2023

Robustness of Image-Based Malware Classification Models Trained with Generative Adversarial Networks

Ciaran Reilly
Technological University Dublin, ciaran.reilly@tudublin.ie

Stephen O Shaughnessy
Technological University Dublin, stephen.oshaughnessy@tudublin.ie

Christina Thorpe
Technological University Dublin, christina.thorpe@tudublin.ie

Funder: This research received no external funding

Robustness of Image-Based Malware Classification Models trained with Generative Adversarial Networks

Ciaran Reilly
B00123609@mytudublin.ie
Technological University Dublin
Dublin, Ireland

Stephen O'Shaughnessy
stephen.oshaughnessy@tudublin.ie
Technological University Dublin
Dublin, Ireland

Christina Thorpe
christina.thorpe@tudublin.ie
Technological University Dublin
Dublin, Ireland

ABSTRACT

As malware continues to evolve, deep learning models are increasingly used for malware detection and classification, including image-based classification. However, adversarial attacks can be used to perturb images so as to evade detection by these models. This study investigates the effectiveness of training deep learning models with Generative Adversarial Network-generated data to improve their robustness against such attacks. Two image conversion methods, byteplot and space-filling curves, were used to represent the malware samples, and a ResNet-50 architecture was used to train models on the image datasets. The models were then tested against a projected gradient descent attack. It was found that without GAN-generated data, the models' prediction performance drastically decreased from 93-95% to 4.5% accuracy. However, the addition of adversarial images to the training data almost doubled the accuracy of the models. This study highlights the potential benefits of incorporating GAN-generated data in the training of deep learning models to improve their robustness against adversarial attacks.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**.

KEYWORDS

Malware Analysis, Adversarial AI, Cybersecurity, Incident Response, Generative Adversarial Networks

ACM Reference Format:
Ciaran Reilly, Stephen O'Shaughnessy, and Christina Thorpe. 2023. Robustness of Image-Based Malware Classification Models trained with Generative Adversarial Networks. In *European Interdisciplinary Cybersecurity Conference (EICC 2023), June 14–15, 2023, Stavanger, Norway*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3590777.3590792

1 INTRODUCTION

There exists a perpetual "arms race" between security analysts and adversarial malware developers as malevolent programs evolve and countermeasures are developed to detect and eradicate them. The ease with which malicious actors can produce variants of malware has seen significant increases in the amount of malware processed by antivirus companies, with some reporting receiving upwards of 350,000 samples per day [3]. As such, this amount of malware cannot feasibly be analysed manually, so analysts and AV companies are relying more and more on machine learning and artificial intelligence to expedite the analysis process.

Deep learning is advantageous in that it is not reliant on expert domain knowledge to identify salient features for classification. Instead, features are determined by the deep neural networks. This is a highly motivating factor for utilising deep learning in the malware classification space. Previous research efforts have demonstrated the versatility of applying deep learning to enhance the classification and detection of malware. For example, Raff et al. [22] used a convolutional neural network to extract malware features from the raw bytes of the Portable Executable file and achieved an accuracy of greater than 88%. AL-Hawawreh et al. [2] used a combination of a deep autoencoder and a feed-forward neural network to learn malicious activity from TCP/IP packets and achieved a detection rate greater than 92% depending on the dataset. Rezende et al. [23] took advantage of a pre-trained ResNet-50 by freezing the convolutional layers and modifying the last layer to malware classification. The malware samples were represented as grayscale images and accuracy of over 98% was achieved.

Generative Adversarial Networks (GANs) were first proposed by [10]. They are comprised of a discriminative model and a generative model which form a zero sum minimax two player game. The generator generates adversarial examples of the training data in order to trick the discriminator into determining the example is real data. Both models learn from independent back-propagation in order for the generator to produce better images and for the discriminator to better distinguish between real and synthetic data. GANs have allowed researchers to use the highly trained generators to produce superficially authentic data when compared to the real dataset. This allows researchers to train high-performance models even though the dataset available contains insufficient samples to train their neural network. Lu et al. [15] used this approach in generating more malware image samples to train their classification model and thus improve the performance. Their model had a higher accuracy compared to models trained without GANs.

Deep learning models are vulnerable to adversarial examples, which are slightly perturbed images resembling natural images but maliciously crafted to fool pre-trained models [21]. Attacks such as Projected Gradient Descent (PGD), Fast Gradient Sign Method (FGSM) and Jacobian-based Saliency Map Attack (JSMA) can be used to create perturbed images. Although deep learning models provide many advantages, these risks require careful consideration for use in critical environments such as self-driving cars or malware classification and detection.

EICC 2023, June 14–15, 2023, Stavanger, Norway

ACM ISBN 978-1-4503-9829-9/23/06.
https://doi.org/10.1145/3590777.3590792

Figure 1 is an example of an adversarial image created by FGSM. It is evident from the adversarial example how the two panda images look identical to the human eye but the deep learning model classified the image on the right as a gibbon. The same theory holds true for malware samples converted to images. Our problem definition is as follows: The use of deep learning has helped to advance the field of malware classification as expert domain knowledge is not always necessary. However, these models are vulnerable to adversarial attacks. Attackers can use these samples to circumvent systems built on deep learning models such that the sample is misclassified. While GANs have been used in the training of image-based malware classification models, only their effect on the performance of the model has been considered. This research aims to first determine which image conversion method produces the highest accuracy with and without the use of generative adversarial networks. Secondly, it sets out to determine the robustness of the image-based malware classification models trained with and without generative adversarial networks.

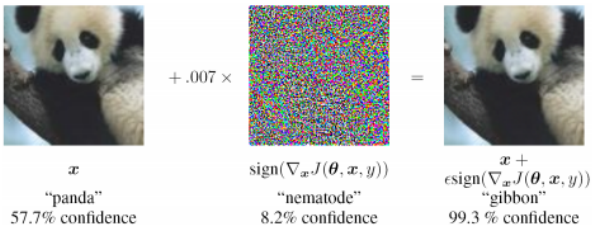

Figure 1: FGSM generated adversarial image [32]

To achieve these aims, the following questions will be examined:

(1) How does the method of image conversion affect the accuracy of the classification model?
(2) How robust are image-based malware classification models trained without GANs?
(3) How robust are image-based malware classification models trained with GANs?

The research objectives are

(1) To determine which method of image conversion produces the optimum classification model.
(2) To determine the robustness of image-based malware classification models trained both with and without GANs.

2 ATTACKER MODEL

There is an important difference between malware obfuscation and adversarial attacks in the context of malware detection. Malware obfuscation is a technique used by malware distributors to evade detection by antivirus software. It involves modifying the code of malware in a way that makes it more difficult for antivirus software to detect. This may involve changing the signature of the malware, adding extra code to the malware to confuse detection algorithms, or using encryption to make the malware harder to analyse.

On the other hand, adversarial attacks in image based malware detection involve creating small perturbations in images to fool a machine learning classifier. The goal is to make the classifier misclassify the image, even though the perturbations are not visible to the human eye. This is done by adding carefully crafted noise to the image to mislead the classifier.

While both techniques involve modifying the malware or image in some way, they are different in their goals and methods. Malware obfuscation is aimed at evading detection by antivirus software, while adversarial attacks aim to bypass image-based malware detection models.

Let A = (C, K, R, O, S) be an attacker model for an adversarial attack on an image classification system, where:

C = {c1, c2, ..., cm} represents the set of attacker capabilities, such as adversarial perturbations, image manipulation, image synthesis, model inversion, and black-box attacks.

K = {k1, k2, ..., kn} represents the set of attacker knowledge, such as knowledge of the architecture and parameters of the image classification system, knowledge of the training data distribution, and knowledge of the system's use case.

R = {r1, r2, ..., rp} represents the set of attacker resources, such as access to powerful GPUs, access to a large amount of training data, and access to specialised knowledge or tools.

O = {o1, o2, ..., oq} represents the set of attacker objectives, such as causing misclassification in the image classification system and evading detection by the image classification system.

S = {s1, s2, ..., st} represents the set of attacker scenarios, such as specific use cases or user profiles, that the attacker assumes when crafting adversarial examples. Adversarial attacks in this context are specialised targeted attacks that involve a lengthy reconnaissance and a sophisticated multi-phase exploit in order to succeed. They would be conducted by highly skilled and motivated attackers who are willing to take risks, spend a long time covertly gathering information, and invest lots of resources to infiltrate a high value target, e.g., state actors attempting to install malware to gather sensitive information or organised cybercriminal gangs motivated by significant financial gain.

Training a classifier on adversarial example images can potentially make it more robust against obfuscated malware, but this is not guaranteed. Adversarial examples are created by perturbing the original image in a way that is not perceptible to the human eye but can cause misclassification by a classifier. By training a classifier on adversarial examples, the classifier is exposed to different types of perturbations, which can potentially improve its ability to distinguish between different types of malware, including obfuscated malware. However, this is not a foolproof solution as attackers can also create new adversarial examples that are specifically designed to evade the classifier's defences.

3 RELATED WORK

This section provides a review of related literature and research associated with malware classification. The focal points of the review are central to our research, namely binary visualisation, image-based malware classification and related work on GANs.

3.1 Binary Data Visualization

The first binary-to-image conversion technique considered is the byteplot [6] and was used by [18] to convert malware to images. This is a byte-to-pixel mapping, where each malware binary is

read as a stream of 8-bit unsigned integers and organised in a 2-dimensional array. The width of the image is fixed and the height of the image is allowed to vary. The values in the 2-dimensional array were between 0 (black) and 255 (white) to produce a grayscale image. Fig. 2 shows a sample of the Trojan Dontovo in byteplot format. The .text section contains executable code, which is mapped to a distinct fine-grained texture. Following the code, the rest of the section is padded with zeros, which are represented by the black area in the image. The other sections within the file have distinct texture patterns. The byte-to-pixel mapping of the byteplot method means that similar binary code, such as that found in malware variants maps to similar texture regions.

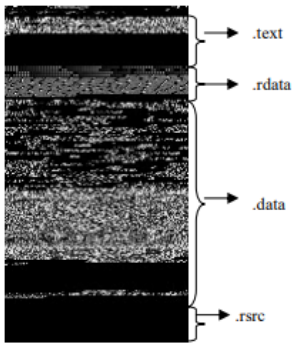

Figure 2: Byteplot representation of the Trojan downloader malware Dontovo.A [18]

The second binary-to-image conversion method considered was space-filling curves (SFCs). In mathematical analysis, SFCs are curves whose range contains the entire 2-dimensional unit square or more generally an n-dimensional unit hypercube, but for the purposes of this research only the 2-dimensional space was considered, since the outputs are 2-dimensional images. An SFC traces a continuous curve through every unit square, i.e., pixel, in the image, such that closely located points in the binary file space will also be closely located when mapped through the SFC. SFC techniques therefore have a practical application in image-based malware classification in that locality is preserved, i.e., similar regions in the binary code can be grouped together by clusters or regions of distinct colours or textures. The locality preservation property of SFCs means that similar malware binaries will share similar images, which can then be used to classify them according to their respective families. Fig. 3 shows three different variants from the Locky ransomware family, mapped to images through the Z-order SFC. It is evident that there is a high degree of similarity between the image texture patterns, indicating very similar binary source code.

3.2 Image-based Malware classification

The method of using image processing techniques for malware classification was first introduced by [18] where they converted malware samples to gray-scale images. The study achieved an accuracy of 98% on samples from the Anubis analysis system using the k-nearest neighbors algorithm. Ahmadi et al. [1] achieved an accuracy of 99.8% on the Microsoft Malware Classification Challenge

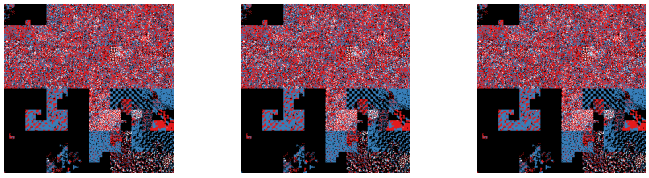

Figure 3: Variants of Locky ransomware in SFC format.

[24] data set using the XGBoost algorithm. Considerable domain knowledge is necessary and a complex feature extraction and fusion method needs to be engineered for this approach.

Further studies such as [23] and [15] converted the byteplot grayscale images further into RGB images. This was done by duplicating the grayscale channel to form three channels as RGB images consist of three channels. O'Shaughnessy [20] used various space filling curve approaches to classify malware according to their family classes. The Hilbert, Z-order and Gray-code variations of space filling curves were evaluated in the study. Using a dataset comprising 9,235 Windows 32-bit executable samples from 28 distinct families, the three conversion methods were evaluated using Local Binary Patterns (LBP), Gabor filters and Histogram of Gradients (HOG) for feature extraction and K-Nearest Neighbour (KNN), Random Forest (RF), and Decision Trees (DT) for supervised classification. The Hog and KNN method produced the best performance with the Z-order dataset, obtaining precision, recall and accuracy scores of 82%, 80% and 83% respectively. It should be noted that the Z-order SFC method was used to produce the image dataset used in this paper, due to its superior results over other SFC methods.

Deep learning can be used to extract image features and form a good representation of the malware sample with very little domain knowledge. Rezende et al. [23] took advantage of a pre-trained ResNet-50 which was trained on the ImageNet data set. Transfer learning was used and the last softmax layer was adapted for classification on the MalIMG dataset. The byteplot grayscale images were converted to RGB and scaled to 224×224. An accuracy of 98.62% was achieved. Gibert et al.[9] used a Convolutional Neural Network (CNN) on the MalIMG and Microsoft Malware Classification Challenge data sets to achieve accuracy's of 98.4% and 97.5% respectively. Lu et al. [15] used a Deep Convolutional Generative Adversarial Network (DCGAN) to generate synthetic malware samples for malware families with small data sets in order to improve the accuracy of their 18 layer deep residual network (ResNet-18). Vasan et al. [28] approach is an Image-based malware classification model using fine-tuned convolutional neural network architecture. The research compares the accuracy of its IMCFN algorithm against a pre-trained VGG-16 model, Googles Inception-v3 model and a ResNet-50 model. IMCFN achieves an accuracy of 98.82%.

3.3 Adversarial Attacks on DL Models

The concept of adversarial examples was first introduced by Szegedy et al. [26] where it was proved that the addition of an imperceptible non-random perturbation to an image could change a models prediction. The adversarial examples were produced using the L-BFGS method which is the Limited Memory Broyden Fletcher Goldfarb Shanno algorithm. The Fast Gradient Sign Method (FGSM) was

first introduced by [11] as a practical means to generate adversarial examples. In contrast to the L-BFGS linear search method the FGSM method updates one step along the sign of the gradient for each pixel. The study showed that training deep networks with adversarial examples improved the robustness of the model by decreasing the error rate from 89.4% to 17.9%. Goodfellow et al. [13] introduced the Basic Iterative Method (BIM) and the Iterative Least-Likely Class (ILLC) method to generate their adversarial examples as well as using the FGSM method. The study was conducted in the physical world with printed adversarial images fed to an Inception v3 image classification neural network through pictures taken on a mobile phone. It was found that the FGSM is more robust to photo transformation than the iterative methods. Projected Gradient Decent (PGD) is introduced by [16] as a first order adversary. That is the strongest attack using the first order information of the network. Deng et al. [8] recently proposed the Universal Projected Gradient Descent (UPGD) attack method which produces image agnostic universal perturbations. This method has shown good generalization across models such as VGG-16 and ResNet-50.

3.4 Generative Adversarial Networks as Defence

GANs have been added to the training of deep learning algorithms in order to achieve a more robust model. Goodfellow et al. [11] demonstrated that adversarial training can provide an additional regularisation benefit to deep neural networks which also provides for a more robust model.

Adversarial examples generalise well across models in that an adversarial example created on one model will be difficult to predict on another. This is the case even with different hyper-parameters between the models and a disjoint of the data set. These characteristics allow researchers to conclude that models trained with adversarial examples should be protected against examples generated by an attacker. Samangouei et al. [25] introduced Defense-GAN which is a framework for protecting classifiers against adversarial attacks using generative models. This has been proven to be effective against both black-box and white-box attacks on the MNIST [14] and F-MNIST [31] data sets.

While malware classification studies have used GANs to enlarge the data set of malware samples, none have studied the approach in relation to adding robustness to the model. The work in this paper will specifically investigate the potential for increasing the robustness of a model when trained on GAN-generated samples.

4 METHODOLOGY

In this section we discuss the methodology used to complete the objectives of the research. The research approach used in this study was quantitative and empirical. Two datasets were created for the application of this study and transfer learning was used to train deep learning algorithms for malware classification. Adversarial examples were generated using a DCGAN and PGD attacks were carried out on the models.

The research steps are as follows:

(1) An extensive literature review was conducted to better understand the research topic and formulate research questions.
(2) Two malware datasets were compiled from the VirusTotal (VT) academic collection 19-11-2020 edition, using samples in Portable Executable (PE) file format [29]. No such datasets existed as all previous works were based on an individual conversion method and thus have no correlation. One dataset was converted using the byteplot method and the other using the space filling curve method, as discussed in Sec. 4.2
(3) Two ResNet-50 models were trained on each method and fine-tuned to obtain the best accuracy. The models were trained on a combination of Google Colabs and Kaggle due to the GPU usage limitations. On the completion of the DCGAN phase, a further model were trained on the z-order dataset, which included the adversarial examples.
(4) The DCGAN phase involved the creation of adversarial examples by training a DCGAN on the z-order dataset. The examples produced were saved and later added to the dataset for the training of the final ResNet-50 model.
(5) The final phase of the research was to conduct the PGD attacks on the three models produced from the previous phases. Analysis was then conducted on the results to answer the research questions.

4.1 Data Preparation and Preprocessing

The VT data set consists of 14,068 malware samples comprising 305 distinct family classes of malware with one of the classes representing unidentified samples, referred to as Singletons.

The Singletons were removed from the data set and only classes with samples of fifty or more were considered for this research. This left a total of 11,554 samples across forty different malware families. They were: virlock (1494), sfone (1369), salgorea (991), vobfus (811), razy (802), hematite (601), berbew (574), sytro (524), zusy (385), allaple (351), stihat (320), ceeinject (256), sillywnse (220), xcnfe (181), tiggre (176), cuegoe (167), mira (151), wofith (146), wabot (140), wabot 140, picsys (127), drolnux (124), fsysna (110), dinwod (109), wapomi (103), sillyp2p (100), dorkbot (99), oberal (98), sivis (84), blackmoon (81), gify (80), zbot (78), smallagent (75), pykspa (69), mepaow (66), packedent (64), gamania (61), wacatac (56), auitinj (55), aohk (50).

The VirusTotal malware data is contained within one folder which is composed of the malware samples named by their sha256 hash and a .json file with their VirusTotal scan results. We separate these files into individual directories. To find the families of the malware we use AVClass [12] which is available on GitHub. The library helps to read from the .json files and extract the family name of the malware samples, which are written to a CSV file and then sorted using the sortMalware.py script. The script iterates through the CSV file moving the malware samples into their family named directories or creating the new directory if necessary.

4.2 Image Conversion Design

To convert the malware binaries to Z-order images, the GitHub Scurve [7] library was used with its implementation of Binvis. The library was chosen as it was used in the study on space filling curves by [20]. As it is an outdated library some import changes need to be made in order for it to compile. A bash script is used in order to facilitate an automated method of calling the library on the different malware sample directories. The script handles the

sample source and the save location for all files and folders in a given directory.

The byteplot conversion was handled by a python script formulated by the merging of two libraries from GitHub. Both the BinaryToImage [33] and Binary-Image Converter [17] were combined in order to facilitate the image width depending on the file size and running of the library as a script. Similar to the Z-order conversion, a bash script was used to handle the sample source and the save location for all files and folders in a given directory. Figure 4 shows the byteplot and Z-order conversion of the malware sample with the sha256 hash ending in ab1f72a4fb660afc10c2770f. The sample has been identified as the Allaple malware family.

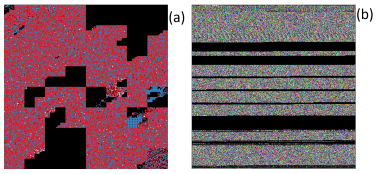

Figure 4: (a) Space Filling Curve z-order (b) Byteplot

4.3 ResNet-50 Architecture Strategy

The model chosen for the classification of the malware samples was the pre-trained ResNet-50. It has been proven to be effective for transfer learning by [23] who achieved an accuracy of 98.62% in malware classification. The model's last layer is a fully-connected (fc) layer of 1,000 nodes with softmax activation. In order to match the datasets created, this layer is changed to contain 40 nodes, one for each malware family. All previous layers to the last layer are frozen during training.

4.4 DCGAN Architecture Strategy

The study by [15] used a DCGAN to generate synthetic data in order to improve the accuracy of their ResNet-18 model by 6%. The study used an image size of 32×32.

As this study is using an image size of 64×64 pixels, an extra layer needs to be added to the convolutional network of both the generator and the discriminator. The convolutional networks will be five layers deep with the discriminator using the LeakyReLU activation function while the generator uses the ReLU function.

4.5 Projected Gradient Descent Attack

To carry out the PGD attack, the Advertorch [4] library was used. Advertorch is a Python toolbox for adversarial robustness training, containing modules for generating adversarial perturbations and defending against adversarial examples. For the purposes of this experiment the `advertorch.attacks.PGDAttack` function was applied. The function is an iterative PGD attack which takes in the trained model, maximum distortion value, number of iterations and the learning rate. The output was then fed images through the perturb method and produced adversarial images which could then be classified by the model to test its classification performance. The settings used were: 0.1 maximum distortion, 5 iterations, step size 0.0001.

4.6 Experimental Design and Limitations

The data processing of the malware samples is carried out on a VirtualBox [19] virtual machine running the Ubuntu 20.04 LTS ISO image [27]. As previously discussed in Section 4.2 Bash and Python scripts are also used within the virtual machine for the image conversion process. The training/testing of the deep learning models is carried out on both Google Colab and Kaggle using Python, PyTorch, Matplotlib and Numpy.

The main limitations of this research are due to the computational cost and processing power required for training deep learning models. Without the ability to use a locally available GPU, the experiments had to be carried out on the cloud-based resources, Google Colabs and Kaggle. While both provide free access to the machine learning field they can also be quite limiting. Google Colabs allows sessions of up to twelve hours only and also use captcha popups during the session. Kaggle proves to be the more reliable tool but its session times are limited to nine hours. A GPU can be used for up to a total of thirty six hours per week.

5 EXPERIMENTAL ANALYSIS

The research questions proposed in this study were answered through three distinct experiments. Experiments one and two tested the robustness of trained models to adversarial examples on the byteplot and space-filling curve image conversion techniques. Experiment three tested the robustness of the space-filling curve model trained on adversarial examples.

5.1 Model Performance Against PGD Attack

This section details the robustness testing of the Byteplot and Z-order models against an adversarial PGD attack. For each dataset, the data was separated into an 80/20 split with the training data being 9,243 samples and the validation data being 2,311 samples. The random seed was set to zero for repeatable results. Bayesian Optimisation was used to find the optimal batch size and learning rate. For the Z-order dataset, the batch size was set to 97 and the learning rate to a value of 0.00545; for the Byteplot dataset, the batch size was set to 58 and the learning rate to a value of 0.000731.

In each case, the pre-trained ResNet-50 was then trained for a total of 25 epochs. Once the pre-trained ResNet-50 models completed training of the final fully connected layer, the PGD attack was executed on the saved Byteplot and Z-order models. The validation training set was used for the creation and testing of the perturbed images.

5.2 Robustness of GAN-trained Model

Due to the resource constraints on Google Colab and Kaggle, it was not feasible within the time-frame to train a DCGAN sufficiently to learn the distribution of the byteplot dataset, thus it is not considered for robustness testing.

To create the adversarial examples, a DCGAN was trained for 1,000 epochs on the training data of the Z-order dataset. After 100 epochs, a batch of generated samples was saved every 500 iterations. The batch contained one sample per class and generated over 250 samples per class on completion. The last 250 samples were taken for each class giving a total of 10,000 new samples. The adversarial

examples were added to the training dataset by concatenation and the same process was followed as in 5.1.

6 RESULTS

This section presents the results of the experiments outlined in Section 5. A comparative analysis of the Byteplot and Z-order models results from malware classification with and without the adversarial PGD attack is presented. Next, the performance of the Z-order dataset trained with adversarial examples is discussed.

6.1 Evaluation Metrics

To compare the models in this study, the metrics Accuracy, Precision, Recall and F1-score were employed. As the class distribution was uneven, a weighted average was used for Precision, Recall and F1-score. Accuracy was used to determine the robustness of the model. A Confusion Matrix was also used to determine the performance of the models on a per-class basis. In order to compute these metrics, we used the following values obtained from the validation process: True positive (TP), which represents items of the target class are classified as the target class. True negative (TN), which represents items not of the target class are not classified as the target class. False-positive (FP), which represents items not of the target class are classified as the target class. False-negative (FN), which represents items of the target class are not classified as the target class. Accuracy measures how well a model can correctly classify the input data into the correct class labels and is defined as the ratio of the number of correctly classified examples to the total number of examples in the dataset [20].

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision is the ratio of correctly predicted positives of all the predicted positives. All predicted positives includes the true positives and false positives [20]. For malware classification precision means, for all the malware labelled as a particular family, how many were correct?

$$Precision = \frac{TP}{TP + FP}$$

Recall is the ratio of correctly predicted positives of a class with respect to all the samples of the class [20]. In this case, it is the ratio of the correctly predicted malware to the total number for that family, i.e., for each malware family, how many that should have been labelled as that family, were labelled correctly?

$$Recall = \frac{TP}{TP + FN}$$

F1-score is the harmonic mean between precision and recall and is used to check the balance in an uneven class distribution [15].

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

6.2 Comparative Analysis of Byteplot and Z-order Model Classification

As described in 5.1, the Byteplot and Z-order model's performances were first evaluated on unperturbed data and then against an adversarial PGD attack. Both model's classification performance metrics are shown in Table 1. From the table, both models performed comparatively well on unperturbed data, with the Byteplot achieving ~2% improvement over the Z-order model, with Precision, Recall, Accuracy and F1-score of 95.8%, 95.9%, 95.% and 95.7% respectively.

Model	Prec.	Recall	Acc.	F1	PGD Acc.
Z-order	93.1	93.8	93.1	93.0	4.54
Byteplot	95.8	95.9	95.8	95.7	4.59

Table 1: Performance of Z-order vs. Byteplot models (%)

Figure 5 shows the confusion matrix for the Z-order model on unperturbed data. It is evident from the matrix that there are misclassification errors with the mepaow, wofith, sfone and stihat families. It was observed in the mepaow and stihat families that the samples were obfuscated and had large amounts of null padding, which could have influenced the predictive capabilities of the model. In the case of wofith and sfone, several samples from each family were uploaded to VirusTotal and it was observed that wofith is used as an alias for sfone, hence the confusion between the two families. A sample of the VirusTotal detection outputs can be found at [30].

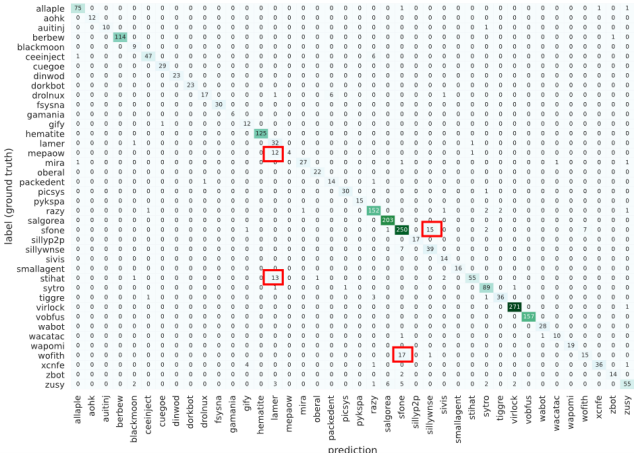

Figure 5: Vanilla Z-order Confusion Matrix

Figure 6 shows the confusion matrix for the Byteplot model. From the matrix, it is evident that there are fewer errors than that of the Z-order model. The only significant error is the wofith family being predicted as sfone, as explained previously.

The Z-order and Byteplot models were then subjected to an adversarial PGD attack. The accuracy results of the post-PGD attack for both models is shown in the last column of Table 1. It can be seen that the adversarial attacks drastically affected the performance of both models, reducing accuracy to ~4.5% in each case.

6.3 Robustness of GAN-trained Z-order Model

To test the robustness of the Z-order model trained with adversarial samples, the ResNet-50 model was trained on a dataset containing the Z-order unperturbed samples combined with the generated GAN samples. The same adversarial PGD attack as described previously was then performed on the resulting model. The model,

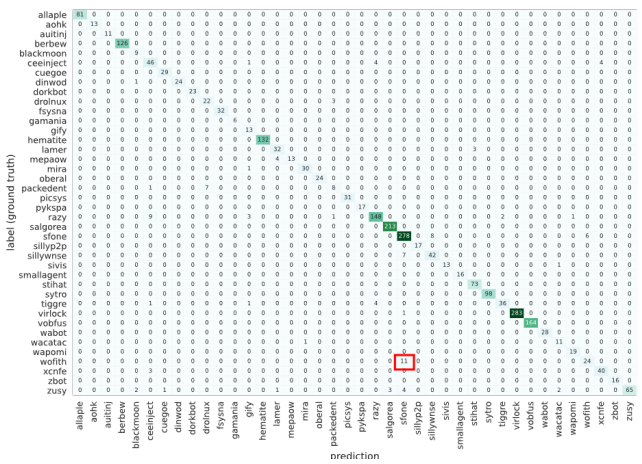

Figure 6: Vanilla Byteplot Confusion Matrix

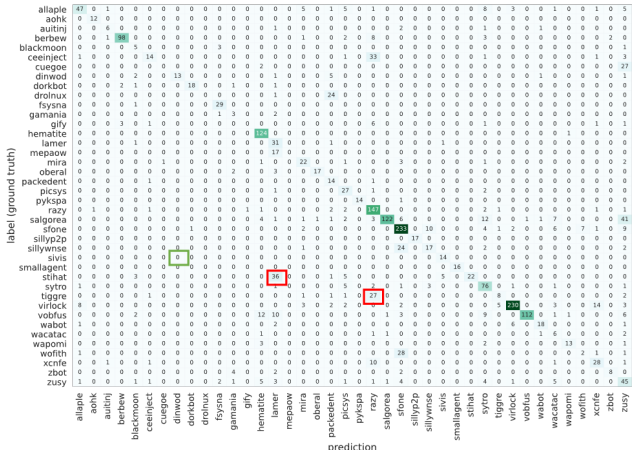

Figure 7: GAN Z-order Confusion Matrix

which we will refer to as GAN Z-order, achieved an accuracy of 70.45%, a precision of 74.79%, recall of 70.45% and an F1 score of 69.74%, as seen in Table 2.

Prec.	Recall	Acc.	F1-score	PGD Acc.
74.79	70.45	70.45	69.74	8.61

Table 2: GAN Z-order results in percentages

The confusion matrix for the GAN Z-order model's performance is shown in Figure 7. The addition of the adversarial samples had a significant negative effect on the model's predictive capabilities, which is evidenced from the resulting matrix. It can be seen that there are multiple prediction errors, which negatively affected the overall accuracy of the model, reducing it to 70.45%. The primary cause of the marked decline in overall classification accuracy can be attributed to the un-optimised method employed for generating adversarial examples using DCGAN. A reasonable assumption is that the out-of-the-box approach used to create adversarial examples was not optimal for certain families of malware, and resulted in lower-quality images that failed to retain the distinctive features of those families, leading to misclassifications. For instance, the Stihat and Tiggre families were misclassified as the Lamer and Razy families, respectively (13 Stihat predictions were misclassified as Lamer in the original data also). Further analysis of these families revealed that the adversarial examples created for the Stihat/Lamer and Tiggre/Razy families were significantly more similar to the other than the original images generated for these families. We also looked at a comparison between the Sivis and Dinwod families as there was zero mislassification from models trained without and with adversarial examples, and we can see that there is no significant difference between the similarity of the original data and the adversarial examples. We used Structural Similarity Index Measure (SSIM) to measure similarity [5], which is a score between -1 and 1 and can be interpreted as the higher the score the higher the similarity. The detailed findings of the SSIM comparisons using both RGB and YCbCr colour spaces are presented in Table 3.

	Average	Max	Min	Std
Stihat and Lamer SSIM Comparison (36 misclassifications)				
Org RGB	0.208553	0.764172	-0.454563	0.185787
Org YCbCr	0.345199	0.806296	-0.26673	0.204001
Adv RGB	0.293365	0.812677	-0.32338	0.204001
Adv YCbCr	0.551854	0.888554	-0.03792	0.149135
Tiggre and Razy Comparison (27 misclassifications)				
Org RGB	0.160677	0.947724	-0.07818	0.044445
Org YCbCr	0.298973	0.955656	0.123103	0.039511
Adv RGB	0.765138	1	0.193337	0.108254
Adv YCbCr	0.870646	1	0.47245	0.065808
Sivis and Dinwod Comparison (0 misclassifications)				
Org RGB	0.090456	0.543959	-0.20318	0.137572
Org YCbCr	0.24172	0.603095	-0.07483	0.117178
Adv RGB	0.066774	0.43929	-0.35923	0.109251
Adv YCbCr	0.390751	0.694679	-0.05338	0.083962

Table 3: SSIM Comparison of Images from Malware Families

Carrying out the PGD attack against the GAN Z-order model further reduced the accuracy of the model from 70.45% to 8.61%. Although the notion of enhancing the resilience of image-based classification models with adversarial AI displays immense promise, achieving this goal necessitates thorough investigation in the future.

7 ANALYSIS

This paper defined several research questions; the first was to determine which method of image conversion produces the most accurate model. From the results, it is evident the byteplot conversion technique produced a more accurate model with 95.76% versus 93.12% accuracy of the Z-order conversion. In comparison, the K-Nearest Neighbour Z-order model in [20] achieved an accuracy of 83%. However, the Z-order conversion method proved to be more efficient in terms of size and computational cost; the byteplot data set created was 13.6GB in size while the Z-order data set was 0.274GB in size for the same amount of samples.

The second question was to determine how robust are image-based malware classification models trained without GANs. For

both the byteplot and Z-order models, there is an evident lack of robustness against the attacks. Both models were reduced from 93-95% accuracy to ~4.5% accuracy showing a near-total collapse in the performance of the model.

The third question was to determine the robustness of image-based malware classification models trained with GANs. While the trained GAN Z-order model achieved a reduced accuracy of 70.45% on unperturbed data, it achieved greater accuracy than the vanilla models on the perturbed images. The accuracy of 8.61% is almost double that of the 4.59% and 4.54% achieved by the byteplot and Z-order vanilla models respectively. Despite the low overall accuracy achieved, the results indicate that the addition of adversarial examples improves the robustness of the model. Better adversarial examples may improve the accuracy and robustness further.

8 CONCLUSION

This research compared two methods, byteplot and space-filling curve conversion, for classifying malware images and evaluated their robustness when trained with and without GANs. Two datasets were created using both methods and models were trained using a pre-trained ResNet-50. The vanilla byteplot model achieved 95.76% accuracy and the vanilla Z-order achieved 93.12% accuracy, but both were susceptible to the PGD attack with an accuracy loss of 91.21 and 88.56 percentage points respectively. Training with adversarial images generated by a DCGAN improved the robustness and reduced the accuracy loss to 61.84 percentage points. The study suggests that including adversarial images in training can improve the robustness of malware classification models.

To build upon the results of this study, it would be valuable to investigate the robustness of a model trained with adversarial images generated from our byteplot dataset. Additional research could explore the use of different GAN architectures and their parameters to improve the adversarial examples and thus the accuracy of models trained on both z-order and byteplot. Additionally, the computational and space savings afforded by the space filling curve method used in this study suggests the need for further investigation into ways to enhance this process.

REFERENCES

[1] Mansour Ahmadi, Dmitry Ulyanov, Stanislav Semenov, Mikhail Trofimov, and Giorgio Giacinto. 2015. Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification. (/11/13 2015). https://arxiv.org/abs/1511.04317v2

[2] Muna AL-Hawawreh, Nour Moustafa, and Elena Sitnikova. 2018. Identification of malicious activities in industrial internet of things based on deep learning models. *Journal of information security and applications* 41 (Aug 2018), 1–11. https://doi.org/10.1016/j.jisa.2018.05.002

[3] AV-TEST. 2021. *(Malware Statistics & Trends Report.* https://www.av-test.org/en/statistics/malware/, (accessed: 14.03.2021).

[4] BorealisAI. 2018. Advertorch. (2018). https://advertorch.readthedocs.io/en/latest/index.html, (accessed: 17.03.2021).

[5] Dominique Brunet, Edward R Vrscay, and Zhou Wang. 2011. On the mathematical properties of the structural similarity index. *IEEE Transactions on Image Processing* 21, 4 (2011), 1488–1499.

[6] Gregory Conti, Erik Dean, Matthew Sinda, and Benjamin Sangster. 2008. Visual reverse engineering of binary and data files. In *International Workshop on Visualization for Computer Security*. Springer, 1–17.

[7] Aldo Cortesi. 2015. Scurve. (2015). https://github.com/cortesi/scurve, (accessed: 17.03.2021).

[8] Y. Deng and L. J. Karam. October 2020. Universal Adversarial Attack Via Enhanced Projected Gradient Descent. 1241–1245. https://doi.org/10.1109/ICIP40778.2020.9191288

[9] Daniel Gibert, 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 (2019), 15–28.

[10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. (/06/10 2014). https://arxiv.org/abs/1406.2661v1

[11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. (/12/20 2014). https://arxiv.org/abs/1412.6572v3

[12] Malicia Lab IMDEA Software Institute. 2020. AVClass. (October 20, 2020). https://github.com/malicialab/avclass, (accessed: 17.03.2021).

[13] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. (/07/08 2016). https://arxiv.org/abs/1607.02533v4

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (November 1998), 2278–2324. https://doi.org/10.1109/5.726791

[15] Y. Lu and J. Li. December 2019. Generative Adversarial Network for Improving Deep Learning Based Malware Classification. 584–593. https://doi.org/10.1109/WSC40007.2019.9004932

[16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. (/06/19 2017). https://arxiv.org/abs/1706.06083v4

[17] Jasper Miller-Waugh. 2020. binaryimageconverter. (2020). https://github.com/Fallstop/binaryimageconverter, (accessed: 17.03.2021).

[18] Lakshmanan Nataraj, Sreejith Karthikeyan, Gregoire Jacob, and Bangalore S. Manjunath. 2011. Malware images: visualization and automatic classification. In *Proceedings of the 8th international symposium on visualization for cyber security*. 1–7.

[19] Oracle. 2021. VirtualBox. (2021). https://www.virtualbox.org/wiki/Downloads, (accessed: 17.03.2021).

[20] S. O'Shaughnessy. October 2019. Image-based Malware Classification: A Space Filling Curve Approach. 1–10. https://doi.org/10.1109/VizSec48167.2019.9161583

[21] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. 2017. Generative Adversarial Perturbations. (/12/06 2017). https://arxiv.org/abs/1712.02328v3

[22] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles Nicholas. 2017. Malware Detection by Eating a Whole EXE. (/10/25 2017). https://arxiv.org/abs/1710.09435v1

[23] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos, and P. de Geus. December 2017. Malicious Software Classification Using Transfer Learning of ResNet-50 Deep Neural Network. 1011–1014. https://doi.org/10.1109/ICMLA.2017.00-19

[24] Royi Ronen, Marian Radu, Corina Feuerstein, Elad Yom-Tov, and Mansour Ahmadi. 2018. Microsoft Malware Classification Challenge. *CoRR* abs/1802.10135 (2018). arXiv:1802.10135 http://arxiv.org/abs/1802.10135

[25] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. 2018. Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. (/05/17 2018). https://arxiv.org/abs/1805.06605v2

[26] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. (/12/21 2013). https://arxiv.org/abs/1312.6199v4

[27] Ubuntu. 2021. Ubuntu. (2021). http://releases.ubuntu.com/20.04/, (accessed: 17.03.2021).

[28] Danish Vasan, Mamoun Alazab, Sobia Wassan, Hamad Naeem, Babak Safaei, and Qin Zheng. 2020. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks* 171 (April 22, 2020), 107138. https://doi.org/10.1016/j.comnet.2020.107138

[29] VirusTotal. 2020. VirusTotal. (October 20, 2020). https://www.virustotal.com/gui/, (accessed: 17.03.2021).

[30] VirusTotal. 2021. *(VirusTotal Report.* https://www.virustotal.com/gui/file/0bd4f793e0e9b196dc087489a4356927b2cefe2318779445e22cb5bfd41418ad, (accessed: 14.03.2021).

[31] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. (/08/25 2017). https://arxiv.org/abs/1708.07747v2

[32] X. Yuan, P. He, Q. Zhu, and X. Li. 2019. Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems* 30, 9 (Sep. 2019), 2805–2824. https://doi.org/10.1109/TNNLS.2018.2886017

[33] Necmettin Çarkacı. 2018. binary-to-image. (2018). https://github.com/ncarkaci/binary-to-image, (accessed: 17.03.2021).