Reconciling Adversarial Robustness with Accuracy via Randomized Weights

Anonymous authors
Paper under double-blind review

Abstract

Recent years have seen a rapid growth of research on building more robust deep neural networks against adversarial examples. Among them, adversarial training has been shown to be one of the most effective approaches. To balance the robustness of adversarial examples and the accuracy of clean examples, a series of works design enhanced adversarial training methods to strike a trade-off between them with *deterministic* model parameters (i.e., weights). Noting that clean and adversarial examples are highly entangled with the network weights, we propose to study such a trade-off from another perspective, by *treating weights as random variables* in order to harvest the insights yielded from statistical learning theory. Inspired by recent advances of information-theoretic generalization error bound, we found that adversarial training over the randomized weight space can potentially narrow the generalization bound of both clean and adversarial data, and improve both adversarial robustness and clean accuracy simultaneously. Building upon such insights, we propose a novel adversarial training method via Taylor expansion in the hypothesis space of the randomized weights. With PGD, CW, and Auto Attacks, an extensive set of experiments demonstrate that our method further enhances adversarial training, boosting both robustness and clean accuracy.

1 Introduction

Recent works have studied the trade-off between adversarial robustness and clean accuracy (Schmidt et al., 2018; Su et al., 2018; Zhang et al., 2019), and this trade-off demonstrably exists in some cases (Tsipras et al., 2019; Raghunathan et al., 2020; Javanmard et al., 2020; Yu et al., 2021). A number of techniques have been adopted to alleviate the loss of clean accuracy when improving robustness, including TRADES (Zhang et al., 2019), data augmentation (Alayrac et al., 2019; Carmon et al., 2019; Hendrycks et al., 2019), early-stopping (Rice et al., 2020; Zhang et al., 2020), instance reweighting (Balaji et al., 2019; Zhang et al., 2021a) and other various adversarial training methods (Wu et al., 2020; Lee et al., 2020; Cui et al., 2021; Jin et al., 2022). Furthermore, we present more related works and discussions in Appendix A.

Adversarial training is believed to be the most effective defense method against adversarial attacks, which is usually formulated as a minimax optimization problem where the network weights are assumed deterministic in each alternating iteration. Given the fact that both clean and adversarial examples are drawn from unknown distributions interact with one another through the network weights, it is probably unnecessary to restrict ourselves within the deterministic models. That is, a randomized model which is optimized in multi directions of a small area may be more robust against new clean/adversarial examples (Fig. 1).

Building upon such intuition, this paper takes a drastically different view to balance robustness and clean accuracy in adversarial training, *by modeling neural network weights as random variables*. Whilst this perspective is not new in statistical learning theory, where the random weights framework has been frequently used in many previous works e.g., generalization analysis (Dziugaite and Roy, 2017; Neyshabur et al., 2017; Xu and Raginsky, 2017), we hope to advance the understanding of the robustness-accuracy trade-off problem in adversarial training by leveraging the rich tool sets in statistical learning. Remarkably, it turns out adversarial training with the optimization over randomized weights can further improve both the adversarial robustness and clean accuracy.

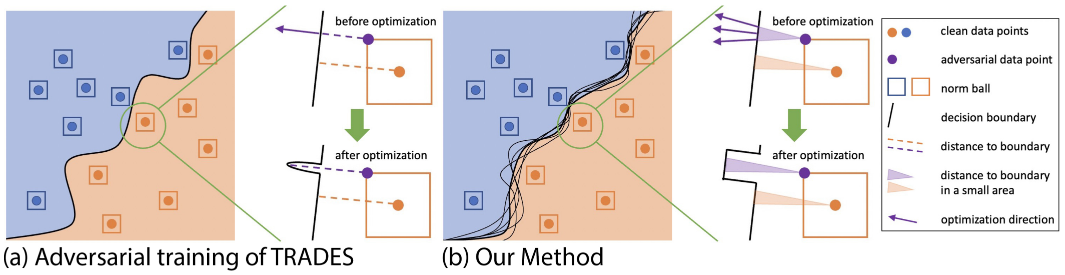

Figure 1: A conceptual illustration of decision boundaries learned via **(a)** adversarial training of TRADES and **(b)** our method. **(a)** shows that TRADES considers a deterministic model and optimizes the distance of adversarial data and boundary only through one direction. Our method in **(b)** takes into account randomized weights (perturbed boundaries) and optimizes the distance of adversarial data and boundary via multi directions in a small area. The boundary learned by our method can be more robust against new data.

By modeling network weights as random variables with an artificially injected weight perturbation, we start with bridging adversarial training to robustness and generalization error bound in statistical learning. In particular, we present such a connection from the information-theoretic perspective (Xu and Raginsky, 2017; Russo and Zou, 2019) in Sec. 3 where we try to answer the question **why randomized weights should be considered in adversarial training**. We show that optimizing over weight hypothesis space is a potential way to tighten the information-theoretic bound of generalization gap over both clean and adversarial data.

Inspired by the above theoretical analyses, we show **how to optimize with randomized weights during adversarial training** in Sec. 4. We propose a novel adversarial training method that reconciles adversarial robustness with clean accuracy, by closing the gap between clean latent space and adversarial latent space over randomized weights. Specifically, we utilize Taylor series to expand the objective function over weights, in such a way that we can deconstruct the function into Taylor series (e.g., zeroth term, first term, second term, etc). From an algorithmic viewpoint, these Taylor terms can thus replace the objective function effectively and time-efficiently. As Fig. 1 shows, our method takes randomized models into consideration during training and makes the learned boundary more robust in a small perturbed area.

In Sec. 5, we **validate the effectiveness of our optimization method** with the first derivative term and the second derivative term of Taylor series. In consideration of complexity and performance, we omit the third and higher derivative terms. Through extensive experiments and comparison with the state-of-the-art on various data sets (CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100, SVHN (Netzer et al., 2011)) and model architectures (ResNet (He et al., 2016), WideResNet (Zagoruyko and Komodakis, 2016), VGG (Simonyan and Zisserman, 2015), MobileNetV2 (Sandler et al., 2018)), we find that our method can further enhance adversarial training consistently with improved adversarial robustness and clean accuracy.

Overall, this paper makes the following contributions. **1.** We inspect the trade-off between adversarial robustness and clean accuracy under a stochastic framework with randomized weights, extending from previous works (Xu and Raginsky, 2017; Russo and Zou, 2019) (Sec. 3), and offer new insights to adversarial training algorithm designs in achieving enhanced trade-off. **2.** Building upon the above insights, we propose a novel adversarial training method under a randomized model. The key enabler is the Taylor series expansion (in Sec. 4) of the robustness loss function over the randomized weight space, so that the optimization can be simultaneously done over the zeroth, first, and second orders of Taylor series. In doing so, the proposed method can effectively enhance adversarial robustness without sacrificing a lot of clean accuracy. **3.** An extensive set of empirical results is provided to demonstrate that our method can improve both robustness and clean accuracy consistently over different datasets and different network architectures (Sec. 5).

2 PRELIMINARIES

Basic Notation. Consider the classification task with the training set $\mathcal{S} = \{\mathbf{s}_1, ..., \mathbf{s}_m\}$ where m samples are drawn from the data distribution $\mathcal{D}$ in the input space $\mathcal{X}$. For notational convenience, we omit the label y of sample $\mathbf{s}$. The adversarial sample $\mathbf{s}'$ is generally not in the natural data distribution $\mathcal{D}$, and we thus let $\mathcal{S}'$ and $\mathcal{D}'$ be the adversarial set and distribution for a specific model, respectively, such that $||\mathbf{s}' - \mathbf{s}|| \le \epsilon$ where $||\cdot||$ is by default the ℓ_2-norm. $\mathcal{L}(f_w(\mathbf{s}), y)$ is the cross-entropy loss between $f_w(\mathbf{s})$ and y with normalization. Define $\mathcal{L}(f_w(\mathcal{S}), \mathcal{Y}) := \mathbb{E}_{\mathbf{s}\in\mathcal{S}}[\mathcal{L}(f_w(\mathbf{s}), y)]$.

From Deterministic to Randomized Weights. Let W be the collection of model parameters and $w \in \mathbb{R}^{d\times 1}$ be the vectorization of W. w includes both weights and bias of all layers, where the latter can be seen as the weight with input 1. To make a clear distinction, let $\mathbf{w}$ be the randomized version of w, random weights, modeled as a multivariate random variable or random vector in hypothesis space $\mathbb{W}$. We consider $\mathbf{w}$ is generated by normal training with clean data set $\mathcal{S}$, $\mathbf{w}+\mathbf{u}$ is generated by adversarial training where the noise $\mathbf{u}$ is brought by adversarial data set $\mathcal{S}'$.

Assumption 1 *For simplicity, we consider an additive randomization model, where $\mathbf{w}$ is a Gaussian, $\mathbf{u}$ is another Gausian and $\mathbf{w}+\mathbf{u}$ is still a Gaussian. Following previous works (Neyshabur et al., 2017; 2018; Jiang et al., 2020; Dziugaite et al., 2020) of randomized weights, let $\mathbf{w}+\mathbf{u}$ be generated by injecting small Gaussian noise into deterministic w, so that $\mathbf{w}+\mathbf{u}$ can be seen as a multivariate Gaussian that scatters around the deterministic vector w in hypothesis space $\mathbb{W}$ (i.e., the probability space around the mean value w), i.e., $\mathbf{w}+\mathbf{u} = w + \mathbf{u}'$, where $\mathbf{u}' \sim \mathcal{N}(0, \sigma^2\mathbf{I}_d)$ is a small zero mean spherical Gaussian which satisfies $\mathcal{L}(f_{w+\mathbf{u}'}(\mathbf{s}), y) \approx \mathcal{L}(f_w(\mathbf{s}), y)$.*

Adversarial Training. Adversarial training can be formulated as a robust optimization problem (Madry et al., 2018)

$$\min_w \left\{ \mathop{\mathbb{E}}_{\mathbf{s}\sim\mathcal{D}} \left[\max_{\mathbf{s}':||\mathbf{s}'-\mathbf{s}||\le\epsilon} \mathcal{L}(f_w(\mathbf{s}'), y) \right] \right\}, \tag{1}$$

where $\mathbf{s}'$ is an adversarial example causing the largest loss within an ϵ-ball centered at a clean example $\mathbf{s}$ with respect to a norm distance, $f_w(\cdot)$ is the neural network function parameterized by w.

3 THEORETICAL PERSPECTIVE

This section explores theoretical implication of the use of randomized weights on robustness. Specifically, we discuss how randomized weights affect the information-theoretic generalization bound of both clean and adversarial data. **We clarify the theoretical motivation of this work and demonstrate how our method of adversarial training in Sec. 4 narrows the generalization bound over both clean and adversarial data.**

Under the information-theoretic context, a learning algorithm can be taken as a randomized mapping, where training data set is input and hypothesis is output. With that, Xu and Raginsky (2017); Russo and Zou (2019) considered a generalization bound based on the information contained in weights $I(\Lambda_{\mathbb{W}}(\mathcal{S}); \mathbf{w})$, where $\Lambda_{\mathbb{W}}(\mathcal{S}) := \big(\mathcal{L}(f_w(\mathcal{S}), \mathcal{Y})\big)_{w\in\mathbb{W}}$ is the collection of empirical losses of the hypotheses in $\mathbb{W}$. Let $\mathbb{P}(\mathcal{L}(f_{w_1}(\mathcal{S}), \mathcal{Y}), w_2) = 0$ where $w_1 \ne w_2$, we can use $I(\mathcal{L}(f_{\mathbf{w}}(\mathcal{S}), \mathcal{Y}); \mathbf{w})$ to approximate $I(\Lambda_{\mathbb{W}}(\mathcal{S}); \mathbf{w})$ where $\mathbf{w}$ is the randomized weight distributed in $\mathbb{W}$, then get the following theory.

Theorem 3.1 *(Russo and Zou, 2019). Suppose $\mathcal{L}(f_{\mathbf{w}}(\mathbf{s}), y)$ is σ_*-sub-Gaussian, $\mathcal{D}$ is the clean data distribution and $\mathcal{S}$ is the training data set with m samples, then*

$$\mathbb{E}_{\mathbf{w}}\Big(\mathcal{L}\big(f_{\mathbf{w}}(\mathcal{D}), \mathcal{Y}\big) - \mathcal{L}\big(f_{\mathbf{w}}(\mathcal{S}), \mathcal{Y}\big)\Big) \le \sqrt{\frac{2\sigma_*^2}{m} I\Big(\mathcal{L}\big(f_{\mathbf{w}}(\mathcal{S}), \mathcal{Y}\big); \mathbf{w}|\mathcal{S}\Big)}. \tag{2}$$

Thm. 3.1 provides the upper bound on the expected generalization error of randomized weights. Building upon the above bound, we consider generalization errors of both clean and adversarial data based on discrete distribution, then obtain the following proposition.

Proposition 3.2 *Let $\mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}), y)$ and $\mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}'), y)$ be σ_*-sub-Gaussian. We suppose the adversarial trained model contains information of $\mathcal{S} \vee \mathcal{S}'$, where $\mathcal{S} \vee \mathcal{S}'$ is the joint set and the training samples are chosen at random from $\mathcal{S}$ and $\mathcal{S}'$ with probabilities q and $1-q$, i.e., $\mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}\vee\mathcal{S}'), \mathcal{Y}) = q\mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}) + (1-q)\mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y})$. We let $q \in (0, 0.5)$, since $\mathcal{S}'$ occupies a large proportion in adversarial training, then*

$$\begin{aligned} &\mathbb{E}_{\mathbf{w}+\mathbf{u}}\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{D}\vee\mathcal{D}'), \mathcal{Y}\big) - \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}\vee\mathcal{S}'), \mathcal{Y}\big)\Big) \\ &\le \sqrt{\frac{2\sigma_*^2}{m} I\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}\vee\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}|\mathcal{S}\vee\mathcal{S}'\Big)} \\ &\le \sqrt{\frac{2\sigma_*^2}{m} I\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big), \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}|\mathcal{S}\vee\mathcal{S}'\Big)}. \end{aligned} \tag{3}$$

Remark 3.2 *We now make several observations about Prop. 3.2. First, it is obvious that more training data (larger m) helps adversarial training to get a high-performance model. Second, take into account both generalization errors of clean and adversarial data with coefficient q, a lower mutual information between $\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big), \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big)\big)$ and $\mathbf{w}+\mathbf{u}$ is essential to get a better performance of robustness and clean accuracy.*

The above mutual information is a statistic over high-dimensional space, thus we are almost impossible to directly estimate and optimize it during training. Nevertheless, we can reduce it implicitly through the following lemmas.

Lemma 3.3 $I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big), \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big)$ *is lower bounded by* $I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big)$ *and upper bounded by* $I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big) + I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big)$*, i.e.,*

$$\begin{aligned} I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big) &\leq I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big), \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big) \\ &\leq I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big) + I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big). \end{aligned} \tag{4}$$

Remark 3.3 *We provide the details of proof in Appendix B. Lem. 3.3 gives us an upper bound and a lower bound. The upper bound represents the worst case of adversarial trained model where $\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big)$ and $\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big)$ are radically different (uncorrelated). To some extent, it means the trained model is failed to extract common features of clean data and adversarial data, thus needs to use more parameters to recognize clean and adversarial examples respectively. Lem. 3.3 also charts a realizable optimization direction for the model, the lower bound $I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big)$ is the optimal case of adversarial training for $I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big), \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big)$. In this situation, $\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big)$ and $\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big)$ are completely correlated, that is, the optimal adversarial trained model is successful at extracting common features of clean data and adversarial data.*

It is obvious that $I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big), \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big)$ is still difficult to be estimated in a high-dimensional space. Fortunately, Lem. 3.3 allows us to optimize it via narrowing the distance between $\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big)$ and $\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big)$, which makes $I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big), \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big)$ close to the lower bound $I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big)$.

Lemma 3.4 *In this lemma, we consider the case of binary response $y \in \{0, 1\}$, then the gap between $\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}), y\big)$ and $\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}'), y\big)$ is positive correlated with the KL divergence between $f_{\mathbf{w}+\mathbf{u}}(\mathbf{s})$ and $f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}')$, i.e.,*

$$|\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}), y\big) - \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}'), y\big)| \propto \mathrm{KL}\big(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}) || f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}')\big), \tag{5}$$

where we let $\propto$ represent positive correlation.

Remark 3.4 *The proof is provided in Appendix C. Lem. 3.4 demonstrates $|\mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}), y) - \mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}'), y)|$ is positive correlated with $\mathrm{KL}(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}) || f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}'))$ in a binary case, this can also approximately hold in a multi-class case. Thus, it allows us to optimize the mutual information utilizing a simplified term of $\mathrm{KL}(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}) || f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}'))$. Although we are still difficult to directly deal with this KL term during training, it can be decomposed by our method in Sec. 4 with Taylor series.*

4 ALGORITHMIC DESIGN

Motivated by TRADES (Zhang et al., 2019) and the discussion of generalization bound of both clean data and adversarial data in Sec. 3, we propose that the robust optimization problem of adversarial training can be further enhanced by

$$\min\ \mathbb{E}_{\mathbf{w}+\mathbf{u}}\Big[\mathbb{E}_{\mathbf{s}}(\mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}), y)) + \mathbb{E}_{\mathbf{s}} \max_{\mathbf{s}':||\mathbf{s}'-\mathbf{s}||\leq\epsilon} \mathrm{KL}(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}) || f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}'))/\lambda\Big], \tag{6}$$

where the first term contributes to clean accuracy, and the second term with hyperparameter λ can be seen as a regularization for adversarial robustness with randomized weights $\mathbf{w}+\mathbf{u}$.

Algorithmically, following Assumption 1 and Zhang et al. (2019), we extend Eq. (6) by replacing $\mathrm{KL}(\cdot||\cdot)$ with a multi-class calibrated loss $\mathcal{L}(\cdot, \cdot)$:

$$\min_{w}\ \mathbb{E}_{\mathbf{u}'}\Big[\mathbb{E}_{\mathbf{s}}(\mathcal{L}(f_w(\mathbf{s}), y)) + \mathbb{E}_{\mathbf{s}} \max_{\mathbf{s}':||\mathbf{s}'-\mathbf{s}||\leq\epsilon} \mathcal{L}(f_{w+\mathbf{u}'}(\mathbf{s}), f_{w+\mathbf{u}'}(\mathbf{s}'))/\lambda\Big]. \tag{7}$$

Algorithm 1 Adversarial training with randomized weights

```
Input: minibatch $\{\mathbf{s}_i\}_{i=1}^n$, network architecture parametrized by $w$, learning rate $\eta_l$, step size $\eta_s$,
  hyper-parameters $\eta$, $\lambda$, zero mean Gaussian $\mathbf{u}'$, number of iterations $K$ in inner optimization.
Output: Robust network $f_w$
Randomly initialize network $f_w$, or initialize network with pre-trained configuration
repeat
  ▶ Generate adversarial examples:
  Sample $u'$ from $\mathbf{u}'$
  for $i = 1$ to $n$ do
    $\mathbf{s}'_i \leftarrow \mathbf{s}_i + 0.001 \cdot \mathcal{N}(\mathbf{0}, \mathbf{I})$
    for $k = 1$ to $K$ do
      $\mathbf{s}'_i \leftarrow \Pi(\eta_s \mathbf{sign}(\nabla_{\mathbf{s}'_i} \mathcal{L}(f_{w+u'}(\mathbf{s}_i), f_{w+u'}(\mathbf{s}'_i))) + \mathbf{s}'_i)$, subject to $||\mathbf{s}_i - \mathbf{s}'_i|| \le \epsilon$,
      where $\Pi$ is the projection operator.
    end for
  end for
  ▶ Optimization: $w \leftarrow w - \eta_l \frac{1}{n} \sum_{i=1}^n \nabla_w \Big[\mathcal{L}(f_w(\mathbf{s}_i), y_i)$
        (zeroth term)    $+\mathcal{L}(g_{\mathbf{s}_i}(w), g_{\mathbf{s}'_i}(w))/\lambda$
        (first term)     $+\eta \mathbb{E}_{\mathbf{u}'} \left(\mathcal{L}(g'_{\mathbf{s}_i}(w)^T \mathbf{u}', g'_{\mathbf{s}'_i}(w)^T \mathbf{u}')\right)$
        (second term)    $+\frac{\eta}{2} \mathbb{E}_{\mathbf{u}'} \left(\mathcal{L}(\mathbf{u}'^T g''_{\mathbf{s}_i}(w) \mathbf{u}', \mathbf{u}'^T g''_{\mathbf{s}'_i}(w) \mathbf{u}')\right)\Big]$,
  where various optimization methods can be applied: zeroth term optimization (TRADES),
  zeroth + first terms optimization and zeroth + first + second terms optimization. The first
  term and second term are optimized through Eqs. (13), (14) respectively.
until training converged
```

As the minimax optimization is entangled with expectation of randomized weights, it is challenging to solve such optimization problem directly. Instead, we propose to solve this problem in an alternating manner between the inner maximization and outer minimization.

For the inner maximization, with a given model weight w, we solve it by adopting the commonly used gradient-based approach (e.g., PGD) to generate the adversarial perturbation. That is, at the t-th iteration, the adversarial perturbation is produced iteratively by letting

$$\mathbf{s}'_{t+1} \leftarrow \Pi\Big(\mathbf{s}'_t + \eta_s \mathbf{sign}\big(\nabla_{\mathbf{s}'_t} \mathcal{L}(f_{w+u'}(\mathbf{s}), f_{w+u'}(\mathbf{s}'_t))\big)\Big) \tag{8}$$

until the maximum allowed iterations are reached, where Π is the projection operator, η_s is the step size and u' is a sample of $\mathbf{u}'$. To reduce complexity, we only use one sample u' to generate adversarial example for each minibatch. Nevertheless, sufficient samples of $\mathbf{u}'$ can still be produced in one epoch as there are many minibatches. As Fig. 1 shows, the adversarial example is generated by one model (boundary), then the optimization with randomized weights in multiple directions can be executed through the following minimization method.

For the outer minimization, when the optimal perturbed sample $\mathbf{s}'$ is generated by the inner maximization, we solve the following problem

$$\min_w \mathbb{E}_{\mathbf{s}} \Big[\mathcal{L}(f_w(\mathbf{s}), y) + \mathbb{E}_{\mathbf{u}'} \mathcal{L}(f_{w+\mathbf{u}'}(\mathbf{s}), f_{w+\mathbf{u}'}(\mathbf{s}'))/\lambda\Big]. \tag{9}$$

This optimization problem is challenging due to the entanglement between the deterministic weight w and the random perturbation $\mathbf{u}'$, which makes gradient updating much involved.

To resolve this issue, we decompose the two components in such a way that gradient updating and model averaging can be done separately. Let $f_{w+\mathbf{u}'}(\mathbf{s}) = g_{\mathbf{s}}(w + \mathbf{u}')$, by Taylor expansion of $g_{\mathbf{s}}(w + \mathbf{u}')$ at w, we have

$$g_{\mathbf{s}}(w + \mathbf{u}') = g_{\mathbf{s}}(w) + g'_{\mathbf{s}}(w)^T \mathbf{u}' + \mathbf{u}'^T \frac{g''_{\mathbf{s}}(w)}{2!} \mathbf{u}' + \mathcal{O}_{\mathbf{s}}(||\mathbf{u}'||^2), \tag{10}$$

where $g'_{\mathbf{s}}(w) = \frac{\partial g_{\mathbf{s}}(w)}{\partial w} \in \mathbb{R}^{d \times 1}$ and $g''_{\mathbf{s}}(w) = \frac{\partial^2 g_{\mathbf{s}}(w)}{\partial w \partial w} \in \mathbb{R}^{d \times d}$ are the first and second derivative of the function $g_{\mathbf{s}}(w)$ versus the weight w, respectively, and $\mathcal{O}_{\mathbf{s}}(||\mathbf{u}'||^2)$ tends to be zero with respect to $||\mathbf{u}'||^2$. By such approximation, the gradient computation will be solely done on the deterministic part w, and model smoothing with averaged random variable $\mathbf{u}'$ is done independently.

The intuition behind is the following. For the randomized model $w + \mathbf{u}'$ with a specific input $\mathbf{s}$, the Taylor approximation explicitly takes into account the deterministic model $g_{\mathbf{s}}(w)$, the projection of its gradient onto the random direction $g'_{\mathbf{s}}(w)^T\mathbf{u}'$, and the projection of its curvature onto the subspace spanned by $\mathbf{u}'^T \frac{g''_{\mathbf{s}}(w)}{2!}\mathbf{u}'$, capturing higher-order information of the loss function with respect to $\mathbf{u}'$.

To further simplify the computation, we minimize an upper bound of Eq. (9) instead. Due to the local convexity of loss landscape, by Jensen's inequality, we conclude that

$$\mathcal{L}(\sum_i x_i, \sum_i y_i) \leq \sum_i \mathcal{L}(x_i, y_i). \tag{11}$$

Therefore, the second term in Eq. (9) can be upper bounded by

$$\begin{aligned}\mathbb{E}_{\mathbf{u}'}\mathcal{L}(f_{w+\mathbf{u}'}(\mathbf{s}), f_{w+\mathbf{u}'}(\mathbf{s}')) \leq \mathcal{L}(g_{\mathbf{s}}(w), g_{\mathbf{s}'}(w)) &+ \mathbb{E}_{\mathbf{u}'}\left(\mathcal{L}(g'_{\mathbf{s}}(w)^T\mathbf{u}', g'_{\mathbf{s}'}(w)^T\mathbf{u}')\right) \\ &+ \frac{1}{2}\mathbb{E}_{\mathbf{u}'}\left(\mathcal{L}(\mathbf{u}'^T g''_{\mathbf{s}}(w)\mathbf{u}', \mathbf{u}'^T g''_{\mathbf{s}'}(w)\mathbf{u}')\right),\end{aligned} \tag{12}$$

where the terms on the right-hand side are referred to as the zeroth, first, and second order Taylor expressions. Instead of minimizing Eq. (9) directly, we minimize the upper bound in Eq. (12), which is easier to compute in practical models. We notice that minimizing $\mathbb{E}_{\mathbf{u}'}(\mathcal{L}(g'_{\mathbf{s}}(w)^T\mathbf{u}', g'_{\mathbf{s}'}(w)^T\mathbf{u}'))$, $\mathbb{E}_{\mathbf{u}'}(\mathcal{L}(\mathbf{u}'^T g''_{\mathbf{s}}(w)\mathbf{u}', \mathbf{u}'^T g''_{\mathbf{s}'}(w)\mathbf{u}'))$ is almost equal to reducing the distance between $g'_{\mathbf{s}}(w)$ and $g'_{\mathbf{s}'}(w)$, $g''_{\mathbf{s}}(w)$ and $g''_{\mathbf{s}'}(w)$, respectively. To further reduce complexity, we apply the first and second terms optimization by reducing the distance of $g'_{\mathbf{s}}(w)$ and $g'_{\mathbf{s}'}(w)$, $g''_{\mathbf{s}}(w)$ and $g''_{\mathbf{s}'}(w)$ only on the final (L-th) layer through minimizing

$$\begin{aligned}&\frac{1}{2}\mathcal{L}\Big(g_{\mathbf{s}'}(w), \Big[\sum_i \frac{\partial(g_{\mathbf{s}}(w) - g_{\mathbf{s}'}(w))}{\partial W_{L(1,i)}}, ..., \sum_i \frac{\partial(g_{\mathbf{s}}(w) - g_{\mathbf{s}'}(w))}{\partial W_{L(N_L,i)}}\Big]^T\Big) \\ &+ \frac{1}{2}\mathcal{L}\Big(g_{\mathbf{s}}(w), \Big[\sum_i \frac{\partial(g_{\mathbf{s}'}(w) - g_{\mathbf{s}}(w))}{\partial W_{L(1,i)}}, ..., \sum_i \frac{\partial(g_{\mathbf{s}'}(w) - g_{\mathbf{s}}(w))}{\partial W_{L(N_L,i)}}\Big]^T\Big),\end{aligned} \tag{13}$$

$$\begin{aligned}&\frac{1}{2}\mathcal{L}\Big(g_{\mathbf{s}'}(w), \Big[\sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}}(w) - \partial^2 g_{\mathbf{s}'}(w)}{\partial W_{L(1,i)}\partial W_{L(j,k)}}, ..., \sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}}(w) - \partial^2 g_{\mathbf{s}'}(w)}{\partial W_{L(N_L,i)}\partial W_{L(j,k)}}\Big]^T\Big) \\ &+ \frac{1}{2}\mathcal{L}\Big(g_{\mathbf{s}}(w), \Big[\sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}'}(w) - \partial^2 g_{\mathbf{s}}(w)}{\partial W_{L(1,i)}\partial W_{L(j,k)}}, ..., \sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}'}(w) - \partial^2 g_{\mathbf{s}}(w)}{\partial W_{L(N_L,i)}\partial W_{L(j,k)}}\Big]^T\Big),\end{aligned} \tag{14}$$

where $W_{L(j,k)}$ is the element of j-th row, k-th column of W_L on L-th (final) layer, N_L is the number of neurons (units) on L-th layer. More discussions about the above replacement optimization are given in Appendix D.

The pseudo-code of the proposed adversarial training method is presented in Algorithm 1. Note that the zeroth term optimization in Algorithm 1 is almost the same as TRADES (Zhang et al., 2019), thus we use TRADES and AWP-TRADES (Wu et al., 2020) as (zeroth term optimization) baselines in our experiments in Sec. 5, and verify how much first and second terms optimization can improve.

For all of our experiments, we limit the noise variance of $\mathbf{u}'$ to be very small, e.g., $\sigma^2 = 0.0001$. For the zeroth term hyper-parameter $1/\lambda$ in Algorithm 1, we set $1/\lambda = 6$ for all of our experiments, which is a widely-used setting for TRADES (Zhang et al., 2019; Wu et al., 2020; Pang et al., 2022). The number of inner optimization iterations K is set to 10 as usual. To make our method more flexible, we set η and $\frac{\eta}{2}$ as the first term and second term hyper-parameters, respectively, and analyze the sensitivity of η in Sec. 5.1.

The main novelty of our proposed method is two-fold: **(1)** Instead of considering a single model with a deterministic weight w, we consider a model ensemble with randomized weights $w + \mathbf{u}'$ that possesses the same generalization ability. By averaging these models during adversarial training, we come up with a robust model with smoothing classification boundary tolerant to potential adversarial weight perturbation that may resulted from adversarial examples. **(2)** When averaging over the ensemble of randomized models during training, we disentangle gradient from random weight perturbation so that the gradient updating can be computed efficiently. In particular, we apply Taylor

Table 1: First and Second Derivative terms optimization on CIFAR-10 with ℓ_∞ threat model for PreAct ResNet18. Classification accuracy (%) on clean images and under PGD-20 attack, CW-20 attack and Auto Attack with different hyper-parameters $\eta = 0.05, 0.1, 0.2, 0.3, 0.4$. We highlight the best results in **bold**.

Method	Clean	PGD-20	CW-20	AA	Method	Clean	PGD-20	CW-20	AA
1_{st} (0.05)	83.19	53.98	52.12	48.2	1_{st}+2_{nd} (0.05)	83.25	54.07	51.93	48.4
1_{st} (0.1)	82.86	54.29	52.24	49.3	1_{st}+2_{nd} (0.1)	83.47	54.42	52.28	49.7
1_{st} (0.2)	83.55	54.86	**52.65**	48.8	1_{st}+2_{nd} (0.2)	84.13	**55.13**	**52.53**	**50.8**
1_{st} (0.3)	**83.96**	**55.05**	52.54	**49.7**	1_{st}+2_{nd} (0.3)	**84.27**	54.36	51.80	48.9
1_{st} (0.4)	83.93	54.65	52.23	48.6	1_{st}+2_{nd} (0.4)	84.14	54.38	51.56	49.6

expansion at the mean weight w (i.e., the deterministic component) and approximate the cross-entropy loss function with the zeroth, first, and second Taylor terms. In doing so, the deterministic and statistical components of $\mathbf{w} + \mathbf{u}$ can be decomposed and then computed independently.

5 Empirical results

In this section, we first discuss the hyper-parameter sensitivity of our method, and then evaluate the robustness on benchmark data sets against various white-box, black-box attacks and Auto Attack with ℓ_2 and ℓ_∞ threat models.

Adversarial Training Setting. We train PreAct ResNet-18 (He et al., 2016) for ℓ_∞ and ℓ_2 threat models on CIFAR-10/100 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011). In addition, we also train WideResNet-34-10 (Zagoruyko and Komodakis, 2016) for CIFAR-10/100, VGG16 and MobileNetV2 for CIFAR-10, with ℓ_∞ threat model. We adopt the widely used adversarial training setting (Rice et al., 2020): for the ℓ_∞ threat model, $\epsilon = 8/255$ and step size $2/255$; for the ℓ_2 threat model, $\epsilon = 128/255$ and step size $15/255$. For normal adversarial training, the training examples are generated with 10 steps. All models (except SVHN) are trained for 200 epochs using SGD with momentum 0.9, batchsize 128, weight decay 5×10^{-4}, and an initial learning rate of 0.1 that is divided by 10 at the 100th and 150th epochs. Except for setting the starting learning rate to 0.01 for SVHN, we utilize the same other settings. Simple data augmentations are used, such as 32×32 random crop with 4-pixel padding and random horizontal flip. We report the highest robustness that ever achieved at different checkpoints for each data set and report the clean accuracy on the model which gets the highest PGD-20 accuracy. We omit the standard deviations of 3 runs as they are very small ($< 0.40\%$) and implement all models on NVIDIA A100.

Evaluation Setting. We evaluate the robustness with wihte-box attacks, black-box attacks and auto attack. For **white-box attacks**, we adopt PGD-20 (Madry et al., 2018) and CW-20 (Carlini and Wagner, 2017) (the ℓ_∞ version of CW loss optimized by PGD-20) to evaluate trained models. For **black-box attacks**, we generate adversarial perturbations by attacking a surrogate normal adversarial training model (with same setting) (Papernot et al., 2017), and then apply these adversarial examples to the defense model and evaluate the performances. The attacking methods for black-box attacks we have used are PGD-20 and CW-20. For **Auto Attack (AA)** (Croce and Hein, 2020b), one of the strongest attack methods, we adopt it through a mixture of different parameter-free attacks which include three white-box attacks (APGD-CE (Croce and Hein, 2020b), APGD-DLR (Croce and Hein, 2020b), and FAB (Croce and Hein, 2020a)) and one black-box attack (Square Attack (Andriushchenko et al., 2020)). We provide more details about the experimental setups in Appendix E.

5.1 Sensitivity of hyper-parameter

In our proposed algorithm, the regularization hyper-parameter η is crucial. We use numerical experiments on CIFAR-10 with PreAct ResNet-18 to illustrate how the regularization hyper-parameter influences the performance of our robust classifiers. To develop robust classifiers for multi-class tasks, we use the gradient descent update formula in Algorithm 1, with $\mathcal{L}$ as the cross-entropy loss. Note that, for the zeroth term hyper-parameter $1/\lambda$ in Algorithm 1, we set $1/\lambda = 6$ for all of our experiments, which is a widely used setting for TRADES. We train the models with $\eta = 0.05$, 0.1, 0.2, 0.3, 0.4 respectively. All models are trained with 200 epochs and 128 batchsize.

In Tab. 1, we observe that as the regularization parameter η increases, the clean accuracy and robust accuracy almost both go up first and then down. In particular, the accuracy under Auto Attack is

Table 2: First and Second Derivative terms optimization on CIFAR-10/CIFAR-100 with ℓ_∞ threat model for WideResNet, compared with current state-of-the-art. Classification accuracy (%) on clean images and under PGD-20 attack, CW-20 attack and Auto Attack. The results of our methods are in **bold**. Note that * is under PGD-40 attack and ** is under PGD-10 attack.

Data Set	Method	Architecture	Clean	PGD-20	CW-20	AA
CIFAR-10 ($\ell_\infty, \epsilon = 8/255$)	Lee et al. (2020)	WRN-34-10	92.56	59.75	54.53	39.70
	Wang et al. (2020)	WRN-34-10	83.51	58.31	54.33	51.10
	Rice et al. (2020)	WRN-34-20	85.34	-	-	53.42
	Zhang et al. (2020)	WRN-34-10	84.52	-	-	53.51
	Pang et al. (2021)	WRN-34-20	86.43	57.91**	-	54.39
	Gowal et al. (2020)	WRN-70-16	85.29	58.22*	-	57.20
	Zhang et al. (2019) (0_{th})	WRN-34-10	84.65	56.68	54.49	53.0
	+ **Ours (1_{st})**	WRN-34-10	**85.51**	**58.72**	**56.06**	**54.0**
	+ **Ours (1_{st}+2_{nd})**	WRN-34-10	**85.98**	**58.84**	**56.13**	**54.2**
	Wu et al. (2020) (0_{th})	WRN-34-10	85.17	59.64	57.33	56.2
	+ **Ours (1_{st})**	WRN-34-10	**86.10**	**61.47**	**58.09**	**57.1**
	+ **Ours (1_{st}+2_{nd})**	WRN-34-10	**86.12**	**61.45**	**58.22**	**57.4**
CIFAR-100 ($\ell_\infty, \epsilon = 8/255$)	Cui et al. (2021)	WRN-34-10	60.43	35.50	31.50	29.34
	Gowal et al. (2020)	WRN-70-16	60.86	31.47*	-	30.03
	Zhang et al. (2019) (0_{th})	WRN-34-10	60.22	32.11	28.93	26.9
	+ **Ours (1_{st})**	WRN-34-10	**63.01**	**33.26**	**29.44**	**28.1**
	+ **Ours (1_{st}+2_{nd})**	WRN-34-10	**62.93**	**33.65**	**29.61**	**27.9**
	Wu et al. (2020) (0_{th})	WRN-34-10	60.38	34.09	30.78	28.6
	+ **Ours (1_{st})**	WRN-34-10	**63.98**	**35.36**	**31.63**	**29.8**
	+ **Ours (1_{st}+2_{nd})**	WRN-34-10	**64.71**	**35.73**	**31.41**	**30.2**

Table 3: Adversarial training across data sets on PreAct ResNet18 with ℓ_2 threat model. Classification accuracy (%) on clean images and under PGD-20 attack and Auto Attack. The results of our methods are in **bold**.

Method	CIFAR-10			CIFAR-100			SVHN		
	Clean	PGD	AA	Clean	PGD	AA	Clean	PGD	AA
Wu et al. (2020)(0_{th})	87.05	72.08	71.6	62.87	45.11	41.8	92.86	72.45	63.8
+ **Ours (1_{st}+2_{nd})**	**88.41**	**72.85**	**72.0**	**65.59**	**46.88**	**42.4**	**93.98**	**73.07**	**64.5**

sensitive to the regularization hyper-parameter η. Considering both robustness accuracy and clean accuracy, it is not difficult to find that $\eta = 0.3$ is the best for first term optimization and $\eta = 0.2$ is the best for first + second terms optimization. Thus in the following experiments, we set $\eta = 0.3$ for first term optimization and $\eta = 0.2$ for first + second terms optimization as default.

5.2 Comparison with state-of-the-art on WideResNet

In Tab. 2, we compare our method with state-of-the-art on WideResNet with CIFAR-10 and CIFAR-100. For zeroth term optimization, we use TRADES (Zhang et al., 2019) and AWP-TRADES (Wu et al., 2020) as baselines with $1/\lambda = 6$. We also report other state-of-the-art, include AVMixup (Lee et al., 2020), MART (Wang et al., 2020), Rice et al. (2020), Zhang et al. (2020), Pang et al. (2021), TRADES+LBGAT (Cui et al., 2021) and Gowal et al. (2020).

The results in Tab. 2 demonstrate that our method can both improve clean accuracy and robustness accuracy consistently over different datasets and models. Especially for Auto Attack, our method can get 57.4% on CIFAR-10 and 30.2% on CIFAR-100 with WRN-34-10, even surpass the performance of WRN-70-16 from Gowal et al. (2020).

5.3 Other empirical results

Adversarial training with ℓ_2 threat model. For the experiments with ℓ_2 threat model on CIFAR-10, CIFAR-100 and SVHN in Tab. 3, the results still support that our method can enhance the performance under clean data, PGD-20 attack and Auto Attack. For CIFAR-100, it can even increase about 3% on clean accuracy.

Table 4: Adversarial training across data sets on PreAct ResNet18 with ℓ_∞ threat model. Classification accuracy (%) on clean images and black-box attacks. Black-box adversarial examples are generated by a surrogate normal adversarial training model (of same setting) with PGD-20 attack and CW-20 attack. The results of our methods are in **bold**.

Method	CIFAR-10			CIFAR-100			SVHN		
	Clean	PGD	CW	Clean	PGD	CW	Clean	PGD	CW
Wu et al. (2020)(0_{th})	82.78	61.79	59.42	58.33	38.55	36.70	93.77	63.80	59.54
+ **Ours (1_{st}+2_{nd})**	**84.27**	**62.56**	**60.58**	**61.08**	**39.82**	**37.84**	**95.11**	**66.16**	**63.59**

Table 5: Adversarial training across VGG16, MobileNetV2 on CIFAR-10 with ℓ_∞ threat model. Classification accuracy (%) on clean images and under PGD-20 attack, CW-20 attack and Auto Attack. The results of our methods are in **bold**.

Method	VGG16				MobileNetV2			
	Clean	PGD-20	CW-20	AA	Clean	PGD-20	CW-20	AA
Zhang et al. (2019) (0_{th})	79.78	49.88	46.95	44.3	79.73	51.41	48.43	46.4
+ **Ours (1_{st}+2_{nd})**	**80.99**	**50.13**	**47.09**	**44.4**	**81.86**	**53.34**	**49.93**	**47.8**
Wu et al. (2020) (0_{th})	78.46	51.19	47.41	46.3	79.86	53.56	50.11	47.7
+ **Ours (1_{st}+2_{nd})**	**80.31**	**52.71**	**48.38**	**46.5**	**81.95**	**55.37**	**51.55**	**49.4**

Table 6: Time consumption and GPU memory usage for WideResNet on CIFAR-10 with ℓ_∞ threat model. We deploy each model on a single NVIDIA A100 with batchsize 128. The results of our methods are in **bold**.

Method	WideResNet-34-10	
	Time/Epoch	GPU Memory
Zhang et al. (2019) (0_{th})	1666s	12875MB
+ **Ours (1_{st})**	**2161s**	**20629MB**
+ **Ours (1_{st}+2_{nd})**	**2362s**	**26409MB**

Robustness under black-box attacks. We train PreAct ResNet18 for ℓ_∞ threat model on CIFAR-10, CIFAR-100 and SVHN. The black-box adversarial examples are generated by a surrogate normal adversarial training model (of same setting) with PGD-20 attack and CW-20 attack. Tab. 4 shows our method is also effective under black-box attacks. For SVHN, ours can even obtain a significant improvement over the existing ones.

Robustness on other architectures. We train VGG16 and MobileNetV2 for ℓ_∞ threat model on CIFAR-10. Our results in Tab. 5 show a comprehensive improvement under clean data and robustness accuracy. Particularly, for MobileNetV2, ours can achieve an improvement greater than 1% and 2% under Auto Attack and clean data, respectively.

We also discuss different types of noise $\mathbf{u}'$ in adversarial training in Appendix F.1 and the empirical results of adversarial examples generated by Expectation Over Transformation (EOT) method (Athalye et al., 2018) with $\mathbf{u}'$ in Appendix F.2.

6 Limitations and Future Work

We use some approximation methods, e.g., Eqs. (13), (14), to reduce the complexity of our adversarial training method. The results in Tab. 6 show that though our method with WideResNet-34-10 can work on a single NVIDIA A100, the growth of training time and GPU memory still cannot be ignored. In the future work, we plan to further reduce the complexity of our algorithm and apply the first and second terms optimization on more layers.

7 Conclusion

This work studied the trade-off between robustness and clean accuracy over the randomized weights. Through theoretical study of information-theoretic bound over both clean and adversarial data, algorithmic design (via Taylor expansion), and extensive experiments, we demonstrated that optimizing over randomized weights can further improve the performance of adversarial training methods.

REFERENCES

Alayrac, J., Uesato, J., Huang, P., Fawzi, A., Stanforth, R., and Kohli, P. (2019). Are labels required for improving adversarial robustness? In *NeurIPS*.

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. (2020). Square attack: a query-efficient black-box adversarial attack via random search. In *ECCV*.

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. (2018). Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR.

Balaji, Y., Goldstein, T., and Hoffman, J. (2019). Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. *CoRR*.

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

Carmon, Y., Raghunathan, A., Schmidt, L., Duchi, J. C., and Liang, P. (2019). Unlabeled data improves adversarial robustness. In *NeurIPS*.

Chen, T., Zhang, Z., Liu, S., Chang, S., and Wang, Z. (2020). Robust overfitting may be mitigated by properly learned smoothening. In *ICLR*.

Croce, F. and Hein, M. (2020a). Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*.

Croce, F. and Hein, M. (2020b). Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*.

Cui, J., Liu, S., Wang, L., and Jia, J. (2021). Learnable boundary guided adversarial training. In *ICCV*.

DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Ding, G. W., Sharma, Y., Lui, K. Y. C., and Huang, R. (2020). Mma training: Direct input space margin maximization through adversarial training. In *ICLR*.

Dong, Y., Deng, Z., Pang, T., Zhu, J., and Su, H. (2020a). Adversarial distributional training for robust deep learning. In *NeurIPS*.

Dong, Y., Fu, Q., Yang, X., Pang, T., Su, H., Xiao, Z., and Zhu, J. (2020b). Benchmarking adversarial robustness on image classification. In *CVPR*.

Dziugaite, G. K., Drouin, A., Neal, B., Rajkumar, N., Caballero, E., Wang, L., Mitliagkas, I., and Roy, D. M. (2020). In search of robust measures of generalization. In *NeurIPS*.

Dziugaite, G. K. and Roy, D. M. (2017). Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *UAI*.

Engstrom, L., Ilyas, A., and Athalye, A. (2018). Evaluating and understanding the robustness of adversarial logit pairing. *arXiv preprint arXiv:1807.10272*.

Farnia, F., Zhang, J. M., and Tse, D. (2019). Generalizable adversarial training via spectral normalization. In *ICLR*.

Goldfeld, Z., van den Berg, E., Greenewald, K. H., Melnyk, I., Nguyen, N., Kingsbury, B., and Polyanskiy, Y. (2019). Estimating information flow in deep neural networks. In *ICML*.

Gowal, S., Qin, C., Uesato, J., Mann, T. A., and Kohli, P. (2020). Uncovering the limits of adversarial training against norm-bounded adversarial examples. *CoRR*.

Gowal, S., Rebuffi, S., Wiles, O., Stimberg, F., Calian, D. A., and Mann, T. A. (2021). Improving robustness using generated data. In *NeurIPS*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.

Hendrycks, D., Lee, K., and Mazeika, M. (2019). Using pre-training can improve model robustness and uncertainty. In *ICML*.

Javanmard, A., Soltanolkotabi, M., and Hassani, H. (2020). Precise tradeoffs in adversarial training for linear regression. In *COLT*.

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. (2020). Fantastic generalization measures and where to find them. In *ICLR*.

Jin, G., Yi, X., Huang, W., Schewe, S., and Huang, X. (2022). Enhancing adversarial training with second-order statistics of weights. In *CVPR*.

Jin, G., Yi, X., Zhang, L., Zhang, L., Schewe, S., and Huang, X. (2020). How does weight correlation affect generalisation ability of deep neural networks? In *NeurIPS*.

Kannan, H., Kurakin, A., and Goodfellow, I. (2018). Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Langford, J. and Shawe-Taylor, J. (2002). Pac-bayes & margins. In *NeurIPS*.

Lee, S., Lee, H., and Yoon, S. (2020). Adversarial vertex mixup: Toward better adversarially robust generalization. In *CVPR*.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *ICLR*.

Mao, C., Zhong, Z., Yang, J., Vondrick, C., and Ray, B. (2019). Metric learning for adversarial robustness. In *NeurIPS*.

McAllester, D. A. (1999). Pac-bayesian model averaging. In *COLT*.

Müller, R., Kornblith, S., and Hinton, G. (2019). When does label smoothing help? In *NeurIPS*.

Negrea, J., Haghifam, M., Dziugaite, G. K., Khisti, A., and Roy, D. M. (2019). Information-theoretic generalization bounds for sgld via data-dependent estimates. In *NeurIPS*.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017). Exploring generalization in deep learning. In *NeurIPS*.

Neyshabur, B., Bhojanapalli, S., and Srebro, N. (2018). A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *ICLR*.

Pang, T., Lin, M., Yang, X., Zhu, J., and Yan, S. (2022). Robustness and accuracy could be reconcilable by (proper) definition. *CoRR*.

Pang, T., Yang, X., Dong, Y., Su, H., and Zhu, J. (2021). Bag of tricks for adversarial training. In *ICLR*.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical black-box attacks against machine learning. In *AsiaCCS*.

Parrado-Hernández, E., Ambroladze, A., Shawe-Taylor, J., and Sun, S. (2012). Pac-bayes bounds with data dependent priors. *J. Mach. Learn. Res.*

Raghunathan, A., Xie, S. M., Yang, F., Duchi, J. C., and Liang, P. (2020). Understanding and mitigating the tradeoff between robustness and accuracy. In *ICML*.

Rebuffi, S., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., and Mann, T. A. (2021). Fixing data augmentation to improve adversarial robustness. *CoRR*.

Rice, L., Wong, E., and Kolter, J. Z. (2020). Overfitting in adversarially robust deep learning. In *ICML*.

Russo, D. and Zou, J. (2019). How much does your data exploration overfit? controlling bias via information usage. *IEEE Transactions on Information Theory*, 66(1):302–323.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. (2018). Adversarially robust generalization requires more data. In *NeurIPS*.

Seeger, M. W. (2002). Pac-bayesian generalisation error bounds for gaussian process classification. *J. Mach. Learn. Res.*

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Staib, M. and Jegelka, S. (2017). Distributionally robust deep learning as a generalization of adversarial training. *NIPS workshop on Machine Learning and Computer Security*.

Su, D., Zhang, H., Chen, H., Yi, J., Chen, P., and Gao, Y. (2018). Is robustness the cost of accuracy? - A comprehensive study on the robustness of 18 deep image classification models. In *ECCV*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *CVPR*.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2019). Robustness may be at odds with accuracy. In *ICLR*.

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. (2020). Improving adversarial robustness requires revisiting misclassified examples. In *ICLR*.

Wu, D., Xia, S., and Wang, Y. (2020). Adversarial weight perturbation helps robust generalization. In *NeurIPS*.

Xu, A. and Raginsky, M. (2017). Information-theoretic analysis of generalization capability of learning algorithms. In *NeurIPS*.

Yu, Y., Yang, Z., Dobriban, E., Steinhardt, J., and Ma, Y. (2021). Understanding generalization in adversarial training via the bias-variance decomposition. *CoRR*.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In *BMVC*.

Zhang, H. and Wang, J. (2019). Defense against adversarial attacks using feature scattering-based adversarial training. In *NeurIPS*.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. (2019). Theoretically principled trade-off between robustness and accuracy. In *ICML*.

Zhang, J., Xu, X., Han, B., Niu, G., Cui, L., Sugiyama, M., and Kankanhalli, M. S. (2020). Attacks which do not kill training make adversarial learning stronger. In *ICML*.

Zhang, J., Zhu, J., Niu, G., Han, B., Sugiyama, M., and Kankanhalli, M. S. (2021a). Geometry-aware instance-reweighted adversarial training. In *ICLR*.

Zhang, L., Deng, Z., Kawaguchi, K., Ghorbani, A., and Zou, J. (2021b). How does mixup help with robustness and generalization? In *ICLR*.

Zheng, T., Chen, C., and Ren, K. (2019). Distributionally adversarial attack. In *AAAI*.

A More Related Works

A.1 Adversarial training

Adversarial training methods can generally be divided into three groups. In the first group, Eq. (1) is translated into an equivalent, (or approximate, expression), which is mainly used to narrow the distance between $f_w(\mathbf{s})$ and $f_w(\mathbf{s}')$. For example, ALP (Engstrom et al., 2018; Kannan et al., 2018) estimates the similarity between $f_w(\mathbf{s})$ and $f_w(\mathbf{s}')$, and maximizes this similarity in the objective function. MMA (Ding et al., 2020) proposes each correctly classified instance $\mathbf{s}$ to leave sufficiently the decision boundary, through making the size of shortest successful perturbation as large as possible. TRADES (Zhang et al., 2019) looks for the adversarial example $\mathbf{s}'$ which can obtain the largest KL divergence between $f_w(\mathbf{s})$ and $f_w(\mathbf{s}')$, then trains with these adversarial examples. Besides, Zheng et al. (2019) adopts the similarity between local distributions of natural and adversarial example, Staib and Jegelka (2017) measures the similarity of local distributions with Wasserstein distance, and Mao et al. (2019); Dong et al. (2020a;b); Pang et al. (2021) try to optimize with the distribution over a series of adversarial examples for a single input.

In the second group, adversarial examples are pre-processed before being used for training rather than being directly generated by attack algorithms. For example, label smoothing (Szegedy et al., 2016; Chen et al., 2020) replaces the hard label y with the soft label $\tilde{y}$, where $\tilde{y}$ is a combination of the hard label y and uniform distribution. A further empirical exploration with of how label smoothing works is provided in Müller et al. (2019). AVMixup (Zhang et al., 2021b; Lee et al., 2020) defines a virtual sample in the adversarial direction based on these. Through linear interpolation of the virtual sample and the clean sample, it extends the training distribution with soft labels. Instead of smoothing labels, Zhang and Wang (2019) perturbs the local neighborhoods with an unsupervised way to produce adversarial examples. In addition, data augmentation (Rebuffi et al., 2021; Gowal et al., 2021) is also shown to be an invaluable aid for adversarial training.

The above two groups merely adjust the component parts of the min-max formalism. With AWP (Wu et al., 2020), one more maximization is performed to find a weight perturbation based on the generated adversarial examples. The altered weights (DeVries and Taylor, 2017) are then used in the outer minimization function to reduce the loss caused by the adversarial cases.

Different from previous work, this paper optimizes the distance between $f_{w+\mathbf{u}'}(\mathbf{s})$ and $f_{w+\mathbf{u}'}(\mathbf{s}')$ through randomized weights, and therefore, effectively, it considers the optimization problem where the decision boundary are smoothed by these randomized models.

A.2 Randomized weights

In the previous work to study generalization or robustness, modeling neural network weights as random variables has been frequently used. For example, Xu and Raginsky (2017); Negrea et al. (2019) estimate the mutual information between random weights and data set, provide the expected generalization bound over weight distribution. Another example of random weights is on the PAC-Bayesian framework, a well-known theoretical tool to bound the generalization error of machine learning models (McAllester, 1999; Seeger, 2002; Langford and Shawe-Taylor, 2002; Parrado-Hernández et al., 2012). In the recent years, PAC-Bayes is also developed to bound the generalization error or robustness error of DNNs (Dziugaite and Roy, 2017; Neyshabur et al., 2017; Farnia et al., 2019; Jiang et al., 2020; Jin et al., 2020). In addition, Goldfeld et al. (2019) draws random noise into deterministic weights, to measure the mutual information between input data set and activations under information bottleneck theory.

While we also consider randomized weights, a major difference from previous theoretical works is that randomized weights are applied to robustness bound, from which we find an insight to design a novel adversarial training method which decomposes objective function (parameterized with randomized weights) with Taylor series.

B PROOF OF LEMMA 3.3

Proof 3.3 *We suppose the adversarial trained model contains information of $\mathcal{S} \vee \mathcal{S}'$, where $\mathcal{S} \vee \mathcal{S}'$ is the joint set with $\mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S} \vee \mathcal{S}'), \mathcal{Y}) = q\mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}) + (1-q)\mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y})$. We let $q \in (0, 0.5)$ because $\mathcal{S}'$ occupies a large proportion in adversarial training. Randomized weight $\mathbf{w}$ is generated by normal training with data set $\mathcal{S}$ and $\mathbf{w}+\mathbf{u}$ is generated by adversarial training with joint set $\mathcal{S} \vee \mathcal{S}'$, then*

$$\begin{aligned}
&I\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big), \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\Big) \\
&= H\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big), \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big)\Big) \\
&\quad - H\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big), \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big)\Big|\mathbf{w}+\mathbf{u}\Big) \\
&= H\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big)\Big) + H\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big)\Big|\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big)\Big) \\
&\quad - H\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big)\Big|\mathbf{w}+\mathbf{u}\Big) - H\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big)\Big|\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big), \mathbf{w}+\mathbf{u}\Big) \\
&= H\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big)\Big) + H\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big)\Big) - I\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big); \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big)\Big) \\
&\quad - H\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big)\Big|\mathbf{w}+\mathbf{u}\Big) - H\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big)\Big|\mathbf{w}+\mathbf{u}\Big) \\
&\quad + I\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big); \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big)\Big|\mathbf{w}+\mathbf{u}\Big) \\
&= I\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\Big) + I\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\Big) \\
&\quad - I\Big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big); \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\Big).
\end{aligned} \tag{15}$$

Note that $I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big) \leq I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big)$ holds, due to the data processing inequality and $\mathcal{S} - \mathcal{S} \vee \mathcal{S}' - \mathbf{w}+\mathbf{u}$ forms a Markov chain under adversarial training where $\mathcal{S}'$ occupies a larger proportion in $\mathcal{S} \vee \mathcal{S}'$. As $I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big) \geq I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big); \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big)$, we can easily get Lem. 3.3.

C PROOF OF LEMMA 3.4

Proof 3.4 *Let $f_{\mathbf{w}+\mathbf{u}}(\mathbf{s})_{true}$ be the normalized output of $f_{\mathbf{w}+\mathbf{u}}(\mathbf{s})$ for true label and we consider the case of binary response $y \in \{0, 1\}$ in Lem. 3.4. Then,*

$$\begin{aligned}
&|\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}), y\big) - \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}'), y\big)| \\
&= \Big|\log\frac{f_{\mathbf{w}+\mathbf{u}}(\mathbf{s})_{true}}{f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}')_{true}}\Big| \\
&\propto f_{\mathbf{w}+\mathbf{u}}(\mathbf{s})_{true}\log\frac{f_{\mathbf{w}+\mathbf{u}}(\mathbf{s})_{true}}{f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}')_{true}} + (1 - f_{\mathbf{w}+\mathbf{u}}(\mathbf{s})_{true})\log\frac{1 - f_{\mathbf{w}+\mathbf{u}}(\mathbf{s})_{true}}{1 - f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}')_{true}} \\
&= \mathrm{KL}\big(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s})||f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}')\big),
\end{aligned} \tag{16}$$

where we let $\propto$ represent positive correlation.

The above proof demonstrates $|\mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}), y) - \mathcal{L}(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}'), y)|$ is positive correlated with $\mathrm{KL}(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s})||f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}'))$ in a binary case, this can also approximately hold in a multi-class case. Thus, it allows us to optimize the mutual information of $I\big(\mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}), \mathcal{Y}\big), \mathcal{L}\big(f_{\mathbf{w}+\mathbf{u}}(\mathcal{S}'), \mathcal{Y}\big); \mathbf{w}+\mathbf{u}\big)$ utilizing a simplified term of $\mathrm{KL}(f_{\mathbf{w}+\mathbf{u}}(\mathbf{s})||f_{\mathbf{w}+\mathbf{u}}(\mathbf{s}'))$.

D Discussion of Equations (13), (14)

It is easy to see that minimizing $\mathbb{E}_{\mathbf{u}'}(\mathcal{L}(g'_{\mathbf{s}}(w)^T\mathbf{u}', g'_{\mathbf{s}'}(w)^T\mathbf{u}'))$, $\mathbb{E}_{\mathbf{u}'}(\mathcal{L}(\mathbf{u}'^T g''_{\mathbf{s}}(w)\mathbf{u}', \mathbf{u}'^T g''_{\mathbf{s}'}(w)\mathbf{u}'))$ equals reducing the distance between $g'_{\mathbf{s}}(w)$ and $g'_{\mathbf{s}'}(w)$, $g''_{\mathbf{s}}(w)$ and $g''_{\mathbf{s}'}(w)$, respectively. As a first step, we consider to reduce this distance and only optimize it on the final (L-th) layer to further reduce the complexity of our method. Normally, the ℓ_2 distance between vectors $g'_{\mathbf{s}}(w)$ and $g'_{\mathbf{s}'}(w)$ can be defined as

$$\Big|\Big|\frac{\partial g_{\mathbf{s}}(w)}{\partial w_L} - \frac{\partial g_{\mathbf{s}'}(w)}{\partial w_L}\Big|\Big|_2^2 = \sum_{j=1}^{N_L}\sum_i \Big[\frac{\partial(g_{\mathbf{s}}(w) - g_{\mathbf{s}'}(w))}{\partial W_{L(j,i)}}\Big]^2. \tag{17}$$

We extend Eq. (17) by considering the sum of each row vector of $\frac{\partial(g_{\mathbf{s}}(w)-g_{\mathbf{s}'}(w))}{\partial W_L}$ and define the distance between $\frac{\partial g_{\mathbf{s}}(w)}{\partial w_L}$ and $\frac{\partial g_{\mathbf{s}'}(w)}{\partial w_L}$ as

$$\sum_{j=1}^{N_L}\Big[\sum_i \frac{\partial(g_{\mathbf{s}}(w) - g_{\mathbf{s}'}(w))}{\partial W_{L(j,i)}}\Big]^2. \tag{18}$$

We notice that, according to chain rule, $\Big[\sum_i \frac{\partial(g_{\mathbf{s}}(w)-g_{\mathbf{s}'}(w))}{\partial W_{L(j,i)}}\Big]^2$ corresponds to $g_{\mathbf{s}}(w)_j$ and $g_{\mathbf{s}'}(w)_j$, where $g_{\mathbf{s}}(w)_j$ and $g_{\mathbf{s}'}(w)_j$ are the j-th output of $g_{\mathbf{s}}(w)$ and $g_{\mathbf{s}'}(w)$ respectively. We can increase $g_{\mathbf{s}'}(w)_j$ to reduce $\sum_i \frac{\partial(g_{\mathbf{s}}(w)-g_{\mathbf{s}'}(w))}{\partial W_{L(j,i)}}$ when $\sum_i \frac{\partial(g_{\mathbf{s}}(w)-g_{\mathbf{s}'}(w))}{\partial W_{L(j,i)}}$ is large. In contrast, we can increase $g_{\mathbf{s}}(w)_j$ to reduce $\sum_i \frac{\partial(g_{\mathbf{s}'}(w)-g_{\mathbf{s}}(w))}{\partial W_{L(j,i)}}$ when $\sum_i \frac{\partial(g_{\mathbf{s}'}(w)-g_{\mathbf{s}}(w))}{\partial W_{L(j,i)}}$ is large. Thus, we can approximately optimize $\mathbb{E}_{\mathbf{u}'}(\mathcal{L}(g'_{\mathbf{s}}(w)^T\mathbf{u}', g'_{\mathbf{s}'}(w)^T\mathbf{u}'))$ through minimizing

$$\begin{aligned}&\frac{1}{2}\mathcal{L}\Big(g_{\mathbf{s}'}(w), \Big[\sum_i \frac{\partial(g_{\mathbf{s}}(w) - g_{\mathbf{s}'}(w))}{\partial W_{L(1,i)}}, ..., \sum_i \frac{\partial(g_{\mathbf{s}}(w) - g_{\mathbf{s}'}(w))}{\partial W_{L(N_L,i)}}\Big]^T\Big) \\ &+ \frac{1}{2}\mathcal{L}\Big(g_{\mathbf{s}}(w), \Big[\sum_i \frac{\partial(g_{\mathbf{s}'}(w) - g_{\mathbf{s}}(w))}{\partial W_{L(1,i)}}, ..., \sum_i \frac{\partial(g_{\mathbf{s}'}(w) - g_{\mathbf{s}}(w))}{\partial W_{L(N_L,i)}}\Big]^T\Big).\end{aligned} \tag{19}$$

Similarly, we define the distance between $\frac{\partial^2 g_{\mathbf{s}}(w)}{\partial w_L \partial w_L}$ and $\frac{\partial^2 g_{\mathbf{s}'}(w)}{\partial w_L \partial w_L}$ as

$$\sum_{l=1}^{N_L}\Big[\sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}}(w) - \partial^2 g_{\mathbf{s}'}(w)}{\partial W_{L(l,i)}\partial W_{L(j,k)}}\Big]^2. \tag{20}$$

We also notice that, according to chain rule, $\Big[\sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}}(w)-\partial^2 g_{\mathbf{s}'}(w)}{\partial W_{L(l,i)}\partial W_{L(j,k)}}\Big]^2$ corresponds to $g_{\mathbf{s}}(w)_l$ and $g_{\mathbf{s}'}(w)_l$. We can increase $g_{\mathbf{s}'}(w)_l$ to reduce $\sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}}(w)-\partial^2 g_{\mathbf{s}'}(w)}{\partial W_{L(l,i)}\partial W_{L(j,k)}}$ when $\sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}}(w)-\partial^2 g_{\mathbf{s}'}(w)}{\partial W_{L(l,i)}\partial W_{L(j,k)}}$ is large. In contrast, we can increase $g_{\mathbf{s}}(w)_l$ to reduce $\sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}'}(w)-\partial^2 g_{\mathbf{s}}(w)}{\partial W_{L(l,i)}\partial W_{L(j,k)}}$ when $\sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}'}(w)-\partial^2 g_{\mathbf{s}}(w)}{\partial W_{L(l,i)}\partial W_{L(j,k)}}$ is large. Thus, we can approximately optimize $\mathbb{E}_{\mathbf{u}'}(\mathcal{L}(\mathbf{u}'^T g''_{\mathbf{s}}(w)\mathbf{u}', \mathbf{u}'^T g''_{\mathbf{s}'}(w)\mathbf{u}'))$ through minimizing

$$\begin{aligned}&\frac{1}{2}\mathcal{L}\Big(g_{\mathbf{s}'}(w), \Big[\sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}}(w) - \partial^2 g_{\mathbf{s}'}(w)}{\partial W_{L(1,i)}\partial W_{L(j,k)}}, ..., \sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}}(w) - \partial^2 g_{\mathbf{s}'}(w)}{\partial W_{L(N_L,i)}\partial W_{L(j,k)}}\Big]^T\Big) \\ &+ \frac{1}{2}\mathcal{L}\Big(g_{\mathbf{s}}(w), \Big[\sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}'}(w) - \partial^2 g_{\mathbf{s}}(w)}{\partial W_{L(1,i)}\partial W_{L(j,k)}}, ..., \sum_i\sum_j\sum_k \frac{\partial^2 g_{\mathbf{s}'}(w) - \partial^2 g_{\mathbf{s}}(w)}{\partial W_{L(N_L,i)}\partial W_{L(j,k)}}\Big]^T\Big).\end{aligned} \tag{21}$$

Table 7: Adversarial training for PreAct ResNet18 on CIFAR-10 with ℓ_∞ threat model, $[-0.0001, 0.0001]$ uniform noise $\mathbf{u}'$. Classification accuracy (%) on clean images and under PGD-20 attack, CW-20 attack and Auto Attack. The results of our methods are in **bold**. $\eta = 0.3$ for first term optimization and $\eta = 0.2$ for first + second terms optimization.

Method	CIFAR-10			
	Clean	PGD-20	CW-20	AA
Zhang et al. (2019) (0_{th})	82.89	53.81	51.83	48.6
+ Ours (1_{st})	**83.78**	**55.01**	**52.69**	**49.5**
+ Ours (1_{st}+2_{nd})	**84.21**	**55.34**	**52.69**	**50.7**

Table 8: Adversarial training for PreAct ResNet18 on CIFAR-10 with ℓ_∞ threat model. Classification accuracy (%) on clean images and under PGD-20 attack, CW-20 attack and Auto Attack. All adversarial examples are generated by Expectation over Transformation method (EoT) method, i.e., the adversarial example is the average of 5 adversarial examples generated by $w + \mathbf{u}'$, $\mathbf{u}' \sim \mathcal{N}(\mathbf{0}, 0.0001 \cdot \mathbf{I})$. The results of our methods are in **bold**. $\eta = 0.3$ for first term optimization and $\eta = 0.2$ for first + second terms optimization.

Method	CIFAR-10			
	Clean	PGD-20	CW-20	AA
Zhang et al. (2019) (0_{th})	82.89	54.05	51.89	48.7
+ Ours (1_{st})	**83.96**	**55.12**	**52.49**	**49.7**
+ Ours (1_{st}+2_{nd})	**84.13**	**55.19**	**52.61**	**50.8**

E Details of the experiments

E.1 Network Architecture

For all of our experiments in Sec. 5, we use 4 network architectures as ResNet, WideResNet, VGG and MobileNetV2. We present the details in the following.

- ResNet/WideResNet: Architectures used are PreAct ResNet. All convolutional layers (except downsampling convolutional layers) have kernel size 3×3 with stride 1. Downsampling convolutions have stride 2. All the ResNets have five stages (0-4) where each stage has multiple residual/downsampling blocks. These stages are followed by a max-pooling layer and a final linear layer. We study the PreAct ResNet 18 and WideResNet-34-10.
- VGG: Architecture consists of multiple convolutional layers, followed by multiple fully connected layers and a final classifier layer (with output dimension 10 or 100). We study the VGG networks with 16 layers.
- MobileNetV2: Architecture is built on an inverted residual structure, with residual connections between bottleneck layers. As a source of non-linearity, the intermediate expansion layer filters features with lightweight depthwise convolutions. As a whole, the architecture of MobileNetV2 includes a fully convolutional layer with three filters, followed by 19 residual bottleneck layers.

E.2 Checkpoints

We set checkpoints for each epoch between $100-110$ and $150-160$, each 5 epoch between $110-150$ and $160-200$. All best performances are got from these checkpoints.

F Supplementary experiment

F.1 Different noises of $\mathbf{u}'$

In our method, the core is bringing in randomness and using Tylor expansion to optimize the model in multi directions. The type of small perturbation $\mathbf{u}'$ is relatively unimportant. As Tab. 7 shows, the empirical results of small uniform perturbation are similar with the results of small Gaussian perturbation in Tab. 1.

F.2 EOT METHOD TO GENERATE ADVERSARIAL EXAMPLES

The results in Tab. 8 show that our method can still improve the performance of both robustness and clean accuracy, where the adversarial examples are generated by Expectation over Transformation method (EoT).