Report for Assignment 3

Course Code: COMP 8610
Course Title: Neural Networks and Deep Learning
Submit to: Dr. Alioune Ngom

Submitted by: Siyam Sajnan Chowdhury (110124636)
Kasra Mojallal (110124782)
Kimia Tahayori (110124141)

All of the group members contributed equally to this assignment.

Question 1:

Classifying 1 from not-1 from the MNIST dataset using Shallow Neural Network

We implemented a shallow neural network using tensorflow on the MNIST dataset in order to classify the 1s in the dataset from all the other images that did not contain 1s. This was done as a precursor to implementing the Deep Neural Network.

We imported the MNIST dataset from the tensorflow keras stored datasets. We then reshaped the 28x28 images to the dimension 784x1 and normalized the 255 different possible greyscale intensities by dividing it by 255. We took the

first 5000 data as the validation set and the next 55,000 data are used as the training data. It has 10,000 data as the test set.
Next, we created 3 new lists of train, validation and test where it was a list of Boolean where it was true where it found a 1.

We then created the single layer model with the input layer's activation function was relu which took in 10 values and the output layer's activation function was set to sigmoid.

Since it was a binary classification, we set the loss function to binary cross entropy and sgd as the learning algorithm. We then fit the data with batch sizes of 32 over 10 epochs. We then plotted using plotly.

The loss that we obtained over the epochs is shown in figure 1:

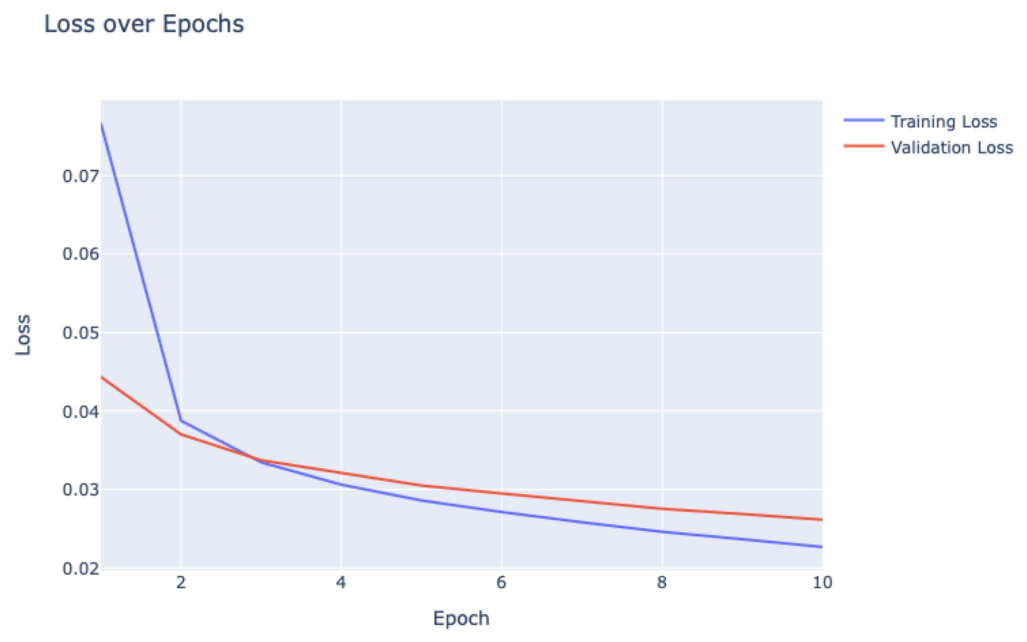

Figure 1: Loss over Epoch for Shallow Neural Network

The accuracy improving over the 10 epochs is illustrated in figure 2:

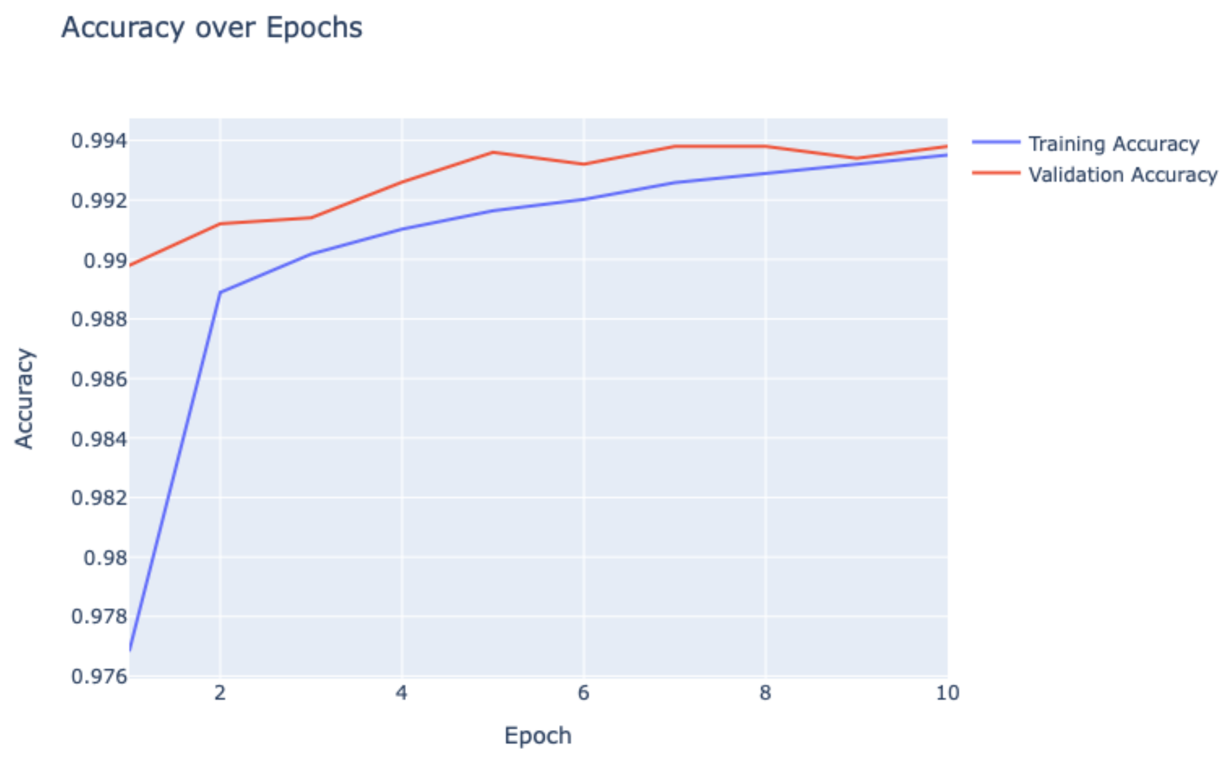

Figure 2: Accuracy over Epoch for Shallow Neural Network

The final accuracy is 99.5% and the final loss is 0.0189.

Multiclass classification from the MNIST dataset using Deep Neural Network with Minibatch SGD.

We used a train and validation split of 80%.

We defined a Deep NN with 4 layers. The input layer is the flatten layer which flattens the 28x28 input to a 1d vector. The second layer is a fully connected layer with 256 units and the ReLU as the activation function. It takes the flattened input and applies a linear transformation followed by the activation function. The third layer is another fully connected layer with 128 units and the ReLU activation function. It takes the output from the previous layer and performs another linear transformation followed by the activation function.

The final layer is the output layer with 10 units, corresponding to the 10 possible classes in the problem. It uses the softmax activation function to generate probabilities for each class.

We then compiled the model with stochastic gradient descent (SGD) as the optimizer, sparse categorical cross-entropy as the loss function for multi-class classification, and accuracy as the metric to monitor during training. This configuration enables the model to be trained using SGD to minimize the cross-entropy loss and evaluate its performance based on accuracy.

We then fit the data and stored in in a variable history in order to access the different metrics saved in the history object such as loss, accuracy etc. over the different epochs.

We then plotted the graph of loss over epochs and accuracy over epochs using plotly in Figure 3 and 4.

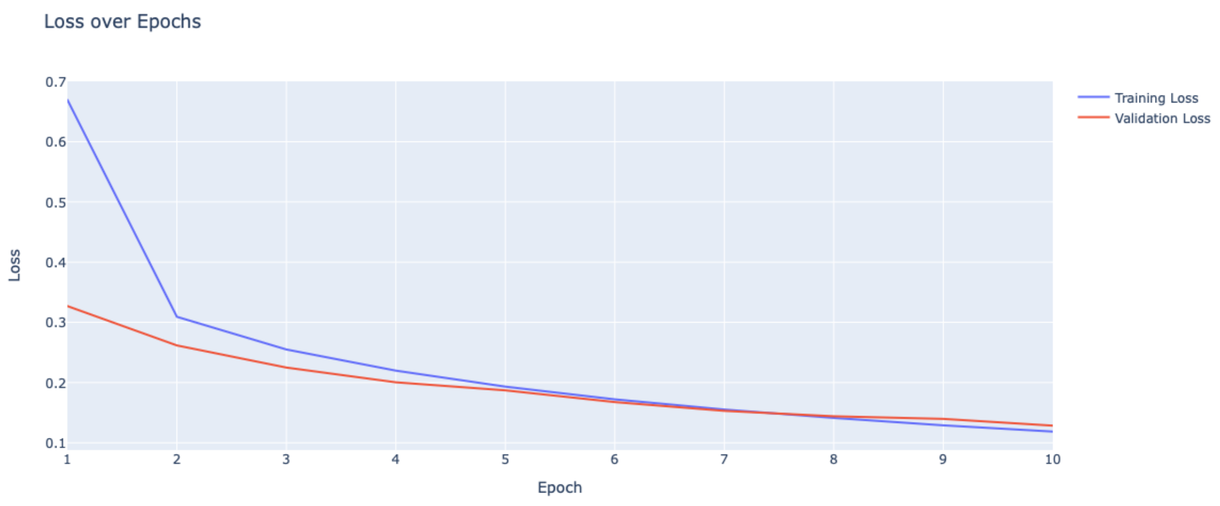

Figure 3: Loss over Epoch for Deep Neural Network using SGD

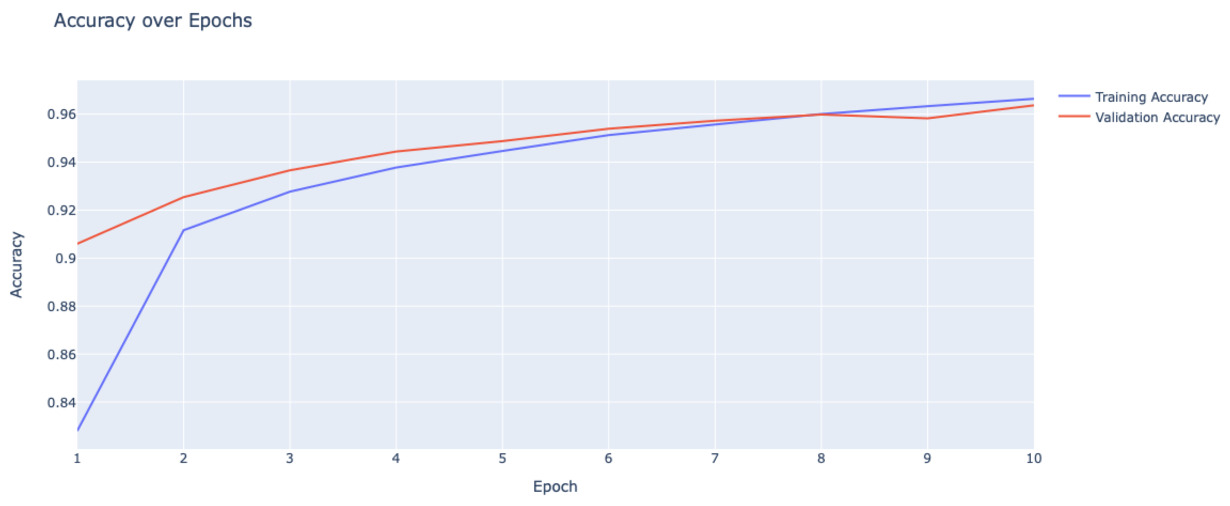

Figure 4: Accuracy over Epoch for Deep Neural Network using SGD

We attained an accuracy of 96.32% and a loss of 0.126.

Multiclass classification from the MNIST dataset using Deep Neural Network with Adam.

We used a very similar method for this as the previous one – the same splits, the same Neural Network setup. While compiling the model, we used Adam as the optimizer instead of SGD keeping the loss function the same – sparse categorical cross entropy. We then fit the data over 10 epochs with the default batch size.

The results are illustrated in figure 5 and 6.

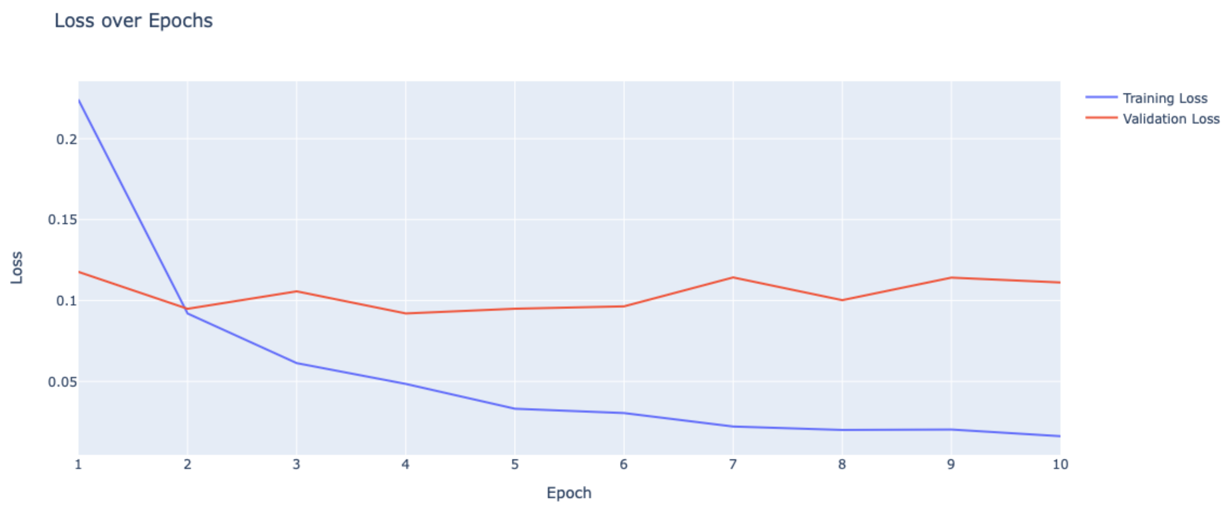

Figure 5: Loss over Epoch for Deep Neural Network using Adam

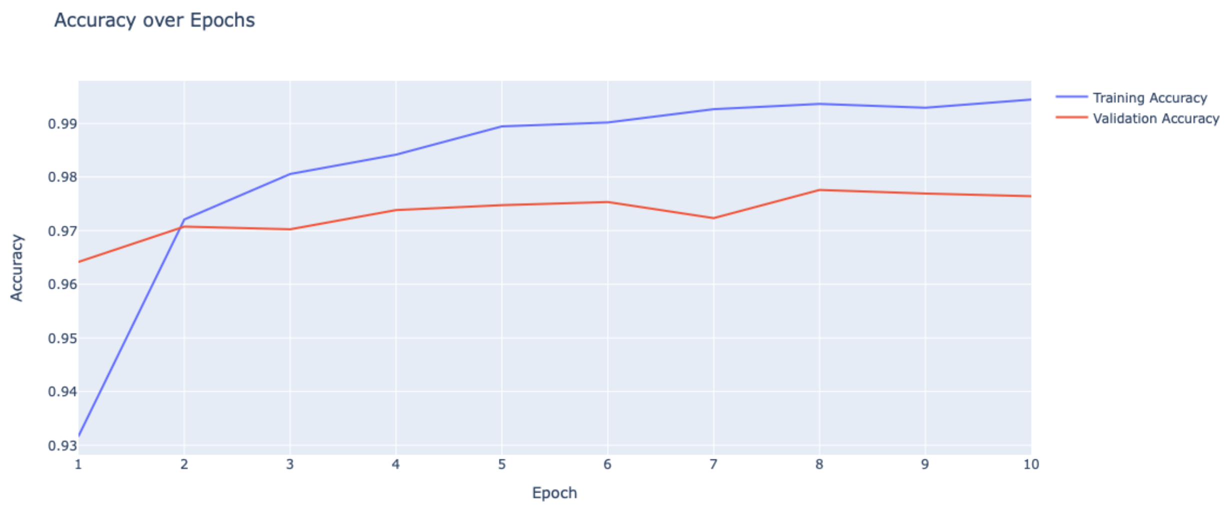

Figure 6: Accuracy over Epoch for Deep Neural Network using Adam

We got a 97.8% accuracy and a loss of 0.09. It performed slightly better than the SGD which could be a result of its adaptive learning rate and the update of weights using momentum thus leading to better generalized performance.

We used GridSearchCV in order to find out the optimal parameters. We had to use a wrapper because our model was a custom one and it does not directly fit in GridSearchCV. We tested out 2 learning rates: 0.01 and 0.1 and tested out 3 batch sizes of 16, 32 and 64.

The 6 combinations are set out in Figure 7.

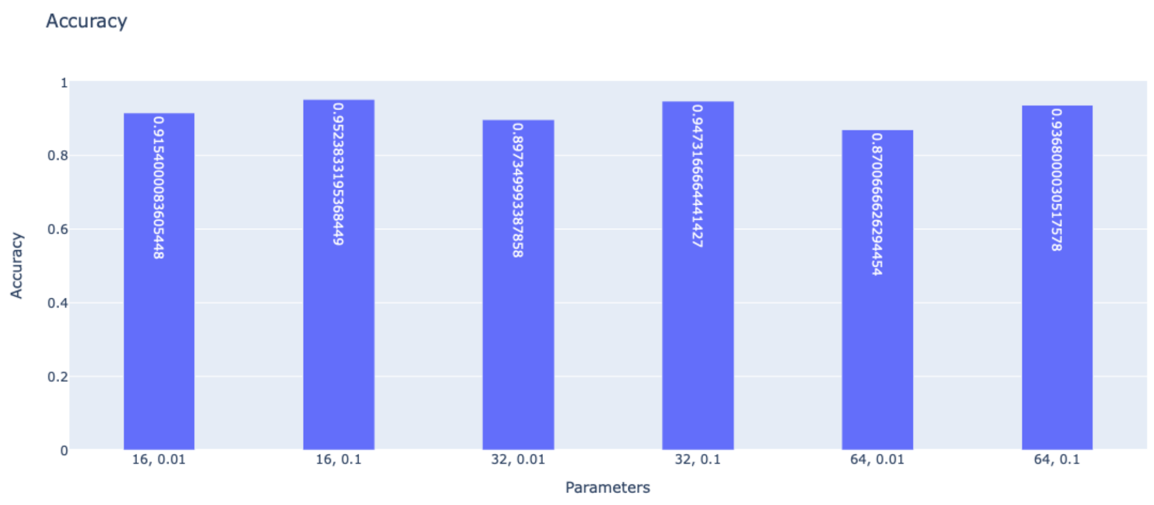

Figure 7: Accuracy varying over the different parameters

The differences were somewhat minimal but the best performing parameter combination was a learning rate of 0.1 over a batch size 16.

<u>Question 2</u>

Multiclass classification from the MNIST dataset using Deep Neural Network with Minibatch SGD with L^2-regualrization.

We utilized a similar test train split to question 1 and reshaped the x values to a 1d vector of length 784.

We defined the model with three hidden layers and one output layer. The input layer is a Flatten layer that transforms the 2D input shape of 28x28 into a 1D vector of size 784. The two hidden layers have 128 and 64 units, respectively, with the ReLU activation function and L2 regularization with a parameter of 0.01 to prevent overfitting. The output layer has 10 units with the softmax activation function, suitable for multi-class classification tasks. This architecture could classify the data into any of the ten output classes with the regularization helping to prevent overfitting and improve generalization where it would perform well with different other types of data.

We then used SGD as the optimization algorithm and sparse categorical cross entropy as the loss function. We then fit the model using a batch size of 32 over 10 epochs saving it in a history_regularized_model variable in order to access the metrics from the history object. The obtained loss and accuracy over different epochs are illustrated in figure 8 and 9.

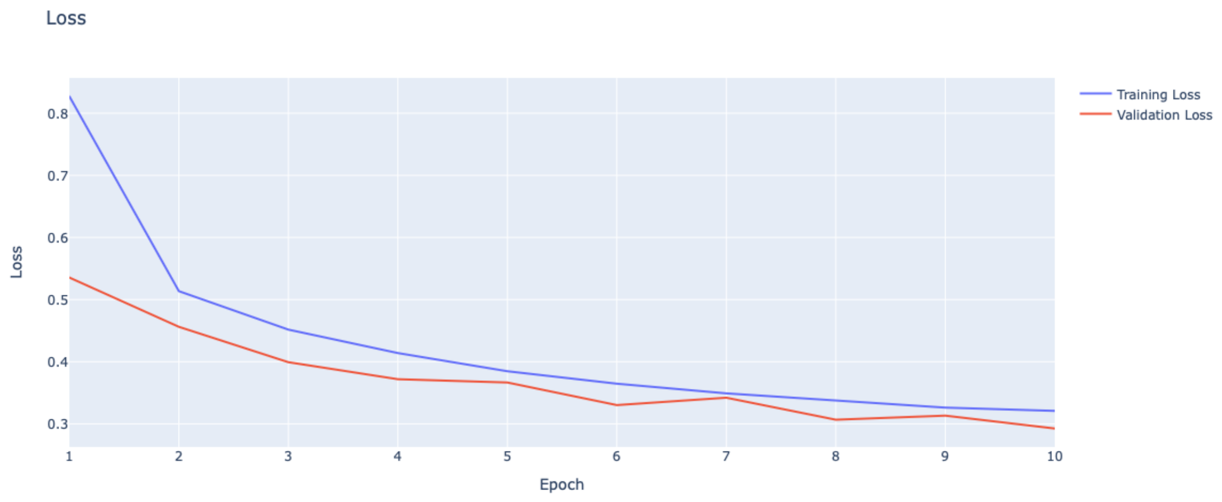

Figure 8: Loss over Epoch for Deep Neural Network using L2 Regularization

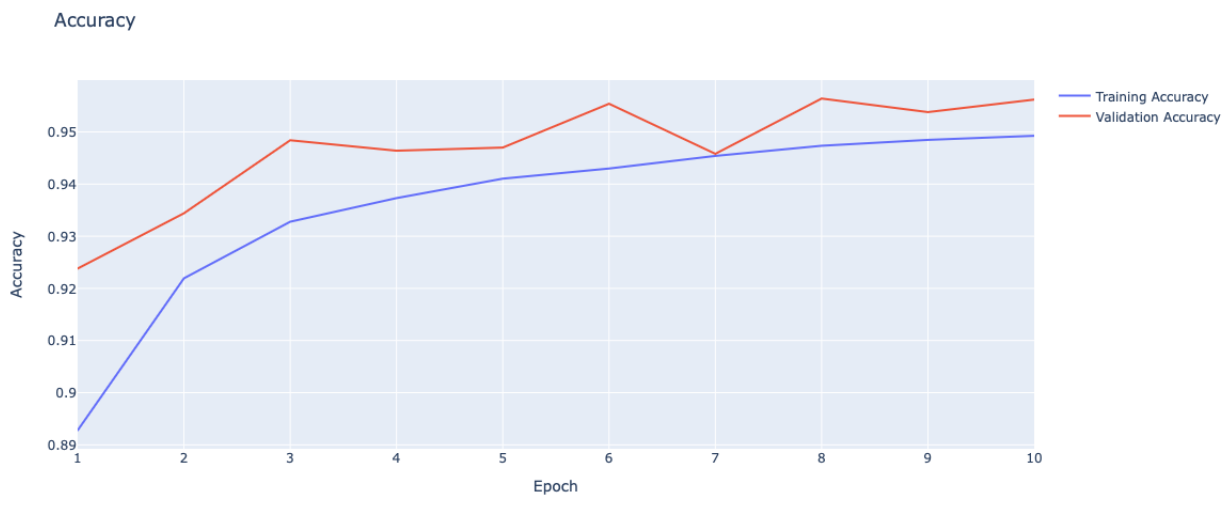

Figure 9: Accuracy over Epoch for Deep Neural Network using L2 Regularization

The accuracy came out to 94% with the loss being 0.4099. It is slightly higher compared to the non-regularized SGD due to the regularization penalty. Although the loss has slightly increased, this means that this data is better suited for a more generalized dataset.

<u>Question 3</u>

Building a 3-layer feedforward neural network for MNIST

We utilized a similar test train split to question 1 and reshaped the x values to a 1d vector of length 784. We converted the y values to categorical data.

We then defined the architecture of the model with the input layer being a fully connected layer having 500 units and ReLU as the activation function. The hidden layer is also the same as the input layer and the final layer classifying the input from the hidden layer to one of the 10 outputs using softmax as the activation function.

We compiled the model using adam and categorical cross entropy as the loss function. We then fit the model over 250 epochs with batch sizes of 128. The cross entropy loss over the epochs and the classification error on the validation data is illustrated in figure 10 and 11.

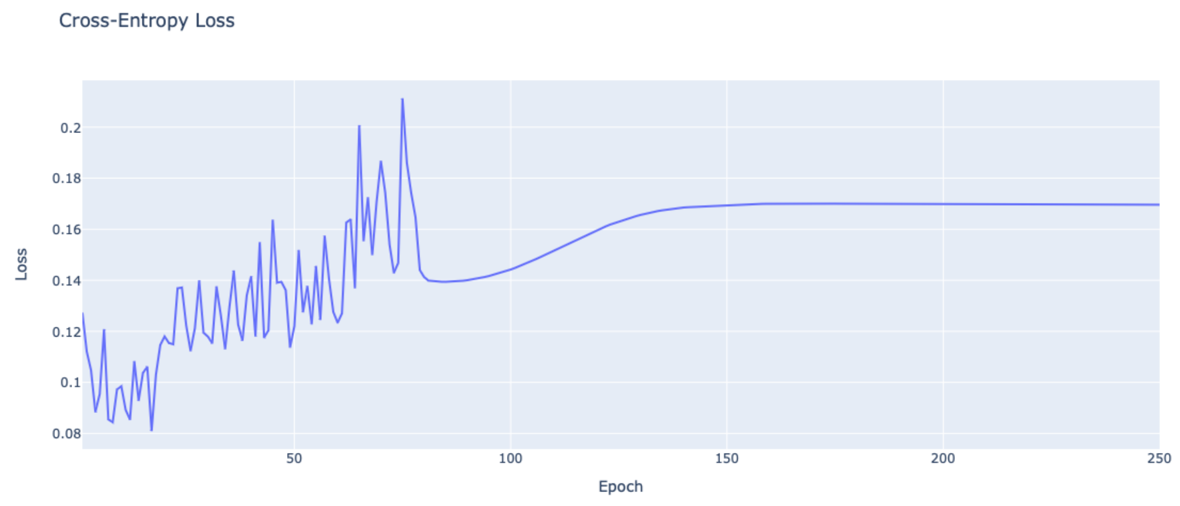

Figure 10: Cross-Entropy loss over Epoch for 3 layer feedforward NN

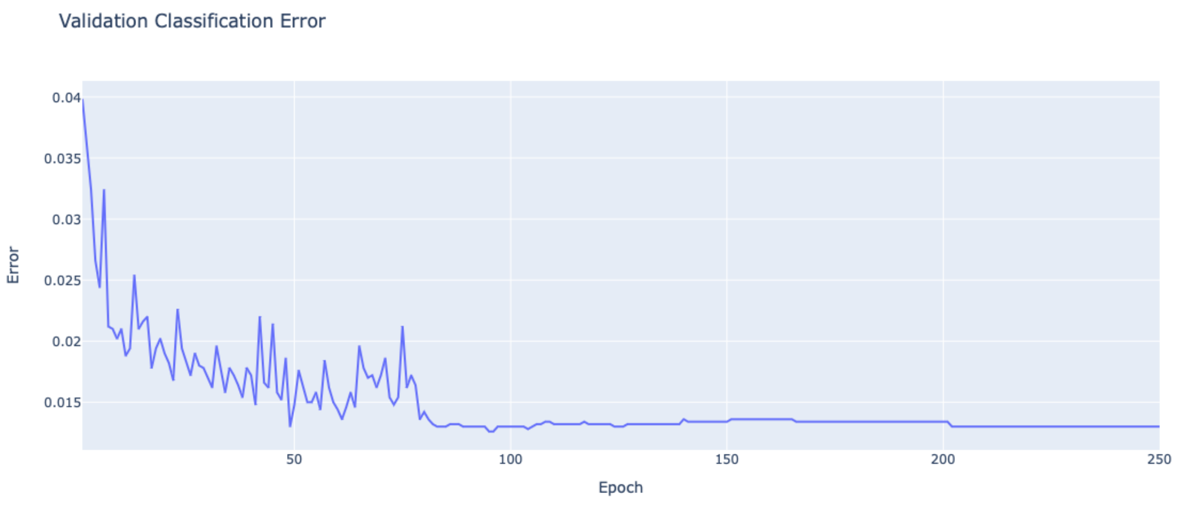

Figure 11: Validation error over Epoch for 3-layer feedforward NN

As it can be seen, without any regularization over 250 epochs, the cross entropy loss oscillates as it goes up and finally stabilizing and so does the validation error as it keeps decreasing until it flattens suggesting that overfitting might have occurred.

Question 4

3-layer feedforward neural network on MNIST with regularization

The initial steps are the exact same as question 3. When it came to defining the architecture, we started off with a layer with 500 fully connected neurons with ReLU as the activation function and l2 regularization with a C of 0.001. We then added a dropout of 0.5 meaning that 50% of the units would randomly be set to 0 thus turning those off. This generalization technique means that overfitting could be reduced and the model could learn better with more robust representation. We added a second layer the same as the first one with a same dropout. We then added the output layer with softmax as the activation function.

We then compiled the model with Adam optimizer, categorical cross entropy as the loss function striving for the best accuracy. We then set the batch size to 128 and epochs to 250 and fit the model. We then ran it again but with early stopping where a specific metric is monitored and if does not improve over a set number of epoch (patience set to 10), it would stop early and save that as the result and consider that to be the best.

After that, we plot the different graphs to compare and contrast the different metrics.

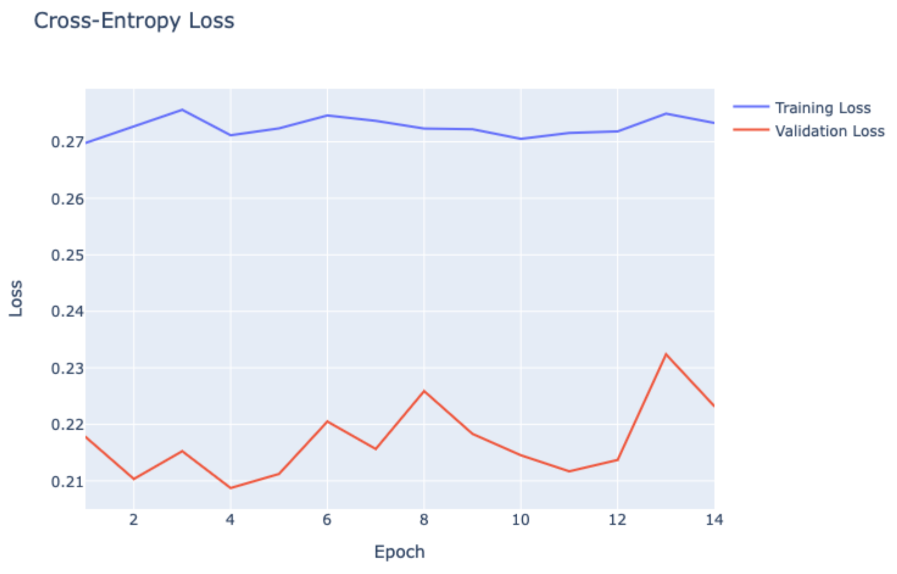

Figure 12: Cross-entropy loss over Epoch for 3-layer feedforward NN with l2 and dropout

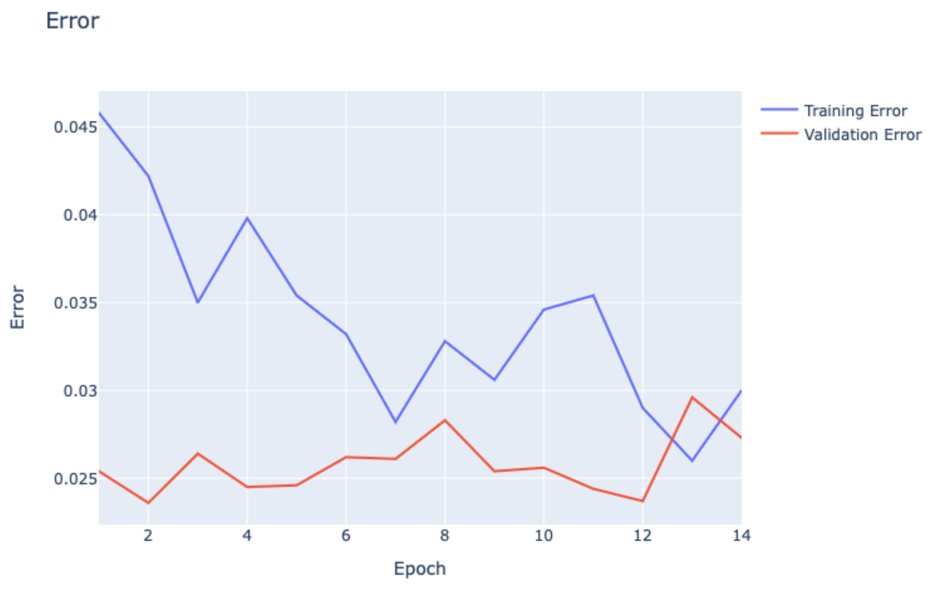

Figure 13: Validation error over Epoch for 3-layer feedforward NN with l2 and dropout

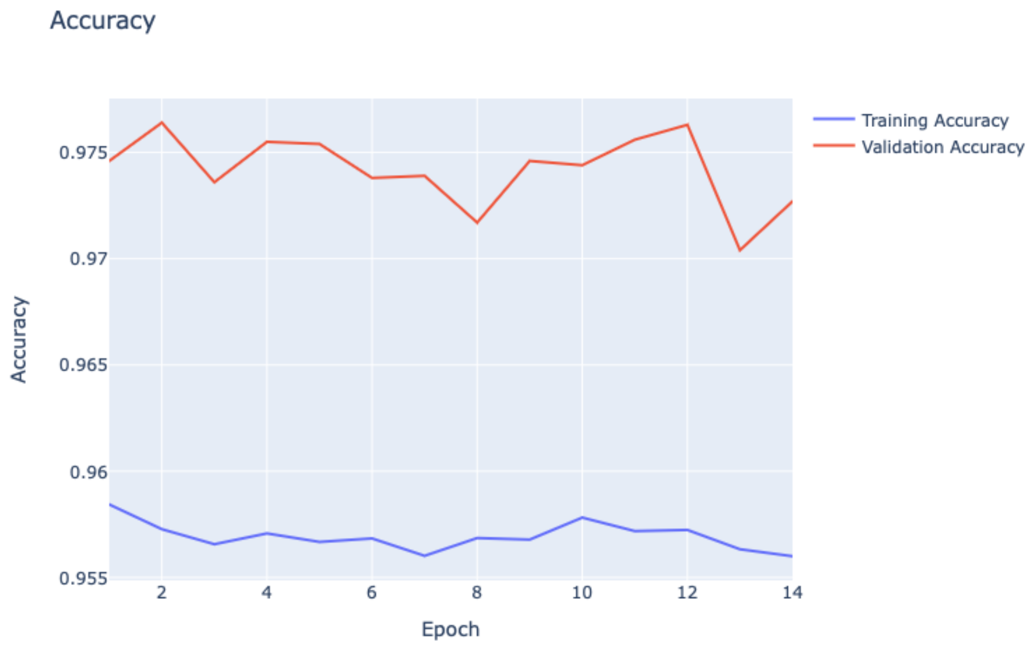

Figure 14: Accuracy over Epoch for 3-layer feedforward NN with l2 and dropout

<u>Question 5</u>

Multiclass classification on the CIFAR-10 dataset using CNN with L2 regularization and dropout

We first loaded the dataset and normalized the greyscale intensities of the x values by dividing it by 255 and converting the y values to categories. We then did a split of 70-10-20 train-validation-test.

We then defined the CNN architecture. We first added a convolutional layer with 32 filters with a kernel of 3x3 with ReLU as the activation and l2 regularizer. This exact layer was repeated followed by a max pool layer to reduce spatial dimensions using a pool size of 2x2. This layer was followed by a dropout layer of 20%.

This whole process was repeated two more times – one time with a convolutional layer with 64 filters and dropout of 30% and once with 128 filters in the convolutional layer and dropout of 40%.

The output from these layers were then flattened into a 1d layer and then a fully connected layer with 128 units and ReLU activation function followed by a dropout layer of 50% and then the final output layer with 10 class and an activation function softmax.

We then set the optimizer to minibatch SGD and then we compile the model with the loss function set to categorical cross entropy.

We then trained it over 10 epochs with a batch size of 128. We then plot the different graphs to analyze the results.

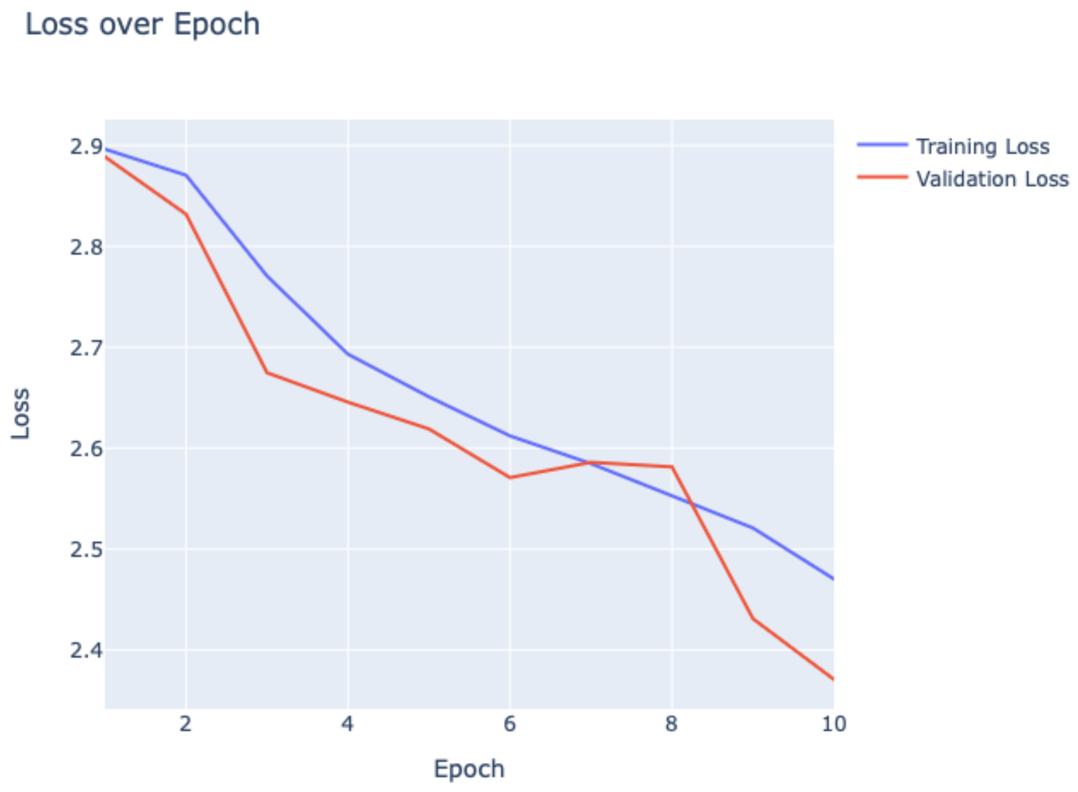

Figure 15: Loss over Epoch for CNN with l2 and dropout

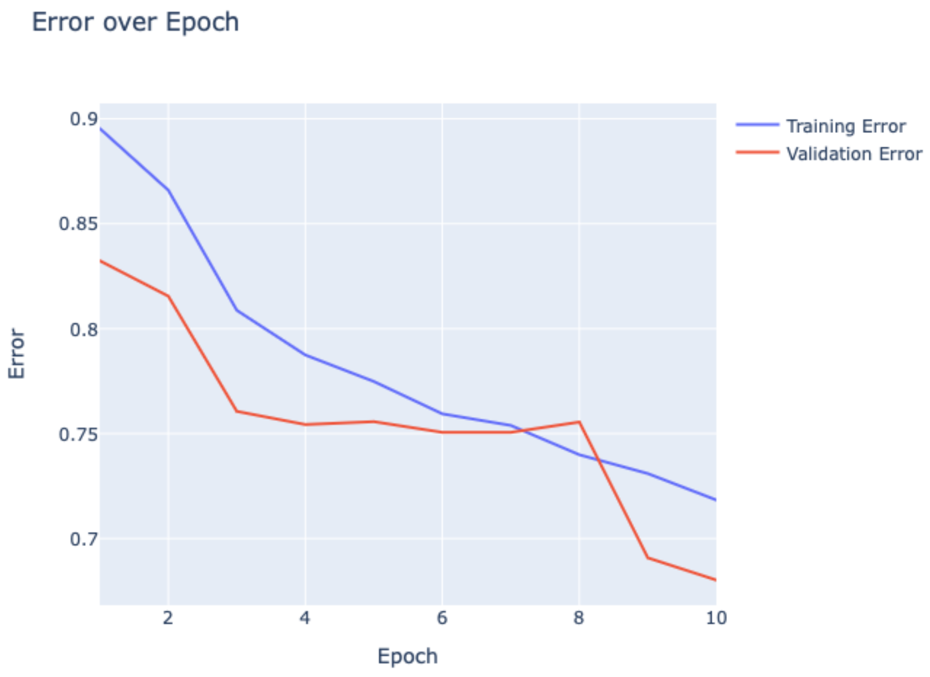

Figure 16: Error over Epoch for CNN with l2 and dropout

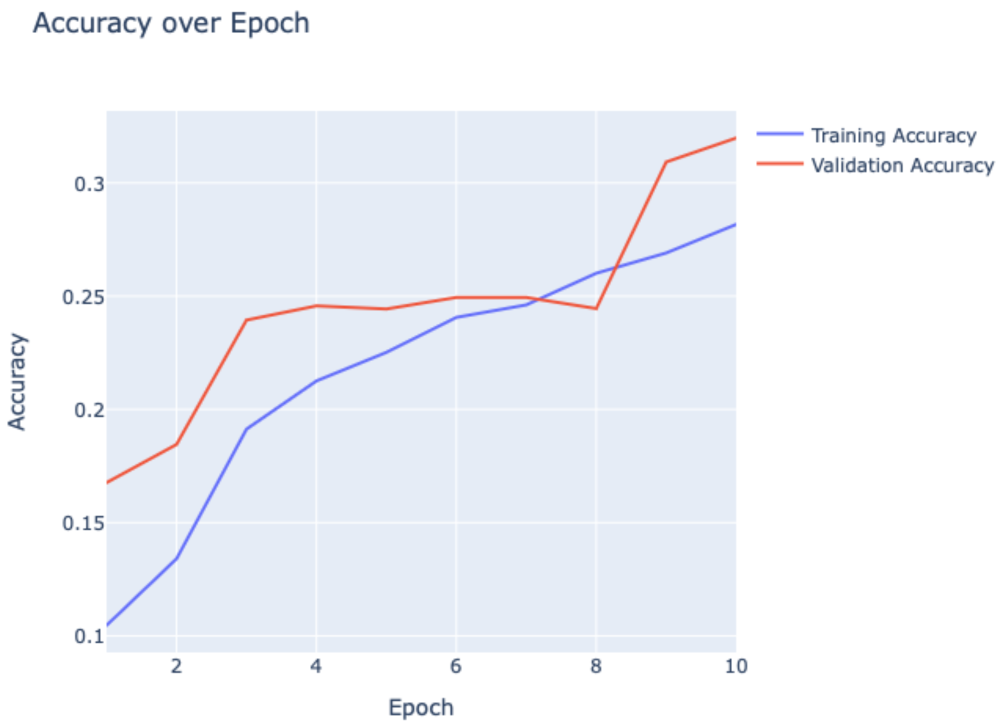

Figure 17: Accuracy over Epoch for CNN with l2 and dropout