Defending DNN Adversarial Attacks with Pruning and Logits Augmentation

Siyue Wang[1*], Xiao Wang[2*], Shaokai Ye[3], Pu Zhao[1] & Xue Lin[1]
1. Northeastern University 2. Boston University 3. Syracuse University *Equal Contribution
wang.siy@husky.neu.edu kxw@bu.edu sye106@syr.edu
zhao.pu@husky.neu.edu xue.lin@northeastern.edu

***Abstract*—Deep neural networks (DNNs) have been shown to be powerful models and perform extremely well on many complicated artificial intelligent tasks. However, recent research found that these powerful models are vulnerable to adversarial attacks, i.e., intentionally added imperceptible perturbations to DNN inputs can easily mislead the DNNs with extremely high confidence. In this work, we enhance the robustness of DNNs under adversarial attacks by using pruning method and logits augmentation, we achieve both effective defense against adversarial examples and DNN model compression. We have observed defense against adversarial attacks under the white box attack assumption. Our defense mechanisms work even better under the grey box attack assumption.**

***Index Terms*—robust DNN, defending adversarial attacks, weight pruning, logits augmentation**

I. Introduction

Deep neural networks (DNNs) are powerful models that achieve state-of-the-art performance in various speech and visual recognition tasks, including speech recognition, natural language processing, scene understanding, object recognition, etc. As a key enabler of DNNs, the large model size also demands increasing computation and memory resources from the computing platforms. It has been investigated that DNNs are robust to random perturbations [1]. However, recent study [2], [3], [4] show that DNNs are vulnerable to adversarial attacks, that is, intentionally added imperceptible perturbations to DNN inputs can easily mislead the DNNs with extremely high confidence [5], [6].

Goodfellow et al. [3], Kurakin et al. [4], Papernot et al. [7] and Carlini & Wagner [8] have implemented the adversarial attacks by generating adversarial examples. If the neural network classifies a legal input x with label $C(x)$, an adversarial example x' is the one that very similar to x according to some distance metrics and will be labeled by the neural network as $C(x') \neq C(x)$.

Goodfellow et al. [3] proposed Fast Gradient Sign Method (FGSM) that uses the gradient of the loss function to determine the direction of the perturbation and sets a constant for the intensity of the perturbation. This method features very fast speed for generating adversarial examples.

Kurakin et al. [4] enhanced the FGSM by taking multiple smaller steps in the direction of gradient sign, which is known as Basic Iterative Method (BIM).

Carlini & Wagner [8] claimed to generate the strongest attacks (the C&W method) by solving an optimization problem with minimizing some distance metrics, i.e., L_p norm, where $p = 0$, 2, and ∞ are used. C&W is the strongest attacks in that it conquered multiple defense methods in the *white box* assumption, i.e., the attacker has perfect knowledge about the targeted DNN model and data set.

Previous works have been done on improving the robustness of DNNs under adversarial attacks [2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14, 15]. Papernot et al. [16] proposed their defensive mechanism called defensive distillation by using tempriture in the softmax function in order to increase the robustness of DNNs. Others [17], [18] tried to defend the adversarial examples by attempting to detect them. Feinman et al. [17], through modeling the outputs from the final hidden layer of DNNs, indicated that there exists the difference in the distribution of adversarial examples and legal examples, while Bhagoji et al. [18] proposed a defense mechanism based on dimensionality reduction.

In this work, we enhance the robustness of DNNs under adversarial attacks by using pruning method and logits augmentation, therefore, we achieve both higher defense against adversarial examples and more compressed DNN models. We have observed defense against adversarial attacks under the white box attack assumption. Our defense mechanisms work even better under the grey box attack assumption.

II. Related Works

A. Preliminaries

In this paper we focus on neural networks used as image classifiers. In this case, the input images can be denoted as 3-dimensional tensors $x \in \mathbb{R}^{h \times w \times c}$, where h, w and c denote the height, width and number of channels. For a gray scale image (e.g. MNIST), $c = 1$; and for a colored RGB image (e.g. CIFAR-10), $c = 3$. For both attacks and defends, all pixel values in the images are scaled to $[0, 1]$ for easy calculation, and therefore a valid input image should be inside a unit cube in the high dimensional space. We use model $F(x) = y$ to denote a neural network, where F accepts an input x and generates an output y.

Here we denote the neural network as an m-class classifier and the output layer performs softmax operation. The logits, $Logit(x)$, are the output of all layers except for the softmax layer, and the input to the softmax layer. So $F(x) = \text{softmax}(Logit(x)) = y$. The elements of the output vector y denote the probability that input x belongs to each

class. They must be in the range of $[0, 1]$ and their sum should be one. The classifier assigns the label $C(x) = \arg\max_i y_i$ to input x according to the maximum probability.

Adversarial attacks can be either targeted or untargeted. The targeted adversarial attacks is to find an input x' such that it's classified as label t while it is close to the original input x according to some distance metrics. Note that the correct label of x, t^*, is different from the target label t. The malicious input x' satisfying these requirements is an adversarial example. The untargeted adversarial attack is to find an input x' that satisfying $C(x') \neq t^*$ and x and x' are close according to some distance metrics. The untargeted adversarial attack does not specify any target label t to mislead the classifier.

The general problem of constructing adversarial examples can be formulated as: **Given** an original legal input x,

$$\begin{array}{ll} \text{minimize} & D(\delta) \\ \text{subject to} & C(x+\delta) = t \\ & x + \delta \in [0,1]^n \end{array} \tag{1}$$

where δ is the distortion added onto input x, $D(\delta)$ is a measure of the added distortion.

We need to measure the distortion between the original legal input x and the adversarial example $x' = x + \delta$. L_p norms are the most commonly used measures in the literature. The L_p norm of the distortion is defined as:

$$\|x - x'\|_p = \left(\sum_{i=1}^{n} |x_i - x'_i|^p \right)^{\frac{1}{p}} \tag{2}$$

We see the use of L_0, L_1, L_2, and L_∞ norms in different attacks. L_0 *norm* measures the number of mismatched elements between x and x'. L_1 *norm* measures the sum of the absolute values of the differences between x and x'. L_2 *norm* measures the standard Euclidean distance between x and x'. And L_∞ *norm* measures the maximum difference between x_i and x'_i for all i's.

B. Attacks

1) Fast Gradient Sign Method (FGSM) [3]:: is an L_∞ attack and utilizes the gradient of the loss function to determine the direction to modify the pixels. They are designed to be fast, rather than optimal. They can be used for adversarial training by directly changing the loss function instead of explicitly injecting adversarial examples into the training data. FGSM generates adversarial examples following:

$$x' = x - \epsilon \cdot \text{sign}(\nabla(loss_{F,t}(x))) \tag{3}$$

where ϵ is the magnitude of the added distortion, t is the target label. Using backpropagation, FGSM calculates the gradient of the loss function with respect to the label t to determine the direction to change the pixel values.

2) Basic Iterative Method (BIM) [4]:: gives a further modify of FGSM. Instead of taking a single step ϵ, BIM takes multiple steps α Given an initial setting:

$$x'_0 = x$$

for each iteration

$$x'_i = x'_{i-1} - clip_\epsilon(\alpha sign(\nabla(loss_{F,y}(x'_{i-1}))))$$

Notice that here BIM clips pixel values of intermediate results after each step to ensure that they are in an ϵ-neighbourhood of the original image.

3) Jacobian-based Saliency Map Attack (JSMA) [7]:: is an L_0 attack and uses a greedy algorithm that picks the most influential pixels by calculating Jacobian-based Saliency Map and modifies the pixels iteratively. The computational complexity of this attack method is very high.

4) C&W [8]:: is a series of L_0, L_2, and L_∞ attacks that achieve 100% attack success rate with much lower distortions comparing with the above-mentioned attacks. In particular, the C&W L_2 attack is superior to other L_2 attacks because it uses a better objective function. C&W formulates the problem of generating adversarial examples in an alternative way that can be better optimized:

$$\begin{array}{ll} \text{minimize} & D(\delta) + c \cdot f(x+\delta) \\ \text{subject to} & x + \delta \in [0,1]^n \end{array} \tag{4}$$

where $c > 0$ is a constant to be chosen and the objective function f has the following form:

$$f(x+\delta) = \max(\ \max\{Logit(x+\delta)_i : i \neq t\} - Logit(x+\delta)_t, -\kappa) \tag{5}$$

Here, κ is a parameter that controls the confidence in attacks. Stochastic gradient decent methods can be used to solve this problem. For example, the Adam optimizer [19] is used due to its fast and robust convergence behavior.

III. METHODOLOGY

A. Model Compression Using Pruning

To reduce model size and facilitate implementing DNNs for customer applications, [20] proposed the DNN pruning method that reduces the number of weights while preserving the accuracy of the compressed DNN models. The pruning process starts from learning the connectivity through normal network training, followed by pruning the connections whose weights are below a given threshold. After making it a sparser network, the DNN is retrained to finalize weights of the remaining connections. This pruning-and-retraining process is performed iteratively until the network is pruned to the largest extent without accuracy loss.

In this work, we use a network structure with 4 convolutional layers and 3 fully connected layers. In each iteration, we prune 10% nonzero weights for fully connected layers and 5% nonzero weights for convolutional layers. We prune and train for 20 iterations maintaining the accuracy and the network model can be compressed by 7×. We demonstrate in the later Section that the pruning-based model compression method can defend the adversarial attacks, that is, by using pruning method we can achieve both compressed network model size and defense against adversarial attacks.

TABLE I: Adversarial attack successful rate (and distortion) of the unprotected model M0, Level One model M1, and Level Two model M2 under four attacks (untargeted FGSM, targeted FGSM, targeted BIM, and C&W) using MNIST dataset.

Attack Method	Untargeted FGSM			Targeted FGSM			Targeted BIM			C&W
Parameters	ε= 0.1	ε = 0.15	ε = 0.25	ε = 0.1	ε= 0.15	ε = 0.25	ε= 0.1	ε = 0.15	ε = 0.25	iter = 100
M0	9.0% (2.19)	17.0% (3.28)	45.6% (5.45)	1.97% (2.17)	4.52% (3.25)	12.0% (5.39)	3.89% (2.11)	14.81% (3.11)	39.64% (5.28)	99.6% (2.03)
M1	7.4% (2.16)	8.7% (3.25)	20.2% (5.38)	1.17% (2.15)	1.68% (3.22)	4.04% (5.35)	3.14% (2.14)	9.9% (3.13)	31.26% (5.07)	96.97% (2.28)
M2	1.1% (2.28)	1.1% (3.41)	1.1% (5.65)	1.04% (2.15)	1.5% (3.22)	3.87% (5.35)	2.71% (2.15)	7.9% (3.1)	21.12% (5.1)	95.93% (2.5)

The experiment is evaluated on 1000 source samples from MNIST. We set the search step for line search in C&W as 10.

TABLE II: Adversarial attack successful rates (and distortion) of the unprotected model C0, Level One model C1, and Level Two model C2 under four attacks using CIFAR-10 dataset.

Attack Method	Untargeted FGSM			Targeted FGSM			Targeted BIM			C&W
Parameters	ε= 0.1	ε = 0.15	ε = 0.25	ε = 0.1	ε= 0.15	ε = 0.25	ε= 0.1	ε = 0.15	ε = 0.25	iter = 100
C0	84.6% (5.43)	86.3% (8.05)	87.1% (13.0)	17.71% (5.43)	14.78% (8.05)	11.49% (13.0)	63.59% (4.48)	65.83% (6.66)	65.73% (10.8)	99.54% (2.06)
C1	70.3% (5.43)	75.3% (8.05)	80.9% (13.0)	11.2% (5.42)	10.5% (8.05)	10.1% (13.03)	25.3% (4.47)	23.8% (6.64)	19.3% (10.8)	85.0% (3.55)
C2	24.6% (1.42)	24.5% (2.11)	25% (3.41)	11.12% (5.33)	11.25% (7.91)	11.16% (12.8)	43.41% (4.43)	44.9% (6.5)	41.2% (10.7)	83.9% (4.31)

The experiment is evaluated on 1000 source samples from CIFAR-10. We set the search step for line search in C&W as 10.

B. Logits Augmentation

To further improve the robustness of DNNs under adversarial attacks, we propose to use the logits augmentation on top of the pruning method. Inspired by the gradient inhibition method [21], which changes the weights in the last few layers as

$$w = w + \tau * sign(w). \tag{6}$$

In our logits augmentation, we modify the weights in the last fully connected layer by

$$w = \tau \times w \tag{7}$$

In our experiments, we set the value of τ to fine-tune the defense effectiveness. Through a detailed analysis, we find that both the pruning and the logits augmentation can change the distribution of weights in a DNN and therefore achieve some level of defense against adversarial examples.

IV. Experiments and Results

A. Experiment Setup

In order to test our defense mechanisms against adversarial examples, we use three adversarial example generating methods i.e., FGSM [3], BIM [4], and C&W [8]. We use both MNIST [22] and CIFAR-10 [23] datasets to train the network models. The three attacks have been implemented in Cleverhans [24], a Python library to benchmark machine learning systems' vulnerability to adversarial examples, and we use those source codes directly for generating adversarial examples. We are using the *white box* assumption when generating the adversarial examples, i.e., the attackers have perfect knowledge of all the targeted neural network models, training and testing datasets.

We start from training unprotected neural network models i.e., M0 and C0, achieving near state-of-the-art accuracy, i.e., 99.4% and 80%, respectively, on MNIST and CIFAR-10 datasets. For defense mechanisms, we test two defending levels: Level One exploits the pruning method [20] only as defense, while Level Two exploits both pruning and logits augmentation as defense. We have M1 and C1 network models for Level One defense, respectively, for the MNIST and CIFAR-10 datasets. We have M2 and C2 network models for Level Two defense. Please note that we do not lose any test accuracy under Level One and Level Two models.

B. Experiment Results

Table I and Table II demonstrate the adversarial attack successful rates of unprotected models, Level One models, and Level Two models under four attacks (untargeted FGSM, targeted FGSM, targeted BIM, and C&W) on MNIST and CIFAR-10 datasets, respectively. We also report with the attack successful rate and the average distortion (the one in the parentheses) of the adversarial examples compared to the legal

examples. The ε parameter is related to the distortion. When a larger ε value is used, the resulted distortion is higher. For the untargeted FGSM, the adversarial attack successful rate for MNIST dataset (the fourth column in Table I) is reduced from 45.6% by unprotected model M0 to 20.2% by Level One model M1 and to 1.1% by Level Two model M2. Similarly, we can observe decrease in the adversarial example successful rate under other attacks and for CIFAR-10 dataset. Here we find that our defense mechanisms do not defend the C&W attacks as much as we do for the other attacks under the pure white box assumption.

Furthermore, we test our defense mechanisms against C&W attacks on a grey box attack assumption when generating the adversarial examples, i.e., the attackers know the pruning percentage used on each layer and have perfect knowledge of all the training and testing datasets. Whereas black box attack assumes the adversary has no information about the DNN model, only observe labels assigned by the DNN for chosen inputs. Under the grey box assumption, we generate adversarial examples using C0 and attack the Level One model C1. We reduce the adversarial example successful rate by Level One model C1 to 18.63%, which was 85.0% under white box attack. It demonstrates that our defense mechanism is more effective under grey box attacks.

V. Conclusion

In this paper, we enhance the robustness of deep neural networks by using pruning method and logits augmentation. Providing a solution through compressing DNN models and modifying the weights in the last layer, we achieve both effective defense against adversarial attacks and DNN model compression. We test our Level One models, and Level Two models under four attacks (untargeted FGSM, targeted FGSM, targeted BIM, and C&W) on MNIST and CIFAR-10 datasets, respectively. We have observed defense against adversarial attacks under the white box attack assumption. Experiment results demonstrate our defense mechanism is more effective under grey box attack assumption.

Acknowledgements

This work is supported by the National Science Foundation CCF-1733701, Air Force Research Laboratory FA8750-18-2-0058, and U.S. Office of Naval Research.

References

[1] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard, "Robustness of classifiers: from adversarial to random noise," in *Advances in Neural Information Processing Systems*, pp. 1632–1640, 2016.

[2] 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.

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

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

[5] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands.," in *USENIX Security Symposium*, pp. 513–530, 2016.

[6] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436, 2015.

[7] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387, IEEE, 2016.

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

[9] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Ead: elastic-net attacks to deep neural networks via adversarial examples," *arXiv preprint arXiv:1709.04114*, 2017.

[10] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[11] P. Zhao, S. Liu, Y. Wang, and X. Lin, "An admm-based universal framework for adversarial attacks on deep neural networks," *arXiv preprint arXiv:1804.03193*, 2018.

[12] J. Su, D. V. Vargas, and S. Kouichi, "One pixel attack for fooling deep neural networks," *arXiv preprint arXiv:1710.08864*, 2017.

[13] S. Wang, X. Wang, P. Zhao, W. Wen, D. Kaeli, P. Chin, and X. Lin, "Defensive dropout for hardening deep neural networks under adversarial attacks," *arXiv preprint arXiv:1809.05165*, 2018.

[14] M. Hong and Z.-Q. Luo, "On the linear convergence of the alternating direction method of multipliers," *Mathematical Programming*, vol. 162, pp. 165–199, Mar 2017.

[15] H. Wang and A. Banerjee, "Bregman alternating direction method of multipliers," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2816–2824, Curran Associates, Inc., 2014.

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

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

[18] A. N. Bhagoji, D. Cullina, and P. Mittal, "Dimensionality reduction as a defense against evasion attacks on machine learning classifiers," *arXiv preprint arXiv:1704.02654*, 2017.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *2015 ICLR*, vol. arXiv preprint arXiv:1412.6980, 2015.

[20] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, pp. 1135–1143, 2015.

[21] Q. Liu, T. Liu, Z. Liu, Y. Wang, Y. Jin, and W. Wen, "Security analysis and enhancement of model compressed deep learning systems under adversarial attacks," in *Design Automation Conference (ASP-DAC), 2018 23rd Asia and South Pacific*, IEEE, 2018.

[22] L. Yann, C. Corinna, and J. Christopher, "The mnist database of handwritten digits," *URL http://yhann. lecun. com/exdb/mnist*, 1998.

[23] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[24] N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel, "cleverhans v1.0.0: an adversarial machine learning library," *arXiv preprint arXiv:1610.00768*, 2016.