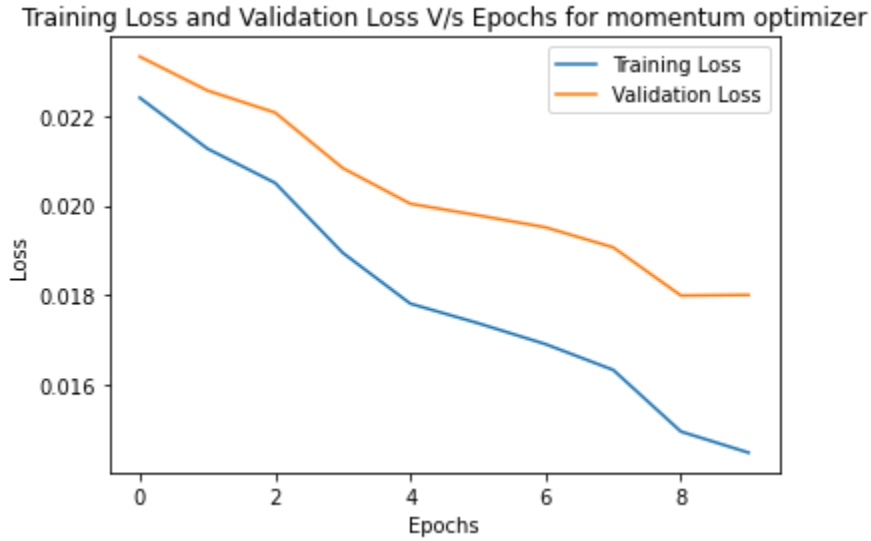
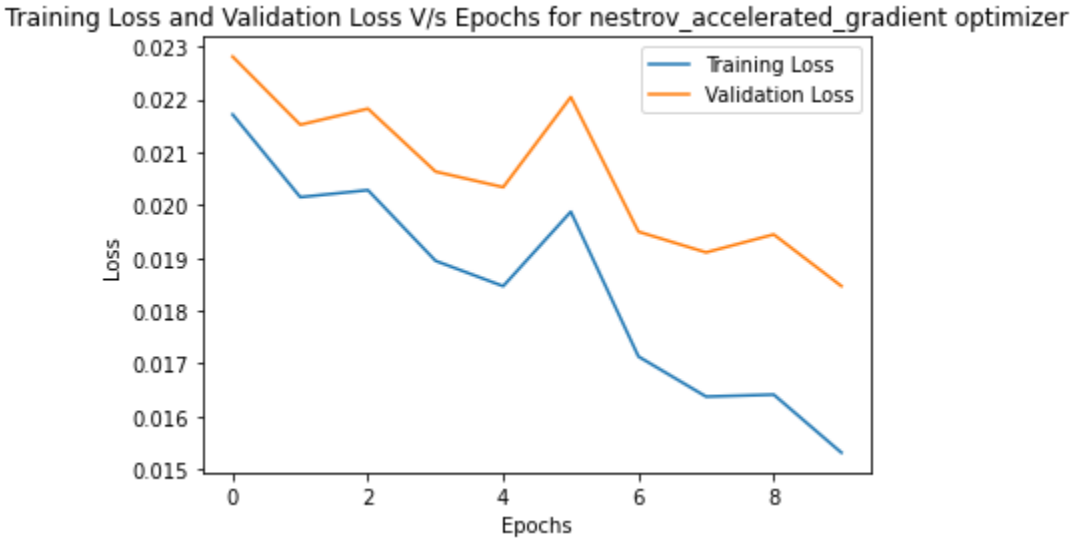
DL Assignment 2 Report

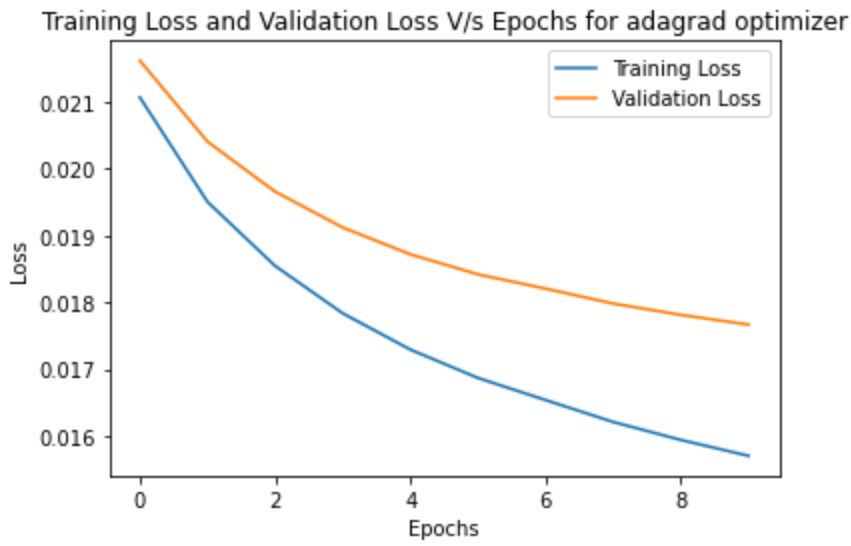
Neemesh Yadav, 2020529
Mohammad Aflah Khan, 2020082

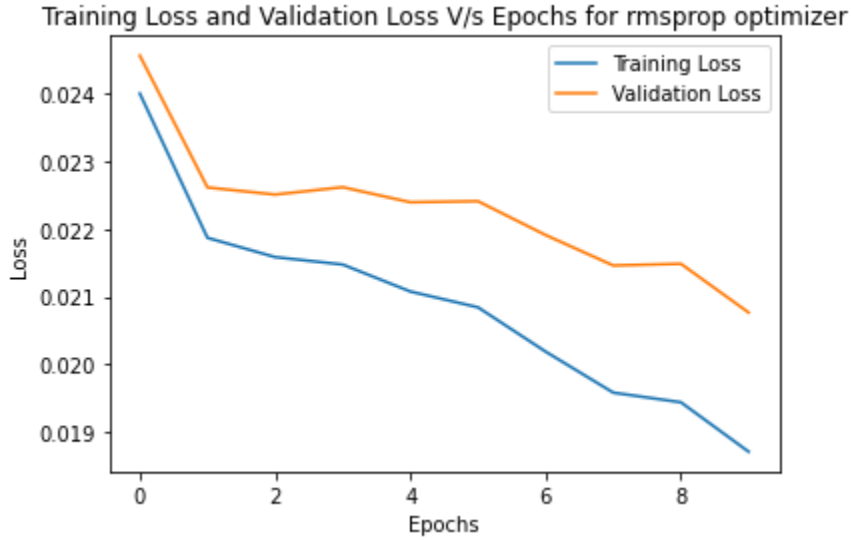
Part A

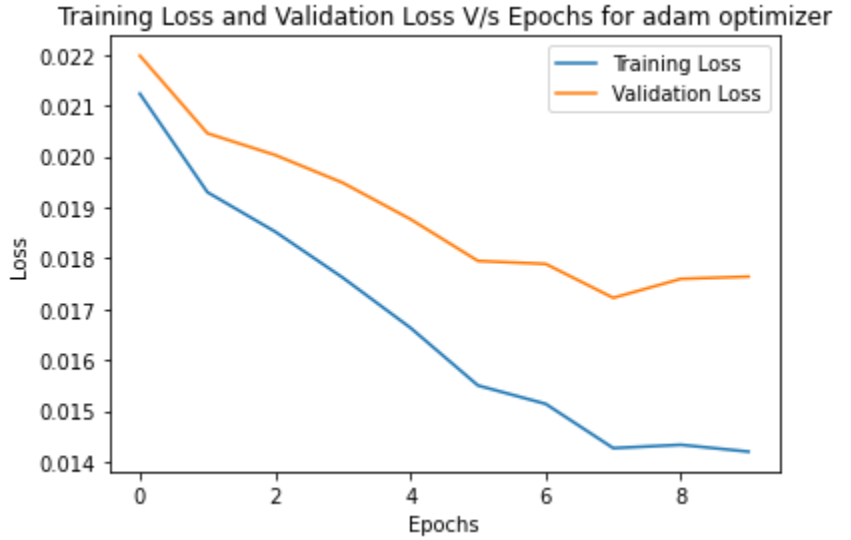
a) Loss Plots -



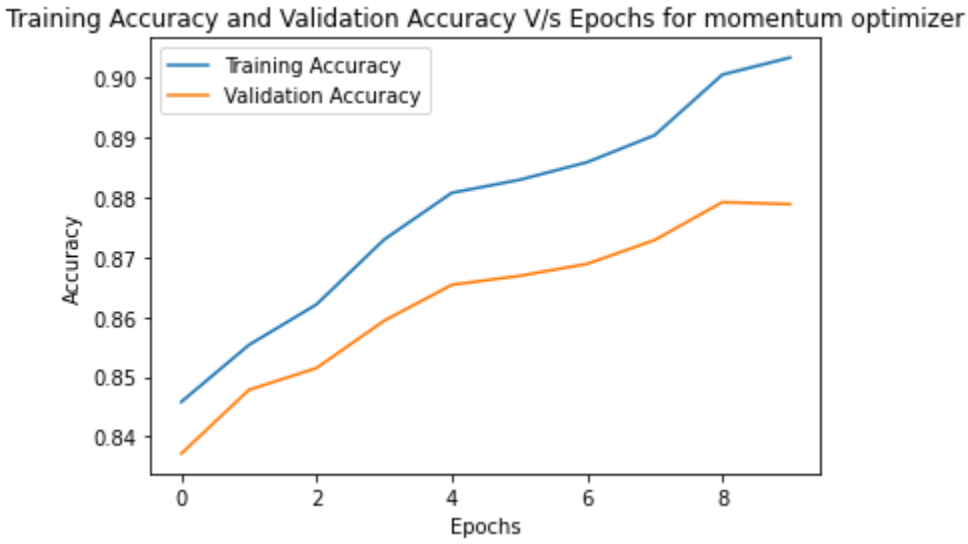


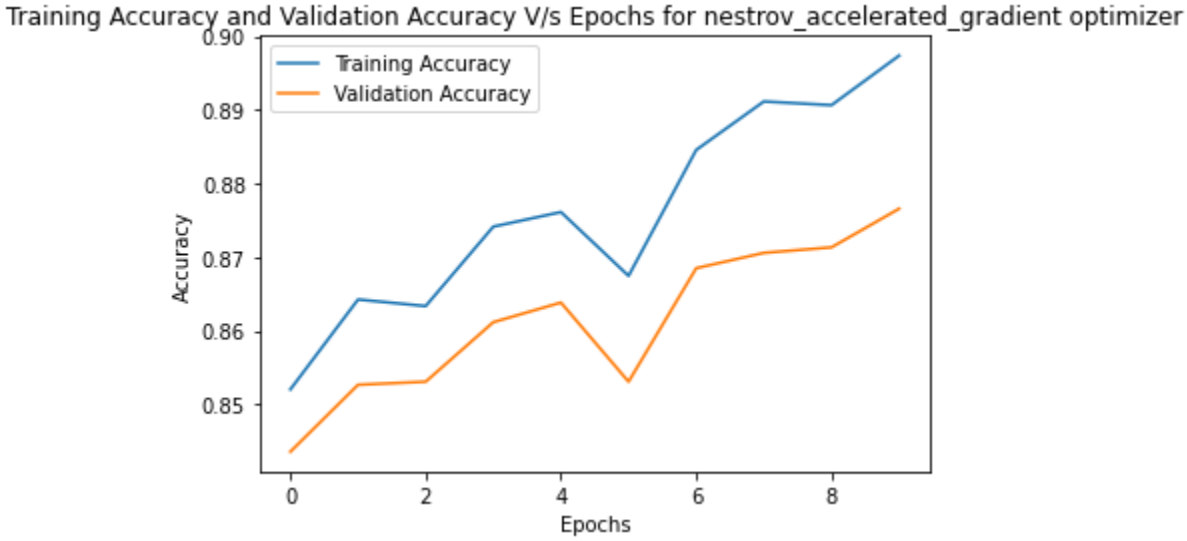

Training Loss and Validation Loss V/s Epochs for adagrad optimizer
Training Loss
Validation Loss
Loss
Epochs
0.021
0.020
0.019
0.018
0.017
0.016
0
2
4
6
8


Training Loss and Validation Loss V/s Epochs for rmsprop optimizer
Training Loss
Validation Loss
Loss
Epochs
0.024
0.023
0.022
0.021
0.020
0.019
0
2
4
6
8

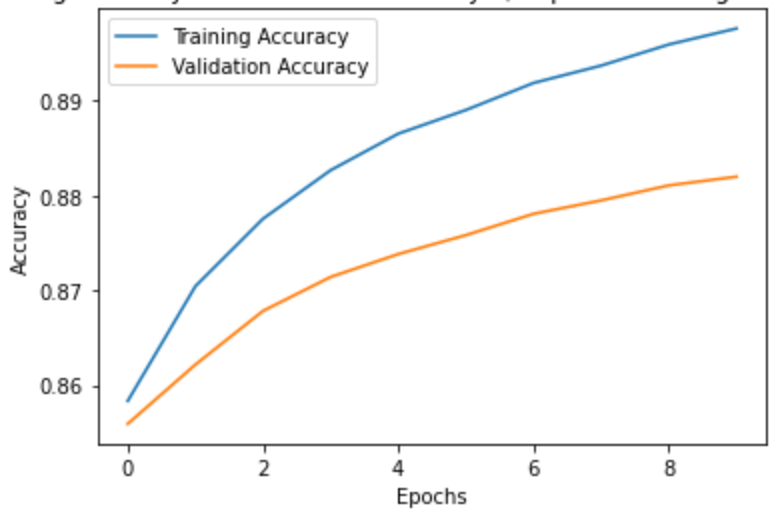

Training Loss and Validation Loss V/s Epochs for adam optimizer
Training Loss
Validation Loss
Loss
Epochs
0.022
0.021
0.020
0.019
0.018
0.017
0.016
0.015
0.014
0
2
4
6
8

b) Accuracy Plots

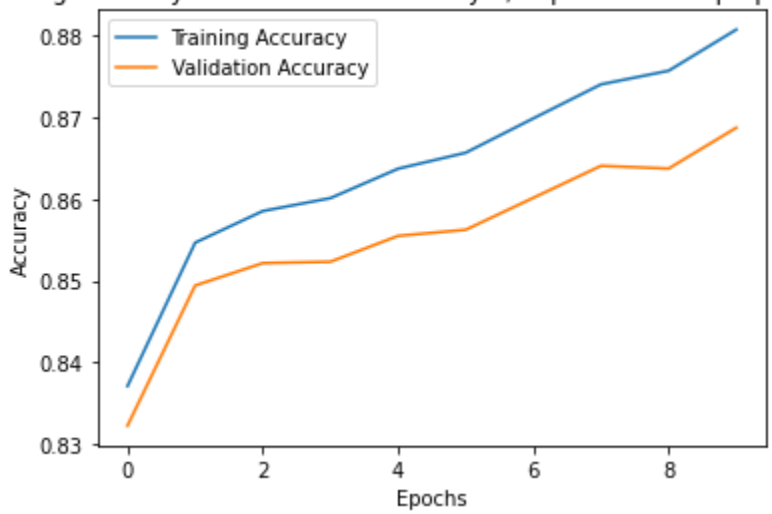




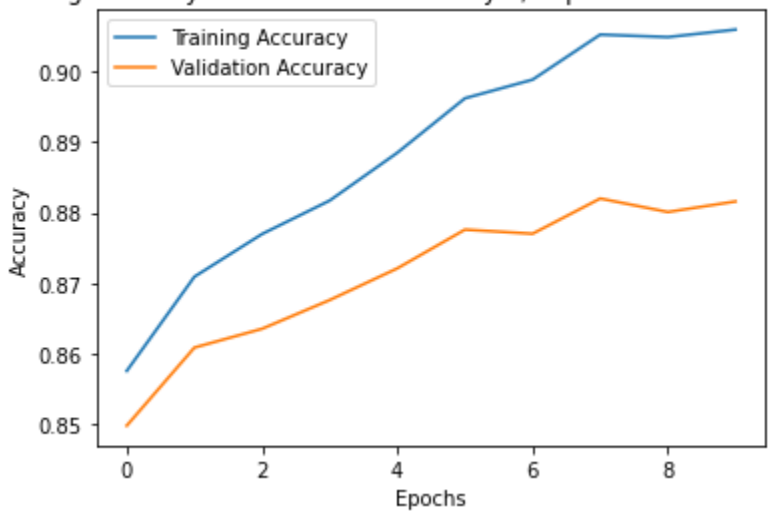
Training Accuracy and Validation Accuracy V/s Epochs for adagrad optimizer



Training Accuracy and Validation Accuracy V/s Epochs for rmsprop optimizer



Training Accuracy and Validation Accuracy V/s Epochs for adam optimizer

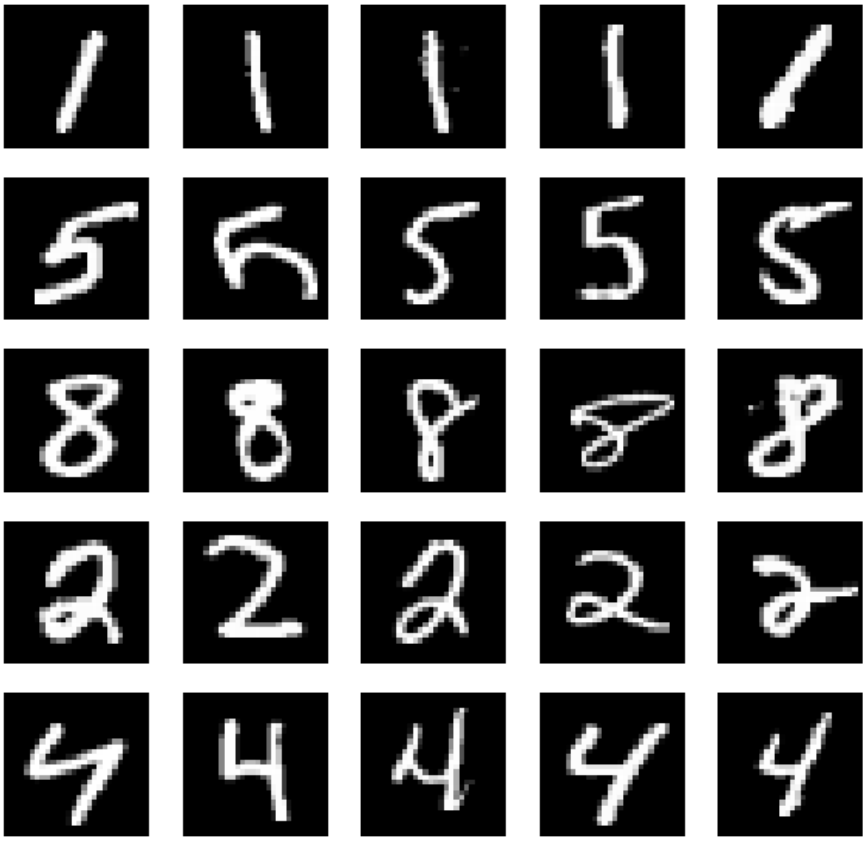


c) The above plots show that all the models converge to a reasonable accuracy within 10 epochs. The validation and training curves also stay pretty close showing that the model is learning well and not yet overfitting. Even after the 1st epoch all models are at an 80%+ Accuracy and the graphs seem to show a trend of Momentum being slower than Nestrov Accelerated Momentum. Overall all the optimizers seem to work well and reach around 88% Testing Accuracy.
The hyperparameters used are:
   1) Momentum - 'gamma': 0.9, ‘lr’: 0.01
   2) Nestrov Accelerated Momentum - 'gamma': 0.9, ‘lr’: 0.01
   3) Adagrad - 'epsilon': 1e-8, ‘lr’: 0.01
   4) RMSProp - 'epsilon': 1e-8, 'beta': 0.9, ‘lr’: 0.0001
   5) Adam - 'beta1': 0.9, 'beta2': 0.999, 'epsilon': 1e-6, ‘lr’: 0.0001

Part B

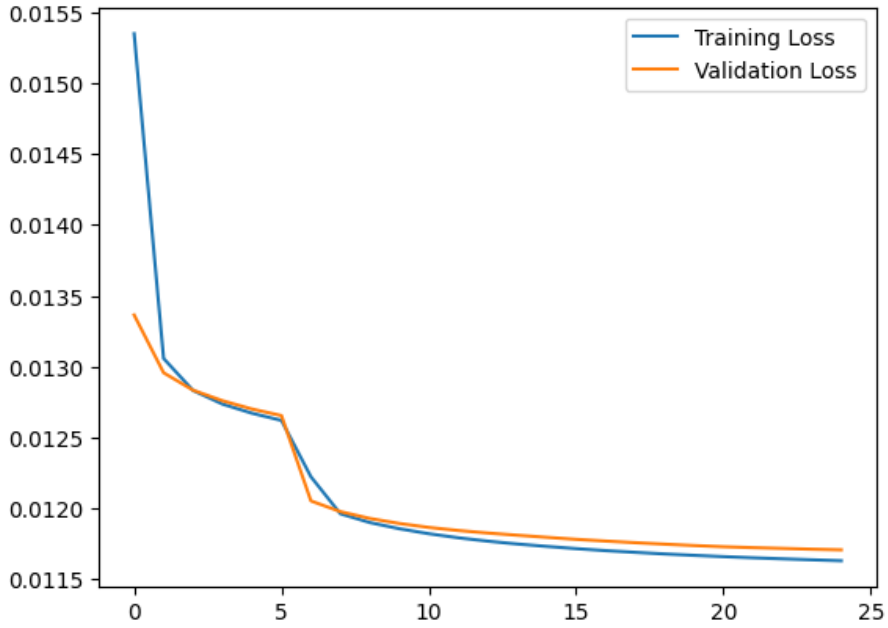
The hyperparameters used for this and the consequent parts are:
BATCH_SIZE = 128
NUM_EPOCHS = 5
LEARNING_RATE = 0.0001
SEED = 2809

Randomly sampled 5 images with their labels from the train dataset:

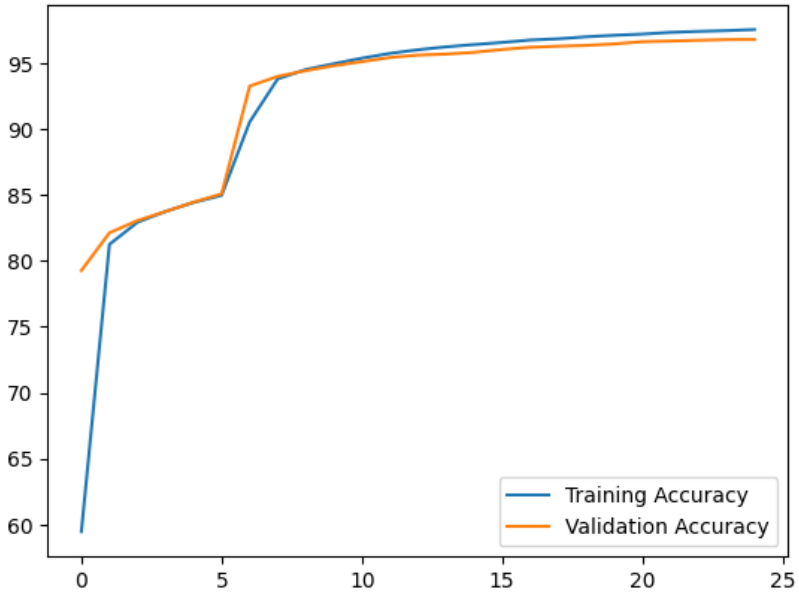


The experiments we followed in the next two parts were done using a random seed of 2809, to ensure that all the experiments follow the same form of splitting of data.
All the X-Axis values denote the number of epochs, and all the Y-Axis values denote either the Loss, or Accuracy.

1) CNN Architecture without residual connections
    a) Loss Plots



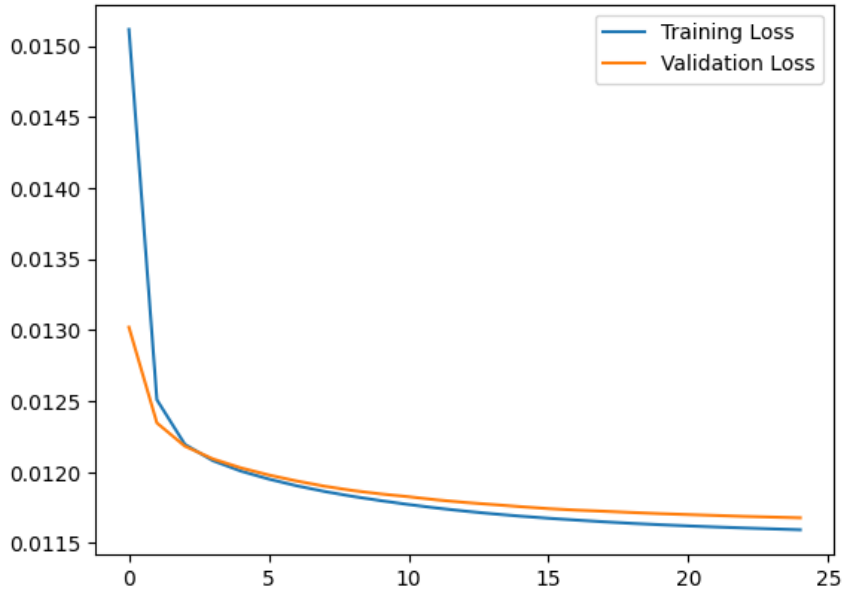
    b) Accuracy Plots



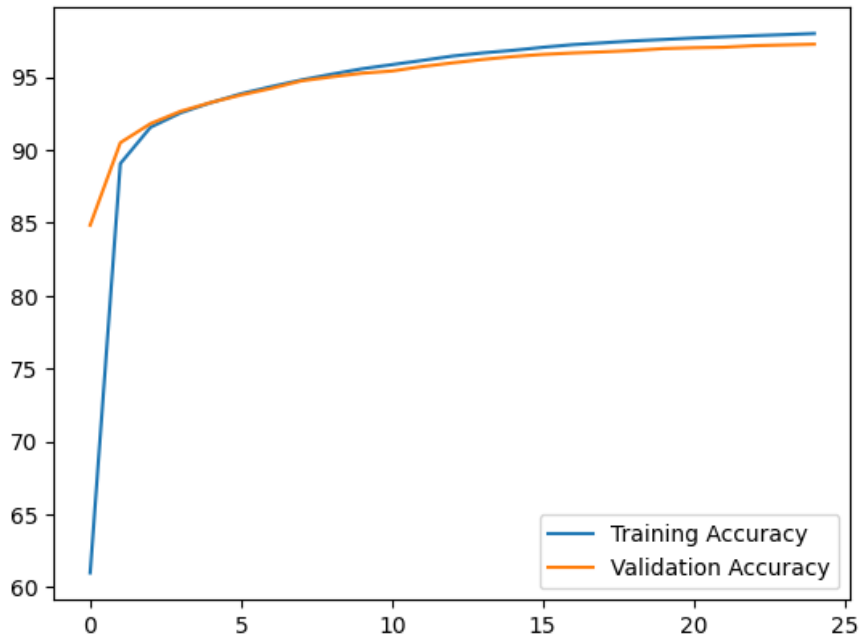
    c) The above loss plot shows that our model is able to converge after a certain number of epochs and how its performance saturates after a point. The accuracy

plot also shows that the model's training was stable and performance was ever increasing before reaching a saturation.

2) With a residual connection from input to the output of 2nd conv layer.
   a) Loss Plots



   b) Accuracy Plots



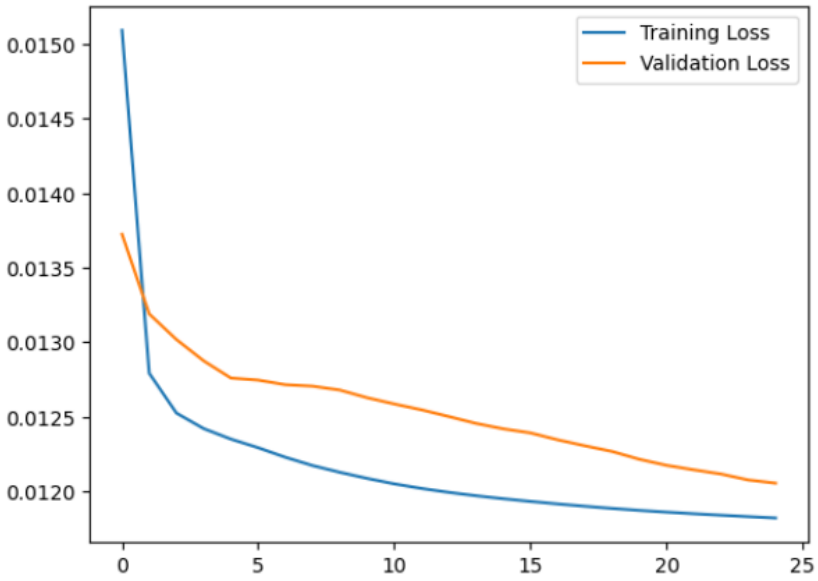
   c) The above loss plot shows that this model is also able to converge after a certain number of epochs, but an important thing to note here is how the loss dips, which also implies that the performance increase is substantial after a certain point, but saturates over the remaining number of epochs.

Our resnet model works better and hence we use it for our data augmentation experiments.

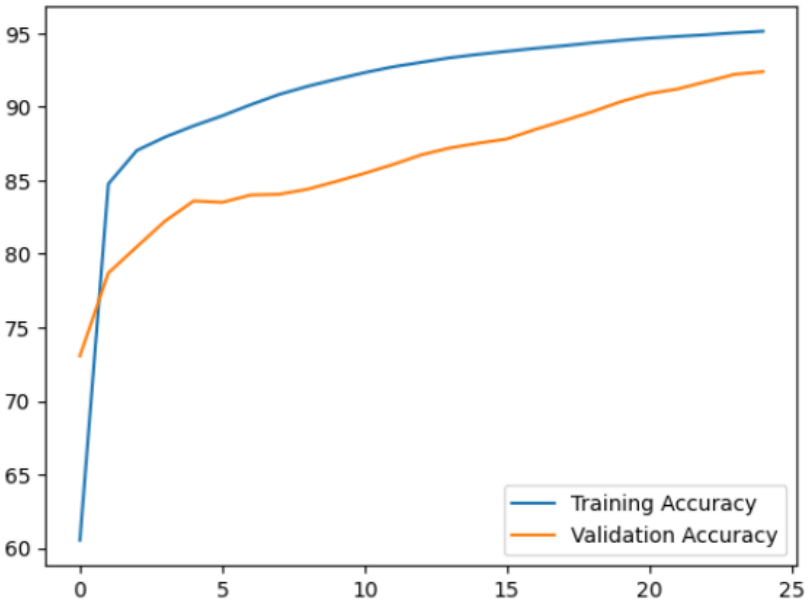
Part C

Data Augmentation Techniques on CNN without residual connection. We didn't implement resizing of image as it was clarified in the comments that there is no need to do so.

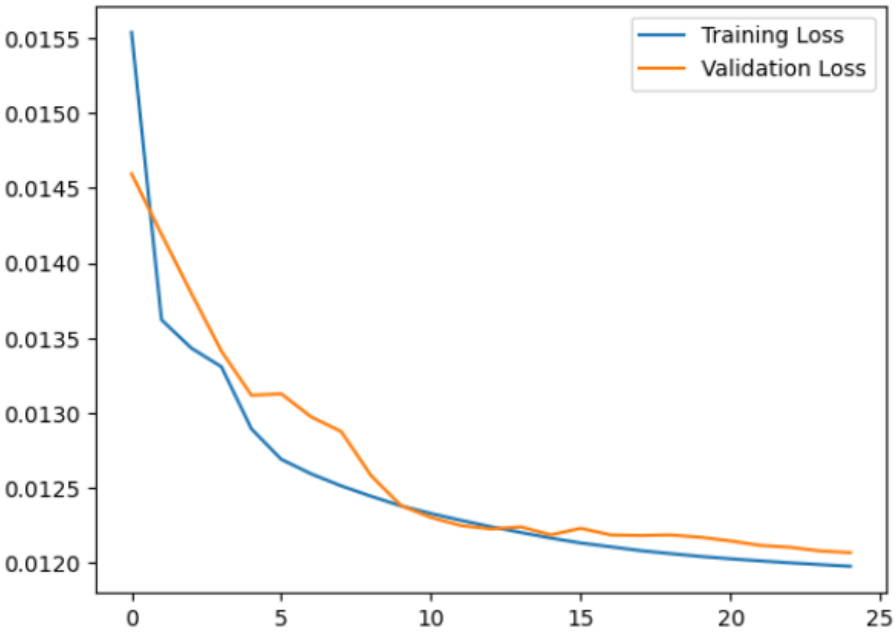
1) Left flipping the original data.
    a) Loss Plots



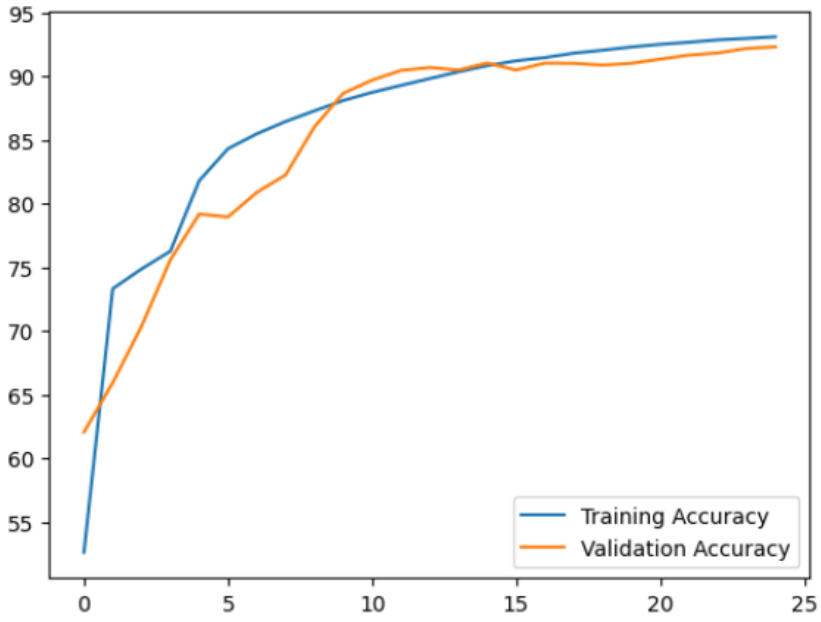
    b) Accuracy Plots



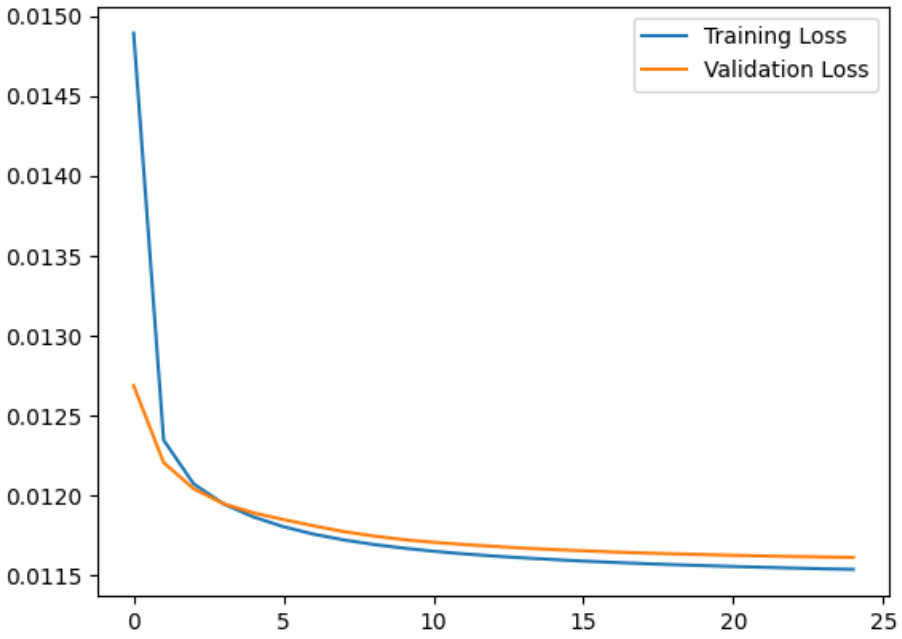
2) Rotating the data by 90 degrees
    a) Loss Plots



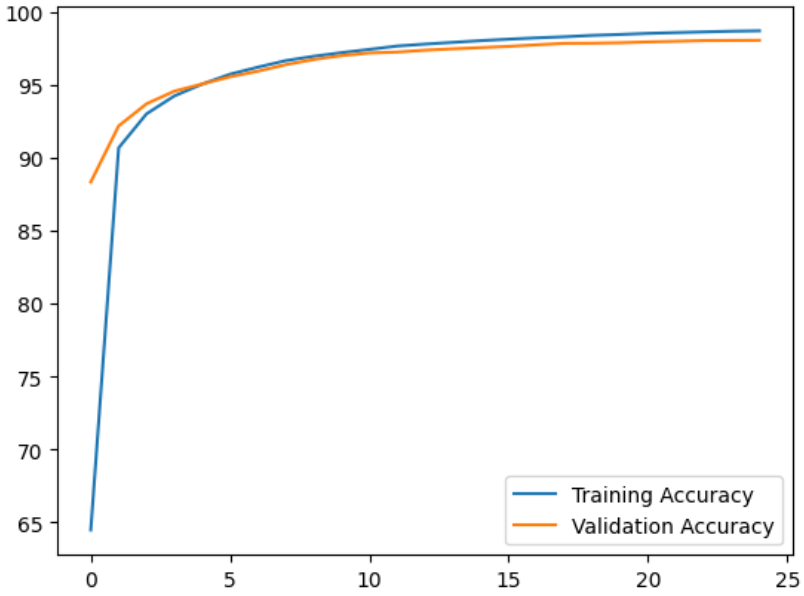
b) Accuracy Plots



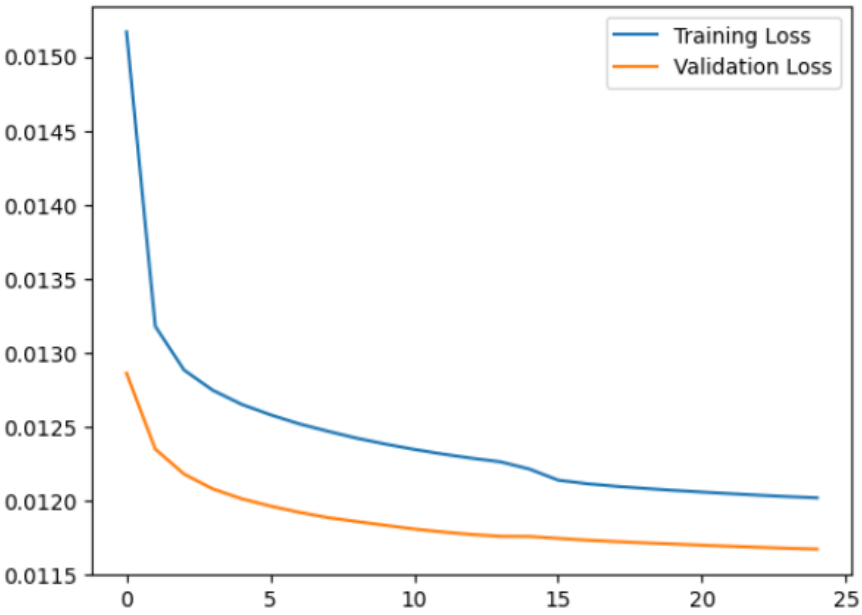
3) Adding some Gaussian Noise
   a) Loss Plots



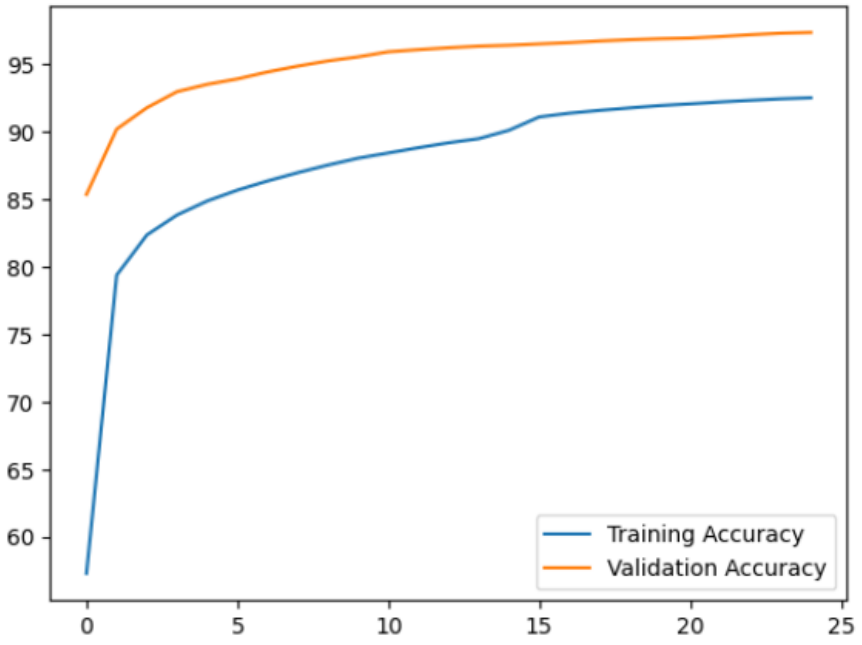
b) Accuracy Plots



4) Combining all these techniques
   a) Loss Plots



b) Accuracy Plots



After applying the left flip augmentation to the data, the loss plots show that we might expect a slower training in this case in comparison to the original Residual Connection architecture we implemented. Applying Rotation augmentation and Gaussian Noise to the data, we see that the results here are quite different as there is a much quicker convergence, but unstable training for rotation, whereas there is a much smoother training for the gaussian noise augmentation. After combining all the data augmentation techniques, we made an interesting and a counter-intuitive observation that our validation loss was coming out to be much lower than the training loss.

We see an improvement in performance on the model trained after applying the gaussian noise augmentation to the training data.