Efficient Utilization of Adversarial Training towards Robust Machine Learners and its Analysis

(Invited Paper)

Sai Manoj P D
George Mason University
Fairfax, VA
spudukot@gmu.edu

Sairaj Amberkar
George Mason University
Fairfax, VA
samberka@gmu.edu

Setareh Rafatirad
George Mason University
Fairfax, VA
srafatir@gmu.edu

Houman Homayoun
George Mason University
Fairfax, VA
hhomayou@gmu.edu

ABSTRACT

Advancements in machine learning led to its adoption into numerous applications ranging from computer vision to security. Despite the achieved advancements in the machine learning, the vulnerabilities in those techniques are as well exploited. Adversarial samples are the samples generated by adding crafted perturbations to the normal input samples. An overview of different techniques to generate adversarial samples, defense to make classifiers robust is presented in this work. Furthermore, the adversarial learning and its effective utilization to enhance the robustness and the required constraints are experimentally provided, such as up to 97.65% accuracy even against CW attack. Though adversarial learning's effectiveness is enhanced, still it is shown in this work that it can be further exploited for vulnerabilities.

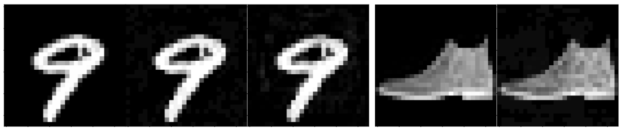
ACM Reference Format:
Sai Manoj P D, Sairaj Amberkar, Setareh Rafatirad, and Houman Homayoun. 2018. Efficient Utilization of Adversarial Training towards Robust Machine Learners and its Analysis. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '18), November 5–8, 2018, San Diego, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3240765.3267502

1 INTRODUCTION

Machine learning (ML), especially the deep neural networks (DNNs) and the convolutional neural networks (CNNs) have transformed the processing capabilities of the present day computing systems. These techniques are widely deployed in different domains ranging from computer vision to hardware security. For instance, autonomous driving is envisaged due to the advancements in the field of ML and computer vision [1–3]. Similarly, ML has made its impact on malware and side-channel attack detection towards securing the computing systems [4–8]. Despite the benefits offered by the advancements in the ML, it has also been exploited for the vulnerabilities in the existing ML techniques.

Though the ML techniques are shown to be robust to the noise, the exposed vulnerabilities have shown that the outcome of the ML can be manipulated by adding crafted perturbations to the input data [9–12], often referred as *Adversarial samples.* These adversarial samples are constructed by perturbing the input in one or multiple cycles iteratively under certain constraints in order to amplify the classification error rate.

A simple adversarial sample generated from the MNIST digit dataset [13] for digit '9' is shown in Figure 1. The Figure 1(a) is the normal sample which is classified as 9 by the DNN classifier, presented in Section 3. The images in Figure 1(b), (c) are generated by the fast gradient sign method (FGSM) and Carlini Wagner (CW) attack techniques, respectively. One can observe from the Figure 1(a), (b) and (c) that the normal and adversarial samples look similar for human observation. It needs to be noted that the noise in Figure 1 (b) and (c) can be increased or reduced by tuning the parameters of the attack. With the change in attack parameters, the classifier output and it's confidence will be modified. More details on generating the adversarial attacks are presented in Section 2.1, and the details regarding the classifier architecture and the dataset are presented in Section 3.



(a) Normal MNIST Digit '9' (b) FGSM - classified as Digit '4' (c) CW - classified as Digit '4' (d) MNIST image of a 'shoe' (e) FGSM - classified as Sneaker

Figure 1: (a) A normal MNIST Digit image classified correctly; (b) FGSM generated adversarial sample for image in (a); (c) CW generated adversarial sample for image in (a); (d) Normal MNIST Fashion image classified as shoe; (e) FGSM generated image of (e) classified as sneakers

Though the term *adversarial samples* in the context of ML is introduced in the recent few years, similar concepts date back to 2004 [14] in the context of spam filtering. The work in [15] has shown that the linear classifiers can be fooled by crafted modifications in the content of spam emails to classify them as normal emails. Similar work on biometric recognition fooling is proposed in [16]. The adversarial attacks can be broadly classified into two

ICCAD '18, November 5–8, 2018, San Diego, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5950-4/18/11...$15.00
https://doi.org/10.1145/3240765.3267502

categories: (a) poisoning attacks and (b) evasion attacks. Poisoning attacks are attacks on the ML classifier during the training phase [17–21], and the evasive attacks are targeted for inference stage of ML techniques. As the poisoning attacks focus on attacking the classifiers during training phase it is more suitable for online environments. Thus, this work focuses on the evasive attacks, as many of the existing ML works are primarily offline learning based and are constrained by resources and the processing time requirements.

In this work, we first provide an overview of evasive attacks on the ML classifiers. Further, we present different existing defense techniques for the adversarial attacks. As FGSM is one of the fastest evasive attacks, an in-depth discussion regarding the FGSM adversarial attack is provided. In this work, we look at initially introduced defense against adversarial samples, Adversarial training is one of the defense techniques introduced for adversarial attacks. Adversarial training [22] is similar to a brute force solution, where one generates an ample number of adversaries and train the classifier to alleviate the impact of adversarial attacks. Though, adversarial training is shown to be confined to be efficient for one or few attacks, it is not always the case. Further, in this work, we show how to efficiently utilize the adversarial training in order to enhance the robustness of the ML classifier even against the recent and powerful adversarial attacks such as CW. Contrarily, we also provide the information regarding under which circumstances this robust adversarial training fails. We show that adversarial training with FGSM can show high robustness to even CW attacks, under certain conditions by having a classification accuracy of up to 97% against adversarial attacks. Having said this, it fails when the number of binary steps as well as the number of iterations are increased , in a nutshell if the attacker has more computational capability

The rest of this paper is organized as follows. Section 2 introduces the existing adversarial attacks and defense mechanisms for the adversarial attacks. The simulation results and effective ways to utilize the adversarial training and its analysis are presented in Section 3 with conclusions drawn in Section 4.

2 BACKGROUND

Adversarial samples are the samples that are generated by introducing crafted perturbations into the normal input data generated by introducing optimum yet worst-case perturbations in order to make the adversarial data look similar to the normal input data, but still the ML model mispredicts the class with a high probability. These adversarial samples can be considered as an optical illusion for the ML classifiers. In this section, we present different techniques widely used for generating the adversarial samples, and review some of the popular defense techniques deployed.

2.1 Adversarial Attacks

We present an overview of different adversarial attacks that are effective against the ML classifiers here.

2.1.1 Fast Gradient Sign Method (FGSM).

The most common adversarial attack technique is to perturb the image with gradient of the loss with respect to the image or input, gradually increasing the magnitude of the perturbation until the image is misclassified.

Fast Gradient Sign method (FGSM) [10] is one of the first known adversarial attacks. The complexity to generate FGSM attack is lower compared to other adversarial attacks, even against deep learning models. Some of the advantages of this technique are its low complexity, fast implementation. Consider a ML classifier model with θ as the parameter, x being the input to the model, and y is the output for a given input x, and $L(\theta, x, y)$ be the cost function used to train the neural network. Then the perturbation with FGSM is computed as the sign of the model's cost function gradient. The adversarial perturbation generated with FGSM [10] is mathematically given as

$$x^{adv} = x + \epsilon sign(\nabla_x L(\theta, x, y)) \tag{1}$$

where ϵ is a scaling constant ranging between 0.0 to 1.0 is set to be very small such that the variation in x (δx) is undetectable. One can observe that in FGSM the input x is perturbed along each dimension in the direction of gradient by a perturbation magnitude of ϵ. Considering a small ϵ leads to well-disguised adversarial samples. Also, a large ϵ, is likely to introduce large perturbations.

2.1.2 Basic Iterative Method (BIM).

As seen previously, FGSM adds perturbation in each of the dimension, however, no optimization on perturbations are performed. Kurakin proposed an iterative version of FGSM, called as Basic iterative method (BIM) in [23]. BIM is an extension of FGSM technique, where instead of applying the adversarial perturbation once with ϵ, the perturbation is applied multiple times iteratively with small ϵ. This produces a recursive noise on the input and optimized application of noise, given mathematically as follows:

$$\begin{aligned} x_0^{adv} &= x \\ x_{N+1}^{adv} &= Clip_{x,\epsilon}(x_N^{adv} + \epsilon sign(\nabla_x L(\theta, x_N^{adv}, y)) \end{aligned} \tag{2}$$

In the above expression, $Clip_{x,\epsilon}$ represents the clipping of the adversarial input magnitudes such that they are within the neighborhood of the original sample x. This technique allows more freedom for the attack compared to the FSGM method, as the perturbation can be controlled and the distance of the adversarial sample from the classification boundary can be carefully fine-tuned. The simulations in [23] have shown that BIM can cause higher misclassifications compared to the FGSM attack on the Imagenet samples.

2.1.3 Momentum Iterative Method (MIM).

The momentum method is an accelerated gradient descent technique that accumulates the velocity vector in the direction of the gradient of the loss function across multiple iterations. In this technique, the previous gradients are stored, which aids in navigating through narrow valleys of the model, and alleviate problems of getting stuck at local minima or maxima. This momentum method also shows its effectiveness in stochastic gradient descent (SGD) to stabilize the updates. This MIM principle is deployed in [24] to generate adversarial samples. MIM has shown a better transferability and shown to be effective compared to FGSM attack.

2.1.4 Jacobian-based Saliency Map Attack (JSMA).

In contrast to application of noise in each of the directions, Papernot in [25] proposed a simple iterative method where the forward derivative of DNN is exploited for adding the perturbations. Consider a neural network F with input x, and the output of class j, denoted by $F_j(x)$. The main principle of this work is: in order to achieve a target class t, the probability for $F_t(X)$ must be increased, on the other hand the probabilities of $F_j(X)$ for all the other classes $j \neq t$ have to be decreased, until $t = arg\ max_j F_j(X)$ is achieved. This is a targeted attack, however it can be used as an untargeted attack as well. This is accomplished by exploiting the saliency map, as defined below

$$S(X,t)[i] = \begin{cases} 0, \text{if} \frac{\partial F_t(X)}{\partial X_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j(X)}{\partial X_i} > 0 \\ (\frac{\partial F_t(X)}{\partial X_i})|\sum_{j \neq t} \frac{\partial F_j(X)}{\partial X_i}|, \text{otherwise} \end{cases} \tag{3}$$

For an input feature i starting with the normal input x, we determine the pair of features $\{i, j\}$ that maximizes $S(X,t)[i] + S(X,t)[j]$ and perturb each of the features by a constant offset ϵ. This process is repeated iteratively until the target misclassification is achieved.

2.1.5 *DeepFool Attack.*

DeepFool (DF) is an untargeted adversarial attack optimized for L_2 norm, introduced in [26]. DF is efficient and produces adversarial samples which are more similar to the original inputs as compared to the discussed adversarial samples generated by FGSM and BIM attacks. The principle of the Deepfool attack is to assume neural networks as completely linear with a hyper-plane separating each class from another. Based on this assumption, an optimal solution to this simplified problem is derived to construct adversarial samples. As the neural networks are non-linear in reality, the same process is repeated considering the non-linearity into the model. This process is repeated multiple times for creating the adversaries. This process is terminated when an adversarial sample is found i.e., misclassification happens. Considering the brevity and focus of the current work, we limit the details in this draft. However, the interested readers can refer to the work in [26] for exact formulation of DF.

2.1.6 *Carlini and Wagner Attack (CW).*

One of the most recent adversarial attacks is introduced by Carlini and Wagner in [27], popularly called as Carlini and Wagner (CW) attack. The CW attack is shown to outperform adversarial defense techniques such as defensive distillation. It is an iterative attack that finds adversarial samples against multiple defenses as compared to other attacks. At a high level, this attack is iterative using Adam optimizer and a specially chosen loss function to find adversarial examples with lower distortions than the other attack. This comes at the cost of speed as this attack is much slower than the other attacks. It encompasses a range of attacks based on the norms, all cast through the same optimization framework, thus resulting in 3 powerful attacks, that are designed employing L_0, L_2, and L_∞ norms. For the L_2 attack, which is considered in this work, the perturbation in the input i.e., δ is defined in terms of an auxiliary variable ω. The objective of the CW attack with L_2 norm can be mathematically defined as

$$\delta_i^* = \frac{1}{2}(\tanh(\omega_i + 1)) - x_i \tag{4}$$

Then, the δ_i^* which is an unrestricted perturbation is optimized over ω as follows:

$$\min_\omega ||\frac{1}{2}(\tanh(\omega) + 1) - x||_2^2 + cf(\frac{1}{2}\tanh(\omega) + 1) \tag{5}$$

Similarly, if the L_2 is considered, the optimization becomes

$$\min_\delta ||\delta||_2 + c \cdot f(x + \delta) \tag{6}$$

$$S.T.x + \delta \in [0,1]^n \tag{7}$$

where f (objective function) is defined as

$$f(x') = \max(\max\{Z(x') : i \neq t\} - Z(x') - k) \tag{8}$$

Here, $Z(x')$ is the pre-softmax output for class i, t is the target class, and k is the parameter that controls the confidence with which the misclassification occurs. The parameter k encourages the solver to find an adversarial instance x' that will be classified as class t with high confidence. The three variants of this attack were shown to be quite effective in comparison to other attacks on a network trained with defensive distillation [27].

2.2 Adversarial Defenses

Till now, different adversarial attack techniques are discussed. Here, we discuss some of the prominent existing defenses against the above discussed attacks.

2.2.1 *Adversarial Training.*

Adversarial training is one of the preliminary solutions for making the ML classifiers robust against the adversarial examples, proposed in [22]. The preliminary idea is to train the ML classifier with the adversarial examples so that the ML classifier can have adversarial information [9, 10, 26] and adapt its model based on the learned adversarial data. One of the major drawbacks of this technique is to determine what kind of attack is going to happen and train the classifier based on those attacks and determining the criticality of the adversarial component.

2.2.2 *Defensive Distillation.*

Defensive distillation is another defense technique proposed in [25], that trains the classifier using the distillation training techniques and hides the gradient between softmax layer and the pre-softmax layer. This makes it complex to generate adversarial examples directly on the network [28], as the knowledge is imparted from a bigger network during the training process. However, [27] shows that such a defense can be bypassed with one of the following three strategies: (1) choosing a more proper loss function; (2) calculation of gradients from pre-softmax layer rather than softmax layer; or (3) attack an easy-to-attack dummy network first and then transfer to the distilled network, similar to the distillation defense. The generation of adversaries can be simpler if the attacker knows the parameters and architecture of the defense network i.e., whitebox attack.

2.2.3 *MagNet.*

MagNet is proposed in [29], where a two-level strategy with detector and reformer is proposed. In the detector phase(s), the system learns to differentiate between normal and adversarial examples by approximating the manifold of the normal examples. This is performed with the aid of auto-encoders. Further, in the reformer, the adversarial samples are moved close the manifold of normal samples with small perturbations. Further using the diversity metric, the MagNet can differentiate the normal and adversarial samples. MagNet is evaluated against different adversarial attacks presented previously and has shown to be robust in [29].

2.2.4 Detecting Adversaries.

Another idea of defense proposed in the existing works is to detect adversarial examples with the aid of statistical features [30] or separate classification networks [31]. In [31], for each adversarial technique, a DNN classifier is built to classify whether the input is a normal sample or an adversary. The detector was directly trained on both normal and adversarial examples. The detector showed good performance when the training and testing attack examples were generated from the same process and the perturbation is large enough. However, it does not generalize well across different attack parameters and attack generation processes.

3 EXPERIMENTAL RESULTS

The impact of adversarial training on different attacks is analyzed here. We evaluated the accuracy on the MNIST Digit [13] and MNIST Fashion [32] datasets. The adversarial attacks are generated using Cleverhans library [33]. The source code to reproduce the experiments presented in this work can be found on github at the urlfound at bottom of this page [1].

3.1 Network Architecture

We used the ML classifier i.e., DNN architecture similar to the [34] for classifying the MNIST Digits dataset. The MNIST dataset comprises of 60,000 examples for training, and 10,000 examples for testing. On a normal classifier, we achieve an accuracy of 98.15% on MNIST Digits dataset and 89.36% on MNIST fashion dataset with the employed classifier architecture, which are close to the state-of-the-art results. More details on network architecture and configuration are presented in Table 1, and 2 respectively. For generating the adversarial attacks, we have employed the L_2 norms, and the most non-trivial parameters influencing the accuracy are reported in Table 3.

Table 1: Architectural details of employed ML classifier (DNN)

Parameter	MNIST Digits	MNIST Fashion
Input	28×28	28×28
# hidden layers	2	3
Input layer	784 neurons	784 neurons
Hidden layer 1 (ReLu)	512 neurons	512 neurons
Dropout	0.2	0.2
Hidden layer 2 (ReLu)	512 neurons	512 neurons
Dropout	0.2	0.2
Hidden layer 3 (ReLu)		512 neurons
Dropout		0.2
Output layer (Softmax)	10 neurons	

Table 2: Training parameters of the employed classifiers

Parameter	MNIST Digits	MNIST Fashion
Optimization method	ADAM	ADAM
Batch size	128	128
Epochs	20	20
Learning rate	0.001	0.001
Loss	Cross entropy	Cross entropy

Table 3: Accuracy of the classifier after and before adversarial attacks

Attack	Parameter	No attack	With attack
FGSM	$\epsilon = 0.3$	98.15%	6.59%
	$\epsilon = 0.5$	98.15%	3.09%
BIM	$\epsilon = 0.3$	98.15%	1.41%
	$\epsilon = 0.5$	98.15%	1.33%
MIM	$\epsilon = 0.3$	98.15%	1.46%
	$\epsilon = 0.6$	98.15%	1.29%
JSMA	$\theta = 0.1, \gamma = 1$	98.15%	3.60%
	$\theta = 1, \gamma = 1$	98.15%	2.26%
DF	MI[1]=10	98.15%	1.36%
	MI[1]=100	98.15%	1.29%
CW	BS[2]=10, MI[1]=300	98.15%	4.32%
	BS[2]=5, MI[1]=1000	98.15%	1.41%

[1] Maximum iterations

[2] Binary step

3.2 Performance with Adversarial Attacks

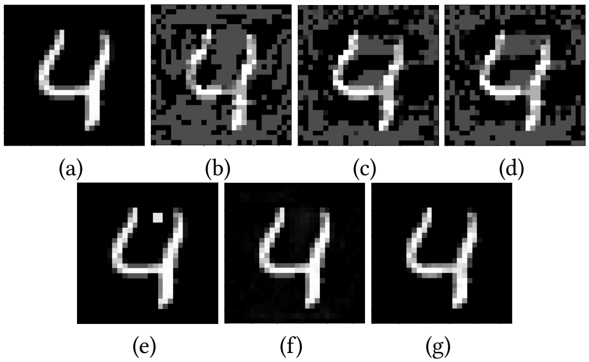


Figure 2: One of the normal MNIST Digit image is represented in (a), followed by generated adversarial images generated from (b) FGSM; (c) BIM; (d) MIM; (e) JSMA; (f) DF; and (g) CW attacks

Table 3 reports the performance of the employed neural network on MNIST Digits dataset.

Normal Classification Accuracy: In the absence of adversarial samples, the classifier achieves an accuracy of 98.15%, loss of 0.088, precision of 0.98, and recall of 0.98. Similarly, for MNIST Fashin, the classifier achieves an accuracy of 89.36%, loss of 0.3144, precision of 0.89, and recall of 0.89.

Effect of Adversaries: The adversarial samples generated from the discussed adversarial attacks are shown in Figure 2. As one can observe that the adversarial samples generated with FGSM, MIM, and BIM look alike and the adversarial samples from JSMA, DF, and CW look alike. It needs to be noted that the digit '4' is classified as '9' in all the cases. The noise in each of them can be altered, which leads to differences in the confidence of output.

Table 3 shows the accuracy of the classifier in the presence of difference attacks. The number of adversarial samples are 10,000 in each case, and one can observe that in the presence of adversaries the classification accuracy falls to as low as 1.3%. With the increase in ϵ, the accuracy decreases in case of FGSM, MIM, and BIM. With

[1] https://github.com/saimanojpd/ICCAD-18_Adversarial_Training.git

Table 4: Accuracy (%) for MNIST Digit Classification under Different Adversarial Attacks on Different Adversarial Trained Networks

	BIM			MIM			FGSM		
Adv. training with	$\epsilon = 0.3$	$\epsilon = 0.5$	$\epsilon = 0.9$	$\epsilon = 0.3$	$\epsilon = 0.5$	$\epsilon = 0.9$	$\epsilon = 0.3$	$\epsilon = 0.5$	$\epsilon = 0.9$
FGSM	96.38	89.63	48.69	97.35	94.84	60.13			
CW	62.85	44.34	34.12	63.61	45.29	30.80	51.80	32.60	26.59
JSMA	9.86	1.28	0.59	8.06	1.62	0.64	23.83	13.50	10.18
DF	53.40	31.68	25.68	54.25	32.13	22.92	41.04	24.95	18.94
MIM	99.86	97.22	71.52				87.60	61.69	40.31
BIM				99.17	91.99	76.00	84.45	53.46	34.33

	DF			JSMA (θ, γ)			CW (BS, MI)		
Adv. training with	MI=50	MI=100	MI=10	(1, 1)	(0.9, 0.9)	(1, 0.1)	(5, 1000)	(9, 200)	(10, 300)
FGSM	97.33	90.66	90.66	81.47	86.15	92.25	97.65	88.34	86.75
CW	99.70	44.74	44.74	72.36	80.83	92.27			
JSMA	92.40	6.22	6.22				93.20	88.70	85.40
DF				73.43	82.29	89.40	99.75	92.15	90.12
MIM	98.07	44.33	44.33	78.55	85.91	92.68	98.28	88.39	85.32
BIM	97.58	47.65	47.65	78.67	83.98	91.16	97.86	87.79	84.36

the number of iterations, the accuracy decreases for DF and CW attacks. The step size for each attack iteration ϵ_{iter} is set to 0.06 in the simulations. For the FGSM, with the increase in the θ, γ the attack can hamper the classification accuracy of a neural network classifier.

3.3 Effective Adversarial Training

As seen from Figure 1 and Section 2.1.1, the FGSM samples are generated by perturbing almost all the pixels in the original input. As such, the other attacks can be seen as selective tweaking of the FGSM. Thus, the adversarial training with FGSM can enhance the robustness of the classifier. However, the perturbations based on correlations and optimization might not be fully captured in FGSM samples, as there is no specific optimization involved. Here, we analyze the effect of adversarial training when the classifier is trained with the samples generated by different attacks. We consider six different attacks presented in Section 2.1 for adversarial training.

3.3.1 Performance Evaluation and Comparison.

Table 4 presents the performance (accuracy) of the employed classifier (DNN) when trained with adversarial samples generated from different attacks and tested with all the six attacks for the MNIST Digits dataset. For instance, the row with `FGSM` indicates that the classifier is trained with adversarial samples generated by FGSM attack. The classifier is provided with adversarial samples generated with the attacks mentioned in the top row of Table 4. As the training and testing with same kind of attacks have shown accuracies of nearly 99%, we have not reported them in the Table to avoid confusion and wrong analysis. The following observations can be made from the reported results in Table 4:

- FGSM, MIM, and BIM based adversarial training achieve good classification accuracy even when tested with attacks such as CW and DF.
- However, the FGSM based adversarial training outperforms MIM, and BIM. For instance, with DF attack, only FGSM based adversarial training achieves higher accuracy compared to MIM, and BIM.
- The classifier trained with CW/JSMA/DF performs better compared to normal classifier when attacked with any of the CW/JSM/DF attacks. However, the samples generated by the FGSM, MIM, and BIM still keeps the misclassification rate high.

Based on these observations and Figure 2, one can notice that the FGSM, MIM, and BIM have similar characteristics. Also, FGSM based adversarial training outperforms the others and can aid achieving robustness (to some extent) even against the most advanced (unessen) attacks such as CW, and DF. On the other hand, CW, JSMA, and DF has shown similar performances and trends. As such, when trained with one of them, it can aid to achieve robustness against the other two, compared to no defense classifiers.

In addition to MNIST Digits, we have also tested with MNIST Fashion dataset. The deployed DNN achieves an accuracy of 88%, whereas state-of-the-art network [35] achieves an accuracy of 96%, at max. However, as the major intention of the work is not on performance improvement, rather on adversarial analysis. We have performed adversarial training on the MNIST Fashion dataset as well. It has shown to follow similar trend as what is observed with MNIST Digits test case.

A glance of the results are presented below:

- With the FGSM based adversarial training, the adversarial training achieves accuracies of 80%, 84%, and 81% when the number of iterations of DF are kept 50, 100, and 10 respectively. Similar trends are obtained when tested with CW, and JSMA.
- On the other hand, when the adversarial training is performed with DF and tested with FGSM, the accuracies are 29%, 17%, and 11% with ϵ of 0.3, 0.5, and 0.9, respectively. Similar trends are obtained when tested with BIM, and MIM.

In this work, we performed the adversarial training and testing on the same kind of network, as the adversaries are generated for the same or similar network architecture as the testing network architecture. From the above analysis, it needs to be noted that the FGSM based adversarial training enhance the robustness even against unseen attacks such CW and DF.

Table 5: Pitfalls of Adversarial Training

	BIM		MIM	
	$\epsilon = 0.6$, $\epsilon_{iter} = 0.6$	$\epsilon = 0.9$ $\epsilon_{iter} = 0.6$	$\epsilon = 0.7$, $\epsilon_{iter} = 0.5$	$\epsilon = 0.7$ $\epsilon_{iter} = 0.7$
FGSM	0.7345	0.4869	0.6013	0.7352
DF	0.3044	0.2568	0.2292	0.2564
JSMA	0.0063	0.0059	0.0064	0.0066
CW	0.3971	0.3412	0.308	0.3385

3.3.2 *Pitfalls.*

Though the adversarial training, especially FGSM retrained classifier has shown robustness to the adversarial attacks, the adversarial training has some of the shortcoming in addition to what is exposed in literature. Table 5 reports the classification accuracies for MNIST Digits dataset showing the pitfalls of adversarial training based approach. Though, FGSM retrained classifier is robust to adversarial attacks caused by MIM and BIM based adversarial samples, it fails when the ϵ, that is if the perturbation is increased rapidly i.e., the magnitude of perturbations increase drastically. This can also be observed from Table 4. Similarly, the classifier trained with DF/JSM/CW fails fatally when the number of iterations are increased with additional processing capabilities. However, under certain scenarios such as maximum perturbations and if attacker has more computational power, the FGSM based retraining still has to be enhanced. As such the main pitfall of the adversarial training is its ineffectiveness to the large perturbations and increased iteration based (optimized) advanced attacks.

4 CONCLUSION

Among different adversarial attacks, FGSM is one of the fastest and less complex adversarial attacks that perturbs the input in all dimensions. Furthermore, it has been seen that the FGSM, BIM, and MIM adversarial samples look alike, and the adversarial samples generated with JSMA, CW, and DeepFool are similar. It has been seen in this work, that the adversarial training with FGSM is shown to be effective even against some of the recent attacks such as CW, and DeepFool. Adversarial learning with FGSM has achieved up to 97.65% accuracy even against CW attacks. Though the adversarial training with FGSM is seen to make the ML classifiers robust against the adversarial attacks, the FGSM adversarial training fails when the difference in perturbation between trained and tested adversarial samples are seen to be very large.

REFERENCES

[1] M. Wess, S. M. P. Dinakarrao, and A. Jantsch, "Weighted quantization-regularization in DNNs for weight memory minimization towards HW implementation," *IEEE Transactions on Computer Aided Systems of Integrated Circuits and Systems*, 2018.

[2] E. Ackerman, "How drive.ai is mastering autonomous driving with deep learning," accessed August 2018. [Online]. Available: https://spectrum.ieee.org/cars-that-think/transportation/self-driving/how-driveai-is-mastering-autonomous-driving-with-deep-learning

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *International Conference on Neural Information Processing Systems*, 2012.

[4] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," in *International Symposium on Computer Architecture*, 2013.

[5] M. Chiappetta, E. Savas, and C. Yilmaz, "Real time detection of cache-based side-channel attacks using hardware performance counters," *Appl. Soft Comput.*, vol. 49, no. C, Dec 2016.

[6] K. N. Khasawneh, M. Ozsoy, C. Donovick, N. Abu-Ghazaleh, and D. Ponomarev, "EnsembleHMD: Accurate hardware malware detectors with specialized ensemble classifiers," 2018.

[7] F. Brasser and et al, "Hardware-assisted security: Understanding security vulnerabilities and emerging attacks for better defenses," in *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, 2018.

[8] H. Sayadi, N. Patel, P. D. S. Manoj, A. Sasan, S. Rafatirad, and H. Homayoun, "Ensemble learning for hardware-based malware detection: A comprehensive analysis and classification," in *ACM/EDAA/IEEE Design Automation Conference*, 2018.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2015.

[11] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE European Symposium on Security and Privacy (Euro S&P)*, 2016.

[12] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *International Conference on Learning Representations (ICLR)*, 2017.

[13] Y. LeCun, C. Cortes, and C. J. Burges, "Mnist digit dataset," accessed August 2018. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[14] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[15] D. Lowd and C. Meek, "Adversarial learning," in *ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005.

[16] T. Matsumoto, H. Matsumoto, K. Yamada, and S. Hoshino, "Impact of artificial "gummy" fingers on fingerprint systems." vol. 26, 04 2002.

[17] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter," in *Usenix Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[18] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar, "ANTIDOTE: Understanding and defending against poisoning of anomaly detectors," in *ACM SIGCOMM Conference on Internet Measurement*, 2009.

[19] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *International Conference on Machine Learning*, 2012.

[20] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *International Conference on Machine Learning*, 2015.

[21] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *ACM Workshop on Artificial Intelligence and Security*, 2017.

[22] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: increasing local stability of neural nets through robust optimization," *ArXiv e-prints*, 2015.

[23] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *International Conference on Learning Representations*, 2017.

[24] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Neural Information Processing Systems Conference*, 2017.

[25] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE Symposium on Security and Privacy (S&P)*, 2016.

[26] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," *CoRR*, vol. abs/1511.04599, 2015.

[27] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy (SP)*, 2017.

[28] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *ArXiv e-prints*, 2015.

[29] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[30] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. D. McDaniel, "On the (statistical) detection of adversarial examples," *CoRR*, vol. abs/1702.06280, 2017.

[31] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations." in *International Conference on Learning Representations*, 2017.

[32] zalandoresearch, "Mnist fashion dataset," accessed August 2018. [Online]. Available: https://github.com/zalandoresearch/fashion-mnist

[33] N. Papernot and et al, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.

[34] keras, "Mnist model," accessed August 2018. [Online]. Available: https://github.com/keras-team/keras/blob/master/examples/mnist_mlp.py

[35] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *arXiv preprint arXiv:1708.04896*, 2017.