MixACM: Mixup-Based Robustness Transfer via Distillation of Activated Channel Maps

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep neural networks (DNNs) are susceptible to adversarial examples – small and imperceptible changes in the natural inputs yet incorrectly classified by models. The most effective defense mechanism against these examples is adversarial training which constructs adversarial examples during training by iterative maximization of loss. The model is then trained to minimize the loss on these constructed examples. This min-max optimization requires more data, larger capacity models, more compute power, and degrades the standard generalization performance of a model. Can we have the robustness more efficiently? In this work, we explore this question from the perspective of knowledge transfer. We first theoretically show the robustness transfer from a robust teacher model to a student model with the help of mixup augmentation. We then propose a novel robustness transfer method called Mixup-Based Activated Channel Maps (MixACM) Transfer. MixACM transfer robustness from a robust teacher to a student by matching activated channel maps generated without expensive adversarial perturbations. Extensive experiments on multiple datasets and different settings show our method can transfer robustness while also improving generalization on natural images.

1 Introduction

Deep learning models have achieved impressive performance on a wide variety of challenging tasks such as image recognition, natural language generation, game playing, etc. However, these models are also susceptible to even small changes in the input. It is possible to craft tiny changes in the input such that a model classifies unaltered examples correctly but classifies the same examples incorrectly after small and visually imperceptible perturbations[68, 24]. These altered examples are called adversarial examples, and this vulnerability of the state-of-the-art models has raised serious security questions [38, 55, 7, 44].

Many defense mechanisms have been proposed to train DNNs to be robust against such adversarial perturbations. These techniques include defensive distillation [54], gradient regularization [26, 55, 59], model compression [18, 42], activation pruning [20, 56], adversarial training [46], etc. Among them, Adversarial Training (AT) is one general strategy that is the most effective [4]. In general, adversarial training is a kind of data augmentation technique that trains a model on examples augmented with adversarial changes [46]. The adversarial training consists of an inner, iterative maximization to augment natural examples with adversarial perturbations, and an outer minimization loop similar to normal training. Many different methods have been introduced to improve robustness [73, 12, 49, 88, 76, 63, 91, 75, 5, 78, 36, 41, 53], but all of them are fundamentally based on the principle of training on adversarially augmented examples.

Adversarial training is more challenging compare to normal training. Better robust generalization requires larger capacity models [46, 50, 77] and even over-parameterized models which can easily

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

fit data for natural training [89] may have insufficient capacity to fit adversarially augmented data [94]. The sample complexity of adversarial training can be significantly higher than normal training, and it requires more labeled [61] or unlabeled data [73, 12, 49, 88]. The inner maximization for the generation of adversarial perturbations for adversarial training is significantly more expensive computationally as it requires iterative gradient steps with respect to the input (e.g. adversarial training takes $\sim$7x more time compare to normal training). Adversarial training also degrades the performance of a model on natural examples significantly [50, 91].

Can we train robust models more efficiently e.g., with less data, small capacity models, without extra back-propagation steps, and no significant sacrifice of clean accuracy? For normal training, knowledge transfer is one possible paradigm for the efficient training of deep neural networks. In normal knowledge transfer, a pre-trained teacher model f_T is leveraged to efficiently train a student model f_S on the same or similar dataset [67]. However, knowledge transfer methods are designed to transfer features related to normal generalization which may be at odds with robustness [71, 35]. A recent line of work has explored transferring or distilling robustness from pre-trained models [29, 64, 22, 13]. Although these techniques are effective, they also require extra gradient computation and work in specific cases. In this work, we argue that a better approach for robustness transfer is to distill intermediate features of a robust teacher, generated on mixup examples. Our proposed approach does not require any extra gradient computation and works well with smaller models and fewer data samples. It can also effectively transfer robustness across models and datasets.

We begin with the theoretical analysis and showed that adversarial loss can be bounded with two terms: natural loss and distance between the student f_S and teacher model f_T on mixup examples. To minimize the distance between robust teacher and student, we proposed a new distillation method. Our proposed distillation method is based on channel-wise activation analysis of robustness by Bai et al. [5] and studies showing that adversarially trained models learn fundamentally different features [35, 71, 96]. Convolution channels of a convolutional neural network learn different patterns which, when combined, describe a specific object. Samples of different classes activate different channels for an intermediate layer. However, adversarial examples make these channels activate more uniformly thereby destroying class-related information. Adversarial training solves this issue by forcing a similar channel-activation pattern for normal and adversarial examples [5]. Our proposed distillation method generates activated channel maps from a robust teacher which has already learned robust features. The student, then, is forced to match these activated channel maps.

We conduct extensive experiments to show the effectiveness of our method in many different settings. We start by exhibiting the ability of our method to transfer robustness without using adversarial examples. We then show that our method can distill robustness from large pre-trained models to smaller models and it can transfer robustness across datasets. We also show that our method is capable to robustify a model even with a small number of examples. Our contributions are as follows:

1. We show that adversarial loss of a student model can be bounded with the distance between robust teacher and student model on mixup examples.
2. We propose a new method to transfer robustness from the intermediate features of robust pre-trained models.
3. We analyzed our method with experiments on CIFAR10, CIFAR100, and ImageNet datasets, and showed that our method can achieve comparable adversarial robustness to SOTA methods without generating adversarial examples. Our method improves clean accuracy significantly.

2 Related Work

Adversarial Training and Robust Features. Adversarial attacks are considered to be a security risk [6, 68, 24, 11], and many methods have been proposed to defend against them [26, 24, 54, 27, 55, 70, 46, 9, 40, 59, 18, 42, 20, 56]. Adversarial training [46], among them, is the most effective defense [4]. The fundamental principle of adversarial training is to generate adversarial perturbation during training by doing iterative back-propagation w.r.t. input. The model is then trained on these perturbed examples. Building upon this principle, different aspects of adversarial training have extensively been studied e.g. effect of large unlabeled data [73, 12, 88], theoretical aspects to improve the trade-off between accuracy and robustness [91], improving parts of adversarial training for better

robustness [75, 78, 53, 5, 25], making adversarial training fast [49, 63, 76, 1], pruning with adversarial training [84, 62], adversarial training with unlabeled data [36], etc. Our work is different from these methods as our main concern is robustness transfer without generating adversarial perturbations. Another line of work studied the effects of adversarial training and showed that adversarial training learns 'fundamentally different' [50] features compare to normal training [35, 83]. Our method is motivated by these studies and is designed to leverage these robust features.

Knowledge and Robustness Transfer. Knowledge transfer is training a student model f_S more efficiently on a dataset with the help of a teacher model f_T trained on a similar or related dataset [67]. Many different settings have been explored to achieve this objective by using soft labels produced by teacher [8, 32], knowledge learned by intermediate layers of teacher also called feature distillation [58, 86, 34, 85, 72, 37, 65, 31, 30, 69], or matching input gradients [86, 67]. However, the main objective of these knowledge transfer methods is about improving the generalization of natural images. Our work is fundamentally different from them as we want to transfer robustness.

Our work is motivated by [29, 64, 22, 13] showing transferability of robustness. Hendrycks et al. [29] showed that robust features learned on large datasets like ImageNet can improve adversarial training, Shafahi et al. [63] showed that robust pre-trained models can act as feature extractors in transfer learning. These two methods are limited in application as pre-trained models are used as backbone models. In [22], the authors showed that the robustness can be distilled more efficiently from a large pre-trained model to a smaller model by minimizing the softened class scores along with adversarial training. Compared to [22], our method does not require any adversarial training. In [13], the distillation is performed by matching the gradient of the teacher and student, requiring fine-tuning on target tasks, back-propagation to get gradients of the teacher w.r.t. input and training of a discriminator. Compared to [13], our proposed method does not require any extra back-propagation steps.

Activation and Robustness. Many works have explored the role of activations in the robustness from different angles e.g. modifying output of intermediate layers to improve adversarial training [79, 20, 45, 48, 81], using different activation functions [56, 74, 82, 25, 52, 5] or interpretation of what is learned by adversarial training with activations [83, 5]. However, we leveraged activations to transfer robustness. Our distillation method is inspired by the channel-wise analysis of robustness proposed by Bai et al. [5].

Mixup Augmentation and Robustness. Mixup augmentation was introduced to improve the generalization, however, it also improved performance against one-step adversarial attacks [92]. This motivated many mixup-based adversarial defense methods e.g. Pang et al. [51], Lee et al. [39], Archambault et al. [3], Bunk et al. [10], Chen et al. [14]. These defense methods primarily improve adversarial training by mixup. Our work, however, is different as we explored the role of the mixup in robustness transfer from both theoretical and algorithmic perspectives. Our theoretical work is based on [95] who showed a connection between mixup and robustness theoretically. Goldblum et al. [22] also briefly empirically explored the effect of some augmentation methods on robustness transfer.

3 Setup

We consider task of mapping input $x \in \mathcal{X} \subseteq \mathbb{R}^d$ to label $y \in \mathcal{Y} = \{1, 2, \ldots, K\}$. Given the training data $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$, the goal is to learn a classifier $f : \mathcal{X} \to \mathbb{R}^k$ from a hypothesis space $\mathcal{F}$. Let $L\big(f(x), y\big)$ be the loss function that measures that how poorly the classifier $f(x)$ predicts the label y. Suppose f can be decomposed into $g \circ h$, where $h \in \mathcal{H}$ is a feature extractor and $g \in \mathcal{G}$ stands for the top model.

Adversarial Augmentation: Adversarial training augments natural examples with adversarial perturbation which maximizes the loss, i.e. $\tilde{x} = x + \delta$ where $\delta = \arg\max_{\|\delta'\|_p \leq \epsilon} L\big(f(x + \delta'), y\big)$. Here $\|\cdot\|_p$ stands for L_p norm. Throughout this paper, we take $p = \infty$. We denote the adversarial loss [46] as:

$$\tilde{\mathcal{L}}(f, \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \max_{\|\delta\|_\infty \leq \epsilon} L(f(x_i + \delta), y_i),$$

where ϵ is the perturbation budget that governs the adversarial robustness of the model.

Mixup Augmentation: Mixup constructs virtual examples by linearly combining them [51]: $\tilde{x}_{ij}(\lambda) = \lambda x_i + (1-\lambda)x_j$, and $\tilde{y}_{ij}(\lambda) = \lambda y_i + (1-\lambda)y_j$, where $\lambda \in [0,1]$ follows the distribution P_λ. The mixup loss is

$$\mathcal{L}_{mix}(f,\mathcal{D}) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{E}_{\lambda\sim P_\lambda}[L(f(\tilde{x}_{ij}(\lambda)),\tilde{y}_{ij}(\lambda))].$$

4 Analysis: How to Transfer Robustness?

In this section, we provide an analysis on how to transfer robustness with knowledge distillation and mixup augmentation. We first explored this problem from a theoretical perspective and showed that adversarial loss can be decomposed into the sum of an adversarial loss and a distillation loss. Furthermore, these two loss functions can be approximated by mixup-based augmentation. We then discussed how we can design a distillation loss that distills robustness from the intermediate features of a robust model.

4.1 Theoretical Analysis

Robustness Transfer We first give a generalization bound for the robust test error via the distillation. Different to the classical generalization bound, we compress the hypothetical space $\mathcal{F}$ into a smaller space $\{g \circ h^T, g \in \mathcal{G}\}$ derived by the teacher feature extractor. Hsu et al. [33] points out that the distillation helps to derive a fine-grained analysis for the prediction risk and avoid getting a vacuous generalization bound. Therefore, we take the following theorem as a starting point to derive robustness transfer. We consider the loss function $L(f(x),y) = 1 - \phi_\gamma\big(f(x),y\big)$, where ϕ_γ is the softmax function with temperature $\gamma > 0$:

$$\phi_\gamma\big(f(x),y\big) = \frac{\exp(f(x)_y/\gamma)}{\sum_{k=1}^{K}\exp(f(x)_k/\gamma)}.$$

Theorem 4.1. *Let the temperature $\gamma > 0$ be given. Then, with probability $1-\delta$, for any $f \in \mathcal{F}$,*

$$\mathbb{E}\big[\operatorname*{arg\,max}_{\|\delta\|_\infty\le\epsilon}\mathbb{I}\{\hat{y}(x+\delta)\neq y\}\big] \le 2\tilde{\mathcal{L}}(f,\mathcal{D}) + \frac{8}{\gamma}\tilde{\mathcal{L}}_{dis}(f,\mathcal{D}) + \frac{4}{\gamma}\mathfrak{R}_S(\mathcal{F}^T) + \frac{16}{n} + 6\sqrt{\frac{2/\delta}{2n}}, \quad (1)$$

where $\mathfrak{R}_S$ is the Rademacher complexity, $\mathcal{F}^T = \{\max_{\|\delta\|_\infty<\epsilon} g\circ h^T(x+\delta),\ g\in\mathcal{G}\}$ and

$$\tilde{\mathcal{L}}_{dis}(f,\mathcal{D}) = \frac{1}{n}\inf_{g\in\mathcal{G}}\sum_{i=1}^{n}\max_{\|\delta\|_\infty<\epsilon}\big|f(x_i+\delta) - g\circ h^T(x_i+\delta)\big|.$$

On the right hand side of (1), two terms depend on the classifier f, i.e., the adversarial loss $\tilde{\mathcal{L}}(f,\mathcal{D})$ and the distillation loss $\tilde{\mathcal{L}}_{dis}(f,\mathcal{D})$. Thus, we can minimize $2\tilde{\mathcal{L}}(f,\mathcal{D}) + 8\tilde{\mathcal{L}}_{dis}(f,\mathcal{D})/\gamma$ to find a classifier with small robust test error. Note that the loss function L is bound above by the cross-entropy loss. So $\tilde{\mathcal{L}}(f,\mathcal{D})$ can be replaced with a common-used adversarial loss based on cross-entropy.

Adversarial Robustness, Transfer and Mixup However, solving the maximization problem for δ is time-consuming, which is a severe problem in adversarial training especially for large models. For binary classification task with logistic loss function, [95] proves that $\tilde{\mathcal{L}}(f,\mathcal{D})$ can be bounded above by $\mathcal{L}_{mix}(f,\mathcal{D})$. In addition, the distillation loss that measures the distance between the student model and the teacher model also considers the worst-case δ. In our algorithm, we use the KL divergence or Euclidean distance and mixup sample to design the distillation loss. Inspired by the Theorem 3.3 in [95], we present the following example to illustrate the rationality behind our design. Please see Appendix for more details and complete proof.

Theorem 4.2. *Suppose that $\mathcal{Y} = \{0,1\}$ and L is the logistic loss. Let f be a fully connected neural network with ReLU activation function or max-pooling. Suppose a learnt teacher model f^T with the same network structure is given. Denote $\mathcal{D}_{dis} = \{\big(x_i, f^T(x_i)\big), i = 1,\ldots,n\}$. Under certain assumptions, we have for any positive scalar α,*

$$\tilde{\mathcal{L}}(f,\mathcal{D}) + \alpha\tilde{\mathcal{L}}(f,\mathcal{D}_{dis}) \le \mathcal{L}_{mix}(f,\mathcal{D}) + \alpha\mathcal{L}_{mix}(f,\mathcal{D}_{dis}).$$

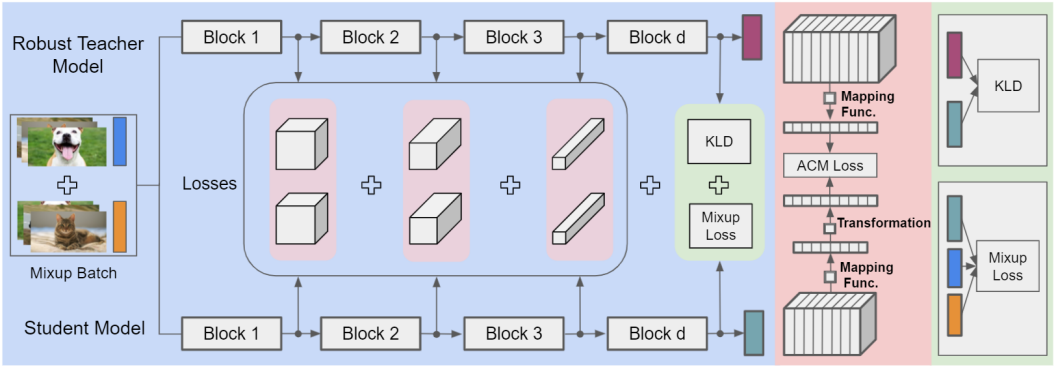

Figure 1: **Overview of our approach**. The mixup augmented examples are passed through a robust teacher and a student to get intermediate features. These features are then passed through a mapping function to get activated channel maps. The student mimics the activated channel maps of robust teacher by minimizing ACM loss. The student model also minimizes the standard KLD and Mixup loss.

4.2 Considerations for Algorithmic Design

Our theoretical analysis suggests minimizing the distance between teacher and student on mixup examples. However, as observed by [22] and our experiments in Section 6, minimization of KL-Divergence between the outputs of teacher and student models is not enough. How can we reduce the gap between the teacher and student more effectively? Based on previous work [21, 83, 5], we hypothesize that the gap between latent space of robust teacher and student is large and it requires distillation from intermediate features.

To reduce the distance between the latent spaces of robust teachers and student, we get inspiration from recent channel-based analysis of robustness [5] and works on the interpretation of what is learned by robust models [50, 35, 83]. Specifically, we consider convolution channels of CNNs to be fundamental learning blocks: each channel learns class-specific patterns and the output of the model is a combination of these basic patterns. As noted by [35], these basic blocks learn both robust and non-robust features with natural training. Adversarial examples amplify non-robust features while suppressing the robust features e.g. adversarial examples frequently activate channels that are rarely activated by natural examples [5]. Adversarial training works by suppressing non-robust channels and making the activation frequency of natural and adversarial examples similar.

Motivated by this analysis, we propose Activated Channel Map (ACM) transfer loss. Our method extracts the activated channel maps of the robust model's intermediate features on mixup examples. The student is then forced to mimic these maps. Specifically, we consider the maximum value of each channel as a proxy for the extent of how activated a channel is: a channel is less activated if it has a smaller maximum value and more activated if it has a high maximum value. We then make the student match the normalized activation map of the teacher. Further analysis of our method is presented in the supplementary material. We present our method in the next section.

5 Methodology

In this section, we introduce our Mixup-Based Robustness Transfer method. As discussed above, the method aims to transfer robustness from a robust teacher model f_T to a student model f_S via matching activated channel maps of a robust teacher on mixup augmented examples. An overview of our method is given in Figure 1.

5.1 Activated Channels Maps (ACM) Transfer Loss

The purpose of robust teacher is to supervise the student's focus on robust features. This is achieved by matching activated channel maps of teacher and student. The activated channel maps of a model is extracted as follows: given i-th block activations $\mathcal{A}_i$ (output of i-th layer), we apply a function

$\phi_c : \mathbb{R}^{B\times C\times H\times W} \rightarrow \mathbb{R}^{B\times C\times 1\times 1}$ to get the activated channel map a_i e.g. $a_i = \phi_c(\mathcal{A}_i)$ as illustrated in Figure 2.

The activated channel map of student and teacher may differ in scale. The contribution of each channel in the final output, on the other hand, depends only on the relative value compared to other channels. Therefore, we first normalize this map. To force the student to mimic the teacher's activated channel maps, we use ℓ_2 distance between the normalized activated channel maps of teacher and student. For i-th layer, we can define ACM loss as:

$$\mathcal{L}_{acm_i}(\mathcal{A}_i^T, \mathcal{A}_i^S) = \| \frac{a_i^T}{\|a_i^T\|_2} - \frac{a_i^S}{\|a_i^S\|_2} \|_2^2.$$

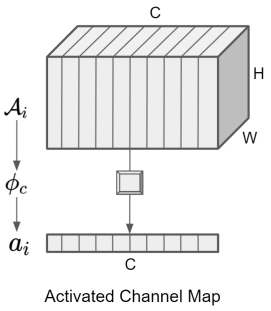

Figure 2: Mapping function ϕ_c is applied channel-wise to get the Activated Channel Maps (ACM).

We minimize this distance at the end of each block of the models. A block is a set of convolution layers grouped like residual blocks in ResNets [28]. For a student and teacher model with d such blocks, the total ACM loss becomes as follows: $\mathcal{L}_{acm}(x; f^T, f^S) = \sum_{i=0}^{d} \mathcal{L}_{acm_i}(f^T(x)_i, f^S(x)_i)$, where $f(x)_i$ represents output of the model at i-th block.

Mapping Function (ϕ_c): The aim of activated channel map function (ϕ_c) is to extract a map of least to most activated channels. For this purpose, we used maximum value of each channel: $\phi_c(x) = \max_i x_i \forall i \in \{0, ..., c_l\}$, where c_l is number of channels in l-th layer.

5.2 Transferring Robustness with Different Feature Sizes

Our method extracts a scalar value corresponding to each channel in a layer. Until now, we assume that the teacher and student have the same number of channels. For this case, we only need the ϕ_c mapping function. To accomplish knowledge transfer across models with a different number of channels, we need to apply transformation either on a teacher or student's activated channel maps. We experimented with two transformations: adaptive pooling and affine transformation.

Adaptive Pooling: Pooling is used in most CNN architectures to reduce the size of feature vectors and gain invariance to small transformations of the input. The pooling layer works by dividing the input into patches and taking an average or maximum value over these patches. By changing the size of these patches, pooling can also be used for up-sampling. Pooling operation can be described as follows: $y[p] = \max_{q\in\Omega(p)} x[q]$, where Ω represents a patch or neighbourhood of $x[p]$. We use PyTorch implementation of adaptive maximum pooling 1d to either down or up-sample teacher or student output.

Affine Transformation: We can also use the learning function to convert the input size to output size. For this, we use a fully connected layer. Based on our ablation studies described in supplementary material, we use adaptive pooling as a default in our method.

5.3 KL Divergence Loss with Soft Labels

Soft labels have shown to be useful for adversarial robustness by Pang et al. [52]. Therefore, for distillation on the same dataset (e.g. where teacher and student have the same classification layer), we minimize the KL divergence between the logits provided by teacher and student following [32, 22]. The KLD loss becomes as follows: $\mathcal{L}_{kld}(x; f^T, f^S) = \gamma^2 KL(y^T(\gamma), y^S(\gamma))$, where γ is the temperature parameter, and $y(\gamma)$ is output logits of a model normalized by temperature parameter as used by Hinton et al. [32]. We omit this loss term when transferring from one dataset to another.

5.4 Mixup Fueled Distillation

Our theoretical analysis suggest minimizing distance between teacher and student with mixup examples. To this end, we use mixup to linearly combine inputs and outputs and minimize loss on these inputs. The final training objective for the student is as follows:

$$\min_{f^S} \mathbb{E}_{(x,y)\sim\mathcal{D}} \mathbb{E}_{\lambda\sim P_\lambda} \left[(1-\alpha_{kld})\mathcal{L}_{mix} + \alpha_{kld}\mathcal{L}_{kld} + \alpha_{acm}\mathcal{L}_{acm} \right].$$

Table 1: **Detailed Comparison**. CIFAR-10 test accuracy and robustness comparison under two different ℓ_∞ attacks of magnitude $\epsilon = 8/255$. Our method outperforms robustness distillation methods in both accuracy and robustness. It outperforms existing state-of-the-art robust models in clean accuracy while retaining comparable robustness. All reported values for our method are mean $\pm$ std of five repetitions. B.P. in the table stands for Back Propagation.

Method	Model	Extra B.P. Steps	Clean Acc.	PGD 100	Auto Attack
AT (Madry et al. [46])	WRN-34-10	✓	87.14	47.04	44.04
LAT (Singh et al. [66])	WRN-34-10	✓	87.80	53.04	49.12
TLA (Mao et al. [47])	WRN-34-10	✓	86.21	50.03	47.41
YOPO (Zhang et al. [90])	WRN-34-10	✓	87.20	47.98	44.83
Free AT (Shafahi et al. [63])	WRN-34-10	✓	86.11	46.19	41.47
TRADES (Zhang et al. [91])	WRN-34-10	✓	84.92	56.43	53.08
MART (Wang et al. [75])	WRN-34-10	✓	83.07	55.57	-
AWP (Wu et al. [78])	WRN-34-10	✓	85.36	59.12	56.17
FAT (Zhang et al. [93])	WRN-34-10	✓	84.52	54.36	53.51
Overfitting (Rice et al. [57])	WRN-34-20	✓	85.34	58.00	53.42
BoT TRADES (Pang et al. [52])	WRN-34-20	✓	86.43	54.39	54.39
Adv. Pretraining (Hendrycks et al. [29])	WRN-28-10	✓	87.10	57.40	54.92
IGAM (Chan et al. [13])	WRN-34-10	✓	88.70	43.00	-
RKD (Goldblum et al. [23])	WRN-34-10	×	89.38	0.21	0
Ours (w/o Mixup)	WRN-34-10	×	91.17±0.06	49.53±1.44	47.38±1.51
Ours (w/ Mixup)	WRN-34-10	×	90.76±0.05	56.65±0.93	53.93±0.77

6 Experiments

In this section, we empirically evaluate our method for robustness transfer. We primarily show the effectiveness of our method under three settings: 1) Robustness Transfer in Section 6.1 to demonstrate that our method can transfer robustness from a pre-trained teacher to student without generating adversarial examples; 2) Robustness transfer from larger to smaller models in Section 6.2 to show its effectiveness under distillation settings; and 3) robustness transfer under transfer learning settings e.g., from one dataset to another dataset and robustness with smaller dataset sizes in Section 6.3.

Evaluation of Robustness: We evaluate MixACM and other methods against standard FGSM and PGD-k as well as the strongest auto-attack by Croce and Hein [16]: a parameter-free adversarial attack designed to give a reliable evaluation of robustness. It is an ensemble of four white and black-box attacks. Apart from these, we also report robustness of our robust models on auto pgd-ce [16], auto pgd-dlr [16], FAB [15] and query-based black-box attack called Square attack [2]. For PGD-k, we use the standard value of step size i.e. $\alpha = 2/255$, and evaluated with an increasing number of iterations (k) including an extreme version, PGD-500.

Pre-Trained Robust Models: For CIFAR experiments, we use WideResNet-28-10 and WideResNet-34-20 [87] provided by Gowal et al. [25]. For ImageNet experiments, we use adversarially trained ResNet-50 provided by Salman et al. [60].

Training Details: For CIFAR experiments, we use WideResNet-34-10 [87] as student model following [13]. All student models for our proposed method are trained without adversarial perturbations. We use 200 epochs, batch size of 128, a learning rate of 0.1, cosine learning rate scheduler [43], momentum optimizer with weight decay of 0.0005. For our loss, we use $\alpha_{acm} = 5000$. For KD loss, we use temperature value of $\gamma = 10$ and $\alpha_{kld} = 0.95$ and the value for mixup coefficient is $\lambda = 1$. ImageNet models are trained for 120 epochs. Further training details are in the supplementary material.

6.1 Robustness Transfer

CIFAR10: To show the effectiveness of our method for robustness transfer and compare it with state-of-the-art, we follow settings of [13]: WideResNet-34-10 as a student on CIFAR10. We compared our method with two kinds of defense methods: general variants of adversarial training and three robustness transfer methods. For average $\pm$ variance of five runs for our method are shown in Table 1.

Table 2: **Comparison on ImageNet.** Results of robustness transfer for large-scale ImageNet dataset with ResNet-50 with two different training settings. Our method has significantly higher clean accuracy and robustness compare to two other methods although our method does not require any back-propagation steps.

Method	Train ϵ	Acc.	PGD-10	PGD-50	PGD-100	Train ϵ	Acc.	PGD-10	PGD-50	PGD-100
	Test $\epsilon = 2/255$					Test $\epsilon = 4/255$				
Free-AT [63]	2	**64.45**	43.52	43.39	43.40	4	60.21	32.77	31.88	31.82
Fast-AT [76]	2	60.90	-	43.46	-	4	55.45	-	30.28	-
Ours	0	59.64	46.03	45.96	45.92	0	59.64	32.70	31.51	31.39
Ours+RA	0	62.05	**48.54**	**48.45**	**48.44**	0	**62.05**	**34.93**	**33.71**	**33.63**

Table 3: **Distillation Results.** Performance for robustness distillation on CIFAR10 and CIFAR-100 datasets. The teacher model is WideResNet-28-10 and student model is WideResNet-16-10. Robustness is reported for diverse set of attacks of smagnitude $\epsilon = 8/255$.

Dataset	Method	Acc.	FGSM	PGD 20	PGD 500	APGD CE	APGD DLR	FAB	Square	Auto Attack
CIFAR-10	Natural	93.79	5.34	0	0	0	0	0	0	0
	PGD7-AT	83.90	53.07	47.55	47.36	47.28	44.52	44.89	53.26	44.51
	RKD	88.60	27.37	0.12	0	0	0	0	7.31	0
	Ours w/o mixup	90.47	56.98	40.69	39.69	39.24	36.46	37.21	56.30	36.42
	Ours	89.98	59.48	48.09	47.51	47.31	43.73	44.46	60.20	43.64
CIFAR-100	Natural	76.91	3.17	0	0	0	0	0	0	0
	PGD7-AT	60.29	28.37	24.94	24.75	24.71	21.94	22.16	27.97	21.94
	RKD	67.19	10.27	0.23	0.16	0.04	0	0	3.73	0
	Ours w/o mixup	60.81	30.13	25.38	25.12	24.93	19.51	19.83	29.74	19.50
	Ours	59.35	31.89	28.24	28.20	27.96	22.02	22.26	31.60	22.02

The robustness of our method is comparable to most state-of-the-arts methods while significantly higher clean accuracy.

ImageNet: We also consider high-resolution, large-scale and challenging ImageNet dataset [19]. It is also significantly more challenging for robustness [81, 80]. The results of our method for ResNet=50 student are shown in Table 2. For our method, we show two results: one with only mixup augmentation and one with mixup and random augmentation [17]. We observed that addition of random augmentation further improves the results of our method. Our method outperforms both Free [63] and Fast [76] AT in accuracy and robustness, significantly.

6.2 Distillation

A benefit of our method is its ability to distill robustness across models. To show this, we conduct two sets of experiments. First, we show robustness distillation from WideResNet-28-10 to WideResNet-16-10 for CIFAR10 and CIFAR100 datasets. The results are shown in Table 3. To see the effect of robustness transfer across different models, we also conduct an experiment where we changed the size of the model in terms of depth, the number of channels, or both e.g. WideResNet-28-10 to WideResNet-16-5. The results are shown in Table 4. Our method works well for both larger-to-smaller as well as smaller-to-larger models. However, our method does not perform well when both depth and number of channels are decreased dramatically, e.g., from WideResNet-34-20 to WideResNet-10-1 (student's size is 0.04% of teacher's size). To elaborate further, we discuss this in the supplementary material.

6.3 Transfer Learning

To further show the effectiveness of our method, we conduct two sets of experiments under transfer learning settings: transfer learning from one dataset to another dataset and transfer learning with fewer data samples. For first experiment, we follow Hendrycks et al. [29]'s setting: our teacher model is a WideResNet-28-10 trained on ImageNet of size 32×32 and student is a WideResNet-34-10. The results for the CIFAR100 dataset under these settings are shown in Table 5.

Table 4: **Model Compression**. Comparison of our method with PGD-7 Adversarial Training for accuracy and robustness on various sizes of models. The robustness is reported for ℓ_∞ PGD-20 attack of magnitude $\epsilon = 8/255$. Our method works well for robustness transfer from larger to smaller as well as, for smaller to larger models.

Distill Type	Teacher	Student	Size Ratio	PGD7-AT		Ours	
				Acc.	Rob.	Acc.	Rob.
Depth	WRN-28-10	WRN-16-10	46.92%	83.90	47.55	89.93	48.09
Channel	WRN-28-10	WRN-28-5	25.04%	83.90	47.74	90.26	48.75
Channel	WRN-34-20	WRN-34-10	25.01%	84.30	49.72	87.26	52.83
Depth & Channel	WRN-28-10	WRN-16-5	11.76%	83.51	48.11	88.96	36.95
Depth & Channel	WRN-34-20	WRN-16-10	9.227%	83.90	47.55	86.31	47.18
Tiny network	WRN-28-10	WRN-10-1	0.21%	67.11	38.85	66.65	1.15
Tiny network	WRN-34-20	WRN-10-1	0.04%	67.11	38.85	63.19	3.75
Same network	WRN-28-10	WRN-28-10	100%	84.17	49.23	90.48	54.05
Same network	WRN-34-20	WRN-34-20	100%	84.20	49.11	87.17	53.60
Up Transfer	WRN-28-10	WRN-34-20	505.85%	84.20	49.11	90.67	58.36

Figure 3: Robustness Transfer with Smaller Data. Table shows robustness and accuracy (%) for the student model trained with different fractions of CIFAR10 dataset. Our method achieves significantly better clean accuracy and robustness compare to adversarial training. Dotted blue line shows the performance of adversarial training with full data.

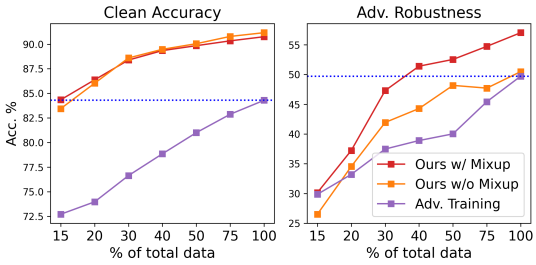

Table 5: Robustness Transfer from ImageNet to CIFAR100. Teacher model is adversarially trained on ImageNet and student is trained on CIFAR100. Our method achieves significantly better clean accuracy and better robustness compared to both adversarial training and IGAM [13]. The robustness is reported according to [13] for a fair comparison. Reported results for our method is mean of five repetitions.

Defense	Acc.	FGSM	PGD-k		
			5	10	20
Natural [13]	78.70	7.95	0.13	0.03	0
PGD7-AT [13]	60.40	29.10	29.30	24.30	23.50
IGAM [13]	62.39	34.31	29.59	24.05	21.74
Ours w/o Mixup	**68.91**	29.09	28.65	21.70	20.27
Ours	65.69	32.09	**32.04**	**25.58**	**24.14**

For the second part, we show the effectiveness of our method with small data. To this end, we randomly choose a smaller portion of CIFAR10 and conducted experiments on this smaller portion of data with our method as well with PGD-7 adversarial training. The results are shown in Table 3. Our method outperforms adversarial training significantly for both robustness and clean accuracy. Our method with just 30% of data can match the robustness and clean accuracy of a robust model trained with full data.

7 Conclusion

In this paper, we investigated robustness transfer from a theoretical and algorithmic perspective. The general principle behind most successful adversarial defense methods is the generation of adversarial examples during training. However, adversarial training is complicated as it requires more data, larger models, higher compute resources and degrades clean accuracy. We investigated whether we can have robustness more efficiently through robustness transfer? To this end, we first presented a theoretical result bounding adversarial loss of student model with the distance between robust teacher and student on mixup examples. We, then, proposed a novel robustness transfer method that is based on channel-wise analysis of robustness. Our method can achieve significant robustness against challenging and reliable auto-attack as well as other standard attacks while outperforming in terms of clean accuracy. The proposed method also works well for the distillation of robustness to the smaller model, and transfer learning with fewer data points.

References

[1] Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems*, 33, 2020.

[2] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer, 2020.

[3] Guillaume P Archambault, Yongyi Mao, Hongyu Guo, and Richong Zhang. Mixup as directional adversarial training. *arXiv preprint arXiv:1906.06875*, 2019.

[4] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, pages 274–283. PMLR, 2018.

[5] Yang Bai, Yuyuan Zeng, Yong Jiang, Shu-Tao Xia, Xingjun Ma, and Yisen Wang. Improving adversarial robustness via channel-wise activation suppressing. *arXiv preprint arXiv:2103.08307*, 2021.

[6] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[7] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[8] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[9] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018.

[10] Jason Bunk, Srinjoy Chattopadhyay, BS Manjunath, and Shivkumar Chandrasekaran. Adversarially optimized mixup for robust classification. *arXiv preprint arXiv:2103.11589*, 2021.

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

[12] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang, and John C Duchi. Unlabeled data improves adversarial robustness. *arXiv preprint arXiv:1905.13736*, 2019.

[13] Alvin Chan, Yi Tay, and Yew-Soon Ong. What it thinks is important is important: Robustness transfers through input gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 332–341, 2020.

[14] Chen Chen, Jingfeng Zhang, Xilie Xu, Tianlei Hu, Gang Niu, Gang Chen, and Masashi Sugiyama. Guided interpolation for adversarial training. *arXiv e-prints*, pages arXiv–2102, 2021.

[15] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pages 2196–2205. PMLR, 2020.

[16] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

[17] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[18] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E Kounavis, and Duen Horng Chau. Compression to the rescue: Defending from adversarial attacks across modalities. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2018.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[20] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy D Bernstein, Jean Kossaifi, Aran Khanna, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018.

[21] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945*, 2019.

[22] Micah Goldblum, Liam Fowl, Soheil Feizi, and Tom Goldstein. Adversarially robust distillation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. doi: 10.1609/aaai.v34i04.5816.

[23] Micah Goldblum, Liam Fowl, Soheil Feizi, and Tom Goldstein. Adversarially robust distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3996–4003, 2020.

[24] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[25] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020. URL `https://arxiv.org/pdf/2010.03593`.

[26] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[27] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[29] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. *Proceedings of the International Conference on Machine Learning*, 2019.

[30] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1921–1930, 2019.

[31] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3779–3787, 2019.

[32] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[33] Daniel Hsu, Ziwei Ji, Matus Telgarsky, and Lan Wang. Generalization bounds via distillation. *arXiv preprint arXiv:2104.05641*, 2021.

[34] Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*, 2017.

[35] Andrew Ilyas, Shibani Santurkar, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

[36] Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[37] Animesh Koratana, Daniel Kang, Peter Bailis, and Matei Zaharia. Lit: Learned intermediate representation training for model compression. In *International Conference on Machine Learning*, pages 3509–3518. PMLR, 2019.

[38] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

[39] Saehyung Lee, Hyungyu Lee, and Sungroh Yoon. Adversarial vertex mixup: Toward better adversarially robust generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 272–281, 2020.

[40] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018.

[41] Wei-An Lin, Chun Pong Lau, Alexander Levine, Rama Chellappa, and Soheil Feizi. Dual manifold adversarial robustness: Defense against lp and non-lp adversarial attacks. *Advances in Neural Information Processing Systems Foundation (NeurIPS)*, 2020.

[42] Qi Liu, Tao Liu, Zihao Liu, Yanzhi Wang, Yier Jin, and Wujie Wen. Security analysis and enhancement of model compressed deep learning systems under adversarial attacks. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 721–726. IEEE, 2018.

[43] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *International Conference on Learning Representations*, 10:3, 2017.

[44] Xingjun Ma, Yuhao Niu, Lin Gu, Yisen Wang, Yitian Zhao, James Bailey, and Feng Lu. Understanding adversarial attacks on deep learning based medical image analysis systems. *Pattern Recognition*, 110: 107332, 2021.

[45] Divyam Madaan, Jinwoo Shin, and Sung Ju Hwang. Adversarial neural pruning with latent vulnerability suppression. In *International Conference on Machine Learning*, pages 6575–6585. PMLR, 2020.

[46] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[47] Chengzhi Mao, Ziyuan Zhong, Junfeng Yang, Carl Vondrick, and Baishakhi Ray. Metric learning for adversarial robustness. *arXiv preprint arXiv:1909.00900*, 2019.

[48] Aamir Mustafa, Salman H Khan, Munawar Hayat, Roland Goecke, Jianbing Shen, and Ling Shao. Deeply supervised discriminative learning for adversarial defense. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[49] Amir Najafi, Shin-ichi Maeda, Masanori Koyama, and Takeru Miyato. Robustness to adversarial perturbations in learning from incomplete data. *arXiv preprint arXiv:1905.13021*, 2019.

[50] Preetum Nakkiran. Adversarial robustness may be at odds with simplicity. *arXiv preprint arXiv:1901.00532*, 2019.

[51] Tianyu Pang, Kun Xu, and Jun Zhu. Mixup inference: Better exploiting mixup to defend adversarial attacks. *arXiv preprint arXiv:1909.11515*, 2019.

[52] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. *arXiv preprint arXiv:2010.00467*, 2020.

[53] Tianyu Pang, Xiao Yang, Yinpeng Dong, Kun Xu, Jun Zhu, and Hang Su. Boosting adversarial training with hypersphere embedding. *arXiv preprint arXiv:2002.08619*, 2020.

[54] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[55] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[56] Adnan Siraj Rakin, Jinfeng Yi, Boqing Gong, and Deliang Fan. Defend deep neural networks against adversarial examples via fixed and dynamic quantized activation functions. *arXiv preprint arXiv:1807.06714*, 2018.

[57] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020.

[58] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[59] Andrew Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[60] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *arXiv preprint arXiv:2007.08489*, 2020.

[61] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Mądry. Adversarially robust generalization requires more data. *arXiv preprint arXiv:1804.11285*, 2018.

[62] Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. Hydra: Pruning adversarially robust neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 7, 2020.

[63] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019.

[64] Ali Shafahi, Parsa Saadatpanah, Chen Zhu, Amin Ghiasi, Christoph Studer, David Jacobs, and Tom Goldstein. Adversarially robust transfer learning. In *International Conference on Learning Representations*, 2019.

[65] Zhiqiang Shen, Zhankui He, and Xiangyang Xue. Meal: Multi-model ensemble via adversarial learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4886–4893, 2019.

[66] Mayank Singh, Abhishek Sinha, Nupur Kumari, Harshitha Machiraju, Balaji Krishnamurthy, and Vineeth N Balasubramanian. Harnessing the vulnerability of latent layers in adversarially trained models. *arXiv preprint arXiv:1905.05186*, 2019.

[67] Suraj Srinivas and François Fleuret. Knowledge transfer with jacobian matching. In *International Conference on Machine Learning*, pages 4723–4731. PMLR, 2018.

[68] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[69] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2019.

[70] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[71] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[72] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1365–1374, 2019.

[73] Jonathan Uesato, Jean-Baptiste Alayrac, Po-Sen Huang, Robert Stanforth, Alhussein Fawzi, and Pushmeet Kohli. Are labels required for improving adversarial robustness? *arXiv preprint arXiv:1905.13725*, 2019.

[74] Bao Wang, Alex T Lin, Wei Zhu, Penghang Yin, Andrea L Bertozzi, and Stanley J Osher. Adversarial defense via data dependent activation function and total variation minimization. *arXiv preprint arXiv:1809.08516*, 2018.

[75] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2019.

[76] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

[77] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. Does network width really help adversarial robustness? *arXiv preprint arXiv:2010.01279*, 2020.

[78] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33, 2020.

[79] Chang Xiao, Peilin Zhong, and Changxi Zheng. Resisting adversarial attacks by k-winners-take-all. *arXiv preprint arXiv:1905.10510*, 1(2), 2019.

[80] Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. In *ICLR*, 2020.

[81] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019.

[82] Cihang Xie, Mingxing Tan, Boqing Gong, Alan Yuille, and Quoc V Le. Smooth adversarial training. *arXiv preprint arXiv:2006.14536*, 2020.

[83] Kaidi Xu, Sijia Liu, Gaoyuan Zhang, Mengshu Sun, Pu Zhao, Quanfu Fan, Chuang Gan, and Xue Lin. Interpreting adversarial examples by activation promotion and suppression. *arXiv preprint arXiv:1904.02057*, 2019.

[84] Shaokai Ye, Kaidi Xu, Sijia Liu, Hao Cheng, Jan-Henrik Lambrechts, Huan Zhang, Aojun Zhou, Kaisheng Ma, Yanzhi Wang, and Xue Lin. Adversarial robustness vs. model compression, or both? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 111–120, 2019.

[85] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4133–4141, 2017.

[86] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

[87] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *NIN*, 8:35–67, 2016.

[88] Runtian Zhai, Tianle Cai, Di He, Chen Dan, Kun He, John Hopcroft, and Liwei Wang. Adversarially robust generalization just requires more unlabeled data. *arXiv preprint arXiv:1906.00555*, 2019.

[89] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[90] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. *arXiv preprint arXiv:1905.00877*, 2019.

[91] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR, 2019.

[92] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[93] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *International Conference on Machine Learning*, pages 11278–11287. PMLR, 2020.

[94] Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. *arXiv preprint arXiv:2010.01736*, 2020.

[95] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? *arXiv preprint arXiv:2010.04819*, 2020.

[96] Tianyuan Zhang and Zhanxing Zhu. Interpreting adversarially trained convolutional neural networks. In *International Conference on Machine Learning*, pages 7502–7511. PMLR, 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section 6.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In the supplementary material and in the Section 6.3.

(c) Did you discuss any potential negative societal impacts of your work? [N/A]

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes] In Section 4.1 and supplementary material.

(b) Did you include complete proofs of all theoretical results? [Yes] In the supplementary material.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code and the data are proprietary.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In Section 6 and in the supplementary material.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In the supplementary material.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [N/A]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]