Overfitting in adversarially robust deep learning

Leslie Rice [*1] **Eric Wong** [*2] **J. Zico Kolter** [1]

Abstract

It is common practice in deep learning to use overparameterized networks and train for as long as possible; there are numerous studies that show, both theoretically and empirically, that such practices surprisingly do not unduly harm the generalization performance of the classifier. In this paper, we empirically study this phenomenon in the setting of *adversarially trained deep networks*, which are trained to minimize the loss under worst-case adversarial perturbations. We find that overfitting to the training set does in fact harm robust performance to a very large degree in adversarially robust training across multiple datasets (SVHN, CIFAR-10, CIFAR-100, and ImageNet) and perturbation models (ℓ_∞ and ℓ_2). Based upon this observed effect, we show that the performance gains of virtually all recent algorithmic improvements upon adversarial training can be matched by simply using early stopping. We also show that effects such as the double descent curve *do* still occur in adversarially trained models, yet fail to explain the observed overfitting. Finally, we study several classical and modern deep learning remedies for overfitting, including regularization and data augmentation, and find that no approach in isolation improves significantly upon the gains achieved by early stopping. All code for reproducing the experiments as well as pretrained model weights and training logs can be found at `https://github.com/locuslab/robust_overfitting`.

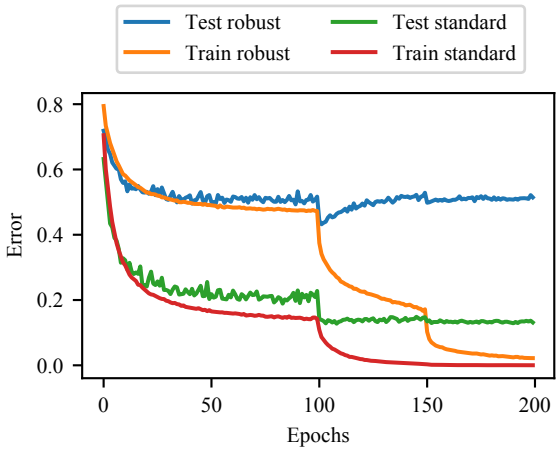

Figure 1. The learning curves for a robustly trained model replicating the experiment done by Madry et al. (2017) on CIFAR-10. The curves demonstrate "robust overfitting"; shortly after the first learning rate decay the model momentarily attains 43.2% robust error, and is actually more robust than the model at the end of training, which only attains 51.4% robust test error against a 10-step PGD adversary for ℓ_∞ radius of $\epsilon = 8/255$. The learning rate is decayed at 100 and 150 epochs.

1. Introduction

One of the surprising characteristics of deep learning is the relative *lack* of overfitting seen in practice (Zhang et al., 2016). Deep learning models can often be trained to zero training error, effectively memorizing the training set, seemingly without causing any detrimental effects on the generalization performance. This phenomenon has been widely studied both from the theoretical (Neyshabur et al., 2017) and empirical perspectives (Belkin et al., 2019), and remains such a hallmark of deep learning practice that it is often taken for granted.

In this paper, we consider the empirical question of overfitting in a similar, but slightly different domain: the setting of *adversarial training* for robust networks. Adversarial training is a method for hardening classifiers against adversarial attacks, i.e. small perturbations to the input which can drastically change a classifier's predictions, that involves training the network on adversarially perturbed inputs instead of on clean data (Goodfellow et al., 2014). It is generally regarded as one of the strongest empirical defenses against these attacks (Madry et al., 2017).

A key finding of our paper is that, unlike in traditional deep learning, *overfitting is a dominant phenomenon in*

[*]Equal contribution [1]Computer Science Department, Carnegie Mellon University, Pittsburgh PA, USA [2]Machine Learning Department, Carnegie Mellon University, Pittsburgh PA, USA. Correspondence to: Leslie Rice <larice@cs.cmu.edu>, Eric Wong <ericwong@cs.cmu.edu>.

Preprint.

adversarially robust training of deep networks. That is, adversarially robust training has the property that, after a certain point, further training will continue to substantially decrease the robust training loss of the classifier, while increasing the robust test loss. This is shown, for instance, in Figure 1 for adversarial training on CIFAR-10, where the robust test error dips immediately after the first learning rate decay, and only increases beyond this point. We show that this phenomenon, which we refer to as "robust overfitting", can be observed on multiple datasets beyond CIFAR-10, such as SVHN, CIFAR-100, and ImageNet.

Motivated by this initial finding, we make several contributions in this paper to further study and diagnose this problem. First, we emphasize that virtually all the recent gains in adversarial performance from newer algorithms beyond simple projected gradient descent (PGD) based adversarial training (Mosbach et al., 2018; Xie et al., 2019; Yang et al., 2019; Zhang et al., 2019c) can be attained by a much simpler approach: using early stopping. Specifically, by just using an earlier checkpoint, the robust performance of adversarially trained deep networks can be drastically improved, to the point where *the original PGD-based adversarial training method can actually achieve the same robust performance as state-of-the-art methods*. For example, vanilla PGD-based adversarial training (Madry et al., 2017) can achieve 43.2% robust test error against a PGD adversary with ℓ_∞ radius 8/255 on CIFAR-10 when training is stopped early, on par with the 43.4% robust test error reported by TRADES (Zhang et al., 2019c) against the same adversary. This phenomenon is not unique to ℓ_∞ perturbations and is also seen in ℓ_2 adversarial training. For instance, early stopping a CIFAR-10 model trained against an ℓ_2 adversary with radius 128/255 can decrease the robust test error from 31.1% to 28.4%.

Second, we study various empirical properties of overfitting for adversarially robust training and how they relate to standard training. Since the effects of such overfitting appear closely tied to the learning rate schedule, we begin by investigating how changes to the learning rate schedule affect the prevalence of robust overfitting and its impacts on model performance. We next explore how known connections between the hypothesis class size and generalization in deep networks translate to the robust setting, and show that the "double descent" generalization curves seen in standard training (Belkin et al., 2019) also hold for robust training (Nakkiran et al., 2019). However, although this is used as a justification for the lack of overfitting in the standard setting, surprisingly, changing the hypothesis class size does not actually mitigate the robust overfitting that is observed during training.

Our final contribution is to investigate several techniques for preventing robust overfitting. We first explore the effects of classic statistical approaches for combating overfitting beyond early stopping, namely explicit ℓ_1 and ℓ_2 regularization. We then study more modern approaches using data augmentation, including cutout (DeVries & Taylor, 2017), mixup (Zhang et al., 2017), and semisupervised learning methods, which are known to empirically reduce overfitting in deep networks. Ultimately, while these methods can mitigate robust overfitting to varying degrees, when trained to convergence, *we find that no other approach to combating robust overfitting performs better than simple early stopping*. In fact, even combining regularization methods with early stopping tends to not significantly improve on early stopping alone. We find that the one exception is data augmentation with semi-supervised learning, where although the test performance can vary wildly even when training has converged, at select epochs it is possible to find a model with improved robust performance over simple early stopping. Code for reproducing all the experiments in this paper along with pretrained model weights and training logs can be found at `https://github.com/locuslab/robust_overfitting`.[1]

2. Background and related work

One of the first approaches to using adversarial training was with a single step gradient-based method for generating adversarial examples known as the fast gradient sign method (FGSM) (Goodfellow et al., 2014). The adversary was later extended to take multiple smaller steps, in a technique known as the basic iterative method (Kurakin et al., 2016), and eventually reincorporated into adversarial training with random restarts, commonly referred to as projected gradient descent (PGD) adversarial training (Madry et al., 2017). Further improvements to both the PGD adversary and the training procedure include incorporating momentum into the adversary (Dong et al., 2018), leveraging matrix estimation (Yang et al., 2019), logit pairing (Mosbach et al., 2018), and feature denoising (Xie et al., 2019). Most notably, Zhang et al. (2019c) proposed the method TRADES for adversarial training that balances the trade-off between standard and robust errors, and achieves state-of-the-art performance on several benchmarks.

Because PGD training is significantly more time consuming than standard training, several works have focused on improving the efficiency of adversarial training by reducing the computational complexity of calculating gradients and reducing the number of attack iterations (Shafahi et al., 2019; Zhang et al., 2019a; Wong et al., 2020). Separate works have also expanded the general PGD adversarial training

[1] Since there are over 75 models trained in this paper, we selected a subset of pretrained models to release (e.g. those which are for Wide ResNets since those take the most time to train, and can achieve the best performance in the paper)

algorithm to different threat models including image transformations (Engstrom et al., 2017; Xiao et al., 2018a), different distance metrics (Wong et al., 2019), and multiple threat models (Maini et al., 2019; Tramèr & Boneh, 2019).

Other adversarial defenses that have been proposed were not always successful, such as distillation (Papernot et al., 2016; Carlini & Wagner, 2017b) and detection of adversarial examples (Metzen et al., 2017; Feinman et al., 2017; Carlini & Wagner, 2017a; Tao et al., 2018; Carlini, 2019), which eventually were defeated by stronger attacks. Adversarial examples were also believed to be ineffective in the real world across different viewpoints (Lu et al., 2017) until proven otherwise (Athalye et al., 2017), and a large number of adversarial defenses were shown to be relying on obfuscated gradients and ultimately rendered ineffective (Athalye et al., 2018), including thermometer encoding (Buckman et al., 2018) and various preprocessing techniques (Guo et al., 2017; Song et al., 2017).

Because many defenses were "broken" by stronger adversaries, a separate but related line of work has looked at generating certificates which can guarantee or prove robustness of the network output to norm-bounded adversarial perturbations. While not always scalable to large convolutional networks, methods for generating these robustness certificates range from using Satisfiability Modulo Theories (SMT) solvers (Ehlers, 2017; Huang et al., 2017; Katz et al., 2017) and mixed-integer linear programs (Tjeng et al., 2019) for exact certificates, to semi-definite programming (SDP) solvers for relaxed but still accurate certificates (Raghunathan et al., 2018a;b; Fazlyab et al., 2019). Other methods focus on generating more tractable but relaxed certificates, which provide looser guarantees but can be optimized during training. These methods leverage techniques such as duality and linear programming (Wong & Kolter, 2017; Dvijotham et al.; Wong et al., 2018; Salman et al., 2019b; Zhang et al., 2019b), randomized smoothing (Cohen et al., 2019; Lecuyer et al., 2019; Salman et al., 2019a), distributional robustness (Sinha et al., 2017), abstract interpretations (Gehr et al., 2018; Mirman et al., 2018; Singh et al., 2018), and interval bound propagation (Gowal et al., 2018). Another approach is to use theoretically justified training heuristics (Croce et al., 2018; Xiao et al., 2018b) which result in models which are verifiable by an independent certification method.

Highly relevant to this work are those that study the general problem of overfitting in machine learning. Both regularization (Friedman et al., 2001) and early stopping (Strand, 1974) have been well-studied in classical statistical settings to reduce overfitting and improve generalization, and connections between the two have been established in various settings such as in kernel boosting algorithms (Wei et al., 2017), least squares regression (Ali et al., 2018), and strongly convex problems (Suggala et al., 2018). Although ℓ_2 regularization (also known as weight decay) is commonly used for training deep networks (Krogh & Hertz, 1992), early stopping is less commonly used despite being studied as an implicit regularizer for controlling model complexity for neural networks at least 30 years ago (Morgan & Bourlard, 1990).[2] Indeed, it is now known that the standard bias-variance trade-off from classical statistical learning theory fails to explain why deep networks can generalize so well (Zhang et al., 2016). Consequently, it is now standard practice in many modern deep learning tasks to train for as long as possible and use large overparameterized models, since test set performance typically continues to improve past the point of dataset interpolation in what is known as "double descent" generalization (Belkin et al., 2019; Nakkiran et al., 2019). The generalization gap for robust deep networks has also been studied from a learning theoretic perspective in the context of data complexity (Schmidt et al., 2018) and Rademacher complexity (Yin et al., 2018).

Also relevant to this work are methods specific to deep learning that empirically reduce overfitting and improve performance of deep networks. For example, Dropout is a commonly used stochastic regularization technique that randomly drops units and their connections from the network during training (Srivastava et al., 2014) with the intent of preventing complex co-adaptations on the training data. Data augmentation is another technique frequently used when training deep networks that has been empirically shown to reduce overfitting. Cutout (DeVries & Taylor, 2017) is a form of data augmentation that randomly masks out a section of the input during training, which can be considered as augmenting the dataset with occlusions. Another technique known as mixup (Zhang et al., 2017) trains on convex combinations of pairs of data points and their corresponding labels to encourage linear behavior in between data points. Semi-supervised learning methods augment the dataset with unlabeled data, and have been shown to improve generalization when used in the adversarially robust setting (Carmon et al., 2019; Zhai et al., 2019; Alayrac et al., 2019).

3. Adversarial training and robust overfitting

In order to learn networks that are robust to adversarial examples, a commonly used method is adversarial training, which solves the following robust optimization problem

$$\min_{\theta} \sum_{i} \max_{\delta \in \Delta} \ell(f_\theta(x_i + \delta), y_i), \tag{1}$$

[2] It is common practice in deep learning to save the best checkpoint which can be seen as early stopping. However, in the standard setting, the test loss tends to gradually improve over training, and so the best checkpoint tends to just select the best performance at the end of training, rather than stopping before training loss has converged.

Table 1. Robust performance showing the occurrence of robust overfitting across datasets and perturbation threat models. The "best" robust test error is the lowest test error observed during training. The final robust test error is averaged over the last five epochs. The difference between final and best robust test error indicates the degradation in robust performance during training.

			Robust test error (%)		
Dataset	Norm	Radius	Final	Best	Diff
SVHN	ℓ_∞	$8/255$	45.6 ± 0.40	39.0	6.6
	ℓ_2	$128/255$	26.4 ± 0.27	25.2	1.2
CIFAR-10	ℓ_∞	$8/255$	51.4 ± 0.41	43.2	8.2
	ℓ_2	$128/255$	31.1 ± 0.46	28.4	2.7
CIFAR-100	ℓ_∞	$8/255$	78.6 ± 0.39	71.9	6.7
	ℓ_2	$128/255$	62.5 ± 0.09	56.8	5.7
ImageNet	ℓ_∞	$4/255$	85.5 ± 8.87	62.7	22.8
	ℓ_2	$76/255$	94.8 ± 1.16	63.0	31.8

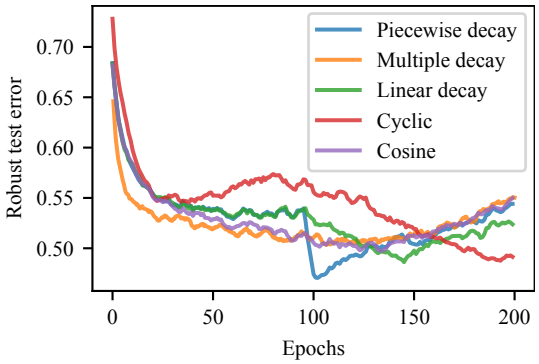

Figure 2. Robust test error over training epochs for various learning rate schedules on CIFAR-10. None of the alternative smoother learning rate schedules can achieve a peak performance competitive with the standard piecewise decay learning rate, indicating that the peak performance is obtained by having a single discrete jump. Note that the multiple decay schedule is actually run for 500 epochs, but compressed into this plot for a clear comparison.

where f_θ is a network with parameters θ, (x_i, y_i) is a training example, ℓ is the loss function, and Δ is the perturbation set. Typically the perturbation set Δ is chosen to be an ℓ_p-norm ball (e.g. ℓ_2 and ℓ_∞ perturbations, which we consider in this paper), such that $\Delta = \{\delta : ||\delta||_p \leq \epsilon\}$ for $\epsilon > 0$. Adversarial training approximately solves the inner optimization problem, also known as the robust loss, using some adversarial attack method, typically with projected gradient descent (PGD), and then updates the model parameters θ using gradient descent (Madry et al., 2017). For example, an ℓ_∞ PGD adversary would start at some random initial perturbation $\delta^{(0)}$ and iteratively adjust the perturbation with the following ℓ_∞ gradient steps while projecting back onto the ℓ_∞ ball with radius ϵ:

$$\begin{aligned} \tilde{\delta} &= \delta^{(t)} + \alpha \cdot \text{sign}\nabla_x \ell(f(x), y)) \\ \delta^{(t+1)} &= \max(\min(\tilde{\delta}, \epsilon), -\epsilon) \end{aligned} \tag{2}$$

We denote error rates when attacked by a PGD adversary as the "robust error", and error rates on the clean, unperturbed data as "standard error".

3.1. Robust overfitting: a general phenomenon for adversarially robust deep learning

In the standard, non-robust deep learning setting, it is common practice to train for as long as possible to minimize the training loss, as modern convergence curves for deep learning generally observe that the testing loss continues to decrease with the training loss. On the contrary, for the setting of adversarially robust training we make the following discovery:

Unlike the standard setting of deep networks, overfitting for adversarially robust training can result in worse test set performance.

This phenomenon, which we refer to as "robust overfitting", results in convergence curves as shown earlier in Figure 1. Although training appears normal in the earlier stages, after the learning rate decays, the robust test error briefly decreases but begins to increase as training progresses. This behavior indicates that the optimal performance is not obtained at the end of training, unlike in standard training for deep networks.

We find that robust overfitting occurs across a variety of datasets, algorithmic approaches, and perturbation threat models, indicating that it is a general property of the adversarial training formulation and not specific to a particular problem, as can be seen in Table 1 for ℓ_∞ and ℓ_2 perturbations on SVHN, CIFAR-10, CIFAR-100, and ImageNet. A more detailed and expanded version of this table summarizing the full extent of robust overfitting as well as the corresponding learning curves for each setting can be found in Appendix A. We consistently find that there is a significant gap between the best robust test performance during training and the final robust test performance at the end of training, observing an increase of 8.2% robust error for CIFAR-10 and 22.8% robust error for ImageNet against an ℓ_∞ adversary, to highlight a few. Robust overfitting is also not specific to PGD-based adversarial training, and affects faster adversarial training methods such as FGSM adversarial training[3] (Wong et al., 2020) as well as top performing algorithms for adversarially robust training such as TRADES (Zhang et al., 2019c).

[3] Wong et al. (2020) also observe a different form of overfitting specifically for FGSM adversarial training which they refer to as "catastrophic overfitting". This is separate behavior from the robust overfitting described in this paper, and the specifics of this distinction are discussed further in Appendix A.4.

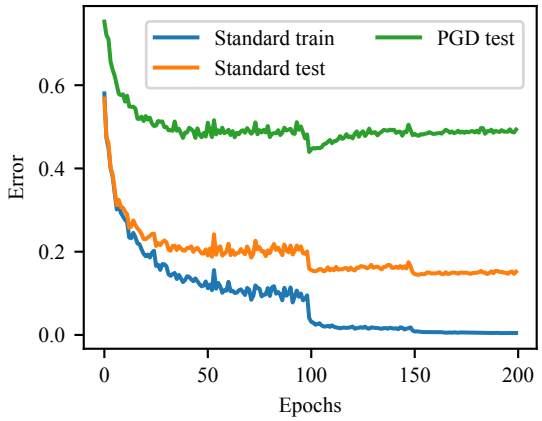

Figure 3. Learning curves showing standard and robust error rates for a Wide ResNet model trained with TRADES on CIFAR-10. Early stopping after the initial learning rate decay is crucial in order to achieve the 43.4% robust test error reported by Zhang et al. (2019c), which eventually degrades to 50.6% robust test error when the training has converged.

Learning rate schedules and robust overfitting Since the change in performance appears to be closely linked with the first drop in the scheduled learning rate decay, we explore how different learning rate schedules affect robust overfitting on CIFAR-10, as shown in Figure 2, with complete descriptions of the various learning rate schedules in Appendix B.1. In summary, we find that smoother learning rate schedules (which take smaller decay steps or interpolate the change in learning rate over epochs) simply result in smoother curves that still exhibit robust overfitting. Furthermore, with each smoother learning rate schedule, the best robust test performance during training is strictly worse than the best robust test performance during training with the discrete piecewise decay schedule. In fact, the parameters of the discrete piecewise decay schedule can even be tuned to slightly exacerbate the sudden improvement in performance after the first learning rate decay step, which we discuss further in Appendix B.2

3.2. Mitigating robust overfitting with early stopping

Proper early stopping, an old form of implicit regularization, calculates a metric on a hold-out validation set to determine when to stop training in order to prevent overfitting. Since the test performance does not monotonically improve during adversarially robust training due to robust overfitting, it is advantageous for robust networks to use early stopping to achieve the best possible robust performance.

We find that, for example, the TRADES approach relies heavily on using the best robust performance on the test set from an earlier checkpoint in order to achieve their top reported result of 43.4% robust error against an ℓ_∞ PGD adversary with radius $8/255$ on CIFAR-10, a number which is typically viewed as a substantial algorithmic improvement in adversarial robustness over standard PGD-based adversarial training. In our own reproduction of the TRADES experiment, we confirm that allowing the TRADES algorithm to train until convergence results in significant degradation of robust performance as seen in Figure 3. Specifically, the robust test error of the model at the checkpoint with the best performance on the test set is 44.1% whereas the robust test error of the model at the end of training has increased to 50.6%.[4]

Surprisingly, when we early stop vanilla PGD-based adversarial training, selecting the model checkpoint with the best performance on the test set, we find that PGD-based adversarial training performs just as well as more recent algorithmic approaches such as TRADES. Specifically, when using the *same* architecture (a Wide ResNet with depth 28 and width factor 10) and the *same* 20-step PGD adversary for evaluation used by Zhang et al. (2019c) for TRADES, the model checkpoint with the best performance on the test set from vanilla PGD-based adversarial training achieves 42.3% robust test error, which is actually slightly better than the best reported result for TRADES from Zhang et al. (2019c).[5]

Similarly, we find early stopping to be a factor in the robust test performance for publicly released pre-trained ImageNet models (Engstrom et al., 2019). Continuing to train these models degrades the robust test performance from 62.7% to 85.5% robust test error for ℓ_∞ robustness at $\epsilon = 4/255$ and 63.0% to 94.8% robust test error for ℓ_2 robustness at $\epsilon = 128/255$. This shows that these models are also susceptible to robust overfitting and benefit greatly from early stopping.[6] The corresponding learning curves are shown in Appendix A.3.

Validation-based early stopping Early stopping based on the test set performance, however, leaks test set information and goes against the traditional machine learning paradigm. Instead, we find that it is still possible to recover the best test performance achieved during training with a true hold-out validation set. By holding out 1,000

[4]We used the public implementation of TRADES available at `https://github.com/yaodongyu/TRADES` and simply ran it to completion using the same learning rate decay schedule used by Madry et al. (2017).

[5]We found that our implementation of the PGD adversary to be slightly more effective, increasing the robust test error of the TRADES model and the PGD trained model to 45.0% and 43.2% respectively.

[6]We use the publicly available framework from `https://github.com/madrylab/robustness` and continue training checkpoints obtained from the authors using the same learning parameters.

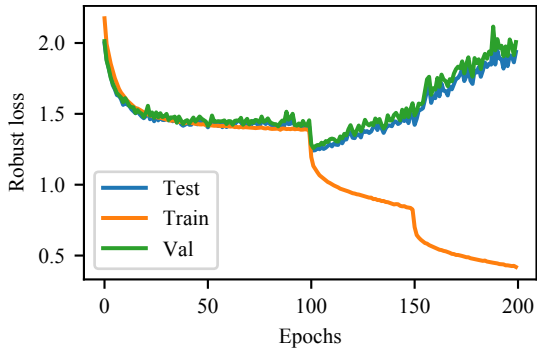

Figure 4. Learning curves for a CIFAR-10 pre-activation ResNet18 model trained with a hold-out validation set of 1,000 examples. We find that the hold-out validation set is enough to reflect the test set performance, and stopping based on the validation set is able to prevent overfitting and recover 46.9% robust test error, in comparison to 46.7% achieved by the best-performing model checkpoint.

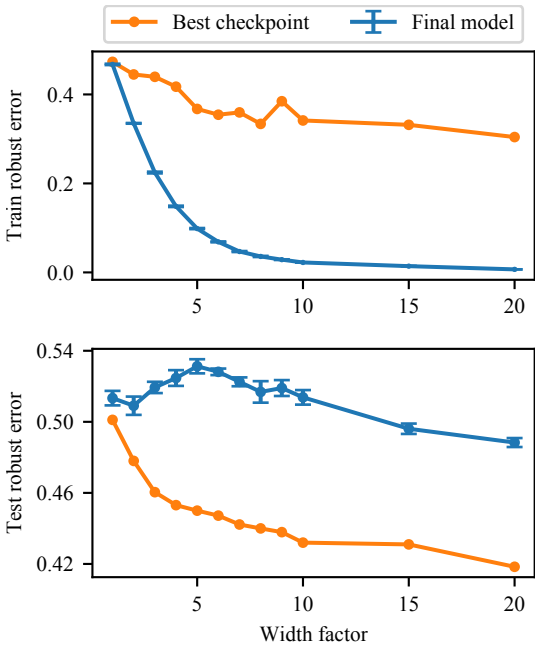

Figure 5. Generalization curves depicting double descent for adversarially robust generalization, where hypothesis class complexity is controlled by varying the width factor for a wide residual network. Each final model point represents the average performance over the last 5 epochs with the corresponding width factor from training until convergence. The best checkpoint refers to the lowest robust test error achieved by a model checkpoint during training, and illustrates the significant gap in performance between the best and final models resulting from robust overfitting.

examples from the CIFAR-10 training set for validation purposes, we use validation-based early stopping to achieve 46.9% robust error on the test set *without looking at the test set*, in comparison to the 46.7% robust error achieved by the best-performing model checkpoint for a pre-activation ResNet18. The resulting validation curve during training closely matches the testing curve as seen in Figure 4, and suggests that although robust overfitting degrades the robust test set performance, selecting the best checkpoint in adversarially robust training for deep networks still does not appear to significantly overfit to the test set (which has been previously observed in the standard, non-robust setting (Recht et al., 2018)).

3.3. Reconciling double descent curves

Modern generalization curves for deep learning typically show improved test set performance for increased model complexity beyond data point interpolation in what is known as *double descent* (Belkin et al., 2019). This suggests that overfitting by increasing model complexity using overparameterized neural networks is beneficial and improves test set performance. However, this appears to be at odds with the main findings of this paper; since training for longer can also be viewed as increasing model complexity, the fact that training for longer results in worst test set performance seems to contradict double descent.

We find that, while increasing either training time or architecture size can be viewed as increasing model complexity, these two approaches actually have separate effects; training for longer degrades the robust test set performance regardless of architecture size, while increasing the model architecture size still improves the robust test set performance despite robust overfitting. This was briefly noted by Nakkiran et al. (2019) for the ℓ_2 robust setting, and so in this section we show that this generally holds also in the ℓ_∞ robust setting. We explore these properties by training multiple adversarially robust Wide ResNets (Zagoruyko & Komodakis, 2016) with varying widths to control model complexity. In Figure 5, we see that no matter how large the model architecture is, robust overfitting still results in a significant gap between the best and final robust test performance. However, we also see that adversarially robust training still produces the double descent generalization curve, as the robust test performance increases and then decreases again with architecture size, suggesting that the double descent and robust overfitting are separate phenomenon. Even the lowest robust test error achieved during training continues to descend with increased model complexity, suggesting that larger architecture sizes are still beneficial for adversarially robust training despite robust overfitting. More details and learning curves for a wide range of architecture sizes can be found in Appendix C.

Table 2. Robust performance of PGD-based adversarial training with different regularization methods on CIFAR-10 using a PreActResNet18 for ℓ_∞ with radius $8/255$. The "best" robust test error is the lowest test error achieved during training whereas the final robust test error is averaged over the last five epochs. Each of the regularization methods listed is trained using the optimally chosen hyperparameter. Pure early stopping is done with a validation set.

	Robust Test Error (%)		
Reg Method	Final	Best	Diff
Early stopping w/ val	**46.9**	46.7	0.2
ℓ_1 regularization	53.0 ± 0.39	48.6	4.4
ℓ_2 regularization	55.2 ± 0.4	46.4	55.2
Cutout	48.8 ± 0.79	46.7	2.1
Mixup	49.1 ± 1.32	46.3	2.8
Semi-supervised	47.1 ± 4.32	40.2	6.9

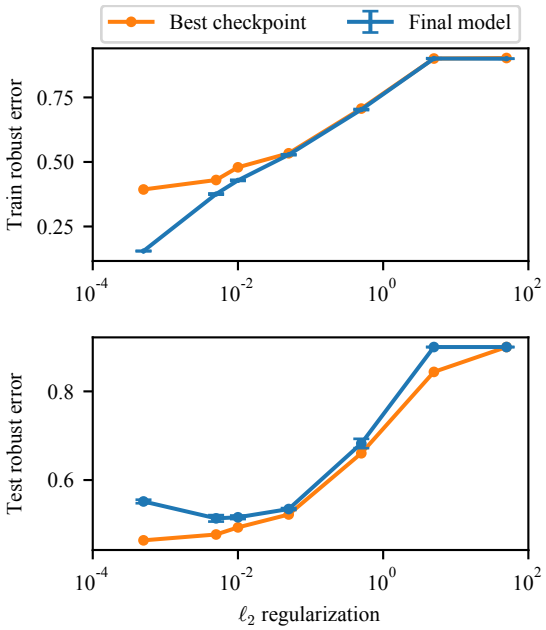

Figure 6. Robust performance on the train and test set for varying degrees of ℓ_2 regularization. ℓ_2 regularization is unable to match the same performance of early stopping without also using early stopping, even with an optimally chosen hyperparameter of $\lambda = 5 \cdot 10^{-3}$ which achieves 55.2% robust test error.

4. Alternative methods to prevent robust overfitting

In this section, we explore whether common methods for combating overfitting in standard training are successful at mitigating robust overfitting in adversarial training. We run a series of ablation studies on CIFAR-10 using classical and modern regularization techniques, yet ultimately find that no technique performs as well in isolation as early stopping, as shown in Table 2 (a more detailed table including standard error can be found in Appendix D.2). Unless otherwise stated, we begin each experiment with the standard setup for ℓ_∞ PGD-based adversarial training with a 10-step adversary with step size $2/255$ using a pre-activation ResNet18 (He et al., 2016) (details for the training procedure and the PGD adversary can be found in Appendix D.1). All experiments in this section were run with one GeForce RTX 2080ti unless a Wide ResNet was trained, in which case two GPUs were used.

4.1. Explicit regularization

A classical method for preventing overfitting is to add an explicit regularization term to the loss, penalizing the complexity of the model parameters. Specifically, the term is typically of the form $\lambda\Omega(\theta)$, where θ contains the model parameters, $\Omega(\theta)$ is some regularization penalty, and λ is a hyperparameter to control the regularization effect. A typical choice for Ω is ℓ_p regularization for $p \in \{1, 2\}$, where ℓ_2 regularization is canonically known as weight decay and commonly used in standard training of deep networks, and ℓ_1 regularization is known to induce sparsity properties.

We explore the effects of using ℓ_1 and ℓ_2 regularization when training robust networks on robust overfitting, and sweep across a range of hyperparameter values as seen in Figure 6 for ℓ_2.[7] Although explicit regularization does improve the performance to some degree, on its own, it is still not as effective as early stopping, with the best explicit regularizer achieving 55.2% robust test error with ℓ_2 regularization and parameter $\lambda = 5 \cdot 10^{-2}$. Additionally, neither of these regularization techniques can completely remove the detrimental effects of robust overfitting without drastically over-regularizing the model, which is shown and discussed further in Appendix D.3, along with the corresponding plots for ℓ_1 regularization.

4.2. Data augmentation for deep learning

Data augmentation has been empirically shown to reduce overfitting in modern deep learning tasks that involve very high-dimensional data by enhancing the quantity and diversity of the training data. Such techniques range from simple augmentations like random cropping and horizontal flipping to more recent approaches leveraging unlabeled data for semi-supervised learning, and some work has argued that robust deep learning requires more data than standard deep learning (Schmidt et al., 2018).

[7] Proper parameter regularization only applies the penalty to the weights w of the affine transformations at each layer, excluding the bias terms and batch normalization parameters.

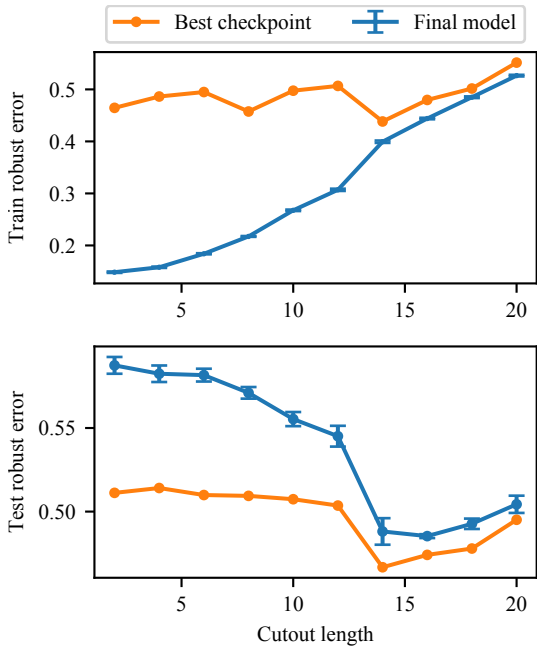

Figure 7. Robust performance on the train and test set with cutout across varying patch lengths. Even with the optimal patch length of 14, cutout does not surpass the performance of early stopping, achieving at best 48.8% robust test error at the end of training.

Cutout and mixup Recent data augmentations techniques for deep networks, such as cutout (DeVries & Taylor, 2017) and mixup (Zhang et al., 2017), are known to reduce overfitting and improve generalization in the standard training setting. We scan a range of hyperparameters for these approaches when applicable, and find a similar story to that of explicit ℓ_p regularization; either the regularization effect of cutout and mixup is too low to prevent robust overfitting, or too high and the model is over-regularized, as seen in Figures 7 for cutout. When trained to convergence, neither cutout nor mixup is as effective as early stopping, achieving at best 48.8% robust test error for cutout with a patch length of 14 and 49.1% robust test error for mixup with $\alpha = 1.4$. [8] The corresponding plots for mixup and the learning curves for both methods are in Appendix D.4, where we see significant robust overfitting cutout but less so for mixup, which appears to be more regularized.

Semi-supervised learning We additionally consider a semi-supervised data augmentation technique (Carmon et al., 2019; Zhai et al., 2019; Alayrac et al., 2019) which uses a standard classifier to label unlabeled data for use in robust training. Although there is a large gap between best

[8]We used the public implementations of cutout and mixup available at https://github.com/davidcpage/cifar10-fast and https://github.com/facebookresearch/mixup-cifar10

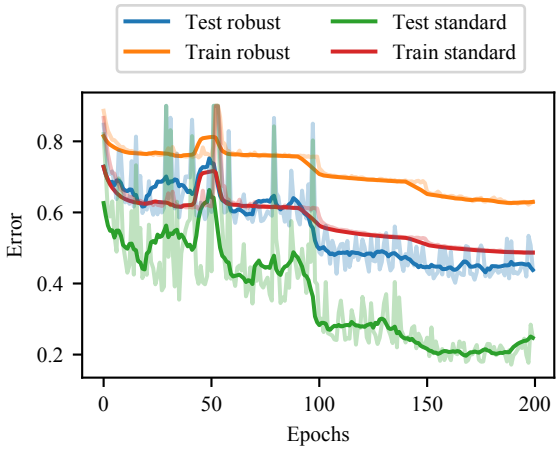

Figure 8. Learning curves for robust training with semi-supervised data augmentation, where we do not see a severe case of robust overfitting. When robust training error has converged, there is a significant amount of variance in the robust test error, so the average final model performance is on par with pure early stopping. Combining early stopping with semi-supervised data augmentation to avoid this variance is the only method we find that significantly improves on pure early stopping, reaching 40.2% robust test error.

and final robust performance shown in Table 2, we find that this is primarily driven by high variance in the robust test error during training rather than from robust overfitting, even when the model has converged as seen in Figure 8. Due to this variance, the final model's average robust performance of 47.1% robust test error is similar to the performance obtained by early stopping. By combining early stopping with semi-supervised data augmentation, this variance can be avoided. In fact, we find that the combination of early stopping and semi-supervised data augmentation is the only method that results in significant improvement over early stopping alone, resulting in 40.2% robust test error. Experimental details and further discussion for this approach can be found in Appendix E. [9]

5. Conclusion

Unlike in standard training, overfitting in robust adversarial training decays test set performance during training in a wide variety of settings. While overfitting with larger architecture sizes results in better test set generalization, it does not reduce the effect of robust overfitting. Our extensive suite of experiments testing the effect of implicit and explicit regularization methods on preventing overfitting found that most of these techniques tend to over-regularize the model or do not prevent robust overfitting, and all of

[9]We used the data from https://github.com/yaircarmon/semisup-adv containing 500K pseudo-labeled TinyImages

them in isolation do not improve upon early stopping.

Especially due to the prevalence of robust overfitting in adversarial training, we particularly urge the community to use validation sets when performing model selection in this regime, and to analyze the learning curves of their models. This work exposes a key difference in generalization properties between standard and robust training, which is not fully explained by either classic statistics or modern deep learning, and re-establishes the competitiveness of the simplest adversarial training baseline.

References

Alayrac, J.-B., Uesato, J., Huang, P.-S., Fawzi, A., Stanforth, R., and Kohli, P. Are labels required for improving adversarial robustness? In *Advances in Neural Information Processing Systems*, pp. 12192–12202, 2019.

Ali, A., Kolter, J. Z., and Tibshirani, R. J. A continuous-time view of early stopping for least squares regression. *arXiv preprint arXiv:1810.10082*, 2018.

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Buckman, J., Roy, A., Raffel, C., and Goodfellow, I. Thermometer encoding: One hot way to resist adversarial examples. 2018.

Carlini, N. Is ami (attacks meet interpretability) robust to adversarial examples? *arXiv preprint arXiv:1902.02322*, 2019.

Carlini, N. and Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14. ACM, 2017a.

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

Carmon, Y., Raghunathan, A., Schmidt, L., Liang, P., and Duchi, J. C. Unlabeled data improves adversarial robustness. *arXiv preprint arXiv:1905.13736*, 2019.

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

Croce, F., Andriushchenko, M., and Hein, M. Provable robustness of relu networks via maximization of linear regions. *arXiv preprint arXiv:1810.07481*, 2018.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, 2018.

Dvijotham, K., Stanforth, R., Gowal, S., Mann, T. A., and Kohli, P. A dual approach to scalable verification of deep networks.

Ehlers, R. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pp. 269–286. Springer, 2017.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. A rotation and a translation suffice: Fooling cnns with simple transformations. *arXiv preprint arXiv:1712.02779*, 2017.

Engstrom, L., Ilyas, A., Santurkar, S., and Tsipras, D. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness.

Fazlyab, M., Morari, M., and Pappas, G. J. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *arXiv preprint arXiv:1903.01287*, 2019.

Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

Friedman, J., Hastie, T., and Tibshirani, R. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., and Vechev, M. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Mann, T., and Kohli, P. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Guo, C., Rana, M., Cisse, M., and Van Der Maaten, L. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*, pp. 3–29. Springer, 2017.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

Krogh, A. and Hertz, J. A. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pp. 950–957, 1992.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 656–672. IEEE, 2019.

Lu, J., Sibai, H., Fabry, E., and Forsyth, D. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv preprint arXiv:1707.03501*, 2017.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Maini, P., Wong, E., and Kolter, J. Z. Adversarial robustness against the union of multiple perturbation models. *arXiv preprint arXiv:1909.04068*, 2019.

Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pp. 3575–3583, 2018.

Morgan, N. and Bourlard, H. Generalization and parameter estimation in feedforward nets: Some experiments. In *Advances in neural information processing systems*, pp. 630–637, 1990.

Mosbach, M., Andriushchenko, M., Trost, T., Hein, M., and Klakow, D. Logit pairing methods can fool gradient-based attacks. *arXiv preprint arXiv:1810.12042*, 2018.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597. IEEE, 2016.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018a.

Raghunathan, A., Steinhardt, J., and Liang, P. S. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, pp. 10877–10887, 2018b.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*, 2018.

Salman, H., Yang, G., Li, J., Zhang, P., Zhang, H., Razenshteyn, I., and Bubeck, S. Provably robust deep learning via adversarially trained smoothed classifiers. *arXiv preprint arXiv:1906.04584*, 2019a.

Salman, H., Yang, G., Zhang, H., Hsieh, C.-J., and Zhang, P. A convex relaxation barrier to tight robustness verification of neural networks. In *Advances in Neural Information Processing Systems*, pp. 9832–9842, 2019b.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pp. 5014–5026, 2018.

Shafahi, A., Najibi, M., Ghiasi, A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems*, pp. 10802–10813, 2018.

Sinha, A., Namkoong, H., and Duchi, J. Certifying some distributional robustness with principled adversarial training. *arXiv preprint arXiv:1710.10571*, 2017.

Smith, L. N. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472. IEEE, 2017.

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Strand, O. N. Theory and methods related to the singular-function expansion and landwebers iteration for integral equations of the first kind. *SIAM Journal on Numerical Analysis*, 11(4):798–825, 1974.

Suggala, A., Prasad, A., and Ravikumar, P. K. Connecting optimization and regularization paths. In *Advances in Neural Information Processing Systems*, pp. 10608–10619, 2018.

Tao, G., Ma, S., Liu, Y., and Zhang, X. Attacks meet interpretability: Attribute-steered detection of adversarial samples. In *Advances in Neural Information Processing Systems*, pp. 7717–7728, 2018.

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

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

Wei, Y., Yang, F., and Wainwright, M. J. Early stopping for kernel boosting algorithms: A general analysis with localized complexities. In *Advances in Neural Information Processing Systems*, pp. 6065–6075, 2017.

Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017.

Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems*, pp. 8400–8409, 2018.

Wong, E., Schmidt, F. R., and Kolter, J. Z. Wasserstein adversarial examples via projected sinkhorn iterations. *arXiv preprint arXiv:1902.07906*, 2019.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJx040EFvH.

Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., and Song, D. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018a.

Xiao, K. Y., Tjeng, V., Shafiullah, N. M., and Madry, A. Training for faster adversarial robustness verification via inducing relu stability. *arXiv preprint arXiv:1809.03008*, 2018b.

Xie, C., Wu, Y., Maaten, L. v. d., Yuille, A. L., and He, K. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 501–509, 2019.

Yang, Y., Zhang, G., Katabi, D., and Xu, Z. Me-net: Towards effective adversarial robustness with matrix estimation. *arXiv preprint arXiv:1905.11971*, 2019.

Yin, D., Ramchandran, K., and Bartlett, P. Rademacher complexity for adversarially robust generalization. *arXiv preprint arXiv:1810.11914*, 2018.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhai, R., Cai, T., He, D., Dan, C., He, K., Hopcroft, J., and Wang, L. Adversarially robust generalization just requires more unlabeled data. *arXiv preprint arXiv:1906.00555*, 2019.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Zhang, D., Zhang, T., Lu, Y., Zhu, Z., and Dong, B. You only propagate once: Painless adversarial training using maximal principle. *arXiv preprint arXiv:1905.00877*, 2019a.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Zhang, H., Chen, H., Xiao, C., Li, B., Boning, D., and Hsieh, C.-J. Towards stable and efficient training of verifiably robust neural networks. *arXiv preprint arXiv:1906.06316*, 2019b.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019c.

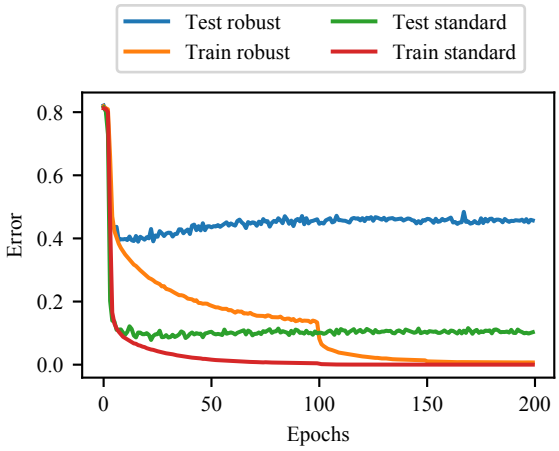

Figure 9. Learning curves for training an SVHN classifier which is adversarially robust to ℓ_∞ perturbations of radius 8/255. Note that robust overfitting occurs before the learning rate has decayed, likely due to the lower initial learning rate.

A. Full set of results for Table 1

In this section, we extend Table 1 to additionally include standard error and results from different adversarial training schemes (FGSM and TRADES), as shown in Table 3. The final error is an average over the final 5 epochs of when the model has converged, along with the standard deviation. The best error is the lowest test error of all model checkpoints during training. For convenience we also show the difference in the final model's error and the best model's error, which indicates the amount of degradation incurred by robust overfitting.

The remainder of this section contains the experimental details for reproducing these experiments, as well as the learning curves for each experiment as visual evidence of robust overfitting. We default to using pre-activation ResNet18s for our experiments, with the exception of Wide ResNets with width factor 10 for ℓ_∞ adversaries on CIFAR-10 (for a proper comparison to what is reported for TRADES), and ResNet50s for ImageNet. For CIFAR-10 and CIFAR-100, we train with the SGD optimizer using a batch size of 128, a step-wise learning rate decay set initially at 0.1 and divided by 10 at epochs 100 and 150, and weight decay $5 \cdot 10^{-4}$. For SVHN, we use the same parameters except with a starting learning rate of 0.01 instead. For ImageNet, we use the same learning configuration used to train the pretrained models and simply run them for longer epochs and lower learning rates using the publicly released repository available at `https://github.com/madrylab/robustness`.

ℓ_∞ adversary We consider the ℓ_∞ threat model with radius 8/255, with the PGD adversary taking 10 steps of size 2/255 on all datasets except for ImageNet. For Im-

Table 3. Performance of adversarially robust training over a variety of datasets, adversarial training algorithms, and perturbation threat models, where the best error refers to the lowest robust test error achieved during training and the final error is an average of the robust test error over the last 5 epochs. We observe robust overfitting to occur across all experiments.

				Robust Test Error (%)			Standard Test Error (%)		
Dataset	Adversary	Norm	Radius	Final	Best	Diff	Final	Best	Diff
SVHN	PGD	ℓ_∞	8/255	45.6 ± 0.40	39.0	6.6	10.0 ± 0.15	10.2	-0.2
		ℓ_2	128/255	26.4 ± 0.27	25.2	1.2	7.0 ± 0.23	7.2	-0.2
CIFAR-10	PGD	ℓ_∞	8/255	51.4 ± 0.41	43.2	8.2	13.4 ± 0.19	13.9	-0.5
		ℓ_2	128/255	31.1 ± 0.46	28.4	2.7	11.0 ± 0.08	11.3	-0.3
	FGSM	ℓ_∞	8/255	59.8 ± 0.09	53.7	6.1	12.4 ± 0.21	13.6	-1.2
		ℓ_2	128/255	31.6 ± 0.18	29.2	2.4	9.9 ± 0.16	10.5	-0.6
	TRADES	ℓ_∞	8/255	50.6 ± 0.31	45.0	5.6	14.97 ± 0.24	15.9	-0.9
		ℓ_2	128/255	58.2 ± 0.66	53.6	4.6	33.9 ± 0.95	15.7	18.2
CIFAR-100	PGD	ℓ_∞	8/255	78.6 ± 0.39	71.9	6.7	45.9 ± 0.23	47.3	-1.4
		ℓ_2	128/255	62.5 ± 0.09	56.8	5.7	39.9 ± 0.22	37.5	2.4
ImageNet	PGD	ℓ_∞	4/255	85.5 ± 8.87	62.7	22.8	50.5 ± 14.32	37.0	13.5
		ℓ_2	76/255	94.8 ± 1.16	63.0	31.8	63.2 ± 6.80	40.1	23.1

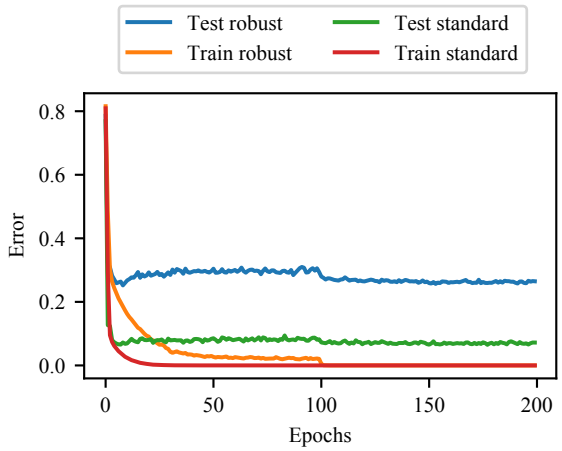

Figure 10. Learning curves for training an SVHN classifier which is adversarially robust to ℓ_2 perturbations of radius 128/255. Robust overfitting occurs early here as well, with robust test error increasing after the 9th epoch.

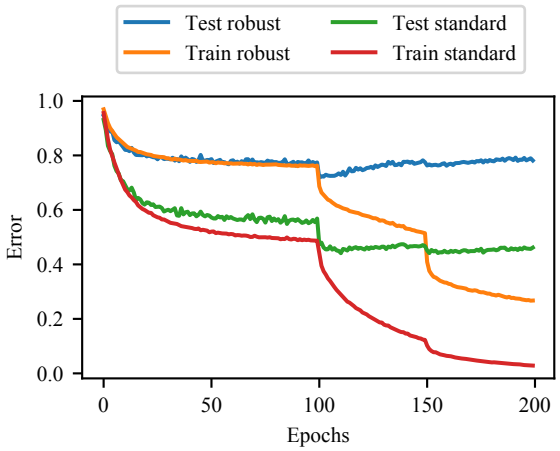

Figure 11. Learning curves showing robust overfitting on CIFAR-100 for the ℓ_∞ perturbation model.

ageNet, we fine-tune the pretrained model from https://github.com/madrylab/robustness (Engstrom et al., 2019) and continue training with the exact same parameters with a learning rate of 0.001, which uses an adversary with 5 steps of size 0.9/255 within a ball of radius 4/255.

ℓ_2 adversary We consider the ℓ_2 threat model with radius 128/255, with the PGD adversary taking 10 steps of size 15/255 on all datasets except for ImageNet. For Imagenet, we fine-tune the pretrained model from https://github.com/madrylab/robustness (Engstrom et al., 2019) and continue training with the exact same parameters with a learning rate of 0.001, which uses an adversary with 7 steps of size 0.5 within a ball of radius 3.

A.1. SVHN experiments

Figures 9 and 10 contain the convergence plots for the PGD-based adversarial training experiments on SVHN for ℓ_∞ and ℓ_2 perturbations respectively. We find that robust overfitting occurs even earlier on this dataset, before the initial learning rate decay, indicating that the learning rate threshold at which robust overfitting begins to occur has already been passed. The best checkpoint for ℓ_∞ achieves 39.0% robust error, which is a 6.6% improvement over the 45.6% robust

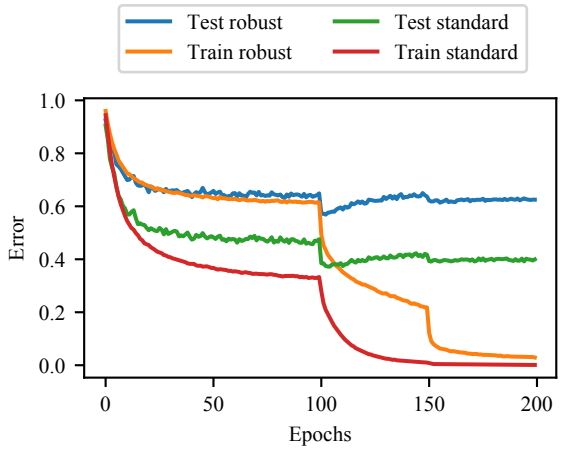

Figure 12. Learning curves showing robust overfitting on CIFAR-100 for the ℓ_2 perturbation model.

error achieved at the end of training.

A.2. CIFAR-100 experiments

Figures 11 and 12 contain the convergence plots for the PGD-based adversarial training experiments on CIFAR-100 for ℓ_∞ and ℓ_2 perturbations respectively. We find that robust overfitting on this dataset reflects the CIFAR-10 case, occurring after the initial learning rate decay. Note that in this case, both the robust test accuracy and the standard test accuracy are degraded from robust overfitting. The best checkpoint for ℓ_∞ achieves 71.9% robust error, which is a 6.7% improvement over the 78.6% robust error achieved at the end of training.

A.3. ImageNet experiments

Figure 13 contains the convergence plots for our continuation of PGD-based adversarial training experiments on ImageNet for ℓ_∞ and ℓ_2 perturbations respectively. Thanks to logs provided by the authors (Engstrom et al., 2019), we know the pretrained ℓ_2 robust ImageNet model had already been trained for 100 epochs at learning rate 0.1 followed by at least 10 epochs at learning rate 0.01, and so we continue training from there and further decay the learning rate at the 150th epoch to 0.001. Logs could not be found for the pretrained ℓ_∞ model, and so it is unclear how long it was trained and under what schedule, however the pretrained model checkpoint indicated that the model had been trained for at least one epochs at learning rate 0.001, so we continue training from this point on.

The ℓ_∞ pre-trained model appeared to have not yet converged for the checkpointed learning rate, and so further training without any form of learning rate decay was able to gradually deteriorate the performance of the model. The

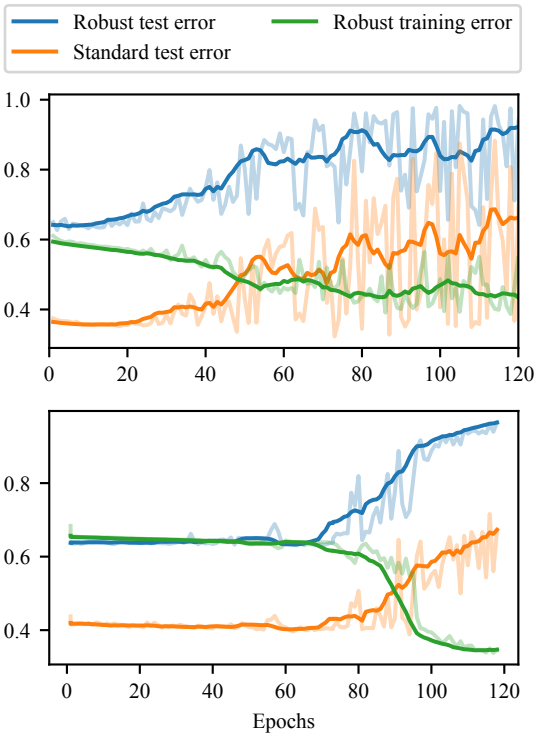

Figure 13. Continuation of training released pre-trained ImageNet models for ℓ_∞ (top) and ℓ_2 (bottom). The number of epochs indicate the number of additional epochs the pre-trained models were trained for.

ℓ_2 pre-trained model seemed to have already converged at the checkpointed learning rate, and so we do not see any significant changes in performance until after decaying the learning rate down to 0.001.

Note that the learning curves here are smoothed by taking an average over a consecutive 10 epoch window, as the actual curves are quite noisy in comparison to other datasets. This noise is reflected in Table 3, where ImageNet has the greatest variation in final error rates (both robust and standard). Training the models further can in fact improve the performance of the pretrained model slightly at specific checkpoints (e.g. from 66.4% initial robust test error down to 62.7% robust test error at the best checkpoint for ℓ_∞), however eventually the ImageNet models suffer greatly from robust overfitting, with an average increase of 22.8% robust error for the ℓ_∞ model and 31.8% robust error for the ℓ_2 model.

A.4. CIFAR-10 experiments

For CIFAR-10, in addition to the standard PGD training algorithm, we also consider the FGSM adversarial training algorithm (Wong et al., 2020) and TRADES (Zhang et al., 2019b). The convergence curves showing that robust over-

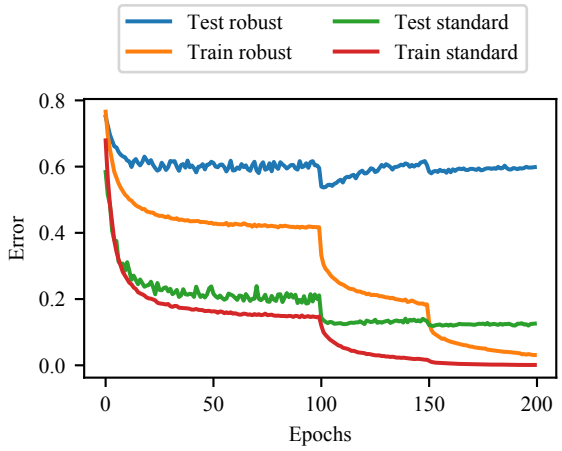

Figure 14. Learning curves showing robust overfitting from training with an FGSM adversary on CIFAR-10 for the ℓ_∞ perturbation model.

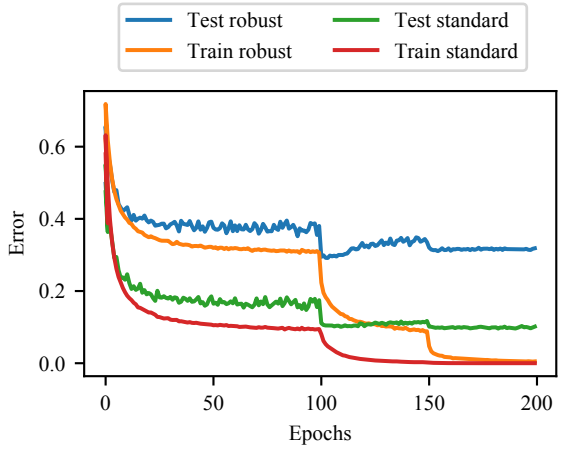

Figure 15. Learning curves showing robust overfitting from training with an FGSM adversary on CIFAR-10 for the ℓ_2 perturbation model.

fitting still occurs for these two algorithms in both the ℓ_∞ and ℓ_2 setting are shown in Figures 14 and 15 for FGSM and Figures 16 and 17 for TRADES.

FGSM adversarial training For FGSM adversarial training, we use the random initialization described by Wong et al. (2020). However, we find that when training until convergence using the piecewise decay learning rate schedule, the recommended step size of $\alpha = 10/255$ for ℓ_∞ training eventually results in catastrophic overfitting. We resort to reducing the step size of the ℓ_∞ FGSM adversary to 7/255 to avoid catastrophic overfitting, but still see robust overfitting.

We also note that Wong et al. (2020) use a cyclic learning rate schedule to further boost the speed of convergence, which differs from the piecewise decay schedule we discuss in this paper. If we run FGSM adversarial training in a more similar fashion to Wong et al. (2020) with the cyclic learning rate and fewer epochs, we find that this can sidestep the robust overfitting phenomenon and converge directly to the best checkpoint at the end of training. However, this requires a careful selection of the number of epochs: too few epochs and the final model underperforms, whereas too many epochs and we observe robust overfitting. In our setting, we find that training against an FGSM adversary for 50 epochs using a cyclic learning rate with a maximum learning rate of 0.2 allows us to recover a final robust test error of 53.22%, similar to the best checkpoint of FGSM adversarial training with piecewise decay and 200 epochs which achieved 53.7% robust test error in Table 3.

Relation of robust overfitting to catastrophic overfitting Previous work studying the effectiveness of an FGSM adversary for robust training noted that it is necessary to prevent "catastrophic overfitting" in order for FGSM training to be successful, which can be avoided by evaluating a PGD adversary on a training minibatch (Wong et al., 2020). Here we note that this is a distinct and separate behavior from robust overfitting: while catastrophic overfitting is a product of a model overfitting to a weaker adversary and can be detected by a stronger adversary on the training set, robust overfitting is a degradation of robust test set performance under the *same* adversary used during training which *cannot be detected on the training set*. Indeed, even successful FGSM adversarial training can suffer from robust overfitting when given enough epochs without catastrophically overfitting, as shown in Figure 14, suggesting that this is related to the generalization properties of adversarially robust training rather than the strength of the adversary.

TRADES For TRADES we use the publicly released implementation of both the defense and attack available at `https://github.com/yaodongyu/TRADES` to remove the potential for any confounding factors resulting from differences in implementation. We consider two possible options for learning rate schedules: the default schedule used by TRADES which decays at 75 and 90 epochs and runs for 100 epochs total (denoted TRADES learning rate),[10] and the standard learning rate schedule used by Madry et al. (2017) for PGD adversarial training, which decays at 100 epochs and 150 epochs. We additionally explore both the pre-activation ResNet18 architecture that we use

[10]This is the learning rate schedule described in the paper by Zhang et al. (2019c). Note that this differs slightly from the implementation in the TRADES repository, which uses the same schedule but only trains for 76 epochs, which is one more epoch after decaying. In our reproduction of the TRADES experiment, the checkpoint after the initial learning rate decay ends up with the best test performance over all 100 epochs.

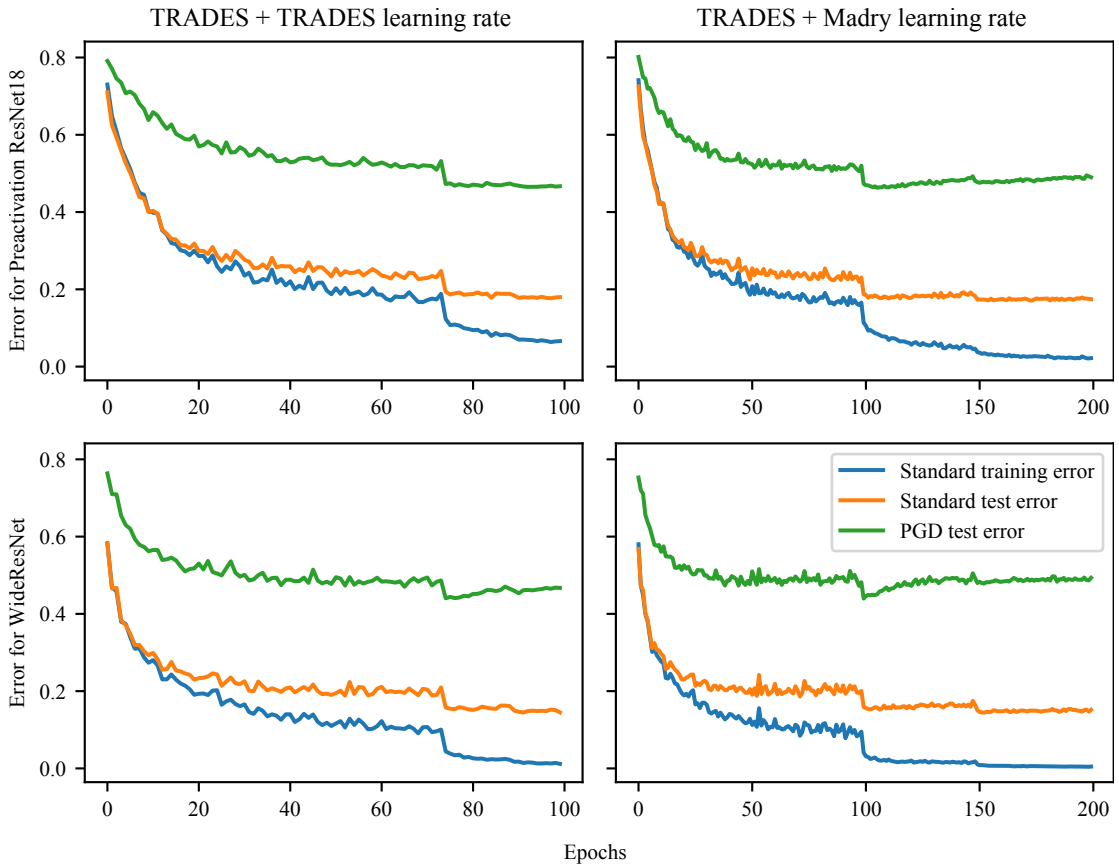

Figure 16. Learning curves when running TRADES for robustness to ℓ_∞ perturbations of radius 8/255 on combinations of learning rates and architectures for CIFAR10.

extensively in this paper, as well as the Wide ResNet architecture which TRADES uses. The corresponding learning curves for each combination of learning rate and model can be found in Figure 16 for ℓ_∞.

We note that in three of the four cases, we see a clear instance of robust overfitting. Only the default learning rate schedule used by TRADES on the smaller, pre-activation ResNet18 model doesn't indicate any degradation in robust test set performance. This is likely due the shortened learning rate schedule which implicitly early stops combined with the regularization induced by a smaller architecture having less representational power. The results here are consistent with our earlier findings on the impact of architecture size, where the Wide ResNet architecture achieves better performance than the ResNet18. The shortened TRADES learning rate schedule does not show the full extent of robust overfitting, as the models have not yet converged, whereas the Madry learning rate does (and also achieves a slightly better best checkpoint).

Figure 17 shows a corresponding curve for ℓ_2 robustness using TRADES for the pre-activation ResNet18 model with the Madry learning rate, which was the optimal combination from ℓ_∞ training. Note that the TRADES repository does not provide default training parameters or a PGD adversary for ℓ_2 training on CIFAR-10 nor could we find any such description in the corresponding paper, and so we used our attack parameters which were successful for PGD-based adversarial training (10 steps of size 15/255).

B. Experiments for various learning rate schedules

In this section, we explore the effect of the learning rate schedule with greater detail on the CIFAR10 dataset with a pre-activation ResNet18. Our search begins with a sweep over a range of different potential schedules which are commonly used in deep learning. Following this, we tune the best learning rate schedule to investigate its effect on the prevalence of robust overfitting.

B.1. Different types of schedules

We consider the following types of learning rates for our setting.

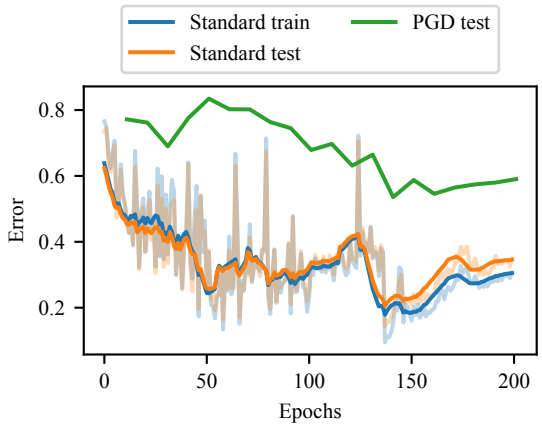

Figure 17. Learning curves when running TRADES for robustness to ℓ_2 perturbations of radius 128/255 for CIFAR10.

1. **Piecewise decay:** This is a fairly common learning rate used in deep learning, which decays the learning rate by a constant factor at fixed epochs. We begin with a learning rate of 0.1 and decay it by a factor of 10 at the 100th and 150th epochs, for 200 total epochs.

2. **Multiple decay:** This is a more gradual version of the piecewise decay schedule, with a piecewise constant schedule which reduces the learning rate at a linear rate in order to make the drop in learning rate less drastic. Specifically, the learning rate begins at 0.1 and is reduced by 0.01 every 50 epochs over 500 total epochs, eventually reaching a learning rate of 0.01 in the last 50 epochs.

3. **Linear decay:** This schedule does a linear interpolation of the drop from 0.1 to 0.01, resulting in a piecewise linear schedule. The learning rate is trained at 0.1 for the first 100 epochs, then linearly reduced down to 0.01 over the next 50 epochs, and further trained at 0.01 for the last 50 epochs for a total of 200 epochs.

4. **Cyclic:** This schedule grows linearly from 0 to to some maximum learning rate λ, and then is reduced linearly back to 0 over training as proposed by Smith (2017). We adopt the version from Wong et al. (2020) which already computed the maximum learning rate for the CIFAR10 setting on the same architecture which peaks 2/5 of the way through training at a learning rate of 0.2 over 200 epochs.

5. **Cosine:** This schedule reduces the learning rate using the cosine function to interpolate from 0.1 to 0 over 200 epochs. This type of schedule was used by Carmon et al. (2019) when leveraging semi-supervised data augmentation to improve adversarial robustness.

Table 4. Tuning experiments using stochastic gradient descent to optimize the best robust test error obtained from the piecewise decay schedule for a pre-activation ResNet18 on CIFAR-10.

DECAY EPOCH	START LR	END LR	BEST ROB ERR
100	0.1	0.01	46.7%
60	0.1	0.01	47.4%
70			47.3%
80			46.9%
90			47.3%
100	0.06	0.01	47.4%
	0.08		46.7%
	0.3		48.7%
	0.5		51.0%
100	0.1	0.006	46.0%
		0.008	46.1%
		0.03	47.8%
		0.05	49.3%

Note that the piecewise decay schedule is the primary learning rate schedule used in this paper. All of these approaches beyond the standard piecewise decay schedule dampen the initial drop in robust test error experienced by the piecewise decay schedule. As a result, the best checkpoints of these alternatives end up with worse performance than the best checkpoint of the piecewise decay schedule, since all of the learning rates eventually start increasing in robust test error due to robust overfitting after the initial drop. Robust overfitting appears to be ubiquitous across different schedules, as most approaches achieve their best checkpoint well before training has converged.

The cyclic learning rate is the exception here, which has two phases corresponding to when the learning rate is growing and shrinking, with the best checkpoint occurring near the end of the second phase. In both phases, the robust performance begins to improve, but then robust overfitting eventually occurs and keeps the model from improving any further. We found that stretching the cyclic learning rate over a longer number of epochs (e.g. 300) results in a similar learning curve but with worse robust test error for both the best checkpoint and the final converged model.

B.2. Tuning the piecewise decay schedule

Since the piecewise decay schedule appeared to be the most effective method for finding a model with the best robust performance, we investigate whether this schedule can be potentially tuned to improve the robust performance of the best checkpoint even further. The discrete piecewise decay schedule has three possible parameters: the starting learning rate, the ending learning rate, and the epoch at which the decay takes effect. We omit the last 50 epochs of the final decay, since the bulk of the impact from robust overfitting

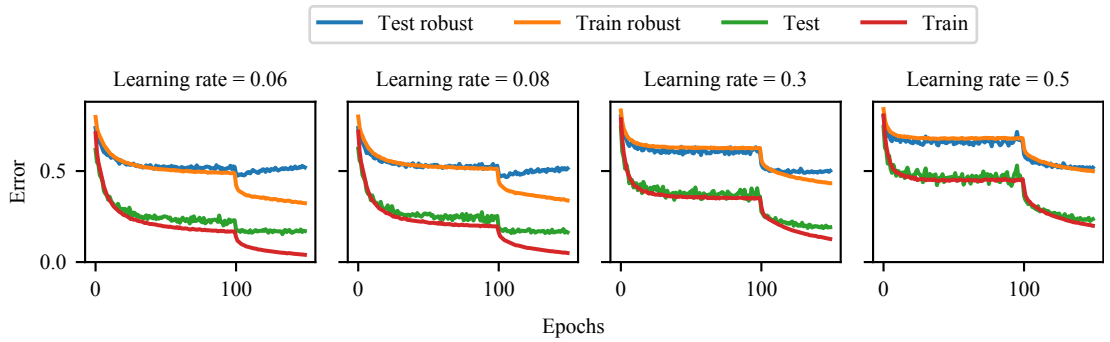

Figure 18. Learning curves for a piecewise decay schedule with a modified starting learning rate.

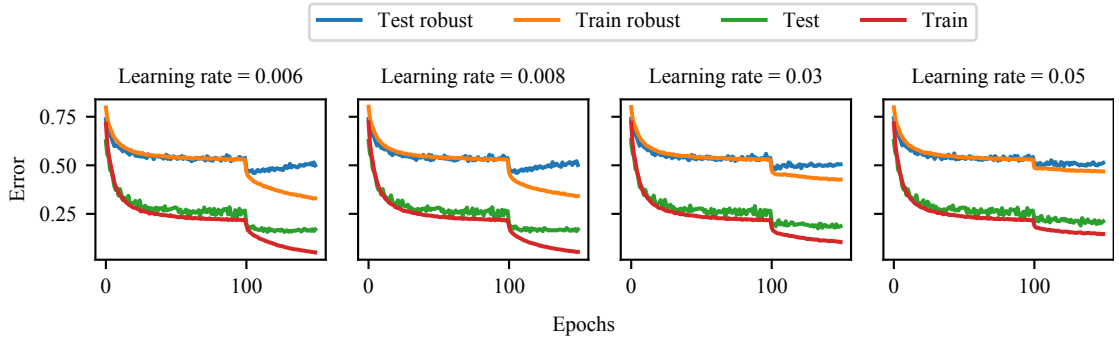

Figure 19. Learning curves for a piecewise decay schedule with a modified ending learning rate.

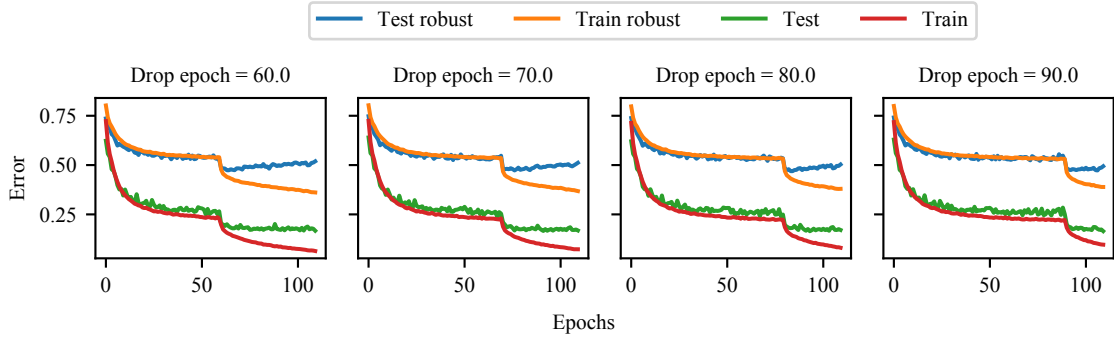

Figure 20. Learning curves for a piecewise decay schedule with a modified epoch at which the decay takes effect.

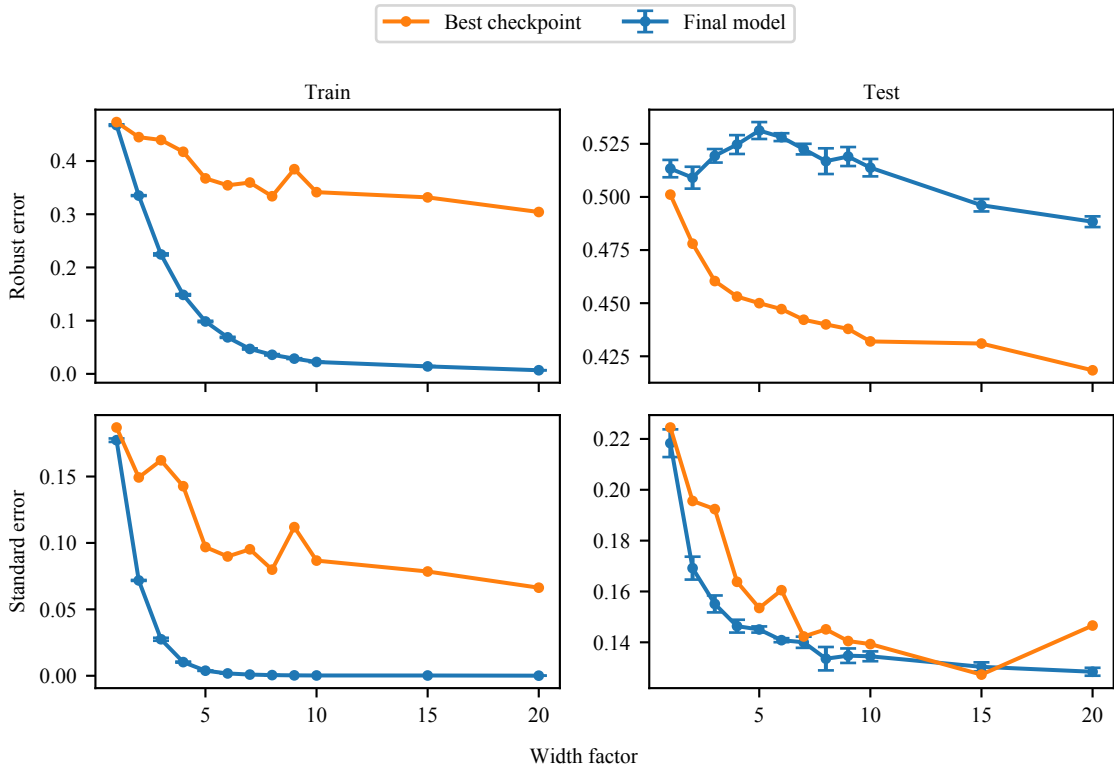

Figure 21. Standard and robust performance on the train and test set across Wide ResNets with varying width factors.

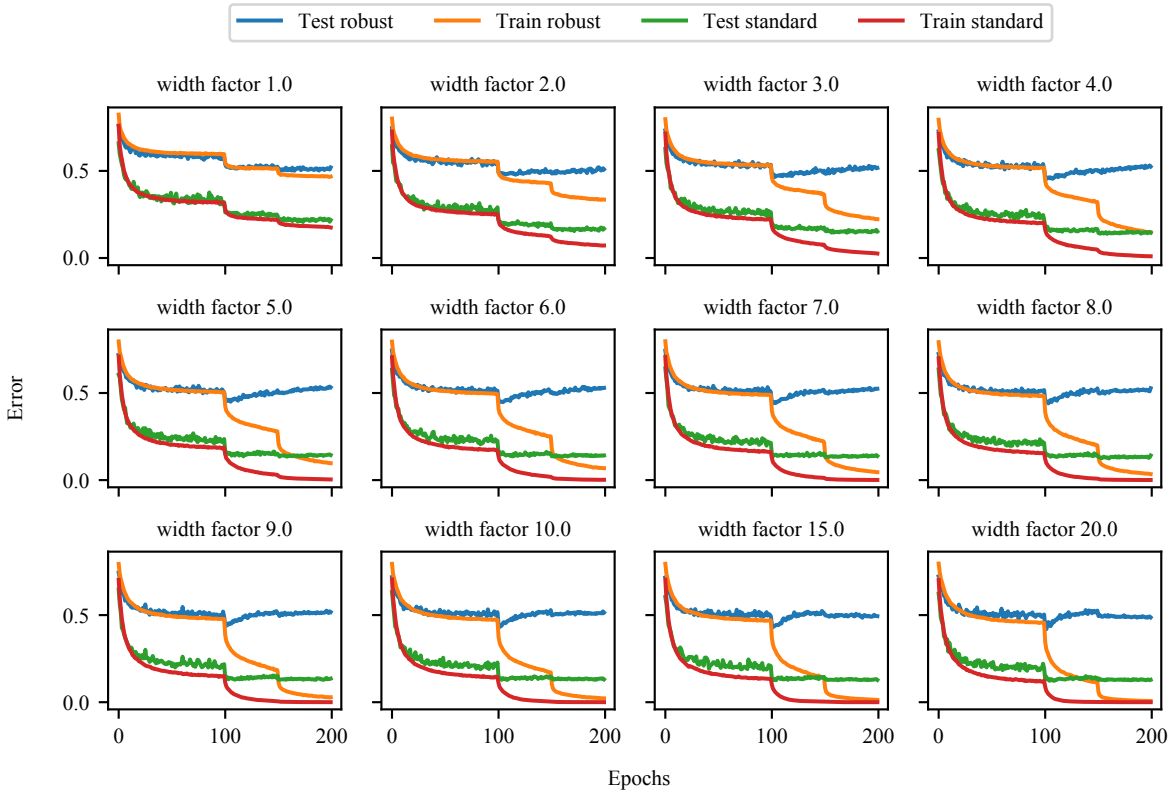

Figure 22. Learning curves for training Wide ResNets with different width factors.

occurs shortly after the first decay in this setting.

While tuning the starting learning rate and the decay epoch largely results in either similar or worse performance, we find that adjusting the learning rate used after the decay epoch can actually slightly improve the robust performance of the best checkpoint by 0.5%, as seen in Table 4. Note that robust overfitting still occurs in these tuned learning rate schedules as seen in Figures 18, 19, and 20, which show the learning curves for each one of the models shown in Table 4.

C. Double descent: exploring architecture sizes

For architecture size experiments, we use a Wide ResNet architecture (Zagoruyko & Komodakis, 2016) with depth 28 and varying widths to control the size of the network. For each width tested, we plot the standard and robust performance from the best checkpoint and final model in Figure 21. Learning curves for each width can be found in Figure 22. All models were trained with the same training parameters described in Section 4. Mean and standard deviation of the final model was taken over the last 5 epochs.

From both the generalization curves and the individual convergence plots, we see that no matter how large the architecture is, the checkpoint which achieves the lowest robust test error always has higher training robust error than the final model at convergence. We also find that both the final model at the end of convergence as well as the best checkpoint found during training all benefit from the increase in architecture size. Consequently, we find that robust overfitting and double descent can occur at the same time, despite having seemingly opposite effects on the notion of overfitting.

In contrast to the standard setting, we observe that the double descent occurs well before robust interpolation of the training data at a width factor of 5, after which the robust test set performance of the final model continues to improve with even larger architecture sizes. The network with width factor 20, the largest that we could run on our hardware, achieves 48.8% robust test error at the end of training and 41.8% robust test error at the best checkpoint. This marks a further improvement over the more typical choice of width factor 10 which achieves 51.4% robust test error at the end of training and 43.2% robust test error at the best checkpoint.

D. Preventing overfitting

D.1. Experimental setup

For the experiments in preventing overfitting, we use a PGD adversary with random initialization and 10 steps of step size 2/255. This is a slightly stronger adversary than considered in Madry et al. (2017) by using 3 additional steps, and we found the attack to be more effective than the adversary implemented by TRADES, achieving approximately 1% more PGD error than the TRADES adversary. However, our goal here is to explore the prevention of robust overfitting, and so it is not necessary to have strongest possible adversarial attack, and so for our purposes this adversary is good enough (and is known to be reasonably strong in the ℓ_∞ setting). For training, we use the same parameters as used for the CIFAR-10 experiments in Appendix A.4 (batch size, learning rate, weight decay, number of epochs). We primarily use the pre-activation ResNet18 since it is already sufficient for exhibiting the robust overfitting behavior.

D.2. Full set of results for Table 2

In this section, we present the expanded version of Table 2 to include standard test error metrics. The final robust and standard errors are an average of over the final 5 epochs of training when the model has converged, from which the standard deviation is also computed. The one exception is validation-based early stopping, where the final error is taken from the checkpoint chosen by the validation set, and consequently does not have a standard deviation. The best robust error is the lowest test robust error of all checkpoints through training, and the best standard error is the corresponding standard error which comes from this same checkpoint. For convenience we also show the difference in the final model's error and the best model's error, which indicates the amount of degradation incurred by robust overfitting.

D.3. Explicit regularization

In this section, we extend the plots depicting the robust and standard error over various regularization hyperparameters to also show the performance on the training set. We also show the learning curves for models trained with explicit regularization to show the extent of robust overfitting on various hyperparameter choices.

ℓ_1 regularization Figure 23 shows the training and testing performance of models using various degrees of ℓ_1 regularization. We performed a search over regularization parameters $\lambda = \{5 \cdot 10^{-6}, 5 \cdot 10^{-5}, 5 \cdot 10^{-4}, 5 \cdot 10^{-3}\}$, and found that both the final checkpoint and the best checkpoint have an optimal regularization parameter of $5 \cdot 10^{-5}$. Note that we only see robust overfitting at smaller amounts of regularization, since the larger amounts of regularization actually regularize the model to the point where the performance is being severely hurt.

Figure 24 shows the corresponding learning curves for these four models. We see clear robust overfitting for the smaller two options in λ, and find no overfitting but highly regu-

Table 5. Performance of adversarially robust training over a variety of regularization techniques for PGD-based adversarial training on CIFAR-10 for ℓ_∞ with radius $8/255$.

	Robust Test Error (%)			Standard Test Error (%)		
Regularization method	Final	Best	Diff	Final	Best	Diff
Early stopping w/ val	**46.9**	46.7	**0.2**	18.2	18.2	0.0
ℓ_1 regularization	53.0 ± 0.39	48.6	4.4	15.9 ± 0.13	15.4	0.5
ℓ_2 regularizaiton	51.4 ± 0.73	46.4	8.8	15.7 ± 0.21	14.9	0.8
Cutout	48.8 ± 0.79	46.7	2.1	16.8 ± 0.21	16.4	0.4
Mixup	49.1 ± 1.32	46.3	2.8	23.3 ± 3.04	19.0	4.3
Semi-supervised	47.1	40.2	6.9	23.0 ± 3.82	17.2	5.8

larized models for the larger two options, to the extent that there is no generalization gap and the training and testing curves actually appear to match.

ℓ_2 regularization Figure 25 shows the training and testing performance of models using various degrees of ℓ_2 regularization. We performed a search over regularization parameters $\lambda = \{5 \cdot 10^k\}$ for $k \in \{-4, -3, -2, -1, 0\}$ as well as $\lambda = 0.01$. Note that $5 \cdot 10^{-4}$ is a fairly widely used value for weight decay in deep learning. We find that only the smallest choices for λ result in robust overfitting (e.g. $\lambda \leq 0.1$. However, inspecting the corresponding learning curves in Figure 26 reveals that the larger choices for λ have a similar behavior to the larger forms of ℓ_1 regularization, and end up with highly regularized models whose test performance perfectly matches the training performance at the cost of converging to a worse final robust test error.

D.4. Data augmentation

In this section, we present additional details for the data augmentation approaches for preventing overfitting, namely cutout, mixup, and semi-supervised data.

Cutout To analyze the effect of cutout on generalization, we range the cutout hyperparameter of patch length from 2 to 20. Figure 27 shows the training and testing performance of models using varying choices of patch lengths. Additionally, for each hyperparameter choice, we plot the resulting learning curves in Figure 28.

We find the optimal length of cutout patches to be 14, which on it's own is not quite as good as vanilla early stopping, but when combined with early stopping merely matches the performance of vanilla early stopping. In all cases, we observe robust overfitting to steadily degrade the robust test performance throughout training, with less of an effect as we increase the cutout patch length.

Mixup When training using mixup, we vary the hyperparameter α from 0.2 to 2.0. The training and testing performance of models using varying degrees of mixup can be found in Figure 29. The resulting learning curves for each choice of α can be found in Figure 30.

For mixup, we find an optimal parameter value of $\alpha = 1.4$. Similar to cutout, when combined with early stopping, it can only attain similar performance to vanilla early stopping, and otherwise converges to a worse model. However, although the learning curves for mixup training are significantly noisier than other methods, we do observe the robust test error to steadily decrease over training, indicating that mixup does stop robust overfitting to some degree (but does not obtain significantly better performance).

E. Semi-supervised approaches

For semi-supervised training, we use a batch size of 128 with equal parts labeled CIFAR-10 data and pseudo-labeled TinyImages data, as recommended by Carmon et al. (2019). Each epoch of training is now equivalent in computation to two epochs of standard adversarial training. Note that the pre-activation ResNet18 is a smaller architecture than used by Carmon et al. (2019), and so in our reproduction, the best checkpoint which achieves 40.2% error is about 2% higher than 38.5%, which is what Carmon et al. (2019) can achieve with a Wide ResNet. Note that in the typical adversarially robust setting without additional semi-supervised data, a Wide ResNet can achieve about 3.5% lower error than a pre-activation ResNet18.

We observe that the semi-supervised approach does not exhibit severe robust overfitting, as the smoothed learning curves tend to be somewhat relatively flat and don't show significant increases in robust test error. However, relative to the base setting of using only the original dataset, the robust test performance is extremely variable, with a range spanning almost 10% robust error even when training error is relatively flat and has converged. As a result, it is critical to still use the best checkpoint even without robust overfitting, in order to avoid the fluctuations in test performance induced by the augmented training data.

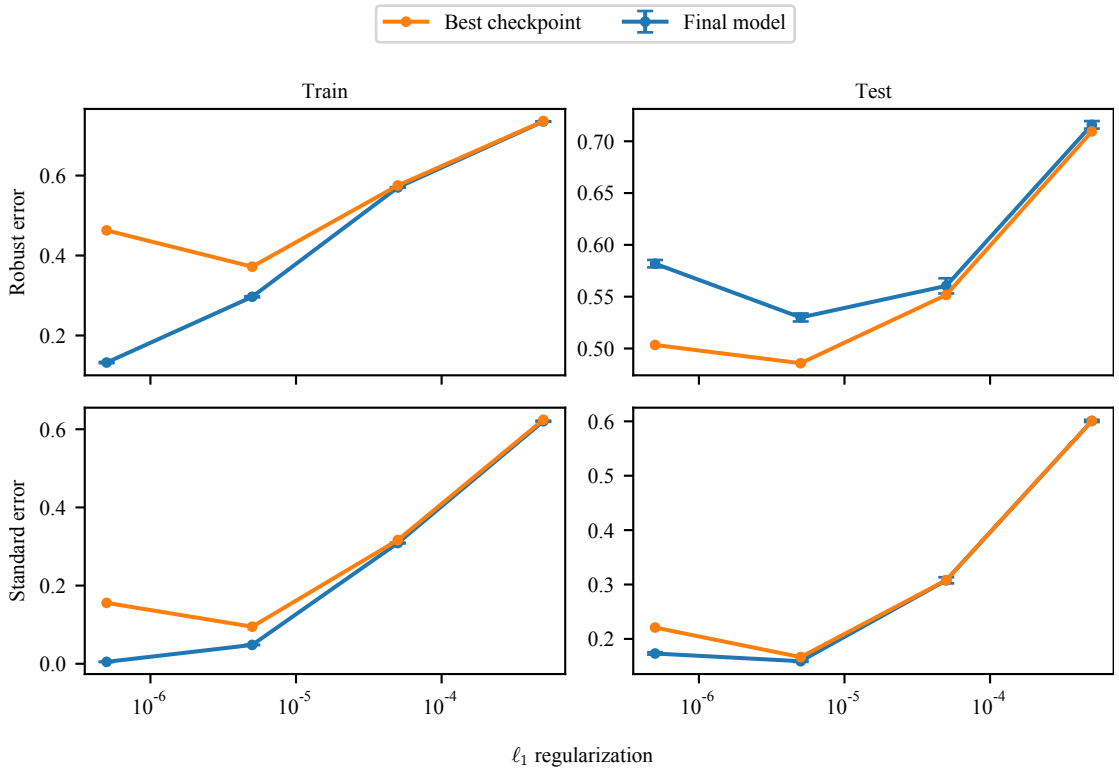

Figure 23. Standard and robust performance on the train and test set using varying degrees of ℓ_1 regularization.

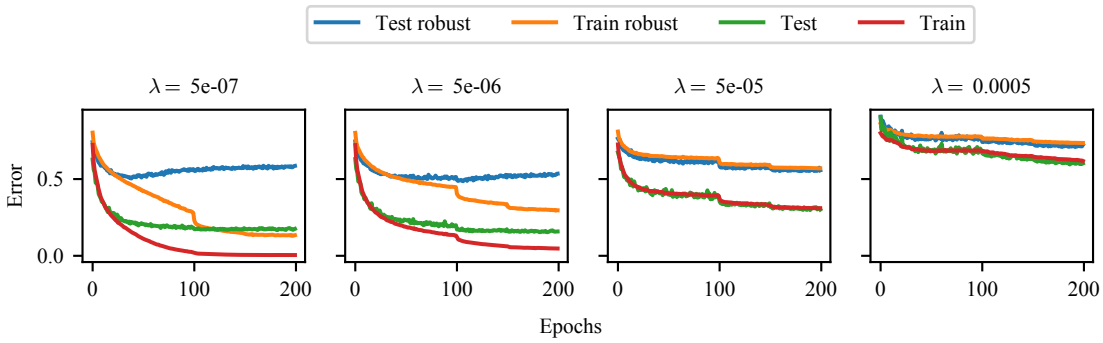

Figure 24. Learning curves for adversarial training using ℓ_1 regularization.

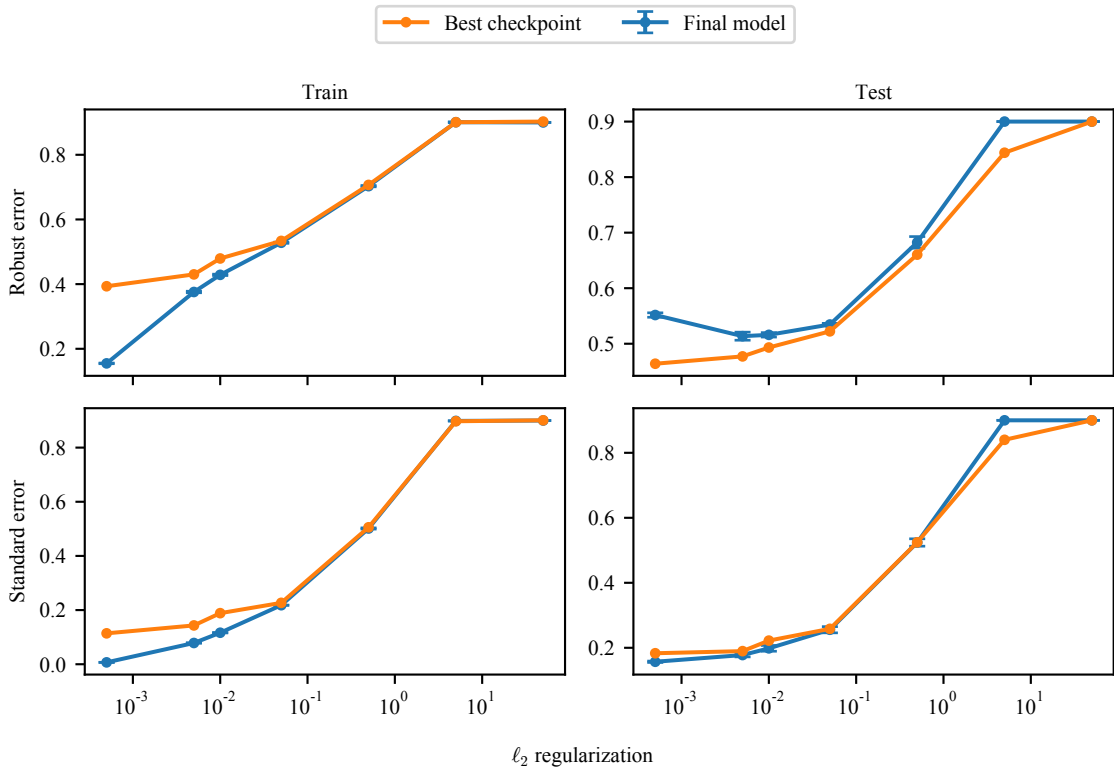

Figure 25. Standard and robust performance on the train and test set using varying degrees of ℓ_2 regularization.

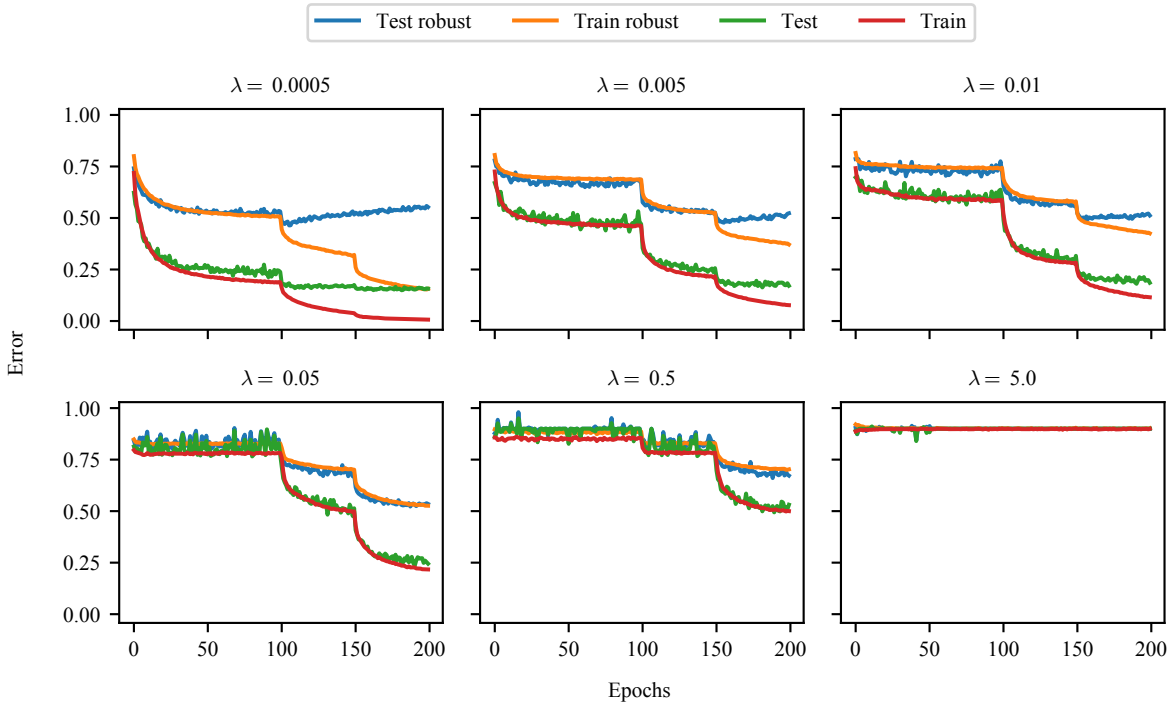

Figure 26. Learning curves for adversarial training using ℓ_2 regularization.

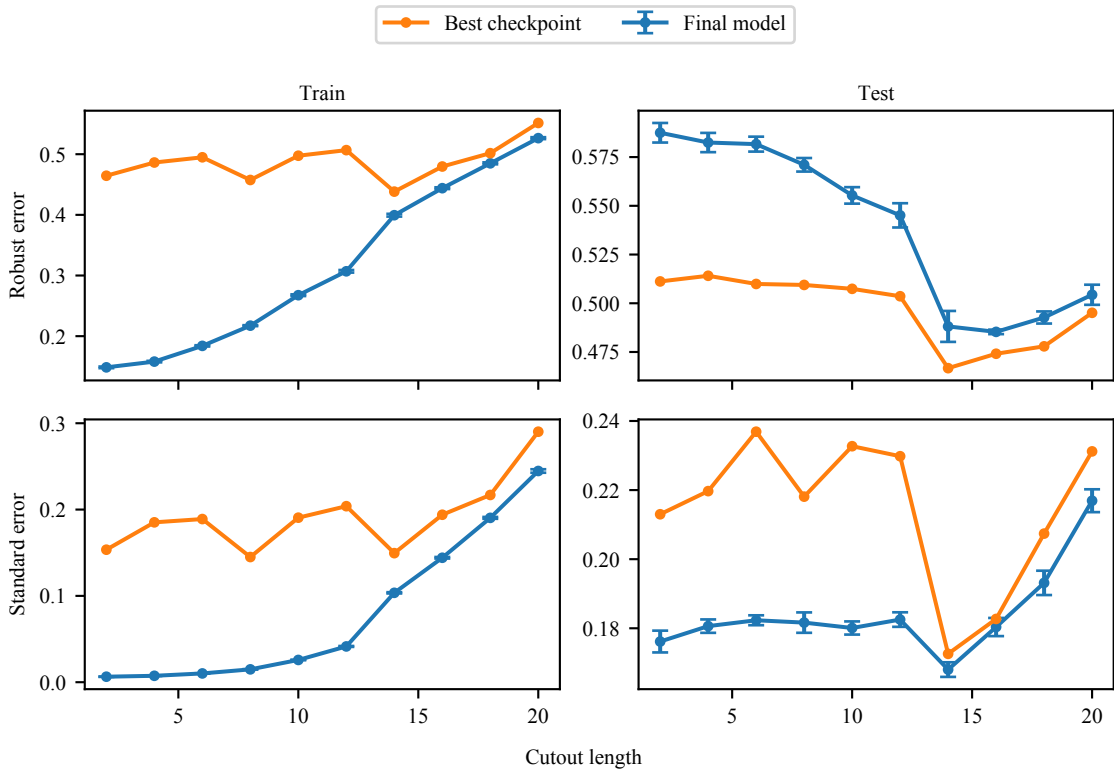

Figure 27. Standard and robust performance on the train and test set for varying cutout patch lengths.

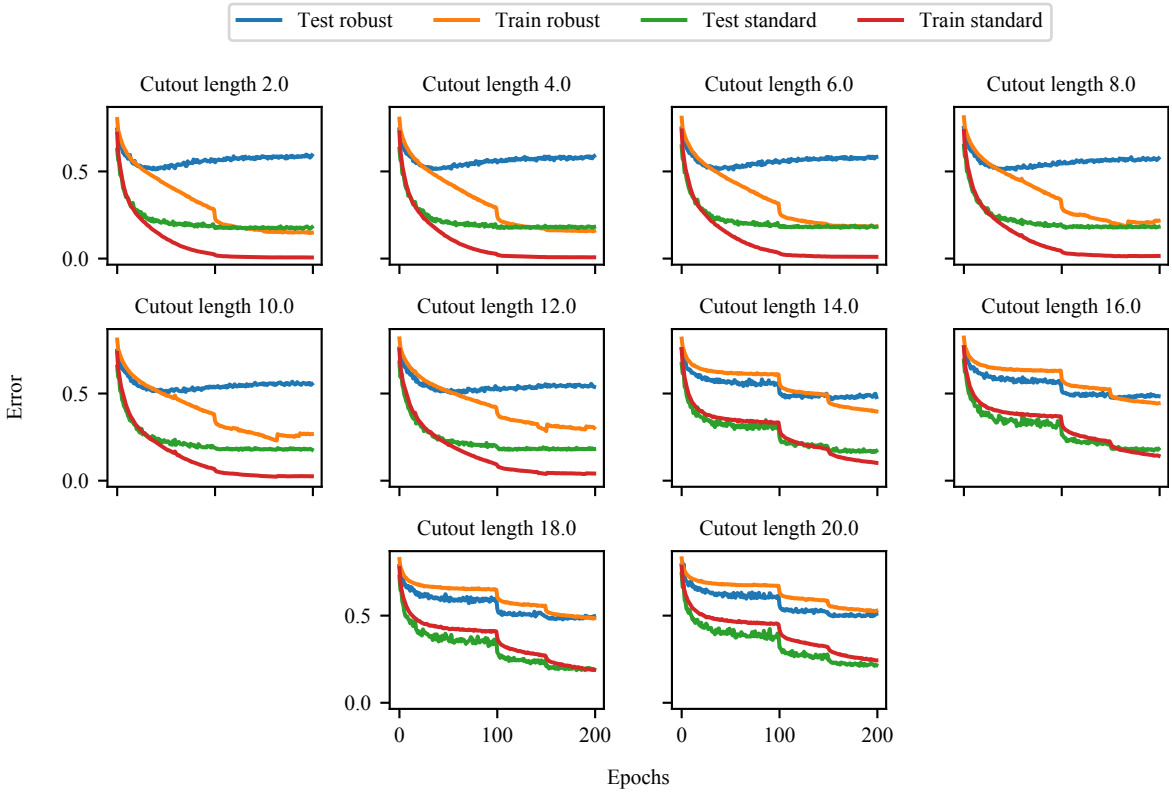

Figure 28. Learning curves for adversarial training using cutout data augmentation with different cutout patch lengths.

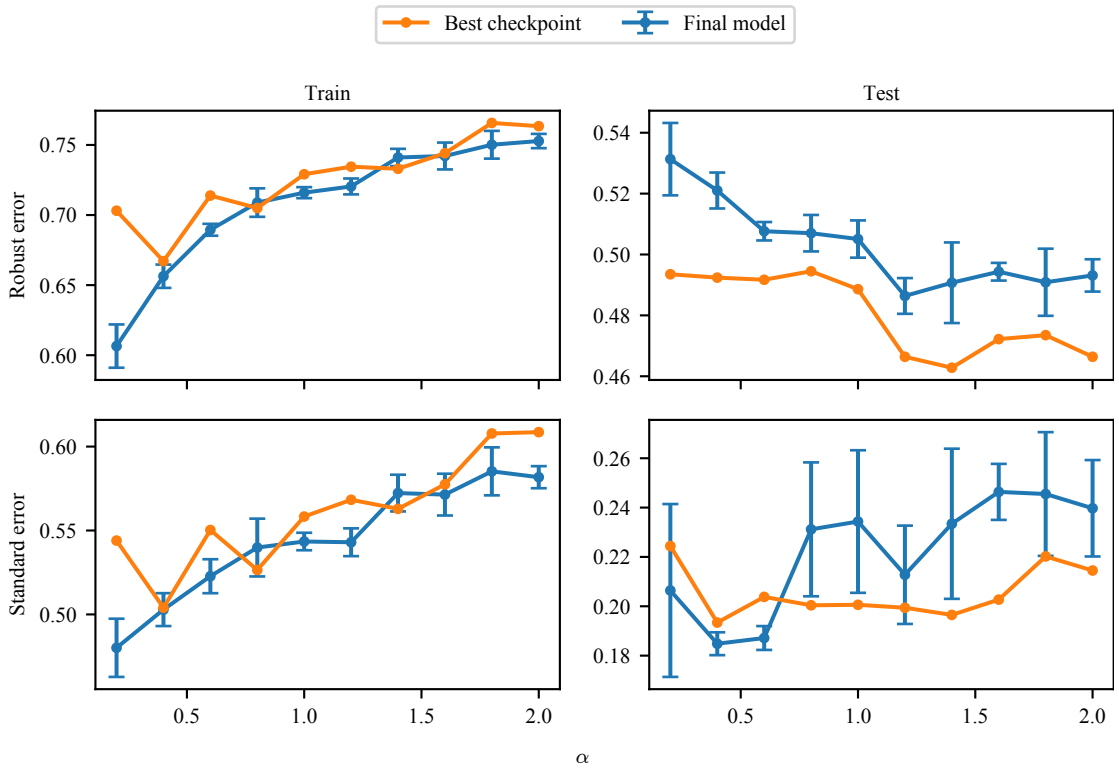

Figure 29. Standard and robust performance on the train and test set for varying degrees of mixup.

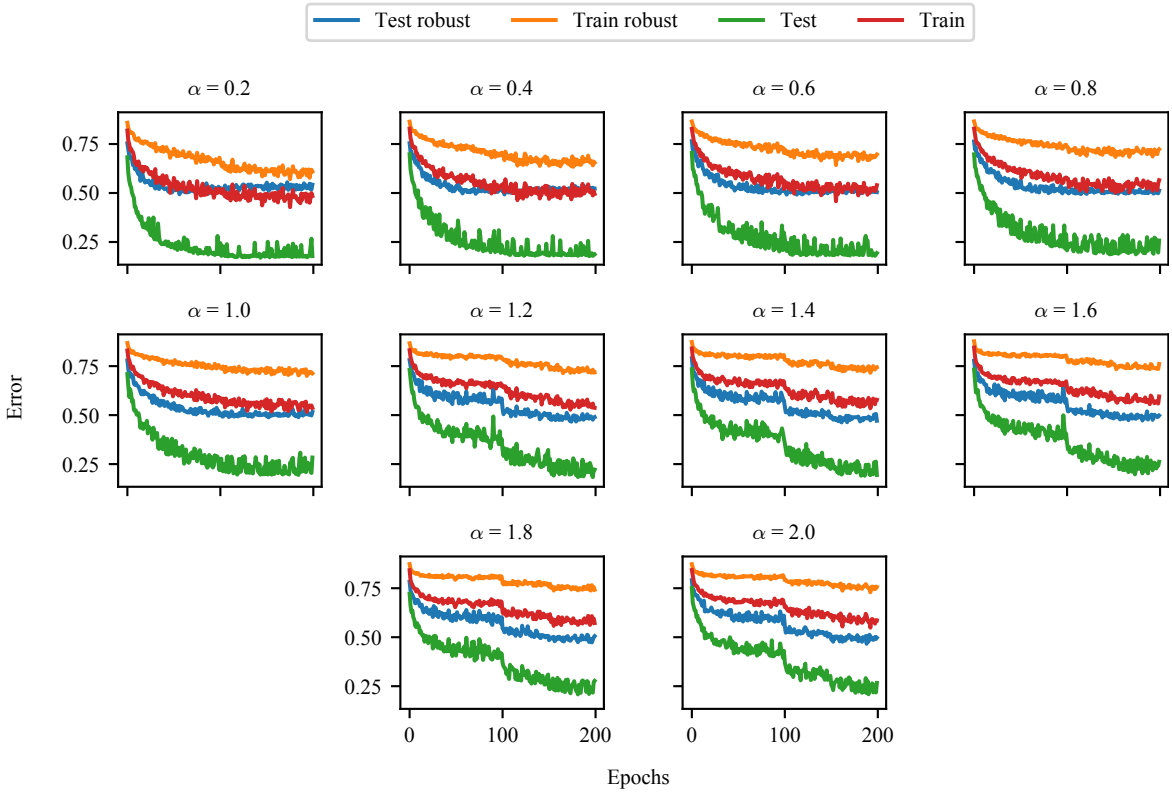

Figure 30. Learning curves for adversarial training using mixup with different choices of hyperparameter α.