**Summary Report on IMDB Neural Network Model**

**With different Approaches to**

**improve Performance**

**Professor Submitted By**

**CJ Wu Pavan Chaitanya Bommadevara**

**Summary:**

Modify an existing IMDB neural network model to improve performance and explain how different approaches affect the performance of the model.

**Procedure**:

They are total of 6 steps involved in this Procedure. They are

* Importing the Libraries and IMDB Dataset.
* Preparing the Required Data for Building Network.
* Building the Network Model.
* Validation of our Network Model.
* Plotting the Results.
* Retrain the model for Predictions on New Data.

**Importing the Libraries and the IMDB Dataset:**

My approach towards the problem is initially learned the importance of NumPy ,Pandas, warnings, tensorflow, models, layers, optimizers, losses matplotlib libraries.

Then I have imported the imdb dataset that is present in keras.datasets.

These are the required imports for our problem



IMDB dataset has 50,000 reviews. We are now using just the top 10,000 reviews only which will be loaded via imdb.load\_data(num\_words=10000) variable.

Train\_data ,Test\_data are the data reviews that we will be using train\_labels, test\_labels are just the Positive and Negative Labels.

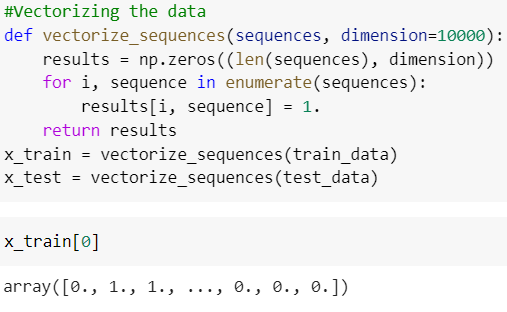
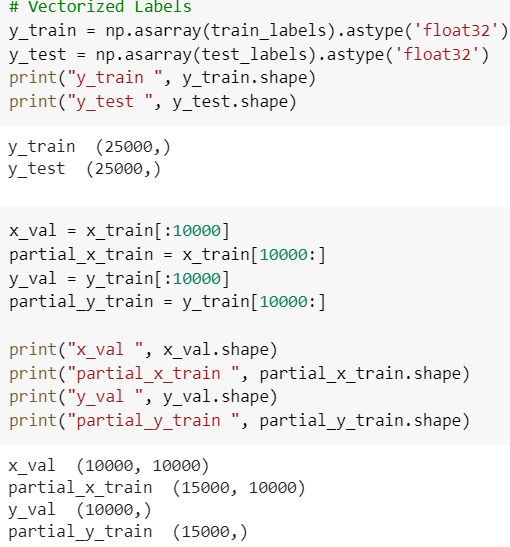
**Preparing the Required Data for Building Network:**

We cannot feed the integers into the neural Network. We must turn our lists to tensors.

1. Pad out lists so that they all have same length, turn them into the integer tensor via (samples,word\_indices) and use them as the first layer of our network.
2. By One-Hot encoding we can turn our lists to vectors of 0’s and 1’s.

The Padding and the One-hot Encoding looks like the below code:

Vectorizing the Labels: Vectorizing the Data:



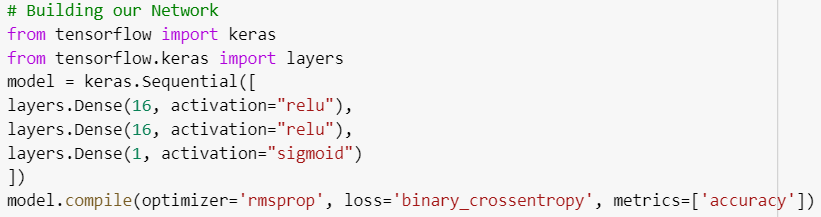
So, by now we have what the sample data for our problem looks like, we are ready to build our Network Model and to feed our Prepared Data to our Network.

**Building the Network Model:**

For our Network Model the input data is vector and our input labels are scalar. So this type of the input data we can go with the (Dense layers, Relu Activation Function, rmsprop optimizer, binary\_crossentropy) and then finally with the sigmoid activation function.

This would be the very effective way to build the neural network model.

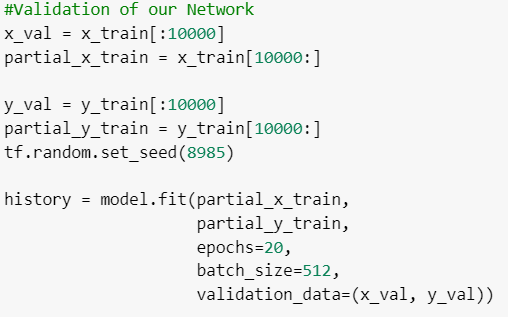
The code for building the model looks like the below figure:



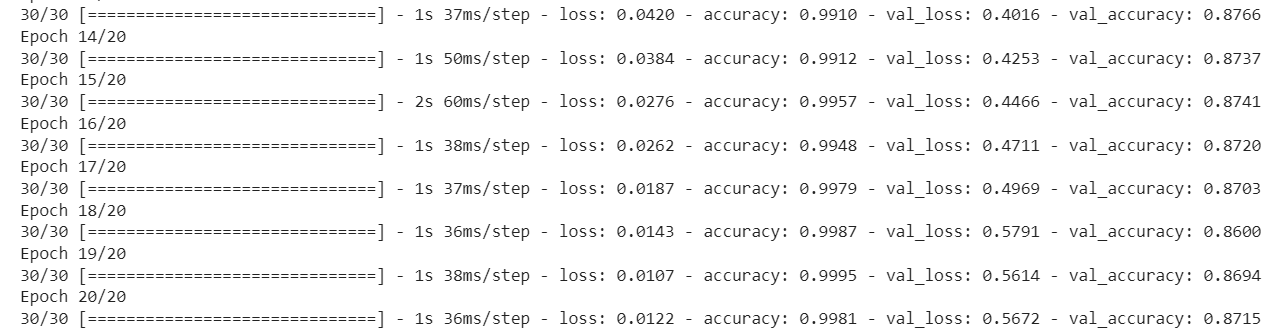
**Validation of our Network Model:**

To know the accuracy of the model that we built we will use a validation set data which is a part of original training data which contains 10,000 sample data which the model has never seen before.

The code for validation looks like:



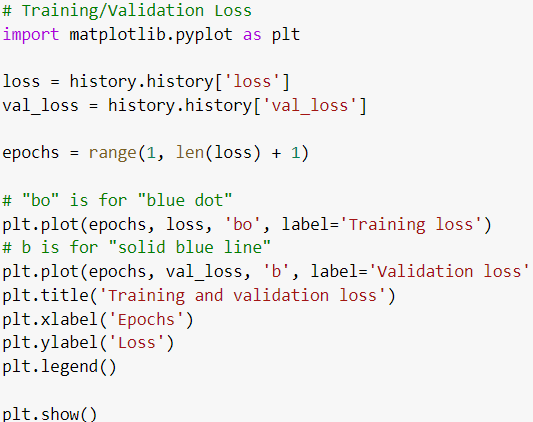
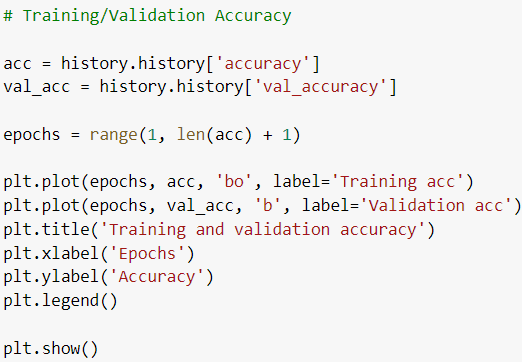
We have used 20 epochs (Iterations) with each a batch size of 512 of validation set data splitting into training with help of model.fit() function. The results for validation look like



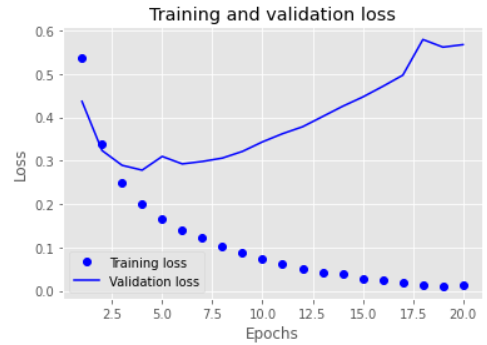
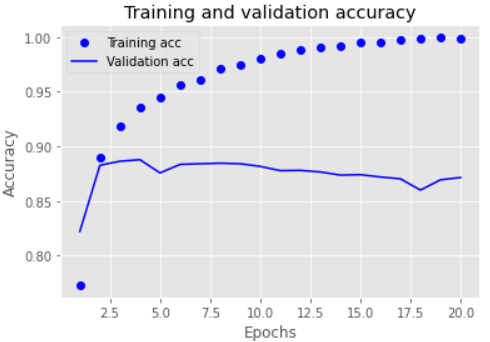
It will iterate for 20 epochs.

**Plotting the Results:**

We will now use the Matplotlib to plot the training, validation loss as well as the training, validation accuracy. The code for Plotting will be:

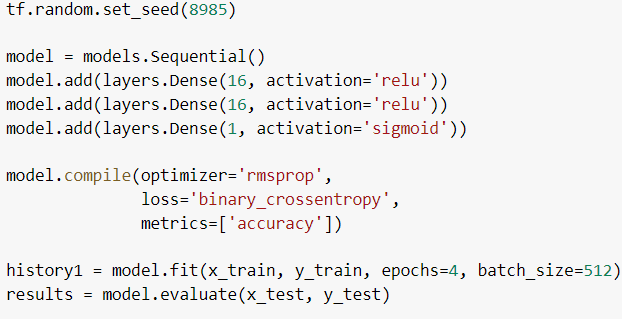
The Graphs looks like:

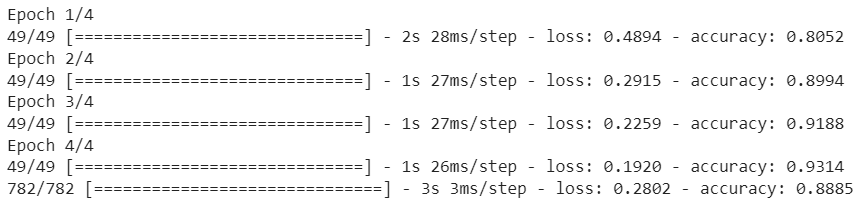
**Retrain the model for Predictions on New Data:**

Let us retrain our neural network model to predict on the new data that it hasn’t seen. For that we will use just the 4 epochs.

The code for this will look like:



The generated results will look similar like this:



Up to now we have built the model and make predictions on new data. We now take this to one step ahead by doing this Scenarios. The Scenarios are

1. You used two hidden layers. Try using one or three hidden layers, and see how doing so affects validation and test accuracy.

2. Try using layers with more hidden units or fewer hidden units: 32 units, 64 units, and so on.

3. Try using the mse loss function instead of binary\_crossentropy.

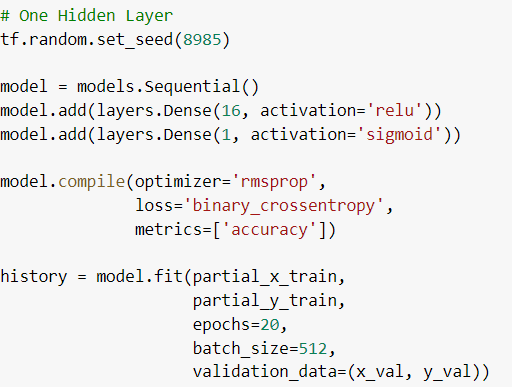
4. Try using the tanh activation (an activation that was popular in the early days of neural networks) instead of relu.

5. Use any technique we studied in class, and these include regularization, dropout, etc., to get your model to perform better on validation.

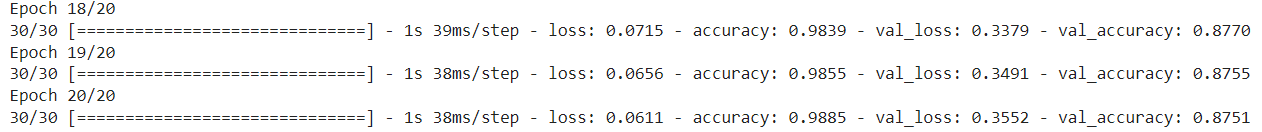
So, let us do every scenario and try to identify the results for Accuracy and the Loss for each Scenario.

**1 Layer,16 Hidden Units, Relu Activation, binary cross entropy:**

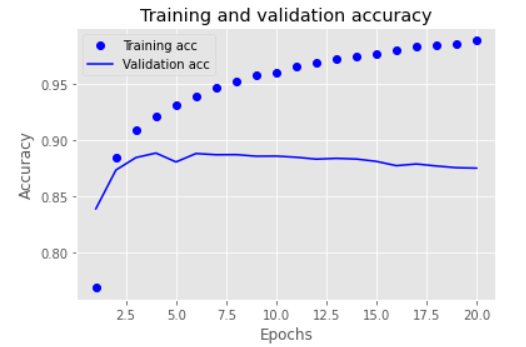
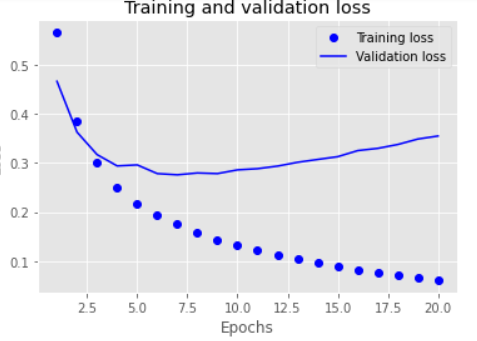
In this scenario we will be using 1 hidden Layer, 16 Hidden Units, Relu Activation Function and binary cross entropy as the loss function.



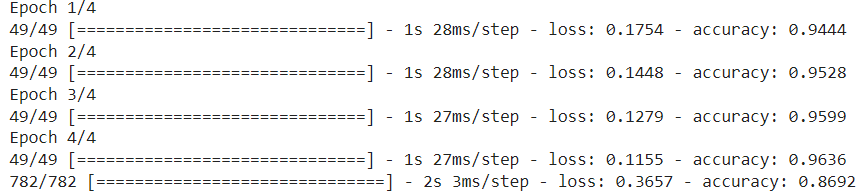
This is generating the Results like this for 20 iterations:



The Graph will be plotted like this:

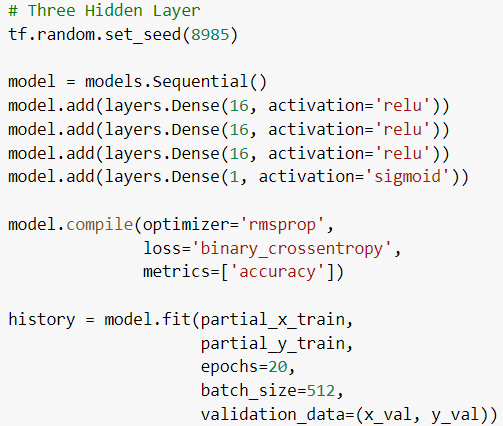


The results for 4 epochs for predicting will be like:

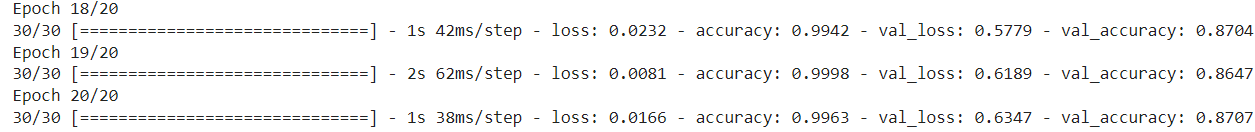


**3 Layer,16 Hidden Units, Relu Activation, binary cross entropy:**

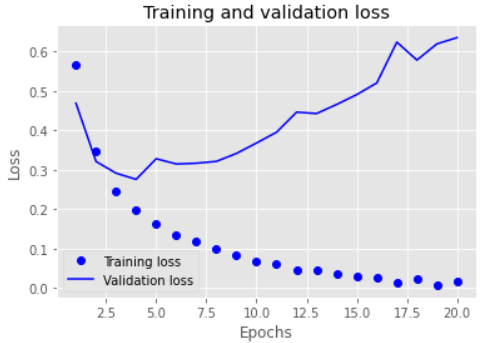
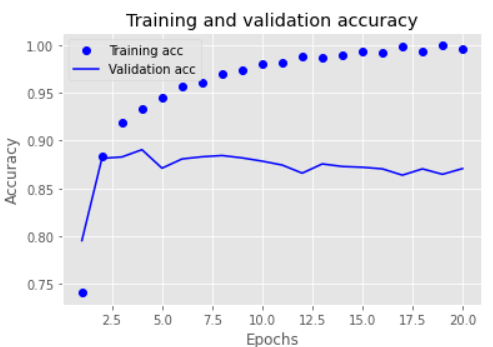
In this scenario we will be using 3 hidden Layer, 16 Hidden Units, Relu Activation Function and binary cross entropy as the loss function.



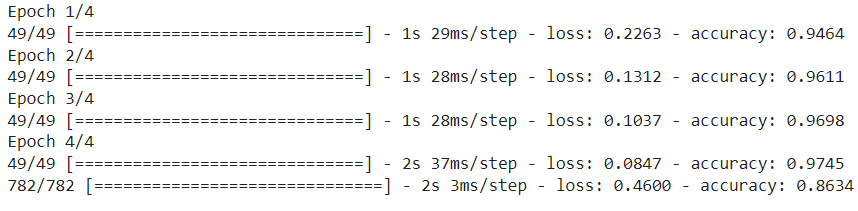
This is generating the Results like this for 20 iterations:



The Graph will be plotted like this:

The results for 4 epochs for predicting will be like:

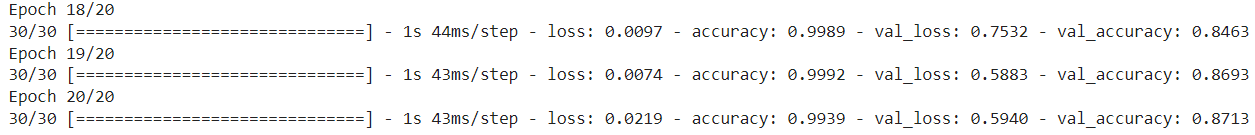


**2 Layer,32 Hidden Units, Relu Activation, binary cross entropy:**

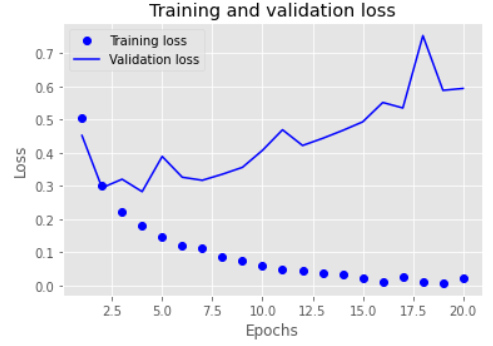
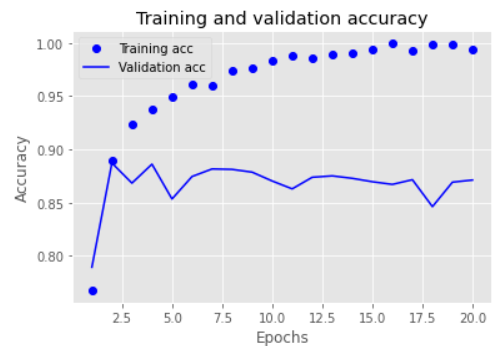
In this scenario we will be using 2 hidden Layer, 32 Hidden Units, Relu Activation Function and binary cross entropy as the loss function.



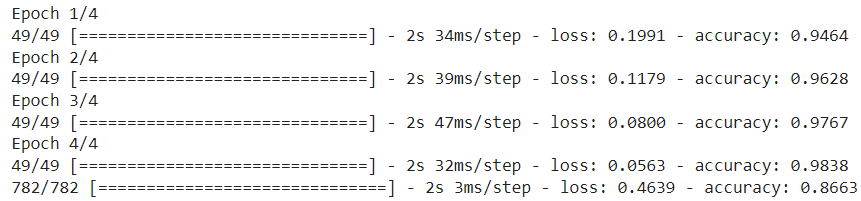
This is generating the Results like this for 20 iterations:



The Graph will be plotted like this

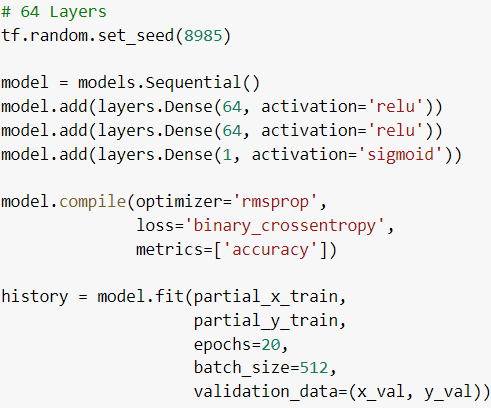
 

The results for 4 epochs for predicting will be like:

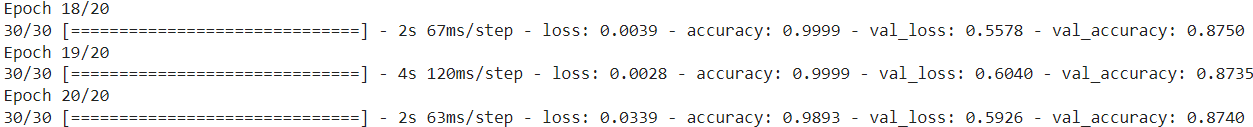


**2 Layer,64 Hidden Units, Relu Activation, binary cross entropy:**

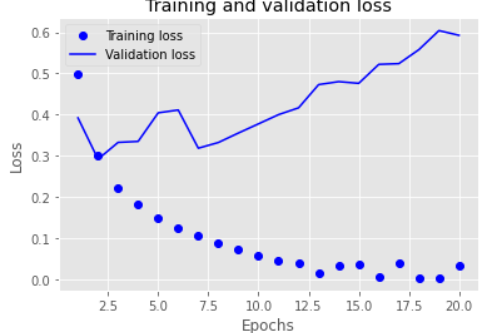
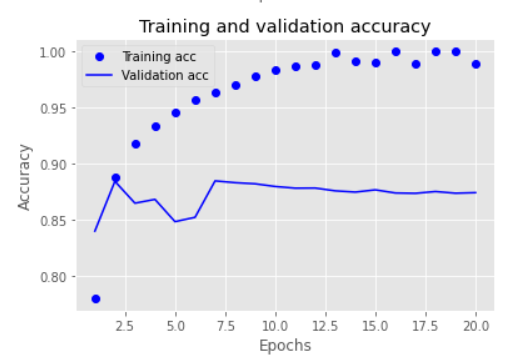
In this scenario we will be using 2 hidden Layer, 64 Hidden Units, Relu Activation Function and binary cross entropy as the loss function.



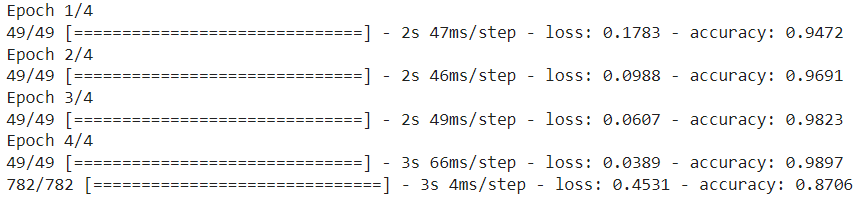
This is generating the Results like this for 20 iterations:



The Graph will be plotted like this:

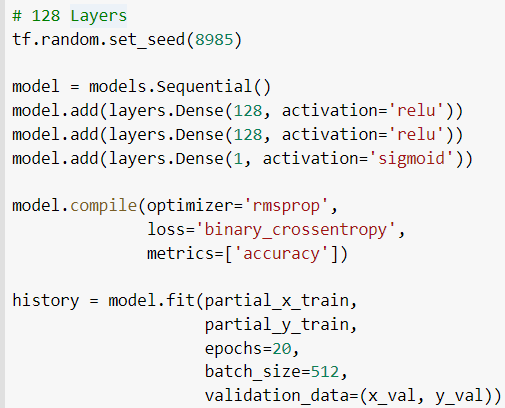
 

The results for 4 epochs for predicting will be like:

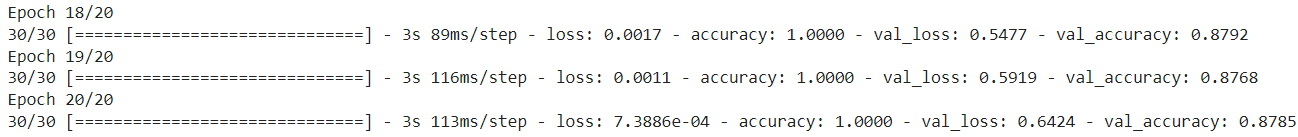


**2 Layer,128 Hidden Units, Relu Activation, binary cross entropy:**

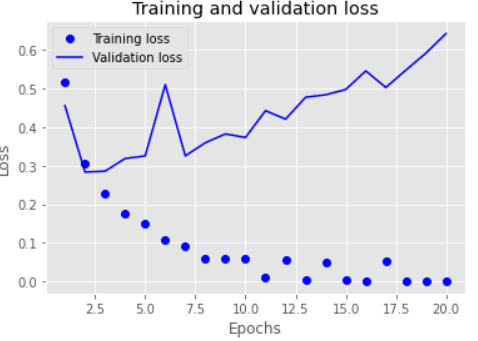
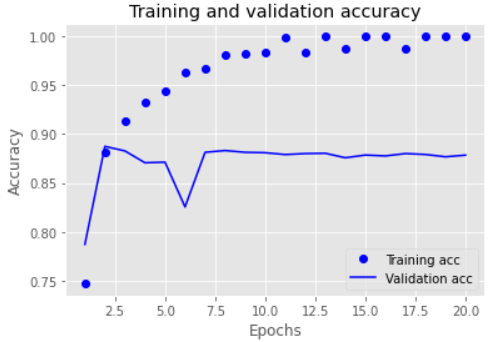
In this scenario we will be using 2 hidden Layer, 128 Hidden Units, Relu Activation Function and binary cross entropy as the loss function.



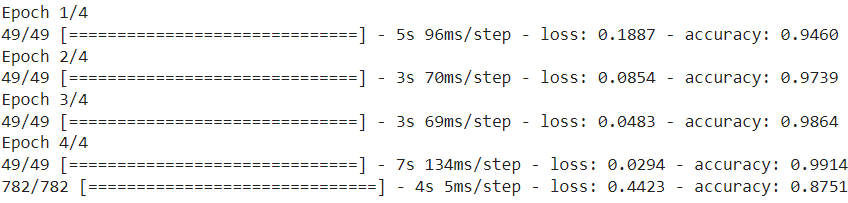
This is generating the Results like this for 20 iterations:



The Graph will be plotted like this:

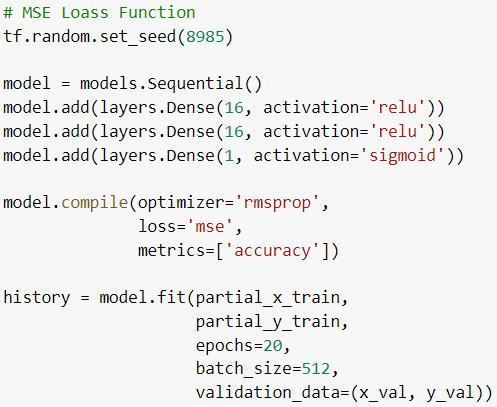
 

The results for 4 epochs for predicting will be like:

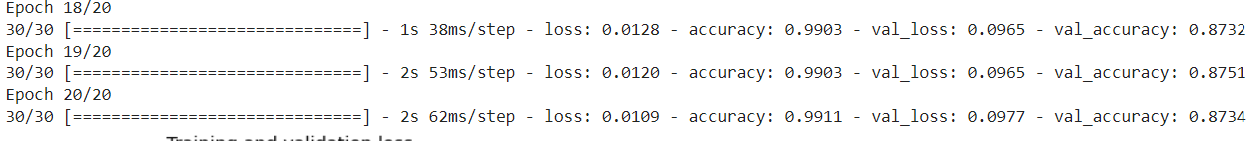


**2 Layer,16 Hidden Units, Relu Activation, MSE:**

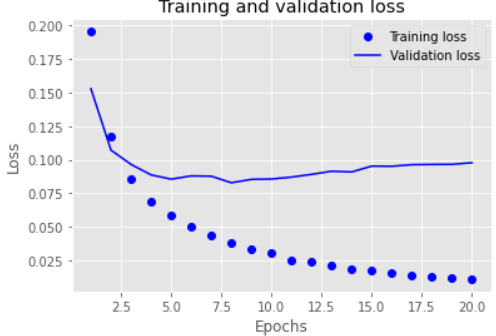
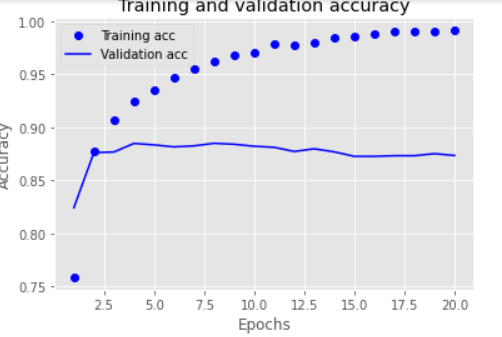
In this scenario we will be using 2 hidden Layer, 16 Hidden Units, Relu Activation Function and MSE as the loss function.



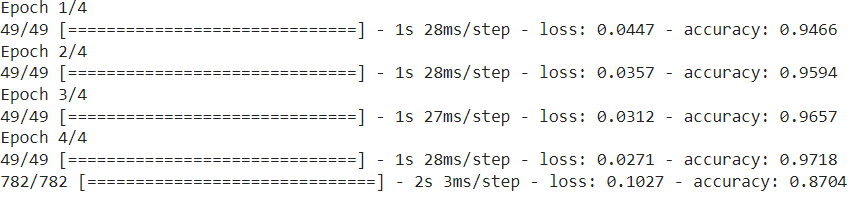
This is generating the Results like this for 20 iterations:



The Graph will be plotted like this:

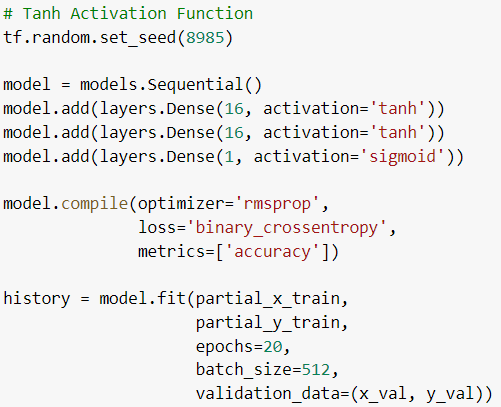
 

The results for 4 epochs for predicting will be like:

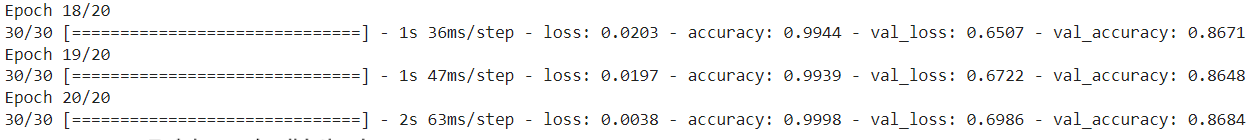


**2 Layer,16 Hidden Units, Tanh Activation, binary cross entropy:**

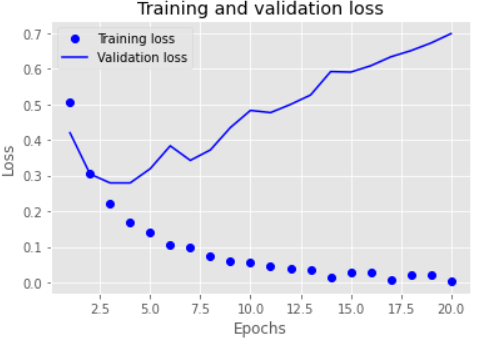
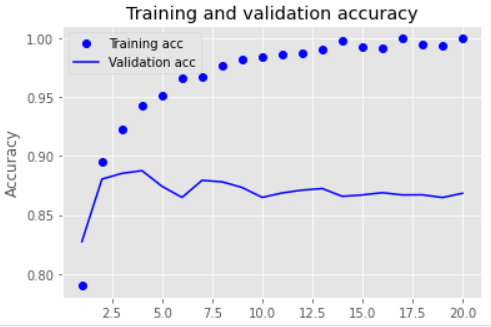
In this scenario we will be using 2 hidden Layer, 16 Hidden Units, Tanh Activation Function and binary cross entropy as the loss function.



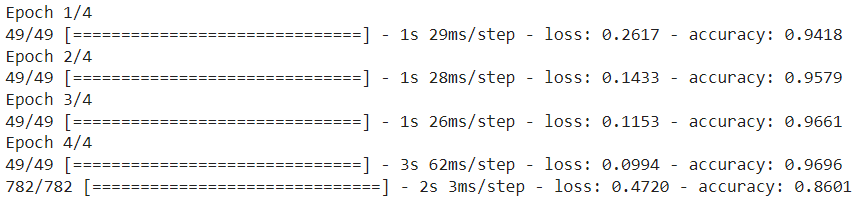
This is generating the Results like this for 20 iterations:



The Graph will be plotted like this:

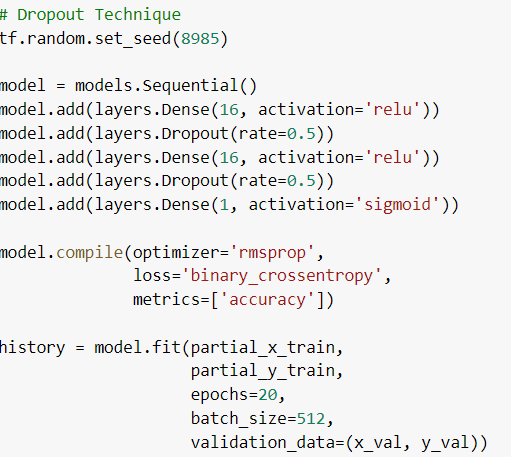
 

The results for 4 epochs for predicting will be like:

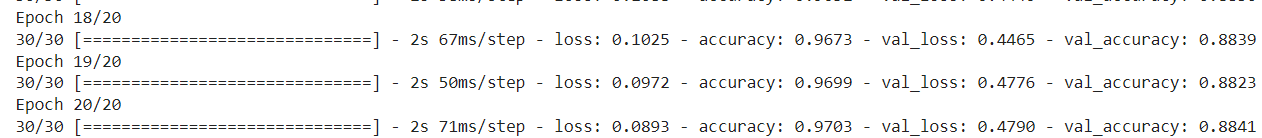


**2 Layer,16 Hidden Units, Relu Activation, binary cross entropy, dropout:**

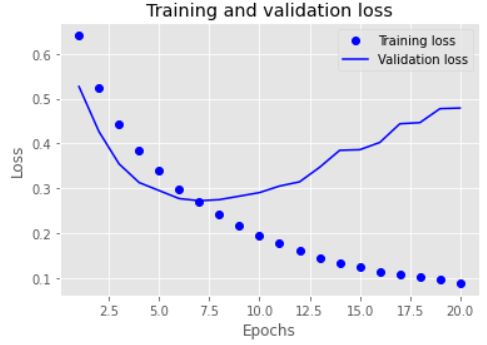
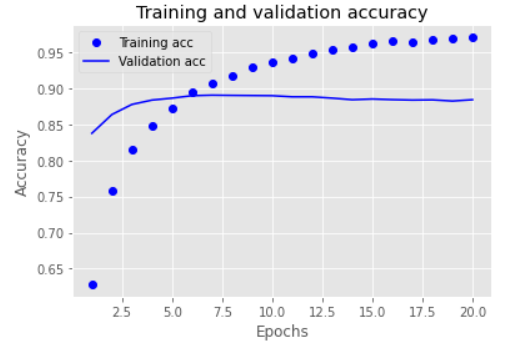
In this scenario we will be using 2 hidden Layer, 16 Hidden Units, Relu Activation Function and binary cross entropy as the loss function, dropout technique.



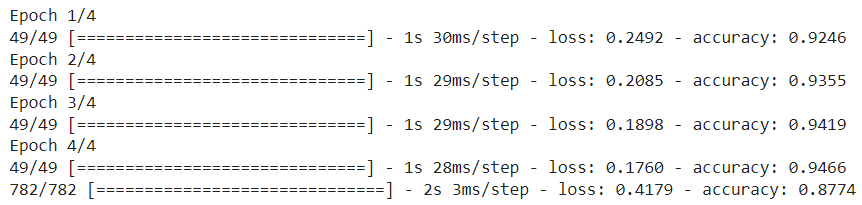
This is generating the Results like this for 20 iterations:



The Graph will be plotted like this:

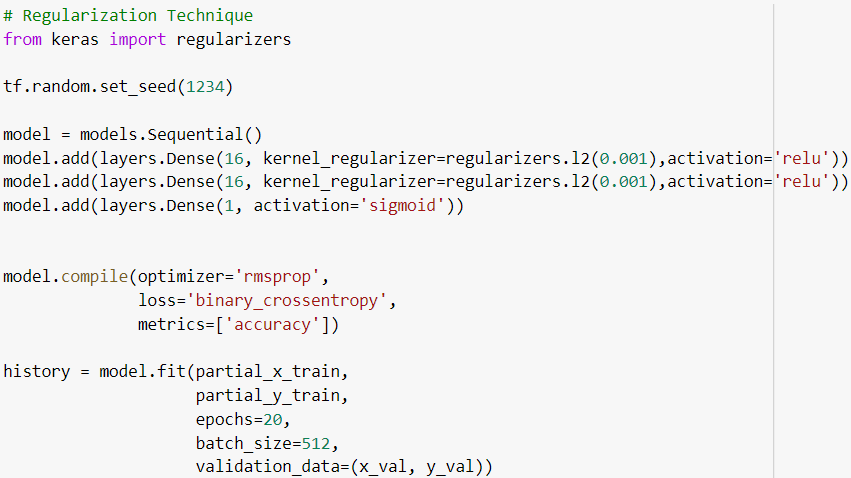
 

The results for 4 epochs for predicting will be like:

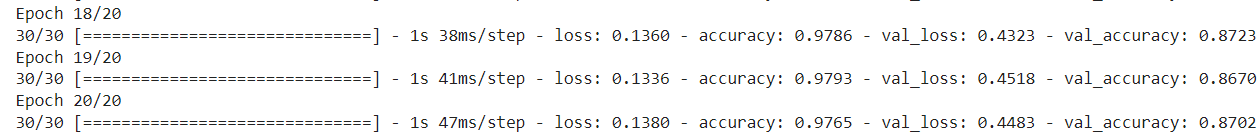


**2 Layer,16 Hidden Units, Relu Activation, binary cross entropy, Regularization:**

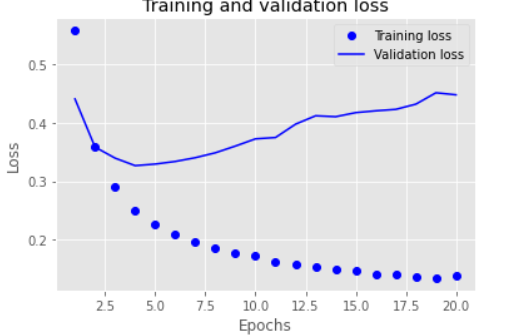
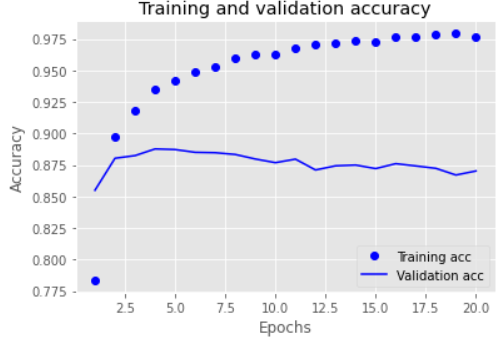
In this scenario we will be using 2 hidden Layer, 16 Hidden Units, Relu Activation Function and binary cross entropy as the loss function, Regularization technique.



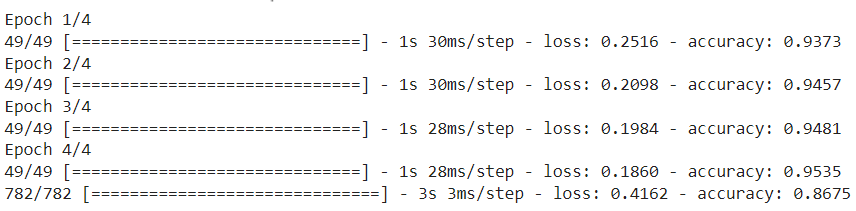
This is generating the Results like this for 20 iterations:



The Graph will be plotted like this:

The results for 4 epochs for predicting will be like:



**Summary:**

Now we will summarize all the results of accuracy, loss in the tabular Form for clear understanding.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenarios Used | Loss | | Accuracy | |
| Validation | Test | Validation | Test |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 Layer,16 Hidden Units, Relu Activation, binary cross entropy | 0.3552 | 0.3657 | 87.51% | 86.92% |
| 2 Layer,16 Hidden Units, Relu Activation, binary cross entropy | 0.5672 | 0.2802 | 87.15% | 88.85% |
| 3 Layer,16 Hidden Units, Relu Activation, binary cross entropy | 0.6347 | 0.4600 | 87.07% | 86.34% |
| 2 Layer,32 Hidden Units, Relu Activation, binary cross entropy | 0.5940 | 0.4639 | 87.13% | 86.63% |
| 2 Layer,64 Hidden Units, Relu Activation, binary cross entropy | 0.5926 | 0.4531 | 87.40% | 87.06% |
| 2 Layer,128 Hidden Units, Relu Activation, binary cross entropy | 0.6424 | 0.4423 | 87.85% | 87.51% |
| 2 Layer,16 Hidden Units, Relu Activation, MSE | 0.0977 | 0.1027 | 87.34% | 87.04% |
| 2 Layer,16 Hidden Units, Tanh Activation, binary cross entropy | 0.6986 | 0.4720 | 86.84% | 86.01% |
| 2 Layer,16 Hidden Units, Relu Activation, binary cross entropy, dropout | 0.4790 | 0.4179 | 88.41% | 87.74% |
| 2 Layer,16 Hidden Units, Relu Activation, binary cross entropy, Regularization | 0.4483 | 0.4162 | 87.02% | 86.75% |

**Conclusions:**

1 Hidden layer Validation Accuracy is best compared to 2 and 3 Hidden Layers.

The greater number of hidden Units the better the validation Accuracy.

The MSE loss function performs better than the binary\_crossentropy loss function.

When we are using the relu or tanh functions, there is a slightly difference in the Validation Accuracy

On both the test and validation datasets, the model with 2 Dense Layers, 16 Hidden Units, Relu Activation, binary\_crossentropy loss function, and dropout (0.5) approach has the maximum accuracy.

On the test dataset, the model with 2 dense layers, 16 hidden units, Tanh Activation, and binary\_crossentropy loss function has the lowest accuracy.

On the validation dataset, the model with 2 Dense Layers, 16 Hidden Units, Tanh Activation, and binary\_crossentropy loss function, Regularization has the lowest accuracy.

**References:**

Lecture 4: Examples of NN 2:

<https://kent.instructure.com/courses/57954/pages/lecture-4-examples-of-nn-2?module_item_id=2966279>

IMDB - Sentiment analysis Keras and TensorFlow:

<https://www.kaggle.com/code/drscarlat/imdb-sentiment-analysis-keras-and-tensorflow/notebook>

How to Build a Neural Network With Keras Using the IMDB Dataset:

<https://builtin.com/data-science/how-build-neural-network-keras>