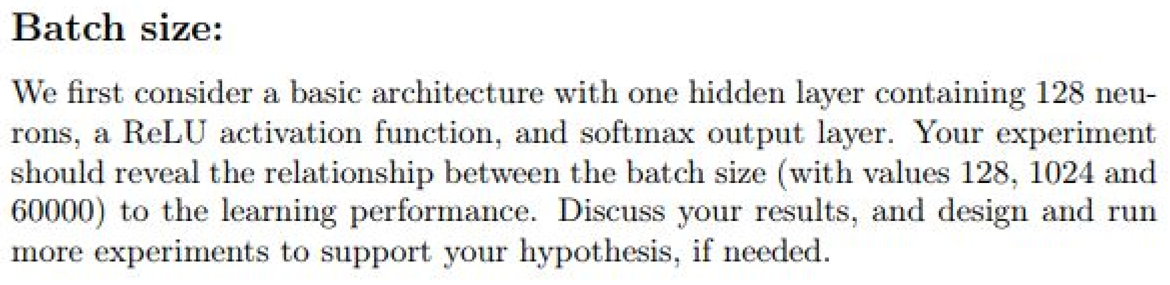
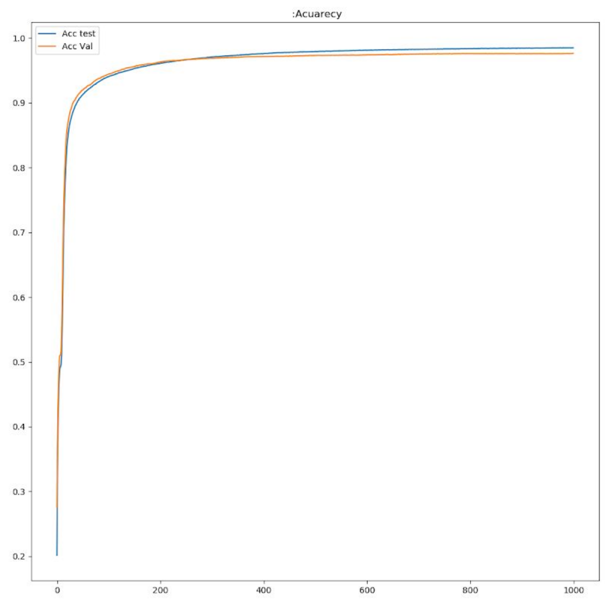
Hw2 Deep learning Report
Alejandro Moscoso 332336908
Alex Finkelshtein 307573378

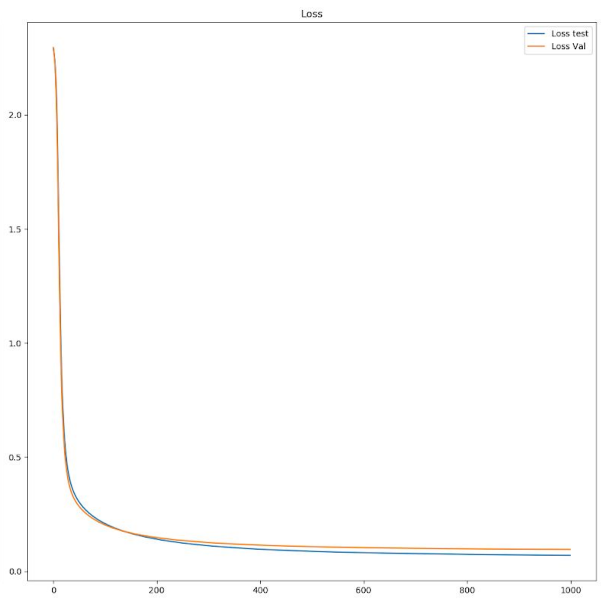


Batch size:

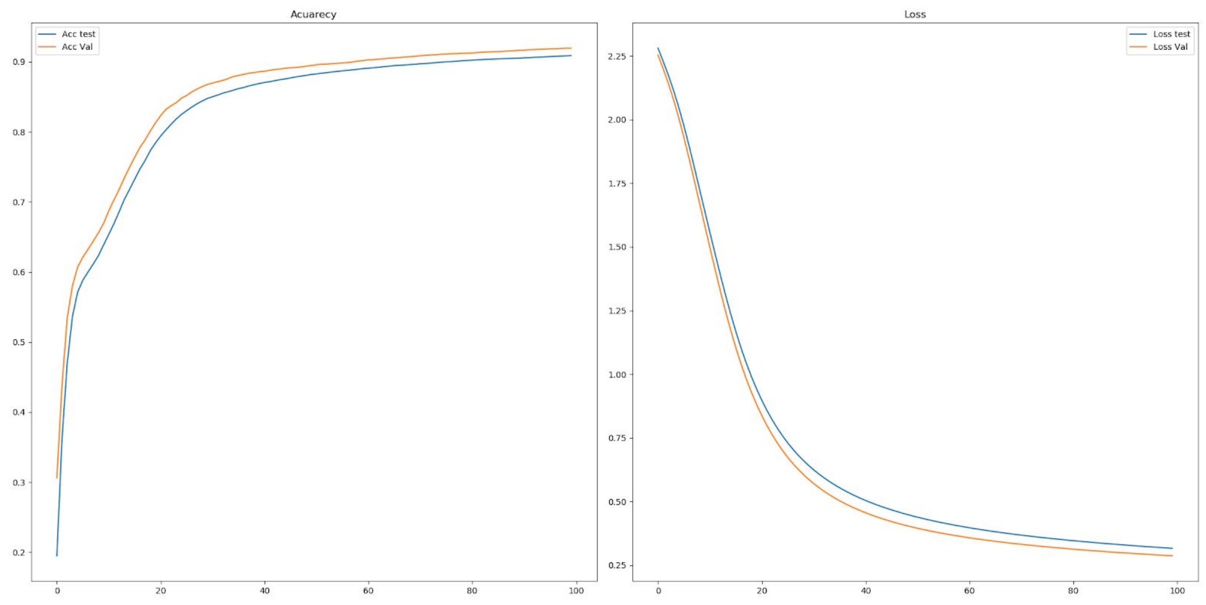
We first consider a basic architecture with one hidden layer containing 128 neurons, a ReLU activation function, and softmax output layer. Your experiment should reveal the relationship between the batch size (with values 128, 1024 and 60000) to the learning performance. Discuss your results, and design and run more experiments to support your hypothesis, if needed.

Hypothesis -- We will not see much of a difference in the rate of convergence, or the final accuracy and loss as a function of batch size. However, we presume that the run time per batch will be far lower for smaller batches (greater use of parallelism).

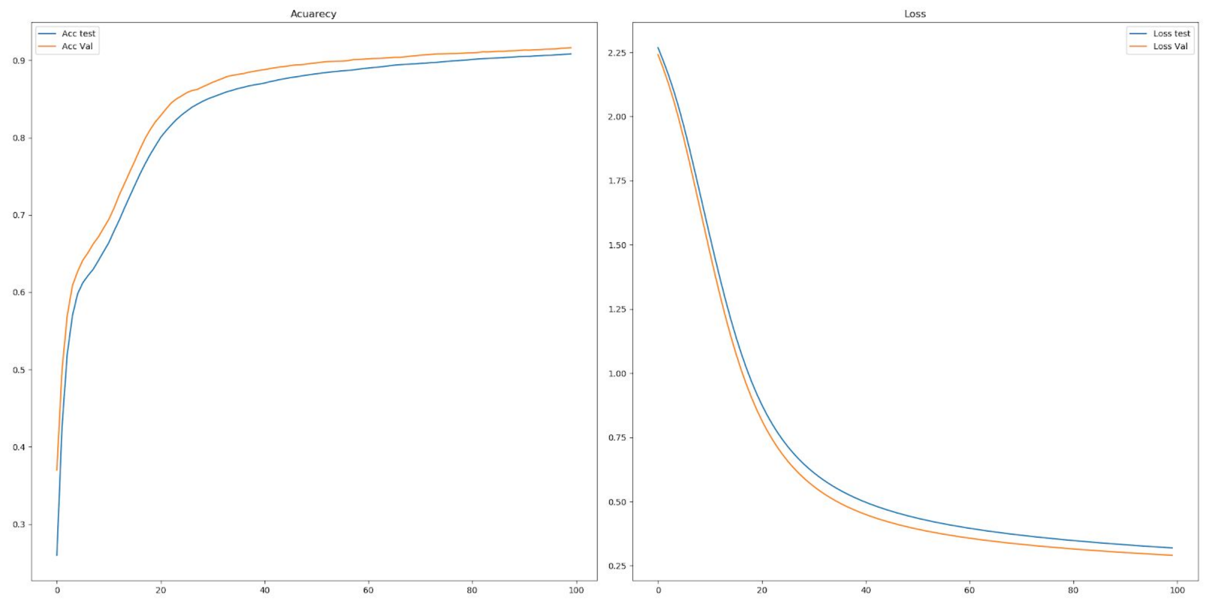




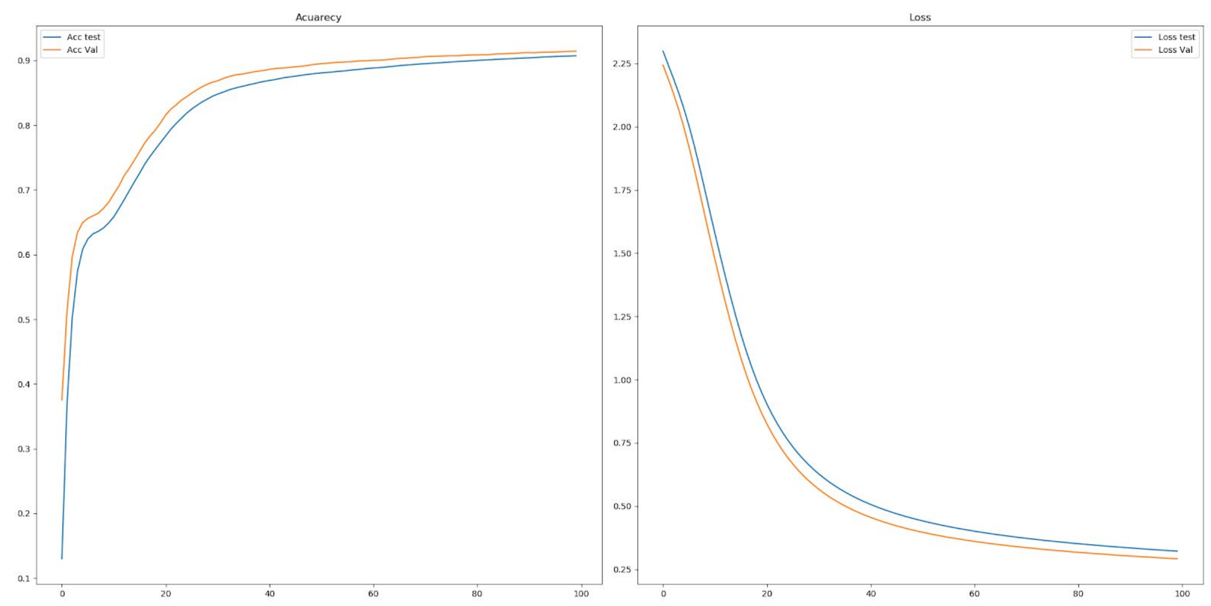
Test 1 layer relu and 1 softmax
Learning rate : 0.4
Batch size = 128



Batch size = 1024



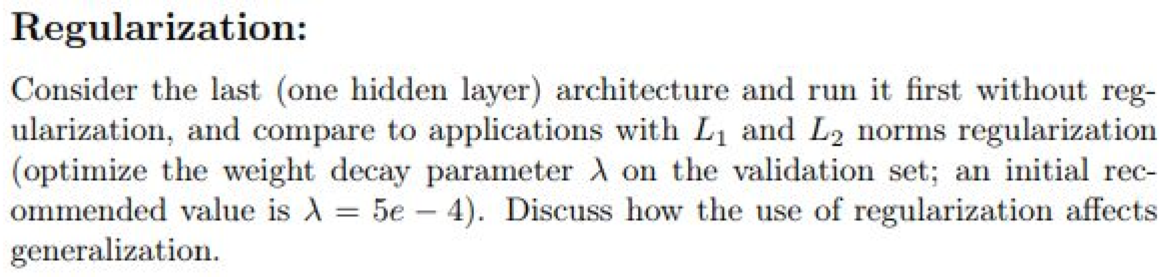
Batch 60000



Conclusion -- From the above we recognize that (as we expected) the shapes of our graphs did not change as a result of our batch size. Indeed, they all converge at the highest rate until about the 20th epoch. Moreover, they all reach around the same loss by the final epoch (~0.325) and around the same accuracy by then too (~0.9).

As for our prediction about runtime, since we used Python on our personal computers to run this experiment, our average runtime for a single epoch remained constant. This is because Python is locked to only run on a single process.

We did notice, however, that when given very large sample sizes, we would run out of memory. This problem was averted using bathes.



Regularization:

Consider the last (one hidden layer) architecture and run it first without regularization, and compare to applications with L_1 and L_2 norms regularization (optimize the weight decay parameter λ on the validation set; an initial recommended value is $\lambda = 5e - 4$). Discuss how the use of regularization affects generalization.

We recall that l1 regularization tends to bring less critical weights towards 0 while l2 regularization does not. We also recall that l2 heavily penalizes larger outliers at the expense of many smaller ones.

Hypothesis:

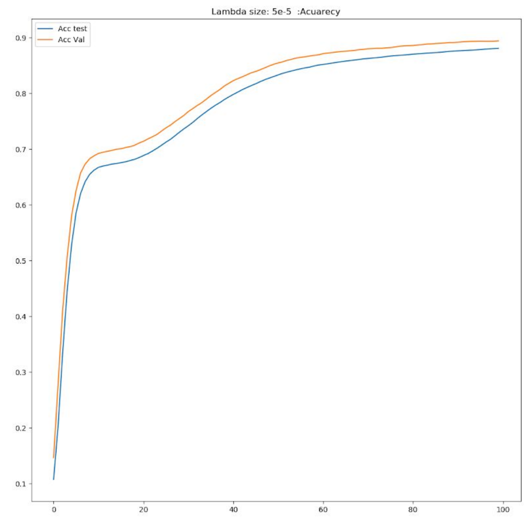
L1 regularization:

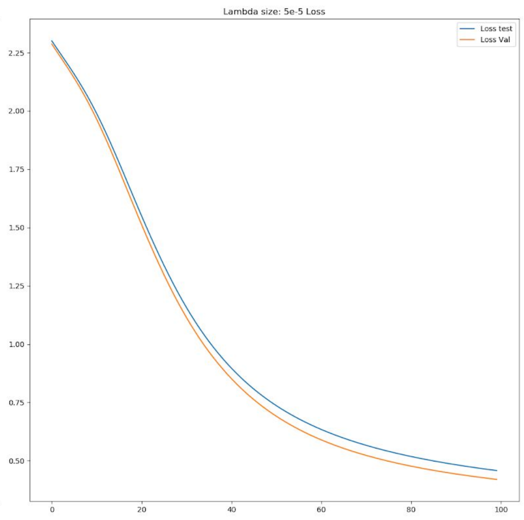
We expect that when lambda is small, the impact on the loss will be small, therefore, our initial loss and accuracy will be similar to what we found without regularization. If the weights are smaller than 1, we anticipate a slower convergence rate here than in L2 regularization. We anticipate that the general shape of the graph will be similar to that seen without regularization. If lambda is too large, we expect that we will not learn the correct function. This means a worse final loss and accuracy.

L2 regularization:

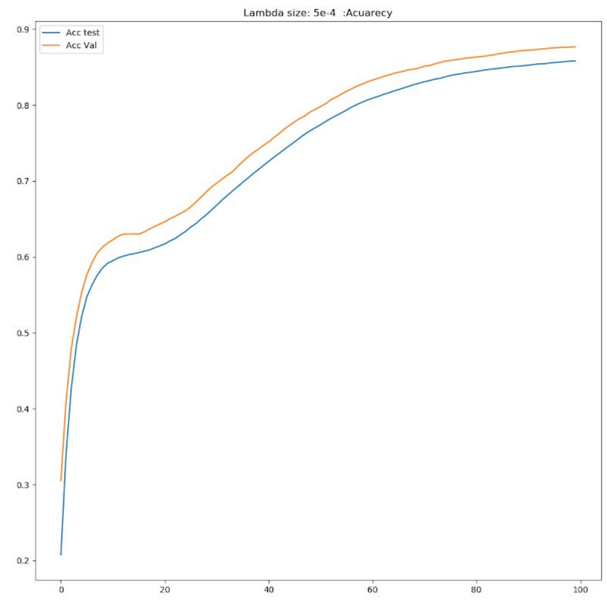
Similar to the case with L1 regularization, we expect that when lambda is small, the initial loss and accuracy will be similar to what we found without regularization. Moreover, also like in L1 regularization, if lambda is too large, we will learn the wrong problem (resulting in a lower final accuracy and loss).

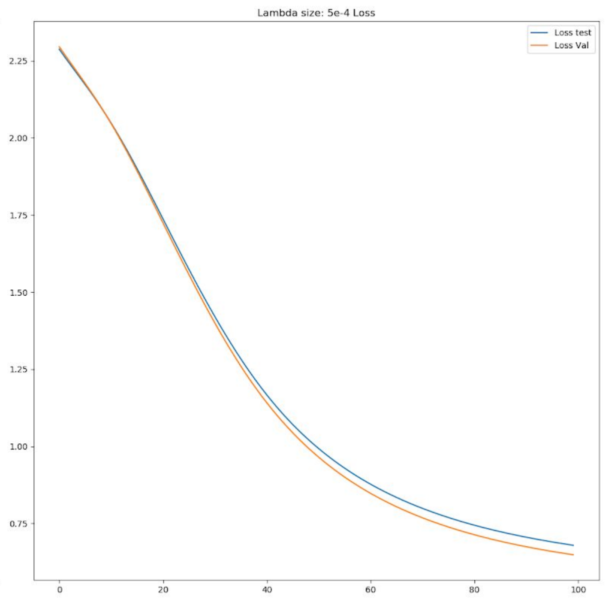
L1 learning rate: 0.2
Lambda 5e-5





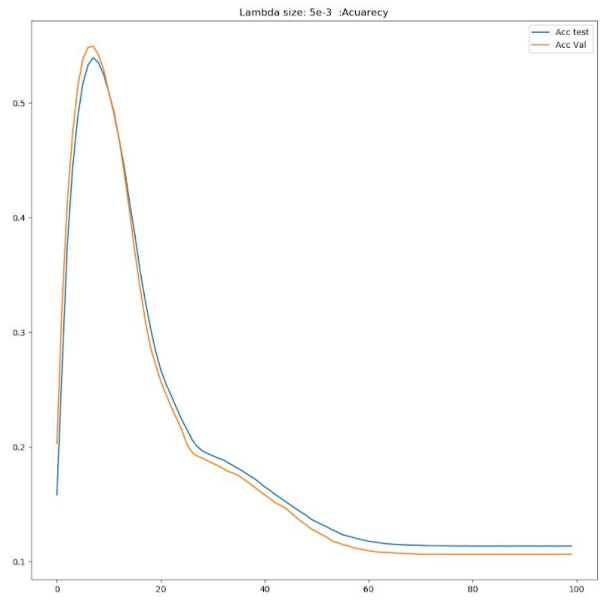
L1 regularization
Lambda 5e-4

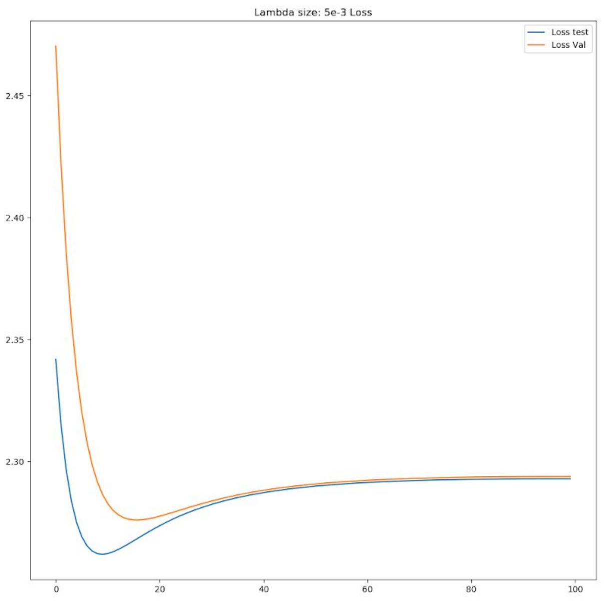




L1 regularization

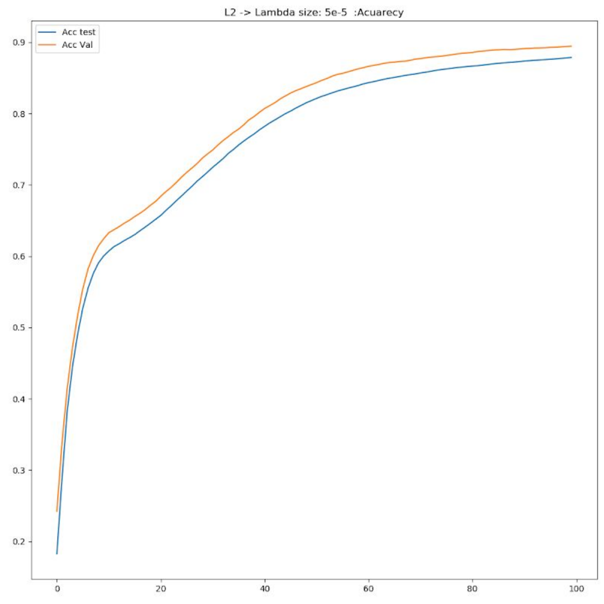
Lambda 5e-3

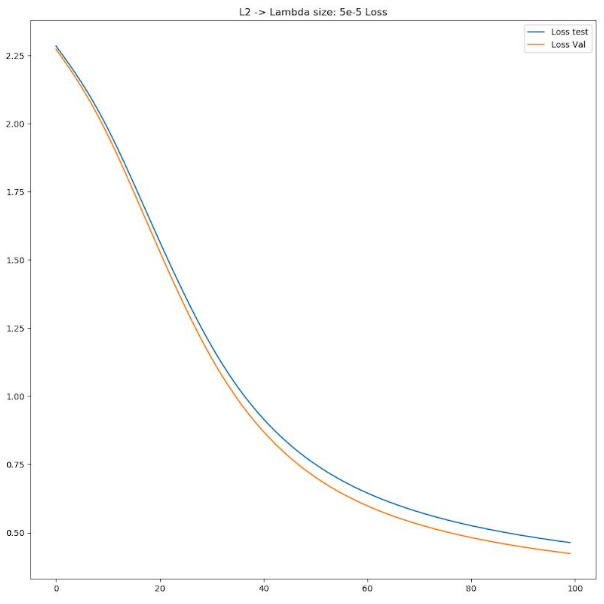




L2 , learning rate : 0.2

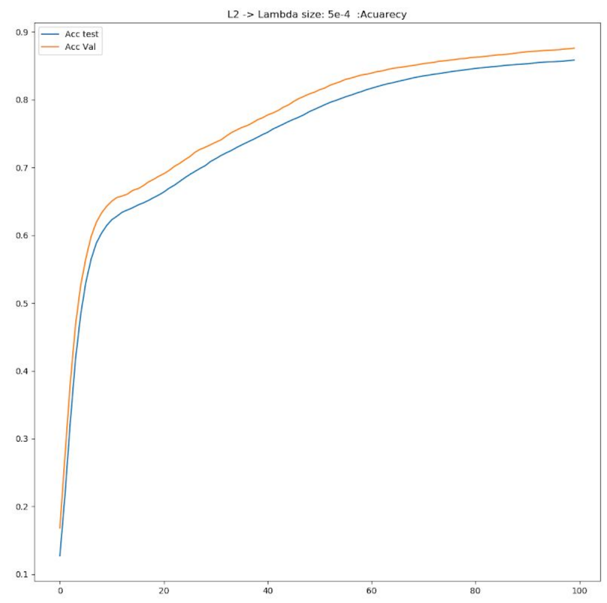
Lambda 5e-5

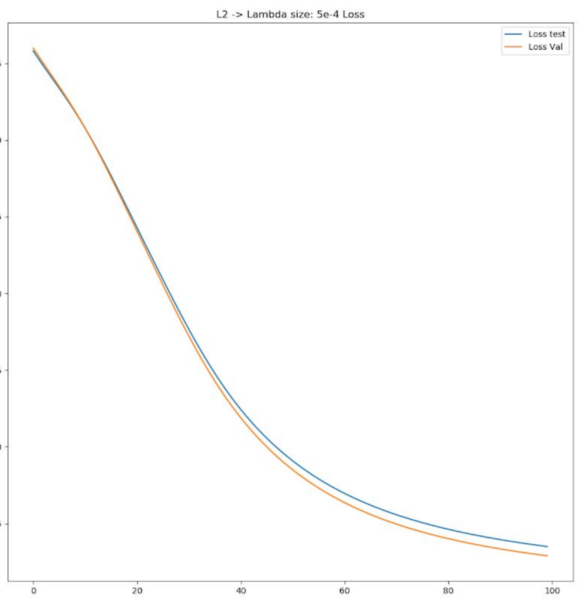




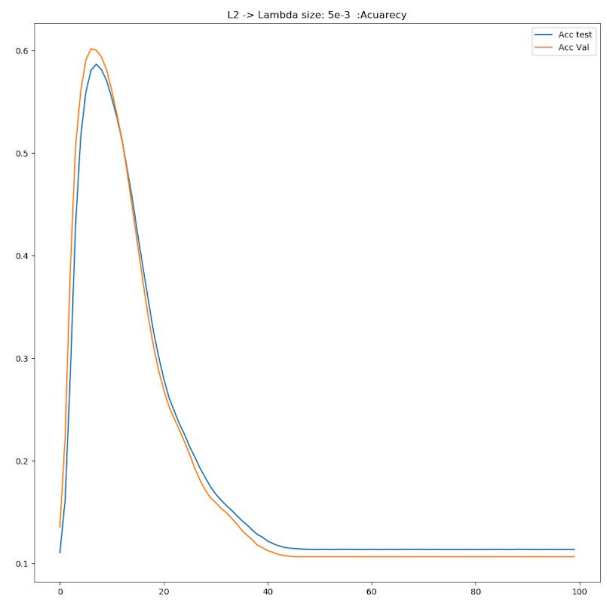
L2 regularization

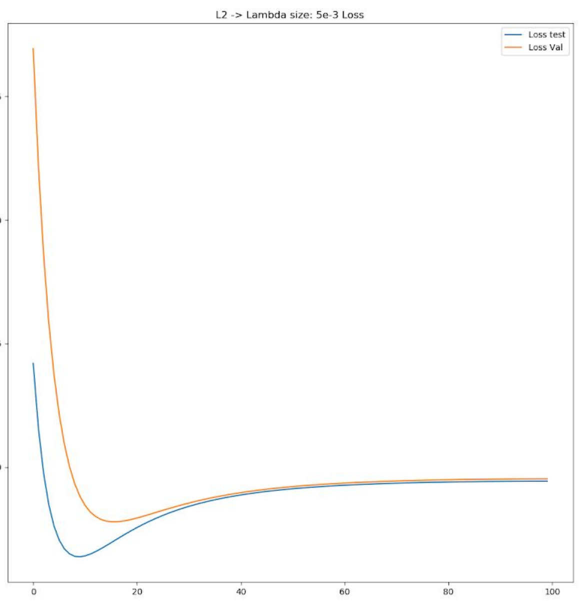
5e-4





L2 regularization
5e-3



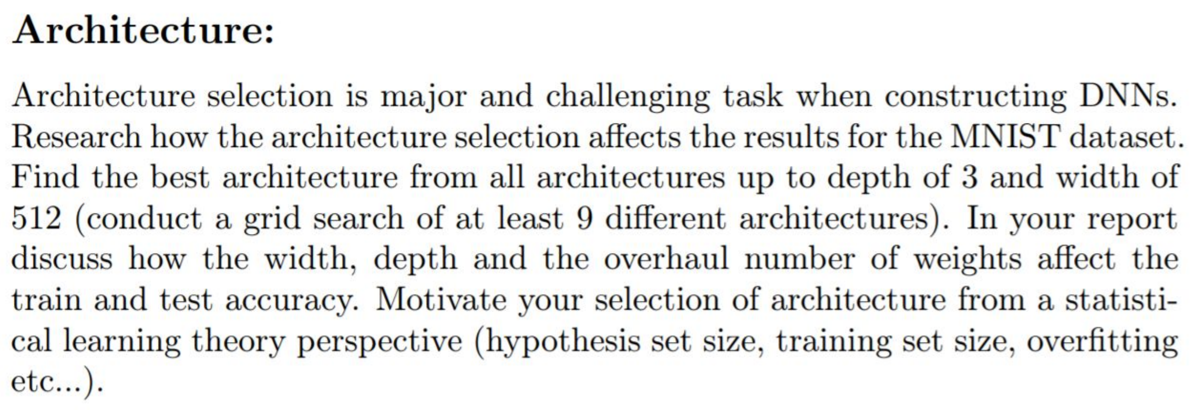


Conclusion:

We see that the best results were taken with L2 regularization and lambda values of 5e-5 and 5e-4 and with L1 regularization given a lambda value of 5e-5 and 5e-4. These graphs followed what we expected from our hypothesis.

However, the graph for L2 given a lambda value of 5e-3 was not consistent with our expectations. It rose sharply to ~60% accuracy before falling to ~%10 (nearly the same probability as random guessing). This could be because the function slowly converged towards the wrong function.

For L1 graphs, a lambda value of 5e-3 yielded results inconsistent with our hypothesis. The results and reasoning are similar to those found with a lambda value of 5e-3 in L1 regularization.

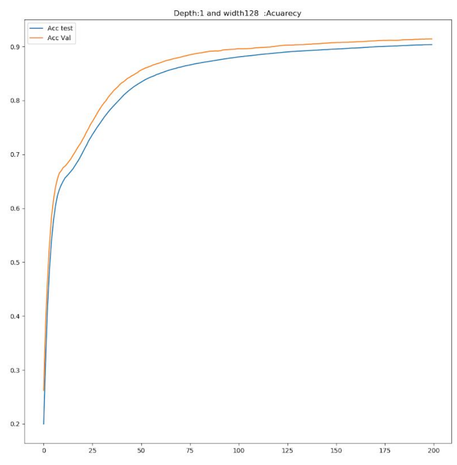


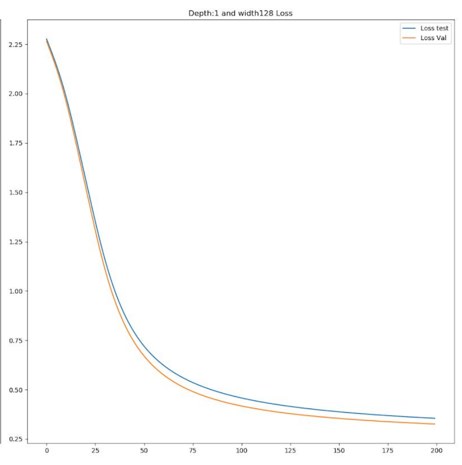
Architecture:

Architecture selection is major and challenging task when constructing DNNs. Research how the architecture selection affects the results for the MNIST dataset. Find the best architecture from all architectures up to depth of 3 and width of 512 (conduct a grid search of at least 9 different architectures). In your report discuss how the width, depth and the overhaul number of weights affect the train and test accuracy. Motivate your selection of architecture from a statistical learning theory perspective (hypothesis set size, training set size, overfitting etc...).

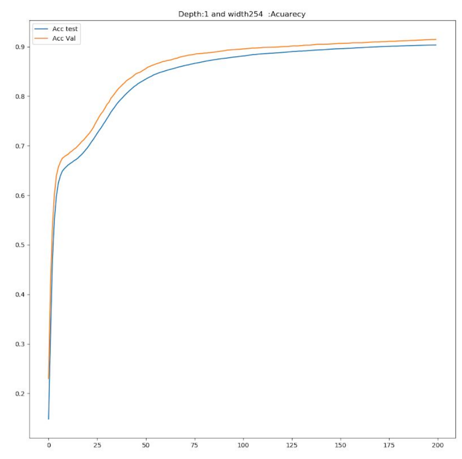
After collaborating with peers and analyzing numerous articles, we found that in deep networks, it is best to decrease layer width by a linear factor. We chose to decrease width at first by a factor of ½ and then by a factor of ¾ between one layer and the next. We found that when decreasing by a factor of ¾, our results increased significantly.

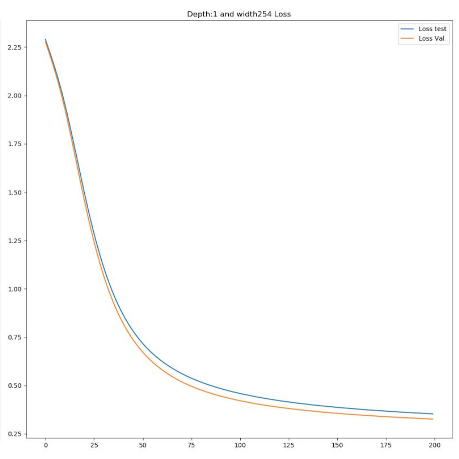
This network has a single layer of width 128



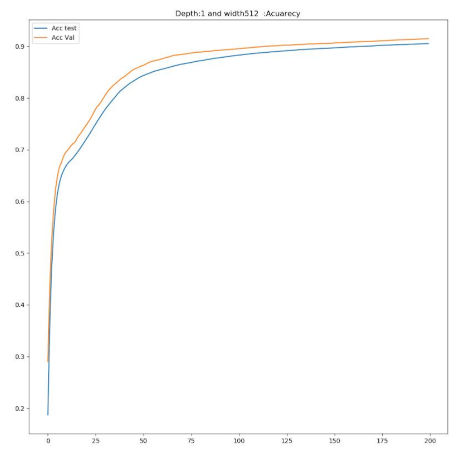


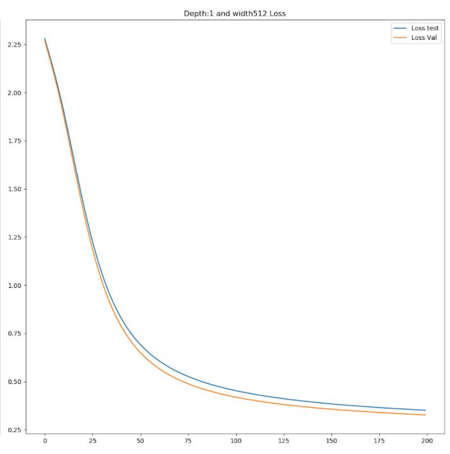
This network has a single layer of width 254





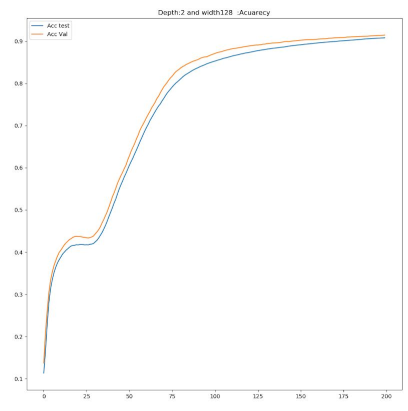
This network has a single layer of width 512

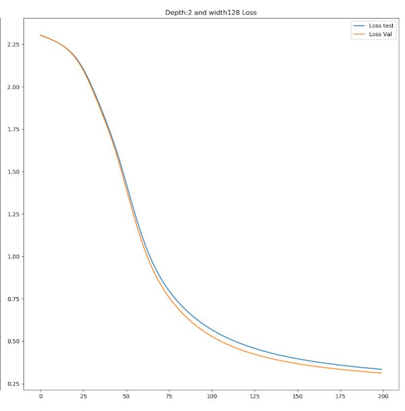




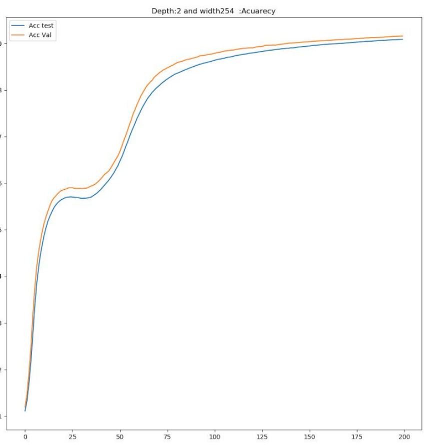
For networks of depth 1, the width did not seem to affect accuracy. They all reached ~90% accuracy by 200 epochs and had approximately the same graph shapes. The wider the layers, the longer the runtime. Accuracy was highest for width of 512, followed by 128, and finally 254.

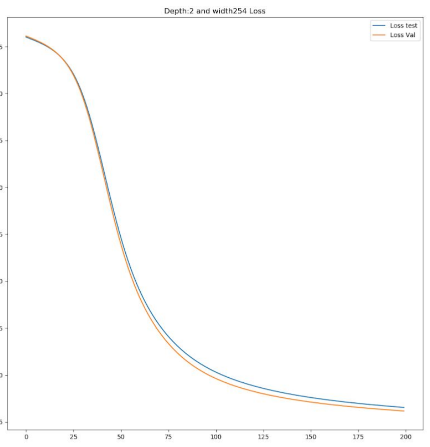
This network has 2 layers. The first one is of width 128, and the second is of width 96.



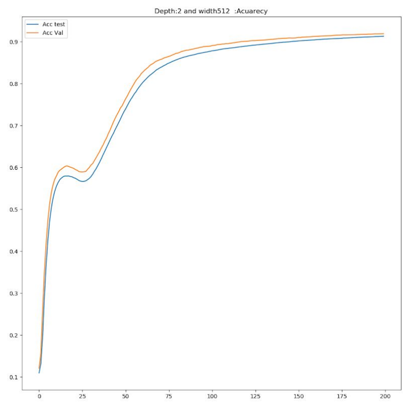


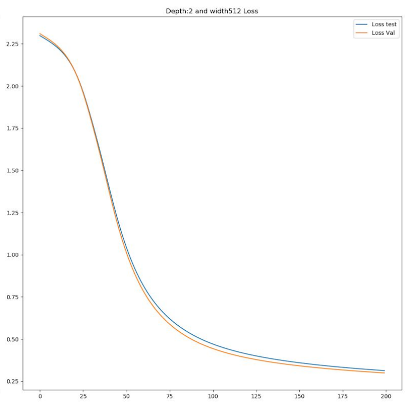
This network has 2 layers. The first one is of width 254, and the second is of width 190.





This network has 2 layers. The first one is of width 512. The second is of width 384.

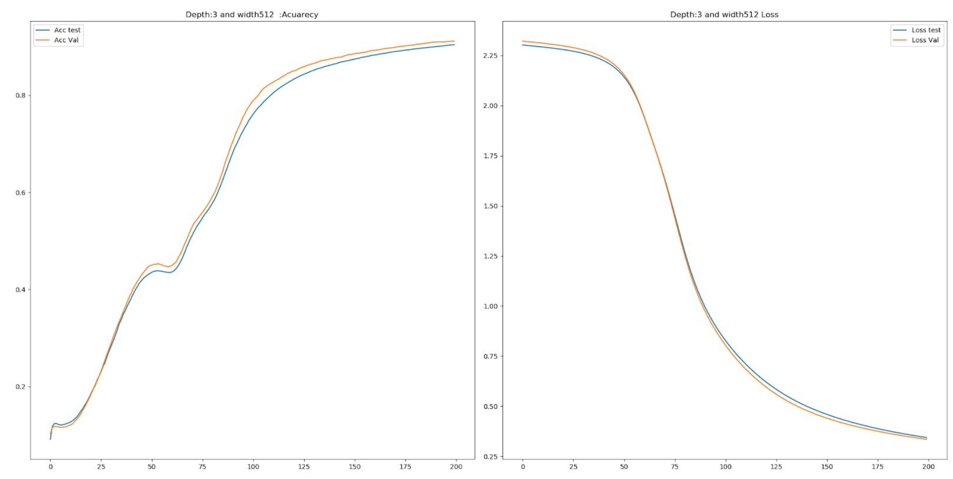




For networks of depth 2, we note all accuracies are higher than those obtained by a network which is only 1 layer deep (by ~1%). We also note that in each of these graphs, there is a slight dip in accuracy around epoch 20. This dip is more severe the wider the layer is. Nonetheless, the highest accuracy was obtained by the network with the widest layers. From the perspective of runtime, a network that is 2 layers deep has a longer runtime than a network that is 1 layer deep. The wider the layers, the longer the runtime.

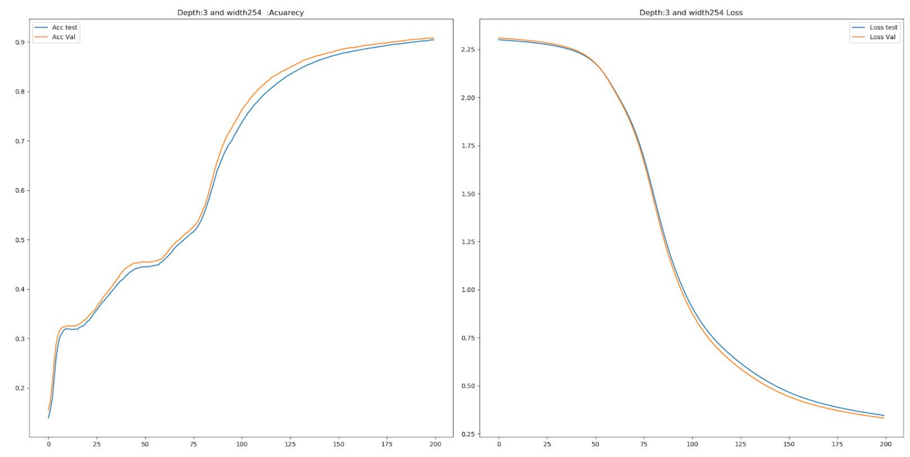
The network represented by the graph below has the following layer widths:

- 128
- 96
- 72



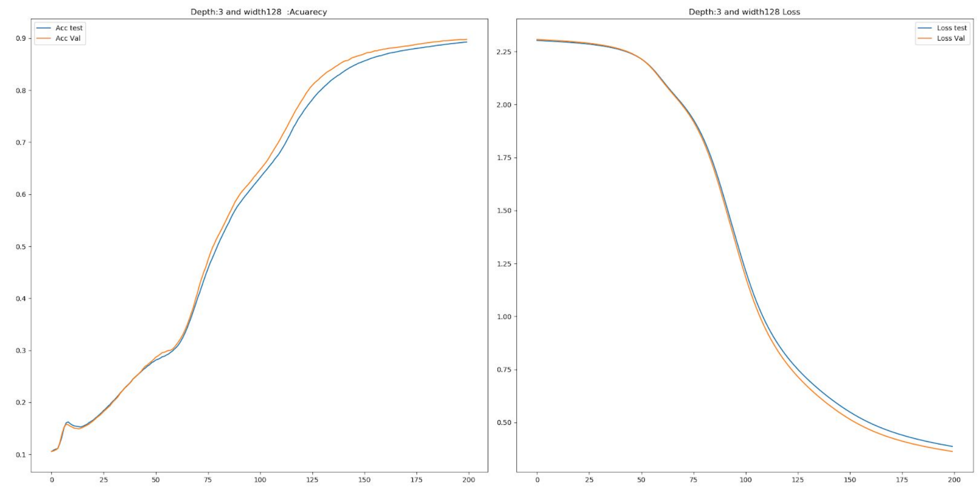
The network represented by the graph below has the following layer widths:

- 254
- 190
- 142



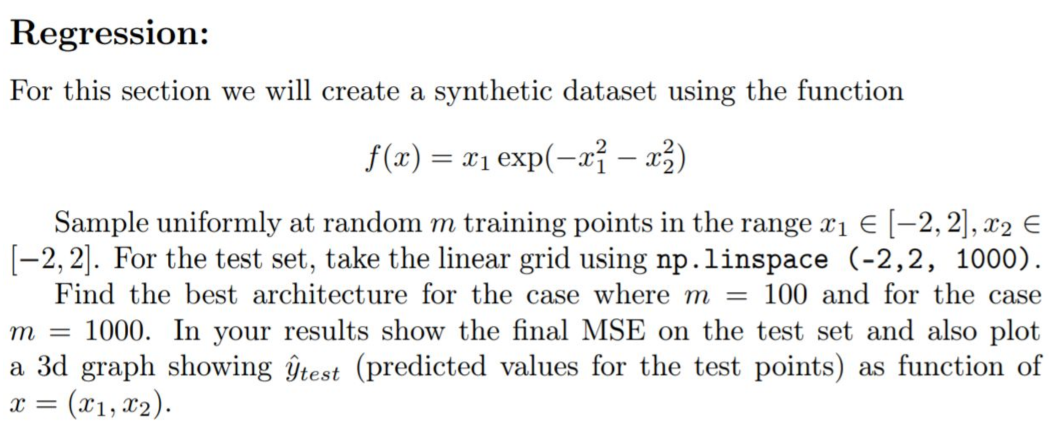
The network represented by the graph below has the following layer widths:

- 512
- 384
- 288



Given a depth of 3, the overall accuracy and loss seem to go down (even below a 1 deep network). Moreover the rate of convergence is much slower (far less steep). This is likely because the deeper the network, the more “difficult” it is to associate specific weights with loss in backprop. The best accuracy was obtained with an initial width of 254, followed by an initial weight of 512, and finally 128. Runtime for these networks is slower (but not by much than networks of depth 1 or 2/

Finally, we summarize that a network that is 2 layers deep and has widths 512 and 384 for its layers respectively obtains the best accuracy and lowest loss.



Regression:

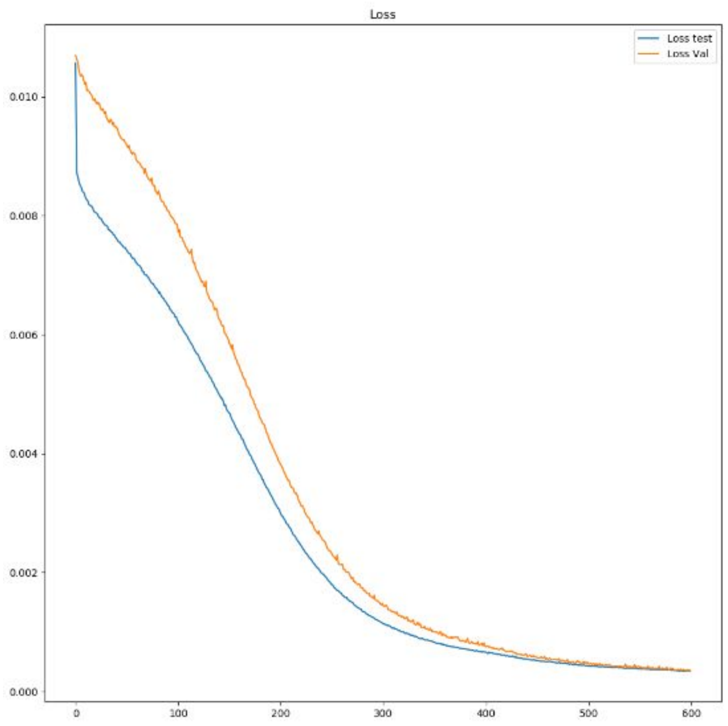
For this section we will create a synthetic dataset using the function

$$f(x) = x_1 \exp(-x_1^2 - x_2^2)$$

Sample uniformly at random m training points in the range $x_1 \in [-2, 2], x_2 \in [-2, 2]$. For the test set, take the linear grid using `np.linspace (-2,2, 1000)`.

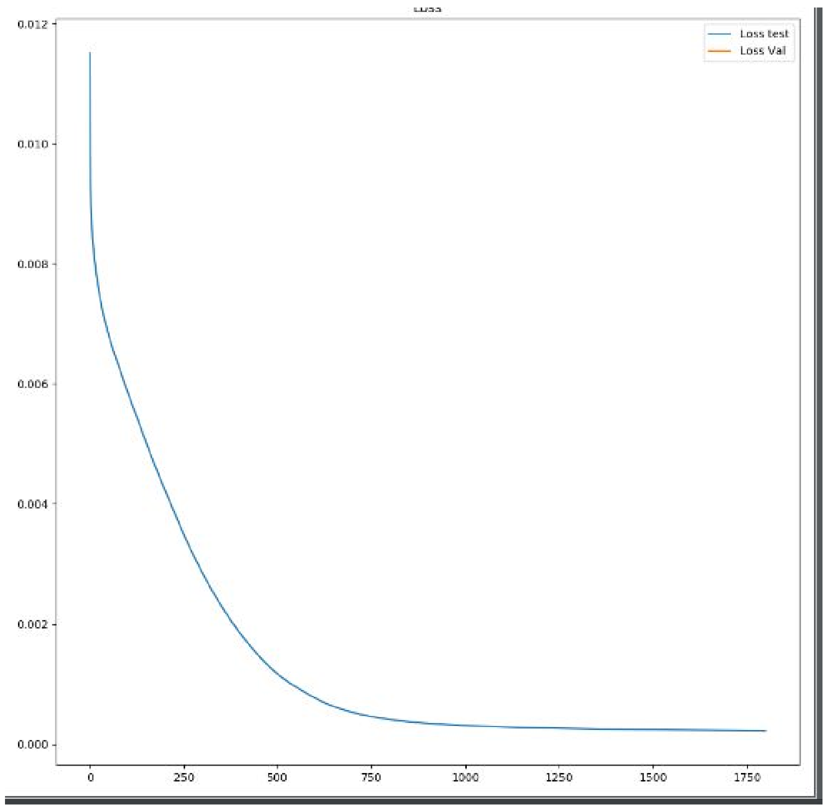
Find the best architecture for the case where $m = 100$ and for the case $m = 1000$. In your results show the final MSE on the test set and also plot a 3d graph showing $\hat{y}_{test}$ (predicted values for the test points) as function of $x = (x_1, x_2)$.

Big (data set where m=1000)



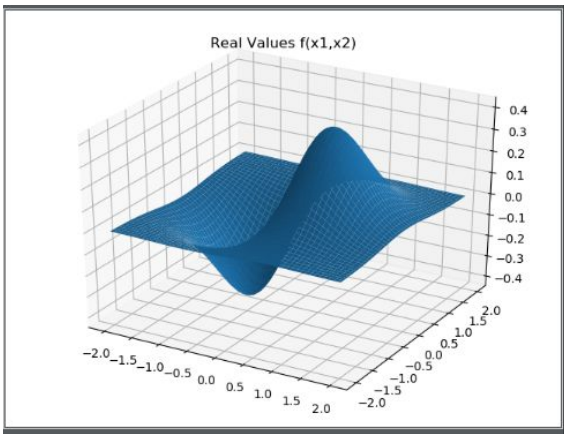
MSE Loss on test set: 0.00573

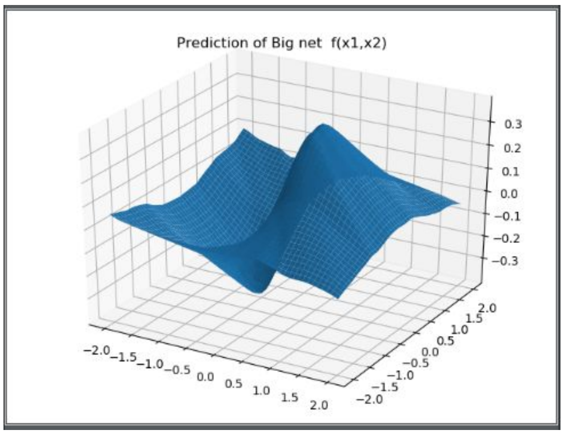
Small (data set where m=100)

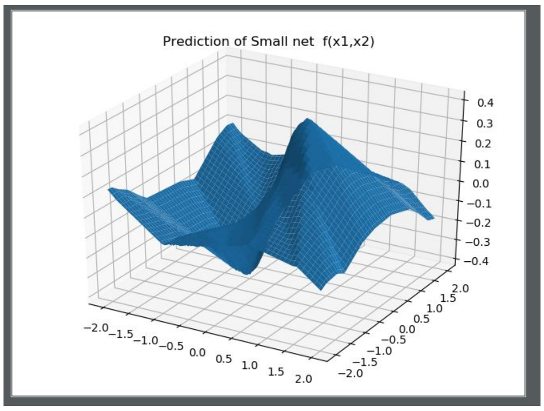


MSE Loss on test set: 0.0135

Below you will find our 3d graphs as requested for the regression problem


Real Values f(x1,x2)


Prediction of Big net f(x1,x2)


Prediction of Small net f(x1,x2)