Appendix: Exploring the Relationship between Architectural Design and Adversarially Robust Generalization

Aishan Liu[1], Shiyu Tang[1], Siyuan Liang[2], Ruihao Gong[1,6], Boxi Wu[3],
Xianglong Liu[1,4,5], Dacheng Tao[7]
[1]Beihang University, [2]Chinese Academy of Sciences, [3]Zhejiang University,
[4]Zhongguancun Laboratory, [5]Hefei Comprehensive National Science Center,
[6]SenseTime, [7]JD Explore Academy

1. Theoretical Proof

We prove Theorem 1 by unpacking the upper bound of a simple transformer, which is similar to a two-layer neural network in the paper [4]. The simple transformer consists of two fundamental blocks $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}_1\phi(\boldsymbol{A}_1\boldsymbol{x})$, where $\boldsymbol{A}_1$ and $\boldsymbol{W}_1$ respectively denotes the Mutil-head Self-Attention layer module and the Multi-Layer Perception layer module, which are connected with a non-linear activation function ϕ. Generally speaking, a Mutil-head Self-Attention $\boldsymbol{A}_1 = \text{SoftMax}(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{d} + \boldsymbol{B})\boldsymbol{V}$ is non-linear layer due to the existence of the non-linear activation function SoftMax.

We can bound the ℓ_1 norm of the gradient of the transformer in the adversarial training, denoted as $||f'_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta})||_1$. Due to the existence of non-linear activation functions, the activation derivatives $\phi'(\boldsymbol{A}'_1(\boldsymbol{x}+\boldsymbol{\delta}))$ will depend on the adversarial example if we apply the chain rule. In some cases, the Mutil-head Self-Attention layer can be replaced with the approximate linear layer to settle the computational complexity brought from the dimension of the input image [3,5,7]. To remove the dependence on the adversarial example, we can optimize over possible activation derivatives s for the activation layer because $\forall z \in \mathbb{R}, \phi'(z) \in [0,1]^m$. Thus, the upper bound of the ℓ_1 norm of gradient from the [4] remains unchanged:

$$\begin{aligned} ||f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta})||_1 &= ||\boldsymbol{A}_1^\top \text{diag}(\boldsymbol{W}_1)\phi'(\boldsymbol{A}'_1(\boldsymbol{x}+\boldsymbol{\delta}))||_1 \\ &\le \max_{s\in[0,1]^m} ||\boldsymbol{A}_1^\top \text{diag}(\boldsymbol{W}_1)s||_1 \\ &= \max_{s\in[0,1]^m, t\in[-1,1]^d} t^\top \boldsymbol{A}_1^\top \text{diag}(\boldsymbol{W}_1)s, \end{aligned} \tag{1}$$

where the identity $||z||_1 = \max_{t\in[-1,1]^d} t^\top z$.

Based on satisfying the Eq. 1, we can directly use the generalization upper bound in the multi-class model, *i.e.*, the SDP upper bound [4]. Then the generalization to multi-class is:

$$f^{ij}_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}) \le f^{ij}_{SDP}(\boldsymbol{x}) + \frac{\epsilon}{4}\max_{\boldsymbol{P}\succeq 0, \text{diag}\le 1}\langle Q^{ij}(\boldsymbol{W}_1, \boldsymbol{A}_1), \boldsymbol{P}\rangle, \tag{2}$$

where f^{ij} is the margin between class i with class j.

Thus, we can easily prove Theorem 1 according to the generalization bound for surrogate adversarial loss [8]. We define $M_{\mathcal{F}} = \{(\boldsymbol{x}, \boldsymbol{y}) \mapsto M(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}) : f_{\boldsymbol{\theta}} \in \mathcal{F}\}$ and M denotes the margin loss, then we have:

$$\begin{aligned} &\mathbb{P}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{S}}\{\exists \boldsymbol{\delta} \in \mathbb{B}(\epsilon), \quad s.t. \quad \boldsymbol{y} \neq \arg\max[f_{\boldsymbol{w}}(\boldsymbol{x}+\boldsymbol{\delta})]_{\boldsymbol{y}'}\} \\ &\le \frac{1}{\gamma}(R_{\mathcal{S}}(M_{\mathcal{F}}) + \\ &\frac{\epsilon}{2n}\mathbb{E}_{\boldsymbol{\sigma}}[\sup_{f_{\boldsymbol{\theta}}\in\mathcal{F}}\sum_{i=1}^{n}\sigma_i \max_{k\in[K], z=\pm 1}\max_{\boldsymbol{P}\succeq 0, \text{diag}\le 1}\langle zQ(\boldsymbol{w}_{1,k}, \boldsymbol{A}_1), \boldsymbol{P}\rangle]. \end{aligned} \tag{3}$$

Since we have $\| \boldsymbol{A}_1 \|_p \le s_1, \| \boldsymbol{W}_1 \|_p \le s_2, \| \boldsymbol{A}_1 \|_1 \le b_1, \| \boldsymbol{W}_{1,1} \|_1 \le b_2$, we can apply the Rademacher complexity and obtain:

$$\begin{aligned} R_{\mathcal{S}}(M_{\mathcal{F}} &\le \frac{4}{n^{\frac{3}{2}}} \\ &+ \frac{60\log(n)\log(2d_{\max})}{n} s_1 s_2 ((\frac{b_1}{s_1})^{2/3} + \\ &(\frac{b_2}{s_2})^{2/3})^{3/2} \| \boldsymbol{X} \|_F . \end{aligned} \tag{4}$$

Next, based on Khintchine's inequality, Holder's inequality, the definition of $Q(,)$, and $||\boldsymbol{W}_1||_\infty \le b_2$, we can obtain the upper bound of the second term in Eq. 3 as follows:

$$\begin{aligned} &\frac{\epsilon}{2n}\mathbb{E}_{\boldsymbol{\sigma}}[\sup_{f_{\boldsymbol{\theta}}\in\mathcal{F}}\sum_{i=1}^{n}\sigma_i \max_{k\in[K], z=\pm 1} \\ &\max_{\boldsymbol{P}\succeq 0, \text{diag}\le 1}\langle zQ(\boldsymbol{w}_{1,k}, \boldsymbol{A}_1), \boldsymbol{P} \\ &\rangle] \le \frac{2\epsilon b_1 b_2}{\sqrt{n}}. \end{aligned} \tag{5}$$

Bringing Eq. 4 and Eq. 5 into Eq. 3, Theorem 1 is proved.

The first three authors contribute equally. Corresponding author: Xianglong Liu, xlliu@buaa.edu.cn.

2. Detailed Experiment Setup

We conduct comprehensive experiments to verify our conclusions. All of our experiments are running on Nvidia GTX 1080Ti GPU for about 0.4 GPU years since most of the experiments contain adversarial training which is time-consuming. We also release our code for experiments, which is under Apache License 2.0. All used datasets and code in the experiment are downloaded from the official or other open-source releases. We report the detailed experiment setup as follows.

2.1. Model architectures

Specifically, on ImageNette, for CNNs we use ResNet-50, ResNeXt-50-32x4d, WideResNet-50-2, PreActResNet-50, VGG-16, and DenseNet-201; for Transformers we use ViT-S/16, Mixer-S/16, PoolFormer-S24, Swin-T, ViTAE-S, CCT-14/7x2, MobileViT-S, CPVT-S, BoTNet-50, CeiT-T, CoAtNet-1, CvT-13, LeViT-256, and PVTv2-B2. On CIFAR-10, since many architectures do not provide official implementation on CIFAR-10, we mainly use the unofficial implementation code in github .

2.2. Training settings

To avoid the influence of different training techniques (*e.g.*, data augmentation, weight average) on adversarially robust generalization, we use the aligned training settings for all models in adversarial training. Specifically, on CIFAR-10 dataset, we use AutoAugment (CIFAR-10 policy) as data augmentation for all models; for the optimizer, we use AdamW with weight decay=0.05; for the scheduler, we use cosine scheduler with learning rate warm-up in the first 8 epochs; we set lr=0.001 for all CNNs and lr=0.00025 for some of the Transformers; for batch size and epochs we set bs=128 and epoch=100 for all models; for adversarial training, we set ϵ=8/255, steps=15 and step size=$\frac{\epsilon}{10}$. On ImageNette dataset, we use standard ImageNet data augmentation which contains random resized crop, random horizontal flip, and color jitter since strong augmentation (*e.g.*, AutoAugment, Mixup) on ImageNet would make it hard for adversarial training to converge [1, 6]; for the optimizer, we also use AdamW with weight decay=0.05; for the scheduler, we also use cosine scheduler with warmup; for all models, we set batch size=128 and training epoch=100; for adversarial training, we set ϵ=8/255, steps=20 and step size=$\frac{\epsilon}{15}$.

3. Additional Experiment Results

In this section, we provide more evaluation results.

https : / / github . com / pprp / pytorch-cifar-model-zoo

3.1. Adversarially robust generalization evaluation on other datasets

ImageNette. We first report the adversarially robust generalization results for 20 architectures on ImageNette dataset, as shown in Table 1. Although the results for some specific architectures are slightly different, the main conclusions keep consistent with the results on CIFAR-10. Generally speaking, Transformers show better robust generalization (*e.g.*, PVTv2), while CNNs (*e.g.*, ResNet and its variants) also tend to overfit the trained ℓ_∞ adversarial noises and show bad robust generalization on PGD-ℓ_2 and PGD-ℓ_1 adversarial noises.

ImageNet. We conduct PGD-ℓ_∞ adversarial training (ϵ=4/255) on ImageNet, and the selected models are ViT and ResNet-101. We evaluate the adversarially robust generalization of the two adversarially-trained models under different attack norms and attack methods, including PGD-ℓ_1, PGD-ℓ_2, PGD-ℓ_∞, and AA-ℓ_∞. The experimental settings are kept the same as those in the main paper. In Table 2, we found that the generalization trend of model adversarial training is not much different from other datasets. Meanwhile, as the number of samples in the data set increases, though the Transformer still has an advantage over CNN on adversarially robust generalization, the advantage gradually shrinks.

CIFAR-100. We then conduct PGD-ℓ_∞ adversarial training (ϵ=8/255) on CIFAR-100. The experimental settings are similar to our main experiment in the main paper. Specifically, we choose some representative architectures including PVTv2, CoAtNet, ViT, ResNet, WideResNet, and CvT. As shown in Table 3, the adversarially robust generalization on CIFAR-100 shows a similar tendency with those on CIFAR-10 and ImageNette. PVTv2 and CoAtNet have the best adversarially robust generalization ability and CNNs (ResNet and WideResNet) also tend to overfit the trained PGD-ℓ_∞ noises and have bad robust generalization.

3.2. Adversarially robust generalization using TRADES

Similar to the standard PGD-ℓ_∞ adversarial training experiment, we also conduct adversarial training on 20 architectures using TRADES, which is one of the state-of-the-art adversarial training techniques. Specifically, for all models, we set ϵ=8/255, steps=15, step size=$\frac{\epsilon}{10}$, and we set the trade-off regularization parameter beta=6.0. Results on CIFAR-10 are shown in Table 4. We can observe that compared to standard PGD-ℓ_∞ adversarial training, TRADES indeed largely improve the robust generalization for most of the architectures (*e.g.*, 14.32 $\rightarrow$ 21.62 for the worst-case robust accuracy of ResNet). As for the robust generalization of different architectures, the evaluation results under TRADES is much similar to those under standard PGD-ℓ_∞ training, indicating that the adversarially robust generaliza-

Table 1. Evaluation results for all model architectures on ImageNette dataset. We set $\epsilon = 8/255$ for PGD-ℓ_∞ adversarial training, and $\epsilon = 8/255, 8.0, 1600.0$ for PGD-ℓ_∞, ℓ_2 and ℓ_1 testing. Results are ranked by the worst-case robust accuracy.

			PGD-ℓ_∞ Adversarial Training					
Architecture	**Params (M)**	**Vanilla Acc**	**Clean Acc**	**PGD-ℓ_∞**	**AA-ℓ_∞**	**PGD-ℓ_2**	**PGD-ℓ_1**	**Worst-case Acc**
PVTv2	23.11	89.73	85.41	63.63	58.53	57.05	56.39	52.68
ViTAE	22.81	90.04	88.59	63.58	58.03	54.34	53.23	50.31
CvT	17.58	87.09	83.33	59.08	53.42	53.73	54.16	48.60
VGG	134.31	90.62	85.67	64.19	58.03	51.79	46.02	44.68
CoAtNet	39.41	92.60	89.62	64.08	57.84	50.28	45.94	43.89
BoTNet	18.92	91.36	86.41	61.13	53.39	48.49	45.94	42.73
CeiT	5.59	89.88	85.99	59.11	52.63	47.86	44.97	42.52
WideResNet	66.85	88.83	76.11	53.68	46.70	46.86	46.91	42.50
MLP-Mixer	17.32	82.99	78.80	47.49	39.88	44.54	49.15	38.88
CCT	21.98	88.94	82.91	51.60	43.57	43.94	44.99	37.78
Swin Transformer	27.52	84.62	72.80	48.10	42.25	38.25	34.46	32.41
PoolFormer	20.87	86.83	63.74	40.07	34.30	35.43	35.75	31.91
ViT	21.67	80.46	74.01	45.68	36.48	34.17	32.20	31.75
DenseNet	18.11	90.49	89.57	67.50	60.40	44.02	32.83	31.34
MobileViT	5.07	92.60	87.15	66.58	60.21	45.60	31.27	30.83
CPVT	20.32	82.06	76.85	47.91	41.41	35.57	31.93	30.04
LeViT	17.96	85.70	71.24	44.65	37.15	35.49	33.57	29.83
ResNeXt	23.00	84.96	80.17	57.74	51.10	35.96	21.80	21.53
PreActResNet	23.52	89.59	87.73	66.90	60.76	15.98	2.50	2.50
ResNet	23.08	91.25	81.77	60.68	54.29	6.68	0.21	0.21

Table 2. Adversarially robust generalization results on ImageNet using PGD-ℓ_∞ adversarial training for ViT and ResNet-101.

Architecture	**Clean Acc**	**PGD-ℓ_1**	**PGD-ℓ_2**	**PGD-ℓ_∞**	**AA-ℓ_∞**	**Worst-case Acc**
ViT	55.61	35.26	40.02	33.61	28.12	24.77
ResNet-101	61.16	33.81	37.61	31.09	25.24	20.08

Table 3. Adversarially robust generalization results on CIFAR-100 using PGD-ℓ_∞ adversarial training. Results in each category are ranked by the worst-case robust accuracy.

Architecture	**Clean Acc**	**PGD-ℓ_1**	**PGD-ℓ_2**	**PGD-ℓ_∞**	**AA-ℓ_∞**	**Worst-case Acc**
PVTv2	49.75	24.60	18.24	23.99	19.44	18.08
CoAtNet	52.06	22.92	17.38	24.74	20.38	17.09
ViT	51.67	20.69	15.51	22.80	18.84	14.92
ResNet	62.65	9.83	9.01	26.95	24.02	8.59
WideResNet	66.37	13.28	12.32	29.04	22.13	11.87
CvT	46.26	3.77	5.49	18.66	16.09	3.42

tion of architectures keeps consistent under different adversarial training techniques. Specifically, Transformers and hybrid models often achieve better adversarially robust generalization than CNNs, and PVTv2 and CoAtNet also exhibit relatively the best generalization ability. For CNNs, although TRADES mitigates the overfitting phenomenon on the trained ℓ_∞ noises, the robust generalization towards PGD-ℓ_1 and PGD-ℓ_2 is still worse than Transformers.

3.3. Evaluations on different architectures under optimal training settings

In our main paper, we report the results of all models with the aligned training settings following [1, 6]. Here, we further investigate adversarially robust generalization using several representative architectures under optimal settings on ImageNette (*e.g.*, more data augmentations techniques for Transformers and CNNs).

Since standard adversarial training on ImageNet with strong data augmentation (*e.g.*, AutoAugment, Mixup) would be hard to converge [1, 6], we only use the standard data augmentation techniques in our ImageNette adversarial training. To verify our conclusions under the relatively optimal settings on ImageNette, we choose 4 representative models (*i.e.*, PVTv2, CoAtNet, ResNet, and WideResNet) and conduct standard PGD-ℓ_∞ adversarial training (ϵ=8/255) with AutoAugment based on the standard setting. Specifically, we propose a new warm-up strategy in which we linearly increase the perturbation ϵ and PGD steps in the first 20 epochs. In contrast to the warm-up strategy proposed by [1] which gradually warmup the data augmentation strength, our warm-up strategy is easier to implement. The robust generalization results are shown in Table 5. We can see that the conclusions under the relatively optimal setting are much similar to those under the adversarial training on standard data augmentation.

Table 4. Adversarially robust generalization results on CIFAR-10 using TRADES adversarial training. Results in each category are ranked by the worst-case robust accuracy.

Architecture	Params (M)	Vanilla Acc	TRADES-ℓ_∞ Adversarial Training					
			Clean Acc	PGD-ℓ_∞	AA-ℓ_∞	PGD-ℓ_2	PGD-ℓ_1	Worst-case Acc
PVTv2	12.40	88.34	76.35	46.38	38.75	36.92	46.43	35.63
CPVT	9.49	90.34	78.30	46.55	38.86	34.63	43.64	33.53
ViT	9.78	86.73	77.92	48.13	41.07	34.43	41.79	33.50
CoAtNet	16.99	90.73	76.06	46.47	38.11	34.45	42.75	33.38
ViTAE	23.18	88.24	75.75	43.93	35.61	33.98	44.21	32.33
CCT	3.76	92.27	81.31	53.03	44.99	33.21	36.92	32.11
MobileViT	5.00	91.47	77.28	48.85	41.16	32.34	36.81	31.38
VGG	14.72	94.01	83.18	52.25	43.23	31.88	35.52	30.53
PoolFormer	11.39	89.26	73.95	45.34	37.44	30.30	36.02	29.67
Swin Transformer	27.42	91.58	79.67	51.34	44.18	30.54	34.02	29.24
WideResNet	55.85	96.47	84.67	59.89	52.72	29.36	32.60	28.12
MLP-Mixer	0.68	83.43	65.76	37.65	30.62	28.07	35.52	27.31
CeiT	5.56	85.24	72.39	37.11	28.97	26.89	34.38	26.10
BoTNet	18.82	94.16	79.50	52.06	43.80	28.50	28.54	26.01
DenseNet	1.12	94.42	80.50	52.86	44.77	27.28	26.57	24.48
PreActResNet	23.50	95.86	83.60	57.22	49.66	27.38	24.81	23.36
ResNeXt	9.12	95.64	80.98	55.43	47.77	26.51	23.09	22.06
ResNet	23.52	95.60	84.99	57.40	49.01	25.95	22.65	21.62
LeViT	6.67	89.01	75.85	46.35	38.70	21.71	19.48	18.85
CvT	19.54	87.81	74.16	42.43	35.52	16.23	12.43	12.23

Table 5. Evaluation results on ImageNette dataset using adversarial training under optimal training setting.

Architecture	Vanilla Acc	PGD-ℓ_∞ Adversarial Training					
		Clean Acc	PGD-ℓ_∞	AA-ℓ_∞	PGD-ℓ_2	PGD-ℓ_1	PGD-ℓ_∞ Worst-case Acc
PVTv2	92.04	90.62	67.21	61.95	60.55	60.92	56.50
CoAtNet	94.39	91.94	68.16	61.90	55.05	49.68	47.94
ResNet	93.60	84.83	62.71	55.00	18.45	4.60	4.58
WideResNet	93.62	88.54	67.56	59.63	21.27	5.71	5.68

3.4. Generalization on common corruptions

In our main paper, we discuss the generalization of adversarially-trained models on common corruptions using CIFAR-10-C. Here, we further report the results on ImageNette-C dataset.

As shown in Table 6, we can observe that as for the generalization of adversarial training towards corruptions on ImageNette, DenseNet behaves better than other Transformers. However, other adversarially-trained CNNs (such as ResNet) fail to show better generalization on corruption than Transformers (*e.g.*, CoAtNet, ViTAE). We will put them in future studies.

3.5. Improving the sparsity of Transformers

We choose two representative models PVTv2 and CoAtNet and conduct PGD-ℓ_∞ adversarial training (ϵ=8/255) on CIFAR-10 with (weight decay=0.05 in AdamW) and without regularization on them to get different weight sparsities.

Table 6. Robust generalization for different models towards common corruptions on ImageNette dataset. For natural noises we leverage mean Corruption Error (mCE) [2] and use 1-mCE as our ImageNette-C Acc (the higher the better).

	ImageNette Dataset			
	Vanilla Training		PGD-ℓ_∞ Training	
	Vanilla Acc	ImageNette-C Acc	Clean Acc	ImageNette-C Acc
DenseNet	90.49	67.29	89.57	66.49
BoTNet	91.36	62.35	86.41	65.56
CoAtNet	92.60	65.17	89.62	64.82
PVTv2	89.73	67.50	85.41	64.70
ViTAE	90.04	67.81	88.59	64.40
CeiT	89.88	66.14	85.99	63.67
MobileViT	92.60	63.69	87.15	62.34
PreActResNet	89.59	63.59	87.73	62.21
CVT	87.09	61.72	83.33	61.89
VGG	90.62	63.34	85.67	61.78
MLP-Mixer	82.99	64.22	78.80	61.26
WideResNet	88.83	60.81	76.11	61.22
CCT	88.94	60.21	82.91	60.69
CPVT	82.06	65.95	76.85	60.68
ResNeXt	84.96	58.67	80.17	59.43
ResNet	91.25	63.10	81.77	57.79
ViT	80.46	61.17	74.01	56.17
Swin Transformer	84.62	53.62	72.80	52.92
LeViT	85.70	53.19	71.24	49.07
PoolFormer	86.83	56.08	63.74	43.32

The robust generalization results are shown in Table 7. We can see that when decreasing the weight sparsity of PVTv2 and CoAtNet (PVTv2$_{nodecay}$ and CoAtNet$_{nodecay}$, namely without any weight decay regularization), their adversarially robust generalization also becomes worse, which is consistent with our conclusion that models exhibit better ad-

Table 7. Directly changing the weight sparsity of Transformers during training by weight decay.

Architecture	Clean Acc	PGD-ℓ_1	PGD-ℓ_2	PGD-ℓ_∞	AA-ℓ_∞	Worst-case Acc
PVTv2	75.99	46.14	35.77	46.48	38.18	33.54
PVTv2$_{nodecay}$	76.95	45.63	35.03	45.64	36.74	33.01
CoAtNet	77.73	42.30	33.80	48.27	39.85	32.17
CoAtNet$_{nodecay}$	75.58	39.80	31.63	46.16	37.80	30.18

versarially robust generalization tend to have higher weight sparsity. However, directly changing the weight sparsity of Transformers (*e.g.*, PVTv2) during training also decreases the clean accuracy. We will further study this in the future.

3.6. Training/testing accuracy

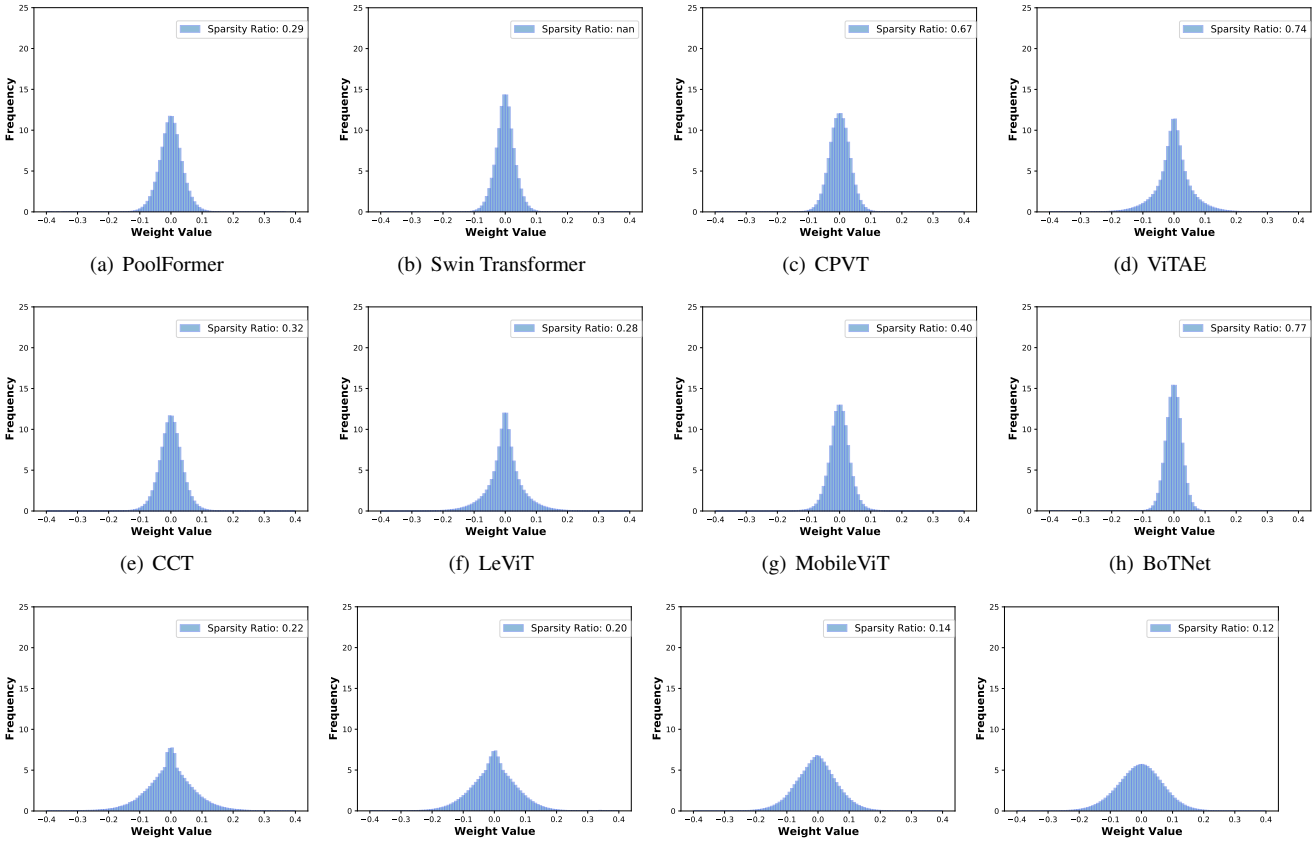
To demonstrate that our adversarial-trained models do not suffer from adversarial overfitting problems, we report the robust training/testing accuracy on PGD-ℓ_∞ attacks at every ten epochs during adversarial training on CIFAR-10. As shown in Figure 8, we can see that there seems no obvious adversarial overfitting during the whole adversarial training.

4. Weight Sparsity Understanding for More Models

Here, we report more visualization results for model weight sparsity as shown in Figure 1. From the results, we can find that the weight distributions of Transformers are often sparser than CNNs, which is consistent with our conclusions. In addition, for Transformers that exhibit bad robust generalization (*e.g.*, CeiT, CvT), their weight distributions are much denser than other Transformers; while for architectures that have relatively good robust generalization (*e.g.*, ViT, CPVT), their weight distributions also exhibit good sparsity.

Table 8. Robust training/testing accuracy during PGD-ℓ_∞ adversarial training on CIFAR-10 at every 10 epoch. Results are shown in "training/testing" accuracy.

Architecture/ Epoch	10	20	30	40	50	60	70	80	90	100
WideResNet	16.54 / 19.23	21.79 / 23.68	24.21 / 28.44	30.39 / 34.21	35.34 / 38.45	44.69 / 49.32	50.23 / 52.17	54.27 / 54.21	56.31 / 54.35	57.95 / 55.19
BotNet	9.56 / 9.47	12.94 / 17.00	23.23 / 29.71	18.04 / 22.81	30.84 / 37.37	35.65 / 40.56	39.41 / 44.79	40.09 / 45.57	42.38 / 46.72	43.79 / 47.64



(a) PoolFormer (b) Swin Transformer (c) CPVT (d) ViTAE

(e) CCT (f) LeViT (g) MobileViT (h) BoTNet

(i) MLP-Mixer (j) CeiT (k) PreActResNet (l) ResNeXt

Figure 1. Weight distribution visualization for more adversarially-trained models on CIFAR-10.

References

[1] Y. Bai, J. Mei, A. L. Yuille, and C. Xie. Are transformers more robust than cnns? *Advances in Neural Information Processing Systems*, 34, 2021. 2, 3

[2] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018. 4

[3] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020. 1

[4] A. Raghunathan, J. Steinhardt, and P. Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018. 1

[5] Z. Shen, M. Zhang, H. Zhao, S. Yi, and H. Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3531–3539, 2021. 1

[6] S. Tang, R. Gong, Y. Wang, A. Liu, J. Wang, X. Chen, F. Yu, X. Liu, D. Song, A. Yuille, et al. Robustart: Benchmarking robustness on architecture design and training techniques. *arXiv preprint arXiv:2109.05211*, 2021. 2, 3

[7] Y.-H. H. Tsai, S. Bai, M. Yamada, L.-P. Morency, and R. Salakhutdinov. Transformer dissection: A unified understanding of transformer's attention via the lens of kernel. *arXiv preprint arXiv:1908.11775*, 2019. 1

[8] D. Yin, R. Kannan, and P. Bartlett. Rademacher complexity for adversarially robust generalization. In *International conference on machine learning*, pages 7085–7094. PMLR, 2019. 1