Group 3: Adversiral training 1

Hao Liu, Ziyuan Qin, Haozhuang Chi

August 1, 2022

This report serves mainly to show the ideas of our extensions.

1 Paper 1 [2]

After we have re-implemented experiments in the 1-dimensional setting and with perturbation of only L_∞ norm as described in the original paper for the three models: Gaussian Mixture with Linear Loss, SVM, and Linear Regression, 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.

$$w_n^{\text{rob}} = \underset{\|w\|_\infty \le W}{\arg\min} \sum_{i=1}^{n} \max_{\tilde{x}_i \in B_{x_i}^{\infty}(\varepsilon)} \left(-y_i \langle w, \tilde{x}_i \rangle\right) = \underset{\|w\|_\infty \le W}{\arg\max} \sum_{i=1}^{n} \min_{\tilde{x}_i \in B_{x_i}^{\infty}(\varepsilon)} y_i \langle w, \tilde{x}_i \rangle$$

where Θ is the parameter space and $B_{x_i}^{\infty}(\epsilon) \triangleq \widetilde{x_i} \in \mathbb{R}^d \| || \widetilde{x_i} - x||_\infty \le \epsilon$ 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.

1.1 Different Attack Methods

i) Optimal robust classifier:

An analytical solution for the models Gaussian Mixture with Linear Loss and Linear Regression can be derived from the inner optimization problem. The specifics are shown in the notebook.

ii) The Fast Gradient Sign Method (FGSM):

According to [1], if we want to minimize some function $f : \mathbb{R}^n \to \mathbb{R}$ over the input z, the traditional gradient descent algorithm repeats the update:

$$z := z - \alpha \nabla_z f(z)$$

But the normalized steepest descent method chooses v to maximize the inner product between v and the gradient subject to a norm constraint on v:

$$\text{argmax}\, z := z - \underset{\|v\| \le \alpha}{\text{argmax}}\, v^T \nabla_z f(z)$$

Under an L_∞ norm constraint on v,

$$\underset{\|v\|_\infty \leq \alpha}{\operatorname{argmax}} v^T \nabla_z f(z) = \alpha \cdot \operatorname{sign}(\nabla_z f(z))$$

iii) Fast Gradient Method (FGM):

Under an L_2 norm constraint on v,

$$\underset{\|v\|_2 \leq \alpha}{\operatorname{argmax}} v^T \nabla_z f(z) = \alpha \frac{\nabla_z f(z)}{\|\nabla_z f(z)\|_2}$$

iv) Projected gradient descent (PGD):
FGSM and FGM directly calculates the adversarial disturbance through a single step, which may not be optimal. Therefore, PGD has been improved, iterating several times, and slowly finding the optimal disturbance. PGD walks in small steps and takes a few more steps. If the disturbance exceeds a given radius, it will be mapped back to the "sphere" to ensure that the disturbance is not too large.

The basic PGD algorithm simply iterates the updates:

$$\text{Repeat} \quad \delta := \mathcal{P}(\delta + \alpha \nabla_\delta \ell(h_\theta)$$

1.2 Data with Higher Dimensions

In 1-dimensional data set, we have observed the three adversary regimes: weak, medium, and strong. We want to know if the three regimes can be observed for higher-dimensional data set. Also, we want to investigate whether the same attack method would yield the same regime of test losses for same set of epsilons for higher dimensions.

1.3 SVM with RBF Kernel

As 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:

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

1.4 Gaussian Process

As 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.

2 Paper 2 [3]

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

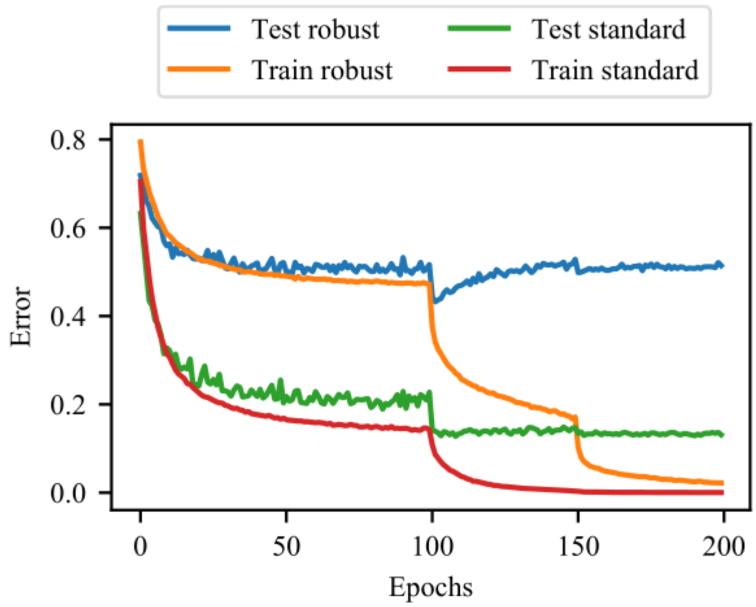

Figure 1: Robust Overfitting

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”, as shown in the figure 1.

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.

[3] tried many common methods to prevent overfitting, including l1 and l2 regularization, data augmentation (cutout, mixup, semi-supervised learning), although these methods can alleviate robust overfitting to varying degrees, but in addition to semi-supervised learning with additional data, other methods are inferior to simple early stopping.

2.1 Empirical Results of original reproduction

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 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,

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 2: original table

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.

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.

i) 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.

ii) 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.

iii) 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.

iv) 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.

v) Semi-supervised training:
Semi-supervised learning methods augment the dataset wit unlabeled data, and have been shown to improve generalization when used in the adversarially robust setting. Note that test error and test robust error oscillate a lot, but it can be seen that robust overfitting is not stark.

	Robust Test Error(%)		
REG METHOD	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

Figure 3: reproduction table

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.

2.2 Empirical Results of extensions

The extensions of paper 2 are beased on following ideas:

i) add other methods of data augmentation (Mixup+Dropout)

ii) add dropout

iii) try different 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 repo.

2.2.1 Extension based on original PGD

i) 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 len = 10, Mixup alpha = 1.0

ii) Dropout:
Distributed representation is a core idea of artificial neural network research. Simply put, when we express a concept, the neurons and concepts are not stored in a one-to-one mapping (map), but the relationship between them is many-to-many. Specifically, a concept can be defined and expressed by multiple neurons, and a neuron can also participate in the expression of multiple different concepts, but the weights are different. Using the distributed feature expression of the neural network (as long as the core features can be retained), it can not only achieve the successful completion of the task (for example, successfully identify the picture as a cat), but also can be used to prevent the occurrence of overfitting. The distributed feature expression can be called the origin of Dropout .

A technique widely used in deep learning: Dropout Learning. The core idea is similar to the distributed feature expression explained above, and the key is to retain the core features.

"Dropout" means that during the training process of the deep learning network, for the neural network unit, it is temporarily dropped from the network according to a certain probability. It is usually divided into two phases: the learning phase and the testing phase.

For each dropout network, when training, it is equivalent to doing Data Augmentation.

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

Figure 4: extension table 1

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.

2.2.2 Extension based on 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

Figure 5: extension table 2

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.

References

[1] Z. Kolter. Chapter 3 - adversarial examples, solving the inner maximization, 2022. URL https://adversarial-ml-tutorial.org/adversarial_examples/.

[2] Y. Min, L. Chen, and A. Karbasi. The curious case of adversarially robust models: More data can help, double descend, or hurt generalization. In *Uncertainty in Artificial Intelligence*, pages 129–139. PMLR, 2021.

[3] L. Rice, E. Wong, and Z. Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020.