Transferring Adversarial Robustness Through Robust Representation Matching

Pratik Vaishnavi, *Stony Brook University;* Kevin Eykholt, *IBM;*
Amir Rahmati, *Stony Brook University*

https://www.usenix.org/conference/usenixsecurity22/presentation/vaishnavi

This paper is included in the Proceedings of the
31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Proceedings of the
31st USENIX Security Symposium is
sponsored by USENIX.

Transferring Adversarial Robustness Through Robust Representation Matching

Pratik Vaishnavi
Stony Brook University
pvaishnavi@cs.stonybrook.edu

Kevin Eykholt
IBM
kheykholt@ibm.com

Amir Rahmati
Stony Brook University
amir@cs.stonybrook.edu

Abstract

With the widespread use of machine learning, concerns over its security and reliability have become prevalent. As such, many have developed defenses to harden neural networks against adversarial examples, imperceptibly perturbed inputs that are reliably misclassified. Adversarial training in which adversarial examples are generated and used during training is one of the few known defenses able to reliably withstand such attacks against neural networks. However, adversarial training imposes a significant training overhead and scales poorly with model complexity and input dimension. In this paper, we propose *Robust Representation Matching (RRM)*, a low-cost method to transfer the robustness of an adversarially trained model to a new model being trained for the same task irrespective of architectural differences. Inspired by student-teacher learning, our method introduces a novel training loss that encourages the student to learn the teacher's robust representations. Compared to prior works, RRM is superior with respect to both model performance and adversarial training time. On CIFAR-10, RRM trains a robust model $\sim 1.8\times$ faster than the state-of-the-art. Furthermore, RRM remains effective on higher-dimensional datasets. On Restricted-ImageNet, RRM trains a ResNet50 model $\sim 18\times$ faster than standard adversarial training.

1 Introduction

Despite state-of-the-art performance in numerous domains, deep neural networks (DNNs) remain vulnerable to adversarial examples, inputs that are imperceptibly modified such that they are misclassified by DNNs [23]. In response to the discovery of adversarial examples, several techniques have been proposed to improve the robustness of DNNs against such inputs [15, 19, 27]. Adversarial training is one such technique that augments the training data with adversarial examples. During training, adversarial examples are generated on the fly and used to tune the network weights. Although adversarial training is simple to implement and secure against a wide array of attacks [1, 15], it slows down the training process significantly and scales poorly with respect to model complexity and input dimension. In our experiments, for example, adversarial training is on average $\sim 7\times$ slower than natural training. The expensive computational cost of adversarial training is only exacerbated by improvements to model architecture or new data. When new state-of-the-art model architectures are developed, adversarial training must be redone in order to obtain adversarially robust models. These events make adversarial training impractical to use in real world settings, where models are frequently tweaked to improve performance. Therefore, it is desirable to be able to transfer adversarial robustness between models of different architectures to reduce the cost associated with adversarial training.

Ilyas *et al.* [14] demonstrated that adversarial examples are the result of a model's reliance on non-robust features, *i.e.,* highly predictive features that are incomprehensible to humans, whose correlation with the predicted label can be easily flipped with a small amount of noise. They argue that adversarial training works by forcing the model to assign higher priority to the robust features for classification. Thus, if the non-robust information can be removed from the dataset, adversarially robust models should be obtainable through standard training. To this end, they design a *robust dataset* generation process, which first adversarially trains a model and then uses the learned features in the model to transform the original dataset. Their results demonstrated that new models naturally trained on the robust dataset were more adversarially robust than the models naturally trained on the original dataset.

While the work by Ilyas *et al.* [14] is a step towards transferable adversarial robustness, it suffers from two significant limitations. First, compared to an adversarially trained model, the adversarial robustness of a model trained on the robust dataset is poor. When evaluated against ℓ_2-bounded adversary with $\varepsilon = 0.5$, the adversarial constraint, a ResNet50 model trained on the robust dataset achieves 21.8% adversarial accuracy. This is a significant improvement over the same model

trained on original dataset, which exhibits 0% adversarial accuracy. However, the performance falls significantly when evaluated against $\varepsilon = 1.0$, the value used for adversarial training. In this case, the ResNet50 model trained on robust data achieves only 2.3% adversarial accuracy. Second, generating a robust dataset from an existing adversarially trained model is slow. In our experiments, it took approximately 6 hours with a Titan V GPU to generate a robust CIFAR-10 dataset using an adversarially trained ResNet18 model and the default hyperparameters provided by Ilyas *et al.* [14].

In this paper, we propose Robust Representation Matching (RRM), a novel, low-cost method to transfer adversarial robustness using a student-teacher framework. Similar to prior works, RRM first adversarially trains a model. Then, using the adversarially trained model as a teacher, RRM trains a new student model by modifying the training loss to include a novel **robust representation loss** term. This new term encourages the student model to learn the teacher's robust features. In essence, RRM transfers the teacher's robust features directly to the student as part of standard training rather than expecting the student to learn robust features from the data. RRM can transfer robustness even between models of different architectures. Our proposed method outperforms other adversarial robustness transfer methods including the method demonstrated by Goldblum *et al.* [10], which employs a more traditional distillation approach [12] to transfer adversarial robustness from an adversarially trained teacher to a student using standard training.

Using RRM's student-teacher paradigm significantly speeds up the process of training adversarially robust models. To demonstrate this, we compare it against other approaches that speed up adversarial training [22, 28] using CIFAR-10. Given a pre-trained teacher, RRM is able to achieve performance comparable to adversarial training [15] in the least amount of training time. Additionally, we demonstrate that RRM can benefit from techniques that speed up adversarial training such as Fast Adversarial Training [28]. When combined with Fast Adversarial training, we show RRM achieves higher performance ($\sim$ **2.5**% higher standard accuracy and $\sim$ **3.5**% higher adversarial accuracy) while requiring significantly lower total training time (converges $\sim$ **1.8**$\times$ faster) compared to Free Adversarial Training [22]. Furthermore, we show that RRM is able to scale to higher dimensional datasets using the Restricted-ImageNet dataset. In the presence of an adversarially trained AlexNet model, we are able to train a ResNet50 model $\sim 18\times$ faster than adversarial training while achieving competitive performance on both natural and adversarial images.

Our contributions.

- We introduce Robust Representation Matching (RRM), a technique that allows for transfer of adversarial robustness between two models of varying architecture.
- We evaluate RRM on the CIFAR-10 and Restricted-ImageNet datasets. On CIFAR-10, RRM is able to achieve performance comparable to adversarial training in least amount of training time compared to prior works.
- We show that RRM can scale to higher dimensional datasets such as Restricted-ImageNet. Compared to adversarial training, RRM trains a robust model $\sim 18\times$ faster with a modest reduction in performance (~ 6% on natural images and ~ 12% on adversarial images).

2 Background

In this section, we formally define the problem of adversarial robustness and briefly discuss the concepts foundational to our solution's design.

2.1 Preliminaries

In this paper, we focus on the DNN-based image classification models. The process for training a C-class image classifier F parameterized by θ, *i.e.,* $F_\theta : \mathbb{R}^d \to \{1 \cdots C\}$, involves updating θ so as to minimize the empirical risk over image $x \in \mathbb{R}^d$ and label $y \in \{1, \cdots, C\}$ pairs sampled from an underlying data distribution $\mathcal{D}$. This process, referred to as Empirical Risk Minimization (ERM), can be formalized as follows:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathcal{L}(F_\theta(x), y)] \tag{1}$$

Here $\mathcal{L}$ is a loss function suitable for the task at hand. For image classification, the cross-entropy loss function is typically used for this purpose. In adversarial machine learning literature, training using the ERM objective is popularly referred to as **natural** or **standard training**.

2.2 Adversarial Evasion Attacks

Recent literature has exposed several previously unknown vulnerabilities associated with DNN-based image classifiers [4]. One such class of vulnerabilities, called *adversarial evasion attacks*, tries to compute *imperceptible* perturbations to the input such that the perturbed input is misclassified by a classifier [11, 23]. Since their discovery, several attacks have been proposed. The most powerful class of attacks uses the first-order gradients of the classifier to compute the necessary perturbations to cause misclassification. The optimization objective for adversarially perturbing a given image x, *i.e.,* adversary's objective, can be formalized as follows:

$$\max_{\delta:\, d(x+\delta,x)\leq\varepsilon} \mathcal{L}(F_\theta(x+\delta), y) \tag{2}$$

Here, δ represents the adversarial perturbation. A distance function d and a scalar ε are used to define the set of all permissible adversarial perturbations (or adversary's budget).

With respect to images, this is also referred to as the *imperceptibility condition* and it used to ensure that the adversary perturbs the image imperceptibly in order to launch a stealthy attack. The imperceptibility condition for images is often defined using ℓ_p-norm: $||\delta||_p \leq \varepsilon$. Note that most gradient-based attacks assume white-box access to the classifier, *i.e.,* the same access as the entity that trained the classifier. Some notable gradient-based attacks are JSMA [18], PGD [15], and CW [3].

2.3 Defending against Evasion Attacks

To mitigate the security risks associated with evasion attacks, several defenses have been proposed in recent literature [6,15,19,27,29]. Papernot *et al.* [19] used knowledge distillation [12] to train image classifiers that are robust against evasion attacks in a process they called defensive distillation. It was later shown that defensive distillation was effective only because the proposed method caused the gradients to vanish, making it difficult for the adversary to find a solution for Equation 2. Through proper scaling of the classifier's outputs, Carlini and Wagner [2] were able to resolve the issue of vanishing gradients caused by defensive distillation, allowing existing attacks to converge to a solution successfully.

Several subsequent defenses faced similar issues as defensive distillation as all of these works relied on some form of *gradient obfuscation*. Gradient obfuscation prevents an attacker from using the gradient in order to find a solution for Equation 2. However, this approach only serves to protect a model against naive attackers. Adaptive attackers aware of gradient obfuscation can use alternative methods to approximate the gradients and circumvent the defense. In their paper, Athalye *et al.* [1] proposed a general optimization strategy for breaking gradient obfuscation defenses and demonstrated its effectiveness on several published defenses.

One popular defense that has stood the test of time is adversarial training, first proposed by Madry *et al.* [15], which modifies the training process to create adversarially robust models. Recently, similar training modification defense strategies have been proposed, but with a focus on establishing mathematically provable guarantees [6,27,29] of performance in adversarial environments.

2.3.1 Adversarial Training

The traditional ERM objective (see Equation 1) only optimizes for performance in the standard scenario where the data distribution during testing closely resembles the training distribution $\mathcal{D}$. Therefore, an image classifier trained using ERM can not be expected to perform well on adversarial inputs as these inputs deviate significantly from the distribution $\mathcal{D}$. Madry *et al.* [15] recognize this drawback of ERM and propose modifications to it that enables training of adversarially robust image classifiers. Instead of minimizing the risk over examples drawn from $\mathcal{D}$, they minimize the risk over the *adversarially perturbed* version of these examples. In essence, they augment the training data with adversarial examples. This process is called **adversarial training**, and it can be formalized using the following min-max objective:

$$\min_{\theta} \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[\max_{\delta:\, d(x+\delta,x)\leq\varepsilon} \mathcal{L}(F_\theta(x+\delta),y)\Big] \quad (3)$$

Note that the inner maximization is the adversary's objective from Equation 2 and the outer minimization aims to make it harder for the adversary to achieve its objective. This can be viewed as the formalization of the defender's objective. In their work, Madry *et al.* used the Projected Gradient Descent (PGD) attack, an iterative form of the FGSM attack [11], to find an approximate solution for the inner maximization objective. Their results showed that adversarial training creates MNIST and CIFAR-10 classifiers with high adversarial robustness compared to standard training.

One major drawback of adversarial training is that it has a high computational cost. Standard training involves one forward and one backward pass through the classifier at every training iteration. Adversarial training requires several forward and backward passes per training iteration. For example, the model trained by Madry *et al.* [15] on CIFAR-10 requires 8 forward and backward passes in total (7 for the inner maximization and 1 for the outer minimization). This slows down the training process significantly. Shafahi *et al.* [22] report that adversarially training a model on CIFAR-10 (similar to Madry *et al.*) is 7× slower than standard training.

2.4 Transferring Adversarial Robustness

Ilyas *et al.* [14] discuss that adversarial examples are the result of *non-robust features* that are born out of statistical patterns in the underlying data distribution. These features, although weakly correlated with the correct output, result in high predictive performance on non-adversarial images. Therefore, they can act as potential attack vectors for adversarial evasion attacks as this weak correlation can be easily manipulated using small amounts of perturbations in the image. *Robust features*, however, have a strong correlation with the correct output and therefore are harder to manipulate under the given adversarial budget. Therefore, if one can encourage a model to learn robust features instead of non-robust ones, meaningful adversarial robustness can be achieved using standard training. To achieve this, Ilyas *et al.* propose a method for removing non-robust features from images in a dataset. They begin with the assumption that models trained using adversarial training [15] learn robust features. They then propose generating a *robustified* version x_r of any given image x using the following optimization:

$$\min_{x_r} \quad ||g(x_r) - g(x)||_2 \quad (4)$$

where $g(\cdot)$ returns the penultimate layer outputs of an adversarially trained model. This optimization is solved using gradient descent with x_r initialized using an image randomly sampled from the training data, independently of label of x. This ensures that x_r has minimum amount of non-robust features correlated with the label of x in expectation. Training a classifier using this robustified training data results in the model exhibiting significantly higher adversarial robustness, with a small loss in standard accuracy, as compared to a classifier trained on the original training data.

The work of Ilyas *et al.* [14] is pivotal as it shows that non-trivial adversarial robustness can be achieved using the standard training framework. What they do, in essence, is *transfer* an adversarially trained model's knowledge of robust features to another dataset. However, their robustification process still produces images with some amount of non-robust features as evidenced by empirical results. Models trained on the robustified data only exhibit meaningful adversarial robustness for small values of ε. When evaluated against higher values of ε, the model's performance on adversarial examples becomes worse than random chance. This phenomenon is further discussed in Section 6.3.

3 Why Transfer Adversarial Robustness?

The rise of deep learning has been accompanied by its widespread adoption in commercial systems. Autonomous driving systems, news aggregators, virtual assistants, voice recognition, and fraud detection systems are just some of the systems we rely on every day that use deep learning models to accomplish their tasks. Even in their current imperfect state [16], deep learning models are experiencing rapid improvements, and much of their potential is believed to be unrealized [24]. Like other innovations in computer systems, the increased adoption of deep learning puts a spotlight on their reliability and robustness against adversaries whose goal is to cause the system to misbehave. Early examples of these concerns have manifested as physically realizable adversarial attacks on deep learning models [9]. For safety-critical commercial systems using deep learning models (*e.g.,* self-driving cars), robustness against such attacks is highly desirable. We argue that to be usable in real world systems, the process of robustification of deep learning models should have the following characteristics:

- ***Low standard accuracy reduction.*** Tsipras *et al.* [26] established that there is a trade-off between a model's performance on adversarial and non-adversarial (*i.e.,* natural) inputs. In real world scenarios, adversarial attacks are expected to be rare anomalous occurrences, much like the instances of fraud in credit card transactions. Therefore, commercial systems with strict requirements for standard accuracy (*i.e.,* accuracy on natural inputs) will be hesitant to adopt solutions that significantly impact their functionality in the absence of an adversary. Taking this into consideration, a robustification process should minimally reduce standard accuracy while conferring non-trivial adversarial accuracy.

- ***Low amortized training cost.*** In production, the lifecycle of a deep learning model involves regular retraining and fine-tuning because of the availability of new data and improvements in the models. For example, Figure 1 presents the rate at which the state-of-the-art accuracy on ImageNet dataset has improved in the last decade. The 40% gain in the accuracy during this period results from hundreds, if not thousands, of iterative improvements and modifications and our increased capacity to train deeper models. Adoption to a rapidly evolving environment mandates any robustification technique to impose minimal average overhead in terms of training cost and scale well with the complexity of the newer generation of models.

As one of the most recognized defense approaches, adversarial training, in its current formulation, is poorly suited for real world use. While the models trained using adversarial training show significant adversarial robustness [1], the training overhead remains an issue. Due to the need to generate adversarial examples on the fly during training, multiple forward and backward passes are made, making adversarial training computationally expensive. Furthermore, the training overhead only gets worse as the model and data increase in complexity and dimensionality. In our experiments, adversarially training a classifier (using 7-step PGD attack) was on average 7$\times$ slower than standard training. Any modifications in the model requires a complete repeat of this expensive process as reuse of the existing adversarially trained model is not possible.

RRM seeks to improve the usability of adversarial training by allowing adversarial robustness to transfer across models, thereby eliminating the cost of repeated adversarial training. Through enabling transferable adversarial robustness, our proposed method allows for the reuse of older adversarially trained models to train new adversarially trained models using a modified ERM objective. Furthermore, we can exploit our approach to speed up standard adversarial training at the cost of a small amount of adversarial accuracy by first adversarially training a smaller, faster model and then transferring the robustness to a larger, slower model.

4 Robust Representation Matching

Our objective is to make the process of training adversarially robust models computationally efficient. To this end, we propose *Robust Representation Matching* (RRM), a student-teacher framework to transfer adversarial robustness between models. RRM allows us to train adversarially robust models

[1] The robustness of adversarial training is empirically validated against first-order adversaries.

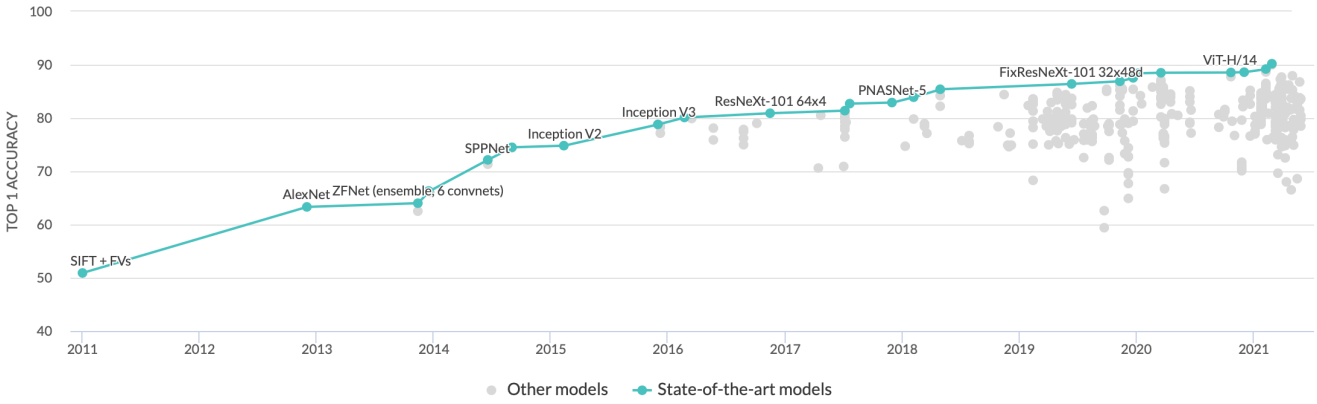

Figure 1: State-of-the-art performance on ImageNet, a popular visual recognition benchmark, over the last decade [20].

at a computational cost comparable to standard training. Empirically, we show that our method helps models attain high standard and adversarial accuracy in the smallest training time as compared to prior works [10, 22].

Our design begins with the same assumption as Ilyas *et al.* [14], *i.e.,* advesarially trained models learn robust features. Thus, given an adversarially trained teacher model T_ϕ, we train a student model S_θ to match the teacher's penultimate layer representations on natural (*i.e.,* non-adversarial) images. This is done using the following training objective:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\lambda \cdot \mathcal{L}_{CE}(S_\theta(x), y) + \mathcal{L}_R(g_S(x), g_T(x))] \quad (5)$$

The functions $g_S(\cdot)$ and $g_T(\cdot)$ return the penultimate layer representations of classifiers S_θ and T_ϕ respectively. Our loss function includes two terms: (1) the cross-entropy loss $\mathcal{L}_{CE}$; and (2) the *robust representation loss* $\mathcal{L}_R$, which computes the distance between the penultimate layer representations of S_θ and T_ϕ. Updating θ to minimize $\mathcal{L}_R$, while keeping ϕ frozen, forces S_θ to learn penultimate layer representations that resemble those of adversarially robust T_ϕ. Matching the penultimate layer representations in this way allows us to transfer adversarial robustness from T_ϕ to S_θ. Algorithm 1 provides the pseudo-code of the RRM training method.

The coefficient λ is used to appropriately weigh the contribution of $\mathcal{L}_{CE}$ towards the total loss. The higher the value of λ, the more S_θ biases towards maximizing standard accuracy with a smaller focus on the teacher's robust representations, which in turn lowers the student's adversarial robustness. This observation is consistent with the findings of prior works [14, 26]. Thus, if we select a small value for λ, we can instead bias S_θ to focus more on robust representations and improve its adversarial robustness through the supervision provided by $\mathcal{L}_R$. The value of λ can be tuned based on the user's requirement[2].

[2]Note that it is mandatory to set λ to a non-zero value. Otherwise, no loss gradient will be present to train the final layer of S_θ as $\mathcal{L}_R$ does not depend on the final layer.

As we are training S_θ with natural images only, the computational cost of our approach is comparable to that of standard training and much lower than adversarial training [15]. In addition to the backpropagation step present in standard training, RRM requires an additional forward pass through classifier T_ϕ to compute $\mathcal{L}_R$, which we later show is only a small amount of additional overhead. Although adversarially training T_ϕ is necessary, its cost is amortized as we can robustly train future student models, as might be required in large-scale commercial systems.

Algorithm 1: Robust Representation Matching (RRM)

Input: Training data distribution $\mathcal{D}$, learning rate η, training iterations $\mathcal{T}$
Output: Robust student classifier S_θ

$T_\phi \leftarrow$ AdversarialTraining($\mathcal{D}$) ;
$S_\theta \leftarrow$ random initialization ;
$g_T \leftarrow T_\phi$'s mapping from input to penultimate layer ;
$g_S \leftarrow S_\theta$'s mapping from input to penultimate layer ;
$i \leftarrow 1$;
while $i < \mathcal{T}$ **do**
  Sample input batch $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ from $\mathcal{D}$;
  $l_r \leftarrow \frac{1}{n}\sum_{j=1}^{n} \mathcal{L}_R(g_S(x_j), g_T(x_j))$;
  $l_{ce} \leftarrow \frac{1}{n}\sum_{j=1}^{n} \mathcal{L}_{CE}(S_\theta(x_j), y_j)$;
  $l_{total} \leftarrow \lambda \cdot l_{ce} + l_r$;
  $\theta \leftarrow \theta - \eta \cdot \nabla_\theta l_{total}$;
  $i \leftarrow i + 1$;
end while

Why match penultimate layer representations?

Goldblum *et al.* [10] demonstrated that it is possible to transfer adversarial robustness from an adversarially trained teacher to a student using the traditional Knowledge Distillation (KD)

loss [12]. The training objective of traditional KD is defined as follows:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}[(1-\alpha)\mathcal{L}_{CE}(S^t_\theta(x),y) + \alpha t^2 \mathcal{L}_{KL}(S^t_\theta(x), T^t_\phi(x))] \tag{6}$$

Here, $\mathcal{L}_{KL}$ is the KL divergence loss and is applied between the temperature (or t) scaled softmax outputs of the student and the teacher. The hyperparameter α is used to control the contribution of $\mathcal{L}_{KL}$ and the standard cross-entropy loss $\mathcal{L}_{CE}$ towards the total loss. This training objective is similar to the training objective of RRM (Equation 5) with the key difference being the layer whose outputs are being matched. While KD traditionally matches the final layer output of the student and the teacher, RRM instead matches the learned representations (*i.e.,* the penultimate layer output). We recognize that, as a result of minimizing the cross-entropy loss during training, a highly accurate teacher's outputs will closely resemble the ground truth labels. Therefore, matching the final layer outputs may limit the information transferred to the student as opposed to using the learned representations. In Section 6.3, we compare the KL divergence loss used on final layer outputs with our proposed robust representation loss and demonstrate the improved performance when the learned representations are used to transfer knowledge. We will further discuss the differences between RRM and the KD method used by Goldblum *et al.* in Section 8.

Matching the penultimate layer representations also allows us to preserve our **model agnostic** design. While the intermediate representations from an arbitrary layer may be higher in dimensionality, the representation matching approach used by RRM requires that the student and teacher's representations are the same dimensionality. Different model architectures can vary highly with respect to the dimensions of their intermediate layers, and thus, would require invasive changes at the intermediate layer for representation matching to be used. Furthermore, these changes would need to be cascaded downstream. In choosing the penultimate layer, we can ensure that RRM is model agnostic as most popular architectures (VGG, ResNet, *etc.*) have identical penultimate layer dimensions by default. In cases where the dimensions differ, we can simply add a single layer after the penultimate layer to one of the models to ensure proper sizing without the need for downstream changes. Our experiment results presented in Section 6 demonstrate the effectiveness of this approach.

5 Threat Model

In this section, we provide specifications of the threat model under which we perform our evaluations. Our threat model is similar to the one used by Madry *et al.* [15]. Additionally, we claim similar adversarial robustness guarantees as them.

Adversary Goals. In this work, we focus on evasion attacks on image classifiers. The adversary's goal is to imperceptibly perturb a given image such that the resulting image is misclassified by the image classifier. Evasion attacks are of two types: *targeted* and *untargeted*. Targeted evasion attacks seek to perturb images so that the classifier outputs a specific label that is desirable for the adversary. In untargeted evasion attacks, all incorrect labels are of similar value to the adversary. The objective of an untargeted attack is formalized in Equation 2. In this paper, we only evaluate against untargeted evasion attacks. However, based on the work by Madry *et al.* [15], we can safely assume robustness against targeted evasion attacks as well.

Adversarial Capabilities. The adversary is allowed to imperceptibly perturb the input to an image classifier. Similar to related works, we define the imperceptibility condition using ℓ_p-norm, *i.e.,* $||\delta||_p \leq \varepsilon$. The adversary uses the first-order gradients of the classifier to solve Equation 2, as the majority of optimization problems in machine learning are solved using first-order methods.

Adversary Knowledge. We evaluate under the white-box threat model and assume that the adversary has complete knowledge of the classifier and its parameters. Additionally, the adversary is aware of the defense algorithm and can adapt to it. The adversary also has access to the training data used to train the target classifier.

6 Evaluation

We conduct experiments to demonstrate the superiority of RRM along two dimensions: (1) training time (Section 6.2), and (2) effectiveness of transfer (Section 6.3). We perform both set of experiments using the CIFAR-10 dataset and compare against most relevant recent prior works. We also demonstrate that RRM scales to high dimensional datasets using the Restricted-ImageNet dataset [26] (Section 6.4). Through our evaluation, we verify that the results presented are statistically significant. For adversarial accuracy computation, we follow the standard practice in adversarial machine learning literature and use multiple random restarts to ensure that the attack doesn't get stuck in bad local maxima. For epoch timings, we compute the 95% confidence interval to study statistical significance of our speedup results.

6.1 Experimental Setup

All of our experiments were performed using the PyTorch library [21]. Mixed-precision training was performed using the Nvidia Apex library [17]. We follow prior works for choosing the hyperparameters used in our experiments (details in Appendix A). We train a ResNet50 model using different robustification approaches and compare the adversarial robustness of the resulting models to measure the relative effectiveness of these approaches. We measure adversarial robustness at test time using the AutoPGD attack [7]. The attack uses the cross-entropy loss and is run for 50 iterations

Table 1: Comparing the performance and training time of a robust ResNet50 trained with different approaches. The teachers used for RRM models are noted in the parentheses. The adversarial accuracy evaluation is done using an ℓ_∞-bound AutoPGD attack [7] with $\varepsilon = 8/255$, 50 iterations and 10 random restarts. Compared to SAT, RRM achieves significant speedup while maintaining comparable adversarial accuracy and suffering minor drop in natural accuracy. Compared to Free AT, RRM achieves better natural and adversarial accuracy while converging $\sim 1.8\times$ faster. For epoch time, we report the 95% confidence interval to demonstrate statistical significance.

Method	Epochs	Epoch Time (sec)	Total Time (min)	Natural	AutoPGD
SAT	150	723.03 ± 0.88	1807.58	85.50%	48.38%
Fast AT	40	289.16 ± 0.22	192.77	83.73%	50.47%
Free AT	96	36.58 ± 0.23	58.44	77.74%	45.20%
Free AT	48	36.58 ± 0.03	29.22	71.28%	41.53%
RRM (VGG11)	48	37.78 ± 0.09	30.22	76.17%	49.30%
RRM (ResNet18)	48	39.78 ± 0.10	31.82	80.32%	48.67%

with 10 random restarts (adopted from Wong *et al.* [28]). We use the IBM Adversarial Robustness Toolbox (ART) [13] to perform the attack. For RRM models, we use the **cosine similarity** metric to compute the robust representation loss $\mathcal{L}_R$ in Equation 5. The code supporting our experiments is available at https://github.com/Ethos-lab/robust-representation-matching. Our code is based on the implementation provided by Wong *et al.* [28] [3] and MadryLab's robustness package [8] [4].

Hardware. We ran our experiments on two different machines. The CIFAR-10 experiments were run on a machine with an Intel Xenon(R) Gold 6136 CPU, 16 GB RAM, and an Nvidia Titan V GPU. The Restricted-ImageNet experiments were run on a second machine with an Intel Xenon(R) E5-2690 CPU, 16GB RAM, and an Nvidia V100 GPU. Due to GPU memory limitations on the second machine, the VGG16 experiments were run on the first machine across 2 GPUs - Nvidia Titan V and GeForce RTX 2080 Ti.

6.2 Adversarial Training Speedup

In this section, we demonstrate how RRM can be used to speed up adversarial training and compare it against two adversarial training approaches: (1) Standard Adversarial Training (SAT), proposed by Madry *et al.* [15] and (2) Free Adversarial Training, recently proposed by Shafahi *et al.* [22] to speed up SAT. We apply the DAWNBench improvements[5] proposed by Wong *et al.* [28] to both RRM and Free AT during the experiments and also show its effects on standard adversarial training (Fast AT). Overall we demonstrate that, given a pre-trained teacher, **RRM achieves adversarial robustness comparable to SAT in the least amount of training time**.

[3] https://github.com/locuslab/fast_adversarial

[4] https://github.com/MadryLab/robustness

[5] Mixed-precision training with cyclic learning rate scheduling.

We compare the performance and the time required to train an adversarially robust ResNet50 model with each approach. Using the attack budget from prior work, we conduct the experiments using an ℓ_∞-bound adversary with $\varepsilon = 8/255$. For RRM, we use $\lambda = 5e-3$ and provide results using VGG11 and ResNet18 as teachers to demonstrate RRM's capability in transferring robustness across different class of model architectures. Additionally, to demonstrate that RRM is model-agnostic, we purposely use student-teacher pairs with different penultimate layer dimensions. Following the strategy discussed in Section 4, we remedy the dimensional mismatch by adding a single fully connected layer to the model with higher penultimate layer dimension. For details regarding classifier modifications we make, see Appendix B. The teachers are trained using Fast AT to reduce the teacher training overhead. With respect to the number of training epochs, we report the performance of SAT and RRM at convergence and report the performance of Fast AT and Free AT with their default parameters. We also include the performance of Free AT at the same number of epoch as RRM convergence (48 epochs) for side-by-side comparison. The summary of results is presented in Table 1.

6.2.1 Standard Adversarial Training (SAT)

As we see in Table 1, RRM attains adversarial robustness comparable to SAT and Fast AT but in a fraction of the time. Specifically, RRM achieves an average speedup of $\sim 58\times$ over SAT and of $\sim 6\times$ over Fast AT. We note that the performance of RRM models on natural images is reduced by $\sim 8\%$ when using a VGG11 teacher and $\sim 4\%$ when using a ResNet18 teacher. In Section 7, we discuss the balance between natural and adversarial accuracy based on the value of λ used during training.

6.2.2 Free Adversarial Training (Free AT)

Free AT speeds up SAT by requiring only a single forward and backward pass during each training iteration [22]. A single backward pass is used to compute the gradients of the loss with respect to the model's parameters (to train the model) and the input image (to compute the adversarial perturbation). The PGD attack used in SAT requires several backward passes to compute the adversarial perturbation. Free AT mimics this by repeating the same batch m times and using the adversarial perturbation computed in one iteration to initialize the adversarial perturbation of the next iteration. After m replays, a new batch is used and the adversarial perturbation is reset.

For brevity, we only compare against the version of Free AT [22] with DAWNBench improvements applied as it is comparable to Free AT in terms of adversarial robustness, but superior in terms of total training time. We use $m = 8$ as proposed by the authors. Compared to RRM, Free AT is slightly faster with respect to per epoch time as shown in Table 1. However, **RRM achieves better performance** ($\sim$ 2.5% higher standard accuracy and $\sim$ 3.5% higher adversarial accuracy) and **has significantly lower total training time** as it converges $\sim 1.8\times$ faster than Free AT.

In Figure 2, we plot the performance of a ResNet50 model trained using Free AT and RRM for different number of epochs. The x-axis represents the total number of epochs the models were trained for and the y-axis represents the corresponding accuracy. Solid lines represent accuracy on natural test set and dashed lines represent accuracy on adversarial test set generated using AutoPGD attack with 50 steps and 10 random restarts. Each data-point in the curve is the average model performance across 3 independent training runs. As can be seen, RRM converges faster and has better performance across the entire range of the x-axis.

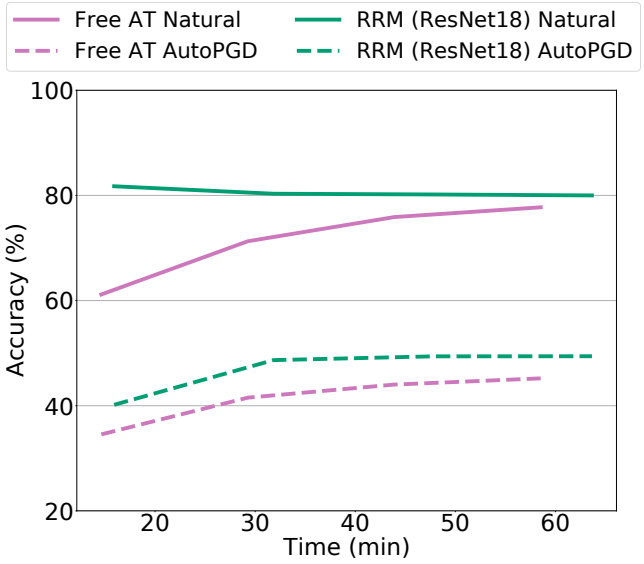

Figure 2: Plotting the performance of a ResNet50 model trained using Free AT and RRM for different amount of training time budget. The x-axis represents the total time (in minutes) the model was trained for and the y-axis represents the accuracy of the trained model. Each data-point in the curve is the average model performance across 3 independent training runs. RRM outperforms Free AT while converging faster.

6.2.3 Teacher Overhead in RRM

The results reported in Table 1 only present the time required to train the ResNet50 model. In case of RRM, we do not include the time required to train the teacher. We argue that the cost to train the teacher can largely be amortized as one teacher training session can be leveraged to train an arbitrary number of students. In Figure 3, however, we compare the training time of RRM with other methods when the teacher's training overhead is included (for numerical results see Appendix D). In this setting, RRM is still on average 17.5$\times$ and 2.7$\times$ faster than SAT and Fast AT, respectively. When compared to Free AT, RRM requires comparable total training time to train a model with better performance.

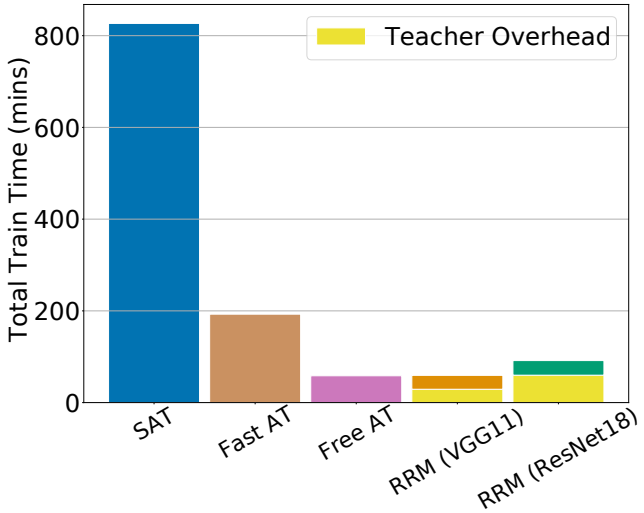

Figure 3: Comparing total training times of SAT, Fast AT, and Free AT with RRM. Yellow regions represent the total time of adversarially training a teacher. If an adversarially robust teacher is already trained, the total training time of RRM is decreased significantly.

6.3 Adversarial Robustness Transfer

In this section, we compare against two prior works that propose techniques to transfer adversarial robustness between models. First, we examine the robust data training approach [14], which creates a robust dataset learned from an adversarially trained model to transfer adversarial robustness. Robustness is transferred to other models through standard

Table 2: Comparing RRM against RDT [14] and KD [10] using ResNet50. Performance of model trained using SAT is provided for reference. The adversary is ℓ_2-bound with $\varepsilon = 1.0$. The evaluation is done using AutoPGD attack [7] with 50 iterations and 10 random restarts. Models trained using RRM exhibit performance comparable to SAT and significantly better than model's trained using RDT and KD.

Method	Teacher	Natural	AutoPGD
SAT	-	82.97	48.49
RDT	VGG11	74.61	1.10
	ResNet18	80.47	1.22
KD	VGG11	80.12	20.89
	ResNet18	83.72	2.83
RRM	VGG11	78.53	47.24
	ResNet18	80.80	46.18

training on the robust dataset. Second, we reproduce the experiment conducted by Goldblum *et al.* [10] that suggested an adversarially trained teacher's robustness can be transferred to a student using knowledge distillation. Given the pre-trained teacher, they included a new loss term, a KL divergence loss between the temperature scaled softmax outputs of the teacher and student models, during standard training. Our empirical results demonstrate that **RRM is superior to both these approaches with respect to the effectiveness of the transfer**.

We compare the performance of an adversarially robust ResNet50 model. To remain consistent with the work by Ilyas *et al.* [14] and lessen hyperparameter tuning, we use an ℓ_2-bound adversary with $\varepsilon = 1.0$. For RRM, we use $\lambda = 5e-5$. The results are summarized in Table 2.

6.3.1 Robust Data Training (RDT)

Ilyas *et al.* [14] propose **dataset robustification** in order to transfer robustness between models. Their method removes the *non-robust features* from the training dataset through an optimization process (Section 2.4) resulting in a new "robustified" dataset. They demonstrated that the "robustified" training data can be used with standard training to train classifiers with non-trivial adversarial robustness. While both their work and ours use an adversarially robust classifier's penultimate layer to identify robust features, their work adds an additional intermediate dataset robustification step. On the other hand, our work adds a feature loss to the standard training loss to directly encode the robust features into the student.

To generate a robust dataset from an adversarially trained classifier, we follow the steps described by Ilyas *et al.*. First, we use a random image from the training set to initialize optimization. Then, 1000 steps of gradient descent with a step size of 0.1 are performed to minimize the loss described in Equation 4. At each step, the ℓ_2-norm of the gradient is normalized. We used an ℓ_2-bound adversary with $\varepsilon = 1.0$ to adversarially train the classifier used for robustification (*i.e.,* teacher model).

In Table 2, we observe that models trained using RRM have comparable standard accuracy and significantly better adversarial accuracy than models trained using RDT. In the original paper, RDT trained models exhibited non-trivial adversarial robustness with respect to smaller values ε of epsilon, which is what we originally observed as well. When we re-evaluated the models with respect to $\varepsilon = 1.0$ (*i.e.,* the value used to train the teacher), the student models exhibit negligible adversarial robustness. Furthermore, we found that the robust data generation process was computationally expensive. Using the proposed hyper parameters and an adversarially trained ResNet18 model, the robust data generation process took approximately 6 hours to complete. Although this cost would be amortized, the poor performance of the student models suggest that this approach is not feasible for transferring robustness.

6.3.2 Knowledge Distillation (KD)

Prior to our work, Goldblum *et al.* [10] demonstrated that adversarial robustness can be transferred from an adversarially trained teacher to a student using knowledge distillation [12] with naturally trained images. KD seeks to minimize the KL divergence between the temperature scaled softmax outputs of a student and a teacher model in addition to minimizing the student's classification loss (see Equation 6). We reproduce this experiment based on information provided by Goldblum *et al.* in their paper. We train both the student and the teacher with temperature $t = 30$, the proposed value. The standard and adversarial accuracy (against a 20 steps PGD attack) of the adversarially trained teacher models are: (1) 78.3% and 47.67% for VGG11; and (2) 82.38% and 51.15% for ResNet18. The performance of the student models is provided in Table 2. Note that while training the student, we set the value of α to 1 as this corresponds to the **maximum** attainable adversarial robustness using this method.

When we adversarially attacked the student model with $t = 1$ (*i.e.,* the default used during evaluation), we observed a non-trivial adversarial robustness similar to what Goldblum *et al.* originally reported. However, when we set $t = 30$ (*i.e.,* the value used during training), the student's adversarial accuracy drops significantly. This phenomenon is due to vanishing gradients originally observed by Carlini and Wagner [2] when analyzing another distillation based defense [19]. Thus, traditional KD is not feasible for transferring robustness.

Table 3: Comparing the performance and training time of a robust ResNet50 and VGG16 models trained using SAT and RRM. An AlexNet model trained using SAT is used as teacher for RRM. The adversarial accuracy evaluation is done using an ℓ_2-bound AutoPGD attack [7] with $\varepsilon = 3$, 20 iterations, and 5 random restarts.

Method	Epochs	Epoch Time (mins)	Total Time (hrs)*	Natural	AutoPGD
ResNet50					
SAT	150	101.49	253.71	95.47%	84.36%
RRM	60	13.66	13.66	88.19%	67.90%
VGG16					
SAT	150	160.01	400.01	92.40%	80.91%
RRM	60	33.30	33.30	87.86%	73.30%

**The ResNet50 and VGG16 models were trained on different machines due to memory constraints.*

6.4 Scaling RRM to Complex Datasets

To examine how well RRM adapts to more complex datasets, we evaluate RRM using the Restricted-ImageNet dataset, which was introduced by Tsipras *et al.* [26] to facilitate adversarial robustness research with high resolution images. The large number of classes present in the original ImageNet dataset makes it difficult to use SAT and achieve acceptable performance on natural and adversarial images. Restricted-ImageNet is generated by grouping together a subset of semantically similar classes from ImageNet into 9 super-classes. For our experiments we follow Ilyas *et al.* [14] and use an ℓ_2-bound adversary with $\varepsilon = 3.0$. We train a ResNet50 and a VGG16 model using SAT and RRM and compare their performance and training times. For RRM, we use ℓ_2 loss to compute robust representation loss $\mathcal{L}_R$, $\lambda = 1e-3$, and an AlexNet teacher trained using SAT.

In Table 3, we compare the standard and adversarial accuracy of the RRM models against their SAT baselines. The RRM models exhibit competitive natural and adversarial accuracy, coming within a few percentage points of the corresponding SAT model's performance. Specifically, across the two models, there is an average reduction of $\sim 6\%$ in natural accuracy and of $\sim 12\%$ in adversarial accuracy. However, relative to SAT, RRM achieves a speedup of $\sim 18\times$ on ResNet50 and $\sim 12\times$ on VGG16. All models were trained till convergence. When including the teacher's training time, RRM achieves a speedup of $5.4\times$ on ResNet50 and $6.0\times$ on VGG16 (for numerical results see Appendix D). Note that we do not use DAWNBench improvements in this set of experiments. The standard adversarial training performance of all models used in Table 3 is reported in in Appendix C.

7 Discussion

In this section we discuss some noteworthy points regarding RRM. First, we discuss how changing the value of λ affects the performance of the student model, which can inform users of RRM how to tune λ based on their use case. Second, we explore the decreasing effectiveness of RRM when the teacher is more complex than the student. In our presented results, the teacher's architecture was always less complex (*i.e.,* faster to train) than the student. Using a ResNet50 teacher, we train multiple student models of decreasing complexity and observe lowering rates of robustness transfer.

7.1 Tuning the λ Parameter

In Figure 4 we plot the performance of two ResNet50 models trained using RRM, while varying the value of λ used. The two models are trained with a VGG11 and a ResNet18 teacher, respectively. The adversarial accuracy is computed using AutoPGD attack with 20 iterations and 5 random restarts.

Equation 5 contains two loss terms: 1) $\mathcal{L}_{CE}$, which improves the model's natural accuracy and 2) $\mathcal{L}_R$, which encourages the model to learn the teacher's robust representations. As we decrease the value of λ, the contribution of $\mathcal{L}_{CE}$ is reduced, which increases the contribution of $\mathcal{L}_R$. In Figure 4, we observe this effect for $\lambda \geq 1e-2$. For $5e-5 \leq \lambda \leq 1e-2$, we observe somewhat of a plateau in performance for the adversarial accuracy, with only a slight negative slope in natural accuracy. Thus, any value in this range will likely result in robust student model. Finally, if λ becomes too small (*i.e.,* $\lambda < 5e-5$), the training focuses too much on matching the robust representation of the teacher that the student model's performance plummets. The drop in adversarial accuracy at this point is attributed to the model's poor natural accuracy rather than a decrease in robustness.

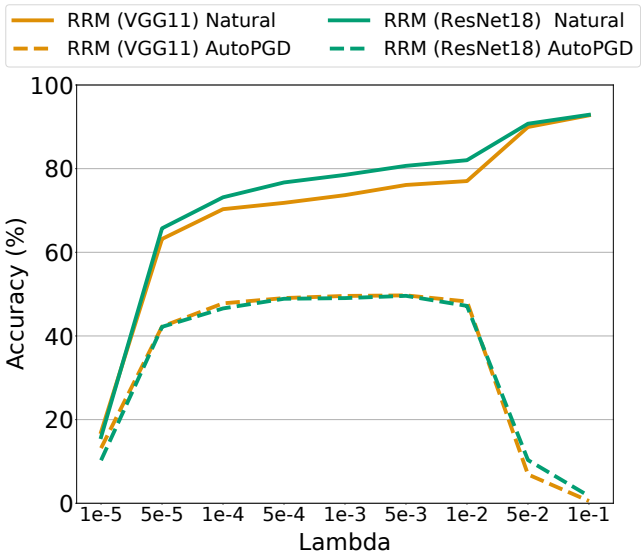

Figure 4: Plotting the performance of ResNet50 trained with RRM using two different teachers while varying the value of λ. We see that there is a plateau in adversarial accuracy for $5e-5 \leq \lambda \leq 1e-2$. λ values outside of this range either result in a model with poor natural accuracy and/or poor adversarial robustness.

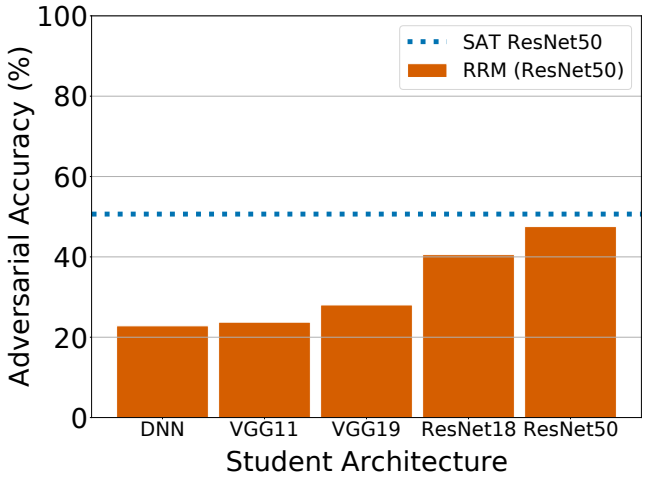

Figure 5: Limit testing RRM on CIFAR-10 by transferring adversarial robustness from a ResNet50 teacher to several students of varying complexities. Adversarial images were generated using an ℓ_2-bound AutoPGD attack [7] with $\varepsilon = 1.0$, iterations $= 20$, and 5 random restarts.

7.2 Limit Testing

Previously, we showed that RRM allows us to efficiently transfer a significant amount of robustness from an adversarially trained teacher to a student despite differences in architecture. However, during our experiments, we noticed a few cases of poor robustness transferability. For example, on CIFAR-10, with a ResNet50 teacher and an ℓ_2-bounded adversary with $\varepsilon = 1$, the VGG11 and VGG19 students were only able to achieve 23.74% and 28.03% adversarial accuracy, respectively. Compare this with the ResNet18 and ResNet50 student models, which achieve 40.56% and 47.54% adversarial accuracy. We observed a similar phenomenon with our Restricted-ImageNet experiments, in which an AlexNet model was only able to achieve 51.72% adversarial accuracy (natural accuracy is 78.11%) when trained with a ResNet50 teacher. In contrast, an AlexNet model trained with SAT achieved 75.24% adversarial accuracy. These observations suggest that, under certain conditions, the effectiveness of RRM is limited. Figure 5 presents our exploration of these limits. We evaluated the adversarial accuracy on CIFAR-10 of several RRM models trained using the ResNet50 teacher and compare them to the adversarial accuracy of the teacher. In addition to the model architectures we already mentioned, we also trained a simple DNN with two convolution layers and two fully connected layers. Using SAT, this DNN achieves 59.55% natural accuracy and 34.15% adversarial accuracy.

If we rank the complexity of a classifier based on its per-epoch training time, then we have the following order: ResNet50 > ResNet18 > VGG19 > VGG11 > DNN. Thus, we rank the ResNet50 (723 seconds per-epoch) as the most complex and the DNN (14.3 seconds per-epoch) as the least complex classifier. Our results suggest that the simpler a student is compared to the teacher, the poorer the student's performance will be. We hypothesize that the per-epoch training time is a rough approximation of the model's expressive power. Thus, the robust features used by a complex teacher are harder for less complex students to learn, resulting in poor transferability in such cases. Further exploration is required to establish a concrete metric to predict the transferability between different classifier architectures when training with RRM. While these results suggest the existence of some limitations with our proposed approach, we note that RRM is still applicable in real world settings where there are relatively small differences in complexity between successive generations of model architectures. Furthermore, in cases where the student and teacher are trained from scratch, it is beneficial to pair a less complex teacher with a more complex student to reduce training overhead. The standard adversarial training performance of all models in Figure 5 are reported in Appendix C.

8 Related Works

Adversarially Robust Distillation (ARD) [10]. In their work, Goldblum *et al.* used traditional knowledge distillation with an adversarially robust teacher to train an adversarially robust student. Here we describe the key differences between the traditional KD method from their paper and RRM. Note that we do not compare against ARD, which is the main contribution of their paper. This is because ARD requires training both the student and the teacher using adversarial training and solely

focuses on improving adversarial robustness of the student without any regards to the total training time.

First, RRM uses a cosine similarity loss focused on the models' penultimate layer (i.e., pre-logit layer) whereas Goldblum *et al.* use a knowledge distillation loss focused on the models' temperature scaled softmax outputs. Our approach encourages the student to utilize the robust representations learned by the teacher. When they have differently shaped representation layers, we add an additional layer of the correct shape after the current penultimate layer to one of the models. In contrast, KD expects the student to learn its own representations to match the teacher's softmax output and requires an additional hyperparameter, the temperature t, as compared to RRM. Depending on its magnitude, t can affect the success rate of gradient based adversarial attacks due to an artificial scaling of the logits [2]. Note that the hyperparameter λ in Equation 5 and α in Equation 6 serve identical purpose of controlling the trade-off between natural and adversarial accuracy.

The second difference is in intent. KD is traditionally used to transfer the performance of a larger more complex model to a smaller less complex model in an effort to "compress" the larger model into the smaller one. Goldblum *et al.* use this to *improve the adversarial robustness* of smaller models by leveraging robust larger models and their experiments reflect this. Their approach augments adversarial training with KD to achieve high performance. As we showed in Section 6.3, distillation alone is insufficient to transfer the robustness of the teacher and, likely, only helps to fine-tune adversarial performance. In contrast, RRM seeks to *reduce the adversarial training overhead* by leveraging smaller models to quickly train robust larger models. We necessarily do not adversarially train the student.

Dataset Robustification [14]. In their work, Ilyas *et al.* designed a dataset transformation approach to create a new dataset composed only of "robust features". With the robust dataset, one could use standard ERM (Equation 1) to train a model with non-trivial adversarial robustness. The key difference between their approach and RRM lies in the technique used to transfer the robust features to the student model. Their work assumes that a model trained on a robustified dataset would automatically learn the robust features. As we showed in Section 6.3, their method only marginally improves the adversarial robustness of the trained models. RRM directly provides the robust feature representations from the adversarially trained teacher to the student, which results in models with much higher adversarial accuracy. Furthermore, our method does not include the additional overhead of generating the robust dataset.

IGAM [5]. Another work that explores transferring adversarial robustness was published by Chan *et al.* [5]. Their approach for transferring adversarial robustness involves matching the student's loss gradients (with respect to the input) to the loss gradients of an adversarially robust teacher. While both their work and ours pertain to transferring adversarial robustness between models, their focus is on transferring adversarial robustness across task domains, which is orthogonal to the problem solved by RRM. They do not conduct experiments to transfer performance within the same task between models of different architectures. Instead, they focus on transferring robustness between models having same architecture, but trained for different tasks (using different datasets).

9 Limitations and Future Work

RRM dependence on adversarial training. RRM requires an adversarially trained teacher model. Thus, any shortcomings of adversarial training, such as the large training overhead and potential overfitting [25] still exist with respect to the teacher model. However, we demonstrated that RRM reduces training time when an adversarially trained teacher is already available and in cases when training the student-teacher pair is significantly faster than adversarially training the student. Furthermore, any improvements to adversarial training will improve RRM indirectly. In Section 6.2, we demonstrated that some of the speedup techniques proposed by Wong *et al.* [28], such as mixed-precision training, are compatible with RRM.

RRM in other domains and model types. In this work, we only studied transferring adversarial robustness between deep neural networks in the image classification domain. It is unknown if RRM would work in other domains or with other types of models (e.g. decision trees or LSTMs). With respect to other domains, if adversarial training exists in the domain, we expect RRM to work as its core idea is to encourage the student to use the robust representations learned by a teacher. Regarding transferring between different model types, further investigation is needed to determine if this is feasible and if not, what modifications are required to make it feasible.

10 Conclusion

Adversarial machine learning looms as an ever-present threat to the security and reliability of machine learning systems as research has proven attackers with a certain level of access can reliably cause them to misbehave. As such, it is desirable to train machine learning models that are robust to adversarial attacks in advance rather than wait for a breach to occur. Unfortunately, adversarial training, one of the most well-known and reliable defenses, is impractical to deploy in real world systems. Like software, machine learning models need to be constantly updated to maintain state-of-the-art performance due to the availability of new training data or the development of new model architectures. The high training overhead and poor scalability of adversarial training discourage users from adopting it as part of their training process.

In this paper, we proposed a method to transfer adversarial robustness between models despite differences in their

architectures. RRM enables low-cost, efficient transfer of robust representations learned by an adversarially trained teacher model to a new student model. By adding a new loss term to the standard training objective, an adversarially robust model can be trained on natural images only using RRM. On CIFAR-10, we demonstrated that RRM outperforms state-of-the-art adversarial training speedup techniques. On Restricted-ImageNet, a higher dimensional dataset, we demonstrated that RRM remains effective both in terms of model performance and training speedup.

Acknowledgement

We thank Veena Krish, Farhan Ahmed, the anonymous reviewers, and our shepherd, David Freeman, for their valuable feedback. This work was supported by the Office of Naval Research under grants N00014-20-1-2858 and N00014-22-1-2001, and Air Force Research Lab under grant FA9550-22-1-0029. Any opinions, findings, or conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] 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 (ICML)*, 2018.

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

[3] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE symposium on security and privacy (S&P)*, 2017.

[4] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. 2018. arXiv:1810.00069.

[5] Alvin Chan, Yi Tay, and Yew-Soon Ong. What it thinks is important is important: Robustness transfers through input gradients. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[6] Jeremy M Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning (ICML)*, 2019.

[7] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning (ICML)*, 2020.

[8] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. https://github.com/MadryLab/robustness.

[9] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *International Conference on Learning Representations (ICLR)*, 2018.

[10] Micah Goldblum, Liam Fowl, Soheil Feizi, and Tom Goldstein. Adversarially robust distillation. In *AAAI Conference on Artificial Intelligence*, 2020.

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2014.

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

[13] IBM. Adversarial Robustness Toolbox (ART). https://github.com/Trusted-AI/adversarial-robustness-toolbox.

[14] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[15] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representation (ICLR)*, 2018.

[16] Gary Marcus. Deep learning: A critical appraisal. 2018. arXiv:1801.00631.

[17] NVIDIA. Apex - a PyTorch extension: Tools for easy mixed precision and distributed training in PyTorch. https://github.com/NVIDIA/apex.

[18] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European symposium on security and privacy (EuroS&P)*, 2016.

[19] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE symposium on security and privacy (S&P)*, 2016.

[20] Papers with Code. Image classification on imagenet. https://paperswithcode.com/sota/image-classification-on-imagenet. Accessed: 2021-06-03.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[22] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[23] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[24] Rob Toews. Deep learning has limits. but its commercial impact has just begun. https://www.forbes.com/sites/robtoews/2020/02/09/deep-learning-has-limits-but-its-commercial-impact-has-just-begun/?sh=5c38edca6e1a. Accessed: 2021-06-02.

[25] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations (ICLR)*, 2018.

[26] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representation (ICLR)*, 2019.

[27] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)*, 2018.

[28] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations (ICLR)*, 2019.

[29] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representation (ICLR)*, 2019.

A Hyperparameters

Here we provide the hyperparameters used to train the models in our experiments. Table 4 provides the hyperparameters used to adversarially train each model. Table 5 provides the hyperparameters used when training a student model using RRM. Table 6 provides the hyperparameters used when training a student model using prior works: Robust Data Training (RDT) [14], Knowledge Distillation (KD) [10], Fast Adversarial Training (Fast AT) [28], and Free Adversarial Training (Free AT) [22].

Table 4: Hyperparameters used to train models using standard adversarial training [15] on different datasets.

Dataset	LR	Batch Size	Epochs	LR Decay
CIFAR-10	0.1	128	150	50,100
Restricted-ImageNet	0.01	128	150	125

Table 5: Hyperparameters used to train models using RRM on different datasets.

Dataset	LR	Batch Size	Epochs	LR Decay
CIFAR-10	0.1	128	48	cosine
Restricted-ImageNet	0.1	128	60	35,50

Table 6: Hyperparameters used to train models using prior works on CIFAR-10.

Method	LR	Batch Size	Epochs	LR Decay
RDT [14]	0.1	128	100	65,90
KD [10]	0.1	128	100	65,90
Fast AT [28]	0.2 (max)	128	40	cyclic
Free AT [22]	0.04 (max)	128	96	cyclic

B Penultimate Layer Dimensions

In order to demonstrate that RRM is model-agnostic, we used a variety of classifiers in our experiments. These classifiers belong to different class of architectures (VGG, ResNet *etc.*) as well as have different penultimate layer dimensions. Since our robust representation loss requires the penultimate layer features of the student and teacher to be of the same dimension, we add an additional fully connected layer after the penultimate layer in certain classifiers. These modifications are summarized in Table 7.

Table 7: Dimensions of the penultimate layer features of the various classifiers we use in our experiments. To be able to use or robust representation loss, we add an additional fully connected layer to some architectures as specified below.

Classifier	Penultimate Layer Dimension	
	Original	After Modification
CIFAR-10		
VGG11	512	N/A
VGG19	512	N/A
ResNet18	512	N/A
ResNet50	2048	512
Restricted-ImageNet		
AlexNet	4096	2048
VGG16	4096	2048
ResNet50	2048	N/A

C Adversarial Training Results

In this section, we include the performance of all the adversarially trained models (SAT [15] and Fast AT [28]) that we used in our experiments. Information regarding the adversary used is provided in parentheses. Refer to Table 8 for these results.

Table 8: Performance of adversarially trained models we used in our experiments. The AutoPGD attack [7] was performed using 20 iterations and 5 random restarts.

Threat Model	Classifier	Method	Natural	AutoPGD
CIFAR-10				
$\ell_2, \varepsilon = 1.0$	DNN	SAT	59.95	34.15
	VGG11	SAT	78.81	46.08
	VGG19	SAT	74.64	45.52
	ResNet18	SAT	82.81	48.99
	ResNet50	SAT	82.97	48.65
$\ell_\infty, \varepsilon = 8/255$	VGG11	Fast AT	76.94	44.11
	ResNet18	Fast AT	82.60	51.30
	ResNet50	SAT	85.50	49.60
	ResNet50	Fast AT	83.73	50.65
	ResNet50	Free AT	77.74	45.41
Restricted-ImageNet				
$\ell_2, \varepsilon = 3.0$	AlexNet	SAT	87.67	75.24
	VGG16	SAT	92.40	80.91
	ResNet50	SAT	95.47	84.36

D Training Time Results

The total time required to train a ResNet50 model using different methods is reported in this section. For completeness, we report the total training time with teacher overhead included in case of RRM. The total training times on CIFAR-10 are reported in Table 9 and for Restricted-ImageNet are reported in Table 10.

Table 9: Comparing RRM training time to Free AT [22] on CIFAR-10 using ResNet50. Both methods have been accelerated using DAWNBench improvements [28]. For completeness we provided comparisons when teacher's overhead is taken into account when training models using RRM. Teachers VGG11 and ResNet18 have been trained using Fast AT.

Method	Train Time (mins)	
	w/o Teacher	w/ Teacher
SAT	1807.58	-
Fast AT	192.77	-
Free AT	58.44	-
RRM (VGG11)	30.22	57.91
RRM (ResNet18)	31.82	91.77

Table 10: Comparing RRM training time to SAT [15] on Restricted-ImageNet using VGG16 and ResNet50. An adversarially trained AlexNet model is used as teacher. For completeness we provided comparisons when teacher's overhead is taken into account when training models using RRM.

Method	Train Time (hrs)*	
	w/o Teacher	w/ Teacher
ResNet50		
SAT	253.71	-
RRM	13.66	46.69
VGG16		
SAT	400.01	-
RRM	33.30	66.33

**The ResNet50 and VGG16 models were trained on different machines due to memory constraints.*