Randomized Adversarial Training via Taylor Expansion

Gaojie Jin\textsuperscript{1,2}, Xinping Yi\textsuperscript{2}\textsuperscript{3}, Dengyu Wu\textsuperscript{2}, Ronghui Mu\textsuperscript{3}, Xiaowei Huang\textsuperscript{2}\textsuperscript{3}

\textsuperscript{1}State Key Laboratory of Computer Science, Institute of Software, CAS, Beijing, China
\textsuperscript{2}University of Liverpool, Liverpool, UK
\textsuperscript{3}Lancaster University, Lancaster, UK
\{g.jin3, xinping.yi, xiaowei.huang\}@liverpool.ac.uk

Corresponding Author

Abstract

In recent years, there has been an explosion of research into developing more robust deep neural networks against adversarial examples. Adversarial training appears as one of the most successful methods. To deal with both the robustness against adversarial examples and the accuracy over clean examples, many works develop enhanced adversarial training methods to achieve various trade-offs between them \cite{19, 38, 80}. Leveraging over the studies \cite{8, 32} that smoothed update on weights during training may help find flat minima and improve generalization, we suggest reconciling the robustness-accuracy trade-off from another perspective, i.e., by adding random noise into deterministic weights. The randomized weights enable our design of a novel adversarial training method via Taylor expansion of a small Gaussian noise, and we show that the new adversarial training method can flatten loss landscape and find flat minima. With PGD, CW, and Auto Attacks, an extensive set of experiments demonstrate that our method enhances the state-of-the-art adversarial training methods, boosting both robustness and clean accuracy. The code is available at https://github.com/Alexkael/Randomized-Adversarial-Training.

1. Introduction

Trade-off between adversarial robustness and clean accuracy has recently been intensively studied \cite{63, 68, 80} and demonstrated to exist \cite{34, 58, 70, 77}. Many different techniques have been developed to alleviate the loss of clean accuracy when improving robustness, including data augmentation \cite{1, 5, 29}, early-stopping \cite{60, 81}, instance reweighting \cite{3, 82}, and various adversarial training methods \cite{11, 36, 42, 74, 80}. Adversarial training is believed to be the most effective defense method against adversarial attacks, and 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 which interact with one another through the network weights, it is reasonable to relax the assumption that neural networks are simply deterministic models where weights are scalar values. This paper is based on a view, illustrated in Fig. 1, that randomized models enable the training optimization to consider multiple directions within a small area and may achieve smoothed weights update – in a way different from checkpoint averaging \cite{8, 32} – and obtain robust models against new clean/adversarial examples.

Building upon the above view, we find a way, drastically different from most existing studies in adversarial training, to balance robustness and clean accuracy, that is, embedding neural network weights with random noise. Whilst the randomized weights framework is not new in statistical learning theory, where it has been used in many previous works for e.g., generalization analysis \cite{18, 52, 75}, we hope to advance the empirical 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 improve the state-of-the-art adversarial training methods over both the adversarial robustness and clean accuracy.

By modeling weights as randomized variables with an artificially injected weight perturbation, we start with an empirical analysis of flatness of loss landscape in Sec. 3. We show that our method can flatten loss landscape and find flatter minima in adversarial training, which is generally regarded as an indicator of good generalization ability. After the flatness analysis, we show how to optimize with randomized weights during adversarial training in Sec. 4.
novel adversarial training method based on TRADES [80] is proposed to reconcile 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, since our method takes randomized models into consideration during training, the learned boundary is smoother and the learned model is more robust in a small perturbed area.

We validate the effectiveness of our optimization method with the first and the second derivative terms of Taylor series over weights, in such a way that we can deconstruct the robustness loss function over randomized weights, and offer a new insight on the smoothed adversarial training of TRADES and 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 smoother and more robust against new data.

Our method can further enhance the state-of-the-art adversarial training methods, such as TRADES [80], MobileNetV2 [62]), we find that the first and the second derivative terms can thus replace the objective function effectively and time-efficiently. As Fig. 1 shows, since our method takes randomized models into consideration during training, the learned boundary is smoother and the learned model is more robust in a small perturbed area.

Overall, this paper makes the following contributions:

- We propose a novel adversarial training method based on TRADES [80] to reconcile adversarial robustness with clean accuracy by closing the gap between clean latent space and adversarial latent space over randomized weights.
- We conduct a pilot study of the trade-off between adversarial robustness and clean accuracy for adversarial training methods, such as TRADES [80], MobileNetV2 [62]), and offer a new insight on the smoothed adversarial training.
- We demonstrate that our method can improve both robustness and clean accuracy consistently across different datasets and different network architectures (Sec. 5).

2. Preliminaries

Basic Notation. Consider the classification task with the training set \(S = \{s_1, ..., s_m\}\) where \(m\) samples are drawn from the data distribution \(D\). For notational convenience, we omit the label \(y\) of sample \(s\). The adversarial example \(s'\) is generally not in the natural dataset \(S\), such that \(\|s' - s\|_p \leq \epsilon\) where \(\| \cdot \|_p\) is by default the \(\ell_p\)-norm. Let \(f_w(\cdot)\) be a learning model parameterized by \(w\), where \(w\) is the vectorization of weight matrix \(W\). \(L(f_w(s), y)\) is the cross-entropy loss between \(f_w(s)\) and \(y\) with normalization. Define \(L(f_w(S), Y) := \mathbb{E}_{s \in S}[L(f_w(s), y)]\).

Vanilla Adversarial Training. Adversarial training can be formulated as a minimax optimization problem [45]

\[
\min_w \left\{ \mathbb{E}_{s \sim D} \left[ \max_{s' : \|s' - s\|_p \leq \epsilon} L(f_w(s'), y) \right] \right\},
\]

where \(s'\) is an adversarial example causing the largest loss within an \(\epsilon\)-ball centered at a clean example \(s\) with respect to a norm distance. Adversarial training aims to learn a model with the loss of adversarial examples minimized, without caring about the loss of the clean examples.

TRADES-like Adversarial Training. Advanced adversarial training methods, such as TRADES [80], strike a trade-off between clean accuracy and adversarial robustness, by optimizing the following problem

\[
\min_w \left\{ \mathbb{E}_{s \sim D} \left[ L(f_w(s), y) + \max_{s' : \|s' - s\|_p \leq \epsilon} L(f_w(s), f_w(s')) / \lambda \right] \right\},
\]

where the first term contributes to clean accuracy, and the second term with hyperparameter \(\lambda\) can be seen as a regularization for adversarial robustness that balances the outputs of clean and adversarial examples.

Nevertheless, each adversarial example \(s'\) is generated from a clean example \(s\) based on a specific weight \(w\), and
Figure 2. Comparison of loss landscapes of TRADES trained model (the first row) and TRADES+1st+2nd (our method, the second row) trained model. Loss plots in each column are generated from the same original image randomly chosen from the CIFAR-10 test dataset. Following the settings in [20], the $z$ axis represents the loss, the $x$ and $y$ axes represent the magnitude of the perturbation added in the directions of $\text{sign} \nabla_s f(s)$ and Rademacher(0.5) respectively. We provide more empirical results in Appendix A.

the weight $w$ is obtained by optimizing both clean and adversarial data with SGD. This is a chick-and-egg problem. A not-very-well-trained model $w$ may generate some misleading adversarial data samples, which in turn drive the training process away from the optimum.

Inspired by the idea of model smoothing, we believe that the introduction of noise term into the training process may smooth the update of weights and thus robustify the models. To elaborate on this, we introduce our method from the perspective of flatness of loss landscape in Sec. 3. This is followed by a formalization in Sec. 4, where a noise term is injected to network weight $w$ during adversarial training to transform deterministic weights into randomized weights.

3. Motivation

Flatness is commonly thought to be a metric of standard generalization: the loss surface at the final learnt weights for well-generalizing models is generally “flat” [18, 21, 35, 52]. Moreover, [73] believed that a flatter adversarial loss landscape reduces the generalization gap in robustness. This aligns with [28], where the authors demonstrated that the local Lipschitz constant can be used to explicitly quantify the robustness of machine learning models. Many empirical defensive approaches, such as hessian/curvature-based regularization [48], gradient magnitude penalty [71], smoothing with random noise [44], or entropy regularization [33], have echoed the flatness performance of a robust model. However, all the above approaches require significant computational or memory resources, and many of them, such as hessian/curvature-based solutions, may suffer from standard accuracy decreases [26].

Stochastic weight averaging (SWA) [32] is known to find solutions that are far flatter than SGD, is incredibly simple to implement, enhances standard generalization, and has almost no computational overhead. It was developed to guarantee weight smoothness by simply averaging various checkpoints throughout the training trajectory. SWA has been applied to semi-supervised learning, Bayesian inference, and low-precision training. Especially, [8] used SWA for the first time in adversarial training in order to smooth the weights and locate flatter minima, and showed that it enhances the adversarial generalization.

Different from SWA, we propose a novel method (with details provided in Sec. 4) via Taylor expansion of a small Gaussian noise to smooth the update of weights. This method regards the TRADES [80], a state-of-the-art adversarial training method, as a special case with only zero-th term, and is able to locate flatter minima. To demonstrate the latter, we visualize the loss landscape with respect to both input and weight spaces. As shown in Fig. 2, compared to the TRADES, our method significantly flattens the rugged landscape. In addition, it is worth noting that our empirical results (in Sec. 5) demonstrate that our method can further improve not only clean accuracy but also adversarial accuracy compared with other SOTA (includes SWA from [8]).

4. Methodology

Given a deterministic model of weight $w$, we consider an additive small Gaussian noise $u$ to smooth $w$. Based on TRADES [80], we propose that the robust optimization problem can be further enhanced by

\[
\min_w \mathbb{E}_u \left[ \mathbb{E}_s (\mathcal{L}(f_{w+u}(s), y)) + \frac{\mathbb{E}_{s, s'}: |s' - s|_p \leq \epsilon \max_{s'} \mathbb{KL}(f_{w+u}(s)||f_{w+u}(s'))/\lambda \right], \tag{2}
\]

where the first term contributes to clean accuracy, and the second term with hyperparameter $\lambda$ can be seen as a regu-
Algorithm 1: Adversarial training with randomized weights

**Input:** minibatch \( \{s_i\}_{i=1}^n \), network architecture parametrized by \( w \), learning rate \( \eta \), step size \( \eta_s \), hyper-parameters \( \eta, \lambda \), zero mean Gaussian \( u \), 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 \( u \)
  - for \( i = 1 \) to \( n \) do
    - \( s_i' \leftarrow s_i + 0.001 \cdot \mathcal{N}(0, I) \)
    - for \( k = 1 \) to \( K \) do
      - \( s_i' \leftarrow \Pi(\eta_s \text{sign}(\nabla_{s_i'} \mathcal{L}(f_{w+u}(s_i), f_{w+u}(s_i'))) + s_i') \), subject to \( ||s_i - s_i'||_p \leq \epsilon \), where \( \Pi \) is the projection operator.
  - end for
- Optimization: \( w \leftarrow w - \eta \sum_{i=1}^{n} \nabla_w \mathcal{L}(f_w(s_i), y_i) \) + \( L(g_w(w), g_{w'}(w))/\lambda \)
  - (zeroth term)
  - + \( \eta \mathbb{E}_u (L(g_{s_i}'(w)^T u, g_{s_i}'(w)^T u)) \)
  - (first term)
  - + \( \frac{\eta}{2} \mathbb{E}_u (L(u^T g_{s_i}'(w) u, u^T g_{s_i}'(w) u)) \)
  - (second term)

where various optimization methods can be applied: zeroth term optimization (TRADES), zeroth + first terms optimization and zeroth + first + second terms optimization.

until training converged

larization for adversarial robustness with weights \( w \) and a small Gaussian noise \( u \).

Following \cite{17, 35}, we let the Gaussian \( u \) be small enough to maintain the generalization performance on clean data. Then, we can extend Eq. (2) by replacing KL(\( \cdot \mid \cdot \)) with a multi-class calibrated loss \( \mathcal{L}(:,:, :) \) \cite{80}:

\[
\min_w \mathbb{E}_s \left[ \mathcal{L}(f_w(s), y) \right] + \mathbb{E}_s \max_{s':||s'-s||_p \leq \epsilon} \mathcal{L}(f_{w+u}(s), f_{w+u}(s'))/\lambda. \tag{3}
\]

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

\[
s_{t+1}^i \leftarrow \Pi \left( s_{t}^i + \eta_s \text{sign}(\nabla_{s_i'} \mathcal{L}(f_{w+u}(s_i), f_{w+u}(s_i')) + s_i') \right) \tag{4}
\]

until the maximum allowed iterations are reached, where \( \Pi \) is the projection operator, \( \eta_s \) is the step size and \( u \) is a sample of \( u \). To reduce complexity, we only use one sample \( u \) to generate adversarial example for each minibatch. Nevertheless, sufficient samples of \( u \) 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 \( s' \) is generated by the inner maximization, we solve the following problem

\[
\min_w \mathbb{E}_s \left[ \mathcal{L}(f_w(s), y) + \mathbb{E}_u \mathcal{L}(f_{w+u}(s), f_{w+u}(s'))/\lambda \right]. \tag{5}
\]

This optimization problem is challenging due to the entanglement between the deterministic weight \( w \) and the random perturbation \( u \), 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 \( f_{w+u}(s) = g_w(w + u) \), by Taylor expansion of \( g_w(w + u) \) at \( w \), we have

\[
g_w(w + u) = g_w(w) + g_w'(w)^T u + u^T \frac{g_w''(w)}{2} u + O(u(||u||^2)), \tag{6}
\]

where \( g_w'(w) \) and \( g_w''(w) \) are the first and second derivative of the function \( g_w(w) \) versus the weight \( w \), respectively, and \( O_u(||u||^2) \) tends to be zero with respect to ||\( u \)||^2. By such approximation, the gradient computation will be solely done on the
deterministic part $w$, and model smoothing with averaged random variable $u$ is done independently.

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

To further simplify the computation, we minimize an upper bound of Eq. (5) instead. Due to the local convexity of loss landscape, by Jensen’s inequality, we conclude that

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

Therefore, the second term in Eq. (5) can be upper bounded by

$$\mathbb{E}_u \mathcal{L}(f_{w+u}(s), f_{w+u}(s')) \leq \mathcal{L}(g_a(w), g_a(w)) + \mathbb{E}_u \mathcal{L}(g''_a(w)^T u, g''_a(w)^T u) + \frac{1}{2} \mathbb{E}_u \mathcal{L}(u^T g''_a(w) u, u^T g''_a(w) u),$$

where the terms on the right-hand side are referred to as the zeroth, first, and second order Taylor expressions. In doing so, the deterministic and statistical models 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 datasets against various white-box, black-box attacks and Auto Attack with $\ell_2$ and $\ell_\infty$ threat models.

Adversarial Training Setting. We train PreAct ResNet-18 [27] for $\ell_\infty$ and $\ell_2$ threat models on CIFAR-10/100 [40] and SVHN [51]. In addition, we also train WideResNet-34-10 [78] for CIFAR-10/100, VGG16 and MobileNetV2 for CIFAR-10, with $\ell_\infty$ threat model. We adopt the widely used adversarial training setting [60]: for the $\ell_\infty$ threat model, $\epsilon = 8/255$ and step size $2/255$; for the $\ell_2$ 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, batch size 128, weight decay $5 \times 10^{-4}$, 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 $32 \times 32$ random crop with 4-pixel padding and random horizontal flip. We report the highest robustness that ever achieved at different checkpoints for each dataset 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 white-box attacks, black-box attacks and auto attack. For white-box attacks, we adopt PGD-20 [45] and CW-20 [4] (the $\ell_\infty$ 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) [55], 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) [10], 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 [10], APGD-DLR [10], and FAB [9]) and one black-box attack (Square Attack [2]).
We highlight the best results in bold. Accuracy under Auto Attack is sensitive to the regularization hyper-parameter $\eta$. In particular, the accuracy under Auto Attack is sensitive to the regularization hyper-parameter $\eta$. 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.

### 5.1. Sensitivity of hyper-parameter

In our proposed algorithm, the regularization hyper-parameter $\eta$ 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 $L$ 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 $\eta$ increases, the clean accuracy and robust accuracy almost both go up first and then down. In particular, the accuracy under Auto Attack is sensitive to the regularization hyper-parameter $\eta$. 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 SOTA 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 [80] and AWPTRADES [74] as baselines with $1/\lambda = 6$. We also report other state-of-the-art, include AVMixup [42], MART [72], [60], [81], [54], TRADES+LBGAT [11] and [24].

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...
Table 3. Adversarial training across datasets on PreAct ResNet18 with $\ell_2$ threat model. Classification accuracy (%) on clean images and under PGD-20 attack and Auto Attack. The results of our methods are in **bold**.

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
<th>SVHN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean</td>
<td>PGD</td>
<td>AA</td>
</tr>
<tr>
<td>Wu et al. (2020) [74]</td>
<td>87.05</td>
<td>72.08</td>
<td>71.6</td>
</tr>
<tr>
<td>+ Ours ($1_{st}+2_{nd}$)</td>
<td><strong>88.41</strong></td>
<td><strong>72.85</strong></td>
<td><strong>72.0</strong></td>
</tr>
</tbody>
</table>

Table 4. Adversarial training across datasets on PreAct ResNet18 with $\ell_{\infty}$ 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**.

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
<th>SVHN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean</td>
<td>PGD</td>
<td>CW</td>
</tr>
<tr>
<td>Wu et al. (2020) [74]</td>
<td>82.78</td>
<td>61.79</td>
<td>59.42</td>
</tr>
<tr>
<td>+ Ours ($1_{st}+2_{nd}$)</td>
<td><strong>84.27</strong></td>
<td><strong>62.56</strong></td>
<td><strong>60.58</strong></td>
</tr>
</tbody>
</table>

Table 5. Adversarial training across VGG16, MobileNetV2 on CIFAR-10 with $\ell_{\infty}$ 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**.

<table>
<thead>
<tr>
<th>Method</th>
<th>VGG16</th>
<th>MobileNetV2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean</td>
<td>PGD-20</td>
</tr>
<tr>
<td>Zhang et al. (2019) [80] (0.16)</td>
<td>79.78</td>
<td>49.88</td>
</tr>
<tr>
<td>+ Ours ($1_{st}+2_{nd}$)</td>
<td><strong>80.99</strong></td>
<td><strong>50.13</strong></td>
</tr>
<tr>
<td>Wu et al. (2020) [74] (0.16)</td>
<td>78.46</td>
<td>51.19</td>
</tr>
<tr>
<td>+ Ours ($1_{st}+2_{nd}$)</td>
<td><strong>80.31</strong></td>
<td><strong>52.71</strong></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>WideResNet-34-10 Time/Epoch</th>
<th>GPU Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang et al. (2019) [80] (0.16)</td>
<td>1665s</td>
<td></td>
</tr>
<tr>
<td>+ Ours ($1_{st}$)</td>
<td>1665s</td>
<td>12875MB</td>
</tr>
<tr>
<td>+ Ours ($1_{st}+2_{nd}$)</td>
<td>2161s</td>
<td>20629MB</td>
</tr>
<tr>
<td>Wu et al. (2020) [74] (0.16)</td>
<td>16453s</td>
<td></td>
</tr>
<tr>
<td>+ Ours ($1_{st}+2_{nd}$)</td>
<td>2362s</td>
<td>26409MB</td>
</tr>
</tbody>
</table>

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 [24].

5.3. Other empirical results

**Adversarial training with $\ell_2$ threat model.** For the experiments with $\ell_2$ 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.

**Robustness under black-box attacks.** We train PreAct ResNet18 for $\ell_{\infty}$ 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 $\ell_{\infty}$ 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.

**Comparison with SWA from [8].** The best PGD-20 accuracy and AA accuracy on CIFAR-10 for ResNet-18 from [8] are 52.14% and 49.44% respectively, whereas ours are 55.13% and 50.8% in Tab. 1 (with $1_{st}+2_{nd}$ (0.2)).

More empirical results are given in Appendix E.

5.4. Limitations and future work

We use some approximation methods, e.g., Eq. (8), 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 larger datasets.

6. Related work

6.1. 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 $f_w(s)$ and $f_w(s')$. For example, ALP [19, 38] estimates the similar-
ity between \( f_w(s) \) and \( f_w(s') \), and maximizes this similarity in the objective function. MMA [13] proposes each correctly classified instance to leave sufficiently the decision boundary, through making the size of shortest successful perturbation as large as possible. TRADES [80] looks for the adversarial example \( s' \) which can obtain the largest KL divergence between \( f_w(s) \) and \( f_w(s') \), then trains with these adversarial examples. Besides, [84] adopts the similarity between local distributions of natural and adversarial example, [67] measures the similarity of local distributions with Wasserstein distance, and [15, 16, 46, 54] try to optimize with the distribution over a series of adversarial examples for a single input.

In the second group, adversarial examples are preprocessed before being used for training rather than being directly generated by attack algorithms. For example, label smoothing [8, 69] 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 [49]. AVMixup [42, 83] 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, [79] perturbs the local neighborhoods with an unsupervised way to produce adversarial examples. In addition, data augmentation [25, 59] 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 [74], one more maximization is performed to find a weight perturbation based on the generated adversarial examples. The altered weights [12] 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 \( f_{w+u}(s) \) and \( f_{w+u}(s') \) through randomized weights, and therefore, effectively, it considers the optimization problem where the decision boundary is smoothed by these randomized models.

### 6.2. Randomized weights

In the previous work to study generalization or robustness, modeling neural network weights as random variables has been frequently used. For example, [50, 75] estimate the mutual information between random weights and dataset, 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 [41, 47, 56, 64]. In the recent years, PAC-Bayes is also developed to bound the generalization error or robustness error of deep neural networks [18, 21, 35, 52]. In addition, [23] draws random noise into deterministic weights, to measure the mutual information between input dataset 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, from which we find an insight to design a novel adversarial training method which decomposes objective function (parameterized with randomized weights) with Taylor series.

### 6.3. Flatness

The generalization performance of machine learning models is believed to be correlated with the flatness of the loss curve [30, 31], especially for deep neural networks [6, 14, 37, 43, 76]. For instance, in pioneering research [30, 31], the minimum description length (MDL) [61] was used to demonstrate that the flatness of the minimum is a reasonable generalization metric to consider. [39] investigated the cause of the decrease in generalization for large-batch networks and demonstrated that large-batch algorithms tend to converge to sharp minima, resulting in worse generalization. [35] undertook a large-scale empirical study and found that flatness-based measurements correlate with generalization more strongly than weight norms and (margin and optimization)-based measures. [22] introduced Sharpness-Aware Minimization (SAM) to locate parameters that lie in neighborhoods with consistently low loss. Moreover, [8] used SWA [32] in adversarial training to smooth the weights and find flatter minima, and showed that this can improve adversarial generalization.

Different from previous work, this paper uses Taylor expansion of the injected Gaussian to smooth the update of weights and search for flat minima during adversarial training. Our empirical results show the effectiveness of our method to flatten loss landscape and improve both robustness and clean accuracy.

### 7. Conclusion

This work studies the trade-off between robustness and clean accuracy through the lens of randomized weights. Through empirical analysis of loss landscape, algorithmic design (via Taylor expansion) on training optimization, and extensive experiments, we demonstrate that optimizing over randomized weights can consistently outperform the state-of-the-art adversarial training methods not only in clean accuracy but also in robustness.

**Acknowledgment.** GJ is supported by the CAS Project for Young Scientists in Basic Research, Grant No.YSBR-040. This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 956123, and is also supported by the U.K. EPSRC through End-to-End Conceptual Guarding of Neural Architectures [EP/T026995/1].

16454
References


[34] Adel Javanmard, Mahdi Soltanolkotabi, and Hamed Hassani. Precise tradeoffs in adversarial training for linear regression. In COLT, 2020. 1
[69] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In CVPR, 2016. 8
[70] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In ICLR, 2019. 1

16456
[76] Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W. Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. In *Advances in Neural Information Processing Systems*, 2018. 8
[80] Hongyang Zhang, Yao Song, Jianfeng Ji, Yang Xie, Laurens Van Der Maaten, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019. 1, 2, 3, 4, 5, 6, 7, 8
[81] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan S. Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *ICML*, 2020. 1, 6