ELEC 576/COMP 576 Fall 2020 : Assignment2

Mst Afroja Akter

October 2020

1 Visualizing a CNN with CIFAR10

(a). CIFAR10 Dataset

CIFAR10 dataset has natural images representing 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. In the problem we used Grayscale images of size 28x28. As a part of prepossessing, we normalized the images dividing by 255 and used one-hot encoding for the image labels.

(b)LeNet5 on CIFAR10 dataset

We used the following configuration to develop LeNet to train CIFAR10 dataset:

1. Convolutional layer with 5x5 kernel, 32 filters and ReLU activation.
2. Max Pooling layer subsampling by 2.
3. Convolutional layer with 5x5 kernel, 64 filters and ReLU activation.
4. Max Pooling layer subsampling by 2.
5. Fully Connected layer with input shape 7x7x64 and output shape1024.
6. Fully Connected layer with input shape 1024 and output shape10.
7. Softmax layer(Softmax Regression + Softmax Nonlinearity).

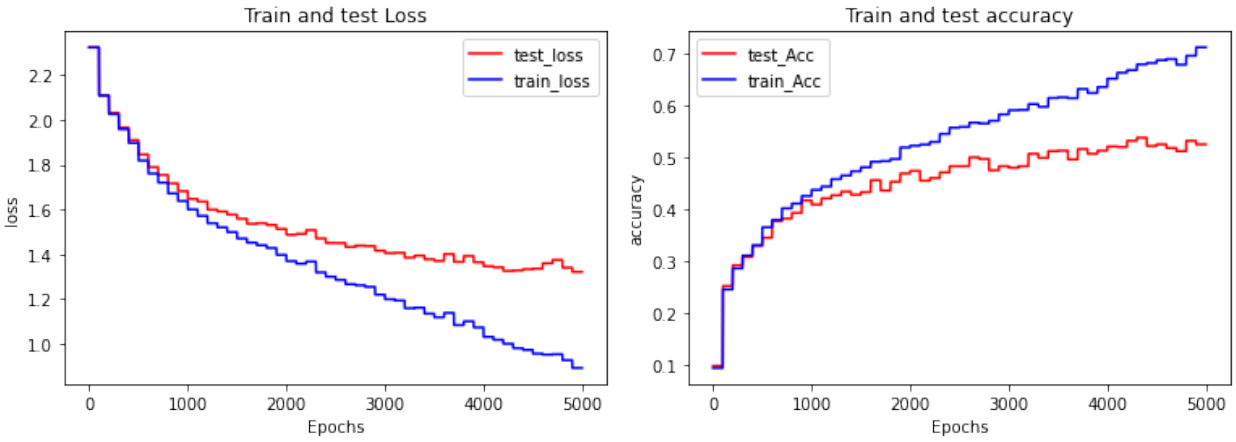


Figure 1: Training and test loss and accuracy curve of Grayscale CIFAR10 dataset.

The network has *Cross Entropy* loss function and *Adam Optimizer* with learning rate $1e{-}4$ and iterate 5500 times to train LeNet5. Figure 1 shows the training and test loss and accuracy of LeNet5 model on

CIFAR10 dataset. Here we see that the LeNet model is doing better on training data but after around 1000 epoch model performs very poorly on test data (only 52% test accuracy). The big gap between train and test loss curves and accuracy curves show that the model is over fitting.

Figure 2 shows the weights of the first convolutional layer on CIFAR10 Grayscale dataset for LeNet5 network. The table shows the statistics of activations in convolutional layers on the test images for LeNet5 network.

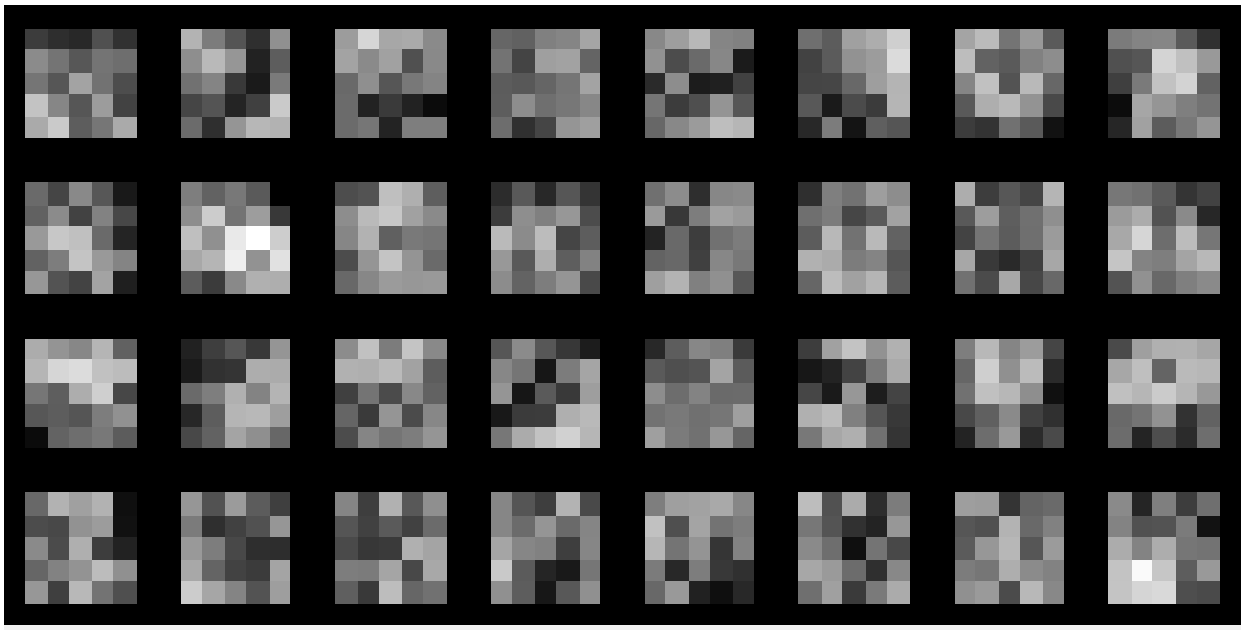
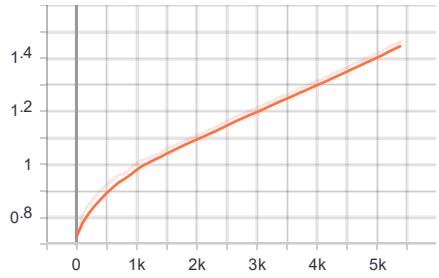
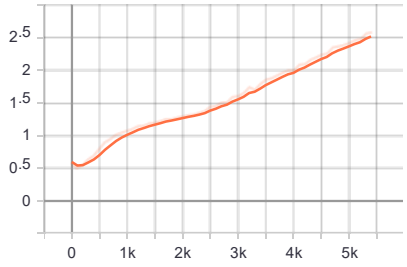
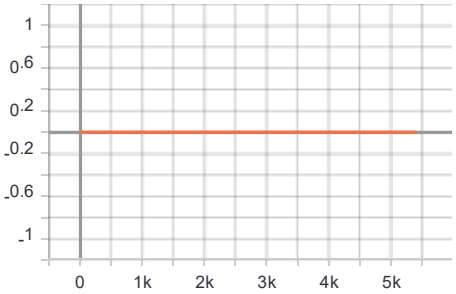
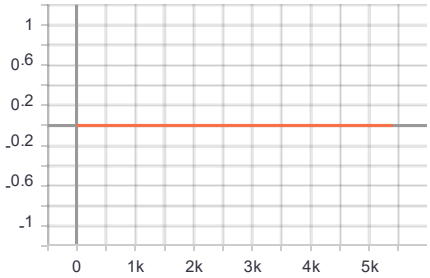
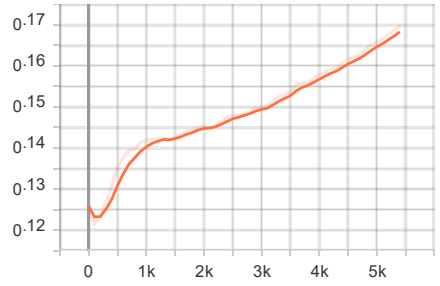
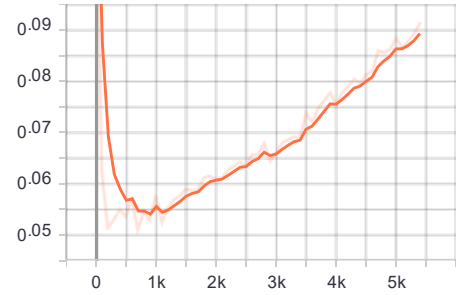
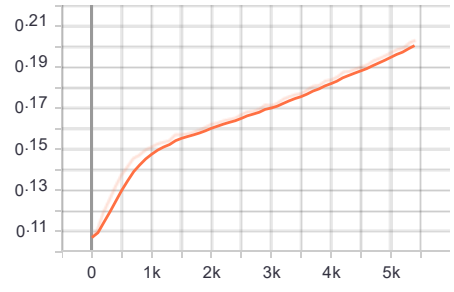
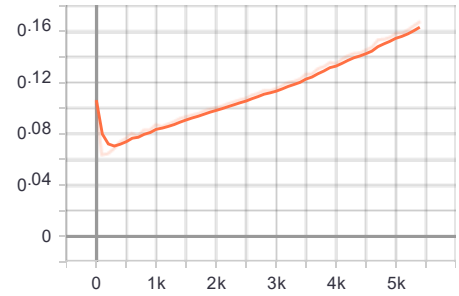
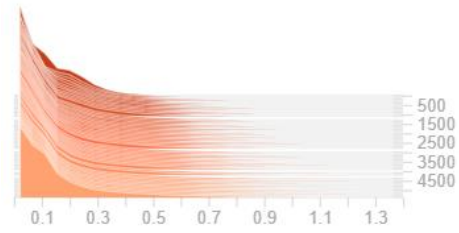
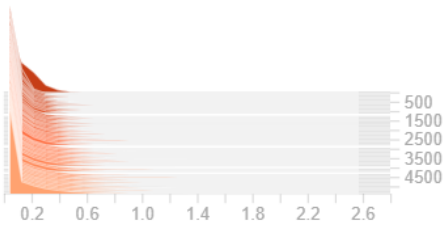


Figure 2: Weight's of first convolutional layer on Grayscale CIFAR10 dataset with LeNet5 network.

Statistics of activations in convolutional layers on the test images on LeNet5 network	
Convolutional Layer 1	Convolutional Layer 2
Conv1: Max of activation after ReLU 	Conv2: Max of activation after ReLU 
Conv1: Min of activation after ReLU 	Conv2: Min of activation after ReLU 
Conv1: Mean of activation after ReLU 	Conv1: Mean of activation after ReLU 
Conv1: Std of activation after ReLU 	Conv1: Std of activation after ReLU 
Conv1: Histogram of activation after ReLU 	Conv1: Histogram of activation after ReLU 

I tried with a different model architecture with adam optimizer and learning rate 0.001 and applied batch normalization and dropout regularization:

1. Convolutional layer with 5x5 kernel, 32 filters and ReLU activation.
2. Max Pooling layer subsampling by 2 and Batch-Normalization
3. Convolutional layer with 3x3 kernel, 64 filters and ReLU activation.
4. Max Pooling layer subsampling by 2 and then Batch-Normalization and dropout
5. Convolutional layer with 3x3 kernel, 64 filters and ReLU activation.
6. Max Pooling layer subsampling by 2 and Batch-Normalization
7. Fully Connected layer with input shape 4x4x64 and output shape256 and then Batch-Normalization and dropout
8. Fully Connected layer with input shape 256 and output shape10.
9. Softmax layer(Softmax Regression + Softmax Nonlinearity).

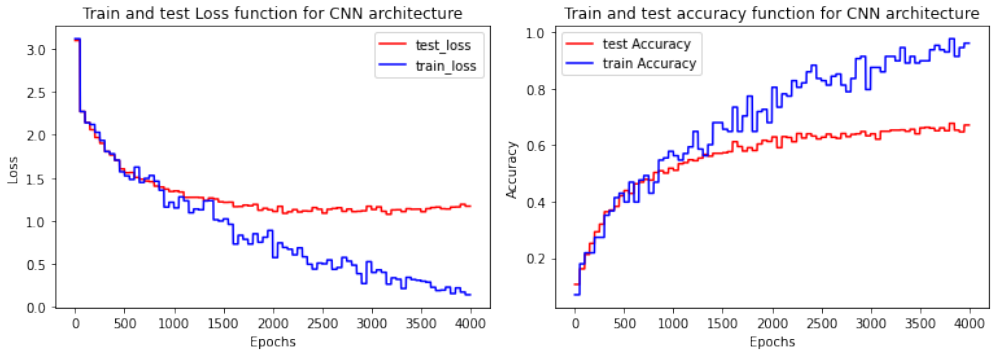


Figure 3: Training and test loss and accuracy curve of Grayscale CIFAR10 dataset.

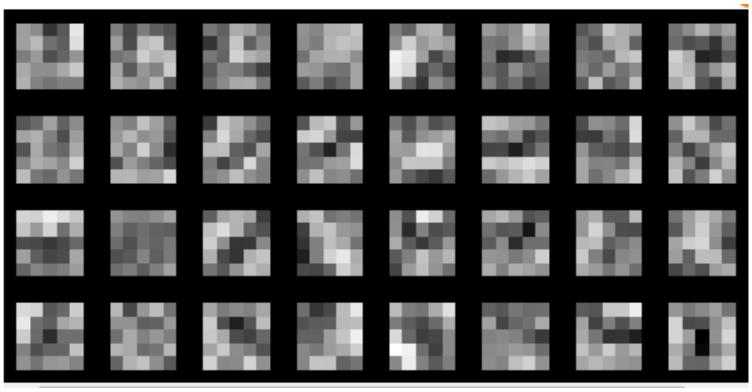
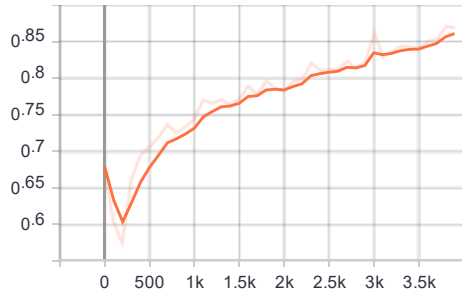
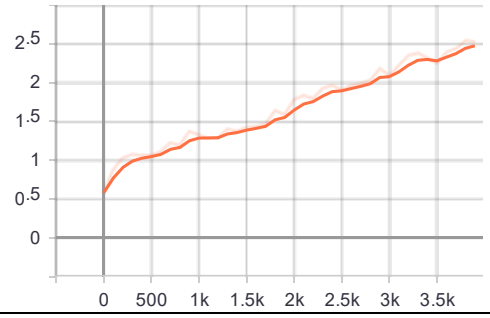
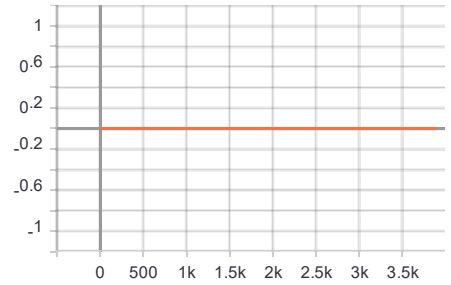
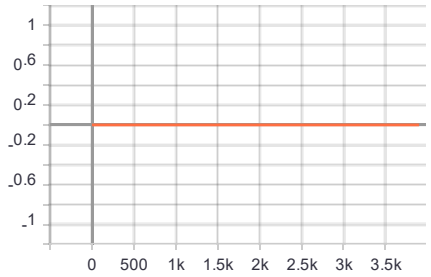
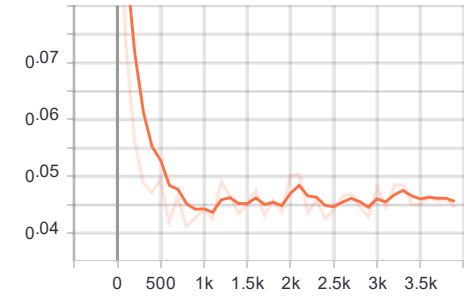
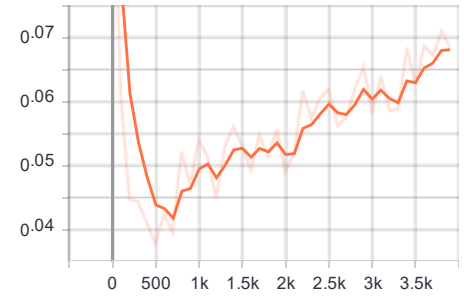
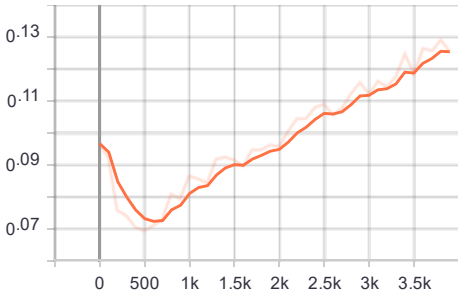
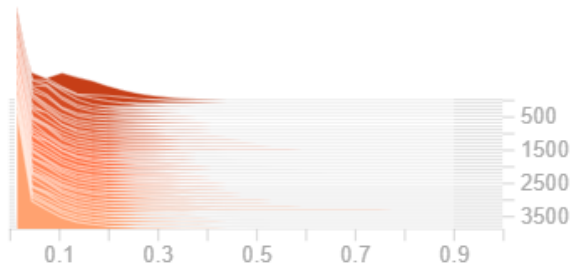
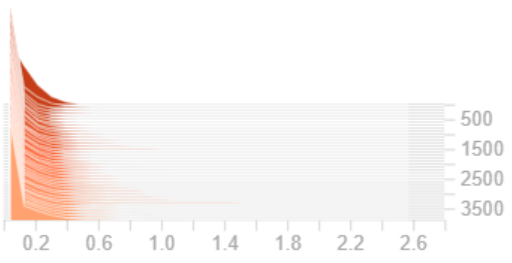


Figure 4: Weight's of first convolutional layer on Grayscale CIFAR10 dataset with new CNN network.

The new CNN model gave little better performance on test data with 67.1% test accuracy. Figure 3 shows the model performance on both test and training data. The model is still overfitting, but with time and GPU limitation I couldn't get better model at this moment.

Figure 4 shows the weights of the first convolutional layer on CIFAR10 Grayscale dataset for new network. The table shows the statistics of activations in convolutional layers on the test images.

Statistics of activations in convolutional layers on the test images on new CNN network	
Convolutional Layer 1	Convolutional Layer 2
Conv1: Max of activation after ReLU 	Conv2: Max of activation after ReLU 
Conv1: Min of activation after ReLU 	Conv2: Min of activation after ReLU 
Conv1: Mean of activation after ReLU 	Conv1: Mean of activation after ReLU 
Conv1: Std of activation after ReLU 	Conv1: Std of activation after ReLU 
Conv1: Histogram of activation after ReLU 	Conv1: Histogram of activation after ReLU 

2 Summary: Visualizing and Understanding Convolutional Networks

Author of this paper introduces a visualization technique, using a Deconvolutional Network, which gives insight of large Conv-Net models. Visualization helps to understand the intermediate features layers and their contribution to the model performance. Author uses this visualization technique to experiment the ImageNet model on other dataset and found the ImageNet model generalizes well on other datasets.

With a deconvnet, author shows a way to map the feature activities of a convnet back to the input pixel space, showing what input pattern originally caused a given activation in the feature maps. A deconvnet is attached to each of the convnet layer, provide a continuous push back to image pixels by successively and repeatedly applying unpool, rectify and filter to reconstruct the activity in the layer which influenced the chosen activation. During training convnet model, the visualization of the first layer filter helps to see that a few of them dominate.

Visualization of the network helps to ravel the different structures of the activation that excites a given feature map by showing it's invariance to the input deformations and show progress of the strongest activation within a given feature map projected back to pixel space. A small transformation of image can have dramatic impact at the first layer of the model but a lesser impact at the top feature layer. In addition, visualization can help to select right network architecture.

The last part of the paper was about experiments on ImageNet dataset and how visualization helps to see the importance of convolutional part of the ImageNet model in obtaining state-of-the-art performance and the ability of feature extraction layers to generalize ImageNet to other datasets.

The novel way of visualizing the model activity reveals that the features in the model are not random, uninterruptible patterns, rather they show many intuitively desirable properties such as compositionality, increasing invariance and class discrimination with ascending layers. Finally, visualization could be a way to debug the model to find better model architecture and achieve better result.

3 Training an RNN on MNIST Dataset:

Even though RNN famous for performance on sequential and time-series data, in this part of the problem we used RNN on image dataset(MNIST) for object recognition.

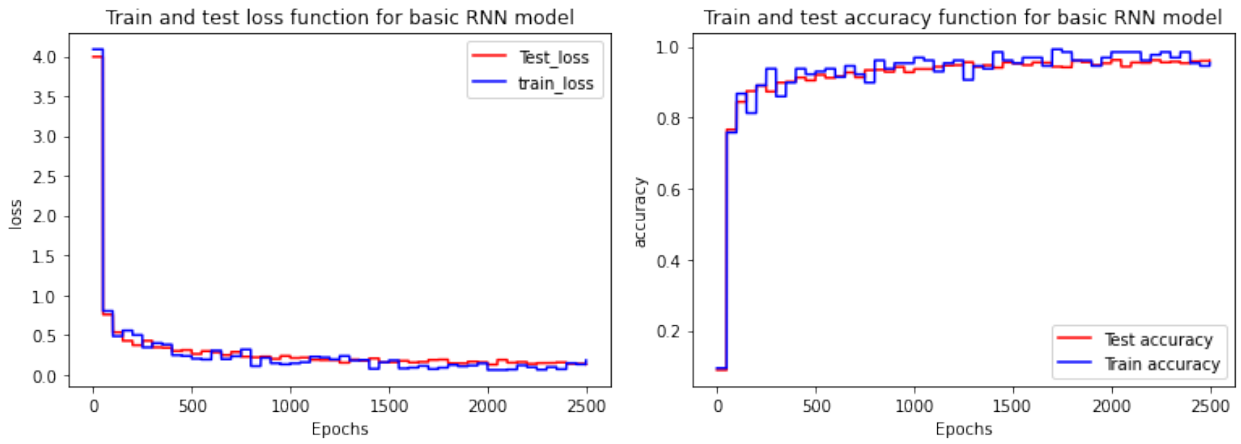


Figure 5: Training and test loss and accuracy curve of Grayscale MNIST dataset for Basic RNN architecture. Both loss and accuracy was recorded on every 50 iterations.

RNN: The inputs to the RNN network are 28 pixel chunks of the images, i.e each row of the images will be sent to the network at each iteration. The RNN network has 128 neurons, learning rate 0.001 and we trained the network for 5500 times. Figure 5 shows the loss and accuracy on both training and test data.

The model achieved 97.27% accuracy on test data and 97.65% accuracy on training data at iteration number 5500.

LSTM: Figure 6 shows the loss and accuracy of the LSTM network on test and training data. Keeping the learning rate and hidden neurons are same as RNN (0.001 and 128 respectively), Figure 6 shows LSTM network performs better than basic RNN network. The model achieved 98.7% test accuracy and 99.99% on training accuracy around iteration number 3900, faster and better performance than the basic RNN network.

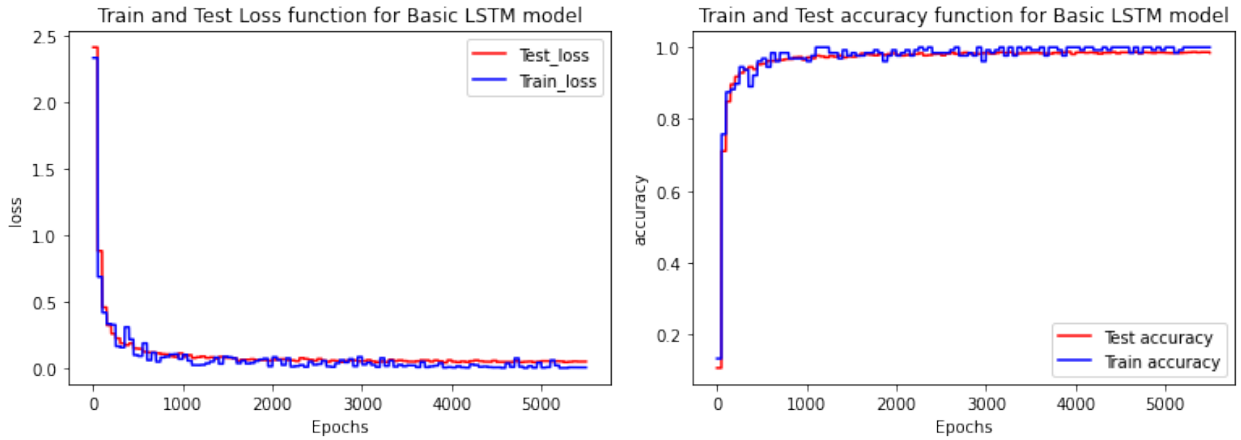


Figure 6: Training and test loss and accuracy curve of Grayscale MNIST dataset for LSTM architecture. Both loss and accuracy was recorded on every 50 iterations.

GRU: The learning rate and the number of neurons are 0.001 and 128 respectively, which is same as basic RNN network and we trained the GRU network for same number of iteration 5500. With GRU model we achieved 99.21% training and 99.8% test accuracy around iteration number 3250, which is better and faster than both RNN and LSTM. Figure 7 shows the behaviour of the test and train loss and accuracy curve.

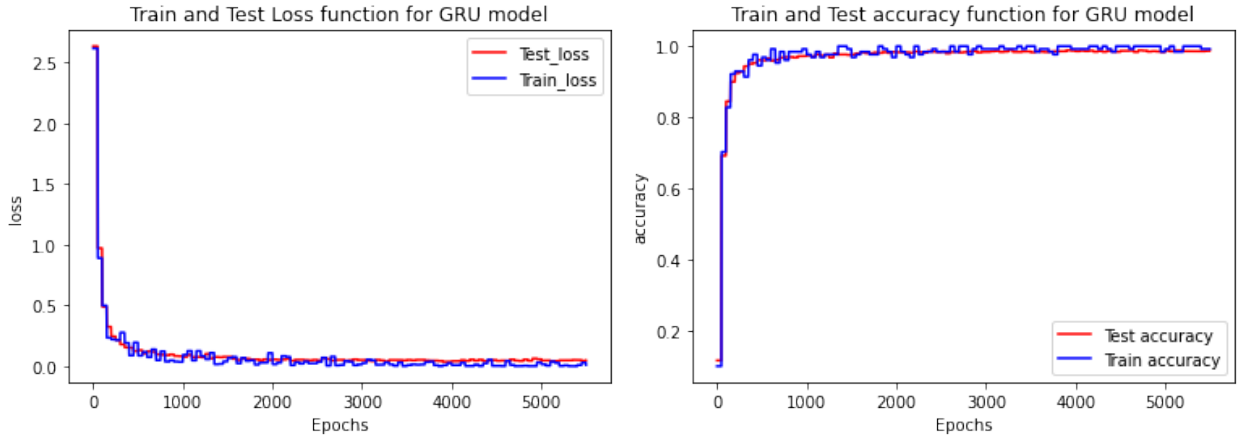


Figure 7: Training and test loss and accuracy curve of Grayscale MNIST dataset for GRU architecture. Both loss and accuracy was recorded on every 50 iterations.

RNN With Different Number of Hidden Units: RNN with 64 hidden units gives 96.38% test and 99.21% training accuracy and with 32 hidden units network gives 98% test and 99.2% training accuracy and both network take larger number of training iteration that the network with 128 hidden units. Figure 8 shows the loss and accuracy function for different hidden units.

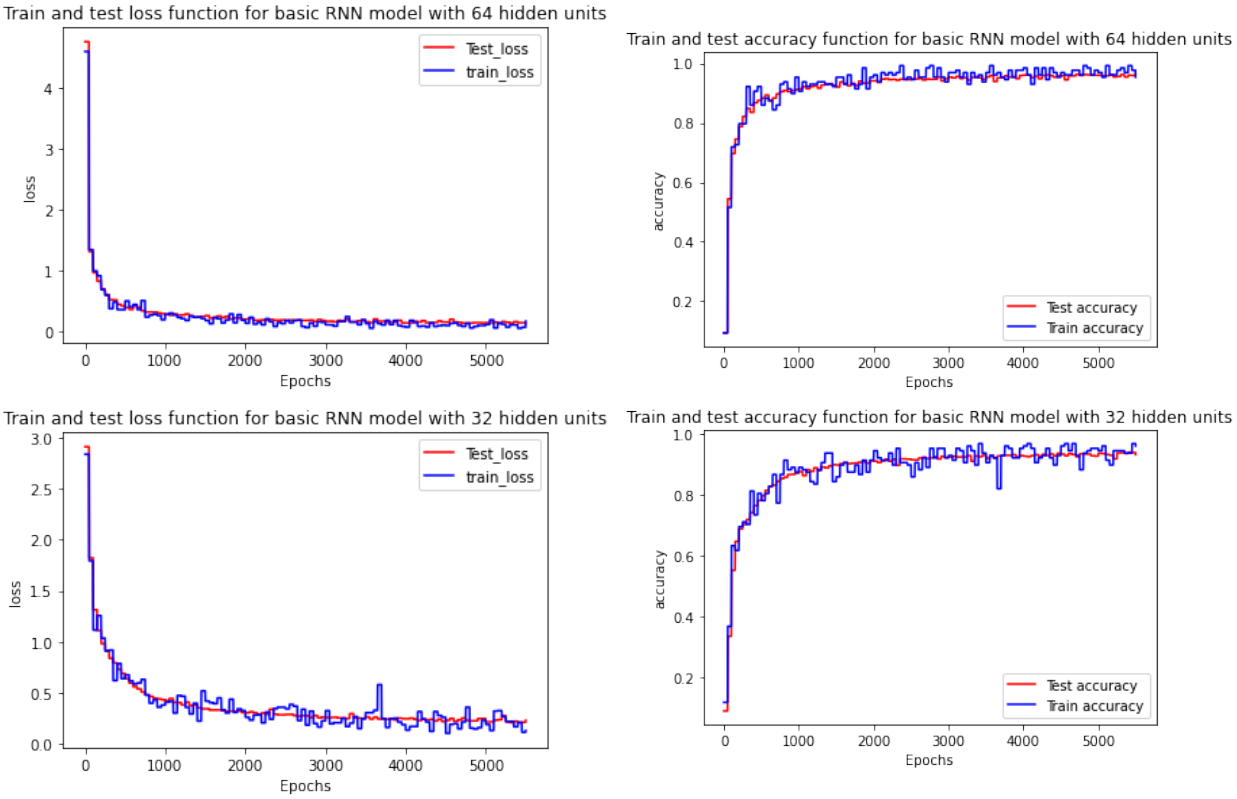


Figure 8: Training and test loss and accuracy on MNIST dataset for RNN architecture with 64 and 32 hidden units. Both loss and accuracy was recorded on every 50 iterations.

LSTM with 64 hidden units gives 98.2% test and 98.4% training accuracy and with 32 hidden units network gives 97.9% test and 99.2% training accuracy and both network take larger number of training iteration that the network with 128 hidden units. Figure 9 shows the loss and accuracy function for different hidden units.

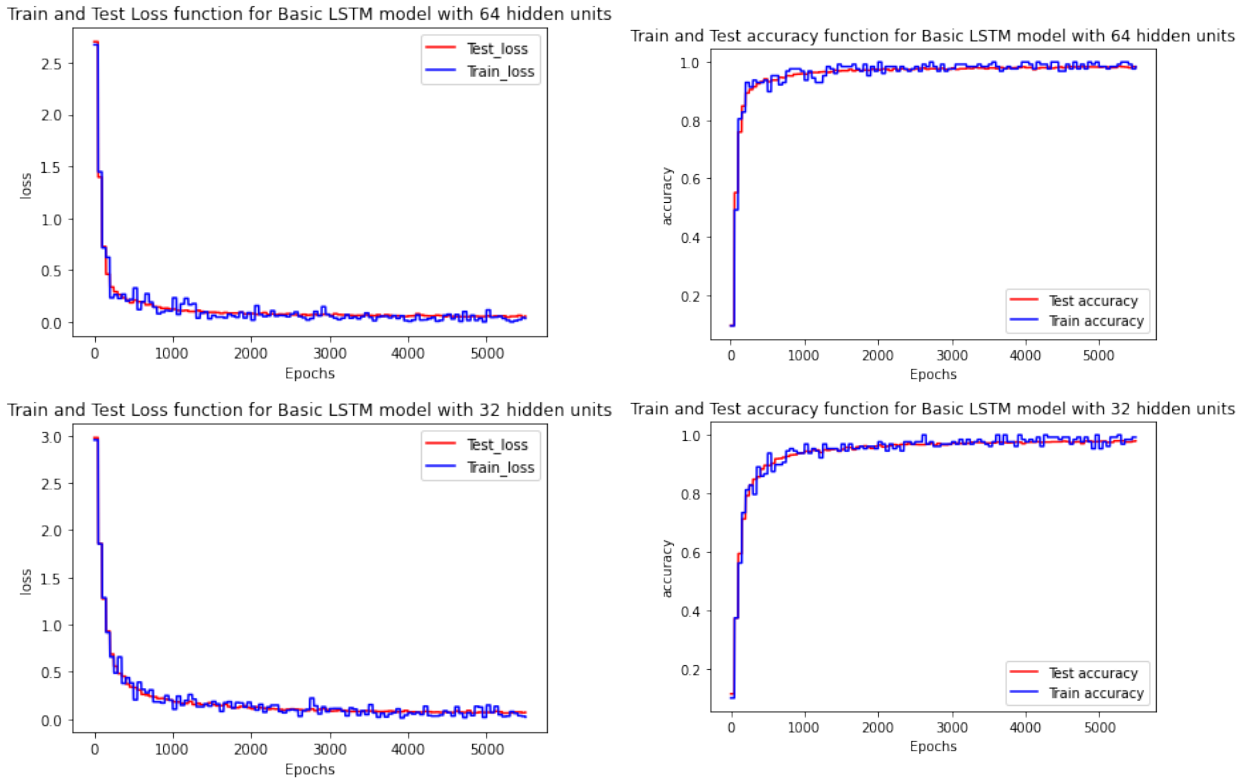


Figure 9: Training and test loss and accuracy on MNIST dataset for LSTM architecture with 64 and 32 hidden units. Both loss and accuracy was recorded on every 50 iterations.

GRU with 64 hidden units gives 98.6% test and 99.99% training accuracy and with 32 hidden units network gives 98% test and 99.2% training accuracy and both network take larger number of training iteration that the network with 128 hidden units. Figure 10 shows the loss and accuracy function for different hidden units for GUR network.

CNN Network Performance on MNIST Dataset: CNN network has same learning rate (0.001), step-size(128), optimizer (Adam) and iteration number (5500) as RNN. Figure 11 shows the loss and accuracy of test and train data on MNIST dataset. CNN network has 99.08% test and 99.99% training accuracy on MNIST dataset. The convergence rate on MNIST dataset with CNN architecture is much faster than RNN network. RNN network achieves test accuracy 97.27% after 5500 iterations, whereas CNN network passes this accuracy benchmark after around 1000 iterations of training. Overall, CNN network performs better than RNN on MNIST dataset.

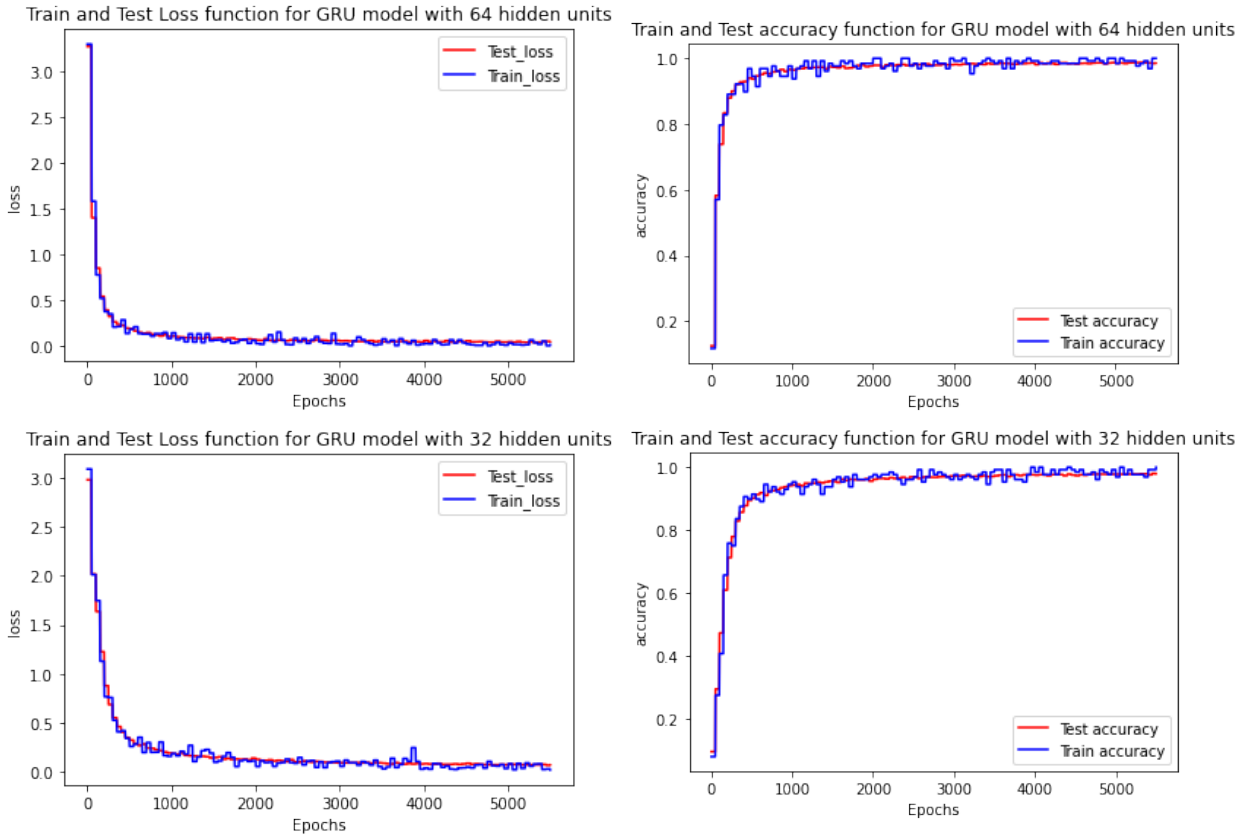


Figure 10: Training and test loss and accuracy on MNIST dataset for GRU architecture with 64 and 32 hidden units. Both loss and accuracy was recorded on every 50 iterations.

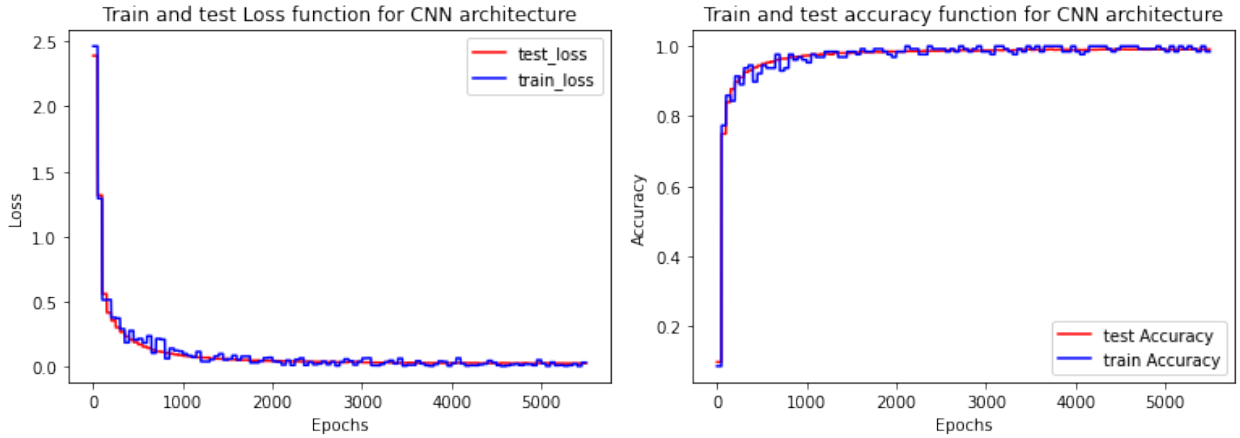


Figure 11: Training and test loss and accuracy curve of Grayscale MNIST dataset for Deep CNN architecture. Both loss and accuracy was recorded on every 50 iterations.

3.1 Resources I used to work on this assignment:

1. Introduction to Deep Learning Lectures
2. Building RNN

3. Convolutional Neural Networks for Visual Recognition
4. Recurrent Neural Networks
5. Convolutional Neural Networks
6. Collaborative Filtering, Embeddings, and more.
7. stackOverflow