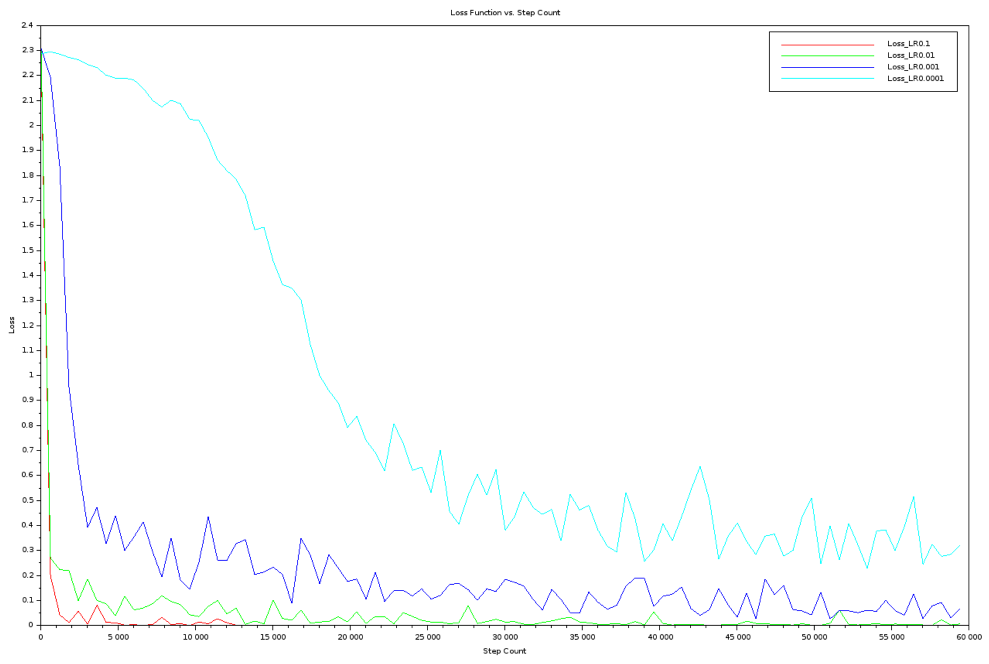
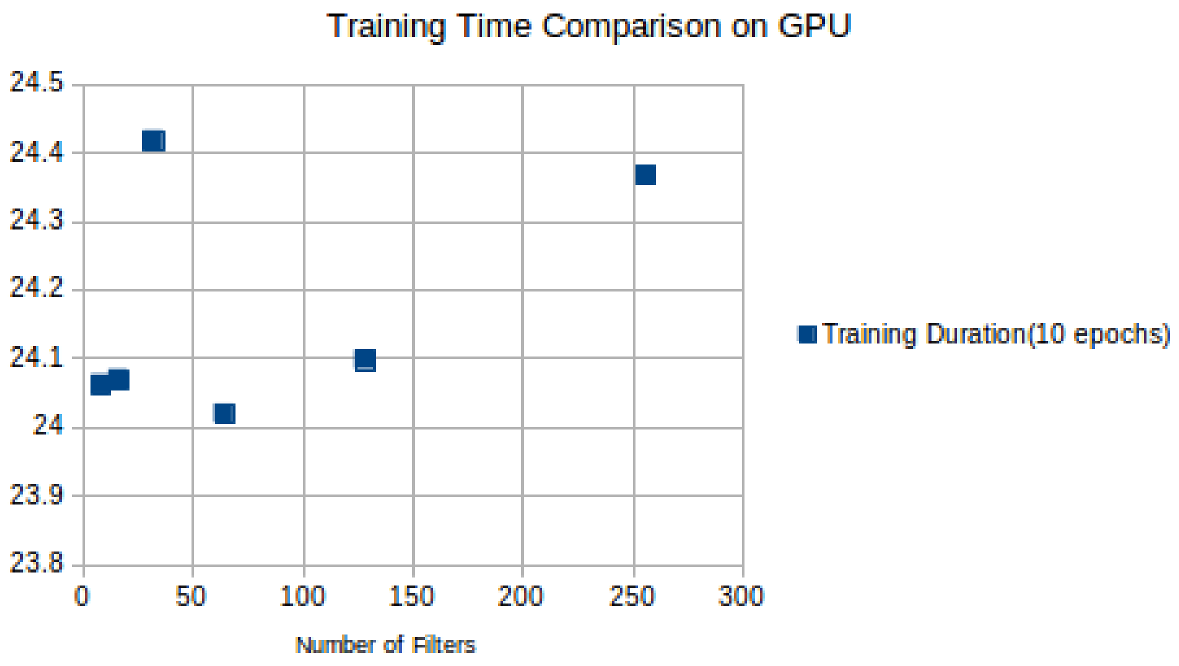
Archit Bansal, 4363946

Assignment 2

- A Convolutional Neural Network (CNN) was implemented over the course of this assignment using the framework TensorFlow, chiefly by following the tutorials provided on the website tensorflow.org.
- A basic understanding of the TensorFlow framework was developed over the course of this assignment. Even though the final product has been implemented using the Estimator API of TensorFlow, the preceding tutorials on the basics were also covered and it is possible to design a complete CNN using the basic Tensor manipulation functionalities provided by TensorFlow.
- To simplify matters, even though this report will consider the number of epochs as a standard hyper-parameter, the actual code and output generated by it have relied on internally converting epochs to steps as that is preferred by TensorFlow.
- A problem was encountered in implementing validation of data at the end of each epoch using a separate validation data set. This needed more advanced knowledge of the Estimator API in order to introduce the appropriate hooks or design more customized model generation / training operation functions. Hence, the loss function plot provided below has been computed on training data and not on validation data.



- The plot presented above depicts the training loss of the CNN on the y-axis and the step count on the x-axis for 4 different learning rate: 0.1, 0.01, 0.001, and 0.0001. On account of time constraints, only 100 epochs were used to train the CNN as that appeared to be sufficient to demonstrate the difference between the 4 learning rates.
- As we can observe, training loss converged very quickly with a learning rate of 0.1 and the rate of convergence kept dropping as the learning rate was increased. This is expected behavior. It is notable that since even with 10 epochs, a learning rate of 0.1 was able to achieve almost 99% accuracy, the key difference in overall performance of learning rates will become clear only after convergence is achieved, and the difference will be marginal. However, we can expect lower learning rates to achieve higher accuracy unless we run into over-fitting. Hence, the choice of learning rate is highly application specific – the more demanding the application, the lower the learning rate required.



- The figure above shows the output of using pythong's time.time() to measure the training time of the CNN for only 10 epochs when a GPU was used. The training time of 48 filters seems to be an outlier caused by the inherent inaccuracies assosciated with using time.time().