REPORT

ML ASSIGNMENT 6

Prepared By:

ANJALI : MT20082

DRISHTI : MT20075

Explanation for Answer to Q1:

Loaded the dataset (CIFAR-10) using the Keras Library using the load_data() functionality. This gives us a training set with 50,000 rows and a validation set of 10,000 rows respectively.

The dataset consists of 10 labels listed below:

0: Airplane

1: Automobile

2: Bird

3: Cat

4: Deer

5: Dog

6: Frog

7: Horse

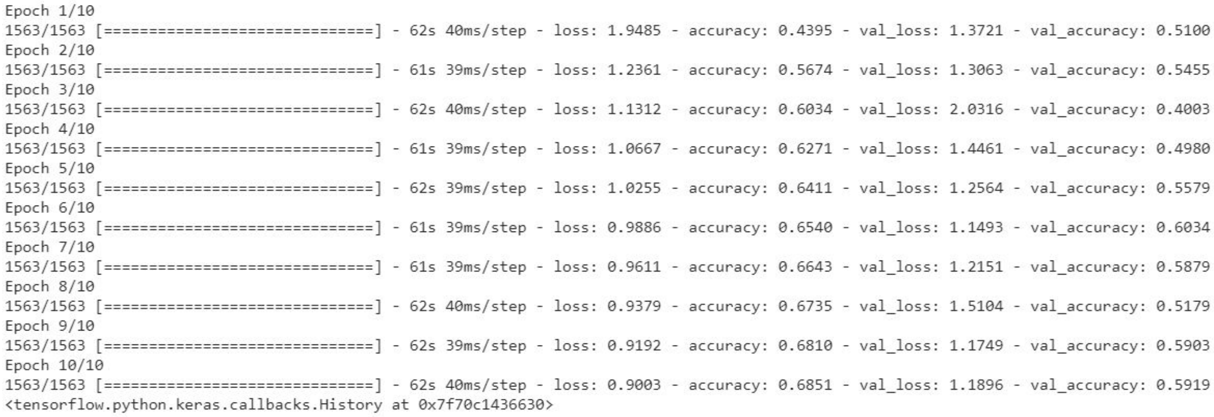
8: Ship

9: Truck

1. Used all the three channels with the input shape as (32 x 32 x 3), as shown below:

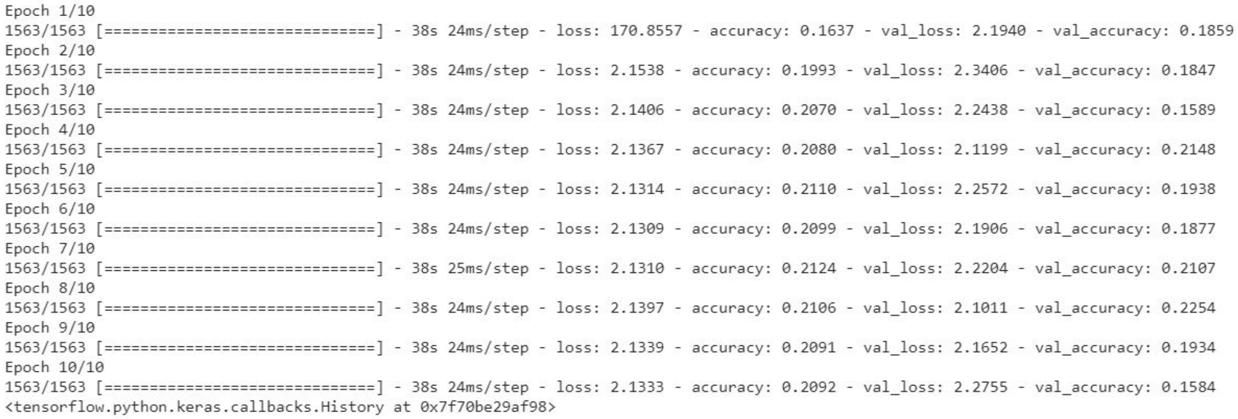
    ```
    model.add(ZeroPadding2D(padding=1, input_shape=(32,32,3)))
    ```

2. At first, in order to check the "SGD" optimizer, "categorical_crossentropy" as loss type and "accuracy" as metrics in model.compile(), we first implemented the model as shown in the architecture without any variations. The accuracy achieved for the architecture-based model is as follows:



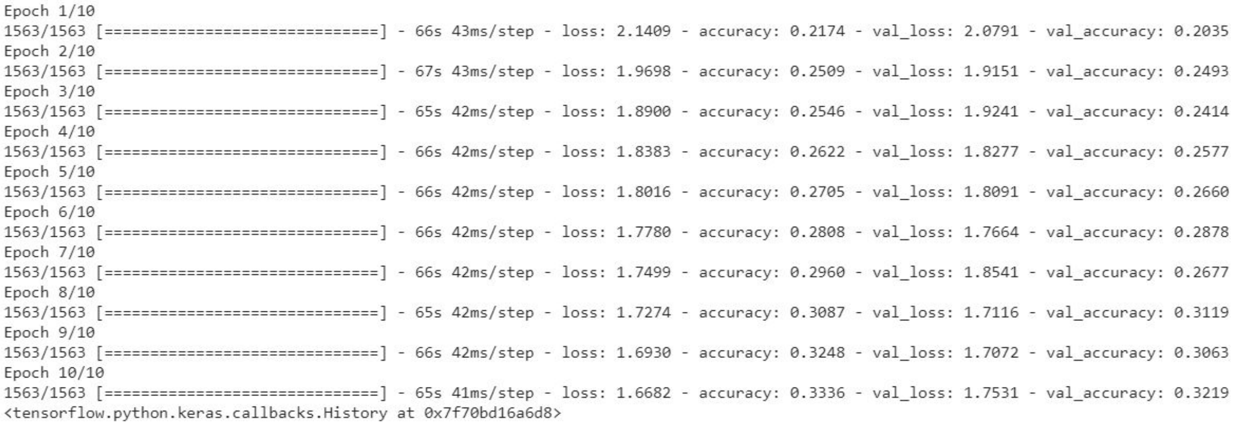
```
Epoch 1/10
1563/1563 [==============================] - 62s 40ms/step - loss: 1.9485 - accuracy: 0.4395 - val_loss: 1.3721 - val_accuracy: 0.5100
Epoch 2/10
1563/1563 [==============================] - 61s 39ms/step - loss: 1.2361 - accuracy: 0.5674 - val_loss: 1.3063 - val_accuracy: 0.5455
Epoch 3/10
1563/1563 [==============================] - 62s 40ms/step - loss: 1.1312 - accuracy: 0.6034 - val_loss: 2.0316 - val_accuracy: 0.4003
Epoch 4/10
1563/1563 [==============================] - 61s 39ms/step - loss: 1.0667 - accuracy: 0.6271 - val_loss: 1.4461 - val_accuracy: 0.4980
Epoch 5/10
1563/1563 [==============================] - 62s 39ms/step - loss: 1.0255 - accuracy: 0.6411 - val_loss: 1.2564 - val_accuracy: 0.5579
Epoch 6/10
1563/1563 [==============================] - 61s 39ms/step - loss: 0.9886 - accuracy: 0.6540 - val_loss: 1.1493 - val_accuracy: 0.6034
Epoch 7/10
1563/1563 [==============================] - 61s 39ms/step - loss: 0.9611 - accuracy: 0.6643 - val_loss: 1.2151 - val_accuracy: 0.5879
Epoch 8/10
1563/1563 [==============================] - 62s 40ms/step - loss: 0.9379 - accuracy: 0.6735 - val_loss: 1.5104 - val_accuracy: 0.5179
Epoch 9/10
1563/1563 [==============================] - 62s 39ms/step - loss: 0.9192 - accuracy: 0.6810 - val_loss: 1.1749 - val_accuracy: 0.5903
Epoch 10/10
1563/1563 [==============================] - 62s 40ms/step - loss: 0.9003 - accuracy: 0.6851 - val_loss: 1.1896 - val_accuracy: 0.5919
<tensorflow.python.keras.callbacks.History at 0x7f70c1436630>
```

3. Now let us look at the variations of this architecture model as below:
   i. No BatchNormalization:



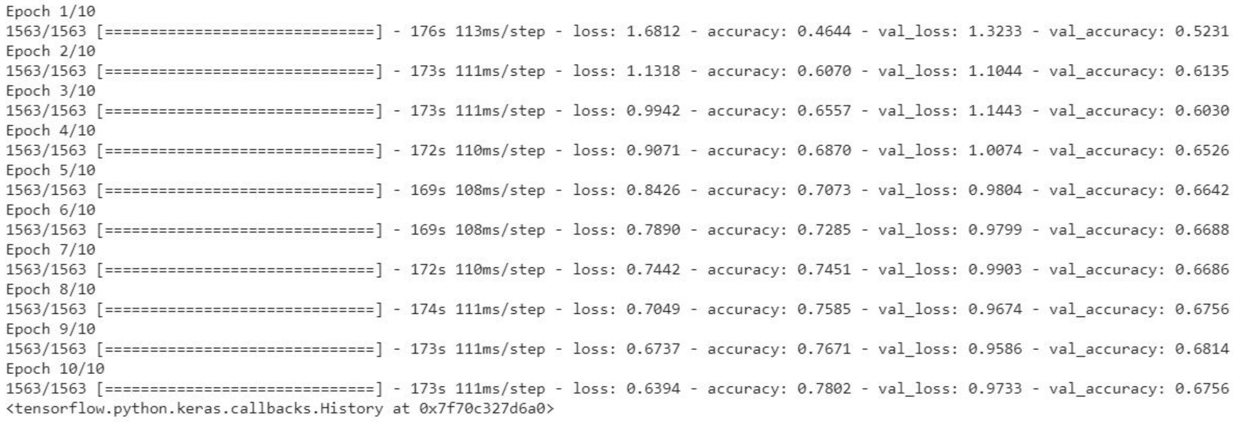
```
Epoch 1/10
1563/1563 [==============================] - 38s 24ms/step - loss: 170.8557 - accuracy: 0.1637 - val_loss: 2.1940 - val_accuracy: 0.1859
Epoch 2/10
1563/1563 [==============================] - 38s 24ms/step - loss: 2.1538 - accuracy: 0.1993 - val_loss: 2.3406 - val_accuracy: 0.1847
Epoch 3/10
1563/1563 [==============================] - 38s 24ms/step - loss: 2.1406 - accuracy: 0.2070 - val_loss: 2.2438 - val_accuracy: 0.1589
Epoch 4/10
1563/1563 [==============================] - 38s 24ms/step - loss: 2.1367 - accuracy: 0.2080 - val_loss: 2.1199 - val_accuracy: 0.2148
Epoch 5/10
1563/1563 [==============================] - 38s 24ms/step - loss: 2.1314 - accuracy: 0.2110 - val_loss: 2.2572 - val_accuracy: 0.1938
Epoch 6/10
1563/1563 [==============================] - 38s 24ms/step - loss: 2.1309 - accuracy: 0.2099 - val_loss: 2.1906 - val_accuracy: 0.1877
Epoch 7/10
1563/1563 [==============================] - 38s 25ms/step - loss: 2.1310 - accuracy: 0.2124 - val_loss: 2.2204 - val_accuracy: 0.2107
Epoch 8/10
1563/1563 [==============================] - 38s 24ms/step - loss: 2.1397 - accuracy: 0.2106 - val_loss: 2.1011 - val_accuracy: 0.2254
Epoch 9/10
1563/1563 [==============================] - 38s 24ms/step - loss: 2.1339 - accuracy: 0.2091 - val_loss: 2.1652 - val_accuracy: 0.1934
Epoch 10/10
1563/1563 [==============================] - 38s 24ms/step - loss: 2.1333 - accuracy: 0.2092 - val_loss: 2.2755 - val_accuracy: 0.1584
<tensorflow.python.keras.callbacks.History at 0x7f70be29af98>
```

ii. Two Dense Layers:



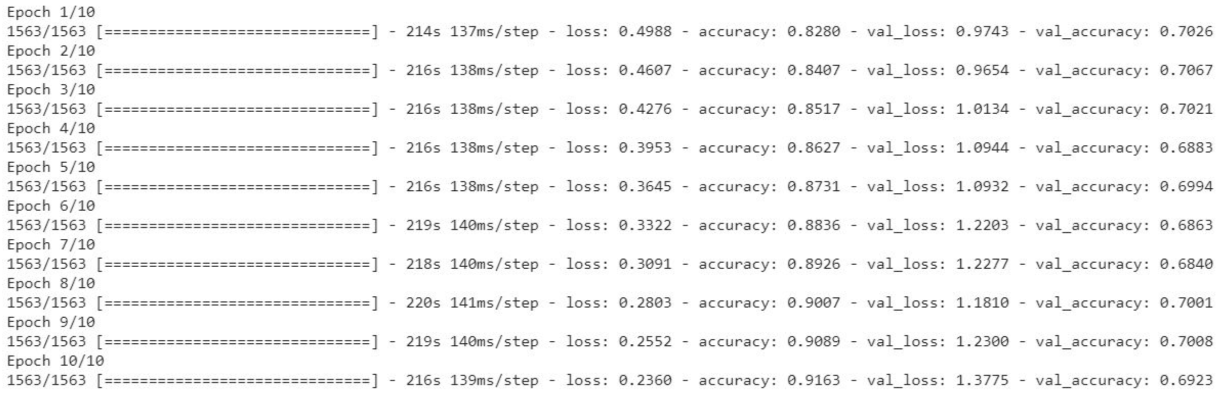
```
Epoch 1/10
1563/1563 [==============================] - 66s 43ms/step - loss: 2.1409 - accuracy: 0.2174 - val_loss: 2.0791 - val_accuracy: 0.2035
Epoch 2/10
1563/1563 [==============================] - 67s 43ms/step - loss: 1.9698 - accuracy: 0.2509 - val_loss: 1.9151 - val_accuracy: 0.2493
Epoch 3/10
1563/1563 [==============================] - 65s 42ms/step - loss: 1.8900 - accuracy: 0.2546 - val_loss: 1.9241 - val_accuracy: 0.2414
Epoch 4/10
1563/1563 [==============================] - 66s 42ms/step - loss: 1.8383 - accuracy: 0.2622 - val_loss: 1.8277 - val_accuracy: 0.2577
Epoch 5/10
1563/1563 [==============================] - 66s 42ms/step - loss: 1.8016 - accuracy: 0.2705 - val_loss: 1.8091 - val_accuracy: 0.2660
Epoch 6/10
1563/1563 [==============================] - 66s 42ms/step - loss: 1.7780 - accuracy: 0.2808 - val_loss: 1.7664 - val_accuracy: 0.2878
Epoch 7/10
1563/1563 [==============================] - 66s 42ms/step - loss: 1.7499 - accuracy: 0.2960 - val_loss: 1.8541 - val_accuracy: 0.2677
Epoch 8/10
1563/1563 [==============================] - 65s 42ms/step - loss: 1.7274 - accuracy: 0.3087 - val_loss: 1.7116 - val_accuracy: 0.3119
Epoch 9/10
1563/1563 [==============================] - 66s 42ms/step - loss: 1.6930 - accuracy: 0.3248 - val_loss: 1.7072 - val_accuracy: 0.3063
Epoch 10/10
1563/1563 [==============================] - 65s 41ms/step - loss: 1.6682 - accuracy: 0.3336 - val_loss: 1.7531 - val_accuracy: 0.3219
<tensorflow.python.keras.callbacks.History at 0x7f70bd16a6d8>
```

iii. 2 Blocks of Cov2D -> BatchNorm2D -> MaxPooling2D:



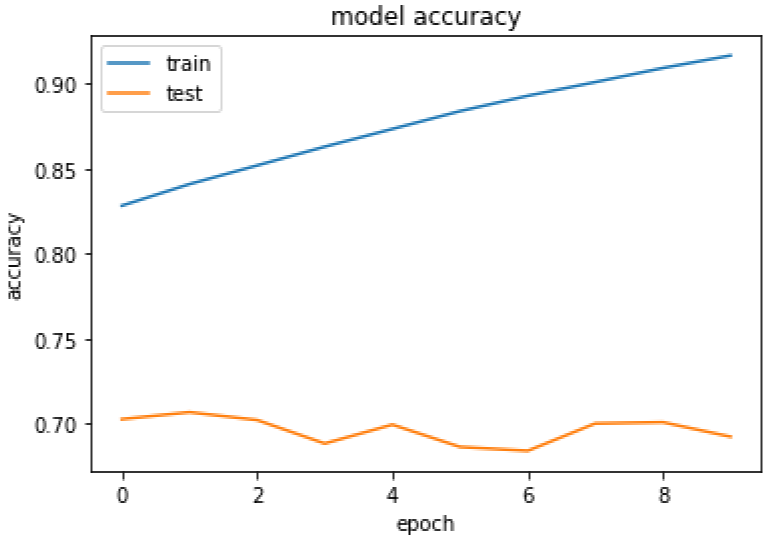
```
Epoch 1/10
1563/1563 [==============================] - 176s 113ms/step - loss: 1.6812 - accuracy: 0.4644 - val_loss: 1.3233 - val_accuracy: 0.5231
Epoch 2/10
1563/1563 [==============================] - 173s 111ms/step - loss: 1.1318 - accuracy: 0.6070 - val_loss: 1.1044 - val_accuracy: 0.6135
Epoch 3/10
1563/1563 [==============================] - 173s 111ms/step - loss: 0.9942 - accuracy: 0.6557 - val_loss: 1.1443 - val_accuracy: 0.6030
Epoch 4/10
1563/1563 [==============================] - 172s 110ms/step - loss: 0.9071 - accuracy: 0.6870 - val_loss: 1.0074 - val_accuracy: 0.6526
Epoch 5/10
1563/1563 [==============================] - 169s 108ms/step - loss: 0.8426 - accuracy: 0.7073 - val_loss: 0.9804 - val_accuracy: 0.6642
Epoch 6/10
1563/1563 [==============================] - 169s 108ms/step - loss: 0.7890 - accuracy: 0.7285 - val_loss: 0.9799 - val_accuracy: 0.6688
Epoch 7/10
1563/1563 [==============================] - 172s 110ms/step - loss: 0.7442 - accuracy: 0.7451 - val_loss: 0.9903 - val_accuracy: 0.6686
Epoch 8/10
1563/1563 [==============================] - 174s 111ms/step - loss: 0.7049 - accuracy: 0.7585 - val_loss: 0.9674 - val_accuracy: 0.6756
Epoch 9/10
1563/1563 [==============================] - 173s 111ms/step - loss: 0.6737 - accuracy: 0.7671 - val_loss: 0.9586 - val_accuracy: 0.6814
Epoch 10/10
1563/1563 [==============================] - 173s 111ms/step - loss: 0.6394 - accuracy: 0.7802 - val_loss: 0.9733 - val_accuracy: 0.6756
<tensorflow.python.keras.callbacks.History at 0x7f70c327d6a0>
```

iv. 3 Blocks of Cov2D -> BatchNorm2D -> MaxPooling3D:

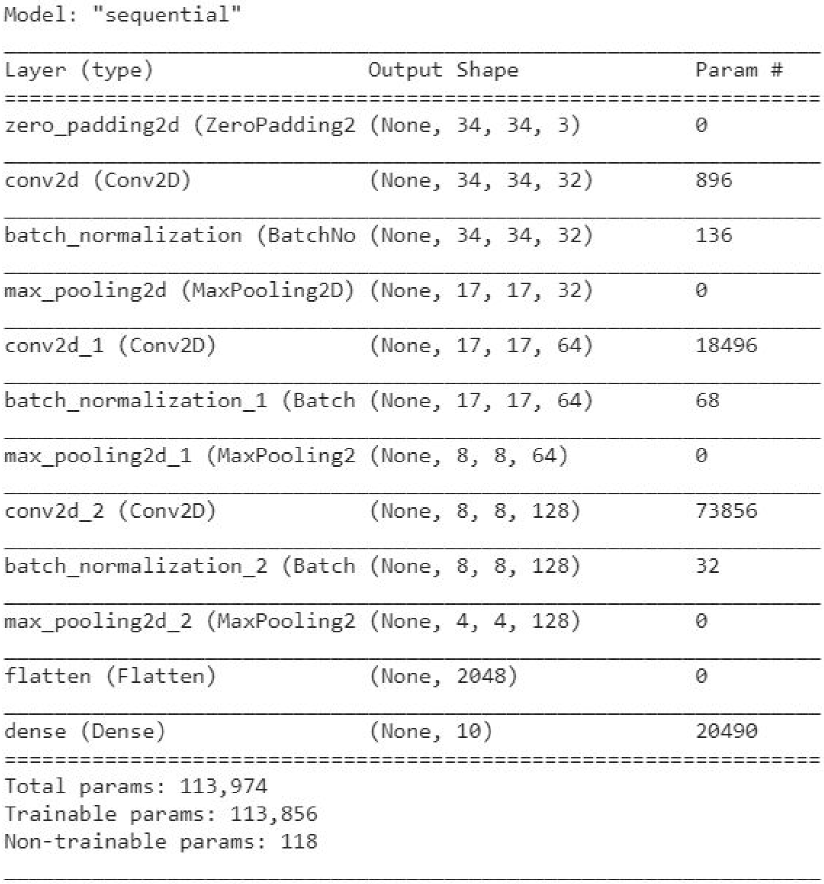


```
Epoch 1/10
1563/1563 [==============================] - 214s 137ms/step - loss: 0.4988 - accuracy: 0.8280 - val_loss: 0.9743 - val_accuracy: 0.7026
Epoch 2/10
1563/1563 [==============================] - 216s 138ms/step - loss: 0.4607 - accuracy: 0.8407 - val_loss: 0.9654 - val_accuracy: 0.7067
Epoch 3/10
1563/1563 [==============================] - 216s 138ms/step - loss: 0.4276 - accuracy: 0.8517 - val_loss: 1.0134 - val_accuracy: 0.7021
Epoch 4/10
1563/1563 [==============================] - 216s 138ms/step - loss: 0.3953 - accuracy: 0.8627 - val_loss: 1.0944 - val_accuracy: 0.6883
Epoch 5/10
1563/1563 [==============================] - 216s 138ms/step - loss: 0.3645 - accuracy: 0.8731 - val_loss: 1.0932 - val_accuracy: 0.6994
Epoch 6/10
1563/1563 [==============================] - 219s 140ms/step - loss: 0.3322 - accuracy: 0.8836 - val_loss: 1.2203 - val_accuracy: 0.6863
Epoch 7/10
1563/1563 [==============================] - 218s 140ms/step - loss: 0.3091 - accuracy: 0.8926 - val_loss: 1.2277 - val_accuracy: 0.6840
Epoch 8/10
1563/1563 [==============================] - 220s 141ms/step - loss: 0.2803 - accuracy: 0.9007 - val_loss: 1.1810 - val_accuracy: 0.7001
Epoch 9/10
1563/1563 [==============================] - 219s 140ms/step - loss: 0.2552 - accuracy: 0.9089 - val_loss: 1.2300 - val_accuracy: 0.7008
Epoch 10/10
1563/1563 [==============================] - 216s 139ms/step - loss: 0.2360 - accuracy: 0.9163 - val_loss: 1.3775 - val_accuracy: 0.6923
```

4. Therefore from the above model variations we see that the 4th model variation of the architecture performs better than others.
   Now saving this as our best model, let us look at the plot of accuracy vs epoch as shown below:



Also the model summary of the best model is as follows:



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
zero_padding2d (ZeroPadding2 (None, 34, 34, 3)         0         
_________________________________________________________________
conv2d (Conv2D)              (None, 34, 34, 32)        896       
_________________________________________________________________
batch_normalization (BatchNo (None, 34, 34, 32)        136       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 17, 17, 32)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 17, 17, 64)        18496     
_________________________________________________________________
batch_normalization_1 (Batch (None, 17, 17, 64)        68        
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 128)         73856     
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 128)         32        
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0         
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0         
_________________________________________________________________
dense (Dense)                (None, 10)                20490     
=================================================================
Total params: 113,974
Trainable params: 113,856
Non-trainable params: 118
_________________________________________________________________
```

Explanation for Answer to Q2

Downloaded the dataset. It was in .txt format. Read it as a csv file by separating the Sentence and Tag columns.

Entry 1 in Tag column indicates positive sentiment and entry 0 indicates negative sentiment.

Preprocessing steps followed on the data for better performance:

- Removing punctuations
- Lowercasing the text
- Removing stop words

The dataset consists of 1000 instances. Divided the dataset into training and test set in the ratio 70:30.

(1) Downloaded the dataset and Recurrent Neural Network is implemented for binary classification.

(2) Used pre-trained GloVe glove.6B.50d.txt obtained from the link provided in the assignment for vectorizing the text. The pre-trained glove 50dimension word embeddings are used as initial input.

(3) The hyper parameters taken:

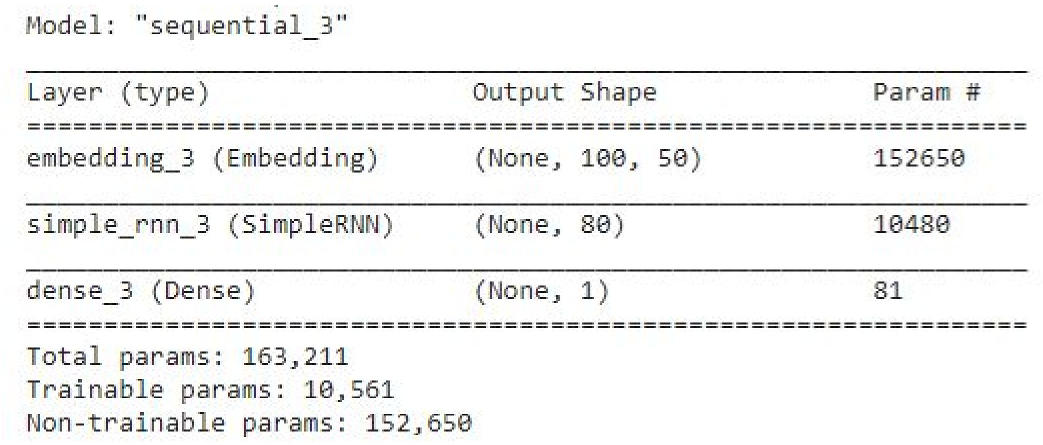
Optimizer - Adam - Converges faster, but works the same as stochastic gradient descent.

Loss criteria - Binary Cross Entropy - Better for the purpose of binary classification.

Layer 1 - simpleRNN - gives reasonable validation accuracy, although more layers can be used.

Before Output- Dense Layer - just 1 unit used as a single output expected out of the two classes.

Activation function used - Sigmoid - Again, sigmoid works better when there are only two classes to be predicted by the model.

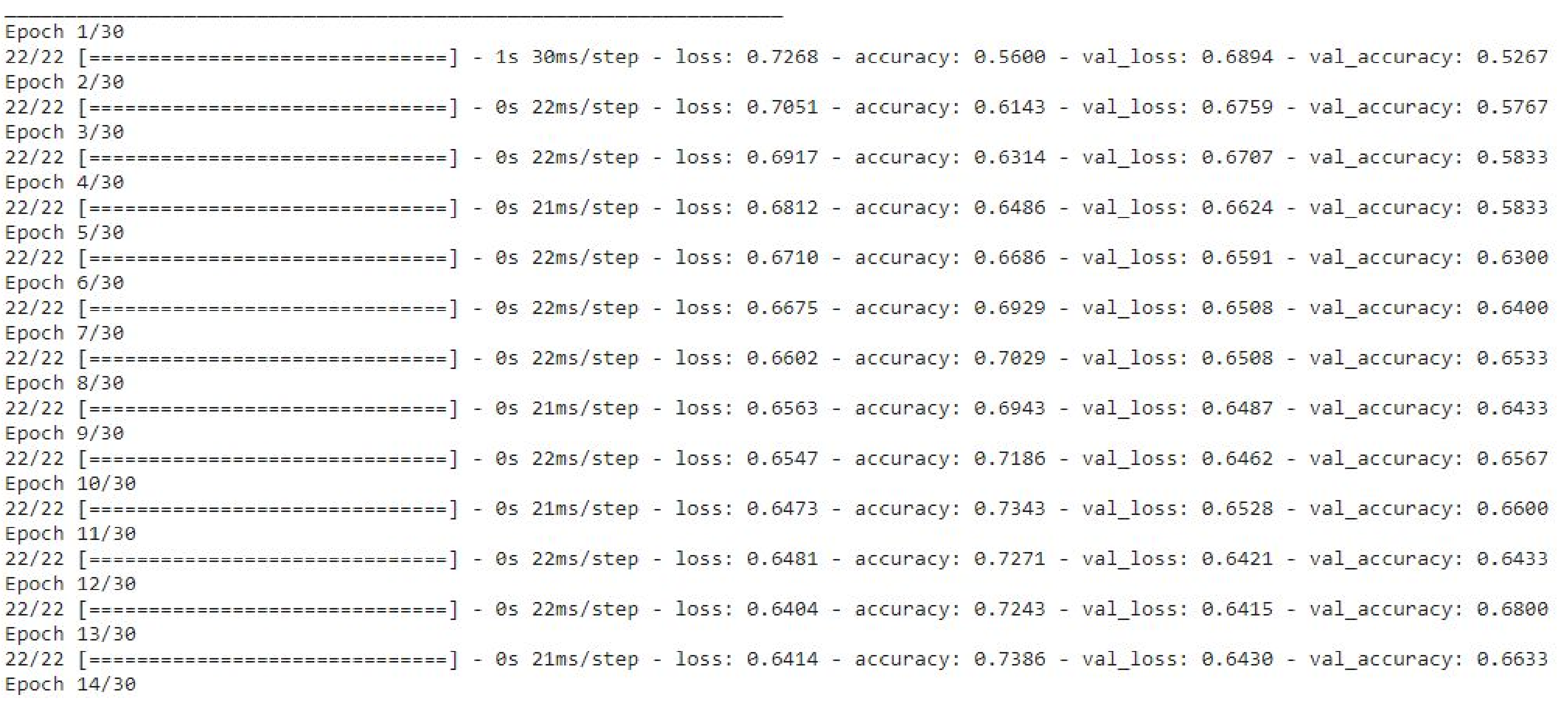


```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_3 (Embedding)      (None, 100, 50)           152650    
_________________________________________________________________
simple_rnn_3 (SimpleRNN)     (None, 80)                10480     
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 81        
=================================================================
Total params: 163,211
Trainable params: 10,561
Non-trainable params: 152,650
```

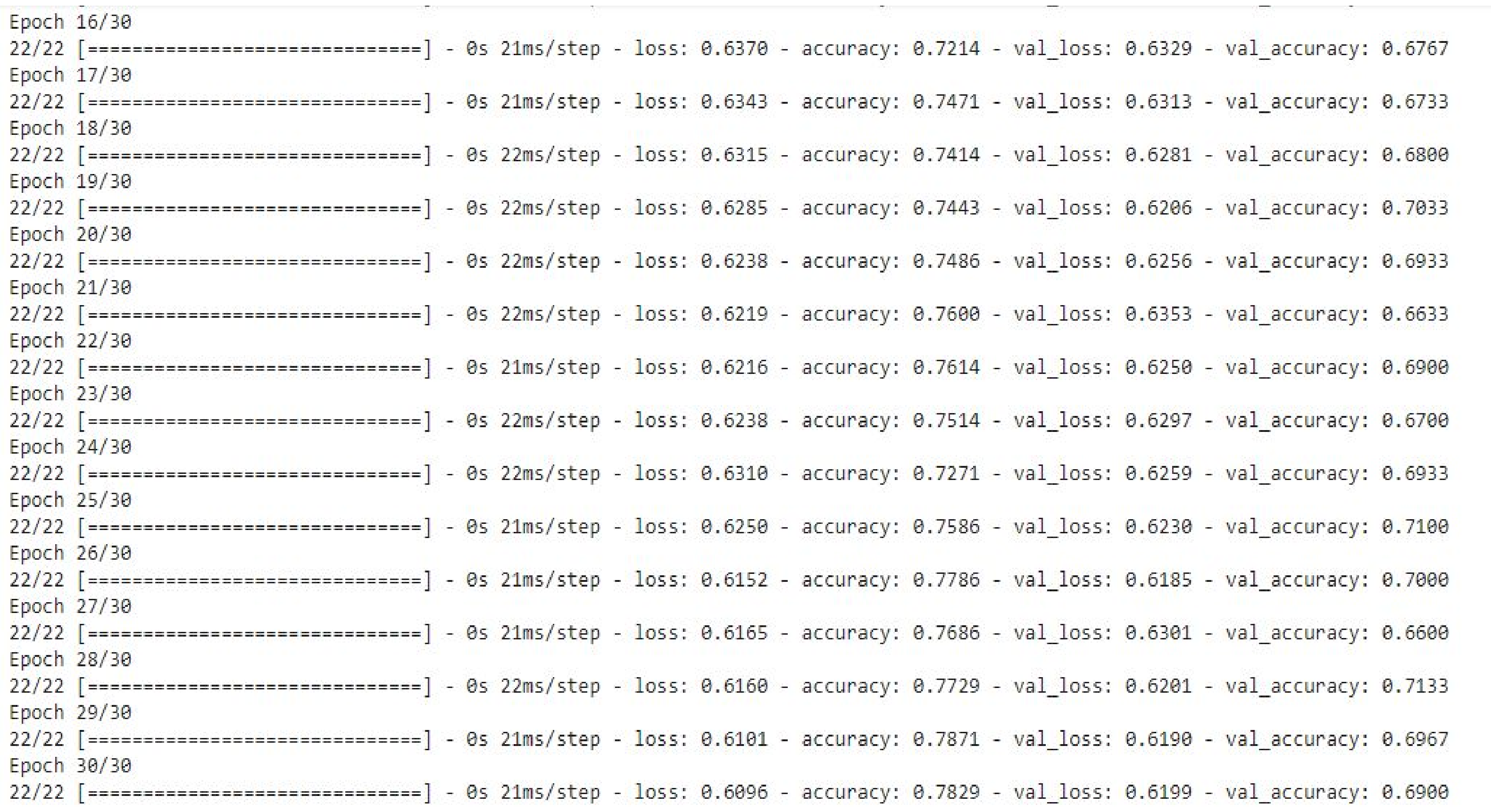
(4) Below are the screenshots for the training and the validation accuracy for different epochs.

TRAINING ACCURACY at 30th epoch - 78.29%

VALIDATION ACCURACY at 30th epoch - 69%



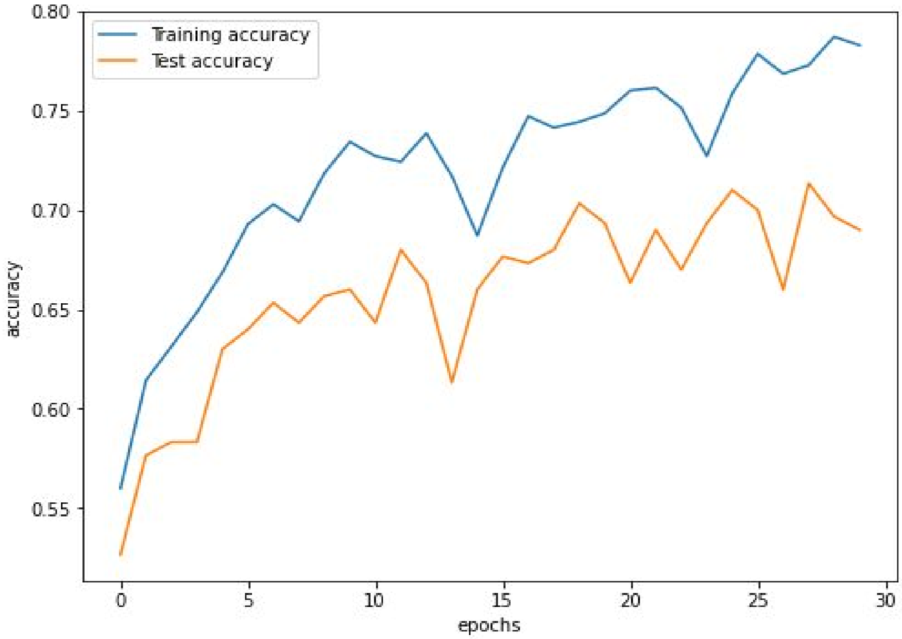
```
Epoch 1/30
22/22 [==============================] - 1s 30ms/step - loss: 0.7268 - accuracy: 0.5600 - val_loss: 0.6894 - val_accuracy: 0.5267
Epoch 2/30
22/22 [==============================] - 0s 22ms/step - loss: 0.7051 - accuracy: 0.6143 - val_loss: 0.6759 - val_accuracy: 0.5767
Epoch 3/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6917 - accuracy: 0.6314 - val_loss: 0.6707 - val_accuracy: 0.5833
Epoch 4/30
22/22 [==============================] - 0s 21ms/step - loss: 0.6812 - accuracy: 0.6486 - val_loss: 0.6624 - val_accuracy: 0.5833
Epoch 5/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6710 - accuracy: 0.6686 - val_loss: 0.6591 - val_accuracy: 0.6300
Epoch 6/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6675 - accuracy: 0.6929 - val_loss: 0.6508 - val_accuracy: 0.6400
Epoch 7/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6602 - accuracy: 0.7029 - val_loss: 0.6508 - val_accuracy: 0.6533
Epoch 8/30
22/22 [==============================] - 0s 21ms/step - loss: 0.6563 - accuracy: 0.6943 - val_loss: 0.6487 - val_accuracy: 0.6433
Epoch 9/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6547 - accuracy: 0.7186 - val_loss: 0.6462 - val_accuracy: 0.6567
Epoch 10/30
22/22 [==============================] - 0s 21ms/step - loss: 0.6473 - accuracy: 0.7343 - val_loss: 0.6528 - val_accuracy: 0.6600
Epoch 11/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6481 - accuracy: 0.7271 - val_loss: 0.6421 - val_accuracy: 0.6433
Epoch 12/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6404 - accuracy: 0.7243 - val_loss: 0.6415 - val_accuracy: 0.6800
Epoch 13/30
22/22 [==============================] - 0s 21ms/step - loss: 0.6414 - accuracy: 0.7386 - val_loss: 0.6430 - val_accuracy: 0.6633
Epoch 14/30
```



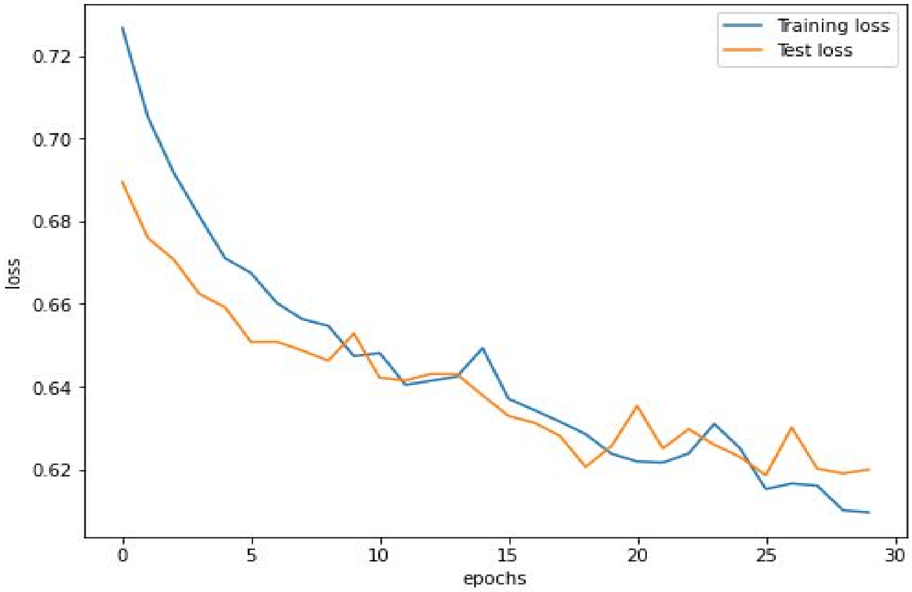
```
Epoch 16/30
22/22 [==============================] - 0s 21ms/step - loss: 0.6370 - accuracy: 0.7214 - val_loss: 0.6329 - val_accuracy: 0.6767
Epoch 17/30
22/22 [==============================] - 0s 21ms/step - loss: 0.6343 - accuracy: 0.7471 - val_loss: 0.6313 - val_accuracy: 0.6733
Epoch 18/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6315 - accuracy: 0.7414 - val_loss: 0.6281 - val_accuracy: 0.6800
Epoch 19/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6285 - accuracy: 0.7443 - val_loss: 0.6206 - val_accuracy: 0.7033
Epoch 20/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6238 - accuracy: 0.7486 - val_loss: 0.6256 - val_accuracy: 0.6933
Epoch 21/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6219 - accuracy: 0.7600 - val_loss: 0.6353 - val_accuracy: 0.6633
Epoch 22/30
22/22 [==============================] - 0s 21ms/step - loss: 0.6216 - accuracy: 0.7614 - val_loss: 0.6250 - val_accuracy: 0.6900
Epoch 23/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6238 - accuracy: 0.7514 - val_loss: 0.6297 - val_accuracy: 0.6700
Epoch 24/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6310 - accuracy: 0.7271 - val_loss: 0.6259 - val_accuracy: 0.6933
Epoch 25/30
22/22 [==============================] - 0s 21ms/step - loss: 0.6250 - accuracy: 0.7586 - val_loss: 0.6230 - val_accuracy: 0.7100
Epoch 26/30
22/22 [==============================] - 0s 21ms/step - loss: 0.6152 - accuracy: 0.7786 - val_loss: 0.6185 - val_accuracy: 0.7000
Epoch 27/30
22/22 [==============================] - 0s 21ms/step - loss: 0.6165 - accuracy: 0.7686 - val_loss: 0.6301 - val_accuracy: 0.6600
Epoch 28/30
22/22 [==============================] - 0s 22ms/step - loss: 0.6160 - accuracy: 0.7729 - val_loss: 0.6201 - val_accuracy: 0.7133
Epoch 29/30
22/22 [==============================] - 0s 21ms/step - loss: 0.6101 - accuracy: 0.7871 - val_loss: 0.6190 - val_accuracy: 0.6967
Epoch 30/30
22/22 [==============================] - 0s 21ms/step - loss: 0.6096 - accuracy: 0.7829 - val_loss: 0.6199 - val_accuracy: 0.6900
```

Below are the plots for loss and accuracy over the epochs.

Plot for accuracy



Plot for Loss


Training loss
Test loss
loss
epochs