

DEEP LEARNING LAB ASSIGNMENT – 3

Introduction:

The given task is to implement Text Classification using CNN, RNN and LSTM. Build and run the models on a new dataset on pycharm. We also compare the models based on some metrics and conclude which is the best model to the considered dataset.

Objectives:

The main objective is to implement the text classification on the IMDB dataset using CNN, RNN and LSTM. We build the model first and run on the dataset. Then using the accuracy and loss we need to see which model suits for the given dataset. During the building of the model, we take 25,000 train and 25,000 test samples.

Approaches:

Firstly, we set the parameters and load the dataset. In this assignment, I had made 25,000 train and 25000 test samples. We build the model and add the embedding layer which gives the word embeddings. We then add a convolution layer and then do the max pooling to decrease the dimension. We then add the hidden layer which are activated by ReLU and project to the single unit output layer which uses Sigmoid function. We then test our trained model using the test samples and calculate the accuracy and loss. Based on those metrics, we will choose which of the three models is a best suit for the dataset. We use same dataset for all the three implementations, so that we can compare the performance of them equally.

Workflow:

1. We first import the required classes, libraries and functions used in the program.
2. We load the IMDB dataset and confine it to only 5000 words and split into test and train datasets by 50%. Each sentence is limited to have 400 words each.
3. With this dataset, we build our model starting building with the embedding layer.
4. We then apply 'Convolution2D' in CNN, 'LSTM' in LSTM and 'SimpleRNN' in RNN models.
5. We use ReLU at the hidden layer and sigmoid at the output layer.
6. The model will be trained for 2 epochs with the batch size of 1000.
7. We use the 'Adam' optimizer to optimize the loss.
8. We then calculate the accuracy and loss for every iteration and calculate the avg of the epoch and compare the values to conclude the best model for the dataset.

Dataset:

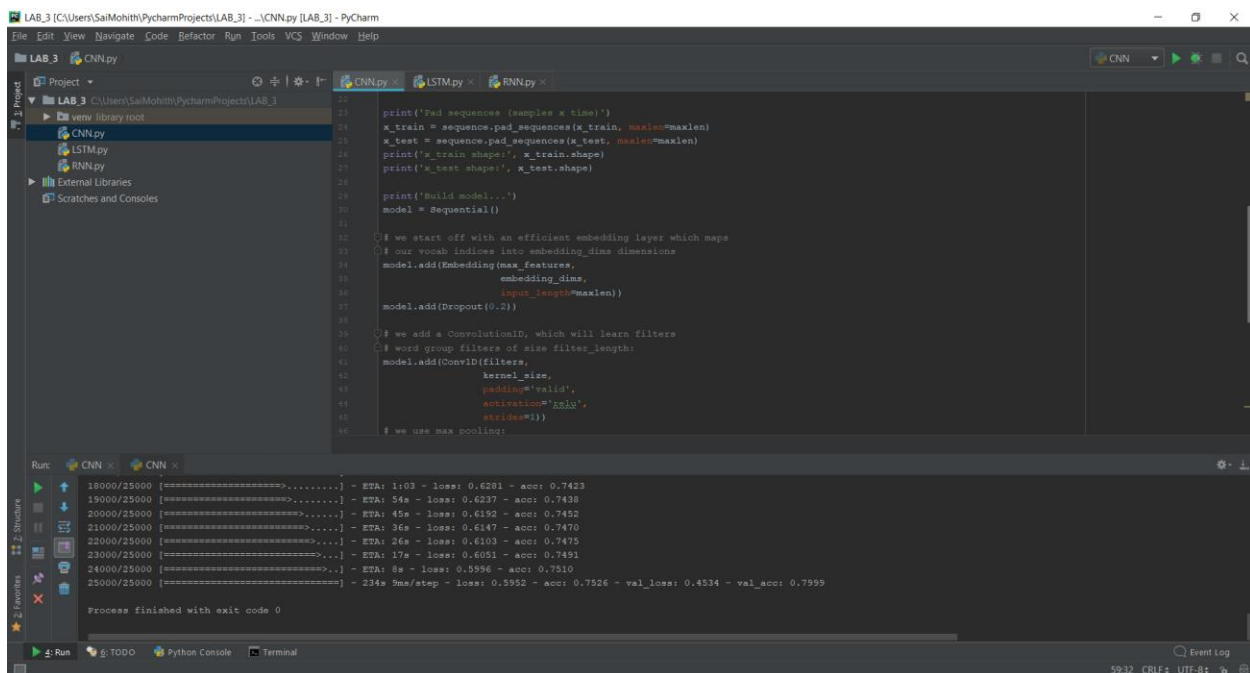
Dataset we used in this assignment is a popular movie review dataset. It has the binary sentiment labels for the data or reviews. We divide the data into Test and train data with a division rate of 50%. 25000 reviews for train and remaining as test.

Parameters:

The parameters considered while building model are max_features which limits the sentences and maxlen which limits the length of the reviews. We use the batch_size which is used to get the data for every iteration. Epochs is nothing but number of iterations to take place on building the data. We used one hidden layer which is activated using ReLU. Our output layer is activated using sigmoid. Optimizer used here is Adam which is best compared to other optimizers available. Loss and acc are parameters for Loss and Accuracy.

Evaluation & Discussion:

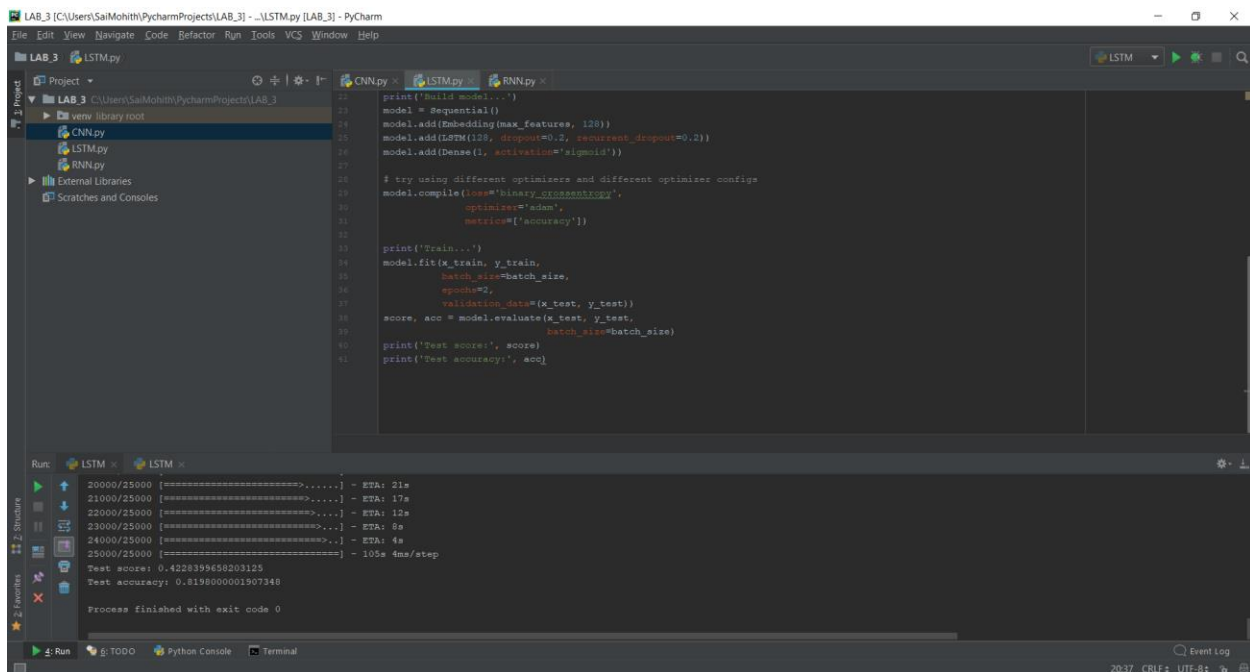
CNN:



The screenshot displays the PyCharm IDE interface. The main editor window shows the code for a CNN model. The code includes data preprocessing, model building, and training. The model architecture consists of an Embedding layer, a Dropout layer, a Conv1D layer, and a MaxPooling layer. The training results are shown in the Run console.

```
LAB_3 [C:\Users\SaiMohith\PycharmProjects\LAB_3] - ... \CNN.py [LAB_3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
LAB_3 CNN.py
Project
LAB_3 C:\Users\SaiMohith\PycharmProjects\LAB_3
venv library root
CNN.py
LSTM.py
RNN.py
External Libraries
Scratches and Consoles
Run: CNN x CNN x
18000/25000 [=====] - ETA: 1:03 - loss: 0.6281 - acc: 0.7423
19000/25000 [=====] - ETA: 54s - loss: 0.6237 - acc: 0.7438
20000/25000 [=====] - ETA: 45s - loss: 0.6192 - acc: 0.7452
21000/25000 [=====] - ETA: 36s - loss: 0.6147 - acc: 0.7470
22000/25000 [=====] - ETA: 26s - loss: 0.6103 - acc: 0.7475
23000/25000 [=====] - ETA: 17s - loss: 0.6051 - acc: 0.7491
24000/25000 [=====] - ETA: 8s - loss: 0.5996 - acc: 0.7510
25000/25000 [=====] - 234s 9ms/step - loss: 0.5952 - acc: 0.7526 - val_loss: 0.4534 - val_acc: 0.7999
Process finished with exit code 0
Run: Run Python Console Terminal
```

LSTM:



The screenshot shows the PyCharm IDE with the LSTM.py file open. The code defines an LSTM model with 128 units, a dropout rate of 0.2, and a sigmoid activation. It uses the Adam optimizer and trains for 2 epochs with a batch size of 2. The training progress is shown in the Run console, and the final test accuracy is 0.819800001907348.

```
print('Build model...')
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

# try using different optimizers and different optimizer configs
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

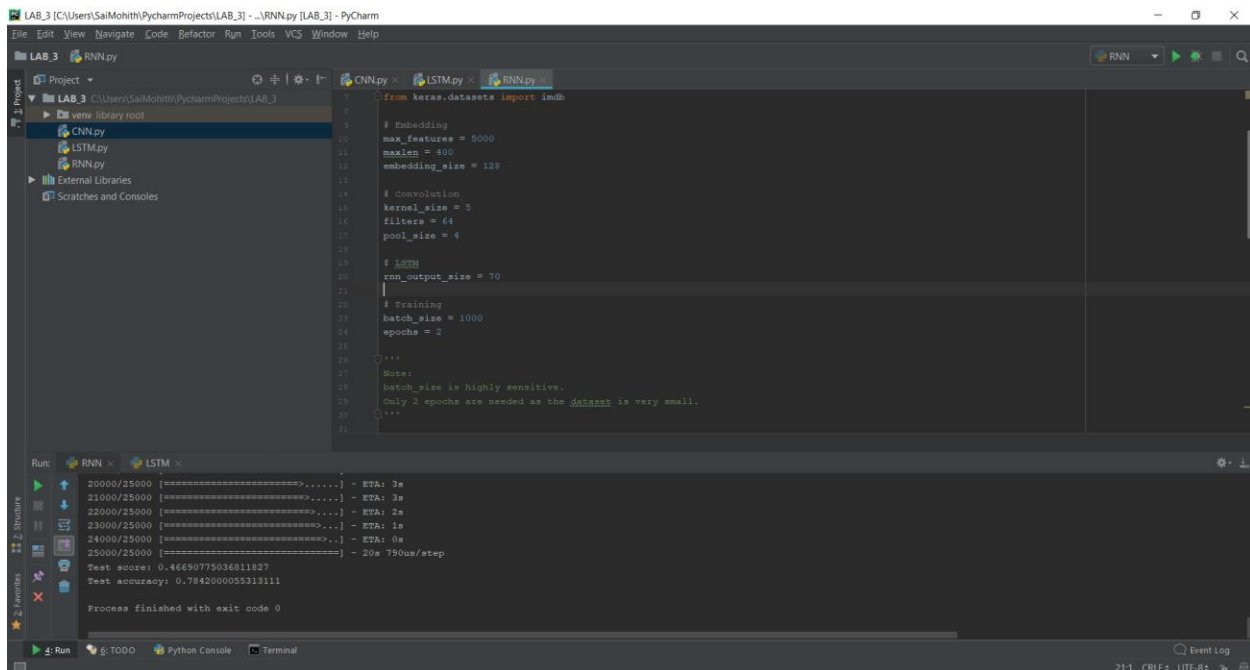
print('Train...')
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=2,
          validation_data=(x_test, y_test))
score, acc = model.evaluate(x_test, y_test,
                             batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

Run: LSTM x LSTM x

Progress	ETA
20000/25000 [=====]>.....	ETA: 21s
21000/25000 [=====]>.....	ETA: 17s
22000/25000 [=====]>.....	ETA: 12s
23000/25000 [=====]>.....	ETA: 8s
24000/25000 [=====]>.....	ETA: 4s
25000/25000 [=====]	105s 4ms/step

Test score: 0.422839658203125
Test accuracy: 0.819800001907348
Process finished with exit code 0

RNN:



The screenshot shows the PyCharm IDE with the RNN.py file open. The code defines an RNN model with 5000 max_features, 400 maxlen, and 128 embedding_size. It uses a Convolutional layer with kernel_size=5, filters=64, and pool_size=4. The model is trained for 2 epochs with a batch size of 2. The training progress is shown in the Run console, and the final test accuracy is 0.784200005313111.

```
from keras.datasets import imdb

# Embedding
max_features = 5000
maxlen = 400
embedding_size = 128

# Convolution
kernel_size = 5
filters = 64
pool_size = 4

# LSTM
rnn_output_size = 70

# Training
batch_size = 1000
epochs = 2

Notes:
batch_size is highly sensitive.
Only 2 epochs are needed as the dataset is very small.
```

Run: RNN x LSTM x

Progress	ETA
20000/25000 [=====]>.....	ETA: 3s
21000/25000 [=====]>.....	ETA: 3s
22000/25000 [=====]>.....	ETA: 2s
23000/25000 [=====]>.....	ETA: 1s
24000/25000 [=====]>.....	ETA: 0s
25000/25000 [=====]	20s 790us/step

Test score: 0.46650775036811827
Test accuracy: 0.784200005313111
Process finished with exit code 0

If we look at the above screenshots, we can see the accuracies of the models. These are obtained running with the model using the same batch_size, maxlen, max_features etc. If we

look at the accuracy obtained for three models, LSTM gave me the highest accuracy of 81.98%. CNN gave a accuracy of 79.99% which is not too bad compared to LSTM. RNN scored the accuracy of 78.42% which is mostly close to CNN but far less than LSTM. If we see LSTM has a good accuracy than other two models. But LSTM took more processing time than other two models. It's probably because of the gates in the LSTM. If we consider the time as main thing, CNN works best than the other two models.

Conclusion:

CNN will be best suit for the IMDB dataset because of good accuracy and less processing time. LSTM, on other hand, has good accuracy (not a lot different than CNN) but takes more time to build and test. RNN has a good accuracy but less than other two models because it first translates and then aggregates which makes it slower than CNN. Considering all these factors, we can choose CNN as a best model for implementing the text classification on this dataset.

References:

<https://stackoverflow.com/questions/41322243/how-to-use-keras-rnn-for-text-classification-in-a-dataset>

https://github.com/keras-team/keras/blob/master/examples/imdb_lstm.py

<http://adventuresinmachinelearning.com/keras-tutorial-cnn-11-lines/>

<https://github.com/jiegzhan/multi-class-text-classification-cnn-rnn>

<https://machinelearningmastery.com/sequence-classification-lstmrecurrent-neural-networks-python-keras/>

<https://datascience.stackexchange.com/questions/11619/rnn-vs-cnn-at-a-high-level>