*big data and cognitive computing*

MDPI

*Article*

# Developing a Robust Defensive System against Adversarial Examples Using Generative Adversarial Networks

**Shayan Taheri †, Aminollah Khormali †, Milad Salem and Jiann-Shiun Yuan ***

Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816, USA; shayan.taheri@knights.ucf.edu (S.T.); aminkhormali@knights.ucf.edu (A.K.); milad73s@gmail.com (M.S.)

* Correspondence: Jiann-Shiun.Yuan@ucf.edu; Tel.: +1-407-823-5719

† These authors contributed equally to this work.

Received: 29 April 2020; Accepted: 19 May 2020; Published: 22 May 2020


**Abstract:** In this work, we propose a novel defense system against adversarial examples leveraging the unique power of Generative Adversarial Networks (GANs) to generate new adversarial examples for model retraining. To do so, we develop an automated pipeline using combination of pre-trained convolutional neural network and an external GAN, that is, Pix2Pix conditional GAN, to determine the transformations between adversarial examples and clean data, and to automatically synthesize new adversarial examples. These adversarial examples are employed to strengthen the model, attack, and defense in an iterative pipeline. Our simulation results demonstrate the success of the proposed method.

**Keywords:** adversarial machine learning; botnet detection; generative adversarial networks; machine learning

## 1. Introduction

Machine learning models, specifically deep learning networks, have proven themselves to be capable tools in a wide range of challenging domains, such as computer vision, healthcare, industry, finance, and so forth. For instance, machine learning applications are leveraged for network traffic analysis and in particular botnet [1] detection in ever growing number of Internet of Things (IoT) devices (28.1 billion devices in 2020 and expected to reach trillions in 2025 [2]). Traditional machine learning-based botnet detection approaches, including statistical methods and behavioral techniques, rely on the extraction of patterns from empirical data using a set of selective features. However, these techniques require expert domain knowledge for best feature engineering. Recently, researchers in the community have proposed representation learning for automatic learning of representative features from raw data as a technique to overcome to problems of traditional machine learning methods. Convolutional Neural Networks (CNNs) are one of the well-established methods for representation learning, where network traffic data can be fed to the model as either numeric or image format [3]. The later is known as visualization-based botnet detection approach, which are utilized for data characteristics understanding and embedding features visualization [3,4]. These deep learning models not only are capable of learning the representative properties of the network traffic data but also can be used for prediction of the label of the future network traffic data in an automated manner.

With great usage and wide adoption of neural networks in sensitive domains, the reliability of their results is a necessary. However, this widespread usage of deep learning models has became an incentive for adversarial entities to manipulate the inference of these learning models such that the output is not correct [5]. In last few years, it has been proven that deep learning models are vulnerable in

making inference from manipulated input data [6,7]. When presented with inputs containing minor perturbation, that is, adversarial examples, deep learning networks would be forced to misclassify the input and be fooled into giving wrong results [6–9]. As deep learning models learn the inherent pattern of input dataset, introducing a small yet carefully designed perturbation into the dataset would adversely impact the output of the model. In the concept of adversarial machine learning, the generated adversarial examples are very similar to the original ones and not necessarily output of the input data distribution. Thus, detection of adversarial examples is a challenging task. There exist several algorithms to systemically generate adversarial examples [10]. Although, there exist algorithms that work based on evolutionary-based optimization methods [11], most of the adversarial example generator algorithms are based on the gradient analysis of the model output with regard to the input. For instance, Goodfellow et al. proposed Fast Gradient Sign Method (FGSM) [12], which generates adversarial examples based on only sign of the gradient. Kurakin et al. proposed Basic Iterative Method (BIM) [13] that basically iterates FGSM method for several steps. DeepFool [8] is proposed by Moosavi-Dezfooli et al. to find the closest distance between a given input and decision boundary of adversarial examples. Papernot et al. presented Jacobian-based Saliency Map Attack (JSMA) [6] that tries to generate an adversarial attack through perturbing a limited number of input features. Although adversarial learning attacks are initially developed in the image domain, they are actively applied on other sensitive domains, as well. For example, Grosse et al. [14] have expand on existing adversarial example generating algorithms to create a highly-effective attack that utilizes adversarial examples against malware detection models. Osadchy et al. [15] have utilized adversarial examples to generate a secure CAPTCHA scheme.

In response to the vulnerability of deep learning networks to adversarial examples, there has been huge interest in building defensive systems to improve the robustness of deep learning models [12,16–18]. However, in a recent work it was shown that approximately all the proposed approaches to defend against adversaries can be circumvented, with the approach of *Retraining* being an exception [19]. Note that any defense mechanism not only should be robust against existing adversarial attacks, but also performs well against future and unseen adversarial attacks within the specified threat model. Thus, any defense proposal should be adaptive, as well.

In this work, we propose a novel defense system leveraging the unique power of Generative adversarial networks (GANs) [20] to generate adversarial examples for retraining. We approach attacking the deep learning network in an automated way, using an external GAN, that is, Pix2Pix [21] conditional GAN, to understand the transformations between adversarial examples and clean data, and to automatically generate unseen adversarial examples. After attacking the neural network, we generate a plethora of adversarial examples in an iterative manner and use them to automate defense against adversaries. Via doing so, we develop a defense mechanism which is practical in real-world application with pre-trained deep learning network. Since the vulnerability against adversaries is inherent to the world of neural networks, using an iterative offensive approach to generate new attacks to help strengthen the neural network is the best defense.

Particularly, the main contributions of this work are structured as follows:

- Developing a novel (attacking) method for adversarial example generation, which learns the initial data distribution of common adversarial examples and shifts it to fool a pre-trained deep learning model.
- Creating new adversarial examples that can pass undetected to models trained on the initial common adversarial examples.
- Attaching a pre-trained CNN to a Pix2Pix GAN and learning the generation with the goal to fool the attached network, a technical feat which has not been done before.
- Implementing a novel iterative pipeline in order to strengthen the model, attack, and defense in an iterative manner. After each iteration the attacker generates stronger adversarial examples, while the robustness of the model increases through retraining and updating associated weights.

- Conducting extensive experiments to evaluate the performance of the proposed method, with demonstrating an application for visualization-based botnet detection systems.

The rest of the paper is organized as follows: Background information and preliminaries are discussed in Section 2. Our developed robust defensive system against adversarial attacks leveraging the power of generative adversarial networks is presented in Section 3. We evaluate the performance of the proposed method and discuss the results in Section 4. Finally, conclusion and future work is placed in Section 5.

## 2. Background and Preliminaries

In this work we conduct a comprehensive study that lies in the intersection of computer vision, artificial intelligence, and cybersecurity in an attempt to develop an automated defensive system against adversarial attacks. In this section, we describe required background information and fundamental concepts.

### 2.1. Network Traffic Data Visualization

Visualization-based network traffic data analysis has recently introduced by Wang et al. [22]. The proposed approach takes three stages to convert network traffic data into gray-scale images, including traffic splitting, traffic clearing, and image generation. These stages are described as: (1) Traffic split: This step is responsible for splitting packet capture files to multiple discrete traffic units into either packet capture for flow-based layers or binary for flow-based application layer. (2) Traffic clear: This step is responsible for anonymization/sanitization, where media access control addresses and the Internet Protocol (IP) addresses are randomized in data link layer and IP layer, respectively. Furthermore, duplicate or empty data are removed. (3) Image generation: In this step, first traffic data that have a significantly different size from the rest of the data, outliers, are removed. Next, the remaining data are adjusted to a specific length, which is then transformed into gray-scale images.

### 2.2. Adversarial Attacks in Deep Learning

A threat model guides the generation of the adversarial examples and also defines the capabilities of the attacker. In this study, the threat model is defined and discussed as follows:

1. The adversaries can only attack at the test stage after the training has been done. Therefore, training data poisoning will not be examined.
2. In this work, deep convolutional neural networks will be examined as the model under attack. While traditional machine learning models, such as like Support Vector Machines (SVM) [23] or Random Forest (RF) [24] can also deliver high accuracy, it has been shown that adversarial examples found in deep neural networks work effectively against these traditional models [25,26].
3. Adversaries aim to compromise integrity, which can be defined using one of the performance metrics such as accuracy or F1-score.
4. The goal of the attacker is adversarial falsification. If an adversary is trying to fool the model to misclassify as input as positive or negative, the attacks differ. In the case of botnet detection, the adversary can try to make a botnet be classified as non-malicious and launch a successful attack or make the normal data be classified as malicious and cause disastrous consequences.
5. In this study, we assume that the attacker is launching attacks frequently and in an iterative manner.
6. The adversary has full knowledge about model structure and its internals, that is, white-box attacks.

Based on the outline of our threat model, it lies in relatively easy attacks, where any attacker can design an attack to make an adverse impact on the output of deep learning models. Note that building

an effective system to defend against such prevalent attacks would immune deep learning models against a huge number of potential threats, which is the main goal of this study.

### *2.3. Generative Adversarial Networks*

Generative adversarial networks are a class of machine learning models comprised of two neural networks, including a Generator network ($G$) and a Discriminator network ($D$) [20,27]. This structure makes GANs capable of synthesizing data points similar to those of the original data distribution. These networks have the capability to produce synthetic images that look perfect to human beings [28], compose music [29], and even generate new molecules [30]. Similarly, this generative power can be used to synthesize adversarial examples.

Generative network is responsible to generate perceptually convincing synthesized samples that appear to have been drawn from a real data distribution $P_{\text{data}}$. The generator takes a noise vector $\mathbf{z}$ from a distribution of $P_z$ and generates a new synthesized sample, $G(\mathbf{z})$, which has same dimensions as real sample. On the other hand, the discriminator is a binary classifier that takes both real sample and the synthesized one as input and calculates the probability that the sample is real rather than fake. Typically, training process of a GAN is involved with solving the following optimization problem (1), where loss of the generator is directly related to the performance of the discriminator.

$$\min_G \max_D V(D,G) = \underset{x\sim P_{\text{data}}}{E}[\log D(\mathbf{x})] + \underset{z\sim P_z}{E}[\log(1 - D(G(\mathbf{z})))] \quad (1)$$

Here, $V(D,G)$ is the objective function, $P_{\text{data}}$ is real data distribution, $D(x)$ denotes the probability that $D$ discriminates $x$ as real data, $P_z$ is noise distribution, $G(z)$ is the sample generated by the generator, and $D(G(z))$ indicates the probability that $D$ determines the sample created by generator $G(z)$. While the discriminator tries to maximize its cost function through minimizing prediction errors, the generator tries to minimize its cost function through generating samples that are not detectable by the discriminator. More detailed information about GANs can be found in References [20,27].

## 3. Methodology

The detail of our proposed attack and defense system is proposed in this section. As it can be seen in Figure 1, the proposed method is composed of four main components, including DL-based botnet detector, gradient-based adversarial attacks, GAN-based adversarial attacks, and defensive mechanism.

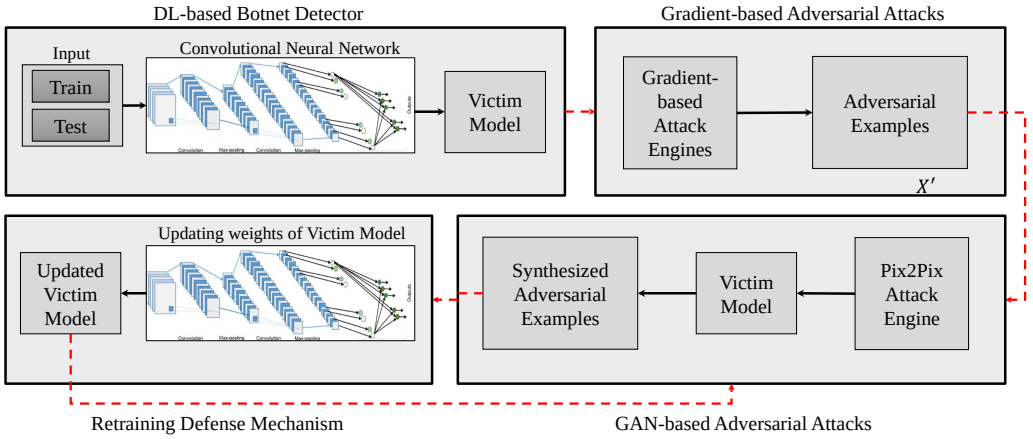



**Figure 1.** General structure of our proposed attack and defense system. It is composed of four main components, including DL -based botnet detector, Gradient-based adversarial attacks, GAN -based Adversarial attacks, and defense mechanism based on retraining of victim model.

### *3.1. Victim Model*

Convolutional neural networks are one of the well-known representation learning techniques, which is widely employed in a wide range of sensitive domains ranging from computer vision to visualization-based botnet detection. Thus, in this study we employed a convolutional neural network as our baseline botnet detection model. We refer to this baseline model as victim model, as we assumed that this model is under adversarial attacks. Note that, as we explain in more details in next sections, the main goal of this work is to develop a defensive system to improve the robustness of this model against adversarial attacks leveraging unique power of generative adversarial attacks.

To build our victim model, network traffic data is converted into gray-scale images. These images contain representative information about nature of the traffic, whether they are normal traffic data or related to malicious activity. Our botnet detector is composed of two consecutive convolutional layers as feature extractor and a fully connected layer as classifier.

### *3.2. Attack Engines*

In this section, we described our attack engines that have been employed to generate adversarial examples to attack our victim model. First, we utilized generic gradient-based attack engines to generate adversarial examples. These adversarial attack methods are well-established and include, fast gradient sign method [12], DeepFool [8], and projected gradient descent [31], which will be discussed in Section 3.2.1. Second, we employed a custom generative adversarial network that is able to generate unlimited number of adversarial examples, which will be described in Section 3.2.2.

#### 3.2.1. Gradient-Based Attack Engine

In recent couple of years, researchers have introduced several adversarial attack methods on deep learning models. Most of these attack methods are working based on gradient descent. In this study, we employed three most common attack engines, including FGSM, DeepFool, and PGD , as examples to conduct adversarial attacks on DL-based botnet detection system. General structure of our gradient-based adversarial attacks on DL-based botnet detection system is shown in Figure 2. In the following, we describe each of these attack engines briefly, and refer interested readers to original papers for more information.

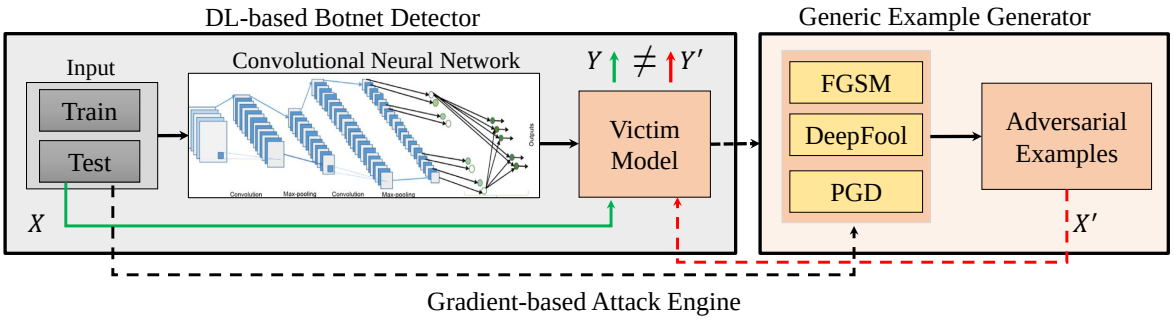



**Figure 2.** General structure of the generic adversarial attack on deep learning-based botnet detection systems. In this work, we utilized three common gradient-based adversarial attacks on our DL-based botnet detector (victim model), including Fast Gradient Sign Method (FGSM), DeeFool, and PGD.

- **FGSM:** FGSM is a fast method introduced by Goodfellow et al. in 2015 [12], which updates each feature in the direction of the sign of the gradient. This perturbation can be created after performing back-propagation. Being fast makes this attack a predominant attack in real world scenarios. FGSM can be formulated using (2).

$$X_{\text{Adv}} = X + \varepsilon \cdot \text{sign}\left(\nabla_X J(X, Y)\right) \tag{2}$$

Here, $X$ is the clean sample, $X_{\text{Adv}}$ is the adversarial example, $J$ is the classification loss, $Y$ is the label of the clean sample, and $\varepsilon$ is a tunable parameter that controls the magnitude of the perturbation. Note that, in FGSM only direction of the gradient is important not its magnitude.

- **DeepFool:** This attack is introduced by Moosavi-Dezfooli et al. [8] as an untargeted iterative attack based on the $L_2$ distance metric. In this attack the closest distance from the clean input to the decision boundary is found. Decision boundaries are the boundaries that divide different classes in the hyper-plane created by the classifier. Perturbations are created in a manner that pushes the adversarial example outside of the boundary, causing it to be misclassified as another class, as demonstrated in Algorithm 1.

**Algorithm 1:** The process of generating adversarial examples based on DeepFool method [8].

input: Image $x$, classifier $f$
output: Perturbation $\hat{\boldsymbol{r}}$
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}, i \leftarrow 0$
while $\operatorname{sign}(f(\boldsymbol{x}_i)) = \operatorname{sign}(f(\boldsymbol{x}_0))$ do
  $\boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla f(x_i)\|_2^2} \nabla f(\boldsymbol{x}_i)$
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
  $i \leftarrow i + 1$
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$

- **PGD:** This attack is proposed by Madry et al. [31] as an iterative adversarial attack that creates adversarial examples based on applying FGSM on a data point $x_0$, in an iterative manner, that is obtained by adding a random perturbation of magnitude $\alpha$ to the original input $x$. Then the perturbed output is projected to a valid constrained space. The projection is conducted by finding the closet point to the current point within a feasible region. This attack can be formulated based on the following equation.

$$x^{i+1} = \text{Proj}_{x+S}\left(x^i + \alpha \operatorname{sign}\left(\nabla_{x^i} J\left(\theta, x^i, y\right)\right)\right), \tag{3}$$

where $x^{i+1}$ is the perturbed input at iteration $i+1$ and $S$ denotes the set of feasible perturbations for $x$.

#### 3.2.2. GAN-Based Attack Engine

Generative adversarial networks have the capability to produce synthetic images that look perfect to human beings [28], compose music [29], and even generate new molecules [30]. In this work, we utilize a conditional generative adversarial network specifically designed for image domain, that is, Pix2Pix [21]. In conditional GANs additional information, such as source image, is provided. Thus, loss function of a conditional GAN can be determined as (4).

$$\mathcal{L}_{cGAN}(G.D) = E_{x\sim p_{\text{data}}(x)}[\log(D(x.y))] + E_{z\sim p_z(z)}[\log(1 - D(G(z.y).y))]. \tag{4}$$

This fact makes Pix2Pix a perfect fit for the image classification and botnet detection domain to be used to understand the transformations between normal data and malicious data and generating unlimited amounts of new adversarial examples. In Pix2Pix, the generator is build based on the U-net

structure, while the discriminator has built based on PatchGAN architecture. We refer the interested readers to the source paper for more information about Pix2Pix structure [21].

Pix2Pix is a type of conditional GAN employs a loss function to train the mapping from input image to output image. This adversarial loss function, shown in (5), forces the generator to create a sample that resembles the conditioning variable $x$. Finally, this adversarial loss function is added to (4).

$$\mathcal{L}_{L1}(G) = E_{x,y,z}\left[\|x - G(z.y)\|_1\right]. \tag{5}$$

The loss function in this work is as (6).

$$\mathcal{L}(G.D) = \mathcal{L}_{cGAN}(G.D) + \lambda\mathcal{L}_{L1}(G), \tag{6}$$

where $\lambda$ is a hyper-parameter that controls the weight of the term.

The general architecture of our Pix2Pix-based adversarial example generator is shown in Figure 3. After the transformations are fond, any given clean data from the dataset can be transformed into a malicious data. In other words, the Pix2Pix learns how to add perturbation to the data in an automated manner. This automated attacking is leveraged, and a huge corpus of new attack data is generated. To enable the Pix2Pix to generate better attacks, each generated image is fed to the victim model and is tested to see if it fools the model or not. This can be formulated via adding a loss function for the attack on the model, which can help the training of the Pix2Pix. By doing so, the Pix2Pix autonomously attacks the model using the understanding of the perturbation distribution and is able to generate huge corpus of unseen adversarial examples.

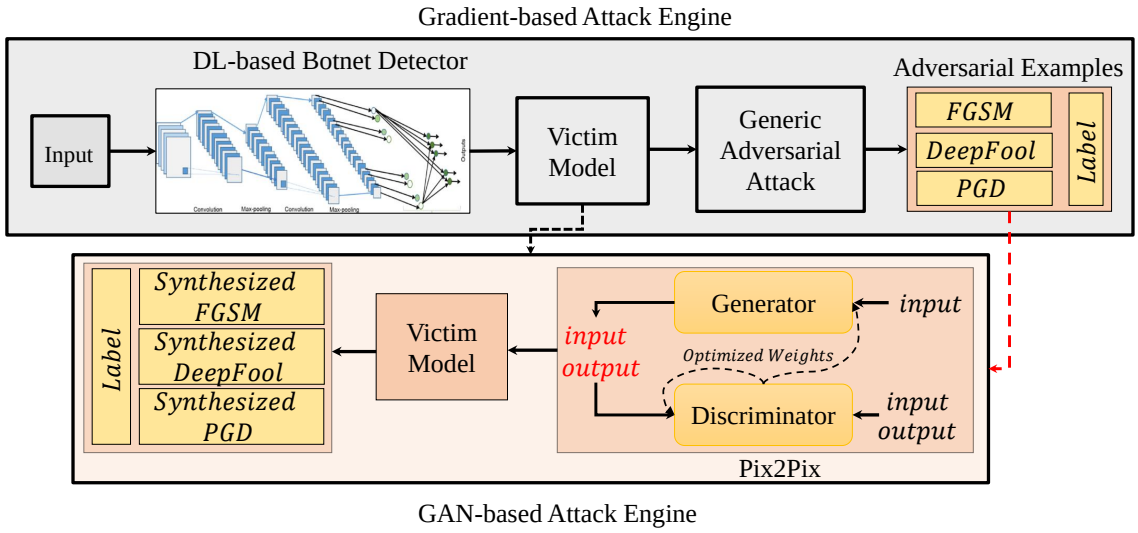



**Figure 3.** General structure of the GAN-based adversarial attacks on DL-based botnet detection system. Here, we employed Pix2Pix to synthesize new adversarial examples that are similar to adversarial examples generated in Section 3.2.

### *3.3. Defense Mechanism*

Having generated a substantial number of adversarial examples in an automated manner, we can defend against them using retraining approach. Retraining consists of feeding the adversarial data back to the victim model and training it again. Having done so, we can convey to the neural networks that it is making mistakes in certain inputs. The neural network then learns how to avoid these mistakes by improving its decision boundaries, as shown in Figure 4. This method has been proven to withstand adversarial attacks when other defenses, such as change in structure, have failed.

General structure of our proposed defensive method is shown in Figure 5. This can be done in the following steps; Step 1—Fine-tune the neural network using the generative adversarial examples,

improving the model. Step 2—Generate more examples via using the formulated automated attack on the improved model. Step 3—Iterate by going back to step 1 until desired performance metric is reached.

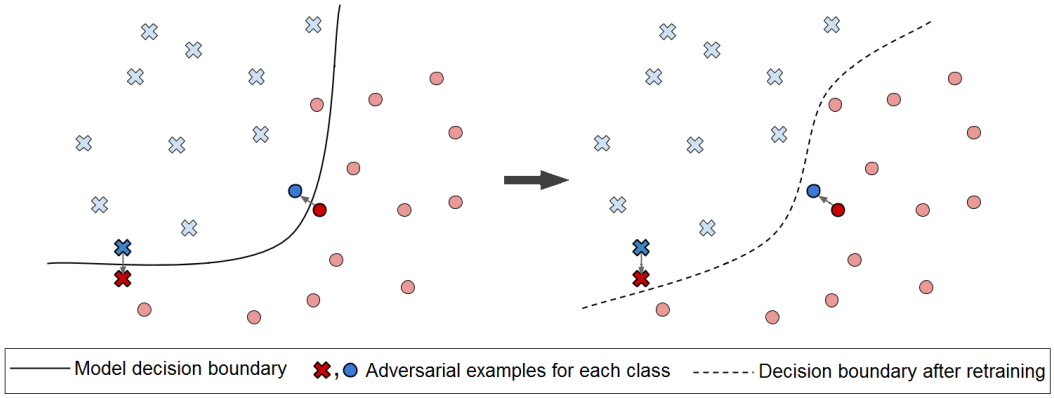



**Figure 4.** In retraining the defensive approach the victim model learns how to avoid mistakes by improving its decision boundaries.

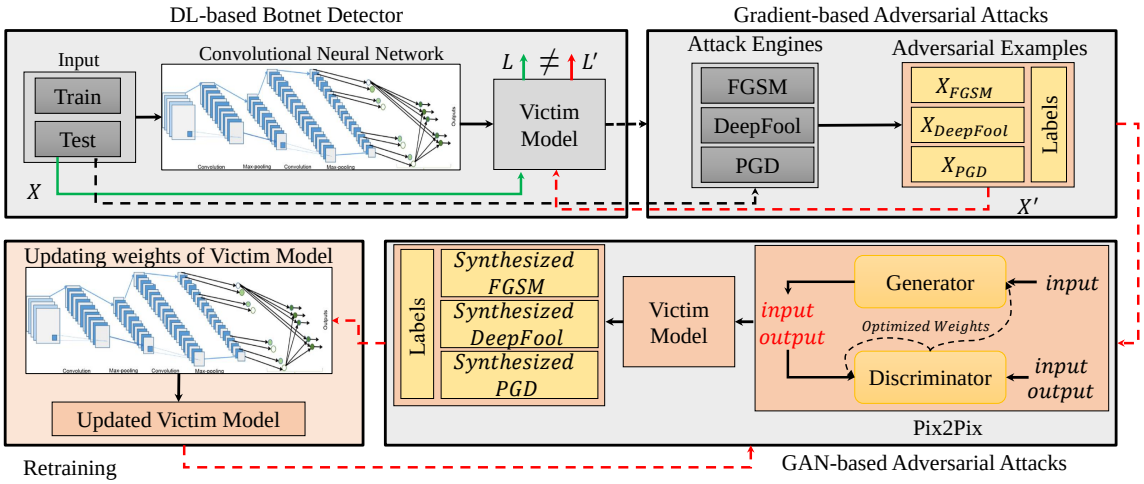



**Figure 5.** General architecture of our proposed attack and defense system. In the retraining process the weights of the victim model are being updated in each iteration, thus improves its decision boundaries. In return, this process also yields to more powerful adversarial examples after each iteration.

This iterative approach will cause the neural network to evolve over time. The impact of this defense can be significant on neural networks in security domain. With the example of malware detection in mind, this technique allows for new adversarial examples to be generated on each iteration for a specific pre-trained neural network. This model which carries out the task of malware detection in real life is then retrained on the new attack data and becomes more robust towards them. This task iterates, and the model evolves until certain accuracy is reached.

## 4. Evaluation and Discussion

We evaluate the performance of the proposed method for both generating new adversarial examples and defending against them through comprehensive experiments. In the following, first we give a brief description about utilized dataset, experimental setup, and evaluation metrics in Section 4.1, Section 4.2 and Section 4.3, respectively. Finally, the obtained results are discussed in Section 4.4.

### *4.1. Dataset*

To evaluate the performance of the proposed approach, we utilized CTU-13 dataset [32] that is a dataset of botnet, normal, and background traffic with corresponding labels. In this study, we only used normal and botnet traffic data. Table 1 shows the distribution of the normal and botnet samples in this study. We randomly selected a smaller subset of train and test data, while maintaining the original distribution, totalling into 50,000 samples as train samples and 10,000 samples as test data, to conduct our experiments. All train and test data are converted into gray-scale images based on the technique described in Section 2.1.

**Table 1.** Distribution of train and test samples across normal and botnet classes.

| | Train | Test |
|---|---|---|
| **Normal** | 34,144 | 7376 |
| **Botnet** | 15,856 | 2624 |
| **Total** | 50,000 | 10,000 |

### *4.2. Experimental Setup*

For reproducibility of our proposed method, in the following we provide detailed information about the experimental setup we employed in this study.

**Implementation** To implement our attack algorithms we utilized a Python library, Cleverhans [33], which is a library to benchmark machine learning systems' vulnerability to adversarial examples. It is an adversarial example library for constructing attacks, building defenses, and benchmarking [33].

**Parameters** The proposed method has several methods for each attack and defense scenario. Several experiments were conducted with different values for each parameter to achieve desirable results, that is, higher misclassification rate in attack phase, while lower fooling rate in defence phase. For instance, PGD has two parameters to be tuned, $\epsilon$ and number of iterations, which were set to 0.15 and 10, respectively. With this setting we were able to achieve a fooling rate of 99.98%.

**Evaluation System** All of the experiments are conducted on a Lambda Quad deep learning workstation with Ubuntu 18.04 OS, Intel Xeon E5-1650 v4 CPU, 64 GB DDR4 RAM, 2TB SSD, 4TB HDD, and 4 NVIDIA Titan-V Graphics Processing Units (GPUs).

### *4.3. Evaluation Metrics*

In order to find out how well the attacks and defense mechanisms are performing, a set of proper performance metrics need to be defined. In classification tasks this metrics can be accuracy or F1 score as defined in (7) and (8), respectively. Although in the adversarial example generation domain accuracy score is a mandatory metric, we need more evaluation metrics to measure the quality of the synthesized examples and their fooling rate. The quality of generated adversarial examples is examined via calculating the distance metric as average of their $L_2$ norm distance between the normalized form of clean data and the synthesized data. Furthermore, the fooling rate is considered to be the number of samples that are not classified correctly.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{7}$$

$$\text{F1 Score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{8}$$

where True Positive (TP) and True Negative (TN) are correctly identified botnet and normal samples, respectively. While False Positive (FP) is normal samples that are classified as botnet, False Negative (FN) is botnet samples that are classified as normal. These metrics are used to create the confusion matrix, as shown in Table 2.

**Table 2.** The confusion matrix for botnet detection.

| | | **True Label** | |
|---|---|---|---|
| | | **Botnet** | **Normal** |
| **Predicted Label** | **Botnet** | True Positive (TP) | False Positive (FP) |
| | **Normal** | False Negative (FN) | True Negative (TN) |

### *4.4. Results*

In this section, we present our experiments with the systematic approaches described above to see the proposed approach's capabilities in generating new adversarial examples and defending against them. All our experiments are conducted on the CTU-13 dataset [32], which is a dataset with given labels for botnet, normal and background traffic. In this study, we only employed the botnet and the normal traffic data. The collected set are transformed into gray-scale images using the technique described in Section 2.1.

**Victim Model Performance** The generated images are utilized to build our victim model based on convolutional neural network architecture, as described in Section 3.1. The evaluation results of these experiments are shown in Table 3. As it can be seen from this table, our model is able to achieve an accuracy rate of 99.99% and an F1 score of 99.98%. Note that this model will be used as our victim model (baseline) model for generating adversarial examples and defending against them.

**Table 3.** Analysis of confusion matrix of victim model in classification of botnet and normal traffic data.

| **Confusion Matrix** | | | | **Accuracy Rate (%)** | **F1 Score (%)** |
|---|---|---|---|---|---|
| | | Predicted Label | | 99.99 | 99.98 |
| | | Botnet | Normal | | |
| True Label | Botnet | TP = 2624 | FN = 0 | | |
| | Normal | FP = 1 | TN = 7375 | | |

**Gradient-based Adversarial Attacks** In order to examine the robustness of our victim model against gradient-based adversarial attacks, we used three well-established adversarial methods for experiments. Our obtained results demonstrate that all FGSM, DeepFool, and PGD methods are able to fool the classifier with high success rate. The obtained results for these experiments are presented in Table 4. For instance, PGD is able to achieve a fooling rate of 99.98% with an average distortion rate of 18.93%, outperforming the other two attacks. We assume that lower fooling rate of the DeepFool is due to its inherent presumptions, where classifiers are considered to be linear with hyperplanes separating each of the classes from another.

**Table 4.** Fooling and average distortion rate of each adversarial attack method.

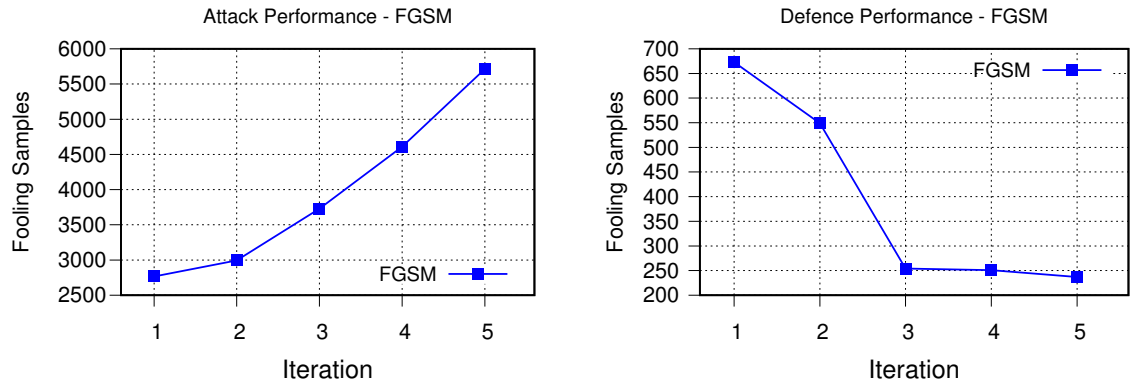
| **Attack** | **Fooling Rate (%)** | **Distortion Rate (%)** |
|---|---|---|
| **FGSM** | 99.69 | 39.73 |
| **DeepFool** | 67.73 | 43.93 |
| **PGD** | 99.98 | 18.93 |

**GAN-based Iterative Attack and Defense** By generating a plethora of adversarial examples using gradient-based attack methods, the main goal of the GAN-based iterative attack and defense is to generate both more powerful adversarial examples and improve the robustness of the model at the same time. We evaluated performance of this attack and defense strategy with all three gradient-based adversarial attacks. In the nutshell, we synthesize adversarial examples similar to that of gradient-based approach to retrain the victim model and then test the robustness of the victim

model against original adversarial example, which are not seen by the victim model. This means that we are improving the robustness of the victim model against gradient-based adversarial examples without actually showing them to the victim model.

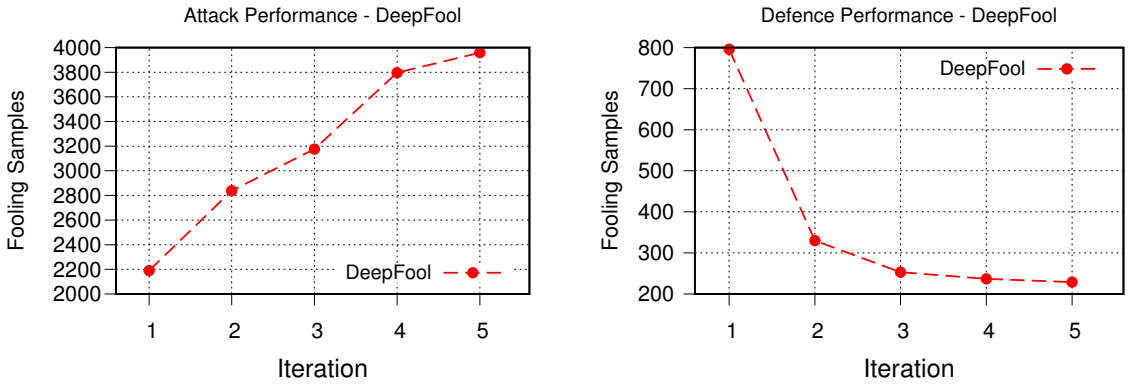
- **FGSM** Figure 6 demonstrates performance of proposed method in generating stronger adversarial examples, which are used to improve the performance of the victim model to defend against FGSM-based adversarial examples. As can be seen, after each iteration it was able to synthesize more samples that are fooling the victim model. Retraining the model using these synthesized examples, on the other hand, as it was expected improves the decision boundaries of the victim model such that fooling rate drops from 673 to 237 samples only after five iterations.
- **DeepFool** The obtained results for iterative attack and defense based on DeepFool are shown in Figure 7. Although the fooling rate of the DeepFool algorithm was only 67.73%, the GAN-based algorithm is able to generate similar number of successful synthesized adversarial examples as FGSM. This is promising as it demonstrates that our GAN-based approach is able to generate new and strong adversarial examples with even fewer number of samples. Similarly, the robustness of the retrained victim model is improved.
- **PGD** The obtained results for iterative attack and defense based on PGD is shown in Figure 8. The obtained results from our experiments demonstrates a similar trend to that of both FGSM and DeepFool methods.

In order to better understand the results, we present our obtained results in a normalized format in Figure 9. As can be seen from this figure, FGSM and PGD methods follow very similar trends both in attack and defense phases. The DeepFool method performs relatively better in attack phase; however, it shows less success in the defense phase. In addition, the trend shows that although number of new fooling samples increases after each iteration, the defensive power of the system saturates after few iterations. This is due to the fact that distribution of the synthesized adversarial samples experiences a shifts after each iteration. Therefore, the victim model starts to learning new distribution other than the original distribution.



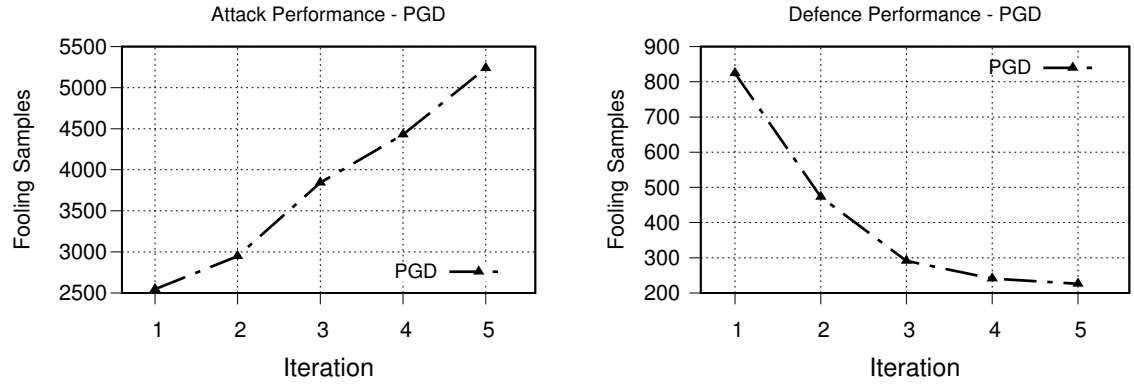

(**a**) Attack Performance (**b**) Defence Performance

**Figure 6.** The results of FGSM.



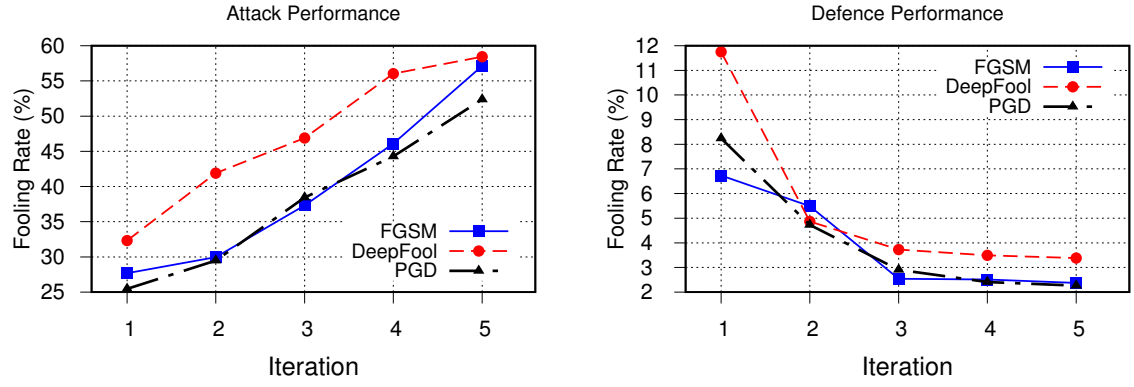

(**a**) Attack Performance (**b**) Defence Performance

**Figure 7.** The results of DeepFool.




(**a**) Attack Performance (**b**) Defence Performance

**Figure 8.** Our simulation results based on PGD attack method. While (**a**) presents attack performance, (**b**) shows the success of our proposed defensive system for five iterations.




(**a**) Attack Performance (**b**) Defence Performance

**Figure 9.** Overall simulation results for all three attack types. We observed similar trends for all three attack methods in both attack and defence scenarios.

## 5. Conclusions

Thanks to the powerful learning capabilities of neural networks, they are actively being used in a wide range of domains, specifically sensitive domains. However, it has been shown that adversarial entities are able to manipulate the inference of the model through adversarial examples. Although there exist several defensive methods, it has shown that most of them can be circumvented. Thus, in this

work, we propose a novel defense system leveraging the unique power of GANs to generate adversarial examples for retraining. We approach attacking the neural network in an automated way, using an external GAN, that is, Pix2Pix conditional GAN, to understand the transformations between adversarial examples and clean data, and to generate unseen adversarial examples automatically. After attacking the neural network, we create a plethora of adversarial examples in an iterative manner and use them to automate defense against adversaries. Via doing so, we develop a defense mechanism which is practical in real-world application with pre-trained neural networks. We evaluate the performance of our developed method against visualization-based botnet detection systems. Our results demonstrate the success of our proposed method. For example, the number of fooling adversarial examples generated by the PGD method is decreased from 824 to 226 samples after only five iterations.

**Author Contributions:** M.S. and S.T. came up with the ideas. S.T. and A.K. ran the experiments. A.K. wrote the manuscript. J.-S.Y. provided technical feedback and A.K. revised the manuscript. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported in part by Florida Center for Cybersecurity.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Silva, S.S.; Silva, R.M.; Pinto, R.C.; Salles, R.M. Botnets: A survey. *Comput. Netw.* **2013**, *57*, 378–403.
2. Manyika, J.; Chui, M.; Bisson, P.; Woetzel, J.; Dobbs, R.; Bughin, J.; Aharon, D. *Unlocking the Potential of the Internet of Things*; McKinsey Global Institute: Washington, DC, USA, 2015.
3. Taheri, S.; Salem, M.; Yuan, J.S. Leveraging Image Representation of Network Traffic Data and Transfer Learning in Botnet Detection. *Big Data Cogn. Comput.* **2018**, *2*, 37.
4. Vinayakumar, R.; Alazab, M.; Srinivasan, S.; Pham, Q.V.; Padannayil, S.K.; Simran, K. A Visualized Botnet Detection System based Deep Learning for the Internet of Things Networks of Smart Cities. *IEEE Trans. Ind. Appl.* **2020**, doi:10.1109/TIA.2020.2971952.
5. Chen, B.; Ren, Z.; Yu, C.; Hussain, I.; Liu, J. Adversarial examples for CNN-based malware detectors. *IEEE Access* **2019**, *7*, 54360–54371.
6. Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z.B.; Swami, A. The limitations of deep learning in adversarial settings. In Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P), San Sebastian, Spain, 7–8 July 2016; pp. 372–387.
7. Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z.B.; Swami, A. Practical black-box attacks against machine learning. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, Abu Dhabi, UAE, 2–6 April 2017; pp. 506–519.
8. Moosavi-Dezfooli, S.M.; Fawzi, A.; Frossard, P. Deepfool: A simple and accurate method to fool deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2574–2582.
9. Wang, B.; Yao, Y.; Viswanath, B.; Zheng, H.; Zhao, B.Y. With great training comes great vulnerability: Practical attacks against transfer learning. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 1281–1297.
10. Yuan, X.; He, P.; Zhu, Q.; Li, X. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2805–2824.
11. Su, J.; Vargas, D.V.; Sakurai, K. One pixel attack for fooling deep neural networks. *IEEE Trans. Evol. Comput.* **2019**, *23*, 828–841.
12. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. In Proceedings of the International Conference on Learning Representation ICLR, San Diego, CA, USA, 7–9 May 2015.
13. Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial examples in the physical world. *arXiv* **2016**, arXiv:1607.02533.
14. Grosse, K.; Papernot, N.; Manoharan, P.; Backes, M.; McDaniel, P. Adversarial examples for malware detection. In Proceedings of the European Symposium on Research in Computer Security, Oslo, Norway, 11–15 September 2017; pp. 62–79.

15. Osadchy, M.; Hernandez-Castro, J.; Gibson, S.; Dunkelman, O.; Pérez-Cabo, D. No bot expects the DeepCAPTCHA! Introducing immutable adversarial examples, with applications to CAPTCHA generation. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2640–2653.
16. Buckman, J.; Roy, A.; Raffel, C.; Goodfellow, I. Thermometer encoding: One hot way to resist adversarial examples. In Proceedings of the International Conference on Learning Representation ICLR, Vancouver, BC, Canada, 30 April–3 May 2018 .
17. Guo, C.; Rana, M.; Cisse, M.; Van Der Maaten, L. Countering adversarial images using input transformations. In Proceedings of the International Conference on Learning Representation ICLR, Vancouver, BC, Canada, 30 April–3 May 2018 .
18. Song, Y.; Kim, T.; Nowozin, S.; Ermon, S.; Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In Proceedings of the International Conference on Learning Representation ICLR, Vancouver, BC, Canada, 30 April–3 May 2018 .
19. Athalye, A.; Carlini, N.; Wagner, D. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In Proceedings of the 35th International Conference on Machine Learning, Vienna, Austria, 25–31 July 2018; Volume 80, pp. 274–283.
20. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.
21. Isola, P.; Zhu, J.Y.; Zhou, T.; Efros, A.A. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–28 July 2017; pp. 1125–1134.
22. Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; Sheng, Y. Malware traffic classification using convolutional neural network for representation learning. In Proceedings of the 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 11–13 January 2017; pp. 712–717.
23. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297.
24. Ho, T.K. Random decision forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 14–16 August 1995; Volume 1, pp. 278–282.
25. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 3320–3328.
26. Dong, Y.; Pang, T.; Su, H.; Zhu, J. Evading defenses to transferable adversarial examples by translation-invariant attacks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 July 2019; pp. 4312–4321.
27. Pan, Z.; Yu, W.; Yi, X.; Khan, A.; Yuan, F.; Zheng, Y. Recent progress on generative adversarial networks (GANs): A survey. *IEEE Access* **2019**, *7*, 36322–36333.
28. Tolosana, R.; Vera-Rodriguez, R.; Fierrez, J.; Morales, A.; Ortega-Garcia, J. DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection. *arXiv* **2020**, arXiv:2001.00179.
29. Engel, J.; Agrawal, K.K.; Chen, S.; Gulrajani, I.; Donahue, C.; Roberts, A. Gansynth: Adversarial neural audio synthesis. *arXiv* **2019**, arXiv:1902.08710.
30. De Cao, N.; Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *arXiv* **2018**, arXiv:1805.11973.
31. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards deep learning models resistant to adversarial attacks. In Proceedings of the International Conference on Learning Representation ICLR, Vancouver, BC, Canada, 30 April–3 May 2018.
32. Garcia, S.; Grill, M.; Stiborek, J.; Zunino, A. An empirical comparison of botnet detection methods. *Comput. Secur.* **2014**, *45*, 100–123.
33. Papernot, N.; Faghri, F.; Carlini, N.; Goodfellow, I.; Feinman, R.; Kurakin, A.; Xie, C.; Sharma, Y.; Brown, T.; Roy, A.; et al. Technical Report on the CleverHans v2.1.0 Adversarial Examples Library. *arXiv* **2018**, arXiv:1610.00768.