Adversarially Robust Distillation

Micah Goldblum∗
Department of Mathematics
University of Maryland
goldblum@math.umd.edu

Liam Fowl∗
Department of Mathematics
University of Maryland
lfowl@math.umd.edu

Soheil Feizi
Department of Computer Science
University of Maryland
sfeizi@cs.umd.edu

Tom Goldstein
Department of Computer Science
University of Maryland
tomg@cs.umd.edu

Abstract

Knowledge distillation is effective for producing small high-performance neural networks for classification, but these small networks are vulnerable to adversarial attacks. We first study how robustness transfers from robust teacher to student network during knowledge distillation. We find that a large amount of robustness may be inherited by the student even when distilled on only clean images. Second, we introduce Adversarially Robust Distillation (ARD) for distilling robustness onto small student networks. ARD is an analogue of adversarial training but for distillation. In addition to producing small models with high test accuracy like conventional distillation, ARD also passes the superior robustness of large networks onto the student. In our experiments, we find that ARD student models decisively outperform adversarially trained networks of identical architecture on robust accuracy. Finally, we adapt recent fast adversarial training methods to ARD for accelerated robust distillation.

1 Introduction

State-of-the-art deep neural networks for many computer vision tasks have tens of millions of parameters, hundreds of layers, and require billions of operations per inference [9, 25]. However, networks are often deployed on mobile devices with limited compute and power budgets or on web servers without GPUs. Such applications require efficient networks with light-weight inference costs [7, 21, 4, 27].

Knowledge distillation was introduced as a way to transfer the knowledge of a large pre-trained teacher network to a smaller light-weight student network [10]. Instead of training the student network on one-hot class labels, distillation involves training the student network to emulate the outputs of the teacher. Knowledge distillation yields compact student networks that surpass the performance achievable by training from scratch without a teacher [10].

Classical distillation methods achieve high efficiency and accuracy but neglect security. Standard neural networks are easily fooled by adversarial examples, in which small perturbations to inputs cause mis-classification [8, 26]. This phenomenon leads to major security vulnerabilities for high-stakes applications like self-driving cars and medical diagnosis [22, 12]. In such domains, efficiency and accuracy are not enough - networks must also be robust.

∗Authors contributed equally.

Preprint. Under review.
We study distillation methods that produce robust student networks. Unlike conventional adversarial training \cite{15, 24}, which encourages a network to output correct labels within an $\epsilon$-ball of training samples, the proposed Adversarially Robust Distillation (ARD) instead encourages student networks to mimic their teacher’s output within an $\epsilon$-ball of training samples (See Figure 1). Thus, ARD is a natural analogue of adversarial training but in the context of distillation. Formally, we solve the minimax problem

$$\min_{\theta} \mathbb{E}_{(X,y) \sim D} \left[ \alpha t^2 \text{KL}(S^T_\theta(X+\delta_\theta), T^t(X)) + (1 - \alpha) \ell(S^T_\theta(X), y) \right],$$

where $\delta_\theta = \arg \max_{\|\delta\|_p < \epsilon} \ell(S^T_\theta(X + \delta), y)$, $T$ and $S$ are teacher and student networks, and $D$ is the data generating distribution. See Section 5 for a more thorough description. Below, we summarize our contributions in this paper:

- We show that knowledge distillation using only natural images can preserve much of the teacher’s robustness to adversarial attacks (see Table 1), enabling the production of efficient robust models without the cost of adversarial training.
- We introduce Adversarially Robust Distillation (ARD) for producing small robust student networks. In our experiments, ARD students exhibit higher robust accuracy than adversarially trained models with identical architecture, and ARD often exhibits higher natural accuracy simultaneously. Interestingly, ARD students may exhibit even higher robust accuracy than their teacher (see Table 1).
- We accelerate our method for efficient training by adapting fast adversarial training methods.

| Model             | Robust Accuracy ($A_{adv}$) |
|-------------------|-----------------------------|
| AT ResNet18 teacher | 44.46%                      |
| AT ResNet18 $\rightarrow$ MobileNetV2 | 38.21%                      |
| AT ResNet18 $\rightarrow$ MobileNetV2 | 50.22%                      |

Table 1: Performance of an adversarially trained (AT) teacher network and its student on CIFAR-10, where robust accuracy ($A_{adv}$) is with respect to a 20-step PGD attack as in \cite{15}. "→" denotes "knowledge distillation onto". "ARD" denotes "adversarially robust distillation onto".

Table 1 shows that a student may learn robust behavior from a robust teacher, even if it only sees clean images during distillation. Our method, ARD, outperforms knowledge distillation for producing robust students. To gain intuition for the differences between these methods, we visualize decision boundaries for a toy problem in Figure 2. Randomly generated training data are depicted as colored dots with boxes showing the desired $\ell_\infty$ robustness radius. Background colors represent the classification regions of the networks (10-layer teacher and 5-layer student). In each case, the network achieves perfect training accuracy. A training point is vulnerable to attack if its surrounding box contains multiple colors. Results in Figure 2 are consistent with our experiments in Section 5 on more complex datasets.

![Figure 1](image1.png)

Figure 1: Adversarially Robust Distillation (ARD) works by minimizing discrepancies between softmax outputs of the teacher on natural images and outputs of a student on adversarial images.
Figure 2: A student distilled from a robust teacher using knowledge distillation is more robust than a naturally trained network, but ARD produces a more robust network than either and closely mimics the teacher’s decision boundary. Adversarially vulnerable training points have $\ell_\infty$ boxes outlined in red.

2 Related Work

Early schemes for compressing neural networks involved binarized weights to reduce storage and computation costs [4]. Other early efforts focused on speeding up calculations via low-rank regularization and pruning weights to reduce computation costs [27, 13]. Knowledge distillation teaches a student network to mimic a more powerful teacher [10]. The student network is usually a small, lightweight architecture like MobileNetV2 [21].

Knowledge distillation has also been adapted for robustness in a technique called defensive distillation [19]. In this setting, the teacher and student networks have identical architectures. An initial network is trained on the class labels of a problem and then distilled at temperature $t$ onto a network of identical architecture. Defensive distillation improves robustness to a certain $\ell_1$ attack, while maintaining natural accuracy [19]. However, defensive distillation gains robustness due to gradient masking, and this defense has been broken using $\ell_0$, $\ell_\infty$, and $\ell_2$ attacks [1, 2].

Various methods exist that modify networks to achieve robustness. Some model-specific variants of adversarial training utilize surrogate loss functions to minimize the difference in network output on clean and adversarial data [31, 16], feature denoising blocks [29], and adversarial logit pairing to squeeze logits from clean and adversarial inputs [11]. Still other methods, such as JPG compression, pixel deflection, and image superresolution, are model-agnostic and seek to minimize the effect of adversarial examples by transforming inputs to the network [7, 20, 18].

Recently, there has been work on defensive-minded compression. The authors of [28] and [32] study the preservation of robustness under quantization. Defensive Quantization involves quantizing a network while minimizing the Lipschitz constant to encourage robustness [14]. While quantization reduces space complexity, it does not reduce the number of Multiply-Add (MAdd) operations needed for inference, although operations performed at lower precision may be faster depending on hardware. Moreover, Defensive Quantization is not evaluated against strong attackers.

3 Problem Setup

Knowledge distillation employs a teacher-student paradigm in which a small student network learns to mimic the output of an often much larger teacher model [10]. Knowledge distillation in its purist form entails the minimization problem,

$$\min_{\theta} \ell_{\text{KD}}(\theta), \quad \ell_{\text{KD}}(\theta) = \mathbb{E}_{X \sim \mathcal{D}} \left[ \text{KL}(S_{\theta}(X), T_t(X)) \right], \quad (2)$$

where KL is KL divergence, $S$ is a student network with parameters $\theta$ and temperature $t$, $T$ is a teacher network with temperature $t$, and $X$ is an input to the networks drawn from data generating distribution $\mathcal{D}$. The temperature constant refers to a number by which the logits are divided before feeding them into the softmax function. Intuitively, knowledge distillation involves minimizing the average distance from the student’s output to the teacher’s output over data from a distribution. The softmax outputs of the teacher network, also referred to as soft labels, may be more informative than true data labels alone. In [10], the authors suggest using a linear combination of the loss function, $\ell_{\text{KD}}(\theta)$, and the cross-entropy between the softmax output of the student network and the one-hot
vector representing the true label in order to improve natural accuracy of the student model, especially on difficult datasets. In this case, we have

$$\ell_{\text{KD}}(\theta) = \mathbb{E}_{(X,d) \sim D} \left[ \alpha t^2 \text{KL}(S_\theta^t(X), T^t(X) + (1 - \alpha)\ell(S_\theta^t(X), y)) \right],$$  

(3)

where $\ell$ is the standard cross-entropy loss, and $y$ is the label. In our experiments, we use $\alpha = 1$ except where otherwise noted. We investigate if students trained using this method inherit their teachers’ robustness to adversarial attacks. We also combine knowledge distillation with adversarial training.

Adversarial training is another method for encouraging robustness to adversarial attacks during training [24]. Adversarial training involves the minimax optimization problem,

$$\min_\theta \mathbb{E}_{(X,y) \sim D} \left[ \max_{\|\delta\|_p < \epsilon} L_\theta(X + \delta, y) \right],$$  

(4)

where $L_\theta(X + \delta, y)$ is the loss of network with parameters $\theta$, input $X$ perturbed by $\delta$, and label $y$. Adversarial training encourages a student to produce the correct label in an $\epsilon$-ball surrounding data points. Virtual Adversarial Training (VAT) and TRADES instead use as a loss function a linear combination of cross-entropy loss and KL divergence between the network’s softmax output from clean input and from adversarial input [16, 31]. The KL divergence term acts as a consistency regularizer which trains the neural network to produce identical output on a natural image and adversarial images generated from the natural image. As a result, this term encourages the neural network’s output to be constant in $\epsilon$-balls surrounding data points.

Knowledge distillation is useful for producing accurate student networks when highly accurate teacher networks exist. However the resulting student networks may not be robust to adversarial attacks [2]. We combine knowledge distillation and adversarial training to similarly produce robust student networks when robust teacher networks exist.

We focus on adversarial robustness to $\ell_\infty$ attacks since these are pervasive in the literature. Thus, we carry out both adversarial training and ARD with FGSM-based PGD $\ell_\infty$ attacks similarly to [15, 31]. We re-implemented the methods from these papers to perform adversarial training. In our experiments, we use WideResNet (34-10) and ResNet18 teacher models as well as MobileNetV2 students. Adversarial training and TRADES teacher models are as described in [15, 31]. See appendix sections A and G for more information on models, attacks, and training.

4 Adversarial robustness is preserved under knowledge distillation

It is now accepted that distillation from a naturally trained teacher does not produce robust models. But if we distill from an adversarially hardened teacher, will the student learn to be robust?

If robustness transfers from teacher to student, we can produce accurate, robust, and efficient student networks. Moreover, since adversarial training is slow due to the bottleneck of crafting adversarial samples for every batch, we could create many different student networks from one teacher, and adversarial training would only need to be performed once. This routine of training a robust teacher and then distilling onto robust students would be far more time efficient than training many robust student networks individually.

4.1 Non-robust teachers produce non-robust students

Knowledge distillation and defensive distillation with non-robust teacher networks are known to produce student networks that are vulnerable to a variety of adversarial attacks [2]. To establish a baseline for comparison, we distill using non-robust ResNet18 as a teacher, and evaluate against the same 20-step PGD attack on which we will evaluate our models. These results can be found in Table 2.
Table 2: Performance of a naturally trained teacher network and its students distilled (with $t = 30$) on CIFAR-10, where robust accuracy is with respect to a 20-step PGD attack as in [15]. $A_{nat}$ denotes natural accuracy.

| Model                  | $A_{nat}$ | $A_{adv}$ |
|------------------------|-----------|-----------|
| ResNet18 teacher       | 94.75%    | 0.0%      |
| ResNet18 $\rightarrow$ ResNet18 | 94.92%    | 0.0%      |
| ResNet18 $\rightarrow$ MobileNetV2 | 93.53%    | 0.0%      |

4.2 Robust teachers can produce robust students, even distilling on only clean data

Next, we substitute in a robust adversarially trained ResNet18 teacher network, and run the same experiments. We find that our new student networks are far more robust than students of the non-robust teacher (see Table 3). In fact, the student networks learn most of the teacher’s robust accuracy.

Table 3: Performance of an adversarially trained ResNet18 teacher network and student networks of various size distilled on CIFAR-10, where robust accuracy is with respect to a 20-step PGD attack as in [15].

| Student Model            | $A_{nat}$ | $A_{adv}$ |
|--------------------------|-----------|-----------|
| AT ResNet18 teacher      | 76.54%    | 44.46%    |
| AT ResNet18 $\rightarrow$ ResNet18 | 76.13%    | 40.13%    |
| AT ResNet18 $\rightarrow$ MobileNetV2 | 76.86%    | 38.21%    |

4.3 Not all robust networks are good teachers, and robustness does not transfer on some datasets

In the previous experiments, we see that a student network may inherit a significant amount of robustness from a robust teacher network during knowledge distillation. However, some robust teachers are not conducive to this robustness transfer. We use robust WideResNet (34-10) models trained using adversarial training and TRADES to show that these models do not transfer robustness well during knowledge distillation (See Table 4). Additionally, a robust WRN teacher model transfers almost no robustness under knowledge distillation against 20-step PGD untargeted attacks on CIFAR-100, a much harder dataset for robustness than CIFAR-10 (See Table 5).

For these reasons, we develop ARD for distilling a variety of teachers onto produce robust students. Using this methods, robustness is preserved on architectures and datasets that do not transfer robustness under vanilla distillation.

Table 4: Robust WRN34-10 teacher models and their students on CIFAR-10, where robust accuracy is with respect to a 20-step PGD attack as in [15].

| Model                  | $A_{nat}$ | $A_{adv}$ |
|------------------------|-----------|-----------|
| AT WRN teacher         | 84.41%    | 45.75%    |
| TRADES WRN teacher     | 84.92%    | 56.61%    |
| AT WRN $\rightarrow$ MN | 92.49%    | 5.46%     |
| TRADES WRN $\rightarrow$ MN | 85.6%     | 21.69%    |

Table 5: Robust teacher network and its students on CIFAR-100, where robust accuracy is with respect to a 20-step PGD attack as in [15].

| Model                  | $A_{nat}$ | $A_{adv}$ |
|------------------------|-----------|-----------|
| AT WRN teacher         | 59.9%     | 28.36%    |
| AT WRN $\rightarrow$ MN | 25.54%    | 1.30%     |
| AT WRN $\rightarrow$ MN ($\alpha = 0.95$) | 75.94%    | 0.02%     |
| AT WRN $\rightarrow$ MN ($\alpha = 0.93$) | 76.38%    | 0.00%     |
5 Improving the robustness of student models with Adversarially Robust Distillation (ARD)

We combine knowledge distillation, adversarial training, and TRADES/VAT to produce small robust student models from much larger robust teacher models using a method we call Adversarially Robust Distillation. ARD not only produces more robust students than knowledge distillation, but it also works for teachers and datasets on which knowledge distillation is ineffective for transferring robustness. Our procedure is a natural analogue of adversarial training but in a distillation setting. During standard adversarial training, we force a network to produce the ground truth label corresponding to a clean input when the network is exposed to an adversary. Along the same lines, our method treats the teacher network’s softmax output on clean data as the ground truth and trains a student network to reproduce this ground truth when exposed to adversarial examples. We start with a robust teacher model, $T$, and we train the student model $S^\theta$ by solving the following optimization problem:

$$\min_\theta \mathbb{E}_{(X,y) \sim D} \left[ \alpha t^2 \text{KL}(S^\theta(X + \delta^\theta), T^t(X)) + (1 - \alpha)\ell(S^\theta(X), y) \right],$$

where $\delta^\theta = \arg \max_{\|\delta\|_p < \epsilon} \ell(S^\theta(X + \delta), y)$, $\ell$ is cross-entropy loss and we divide the logits of both student and teacher models by temperature term $t$ during training. The $t^2$ term is used as in [10] since dividing the logits shrinks the gradient. The cross-entropy loss, which encourages natural accuracy, can be seen as a standard training term. Thus, $\alpha$ is a hyperparameter that prioritizes similarity to the teacher over natural accuracy. In our experiments, we set $\alpha = 1$ except where otherwise noted, eliminating the cross-entropy term. Our training routine involves the following procedure:

**Algorithm 1:** Adversarially Robust Distillation (ARD)

**Require:** Student and teacher networks $S$ and $T$, learning rate $\gamma$, dataset $\{(x_i, y_i)\}$, number of steps, $K$, per PGD attack, and $\epsilon$ maximum attack radius.

Initialize $\theta$, the weights $S$;

for Epoch = 1,...,$N_{\text{epochs}}$ do

for Batch = 1,...,$N_{\text{batches}}$ do

Construct adversarial example $x'_i$ for each $x_i \in$ Batch by maximizing cross-entropy between $S^\theta(x'_i)$ and $y_i$ constrained to $\|x_i - x'_i\|_p < \epsilon$ using $K$-step PGD.

Compute $\nabla_\theta \ell_{\text{ARD}}(\{x_i\}, \theta) = \sum_i \nabla_\theta [\alpha t^2 \text{KL}(S^\theta(x'_i), T^t(x_i)) + (1 - \alpha)\ell(S^\theta(X), y)]$, over the current batch.

$\theta \leftarrow \theta - \gamma \nabla_\theta \ell_{\text{ARD}}(\{x_i\}, \theta)$

end for

end for

See appendix sections C and G for additional information about training and our choice of loss function.

5.1 ARD works with large teachers that fail to transfer robustness under knowledge distillation

In Section[4] we saw that that some teacher networks do not readily transfer robustness to students. We see through our experiments in Table[5] that ARD is able to create robust students from teachers whose robustness failed to transfer during conventional knowledge distillation. TRADES and adversarially trained MobileNetV2 models are each simultaneously beaten in natural and robust accuracy by an ARD variant. In these experiments, the WideResnet teacher model contains 20× as many parameters and performs 70× as many MAdd operations as the MobileNetV2 student.
| Model                        | \(A_{nat}\) | \(A_{adv}\) |
|------------------------------|--------------|--------------|
| TRADES WRN teacher          | 84.92%      | 56.61%      |
| AT MobileNetV2              | 80.50%      | 46.90%      |
| TRADES MobileNetV2          | 83.59%      | 44.79%      |
| TRADES WRN \(\xrightarrow{\text{ARD}}\) MobileNetV2 | 82.63% | **50.42%** |
| TRADES WRN \(\xrightarrow{\text{ARD}}\) MobileNetV2 (\(\alpha = 0.95\)) | 84.70% | 46.28% |

Table 6: Performance on CIFAR-10, where robust accuracy is with respect to a 20-step PGD attack as in [15].

5.2 ARD works on datasets where knowledge distillation fails

In Section 4, we saw that a student network inherited little to no robustness from an adversarially trained teacher network on CIFAR-100. This dataset is very difficult to protect from untargeted attacks because it contains many classes that are similar in appearance. In Table 7, we see that a MobileNetV2 model trained on CIFAR-100 using ARD from an adversarially trained WideResNet is more robust than an adversarially trained MobileNetV2. In fact, the ARD model is nearly as robust as its teacher.

| Model                        | \(A_{nat}\) | \(A_{adv}\) |
|------------------------------|--------------|--------------|
| AT WRN teacher              | 59.90%      | 28.36%      |
| AT MobileNetV2              | 55.62%      | 22.80%      |
| AT WRN \(\xrightarrow{\text{ARD}}\) MobileNetV2 (\(\alpha = 0.93\)) | 55.47% | **27.64%** |

Table 7: Performance on CIFAR-100, where robust accuracy is with respect to a 20-step PGD attack as in [15].

5.3 ARD can produce networks more robust than their teacher

In some experiments, ARD student networks are more robust than their teacher. Interestingly, this behavior does not depend on distilling from a high-capacity network to a low capacity network; distilling ResNet18 onto itself and MobileNetV2 onto itself using ARD results in better robustness than adversarial training alone.

We seek to understand if this increased robustness is caused by differences between the student and teacher architectures, and so we use ARD to distill a teacher network onto a student network with an identical architecture. In our experiments, ARD boosted the robustness of both the ResNet18 and MobileNetV2 models (See Table 8).

| Model                        | \(A_{nat}\) | \(A_{adv}\) |
|------------------------------|--------------|--------------|
| AT ResNet18                  | 76.54%      | 44.46%      |
| AT MobileNetV2               | 80.50%      | 46.90%      |
| AT ResNet18 \(\xrightarrow{\text{ARD}}\) ResNet18 | 79.49% | **51.21%** |
| AT MobileNetV2 \(\xrightarrow{\text{ARD}}\) MobileNetV2 | 81.22% | 47.95% |
| AT ResNet18 \(\xrightarrow{\text{ARD}}\) MobileNetV2 | 79.47% | 50.22% |

Table 8: Performance on CIFAR-10, where robust accuracy is with respect to a 20-step PGD attack as in [15].
5.4 Accelerating ARD using fast adversarial training methods

The ARD procedure described above takes approximately the same amount of time as adversarial training. Adversarial training is slow since it requires far more gradient calculations than natural training. Several methods have been proposed recently for performing adversarial training in the same amount of time as natural training [23, 30]. We demonstrate accelerated performance for ARD by adapting the “free” adversarial training from [23] for distillation.

This version, Fast-ARD, described in Algorithm 2, is equally fast to plain knowledge distillation. During training, we replay each mini-batch several times in a row. On each replay, we simultaneously compute the gradient of the loss w.r.t. the image and parameters using the same backward pass. Then, we update the adversarial attack and the network’s parameters simultaneously. Empirically, Fast-ARD produces less robust students than the full ARD above, but it produces higher robust accuracy compared to free trained models with identical architecture as seen in Table 9. Furthermore, Fast-ARD from a TRADES WideResNet onto MobileNetV2 produces a more robust student than our most robust MobileNetV2 produced during knowledge distillation and in the same amount of training time. Our accelerated algorithm is detailed in Algorithm 2.

Algorithm 2: Fast-ARD with free adversarial training

Requires: Student and teacher networks $S$ and $T$, learning rate $\gamma$, norm $p$, dataset $\{(x_i, y_i)\}$, and attack step-size $r$ and radius $\epsilon$

Initialize $\theta$, the weights of network $S$, and set $\delta = 0$.

for Epoch = 1,..., $\frac{N_{epochs}}{m}$ do
  for Batch = 1,..., $\frac{N_{batches}}{m}$ do
    for $j = 1,..., m$ do
      For $x_i \in$ Batch, find new perturbation $\delta_i'$ by maximizing cross-entropy between $S_\theta(x_i + \delta_i + \delta_i')$ and $y_i$, over $\delta_i'$, constrained to $\|\delta_i + \delta_i'\|_p < \epsilon$, using a 1-step PGD attack.
      Compute the gradient of the loss function $\nabla_\theta \ell_{\text{ARD}}(\{x_i\}, \theta) = \sum_i \nabla_\theta [\alpha t^2 \text{KL}(S_\theta(x'_i), T_\theta(x_i)) + (1 - \alpha)\ell(S_\theta(x'_i), y_i)]$, over current batch.
      $\delta_i \leftarrow \delta_i + \delta_i'$
      $\theta \leftarrow \theta - \gamma \nabla_\theta \ell_{\text{ARD}}(\{x_i\}, \theta)$

Table 9: Performance of MobileNetV2 classifiers, free trained and Fast-ARD, on CIFAR-10, where robust accuracy is with respect to a 20-step PGD attack as in [15]. "F-ARD\,\rightarrow\,\rightarrow" denotes "Fast-ARD onto".

5.5 ARD and Fast-ARD models are more robust than their adversarially trained counterparts

While 20-step PGD is a powerful attack, we also test ARD against other $\ell_\infty$ attackers including Momentum Iterative Fast Gradient Sign Method [6], DeepFool [17], 1000-step PGD, and PGD with random restarts. We find that our ARD and Fast-ARD models outperform adversarial training and free training respectively across all attacks we tried (see Table 10).
| Model                      | MI-FGSM_{20} | DeepFool | 1000-PGD | 20-PGD 100-restarts |
|---------------------------|--------------|----------|----------|---------------------|
| AT MN                     | 50.82%       | 57.74%   | 46.51%   | 46.79%              |
| $\xrightarrow{\text{ARD}}$ MN | 55.16%       | 64.61%   | 49.98%   | 50.30%              |
| Free trained MN (m=4)     | 30.60%       | 41.09%   | 22.23%   | 22.94%              |
| $\xrightarrow{\text{F-ARD}}$ MN (m=4) | 44.78%       | 60.03%   | 36.01%   | 36.88%              |

Table 10: Robust validation accuracy of adversarially trained and free trained MobileNetV2 and TRADES WRN ARD (and Fast-ARD) onto MobileNetV2 on CIFAR-10 under various attacks. All attacks use $\epsilon = \frac{8}{255}$.

6 Discussion

We find that knowledge distillation allows a student network to absorb a large amount of a teacher network’s robustness to adversarial attacks, even when the student is only trained on clean data. Moreover, we see that the student model loosely mimics the decision boundary of the teacher model in simple cases. However, in some cases, a distilled student model is still far less robust than the teacher. In these same simple cases, the ARD student closely mimics the decision boundary of the teacher. To improve student robustness, we introduce Adversarially Robust Distillation (ARD) for producing robust student models from robust teachers. In our experiments, student models trained using our method outperform similar networks trained using adversarial training in robust and sometimes natural accuracy. Furthermore, we adapt free adversarial training for our method and demonstrate accelerated performance.

Recent work on distillation has produced significant improvements over vanilla knowledge distillation [3]. We believe that Knowledge Distillation with Feature Maps could improve both natural and robust accuracy of student networks. Adaptive data augmentation like AutoAugment [5] may also improve performance of both normal knowledge distillation for robust teachers and ARD. Finally, with the recent publication of fast adversarial training methods [23, 30], we hope to further accelerate ARD.

Our implementation of ARD can be found at github.com/goldblum/AdversariallyRobustDistillation

Acknowledgments

Tom Goldstein was supported by the Office of Naval Research (N00014-17-1-2078), DARPA Lifelong Learning Machines (FA8650-18-2-7833), the DARPA YFA program (D18AP0055), and the Sloan Foundation.
References

[1] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. arXiv preprint [arXiv:1607.04311], 2016.

[2] 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.

[3] Wei-Chun Chen, Chia-Che Chang, Chien-Yu Lu, and Che-Rung Lee. Knowledge distillation with feature maps for image classification. arXiv preprint [arXiv:1812.06660], 2018.

[4] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint [arXiv:1602.02830], 2016.

[5] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. arXiv preprint [arXiv:1805.09501], 2018.

[6] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 9185–9193, 2018.

[7] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. arXiv preprint [arXiv:1608.00853], 2016.

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

[9] 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.

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

[11] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. arXiv preprint [arXiv:1803.06373], 2018.

[12] June-Goo Lee, Sanghoon Jun, Young-Won Cho, Hyunna Lee, Guk Bae Kim, Joon Beom Seo, and Namkug Kim. Deep learning in medical imaging: general overview. Korean journal of radiology, 18(4):570–584, 2017.

[13] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. arXiv preprint [arXiv:1608.08710], 2016.

[14] Ji Lin, Chuang Gan, and Song Han. Defensive quantization: When efficiency meets robustness. arXiv preprint [arXiv:1904.08444], 2019.

[15] 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.

[16] Takeru Miyato, Shin-ichi Maeda, Shin Ishii, and Masanori Koyama. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. IEEE transactions on pattern analysis and machine intelligence, 2018.

[17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2574–2582, 2016.

[18] Aamir Mustafa, Salman H Khan, Munawar Hayat, Jianbing Shen, and Ling Shao. Image super-resolution as a defense against adversarial attacks. arXiv preprint [arXiv:1901.01677], 2019.

[19] 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.

[20] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 8571–8580, 2018.
[21] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4510–4520, 2018.

[22] Eder Santana and George Hotz. Learning a driving simulator. arXiv preprint arXiv:1608.01230, 2016.

[23] Ali Shafahi, Mahyar Najibi, Xu-Zheng Ghiasi, Amin, John Dickerson, Christof Studer, Larry S. Davis, Gavin Taylor, and T. Goldstein. Adversarial training for free! arXiv preprint arXiv:1904.12845, 2019.

[24] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. Neurocomputing, 307:195–204, 2018.

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2818–2826, 2016.

[26] 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.

[27] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. arXiv preprint arXiv:1511.06067, 2015.

[28] Arie Wahyu Wijayanto, Choong Jun Jin, Kaushalya Madhawa, and Tsuyoshi Murata. Robustness of compressed convolutional neural networks. In 2018 IEEE International Conference on Big Data (Big Data), pages 4829–4836. IEEE, 2018.

[29] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. arXiv preprint arXiv:1812.03411, 2018.

[30] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Painless adversarial training using maximal principle. arXiv preprint arXiv:1905.00877, 2019.

[31] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. arXiv preprint arXiv:1901.08573, 2019.

[32] Yiren Zhao, Ilia Shumailov, Robert Mullins, and Ross Anderson. To compress or not to compress: Understanding the interactions between adversarial attacks and neural network compression. arXiv preprint arXiv:1810.00208, 2018.
7 Appendix

7.1 Appendix A: Space and time efficiency of student and teacher models

We perform our primary experiments for ARD with WideResNet (34-10) and ResNet18 teacher models as well as a MobileNetV2 student model. We consider two network qualities for evaluating the efficiency of our teacher and student networks in order to quantify compression. First, we study space efficiency by counting the number of parameters in a network. Second, we study time complexity. To this end, we compute the total multiply-add (MAdd) operations performed during a single inference. The real time elapsed during this inference will also vary as a result of parallelization harnessed in implementation and deep learning framework, so we use MAdd, which is invariant under implementation and framework, to study time complexity.

The WideResNet and ResNet18 teacher models we employ contain \( \sim 46.2 \text{M} \) and \( \sim 11.2 \text{M} \) parameters respectively, while the MobileNetV2 student model contains \( \sim 2.3 \text{M} \) parameters. A forward pass through the WRN and ResNet18 teacher models takes \( \sim 13.3 \text{B} \) MAdd operations and \( \sim 1.1 \text{B} \) MAdd operations respectively, while a forward pass through the student model takes \( \sim 187 \text{M} \) MAdd operations. To summarize these network traits, compared to a WideResnet (34-10) teacher, the MobileNetV2 student model:

- Contains \( \sim 5\% \) as many parameters
- Performs \( \sim 1.4\% \) as many MAdd operations during a forward pass

7.2 Appendix B: The effects of temperature, \( \alpha \), and data augmentation for knowledge distillation of robust teacher models

We show the results of experiments in Table 11, varying temperature and \( \alpha \). We see that only very low temperature terms are not conducive to robustness preservation, but knowledge distillation is not highly sensitive to temperature, and a wide array of temperature terms are effective for producing robust students. On the other hand, robustness transfer decays rapidly when \( \alpha \) decreases. This effect produces a robustness-accuracy tradeoff.

Data augmentation yields dramatic improvements for robustness transfer as the student gets to see the teacher’s behavior at more data points, as can be seen in Table 12. A natural idea is to teach the student the teacher’s behavior at adversarial points as well. However, this data augmentation technique greatly decreases training speed and does not provide significant improvement.

| Temperature | \( \mathcal{A}_{\text{nat}} \) | \( \mathcal{A}_{\text{adv}} \) | \( \alpha \) | \( \mathcal{A}_{\text{nat}} \) | \( \mathcal{A}_{\text{adv}} \) |
|-------------|------------------|------------------|--------|------------------|------------------|
| 1           | 77.37%           | 35.44%           | 1.00   | 76.86%           | 38.21%           |
| 10          | 78.1%            | 38.55%           | 0.99   | 82.93%           | 28.36%           |
| 50          | 76.86%           | 38.21%           | 0.95   | 91.58%           | 9.65%            |
| 100         | 76.28%           | 38.5%            | 0.90   | 92.34%           | 7.96%            |
|             | 76.44%           | 38.65%           | 0.70   | 92.33%           | 7.54%            |
|             |                  |                  | 0.50   | 92.14%           | 2.73%            |

Table 11: Performance of adversarially trained ResNet18 teacher network distilled onto MobileNetV2 (MN) with different temperature terms and \( \alpha \) terms on CIFAR-10, where robust accuracy is with respect to a 20-step PGD attack as in [15].

7.3 Appendix C: Notes on loss functions for ARD

We tried other versions of the loss function with the addition of a (non-adversarial) knowledge distillation KL divergence term, which increases natural accuracy slightly but sharply decreases robust accuracy, and the addition of cross-entropy between the softmax of the adversarially attacked student and the one-hot label vectors, which is redundant in combination with the ARD loss and empirically harms robustness. We also explored using \( T^a(x'_i) \) instead of \( T^a(x_i) \) in the KL divergence term in the loss function, but the accuracy, both natural and robust, decreased. Additionally, passing an adversarial example through the teacher model will slow down training and sharply increase memory consumption if the teacher is much larger than the student. Our method does not require forward
Table 12: Performance of adversarially trained WideResNet distilled onto MobileNetV2 (MN) with various data augmentation on CIFAR-10, where robust accuracy is with respect to a 20-step PGD attack as in [15].

| Data Augmentation                              | $A_{nat}$ | $A_{adv}$ |
|-----------------------------------------------|-----------|-----------|
| No augmentation                               | 75.76%    | 32.05%    |
| Horizontal flips                              | 76.24%    | 35.39%    |
| Random crops                                  | 76.31%    | 37.31%    |
| Both flips and crops                          | 76.86%    | 38.21%    |
| Adversarial examples generated from teacher net| 76.76%    | 38.33%    |

passes through the teacher during training as long as logits of the training data are stored ahead of time. We explored generating adversarial attacks during training by maximizing KL divergence instead of cross-entropy. This technique lowers natural accuracy without improving robust accuracy.

In our training routine, we employ data augmentation in the form of random crops and horizontal flips. We intend to explore adaptive data augmentation methods tailored specifically for robust distillation.

7.4 Appendix D: ARD with naturally trained teacher models

ARD encourages a student to produce, for all images within an $\epsilon$-ball of a data point, the teacher’s output at that data point. Thus, it seems reasonable to try using ARD with a naturally trained teacher model. As evident in Table 13, naturally trained teachers produce robust students, but these students may be less robust than those of robust teachers and less robust than adversarially trained models with identical architecture.

| Model               | $A_{nat}$ | $A_{adv}$ |
|---------------------|-----------|-----------|
| ResNet18 $\xrightarrow{ARD}$ MobileNetV2 | 84.18%    | 44.61%    |
| WRN $\xrightarrow{ARD}$ MobileNetV2        | 84.43%    | 42.51%    |

Table 13: Performance of MobileNetV2 students distilled from naturally trained teachers on CIFAR-10, where robust accuracy is with respect to a 20-step PGD attack as in [15].

7.5 Appendix E: Improving the speed of ARD by reducing the number of attack steps

Another way to improve the speed of ARD training is to reduce the number of attack steps in order to reduce the number of gradient calculations. In our experiments shown in Table 14, reducing the number of attack steps improves natural accuracy while decreasing robust accuracy. This procedure has a similar effect to reducing $\alpha$. We suggest this strategy over reducing $\alpha$ since it has the added benefit of accelerating training.

| Attack Steps (training) | $A_{nat}$ | $A_{adv}$ |
|-------------------------|-----------|-----------|
| 4                       | 84.28%    | 46.49%    |
| 6                       | 83.38%    | 48.35%    |
| 10                      | 82.63%    | 50.42%    |

Table 14: Performance of TRADES WideResNet distilled onto MobileNetV2 using ARD with adversaries generated using different numbers of attack steps on CIFAR-10. Robust accuracy is with respect to a 20-step PGD attack as in [15].

7.6 Appendix F: Sensitivity of ARD to temperature and $\alpha$

Compared to knowledge distillation for preserving robustness, ARD is less sensitive to the temperature parameter. We found that varying the temperature parameter does not significantly impact either the natural or robust accuracy of the resulting student (See Table 15). Like with knowledge distillation, varying $\alpha$ presents an accuracy-robustness tradeoff. However, unlike with knowledge distillation, we
find that under ARD, robust accuracy decays far slower as $\alpha$ decreases so that ARD is less sensitive to this parameter.

| Temperature | $A_{nat}$ | $A_{adv}$ | $\alpha$ | $A_{nat}$ | $A_{adv}$ |
|-------------|-----------|-----------|----------|-----------|-----------|
| 1           | 81.41%    | 49.57%    | 1.00     | 82.63%    | 50.42%    |
| 5           | 82.24%    | 48.77%    | 0.95     | 84.7%     | 46.28%    |
| 10          | 82.33%    | 48.93%    | 0.90     | 86.58%    | 41.16%    |
| 30          | 82.63%    | 50.42%    | 0.70     | 90.57%    | 25.18%    |
| 50          | 82.14%    | 48.97%    | 0.50     | 83.00%    | 13.09%    |

Table 15: Performance of TRADES WideResNet distilled onto MobileNetV2 using ARD with different temperature terms and values of $\alpha$ on CIFAR-10, where robust accuracy is with respect to a 20-step PGD attack as in [15].

7.7 Appendix G: Our hyperparameters for training

We train our models for 200 epochs with SGD and a momentum term of $2(10^{-4})$. We tried using other optimizers such as ADAM, but SGD yields superior results. Fast-ARD models are trained for $200$ epochs so that they take the same amount of time as natural distillation. We use an initial learning rate of 0.1 and we decrease the learning rate by a factor of 10 on epochs 100 and 150 (epochs $\frac{100}{10}$ and $\frac{150}{10}$ for Fast-ARD). Except for our temperature experiments, we used a temperature term of $30$ for CIFAR-10 and $5$ for CIFAR-100. We did not run temperature experiments for CIFAR-100, and another choice may improve results. To craft adversarial examples during training, we use FGSM-based PGD with 10 steps, $\ell_\infty$ attack radius of $\epsilon = \frac{8}{255}$, a step size of $\frac{2}{255}$, and a random start [15]. For our experiments, we employ a heterogeneous mix of local machines and cluster nodes with NVIDIA P5000, P100, and GV100 GPUs, so we use hardware-agnostic measures of efficiency and training time.