Deep Learning Homework 2

İpek Erdoğan,2019700174

May 27, 2021

Explanations and Discussions

I have started with a base model, which I called Model 1. (Please see the "Model Drawings and Results" section for more details.) It had 3 convolutional layers followed by max-pooling layers and there were 2 fully connected layers at the end of the model. This model made me start with 63 accuracy in both training and testing. Then I tried to change the kernel sizes of these convolutional layers (Model 2) but it didn't effected results that much. I decided to add Batch Normalization between the convolutional layers and activation functions. This resulted with improvement in training and testing accuracies.

Then I made data augmentation with "Random Cropping" (Model 4 has no visualization since it's same with Model3 3. Nothing were different except data augmentation.) Random Cropping decreased model's performance. That's probably because there were no overfitting, the model haven't learn well yet. Improving generalization ended up with underfitting. Also, at this point, I have realized that adding max pooling layer after all of the convolutional layers may cause underfitting, too.

Since max-pooling layer undersamples it's inputs, I decided to remove one of the max-pooling layers (Model 5). I had a much better result. I also wanted to add "Random Vertical Flip" (Model 6) which ended up with again, bad results. So I removed it in Model 7 and also I have changed the first convolutional layer's kernel size. Again, my model's performance not good enough. I tried to remove the second max-pooling layer too, and changed the second convolutional layer's kernel size (Model 8). Still, results were not better. So I decided to go back to my best models architecture (Model 5) but I also removed the second max-pooling layer and add a second fully connected layer. This brought me my best model with best results (Model 9).

Adding one more fully connected layer probably prevented information loss. The dimension difference were high in between the input and output of the fully connected layers, when I was using 2 FC.

t-SNE Plotting

Figure 1: tSNE Plot, Epoch 0

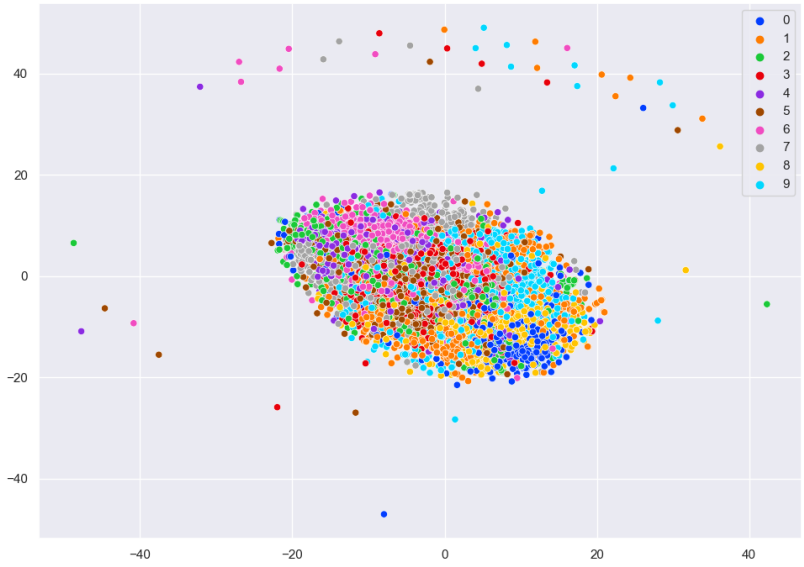

Figure 2: tSNE Plot, Epoch 20

Figure 3: tSNE Plot, Epoch 40

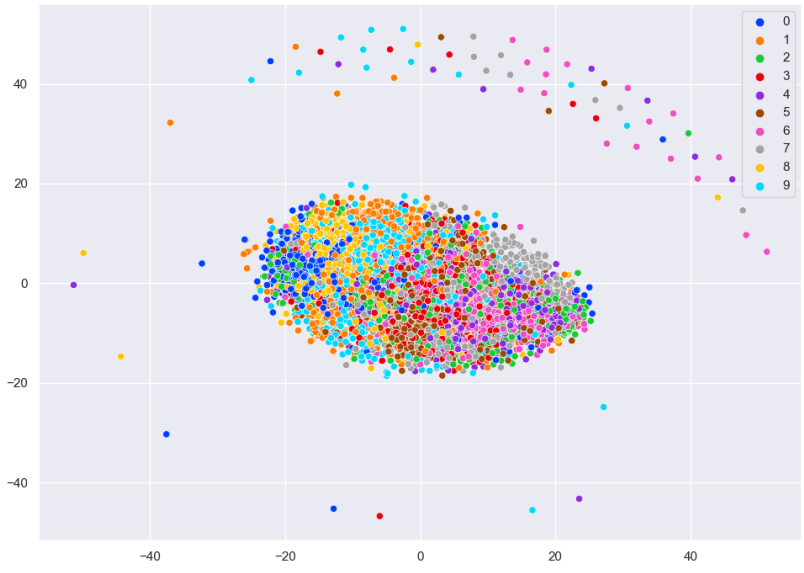

Figure 4: tSNE Plot, Epoch 50

Normally, I would expect tSNE plot to be more sparsed in last epochs. Yet it gave these graphs. It may because my latent representation dimension was high(2704) or my tSNE function's iteration parameter was not high enough.

Model Drawings and Results

Model 1

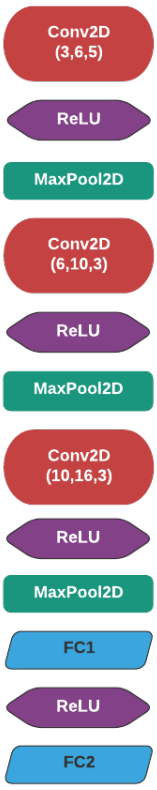

Figure 5: Model 1

Training Accuracy: %65
Testing Accuracy: %63

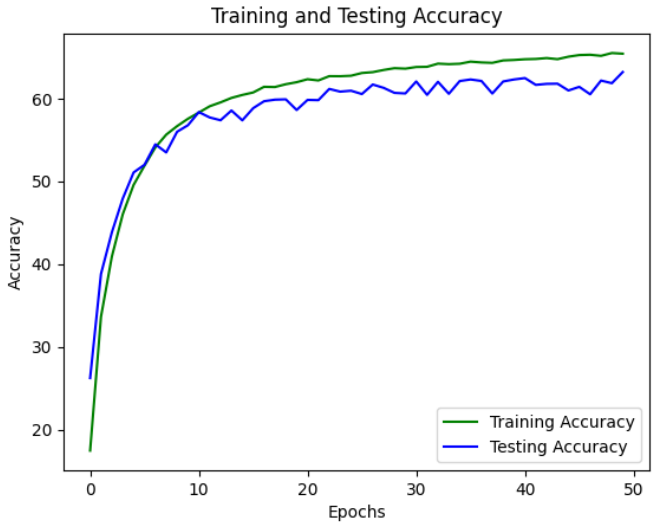

Figure 6: Model 1 Accuracy

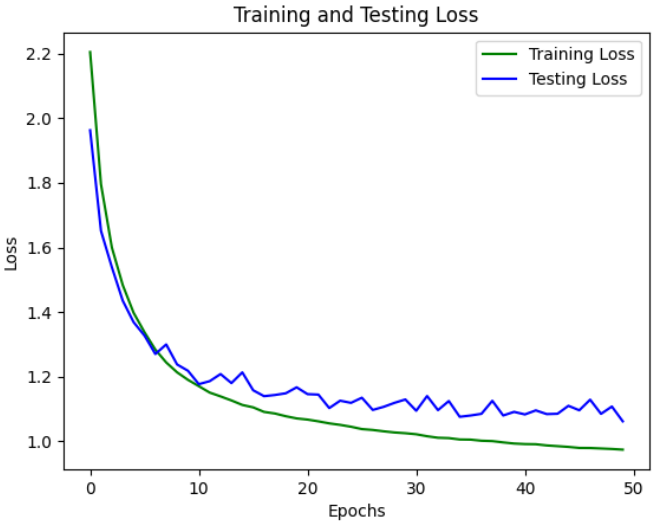

Figure 7: Model 1 Loss

Model 2

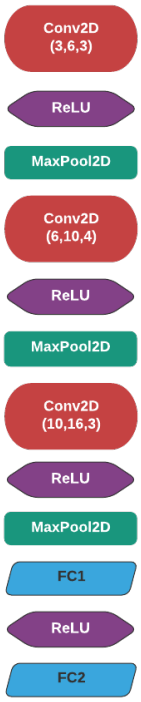

Figure 8: Model 2

Training Accuracy: %64
Testing Accuracy: %60

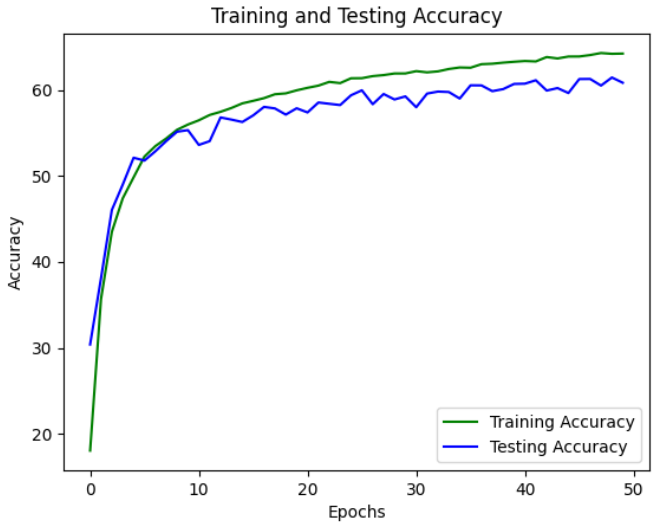

Figure 9: Model 2 Accuracy

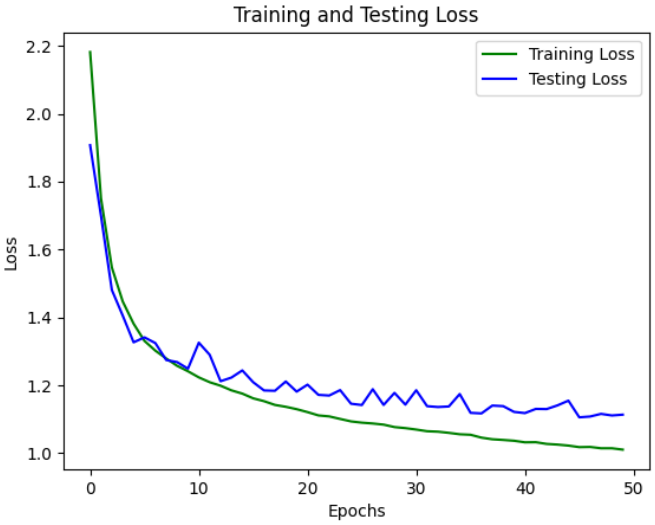

Figure 10: Model 2 Loss

Model 3

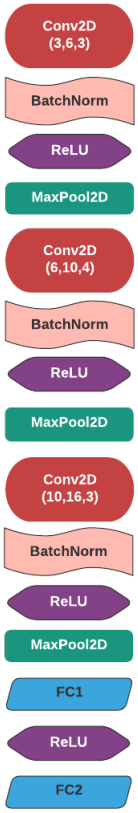

Figure 11: Model 3

Training Accuracy: %67
Testing Accuracy: %62

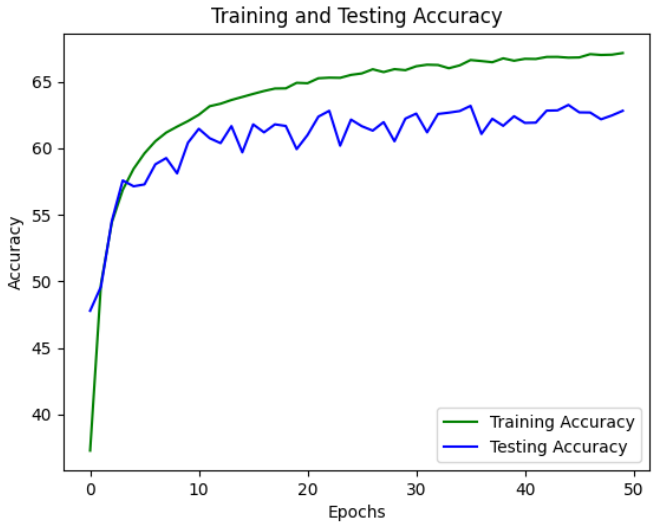

Figure 12: Model 3 Accuracy

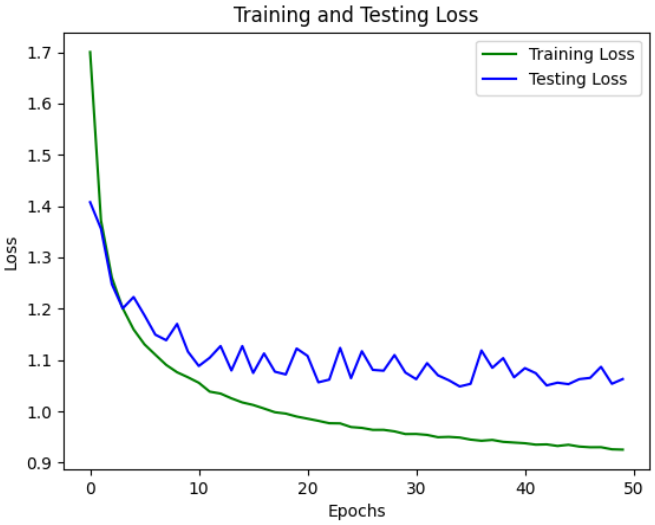

Figure 13: Model 3 Loss

Model 4

Training Accuracy: %60
Testing Accuracy: %62

Figure 14: Model 4 Accuracy

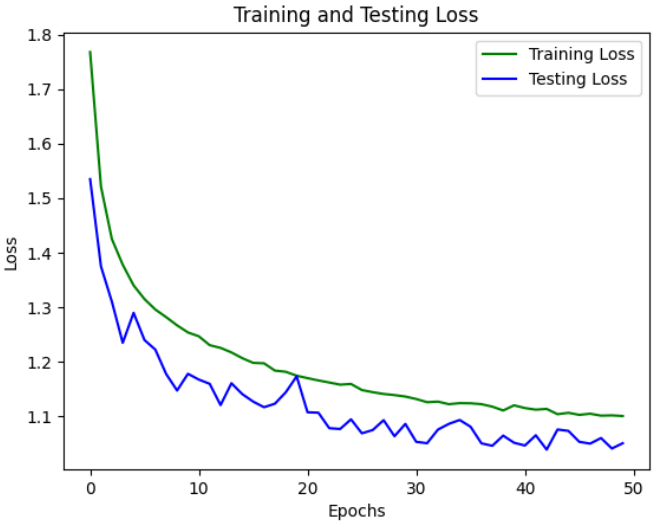

Figure 15: Model 4 Loss

Model 5

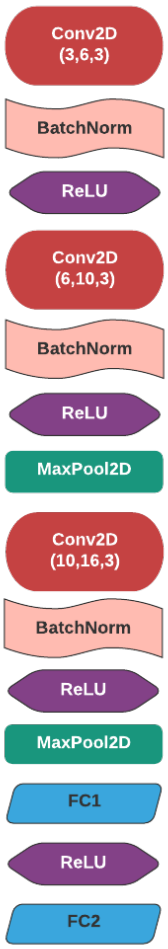

Figure 16: Model 5

Training Accuracy: %68
Testing Accuracy: %70

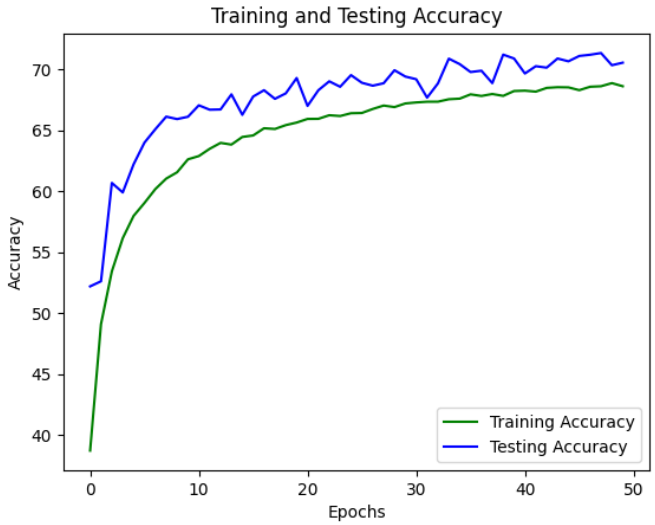

Figure 17: Model 5 Accuracy

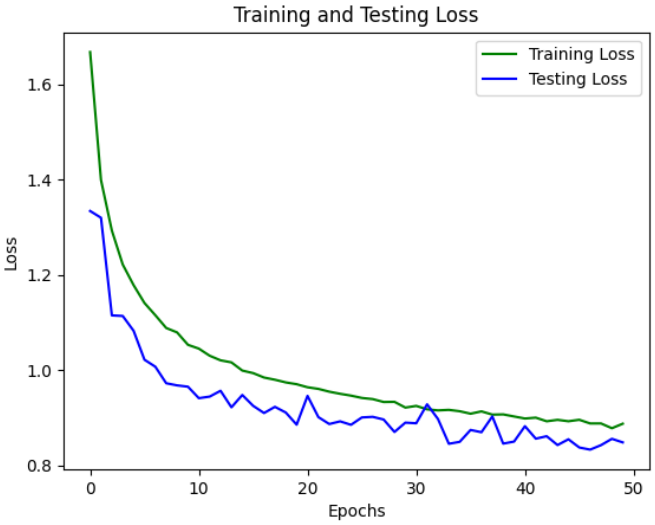

Figure 18: Model 5 Loss

Model 6

Training Accuracy: %62
Testing Accuracy: %65

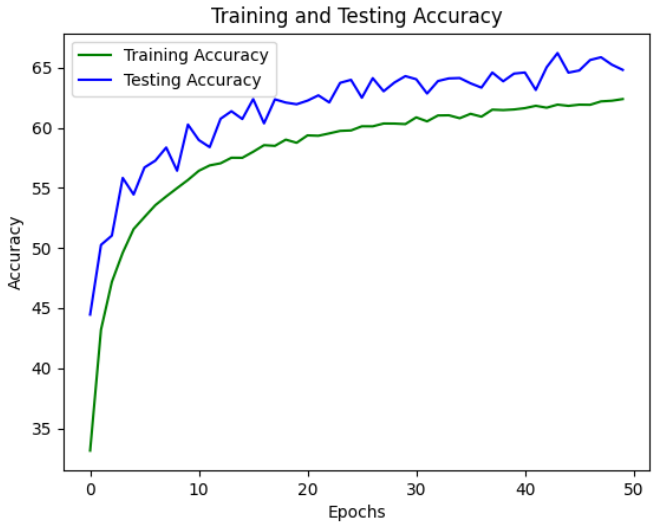

Figure 19: Model 6 Accuracy

Figure 20: Model 6 Loss

Model 7

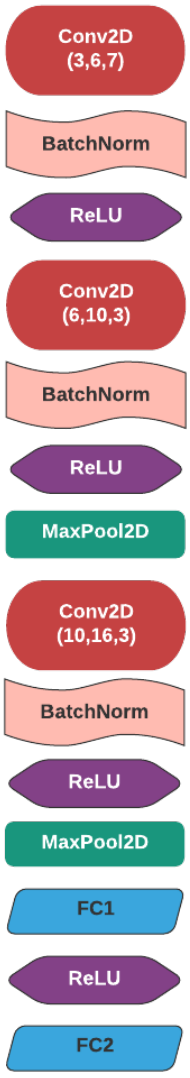

Figure 21: Model 7

Training Accuracy: %66
Testing Accuracy: %68

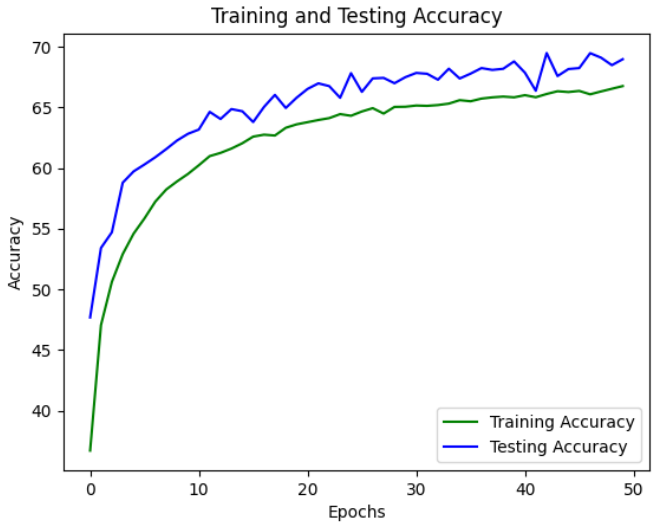

Figure 22: Model 7 Accuracy

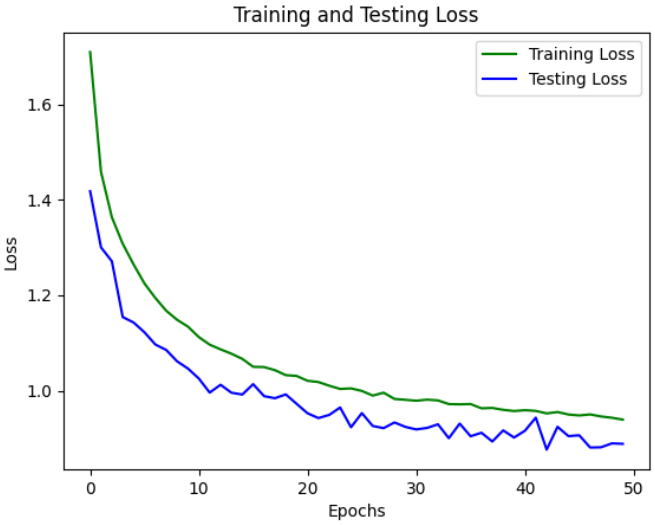

Figure 23: Model 7 Loss

Model 8

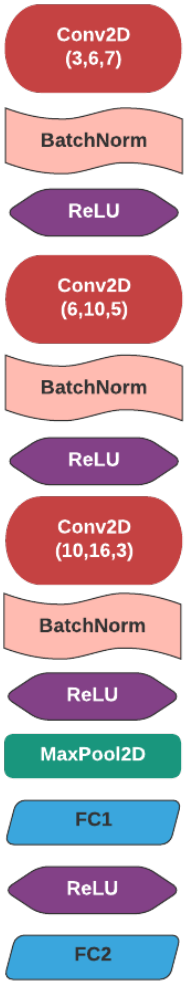

Figure 24: Model 8

Training Accuracy: %64
Testing Accuracy: %66

Figure 25: Model 8 Accuracy

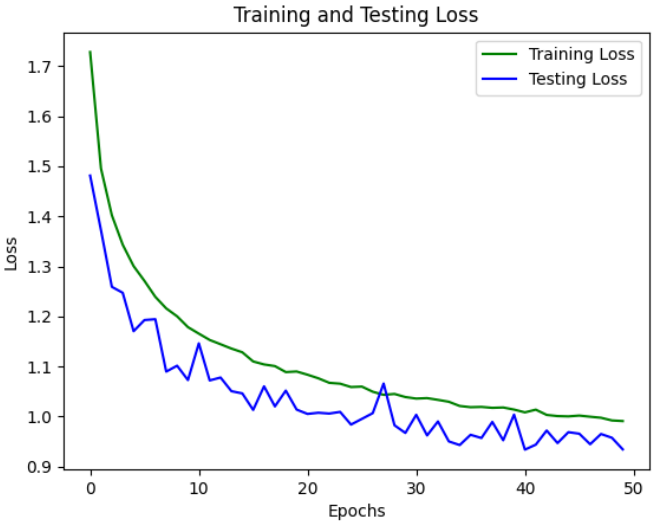

Figure 26: Model 8 Loss

Model 9

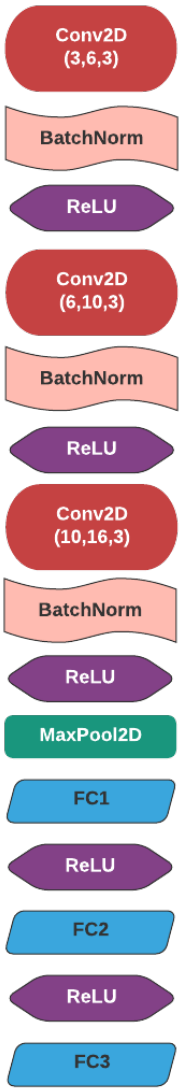

Figure 27: Model 9

Training Accuracy with SGD(momentum): %79
Testing Accuracy with SGD(momentum: %75

Training Accuracy with Adam: %74
Testing Accuracy with Adam: %73

Training Accuracy with RNSprob: %71
Testing Accuracy with RNSprob: %72

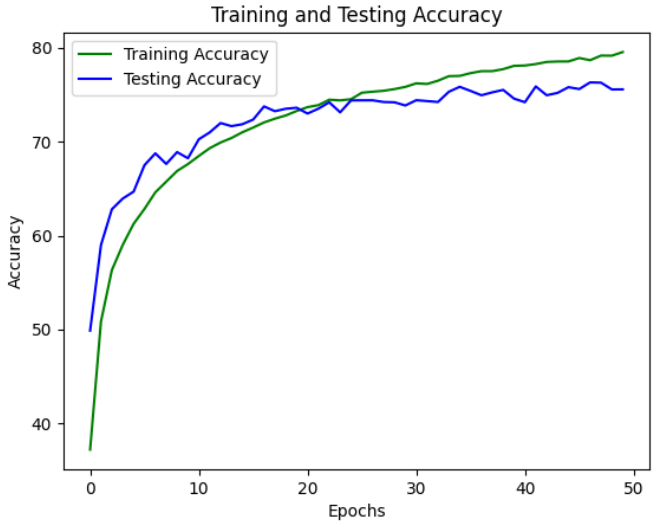

Figure 28: Model 9 Accuracy with SGD(Momentum)

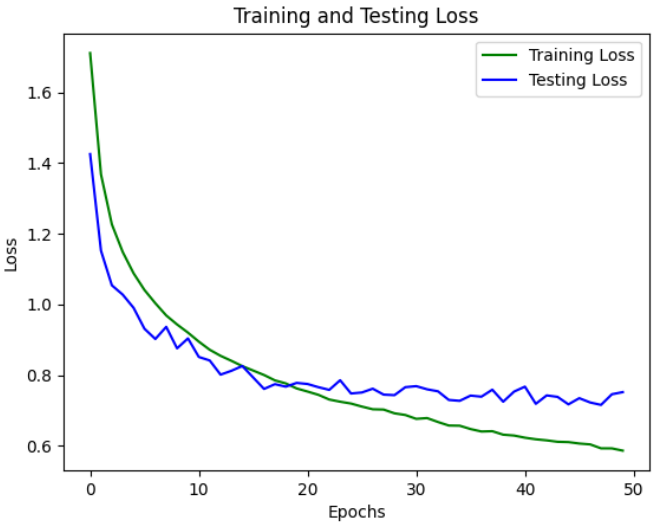

Figure 29: Model 9 Loss with SGD(Momentum)

Figure 30: Model 9 Accuracy with Adam

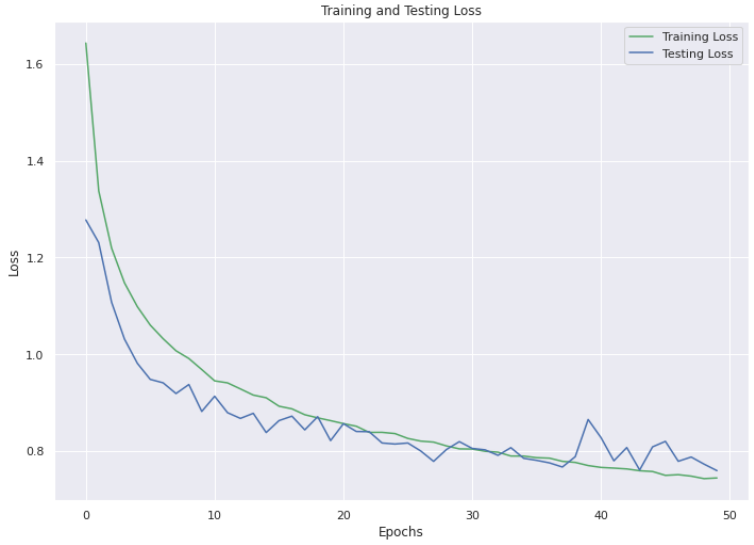

Figure 31: Model 9 Loss with Adam

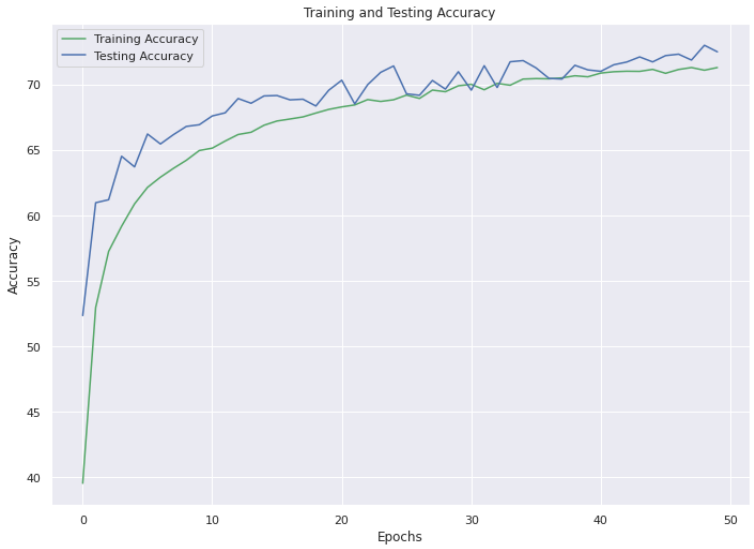

Figure 32: Model 9 Accuracy with RNSprob

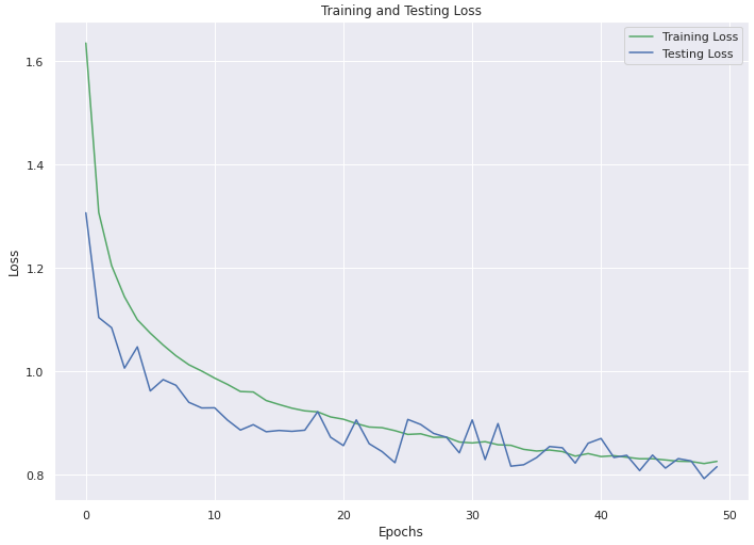

Figure 33: Model 9 Loss with RNSprob

Additional Notes

Lecture slides, Pytorch Documentations [1], Pytorch Tutorial [2],"Deep Learning" book [3] and Batch Norm Paper [4] have been used during this study. For plotting the tSNE graph, a ready-to-go code has been used [5].

References

[1] "Pytorch Documentation", https://pytorch.org/docs/master/nn.html.

[2] "Pytorch Tutorial", https://github.com/pytorch/tutorials/blob/master/beginner_source/blitz/cifar10_tutorial.py.

[3] Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016, http://www.deeplearningbook.org.

[4] Ioffe, S. and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 448–456, JMLR.org, 2015.

[5] "Visualizing Word Vectors with t-SNE", https://www.kaggle.com/jeffd23/visualizing-word-vectors-with-t-sne.