Lamb, Alex; Verma, Vikas; Kawaguchi, Kenji; Matyasko, Alexander; Khosla, Savya; Kannala, Juho; Bengio, Yoshua

Interpolated Adversarial Training: Achieving robust neural networks without sacrificing too much accuracy

Contents lists available at ScienceDirect
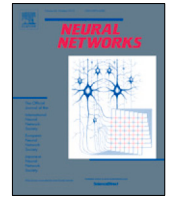
Neural Networks

Interpolated Adversarial Training: Achieving robust neural networks without sacrificing too much accuracy

Alex Lamb [a],[*],[1], Vikas Verma [b],[*],[1], Kenji Kawaguchi [c], Alexander Matyasko [d], Savya Khosla [e], Juho Kannala [b], Yoshua Bengio [a]

[a] Montreal Institute for Learning Algorithms (MILA), Canada
[b] Aalto University, Finland
[c] Harvard University, USA
[d] National University of Singapore, Singapore
[e] Google, India

ABSTRACT

Adversarial robustness has become a central goal in deep learning, both in the theory and the 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 the 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 the 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%. Moreover, we provide mathematical analysis of Interpolated Adversarial Training to confirm its efficiencies and demonstrate its advantages in terms of robustness and generalization.

1. Introduction

Deep neural networks have been highly successful across a variety of tasks. This success has driven applications in the areas where reliability and security are critical, including face recognition (Sharif, Bhagavatula, Bauer, & Reiter, 2017), self-driving cars (Bojarski et al., 2016), health care, and malware detection (LeCun, Bengio, & Hinton, 2015). Security concerns emerge when adversaries of the system stand to benefit from a system performing poorly. Work on *Adversarial examples* (Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow, & Fergus, 2013) has shown that neural networks are vulnerable to the attacks perturbing the data in imperceptible ways. Many defenses have been proposed, but most of them rely on *obfuscated gradients* (Athalye, Carlini, & Wagner, 2018) to give a false illusion of defense by lowering the quality of the gradient signal, without actually improving robustness (Athalye et al., 2018). Of these defenses, only adversarial training (Kurakin, Goodfellow, & Bengio, 2016b) was still effective after addressing the problem of obfuscated gradients.

However, adversarial training has a major disadvantage: it drastically reduces the generalization performance of the networks on unperturbed data samples, especially for small networks. For example, Madry, Makelov, Schmidt, Tsipras, and Vladu (2017) report 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, a situation leading to great negative impact.

Our contribution: We propose to augment the adversarial training with the 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, CIFAR100 and SHVN) and benchmark architectures (Wide-ResNet and ResNet): Section 5.1.

---

* Corresponding authors.
  E-mail addresses: lambalex@iro.umontreal.ca (A. Lamb), vikas.verma@aalto.fi (V. Verma), kkawaguchi@fas.harvard.edu (K. Kawaguchi), alex.m@nus.edu.sg (A. Matyasko), juho.kaanala@aalto.fi (J. Kannala), yoshua.bengio@mila.quebec (Y. Bengio).
  [1] Equal contribution.

- We demonstrate that our approach does not suffer from *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, hence resulting in improved standard test error : Section 6.
- We mathematically analyze the benefit of the proposed method in terms of robustness and generalization. For robustness, we show that Interpolated Adversarial Training corresponds to approximately minimizing an upper bound of the adversarial loss with additional adversarial perturbations. This explains why models obtained by the proposed method preserve the adversarial robustness and can sometimes further improve the robustness when compared to standard adversarial training. For generalization, we prove a new generalization bound for Interpolated Adversarial Training and analyze the benefits of the proposed method.

2. Related work

The trade-off between standard test error and adversarial robustness has been studied in Madry et al. (2017), Raghunathan, Xie, Yang, Duchi, and Liang (2019), Tsipras, Santurkar, Engstrom, Turner, and Madry (2018) and Zhang, Yu, Jiao, Xing, Ghaoui, and Jordan (2019a). While Madry et al. (2017), Tsipras et al. (2018) and Zhang et al. (2019a) empirically demonstrate this trade-off, Tsipras et al. (2018) and Zhang et al. (2019a) demonstrate this trade-off theoretically as well on the constructed learning problems. Furthermore, Raghunathan et al. (2019) study this trade-off from the point-of-view of the statistical properties of the *robust objective* (Ben-Tal, El Ghaoui, & Nemirovski, 2009) 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, CIFAR-10, and CIFAR-100 datasets (Section 5.1) also consistently show higher standard test error with PGD adversarial training.

While Tsipras et al. (2018) 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 (Zoph & Le, 2016) was used to find architectures which achieve high robustness to PGD attacks as well as better test error on the unperturbed data (Cubuk, Zoph, Schoenholz, & Le, 2018). This improved test error on the unperturbed data and a direct comparison to our method is in Table 2. However, the method of Cubuk et al. (2018) 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 Cubuk et al., 2018), 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 Kolter and Wong (2017) 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). Wong, Schmidt, Metzen, and Kolter (2018) 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 type defenses of Madry et al., 2017) 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. Schmidt, Santurkar, Tsipras, Talwar, and Madry (2018) observe that a higher sample complexity is needed for better adversarially robust generalization. Yin, Ramchandran, and Bartlett (2018) demonstrate that adversarial training results in higher complexity models and hence poorer adversarially robust generalization. Furthermore, Schmidt et al. (2018) suggest that adversarially robust generalization requires more data and Carmon, Raghunathan, Schmidt, Liang, and Duchi (2019), Zhai et al. (2019) 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 (Berthelot, Carlini, Goodfellow, Papernot, Oliver, & Raffel, 2019; Jeong, Verma, Hyun, Kannala, & Kwak, 2021; Verma et al., 2019; Verma, Lamb, Kannala, Bengio, & Lopez-Paz, 2019; Verma, Qu, Lamb, Bengio, Kannala, & Tang, 2019; Zhang, Cissé, Dauphin, & Lopez-Paz, 2017) 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, Zhang, Hsieh, and Tao (2018) study the theoretical properties of interpolation based training techniques such as Mixup (Zhang et al., 2017)

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 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 $(x_1, y_1), \ldots\ldots, (x_N, y_N)$.

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 does not guarantee good performance on examples which are carefully perturbed to fool the model (Goodfellow, Shlens,

& Szegedy, 2014; Szegedy et al., 2013). That is, the empirical risk minimization framework suffers from a lack of robustness to adversarial attacks.

Madry et al. (2017) 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 Eq. (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 (Goodfellow et al., 2014) which is a single step and can still be effective against many networks. Secondly, we consider the projected gradient descent attack (Kurakin et al., 2016b) 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 (Madry et al., 2017). We briefly describe these two attacks as follows:

Fast Gradient Sign Method (FGSM). The Fast Gradient Sign Method (Goodfellow et al., 2014) 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 (Goodfellow et al., 2014; Madry et al., 2017), especially when no adversarial defenses are employed.

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

Projected Gradient Descent (PGD). The projected gradient descent attack (Madry et al., 2017), sometimes referred to as FGSM^k, is a multi-step extension of the FGSM attack characterized as follows:

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

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 (Athalye, Engstrom, Ilyas, & Kwok, 2017; Papernot, McDaniel, Sinha, & Wellman, 2016). 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 Tables 2–7.

3.4. Adversarial training

Adversarial training (Goodfellow et al., 2014) 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 (Zhang et al., 2017) and Manifold Mixup (Verma et al., 2019) 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 an 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) (Madry et al., 2017) or Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014)). 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 Kurakin, Goodfellow, and Bengio (2016a), Tsipras et al. (2018), 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 (Zhang et al., 2017) or Manifold Mixup (Verma et al., 2019), we summarize these supporting methods in more detail. The Mixup method (Zhang et al., 2017) 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.

Algorithm 1 The Interpolated Adversarial Training Algorithm

Require: f_θ: Neural Network
Require: *Mix*: A way of combining examples (Mixup or Manifold Mixup)
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 as in Madry et al., 2017)
    $\mathcal{L}_a = Loss(f_\theta, Mix, \tilde{x}_i, y_i)$ ▷ 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
    $\theta \leftarrow \text{Step}(\theta, g_\theta)$ ▷ Update parameters using gradients g (e.g. SGD)
  end for

The Manifold Mixup method (Verma et al., 2019) 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: Raghunathan et al. (2019) have shown that adversarial training could require more training samples to attain a higher standard test accuracy. Mixup (Zhang et al., 2017) and Manifold Mixup (Verma et al., 2019) 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: Shwartz-Ziv and Tishby (2017) and Tishby and Zaslavsky (2015) 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,

Table 1
Soft Rank (sum of singular values divided by largest singular value) of the representations (following first layer) from models trained with various methods. We report separately per MNIST class. FGSM and PGD refer to models trained with adversarial training. We note that FGSM slightly increases the numerical rank, but PGD (a much stronger attack) often dramatically increases it.

Class	Baseline	Manifold mixup	FGSM	PGD
0	2.87	2.14	3.34	3.91
1	2.90	1.91	2.92	4.15
2	3.74	2.64	4.29	6.51
3	3.27	2.66	4.29	5.48
4	3.18	2.41	3.58	4.70
5	3.72	2.74	4.82	6.75
6	3.22	2.26	3.66	5.90
7	3.43	2.39	3.66	4.42
8	3.09	2.78	4.50	6.84
9	3.20	2.46	3.71	5.19

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 (Verma et al., 2019) 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.

To provide further evidence for a difference in the compression characteristics, we trained 5-layer fully-connected models on MNIST and considered a bottleneck layer of 30 units directly following the first hidden layer. We then performed singular value decomposition on the per-class representations and looked at the spectrum of singular values (Fig. 1 and Table 1). We found that PGD dramatically increased the number of singular values with large values relative to a baseline model (FGSM was somewhere in-between baseline and PGD).
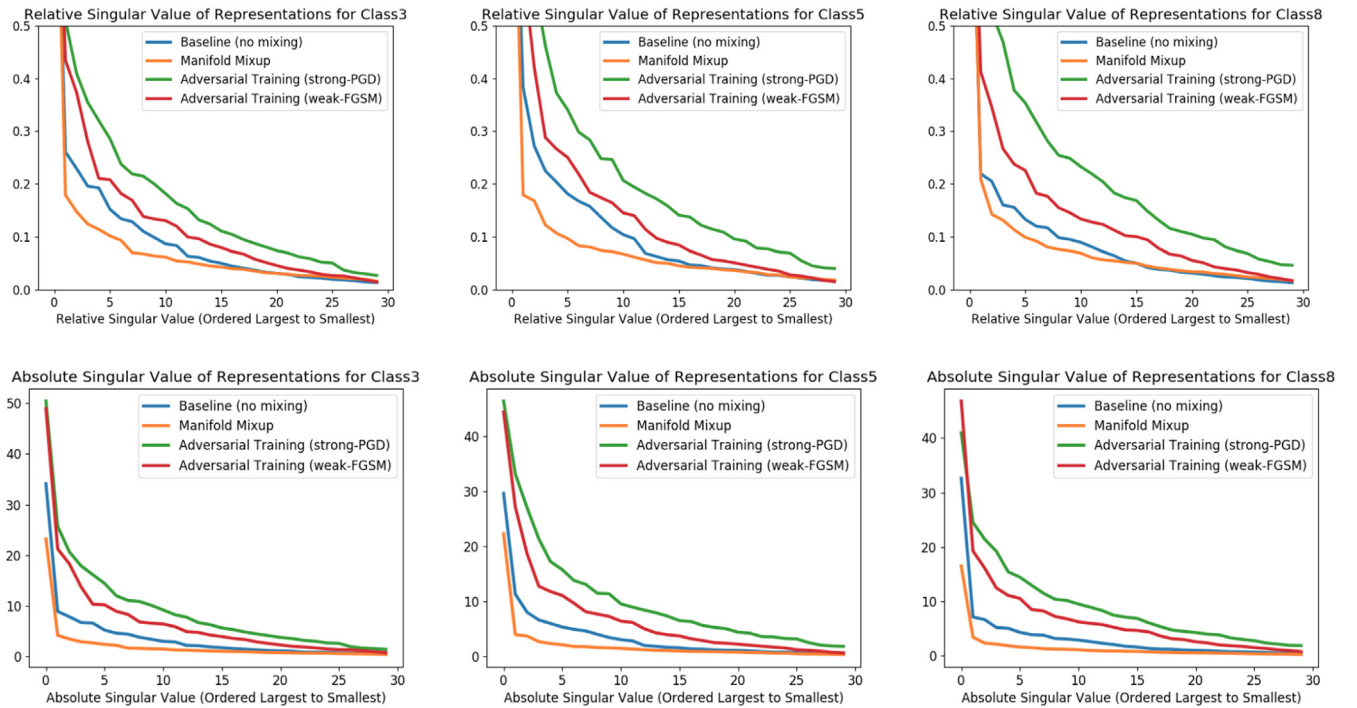
Fig. 1. Adversarial Training (especially with PGD) training makes representations have substantially more directions of significant variability (both when measured in an absolute sense and when measured relative to the largest singular value).

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 in Madry et al., 2017; Tsipras et al., 2018; Zhang et al., 2019a) 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 (Athalye et al., 2018).

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), the Carlini–Wagner attack (Carlini & Wagner, 2016), and the AutoAttack (Croce & Hein, 2020).

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 He, Zhang, Ren, and Sun (2015), Zagoruyko and Komodakis (2016) and used in Madry et al. (2017) for adversarial training[2]. Second, the PreActResnet18 architecture which is a variant of the residual architecture of He et al. (2015). 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 three benchmark datasets (CIFAR10, CIFAR100 and SVHN), which are commonly used in the adversarial robustness

literature (Croce & Hein, 2020; Madry et al., 2017). Both CIFAR-10 and CIFAR-100 datasets consist of 60000 color images each of size 32×32, split between 50K training and 10K test images. The CIFAR-10 dataset has ten classes, which include pictures of cars, horses, airplanes and deer. The CIFAR-100 dataset has one hundred classes grouped into 20 superclasses such as people, trees, vehicles, etc. 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 Madry et al. (2017). Namely, we use random cropping and horizontal flip for CIFAR10 and CIFAR100. 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 hyper-parameters as described above. For performing interpolation, we used either Manifold Mixup with $\alpha = 2.0$ as suggested in Verma et al. (2019) or Mixup with $alpha = 1.0$ as suggested in Zhang et al. (2017). 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 Verma et al. (2019).

Results: The results for CIFAR10, CIFAR100, SVHN datasets are presented in Tables 2–3, 4–5, 6–7, respectively. 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 2, 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

---

[2] While Madry et al. (2017) use WRN32-10 architecture, we use the standard WRN28-10 architecture, so our results are not directly comparable to their results.

Table 2

CIFAR10 results (error in %) to white-box attacks on WideResNet28-10 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[a]. 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 Cubuk et al. (2018) 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 (Madry et al., 2017)	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 (Cubuk et al., 2018)	6.80	36.4	49.9	–
PGD (7 steps) (Madry et al., 2017)	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

[a] 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).

Table 3

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

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 4

CIFAR100 results (error in %) to white-box attacks on WideResNet28-10 evaluated on the test data. The rows correspond to the training mechanism and columns correspond to adversarial attack methods. 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 and adversarial robustness.

Training	Adversary			
	No attack	FGSM	PGD (7 steps)	PGD (20 steps)
Baseline	22.23 ± 0.31	67.26 ± 0.42	76.08 ± 0.35	80.03 ± 0.58
Mixup	19.37 ± 0.09	61.97 ± 0.31	88.12 ± 1.77	93.77 ± 1.64
Manifold Mixup	18.75 ± 0.13	64.76 ± 0.29	93.42 ± 2.05	97.21 ± 1.3
PGD (7 steps)	40.80 ± 0.64	71.95 ± 0.47	77.21 ± 0.43	79.36 ± 0.49
Interpolated Adversarial Training (with Mixup)	33.43 ± 0.30	**67.47 ± 0.14**	**74.43 ± 0.14**	**77.7 ± 0.1**
Interpolated Adversarial Training (Manifold Mixup)	**33.27 ± 0.55**	67.93 ± 0.48	76.21 ± 0.61	79.80 ± 0.59

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 degree of adversarial robustness remains approximately unchanged, across varies type of adversarial attacks. We also considered using early stopping technique (Rice, Wong, & Kolter, 2020) to prevent robust overfitting. We found that early stopping reduced a PGD (20 steps) test error of the model trained with IAT on CIFAR-10 from 61.62% to 58.58%, albeit, with a slight increase of the standard test error from 10.12% to 10.62%. Thus, early stopping can be considered complementary to our method and can be used to further improve the adversarial robustness of IAT trained models.

We additionally ran with the challenging Carlini–Wagner (Carlini & Wagner, 2016) attack and the ensemble-based AutoAttack Croce & Hein, 2020 on CIFAR-10. With the Carlini–Wagner attack, the IAT trained network had consistently improved robustness, with test error of 28.61% on the un-targeted attack and 29.53% on the targeted attack. By contrast, the baseline model had higher test errors of 34.42% on the un-targeted attack and 33.37% on the targeted attack. For the L-inf version of Auto-Attack, IAT slightly increased the test error from 62.56% to 66.35%. For the L2 version of the attack, IAT improved the test error from 43.90% to 41.57%. We also compared IAT with Feature Scattering-based Adversarial Training (FSAT) (Zhang & Wang, 2019) and

Table 5
CIFAR100 results (error in %) to white-box attacks on PreActResNet18 evaluated on the test data. Rest of the details are same as Table 4.

Training	Adversary			
	No attack	FGSM	PGD (7 steps)	PGD (20 steps)
Baseline	25.0 ± 0.36	89.17 ± 0.28	99.92 ± 0.03	100.0 ± 0.0
Mixup	23.4 ± 0.27	75.26 ± 0.55	99.39 ± 0.03	99.94 ± 0.01
Manifold Mixup	22.36 ± 0.26	74.46 ± 0.25	99.06 ± 0.23	99.9 ± 0.01
PGD (7 steps)	43.49 ± 0.23	77.22 ± 0.35	82.01 ± 0.23	83.57 ± 0.38
Interpolated Adversarial Training (with Mixup)	$\mathbf{37.82 \pm 0.03}$	$\mathbf{72.27 \pm 0.52}$	79.21 ± 0.53	81.75 ± 0.57
Interpolated Adversarial Training (Manifold Mixup)	39.48 ± 0.49	72.52 ± 0.55	$\mathbf{79.12 \pm 0.43}$	$\mathbf{81.17 \pm 0.49}$

Table 6
SVHN results (error in %) to white-box attacks on WideResNet28-10 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	$\mathbf{22.08 \pm 0.15}$	45.74 ± 0.11	58.40 ± 0.46
Interpolated Adversarial Training (Manifold Mixup)	$\mathbf{3.38 \pm 0.22}$	22.30 ± 1.07	$\mathbf{42.61 \pm 0.40}$	$\mathbf{52.79 \pm 0.22}$

Table 7
SVHN results (error in %) to white-box attacks on PreActResnet18. Rest of the details are same as Table 6.

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	$\mathbf{23.57 \pm 0.64}$	47.69 ± 0.22	54.62 ± 0.18
Interpolated Adversarial Training (Manifold Mixup)	$\mathbf{3.61 \pm 0.22}$	24.95 ± 0.92	$\mathbf{46.62 \pm 0.28}$	54.13 ± 1.08

TRADES (Zhang, Yu, Jiao, Xing, Ghaoui, & Jordan, 2019b). We found that FSAT and TRADES had higher standard test errors of 10.49% and 27.5%, than IAT, which had a standard test error of 10.12%. We also compared the robustness of IAT, FSAT and TRADES trained models to PGD (7 steps) and AutoAttack. We found that the IAT trained model had lower robustness to PGD (7 steps) attack than FSAT and TRADES, with IAT having a PGD (7 steps) error of 55.43%, FSAT having a PGD (7 steps) error of 49.48%, and TRADES having a PGD (7 steps) error of 55.02%. Against AutoAttack, IAT improved robustness compared to FSAT, with IAT having a robust test error of 66.35% and FSAT having a robust test error of 67.34%. The TRADES trained model had a robust test error of 59.65% to AutoAttack. Overall, IAT improved standard test error while still achieving robustness comparable to state-of-the-art methods, such as FSAT and TRADES.

5.2. Transfer attacks

As a sanity check that *Interpolated Adversarial Training* does not suffer from gradient obfuscation (Athalye et al., 2018), 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 8) we found that black-box transfer were always substantially weaker than white-box attacks, hence *Interpolated Adversarial Training* does not suffer from gradient obfuscation (Athalye et al., 2018). Additionally, in Table 9, we observe that increasing ϵ results in 100% success of attack, providing added evidence that *Interpolated Adversarial Training* does not suffer from gradient obfuscation (Athalye et al., 2018).

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 (Engstrom, Ilyas, & Athalye, 2018) have been found to have increasing vulnerability when exposed to attacks with a large number of iterations. We studied this (Table 10) 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 9).

Table 8
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 (Athalye et al., 2018).

ϵ	2			5			10		
Target	Attack								
	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 9
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.

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

Model	Num. iterations					
	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

Table 11
Clean and Adversarial Test Errors on CIFAR-10 test data as a function of weighting of clean and adversarial losses during IAT training. For PGD, we used a ℓ_∞ projected gradient descent with 7 steps, step size $\alpha = 2$, and $\epsilon = 8$.

Weighting	Model			
	Vanilla (No attack)	IAT (No attack)	Vanilla (PGD)	IAT (PGD)
20% $\mathcal{L}_c$, 80% $\mathcal{L}_a$	15.78%	**11.24%**	57.92%	**55.41%**
40% $\mathcal{L}_c$, 60% $\mathcal{L}_a$	14.79%	**10.63%**	58.02%	**55.44%**
50% $\mathcal{L}_c$, 50% $\mathcal{L}_a$	14.12%	**10.12%**	57.76%	**55.43%**
60% $\mathcal{L}_c$, 40% $\mathcal{L}_a$	14.15%	**9.89%**	58.42%	**56.48%**
80% $\mathcal{L}_c$, 20% $\mathcal{L}_a$	12.2%	**10.30%**	**58.56%**	64.42%

5.4. Analysis of weighting of loss terms

IAT introduces a hyperparameter for the weighting of the clean loss and the adversarial loss, which by default can be set to an even weighting of both terms. We found that when we weighted the clean loss more, we had improved clean test accuracy and when we weighted the adversarial loss more, we had improved adversarial robustness. These results are shown in Table 11.

6. Theoretical analysis

In this section, we establish mathematical properties of IAT with Mixup. We begin in Section 6.1 with additional notation and then analyze the effect of IAT on adversarial robustness in

Section 6.2. Moreover, we discuss the effects of IAT on generalization in Section 6.3 by showing how ICT can reduce overfitting and lead to better generalization behaviors. The proofs of all theorems and propositions are presented in Appendix B using a key lemma proven in Appendix A.

6.1. Notation

In order to present our analysis succinctly, we introduce additional notation as follows. The standard mixup loss $\mathcal{L}_c$ can be written as

$$\mathcal{L}_c = \frac{1}{n^2} \sum_{i,j=1}^n \mathbb{E}_{\lambda \sim \mathcal{D}_\lambda} \ell(f_\theta(\tilde{x}_{i,j}(\lambda)), \tilde{y}_{i,j}(\lambda)), \tag{4}$$

where $\tilde{x}_{i,j}(\lambda) = \lambda x_i + (1 - \lambda) x_j$, $\tilde{y}_{i,j}(\lambda) = \lambda y_i + (1 - \lambda) y_j$, and $\lambda \in [0, 1]$. Here, $\mathcal{D}_\lambda$ represents the Beta distribution $Beta(\alpha, \beta)$ with some hyper-parameters $\alpha, \beta > 0$. Similarly, the adversarial-mixup loss $\mathcal{L}_a$ used in IAT can be defined by

$$\mathcal{L}_a = \frac{1}{n^2} \sum_{i,j=1}^n \mathbb{E}_{\lambda \sim \mathcal{D}_\lambda} \ell(f_\theta(\check{x}_{i,j}(\lambda)), \tilde{y}_{i,j}(\lambda)), \tag{5}$$

where $\hat{\delta}_i = \text{argmax}_{\delta_i : \|\delta_i\|_\rho \leq \epsilon} \ell(f_\theta(x_i + \delta_i), y_i)$, $\hat{x}_i = x_i + \hat{\delta}_i$, and $\check{x}_{i,j}(\lambda) = \lambda \hat{x}_i + (1 - \lambda) \hat{x}_j$. Using these two types of losses, the whole IAT loss is defined by

$$\mathcal{L} = \frac{\mathcal{L}_c + \mathcal{L}_a}{2}. \tag{6}$$

In this section, we focus on the following family of loss functions: $\ell(q, y) = h(q) - yq$, for some twice differentiable function h. This family of loss functions ℓ includes many commonly used losses, including the logistic loss and the cross-entropy loss.

Given a set $\mathcal{F}$ of functions $x \mapsto f(x)$, the Rademacher complexity (Bartlett & Mendelson, 2002; Mohri, Rostamizadeh, & Talwalkar, 2012) of the set $\ell \circ \mathcal{F} = \{(x, y) \mapsto \ell(f(x; \theta), y) : f \in \mathcal{F}\}$ can be defined by

$$\mathcal{R}_n(\ell \circ \mathcal{F}) := \mathbb{E}_{S,\sigma} \left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \sigma_i \ell(f(x_i), y_i) \right],$$
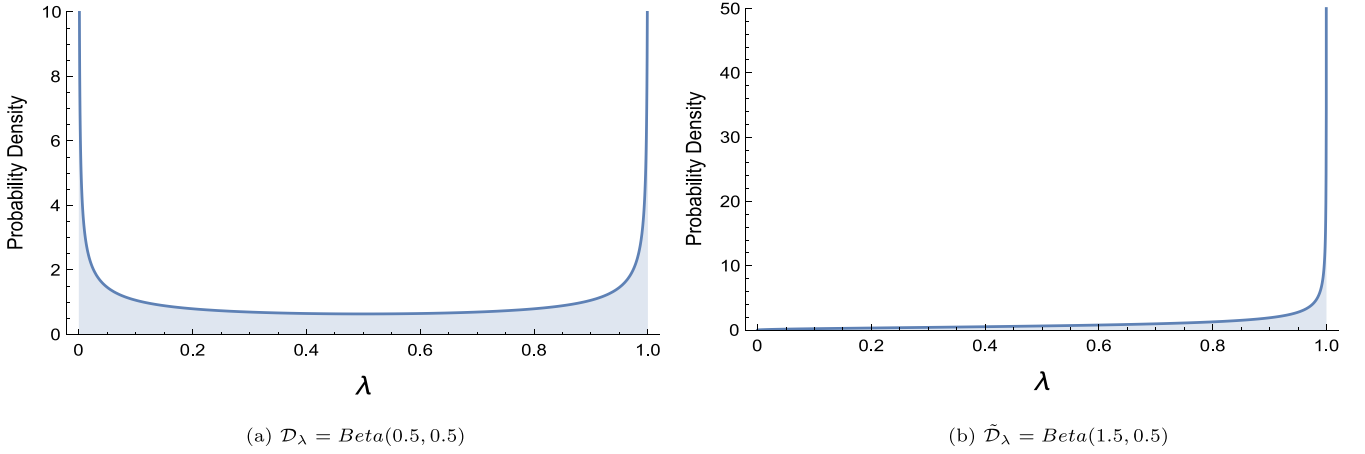
(a) $\mathcal{D}_\lambda = Beta(0.5, 0.5)$

(b) $\tilde{\mathcal{D}}_\lambda = Beta(1.5, 0.5)$

Fig. 2. The comparison of the distribution $\mathcal{D}_\lambda$ of the Mixup coefficient λ and the distribution $\tilde{\mathcal{D}}_\lambda$ of $\tilde{\lambda}$ in the error term E.

where $S = ((x_i, y_i))_{i=1}^n$. Here, $\sigma_1, \ldots, \sigma_n$ are independent uniform random variables taking values in $\{-1, 1\}$. We denote by $\tilde{\mathcal{D}}_\lambda$ the uniform mixture of two Beta distributions, $\frac{\alpha}{\alpha+\beta}Beta(\alpha + 1, \beta) + \frac{\beta}{\alpha+\beta}Beta(\beta + 1, \alpha)$. We let $\mathcal{D}_{\hat{x}}$ be the empirical distribution of the perturbed training samples $(\hat{x}_1, \ldots, \hat{x}_n)$, and define $\mathcal{D}_x$ to be the empirical distribution of the training samples $(x_1, \ldots, x_n)$. Let $\cos(a, b)$ be the cosine similarity of two vector a and b.

6.2. The effect of IAT on robustness

In this subsection, we study how adding Mixup to adversarial training affects the robustness of the model by analyzing the adversarial-mixup loss $\mathcal{L}_a$ used in IAT. This subsection focuses on the binary cross-entropy loss ℓ by setting $h(z) = \log(1 + e^z)$ and $y \in \{0, 1\}$, whereas the next section considers a more general setting. We define a set Θ of parameter vectors by

$$\Theta = \left\{ \theta \in \mathbb{R}^d : y_i(f_\theta(x_i + \hat{\delta}_i)) + (y_i - 1)(f_\theta(x_i + \hat{\delta}_i)) \geq 0 \right.$$
$$\left. \text{for all } i = 1, \ldots, n \right\}.$$

Note that this set Θ contains the set of all parameter vectors with correct classifications of training points (before Mixup) as

$$\Theta \supseteq \{\theta \in \mathbb{R}^d : \mathbb{1}\{f_\theta(x_i + \hat{\delta}_i) \geq 0\} = y_i \text{ for all } i = 1, \ldots, n\}.$$

Therefore, the condition of $\theta \in \Theta$ is satisfied when the deep neural network classifies all labels correctly for the training data with perturbations before Mixup. As the training error (although not training loss) becomes zero in finite time in many practical cases, the condition of $\theta \in \Theta$ is satisfied in finite time in many practical cases. Accordingly, we study the effect of IAT on robustness in the regime of $\theta \in \Theta$.

Theorem 1 shows that the adversarial-mixup loss $\mathcal{L}_a$ is approximately an upper bound of the adversarial loss with the adversarial perturbations of $x_i \mapsto x_i + \hat{\delta}_i + \delta_i^{\text{mix}}$ where $\|\delta_i\|_\rho \leq \epsilon$ is the standard adversarial perturbation and $\|\delta_i\|_2 \leq \epsilon_i^{\text{mix}}$ is the non-standard *additional* perturbation due to IAT. In other words, IAT is approximately minimizing the upper bound of the adversarial loss with additional adversarial perturbation $\|\delta_i\|_2 \leq \epsilon_i^{\text{mix}}$ with data-dependent radius ϵ_i^{mix} for each $i \in \{1, \ldots, n\}$. Therefore, adding Mixup to adversarial training (i.e., IAT) does not decrease the effect of the original adversarial training on the robustness approximately (where the approximation error is in the order of $(1 - \tilde{\lambda})^3$ as discussed below). This is non-trivial because, without Theorem 1, it is uncertain whether or not adding Mixup reduces

the effect of adversarial training in terms of the robustness. Moreover, Theorem 1 shows that IAT further improves the robustness depending on the values of data-dependent radius ϵ_i^{mix} when compared to standard adversarial training without Mixup. These are consistent with our experimental observations.

Theorem 1. *Let $\theta \in \Theta$ be a point such that $\nabla f_\theta(x_i + \hat{\delta}_i)$ and $\nabla^2 f_\theta(x_i + \hat{\delta}_i)$ exist for all $i = 1, \ldots, n$. Assume that $f_\theta(x_i + \hat{\delta}_i) = \nabla f_\theta(x_i + \hat{\delta}_i)^\top(x_i + \hat{\delta}_i)$ and $\nabla^2 f_\theta(x_i + \hat{\delta}_i) = 0$ for all $i \in \{1, \ldots, n\}$. Suppose that $\mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}[r] = 0$ and $\|x_i + \hat{\delta}_i\|_2 \geq c_x\sqrt{d}$ for all $i \in \{1, \ldots, n\}$. Then, there exists a pair $(\varphi, \bar{\varphi})$ such that $\lim_{z \to 0} \varphi(z)$, $\lim_{z \to 0} \bar{\varphi}(z) = 0$, and*

$$\mathcal{L}_a \geq \frac{1}{n} \sum_{i=1}^n \max_{\|\delta_i^{\text{mix}}\|_2 \leq \epsilon_i^{\text{mix}}} \ell(f_\theta(x_i + \hat{\delta}_i + \delta_i^{\text{mix}}), y_i) + E_1 + E_2,$$

where $\epsilon_i^{\text{mix}} := R_i c_x \mathbb{E}_\lambda[(1 - \lambda)]\sqrt{d}$, $R_i := |\cos(\nabla f_\theta(x_i + \hat{\delta}_i), x_i + \hat{\delta}_i)|$, $E_1 := \mathbb{E}_{\tilde{\lambda} \sim \tilde{\mathcal{D}}_\lambda}[(1 - \tilde{\lambda})^2 \varphi(1 - \tilde{\lambda})]$, and $E_2 := \mathbb{E}_{\tilde{\lambda} \sim \tilde{\mathcal{D}}_\lambda}[(1 - \tilde{\lambda})]^2 \bar{\varphi} (\mathbb{E}_{\tilde{\lambda} \sim \tilde{\mathcal{D}}_\lambda}[(1 - \tilde{\lambda})])$.

The assumption of $f_\theta(z) = \nabla f_\theta(z)^\top z$ and $\nabla^2 f_\theta(z) = 0$ in Theorem 1 is satisfied, for example, by linear models as well as deep neural networks with ReLU activation function and max-pooling. In Theorem 1, the approximation error terms E_1 and E_2 are in the order of $(1 - \tilde{\lambda})^3$ (since $\lim_{z \to 0} \varphi(z)$, $\lim_{z \to 0} \bar{\varphi}(z) = 0$), and $\tilde{\lambda}$ tends to be close to one since $\tilde{\lambda} \sim \tilde{\mathcal{D}}_\lambda$ where $\tilde{\mathcal{D}}_\lambda$ is the uniform mixture of two Beta distributions, $\frac{\alpha}{\alpha+\beta}Beta(\alpha + 1, \beta) + \frac{\beta}{\alpha+\beta}Beta(\beta + 1, \alpha)$, given the distribution of $\mathcal{D}_\lambda = Beta(\alpha, \beta)$ for Mixup coefficient λ. For example, if the IAT algorithm uses the Beta distribution $\mathcal{D}_\lambda = Beta(0.5, 0.5)$ for Mixup, then we have $\tilde{\mathcal{D}}_\lambda = Beta(1.5, 0.5)$, for which $\tilde{\lambda} \sim \tilde{\mathcal{D}}_\lambda$ tends to be close to one as illustrated in Fig. 2. Therefore, the approximation error terms E_1 and E_2 tend to be close to zero.

6.3. The effect of IAT on generalization

In this subsection, we mathematically analyze the effect of IAT on generalization properties. We start in Section 6.3.1 with th general setting with arbitrary h and f_θ, and prove a generalization bound for IAT loss. In Section 6.3.2, we then make assumptions on h and f_θ and study the regularization effects of IAT.

6.3.1. Generalization bounds

The following theorem presents a generalization bound for the IAT loss $\frac{\mathcal{L}_c + \mathcal{L}_a}{2}$ — the upper bound on the difference between the

expected error on unseen data and the IAT loss, $\mathbb{E}_{x,y}[\ell(f(x),y)] - \frac{\mathcal{L}_c + \mathcal{L}_a}{2}$:

Theorem 2. *Let $\rho \geq 1$ be a real number and $\mathcal{F}$ be a set of maps $x \mapsto f(x)$. Assume that the function $|\ell(q,y) - \ell(q',y)| \leq \tau$ for any $q, q' \in \{f(x+\delta) : f \in \mathcal{F}, x \in \mathcal{X}, \|\delta\|_\rho \leq \epsilon\}$ and $y \in \mathcal{Y}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over an i.i.d. draw of n i.i.d. samples $((x_i, y_i))_{i=1}^n$, the following holds: for all maps $f \in \mathcal{F}$, there exists a function $\varphi : \mathbb{R} \to \mathbb{R}$ such that*

$$\mathbb{E}_{x,y}[\ell(f(x),y)] - \frac{\mathcal{L}_c + \mathcal{L}_a}{2} \tag{7}$$

$$\leq 2\mathcal{R}_n(\ell \circ \mathcal{F}) + 2\tau\sqrt{\frac{\ln(1/\delta)}{2n}} - \frac{Q(f)}{2}$$

$$- \mathbb{E}_X\left[\sum_{k=1}^3 \frac{G_k(\hat{X}, \mathcal{D}_{\hat{x}}) + G_k(X, \mathcal{D}_x)}{2}\right] - E_1,$$

where $\lim_{q \to 0} \varphi(q) = 0$, $X := (x_1, \ldots, x_n)$, $\hat{X} := (\hat{x}_1, \ldots, \hat{x}_n)$,

$$Q(f) := \frac{1}{n}\mathbb{E}_S\left[\sum_{i=1}^n \left(\max_{\delta_i : \|\delta_i\|_\rho \leq \epsilon} \ell(f(x_i + \delta_i), y_i) - \ell(f(x_i), y_i)\right)\right] \geq 0,$$

$$G_1(\hat{X}, \mathcal{D}_{\hat{x}}) := \frac{\mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[1 - \lambda]}{n}\sum_{i=1}^n (h'(f(\hat{x}_i)) - y_i)\nabla f(\hat{x}_i)^\top \mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}[r - \hat{x}_i],$$

$$G_2(\hat{X}, \mathcal{D}_{\hat{x}}) := \frac{\mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[(1 - \lambda)^2]}{2n}\sum_{i=1}^n h''(f(\hat{x}_i))\nabla f(\hat{x}_i)^\top \mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}$$
$$[(r - \hat{x}_i)(r - \hat{x}_i)^\top]\nabla f(\hat{x}_i),$$

$$G_3(\hat{X}, \mathcal{D}_{\hat{x}}) := \frac{\mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[(1 - \lambda)^2]}{2n}\sum_{i=1}^n (h'(f(\hat{x}_i)) - y_i)\mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}$$
$$[(r - \hat{x}_i)\nabla^2 f(\hat{x}_i)(r - \hat{x}_i)^\top].$$

To understand this generalization bound further, we now compare it with a generalization bound for IAT without using Mixup on terms $\mathcal{L}_c$ and $\mathcal{L}_a$. IAT without Mixup is the adversarial training along with the standard training, which minimizes the loss of

$$\mathcal{L}' = \frac{\mathcal{L}'_c + \mathcal{L}'_a}{2},$$

where

$$\mathcal{L}'_c = \frac{1}{n}\sum_{i=1}^n \ell(f_\theta(x_i), y_i), \text{ and} \tag{8}$$

$$\mathcal{L}'_a = \frac{1}{n}\sum_{i=1}^n \ell(f_\theta(x_i + \hat{\delta}_i), y_i). \tag{9}$$

The following theorem presents a generalization bound for IAT without Mixup on terms $\mathcal{L}_c$ and $\mathcal{L}_a$:

Theorem 3. *Let $\rho \geq 1$ be a real number and $\mathcal{F}$ be a set of maps $x \mapsto f(x)$. Assume that the function $|\ell(q,y) - \ell(q',y)| \leq \tau$ for any $q, q' \in \{f(x+\delta) : f \in \mathcal{F}, x \in \mathcal{X}, \|\delta\|_\rho \leq \epsilon\}$ and $y \in \mathcal{Y}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over an i.i.d. draw of n i.i.d. samples $((x_i, y_i))_{i=1}^n$, the following holds: for all maps $f \in \mathcal{F}$,*

$$\mathbb{E}_{x,y}[\ell(f(x),y)] - \frac{\mathcal{L}'_c + \mathcal{L}'_a}{2} \leq 2\mathcal{R}_n(\ell \circ \mathcal{F}) + 2\tau\sqrt{\frac{\ln(1/\delta)}{2n}} - \frac{Q(f)}{2}. \tag{10}$$

By comparing Theorems 2 and 3, we can see that the benefit of IAT with Mixup comes from the two mechanisms in terms of generalization. The first mechanism is based on the term of $\mathbb{E}_X\left[\sum_{k=1}^3 \frac{G_k(\hat{X}, \mathcal{D}_{\hat{x}}) + G_k(X, \mathcal{D}_x)}{2}\right] + E_1$. If this term is positive, then IAT

with Mixup has a better generalization bound than that of IAT without Mixup (if we suppose that the Rademacher complexity term $\mathcal{R}_n(\ell \circ \mathcal{F})$ is the same for both methods). The second mechanism is based on the model complexity term $\mathcal{R}_n(\ell \circ \mathcal{F})$. As the model complexity term is bounded by the norms of trained weights (e.g., Bartlett, Foster, & Telgarsky, 2017), this term differs for different training schemes — IAT with Mixup and IAT without Mixup. Accordingly, we study the regularization effects of IAT on the norms of weights in the next subsection.

6.3.2. Regularization effects

The generalization bounds in the previous subsection contain the model complexity term, which are controlled by the norms of the weights in the previous studies (e.g., Bartlett et al., 2017). Accordingly, we now discuss the regularization effects of IAT on the norms of weights. This subsection considers the models where $f_\theta(x_i + \hat{\delta}_i) = \nabla f_\theta(x_i + \hat{\delta}_i)^\top(x_i + \hat{\delta}_i)$ and $\nabla^2 f_\theta(x_i + \hat{\delta}_i) = 0$ for $i = 1, \ldots, n$. This is satisfied by linear models as well as deep neural networks with ReLU activation functions and max-pooling. We let $y \in \{0, 1\}$ and $h(z) = \log(1 + e^z)$, which makes the loss function ℓ to represent the binary cross-entropy loss. Define g to be the logic function as $g(z) = \frac{e^z}{1 + e^z}$. This definition implies that $g(z) \in (0, 1)$ for $z \in \mathbb{R}$.

The following theorem shows that the IAT term has the additional regularization effect on $\|\nabla f_\theta(\hat{x}_i)\|_2$ and $\|\nabla f_\theta(\hat{x}_i)\|^2_{\mathbb{E}_r[(r - \hat{x}_i)(r - \hat{x}_i)^\top]}$. This theorem explains the additional regularization effects of the IAT term on the norm of weights, since $\nabla f_\theta(\hat{x}_i) = w$ for linear models and $\nabla f_\theta(\hat{x}_i) = \|W^H \dot{\sigma}^H W^{H-1} \dot{\sigma}^{H-1} \ldots \dot{\sigma}^1 W^1\|$ for deep neural networks with ReLU and max-pooling.

Theorem 4. *Assume that $f_\theta(x_i + \hat{\delta}_i) = \nabla f_\theta(x_i + \hat{\delta}_i)^\top(x_i + \hat{\delta}_i)$ and $\nabla^2 f_\theta(x_i + \hat{\delta}_i) = 0$. Then, there exists a function $\varphi : \mathbb{R} \to \mathbb{R}$ such that*

$$\mathcal{L}_a = \frac{1}{n}\sum_{i=1}^n \ell(f_\theta(\hat{x}_i), y_i) + C_1\|\nabla f_\theta(\hat{x}_i)\|_2 + C_2\|\nabla f_\theta(\hat{x}_i)\|^2_{\mathbb{E}_r[(r - \hat{x}_i)(r - \hat{x}_i)^\top]} + E_1,$$

$$(11)$$

where $\lim_{q \to 0} \varphi(q) = 0$ and

$$C_1 = \frac{\mathbb{E}_\lambda[(1 - \lambda)]}{n}\sum_{i=1}^n (y_i - g(f_\theta(\hat{x}_i)))\|\mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}[r - \hat{x}_i]\|_2$$
$$\cos(\nabla f_\theta(\hat{x}_i), \mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}[r - \hat{x}_i]),$$

$$C_2 = \frac{\mathbb{E}_\lambda[(1 - \lambda)^2]}{2n}\sum_{i=1}^n |g(f_\theta(\hat{x}_i))(1 - g(f_\theta(\hat{x}_i)))|.$$

In Theorem 4, C_2 is always strictly positive since $g(z) \in (0, 1)$ for all $z \in \mathbb{R}$. While C_1 can be negative in general, the following proposition shows that C_1 will be also non-negative in the later phase of IAT training:

Proposition 1. *If $\theta \in \Theta'$, then $C_1 \geq 0$ where $\Theta' = \{\theta \in \mathbb{R}^d : y_i(f_\theta(x_i + \delta_i(\theta)) - \zeta_i) + (y_i - 1)(f_\theta(x_i + \delta_i(\theta)) - \zeta_i) \geq 0$ for all $i = 1, \ldots, n\}$, and $\zeta_i = \nabla f_\theta(x_i + \delta_i(\theta))^\top \mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}[r]$.*

Here, we have that

$$\Theta' \supseteq \{\theta \in \mathbb{R}^d : \hat{\mathbb{1}}\{f_\theta(x_i + \hat{\delta}_i) - \zeta_i \geq 0\} = y_i \text{ for all } i = 1, \ldots, n\}.$$

Therefore, the condition of $\theta \in \Theta'$ is satisfied when the model classifies all labels correctly with margin ζ_i for adversarial perturbations. As the training error (although not training loss) becomes zero in finite time in many practical cases and margin increases via implicit bias of gradient descent after that Lyu and Li (2020), the condition of $\theta \in \Theta'$ is satisfied in finite time in many practical cases. Theorem 4 and Proposition 1 together show that IAT
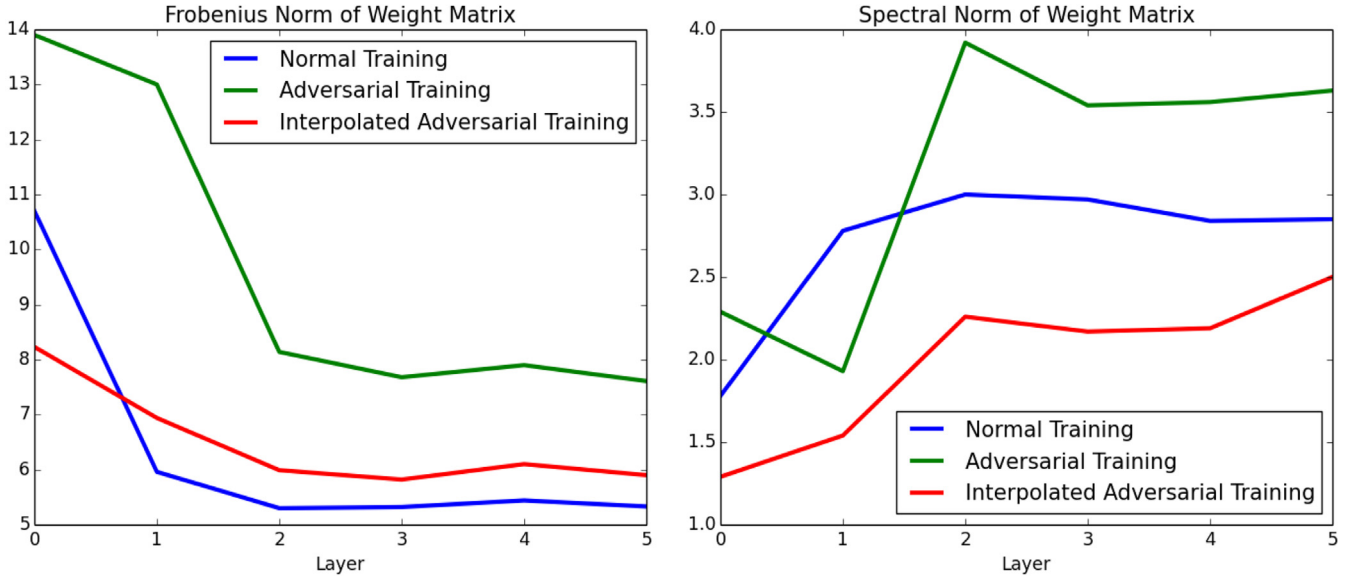
Fig. 3. We analyzed the Frobenius and spectral norms of the weight matrices on a 6-layer network. Generally Adversarial Training makes these norms larger, whereas *Interpolated Adversarial Training* brings these norms closer to their values when doing normal training.

can reduce the norms of weights when compared to adversarial training.

Zhang, Deng, and Kawaguchi (2021) showed that the standard mixup loss $\mathcal{L}_c$ also has the regularization effect on the norm of weights and thus contribute to reduce the model complexity. Therefore, our result together with that of the previous study (Zhang et al., 2021) shows the benefit of IAT in terms of reducing the norm of weights to control the model complexity. As the recent study only considers the standard Mixup without adversarial training, our result complements the recent study to understand IAT.

To validate this theoretical prediction, we computed the norms of weights for a 6-layer fully-connected network with 512 hidden units trained on Fashion-MNIST and report the results in Fig. 3. On the one hand, adversarial training increased the Frobenius norms across all the layers and increased the spectral norm of the majority of the layers. On the other hand, IAT avoided or mitigated these increases in the norms of weights. This is consistent with our theoretical predictions and suggests that IAT learns lower complexity classifiers than normal adversarial training.

To further understand why adversarial training tends to increase the norms, consider the case of linear regression:

$$L(\theta) = \frac{1}{2} \|Xw - Y\|_F^2.$$

Then, we have

$$\nabla L(\theta) = X^\top (Xw - Y).$$

Therefore, each step of (stochastic) gradient descent only adds some vector in the column space of $X^\top$ to w as

$$w_{t+1} = w_t + v_t \quad \text{where } v_t \in \text{Col}(X^\top).$$

Here, the solutions of the linear regression are any w such that

$$w = X^\dagger Y + v^\perp \quad \text{where } v^\perp \in \text{Null}(X).$$

Thus, (stochastic) gradient descent does not add any unnecessary element to w, implicitly minimizing the norm of the weights. Accordingly, if we initialize w as $w_0 \in \text{Col}(X^\top)$, then we achieve the minimum norm solution implicitly via (stochastic) gradient descent.

In this context, we can easily see that by conducting adversarial training, we add vectors $v^\perp \in \text{Null}(X)$, breaking the implicit bias and increasing the norm of w. Similarly, in the case of deep neural networks, (stochastic) gradient descent has the implicit bias that restricts the search space of w and hence tend to minimize the norm without unnecessary elements (Lyu & Li, 2020; Moroshko, Gunasekar, Woodworth, Lee, Srebro, & Soudry, 2020; Woodworth et al., 2020). Thus, similarly to the case of linear models, adversarial training adds extra elements via the perturbation and tends to increase the norm of weights. Our results show that we can minimize this effect via the additional regularization effects of IAT to reduce overfitting for better generalization behaviors.

7. Conclusion

Robustness to the 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 the unperturbed data. This has both the theoretical and the 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 reduces the model's ability to perform well on unperturbed test data. This degradation in the generalization is critically urgent to the practitioners whose systems are threatened by the adversarial attacks. With current techniques those wishing to deploy machine learning systems need to consider a severe trade-off between performance on the unperturbed data and the robustness to the 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 (Verma et al., 2019; Zhang et al., 2017). We found that this substantially improves generalization on unperturbed data while preserving adversarial robustness. Our analysis showed why and how the proposed method can improve generalization and preserve adversarial robustness when compared to standard adversarial training.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Appendix A. Lemma on adversarial-mixup loss

This appendix provides the key lemma, Lemma 1, which is used to prove theorems in Appendix B.

Lemma 1. *For any f_θ, there exists a function $\varphi : \mathbb{R} \to \mathbb{R}$ such that*

$$\mathcal{L}_a = \frac{1}{n} \sum_{i=1}^n \ell(f_\theta(\hat{x}_i), y_i) + \sum_{i=1}^3 G_i + \mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[(1-\lambda)^2 \varphi(1-\lambda)],$$

where $\lim_{q \to 0} \varphi(q) = 0$ and

$$G_1 = \frac{\mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[1-\lambda]}{n} \sum_{i=1}^n (h'(f_\theta(\hat{x}_i)) - y_i) \nabla f_\theta(\hat{x}_i)^\top \mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}[r - \hat{x}_i],$$

$$G_2 = \frac{\mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[(1-\lambda)^2]}{2n} \sum_{i=1}^n h''(f_\theta(\hat{x}_i)) \nabla f_\theta(\hat{x}_i)^\top \mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}[(r-\hat{x}_i)(r-\hat{x}_i)^\top] \nabla f_\theta(\hat{x}_i),$$

$$G_3 = \frac{\mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[(1-\lambda)^2]}{2n} \sum_{i=1}^n (h'(f_\theta(\hat{x}_i)) - y_i) \mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}[(r-\hat{x}_i) \nabla^2 f_\theta(\hat{x}_i)(r-\hat{x}_i)^\top].$$

Proof. Since $\ell(q, y) = h(q) - yq$, we have that

$$\frac{1}{n} \sum_{i=1}^n \ell(f_\theta(\hat{x}_i), y_i) = \frac{1}{n} \sum_{i=1}^n [h(f_\theta(\hat{x}_i)) - y_i f_\theta(\hat{x}_i)],$$

and

$$\mathcal{L}_a = \frac{1}{n^2} \mathbb{E}_{\lambda \sim Beta(\alpha,\beta)} \sum_{i,j=1}^n [h(f_\theta(\check{x}_{i,j}(\lambda))) - (\lambda y_i + (1-\lambda)y_j)f_\theta(\check{x}_{i,j}(\lambda))].$$

By expanding $(\lambda y_i + (1-\lambda)y_j)f_\theta(\check{x}_{i,j}(\lambda))$ and using $h(f_\theta(\check{x}_{i,j}(\lambda))) = \lambda h(f_\theta(\check{x}_{i,j}(\lambda))) + (1-\lambda)h(f_\theta(\check{x}_{i,j}(\lambda)))$,

$$\mathcal{L}_a = \frac{1}{n^2} \mathbb{E}_{\lambda \sim Beta(\alpha,\beta)} \sum_{i,j=1}^n \Big\{ \lambda[h(f_\theta(\check{x}_{i,j}(\lambda))) - y_i f_\theta(\check{x}_{i,j}(\lambda))]$$
$$+ (1-\lambda)[h(f_\theta(\check{x}_{i,j}(\lambda))) - y_j f_\theta(\check{x}_{i,j}(\lambda))] \Big\}.$$

Using the fact that $\mathbb{E}_{B \sim Bern(\lambda)}[B] = \lambda$, we have

$$\mathcal{L}_a = \frac{1}{n^2} \mathbb{E}_{\lambda \sim Beta(\alpha,\beta)} \mathbb{E}_{B \sim Bern(\lambda)} \sum_{i,j=1}^n \Big\{ B[h(f_\theta(\check{x}_{i,j}(\lambda))) - y_i f_\theta(\check{x}_{i,j}(\lambda))]$$
$$+ (1-B)[h(f_\theta(\check{x}_{i,j}(\lambda))) - y_j f_\theta(\check{x}_{i,j}(\lambda))] \Big\}.$$

Since $\lambda \sim Beta(\alpha, \beta), B|\lambda \sim Bern(\lambda)$, by conjugacy, we can exchange them to have

$$B \sim Bern(\frac{\alpha}{\alpha + \beta}), \lambda \mid B \sim Beta(\alpha + B, \beta + 1 - B).$$

Thus,

$$\mathcal{L}_a = \frac{1}{n^2} \sum_{i,j=1}^n \Big\{ \frac{\alpha}{\alpha + \beta} \mathbb{E}_{\lambda \sim Beta(\alpha+1,\beta)}[h(f_\theta(\check{x}_{i,j}(\lambda))) - y_i f_\theta(\check{x}_{i,j}(\lambda))]$$
$$+ \frac{\beta}{\alpha + \beta} \mathbb{E}_{\lambda \sim Beta(\alpha,\beta+1)}[h(f_\theta(\check{x}_{i,j}(\lambda))) - y_j f_\theta(\check{x}_{i,j}(\lambda))] \Big\}.$$

Since $1 - Beta(\alpha, \beta + 1)$ and $Beta(\beta + 1, \alpha)$ represent the same distribution and $\check{x}_{ij}(1 - \lambda) = \check{x}_{ji}(\lambda)$, we have

$$\sum_{i,j} \mathbb{E}_{\lambda \sim Beta(\alpha,\beta+1)}[h(f_\theta(\check{x}_{i,j}(\lambda))) - y_j f_\theta(\check{x}_{i,j}(\lambda))]$$
$$= \sum_{i,j} \mathbb{E}_{\lambda \sim Beta(\beta+1,\alpha)}[h(f_\theta(\check{x}_{i,j}(\lambda))) - y_i f_\theta(\check{x}_{i,j}(\lambda))].$$

By defining $\tilde{\mathcal{D}}_\lambda = \frac{\alpha}{\alpha+\beta} Beta(\alpha + 1, \beta) + \frac{\beta}{\alpha+\beta} Beta(\beta + 1, \alpha)$,

$$\mathcal{L}_a = \frac{1}{n^2} \sum_{i,j=1}^n \mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[h(f_\theta(\check{x}_{i,j}(\lambda))) - y_i f_\theta(\check{x}_{i,j}(\lambda))]$$
$$= \frac{1}{n^2} \sum_{i,j=1}^n \mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda} \ell(f_\theta(\lambda \hat{x}_i + (1-\lambda)\hat{x}_j), y_i).$$

By defining $\mathcal{D}_{\hat{x}}$ to be the empirical distribution induced by perturbed training samples $(\hat{x}_j)_{j=1}^n$,

$$\mathcal{L}_a = \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda} \mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}} \ell(f_\theta(\lambda \hat{x}_i + (1-\lambda)r), y_i)$$

Let $\check{x}_i = \lambda \hat{x}_i + (1-\lambda)r$, $\alpha = 1 - \lambda$, and $\psi_i(\alpha) = \ell(f_\theta(\check{x}_i), y_i)$. Then, using the definition of the twice-differentiability of function ψ_i,

$$\ell(f_\theta(\check{x}_i), y_i) = \psi_i(\alpha) = \psi_i(0) + \psi_i'(0)\alpha + \frac{1}{2}\psi_i''(0)\alpha^2 + \alpha^2 \varphi_i(\alpha),$$

where $\lim_{z \to 0} \varphi_i(z) = 0$. Therefore,

$$\mathcal{L}_a = \frac{1}{n} \mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda} \mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}} \sum_{i=1}^n [\psi_i(0) + \psi_i'(0)\alpha + \frac{1}{2}\psi_i''(0)\alpha^2]$$
$$+ \mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[(1-\lambda)^2 \varphi(1-\lambda)], \tag{A.1}$$

where $\varphi(\alpha) = \frac{1}{n} \sum_{i=1}^n \varphi_i(\alpha)$. By linearity and chain rule,

$$\psi_i'(\alpha) = h'(f_\theta(\check{x}_i)) \frac{\partial f_\theta(\check{x}_i)}{\partial \check{x}_i} \frac{\partial \check{x}_i}{\partial \alpha} - y_i \frac{\partial f_\theta(\check{x}_i)}{\partial \check{x}_i} \frac{\partial \check{x}_i}{\partial \alpha}$$
$$= h'(f_\theta(\check{x}_i)) \frac{\partial f_\theta(\check{x}_i)}{\partial \check{x}_i}(r - \hat{x}_i) - y_i \frac{\partial f_\theta(\check{x}_i)}{\partial \check{x}_i}(r - \hat{x}_i)$$

where we used $\frac{\partial \check{x}_i}{\partial \alpha} = (r - \hat{x}_i)$. Moreover,

$$\frac{\partial}{\partial \alpha} \frac{\partial f_\theta(\check{x}_i)}{\partial \check{x}_i}(r - \hat{x}_i) = \frac{\partial}{\partial \alpha}(r - \hat{x}_i)^\top \left[\frac{\partial f_\theta(\check{x}_i)}{\partial \check{x}_i} \right]^\top$$
$$= (r - \hat{x}_i)^\top \nabla^2 f_\theta(\check{x}_i) \frac{\partial \check{x}_i}{\partial \alpha}$$
$$= (r - \hat{x}_i)^\top \nabla^2 f_\theta(\check{x}_i)(r - \hat{x}_i).$$

Therefore,

$$\psi_i''(\alpha) = h'(f_\theta(\check{x}_i))(r - \hat{x}_i)^\top \nabla^2 f_\theta(\check{x}_i)(r - \hat{x}_i)$$
$$+ h''(f_\theta(\check{x}_i))[\frac{\partial f_\theta(\check{x}_i)}{\partial \check{x}_i}(r - \hat{x}_i)]^2 - y_i(r - \hat{x}_i)^\top \nabla^2 f_\theta(\check{x}_i)(r - \hat{x}_i).$$

By setting $\alpha = 0$,

$$\psi_i'(0) = h'(f_\theta(\hat{x}_i)) \nabla f_\theta(\hat{x}_i)^\top (r - \hat{x}_i) - y_i \nabla f_\theta(\hat{x}_i)^\top (r - \hat{x}_i)$$
$$= (h'(f_\theta(\hat{x}_i)) - y_i) \nabla f_\theta(\hat{x}_i)^\top (r - \hat{x}_i),$$

and

$$\psi_i''(0) = h'(f_\theta(\hat{x}_i))(r - \hat{x}_i)^\top \nabla^2 f_\theta(\hat{x}_i)(r - \hat{x}_i)$$
$$+ h''(f_\theta(\hat{x}_i))[\nabla f_\theta(\hat{x}_i)^\top (r - \hat{x}_i)]^2$$
$$- y_i(r - \hat{x}_i)^\top \nabla^2 f_\theta(\hat{x}_i)(r - \hat{x}_i)$$
$$= h''(f_\theta(\hat{x}_i)) \nabla f_\theta(\hat{x}_i)^\top (r - \hat{x}_i)(r - \hat{x}_i)^\top \nabla f_\theta(\hat{x}_i)$$
$$+ (h'(f_\theta(\hat{x}_i)) - y_i)(r - \hat{x}_i)^\top \nabla^2 f_\theta(\hat{x}_i)(r - \hat{x}_i).$$

By substituting these into (A.1), we obtain the statement of this lemma. □

Appendix B. Proofs

Using Lemmas 1 proven in Appendix A, this appendix provides the complete proofs of Theorems 1, 2, 3, 4, and Proposition 1.

Proof of Theorem 1. Let $\hat{x}_i = x_i + \hat{\delta}_i$. From the assumption, we have $f_\theta(\hat{x}_i) = \nabla f_\theta(\hat{x}_i)^\top \hat{x}_i$ and $\nabla^2 f_\theta(\hat{x}_i) = 0$. Since $h(z) = \log(1+e^z)$, we have $h'(z) = \frac{e^z}{1+e^z} = g(z) \geq 0$ and $h''(z) = \frac{e^z}{(1+e^z)^2} = g(z)(1 - g(z)) \geq 0$. By substituting these into the equation of Lemma 1 with $\mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}[r] = 0$,

$$\mathcal{L}_a = \frac{1}{n} \sum_{i=1}^n \ell(f_\theta(\hat{x}_i), y_i) + G_1 + G_2 + E_1, \qquad (B.1)$$

where

$$G_1 = \frac{\mathbb{E}_\lambda[(1 - \lambda)]}{n} \sum_{i=1}^n (y_i - g(f_\theta(\hat{x}_i))) f_\theta(\hat{x}_i)$$

$$G_2 = \frac{\mathbb{E}_\lambda[(1 - \lambda)^2]}{2n} \sum_{i=1}^n |g(f_\theta(\hat{x}_i))(1 - g(f_\theta(\hat{x}_i)))| \nabla f_\theta(\hat{x}_i)^\top$$
$$\mathbb{E}_r[(r - \hat{x}_i)(r - \hat{x}_i)^\top] \nabla f_\theta(\hat{x}_i)$$
$$\geq \frac{\mathbb{E}_\lambda[(1 - \lambda)]^2}{2n} \sum_{i=1}^n |g(f_\theta(\hat{x}_i))(1 - g(f_\theta(\hat{x}_i)))| \nabla f_\theta(\hat{x}_i)^\top$$
$$\mathbb{E}_r[(r - \hat{x}_i)(r - \hat{x}_i)^\top] \nabla f_\theta(\hat{x}_i)$$

where we used $\mathbb{E}[z^2] = \mathbb{E}[z]^2 + \mathrm{Var}(z) \geq \mathbb{E}[z]^2$ and $\nabla f_\theta(\hat{x}_i)^\top \mathbb{E}_r[(r - \hat{x}_i)(r - \hat{x}_i)^\top] \nabla f_\theta(\hat{x}_i) \geq 0$. Since $\mathbb{E}_r[(r - \hat{x}_i)(r - \hat{x}_i)^\top] = \mathbb{E}_r[rr^\top - r\hat{x}_i^\top - \hat{x}_i r^\top + \hat{x}_i \hat{x}_i^\top] = \mathbb{E}_r[rr^\top] + \hat{x}_i \hat{x}_i^\top$ where $\mathbb{E}_r[rr^\top]$ is positive semidefinite,

$$G_2 \geq \frac{\mathbb{E}_\lambda[(1 - \lambda)]^2}{2n} \sum_{i=1}^n |g(f_\theta(\hat{x}_i))(1 - g(f_\theta(\hat{x}_i)))| \nabla f_\theta(\hat{x}_i)^\top$$
$$(\mathbb{E}_r[rr^\top] + \hat{x}_i \hat{x}_i^\top) \nabla f_\theta(\hat{x}_i).$$
$$\geq \frac{\mathbb{E}_\lambda[(1 - \lambda)]^2}{2n} \sum_{i=1}^n |g(f_\theta(\hat{x}_i))(1 - g(f_\theta(\hat{x}_i)))| (\nabla f_\theta(\hat{x}_i)^\top \hat{x}_i)^2$$
$$= \frac{\mathbb{E}_\lambda[(1 - \lambda)]^2}{2n} \sum_{i=1}^n |g(f_\theta(\hat{x}_i))(1 - g(f_\theta(\hat{x}_i)))| \|\nabla f_\theta(\hat{x}_i)\|_2^2 \|\hat{x}_i\|_2^2$$
$$(\cos(\nabla f_\theta(\hat{x}_i), \hat{x}_i))^2$$
$$\geq \frac{1}{2n} \sum_{i=1}^n |g(f_\theta(\hat{x}_i))(1 - g(f_\theta(\hat{x}_i)))| \|\nabla f_\theta(\hat{x}_i)\|_2^2 R_i^2 c_x^2 \mathbb{E}_\lambda[(1 - \lambda)]^2 d$$

Now we bound $G_1 = \frac{\mathbb{E}_\lambda[(1-\lambda)]}{n} \sum_{i=1}^n (y_i - g(f_\theta(\hat{x}_i))) f_\theta(\hat{x}_i)$ by using $\theta \in \Theta$. Since $\theta \in \Theta$, we have $y_i f_\theta(\hat{x}_i) + (y_i - 1) f_\theta(\hat{x}_i) \geq 0$, which implies that $f_\theta(\hat{x}_i) \geq 0$ if $y_i = 1$ and $f_\theta(\hat{x}_i) \leq 0$ if $y_i = 0$. Thus, if $y_i = 1$,

$$(y_i - g(f_\theta(\hat{x}_i)))(f_\theta(\hat{x}_i)) = (1 - g(f_\theta(\hat{x}_i)))(f_\theta(\hat{x}_i)) \geq 0,$$

since $(f_\theta(\hat{x}_i)) \geq 0$ and $(1 - g(f_\theta(\hat{x}_i))) \geq 0$ due to $g(f_\theta(\hat{x}_i)) \in (0, 1)$. If $y_i = 0$,

$$(y_i - g(f_\theta(\hat{x}_i)))(f_\theta(\hat{x}_i)) = -g(f_\theta(\hat{x}_i))(f_\theta(\hat{x}_i)) \geq 0,$$

since $(f_\theta(\hat{x}_i)) \leq 0$ and $-g(f_\theta(\hat{x}_i)) < 0$. Therefore, for all $i = 1, \ldots, n$,

$$(y_i - g(f_\theta(\hat{x}_i)))(f_\theta(\hat{x}_i)) \geq 0,$$

which implies that, since $\mathbb{E}_\lambda[(1 - \lambda)] \geq 0$,

$$G_1 = \frac{\mathbb{E}_\lambda[(1 - \lambda)]}{n} \sum_{i=1}^n |y_i - g(f_\theta(\hat{x}_i))| |f_\theta(\hat{x}_i)|$$
$$= \frac{\mathbb{E}_\lambda[(1 - \lambda)]}{n} \sum_{i=1}^n |g(f_\theta(\hat{x}_i)) - y_i| \|\nabla f_\theta(\hat{x}_i)\|_2 \|\hat{x}_i\|_2 |\cos(\nabla f_\theta(\hat{x}_i), \hat{x}_i)|$$
$$\geq \frac{1}{n} \sum_{i=1}^n |g(f_\theta(\hat{x}_i)) - y_i| \|\nabla f_\theta(\hat{x}_i)\|_2 R_i c_x \mathbb{E}_\lambda[(1 - \lambda)] \sqrt{d}$$

By substituting these lower bounds of G_1 and G_2 into (B.1), we obtain

$$\mathcal{L}_a - \frac{1}{n} \sum_{i=1}^n \ell(f_\theta(\hat{x}_i), y_i) \qquad (B.2)$$
$$\geq \frac{1}{n} \sum_{i=1}^n |g(f_\theta(\hat{x}_i)) - y_i| \|\nabla f_\theta(\hat{x}_i)\|_2 \epsilon_i^{\mathrm{mix}}$$
$$+ \frac{1}{2n} \sum_{i=1}^n |h''(f_\theta(\hat{x}_i))| \|\nabla f_\theta(\hat{x}_i)\|_2^2 (\epsilon_i^{\mathrm{mix}})^2 + E_1$$

On the other hand, for any $z_1, \ldots, z_n$, there exist functions φ_i' such that $\lim_{z \to 0} \varphi_i'(z) = 0$, and

$$\frac{1}{n} \sum_{i=1}^n \max_{\|\delta_i\|_2 \leq \epsilon_i} \ell(f_\theta(z_i + \delta_i), y_i) - \frac{1}{n} \sum_{i=1}^n \ell(f_\theta(z_i), y_i)$$
$$\leq \frac{1}{n} \sum_{i=1}^n |g(f_\theta(z_i)) - y_i| \|\nabla f_\theta(z_i)\|_2 \epsilon_i$$
$$+ \frac{1}{2n} \sum_{i=1}^n |h''(f_\theta(z_i))| \|\nabla f_\theta(z_i)\|_2^2 \epsilon_i^2 + \frac{1}{n} \sum_{i=1}^n \max_{\|\delta_i\|_2 \leq \epsilon_i} \|\delta_i\|_2^2 \varphi_i'(\delta_i)$$
$$\leq \frac{1}{n} \sum_{i=1}^n |g(f_\theta(z_i)) - y_i| \|\nabla f_\theta(z_i)\|_2 \epsilon_i$$
$$+ \frac{1}{2n} \sum_{i=1}^n |h''(f_\theta(z_i))| \|\nabla f_\theta(z_i)\|_2^2 \epsilon_i^2 + \frac{1}{n} \sum_{i=1}^n \epsilon_i^2 \varphi_i''(\epsilon_i) \qquad (B.3)$$

where $\varphi_i''(\epsilon_i) = \max_{\|\delta_i\|_2 \leq \epsilon_i} \varphi_i'(\delta_i)$. Note that $\lim_{q \to 0} \varphi_i''(q) = 0$. By combining (B.2) and (B.3),

$$\mathcal{L}_a \geq \frac{1}{n} \sum_{i=1}^n \max_{\|\delta_i^{\mathrm{mix}}\|_2 \leq \epsilon_i^{\mathrm{mix}}} \ell(f_\theta(x_i + \hat{\delta}_i + \delta_i^{\mathrm{mix}}), y_i)$$
$$+ \mathbb{E}_\lambda[(1 - \lambda)^2 \varphi(1 - \lambda)] - \frac{1}{n} \sum_{i=1}^n (\epsilon_i^{\mathrm{mix}})^2 \varphi''(\epsilon_i^{\mathrm{mix}}),$$

where $\epsilon_i^{\mathrm{mix}} = R_i c_x \mathbb{E}_\lambda[(1-\lambda)] \sqrt{d}$. Define $\bar{\varphi}(q) = -\frac{1}{n} \sum_{i=1}^n (R_i c_x \sqrt{d})^2 \varphi_i''(R_i c_x \sqrt{d} q)$. Note that $\lim_{q \to 0} \bar{\varphi}(q) = 0$. Then, since $\frac{1}{n} \sum_{i=1}^n (\epsilon_i^{\mathrm{mix}})^2 \varphi''(\epsilon_i^{\mathrm{mix}}) = -E_2$,

$$\mathcal{L}_a \geq \frac{1}{n} \sum_{i=1}^n \max_{\|\delta_i^{\mathrm{mix}}\|_2 \leq \epsilon_i^{\mathrm{mix}}} \ell(f_\theta(x_i + \hat{\delta}_i + \delta_i^{\mathrm{mix}}), y_i) + E_1 + E_2.$$

where $\lim_{z \to 0} \bar{\varphi}(z) = 0$ and $\lim_{z \to 0} \varphi(z) = 0$. □

Proof of Theorem 2. Let $S = ((x_i, y_i))_{i=1}^n$ and $S' = ((x_i', y_i'))_{i=1}^n$. Define

$$\varphi(S) = \sup_{f \in \mathcal{F}} \mathbb{E}_{x,y}[\ell(f(x), y)] - \frac{\mathcal{L}_c + \mathcal{L}_a}{2}. \tag{B.4}$$

To apply McDiarmid's inequality to $\varphi(S)$, we compute an upper bound on $|\varphi(S) - \varphi(S')|$ where S and S' be two test datasets differing by exactly one point of an arbitrary index i_0; i.e., $S_i = S_i'$ for all $i \neq i_0$ and $S_{i_0} \neq S_{i_0}'$. Then,

$$\varphi(S') - \varphi(S) \leq \frac{\tau(2n-1)}{n^2} \leq \frac{2\tau}{n}, \tag{B.5}$$

where we use the fact that both $\mathcal{L}_c$ and $\mathcal{L}_a$ have n^2 terms and $2n-1$ terms differ for S and S', each of which is bounded by the constant τ. Similarly, $\varphi(S) - \varphi(S') \leq \frac{2\tau}{n}$. Thus, by McDiarmid's inequality, for any $\delta > 0$, with probability at least $1 - \delta$,

$$\varphi(S) \leq \mathbb{E}_S[\varphi(S)] + 2\tau \sqrt{\frac{\ln(1/\delta)}{2n}}. \tag{B.6}$$

Moreover, by using Lemma 1, there exist functions φ' and φ'' such that

$$\mathcal{L}_a = \frac{1}{n}\sum_{i=1}^n \ell(f_\theta(\hat{x}_i), y_i) + \sum_{i=1}^3 G_i + \mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[(1-\lambda)^2 \varphi'(1-\lambda)], \tag{B.7}$$

and

$$\mathcal{L}_c = \frac{1}{n}\sum_{i=1}^n \ell(f_\theta(x_i), y_i) + \sum_{i=1}^3 R_i + \mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[(1-\lambda)^2 \varphi''(1-\lambda)], \tag{B.8}$$

where $\lim_{q \to 0} \varphi'(q) = 0$ and $\lim_{q \to 0} \varphi''(q) = 0$. Thus, by defining

$$Q(f) = \frac{1}{n}\mathbb{E}_S\left[\sum_{i=1}^n \left(\max_{\delta_i: \|\delta_i\|_\rho \leq \epsilon} \ell(f(x_i + \delta_i), y_i) - \ell(f(x_i), y_i)\right)\right], \tag{B.9}$$

and

$$V(f) = \mathbb{E}_S\left[\sum_{i=1}^3 \frac{G_i + R_i}{2}\right] - \mathbb{E}_{\lambda \sim \tilde{\mathcal{D}}_\lambda}[(1-\lambda)^2 \varphi(1-\lambda)], \tag{B.10}$$

we have that

$$\mathbb{E}_S[\varphi(S)] \tag{B.11}$$

$$= \mathbb{E}_S\left[\sup_{f \in \mathcal{F}} \mathbb{E}_{S'}\left[\frac{1}{n}\sum_{i=1}^n \ell(f(x_i'), y_i')\right] - \frac{\mathcal{L}_c + \mathcal{L}_a}{2}\right] \tag{B.12}$$

$$= \mathbb{E}_S\left[\sup_{f \in \mathcal{F}} \mathbb{E}_{S'}\left[\frac{1}{n}\sum_{i=1}^n \ell(f(x_i'), y_i')\right] - \frac{1}{n}\sum_{i=1}^n \ell(f(x_i), y_i)\right] - \frac{Q(f)}{2} - V(f) \tag{B.13}$$

$$\leq \mathbb{E}_{S,S'}\left[\sup_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^n (\ell(f(x_i'), y_i') - \ell(f(x_i), y_i))\right] - \frac{Q(f)}{2} - V(f) \tag{B.14}$$

$$\leq \mathbb{E}_{\xi,S,S'}\left[\sup_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^n \xi_i(\ell(f(x_i'), y_i') - \ell(f(x_i), y_i))\right] - \frac{Q(f)}{2} - V(f) \tag{B.15}$$

$$\leq 2\mathbb{E}_{\xi,S}\left[\sup_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^n \xi_i \ell(f(x_i), y_i))\right] - \frac{Q(f)}{2} - V(f) \tag{B.16}$$

$$= 2\mathcal{R}_n(\ell \circ \mathcal{F}) - \frac{Q(f)}{2} - V(f) \tag{B.17}$$

where the second line follows the definitions of each term, the third line uses $\pm \frac{1}{n}\sum_{i=1}^n \ell(f(x_i), y_i)$ inside the expectation and the linearity of expectation, the fourth line uses the Jensen's inequality and the convexity of the supremum, and the fifth line follows that for each $\xi_i \in \{-1, +1\}$, the distribution of each

term $\xi_i(\ell(f(x_i'), y_i') - \ell(f(x_i), y_i))$ is the distribution of $(\ell(f(x_i'), y_i') - \ell(f(x_i), y_i))$ since S and S' are drawn iid with the same distribution. The sixth line uses the subadditivity of supremum.

Finally, by noticing that $Q(f) \geq 0$ from the definition of $Q(f) \geq 0$ (since $\max_{\delta_i: \|\delta_i\|_\rho \leq \epsilon} \ell(f(x_i + \delta_i), y_i) - \ell(f(x_i), y_i) \geq 0$) and by combining Eqs. (B.6) and (B.17), we have the desired statement. □

Proof of Theorem 3. Let $S = ((x_i, y_i))_{i=1}^n$ and $S' = ((x_i', y_i'))_{i=1}^n$. Define

$$\varphi(S) = \sup_{f \in \mathcal{F}} \mathbb{E}_{x,y}[\ell(f(x), y)] - \frac{\mathcal{L}_c' + \mathcal{L}_a'}{2}. \tag{B.18}$$

To apply McDiarmid's inequality to $\varphi(S)$, we compute an upper bound on $|\varphi(S) - \varphi(S')|$ where S and S' be two test datasets differing by exactly one point of an arbitrary index i_0; i.e., $S_i = S_i'$ for all $i \neq i_0$ and $S_{i_0} \neq S_{i_0}'$. Then,

$$\varphi(S') - \varphi(S) \leq \frac{2\tau}{n}, \tag{B.19}$$

since $\sup_{f \in \mathcal{F}} \frac{\max_{\delta_{i_0}: \|\delta_{i_0}\|_\rho \leq \epsilon} \ell(f(x_{i_0} + \delta_{i_0}), y_{i_0}) - \max_{\delta_{i_0}: \|\delta_{i_0}\|_\rho \leq \epsilon} \ell(f(x_{i_0}' + \delta_{i_0}), y_{i_0}')}{2n} \leq \frac{\tau}{n}$. Similarly, $\varphi(S) - \varphi(S') \leq \frac{2\tau}{n}$. Thus, by McDiarmid's inequality, for any $\delta > 0$, with probability at least $1 - \delta$,

$$\varphi(S) \leq \mathbb{E}_S[\varphi(S)] + 2\tau \sqrt{\frac{\ln(1/\delta)}{2n}}. \tag{B.20}$$

Moreover, by defining

$$Q(f) = \frac{1}{n}\mathbb{E}_S\left[\sum_{i=1}^n \left(\max_{\delta_i: \|\delta_i\|_\rho \leq \epsilon} \ell(f(x_i + \delta_i), y_i) - \ell(f(x_i), y_i)\right)\right], \tag{B.21}$$

we have that

$$\mathbb{E}_S[\varphi(S)] \tag{B.22}$$

$$= \mathbb{E}_S\left[\sup_{f \in \mathcal{F}} \mathbb{E}_{S'}\left[\frac{1}{n}\sum_{i=1}^n \ell(f(x_i'), y_i')\right] - \frac{\mathcal{L}_c' + \mathcal{L}_a'}{2}\right] \tag{B.23}$$

$$= \mathbb{E}_S\left[\sup_{f \in \mathcal{F}} \mathbb{E}_{S'}\left[\frac{1}{n}\sum_{i=1}^n \ell(f(x_i'), y_i')\right] - \frac{1}{n}\sum_{i=1}^n \ell(f(x_i), y_i)\right] - \frac{Q(f)}{2} \tag{B.24}$$

$$\leq \mathbb{E}_{S,S'}\left[\sup_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^n (\ell(f(x_i'), y_i') - \ell(f(x_i), y_i))\right] - \frac{Q(f)}{2} \tag{B.25}$$

$$\leq \mathbb{E}_{\xi,S,S'}\left[\sup_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^n \xi_i(\ell(f(x_i'), y_i') - \ell(f(x_i), y_i))\right] - \frac{Q(f)}{2} \tag{B.26}$$

$$\leq 2\mathbb{E}_{\xi,S}\left[\sup_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^n \xi_i \ell(f(x_i), y_i))\right] - \frac{Q(f)}{2} = 2\mathcal{R}_n(\ell \circ \mathcal{F}) - \frac{Q(f)}{2} \tag{B.27}$$

where the second line follows the definitions of each term, the third line uses $\pm \frac{1}{n}\sum_{i=1}^n \ell(f(x_i), y_i)$ inside the expectation and the linearity of expectation, the fourth line uses the Jensen's inequality and the convexity of the supremum, and the fifth line follows that for each $\xi_i \in \{-1, +1\}$, the distribution of each term $\xi_i(\ell(f(x_i'), y_i') - \ell(f(x_i), y_i))$ is the distribution of $(\ell(f(x_i'), y_i') - \ell(f(x_i), y_i))$ since S and S' are drawn iid with the same distribution. The sixth line uses the subadditivity of supremum.

Finally, by noticing that $Q(f) \geq 0$ from the definition of $Q(f) \geq 0$ (since $\max_{\delta_i: \|\delta_i\|_\rho \leq \epsilon} \ell(f(x_i + \delta_i), y_i) - \ell(f(x_i), y_i) \geq 0$)

and by combining Eqs. (B.20) and (B.27), we have the desired statement. □

Proof of Theorem 4. From the assumption, we have $f_\theta(\hat{x}_i) = \nabla f_\theta(\hat{x}_i)^\top \hat{x}_i$ and $\nabla^2 f_\theta(\hat{x}_i) = 0$. Since $h(z) = \log(1 + e^z)$, we have $h'(z) = \frac{e^z}{1+e^z} = g(z) \geq 0$ and $h''(z) = \frac{e^z}{(1+e^z)^2} = g(z)(1 - g(z)) \geq 0$. By substituting these into the equation of Lemma 1,

$$\mathcal{L}_a = \frac{1}{n} \sum_{i=1}^n \ell(f_\theta(\hat{x}_i), y_i) + G_1 + G_2 + \mathbb{E}_\lambda[(1-\lambda)^2 \varphi(1-\lambda)], \quad \text{(B.28)}$$

where

$$G_1 = \frac{\mathbb{E}_\lambda[(1-\lambda)]}{n} \sum_{i=1}^n (y_i - g(f_\theta(\hat{x}_i))) \|\mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}[r - \hat{x}_i]\|_2$$

$$\cos(\nabla f_\theta(\hat{x}_i), \mathbb{E}_{r \sim \mathcal{D}_{\hat{x}}}[r - \hat{x}_i]) \|\nabla f_\theta(\hat{x}_i)\|_2,$$

$$G_2 = \frac{\mathbb{E}_\lambda[(1-\lambda)^2]}{2n} \sum_{i=1}^n |g(f_\theta(\hat{x}_i))(1 - g(f_\theta(\hat{x}_i)))| \|\nabla f_\theta(\hat{x}_i)\|^2_{\mathbb{E}_r[(r-\hat{x}_i)(r-\hat{x}_i)^\top]}. \quad \Box$$

Proof of Proposition 1. Since $\theta \in \Theta'$, we have $y_i(f_\theta(\hat{x}_i) - \zeta_i) + (y_i - 1)(f_\theta(\hat{x}_i) - \zeta_i) \geq 0$, which implies that $f_\theta(\hat{x}_i) - \zeta_i \geq 0$ if $y_i = 1$ and $f_\theta(\hat{x}_i) - \zeta_i \leq 0$ if $y_i = 0$. Thus, if $y_i = 1$,

$$(y_i - g(f_\theta(\hat{x}_i)))(f_\theta(\hat{x}_i) - \zeta_i) = (1 - g(f_\theta(\hat{x}_i)))(f_\theta(\hat{x}_i) - \zeta_i) \geq 0,$$

since $(f_\theta(\hat{x}_i) - \zeta_i) \geq 0$ and $(1 - g(f_\theta(\hat{x}_i))) \geq 0$ due to $g(f_\theta(\hat{x}_i)) \in (0, 1)$. If $y_i = 0$,

$$(y_i - g(f_\theta(\hat{x}_i)))(f_\theta(\hat{x}_i) - \zeta_i) = -g(f_\theta(\hat{x}_i))(f_\theta(\hat{x}_i) - \zeta_i) \geq 0,$$

since $(f_\theta(\hat{x}_i) - \zeta_i) \leq 0$ and $-g(f_\theta(\hat{x}_i)) < 0$. Therefore, for all $i = 1, \ldots, n$,

$$(y_i - g(f_\theta(\hat{x}_i)))(f_\theta(\hat{x}_i) - \zeta_i) \geq 0.$$

This implies that, since $\mathbb{E}_\lambda[(1-\lambda)] \geq 0$ and $f_\theta(\hat{x}_i) = \nabla f_\theta(\hat{x}_i)^\top \hat{x}_i$,

$$G_1 = \frac{\mathbb{E}_\lambda[(1-\lambda)]}{n} \sum_{i=1}^n (y_i - g(f_\theta(\hat{x}_i)))(f_\theta(\hat{x}_i) - \zeta_i) \geq 0.$$

Thus, we have that $0 \leq G_1 = C_1 \|\nabla f_\theta(\hat{x}_i)\|_2$ where $\|\nabla f_\theta(\hat{x}_i)\|_2 \geq 0$, which implies that $C_1 \geq 0$. □

References

Athalye, A., Carlini, N., & Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. arXiv e-prints arXiv:1802.00420.

Athalye, A., Engstrom, L., Ilyas, A., & Kwok, K. (2017). Synthesizing robust adversarial examples. CoRR, abs/1707.07397. URL http://arxiv.org/abs/1707.07397, arXiv:1707.07397.

Bartlett, P. L., Foster, D. J., & Telgarsky, M. (2017). Spectrally-normalized margin bounds for neural networks. CoRR, abs/1706.08498. URL http://arxiv.org/abs/1706.08498, arXiv:1706.08498.

Bartlett, P. L., & Mendelson, S. (2002). Rademacher and Gaussian complexities: Risk bounds and structural results. JMLR.

Ben-Tal, A., El Ghaoui, L., & Nemirovski, A. (2009). Princeton series in applied mathematics, Robust optimization. Princeton University Press.

Berthelot, D., Carlini, N., Goodfellow, I. J., Papernot, N., Oliver, A., & Raffel, C. (2019). Mixmatch: A holistic approach to semi-supervised learning. CoRR, abs/1905.02249. URL http://arxiv.org/abs/1905.02249, arXiv:1905.02249.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., et al. (2016). End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316.

Carlini, N., & Wagner, D. A. (2016). Towards evaluating the robustness of neural networks. CoRR, abs/1608.04644. URL http://arxiv.org/abs/1608.04644, arXiv:1608.04644.

Carmon, Y., Raghunathan, A., Schmidt, L., Liang, P., & Duchi, J. C. (2019). Unlabeled data improves adversarial robustness. arXiv e-prints, arXiv:1905.13736.

Croce, F., & Hein, M. (2020). Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. CoRR, abs/2003.01690. URL http://arxiv.org/abs/2003.01690, arXiv:2003.01690.

Cubuk, E. D., Zoph, B., Schoenholz, S. S., & Le, Q. V. (2018). Intriguing properties of adversarial examples. URL https://openreview.net/forum?id=rk6H0ZbRb.

Engstrom, L., Ilyas, A., & Athalye, A. (2018). Evaluating and understanding the robustness of adversarial logit pairing. arXiv preprint arXiv:1807.10272.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv e-prints arXiv:1412.6572.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. CoRR, abs/1512.03385, URL http://arxiv.org/abs/1512.03385, arXiv:1512.03385.

Jeong, J., Verma, V., Hyun, M., Kannala, J., & Kwak, N. (2021). Interpolation-based semi-supervised learning for object detection. In 2021 IEEE/CVF conference on computer vision and pattern recognition.

Kolter, J. Z., & Wong, E. (2017). Provable defenses against adversarial examples via the convex outer adversarial polytope. CoRR, abs/1711.00851. http://arxiv.org/abs/1711.00851, arXiv:1711.00851.

Kurakin, A., Goodfellow, I. J., & Bengio, S. (2016a). Adversarial examples in the physical world. CoRR, abs/1607.02533. URL http://arxiv.org/abs/1607.02533, arXiv:1607.02533.

Kurakin, A., Goodfellow, I. J., & Bengio, S. (2016b). Adversarial machine learning at scale. CoRR, abs/1611.01236. URL http://arxiv.org/abs/1611.01236, arXiv:1611.01236.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436.

Lyu, K., & Li, J. (2020). Gradient descent maximizes the margin of homogeneous neural networks. In International conference on learning representations.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. arXiv e-prints, arXiv:1706.06083.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2012). Foundations of machine learning. MIT Press.

Moroshko, E., Gunasekar, S., Woodworth, B., Lee, J. D., Srebro, N., & Soudry, D. (2020). Implicit bias in deep linear classification: Initialization scale vs training accuracy. arXiv preprint arXiv:2007.06738.

Papernot, N., McDaniel, P. D., Sinha, A., & Wellman, M. P. (2016). Towards the science of security and privacy in machine learning. CoRR, abs/1611.03814. URL http://arxiv.org/abs/1611.03814, arXiv:1611.03814.

Raghunathan, A., Xie, S. M., Yang, F., Duchi, J. C., & Liang, P. (2019). Adversarial training can hurt generalization. arXiv e-prints, arXiv:1906.06032.

Rice, L., Wong, E., & Kolter, Z. (2020). Overfitting in adversarially robust deep learning. In H. D. III, & A. Singh (Eds.), Proceedings of machine learning research: Vol. 119, Proceedings of the 37th international conference on machine learning (pp. 8093–8104). PMLR, URL http://proceedings.mlr.press/v119/rice20a.html.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., & Madry, A. (2018). Adversarially robust generalization requires more data. CoRR, abs/1804.11285. URL http://arxiv.org/abs/1804.11285, arXiv:1804.11285.

Sharif, M., Bhagavatula, S., Bauer, L., & Reiter, M. K. (2017). Adversarial generative nets: Neural network attacks on state-of-the-art face recognition. arXiv preprint arXiv:1801.00349.

Shwartz-Ziv, R., & Tishby, N. (2017). Opening the black box of deep neural networks via information. CoRR, abs/1703.00810. URL http://arxiv.org/abs/1703.00810, arXiv:1703.00810.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., et al. (2013). Intriguing properties of neural networks. arXiv e-prints, arXiv:1312.6199.

Tishby, N., & Zaslavsky, N. (2015). Deep learning and the information bottleneck principle. CoRR, abs/1503.02406. URL http://arxiv.org/abs/1503.02406, arXiv:1503.02406.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., & Madry, A. (2018). Robustness may be at odds with accuracy. arXiv e-prints, arXiv:1805.12152.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., et al. (2019). Manifold mixup: Better representations by interpolating hidden states. In K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of machine learning research: Vol. 97, Proceedings of the 36th international conference on machine learning (pp. 6438–6447). Long Beach, California, USA: PMLR, URL http://proceedings.mlr.press/v97/verma19a.html.

Verma, V., Lamb, A., Kannala, J., Bengio, Y., & Lopez-Paz, D. (2019). Interpolation consistency training for semi-supervised learning. arXiv e-prints, arXiv:1903.03825.

Verma, V., Qu, M., Lamb, A., Bengio, Y., Kannala, J., & Tang, J. (2019). Graph-mix: Regularized training of graph neural networks for semi-supervised learning. CoRR, abs/1909.11715. URL http://arxiv.org/abs/1909.11715, arXiv:1909.11715.

Wong, E., Schmidt, F., Metzen, J. H., & Kolter, J. Z. (2018). Scaling provable adversarial defenses. CoRR, abs/1805.12514. URL http://arxiv.org/abs/1805.12514, arXiv:1805.12514.

Woodworth, B., Gunasekar, S., Lee, J. D., Moroshko, E., Savarese, P., Golan, I., et al. (2020). Kernel and rich regimes in overparametrized models. arXiv preprint arXiv:2002.09277.

Yin, D., Ramchandran, K., & Bartlett, P. (2018). Rademacher complexity for adversarially robust generalization. CoRR, abs/1810.11914. URL http://arxiv.org/abs/1810.11914, arXiv:1810.11914.

Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. CoRR, abs/1605.07146. URL http://arxiv.org/abs/1605.07146, arXiv:1605.07146.

Zhai, R., Cai, T., He, D., Dan, C., He, K., Hopcroft, J., et al. (2019). Adversarially robust generalization just requires more unlabeled data. arXiv e-prints, arXiv:1906.00555.

Zhang, H., Cissé, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). Mixup: Beyond empirical risk minimization. CoRR, abs/1710.09412. URL http://arxiv.org/abs/1710.09412, arXiv:1710.09412.

Zhang, L., Deng, Z., & Kawaguchi, K. (2021). How does mixup help with robustness and generalization? In *International conference on learning representations (ICLR)*.

Zhang, C., Hsieh, M., & Tao, D. (2018). Generalization bounds for vicinal risk minimization principle. CoRR, abs/1811.04351. URL http://arxiv.org/abs/1811.04351, arXiv:1811.04351.

Zhang, H., & Wang, J. (2019). Defense against adversarial attacks using feature scattering-based adversarial training. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, & R. Garnett (Eds.), *Vol. 32, Advances in neural information processing systems*. Curran Associates, Inc., URL https://proceedings.neurips.cc/paper/2019/file/d8700cbd38cc9f30cecb34f0c195b137-Paper.pdf.

Zhang, H., Yu, Y., Jiao, J., Xing, E., Ghaoui, L. E., & Jordan, M. (2019a). Theoretically principled trade-off between robustness and accuracy. In K. Chaudhuri, & R. Salakhutdinov (Eds.), *Proceedings of machine learning research*: *Vol. 97*, *Proceedings of the 36th international conference on machine learning* (pp. 7472–7482). Long Beach, California, USA: PMLR, URL http://proceedings.mlr.press/v97/zhang19p.html.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., & Jordan, M. I. (2019b). Theoretically principled trade-off between robustness and accuracy. CoRR, abs/1901.08573. URL http://arxiv.org/abs/1901.08573, arXiv:1901.08573.

Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. CoRR, abs/1611.01578. URL http://arxiv.org/abs/1611.01578, arXiv:1611.01578.