Resilience against Adversarial Examples: Data-Augmentation Exploiting Generative Adversarial Networks

Mingu Kang[1], HyeungKyeom Kim[2], Suchul Lee[2*] and Seokmin Han[2]
[1] Korea Advanced Institute of Science and Technology
291 Daehak-ro, Yuseong-gu, Daejeon 34141, Republic of Korea
[e-mail: mingu.kang@kaist.ac.kr]
[2] Korea National University of Transportation
157, Cheoldobangmulgwan-ro, Uiwang-si, Gyeonggi-do 16106, Republic of Korea
[e-mail:{sclee, seokmin.han}@ut.ac.kr]
*Corresponding author: Suchul Lee

Received July 30, 2020; revised May 4, 2021; accepted September 24, 2021; published November 30, 2021

Abstract

Recently, malware classification based on Deep Neural Networks (DNN) has gained significant attention due to the rise in popularity of artificial intelligence (AI). DNN-based malware classifiers are a novel solution to combat never-before-seen malware families because this approach is able to classify malwares based on structural characteristics rather than requiring particular signatures like traditional malware classifiers. However, these DNN-based classifiers have been found to lack robustness against malwares that are carefully crafted to evade detection. These specially crafted pieces of malware are referred to as *adversarial examples*. We consider a clever adversary who has a thorough knowledge of DNN-based malware classifiers and will exploit it to generate a crafty malware to fool DNN-based classifiers. In this paper, we propose a DNN-based malware classifier that becomes resilient to these kinds of attacks by exploiting Generative Adversarial Network (GAN) based data augmentation. The experimental results show that the proposed scheme classifies malware, including AEs, with a false positive rate (FPR) of 3.0% and a balanced accuracy of 70.16%. These are respective 26.1% and 18.5% enhancements when compared to a traditional DNN-based classifier that does not exploit GAN.

Keywords: Malware classification, Microsoft Malware Classification Challenge (BIG2015), conditional GAN, Data augmentation, Adversarial Examples

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government. (MSIT) (2017R1C1B5017028, 2020R1F1A1063942).

http://doi.org/10.3837/tiis.2021.11.013 **ISSN : 1976-7277**

1. Introduction

Malware is defined as software designed to perform harmful actions on infected communication devices such as computers and mobile phones. With the rapid development of IT technology, malware detection techniques are constantly evolving, at the same time malware is becoming increasingly complex and sophisticated to evade detection from malware detectors such as vaccines, firewalls, and IPS/IDSes. In order to respond to the rapid changes in malware, several machine learning (ML) based malware classification methods have recently been introduced [1-3].

Deep Neural Networks (DNN), also known as deep learning, are an increasingly popular ML technique that have emerged as a viable alternative for malware classification. DNN-based malware classification is performed based on similarity among malware. For example, malware that belong to the same family often reuse the same libraries or functions, this gives them high similarities at the binary-level [2]. Inspired by these similarities, we convert malware binaries into gray scale images. This transforms the existing malware classification problem into an image classification problem. It is widely known that the performance of DNN-based image classifiers is generally excellent. Our experiments using the BIG2015 malware dataset also confirms this top-notch performance [4].

Despite the excellent performance of DNN-based classifiers, several issues remain when it comes to the actual use of those classifiers in practice. Consider an attacker who has prior knowledge about the training data or techniques used by a DNN-based classifier, e.g., the architecture of a DNN. Such an attacker may take tailored countermeasures to the DNN-based classifier [5]. For example, some studies have shown that DNN-based classifiers can be fooled by intentional perturbations [6-9]. Such perturbations typically include polymorphisms and/or obfuscation of malware. These are hardly imperceptible, and easily neutralize well-trained DNN-based malware classifiers/detectors. In general, malware variants crafted by adding such intentional perturbations, especially those for neutralizing ML-based classifiers, are referred to as Adversarial Examples (AEs). To successfully apply ML techniques to security applications, a thorough vetting of their resistance to AEs is required [10].

In this paper, we present a DNN-based malware classifier that is resilient to AEs. In order to generate such a classifier, we first assume a clever adversary who has a thorough understanding of our DNN-based classifier[1]. Against this adversary, we augment the training data by adding artificially crafted malware samples, i.e., AEs that are generated leveraging a Generative Adversarial Network (GAN) [11]. Here, the AE is a geometric transformation of an original image of the malware. Note that we only apply label-preserving linear transformations such as translation, rotation, scaling, and horizontal shearing, so as not to change the data distribution of the image where it can be determined by higher-level features [13]. Augmenting training data has recently been introduced as an effective countermeasure against such an adversary, this is referred to as adversarial training (AT) [6, 8].

The rationale behind the use of a GAN as a data augmentation method is as follows. The best strategy for a malware author is to perform small re-writes or to add minimal extra malware code to a sufficient degree to deceive the target malware detector while preserving the functionality of the original malware. This implies that an AE image should contain enough geometrical perturbations to fool the target DNN-based classifier. On the other hand, in order

[1] Here, this DNN-based classifier refers to the malware classification scheme presented in [4]. Therefore, in this paper, we mainly discuss on how to improve the scheme in [4] so as to be resilient against AEs.

to detect an AE, it is necessary to predict possible perturbations and incorporate such predicted perturbations into the DNN-model in advance. This extends the distribution of training data to capture the features of malware variants including AEs. Through this extension, we hypothesize that our DNN-based model will be capable of identifying a variety of malware variants including AEs [14].

Contributions: The main contributions of this paper are the following.

- We present a novel DNN-based malware classifier that is capable of detecting a variety of malware variants including AEs. We designed a new DNN architecture to fully learn the features of malware variants, including AEs crafted by exploiting GAN.
- Further, we examined various features that could be an impediment to a DNN-based classifier by extensive experiments. These newly discovered features are proven to enhance the resilience against AEs and are incorporated into our scheme. This examination opens a new research direction on how to deal with the well-known weak point of DNN-based classifiers, i.e., AEs.
- We present experimental evidence[2] for the superior performance of the proposed scheme. Our experiment was conducted using Microsoft Malware Classification Challenge (BIG 2015). The experimental results show that the proposed scheme classifies malware, including AEs, with a false positive rate (FPR) of 3.0% and a balanced accuracy of 70.16%. These are respective 26.1% and 18.5% enhancements when compared to a traditional DNN-based classifier that does not exploit GAN.

Organization: This paper proceeds as follows. Section 2 reviews the related work. Section 3 summarizes our previous DNN-based malware classification scheme. Section 4 briefly explains GAN and provides the details of how to augment training data for the DNN. Then, we describe how the augmented training data are incorporated into the proposed scheme. Section 5 evaluates the scheme. Finally, section 6 concludes the paper.

2. Related Work

2.1 Malware Visualization and Classification

Malware visualization: Malware analysis is a time-consuming and cumbersome task because there is no perfect approach and/or specified way to address it. So far various approaches have been studied, we have introduced several malware classification schemes based on image processing [1-2, 5, 16-17]. [2, 5] introduced the idea of representing malicious software binaries as gray scale images. Different features are extracted from each malware image sample using GIST [5]. The K-nearest neighbor algorithm is used for classification. We have also focused on enhancing existing image-based classifiers so they can detect malware crafted with the purpose of evading those classifiers.

Adversarial examples and countermeasures: When classification is performed in the image domain, it has been found that perturbations can cause increasing the misclassification rate even though only a few pixels on the image were changed, these changes are hardly visually discernible [6-9]. [18] proposed a new method of generating AEs by a GAN utilizing the distribution of the original instance. Studies that have generated AEs [10, 20-23] assumed several constraints for modifying malware binaries while preserving their functionality. These

[2] Interested readers may find the source code at https://sites.google.com/site/daxttt/resilience-to-aes-gans and can re-produce the experimental results

restrictions limit the range and amount of the perturbations introduced. For example, [22] generated AEs by injecting a few bytes into areas that are not frequently used amongst the malware files. [20] fixed only the manifest file of a piece of malware. [10, 21] modified only a few bytes at the end of each malware sample for a gradient-based attack. [23] presented MalGAN that can generate AEs using a GAN. On the other side, defense techniques against AEs have also been studied in response to the development of these AE generation technologies. One promising method is to elaborate on the training data to train a more powerful classifier [6, 8]. [19] modified the training process of a DNN so that the learned classification model is less likely to be affected by AEs. [8] can be regarded as the most similar to our research in that the research goals are similar. However, our paper differs from [8] in that our GAN is integrated into the DNN-based classifier in a way that effectively introduces a new DNN architecture that is resistant to AEs.

2.2 Data Augmentation Based on GAN

Since a GAN was first proposed in [11], they have been applied several times in literature [1, 13, 14, 23, 24] and have been extended in various ways [25-27]. Recently, GANs have frequently been used as an effective way to generate additional artificial data to enlarge the dataset. This is a common use case for data augmentation. It is appropriate for addressing a class imbalance problem [24] and/or a lack of training data. Using malware samples generated by a GAN, a DNN can significantly reduce overfitting, thus this deep learning approach not only improves accuracy but also scalability [29]. For malware classes with relatively few samples in the dataset, malware samples have been artificially generated using a GAN [24]. [14] showed that a GAN can reduce the effect of adversarial perturbations, by projecting input images onto the range of the GAN's generator prior to feeding them to the classifier. A GAN's generation process does not depend on specific classes, so in theory, a GAN can be applied to identify unknown data classes [10].

3. DNN-based Malware Classification

In this section, we summarize the malware visualization and DNN-based classification scheme that we proposed previously in [4]. Several malware visualization schemes exist in the literature, we applied the scheme introduced in [2]. This converts malware classification into image classification.

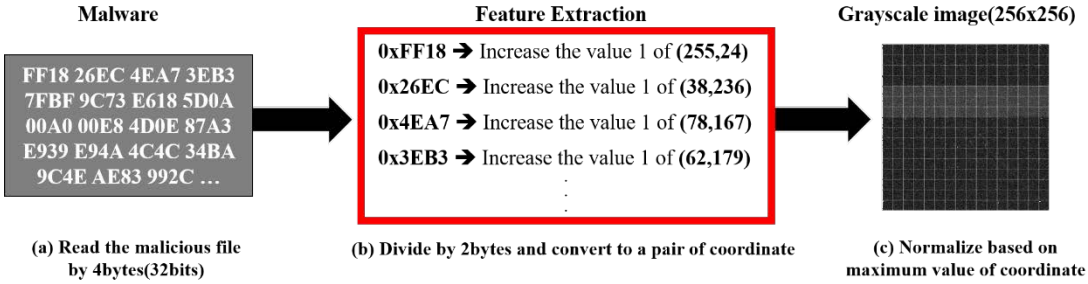

Fig. 1. Malware visualization process in the IC algorithm.

To apply DNN-based malware classification, each malware sample should be converted into a corresponding image of fixed size. **Fig. 1** illustrates how malware files are converted into Incremental Coordinate (IC) [2] image files. First, the malware binary is read in 4-byte

increments. The first 2 bytes are converted to x coordinates, the last 2 bytes are converted to y coordinates, the value represented by those coordinates is incremented by 1. This process is performed until the end of the file is reached. This generates a 256 by 256 gray scale image from each malware sample. The key advantage of visualizing a malicious PE file

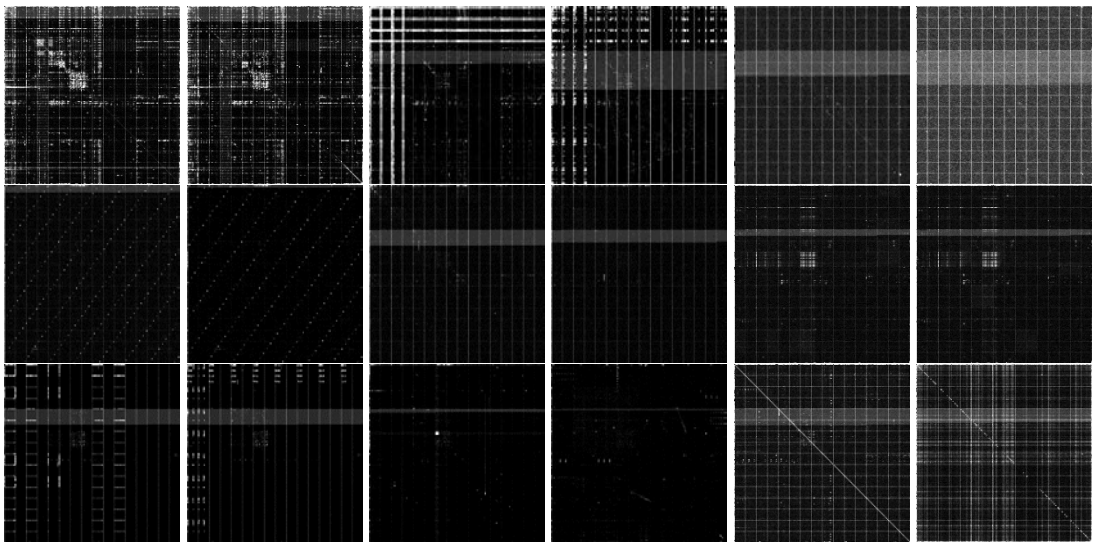

Fig. 2. Examples of malware images generated using the IC algorithm.

is that it can perform malware detection/classification without running malware. In addition, small changes that were not detectable in malware binaries may be detectable in malware converted to images [16].

In **Fig. 2**, we show two samples per class for classes 1 to 9. It can be easily seen that the images of malware belonging to the same class are quite similar. This can be explained by the practice of reusing existing malware code in the production of a malware variant [5]. Therefore, the main binary signatures of malware from the same malware family are similar or even identical to each other. We trained a DNN-based classifier using images generated through an IC algorithm as training data. The DNN architecture used was Google Inception V3, a.k.a., GoogLeNet [9], this algorithm is known to provide decent performance in various fields. The result of a performance evaluation using the BIG 2015 data set was our DNN-based malware classifier had an overall accuracy of 97.4%.

4. Malware Classification Resilient against AEs

In this section, we extend our previous work [4] to gain resilience against AEs. We begin with briefly summarizing GANs and the enhanced version we use, a conditional GAN. Then, we provide our assumptions regarding AEs and describe how we craft adversarial examples against legitimate malware based on these assumptions.

4.1 Generative Adversarial Network

A GAN [11] consists of two deep convolutional networks: the discriminative network and the generative network. The generative network $\mathbf{G}$ produces data from a latent noise vector $\mathbf{z}$ that follows a Gaussian distribution ($\mathbf{G}(\mathbf{z})$ = fake images). The discriminative network $\mathbf{D}$ can learn

feature distributions from training data and synthesized data [11] and it outputs a probability distribution over possible image source [26]. During the adversarial training, data flows in batches through $\mathbf{G}$ and $\mathbf{D}$ and their weights are fine tuned to optimize their loss functions $\mathbf{L}$ [24]. The whole adversarial training process can be described as follows:

$$\min_{\mathbf{G}} \max_{\mathbf{D}} \mathbf{L}(\mathbf{D}, \mathbf{G}) = \mathbb{E}_{x \sim p_{data}}[\log(\mathbf{D}(x))] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - \mathbf{D}(\mathbf{G}(\mathbf{z})))]. \quad (1)$$

A traditional GAN cannot control the generated data, but it can control the generated data through conditioning [25]. Specifically, a GAN can generate a fake image without a label. To apply to multiclass classification, the GAN needs to be extended to a conditional version. Thus, we use a conditional GAN, which extends a traditional GAN so as to feed extra information to our DNN, i.e., labels. Note that the extra information $\mathbf{Y}$ could be any kind of auxiliary information. Now, a loss function for our new conditional GAN can be re-written as follows:

$$\min_{\mathbf{G}} \max_{\mathbf{D}} \mathbf{L}(\mathbf{D}, \mathbf{G}) = \mathbb{E}_{x,y \sim p_{data}(x,y)}[\log(\mathbf{D}(x, y))] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z}), \mathbf{y} \sim p_{\mathbf{z}}(\mathbf{y}),}[\log(1 - \mathbf{D}(\mathbf{G}(\mathbf{z}, \mathbf{y})))]. \quad (2)$$

4.2 Crafting Adversarial Examples

Here we describe how we synthesize AEs. AEs can be considered as malicious manipulations of previous malware to avoid detection from malware detectors. Note that the DNN-based classifier we previously proposed used the IC algorithm for malware visualization. Since the IC algorithm captures the structural features in malware, it can be vulnerable to an attacker who attempts to manipulate those features. Injecting junk code can be done without compromising the functionality of the original program. For example, an attacker could create AEs by adding a large amount of duplicate data or by adding some hostile bytes. Attacks of this kind have been addressed in [22, 31].

In this paper, we consider a similar attacker strategy that injects or replaces some random bytes. When malware is converted to an image file using the IC algorithm, any bytes inserted or replaced are converted into discrete values, which can be considered as additional noise newly introduced. Surprisingly, although the generated malware variants are not much different from the original malware at the binary as well as the image level, DNN-based classifiers fail to classify many of those malware variants. Now, we assume that AEs are malware images that contain a small perturbation δ, i.e., noise, that is sufficient to fool a malware detector while retaining its structural features [5].

The basis of an adversarial attack strategy is to approach a specific optimal point using gradient-based optimization [6-9]. Its main goal is to achieve a high misclassification rate while adding only minimal perturbations to the AEs. The crafted AEs can be expressed by the following formula:

$$x_{AE} \triangleq x_{clean} + \mathbf{argmin}_{\delta} \mathbb{E}_{\delta \in [-\epsilon, \epsilon]} \{\delta : F(x_{clean} + \delta) \neq F(x_{clean})\}. \quad (3)$$

Generation of AEs: When generating AEs obtaining an optimal solution through a gradient-based method requires complete knowledge of the DNN architecture, training parameters, and data, so this type of attack is infeasible in practice. In general, we can assume that an attacker knows the training algorithms and data in advance. Despite the fact that the attacker has extensive knowledge of DNN-based classifiers, finding a specific perturbation value δ that satisfies **Eq. 3** is a difficult task.

In addition, an attacker must consider two requirements at the same time: (i) to maintain core functionalities of the malware, and (ii) to evade malware detectors effectively. Manipulating a PE file with these goals is, in general, a non-trivial task because malware can lose its key

function even if just a single byte is changed [21]. Thus, we apply the following when crafting AEs:

- We use additive Gaussian white noise (AGWN) to craft AEs. AGWN noise is represented by some discrete values in the resulting malware image and these can be generated by injecting junk code into the malware. Injecting some duplicate data or adding some hostile bytes can be done without compromising the functionality of the malware.
- We limit the amount of perturbation. Specifically, we limit the maximum value of the standard deviation of the AGWN to 0.001. According to our experiments, if the standard deviation value exceeds 0.001, the generated AEs are totally unpredictable. Therefore, they can no longer be considered relevant to the original malware class.

We use Multiscale-SSIM (MS-SSIM) to quantify the similarity between the original image and the fake image. MS-SSIM is based on the assumptions that human visual perception when extracting structural information from images performs well in principle. Therefore, a structural similarity measurement can be used as an approximate measure of similarity between images [32]. The range of MS-SSIM values is between 0.0 and 1.0. The higher the MS-SSIM, the more similar the image.

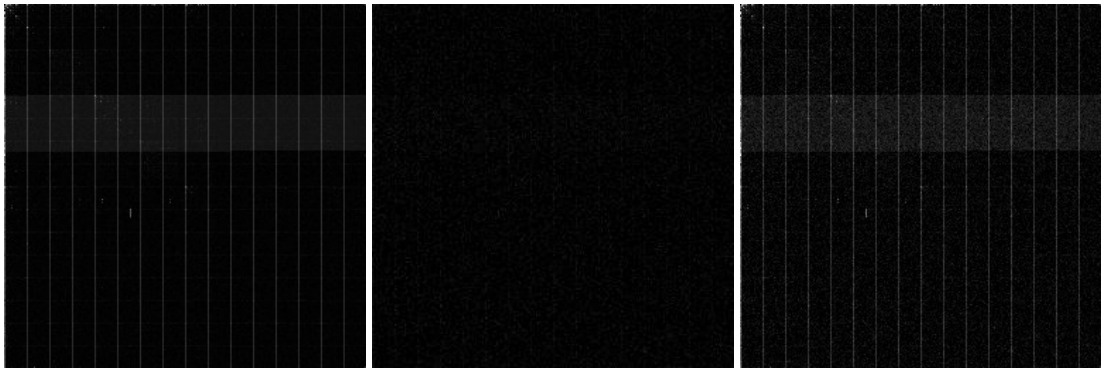

Fig. 3. The image on the left is the original malware image. The center is AGWN with a mean of 0 and a stddev of 0.001. The image on the right is an AE created by combining the two images on the left.

We compare the generated AE images with the original malware images in terms of MS-SSIM. The smaller the perturbation, the harder it is to discern the difference. Therefore, a well-written AE has a high MS-SSIM score but has an enough perturbation to deceive the classifier. **Fig. 3** shows some AE image examples that have MS-SSIM > 0.95. These images are visually and structurally similar to the original images. The original image was correctly classified with a confidence value of 0.99. However, despite the high MS-SSIM score, AE images are misclassified with a confidence value of 0.90.

Fig. 4 shows the MS-SSIM distribution of each malware class. We first randomly choose a pair of images per malware family from the set of original images, without duplication, and measure the MS-SSIM for each pair. This random choice is conducted 10 times in each of the 9 different classes, thus a total of 90 pairs are chosen. Next, we randomly choose two images, one from the set of AE images the other from the set of original images, per each malware family, we then measure the MS-SSIM for the selected pair. This also is conducted 10 times. As shown in **Fig. 4**, the value of the MS-SSIM between the original malware image and the AE image is not significantly different from the value of the MS-SSIM between the original

malware images. This means that there are only small visual or structural differences between the original malware and the AEs.

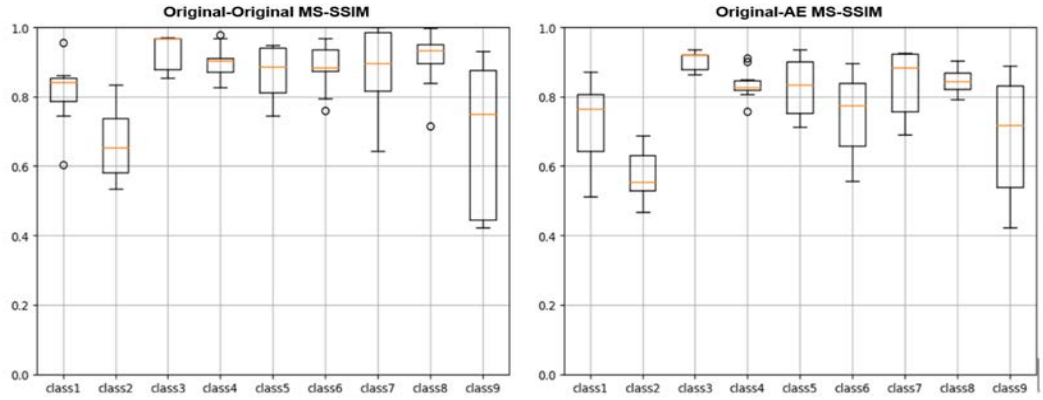

Fig. 4. The image on the left is the distribution of MS-SSIM for each malware class, and the image on the right is the distribution of MS-SSIM between the original image and the AE image for each malware class.

Quantification of the impact of perturbation: To quantify the impact of the perturbations, we measure balanced accuracy and misclassification rate. Balanced accuracy is the average of the proportion corrects for each class individually and the misclassification rate is the percentage of malware samples that were classified correctly without any noise injection, but were then misclassified after noise injection. Due to the use of AGWN, greater deviations mean that more noise is injected or a more demanding AEs is created. **Fig. 5** shows the performance of the DNN-based malware classification scheme in [4] for malware datasets including AEs. We observe that the performance decreases significantly as the standard deviation of the AGWN increases.

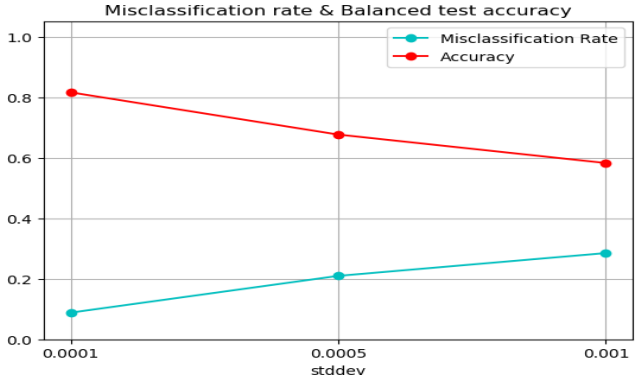

Fig. 5. Misclassification rate and balance accuracy according to the degree of the standard deviation of AGWN for DNN-based classifier presented in [4]

4.3 Resilient Malware Classifier Exploiting Data Augmentation

The primary goal of this research is to build a DNN-based malware classifier that has robustness against AEs. **Eq. 4** describes a modified process to train the DNN-based malware classifier using data augmentation, where C is the class label and N is the number of added

samples per class. Here, data augmentation means that the AEs generated by the conditional GAN's generative network **G** and the original malware images are used together as training data for a new DNN-based classifier. $\mathbb{P}^*_{data}$ is the distribution of extended training data. Now, we need to minimize the loss and find the optimal parameters.

$$\mathbb{P}^*_{data} = \left([x + \sum_{c=1}^{C} \sum_{n=1}^{N} \mathbb{E}_{x \sim p_z(z)}\{G(z_c|y_n)\}], y\right) \sim \mathbb{P},$$
$$\boldsymbol{\theta}^* = \underset{\theta}{\mathbf{argmin}}\, \mathbb{E}_{(x,y) \in \mathbb{P}^*_{data}} \boldsymbol{L}(\theta, x, y). \quad (4)$$

This extension smoothes the data distribution so that a classifier cannot be sensitive to small perterbations. As the number of training data increases, as long as the data is informative a classifier can continue exploring the loss function landscape. Also, this may prevent a classifier from local optimization. Therefore, we conjecture that GAN-based augmentation can complement the classifier's brittleness to AEs.

To generate such AEs, we need an extensively pre-trained generative model **G**. We use label $\boldsymbol{y}$ for the additional information from the conditional GAN. Then, **Y** is fed into the pre-trained generator **G** with latent vector **Z**. Generated AEs $G(\mathbf{X}|\mathbf{Y}), x_n \in \mathbf{X}, y_n \in \mathbf{Y}$ are combined with the original training dataset **X**, producing augmented training data **R**. Using this training data set, the DNN-based classifier is retrained to be robust to the AEs $G(\mathbf{X}|\mathbf{Y})$. When it comes to the particular DNN architecture, the Google Inception V3 (GoogLeNet) model was chosen because it performed best in our extensive experiments.

Experiments: We describe the experimental parameter values. 75% (8,245 samples) of the entire dataset are randomly selected as the training set, 15% (1,624 samples) are used as the validation set, and the remaining 10% (1,089 samples) are used as the test set. AEs generated from the generative network **G** are incorporated into the training set. For the test set, the number of samples in malware class 5 (simda) is particularly small (only four samples). Therefore, for class 5, ten samples are used as an exception.

In addition, the BIG2015 dataset used in the experiment has a class imbalance issue that could compromise the GAN's training process. If the generator **G** is not optimally trained due to the unbalanced dataset, it often focuses on generating images that belong to classes with a large number of samples. This is a natural consequence of the definition of the loss function [24]. If we use too much training data, the GAN faces computational complexity issues. On the other hand, the quality of AEs is not guaranteed making the construction of the data set a delicate

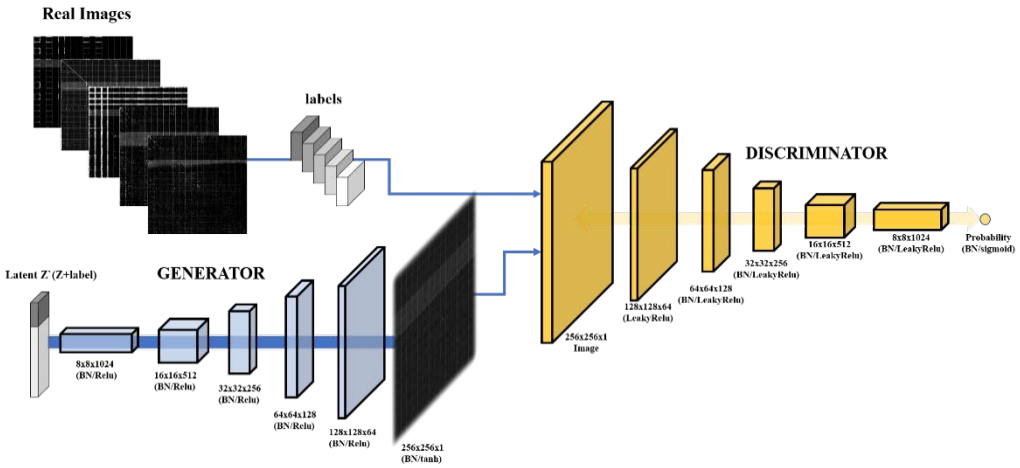

Fig. 6. The architecture of our proposed conditional DCGAN model.

trade-off. We randomly select 32 images for each malware class within the training set. The reason for selecting 32 images for each class is that in the case of class 5 with the smallest number of samples, only 32 samples can be used. A GAN generally requires a sufficiently large dataset to conduct reliable training. To overcome the limitations of the BIG2015 dataset, however, we experimentally improve the performance of the GAN by elaborating on the architecture of the DNN and fine-tuning the parameters. Our Conditional Deep Convolutional GAN (CDCGAN) architecture is illustrated in **Fig. 6**.

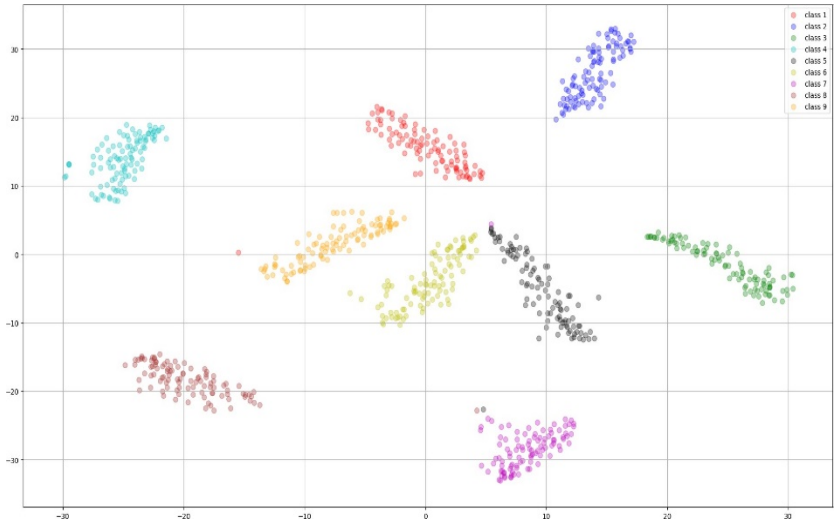

Fig. 7. Visualization by the t-SNE algorithm with 100 AE images per malware family

After the CDCGAN model is fully trained, it generates $n(n_1 = 10,\ n_2 = 20, n_3 = 100)$ AEs for each malware family. Combined with the training data, the generated AEs are used to rebuild a new DNN based classifier that is resilient against AEs. We chose the Stochastic Gradient Descent (SGD) optimizer for training a classifier. We used the SGD optimizer with a learning rate of 0.01 and a batch size of 100. We visualize the AEs using a t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm to ensure that the generator fully learns the characteristics of each malware class [28]. Here, the t-SNE algorithm visualizes the distribution of data by reducing the dimensions of the data and shows whether clusters form properly or not. As revealed in **Fig. 7**, the generator has been trained and disentangled the features of each class properly.

How we solve the overfitting problem: Since the number of training samples is not large, the discriminator **D** of the GAN can become overfitted. If the loss of **D** converges to a sufficiently small value and the loss of **G** converges to a sufficiently high value, further training is

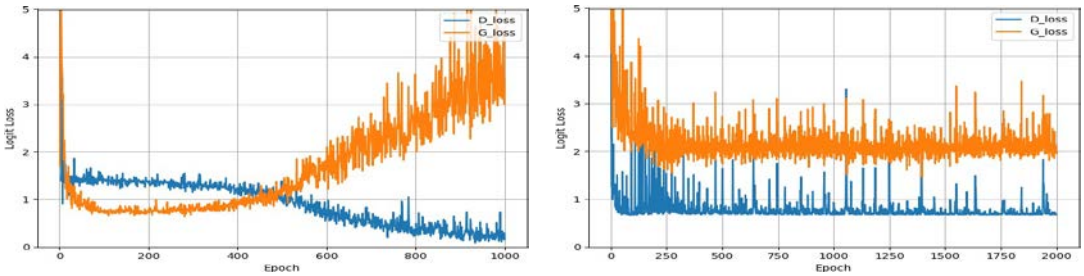

Fig. 8. The figure on the left shows loss in a situation where overfitting occurs while training the GAN, the figure on the right shows loss in a situation where GAN training progresses properly.

meaningless and the generator **G** can no longer be improved. That is, the discriminator **D** can clearly distinguish AEs. However, when this happens the generator **G** performs poorly. This is why for several epochs in **Fig. 8** **D** reaches a loss of 0.1. This means that AEs generated by **G** in this situation are not suitable for use as training data. To avoid this problem, we apply the DCGAN architecture and optimize **G** twice after **D** was optimized. Also, we tried to find the best optimizer for our GAN model and found that the Adam optimizer performed better than the SGD and RMSprop optimizers. Thus, we use the Adam optimizer to optimize **D** and **G** with a learning rate of 10^{-4} and 10^{-5}, respectively. The batch size was set to 96 and the model was trained over 2000 epochs.

5. Performance Evaluation

5.1 Experimental Setup

Dataset: We use a publicly available dataset from the Microsoft Malware Classification Challenge (BIG 2015) provided by Kaggle. The distribution of the dataset is shown in **Table 1**. Each malware sample file contains raw data consisting of a hexadecimal representation of the malware contents without the header. The dataset has a ground truth, i.e., each sample is labelled with one of the nine malware families, this is also provided by Kaggle. Interested readers may find more information about the data in [33].

Table 1. Applications in each class

Family name	Number of training samples	Type
Ramnit	1,541	Worm
Lolipop	2,478	Adware
Kelihos ver3	2,942	Backdoor
Vundo	475	Trojan
Simda	42	Backdoor
Tracur	751	Trojan Downloader
Kelihos ver1	398	Backdoor
Obfuscator.ACY	1,228	Any kind of obfuscated malware
Gatak	1,013	Backdoor
Sum	10,868	-

Performance Metrics: To examine the impact of AEs on the original classifier, we use the false positive rate (FPR) and true positive rate (TPR) [17] as performance metrics. We also consider the balanced accuracy to measure the effect of the proposed method where the balanced accuracy is defined as the average accuracy obtained by either class [34]. Note that the overall accuracy should not be used in our evaluation because it focuses on only the correct classification rate while the dataset used in our experiment has a significant imbalance in malware classes, as shown in **Table 1**. For example, a classification model that predicts class 2 (Lolipop) or class 3 (Kelihos) samples well, would be judged unfairly as superior to a model that predicts a class 5 (Simda) sample well due to the imbalance. This may grant an unnecessary advantage to a model that is specialized to classify a certain malware family which is well represented in the data set [16]. Thus, the overall accuracy cannot be free from bias in this experiment.

In addition, we use the Receive Operating Characteristic (ROC) curve to reveal the underlying characteristics of our proposed scheme, with many different levels of threshold. Typically, the ROC curve is used to show the discrimination ability of the classifier at various threshold settings. The curve is drawn with the true positive rate (TPR) against the false positive rate (FPR). The Area Under the Curve (AUC) is a metric which attempts to summarize the ROC curve to evaluate the quality of a classifier. The closer the AUC is to 1, the better the classifier performs [35].

5.2 Results

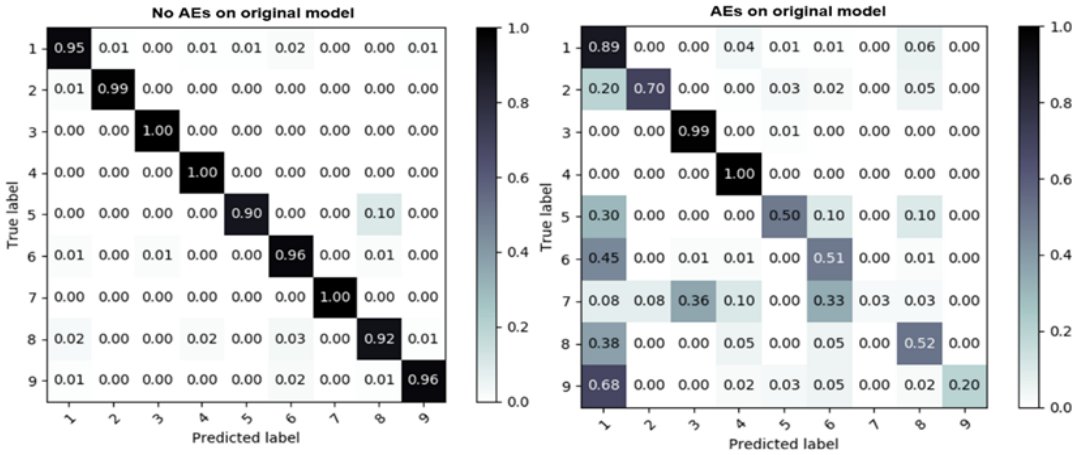

Fig. 9. Impact of AEs. Class 3 and 4 are not affected by small perturbations, but the other classes are highly affected, especially class 7 which was perfect before.

The impact of AEs on DNN-based classifiers: Before evaluating the performance of our DNN-based classifier, we show how the classification performance of a previously proposed scheme [1] was degraded while classifying a dataset including AEs. In **Fig. 9**, the figure on the left presents the classification results of our original malware classifier. When AEs are included, the performance is significantly degraded, as shown in the figure on the right of **Fig. 9**. The test accuracy and the balanced test accuracy are decreased from 97.4% and 96.4% to 71.1% and 59.2%, respectively. Additionally, the misclassification rate rises to 28.46%. The results show that a DNN-based malware classifier generally performs well, but even small perturbations can significantly degrade its performance despite the fact that AEs are structurally similar with clean images (See **Fig. 4**). This implies that DNN-based malware detectors are vulnerable to AEs. We now present the performance of our DNN-based malware classifier that leverages the generative power of a GAN and investigate the effectiveness of GANs against AEs.

Mitigating performance degradation by augmenting training data: To evaluate the influence of data augmentation, we investigate the balanced test accuracy and FPR with varying numbers of AEs. For a fair comparison, we use the same DNN-architecture and hyper parameters. **Fig. 10** and **Table 2** show the results of the proposed DNN-based malware classifier with the data augmentation leveraging a GAN. The test accuracy, the balanced accuracy, and the average FPR are 75.11%, 70.16%, and 3.0%, respectively. This is the best result we obtained, it was achieved with a test set containing 20 AEs from each class.

Table 2. The impact of AEs.

Metrics/Test	Baseline	n_1=10	n_2=20	n_3=100
Original - Accuracy(balanced)	97.4% (96.4%)	97.4% (95.4%)	97.2% (95.2%)	97.5% (95.4%)
AEs - Accuracy(balanced)	71.1% (59.2%)	74.56% (66.78%)	**75.11% (70.16%)**	67.3% (65.41%)
Average FPR	3.78%	**2.97%**	3.0%	3.8%

Table 2 and **Fig. 11** show that the use of more AE image samples does not necessarily mean a performance improvement. According to the experimental results, the FPR was higher than the original model only for the result that used 100 additional AE samples.

We depict the ROC curves with varying and multiple thresholds because the ROC curves consider all possible thresholds. The overall or balanced accuracies are based on one specific cut-point (or threshold), this means that the accuracy could vary with a different cut-point. On the other hand, the ROC curves take into account all of the possible cut-points. As various thresholds result in different true positive/false positive rates, the ROC curve can show the detailed characteristics of the proposed scheme. In addition, the higher the AUC, the better the classifier's performance. In **Fig. 11**, some class' AUC increases as the number of additional samples increases. The average AUC of the re-trained classifier was the highest with 20 additional samples. Compared to the original model, all re-trained models were improved, however, when the number of the additional samples that are incorporated into the training data is too large, it may affect the classifier negatively; a similar phenomenon is reported in [20]. This suggests a tipping point associated with the amount of added data, beyond which GAN augmentation degrades classifier performance rather than improves it.

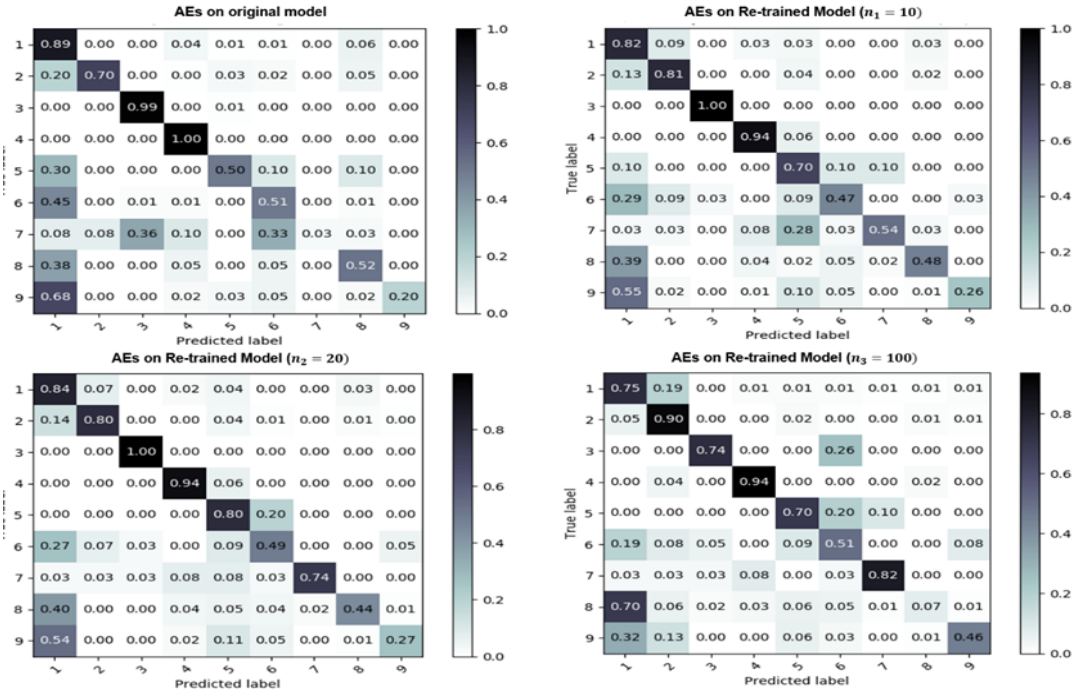

Fig. 10. Confusion Matrices with different numbers of AE per class. The center diagonal values give the proportion of malware images correctly classified. Other values are a proportion of malware images misclassified to that incorrect class.

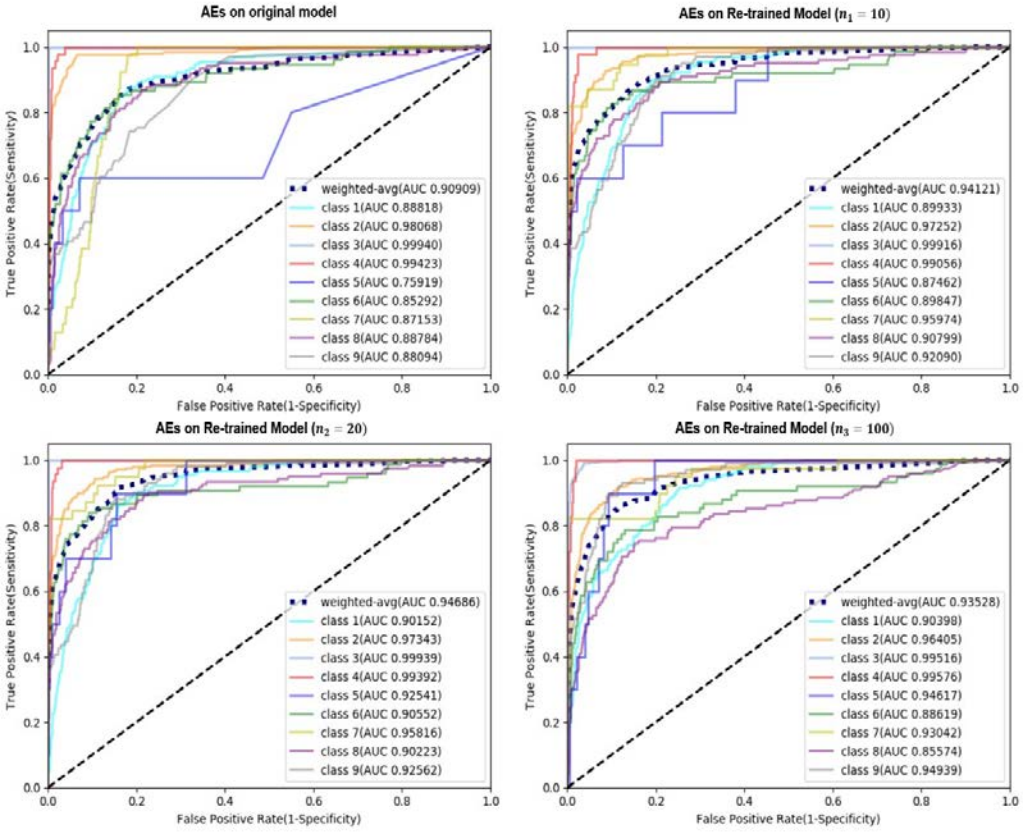

Fig. 11. ROC curves from the classifier according to the numbers of AEs

Summary of findings: The main findings of this paper can be summarized as follows:

- We observe that GAN-based data augmentation can improve a classifier's robustness against AEs. However, excessive data may result in performance degradation for the DNN-based classifier when using a GAN. This can be explained by the fact that when too many additional samples are used for training, the classifier becomes more likely to overfit to some particular perturbation style [20].
- We used only 32 images per class, the additional image samples generated by **G** may not be representative of the entire data distribution. For example, sometimes the trained model was only improved a little (see **Table 2**) because there was not enough data to sufficiently train the GAN. The TPR and AUC of a certain class increases or decreases according to n (the number of added AE image samples for that class). This occurs when training data is so deficient that the generator **G** has been biased to generate only specific classes that are easy to generate, this is called mode collapse. Intuitively, if too many samples are generated from a biased generative network, the classifier will be biased too [29].
- The GAN augmentation method used might degrade the quality of the real images - even when using a conditional version of the generator. Nevertheless, according to the results in **Table 2**, the GAN's reconstruction power shows that it is effective in

improving the model to become much better at detecting AEs without adversely affecting the performance against standard malware.

6. Conclusion

In this paper, we presented a methodology to improve the robustness of DNN-based malware detection systems against AEs. We provided experimental evidence that a GAN can be successfully used as a defense mechanism against (unseen) malware variants. While performing our experiments, we did not use any image re-sizing as this may result in image feature loss. However, image down-sizing may reduce computational complexity and thus enable the use of more adversarial images in the training data. In our future work, we plan to use the entire dataset to train the GAN to further investigate the GAN's efficacy in this task. Considering more sophisticated adversary models will also be part of our future work.

References

[1] Kim, Jin-Young, Seok-Jun Bu, and Sung-Bae Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Information Sciences*, 460-461, 83-102, 2018. Article (CrossRef Link)

[2] Hwantae Ji, Eulgyu Im, "Malware Classfication Using Machine Learning and Binary Visualization," *The Korean Institute of Information Scientists and Engineers*, 24, 198-203, 2018. Article (CrossRef Link)

[3] Saxe, Joshua, and Konstantin Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. of 2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, IEEE, 2015. Article (CrossRef Link)

[4] Kim, H., Han, S., Lee, S., & Lee, J.-R., "Visualization of Malwares for Classification Through Deep Learning," *Journal of Internet Computing and Services*, 19(5), 67–75, 2018. Article (CrossRef Link)

[5] Nataraj, Lakshmanan, et al., "Malware images: visualization and automatic classification," in *Proc. of the 8th international symposium on visualization for cyber security*, ACM, 1-7, 2011. Article (CrossRef Link)

[6] Szegedy, Christian, et al., "Intriguing properties of neural networks," *arXiv preprint arXiv: 1312.6199*, 2013.

[7] Nguyen, Anh, Jason Yosinski, and Jeff Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2015. Article (CrossRef Link)

[8] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv: 1412.6572*, 2014.

[9] Papernot, Nicolas, et al., "The limitations of deep learning in adversarial settings," in *Proc. of 2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2016. Article (CrossRef Link)

[10] Biggio, Battista, et al., "Evasion attacks against machine learning at test time," in *Proc. of Joint European conference on machine learning and knowledge discovery in databases*, Springer, Berlin, Heidelberg, 387-402, 2013. Article (CrossRef Link)

[11] Goodfellow, Ian, et al., "Generative adversarial nets," *Advances in neural information processing systems*, 2014.

[12] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in neural information processing systems*, 2012.

[13] Zhu, Xinyue, et al., "Emotion classification with data augmentation using generative adversarial networks," in *Proc. of Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, Cham, 349-360, 2018. Article (CrossRef Link)

[14] Samangouei, Pouya, Maya Kabkab, and Rama Chellappa, "Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models," 2018.
[15] Oliva, A. and Torralba. A, "Modeling the shape of a scene: a holistic representation of the spatial envelope," *International Journal of Computer Vision*, 42(3), 145-175, 2001. Article (CrossRef Link)
[16] G. Daniel, et al., "Using convolutional neural networks for classification of malware represented as images," *Journal of Computer Virology and Hacking Techniques*, 15.1, 15-28, 2019. Article (CrossRef Link)
[17] Chen, Li, "Deep transfer learning for static malware classification," *arXiv preprint arXiv:1812.07606*, 2018.
[18] Xiao, Chaowei, et al., "Generating adversarial examples with adversarial networks," *arXiv preprint arXiv:1801.02610*, 2018.
[19] Papernot, Nicolas, et al., "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. of 2016 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2016. Article (CrossRef Link)
[20] Grosse, Kathrin, et al., "Adversarial examples for malware detection," in *Proc. of European Symposium on Research in Computer Security*, Springer, Cham, 62-79, 2017. Article (CrossRef Link)
[21] Kolosnjaji, Bojan, et al., "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *Proc. of 2018 26th European Signal Processing Conference (EUSIPCO)*, IEEE, 2018. Article (CrossRef Link)
[22] Anderson, Hyrum S., et al., "Evading machine learning malware detection," *Black Hat*, 2017. Article (CrossRef Link)
[23] Hu, Weiwei, and Ying Tan, "Generating adversarial malware examples for black-box attacks based on GAN," *arXiv preprint arXiv:1702.05983*, 2017.
[24] Mariani, Giovanni, et al., "Bagan: Data augmentation with balancing gan," *arXiv preprint arXiv:1803.09655*, 2018.
[25] Mirza, Mehdi, and Simon Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
[26] Odena, Augustus, Christopher Olah, and Jonathon Shlens, "Conditional image synthesis with auxiliary classifier gans," in *Proc. of the 34th International Conference on Machine Learning-Volume 70. JMLR. org*, 2017.
[27] Radford, Alec, Luke Metz, and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.
[28] Maaten, Laurens van der, and Geoffrey Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, 9, 2579-2605, 2008.
[29] Bowles, Christopher, et al., "GAN augmentation: augmenting training data using generative adversarial networks," *arXiv preprint arXiv:1810.10863*, 2018.
[30] Antoniou, Antreas, Amos Storkey, and Harrison Edwards, "Data augmentation generative adversarial networks," *arXiv preprint arXiv:1711.04340*, 2017.
[31] Kolosnjaji, Bojan, et al., "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *Proc. of 2018 26th European Signal Processing Conference (EUSIPCO)*, IEEE, 2018. Article (CrossRef Link)
[32] Wang, Zhou, Eero P. Simoncelli, and Alan C. Bovik, "Multiscale structural similarity for image quality assessment," in *Proc. of The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, Ieee, Vol. 2, 2003. Article (CrossRef Link)
[33] Ronen, Royi, et al., "Microsoft malware classification challenge," *arXiv preprint arXiv:1802.10135*, 2018.
[34] Brodersen, Kay Henning, et al., "The balanced accuracy and its posterior distribution," in *Proc. of 2010 20th International Conference on Pattern Recognition*, IEEE, 2010. Article (CrossRef Link)
[35] Hanley, James A., and Barbara J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, 143.1, 29-36, 1982. Article (CrossRef Link)

Mingu Kang received the B.S. degree in computer science from Korea National University of Transportation and the M.S degree from Korea Advanced Institute of Science & Technology, Korea, in 2013 and 2019, respectively. His research motivation stems from making the Deep Neural Networks being robust against uncertainty of real-world. His current research topic includes adversarial attack, semi-supervised learning, self-supervised learning and medical AI.

Hyeonggyeom Kim received the B.S. degree in computer science from Korea National University of Transportation in 2015 and 2018. His current research topic includes federated learning and malware classification.

Suchul Lee received the BS and Ph. D degrees in computer science from Seoul National University, Seoul, South Korea in 2008 and 2014, respectively. He is currently an associate professor in the Department of Data Science at Korea National University of Transportation (KNUT), Uiwang, Korea. From 2014 to 2016, he was a member of research staff in in National Security Research Institute, Daejon, Korea. His research interests include computer and information security, wireless and mobile systems, Internet applications, such as 802.11, cognitive radio, and traffic analysis with an emphasis on system performance optimization.

Seokmin Han is currently an associate professor in the major of Data Science at KNUT(Korea National University of Transportation). He obtained B.S., M.S, Ph.D degrees from Seoul National University in 2000, 2003,2008, respectively. His research interests include Deep learning and Image Processing.