CS5590 – Deep Learning – Lab Assignment 3 Report
Comparison of Text Classification using Standard RNN, LSTM and CNN
Class Id 52: Vinay Chandra Vasamsetti

I. Introduction

Text classification is a very classic problem, where the goal is to classify text into predefined categories. The task for this assignment to apply various methods (Standard RNN, LSTM and CNN) to perform Text classification for a given dataset.

Standard RNNs suffer from vanishing gradient problems. LSTM deals with this problem by introducing various gates for better preservation of long-range dependencies. CNN is well suited for image recognition or text classification.

For simplicity of the code, Keras is used instead of TensorFlow. Also the dataset used is from the Keras datasets. The dataset used is IMDB dataset from Keras, where the goal is to classify movie reviews into sentiment positive or negative.

II. Objectives

The objective is to compare various neural network models (Standard RNN, LSTM and CNN) for text classification and evaluate the performance based on train accuracy and validation accuracy.

III. Approaches / Methods

The common steps involved in building the models are:

1. Load the data from Keras datasets.
2. Preprocess the data to make them in a sequence and reshape them.
3. Set hyper parameters as: max features = 5000, max length = 100, no of classes = 1, batch size = 32, dropout rate = 0.5, no of epochs = 10 etc.
4. Build the model for RNN, LSTM and CNN.
5. Configure the model by compiling with metric as ‘accuracy’, optimizer as ‘adam’ and function as ‘binary_crossentropy’.
6. Set the logdir for TensorBoard to display graphs and scalar metrics.
7. Train the model and calculate the train accuracy and the validation accuracy.

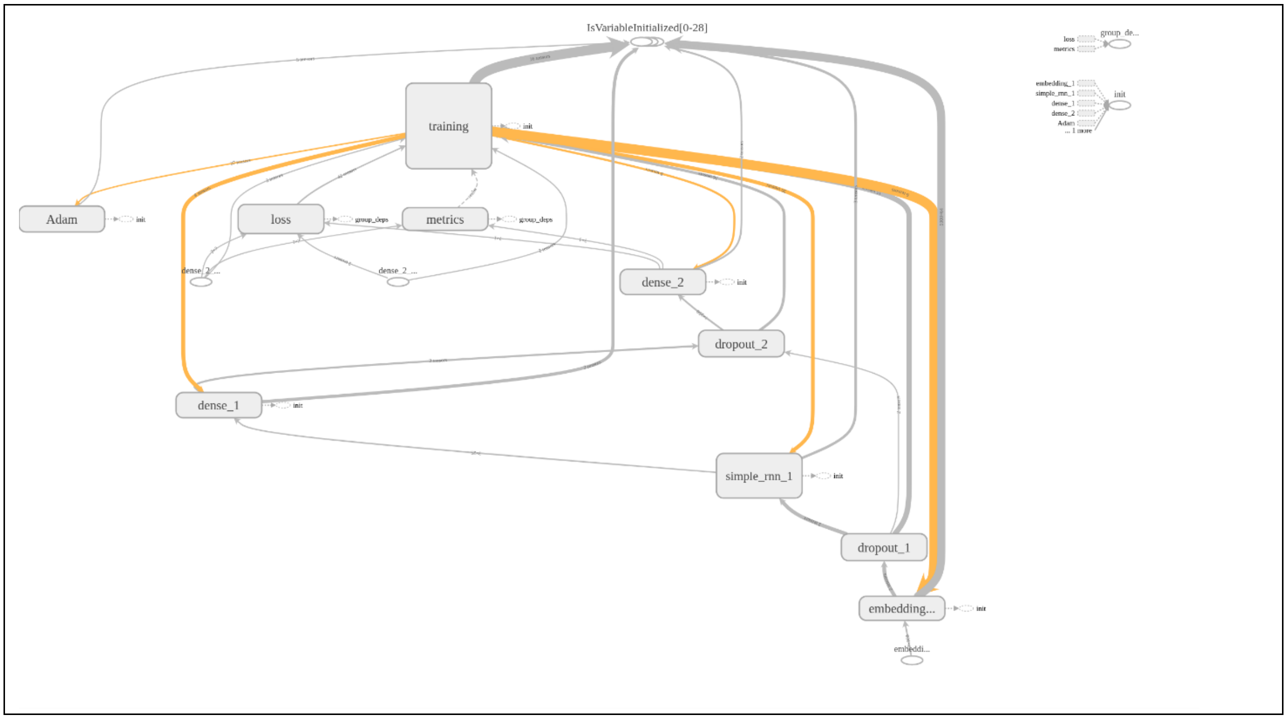
RNN Architecture: Embedding Layer, Dropout Layer, SimpleRNN Layer, Hidden Dense Layer, Dropout Layer, Output Dense Layer

LSTM Architecture: Embedding Layer, BiDirectional LSTM Layer, Dropout Layer, Output Dense Layer

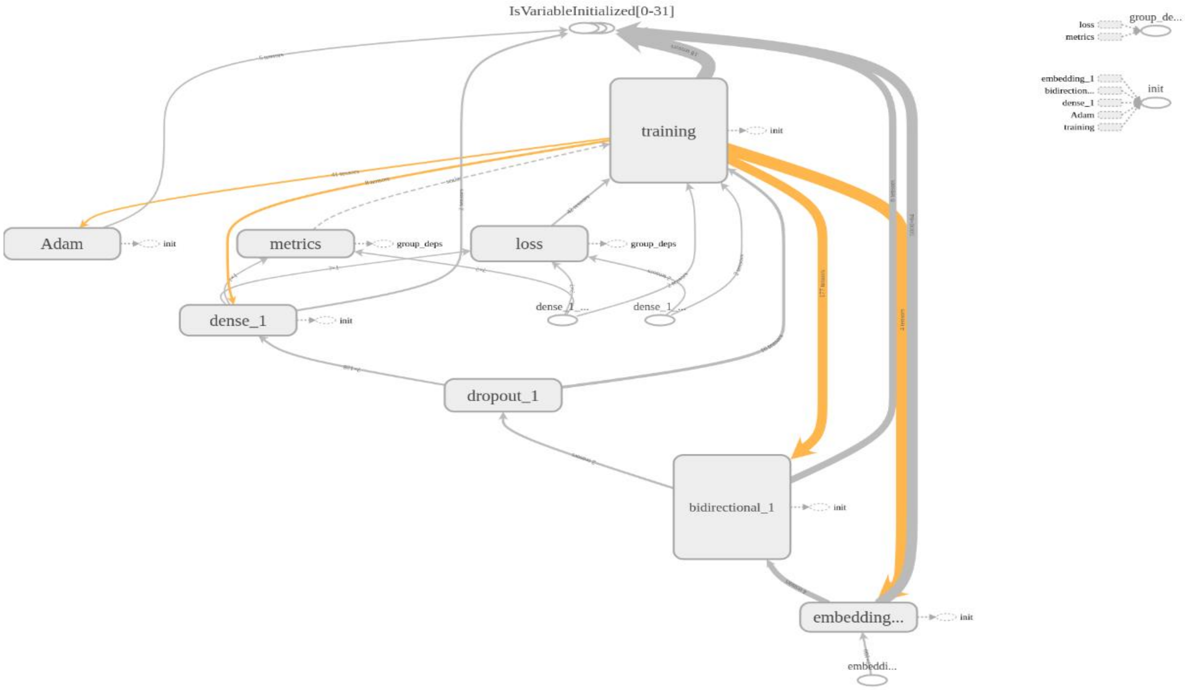
CNN Architecture: Embedding Layer, Dropout Layer, 1D Convolutional Layer, MaxPooling Layer, Hidden Dense Layer, Dropout Layer, Output Dense Layer

IV. Workflow

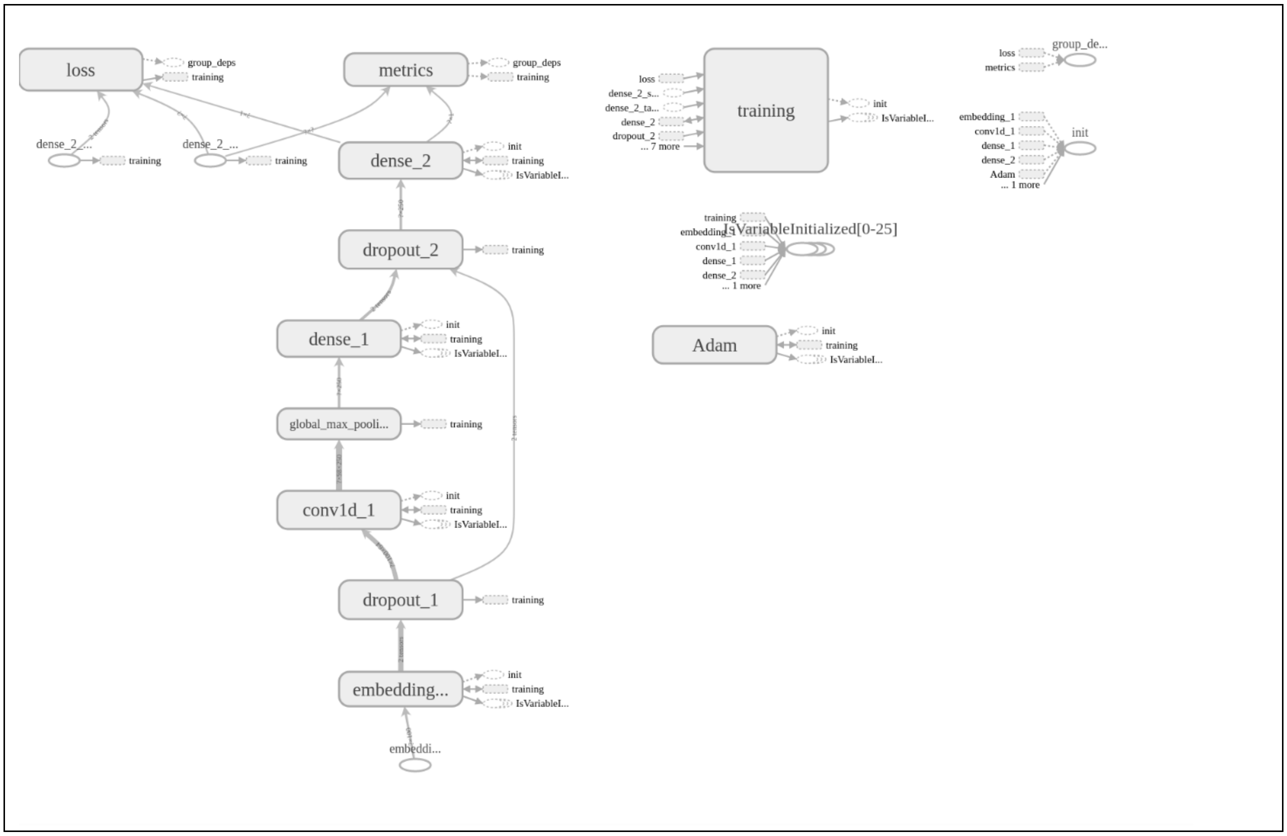
The graph for SimpleRNN can be visualized in TensorBoard as :



The graph for LSTM can be visualized in TensorBoard as:



The graph for CNN can be visualized in TensorBoard as:



V. Datasets

The dataset comprises of 25000 movie surveys from IMDB. The labels are assumption positive or negative. More subtle elements can be found on keras.io/datasets link.

VI. Parameters

The hyperparmeters are as follows:

no_epochs = 10

no_classes = 1

dropout_rate = 0.5

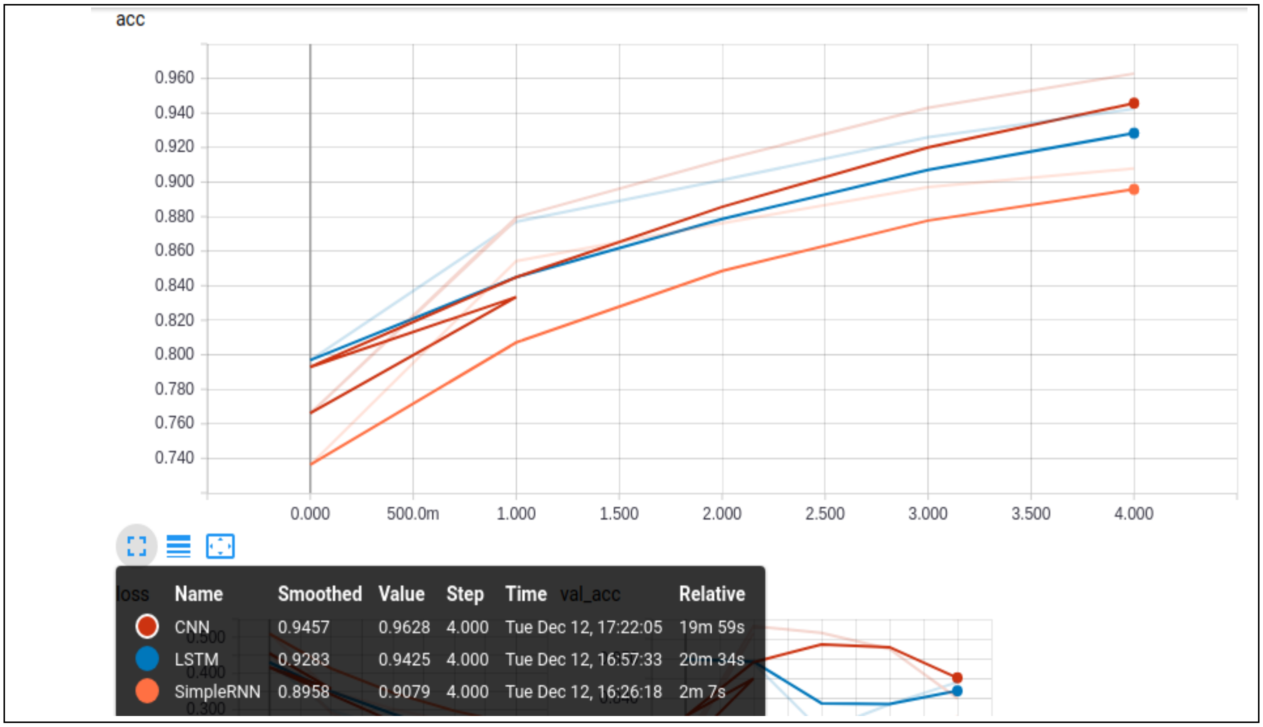
batch_size = 32

embedding_size = 64

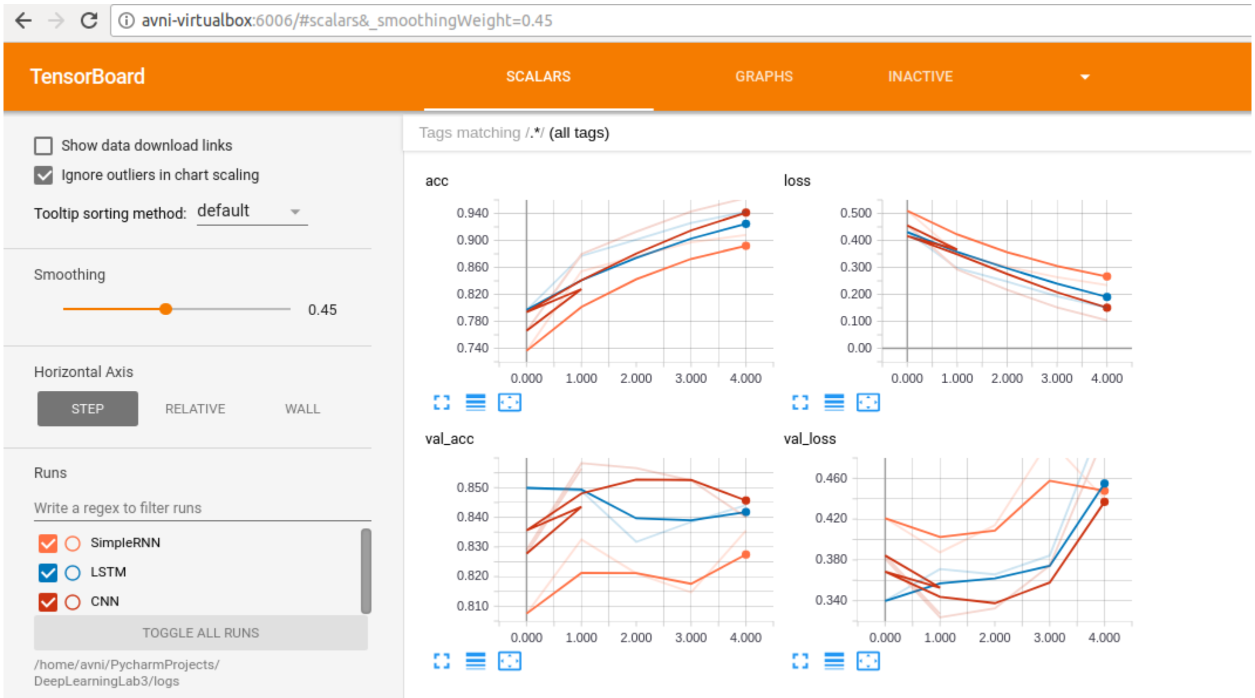
kernel_size = 3

VII. Evaluation & Discussion

Three models were created and were run on the IMDB dataset for 10 epochs.



Name	Smoothed	Value	Step	Time	Relative
● CNN	0.9457	0.9628	4.000	Tue Dec 12, 17:22:05	19m 59s
● LSTM	0.9283	0.9425	4.000	Tue Dec 12, 16:57:33	20m 34s
● SimpleRNN	0.8958	0.9079	4.000	Tue Dec 12, 16:26:18	2m 7s



avni-virtualbox:6006/#scalars&_smoothingWeight=0.45

Tags matching /.*/ (all tags)

□ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.45

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

☑ SimpleRNN

☑ LSTM

☑ CNN

TOGGLE ALL RUNS

/home/avni/PycharmProjects/DeepLearningLab3/logs

From the above graphs for train accuracy and validation accuracy, we see that CNN and LSTM both are good models for text classification. RNN fall behind in terms of accuracy for both train and validation accuracy.

After running all three programs we observe the following:

Model	Train Accuracy Range	Validation Accuracy Range
SimpleRNN	73.63% (Epoch1) to 90.79% (Epoch5)	80.75% (Epoch1) to 83.55% (Epoch5)
LSTM	79.69% (Epoch1) to 94.25% (Epoch5)	84.99.63% (Epoch1) to 84.41% (Epoch5)
CNN	76.60% (Epoch1) to 96.28% (Epoch5)	82.91% (Epoch1) to 84% (Epoch5)

VIII. Conclusion

Convolution Neural Network proves to be the best model out of three. It has greatest train accuracy of 96.28% and its test accuracy is also high. Bi-directional LSTM comes closely behind CNN with train accuracy of 94.25%, while its test accuracy drops a bit. Standard RNN gives the worst performance with train accuracy of just 90.79 and test accuracy of 83.55%. RNN has the issue of forgetting past data for longer steps.

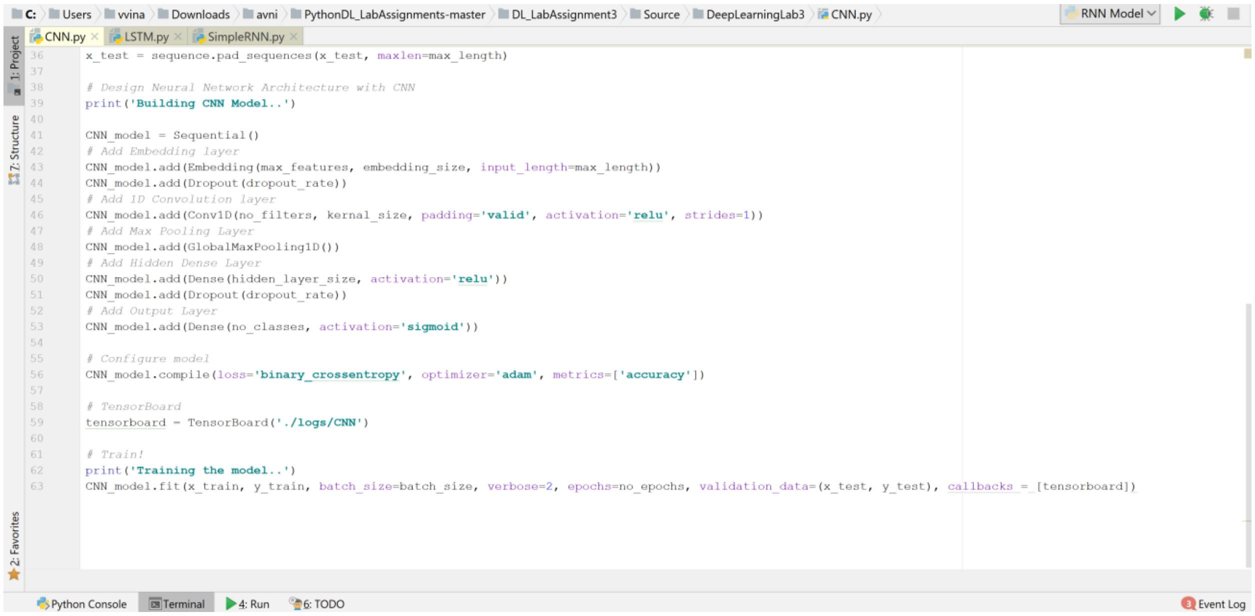
In conclusion, for IMDB dataset text classification, either CNN model or LSTM would be a good fit. When we increase the no. of epochs to 200, we could more precisely determine the best model out of all three.

REFERENCE :

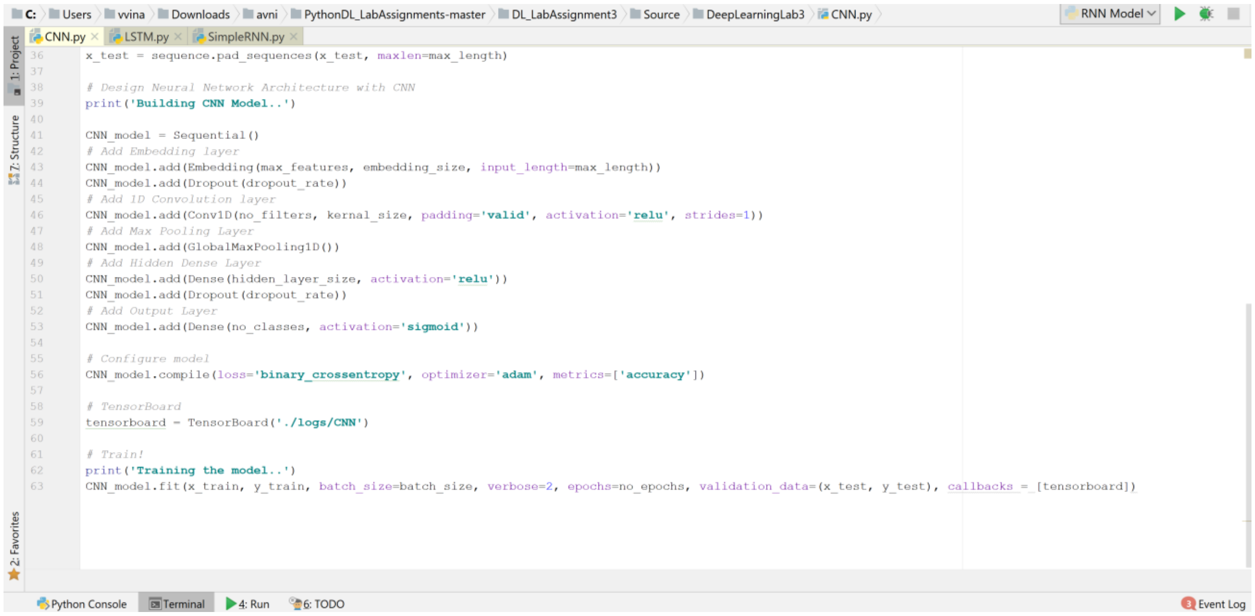
https://github.com/stratospark/food-101-keras
https://github.com/matterport/Mask_RCNN
http://blog.stratospark.com/deep-learning-applied-food-classification-deep-learning-keras.html

CODE FOR THREE TASKS :

1)

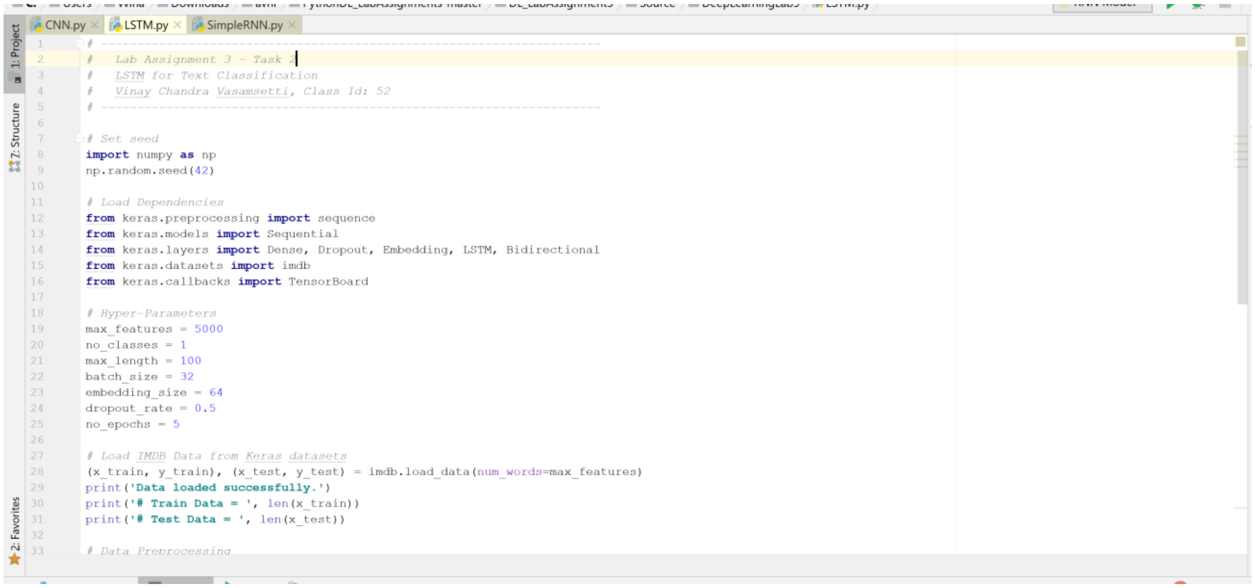


```
x_test = sequence.pad_sequences(x_test, maxlen=max_length)

# Design Neural Network Architecture with CNN
print('Building CNN Model..')

CNN_model = Sequential()
# Add Embedding layer
CNN_model.add(Embedding(max_features, embedding_size, input_length=max_length))
CNN_model.add(Dropout(dropout_rate))
# Add 1D Convolution layer
CNN_model.add(Conv1D(no_filters, kernal_size, padding='valid', activation='relu', strides=1))
# Add Max Pooling Layer
CNN_model.add(GlobalMaxPooling1D())
# Add Hidden Dense Layer
CNN_model.add(Dense(hidden_layer_size, activation='relu'))
CNN_model.add(Dropout(dropout_rate))
# Add Output Layer
CNN_model.add(Dense(no_classes, activation='sigmoid'))

# Configure model
CNN_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# TensorBoard
tensorboard = TensorBoard('./logs/CNN')

# Train!
print('Training the model..')
CNN_model.fit(x_train, y_train, batch_size=batch_size, verbose=2, epochs=no_epochs, validation_data=(x_test, y_test), callbacks = [tensorboard])
```

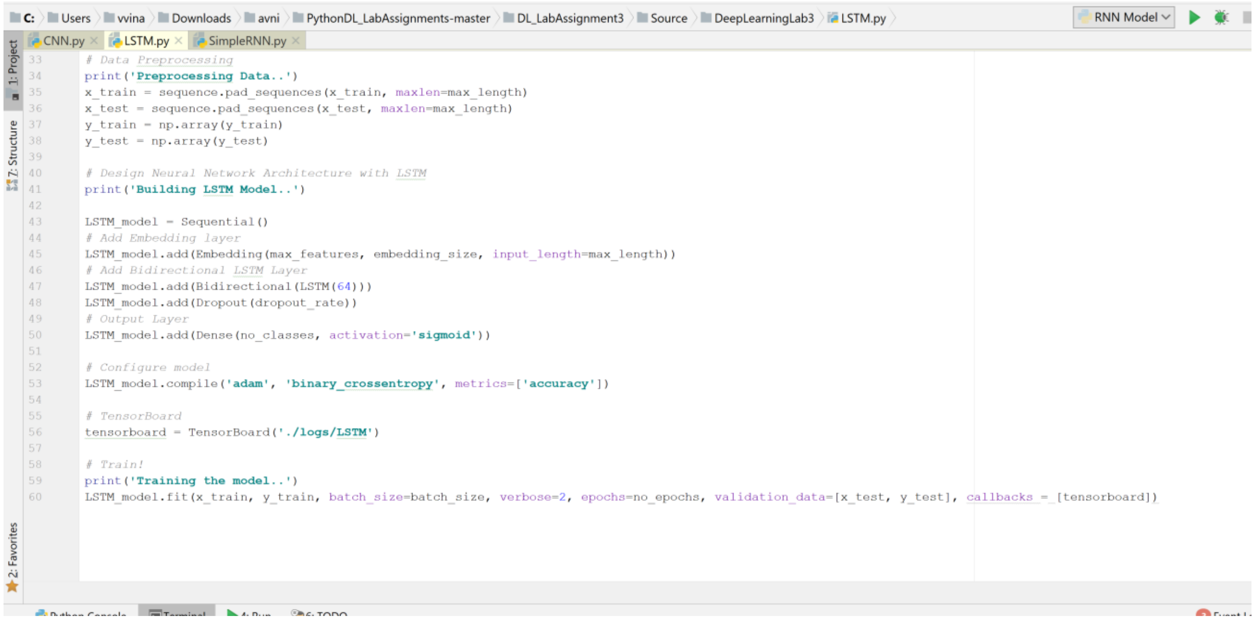


```
x_test = sequence.pad_sequences(x_test, maxlen=max_length)

# Design Neural Network Architecture with CNN
print('Building CNN Model..')

CNN_model = Sequential()
# Add Embedding layer
CNN_model.add(Embedding(max_features, embedding_size, input_length=max_length))
CNN_model.add(Dropout(dropout_rate))
# Add 1D Convolution layer
CNN_model.add(Conv1D(no_filters, kernal_size, padding='valid', activation='relu', strides=1))
# Add Max Pooling Layer
CNN_model.add(GlobalMaxPooling1D())
# Add Hidden Dense Layer
CNN_model.add(Dense(hidden_layer_size, activation='relu'))
CNN_model.add(Dropout(dropout_rate))
# Add Output Layer
CNN_model.add(Dense(no_classes, activation='sigmoid'))

# Configure model
CNN_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# TensorBoard
tensorboard = TensorBoard('./logs/CNN')

# Train!
print('Training the model..')
CNN_model.fit(x_train, y_train, batch_size=batch_size, verbose=2, epochs=no_epochs, validation_data=(x_test, y_test), callbacks = [tensorboard])
```

2)

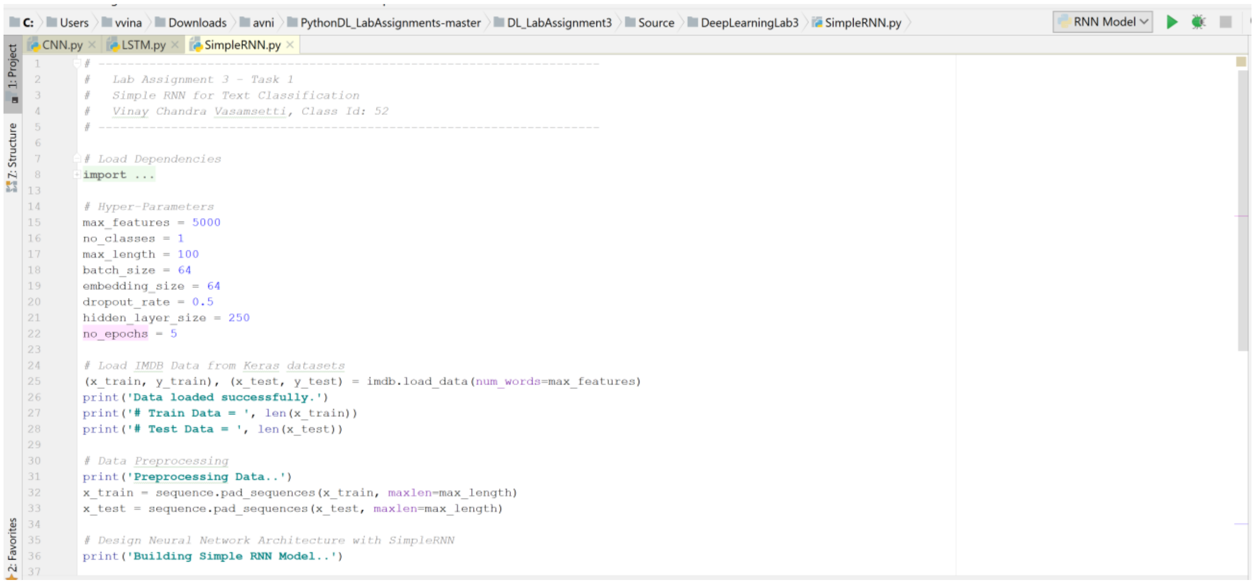


```
# ---------------------------------------------------------------------
#   Lab Assignment 3 - Task 2
#   LSTM for Text Classification
#   Vinay Chandra Vasamsetti, Class Id: 52
# ---------------------------------------------------------------------

# Set seed
import numpy as np
np.random.seed(42)

# Load Dependencies
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Embedding, LSTM, Bidirectional
from keras.datasets import imdb
from keras.callbacks import TensorBoard

# Hyper-Parameters
max_features = 5000
no_classes = 1
max_length = 100
batch_size = 32
embedding_size = 64
dropout_rate = 0.5
no_epochs = 5

# Load IMDB Data from Keras datasets
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print('Data loaded successfully.')
print('# Train Data = ', len(x_train))
print('# Test Data = ', len(x_test))

# Data Preprocessing
```

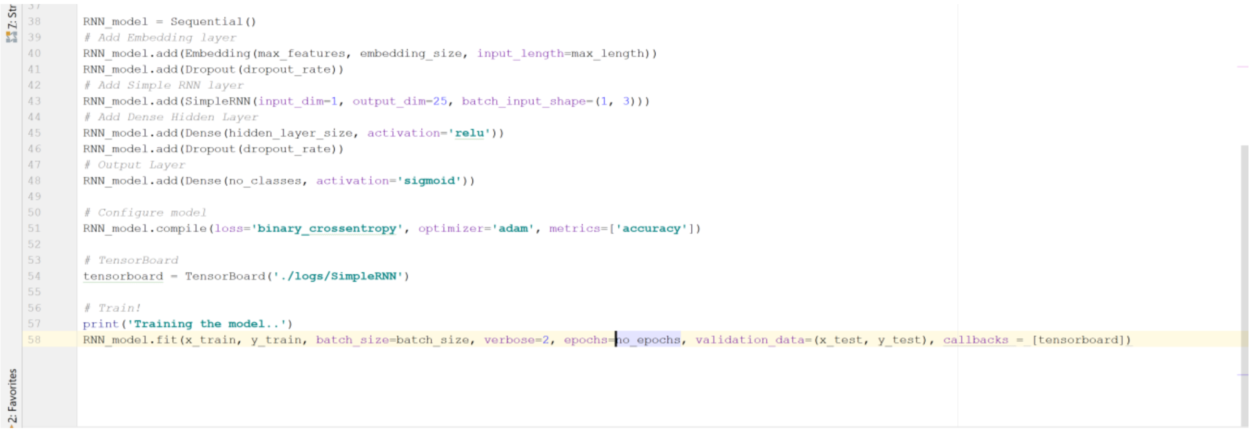

C: › Users › vvina › Downloads › avni › PythonDL_LabAssignments-master › DL_LabAssignment3 › Source › DeepLearningLab3 › LSTM.py
RNN Model
CNN.py
LSTM.py
SimpleRNN.py
1: Project
7: Structure
2: Favorites

```
# Data Preprocessing
print('Preprocessing Data..')
x_train = sequence.pad_sequences(x_train, maxlen=max_length)
x_test = sequence.pad_sequences(x_test, maxlen=max_length)
y_train = np.array(y_train)
y_test = np.array(y_test)

# Design Neural Network Architecture with LSTM
print('Building LSTM Model..')

LSTM_model = Sequential()
# Add Embedding layer
LSTM_model.add(Embedding(max_features, embedding_size, input_length=max_length))
# Add Bidirectional LSTM Layer
LSTM_model.add(Bidirectional(LSTM(64)))
LSTM_model.add(Dropout(dropout_rate))
# Output Layer
LSTM_model.add(Dense(no_classes, activation='sigmoid'))

# Configure model
LSTM_model.compile('adam', 'binary_crossentropy', metrics=['accuracy'])

# TensorBoard
tensorboard = TensorBoard('./logs/LSTM')

# Train!
print('Training the model..')
LSTM_model.fit(x_train, y_train, batch_size=batch_size, verbose=2, epochs=no_epochs, validation_data=[x_test, y_test], callbacks = [tensorboard])
```

3)


C: › Users › vvina › Downloads › avni › PythonDL_LabAssignments-master › DL_LabAssignment3 › Source › DeepLearningLab3 › SimpleRNN.py
RNN Model
CNN.py
LSTM.py
SimpleRNN.py
1: Project
7: Structure
2: Favorites

```
# ----------------------------------------------------------------------
#   Lab Assignment 3 - Task 1
#   Simple RNN for Text Classification
#   Vinay Chandra Vasamsetti, Class Id: 52
# ----------------------------------------------------------------------

# Load Dependencies
import ...

# Hyper-Parameters
max_features = 5000
no_classes = 1
max_length = 100
batch_size = 64
embedding_size = 64
dropout_rate = 0.5
hidden_layer_size = 250
no_epochs = 5

# Load IMDB Data from Keras datasets
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print('Data loaded successfully.')
print('# Train Data = ', len(x_train))
print('# Test Data = ', len(x_test))

# Data Preprocessing
print('Preprocessing Data..')
x_train = sequence.pad_sequences(x_train, maxlen=max_length)
x_test = sequence.pad_sequences(x_test, maxlen=max_length)

# Design Neural Network Architecture with SimpleRNN
print('Building Simple RNN Model..')
```



```
RNN_model = Sequential()
# Add Embedding layer
RNN_model.add(Embedding(max_features, embedding_size, input_length=max_length))
RNN_model.add(Dropout(dropout_rate))
# Add Simple RNN layer
RNN_model.add(SimpleRNN(input_dim=1, output_dim=25, batch_input_shape=(1, 3)))
# Add Dense Hidden Layer
RNN_model.add(Dense(hidden_layer_size, activation='relu'))
RNN_model.add(Dropout(dropout_rate))
# Output Layer
RNN_model.add(Dense(no_classes, activation='sigmoid'))

# Configure model
RNN_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# TensorBoard
tensorboard = TensorBoard('./logs/SimpleRNN')

# Train!
print('Training the model..')
RNN_model.fit(x_train, y_train, batch_size=batch_size, verbose=2, epochs=no_epochs, validation_data=(x_test, y_test), callbacks = [tensorboard])
```