Experiments:

Eyal Ben David - 305057416

May 20, 2018

Let's consider the MNIST dataset in order to test our deap neural network package implementation. As instructed, we have implemented the package using only Python and Numpy. The implemented solution is able to construct, train and apply any user-defined deep architecture containing any number of fully connected layers whose activation can be ReLu, Sigmoid and Softmax functions. Training is implemented using only basic Stochastic Gradient Descent (SGD) with either cross-entropy or mean squared error (MSE), and can be applied either on full or mini batches.

Batch Size:

We consider a basic architecture with one hidden layer containing 128 neurons, a ReLu activation function and a Softmax output layer. We used different batch size's in order to reveale the correlation between the batch size and the learning performence.

Figure 1: Accuracy in correlation with number of epochs, for different batch size's.

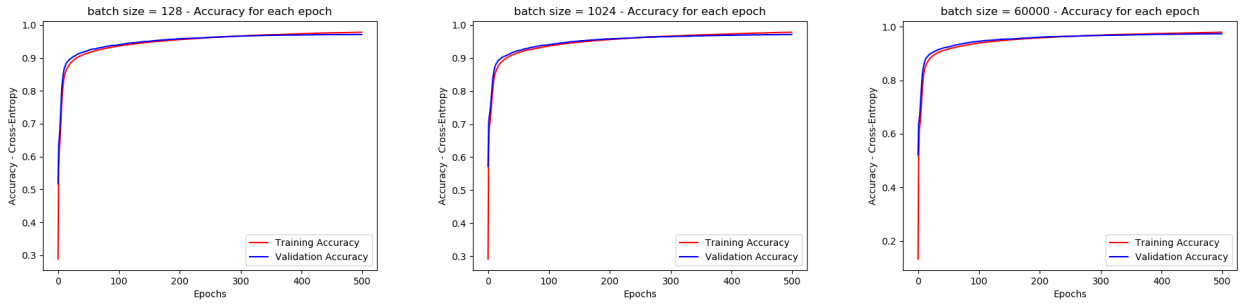


Figure 2: Cross-Entropy Loss value in correlation with number of epochs, for different batch size's.

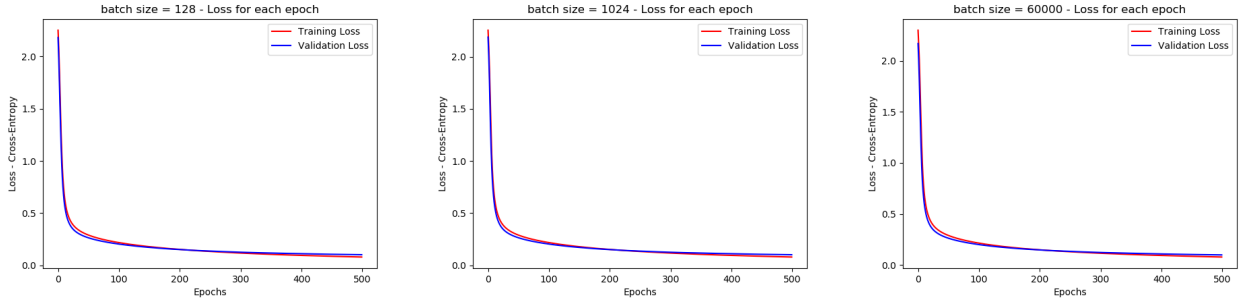


Figure 3: Accuracy in correlation with number of backpropagation's, for different batch size's.

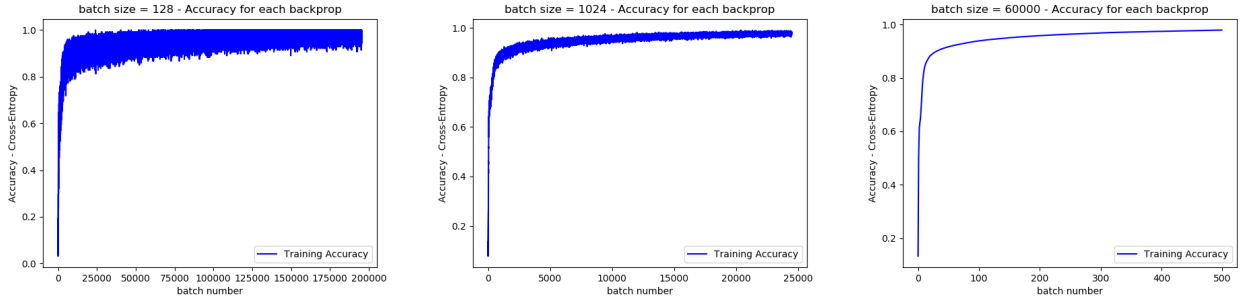


Figure 4: Cross-Entropy Loss value in correlation with number of backpropagation's, for different batch size's.

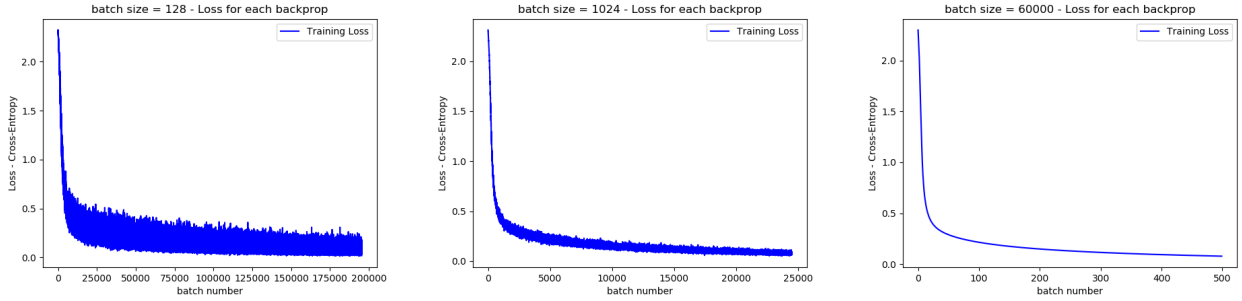


Table 1: Final Cross-Entropy Loss for different batch size's.

Batch size	128	1,024	60,000
Training	0.088	0.084	0.081
Validation	0.105	0.102	0.101

Table 2: Final Accuracy for different batch size's.

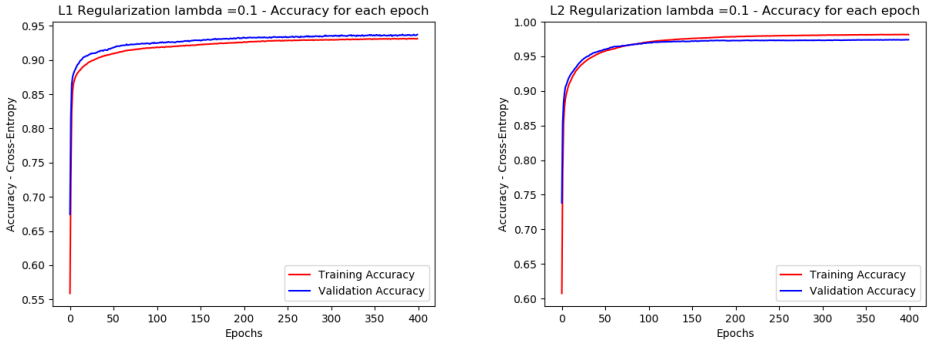
Batch size	128	1,024	60,000
Training	98.9	99.1	99.35
Validation	97.52	97.58	97.61

Despite receiving similar results for all batch size's, the batch size affectsthe speed of learning and the optimal learning rate. As can be seen, the smaller the batch size is, we add more noise and randomness to the learning procces and hence we explore more options for minimal loss. Adittionally, large batches require more computational power and larger memory.

Regularization:

We consider the same architecture as above. We want to explore the learning result's as correlation to Regularization. Hence we will use different regularizetion's method's (L1, L2) and different decay's inorder to explore this subject.

Figure 5: Accuracy in correlation with number of epochs, using different regulariztion method's, decay values are 0.1, 0.05.



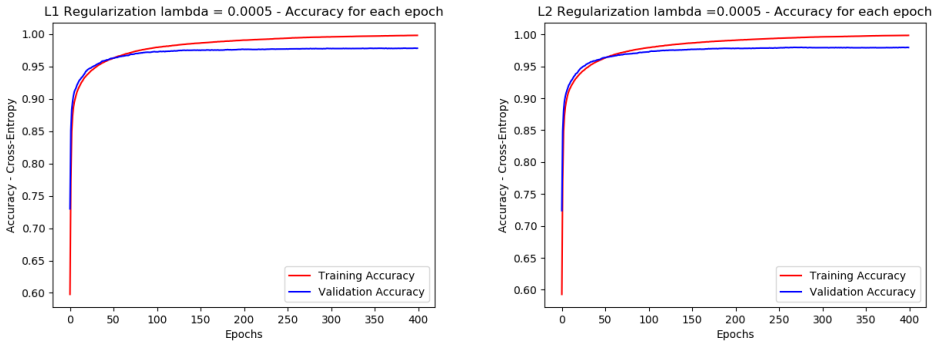


Figure 6: Cross-Entropy Loss value in correlation with number of epochs, using different regulariztion method's, decay values are 0.1, 0.05.

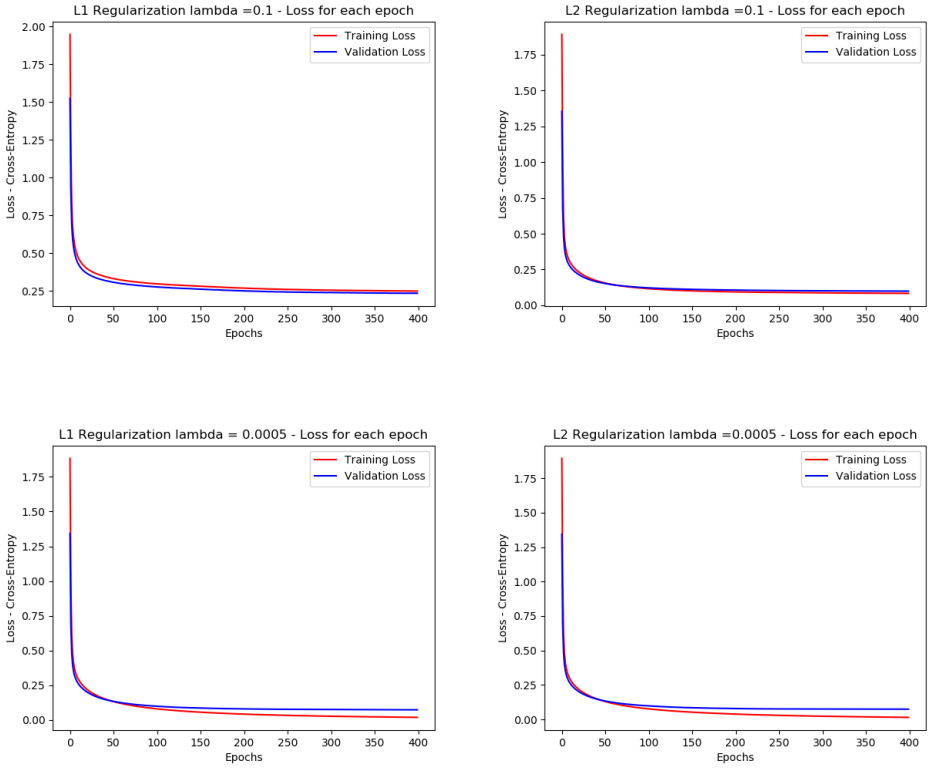


Figure 7: Accuracy in correlation with number of Backpropagation's, using different regulariztion method's, decay values are 0.1, 0.05.

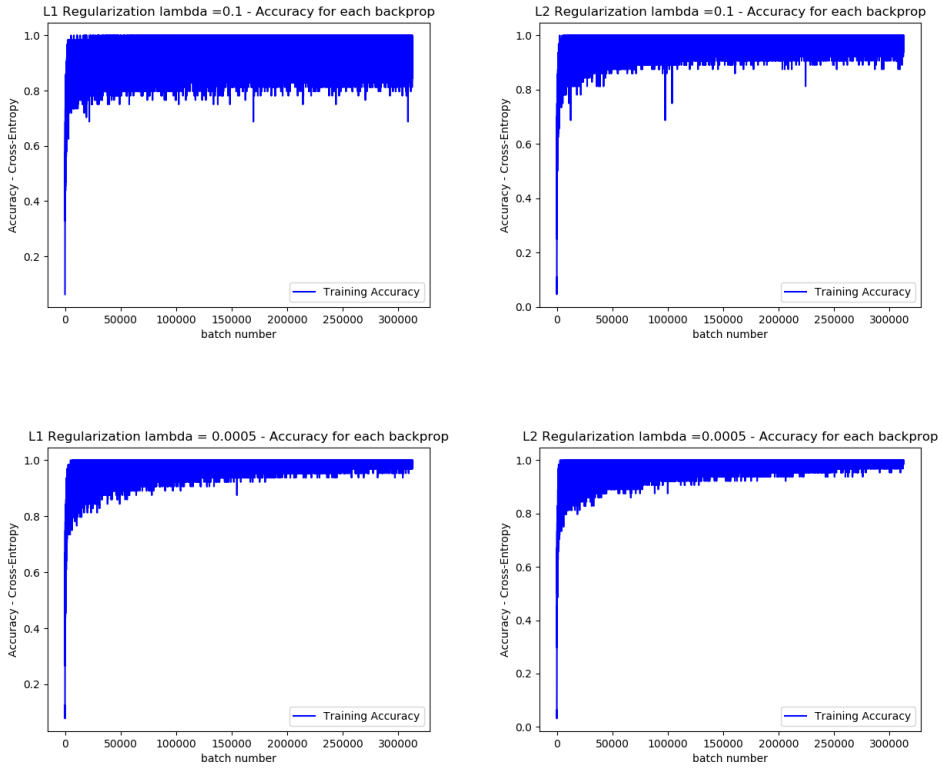
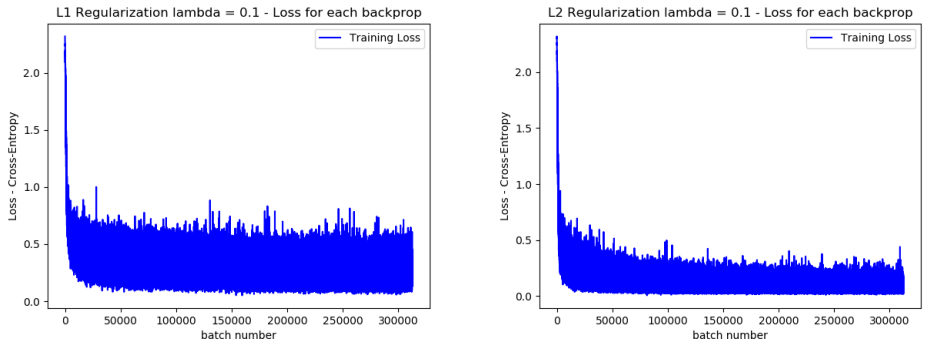
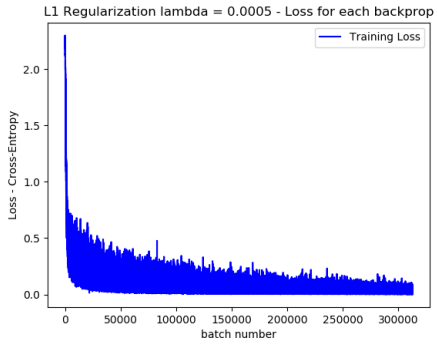


Figure 8: Cross-Entropy Loss value in correlation with number of Backpropagation's, using different regulariztion method's, decay values are 0.1, 0.05.





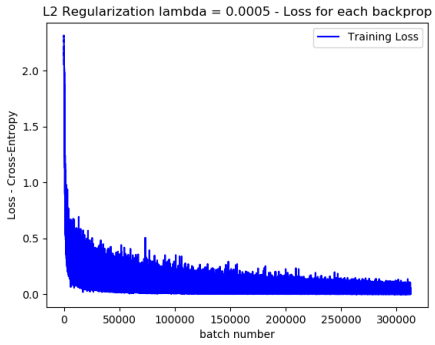


Table 3: Final Cross-Entropy Loss for different regularization method's, using different decay's.

Regularization method	L1, Decay=0.1	L2, Decay=0.1	L1, Decay=0.0005	L2, Decay=0.0005
Training	0.289	0.0925	0.068	0.063
Validation	0.286	0.0934	0.0942	0.093

Table 4: Final Accuracy for different batch size's.

Regularization method	L1, Decay=0.1	L2, Decay=0.1	L1, Decay=0.0005	L2, Decay=0.0005
Training	91.21	97.51	98.3	98.52
Validation	91.34	96.8	96.2	97.01

As can be seen from the result's above, a very strong regularization keeps the model from overfiting but at a cost. The overall training accuracy has fallen significately by about 7 percntage. It can be assumed from these experiments that the L2 regularization fit's more to the MNIST dataset then the L1 regularization and improved the generalization of the classifier.

Architecture:

We would like to research how the architecture selection affects the results for the MNIST dataset. Inorder to do so we have conducted different architecture's up to a depth of 3 (maximum 2 hidden layers) and up to a width of 512. Before disccusing the result, let's show them.

Figure 9: Accuracy in correlation with number of epochs, using different Architecture's.

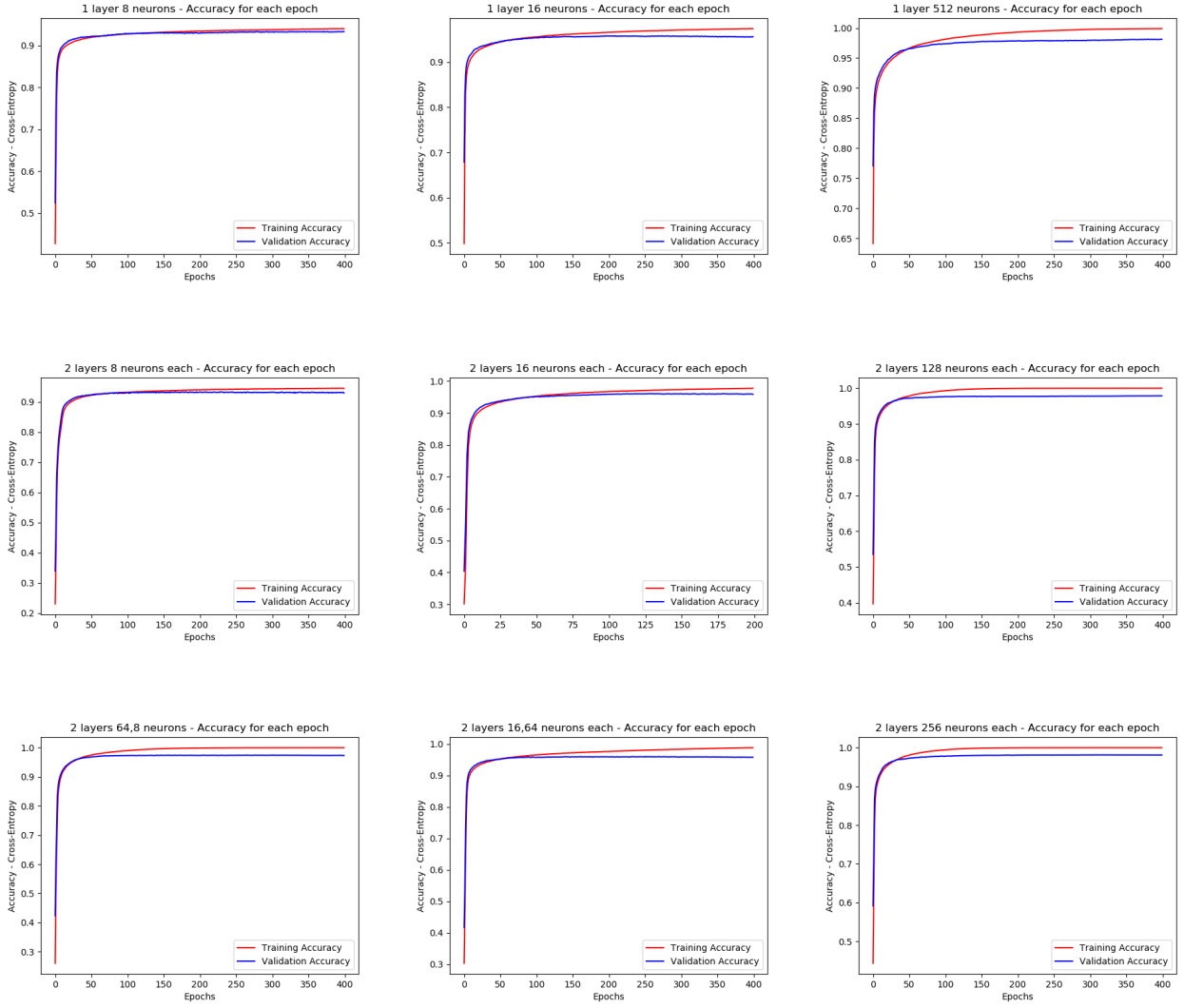


Figure 10: Cross-Entropy Loss value in correlation with number of epochs, using different Architecture's.

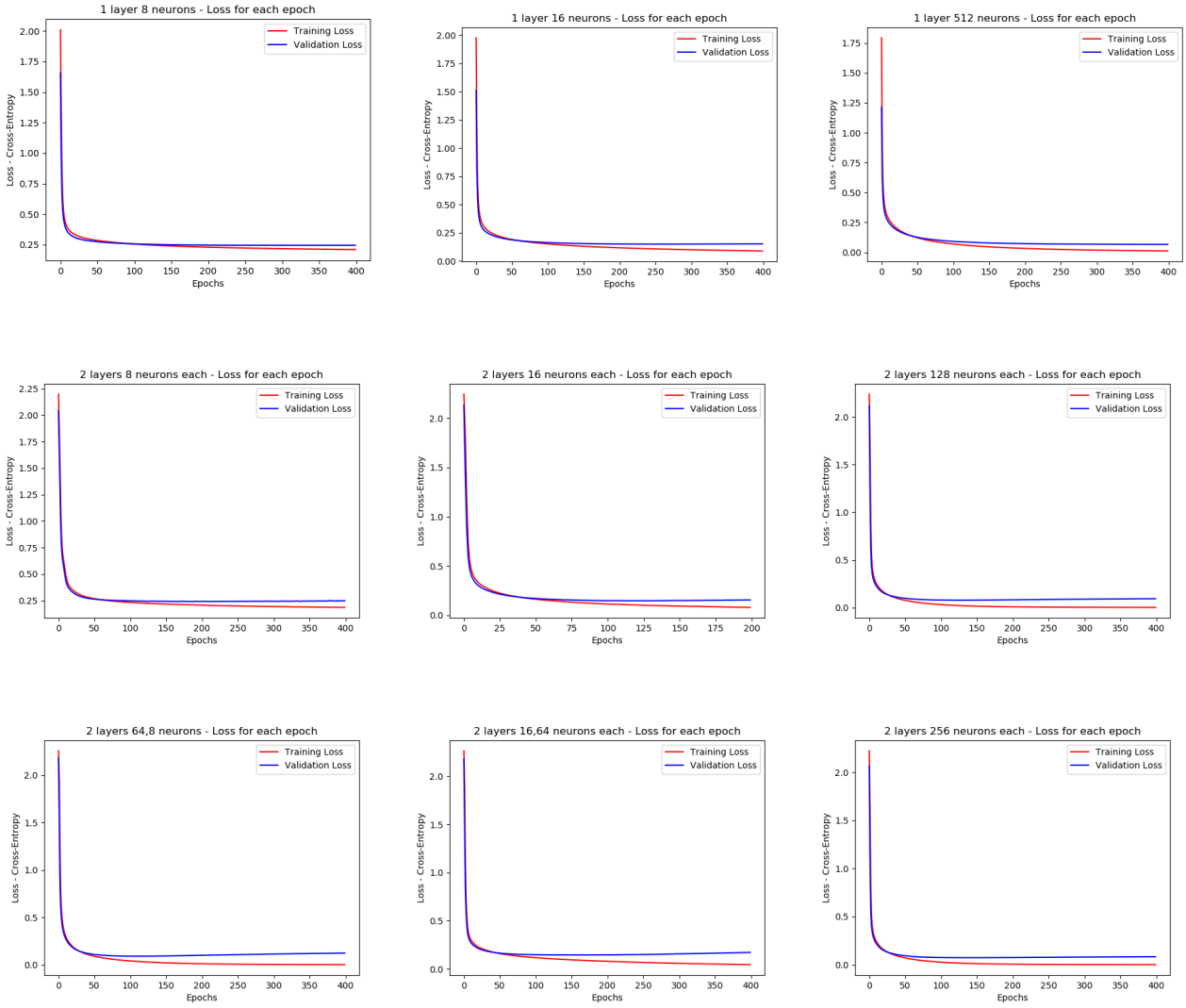


Table 5: Final Cross-Entropy Loss for different architecture's - one hidden layer.

Number of neurons	8	16	512
Training	0.311	0.165	0.01
Validation	0.315	0.198	0.098

Table 6: Final Cross-Entropy Loss for different architecture's - two hidden layer's.

Number of neurons	**8,8**	**16,16**	**128,128**	**64,8**	**16,64**	**256,256**
Training	**0.236**	**0.081**	**0.017**	**0.031**	**0.0169**	**0.01**
Validation	**0.263**	**0.107**	**0.098**	**0.1002**	**0.109**	**0.101**

Table 7: Final Accuracy for different architecture's - one hidden layer.

Number of neurons	8	16	512
Training	91.12	95.21	1
Validation	91.08	94.5	97.2

Table 8: Final Accuracy for different architecture's - two hidden layer's.

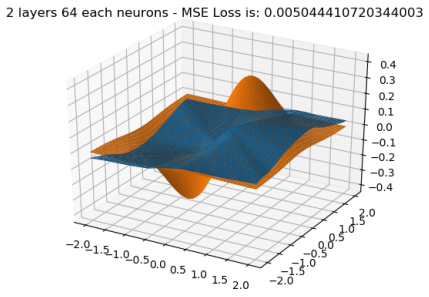
Number of neurons	**8,8**	**16,16**	**128,128**	**64,8**	**16,64**	**256,256**
Training	**93.31**	**97.6**	**99.54**	**99.01**	**99.6**	**1**
Validation	**92.8**	**96.9**	**97.17**	**97.04**	**96.8**	**97.1**

As can be seen from the result's above, the architecture has a major influence on the learning performence. While building a machine with many neurons and many layes may fit very well to the training set, it might give bad result's on new data arriving, which is reflected by a decrease in the validation results. On the other hand, choosing a simple network with a low number of neurons can improve the overfit but might not give a rich enough hypothesys region inorder to reflect the target function. We conclude that the best architecture for the MNIST dataset is the two hidden layers with 64 and 8 neurons respectively, both with ReLu activation function. Moreover, we saw from our experiments that the ReLu activation function worked much better on the given dataset.

Regression:

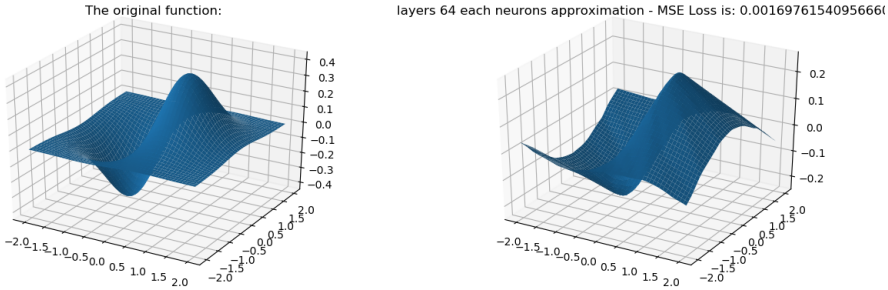
For this section we have created a synthetic dataset by sampling number of point's uniformally random between [-2,2], and creating the function: $f(x) = exp(-x_1^2 - x_2^2)$.
After training the model with 100 sampled point's and testing it once- with 100 sampled point's and once with 1,000 sampled point's, we have found that the best fit architecture is with two hidden layers, both with 64 neurons of ReLu activation function. Here are the results of our best architecture for this task:

Figure 11: Test Graph of f(x) and estimated f'(x) out of regression, for 100 test point's.



Altough the MSE gives low cost function, it is clear that the regression didn't provide
good results in the edges of the function, it is probably because of low amount of training set.

Figure 12: Test Graph of f(x) and estimated f'(x) out of regression, for 1000 test point's.



It can be seen from both graphs that the chosen architecture gives a good approximation of the original function. Furthermore, it is clear that the more examples we use, the better prediction we get. for to complicated architecture we can get better results on the train set, but worse results on test set.