As we can see from the Figure 8, our proposal worked efficiently by producing the best accuracy compare to other similar models [20], [23] and similar work [24]. Our proposed model produced the best detection accuracy – that is, above 99% during the training phase while around 90% during the testing phase. It must note that our proposed model didn't have any large decline in the detection rate, as observed in other similar approaches, in the presence of the increasing N-way pairs.

Methods	5-way	10-way	15-way
KNN [23]	63.8	26.4	25.2
SNN [20]	86	69.2	64
Prototypical Network [24]	89.5	87.9	82.6
Matching Network [24]	86.2	83.2	81.8
BN-SNN	95.5	88.4	84.2
Ours(train)	99.2	99.6	99.2
Ours(test)	93.8	89.6	89.2

FIGURE 8. The accuracy of N-way one-shot learning for different methods.

C. DISTANCE MEASURE EFFECTIVENESS

We also conducted an experiment to analyze the effectiveness of our algorithm on distance measure using Receiver Operating Characteristic (ROC) curve. We denote $f_0(p)$ as the probability density function of predictions $p(x)$ from our algorithm of negative pairs that are labeled as 0 and $f_1(p)$ are the probability from positive pairs that are labeled 1. The true positive rate (TPR) and false positive rate (FPR) for a given discrimination threshold p are the integrals of the tails of these distributions.

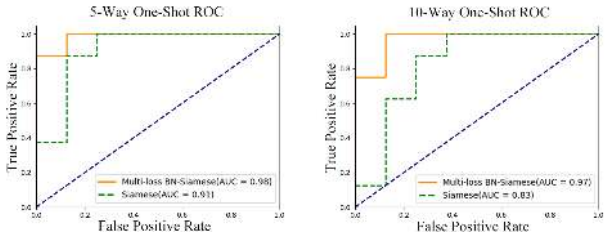
$$FPR(P^*) := \int_{p^*}^{1} f_0(p)dp$$
$$TPR(P^*) := \int_{p^*}^{1} f_1(p)dp \tag{11}$$

The ROC curve is the function TPR(FPR) and thus the area under the curve (AUC) is:

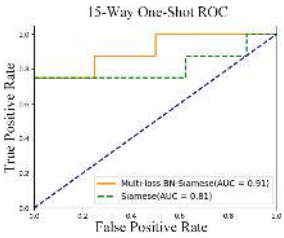
$$AUC = \int_{0}^{1} TPR(FPR)D(FPR) \tag{12}$$

from the equation we can see that the AUC (Area Under the Curve) is the probability that a randomly chosen point

from class 0 ranks below a randomly chosen point from class 1. If the classifiers perform well, the AUC is to be closer to 1. We benchmarked our model against the popular original implementation of the Siamese Neural Network [25]. As Figure 9 shows, the result is on a set of points in the true positive rate - false positive rate plane, which is the curve for our data set. When the probability value of AUC is close to 0.5, it means that an algorithm randomly guessing whether a given sample is malware or benign. We respectively achieve the AUC equals to 0.98, 0.97, and 0.91 respectively under the 5-way, 10-way, and 15-way. It is noted that our model always performs better as the AUC area of our proposal is larger against other benchmark.

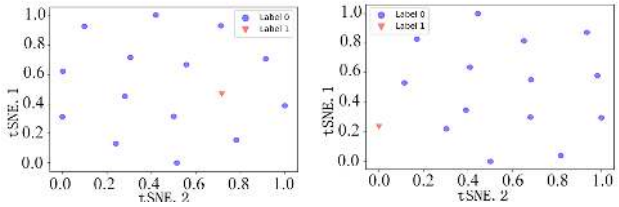


(a) ROC of 5-way One-shot (b) ROC of 10-way One-shot

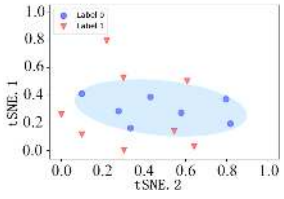


(c) ROC of 15-way One-shot

FIGURE 9. **ROC curves under the N-way One-shot.**



(a) Original Binary Classifier (b) Our Binary Classifier



(c) Few-Shot Binary Classifier

FIGURE 10. **Feature Visualization Using t-SNE.**

D. VISUALIZING NEAREST NEIGHBORS WITH T-SNE

Figure 10 displays the feature vectors of 4096 dimension extracted from our model. With the PCA initialization, t-Distributed Stochastic Neighbor Embedding (t-SNE) technique projects these feature vectors into two dimensional for visualization. For this visualization, we only show the positions of 15 image pairs which include 14 negative pairs (blue circle point) and 1 positive pairs (red triangle point). Compared with the original classifier, as shown in the Figure 10 (a), the the positive pairs are isolated by another negative pairs well in the Figure 10 (b). We also do multiple positive pairs for visualizing this model performance. In the Figure 10 (c), the blue area cover all 7 negative pairs, and it only contains one wrong positive pair.

V. CONCLUSION AND LIMITATIONS

In this paper, we propose a new neural network model based on one-shot learning for malware detection. This model effectively solves the problem that the traditional model cannot detect unknown malware, and further optimizes the feature space so that positive samples from the same class have a local distance greater than samples of different malware classes.

The experiments showed that our algorithm performed better than other base-line methods, such as Siamese Neural Network, and KNN. For the further study, we plan to make a use of other metric learning applied to our model to improve recognition accuracy. We also have a plan to add the Spatial Pyramid Pooling layer into Siamese network since it allows the convolutional network taking arbitrary size of images to avoid the information loss of malicious code. By adding this layer, the arbitrary size of feature maps can be adjusted via the spatial pooling regions to be the appropriately proportional to the size of the output matched with the fully connected layers.

However, there are a number of limitations of our current proposed model. Our proposed model, for example, could not correctly classify a polymorphic malware (i.e., disguised as a new malware by changing or moving some parts of the code elsewhere) but would recognize it as a new malware [26]. The validity of the synthetic data produced by the data argumentation strategy we proposed has not been fully examined to ensure whether the synthetic data captures the characteristic of real malware. In addition, our current model can be vulnerable to misclassification against adversarial attack if the original data is modified, especially when the original malware sample is in the raw binary file format.

REFERENCES

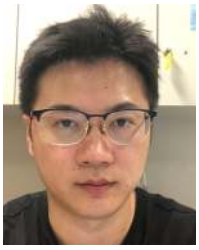
[1] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3187–3196, Jul. 2018.

[2] Z. Xu, A. Nappa, R. Baykov, G. Yang, J. Caballero, and G. Gu, "AUTOPROBE: Towards automatic active malicious server probing using dynamic binary analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2014, pp. 179–190.

[3] M. Krčál, O. Švec, M. Bálek, and O. Jašek, "Deep convolutional malware classifiers can learn from raw executables and labels only," in *Proc. ICLR*, 2018, pp. 1–4.

[4] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. K. Nicholas, "Malware detection by eating a whole EXE," in *Proc. Workshops 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–6.

[5] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2018, pp. 533–537.
[6] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. 8th Int. Symp. Vis. Cyber Secur. (VizSec)*, 2011, p. 4.
[7] Q. Wang, W. Guo, K. Zhang, A. G. Ororbia, X. Xing, X. Liu, and C. L. Giles, "Adversary resistant deep neural networks with an application to malware detection," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1145–1153.
[8] 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.
[9] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, "Malware classification with deep convolutional neural networks," in *Proc. 9th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Feb. 2018, pp. 1–5.
[10] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 1916–1920.
[11] L. Berrada, A. Zisserman, and M. P. Kumar, "Smooth loss functions for deep top-k classification," 2018, *arXiv:1802.07595*. [Online]. Available: http://arxiv.org/abs/1802.07595
[12] W. Wang, M. Zhao, and J. Wang, "Effective Android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 8, pp. 3035–3043, Aug. 2019.
[13] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 240–254, Mar. 1994.
[14] S. T. Hsu, C. Moon, P. Jones, and N. Samatova, "A hybrid CNN-RNN alignment model for phrase-aware sentence classification," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics*, vol. 2, 2017, pp. 443–449.
[15] M. Fujimoto and H. Kawai, "Comparative evaluations of various factored deep convolutional RNN architectures for noise robust speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 4829–4833.
[16] X. Xiao, S. Zhang, F. Mercaldo, G. Hu, and A. K. Sangaiah, "Android malware detection based on system call sequences and LSTM," *Multimedia Tools Appl.*, vol. 78, no. 4, pp. 3979–3999, Feb. 2019.
[17] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, "Malware detection with deep neural network using process behavior," in *Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jun. 2016, pp. 577–582.
[18] H. S. Malvar, L.-W. He, and R. Cutler, "High-quality linear interpolation for demosaicing of bayer-patterned color images," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 3, May 2004, iii-485.
[19] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. ICML Deep Learn. Workshop*, Lille, France, vol. 2, 2015, pp. 1–8.
[20] S.-C. Hsiao, D.-Y. Kao, Z.-Y. Liu, and R. Tso, "Malware image classification using one-shot learning with siamese networks," *Procedia Comput. Sci.*, vol. 159, pp. 1863–1871, Jan. 2019.
[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167
[22] J.-W. Jang, H. Kang, J. Woo, A. Mohaisen, and H. K. Kim, "Andro-dumpsys: Anti-malware system based on the similarity of malware creator and malware centric information," *Comput. Secur.*, vol. 58, pp. 125–138, May 2016.
[23] W.-J. Hwang, "Fast kNN classification algorithm based on partial distance search," *Electron. Lett.*, vol. 34, no. 21, pp. 2062–2063, Oct. 1998.
[24] T. K. Tran, H. Sato, and M. Kubo, "Image-based unknown malware classification with few-shot learning models," in *Proc. 7th Int. Symp. Comput. Netw. Workshops (CANDARW)*, Nov. 2019, pp. 401–407.
[25] C. Zhang, W. Liu, H. Ma, and H. Fu, "Siamese neural network based gait recognition for human identification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 2832–2836.
[26] M. Alazab, S. Venkataraman, and P. Watters, "Towards understanding malware behaviour by the extraction of API calls," in *Proc. 2nd Cybercrime Trustworthy Comput. Workshop*, Jul. 2010, pp. 52–59.



JINTING ZHU received the master's degree in computer science from the Kunming University of Science and Technology, China. He is currently pursuing the Ph.D. degree with the School of Natural and Computational Sciences, Massey University–Auckland, New Zealand. His research interests include data analysis, few-shot learning, multimedia application, and malware detection.

JULIAN JANG-JACCARD received the B.Bus. (Comp.) degree from Western Sydney University, Australia, and the MInfTech and Ph.D. degrees from The University of Sydney, Australia. She is currently an Associate Professor and also the lead of the Cyber Security Laboratory, Comp Sci/Info Tech, School of Natural and Computational Sciences, Massey University–Auckland, New Zealand. She has a national reputation as a Cybersecurity Expert and has worked in the top Australian ICT industries, universities, and the Premier Australian Federal Government Research Agency, CSIRO. Her research interests include, but not limited to privacy-preservation techniques, AI-based threat detection/response, cyber resilient system, and applied cryptography. The techniques she developed have been applied in real-life telco and health informatics trials with success and news coverage. She is an active member of several database, cybersecurity, and health data informatics research communities, and has published more than 70 articles in the leading conferences and journal venues, including IEEE and ACM. She was a recipient of many multi-million dollar research funds both from Australian and NZ governments, often collaborating with the top international ICT companies, and the university around the world.

PAUL A. WATTERS is currently an Honorary Professor of security studies and criminology with Macquarie University, and an Adjunct Professor of cybersecurity with La Trobe University. He is the Founder of the 100 Point Cyber Check TM, and the CEO of Cyberstronomy Pty Ltd. His research interests include malware detection and analysis, phishing, and cyberepidemiology.