ECE 1504 Statistical Learning

Programming Assignment 2

Hitarth Choubisa (1004965479)   Alex Labach (1000698248)

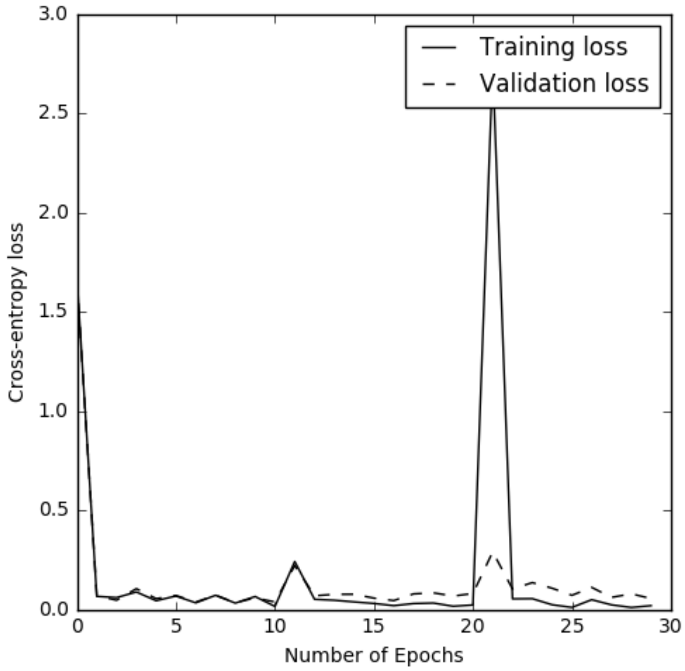
November 25, 2018

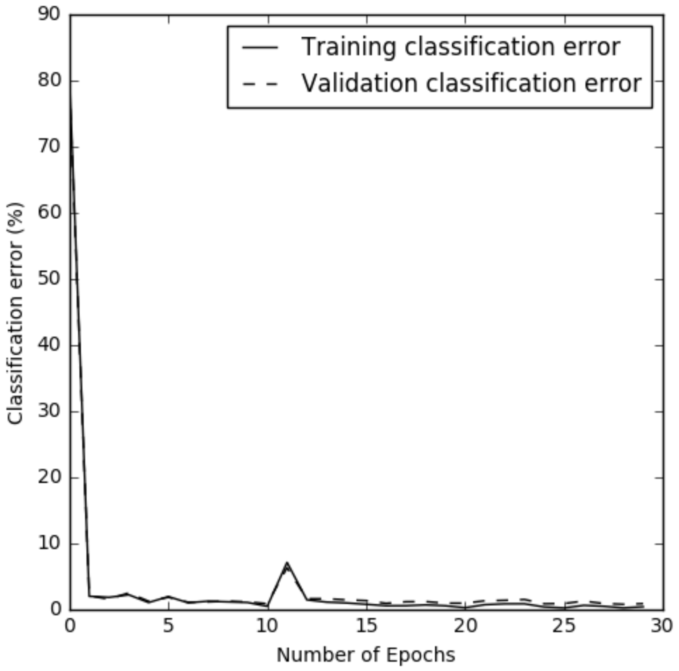
The contribution from each group member was 50%.

1. Deep Learning

1.2.2 Training

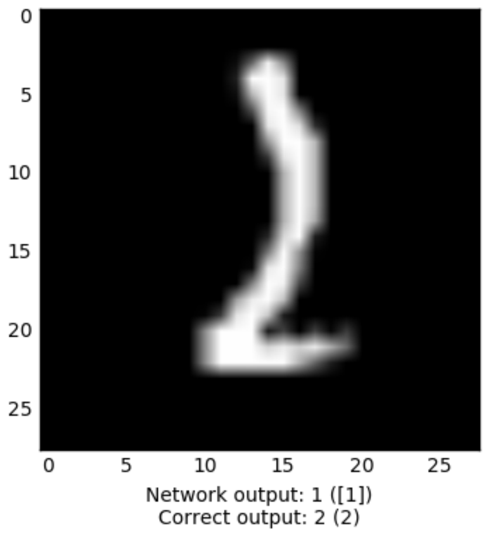
Using the given network model and training parameters, the following training and validation losses and classification errors were recorded:





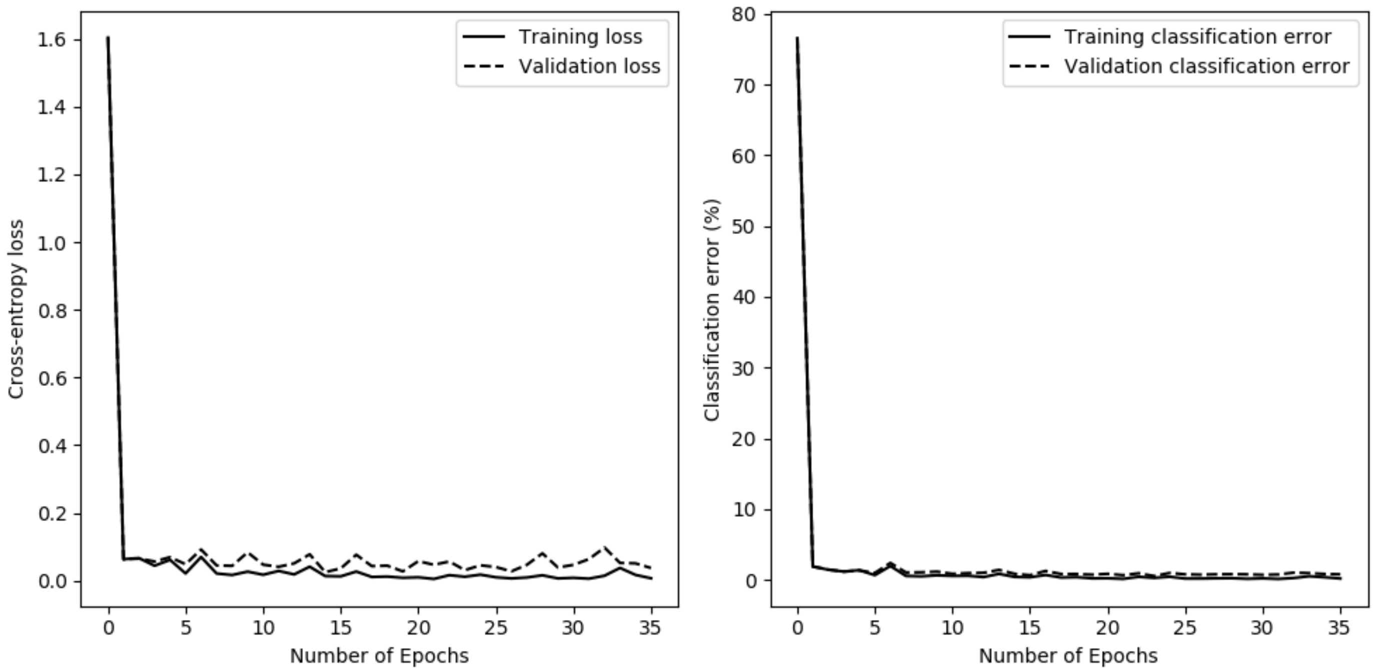
The classification accuracy achieved over the test set was 98.51%. The network stopped training after 27 epochs, with the best validation loss achieved after epoch 6.

This is an example of a number that was incorrectly classified by the network:

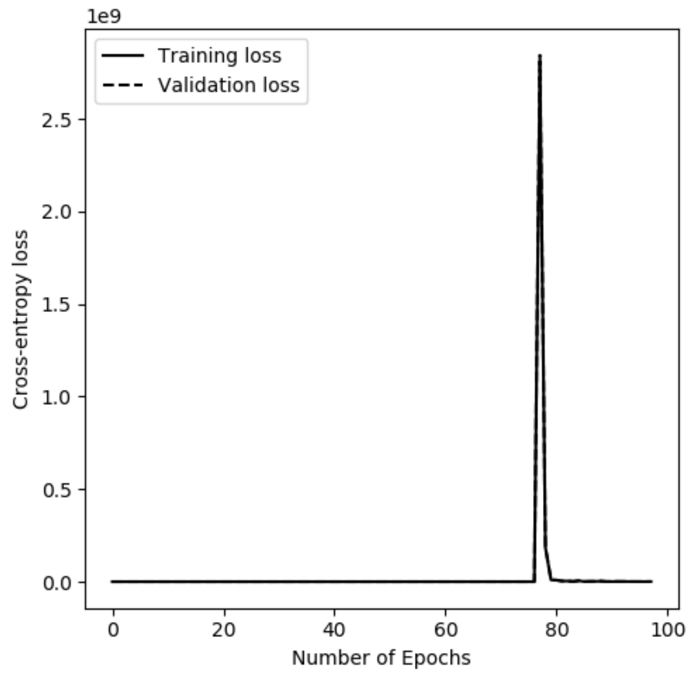


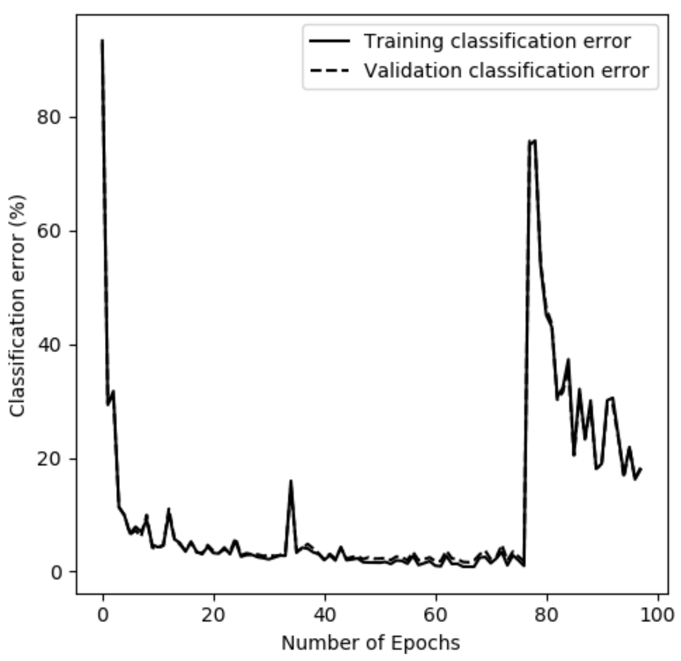
1.2.3 Tuning Hyperparameters

The following results are for batch size = 20, number of epochs = 1000, number of neurons per layer = 50, learning rate = 0.005, activation function ReLU, and maximum number of epochs without progress = 20. The classification accuracy over the test set was 98.94%, stopping after 34 epochs.



The following results are for batch size = 500, number of epochs = 1000, number of neurons per layer = 140, learning rate = 0.1, activation function Leaky ReLU with alpha=0.1, and maximum number of epochs without progress = 30. The classification accuracy over the test set was 97.88%, stopping after 96 epochs.





1.2.4 Batch Normalization

When training using batch normalization, it is desirable to normalize over a large portion of the training set rather than over a single minibatch. Using momentum approximates this. Moving averages are kept of the mean and variance used in normalization, and are updated for each minibatch with a rule of the form $x \leftarrow \text{momentum} \cdot x_{\text{previous}} + (1 - \text{momentum}) \cdot x_{\text{new}}$. This means that the mean and variance keep information from previous minibatches, allowing the normalization to occur over a large portion of the training set.

The following table shows the results achieved using batch normalization with different momentum values. All values achieved the same accuracy over the test set, but the best validation loss and accuracy were achieved with a momentum value of 0.99.

Momentum	Training loss	Training accuracy	Validation loss	Validation accuracy	Test accuracy
0.85	0.005854	99.84%	0.02186	99.40%	99.18%
0.9	0.005702	99.84%	0.02101	99.40%	99.18%
0.95	0.007039	99.92%	0.01911	99.45%	99.18%
0.99	0.007265	99.93%	0.01881	99.52%	99.18%

1.2.5 Dropout

The following table shows the results achieved using dropout with different rates and with or without batch normalization. The batch normalization momentum used was 0.99.

Dropout Rate	Batch Normalization	Training loss	Training accuracy	Validation loss	Validation accuracy	Test accuracy
0.1	N	0.02221	99.35%	0.03565	99.06%	98.75%
0.3	N	0.04813	98.65%	0.04141	98.73%	98.17%
0.1	Y	0.002375	99.93%	0.01296	99.52%	98.99%
0.3	Y	0.003468	99.91%	0.01663	99.50%	99.09%

1.2.6 Impact of Regularization Methods

Applying batch normalization improved test set accuracy for all tested momentum values compared to not using regularization. The test set accuracy was the same for all tested momentum values, although for higher momentum values, validation loss and accuracy were improved. Batch normalization with a momentum value of 0.99 achieved the best results over all experiments.

Applying dropout with a rate of 0.1 improved test set accuracy compared to not using regularization, although applying dropout with a rate of 0.3 decreased accuracy. This suggests that with tuning, dropout regularization can improve results, although the measured results were not as good as with batch normalization.

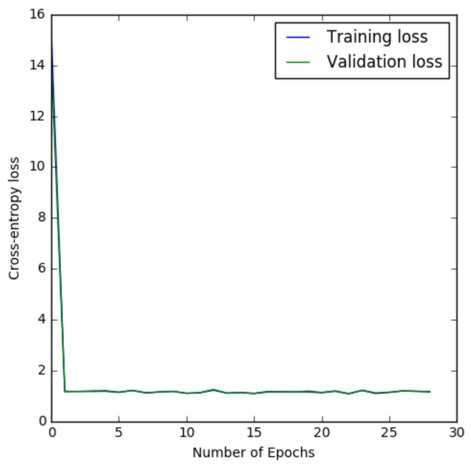
Applying both batch normalization and dropout also improved test set accuracy compared to not using regularization. This approach also achieved the lowest training and validation losses of all of the experiments, and produced better test set accuracies than dropout alone. However, it did not produce as good a test set accuracy as batch normalization alone. In this case, a dropout rate of 0.3 provided a better test set accuracy than a rate of 0.1.

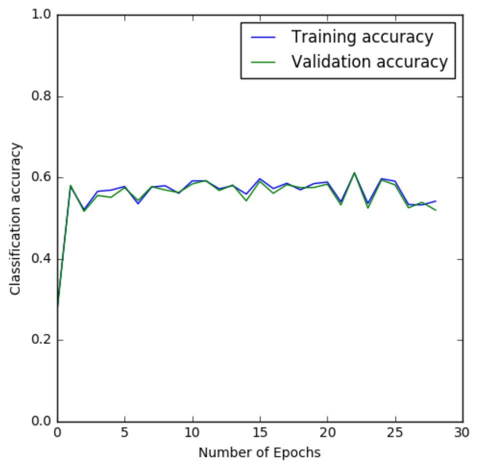
Generally speaking, either regularization method improved performance over not using regularization, as did using both regularization methods at once, but the best performance was measured using only batch normalization.

2. Transfer Learning

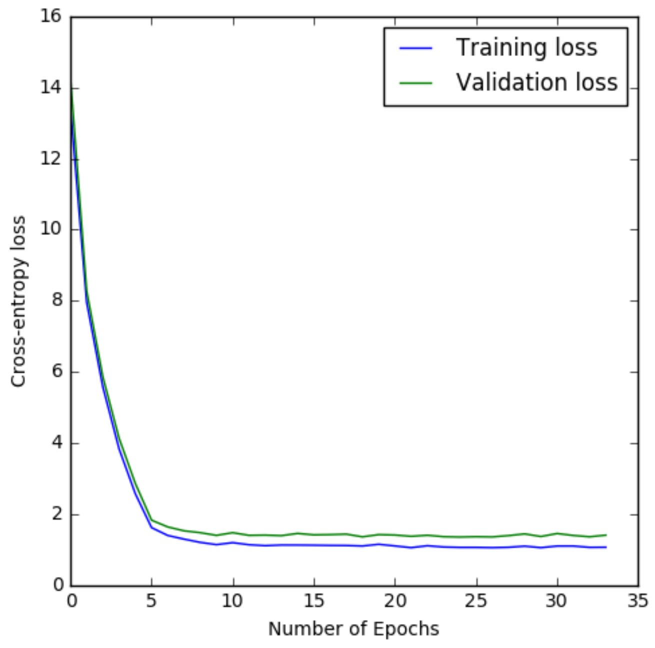
2.2.2 Reusing the Model

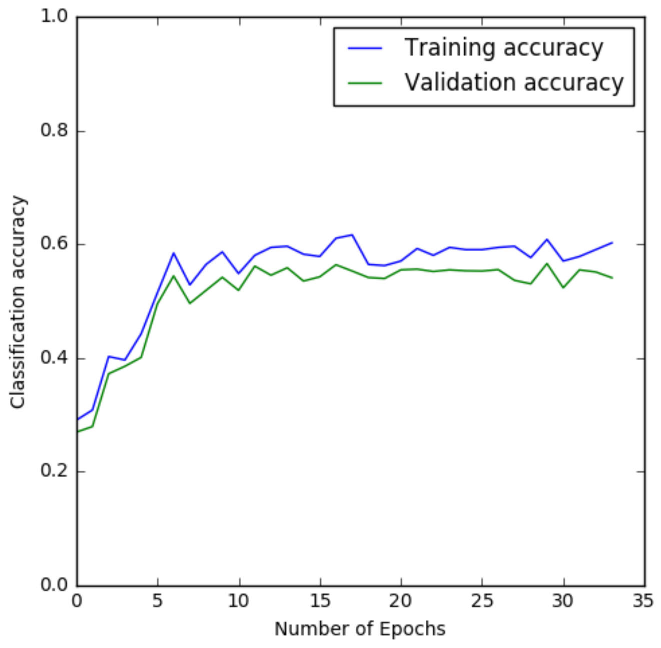
In this subsection, we imported the trained model from 1.2.2 and just replaced the softmax output layer which we trained using our new training set. When we used the whole dataset, we obtain a test accuracy **0.64** and it takes **27** iterations for training. The training and validation error and accuracy plots are,





Then, we try out training with 100 samples of each kind. The training duration is negligible in comparison to the previous case when we used the whole training dataset. We obtain a comparable test accuracy of **0.61** and a comparable number of iterations **32**. The training and validation error and accuracy plots are,

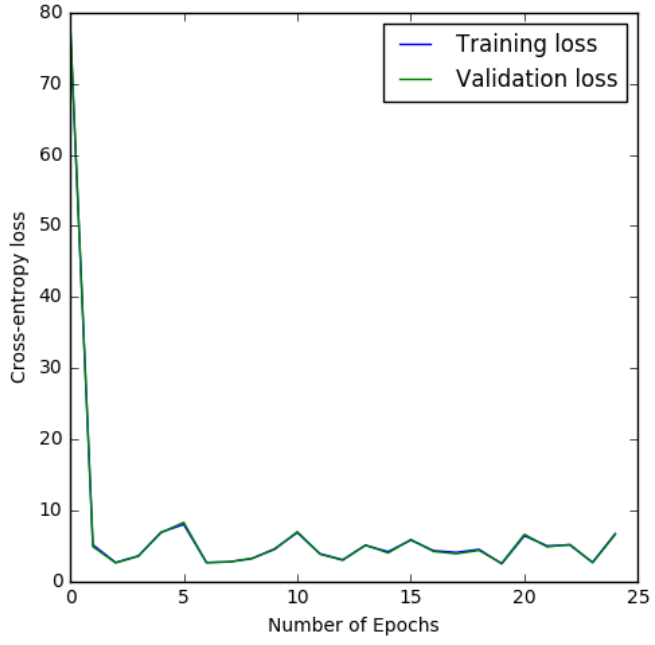


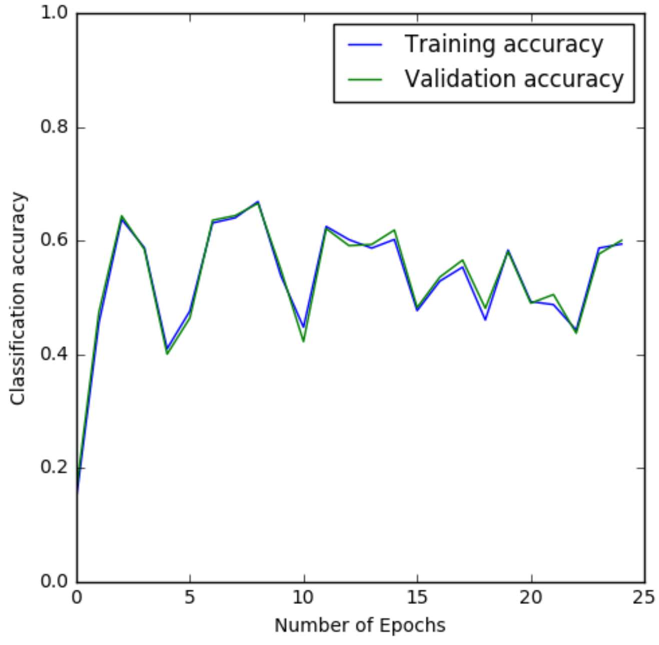


We can argue for the comparable accuracies considering the fact that the number of parameters is a small number; thus our model is not able to really learn from the large dataset.

2.2.3 Removing the last hidden layer

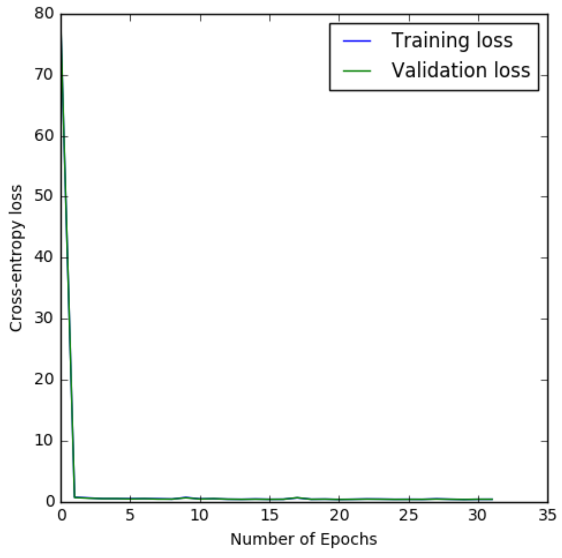
In this subsection, we removed the 5th hidden layer and trained again just the output layer with the training dataset. As expected, we obtain poor performance as compared to the case when we reused all the 5 hidden layers and trained the output layer. Here, we obtain a test accuracy of **0.61**. The accuracy is comparable to that of 2.2.2. This is expected since the higher level hidden layers search for more refined features of the datasets and since in our case, they are trained for identifying digits from 0 to 4, they don't really matter much for identifying features for digits 5-9.

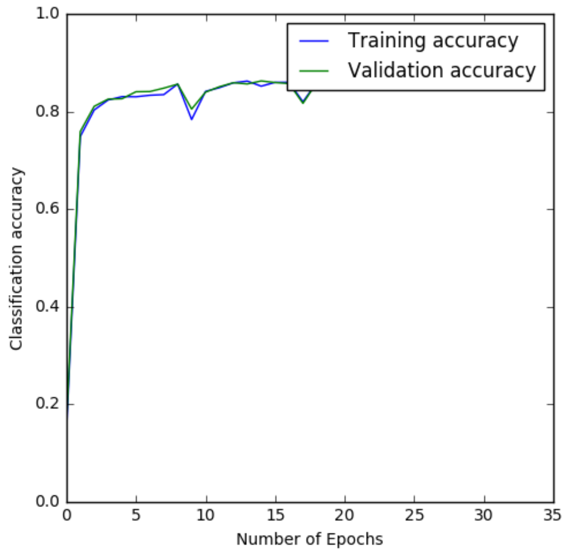




2.2.4 Unfreezing hidden layers 3 and 4

In this subsection, we retrain layer 3, 4 and output layer weights. We are able to obtain a test accuracy of **0.86**. This is way higher than we obtained in 2.2.3 or 2.2.2. The reason is simple because we are allowing our network more freedom to adjust to the changed dataset by giving a larger set of parameters. Our model is better able to identify the features of digits (5-9) now.


Training loss
Validation loss
Cross-entropy loss
Number of Epochs


Training accuracy
Validation accuracy
Classification accuracy
Number of Epochs