ADVERSARIAL ROBUSTNESS AGAINST THE UNION OF MULTIPLE PERTURBATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Owing to the susceptibility of deep learning systems to adversarial attacks, there has been a great deal of work in developing (both empirically and certifiably) robust classifiers, but the vast majority has defended against single types of attacks. Recent work has looked at defending against multiple attacks, specifically on the MNIST dataset, yet this approach used a relatively complex architecture, claiming that standard adversarial training can not apply because it "overfits" to a particular norm. In this work, we show that it is indeed possible to adversarially train a robust model against a union of norm-bounded attacks, by using a natural generalization of the standard PGD-based procedure for adversarial training to multiple threat models. With this approach, we are able to train standard architectures which are robust against ℓ_∞, ℓ_2, and ℓ_1 attacks, outperforming past approaches on the MNIST dataset and providing the first CIFAR10 network trained to be simultaneously robust against $(\ell_\infty, \ell_2, \ell_1)$ threat models, which achieves adversarial accuracy rates of (47.6%, 64.3%, 53.4%) for $(\ell_\infty, \ell_2, \ell_1)$ perturbations with radius $\epsilon = (0.03, 0.5, 12)$.

1 INTRODUCTION

Machine learning algorithms have been shown to be susceptible to *adversarial examples* (Szegedy et al., 2014) through the existence of data points which can be adversarially perturbed to be misclassified, but are "close enough" to the original example to be imperceptible to the human eye. Methods to generate adversarial examples, or "attacks", typically rely on gradient information, and most commonly use variations of projected gradient descent (PGD) to maximize the loss within a small perturbation region, usually referred to as the adversary's threat model. Since then, a number of heuristic defenses have been proposed to defend against this phenomenon, e.g. distillation (Papernot et al., 2016) or more recently logit-pairing (Kannan et al., 2018). However, as time goes by, the original robustness claims of these defenses typically don't hold up to more advanced adversaries or more thorough attacks (Carlini & Wagner, 2017; Engstrom et al., 2018; Mosbach et al., 2018). One heuristic defense that seems to have survived (to this day) is to use *adversarial training* against a PGD adversary (Madry et al., 2018), which remains quite popular due to its simplicity and apparent empirical robustness. The method continues to perform well in empirical benchmarks even when compared to recent work in provable defenses, although it comes with no formal guarantees.

Some recent work, however, has claimed that adversarial training "*overfits*" to the particular type of perturbation used to generate the adversarial examples, and used this as motivation to propose a more complicated architecture in order to achieve robustness to multiple perturbation types on the MNIST dataset (Schott et al., 2019).

In this work, we offer a contrasting viewpoint: we show that it is indeed possible to use adversarial training to learn a model which is simultaneously robust against multiple types of ℓ_p norm bounded attacks (we consider ℓ_∞, ℓ_2, and ℓ_1 attacks, but the approach can apply to more general attacks). First, we show that simple generalizations of adversarial training to multiple threat models can already achieve some degree of robustness against the union of these threat models. Second, we propose a slightly modified PGD-based algorithm called multi steepest descent (MSD) for adversarial training which more naturally incorporates the different perturbations within the PGD iterates, further improving the adversarial training approach. Third, we show empirically that our approach improves

upon past work by being applicable to standard network architectures, easily scaling beyond the MNIST dataset, and outperforming past results on robustness against multiple perturbation types.

2 RELATED WORK

After their original introduction, one of the first widely-considered attacks against deep networks had been the Fast Gradient Sign Method (Goodfellow et al., 2015), which showed that a single, small step in the direction of the sign of the gradient could sometimes fool machine learning classifiers. While this worked to some degree, the Basic Iterative Method (Kurakin et al., 2017) (now typically referred to as the PGD attack) was significantly more successful at creating adversarial examples, and now lies at the core of many papers. Since then, a number of improvements and adaptations have been made to the base PGD algorithm to overcome heuristic defenses and create stronger adversaries. Adversarial attacks were thought to be safe under realistic transformations (Lu et al., 2017) until the attack was augmented to be robust to them (Athalye et al., 2018b). Adversarial examples generated using PGD on surrogate models can transfer to black box models (Papernot et al., 2017). Utilizing core optimization techniques such as momentum can greatly improve the attack success rate and transferability, and was the winner of the NIPS 2017 competition on adversarial examples (Dong et al., 2018). Uesato et al. (2018) showed that a number of ImageNet defenses were not as robust as originally thought, and Athalye et al. (2018a) defeated many of the heuristic defenses submitted to ICLR 2018 shortly after the reviewing cycle ended, all with stronger PGD variations.

Throughout this cycle of attack and defense, some defenses were uncovered that remain robust to this day. The aforementioned PGD attack, and the related defense known as adversarial training with a PGD adversary (which incorporates PGD-attacked examples into the training process) has so far remained empirically robust (Madry et al., 2018). Verification methods to certify robustness properties of networks were developed, utilizing techniques such as SMT solvers (Katz et al., 2017), SDP relaxations (Raghunathan et al., 2018b), and mixed-integer linear programming (Tjeng et al., 2019), the last of which has recently been successfully scaled to reasonably sized networks. Other work has folded verification into the training process to create provably robust networks (Wong & Kolter, 2018; Raghunathan et al., 2018a), some of which have also been scaled to even larger networks (Wong et al., 2018; Mirman et al., 2018; Gowal et al., 2018). Although some of these could potentially be extended to apply to multiple perturbations simultaneously, most of these works have focused primarily on defending against and verifying only a *single* type of adversarial perturbation at a time.

Last but most relevant to this work are adversarial defenses that attempt to be robust against multiple types of attacks simultaneously. Schott et al. (2019) used multiple variational autoencoders to construct a complex architecture for the MNIST dataset that is not as easily attacked by ℓ_∞, ℓ_2, and ℓ_0 adversaries. Importantly, Schott et al. (2019) compare to adversarial training with an ℓ_∞-bounded PGD adversary as described by Madry et al. (2018), claiming that the adversarial training defense overfits to the ℓ_∞ metric and is not robust against other types of perturbations. Following this, a number of concurrent papers have since been released. While not studied as a defense, Kang et al. (2019) study the transferability of adversarial robustness between models trained against different threat models. Croce & Hein (2019) propose a provable adversarial defense against all ℓ_p norms for $p \geq 1$ using a regularization term. Finally, Tramèr & Boneh (2019) study the theoretical and empirical trade-offs of adversarial robustness in various settings when defending against multiple adversaries, however, they use a rotation and translation adversary instead of an ℓ_2 adversary for CIFAR10.

Contributions In contrast to the claim that adversarial training overfits to a particular metric, in this work we demonstrate that adversarial training can in fact be used to learn models that are robust against a *union* of multiple perturbation models, as long as you train against a union of adversaries. First, we show that even simple aggregations of different adversarial attacks can achieve competitive universal robustness against multiple perturbations models without resorting to complex architectures. Second, we propose a modified PGD iteration that more naturally considers multiple perturbation models within the inner optimization loop of adversarial training. Third, we evaluate all approaches on the MNIST and CIFAR10 datasets, showing that our proposed generalizations of adversarial training can significantly outperform past approaches for ℓ_∞, ℓ_2, and ℓ_1 attacks. Specifically, on MNIST, our model achieves 63.7%, 82.6%, and 62.3% adversarial accuracy against all three attacks $(\ell_\infty, \ell_2, \ell_1)$

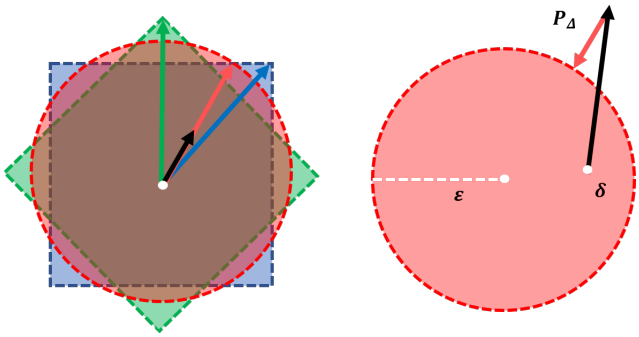

Figure 1: (left) A depiction of the steepest descent directions for ℓ_∞, ℓ_2, and ℓ_1 norms. The gradient is the black arrow, and the α radius step sizes and their corresponding steepest descent directions ℓ_∞, ℓ_2, and ℓ_1 are shown in blue, red, and green respectively. (right) An example of the projection back to an ℓ_2 ball of radius ϵ after a steepest descent step from the starting perturbation δ. The steepest descent step is the black arrow, and the corresponding projection back onto the ℓ_2 ball is red arrow.

for $\epsilon = (0.3, 1.5, 12)$ respectively, substantially improving upon the multiple-perturbation-model robustness described in Schott et al. (2019). Unlike past work, we also train a CIFAR10 model, which achieves 47.6%, 64.3%, and 53.4% adversarial accuracy against all three attacks $(\ell_\infty, \ell_2, \ell_1)$ for $\epsilon = (0.03, 0.5, 12)$. Finally, for completeness, we also draw relevant comparisons to concurrent work at revised ϵ thresholds, and show that the relative advantage of our approach still holds.

3 Overview of adversarial training

Adversarial training is an approach to learn a classifier which minimizes the worst case loss within some perturbation region (the threat model). Specifically, for some network f_θ parameterized by θ, loss function ℓ, and training data $\{x_i, y_i\}_{i=1\ldots n}$, the robust optimization problem of minimizing the worst case loss within ℓ_p norm-bounded perturbations with radius ϵ is

$$\min_\theta \sum_i \max_{\delta \in \Delta_{p,\epsilon}} \ell(f_\theta(x_i + \delta), y_i), \tag{1}$$

where $\Delta_{p,\epsilon} = \{\delta : \|\delta\|_p \leq \epsilon\}$ is the ℓ_p ball with radius ϵ centered around the origin. To simplify the notation, we will abbreviate $\ell(f_\theta(x + \delta), y) = \ell(x + \delta; \theta)$.

3.1 Solving the inner optimization problem

We first look at solving the inner maximization problem, namely

$$\max_{\delta \in \Delta_{p,\epsilon}} \ell(x + \delta; \theta). \tag{2}$$

This is the problem addressed by the "attackers" in the space of adversarial examples, hoping that the classifier can be tricked by the optimal perturbed image, $x + \delta^\star$. Typical solutions solve this problem by running a form of projected gradient descent, which iteratively takes steps in the gradient direction to increase the loss followed by a projection step back onto the feasible region, the ℓ_p ball. Since the gradients at the example points themselves (i.e., $\delta = 0$) are typically too small to make efficient progress, more commonly used is a variation called *projected steepest descent*.

Steepest descent For some norm $\|\cdot\|_p$ and step size α, the direction of steepest descent on the loss function ℓ for a perturbation δ is

$$v_p(\delta) = \arg\max_{\|v\|_p \leq \alpha} v^T \nabla \ell(x + \delta; \theta). \tag{3}$$

Then, instead of taking gradient steps, steepest descent uses the following iteration

$$\delta^{(t+1)} = \delta^{(t)} + v_p(\delta^{(t)}). \tag{4}$$

In practice, the norm used in steepest descent is typically taken to be the same ℓ_p norm used to define the perturbation region $\Delta_{p,\epsilon}$. However, depending on the norm used, the direction of steepest descent can be quite different from the actual gradient (Figure 1). Note that a single steepest descent step with respect to the ℓ_∞ norm reduces to $v_\infty(x) = \alpha \cdot \text{sign}(\nabla \ell(x + \delta; \theta))$, better known in the adversarial examples literature as the Fast Gradient Sign Method (Goodfellow et al., 2015).

Projections The second component of projected steepest descent for adversarial examples is to project iterates back onto the ℓ_p ball around x. Specifically, projected steepest descent performs the following iteration

$$\delta^{(t+1)} = \mathcal{P}_{\Delta_{p,\epsilon}}\left(\delta^{(t)} + v_p(\delta^{(t)})\right) \tag{5}$$

where $\mathcal{P}_{\Delta_{p,\epsilon}}(\delta)$ is the standard projection operator that finds the perturbation $\delta' \in \Delta_{p,\epsilon}$ that is "closest" in Euclidean space to the input δ, defined as

$$\mathcal{P}_{\Delta_{p,\epsilon}}(\delta) = \underset{\delta' \in \Delta_{p,\epsilon}}{\arg\min} \|\delta - \delta'\|_2^2. \tag{6}$$

Visually, a depiction of this procedure (steepest descent followed by a projection onto the perturbation region) for an ℓ_2 adversary can be found in Figure 1. If we instead project the steepest descent directions with respect to the ℓ_∞ norm onto the ℓ_∞ ball of allowable perturbations, the projected steepest descent iteration reduces to

$$\begin{aligned} \delta^{(t+1)} &= P_{\Delta_{\infty,\epsilon}}(\delta^{(t)} + v_\infty(\delta^{(t)})) \\ &= \underset{[-\epsilon,\epsilon]}{\text{clip}}\left(\delta^{(t)} + \alpha \cdot \text{sign}(\nabla \ell(x + \delta^{(t)}; \theta))\right) \end{aligned} \tag{7}$$

where $\text{clip}_{[-\epsilon,+\epsilon]}$ "clips" the input to lie within the range $[-\epsilon, \epsilon]$. This is exactly the Basic Iterative Method used in Kurakin et al. (2017), typically referred to in the literature as an ℓ_∞ PGD adversary.

3.2 Solving the outer optimization problem

We next look at how to solve the outer optimization problem, or the problem of learning the weights θ that minimize the loss of our classifier. While many approaches have been proposed in the literature, we will focus on a heuristic called adversarial training, which has generally worked well in practice.

Adversarial training Although solving the min-max optimization problem may seem daunting, a classical result known as Danskin's theorem (Danskin, 1967) says that the gradient of a maximization problem is equal to the gradient of the objective evaluated at the optimum. For learning models that minimize the robust optimization problem from Equation (1), this means that

$$\nabla_\theta \left(\sum_i \max_{\delta \in \Delta_{p,\epsilon}} \ell(x_i + \delta; \theta) \right) = \sum_i \nabla_\theta \ell(x_i + \delta^*(x_i); \theta) \tag{8}$$

where $\delta^*(x_i) = \arg\max_{\delta \in \Delta_{p,\epsilon}} \ell(x_i + \delta; \theta)$. In other words, this means that in order to backpropagate through the robust optimization problem, we can solve the inner maximization and backpropagate through the solution. Adversarial training does this by empirically maximizing the inner problem with a PGD adversary. Note that since the inner problem is not solved exactly, Danskin's theorem does not strictly apply. However, in practice, adversarial training does seem to provide good empirical robustness, at least when evaluated against the ℓ_p threat model it was trained against.

4 Adversarial training for multiple perturbation models

We can now consider the core of this work, adversarial training procedures against multiple threat models. More formally, let $\mathcal{S}$ represent a set of threat models, such that $p \in \mathcal{S}$ corresponds to the ℓ_p perturbation model $\Delta_{p,\epsilon}$, and let $\Delta_{\mathcal{S}} = \bigcup_{p \in \mathcal{S}} \Delta_{p,\epsilon}$ be the union of all perturbation models in $\mathcal{S}$. Note that the ϵ chosen for each ball is *not* typically the same, but we still use the same notation ϵ for simplicity, since the context will always make clear which ℓ_p-ball we are talking about. Then, the generalization of the robust optimization problem in Equation (1) to multiple perturbation models is

$$\min_\theta \sum_i \max_{\delta \in \Delta_{\mathcal{S}}} \ell(x_i + \delta; \theta). \tag{9}$$

The key difference is in the inner maximization, where the worst case adversarial loss is now taken over *multiple* ℓ_p perturbation models. In order to perform adversarial training, using the same motivational idea from Danskin's theorem, we can backpropagate through the inner maximization by first finding (empirically) the optimal perturbation,

$$\delta^* = \arg\max_{\delta \in \Delta_{\mathcal{S}}} \ell(x + \delta; \theta). \tag{10}$$

To find the optimal perturbation over the union of threat models, we begin by considering straightforward generalizations of standard adversarial training, which will use PGD to approximately solve the inner maximization over multiple adversaries.

4.1 Simple combinations of multiple perturbations

First, we propose two simple approaches to generalizing adversarial training to multiple threat models. These methods already perform quite well in practice and are competitive with existing, state-of-the-art approaches without relying on complicated architectures, showing that adversarial training can in fact generalize to multiple threat models.

Worst-case perturbation One way to generalize adversarial training to multiple threat models is to use each threat model independently, and train on the adversarial perturbation that achieved the maximum loss. Specifically, for each adversary $p \in \mathcal{S}$, we solve the innermost maximization with an ℓ_p PGD adversary to get an approximate worst-case perturbation δ_p,

$$\delta_p = \arg\max_{\delta \in \Delta_{p,\epsilon}} \ell(x + \delta; \theta), \tag{11}$$

and then approximate the maximum over all adversaries as

$$\delta^* \approx \arg\max_{\delta_p} \ell(x + \delta_p; \theta). \tag{12}$$

When $|\mathcal{S}| = 1$, then this reduces to standard adversarial training. Note that if each PGD adversary solved their subproblem from Equation (11) exactly, then this is exactly the optimal perturbation $\delta^\star$.

PGD augmentation with all perturbations Another way to generalize adversarial training is to train on all the adversarial perturbations for all $p \in \mathcal{S}$ to form a larger adversarial dataset. Specifically, instead of solving the robust problem for multiple adversaries in Equation (9), we instead solve

$$\min_\theta \sum_i \sum_{p \in \mathcal{S}} \max_{\delta \in \Delta_{p,\epsilon}} \ell(x_i + \delta; \theta) \tag{13}$$

by using individual ℓ_p PGD adversaries to approximate the inner maximization for each threat model. Again, this reduces to standard adversarial training when $|\mathcal{S}| = 1$.

While these methods work quite well in practice (which is shown later in Section 5), both approaches solve the inner maximization problem independently for each adversary, so each individual PGD adversary is not taking advantage of the fact that the perturbation region is enlarged by other threat models. To take advantage of the full perturbation region, we propose a modification to standard adversarial training, which combines information from all considered threat models into a single PGD adversary that is potentially stronger than the combination of independent adversaries.

4.2 Multi steepest descent

To create a PGD adversary with full knowledge of the perturbation region, we propose an algorithm that incorporates the different threat models within each step of projected steepest descent. Rather than generating adversarial examples for each threat model with separate PGD adversaries, the core idea is to create a single adversarial perturbation by simultaneously maximizing the worst case loss over all threat models at each projected steepest descent step. We call our method *multi steepest descent* (MSD), which can be summarized as the following iteration:

$$\begin{aligned} \delta_p^{(t+1)} &= P_{\Delta_{p,\epsilon}}(\delta^{(t)} + v_p(\delta^{(t)})) \ \text{ for } \ p \in \mathcal{S} \\ \delta^{(t+1)} &= \arg\max_{\delta_p^{(t+1)}} \ell(x + \delta_p^{(t+1)}) \end{aligned} \tag{14}$$

Algorithm 1 Multi steepest descent for learning classifiers that are simultaneously robust to ℓ_p attacks for $p \in \mathcal{S}$

Input: classifier f_θ, data x, labels y
Parameters: ϵ_p, α_p for $p \in \mathcal{S}$, maximum iterations T, loss function ℓ
$\delta^{(0)} = 0$
for $t = 0 \ldots T-1$ **do**
  for $p \in \mathcal{S}$ **do**
    $\delta_p^{(t+1)} = P_{\Delta_{p,\epsilon}}(\delta^{(t)} + v_p(\delta^{(t)}))$
  end for
  $\delta^{(t+1)} = \arg\max_{\delta_p^{(t+1)}} \ell(f_\theta(x + \delta_p^{(t+1)}), y)$
end for
return $\delta^{(T)}$

The key difference here is that at each iteration of MSD, we choose a projected steepest descent direction that maximizes the loss over all attack models $p \in \mathcal{S}$, whereas standard adversarial training and the simpler approaches use comparatively myopic PGD subroutines that only use one threat model at a time. The full algorithm is in Algorithm 1, and can be used as a drop in replacement for standard PGD adversaries to learn robust classifiers with adversarial training. We direct the reader to Appendix A for a complete description of steepest descent directions and projection operators for ℓ_∞, ℓ_2, and ℓ_1 norms[1].

5 Results

In this section, we present experimental results on using generalizations of adversarial training to achieve simultaneous robustness to ℓ_∞, ℓ_2, and ℓ_1 perturbations on the MNIST and CIFAR10 datasets. Our primary goal is to show that adversarial training can in fact be adapted to a union of perturbation models using standard architectures to achieve competitive results, without the pitfalls described by Schott et al. (2019). Our results improve upon the state-of-the-art in three key ways. First, we can use simpler, standard architectures for image classifiers, without relying on complex architectures or input binarization. Second, our method is able to learn a single MNIST model which is simultaneously robust to all three threat models, whereas previous work was only robust against two at a time. Finally, our method is easily scalable to datasets beyond MNIST, providing the first CIFAR10 model trained to be simultaneously robust against ℓ_∞, ℓ_2, and ℓ_1 adversaries.

We trained models using both the simple generalizations of adversarial training to multiple adversaries and also using MSD. Since the analysis by synthesis model is not scalable to CIFAR10, we additionally trained CIFAR10 models against individual PGD adversaries to measure the changes and tradeoffs in universal robustness. We evaluated these models with a broad suite of both gradient and non-gradient based attacks using Foolbox[2] (the same attacks used by Schott et al. (2019)), and also incorporated all the PGD-based adversaries discussed in this paper. All aggregate statistics that combine multiple attacks compute the worst case error rate over all attacks for *each* example.

Summaries of these results at specific thresholds can be found in Tables 1 and 2, where B-ABS and ABS refer to binarized and non-binarized versions of the analysis by synthesis models from Schott et al. (2019), P_p refers to a model trained against a PGD adversary with respect to the p-norm, Worst-PGD and PGD-Aug refer to models trained using the worst-case and data augmentation generalizations of adversarial training, and MSD refers to models trained using multi steepest descent. Full tables containing the complete breakdown of these numbers over all individual attacks used in the evaluation are in Appendix C.

[1]The pure ℓ_1 steepest descent step is inefficient since it only updates one coordinate at a time. It can be improved by taking steps on multiple coordinates, similar to that used in Tramèr & Boneh (2019), and is also explained in Appendix A.

[2]https://github.com/bethgelab/foolbox (Rauber et al., 2017)

Table 1: Summary of adversarial accuracy results for MNIST (higher is better)

	P_∞	P_2	P_1	B-ABS[4]	ABS[4]	Worst PGD	PGD Aug	MSD
Clean Accuracy	99.1%	99.4%	98.9%	99%	99%	98.9%	99.1%	98.2%
ℓ_∞ attacks ($\epsilon = 0.3$)	90.3%	0.4%	0.0%	77%	8%	68.4%	83.7%	63.7%
ℓ_2 attacks ($\epsilon = 1.5$)	45.3%	87.0%	70.3%	39%	80%	82.1%	75.0%	82.6%
ℓ_1 attacks ($\epsilon = 12$)	1.4%	43.4%	71.8%	82%	78%	54.6%	15.6%	62.3%
All Attacks	1.4%	0.4%	0.0%	39%	8%	53.7%	15.6%	**58.7**%

5.1 Experimental setup

Architectures and hyperparameters For MNIST, we use a four layer convolutional network with two convolutional layers consisting of 32 and 64 5×5 filters and 2 units of padding, followed by a fully connected layer with 1024 hidden units, where both convolutional layers are followed by 2×2 Max Pooling layers and ReLU activations (this is the same architecture used by Madry et al. (2018)). This is in contrast to past work on MNIST, which relied on per-class variational autoencoders to achieve robustness against multiple threat models (Schott et al., 2019), which was also not easily scalable to larger datasets. Since our methods have the same complexity as standard adversarial training, they also easily apply to standard CIFAR10 architectures, and in this paper we use the well known pre-activation version of the ResNet18 architecture consisting of nine residual units with two convolutional layers each (He et al., 2016).

A complete description of the hyperparameters used is in Appendix B, with hyperparameters for PGD adversaries in Appendix B.1, and hyperparameters for adversarial training in Appendix B.2. All reported ϵ are for images scaled to be between the range $[0, 1]$. All experiments can be run on modern GPU hardware (e.g. a single 1080ti).

Attacks used for evaluation To evaluate the model, we incorporate the attacks from Schott et al. (2019) as well as our PGD based adversaries using projected steepest descent, however we provide a short description here. Note that we exclude attacks based on gradient estimation, since the gradient for the standard architectures used here are readily available.

For ℓ_∞ attacks, although we find the ℓ_∞ PGD adversary to be quite effective, for completeness, we additionally use the Foolbox implementations of Fast Gradient Sign Method (Goodfellow et al., 2015), PGD adversary (Madry et al., 2018), and the Momentum Iterative Method (Dong et al., 2018).

For ℓ_2 attacks, in addition to the ℓ_2 PGD adversary, we use the Foolbox implementations of the same PGD adversary, the Gaussian noise attack (Rauber et al., 2017), the boundary attack (Brendel et al., 2017), DeepFool (Moosavi-Dezfooli et al., 2016), the pointwise attack (Schott et al., 2019), DDN based attack (Rony et al., 2018), and C&W attack (Carlini & Wagner, 2017).

For ℓ_1 attacks, we use both the ℓ_1 PGD adversary as well as additional Foolbox implementations of ℓ_0 attacks at the same radius, namely the salt & pepper attack (Rauber et al., 2017) and the pointwise attack (Schott et al., 2019). Note that an ℓ_1 adversary with radius ϵ is strictly stronger than an ℓ_0 adversary with the same radius, and so we choose to explicitly defend against ℓ_1 perturbations instead of the ℓ_0 perturbations considered by Schott et al. (2019).

We make **10 random restarts** for each of the evaluation results mentioned hereon for both MNIST and CIFAR10 [3]. We encourage future work in this area to incorporate the same, since the success of all attacks, specially decision based or gradient free ones, is observed to increase significantly over restarts.

[3] All attacks were run on a subset of the first 1000 test examples with 10 random restarts, with the exception of Boundary Attack, which by default makes 25 trials per iteration and DDN based Attack which does not benefit from the same owing to a deterministic initialization of δ.

[4] Results are from Schott et al. (2019), which used an ℓ_0 threat model of the same radius and evaluated against ℓ_0 attacks. So the reported number here is an upper bound on the ℓ_1 adversarial accuracy. Further, they evaluate their model without restarts and the adversarial accuracy against all attacks is an upper bound based on the reported accuracies for the individual threat models. Finally, all ABS results were computed using numerical gradient estimation, since gradients are not readily available.

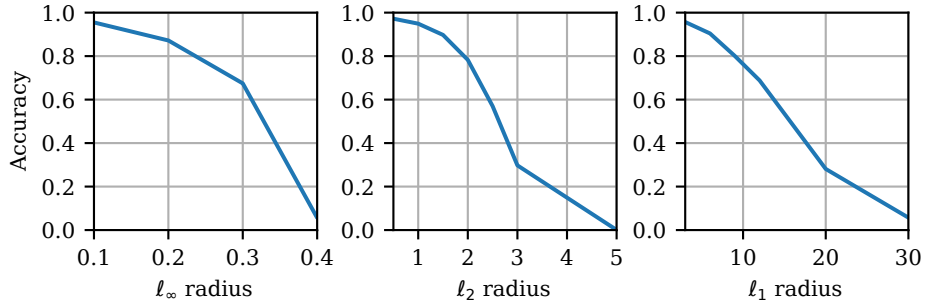

Figure 2: Robustness curves showing the adversarial accuracy for the MNIST model trained with MSD against ℓ_∞ (left), ℓ_2 (middle), and ℓ_1 (right) threat models over a range of epsilon.

Table 2: Summary of adversarial accuracy results for CIFAR10 (higher is better)

	P_∞	P_2	P_1	Worst-PGD	PGD-Aug	MSD
Clean accuracy	83.3%	90.2%	73.3%	81.0%	84.6%	81.7%
ℓ_∞ attacks ($\epsilon = 0.03$)	50.7%	28.3%	0.2%	44.9%	42.5%	47.6%
ℓ_2 attacks ($\epsilon = 0.5$)	57.3%	61.6%	0.0%	61.7%	65.0%	64.3%
ℓ_1 attacks ($\epsilon = 12$)	16.0%	46.6%	7.9%	39.4%	54.0%	53.4%
All attacks	15.6%	27.5%	0.0%	34.9%	40.6%	**46.1**%

5.2 MNIST

We first present results on the MNIST dataset, which are summarized in Table 1 (a more detailed breakdown over each individual attack is in Appendix C.1). While considered an "easy" dataset, we note that the previous state-of-the-art result for multiple threat models on MNIST (and our primary comparison) is only able to defend against two out of three threat models at a time (Schott et al., 2019) using comparatively complex variational autoencoder architectures. In contrast, we see that both simple generalizations of adversarial training are able to achieve competitive results on standard models, notably being able to defend against all three threat models simultaneously, while the model trained with MSD performs even better, achieving error rates of 63.7%, 82.6%, and 62.3% for ℓ_∞, ℓ_2, and ℓ_1 perturbations with radius $\epsilon = 0.3$, 1.5, and 12. A complete robustness curve over a range of epsilons for the MSD model over each threat model can be found in Figure 2, and robustness curves for other models are deferred to Appendix C.1. A comparison of our results with concurrent work (Tramèr & Boneh, 2019) can be found in Appendix D.

5.3 CIFAR10

Next, we present results on the CIFAR10 dataset, which are summarized in Table 2 (a more detailed breakdown over each individual attack is in Appendix C.2). Our MSD model achieves $(47.6\%, 64.3\%, 53.4\%)$ adversarial accuracy for $(\ell_\infty, \ell_2, \ell_1)$ perturbations of size $\epsilon = (0.03, 0.5, 12)$, reaching an overall adversarial adversarial accuracy of 46.1% over all threat models. Interestingly, note that the P_1 model trained against an ℓ_1 PGD adversary is not very robust when evaluated against other attacks, even though it can defend reasonably well against the ℓ_1 PGD attack in isolation (Table 4 in Appendix C.2). A complete robustness curve over a range of epsilons for the MSD model over each threat model can be found in Figure 3, and robustness curves for other models are deferred to Appendix C.2. A comparison of our results with concurrent work (Tramèr & Boneh, 2019) can be found in Appendix D.

6 Conclusion

In this paper, we showed that adversarial training can be quite effective when training against a union of multiple perturbation models. We compare two simple generalizations of adversarial training and an improved adversarial training procedure, multi steepest descent, which incorporates the different perturbation models directly into the direction of steepest descent. MSD based adversarial training

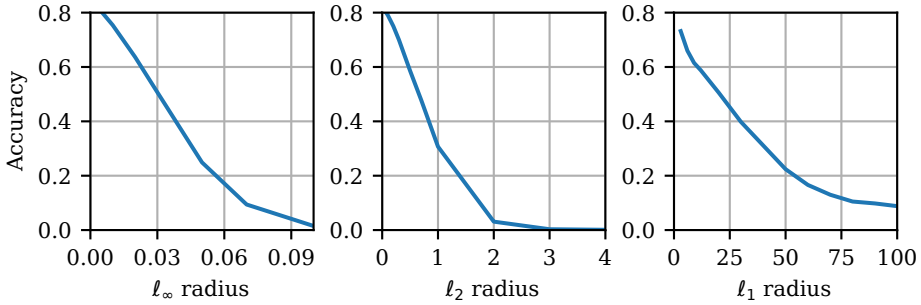

Figure 3: Robustness curves showing the adversarial accuracy for the CIFAR10 model trained with MSD against ℓ_∞ (left), ℓ_2 (middle), and ℓ_1 (right) threat models over a range of epsilon.

procedure is able to outperform past approaches, demonstrating that adversarial training can in fact learn networks that are robust to multiple perturbation models (as long as they are included in the threat model) while being scalable beyond MNIST and using standard architectures.

REFERENCES

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, July 2018a. URL https://arxiv.org/abs/1802.00420.

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 284–293, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018b. PMLR. URL http://proceedings.mlr.press/v80/athalye18b.html.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 39–57. IEEE, 2017.

Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: Elastic-net attacks to deep neural networks via adversarial examples, 2017.

Francesco Croce and Matthias Hein. Provable robustness against all adversarial l_p-perturbations for $p \geq 1$. *CoRR*, abs/1905.11213, 2019. URL http://arxiv.org/abs/1905.11213.

John M Danskin. *The theory of max-min and its application to weapons allocation problems*, volume 5. Springer Science & Business Media, 1967.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 272–279, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390191. URL http://doi.acm.org/10.1145/1390156.1390191.

Logan Engstrom, Andrew Ilyas, and Anish Athalye. Evaluating and understanding the robustness of adversarial logit pairing. *arXiv preprint arXiv:1807.10272*, 2018.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *CoRR*, abs/1810.12715, 2018. URL http://arxiv.org/abs/1810.12715.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Daniel Kang, Yi Sun, Tom Brown, Dan Hendrycks, and Jacob Steinhardt. Transfer of adversarial robustness between perturbation types. *arXiv preprint arXiv:1905.01034*, 2019.

Harini Kannan, Alexey Kurakin, and Ian J. Goodfellow. Adversarial logit pairing. *CoRR*, abs/1803.06373, 2018. URL http://arxiv.org/abs/1803.06373.

Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. *arXiv preprint arXiv:1702.01135*, 2017.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *ICLR Workshop*, 2017. URL https://arxiv.org/abs/1607.02533.

Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv preprint arXiv:1707.03501*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning (ICML)*, 2018. URL https://www.icml.cc/Conferences/2018/Schedule?showEvent=2477.

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*, pp. 2574–2582, 2016.

Marius Mosbach, Maksym Andriushchenko, Thomas Trost, Matthias Hein, and Dietrich Klakow. Logit pairing methods can fool gradient-based attacks. *arXiv preprint arXiv:1810.12042*, 2018.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pp. 582–597. IEEE, 2016.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pp. 506–519, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4944-4. doi: 10.1145/3052973.3053009. URL http://doi.acm.org/10.1145/3052973.3053009.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018a. URL https://openreview.net/forum?id=Bys4ob-Rb.

Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10900–10910. Curran Associates, Inc., 2018b. URL http://papers.nips.cc/paper/8285-semidefinite-relaxations-for-certifying-robustness-to-adversarial-examples.pdf.

Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017. URL http://arxiv.org/abs/1707.04131.

Jérôme Rony, Luiz G. Hafemann, Luiz S. Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based L2 adversarial attacks and defenses. *CoRR*, abs/1811.09600, 2018. URL http://arxiv.org/abs/1811.09600.

Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1EHOsC9tX.

Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

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*, 2014. URL http://arxiv.org/abs/1312.6199.

Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyGIdiRqtm.

Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. *arXiv preprint arXiv:1904.13000*, 2019.

Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aaron van den Oord. Adversarial risk and the dangers of evaluating against weak attacks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5025–5034, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/uesato18a.html.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pp. 5283–5292, 2018.

Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. Scaling provable adversarial defenses. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 8410–8419. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8060-scaling-provable-adversarial-defenses.pdf.

Algorithm 2 Projection of some perturbation $\delta \in \mathbb{R}^n$ onto the ℓ_1 ball with radius ϵ. We use $|\cdot|$ to denote element-wise absolute value.

Input: perturbation δ, radius ϵ
Sort $|\delta|$ into $\gamma : \gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_n$
$\rho := \max\left\{j \in [n] : \gamma_j - \frac{1}{j}\left(\sum_{r=1}^{j} \gamma_r - \epsilon\right) > 0\right\}$
$\eta := \frac{1}{\rho}\left(\sum_{i=1}^{\rho} \gamma_i - \epsilon\right)$
$z_i := \text{sign}(\delta_i) \max\{\gamma_i - \eta, 0\}$ for $i = 1 \ldots n$
return z

A STEEPEST DESCENT AND PROJECTIONS FOR ℓ_∞, ℓ_2, AND ℓ_1 ADVERSARIES

In this section, we show what the steepest descent and projection steps are for ℓ_p adversaries for $p \in \{\infty, 2, 1\}$; these are standard results, but included for a complete description of the algorithms. Note that this differs slightly from the adversaries considered in Schott et al. (2019): while they used an ℓ_0 adversary, we opted to use an ℓ_1 adversary with the same radius. The ℓ_0 ball with radius ϵ is contained within an ℓ_1 ball with the same radius, so achieving robustness against an ℓ_1 adversary is strictly more difficult.

ℓ_∞ space The direction of steepest descent with respect to the ℓ_∞ norm is

$$v_\infty(\delta) = \alpha \cdot \text{sign}(\nabla l(x + \delta; \theta)) \tag{15}$$

and the projection operator onto $\Delta_{\infty,\epsilon}$ is

$$\mathcal{P}_{\Delta_{\infty,\epsilon}}(\delta) = \text{clip}_{[-\epsilon,\epsilon]}(\delta) \tag{16}$$

ℓ_2 space The direction of steepest descent with respect to the ℓ_2 norm is

$$v_2(\delta) = \alpha \cdot \frac{\nabla \ell(x + \delta; \theta)}{\|\nabla \ell(x + \delta; \theta)\|_2} \tag{17}$$

and the projection operator onto the ℓ_2 ball around x is

$$\mathcal{P}_{\Delta_{2,\epsilon}}(\delta) = \epsilon \cdot \frac{\delta}{\max\{\epsilon, \|\delta\|_2\}} \tag{18}$$

ℓ_1 space The direction of steepest descent with respect to the ℓ_1 norm is

$$v_1(\delta) = \alpha \cdot \text{sign}\left(\frac{\partial \ell(x + \delta; \theta)}{\partial \delta_{i^\star}}\right) \cdot e_{i^\star} \tag{19}$$

where

$$i^\star = \arg\max_i |\nabla l(x + \delta; \theta)_i| \tag{20}$$

and e_{i^*} is a unit vector with a one in position i^*. Finally, the projection operator onto the ℓ_1 ball,

$$\mathcal{P}_{\Delta_{1,\epsilon}}(\delta) = \arg\min_{\delta' : \|\delta'\|_1 \leq \epsilon} \|\delta - \delta'\|_2^2, \tag{21}$$

can be solved with Algorithm 2, and we refer the reader to Duchi et al. (2008) for its derivation.

A.1 ENHANCED ℓ_1 STEEPEST DESCENT STEP

Note that the steepest descent step for ℓ_1 only updates a single coordinate per step. This can be quite inefficient, as pointed out by Tramèr & Boneh (2019). To tackle this issue, and also empirically improve the attack success rate, Tramèr & Boneh (2019) instead select the top k coordinates according to Equation 20 to update. In this work, we adopt a similar but slightly modified scheme: we randomly sample k to be some integer within some range $[k_1, k_2]$, and update each coordinate with step size $\alpha' = \alpha/k$. We find that the randomness induced by varying the number of coordinates aids in avoiding the gradient masking problem observed by Tramèr & Boneh (2019).

A.2 RESTRICTING THE STEEPEST DESCENT COORDINATE

The steepest descent direction for both the ℓ_0 and ℓ_1 norm end up selecting a single coordinate direction to move the perturbation. However, if the perturbation is already at the boundary of pixel space (for MNIST, this is the range [0,1] for each pixel), then it's possible for the PGD adversary to get stuck in a loop trying to use the same descent direction to escape pixel space. To avoid this, we only allow the steepest descent directions for these two attacks to choose coordinates that keep the image in the range of real pixels.

B EXPERIMENTAL DETAILS

B.1 HYPERPARAMETERS FOR PGD ADVERSARIES

In this section, we describe the parameters used for all PGD adversaries in this paper.

MNIST The ℓ_∞ adversary used a step size $\alpha = 0.01$ within a radius of $\epsilon = 0.3$ for 50 iterations.

The ℓ_2 adversary used a step size $\alpha = 0.1$ within a radius of $\epsilon = 1.5$ for 100 iterations.

The ℓ_1 adversary used a step size of $\alpha = 0.05$ within a radius of $\epsilon = 12$ for 50 iterations. By default the attack is run with two restarts, once starting with δ = 0 and once by randomly initializing δ in the allowable perturbation ball. k_1 = 5, k_2 = 20 as described in A.1.

The MSD adversary used step sizes of $\alpha = (0.01, 0.2, 0.05)$ for the $(\ell_\infty, \ell_2, \ell_1)$ directions within a radius of $\epsilon = (0.3, 1.5, 12)$ for 100 iterations.

At test time, we increase the number of iterations to $(100, 200, 100)$ for $(\ell_\infty, \ell_2, \ell_1)$.

CIFAR10 The ℓ_∞ adversary used a step size $\alpha = 0.003$ within a radius of $\epsilon = 0.03$ for 40 iterations.

The ℓ_2 adversary used a step size $\alpha = 0.05$ within a radius of $\epsilon = 0.5$ for 50 iterations.

The ℓ_1 adversary used a step size $\alpha = 0.1$ within a radius of $\epsilon = 12$ for 50 iterations. k_1 = 5, k_2 = 20 as described in A.1.

The MSD adversary used step sizes of $\alpha = (0.003, 0.05, 0.05)$ for the $(\ell_\infty, \ell_2, \ell_1)$ directions within a radius of $\epsilon = (0.03, 0.3, 12)$ for 50 iterations. Note that the MSD model trained for ℓ_2 radius of 0.3 is in fact robust to a higher radius of 0.5.

B.2 TRAINING HYPERPARAMETERS

In this section, we describe the parameters used for adversarial training. For all the models, we used the SGD optimizer with momentum 0.9 and weight decay $5 \cdot 10^{-4}$.

MNIST We train the models to a maximum of 20 epochs. We used a variation of the learning rate schedule from Smith (2018), which is piecewise linear from 0 to 0.1 over the first 7 epochs, down to 0.001 over the next 8 epochs, and finally back down to 0.0001 in the last 5 epochs.

CIFAR10 We used a variation of the learning rate schedule from Smith (2018) to achieve super-convergence in 50 epochs, which is piecewise linear from 0 to 0.1 over the first 20 epochs, down to 0.005 over the next 20 epochs, and finally back down to 0 in the last 10 epochs.

C EXTENDED RESULTS

Here, we show the full tables which break down the overall adversarial error rates over individual attacks for both MNIST and CIFAR10, along with robustness curves for all models in the paper.

Table 3: Summary of adversarial accuracy results for MNIST

	P_∞	P_2	P_1	B-ABS	ABS	Worst PGD	PGD Aug	MSD
Clean Accuracy	99.1%	99.4%	98.9%	99%	99%	98.9%	99.1%	98.2%
PGD-ℓ_∞	90.3%	0.4%	0.0%	-	-	68.4%	83.7%	63.7%
FGSM	94.9%	68.6%	6.4%	85%	34%	82.4%	90.9%	81.8%
PGD-Foolbox	92.1%	8.5%	0.1%	86%	13%	72.1%	85.7%	67.9%
MIM	92.3%	14.5%	0.1%	85%	17%	73.9%	87.3%	71.0%
ℓ_∞ attacks ($\epsilon = 0.3$)	**90.3**%	0.4%	0.0%	77%	8%	68.4%	83.7%	63.7%
PGD-ℓ_2	83.8%	87.0%	70.8%	-	-	85.3%	87.9%	84.2%
PGD-Foolbox	93.4%	89.7%	74.4%	63%	87%	86.9%	91.5%	86.9%
Gaussian Noise	98.9%	99.6%	98.0%	89%	98%	97.4%	99.0%	97.8%
Boundary Attack	52.6%	92.1%	83.0%	91%	83%	86.9%	79.1%	88.6%
DeepFool	95.1%	92.2%	76.5%	41%	83%	87.9%	93.5%	87.9%
Pointwise Attack	74.3%	97.4%	96.6%	87%	94%	92.7%	89.0%	95.1%
DDN	82.7%	87.0%	70.8%	-	-	85.1%	85.2%	84.3%
CWL2	88.2%	88.1%	75.5%	-	-	85.2%	87.5%	85.1%
ℓ_2 attacks ($\epsilon = 1.5$)	45.3%	**87.0**%	70.3%	39%	80%	82.1%	75.0%	82.6%
PGD-ℓ_1	51.8%	49.9%	71.8%	-	-	66.5%	57.4%	64.8%
Salt & Pepper	55.5%	96.3%	95.6%	96%	95%	86.4%	71.9%	92.2%
Pointwise Attack	2.4%	66.4%	85.2%	82%	78%	60.1%	17.1%	72.8%
ℓ_1 attacks ($\epsilon = 12$)	1.4%	43.4%	71.8%	**82**%	78%	54.6%	15.6%	62.3%
All attacks	1.4%	0.4%	0.0%	39%	8%	53.7%	15.6%	**58.7**%

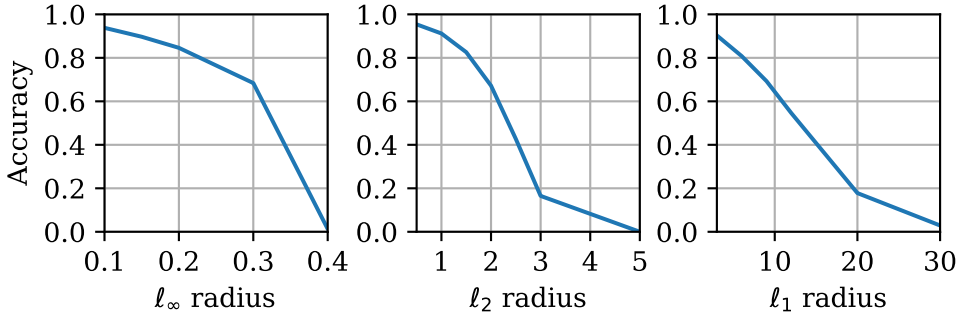

Figure 4: Robustness curves showing the adversarial accuracy for the MNIST model trained with the worst case generalization for adversarial training (Worst-PGD) against ℓ_∞ (left), ℓ_2 (middle), and ℓ_1 (right) threat models over a range of epsilon.

C.1 MNIST RESULTS

Expanded table of results Table 3 contains the full table of results for all attacks on all models on the MNIST dataset. All attacks were run on a subset of the first 1000 test examples with 10 random restarts, with the exception of Boundary Attack, which by default makes 25 trials per iteration, and DDN attack, which does not benefit from restarts owing to a deterministic starting point. Note that the results for B-ABS and ABS models are from Schott et al. (2019), which uses gradient estimation techniques whenever a gradient is needed, and the robustness against all attacks for B-ABS and ABS is an upper bound based on the reported results. Further, these models are not evaluated with restarts, pushing the reported results even higher than actual.

Robustness curves Here, we plot the full robustness curves for the remaining models trained with the simpler generalizations of adversarial training, namely the worst case method (Figure 4) and the data augmentation method (Figure 5).

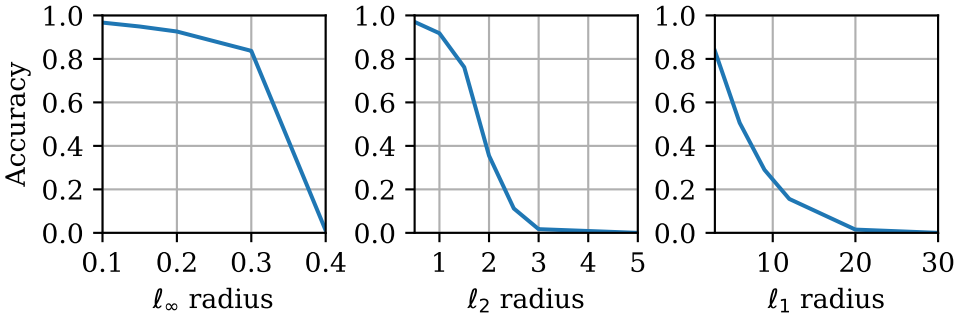

Figure 5: Robustness curves showing the adversarial accuracy for the MNIST model trained with the data augmentation generalization for adversarial training (PGD-Aug) against ℓ_∞ (left), ℓ_2 (middle), and ℓ_1 (right) threat models over a range of epsilon.

Table 4: Summary of adversarial accuracy results for CIFAR10

	P_∞	P_2	P_1	Worst-PGD	PGD-Aug	MSD
Clean accuracy	83.3%	90.2%	73.3%	81.0%	84.6%	81.7%
PGD-ℓ_∞	50.3%	48.4%	29.8%	44.9%	42.8%	49.8%
FGSM	57.4%	43.4%	12.7%	54.9%	51.9%	55.0%
PGD-Foolbox	52.3%	28.5%	0.6%	48.9%	44.6%	49.8%
MIM	52.7%	30.4%	0.7%	49.9%	46.1%	50.6%
ℓ_∞ attacks ($\epsilon = 0.03$)	50.7%	28.3%	0.2%	44.9%	42.5%	47.6%
PGD-ℓ_2	59.0%	62.1%	28.9%	64.1%	66.9%	66.0%
PGD-Foolbox	61.6%	64.1%	4.9%	65.0%	68.0%	66.4%
Gaussian Noise	82.2%	89.8%	62.3%	81.3%	84.3%	81.8%
Boundary Attack	65.5%	67.9%	2.3%	64.4%	69.2%	67.9%
DeepFool	62.2%	67.3%	0.9%	64.4%	67.4%	65.7%
Pointwise Attack	80.4%	88.6%	46.2%	78.9%	83.8%	81.4%
DDN	60.0%	63.5%	0.1%	64.5%	67.7%	66.2%
CWL2	62.0%	71.6%	0.1%	66.9%	71.5%	68.7%
ℓ_2 attacks ($\epsilon = 0.05$)	57.3%	61.6%	0.0%	61.7%	65.0%	64.3%
PGD-ℓ_1	16.5%	49.2%	69.1%	39.5%	54.0%	53.4%
Salt & Pepper	63.4%	74.2%	35.5%	75.2%	80.7%	75.6%
Pointwise Attack	49.6%	62.4%	8.4%	63.3%	77.0%	72.8%
ℓ_1 attacks ($\epsilon = 12$)	16.0%	46.6%	7.9%	39.4%	54.0%	53.4%
All attacks	15.6%	27.5%	0.0%	34.9%	40.6%	**46.1**%

C.2 CIFAR10 RESULTS

Expanded table of results Table 4 contains the full table of results for all attacks on all models on the CIFAR10 dataset. All attacks were run on a subset of the first 1000 test examples with 10 random restarts, with the exception of Boundary Attack, which by default makes 25 trials per iteration, and DDN attack, which does not benefit from restarts owing to a deterministic starting point. Further note that salt & pepper and pointwise attacks in the ℓ_1 section are technically ℓ_0 attacks, but produce perturbations in the ℓ_1 ball. Finally, it is clear here that while the training against an ℓ_1 PGD adversary defends against said PGD adversary, it does not seem to transfer to robustness against other attacks.

Robustness curves Here, we plot the full robustness curves for the remaining models trained with the simpler generalizations of adversarial training, namely the worst case method (Figure 6) and the data augmentation method (Figure 7).

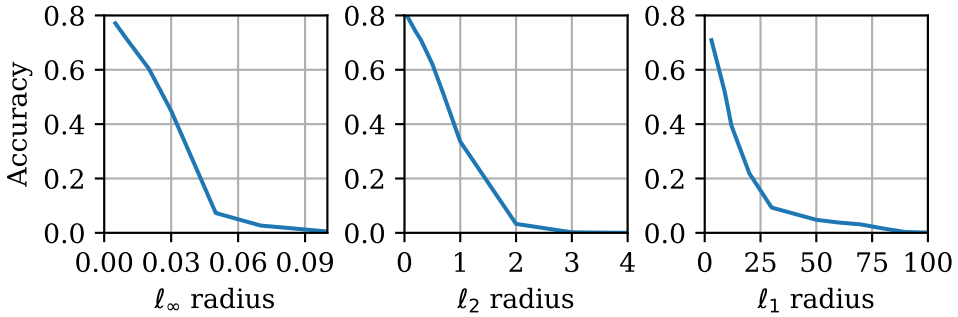

Figure 6: Robustness curves showing the adversarial accuracy for the CIFAR10 model trained with the worst case generalization for adversarial training (Worst-PGD) against ℓ_∞ (left), ℓ_2 (middle), and ℓ_1 (right) threat models over a range of epsilon.

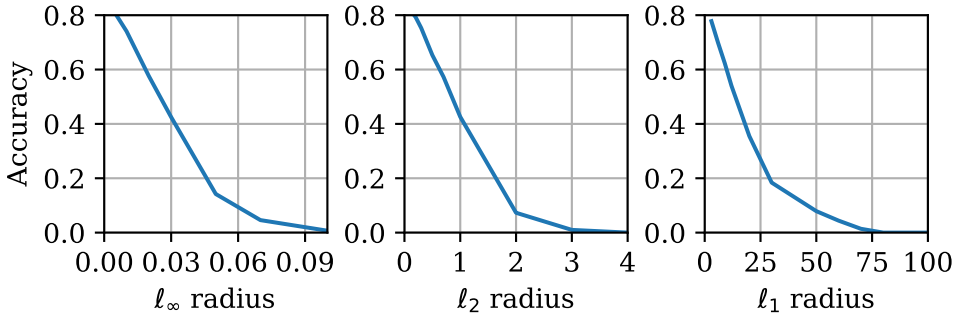

Figure 7: Robustness curves showing the adversarial accuracy for the CIFAR10 model trained with the data augmentation generalization for adversarial training (PGD-Aug) against ℓ_∞ (left), ℓ_2 (middle), and ℓ_1 (right) threat models over a range of epsilon.

D COMPARISON WITH CONCURRENT WORK

In this section we compare the results of our trained MSD model with that of Tramèr & Boneh (2019), who study the theoretical and empirical trade-offs of adversarial robustness in various settings when defending against multiple adversaries. Training methods presented by them in their comparisons, namely Adv_{avg} and Adv_{max} closely resemble the naive approaches discussed in this paper: PGD-Aug and Worst-PGD respectively. We use the results as is from their work, and additionally compare the position of our MSD models at the revised thresholds used by Tramèr & Boneh (2019) without specially retraining them.

The results of Tables 5 and 6 show that the relative advantage of MSD over naive techniques does hold up. While we do make a comparison to the most relevant concurrent work for completeness, the following differences can bias the robust accuracies reported for the MSD models to relatively lower than expected (and correspondingly, the robust accuracies reported for the other models are relatively higher than expected):

1. **Use of random restarts:** We observe in our experiments that using up to 10 restarts for all our attacks leads to a decrease in model accuracy from 5 to 10% across all models. Tramèr & Boneh do not mention restarting their attacks for these models and so the results for models apart from MSD in Tables 5, 6 could potentially be lowered with random restarts.

2. **Different training and testing thresholds:** The MSD model for the MNIST dataset was trained at ϵ = (0.3, 1.5, 12) for the ℓ_∞, ℓ_2, ℓ_1 perturbation balls respectively, while Tramèr & Boneh (2019) tested at ϵ = (0.3, 2.0, 10). This may lower the robust accuracy at these thresholds for the MSD model, since it was not trained for that particular threshold. Likewise, the MSD model for CIFAR10 was also trained at ϵ = (0.03, 0.05, 12) for the ℓ_∞, ℓ_2, ℓ_1 perturbation balls respectively, while Tramèr & Boneh (2019) tested at ϵ = ($\frac{4}{255}$, 0, $\frac{2000}{255}$).

Table 5: Comparison with contemporary work on MNIST (higher is better). Results for all models except MSD are taken as is from Tramèr & Boneh (2019)

	Vanilla	Adv_∞	Adv_1	Adv_2	Adv_{avg}	Adv_{max}	**MSD**
Clean accuracy	99.4%	99.1%	98.9%	98.5%	97.3%	97.2%	98.2%
ℓ_∞ attacks ($\epsilon = 0.3$)	0.0%	91.1%	0.0%	0.4%	76.7%	71.7%	63.7%
ℓ_2 attacks ($\epsilon = 2.0$)	12.4%	12.1%	50.6%	71.8%	58.3%	56.0%	67.4%
ℓ_1 attacks ($\epsilon = 10$)	8.5%	11.3%	78.5%	68.0%	53.9%	62.6%	70.0%
All attacks	0.0%	6.8%	0.0%	0.4%	49.9%	52.4%	**60.9**%

Table 6: Comparison with contemporary work on CIFAR10 (higher is better). Results for all models except MSD are taken as is from Tramèr & Boneh (2019)

	Vanilla	Adv_∞	Adv_1	Adv_{avg}	Adv_{max}	**MSD**
Clean accuracy	95.7%	92.0%	90.8%	91.1%	91.2%	82.1%
ℓ_∞ attacks ($\epsilon = \frac{4}{255}$)	0.0%	71.0%	53.4%	64.1%	65.7%	65.6%
ℓ_1 attacks ($\epsilon = \frac{2000}{255}$)	0.0%	16.4%	66.2%	60.8%	62.5%	62.0%
All attacks	0.0%	16.4%	53.1%	59.4%	61.1%	**61.7**%

3. **Different perturbation models:** For the CIFAR10 results in Table 6, Adv_{avg} & Adv_{max} models are trained and tested only for ℓ_1 and ℓ_∞ adversarial perturbations, whereas the MSD model is robust to the union of ℓ_1, ℓ_2 and ℓ_∞, achieving a much harder task.
4. **Larger Suite of Attacks Used:** The attacks used by Tramèr & Boneh are PGD, EAD (Chen et al., 2017) and Pointwise Attack (Schott et al., 2019) for ℓ_1; PGD, C&W (Carlini & Wagner, 2017) and Boundary Attack (Brendel et al., 2017) for ℓ_2; and PGD for ℓ_∞ adversaries. We use a more expansive suite of attacks as shown in Appendix C. Some of the attacks like DDN, which proved to be strong adversaries in most cases, were not considered by Tramèr & Boneh (2019) and thus were only used to attack the MSD models in Tables 5 and 6.