Team members: Anishia Ghimire, Michael Homer, Syed Mushfiq

**Introduction**

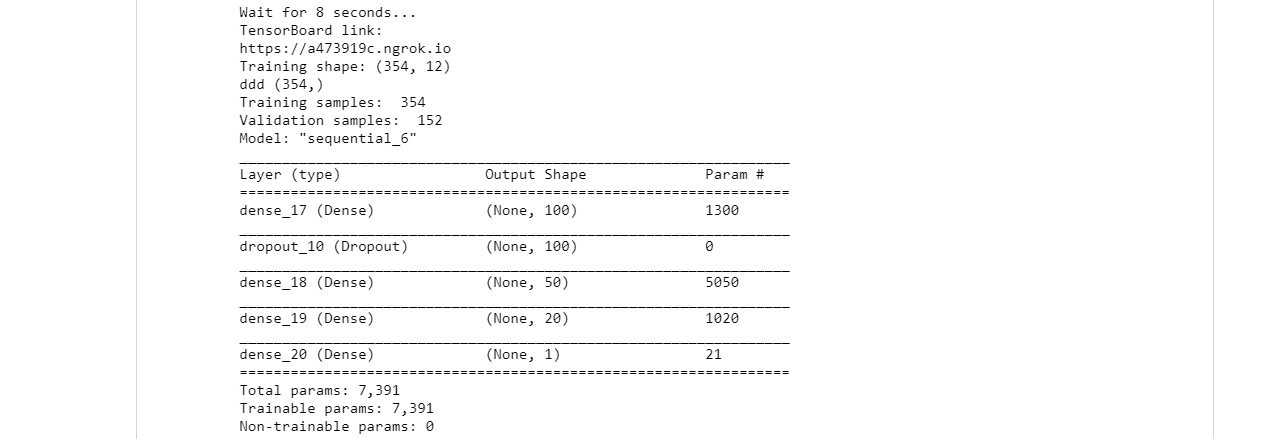
This lab serves as a way for our group to take what we have learned in class and apply it ourselves to new sets of data. This lab walks us through many different techniques for using neural networks to solve questions about data or to perform predictions off of previously seen cases. In this lab we will perform the following: Linear regression on a dataset of our choosing, Logistic Regression on Heart disease data, image classification using a convolutional neural network, text classification on movie reviews also using a CNN, text classification on movie reviews using the LSTM model, and finally applying the use of autoencoders to encode and decode images.

**Objectives**

The objectives of this Lab are to use knowledge from what we have learned in class on real-world datasets. This lab helps us discover the power of neural networks and the breadth of tasks they can be created to take on. We will also be exploring the effects hyperparameters have on both the performance and accuracy of a neural network (epochs, learning rate, batch size, etc)

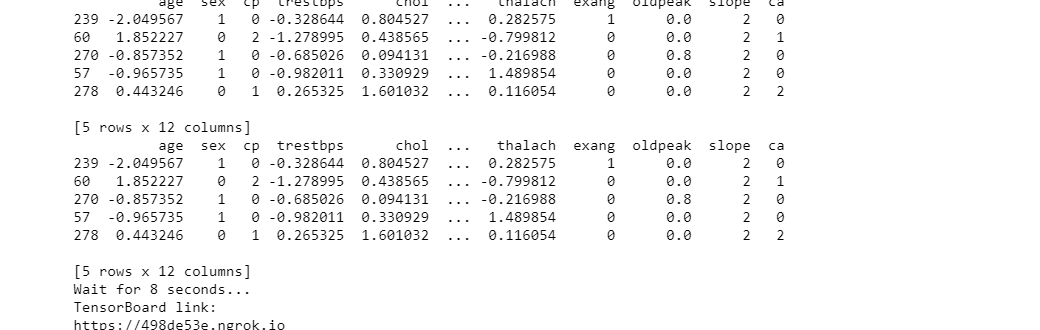
**Approaches/Methods**

**Question 1:** For question 1, we used the dataset being discussed in the class that is ‘Boston.csv’. Here to build a sequential model using keras to implement Linear Regression, we read the given data set, model 1 and 2 were created with some parameters, trained the data set. After then we plot the graph summarizing history of MAE and history for loss. The result was the algorithm waits for 8 seconds with training samples 354 and validation sample 152 and the total parameters used were 7,391.



**Question 2:**

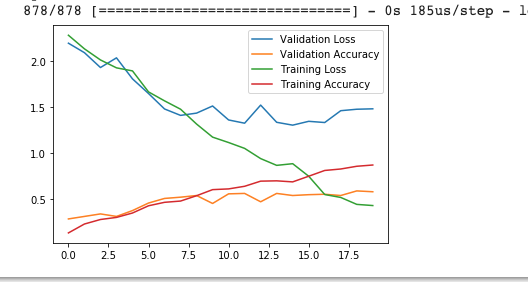
For question 2, we used the data set ‘heart.csv’ follow through selecting numerical columns which needs to be normalized. We even normalize the training data, convert numpy array to dataframe, normalize testing data by using mean and Sd of training set. Furthermore, using tensorboard colab, we build network model with normalized data through changing hyperparameters to get better result which is shown below;



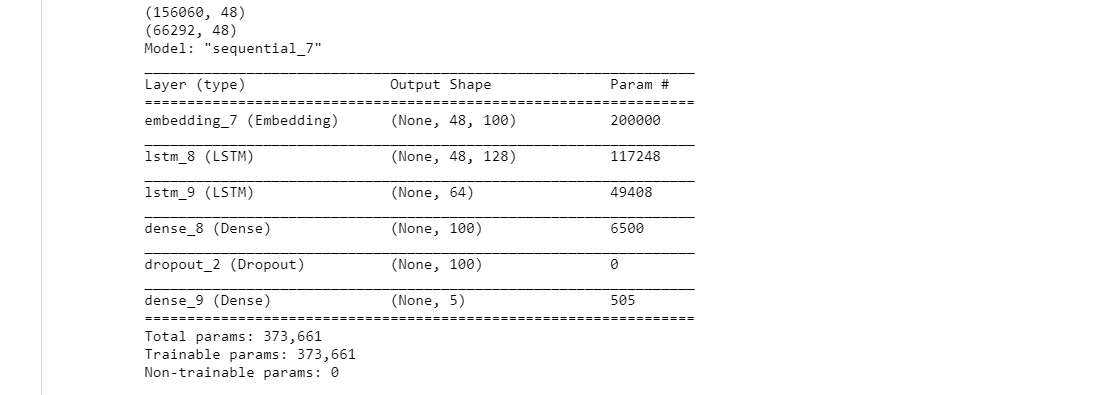
**Question 3**

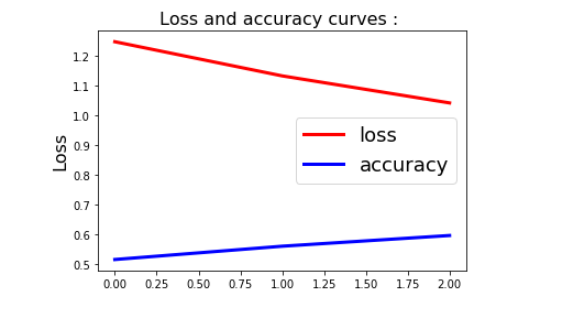
For question 3 the first thing we had to do was load in the images. The first issue was that it was at first unknown how we would load in custom images from a source outside the ones we normally use, since those datasets are already in a configuration ready to be included in a model. Luckily, keras’ preprocessing library had a built in function that we could use for loading images. The second issue was that the images we split up into several different directories based on their labels. What we did was we iterated through each of the directories to load their pictures and assigned each image with the label of the directory in which it was found. Then we passed each of the labels through an encoder so that they can be used within the model.

The model was configured with the following layers: Conv2D(32), Conv2D(64), MaxPooling2D, Dropout(0.25), Flatten(), Dense(128), Dropout(0.5),Dense(10). Additionally an SGD was used to set hyper parameters (epochs, learning rate, etc) and then the model was compiled. The following are the results.



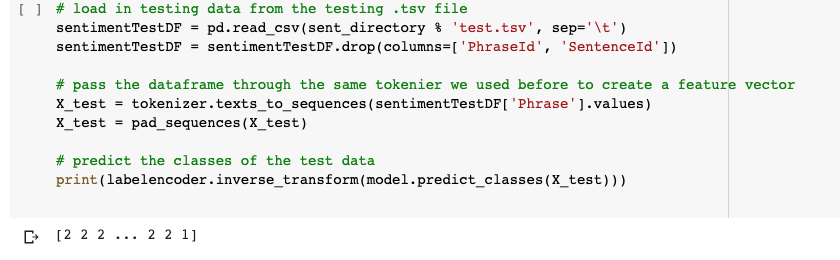
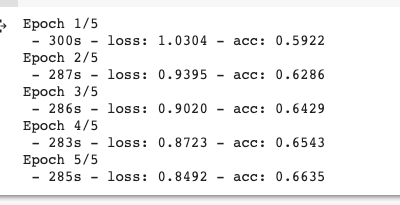
**Question 4:**

From the given data set, first we read the data then we did data preprocessing where we make text lower case and leave only letters from a-z and digits then we did model building, where we got the output as total parameters of 373,661 and got the graph for loss and accuracy curves.



**Question 5:**

After loading the data into a dataframe, the columns “PhraseID” and “SentenceID” were dropped since their values didn’t mean anything for the prediction. To keep the number of tokens down, we set the make number of features for a review to 2000. We then gave this number and the dataset to the tokenizer for it to generate the feature vectors for our dataset. We set up our model to have the following layers: Embedding, LSTM(lstm\_out, dropout=0.2), Dense(5. We then compiled our model and used the ‘categorical\_crossentropy’ loss function and ‘adam’ optimizer. Before we passed our data to the mode, we first had to encode our labels so that they are categorical and can be read by the model. These were our results:



**Question 6:**

Based on our results, since the lost and accuracy were almost the same, we concluded that the LSTM model worked well. However, it takes forever to train three epochs. One way to speed up the training time is to improve the network adding “Convolutional” layer. Convolutional Neural Networks (CNN) come from image processing. It passes a “filter” over the data and calculate a higher-level representation. Also, it has been shown to work surprisingly well for text, even though they have none of the sequences processing ability of LSTMs.

**Question 7:**

First step was to set up all of the encode and decode layers. In total there are 6 layers; From 128, down to 32, back up to 784. After the layers have been created, we passed them all into the autoencoder model and set the input dimensions to (784,). Then took the first three layers and passed those into an encoder model. Then the last three layers were passed into a decoder model. All of the images were loaded in from the mnist dataset and converted to floats with values divided by 255. After cleaning and reshaping, the data was passed into the autoencoder model. After training was performed, we then used the encoder and decoders produced by the autoencoder on our test image sets. Here was our results:



**Workflow**

Each team member was assigned specific questions to work on. After completing their section, team members would commit their code onto a Github repository so that others may view/edit it. Once all tasks were complete, each team member took part in creation of the write up as well as the lab wiki

**Datasets**

Question 1: We used the dataset being discussed in the class that is ‘Boston.csv’

Question 2: We used the data set ‘heart.csv’

Question 3:

<https://www.kaggle.com/slothkong/10-monkey-species>

This dataset is comprised of images of ten different species of monkey. The goal was to build a model that could predict the class (species) of monkey in the image.

Question 4: <https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data>

This dataset consists of movie reviews with the sentiment of the review labeled. This was used to predict the sentiment of future review

Question 5:

<https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data>

This dataset consists of movie reviews with the sentiment of the review labeled. This was used to predict the sentiment a future review might have

Question 6:<https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data>

Question 7:

We used the built in MNIST dataset for the images in this question

**Parameters**

Epochs, learning rates, testing and training sizes, quality of data sets, size of datasets, both image and textual data were used. Quality of datasets was a huge importance. Some of the results for our CNN model may have been affected by many of the images in the dataset being stock images, so they still had watermarks all over the image. This could have potentially confused the models.

**Evaluation and Discussion**

We were successful to evaluate loss and accuracy of tensor board. We normalize model and also change three hyper parameters. We implemented image classification with CNN model and text classification with LSTM model and then compare the results of CNN and LSTM model. Also, we apply autoencoder on MNIST dataset to show the encoding and decoding.

**Conclusion**

Overall, this class and the lab assignment were very helpful for us to learn about python.However, when we get to deep learning portion, the course materials were getting tougher. It was hard to implement code on pycharm so we all switched to codecademy. TAs and instructor were very helpful throughout the semester. Questions were comparatively harder than lab 1 and we were all close to finals week with our finals around so it was lack of time and a lot of pressure to coordinate among team members and complete the assignment on time however, we were able to complete our assignment. Overall, this assignment made us familiar with the python deep learning which we can implement in the real world.