A Convolutional Transformation Network for Malware Classification

Duc-Ly Vu[1], Trong-Kha Nguyen[2], Tam V. Nguyen[3], Tu N. Nguyen[4], Fabio Massacci[1], and Phu H. Phung[3,*]

[1]Department of Information Engineering and Computer Science, University of Trento, Italy
Email: {ducly.vu, fabio.massacci}@unitn.it

[2]Hongik University, South Korea
Email: {nguyentrongkha92}@gmail.com

[3]Department of Computer Science, University of Dayton, U.S.A.
Email: tamnguyen@udayton.edu, *Corresponding author: phu@udayton.edu

[4]Department of Computer Science, Purdue University Fort Wayne, U.S.A.
Email: nguyent@pfw.edu

***Abstract*—Modern malware evolves various detection avoidance techniques to bypass the state-of-the-art detection methods. An emerging trend to deal with this issue is the combination of image transformation and machine learning techniques to classify and detect malware. However, existing works in this field only perform simple image transformation methods that limit the accuracy of the detection. In this paper, we introduce a novel approach to classify malware by using a deep network on images transformed from binary samples. In particular, we first develop a novel hybrid image transformation method to convert binaries into color images that convey the binary semantics. The images are trained by a deep convolutional neural network that later classifies the test inputs into benign or malicious categories. Through the extensive experiments, our proposed method surpasses all baselines and achieves 99.14% in terms of accuracy on the testing set.**

I. Introduction

The number of new malware and variants on the Internet has been continuously increasing. According to a technical report from Kaspersky Lab [1], 323,000 new malware files are detected daily, and there are one billion unique malware files in their cloud database. This significant growth is due to the existing of many automatic malware creation toolkits such as Zeus and SpyEye [2]. These kits commonly employ different evading techniques such as encryption, obfuscation, packing to create new malware variants from existing malware samples. These evading techniques allow the variants to bypass the detection of anti-malware scanners [2].

In recent years, machine learning and deep learning techniques have been adopted to the malware classification domain to capture the malware evading techniques [2], [3]. Machine learning algorithms in malware classification are based on a set of features extracted from file samples by static or dynamic analysis techniques. The analyses require either the disassembly code or code execution, and the accuracy of these models is, therefore, dependent on the analysis tools and selected features from the analyses. Also, the analysis and feature selecting process sometimes needs security experts to revise and disambiguate intermediate results, thus cannot be fully automated [1], [4]. An alternative approach to overcoming the limitations mentioned above was the adoption of image processing and classification techniques to the domain of malware. This approach does not require disassembly code or code execution since binary files are converted and mapped to images so that it can be resilient to the known anti-analysis techniques [4], [5].

Although image-based approach for malware classification can avoid anti-analysis techniques, the existing works in this domain e.g., [5], [6], [7], [8], [9], [10] only use simple mapping algorithms to transform malware binaries to images. Thus the semantics of the malware may be disregarded. Our observation is that the more information given to classifiers, the more accuracy rate can be archived. Motivated by this, our work proposes a novel hybrid image transformation method for binaries for malware classification. Our new technique tackles the semantic issue by adding and highlighting essential sequences visually using the entropy technique. A binary file is transformed into a color image where its channels encode semantic information. By visualizing continuous sequences of same semantic entropy values, the image can highlight suspicious sections in a binary for further analysis. For example, the transformed images can show packed or encrypted sections or small cryptographic artifacts like decryption keys or passwords, which is hidden from the human view. Our approach starts with a simple color scheme where bytes are classified into a small number of categories to get an overview of the structure of a file. This approach allows us to select the best color scheme to represent the semantic meaning of the binary data. Then, we take the byte stream and split them into 32-bit blocks and calculate the Shannon entropy on them. Using the second scheme, we can locate encrypted or packed sections. Taken from 256 different byte values, we compress them down into a few common character classes and calculate

978-1-7281-5163-2/19/$31.00 ©2019 IEEE

byte entropy over a sliding window of these selected bytes. Each of these color schemes has its advantages in representing malware behavior. The character class scheme covers the most common padding bytes, nicely highlights strings in malware while the entropy scheme locates encrypted and compressed sections.

To automatically recognize malware variants, their shared patterns should be identified and learned. If malware authors make a small change in the original binary, its image retains the global structure [2]. Thus, image representations of different binaries from the same malware family appear to be similar. Based on these observations, we have developed a deep neural network to learn the global patterns shared among malware.

Our proposed method consists of the following steps in a supervised training phase. First, we divide a given binary executable (benign or malicious) into blocks of sequence and calculate the Shannon entropy value for each block. Next, these blocks are analyzed conditionally and assigned to the corresponding color. The transformed images are fed into convolutional/pooling/fully connected layers. Finally, the network outputs the predicted label. In summary, the contributions of our work are as follows.

- We propose a novel image transformation method to convert binary executables into color images that convey the semantics of the binary data. Semantic information in our context represents the meaning of different file locations, for example, file headers or imports.
- We develop a Convolutional Transformation Network (CTN) for classifying malware based on transformed images.
- We report the experimental results of our proposed method on a dataset of malicious and benign binary samples. Through extensive experiments, our proposed method achieves 99.14% of accuracy, a much higher rate compared with similar work on the same test set.

The composition of this paper is as follows: Section II describes related work on malware analysis, detection, and classification methods. In Section III, we present our proposed method of malware analysis using color images. Section IV illustrates the experimental results. Finally, we conclude our contributions and discuss future work in Section V.

II. Related Work

In this section, we first review the progress of image classification. Then, we summarize the malware classification and the integration of the image transformation into the malware classification.

A. Image classification

Image classification is a fundamental problem in computer vision. In the early stage, Haralick *et al.* [11] describes some easily computable textural features based on gray-tone spatial dependencies and illustrates their application in category-identification tasks. Later, LeCun *et al.* [12] proposed Convolutional neural networks (ConvNets or CNNs) by stacking several convolutional operators into a network. CNNs can create a hierarchy of progressively more abstract features and show a good performance on hand-writing digit classification. However, the limitation of hardware resource resources has restricted CNNs from further investigation. Therefore, there exist many works using hand-crafted features for image classification. On the global scale, GIST [13] is computed over the entire image as a global image descriptor for scene classification. On the local scale, Lowe [14] introduced SIFT feature extracted from interest points. Similarly, Dalal *et al.* [15] proposed a histogram of gradients (HOG) which can be used for both image classification and object detection in a sliding window manner.

Recently, along with the development of GPUs, CNNs were resurrected in deep learning for image classification. Krizhevsky *et al.* [16] trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. The network has many layers, namely, convolution layers and max-pooling layers, and fully-connected layers with a final 1000-class softmax. Furthermore, there are many extensions to deeper networks for higher performance in terms of classification rate [17], [18], [19].

B. Malware Classification

Malware classification is a method to detect whether a given software program is malicious. Conventional techniques in the anti-malware industry used signature-based algorithms, however, these techniques cannot detect new malware or modified malware using evading techniques such as encryption, packing, polymorphism, obfuscation, and meta-morphism [2], [3]. To detect these new type types of malware, in recent years, there have been various efforts to adopt advanced machine learning techniques in the domain of malware classification. In this subsection, we summary the latest efforts in this area. We also highlight several works closely related to our work that adopted image classification techniques to detect malware.

1) Feature-based Approach: These classification techniques first extract various features from file samples and use these features to train the classifier using machine learning methods. Feature extraction can be performed by static analysis, dynamic analysis, or a hybrid combination of the two. Static analysis techniques performed a string search on the program to collect some features. Many efforts used static analysis to construct a feature vector for classification such as [20], [21], [22], [23], [24], [25]. The limitations of static analysis are that static-based features suffer from binary obfuscation and are limited in representing true behavior of malware. Moreover, these techniques are platform-specific and only applicable to Windows PE. Various other works used dynamic techniques to extract features from operations on system resources [26], [27], call sequences [28], control flow or function call graph [29]. According to recent findings, e.g., [3], [30], [31] features-based malware classification methods are still facing evading techniques implemented in modern malware.

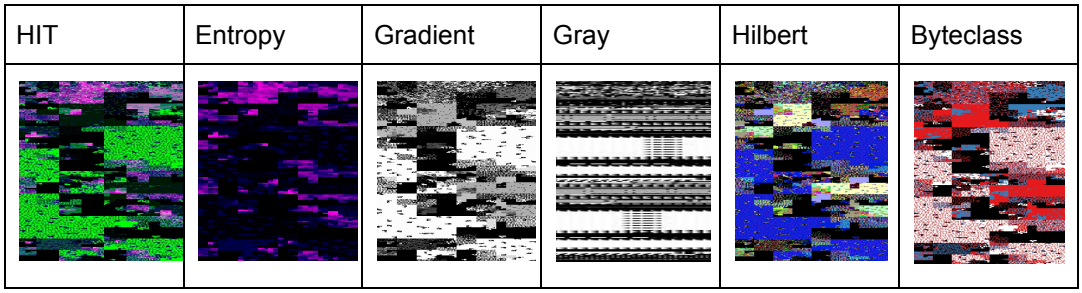


Fig. 1. Visualization of a sample using different transformation schemes.

2) Image-based Approach: There have been several efforts that adapt the techniques in image processing to malware classification. In [2], [4], various visualization techniques, including image processing for malware analysis, have been surveyed. Lately, more efforts are adopting the image-based approach for malware classification, e.g., [7], [8], [9], [10]. A common feature of these efforts is that they transform binary malware samples into different image forms, then image classification techniques are used to classify the malware based on the image representation of the malware. We highlight a few works closely related to ours and categorize them based on the image transformation methods and the representation of the malware in images. In [6], [32], the binary samples are mapped to a vector of 8-bit unsigned integers of byte values, which are reshaped and converted into a gray-scale image in the range [0,255]. The malware classification is calculated using computed texture features from the images using K-nearest neighbors. Han et al. [33], [34] proposed several transformation methods to convert the opcode sequences extracted from malware samples into image matrices represented in RGB-colored pixels. In a later work of the same authors [9], binaries are converted into bitmap images, which are then converted to entropy graphs to calculate the similarities. This approach however, only works for Windows PE file because it needs PE header information to decide sections to be converted. Besides, this method cannot deal with packed samples. Liu et al. [8] proposed a method to transform disassembly files to gray-scale images, which are later compressed and mapped into feature vectors for classification using K-means and diversity selection. In [10], malware is executed on a virtual machine to capture user-mode API calls, which are then sorted assigned a color based on their maliciousness. The most malicious APIs such as DeleteFileA are assigned hot colors (stars from red (1,0,0) in RGB) while the most benign APIs are mapped to cold colors (the coldest color is blue (0,0,1) in RGB). The API color points at the time they appear in log trace are rendered into an image that is used to classify the maliciousness. Kancherla *et al.* [35] plots raw bytes values into 2-D dimensional images and extract intensity, Wavelet, and Gabor-based features. These features are then fed into the SVM classifier to do malware classification. Our work stands apart from the literature by introducing a novel transformation method to convert binaries to images with multiple layers. Thanks to this approach, more features of binaries are carried in the images, resulting in a higher accuracy rate of classification compared with existing works.

III. System design

In this section, we introduce our proposed Convolutional Transformation Network (CTN) for malware classification. Our network consists of two major components, namely, input transformation and convolutional network-based classification.

A. Motivation

Malware uses obfuscation technique techniques to bypass Antivirus and hide its malicious activities. To better capture their behavior, we not only use static features like *strings*, *imports* but also need to understand the encoding techniques. The output of the malware detection system can give a better intuition of malware to users rather than giving a single decision. In other words, it is beneficial to point out suspicious section sections in a program, and security analysts can perform further analysis on them. To achieve these goals, we first analyze the raw byte contents of programs and split it into blocks of byte sequences. We then calculate entropy for each block and transform it into color images. The color images have been proven to be more effective than the grayscale counterpart by the work [36].

B. Input Transformation

We explore multiple ways to encode and transform binary inputs into images. Example images generated by these methods are illustrated in Fig. 1. First, we start with a simple technique called the *Byte class*

1) Byte class: This scheme only includes information about *strings*. Specifically, a character belongs to one of the four categories: the lowest byte value (0), the highest, lowest byte value, the printable strings, and non-printable strings. For example Tab (09), newline (0a) and carriage return (0d) are considered to be texts. This method can be used to get an overview of the file structure.

2) Gradient based: The Gradient color scheme is similar to the byteclass method except that they vary colors with byte ordinal values, which is from 0 to 255.

This scheme can reveal structural details that do not appear in the byteclass scheme.

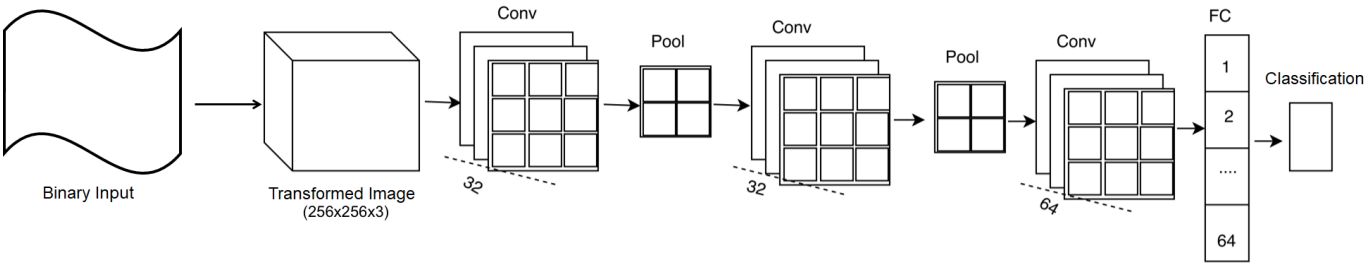


Fig. 2. Pineline of our Convolutional Transformation Network for malware classification.

TABLE I
MAPPING TABLE FOR 8 RANGES

Byte value	Bytes
0	RGB(r, 0, b)
255	RGB(r, 255, b)
[a-w]	RGB(r, 126, b)
[A-W]	RGB(r, 64, b)
[0-9]	RGB(r, 32, b)
special character	RGB(r, 16, b)

3) Hilbert: The detail color scheme assigns a color to each different byte value. It tries to maximize the difference between colors, while at the same time keeping colors for bytes that are close in value as similar as possible. To balance these two conflicting constraints, we again resort to the Hilbert curve. In this paper, we project the 1-dimensional sequence of byte values into a 3-dimensional Hilbert curve traversal of the RGB color cube.

4) Entropy: Entropy is used to detect packed or encrypted malware [37]. In this method a file is read from the beginning, and divided into blocks of byte sequences. Using the Shannon Entropy [38]:

$$H(X) = -\sum_{i=0}^{N-1} p_i log_b p_i \quad (1)$$

where X is a random variable of 256 symbol map of N; $i = 0, \ldots, 255$; block size b is the block size. The probability mass function, p_i is the probability of byte value i within a given byte block. Entropy values are then rescaled to between 0 and 1. We assign entropy values to printable and encoded strings, and transform them into color RGB images.

C. Hybrid Image Transformation (HIT)

We extend the entropy color scheme in the previous section to capture more semantic information about PE files. By encoding byte values in a wide range, HIT can not only capture obfuscation information but also be able to represent semantic information in the file headers like imported functions or libraries. We encode the semantic information into the green channel of the RGB image. To select the best number of partitions in the 256 symbols, we base on the observation that information in the PE headers exists in a visible form of strings and numbers. We start by splitting the symbol range into four smaller ranges of lowest, highest, printable, and other bytes, and keep splitting the range by a binary value 2^n where n is the index of partition until getting the best performance. We reserve the red and blue as same as the entropy method and put more light on the green channel on standard characters. Table I illustrates a typical example of byte splitting and encoding. The motivation behind our color scheme is that a detailed image can improve classification performance [39] and by using more ranges of byte values metadata information such as PE headers, in an executable can be visually identified. As almost malware samples are packed to hinder the analysis, HIT can be applied to detect not only obfuscation patterns and malicious indicators in executables. A regular binary has lower entropy than a packed binary since it follows a software coding standard and contains printable characters. Packed malware has higher entropy than benign since obfuscated or packed makes bytes randomly.

We store entropy information into the red and blue channels while put string information into the green channel as it is most sensitive to human vision [40] and take the highest coefficient value in image grayscale-color conversion. In particular, our HIT method defines entropy value for red and blue and fixed green value by a binary value 2^n, where n is the index of partition. Thus, HIT can output a lower entropy value that makes red and blue lower, and a pixel tends to green. In this way, regular files have more green pixel pixels than malicious files, which contain higher entropy due to the higher red/blue values.

Designing HIT, however, requires selecting the number of partitions. If we divide the color range into many partitions, the output pixels in the image are going to be random since the patterns were removed. Furthermore, when the number of partitions increases, the classifiers learn patterns from an image because it contains random pixels or the training process are easier to be overfitted. We do a heuristic search for the best cut.

D. Network Layers

There exist standard convolutional/pooling layers that widely used in image classification such as AlexNet [16]. In our work, we design a simple neural network that can be used to extract features from images. In particular, our CTN model contains three convolutional layers, two pooling layers, and one fully connected layer, respectively, to learn to classify

TABLE II
PERFORMANCE COMPARISON. THE BEST PERFORMANCE OF EACH CATEGORY IS HIGHLIGHTED WITH BOLDFACE FONT.

Scheme	Training (GIST)	Validation (GIST)	Training (CNN)	Validation (CNN)
Class	90.78	81.81	94.50	94.40
Gray	**93.57**	**94.27**	97.62	95.31
Hilbert	88.88	82.32	93.42	96.61
Gradient	88.41	82.91	97.57	93.23
Entropy	88.76	77.77	97.42	93.88
HIT	91.15	84.34	**98.82**	**99.14**

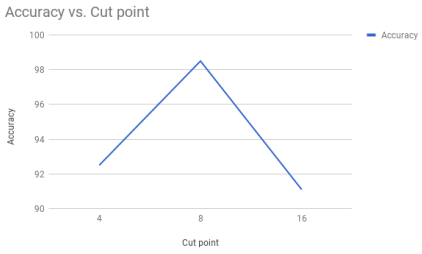
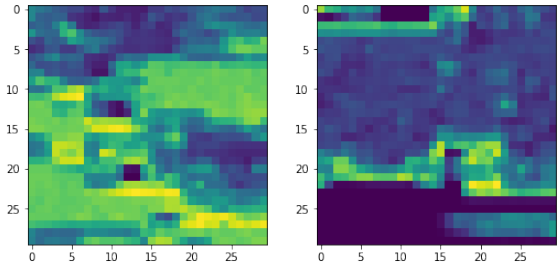


Fig. 3. Detail selection



(a) Benign average (b) Malicious average

Fig. 4. Mean output of benign (left) and malicious (right) samples.

transformed images automatically. Note that the input of our network is the transformed images, as discussed earlier. Fig. 2 represents network operators used in our model.

IV. EXPERIMENTS

A. Dataset

For the evaluation, we collected malware samples from Virusshare [41] and Windows executable software as benign files. We utilize the Microsoft Software Removal Tool [42] to label the samples. Our dataset contains 525 malicious and 525 benign selected samples. We further partition the dataset into two parts: the training set (80%), the validation set (20%). We have performed our experiments on the training and testing phases, each of which has been repeated ten times to get the average results. We refer interested readers to an extended version of this work [43], where we present revised methods with a large-scale dataset evaluation.

B. Experimental Results

We first conduct the parameter selection. As shown in Fig. 3, our proposed model achieves the best performance at the cut point 8. In case we increase the cut point to 16, the performance of the model is decreasing as more random pixels cannot be learned well. Also, it slows down and overfits classifiers. Therefore, we adopt the cut point 8 for the rest of our experiments.

Table II demonstrates the performance of different image transformation methods and different image features. CNN generally performs better than GIST. Our HIT performs the best on both training and test sets. That means the HIT can be able to generalize well on unseen samples. The performance on the training set is generally better than the one on the validation. There is one exception that the entropy transformation performs better with new unseen data on CNN. Regarding the GIST feature, the gray image transformation performs the best with 94.27%. While using CNN features, HIT reaches the top performance with **99.14%** accuracy. The results indicates that using color images with CNN architecture is better than gray scale images.

Fig. 4 visualizes the output average taken from the last network of our model. We can observe that the mean outputs of benign and malicious samples are different. Fig. 4(a) shows that benign samples on average have almost locations in the green regions, while malicious samples have more sections in the dark blue regions, as shown in Fig. 4(b). It indicates that malware is evolved with obfuscation techniques to hide visible indicators.

V. CONCLUSIONS

In this paper, we present a Convolutional Transformation Network for malware classification based on the combination of deep learning and the conversion of binary files into color images. In other words, we cast the malware classification problem into the image classification task. We improve the accuracy rate by enhancing the image color coding. The results of malware classification show that our method achieves over 99.14% regarding accuracy surpassing all the baselines.

In the future, we would like to extend our work to the malware segmentation problem to detect the specific malicious segments inside malware programs. Also, we aim to investigate our work on polymorphism malware classification.

ACKNOWLEDGEMENTS

The project leading to this paper has received funding from the European Union's Horizon 2020 research and innovation

programme under grant agreement No 675320 (NeCS: European Network for Cyber Security).

REFERENCES

[1] Kaspersky Lab, "Kaspersky Lab Number of the Year 2016: 323,000 Pieces of Malware Detected Daily," https://goo.gl/ELzMyu, December 2016.

[2] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Comput. Surv.*, vol. 50, no. 3, pp. 41:1–41:40, Jun. 2017. [Online]. Available: http://doi.acm.org/10.1145/3073559

[3] H. S. Anderson, A. Kharkar, B. Filar, and P. Roth, "Evading Machine Learning Malware Detection," in *Black Hat USA 2017, July 22-27, 2017, Las Vegas, NV, USA*, 2017.

[4] M. Wagner, F. Fischer, R. Luh, A. Haberson, A. Rind, D. A. Keim, and W. Aigner, "A Survey of Visualization Systems for Malware Analysis," in *Eurographics Conference on Visualization (EuroVis) – STARs*, R. Borgo, F. Ganovelli, and I. Viola, Eds. The Eurographics Association, 2015.

[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] L. Nataraj and B. Manjunath, "SPAM: Signal Processing to Analyze Malware," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 105–117, Mar 2016.

[7] A. Makandar and A. Patrot, "Malware class recognition using image processing techniques," in *2017 International Conference on Data Management, Analytics and Innovation (ICDMAI)*, Feb 2017, pp. 76–80.

[8] L. Liu and B. Wang, "Malware classification using gray-scale images and ensemble learning," in *Systems and Informatics (ICSAI), 2016 3rd International Conference on*. IEEE, 2016, pp. 1018–1022.

[9] K. S. Han, J. H. Lim, B. Kang, and E. G. Im, "Malware analysis using visualized images and entropy graphs," *International Journal of Information Security*, vol. 14, no. 1, pp. 1–14, 2015.

[10] S. Z. M. Shaid and M. A. Maarof, "Malware behavior image for malware variant identification," in *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*. IEEE, 2014, pp. 238–243.

[11] R. M. Haralick, K. S. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 3, no. 6, pp. 610–621, 1973.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] 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.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[15] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of Advances in Neural Information Processing Systems Conference*, 2012, pp. 1097–1105.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[20] J. Z. Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild," *Journal of Machine Learning Research*, vol. 7, no. Dec, pp. 2721–2744, 2006.

[21] 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.

[22] R. Tian, L. Batten, R. Islam, and S. Versteeg, "An automated classification system based on the strings of trojan and virus families," in *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*. IEEE, 2009, pp. 23–30.

[23] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 646–656, 2013.

[24] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3422–3426.

[25] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Malicious and Unwanted Software (MALWARE), 2015 10th International Conference on*. IEEE, 2015, pp. 11–20.

[26] 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.

[27] W. Huang and J. W. Stokes, "MtNet: a multi-task neural network for dynamic malware classification," in *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2016, pp. 399–418.

[28] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Proceedings of 29th Australasian Joint Conference on Artificial Intelligence*. Springer International Publishing, 2016, pp. 137–149.

[29] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 3854–3861.

[30] Y. Q. Weilin Xu, David Evans, "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks," in *2018 Network and Distributed System Security Symposium (NDSS)*, Feb 2018.

[31] D. E. Weilin Xu, Yanjun Qi, "Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers," in *2016 Network and Distributed System Security Symposium (NDSS)*, Feb 2016.

[32] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, "A comparative assessment of malware classification using binary texture analysis and dynamic analysis," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*. ACM, 2011, pp. 21–30.

[33] K. Han, J. H. Lim, and E. G. Im, "Malware analysis method using visualization of binary files," in *Proceedings of the 2013 Research in Adaptive and Convergent Systems*. ACM, 2013, pp. 317–321.

[34] K. Han, B. Kang, and E. G. Im, "Malware analysis using visualized image matrices," *The Scientific World Journal*, vol. 2014, 2014.

[35] K. Kancherla and S. Mukkamala, "Image visualization based malware detection," in *Computational Intelligence in Cyber Security (CICS), 2013 IEEE Symposium on*. IEEE, 2013, pp. 40–44.

[36] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek, "Evaluating color descriptors for object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1582–1596, 2010.

[37] R. Lyda and J. Hamrock, "Using entropy analysis to find encrypted and packed malware," *IEEE Security & Privacy*, vol. 5, no. 2, 2007.

[38] C. E. Shannon, "Prediction and entropy of printed english," *Bell Labs Technical Journal*, vol. 30, no. 1, pp. 50–64, 1951.

[39] K. Van De Sande, T. Gevers, and C. Snoek, "Evaluating color descriptors for object and scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1582–1596, 2010.

[40] B. Fortner and T. E. Meyer, *Number by colors: A guide to using color to understand technical data*. Springer Science & Business Media, 2012.

[41] virusshare. (2019) Virusshare.com. [Online]. Available: https://virusshare.com/

[42] Microsoft. (2019) Microsoft windows malicious software removal tool. [Online]. Available: https://www.microsoft.com/enus/download/malicious-software-removal-tool-details.aspx

[43] D.-L. Vu, T.-K. Nguyen, T. V. Nguyen, T. N. Nguyen, F. Massacc, and P. H. Phung, " HIT4Mal:Hybrid Image Transformation for Malware Classification," *Transactions on Emerging Telecommunications Technologies (ETT)*, 2019, http://doi.org/10.1002/ett.3789. To appear.