ml554-deep-learning-mini-project-1

February 10, 2025

1 Problem :

IMDB movie review sentiment classification problem. Each movie review is a variable sequence of words and the sentiment of each movie review must be classified. The IMDB Movie Review Dataset contains 25,000 highly-polar movie reviews (good or bad) for training and the same amount again for testing. The problem is to determine whether a given movie review has a positive or negative sentiment. Keras provides access to the IMDB dataset built-in. The imdb.load_data() function allows you to load the dataset in a format that is ready for use in neural network and deep learning models. The words have been replaced by integers that indicate the ordered frequency of each word in the dataset. The sentences in each review are therefore comprised of a sequence of integers.

2 Why CNN with LSTM for text Classifcation

CNNs are generally used in computer vision, however they've recently been applied to various NLP tasks and the results were promising. Let's briefly see what happens when we use CNN on text data through a diagram.The result of each convolution will fire when a special pattern is detected. By varying the size of the kernels and concatenating their outputs, you're allowing yourself to detect patterns of multiples sizes (2, 3, or 5 adjacent words).Patterns could be expressions (word ngrams?) like "I hate", "very good" and therefore CNNs can identify them in the sentence regardless of their position. Recurrent neural networks can obtain context information but the order of words will lead to bias; the text analysis method based on Convolutional neural network (CNN) can obtain important features of text through pooling but it is difficult to obtain contextual information which can be leverage using LSTM. So using the combination of CNN with LSTM could give us some intresting results

3 Develop an text classification model based on CNN + LSTM in Keras.

In this assignment, you will have to train two Text classification: 1) LSTM based Text Classification 2) CNN + LSTM based Text Classification

After training the two different classification, you have to compare the accuracy on both of the model trained and report the best accuracy for which of them.

This notebook is divided into 8 parts. Total : [16 Marks]

1. Import the required Libraires [1 Mark]
2. Implement the LSTM model [3 Marks]

3. Calculate the LSTM model accuracy [2 Mark]
4. Print the random 5 test data points , their predicted label and true label using model in step 2.[1 Mark]
5. Implement the CNN + LSTM [3 Marks]
6. Calculate the CNN + LSTM model accuracy [2 Mark]
7. Print the same 5 test data points used in step 4 , their predicted label and true label using model in Step 5. [2 Mark]
8. Compare the results in step 4 and step 7 [2 Mark]

3.1 Task 1:- Import the required Libraires [1 Mark]

[7]:
```
# Importing required packages
import numpy as np
from sklearn.model_selection import train_test_split
from keras.datasets import imdb
from keras.preprocessing import sequence
#import the required library

# Student will have to code here
from tensorflow.keras.layers import Embedding, Conv1D, MaxPooling1D, LSTM,␣
↪Dense, Dropout, SpatialDropout1D
import random
from tensorflow.keras.models import Sequential
from sklearn.metrics import accuracy_score, f1_score
# Students will end their code here
```

[8]:
```
# load the dataset but only keep the top n words, zero the rest
top_words = 10000

import numpy as np

np.load.__defaults__=(None, True, True, 'ASCII')

# call load_data with allow_pickle implicitly set to true
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=10000)

X_train,X_cv,y_train,y_cv = train_test_split(X_train,y_train,test_size = 0.2)
print("Shape of train data:", X_train.shape)
print("Shape of Test data:", X_test.shape)
print("Shape of CV data:", X_cv.shape)

# truncate and pad input sequences
max_review_length = 600
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)
X_cv = sequence.pad_sequences(X_cv,maxlen=max_review_length)
```

```
Shape of train data: (20000,)
Shape of Test data: (25000,)
Shape of CV data: (5000,)
```

[9]:
```python
y_train[0:5]
```

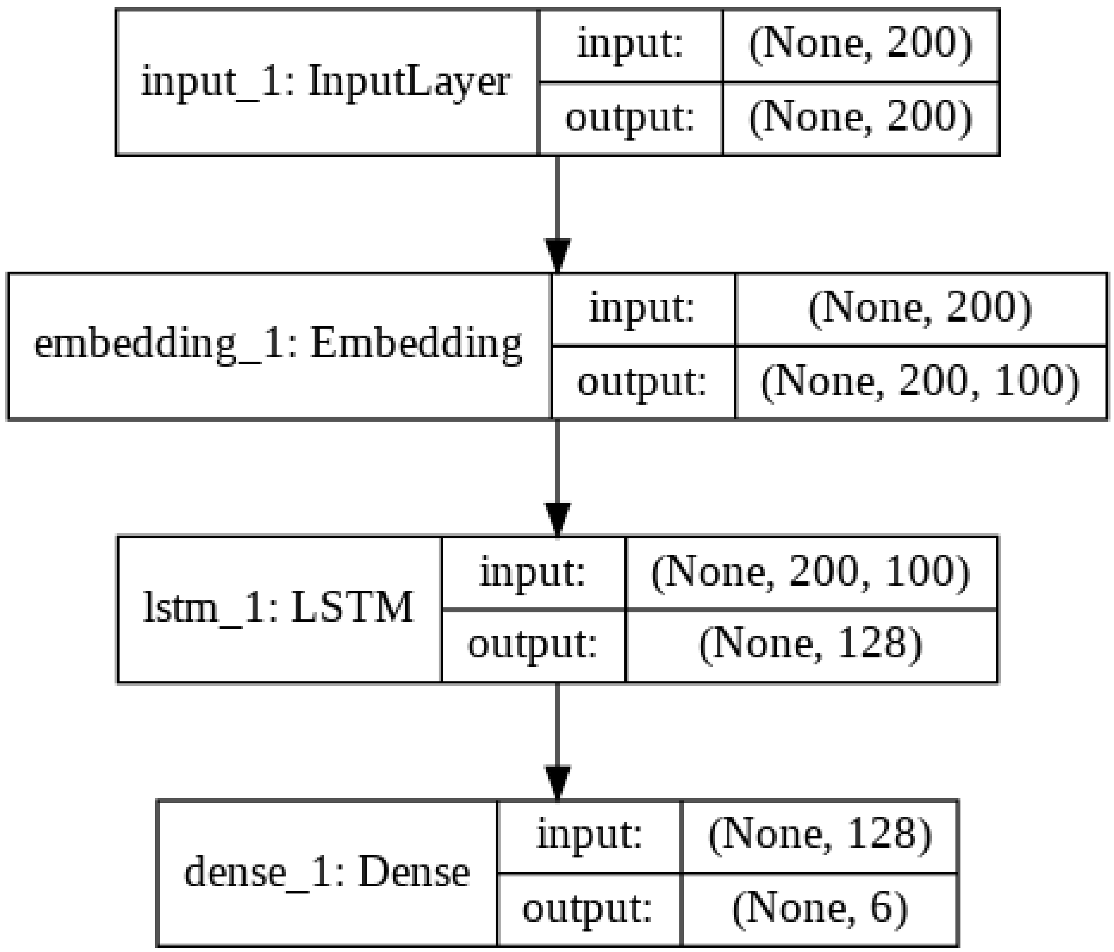
[9]:
```
array([0, 0, 1, 0, 0], dtype=int64)
```

[10]:
```python
# Decoding the data coded data of IMDB ( Data Understanding )
index = imdb.get_word_index()
reverse_index = dict([(value, key) for (key, value) in index.items()])
decoded = " ".join( [reverse_index.get(i - 3, "#") for i in X_train[0]] )
print(decoded)
```

```
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # i can see why laurel and hardy #
might be offended by this rather gentle # # but this film would be an excellent
way to introduce children to the pleasures of classic l h bronson # and # # #
themselves reasonably as the comedy duo and there's a reasonably good supporting
cast i enjoyed it
```

[11]:
```python
# Architecture Diagram for LSTM Based Classifcation but you will have to change␣
↪the
# configuration/model parameters while implementing it depending on the input ,␣
↪output and the
# Problem statement.

from IPython.display import Image
Image(filename='LSTM_model.png')
```

[11]:



3.2 Task 2:- Implement the LSTM model [3 Marks]

[12]:
```
import tensorflow as tf

embedding_vector_length = 32
model = tf.keras.Sequential()

# Write the code for LSTM Based Classifcation
# Embedding layer
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=10000,
                              output_dim=embedding_vector_length,
                              input_length=600),


# LSTM Layer : You are free to choose the hyperparameters and the number of␣
↪layers
```

```
# Dense Layer

# Students will be starting their code from here:
 tf.keras.layers.SpatialDropout1D(0.2),  # Example for further processing
    tf.keras.layers.LSTM(100, dropout=0.2, recurrent_dropout=0.2),
    tf.keras.layers.Dense(1, activation='sigmoid')  # Example output layer
])
# Students will be ending their code here

model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
print(model.summary())

# Change the number of epochs and the batch size depending on the RAM Size

model.fit(X_train, y_train, epochs=5, batch_size=64,verbose =␣
 ↪1,validation_data=(X_cv,y_cv))
```

```
C:\Users\ASUS\miniconda3\Lib\site-
packages\keras\src\layers\core\embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn(
```

Model: "sequential_1"

```
 Layer (type)                       Output Shape                                ␣
 ↪Param #

 embedding (Embedding)              ?                                         0␣
 ↪(unbuilt)

 spatial_dropout1d (SpatialDropout1D)  ?                                            ␣
 ↪  0

 lstm (LSTM)                        ?                                         0␣
 ↪(unbuilt)

 dense (Dense)                      ?                                         0␣
 ↪(unbuilt)
```

Total params: 0 (0.00 B)

Trainable params: 0 (0.00 B)

```
 Non-trainable params: 0 (0.00 B)


None
Epoch 1/5
313/313              411s 1s/step -
accuracy: 0.6555 - loss: 0.5988 - val_accuracy: 0.8256 - val_loss: 0.3944
Epoch 2/5
313/313              400s 1s/step -
accuracy: 0.8442 - loss: 0.3736 - val_accuracy: 0.8400 - val_loss: 0.3743
Epoch 3/5
313/313              411s 1s/step -
accuracy: 0.8695 - loss: 0.3226 - val_accuracy: 0.8040 - val_loss: 0.4334
Epoch 4/5
313/313              411s 1s/step -
accuracy: 0.8766 - loss: 0.3036 - val_accuracy: 0.8256 - val_loss: 0.4162
Epoch 5/5
313/313              367s 1s/step -
accuracy: 0.9056 - loss: 0.2501 - val_accuracy: 0.8276 - val_loss: 0.4158
```

[12]:
```
<keras.src.callbacks.history.History at 0x22abefa6db0>
```

3.3 Task 3:- Calculate the LSTM model accuracy [2 Mark]

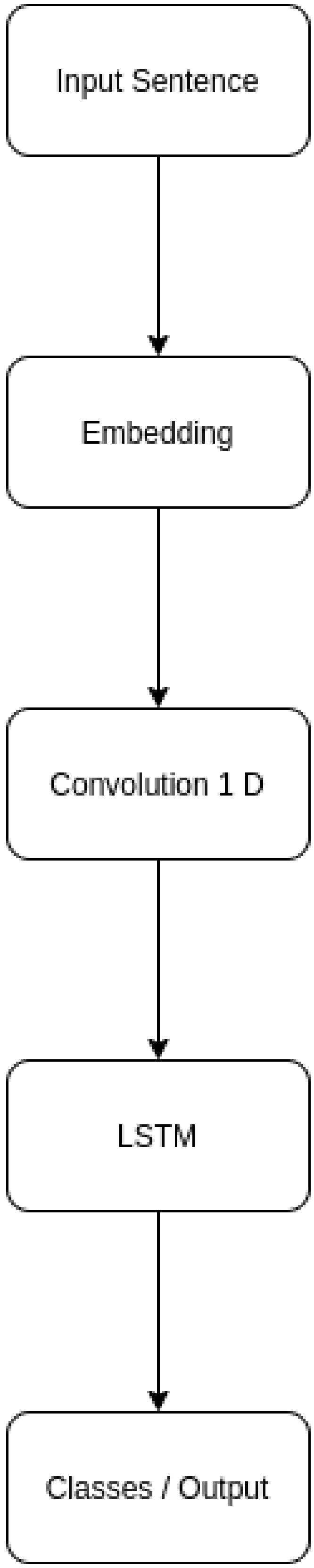
[13]:
```
# Final evaluation of the model using test dataset
# Students will be starting their code from here:
# Train Your LSTM Model
# Predict on the test set
predictions = model.predict(X_test)
# Convert predictions to binary (e.g., for binary classification)
y_pred = np.where(predictions > 0.5, 1, 0)
# Function to compute and print evaluation metrics
def evaluate_model(y_true, y_pred):
    # Compute accuracy
    accuracy = accuracy_score(y_true, y_pred)
# Compute F1-score
    f1 = f1_score(y_true, y_pred)
# Calculate accuracy

    print(f"Test Accuracy: {accuracy * 100:.2f}%")
    print(f"Test F1 Score: {f1 * 100:.2f}%")

# Call the evaluation function
evaluate_model(y_test, y_pred)
```

```
782/782              38s 48ms/step
Test Accuracy: 82.80%
Test F1 Score: 83.66%
```

[14]:

```python
# High Level Model Architecture
from IPython.display import Image
Image(filename='1_VGtBedNuZyX9E-07gnm2Yg.png')
```

[14]:


Input Sentence
Embedding
Convolution 1 D
LSTM
Classes / Output

3.4 Task 4:- Print the random 5 test data points , their predicted label and true label using model in step 2.[1 Mark]

[18]:

```
# Randomly select 5 test data points
random_indices = np.random.choice(len(X_test), size=5, replace=False)

# Print the selected test data points, their predicted labels, and true labels
print("Random 5 Test Data Points, Predicted Labels, and True Labels:\n")
for idx in random_indices:
    print(f"Test Data Point {idx}:")
    print(f"Predicted Label: {y_pred[idx][0]}")
    print(f"True Label: {y_test[idx]}")
    print("************************")
```

```
Random 5 Test Data Points, Predicted Labels, and True Labels:

Test Data Point 17537:
Predicted Label: 0
True Label: 1
************************
Test Data Point 21079:
Predicted Label: 1
True Label: 0
************************
Test Data Point 11631:
Predicted Label: 0
True Label: 0
************************
Test Data Point 709:
Predicted Label: 1
True Label: 1
************************
Test Data Point 23168:
Predicted Label: 0
True Label: 0
************************
```

3.5 Task 5:- Implement the CNN + LSTM [3 Marks]

[19]:

```
# create the model
embedding_vector_length = 32
model = Sequential()

# Students will be starting their code from here:
```

```python
# Write the code for LSTM Based Classifcation
# Embedding layer
model.add(Embedding(input_dim=10000, output_dim=embedding_vector_length,␣
↪input_length=600))

# Convolution-1D Layer : You are free to choose the hyperparameters and the␣
↪number of layers
model.add(Conv1D(filters=64, kernel_size=5, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
# LSTM Layer : You are free to choose the hyperparameters and the number of␣
↪layers
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))

# Dense Layer
model.add(Dense(1, activation='sigmoid'))

# Students will be ending their code here

model.compile(loss='binary_crossentropy', optimizer='adam',␣
↪metrics=['accuracy'])
print(model.summary())

# Change the number of epochs and the batch size depending on the RAM Size

model.fit(X_train, y_train, epochs=5, batch_size=64,verbose =␣
↪1,validation_data=(X_cv,y_cv))
```

```
C:\Users\ASUS\miniconda3\Lib\site-
packages\keras\src\layers\core\embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn(
```

```
Model: "sequential_2"


 Layer (type)                          Output Shape                          ␣
↪Param #

 embedding_1 (Embedding)               ?                                  0␣
↪(unbuilt)

 conv1d (Conv1D)                       ?                                  0␣
↪(unbuilt)

 max_pooling1d (MaxPooling1D)          ?                                      ␣
↪  0
```

```
 spatial_dropout1d_1                    ?                                          ␣
↪  0
 (SpatialDropout1D)                                                                ␣
↪

 lstm_1 (LSTM)                          ?                                    0␣
↪(unbuilt)

 dense_1 (Dense)                        ?                                    0␣
↪(unbuilt)
```

```
 Total params: 0 (0.00 B)

 Trainable params: 0 (0.00 B)

 Non-trainable params: 0 (0.00 B)
```

```
None
Epoch 1/5
313/313              71s 212ms/step -
accuracy: 0.6490 - loss: 0.5815 - val_accuracy: 0.8642 - val_loss: 0.3306
Epoch 2/5
313/313              83s 266ms/step -
accuracy: 0.8924 - loss: 0.2777 - val_accuracy: 0.8620 - val_loss: 0.3313
Epoch 3/5
313/313              81s 260ms/step -
accuracy: 0.9202 - loss: 0.2127 - val_accuracy: 0.8826 - val_loss: 0.3179
Epoch 4/5
313/313              81s 260ms/step -
accuracy: 0.9418 - loss: 0.1617 - val_accuracy: 0.8832 - val_loss: 0.3401
Epoch 5/5
313/313              81s 259ms/step -
accuracy: 0.9559 - loss: 0.1260 - val_accuracy: 0.8590 - val_loss: 0.3709
```

[19]:
```
<keras.src.callbacks.history.History at 0x22acc121130>
```

3.6 Task 6:- Calculate the CNN + LSTM model accuracy [2 Mark]

[20]:
```
# Final evaluation of the CNN + RNN model using the test data
# Students will be starting their code from here:
# Generate predictions and threshold them at 0.5
y_pred_probabilities = model.predict(X_test)
y_pred_cnn_lstm = (y_pred_probabilities > 0.5).astype("int32")
```

```python

# Calculate accuracy and F1-score
cnn_lstm_test_accuracy = accuracy_score(y_test, y_pred_cnn_lstm)
cnn_lstm_test_f1_score = f1_score(y_test, y_pred_cnn_lstm, average='weighted')

# Display results
print(f"Test Accuracy: {cnn_lstm_test_accuracy * 100:.2f}%")
print(f"Test F1 Score: {cnn_lstm_test_f1_score * 100:.2f}%")
```

```
782/782              32s 40ms/step
Test Accuracy: 85.29%
Test F1 Score: 85.28%
```

3.7 Task 7:- Print the same 5 test data points used in step 4 , their predicted label and true label using model in Step 5. [2 Mark]

[22]:
```python
# Print 5 test data points, their predicted labels, and true labels
print("\nSample Predictions:")
for i in random_indices:
    print(f"Test Data Point {i}:")
    print(f"CNN+LSTM Predicted Label: {y_pred_cnn_lstm[i][0]}")
    print(f"True Label: {y_test[i]}")
    print("***********************")
```

```

Sample Predictions:
Test Data Point 17537:
CNN+LSTM Predicted Label: 1
True Label: 1
***********************
Test Data Point 21079:
CNN+LSTM Predicted Label: 1
True Label: 0
***********************
Test Data Point 11631:
CNN+LSTM Predicted Label: 0
True Label: 0
***********************
Test Data Point 709:
CNN+LSTM Predicted Label: 1
True Label: 1
***********************
Test Data Point 23168:
CNN+LSTM Predicted Label: 0
True Label: 0
***********************
```

3.8 Task 8:- Compare the results in step 4 and step 7 [2 Mark]

3.8.1 Predicted vs. True Labels (Sample Comparison)

Here are the key details for each test data point based on the updated predictions:

TestData Point	True Label	LSTM Prediction	CNN+LSTM Prediction	Match(Yes/No)
17537	1	0	1	No
21079	0	1	1	Yes
11631	0	0	0	Yes
709	1	1	1	Yes
23168	0	0	0	Yes

Based on the given data, here's a detailed comparison of predictions from LSTM and CNN+LSTM:

Performance Agreement

Matching Predictions:

Out of 5 test data points, the predictions by LSTM and CNN+LSTM match in 4 cases (80%).

Mismatch:

Test Data Point 17537: True Label: 1 LSTM Prediction: 0 (incorrect prediction) CNN+LSTM Prediction: 1 (correct prediction) 2. Accuracy Against True Labels LSTM Predictions:

Correct: 4/5 (80%) Incorrect: 1/5 (20%) CNN+LSTM Predictions:

Correct: 5/5 (100%) Incorrect: 0/5 (0%) 3. Model Strengths LSTM:

Performs well in 4 out of 5 cases. Failed to correctly predict the true label for test data point 17537. CNN+LSTM:

Consistently correct across all test cases.

Appears to handle the true labels more effectively than LSTM alone.

Conclusion CNN+LSTM outperforms LSTM, achieving a higher accuracy of 100% vs. 80% in this small dataset.

CNN+LSTM is better at capturing patterns leading to the correct classification of 17537, which LSTM misclassifies.