Group 3: Adversiral training 1

Research Focus:

Deep learning architectures are vulnerable to adversarial perturbations. That is, indistinguishable modification to the data would cause the model to misclassify or predict wrong. Model robustness against such attacks is improved by adversarial training. Our research focus is to study generalisation and robustness of such models.

Paper 1:

The Curious Case of Adversarially Robust Models: More Data Can Help, Double Descend, or Hurt Generalization. *Yifei Min, Lin Chen, Amin Karbasi*

1. Introduction

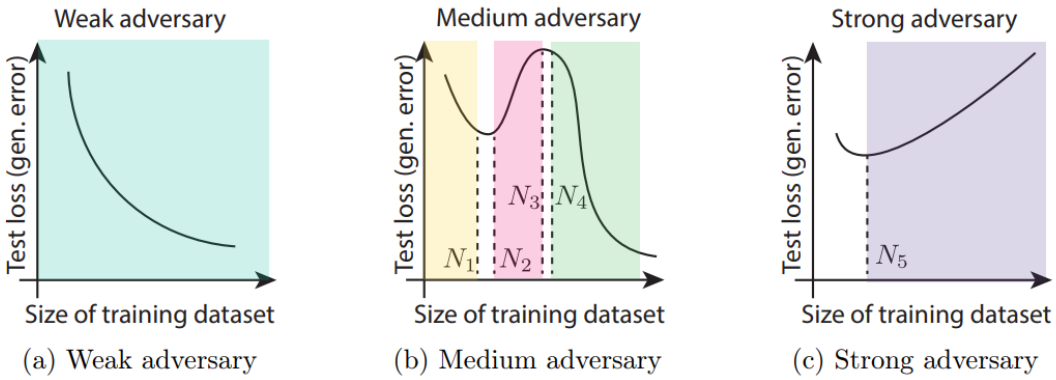
Adversarial training can generate models to be robust against perturbations. However, the ensuing problem is a decrease in accuracy. To mitigate this issue, it is widely believed that more training data will eventually help the adversarially robust model better generalize the benign/unperturbed test data. However, this paper challenges this conventional blief and show that more training data could impair the generalization performance of adversarially trained models.

2. Related Work

To demonstrate that more training data could hurt the generalization of adversarial robust models, we first consider linear classification problems with linear loss functions and identify three scenarios with different adversarial strengths, i.e., weak, medium, and strong adversarial regimes.

- In the weak adversary regime, the generalization is consistently improved with more training data.
- The medium adversary regime is probably the most interesting one among the three regimes. In this regime, the evolution of the generalization performance of adversarially robust models could be a double descent curve.

- In the strong adversary regime, the generalization of adversarially robust models deteriorates with more training data, except for a possible short initial stage where the generalization is improved with more data.



(a) Weak adversary (b) Medium adversary (c) Strong adversary

3.Empirical Results reproduction

In this section we study three different binary classification models.

In Section 3.1, we analyze the Gaussian mixture model under linear loss and see the existence of all three possible regimes (weak, medium, and strong adversary regimes), in which more training data can help, double descend, or hurt generalization of the adversarially trained model, respectively.
In Section 3.2, we construct a linear regression model with the squared loss that enables us to identify similar phenomenon in one-dimensional linear regression.
In Section 3.3, We study the soft-margin support vector machine with hinge loss. We find that for small, medium, and large ε can help, double descend, or hurt generalization.

3.1 Gaussian Mixture with Linear Loss

The distribution for the data generation is specified by

$$y \sim Unif(\pm 1)$$

and

$$x|y \sim N(y\mu, \Sigma)$$

We consider the linear loss

$$l(x, y; w) = -y < wx >$$

and we set the constraint set as

$$w \in \Theta = \{w \in \mathbb{R}^d | ||w||_\infty \leq \mathbf{W}\}$$

In this setting, the robust classifier is:

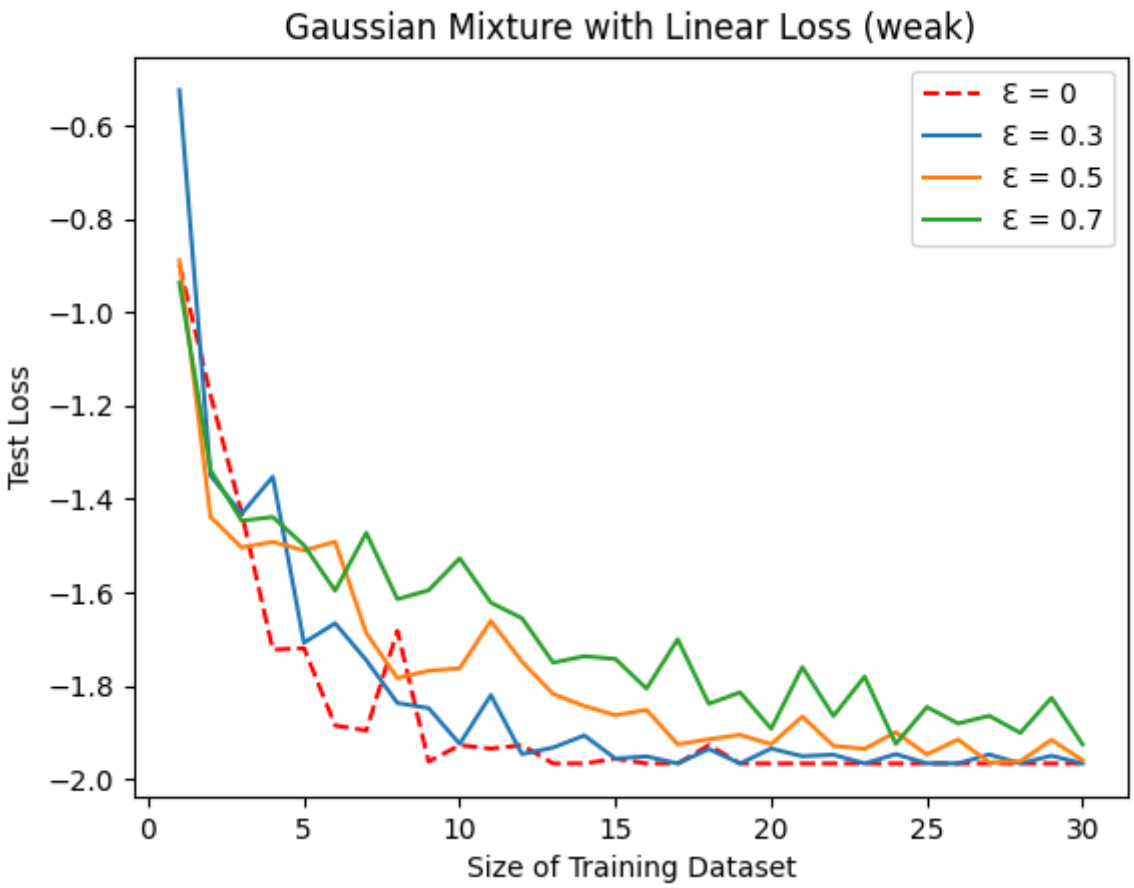
$$\underset{||w||_\infty \leq \mathbf{W}}{\arg\min} \sum_{i=1}^{n} \max_{\widetilde{x_i} \in B_{x_i}^\infty(\epsilon)} (-y_i < w, \widetilde{x_i} >) = \underset{||w||_\infty \leq \mathbf{W}}{\arg\max} \sum_{i=1}^{n} \min_{\widetilde{x_i} \in B_{x_i}^\infty(\epsilon)} (y_i < w, \widetilde{x_i} >)$$

where Θ is the parameter space and $B_{x_i}^\infty(\epsilon) \triangleq \widetilde{x_i} \in \mathbb{R}^d | ||\widetilde{x_i} - x||_\infty \leq \epsilon$ is an l^∞ ball centered at x with radius ϵ. The radius ϵ characterizes the strength of the adversary. A larger ϵ means a stronger adversary. This robust classifier minimizes the robust loss, or equivalently, maximizes the robust reward.
The generalization error of the robust classifier is given by:

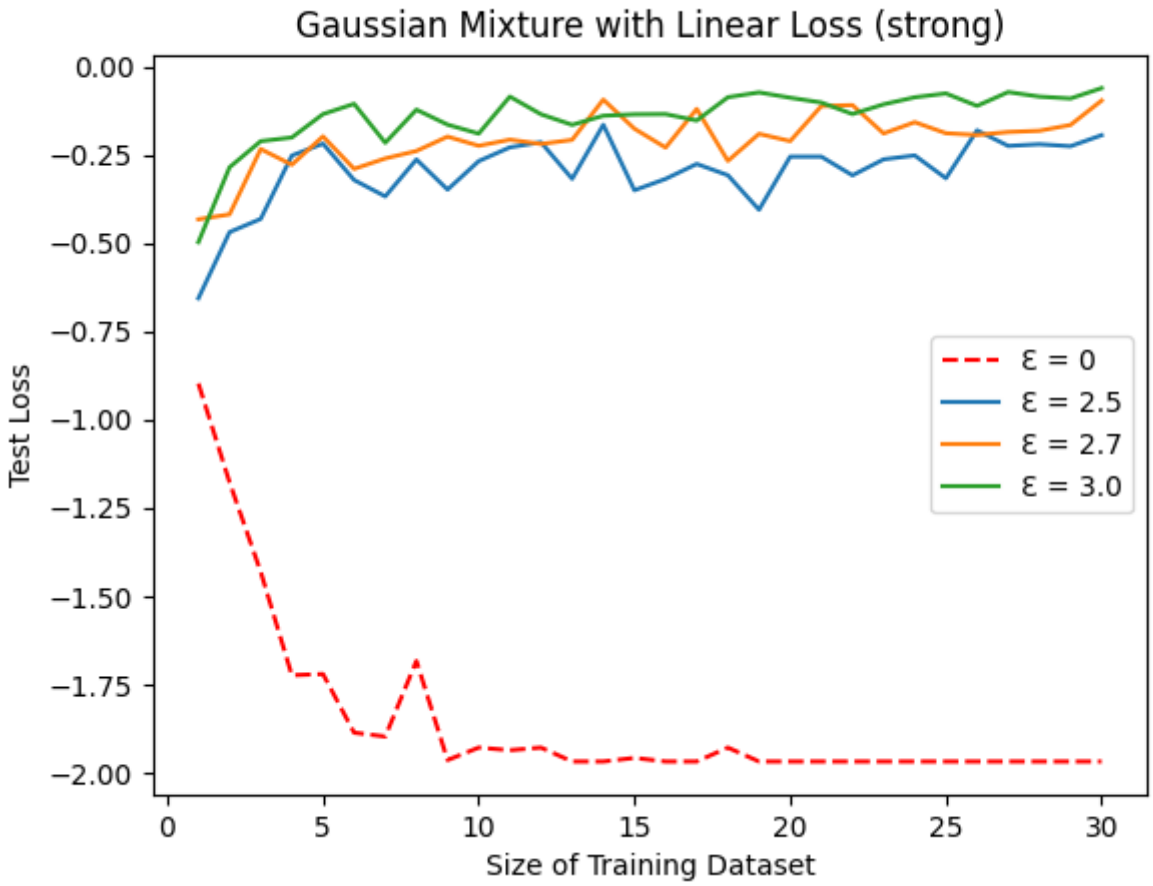
$$L_n = \mathbb{E}_{\{(x_i,y_i)\}_{i=1}^n \overset{i.i.d.}{\sim} D_{\mathcal{N}}} [\mathbb{E}_{(x,y)\sim D_{\mathcal{N}}} [-y < w^{rob}, x >]]$$

where the inner expectation is over the randomness of the test data point and the outer expectation is over the randomness of the training dataset. The test and training data are assumed to be independently sampled from the same distribution. The generalization error can be interpreted as the expected loss of the robust model over standard/unperturbed test data.

Follow are the two pictures showing test loss under weak and strong attacks.


Gaussian Mixture with Linear Loss (weak)
Test Loss
Size of Training Dataset
ε = 0
ε = 0.3
ε = 0.5
ε = 0.7
−0.6
−0.8
−1.0
−1.2
−1.4
−1.6
−1.8
−2.0
0
5
10
15
20
25
30


Gaussian Mixture with Linear Loss (medium)
Test Loss
Size of Training Dataset
ε = 0
ε = 1.1
ε = 1.3
ε = 1.5
−0.6
−0.8
−1.0
−1.2
−1.4
−1.6
−1.8
−2.0
0
5
10
15
20
25
30



By adjusting the parameter Ɛ, the shapes of test loss would differ. In the weak adversarial regime, all three curves decrease along with training data size. As Ɛ increases, test loss increases in total. In the medium adversarial attack regime, test loss would initially decrease, but soon increase, and at last slowly decrease. With the strong adversarial attack, test loss would initially decrease, but later increase with size of training dataset.

3.2 Linear regression model with the squared loss

In contrast to the classification model, we were very curious to see how the results would change if one-dimensional linear regression:

$$y = w^* x + e$$

was defined, where $e \sim N(0, 1)$ with squared loss:

$$l(x, y; w) = (y - wx)^2$$

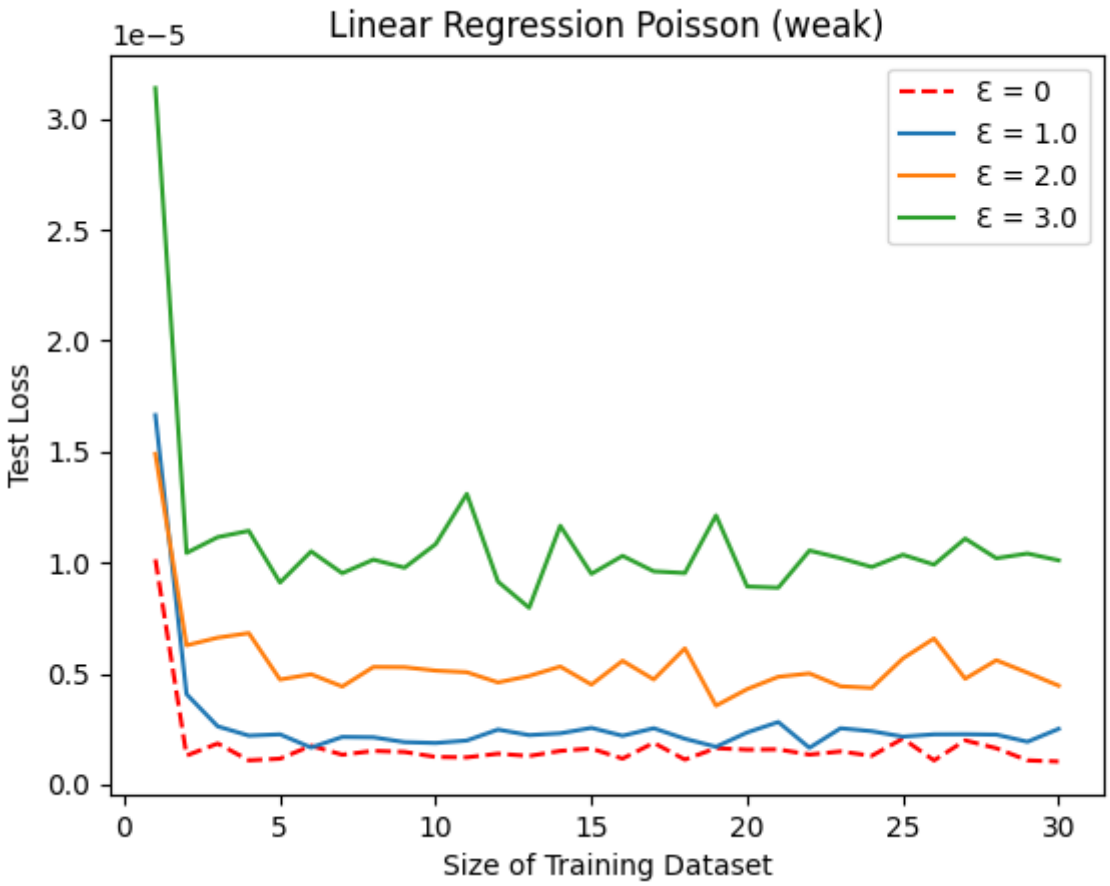
The data generation follows two different distributions for $\mathbf{X}$: the Gaussian distribution $\mathbf{N\ (0,\ 1)}$ and the shifted Poisson distribution $\mathbf{Poisson(5) + 1}$.

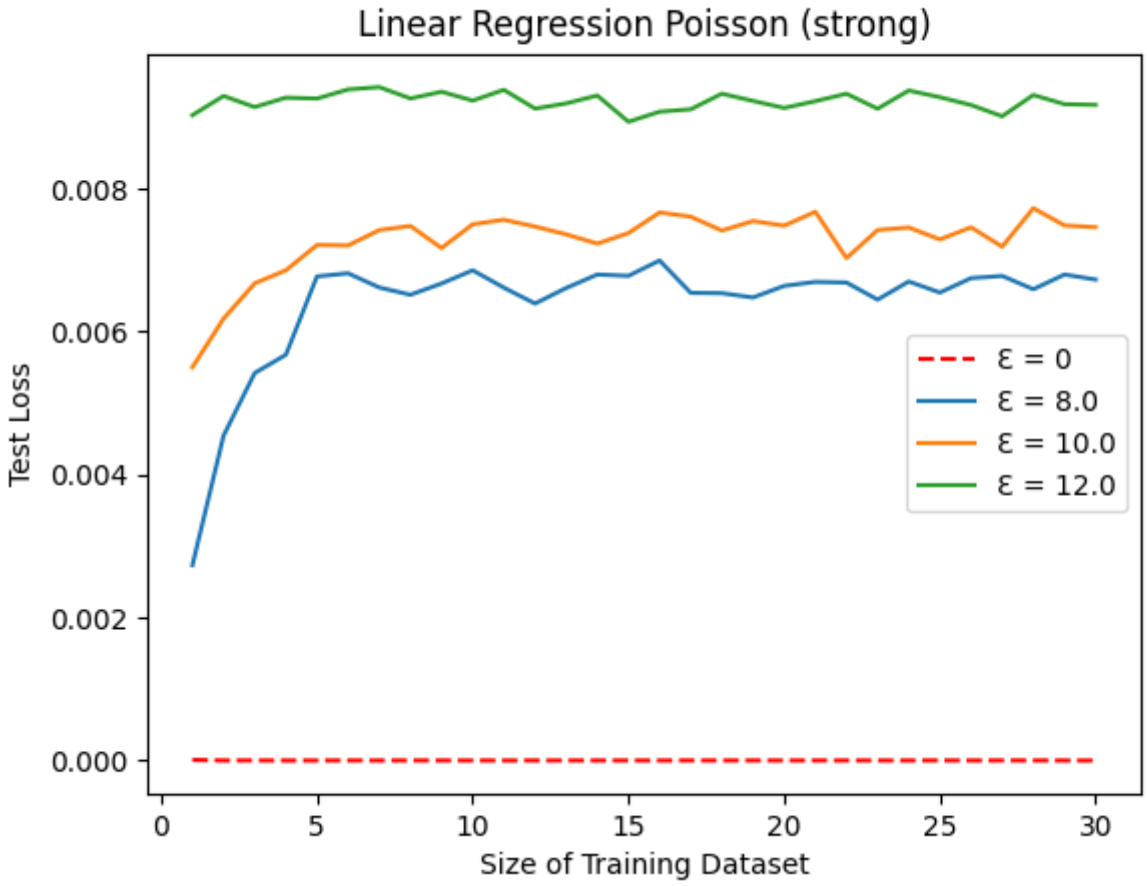
Given the coefficient w trained on n data points, the test loss is:

$$L_n = \mathbb{E}_{x,y}[(y - wx)^2] = \mathbb{E}_{x,e}[(w^* x + e - wx)^2] = \mathbb{E}((w^* - w)x)^2 + \mathbb{E}_e^2 = (w - w^*)^2 \mathbb{E}_x^2 + \mathbb{E}_e^2$$

Therefore, we report the scaled test loss

$$\widetilde{L_n} = (L_n - \mathbb{E}_e^2)/\mathbb{E}_x^2 = (w - w^*)^2$$





In both Gaussian and Poisson cases, we observe weak and strong regimes. But for Gaussian our model is not sensitive to adversarial attack, so that we choose to plot Poisson here to show the Phenomenon. When the

perturbation strength ε is less than a threshold, it falls into the weak regime where the (scaled) test loss is reduced with more training data. When ε exceeds the threshold, it exhibits the strong regime where more data hurts the (scaled) test loss. However, the threshold is remarkably between 3.0 and 8.0 for Poisson(5) + 1.

3.3 Support Vector Machine

We study the soft-margin support vector machine with hinge loss.

The dimension **d** equals **2** and the data is generated as :

$$y \sim Unif(\pm 1)$$

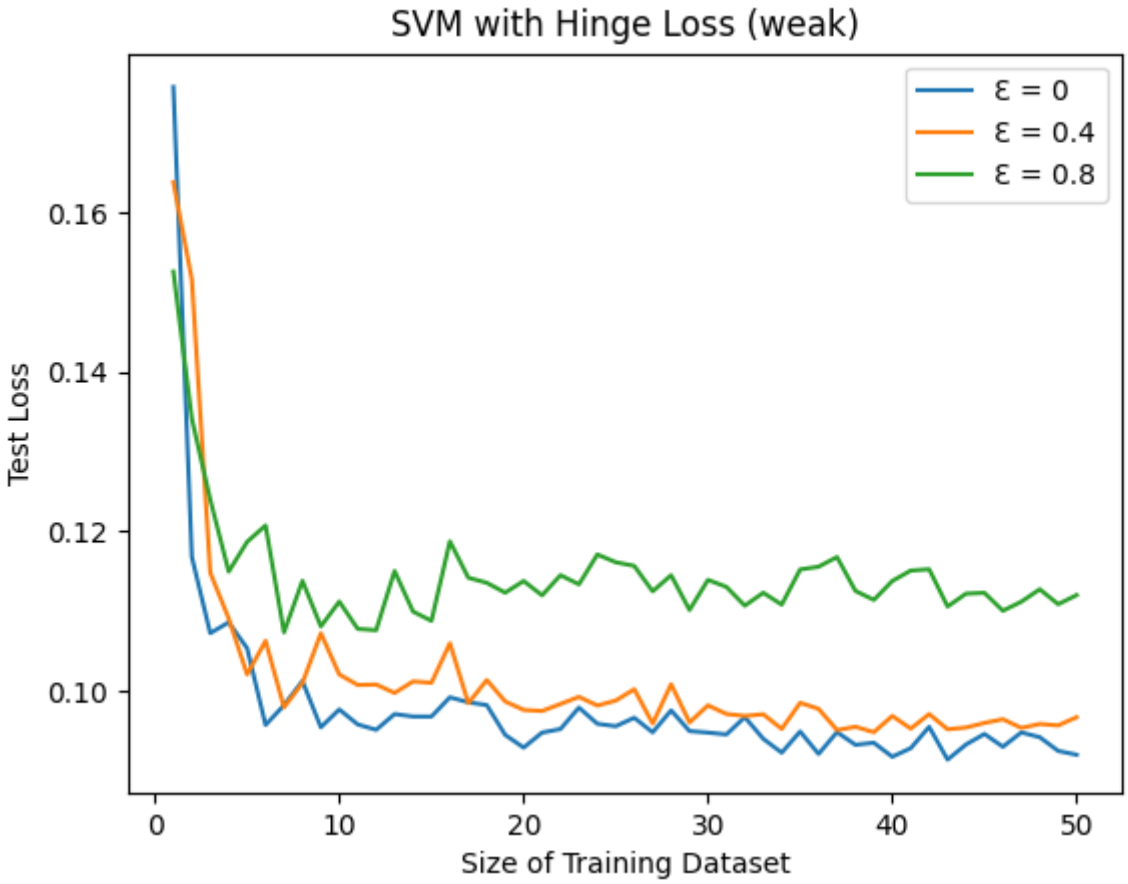
and

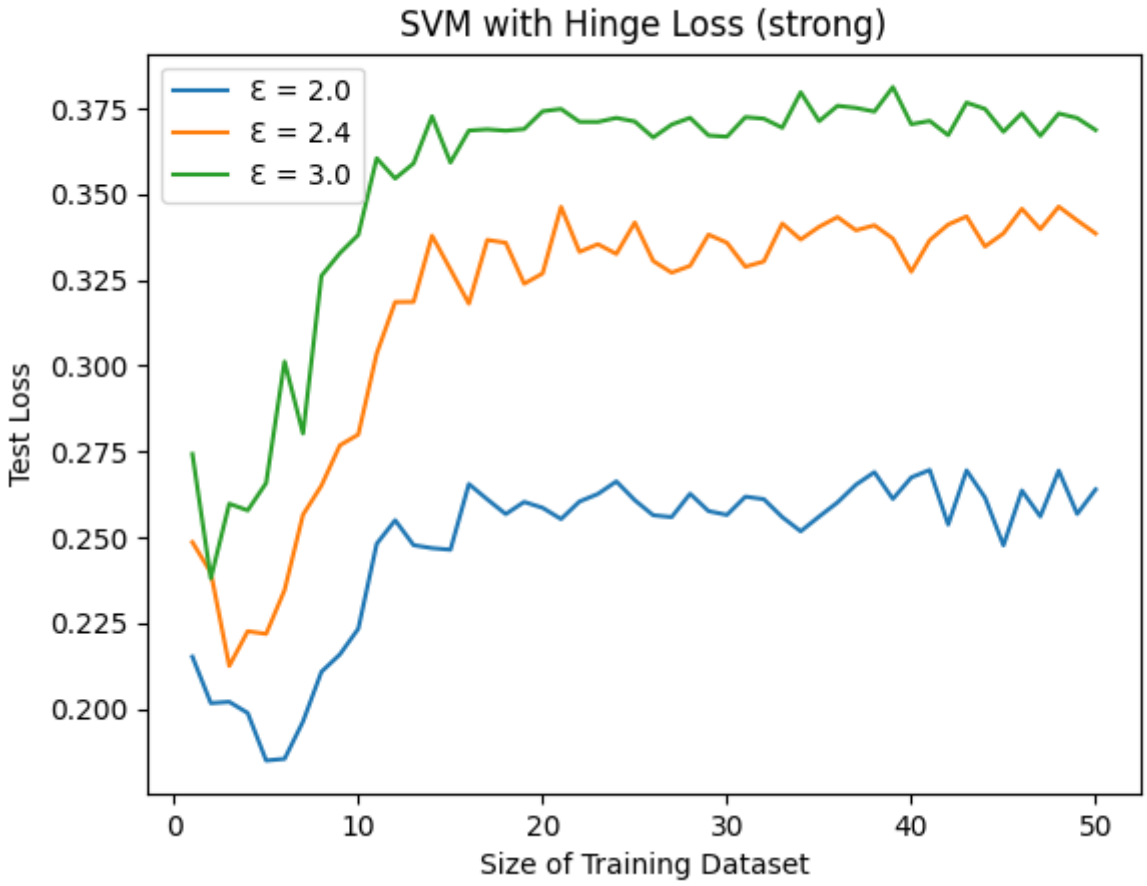
$$X \sim N(y\mu, I)$$

where

$$\mu = (1, 1)^T$$

Overall the tested results are in line with the three regimes we expected. But as epsilon grows, the test loss increases even though the opponent is still weaker, with the threshold mentioned in the paper at around ϵ = 0.8. This may indicate that for more complicated models (such as SVMs), even relatively weaker adversaries can result in situations where more data always increases the test loss.





In SVM, we observe three adversarial regimes. The weak and strong regimes are analogous to the Gaussian Mixture Linear Loss case. The medium attack exposes a clear double descent for test loss. The curve first

decreases, then increase, and finally slowly decreases with training data size.

4. Conclusion

We still expect our robust model on both perturbed and unperturbed test sets. However, our results show that in some cases, the current approach cannot achieve low generalization error on both datasets simultaneously. That is, more data should help us to learn better. Our results show that this is not necessarily true: when the attack intensity is high, more data nevertheless produces a larger error. The current adversarial training framework may not be ideal. So new ideas may currently be needed to develop models that can reliably perform well on both accuracy and robustness.

Paper 2:

Overfitting in adversarially robust deep learning. *Leslie Rice, Eric Wong, J. Zico Kolter*

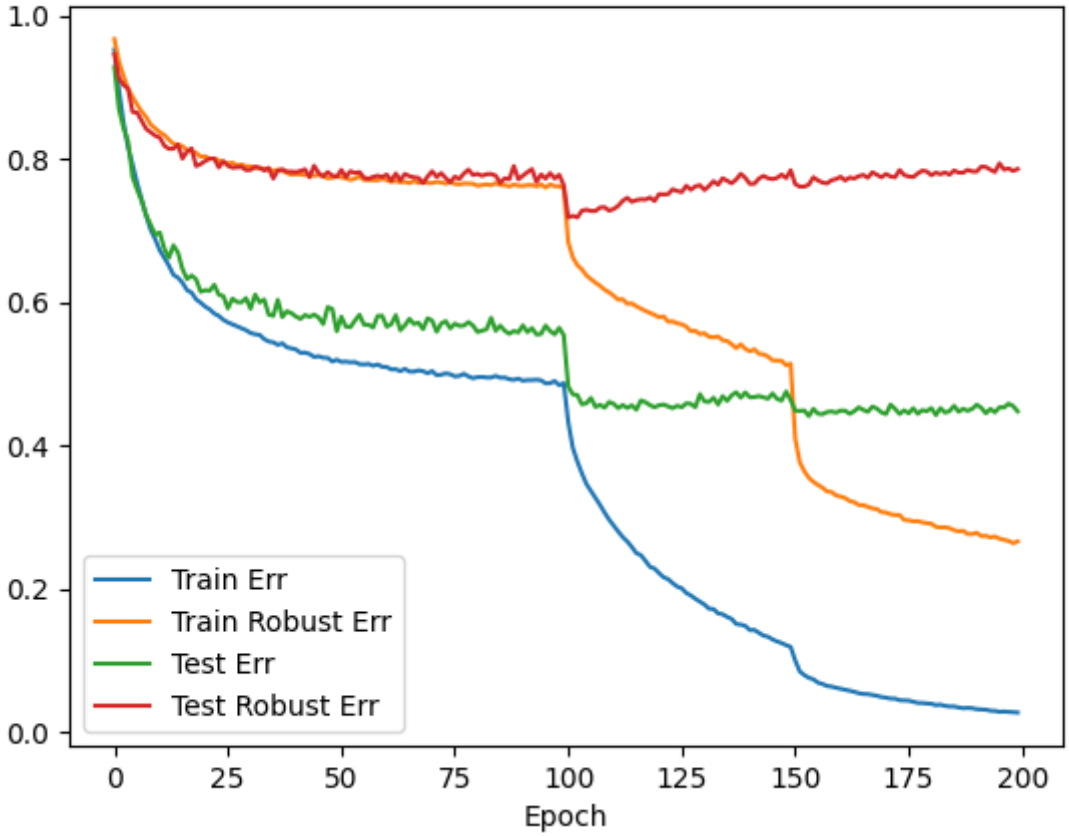
1. Introduction

It has been observed that too much training during the model's learning process can lead to overfitting in many machine learning models that perform well on the training set, but not well when predicting new data. In deep learning, however, it is common to use overparameterized networks and train for as long as possible, as numerous studies show, theoretically and empirically, that such practices surprisingly do not unduly harm the generalization performance of the classifier. Surprisingly, different from neural networks trained without perturbations, the *adversarially trained deep networks*, which are trained to minimize the loss under worst-case adversarial perturbations, has overfitting, a prevalent phenomenon which is called in the paper as **"robust overfitting"**.

The 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 in adversarial training on CIFAR-100, 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.

2. Robust Overfitting

Robust overfitting is a prevalent phenomenon in adversarial training. In the figure below, after the first learning rate decay, further training leads to a short decrease in in test robust error, reaching the lowest value, but soon increases and plateaus till the training session ends. On the contrary, the train error and train robust error keep decreasing. Though the following graph is obtained by CIFAR-100, this phenomenon exists extensively on other datasets, such as CIFAR-10, SVHN, ImageNet.



3.Empirical Results reproduction

All the results of the reproduction can also be checked on this link:
https://github.com/franklinqin0/robust_overfitting/tree/master/reproduce_results
(https://github.com/franklinqin0/robust_overfitting/tree/master/reproduce_results)

All of our follwing reproductions are based on Cifar-10.

We use the the follwing table from paper to compare the empirical results to see if the other methods still have good effect on robust overfitting. The way to compare the effectness of other methods with early stopping is to juxapose the robust test errors, which are the red lines in the figures of the reproductions below.

We can see from the reproductions using other methods to prevent overfitting (Mixup, Cutout, Regularization etc.), which are not as good as the result of the early stopping. Early stopping serves to calculate the robust error of the validation data at the end of each epoch (an epoch set is a round of traversal of all training data), and stop training when the error no longer decreases in recent epochs. Pure early stopping is done with a hold-out validation set, because if the robust error is otherwise based on the test set performance, test set information is leaked and goes against the traditional machine learning paradigm.

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

- As shown in the Table 2 are the experiments results from the original paper. Robust performance of PGD-based adversarial training with different regularization methods on CIFAR-10 using a PreActResNet18 for `∞ 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. (this specific table **from the original paper**, and there is a very obvious error that the difference between final and best of the method l2 regularization is clearly not 55.2)

How to compare the methods on reduing the robust overfitting error: with understanding the meaning of the robust overfitting error, this is quite simple to point out that the difference gap of the robust test error value between final and best (= DIFF in the table) represents how good the method is on preventing the robust overfitting.

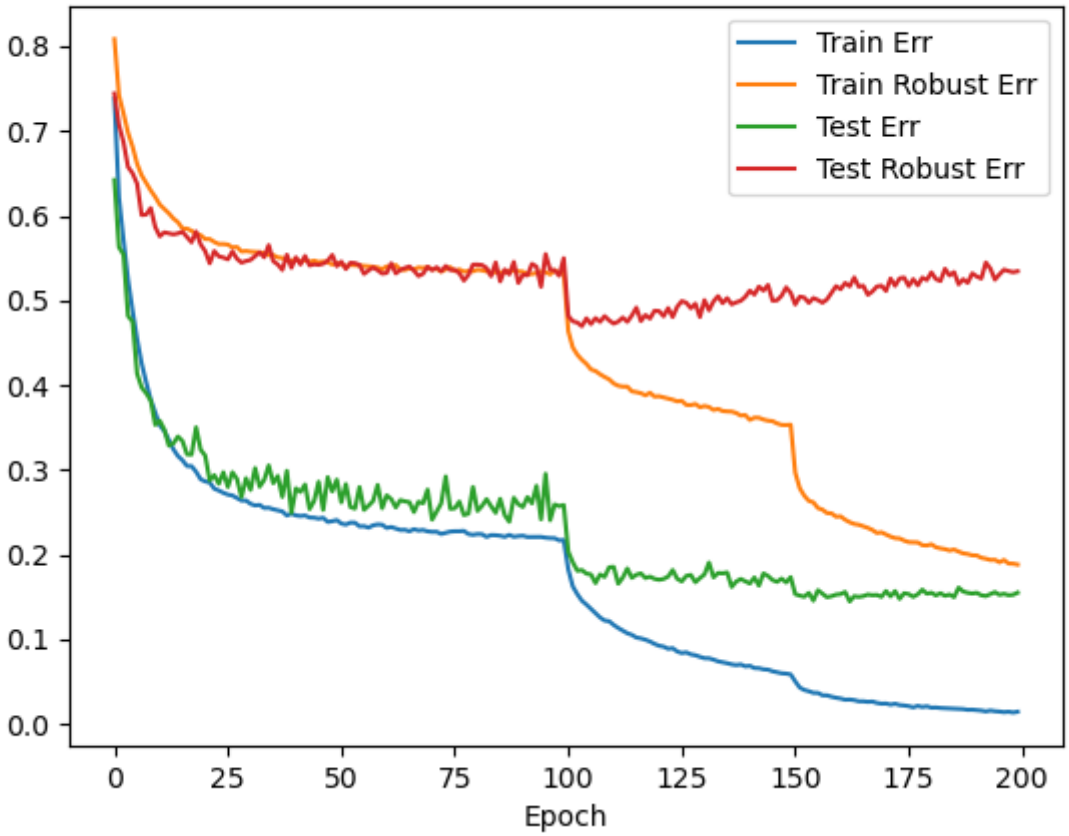
3.1 Cutout

Cutout is to delete several rectangular areas at random (the pixel value is changed to 0). Randomly cut out some areas in the sample and fill with 0 pixel values, and the classification result remains unchanged.

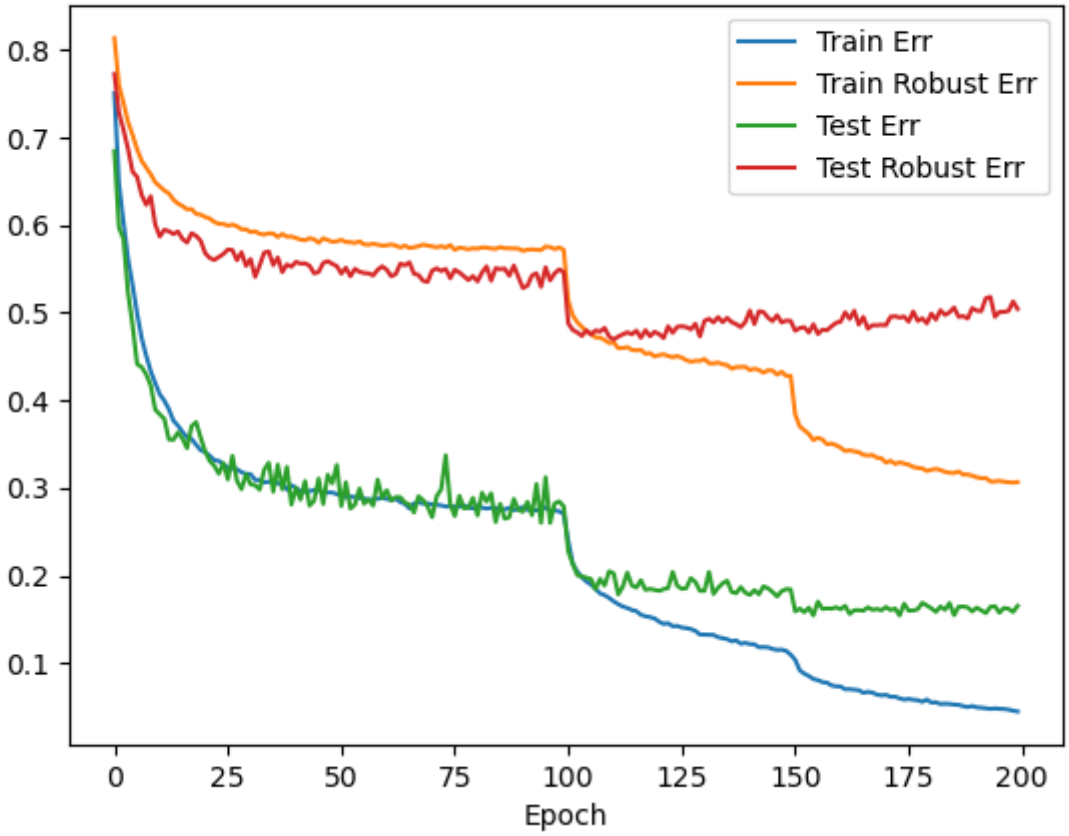
In this part, we set the parameter value of the cutout as: 2, 10, 20.

Note that only when the cutout length is larger, such as 20 , robust overfitting is not observed.

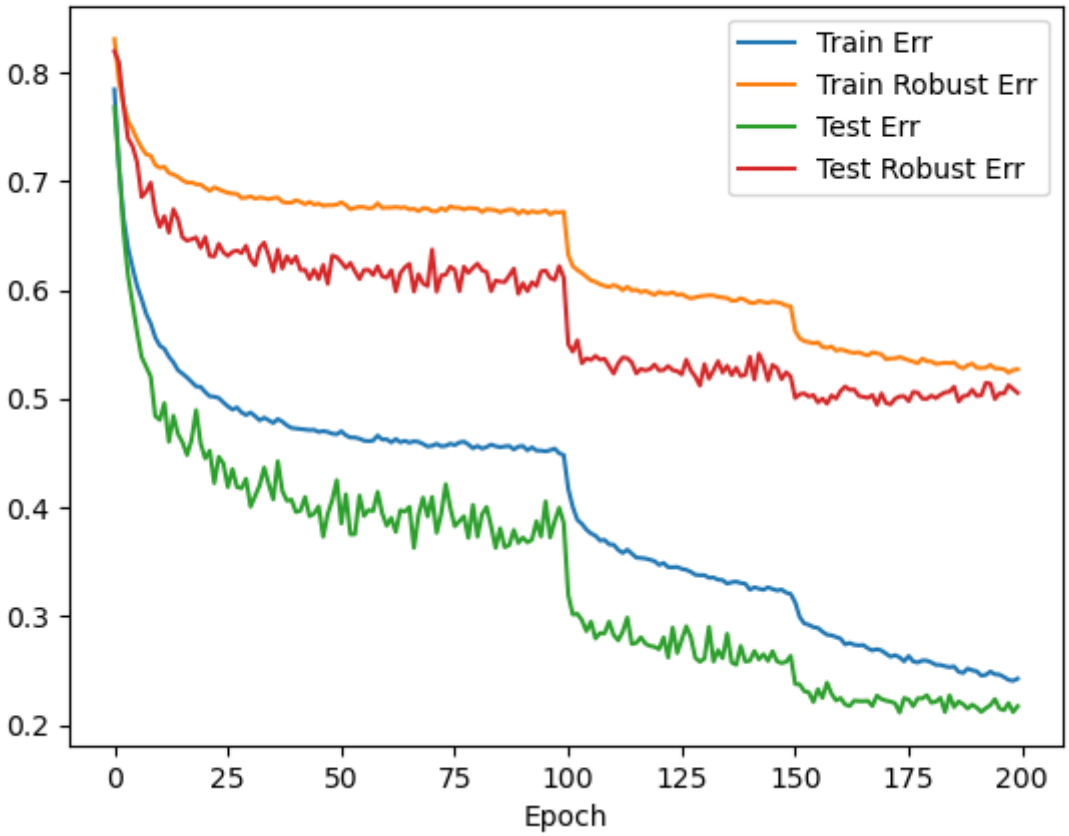
- Cutout len = 2



- Cutout len = 10



- Cutout len = 20



In [ ]:

```
# an example command of cutout len paramter = 20
python3 train_cifar.py --fname 'experiments/cifar10_cutout/preactresnet18_20'  --cutout --cutout-l
```

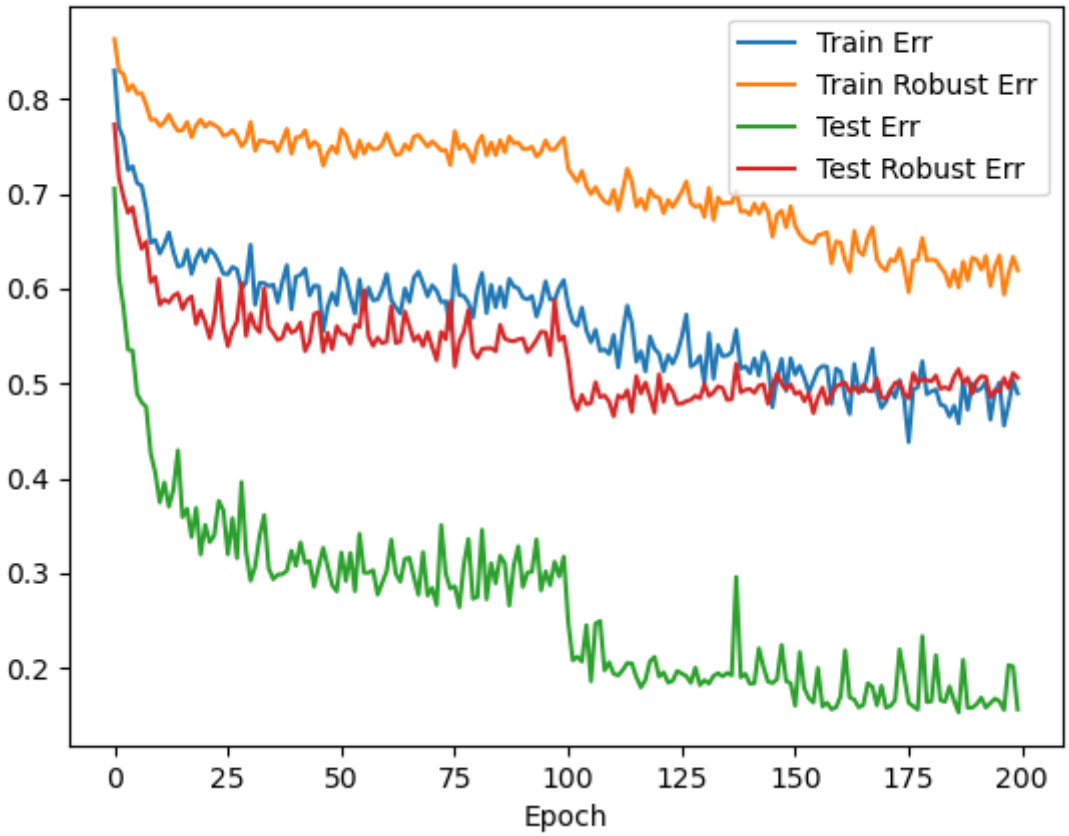
3.2 Mixup

The two random samples are mixed proportionally, and the classification results are distributed proportionally. The pixels at each position of the two images are superimposed according to a certain ratio, and the labels are allocated according to the pixel superposition ratio.

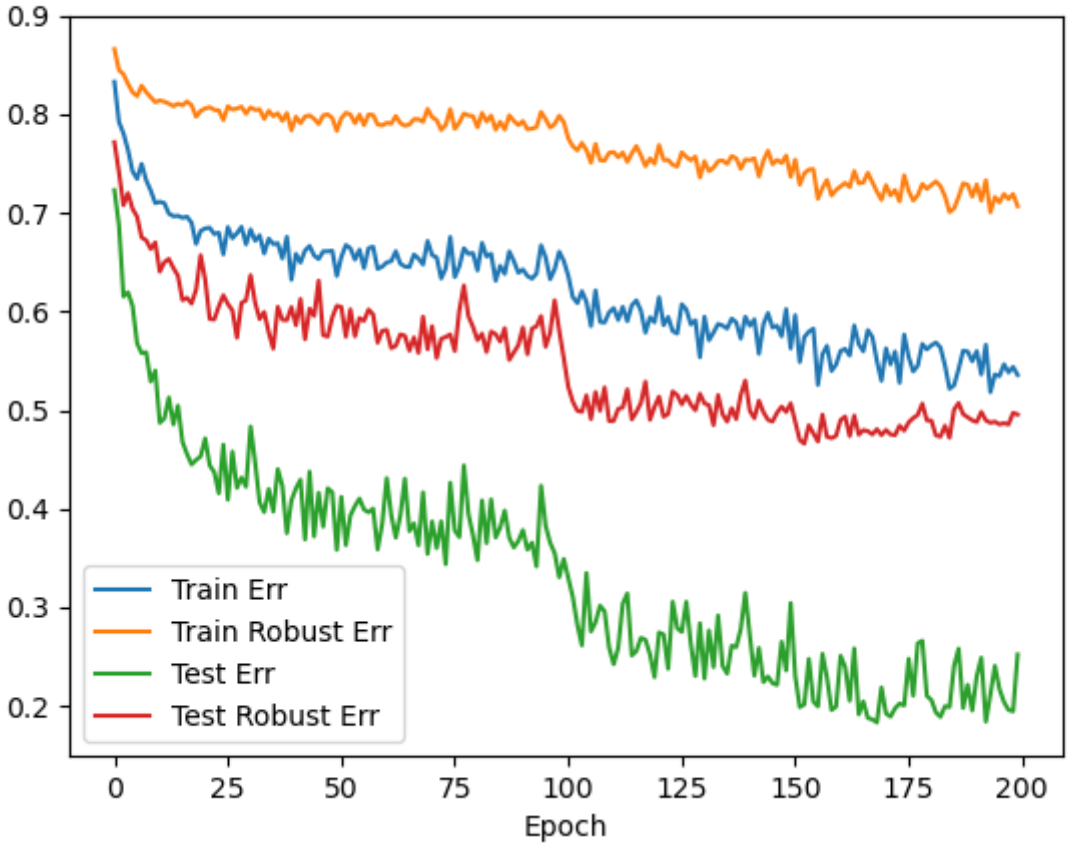
In this part we change the paramter value of mixup alpha.

Note that only when the mixup alpha is larger, such as > 1.0 , robust overfitting is not observed.

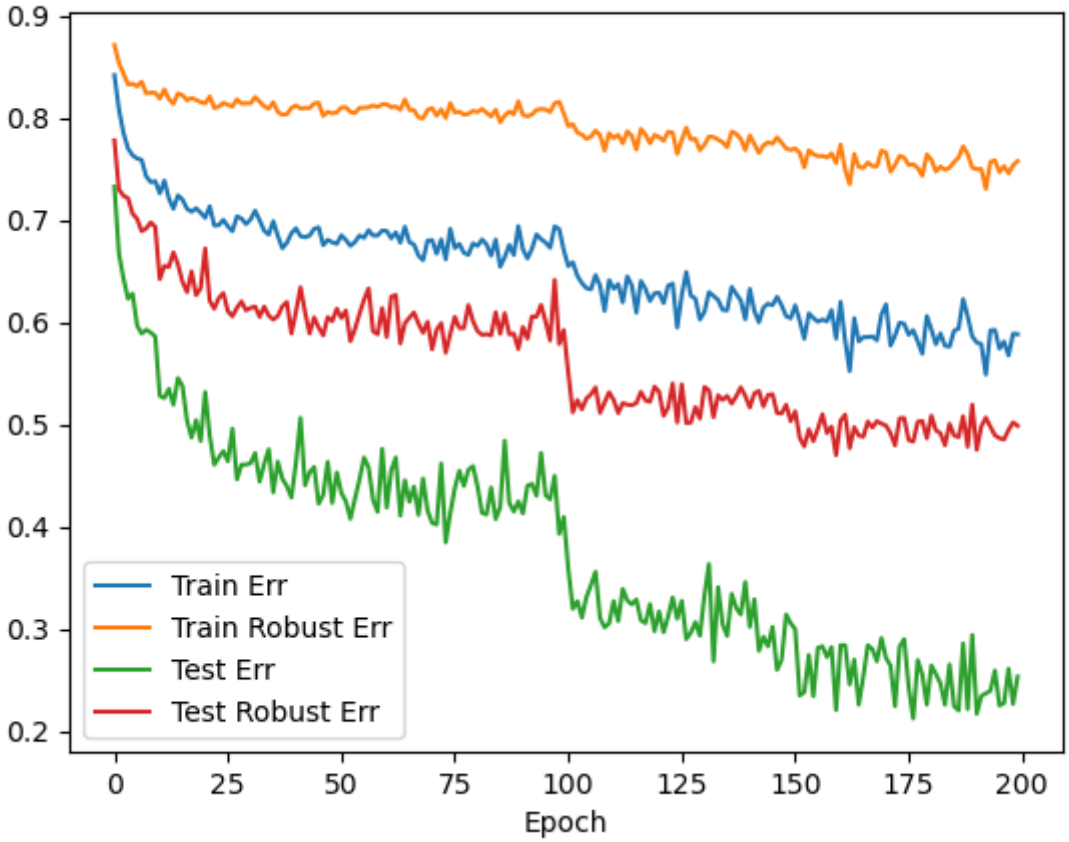
- Mixup alpha = 0.2



- Mixup alpha = 1.0



- Mixup alpha = 2.0



In [ ]:

```
# an example command of mixup alpha = 0.2
python3 train_cifar.py --fgsm-alpha 1 --fname 'experiments/cifar10_mixup/preactresnet18_0.2'  --mi
```

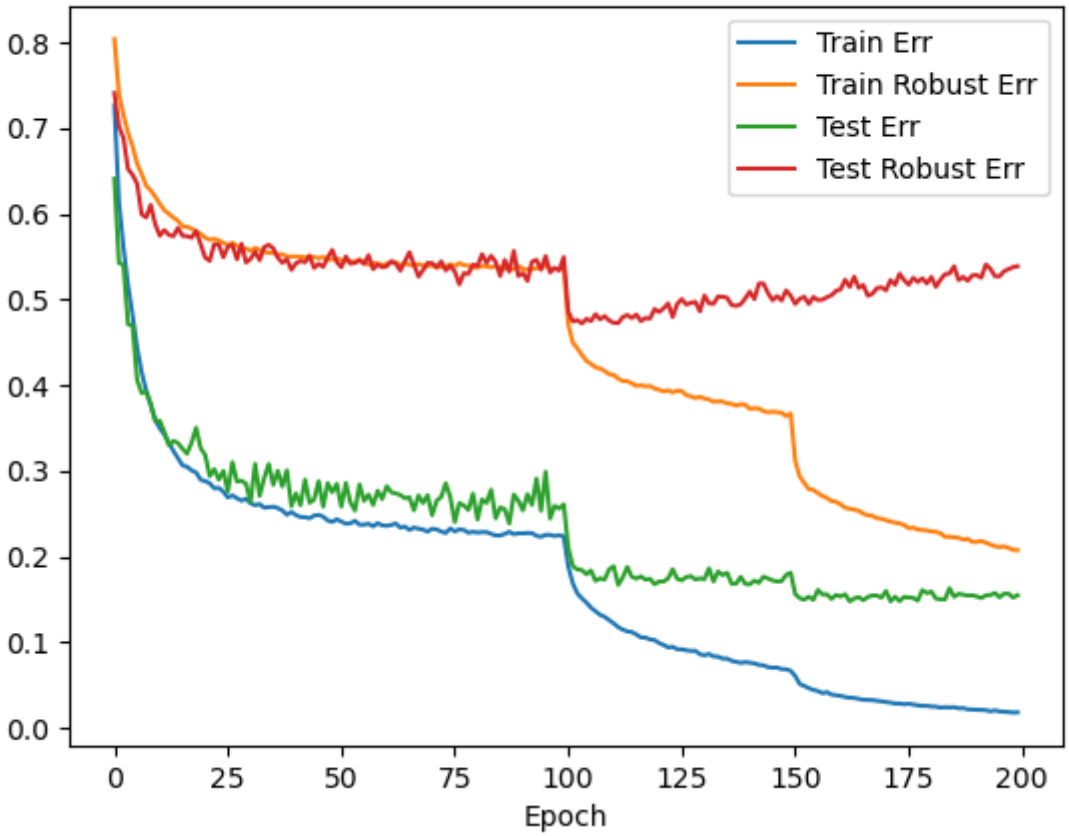
3.3 L1 & L2 Regularization

Explicit egularization refers to explicitly adding a term to the optimization problem, in our case, the loss function, to prevent overfitting and improve model generalization performance. It penalizes large parameter values and thus overfitting. We use both L1 and L2 regularization techniques.

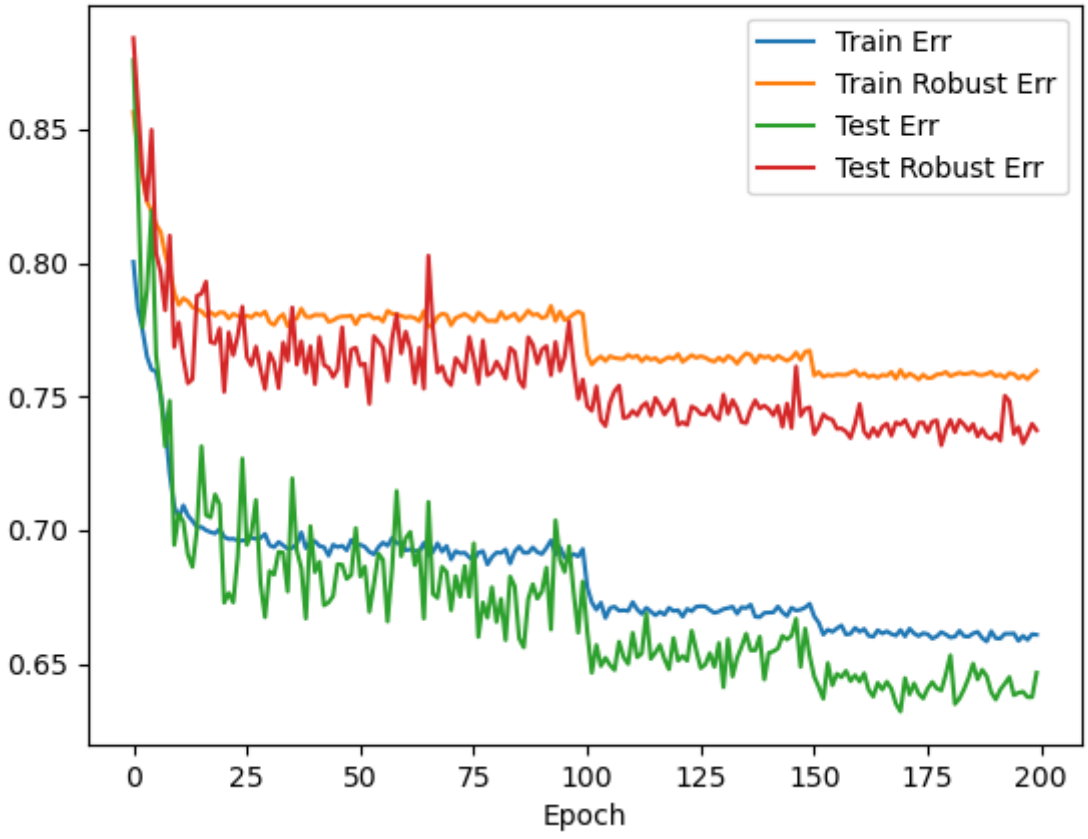
3.3.1 L1 Regularization

In this part we change the paramter value of l1.

- l1 = 0.0005


Train Err
Train Robust Err
Test Err
Test Robust Err
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
25
50
75
100
125
150
175
200
Epoch

- l1 = 5e-07


Train Err
Train Robust Err
Test Err
Test Robust Err
0.85
0.80
0.75
0.70
0.65
0
25
50
75
100
125
150
175
200
Epoch

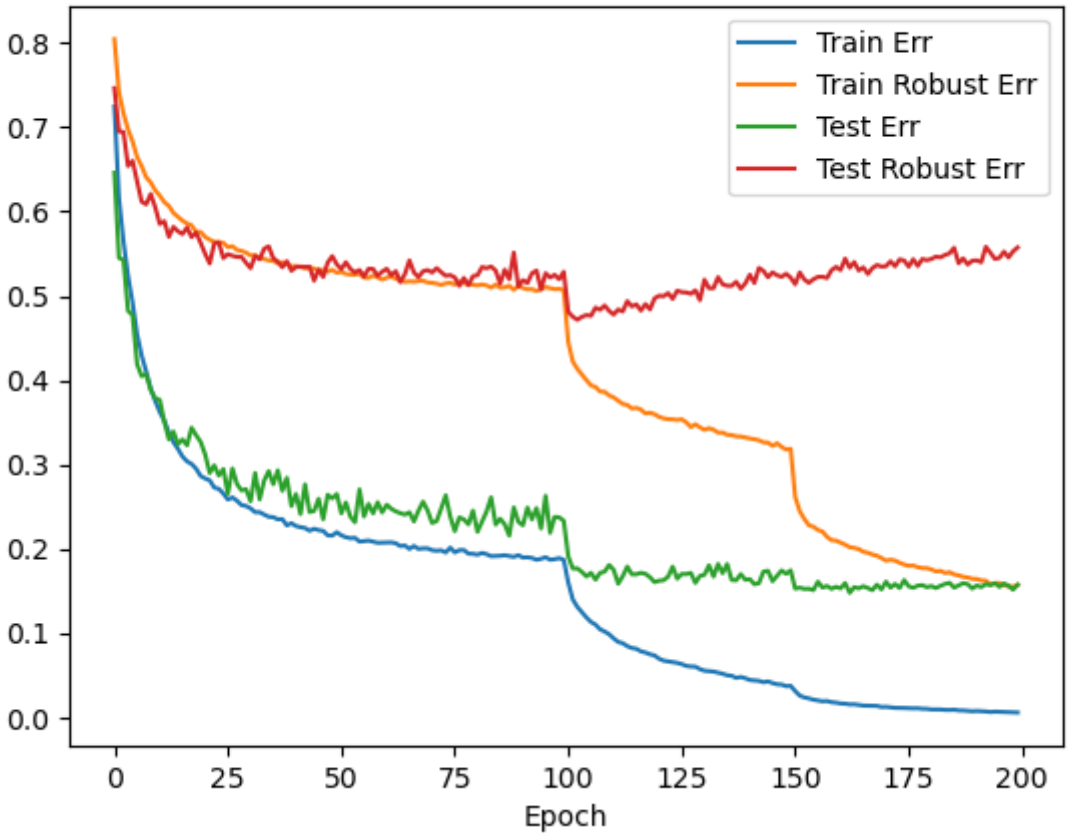
In [ ]:

```
# an example command of changing the parameter value of l1
python3 train_cifar.py --fgsm-alpha 1 --fname 'experiments/cifar10_l1/preactresnet18_5e-4' --l1 0.(
python3 train_cifar.py --fgsm-alpha 1 --fname 'experiments/cifar10_l1/preactresnet18_5e-7' --l1 5e-
```

3.3.2 L2 Regularization

In this part we show the result of l2 regularization.

- l2 = 0.0005



In [ ]:

```
# an example command of changing the parameter value of l2
python3 train_cifar.py --fgsm-alpha 1 --fname 'experiments/cifar10_l2/preactresnet18_5e-4' --l2 0.(
python3 train_cifar.py --fgsm-alpha 1 --fname 'experiments/cifar10_l2/preactresnet18_50' --l2 50.0
```

3.4 Standard training with validation set

In the early stopping with validation set method, we observe that at around 100 epochs the best robust test error is obtained.

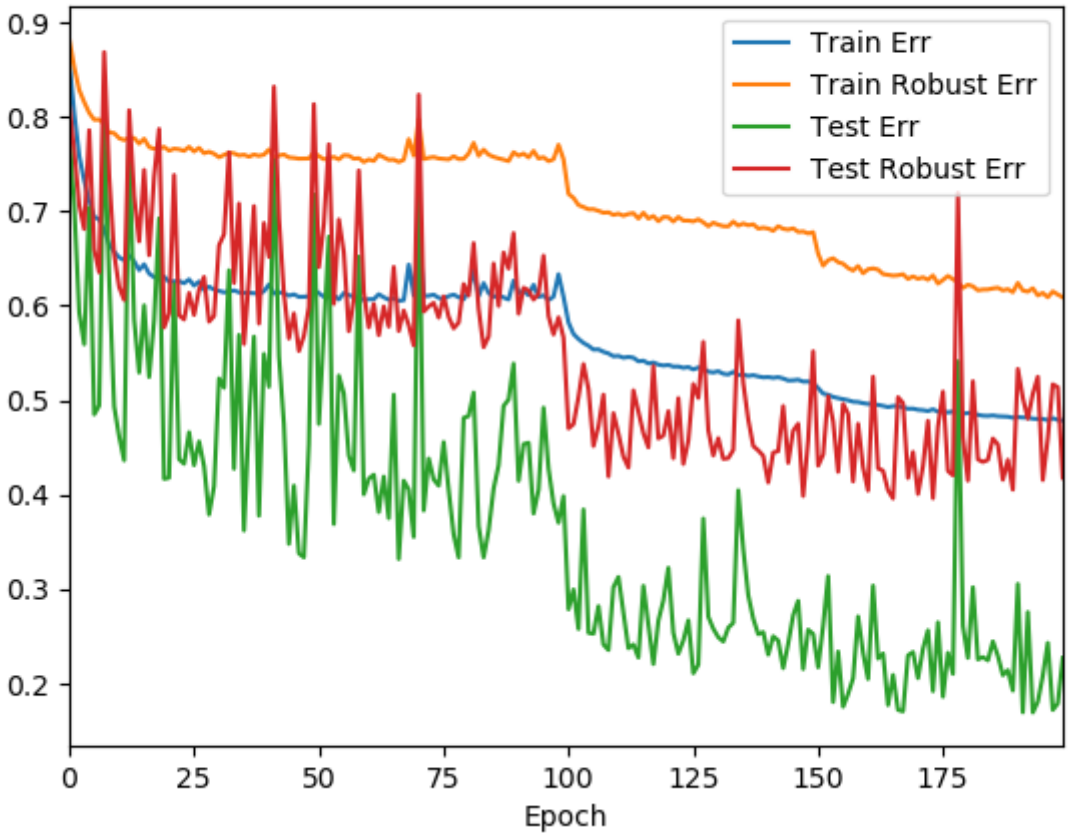
In [ ]:

```
# an example command of setting a validation set in the standard training
python3 train_cifar.py --fgsm-alpha 1 --fname 'experiments/cifar10_validation/preactresnet18' --va
```

3.5 Semi-supervised training

- CIFAR-10 training with semisupervised data is done in train_cifar_semisupervised_half.py, and uses the 500K pseudo-labeled TinyImages data from https://github.com/yaircarmon/semisup-adv (https://github.com/yaircarmon/semisup-adv).

Note that test error and test robust error oscillate a lot, but it can be seen that robust overfitting is not stark.



4. Conclusion

REG METHOD	Robust Test Error(%) FINAL	BEST	DIFF
EARLY STOPPING W/VAL	**46.9**	46.7	0.2
L1 REGULARIZATION	53.88	47.24	6.64
L2 REGULARIZATION	55.74	47.17	8.57
CUTOUT	50.45	46.94	3.51
MIXUP	49.56	47.2	2.36
SEMI-SUPERVISED	41.81	39.62	2.19

- This Table summarize our own reproduction results. As mentioned at the beginning of the reproduction, we can check differences between final and the best robust test errors. As shown in the table, the overall trend and values of robust test error are similar to those of the original experiments results in the paper. Through extensive experiments, we could therefore reach the following conclusions:
- Found that early stopping, compared to other methods, is the most effective way to solve robust overfitting.
- Tried L1 and L2 regularization, mixup cutout, semi-supervised learning, and found that these methods can alleviate robust overfitting, but are not as good as early stopping.

5. Extensions

Paper 1:

- multidimensional training dataset
- extend SVM
  - with RBF kernel
  - on linearly inseparable data
- uniform data rather than normal for classification

Paper 2:

- add dropout
- activation functions (not sure yet)
- adjust attack strength on image data

Both:

- try more attack methods

In [ ]: