Malicious Software Classification using Transfer Learning of ResNet-50 Deep Neural Network

Edmar Rezende*†, Guilherme Ruppert†, Tiago Carvalho‡, Fabio Ramos§ and Paulo de Geus*
*University of Campinas, Campinas, SP, Brazil - Email: {edmar,paulo}@lasca.ic.unicamp.br
†Center for Information Technology Renato Archer, Campinas, SP, Brazil - Email: guilherme.ruppert@cti.gov.br
‡Federal Institute of São Paulo, Campinas, SP, Brazil - Email: tiagojc@gmail.com
§University of Sydney, Sydney, NSW, Australia - Email: fabio.ramos@sydney.edu.au

***Abstract*—Malicious software (malware) has been extensively used for illegal activity and new malware variants are discovered at an alarmingly high rate. The ability to group malware variants into families with similar characteristics makes possible to create mitigation strategies that work for a whole class of programs. In this paper, we present a malware family classification approach using a deep neural network based on the ResNet-50 architecture. Malware samples are represented as byteplot grayscale images and a deep neural network is trained freezing the convolutional layers of ResNet-50 pre-trained on the ImageNet dataset and adapting the last layer to malware family classification. The experimental results on a dataset comprising 9,339 samples from 25 different families showed that our approach can effectively be used to classify malware families with an accuracy of 98.62%.**

I. Introduction

Malicious software (malware) is one of major security threats facing the Internet today. The ability to automatically detect malware and group them into families with similar characteristics is beneficial because it becomes significantly easier to create new mitigation strategies that work for a whole class of programs.

Machine learning methods have been employed to automate the malware detection and classification task. In these methods, classifiers are applied to learn patterns in a set of features extracted from static and dynamic malware analysis. However, designing a feature extractor that would transform raw data into a suitable feature vector requires careful engineering and considerable domain expertise.

Whereas many researchers have relied on hand-crafted malware representations, there are machine learning algorithms that can be used to come up with good malware representations. Deep learning [1] are representation learning methods with multiple levels of representation. Deep learning's key aspect is that these feature layers are not designed by human engineers, rather they are learned from data using a general purpose learning procedure.

Deep Convolutional Neural Networks (DCNN) have become the standard approach for classification tasks within the last years. This transition from traditional approaches based on hand-crafted feature extractors combined with shallow classifiers to DCNNs is due to the overwhelming performance of DCNNs on classification challenges such as the ILSVRC—ImageNet Large Scale Visual Recognition Challenge [2].

Over the years, there has been a trend where the deeper the model is, the better performance the model can achieve on the ImageNet challenge. In 2012, the AlexNet architecture with 8 layers resulted in a top-5 classification error of 16.4% on the ImageNet challenge. In 2014, the VGG16 model with 16 layers and VGG19 model with 19 layers resulted in a top-5 classification error of 7.3%, and the GoogLeNet model with 22 layers resulted in a top-5 classification error of 6.7%. And finally in 2015, the ResNet model with 152 layers resulted in a top-5 classification error of 3.57%.

In this paper we present an approach for malware family classification using a DCNN model based on the Residual Network architecture with 50 layers (ResNet-50) [3]. We represent malware samples as byteplot images and using a transfer learning approach we transfer the convolutional layers' parameters of ResNet-50 pre-trained on ImageNet dataset to our model, adapting the last softmax layer to our problem. The experimental results on a dataset comprising 9,339 samples from 25 different malware families showed that our approach can effectively be used to classify malware families with an accuracy of 98.62%, outperforming the reference work on this dataset.

The remaining of the paper is organized as follows: Section II presents malware classification related work. Section III describes the method proposed in this work in details. Section IV presents our experimental results. The conclusions follow in Section V.

II. Related Work

The use of machine learning for automatically classifying malware families has been extensively studied in the literature. Some approaches based on the use of visualization techniques have been proposed to support malware analysis with respect to feature extraction and pattern recognition of malware samples.

Conti et al. [4] proposed a method called byteplot for visualizing binary data objects as grayscale images, arguing that visual analysis of binary data helps distinguish structurally different regions of data. Nataraj et al. [5] proposed a method for classifying malware represented as byteplot grayscale images using image processing techniques. Using Gabor filters to extract texture features and then using a k-nearest neighbors (kNN) classifier with $k = 3$, they obtained an accuracy

of 97.18% in a dataset consisting of 25 malware families. Kancherla and Mukkamala [6] presented a malware detection approach using intensity and texture features extracted from byteplot grayscale images. Using Support Vector Machines (SVMs), they obtained an accuracy of 95% on a dataset containing 36 thousand samples.

In the last few years, researchers have applied deep learning techniques to learn patterns in a set of features extracted from static and dynamic malware analysis in order to classify new samples.

Kolosnjaji et al. [7] used a neural network that combines convolutional and recurrent layers for malware classification using system call n-grams obtained from dynamic analysis. Their evaluation results achieved an average of 89.4% on accuracy in a dataset containing 4,753 malware samples from 10 different families. Saxe and Berlin [8] deployed a malware detector using static features: byte/entropy/string histograms, PE imports and PE metadata. Training a feedforward neural network consisting of four layers, they achieved a 95% detection rate in a dataset containing 431,926 binaries.

Unlike previous work, our approach does not require any feature engineering, using raw pixel values of byteplot images as our underlying malware representation. Additionally, we explore knowledge transfer from a deep neural network trained for object detection task on a different dataset to discover good malware representations, improving the classification results.

III. Proposed Method

The proposed method consists of the following basic steps:

1) Given a dataset of labeled malware executables, convert each sample to its respective byteplot grayscale image;
2) Convert each byteplot to an RGB image, rescaling it to 224×224 dimension and subtracting the mean RGB value computed on the ImageNet dataset from each pixel, as proposed by Krizhevsky et al. [9], to feed it to the deep neural network;
3) Build a deep convolutional neural network (DCNN) based on the ResNet architecture with 50 layers (ResNet-50), replacing the last 1000 fully-connected softmax layer by a 25 fully-connected softmax layer and transferring the parameters of ResNet-50 convolutional layers to the convolutional layers of the DCNN model;
4) Freeze the transferred convolutional layer's parameters and train the DCNN model to classify each sample into its malware family.

An overview of the entire method's pipeline is given in Figure 1.

A. Byteplot Visualization

The byteplot visualization method was initially proposed by Conti et al. [4] to represent binary data objects as images where each byte corresponds to one pixel color rendered as a grayscale. As can be seen in Fig. 2(a), visual analysis of binary data in fact helps distinguish structurally different regions of data, facilitating a wide range of analytic tasks.

Nataraj et al. [5] proposed a method for classifying malware represented as byteplot images using image processing techniques. They observed significant visual similarities in image texture for malware belonging to the same family, as shown in Fig. 2(b), possibly explained by the common practice of reusing code to create new malware variants.

To transform malware samples into byteplot images, a given malware binary is read as a vector of 8-bit unsigned integers and then organized into a 2D array, where the width is defined by the file size, based on empirical observations made by Nataraj et al. [5]. The height is allowed to vary depending on the width and the file size.

B. Transfer Learning of ResNet-50

Residual Networks (ResNets) [3] are deep convolutional networks where the basic idea is to skip blocks of convolutional layers by using shortcut connections. The basic blocks named "bottleneck" blocks follow two simple design rules: (i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the number of filters is doubled. The down-sampling is performed directly by convolutional layers that have a stride of 2 and batch normalization is performed right after each convolution and before ReLU activation. When the input and output are of the same dimensions, the identity shortcut is used. When the dimensions increase, the projection shortcut is used to match dimensions through 1×1 convolutions. In both cases, when the shortcuts go across feature maps of two sizes, they are performed with a stride of 2. The network ends with a 1,000 fully-connected (fc) layer with softmax activation. The total number of weighted layers is 50, with 23,534,592 trainable parameters. The architecture of the original ResNet-50 is illustrated in Fig. 1.

In our method, we use ResNet-50 as the base model, pre-trained for object detection task on the ImageNet dataset [10]. Our hypothesis is that despite the disparity between natural images and malware byteplot images, ResNet-50 comprehensively trained on the ImageNet may still be transferred to make malware image recognition tasks more effective, since collecting and annotating large numbers of malware samples still poses significant challenges.

Using transfer learning techniques [11], we transferred the first 49 layers of ResNet-50, which are left frozen on the malware classification task. These layers can be seen as learned feature extraction layers. The activation maps generated by those learned features extraction layers are usually called bottleneck features. Using the bottleneck features of our malware byteplot images as input, we train a 25 fully-connected softmax, since we have 25 classes, and then replace the 1,000 fully-connected softmax by our trained 25 fully-connected softmax, as illustrated in Fig. 1.

IV. Experimental Results

In this section, we evaluate the performance of the proposed DCNN model using a public dataset and compare it with approaches proposed in the literature.

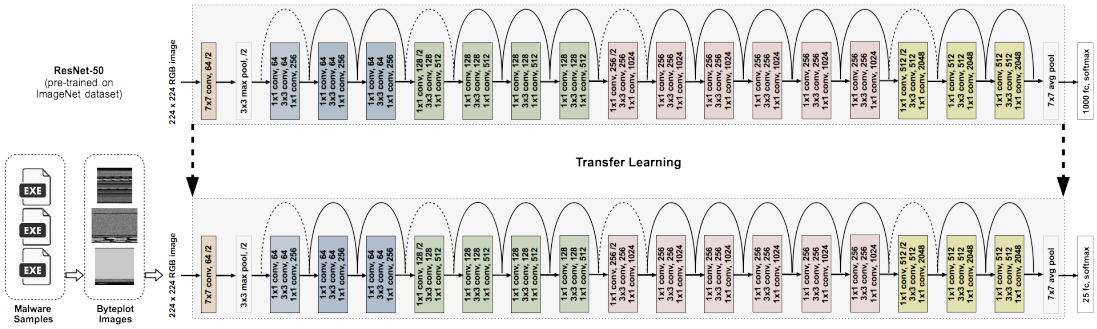
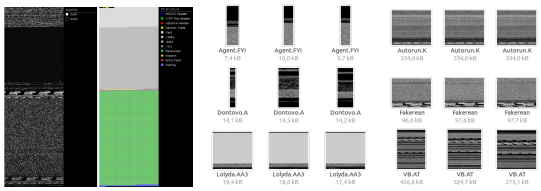


Fig. 1. Overview of proposed method. ResNet-50 layers pre-trained on ImageNet dataset are transferred to our DCNN model, replacing the last 1000 fully-connected (fc) softmax layer by a 25 fully-connected softmax layer and freezing the parameters of the convolutional layers during the training process.



(a) (b)

Fig. 2. (a) Byteplot and PE Structure comparison. (b) Byteplot visualization of malware samples from different families.

A. Dataset

The DCNN method has been evaluated with the Malimg dataset[1] proposed by Nataraj et al. [5]. The dataset comprises 9,339 byteplot images from 25 different malware families. As reported by the authors, all byteplot images were generated from malware executables submitted to the Anubis analysis system[2] and labels provided by Microsoft Security Essentials were used to obtain the ground truth for the dataset.

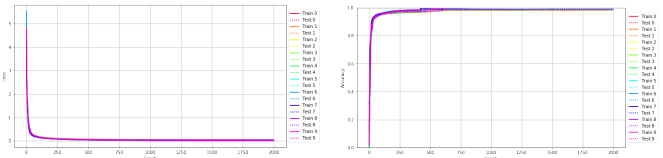
B. Validation Protocol

To evaluate the proposed model performance we used a stratified 10-fold cross-validation, randomly partitioning the samples into ten disjoint sets of equal size containing roughly the same proportions of the class labels in each fold, selecting one as a testing set and combining the remaining nine to form a training set. We conducted ten such runs using each partition as the testing set and reported the average classification accuracy, the accuracy and the execution time for each fold.

C. Malware Classification Results

To train our DCNN model on the malware classification task, we first use the ResNet-50 transferred convolutional layers to extract the bottleneck features of our byteplot images, which are then used as input to train a custom 25 fully-connected softmax layer using a categorical cross-entropy cost function and the Adam optimizer for 2,000 epochs. The weights have been initialized using the Glorot uniform approach and the bias terms were initialized to zero.

The trained softmax layer is stacked on top of the layers transferred from ResNet-50 to build our DCNN model, which is used to classify the test set. The train/test loss and accuracy of our DCNN model for each fold is presented in Figures 3(a) and 3(b), respectively. Solid lines represent the performance in the training set while dashed lines represent performance in the test set.



(a) (b)

Fig. 3. DCNN 10-fold cross-validation train/test (a) loss and (b) accuracy.

It is possible to observe in those figures that after 750 epochs both loss and accuracy tend to stabilize for all folds and the accuracy presents very small improvements for both train and test sets, converging to an average accuracy of 0.9862 ($\pm$0.0025).

D. Comparative Analysis

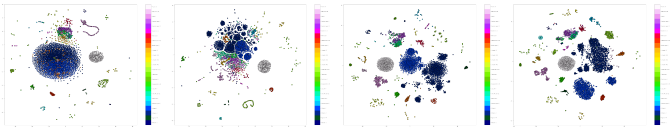
To perform a comparative analysis of our DCNN model with similar work proposed in the literature, we reproduced the approach proposed by Nataraj et al. [5] using GIST filters to extract texture features from the byteplot grayscale images combined with a kNN ($k = 1$–10) classifier using the Euclidean distance for malware classification, using the code provided by the authors[3].

To perform a qualitative analysis, we generated data visualization using the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm [12]. Fig. 4 provides a 2-dimensional t-SNE visualization of the Malimg dataset using four different input features. Each node corresponds to one malware sample and each color represents one malware family.

The figure illustrates that samples of the same malware family are most clustered together in the bottleneck features

[1]Available at http://old.vision.ece.ucsb.edu/spam/malimg.shtml
[2]Available at http://anubis.iseclab.org

[3]Available at http://sarvamblog.blogspot.com.br/2014/08/supervised-classification-with-k-fold.html



(a) (b) (c) (d)

Fig. 4. t-SNE visualization of the Malimg dataset using four different input features: (a) the grayscale image pixels, (b) RGB image pixels, (c) GIST features and (d) bottleneck features.

space, demonstrating that the DCNN activation features indeed provide good representations of malware. Furthermore, it is possible to observe that the operations performed by ResNet-50 transferred layers projected the raw pixels into a better separable feature space, with a degree of separability comparable to the GIST features space.

To perform a quantitative analysis, we implemented malware classification with a kNN classifier using the bottleneck features instead of the GIST features. Using GIST features, the best average accuracy of 0.9748 ($\pm$0.0039) was obtained for $k = 4$, while using bottleneck features we obtained an average accuracy of 0.9800 ($\pm$0.0032) for the same k.

The higher accuracy obtained when using bottleneck features attests that the ResNet-50 layers transferred to our DCNN model generate a meaningful representation for the malware, outperforming the GIST features representation and resulting in successful detection of a high percentage of the malware variants.

To conclude our quantitative analysis, we compare the best results obtained using GIST features and kNN ($k = 4$) classifier with our DCNN results for malware classification. Fig. 5 presents the test accuracy comparison for each method by fold.

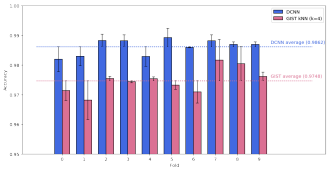


Fig. 5. Accuracy of GIST with kNN (k=4) and DCNN.

In the picture, it is possible to see that the DCNN method outperforms the GIST method in all folds, achieving an average accuracy of 0.9862 ($\pm$0.0025) against an average accuracy of 0.9748 ($\pm$0.0039) achieved by the GIST approach.

V. Conclusion

In this work we propose a malware classification mechanism using byteplot malware images and deep learning techniques. Our results confirm that visual malware similarities can be used for accurate malware classification. We evaluated our approach on a dataset consisting of 9,339 samples from 25 malware families, obtaining an average accuracy of 98.62%.

Whereas many solutions have relied on hand-crafted feature extractors, our approach does not require any feature engineering, using raw pixel values of byteplot images as our underlying malware representation.

Moreover, we demonstrated that the knowledge obtained in the ImageNet classification task can be successfully transferred to malware classification tasks. In our experiments, the feature extractor learned by ResNet-50 performed better than the hand-crafted GIST feature extractor proposed in the literature.

References

[1] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *arXiv preprint arXiv:1409.0575*, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] G. Conti, E. Dean, M. Sinda, and B. Sangster, "Visual reverse engineering of binary and data files," *Visualization for Computer Security*, pp. 1–17, 2008.

[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] 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.

[7] B. Kolosnjaji, A. Zarras, G. D. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences." in *Australasian Conference on Artificial Intelligence*, 2016, pp. 137–149.

[8] 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.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *Springer IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[11] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[12] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.