Adversarial Parameter Attack on Deep Neural Networks

Lijia Yu [1 2] **Yihan Wang** [1 2] **Xiao-Shan Gao** [1 2]

Abstract

The parameter perturbation attack is a safety threat to deep learning, where small parameter perturbations are made such that the attacked network gives wrong or desired labels of the adversary to specified inputs. However, such attacks could be detected by the user, because the accuracy of the attacked network will reduce and the network cannot work normally. To make the attack more stealthy, in this paper, the adversarial parameter attack is proposed, in which small perturbations to the parameters of the network are made such that the accuracy of the attacked network does not decrease much, but its robustness against adversarial example attacks becomes much lower. As a consequence, the attacked network performs normally on standard samples, but is much more vulnerable to adversarial attacks. The existence of nearly perfect adversarial parameters under L_∞ norm and L_0 norm is proved under reasonable conditions. Algorithms are given which can be used to produce high quality adversarial parameters for the commonly used networks trained with various robust training methods, in that the robustness of the attacked networks decreases significantly when they are evaluated using various adversarial attack methods.

1. Introduction

The deep neural network (DNN) (LeCun et al., 2015) has become one of the most powerful machine learning methods, which has been successfully applied in computer vision, natural language processing, and many other fields. In security-critical systems using DNNs, such as autonomous vehicles, banking systems, etc., safety is a key desired feature of DNNs, which has been studied extensively. Refer to the survey papers (Akhtar & Mian, 2018; Bai et al., 2020; Zhang et al., 2020) for details.

The most widely studied safety issue for DNNs is the adversarial attack (Szegedy et al., 2013; Goodfellow et al., 2014; Carlini & Wagner, 2017), that is, it is possible to intentionally make small modifications to a standard sample to generate adversarial examples which are essentially imperceptible to the human eye, but the DNN outputs a wrong label or even any label given by the adversary. The existence of adversary examples makes the DNN vulnerable in safety-critical applications and many effective methods were proposed to develop more robust DNNs against adversarial attacks (Madry et al., 2017; Akhtar & Mian, 2018; Bai et al., 2020; Zhang et al., 2020). On the other hand, it was shown that adversaries examples are inevitable for commonly used DNN models in certain sense (Azulay & Weiss, 2018; Shafahi et al., 2018; Bastounis et al., 2021; Gao et al., 2022).

The parameter perturbation attacks (Liu et al., 2017; Zhao et al., 2019; Breier et al., 2018; Tsai et al., 2021; Sun et al., 2021; Weng et al., 2020; Tyukin et al., 2020; 2021) were also shown to be a safety threat to DNNs. It was shown that by making small parameter perturbations, the attacked DNN can output wrong or desired labels to specified inputs and still gives the correct labels to other samples (Liu et al., 2017; Zhao et al., 2019; Tyukin et al., 2020; 2021). However, these attacks are easy to be detected by users, possibly due to a prominent decline in performance such as the training accuracy.

In this paper, the adversarial parameter attack is proposed as a more stealthy parameter perturbation attack, in which small perturbations to the parameters of a DNN are made such that the attack to the DNN is essentially imperceptible to the user, but the robustness of the DNN becomes much lower. The adversarial parameter attack is stronger than previous parameter perturbation attacks in that not only the accuracy but also the robustness of DNNs are considered.

The goal of this paper is to demonstrate such type of potential threat to neural network security. Our inspiration came from an unexpected error in a License Plate Recognition System. A system that has been used for a long time suddenly cannot recognize dirty license plates, but clean license plates can still be recognized. So we suspect that the system

[1]Academy of Mathematics and Systems Science, Chinese Academy of Sciences [2]University of Chinese Academy of Sciences. Correspondence to: Xiao-Shan Gao <xgao@mmrc.iss.ac.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

has lost robustness due to some attacks but still maintains the recognition ability for the clean ones.

Let $\mathcal{F}_\Theta$ be a DNN with parameter Θ. A perturbed parameter Θ_a is called *adversarial parameter* of $\mathcal{F}_\Theta$, if the following conditions are satisfied 1) Θ_a is close to Θ according to certain measurement; 2) the accuracy of $\mathcal{F}_{\Theta_a}$ over the given data distribution is almost the same as that of $\mathcal{F}_\Theta$; 3) $\mathcal{F}_{\Theta_a}$ is much less robust than $\mathcal{F}_\Theta$, in that, $\mathcal{F}_{\Theta_a}$ has much more adversarial examples than $\mathcal{F}_\Theta$.

Condition 1) is to ensure that the attack can be implemented and essentially imperceptible; condition 2) is to make the attack more stealthy; and condition 3) is to make the new DNN less safe against adversarial attacks.

The existence of nearly perfect adversarial parameters is proved under certain assumptions. It is shown that if the width of certain DNN $\mathcal{F}_\Theta$ is sufficiently large, then $\mathcal{F}_\Theta$ has L_∞ norm adversarial parameters Θ_a sufficiently near Θ such that $\mathcal{F}_{\Theta_a}$ has adversarial samples sufficiently near a given benign sample (refer to Theorem 4.1), which implies that adversarial parameters are inevitable in a certain sense, similar to adversarial examples (Bastounis et al., 2021; Azulay & Weiss, 2018; Shafahi et al., 2018). It is also proved that when the width of the network is sufficiently large, by changing any small proportion of parameters, there exist L_0 norm adversarial parameters such that the attacked network $\mathcal{F}_{\Theta_a}$ has the same accuracy as $\mathcal{F}_\Theta$, but zero adversarial accuracy (refer to Theorem 4.5).

Finally, algorithms are given to compute adversarial parameters and numerical experiments are used to demonstrate that the algorithms are effective to produce high-quality adversarial parameters for frequently-used networks like VGG, deep VGG (Simonyan & Zisserman, 2014), ResNet (He et al., 2016), Wide-ResNet on the frequently-used datasets like CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009) and Tiny-ImageNet (Le & Yang, 2015). Furthermore, the original network $\mathcal{F}_\Theta$ is trained with various robust training methods including adversarial training (Madry et al., 2017), TRADES (Zhang et al., 2019), AWP (Wu et al., 2020b). The robustness of the attacked network $\mathcal{F}_{\Theta_a}$ is evaluated using various attack methods including PGD (Madry et al., 2017), target attack, Proxy model attack, and Autoattack (Croce & Hein, 2020). It is shown that the robustness of $\mathcal{F}_{\Theta_a}$ decreases significantly comparing with the original network.

2. Related work

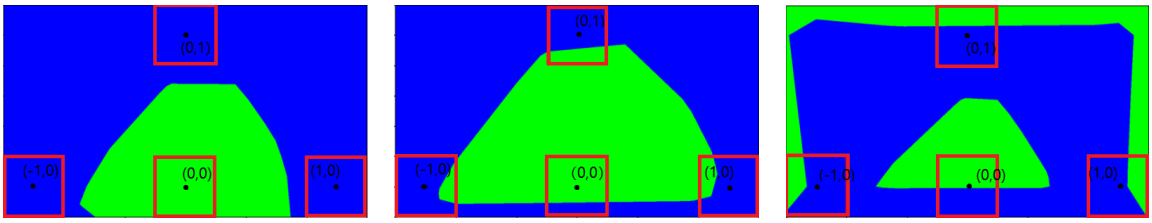
Parameter perturbation attacks. Parameter perturbation attacks were given under different names such as fault injection attack, fault sneaking attack, stealth attack, and weight corruption. The fault injection attack (Liu et al., 2017) was first proposed by Liu et al, and was further studied in (Breier et al., 2018; Zhao et al., 2019). In (Breier et al., 2018), the first physical fault injection attack on DNNs was given, by using the laser injection technique on embedded systems. In (Weng et al., 2020; Tsai et al., 2021; Sun et al., 2021), some security boundaries for parameter perturbations were given. In (Tyukin et al., 2020; 2021), the stealth attack was proposed to make the attacked DNN output a desired label for a given input image.

The adversarial parameter attack has the following advantages compared to previous works. First, by keeping the accuracy and reducing the robustness, the adversarial parameter attack is more difficult to be recognized. Second, we prove the existence of adversarial parameters under reasonable assumptions, while most previous works rely on experimental results.

Algorithms to train robust DNNs. Many methods were proposed to train more robust DNNs to defend adversarial samples (Xu et al., 2020). The adversarial training proposed by (Madry et al., 2017) is considered to be one of the most effective methods to train robust networks, and is used to compute adversarial parameters in this paper. Methods to train DNNs that are more robust against parameter perturbation attacks were also proposed (Liu et al., 2017; Zhao et al., 2019; Wu et al., 2020b). In (Wu et al., 2020a;b), the adversarial weight perturbation was proposed which was a generalization of the adversarial training by considering both the adversarial examples and the adversarial parameters, and hence led to more robust networks.

Theory of adversarial examples. The existence of adversarial examples was usually demonstrated with numerical experiments, and mathematical guaranteed results were desirable. Along this line of research, it was proved that a well-learned DNN always has adversarial examples for certain classification functions and data distributions (Bastounis et al., 2021). In (Tyukin et al., 2020; 2021), it was proved that there exist attacked DNNs that give a desired label for any sample.

Theories for certified robustness of DNNs were given in several aspects. In (Hein & Andriushchenko, 2017; Yu & Gao, 2022; Raghunathan et al., 2018), some security boundaries of adversaries were given. In (Cohen et al., 2019), the randomized smoothing method was proposed and security boundaries of adversaries were given. Lower bounds on stability in terms of the classification function were also given in (Shafahi et al., 2018; Tyukin et al., 2020). However, these safety bounds are usually very small when the depth of the DNN is large. In (Yu & Gao, 2021), the information-theoretically safe bias classifier was introduced by making the gradient of the DNN random. In this paper, we show that by making small perturbations to the parameters, the DNN will have adversarial examples with a high probability for any input.



(a) Original Network (b) Network after L_∞ parameters attack (c) Network after L_0 parameters attack

Figure 1. An illustrative example. For each network, the decision area for the data with label 0 (1) is marked green (blue). Adversarial examples are sought inside the squares with red edges, that is, we compute adversarial examples with budget 0.2 under L_∞ norm. The attacked networks (b) and (c) are accurate but not robust.

3. Compute adversarial parameters

In this section, we define the adversarial parameters and give algorithms to compute them.

3.1. Adversarial parameters of DNNs

In this paper, we assume that the data to be classified satisfy a distribution $\mathcal{D}_S$ over $\mathbb{D} = S \times [m]$, where $S \subset [0,1]^n$ and $[m] = \{1, \ldots, m\}$ is the label set. Finite datasets used for training and testing are chosen iid according to $\mathcal{D}_S$.

Let $\mathcal{F} : S \to \mathbb{R}^m$ be a classification DNN, where Relu is used as the activation function and the output layer does not have activation functions. Notice that the network $\mathcal{F}$ is defined over $\mathbb{R}^n$, but for the classification problem, only the values of $\mathcal{F}$ on S are needed.

Denote Θ to be the parameters of $\mathcal{F}$, and $\mathcal{F}$ is denoted as $\mathcal{F}_\Theta$ if the parameters need to be mentioned explicitly. Denote $\mathcal{F}_l(x)$ to be the l-th component of $\mathcal{F}(x)$ for $l \in [m]$ and $\widehat{\mathcal{F}}(x) = \arg\max_{l=1}^m \mathcal{F}_l(x)$ to be the classification result of $\mathcal{F}(x)$. The accuracy of $\mathcal{F}$ is

$$\mathrm{AC}(\mathcal{F}) = \mathbb{P}_{(x,y)\sim D_S}(\widehat{\mathcal{F}}(x) = y).$$

Given an adversarial budget ϵ, the following *adversarial accuracy* is used to measure the robustness of network $\mathcal{F}$

$$\mathrm{AA}(\mathcal{F}, \epsilon) = \mathbb{P}_{(x,y)\sim \mathcal{D}_S}(\widehat{\mathcal{F}}(x') = y, \forall x' \; s.t. ||x' - x||_\infty \le \epsilon).$$

Notice that we consider adversarial examples with L_∞ norm.

Adversarial Parameters. Let $\mathcal{F}_\Theta$ be a trained network with parameter set $\Theta \in \mathbb{R}^k$. Then $\Theta_a \in \mathbb{R}^k$ is called an *adversarial parameter* of Θ, if (1) the L_p distance between Θ_a and Θ is smaller than certain given bound to be given subsequently; (2) $\mathrm{AC}(\mathcal{F}_{\Theta_a}) \ge \gamma_{\mathrm{ac}} \mathrm{AC}(\mathcal{F}_\Theta)$ for a given $\gamma_{\mathrm{ac}} \in (0,1)$ which is close to 1, say $\gamma_{\mathrm{ac}} = 90\%$; and (3) $\mathrm{AA}(\mathcal{F}_{\Theta_a}, \epsilon)$ is much smaller than $\mathrm{AA}(\mathcal{F}_\Theta, \epsilon)$, that is, $\mathcal{F}_{\Theta_a}$ has much more adversarial examples than $\mathcal{F}_\Theta$.

Condition (1) is to make the changes of the parameter small, similar to the adversarial attack. Condition (2) is to make the attack more stealthy in that the attacked network perform normally on the training samples. Condition (3) is to make the network vulnerable to adversarial attacks, which is the ultimate goal of the attacks.

All parameters are stored in the form of 0,1 bits. If using the laser injection technique in the parameter attack, we need to modify the bits of the parameters. For faster and low-cost parameter attacks, we should consider modifying fewer bits in the attack. So we consider two ways to reduce the number of bits that need to be changed: (1) Change fewer bits for each parameter, corresponding to L_∞ norm attack. (2) Change fewer parameters to reduce the number of bits need to be changed, corresponding to L_0 attack.

L_∞ Norm Adversarial Parameters. For an attack budget ratio $\gamma \ll 1$, the adversarial parameters of Θ are taken in

$$\begin{aligned} &\mathbb{B}_\infty(\Theta, \gamma) = \\ &\{\Theta_a \in \mathbb{R}^k : |\Theta_a - \Theta|^{(i)} \le \gamma |\Theta|^{(i)}, \; \forall\, i \in [k]\} \end{aligned} \tag{1}$$

where $V^{(i)}$ is the i-th entry of a vector V. The above definition makes sure that no parameters are modified too much, which is more reasonable than using the standard L_∞ norm.

L_0 Norm Adversarial Parameters. For an attack budget ratio $\eta \ll 1$, the adversarial parameters of Θ are taken in

$$\mathbb{B}_0(\Theta, \eta) = \{\Theta_a \in \mathbb{R}^k : ||\Theta_a - \Theta||_0 \le \eta k\}. \tag{2}$$

An illustrative example for adversarial parameters is given in Figure 1. We consider a binary classification dataset $T = \{((0,0),0), ((1,0),1), ((0,1),1), ((0,1),1)\}$ and use the network $\mathcal{F} : \mathbb{R}^2 \to \mathbb{R}^2$ which has one hidden layer and width 24 to classify T. We compute the L_∞ norm and L_0 norm adversarial parameters of $\mathcal{F}$ and give the decision areas of each network in Figure 1. It can be seen that the original network $\mathcal{F}$ is robust: $\mathrm{AC}(\mathcal{F}) = 1$ and $\mathrm{AA}(\mathcal{F}, 0.2) = 1$, but the attacked networks $\mathcal{F}'$ are not robust but still accurate on the data: $\mathrm{AC}(\mathcal{F}') = 1$ and $\mathrm{AA}(\mathcal{F}', 0.2) = 0$.

3.2. Algorithms for adversarial parameters

According to the definition, adversarial parameters can be computed by solving the following optimization problem:

$$\arg\max_{\Theta_a \in \mathbb{B}_p(\Theta,\varepsilon)} \frac{\mathbf{I}(\mathrm{AC}(\mathcal{F}_{\Theta_a}) \geq \gamma_{\mathrm{ac}} \mathrm{AC}(\mathcal{F}_\theta))}{\mathrm{AA}(\mathcal{F}_{\Theta_a}, \epsilon)} \tag{3}$$

where $\mathbf{I}(t) = 1$ if t is true and $\mathbf{I}(t) = 0$ otherwise, and $p = \infty$ or 0. Once $\frac{\mathrm{AC}(\mathcal{F}_{\Theta_a})}{\mathrm{AC}(\mathcal{F}_\Theta)} < \gamma_{\mathrm{ac}}$, the numerator becomes 0 and the whole value is 0; so the solution Θ_a of optimization problem (3) must make $\mathrm{AA}(\mathcal{F}_{\Theta_a}, \epsilon)$ the smallest while ensuring $\frac{\mathrm{AC}(\mathcal{F}_{\Theta_a})}{\mathrm{AC}(\mathcal{F}_\Theta)} \geq \gamma_{\mathrm{ac}}$.

To solve optimization problem (3), following the common practices in deep learning, we will use the empirical loss to represent the accuracy and use the adversarial empirical loss to represent the adversarial accuracy.

For $(x, y) \sim \mathcal{D}_S$ and an adversarial budget $\epsilon \in \mathbb{R}_+$, we use PGD (Madry et al., 2017) to compute

$$\chi = \arg\max_{\delta \in \mathbb{R}^n, ||\delta||_\infty < \epsilon} L_{\mathrm{CE}}(\mathcal{F}(x+\delta), y), \tag{4}$$

and define the *adversarial loss* L_{AT} as

$$L_{\mathrm{AT}}(\mathcal{F}(x), y) = L_{\mathrm{CE}}(\mathcal{F}(x+\chi), y). \tag{5}$$

Then problem (3) can be approximated by the following optimization problem:

$$\arg\min_{\Theta_a \in \mathbb{B}_p(\Theta,\varepsilon)} \frac{\mathbb{E}_{(x,y)\sim\mathcal{D}_S} L_{\mathrm{CE}}(\mathcal{F}_{\Theta_a}(x), y)}{\mathbb{E}_{(x,y)\sim\mathcal{D}_S} L_{\mathrm{AT}}(\mathcal{F}_{\Theta_a}(x), y)} \tag{6}$$

which will be used to compute adversarial parameters. The condition $\mathrm{AC}(\mathcal{F}_{\Theta_a}) \geq \gamma_{\mathrm{ac}} \mathrm{AC}(\mathcal{F}_\Theta)$ will be checked after computing Θ_a.

Compute adversarial L_∞ norm parameters. To solve problem (6), we need a training set $T = \{(x_i, y_i), i \in [N]\}$ selected i.i.d. according to $\mathcal{D}_S$ and use the original parameters Θ as the initiation point. The training procedure consists of two phases:

Phase one: Preliminary training with the loss function

$$-\frac{1}{|T|} \sum_{(x,y)\in T} L_{\mathrm{AT}}(\mathcal{F}_\Theta(x), y). \tag{7}$$

Phase two: Main training with the loss function

$$\frac{\sum_{(x,y)\in T} L_{\mathrm{CE}}(\mathcal{F}_\Theta(x), y)}{\sum_{(x,y)\in T} L_{\mathrm{AT}}(\mathcal{F}_\Theta(x), y)} \tag{8}$$

which corresponds to optimization problem (6). A sketch of the algorithm is given below.

Algorithm 1 Adversarial Parameter Attack under L_∞ norm

Input:
The parameter $\Theta \in \mathbb{R}^k$ of $\mathcal{F}$;
The parameter attack ratio γ;
A training set T;
Hyper-parameters: $\alpha \in \mathbb{R}_+, n_1, n_2 \in \mathbb{N}$.
Output:
Adversarial parameter Θ_a in $\mathbb{B}_\infty(\Theta, \gamma)$.
Let $i = 0$, $\Theta_a = \Theta$.
For all $i \in [n_1 + n_2]$:
If $i < n_1$:
$L = -\frac{1}{|T|} \sum_{(x,y)\in T} L_{\mathrm{AT}}(\mathcal{F}_{\Theta_a}(x), y)$.
Else:
$L = \frac{\sum_{(x,y)\in T} L_{\mathrm{CE}}(\mathcal{F}_{\Theta_a}(x), y)}{\sum_{(x,y)\in T} L_{\mathrm{AT}}(\mathcal{F}_{\Theta_a}(x), y)}$.
$\widetilde{\Theta} = \Theta_a - \alpha \bigtriangledown L$.
$\Theta_a = \mathrm{Proj}(\widetilde{\Theta}, \mathbb{B}_\infty(\Theta, \gamma))$.
(Project $\widetilde{\Theta}$ into $\mathbb{B}_\infty(\Theta, \gamma)$)
Output: Θ_a.

Remark 3.1. Phase one is necessary. Otherwise, we may have the gradient vanishing problem in Phase two, as shown by experiments in Appendix C.

Algorithm 2 Adversarial Parameter Attack under L_0 norm

Input:
The parameter $\Theta \in \mathbb{R}^k$ of $\mathcal{F}$;
The parameter attack ratio η;
A training set T;
Hyper-parameters: $\alpha \in \mathbb{R}_+, n_1, n_2 \in \mathbb{N}$.
Ensure:
Adversarial parameter Θ_a in $\mathbb{B}_0(\Theta, \eta)$.
Let $i = 0$, $\Theta_a = \Theta$, $P_1 = \{\}$ and $P_2 = \{\}$;
Calculate $V_1 = |\bigtriangledown_{\Theta_a} \sum_{(x,y)\in T} L_{\mathrm{AT}}(\mathcal{F}_{\Theta_a}(x), y)| \in \mathbb{R}^k$;
For all $j \in [k]$, add j to P_1 if $V_1^{(j)}$ is within the maximum $\eta/2$ percent of all elements in V_1;
For all $i \in [n_1]$:
$L = -\sum_{(x,y)\in T} \frac{1}{|T|} L_{\mathrm{AT}}(\mathcal{F}_{\Theta_a}(x), y)$.
For each $j \in P_1$:
$\Theta_a^{(j)} = \Theta_a^{(j)} - \alpha(\bigtriangledown L)^{(j)}$.
Calculate $V_2 = |\bigtriangledown_{\Theta_a} \frac{\sum_{(x,y)\in T} L_{\mathrm{CE}}(\mathcal{F}_{\Theta_a}(x), y)}{\sum_{(x,y)\in T} L_{\mathrm{AT}}(\mathcal{F}_{\Theta_a}(x), y)}| \in \mathbb{R}^k$;
For all $j \in [k]$, add j to P_2 if $V_2^{(j)}$ is within the maximum $\eta/2$ percent of all elements of V_2;
For all $i \in \{n_1, n_1+1, \ldots, n_1+n_2\}$:
$L = \frac{\sum_{(x,y)\in T} L_{\mathrm{CE}}(\mathcal{F}_{\Theta_a}(x), y)}{\sum_{(x,y)\in T} L_{\mathrm{AT}}(\mathcal{F}_{\Theta_a}(x), y)}$.
For each $j \in P_2$:
$\Theta_a^{(j)} = \Theta_a^{(j)} - \alpha(\bigtriangledown L)^{(j)}$.
Output: Θ_a.

Compute L_0 norm adversarial parameters. The algorithm is similar to Algorithm 1. What we need to do additionally is to select positions of the parameters to be changed.

For that purpose, in each phase, we select the top $\eta/2$ parameters at which the gradient magnitudes are the largest and change the parameters at these positions during the training. The algorithm is given in Algorithm 2.

4. Existence of adversarial parameters

In this section, we prove that adversarial parameters with low adversarial accuracies exist under certain conditions. Proofs of theorems in this section are given in Appendix A.

Let $S \subset [0,1]^n$ and $\mathcal{D}_S$ a distribution defined on $S \times [m]$. In this section, we consider the following network $\mathcal{F}_\Theta : S \to \mathbb{R}^m$ with one hidden layer:

$$\mathcal{F}_\Theta(x) = W_2\,\mathrm{Relu}(W_1 x + b_1) + b_2, \tag{9}$$

where $W_1 \in \mathbb{R}^{n_1 \times n}, b_1 \in \mathbb{R}^{n_1}, W_2 \in \mathbb{R}^{m \times n_1}, b_2 \in \mathbb{R}^m$. $\Theta = \{W_i, b_i\}_{i=1}^2 \in \mathbb{R}^k$ is the parameter set of $\mathcal{F}_\Theta$, and $k = |\Theta| = (n+m+1)n_1 + m$.

4.1. L_∞ norm adversarial parameter

We introduce several notations. Let $||x||_{-\infty} = \min_{i\in[n]}\{|x_i|\}$ for $x \in \mathbb{R}^n$ and $W^{(i)}$ the i-th row of a matrix W. For $x \in S \subset \mathbb{R}^n$ and $\epsilon \in \mathbb{R}_{>0}$, the adversarial examples of x are taken in

$$S_\infty(x,\epsilon) = \{x' \in \mathbb{R}^n \,:\, ||x'-x||_\infty \le \epsilon\}.$$

We first consider a simple case: adversarial parameters for a given data. A small perturbation Θ_a of Θ is called *adversarial parameter* for a given data $(x,y) \sim \mathcal{D}_S$, if $\widehat{\mathcal{F}}_{\Theta_a}(x) = y$ and the robustness radius

$$\mathcal{R}(x) = \max\{\zeta \in \mathbb{R}_+ : \widehat{\mathcal{F}}_{\Theta_a}(x') = y, \forall x' \in S_\infty(x,\zeta)\}$$

of x is greatly reduced. The following theorem shows the existence of adversarial parameters for a given data $(x_0, y_0) \sim \mathcal{D}_S$.

Theorem 4.1. *Let $\mathcal{F}_\Theta : S \to \mathbb{R}^m$ be a trained network with structure in* (9)*, which gives the correct label y_0 for $x_0 \in S$. Further, assume the following conditions.*

- C_1. *$|\mathcal{F}_i(x) - \mathcal{F}_j(x)| < A$ for all $i,j \in [m]$, $x \in S_\infty(x_0,a)$, and $a, A \in \mathbb{R}_+$, that is, the difference between logits of any perturbed input is bounded.*
- C_2. *$||W_2^{(i)} - W_2^{(j)}||_{-\infty} > c$ for all $i,j \in [m]$, $i \ne j$ and $c \in \mathbb{R}_+$.*
- C_3. *At least ηn_1 coordinates of $|\mathrm{Relu}(W_1 x + b_1)|$ are bigger than b, where $\eta \in (0,1)$ and $b \in \mathbb{R}_+$.*

For $\forall \mu, \epsilon \in \mathbb{R}_+$ satisfying $\epsilon < a$, if $n_1 > \frac{2A}{\min\{\epsilon\mu(n-1), b\}c\eta}$, then there exists a $\Theta_a \in \mathbb{R}^k$ such that $||\Theta_a - \Theta||_\infty \le \mu$, $\widehat{\mathcal{F}}_{\Theta_a}(x_0) = y_0$, and $\mathcal{F}_{\Theta_a}$ has adversarial examples to x_0 in $S_\infty(x_0,\epsilon)$.

Remark 4.2. By Theorem 4.1, if the width n_1 of $\mathcal{F}_\Theta$ is sufficiently large, then Θ has adversarial parameters which are as close as possible to Θ, and $\mathcal{F}_{\Theta_a}$ has adversarial examples which are as close as possible to x_0. Thus, we may say that adversarial parameters are inevitable in this case.

Theorem 4.3. *Let $\mathcal{F}_\Theta : S \to \mathbb{R}^m$ be a trained DNN with structure in* (9) *and suppose that $\mathcal{F}$ and S satisfy the following conditions:*

- C_1. *$|\mathcal{F}_i(x) - \mathcal{F}_j(x)| < A$ for all $i,j \in [m]$ and $x \in \bigcup_{x_0 \in S} S_\infty(x_0,a)$, where $A, a \in \mathbb{R}_+$.*
- C_2. *$||W_2^{(i)} - W_2^{(j)}||_{-\infty} > c$ for all $i,j \in [m]$, $i \ne j$, where $c \in \mathbb{R}_+$.*
- C_3. *For all $x \in S$, at least ηn_1 coordinates of $|\mathrm{Relu}(W_1 x + b_1)|$ are bigger than b, where $\eta \in (0,1)$ and $b \in \mathbb{R}_+$.*
- C_4. *The dimension of S is lower than $n - m$.*

For $\forall \mu, \epsilon \in \mathbb{R}_+$ satisfying $\epsilon < a$, if $n_1 > \frac{2A}{min\{\epsilon\mu/m, b\}c\eta}$, then there exists a $\Theta_a \in \mathbb{R}^k$ such that $||\Theta_a - \Theta||_\infty \le \mu$, $\mathrm{AC}(\mathcal{F}_{\Theta_a}) = \mathrm{AC}(\mathcal{F}_\Theta)$, and $\mathrm{AA}(\mathcal{F}_{\Theta_a}, \epsilon) \le 0.5$.

By Theorem 4.3, when the width n_1 of $\mathcal{F}$ is sufficiently large, there exist adversarial parameters as close as possible to Θ, such that the corresponding network has adversarial accuracy at most 50%.

Remark 4.4. By training two-layer networks with widths ranging from 100 to 10000 on dataset MNIST, we have the following estimates for the parameters in the conditions of Theorems 4.1 and 4.3: $A < 30, c > 10^{-4}, b = 0.1, \eta > 0.2$. The lower bound of n_1 given by the Theorem 4.1(4.3) is about $\frac{10^5}{\epsilon\mu}(\frac{10^7}{\epsilon\mu})$, which is larger than that of the usually used networks.

4.2. L_0 norm adversarial parameter

The following theorem shows that there exist L_0 norm adversarial parameters which have zero adversarial accuracy.

Theorem 4.5. *Let $\mathcal{F}_\Theta : S \to \mathbb{R}^m$ be a trained DNN with structure in* (9) *and suppose that $\mathcal{F}$ and S satisfy the following conditions:*

- C_1. *Each row of W_2 has at least one negative component; each column of W_2 has at least one non-negative component.*
- C_2. *The dimension of S is lower than $n - m$.*

For any $\eta, \epsilon \in (0,1)$, if $n_1 \ge m/\eta$, then there exists a $\Theta_a \in \mathbb{B}_0(\Theta, \eta)$ such that $\mathrm{AC}(\mathcal{F}_{\Theta_a}) = \mathrm{AC}(\mathcal{F}_\Theta)$ and $\mathrm{AA}(\mathcal{F}_{\Theta_a}, \epsilon) = 0$.

Remark 4.6. Theorem 4.5 implies that there exist perfect adversarial parameters when the width of the network is sufficiently large.

Remark 4.7. The conditions of Theorems 4.5 can be satisfied in most cases. Since we usually have $m \ll n \leq n_1$, the n_1 satisfying $n_1 > m/\eta$ is within practical range, say for $\eta = 1\%$. The training procedure usually starts with a random initial point and uses stochastic gradient decent to update the weights. Its output is usually not far from the initial point according to our experimental observations, so condition C_1 is valid in most cases.

5. Experimental results

In this section, we verify the effectiveness of Algorithms 1, 2 and the theoretical results in Section 4 with experimental results. The codes of the experiments can be found in `https://github.com/EhanW/adversarial-parameter-attack`.

5.1. Setups

The original network $\mathcal{F}_\Theta$ is trained respectively with adversarial training (Madry et al., 2017), TRADES (Zhang et al., 2019), and AWP (Wu et al., 2020b), where PGD-(10,8/255) is used. We use networks VGG-16, VGG-24, VGG-32 (Simonyan & Zisserman, 2014), ResNet-18 (He et al., 2016), WRN-32-4, and WRN-32-10 (Zagoruyko & Komodakis, 2016). The datasets include CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), and Tiny-ImageNet (Le & Yang, 2015).

The adversarial accuracies of the attacked network $\mathcal{F}_{\Theta_a}$ on the test set are computed using PGD-(10, 8/255) (Madry et al., 2017) in Sections 5.2, 5.3, 5.5 and 5.6. In Section 5.4, we use four different attacks to measure the robustness. These four attacks are:

(1) Target attack. For (x, y) in the test set, we use PGD$(10, 8/255, (y+1) \mod 10)$ (10 steps, budget 8/255, target label $(y+1) \mod 10$) to find adversarial examples.

(2) Proxy model attack. This is a black box attack. We use $\{(x, \widehat{\mathcal{F}}_\Theta(x))\}$ to train three networks which have the same structure with $\mathcal{F}_\Theta$, compute adversarial examples with PGD-(10,8/255) for these models, and use them to attack $\mathcal{F}_\Theta$.

(3) Autoattack (Croce & Hein, 2020). Refer to the original paper (Croce & Hein, 2020) for details, where budget $8/255$ is used.

(4) Random Attack. For each sample, use the normal distribution $\mathbb{N}(0, (8/255)^2)^n$ to randomly select noises and add them to the sample, where n is the dimension of the samples.

We give the training details below. For more details about the training, please refer to Appendix B.

Training Details for Algorithm 1. In Phase one, we train with 10 epochs, and each epoch has learning rate 0.1. We monitor the accuracy on the training set during the training. Once the accuracy on the training set is lower than 20% after an epoch, we terminate Phase one and go to the Phase two. In Phase two, we train with 40 epochs, and each epoch has learning rate 0.002. The learning rate reduces by half at the 20-th epoch.

Training Details for Algorithm 2. In Phase one, we train with 10 epochs, and each epoch has learning rate 0.01. To prevent gradient explosion, we limit the change of each parameter to 0.01 for each gradient descent. At the same time, we monitor the loss function on the training set as training goes on. Once the value of the loss function of a batch is ≥ 50, terminate Phase one and go to Phase two.

In Phase two, we train with 40 epochs, and each epoch has learning rate 0.05. At 21-th and 31-th epochs, the learning rate reduces by half. To prevent gradient explosion, we limit the change of each parameter to 0.01 for each gradient descent.

Generally speaking, training a robust network on CIFAR-10 dataset needs 100-200 epochs, but for Algorithms 1 and 2, we achieve good results after 50 epochs, so training adversarial parameters is faster than normal training.

For convenience, we write the L_∞ norm adversarial parameter attack with ratio γ as $L_{\infty,\gamma}$ and the L_0 norm adversarial parameter attack with ratio η as $L_{0,\eta}$. Refer (1) and (2) for details. The parameter γ_{ac} in optimization problem (3) is set to be 90%.

5.2. Experiments with various networks

In this section, we test Algorithms 1 and 2 with six networks and different attacking budgets for CIFAR-10. The results are given in Tables 1 and 2.

From Tables 1 and 2, the accuracies of the attacked networks are larger than $\gamma_{\text{ac}} = 90\%$ of that of the original networks, so condition (2) in the definition of adversarial parameters is always satisfied. Algorithm 1 generates high quality L_∞ norm adversarial parameters when $\gamma \geq 10\%$: the adversarial accuracies decline to about 15% of the original ones. Similarly, Algorithm 2 generates high quality $L_{0,\eta}$ adversarial parameters when $\eta \geq 0.75\%$. For ResNet-18, WRN-32-4, and WRN-32-10, the L_0 attacks achieve near zero adversarial accuracies, as stated in Theorem 4.5.

Overall, our algorithms generate high-quality adversarial parameters effectively for CIFAR-10, since high-quality adversarial parameters can be generated with quite small ratios $\gamma = 10\%$ and $\eta = 0.75\%$.

Table 1. Adversarial parameter attack for networks VGG-16 and ResNet-18 on CIFAR-10. AC: test accuracy, AA: adversarial accuracy on the test set.

Attack	VGG-16		ResNet-18	
	AC	AA	AC	AA
No attack	80%	39%	84%	52%
$L_{\infty,2\%}$	76%	30%	86%	48%
$L_{\infty,4\%}$	75%	22%	84%	37%
$L_{\infty,6\%}$	76%	15%	85%	24%
$L_{\infty,8\%}$	76%	10%	83%	14%
$L_{\infty,10\%}$	77%	6%	85%	6%
$L_{0,0.25\%}$	75%	24%	82%	5%
$L_{0,0.5\%}$	76%	19%	83%	0%
$L_{0,0.75\%}$	75%	17%	83%	2%
$L_{0,1.0\%}$	77%	16%	84%	2%

Table 2. Adversarial parameter attack for networks Deep VGG and WRN-32 on CIFAR-10.

Attack	VGG-24		VGG-32	
	AC	AA	AC	AA
No attack	80%	37%	78%	38%
$L_{\infty,2\%}$	75%	28%	75%	25%
$L_{\infty,6\%}$	77%	15%	74%	11%
$L_{\infty,10\%}$	79%	4%	76%	4%
$L_{0,0.5\%}$	78%	19%	76%	16%
$L_{0,1.0\%}$	76%	16%	75%	13%
Attack	WRN-32-4		WRN-32-10	
	AC	AA	AC	AA
No attack	86%	54%	88%	54%
$L_{\infty,2\%}$	88%	48%	88%	47%
$L_{\infty,6\%}$	87%	19%	88%	24%
$L_{\infty,10\%}$	85%	2%	89%	4%
$L_{0,0.5\%}$	84%	0%	87%	0%
$L_{0,1.0\%}$	83%	5%	86%	5%

Comparing the results of VGG-16, VGG-24, and VGG-32, we find that when the depth increases, the networks become easier to attack. On the other hand, the results of WRN-32-4 and WRN-32-10 show that the attacks are not sensitive to the width.

To generate adversarial parameters for a given data as described in Theorem 4.1 is much easier than for a dataset, as expected. Algorithms 1 and 2 still work in this case, where T contains a single data. We conduct a simple experiment to verify this. A data (x, y) is called *robust* if we cannot find adversarial examples for it using PGD-(10, 8/255) with L_∞ norm. We take a *robust* sample (x, y) from the test set, use Algorithms 1 and 2 to generate $L_{\infty,2\%}$ and $L_{0,1\%}$ norm adversarial parameters for networks VGG-16 and ResNet-18, respectively. Our experiments show that for all 100 such data, the attacked networks still give the correct label, but are not robust anymore.

5.3. Networks trained with more defense methods

In Section 5.2, we show that our algorithms can compute high-quality adversarial parameters if the original network is trained with adversarial training. In this section, we further show that if the original network is trained with other defenses such as TRADES (Zhang et al., 2019) or AWP (Wu et al., 2020b), the algorithms still work. The results for VGG-16 and ResNet-18 are given in Figure 2, where only the adversarial accuracies are given and the accuracies do not decrease by more than 10%.

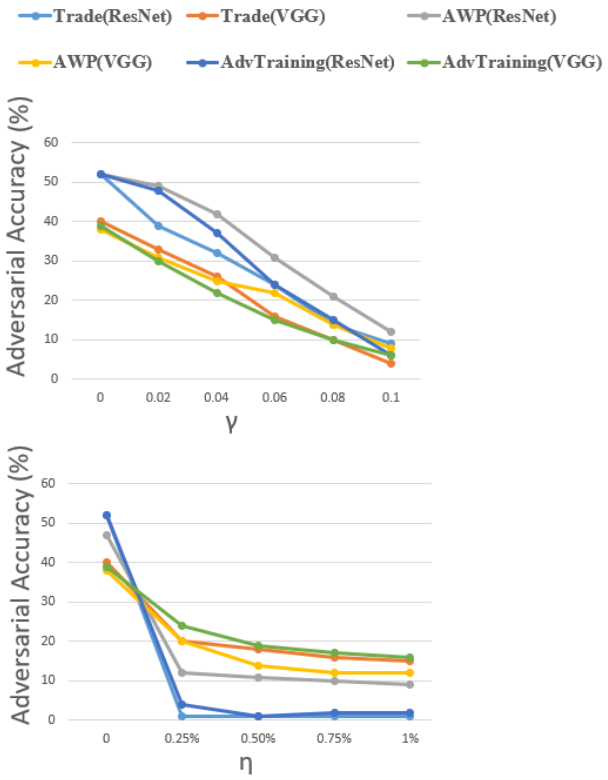


Figure 2. Experiments for networks VGG-16 and ResNet-18 trained with TRADES, AWP, and adversarial training. Up figure is for $L_{\infty,\gamma}$ norm parameter attack. Down figure is for $L_{0,\eta}$ norm attack.

From Figure 2, the adversarial parameter attacks also work for networks trained with TRADES and AWP. For the $L_{\infty,10\%}$ and $L_{0,1\%}$ attacks, the adversarial accuracies drop below 15% and are smaller than 30% of original adversarial accuracies on average. Note that AWP is designed to defend both adversarial examples and parameter perturba-

tions. It performs better than the other two defenses in most cases, particularly for the L_0 attack of ResNet-18, where the adversarial accuracies drop to about 10%, instead of 0%.

5.4. Experiments with more attack methods

In Sections 5.2 and 5.3, we measure the robustness of the attacked network $\mathcal{F}_{\Theta_a}$ by calculating the adversarial accuracy with PGD-10. In this section, we conduct experiments with three more adversarial attacks on the attacked network $\mathcal{F}_{\Theta_a}$: the target attack, the Proxy model attack, and the Autoattack (Croce & Hein, 2020). We also consider the random attack.

From Figure 3, we can see that as the attack budget ratios increase, the adversarial accuracy gradually decreases except for the random attack. Random attack is a very weak attack, which can hardly produce adversarial samples. However, random noises are often encountered in applications, and the above results show that even the attacked network has certain robustness against random noises.

We can see that Autoattack is the most effective attack method for the networks with adversarial parameters. PGD-10 has similar power to Autoattack, and the other two methods are relatively weak. For Autoattack, the adversarial accuracy is $4\%(5\%)$ for $L_{\infty,10\%}(L_{0,2\%})$ attacked network, and the adversarial accuracy declines to about $14\%(17\%)$ of the original adversarial accuracy. The adversarial accuracies of the $L_{\infty,10\%}(L_{0,2\%})$ attacked networks for PGD attack, Target attack, Proxy model attack decline to $15\%(21\%)$, $56\%(55\%)$, $63\%(57\%)$ of the original adversarial accuracy, respectively.

From Sections 5.2, 5.3, and 5.4, we experimentally verify our observations in Section 4: **adversarial parameters are inevitable for commonly used DNNs in a certain sense**, similar to the inevitability of adversarial examples.

5.5. Generate adversarial parameters with small training set

In the above experiments, the training set to generate adversarial parameters is the total CIFAR-10 training set. Sometimes, the adversary does not have many resources. So in this section, we will show that a smaller training set is enough to find adversarial parameters.

We randomly select a subset T of the training set of CIFAR-10, which contains 1/10 samples of the total training set. We use T as the training set to get adversarial parameters. The results are given in Figure 4 and Figure 5.

From Figure 4, we can see that even with a small training set, Algorithms 1 and 2 still work. For L_∞ norm attacks, the results for smaller training sets are slightly worse than that of the whole training set. For L_0 norm attacks, the results are almost the same.

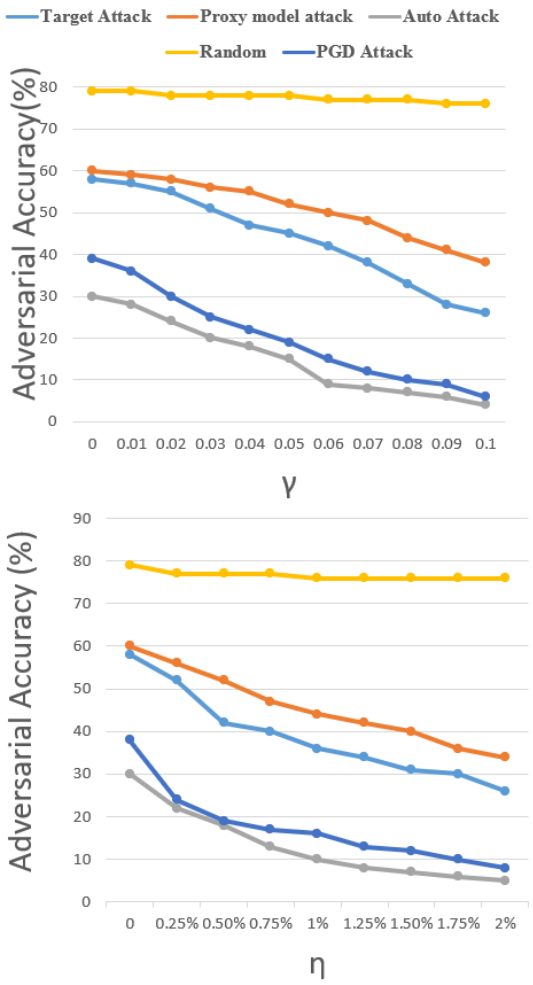


Figure 3. Adversarial accuracies of network VGG-16 on CIFAR-10 for the target attack, the Proxy model attack, and the Autoattack, the PGD attack, and random attack. Up figure is for the $L_{\infty,\gamma}$ attack. Down figure is for the $L_{0,\eta}$ attack.

5.6. Experiments for more datasets

In this section, we will show that adversarial parameters can be found for datasets CIFAR-100 and Tiny-ImageNet.

For each dataset, we use 100 epochs of adversarial training to train network $\mathcal{F}_{\Theta_a}$ and use PGD-(10,8/255) to attack $\mathcal{F}_{\Theta_a}$. The network is WRN-34-10, which is the commonly used network for these two datasets. Then compute $L_{\infty,6\%}$, $L_{\infty,10\%}$, $L_{0,0.5\%}$, $L_{0,1\%}$ types adversarial parameters with Algorithms 1 and 2. The results are given in Table 3. From Table 3, Algorithms 1 and 2 still work for CIFAR-100 and Tiny-ImageNet. For CIFAR-100 (Tiny-ImageNet), the adversarial accuracies of $L_{\infty,10\%}$ and $L_{0,1\%}$ attacked networks are 32% (26%) and 32% (42%) of that of the original network, which are significant declines although

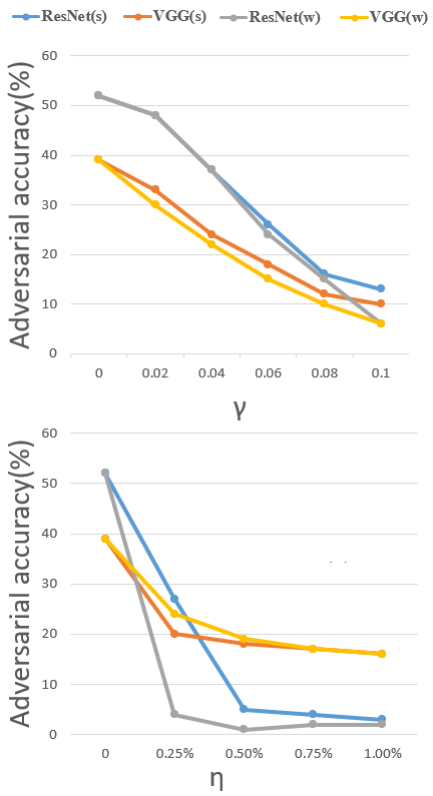


Figure 4. Adversarial parameter attack for VGG-16 and ResNet-18 on CIFAR-10 dataset, the variation of adversarial accuracy with attack budget. (s) means that a small training set is used in the attack. (w) means that the whole training set is used in the attack. Up figure is for the $L_{\infty,\gamma}$ norm parameter attack. Down figure is for the $L_{0,\eta}$ norm attack.

Table 3. Adversarial parameter attack of network WRN-34-10 for datasets CIFAR-100 and Tiny-ImageNet.

	CIFAR-100		Tiny-ImageNet	
Attack	AC	AA	AC	AA
No attack	58%	28%	40%	19%
$L_{\infty,6\%}$	55%	14%	36%	6%
$L_{\infty,10\%}$	57%	9%	38%	5%
$L_{0,0.5\%}$	56%	14%	40%	11%
$L_{0,1\%}$	56%	9%	40%	8%

not as good as that for CIFAR-10. Notice that the adversarial accuracies for these two datasets are known to be low.

6. Conclusion

The adversarial parameter attack for DNNs is proposed. In the attack, the adversary makes small changes to the parameters of a trained DNN such that the attacked DNNs will keep the accuracy of the original DNN as much as possible, but will make the robustness against adversarial attacks as low as possible. The goal of the attack is that the attacked DNN is more stealthy to the user and at the same time the robustness of the DNN is broken. Effective algorithms are given which can be used to produce high-quality adversarial parameters in a variety of scenarios. The existence of nearly perfect adversarial parameters in several cases is proved under reasonable conditions.

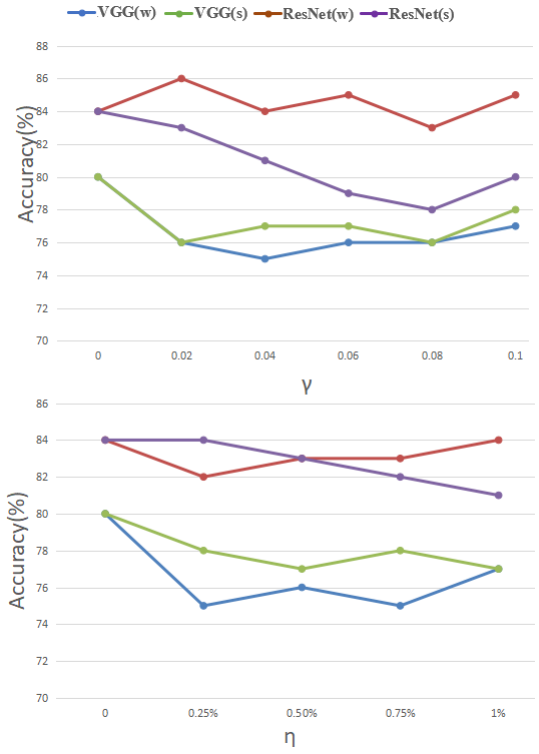


Figure 5. Adversarial parameter attack for VGG-16 and ResNet-18 on CIFAR-10 dataset, the variation of accuracy with attack budget. (s) means that a small training set is used in the attack. (w) means that the whole training set is used in the attack. Up figure is for the $L_{\infty,\gamma}$ norm parameter attack. Down figure is for the $L_{0,\eta}$ norm attack.

Limitations. The parameters attack method proposed in this paper can only be established for white-box attacks, and black box attacks need to be further studied. In the theoretical aspect, it is interesting to improve the bound in Theorem 4.3 to any given percentage. Moreover, we have only proven the existence of adversarial parameters on two-layer networks, and how to extend the results to deep networks remains a problem.

Acknowledgement. This work is partially supported by NKRDP grant No.2018YFA0306702 and NSFC grant No.11688101. We want to thank the anonymous referees for their valuable comments.

References

Akhtar, N. and Mian, A. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

Azulay, A. and Weiss, Y. Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*, 2018.

Bai, T., Luo, J., and Zhao, J. Recent advances in understanding adversarial robustness of deep neural networks. *arXiv preprint arXiv:2011.01539*, 2020.

Bastounis, A., Hansen, A. C., and Vlačić, V. The mathematics of adversarial attacks in ai–why deep learning is unstable despite the existence of stable neural networks. *arXiv preprint arXiv:2109.06098*, 2021.

Breier, J., Hou, X., Jap, D., Ma, L., Bhasin, S., and Liu, Y. Practical fault attack on deep neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2204–2206, 2018.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. Ieee, 2017.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pp. 1310–1320. PMLR, 2019.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pp. 2206–2216. PMLR, 2020.

Gao, X.-S., Liu, S., and Yu, L. Achieving optimal adversarial accuracy for adversarial deep learning using stackelberg games. *arXiv preprint arXiv:2207.08137*, 2022.

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

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hein, M. and Andriushchenko, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. *Advances in neural information processing systems*, 30, 2017.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. *Technical Report TR-2009*, 2009.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Liu, Y., Wei, L., Luo, B., and Xu, Q. Fault injection attack on deep neural network. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 131–138. IEEE, 2017.

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

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

Shafahi, A., Huang, W. R., Studer, C., Feizi, S., and Goldstein, T. Are adversarial examples inevitable? *arXiv preprint arXiv:1809.02104*, 2018.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sun, X., Zhang, Z., Ren, X., Luo, R., and Li, L. Exploring the vulnerability of deep neural networks: A study of parameter corruption. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 11648–11656, 2021.

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

Tsai, Y.-L., Hsu, C.-Y., Yu, C.-M., and Chen, P.-Y. Formalizing generalization and robustness of neural networks to weight perturbations. *arXiv preprint arXiv:2103.02200*, 2021.

Tyukin, I. Y., Higham, D. J., and Gorban, A. N. On adversarial examples and stealth attacks in artificial intelligence systems. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6. IEEE, 2020.

Tyukin, I. Y., Higham, D. J., Bastounis, A., Woldegeorgis, E., and Gorban, A. N. The feasibility and inevitability of stealth attacks. *arXiv preprint arXiv:2106.13997*, 2021.

Weng, T.-W., Zhao, P., Liu, S., Chen, P.-Y., Lin, X., and Daniel, L. Towards certificated model robustness against weight perturbations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 6356–6363, 2020.

Wu, D., Xia, S., and Wang, Y. Adversarial weight perturbation helps robust generalization. In *Proc. NeuelPS*, 2020a.

Wu, D., Xia, S.-T., and Wang, Y. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33:2958–2969, 2020b.

Xu, H., Ma, Y., Liu, H.-C., Deb, D., Liu, H., Tang, J.-L., and Jain, A. K. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, 2020.

Yu, L. and Gao, X.-S. Robust and information-theoretically safe bias classifier against adversarial attacks. *arXiv preprint arXiv:2111.04404*, 2021.

Yu, L. and Gao, X.-S. Improve robustness and accuracy of deep neural network with $l_{2,\infty}$ normalization. *Journal of Systems Science and Complexity*, pp. 1–26, 2022.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pp. 7472–7482. PMLR, 2019.

Zhang, X.-Y., Liu, C.-L., and Suen, C. Y. Towards robust pattern recognition: A review. *Proceedings of the IEEE*, 108(6):894–922, 2020.

Zhao, P., Wang, S., Gongye, C., Wang, Y., Fei, Y., and Lin, X. Fault sneaking attack: A stealthy framework for misleading deep neural networks. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2019.

A. Proofs for theorems in section 4

We first introduce several notations. Let $\mathbf{1}$ be the vector all of whose components are 1 and $\mathbf{1}_i$ the vector whose i-th component is 1 and all other components are 0. For a vector U, denote $U^{(i)}$ to be the i-th component of U. For a matrix W, denote $W^{(i)}$ to be the i-th row of W and $W^{(i,j)}$ the entry of W at the i-th row and j-th column.

For a network $\mathcal{F}: S \to \mathbb{R}^m$ and $S \subset [0,1]^n$, we use $\mathcal{F}_i(x)$ to denote the i-th component of $\mathcal{F}(x)$ for $i \in [m]$.

A.1. Proof of Theorems 4.1

A lemma is proved first.

Lemma A.1. *Let $v \in \mathbb{R}^n$ and $v \neq 0$. Then, there exists a $w \in \mathbb{R}^n$ such that $w \perp v$, $||w||_\infty = 1$, and $||w||_2 \geq \sqrt{n-1}$.*

Proof. Let $v = (v_1, \ldots, v_n)^\tau$ and $P = \arg\min_{P \subseteq [n]}\{|\sum_{i\in P}|v_i| - \sum_{i\in[n]\setminus P}|v_i|\}$. We can assume $\sum_{i\in P}|v_i| - \sum_{i\in[n]\setminus P}|v_i| = k \geq 0$. For any $j \in P$ such that $v_j \neq 0$ and $P_1 = S/\{j\}$, we have $|\sum_{i\in P_1}|v_i| - \sum_{i\in[n]/P_1}|v_i|| = |2|v_j| - k| \geq k$, which implies $k \leq |v_j|$.

We now define $w = (w_1, \ldots, w_n)^\tau \in \mathbb{R}^n$. Set $w_i = 1$ if $v_i = 0$. Select a $j \in P$ such that $v_j \neq 0$ and let $w_i = \text{sign}(v_i)$ if $i \in P/\{j\}$ and $v_i \neq 0$, and $w_j = \frac{-\sum_{i\in P}|v_i| + \sum_{i\in[n]\setminus P}|v_i| + |v_j|}{v_j}$. For $i \in [n] \setminus P$, let $w_i = -\text{sign}(v_i)$ if $v_i \neq 0$. It is easy to check that $||w||_\infty = 1$, $||w||_2 \geq \sqrt{n-1}$, and $w \perp v$. The lemma is proved. □

We now prove Theorem 4.1.

Proof. By Lemma A.1, there exists a vector $v \in \mathbb{R}^n$ such that $v \perp x_0$, $||v||_2 \geq \sqrt{n-1}$ and $||v||_\infty = 1$. Moreover, we can assume that at least $\eta n_1/2$ coordinates of $\text{Relu}(W_1(x+\epsilon v)+b_1)$ are bigger than b. If this is not valid, we just need to change v to $-v$, and then $\text{Relu}(W_1(x+\epsilon v)+b_1) + \text{Relu}(W_1(x-\epsilon v)+b_1) \geq 2\text{Relu}(W_1x+b_1)$, since $\text{Relu}(x) + \text{Relu}(y) \geq \text{Relu}(x+y)$ for all $x, y \in \mathbb{R}$. By condition C_3, at least ηn_1 coordinates of $2\text{Relu}(W_1x+b_1)$ are bigger than $2b$, but fewer than $\eta n_1/2$ coordinates of $\text{Relu}(W_1(x+\epsilon v)+b_1)$ are bigger than b, so at least $\eta n_1/2$ coordinates of $\text{Relu}(W_1(x-\epsilon v)+b_1)$ are bigger than b.

Let $l_2 \in [m]$ such that $l_2 \neq y_0$, $\overline{W}_2 = -(W_2^{(y_0)} - W_2^{(l_2)}) \in \mathbb{R}^{1\times n_1}$, $U_v \in \mathbb{R}^{n_1 \times n}$ all of whose rows are μv^τ (the transposition of v), and $U = \text{diag}(\text{sign}(\overline{W}_2)) \in \mathbb{R}^{n_1\times n_1}$. Let $\widetilde{W}_1 = W_1 + UU_v$ and

$$\mathcal{G}(x) = W_2\text{Relu}(\widetilde{W}_1 x + b_1) + b_2.$$

We will show that $\mathcal{G}$ satisfies the condition of the theorem.

Since $v \perp x_0$, we have $\mathcal{G}(x_0) = \mathcal{F}(x_0)$ and $\mathcal{G}$ gives the correct label for x_0. Since $||v||_\infty = 1$, we have $||\widetilde{W}_1 - W_1||_\infty = ||UU_v||_\infty = ||U_v||_\infty = ||\mu v||_\infty \leq \mu$, and thus $||\Theta_a - \Theta||_\infty \leq \mu$.

So it suffices to show that $\mathcal{G}(x_0+\epsilon v)$ will not give $x_0 + \epsilon v$ label y_0, which means that $\mathcal{G}$ has adversarial samples to x_0 in $S_\infty(x_0, \epsilon)$. Since

$$\begin{aligned} & \mathcal{G}(x_0+\epsilon v) \\ = \; & W_2\text{Relu}(\widetilde{W}_1(x_0+\epsilon v)+b_1)+b_2 \\ = \; & W_2\text{Relu}(W_1(x_0+\epsilon v)+b_1+\epsilon UU_v v)+b_2 \end{aligned}$$

we have

$$\begin{aligned} & \mathcal{G}_{y_0}(x_0+\epsilon v) - \mathcal{G}_{l_2}(x_0+\epsilon v) \\ = \; & \mathcal{F}_{y_0}(x_0+\epsilon v) - \mathcal{F}_{l_2}(x_0+\epsilon v) + \\ & \overline{W}_2(\text{Relu}(W_1(x+\epsilon v)+b_1) - \text{Relu}(W_1(x+\epsilon v)+b_1+\epsilon UU_v v)). \end{aligned}$$

Since $e(\text{Relu}(f) - \text{Relu}(e+f)) = -|e|(|\text{Relu}(f) - \text{Relu}(e+f))|$ for all $e, f \in \mathbb{R}$, we have

$$\begin{aligned} & -\overline{W}_2(\text{Relu}(W_1(x+\epsilon v)+b_1) - \text{Relu}(W_1(x+\epsilon v)+b_1+\epsilon UU_v v)) \\ = \; & |\overline{W}_2|(|\text{Relu}(W_1(x+\epsilon v)+b_1) - \text{Relu}(W_1(x+\epsilon v)+b_1+\epsilon UU_v v)|). \end{aligned}$$

Since $\epsilon UU_v v = \epsilon\mu||v||_2^2 \text{sign}(\overline{W}_2)$ and $||v||_2 \geq \sqrt{n-1}$, each weight of $|\epsilon UU_v v|$ is at least $\epsilon\mu(n-1)$. Note that if $e > 0$ and $f \in \mathbb{R}$, then $|\text{Relu}(e) - \text{Relu}(e-f)| \geq \min\{e, |f|\}$. As a consequence, if i satisfies $(\text{Relu}(W_1(x+\epsilon v)+b_1))_i > b$, then $(|\text{Relu}(W_1(x+\epsilon v)+b_1) - \text{Relu}(W_1(x+\epsilon v)+b_1+\epsilon UU_v v)|)_i \geq \min\{\epsilon\mu(n-1), b\}$. Since at least $\eta n_1/2$ coordinates of $\text{Relu}(W_1(x+\epsilon v)+b_1)$ are bigger than b, we have $||\text{Relu}(W_1(x+\epsilon v)+b_1) - \text{Relu}(W_1(x+\epsilon v)+b_1+\epsilon UU_v v)||_1 \geq \eta n_1/2 \min\{\epsilon\mu(n-1), b\}$. By condition C_2, it is easy see

$$\begin{aligned} & -\overline{W}_2(\text{Relu}(W_1(x+\epsilon v)+b_1) - \text{Relu}(W_1(x+\epsilon v)+b_1+\epsilon UU_v v)) \\ = \; & |\overline{W}_2|(|\text{Relu}(W_1(x+\epsilon v)+b_1) - \text{Relu}(W_1(x+\epsilon v)+b_1+\epsilon UU_v v)|) \\ \geq \; & ||\overline{W}_2||_{-\infty}||\text{Relu}(W_1(x+\epsilon v)+b_1) - \text{Relu}(W_1(x+\epsilon v)+b_1+\epsilon UU_v v)||_1 \\ \geq \; & \min\{\epsilon\mu(n-1), b\} c n_1 \eta/2, \end{aligned}$$

that is, $\overline{W}_2(\text{Relu}(W_1(x+\epsilon v)+b_1) - \text{Relu}(W_1(x+\epsilon v)+b_1+\epsilon UU_v v)) \leq -\min\{\epsilon\mu(n-1), b\} c n_1 \eta/2$. By condition C_1, we have $\mathcal{F}_{y_0}(x_0+\epsilon v) - \mathcal{F}_{l_2}(x_0+\epsilon v) \leq A$. Then we have

$$\begin{aligned} & \mathcal{G}_{y_0}(x_0+\epsilon v) - \mathcal{G}_{l_2}(x_0+\epsilon v) \\ = \; & \mathcal{F}_{y_0}(x_0+\epsilon v) - \mathcal{F}_{l_2}(x_0+\epsilon v) + \\ & \overline{W}_2(\text{Relu}(W_1(x+\epsilon v)+b_1) - \text{Relu}(W_1(x+\epsilon v)+b_1+\epsilon UU_v v)) \\ \leq \; & A - \min\{\epsilon\mu(n-1), b\} c n_1 \eta/2 < 0. \end{aligned}$$

Thus if $n_1 > \frac{2A}{\min\{\epsilon\mu(n-1),b\}c\eta}$, then $\mathcal{G}_{y_0}(x_0+\epsilon v) - \mathcal{G}_{l_2}(x_0+\epsilon v) < 0$ and the label of $\mathcal{G}(x_0+\epsilon v)$ is not y_0. The theorem is proved. □

A.2. Proofs of Theorems 4.3

We first prove a lemma.

Lemma A.2. *Let $W \in \mathbb{R}^{n\times m}$, $x \in \mathbb{R}^m$, $W_1 \in \mathbb{R}^{1\times n}$, and $b \in \mathbb{R}^n$. If $\text{sign}(Wx+b-c\mathbf{1})$ has at least k weights which are 1 for a $c > 0$ and $k \in \mathbb{Z}$, then*

$$W_1\text{Relu}(Wx+b) > W_1\text{Relu}(Wx+b-\gamma\text{sign}(W_1)) + k||W_1||_{-\infty}\min\{c,\gamma\}.$$

Proof. We first have two simple results:

(r1): for all $a > 0$ and $b \in \mathbb{R}$, it holds that $\text{Relu}(a) - \text{Relu}(a-b) = \min\{a, b\}$.

(r2): for all $a, b, c \in \mathbb{R}$, if $bc \geq 0$, then it holds that $b(\text{Relu}(a) - \text{Relu}(a-c)) \geq 0$, because Relu is a monotonally increasing function.

For the j-th weight of $Wx+b$, we have that

(1) If $(Wx+b)^{(j)} > c$, then by (r1), it holds $\text{Relu}(Wx+b)^{(j)} - \text{Relu}(Wx+b-\gamma\text{sign}(W_1))^{(j)} = \min\{(Wx+b)^{(j)}, \gamma\text{sign}(W_1)\}$. So $(W_1)^{(j)}(\text{Relu}(Wx+b)^{(j)} - \text{Relu}(Wx+b-\gamma\text{sign}(W_1))^{(j)}) = (W_1)^{(j)}\min\{(Wx+b)^{(j)}, \gamma\text{sign}(W_1)\} \geq ||W_1||_{-\infty}\min\{c,\gamma\}$.

(2) For all $j \in [n]$, by (r2), we have that $(W_1)^{(j)}(\text{Relu}(Wx+b)^{(j)} - \text{Relu}(Wx+b-\gamma\text{sign}(W_1))^{(j)}) \geq 0$.

Since at least k coordinate j satisfying $(Wx+b)^{(j)} > c$, it holds

$$W_1\text{Relu}(Wx+b) - W_1\text{Relu}(Wx+b-\gamma\text{sign}(W_1)) \geq k||W_1||_{-\infty}\min\{c,\gamma\}.$$

The lemma is proved. □

We now prove Theorem 4.3.

Proof. By condition C_4, for $l \in [m]$, there exist a $v_l \in \mathbb{R}^n$ such that $v_l \perp S$, $v_l \perp v_k$ for $l \neq k$, and $||v_l||_2 = 1$. Then $||v_l||_\infty \leq 1$.

By condition C_3, for any $x \in S$, at least $\eta n_1/2$ coordinates of $\text{Relu}(W_1(x+\epsilon v_y)+b_1)$ are bigger than b or at least $\eta n_1/2$ coordinates of $\text{Relu}(W_1(x-\epsilon v_y)+b_1)$ are bigger than b. This is because, from $\text{Relu}(a)+\text{Relu}(b) \geq \text{Relu}(a+b)$ we have

$$(\text{Relu}(W_1(x+\epsilon v_y)+b_1)+\text{Relu}(W_1(x-\epsilon v_y)+b_1))/2 \geq \text{Relu}(W_1x+b_1).$$

Thus from C_3, at least ηn_1 coordinates of $\text{Relu}(W_1x+b_1)$ are bigger than b, so we obtain the result.

For convenience, we define a function $G(x,y) : (\mathbb{R}^n, \mathbb{R}) \to \mathbb{R}$ as $G(x,y) = ||\text{sign}(x-y\mathbf{1})||_0$. It is easy to see that, $G(x,b)$ is the number of coordinates of x that are bigger than b. We thus have $G(\text{Relu}(W_1(x+\epsilon v_y)+b_1),b) \geq \eta n_1/2$ or $G(\text{Relu}(W_1(x-\epsilon v_y)+b_1),b) \geq \eta n_1/2$ for all x, and hence for $l \in [m]$, we have

$$P_{(x,y)\sim D_s}(G(\text{Relu}(W_1(x+\epsilon v_y)+b_1),b) \geq \eta n_1/2 \text{ or } G(\text{Relu}(W_1(x-\epsilon v_y)+b_1),b) \geq \eta n_1/2) = 1.$$

For events e and f, it is easy to see that $\mathbb{P}(e \text{ or } f) \leq \mathbb{P}(e)+\mathbb{P}(f)$. We thus have $\mathbb{P}_{(x,y)\sim D_S}(G(\text{Relu}(W_1(x+\epsilon v_y)+b_1),b) \geq \eta n_1/2) \geq 0.5$ or $\mathbb{P}_{(x,y)\sim D_S}(G(\text{Relu}(W_1(x-\epsilon v_y)+b_1),b) \geq \eta n_1/2) \geq 0.5$. Without loss of generality, we can assume that

$$\mathbb{P}_{(x,y)\sim D_S}(G(\text{Relu}(W_1(x+\epsilon v_y)+b_1),b) \geq \eta n_1/2) \geq 0.5.$$

For $l \in [m]$, let $\overline{W}_2^{(l)} = W_2^{(l)} - W_2^{(l+1)}$, where $W_2^{(m+1)} = W_2^{(1)}$. Now assume $U_v \in \mathbb{R}^{n_1 \times n}$, whose rows are all μv_l, and $U_l = -\text{diag}(\text{sign}(\overline{W}_2^{(l)}))$. Let $\widetilde{W}_1 = W_1 + \frac{1}{m}\sum_{l=1}^m U_l U_v$, and

$$\mathcal{G}(x) = W_2\text{Relu}(\widetilde{W}_1 x + b_1) + b_2.$$

We will show that $\mathcal{G}(x)$ satisfies the conditions of the theorem.

It is easy to see that $||\Theta_{\mathcal{G}} - \Theta||_\infty \leq \mu$, where Θ is the parameter of $\mathcal{F}$ and $\Theta_{\mathcal{G}}$ is the parameter of $\mathcal{G}$. For any $x \in S$, by C_4, we have $v_l x = 0$ for all $l \in [m]$, so $U_v x = 0$. Then $\widetilde{W}_1 x = W_1 x + \frac{1}{m}\sum_{l=1}^m (U_l U_v)x = W_1 x$, which means $\mathcal{G}(x) = \mathcal{F}(x)$, and thus the accuracy of $\mathcal{G}$ over $\mathcal{D}_S$ is equal to that of $\mathcal{F}$.

Moreover, we have that

$$\begin{aligned}
& \widetilde{W}_1(x+\epsilon v_y) \\
= \;& \widetilde{W}_1 x + \epsilon \widetilde{W}_1 v_y \\
= \;& W_1 x + \epsilon W_1 v_y + \tfrac{\epsilon}{m}\textstyle\sum_{l=1}^m U_l U_v v_y \\
= \;& W_1 x + \epsilon W_1 v_y + \tfrac{\epsilon}{m} U_y U_v^{(y)} v_y \\
= \;& W_1(x+\epsilon v_y) - \tfrac{\epsilon\mu}{m}\text{sign}(\overline{W}_2^{(y)}).
\end{aligned}$$

Let $x \in S$ satisfy $G(\text{Relu}(W_1(x+\epsilon v_y)+b_1),b) > \eta n_1/2$. By conditions C_1 and C_2, we have that

$$\begin{aligned}
& \mathcal{G}_y(x+\epsilon v_y) - \mathcal{G}_{y+1}(x+\epsilon v_y) \\
= \;& (W_2^{(y)} - W_2^{(y+1)})\text{Relu}(\widetilde{W}_1(x+\epsilon v_y)+b_1) + b_2^{(y)} - b_2^{y+1} \\
= \;& \overline{W}_2^{(y)}\text{Relu}(W_1(x+\epsilon v_y) - \tfrac{\epsilon\mu}{m}\text{sign}(\overline{W}_2^{(y)}) + b_1) + b_2^{(y)} - b_2^{y+1} \\
\leq \;& \overline{W}_2^{(y)}\text{Relu}(W_1(x+\epsilon v_y)+b_1) - \min\{\epsilon\mu/m, b\}cn_1\eta/2 + b_2^{(y)} - b_2^{y+1} \text{ (by LemmaA.2)} \\
= \;& \mathcal{F}_y(x+\epsilon v_y) - \mathcal{F}_{y+1}(x+\epsilon v_y) - \min\{\epsilon\mu/m, b\}cn_1\eta/2 \\
\leq \;& A - \min\{\epsilon\mu/m, b\}cn_1\eta/2 < 0.
\end{aligned}$$

Thus $\mathcal{G}$ does not give label y to $x+\epsilon v_y$ and $\mathcal{G}$ has an adversarial example of x in $S_\infty(x,\epsilon)$. Furthermore, since $\mathbb{P}_{(x,y)\sim D_s}(G(\text{Relu}(W_1(x+\epsilon v_y)+b_1),b) \geq \mu n_1/2) \geq 0.5$, we have that

$$\mathbb{P}_{(x,y)\sim D_S}(\mathcal{G} \text{ has an adversarial example of } x \text{ in } \mathbb{B}_\infty(x,\epsilon)) \geq 0.5$$

which is equivalent to the theorem. □

A.3. Proof of Theorem 4.5

We first prove a lemma.

Lemma A.3. *For $U \in \mathbb{R}^n$, $V \in \mathbb{R}^{1\times n}$, and $k \in \mathbb{R}_+$, we have*

(1): If $V^{(1)} \geq 0$, then $V\text{Relu}(U) \leq V\text{Relu}(U + k\mathbf{1}^{(1)})$.

(2): If $V^{(1)} < 0$, then $V\text{Relu}(U) \geq V\text{Relu}(U + k\mathbf{1}^{(1)}) + (k - |||U||_\infty)|V^{(1)}|$.

Proof. It is easy to see

$$V\text{Relu}(U) - V\text{Relu}(U + k\mathbf{1}^{(1)}) = V^{(1)}(\text{Relu}(U))^{(1)} - (\text{Relu}(U + k\mathbf{1}^{(1)}))^{(1)}.$$

Since Relu is a monotonically increasing function, we obtain (1).

For $a, b \in \mathbb{R}$ and $b > 0$, $\text{Relu}(a) - \text{Relu}(a + b) \leq -(b - |a|)$ is valid, so $(\text{Relu}(U))^{(1)} - (\text{Relu}(U + k\mathbf{1}^{(1)}))^{(1)} \leq -(k - |U^{(1)}|) \leq -(k - ||U||_\infty)$. If $V^{(1)} < 0$, then $(V^{(1)})((\text{Relu}(U))^{(1)} - (\text{Relu}(U + k\mathbf{1}^{(1)}))^{(1)}) \geq |V^{(1)}|(k - ||U||_\infty)$, and (2) is proved. □

Now we prove Theorem 4.5.

Proof. We first prove the following claim:

(C). For any $\nu \geq mn$ and $\epsilon > 0$, there exists a $\Theta_a \in \mathbb{R}^k$ such that $||\Theta_a - \Theta||_0 \leq \nu$, $\text{AC}(\mathcal{F}_{\Theta_a}) = \text{AC}(\mathcal{F}_\Theta)$, and $\text{AA}(\mathcal{F}_{\Theta_a}, \epsilon) = 0$.

By condition C_1, for $i \in [m]$, there exists a k_i such that $W_2^{(i,k_i)} < 0$. Let $q = \min_{i\in[m]}\{|(W_2^{(i,k_i)}|\}$. By condition C_2, for $l \in [m]$, there exists a $v_l \in \mathbb{R}^n$ such that $v_l \perp S$, $v_l \perp v_k$ for $l \neq k$, and $||v_l||_2 = 1$, which means $||v_l||_\infty \leq 1$.

Let $T = \max_{l\in[m],||x||_2=1}\{|(W_2^{(l)})||Ux|\}$. For $x \in S$, let $\text{CW}(\mathcal{F}, x) = \max_{i\neq y}\{\mathcal{F}_y(x) - \mathcal{F}_i(x)\}$ where y is the label of x.

Since $S \subset [0, 1]^n$, $\mathcal{F}$ is piece-wise linear and bounded, there exists a $C > 0$ such that $\text{CW}(\mathcal{F}, x) < C$ and $||W_1x+b||_\infty < C$. Let $K = \frac{C/\epsilon+2T}{q} + C/\epsilon$. $\widetilde{W}_1 \in \mathbb{R}^{n_1\times n}$ is defined as $\widetilde{W}_1^{(j)}$=$W_1^{(j)} + K(\sum_{i\in[m]} I(j = k_i)v_i)$ for all $j \in [n_1]$, that is, add Kv_i to the k_i-th row of W_1. Let

$$\mathcal{G}(x) = W_2\text{Relu}(\widetilde{W}_1x + b_1) + b_2.$$

We will show that $\mathcal{G}(x)$ satisfies the condition of the theorem.

For any $x \in S$, by condition C_2, we have $v_lx = 0$ for all $l \in [m]$, so $\widetilde{W}_1x = W_1x$, which means $\mathcal{G}(x) = \mathcal{F}(x)$. Thus the accuracy of $\mathcal{G}$ over D_s is equal to that of $\mathcal{F}$.

For any $x \in S$, we know that x satisfies $\text{CW}(\mathcal{F}, x) < C$ and $||W_1x + b||_\infty < C$, then

$$\begin{aligned}
& \mathcal{G}_y(x + \epsilon v_y) \\
= \;& W_2^{(y)}\text{Relu}(W_1(x + \epsilon v_y) + \mathbf{1}_{k_y}K\epsilon + b) + b_2^{(y)} \\
= \;& W_2^{(y)}\text{Relu}(W_1x + b + \mathbf{1}_{k_y}K\epsilon + \epsilon W_1v_y) + b_2^{(y)} \\
\leq \;& W_2^{(y)}\text{Relu}(W_1x + b + \mathbf{1}_{k_y}K\epsilon) + \epsilon|W_2^{(y)}||W_1v_y| + b_2^{(y)} \\
\leq \;& W_2^{(y)}\text{Relu}(W_1x + b + \mathbf{1}_{k_y}K\epsilon) + \epsilon T + b_2^{(y)} \\
\leq \;& W_2^{(y)}\text{Relu}(W_1x + b) - |W_2^{(y,k_y)}|(K\epsilon - C) \\
& +\epsilon T + b_2^{(y)} \text{ (By Lemma A.3)} \\
\leq \;& W_2^{(y)}\text{Relu}(W_1x + b) - q(K\epsilon - C) + \epsilon T + b_2^{(y)} \\
= \;& W_2^{(y)}\text{Relu}(W_1x + b) - C - \epsilon T + b_2^{(y)} \\
= \;& \mathcal{F}_y(x) - C - \epsilon T.
\end{aligned}$$

By Condition C_1, there exists an $l_2 \neq y$ such that $W_2^{(l_2,k_y)} \geq 0$, so

$$\begin{aligned} & \mathcal{G}_{l_2}(x+\epsilon v_y) \\ = \; & W_2^{(l_2)}\text{Relu}(W_1 x + b + \mathbf{1}_{k_y} K\epsilon + \epsilon W_1 v_y) + b_2^{(l_2)} \\ \geq \; & W_2^{(l_2)}\text{Relu}(W_1 x + b + \mathbf{1}_{k_y} K\epsilon) - \epsilon |W_2^{(l_2)}||W_1 v_y| + b_2^{(l_2)} \\ \geq \; & W_2^{(l_2)}\text{Relu}(W_1 x + b) - \epsilon T + b_2^{(l_2)} \text{ (By LemmaA.3)} \\ = \; & \mathcal{F}_{l_2}(x) - \epsilon T. \end{aligned}$$

Since $\mathcal{F}_y(x) - C < \mathcal{F}_{l_2}(x)$, we have

$$\mathcal{G}_{l_2}(x+\epsilon v_y) \geq \mathcal{F}_{l_2}(x) - \epsilon T > \mathcal{F}_y(x) - C - \epsilon T \geq \mathcal{G}_y(x+\epsilon v_y),$$

so $\mathcal{G}$ does not give the correct label to $x + \epsilon v_y$, which means that $\mathcal{G}$ has an adversarial example of x in $S_\infty(x,\epsilon)$. The claim **(C)** is proved.

Let $\nu = (n+m+1)m \geq nm$. By claim **(C)** just proved and the fact $|\Theta| = (n+m+1)n_1 + m$, if $n_1 \geq m/\eta$, then we have that $||\Theta_a - \Theta||_0 \leq \nu = (n+m+1)m = \frac{(n+m+1)m}{(n+m+1)n_1+m}|\Theta| \leq \frac{n+m+1}{n+m+1+\eta}\eta|\Theta| < \eta|\Theta|$. The theorem is proved. □

B. More training details

In this section, we will show that, Algorithms 1 and 2 are convergent and stable. We use Algorithms 1 and 2 to compute adversarial parameters for VGG-16 on CIRAR-10. PGD-(10,8/255) is used to solve problem (4). The GPU we used in the experiment is NVIDIA GeForce RTX 3090.

When the algorithm goes on, we will observe the empirical loss

$$\text{Loss}_{\text{CE}}(T) = \frac{1}{|T|}\sum_{(x,y)\in T} L_{\text{CE}}(\mathcal{F}_{\Theta_a}(x), y),$$

and the adversarial loss

$$\text{Loss}_{\text{AT}}(T) = \frac{1}{|T|}\sum_{(x,y)\in T} \max_{x'\in S_\infty(x,8/255)} L_{\text{CE}}(\mathcal{F}_{\Theta_a}(x'), y),$$

the accuracy AC, and the adversarial accuracy AA on the T, where T is training set and $|T|$ is the number of examples in the T. The change of these values with training epochs was shown in Figure 6.

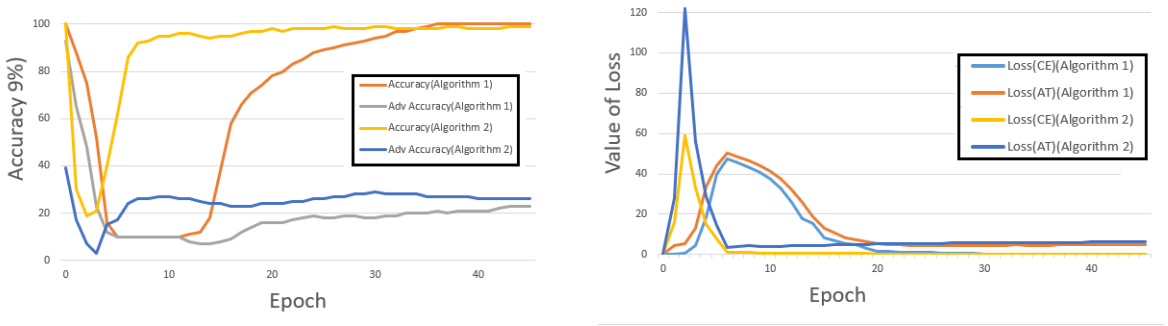


Figure 6. Left: Change trend of accuracy and adversarial accuracy with training going on. Right: Change trend of value of loss function with training going on.

At first, the accuracy and the adversarial accuracy decline rapidly, and correspondingly, the values of loss functions $L_{\text{CE}}(T)$ and $L_{\text{AT}}(T)$ rise rapidly. This is because in Phase one, we are trying to break the accuracy by using loss function $-\frac{1}{|T|}\sum_{(x,y)\in T} L_{\text{AT}}(\mathcal{F}_\Theta(x), y)$, as shown in Section 3.2.

Then, after Phase two starts, $L_{\mathrm{CE}}(T)$ and $L_{\mathrm{AT}}(T)$ fall, but $L_{\mathrm{CE}}(T)$ falls more rapidly and reaches a small value 0.1. But $L_{\mathrm{AT}}(T)$ falls slowly and finally stabilizes at a relatively large value. So, $\frac{L_{\mathrm{CE}}(T)}{L_{\mathrm{AT}}(T)}$ will become small, as shown in Figure 7. Correspondingly, the accuracy rises more rapidly and reaches a high level, and the adversarial accuracy rises more slowly and stabilizes at a low level. We thus obtain adversarial parameters: keep the accuracy and reduce the robustness.

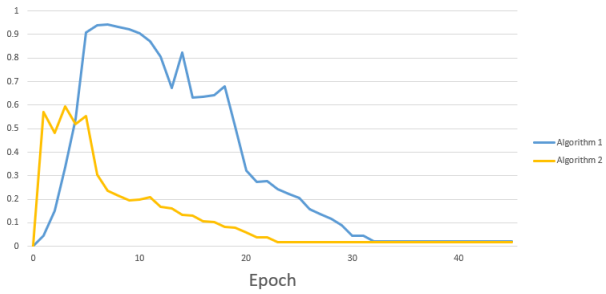


Figure 7. Relation between training epochs and $L_{\mathrm{CE}}(T)/L_{\mathrm{AT}}(T)$.

Moreover, adding a regulation term to the loss function in Phase two will make the training process more stable and quick, that is, using the loss function $\frac{\sum_{(x,y)\in T} L_{\mathrm{CE}}(\mathcal{F}_\Theta(x),y)}{\sum_{(x,y)\in T} L_{\mathrm{AT}}(\mathcal{F}_\Theta(x),y)} + \lambda \sum_{(x,y)\in T} R(\mathcal{F}_\Theta(x),y)/|T|$. For more quick and stable training, we can take $\lambda = 0.1$ and $R(\mathcal{F}_\Theta(x),y) = L_{\mathrm{CE}}(\mathcal{F}_\Theta(x),y)$ for the first two epochs in the Phase two. For subsequent training, we change $R(\mathcal{F}_\Theta(x),y)$ to be $\frac{1}{L_{\mathrm{AT}}(\mathcal{F}_\Theta(x),y)}$ and keep $\lambda = 0.1$. Adding regular terms like this will make the training in the Phase two about six times faster.

C. Phase One of Algorithms 1 and 2 is necessary

We will show that, when Phase one is not used, Phase two may have gradient vanishing problems. We first prove a lemma.

Lemma C.1. *We have*

$$||\nabla_\Theta \frac{L_{\mathrm{CE}}(\mathcal{F}_\Theta(x),y)}{L_{\mathrm{AT}}(\mathcal{F}_\Theta(x),y)}||_\infty = O(\frac{L_{\mathrm{CE}}(\mathcal{F}_\Theta(x),y)}{L_{\mathrm{AT}}(\mathcal{F}_\Theta(x),y)}). \tag{10}$$

Proof. We have

$$\nabla_\Theta \frac{L_{\mathrm{CE}}(\mathcal{F}_\Theta(x),y)}{L_{\mathrm{AT}}(\mathcal{F}_\Theta(x),y)} = \frac{\nabla_\Theta L_{\mathrm{CE}}(\mathcal{F}_\Theta(x),y) L_{\mathrm{AT}}(\mathcal{F}_\Theta(x),y) - \nabla_\Theta L_{\mathrm{AT}}(\mathcal{F}_\Theta(x),y) L_{\mathrm{CE}}(\mathcal{F}_\Theta(x),y)}{L_{\mathrm{AT}}(\mathcal{F}_\Theta(x),y)^2}. \tag{11}$$

Use $\mathcal{F}_{\Theta,i}(x)$ to denote the i-th weight of$\mathcal{F}_\Theta(x)$. Then

$$L_{\mathrm{CE}}(\mathcal{F}_\Theta(x),y) = -\ln \frac{e^{\mathcal{F}_{\Theta,y}(x)}}{\sum_{i\in[m]} e^{\mathcal{F}_{\Theta,i}(x)}},$$

so we have

$$\nabla_\Theta L_{\mathrm{CE}}(\mathcal{F}_\Theta(x),y) = \frac{\sum_{i\neq y} (\nabla_\Theta \mathcal{F}_{\Theta,i}(x)) e^{\mathcal{F}_{\Theta,i}(x)}}{\sum_{i\in[m]} e^{\mathcal{F}_{\Theta,i}(x)}}. \tag{12}$$

Note that $||\nabla_\Theta \mathcal{F}^i_\Theta(x)||_\infty < C$ for any $i \in [m], x \in [0,1]^n$ and a $C \in R$, where C depends on Θ. For two vectors $a, b \in \mathbb{R}^k$, use $a < b$ to denote that $a^{(i)} < b^{(i)}$ for all $i \in [k]$. Then equation (12) becomes

$$|\nabla_\Theta L_{\mathrm{CE}}(\mathcal{F}_\Theta(x),y)| < C \frac{\sum_{i\neq y} e^{\mathcal{F}_{\Theta,i}(x)}}{\sum_{i\in[m]} e^{\mathcal{F}_{\Theta,i}(x)}} \mathbf{1} = C(1 - e^{-L_{\mathrm{CE}}(\mathcal{F}_\Theta(x),y)})\mathbf{1}. \tag{13}$$

Similarly, we also have that

$$|\nabla_\Theta L_{\mathrm{AT}}(\mathcal{F}_\Theta(x),y)| < CI(1 - e^{-L_{\mathrm{AT}}(\mathcal{F}_\Theta(x),y)}) \tag{14}$$

Substituting the inequalities (13) and (14) into equation (11), we have

$$|\nabla_\Theta \frac{L_{\rm CE}(\mathcal{F}_\Theta(x),y)}{L_{\rm AT}(\mathcal{F}_\Theta(x),y)}| < C(\frac{1-e^{-L_{\rm CE}(\mathcal{F}_\Theta(x),y)}}{L_{\rm AT}(\mathcal{F}_\Theta(x),y)} + \frac{L_{\rm CE}(\mathcal{F}_\Theta(x),y)(1-e^{-L_{\rm AT}(\mathcal{F}_\Theta(x),y)})}{(L_{\rm AT}(\mathcal{F}_\Theta(x),y))^2})\mathbf{1}.$$

Note that, when $x \in (0,\infty)$, $x \geq 1-e^{-x}$. Then

$$|\nabla_\Theta \frac{L_{\rm CE}(\mathcal{F}_\Theta(x),y)}{L_{\rm AT}(\mathcal{F}_\Theta(x),y)}| < 2C(\frac{L_{\rm CE}(\mathcal{F}_\Theta(x),y)}{L_{\rm AT}(\mathcal{F}_\Theta(x),y)})\mathbf{1}$$

from which (10) is proved. □

From Lemma C.1, for a well trained $\mathcal{F}_\Theta$, if $L_{\rm CE}(\mathcal{F}_\Theta(x),y)$ is very small comparing with $L_{\rm AT}(\mathcal{F}_\Theta(x),y)$, then the value of $\nabla_\Theta \frac{L_{\rm CE}(\mathcal{F}_\Theta(x),y)}{L_{\rm AT}(\mathcal{F}_\Theta(x),y)}$ also becomes very small and thus we have gradient vanishing. Phase one of Algorithm 1 is to make $L_{\rm CE}(\mathcal{F}_\Theta(x),y)/L_{\rm AT}(\mathcal{F}_\Theta(x),y)$ become large to avoid this problem.

In what below, we will use experiments for CIFAR-10 to verify such phenomenon. The original network is VGG-16, which was trained by adversarial training with PGD(10-8/255), uisng 200 epochs. PGD-(10,8/255) is used to compute the adversarial accuracy of the test set.

Firstly, we show the value of loss function $L_{\rm CE}$ and $L_{\rm AT}$ on the training set, as show in Table 4:

Table 4. Average value of Loss function on the training set.

$L_{\rm CE}$	$L_{\rm AT}$
$7.6 \cdot 10^{-9}$	$6.8 \cdot 10^{-5}$

The value of $L_{\rm CE}$ is much smaller than that of $L_{\rm AT}$.

Now, we find adversarial parameters by using algorithm 1 and 2 but without Phase one, and the result after adversarial parameters attack is in Table 5:

Table 5. Algorithm without Phase one. AC: accuracy, AA: adversarial accuracy,

Attack	AC	AA
No attack	80%	39%
$L_{\infty,0.06}$	79%	38%
$L_{\infty,0.10}$	79%	38%
$L_{0,0.5\%}$	80%	39%
$L_{0,1\%}$	78%	38%

It can be found that the accuracy and the adversarial accuracy hardly change. This is because the gradient is almost zero in the training, and as a consequence, the value of parameters almost unchanged. We can also get this conclusion through the change of the value of the loss function and the accuracy in the training process without Phase one, as shown in the Figure 8.

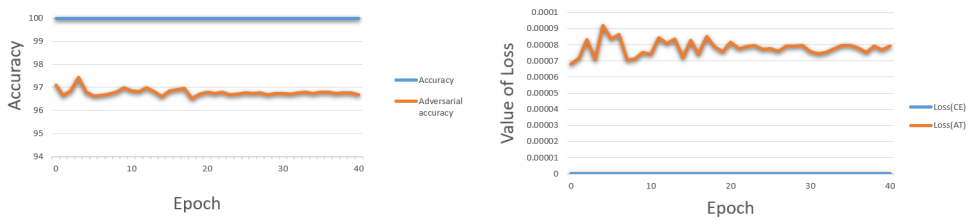


Figure 8. The change of the value of loss function and accuracy in the training process, without Phase one.

It can be seen that these values will hardly change during training, which is caused by the vanishing of gradient.

If using Phase one, the value $L_{\mathrm{CE}}/L_{\mathrm{AT}}$ will increase in Phase one to prevent vanishing gradient in Phase two. To compare, the values of loss functions L_{CE} and L_{AT} on the training set after Phase one is shown in the Table 6, and the accuracy and adversarial accuracy are shown in the Table 7.

Table 6. Average value of Loss function on the training set after Phase one.

L_{CE}	L_{AT}
47.56	61.11

Table 7. Algorithm with Phase one. AC: accuracy, AA: adversarial accuracy,

Attack	AC	AA
No attack	80%	39%
$L_{\infty,0.06}$	76%	15%
$L_{\infty,0.10}$	77%	6%
$L_{0,0.5\%}$	76%	19%
$L_{0,1\%}$	77%	16%

The results are much better than that without Phase one, so Phase one is necessary.

If the original network satisfies $L_{\mathrm{CE}} \approx L_{\mathrm{AT}}$, then the algorithm without Phase one can also work, but using Phase one gives a better result. To show that, we use network Resnet18, and other setting is the same as before. The values of loss functions L_{CE} and L_{AT} on the training set are shown in Table 8.

Table 8. Average value of Loss function on the training set.

L_{CE}	L_{AT}
0.0032	0.0071

The value of L_{CE} is not much smaller than L_{AT}.

Now, we find adversarial parameters using Algorithms 1 and 2, and using Algorithms 1 and 2 without Phase one. The result after adversarial parameters attack is given in Table 9.

Table 9. Accuracy and Adversarial Accuracy for parameters attack, AC: accuracy, AA: adversarial accuracy.

Attack	With Phase one		Without Phase one	
	AC	AA	AC	AA
No Attack	84%	52%	84%	52%
$L_{\infty,0.06}$	85%	24%	87%	32%
$L_{\infty,0.1}$	85%	6%	89%	19%
$L_{0,0.5\%}$	83%	0%	87%	1%
$L_{0,1\%}$	84%	2%	88%	1%

It is easy to see that the algorithm with Phase one has better results.