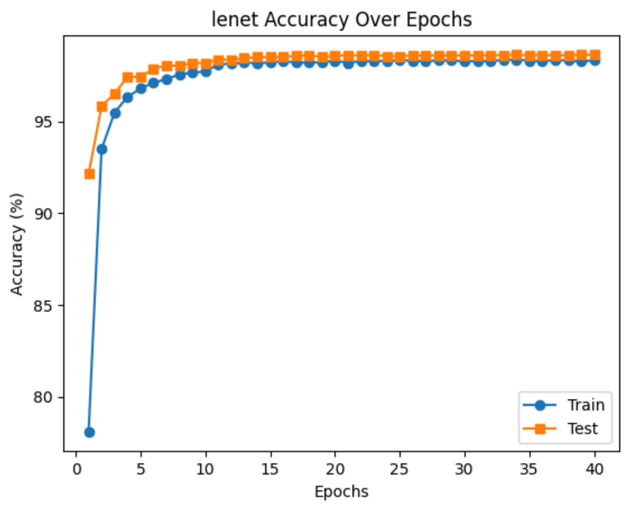
Jake Marlow
CS489
Prof. Suya
Apr 1, 2025

HW 2

Task I: Attack Models using PGD (ResNet18 on CIFAR-10 and LeNet on MNIST)

LeNet:

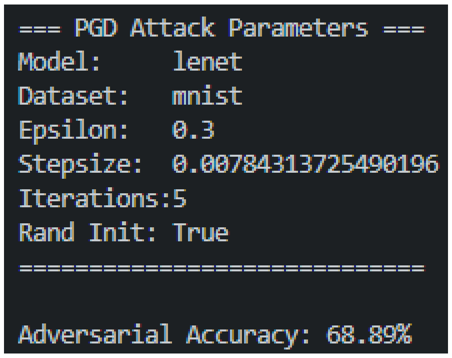


Here is the base model that I started with, which had a test accuracy of about 99%

After using PGD to attack it with these hyperparameters

- niter: 5
- epsilon: 0.3
- stepsize: 2/255.
- randinit: true

I was able to produce this drop in accuracy

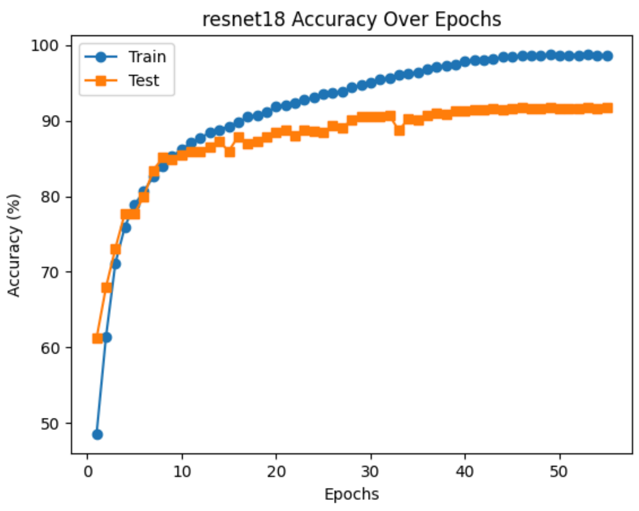


```
=== PGD Attack Parameters ===
Model:     lenet
Dataset:   mnist
Epsilon:   0.3
Stepsize:  0.00784313725490196
Iterations:5
Rand Init: True
=============================

Adversarial Accuracy: 68.89%
```

31% drop in accuracy

ResNet:

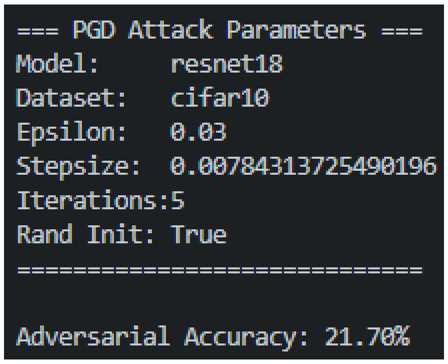


Here is the base model that I started with, which had a test accuracy of about 91.6%

After using PGD to attack it with these hyperparameters

- niter: 5
- epsilon: 0.03
- stepsize: 2/255.
- randinit: true

I was able to produce this drop in accuracy



```
=== PGD Attack Parameters ===
Model:     resnet18
Dataset:   cifar10
Epsilon:   0.03
Stepsize:  0.00784313725490196
Iterations:5
Rand Init: True
=============================

Adversarial Accuracy: 21.70%
```

71% drop in accuracy

Task II:

Subtask II-1: Analyze the Impact of Several Factors on Your Attack Success Rate

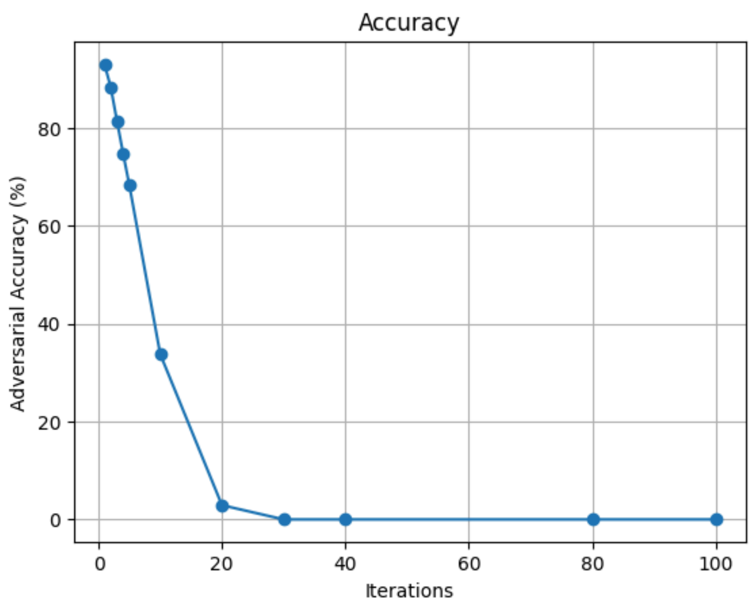
In this task I wrote a script to iteratively test PGD attack hyperparameters on the two models above to examine their effects

Hyperparameters tested:

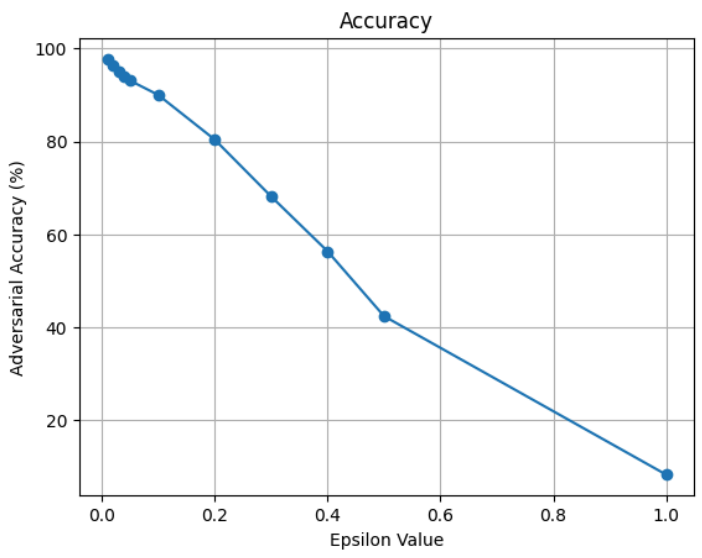
Num iterations = {1, 2, 3, 4, 5, 10, 20, 30, 40, 80, 100}

Epsilon = {0.01 0.02 0.03 0.04 0.05 0.1 0.2 0.3 0.4 0.5 1.0}

LeNet:

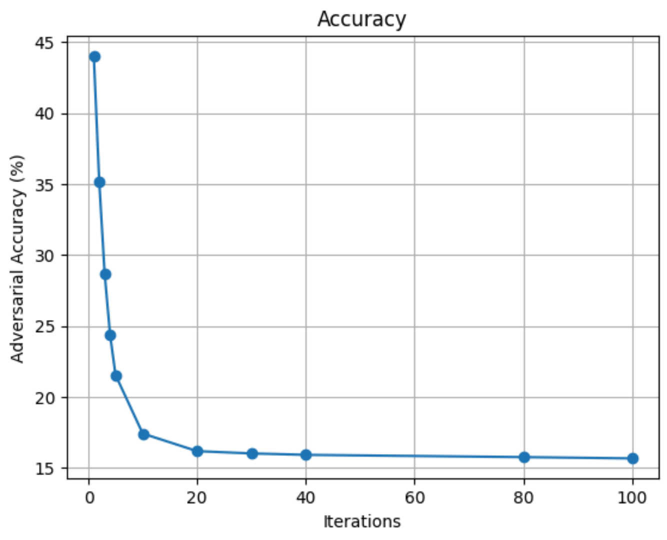


Above is a graph showing the accuracy over the number of iterations tested. The rest of the hyperparameters were the default from the last task. As expected, an increase in iterations results in a decrease in accuracy. After about 30 iterations the accuracy drops all the way to zero.

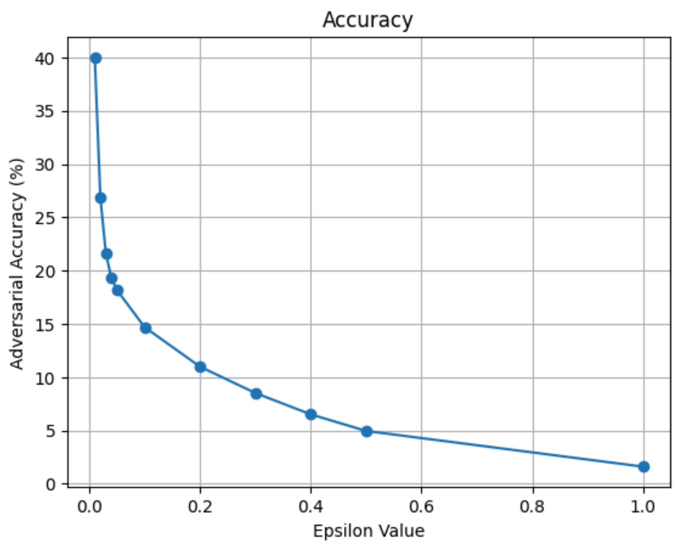


Above is a graph displaying the accuracy over the values of epsilon listed in the task description. As epsilon increases, the maximum allowed perturbations of the pixel values increase, which causes more and more error as they are maximizing the loss. This relationship is to be expected.

ResNet:



Above is a graph showing the accuracy of ResNet over the number of iterations tested. As expected, an increase in iterations results in a decrease in accuracy. After about 10 iterations the accuracy drops all the way to sub 20%, which means that it is really effective.



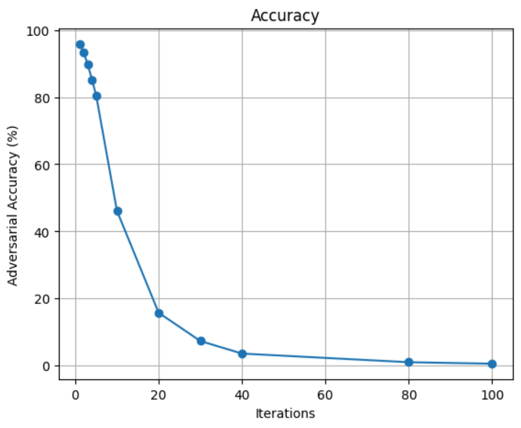
Above is a graph displaying the accuracy over the values of epsilon listed in the task description. Similar to LeNet, the accuracy drops drastically as the allowed pixel space increases.

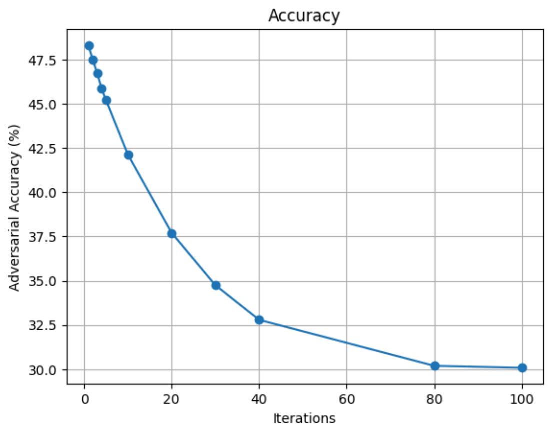
Subtask II-2: Analyze the Impact of the Training Techniques You Use

(1) Augmentation

LeNet

No Augmentation **Augmentation**

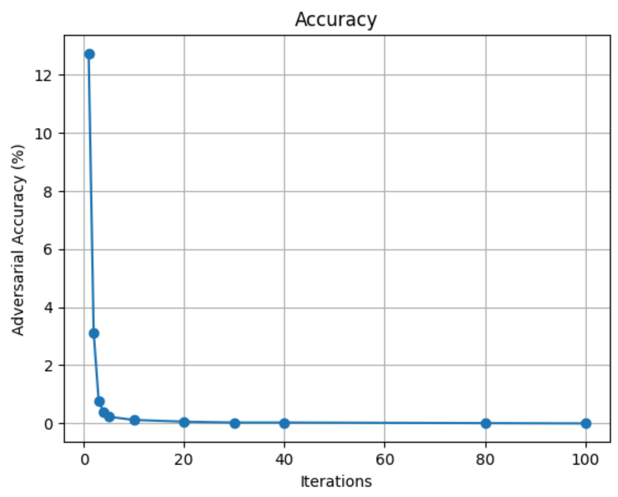




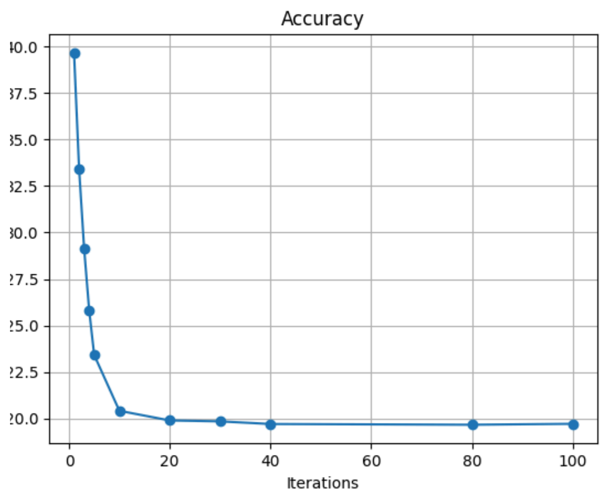
Above is a comparison of how LeNet performed under the same PGD attack with the default parameters over the various iterations from the last task. As can be seen from the graphs, the LeNet model trained with data Augmentation including random horizontal flipping and rotations of up to 70 degrees was slightly more robust in the long run, while the model without augmentation performed well within the first 5 iterations. At iteration 10, the “non-augmented” LeNet begins to dip below and stay below the accuracies of the “augmented” LeNet. This could be empirical evidence that augmentation can improve a model's robustness.

ResNet

No augmentation



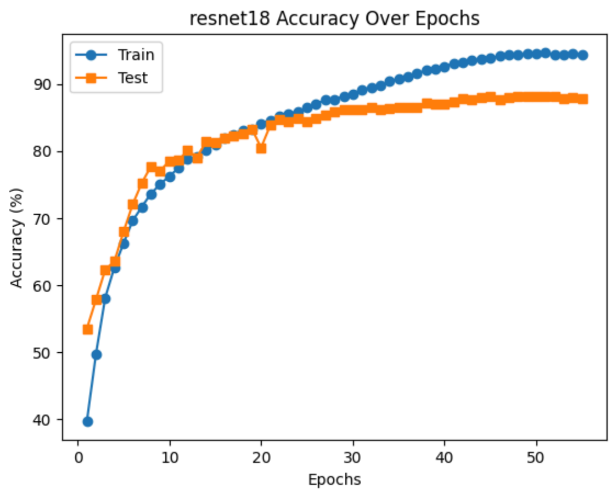
Augmentation



Above is a comparison of how ResNet performed under the same PGD attack with the default parameters over the various iterations from the last task. As can be seen from the graphs, the ResNet model trained with data Augmentation including random horizontal flipping and rotations of up to 70 degrees was noticeably more robust in the long run. At iteration 20, the “non-augmented” ResNet already hovered above 0% accuracy, while the “Augmented ResNet never got below 10% accuracy and held out a little bit longer. Of course the attacks themselves were still very successful on both models. This adds to the evidence that data augmentation can make a model slightly more robust to adversarial examples.

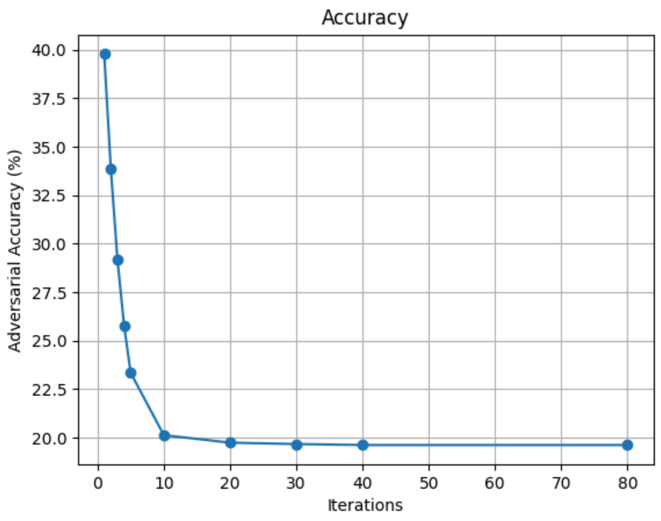
(2) Regularizations

1) Dropout = 0.5



Starting accuracy = 87.63%

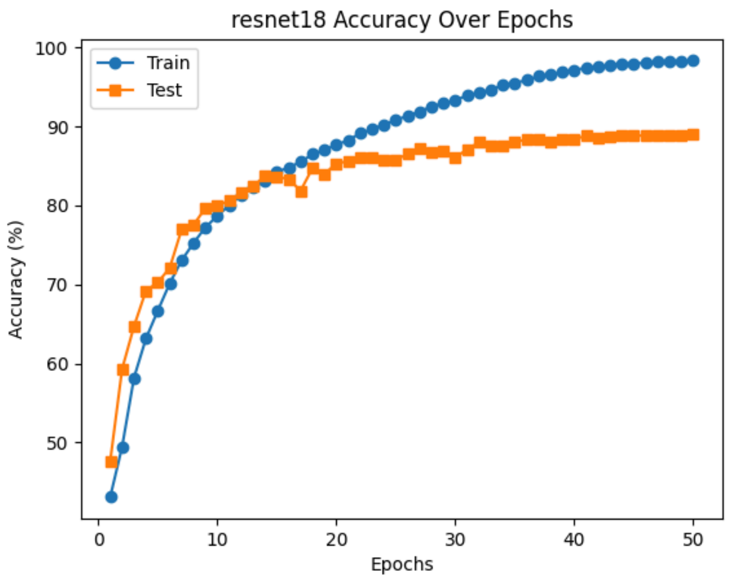
Accuracy during PGD Attack



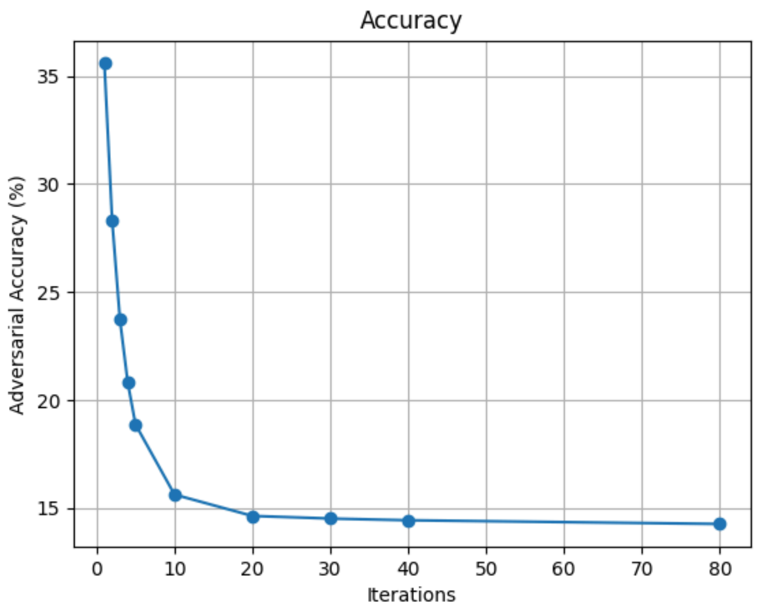
Results: Compared to the next using weight decay below, dropout seemed to actually cause some resilience in the model. Although the accuracy is not great, this model experienced a 5% greater accuracy score from beginning to end compared to the best weight decay models. I think that the nature of using dropout causes the model to behave more like an "ensemble", which slightly increases the accuracy in the face of adversarial attacks.

2) Weight Decay

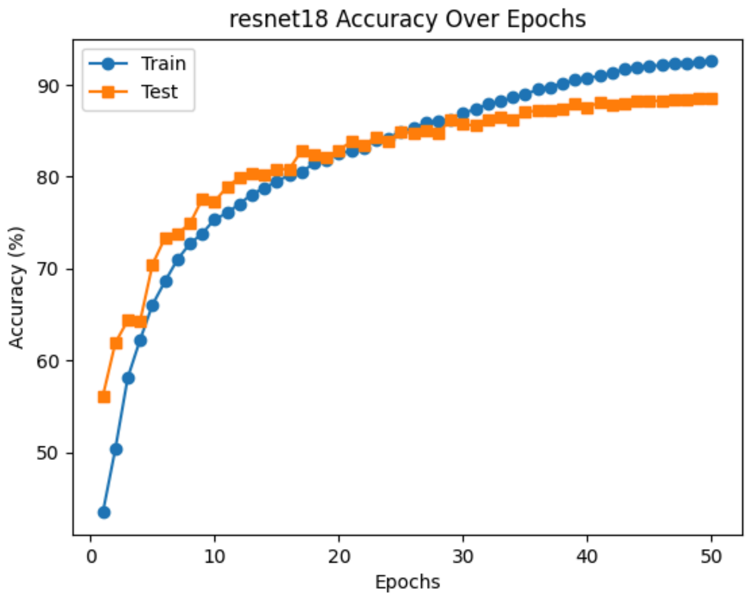
Weight decay = 1e-5, test accuracy = 89%



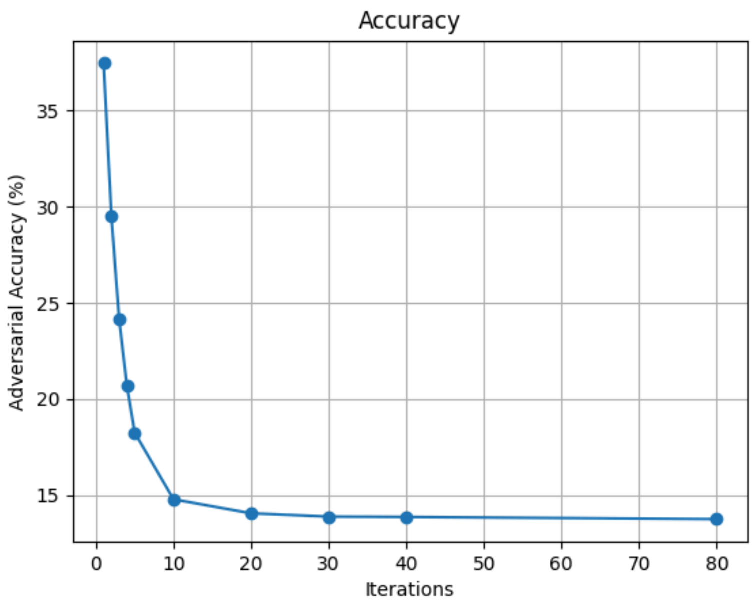
Accuracy during PGD Attack



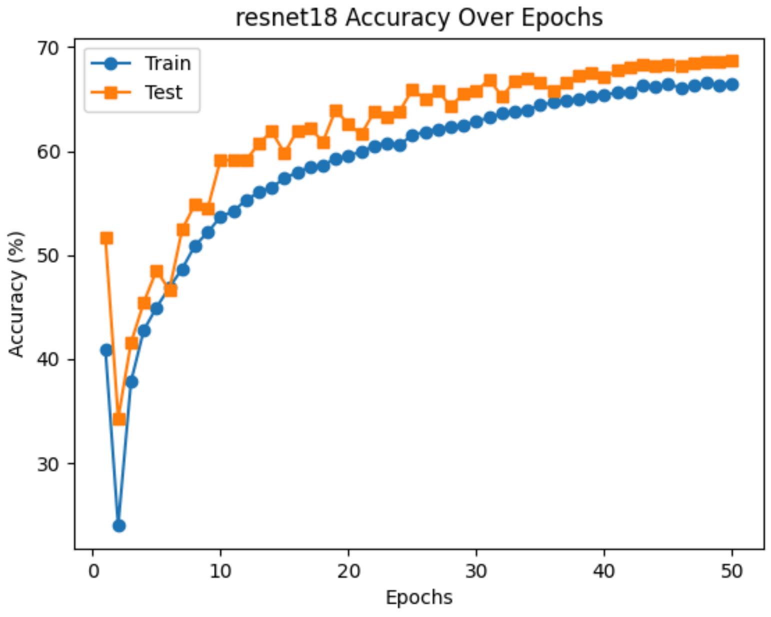
Weight decay = 1e-4, test accuracy = 88%



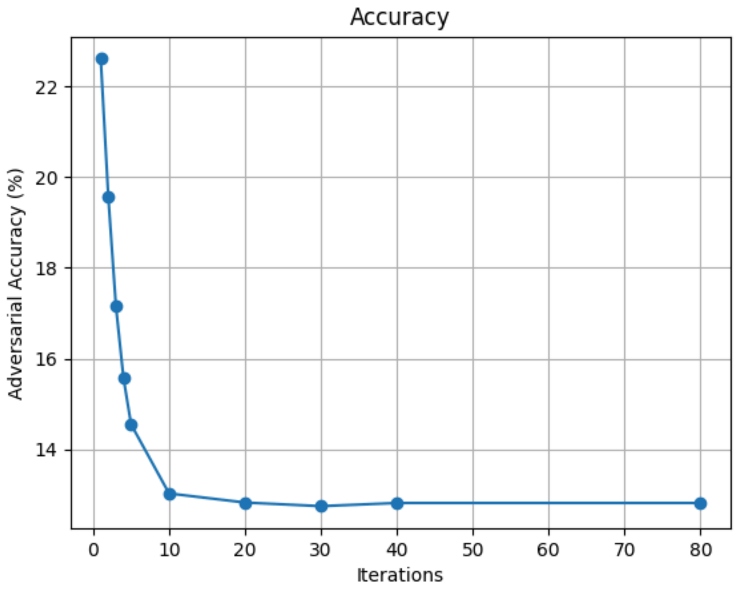
Accuracy during PGD Attack



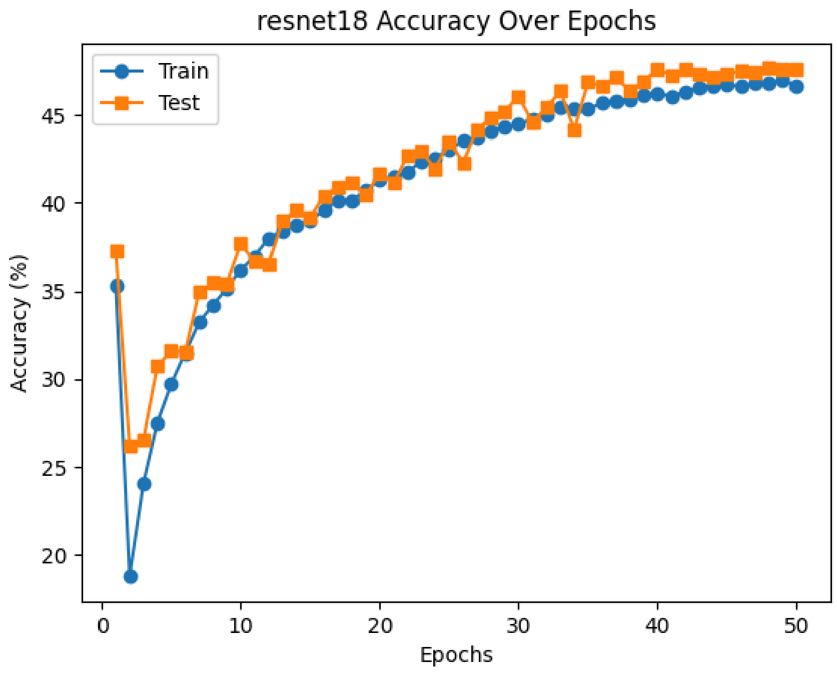
Weight decay = 1e-3, test accuracy = 68%



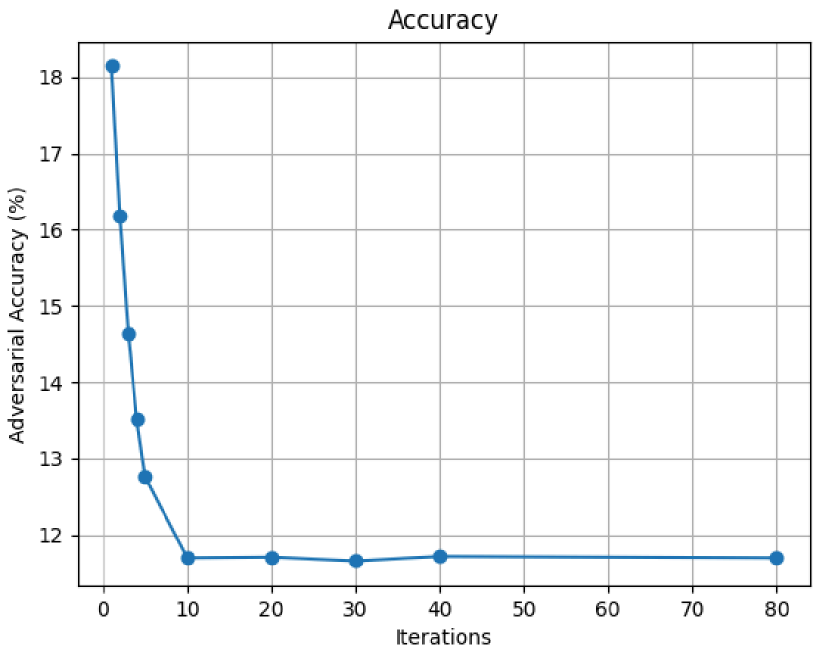
Attack accuracy



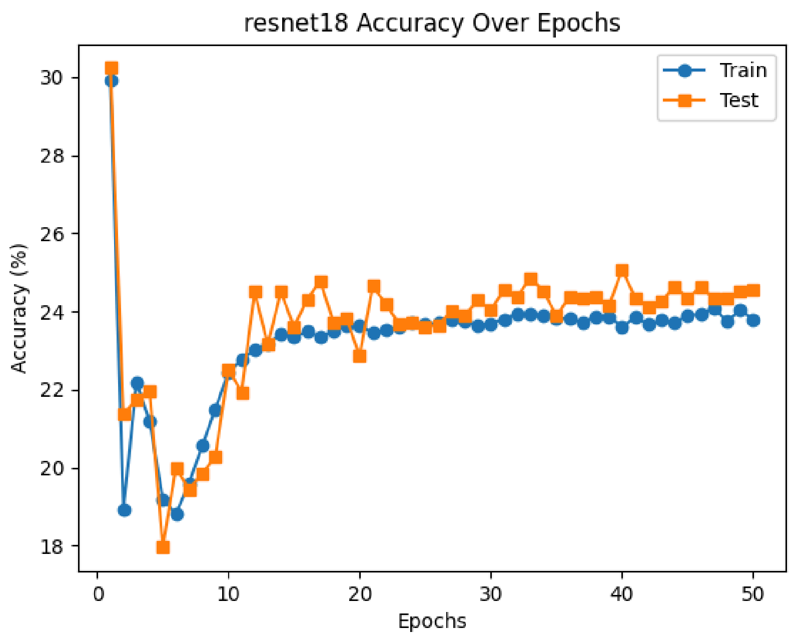
Weight decay = 1-e2, test accuracy = 47%



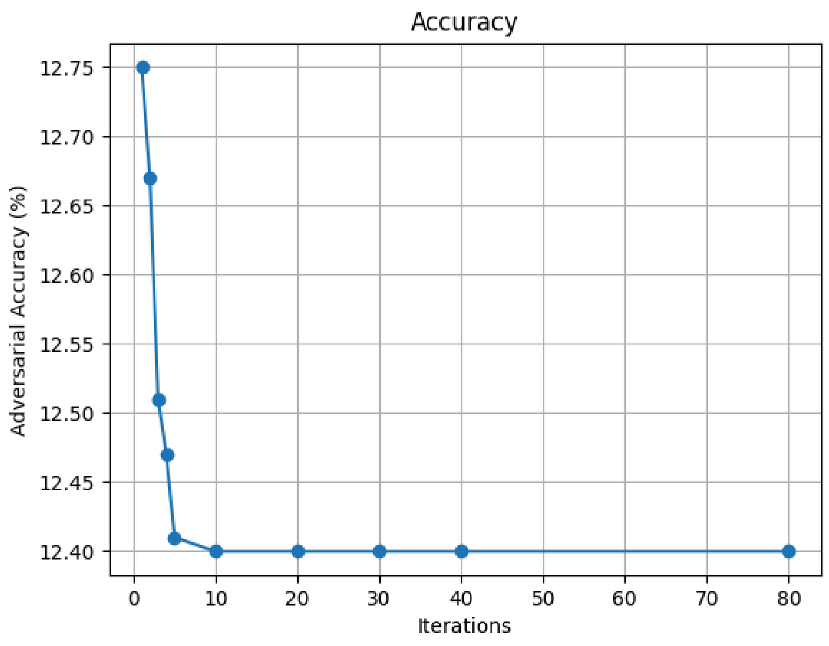
Attack accuracy



Weight decay= 1-e1, test accuracy = 24%



Accuracy during PGD Attack

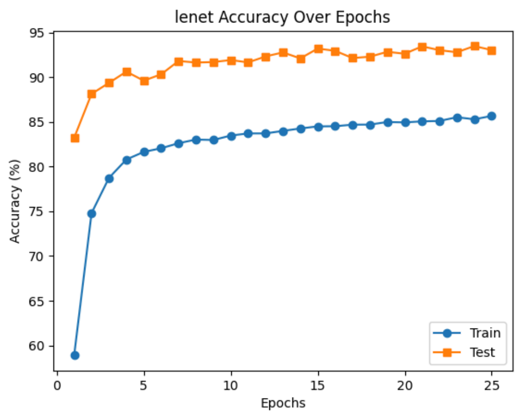


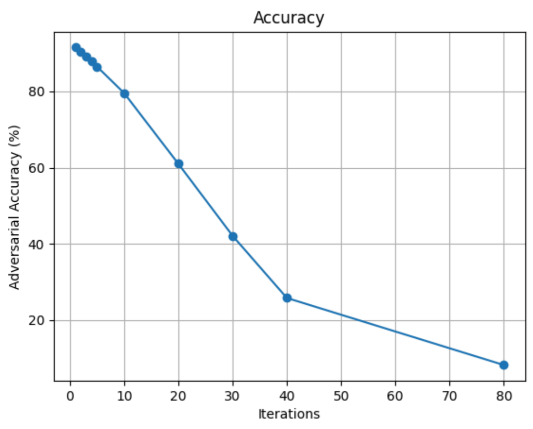
Results: After examining the graphs here, it can be seen that adding weak or strong weight decay is not a useful means of defending against adversarial examples. The only slight improvement was the second model with weight decay = 1-e4 with a similar accuracy to the first model. It experienced a slightly higher accuracy up front at 37.47% compared to the first model at 35.59%, but both (and all) models experienced significant decreases in accuracy.

These results make sense because adding L2 regularization simply helps the model generalize better by penalizing larger weights. They don’t have much to do with more defensive decision boundaries, but might be benefiting from less ability to amplify small perturbations.

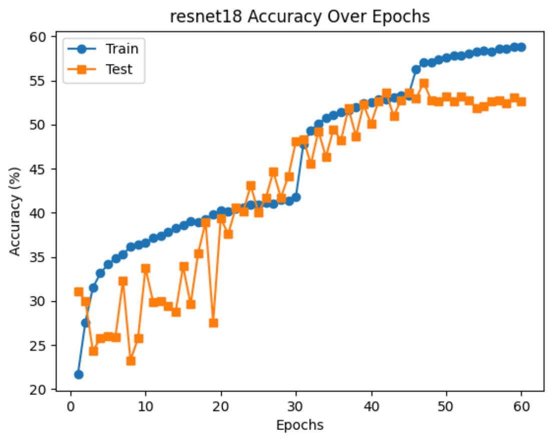
Adversarial Training

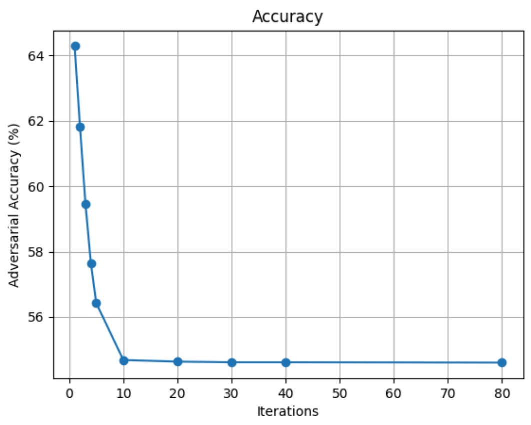
Lenet





ResNet-18





Results of Adversarial Training;

(1) How's your robust models' accuracy on adversarial examples compared to your undefended models?

The robust models for both LeNet and Resnet-18 performed much better on adversarial examples than their counterparts. LeNet, when adversarially trained, had a much more linear curve than the exponential decrease of its undefended model. It wasn't until about 30 iterations that the accuracy was halved, while the undefended model reached almost 0% accuracy after about 20 iterations. ResNet, when adversarially trained, had a steep drop off in accuracy early on but never broke below 54% accuracy, which is much better than the undefended model with the best case at 15% accuracy.

(2) How's your robust models' accuracy on clean test-set examples compared to your undefended models?

LeNet_AdvTrain = 93%	ResNet_AdvTrain = 52.64%
LeNet_Undefended = 98%	ResNet_Undefended = 91.26%

Accuracy for LeNet on MNIST, when adversarially trained, was much better compared to ResNet's accuracy when adversarially trained. (It took MUCH longer and was more difficult to fine tune)

(3) Let's increase the PGD attack iterations from 5 to 7. How does your robust models' accuracy change?

As you can see I ran a bunch of iterations on these models. As more iterations were introduced, the attacks became stronger. But between 5 and 7 iterations there were not large differences in accuracy.