Using Intuition from Empirical Properties to Simplify Adversarial Training Defense

Abstract— **Due to the surprisingly good representation power of complex distributions, neural network (NN) classifiers are widely used in many tasks which include natural language processing, computer vision and cyber security. In recent works, people noticed the existence of adversarial examples. These adversarial examples break the NN classifiers' underlying assumption that the environment is attack free and can easily mislead fully trained NN classifier without noticeable changes. Among defense methods, adversarial training is a popular choice. However, adversarial training with single-step adversarial examples (Single-Adv) can not defend against iterative adversarial examples. Although adversarial training with iterative adversarial examples (Iter-Adv) can defend against iterative adversarial examples, it consumes too much computational power and hence is not scalable. In this paper, we analyze Iter-Adv techniques and identify two of their empirical properties. Based on these properties, we propose modifications which can simplify Iter-Adv defenses. Through preliminary evaluation, we show that the proposed method enhances the test accuracy of state-of-the-art (SOTA) Single-Adv defenses against iterative adversarial examples by up to 9.83% while reducing its computational cost by 28.75%.**

Index Terms—**adversarial training, adversarial example, neural network classifier**

I. Introduction

Adversarial examples were discovered by Szegedy et. al. and presented in [14]. In the image classification tasks, they show that a specially designed perturbation which can be ignored by human eyes can effectively mislead the fully trained NN classifier. Moreover, such perturbation is not a special case but can be found for almost every example. Yet, more scary, the research shows that adversary could arbitrarily control the prediction from NN classifier through carefully designed perturbations and can achieve high success rate against Vanilla classifiers, i.e., classifiers without defenses [2] [8] [9].

Thereafter, great effort has been devoted to designing an effective method to defend against adversarial examples. Some methods utilize augmentation and regularization to enhance test accuracy on adversarial examples. The idea here is to improve the generalization of the classifier as a defense against adversarial examples [11]. Other methods rely on building a protective shells around the classifier to either identify adversarial examples or mitigate the adversarial perturbations [10] [12]. Among all existing defensive approaches, adversarial training is shown to be more successful because unlike many other defensive approaches it does not rely on the false sense of security brought by obfuscated gradient [1]. Within adversarial training, the hidden logic is using a mixture of original and adversarial examples to train the NN. The stronger the adversarial examples used the stronger the obtained defense. Following this logic, the research community originally started with Single-Adv and now moved to utilizing Iter-Adv [5] [7] [9].

One of the biggest problems of Iter-Adv is the huge computational cost in preparing iterative adversarial examples during the training [1]. For example, using Iter-Adv on ImageNet dataset requires a cluster of GPU servers [6]. Nowadays, there is a trend to make the NN classifier more portable such that it can be trained and utilized solely in a smart-phone [3]. Moreover, due to data privacy consideration, some applications calculate the last few layers of NN on local device. Under these scenarios, we need a lightweight adversarial training defense since Iter-Adv is hard to be scaled with limited computational resources [1]. In the following sections, we start with two questions about Iter-Adv. Then, based on our empirical results, we propose two modifications to Iter-Adv and share our idea to simplify it.

Our contributions can be summarized as follows:

- We raise two questions about Iter-Adv and conduct experiments that identify two empirical properties of it.
- Based on the identified properties, we propose two modifications to simplify Iter-Adv.
- Through comparison with SOTA Single-Adv method, we show that our simplified adversarial training defense can enhance test accuracy by 9.83% and reduce training time by 28.75%.

The rest of the work is organized as follows. Section II and III present our questions and the preliminary experiments. Section IV proposes modifications that simplify Iter-Adv. Section V presents our preliminary evaluation results. Section VI concludes the paper and presents the future work.

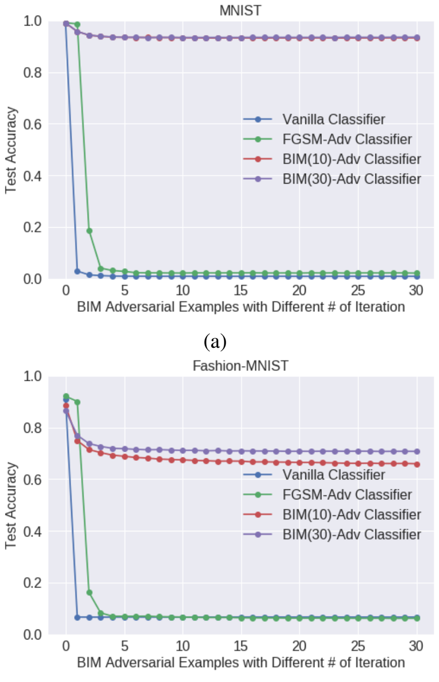
II. Is it Beneficial to keep decreasing per step perturbation?

Let's take a step back and recall the definition of gradient based l_∞ iterative adversarial examples.

$$\delta_i = sign(\nabla_{x_{i-1}}\mathcal{L}(\mathcal{C}(x_{i-1}, \theta), y)) \times \epsilon_i$$
$$x_i = clip(x_{i-1} + \delta_i)$$

where x_0 is the original example, x_i is the ith iteratioin adversarial example, y is the ground truth, $\mathcal{C}$ is the classifier, $\mathcal{L}$ is the loss function, ϵ_i is the perturbation limit in the ith iteration, and δ_i is the calculated perturbation in the ith iteration.

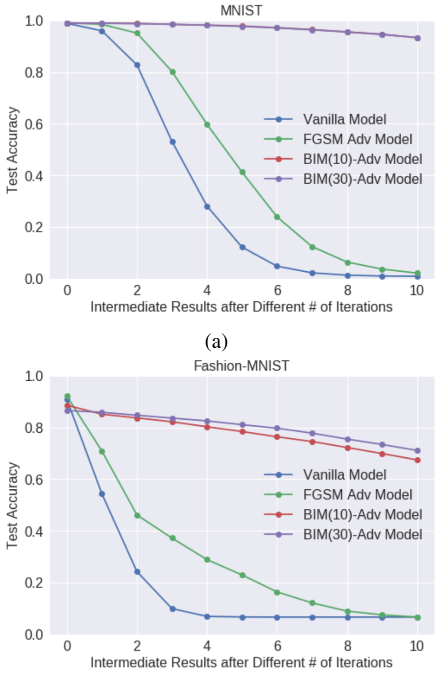
To generate iterative adversarial examples, adversaries apply small per step perturbation several times and update the



(a)

(b)

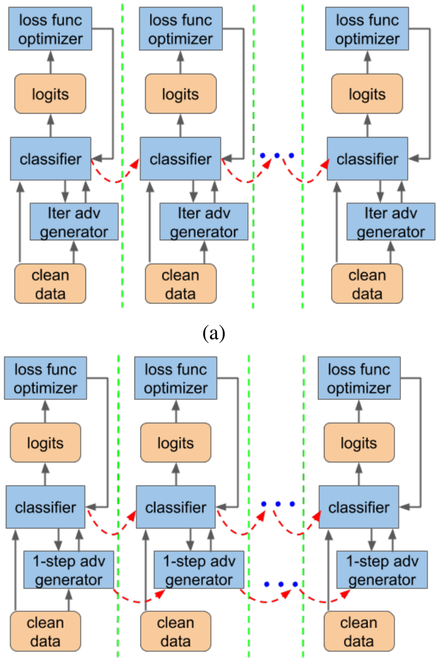
Fig. 1: Test Accuracy on BIM Examples with Different Numbers of Iteration



(a)

(b)

Fig. 2: Test Accuracy on Intermediate BIM Examples after Each Iteration



(a)

(b)

Fig. 3: Flow Chart of Iter-Adv and the Proposed Method

gradient direction based on their observation of the targeted NN after each step. Generally speaking, the smaller per step perturbation they apply the better observation of NN's decision hyperplane they may get. With a better observation of NN's decision hyperplane, adversaries could greedily optimize their objective function and generate more serious adversarial examples.

However, there are still several questions for us to clarify. **Given the relation between iterative adversarial examples and the per step perturbation, how much can we benefit from a smaller per step perturbation? Is there a limit to the per step perturbation where the improvement starts to vanish beyond this limit?**

To answer these questions, we conduct experiments on MNIST and Fashion-MNIST datasets, respectively. On each dataset, we firstly train four different NN classifiers with the same structure and hyper-parameter setting [13] which include: (1) a Vanilla classifier trained on original examples only, (2) a FGSM-Adv classifier trained on a mixture original and FGSM examples [5], (3) two BIM($\cdot$)-Adv classifiers trained on a mixture of original and BIM (different numbers of iteration) examples [7]. The number of iterations for the two BIM-Adv classifiers is 10 and 30, respectively. We evaluate these classifiers in terms of test accuracy when facing BIM examples with different numbers of iteration, N. The total perturbation, ϵ, is fixed to 0.3 (MNIST) and 0.2 (Fashion-MNIST) with l_∞ norm. The per step perturbation is set to $\epsilon_s = \frac{\epsilon}{N}$. The evaluation results are presented in Figure 1.

From Figure 1, it is clear that test accuracy results of all four classifiers converge quiet fast. The Vanilla and FGSM-Adv classifiers can not defend BIM examples and their test accuracy results drop below 10% (random guessing) when the iteration number is larger than 4 on both MNIST and Fashion-MNIST datasets. The BIM-Adv classifiers are shown to be defensive against BIM examples and their test accuracy results also converge after we set iteration number to 5 (MNIST) and 10 (Fashion-MNIST). **Given the fact that adversarial training uses adversarial examples to find blind spots of the under-training classifier and train it, these experiments show that (1) there is a limit for decreasing the per step perturbation of iterative adversarial examples and (2) iterative adversarial examples with per step perturbation lower than the limit only marginally benefit the adversarial training.** This conclusion can also be verified through comparing test accuracy results of BIM-Adv classifiers. Although BIM(30)-Adv is trained on BIM examples with much smaller per step perturbation compared with BIM(10)-Adv, they have almost the same test accuracy on MNIST dataset. On Fashion-MNIST, BIM(30)-Adv is shown to be more defensive than BIM(10)-Adv. However, the difference on test accuracy does not increase with the iteration number. The reason is that BIM(10)-Adv has a stable test accuracy even facing BIM examples with smaller per step perturbation ($N > 10$).

Based on the results and the conclusion in this subsection, our suggestion is the following. When facing the trade-off between defensive performance and computational efficiency in training with iterative adversarial examples, the per step perturbation is not necessary to be very small since it only marginally benefits the adversarial training.

III. Can we benefit from utilizing the intermediate results?

Adversarial training originally uses single-step adversarial examples and recently shifts to iterative adversarial examples. Since iterative adversarial examples are more serious and able to reveal more blind spots of under-training classifier, such shift improves the adversarial training in building classifier's defense against iterative adversarial examples.

However, there are still several questions to be answered during this shift. **When we prepare iterative adversarial examples for training, can we also benefit from using intermediate results from this generation process as well?**

To explore the answer for this question, we conduct another set of experiments on MNIST and Fashion-MNIST datasets, respectively. On each dataset, we continuously use the same NN classifiers (one Vanilla classifier, one FGSM-Adv classifier and two BIM-Adv classifiers) as before. During the evaluation, we test all four classifiers in terms of test accuracy against BIM examples. The total perturbation of test examples is set to $\epsilon = 0.3$ (MNIST) and $\epsilon = 0.2$ (Fashion-MNIST). Different from previous experiments, we generate BIM examples with fixed iteration number, $N = 10$, and evaluate the test accuracy after each iteration. Therefore, the per step perturbation is fixed to $\epsilon_s = \frac{\epsilon}{10}$ while perturbation is increasing after each iteration. The evaluation results are presented in Figure 2.

The experiment results from Figure 2 show that the test accuracy of all four classifiers is monotonically decreasing with the number of iterations. The classifiers without defense, Vanilla and FGSM-Adv classifiers, are defeated (perform worse than random guessing) by the intermediate results before the iterative adversarial examples are ready (around 8 iterations on both MNIST and Fashion-MNIST datasets). Although two BIM-Adv classifiers obtain defense from training and can correctly classify most of adversarial examples, the majority of test accuracy degeneration still happens within first couple of iterations (about 6 iterations on both MNIST and Fashino-MNIST datasets). **Given the fact that adversarial training uses adversarial examples to find blind spots of the under-training classifier and train it, these experiment results show that (1) the majority of blind spots can be revealed by intermediate results during generating iterative adversarial examples and (2) using intermediate results can benefit adversarial training before iterative adversarial examples are ready.**

Based on the results and the conclusion it leads to, we can summarize our second suggestion as follows. To enhance the training efficiency in Iter-Adv, we could utilize the intermediate results in training before the iterative adversarial examples are ready since the majority of weaknesses can be revealed by them at that time. It is worth to clarify that we do not suggest to fully replace iterative adversarial examples with intermediate results but the utilization of intermediate results can reduce the total computation in Iter-Adv.

IV. Simplifying Adversarial Training Defense

From previous sections, we raise two questions regarding the training with iterative adversarial examples and provide an intuition to answer them based on the results from the MNIST and the Fashion-MNIST datasets. In this section, we propose our modifications to simplify Iter-Adv which takes both defensive performance and computational efficiency into consideration.

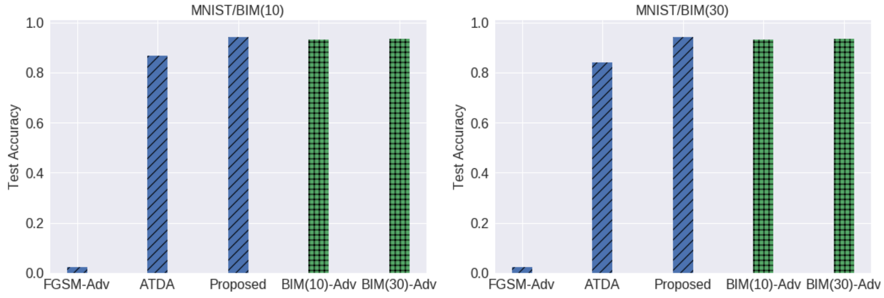
Before moving further, we want to firstly review the working process of Iter-Adv. As we can see from Figure 3a, a flow chart of adversarial training is provided. At the beginning, the clean examples are fed into the classifier and the generator of adversarial examples. The generator then interact with the classifier for several iterations to prepare iterative adversarial examples. When adversarial examples are ready, the classifier generates prediction logits for both clean and adversarial examples through forward propagation [4]. Finally, a predefined loss function optimizer takes these prediction logits to calculate the loss value and update NN parameters in classifier through gradient descent and backward propagation [4]. During the training, these steps are repeated for several epochs (separated by the green dash line) and NN parameters in classifier are carried through epochs (the red dash line with arrow).

Inspired by the suggestions that are summarized from experiments in previous sections, we now propose two modifications to simplify Iter-Adv. The flow chart of our proposed method can be found in Figure 3b. Compared with Iter-Adv, our first modification is utilizing the intermediate results in the training process. In each training epoch, our proposed method will not wait until the final iterative adversarial examples are ready. Instead, it utilizes the intermediate results and carry them to the next training epoch. As a result, it only requires a single-step perturbation in each epoch. The second modification in the proposed method is selecting a relatively large per step perturbation. With this modification, the adversarial examples can quickly reach large perturbation and reveal the majority of blind spots. Therefore, it mitigates the disadvantage brought by using single-step adversarial perturbation, training with weak adversarial examples in the first few training epochs. To catch up the long term changes in classifier's parameters, this epoch-wise iteration process will be reset after a certain number of training epochs.

From Figure 3a and Figure 3b, it is clear that our proposed method significantly reduces computational cost required by preparing iterative adversarial examples in each training epoch. By reusing the adversarial examples in the next training epoch and selecting large per step perturbation, we can also expect the proposed method to perform better than FGSM-Adv and closer to BIM-Adv.

V. Preliminary Results

Here, we compare the proposed method with several Single-Adv and Iter-Adv methods. Iter-Adv methods include FGSM-Adv, BIM(10)-Adv and BIM(30)-Adv. We also include a SOTA Single-Adv method which is called ATDA [13]. This



(a)

(b)

Fig. 4: Evaluation Results of Test Accuracy

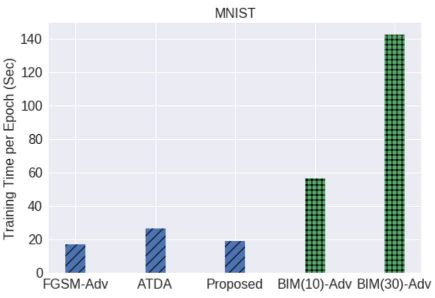


Fig. 5: Evaluation Results of Training Time per Epoch

method trains the classifier with single-step adversarial examples and modifies training loss to achieve domain adaptation. Our proposed method trains the classifier with single-step adversarial examples and the perturbation is limited at $\frac{\epsilon}{10}$. Moreover, the epoch-wise iteration presented in Figure 3b is repeated every 20 training epochs. The results from Single-Adv methods are represented by the blue color bars while the results from Iter-Adv methods are represented by the green color bars.

The evaluation is conducted on MNIST dataset and measures both defensive power and computational consumption in training. To measure defensive power, we evaluate different methods against two white-box l_∞ iterative adversarial examples which are utilized in Section II, BIM(10) and BIM(30). Since all methods are run on the same workstation with a Tesla K20m GPU and converge after the same number of epochs, we utilize the training time per epoch as a measure of computational consumption.

From the results presented in Figure 4, it is clear that FGSM-Adv has no defense against iterative adversarial examples while ATDA, BIM(10)-Adv, BIM(30)-Adv and our proposed method show resistance against them. Our proposed method constantly outperforms ATDA in term of test accuracy. The enhancements are 7.61% on BIM(10) and 9.83% on BIM(30). Compared with BIM(10)-Adv and BIM(30)-Adv, our proposed method achieves the same level and even slightly higher test accuracy. Given that MNIST is a small dataset and its defense is relatively easy to achieve, the evaluation results of test accuracy show that (1) our proposed method beats the SOTA Single-Adv method and matches the defense created by Iter-Adv methods and (2) our proposed method has the potential to perform similarly on larger datasets.

From the results presented in Figure 5, it is clear that Single-Adv methods significantly reduce the training time consumed by Iter-Adv by up to 85%. More importantly, our proposed method can further reduce 28.75% of the training time required by the SOTA Single-Adv method, ATDA.

VI. Conclusion and Future Work

In this work, we raise two questions about Iter-Adv and identify two empirical properties. (1) Iterative adversarial examples with per step perturbation lower than a certain limit only marginally benefit the adversarial training. (2) The majority of blind spots can be revealed by intermediate results during generating iterative adversarial examples. Based on these two properties, we propose a simplified adversarial training method. Through preliminary results, the proposed method outperforms the SOTA Single-Adv method and achieves the same level of defense as Iter-Adv methods. More importantly, the proposed method significantly reduces the training time required by Iter-Adv methods and even save 28.75% of training time required by the SOTA Single-Adv method. In the future, we are going to extend the experiments on larger datasets to validate and refine the proposed method.

References

[1] 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.

[2] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," pp. 39–57, 2017.

[3] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.

[4] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *International Conference on Learning Representations*, 2015.

[6] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," *arXiv preprint arXiv:1803.06373*, 2018.

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

[8] ——, "Adversarial machine learning at scale," *International Conference on Learning Representations*, 2017.

[9] 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.

[10] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," pp. 135–147, 2017.

[11] 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*. IEEE, 2016, pp. 582–597.

[12] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," *arXiv preprint arXiv:1805.06605*, 2018.

[13] C. Song, K. He, L. Wang, and J. E. Hopcroft, "Improving the generalization of adversarial training with domain adaptation," *arXiv preprint arXiv:1810.00740*, 2018.

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