[SPML] Homework 2: Black-box Defense

Jun-Da, Chen
R08946014 Data Science

1 Methods

In this homework, I chose five pretrained models that chosen as target models in HW1: *nin*, *sepreresnet56*, *xdensenet40_2_k24_bc*, *ror3_110* and *resnet110*. Especially, I repalce resnet1001 with resnet110 for speeding up the training time. For each model, I used PGD to generate adversarial examples for each data from training set, and used them to do adversarial training. In order to understand the effect of adversarial examples in adversarial training, I tried different number of these examples. Furthermore, besides normalizing, I also do other preprocessing used in HW1: ColorJitter, CenterCrop and Padding, trying to see whether these methods can enhance robustness or not.

2 Chosen Methods in the Submission

I chose the pretrained model *resnet110*, selecting 25% of training set to generate adversarial examples, and used these examples to do adversarial training. And the inputs fed to this model did not do furher preprocessing like ColorJitter, CenterCrop or Padding, only normalizing. With the choice mentioned above, the adversarial trained model could achieve the accuracy 72% on the benign 10,000 test set and 76% on the adversarial perturbed 10,000 test set, which is the best average performance in my experimental results.

3 Experiments

3.1 Data Set and Setting

The data set is CIFAR10. The number of the training set is 50,000 and the number of the test set is 10,000. Pretrained models are from *pytorchcv* [1]. The method used to generate adversarial examples is PGD [2], and the maximum perturbation is 8/255 after normalizing, just as the same as TA's attack. The step size of PGD is 2/255 and the number of iterative steps is 40. The batch size is 16 and the model are all at most trained 3 epochs.

3.2 Quantitative Analysis

The best result of adversarial trained model comparing with naive pretrained model is shown in Table 1. We could see that without adversarial training, all the accuracy of the model decrease dramatically when the inputs are perturbed. After adversarial training, all model could defend such attack except *nin*. Especially the model *resnet110*, not only with the competitive adversarial accuracy 76%, but also with the high benign accuracy 72% comparing to other models. In the following experiments, we would further analyze the variation of accuracy on different training epoch, percentage of adversarial examples and more preprocessing or not.

Table 2 demonstrates the variation of the the accuracy on different training epoch. The models in the table are all adversarial training with 25% of the adversarial examples. Basically, as the training epoch increases, the benign accuracy decreases and the adversarial accuracy increases, which shows that more epochs the model more overfits on adversarial examples.

Table 1: Adversarial Robustness across Models

	Train on benign		Train on benign + adversarial	
Model	**Benign Accuracy**	**Adversarial Accuracy**	**Benign Accuracy**	**Adversarial Accuracy**
nin	88%	0.02%	10%	10%
sepreresnet56	93%	0.10%	45%	77%
xdensenet40_2_k24_bc	93%	0.00%	59%	80%
ror3_110	93%	0.25%	59%	74%
resnet110	94%	0.40%	**72%**	**76%**

Table 2: Model Robustness across Epochs condition on 25% Adversarial Training Examples

	Epoch 0	**Epoch 1**	**Epoch 2**
Model	**benign / adv.**	**benign / adv.**	**benign / adv.**
nin	10% / 10%	10% / 10%	10% / 10%
sepreresnet56	45% / 77%	43% / 83%	30% / 84%
xdensenet40_2_k24_bc	59% / 80%	43% / 88%	21% / 93%
ror3_110	59% / 74%	39% / 82%	43% / 87%
resnet110	72% / 76%	50% / 81%	44% / 83%

Table 3 shows the variation of the the accuracy on different percentage of the adversarial examples to do adversarial training. The models in the table are all chosen from the first training epoch, i.e., Epoch 0. Like the tendency in Table 2, as the percentage of the adversarial training examples increases, the benign accuracy decreases and the adversarial accuracy increases, which also indicates that the model more overfits with more adversarial training examples

Table 4 further focus on the smaller change around 25% adversarial training examples. The models are all chosen from the epoch 0. Given the model *resnet110*, the benign accuracy with 20% adversarial training examples is 71%, which is lower than 72%, slightly violates the tendency we observe in Table 3. Considering 30% adversarial training examples, although the adversarial accuracy increases 1%, the benign accuracy decreases dramatically. After the analysis, we could assume the model tranined on 25% adversarial training examples and just one epoch achieve the best performance.

Inspired by HW1, I also do more transform besides normalize in the preprocessing phase. These transforms are ColorJitter, CenterCrop and Padding. Where ColorJitter I used naive setting, CenterCrop with size 28 pixels and hence Pad 2 pixels on each border. The fill value in Padding is default, i.e., 0 on three channels, so the padding color is black. In Table 5, we could discover that after more preprocessing, merely the both accuracy of model *nin* increase, other models mostly tends to decrease condition on models are all trained on 25% of adversarial examples. Table 6 also proves that more percentage of adversarial examples the models tends to overfit. And the tendency is more significant when applying more preprocessing comparing to Table 3.

Table 3: Model Robustness across Percentage of Adversarial Training Examples condition on Epoch 0

	0.25	**0.5**	**1.0**
Model	**benign / adv.**	**benign / adv.**	**benign / adv.**
nin	10% / 10%	10% / 10%	10% / 10%
sepreresnet56	45% / 77%	35% / 86%	23% / 90%
xdensenet40_2_k24_bc	59% / 80%	31% / 83%	20% / 95%
ror3_110	59% / 74%	41% / 77%	36% / 83%
resnet110	72% / 76%	44% / 83%	24% / 88%

Table 4: Model Robustness across Percentage of Adversarial Training Examples condition on Epoch 0

	0.2	**0.25**	**0.3**
Model	**benign / adv.**	**benign / adv.**	**benign / adv.**
nin	10% / 10%	10% / 10%	34% / 33%
sepreresnet56	50% / 76%	45% / 77%	47% / 84%
xdensenet40_2_k24_bc	67% / 79%	59% / 80%	58% / 79%
ror3_110	50% / 76%	59% / 74%	47% / 74%
resnet110	71% / 73%	72% / 76%	39% / 77%

Table 5: Model Robustness across Preprocessing condition on 25% Adversarial Training Examples and Epoch 0

	More Preprocessing	**Only Normalize**
Model	**benign / adv.**	**benign / adv.**
nin	34% / 38%	10% / 10%
sepreresnet56	43% / 78%	45% / 77%
xdensenet40_2_k24_bc	54% / 74%	59% / 80%
ror3_110	39% / 75%	59% / 74%
resnet110	55% / 72%	**72% / 76%**

Table 6: Model Robustness across Percentage of Adversarial Examples condition on More Preprocessing and Epoch 0

	0.25	**1.0**
Model	**benign / adv.**	**benign / adv.**
nin	34% / 38%	10% / 10%
sepreresnet56	43% / 78%	21% / 90%
xdensenet40_2_k24_bc	54% / 74%	14% / 95%
ror3_110	39% / 75%	30% / 83%
resnet110	55% / 72%	16% / 88%

4 Findings or Insights

According the experimental results, we can conclude that:

1. As the number of the training epoch increases, the model tends to overfit the adversarial examples, which decreases the benign accuracy.
2. As the number of the adversarial examples used in adversarial training increases, the tendency is as the same as mentioned above, but more significant.
3. More transformation seems not to enhance both benign accuracy and adversarial accuracy for most models used in this homework.

For future work:

1. The adversarial examples generated by PGD are all used to do adversarial training whether the adversarial example could be classified correctly by navie pretrained model or not. Although these correctly classified examples stand a little portion, distinguishing them may improve.

2. I only consider random seed 0 in this homework, trying other random seed may find something new.

3. A new technique called smoothed adversarial training (SAT) [3] could be tried to implement. Because the widely-used ReLU activation function significantly weakens adversarial training due to its non-smooth nature. SAT replace ReLU with its smooth approximations (e.g., SILU, softplus, SmoothReLU) to strengthen adversarial training.

References

[1] https://github.com/osmr/imgclsmob/blob/master/pytorch/README.md

[2] https://github.com/Harry24k/adversarial-attacks-pytorch

[3] https://github.com/cihangxie/SmoothAdversarialTraining

[4] https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html