Malicious Software Classification using VGG16 Deep Neural Network's Bottleneck Features

Edmar Rezende*†, Guilherme Ruppert†, Tiago Carvalho‡, Antonio Theophilo†, Fabio Ramos§ and Paulo de Geus*
*University of Campinas, Campinas, SP, Brazil
†Center for Information Technology Renato Archer, Campinas, SP, Brazil
‡Federal Institute of São Paulo, Campinas, SP, Brazil
§University of Sydney, Sydney, NSW, Australia

***Abstract*—Malicious software (malware) has been extensively employed for illegal purposes and thousands of new samples are discovered every day. The ability to classify samples with similar characteristics into families 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 VGG16 deep neural network's bottleneck features. Malware samples are represented as byteplot grayscale images and the convolutional layers of a VGG16 deep neural network pre-trained on the ImageNet dataset is used for bottleneck features extraction. These features are used to train a SVM classifier for the malware family classification task. The experimental results on a dataset comprising 10,136 samples from 20 different families showed that our approach can effectively be used to classify malware families with an accuracy of 92.97%, outperforming similar approaches proposed in the literature which require feature engineering and considerable domain expertise.**

***Index Terms*—Malicious Software, Classification, Machine Learning, Deep Learning, Transfer Learning.**

I. Introduction

Over the past years, the number of programs developed for malicious and illegal purposes has grown at an extremely high rate. Thousands of new malicious software (malware) samples are discovered every day. Malware authors often reuse code to generate different variants with similar characteristics that can be grouped into one malware family. The ability to identify samples that belong to the same malware family makes significantly easier to derive generalized signatures, implement removal procedures and create new mitigation strategies that work for a whole class of programs.

Several feature extraction approaches based on static and dynamic malware analysis have been used to train machine learning classifiers in order to automate malware classification task. However, designing a feature extractor able to transform raw data into a suitable feature vector from which the learning algorithm can detect patterns requires careful engineering and considerable domain expertise.

In this work we investigate the use deep learning algorithms [1] to learn good malware representations. Deep learning are representation learning methods with multiple levels of representation able to transform the the raw input into a representation at a higher and more abstract level. 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 Neural Networks (DNN) have become the standard approach for many classification tasks within the last few years, due to the overwhelming performance of DNNs on image recognition challenges. Tremendous progress has been made in image recognition, primarily due to the availability of large-scale annotated datasets and the use of DNNs. Large-scale well-annotated datasets are crucial to learning more accurate or generalizable models. The ImageNet [2] is a dataset containing 1.28 million images of 1,000 classes. By the use of transfer learning [3], DNN models trained upon this dataset have been used to significantly improve many image classification tasks using other datasets in different domains.

In this paper we present an approach for malware family classification using the DNN proposed by Visual Geometry Group with 16 layers (VGG16) [4]. First, we represent malware samples as byteplot images, where each byte corresponds to one pixel in a grayscale image. Through transfer learning, we extract the filter activation maps (usually called bottleneck features) using the convolutional layers of VGG16 pre-trained on the ImageNet dataset. The bottleneck features are then used to train a Support Vector Machine (SVM) classifier for the malware family classification task.

Our hypothesis is that despite the disparity between natural images and malware byteplot images, VGG16 parameters may still be transferred to make malware image recognition tasks more effective. The experimental results on a dataset comprising 10,136 samples from 20 different malware families showed that our approach can effectively be used to classify malware families with an accuracy of 92.97%, outperforming similar approaches proposed in the literature which require careful feature engineering and considerable domain expertise.

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.

Kolter and Maloof [5] extracted byte n-grams from Windows executables and trained several classifiers. They used a one-versus-all classification approach and combined the predictions of the individual classifiers. Shabtai et al. [6] evaluated various settings of opcode n-gram sizes and classifiers. The authors concluded that the 2-gram opcodes outperformed the others and the use of byte n-grams appears to produce less accurate classifiers than using opcode n-grams.

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. Nataraj et al. [7] proposed a method for classifying malware represented as byteplot grayscale images using image processing techniques. Using Gabor filters to extract GIST descriptors from the byteplot grayscale images and then using a k-nearest neighbors (kNN) classifier, they obtained an accuracy of 97.18% in a dataset consisting of 25 malware families, totaling 9,458 malware 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. [8] used a hierarchical feature extraction architecture that combines convolutional and recurrent neural network layers for malware classification using system call n-grams obtained from dynamic analysis. Their evaluation results achieved an average accuracy of 89.4% in a dataset containing 4,753 malware samples from 10 different families.

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 employ 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. Methodology

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

In the first step, we convert the malware executable to a byteplot grayscale image. The byteplot representation of binary executables can be used for automatic identification of visual patterns in static malware analysis.

The byteplot grayscale image consists of a variable-resolution image with only one channel, while our DNN model requires constant input dimensionality with 3 channels (RGB). Therefore, in the second step we convert the grayscale image to RGB and rescale it to a fixed resolution of 224×224. Additionally, we subtract the mean RGB value computed on the ImageNet dataset from each pixel of the resulting $224 \times 224 \times 3$ image, as suggested by Krizhevsky et al. [9]. These mean-centered raw RGB values of the pixels are used as input features to our DNN model.

In the third step, we build a Deep Neural Network (DNN) model by transferring convolutional layers of VGG16 model pre-trained on the ImageNet dataset to our DNN model. The transferred convolutional layer's parameters are used to extract the bottleneck features.

In the last step, the bottleneck features are used to train a SVM classifier for malware family classification. This approach is equivalent to replace the fully-connected layers of VGG16 by the SVM classifier freezing the parameters of the convolutional layers during the training process, with the advantage of a much smaller training time.

Finished the training process, the SVM classifier is stacked on the top of the convolutional layers and the whole model is used to classify the test samples.

A. Byteplot Visualization

The byteplot visualization method was initially proposed by Conti et al. [10] to represent binary data objects as grayscale images, where each byte corresponds to one image pixel color rendered as a grayscale (zero is black, 255 is white and other values are intermediate shades of gray). They presented a visual reverse engineering system arguing that visual analysis of binary data presented as grayscale graphical depictions helps distinguish structurally different regions of data and thus facilitates a wide range of analytic tasks such as fragment classification, file type identification, location of regions of interest and other tasks that require an understanding of the primitive data types.

Later, Nataraj et al. [7] observed significant visual similarities in image texture for malware belonging to the same family, as shown in Figure 2, 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. [7]. The height is allowed to vary depending on the width and the file size.

B. VGG16 Architecture

The VGG network architecture was initially proposed by Simonyan and Zisserman [4]. The VGG models with 16 layers (VGG16) and with 19 layers (VGG19) were the basis of their ImageNet Challenge 2014 submission, where the Visual Geometry Group (VGG) team secured the first and the second places in the localization and classification tracks respectively.

The VGG16 architecture, shown at the top of Figure 1, is structured starting with five blocks of convolutional layers followed by three fully-connected layers. Convolutional layers use 3×3 kernels with a stride of 1 and padding of 1 to ensure that each activation map retains the same spatial dimensions as the previous layer. A rectified linear unit (ReLU) activation is performed right after each convolution and a max pooling operation is used at the end of each block to reduce the spatial dimension. Max pooling layers use 2×2 kernels with a stride of 2 and no padding to ensure that each spatial dimension of the activation map from the previous layer is halved. Two fully-connected layers with 4,096 ReLU activated units are then used before the final 1,000 fully-connected softmax layer.

A downside of the VGG16 model is that it is expensive to evaluate and use a lot of memory and parameters. VGG16 has

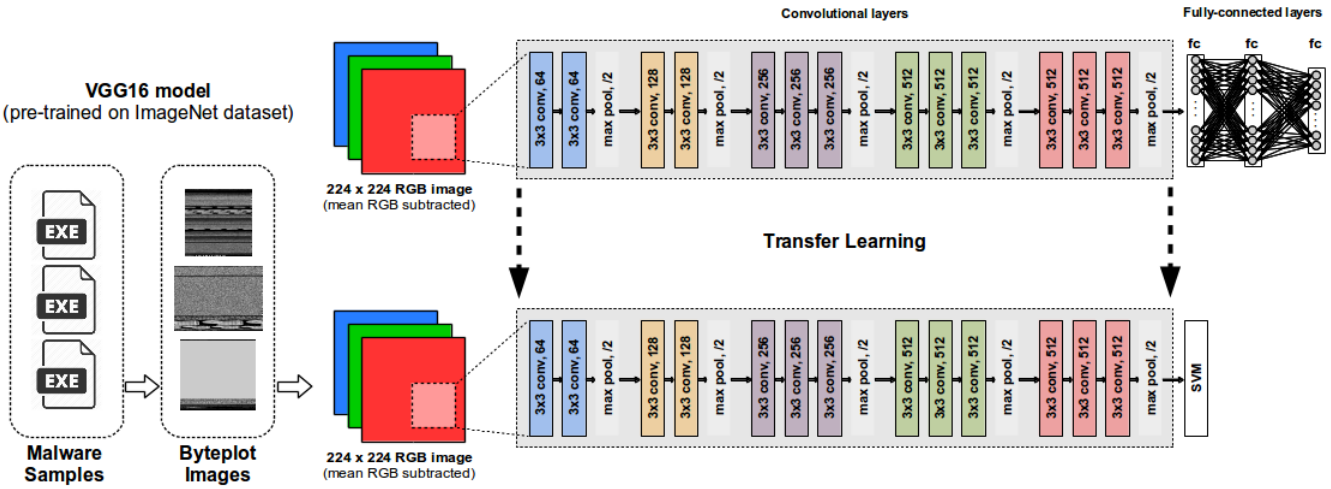


Fig. 1. Overview of proposed method.

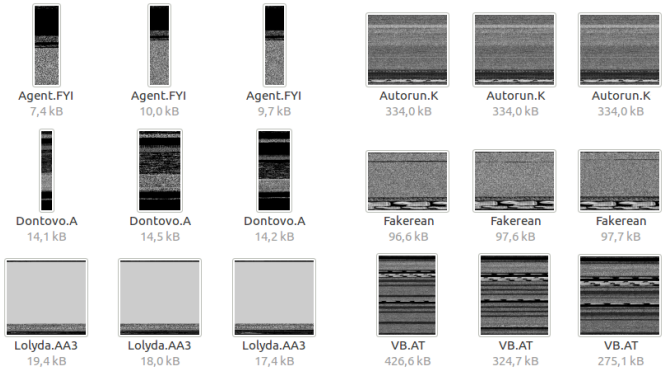


Fig. 2. Byteplot visualization of malware samples from six different families.

approximately 138 million parameters. Most of these parameters (approximately 123 million) are in the fully-connected layers, that are replaced by a SVM classifier in our model, significantly reducing the number of necessary parameters.

C. Transfer Learning

Transfer learning consists in transferring the parameters of a neural network trained with one dataset and task to another problem with a different dataset and task [3]. Many deep neural networks trained on natural images exhibit a curious phenomenon in common: on the first layers they learn features that appear not to be specific to a particular dataset or task, but general in that they are applicable to many datasets and tasks. When the target dataset is significantly smaller than the base dataset, transfer learning can be a powerful tool to enable training a large target network without overfitting.

In the proposed transfer learning approach, we have used VGG16 as the base model, pre-trained for object detection task on the ImageNet dataset. We use the convolutional layers of the VGG16 to extract the bottleneck features of malware byteplot images, that are used as input to train a SVM classifier. Then, we replace the fully-connected layers by the trained SVM classifier in the proposed model.

Our hypothesis is that despite the disparity between natural images and malware byteplot images, VGG16 parameters trained on the large-scale well-annotated ImageNet may still be transferred to make malware image recognition tasks more effective. Collecting and annotating large numbers of malware samples still poses significant challenges. Accordingly, the VGG16 architecture contains millions of parameters to train and thus requires sufficiently large numbers of labeled malware samples.

IV. Experimental Results

In this section, we evaluate the performance of the proposed model and present the experimental results. We applied transfer learning techniques using VGG16 as the base model to extract the bottleneck features of our malware samples, that are used as input to train a SVM classifier. To perform a comparative analysis, we reproduced some approaches used by similar work proposed in the literature.

A. Dataset

The proposed method has been tested over a dataset created with samples collected from VirusSign[1] from August 1, 2014 to January 18, 2015. We obtained 10,136 malware samples and submitted them to Virustotal[2] service to identify their antivirus (AV) labels. Using AVCLASS [11] we have obtained a unique malware family label for each sample. AVCLASS is an automatic labeling tool that given the AV labels for a number of malware samples, outputs the most likely family names for each sample, implementing techniques to address AV label normalization, removal of generic tokens, and alias detection, ranking each candidate family name by the number of AV engines assigned to each sample. The samples are distributed in 20 malware families.

[1]Available at http://www.virussign.com
[2]Available at http://www.virustotal.com

To evaluate the performance of proposed models 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 accuracy by fold, the average classification accuracy and the standard deviation.

B. Feature extraction analysis

To perform a comparative analysis of our model with similar work proposed in the literature, we implemented feature extraction approaches using Gabor filters to extract GIST descriptors from the byteplot grayscale images [7], byte n-grams ($n = 1$) [5] and opcode n-grams ($n = 1$) [6].

We are interested in evaluate how good the VGG16 bottleneck features are compared with other feature extraction methods. To perform a qualitative analysis, we generated data visualization using the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm [12]. The goal of t-SNE is to reduce the dimensionality so that the closer two nodes are to each other in the original high-dimensional space, the closer they would be in the 2-dimensional space.

Figure 3 provides a t-SNE visualization of the dataset using different feature extraction approaches. Each node corresponds to one malware sample and each color represents one malware family. Note that the t-SNE dimensionality reduction process is completely unsupervised and the labels are used for coloring the nodes at plotting time only.

It is possible to observe that the operations performed by the VGG16 convolutional layers projected the grayscale image pixels into a better separable feature space, with a degree of separability comparable to the other feature extraction techniques. Furthermore, it is possible to observe that samples of the same malware family are most clustered together in the VGG16 bottleneck features space, demonstrating that the VGG16 activation features indeed provide good representations of malware. Some clustering errors are expected here (as can be seen in the visualization), since many of these malware families use parts of code from each other, and the distinction even among antivirus detections is blurred.

To perform a quantitative analysis of features, we implemented malware classification with a kNN ($k = 1$) classifier using as input the same features. The accuracy by fold, the average accuracy and the standard deviation are presented in Table I.

VGG16 bottleneck features obtained an average accuracy of 0.9077 ($\pm$0.0064), while other feature extraction approaches obtained lower accuracies.

C. Malware Classification Results

To demonstrate the performance gain provided by the transfer of convolutional layers of VGG16 pre-trained on the ImageNet dataset, we have trained a VGG16 from scratch for the malware family classification task. VGG16 has been trained with categorical cross-entropy cost function and Adam optimizer for 100 epochs. The weights have been initialized using glorot uniform approach and the bias terms were initialized to zero. Figures 4(a) and 4(b) present, respectively, the average loss and accuracy of VGG16 trained from scratch.

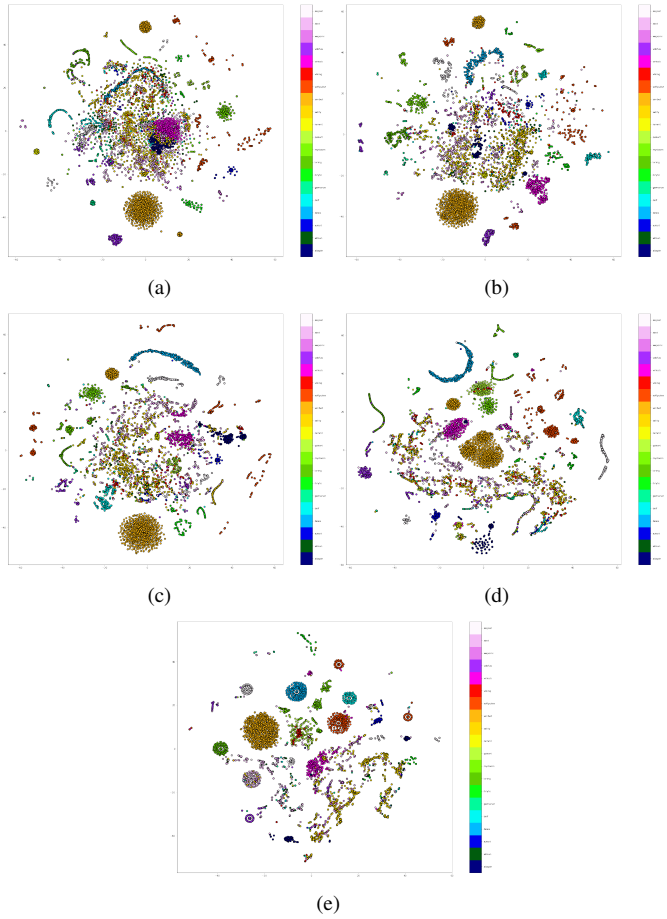


Fig. 3. t-SNE visualization of extracted features: (a) grayscale image pixels, (b) VGG16 bottleneck features, (c) GIST descriptors, (d) Byte 1-gram and (e) Opcode 1-gram.

The network converges quickly to an extremely low average accuracy of 0.1128 ($\pm$0.0434). While in principle VGG16 network is a powerful model, in practice, it is hard to train properly. The reasons why this model is so unwieldy are the vanishing and exploding gradient problems.

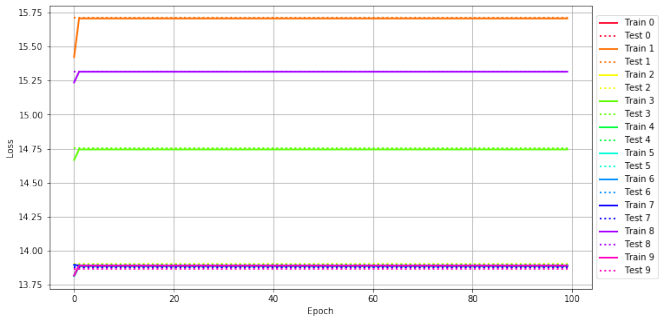
In the proposed approach, we use the convolutional layers of VGG16 pre-trained on the ImageNet dataset to extract bottleneck features which are used to train a SVM classifier with Radial Basis Function (RBF) kernel for the malware family classification task. Then, we replace the VGG16 fully-connected layers by the trained SVM classifier.

The parameters C and gamma of the SVM classifier have been obtained through a gridsearch process with $C \in \left[10^{-2}, 10^{-1}, ..., 10^{10}\right]$ and $gamma \in \left[10^{-9}, 10^{-8}, ..., 10^{3}\right]$. Figure 5 shows the accuracy obtained in gridsearch using VGG16 bottleneck features.

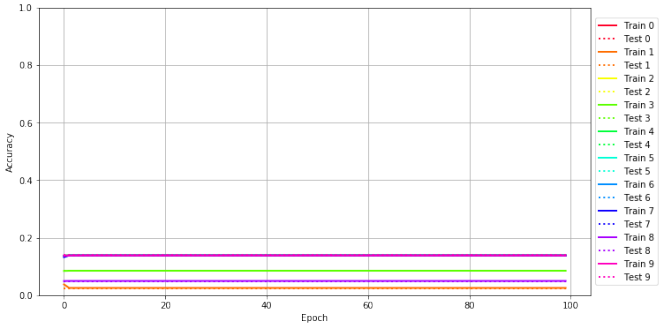
With $C = 100$ and $gamma = 10^{-6}$ we obtained the best average accuracy of 0.9297 ($\pm$0.0063) using VGG16 bottle-

TABLE I
ACCURACY OBTAINED WITH A KNN (K=1) CLASSIFIER.

Features	Fold										Avg Acc	Std Dev
	0	1	2	3	4	5	6	7	8	9		
Grayscale	0.7705	0.7769	0.7765	0.7520	0.7569	0.7587	0.7453	0.7540	0.7664	0.7550	0.7612	0.0108
VGG16	0.9043	0.9090	0.9108	0.8967	0.9075	0.8981	0.9138	0.9077	0.9135	0.9153	0.9077	0.0064
GIST	0.9014	0.8943	0.8961	0.8858	0.8878	0.8912	0.8949	0.8899	0.9016	0.9004	0.8943	0.0057
Byte 1-gram	0.8340	0.8415	0.8363	0.8297	0.8248	0.8437	0.8404	0.8393	0.8380	0.8406	0.8368	0.0058
Opcode 1-gram	0.8799	0.9031	0.8971	0.8996	0.8839	0.8783	0.8890	0.8919	0.8897	0.9014	0.8914	0.0089



(a)



(b)

Fig. 4. VGG16 train/test average (a) loss and (b) accuracy.

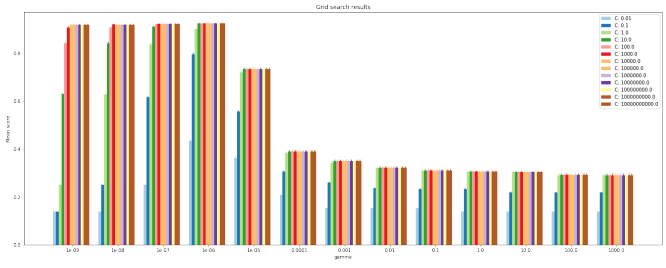
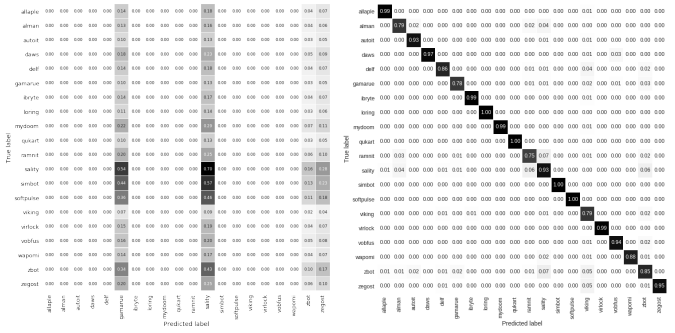


Fig. 5. Gridsearch of C and gamma parameters using VGG16 bottleneck features with SVM classifier.

neck features. Table II presents the comparison of accuracy obtained with VGG16 trained from scratch and VGG16 with transfer and SVM.

Figure 6 presents the normalized confusion matrix obtained with VGG16 trained from scratch and VGG16 with transfer and SVM.

As shown in the picture, the VGG16 trained from scratch is able to recognize only four malware families, while VGG16



(a) (b)

Fig. 6. Normalized confusion matrix of (a) VGG16 trained from scratch and (b) VGG16 with transfer and SVM.

with transfer and SVM is able to identify all families with a high accuracy.

V. CONCLUSION

In this work we propose a malware classification mechanism using byteplot malware images and deep learning techniques. We evaluated our approach on a dataset consisting of 10,136 malware samples from 20 malware families, obtaining an average accuracy of 92.97%. The experimental results show that our method achieved a better accuracy compared to similar work proposed in the literature. Our results confirm that visual malware similarities can be used for accurate malware classification.

Whereas many solutions have relied solely on hand-crafting representations obtained by static and dynamic feature extraction procedures, the use of deep learning algorithms seems to be a promising alternative to discover good malware representations without laborious feature engineering process. Moreover, we demonstrated that the knowledge obtained in the ImageNet classification task can be successfully transferred to malware classification. In our experiments, the accuracy obtained with VGG16 using transfer learning and SVM outperformed VGG16 trained from scratch.

The VGG16 learned feature extractor can still be fine-tuned to malware classification, backpropagating the errors from the last layers into the VGG16 transferred convolutional layers to fine-tune them, possibly improving the performance of the classifier.

TABLE II
COMPARISON OF ACCURACY OBTAINED WITH VGG16 TRAINED FROM SCRATCH AND VGG16 WITH TRANSFER AND SVM.

Model	Transfer	Top	Fold										Avg Acc	Std Dev
			0	1	2	3	4	5	6	7	8	9		
VGG16	no	fully-connected	0.1377	0.0254	0.1373	0.0846	0.1378	0.1385	0.1388	0.1389	0.0497	0.1394	0.1128	0.0434
VGG16	yes	SVM	0.9297	0.9364	0.9333	0.9154	0.9301	0.9268	0.9376	0.9276	0.9264	0.9333	0.9297	0.0063

ACKNOWLEDGMENT

This work has been partially supported by Brazilian National Council for Scientific and Technological Development (grants 302923/2014-4 and 313152/2015-2). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPUs used for this research.

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," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[3] 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.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] J. Z. Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild," *The Journal of Machine Learning Research*, vol. 7, pp. 2721–2744, 2006.

[6] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici, "Detecting unknown malicious code by applying classification techniques on opcode patterns," *Security Informatics*, vol. 1, no. 1, pp. 1–22, 2012.

[7] 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.

[8] 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.

[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] 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.

[11] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "Avclass: A tool for massive malware labeling," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2016, pp. 230–253.

[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.