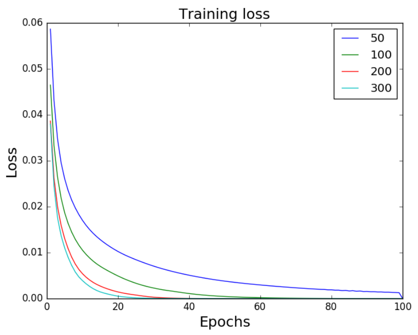
Programming Assignment 1

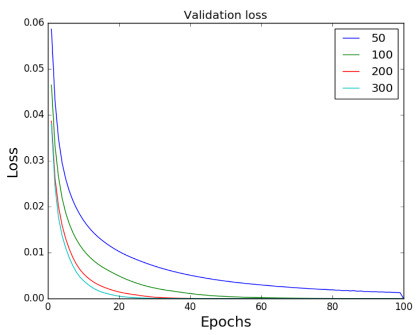
Priyesh V, CS15S012

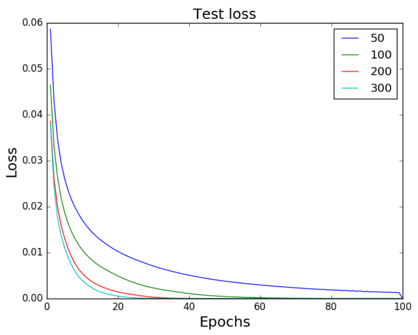
1 Performance with 1 hidden layer

All the experiments were done with sigmoid as the final output layer.

The experiments for questions $1 - 4$ where run without any annealing and where run till 100 epochs.

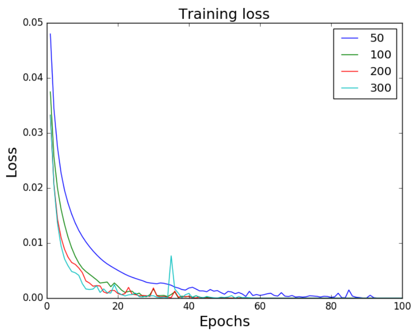


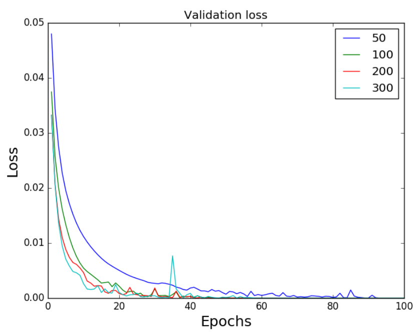


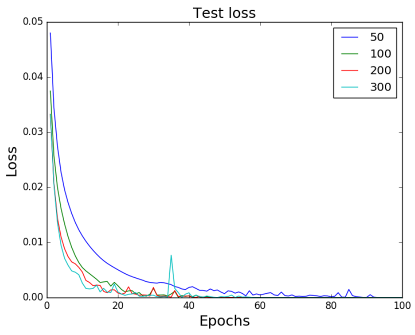


It can be seen from the plots that the error converges steadily with minimal fluctuations. Moreover, with the increase in number of hidden neurons, the network converges faster.

2 Performance with 2 hidden layers

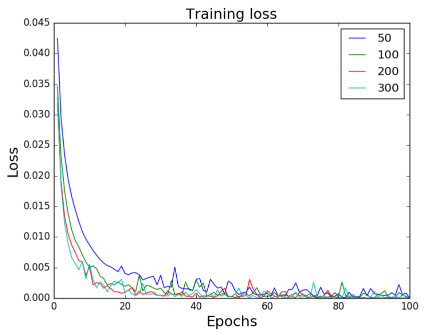


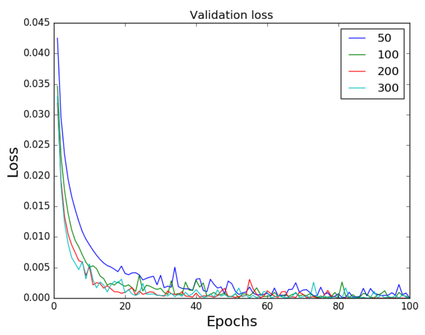


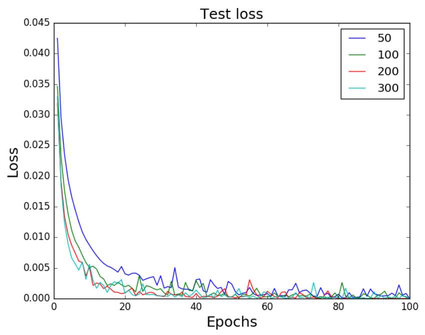


As we increase the number of hidden layers, error fluctuations starts increasing. More importantly the fluctuations are higher for networks which have greater number of neurons. This aligns with the intuition that number of parameters in these model makes the loss surface a highly non-linear space thus creating lots of sub-optimal local optimas. Like before, since the capacity of the network increases with number of parameters, convergence quickens when size of hidden layers are increased.

3 Performance with 3 hidden layers

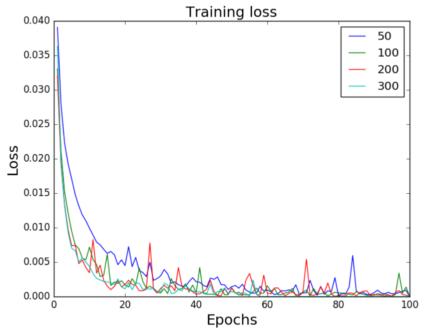


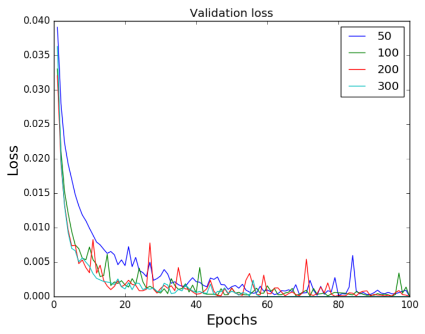


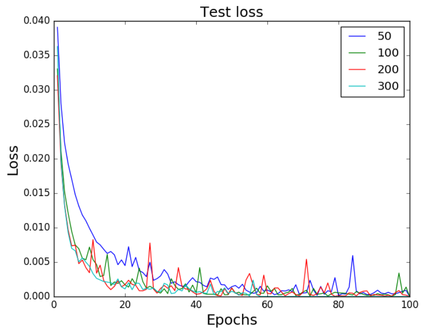


With 3 hidden layers, the fluctuations increase dramtically. Even for the one with least number of units, we can now see significant fluctuations. This demonstrates the property that non-linearity of our model increases with the increase in depth. Note that, even near 100 epochs, it doesn't converge to a single fixed value and keep oscillating.

4 Performance with 4 hidden layers



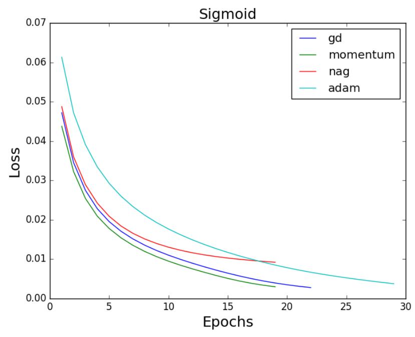


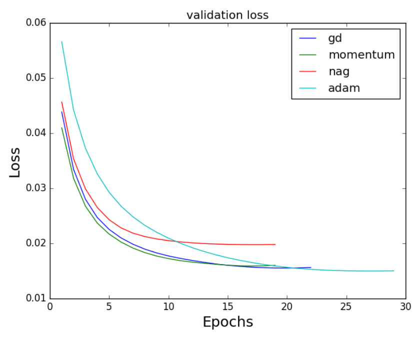


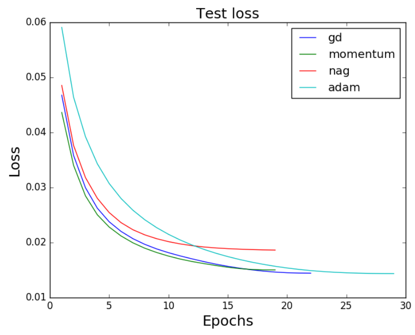
Just like the previous experiment, the increase in depth significantly magnifies the amount of fluctuations. Important thing to note here is that apart from 50 hidden unit's plot, all other plot almost overlap with each other and demonstrate similar performance and convergence rate, thus it can be inferred that the overall capacity required for the given MNIST problem is easily obtained when we hit this mark.

5 Compare different optimization strategies

The following experiments learning rate and momentum are picked based on the best validation loss and not by how fast the loss drops. The experiments were terminated with patience 2, i.e we stop the experiment when the validation loss increases twice.



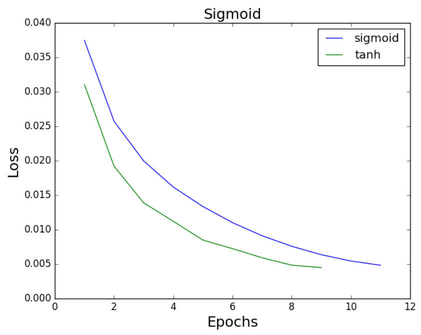


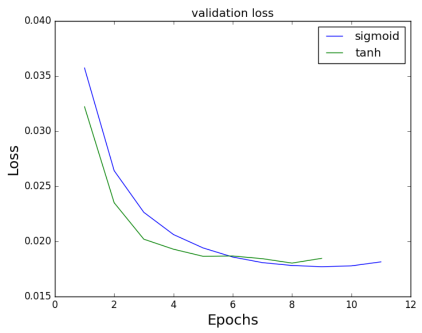


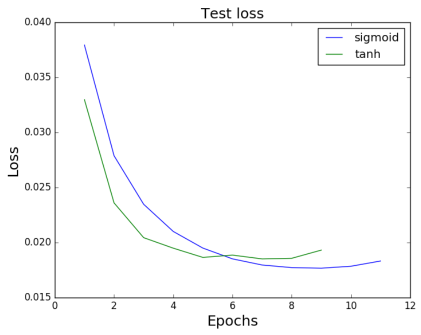
Counter intuitive to our belief, Adam (which was expected to perform the best) converged significantly slower as compared to other methods, though it still converges to better values in test and validation sets. Even Nag, performed poor in comparison to vanilla SGD or momentum. Possibly, because of this being a toy dataset, simple momentum/SGD works best.

6 Sigmoid Vs Tanh activation functions

The following experiments learning rate and momentum are picked based on the best validation loss and not by how fast the loss drops. The experiments were terminated with patience 2, i.e we stop the experiment when the validation loss increases twice.



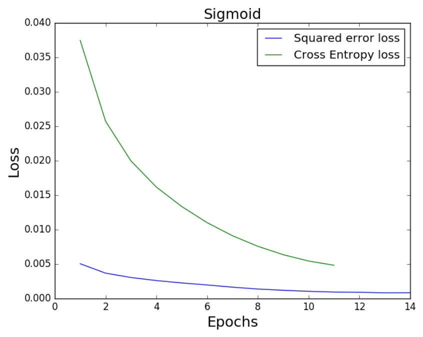


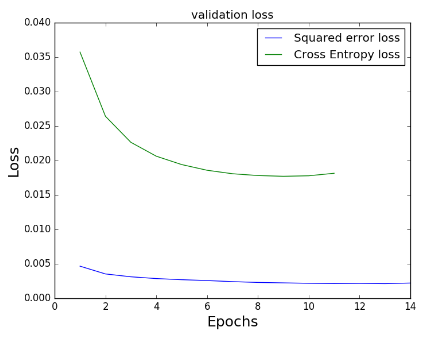


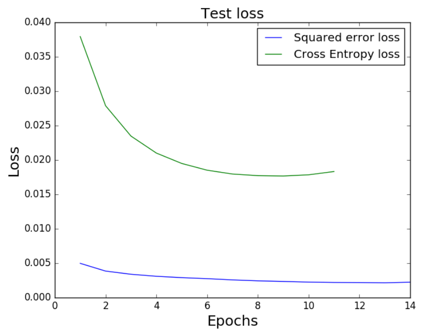
It's evident from all the plots that tanh being centres at 0, converges much quicker than sigmoid. Though it's interesting to see that on the longer run sigmoid tends to marginally outperform tanh. Possible reason for this might be that the tanh function is mean shifted and scaled version of sigmoid, which essentially the network is itself capable of doing but requires some update iterations before it achieve that.

7 Cross Entropy vs Squared error Loss

The following experiments learning rate and momentum are picked based on the best validation loss and not by how fast the loss drops. The experiments were terminated with patience 2, i.e we stop the experiment when the validation loss increases twice.







Interestingly, squared error loss converges almost immediately, that too to a much lower error as compared to the cross-entropy. This might be because we use sigmoid output layer rather that softmax. If we had used softmax output layer, cross entropy would have converged faster.

8 Best performance

I get best performance of 98.57 with $activation = relu$, $loss = squared-error$, $hidden_{l}ayers_{s}ize = 625, 625$, $lr = 0.01$, $opt = adam$, $anneal = true$