USING TRANSFER LEARNING FOR MALWARE CLASSIFICATION

PRIMA Bouchaib, BOUHORMA Mohamed

1 Computer Science, Systems and Telecommunication Laboratory, Faculty of Sciences and Techniques, Abdelmalek Essaâdi University, Tangier 90000, Morocco - bouchaib.prima@etu.uae.ac.ma bouhorma@gmail.com

KEY WORDS: Cybersecurity, Malware, Machine Learning, Deep Learning, Transfer Learning, Convolutional Neural Network.

ABSTRACT:

In this paper, we propose a malware classification framework using transfer learning based on existing Deep Learning models that have been pre-trained on massive image datasets. In recent years there has been a significant increase in the number and variety of malwares, which amplifies the need to improve automatic detection and classification of the malwares. Nowadays, neural network methodology has reached a level that may exceed the limits of previous machine learning methods, such as Hidden Markov Models and Support Vector Machines (SVM). As a result, convolutional neural networks (CNNs) have shown superior performance compared to traditional learning techniques, specifically in tasks such as image classification. Motivated by this success, we propose a CNN-based architecture for malware classification. The malicious binary files are represented as grayscale images and a deep neural network is trained by freezing the pre-trained VGG16 layers on the ImageNet dataset and adapting the last fully connected layer to the malware family classification. Our evaluation results show that our approach is able to achieve an average of 98% accuracy for the MALIMG dataset.

1. INTRODUCTION

Malware and associated computer security threats have become more and more developed, and also malware developers have become more creative and use increasingly complex escape techniques (obfuscation, packers, cryptor, protector, Advanced Evasion Techniques (AET) and Network evasion) (Sibi Chakkaravarthy, Sangeetha, and Vaidehi 2019). The latest McAfee report indicates that the new PowerShell malwares increased by 689% in the 1st quarter of 2020 compared to the previous quarter, and the number of new macro malwares has increased by 412% in the first quarter of 2020. (McAfee Labs Threats Report, juillet 2020).

This increase in the number of malware and the complexity of the escape techniques used, has led researchers to use detection and classification techniques based on machine learning, motivated by the success of this technique recently in the fields of computer vision and natural language processing.

The use of machine learning has shown favorable results compared to traditional malware analysis techniques that often require a lot of time and resources in feature engineering. Also, recently the Convolutional Neural Network (CNN) has been used for malware classification and this architecture has been able to achieve more satisfactory results in terms of accuracy. The principle of these techniques is presented in Section 2.

Based on the work of (Nataraj et al. 2011) who presented the malware presentation in grayscale images, we realize a malware classification system based on deep learning and we use transfer learning technique to train our CNN model based on VGG16 (Simonyan and Zisserman 2015) pre-trained model on larger dataset. Also, we make a comparative study of the different used techniques for malware classification.

We adapt VGG16 pre-trained model to make a malware classification and we make a comparative study of the obtained results with the literature, we prove that the transfer learning realizes a superior performance for malware classification then training our deep learning model from scratch.

2. RELATED WORK

In this section, we present the progress of the research as well as the techniques used to detect and classify malwares.
2.1 Static and Dynamic analysis

The objective of the malware analysis is to study the behavior and structure of malware, and there are two types: static analysis and dynamic analysis.

- **Static Analysis:**
  The static analysis is performed without executing the malware, for Windows portable executable (PE) files we can proceed in two ways, either based on the binary file or on the disassembled malware program. This method of reverse engineering can be done on PE files executable by several tools the most used are: IDA Pro and Radare.

- **Dynamic Analysis:**
  The dynamic analysis is performed by executing the malware on a testing environment (Sandbox) where we can analyse its behavior and have all traces made by this malware. This analysis is usually used if we were not able to collect much information about the malware by static analysis due to the complex obfuscation used by the malware developer or can be used as a complementary analysis to extract more features. This scan should be performed on a completely isolated environment to avoid impacting our system, there are several environments to use, the most well-known is Cuckoo Sandbox.

(Talukder 2020; Sibi Chakkaravarthy, Sangeetha, and Vaidehi 2019) they summarize the tools used for each type of analysis and the extracted information.

2.2 Method based on Machine Learning

The classification and detection of malware using Machine Learning (ML) is based on the following steps:

1. Features extraction.
2. Features selection.
3. Classification algorithm.

The work of (Ahmadi et al. 2016) is focused on extracting and selecting a new set of features from binary files and disassembled files to effectively represent malware samples. Once the features are extracted and selected, they will be used to train the malware classification model or malware detection in case of binary classification (malicious or benign file) using a dataset of benign file features. (Ranveer and Hiray 2015)

There are several works that have performed the malware classification based on machine learning (ML) method such as: (Nataraj et al. 2011) after presenting binary malware files as grayscale images, they performed a classification of the images based on GIST as features and they used machine learning algorithm k-nearest neighbors with Euclidean distance for malware classification. (Kong and Yan 2013) based on the features (function call graphs) extracted from the malware they calculate the similarity of the two malwares using SVM, KNN. (Abou-Assaleh et al. 2004) in this work, they used text classification techniques based on n-grams (is a subsequence of n elements built from a sequence of text), extracted from the signatures of malware, and they performed the KNN algorithm to perform the classification.

2.3 Method based on Deep Learning

The malware visualization has successfully introduces deep convolutional neural networks into malware classification problems. (Xiao et al. 2020) After they displayed the binary malware as entropy graphs they used deep learning to do feature extraction automatically and then used SVM to classify the malware based on the extracted features. (Gibert et al. 2019) Based on the presentation of malware as an image, the following work presents a convolutional neural network (CNN) composed of three convolution layers followed by a fully-connected layer used for the classification of malware. They made a comparative study to prove that CNN has better results than KNN.

To resume, the methods based on traditional machine learning use a high computational cost because they often have to define and extract in advance a group of features and are not adapted for processing massive data. On the other hand the Deep learning automates the feature extracting and selecting, avoids the high computational cost. However, the literature has proved that the Deep Learning methods are more performant than the Machine Learning methods in term of accuracy.

3. METHODOLOGY

In this section we discuss the dataset and implementation details of our proposed models.

3.1 Visualizing Malware as an Image

Our work is based on the visualization of malware as an image, this approach initiated by (Nataraj et al. 2011) allowing to read a given malware binary as a vector of 8 bit unsigned integers and then organized into a 2D array. Finally this can be visualized as a gray scale image in the range [0,255] (0: black, 255: white).

![Visualizing Malware as an Image](image)

Figure 2. Visualizing malware as a grayscale image process

This presentation allows us to visualize malware belonging to the same family with a very similar image.

However, this malware visualization is based on the binary code, so if a malware developer is going to create a new malware by modifying the code of an old malware, with this approach the new malware will be visualised with a very similar image. Then we can use our classification model (CNN) presented later to easily classify it into the same family.

3.2 Dataset

The MalImg dataset was provided by (Nataraj et al. 2011) contains 9435 grayscale images of malwares packed with UPX, collected from 25 families:
| Family         | Family Name     | Number of samples |
|---------------|-----------------|-------------------|
| Worm          | Allaple.L       | 1591              |
| Worm          | Allaple.A       | 2949              |
| Worm          | Yuner.A         | 800               |
| PWS           | Lolyda.AA 1     | 213               |
| PWS           | Lolyda.AA 2     | 184               |
| PWS           | Lolyda.AA 3     | 123               |
| Trojan        | C2Lop.P         | 146               |
| Trojan        | C2Lop.gen!G     | 200               |
| Dialer        | Instantaccess   | 431               |
| Trojan Downloader | Swizzor.gen!I | 132               |
| Trojan Downloader | Swizzor.gen!E | 128               |
| Worm          | VB.AT           | 408               |
| Rogue         | Fakerean        | 381               |
| Trojan        | Alueron.gen!J   | 198               |
| Trojan        | Malex.gen!J     | 136               |
| PWS           | Lolyda.AT       | 159               |
| Dialer        | Adluer.AT       | 125               |
| Trojan Downloader | Wintrim.BX    | 97                |
| Dialer        | Dialplatform.B  | 177               |
| Trojan Downloader | Dontovo.A     | 162               |
| Trojan Downloader | Obfuscator_AD  | 142               |
| Backdoor      | Agent.FYI       | 116               |
| Worm:AutoIT   | Autorun.K       | 106               |
| Backdoor      | Rbot!gen       | 158               |
| Trojan        | Skintrim.N      | 80                |

Table 1. MalImg: Distribution of Samples

After analysing the number of samples of this dataset, we can notice that the MALIMG dataset is quite unbalanced: more than 30% of the images belong to class: Allaple.A and 17% to class: Allaple.L!

3.3 Transfer Learning for Malware Classification

The general structure of a CNN is the combination of two components: The feature extractor in the first stage and the classifier:

The transfer learning is to replace the Classifier component of the pre-trained model, VGG16 in our case, from VGG family (Visual Geometry Group at University of Oxford) with a customized classifier to resolve our classification problem.

In practice we replace the last layer of the VGG16 (Figure 6), which takes a probability for each of the 1000 classes in the ImageNet (Krizhevsky, Sutskever, and Hinton 2012) and replaces it with a Fully Connected layer that takes 25 probabilities corresponding to 25 families of malwares. This way, we use all the knowledge that VGG16 has trained on the ImageNet dataset and apply it to our malware classification problem.
The VGG16 network architecture (Simonyan and Zisserman 2015) is shown in figure 6.

The input layer is an RGB image of fixed size 224 × 224, then the image is passed through a stack of convolutional layers, where the size of the filters used is 3 x 3 with a stride 1, and it always uses the same padding and maxpool layer of 2x2 filter of stride 2.

Finally for classification, it has 2 fully connected layers followed by a softmax for output.

The 16 in VGG16 refers to it has 16 layers in total:
- 5 convolutional layers,
- 5 max pooling layers,
- 3 fully-connected layers
- output layer (softmax)

### 3.4 Our proposed models

As explained above, we propose a CNN model for malware classification based on the pre-trained model VGG16, using transfer learning (Figure 8). And we make a performance comparison with a second CNN model (Figure 7) trained from scratch.

The input of our network is a malicious program represented as a grayscale image, and the output is the predicted class of the malware sample.

For our first proposed architecture (Figure 7), we used three convolutional layers where the size of the filters used is 3x3 that scans the whole images and create a feature map to predict the class probabilities for each feature.

After each convolution layer we used a max pooling layer of 2x2 filters to scales down the amount of information generated for each feature and maintains only the most essential information.

At the end, the generated feature maps are flattened and combined to be used as input of the following fully connected layer composed of 256 neurons. Lastly, the output of the fully connected layer passes to a Softmax layer to classify the binary malware into its corresponding family.

To prevent overfitting during the training phase, we employed one dropout layer (Srivastava et al. 2014) to ignoring units of certain set of neurons which is chosen at random.

In this second model we customize the VGG16 architecture to our classification problem by adding a fully-connected layer containing 25 neurons corresponding to 25 malware families, instead of final fully connected layer (intended for 1000 classes).

The objective of this architecture is to employ the initial weights of pre-trained CNN of natural images (ImageNet dataset) to classify the binary malwares.

### 3.5 K-fold cross validation

To evaluate the generalization performance of our models we used K-fold cross validation. The dataset is divided into K equal size folds. Of the K subsamples, a single subsample is retained as the validation data for testing the model and the remaining subsamples are used as training data. This procedure is repeated as many times as there are folds, with each of the K folds used exactly once as the validation data.

### 3.6 The performance metric

To train our two models we will use an unbalanced dataset (Malimg). Furthermore, the accuracy is not the best metric to use when evaluating unbalanced datasets as it can be very misleading.

However, for our comparative study we will use the following metrics: precision, recall and F1 score and confusion matrix:

**Precision (P):** is the number of true positives predictions (TP) divided by all true positive predictions (TP) plus the number of false positives (FP).
Recall (R): is the number of true positives (Tp) divided by the number of true positives plus the number of false negatives (Fp)

\[
R = \frac{Tp}{Tp + Fn}
\]

**F1 Score**: the weighted average of precision and recall.

\[
F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

**Macro averaged F1 Score**: is the average of the individual F1 scores obtained for each class.

\[
\text{macro}_F1 = \frac{1}{q} \sum_{i=1}^{q} F1_i
\]

Where \(q\) = number of classes in the dataset \(F1_i\) = F1 score of classe \(i\)

**Confusion Matrix**: is a table showing the correct predictions and the incorrect types of predictions.

### 4. RESULTS AND DISCUSSION

We performed two different experiments and we made a comparative study of the obtained results. We present in this section the performed experiments and we discuss the results.

After several experiments we have optimized the hyper-parameters for the two proposed models (batch-size, epochs, and number of folds) to achieve the best performance.

#### 4.1 Experiment 1

To train our first CNN model (Figure 7) using Malimg dataset we used the Cross-Validation algorithm (defined above) with 10 Folds and 40 epochs, and we downsampled the images to a fixed size. The size of the new images was set to 200*200 pixels.

According to the obtained results, this first model is very powerful for all malwares given in input except the following families:

- The Autorun.K family is classified incorrectly as Yunner.A, as you can see in the (figure 10), the precision of the Autorun.K family is 0. That is because these two families are very similar and are indistinguishable by the human eye (Figure 11).
Also, the model can't distinguish correctly the samples belonging to the same family: Swizzer.genE and Switzer.gen! I. (precision 0.61 and 0.66).

4.2 Experiment 2

To train our second model (Figure 8) (based on the transfer learning) using Malimg dataset, we used 5 Folds cross validation and 10 epochs, and we downscaled the images to a fixed size. The size of the new images was set to 200*200 pixels. This model has proven the best performance by using only 90% of dataset.

4.3 Comparison of models performance:

To compare the performance of our two models, we will use the following metrics already explained above. However, we obtain an overall classification accuracy of 97% for the CNN model with the simple architecture and trained from scratch, which represents a significant decline from the VGG16 model accuracy of 98%.

The others performance metrics are summarized in the following table.

| Method | Accuracy | Precision | Recall | F1 score |
|--------|----------|-----------|--------|----------|
| CNN    | 0.97     | 0.91      | 0.91   | 0.91     |
| Using  | 0.98     | 0.95      | 0.95   | 0.95     |
| VGG16   |          |           |        |          |

The following table present a comparison of accuracy performance of our two models with the literature.
Our experiments, has proven that the transfer learning technique can be used for the classification of malware.

This study can be considered as an introduction to many new experiments in the field of using transfer learning for malware classification.

**REFERENCES**

Abou-Assaleh, Tony, Nick Cercone, Vlado Kes’elj, and Ray Sweidan. 2004. Detection of New Malicious Code Using N-Grams Signatures. Second Annual Conference on Privacy, Security and Trust.

Ahmadi, Mansour, Dmitry Ulyanov, Stanislav Semenov, Mikhail Trofimov, and Giorgio Giacinto. 2016. Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification. In Proceedings of the Sixth ACM on Conference on Data and Application Security and Privacy - CODASPY ’16 Pp. 183–194. New Orleans, Louisiana, USA: ACM Press. http://dl.acm.org/citation.cfm?doid=2857705.2857713.

Gibert, Daniel, Carles Mateu, Jordi Planes, and Ramon Vicens 2019. Using Convolutional Neural Networks for Classification of Malware Represented as Images. Journal of Computer Virology and Hacking Techniques 15(1): 15–28.

Kong, Deguang, and Guanhua Yan, 2013. Discriminant Malware Distance Learning on Structural Information for Automated Malware Classification. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: 9.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton 2012 ImageNet Classification with Deep Convolutional Neural Networks. Communications of the ACM 60(6): 84–90.