CS 6375 – MACHINE LEARNING PROJECT 2:

Neural Networks

Yash Vijaynarayan Gupta

Data loading and pre-processing

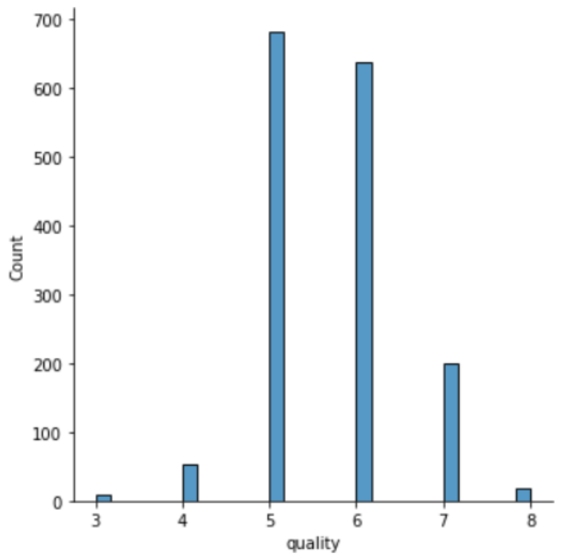
Data from UCI website Data Folder (also hosted on GitHub in the code for consistency):

`https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv`

The data has been pre-processed using following steps to avoid noise that can spoil the model or even result in overfitting. The Wine Quality dataset has a target of wine quality from [1-10] based on various attributes for which we have calculated the parameters or weights like tuning the knobs of our network for desired finer results.

Removal of Nan values - No such values found to be removed in the dataset that can create noise.

Encoding of Categorical data - Encoding of target variable was done using One Hot encoding and data was scaled using Standard Scaler.

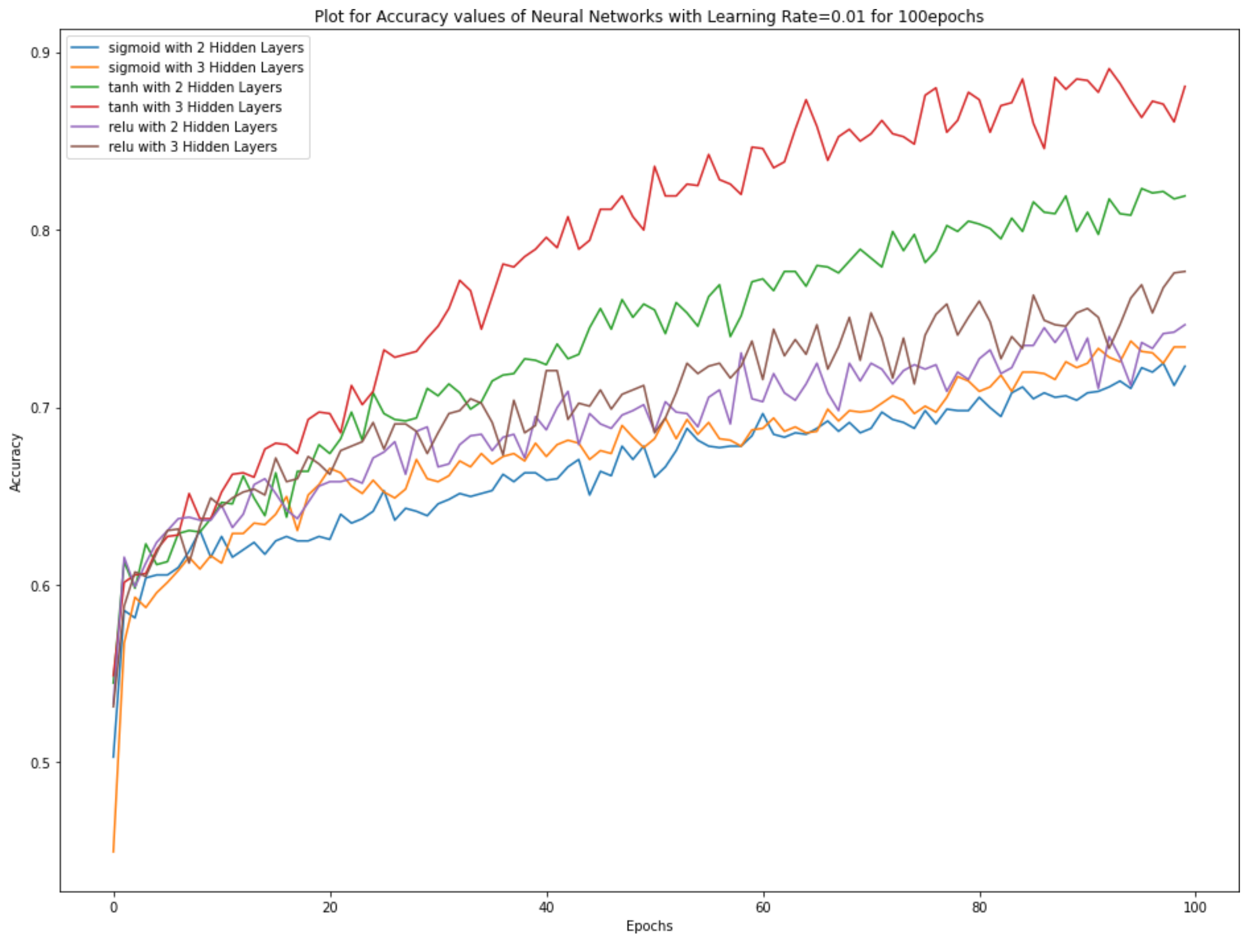


We were able to see that the graph was bell shaped and most of the wines had a score of 5 or 6.

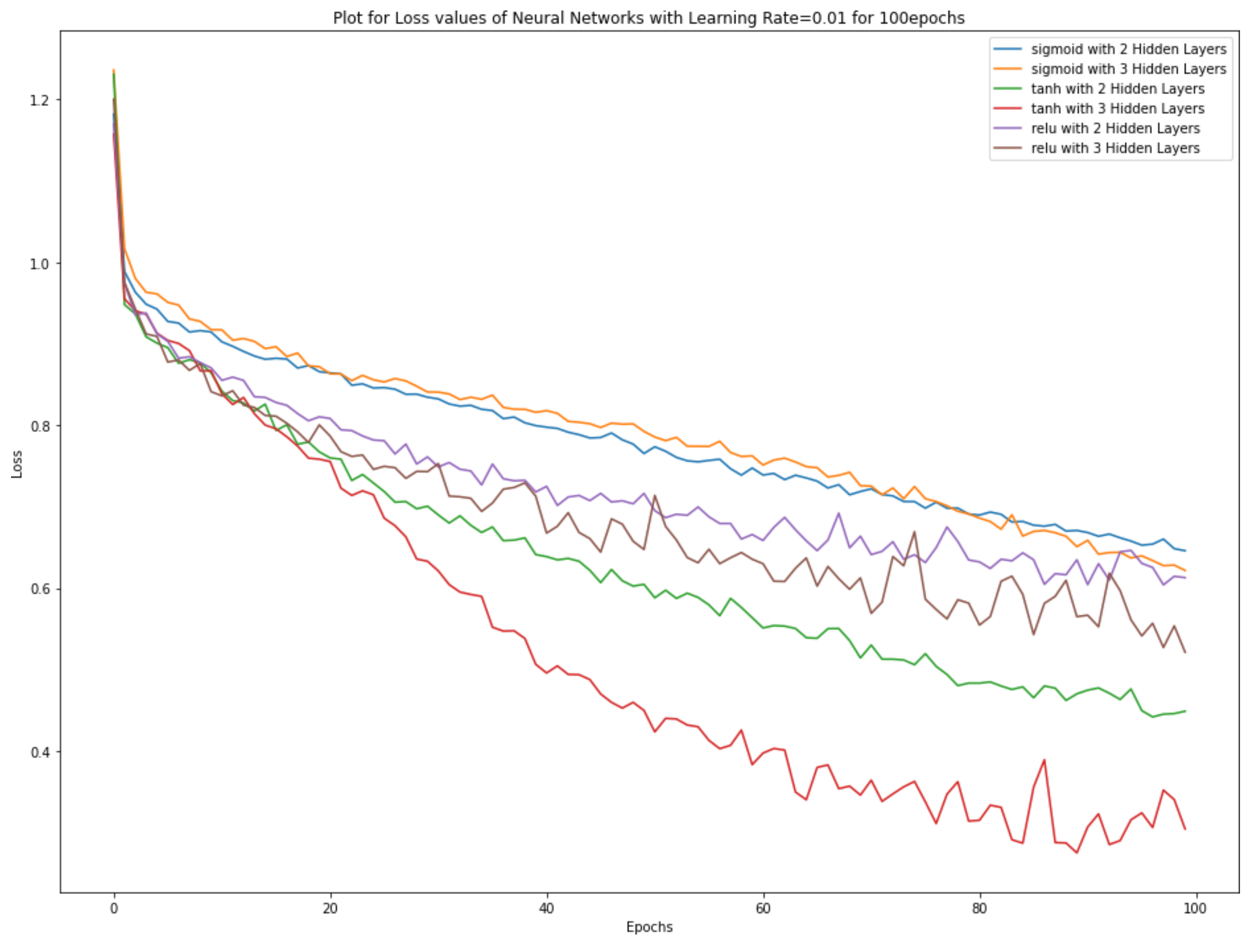
Model history Accuracy and loss for training data plots:

1. **LR = 0.01**
   Epochs = 100

Accuracy plot for each iteration



Loss for each iteration

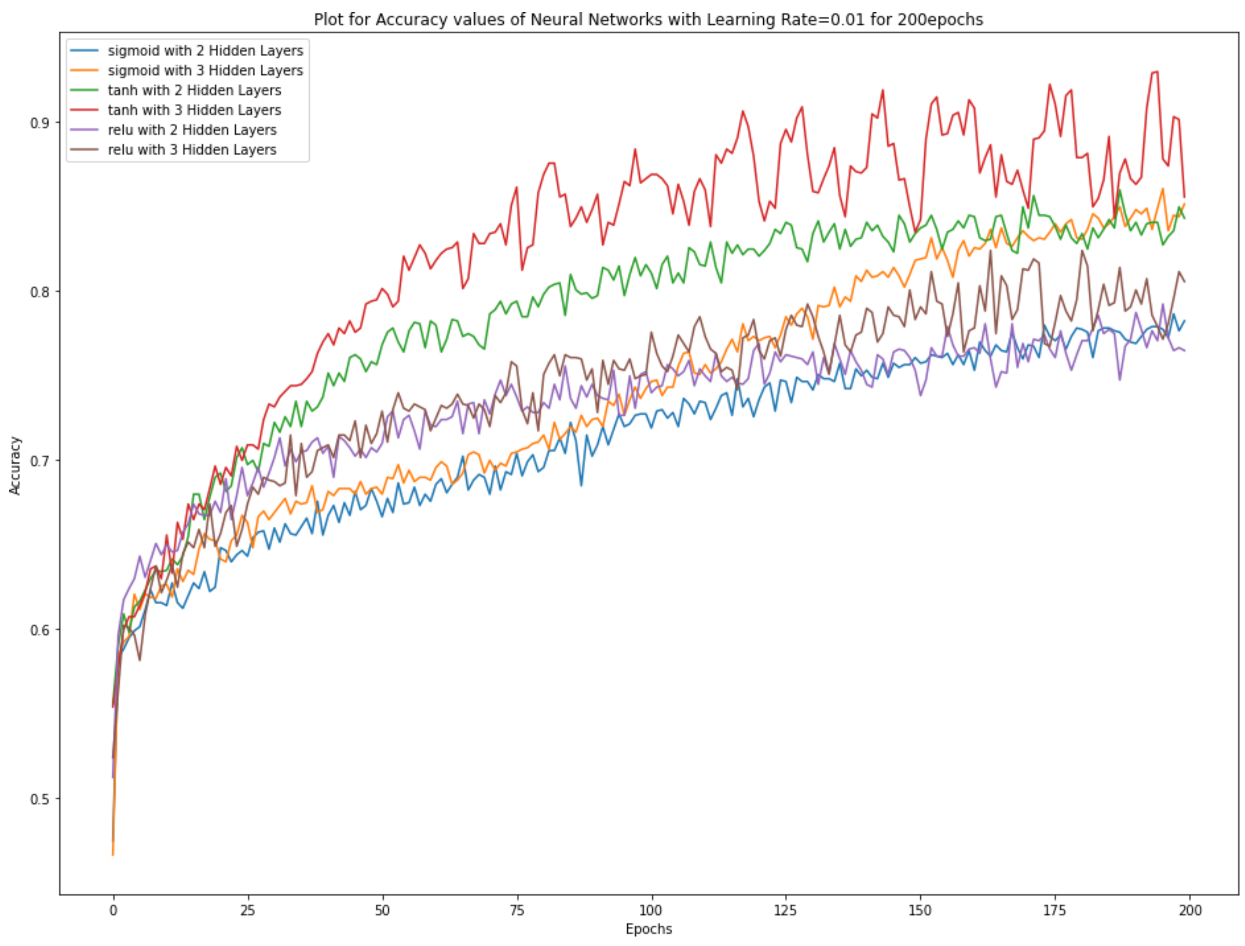


2.

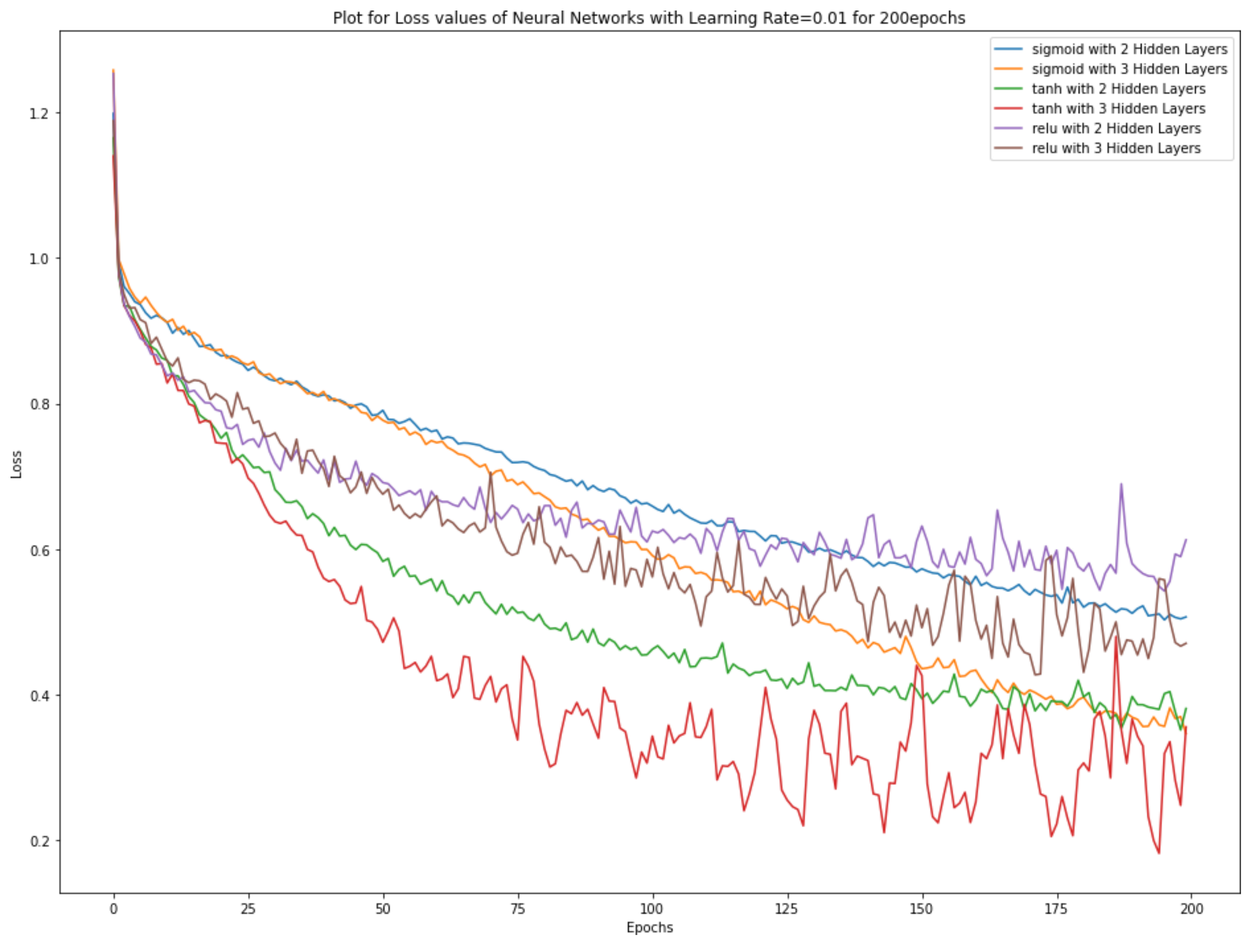
LR = 0.01

Epochs = 200

Accuracy plot for each iteration

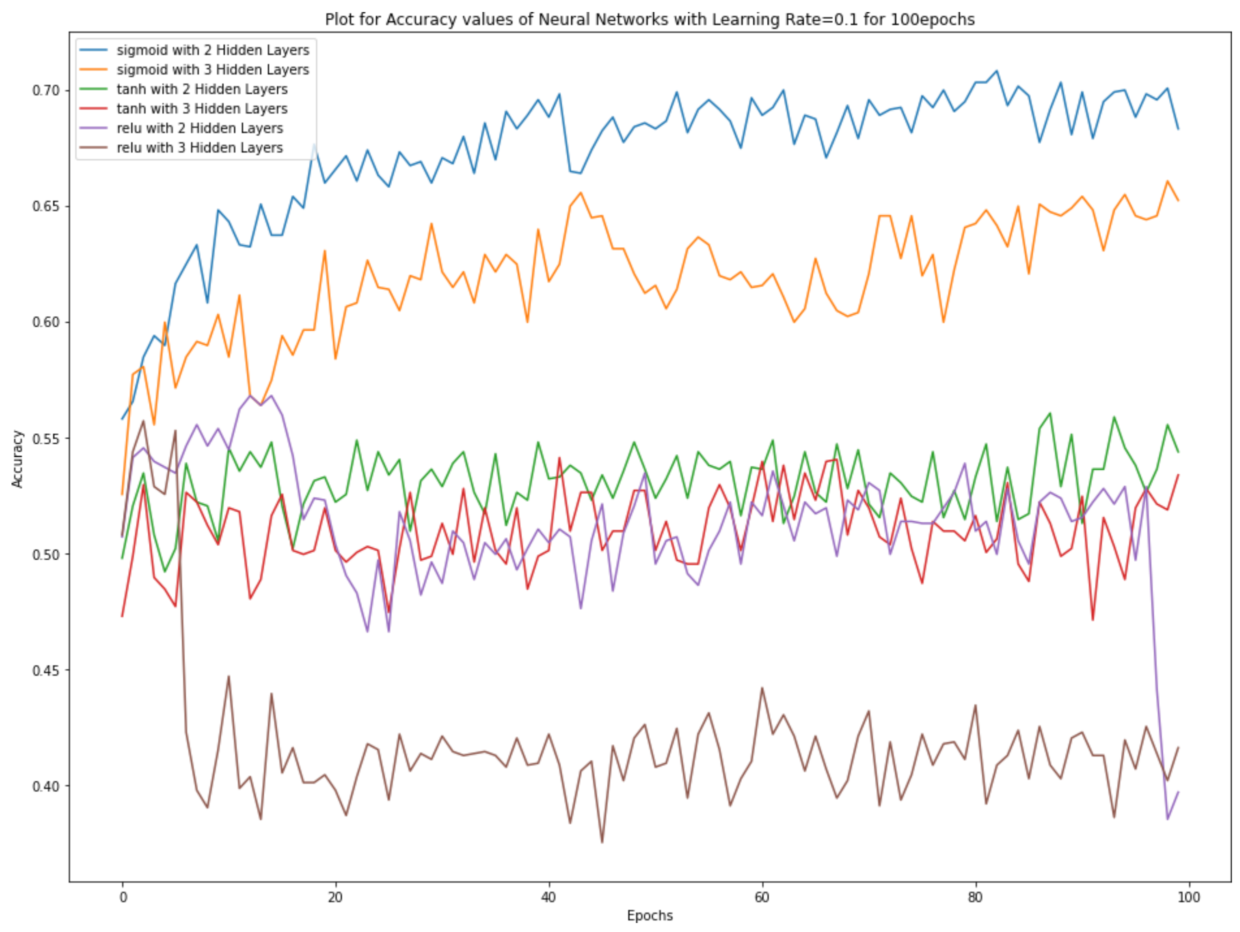


Loss for each iteration

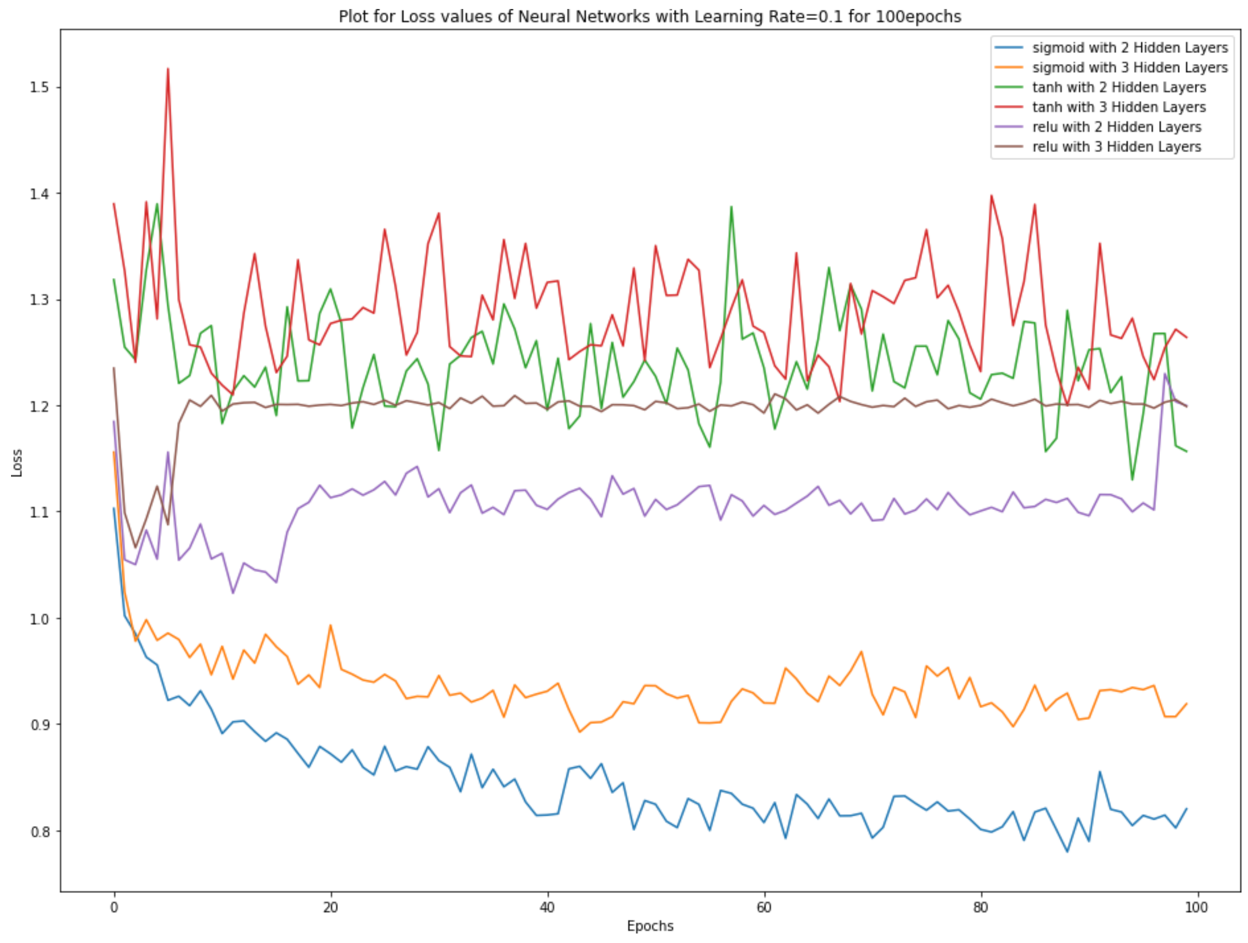


3. **LR = 0.1**
Epochs = 100

Accuracy plot for each iteration

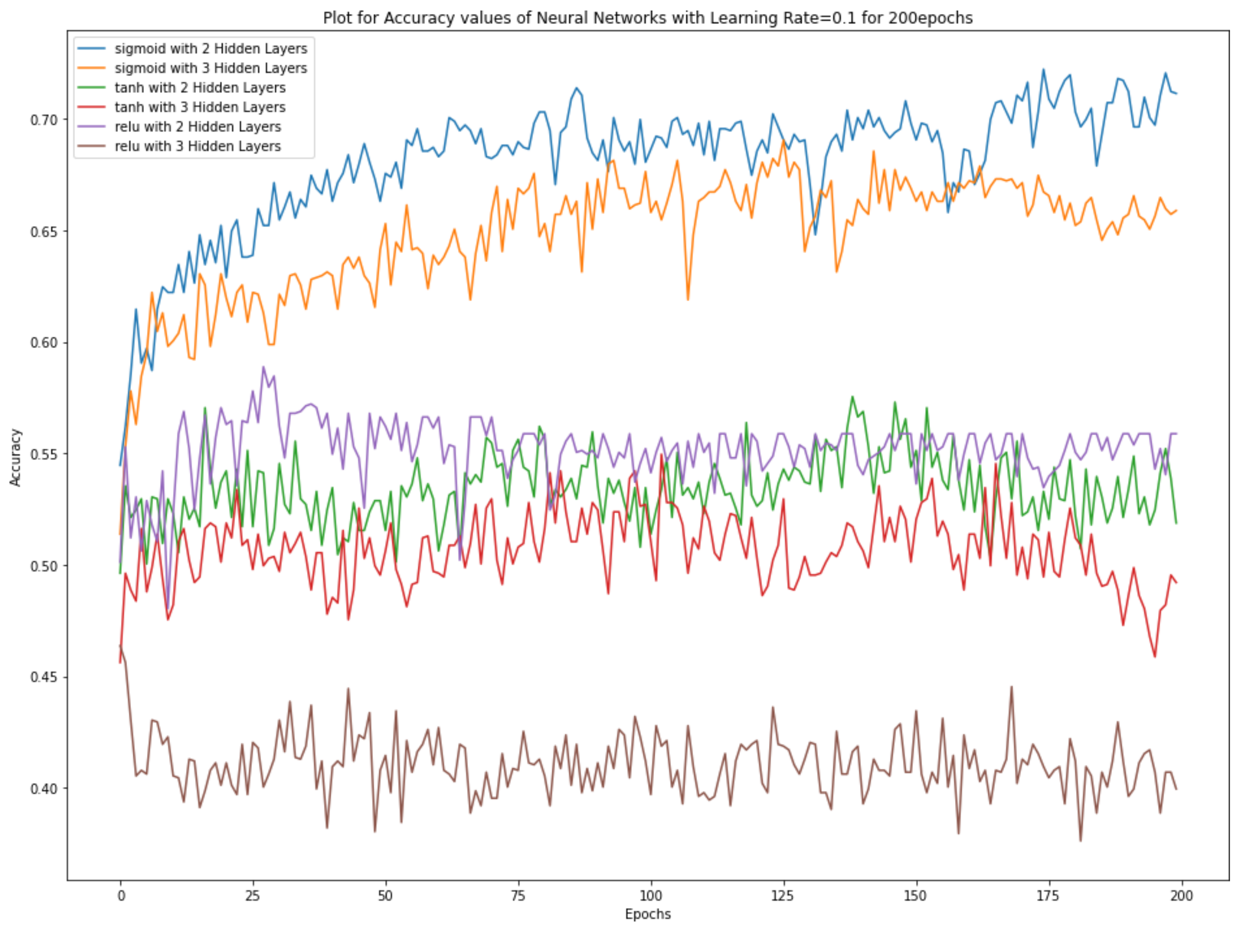


Loss for each iteration

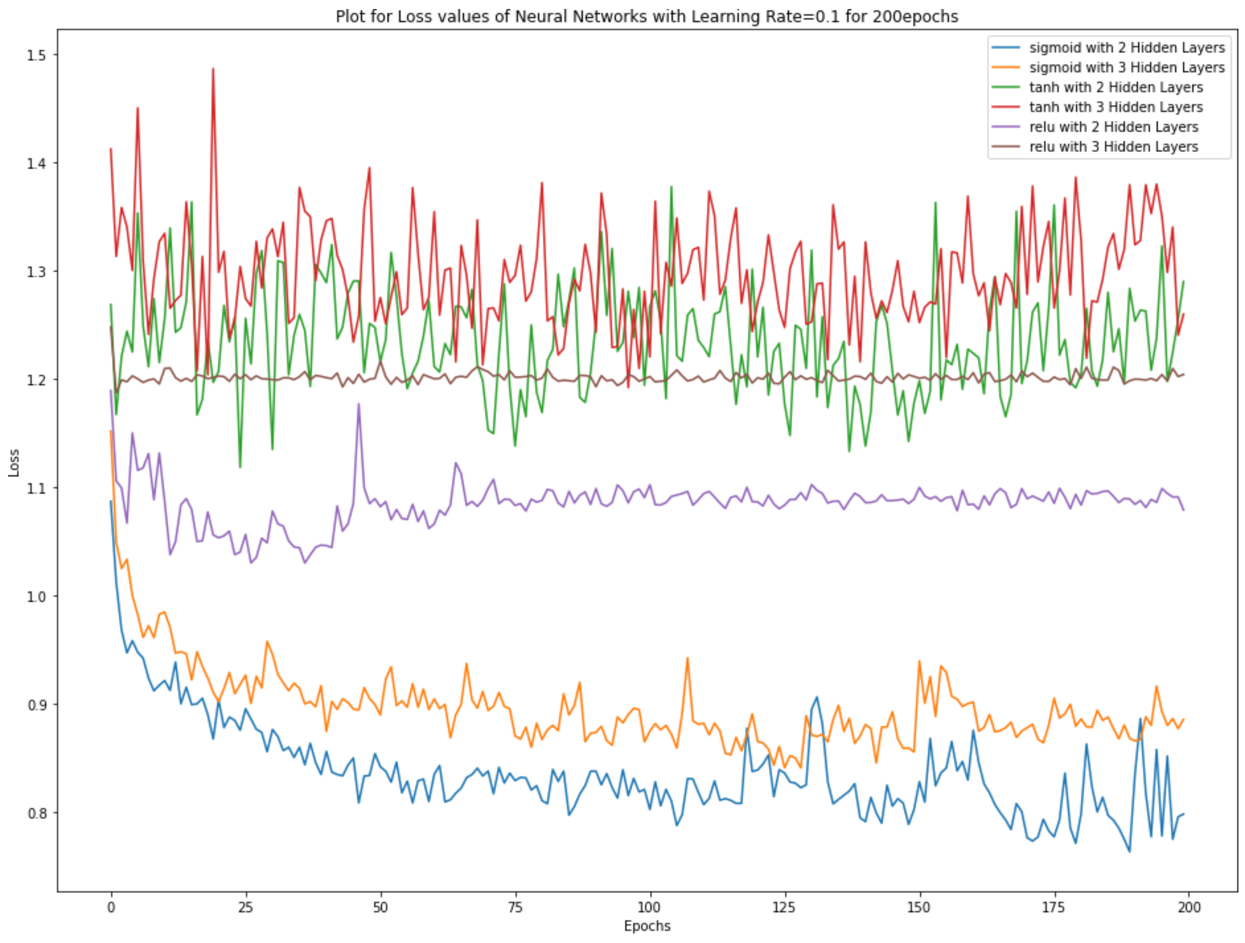


4. LR = 0.1
Epochs = 200

Accuracy plot for each iteration

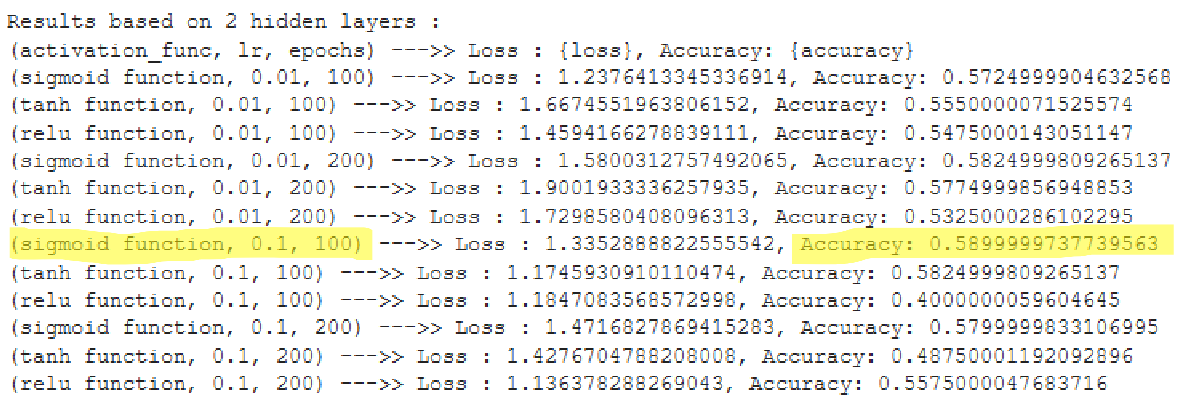


Loss for each iteration



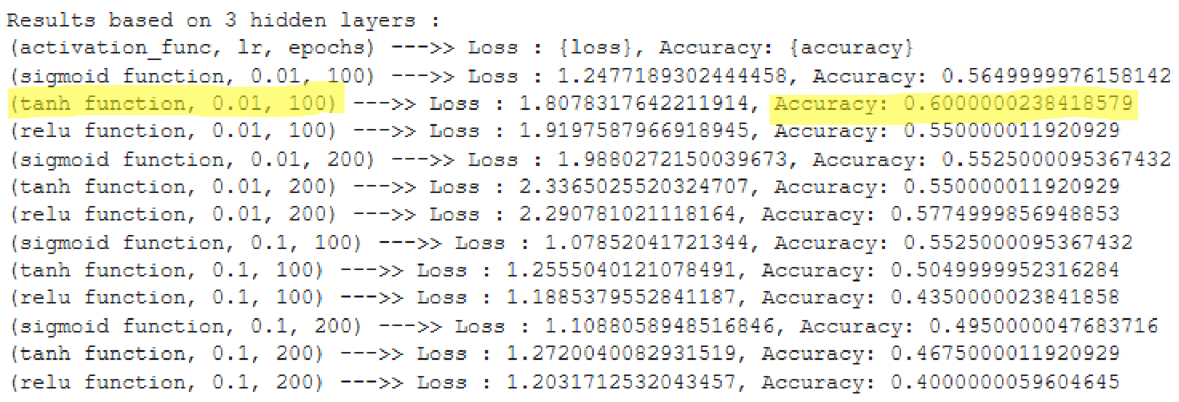
Output of Dataset on Test data:

1. Based on 2 hidden layers:



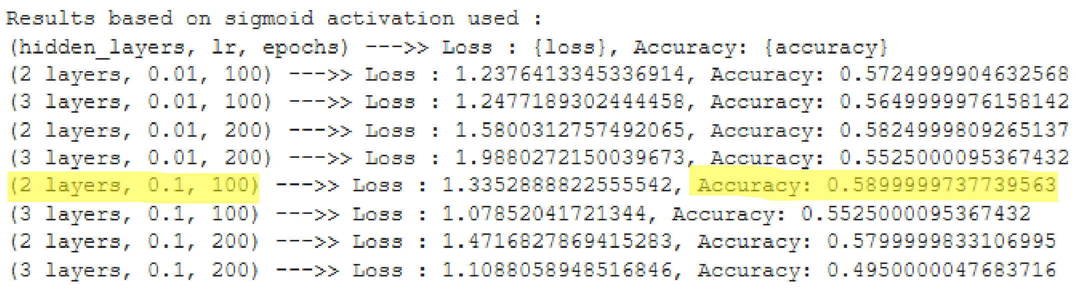
```
Results based on 2 hidden layers :
(activation_func, lr, epochs) --->> Loss : {loss}, Accuracy: {accuracy}
(sigmoid function, 0.01, 100) --->> Loss : 1.2376413345336914, Accuracy: 0.5724999904632568
(tanh function, 0.01, 100) --->> Loss : 1.6674551963806152, Accuracy: 0.5550000071525574
(relu function, 0.01, 100) --->> Loss : 1.4594166278839111, Accuracy: 0.5475000143051147
(sigmoid function, 0.01, 200) --->> Loss : 1.5800312757492065, Accuracy: 0.5824999809265137
(tanh function, 0.01, 200) --->> Loss : 1.9001933336257935, Accuracy: 0.5774999856948853
(relu function, 0.01, 200) --->> Loss : 1.7298580408096313, Accuracy: 0.5325000286102295
(sigmoid function, 0.1, 100) --->> Loss : 1.3352888822555542, Accuracy: 0.5899999737739563
(tanh function, 0.1, 100) --->> Loss : 1.1745930910110474, Accuracy: 0.5824999809265137
(relu function, 0.1, 100) --->> Loss : 1.1847083568572998, Accuracy: 0.4000000059604645
(sigmoid function, 0.1, 200) --->> Loss : 1.4716827869415283, Accuracy: 0.5799999833106995
(tanh function, 0.1, 200) --->> Loss : 1.4276704788208008, Accuracy: 0.48750001192092896
(relu function, 0.1, 200) --->> Loss : 1.136378288269043, Accuracy: 0.5575000047683716
```

2. Based on 3 hidden layers:



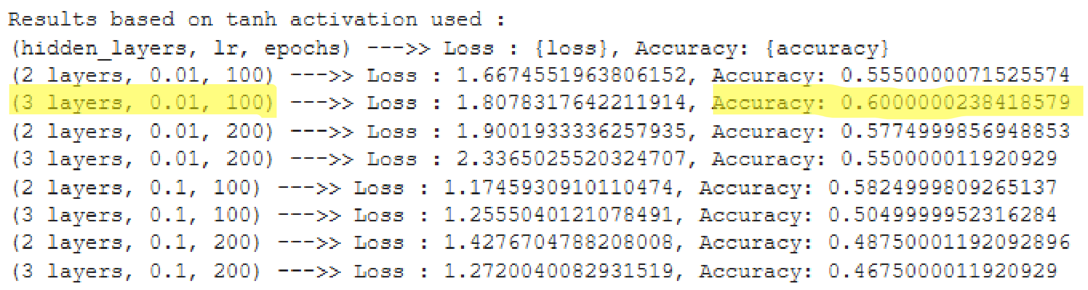
```
Results based on 3 hidden layers :
(activation_func, lr, epochs) --->> Loss : {loss}, Accuracy: {accuracy}
(sigmoid function, 0.01, 100) --->> Loss : 1.2477189302444458, Accuracy: 0.5649999976158142
(tanh function, 0.01, 100) --->> Loss : 1.8078317642211914, Accuracy: 0.6000000238418579
(relu function, 0.01, 100) --->> Loss : 1.9197587966918945, Accuracy: 0.550000011920929
(sigmoid function, 0.01, 200) --->> Loss : 1.9880272150039673, Accuracy: 0.5525000095367432
(tanh function, 0.01, 200) --->> Loss : 2.3365025520324707, Accuracy: 0.550000011920929
(relu function, 0.01, 200) --->> Loss : 2.290781021118164, Accuracy: 0.5774999856948853
(sigmoid function, 0.1, 100) --->> Loss : 1.07852041721344, Accuracy: 0.5525000095367432
(tanh function, 0.1, 100) --->> Loss : 1.2555040121078491, Accuracy: 0.5049999952316284
(relu function, 0.1, 100) --->> Loss : 1.1885379552841187, Accuracy: 0.4350000023841858
(sigmoid function, 0.1, 200) --->> Loss : 1.1088058948516846, Accuracy: 0.4950000047683716
(tanh function, 0.1, 200) --->> Loss : 1.2720040082931519, Accuracy: 0.4675000011920929
(relu function, 0.1, 200) --->> Loss : 1.2031712532043457, Accuracy: 0.4000000059604645
```

3. Based on Sigmoid Activation function:



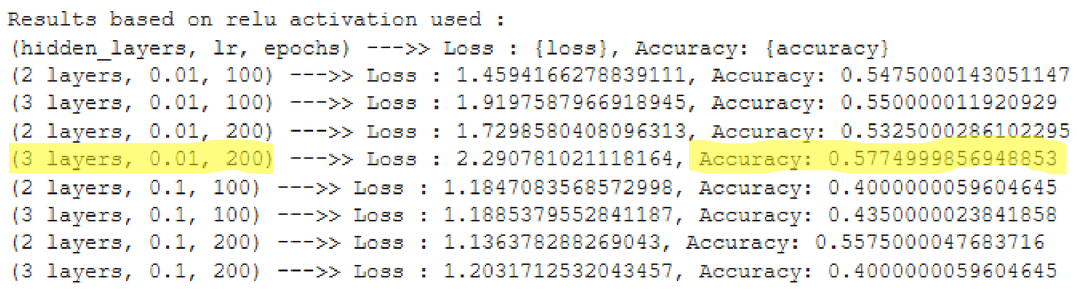
```
Results based on sigmoid activation used :
(hidden_layers, lr, epochs) --->> Loss : {loss}, Accuracy: {accuracy}
(2 layers, 0.01, 100) --->> Loss : 1.2376413345336914, Accuracy: 0.5724999904632568
(3 layers, 0.01, 100) --->> Loss : 1.2477189302444458, Accuracy: 0.5649999976158142
(2 layers, 0.01, 200) --->> Loss : 1.5800312757492065, Accuracy: 0.5824999809265137
(3 layers, 0.01, 200) --->> Loss : 1.9880272150039673, Accuracy: 0.5525000095367432
(2 layers, 0.1, 100) --->> Loss : 1.3352888822555542, Accuracy: 0.5899999737739563
(3 layers, 0.1, 100) --->> Loss : 1.07852041721344, Accuracy: 0.5525000095367432
(2 layers, 0.1, 200) --->> Loss : 1.4716827869415283, Accuracy: 0.5799999833106995
(3 layers, 0.1, 200) --->> Loss : 1.1088058948516846, Accuracy: 0.4950000047683716
```

4. Based on tanh Activation function:



```
Results based on tanh activation used :
(hidden_layers, lr, epochs) --->> Loss : {loss}, Accuracy: {accuracy}
(2 layers, 0.01, 100) --->> Loss : 1.6674551963806152, Accuracy: 0.5550000071525574
(3 layers, 0.01, 100) --->> Loss : 1.8078317642211914, Accuracy: 0.6000000238418579
(2 layers, 0.01, 200) --->> Loss : 1.9001933336257935, Accuracy: 0.5774999856948853
(3 layers, 0.01, 200) --->> Loss : 2.3365025520324707, Accuracy: 0.550000011920929
(2 layers, 0.1, 100) --->> Loss : 1.1745930910110474, Accuracy: 0.5824999809265137
(3 layers, 0.1, 100) --->> Loss : 1.2555040121078491, Accuracy: 0.5049999952316284
(2 layers, 0.1, 200) --->> Loss : 1.4276704788208008, Accuracy: 0.48750001192092896
(3 layers, 0.1, 200) --->> Loss : 1.2720040082931519, Accuracy: 0.4675000011920929
```

5. Based on relu Activation function:



```
Results based on relu activation used :
(hidden_layers, lr, epochs) --->> Loss : {loss}, Accuracy: {accuracy}
(2 layers, 0.01, 100) --->> Loss : 1.4594166278839111, Accuracy: 0.5475000143051147
(3 layers, 0.01, 100) --->> Loss : 1.9197587966918945, Accuracy: 0.550000011920929
(2 layers, 0.01, 200) --->> Loss : 1.7298580408096313, Accuracy: 0.5325000286102295
(3 layers, 0.01, 200) --->> Loss : 2.290781021118164, Accuracy: 0.5774999856948853
(2 layers, 0.1, 100) --->> Loss : 1.1847083568572998, Accuracy: 0.4000000059604645
(3 layers, 0.1, 100) --->> Loss : 1.1885379552841187, Accuracy: 0.4350000023841858
(2 layers, 0.1, 200) --->> Loss : 1.136378288269043, Accuracy: 0.5575000047683716
(3 layers, 0.1, 200) --->> Loss : 1.2031712532043457, Accuracy: 0.4000000059604645
```

Based on the results:

We can see that the best results were found on:
Tanh as the activation function
Number of layers: 3
Learning Rate = 0.01
Epochs – 100

Conclusion

We have a total of 4 pair of subplots for loss and accuracy where comparison subplots are grouped based on learning rate and epochs because comparing results of different learning rates and max iterations won't make much sense. Hence, grouping done.
Also, the structure of network including activation functions used in every hyper parameter combination (total of 24 combinations) was mostly consistent to observe the changes resultant from hyper-parameter variations much clearly.

Since our aim was to study the behavior of network on the variations of the 4 mentioned hyper-parameters, we can draw some of the following conclusions:

- Learning rate plays a very important role and we believe that diminishing this hyper-parameter gives much better smooth results asevident from loss and accuracy plots for lr=0.01
- Epochs value increase shows more disrupted curves but higher possibilities of favorable loss and accuracy as per the plots of 100 and 200 max iterations values. This also depends on the data used. So, it is better to observe results on different epochs for better understanding of the performance evaluations.
- The plots are usually close for loss and accuracy for consistent activation functions throughout network layers. However, the test scores of loss and accuracy show best result for highlighted instance as shown in above pics. The increase in hidden layer size surely creates difference but it makes more significant difference when combined its effect with activation function variations.

Assumptions made:

Feature selection was not the focus, learning how to tweak the network based on hyper parameters and comparing results of various hyper parameters was the