WHITE PAPER

STAMINA: Scalable Deep Learning Approach for Malware Classification

Li Chen, Ravi Sahita – Intel Corporation
Jugal Parikh, Marc Marino – Microsoft Corporation

April 2020

Introduction

Intel Labs and Microsoft Threat Protection Intelligence Team are collaborating to research the application of deep learning for malware threat detection. The goals of the joint research are:

- Leveraging deep learning techniques to avoid time-consuming manual feature engineering with high accuracy and low false positives.
- Optimizing deep learning techniques in terms of model size and leveraging platform hardware capabilities to optimize execution of deep-learning malware detection approaches.

In this paper, we focused on the first goal, leaving the other areas for future analysis.

Typically, malware analysis using machine-learning techniques can leverage static characteristics of programs and/or dynamic characteristics of programs. For static analysis, observable artifacts of the objects analyzed are utilized for deep learning. For dynamic analysis, the static information is augmented with dynamically generated information derived from execution of the objects (or execution of the programs that handle the objects, such as a PDF file). For this paper, the we have focused on static analysis to allow the broadest possible applicability of the approach to malware classification; we will cover the dynamic analysis techniques in the future. We defined malware as a combination of known malware (previously classified), potentially unwanted applications (PUAs), and unknown binaries (with no known provenance or history).

We studied the practical benefits of applying deep transfer learning from computer vision to static malware classification. Recall that in the transfer learning scheme, we borrowed knowledge from natural images or objects and applied it to the target domain of static malware detection. The training time of deep neural networks is accelerated while high classification performance is still maintained. In this paper, Intel Labs and the Microsoft Threat Intelligence Team have demonstrated the effectiveness of this approach on a real-world user dataset and have shown that transfer learning from computer vision for malware classification can achieve highly desirable classification performance. For this collaboration, we called this approach **STA**tic **M**alware-as-**I**mage **N**etwork **A**nalysis (**STAMINA**), which will be used throughout this paper to refer to this approach.

Malware is a type of software that possesses malicious characteristics to cause damage to the user, computer, or network. Static analysis is a quick and straightforward way to detect malware without executing the application or monitoring the run time behavior. Signature matching, a static analysis technique, is used to match malicious signatures. However, as malware signatures are increasing exponentially every day, signature matching must keep up with malware signatures in order to be effective.

Previously, Intel Labs proposed an enhanced malware detection framework [1] that employs deep transfer learning to train directly on malware images. The approach was motivated by visual inspection of application binaries plotted as grey-scale images: there are textural and structural similarities among malware from the same family and dissimilarities between malware and benign software as well as across different malware families. The paper examined three public benchmarks. Intel Labs collaborated with Microsoft to establish the practical value of this image-based transfer learning approach for static malware classification, based on a real-world data set.

Classical malware detection approaches involve extracting the binary signatures or fingerprints of the malware. However, the rapid increase of signatures, often in exponential growth, makes the signature matching less straightforward. Other approaches include static and dynamic analysis, both of which have advantages and disadvantages. Static analysis disassembles the code, but its performance can suffer from code obfuscation. Dynamic analysis, while able to unpack the code, can be time consuming. Resizing as a preprocessing step does not negatively impact the classification result, since our system trains a very deep neural network to extract the deep-represented features. As seen in the experimental results, our system can outperform many other classifiers and results from prior-art. Furthermore, for malware from the same family, resizing still results in similar patterns.

STAMINA Steps

To recap the proposed method [1], STAMINA essentially consists of four steps: preprocessing (image conversion), transfer learning, evaluation and interpretation. For this study, we describe the first three steps in detail and refer to the reader to [1] for more information on the interpretation stage.

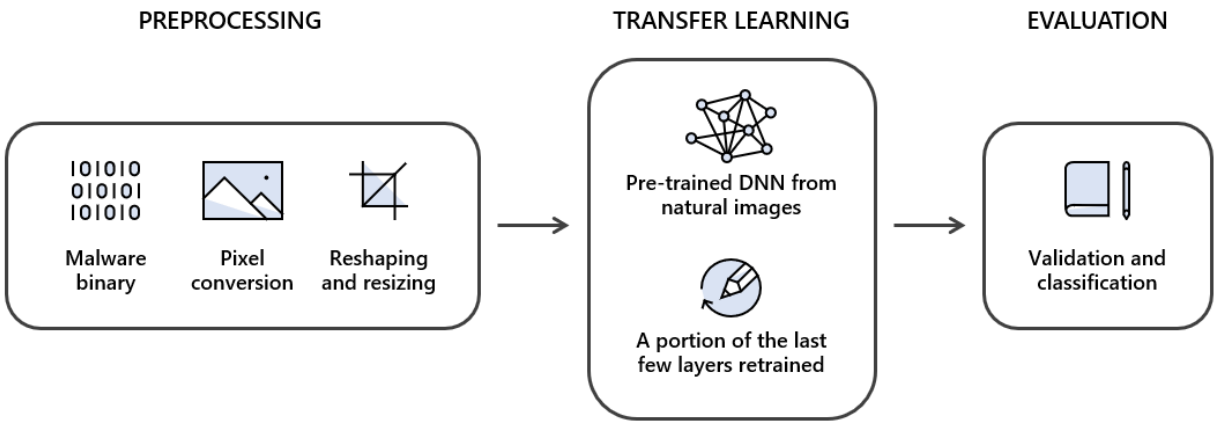


Figure 1: First three steps of the STAMINA method

Preprocessing (Image Conversion)

The image conversion step consists of three sub-steps: pixel conversion, reshaping and resizing. Given a binary application, pixel conversion is straightforward: we read every byte into a value between 0 and 255, directly corresponding to pixel intensity. This step converts a binary into a one-dimensional pixel stream. In order to apply transfer learning and computer vision, we needed to reshape the pixel streams into two dimensions. The width and height were determined by the file size after converting to pixel stream, following an empirically validated table, as shown below. In some prior publications, authors used the file size and image width directly. Our table uses the pixel file size and image width relationship. The pixel file size is a multiplier of the file size, so that the relationships are still linearly scaled. We recommend using this table because it helps set the width and height more concretely.

Pixel File Size	Image Width
Between 0 to 10	32
Between 10 and 30	64
Between 30 and 60	128
Between 60 and 100	256
Between 100 and 200	384
Between 200 and 1000	512
Between 1000 and 1500	1024
Greater than 1500	2048

The image height is calculated as the number of pixels divided by the width. If the height is a decimal number, we rounded it up and padded the extra pixels as zeros.

After reshaping, we considered resizing the images so that they can be used by transfer learning techniques. Resizing as a preprocessing step does not negatively impact the classification result, since our system trains a very deep neural network to extract the deep-represented features. As seen in the experimental results, our system can outperform many other classifiers and results from prior-art. Furthermore, for malware from the same family, resizing still results in similar patterns.

We recommend resizing to 224 or 299 so that the image models trained on ImageNet can be used for fine tuning on the images. There are many ways to resize images. We recommend using the bilinear interpolation or nearest neighbor algorithms for resizing. We advise not to use crop and resize combination since we are dealing with benign and malware images, not natural images.

Transfer Learning Step

Deep learning has demonstrated state-of-the-art performance on large-scale image classification. In particular, transfer learning has been heavily employed in computer vision. The idea of transfer learning is to borrow knowledge learned from a model used in one domain and apply it to another targeted domain. Typically, practitioners take a pre-trained model from a type of image dataset, freeze a portion of the layers, and fine-tune the last few layers on the newly obtained dataset. The advantages of transfer learning include accelerating training time, reducing parameters and architecture search for deep neural networks, and maintaining high classification performance, especially on relatively smaller-sized datasets.

Our proposed method leverages the value of transfer learning to train a highly effective malware classifier for static malware classification. The transfer learning step was done on the malware and benign images produced in the preprocessing step. In practice, due to the limitation of datasets, training an entire deep neural network from scratch can be difficult. What has been done in the computer vision space is that, for specific tasks, models pre-trained on a large number of images are used, and transfer learning is conducted on target tasks. Major transfer learning schemes include using as a feature extractor and fine-tuning the network.

In using as a feature extractor, the last fully connected layer is removed, with the rest of the network treated as a fixed feature extractor. Fine-tuning the network means retraining the top of the pre-trained network and fine-tuning the weights of the pretrained network via continued back propagation step. For this study, using as a fixed feature extractor was not applicable to because the malware classification task is further away from natural image classification.

Our malware dataset, which we describe below, is relatively large - but still small compared with ImageNet and different from the original ImageNet dataset. We expected the cost to train a deep neural network such as Inception or Resnet manageable. However, initializing weights from the pretrained model produces effective classification performance.

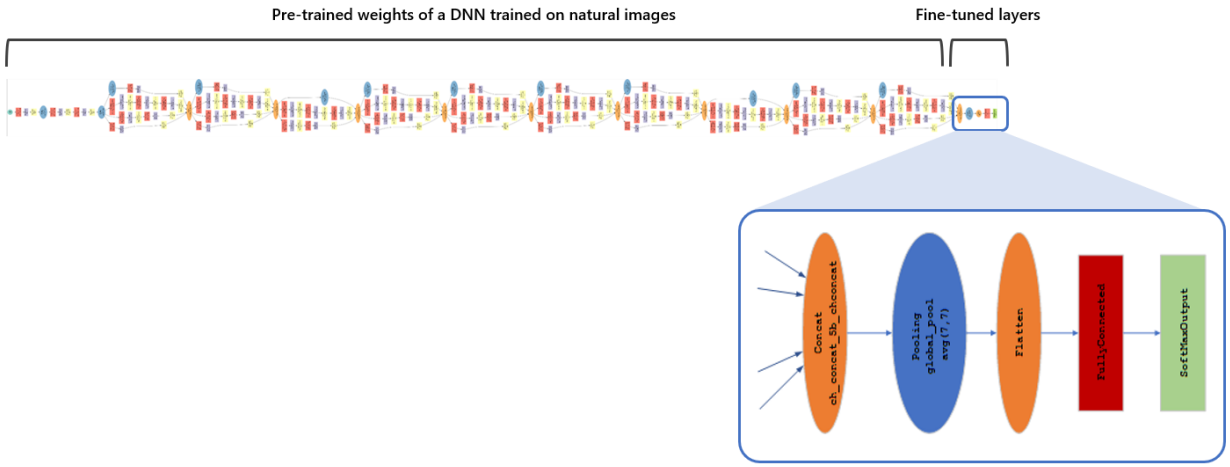


Figure 2: We retrained the last fully connected layer and softmax.

Evaluation

The last step of STAMINA is evaluation. We considered accuracy, false positive rate, precision, recall, F1 score, and area under the receiver operating curve (ROC) as evaluation. In particular, per feedback from malware analysis practitioners, we also reported recall at 0.1% – 10% false positive rate via ROC.

All metrics are evaluated on the test set. The accuracy is defined as the number of correctly classified samples over the total number of test samples. False positive rate is the number of benign software being misclassified as malware divided by the total number of benign software. Precision is the ratio of true positive divided by the sum of true positive and false positive. Recall is the ratio of true positive over the sum of true positive and false negative. F1 score is the harmonic mean of precision and sensitivity (one minus sensitivity).

Data Description

The analysis was done on a Microsoft dataset of 2.2 million hashes of malware binaries and 10 columns of data information. We split the training set, validation set and testing set 60:20:20, segmented along first time seen for benign and malicious. For the malware set, we initially prepared 1,241,218 training hashes, 413,739 validation hashes and 413,739 testing hashes, where the training hashes are seen at an earlier time than validation hashes and test hashes are seen last. Similarly, we also prepared 119,362 training hashes, 39,787 validation hashes and 39,787 test hashes for benign data, also selected based on the time axis of file last seen by Microsoft. The malicious files span through the past six months while the benign files span the past seven days.

We were able to successfully acquire 197,604 benign samples and 584,606 malicious samples. After removing files with size zero due to network limitation, we collected 782,224 binary applications from both benign and malicious classes. We split the training and testing samples using the time axes by benign and malicious files respectively. Further due to network limitation, we missed downloading many malware files in the mid-20% of the time axes. Hence, we combined the training and validation set together and still tested on the latest 20% of files seen. We obtained 157,837 benign training and validation samples and 39,781 benign test samples, 495,077 malicious training and validation samples and 89,529 malicious test samples. We noted that the malware-to-benign ratio from the training and validation set is approximately 3.143:1 the malware to benign ratio from the test set is approximately 2.251:1. Hence, the random guessing baseline is 74.74%, higher than 50%.

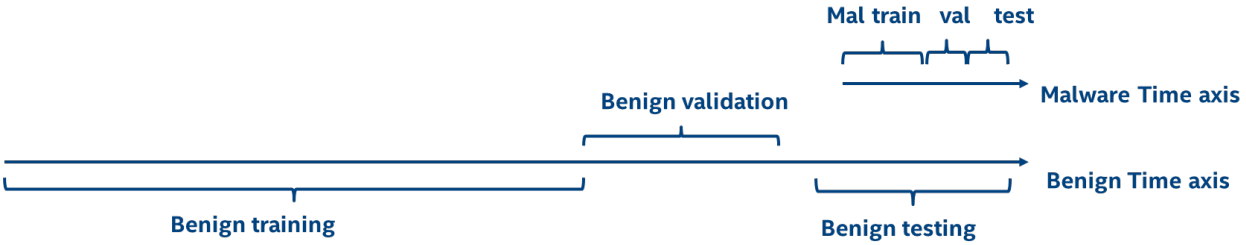


Figure 3: Benign training / validation / testing process

Because we received real-world data for our analysis, the file sizes are not distributed uniformly. We wanted to understand how much correlation between file sizes and labels exist in this dataset. This correlation analysis is important because if the file sizes are strongly correlated with labels, any model trained using the file size information is vulnerable. An attacker could manipulate the file sizes to cause the system to misclassify a malicious file into a benign application.

On the file sizes, we trained a light gradient boosting machine (lightGBM) [3] which is a gradient boosting framework that uses tree-based learning algorithms. Essentially, gradient boosting is an ensemble of weak learners, in this case, trees. To train gradient boosting, we first fit a tree to the training set, calculate the residuals, then fit a second tree to the residuals to obtain predicted residuals. We repeated this process until the error reached a pre-specified threshold. LightGBM was trained for epochs 5,000 but had an early stop at the 147th epoch.

The result of using file size as a 1-D feature yielded the following classification result. We used such a result to understand whether file size can be effective for classification. Compared with our guessing baseline at 75% accuracy, with 79.48%, we do not consider file size being very influential for malware classification. Hence for our initial approach, in order to deal with files with many different levels of file sizes, we proposed a file size gating scheme for our STAMINA model.

Dataset	Acc	FPR	Prec	Recall	F1
Test	79.48	3.07	85.33	40.30	54.65

Approach I: Resizing and segmentation after file gating

Our initial approach was the following: To combat the highly skewed file size distribution, we proposed using a file size gate to deal with applications of different size levels. To determine the file size to be used as a resizing threshold, we calculated the pixel file sizes for all the samples in the training and validation dataset. We noted that it is important to only use the training and validation samples to calculate the file size distribution, because we do

not want to peek into the test set. We selected median pixel file size 3.9 MB in the training and validation set to be the cut-off gate. We will describe the results from this approach and discuss the pros and cons for this approach toward the end of this paper.

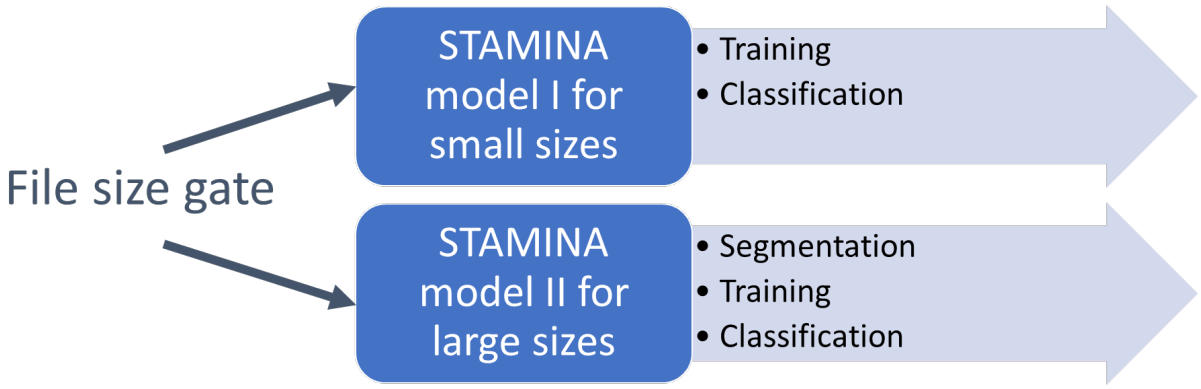


Figure 4: File size gating approach

Resizing

For STAMINA model I, we used Inception-v1 for the fine-tuning step. The model starts to converge at the 10th epoch. Due to bias-variance tradeoff, we selected the model trained from the 10th epoch. In our Inception framework, we downloaded the MxNet [4] model trained at epoch 126. We trained using transfer learning with Inception-v1 model on set of training + validation containing 81,417 benign and 327,592 malicious samples. Each epoch ttook 2.5 hours to train with 40.8k batches of computation at batch size 128. The test contained 19,897 benign and 55,931 malicious samples. Below is STAMINA performance on the test set.

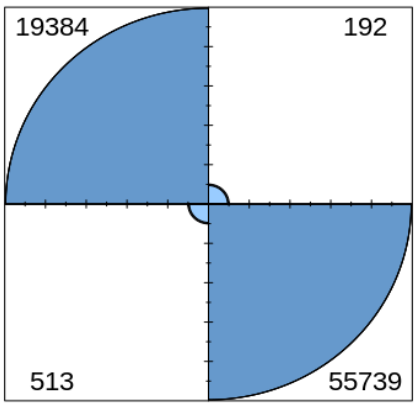
Looking at the training progress for STAMINA model I, the validation accuracy starts to decrease after the 10th epoch. To avoid overfitting, we used the model checkpointed at epoch 10 as our final model and apply the model on the test set.

Epoch no.	Training acc	Val acc
1	98.00	98.36
2	98.57	98.92
10	99.07	99.14
11	99.08	99.12
12	99.06	99.07
13	99.11	98.71

The classification result on the test set is as follows: The accuracy is 99.07% with false positive rate at 2.58%. The precision is at 99.09% and recall at 99.66%. F1 score is 0.9937.

Accuracy	FPR	Precision	Recall	F1	AUC	Balanced accuracy*
99.07	2.58	99.09	99.66	99.37	99.91	98.54

We also plotted the confusion matrix below. The diagonals are the true positives and true negatives.



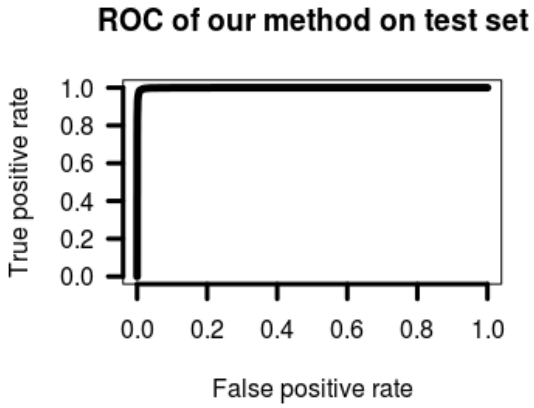


Figure 5: Confusion matrix for test set

At FPR 10%, we reported the recall or true positive rate at 99.94%. At FPR 0.1%, the recall is 87.05%. We also reported the recall (or true positive rate) across all values of false positive rate via ROC.

Segmentation

For STAMINA model II, we used segmentation and resizing to deal with large files. Our initial approach used file size as an extra gate, which is the model trained on large sized files. After examining the file size distribution for benign and malicious files respectively, we decided to segment the benign files only – for each benign file, we segmented and converted it into multiple images and for each malicious file, we resized and converted it into one single image. By doing this, we also addressed the data imbalance issue between benign and malicious. The number of segmented benign files for training is 31,965. The number of resized malware images for training is 58,630. We trained STAMINA model II on this set and tested on a set-aside test set.

Using Inception-v1, our model started to converge on the 10th epoch. After the 25th epoch, we noticed the accuracy on the validation set started to drop. Hence due to bias-variance tradeoff, we selected the model checkpointed at epoch 25. We reported the classification result on the test set. The accuracy is 95.97% with sensitivity at 97.90%, false positive rate at 4.49%. F1 score is 90.35. Precision is 83.89%. Recall is 97.90%.

The classification performance for STAMINA model II with segmentation processing is summarized as follows.

Accuracy	False positive rate	Precision	Recall	F1
95.97	4.49	83.88	97.90	90.35

Individual and aggregated classification result

Since we performed segmentation on benign images, we finally aggregated the individual predictions of the benign images and produced the overall scoring by taking the average of the individual scores.

We reported the result in a ROC curve as shown in Figure 6 below. The area under the curve is 0.8975. Because we segmented the benign files, we finally aggregated the benign predictions by taking the average. The area under the curve is slightly lower at 0.8750.

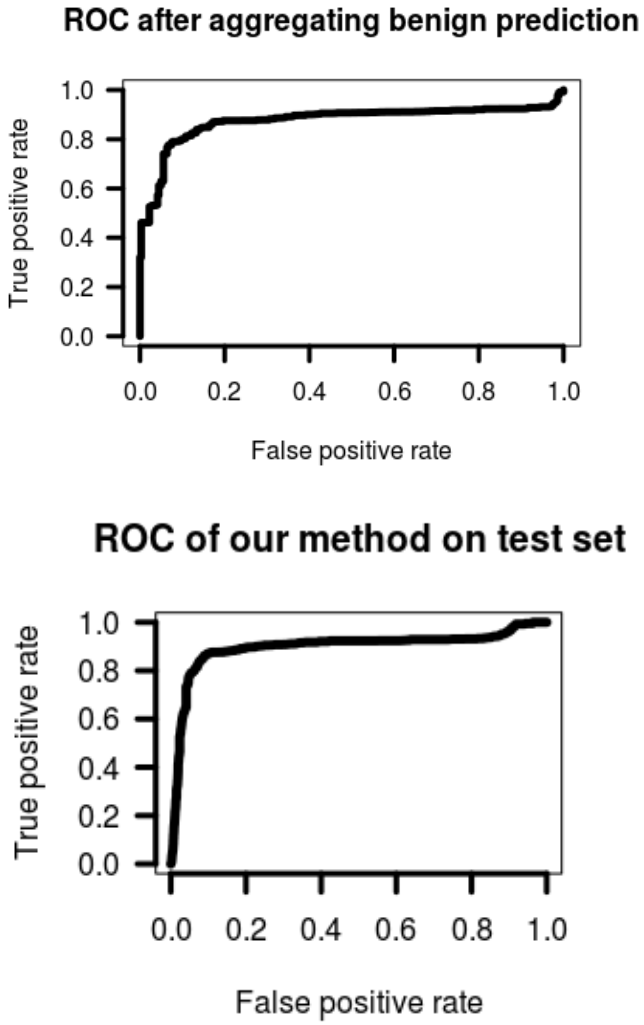


Figure 6: ROC results on test set and after aggregation of benign prediction

Limitation of segmentation step

We commented on the pros and cons of the segmentation approach stated above. STAMINA model II also showed excellent performance on the files with large file size. The reason we proposed the processing step in STAMINA

model II is because benign files are often in much larger size than malicious files. If resizing the giant images directly, information will be lost for benign images.

However, we caution that the preprocessing step in STAMINA model II should be used along with a prior model to infer the probability of a benign or malicious file. To elaborate, STAMINA model II is trained on large file sizes. We noticed that the most benign files have much larger file sizes than malicious files, so we proposed image segmentation. Most benign files result in more than 1,000 segmented images and malware images mostly result in less than 100 segmented images. Hence, we only applied the segmentation step on the benign images, not malware images. Although the results are very good, we must caution that processing the benign and malicious files differently means one needs to have some knowledge about the labels of files a priori. Such information may or may not be obtained during test time. Hence, we recommend that during deployment for STAMINA model II, when there is some knowledge of the labels either from security domain expertise or from simple machine learning classification, then STAMINA model II can achieve the best result.

Approach II: Resizing after file gating

We also explored the preprocessing technique of only resizing. However, we still stuck with the file size gating for the following reasons: When very large files are mixed with very small files for training, the textural information is much less preserved. The texture in the large files is more downsampled and the texture in the small files are enlarged, where both cases can cause performance degradation. We pointed out some issues with directly resizing the large application files. Many large files tend to result in correct JPEG data when reading with MxNet's imread function. Because the benign files are in several GBs in size, saving them into JPEG format results in corrupt JPEG with extraneous bytes before marker. This is an internal bug in libjpeg that ships with OpenCV.

Related work

There have been various related efforts to apply deep learning on disassembled byte code as well as gray scale images. The dissasembly based approaches [2] require preprocessing to dissasemble the binaries and specifically anotating or padding instructions – in image-based approaches we avoid the preprocessing steps. In contrast to alternate approaches using gray scale images corresponding to programs [5], we apply transfer learning, so we bypass the heuristic search for parameters (and architectures) – saving time and achieving almost similar accuracy. Other experiments have also analyzed using transfer learning for image-based malware classification [6] – our study is one of the first to evaluate this approach at a large scale for portable executable (PE) binaries.

Conclusion and insights

Classical malware detection approaches involve extracting binary signatures or fingerprints of the malware. However, the exponential growth of signatures makes signature-matching inefficient. Other approaches include static and dynamic analysis, both of which have advantages and disadvantages. Static analysis disassembles the code, but its performance can suffer from code obfuscation. Dynamic analysis, while able to unpack the code, can be time-consuming.

Our study indicates the pros and cons between sample-based and meta data-based methods. The major advantages are that we can go in-depth into the samples and extract textural information, so all the characteristics of the malware files are captured during training. However, for bigger size applications, STAMINA becomes less effective due to software not being able to convert billions of pixels into JPEG images and then resizing. In cases like this, meta-data-based methods show advantages over sample-based models.

As malware variants continue to grow, traditional signature matching techniques cannot keep up. We looked to applying deep-learning techniques to avoid costly feature engineering and used machine learning techniques to learn and build classification systems that can effectively identify malware program binaries. We explored a novel image-based technique on x86 program binaries, which resulted in 99.07% accuracy with 2.58% false positive rate.

For future work, we would like to evaluate hybrid models of using intermediate representations of the binaries and information extracted from binaries with deep learning approaches – these datasets are expected to be bigger but may provide higher accuracy. We also will continue to explore platform acceleration optimizations for our deep learning models so we can deploy such detection techniques with minimal power and performance impact to the end-user.

References

[1] L. Chen, "Deep transfer learning for static malware classification," *arXiv preprint arXiv:1812.07606*, 2018.

[2] A. Davis, M. Wolff, "Deep learning on disassembly data," *Blackhat*, 2015.

[3] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," *Advances in Neural Information Processing Systems*, 2017.

[4] T. Chen et al., "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.

[5] D. Gibert, "Convolutional neural networks for malware classification," *Department of Computer Science – A thesis presented for the degree of Master in Artificial Intelligence, 2016*.

[6] N. Bhodia, P. Prajapati, F. Di Troia, M. Stamp, "Transfer learning for image-based malware classification," *arXiv preprint arXiv:1903.11551,* 2019.

Notices & Disclaimers