Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds

GENERATIVE ADVERSARIAL NETWORK FOR IMPROVING DEEP LEARNING BASED MALWARE CLASSIFICATION

Yan Lu

Department of Modeling, Simulation and Visualization Engineering
Old Dominion University
5115 Hampton Blvd
Norfolk, VA 23529, USA

Jiang Li

Department of Electrical and Computer Engineering
Old Dominion University
5115 Hampton Blvd
Norfolk, VA 23529, USA

ABSTRACT

The generative adversarial network (GAN) had been successfully applied in many domains in the past, the GAN network provides a new approach for solving computer vision, object detection and classification problems by learning, mimicking and generating any distribution of data. One of the difficulties in deep learning-based malware detection and classification tasks is lacking of training malware samples. With insufficient training data the classification performance of the deep model could be compromised significantly. To solve this issue, in this paper, we propose a method which uses the Deep Convolutional Generative Adversarial Network (DCGAN) to generate synthetic malware samples. Our experiment results show that by using the DCGAN generated adversarial synthetic malware samples, the classification accuracy of the classifier – a 18-layer deep residual network is significantly improved by approximately 6%.

1 INTRODUCTION

Malicious software known as malware is one of the major computer security threats, it had been growing exponentially in the past years. Many research attempts have been made to improve the adaptabilities and capabilities of the malware defense systems to detect and classify the new malware instances automatically in real time. These methods range from the traditional signature-based methods in the early days to the current deep learning methods. The traditional signature-based malware detection approaches rely heavily on domain expert knowledges and special software environments, which are both computation resources consuming and time consuming. Besides, the traditional malware detection methods do not adapt well when the malwares are modified polymorphically or metamorphically when the malwares evolve or propagate.

With the fast advancement of the deep learning models, many deep learning-based malware classification approaches have been explored including the deep belief networks, the deep recurrent networks, the deep convolutional neural networks and the deep residual networks etc. In most of these deep learning based malware classification models, the malware raw bytecodes are usually converted into images at first, which also converts the malware classification problem into an image classification problem. The deep learning methods significantly reduced the massive domain expert knowledge needed in malware classification. The malware classification accuracies have also been significantly improved by using deep learning models comparing to the traditional methods, the best accuracy can be found at as high as 98.62% on some particular malware benchmark dataset.

However, to train a deep learning model it needs large training data. In the real-world scenario, the new types of malwares usually come in limit number of examples, which are also very difficult to collect from all possible environments in short of time. Consequently, the deep learning models are restricted to good generalization performance, the classification and detection performance will be compromised accordingly.

To conquer the small data samples issue, in many previous related researches, the oversampling method such as SMOTE (Chawla et al. 2002; Han et al. 2005) method, the adaptive synthetic sampling method, and the minority oversampling method have been widely adopted. However, most of these methods focus on solving the small data problem by re-using the existing data rather than generating new data. Goodfellow et al. (2014) introduced the generative adversarial networks (GAN) framework in 2014. GAN is a deep learning framework aimed to generate synthetic samples with the same distribution as the real data. After the GAN network is released, GAN model has gained massive attention, it has been successfully applied to many research fields such as computer vision and natural language processing in the past several years.

To explore the possibility of solving the small data issue and improving the classification performance in the malwares classification by using GAN model, in this paper we present a method by using the Deep Convolutional Generative Adversarial Network (DCGAN) to generate synthetic malware samples, and then combined with the real malware data to improve the classification performance of a 18-layer deep residual network (ResNet-18). Our experiment results show that the classification accuracy of the deep residual network has been significantly improved by approximate 6% on the unseen testing data, the GAN generated synthetic data has significant positive impact on improving the classification performance of the deep residual model for malware classification.

The remainder of the paper will be organized as follows. At first, the related work section describes the recent malware classification work, the methods section introduces the dataset we use, the structures of the GAN model and the 18-layer residual network model. In the results section, our experiment results will be discussed. In the conclusion section, our work in this paper and future work will be concluded.

2 RELATED WORK

The previous work on malware classification can be generally classified into two categories: non-machine learning methods and machine learning methods.

The traditional non-machine learning malware classification methods are mostly heuristic and signature-based. There are two types of traditional malware analysis methods: the static methods and dynamic methods. The static methods extract the malware features from the static malware bytecodes, such as processor instructions, null terminated strings and library imports; the dynamic methods extract the features while the code is being executed, collecting information how the executed codes interact with the operating system such as system API calls or interactions with the other OSs and the network. The feature extracting procedure in the traditional malware detection approaches could be time consuming and also rely heavily on domain expert knowledge, it also needs special tools and software environment which could be computation resource consuming as well. Besides, the traditional malware detection methods do not adapt well when the malwares are modified polymorphically or metamorphically by purpose to hide the real code when evolve or propagate.

In order to address the limitation of the traditional malware classification methods, inspired by the fact that the variants of malware families are sharing the similar code patterns, some machine learning methods such as the Support Vector Machine (SVM) by Sahs and Khan (2012), Naïve Bayes classifier by Firdausi et al. (2010), Kernel Machines by Shankarapani et al. (2010) and Random Forests by Dahl et al. (2013) are applied into the applications of malware classification. However, the major drawback of these methods is that the engineered hand-extracted features are still needed in these methods, furthermore these methods use shallow learning techniques, they are not scalable to the new growing malware samples. To tackle this problem, more sophisticated machine learning methods by using deep learning models such as deep convolutional neural networks by Kalash et al. (2018), deep residual network by Lu et al. (2019), deep belief network networks by Ding et al. (2016), and deep recurrent networks by Pascanu et al. (2015) emerge in recent years. The applications of the deep learning models for malware classification are all related to the strategy proposed by Nataraj et al. (2011) to present malware codes as grayscale images by using GIST method to compute the texture features of the malware codes. By this strategy, the malwares are converted to images, and malware classification problems are all converted to images classification problems. The deep models could learn features from the images automatically, besides, the deep models are capable and

scalable to learn very complex representations as well. However, to train a deep model it needs large set of training data, most of the previous research are mostly focusing on the re-use of current data to train the deep model, in our paper we propose a method which utilize another deep model – deep generative adversarial network model to generate new data to assist training the deep classifier model. Based on our research, none of similar research has been conducted before.

3 METHODS

3.1 Malware as Images

Inspired by the fact that variants in the same malware family have similar code patterns, Nataraj et al. (2011) created a method which uses GIST method to compute texture features of the malware codes to convert the malware codes into images. The patterns and features of the malware codes are well captured in the image format. As shown in Figure 1, the variants from the FYI malware family and Diaplatform malware family, the visual dis-similarity of the malwares in layouts and textures in the same family are very minimal, while the appearances of malwares from different families are very different.

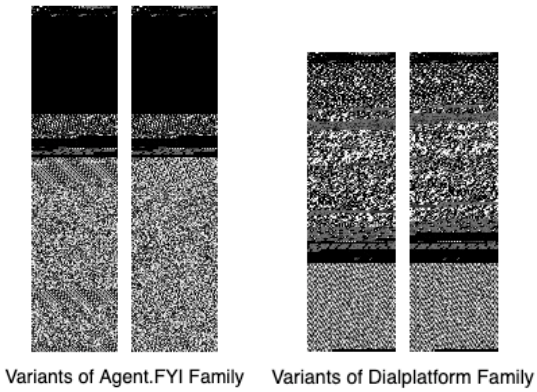


Figure 1: Variants of FYI malware family and Dialplatform malware family.

To convert the malware codes to images, by Nataraj et al. (2011) method it firstly converts the malware binaries to 8-bit vectors (bytecodes).Then it converts the bytecodes into grayscale images with value ranged from 0 to 255, each vector is converted to a pixel with value ranged from 0 to 255. In our experiments, we furtherly convert the grayscale images into 3-channel RGB images by duplicating the grayscale channel for three times, then concatenate all of the three channels to form a RGB image. The codes to images converting procedure is as Figure 2 shown.

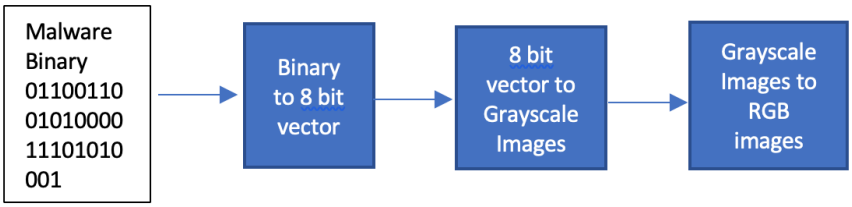


Figure 2: Converting malware binary codes to RGB images.

3.2 Malimg Dataset

In our paper, we use Malimg dataset which is created by Nataraj et al. (2011) as training and testing data, The Malimg dataset was used as the competition benchmark dataset in the Kaggle Microsoft Malware Classification Challenge (BIG 2015), this dataset is adopted in many malware classification researches as benchmark dataset as well. The Malimg dataset consists 25 malware families with different numbers of samples in each class, the details are shown in Table 1. The largest family in this dataset is the Allaple.A malware family, it consists 2949 malware samples; the smallest malware family is the Skintrim.N malware family, it consists 80 malware samples.

Table 1: Malimg malware dataset.

No.	Family	Malware Name	No. of Variants
1.	Worm	Allaple.L	1591
2.	Worm	Allaple.A	2949
3.	Worm	Yuner.A	800
4.	PWS	Lolyda.AA 1	213
5.	PWS	Lolyda.AA 2	184
6.	PWS	Lolyda.AA 3	123
7.	Trojan	C2Lop.P	146
8.	Trojan	C2Lop.gen!G	200
9.	Dialer	Instantaccess	431
10.	Trojan Downloader	Swizzor.gen!I	132
11.	Trojan Downloader	Swizzor.gen!E	128
12.	Worm	VB.AT	408
13.	Rogue	Fakerean	381
14.	Trojan	Alueron,gen!J	198
15.	Trojan	Malex.gen!J	136
16.	PWS	Lolyda.AT	159
17.	Dialer	Adialer.C	125
18.	Trojan Downloader	Wintrim.BX	97
19.	Dialer	Dialplatform.B	177
20.	Trojan Downloader	Dontovo.A	162
21.	Trojan Downloader	Obfuscator.AD	142
22.	Backdoor	Agent.FYI	116
23.	Worm:AutoIT	Autorun.K	106
24.	Backdoor	Rbot!gen	158
25.	Trojan	Skintrim.N	80

The samples in Malimg dataset are all converted to grayscale images with a size of 32 by 32 originally. In our experiment we furtherly convert the samples into 3-channel RGB images by duplicating one channel for three times and then concatenate the three channel as RGB channels. The size of each malware sample in each malware family is converted to 32 by 32 by 3 channels ultimately. Figure 3 shows one group of the converted RGB images of the 25 classes malwares, each small square image named as “malware N” is one random sample from that N class.

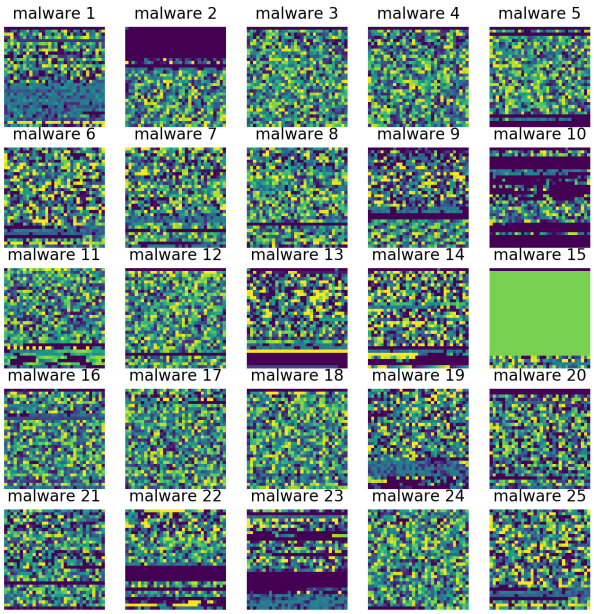


Figure 3: Converted RGB images of the Malimg dataset.

3.3 Deep Residual Network

Many complex visual recognition and image classification tasks show benefits from very deep models. However, when the deep model goes deep, it is getting harder to train the model, the performance of the model starts to decline, the degradation problem is starting to be exposed as well. To solve this problem, He et al. (2016) proposed the deep residual learning framework. The deep residual network is originated from deep convolutional neural network, it uses identity mapping for the shortcut connections between the input and weighted layers in the deep convolutional network. The deep residual network has led to a series of breakthroughs for image classification tasks, it won the ImageNet ILSVRC 2015 classification task, it achieved the state-of-the-art performances in many other computer vision and classification tasks as well, such as winning the 1st places in ImageNet localization, COCO detection, COCO segmentation tasks in ILSVRC and COCO 2015 competition. The strong evidences show that the deep residual framework is generic and highly promising in computer vision problems.

In our proposed method we use an 18-layers deep residual net as the malware classifier. The architecture of the deep residual network is as shown in Figure 4. It consists 17 convolutional layers with filters in different sizes. Following each convolutional layer, there are a normalization layer (function), a rectified linear unit layer (function), and a pooling layer (function). At the last layer there is a fully connection layer and SoftMax layer (function) as the classifier. To avoid overfitting problem, dropout layers are added in between of the convolutional layers and normalization layers.

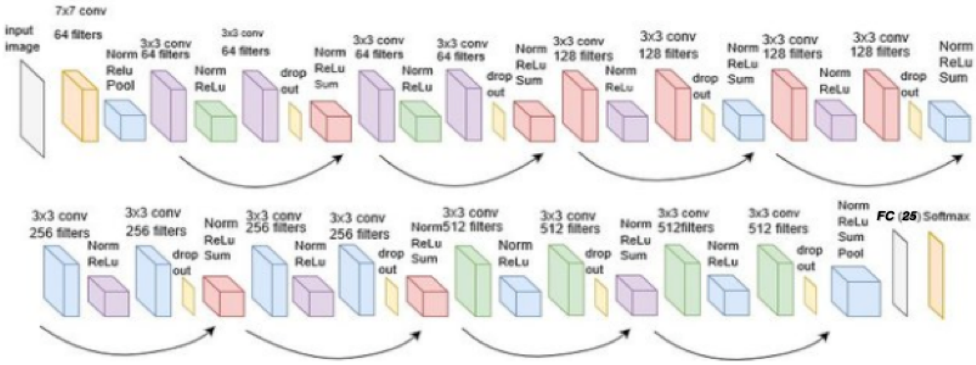


Figure 4: The architecture of 18-layers deep residual network.

3.4 Deep Convolutional Generative Adversarial Network (DCGAN)

To generate the synthetic malware samples, we use a DCGAN model. DCGAN is one of the most popular and successful GAN network. A traditional GAN structure consists two parts of network: the generator and the discriminator, as Figure 5 shown.

A GAN network can be trained to generated images from random noises. At the initialization stage, a series of random noised generated as input to the generator, after the generator generates a fake image, this fake image and the real image will be input to the discriminator. In the discriminator model, it classifies whether the image is real of not: if the image is from the generator then the discriminator is supposed to classify it as fake; if the image is from real data the discriminator is supposed to classify it as real. During the training , the generator is constantly trying to fool the discriminator by generating better fake images, while the discriminator is working to become better to distinguish the real and fake images. The equilibrium of this model is when the discriminator is always guess at 50% confident that the generator output is real or fake.

A GAN network can simultaneously learn from the trained data: the generator captures the potential distribution of the real data and generates synthetic samples; while the discriminator discriminates the difference between the real samples and the synthetic samples as accurately as possible. Multiple convolutional and convolutional-transpose layers are used in the discriminator and generator.

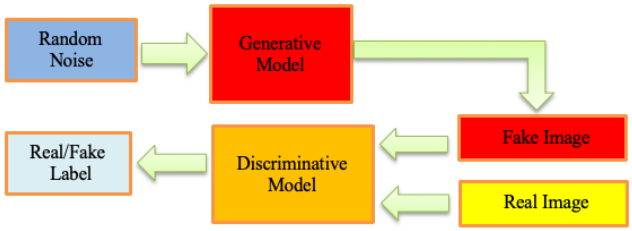
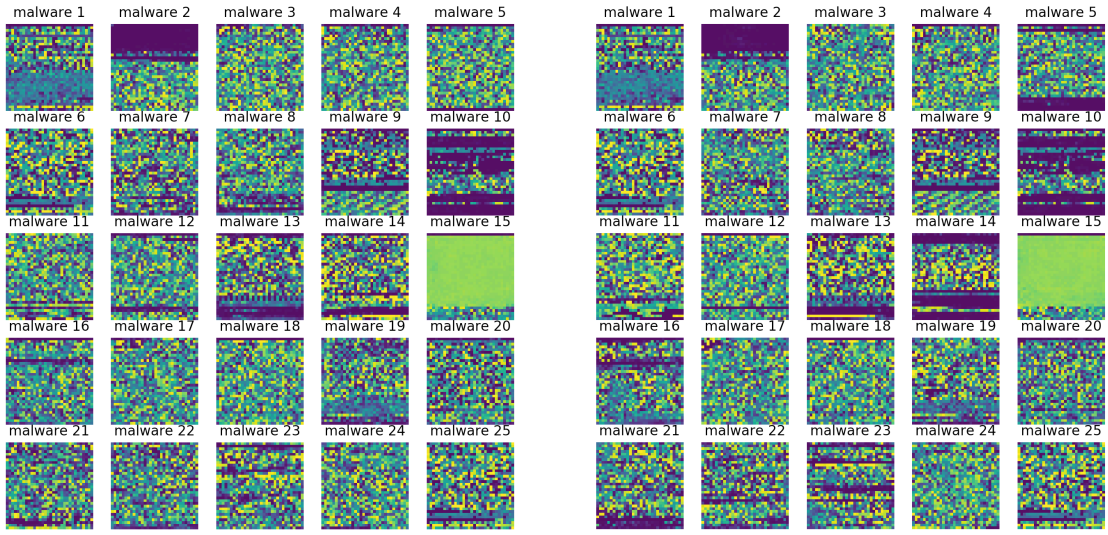


Figure 5: A GAN network model.

3.5 GAN Generated Malware Samples

We train the GAN network for 10000 epochs to generate the fake samples, starting from the 1000 training epochs, we save 25 generated samples for every 100 epochs for each class. So after the training is done, we have 2250 generated synthetic samples for each class. Figure 6 shows four groups of examples of the generated samples, each small square image named “malware N” is one random GAN-generated sample from that N class.



a) GAN-generated samples group 1.

b) GAN-generated samples group 2.

c) GAN-generated samples group 3.

d) GAN-generated samples group 4.

Figure 6: GAN-generated malware samples examples (4 groups).

3.6 Training the Classifier with Synthetic Malware Data

We take the first 30 samples out of each malware class in the Malimg dataset as the unseen testing data. We selected our baseline training data from the data after taking out the testing samples. The selection rules are: if the size of the samples left in that malware class is more than 200, we randomly select 170 samples; if the size of the samples left in that malware class is smaller than 200, we use all left samples in that class as the training data. Then we combined all training data selected from the 25 classes as our baseline training data. The baseline training data are all original real malware data. The classifier trained by this data is our baseline model.

We take 25 GAN-generated synthetic samples as a base unit. Then we take one time, two times, three times, …, until ten times of the unit from each class in the synthetic samples and add it to the corresponding class in the baseline training data. In other words, we take 25, 50, 75, 100, 125, 150, …, 250 fake samples from each class of the GAN-generated data and then add it to the corresponding class of the baseline training data. Then we train the 18-layer residual network by using the ten sets of new training data which are with different percentage of fake synthetic data. Every time we add new synthetic samples, we re-train the model until the residual network converges and then test it on the unseen testing data as described in 3.6. All trainings are carried on GPU Nvidia Tesla V100, we use a batch size of 256, the time for each training epoch is about 15 seconds.

4 RESULTS

4.1 Testing Accuracy

As Section 3.6 described, we use the baseline training data to train the classifier at first, the overall average testing accuracy of the residual network is 0.8413. Then we add 25, 50, 75, 100…250 synthetic data into each class and then re-train and re-test the classifier. The testing results are as shown in the Table 2, by adding different numbers or percentages of the GAN-generated synthetic samples, the overall average testing accuracy on the unseen testing data is improved from 0.84 to 0.90.

Table 2: Testing accuracies by adding different percentage of GAN-generated synthetic data.

Synthetic Data Samples	0	25	50	75	100	125	150	175	200	225	250
Average Testing Accuracy	0.84	0.86	0.87	0.83	0.88	0.88	0.89	0.90	0.89	0.89	0.90

4.2 Precisions, Recalls, and f1-scores

We also use the precision, recall and f1-scores to evaluate the performance of the deep residual network. Generally for each class we use precision to show how precise/accurate of the deep residual network model. We use recall to calculates how many of the actual positives the model catches by labeling it as positive. We use f1-score function check the balance between precision and recall when there is a uneven class distribution. The calculation are as the following formulas shown:

$$precision = true\ positive/(true\ positive + false\ positive),$$

$$recall = true\ positive/(true\ positive + false\ negative),$$

$$f1 - score = 2 \times (precision \times recall)/(precision + recall).$$

By expanding the training data samples using the GAN-generated synthetic malware data, the precision, recall and f1-scores for most of the classes are improved as well, especially for the malware families with small samples such as the Agent.FYI family, C2Lop.P family. Table 3 shows the comparison of the precisions, recalls and f1-scores between the results with and without using 250 synthetic malware samples in each class. All results are testing on the unseen testing data as described in Section 3.6, 30 testing samples from each class.

Table 3: Malware classification result by deep residual network.

	Precision 0/250	Recall 0/250	f1-score 0/250	Testing Samples Number
Allaple.L	1.00/1.00	1.00/1.00	1.00 /1.00	30
Allaple.A	1.00/1.00	1.00/1.00	1.00/1.00	30
Yuner.A	0.54/0.62	0.63/0.60	0.58/0.61	30
Lolyda.AA 1	0.68/0.65	0.87/0.80	0.76/0.72	30
Lolyda.AA 2	0.94/1.00	1.00/1.00	0.97/1.00	30
Lolyda.AA 3	1.00/1.00	1.00/1.00	1.00/1.00	30
C2Lop.P	0.42/0.52	0.60/0.53	0.49/0.52	30
C2Lop.gen!G	0.55/0.61	0.20/0.63	0.29/0.62	30
Instantaccess	1.00/1.00	0.93/0.97	0.97/0.98	30
Swizzor.gen!l	1.00/1.00	1.00/1.00	1.00/1.00	30
Swizzor.gen!E	0.86/0.97	1.00/0.97	0.92/0.97	30
VB.AT	1.00/1.00	1.00/1.00	1.00/1.00	30
Fakerean	0.88/0.97	0.97/1.00	0.92/1.00	30
Alueron,gen!J	0.97/1.00	0.97/1.00	0.97/0.98	30
Malex.gen!J	0.97/1.00	1.00/1.00	0.98/1.00	30
Lolyda.AT	0.93/1.00	0.90/0.97	0.92/0.98	30
Adialer.C	0.67/0.81	0.87/0.83	0.75/0.82	30
Wintrim.BX	1.00/1.00	1.00/1.00	1.00/1.00	30
Dialplatform.B	0.94/0.93	0.97/0.90	0.95/0.92	30
Dontovo.A	1.00/0.97	0.97/1.00	0.98/0.98	30
Obfuscator.AD	0.44/0.81	0.37/0.73	0.40/0.77	30
Agent.FYI	0.37/0.72	0.23/0.70	0.29/0.71	30
Autorun.K	0.97/0.97	0.97/1.00	0.97/0.98	30
Rbot!gen	0.91/1.00	0.67/0.83	0.77/0.91	30
Skintrim.N	1.00/1.00	1.00/1.00	1.00/1.00	30
micro avg	0.84/0.90	0.84/0.90	0.84/0.90	750
macro avg	0.84/0.90	0.84/0.90	0.84/0.90	750
weighted avg	0.84/0.90	0.84/0.90	0.84/0.90	750
samples avg	0.84/0.90	0.84/0.90	0.84/0.90	750

5 CONCLUSION

In this paper, we proposed a method by using Generative Adversarial Network (GAN) to generate synthetic malware samples to solve the small data issue in malware classification problem. Our experiment results show that by using GAN-generated synthetic malware samples, the classification performance of the deep residual network is significantly increased by approximately 6%.

Before adding the GAN-generated malware samples, the training data size of some classes is fairly small, such as the Skintrim.N family, it only has 50 original training data. The overall average testing accuracy of the deep residual network is 84%. The classification accuracies of some classes such as the Agent.FYI or Yuner malware family are a lot lower than the other classes. After adding the GAN-generated synthetic malware samples to train the classifier, the overall average testing accuracy of the deep residual

model on the unseen testing data is improved, the highest testing accuracy is found at 90%. The precisions, recalls, and f1-scores of the classes with smaller samples size are also correspondingly improved. By using the GAN model generated synthetic data, the performance of the malware classification model is improved significantly. Besides, by our proposed method we convert the malwares raw bytecodes into RGB images, then use deep learning model for classification. This malware classification method significantly reduced the domain expert knowledge needed comparing to the traditional malware classification methods.

REFERENCES

Chawla, N. V., K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. "SMOTE: Synthetic Minority over-sampling Technique". *Journal of artificial intelligence research 16:*321-357.

Dahl, G. E., J. W. Stokes, L. Deng, and D. Yu. 2013. "Large-scale Malware Classification Using Random Projections and Neural Networks". In *Proceedings of the 2013 International Conference on Acoustics, Speech and Signal Processing,* May 26th – 31st, 2013, Vancouver, Canada, 3422-3426.

Ding, Y., S. Chen, and J. Xu. 2016. "Application of Deep Belief Networks for Opcode Based Malware Detection". In *Proceedings of the 2016 International Joint Conference on Neural Networks,* July 14th – 19th, 2016, Vancouver, Canada, 3901-3908.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. "Generative Adversarial Nets". In *Proceedings of the 2014 Advances in neural information processing systems*, December 8th –13th, 2014, Montreal, Canada, 2672-2680.

Firdausi, I., A. Erwin, and A. S. Nugroho. 2010. "Analysis of Machine Learning Techniques Used in Behavior-based Malware Detection". In *Proceedings of the 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies*, December 2nd – 3rd, 2010, Jakarta, Indonesia, 201-203.

Han, H., W. Y. Wang, and B. H. Mao. 2005. "Borderline-SMOTE: A New Over-sampling Method in Imbalanced Data Sets Learning". In *Proceedings of the International conference on intelligent computing*, August 23rd – 26th , 2005, Hefei, China, 878-887.

He, K., X. Zhang, S. Ren, and J. Sun, 2016. "Deep Residual Learning For Image Recognition". In *Proceedings of the 2016 Institute of Electrical and Engineers Conference on Computer Vision and Pattern Recognition,* June 26th – July 1st , 2016, Las Vegas, Nevada, 770-778.

Kalash, M., M. Rochan, N. Mohammed, N. D. Bruce, Y. Wang, and F. Iqbal. 2018. "Malware Classification with Deep Convolutional Neural Networks". In *Proceedings of the 2018 9th International Conference on New Technologies, Mobility and Security,* February 26th – 28th, 2018, Paris, France,1-5.

Lu,Y., J. Graham, and J. Li. 2019. "Deep Learning Based Malware Classification Using Deep Residual Network". In *Proceedings of the 2019 Modeling, Simulation and Visualization Student Capstone Conference*, April 18th, 2019, Suffolk, Virginia.

Nataraj, L., S. Karthikeyan, G. Jacob, and B. S. Manjunath. 2011. "Malware Images: Visualization and Automatic Classification". In *Proceedings of the 8th International Symposium on Visualization for Cyber Security,* July 20th, 2011, Pittsburg, Pennsylvania, 4.

Pascanu, R., J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, A. 2015. "Malware Classification with Recurrent Networks". In *Proceedings of the 2015 International Conference on Acoustics, Speech and Signal Processing*, April 19th – 24th, 2015, Brisbane, Australia, 1916-1920.

Sahs, J. and L. Khan. 2012. "A Machine Learning Approach to Android Malware Detection". In *Proceedings of the 2012 European Intelligence and Security Informatics Conference*, August 22nd – 24th, 2012, Odense, Denmark, 141-147.

Shankarapani, M., K. Kancherla, S. Ramammoorthy, R. Movva, and S. Mukkamala. 2010. "Kernel Machines for Malware Classification and Similarity Analysis". In *Proceedings of the 2010 International Joint Conference on Neural Networks,* July18th – 23rd, 2010, Barcelona, Spain, 1-6.

AUTHOR BIOGRAPHIES

YAN LU is a doctoral candidate of Department of Modeling, Simulation and Visualization Engineering at Old Dominion University. She works with deep learning, computer vision. She holds master degree in Computer Science from Virginia Commonwealth University and master degree in Circuit and System from Chinese Academy of Sciences. Her email address is: yxxlu003@odu.edu.

JIANG LI is an Associate Professor in the Department of Electrical and Computer Engineering at Old Dominion University. He received his Ph.D degree in electrical engineering from the University of Texas at Arlington. His research interests include the computer vision, deep learning, and machine learning. His email address is: jli@odu.edu