R^2AD : Randomization and Reconstructor-based Adversarial Defense for Deep Neural Networks

Marzieh Ashrafiamiri
mashrafi@uci.edu
University of California Irvine

Sai Manoj Pudukotai
Dinakarrao
spudukot@gmu.edu
George Mason University

Amir Hosein Afandizadeh
Zargari
amir.zargari@uci.edu
University of California Irvine

Minjun Seo
minjun.seo@uci.edu
University of California Irvine

Fadi Kurdahi
kurdahi@uci.edu
University of California Irvine

Houman Homayoun
hhomayoun@ucdavis.edu
University of California Davis

ABSTRACT

Machine learning (ML) has been widely adopted in a plethora of applications ranging from simple time-series forecasting to computer security and autonomous systems. Despite the robustness by the ML algorithms against random noise, it has been shown that inclusion of specially crafted perturbations to the input data termed as *adversarial samples* can lead to a significant degradation in the ML performance. Existing defenses to mitigate or minimize the impact of adversarial samples including adversarial training or randomization are confined to specific categories of adversaries, compute-intensive and/or often lead to reduce performance even without adversaries. To overcome the shortcomings of the existing works on adversarial defense, we propose a two-stage adversarial defense technique (R^2AD). To thwart the exploitation of the deep neural network by the attacker, we first include a random nullification (RNF) layer. The RNF nullifies/removes some of the features from the input randomly to reduce the impact of adversarial noise and minimizes attacker's feasibility to extract the model parameters. However, the removal of input features through RNF leads to a reduction in the performance of the ML. As an antidote, we equip the network with a Reconstructor. The Reconstructor primarily contributes to reconstructing the input data by utilizing an autoencoder network, but based on the distribution of the normal samples, thereby improving the performance, and also being robust to the adversarial noise. We evaluated the performance of proposed multi-stage R^2AD on the MNIST digits and Fashion-MNIST datasets against multiple adversarial attacks including FGSM, JSMA, BIM, Deepfool, and CW attacks. Our findings report improvements as high as 80% in the performance compared to the existing defenses such as adversarial training and randomization-based defense.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by classification**; **Adversarial learning**; **Neural networks**; **Image processing**.

This work is licensed under a Creative Commons Attribution International 4.0 License.

MLCAD '20, November 16–20, 2020, Virtual Event, Iceland
© 2020 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7519-1/20/11.
<https://doi.org/10.1145/3380446.3430628>

KEYWORDS

Computer vision, adversarial learning, machine learning, randomization, data augmentation

ACM Reference Format:

Marzieh Ashrafiamiri, Sai Manoj Pudukotai Dinakarrao, Amir Hosein Afandizadeh Zargari, Minjun Seo, Fadi Kurdahi, and Houman Homayoun. 2020. R^2AD : Randomization and Reconstructor-based Adversarial Defense for Deep Neural Networks. In *2020 ACM/IEEE Workshop on Machine Learning for CAD (MLCAD '20), November 16–20, 2020, Virtual Event, Iceland*. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3380446.3430628>

1 INTRODUCTION

Application of machine learning (ML), especially deep learning, has been widely adopted in a plethora of applications such as image processing [23], hardware security [2, 6, 19], natural language processing [25] and autonomous systems [9]. Deep learning has showcased an appreciable performance in these applications and has also shown resilience against random noise in the input. Despite robustness against random noise, the recent works have shown that the machine learning techniques are vulnerable to specially crafted perturbations, adversarial samples [5, 7, 12, 17, 21]. The perturbed samples include a minimal amount of noise that can not even be seen by human's naked eyes nor affect the recognition capability by naked human eyes, but can be misclassified by the ML classifiers.

To alleviate the misclassification caused by the adversarial perturbations, multiple defenses have been proposed in the literature. Some of the prominent defenses are: a) adversarial training [20] - train the classifier with adversarial examples; b) adversarial sample detection [8, 14] - detect the normal and adversarial samples; c) defensive distillation [18] - blocks the attacker from obtaining the loss gradient, thus, decreasing the feasibility to attack; d) random nullification features [22] - utilize random nullification layer as a defense against adversaries; and e) digitization [3] - techniques such as thermometer encoding to minimize the sensitivity exploitation.

Despite the elevated robustness against the adversarial samples, these techniques suffer from setbacks. For instance, adversarial training is confined to a particular attack; techniques such as random nullification features and digitization, though effective on adversaries, shows a negative performance on normal (benign) samples. We perform a case study to showcase the impact of random nullification on different adversarial attacks. Experimental details and network parameters are mentioned in Section 4.1. But, in this experiment, we use randomized inputs both for training and testing phase (see Section 3.1 for further information).

Table 1: Comparison of neural network performance with and without randomization

	MNIST digits dataset	
	clean dataset	FGSM ($\epsilon=0.3$)
Accuracy of Network	99.21%	27.77%
Accuracy of network with traditional random nullification	98.87%	66.36%

As a case study, we have implemented the random nullification layer as the pre-processing of the input to alleviate the impact of adversaries. We have implemented fast sign gradient method (FGSM) attack to craft the adversaries. The hyper-parameter needed for FGSM is set to 0.3 for this case study. As seen from Table 1, despite randomization technique aids in eluding the adversarial attacks, performance degradation is observed in the case of normal samples. Thus, it is of a dire need to devise a technique that is effective against adversarial attacks and, at the same time, maintains the performance in the event of standard samples. To address the shortcomings of existing works and address the challenges as mentioned above, this work introduces R^2AD - randomization+reconstructor-based adversarial defense.

The proposed R^2AD is a multi-stage approach, where the random nullification layer (RNF) is initially designed to randomly remove the pixels from the input samples (normal or adversarial), thus reducing the impact of adversarial noise. However, such a random removal of pixels (input features) might affect normal/benign pixels, resulting in a degraded performance. To mitigate such degradation, we introduce an autoencoder-based reconstructor network, which aids in reconstructing the input (image in our case) with the same distribution as normal samples. It needs to be noted that the reconstructor network is trained with benign data, thus reconstructs the input whose data distribution is close to the original/benign samples. We evaluated the proposed R^2AD on MNIST digits [11] and Fashion-MNIST [24] datasets with different attacks ranging from FGSM [7] to recently proposed Carlini & Wagner (C&W) [4] attacks. The rationale to showcase for this dataset is that this is one of the simplest datasets that can be adversarially perturbed even with minimal noise compared to other data.

The contributions of this work can be outlined in a three-fold manner as follows:

- We introduce a random nullification layer as a part of data pre-processing to decrease the impact of adversarial perturbations and reduce the likelihood of attackers obtaining the classifier model.
- An autoencoder-based reconstructor is introduced during inference for reconstructing input and eventually leading to better performance against normal and adversarial samples.
- Accuracy improvement for both adversarial and normal samples along with minimized adversarial impact through the proposed R^2AD

The rest of this paper is organized as follows: Section 2 introduces the relevant background regarding the existing defenses against adversarial attacks. Section 3 describes the proposed R^2AD with more information on the design of the RNF layer and the reconstructor network. The evaluation of proposed R^2AD is presented in Section 4 with conclusions drawn in Section 5.

2 BACKGROUND AND RELATED WORK

Adversarial samples are often generated by adding carefully crafted noise to the normal input samples. Here, we discuss some of the prominent existing defenses against different adversarial attacks.

2.1 Adversarial Training

Adversarial training is one of the preliminary solutions for making the ML classifiers robust against the adversarial examples, proposed in [20]. The preliminary idea is to train the ML classifier with the adversarial examples so that the ML classifier can have adversarial information [7, 15, 21] 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 MagNet

MagNet is proposed in [13], 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 autoencoders. Further, in the reformer, the adversarial samples are moved close to the manifold of normal samples with small perturbations. Further, using the diversity metric, the MagNet can differentiate the normal and adversarial samples. The MagNet is evaluated against different adversarial attacks presented previously and have shown to be robust in [13].

2.3 Randomization

Randomization based defenses can be seen effective against the evasive adversarial attacks, as they try to hide the model or lead to minimal data retrieval for crafting an adversary. For instance, the work in [22] introduces a randomization feature nullification at both training and testing phases to make the DNN models robust against the adversarial samples. This randomization can be seen as a special case of drop-out techniques. Even if the adversarial attackers manage to obtain the model, the use of randomly nullifying the features try to reduce the adversarial impact on the network by removing the features. Despite, randomization aids in combating adversaries, a performance loss is seen for normal samples, i.e., samples without adversaries, as shown in Table 1.

2.4 Detecting Adversaries

Another idea of defense proposed in the existing works is to detect adversarial examples with the aid of statistical features [8] or separate classification networks [14]. In [14], 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.

In contrast to the existing works, the proposed R^2AD utilizes the concepts of randomization to prevent the attacker from obtaining the model and reducing the impact of adversaries through random nullification features. To address the prevailing challenge of performance degradation for normal samples after random nullification,

Figure 1: Proposed Defense Network

the R^2AD utilizes a reconstructor network, i.e., autoencoder that reconstructs the input after randomization, thereby retaining the performance. It needs to be noted that the reconstructor rebuilds the input after randomization. However, special care is taken to ensure that adversaries are not reconstructed by training the reconstructor with the data distribution of original (not adversarial) data. Thus, the proposed R^2AD is capable of minimizing the impact of adversaries through random nullification technique and can retain the performance through autoencoder-based input reconstruction.

3 PROPOSED R^2AD DEFENSE

In this paper, we propose a network consisting of two types of layers random nullification layer called RNF, and an autoencoder-based reconstructor, showing in Figure 1. RNF is a layer that simply omits some features of our input data. The reconstructor layer helps the network in classifying images. In the following sections, we will describe these two layers. It needs to be noted that the reconstructor layer has the duty of denoising and rebuilding images; it just helps us to show the strength of our proposed network.

Based on the Figure 1, the defense proposed as R^2AD includes two phases, training phase and testing phase. In the training phase, we utilize the Random Nullification layer to train a neural network. Moreover, an auto-encoder based reconstructor is being trained using clean and manually-added-noise data. In the testing phase, first, reconstructed samples of test data are made by using the reconstructor network. For the next step, by using the calculated reconstruction error, R^2AD will decide if the data point is unreconstructable or not. Finally, the RNF-trained network will classify each data point which is considered as reconstructable.

3.1 Random Nullification Layer (RNF)

One of the efficient ways to defend against adversarial attacks is having random nullification layer, as described in [22]. In simple terms, this defensive method is adding an extra layer, called RNF, to the neural network. The RNF layer, with the responsibility of nullifying some features of input data, is located between the inputs and the first hidden layer in the training phase (this location is selected among all possibilities). Suppose we have some images as training inputs for our method. The RNF layer will randomly omit some of the pixels of the data points, meaning that it will set them to zero and lead them to the trained network. The number of features that are nullified in each input data is specific and follows a Gaussian distribution.

Figure 2: Process of nullification i.e., impact of three different RNF masks

Hence, the first step is to choose the parameters of the Gaussian distribution and calculate the number of features in each input which are going to be nullified. Based on the number calculated beforehand, some features are selected from a uniform distribution and are being nullified. The worth mentioning aspect of this method is that the probability of each feature chosen to be nullified is the same. So, for each input sample x , the I^x should be made which is the same size of input and have random zeros and ones in it. Then, the x^* is product of these arrays, as shown below.

$$x^* = x \odot I^x \quad (1)$$

Randomly removing pixels helps to reduce the impact of adversaries, thus added noise's impact is reduced in the existing attacks as shown in Figure 2. However, a challenge with having RNF is performance degradation, even on normal samples, due to the randomness of nullifying features which may delete some required features. We will practically demonstrate that gaining accuracy in deleting noise of adversarial example is much notable than losing it on normal samples.

There exist different feasible choices regarding when the RNF layer can be deployed: (1) only in training phase, (2) only in testing phase, and (3) in both phases. We analyze the impact of utilizing the RNF on these different choices. Table 2 demonstrates the results based on the location when the FGSM attack is implemented on MNIST digits data. As seen, using the RNF layer in both training and testing phases leads to the best performance compared to other scenarios. In this scenario, the model is adapted to feature nullification and when deploying the removing pixels to the testing phase, the model will easily obtain better performance.

Table 2: Results for different locations of RNF

Utilization of RNF	Accuracy for FGSM $\epsilon=0.3$
Only training	10.02%
Only testing	29.69%
Training and Testing	66.36%

From all the possible choices to utilize the RNF layer, we propose the one that is used for training the network. However, it has lower

performance than other scenarios. The power of RNF just in training will be flourished by using the reconstructor.

In the chosen scenario, first we will use RNF layer on our training set, means clean MNIST (both MNIST digits and Fashion-MNIST in different experiments). Then, we will train the network. Due to the training phase, our model is now adapted to considering some noise/nullification. Since dealing with adversarial data is the purpose of the network in the testing phase, the model trained with RNF shows better performance when reconstructor is added. In the Section 4, the Table 4 witness the statement. As aforementioned, while passing through the RNF layer, the pixels of the input data will be randomly nullified/omitted. These pixels chosen to be omitted are specific to one input data, and their coordinations are different among different data points. When all the data points pass through the RNF layer, the nullified data points are being used for the training phase. Hence, in all of the epochs, the nullified pixel indices are similar.

3.2 Reconstructor

One way of defending against attacks is determining the adversarial input and trying to denoise it. The noticeable advantage of the proposed function is attack-independent, unlike the previous methods like adversarial training. In other words, having just normal data is sufficient for making a reconstructor.

The reconstructor first calculates the distance between the input sample (adversarial or normal) and the denoised sample, as presented in Equation (2). Further, depending on the specified threshold, it determines whether the given sample is normal or adversarial. The threshold should neither be very large to misunderstand adversarial input to normal data nor be very small to increase the probability of deciding the normal sample as untrusted data.

For the implementation of the reconstructor layer, autoencoders are used, which consists of an encoder and a decoder as shown in Figure 3. For training the autoencoder, some random noise should be added to a clean dataset (these are not adversarial data). The noise factor we choose is equal to 0.1.

An ideal autoencoder should not change the classification result of normal data. Encoders are used in learning a latent representation of data, and decoders use these features for the reconstruction of data. Although in the case of normal inputs, the reconstruction error will be low, for adversarial inputs it leads to a high amount of error. There exist many ways to measure the reconstruction error between data and normal samples which L^1 and L^2 norms are chosen practically. For these p -norms errors, increasing p will increase the importance of maximum distance between all pixels, but decreasing p will lead to more attention on each pixel itself. If we consider $ae(x)$ be the output of autoencoder for the input x , the reconstruction error is then calculated by (2).

$$E(x) = ||x - ae(x)||_p \quad (2)$$

Based on the threshold obtained from the clear and normal input sample data, the reconstructor will decide if a sample is labeled as reconstructable or unreconstructable data. The unreconstructable sample shows so much noise that the actual data is unrecognizable. It is one of our goals to intercept such samples and prevent them from moving forward, due to the potential of downgrading the

Figure 3: Design of reconstructor

Table 3: Architecture of network and reconstructor for MNIST digits dataset

classifier		reconstructor		
Conv+Relu	32×3×3	encoder	Conv+Sigmoid	1×3×3
Conv+Relu	64×3×3		Average Pooling	2×2
Max Pooling	2×2		Conv+Sigmoid	3×3×3
Dense+Relu	128		Conv+Sigmoid	3×3×3
Dense+softmax	10	decoder	Up Sampling	2×2
			Conv+Sigmoid	3×3×3
			Conv+Sigmoid	1×3×3

performance. All less noisy data (normal or adversarial) will then go through the trained network to be classified.

Among the low noisy forwarded inputs, some are normal, and some are adversarial. Normal data as inputs should remain unmodified after going through the autoencoder. On the other hand, the autoencoder will attempt to denoise the adversarial inputs so that they resemble normal data, and the adversarial noise is filtered out. Using this technique has two important advantages. First, because the normal data remains the same, the method will not change the classification accuracy. Also, the classification accuracy of adversarial samples will increase because of denoising.

4 SIMULATION RESULTS

4.1 Experimental Setup

The performance of the proposed R^2AD is evaluated against multiple adversarial attacks (FGSM [7], BIM [10], JSMA [18], DeepFool [15] and Carlini and Wagner (CW) [4]). The experiments are executed on the Intel Xeon E3-1225 V5 processor with 16 GB RAM and also NVIDIA GeForce GTX 1660. We evaluated the MNIST digits dataset with 50000 images utilized for training the neural network architecture and 10000 utilized for testing, i.e. 80-20% data split for training and testing. We utilized Tensorflow library [1] to implement and evaluate our proposed R^2AD . Also, we have employed Cleverhans library [16] to craft the adversaries and evaluate them. For evaluation, we considered whitebox and blackbox scenarios, i.e. attack generation and testing on the same network termed as whitebox attack scenario and different networks for generation and testing for blackbox scenario. The architectural details of the neural network used for R^2AD evaluation of MNIST digits are presented in Table 3. The accuracy of classifying the MNIST digits dataset using the network presented in Table 3 is 99.21%.

As explained in Section 3, the reconstructor is made of an autoencoder. The details of autoencoder's architecture, which is used in the experiment of the MNIST digits dataset, can be found in Table 3. After denoising each sample, the reconstructor will calculate the reconstruction error. Samples with a lower error than

Table 4: Performance (accuracy (%)) against blackbox attack on MNIST digits dataset

Variety of MNIST digits dataset		Blackbox attack			
Attack	Attack Parameter	No Defense	Random Nullification	Reconstructor	R^2AD
FGSM	0.3	27.77	10.02	90.74	91.90
BIM	0.3	23.13	9.87	91.53	92.87
JSMA	0.1	61.93	28.72	92.45	93.04
Carlini	5	22.01	19.71	68.95	73.68
DeepFool	0.1	41.95	23.05	96.39	96.95

the threshold will then go through the RNF trained network. The RNF layer, which inherits features of a normal distribution, needs some parameters to be determined before. The two most important parameters are Mean and standard deviation, which are 3 and 4 for the experiments.

4.2 Performance Against Different Attacks

As aforementioned, we evaluate the impact of adversarial attacks against the proposed R^2AD . First, let us consider the case of the blackbox scenario, where the attacker has no information regarding the network for which attack is crafted, i.e. the attacker relies on the transferable properties of the attack. Similarly, we have considered the case of the whitebox scenario, i.e. the attacker has the information regarding the classification network and can generate the attack based on that information.

4.2.1 Evaluation with MNIST Digits Dataset:

Table 4 presents the performance of the different attacks of MNIST digits data on the network defined in Table 3. The attack parameter like $\epsilon \in [0, 1]$ for FGSM indicates the amount of perturbation added to the input. For the scenario of blackbox attack: as one can observe that without any defense, the performance of classification reduces to 27.7%. Similarly, when only utilizing Random Nullification Layer (RNF) as defense [22], the performance reduces to 10.02%, where the loss is due to the loss of information. Similarly, we replicated the work presented in 3 where the reconstructor is used. As seen, the utilizing of the RNF also improved the performance against all attacks, when using with the reconstructor. The rationale for this is that the FGSM is similar to a brute force kind of approach in terms of adding the noise and can be easily circumvented compared to the rest of the adversarial attacks.

In the case of BIM attack, as it can be seen clearly from Table 4, the attack samples are transferable - as the accuracy drops to 23.13% on an average when no defense is deployed. Similar behavior is observed in this case, which is an iterative version of FGSM. Compared to the simple usage of the RNF layer [22] or utilizing reconstructor [13], the proposed R^2AD yields better performance.

JSMA [18] is an advanced attack compared to the FGSM and BIM, and is a more sophisticated attack. For JSMA, the perturbation is determined by the parameter η . We evaluate the performance of the proposed R^2AD , reconstructor only and [22] (RNF only) techniques, whose performances are listed in Table 4. As seen that without defense, the adversaries lead to a performance of 61.93%. When employing defense like [22], the performance decreases to 28.72% in the case of blackbox attack. In such scenarios, as the adversaries are transferable, if the attack is generated on a different network, the performance of the reconstructor is lower compared to the proposed R^2AD . Similar behavior is observed with the DeepFool attack, as showcased in Table 4.

Lastly, we evaluate the performance against the CW attack, which is the advanced adversarial attack so far. The parameters

Table 5: Performance (accuracy (%)) against whitebox attacks on MNIST digits dataset

Attack	RNF Parameters		whitebox attack			
	Mean	Standard Deviation	No Defense	Random Nullification	Reconstructor	R^2AD
FGSM	7	5	27.77	19.7	90.74	92.28
FGSM	7	10	27.77	23.98	90.74	92.59
FGSM	5	10	27.77	40.66	90.74	93.05
Carlini	7	5	22.01	23.03	68.95	76.15
Carlini	7	10	22.01	41.28	68.95	77.85
Carlini	5	10	22.01	38.55	68.95	76.01

utilized for providing perturbations(confidence) is set to 5. Table 4 shows the performance with CW attack utilizing L_2 norm. As one can observe that without deploying any defense, the performance is 22.01%. However, using the RNF layer can lower the performance. Similarly, the reconstructor shows a good performance; however, it is lower than the proposed R^2AD . Also, it needs to be noted that Table 4 shows the performance for the blackbox attack when using MNIST digits dataset.

As mentioned in the previous part, the experiments divide into two major parts. First of all, we evaluated the robustness of the proposed defense network against blackbox attacks. For assessing the effectiveness of this method, we craft MNIST digits data with five different attacks. Additionally, for FGSM and CW attacks, we try to measure the accuracy of the suggested method for whitebox attacks as well. In this scenario, the amount of perturbation is related to the mean and standard deviation of the RNF layer. In other words, the parameters of the RNF layer are known for the attacker. The parameters used for FGSM and CW in Table 5 are 0.3 and 5, respectively.

4.2.2 Evaluation with Fashion-MNIST Dataset:

The noticeable efficiency of the proposed network on the MNIST digits dataset persuades us to investigate more on the correctness of R^2AD . To prove the effectiveness of R^2AD , we repeat the same experiment on the more sophisticated dataset named Fashion-MNIST. Fashion-MNIST consists of 10 labels like MNIST digits; but instead of hand-written digits, the labels are regarded to fashion like shoes and shirts. By comparing the accuracies for the network without defense and our proposed defense method (R^2AD) based on the information given in Table 4, FGSM and BIM are considered as the two top attacks with the most increase in their accuracies. Hence, we decided to use these two attacks with the Fashion-MNIST dataset. Table 6 contains the architectural details of the neural network used for the evaluation of R^2AD . Using the network presented in Table 6 will result in accuracy equal to 77.91% when evaluating Fashion-MNIST dataset. As mentioned in the previous sections, an autoencoder is the skeleton of the reconstructor, which is used in our proposed defense technique. The architectural details of the autoencoder used for Fashion-MNIST dataset are also in Table 6.

The accuracy of our proposed network and other methods for the Fashion-MNIST dataset are presented in Table 9. The amount of perturbation added to the Fashion-MNIST dataset in both attacks is equal in comparison to these numbers for the MNIST digits dataset. In other words, the attack parameter for FGSM and BIM are considered as 0.3 and 0.3, respectively. For the FGSM attack, the accuracy of classification without any performance is 22.60%. In comparison to the MNIST digits, each image in the Fashion-MNIST dataset includes more pixels with valuable information. Therefore, nullifying some attacked pixels will decrease the amount of loss which is caused by the attack. Hence, using the random nullification

Table 6: Architecture of classifier and reconstructor for Fashion-MNIST dataset

classifier		reconstructor	
Conv+Relu	32×3×3	encoder	Conv+Relu 64×3×3
Max Pooling	2×2		Max Pooling 2×2
Dropout	0.25		Conv+Relu 32×3×3
Conv+Relu	64×3×3		Max Pooling 2×2
Max Pooling	2×2		Conv+Relu 16×3×3
Dropout	0.25	decoder	Max Pooling 2×2
Conv+Relu	128×3×3		Conv+Relu 16×3×3
Dropout	0.4		Up Sampling 2×2
Dense+Relu	128		Conv+Relu 32×3×3
Dropout	0.3		Up Sampling 2×2
Dense+softmax	10		Conv+Relu 64×3×3
			Up Sampling 2×2
			Conv+Relu 1×3×3

Table 7: Performance (accuracy (%)) against blackbox attack on Fashion-MNIST dataset

Variety of Fashion-MNIST dataset		Blackbox attack			
Attack	Attack Parameter	No Defense	Random Nullification	Reconstructor	R^2AD
FGSM	0.3	22.60	28.20	54.42	59.12
BIM	0.3	15.91	24.63	55.62	60.48

layer will raise the accuracy around 6%. Similarly, we replicated the work presented in 3 where the reconstructor is used. Utilizing the reconstructor itself helps the network against the FGSM attack, and the performance reaches 54.42%. Lastly, the accuracy with applying the proposed R^2AD , will increase the accuracy to 59.12%.

Using the BIM attack on the Fashion-MNIST dataset will result in a similar pattern when compared to the FGSM attack. The accuracy of bare-network will grow when deploying the random nullification layer. Among the four existing techniques, our results prove that the performance of our proposed R^2AD will defeat others in the Fashion-MNIST dataset as well.

The worth-mentioning point in all the experiments about the proposed reconstructor is that it has a very low rate of false negatives, which means that among discarded images (detected as adversarial data) only a few of them will be classified correctly by the network. Experiments show that in the lower amount of perturbation, this number is equal to zero. The more amount of perturbation leads to more number of false negatives, which will be as low as 15%.

5 CONCLUSIONS

In this paper, we proposed a method to defend against adversarial attacks that are intelligently crafted. To address this, we propose a 2 layer defense. Random Nullification will first try to minimize the effect of adversaries for the trained network. To address the impact of removed features by the random nullification layer, an autoencoder based reconstructor is introduced to rebuild the input features and ensure the rebuilt features follow the data distribution of normal samples. This reconstruction minimizes the impact of randomly nullified features. We advocate that the impressive defense against adversaries should not rely on any data from the process of generating adversarial data. In other words, it should be attack-independent. R^2AD is evaluated under whitebox and black-box scenarios and have shown high robustness against adversaries and higher performance on normal samples. Experimental evaluation with different levels of noise is performed and compared with some of the recent defenses. It is seen that with the proposed R^2AD

up to 80% higher performance is achieved compared to some of the recently proposed defenses.

ACKNOWLEDGMENT

This research was funded by the National Science Foundation through award CCF-1704859.

REFERENCES

- [1] Martin Abadi and et al. 2016. Tensorflow: A system for large-scale machine learning. In *USENIX Symposium on Operating Systems Design and Implementation*.
- [2] Ferdinand Brasser and et al. 2018. Hardware-Assisted Security: Understanding Security Vulnerabilities and Emerging Attacks for Better Defenses. In *Conf. on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*.
- [3] J. Buckman, A. Roy, and I. Goodfellow. 2018. Thermometer Encoding: One Hot Way to Resist Adversarial Examples. In *Int. Conf. on Learning Representations*.
- [4] N. Carlini and D. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy (SP)*.
- [5] S. M. P. Dinakarrao and et al. 2018. Enhancing Adversarial Training towards Robust Machine Learners and its Analysis. In *Int. Conf. on Computer-Aided Design*.
- [6] S. M. P. Dinakarrao and et al. 2019. Adversarial Attack on Microarchitectural Events based Malware Detectors. In *Design Automation Conf.*
- [7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations (ICLR)*.
- [8] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick D. McDaniel. 2017. On the (Statistical) Detection of Adversarial Examples. *CoRR abs/1702.06280* (2017).
- [9] R. S. Gutzwiller and J. Reeder. 2017. Human interactive machine learning for trust in teams of autonomous robots. In *IEEE Conf. on Cognitive and Computational Aspects of Situation Management*.
- [10] A. Kurakin, I. J. Goodfellow, and S. Bengio. 2017. Adversarial examples in the physical world. In *International Conference on Learning Representations*.
- [11] Yann LeCun and et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
- [12] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into Transferable Adversarial Examples and Black-box Attacks. In *International Conference on Learning Representations (ICLR)*.
- [13] Dongyu Meng and Hao Chen. 2017. MagNet: a Two-Pronged Defense against Adversarial Examples. In *ACM Conf. on Computer and Communications Security*.
- [14] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations.. In *Int. Conf. on Learning Representations*.
- [15] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2015. DeepFool: a simple and accurate method to fool deep neural networks. *CoRR abs/1511.04599* (2015).
- [16] Nicolas Papernot and et al. 2018. Technical Report on the CleverHans v2.1.0 Adversarial Examples Library. *arXiv preprint arXiv:1610.00768* (2018).
- [17] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. 2016. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy (Euro S&P)*.
- [18] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. 2016. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*.
- [19] H. Sayadi, N. Patel, P. D. Sai Manoj, A. Sasan, S. Rafatirad, and H. Homayoun. 2018. Ensemble Learning for Hardware-based Malware Detection: A Comprehensive Analysis and Classification. In *ACM/EDAA/IEEE Design Automation Conference*.
- [20] Uri Shaham, Yutaro Yamada, and Sahand Negahban. 2015. Understanding adversarial training: increasing local stability of neural nets through robust optimization. *ArXiv e-prints* (2015).
- [21] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*.
- [22] Qinglong Wang, Wenbo Guo, Kaixuan Zhang, Alexander G. Ororbia, II, Xinyu Xing, Xue Liu, and C. Lee Giles. 2017. Adversary Resistant Deep Neural Networks with an Application to Malware Detection. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- [23] M. Wess, S. M. P. Dinakarrao, and A. Jantsch. 2018. Weighted Quantization-Regularization in DNNs for Weight Memory Minimization Toward HW Implementation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 37, 11 (Nov 2018), 2929–2939.
- [24] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).
- [25] T. Young, D. Hazarika, S. Poria, and E. Cambria. 2018. Recent Trends in Deep Learning Based Natural Language Processing. *IEEE Computational Intelligence Magazine* 13, 3 (Aug 2018), 55–75.