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

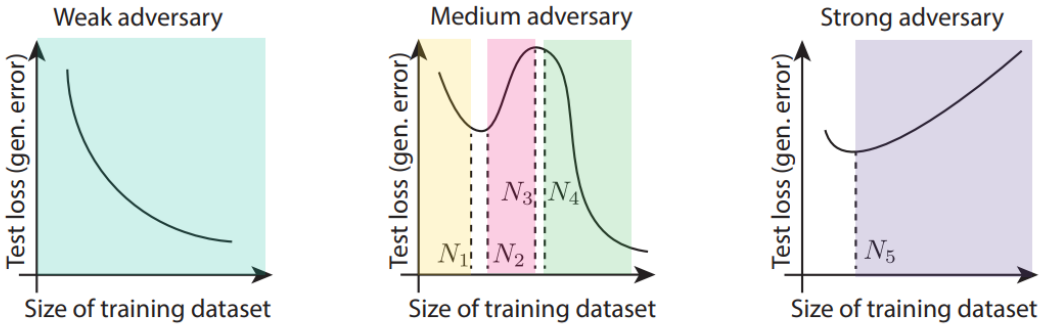
The Curious Case of Adversarially Robust Models: More Data Can Help, Double Descend, or Hurt Generalization. *Yifei Min, Lin Chen, Amin Karbasi*

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. Three Adversary 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 Extensions

After we have re-implemented experiments in the 1-dimensional setting and with perturbation of only L_∞ norm as described in the original paper, we extend the hypothesis of different adversary regimes to more scenarios, specifically, on **multi-dimensional data** with **various attack methods**, including analytical optimal attack, FGSM with L_∞ norm, FGM (FGSM with L_2 norm), as well as PGD with L_∞ norm, and apply similar attacks to related models.

In Section 3.1, we analyze the **Gaussian mixture model with linear loss** with multi-dimensional data, under different attack methods, 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 the **linear regression model with mean squared loss** with multi-dimensional data, under different attack methods 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** with multi-dimensional data, under different attack methods. We find that for small, medium, and large ε can help, double descend, or hurt generalization.

In Section 3.4, we show results of **Gaussian process regression** with 1-dimensional data under FGSM attack and compare with linear regression results.

In Section 3.5, we extend SVM model to **SVM with RBF kernel** with 2-dimensional data under FGSM and FGM attacks.

3.1 Gaussian Mixture with Linear Loss

Following are the result plots showing test loss with $\mathbf{1, 5, 10}$ dimensions and 4 different attacks $\mathbf{(OPT, FGM, FGSM, PGD)}$.

Fig. 3.1a: OPT in 1, 5, 10 dimensions.

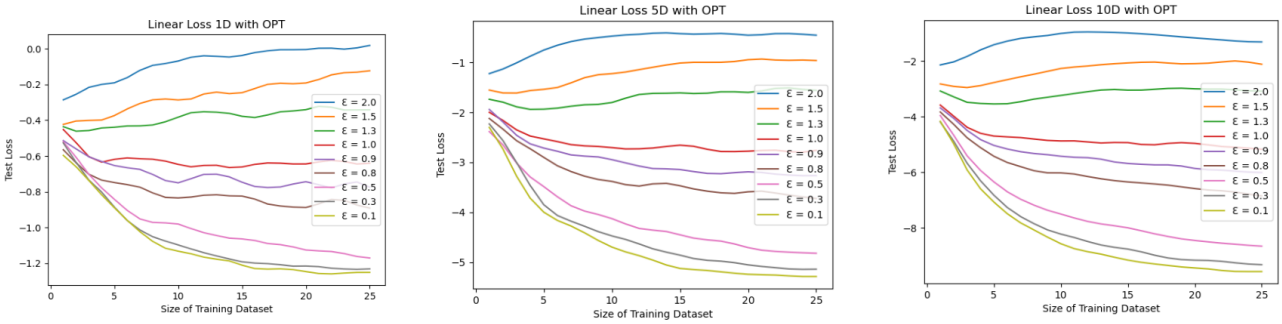


Fig. 3.1b: FGSM in 1, 5, 10 dimensions.

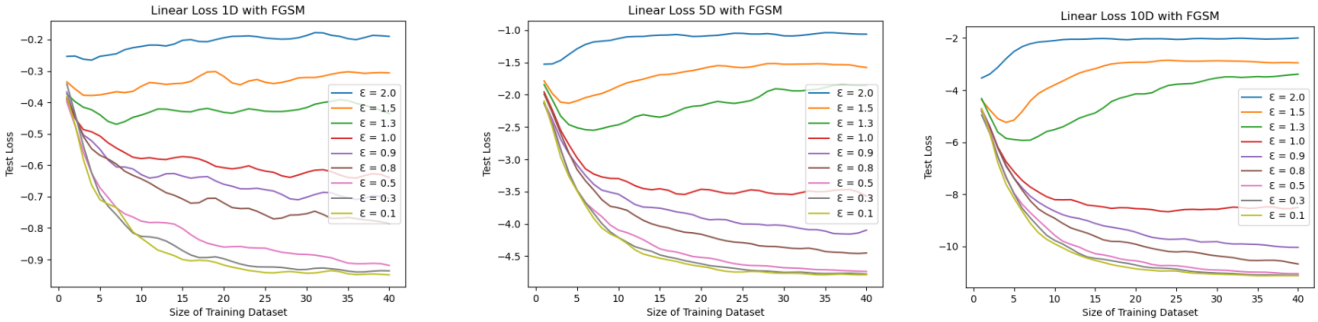


Fig. 3.1c: FGM in 1, 5, 10 dimensions.

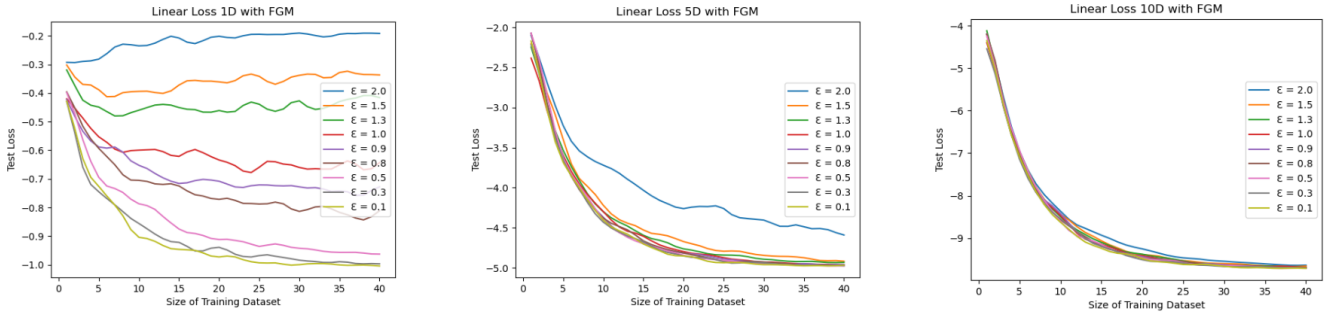


Fig. 3.1d: FGM+sqrt(dim) in 5 and 10 dimensions.

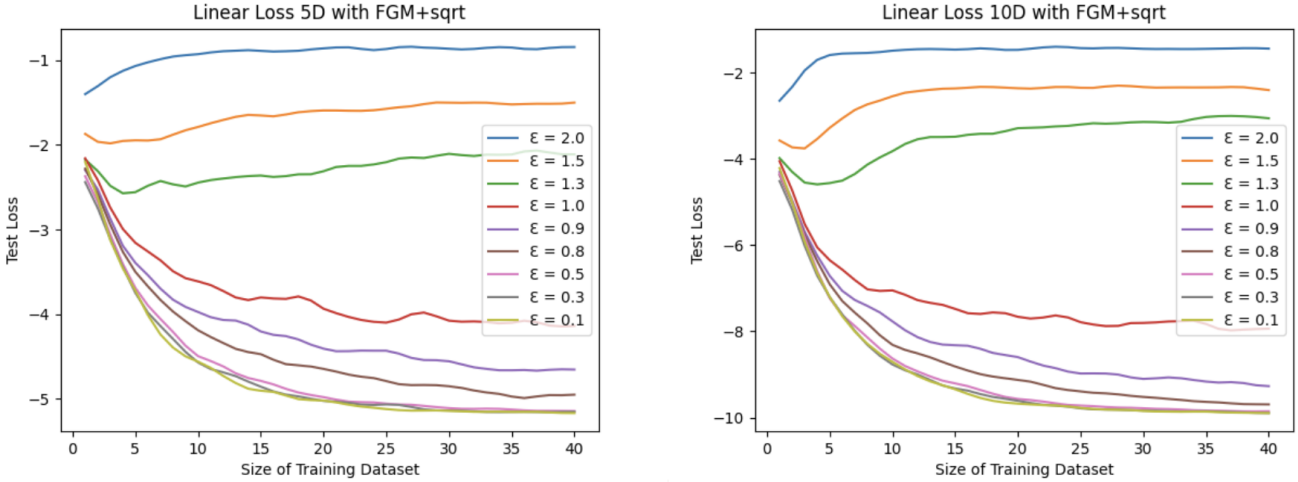
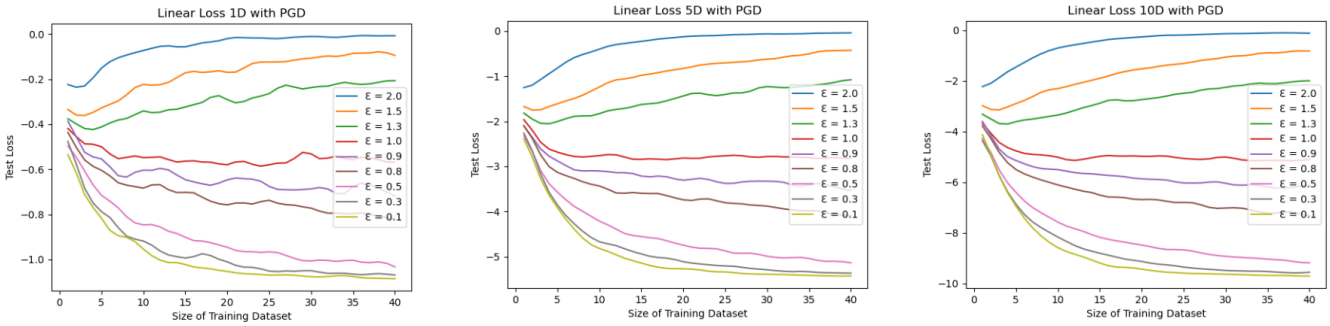


Fig. 3.1e: PGD in 1, 5, 10 dimensions.



Discussion

In ***Figure 3.1a and 3.1b*** (OPT and FGSM, respectively), we can clearly see common trends in both low dimensional data and high dimensional data. By adjusting the parameter ϵ, the shapes of test loss curves would differ and show the three adversary regimes described in Section 2. The math below shows that **FGSM is is exactly the optimal attack against a linear binary classification model under the L_∞ norm**.

We first see **how the optimal solution is derived**. The optimization problem is described by the objective to maximize the loss given some perturbation δ under the constraint that the p-norm (for now, $p \to \infty$) of δ is no larger than ϵ:

$$\underset{||\delta||_p \leq \epsilon}{max}\ \ell = -y \cdot \langle w, \tilde{x} \rangle = -y \cdot (w^T \tilde{x}) = -y \cdot (w^T (x + \delta))$$

where x is the original training point, $\tilde{x} = x + \delta$ is the perturbed training point, y is a scalar training label being either -1 or $+1$, and w is the linear model parameters.

Or we could write the optimization problem as a minimization, due to the negative sign:

$$\underset{||\delta||_p \leq \epsilon}{min}\ y \cdot (w^T (x + \delta))$$

Since we are solving for the optimal δ, only the terms involving δ matters:

$$\underset{||\delta||_p \leq \epsilon}{min}\ y \cdot (w^T \delta)$$

When $y = +1$, the objective is clearly minimized when $\delta_i = -\epsilon$ for $w_i \geq 0$ and $\delta_i = \epsilon$ for $w_i \leq 0$. For $y = -1$, these quantities are simply flipped. Thus, for $p = \infty$, the optimal perturbation is:

$$\delta^\star = -y\epsilon \cdot \text{sign}(w)$$

and the modified loss is:

$$y \cdot w^T \delta^\star = y \cdot \sum_{i=1}^{d} -y\epsilon \cdot \text{sign}(w_i) w_i = -y^2 \epsilon \sum_i |w_i| = -\epsilon \|w\|_1$$

, where d denotes the number of dimensions.

For FGSM, the perturbation is characterized as:

$$\delta := \epsilon \cdot \text{sign}(\nabla_\delta \ell) = \epsilon \cdot \text{sign}(-y \cdot w) = -y\epsilon \cdot \text{sign}(w) \equiv \delta^\star$$

Thus, FGSM is the optimal attack for the current model.

In ***Figure 3.1c***, we can see that for 1-dimensional data, FGM has the similar effect, but for high dimensional data of 5 and 10 dimensions, FGM shows a continuous convergence for all ε's. Thus, FGM is not as strong an attack method as FGSM. This can be understood by the following math. We use projedcted normalized steepest descent method (see the report) for the L_2 ball:

$$\delta := \mathcal{P}_\epsilon \left(\delta - \alpha \frac{\nabla_\delta \ell(h_\theta(x+\delta), y)}{\|\nabla_\delta \ell(h_\theta(x+\delta), y)\|_2} \right)$$

, where P_ϵ now denotes the projection onto the L_2 ball of radius ϵ, and perturbation δ has the following form:

$$\mathcal{P}_\epsilon(z) = \epsilon \frac{z}{\max\{\epsilon, \|z\|_2\}}$$

, where z denotes everything inside $P_\epsilon(\cdot)$. Note that now perturbation δ cannot have a larger L_2 norm than ϵ, whereas in the L_∞ case, δ can have ϵ perturbation in each dimension. As the maximum perturbation is worsened as dimension increases for FGM, it performs weaker attack than the other attack methods.

In ***Figure 3.1d***, for FGM, we multiply the perturbation by $sqrt(d)$, so that the maximum perturbation achieved by L_2 norm is equal to that of L_∞. The result shows that after scaling, the three regimes are evident in higher dimensions for FGM, too.

In ***Figure 3.1e***, PGD shows similar results as OPT and FGSM. This can be explained by the iterative nature of PGD:

$$\begin{aligned} &\text{Repeat:} \\ &\quad \delta := \mathcal{P}(\delta + \alpha \nabla_\delta \ell) \end{aligned}$$

, where $\mathcal{P}$ denotes the projection onto the L_∞ ball. Setting $\alpha = \epsilon$ yields valid single-step perturbation without projection: FGSM. For the current one, w would progress with epochs and PGD would yield quite close to optimal result given a number of iterations.

3.2 Linear regression model with the squared loss

In contrast to the linear classification model, we were very curious to see how linear regression would perform in the cases of **multi-dimensional data** und **4 different attack methods**(OPT, FGSM, FGM, PGD). Here the training data are under ***Gaussian distribution*** $N(0, 1)$ and shifted ***Poisson distribution*** $Poisson(5) + 1$ respectively for experiments and comparisons.

Fig. 3.2a: OPT in 1, 5, 10 dimensions Gaussion.

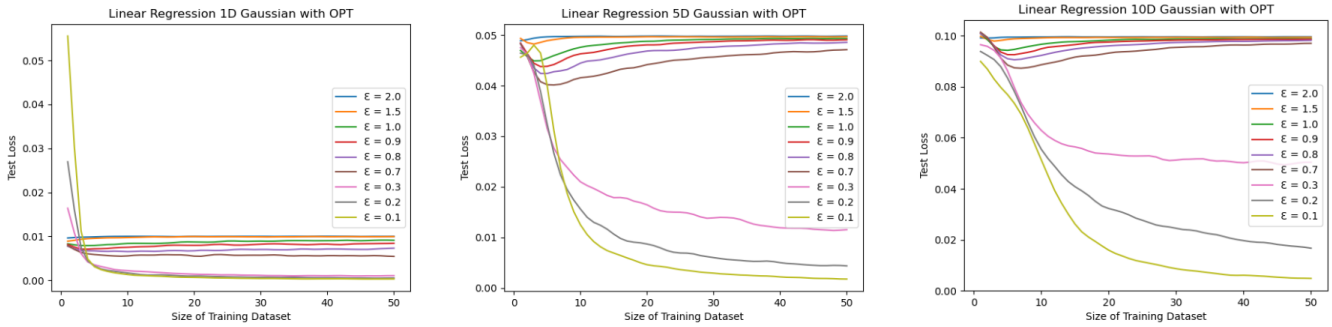


Fig. 3.2b: FGSM in 1, 5, 10 dimensions Gaussion.

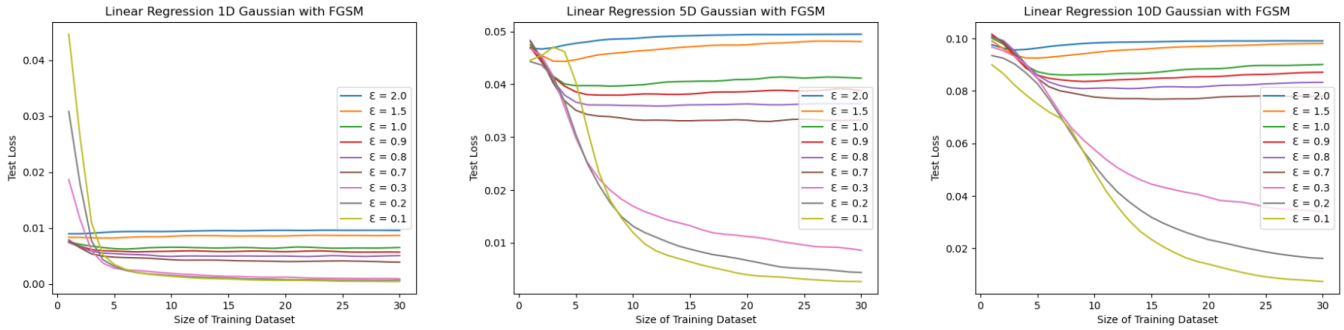


Fig. 3.2c: FGM in 1, 5, 10 dimensions Gaussion.

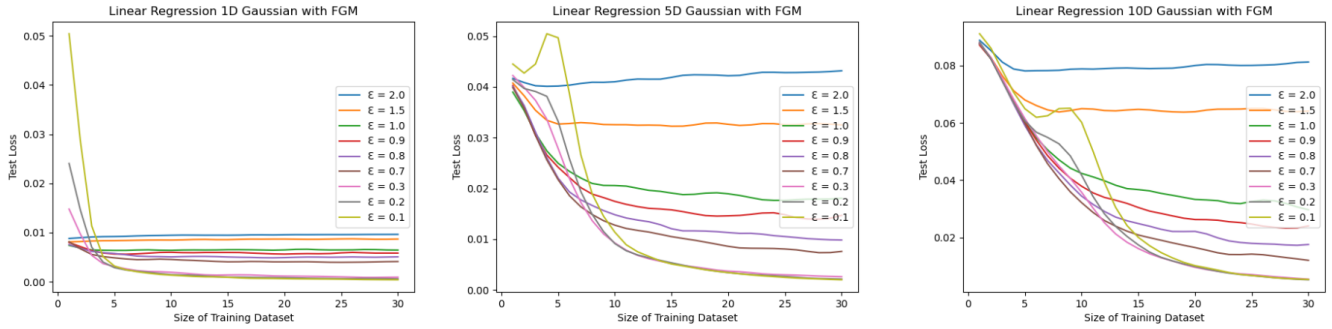


Fig. 3.2d: FGM+sqrt in 5 and 10 dimensions Gaussion.

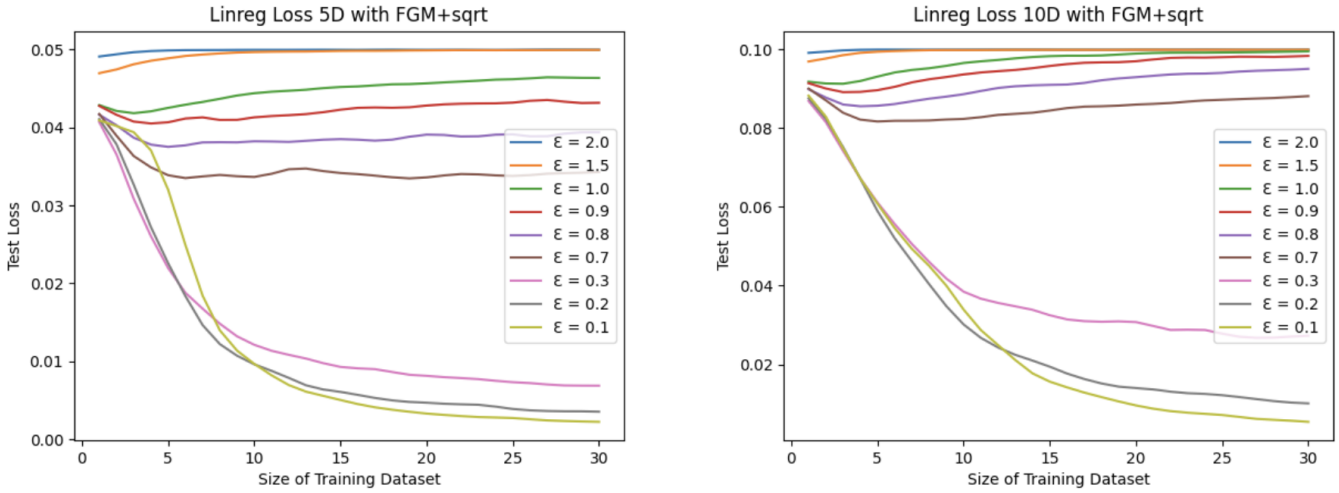
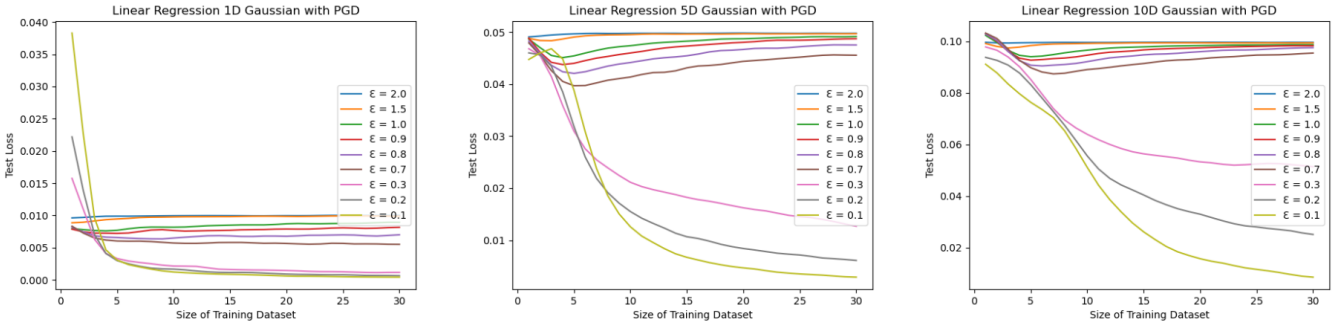


Fig. 3.2e: PGD in 1, 5, 10 dimensions Gaussion.



Discussion

We see that FGSM yields results quite similar to OPT again. The OPT method is shown in Observation 5 in [1]. However, **FGSM is no longer the optimal attack method for linear regression**, mainly because the loss is now MSE rather than linear. It should be noted that test losses for PGD attacks almost coincide with that of the OPT.

In ***figures 3.2d***, we see again that FGM has stronger attack after ϵ is scaled by $\sqrt{d}$.

In ***figures 3.2a, b, c, e***, we see the three adversary regimes. As epsilon increases, test loss keeps rising. But for higher dimensions such as 5 and 10, the trend is clear because there is **a larger gap between the convergences of weak and strong regimes because test loss is roughly proportional to the number of dimensions** (due to the inner product between w and $\tilde{x}$).

However, for 1-dimensional data, as the gap is not as large, we could not see the upward trend of strong regime clearly. To observe more closely, we choose to split the three regimes for 1D data.

Fig. 3.2e: OPT in 1 dimension Gaussion.

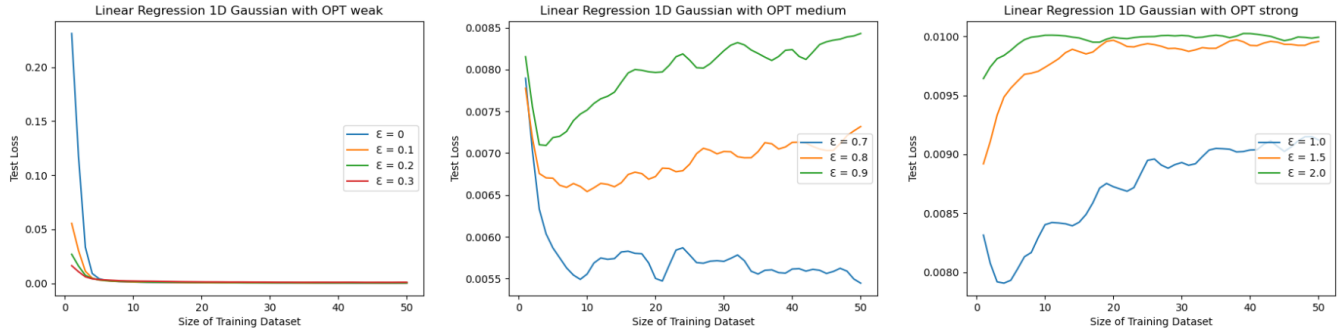


Fig. 3.2f: FGSM in 1 dimension Gaussion.

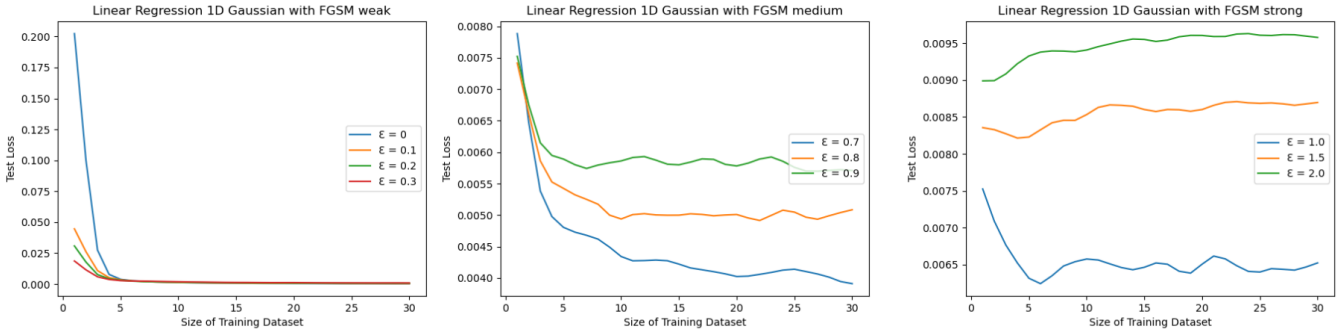


Fig. 3.2g: FGM in 1 dimension Gaussion.

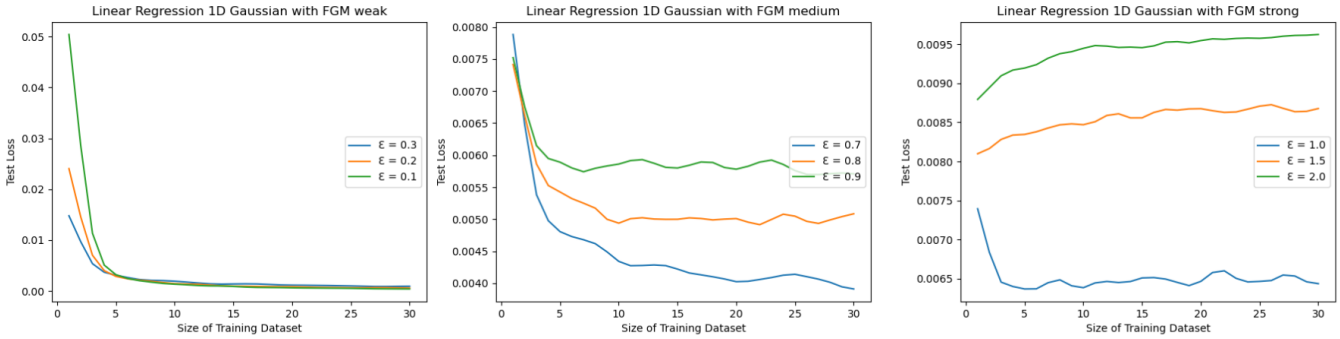
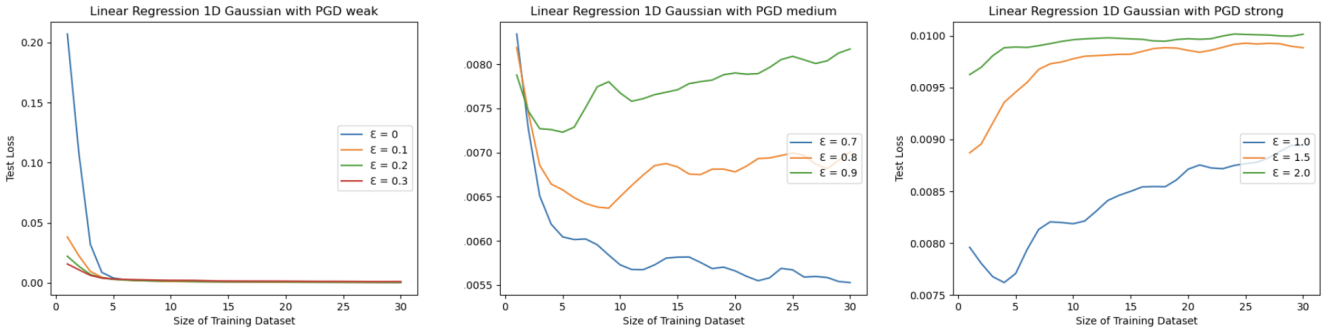


Fig. 3.2h: PGD in 1 dimension Gaussion.



Let's take the next step to see how the results change for the ***Poisson*** distribution.

Fig. 3.2i: OPT in 1, 5, 10 dimensions Poisson.

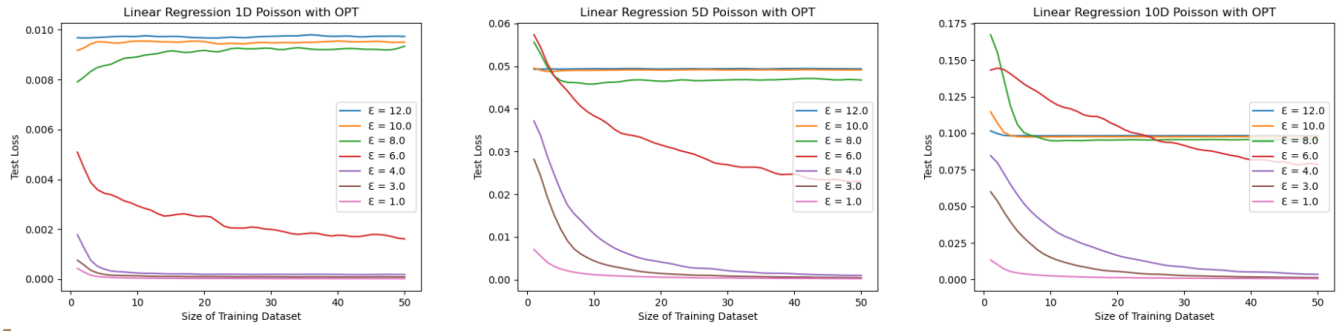


Fig. 3.2g: FGSM in 1, 5, 10 dimensions Poisson.

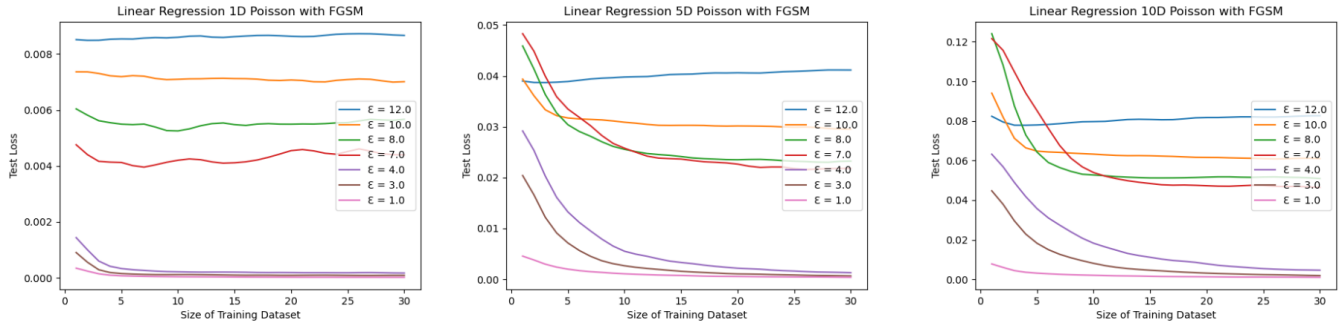


Fig. 3.2k: FGM in 1, 5, 10 dimensions Poisson.

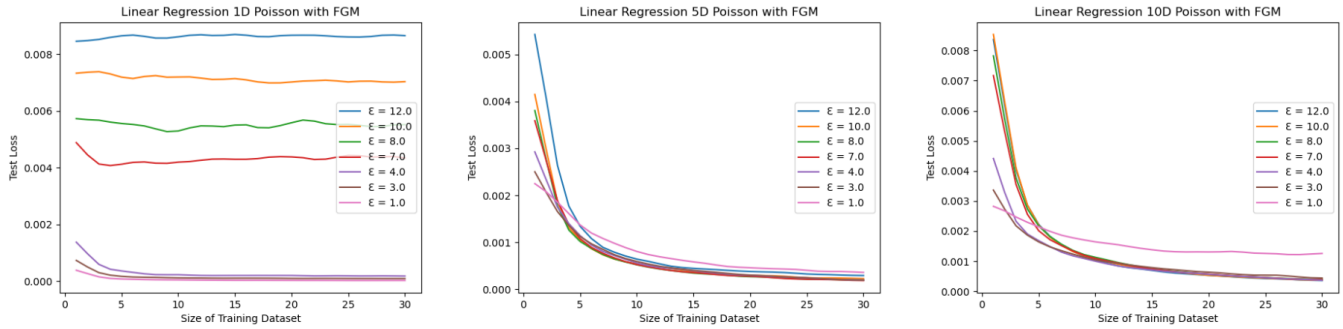


Fig. 3.2l: PGD in 1, 5, 10 dimensions Poisson.

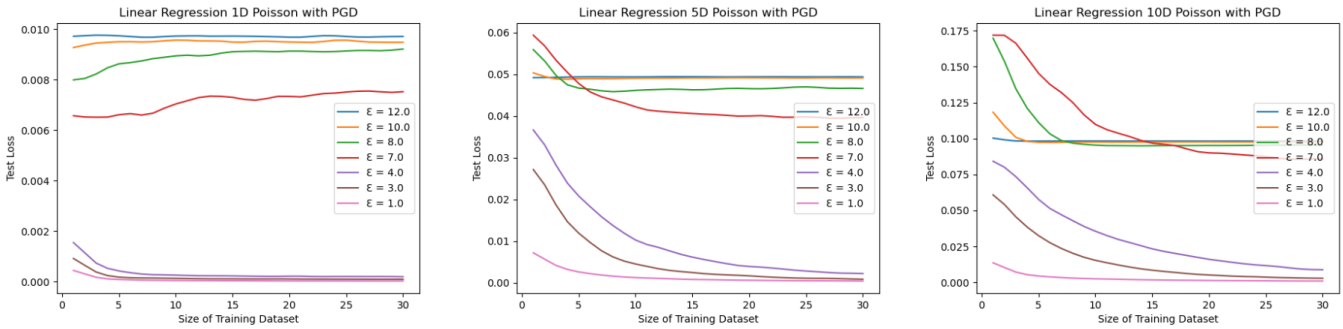
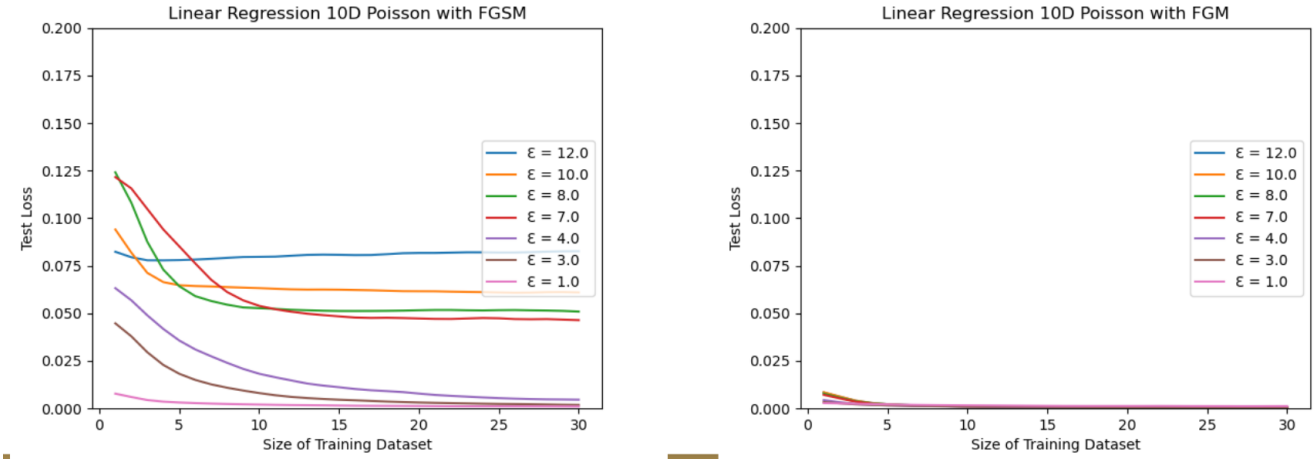


Fig. 3.2m: FGSM vs FGM



Discussion

While the previous Gaussian distribution has medium adversary for smaller ϵ's, now it takes more perturbation for Poisson data, namely, $6 - 8$. The trends of test losses curves for higher-

dimensional case are similar to what we saw in 1-dimensional case.

In ***Figure 3.2k*** and comparing FGSM and FGM under the same scale in ***Figure 3.2m***, we can see again that FGM is not as strong as other attack methods.

3.3 Support Vector Machine

We study the soft-margin support vector machine with hinge loss in testing:

$$\max\{0, 1 - y(\langle w, x\rangle - b)\}$$

For training, the following regularization term is added:

$$\frac{1}{2}c\|w\|_2^2$$

, where we set c to be 0.1 for all the results shown below.

The training data dimension **d** equals **2, 5, 10** and results are shown as follows:

Fig 3.3a: FGSM three regimes in 2 dimensions.

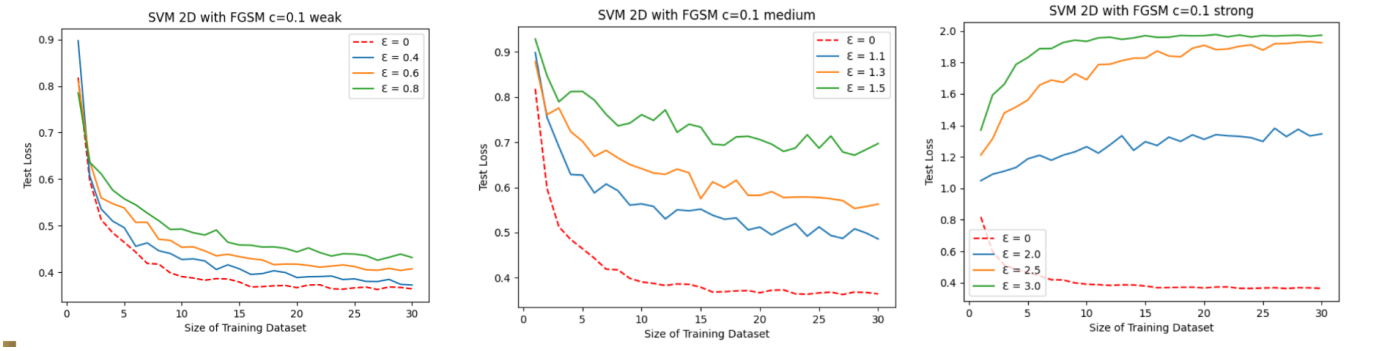


Fig 3.3b: FGSM three regimes in 5 dimensions.

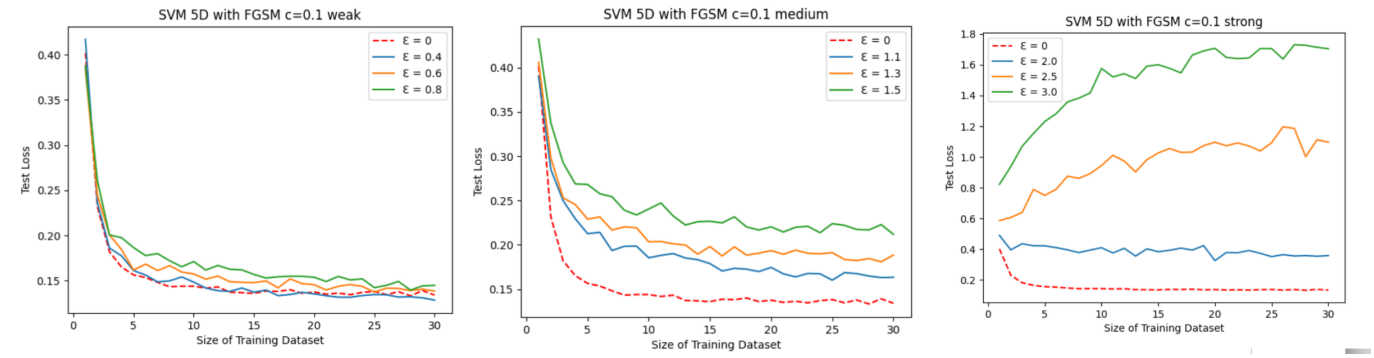
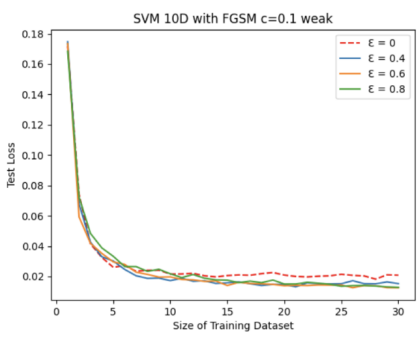
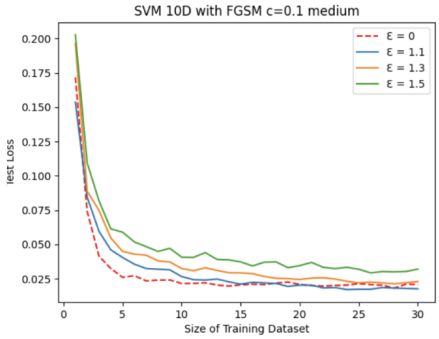


Fig 3.3c: FGSM three regimes in 10 dimensions.





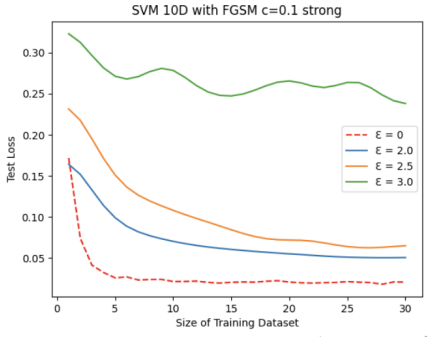
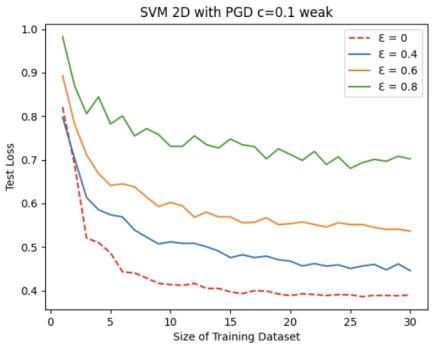
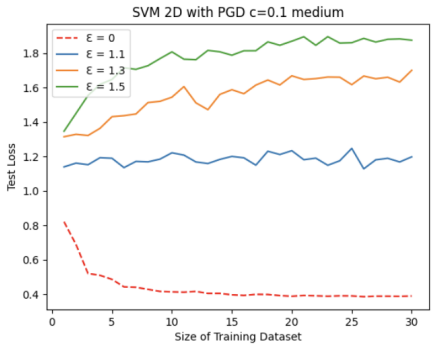


Fig 3.3d: PGD three regimes in 2 dimensions.





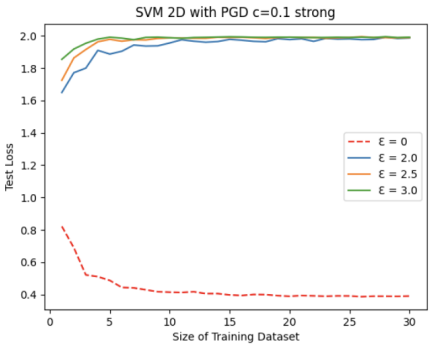
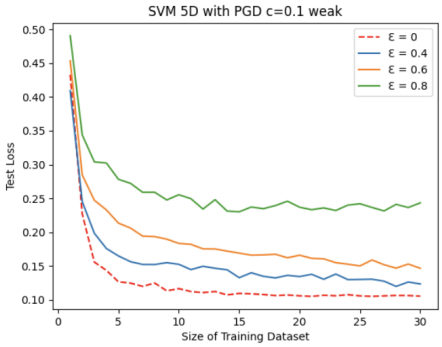
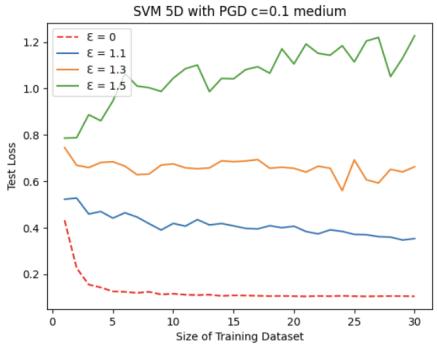


Fig 3.3e: PGD three regimes in 5 dimensions.





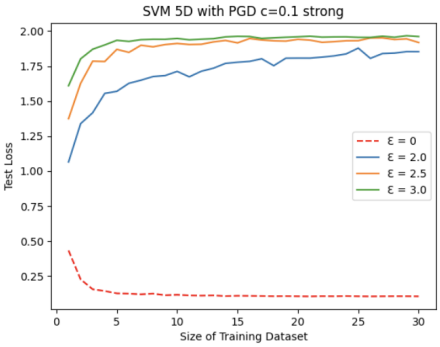
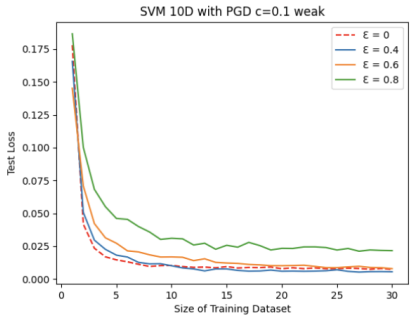
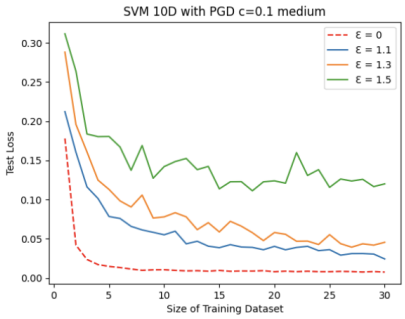
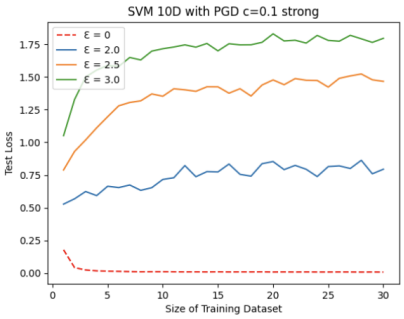


Fig 3.3f: PGD three regimes in 10 dimensions.







Discussion

As shown in ***figures 3.3a, b, c***, for FGSM, as dimension increases, the test loss curves in strong adversary regime yield a more and more downward trend, meaning that FGSM in SVM is also not optimal.

As shown in ***figures 3.3d, e, f***, for PGD, as dimension increases, we see similar trends in all three adversary regimes. This accords with the general case that PGD is close to optimal attack.

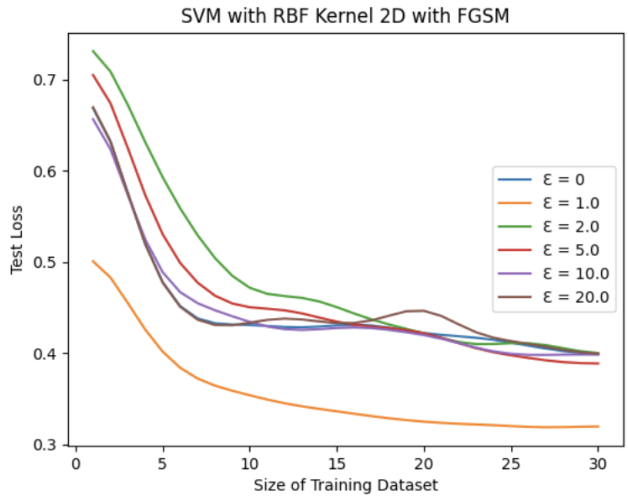
3.4 SVM with RBF Kernel

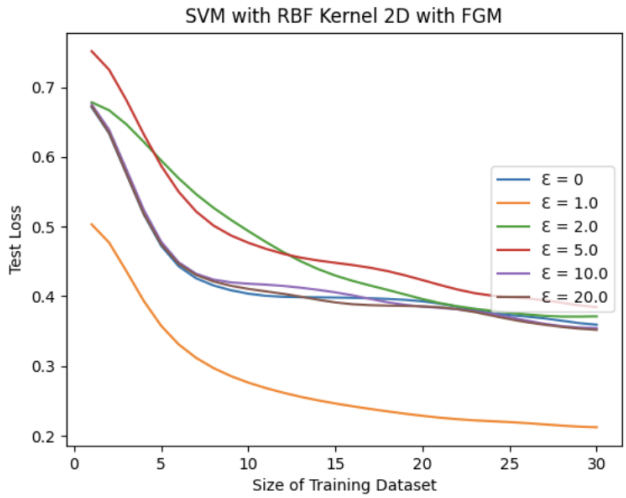
As Section 3.3 shows, the SVM model can easily be perturbed by attack methods such as FGSM and PGD. Thus, we desire to investigate a more robust SVM model, namely, adding the RBF (Radial Basis Function) kernel to make the model a universal approximator:

$$\mathrm{K}(\mathbf{x}, \mathbf{z}) = e^{\frac{-\|\mathbf{x}-\mathbf{z}\|^2}{\sigma^2}}$$

The results for FGSM and FGM attacks with 2-dimensional training data are showns below.

Fig 3.4: FGSM and FGM in 2 dimensions.





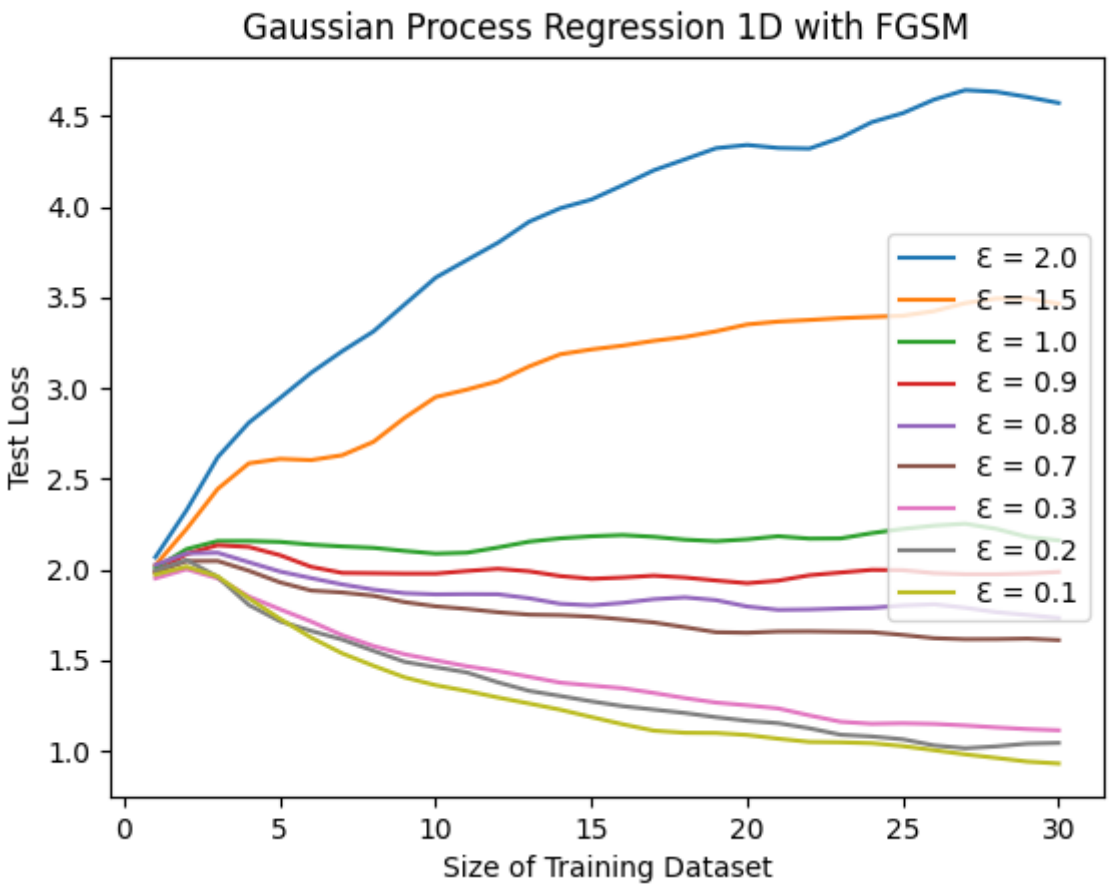
Discussion

When ϵ goes as high as 20 for SVM with RBF kernel, both attacks fail to yield an upward trend for test loss curves. This remarkable result shows that due to the high dimensions of the kernel matrix, the **model is very robust against single-step adversarial attacks** such as FGSM and FGM.

3.5 Gaussian Process Regression

As seen in Section 3.2, linear regression is easily perturbed under various attack methods, we yearn to try a powerful regression algorithm that works well with a small training data set size: Gaussian Process Regression. In training, we utilize negative log marginal likelihood, whereas in testing stage, we use the MSE loss to evaluate the difference of the mean of the posterior distribution and target label, so that the loss can be compared with linear regression.

The result is shown below.



Discussion

With FGSM attack for 1-dimensional data set, we could still see the three adversary regimes shown in the paper. Therefore, **Gaussian Process regression is not much more robust than linear regression under FGSM attack**.

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* [2]

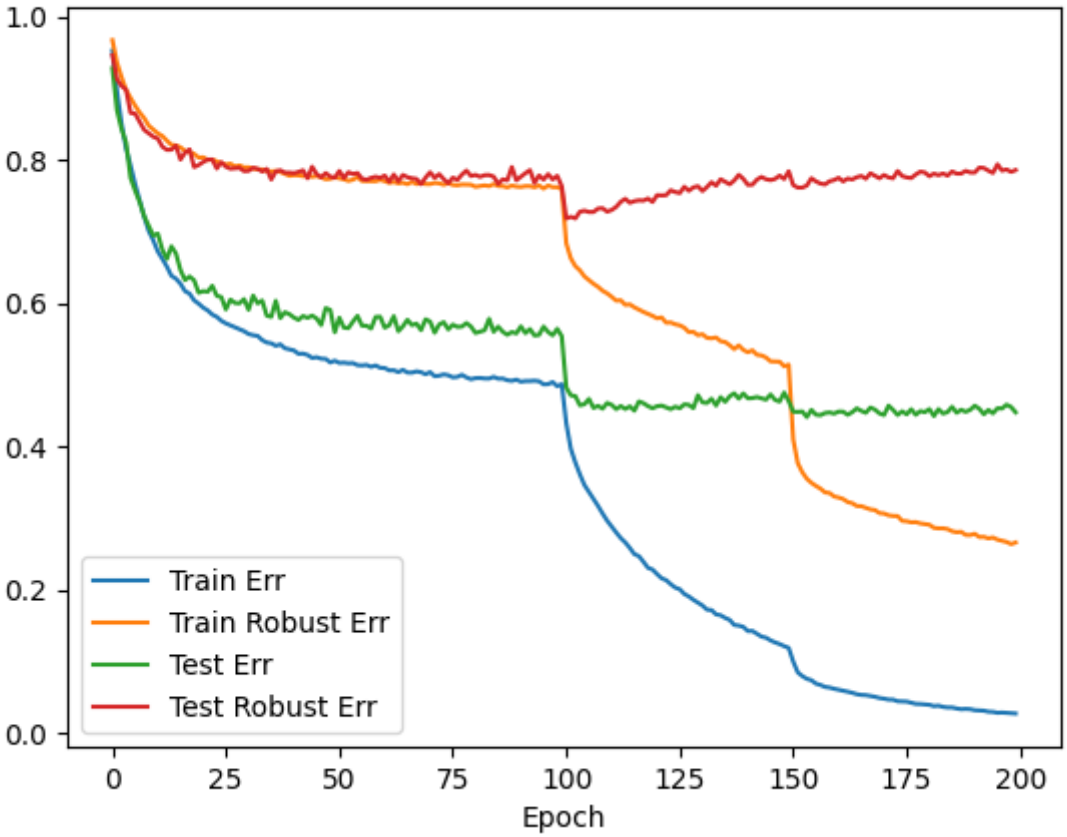
1. Introduction

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

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

We use the the following 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 effectiveness 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.

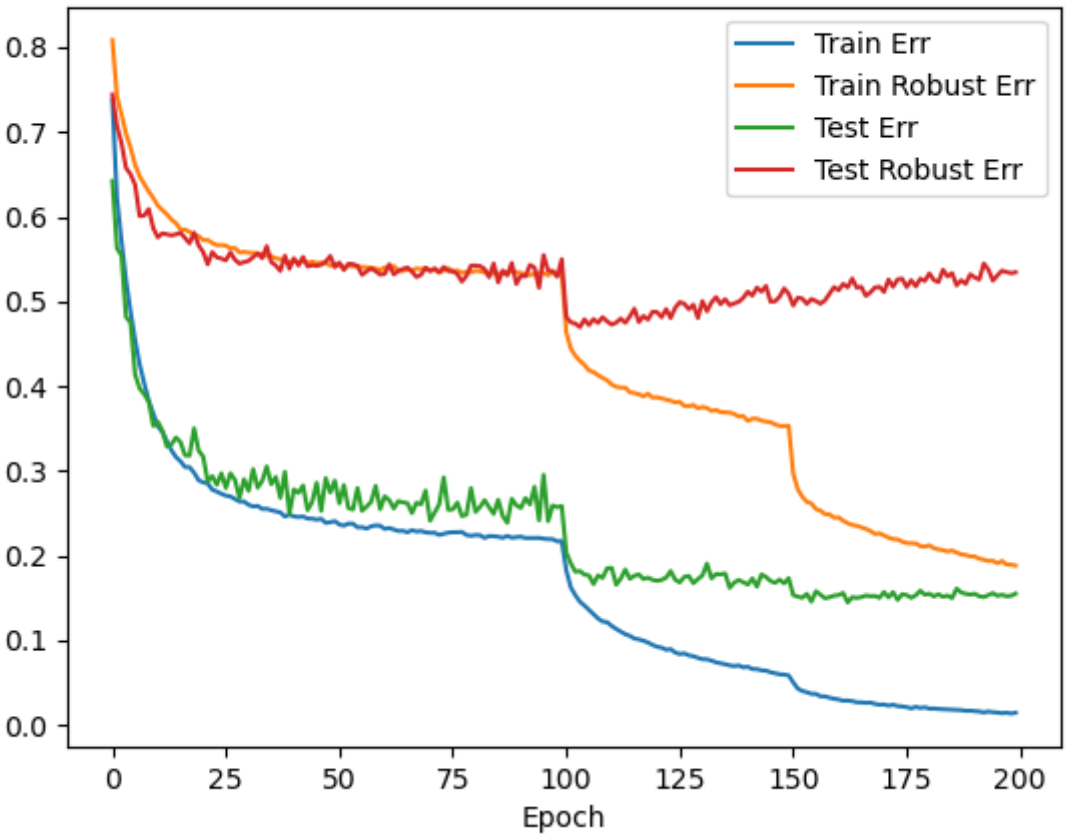
3.1 Cutout

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.

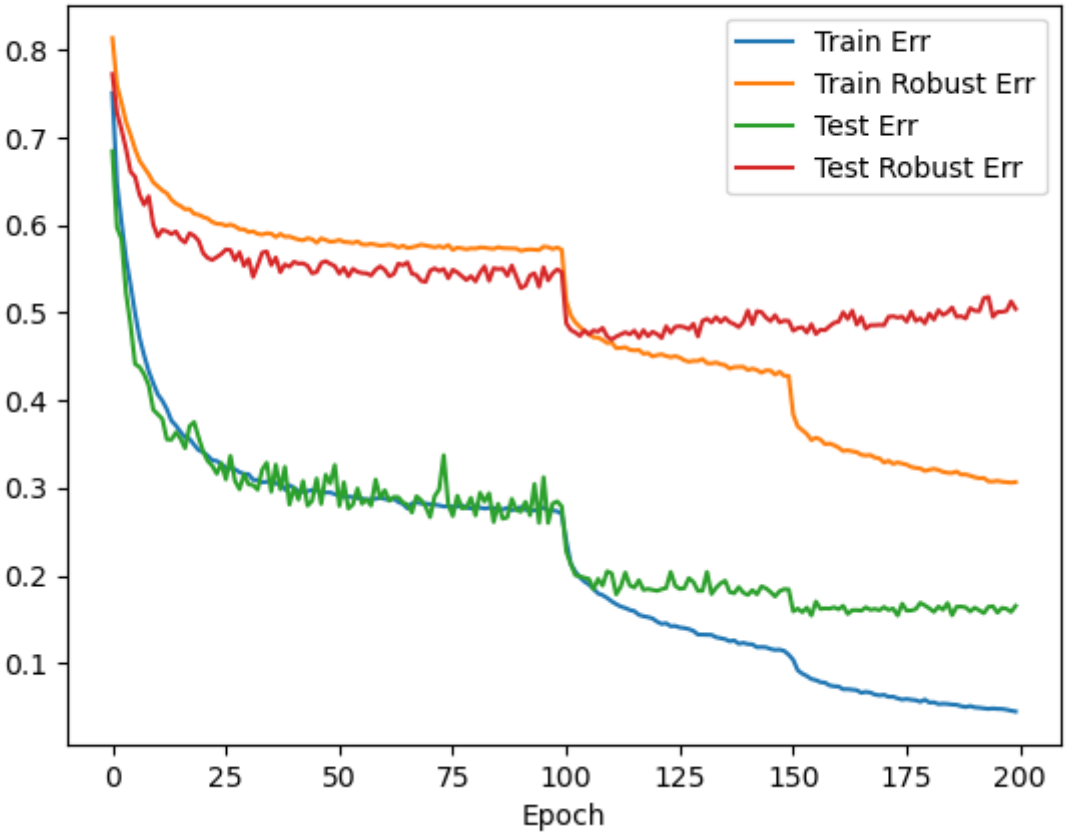
In this part, we set the parameter value of the cutout as: 2, 10, 20.

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

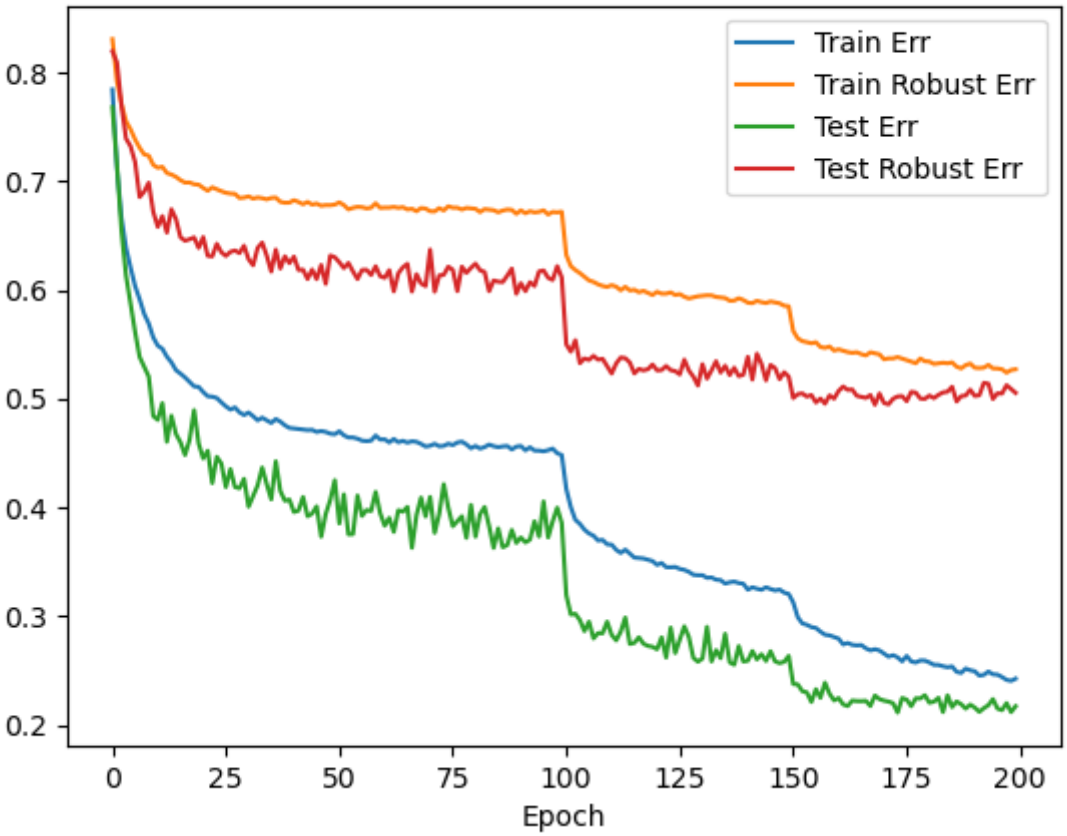
- Cutout len = 2



- Cutout len = 10



- Cutout len = 20



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

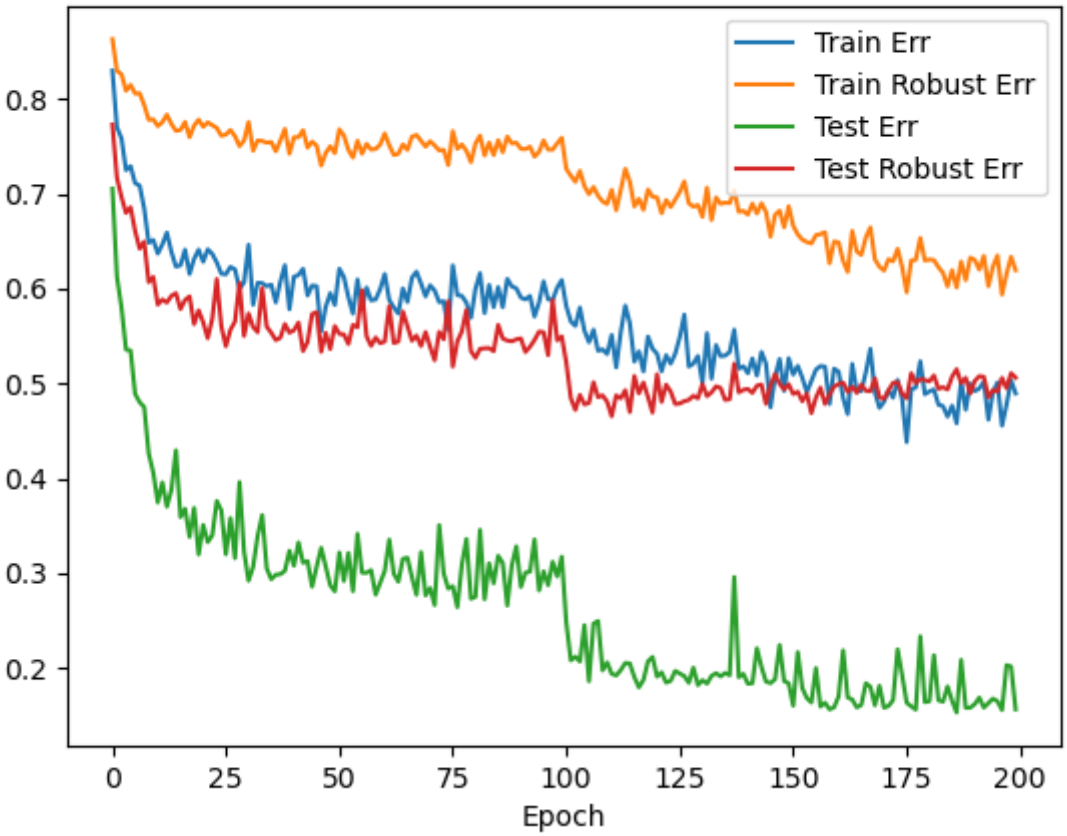
3.2 Mixup

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.

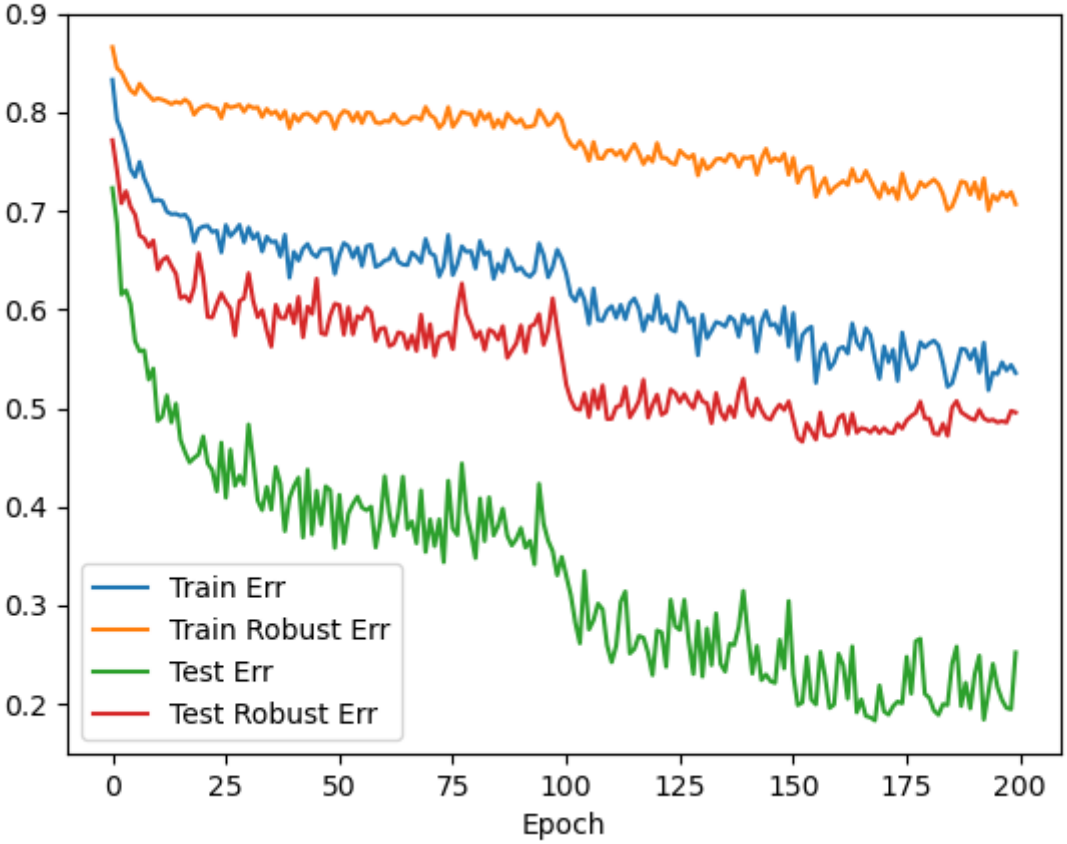
In this part we change the paramter value of mixup alpha.

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

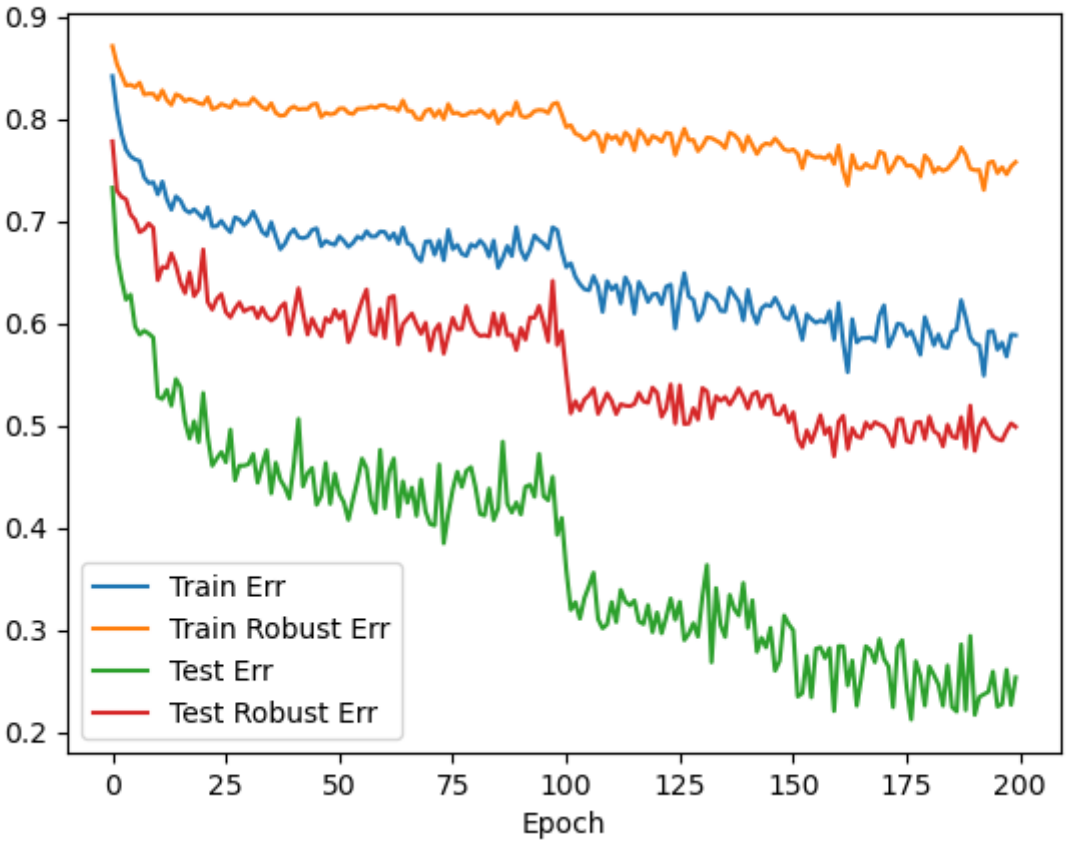
- Mixup alpha = 0.2



- Mixup alpha = 1.0



- Mixup alpha = 2.0



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

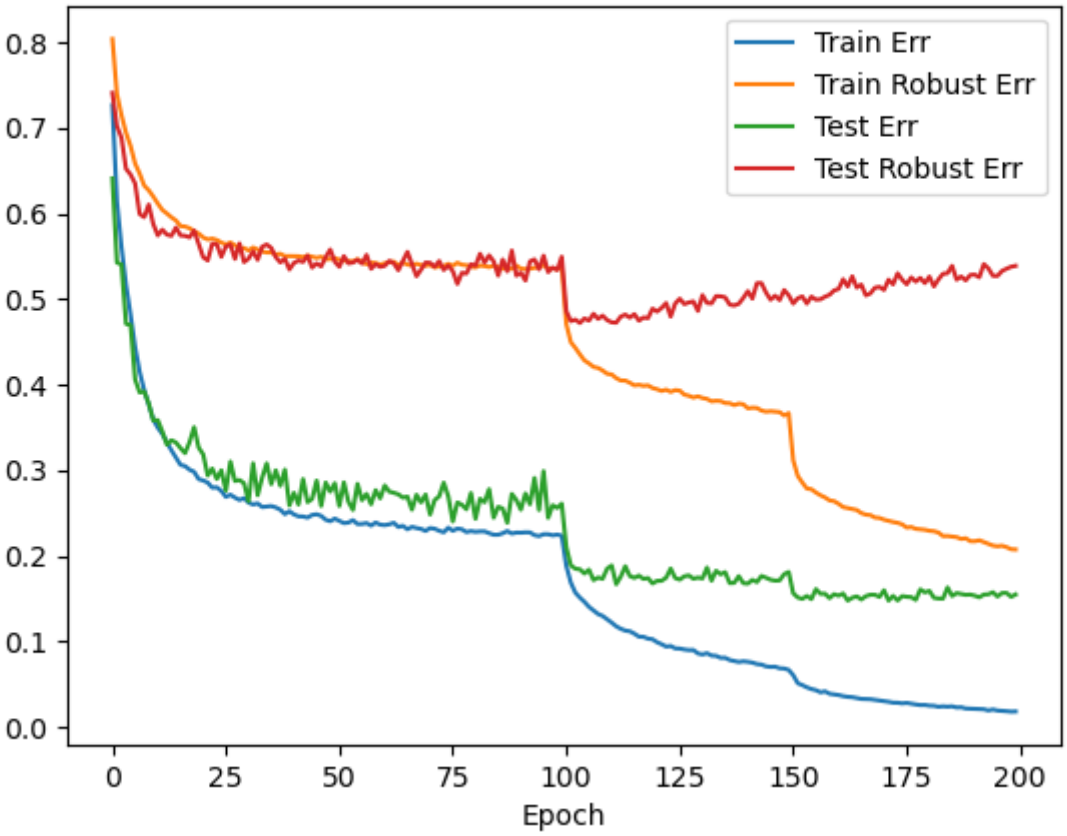
3.3 L1 & L2 Regularization

Explicit regularization 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.

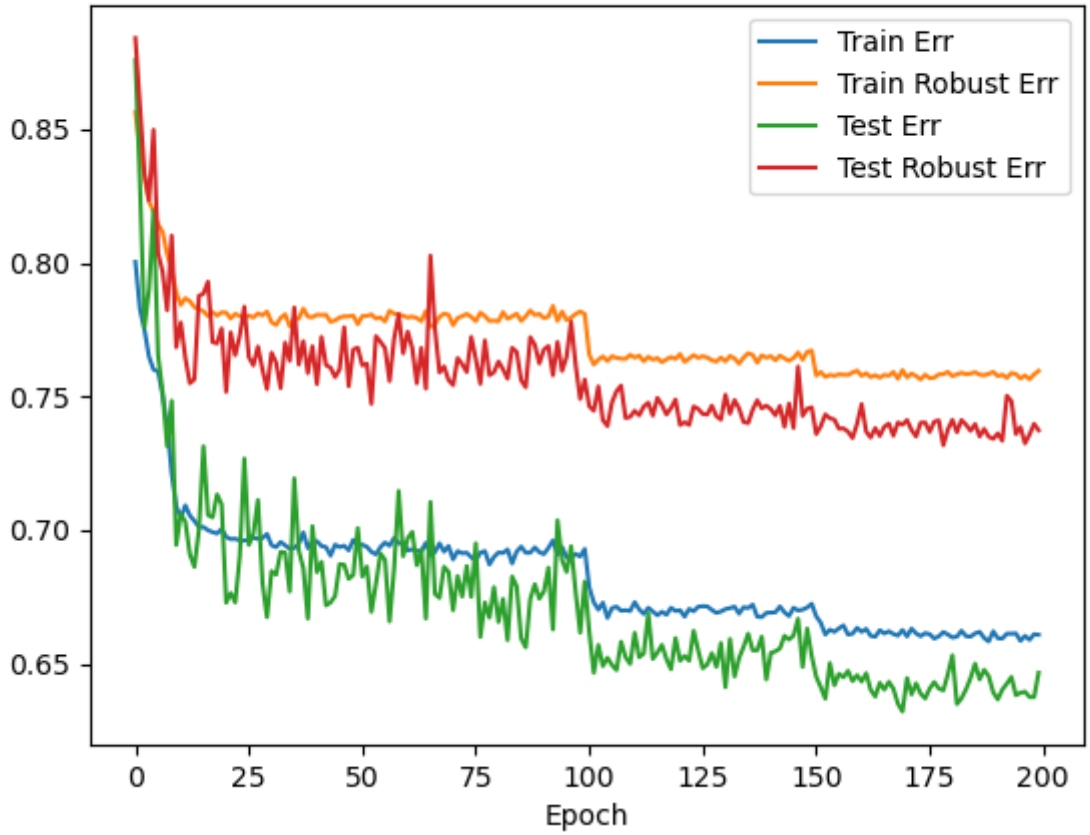
3.3.1 L1 Regularization

In this part we change the paramter value of l1.

- l1 = 0.0005



- l1 = 5e-07

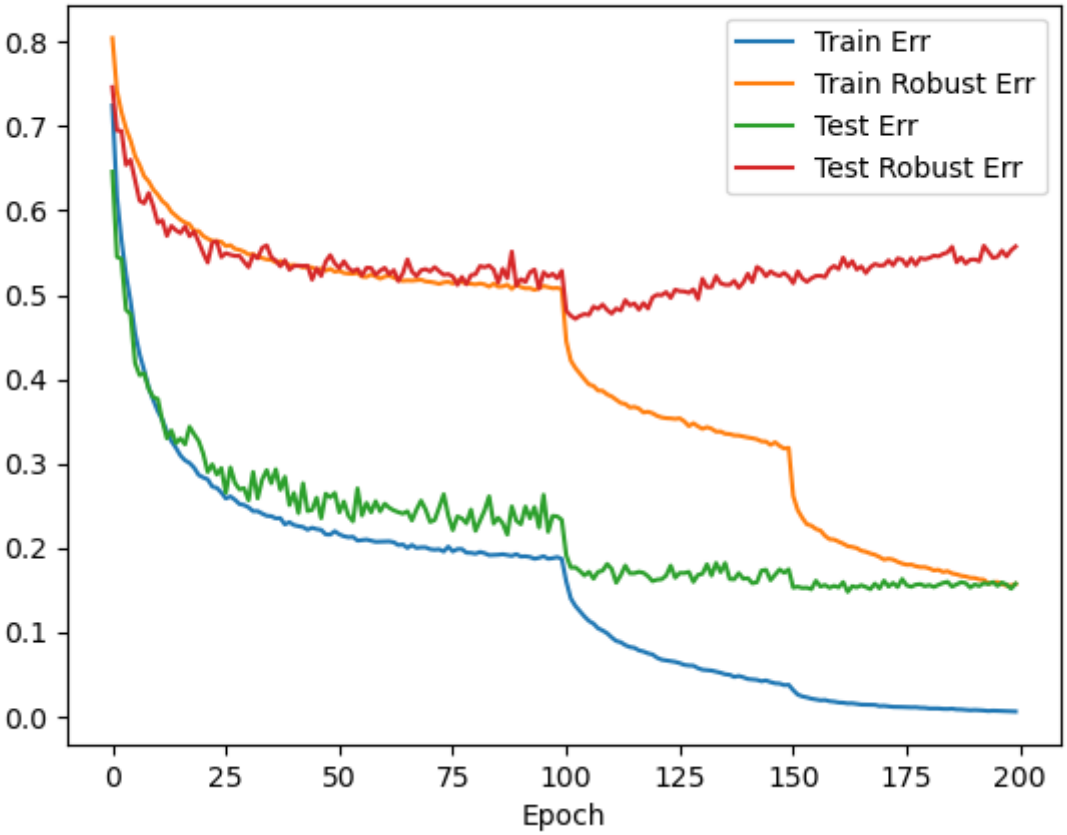



```
# an example command of changing the parameter value of l1
python3 train_cifar.py --fgsm-alpha 1 --fname 'experiments/cifar10_l1/preactresnet18_5e-4' --
python3 train_cifar.py --fgsm-alpha 1 --fname 'experiments/cifar10_l1/preactresnet18_5e-7' --
```

3.3.2 L2 Regularization

In this part we show the result of l2 regularization.

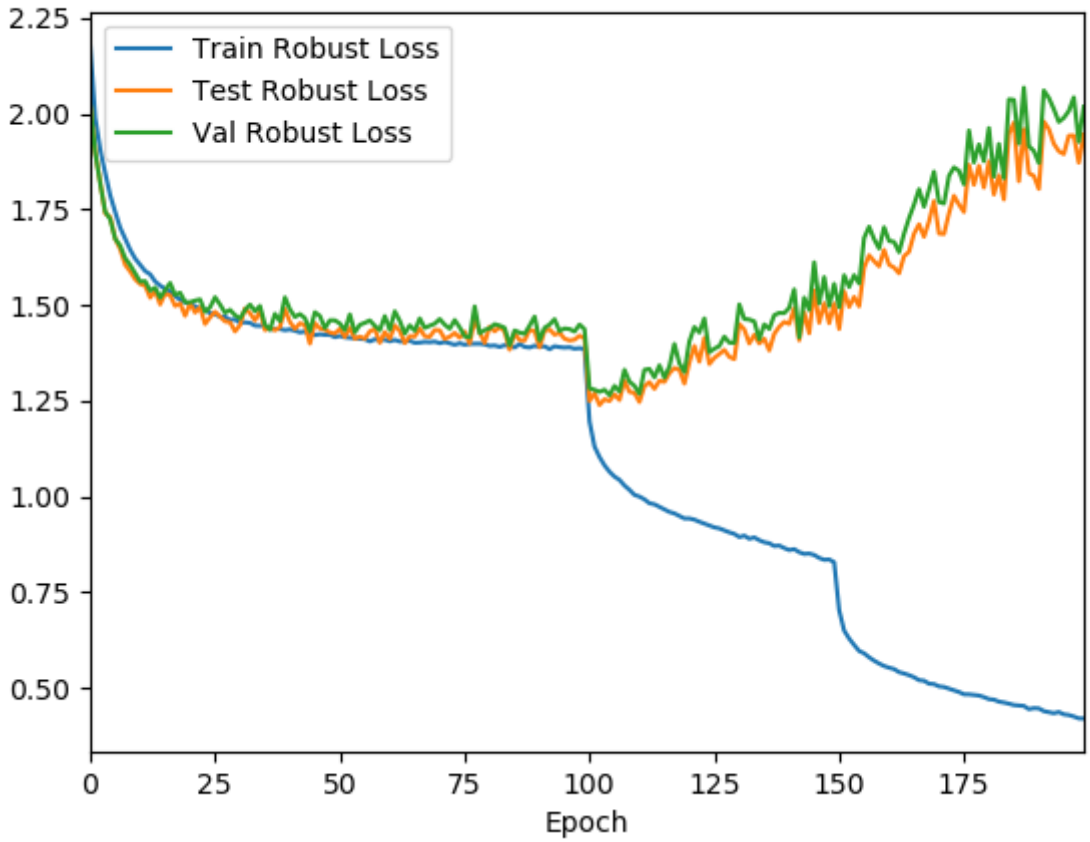
- l2 = 0.0005




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

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.

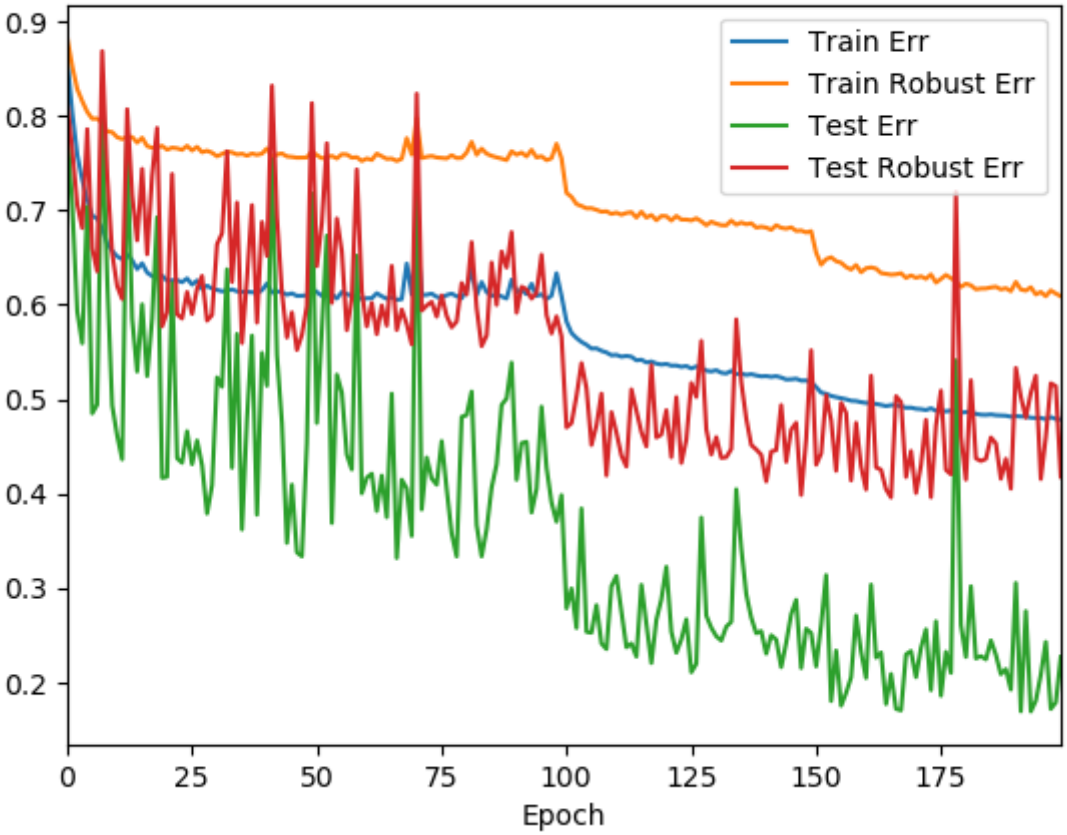



```
# an example command of setting a validation set in the standard training
python3 train_cifar.py --fgsm-alpha 1 --fname 'experiments/cifar10_validation/preactresnet18'
```

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.

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



4. Extensions

The extensions of paper 2 are beased on following ideas:

- add other methods of data augmentation (Mixup+Dropout)
- add dropout
- try more attack methods

In this part we can use some of the experience from the already existing empirical results to get some efficient ideas of our extensions. For example, as for the value of the hyperparameters, we can already do the "cherry-pick" to choose some of the values which have the best chance of potentially best result. Especially when we use new attack methods, some of the previous experience can be very helpful at that part. In this case, like we use same data augmentation methods on different attack methods, then theoritically we should expect similar behaviors under same style of tunning of all the values of the hyperparameters. And with some of the help of previous experience, in the extension part, we can save a lot of time trying different values setting. Compared with spending tons of time on pick the best or the "good" hyperparameter value, in this part we can focus more just on the robust error itself and use it to make the conclusion from the previous reproduction part strengthend.

And as for all the experiments' results of this part (extensions), they can be checked on our own

4.1 Cutout + Mixup

In this specific extension, we used a rather tricky way: Combing the Cutout and Mixup directly and see if some good result could be achieved. Since it's a combination of 2 different but still same style data augmentation methods.

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.

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 set the parameter value of the cutout as 10 and the mixup as 1.0 .

Explanation of the value: as stated at the beginning of this part, we can see from the value setting that we directly use some of the potentially "best" values of the hyperparameters in this case. To avoid spending too much resource.

- Cutout len = 10
- Mixup alpha = 1.0

4.2 Dropout

When training CNN networks, dropout is often used to make the model have better generalization and prevent overfitting. The essence of dropout is to make the data input to the network become 0 in some dimensions with a certain probability, which can make the model training more effective.

The Dropout method is a trick that is widely used when training models. The purpose is to prevent the model from overfitting. The principle is to make each parameter of a certain layer in the network masked (to 0) with a certain probability, and only the remaining The parameters are trained to prevent the model from overfitting.

Taking the implementation in Pytorch as an example, we commonly use the `torch.nn.Dropout(p=0.5, inplace=False)` method, and the underlying function it calls is `torch.nn.functional.dropout()` .

In this Dropout section, we have done several experiments using many different settings of the Dropout value. To be exact, we used the value: 0.2, 0.3, 0.5, 0.6, 0.8.

As for the performance of the dropout, the detailed resultcan be checked from the table at the conclusion part.

4.3 New attack method: FGSM + various methods were used in PGD

In the preivious reproduction, all the attack methods were PGD, now we switch to another attack method - FGSM, in order to check if all the methods that can be used to prevent robust overfitting can also used on different attack method like FGSM and can have similar behavior and phenomenon.

And also for this part, in order to achieve some kind of meaningful comparison between the results from the original reproductions with PGD, we design the whole experiments according to the original reproduction results. For example, for all the methods that were used in PGD like Cutout, Mixup, etc. We will basically use them in this part too with FGSM.

Note: for every experiment in this part, we set the hyperparameter value of the FGSM to fgsm-alpha = 0.875. As for the reason of picking this value: although this part is not really mentioned and used to get to the conclusion of the original paper, the author still provide some of the experiment settings in the original repo that can give us some insights, therefore after analysing the repo we decided to use the value of fgsm-alpha equal to 0.875.

Since the overall logic of the reproduction is very similar to the previous reproduction and also with the consideration of the idea from the original paper, it is not really necessary to provide any more images about the whole process. Since the conclusion should come from the table, which should illustrate the gap of the value between the best and the final result of the robust error. With this value, we could already come to a conclusion on which of the methods are useful and which of them are the best to use.

5. Conclusion

This conclusion part consists of two parts:

1. The conclusion can be concluded from the original reprodution of the original paper (which can be checked from the part of the empirical results production).
2. The conclusion from the extension that we produced and see if those two parts can have similar results, leading to the same theory and conclusion.

5.1 Reprodution of the original paper

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:

1. Found that early stopping, compared to other methods, is the most effective way to solve robust overfitting.
2. 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.

5.2 Extension

In this part we need to use the same idea of the reproduction to compare the results of the extensions. And as for the extension it can devided into 2 parts, one is the extension of the PGD methods, so we can directly extend the previous table as shown above, as for the second part, it is an extension of another attack method - FGSM, so we use a new table to present the results in order to achieve a better illustration performance.

5.2.1 Extension of the PGD

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
CUTOUT + MIXUP	50.80	46.89	3.91
Dropout 0.2	56.47	49.07	7.40
Dropout 0.3	55.52	50.05	5.47
Dropout 0.5	54.12	51.31	2.81
Dropout 0.6	54.76	52.86	1.90
Dropout 0.8	66.45	64.80	2.08

- This Table summarize our own reproduction results and the extension results based on the PGD. 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:

1. Found that early stopping, compared to other methods including the new extension methods, is the most effective way to solve robust overfitting.
2. 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.

5.2.2 Extension of the another attack method - FGSM

REG METHOD	**FGSM** Robust Test Error(%) FINAL	BEST	DIFF
FGSM 0.875 EARLY STOPPING W/VAL	**54.11**	54.11	0
FGSM 0.75	60.00	55.36	4.64
FGSM 0.875	58.79	53.56	5.23
FGSM 0.875+L1	59.17	53.61	5.56
FGSM 0.875+L2	62.27	54.96	7.31
FGSM 0.875+MIXUP	57.94	55.57	2.37
FGSM 0.875+CUTOUT	58.04	52.92	5.12
FGSM 0.875+CUTOUT+MIXUP	58.53	56.70	1.83

- This Table summarize our own extension results based on another attack method - FGSM. 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:

1. Found that early stopping, compared to other methods including the new extension methods, is the most effective way to solve robust overfitting.
2. 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.

5.3 Overall conclusion

From the results of both reproductions and extensions, we can see very similar behavior of the overall process and we got same result. Therefore, with all those experiments, we can varify the conclusion from the paper: Early stopping is the best method to prevent "**robust overfitting**" compared with other methods we used.

References

[1] Chen, Lin, et al. "More data can expand the generalization gap between adversarially robust and standard models." International Conference on Machine Learning. PMLR, 2020.

[2] Rice, Leslie, Eric Wong, and Zico Kolter. "Overfitting in adversarially robust deep learning." International Conference on Machine Learning. PMLR, 2020.