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 $L_\infty$ 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 adversary 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
has lost robustness due to some attacks but still maintains the recognition ability for the clean ones.

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

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 $F_{\Theta}$ is sufficiently large, then $F_{\Theta}$ has $L_\infty$ norm adversarial parameters $\Theta_a$ sufficiently near $\Theta$ such that $F_{\Theta_a}$ has adversarial examples sufficiently near a given benign sample (refer to Theorem 4.5), 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 $F_{\Theta_a}$ has the same accuracy as $F_{\Theta}$, 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 $F_{\Theta}$ 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 $F_{\Theta_a}$ 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 $F_{\Theta_a}$ 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.
Adversarial Parameter Attack on Deep Neural Networks

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 \(D_S \) over \(\mathbb{D} = S \times [m] \), where \(S \subseteq \{0,1\}^n\) and \([m] = \{1, \ldots , m\}\) is the label set. Finite datasets used for training and testing are chosen iid according to \(D_S\).

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 \(\Theta\) 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 \(\tilde{\mathcal{F}}(x) = \arg\max_{\mathcal{F}_l(x)}\) to be the classification result of \(\mathcal{F}(x)\). The accuracy of \(\mathcal{F}\) is

\[
AC(\mathcal{F}) = P_{(x,y) \sim D_S}(\tilde{\mathcal{F}}(x) = y).
\]

Given an adversarial budget \(\epsilon\), the following adversarial accuracy is used to measure the robustness of network \(\mathcal{F}\)

\[
AA(\mathcal{F}, \epsilon) = P_{(x,y) \sim D_S}(\tilde{\mathcal{F}}(x') = y, \forall x' \text{ s.t. } ||x' - x||_\infty \leq \epsilon).
\]

Notice that we consider adversarial examples with \(L_\infty\) 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 \(\Theta\), if (1) the \(L_p\) distance between \(\Theta_a\) and \(\Theta\) is smaller than certain given bound to be given subsequently; (2) \(AC(\mathcal{F}_{\Theta_a}) \geq \gamma_{ac} AC(\mathcal{F}_{\Theta})\) for a given \(\gamma_{ac} \in (0, 1)\) which is close to 1, say \(\gamma_{ac} = 90\%\); and (3) \(AA(\mathcal{F}_{\Theta_a}, \epsilon)\) is much smaller than \(AA(\mathcal{F}_{\Theta}, \epsilon)\), that is, \(\mathcal{F}_{\Theta_a}\) has much more adversarial examples than \(\mathcal{F}_{\Theta}\).

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 \(L_\infty\) norm. The attacked networks (b) and (c) are accurate but not robust.

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 attack, 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 \(L_\infty\) norm attack. (2) Change fewer parameters to reduce the number of bits need to be changed, corresponding to \(L_0\) attack.

\(L_\infty\) Norm Adversarial Parameters. For an attack budget ratio \(\gamma < 1\), the adversarial parameters of \(\Theta\) are taken in

\[
B_{\infty}(\Theta, \gamma) = \{\Theta_a \in \mathbb{R}^k : |\Theta_a - \Theta|^{(i)} \leq \gamma |\Theta|^{(i)}, \forall i \in [k]\}\tag{1}
\]

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 \(L_\infty\) norm.

\(L_0\) Norm Adversarial Parameters. For an attack budget ratio \(\eta < 1\), the adversarial parameters of \(\Theta\) are taken in

\[
B_{0}(\Theta, \eta) = \{\Theta_a \in \mathbb{R}^k : ||\Theta_a - \Theta||_0 \leq \eta k\}\tag{2}
\]

An illustrative example for adversarial parameters is given in Figure 1. We consider a binary classification dataset \(T = \{(0,0), (1,0), (0,1), (0,1), (1,0), (1,1)\}\) and use the network \(\mathcal{F} : \mathbb{R}^2, 2 \to \mathbb{R}^2\) which has one hidden layer and width 24 to classify \(T\). We compute the \(L_\infty\) 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: \(AC(\mathcal{F}) = 1\) and \(AA(\mathcal{F}, 0.2) = 1\), but the attacked networks \(\mathcal{F}'\) are not robust but still accurate on the data: \(AC(\mathcal{F}') = 1\) and \(AA(\mathcal{F}', 0.2) = 0\).
3.2. Algorithms for adversarial parameters

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

$$\arg\max_{\theta \in B_p(\Theta, \epsilon)} \frac{I(A_\epsilon(F_{\theta}), y) \geq \gamma_{\mathrm{AC}} A_{\epsilon}(F_{\theta})}{A_{\epsilon}(F_{\theta}, y)}$$

where $I(t) = 1$ if $t$ is true and $I(t) = 0$ otherwise, and $p = \infty$ or $0$. Once $A_{\epsilon}(F_{\theta}) < \gamma_{\mathrm{AC}}$, the numerator becomes 0 and the whole value is 0; so the solution $\Theta_a$ of optimization problem (3) must make $A(A_{\epsilon}(F_{\theta}), \epsilon)$ the smallest while ensuring $A_{\epsilon}(F_{\theta}) \geq \gamma_{\mathrm{AC}}$.

To solve optimization problem (3), we need a training set $D$ selected i.i.d. according to $D$, which will be used to compute adversarial parameters. The phase one consists of two phases: 1) Preliminary training with the loss function $\mathcal{L}_{\text{CE}}(F(x), y)$ for all $(x, y) \sim D$; 2) Main training with the loss function $\mathcal{L}_{\text{CE}}(F_{\theta}(x), y)$ for all $(x, y) \sim D$.

**Algorithm 1** Adversarial Parameter Attack under $L_\infty$ norm

**Input:**
- The parameter $\Theta \in \mathbb{R}^k$ of $F$;
- The parameter attack ratio $\gamma$;
- A training set $T$;
- Hyper-parameters: $\alpha \in \mathbb{R}_+, n_1, n_2 \in \mathbb{N}$.

**Output:**
- Adversarial parameter $\Theta_a$ in $B_\infty(\Theta, \gamma)$.

Let $i = 0, \Theta_a = \Theta$.
For all $i \in [n_1 + n_2]$:
- If $i < n_1$:
  - $L = -\frac{1}{|T|} \sum_{(x, y) \in T} \mathcal{L}_{\text{CE}}(F_{\theta_a}(x), y)$.
- Else:
  - $L = \frac{\sum_{(x, y) \in T} \mathcal{L}_{\text{CE}}(F_{\theta_a}(x), y)}{|T|} - \frac{1}{|T|} \sum_{(x, y) \in T} \mathcal{L}_{\text{AT}}(F_{\theta_a}(x), y).
- $\bar{\Theta} = \Theta_a - \alpha \nabla L$.
- $\Theta_a = \text{Proj}(\bar{\Theta}, B_\infty(\Theta, \gamma))$.
  - (Project $\bar{\Theta}$ into $B_\infty(\Theta, \gamma)$)
**Output:** $\Theta_a$.

**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 $F$;
- The parameter attack ratio $\gamma$;
- A training set $T$;
- Hyper-parameters: $\alpha \in \mathbb{R}_+, n_1, n_2 \in \mathbb{N}$.

**Ensure:**
- Adversarial parameter $\Theta_a$ in $B_0(\Theta, \eta)$.
Let $i = 0, \Theta_a = \Theta, P_1 = \emptyset, P_2 = \emptyset$;
Calculate $V_1 = |\nabla \Theta_a \sum_{(x, y) \in T} \mathcal{L}_{\text{CE}}(F_{\Theta_a}(x), y)| \in \mathbb{R}_k$;
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 $i \in [n_1]$:
- $L = -\frac{1}{|T|} \sum_{(x, y) \in T} \mathcal{L}_{\text{CE}}(F_{\Theta_a}(x), y)$.
- For each $j \in P_1$:
  - $\Theta_a(j) = \Theta_a(j) - \alpha(\nabla L)^{(j)}$.
Calculate $V_2 = |\nabla \Theta_a \sum_{(x, y) \in T} \mathcal{L}_{\text{AT}}(F_{\Theta_a}(x), y)| \in \mathbb{R}_k$;
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 $i \in [n_1 + n_2 + 1, \ldots, n_1 + n_2]$: $L = \sum_{(x, y) \in T} \mathcal{L}_{\text{AT}}(F_{\Theta_a}(x), y)$.
- For each $j \in P_2$:
  - $\Theta_a(j) = \Theta_a(j) - \alpha(\nabla L)^{(j)}$.
**Output:** $\Theta_a$.

**Compute $L_0$ norm adversarial parameters.** The algorithm is similar to Algorithm I. What we need to do additionally is to select positions of the parameters to be changed.
Adversarial Parameter Attack on Deep Neural Networks

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 $D_S$ a distribution defined on $S \times [m]$. In this section, we consider the following network $F_\Theta : S \to \mathbb{R}^m$ with one hidden layer:

$$F_\Theta(x) = W_2 \text{Relu}(W_1 x + b_1) + b_2,$$

where $W_1 \in \mathbb{R}^{n_1 \times n}, b_1 \in \mathbb{R}^{n_1}, W_2 \in \mathbb{R}^{m \times n_1}, b_2 \in \mathbb{R}^m$. We introduce several notations. Let $C_x$ of $F$ for a given data. A small perturbation $x$ examples of $\Theta$ for $\mathbb{R}^n$ is called $\epsilon$-adversarial parameter of $\Theta$, and $k = ||\Theta|| = (n + m + 1)n_1 + m$.

4.1. $L_\infty$ norm adversarial parameter

We introduce several notations. Let $||x|| = \min_{i \in [n]} ||x_i||$ 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

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

We first consider a simple case: adversarial parameters for a given data. A small perturbation $\Theta_\alpha$ of $\Theta$ is called adversarial parameter for a given data $(x, y) \sim D_S$, if $\tilde{F}_{\Theta_\alpha}(x) = y$ and the robustness radius

$$R(x) = \max\{\zeta \in \mathbb{R}_+: \tilde{F}_{\Theta_\alpha}(x') = y, \forall x' \in S_\infty(x, \zeta)\}$$

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

Theorem 4.1. Let $F_\Theta : S \to \mathbb{R}^m$ 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:

C1. $|F_i(x) - F_j(x)| < A$ 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.

C2. $||W_2^{(i)} - W_2^{(j)}||_\infty > c$ for all $i, j \in [m], i \neq j$ and $c \in \mathbb{R}_+$.

C3. At least $\eta n_1$ coordinates of $|\text{Relu}(W_1 x + b_1)|$ are bigger than $b$, where $\eta \in (0, 1)$. For $\eta, \epsilon \in \mathbb{R}_+$ satisfying $\epsilon < a$, if $n_1 > \frac{2A}{\min(\epsilon(m-1), b)v\eta}$, then there exists $\epsilon_1 \in \mathbb{R}_+$ such that $||\Theta_\alpha - \Theta||_\infty \leq \mu$, $\tilde{F}_{\Theta_\alpha}(x_0) = y_0$, and $F_{\Theta_\alpha}$ 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 $F_\Theta$ is sufficiently large, then $\Theta$ has adversarial parameters which are as close as possible to $\Theta$, and $F_{\Theta_\alpha}$ 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 $F_\Theta : S \to \mathbb{R}^m$ be a trained DNN with structure in (9) and suppose that $F$ and $S$ satisfy the following conditions:

C1. $|F_i(x) - F_j(x)| < A$ for all $i, j \in [m]$ and $x \in \bigcup_{x_0 \in S} S_\infty(x_0, a)$, where $A, a \in \mathbb{R}_+$.

C2. $||W_2^{(i)} - W_2^{(j)}||_\infty > c$ for all $i, j \in [m], i \neq j$, where $c \in \mathbb{R}_+$.

C3. For all $x \in S$, at least $\eta n_1$ coordinates of $|\text{Relu}(W_1 x + b_1)|$ are bigger than $b$, where $\eta \in (0, 1)$ and $b \in \mathbb{R}_+$.

C4. The dimension of $S$ is lower than $n - m$.

For $\forall \mu, \epsilon \in \mathbb{R}_+$ satisfying $\epsilon < a$, if $n_1 > \frac{2A}{\min(\epsilon(m-1), b)v\eta}$, then there exists $\Theta_\alpha \in \mathbb{R}^k$ such that $||\Theta_\alpha - \Theta||_\infty \leq \mu$, $AC(F_{\Theta_\alpha}) = AC(F_\Theta)$, and $\text{AA}(F_{\Theta_\alpha}, \epsilon) \leq 0.5$.

By Theorem 4.3, when the width $n_1$ of $F$ is sufficiently large, there exist adversarial parameters as close as possible to $\Theta$, 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 is about $10^5 (10^5)$, 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 $F_\Theta : S \to \mathbb{R}^m$ be a trained DNN with structure in (9) and suppose that $F$ and $S$ satisfy the following conditions:

C1. Each row of $W_2$ has at least one negative component; each column of $W_2$ has at least one non-negative component.

C2. The dimension of $S$ is lower than $n - m$.

For any $\eta, \epsilon \in (0, 1)$, if $n_1 \geq m/\eta$, then there exists $\Theta_\alpha \in \mathbb{R}^k(\Theta, \eta)$ such that $AC(F_{\Theta_\alpha}) = AC(F_\Theta)$ and $\text{AA}(F_{\Theta_\alpha}, \epsilon) = 0$. 
5. Experimental results

In this section, we verify the effectiveness of Algorithms 1 and 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 \( F_\Theta \) 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, CIFAR-1000, WRN-32-4, and WRN-32-10 [Krizhevsky et al., 2009], and Tiny-ImageNet [Le & Yang, 2015].

The adversarial accuracies of the attacked network \( F_{\Theta_a} \) 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 \(\{x, F_\Theta(x)\}\) to train three networks which have the same structure with \(F_\Theta\), compute adversarial examples with PGD-(10,8/255) for these models, and use them to attack \(F_\Theta\).

(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 \(N(0, (8/255)^2)\) 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 A.

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 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 \(\geq 50\), 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 \( L_\infty \) norm adversarial parameter attack with ratio \( \gamma \) as \( L_\infty, \gamma \) and the \( L_0 \) norm adversarial parameter attack with ratio \( \eta \) as \( L_0, \eta \). Refer [1] and [2] for details. The parameter \( \gamma_{ac} \) in optimization problem (3) is set to 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 \( \gamma_{ac} = 90\% \) of that of the original networks, so condition (2) in the definition of adversarial parameters is always satisfied. Algorithm 1 generates high quality \( L_\infty \) 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 \( \eta \geq 75\% \). For ResNet-18, WRN-32-4, and WRN-32-10, the \( L_0 \) attacks achieve near zero adversarial accuracies, as stated in Theorem 4.3.

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 \( \eta = 75\% \).
we find that when the depth increases, the networks become harder 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, it 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 $L_\infty$ 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%.

#### Table 1. Adversarial parameter attack for networks VGG-16 and ResNet-18 on CIFAR-10.

| 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%    |
| $L_{\infty,0.25\%}$ | 75%    | 24%       | 82%    | 5%    |
| $L_{\infty,0.5\%}$ | 76%    | 19%       | 83%    | 0%    |
| $L_{\infty,0.75\%}$ | 75%    | 17%       | 83%    | 2%    |
| $L_{\infty,1.0\%}$ | 77%    | 16%       | 84%    | 2%    |

Comparing the results of VGG-16, VGG-24, and VGG-32, we find that when the depth increases, the networks become harder 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.

#### 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%    |
| $L_{\infty,0.5\%}$ | 78%    | 19%    | 76%    | 16%   |
| $L_{\infty,1.0\%}$ | 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%    |
| $L_{\infty,0.5\%}$ | 84%     | 0%       | 87%    | 0%    |
| $L_{\infty,1.0\%}$ | 83%     | 5%       | 86%    | 5%    |

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 $F_{\Theta_a}$ by calculating the adversarial accuracy with PGD-10. In this section, we conduct experiments with three more adversarial attacks on the attacked network $F_{\Theta_a}$: 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 $L_{\infty}$ 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.

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 $F_{\Theta_a}$ and use PGD-(10,8/255) to attack $F_{\Theta_a}$. 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
Adversarial Parameter Attack on Deep Neural Networks

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.

| Attack | CIFAR-100 | Tiny-ImageNet |
|--------|-----------|---------------|
| No attack | 58% | 28% | 40% | 19% |
| $L_{\infty, 0.6}$ | 55% | 14% | 36% | 6% |
| $L_{\infty, 0.1}$ | 57% | 9% | 38% | 5% |
| $L_{0, 0.5}$ | 56% | 14% | 40% | 11% |
| $L_{0, 0.1}$ | 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.

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-2009-09*, 2009.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 23IN*, 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. NeurIPS*, 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.
We will show that \( \eta_n \) we can assume that at least

\[
W^{(i)} = \sum_{j \in [m]} W^{(i,j)} v_j.
\]

Proof. For a network \( F : S \to \mathbb{R}^m \) and \( S \subset [0, 1]^n \), we use \( F_i(x) \) to denote the i-th component of \( F(x) \) 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 \( v = (v_1, \ldots, v_n)^T \) and \( P = \arg \min_{P \subseteq [n]} \{|\sum_{i \in P} |v_i| - \sum_{i \notin [n] \setminus P} |v_i|\} \). We can assume \( \sum_{i \in P} |v_i| - \sum_{i \notin [n] \setminus P} |v_i| = k \geq 0 \). For any \( j \in P \) such that \( v_j \neq 0 \) and \( P_j = S/\{j\} \), we have \( |\sum_{i \in P_j} |v_i| - \sum_{i \notin [n] / P_j} |v_i|\| = |2|v_j| - k| \geq k \), which implies \( k \leq |v_j| \).

We now define \( w = (w_1, \ldots, w_n)^T \in \mathbb{R}^n \). 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 \( w_j = -\sum_{\forall i \in P \setminus j} |v_i| + \sum_{\forall i \in [n] \setminus P} |v_i| + v_j \). 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 \( \eta_n/2 \) 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 \( \text{Relu}(W_1(x + \epsilon v) + b_1) + \text{Relu}(W_1(x - \epsilon v) + b_1) \geq 2\text{Relu}(W_1 x + b_1) \), since \( \text{Relu}(x) + \text{Relu}(y) \geq \text{Relu}(x+y) \) for all \( x, y \in \mathbb{R} \). By condition C3, at least \( \eta_n/2 \) coordinates of \( 2\text{Relu}(W_1 x + b_1) \) are bigger than \( 2b \), but fewer than \( \eta_n/2 \) coordinates of \( \text{Relu}(W_1(x + \epsilon v) + b_1) \) are bigger than \( b \), so at least \( \eta_n/2 \) coordinates of \( \text{Relu}(W_1(x - \epsilon v) + b_1) \) are bigger than \( b \).

Let \( l \in [m] \) such that \( l \neq y_0, \bar{W}_2 = -(W_2^{(y_0)} - W_2^{(l)}) \in \mathbb{R}^{1 \times n}, U \in \mathbb{R}^{n_1 \times n} \) all of whose rows are \( \mu v^T \) (the transposition of \( v \)), and \( U = \text{diag}(\text{sign}(W_2)) \in \mathbb{R}^{n_1 \times n} \). Let \( W_1 = W_1 + U U_v \), and

\[
G(x) = W_2 \text{Relu}(W_1 x + b_1) + b_2.
\]

We will show that \( G \) satisfies the condition of the theorem.

Since \( v \perp x_0 \), we have \( G(x_0) = F(x_0) \) and \( G \) gives the correct label for \( x_0 \). Since \( ||v||_\infty = 1 \), we have \( ||W_1 - W_1||_\infty = ||UU_v||_\infty = ||U||_\infty = ||\mu v||_\infty \leq \mu \), and thus \( ||\Theta_a - \Theta||_\infty \leq \mu \).

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

\[
G(x_0 + \epsilon v)
= W_2 \text{Relu}(W_1 x_0 + \epsilon v) + b_2
= W_2 \text{Relu}(W_1 x_0 + \epsilon v) + b_1 + \epsilon U U_v + b_2
\]

we have

\[
G_{y_0}(x_0 + \epsilon v) - G_{l_2}(x_0 + \epsilon v)
= F_{y_0}(x_0 + \epsilon v) - F_{l_2}(x_0 + \epsilon v)+
\bar{W}_2(\text{Relu}(W_1 x_0 + \epsilon v) + b_1) - \text{Relu}(W_1 x_0 + \epsilon v) + b_1 + \epsilon U U_v).
\]

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

\[
-\bar{W}_2(\text{Relu}(W_1 x_0 + \epsilon v) + b_1) - \text{Relu}(W_1 x_0 + \epsilon v) + b_1 + \epsilon U U_v)
= ||\bar{W}_2|| (\text{Relu}(W_1 x_0 + \epsilon v) + b_1) - \text{Relu}(W_1 x_0 + \epsilon v) + b_1 + \epsilon U U_v).
\]
Since \( \epsilon U_U v = \epsilon \mu ||v||_2^2 \text{sign}(W_2) \) and \( ||v||_2 \geq \sqrt{n-1} \), each weight of \( |\epsilon U_U v| \) 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 \( (\text{Relu}(W_1(x + v) + b_1))_i > b \), then \( (|\text{Relu}(W_1(x + v) + b_1) - \text{Relu}(W_1(x + v) + b_1 + \epsilon U_U v))_i \geq \min\{\mu (n-1), b\} \). Since at least \( \eta_1/2 \) coordinates of \( \text{Relu}(W_1(x + v) + b_1) \) are bigger than \( b \), we have \( ||\text{Relu}(W_1(x + v) + b_1) - \text{Relu}(W_1(x + v) + b_1 + \epsilon U_U v)||_1 \geq \eta_1/2 \min\{\mu (n-1), b\} \). By condition \( C_2 \), it is easy see

\[
-\overline{W}_2(\text{Relu}(W_1(x + v) + b_1) - \text{Relu}(W_1(x + v) + b_1 + \epsilon U_U v))
\]

\[
= |\overline{W}_2|(|\text{Relu}(W_1(x + v) + b_1) - \text{Relu}(W_1(x + v) + b_1 + \epsilon U_U v))|
\]

\[
\geq \|\overline{W}_2\|_{-\infty}||\text{Relu}(W_1(x + v) + b_1) - \text{Relu}(W_1(x + v) + b_1 + \epsilon U_U v)||_1
\]

\[
\geq \min\{\mu (n-1), b\} c n_1 \eta/2,
\]

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

\[
\mathcal{G}_{y_0}(x_0 + e v) - \mathcal{G}_{l_2}(x_0 + e v)
\]

\[
= \mathcal{F}_{y_0}(x_0 + e v) - \mathcal{F}_{l_2}(x_0 + e v) + \overline{W}_2(\text{Relu}(W_1(x + v) + b_1) - \text{Relu}(W_1(x + v) + b_1 + \epsilon U_U v))
\]

\[
\leq A - \min\{\mu (n-1), b\} c n_1 \eta/2 < 0.
\]

Thus if \( n_1 > \frac{2A}{\min\{\mu (n-1), b\} c n_1} \), then \( \mathcal{G}_{y_0}(x_0 + e v) - \mathcal{G}_{l_2}(x_0 + e v) < 0 \) and the label of \( \mathcal{G}(x_0 + e v) \) 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 \( \text{sign}(W x + b - c1) \) has at least \( k \) weights which are 1 for \( a > 0 \) and \( k \in \mathbb{Z} \), then

\[
W_1 \text{Relu}(W x + b) > W_1 \text{Relu}(W x + b - \gamma \text{sign}(W_1)) + k ||W||_{-\infty} \min\{c, \gamma\}.
\]

**Proof.** We first have two simple results:

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

(2) 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 \( \text{Relu} \) is a monotonically increasing function.

For the \( j \)-th weight of \( W x + b \), we have that

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

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

Since at least \( k \) coordinate \( j \) satisfying \( (W x + b)_j > c \), it holds

\[
W_1 \text{Relu}(W x + b) - W_1 \text{Relu}(W x + b - \gamma \text{sign}(W_1)) \geq k ||W||_{-\infty} \min\{c, \gamma\}.
\]

The lemma is proved.

**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 \( ||v||_2 = 1 \). Then \( ||v||_\infty \leq 1. \)
By condition $C_3$, for any $x \in S$, at least $\eta n_1/2$ coordinates of Relu($W_1(x + \epsilon v_y) + b_1$) are bigger than $b$ or at least $\eta n_1/2$ coordinates of Relu($W_1(x - \epsilon v_y) + b_1$) are bigger than $b$. This is because, from Relu($a$) + Relu($b$) $\geq$ Relu($a + b$) we have

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

Thus from $C_3$, at least $\eta n_1$ coordinates of Relu($W_1 x + b_1$) 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 $G(x, y) = \|\text{sign}(x - y1)\|_0$. 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 $P(e \text{ or } f) \leq P(e) + P(f)$. We thus have $P_{(x,y) \sim D_S}(G(\text{Relu}(W_1(x + \epsilon v_y) + b_1), b) \geq \eta n_1/2) \geq 0.5 \text{ or } P_{(x,y) \sim D_S}(G(\text{Relu}(W_1(x - \epsilon v_y) + b_1), b) \geq \eta n_1/2) \geq 0.5$. Without loss of generality, we can assume that

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

For $l \in [m]$, let $\bar{W}_2^{(l)} = W_2^{(l)} - W_2^{(l+1)}$, where $W_2^{(m+1)} = W_2^{(1)}$. Now assume $U_v \in \mathbb{R}^{n_1 \times n}$, whose rows are all $\mu v_1$, and $U_l = -\text{diag}(\text{sign}(\bar{W}_2^{(l)})).$ Let $\bar{W}_1 = W_1 + \sum_{l=1}^{m} U_l U_v$, and

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

We will show that $G(x)$ satisfies the conditions of the theorem.

It is easy to see that $\|\Theta_\gamma - \Theta\|_\infty \leq \mu$, where $\Theta$ is the parameter of $F$ and $\Theta_\gamma$ is the parameter of $G$. For any $x \in S$, by $C_4$, we have $v_1 x = 0$ for all $l \in [m]$, so $U_v x = 0$. Then $\bar{W}_1 x = W_1 x + \sum_{l=1}^{m} (U_l U_v) x = W_1 x$, which means $G(x) = F(x)$, and thus the accuracy of $G$ over $D_S$ is equal to that of $F$.

Moreover, we have that

$$\bar{W}_1(x + \epsilon v_y) = W_1 x + \epsilon \bar{W}_1 v_y = W_1 x + \epsilon W_1 v_y + \frac{\epsilon}{m} \sum_{l=1}^{m} U_l v_y = W_1 x + \epsilon W_1 v_y + \frac{\epsilon}{m} U_l U_v(y) v_y = W_1(x + \epsilon v_y) - \frac{\epsilon}{m} \text{sign}(\bar{W}_2^{y}).$$

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

$$G_y(x + \epsilon v_y) - G_y+1(x + \epsilon v_y) = (W_2^{y} - W_2^{y+1}) \text{Relu}(\bar{W}_1(x + \epsilon v_y) + b_1) + b_2^{y} - b_2^{y+1} = \bar{W}_2^{y} \text{Relu}(W_1(x + \epsilon v_y) - \frac{\mu \epsilon}{m} \text{sign}(\bar{W}_2^{y}) + b_1) + b_2^{y} - b_2^{y+1} \leq \bar{W}_2^{y} \text{Relu}(W_1(x + \epsilon v_y) + b_1) - \min\{\epsilon \mu / m, b\} cn_1 \eta / 2 + b_2^{y} - b_2^{y+1} \text{ (by Lemma 4.2)}$$

$$\leq A - \min\{\epsilon \mu / m, b\} cn_1 \eta / 2 < 0.$$

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

$$P_{(x,y) \sim D_S}(G \text{ has an adversarial example of } x \in B_{\infty}(x, \epsilon)) \geq 0.5$$

which is equivalent to the theorem. 

$\square$
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 $V_{\text{Relu}}(U) \leq V_{\text{Relu}}(U + k1^{(1)})$.
2. If $V^{(1)} < 0$, then $V_{\text{Relu}}(U) \geq V_{\text{Relu}}(U + k1^{(1)}) + (k - \|U\|_{\infty})|V^{(1)}|$.

Proof. It is easy to see

$$V_{\text{Relu}}(U) - V_{\text{Relu}}(U + k1^{(1)}) = V^{(1)}(\text{Relu}(U))^{(1)} - (\text{Relu}(U + k1^{(1)}))^{(1)}.$$

Since $\text{Relu}$ 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 $(\text{Relu}(U))^{(1)} - (\text{Relu}(U + k1^{(1)}))^{(1)} \leq -(k - |U^{(1)}|) \leq -(k - \|U\|_{\infty})$. If $V^{(1)} < 0$, then $(V^{(1)})(\text{Relu}(U))^{(1)} - (\text{Relu}(U + k1^{(1)}))^{(1)} \geq |V^{(1)}|(k - \|U\|_{\infty})$, 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 $\Theta_\nu \in \mathbb{R}^k$ such that $\|\Theta_\nu - \Theta\|_0 \leq \nu$, $\text{AC}(\mathcal{F}_{\Theta_\nu}) = \text{AC}(\mathcal{F}_\Theta)$, and $\text{AA}(\mathcal{F}_{\Theta_\nu}, \epsilon) = 0$.

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 $T = \max_{i \in [m], \|x\|_2 = 1} \{(W_{2}^{(i)})\}$. For $x \in S$, let $\text{CW}(\mathcal{F}, x) = \max_{y \neq y} \{(\mathcal{F}_y(x) - \mathcal{F}_x(x))\}$ 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 $\|W_1 x + b\|_\infty < C$. Let $K = C/\epsilon + 2T + C/\epsilon$. If $\tilde{W}_1 \in \mathbb{R}^{n_1 \times n}$ is defined as $\tilde{W}_1^{(j)} = W_1^{(j)} + K(\sum_{i \in [m]} I(j = k_i)v_i)$ for all $j \in [n_1]$, that is, add $Kv_l$ to the $k_l$-th row of $W_1$. Let

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

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

For any $x \in S$, by condition $C_2$, we have $v_l x = 0$ for all $l \in [m]$, so $\tilde{W}_1 x = W_1 x$, which means $\mathcal{G}(x) = \mathcal{F}(x)$. Thus the accuracy of $\mathcal{G}$ over $D_\nu$ is equal to that of $\mathcal{F}$.

For any $x \in S$, we know that $x$ satisfies $\text{CW}(\mathcal{F}, x) < C$ and $\|W_1 x + b\|_\infty < C$, then

$$\mathcal{G}_y(x + \epsilon v_y) = W_2^{(y)} \text{Relu}(W_1(x + \epsilon v_y) + 1_{k_y} K \epsilon + b) + b_2^{(y)}$$

$$= W_2^{(y)} \text{Relu}(W_1 x + b + 1_{k_y} K \epsilon + \epsilon W_1 v_y) + b_2^{(y)}$$

$$\leq W_2^{(y)} \text{Relu}(W_1 x + b + 1_{k_y} K \epsilon) + \epsilon |W_2^{(y)}||W_1 v_y| + b_2^{(y)}$$

$$\leq W_2^{(y)} \text{Relu}(W_1 x + b + 1_{k_y} K \epsilon) + \epsilon T + b_2^{(y)}$$

$$\leq W_2^{(y)} \text{Relu}(W_1 x + b) - |W_2^{(y,k_y)}|(K \epsilon - C) + \epsilon T + b_2^{(y)}$$

$$= \mathcal{F}_y(x) - C - \epsilon T + b_2^{(y)}$$

\[\square\]
Adversarial Parameter Attack on Deep Neural Networks

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

$$G_{l_2}(x + \epsilon v_y) = W_2^{(l_2)}\text{Relu}(W_1 x + b + 1_{k_y}K \epsilon + \epsilon W_1 v_y) + b_2^{(l_2)} \geq W_2^{(l_2)}\text{Relu}(W_1 x + b + 1_{k_y}K \epsilon) - \epsilon|W_2^{(l_2)}||W_1 v_y| + b_2^{(l_2)}$$

(Since $F_y(x) - C < F_{l_2}(x)$, we have $G_{l_2}(x + \epsilon v_y) \geq F_{l_2}(x) - \epsilon T > F_y(x) - C - \epsilon T \geq G_y(x + \epsilon v_y)$, so $G$ does not give the correct label to $x + \epsilon v_y$, which means that $G$ 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 $|\Theta| - \Theta| \leq \nu = (n + m + 1)m = \frac{(n + m + 1)m}{n_1 + m}|\Theta| \leq \frac{n + m + 1}{n + m + 1 + \eta}|\Theta| < \eta|\Theta|$. The theorem is proved. $\square$

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

$$\text{Loss}_{CE}(T) = \frac{1}{|T|} \sum_{(x,y) \in T} L_{CE}(F_{\Theta a}(x), y),$$

and the adversarial loss

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

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](image)
Then, after Phase two starts, $L_{CE}(T)$ and $L_{AT}(T)$ fall, but $L_{CE}(T)$ falls more rapidly and reaches a small value 0.1. But $L_{AT}(T)$ falls slowly and finally stabilizes at a relatively large value. So, $\frac{L_{CE}(T)}{L_{AT}(T)}$ 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.

\[ L(a < b) \]

Note that \( ||\nabla F(x, y)|| \leq C \) for any \( i \in [m] \), \( x \in [0, 1]^n \) and \( C \) depends on \( \Theta \). 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

\[ |\nabla L_{CE}(F_{\Theta}(x), y)| < C(1 - e^{-L_{CE}(F_{\Theta}(x), y)}) \]

Similarly, we also have that

\[ |\nabla L_{AT}(F_{\Theta}(x), y)| < CI(1 - e^{-L_{AT}(F_{\Theta}(x), y)}) \]
Substituting the inequalities (13) and (14) into equation (11), we have
\[
|\nabla_{\Theta} L_{\text{CE}}(F_{\Theta}(x), y) - L_{\text{AT}}(F_{\Theta}(x), y)| < C\left( \frac{1 - e^{-L_{\text{CE}}(F_{\Theta}(x), y)}}{L_{\text{AT}}(F_{\Theta}(x), y)} + \frac{L_{\text{CE}}(F_{\Theta}(x), y)(1 - e^{-L_{\text{AT}}(F_{\Theta}(x), y)})}{(L_{\text{AT}}(F_{\Theta}(x), y))^2} \right)\mathbf{1},
\]
Note that, when \( x \in (0, \infty), x \geq 1 - e^{-x} \). Then
\[
|\nabla_{\Theta} L_{\text{CE}}(F_{\Theta}(x), y) - L_{\text{AT}}(F_{\Theta}(x), y)| < 2C \left( \frac{L_{\text{CE}}(F_{\Theta}(x), y)}{L_{\text{AT}}(F_{\Theta}(x), y)} \right)\mathbf{1}
\]
from which (10) is proved.

From Lemma C.1, for a well trained \( F_{\Theta} \), if \( L_{\text{CE}}(F_{\Theta}(x), y) \) is very small comparing with \( L_{\text{AT}}(F_{\Theta}(x), y) \), then the value of \( \nabla_{\Theta} L_{\text{CE}}(F_{\Theta}(x), y) - L_{\text{AT}}(F_{\Theta}(x), y) \) also becomes very small and thus we have gradient vanishing. Phase one of Algorithm 1 is to make \( L_{\text{CE}}(F_{\Theta}(x), y) / L_{\text{AT}}(F_{\Theta}(x), y) \) 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), using 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 \( L_{\text{CE}} \) and \( L_{\text{AT}} \) on the training set, as show in Table 4:

| 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 |

The value of \( L_{\text{CE}} \) is much smaller than that of \( L_{\text{AT}} \).

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:

| 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.
Adversarial Parameter Attack on Deep Neural Networks

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 $L_{CE}/L_{AT}$ 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.

| 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 $L_{CE} \approx L_{AT}$, 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.

| Attack | $L_{CE}$ | $L_{AT}$ |
|--------|---------|---------|
| No attack | 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.

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

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