Malware Classification Through Computer Vision

Prasann Viswanathan Iyer
Department of Electrical Engineering, IIT Bombay
Mumbai, India
prasann@iitb.ac.in

***Abstract*—We propose to apply a transformation to static malware byte files to convert them into RGB images. Following this, we use transfer learning and Decision Tree Classification on the resulting images. The objective of this Machine Learning Classification scheme is to see whether the transformation of Malware Byte Code to RGB images improves various evaluation metrics of classification, such as Accuracy, False Positive Rate, True Positive Rate and *F1* score. Finally, if time permits, we plan to explore the possibility of having a convex combination of the models.**

***Index Terms*—Classification, computer vision, decision trees, machine learning, static malware, transfer learning**

I. Introduction

The term static malware means to classify the files without executing the application or monitoring its runtime behaviour. The textbook method of static malware classification is signature matching, which checks if strings of code match malicious patterns in our database. However, this approach is discarded as it is not robust to obfuscations or changes in code and is neither scalable when the number of signatures grows exponentially.

Consequentially, machine learning has proven to be quite effective for large-scale malware classification. However, feature selection for this classification task is cumbersome. It requires a disassembly step to make the code human-readable and is followed by counting the number of loops, binning the type of function calls, etc. Such processing results in large feature spaces, which further need to be reduced through techniques such as PCA and K-PCA. Therefore we desire a method without much manual effort in feature construction.

To solve these issues, our implementation will consider malware classification as a computer vision problem. The proposed method will consume the entire malware application binary irrespective of code obfuscation and is completely independent of signatures.

Finally, we test out the transformed data through a transfer learning neural network and compare its performance with a Decision Tree Classifier to see if converting byte code to RGB images improves on classification metrics.

II. Methodology and Implementation

The inspiration [**?**] for our method came from visual analysis of the malware binaries rendered in grayscale, where it is seen that malware from the same family shares structural similarities while malware from other families displays distinctive structural or textural characteristics. The malware classification challenge may be approached as a vision classification task with the use of such visual examination.

A. Preprocessing

Given a static binary, we map it directly to an array of integers between 0 and 255. Hence each binary is converted into a one-dimensional array $v \in [0, 255]$. Then the array v is normalized to $[0, 1]$ by dividing by 255. The normalized array v is then reshaped into a two-dimensional array v by the rules mentioned in the below table:

File Size (in Kb)	Width
$(0, 10]$	32 or remove samples
$(10, 30]$	64
$(30, 60]$	128
$(60, 100]$	256
$(100, 200]$	384
$(200, 500]$	512
$(500, 1000]$	768
$(1000, 2000]$	1024
> 2000	2048

Thus we reshape one-dimensional array v into a two-dimensional array v' using the above relationship between file size and width.

The grey-scale photos are resized into a $m \times m$ matrix. To suit the input size of the pre-trained models using ImageNet, the option of m is chosen to be 224 or 229. We duplicate the channel three times to create RGB pictures after scaling the two-dimensional arrays into squared grey-scale images.

B. Training our Model

In the second stage, transfer learning is applied to the scaled and rearranged photos, with some of the layers frozen and the final few layers being retrained on malware images. Our architecture is adaptable to support a variety of deep learning neural networks, including Inception, VGG, ResNet, and DenseNet, because of the transfer learning strategy. We significantly reduce the time spent looking for neural network designs, parameters, and optimizers by using transfer learning. We also contrasted training from scratch with training using transfer learning. We have a lot fewer malware photos than there are images in ImageNet or other models, so training massive deep neural networks from scratch might not converge. Therefore, for comparison, we utilise scaled-down deep neural network models instead.

Then we apply a Decision Tree Classifier to the image

converted data and compare its performance with a similar classifier on a regular (non-image-converted) dataset. This will serve to indicate if the conversion of malware bytecode to RGB channel images assists across all classification models or not.

C. Evaluation

As assessment criteria for our suggested technique, we employ classification accuracy, false positive rate, true positive rate, and F1 score (in binary classification). If our suggested approach exceeds all other chosen machine learning algorithms in terms of all these metrics on real-world datasets, we'll demonstrate this in a subsequent section.

D. Interpretation of Classification Results

Once the results have been generated, we will interpret them to check whether they are consistent to avoid the case of the models performing poorly in real-life scenarios.
We effectively use the local interpretable model-agnostic explanation method to deliver interpretation. Super pixels, which are areas or patches of pixels next to one another, are the initial method we use to depict the virus pictures. The virus graphics incorporate the super-pixel representation as a visually comprehensible feature. Then, each superpixel has a binary vector imposed on it, where 0 denotes the lack of the superpixel and 1 denotes its existence. To train on the binary vectors and get the learned coefficients, we employ sparse linear classifiers. The areas of pixels associated with each super pixel's positive coefficients are those that contribute to the model's classification decision, while those associated with each super pixel's negative coefficients are those that do not.

E. Flowchart

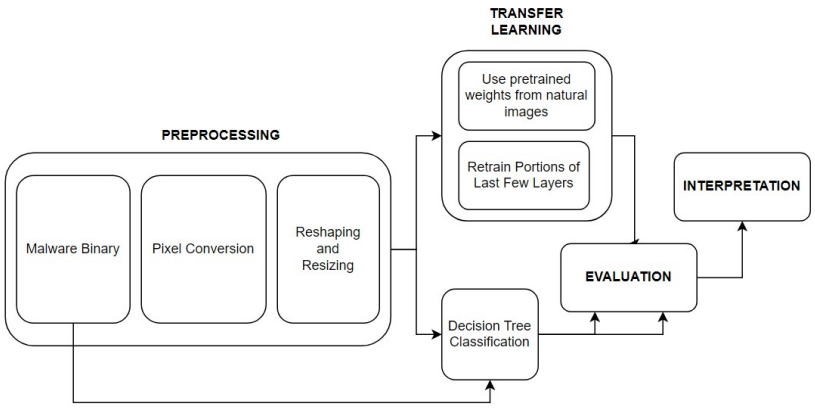

Fig. 1. The implementation Flow Diagram

REFERENCES

B1 Chen, L. (2018). Deep Transfer Learning for Static Malware Classification. arXiv. https://doi.org/10.48550/arXiv.1812.07606