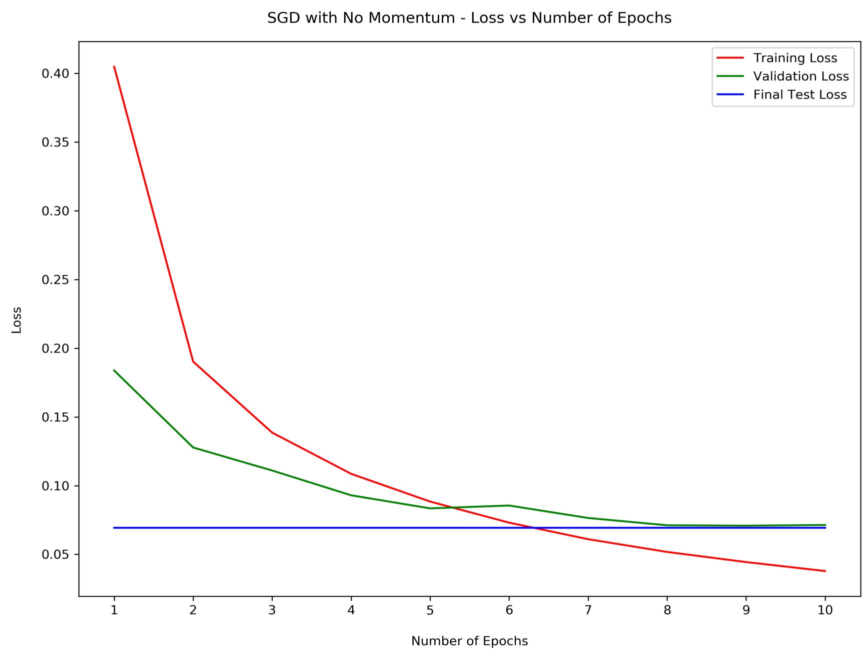
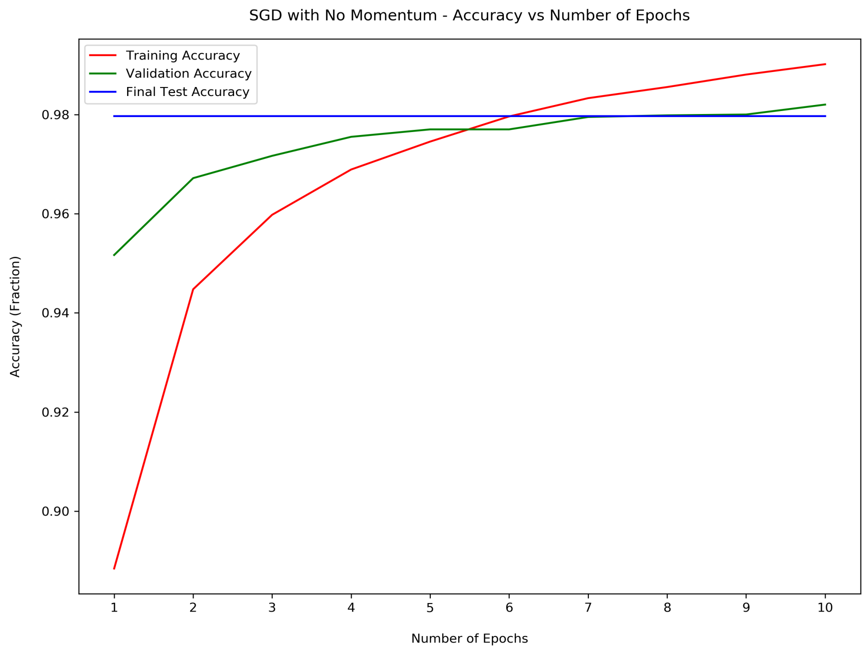
CS 4787, Spring 2021 **Report - Assignment 4**
Name: Aliva Das
NetID: ad677

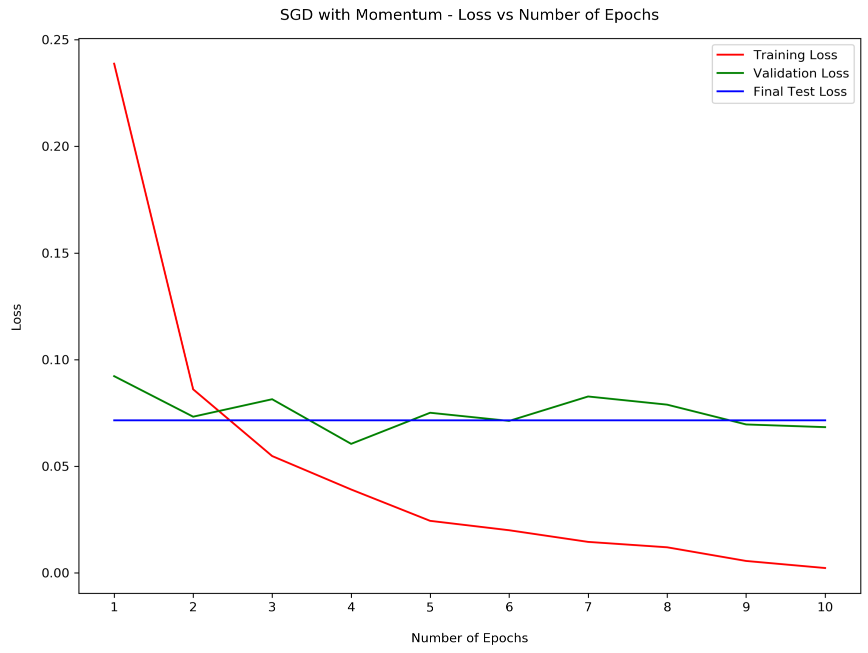
Part 1: Fully Connected Neural Network on MNIST



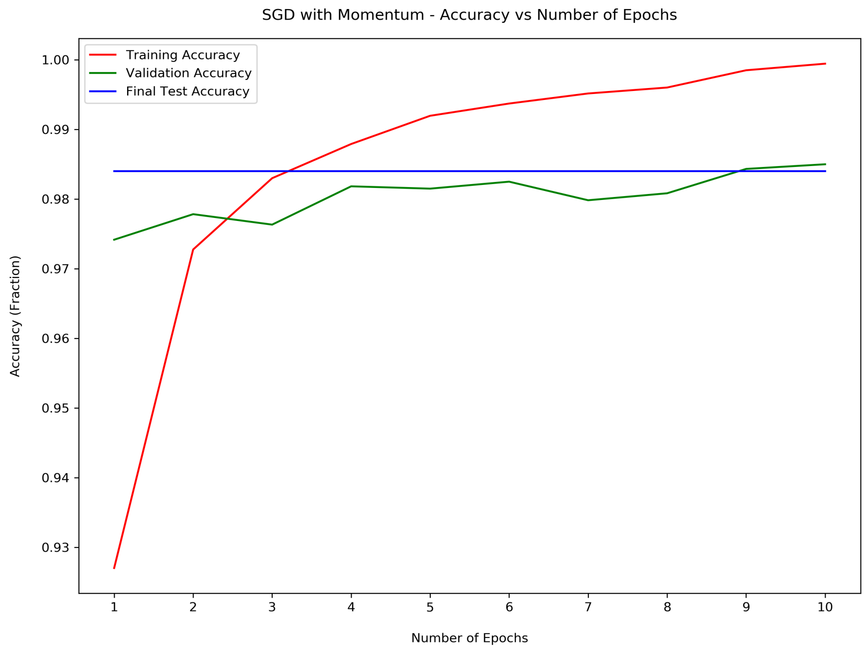
Graph 1 - SGD with No Momentum - Loss vs Number of Epochs



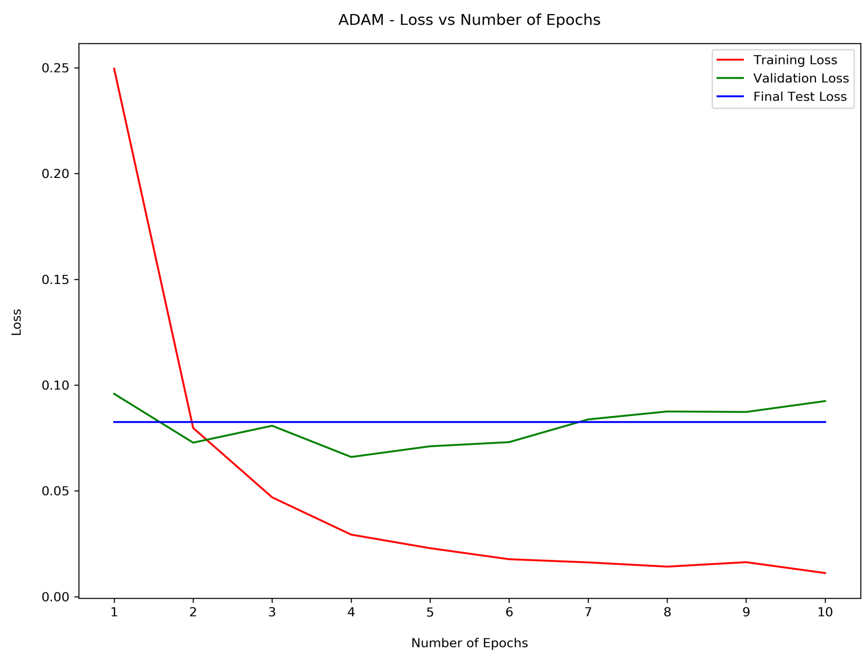
Graph 2 - SGD with No Momentum - Accuracy vs Number of Epochs



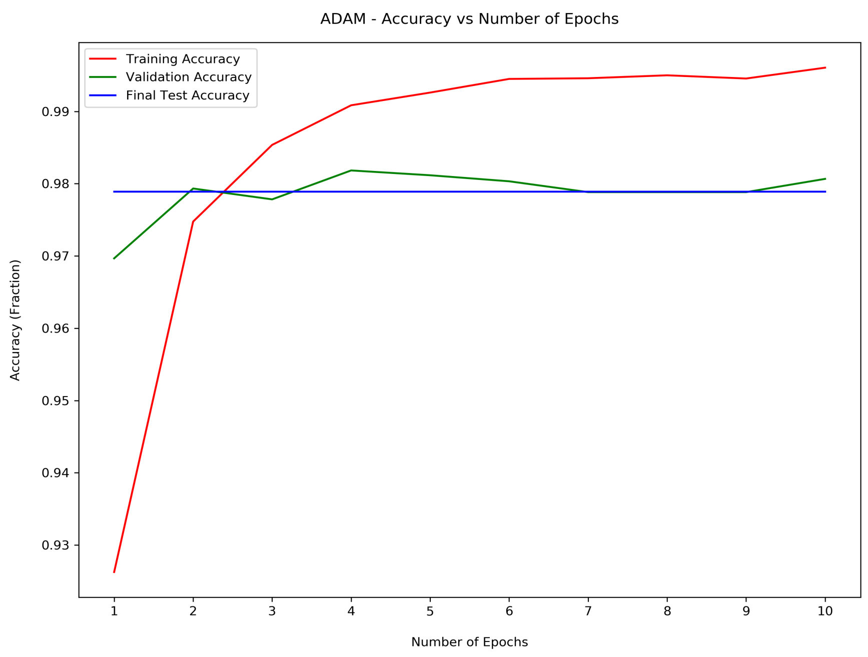
Graph 3 - SGD with Momentum - Loss vs Number of Epochs



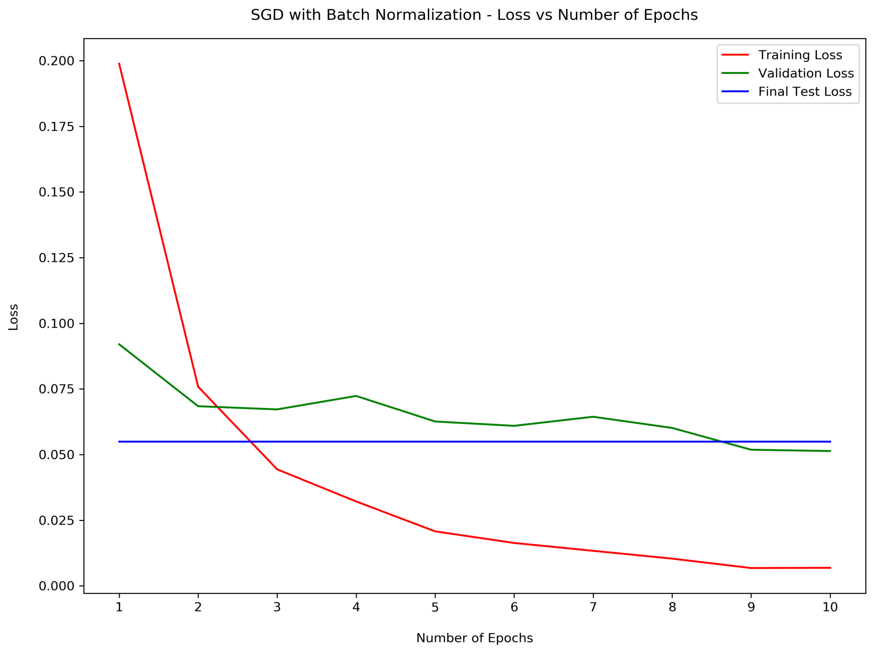
Graph 4 - SGD with Momentum - Accuracy vs Number of Epochs



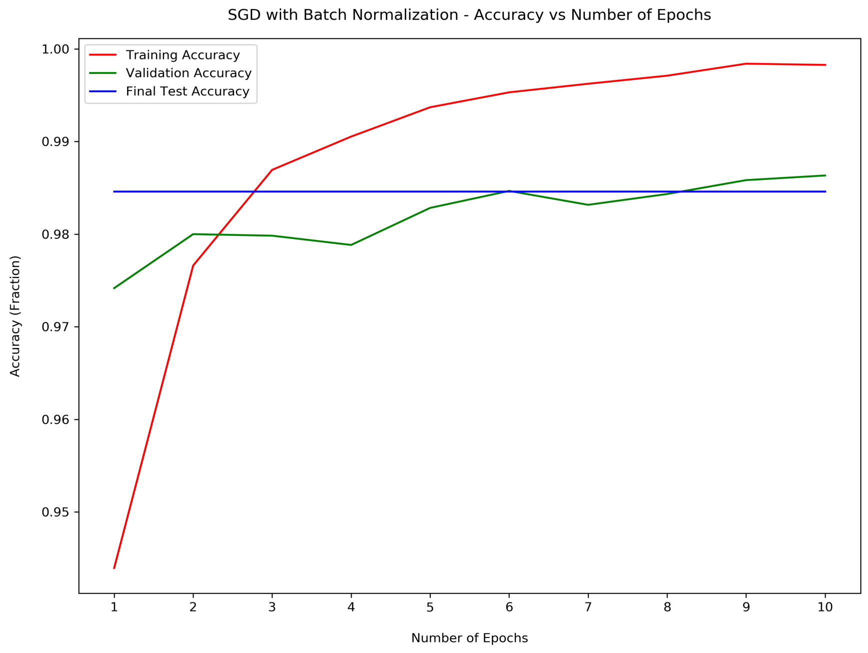
Graph 5 - ADAM - Loss vs Number of Epochs



Graph 6 - ADAM - Accuracy vs Number of Epochs



Graph 7 - SGD with Batch Normalization - Loss vs Number of Epochs



Graph 8 - SGD with Batch Normalization - Accuracy vs Number of Epochs

Model	Running Time (sec)
SGD with No Momentum	53.12
SGD with Momentum	62.89
ADAM	108.17
SGD with Batch Normalization	139.81

Table 1: Runtime for 10 epochs of Different Models

Model	Final Training Loss	Final Training Accuracy	Final Validation Loss	Final Validation Accuracy	Final Testing Loss	Final Testing Accuracy
SGD with No Momentum	0.03786	0.99014	0.07136	0.982	0.06922	0.9797
SGD with Momentum	0.00225	0.99944	0.06924	0.98433	0.07154	0.984
ADAM	0.01114	0.99605	0.09243	0.98066	0.08248	0.9789
SGD with Batch Normalization	0.00686	0.99827	0.05133	0.98633	0.05491	0.9846

Table 2: Metrics at the end of 10 epochs for Different Models

Comparing Algorithm Performance:
Accuracy:
Among the four algorithms, SGD with Batch Normalization had the highest final validation accuracy of 0.98633 and highest final testing accuracy of 0.9846, followed by SGD with Momentum (final validation accuracy 0.98433, final testing accuracy 0.984), then SGD with No Momentum (final validation accuracy 0.982, final testing accuracy 0.9797) and finally ADAM (final validation accuracy 0.98066, final testing accuracy 0.9789).
Convergence:
Among the four algorithms, ADAM converges the fastest to the final training loss (by around 6 epochs), followed by Batch Normalization (by around 9 epochs), followed by SGD with Momentum and SGD with No Momentum.

Part 2: Hyperparameter Optimization

α	Validation Loss	Validation Accuracy
1.0	2.31734	0.105
0.3	0.15233	0.972
0.1	0.06924	0.98433
0.03	0.05851	0.98516
0.01	0.07019	0.979
0.003	0.10179	0.974
0.001	0.17881	0.95283

Table 3: Performance Comparison on Validation Set using Different Values of Learning Rate α for SGD with Momentum

Yes, the step size found by grid search (α = 0.03) is smaller and improves over the step size (α = 0.1) given in the instructions in Part 1.

Grid Search was done using the following hyperparameter values on the SGD with Momentum Model:
- Learning Rate $\alpha \in [0.01, 0.1, 1.0]$
- Momentum Coefficient $\beta \in [0.1, 0.5, 0.9]$
- Number of Units in Second Dense Layer $d_2 \in [128, 256, 512]$

I chose 3 values for the above hyperparameters to limit the size of the grid to be 3×3×3. The SGD with Momentum Model from Part 1 had α = 0.1, β = 0.9 and d_2 = 256. As α could have any value greater than 0, I chose 2 other values, 10 times smaller (0.01) and 10 times larger (1.0) than α in Part 1 to determine the best order of magnitude for α. Momentum Coefficient β takes any value in [0, 1], so I chose 3 values representing parts of this interval - 0.1 (lower end), 0.5 (middle), 0.9 (upper end). The number of units in second dense Layer d_2 usually takes values in powers of 2 (as seen in the literature), so I chose 2 other values, 2 times smaller (128) and 2 times larger (512) than d_2 in Part 1 to better explore the effect of changing d_2 on model perfomance.

α	β	d_2	Final Validation Accuracy	Final Validation Loss	Final Testing Accuracy	Final Testing Loss
0.1	0.9	512	0.98466	0.06633	0.9849	0.06027

Table 4: Best Hyperparameters from Grid Search on Hyperparameters α, β and d_2 of the SGD with Momentum Model

Random Search was done using the following hyperparameter distributions on the SGD with Momentum Model:
- Learning Rate $\alpha \sim LogUniform(0.01, 1.0)$ [continuous]
- Momentum Coefficient $\beta \sim Uniform(0, 1)$ [continuous]
- Number of Units in Second Dense Layer $d_2 \sim Uniform(128, 512)$ [discrete]

As Learning Rate α takes a wide range of values, from 0.01 to 1.0, but not much is known about the underlying distribution of α, the LogUniform continuous distribution between 0.01 and 1.0 is a good approximation for α's distribution. As Momentum Coefficient β takes any value in [0, 1] with equal probability, I chose a uniform continuous distribution between 0 and 1 to represent β's distribution. As Number of Units in Second Dense Layer d_2 can take any integral value in [128, 512] (corresponding to the d_2 space covered by Grid Search) with equal probability, I chose a uniform discrete distribution between 128 and 512 to represent d_2's distribution.

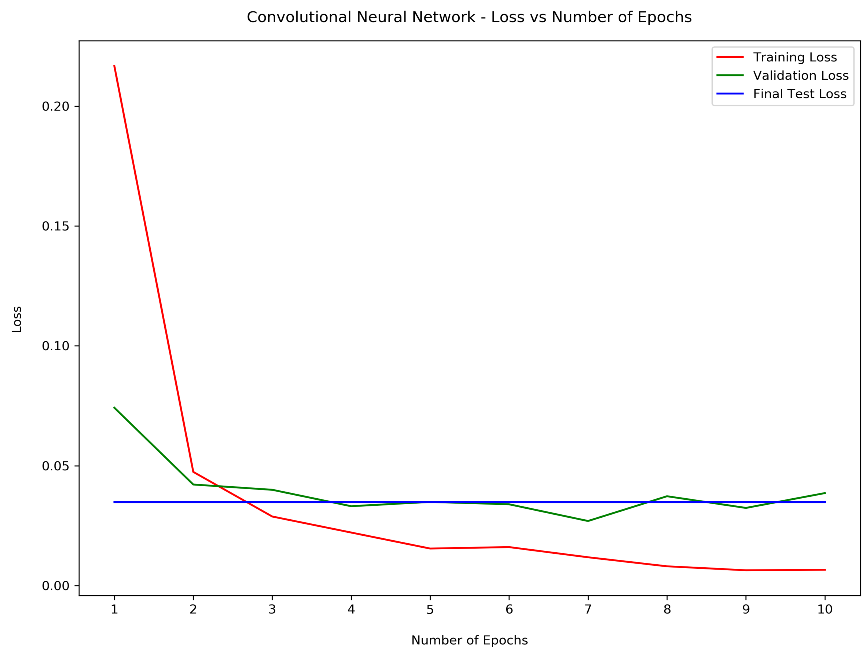
α	β	d_2	Final Validation Accuracy	Final Validation Loss	Final Testing Accuracy	Final Testing Loss
0.26737	0.61457	302	0.98683	0.06694	0.9864	0.05659

Table 5: Best Hyperparameters from Random Search on Hyperparameters α, β and d_2 of the SGD with Momentum Model

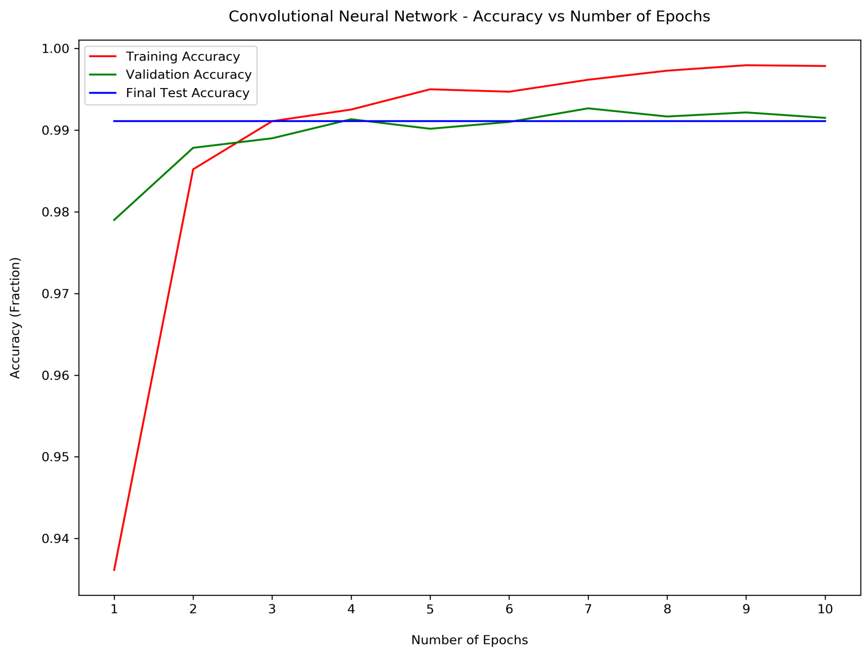
Comparing Grid Search and Random Search:
Random Search resulted in a model with better final validation accuracy 0.98633 and better final testing accuracy 0.9864 as compared to Grid Search (final validation accuracy 0.98466, final testing accuracy 0.9849).

Grid Search over the above hyperparameter combinations took 1901.67 seconds.
Random Search for 10 trials over the above hyperparameter distributions took 726.53 seconds.
The time taken by Grid Search was approximately 1901.67/726.53 = 2.62 times more than that taken by Random Search to cover the same distribution.

Part 3: Convolutional Neural Networks



Graph 9 - Convolutional Neural Network - Loss vs Number of Epochs



Graph 10 - Convolutional Neural Network - Accuracy vs Number of Epochs

Model	Final Training Loss	Final Training Accuracy	Final Validation Loss	Final Validation Accuracy	Final Testing Loss	Final Testing Accuracy	Runtime (sec)
Convolutional Neural Network	0.00657	0.99785	0.03857	0.9915	0.03475	0.9911	355.63

Table 6: Metrics at the end of 10 epochs for Convolutional Neural Network

Comparing Algorithm Performance between Part 1 and Part 3:
The Convolutional Neural Network had higher final validation accuracy of 0.9915 and highest final testing accuracy of 0.9911, as compared to the best performing algorithm in Part 1, which was SGD with Batch Normalization having final validation accuracy of 0.98633 and final testing accuracy of 0.9846. From the training loss graphs, we see that the Convolutional Neural Network converges faster than any of the algorithms in Part 1.

Summary

In this assignment, I explored training a neural network on the MNIST dataset using Tensorflow. I compared the performance of a traditional neural network with 2 linear dense layers with ReLU activation, followed by another linear layer with output size equal to the number of classes in the dataset and softmax activation, with a Convolutional Neural Network.

For the traditional neural network, I experimented with Stochastic Gradient Descent as an optimizer with and without momentum, as well as Batch Normalization. I also used ADAM as an optimizer for the traditional neural network. The experiments with adding momentum, batch normalization and using ADAM was to see if the model could be made to converge faster, which it did for each of the techniques mentioned., with ADAM converging the fastest. However, these methods for faster convergence take more time to run as compared to normal SGD. Out of all the configurations for the traditional neural network, Batch Normalization had the highest validation and test accuracy, but was outperformed by the Convolutional Neural Network.

I also experimented with the values of hyperparameters to check if I could get better results than the baseline. I used Grid Search and Random Search, both covering the same search space for a SGD with Momentum model. In Grid Search, as all possible combinations of hyperparameter values are taken, the number of such values that can be tested is limited, as the number of evaluations required increases exponentially with each added value. On the other hand, Random Search simulates the distribution of each hyperparameter, and chooses random combinations to evaluate. Thus Random Search is shown to have equal, sometimes even better hyperparameter results than Grid Search, in lesser time.