A Adversarial Attack

Given a natural example $\mathbf{x}_i$ with class label y_i and a DNN model $\mathbf{f}_\mathbf{w}$, the goal of an adversary is to find an adversarial example $\mathbf{x}'_i$ that fools the network to make incorrect predictions while still remains in the ϵ-ball centered at $\mathbf{x}_i$ ($\|\mathbf{x}'_i - \mathbf{x}_i\|_p \leq \epsilon$). A lot of attacking methods have been proposed for the crafting of adversarial examples. Here, we only name a few.

Fast Gradient Sign Method (FGSM) [13]. FGSM perturbs the natural example $\mathbf{x}_i$ for one step by the amount of ϵ along the gradient direction:

$$\mathbf{x}'_i = \mathbf{x}_i + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}_i} \ell(\mathbf{f}_\mathbf{w}(\mathbf{x}_i), y_i)). \tag{13}$$

Projected Gradient Descent (PGD) [27]. PGD perturbs the natural example $\mathbf{x}_i$ for K_1 steps with small step size η_1. After each step of perturbation, PGD projects the adversarial example back onto the ϵ-ball of $\mathbf{x}_i$, if it goes beyond the ϵ-ball:

$$\mathbf{x}_i'^{(k+1)} = \Pi_\epsilon\big(\mathbf{x}_i'^{(k)} + \eta_1 \cdot \text{sign}(\nabla_{\mathbf{x}'_i} \ell(\mathbf{f}_\mathbf{w}(\mathbf{x}_i'^{(k)}), y_i))\big), \tag{14}$$

where $\Pi(\cdot)$ is the projection operation, and $\mathbf{x}_i'^{(k)}$ is the adversarial example at the k-th step. There are also other types of attacks including Jacobian-based Saliency Map Attack (JSMA) [35], Carlini and Wagner (CW) [5], and so on.

B Details for the Weight Loss Landscape Visualization Method

In this section, we first provide the pseudo-code of our proposed visualization method for the weight loss landscape in the adversarial training, and then verify its reliability.

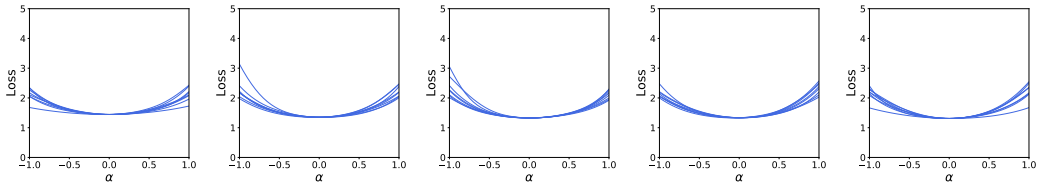
B.1 Pseudo-code of the Visualization Method

As shown in Algorithm 1 for the visualization of weight loss landscape, we firstly sample a random direction $\mathbf{d}$ from a Gaussian distribution. Then, we apply the "filter normalization" technique (Line 3-7) from Li et al. [22] to avoid the scaling effect[§] of DNNs. Next, we calculate the adversarial loss for a series of perturbed weights independently, *i.e.*, $\rho(\mathbf{w} + \alpha\mathbf{d}), \alpha \in \{\alpha_{min}, \cdots, \alpha_{max}\}$. For a given perturbed weights $\mathbf{w} + \alpha\mathbf{d}$, we generate its own adversarial examples using PGD following Madry et al. [27] (Line 9-14). Then, we approximate the adversarial loss of the current perturbed model $f_{\mathbf{w}+\alpha\mathbf{d}}$ by the cross-entropy loss on these on-the-fly generated adversarial examples (Line 15). Finally, we plot the weight loss landscape (Line 17).

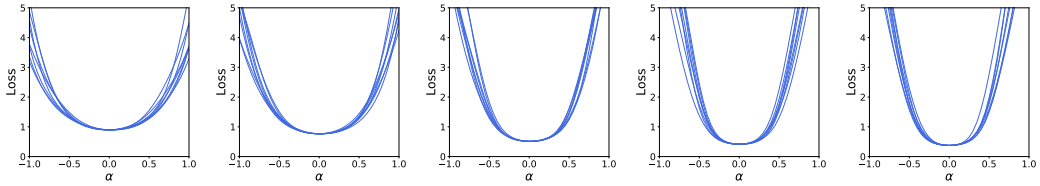
Algorithm 1 Visualization of Weight Loss Landscape

1: **Input:** Network $\mathbf{f}_\mathbf{w}$ with L-layer (F_l filters in the l-th layer), training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, PGD step size η_1, PGD step number K_1, the scalar parameter $\alpha \in [\alpha_{min}, \alpha_{max}]$.
2: Sample a random direction $\mathbf{d} \sim \mathcal{N}(0, 1)$
3: **for** $l = 1, \ldots, L$ **do**
4: **for** $j = 1, \ldots, F_l$ **do**
5: $\mathbf{d}_{l,j} \leftarrow \frac{\mathbf{d}_{l,j}}{\|\mathbf{d}_{l,j}\|_F} \|\mathbf{w}_{l,j}\|_F$
6: **end for**
7: **end for**
8: **for** $\alpha = \alpha_{min}, \cdots, \alpha_{max}$ **do**
9: **for** $i = 1, \ldots, n$ (in parallel) **do**
10: $\mathbf{x}'_i \leftarrow \mathbf{x}_i + \epsilon\delta$, where $\delta \sim$ Uniform$(-1, 1)$
11: **for** $k = 1, \cdots, K_1$ **do**
12: $\mathbf{x}'_i \leftarrow \Pi_\epsilon\big(\mathbf{x}'_i + \eta_1 \text{sign}(\nabla_{\mathbf{x}'_i} \ell(\mathbf{f}_{\mathbf{w}+\alpha\mathbf{d}}(\mathbf{x}'_i), y_i))\big)$
13: **end for**
14: **end for**
15: $\rho(\mathbf{w} + \alpha\mathbf{d}) \leftarrow \frac{1}{n} \sum_{i=1}^n \ell(f_{\mathbf{w}+\alpha\mathbf{d}}(\mathbf{x}'_i), y_i)$
16: **end for**
17: Plot $(\alpha, \rho(\mathbf{w} + \alpha\mathbf{d})), \forall \alpha \in [\alpha_{min}, \alpha_{max}]$

[§] In DNNs with ReLU activation, the network remains unchanged if we multiply the weights in one layer by 10, and divide by 10 at the next layer.

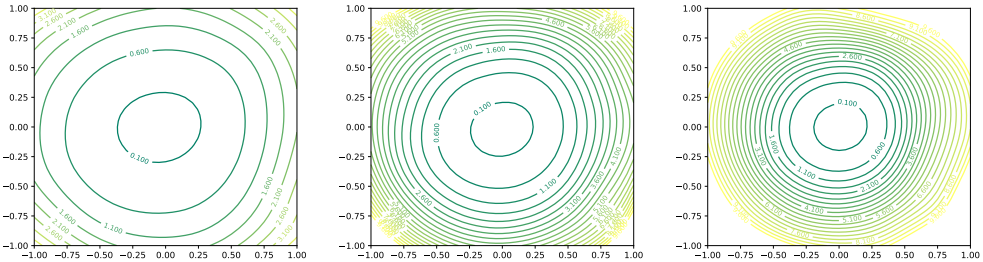


(a) Epoch 20 (b) Epoch 40 (c) Epoch 60 (d) Epoch 80 (e) Epoch 100



(f) Epoch 120 (g) Epoch 140 (h) Epoch 160 (i) Epoch 180 (j) Epoch 200

Figure 5: Repeatability of the 1-D visualization along 10 different random directions.



(a) Epoch 100 (b) Epoch 140 (c) Epoch 200

Figure 6: 2-D visualization of weight loss landscape at the different epoch checkpoints.

B.2 Reliability of the 1-D Visualization

To improve the trustworthiness of our results, we check the reliability of our visualization method from two perspectives: 1) repeatability; 2) comparisons to 2-D visualization.

Repeatability. We first investigate whether different random directions produce dramatically different plots. We show the weight loss landscape of PreAct ResNet-18 during vanilla adversarial training along 10 randomly selected directions in Figure 5. We find the plots of the same checkpoint are very close in shape, which indicates the stability of our visualization method.

Comparisons to 2-D Visualization. Next, we explore whether 1-D visualization obtains similar results to 2-D visualization (a much time-consuming method). The 2-D visualization introduces an extra random filter-normalized direction $\mathbf{e}$, and plots $g(\alpha, \beta) = \rho(\mathbf{w} + \alpha\mathbf{d} + \beta\mathbf{e})$. Different from the standard training scenario where near-zero loss on the training set can be always achieved, the adversarial loss on the training set is usually larger than zero, *i.e.*, the center point $g(0, 0)$ in the weight loss landscape is often larger than zero, which hampers the comparison on the flatness of weight loss landscape. Therefore, we visualize the relative weight loss landscape $|g(\alpha, \beta) - g(0, 0)|$ instead. Figure 6 presents the 2-D visualization of the same model as Figure 5 at the 100-th, 140-th and 200-th epoch checkpoint. We can see that the weight loss landscape is flatter at the 100-th epoch and sharper at the 200-th epoch, which is compatible to the findings from the 1-D visualization. For time costs, it consumes ~ 4 days for the 2-D visualization of a single adversarially trained PreAct ResNet-18 using one GeForce RTX 2080Ti, while it is ~ 2 hours for the 1-D visualization of the same model. Thus, we adopt the 1-D visualization in most cases.

From the above experiments, we can conclude that our 1-D visualization method can characterize the property of the high-dimensional weight loss landscape reliably and efficiently.

C More Evidence for the Connection of Weight Loss Landscape and Robust Generalization Gap

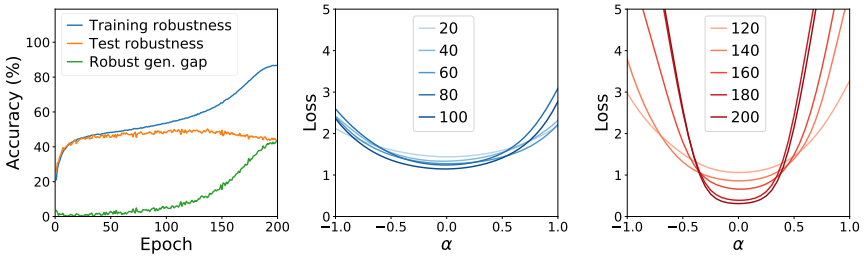
In this section, we provide more empricial evidence to identify the connection of the weight loss landscape and the robust generalization gap across learning rate schedules, model architectures, datasets, and threat models.

C.1 The Connection across Learning Rate Schedules

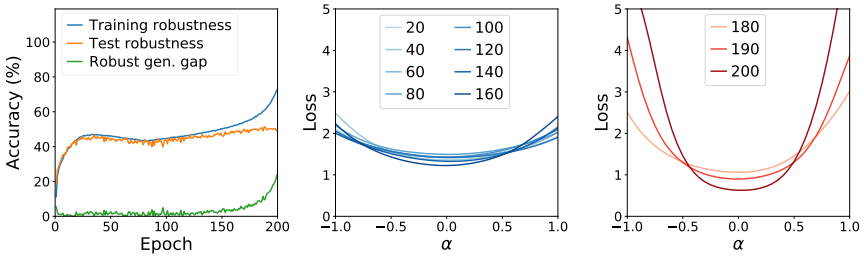
To investigate whether the learning rate schedule affects the connection of weight loss landscape and robust generalization gap, we test another two commonly used learning rate schedules:

- Cosine schedule [24]: We decrease the learning rate lr using the cosine function from 0.1 to 0 over 200 epochs, *i.e.*, $lr = 0.05(\cos(\pi t/200) + 1))$ at the t-th epoch [6];
- Cyclic schedule [44]: We increase lr linearly from 0 to some maximum (0.2 at the 80-th epoch), and then decrease it linearly to 0 until 200-th epoch [55].

We adversarially train PreAct ResNet-18 with different learning rate schedules using the same experimental settings in Section 3. The learning curves are shown on the left column in Figure 7, where the whole training process can be split into two stages: the early stage with small robust generalization gap ($\leq 10\%$) and the late stage with large robust generalization gap ($> 10\%$). In Figure 7, the weight loss landscapes of checkpoints at different epochs from the early stage (blue curves) are on the middle column, while the weight loss landscapes from the late stage (red curves) are on the right column. Due to the different learning rate schedules, the robust generalization gap becomes large at different epochs. The cosine schedule enlarges the robust generalization gap after the 120-th epoch with $lr < 0.34$. The weight loss landscape becomes sharper correspondingly. The cyclic schedule starts to significantly enlarge the gap much later, almost after the 175-th epoch with $lr < 0.16$. Meanwhile, the weight loss landscape also becomes sharp much later. Generally, no matter which epoch and learning rate it is, we can always find that the weight loss landscape becomes sharper immediately after the robust generalization gap increases, which implies a clear correlation between weight loss landscape and robust generalization gap.



(a) Cosine schedule



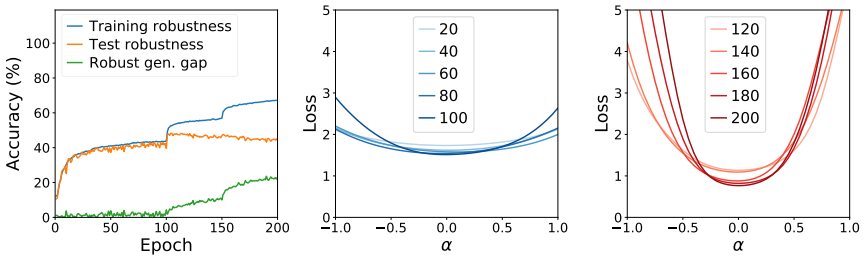
(b) Cyclic schedule

Figure 7: The relationship between weight loss landscape and robust generalization gap across learning rate schedules (cosine and cyclic) with PreAct ResNet-18 on CIFAR-10 under L_∞ attack.

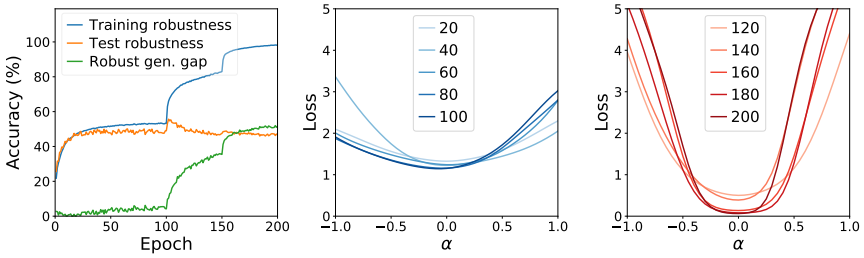
C.2 The Connection across Model Architectures

The previous experiments are all based on PreAct ResNet-18. Here we additionally conduct experiments with VGG-19 [43] and WideResNet-34-10 [60] to verify the connection across network

architectures. The same experimental settings as Section 3 are adopted and the results are shown in Figure 8. They all behave similarly: once the robust generalization gap increases (after the first learning rate decay), the weight loss landscape becomes sharper. This indicates that the connection of weight loss landscape and robust generalization gap still exists across architectures.

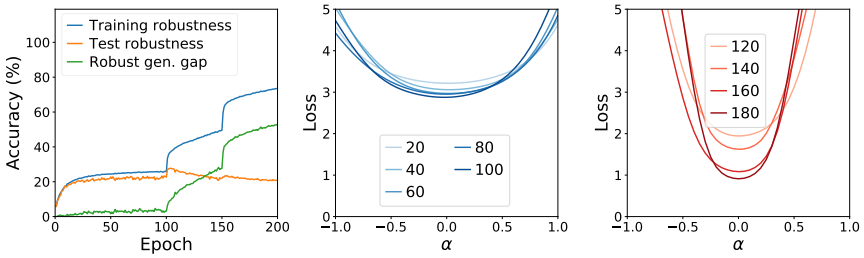


(a) VGG-19

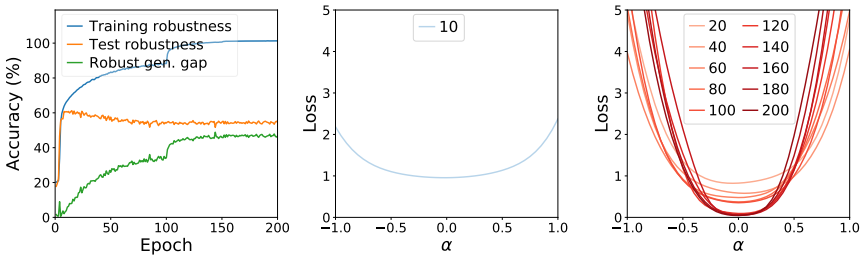


(b) WideResNet-34-10

Figure 8: The relationship between weight loss landscape and robust generalization gap across model architectures (VGG-19 and WideResNet-34-10) on CIFAR-10 using piece-wise learning rate schedule and L_∞ attack.



(a) CIFAR-100



(b) SVHN

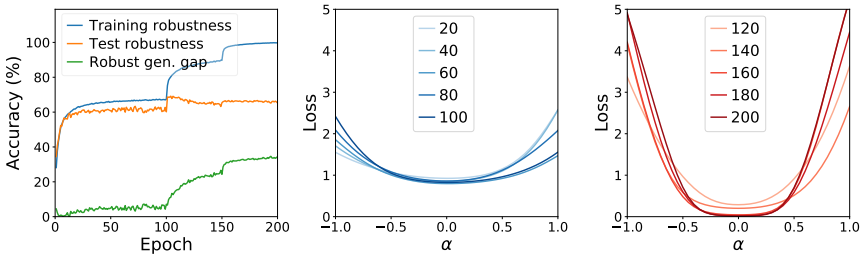
Figure 9: The relationship between weight loss landscape and robust generalization gap across datasets (CIFAR-100 and SVHN) with PreAct ResNet-18 using piece-wise learning rate schedule and L_∞ attack.

C.3 The Connection across Datasets

Next, we demonstrate that the connection is an universal phenomenon across datasets on CIFAR-100 [21] and SVHN [32]. We adversarially train PreAct ResNet-18 on different datasets with the same settings as Section 3. The results are shown in Figure 9. The phenomenon on CIFAR-100 is similar to that on CIFAR-10, *i.e.*, after the first learning rate decay, the robust generalization gap becomes larger and the weight loss landscape becomes sharper as well. For SVHN, the robust generalization gap increases significantly even earlier: it almost achieves its highest robustness around 10-th epoch, and starts overfitting. Meanwhile, the weight loss landscape also keeps flat at 10-th epoch and starts to become sharper. In conclusion, the strong connection of weight loss landscape and robust generalization gap is universal across datasets.

C.4 The Connection on L_2 Threat Model

To further explore the universality of the connection, we additionally conduct experiments on L_2 threat model in Figure 10. The other experimental settings are the same as Section 3. Under the L_2 threat model, the connection still exists: once the robust generalization gap increases (after the first learning rate decay), the weight loss landscape becomes sharper.



(a) L_2 threat model

Figure 10: The relationship between weight loss landscape and robust generalization gap with PreAct ResNet-18 on CIFAR-10 using piece-wise learning rate schedule and L_2 attack.

D Algorithms for the AWP-based Defense

In this section, we first provide the pseudo-code of the AWP-based vanilla advesraial trainig (AT-AWP), and then describe how to satisfy the constraint of the perturbation size in Eq. (8) via the weight update in Eq. (10) and the extensions to other AT variants (TRADES, MART, and RST).

D.1 Pseudo-code of AT-AWP

Algorithm 2 AT-AWP

Input: Network $\mathbf{f_w}$, batch size m, learning rate η_3, PGD step size η_1, PGD steps K_1, AWP constraint γ, AWP step size η_2, AWP steps K_2, alternate iteration A.
Output: Robust model $\mathbf{f_w}$.
repeat
  Read mini-batch $B = \{\mathbf{x}_1, ..., \mathbf{x}_m\}$ from training set
  for $a = 0$ to $A - 1$ **do**
    for $i = 1, \dots, m$ (in parallel) **do**
      $\mathbf{x}_i' \leftarrow \mathbf{x}_i + \epsilon\delta$, where $\delta \sim \text{Uniform}(-1, 1)$
      for $k = 1, \cdots, K_1$ **do**
        $\mathbf{x}_i' \leftarrow \Pi_\epsilon\big(\mathbf{x}_i' + \eta_1 \text{sign}(\nabla_{\mathbf{x}_i'} \ell(\mathbf{f}_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i'), y_i))\big)$
      end for
    end for
    for $k = 1, \dots, K_2$ **do**
      $\mathbf{v} \leftarrow \Pi_\gamma\big(\mathbf{v} + \eta_2 \frac{\nabla_{\mathbf{v}} \frac{1}{m} \sum_i \ell(f_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i'), y_i)}{\|\nabla_{\mathbf{v}} \frac{1}{m} \sum_i \ell(\mathbf{f}_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i'), y_i)\|} \|\mathbf{w}\|\big)$
    end for
  end for
  $\mathbf{w} \leftarrow (\mathbf{w} + \mathbf{v}) - \eta_3 \nabla_{\mathbf{w}+\mathbf{v}} \frac{1}{m} \sum_i \ell(f_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i', y_i)) - \mathbf{v}$
until training converged

D.2 Details for the Weight Update under the Constraint

Recall that we restrict the weight perturbation $\mathbf{v}_l$ in the l-th layer using its relative size to the corresponding weight $\mathbf{w}_l$, *i.e.*, $\|\mathbf{v}_l\| \leq \gamma\|\mathbf{w}_l\|$. To implement this restriction on weight perturbation, we apply a layer-wise projection operation, *i.e.*, once the weight perturbation is out of the ball, we project it back onto the surface of the ball. We have the following layer-wise projection operator,

$$\Pi_\gamma(\mathbf{v}) = \begin{cases} \gamma\dfrac{\|\mathbf{w}_l\|}{\|\mathbf{v}_l\|}\mathbf{v}_l, & \text{if } \|\mathbf{v}_l\| > \gamma\|\mathbf{w}_l\|, \forall l \in \{1, \cdots, L\} \\ \mathbf{v}_l, & \text{if } \|\mathbf{v}_l\| \leq \gamma\|\mathbf{w}_l\|, \forall l \in \{1, \cdots, L\}. \end{cases} \tag{15}$$

By default, we set $\eta_2 = \frac{\gamma}{A \cdot K_2}$ (the notations refer to Algorithm 2).

D.3 Extensions of AWP to Other Adversarial Training Methods

Our proposed AWP is a general method and can be easily extended to other well-recognized adversarial training variants including TRADES, MART and RST where the only difference is the method-specific adversarial loss in Eq. (1).

Specifically, for AWP-based TRADES (TRADES-AWP), we also first generate adversarial examples following TRADES method,

$$\mathbf{x}_i' \leftarrow \Pi_\epsilon\big(\mathbf{x}_i' + \eta_1 \text{sign}(\nabla_{\mathbf{x}_i'} \text{KL}(\mathbf{f}_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i) \| \mathbf{f}_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i')))\big), \tag{16}$$

and then the AWP $\mathbf{v}$ and the DNN parameter $\mathbf{w}$ of TRADES are updated similarly following Eq. (10) and Eq. (11) respectively, where $\ell(\mathbf{f}_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i'), y_i)$ is TRADES-specific as $\text{CE}(\mathbf{f}_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i), y_i) + \beta \cdot \text{KL}(\mathbf{f}_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i) \| f_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i'))$.

Similarly, for AWP-based MART (MART-AWP), we generate adversarial examples following MART method,

$$\mathbf{x}_i' \leftarrow \Pi_\epsilon\big(\mathbf{x}_i' + \eta_1 \text{sign}(\nabla_{\mathbf{x}_i'} \ell(\mathbf{f}_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i'), y_i))\big), \tag{17}$$

and then update the AWP $\mathbf{v}$ follow Eq. (10). Next, the DNN parameter $\mathbf{w}$ of MART is updated using Eq. (11), where $\ell(\mathbf{f}_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i'), y_i)$ is MART-specific loss as $\text{BCE}(\mathbf{f}_{\mathbf{w}}(\mathbf{x}_i'), y_i) + \lambda \cdot \text{KL}(\mathbf{f}_{\mathbf{w}}(\mathbf{x}_i) \| \mathbf{f}_{\mathbf{w}}(\mathbf{x}_i')) \cdot (1 - [\mathbf{f}_{\mathbf{w}}(\mathbf{x}_i)]_{y_i})$.

RST, as an SSL-based method, first generates pseudo labels for unlabeled data, and then adversarially train DNNs using TRADES loss on the new dataset which consists of labeled data and unlabeled data with pseudo labels. Thus, we can incorporate AWP into RST just like TRADES-AWP.

E More Results for Section 4.3: A Case Study on Vanilla AT and AT-AWP

Following Section 4.3, we provide the complete results (test robustness and natural accuracy) of AT and AT-AWP in Table 3. Under L_2 threat model, AWP improves both the robustness and natural accuracy on all datasets. While under L_∞ threat model, AWP improves the robustness on condition of sacrificing the natural accuracy on CIFAR-10 and CIFAR-100. This maybe because natural images (CIFAR) are more complicated than color digits (SVHN). However, for robustness, AWP demonstrates a general behaviour that consistently improves the best and last robustness by a recognizable gain across datasets and threat models.

F More Experiments on Comparison to Other Regularization Techniques

Following Section 5.3, here we provide the complete results (test robustness and natural accuracy) of AWP and several regularization techniques under both L_∞ and L_2 threat models on CIFAR-10. Under the L_∞ threat model, we follow the best hyper-parameters tuned in Rice et al. [40]: $\lambda = 5 \times 10^{-6}/5 \times 10^{-3}$ for L_1/L_2 regularization respectively, patch length 14 for cutout, and $\alpha = 1.4$ for mixup. Under the L_2 threat model, we use the same hyper-parameters since we find that they almost have the same trends after parameter tuning. For AWP, we all set $\gamma = 5 \times 10^{-3}$. We use the same training settings as Section 5.2 and the test attack is PGD-20. We show test robustness and test natural accuracy in Table 4 (L_∞ threat model) and Table 5 (L_2 threat model). We find AWP indeed improves the test robustness of both the best checkpoint and the last checkpoint by a notable margin. Besides, we visualize the learning curves in Figure 11. We find that AWP can constantly

Table 3: Performance (%) of AT and AT-AWP on PreAct ResNet-18 across different datasets and threat models over 5 random runs.

Dataset	Norm	Method	Robustness		Natural accuracy	
			Best	Last	Best	Last
SVHN	L_∞	AT	53.36 ± 0.09	44.49 ± 0.27	92.18 ± 0.15	89.85 ± 0.33
		AT-AWP	$\mathbf{59.12 \pm 0.26}$	$\mathbf{55.87 \pm 0.39}$	$\mathbf{93.85 \pm 0.11}$	$\mathbf{92.59 \pm 0.52}$
	L_2	AT	66.87 ± 0.25	65.03 ± 0.24	93.69 ± 0.12	93.25 ± 0.28
		AT-AWP	$\mathbf{72.57 \pm 0.40}$	$\mathbf{67.73 \pm 0.21}$	$\mathbf{95.95 \pm 0.36}$	$\mathbf{95.22 \pm 0.27}$
CIFAR-10	L_∞	AT	52.79 ± 0.21	44.44 ± 0.39	$\mathbf{85.57 \pm 0.16}$	$\mathbf{84.56 \pm 0.19}$
		AT-AWP	$\mathbf{55.39 \pm 0.39}$	$\mathbf{54.73 \pm 0.16}$	82.00 ± 0.19	81.11 ± 0.39
	L_2	AT	69.15 ± 0.13	65.93 ± 0.35	89.57 ± 0.09	88.96 ± 0.18
		AT-AWP	$\mathbf{72.69 \pm 0.19}$	$\mathbf{72.08 \pm 0.39}$	$\mathbf{90.18 \pm 0.31}$	$\mathbf{89.71 \pm 0.17}$
CIFAR-100	L_∞	AT	27.22 ± 0.16	20.82 ± 0.20	$\mathbf{56.33 \pm 0.23}$	$\mathbf{54.61 \pm 0.33}$
		AT-AWP	$\mathbf{30.71 \pm 0.25}$	$\mathbf{30.28 \pm 0.30}$	54.19 ± 0.39	54.39 ± 0.29
	L_2	AT	41.33 ± 0.09	35.27 ± 0.29	62.65 ± 0.11	60.50 ± 0.17
		AT-AWP	$\mathbf{45.60 \pm 0.23}$	$\mathbf{44.66 \pm 0.22}$	$\mathbf{65.07 \pm 0.31}$	$\mathbf{64.40 \pm 0.41}$

improve the robustness under both L_∞ and L_2 threat models throughout the entire training process, which demonstrates the superiority of AWP over other regularization methods.

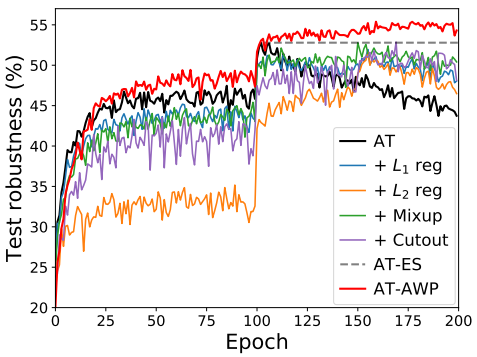
In addition, we find AWP sacrifices the natural accuracy throughout the whole training on CIFAR-10 under L_∞ threat model, which indicates AWP is still affected by the trade-off between robustness and natural accuracy [62]. For L_2 threat model on CIFAR-10, AWP just has similar natural accuracy to vanilla AT. This is because L_2 threat model ($\epsilon = 128/255$) is easier than L_∞ threat model and many techniques have the similar natural accuracy.

Table 4: Performance (%) of AT and AT with other regularization techniques on CIFAR-10 using PreAct ResNet-18 under L_∞ threat model ($\epsilon = 8/288$) over 5 random runs.

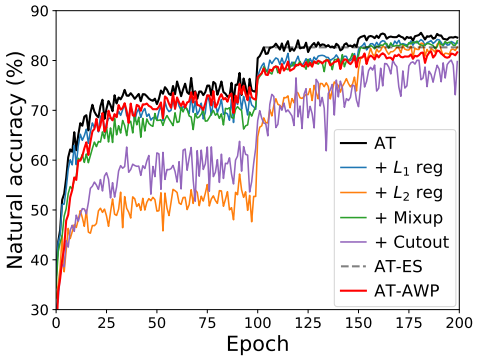
Method	Robustness		Natural accuracy	
	Best	Last	Best	Last
AT	52.79 ± 0.21	44.44 ± 0.39	$\mathbf{85.57 \pm 0.16}$	$\mathbf{84.56 \pm 0.19}$
+ L_1 regularization	51.95 ± 0.51	48.76 ± 0.61	82.77 ± 0.37	83.42 ± 0.26
+ L_2 regularization	51.60 ± 0.41	47.37 ± 0.52	81.05 ± 0.44	81.97 ± 0.50
+ Cutout	52.78 ± 0.14	50.37 ± 0.41	80.99 ± 0.19	83.46 ± 0.33
+ Mixup	52.87 ± 0.47	49.76 ± 0.81	78.76 ± 0.61	78.50 ± 1.21
AT-AWP	$\mathbf{55.39 \pm 0.39}$	$\mathbf{54.73 \pm 0.16}$	82.00 ± 0.19	81.11 ± 0.39

Table 5: Performance (%) of AT and AT with other regularization techniques on CIFAR-10 using PreAct ResNet-18 under L_2 threat model ($\epsilon = 128/255$) over 5 random runs.

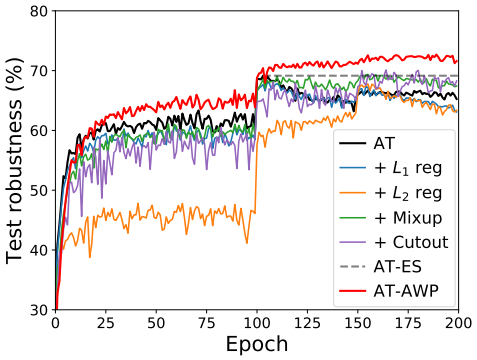
Method	Robustness		Natural accuracy	
	Best	Last	Best	Last
AT	69.15 ± 0.13	65.93 ± 0.35	89.57 ± 0.09	88.96 ± 0.18
+ L_1 regularization	67.99 ± 0.27	63.75 ± 0.40	88.04 ± 0.19	88.49 ± 0.27
+ L_2 regularization	67.78 ± 0.33	63.62 ± 0.46	88.57 ± 0.14	87.75 ± 0.23
+ Cutout	69.38 ± 0.27	67.68 ± 0.30	88.36 ± 0.31	88.01 ± 0.26
+ Mixup	70.11 ± 0.50	68.20 ± 0.46	87.29 ± 0.71	86.91 ± 0.94
AT-AWP	$\mathbf{72.69 \pm 0.19}$	$\mathbf{72.08 \pm 0.39}$	$\mathbf{90.18 \pm 0.31}$	$\mathbf{89.71 \pm 0.17}$



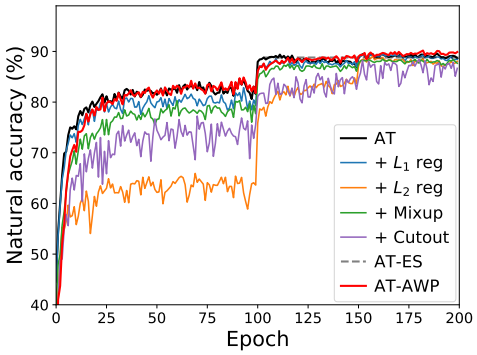
(a) Test robustness under ℓ_∞ threat model



(b) Natural accuracy under ℓ_∞ threat model



(c) Test robustness under ℓ_2 threat model



(d) Natural accuracy under ℓ_2 threat model

Figure 11: Performance (%) on the test set of CIFAR-10 during the training for AT, AT-AWP, and AT with other regularization techniques.