Classifying Malware Images with Convolutional Neural Network Models

Ahmed Bensaoud, Nawaf Abudawaood, and Jugal Kalita
(Corresponding author: Ahmed Bensaoud)

Department of Computer Science, University of Colorado Colorado Springs
1420 Austin Bluffs Pkwy, Colorado Springs, CO 80918, USA
(Email: abensaou@uccs.edu)

(Received Dec. 27, 2019; Revised and Accepted May 23, 2020)

Abstract

Due to increasing threats from malicious software (malware) in both number and complexity, researchers have developed approaches to automatic detection and classification of malware, instead of analyzing methods for malware files manually in a time-consuming effort. At the same time, malware authors have developed techniques to evade signature-based detection techniques used by antivirus companies. Most recently, deep learning is being used in malware classification to solve this issue. In this paper, we use several convolutional neural network (CNN) models for static malware classification. In particular, we use six deep learning models, three of which are past winners of the ImageNet Large-Scale Visual Recognition Challenge. The other three models are CNN-SVM, GRU-SVM and MLP-SVM, which enhance neural models with support vector machines (SVM). We perform experiments using the Malimg dataset, which has malware images that were converted from Portable Executable malware binaries. The dataset is divided into 25 malware families. Comparisons show that the Inception V3 model achieves a test accuracy of 99.24%, which is better than the accuracy of 98.52% achieved by the current state of the art system called the M-CNN model.

Keywords: Convolutional Neural Network; Malware Classification; Malware Detection; ImageNet

1 Introduction

Internet connectivity is an essential infrastructure for business organizations, banking institutions, universities, and governments, and is growing exponentially. This growth is threatened by attackers with malicious codes and network threats [41]. The execution of malware forces a computer to perform operations that are not normal, and may harm a victim's computer systems. The amount of malware in circulation has been increasing rapidly in the recent years, and malware has affected computer systems all over the world [21]. Thousands of malware files are being created daily. Figure 1 presents annual statistics of malware attacks over the last 10 years, showing that the total number of malware in circulation has increased to more than 900 million in 2019, which is a 2000% increase compared to the number of malware in the year 2010 [26].

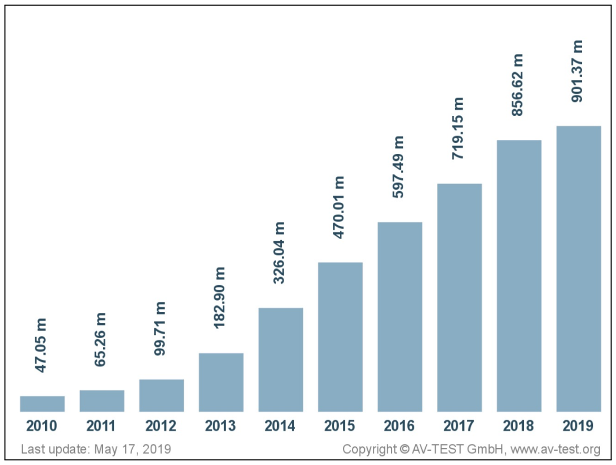

Figure 1: Number of worldwide malware attacks for the last ten years [26].

The cost of malware infection can run into millions of dollars for each incident inflicted upon small and medium sized businesses [32]. Routing protocols alone are not sufficient to detect malware [48]. As a result, researchers and anti-virus vendors employ machine learning to detect and classify malicious software. A large number of studies have focused on malware binary since binaries are normally used to infect computers. Malware is analyzed based on static as well as dynamic analysis. While static analysis extracts malware features that can be used to detect or classify malware employing machine learning, dynamic analysis analyzes malware behavior as it is executed in a controlled environment like Cuckoo Sandbox [14], which is open source, available on GitHub.

Various traditional machine learning approaches such as support vector machine [20], k-nearest neighbors [11], random forests [24], naive bayes [8] and decision tree [31] have been used to detect and classify known malware. In particular, Nataraj et al. [28] proposed a method for visualizing and classifying malware using image processing methods, which first converts malware binaries to grayscale images. Techniques from computer vision, particularly for image classification can be used to obtain high accuracies.

Researchers have classified malware using CNN models, initially used for image classification [36]. It is obvious that in order to use such an approach, the malware binary must first be converted to an "image". The ANN models used include simple multilayer perceptron, and a mix of GRU-based RNNs and CNNs. Kalash et al. [17] used a CNN model called M-CNN, based on a well-known image classification architecture called VGG-16 [37]. Methods have also replaced the last layer of an artificial neural network with an SVM classifier [30].

In this paper, we compare the performance of several CNN-based models which had achieved state-of-the-art results for malware image classification with the CNN-mixed models used by Agarap and Pepito [2], the CNN models we choose have performed well in the large-scale image classification contest called ILSVRC [34], within the last few years.

The paper is organized in the following way. In the next section, we briefly review related work. Section 3 describes the methodology used to classify malware. Section 4 discusses experimental results. Lastly, Section 5 concludes the paper and discusses plans for future work.

2 Related Work

Below, we discuss research effects that primarily convert malware binaries to images before classifying them. Approaches based on traditional machine learning depend on manual feature extraction. Deep learning can extract useful features automatically by avoiding manual feature extraction.

2.1 Methods Based on Traditional Machine Learning

Grayscale images can be extracted from the raw malware executable files showing features of malware [29] [28] [22]. Such images enable analysis of malware by extracting visual features. Nataraj et al. [28] were the first to explore the use of byte plot visualization as grayscale images for automatic malware classification. They used a malware image dataset consisting of 9,342 malware samples belonging to 25 different classes. They extracted GIST [43] features from the grayscale images and classified them using K-nearest neighbor classification with Euclidean distance as metric. Their approach had high computational overhead. Mirza et al. extracted features from malware files and combined decision trees, support vector machines and boosting to detect malware [27]. Zhang et al. proposed a static analysis technique based on n-grams of opcodes to classify ransomware families [49]. Makandar and Patrot [25] used multi-class support vector machine malware classification with malware input as images. They used wavelet transform to build effective texture based feature vectors from the malware images. This reduced the dimensionality of the feature vector and the time complexity.

2.2 Methods Based on Deep Learning

Several studies on malware classification have been performed using CNN architectures. Cui et al. [6] detected code variants that are malicious after converting to grayscale images and using a simple CNN model. Kalash et al. [17] classified malware images by converting malware files into grayscale images, using two different datasets, Malimg [28] and Microsoft malware [33]. They obtained 98.52% and 99.97% accuracies, respectively. Yue [47] proposed a weighted softmax loss for CNNs for imbalanced malware image classification, and achieved satisfactory classification results. Gilbert. et al. [12] built a model consisting of three convolutional layers with one fully connected layer and tested on two datasets, Microsoft Malware Classification Challenge dataset and Malimg dataset. Seonhee et al. [35] proposed a malware classification model using a CNN that classified malware images. Their experiments were divided into two sets. The first set of experiments classified malware into 9 families and obtained accuracies of 96.2%, 98.4% considering the top-1 and top-2 ranked results. The second set of experiments classified malware into 27 families and obtained 82.9% and 89% top-1 and top-2 accuracies. Tobiyama et al. [42] proposed a malware process detection method by training a recurrent neural network (RNN) to extract features of process behavior, and then training a CNN to classify features extracted by the trained RNN. Vinayakumar et al. proposed a deep learning model based on CNN and LSTM for malware family categorization. Experiments showed an accuracy of 96.3% on the Malimg dataset [46]. Su et al. [38] created one-channel grayscale images from executable binaries in two families, and classified them into their related families using a light-weight convolutional Neural Network. They achieved a accuracies of 94.0% and 81.8% for malware and goodware, respectively.

3 Methodology

In this paper, we use six CNN models for malware classification, considering malware binaries as images.

3.1 Malware Binaries

The malware binaries we use are in Portable Executable (PE) form. Generally, PE files are programs that have file name extensions such as .bin, .dll and .exe. PE files are usually recognized through their components, which are called .tex, .rdata, .data and .rsrc. The first component, called .text, is the code section, containing the program's instructions. .rdata is the part that contains read only data, and .data is the part that contains data that can be modified, and .rsrc is the final component that stands for resources used by the malware.

Malicious data binaries can be converted 8 bits at a time to pixels in a grayscale image, consisting of textural patterns. In Figure 2, we see the sections of a malware binary showing different textures, when seen as an image [28]. Based on these patterns, we can classify malware. In this paper, we use the Malimg dataset [28] which is a set of grayscale images corresponding to malware binaries saved in .jpg format. Some examples of malware families are shown in Figure 3.

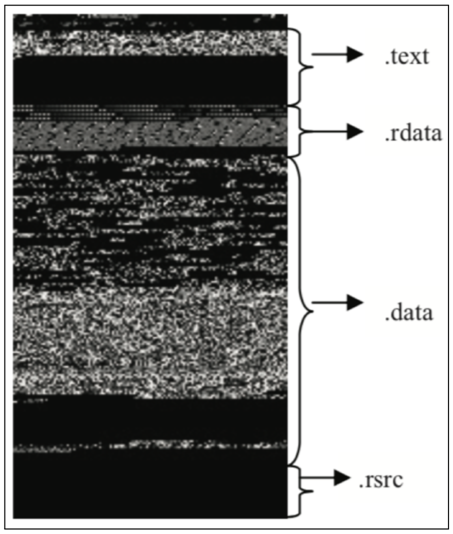

Figure 2: Portable Executable file represented as an image.

3.2 Malware as Image

Researchers and practitioners can understand malware better by visualizing malware binaries as images since, the patterns within such images become clearly visible. Finding patterns within images can be performed well by deep learning [13]. The most important patterns of features in the malware images can be used to identify the malware families also. Images for a specific malware family have similar patterns, allowing a deep learning model to recognize important patterns using automatic extraction of features. In particular, CNN models are good at classifying images because they can extract relevant features within an image by subsampling through convolutions, pooling and other computations. In this case, CNNs look for the most relevant features within an image from a specific malware family for the purpose of classification [6]. Malware binaries can be translated into an images using an algorithm that converts a binary PE file into a sequence of 8 bit vectors or hexadecimal values. An 8 bit

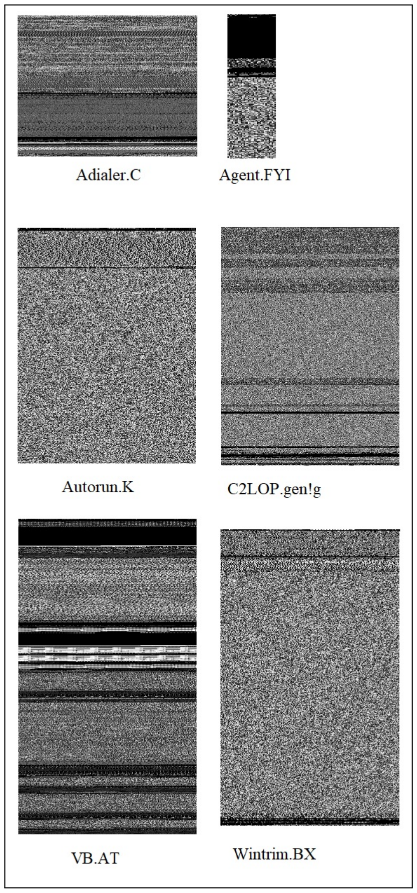

Figure 3: Sample images of malware belonging to different families.

vector can be represented in the range 00000000 (0) to 11111111 (255). Each 8 bit vector represents a number, and can be converted into pixel in a malware image, as shown in Figure 3. Images obtained from different malware families have different characteristics [17].

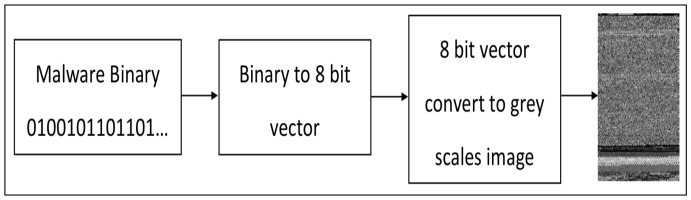

Figure 4: Converting malware binary to an image.

3.3 Problem Statement

The problem that we solve in this paper is classification of malware object code into malware families. We have 9,342 malware samples given in the form of images obtained from their object codes. There are 25 malware families, with the biggest family containing 2,950 samples and the smallest containing 81 samples. We classify these images using deep learning models that have performed well in image classification.

3.4 Motivation and Approach

CNNs have performed well for classification in a variety of domains including object recognition [19], image classification [23], and video classification [18]. CNNs have shown superior performance compared to traditional learning algorithms, especially in tasks such as image classification. Since we represent malware object codes as images, we classify malware based on their corresponding images using CNN models. Malware images are classified into families by extracting patterns within them, because binary image files generated from a malware family are likely to produce similar images. Feature extraction allows image classification models to recognize patterns based on pixel distribution in an image. Before CNNs, features were extracted manually, and it was one of the biggest challenges in image classification. The ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [34] has led to sophisticated CNN-based classification models that have achieved excellent results, demonstrating that the models are likely to perform well in static analysis of malware.

In this paper, we compare the performance of several CNNs-based models for classification of malware binaries that have been converted to images. In particular, we compare the performance of several well-known CNNs-based deep learning models from the ILSVRC competitions and a few additional CNN and CNN-mixed models to classify malware images, that automatically extract features based on the static analysis approach. These models are publicly available.

3.5 CNN Models Used

The experimental work of this paper is to run six deep learning models to classify malware images to detect malware. These models are briefly described below.

3.5.1 VGG16

The first model we use is called VGG-Net16 [37], which was the winner of ILSVRC in 2014. Its contribution was in increasing the depth using 3x3 convolution filters that are small, allowing them to increase the number of layers from 16 to 19. The depth of the representation was very helpful in increasing the accuracy of image classification. On the ImageNet dataset, the VGG model outperformed many complicated models, signifying the importance of the depth.

3.5.2 Inception V3

The Inception V3 model contains 42 layers, and is an improvement over the GoogleNet Inception V1 model that was the winner of ILSVRC in 2015 [39]. The Inception V3 model architecture starts with a 5x Inception module A, 4x Inception module B, 2x Inception module C, and 2x grid size reduction; one of the grid size reductions is done with some modification, and the second one is applied without any modification. An auxiliary classifier is also applied as an extra layer to help improve the results.

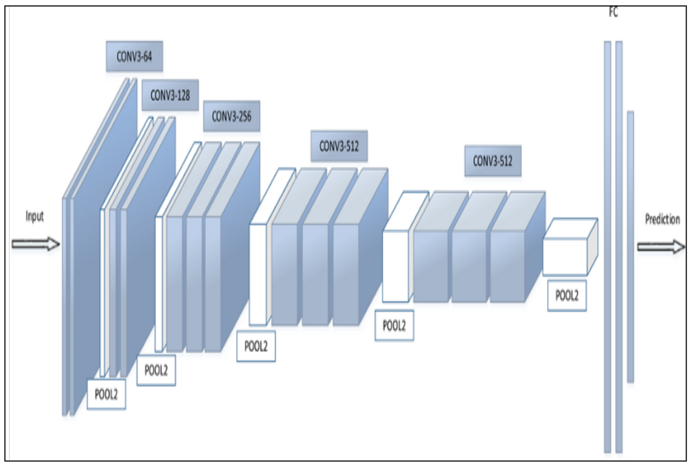

Figure 5: VGG-16 model architecture [3].

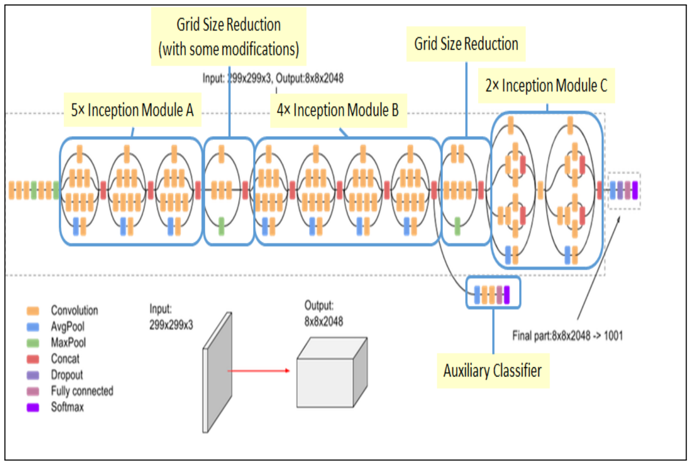

Figure 6: Inception V3 model architecture [45].

3.5.3 ResNet50

The third model we use is called Residual Networks (ResNet50) [15]. ResNet50 was the winner of ILSVRC in 2016. The novel technique that this model introduced provides extra connections between non-contiguous convolutional layers, using shortcut connections. This technique allowed the model to skip through layers to deal with vanishing gradients in order to achieve lower loss and better results. The network had 152 layers, an impressive 8 times deeper than a comparable VGG network. This is an improvement over the VGG16 model with Faster R-CNN, producing an improvement of 28% in accurcy in image classification. The architecture of the original ResNet50 is illustrated in Figure 7.

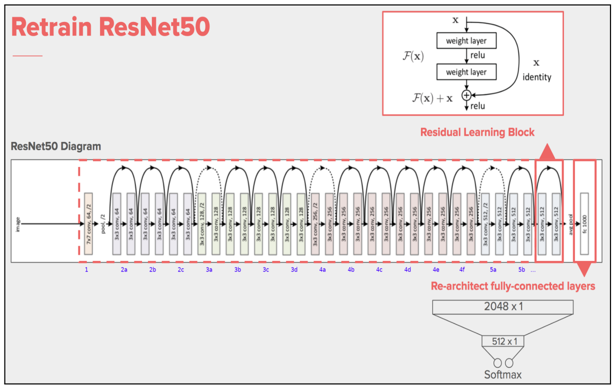

Figure 7: ResNet50 model architecture [5].

3.5.4 CNN-SVM Model

For classification, deep learning models usually use the softmax activation function as the top layer for prediction and minimization of cross-entropy loss. Tang [40] replaced the softmax layer with a linear SVM and applied it on MNIST and CIFAR-10 datasets, and the ICML 2013 Representation Learning Workshop's face expression recognition challenge. The SVM is a linear maximum margin classifier. CNN-SVM allowed for extraction of features for input images with a linear SVM [9]. Agarap and Pepito [2] applied CNN-SVM [40] on Maling and achieved 77.22% accuracy.

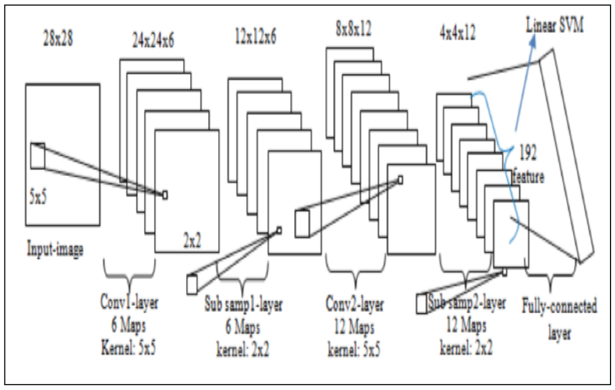
28x28
24x24x6
12x12x6
8x8x12
4x4x12
Linear SVM
5x5
Input-image
2x2
192 feature
Conv1-layer 6 Maps Kernel: 5x5
Sub sampl-layer 6 Maps kernel: 2x2
Conv2-layer 12 Maps kernel: 5x5
Sub samp2-layer 12 Maps kernel: 2x2
Fully-connected layer

Figure 8: Architecture of CNN-SVM [7].

3.5.5 GRU-SVM Model

Agarap and Pepito [2] modified the architecture of a Gated Recurrent Unit (GRU) RNN by using SVM as its final output layer for use in a binary, non-probabilistic classification task (see Figure 8). They used GRU-SVM on the Malimg dataset and achieved 84.92% accuracy.

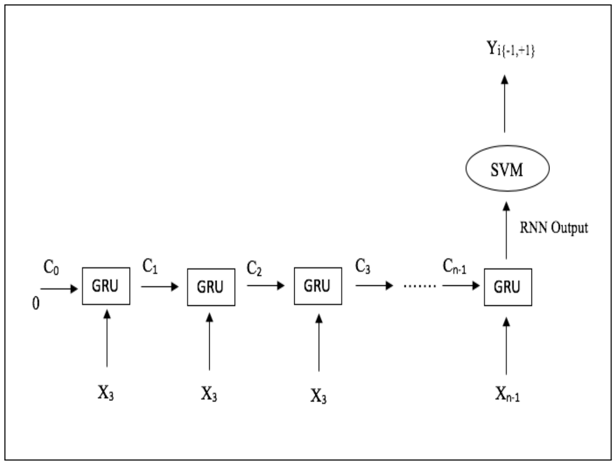

Figure 9: GRU-SVM architecture model, with n GRU cells and SVM for the classification function [1].

3.5.6 MLP-SVM Model

Bellili et al. [4] proposed MLP-SVM for handwritten digit recognition. MLP-SVM is a model that combines both SVM and Multilayer Perceptrons for the classification of binary image. Multilayer Perceptrons are a fully connected network that allows for the inputs to get classified using input features. The MLP-SVM is a hybrid model that run the MLP and SVM classifiers in parallel. The MLP-SVM model was used by Agarap and Pepito [2] on the Malimg dataset with 80.46% accuracy.

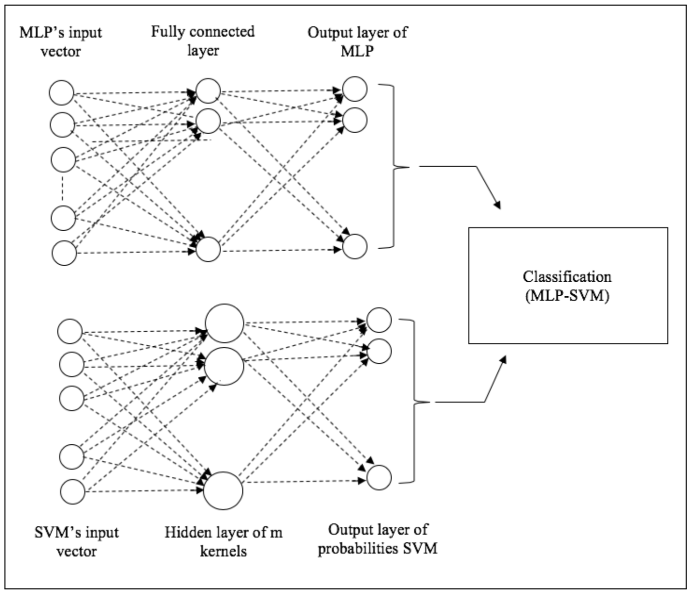

Figure 10: MLP-SVM architecture model [44].

3.6 Dataset

There are a few malware datasets available for academic research. One of the these datasets is Malimg [28]. The dataset contains 9,342 malware images, classified into 25 malware families. The widths and lengths of the malware images vary. The images have been created from various malware families such as Dialer, Backdoor, Worm, Worm-AutoIT, Trojan, Trojan-Downloader, Rouge and PWS. All malware images are PE files that were first converted to an 8-bit vector binary, and then to images. The malware image sizes were modified, so that they can be input to a CNN model. The family breakdown for the Malimg dataset is shown in Table 1.

Table 1: 25 malware families (classes) and the number of samples in each family.

Malware Family	**Samples**	**Malware kind**
Adialer.C	123	Dialer
Agent.FYI	117	Backdoor
Allaple.A	2950	Worm
Allaple.L	1592	Worm
Alueron.gen!J	199	Trojan
Autorun.K	107	Worm AutoIT
C2LOP.gen!g	201	Trojan
C2LOP.p	147	Trojan
Dialplatform.B	178	Dialer
Donoto.A	163	Trojan Downloader
Fakerean	382	Rouge
Instaccess	432	Dialer
Lolyada.AA1	214	PWS
Lolyada.AA2	185	PWS
Lolyada.AA3	124	PWS
Lolyada.AT	160	PWS
Malex.gen!J	137	Trojan
Obfuscator.AD	143	Trojan Downloader
RBot!gen	159	Backdoor
Skintrim.N	81	Trojan
Swizzor.gen!E	129	Trojan Downloader
Swizzor.gen!I	133	Trojan Downloader
VB.AT	409	Worm
Wintrim.BX	98	Trojan Downloader
Yuner.A	801	Worm

4 Experimental Results

All experiments in this study were conducted on NVIDIA GeForce GTX 1080 Ti GPU. As stated, we ran six models on the Malimg dataset: Inception V3, VGG16-Net, ResNet50, CNN-SVM, MLP-SVM and GRU-SVM. Since the Malimg dataset is not similar to the ImageNet dataset, we could not directly use grayscale images with VGG16 and ResNet50 because the input layers require the shape of (3, 224, 224). The 3 is represents Red, Green and Blue (RGB) channels of the image, whereas the grayscale images require (1, 224, 224). VGG16 and ResNet50 showed low performance compared to the other models, since both of these models architectures were designed to recognize colored images that requires RGB format. Therefore, both give low accuracies when tested on the grayscale images. The results for malware prediction using all these models are shown in Table 2 and Figure 11. The Inception V3 model had a significantly higher accuracy at 99.24%. Table 4 shows the best predicted accuracies of the six models when run 10 times. CNN-SVM, GRU-SVM, and MLP-SVM performed well but VGG16 and ResNet50 performed poorly compared to the Inception V3 model. We provide the results of testing the dataset with several traditional models as well as other deep learning models in Table 4.

5 Conclusions and Future Work

These days many antivirus programs rely on deep learning techniques to protect devices from malware. Deep learning architectures have achieved good performance in detecting malware when used with Windows PE binaries. We have presented the performance comparison among six classifiers on a malware image dataset created from PE files. We used the models from the ImageNet Large-Scale Visual Recognition Challenge and three other CNN models to classify grayscale malware images. We successfully trained the six models on the Malimg dataset, and the results indicate that the Inception-V3 model outperforms all compared work. To the best of our knowledge, it is the state-of-the-art of performance in classification on grayscale malware images.
Future work will be focused on conducting results using additional models from leaderboards of image classification competitions. We also want to convert malware images into color RGB images before classification.

References

[1] A. Abien, "A Neural Network Architecture Combining Gated Recurrent Unit (GRU) and Support Vector Machine (SVM) for Intrusion Detection in Network Traffic Data," in *The 10th International Conference on Machine Learning and Computing*, pp. 26–30, 2018.

[2] A. F. Agarap and F. J. H. Pepito, "Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (SVM) for malware classification," *ArXiv*, 2017. (`arXiv:1801.00318`)

[3] S. Bansal, *CNN Architectures : VGG, ResNet, Inception + TL*, 2018. (`https://www.kaggle.com/shivamb/cnn-architectures-vgg-resnet-inception-tl`).

[4] A. Bellili, M. Gilloux, and P. Gallinari, "An hybrid MLP-SVM handwritten digit recognizer," in *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pp. 28–32, 2001.

[5] A. Ciurana, *How to split resnet50 model from top as well as from bottom?*, 2019. (`https://stackoverflow.`

Table 2: Prediction accuracies of the six tested models.

Family	**CNN-SVM**	**GRU-SVM**	**MLP-SVM**	**Inception V3**	**ResNet 50**	**VGG16**
	Prediction Accuracy					
Adialer.C	**99.80%**	99.15%	99.51%	99.40%	23.18%	13.62%
Agent.FYI	95.12%	95.86%	94.87%	**99.50%**	25.41%	14.81%
Allaple.A	94.98%	97.71%	94.32%	**99.72%**	26.94%	14.47%
Allaple.L	99.10%	95.35%	95.48%	**99.73%**	21.52%	15.53%
Alueron.gen!J	96.42%	97.57%	93.20%	**99.48%**	21.37%	15.93%
Autorun.K	92.99%	93.38%	96.68%	**99.06%**	23.27%	14.38%
C2LOP.gen!g	94.75%	93.70%	94.49%	**98.42%**	28.87%	13.78%
C2LOP.P	97.11%	93.45%	95.43%	**99.67%**	27.48%	14.96%
Dialplatform.B.	95.34%	94.85%	96.17%	**99.86%**	23.84%	14.78%
Dontovo.A	97.53%	89.81%	93.44%	**98.25%**	29.76%	15.03%
Fakerean	98.46%	92.11%	93.11%	**98.91%**	26.29%	12.45%
Instantaccess.	93.17%	96.75%	96.63%	**98.24%**	30.15%	13.11%
Lolyda.AA1	91.30%	94.09%	93.97%	**99.40%**	23.79%	14.26%
Lolyda.AA2	89.10%	94.36%	91.64%	**99.34%**	28.32%	13.80%
Lolyda.AA3	87.44%	90.61%	94.13%	**97.39%**	29.59%	13.85%
Lolyda.AT	81.31%	92.51%	90.28%	**99.86%**	31.67%	13.99%
Malex.gen!J	88.79%	94.99%	94.61%	**99.31%**	25.39%	15.22%
Obfuscator.AD.	86.57%	94.76%	96.74%	**99.50%**	21.84%	12.64%
Rbot!gen.	87.60%	93.39%	97.19%	**98.81%**	32.49%	14.45%
Skintrim.N	96.16%	84.10%	87.21%	**99.55%**	34.81%	15.84%
Swizzor.gen!E.	82.45%	96.72%	98.54%	**99.57%**	17.22%	15.30%
Swizzor.gen!I	97.57%	98.14%	96.80%	**99.29%**	33.57%	14.55%
VB.AT	99.36%	98.72%	98.77%	**99.34%**	31.68%	13.92%
Wintrim.BX	99.78%	97.71%	**99.92%**	99.88%	31.71%	15.65%
Yuner.A	88.26%	84.44%	80.64%	**99.79%**	16.38%	11.54%

Table 3: Accuracy averages of the six tested models.

Models	**CNN-SVM**	**GRU-SVM**	**MLP-SVM**	**Inception V3**	**ResNet 50**	**VGG16**
	Average of prediction accuracy					
	93.22%	94.17%	94.55%	99.25%	26.66%	14.31%

Table 4: Comparison of malware detection models, including models we tested.

Model	**VGG16**	**ResNet50**	MLP-SVM [2]	CNN-SVM [2]	GRU-SVM [2]	Random Forest [10]	**MLP-SVM**	**GRU-SVM**	CNN [16]	M-CNN [17]	**CNN-SVM**	**Inception V3**
Accuracy	15.92	35.10%	80.46%	77.22%	84.92%	95.26%	97.25%	97.43%	98.00%	98.52%	99.11%	**99.24%**

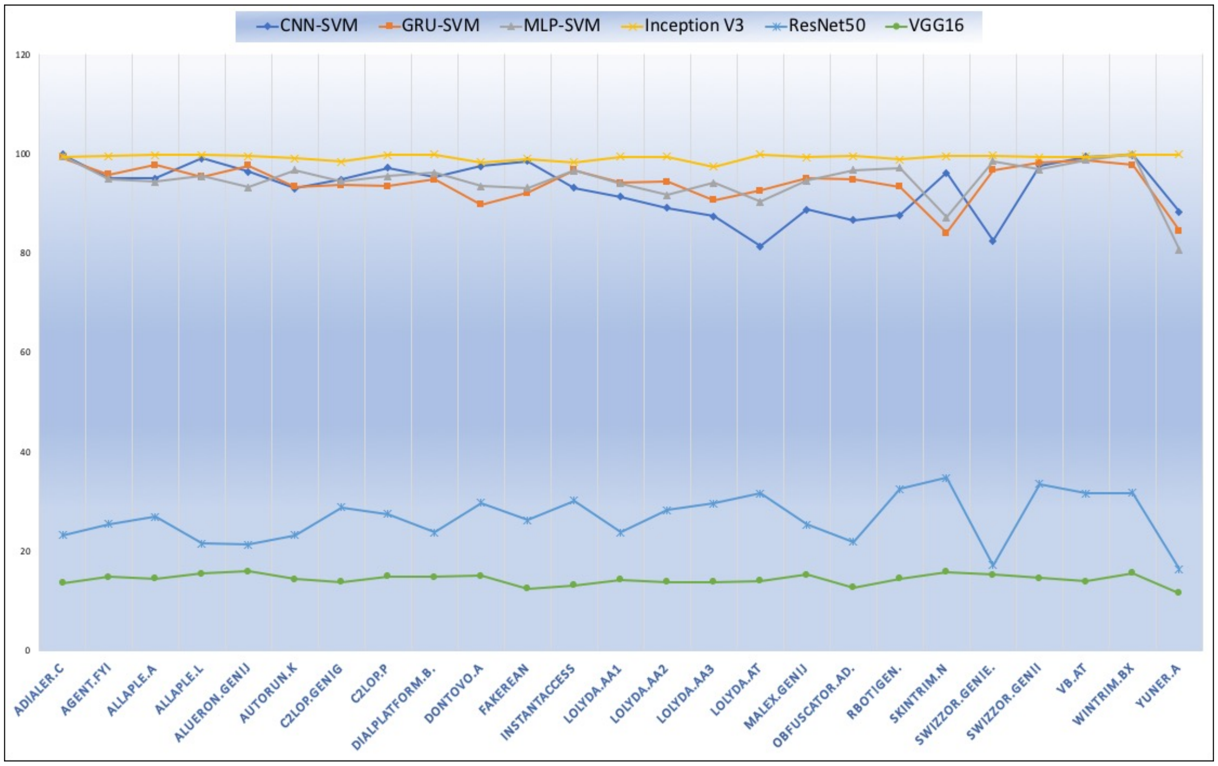

Figure 11: Prediction accuracy of six models

com/questions/54207410/how-to-split-resnet50-model-from-top-as-well-as-from-bottom).

[6] Z. Cui, F. Xue, X. Cai, Y. Cao, G. G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187–3196, 2018.

[7] Darmatasia and M. I. Fanany, "Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM)," *The 5th International Conference on Information and Communication Technology*, pp. 1–6, 2017.

[8] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, no. 2-3, pp. 103–130, 1997.

[9] M. I. Fanany, "Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM)," in *The 5th International Conference on Information and Communication Technology*, pp. 1–6, 2017.

[10] F. C. C. Garcia, I. I. Muga, and P. Felix, "Random forest for malware classification," *Cryptography and Security*, 2016. (arXiv:1609.07770)

[11] F. Gianfelici, "Nearest-neighbor methods in learning and vision," *IEEE Transactions on Neural Networks*, vol. 19, no. 2, pp. 377–377, 2008.

[12] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using convolutional neural networks for classification of malware represented as images," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 15–28, 2019.

[13] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, and J. Cai, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, no. 354–377, 2018.

[14] C. Guarnieri, M. Schloesser, J. Bremer, and A. Tanasi, "Cuckoo Sandbox open source automated malware analysis," *Black Hat USA*, 2013. (https://media.blackhat.com/us-13/US-13-Bremer-Mo-Malware-Mo-Problems-Cuckoo-Sandbox-WP.pdf)

[15] 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*, pp. 770–778, 2016.

[16] E. K. Kabanga and C. H. Kim, "Malware images classification using convolutional neural network," *Journal of Computer and Communications*, vol. 6, no. 01, pp. 153, 2017.

[17] M. Kalash, M. Rochan, N. Mohammed, N. Bruce, Y. Wang, and F. Iqbal, "Malware classification with deep convolutional neural networks," in *The 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS'18)*, pp. 1–5, 2018.

[18] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[19] K. Kavukcuoglu, P. Sermanet, Y. L. Boureau, K. Gregor, M. Mathieu, and Y. L. Cun, "Learning convolutional feature hierarchies for visual recognition," in *Advances in Neural Information Processing Systems*, pp. 1090–1098, 2010.

[20] S. S. Keerthi and E. G. Gilbert, "Convergence of a generalized SMO algorithm for SVM classifier design," *Machine Learning*, vol. 46, no. 1-3, pp. 351–360, 2002.

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

[22] K. Kosmidis and C. Kalloniatis, "Machine learning and images for malware detection and classification," in *Proceedings of the 21st Pan-Hellenic Conference on Informatics*, pp. 1–6, 2017.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[24] A. Liaw and M. Wiener, "Classification and regression by random forest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[25] A. Makandar and A. Patrot, "Malware class recognition using image processing techniques," in *International Conference on Data Management, Analytics and Innovation*, pp. 76–80, 2017.

[26] A. Marx, G. Habicht, and M. Morgenstern, *Malware Statistics and Trends Report*, the AV-TEST Institute, Apr. 2019. (`http://www.av-test.org/en/statistics/malware`)

[27] Q. K. A. Mirza, I. Awan, and M. Younas, "Cloud-intell: An intelligent malware detection system," *Future Generation Computer Systems*, vol. 86, pp. 1042–1053, 2018.

[28] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, pp. 4, 2011.

[29] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, "A comparative assessment of malware classification using binary texture analysis and dynamic analysis," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, pp. 21–30, 2011.

[30] X. X. Niu and C. Y. Suen, "A novel hybrid CNN-SVM classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, no. 4, pp. 1318–1325, 2012.

[31] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[32] Robinson and Cole, *Data Privacy and Cybersecurity for Tax Professionals*, IRS nationwide tax forum, 2019. (`https://www.irs.gov/pub/irs-utl/2019ntf-11.pdf`)

[33] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, *Microsoft Malware Classification Challenge (BIG 2015)*, 2018. (`https://www.kaggle.com/c/malware-classification`)

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[35] S. Seok and H. Kim, "Visualized malware classification based on convolutional neural network," *Journal of The Korea Institute of Information Security & Cryptology*, vol. 26, no. 1, pp. 197–208, 2016.

[36] M. Sewak, S. K. Sahay, and H. Rathore, "Comparison of deep learning and the classical machine learning algorithm for the malware detection," in *The 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD'18)*, pp. 293–296, 2018.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Computational and Biological Learning Society Conference at ICLR*, pp. 1–14, 2015.

[38] J. Su, V. D. Vasconcellos, S. Prasad, S. Daniele, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," in *IEEE the 42nd Annual Computer Software and Applications Conference*, vol. 2, pp. 664–669, 2018.

[39] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[40] Y. Tang, "Deep learning using linear support vector machines," in *Workshop on Challenges in Representation Learning ICML*, 2013. (`arXiv:1306.0239`)

[41] A. Tayal, N. Mishra, and S. Sharma, "Active monitoring and postmortem forensic analysis of network threats: A survey," *International Journal of Electronics and Information Engineering*, vol. 6, no. 1, pp. 49–59, 2017.

[42] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, "Malware detection with deep neural network using process behavior," in *IEEE the 40th Annual Computer Software and Applications Conference*, vol. 2, pp. 577–582, 2016.

[43] A. Torralba, K. P. Murphy, W. T. Freeman, M. A. Rubin, *et al.*, "Context-based vision system for place and object recognition," *International conference of Computer Vision*, vol. 3, pp. 153–167, 2003.

[44] V. Tra, S. Khan, and J. Kim, "Diagnosis of bearing defects under variable speed conditions using energy distribution maps of acoustic emission spectra and convolutional neural networks," *The Journal of the Acoustical Society of America*, vol. 144, no. 4, 2018.

[45] S. H. Tsang, *Review: Inception-v3 — 1st Runner Up (Image Classification) in ILSVRC 2015*, 2018. (`https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c`).

[46] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, ”Robust intelligent malware detection using deep learning,” *IEEE Access*, vol. 7, pp. 46717–46738, 2019.

[47] S. Yue, ”Imbalanced malware images classification: A CNN based approach,” *Computer Vision and Pattern Recognition*, 2017. (`arXiv:1708.08042`)

[48] M. Zareapoor, P. Shamsolmoali, and M. A. Alam, ”Establishing safe cloud: Ensuring data security and performance evaluation,” *International Journal of Electronics and Information Engineering*, vol. 1, no. 2, pp. 88–99, 2014.

[49] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. K. Sangaiah, ”Classification of ransomware families with machine learning based on N-gram of opcodes,” *Future Generation Computer Systems*, vol. 90, pp. 211–221, 2019.

Biography

Ahmed Bensaoud received the B.S. degree from the Benghazi University, Libya, and M.S. from Colorado State University, Fort Collins, Colorado. Currently he is a Ph.D. student at the University of Colorado Colorado Springs. His research interests include malware detection and malware classification.

Nawaf Abudawaood graduated from the University of Colorado at Colorado Springs with a Masters in Engineering in Information Assurance. He received his Bachelors degree from the Old Dominion University Norfolk, Virginia, in Information Systems and Technology. He currently works for The Exchange Hub as a Cyber Security Engineer.

Jugal Kalita received Ph.D. from the University of Pennsylvania, Philadelphia. He is a Professor of Computer Science at the University of Colorado, Colorado Springs. His research interests are in machine learning and natural language processing. He has published over 250 papers in international journals and referred conference proceedings and has written four books.