Robust Machine Learning against Adversarial Samples at Test Time

Jing Lin
ICNS Lab and Cyber Florida
University of South Florida
Tampa, FL. USA
jinglin@mail.usf.edu

Laurent L. Njilla
Cyber Assurance Branch
U.S. Air Force Research Laboratory
Rome, New York, USA
laurent.njilla@us.af.mil

Kaiqi Xiong
ICNS Lab and Cyber Florida
University of South Florida
Tampa, FL. USA
xiongk@usf.edu

***Abstract*—Though the performance of deep learning is remarkable, recent works have shown that deep learning models are vulnerable to adversarial samples that are close to their original samples to human eyes but misclassified by Deep Neural Network (DNN). This is a serious problem as many deep learning models are used in physical infrastructures and critical application domains such as medical diagnosis, self-driving cars, malware detection, as well as digital assistants like Google Assistant, Alexa, and Siri. Many researchers have attempted to secure neural networks through techniques such as defensive distillation and adversarial retraining. Nevertheless, many of these techniques are ineffective to new or slightly strong adversarial attacks such as the Carlini and Wagner (C&W)'s attack. In this paper, we propose a robust adversarial retraining method to iteratively retrain a given model so that it can not only detect the adversarial examples but also maintain the prediction accuracy for the normal dataset. Our experimental results show that the prediction accuracy on the MNIST test set is maintained while the accuracies under FGSM, C&W, and DeepFool attacks increase from 29% to 91%, 7% to 70%, and 29% to 91%, respectively.**

***Index Terms*—Machine Learning, adversarial examples, deep learning (DL), deep neural network (DNN), security**

I. Introduction

The performance of deep learning is remarkable in a variety of domains such as image classification, natural language processing, and machine translation. However, recent studies have shown that machine learning algorithms can be easily fooled by techniques such as Fast Gradient Sign Method (FGSM) [1] and PGD attack [2]. For instance, Figure 1 shows digit 0 through 9 from the MNIST dataset. The original images correctly classified by a classifier are shown in the first row, whereas the second and third rows contain the adversarial images generated by the BIM attack (sometimes also called the PGD attack [2]) and the C&W attack [3], respectively. The predicted labels for those adversarial images are all wrong, though images are similar to the original images (row 1). This is a serious problem as many deep learning techniques are used in security-sensitive and/or safety-critical applications such as medical diagnosis, malware detection [4], self-driving cars [5], as well as digital assistants like Google Assistant, Alexa and Siri. For instance, PDFrate and Hidost are two machine learning classifiers that are based on the random forest and support vector machine (SVM), respectively. Xu *et al*. [4] used genetic programming to generate adversarial examples that can bypass those malware classifiers at 100% of the time. That is, they can modify the malware slightly so that it can evade these classifiers with a 100% success rate. The modification makes those PDF malware classifiers useless. Similarly, an adversarial example can be generated against the semantic segmentation task of a self-driving car by hiding pedestrians. Without noticing the pedestrians on the crosswalk, a self-driving car will not stop and can cause a deathly accident [5]. This is a serious safety problem, which must be addressed.

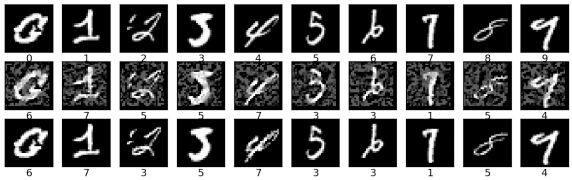

Fig. 1. **Adversarial images illustration using MNIST dataset.** The first row shows the selected original images from the MNIST dataset and their corresponding predicted labels. The second and the third rows show the adversarial images generated by the BIM attack and the Carlini and Wagner (C&W) attack, as well as their corresponding predicted labels, respectively.

Current defense methods against adversarial examples can be broadly classified into two categories, proactive and reactive defenses [6]. Proactive defenses make deep learning models more robust against adversarial examples before an adversary actually attacks it, whereas reactive defenses try to deal with adversarial examples after a deep neural network (DNN) is built. The proactive defenses can be further classified as adversarial retraining [1], [7], defensive distillation [8], and classifier robustifying [9]. Similarly, reactive defenses can be further classified as adversarial detection [10], input reconstruction [11], and network verification [12], [13]. However, all defenses are shown to be either effective only for some attacks or cannot be applied to large networks. For instance, defensive distillation has been shown to be ineffective against the C&W attack. Reluplex, a network verification technique for neural networks with ReLU activation functions, is computationally infeasible for large-scale networks. It only works for networks

DISTRIBUTION A. Approved for public release. Distribution unlimited. Case Number 88ABW-2020-0836. Dated 01 Mar 2020

978-1-7281-5089-5/20/$31.00 ©2020 IEEE

with a few hundred neurons and could only handle the L_∞ norm as a distance metric. In this paper, we introduce a proactive approach to iteratively retrain a classifier to predict the correct label of the adversarial example while preserving the classifier's prediction accuracy for normal images.

Threat model

Machine learning techniques are increasingly used in security-critical applications, such as adblockers [14] and malware detection [4]. However, the existence of adversarial examples limits the application of real-world machine learning, especially to security-critical domains. In this paper, we consider the test time attack and assume that

- a powerful adversary has full access to our trained model, but she/he has no ability to modify the parameters. More precisely, we assume that an adversary has feature knowledge and algorithm knowledge, as well as has the ability to generate the adversarial examples and input them to the classifier.
- An adversary's goal is to inject adversarial inputs that can bypass the classifier.

Mathematically, the threat model can be defined as follows. Given the feature knowledge and algorithm knowledge of a target DNN model $F \in \Gamma$ and an input $(\mathbf{x}, y)$, the adversary aims to generate an adversarial example $\mathbf{x}'$ such that

- $||\mathbf{x} - \mathbf{x}'||_2^2 < \epsilon$, where ϵ is the small perturbation that is not noticeable by human eyes.
- $\mathbf{x}' \in [0, 1]$
- $F(\mathbf{x}') = y'$, $F(\mathbf{x}) = y$, and $y \neq y'$, where y and y' are class labels.

In this paper, we propose an iterative approach to retraining a model in order to robustify it and to reduce the effectiveness of adversarial samples on it. More precisely, the retrained model can identify the adversarial example with a high success rate while maintaining the classifier's prediction accuracy for the natural instance. The key contributions of this paper are in the following:

1) We propose an ensemble model that outputs the final label based on the labels provided by two tests, one with added small random noise and the other without it.
2) Different from other studies, we divide the adversarial examples to two categories depend on the sizes of their perturbation introduced. For smaller perturbation $\eta < \eta_0$ that is smaller than some application-specific perturbation limit η_0, we train our classifier to label the instance with its original label. In this way, we force DNN classifier to learn the most important features or characteristics while ignoring the unimportant ones.
3) We propose an iterative process that is highly parallelizable. The classifier will not only be robust against adversarial examples but also maintains the accuracy of the original classifier.

The remainder of this paper is organized as follows. In section II, we provide necessary background information about this research. Section III discusses existing defense methods against adversarial examples. In section IV, we describe in detail our proposed approach, followed by an evaluation in section V. Section VI concludes this paper.

II. Background

This section provides a brief introduction to neural networks and adversarial examples. Existing defense methods are discussed in the related work section.

A. Neural Networks

Machine learning automates the tasks of writing rules for a computer to follows. That is, giving an input and an desired outcome, machine learning can find a set of rules needed automatically. Deep learning is a subset of machine learning that automates its feature selection process. That is, you do not even need to specify features, and a neural network can extract them from raw input data. For instance, in the image classification, an image is an input to the neural network, and the convolutional layers in the neural network will extract important features from the image directly. The characteristics makes deep learning desirable for many complex tasks, such as image classification and natural language processing, where software engineers have difficulty in writing rules for a computer to learn.

The performance of the neural network is remarkable in domains such as image classification, natural language processing, and machine translation. By the Universal Approximation Theorem, any continuous function in a compact space can be approximated by a feed-forward neural network with at least one hidden layer and the suitable activation function to any desired accuracy. [15]. This theorem explains the broad applicability of deep neural networks. However, it does not give any constructive guidelines on how to find such universal approximator. In 2017, Lu *et al.* [16] established the Universal Approximation Theorem for Width-Bounded ReLU Networks. These Universal Approximation Theorems explain the ability of a neural network to learn.

A feed-forward neural network can be written as a function $F : \mathbf{X} \rightarrow \mathbf{Y}$, where $\mathbf{X}$ is an input space or a sample space, and $\mathbf{Y}$ is its output space. If the task is classification, $\mathbf{Y}$ is the set of discrete classes. If the task is regression, $\mathbf{Y}$ is a subset of R^n. For each instance $\mathbf{x} \in \mathbf{X}$, $F(x) = f_L(f_{L-1}(...(f_1(\mathbf{x}))))$, where $f_i(\mathbf{x}) = A(\mathbf{w}_i * \mathbf{x} + \mathbf{b})$; $A(.)$ is an activation function, e.g., non-linear Rectified Linear Unit (ReLU), sigmoid, softmax, identity, and hyperbolic tangent (tanh); $\mathbf{w}_i$ is a matrix of weight parameters; $\mathbf{b}$ is a bias unit; and L is the total number of layers in a neural network. For classification, the activation function for the last layer is usually a softmax function. The key to the state of the art performance of the neural network is the optimized weight parameters that minimize a loss function J, a measure of the difference between the predicted output and its true label. A standard method used to find optimal weights is the back-propagation algorithm. See, e.g., [17] and [18] for an overview.

B. Adversarial Example

An adversarial example/sample $\mathbf{x}'$ is an adversarially generated sample that is close to natural sample $\mathbf{x}$ to human eyes but misclassified by a DNN model [19]. Adversarial attacks for machine learning models can either be untargeted or targeted. In an untargeted attack, an attacker does not have a specific target label in mind when trying to fool a DNN model. On the contrary, an attacker tries to mislead a DNN model to classify an adversarial sample $\mathbf{x}'$ as a specific target label t in a targeted attack. In 2013, Szegedy *et al.* [19] first generated such an adversarial example using Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS). Given a natural sample $\mathbf{x}$ and a target label $t \neq F(\mathbf{x})$, they used L-BFGS method to find an adversarial sample $\mathbf{x}'$ that satisfies the following box-constrained optimization problem:

$$\min c||\mathbf{x} - \mathbf{x}'||_2^2 + J(\mathbf{x}', t) \tag{1}$$
$$\text{subject to } \mathbf{x}' \in [0,1]^m \text{ and } F(\mathbf{x}') = t,$$

where c is a constant that can be found by linear searching, and $m = hw$ if the image is gray scaled and $m = 3hw$ if the image is colored. Because of this computational intensive linear searching method for optimal c, L-BFGS attack is time consuming and impractical. In the course of the next few years, many other adversarial attacks are proposed such as

- Fast Gradient Sign Method (FGSM)
  FGSM is a one-step algorithm that generates perturbations in the direction of the loss gradient; i.e.,

  $$\eta = \epsilon \text{sgn}(\nabla_{\mathbf{x}} J(\mathbf{x}, y))$$

  and so

  $$\mathbf{x}' = \mathbf{x} + \epsilon \text{sgn}(\nabla_{\mathbf{x}} J(\mathbf{x}, y))$$

  where ϵ is an application-specific imperceptible perturbation adversary intend to inject and $\text{sgn}(\nabla_{\mathbf{x}} J(\mathbf{x}, y))$ is the sign of the loss function [1]. Basic Iterative Method (BIM) is an iterative application of FGSM in which a finer perturbation is obtained at each iteration. This method is introduced by Kurakin *et al.* [20] for generating adversarial example in the physical world. BIM is sometimes also called Projected Gradient Descent (PGD) [2]. Iterative Least-likely Class Method (ILLC) [20] is similar to BIM but uses the least likely class as a targeted class to maximize the cross-entropy loss; Hence, this is a targeted attack algorithm.
- DeepFool
  Instead of loss gradient, DeepFool utilizes the decision boundary of the classifier. More precisely, it searches for the shortest distance to cross the decision boundary using an iterative linear approximation of the classifier and orthogonal projection of the sample point onto it [21]. This untargeted attack generates adversarial examples with smaller perturbation compared to L-BFGS, FGSM, and JSMA [3], [6]. For detail, see [21]. Universal Adversarial Perturbation is an updated version of DeepFool that is transferable [22]. It uses DeepFool method to generate minimal perturbation for each image and find the universal perturbation that satisfies the following two constraints:

  $$||\eta||_p \leq \epsilon$$
  $$P(F(\mathbf{x}) \neq F(\mathbf{x} + \eta)) > 1 - \delta$$

  where $||*||_p$ is p-norm, ϵ specifies the upper limit for perturbation, and $\delta \in [0, 1]$ is a small constant that specifies fooling rate of all the adversarial images.
- Carlini and Wagner (C&W)'s Attack
  C&W's attack [3] is introduced as a targeted attack against defensive distillation, a defense method against adversarial example proposed by Papernot *et al.* [8]. Different from equation 1, Carlini and Wagner defined an objective function f such that $f(\mathbf{x}') \leq 0$ if and only if $F(\mathbf{x}') = t$, and the following optimization problem is solved to find the minimal perturbation η:

  $$\min ||\eta||_2^2 + cf(\mathbf{x}')$$
  $$\text{subject to } \mathbf{x}' \in [0,1]^m$$

  Instead of looking for optimal c using linear searching method as in the L-BFGS attack, Carlini and Wagner just find the smallest value of c for which $f(\mathbf{x}') \leq 0$. To ensure that the box-constraint $\mathbf{x}' \in [0,1]^m$ is satisfied, they proposed three methods, projected gradient descent, clipped gradient descent, and change of variable, to avoid box-constraint. For detail, see [3]. C&W's attack is not only effective for defensive distillation but also effective for many other existing adversarial detecting defenses. For instance, in [23], Carlini and Wagner used C&W's attack against ten detection methods and showed the current limitations of detection methods.

III. Related Work

A recently published survey paper [24] summarized three main proactive countermeasures for adversarial examples: network/defensive distillation [8], adversarial (re)training [1], [7], and classifier robustifying. Network distillation used distillation, a technique usually used to reduce the dimensionality, to decrease the success rate of adversarial sample crafting [8]. Basically, this approach suggests smoothing the softmax output by training it twice. In the first time, a DNN model is trained and its softmax output is smoothed by a temperature parameter $T > 0$ using the equation

$$q_i = \frac{e^{\frac{z_i}{T}}}{\sum_{j=1}^{n} e^{\frac{z_i}{T}}},$$

where q_i is the smoothed probability of class i that will be input to the second DNN, z_i is the probability of class i output by the first DNN, and n is the total number of classes. Then, the original hard labels are replaced by the soft labels $\{q_i\}_{i=1}^n$ when training the second DNN. This way the loss function is smoother with soft targets, and so it robustfy the model. Defensive distillation is shown to reduce the success rate of JSMA attack by 98.6% and 81.36% on the MNIST

and CIFAR-10 dataset, respectively. An alternative method of network distillation is label smoothing [25]. Instead of using the first DNN to generate soft labels, they proposed just to assign a high probability, like 0.9, to the right class and then distribute the rest of probabilities among other $n-1$ classes if there is a total of n classes. For our proposed approach, we also use soft labels. However, we use soft labels not only to reduce adversarial gradients and sensitivity to input perturbation but also to prevent the network from becoming over-confident and to improve generalization and model calibration [26].

Another countermeasure is adversarial (re)training, in which an adversarial example is generated and injected into the training set for retraining [1], [7]. Usually, a base model is used to generate some adversarial samples using techniques such as FGSM, JSMA, BIM, etc. Then, these adversarial samples with the correct labels and the natural samples are mixed and used to retrain the model. However, Grosses [27] proposed a variate adversarial retraining by classifying adversarial examples as the $(n+1)^{th}$ class. Nevertheless, Tramèr et al. [28] showed that an adversarially trained model using single-step attack methods such as FGSM is still vulnerable to other simple and powerful first-step attacks and transfer attacks. Hence, we also check the accuracy of our robustified model under strong attacks in the evaluation. The result shows the performance of the model is significantly improved after retraining. The obtained model is more robust due to better generalization of the model using the soft labels and Gaussian noise for training.

IV. Methodology

The performance of a deep learning model depends mainly on data under studies, neural network architecture, and computing power. Since a neural network learns the rule from raw data, it requires a large amount of data to learn. In addition, the data representation is also important. Consider the scenario in which all data used to train a face recognition classifier are the front face of a group of students; how could you expect the classifier to perform well when a test image is a rear view of a student? Hence, the dataset needs to be big and representative. Furthermore, the neural network architecture is also essential since it can affect your training time and performance. Last but not least, it is the computation power that affects training and testing time. A back-propagation algorithm is currently used by most neural networks for training. However, it is slow and often requires thousands of epochs to learn. If it is under-trained, its performance would not be optimal. Therefore, computational power also determines the performance of a neural network model.

In this paper, we introduce an iterative approach to generating more data from the original dataset to train our proposed model. Multiple data generating methods depicted in Figure 2 are used to generate a good data representation for the model. In order to learn effectively, a model's capacity is a crucial factor. If a model is too simple, it will underfit. Hence, in the evaluation, the model used to train has a high capacity. To reduce overfitting, the dropout is added as well. Besides, we will show that our proposed approach is highly parallelable.

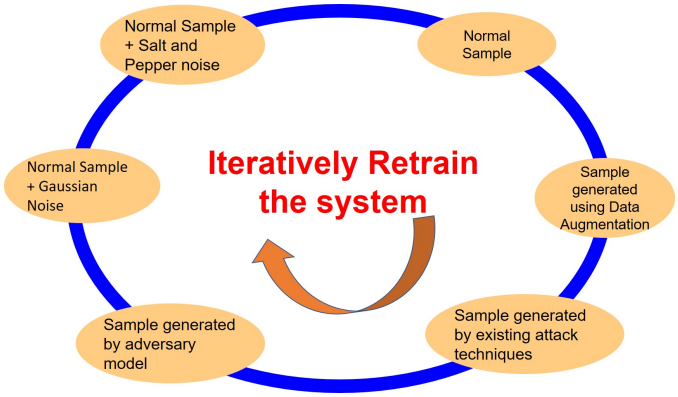

Fig. 2. Iteratively retrain the Detector+Classifier model with data generated from different techniques

As shown in Figure 2, our iterative approach is an adversarial retraining technique. However, it is different from [1] in that we use multiple strong attack techniques to generate the adversarial examples. In the background section, several existing *adversarial example crafting techniques* are introduced and can be used to generate adversarial examples for retraining. The training process can be summarized as follows:

1) Generate adversarial examples using existing *adversarial example crafting techniques*
2) Generate more adversarial like images by adding small random noises to the normal images
3) Combine the normal training set with the adversarial images generated in 1) and 2) to retrain the classifier. Instead of hard labels for these images, the soft-labels are used instead. For instance, hand-written '7' and '1' is similar in that both have long vertical line. Hence, rather than label a hand-written digit as 100% '7' or 100% '1'. We may say it is 80% '7' and 20% '1'. This shows the structure similarity between '7' and '1'.
4) Use the combined training set to retrain the classifier
5) Repeat step 1-4 for k times, where k is the the maximum number of iteration specified. Note other types of stopping criteria are possible.

One common problem of adversarial training and other defense methods is the trade-off between the accuracy and resilience against adversarial examples. This trade-off exists because the network architecture of an original system and/or the dataset size is fixed. However, our proposed approach have larger network capacity with larger dataset generated by multiple techniques. Hence, our proposed approach does not have this trade-off, as shown in the evaluation section.

Another consideration is the training time. However, the proposed approach is highly parallelizable to reduce training time when multiple GPU resources are available. Due to the transferability of adversarial examples, the adversarial example does not have to be generated using the current model. Hence, adversarial crafting and adversarial training can be performed

at the same time. That is, at iteration t, an adversarial example can be generated based on the model generated at iteration $t' < t$, and adversarial training can be performed using the adversarial example generated previously as well. Depend on the memory and computation resource of a computer server, we can store the generated adversarial examples at each iteration and extract a random subset of the generated adversarial examples when performing the adversarial training. The sampling probability can be based on the performance of the model at previous iterations. For instance, we initially assign a probability of $\frac{1}{n}$ to each adversarial example. Then, we increase its probability for the next iteration if the model misclassifies it or it is not selected for the current iteration of training and decrease its probability if the model correctly classifies it.

At the testing time, random noise is added to the test image. This small noise is aimed to distort the intentional perturbation injected by the adversarial. Since our proposed model is trained with random noise, it is robust against it. Therefore, if the label of a given test image without added random noise is different from that of the one with added random noise, this is likely an adversarial image and the system is alerted.

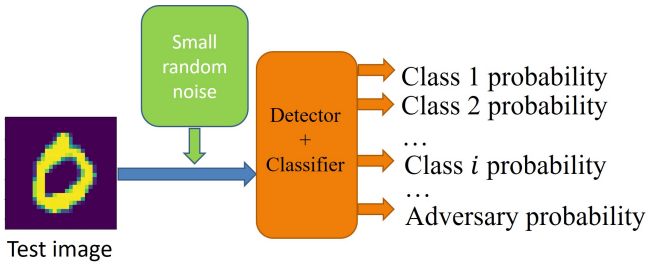

Fig. 3. Small random noise is added to the test image before it is input to the Detector+Classifier. The small noise added to the natural image is not likely to change the label of the image since we have trained our model with small Gaussian noise. However, the small random noise added to the adversarial image will disturb the carefully calculated perturbation injected by the adversary.

V. Evaluation

In this section, we empirically evaluate our proposed approach on the MNIST dataset [29]. The performance metrics considered are the accuracy of the classifier under various attacks.

A. Datasets and Models

In this paper, we consider the standard deep learning dataset, MNIST, that is widely used in the field. The MNIST handwritten digit recognition dataset is normalized, and each image has 28×28 pixels. All the images are black and white. See Table I for dataset summary.

TABLE I
Dataset Summary

Dataset	MNIST
# Training Example	60,000
# Testing Example	10,000
# Classes	10
Image Size	28×28
Original Accuracy	99%

B. Adversarial Crafting

We use the Adversarial Robustness Toolbox (ART v0.10.0) [30] to generate adversarial examples for training and testing. Due to the limited computing resource, we only use C&W, PGD attack, and Gaussian noise to generate adversarial images. Gaussian noise is added to the images to take care of other potential attacks and for better data generalization. The soft labels for those adversarial images are based on the perturbation introduced by these adversarial examples. For instance, under the C&W attack, we use soft a label of 0.41 if the perturbation limit is less than 0.026. 0.026 is selected based on the observation that the perturbation within this limit is hardly noticeable to human eyes. For a similar reason, the perturbation limit of PGD and Gaussian noise is set to 0.15 and 0.25 with a soft label value of 0.41. The network architecture for the MNIST dataset consists of two ReLU convolutional layers, one with 32 filters of size 3 by 3 and follows by another with 64 filters of size 3 by 3. Then, a 2 by 2 max-pooling layer and a dropout with a rate of 0.25 is applied before a ReLU fully connected layer with 1024 units. The last layer is another fully connected layer with 11 units and a softmax activation function for classification. The accuracy of the model is 99%, which is comparable to the accuracy of the state-of-the-art DNN.

At the testing time, we consider the strong white-box attack against our proposed model. The adversarial images are generated using FGSM, DeepFool, and C&W attack with the assumption that an adversary has feature knowledge, algorithm knowledge, and the ability to inject adversarial images. For the FGSM attack, the perturbation of $\epsilon = 0.1$ is considered. This is a reasonable limit because a larger perturbation can be detected by human and/or anomaly detection systems. Similarly, the maximum perturbation for the PGD attack is also set to 0.1 and the attack step size is set to $\frac{0.1}{3}$. The maximum number of iterations for PGD is 40. The setting of the C&W attack is left as default in ART [30].

TABLE II
Classification Accuracy on MNIST Test Dataset Under White-box Attack

Attack	FGSM	C&W	DeepFool
Original	29%	7%	29%
Robust Classifier	91%	70%	91%

C. Result

We consider the accuracy of the classifier on the normal MNIST test images and the accuracies of the classifier under FGSM, C&W, and DeepFool attacks. The prediction accuracy of the original classifier on the normal MNIST test images

is 99%. The prediction accuracy of the robust classifier on the normal MNIST test images is also 99% after retraining. Furthermore, the accuracies of the robust classifier under FGSM, C&W, and DeepFool attacks are shown in Table II. These results show that after retraining, the model performance is improved dramatically under these attacks.

VI. Conclusions and Future work

Adversarial examples limit the applicability of machine learning models, especially for security-sensitive or safety-critical applications, such as malware detection and self-driving cars. In this research, we proposed an iterative adversarial retraining approach for defense against adversarial examples and evaluated the performance on the MNIST dataset. The proposed approach is different from the adversarial retraining approach proposed by [1] in several aspects. First, we use soft-labels to prevent the network from becoming over-confident and improving generation. Instead, using only adversarial examples generated by adversarial sample crafting techniques, we also included the images with random Gaussian noise added. Last, to reduce the trade-off between the accuracy and the resilience against adversarial examples, we used the network with a larger capacity to train the model iteratively. The dropout is also added to prevent an overfitting problem. Since this iterative approach is independent of the DNN models, it can be applied to any machine learning models. Future work should investigate the impact of other adversarial sample crafting techniques on the proposed defense method. Besides, we will evaluate the performance of parallel architecture for iterative adversarial retraining methods.

Acknowledgment

We acknowledge the AFRL Internship Program to support Jing Lin's work and National Science Foundation to partially sponsor Dr. Kaiqi Xiong's work under grants CNS 1620862 and CNS 1620871, and BBN/GPO project 1936 through an NSF/CNS grant. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of NSF.

References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[2] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[3] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.

[4] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proceedings of the 2016 Network and Distributed Systems Symposium*, vol. 10, 2016.

[5] J. H. Metzen, M. C. Kumar, T. Brox, and V. Fischer, "Universal adversarial perturbations against semantic image segmentation," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 2774–2783.

[6] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[7] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, "Learning with a strong adversary," *CoRR*, vol. abs/1511.03034, 2015. [Online]. Available: http://arxiv.org/abs/1511.03034

[8] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.

[9] M. Abbasi and C. Gagné, "Robustness to adversarial examples through an ensemble of specialists," *arXiv preprint arXiv:1702.06856*, 2017.

[10] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[11] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[12] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 97–117.

[13] D. Gopinath, G. Katz, C. S. Pasareanu, and C. Barrett, "Deepsafe: A data-driven approach for checking adversarial robustness in neural networks," *arXiv preprint arXiv:1710.00486*, 2017.

[14] U. Iqbal, P. Snyder, S. Zhu, B. Livshits, Z. Qian, and Z. Shafiq, "Adgraph: A graph-based approach to ad and tracker blocking."

[15] A. R. Barron, "Approximation and estimation bounds for artificial neural networks," *Machine learning*, vol. 14, no. 1, pp. 115–133, 1994.

[16] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," in *Advances in Neural Information Processing Systems*, 2017, pp. 6231–6239.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[18] S. Raschka and V. Mirjalili, *Python Machine Learning : Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow 2, 3rd Edition.* Packt Publishing, Limited, 2019.

[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[20] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[21] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[22] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1765–1773.

[23] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 3–14.

[24] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–20, 2019.

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[26] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" *CoRR*, vol. abs/1906.02629, 2019. [Online]. Available: http://arxiv.org/abs/1906.02629

[27] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," *arXiv preprint arXiv:1702.06280*, 2017.

[28] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[29] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[30] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards, "Adversarial robustness toolbox v0.10.0," *CoRR*, vol. 1807.01069, 2018. [Online]. Available: https://arxiv.org/pdf/1807.01069