Interpolated Adversarial Training: Achieving Robust Neural Networks Without Sacrificing Too Much Accuracy

Alex Lamb*
Montréal Institute for Learning Algorithms
lambalex@iro.umontreal.ca

Vikas Verma*
Aalto University, Finland
Montréal Institute for Learning Algorithms
vikasverma.iitm@gmail.com

Juho Kannala
Aalto University, Finland
juho.kannala@aalto.fi

Yoshua Bengio
Montréal Institute for Learning Algorithms
CIFAR Senior Fellow
yoshua.umontreal@gmail.com

ABSTRACT

Adversarial robustness has become a central goal in deep learning, both in theory and in practice. However, successful methods to improve the adversarial robustness (such as adversarial training) greatly hurt generalization performance on the unperturbed data. This could have a major impact on how achieving adversarial robustness affects real world systems (i.e. many may opt to forego robustness if it can improve accuracy on the unperturbed data). We propose Interpolated Adversarial Training, which employs recently proposed interpolation based training methods in the framework of adversarial training. On CIFAR-10, adversarial training increases the standard test error (when there is no adversary) from 4.43% to 12.32%, whereas with our Interpolated adversarial training we retain adversarial robustness while achieving a standard test error of only 6.45%. With our technique, the relative increase in the standard error for the robust model is reduced from 178.1% to just 45.5%.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Regularization*.

KEYWORDS

Neural Networks; Adversarial Robustness; Standard Test Error; Mixup; Manifold Mixup; Interpolation Based Training

*Both authors contributed equally to this research.

1 INTRODUCTION

Deep neural networks have been highly successful across a variety of tasks. This success has driven applications in areas where reliability and security are critical, including face recognition [20], self-driving cars [6], health care, and malware detection [15]. Security concerns emerge when adversaries of the system stand to benefit from a system performing poorly. Work on *Adversarial examples* [22] has shown that neural networks are vulnerable to attacks perturbing the data in imperceptible ways. Many defenses have been proposed, but most of them rely on *obfuscated gradients* [1] to give a false illusion of defense by lowering the quality of the gradient signal, without actually improving robustness [1]. Of these defenses, only adversarial training [14] was found to still be effective after addressing the problem of obfuscated gradients.

However, adversarial training has a major disadvantage: it drastically reduces the generalization performance of networks on unperturbed data samples, especially for small networks. For example, [16] reports that adding adversarial training to a specific model increases the standard test error from 6.3% to 21.6% on CIFAR-10. This phenomenon makes adversarial training difficult to use in practice. If the tension between the performance and the security turns out to be irreconcilable, then many systems would either need to perform poorly or accept vulnerability.

Our contribution: We propose to augment adversarial training with interpolation based training, as a solution to the above problem.

- We demonstrate that our approach substantially improves standard test error while still achieving adversarial robustness, using benchmark datasets (CIFAR10 and SHVN) and benchmark architectures (Wide-ResNet and ResNet): Section 5.1.
- We demonstrate that our approach does not suffer from the *obfuscated gradient* problem by performing black-box attacks on the models trained with our approach: Section 5.2.
- We perform PGD attack of higher number of steps (up to 1000 steps) and higher value of maximum allowed perturbation/distortion *epsilon*, to demonstrate that the adversarial robustness of our approach remains at the same level as that of the adversarial training : Section 5.3.

- We demonstrate that the networks trained with our approach have lower complexity, resulting in improved standard test error : Section 5.4.

2 RELATED WORK

The trade-off between standard test error and adversarial robustness has been studied in [16, 18, 24, 33]. While [16, 24, 33] empirically demonstrate this trade-off, [24, 33] demonstrate this trade-off theoretically as well on the constructed learning problems. Furthermore, [18] study this trade-off from the point-of-view of the statistical properties of the *robust objective* [4] and the dynamics of optimizing a robust objective on a neural network, and suggest that adversarial training requires more data to obtain a lower standard test error. Our results on SVHN and CIFAR-10 (Section 5.1) also consistently show higher standard test error with PGD adversarial training.

While [24] presented data dependent proofs showing that on certain artificially constructed distributions - it is impossible for a robust classifier to generalize as good as a non-robust classifier. How this relates to our results is an intriguing question. Our results suggest that the generalization gap between adversarial training and non-robust models can be substantially reduced through better algorithms, but it remains possible that closing this gap entirely on some datasets is impossible. An important question for future work is how much this generalization gap can be explained in terms of inherent data properties and how much this gap can be addressed through better models.

Neural Architecture Search [34] was used to find architectures which achieve high robustness to PGD attacks as well as better test error on the unperturbed data [8]. This improved test error on the unperturbed data and a direct comparison to our method is in Table 1. However, the method of [8] is computationally very expensive as each experiment requires training thousands of models to search for optimal architectures (9360 child models each trained for 10 epochs in [8]), whereas our method involves no significant additional computation.

In our work we primarily concern ourselves with adversarial training, but techniques in the research area of the provable defenses have also shown a trade-off between robustness and generalization on unperturbed data. For example, the dual network defense of [12] reported 20.38% standard test error on SVHN for their provably robust convolutional network (most non-robust models are well under 5% test error on SVHN). [27] reported a best standard test *accuracy* of 29.23% using a convolutional *ResNet* on CIFAR-10 (most non-robust ResNets have accuracy of well over 90%). Our goal here is not to criticize this work, as developing provable defenses is a challenging and important area of work, but rather to show that this problem that we explore with *Interpolated Adversarial Training* (on adversarial training [16] type defenses) is just as severe with provable defenses, and understanding if the insights developed here carry over to provable defenses, could be an interesting area for future work.

Adversarially robust generalization: Another line of research concerns *adversarially robust generalization*: the performance of adversarially trained networks on adversarial test examples. [19]

observe that a higher sample complexity is needed for better adversarially robust generalization. [28] demonstrate that adversarial training results in higher complexity models and hence poorer adversarially robust generalization. Furthermore, [19] suggest that adversarially robust generalization requires more data and [7, 30] demonstrate that unlabeled data can be used to improve adversarially robust generalization. In contrast to their work, in this work we focus on improving the generalization performance on unperturbed samples (standard test error), while maintaining robustness on unseen adversarial examples at the same level.

Interpolation based training techniques: Yet another line of research [5, 25, 26, 32] shows that simple interpolation based training techniques are able to substantially decrease standard test error in fully-supervised and semi-supervised learning paradigms. Along these lines, [31] studies the theoretical properties of interpolation based training techniques such as Mixup [32]

3 BACKGROUND

3.1 The Empirical Risk Minimization Framework

Let us consider a general classification task with an underlying data distribution $\mathcal{D}$ which consists of examples $x \in \mathbf{X}$ and corresponding labels $y \in \mathbf{Y}$. The task is to learn a function $f : \mathbf{X} \rightarrow \mathbf{Y}$ such that for a given x, f outputs corresponding y. It can be done by minimizing the risk $\mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathcal{L}(x, y, \theta)]$, where $\mathcal{L}(\theta, x, y)$ is a suitable loss function for instance the cross-entropy loss and $\theta \in \mathbb{R}^p$ is the set of parameters of function f. Since this expectation cannot be computed, therefore a common approach is to to minimize the empirical risk $1/N \sum_{i=1}^{N} \mathcal{L}(x_i, y_i, \theta)$ taking into account only a finite number of examples drawn from the data distribution $\mathcal{D}$, namely $(x_1, y_1), \ldots, (x_N, y_N)$.

3.2 Adversarial Attacks and Robustness

While the empirical risk minimization framework has been very successful and often leads to excellent generalization on the unperturbed test examples, it has the significant limitation that it doesn't guarantee good performance on examples which are carefully perturbed to fool the model. [10, 22]. That is, the empirical risk minimization framework suffers from a lack of robustness to adversarial attacks.

Madry et al. [16] proposed an optimization view of adversarial robustness, in which the adversarial robustness of a model is defined as a min-max problem. Using this view, the parameters θ of a function f are learned by minimizing $\rho(\theta)$ as described in Equation 1. $\mathcal{S}$ defines a region of points around each example, which is typically selected so that it only contains visually imperceptible perturbations.

$$\min_{\theta} \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta \in \mathcal{S}} \mathcal{L}(\theta, x + \delta, y) \right] \quad (1)$$

Adversarial attacks can be broadly categorized into two categories: Single-step attacks and Multi-step attacks. We evaluated the performance of our model as a defense against the most popular and well-studied adversarial attack from each of these categories. Firstly, we consider the Fast Gradient Sign Method [FGSM, 10]

which is a single step and can still be effective against many networks. Secondly, we consider the projected gradient descent attack [14] which is a multi-step attack. It is slower than FGSM as it requires many iterations, but has been shown to be a much stronger attack [16]. We briefly describe these two attacks as follows:

Fast Gradient Sign Method (FGSM). The Fast Gradient Sign Method [10] produces ℓ_∞ bounded adversaries by the following the sign of the gradient based perturbation. This attack is cheap since it only relies on computing the gradient once and is often an effective attack against deep networks [10, 16], especially when no adversarial defenses are employed.

$$\widetilde{x} = x + \varepsilon \, \text{sgn}(\nabla_x \mathcal{L}(\theta, x, y)). \tag{2}$$

Projected Gradient Descent (PGD). The projected gradient descent attack [16], sometimes referred to as FGSM^k, is a multi-step extension of the FGSM attack characterized as follows:

$$x^{t+1} = \Pi_{x+\mathcal{S}} \left(x^t + \alpha \, \text{sgn}(\nabla_x \mathcal{L}(\theta, x, y)) \right). \tag{3}$$

initialized with x^0 as the clean input x. $\mathcal{S}$ formalizes the manipulative power of the adversary. Π refers to the projection operator, which in this context means projecting the adversarial example back onto the region within an $\mathcal{S}$ radius of the original data point, after each step of size α in the adversarial attack.

3.3 Gradient Obfuscation by Adversarial Defenses

Many approaches have been proposed as a defense against adversarial attacks. A significant challenge with evaluating defenses against adversarial attacks is that many attacks rely upon a network's gradient. The defense methods which reduce the quality of this gradient, either by making it flatter or noisier can lead to methods which lower the effectiveness of gradient-based attacks, but which are not actually robust to adversarial examples [2, 17]. This process, which has been referred to as gradient masking or gradient obfuscation, must be analyzed when studying the strength of an adversarial defense.

One method for examining the extent to which an adversarial defense gives deceptively good results as a result of gradient obfuscation relies on the observation that black-box attacks are a strict subset of white-box attacks, so white-box attacks should always be at least as strong as black-box attacks. If a method reports much better defense against white-box attacks than the black-box attack, it suggests that the selected white-box attack is underpowered as a result of the gradient obfuscation. Another test for gradient obfuscation is to run an iterative search, such as projected gradient descent (PGD) with an unlimited range for a large number of iterations. If such an attack is not completely successful, it indicates that the model's gradients are not an effective method for searching for adversarial images, and that gradient obfuscation is occurring. We demonstrate successful results with *Interpolated Adversarial Training* on these sanity checks in Section 5.2. Still another test is to confirm that iterative attacks with small step sizes always outperform single-step attacks with larger step sizes (such as FGSM). If this is not the case, it may suggest that the iterative attack becomes stuck in regions where optimization using gradients is poor due to gradient masking. In all of our experiments for *Interpolated Adversarial Training* , we found that the iterative PGD attacks with

smaller step sizes and more iterations were always stronger than the FGSM attacks (which take a single large step) against our models, as shown in Table 1, Table 2, Table 3,and 4.

3.4 Adversarial Training

Adversarial training [10] encompasses crafting adversarial examples and using them during training to increase robustness against unseen adversarial examples. To scale adversarial training to large datasets and large models, often the adversarial examples are crafted using the fast single step methods such as FGSM. However, adversarial training with fast single step methods remains vulnerable to adversarial attacks from a stronger multi-step attack such as PGD. Thus, in this work, we consider adversarial training with PGD.

4 INTERPOLATED ADVERSARIAL TRAINING

We propose *Interpolated Adversarial Training* (IAT), which trains on interpolations of adversarial examples along with interpolations of unperturbed examples. We use the techniques of Mixup [32] and Manifold Mixup[25] as ways of interpolating examples. Learning is performed in the following four steps when training a network with *Interpolated Adversarial Training* . In the first step, we compute the loss from a unperturbed (non-adversarial) batch (with interpolations based on either Mixup or Manifold Mixup). In the second step, we generate a batch of adversarial examples using an adversarial attack (such as Projected Gradient Descent (PGD) [16] or Fast Gradient Sign Method (FGSM) [10]). In the third step, we train against these adversarial examples with the original labels, with interpolations based on either Mixup or Manifold Mixup. In the fourth step, we obtain the average of the loss from the unperturbed batch and the adversarial batch and update the network parameters using this loss. Note that following [13, 24], we use both the unperturbed and adversarial samples to train the model *Interpolated Adversarial Training* and we use it in our baseline adversarial training models as well. The detailed algorithm is described in Algorithm Block 1.

As *Interpolated Adversarial Training* combines adversarial training with either Mixup [32] or Manifold Mixup [25], we summarize these supporting methods in more detail. The Mixup method [32] consists of drawing a pair of samples from the dataset $(x_i, y_i) \sim p_D$ and $(x_j, y_j) \sim p_D$ and then taking a random linear interpolation in the input space $\tilde{x} = \lambda x_i + (1 - \lambda)x_j$. This λ is sampled randomly on each update (typically from a Beta distribution). Then the network f_θ is run forward on the interpolated input $\tilde{x}$ and trained using the same linear interpolation of the losses $\mathcal{L} = \lambda L(f_\theta(\tilde{x}), y_i) + (1 - \lambda)L(f_\theta(\tilde{x}), y_j)$. Here L refers to a loss function such as cross entropy.

The Manifold Mixup method [25] is closely related to Mixup from a computational perspective, except that the layer at which interpolation is performed, is selected randomly on each training update.

Adversarial training consists of generating adversarial examples and training the model to give these points the original label. For generating these adversarial examples during training, we used the Projected Gradient Descent (PGD) attack, which is also known as iterative FGSM. This attack consists of repeatedly updating an adversarial perturbation by moving in the direction of the sign of the gradient multiplied by some step size, while projecting back to

an L_∞ ball by clipping the perturbation to maximum ϵ. Both ϵ, the step size to move on each iteration, and the number of iterations are hyperparameters for the attack.

Why *Interpolated Adversarial Training* helps to improve the standard test accuracy: We present two arguments for why *Interpolated Adversarial Training* can improve standard test accuracy:

Increasing the training set size: [18] has shown that adversarial training could require more training samples to attain a higher standard test accuracy. Mixup [32] and Manifold Mixup [25] can be seen as the techniques that increase the effective size of the training set by creating novel training samples. Hence these techniques can be useful in improving standard test accuracy.

Information compression: [21, 23] have shown a relationship between compression of information in the features learned by deep networks and generalization. This relates the degree to which deep networks compress the information in their hidden states to bounds on generalization, with a stronger bound when the deep networks have stronger compression.

To evaluate the effect of adversarial training on compression of the information in the features, we performed an experiment where we take the representations learned after training, and study how well these frozen representations are able to successfully predict fixed random labels. If the model compresses the representations well, then it will be harder to fit random labels. In particular, we ran a small 2-layer MLP on top of the learned representations to fit random binary labels. In all cases we trained the model with the random labels for 200 epochs with the same hyperparameters. For fitting 10000 randomly labeled examples, we achieved accuracy of: 92.08% (Baseline) and 97.00% (PGD Adversarial Training): showing that adversarial training made the representations much less compressed.

Manifold Mixup [25] has shown to learn more compressed features. Hence, employing Manifold Mixup with the adversarial training might mitigate the adverse effect of the adversarial training. Using the same experimental setup as above, we achieved accuracy of : 64.17% (Manifold Mixup) and 71.00% (IAT using Manifold Mixup).

These results suggest that adversarial training causes the learned representations to be less compressed which may be the reason for poor standard test accuracy. At the same time, IAT with Manifold Mixup significantly reduces the ability of the model to learn less compressed features, which may potentially improve standard test accuracy.

5 EXPERIMENTS

5.1 Adversarial Robustness

The goal of our experiments is to provide empirical support for our two major assertions: that adversarial training hurts performance on unperturbed data (which is consistent with what has been previously observed [16, 24, 33]) and to show that this difference can be reduced with our *Interpolated Adversarial Training* method. Finally, we want to show that *Interpolated Adversarial Training* is adversarially robust and does not suffer from gradient obfuscation [1].

In our experiments we always perform adversarial training using a 7-step PGD attack but we evaluate on a variety of attacks: FGSM, PGD (with a varying number of steps and hyperparameters).

Architecture and Datasets: We conducted experiments on competitive networks to demonstrate that *Interpolated Adversarial Training* can improve generalization performance without sacrificing adversarial robustness. We used two architectures : First, the WideResNet architecture proposed in [11, 29] and used in [16] for adversarial training [1]. Second, the PreActResnet18 architecture which is a variant of the residual architecture of [11]. We used SGD with momentum optimizer in our experiments. We ran the experiments for 200 epochs with initial learning rate is 0.1 and it is annealed by a factor of 0.1 at epoch 100 and 150. We use the batch-size of 64 for all the experiments.

We used two benchmark datasets (CIFAR10 and SVHN), which are commonly used in the adversarial robustness literature [16]. The CIFAR-10 dataset consists of 60000 color images each of size 32×32, split between 50K training and 10K test images. This dataset has ten classes, which include pictures of cars, horses, airplanes and deer. The SVHN dataset consists of 73257 training samples and 26032 test samples each of size 32×32. Each example is a close-up image of a house number (the ten classes are the digits from 0-9)

Data Pre-Processing and Hyperparameters: The data augmentation and pre-processing is exactly the same as in [16]. Namely, we use random cropping and horizontal flip for CIFAR10. For SVHN, we use random cropping. We use the per-image standardization for pre-processing. For adversarial training, we generated the adversarial examples using a PGD adversary using a ℓ_∞ projected gradient descent with 7 steps of size 2, and $\epsilon = 8$. For the adversarial attack, we used an FGSM adversary with $\epsilon = 8$ and a PGD adversary with 7 steps and 20 steps of size 2 and $\epsilon = 8$.

In the *Interpolated Adversarial Training* experiments, for generating the adversarial examples, we used PGD with the same hyperparameters as described above. For performing interpolation, we used either Manifold Mixup with $\alpha = 2.0$ as suggested in [25] or Mixup with $alpha = 1.0$ as suggested in [32]. For Manifold Mixup, we performed the interpolation at a randomly chosen layer from the input layer, the output of the first resblock or the output of the second resblock, as recommended in [25].

Results: The results are presented in Table 1, Table 2, Table 3 and Table 4. We observe that IAT consistently improves standard test error relative to models using just adversarial training, while maintaining adversarial robustness at the same level. For example, in Table 1, we observe that the baseline model (no adversarial training) has standard test error of 4.43% whereas PGD adversarial increase the standard test error to 12.32%: a relative increase of 178% in standard test error. With *Interpolated Adversarial Training* , the standard test error is reduced to 6.45%, a relative increase of only 45% in standard test error as compared to the baseline, while the

---

[1]While [16] use WRN32-10 architecture, we use the standard WRN28-10 architecture, so our results are not directly comparable to their results.

[2]Since the objective of this work is to demonstrate the effectiveness the *Interpolated Adversarial Training* over adversarial training for improving the standard test error as well as maintaining the adversarial robustness to the same levels, we highlight the best results in the lower part of the Table: the methods in the upper part of the Table have better standard test error ("No-attack" column), but their adversarial robustness is very poor against strong adversarial attacks (PGD, 7 steps and 20 steps)

Algorithm 1 The Interpolated Adversarial Training Algorithm

Require: f_θ: Neural Network
Require: Mix: A way of combining examples (Mixup or Manifold Mixup [25, 32])
Require: D: Data samples
Require: N: Total number of updates
Require: $Loss$: A function which runs the neural network with Mix applied
  for $k = 1, \ldots, N$ **do**
    Sample $(x_i, y_i) \sim D$ ▷ Sample batch
    $\mathcal{L}_c = Loss(f_\theta, Mix, x_i, y_i)$ ▷ Compute loss on unperturbed data using Mixup (or Manifold Mixup)
    $\tilde{x}_i = attack(x_i, y_i)$ ▷ Run attack (e.g. PGD [16])
    $\mathcal{L}_a = Loss(f_\theta, Mix, \tilde{x}_i, y_i)$ ▷ Compute adversarial loss on adversarial samples using Mixup (or Manifold Mixup)
    $\mathcal{L} = (\mathcal{L}_c + \mathcal{L}_a)/2$ ▷ Combined loss
    $g \leftarrow \nabla_\theta \mathcal{L}$ ▷ Gradients of the combined Loss
    $\theta \leftarrow Step(\theta, g_\theta)$ ▷ Update parameters using gradients g (e.g. SGD)
  end for

Table 1: CIFAR10 results (error in %) to white-box attacks on WideResNet20-10 evaluated on the test data. The rows correspond to the training mechanism and columns correspond to adversarial attack methods. The upper part of the Table consists of training mechanisms that do not employ any explicit adversarial defense. The lower part of the Table consist of methods that employ adversarial training as a defense mechanism [2]. For PGD, we used a ℓ_∞ projected gradient descent with size $\alpha = 2$, and $\epsilon = 8$. For FGSM, we used $\epsilon = 8$. Our method of *Interpolated Adversarial Training* improves standard test error in comparison to adversarial training (refer to the first column) and maintains the adversarial robustness on the same level as that of adversarial training. The method of [8] is close to our method in terms of standard test error and adversarial robustness however it needs several orders of magnitude more computation (it trains 9360 models) for its neural architecture search.

Training Adversary	No Attack	FGSM	PGD (7 steps)	PGD (20 steps)
Baseline [16]	4.80	67.3	95.9	96.5
Baseline	4.43±0.09	56.92±0.79	99.83±0.02	100.0±0.0
Mixup	3.25±0.11	32.63±0.88	92.75±0.61	99.27±0.03
Manifold Mixup	3.15±0.09	38.41±2.64	89.77±3.68	98.34±1.03
Neural Architecture Search [8]	6.80	36.4	49.9	-
PGD (7 steps) [16]	12.70	43.90	50.00	54.20
PGD (7 steps) (our code)	12.32±0.14	41.87±0.04	50.97±0.15	**54.87±0.16**
Interpolated Adversarial Training (with Mixup)	**6.45±0.52**	**33.83±0.86**	**49.88±0.55**	54.89±1.37
Interpolated Adversarial Training (Manifold Mixup)	6.48±0.30	35.18±0.30	50.08±0.48	55.18±0.18

degree of adversarial robustness remains approximately unchanged, across varies type of adversarial attacks.

5.2 Transfer Attacks

As a sanity check that *Interpolated Adversarial Training* does not suffer from gradient obfuscation [1], we performed a transfer attack evaluation on the CIFAR-10 dataset using the PreActResNet18 architecture. In this type of evaluation, the model which is used to generate the adversarial examples is different from the model used to evaluate the attack. As these transfer attacks do not use the target model's parameters to compute the adversarial example, they are considered black-box attacks. In our evaluation (Table 5) we found that black-box transfer were always substantially weaker than white-box attacks, hence *Interpolated Adversarial Training* does not suffer from gradient obfuscation [1]. Additionally, in Table 6, we observe that increasing ϵ results in 100% success of attack, providing added evidence that *Interpolated Adversarial Training* does not suffer from gradient obfuscation [1].

Table 2: CIFAR10 results (error in %) to white-box attacks on PreActResnet18. Rest of the details are same as Table 1

Training \ Adversary	No Attack	FGSM	PGD (7 steps)	PGD (20 steps)
Baseline	5.88±0.16	78.11±1.31	99.85±0.18	100.0 ±0.0
Mixup	4.42±0.03	38.32±0.76	97.48±0.15	99.88±0.02
Manifold Mixup	4.10±0.09	37.57±1.31	88.50±3.20	97.80±1.02
PGD (7 steps)	14.12±0.06	48.56±0.14	57.76±0.19	**61.00±0.24**
Interpolated Adversarial Training (with Mixup)	**10.12±0.33**	**40.71±0.65**	**55.43±0.45**	61.62±1.01
Interpolated Adversarial Training (Manifold Mixup)	10.30±0.15	42.48±0.29	55.78±0.67	61.80±0.51

Table 3: SVHN results (error in %) to white-box attacks on WideResNet20-10 using the 26032 test examples. The rows correspond to the training mechanism and columns correspond to adversarial attack methods. For PGD, we used a ℓ_∞ projected gradient descent with step-size $\alpha = 2$, and $\epsilon = 8$. For FGSM, we used $\epsilon = 8$. Our method of *Interpolated Adversarial Training* improves standard test error and adversarial robustness.

Training \ Adversary	No Attack	FGSM	PGD (7 steps)	PGD (20 steps)
Baseline	3.07±0.03	39.36±1.16	94.00±0.65	98.59±0.13
Mixup	2.59±0.08	26.93±1.96	90.18±3.43	98.78±0.79
Manifold Mixup	2.46±0.01	29.74±0.99	77.49±3.82	94.77±1.34
PGD (7 steps)	6.14±0.13	29.10±0.72	46.97±0.49	53.47±0.52
Interpolated Adversarial Training (with Mixup)	3.47±0.11	**22.08±0.15**	45.74±0.11	58.40±0.46
Interpolated Adversarial Training (Manifold Mixup)	**3.38±0.22**	22.30±1.07	**42.61±0.40**	**52.79±0.22**

Table 4: SVHN results (error in %) to white-box attacks on PreActResnet18. Rest of the details are same as Table 3.

Training \ Adversary	No Attack	FGSM	PGD (7 steps)	PGD (20 steps)
Baseline	3.47±0.09	50.73±0.22	96.37±0.12	98.61±0.06
Mixup	2.91±0.06	31.91±0.59	98.43±0.85	99.95±0.02
Manifold Mixup	2.66±0.02	29.86±3.60	72.47±1.82	94.00±0.96
PGD (7 steps)	5.27±0.13	26.78±0.62	47.00±0.22	54.40±0.42
Interpolated Adversarial Training (with Mixup)	3.63±0.05	**23.57±0.64**	47.69±0.22	54.62±0.18
Interpolated Adversarial Training (Manifold Mixup)	**3.61±0.22**	24.95±0.92	**46.62±0.28**	**54.13±1.08**

5.3 Varying the number of iterations and ϵ for Iterative Attacks

To further study the robustness of *Interpolated Adversarial Training*, we studied the effect of changing the number of attack iterations and the range of the adversarial attack ϵ. Some adversarial defenses [9] have been found to have increasing vulnerability when exposed to attacks with a large number of iterations. We studied this (Table 7) and found that both adversarial training and *Interpolated Adversarial Training* have robustness which declines only slightly with an increasing number of steps, with almost no difference between the 100 step attack and the 1000 step attack. Additionally we varied the ϵ to study if *Interpolated Adversarial Training* was more or less vulnerable to attacks with ϵ different from what the model was trained on. We found that *Interpolated Adversarial Training* is somewhat more robust when using smaller ϵ and slightly less robust when using larger ϵ (Table 6).

Table 5: Transfer Attack evaluation of *Interpolated Adversarial Training* on CIFAR-10 reported in terms of error rate (%). Here we consider three trained models, using normal adversarial training (Adv), IAT with mixup (IAT-M), and IAT with manifold mixup (IAT-MM). On each experiment, we generate adversarial examples only using the model listed in the column and then evaluate these adversarial examples on the target model listed in the row. Note that in all of our experiments white box attacks (where the attacking model and target models are the same) led to stronger attacks than black box attacks, which is the evidence that our approach does not suffer from gradient obfuscation [1].

ϵ	2			5			10		
Attack Target	Adv. Train	IAT M	IAT MM	Adv. Train	IAT M	IAT MM	Adv. Train	IAT M	IAT MM
Adv. Train	28.54	21.11	21.87	43.68	28.10	29.21	74.66	44.39	48.14
IAT-M	17.14	25.57	18.07	25.02	45.03	28.85	48.74	78.49	51.35
IAT-MM	18.57	18.74	25.71	26.84	26.7	43.23	50.43	48.11	77.05

Table 6: Robustness on CIFAR-10 PreActResNet18 (Error %) with increasing ϵ and a fixed number of iterations (20). *Interpolated Adversarial Training* and adversarial training both have similar degradation in robustness with increasing ϵ, but *Interpolated Adversarial Training* tends to be slightly better for smaller ϵ and adversarial training is slightly better for larger ϵ

Attack ϵ Model	1	2	10	15	20	25	50
Adversarial Training	21.44	28.54	74.66	92.43	98.53	99.77	100.0
IAT (Mixup)	17.90	25.57	78.49	93.73	98.54	99.72	100.0
IAT (Manifold Mixup)	18.24	25.71	77.05	93.31	98.67	99.85	100.0

Table 7: Robustness on CIFAR-10 PreActResNet-18 (Error %) with fixed $\epsilon = 5$ and a variable number of iterations used for the adversarial attack.

Num. Iterations Model	5	10	20	50	100	1000
Adversarial Training	42.35	43.44	43.68	43.76	43.80	43.83
IAT (Mixup)	41.29	44.23	45.03	45.31	45.42	45.56
IAT (Manifold Mixup)	40.74	42.72	43.23	43.43	43.51	43.60

5.4 Analysis of the complexity of the trained models

In this section, we present the analysis of the networks learned using *Interpolated Adversarial Training* . The spectral complexity measure proposed by [3] suggests that the complexity of a deep network is a function of the frobenius norm and spectral norm of its weight matrices and that this spectral complexity can be used to prove a generalization bound. We computed both of these norms on a small 6-layer fully-connected network with 512 hidden units trained on Fashion-MNIST (Figure 1). We found that Adversarial Training increases the weight matrices frobenius norms across all the layers and increases the spectral norm of the majority of the layers. This is preliminary evidence that *Interpolated Adversarial Training* learns lower complexity classifiers than normal adversarial training.

6 CONCLUSION

Robustness to adversarial examples is essential for ensuring that machine learning systems are secure and reliable. However the most effective defense, adversarial training, has the effect of harming performance on unperturbed data. This has both theoretical and practical significance. As adversarial perturbations are imperceptible (or barely perceptible) to humans and humans are able to generalize extremely well, it is surprising that adversarial training hurts performance on unperturbed test data. This degradation in generalization is critically urgent to practitioners whose systems are threatened by adversarial attacks. With current techniques those wishing to deploy machine learning systems need to consider a severe trade-off between performance on unperturbed data and robustness to adversarial examples, which may mean that security and reliability will suffer in important applications. Our work has addressed both of these issues. We proposed to address this by augmenting adversarial training with interpolation based training
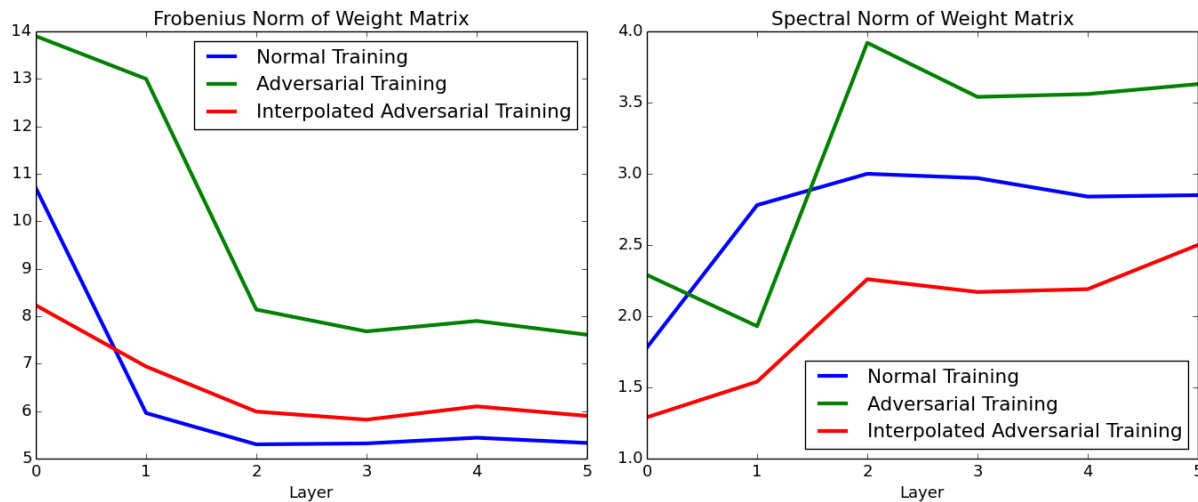
Figure 1: We analyzed the frobenius and spectral norms of the weight matrices on a 6-layer network. Generally Adversarial Training makes these norms larger, whereas *Interpolated Adversarial Training* brings these norms closer to their values when doing normal training.

[25, 32]. We found that this substantially improves generalization on unperturbed data while preserving adversarial robustness.

ACKNOWLEDGMENTS

REFERENCES

[1] A. Athalye, N. Carlini, and D. Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. *ArXiv e-prints* (Feb. 2018). arXiv:cs.LG/1802.00420

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2017. Synthesizing Robust Adversarial Examples. *CoRR* abs/1707.07397 (2017). arXiv:1707.07397 http://arxiv.org/abs/1707.07397

[3] Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. 2017. Spectrally-normalized margin bounds for neural networks. *CoRR* abs/1706.08498 (2017). arXiv:1706.08498 http://arxiv.org/abs/1706.08498

[4] A. Ben-Tal, L. El Ghaoui, and A.S. Nemirovski. 2009. *Robust Optimization.* Princeton University Press.

[5] David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. 2019. MixMatch: A Holistic Approach to Semi-Supervised Learning. *CoRR* abs/1905.02249 (2019). arXiv:1905.02249 http://arxiv.org/abs/1905.02249

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[7] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang, and John C. Duchi. 2019. Unlabeled Data Improves Adversarial Robustness. *arXiv e-prints*, Article arXiv:1905.13736 (May 2019), arXiv:1905.13736 pages. arXiv:stat.ML/1905.13736

[8] Ekin Dogus Cubuk, Barret Zoph, Samuel Stern Schoenholz, and Quoc V. Le. 2018. Intriguing Properties of Adversarial Examples. https://openreview.net/forum?id=rk6H0ZbRb

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

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy. 2014. Explaining and Harnessing Adversarial Examples. *ArXiv e-prints* (Dec. 2014). arXiv:stat.ML/1412.6572

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[12] J. Zico Kolter and Eric Wong. 2017. Provable defenses against adversarial examples via the convex outer adversarial polytope. *CoRR* abs/1711.00851 (2017). arXiv:1711.00851 http://arxiv.org/abs/1711.00851

[13] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *CoRR* abs/1607.02533 (2016). arXiv:1607.02533 http://arxiv.org/abs/1607.02533

[14] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial Machine Learning at Scale. *CoRR* abs/1611.01236 (2016). arXiv:1611.01236 http://arxiv.org/abs/1611.01236

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[16] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. *ArXiv e-prints* (June 2017). arXiv:stat.ML/1706.06083

[17] Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. 2016. Towards the Science of Security and Privacy in Machine Learning. *CoRR* abs/1611.03814 (2016). arXiv:1611.03814 http://arxiv.org/abs/1611.03814

[18] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. 2019. Adversarial Training Can Hurt Generalization. *arXiv e-prints*, Article arXiv:1906.06032 (Jun 2019), arXiv:1906.06032 pages. arXiv:cs.LG/1906.06032

[19] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. 2018. Adversarially Robust Generalization Requires More Data. *CoRR* abs/1804.11285 (2018). arXiv:1804.11285 http://arxiv.org/abs/1804.11285

[20] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. 2017. Adversarial Generative Nets: Neural Network Attacks on State-of-the-Art Face Recognition. *arXiv preprint arXiv:1801.00349* (2017).

[21] Ravid Shwartz-Ziv and Naftali Tishby. 2017. Opening the Black Box of Deep Neural Networks via Information. *CoRR* abs/1703.00810 (2017). arXiv:1703.00810 http://arxiv.org/abs/1703.00810

[22] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. 2013. Intriguing properties of neural networks. *ArXiv e-prints* (Dec. 2013). arXiv:cs.CV/1312.6199

[23] Naftali Tishby and Noga Zaslavsky. 2015. Deep Learning and the Information Bottleneck Principle. *CoRR* abs/1503.02406 (2015). arXiv:1503.02406 http://arxiv.org/abs/1503.02406

[24] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2018. Robustness May Be at Odds with Accuracy. *arXiv e-prints*, Article arXiv:1805.12152 (May 2018), arXiv:1805.12152 pages. arXiv:stat.ML/1805.12152

[25] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold Mixup: Better Representations by Interpolating Hidden States. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, Long Beach, California, USA, 6438–6447. http://proceedings.mlr.press/v97/verma19a.html

[26] Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. 2019. Interpolation Consistency Training for Semi-Supervised Learning. *arXiv e-prints*, Article arXiv:1903.03825 (Mar 2019), arXiv:1903.03825 pages. arXiv:stat.ML/1903.03825

[27] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. 2018. Scaling provable adversarial defenses. *CoRR* abs/1805.12514 (2018). arXiv:1805.12514 http://arxiv.org/abs/1805.12514

[28] Dong Yin, Kannan Ramchandran, and Peter Bartlett. 2018. Rademacher Complexity for Adversarially Robust Generalization. *CoRR* abs/1810.11914 (2018). arXiv:1810.11914 http://arxiv.org/abs/1810.11914

[29] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide Residual Networks. *CoRR* abs/1605.07146 (2016). arXiv:1605.07146 http://arxiv.org/abs/1605.07146

[30] Runtian Zhai, Tianle Cai, Di He, Chen Dan, Kun He, John Hopcroft, and Liwei Wang. 2019. Adversarially Robust Generalization Just Requires More Unlabeled Data. *arXiv e-prints*, Article arXiv:1906.00555 (Jun 2019), arXiv:1906.00555 pages. arXiv:cs.LG/1906.00555

[31] Chao Zhang, Min-Hsiu Hsieh, and Dacheng Tao. 2018. Generalization Bounds for Vicinal Risk Minimization Principle. *CoRR* abs/1811.04351 (2018). arXiv:1811.04351 http://arxiv.org/abs/1811.04351

[32] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2017. mixup: Beyond Empirical Risk Minimization. *CoRR* abs/1710.09412 (2017). arXiv:1710.09412 http://arxiv.org/abs/1710.09412

[33] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. 2019. Theoretically Principled Trade-off between Robustness and Accuracy. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, Long Beach, California, USA, 7472–7482. http://proceedings.mlr.press/v97/zhang19p.html

[34] Barret Zoph and Quoc V. Le. 2016. Neural Architecture Search with Reinforcement Learning. *CoRR* abs/1611.01578 (2016). arXiv:1611.01578 http://arxiv.org/abs/1611.01578