Malware Classification with Deep Convolutional Neural Networks

Mahmoud Kalash[*†], Mrigank Rochan[*†], Noman Mohammed[†], Neil D. B. Bruce[†], Yang Wang[†], Farkhund Iqbal[‡]
[†]*Department of Computer Science, University of Manitoba*, Winnipeg, Canada
{kalashm, mrochan, noman, bruce, ywang}@cs.umanitoba.ca
[‡]*College of Technological Innovation, Zayed University*, Abu Dhabi, UAE
Farkhund.Iqbal@zu.ac.ae

***Abstract*—In this paper, we propose a deep learning framework for malware classification. There has been a huge increase in the volume of malware in recent years which poses a serious security threat to financial institutions, businesses and individuals. In order to combat the proliferation of malware, new strategies are essential to quickly identify and classify malware samples so that their behavior can be analyzed. Machine learning approaches are becoming popular for classifying malware, however, most of the existing machine learning methods for malware classification use shallow learning algorithms (e.g. SVM). Recently, Convolutional Neural Networks (CNN), a deep learning approach, have shown superior performance compared to traditional learning algorithms, especially in tasks such as image classification. Motivated by this success, we propose a CNN-based architecture to classify malware samples. We convert malware binaries to grayscale images and subsequently train a CNN for classification. Experiments on two challenging malware classification datasets, Malimg and Microsoft malware, demonstrate that our method achieves better than the state-of-the-art performance. The proposed method achieves 98.52% and 99.97% accuracy on the Malimg and Microsoft datasets respectively.**

***Index Terms*—Malware classification, convolutional neural networks, deep learning.**

I. Introduction

Malware is malicious software (e.g. viruses, worms, trojan horses, and spyware) that damages or performs harmful actions on computer systems [1]. In this Internet-age, many malware attacks happen that pose serious security threats to financial institutions and everyday users. Fig. 1 presents statistics of malware over the last 10 years. It is clear that the total number of instances of malware has drastically increased over the years. For example, Symantec reported that more than 357 million new variants of malware were observed in 2016 [2]. One of the main reasons for this high volume of malware samples is the extensive use of obfuscation techniques by malware developers, which means that malicious files from the same malware family (i.e. similar code and common origin) are constantly modified and/or obfuscated. In order to cope with the rapid evolution of malware, it is essential to develop robust malware classification techniques that are tolerant of variants of malware files that belong to same family. Towards this endeavor, we propose a deep learning architecture for malware classification.

[*]Both authors contributed equally to this work.

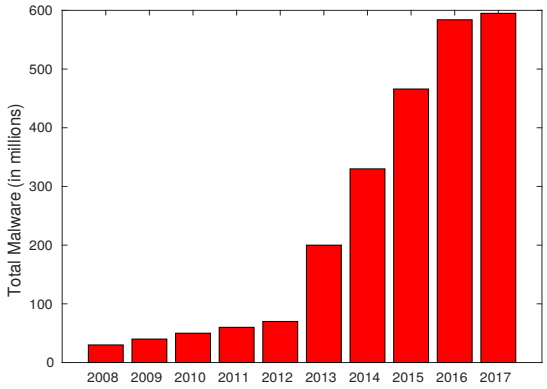

Fig. 1. Last 10 years malware statistics[1]. Total volume of malware has increased drastically over the last 10 years.

Previous research on malware classification suggests that malware samples typically fall into a family that shares common behaviors, i.e. most new malware are variants of existing ones [3]. Hence, the prospect of building a method that can efficiently classify malware based on its family irrespective of being a variant, seems especially fruitful and a means of dealing with the rapid growth of malware.

In this paper, we take a completely different approach to analyze and classify malware compared with traditional methods. We use a Convolutional Neural Network (CNN), a deep learning architecture, to tackle this problem. Recently, deep learning has produced state-of-the-art performance for various tasks in many fields such as natural language processing, computer vision, speech recognition, and bioinformatics. However, the capabilities for applying CNNs has not been well explored in many other fields. One field that may benefit significantly by advances in deep learning is cyber security. With the recent success of deep learning (especially CNNs) in various classification problems, we believe it is possible to classify malware with higher accuracy than any of the shallow learning algorithms such as Support Vector Machines (SVM) [4]. In particular, CNNs have been very successful in problems dealing with images. Motivated by this success, we translate malware classification problem into an image

[1]https://www.av-test.org/en/statistics/malware/

978-1-5386-3662-6/18/$31.00 ©2018 IEEE

classification problem to be addressed using CNNs. Following previous work [5], we represent each malware binary file as a grayscale image (see Section IV-A) and train a CNN architecture for classification. Note that previous work [5] showed that malware belonging to same family are visually similar, which is beneficial with respect to the capacity for a CNN to detect relevant patterns. This is especially true given that the same, or similar code is usually used to generate variants of malware. However, the method proposed in [5] has a number of shortcomings (see Section II).

There is a recent work [6] that uses CNNs for malware classification, but it is still very shallow in its architecture. In computer vision, researchers have shown that deeper CNN architectures (e.g. [7]) are helpful in minimizing error for image classification tasks. In this paper, we take the approach of vision researchers and adopt their technique for malware classification.

Contributions

We make the following main contributions in this paper:

- We develop a deep convolutional neural network (CNN) architecture for malware classification, which is generic in nature, unlike traditional methods. Existing techniques that achieve high accuracy are often tailored for a specific dataset. In contrast, the proposed approach is data independent and learns the discriminative representation from the data itself rather than depending on hand-crafted feature descriptors.
- We perform extensive experiments on two benchmark datasets (Malimg [5] and Microsoft [8]) which demonstrate that our method achieves better than the state-of-the-art performance. We obtain 99.97 % accuracy on the Microsoft dataset while the winning team [9] of the Microsoft Malware Classification Challenge (BIG 2015) [8] achieved an accuracy of 99.87% on the same dataset.

II. Related Work

Previous work on malware classification can be broadly classified into two categories: non-machine learning methods and machine learning-based methods.

Non-Machine Learning Methods: In the past, malware was detected using static or dynamic signature-based techniques [10]. Static analysis uses syntax or structural properties of the program in order to detect malware even before the program under inspection executes. However, malware developers use various encryption, polymorphism and obfuscation techniques [11] [12] to overcome these detection algorithms. In the dynamic approach, malware is executed in a virtual environment and its behavior is analyzed in order to detect harmful actions during or after the program execution. Although dynamic analysis of malware is a promising approach, it is still very complex and time consuming [5]. The major drawback of classical signature-based detection is that it is not scalable and its effectiveness can be undermined with the growing variants of malware [13]. Therefore, we adopt another approach which is based on intelligent machine learning algorithms.

Machine Learning-Based Methods: In order to address the limitations of the aforementioned methods and inspired by the fact that variants of malware families typically share similar behavior patterns [5], anti-malware organizations started to develop more sophisticated classification methods based on data mining and machine learning techniques [14]. These techniques use different feature extraction (i.e. data representation) methods to build more intelligent malware detection systems. They typically use an SVM-based classifier [4], [13], Naïve Bayes classifier [15], or multiple classifiers (naive Bayes, decision trees, SVM) [16]. Our work is related to Nataraj et al. [5] which propose a strategy to represent a malware as a grayscale image and then use GIST [17], [18] to compute texture features. Next, the grayscale malware images are classified using k-nearest neighbor algorithm. The major drawback of this method (and the other methods discussed in this category) is that they use shallow learning techniques which are not very scalable with the growing number of malware samples and also requires to engineer the feature representation by hand. In order to tackle these problems, we develop a deep learning architecture that is robust and more general in nature.

Other techniques are specifically designed to target higher performance on specific datasets such as Microsoft Malware Dataset [8]. For example, Drew et al. [19], [20] applied malware classification on the Microsoft Malware Dataset [8] using modern gene sequence classification tool. Similarly, Ahmadi et al. [21] trained a classifier based on the XGBoost [22] technique. The winning team [9] of the Microsoft Malware Classification Challenge (BIG 2015) [8] used a highly complex combination of features and trained a classifier based on the XGBoost [22] technique. In contrast, our aim in this work is to build a more generic framework (i.e. a technique that is not dataset specific) which can be used with any type of malware sample.

A recent method in [6] is also closely related to our proposed method. This work applies a CNN for malware classification. The author experimented with 3 different architectures by adding an extra block (a block consists of a convolutional layer followed by a Max-pooling layer) each time to its base model. However, their model is still very shallow in nature. In contrast, we explore deeper CNN architecture for malware classification.

III. Background

A Convolution Neural Network (CNN) is a feed-forward neural network that is biologically inspired, specifically by the organization of animal visual cortex [23]. CNN is the current state-of-the-art neural network architecture for image classification problem. CNN is comprised of neurons with learnable weights and biases. CNNs mainly consist of the following three components [24]:

- **Convolutional layers**: These layers apply a certain number of convolution operations (linear filtering) to the image in sequence. Typically these filters extract edge, color, and shape information from the input image. Basically the filters operate on subregions of an image and perform computation such that it produces a single value as output for each subregion. The output (say x) of this layer is typically forwarded to a non-linear function (called ReLU activation) which is defined as $f(x) = max(0, x)$.
- **Pooling layers**: This layer is responsible for downsampling (i.e. reducing the spatial resolution of the input layers) the data produced from convolution layers so that processing time can be reduced, and so that computational resources can handle the scale of the data. This is due to that fact that as a result of pooling, the number of learnable parameters is reduced in the subsequent layers of the network. Max pooling is a commonly used pooling technique that keeps the maximum value in a region (e.g. 2x2 non-overlapping regions of data) and discards the remaining values.
- **Fully connected layers**: This layer performs classification on the output generated from convolution layers and pooling layers. Every neuron in this layer is connected to every neuron present in the previous layer. This type of layer is typically followed by a Dropout layer that improves the generalization capability of the model by preventing over-fitting which is commonly occurring problem in deep learning domain.

IV. Our Approach

A. Visualizing Malware as Image

Malware authors usually change a small part of the previously available code to produce new malware [3]. If we represent malware as an image then these small changes can be easily tracked. Inspired by this and previous work [5], we visualize malware binary files as grayscale images. Fig. 2 demonstrates the process of converting malware binary files to grayscale image.

A given malware binary file is first read in a vector of 8-bits unsigned integers. Next, the binary value of each component of this vector is converted to its equivalent decimal value (e.g. the decimal value for $[00000000]$ in binary is $[0]$ and for $[11111111]$ is $[255]$) which is then saved in a new decimal vector representative of the malware sample. At last, the resulting decimal vector is reshaped to a 2D matrix and visualized as a grayscale image. Selecting width and height of the 2D matrix (i.e. the spatial resolution of the image) mainly depends on the malware binary file size. We use the spatial resolution provided by Nataraj et al. [5] while reshaping the decimal vectors. Malware variants belonging to same family usually have similar texture (i.e. visual appearance). Fig. 3 provides some examples that support this observation.

B. Model Overview

We use a Convolution Neural Network (CNN) for malware classification which we refer as M-CNN. Our model architecture is based on VGG-16 [7]. Input to our network is a malware image and output is set of scores for various malware classes. We then select the class with highest score as our prediction for the given malware. Fig. 4 shows an overview of our proposed CNN architecture (M-CNN).

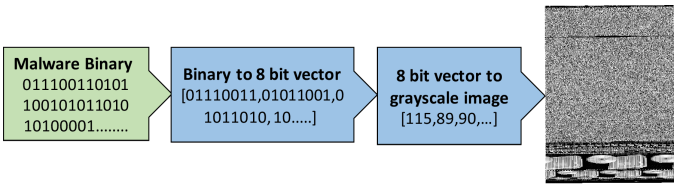

Fig. 2. Overview of malware visulization process. The malware binary file is divided into 8-bit sequences which are then converted to equivalent decimal values. This decimal vector is reshaped and gray-scale image is generated that represent the malware sample.

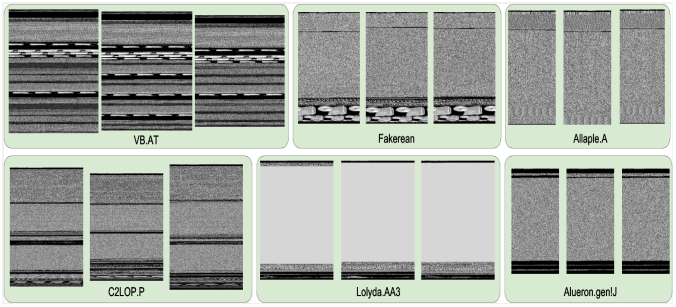

Fig. 3. Samples of malware grayscale images belonging to different malware families. These images are acquired from Malimg Dataset [5]. We can see that malware samples from same family are visually similar. Note that images shown above are rescaled for better visualization.

C. Learning

Since malware is labeled with a class (i.e. family) name, we employ a learning method that optimizes a classification loss. We use cross-entropy loss to train our network. The loss L for a training data y is defined as follows:

$$L = -\log\left(\frac{\exp(f_{y_i})}{\sum_j \exp(f_{y_j})}\right) \tag{1}$$

where f_{y_j} is the score for the j-th class and f_{y_i} is the score for the correct class of the data.

The parameters of the model are learned using stochastic gradient descent (SGD), which tries to minimize the loss incurred on the training data.

V. Experiments

A. Datasets and Experimental Setting

We perform all experiments on the following two challenging malware datasets:

1. Malimg Dataset:

This dataset [5] has a total of 9,339 malware samples that are represented as grayscale images. Each malware sample in the dataset belongs to one of the 25 malware families/classes. Also, the number of samples belonging to a malware family vary across the dataset. In our experiments, we randomly select 90% of malware samples in a family for training and the

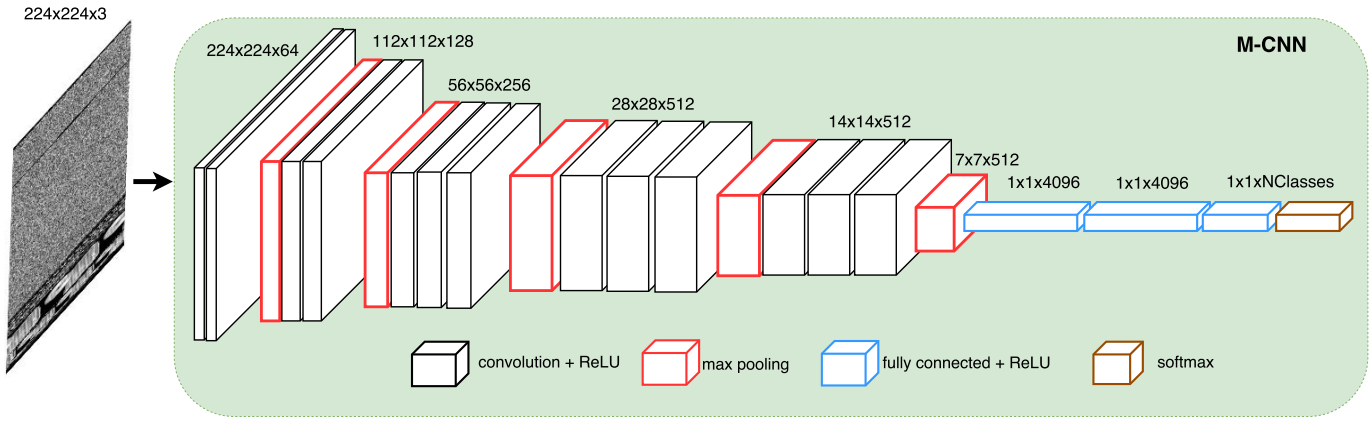

Fig. 4. Overview of our network (M-CNN). The network takes a malware image as input and it produces a set of scores of size equal to the number of malware classes (NClasses) in the dataset as the output. We pick the top-scoring class as the class label for the given malware.

remaining 10% for testing. At the end, we have 8,394 malware samples for training and 945 samples for testing.

2. Microsoft Malware Dataset:

In 2015, Microsoft hosted a Kaggle competition for malware classification [8]. In this challenge, Microsoft released a huge dataset (almost half a terabyte when uncompressed) consisting of 21,741 malware samples. This dataset is divided in two parts, 10,868 samples for training and the other 10,873 samples for testing. Each malware sample belongs to one of 9 different malware families. Like the Malimg dataset, the distribution of malware samples over classes in the training data is not uniform and the number of malware samples of some families significantly outnumbers the samples of other families. There are two files that represent each malware sample, *.bytes* file that contains the raw hexadecimal representation of the file's binary content with the executable headers removed and *.asm* file that contains the disassembled code extracted by the IDA disassembler tool. In our experiments, we only use the *.bytes* files to generate the malware images.

We experiment with two different settings involving this dataset: 1) *Setting-A*: following previous work [19], [20], the original training set (10,868 malware samples) is class-wise randomly divided into two subsets where the first set consists of 90% malware files (9,776) and the rest 10%(1092) is used for testing; 2) *Setting-B*: we follow the original train-test split which is provided by Microsoft (i.e. 10,868 samples for training and the other 10,873 samples for testing).

All experiments are conducted on 64-bit Ubuntu 14.04 Intel(R) Core(TM) i7-5820K CPU (3.30GHz) with 64GB RAM and a NVIDIA Titan X GPU with 12GB memory. We implement our framework using the Torch library [25]. We set the initial learning rate to be 0.001 which is reduced by a factor of 10 every 20 epochs. We set weight decay and momentum to 0.0005 and 0.9 respectively. We initialize the parameters in M-CNN network with the VGG-16 [7] weights. We also randomly shuffle the training data in every epoch.

TABLE I
QUANTITATIVE RESULTS ON MALIMG DATASET.

Method	Accuracy
Nataraj et al. [5]	97.18%
GIST+SVM (ours)	93.23%
M-CNN (ours)	**98.52**%

TABLE II
QUANTITATIVE RESULTS ON MICROSOFT MALWARE DATASET.

Setting-A		*Setting-B* (training set)	
Method	Accuracy	Method	Accuracy
Drew et al. [19]	97.42%	Gibert [6]	99.76%
Drew et al. [20]	98.59%	Ahmadi et al. [21]	99.77%
GIST+SVM (ours)	88.74%	Winner [26] [9]	99.87%
M-CNN (ours)	**98.99**%	M-CNN (ours)	**99.97**%

B. Experiments and Evaluation

Following previous methods, we use accuracy (%) to evaluate our model which indicates the percentage of malware samples labeled correctly in the test data.

Apart from comparing with some previous works, we also implement our own baseline. We extract GIST [17], [18] features (a handcrafted feature that computes texture features) from each malware image. Next, we train a multi-class SVM classifier on both of the datasets which we refer as GIST+SVM.

For the Malimg dataset, we train our proposed network M-CNN for 25 epochs with a batch size of 6. Table I shows the performance of different methods on this dataset. Our baseline method GIST+SVM achieves the accuracy of 93.23%; however, our M-CNN yields the best performance with accuracy of 98.52%.

For the Microsoft dataset, we train M-CNN for 25 epochs with a batch size of 8. Table II presents the results of different approaches on this dataset. On *Setting-A*, M-CNN achieved the best performance with a classification accuracy of 98.99%. On the training set of *Setting-B*, M-CNN model achieves the best

accuracy of 99.97%. Note that this performance is 0.1% higher than the accuracy of the Microsoft Kaggle challenge winner's solution. Also, our model misclasified only 3 malware samples while the winner's model misclassified 15 samples.

Lastly, we evaluate M-CNN on the test set of Microsoft malware dataset, which is same as the test set of *Setting-B*. We submitted our predictions to the evaluation server which evaluates the submissions based on multi-class logarithmic loss $logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j})$ where N is the number of malware samples, M is the number of malware classes, $y_{i,j}$ is 1 if the prediction is correct and 0 otherwise, and $p_{i,j}$ is the predicted probability [8]. Table III shows the performance of different methods in terms of *logloss*. Note that our performance is not directly comparable with the best method as they use both *.bytes* and *.asm* files in their method, whereas we only use *.bytes* files for training our model. We achieve best performance when compared with methods that use only *.bytes* files.

TABLE III
logloss PERFORMANCE ON THE TEST SET OF THE MICROSOFT DATASET.

Method	Files used	Logloss
Gibert [6]	.bytes	0.1176
Drew et al. [19]	.bytes	0.2228
Drew et al. [20]	.bytes + .asm	0.0479
Ahmadi et al. [21]	.bytes + .asm	0.0063
Winner [26] [9]	.bytes + .asm	**0.0028**
M-CNN (ours)	.bytes	0.0571

VI. CONCLUSION AND FUTURE WORK

Malware is increasingly posing a serious security threat to computer systems. It is essential to analyze the behavior of malware and categorize samples so that robust programs to prevent malware attacks can be developed. Towards this endeavor, we have proposed a deep convolutional neural network (CNN) architecture for malware classification. We first convert malware samples to grayscale images and then train a CNN for classification. Experimental results on two benchmark malware classification datasets shows the effectiveness of our proposed method.

VII. ACKNOWLEDGEMENTS

This paper has been accepted for publication in Cybercrime Investigation and Digital forensics Workshop (CID2018). The research is supported in part by the NSERC Discovery Grants (RGPIN-2015-04147) and the University Research Grants Program (URGP) from the University of Manitoba. Work of the first and second author is also supported in part by the Manitoba Graduate Scholarship and the University of Manitoba Graduate Fellowship, respectively.

REFERENCES

[1] "Malware definition," https://techterms.com/definition/malware, 2017, accessed: 2017-03-14.

[2] "Internet security threat report (april 2017)," https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf, accessed: 2017-06-04.

[3] L. Nataraj, S. Karthikeyan, and B. Manjunath, "Sattva: Sparsity inspired classification of malware variants," in *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2015, pp. 135–140.

[4] W. Hardy, L. Chen, S. Hou, Y. Ye, and X. Li, "Dl4md: A deep learning framework for intelligent malware detection," in *Proceedings of the International Conference on Data Mining (DMIN)*, 2016.

[5] L. Nataraj, S. Karthikeyan, G. Jacob, and B. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th international symposium on visualization for cyber security*. ACM, 2011, p. 4.

[6] D. Gibert Llauradó, "Convolutional neural networks for malware classification," Master's thesis, Universitat Politècnica de Catalunya, 2016.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[8] "Microsoft malware classification challenge (big 2015)," https://www.kaggle.com/c/malware-classification, 2017, accessed: 2017-01-30.

[9] "Microsoft malware classification challenge (big 2015) first place team: Say no to overfitting," http://blog.kaggle.com/2015/05/26/microsoft-malware-winners-interview-1st-place-no-to-overfitting/, 2017, accessed: 2017-04-22.

[10] N. Idika and A. P. Mathur, "A survey of malware detection techniques," *Purdue University*, vol. 48, 2007.

[11] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," in *International Conference on Broadband, Wireless Computing, Communication and Applications*. IEEE, 2010, pp. 297–300.

[12] A. H. Sung, J. Xu, P. Chavez, and S. Mukkamala, "Static analyzer of vicious executables (save)," in *Annual Computer Security Applications Conference*. IEEE, 2004, pp. 326–334.

[13] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, "Learning and classification of malware behavior," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2008, pp. 108–125.

[14] M. Siddiqui, M. C. Wang, and J. Lee, "A survey of data mining techniques for malware detection using file features," in *Proceedings of the 46th Annual Southeast Regional Conference on XX*. ACM, 2008, pp. 509–510.

[15] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *IEEE Symposium on Security and Privacy*. IEEE, 2001, pp. 38–49.

[16] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 470–478.

[17] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.

[18] A. Torralba, K. P. Murphy, W. T. Freeman, M. A. Rubin *et al.*, "Context-based vision system for place and object recognition." in *International Conference on Computer Vision*, vol. 3, 2003, pp. 273–280.

[19] J. Drew, T. Moore, and M. Hahsler, "Polymorphic malware detection using sequence classification methods," in *Security and Privacy Workshops*. IEEE, 2016, pp. 81–87.

[20] J. Drew, M. Hahsler, and T. Moore, "Polymorphic malware detection using sequence classification methods and ensembles," *EURASIP Journal on Information Security*, vol. 2017, no. 1, p. 2, 2017.

[21] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*. ACM, 2016, pp. 183–194.

[22] "Xgboost extreme gradient boosting," https://github.com/dmlc/xgboost, 2017, accessed: 2017-04-22.

[23] "Convolutional neural network," https://en.wikipedia.org/wiki/Convolutional_neural_network, 2017.

[24] "Intro to convolutional neural networks," https://www.tensorflow.org/tutorials/layers, 2017.

[25] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, 2011.

[26] "Microsoft malware winners' interview: 1st place, no to overfitting!" https://github.com/xiaozhouwang/kaggle_Microsoft_Malware/blob/master/Saynotooverfitting.pdf, 2017, accessed: 2017-04-22.