A Theoretical View of Linear Backpropagation and Its Convergence

Ziang Li*, Yiwen Guo*, Haodi Liu, and Changshui Zhang, *Fellow, IEEE*

Abstract—Backpropagation (BP) is widely used for calculating gradients in deep neural networks (DNNs). Applied often along with stochastic gradient descent (SGD) or its variants, BP is considered as a de-facto choice in a variety of machine learning tasks including DNN training and adversarial attack/defense. Recently, a linear variant of BP named LinBP was introduced for generating more transferable adversarial examples for performing black-box attacks, by Guo *et al.* [1]. Although it has been shown empirically effective in black-box attacks, theoretical studies and convergence analyses of such a method is lacking. This paper serves as a complement and somewhat an extension to Guo *et al.*'s paper, by providing theoretical analyses on LinBP in neural-network-involved learning tasks, including adversarial attack and model training. We demonstrate that, somewhat surprisingly, LinBP can lead to faster convergence in these tasks in the same hyper-parameter settings, compared to BP. We confirm our theoretical results with extensive experiments. Code for reproducing our experimental results is available at https://github.com/lzalza/LinBP.

Index Terms—Backpropagation, deep neural networks, optimization, convergence, adversarial examples, adversarial training

1 Introduction

Over the past decade, the surge of research on deep neural networks (DNNs) has been witnessed. Powered with large-scale training, DNN-based models have achieved state-of-art performance in a variety of real-world applications. Tremendous amount of research has been conducted to improve the architecture [2], [3], [4], [5], [6] and the optimization [7], [8], [9], [10], [11] of DNNs.

The optimization of DNNs often involves stochastic gradient descent (SGD) [12] (that minimizes some training loss) and backpropagation (BP) [13] (for computing gradients of the loss function). In the work of Guo *et al.* published at NeurIPS 2020 [1], a different way of computing "gradients" was proposed. The method, *i.e.*, linear BP (LinBP), keeps the forward pass of BP unchanged while skips the partial derivative regarding some of the non-linear activations in the backward pass. It was shown empirically that LinBP led to improved results in generating transferable adversarial examples for performing *black-box attacks* [14], in comparison with using the original BP. This paper was written as a complement and somewhat an extension to Guo *et al.*'s paper, in a way that we delve deep into the theoretical properties of LinBP. Moreover, we target two more applications of LinBP, *i.e.*, *white-box attack* and model parameter training, to study the optimization convergence of learning with LinBP.

- *Z. Li, H. Liu, and C. Zhang are with the Institute for Artificial Intelligence, Tsinghua University (THUAI), State Key Lab of Intelligent Technologies and Systems, Beijing National Research Center for Information Science and Technology (BNRist), Department of Automation, Tsinghua University, Beijing 100084, China.*
  E-mail: liza19@mails.tsinghua.edu.cn, liuhd21@mails.tsinghua.edu.cn, zcs@mail.tsinghua.edu.cn.
- *Y. Guo is an independent researcher, Beijing 100000, China.*
  E-mail: guoyiwen89@gmail.com.
- **These two authors contribute equally.*
- *Correspondence to: Y. Guo and C. Zhang.*

In this paper, we will provide convergence analyses for LinBP, and we will demonstrate that, in white-box adversarial attack scenarios, using LinBP can produce more deceptive adversarial examples (similar to the results in black-box scenarios as given in [1]), than using the standard BP in the same hyper-parameter settings. Plausible explanations of such fast convergence will be given. Similarly, we will also show theoretically that LinBP helps converge faster in training DNN models, if the same hyper-parameters are used for the standard BP and LinBP. Simulation experiments confirm our theoretical results, and extensive results will verify our findings in more general and practical settings using a variety of widely used DNNs, including VGG-16 [2], ResNet-50 [3], DenseNet-161 [4], MobileNetV2 [15], and WideResNet (WRN) [16].

The rest of this paper is organized as follows. In Section 2, we will revisit preliminary knowledge about network training and adversarial examples, to introduce LinBP and discuss its applications. In Section 3, we will provide theoretical analyses on the convergence of LinBP in both the white-box attack and model training scenarios. In Section 4, we will conduct extensive simulation and practical experiments to validate our theoretical findings and test LinBP with different DNNs. In Section 5, we draw conclusions.

2 Background and Preliminary Knowledge

2.1 Model Training

Together with SGD or its variant, BP has long been adopted as a default method for computing gradients in training machine learning models. Consider a typical update rule of SGD, we have

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla_{\mathbf{w}_t} \sum_k \mathcal{L}(\mathbf{x}^{(k)}, y^{(k)}), \tag{1}$$

where $\{\mathbf{x}^{(k)}, y^{(k)}\}$ contains the k-th training instance and its label, $\nabla_{\mathbf{w}} \sum_k \mathcal{L}(\mathbf{x}^{(k)}, y^{(k)})$ is the partial gradient of the loss

function with respect to a learnable weight vector $\mathbf{w}$, and $\eta \in \mathbb{R}^+$ represents the learning rate.

2.2 White-box Adversarial Attack

Another popular optimization task in deep learning is to generate adversarial examples, which is of particular interest in both the machine learning community and the security community. Instead of minimizing the prediction loss as in the objective of model training, this task aims to craft inputs that lead to arbitrary incorrect model predictions [17]. The adversarial examples are expected to be perceptually indistinguishable from the benign ones. Adversarial examples can be untargeted or targeted, and in this paper we focus on the former. In the white-box setting, where it is assumed that the architecture and parameters of the victim model are both known to the adversary, we have the typical learning objectives for generating adversarial examples:

$$\max_{\|\mathbf{r}\|_p \leq \epsilon} \mathcal{L}(\mathbf{x}+\mathbf{r}, \mathbf{y}) \tag{2}$$

and

$$\min \|\mathbf{r}\|_p, \text{ s.t., } \arg\max_j \text{prob}_j(\mathbf{x}+\mathbf{r}) \neq \arg\max_j \text{prob}_j(\mathbf{x}) \tag{3}$$

where $\mathbf{r}$ is a perturbation vector whose l_p norm is constrained (*e.g.,* $\|\mathbf{r}\|_p < \epsilon$) to guarantee that the generated adversarial example $\mathbf{x}+\mathbf{r}$ is perceptually similar to the benign example $\mathbf{x}$, function $\text{prob}_j(\cdot)$ provides the prediction probability for the j-th class. For solving the optimization problems in Eq. (2) and (3), a series of methods have been proposed over the last decade. For instance, with $p=\infty$, *i.e.,* for performing l_∞ attack, Goodfellow *et al.* [17] proposed the fast gradient sign method (FGSM) which simply calculates the sign of the input gradient, *i.e.,* $\text{sign}(\nabla_{\mathbf{x}} L(\mathbf{x}))$, and adopted $\mathbf{r} = \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} L(\mathbf{x}))$ as the perturbation. To enhance the power of the attack, follow-up methods proposed iterative FGSM (I-FGSM) [18] and PGD [19] that took multiple steps of update and computed input gradients using BP in each of the steps. The iterative process of these methods is similar to that of model training, except that the update is performed in the input space instead of the parameter space, and normally some constraints are required for attacks. In transfer-based attacks, I-FGSM is widely adopted as a baseline method while it has been reported that PGD is capable of bypassing gradient masking [20]. Other famous attacks include DeepFool [21] and C&W's attack [22], just to name a few.

2.3 Linear Backpropagation

Inspired by the hypothesis made by Goodfellow *et al.* [17] that the linear nature of modern DNNs causes the transferability of adversarial samples, LinBP, a method improves the linearity of DNNs, was proposed. For a d-layer DNN model, the forward pass includes the computation of

$$f_d(\mathbf{x}) = \mathbf{W}_d^T \sigma\left(\mathbf{W}_{d-1}^T \cdots \sigma(\mathbf{W}_1^T \mathbf{x})\right), \tag{4}$$

where $\sigma(\cdot)$ is the activation function and it is often set as the ReLU function, $\mathbf{W}_1, \cdots, \mathbf{W}_d$ are learnable weight matrices in the DNN model. In order to generate a white-box adversarial example on the basis of $\mathbf{x}$ with $f_d(\mathbf{x})$, we shall further compute the prediction loss $\mathcal{L}(\mathbf{x}, \mathbf{y})$ for $\mathbf{x}$ and then backpropagate the loss to compute the input gradient $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{y})$, the default BP for computing the input gradient is formulated as

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{y}) = \mathbf{M}_1 \cdots \mathbf{M}_{d-1} \mathbf{W}_\mathbf{d} \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y})}{\partial f_d(\mathbf{x})}, \tag{5}$$

where $\mathbf{M}_i = \mathbf{W}_i \partial\sigma(f_i(\mathbf{x}))/\partial f_i(\mathbf{x})$. LinBP keeps the computation in the forward pass unchanged while removing the influence of the ReLU activation function (*i.e.,* $\partial\sigma(f_i(\mathbf{x}))/\partial f_i(\mathbf{x})$) in the backward pass. That being said, LinBP computes:

$$\tilde{\nabla}_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{y}) = \mathbf{W}_1 \cdots \mathbf{W}_d \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y})}{\partial f_d(\mathbf{x})}. \tag{6}$$

Since the partial derivative of the activation function has been removed, it is considered linear in the backward pass and thus called linear BP (LinBP). In practice, LinBP may only remove some of the nonlinear derivatives, *e.g.,* only modifies those starting from the $(m+1)$-th layer and uses the following formulation:

$$\tilde{\nabla}_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{y}) = \mathbf{M}_1 \cdots \mathbf{M}_m \mathbf{W}_{m+1} \cdots \mathbf{W}_d \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y})}{\partial f_d(\mathbf{x})}. \tag{7}$$

Experimental results on CIFAR-10 [23] and ImageNet [24] tested the transferability of generated adversarial examples on different source models and found that LinBP achieved state-of-the-arts [1].

In addition to generating adversarial examples, we can also adopt LinBP to compute the gradient with respect to model weights for training DNNs. Note that the gradient of the loss with respect to $\mathbf{W}_i$ in the standard BP is formulated as

$$\begin{aligned} \nabla_{\mathbf{W}_i} \mathcal{L}(\mathbf{x}, \mathbf{y}) =& \sigma\left(f_{i-1}(\mathbf{x})\right) \Bigg(\frac{\partial \sigma(f_i(\mathbf{x}))}{\partial f_i(\mathbf{x})} \mathbf{M}_{i+1} \cdots \\ & \mathbf{M}_{d-1} \mathbf{W}_d \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y})}{\partial f_d(\mathbf{x})} \Bigg)^T, \end{aligned} \tag{8}$$

where we abuse the notations and define $\sigma(f_0(\mathbf{x})) = \mathbf{x}$ for simplicity. Just like in Eq. (6), LinBP computes a linearized version of the gradient, which is formulated as

$$\tilde{\nabla}_{\mathbf{W}_i} \mathcal{L}(\mathbf{x}, \mathbf{y}) = \sigma\left(f_{i-1}(\mathbf{x})\right) \left(\mathbf{W}_{i+1} \cdots \mathbf{W}_d \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y})}{\partial f_d(\mathbf{x})} \right)^T. \tag{9}$$

3 Theoretical Analyses

If we consider skipping ReLU in the backward pass as finding an approximation to the gradient of a non-smooth objective function, then LinBP is somewhat related to the straight-through estimation [25]. However, such a possible relation does not provide much insight of the convergence of LinBP, which is the focus of this paper. In this section, we shall provide convergence analyses for LinBP, and compare it to the standard BP. There have been attempts for studying the training dynamics of neural networks [26], [27], [28], [29]. We follow Tian's work [28] and consider a teacher-student framework with the ReLU activation function and squared l_2 loss, based on no deeper than two-layer networks. We shall start from theoretical analyses for white-box adversarial attacks and then consider model training.

3.1 Theoretical Analyses for Adversarial Attack

Compared to the two-layer teacher-student frameworks in Tian's work [28], we here consider a more general model, which can be formulated as

$$g(\mathbf{W}, \mathbf{V}, \mathbf{x}) = \mathbf{V}\sigma(\mathbf{W}\mathbf{x}), \tag{10}$$

where $\mathbf{x} \in \mathbb{R}^{d_1}$ is the input data vector, $\mathbf{W} \in \mathbb{R}^{d_2 \times d_1}$ and $\mathbf{V} \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices, $\sigma(\cdot)$ is the ReLU function which compares its inputs to 0 in an element-wise manner. In the teacher-student frameworks, we assume the teacher network possesses the optimal adversarial example $\mathbf{x}^\star$, and the student network aims to learn an adversarial example $\mathbf{x}$ from the teacher network, in which the loss function is set as the squared l_2 loss between the output of the student and the teacher network, *i.e.*,

$$\mathcal{L}(\mathbf{x}) = \frac{1}{2}\|g(\mathbf{W}, \mathbf{V}, \mathbf{x}) - g(\mathbf{W}, \mathbf{V}, \mathbf{x}^\star)\|_2^2. \tag{11}$$

We further define $D(\mathbf{W}, \mathbf{x}) := \mathrm{diag}(\mathbf{W}\mathbf{x} > 0)$. From Eq. (10) and Eq. (11), we can easily obtain the analytic expression of the gradient with respect to $\mathbf{x}$ for the standard BP:

$$\begin{aligned}\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}) =& \mathbf{W}^T D(\mathbf{W}, \mathbf{x})\mathbf{V}^T\mathbf{V}(D(\mathbf{W}, \mathbf{x})\mathbf{W}\mathbf{x} \\ &- D(\mathbf{W}, \mathbf{x}^\star)\mathbf{W}\mathbf{x}^\star).\end{aligned} \tag{12}$$

Compared with Eq. (12), the gradient obtained from LinBP removes the derivatives of the ReLU function in the backward pass, *i.e.*,

$$\tilde{\nabla}_{\mathbf{x}}\mathcal{L}(\mathbf{x}) = \mathbf{W}^T\mathbf{V}^T\mathbf{V}\left(D(\mathbf{W}, \mathbf{x})\mathbf{W}\mathbf{x} - D(\mathbf{W}, \mathbf{x}^\star)\mathbf{W}\mathbf{x}^\star\right). \tag{13}$$

Inspired by Theorem 1 in Tian's work [28], we introduce the following lemma to give the analytic expression of the gradients in expectations in adversarial settings.

Lemma 1. *$G(\mathbf{e}, \mathbf{x}) := \mathbf{W}^T D(\mathbf{W}, \mathbf{e})\mathbf{V}^T\mathbf{V}D(\mathbf{W}, \mathbf{x})\mathbf{W}\mathbf{x}$, where $\mathbf{e} \in \mathbb{R}^{d_1}$ is a unit vector, $\mathbf{x} \in \mathbb{R}^{d_1}$ is the input data vector, $\mathbf{W} \in \mathbb{R}^{d_2 \times d_1}$ and $\mathbf{V} \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices. If $\mathbf{W}$ and $\mathbf{V}$ are independent and both generated from the standard Gaussian distribution, we have*

$$\mathbb{E}\left(G(\mathbf{e}, \mathbf{x})\right) = \frac{d_3 d_2}{2\pi}[(\pi - \Theta)\mathbf{x} + \|\mathbf{x}\|\sin\Theta\mathbf{e}],$$

where $\Theta \in [0, \pi]$ is the angle between $\mathbf{e}$ and $\mathbf{x}$.

The proof of Lemma 1 can be found in the appendices. With Lemma 1, the expectation of Eq. (12) and Eq. (13) can be formulated as

$$\begin{aligned}\mathbb{E}[\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x})] &= G(\mathbf{x}/\|\mathbf{x}\|, \mathbf{x}) - G(\mathbf{x}/\|\mathbf{x}\|, \mathbf{x}^\star) \\ &= \frac{d_3 d_2}{2}(\mathbf{x} - \mathbf{x}^\star) + \frac{d_3 d_2}{2\pi}\left(\Theta\mathbf{x}^\star - \frac{\|\mathbf{x}^\star\|}{\|\mathbf{x}\|}\sin\Theta\mathbf{x}\right)\end{aligned} \tag{14}$$

and

$$\begin{aligned}\mathbb{E}[\tilde{\nabla}_{\mathbf{x}}\mathcal{L}(\mathbf{x})] &= G(\mathbf{x}/\|\mathbf{x}\|, \mathbf{x}) - G(\mathbf{x}^\star/\|\mathbf{x}^\star\|, \mathbf{x}^\star) \\ &= \frac{d_3 d_2}{2}(\mathbf{x} - \mathbf{x}^\star),\end{aligned} \tag{15}$$

respectively, where $\Theta \in [0, \pi]$ is the angle between $\mathbf{x}$ and $\mathbf{x}^\star$. We here focus on l_∞ attacks. Different iterative gradient-based l_∞ attack methods may have slightly different update rules. Here, for ease of unified analyses, we simplify their update rules as

$$\mathbf{x}^{(t+1)} = \mathrm{Clip}(\mathbf{x}^{(t)} - \eta\nabla_{\mathbf{x}^{(t)}}\mathcal{L}(\mathbf{x}^{(t)})), \tag{16}$$

where $\mathrm{Clip}(\cdot) = \min(\mathbf{x} + \epsilon\mathbf{1}, \max(\mathbf{x} - \epsilon\mathbf{1}, \cdot))$ performs element-wise input clip to guarantee that the intermediate results always stay in the range fulfilling the constraint of $\|\mathbf{x}^{(t+1)} - \mathbf{x}\|_\infty \leq \epsilon$. We use the updates of the standard BP and LinBP, *i.e.*, $\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x})$ and $\tilde{\nabla}_{\mathbf{x}}\mathcal{L}(\mathbf{x})$, to obtain $\{\mathbf{x}^{(t)}\}$ and $\{\tilde{\mathbf{x}}^{(t)}\}$, respectively. Based on all these results above, we can give the following theorem that provides convergence analysis of LinBP.

Theorem 1. *For the two-layer teacher-student network formulated as in Eq. (10), in which the adversarial attack adopts Eq. (11) and Eq. (16) as the loss function and the update rule, respectively, we assume that $\mathbf{W}$ and $\mathbf{V}$ are independent and both generated from the standard Gaussian distribution, $\mathbf{x}^\star \sim N(\mu_1, \sigma_1^2)$, $\mathbf{x}^{(0)} \sim N(0, \sigma_2^2)$, and η is reasonably small [1]. Let $\mathbf{x}^{(t)}$ and $\tilde{\mathbf{x}}^{(t)}$ be the adversarial examples generated in the t-th iteration of attack using BP and LinBP, respectively, then we have*

$$\mathbb{E}\|\mathbf{x}^\star - \tilde{\mathbf{x}}^{(t)}\|_1 \leq \mathbb{E}\|\mathbf{x}^\star - \mathbf{x}^{(t)}\|_1.$$

The proof of the theorem is deferred to the appendices. Theorem 1 shows that LinBP can produce adversarial examples closer to the optimal adversarial examples $\mathbf{x}^\star$ (when compared with the standard BP) in the same settings for any finite number of iteration steps, which means that LinBP can craft more powerful and destructive adversarial examples even in the white-box attacks. It is also straightforward to derive from our proof that LinBP produces a more powerful adversarial example, if starting optimization from the benign example with low prediction loss. The conclusion is somewhat surprising since popular white-box attack methods like FGSM (I-FGSM) and PGD mostly use the gradient obtained by the standard BP, yet we find LinBP may lead to stronger attacks with the same hyper-parameters.

From our proof, it can further be derived that the norm of Eq. (15) is larger than the norm of Eq. (14), which means LinBP provides larger gradients in expectation. Also, the direction of the update obtained from LinBP is closer to the residual $\mathbf{x} - \mathbf{x}^\star$ in comparison to the standard BP. These may cause LinBP to produce more destructive adversarial examples and more powerful attack in white-box settings. We conducted extensive experiments on deeper networks using common attack methods to verify our theoretical findings. The experimental results will be shown in Section 4.2.

3.2 Theoretical Analyses for Model Training

For model training, we adopt the same framework as in Section 3.1 to analyze the performance of LinBP. By contrast, we mainly consider a one-layer teacher-student framework, since the student network may not converge to the teacher network with two-layer and deeper models. We will show in Section 4.2 that many of our theoretical results still hold in more complex network architectures.

The one-layer network can be formulated as

$$h(\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{x}^T\mathbf{w}), \tag{17}$$

1. See Eq. (22) for more details of the constraint.

where $\mathbf{x} \in \mathbb{R}^d$ is the input vector, $\mathbf{w} \in \mathbb{R}^d$ is the weight vector, and $\sigma(\cdot)$ is the ReLU function. Given a set of training samples, we obtain $h(\mathbf{X}, \mathbf{w}) = \sigma(\mathbf{X}\mathbf{w})$, where $\mathbf{X} = [\mathbf{x}_1^T; ...; \mathbf{x}_N^T]$ is the input data matrix, $\mathbf{x}_k \in \mathbb{R}^d$ is the k-th training instance, for $k = 1, ..., N$. We assume the teacher network has the optimal weight, and the loss function for optimizing the one-layer network can be formulated as

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\|h(\mathbf{X}, \mathbf{w}) - h(\mathbf{X}, \mathbf{w}^\star)\|_2^2, \tag{18}$$

where $\mathbf{w}$ and $\mathbf{w}^\star$ are the weight vectors for the student network and the teacher network, respectively. Therefore, we can derive the gradient with respect to $\mathbf{w}$ for the standard BP as:

$$\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = \mathbf{X}^T\mathbf{D}(\mathbf{X}, \mathbf{w})(\mathbf{D}(\mathbf{X}, \mathbf{w})\mathbf{X}\mathbf{w} - \mathbf{D}(\mathbf{X}, \mathbf{w}^\star)\mathbf{X}\mathbf{w}^\star). \tag{19}$$

By contrast, the gradient with respect to $\mathbf{w}$ obtained from LinBP is formulated as

$$\tilde{\nabla}_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = \mathbf{X}^T(\mathbf{D}(\mathbf{X}, \mathbf{w})\mathbf{X}\mathbf{w} - \mathbf{D}(\mathbf{X}, \mathbf{w}^\star)\mathbf{X}\mathbf{w}^\star). \tag{20}$$

While training the simple one-layer network with SGD, the update rule can be formulated as

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta\nabla_{\mathbf{w}^{(t)}}\mathcal{L}(\mathbf{w}^{(t)}), \tag{21}$$

where we use $\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w})$ and $\tilde{\nabla}_{\mathbf{w}}\mathcal{L}(\mathbf{w})$ showed in Eq. (19) and Eq. (20) to obtain $\{\mathbf{w}^{(t)}\}$ and $\{\tilde{\mathbf{w}}^{(t)}\}$ for the standard BP and LinBP, respectively. The following theorem is given for analyzing the training convergence, when LinBP is used.

Theorem 2. *For the one-layer teacher-student network formulated as in Eq. (17), in which training adopts Eq. (18) and Eq. (21) as the loss function and the update rule, respectively, we assume that $\mathbf{X}$ is generated from the standard Gaussian distribution, $\mathbf{w}^\star \sim N(\mu_1, \sigma_1^2)$, $\mathbf{w}^{(0)} \sim N(0, \sigma_2^2)$, and η is reasonably small*[2]*. Let $\mathbf{w}^{(t)}$ and $\tilde{\mathbf{w}}^{(t)}$ be the weight vectors obtained in the t-th iteration of training using standard BP and LinBP, respectively, then we have*

$$\mathbb{E}\|\mathbf{w}^\star - \tilde{\mathbf{w}}^{(t)}\|_1 \leq \mathbb{E}\|\mathbf{w}^\star - \mathbf{w}^{(t)}\|_1.$$

The proof of Theorem 2 can also be found in the appendices. Theorem 2 shows that LinBP may lead the obtained weight vector to get closer toward the optimal weight vector $\mathbf{w}^\star$ compared to the standard BP with the same learning rate and the same number of training iterations, which means LinBP can also lead to faster convergence in model training.

3.3 Discussions about η and t

In Theorem 1 and Theorem 2, we assume that the update step size in adversarial attacks and the learning rate in model training are reasonably small. Here we would like to discuss more about these assumptions.

Precisely, we mean that η should be small enough to satisfy the following constraints: for Theorem 1, it is required that

$$|\sum_{j=0}^{m-1}\frac{\eta d_3 d_2}{2\pi}(1-\frac{\eta d_3 d_2}{2})^{m-1-j}\mathbf{p}_{ji}| < |(1-\frac{\eta d_3 d_2}{2})^m(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)})|, \tag{22}$$

for $m = 1, \ldots, t$, and $i = 1, \ldots, d_1$, where $\mathbf{p}_j = \theta_j\mathbf{x}^\star - \frac{\|\mathbf{x}^\star\|}{\|\mathbf{x}^{(j)}\|}\sin\theta_j\mathbf{x}^{(j)}$ and i indicates the i-th entry of a d_1-dimensional vector. For Theorem 2, it is required that

$$|\sum_{j=0}^{m-1}\frac{\eta N}{2\pi}(1-\frac{\eta N}{2})^{m-1-j}\mathbf{q}_{ji}| < |(1-\frac{\eta N}{2})^m(\mathbf{w}_i^\star - \mathbf{w}_i^{(0)})|, \tag{23}$$

for $m = 1, \ldots, t$, and $i = 1, \ldots, d$, where $\mathbf{q}_j = \theta_j\mathbf{w}^\star - \frac{\|\mathbf{w}^\star\|}{\|\mathbf{w}^{(j)}\|}\sin\theta_j\mathbf{w}^{(j)}$. It is nontrivial to obtain analytic solutions for Eq. (22) and Eq. (23). However, we can know that they are both more likely to hold when t is small. Since the maximum number of learning iterations for generating adversarial examples is often set to be small (at most several hundred) in commonly adopted methods like I-FGSM and PGD, the assumption in Eq. (22) is easier to be fulfilled than that in Eq. (23), *i.e.,* in the setting of model training which can take tens of thousands of iterations, indicating that the superiority of LinBP can be more obvious in performing adversarial attacks.

4 Experimental Results

In this section, we provide experimental results on synthetic data (see Section 4.1) and real data (see Section 4.2) to confirm our theorem and compare LinBP against BP in more practical settings for white-box adversarial attack/defense and model training, respectively. The practical experiments show that our theoretical results hold in a variety of different model architectures. All the experiments were performed on NVIDIA GeForce RTX 1080/2080 Ti and the code was implemented using PyTorch [30].

4.1 Simulation Experiments

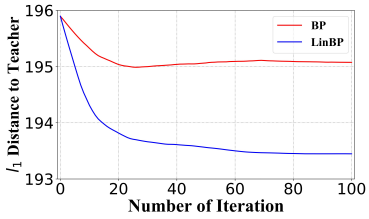
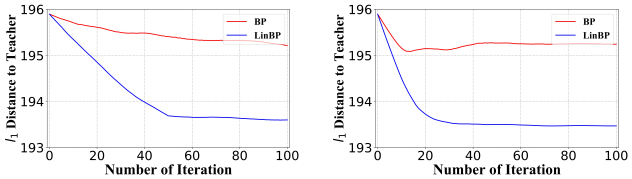


Fig. 1. LinBP leads to more powerful white-box adversarial examples which are closer to the optimal ones.



(a) l_2 normalized updates (b) l_∞ normalized updates

Fig. 2. Using normalized gradients, LinBP still leads to more powerful white-box adversarial examples.

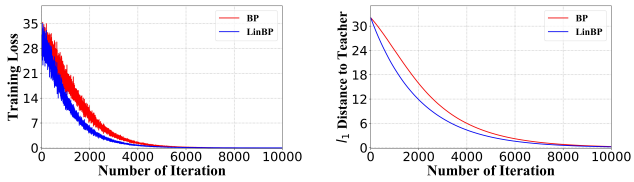
4.1.1 *Adversarial attack*

We constructed a two-layer neural network and performed adversarial attack following the teacher-student framework described in Section 3.1. To be more specific, the victim

2. See Eq. (23) for a precious formulation of the constraint.

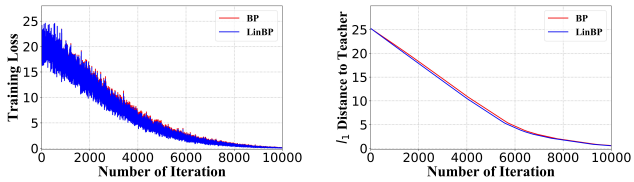
model is formulated as Eq. (10), the loss function is given as Eq. (11), and Eq. (16) is the update rule to generate adversarial examples. Following the assumption in Theorem 1, the weight matrices $\mathbf{V}$ and $\mathbf{W}$ were independent and both generated from the standard Gaussian distribution. Similarly, the optimal adversarial example $\mathbf{x}^\star$ and the initial adversarial example $\mathbf{x}^{(0)}$ were obtained via sampling from Gaussian distributions, *i.e.*, $\mathbf{x}^\star \sim N(\mu_1, \sigma_1^2)$ and $\mathbf{x}^{(0)} \sim N(0, \sigma_2^2)$, where μ_1, σ_1, and σ_2 can in fact be arbitrary constants. Here we set $\mu_1 = 1.0$, $\sigma_1 = 2.0$, and $\sigma_2 = 1.0$. In our experiments, we set $d_1 = 100$, $d_2 = 20$, $d_3 = 10$, $\eta = 0.001$, and $\epsilon = 0.25$. And the maximum number of the iteration step was set to 100. We randomly sampled 10 sets of parameters (including weight matrices, optimal adversarial examples, and initial adversarial examples) for different methods and evaluate the average performance over 10 runs of the experiment. The average l_1 distance between the obtained student adversarial examples and the optimal adversarial examples from the teacher network is shown in Figure 1. It shows that LinBP makes the obtained adversarial examples converge faster to the optimal ones, indicating that LinBP helps craft more powerful adversarial examples in the same hyper-parameter settings, which clearly confirms our Theorem 1.

As we have mentioned in Section 3, the norm of the update obtained from LinBP is larger than that obtained from the standard BP, which may cause faster convergence for LinBP. In this experiment, the l_2 norm and l_∞ norm of the gradient obtained using LinBP are $\sim 1.65\times$ and $\sim 1.60\times$ larger than BP, respectively. For deeper discussions, we have conducted two more experiments. We used l_2 norm and l_∞ norm to normalize the update obtained from LinBP and standard BP. η was set as 0.05 and 0.005, respectively. The results are shown in Figure 2a and Figure 2b, where we can find the conclusion still holds with gradient normalization.



(a) Training loss (b) Approximation to the teacher

Fig. 3. LinBP leads to lower training loss and better approximation to the teacher model, especially at the early stage of training.



(a) Training loss (b) Approximation to the teacher

Fig. 4. LinBP leads to similar training loss and approximation to the teacher model with l_2 normalized gradients.

4.1.2 Model training

As described in Section 3.2, we constructed the one-layer neural network formulated as in Eq. (17) and trained it following the student-teacher framework. Eq. (18) is the loss function. Similar to the experiments in Section 4.1.1, the input data matrix $\mathbf{X}$ was generated from the standard Gaussian distribution, the optimal weight vector $\mathbf{w}^\star$ in the teacher network and the initial weight vector $\mathbf{w}^{(0)}$ in the student network were obtained via sampling from Gaussian distributions with $\mu_1 = 1.0$, $\mu_2 = 0.0$, $\sigma_1 = 3.0$, and $\sigma_2 = 1.0$. We used SGD for optimization and set $N = 100$, $d = 10$, and $\eta = 0.001$ in our experiments. The maximal number of optimization iterations was set to 10000 to ensure training convergence. We used the l_1 distance between the weight vector in the teacher model and that in the student model to evaluate their difference. Figure 3 illustrates the average results over 10 runs and compares the two methods. From the experimental results, we can observe that LinBP leads to more accurate approximation (*i.e.*, smaller l_1 distance) to the teacher as well as lower training loss in comparison with BP, which clearly confirms our Theorem 2, indicating that LinBP can lead to faster convergence in the same hyper-parameter settings.

We found that the norm of gradient obtained from LinBP is larger than that obtained from the standard BP in the early stage of training, similar to the results in Section 4.1.1. To be specific, the l_2 norm of the gradient from LinBP is $\sim 2.2\times$ larger than that from BP in the first 100 iterations, yet over all 10000 iterations, it is only $\sim 0.75\times$. We have also conducted the experiment where the gradient is normalized by its l_2 norm. η was set to 0.0015. The results are shown in Figure 4. We see that the superiority of LinBP drops a lot (when compared with Figure 3), which is different from the results obtained in adversarial attacks.

4.2 More Practical Experiments

We also conducted experiments in more practical settings for adversarial attack and model training on MNIST [33] and CIFAR-10 [23], using DNNs with a variety of different architectures, including a simple MLP, LeNet-5 [33], VGG-16 [2], ResNet-50 [3], DenseNet-161 [4], MobileNetV2 [15], and WRN (WRN) [16].

4.2.1 Adversarial attack on DNNs

We performed white-box attacks on CIFAR-10 using different DNNs, including VGG-16, ResNet-50, DenseNet-161, MobileNetV2, and a robust WRN-28-10 [31] available on the RobustBench [32]. CIFAR-10 test images that could be correctly classified by these models were used to generate adversarial examples, and the attack success rate was adopted to evaluate the performance of the attack. We evaluate the results of PGD [19] as it is popular in white-box attacks, and we report the best attack performance over 5 restarts of PGD. Note that, without restart, PGD and I-FGSM [18] achieve similar performance. The gradient steps were normalized by the l_∞ norm in PGD by default. The update step size of the iterative attack η was fixed to be $1/255$. We tested different settings of the maximum allowed number of update steps K and the perturbation budget ϵ, including $K \in \{10, 20, 100\}$ and $\epsilon \in \{8/255, 16/255\}$. The attack performance using LinBP and BP following all these settings is summarized in Table 1.

It is easy to see in Table 1 that LinBP gains consistently higher attack success rates with PGD, showing that it can

TABLE 1
The highest success rate of white-box attacks using PGD with LinBP and BP in different settings over 5 restarts. K is the maxiumum allowed number of update steps and ϵ is the perturbation budget. Higher success rate (*i.e.*, lower prediction accuracy on the adversarial examples) indicates more powerful attack. "WRN(robust)" represents a robust WRN-28-10 model [31] available on the RobustBench [32].

Method	K	ϵ	VGG-16	ResNet-50	DenseNet-161	MobileNetV2	WRN (robust)
BP	10	8/255	72.28%	66.41%	56.89%	92.94%	33.14%
	20	8/255	87.07%	80.82%	73.33%	98.19%	38.11%
	100	8/255	96.54%	92.53%	86.46%	99.88%	38.94%
	10	16/255	77.25%	71.52%	62.79%	94.90%	33.78%
	20	16/255	94.88%	91.18%	87.38%	99.44%	44.25%
	100	16/255	99.73%	99.81%	98.35%	**100.00%**	59.38%
LinBP	10	8/255	**95.52%**	**85.83%**	**61.87%**	**99.80%**	**33.48%**
	20	8/255	**98.50%**	**93.11%**	**77.72%**	**99.95%**	**39.55%**
	100	8/255	**99.25%**	**95.82%**	**89.03%**	**99.99%**	**40.44%**
	10	16/255	**98.78%**	**92.68%**	**69.04%**	**99.95%**	**34.21%**
	20	16/255	**99.99%**	**99.51%**	**91.18%**	**100.00%**	**45.22%**
	100	16/255	**100.00%**	**99.97%**	**99.07%**	**100.00%**	**60.78%**

TABLE 2
The highest success rate of white-box attacks using Auto-PGD with LinBP and BP in different settings over 5 restarts. ϵ is the perturbation budget. Higher success rate (*i.e.*, lower prediction accuracy on the adversarial examples) indicates more powerful attack. "WRN(robust)" represents a robust WRN-28-10 model [31] available on the RobustBench [32].

Method	ϵ	VGG-16	ResNet-50	DenseNet-161	MobileNetV2	WRN (robust)
BP	8/255	98.06%	95.77%	91.91%	99.97%	39.46%
	16/255	99.97%	99.98%	99.58%	**100.00%**	60.94%
LinBP	8/255	**99.65%**	**97.60%**	**93.35%**	**100.00%**	**41.20%**
	16/255	**100.00%**	**99.99%**	**99.81%**	**100.00%**	**62.99%**

help generate more powerful adversarial examples. For instance, with $K = 10$ and $\epsilon = 8/255$, using LinBP improves the success rate of attacking ResNet-50 by 19.42% (85.83% vs 66.41%). With $K = 100$, the success rates of attacking some (non-robust) models approach ∼100%, and LinBP still helps achieve similar or higher attack performance than using BP. As gradient normalization was by default adopted in PGD, the superiority of LinBP has nothing to do with the norm of gradient, just as in Section 4.1.1.

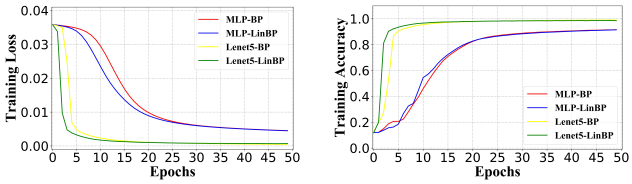
We have also tried using Auto-PGD [34], which was developed as a stronger method for evaluating the adversarial robustness of models. Similar to the experiment with PGD, the best attack performance over 5 restarts are shown (see Table 2). It can be observed that LinBP is still more effective than BP, even with Auto-PGD which has adaptive step sizes. To give more discussions on how LinBP enhances the performance in Auto-PGD, we carefully analyzed when the optimizer adjusted its step sizes with LinBP/BP. The results show that LinBP helps Auto-PGD halve the step size faster. More specifically, the average number of iterations before Auto-PGD halves its step size using LinBP is 22.95, 42.18, 58.59, and 72.11, compared with 25.85, 45.11, 61.98 and 77.45 for BP when attacking ResNet-50 with $\epsilon = 8/255$. Since Auto-PGD adjusts its step size when it reaches "convergence" with the current step size, the results show that LinBP is beneficial to convergence even with such a method in practice.

LinBP can be effective with other activation functions. To gain more insight, we further tested with the Swish/SiLU activation function. For simplicity of experiments, we replace the original ReLU activations in ResNet-50 with the Swish/SiLU activations. With $K = 10$ and $\eta = 1/255$, the PGD attack success rate for LinBP are 75.85% and 87.97% when $\epsilon = 8/255$ and $\epsilon = 16/255$, respectively, compared with 74.24% and 82.65% for BP.

4.2.2 Training DNNs

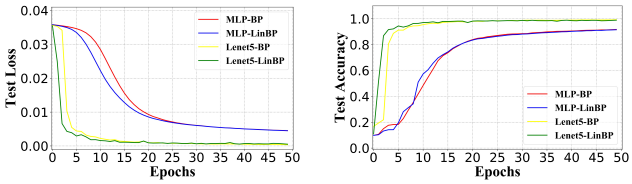
For experiments of DNN training, we first trained and evaluated a simple MLP and LeNet-5 on MNIST. The MLP has four parameterized and learnable layers, and the numbers of its hidden layer units are 400, 200, and 100. We used SGD for optimization and the learning rate was 0.001 and 0.005 for the MLP and LeNet-5, respectively. The training batch size was set to 64, and the training process lasted for at most 50 epochs. The training loss and training accuracy are illustrated in Figure 5, and note that we fix the random seed to eliminate unexpected randomness during training. We can easily observe from the training curves in Figure 5 that the obtained MLP and LeNet-5 models show lower prediction loss and higher accuracy when incorporating LinBP, especially in the early epochs, which suggests that the incorporation of LinBP can be beneficial to the convergence of SGD. The same observation can be made on the test set of MNIST, and the results are shown in Figure 6.

We further report our experimental results on CIFAR-10. ResNet-50 [3], DenseNet-161 [4], and MobileNetV2 [15]



(a) Training loss (b) Training accuracy

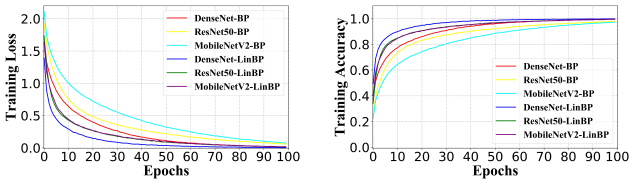
Fig. 5. Compare the training loss and training accuracy of the MLP and LeNet-5 on MNIST, using LinBP or BP.



(a) Test loss (b) Test accuracy

Fig. 6. Compare the test loss and test accuracy of the MLP and LeNet-5 on MNIST, using LinBP or BP.

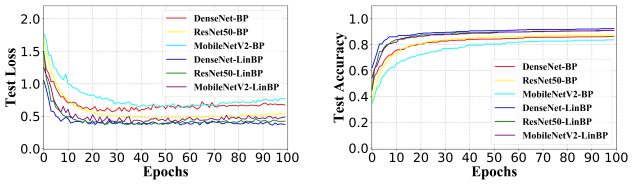
were trained and evaluated on the dataset. The architecture of these networks and their detailed settings can be found in their papers. The optimizer was SGD, and the learning rate was 0.002. We set the batch size to 128 and trained for 100 epochs. We evaluated the prediction loss and prediction accuracy of trained models using LinBP and BP for comparison. Similar to the experiment on MNIST, we fixed the random seed. The training results in Figure 7 and test results in Figure 8 demonstrate that, equipped with LinBP, the models achieved lower prediction loss and higher prediction accuracy in the same training settings, especially when the training just got started. Nevertheless, as training progressed, the superiority of LinBP became less obvious, and the final performance of LinBP became just comparable to that of the standard BP, which is consistent with our theoretical discussions in Section 3.2.



(a) Training loss (b) Training accuracy

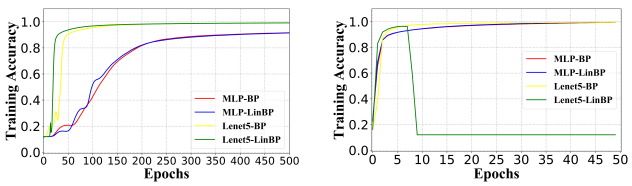
Fig. 7. Training loss and training accuracy of ResNet-50, DenseNet-161, and MobileNetV2 on CIFAR-10, using LinBP or BP.

We also noticed that in certain scenarios, optimization using LinBP may fail to converge with very large learning rates. Here we test various learning rate η in training MNIST models to observe the stability of LinBP. Recall that the base learning rates for training the MLP and LeNet-5 are 0.001 and 0.005, respectively, for obtaining our previous results, and we further tried scaling them by $10\times$ and $0.1\times$ to investigate how the training performance was affected. The training curves for all these settings of learning rates are illustrated in Figure 9. Compared with the results in



(a) Test loss (b) Test accuracy

Fig. 8. Test loss and test accuracy of ResNet-50, DenseNet-161, and MobileNetV2 on CIFAR-10, using LinBP or BP.

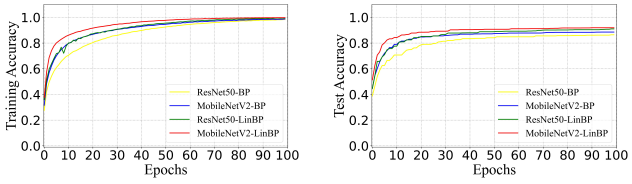


(a) $0.1\times$ learning rate (b) $10\times$ learning rate

Fig. 9. The influence of setting different learning rate to LinBP and BP for training the MLP and LeNet-5 on MNIST.

Figure 5b, it can be seen that, when increasing the base learning rate by $10\times$, the training performance of LinBP and that of BP are similar on the MLP, but, on LeNet-5, the training failed to converge with LinBP while it could still converge with BP. Also, the same observation was made when training VGG-16 on CIFAR-10 using LinBP and a (relatively large) base learning rate of 0.005. The results conform that a reasonably small η helps stabilize training using LinBP and guarantees its performance. We also tried using normalized gradients during training and LinBP performed no better than BP, just like the results in Section 4.1.2.

We used the cosine annealing learning rate scheduler [9] for obtaining the results in Figure 7 and 8. One may also be curious about the training performance with other learning rate scheduler. In this context, we further tested with exponential learning rate scheduler [35] with $\gamma = 0.98$ and find similar results, which are shown in Figure 10.



(a) Training accuracy (b) Test accuracy

Fig. 10. Training ResNet-50 and MobileNetV2 on CIFAR-10 with the exponential learning rate scheduler, using LinBP or BP.

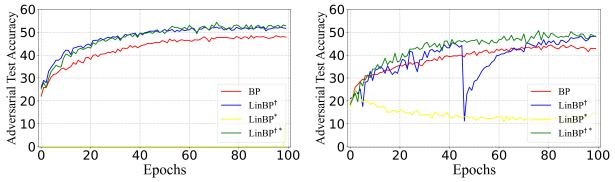
4.2.3 *Adversarial training for DNNs*

Since the discovery of adversarial examples, the vulnerability of DNNs has been intensively discussed. Tremendous effort has been devoted to improving the robustness of DNNs. Thus far, a variety of methods have been proposed, in which adversarial training [17], [19] has become an indispensable procedure in many application scenarios. In this context, we would like to study how LinBP can be adopted to further

enhance the robustness of DNNs. We used the adversarial examples generated by PGD to construct our dataset and we trained our model using simply SGD. There exist multiple strategies of adopting LinBP in adversarial training, *i.e.*, 1) computing gradients for updating model parameters using LinBP as in the model training experiment and 2) generating adversarial examples using LinBP just like in the adversarial attack experiment. We observed that the first strategy is more beneficial than the second strategy when evaluating the robustness of DNNs using BP-generated adversarial examples. Due to overfitting, generating adversarial examples using LinBP during adversarial training leads to obviously larger generalization gap between training and test performance and degraded test robustness against BP-generated adversarial examples.

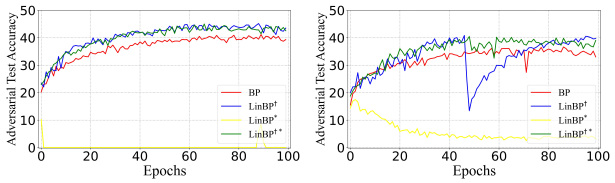
Though effective in the sense of achieving high robust accuracy after convergence, the first strategy sometimes suffers from unstable performance as the adversarial examples and model parameters are updated asynchronously. Further applying the second strategy in combination with the first strategy stabilize adversarial training and is helpful to achieve reliable performance. See Figures 11 and 12 for performance under the PGD attack and AutoAttack, respectively. In the figures, we applied the classification accuracy on adversarial examples to evaluate the robustness. The experiment was performed on CIFAR-10 using MobileNetV2 and ResNet-50, where the attack step size was $2/255$, ϵ was $8/255$, $K = 5$, and the training learning rate was 0.01.

The figures demonstrate that, benefit from it fast convergence, LinBP can be used as an alternative to BP when performing adversarial training for achieving robustness (if used appropriately).



(a) MobileNetV2 (b) ResNet-50

Fig. 11. Compare the PGD robustness of MobileNetV2 and ResNet-50 shielded with different adversarial training methods. We use "LinBP†", "LinBP*", and "LinBP†*" to indicate models trained using the first strategy, the second strategy, and the combination of the two strategies as described in the above paragraph, respectively.



(a) MobileNetV2 (b) ResNet-50

Fig. 12. Compare the AutoAttack robustness of MobileNetV2 and ResNet-50 shielded with different adversarial training methods.

5 Conclusion

In this paper, we have studied the convergence of optimization using LinBP, which skips ReLUs during the backward pass, and compared it to optimization using the standard BP. Theoretical analyses have been carefully performed in two popular application scenarios, *i.e.*, white-box attack and model training. In addition to the benefits in black-box attacks which has been demonstrated in [1], we have shown in this paper that LinBP also leads to generating more destructive white-box adversarial examples and obtaining faster model training. Experimental results using synthesized data confirm our theoretical results. Extensive experiments on MNIST and CIFAR-10 further show that the theoretical results hold in more practical settings on a variety of DNN architectures.

Acknowledgments

This work is funded by the Natural Science Foundation of China (NSFC. No. 62176132) and the Guoqiang Institute of Tsinghua University, with Grant No. 2020GQG0005.

References

[1] Y. Guo, Q. Li, and H. Chen, "Backpropagating linearly improves transferability of adversarial examples," in *NeurIPS*, 2020.
[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.
[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017.
[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.
[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.
[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[8] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019.
[9] ——, "Sgdr: Stochastic gradient descent with warm restarts," in *ICLR*, 2017.
[10] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.
[11] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *ICML*, 2013.
[12] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.
[13] Y. LeCun, "A theoretical framework for back-propagation," in *Proceedings of the 1988 connectionist models summer school*, vol. 1, 1988, pp. 21–28.
[14] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *ACM on Asia conference on computer and communications security*, 2017.
[15] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018.
[16] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.
[17] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.
[18] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *ICLR*, 2017.
[19] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.

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

[21] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *CVPR*, 2016.

[22] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE symposium on security and privacy (SP)*, 2017.

[23] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, 2015.

[25] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[26] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *ICML*, 2019.

[27] S. Arora, S. Du, W. Hu, Z. Li, and R. Wang, "Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks," in *ICML*, 2019.

[28] Y. Tian, "An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis," in *ICML*, 2017.

[29] S. S. Du, X. Zhai, B. Poczos, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks," in *ICLR*, 2019.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019.

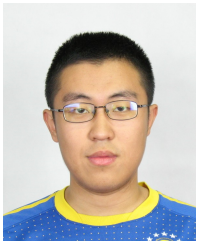
[31] J. Zhang, J. Zhu, G. Niu, B. Han, M. Sugiyama, and M. Kankanhalli, "Geometry-aware instance-reweighted adversarial training," in *ICLR*, 2021.

[32] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein, "Robustbench: a standardized adversarial robustness benchmark," *arXiv preprint arXiv:2010.09670*, 2020.

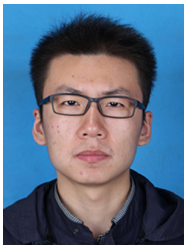
[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

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

[35] Z. Li and S. Arora, "An exponential learning rate schedule for deep learning," *arXiv preprint arXiv:1910.07454*, 2019.



Ziang Li received the B.E. degree from Tsinghua University, Beijing, China, in 2019. He is currently working toward the PhD degree with the Department of Automation, Tsinghua University, Beijing, China. His current research interests include pattern recognition and machine learning.



Yiwen Guo received the B.E. degree from Wuhan University in 2011, and the Ph.D. degree from Tsinghua University in 2016. He is a research scientist at ByteDance AI Lab, Beijing. Prior to this, he was a staff research scientist at Intel Labs China. His current research interests include computer vision, pattern recognition, and machine learning.

Haodi Liu received the B.A. degree from Bard College and B.S. degree from Columbia University in 2018(Dual degree program). And he received the M.S. degree from Columbia University in 2020. Now he is currently working toward the Ph.D. degree with the Department of Automation of Tsinghua University. Prior to this, he was an algorithm engineer at AiBee.Inc. His current research interests include pattern recognition, machine learning and computer vision.

Changshui Zhang (M'02-SM'15-F'18) received the B.S. degree in mathematics from Peking University, Beijing, China, in 1986, and the M.S. and Ph.D. degrees in control science and engineering from Tsinghua University, Beijing, in 1989 and 1992, respectively. In 1992, he joined the Department of Automation, Tsinghua University, where he is currently a professor. His current research interests include pattern recognition and machine learning. He has authored more than 200 articles. He is a fellow of the IEEE.

Appendices: A Theoretical View of Linear Backpropagation and Its Convergence

Ziang Li*, Yiwen Guo*, Haodi Liu, and Changshui Zhang, *Fellow, IEEE*

APPENDIX A
PROOF OF LEMMA 1

We recall the contents of Lemma 1. Assume $D(\mathbf{W}, \mathbf{x}) := \mathrm{diag}(\mathbf{W}\mathbf{x} > 0)$.

Lemma 1. *Denote $G(\mathbf{e}, \mathbf{x}) := \mathbf{W}^T D(\mathbf{W}, \mathbf{e}) \mathbf{V}^T \mathbf{V} D(\mathbf{W}, \mathbf{x}) \mathbf{W}\mathbf{x}$, where $\mathbf{e} \in \mathbb{R}^{d_1}$ is a unit vector, $\mathbf{x} \in \mathbb{R}^{d_1}$ is the input data vector, $\mathbf{W} \in \mathbb{R}^{d_2 \times d_1}$ and $\mathbf{V} \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices. If $\mathbf{W}$ and $\mathbf{V}$ are independent and both generated from the standard Gaussian distribution, we have*

$$\mathbb{E}\left(G(\mathbf{e}, \mathbf{x})\right) = \frac{d_2 d_3}{2\pi}[(\pi - \Theta)\mathbf{x} + \|\mathbf{x}\| \sin\Theta \mathbf{e}],$$

where $\Theta \in [0, \pi]$ is the angle between $\mathbf{e}$ and $\mathbf{x}$.

Proof. Assume $\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_{d_2}]^T$, where $\mathbf{w}_i \in \mathbb{R}^{d_1}$ for $i = 1, \cdots, d_2$. For $G(\mathbf{e}, \mathbf{x})$, we have

$$G(\mathbf{e}, \mathbf{x}) = \sum_{i:\mathbf{w}_i^T\mathbf{x}\geq 0, \mathbf{w}_i^T\mathbf{e}\geq 0} \sum_{j:\mathbf{w}_j^T\mathbf{x}\geq 0, \mathbf{w}_j^T\mathbf{e}\geq 0} \sum_{d=1}^{d_3} v_{di} v_{dj} \mathbf{w}_i \mathbf{w}_j^T \mathbf{x}. \tag{1}$$

We consider the expectation of $G(\mathbf{e}, \mathbf{x})$, there is

$$\mathbb{E}(G(\mathbf{e}, \mathbf{x})) = \sum_{i=1}^{d_2} \sum_{j=1}^{d_2} \mathbb{E}(\sum_{d=1}^{d_3} v_{di} v_{dj}) \mathbb{E}(\mathbf{w}_i \mathbf{w}_j^T \cdot \mathbb{I}_{(\mathbf{w}_i^T\mathbf{x}\geq 0, \mathbf{w}_i^T\mathbf{e}\geq 0, \mathbf{w}_j^T\mathbf{x}\geq 0, \mathbf{w}_j^T\mathbf{e}\geq 0)}(\mathbf{x}, \mathbf{e})\mathbf{x}), \tag{2}$$

where $\mathbb{I}_A(x)$ is the indicator function, *i.e.*, $\mathbb{I}_A(x)$ equals 1 if $x \in A$ and equals 0 if $x \notin A$. As the assumption that $\mathbf{W}$ and $\mathbf{V}$ are independent and both generated from the standard Gaussian distribution, we find $\mathbb{E}(\sum_{d=1}^{d_3} v_{di} v_{dj}) = 0$ when $i \neq j$ and $\mathbb{E}(\sum_{d=1}^{d_3} v_{di} v_{dj}) = d_3$ when $i = j$. Therefore, Eq. (2) can be simplified as

$$\mathbb{E}(G(\mathbf{e}, \mathbf{x})) = \sum_{i=1}^{d_2} \mathbb{E}(\mathbf{w}_i \mathbf{w}_i^T \cdot \mathbb{I}_{(\mathbf{w}_i^T\mathbf{x}\geq 0, \mathbf{w}_i^T\mathbf{e}\geq 0)}(\mathbf{x}, \mathbf{e})\mathbf{x}). \tag{3}$$

We then introduce a coordinate system, where $\mathbf{e} = [1, 0, \cdots, 0]^T$, $\mathbf{x} = \|\mathbf{x}\|[\cos\Theta, \sin\Theta, 0, \cdots, 0]^T$ are hold. Thus, $\mathbf{w_i} = [r\cos\phi_i, r\sin\phi_i, w_{i,3}, \cdots, w_{i,d_1}]^T$. We then can rewrite Eq. (3) as

$$\begin{aligned} \mathbb{E}(G(\mathbf{e}, \mathbf{x})) &= d_3 \mathbb{E} \sum_{i:\mathbf{w}_i^T\mathbf{x}\geq 0, \mathbf{w}_i^T\mathbf{e}\geq 0} \mathbf{w}_i \mathbf{w}_i^T \mathbf{x} \\ &= d_3 \mathbb{E} \sum_{i:\phi_i \in [-\pi/2+\Theta, \pi/2]} \mathbf{w}_i \mathbf{w}_i^T \mathbf{x} \end{aligned} \tag{4}$$

We now compute the following equation,

$$\begin{aligned} R(\phi_0) = \mathbb{E}\left[\frac{1}{d_2} \sum_{i:\phi_i \in [0,\phi_0]} \mathbf{w}_i \mathbf{w}_i^T\right] &= \mathbb{E}\left[\mathbf{w}\mathbf{w}^T | \phi \in [0, \phi_0]\right] \mathbb{P}\left[\phi \in [0, \phi_0]\right] \\ &= \int_{-\infty}^{\infty} \cdots \int_0^{\phi_0} \int_0^{\infty} \mathbf{w}\mathbf{w}^T p(r) p(\phi) \prod_{i=3}^{d} p(w_i) r dr d\phi dw_3 \ldots dw_{d_1}, \end{aligned} \tag{5}$$

where $p(r) = e^{-\frac{r^2}{2}}$ and $p(\phi) = \frac{1}{2\pi}$. Since $\mathbf{W}$ follows gaussian distribution, the off-diagonal and diagonal elements except the first 2×2 block are equal to 0 and $\frac{\phi_0}{2\pi}$ respectively. The first 2×2 block can be formulated as

$$\begin{aligned} R(\phi_0)_{[1:2,1:2]} &= \int_0^{\phi_0} \int_0^{\infty} \begin{bmatrix} r\cos\phi \\ r\sin\phi \end{bmatrix} \begin{bmatrix} r\cos\phi & r\sin\phi \end{bmatrix} p(r)p(\phi) r dr d\phi \\ &= \int_0^{\infty} \frac{r^3 e^{-\frac{r^2}{2}}}{2\pi} dr \int_0^{\phi_0} \begin{bmatrix} \cos^2\phi & \cos\phi\sin\phi \\ \cos\phi\sin\phi & \sin^2\phi \end{bmatrix} d\phi. \\ &= \frac{1}{4\pi} \begin{bmatrix} 2+\sin 2\phi_0 & 1-\cos 2\phi_0 \\ 1-\cos 2\phi_0 & 2-\sin 2\phi_0 \end{bmatrix} \end{aligned} \tag{6}$$

Further we have

$$R(\phi_0) = \frac{1}{2\pi}\mathbf{I}_d + \frac{1}{4\pi} \begin{bmatrix} \sin 2\phi_0 & 1-\cos 2\phi_0 & \mathbf{0} \\ 1-\cos 2\phi_0 & -\sin 2\phi_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{7}$$

Then Eq. (4) can be formulated as

$$\begin{aligned} \mathbb{E}(G(\mathbf{e}, \mathbf{x})) &= d_2 d_3 \left(R(\frac{\pi}{2}) - R(-\frac{\pi}{2} + \Theta) \right) x \\ &= d_2 d_3 \frac{\pi - \Theta}{2\pi} \mathbf{I}_d + \frac{d_2 d_3}{4\pi} (\begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} - \begin{bmatrix} \sin(2\Theta - \pi) & 1-\cos(2\Theta-\pi) \\ 1-\cos(2\Theta-\pi) & -\sin(2\Theta-\pi) \end{bmatrix}) \|x\| \begin{bmatrix} \cos\Theta \\ \sin\Theta \end{bmatrix} \\ &= d_2 d_3 \frac{\pi - \Theta}{2\pi} x + \frac{d_2 d_3 \|x\|}{4\pi} \begin{bmatrix} 2\sin\Theta \\ 0 \end{bmatrix} \\ &= \frac{d_2 d_3}{2\pi} [(\pi - \Theta)\mathbf{x} + \|\mathbf{x}\| \sin\Theta \mathbf{e}] \end{aligned} \tag{8}$$

APPENDIX B
LEMMA FOR PROVING THEOREM 1 AND THEOREM 2

We here propose a lemma before we delve deep into Theorem 1 and Theorem 2. The lemma is described as follows,

Lemma 2. *Let α_i define as*

$$\alpha_i = (u \cdot \mathbf{x}_i^\star - v \cdot \mathbf{x}_i) sgn(\mathbf{x}_i^\star - \mathbf{x}_i) = \begin{cases} u \cdot \mathbf{x}_i^\star - v \cdot \mathbf{x}_i, & if \quad \mathbf{x}_i^\star > \mathbf{x}_i, \\ v \cdot \mathbf{x}_i - u \cdot \mathbf{x}_i^\star, & if \quad \mathbf{x}_i^\star < \mathbf{x}_i, \end{cases}$$

where u and v are constants, $\mathbf{x}_i^\star \sim N(\mu_1, \sigma_1^2)$ and $\mathbf{x}_i \sim N(\mu_2, \sigma_2^2)$. The expectation of α_i can be formulated as

$$\mathbb{E}(\alpha_i) = 2\gamma(u \cdot \sigma_1^2 + v \cdot \sigma_2^2) + (u \cdot \mu_1 - v \cdot \mu_2)(2\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star) - 1),$$

where $\gamma = \frac{1}{\sqrt{2\pi(\sigma_1^2 + \sigma_2^2)}} e^{-(\mu_1 - \mu_2)^2 / 2(\sigma_1^2 + \sigma_2^2)} > 0$. Further we have $\mathbb{E}(\alpha_i) > 0$ when $u > 0$, $v > 0$ and $\mu_2 = 0$.

Proof. From the definition of α_i, the expectation of α_i can be formulated as

$$\mathbb{E}(\alpha_i) = \mathbb{E}(u \cdot \mathbf{x}_i^\star - v \cdot \mathbf{x}_i | \mathbf{x}_i^\star > \mathbf{x}_i) \mathbf{P}(\mathbf{x}_i^\star > \mathbf{x}_i) + \mathbb{E}(v \cdot \mathbf{x}_i - u \cdot \mathbf{x}_i^\star | \mathbf{x}_i^\star < \mathbf{x}_i) \mathbf{P}(\mathbf{x}_i^\star < \mathbf{x}_i). \tag{9}$$

Further, we have $\mathbf{x}_i^\star - \mathbf{x}_i \sim N(\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2)$. Therefore, we can conclude that $\mathbf{P}(\mathbf{x}_i^\star > \mathbf{x}_i) = 1 - F(0)$ and $\mathbf{P}(\mathbf{x}_i^\star < \mathbf{x}_i) = F(0)$, where $F(\cdot)$ denotes the cumulative distribution function for $N(\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2)$.

We then solve the conditional cumulative distribution function for $\mathbf{x}_i^\star$ when $\mathbf{x}_i^\star > \mathbf{x}_i$. We define $f_1(\cdot)$, $f_2(\cdot)$ denote the probability density function of $\mathbf{x}_i^\star$ and $\mathbf{x}_i$ and $F_1(\cdot)$, $F_2(\cdot)$ denote the cumulative distribution function of $\mathbf{x}_i^\star$ and $\mathbf{x}_i$. There we have

$$\begin{aligned} G_1(x) &= \mathbf{P}(\mathbf{x}_i^\star \le x | \mathbf{x}_i^\star > \mathbf{x}_i) \\ &= \frac{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star \le x)}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \\ &= \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \int_{-\infty}^{x} \mathbf{P}(y < \mathbf{x}_i^\star \le x) f_2(y) dy \\ &= \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \int_{-\infty}^{x} (F_1(x) - F_1(y)) f_2(y) dy. \end{aligned} \tag{10}$$

Therefore, the probability density function can be formulated as

$$\begin{aligned} g_1(x) &= \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \left[(F_1(x) \int_{-\infty}^{x} f_2(y) dy)' - (\int_{-\infty}^{x} (F_1(y) f_2(y) dy)' \right] \\ &= \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \left[(f_1(x) \int_{-\infty}^{x} f_2(y) dy) + F_1(x) f_2(x) - F_1(x) f_2(x) \right] \\ &= \frac{f_1(x) F_2(x)}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)}. \end{aligned} \tag{11}$$

Similarly, we first solve the conditional cumulative distribution function for $\mathbf{x}_i$ when $\mathbf{x}_i^\star > \mathbf{x}_i$. There we have,

$$\begin{aligned} G_2(x) &= 1 - \mathbf{P}(\mathbf{x}_i > x | \mathbf{x}_i^\star > \mathbf{x}_i) \\ &= 1 - \frac{\mathbf{P}(x < \mathbf{x}_i < \mathbf{x}_i^\star)}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \\ &= 1 - \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \int_x^\infty \mathbf{P}(x < \mathbf{x}_i < y) f_1(y) dy \\ &= 1 - \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \int_x^\infty (F_2(y) - F_2(x)) f_1(y) dy. \end{aligned} \tag{12}$$

The probability density function can be formulated as

$$\begin{aligned} g_2(x) &= \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \left[(F_2(x) \int_x^\infty f_1(y) dy)' - (\int_x^\infty (F_2(y) f_1(y) dy)' \right] \\ &= \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \left[(f_2(x) \int_x^\infty f_1(y) dy) - F_2(x) f_1(x) + F_2(x) f_1(x) \right] \\ &= \frac{f_2(x)(1 - F_1(x))}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)}. \end{aligned} \tag{13}$$

We assume $\phi(\cdot)$ and $\Phi(\cdot)$ represent the probability density function and cumulative distribution function for standard gaussian distribution, respectively. There we have

$$\begin{aligned} \mathbb{E}(\mathbf{x}_i^\star | \mathbf{x}_i^\star > \mathbf{x}_i) &= \int_{-\infty}^\infty x g_1(x) dx \\ &= \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \int_{-\infty}^\infty \frac{x}{\sigma_1} \phi(\frac{x - \mu_1}{\sigma_1}) \Phi(\frac{x - \mu_2}{\sigma_2}) dx \\ &= \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \int_{-\infty}^\infty \frac{x}{\sqrt{2\pi}\sigma_1} e^{-(x-\mu_1)^2/2\sigma_1^2} \Phi(\frac{x - \mu_2}{\sigma_2}) dx \\ &= \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \int_{-\infty}^\infty \left[\frac{x - \mu_1}{\sqrt{2\pi}\sigma_1} + \frac{\mu_1}{\sqrt{2\pi}\sigma_1} \right] e^{-(x-\mu_1)^2/2\sigma_1^2} \Phi(\frac{x - \mu_2}{\sigma_2}) dx \end{aligned} \tag{14}$$

The integral can be devided into two part. The first part can be formulated as

$$\begin{aligned} \int_{-\infty}^\infty \frac{x - \mu_1}{\sqrt{2\pi}\sigma_1} e^{-(x-\mu_1)^2/2\sigma_1^2} \Phi(\frac{x - \mu_2}{\sigma_2}) dx &= \frac{\sigma_1}{\sqrt{2\pi}} \int_{-\infty}^\infty \Phi(\frac{x - \mu_2}{\sigma_2}) \frac{d(-e^{-(x-\mu_1)^2/2\sigma_1^2})}{dx} dx \\ &= \frac{\sigma_1}{\sqrt{2\pi}} \int_{-\infty}^\infty e^{-(x-\mu_1)^2/2\sigma_1^2} \frac{d\Phi(\frac{x-\mu_2}{\sigma_2})}{dx} dx \\ &= \frac{\sigma_1}{2\sigma_2\pi} \int_{-\infty}^\infty e^{-(x-\mu_1)^2/2\sigma_1^2 - (x-\mu_2)^2/2\sigma_2^2} dx \\ &= \frac{\sigma_1}{2\sigma_2\pi} e^{-(\mu_1-\mu_2)^2/2(\sigma_1^2+\sigma_2^2)} \frac{\sigma_1\sigma_2\sqrt{2\pi}}{\sqrt{\sigma_1^2+\sigma_2^2}} \\ &= \frac{\sigma_1^2}{\sqrt{2\pi(\sigma_1^2+\sigma_2^2)}} e^{-(\mu_1-\mu_2)^2/2(\sigma_1^2+\sigma_2^2)} \end{aligned} \tag{15}$$

Step 3 equals Step 4 because there is $\int_{-\infty}^\infty e^{-(ax^2+bx+c)} dx = e^{(b^2-4ac)/4a} \sqrt{\frac{\pi}{a}}$. The second part can be formulated as

$$\int_{-\infty}^\infty \frac{\mu_1}{\sqrt{2\pi}\sigma_1} e^{-(x-\mu_1)^2/2\sigma_1^2} \Phi(\frac{x - \mu_2}{\sigma_2}) dx = \mu_1 \int_{-\infty}^\infty f_1(x) F_2(x) dx \tag{16}$$

From Eq. (11) we have

$$\int_{-\infty}^\infty g_1(x) dx = \frac{\int_{-\infty}^\infty f_1(x) F_2(x) dx}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} = 1 \tag{17}$$

Therefore we have

$$\int_{-\infty}^\infty \frac{\mu_1}{\sqrt{2\pi}\sigma_1} e^{-(x-\mu_1)^2/2\sigma_1^2} \Phi(\frac{x - \mu_2}{\sigma_2}) dx = \mu_1 \int_{-\infty}^\infty f_1(x) F_2(x) dx = \mu_1 \mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star) \tag{18}$$

As the result in Eq. (15) and Eq. (18), Eq. (14) can be formulated as

$$\begin{aligned} \mathbb{E}(\mathbf{x}_i^\star | \mathbf{x}_i^\star > \mathbf{x}_i) &= \frac{\frac{\sigma_1^2}{\sqrt{2\pi(\sigma_1^2+\sigma_2^2)}} e^{-(\mu_1-\mu_2)^2/2(\sigma_1^2+\sigma_2^2)}}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} + \mu_1 \\ &= \frac{\gamma\sigma_1^2}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} + \mu_1, \end{aligned} \tag{19}$$

where $\gamma = \frac{1}{\sqrt{2\pi(\sigma_1^2+\sigma_2^2)}} e^{-(\mu_1-\mu_2)^2/2(\sigma_1^2+\sigma_2^2)} > 0$.

Similarly, we have

$$\begin{aligned}
\mathbb{E}(\mathbf{x}_i|\mathbf{x}_i^\star > \mathbf{x}_i) &= \int_{-\infty}^{\infty} x g_2(x) dx \\
&= \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \int_{-\infty}^{\infty} \frac{x}{\sigma_2} \phi(\frac{x-\mu_2}{\sigma_2})(1-\Phi(\frac{x-\mu_1}{\sigma_1}))dx \\
&= \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \int_{-\infty}^{\infty} \frac{x}{\sqrt{2\pi}\sigma_2} e^{-(x-\mu_2)^2/2\sigma_2^2}(1-\Phi(\frac{x-\mu_1}{\sigma_1}))dx \\
&= \frac{1}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} \int_{-\infty}^{\infty} \left[\frac{x-\mu_2}{\sqrt{2\pi}\sigma_2} + \frac{\mu_2}{\sqrt{2\pi}\sigma_2}\right] e^{-(x-\mu_2)^2/2\sigma_2^2}(1-\Phi(\frac{x-\mu_1}{\sigma_1}))dx
\end{aligned} \tag{20}$$

The integral can also be devided into two part. The first part can be formulated as

$$\begin{aligned}
&\int_{-\infty}^{\infty} \frac{x-\mu_2}{\sqrt{2\pi}\sigma_2} e^{-(x-\mu_2)^2/2\sigma_2^2}(1-\Phi(\frac{x-\mu_1}{\sigma_1}))dx \\
&= \frac{\sigma_2}{\sqrt{2\pi}} \int_{-\infty}^{\infty} [1-\Phi(\frac{x-\mu_1}{\sigma_1})] \frac{d(-e^{-(x-\mu_2)^2/2\sigma_2^2})}{dx} dx \\
&= \frac{\sigma_2}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-(x-\mu_2)^2/2\sigma_2^2} \frac{d(1-\Phi(\frac{x-\mu_1}{\sigma_1}))}{dx} dx \\
&= \frac{-\sigma_2}{2\sigma_1\pi} \int_{-\infty}^{\infty} e^{-(x-\mu_1)^2/2\sigma_1^2 - (x-\mu_2)^2/2\sigma_2^2} dx \\
&= \frac{-\sigma_2}{2\sigma_1\pi} e^{-(\mu_1-\mu_2)^2/2(\sigma_1^2+\sigma_2^2)} \frac{\sigma_1\sigma_2\sqrt{2\pi}}{\sqrt{\sigma_1^2+\sigma_2^2}} \\
&= \frac{-\sigma_2^2}{\sqrt{2\pi(\sigma_1^2+\sigma_2^2)}} e^{-(\mu_1-\mu_2)^2/2(\sigma_1^2+\sigma_2^2)}
\end{aligned} \tag{21}$$

The second part can be formulated as

$$\int_{-\infty}^{\infty} \frac{\mu_2}{\sqrt{2\pi}\sigma_2} e^{-(x-\mu_2)^2/2\sigma_2^2}(1-\Phi(\frac{x-\mu_1}{\sigma_1}))dx = \mu_2 \int_{-\infty}^{\infty} f_2(x)(1-F_1(x))dx \tag{22}$$

From Eq. (13) we have

$$\int_{-\infty}^{\infty} g_2(x)dx = \frac{\int_{-\infty}^{\infty} f_2(x)(1-F_1(x))dx}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} = 1 \tag{23}$$

Therefore we have

$$\begin{aligned}
\int_{-\infty}^{\infty} \frac{\mu_2}{\sqrt{2\pi}\sigma_2} e^{-(x-\mu_2)^2/2\sigma_2^2}(1-\Phi(\frac{x-\mu_1}{\sigma_1}))dx &= \mu_2 \int_{-\infty}^{\infty} f_2(x)(1-F_1(x))dx \\
&= \mu_2 \mathbf{P}(w_i < w_i^\star)
\end{aligned} \tag{24}$$

Therefore, Eq. (20) can be formulated as

$$\begin{aligned}
\mathbb{E}(\mathbf{x}_i|\mathbf{x}_i^\star > \mathbf{x}_i) &= \frac{\frac{-\sigma_2^2}{\sqrt{2\pi(\sigma_1^2+\sigma_2^2)}} e^{-(\mu_1-\mu_2)^2/2(\sigma_1^2+\sigma_2^2)}}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} + \mu_2 \\
&= \frac{-\gamma\sigma_2^2}{\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star)} + \mu_2.
\end{aligned} \tag{25}$$

Due to symmetry, we also have

$$\mathbb{E}(\mathbf{x}_i|\mathbf{x}_i^\star < \mathbf{x}_i) = \frac{\gamma\sigma_2^2}{\mathbf{P}(\mathbf{x}_i > \mathbf{x}_i^\star)} + \mu_2,$$

and

$$\mathbb{E}(\mathbf{x}_i^\star|\mathbf{x}_i^\star < \mathbf{x}_i) = \frac{-\gamma\sigma_1^2}{\mathbf{P}(\mathbf{x}_i > \mathbf{x}_i^\star)} + \mu_1.$$

Therefore, Eq. (9) can be formulated as

$$\begin{aligned}
\mathbb{E}(\alpha_i) =& \gamma(u\cdot\sigma_1^2 + v\cdot\sigma_2^2) + (u\cdot\mu_1 - v\cdot\mu_2)\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star) \\
&+ \gamma(v\cdot\sigma_2^2 + u\cdot\sigma_1^2) + (v\cdot\mu_2 - u\cdot\mu_1)\mathbf{P}(\mathbf{x}_i > \mathbf{x}_i^\star) \\
=& 2\gamma(u\cdot\sigma_1^2 + v\cdot\sigma_2^2) + (u\cdot\mu_1 - v\cdot\mu_2)(2\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star) - 1),
\end{aligned} \tag{26}$$

where $\gamma = \frac{1}{\sqrt{2\pi(\sigma_1^2+\sigma_2^2)}} e^{-(\mu_1-\mu_2)^2/2(\sigma_1^2+\sigma_2^2)} > 0$. When $\mu_2 = 0$, we have

$$\mathbb{E}(\alpha_i) = 2\gamma(u \cdot \sigma_1^2 + v \cdot \sigma_2^2) + u\mu_1(2\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star) - 1). \tag{27}$$

If $u > 0$ and $v > 0$, we have $\gamma(u \cdot \sigma_1^2 + v \cdot \sigma_2^2) > 0$, and

$$u\mu_1(2\mathbf{P}(\mathbf{x}_i < \mathbf{x}_i^\star) - 1) = u\mu_1(1 - 2F(0)), \tag{28}$$

where F is the cumulative distribution function for $\mathbf{x}_i^\star - \mathbf{x}_i$, and follows the gaussian distribution $N(\mu_1, \sigma_1^2+\sigma_2^2)$. If $\mu_1 > 0$, we have $F(0) < 0.5$, and Eq. (28) is greater than 0. If $\mu_1 < 0$, we have $F(0) > 0.5$, and Eq. (28) is also greater than 0. Therefore, we have $u\mu_1(1 - 2F(0)) \geq 0$. In sum, we have $\mathbb{E}(\alpha_i) > 0$ when $u > 0$, $v > 0$ and $\mu_2 = 0$.

APPENDIX C
PROOF OF THEOREM 1

We first recall the framework of the two-layer network, which can be formulated as

$$g(\mathbf{W}, \mathbf{V}, \mathbf{x}) = \mathbf{V}\sigma(\mathbf{W}\mathbf{x}), \tag{29}$$

where $\mathbf{x} \in \mathbb{R}^{d_1}$ is the input data vector, $\mathbf{W} \in \mathbb{R}^{d_2 \times d_1}$ and $\mathbf{V} \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices, $\sigma(\cdot)$ is the ReLU function. In the teacher-student frameworks, we assume the teacher network possesses the optimal adversarial example $\mathbf{x}^\star$, and the student network learns the adversarial example $\mathbf{x}$ from the teacher network, in which the loss function is set as

$$\mathcal{L}(\mathbf{x}) = \frac{1}{2}\|g(\mathbf{W}, \mathbf{V}, \mathbf{x}) - g(\mathbf{W}, \mathbf{V}, \mathbf{x}^\star)\|_2^2. \tag{30}$$

The update rules is formulated as

$$\mathbf{x}^{(t+1)} = \text{Clip}(\mathbf{x}^{(t)} - \eta\nabla_{\mathbf{x}^{(t)}}\mathcal{L}(\mathbf{x}^{(t)})), \tag{31}$$

where $\text{Clip}(\cdot) = \min(\mathbf{x} + \epsilon\mathbf{1}, \max(\mathbf{x} - \epsilon\mathbf{1}, \cdot))$ and we use the update of standard BP and LinBP, *i.e.*, $\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x})$ and $\tilde{\nabla}_{\mathbf{x}}\mathcal{L}(\mathbf{x})$, to obtain $\{\mathbf{x}^{(t)}\}$ and $\{\tilde{\mathbf{x}}^{(t)}\}$, respectively. The expectation of the gradient obtained by BP and LinBP can be computed from Lemma 1, *i.e.*,

$$\mathbb{E}[\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x})] = G(\mathbf{x}/\|\mathbf{x}\|, \mathbf{x}) - G(\mathbf{x}/\|\mathbf{x}\|, \mathbf{x}^\star) = \frac{d_2 d_3}{2}(\mathbf{x} - \mathbf{x}^\star) + \frac{d_2 d_3}{2\pi}\left(\Theta\mathbf{x}^\star - \frac{\|\mathbf{x}^\star\|}{\|\mathbf{x}\|}\sin\Theta\mathbf{x}\right), \tag{32}$$

and

$$\mathbb{E}[\tilde{\nabla}_{\mathbf{x}}\mathcal{L}(\mathbf{x})] = G(\mathbf{x}/\|\mathbf{x}\|, \mathbf{x}) - G(\mathbf{x}^\star/\|\mathbf{x}^\star\|, \mathbf{x}^\star) = \frac{d_2 d_3}{2}(\mathbf{x} - \mathbf{x}^\star), \tag{33}$$

respectively. Note $\Theta \in [0, \pi]$ is the angle between $\mathbf{x}$ and $\mathbf{x}^\star$. Then we begin our proof.

Theorem 1. *For the two-layer teacher-student network formulated as Eq. (29), the adversarial attack sets Eq. (30) and Eq. (31) as the loss function and the update rule, respectively. Assume that $\mathbf{W}$ and $\mathbf{V}$ follow are independent and both generated from the standard Gaussian distribution, $\mathbf{x}^\star \sim N(\mu_1, \sigma_1^2)$, $\mathbf{x}^{(0)} \sim N(0, \sigma_2^2)$, and η is reasonably small [1]. Let $\mathbf{x}^{(t)}$ and $\tilde{\mathbf{x}}^{(t)}$ be the adversarial examples generated in the t-th iteration of attack using BP and LinBP, respectively, then we have*

$$\mathbb{E}\|\mathbf{x}^\star - \tilde{\mathbf{x}}^{(t)}\|_1 \leq \mathbb{E}\|\mathbf{x}^\star - \mathbf{x}^{(t)}\|_1.$$

Proof. The update rules for BP and LinBP are formulated as

$$\mathbf{x}^{(t+1)} = \text{Clip}(\mathbf{x}^{(t)} - \eta\nabla_{\mathbf{x}^{(t)}}\mathcal{L}(\mathbf{x}^{(t)})), \tag{34}$$

and

$$\tilde{\mathbf{x}}^{(t+1)} = \text{Clip}(\tilde{\mathbf{x}}^{(t)} - \eta\tilde{\nabla}_{\tilde{\mathbf{x}}}^{(t)}\mathcal{L}(\tilde{\mathbf{x}}^{(t)})), \tag{35}$$

respectively. We first analyse the property of the Eq. (35). Assume $\tilde{\mathbf{x}}^{(v)}$ is the first $\{\tilde{\mathbf{x}}^{(t)}\}$ to satisfy $|\tilde{\mathbf{x}}_i^{(v)} - \mathbf{x}_i^{(0)}| = \epsilon$, which also means $|\tilde{\mathbf{x}}_i^{(v-1)} - \mathbf{x}_i^{(0)}| < \epsilon$. If $\tilde{\mathbf{x}}_i^{(v)} = \mathbf{x}_i^{(0)} + \epsilon$, from Eq. (35), we have

$$\begin{aligned}
\tilde{\mathbf{x}}_i^{(v)} &= \text{Clip}(\tilde{\mathbf{x}}_i^{(v-1)} - \frac{\eta d_2 d_3}{2}(\tilde{\mathbf{x}}_i^{(v-1)} - \tilde{\mathbf{x}}_i^\star)) \\
&= \text{Clip}((1 - \frac{\eta d_2 d_3}{2})\tilde{\mathbf{x}}_i^{(v-1)} + \frac{\eta d_2 d_3}{2}\tilde{\mathbf{x}}_i^\star) \\
&= \mathbf{x}_i^{(0)} + \epsilon.
\end{aligned} \tag{36}$$

1. See Eq. (48) for more details of the constraint.

As $|\tilde{\mathbf{x}}_i^{(v-1)} - \mathbf{x}_i^{(0)}| < \epsilon$, we have $\tilde{\mathbf{x}}_i^{\star} > \mathbf{x}_i^{(0)} + \epsilon$. Therefore, for the $v+1$-th step, we have

$$\begin{aligned}\tilde{\mathbf{x}}_i^{(v+1)} &= \text{Clip}(\tilde{\mathbf{x}}_i^{(v)} - \frac{\eta d_2 d_3}{2}(\tilde{\mathbf{x}}_i^{(v)} - \tilde{\mathbf{x}}_i^{\star})) \\ &= \text{Clip}((1 - \frac{\eta d_2 d_3}{2})\tilde{\mathbf{x}}_i^{(v)} + \frac{\eta d_2 d_3}{2}\tilde{\mathbf{x}}_i^{\star}) \\ &= \text{Clip}((1 - \frac{\eta d_2 d_3}{2})(\mathbf{x}_i^{(0)} + \epsilon) + \frac{\eta d_2 d_3}{2}\tilde{\mathbf{x}}_i^{\star}) \\ &= \mathbf{x}_i^{(0)} + \epsilon.\end{aligned} \tag{37}$$

Note that the final step is established because $(1 - \frac{\eta d_2 d_3}{2})(\mathbf{x}_i^{(0)} + \epsilon) + \frac{\eta d_2 d_3}{2}\tilde{\mathbf{x}}_i^{\star} > \mathbf{x}_i^{(0)} + \epsilon$. Further we have $\tilde{\mathbf{x}}_i^{(t)} = \mathbf{x}_i^{(0)} + \epsilon$, for $\forall t > v$. If $\tilde{\mathbf{x}}_i^{(v)} = \mathbf{x}_i^{(0)} - \epsilon$, from Eq. (35), we have

$$\begin{aligned}\tilde{\mathbf{x}}_i^{(v)} &= \text{Clip}((1 - \frac{\eta d_2 d_3}{2})\tilde{\mathbf{x}}_i^{(v-1)} + \frac{\eta d_2 d_3}{2}\tilde{\mathbf{x}}_i^{\star}) \\ &= \mathbf{x}_i^{(0)} - \epsilon.\end{aligned} \tag{38}$$

As $|\tilde{\mathbf{x}}_i^{(v-1)} - \mathbf{x}_i^{(0)}| < \epsilon$, we have $\tilde{\mathbf{x}}^{\star} < \mathbf{x}^{(0)} - \epsilon$. Therefore, for the $v+1$-th step, we have

$$\begin{aligned}\tilde{\mathbf{x}}_i^{(v+1)} &= \text{Clip}((1 - \frac{\eta d_2 d_3}{2})\tilde{\mathbf{x}}_i^{(v)} + \frac{\eta d_2 d_3}{2}\tilde{\mathbf{x}}_i^{\star}) \\ &= \text{Clip}((1 - \frac{\eta d_2 d_3}{2})(\mathbf{x}_i^{(0)} - \epsilon) + \frac{\eta d_2 d_3}{2}\tilde{\mathbf{x}}_i^{\star}) \\ &= \mathbf{x}_i^{(0)} - \epsilon.\end{aligned} \tag{39}$$

Then we have for $\forall t > v$, $\tilde{\mathbf{x}}_i^{(t)} = \mathbf{x}_i^{(0)} - \epsilon$. In sum, for $\forall t > v$, we have $|\tilde{\mathbf{x}}_i^{(t)} - \mathbf{x}_i^{(0)}| = \epsilon$. Similar conclusion can be for made for Eq. (34), further we can find that if a $\mathbf{x}^{(t)}$ (or $\tilde{\mathbf{x}}^{(t)}$) achieve the bound of Clip, the following steps will keep the bounded value. We then let $\mathbf{p}_j = \theta_j \mathbf{x}^{\star} - \frac{\|\mathbf{x}^{\star}\|}{\|\mathbf{x}^{(j)}\|}\sin\theta_j \mathbf{x}^{(j)}$, the l_1 distance bewteen $\mathbf{x}^{(t)}$ ($\tilde{\mathbf{x}}^{(t)}$) and $\mathbf{x}^{\star}$ can be formulated as

$$\begin{aligned}\mathbb{E}\|\mathbf{x}^{\star} - \mathbf{x}^{(t+1)}\|_1 &= \mathbb{E}\|\mathbf{x}^{\star} - \text{Clip}(\mathbf{x}^{(t)} - \eta\nabla_{\mathbf{x}^{(t)}}\mathcal{L}(\mathbf{x}^{(t)}))\|_1 \\ &= \mathbb{E}\|H\left((1 - \frac{\eta d_2 d_3}{2})(\mathbf{x}^{\star} - \mathbf{x}^{(t)}) + \frac{\eta d_2 d_3}{2\pi}\mathbf{p}_t\right)\|_1 \\ &= \mathbb{E}\|H\left((1 - \frac{\eta d_2 d_3}{2})^{t+1}(\mathbf{x}^{\star} - \mathbf{x}^{(0)}) + \sum_{j=0}^{t}\frac{\eta d_2 d_3}{2\pi}(1 - \frac{\eta d_2 d_3}{2})^{t-j}\mathbf{p}_j\right)\|_1,\end{aligned} \tag{40}$$

and

$$\begin{aligned}\mathbb{E}\|\mathbf{x}^{\star} - \tilde{\mathbf{x}}^{(t+1)}\|_1 &= \mathbb{E}\|\mathbf{x}^{\star} - \text{Clip}(\tilde{\mathbf{x}}^{(t)} - \eta\tilde{\nabla}_{\tilde{\mathbf{x}}}^{(t)}\mathcal{L}(\tilde{\mathbf{x}}^{(t)}))\|_1 \\ &= \mathbb{E}\|H\left((1 - \frac{\eta d_2 d_3}{2})^{t+1}(\mathbf{x}^{\star} - \tilde{\mathbf{x}}^{(0)})\right)\|_1,\end{aligned} \tag{41}$$

where $H(\cdot) = \min(\mathbf{x}^{\star} - \mathbf{x}^{(0)} + \epsilon\mathbf{1}, \max(\mathbf{x}^{\star} - \mathbf{x}^{(0)} - \epsilon\mathbf{1}, \cdot))$. From Eq. (40) and Eq. (41), further we have

$$\begin{aligned}\mathbb{E}\|\mathbf{x}^{\star} - \mathbf{x}^{(t+1)}\|_1 &= \mathbb{E}\|H\left((1 - \frac{\eta d_2 d_3}{2})^{t+1}(\mathbf{x}^{\star} - \mathbf{x}^{(0)}) + \sum_{j=0}^{t}\frac{\eta d_2 d_3}{2\pi}(1 - \frac{\eta d_2 d_3}{2})^{t-j}\mathbf{p}_j\right)\|_1 \\ &= \sum_{i=0}^{d}\mathbb{E}|H\left((1 - \frac{\eta d_2 d_3}{2})^{t+1}(\mathbf{x}_i^{\star} - \mathbf{x}_i^{(0)}) + \sum_{j=0}^{t}\frac{\eta d_2 d_3}{2\pi}(1 - \frac{\eta d_2 d_3}{2})^{t-j}\mathbf{p}_{ji}\right)_i|,\end{aligned} \tag{42}$$

and

$$\begin{aligned}\mathbb{E}\|\mathbf{x}^{\star} - \tilde{\mathbf{x}}^{(t+1)}\|_1 &= \mathbb{E}\|H\left((1 - \frac{\eta d_2 d_3}{2})^{t+1}(\mathbf{x}^{\star} - \tilde{\mathbf{x}}^{(0)})\right)\|_1 \\ &= \sum_{i=0}^{d}\mathbb{E}|H\left((1 - \frac{\eta d_2 d_3}{2})^{t+1}(\mathbf{x}_i^{\star} - \tilde{\mathbf{x}}_i^{(0)})\right)_i|.\end{aligned} \tag{43}$$

Note that $\mathbf{x}^{(0)} = \tilde{\mathbf{x}}^{(0)}$ in the theorem. If $|\mathbf{x}_i^{\star} - \mathbf{x}_i^{(0)}| < \epsilon$, then for $\forall t$, $|\mathbf{x}_i^{\star} - \mathbf{x}_i^{(0)}| < \epsilon$ and $|\mathbf{x}_i^{\star} - \tilde{\mathbf{x}}_i^{(0)}| < \epsilon$, where the Clip function can be removed in this case. Eq. (42) and Eq. (43) can be formulated as

$$\begin{aligned}\mathbb{E}\|\mathbf{x}^{\star} - \mathbf{x}^{(t+1)}\|_1 &= \sum_{i=0}^{d}\mathbb{E}|(1 - \frac{\eta d_2 d_3}{2})^{t+1}(\mathbf{x}_i^{\star} - \mathbf{x}_i^{(0)}) + \sum_{j=0}^{t}\frac{\eta d_2 d_3}{2\pi}(1 - \frac{\eta d_2 d_3}{2})^{t-j}\mathbf{p}_{ji}| \\ &= \sum_{i=0}^{d}\mathbb{E}\left((1 - \frac{\eta d_2 d_3}{2})^{t+1}(\mathbf{x}_i^{\star} - \mathbf{x}_i^{(0)}) + \sum_{j=0}^{t}\frac{\eta d_2 d_3}{2\pi}(1 - \frac{\eta d_2 d_3}{2})^{t-j}\mathbf{p}_{ji}\right)\text{sgn}(\mathbf{x}_i^{\star} - \mathbf{x}_i^{(0)}),\end{aligned} \tag{44}$$

and

$$\mathbb{E}\|\mathbf{x}^\star - \tilde{\mathbf{x}}^{(t+1)}\|_1 = \sum_{i=0}^{d} \mathbb{E}\left((1-\frac{\eta d_2 d_3}{2})^{t+1}(\mathbf{x}_i^\star - \tilde{\mathbf{x}}_i^{(0)})\right)\text{sgn}(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)}). \tag{45}$$

If $|\mathbf{x}_i^\star - \mathbf{x}_i^{(0)}| \geq \epsilon$, the sign of $H(\cdot)_i$ is determined by the sign of $\mathbf{x}_i^\star - \mathbf{x}_i^{(0)}$. Therefore Eq. (42) and Eq. (43) can be formulated as

$$\begin{aligned}\mathbb{E}\|\mathbf{x}^\star - \mathbf{x}^{(t+1)}\|_1 &= \sum_{i=0}^{d}\mathbb{E}|(1-\frac{\eta d_2 d_3}{2})^{t+1}(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)}) + \sum_{j=0}^{t}\frac{\eta d_2 d_3}{2\pi}(1-\frac{\eta d_2 d_3}{2})^{t-j}\mathbf{p}_{ji}| \\ &= \sum_{i=0}^{d}\mathbb{E}H\left((1-\frac{\eta d_2 d_3}{2})^{t+1}(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)}) + \sum_{j=0}^{t}\frac{\eta d_2 d_3}{2\pi}(1-\frac{\eta d_2 d_3}{2})^{t-j}\mathbf{p}_{ji}\right)_i \text{sgn}(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)}),\end{aligned} \tag{46}$$

and

$$\mathbb{E}\|\mathbf{x}^\star - \tilde{\mathbf{x}}^{(t+1)}\|_1 = \sum_{i=0}^{d}\mathbb{E}H\left((1-\frac{\eta N}{2})^{t+1}(\mathbf{x}_i^\star - \tilde{\mathbf{x}}_i^{(0)})\right)_i \text{sgn}(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)}). \tag{47}$$

Recall the assumption that η is sufficiently small, to be exact, it should satisfy the following constraints,

$$|\sum_{j=0}^{m-1}\frac{\eta d_2 d_3}{2\pi}(1-\frac{\eta d_2 d_3}{2})^{m-1-j}\mathbf{p}_{ji}| < |(1-\frac{\eta d_2 d_3}{2})^m(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)})|, \tag{48}$$

for $m = 1, \ldots, t$ and $i = 1, \ldots, d$. Under the constraints, we find the sign of $\mathbf{x}_i^\star - \mathbf{x}_i^{(0)}$ determine the sign of $\left((1-\frac{\eta d_2 d_3}{2})^{t-j}\mathbf{p}_{ji}\right)_i$. Therefore, when we compare Eq. (44) and Eq. (45), Eq. (46) and Eq. (47), we actually compare the sign of $\mathbb{E}(\mathbf{p}_{ji}\text{sgn}(\mathbf{w}_i^\star - \mathbf{w}_i^{(0)})$. We first solve the expectation of $\mathbf{x}^{(t)}$. There is

$$\begin{aligned}\mathbb{E}\mathbf{x}^{(t)} &= \mathbb{E}\left((1-\frac{\eta d_2 d_3}{2})\mathbf{x}^{(t-1)} + \frac{\eta d_2 d_3}{2}\mathbf{x}^\star - \frac{\eta d_2 d_3}{2\pi}\mathbf{p}_t\right) \\ &= \mathbb{E}\left((1-\frac{\eta d_2 d_3}{2})^t\mathbf{x}^{(0)} + (1-(1-\frac{\eta d_2 d_3}{2})^t)\mathbf{x}^\star - \frac{\eta d_2 d_3}{2\pi}\sum_{j=0}^{t-1}\left((1-\frac{\eta d_2 d_3}{2})^{(t-1-j)}\mathbf{p}_j\right)\right).\end{aligned} \tag{49}$$

We then solve the following equation,

$$\begin{aligned}\mathbb{E}\left(\mathbf{p}_{si}\text{sgn}(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)})\right) =& \mathbb{E}\left((\theta_s\mathbf{x}_i^\star - \frac{\|\mathbf{x}^\star\|}{\|\mathbf{x}^{(s)}\|}\sin\theta_s\mathbf{x}_i^{(s)})\text{sgn}(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)})\right) \\ =& \mathbb{E}\left(\left(\theta_s - \frac{\|\mathbf{x}^\star\|}{\|\mathbf{x}^{(s)}\|}\sin\theta_s(1-(1-\frac{\eta d_2 d_3}{2})^s)\right)\mathbf{x}_i^\star - \frac{\|\mathbf{x}^\star\|}{\|\mathbf{x}^{(s)}\|}\sin\theta_s(1-\frac{\eta d_2 d_3}{2})^s\mathbf{x}_i^{(0)}\right)\text{sgn}(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)}) \\ &+ \mathbb{E}\left(\frac{\eta d_2 d_3}{2\pi}\sum_{j=0}^{s-1}\left((1-\frac{\eta d_2 d_3}{2})^{(s-1-j)}\mathbf{p}_{ji}\right)\right)\text{sgn}(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)}).\end{aligned} \tag{50}$$

From Eq. (48), we have $\theta_s - \frac{\|\mathbf{x}^\star\|}{\|\mathbf{x}^{(s)}\|}\sin\theta_s > 0$. Recall that $\mathbf{x}^\star \sim N(\mu_1, \sigma_1^2)$ and $\mathbf{x}^{(0)} \sim N(0, \sigma_2^2)$, using the Lemma 2, we first have

$$\mathbb{E}\left(\mathbf{p}_{0i}\text{sgn}(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)})\right) = \mathbb{E}\left(\theta_j\mathbf{x}^\star - \frac{\|\mathbf{x}^\star\|}{\|\mathbf{x}^{(0)}\|}\sin\theta_j\mathbf{x}^{(0)}\right)\text{sgn}(\mathbf{x}_i^\star - \mathbf{x}_i^{(0)}) > 0. \tag{51}$$

Using the mathematical induction, we have

$$\mathbb{E}\left(\mathbf{p}_{ji}\text{sgn}(\mathbf{w}_i^\star - \mathbf{w}_i^{(0)})\right) > 0. \tag{52}$$

With Eq. (52), we find Eq. (44) and Eq. (46) are greater than Eq. (45) and Eq. (47), respectively. That is to say, we have

$$\mathbb{E}\|\mathbf{x}^\star - \tilde{\mathbf{x}}^{(t+1)}\|_1 \leq \mathbb{E}\|\mathbf{x}^\star - \mathbf{x}^{(t+1)}\|_1.$$

APPENDIX D
PROOF OF THEOREM 2

The one-layer network is formulated as

$$h(\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{x}^T\mathbf{w}), \tag{53}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the input vector, $\mathbf{w} \in \mathbb{R}^d$ is the weight vector, and $\sigma(\cdot)$ is the ReLU function. Therefore, $h(\mathbf{X}, \mathbf{w}) = \sigma(\mathbf{X}\mathbf{w})$ where $\mathbf{X} = [\mathbf{x}_1^T; ...; \mathbf{x}_N^T]$ is the input data matrix and $\mathbf{x}_k \in \mathbb{R}^d$ is the k-th training instance, for $k = 1, ..., N$. We assume the teacher network has the optimal weight, and the loss function can be formulated as

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\|h(\mathbf{X}, \mathbf{w}) - h(\mathbf{X}, \mathbf{w}^\star)\|_2^2, \tag{54}$$

where $\mathbf{w}$ and $\mathbf{w}^\star$ are the weight vector for the student network and teacher network, respectively. While training the simple one-layer network with SGD, The update rule for training one-layer network with SGD can be formulated as

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}^{(\mathbf{t})}} \mathcal{L}(\mathbf{w}^{(\mathbf{t})}), \tag{55}$$

where we use the gradients of standard BP and LinBP, *i.e.*, $\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w})$ and $\tilde{\nabla}_{\mathbf{w}}\mathcal{L}(\mathbf{w})$, to obtain $\{\mathbf{w}^{(t)}\}$ and $\{\tilde{\mathbf{w}}^{(t)}\}$, respectively.

Theorem 2. *For the one-layer teacher-student network formulated as Eq. (53), the training task sets Eq. (54) and Eq. (55) as the loss function and the update rule, respectively. Assume that $\mathbf{X}$ is generated from standard Gaussian distribution, $\mathbf{w}^\star \sim N(\mu_1, \sigma_1^2)$, $\mathbf{w}^{(0)} \sim N(0, \sigma_2^2)$, and η is reasonably small[2]. Let $\mathbf{w}^{(t)}$ and $\tilde{\mathbf{w}}^{(t)}$ be the weight vectors obtained in the t-th iteration of training using standard BP and LinBP respectively. Then we have*

$$\mathbb{E}\|\mathbf{w}^\star - \tilde{\mathbf{w}}^{(t)}\|_1 \leq \mathbb{E}\|\mathbf{w}^\star - \mathbf{w}^{(t)}\|_1.$$

Proof. We first recall the partial gradient to $\mathbf{W}$ for BP and LinBP, which are formulated as

$$\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = \mathbf{X}^T\mathbf{D}(\mathbf{X},\mathbf{w})(\mathbf{D}(\mathbf{X},\mathbf{w})\mathbf{X}\mathbf{w} - \mathbf{D}(\mathbf{X},\mathbf{w}^\star)\mathbf{X}\mathbf{w}^\star), \tag{56}$$

and

$$\tilde{\nabla}_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = \mathbf{X}^T(\mathbf{D}(\mathbf{X},\mathbf{w})\mathbf{X}\mathbf{w} - \mathbf{D}(\mathbf{X},\mathbf{w}^\star)\mathbf{X}\mathbf{w}^\star), \tag{57}$$

respectively. From Theorem 1 in [1], we can calculate the expectation of Eq. (56) and Eq. (57) as

$$\mathbb{E}[\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w})] = \frac{N}{2}(\mathbf{w} - \mathbf{w}^\star) + \frac{N}{2\pi}\left(\theta\mathbf{w}^\star - \frac{\|\mathbf{w}^\star\|}{\|\mathbf{w}\|}\sin\theta\mathbf{w}\right), \tag{58}$$

and

$$\mathbb{E}[\tilde{\nabla}_{\mathbf{w}}\mathcal{L}(\mathbf{w})] = \frac{N}{2}(\mathbf{w} - \mathbf{w}^\star), \tag{59}$$

where $\theta \in [0, \pi]$ is the angle between $\mathbf{w}$ and $\mathbf{w}^\star$. The expectation of l_1 distance between $\mathbf{w}^{(t+1)}$ ($\tilde{\mathbf{w}}^{(t+1)}$) and $\mathbf{w}^\star$ can be formulated as

$$\begin{aligned}\mathbb{E}\|\mathbf{w}^\star - \mathbf{w}^{(t+1)}\|_1 &= \mathbb{E}\|\mathbf{w}^\star - \mathbf{w}^{(t)} + \eta\nabla_{\mathbf{w}^{(t)}}\mathcal{L}(\mathbf{w}^{(t)})\|_1 \\ &= \mathbb{E}\|(1 - \frac{\eta N}{2})(\mathbf{w}^\star - \mathbf{w}^{(t)}) + \frac{\eta N}{2\pi}\left(\theta\mathbf{w}^\star - \frac{\|\mathbf{w}^\star\|}{\|\mathbf{w}^{(t)}\|}\sin\theta\mathbf{w}^{(t)}\right)\|_1,\end{aligned} \tag{60}$$

and

$$\begin{aligned}\mathbb{E}\|\mathbf{w}^\star - \tilde{\mathbf{w}}^{(t+1)}\|_1 &= \mathbb{E}\|\mathbf{w}^\star - \tilde{\mathbf{w}}^{(t)} + \eta\tilde{\nabla}_{\mathbf{w}^{(t)}}\mathcal{L}(\mathbf{w}^{(t)})\|_1 \\ &= \mathbb{E}\|(1 - \frac{\eta N}{2})(\mathbf{w}^\star - \tilde{\mathbf{w}}^{(t)})\|_1.\end{aligned} \tag{61}$$

We assume $\mathbf{q}_j = \theta_j\mathbf{w}^\star - \frac{\|\mathbf{w}^\star\|}{\|\mathbf{w}^{(j)}\|}\sin\theta_j\mathbf{w}^{(j)}$. Therefore, Eq. (60) can be formulated as

$$\begin{aligned}\mathbb{E}\|\mathbf{w}^\star - \mathbf{w}^{(t+1)}\|_1 &= \mathbb{E}\|(1 - \frac{\eta N}{2})(\mathbf{w}^\star - \mathbf{w}^{(t)}) + \frac{\eta N}{2\pi}\mathbf{q}_t\|_1 \\ &= \mathbb{E}\|(1 - \frac{\eta N}{2})^2(\mathbf{w}^\star - \mathbf{w}^{(t-1)}) + (1 - \frac{\eta N}{2})\frac{\eta N}{2\pi}\mathbf{q}_{t-1} + \frac{\eta N}{2\pi}\mathbf{q}_t\|_1 \\ &= \mathbb{E}\|(1 - \frac{\eta N}{2})^{t+1}(\mathbf{w}^\star - \mathbf{w}^{(0)}) + \sum_{j=0}^{t}\frac{\eta N}{2\pi}(1 - \frac{\eta N}{2})^{t-j}\mathbf{q}_j\|_1.\end{aligned} \tag{62}$$

And Eq. (61) can be formulated as

$$\begin{aligned}\mathbb{E}\|\mathbf{w}^\star - \tilde{\mathbf{w}}^{(t+1)}\|_1 &= \mathbb{E}\|(1 - \frac{\eta N}{2})(\mathbf{w}^\star - \tilde{\mathbf{w}}^{(t)})\|_1 \\ &= \mathbb{E}\|(1 - \frac{\eta N}{2})^{t+1}(\mathbf{w}^\star - \tilde{\mathbf{w}}^{(0)})\|_1.\end{aligned} \tag{63}$$

Note that $\tilde{\mathbf{w}}^{(0)} = \mathbf{w}^{(0)}$. As mentioned in Theorem 1, we assume η is sufficiently small, to be exact, it should satisfy the following constraints,

$$|\sum_{j=0}^{m}\frac{\eta N}{2\pi}(1 - \frac{\eta N}{2})^{m-j}\mathbf{q}_{ji}| < |(1 - \frac{\eta N}{2})^{m+1}(\mathbf{w}_i^\star - \mathbf{w}_i^{(0)})|, \tag{64}$$

2. See Eq. (64) for a precious formulation of the constraint.

for $m = 1, \ldots, t$ and $i = 1, \ldots, d$. Under the constraints, $\mathbf{w}_i^\star - \mathbf{w}_i^{(0)}$ determines the sign of $\mathbf{w}_i^\star - \mathbf{w}_i^{(t+1)}$ in Eq. (62), then Eq. (62) and Eq. (63) can be calculated as

$$\begin{aligned}
&\mathbb{E}\|\mathbf{w}^\star - \mathbf{w}^{(t+1)}\|_1 \\
&= \sum_{i=0}^{d} \mathbb{E}\left((1 - \frac{\eta N}{2})^{t+1} (\mathbf{w}_i^\star - \mathbf{w}_i^{(0)}) + \sum_{j=0}^{t} \frac{\eta N}{2\pi} (1 - \frac{\eta N}{2})^{t-j} \mathbf{q}_{ji} \right) \mathrm{sgn}(\mathbf{w}_i^\star - \mathbf{w}_i^{(0)}) \\
&= \sum_{i=0}^{d} \mathbb{E}\left((1 - \frac{\eta N}{2})^{t+1} |\mathbf{w}_i^\star - \mathbf{w}_i^{(0)}| + \sum_{j=0}^{t} \frac{\eta N}{2\pi} (1 - \frac{\eta N}{2})^{t-j} \mathbf{q}_{ji} \mathrm{sgn}(\mathbf{w}_i^\star - \mathbf{w}_i^{(0)}) \right),
\end{aligned} \tag{65}$$

and

$$\begin{aligned}
\mathbb{E}\|\mathbf{w}^\star - \tilde{\mathbf{w}}^{(t+1)}\|_1 &= \sum_{i=0}^{d} \mathbb{E}\left((1 - \frac{\eta N}{2})^{t+1} (\mathbf{w}_i^\star - \tilde{\mathbf{w}}_i^{(0)}) \mathrm{sgn}(\mathbf{w}_i^\star - \tilde{\mathbf{w}}_i^{(0)}) \right) \\
&= \sum_{i=0}^{d} \mathbb{E}\left((1 - \frac{\eta N}{2})^{t+1} (\mathbf{w}_i^\star - \mathbf{w}_i^{(0)}) \mathrm{sgn}(\mathbf{w}_i^\star - \mathbf{w}_i^{(0)}) \right) \\
&= \sum_{i=0}^{d} \mathbb{E}\left((1 - \frac{\eta N}{2})^{t+1} |\mathbf{w}_i^\star - \mathbf{w}_i^{(0)}| \right).
\end{aligned} \tag{66}$$

Similar to Theorem 2, using Lemma 2, we have

$$\mathbb{E}\left(\mathbf{q}_{ji} \mathrm{sgn}(\mathbf{w}_i^\star - \mathbf{w}_i^{(0)}) \right) > 0. \tag{67}$$

With Eq. (67), we find Eq. (65) is greater than Eq. (66), i.e.,

$$\mathbb{E}\|\mathbf{w}^\star - \tilde{\mathbf{w}}^{(t+1)}\|_1 \leq \mathbb{E}\|\mathbf{w}^\star - \mathbf{w}^{(t+1)}\|_1.$$

REFERENCES

[1] Y. Tian, "An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis," in *ICML*, 2017.