Lab Assignment-2

Team Members:

Aparna Manda: 23

Lohitha Yenugu: 43

Youtube Link: Video Link

Introduction: This lab assignment focuses on Deep Learning. In this assignment, we implemented image classification with CNN, text classification with CNN and LSTM, Logistic Regression and Linear Regression.

Objective: To implement below tasks:

1. Linear Regression
2. Logistic Regression
3. Text Classification with CNN
4. Text Classification with LSTM
5. Compare 3 and 4 results
6. Image Classification with CNN

Requirements:

1. PyCharm IDE
2. Python 3.7

3. Anaconda Interpreter
4. Keras
5. TensorBoard

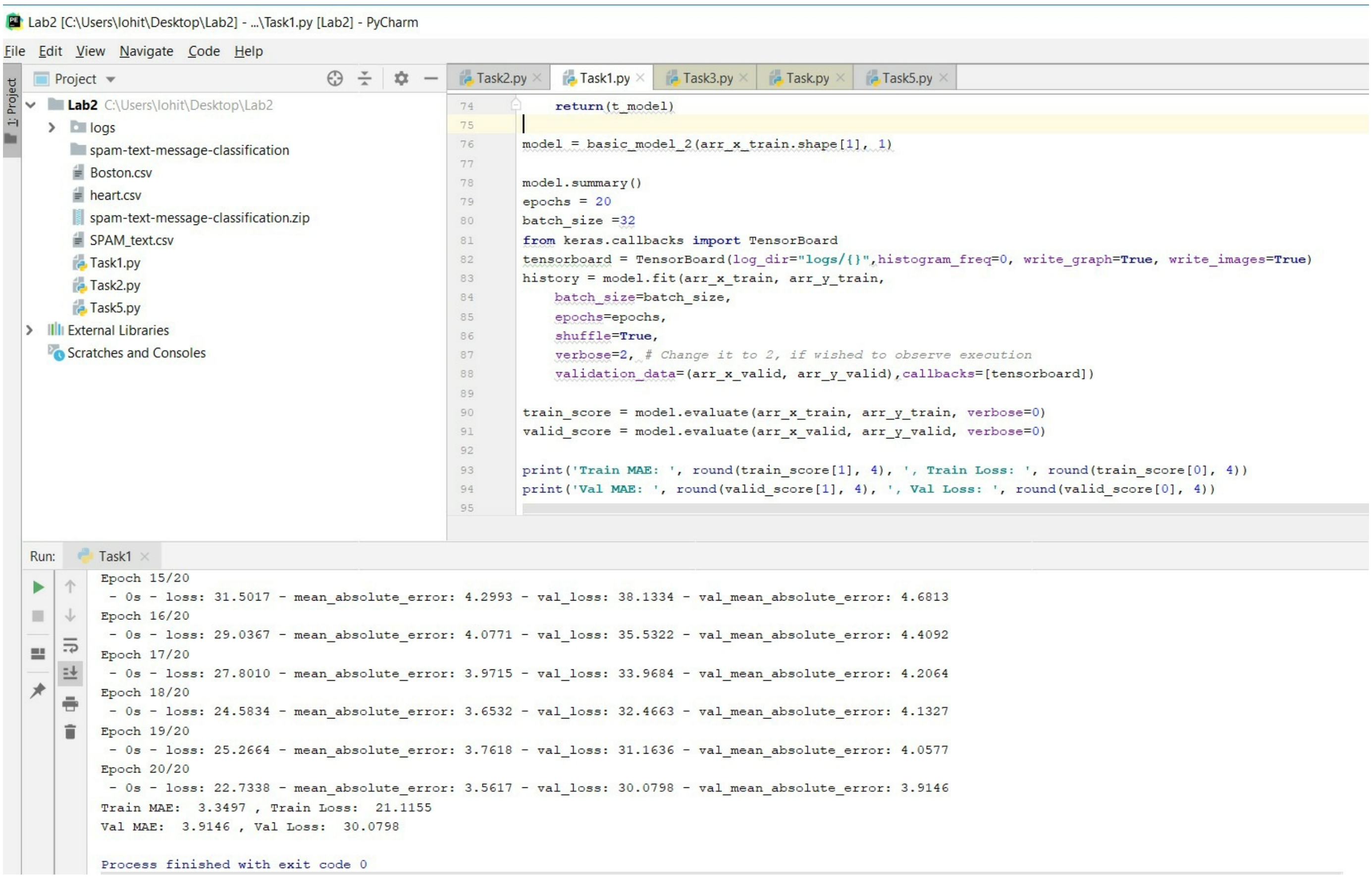
WorkFlow:

Task-1: Implement the LinearRegressio nwith any data set of your choice except the datasets being discussed in the class or source code.

1. Show the graph in TensorBoard.
2. Plot the loss and then change the below parameter and report your view how the result changes in each case
    - learning rate
    - batch size
    - optimizer
    - activation function

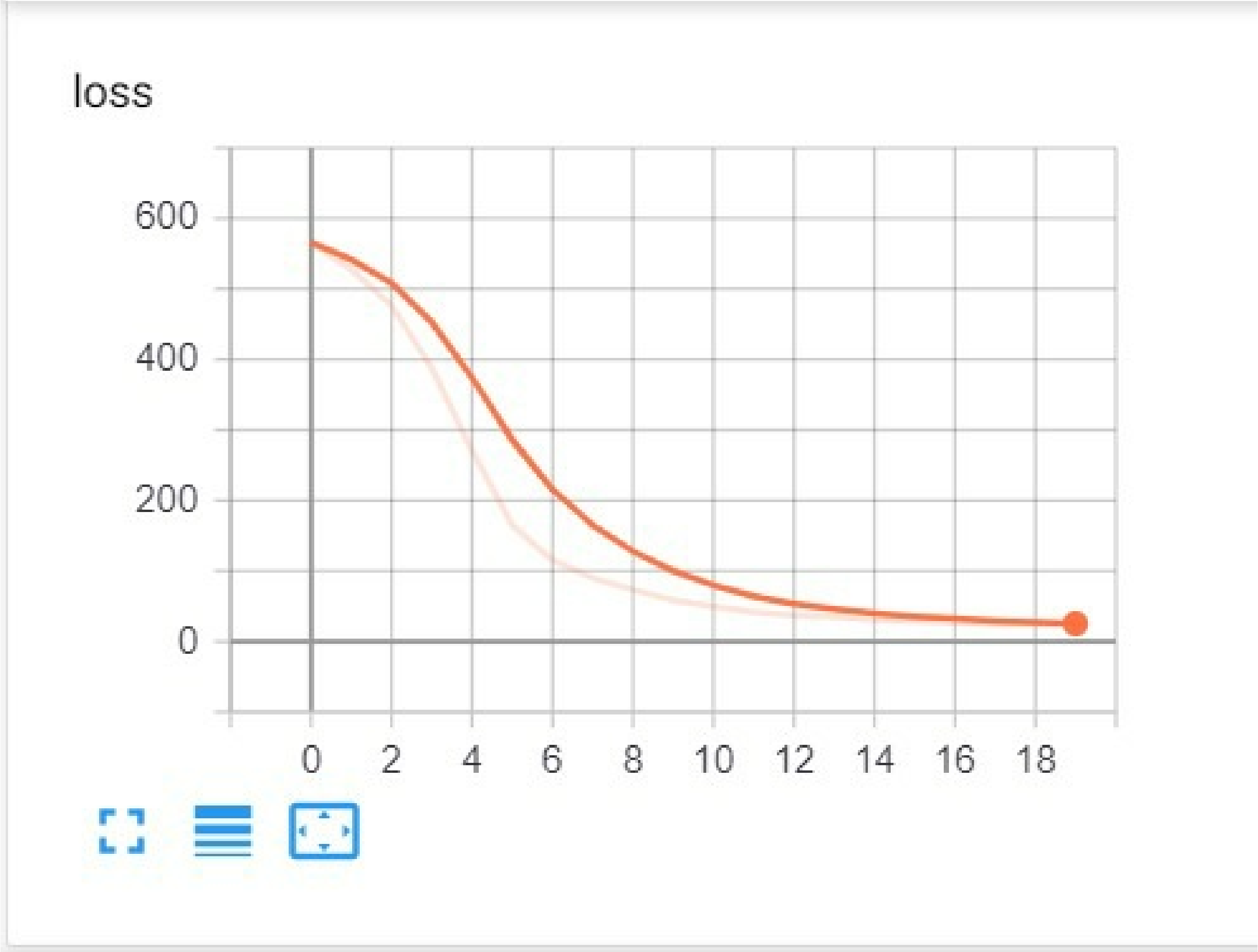
For Linear Regression, we have used boston_housing dataset. The accuracy,loss graphs are shown in tensorboard for different values of learning rate, batch_size and optimizers and activation functions.
The various optimizers used are SGD, adam.
The various activation functions used are Sigmoid,Softmax.

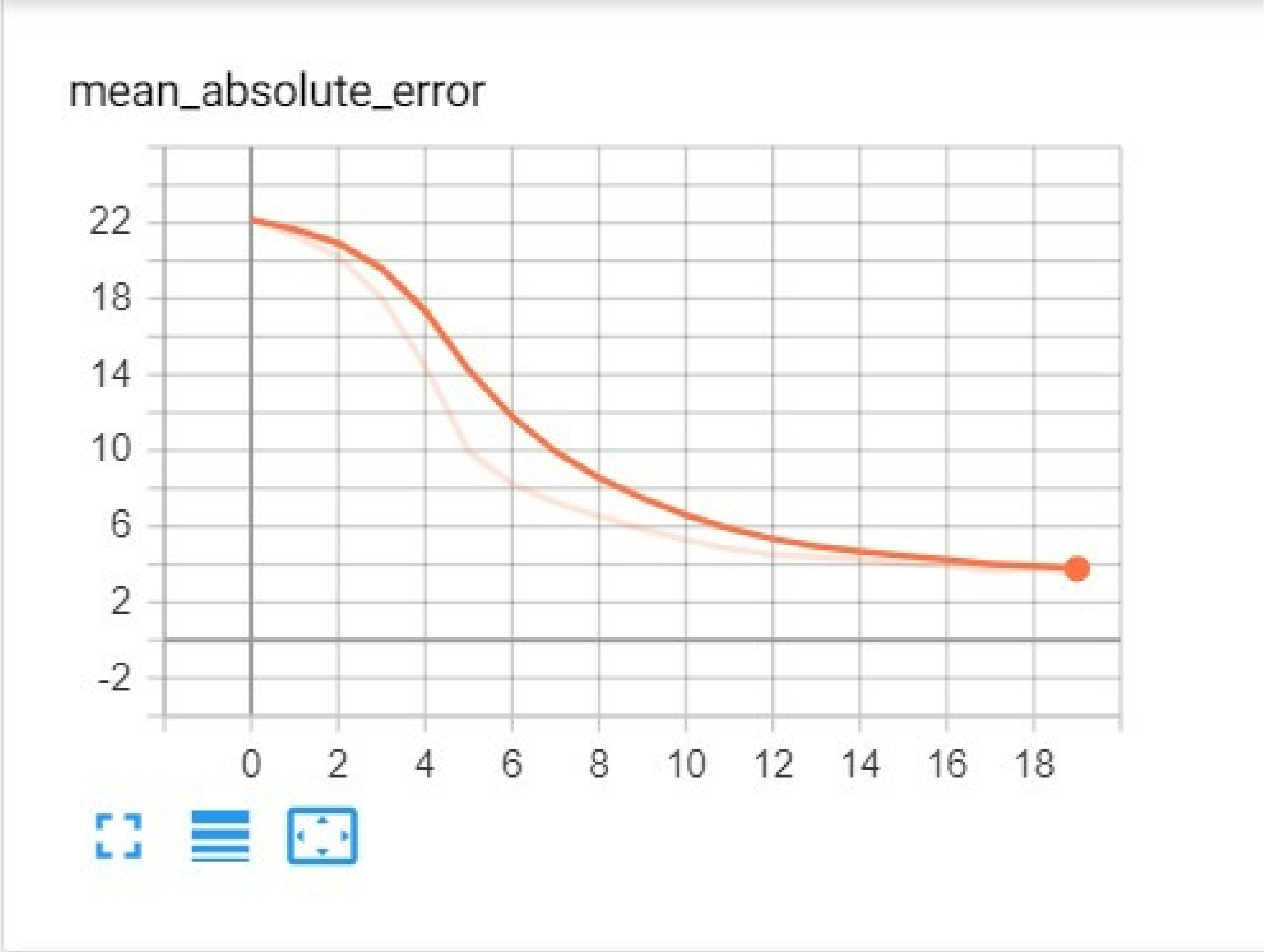
Code & Output:


Lab2 [C:\Users\lohit\Desktop\Lab2] - ...\Task1.py [Lab2] - PyCharm
File Edit View Navigate Code Help
Project
Lab2 C:\Users\lohit\Desktop\Lab2
logs
spam-text-message-classification
Boston.csv
heart.csv
spam-text-message-classification.zip
SPAM_text.csv
Task1.py
Task2.py
Task5.py
External Libraries
Scratches and Consoles
Task2.py
Task1.py
Task3.py
Task.py
Task5.py
1: Project
Run:
Task1

```
    return(t_model)

model = basic_model_2(arr_x_train.shape[1], 1)

model.summary()
epochs = 20
batch_size =32
from keras.callbacks import TensorBoard
tensorboard = TensorBoard(log_dir="logs/{}",histogram_freq=0, write_graph=True, write_images=True)
history = model.fit(arr_x_train, arr_y_train,
    batch_size=batch_size,
    epochs=epochs,
    shuffle=True,
    verbose=2, # Change it to 2, if wished to observe execution
    validation_data=(arr_x_valid, arr_y_valid),callbacks=[tensorboard])

train_score = model.evaluate(arr_x_train, arr_y_train, verbose=0)
valid_score = model.evaluate(arr_x_valid, arr_y_valid, verbose=0)

print('Train MAE: ', round(train_score[1], 4), ', Train Loss: ', round(train_score[0], 4))
print('Val MAE: ', round(valid_score[1], 4), ', Val Loss: ', round(valid_score[0], 4))
```

```
Epoch 15/20
 - 0s - loss: 31.5017 - mean_absolute_error: 4.2993 - val_loss: 38.1334 - val_mean_absolute_error: 4.6813
Epoch 16/20
 - 0s - loss: 29.0367 - mean_absolute_error: 4.0771 - val_loss: 35.5322 - val_mean_absolute_error: 4.4092
Epoch 17/20
 - 0s - loss: 27.8010 - mean_absolute_error: 3.9715 - val_loss: 33.9684 - val_mean_absolute_error: 4.2064
Epoch 18/20
 - 0s - loss: 24.5834 - mean_absolute_error: 3.6532 - val_loss: 32.4663 - val_mean_absolute_error: 4.1327
Epoch 19/20
 - 0s - loss: 25.2664 - mean_absolute_error: 3.7618 - val_loss: 31.1636 - val_mean_absolute_error: 4.0577
Epoch 20/20
 - 0s - loss: 22.7338 - mean_absolute_error: 3.5617 - val_loss: 30.0798 - val_mean_absolute_error: 3.9146
Train MAE:  3.3497 , Train Loss:  21.1155
Val MAE:  3.9146 , Val Loss:  30.0798

Process finished with exit code 0
```





Task-2: Implement the Logistic Regression with any data set of your choice except the datasets being discussed in the class or source code.

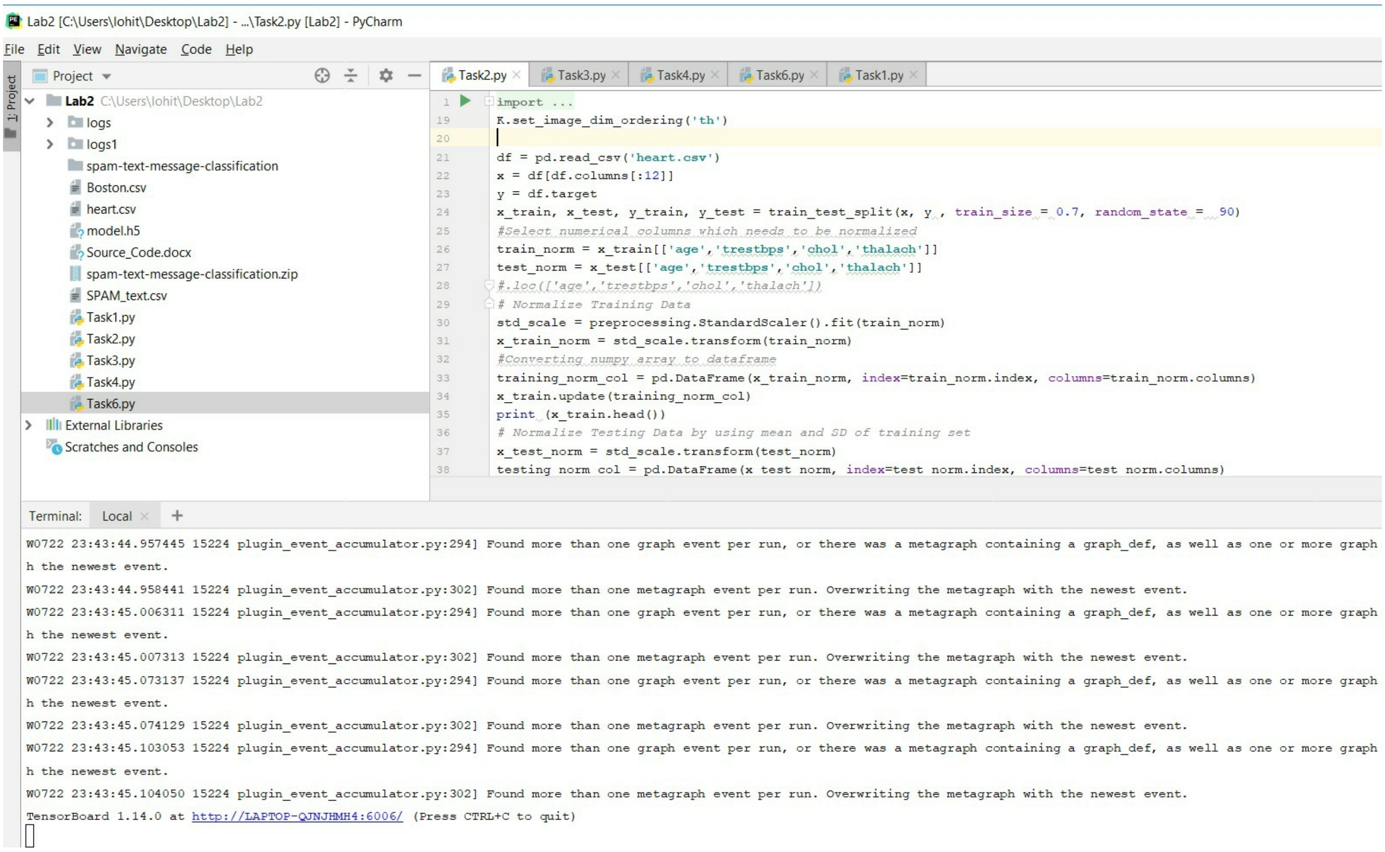
1. Show the graph in TensorBoard
2. Show the Loss in TensorBoard
3. Use score=model.evaluate(x_text,y_test)and then print('test accuracy', score[1])to print the accuracy
4. Change three hyper parameter and report how the accuracy changes

For Logistic Regression, we have used 'heart' dataset.The accuracy,loss graphs are shown in tensorboard for different values of learning rate, batch_size and optimizers and activation functions.

The various optimizers used are SGD, adam.

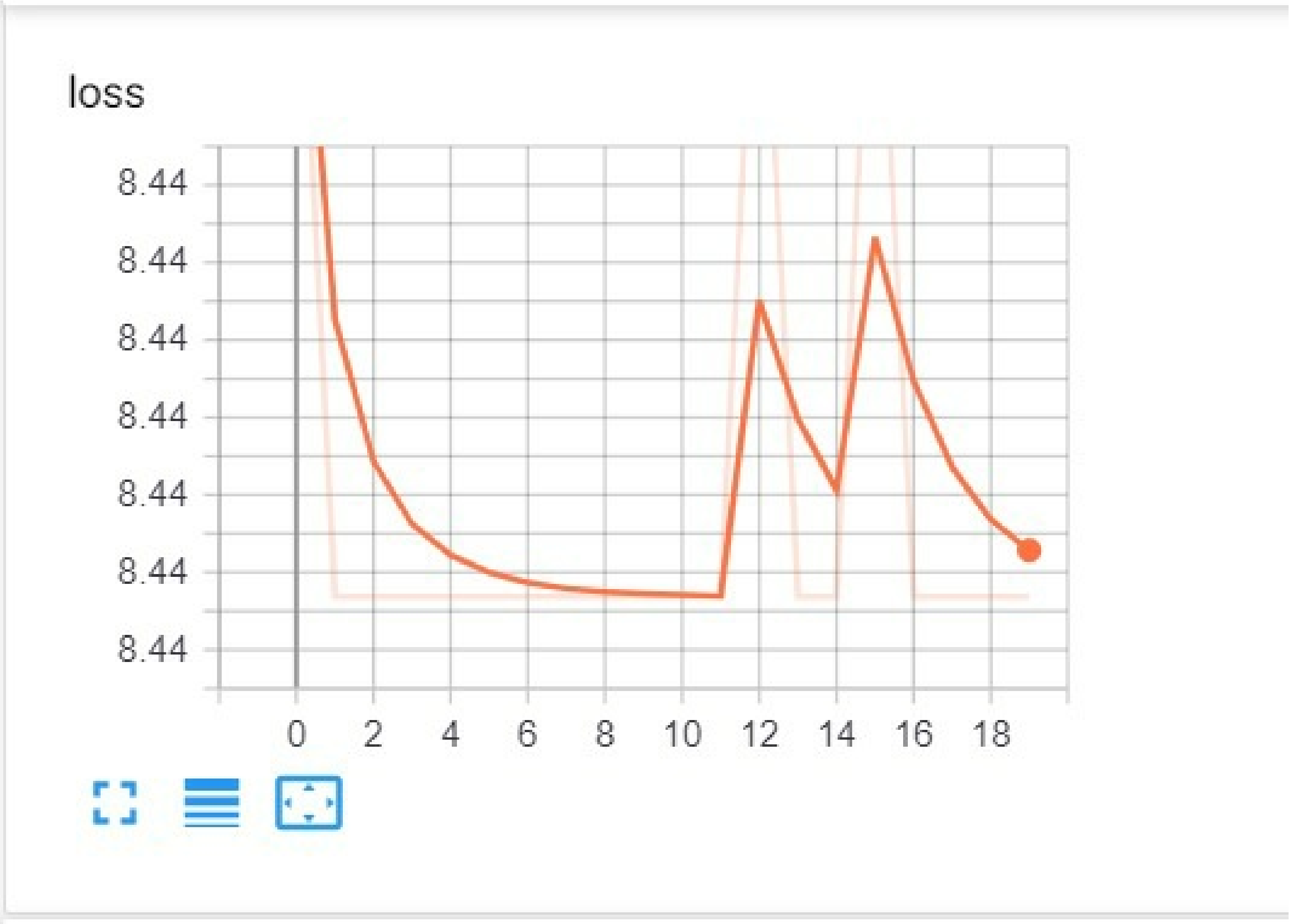
The various activation functions used are Sigmoid,Softmax.

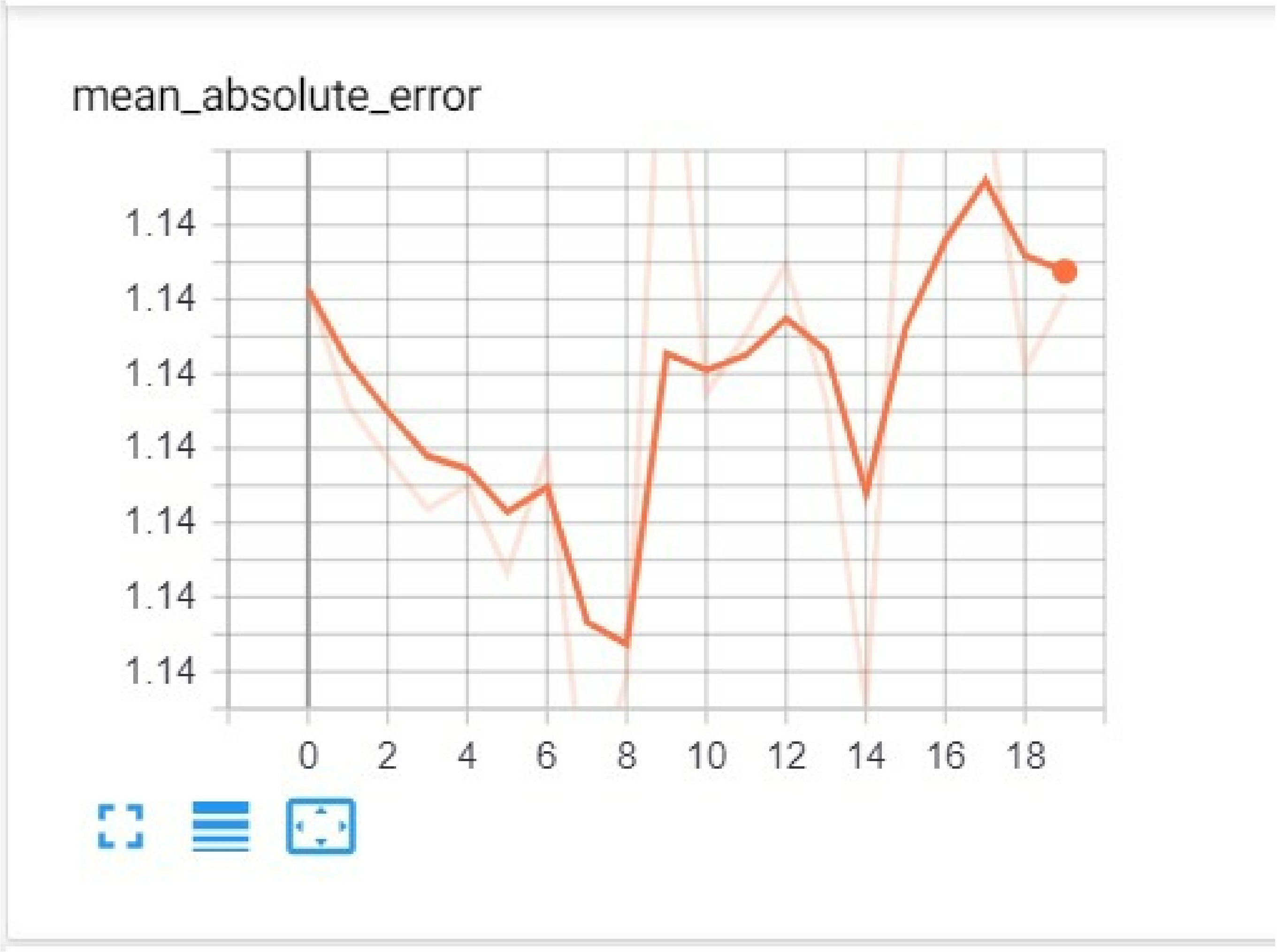
Code & Output:



```
import ...
K.set_image_dim_ordering('th')

df = pd.read_csv('heart.csv')
x = df[df.columns[:12]]
y = df.target
x_train, x_test, y_train, y_test = train_test_split(x, y , train_size = 0.7, random_state =  90)
#Select numerical columns which needs to be normalized
train_norm = x_train[['age','trestbps','chol','thalach']]
test_norm = x_test[['age','trestbps','chol','thalach']]
#.loc(['age','trestbps','chol','thalach'])
# Normalize Training Data
std_scale = preprocessing.StandardScaler().fit(train_norm)
x_train_norm = std_scale.transform(train_norm)
#Converting numpy array to dataframe
training_norm_col = pd.DataFrame(x_train_norm, index=train_norm.index, columns=train_norm.columns)
x_train.update(training_norm_col)
print (x_train.head())
# Normalize Testing Data by using mean and SD of training set
x_test_norm = std_scale.transform(test_norm)
testing_norm_col = pd.DataFrame(x_test_norm, index=test_norm.index, columns=test_norm.columns)
```

```
W0722 23:43:44.957445 15224 plugin_event_accumulator.py:294] Found more than one graph event per run, or there was a metagraph containing a graph_def, as well as one or more graph
h the newest event.
W0722 23:43:44.958441 15224 plugin_event_accumulator.py:302] Found more than one metagraph event per run. Overwriting the metagraph with the newest event.
W0722 23:43:45.006311 15224 plugin_event_accumulator.py:294] Found more than one graph event per run, or there was a metagraph containing a graph_def, as well as one or more graph
h the newest event.
W0722 23:43:45.007313 15224 plugin_event_accumulator.py:302] Found more than one metagraph event per run. Overwriting the metagraph with the newest event.
W0722 23:43:45.073137 15224 plugin_event_accumulator.py:294] Found more than one graph event per run, or there was a metagraph containing a graph_def, as well as one or more graph
h the newest event.
W0722 23:43:45.074129 15224 plugin_event_accumulator.py:302] Found more than one metagraph event per run. Overwriting the metagraph with the newest event.
W0722 23:43:45.103053 15224 plugin_event_accumulator.py:294] Found more than one graph event per run, or there was a metagraph containing a graph_def, as well as one or more graph
h the newest event.
W0722 23:43:45.104050 15224 plugin_event_accumulator.py:302] Found more than one metagraph event per run. Overwriting the metagraph with the newest event.
TensorBoard 1.14.0 at http://LAPTOP-QJNJHMH4:6006/ (Press CTRL+C to quit)
```

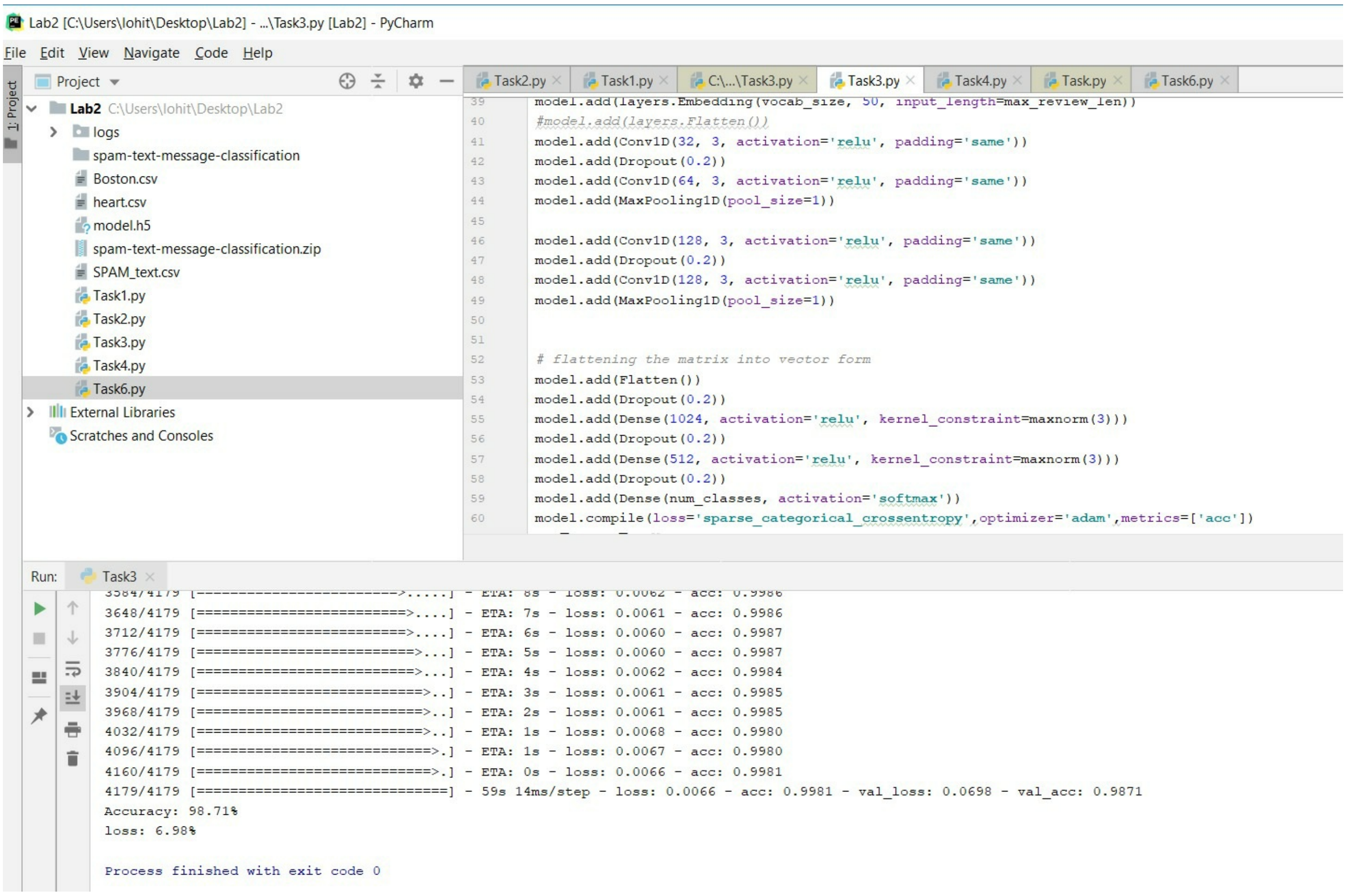

loss
8.44
8.44
8.44
8.44
8.44
8.44
8.44
0
2
4
6
8
10
12
14
16
18



Task-3: Implement the text classification with CNN model on Spam text classification dataset or Reuters dataset.

For this task we have used Spam Text Dataset. We have used 1D Convolutional Kernals, MaxPooling and dropped neurons using dropout to overcome overfitting.

Code & Output:



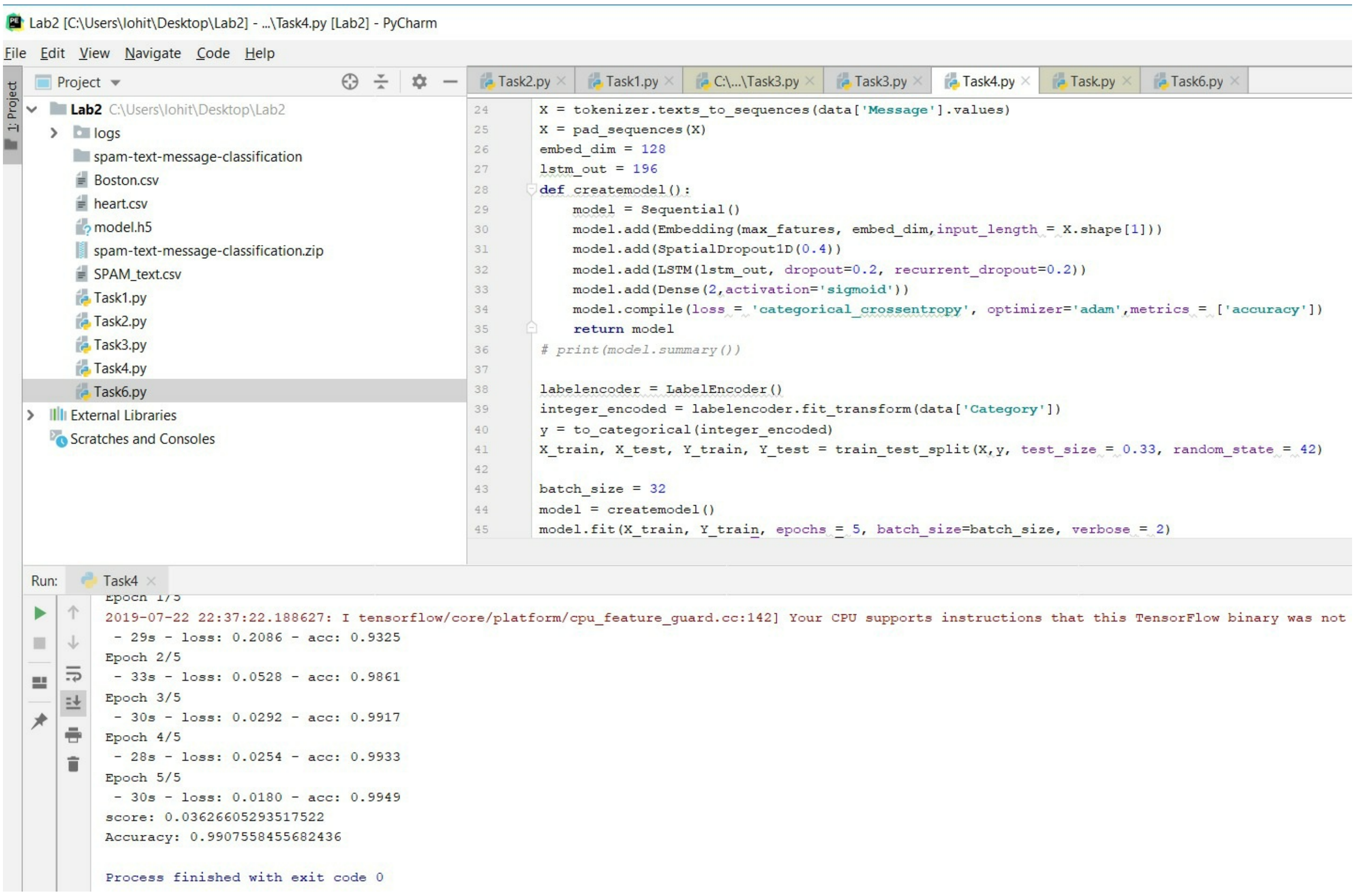
```python
model.add(layers.Embedding(vocab_size, 50, input_length=max_review_len))
#model.add(layers.Flatten())
model.add(Conv1D(32, 3, activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv1D(64, 3, activation='relu', padding='same'))
model.add(MaxPooling1D(pool_size=1))

model.add(Conv1D(128, 3, activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv1D(128, 3, activation='relu', padding='same'))
model.add(MaxPooling1D(pool_size=1))


# flattening the matrix into vector form
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['acc'])
```

```
3584/4179 [========================>.....] - ETA: 8s - loss: 0.0062 - acc: 0.9986
3648/4179 [=========================>....] - ETA: 7s - loss: 0.0061 - acc: 0.9986
3712/4179 [=========================>....] - ETA: 6s - loss: 0.0060 - acc: 0.9987
3776/4179 [==========================>...] - ETA: 5s - loss: 0.0060 - acc: 0.9987
3840/4179 [==========================>...] - ETA: 4s - loss: 0.0062 - acc: 0.9984
3904/4179 [===========================>..] - ETA: 3s - loss: 0.0061 - acc: 0.9985
3968/4179 [===========================>..] - ETA: 2s - loss: 0.0061 - acc: 0.9985
4032/4179 [===========================>..] - ETA: 1s - loss: 0.0068 - acc: 0.9980
4096/4179 [============================>.] - ETA: 1s - loss: 0.0067 - acc: 0.9980
4160/4179 [============================>.] - ETA: 0s - loss: 0.0066 - acc: 0.9981
4179/4179 [==============================] - 59s 14ms/step - loss: 0.0066 - acc: 0.9981 - val_loss: 0.0698 - val_acc: 0.9871
Accuracy: 98.71%
loss: 6.98%

Process finished with exit code 0
```

Task-4: Implement the text classification with LSTM modelon Spam text classification dataset or Reuters dataset.

For this task we have used Spam Text Dataset. Data is tokenized and sequenced.Padding is applied to the sequenced data. LSTM model is applied to this data.If the output is zero the text is identified as not spam.If the output is 1 the text is identified as spam.

Code & Output:



```
X = tokenizer.texts_to_sequences(data['Message'].values)
X = pad_sequences(X)
embed_dim = 128
lstm_out = 196
def createmodel():
    model = Sequential()
    model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1]))
    model.add(SpatialDropout1D(0.4))
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(2,activation='sigmoid'))
    model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
    return model
# print(model.summary())

labelencoder = LabelEncoder()
integer_encoded = labelencoder.fit_transform(data['Category'])
y = to_categorical(integer_encoded)
X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size = 0.33, random_state = 42)

batch_size = 32
model = createmodel()
model.fit(X_train, Y_train, epochs = 5, batch_size=batch_size, verbose = 2)
```

```
Epoch 1/5
2019-07-22 22:37:22.188627: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not
 - 29s - loss: 0.2086 - acc: 0.9325
Epoch 2/5
 - 33s - loss: 0.0528 - acc: 0.9861
Epoch 3/5
 - 30s - loss: 0.0292 - acc: 0.9917
Epoch 4/5
 - 28s - loss: 0.0254 - acc: 0.9933
Epoch 5/5
 - 30s - loss: 0.0180 - acc: 0.9949
score: 0.03626605293517522
Accuracy: 0.9907558455682436

Process finished with exit code 0
```

Task-5: Compare the results of CNN and LSTM models, for the text classification and describe, which model is best for the text classification based on your results.

For text classification with CNN model, accuracy is 98%

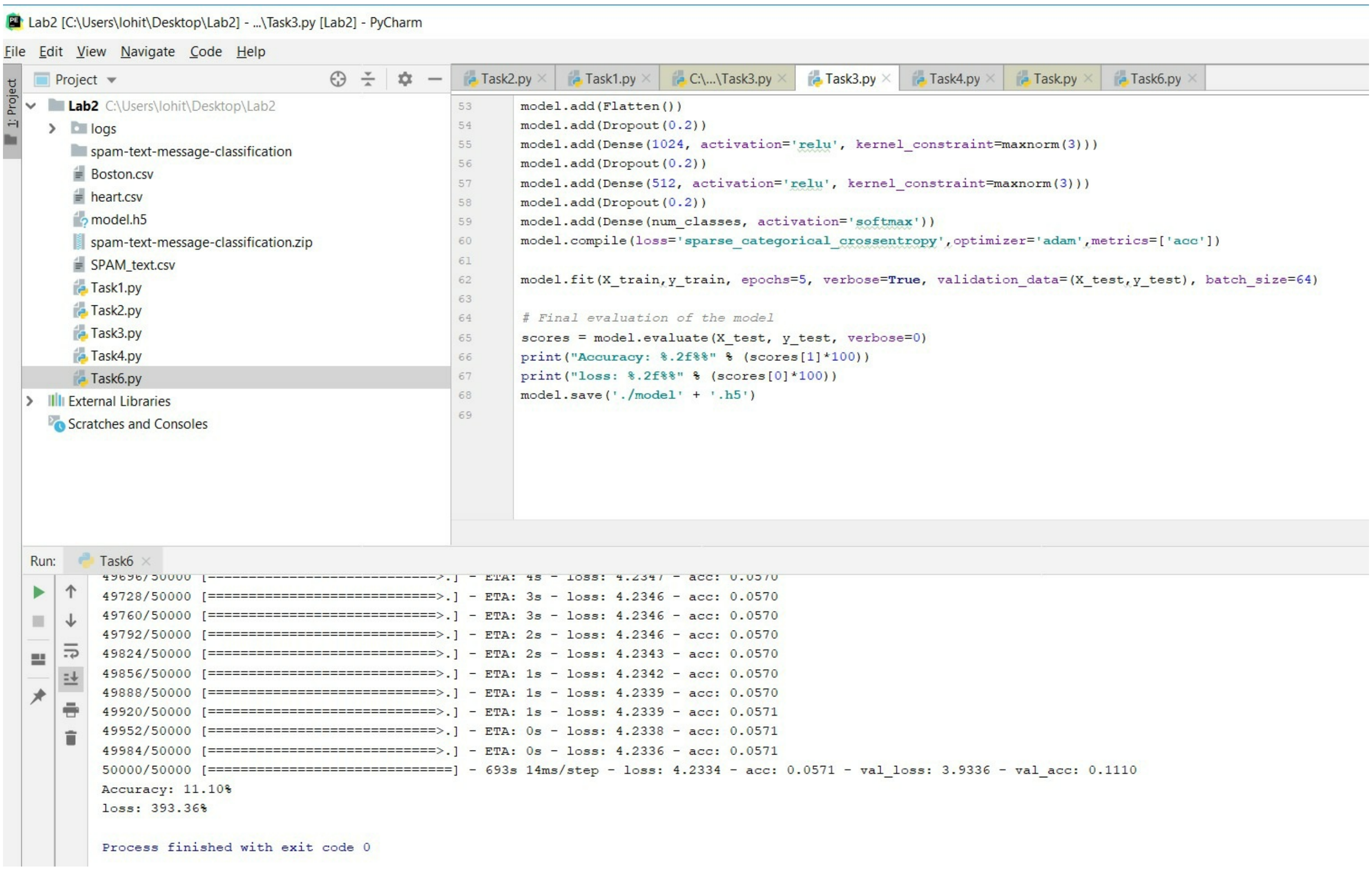
For text classification with LSTM model, accuracy is 99%

The accuracy for LSTM model is higher compared to CNN model.As we are dropping the features recurrently in LSTM,the accuracy is higher. LSTM is slow for which it gives different accuracies for each time the text is given as input, where it trains for each continuous text data feed. So, LSTM model is better for text classification compared to CNN.

Code & Output:

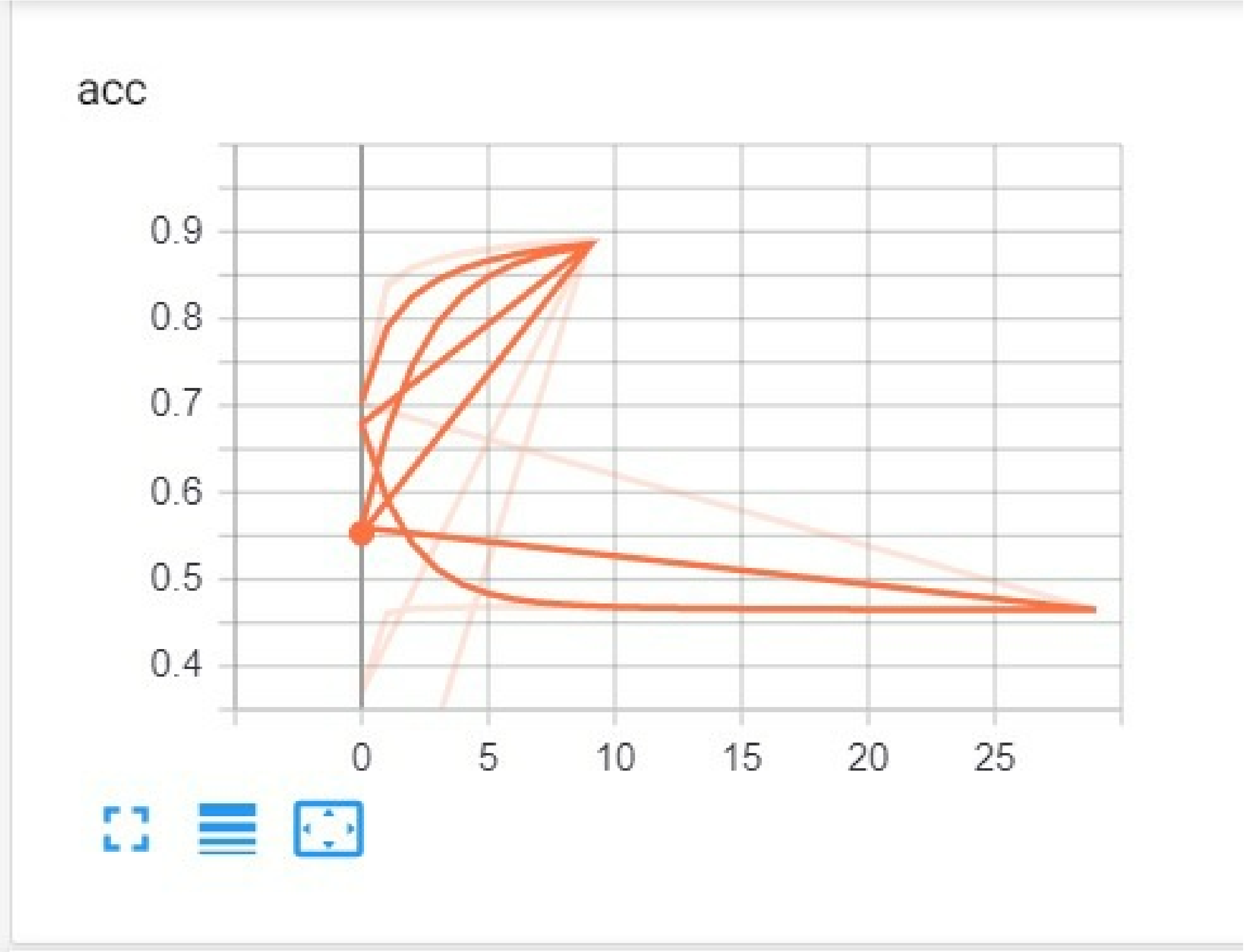
Task-6: Implement the image classification with CNN model, with a new dataset which is not used in the class.

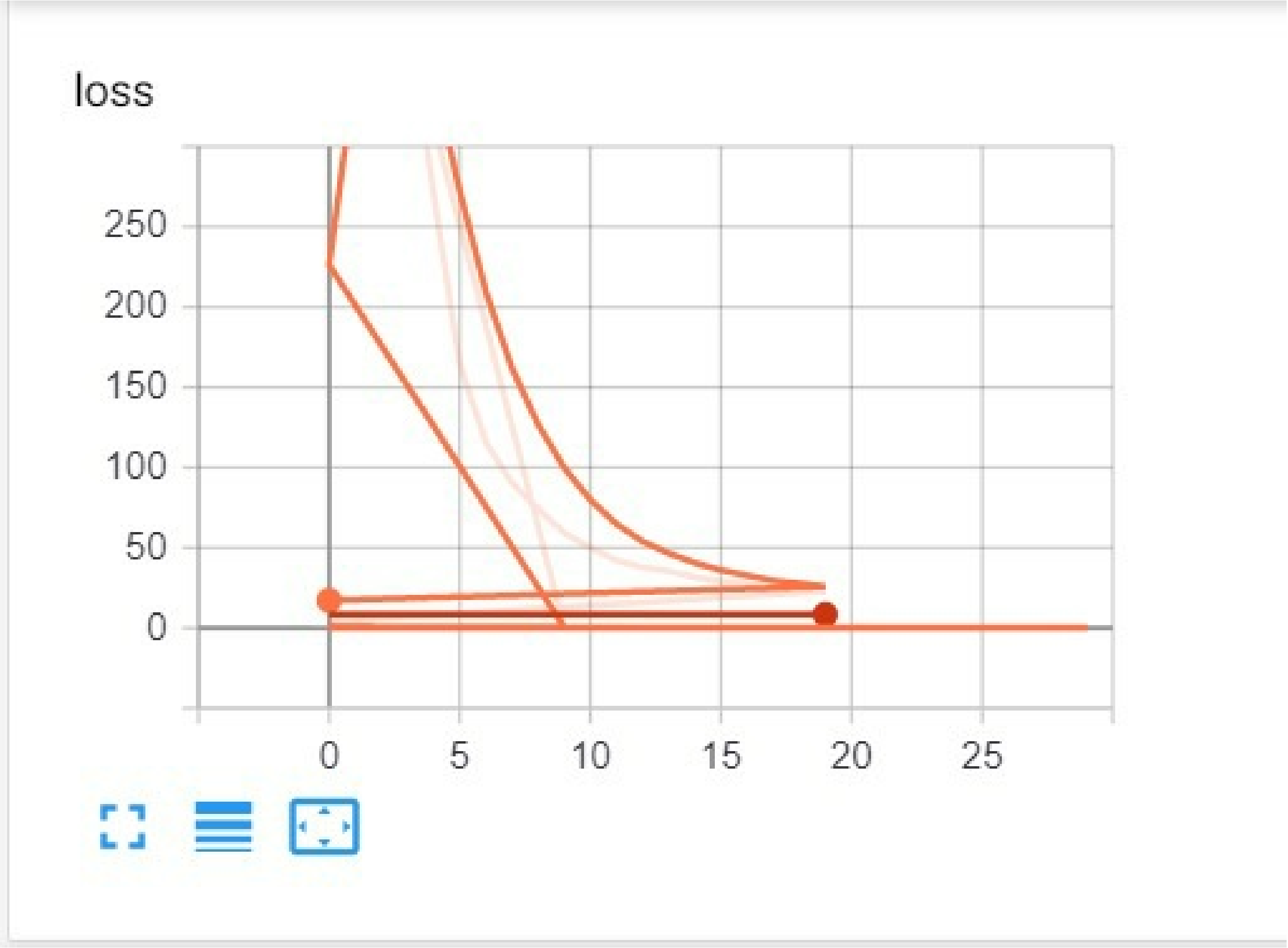
Code & Output:



```
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['acc'])

model.fit(X_train,y_train, epochs=5, verbose=True, validation_data=(X_test,y_test), batch_size=64)

# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
print("loss: %.2f%%" % (scores[0]*100))
model.save('./model' + '.h5')
```

```
49696/50000 [============================>.] - ETA: 4s - loss: 4.2347 - acc: 0.0570
49728/50000 [============================>.] - ETA: 3s - loss: 4.2346 - acc: 0.0570
49760/50000 [============================>.] - ETA: 3s - loss: 4.2346 - acc: 0.0570
49792/50000 [============================>.] - ETA: 2s - loss: 4.2346 - acc: 0.0570
49824/50000 [============================>.] - ETA: 2s - loss: 4.2343 - acc: 0.0570
49856/50000 [============================>.] - ETA: 1s - loss: 4.2342 - acc: 0.0570
49888/50000 [============================>.] - ETA: 1s - loss: 4.2339 - acc: 0.0570
49920/50000 [============================>.] - ETA: 1s - loss: 4.2339 - acc: 0.0571
49952/50000 [============================>.] - ETA: 0s - loss: 4.2338 - acc: 0.0571
49984/50000 [============================>.] - ETA: 0s - loss: 4.2336 - acc: 0.0571
50000/50000 [==============================] - 693s 14ms/step - loss: 4.2334 - acc: 0.0571 - val_loss: 3.9336 - val_acc: 0.1110
Accuracy: 11.10%
loss: 393.36%

Process finished with exit code 0
```


acc
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25



Conclusion:

We have understood and implemented the above-mentioned concepts.The loss and accuracy graphs are plotted in tensorboard. Linear Regression , Logistic Regression models are evaluated. Text Classification is implemented using CNN and LSTM models and the accuracies are compared. Image classification is implemented using CNN.