Detecting Mallicious Android Application Package

Emil Le
Macalester College

***Abstract*—Cyber-defense has always been a crucial task and a constant effort to update algorithms that are precise, accurate and fast against cyber-threat. This paper tries to transform the quantifier of malicious executable binaries into images and apply Convolutional Neural Network models for classifications. The result yields 75% accuracy based on the CICAndMal2017 datasets. By comparisons with k-NN, there is**

I. Introduction

In terms of software security, one of the important task of malware analysis is to detect the functionality and impact of malicious program. In the past, many methodologies have been using signature or pattern associated with a malicious attack on a computer network or system. This all entail to be a classic classifications tasks that can be optimized by Machine Learning. Since the pattern of executable can be a series of bytes in the file (byte sequence) in network traffic, we can transform the quantifiers of problem into a Computer Vision.

In recent development, there are many prominent tech companies that have been applying the results in Machine Learning's researches in order to challenge such tasks. For example, Darktrace earned $78 million in revenue by June 2018 by implemeting unspervised machine learning algorithm to detect cyber threats.

This fact has given rise to statistical and machine learning based detection techniques, which are more robust to code modification. In response, malware writers have developed advanced forms of malware that alter statistical and structural properties of their code. Such "noise" can cause statistical models to misclassify samples. Within the scope of this paper, I try to detect and classify malicious APK (Android application package) using applications of Machine Learning's Transfer Learning to compete against these tasks.

Transfer Learning is an approach to solve a problem by mapping it to a previous solved problem. More specific, I use several pre-trained results in Computer Vision to apply it on the APK dataset. By treating the binary assembly code as images, I use several different architectures of supervised neural network to see which one have the highest performance. Leveraging the power of such models has been shown to yield strong malware detection and classification results (Yajamanam et al., 2018). I expect that models based on image analysis to be more robust, as compared to models that rely on opcodes. In this paper, we compare image-based deep learning (DL) models for malware analysis to a much simpler non-image based technique.

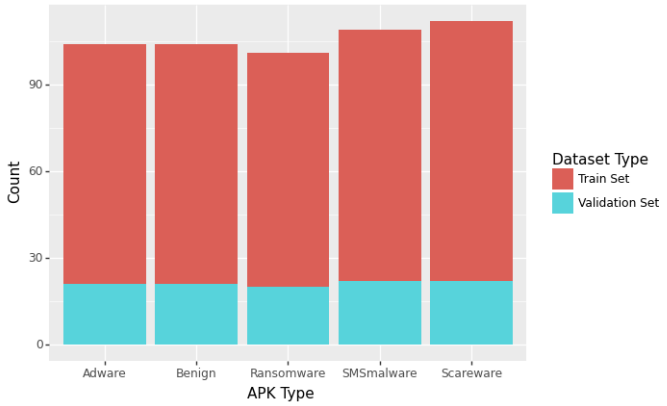

Fig. 1. Distribution of randomized training sets and validation sets

II. Methodology

A. Datasets

I use the dataset CICAndMal2017 collected by Canadian Institute for Cybersecurity (CIC) between December 2017 to December 2018. I believe the dataset CICAndMal2017 is significant due to its recent collection date which might have been adapted to against the aforementioned statistical approaches. CICAndMal2017 consists of more than 10,854 samples (4,354 malware and 6,500 benign) from several sources. These sample malware are divided to 4 categories: Adware, Ransomware, Scareware, SMSmalware. For each APK, there is an CSV files that has been pre-analyzed by CIC, which consists of functionality labels. Due to lack of GPUs to run these algorithms at a large sample size, I only use 500 sampes for the subset which devides into roughly 100 APK for the benign and each types of malware Even by these constrain, to train these model it still typically took Google Colab's GPUs roughly an hour to produce result. For all experiments, the dataset is splitted into the training set and the validation sets with similar distribution, which we can see in Figure 1.

B. Data Preprocessing

The Feature DataFrame will be computed differently for different methodology. Meanwhile, since the datasets already provided the classification of the APK, I only have to do simple Preprocessing to make the Target DataFrame consistent for keras to run its models. The Target (Table I) is simply its benign/malware property and its APK name.

TABLE I
THE TARGET VARIABLE.

Target	Label
Adware	c290b9ecf18c8635165f6dcc10ec14af
Scareware	3d4b4968621c5f42d835447643f37105
Ransomware	b3a694ab3b58de4a944a3d4a03f67c6d
SMSmalware	c124947a5df6b8fd6eee98dc1d63badd
Benign	58a2d00d9c8efa24f9690ee45f793831

C. Converting Binaries to Images

To convert a binary to an image we treat the sequence of bytes representing the binary as the bytes of a grayscale PNG image. In all of our experiments, we use a predefined width of 256, and a variable length, depending on the size of the binary.

Sample images of unrelated binaries are given in Figure 1, while samples from a malware family appear in Figure 2. From these examples, the allure of image-based classification is clear—images tend to smooth out minor within-family differences, while significant (i.e., between family) differences are clearly observed.

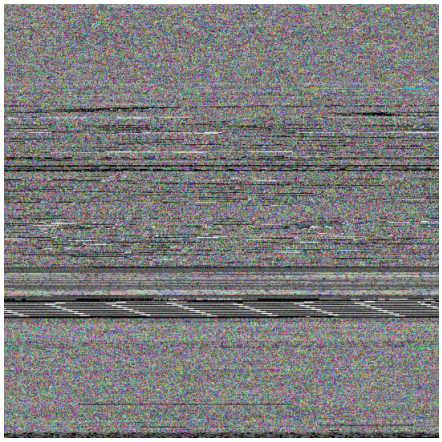

Fig. 2. Benign program

D. Feature Extraction for k-NN

E. Implementation Details

The DL models were implemented using the fast.ai library (Fast.ai, 2018), which is built on top of the PyTorch framework. The choice of this library was influenced by the fact that it incorporates several DL best practices, including learning rate finding, stochastic gradient descent with restarts, and differential learning rates.

For k-NN, we used the popular Scikit-learn library (Pedregosa et al., 2011), which is based on many of the fundamentals described in (Stamp, 2017). The fast.ai library incorporates CUDA support, which allowed us to accelerate the training process by making use of the graphics card.

III. EXPERIMENT AND RESULT

We performed a variety of experiments involving various combinations of datasets, classification level (binary and multiclass), and learning techniques (DL and k-NN). Here, we present results for eight separate experiments, as listed in Table 4. Each experiment represented a specific combination of datasets, classification level, and learning technique. In the remainder of this section, we discuss each of these experiments in some detail. For the DL experiments, that is, experiments 1 through 4 in Table 4, we tested variants of the ResNet model (He et al., 2016), specifically, ResNet34, ResNet50, ResNet101, and ResNext50. We chose ResNet because of its combination of performance and efficiency. ResNet-based architectures won the ImageNet and COCO challenges in 2015. Their key advantage is the use of "residual blocks," which enabled the training of neural networks of unprecedented depth. The models we use were pre-trained on the ImageNet dataset, which contains some 1.2 million images in 1,000 classes.

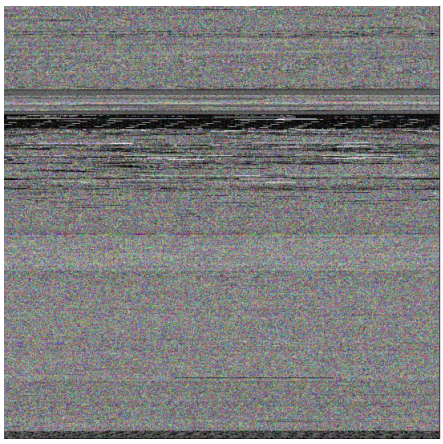

Fig. 3. Malware - Adware

The more complex ResNet variants we experimented with did not yield significant improvement, so we used ResNet34 for all DL experiments reported in this paper. We also tested various combinations of hyperparameters, including the number of epochs, the learning rate, the number of cycles of learning rate annealing, and variations in the cycle length. The training concepts implemented in conjunction with these hyperparameters were cosine annealing, learning rate finding, stochastic gradient descent with restarts, freezing and unfreezing layers in the pre-trained network, and differential learning rates. A description of these techniques and how they are used in concert with the listed hyperparameters is beyond the scope of this paper—the interested reader can refer to (Yajamanam et al., 2018), (Fast.ai, 2018), and (Smith, 2015) for more details.

Perhaps the simplest machine learning technique possible is k-NN, where we classify a sample based on its k nearest neighbors in a given training set. For k-NN, there is no explicit training phase, and all work is deferred to the scoring phase. Once the training data is specified, we score a sample by simply determining its nearest neighbors in the training set, with a majority vote typically used for (binary) classification. In spite of its incredible simplicity, it is often the case that k-NN achieves results that are competitive with far more

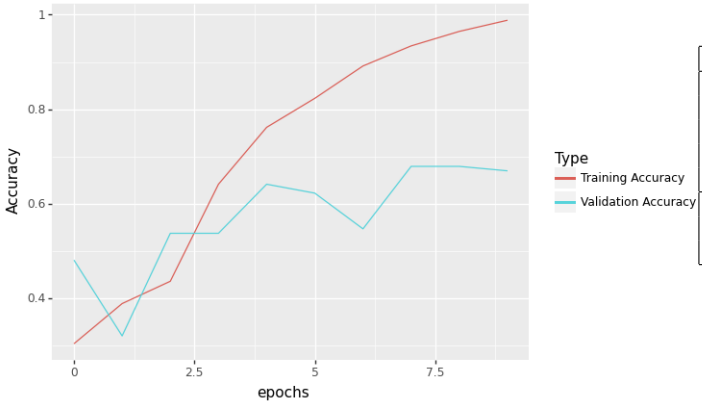

Fig. 4. Accuracy

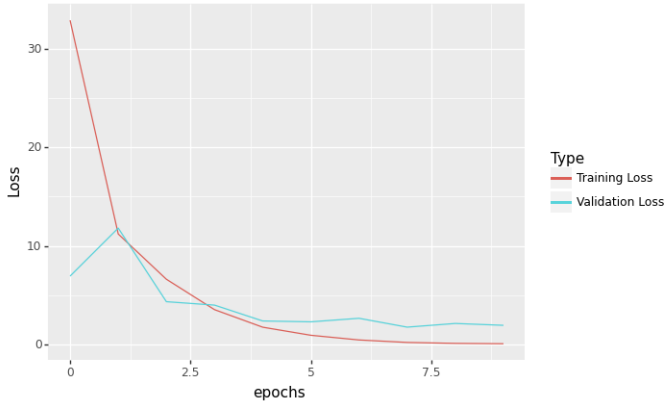

Fig. 5. Loss Function

complex machine learning techniques (Stamp, 2017).

For our k-NN experiments (i.e., experiments 5 and 6), we use Euclidean distance, and hence the only parameter to be determined is the value of k, that is, the number of neighbors to consider when classifying a sample. We experimented with values ranging

For our first set of experiments, we apply the imagebased DL technique outlined above to the Malimg dataset. We consider the following two variations.

For the binary classification problem in experiment 1, we obtained an accuracy of 98.39%, while the multiclass problem in experiment 2 yielded an accuracy of 94.80%. The results of experiment 1 are summarized in Figure 3, while Figures 4 and 5 give the results for experiment 2. These experimental results are comparable to those obtained in (Yajamanam et al., 2018), and serve to confirm our DL implementation.

Experiment 1 For our first experiment, we perform binary classification of malware versus benign, where the malware class is obtained by simply grouping all Malimg families into one malware set. The benign set consists of 3304 Windows samples, which have been converted to images.

We do not have access to the Malimg binary files, so we are unable to compare the DL results for this dataset to alternatives that rely on features extracted directly from executables. Therefore, we next consider the Malicia malware

TABLE II

	Precision	Recall	f1-score	N Obs
Adware	0.73	0.76	0.74	21
Ransomware	0.84	0.76	0.80	21
Scareware	0.63	0.60	0.62	20
SMSmalware	0.74	0.91	0.82	22
Benign	0.84	0.73	0.78	22
Accuracy			0.75	106
Macro avg	0.76	0.75	0.75	106
Weighted avg	0.76	0.75	0.75	106

TABLE III

	Train dataset	Validation dataset
Accuracy	0.9443396	0.75739623
Loss	0.0002561	1.85991073

dataset, which will allow us to compare our image-based DL technique to a simpler k-NN analysis based on non-image features. For the Malicia dataset, we first generate an image corresponding to each binary executable sample in the dataset, as discussed in Section 2.3. Then we perform the analogous experiments to 1 and 2, above, but using the Malicia samples in place of Malimg. Specifically, we perform the following experiments.

Experiment 2 For the corresponding multiclass classification problem, we attempt to classify the malware samples into their respective families, with the Windows benign set treated as an additional "family." Since there are 25 malware families in the Malimg dataset, for this classification problem, we have 26 classes.

Experiment 3 As in experiment 1, we perform binary classification of malware versus benign, but in this case, the malware class consists of all Malicia samples, as images. The benign set consists of the same 3304 Windows samples that were used in experiment 1.

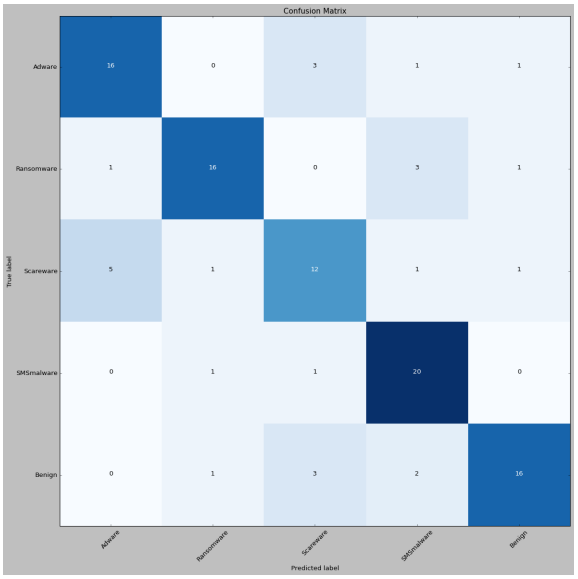

Fig. 6. Confusion Matrix

Experiment 4 For the corresponding multiclass version of this problem, we attempt to classify the Malicia (image) samples into their respective families, with the Windows benign set treated as an additional "family."

IV. Conclusion

In this paper, we treated malware binaries as images and classified samples based on pre-trained deep learning image recognition models. We compared these image-based deep learning (DL) results to a simpler k-nearest neighbor (k-NN) approach based on a more typical set of static features. We carried out a wide variety of experiments, each representing a different combination of dataset, classification level, and learning technique. The multiclass experiments were particularly impressive, with high accuracy attained over a large number of malware families.

Our DL method overall delivered results comparable to previous work, yet it was outperformed by the much simpler k-NN learning technique in some cases. The image-based DL models did outperform k-NN in simulated zero-day experiments, which indicates that this DL implementation better generalizes the training data, as compared to k-NN. This is a significant point, since zero-day malware, arguably, represents the ultimate challenge in malware detection.

There are many promising avenues for future work related to image-based malware analysis. For example, it seems likely that a major strength of any image-based strategy is its robustness. Consequently, additional experiments along these lines would be helpful to better quantify this effect.

Acknowledgment

For the Summary paper submission only, no acknowledgements are allowed.

References

[1] W. H. Cantrell, "Tuning analysis for the high-Q class-E power amplifier," *IEEE Trans. Microwave Theory & Tech.*, vol. 48, no. 12, pp. 2397-2402, December 2000.

[2] W. H. Cantrell, and W. A. Davis, "Amplitude modulator utilizing a high-Q class-E DC-DC converter", *2003 IEEE MTT-S Int. Microwave Symp. Dig.*, vol. 3, pp. 1721-1724, June 2003.

[3] H. L. Krauss, C. W. Bostian, and F. H. Raab, *Solid State Radio Engineering*, New York: J. Wiley & Sons, 1980.

Note: For the Summary paper submission only, references to the authors own work should be cited as if done by others to enable a double-blind review. **Citations must be complete and not redacted, allowing the reviewers to confirm that prior art has been properly identified and acknowledged.**