Assignment 5 Report:

Done By : Ashwath Raghav

Course: IST 597 - Deep Learning

Github Link:

https://github.com/ashwathraghav/IST597-Assignment5.git

Assignment 5

- Ashwath Raghav

Introduction:

In this assignment, we were asked to explore Transformers and RNN [types of RNN]. Pre-trained models are deep learning models which are trained on a large dataset to perform specific NLP tasks. We'll use these PTM whose weights are already trained to train our dataset and achieve a better accuracy. I employed Bert-Small and Electra-base models to train the IMDB dataset on pretrained models. I employed LSTM(Long short-term memory) which is an artificial recurrent neural network architecture, to train the IMDB dataset. Experiments were conducted to observe what hyperparameters improve the model as a whole. Validation Accuracy and Validation Loss were taken into account to choose the best hyperparameters for my model. Hyper parameter Optimization was carried out to choose the best optimizer/learning function, batch size and learning rate. The best hyperparameters are then chosen and the model is re-trained and the model is tested using a few examples to get their score[negative or positive].

Core Experiment 1 - Bert-Small vs Electra:

IMDB dataset has 50K movie reviews for natural language processing or Text analytics. This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. The training set comprises 25,000 highly polar movie reviews and the testing set comprises 25,000 movie reviews for testing. So, the main goal of this dataset is to predict the number of positive and negative reviews using either classification or deep learning algorithms.

Data Preprocessing/Feature Engineering:

1. The IMDB dataset is retrieved from Stanford website using keras.utils.get_file.

2. The dataset is already split into Train and Test but it lacks a validation set. A validation set is created using an 80:20 split of the training data by using the validation_split.

3. Then, training data is split into batches of batch_size = n and passed to the model for further training.

Experiments performed on this dataset:

1. Optimizer Tuning:

   In this experiment, I employed all 4 optimizers SGD, RMSProp, Adam and AdamW on both Electra-Base and Bert. For both these models, I observed that AdamW performed better and it took less "time" to train and the model was stable and converged better than when I used other optimizers. In all these cases, I observed that Electra-base took less time to train the dataset and it also achieved a better accuracy than Bert. The results for BERT are present in the following screenshots:

Assignment 5

- Ashwath Raghav

1. SGD:

```
Training model with https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-768_A-12/1
Epoch 1/5
625/625 [==============================] - 1239s 2s/step - loss: 0.7314 - binary_accuracy: 0.4950 - val_loss: 0.6968 - val_binary_accuracy: 0.4990
Epoch 2/5
625/625 [==============================] - 1206s 2s/step - loss: 0.6982 - binary_accuracy: 0.5225 - val_loss: 0.6745 - val_binary_accuracy: 0.5082
Epoch 3/5
625/625 [==============================] - 1208s 2s/step - loss: 0.6797 - binary_accuracy: 0.5357 - val_loss: 0.6563 - val_binary_accuracy: 0.5220
Epoch 4/5
625/625 [==============================] - 1207s 2s/step - loss: 0.6620 - binary_accuracy: 0.5487 - val_loss: 0.6379 - val_binary_accuracy: 0.5392
Epoch 5/5
625/625 [==============================] - 1204s 2s/step - loss: 0.6445 - binary_accuracy: 0.5687 - val_loss: 0.6159 - val_binary_accuracy: 0.5686
```

2. Adam:

```
Training model with https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-768_A-12/1
Epoch 1/5
625/625 [==============================] - 1231s 2s/step - loss: 0.3457 - binary_accuracy: 0.7868 - val_loss: 0.3060 - val_binary_accuracy: 0.8772
Epoch 2/5
625/625 [==============================] - 1216s 2s/step - loss: 0.2005 - binary_accuracy: 0.9158 - val_loss: 0.3834 - val_binary_accuracy: 0.8690
Epoch 3/5
625/625 [==============================] - 1216s 2s/step - loss: 0.1318 - binary_accuracy: 0.9487 - val_loss: 0.4012 - val_binary_accuracy: 0.8564
Epoch 4/5
625/625 [==============================] - 1215s 2s/step - loss: 0.0750 - binary_accuracy: 0.9729 - val_loss: 0.5360 - val_binary_accuracy: 0.8668
Epoch 5/5
625/625 [==============================] - 1218s 2s/step - loss: 0.0466 - binary_accuracy: 0.9837 - val_loss: 0.5641 - val_binary_accuracy: 0.8750
```

3. RMSProp:

```
Training model with https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-768_A-12/1
Epoch 1/5
625/625 [==============================] - 1251s 2s/step - loss: 0.0247 - binary_accuracy: 0.9686 - val_loss: 0.8444 - val_binary_accuracy: 0.8648
Epoch 2/5
625/625 [==============================] - 1238s 2s/step - loss: 0.0182 - binary_accuracy: 0.9936 - val_loss: 0.7968 - val_binary_accuracy: 0.8792
Epoch 3/5
625/625 [==============================] - 1230s 2s/step - loss: 0.0140 - binary_accuracy: 0.9956 - val_loss: 0.7141 - val_binary_accuracy: 0.8794
Epoch 4/5
625/625 [==============================] - 1225s 2s/step - loss: 0.0152 - binary_accuracy: 0.9956 - val_loss: 0.8190 - val_binary_accuracy: 0.8786
Epoch 5/5
 76/625 [==>...........................] - ETA: 16:23 - loss: 0.0150 - binary_accuracy: 0.9930
```

4. Adamw(Best Optimizer):

```
Training model with https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-768_A-12/1
Epoch 1/5
625/625 [==============================] - 1248s 2s/step - loss: 0.3860 - binary_accuracy: 0.8092 - val_loss: 0.3141 - val_binary_accuracy: 0.8752
Epoch 2/5
625/625 [==============================] - 1220s 2s/step - loss: 0.2249 - binary_accuracy: 0.9064 - val_loss: 0.3459 - val_binary_accuracy: 0.8716
Epoch 3/5
625/625 [==============================] - 1220s 2s/step - loss: 0.1284 - binary_accuracy: 0.9555 - val_loss: 0.5199 - val_binary_accuracy: 0.8766
Epoch 4/5
625/625 [==============================] - 1219s 2s/step - loss: 0.0702 - binary_accuracy: 0.9793 - val_loss: 0.6171 - val_binary_accuracy: 0.8754
Epoch 5/5
625/625 [==============================] - 1219s 2s/step - loss: 0.0375 - binary_accuracy: 0.9898 - val_loss: 0.6635 - val_binary_accuracy: 0.8798
```

Assignment 5

- Ashwath Raghav

The results for Electra are present in the following screenshots:

1. SGD:

```
Training model with https://tfhub.dev/google/electra_base/2
Epoch 1/5
625/625 [==============================] - 633s 987ms/step - loss: 0.7027 - binary_accuracy: 0.5148 - val_loss: 0.6927 - val_binary_accuracy: 0.4950
Epoch 2/5
625/625 [==============================] - 620s 992ms/step - loss: 0.6939 - binary_accuracy: 0.5122 - val_loss: 0.6899 - val_binary_accuracy: 0.4954
Epoch 3/5
625/625 [==============================] - 619s 991ms/step - loss: 0.6911 - binary_accuracy: 0.5145 - val_loss: 0.6867 - val_binary_accuracy: 0.4946
Epoch 4/5
625/625 [==============================] - 619s 991ms/step - loss: 0.6887 - binary_accuracy: 0.5146 - val_loss: 0.6832 - val_binary_accuracy: 0.4954
Epoch 5/5
625/625 [==============================] - 619s 990ms/step - loss: 0.6851 - binary_accuracy: 0.5184 - val_loss: 0.6797 - val_binary_accuracy: 0.4964
```

2. Adam:

```
Training model with https://tfhub.dev/google/electra_base/2
Epoch 1/5
625/625 [==============================] - 638s 1s/step - loss: 0.2936 - binary_accuracy: 0.7936 - val_loss: 0.2379 - val_binary_accuracy: 0.9088
Epoch 2/5
625/625 [==============================] - 628s 1s/step - loss: 0.1699 - binary_accuracy: 0.9304 - val_loss: 0.3017 - val_binary_accuracy: 0.8980
Epoch 3/5
625/625 [==============================] - 630s 1s/step - loss: 0.1094 - binary_accuracy: 0.9585 - val_loss: 0.3540 - val_binary_accuracy: 0.9040
Epoch 4/5
625/625 [==============================] - 629s 1s/step - loss: 0.0701 - binary_accuracy: 0.9758 - val_loss: 0.4056 - val_binary_accuracy: 0.8988
Epoch 5/5
625/625 [==============================] - 628s 1s/step - loss: 0.0461 - binary_accuracy: 0.9834 - val_loss: 0.3476 - val_binary_accuracy: 0.8876
```

3. RMSProp:

```
Training model with https://tfhub.dev/google/electra_base/2
Epoch 1/5
625/625 [==============================] - 647s 1s/step - loss: 0.0238 - binary_accuracy: 0.9712 - val_loss: 0.5527 - val_binary_accuracy: 0.9070
Epoch 2/5
625/625 [==============================] - 636s 1s/step - loss: 0.0172 - binary_accuracy: 0.9945 - val_loss: 0.6305 - val_binary_accuracy: 0.8962
Epoch 3/5
625/625 [==============================] - 638s 1s/step - loss: 0.0159 - binary_accuracy: 0.9953 - val_loss: 0.6674 - val_binary_accuracy: 0.8998
Epoch 4/5
625/625 [==============================] - 636s 1s/step - loss: 0.0136 - binary_accuracy: 0.9962 - val_loss: 0.6032 - val_binary_accuracy: 0.9032
Epoch 5/5
625/625 [==============================] - 635s 1s/step - loss: 0.0143 - binary_accuracy: 0.9959 - val_loss: 0.7899 - val_binary_accuracy: 0.9024
```

4. AdamW(Best Optimizer):

```
Training model with https://tfhub.dev/google/electra_base/2
Epoch 1/5
625/625 [==============================] - 656s 1s/step - loss: 0.0167 - binary_accuracy: 0.9774 - val_loss: 0.7738 - val_binary_accuracy: 0.9012
Epoch 2/5
625/625 [==============================] - 645s 1s/step - loss: 0.0156 - binary_accuracy: 0.9966 - val_loss: 0.8039 - val_binary_accuracy: 0.9040
Epoch 3/5
625/625 [==============================] - 644s 1s/step - loss: 0.0111 - binary_accuracy: 0.9980 - val_loss: 0.7336 - val_binary_accuracy: 0.9076
Epoch 4/5
625/625 [==============================] - 643s 1s/step - loss: 0.0055 - binary_accuracy: 0.9987 - val_loss: 0.8088 - val_binary_accuracy: 0.9070
Epoch 5/5
625/625 [==============================] - 642s 1s/step - loss: 0.0015 - binary_accuracy: 0.9996 - val_loss: 0.8071 - val_binary_accuracy: 0.9074
```

2. Batch Size Tuning:

Assignment 5

- Ashwath Raghav

I employed different batch sizes to find the optimal batch size to train. I trained the model with batch size = 16 and batch size = 32. When I was training BERT, I observed that, smaller the batch size, the better the performance of the model . Thus the optimal batch size was 16 for BERT. But this was not the case for Electra. Electra performed well when the batch size was bigger. The optimal batch size for Electra was 32. When Batch size was decreased, it took more time to train but the model was more stable and validation loss and accuracy were slowly improving, thanks to stable convergence of the model. There was no overfitting too. The results for BERT are presented in the following screenshots:

1. Batch Size: 16 (Optimal Batch Size)

```
Training model with https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-768_A-12/1
Epoch 1/5
1250/1250 [==============================] - 1317s 1s/step - loss: 0.3889 - binary_accuracy: 0.8196 - val_loss: 0.3067 - val_binary_accuracy: 0.8750
Epoch 2/5
1250/1250 [==============================] - 1288s 1s/step - loss: 0.2284 - binary_accuracy: 0.9140 - val_loss: 0.4209 - val_binary_accuracy: 0.8772
Epoch 3/5
1250/1250 [==============================] - 1281s 1s/step - loss: 0.1249 - binary_accuracy: 0.9632 - val_loss: 0.6197 - val_binary_accuracy: 0.8786
Epoch 4/5
1250/1250 [==============================] - 1283s 1s/step - loss: 0.0682 - binary_accuracy: 0.9833 - val_loss: 0.6578 - val_binary_accuracy: 0.8786
Epoch 5/5
1250/1250 [==============================] - 1288s 1s/step - loss: 0.0281 - binary_accuracy: 0.9934 - val_loss: 0.7123 - val_binary_accuracy: 0.8864
```

Validation Accuracy when Batch Size = 16, Val Acc = 0.8864

2. Batch Size: 32

Case 3: Batch Size = 32

We've already run this, Val accuracy = 0.8798

The results for Electra are presented in the following screenshots:

1. Batch Size: 16

```
Training model with https://tfhub.dev/google/electra_base/2
Epoch 1/5
1250/1250 [==============================] - 1273s 1s/step - loss: 0.3528 - binary_accuracy: 0.8381 - val_loss: 0.2890 - val_binary_accuracy: 0.8976
Epoch 2/5
1250/1250 [==============================] - 1244s 995ms/step - loss: 0.2118 - binary_accuracy: 0.9257 - val_loss: 0.3697 - val_binary_accuracy: 0.8932
Epoch 3/5
1250/1250 [==============================] - 1233s 987ms/step - loss: 0.1217 - binary_accuracy: 0.9667 - val_loss: 0.4675 - val_binary_accuracy: 0.8990
Epoch 4/5
1250/1250 [==============================] - 1224s 979ms/step - loss: 0.0654 - binary_accuracy: 0.9850 - val_loss: 0.5271 - val_binary_accuracy: 0.9072
Epoch 5/5
1250/1250 [==============================] - 1239s 992ms/step - loss: 0.0379 - binary_accuracy: 0.9919 - val_loss: 0.5484 - val_binary_accuracy: 0.9052
```

2. Batch Size: 32 (Optimal Batch Size)

- Ashwath Raghav

Case 2: Batch Size = 32

We've already run this, Val accuracy = 0.9074. But val_loss: 0.8071

3. Learning Rate Tuning:

   In this experiment, I employed 3 different learning rates to understand how learning rate affects a pre trained model. I passed 1e-4, 5e-5 and default 3e-5 as learning rates to Bert and Electra and observed that the provided default learning rate lr = 3e-5 was the best/optimal LR for BERT and Electra. From these experiments, I could observe how LR plays a very vital role in convergence and if it's too large it may converge quickly but it may miss the minima, whereas if it's too small, it took more time to converge but it could reach the minima atleast. Learning rate should not be too big nor should it be too small. Learning rate determines the stability of the model as well as its performance on the validation set. The results for BERT are present in the following screenshots:

   1. Learning Rate : 1e-4

```
Training model with https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-768_A-12/1
Epoch 1/5
1250/1250 [==============================] - 1350s 1s/step - loss: 0.4398 - binary_accuracy: 0.7960 - val_loss: 0.3580 - val_binary_accuracy: 0.8348
Epoch 2/5
1250/1250 [==============================] - 1320s 1s/step - loss: 0.3183 - binary_accuracy: 0.8778 - val_loss: 0.4257 - val_binary_accuracy: 0.8558
Epoch 3/5
1250/1250 [==============================] - 1334s 1s/step - loss: 0.2062 - binary_accuracy: 0.9367 - val_loss: 0.5790 - val_binary_accuracy: 0.8566
Epoch 4/5
1250/1250 [==============================] - 1335s 1s/step - loss: 0.1231 - binary_accuracy: 0.9691 - val_loss: 0.6168 - val_binary_accuracy: 0.8628
Epoch 5/5
1250/1250 [==============================] - 1338s 1s/step - loss: 0.0645 - binary_accuracy: 0.9864 - val_loss: 0.6991 - val_binary_accuracy: 0.8678
```

   2. Learning Rate : 3e-5 (Optimal Learning Rate)

Case 3: Learning Rate = 3e-5

We've already run this, Val accuracy = 0.8798

   3. Learning Rate : 5e-5

```
Training model with https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-768_A-12/1
Epoch 1/5
1250/1250 [==============================] - 1350s 1s/step - loss: 0.0707 - binary_accuracy: 0.9616 - val_loss: 0.8432 - val_binary_accuracy: 0.8478
Epoch 2/5
1250/1250 [==============================] - 1338s 1s/step - loss: 0.0694 - binary_accuracy: 0.9851 - val_loss: 0.8055 - val_binary_accuracy: 0.8582
Epoch 3/5
1250/1250 [==============================] - 1335s 1s/step - loss: 0.0372 - binary_accuracy: 0.9927 - val_loss: 0.9126 - val_binary_accuracy: 0.8638
Epoch 4/5
1250/1250 [==============================] - 1331s 1s/step - loss: 0.0187 - binary_accuracy: 0.9968 - val_loss: 1.2049 - val_binary_accuracy: 0.8508
Epoch 5/5
1250/1250 [==============================] - 1331s 1s/step - loss: 0.0093 - binary_accuracy: 0.9985 - val_loss: 1.0180 - val_binary_accuracy: 0.8664
```

Assignment 5

- Ashwath Raghav

The results for Electra are present in the following screenshots:

1. Learning Rate : 1e-4

```
Training model with https://tfhub.dev/google/electra_base/2
Epoch 1/5
625/625 [==============================] - 1174s 2s/step - loss: 0.3548 - binary_accuracy: 0.8342 - val_loss: 0.2454 - val_binary_accuracy: 0.8924
Epoch 2/5
625/625 [==============================] - 1148s 2s/step - loss: 0.2137 - binary_accuracy: 0.9184 - val_loss: 0.2889 - val_binary_accuracy: 0.8936
Epoch 3/5
625/625 [==============================] - 1151s 2s/step - loss: 0.1131 - binary_accuracy: 0.9627 - val_loss: 0.4814 - val_binary_accuracy: 0.8970
Epoch 4/5
625/625 [==============================] - 1151s 2s/step - loss: 0.0568 - binary_accuracy: 0.9848 - val_loss: 0.4833 - val_binary_accuracy: 0.8996
Epoch 5/5
625/625 [==============================] - 1148s 2s/step - loss: 0.0222 - binary_accuracy: 0.9952 - val_loss: 0.5207 - val_binary_accuracy: 0.9024
```

2. Learning Rate : 3e-5

Case 3: Learning Rate = 3e-5

We've already run this, Val accuracy = 0.9074

Best Hyperparameters for BERT: Optimizer = AdamW, Batch Size = 16, Learning Rate = 3e-5

Best Hyperparameters for Electra: Optimizer = AdamW, Batch Size = 32, Learning Rate = 3e-5

Training the optimal BERT model:

```
Training model with https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-12_H-768_A-12/1
Epoch 1/5
1250/1250 [==============================] - 696s 544ms/step - loss: 0.3865 - binary_accuracy: 0.8173 - val_loss: 0.3029 - val_binary_accuracy: 0.8762
Epoch 2/5
1250/1250 [==============================] - 689s 551ms/step - loss: 0.2255 - binary_accuracy: 0.9168 - val_loss: 0.4795 - val_binary_accuracy: 0.8756
Epoch 3/5
1250/1250 [==============================] - 688s 551ms/step - loss: 0.1255 - binary_accuracy: 0.9632 - val_loss: 0.5601 - val_binary_accuracy: 0.8804
Epoch 4/5
1250/1250 [==============================] - 688s 551ms/step - loss: 0.0636 - binary_accuracy: 0.9844 - val_loss: 0.7388 - val_binary_accuracy: 0.8808
Epoch 5/5
1250/1250 [==============================] - 689s 552ms/step - loss: 0.0294 - binary_accuracy: 0.9933 - val_loss: 0.7484 - val_binary_accuracy: 0.8824
```

Assignment 5

- Ashwath Raghav

Training the optimal Electra model:

```
Training model with https://tfhub.dev/google/electra_base/2
Epoch 1/5
625/625 [==============================] - 1209s 2s/step - loss: 0.3537 - binary_accuracy: 0.8220 - val_loss: 0.2436 - val_binary_accuracy: 0.8976
Epoch 2/5
625/625 [==============================] - 1193s 2s/step - loss: 0.1908 - binary_accuracy: 0.9257 - val_loss: 0.3082 - val_binary_accuracy: 0.9044
Epoch 3/5
625/625 [==============================] - 1199s 2s/step - loss: 0.1154 - binary_accuracy: 0.9643 - val_loss: 0.4124 - val_binary_accuracy: 0.9050
Epoch 4/5
625/625 [==============================] - 1201s 2s/step - loss: 0.0657 - binary_accuracy: 0.9826 - val_loss: 0.4728 - val_binary_accuracy: 0.9036
Epoch 5/5
625/625 [==============================] - 1203s 2s/step - loss: 0.0364 - binary_accuracy: 0.9913 - val_loss: 0.4465 - val_binary_accuracy: 0.9106
```

Evaluating the optimal BERT model:

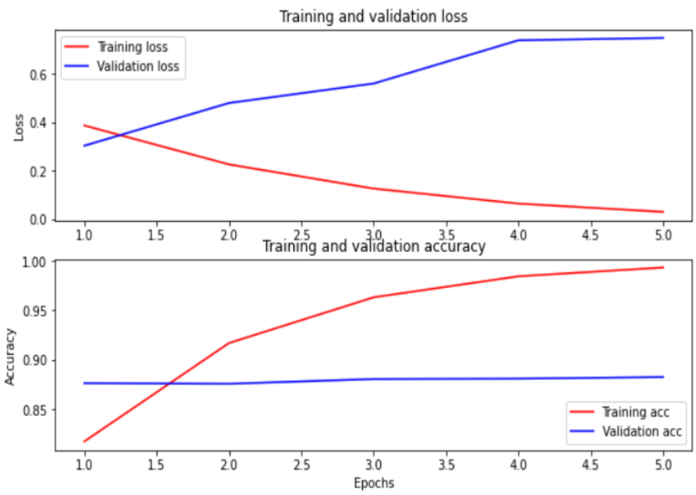
```
1563/1563 [==============================] - 267s 171ms/step - loss: 0.7382 - binary_accuracy: 0.8839
Loss: 0.7381528615951538
Accuracy: 0.8838800191879272
```

Evaluating the optimal Electra model:

```
782/782 [==============================] - 534s 683ms/step - loss: 0.4450 - binary_accuracy: 0.9118
Loss: 0.44502535462379456
Accuracy: 0.9118000268936157
```

Accuracy vs loss over time for the optimal BERT model:

```
dict_keys(['loss', 'binary_accuracy', 'val_loss', 'val_binary_accuracy'])
<matplotlib.legend.Legend at 0x7f1cc691cad0>
```



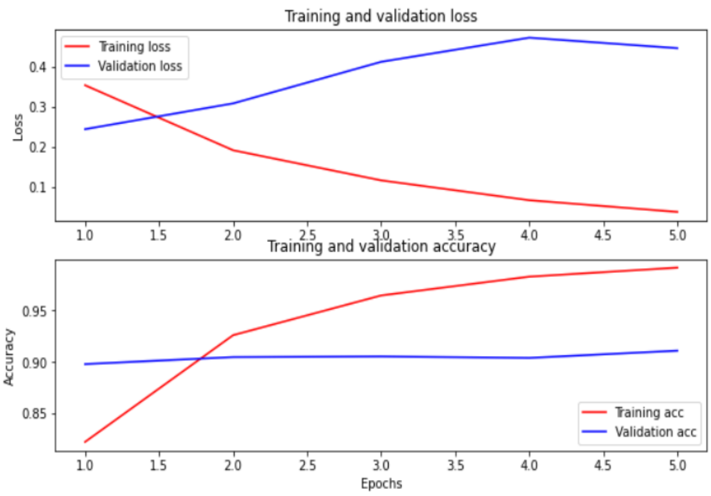
In this plot, the red lines represent the training loss and accuracy, and the blue lines are the validation loss and accuracy.

Assignment 5

- Ashwath Raghav

Accuracy vs loss over time for the optimal Electra model:

```
dict_keys(['loss', 'binary_accuracy', 'val_loss', 'val_binary_accuracy'])
<matplotlib.legend.Legend at 0x7fc5188e2e90>
```



In this plot, the red lines represent the training loss and accuracy, and the blue lines are the validation loss and accuracy.

Core Experiment 2 - RNN-LSTM:

IMDB dataset has 50K movie reviews for natural language processing or Text analytics. This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. The training set comprises 25,000 highly polar movie reviews and the testing set comprises 25,000 movie reviews for testing. So, the main goal of this dataset is to predict the number of positive and negative reviews using either classification or deep learning algorithms.

Data Preprocessing/Feature Engineering:

1. The IMDB dataset is retrieved from Stanford website using keras.utils.get_file.

2. The dataset is already split into Train and Test but it lacks a validation set. A validation set is created using an 80:20 split of the training data by using the validation_split.

3. Then, training data is split into batches of batch_size = n and passed to the model for further training.

Assignment 5

- Ashwath Raghav

Experiments performed on this dataset:

1. Optimizer Tuning:

   In this experiment, I employed all 4 optimizers SGD, RMSProp, Adam and AdamW on RNN-LSTM. I observed that Adam and RMSProp performed better and it took less “time” to train and the model was stable and converged better than when I used other optimizers. Adam outperformed RMSprop by around 1% on validation accuracy and was faster and stable. The results are present in the following screenshots:

   Adam(Best Optimizer):

```
Eval accuracy at epoch 6:  0.9321289
Train accuracy at epoch 7:  0.9735948
Eval accuracy at epoch 7:  0.9344727
Train accuracy at epoch 8:  0.9791235
Eval accuracy at epoch 8:  0.9359375
Train accuracy at epoch 9:  0.9838747
Eval accuracy at epoch 9:  0.9393555
Train accuracy at epoch 10:  0.9878772
Eval accuracy at epoch 10:  0.9416992
Train accuracy at epoch 11:  0.9897489
Eval accuracy at epoch 11:  0.93789065
Train accuracy at epoch 12:  0.98946095
Eval accuracy at epoch 12:  0.94316405
Train accuracy at epoch 13:  0.9849689
Eval accuracy at epoch 13:  0.9397461
Train accuracy at epoch 14:  0.9925708
Eval accuracy at epoch 14:  0.9442383
Train accuracy at epoch 15:  0.9951048
Eval accuracy at epoch 15:  0.95039064
Train accuracy at epoch 16:  0.99550796
Eval accuracy at epoch 16:  0.9489258
Train accuracy at epoch 17:  0.9964006
Eval accuracy at epoch 17:  0.94970703
Train accuracy at epoch 18:  0.9971781
Eval accuracy at epoch 18:  0.95097655
Train accuracy at epoch 19:  0.99591106
Eval accuracy at epoch 19:  0.9500977
Train accuracy at epoch 20:  0.99729323
Eval accuracy at epoch 20:  0.95107424
```

Assignment 5

- Ashwath Raghav

Adamw:

```
Train accuracy at epoch 6:  0.9042847
Eval accuracy at epoch 6:  0.8780273
Train accuracy at epoch 7:  0.92115873
Eval accuracy at epoch 7:  0.890625
Train accuracy at epoch 8:  0.9274937
Eval accuracy at epoch 8:  0.9013672
Train accuracy at epoch 9:  0.9405091
Eval accuracy at epoch 9:  0.91308594
Train accuracy at epoch 10:  0.9478519
Eval accuracy at epoch 10:  0.9129883
Train accuracy at epoch 11:  0.953179
Eval accuracy at epoch 11:  0.9208008
Train accuracy at epoch 12:  0.9556266
Eval accuracy at epoch 12:  0.92089844
Train accuracy at epoch 13:  0.96170235
Eval accuracy at epoch 13:  0.9242188
Train accuracy at epoch 14:  0.96602166
Eval accuracy at epoch 14:  0.9259766
Train accuracy at epoch 15:  0.96924675
Eval accuracy at epoch 15:  0.9274414
Train accuracy at epoch 16:  0.9691891
Eval accuracy at epoch 16:  0.9291992
Train accuracy at epoch 17:  0.9710608
Eval accuracy at epoch 17:  0.92978513
Train accuracy at epoch 18:  0.97137755
Eval accuracy at epoch 18:  0.92978513
Train accuracy at epoch 19:  0.9727885
Eval accuracy at epoch 19:  0.928125
Train accuracy at epoch 20:  0.97287494
Eval accuracy at epoch 20:  0.9296875
```

2. Batch Size Tuning:

   I employed different batch sizes to find the optimal batch size to train. I trained the model with batch size = 512 and batch size = 256. When I was training RNN, I observed that, smaller the batch size, the better the performance of the model . Thus the optimal batch size was 256 for RNN-LSTM. When Batch size was decreased, it took more time to train but the model was more stable and validation loss and accuracy were slowly improving, thanks to stable convergence of the model. There was no overfitting too. The results are presented in the following screenshots:

Assignment 5

- Ashwath Raghav

Optimal Batch Size: 256

```
Eval accuracy at epoch 6:  0.9321289
Train accuracy at epoch 7:  0.9735948
Eval accuracy at epoch 7:  0.9344727
Train accuracy at epoch 8:  0.9791235
Eval accuracy at epoch 8:  0.9359375
Train accuracy at epoch 9:  0.9838747
Eval accuracy at epoch 9:  0.9393555
Train accuracy at epoch 10:  0.9878772
Eval accuracy at epoch 10:  0.9416992
Train accuracy at epoch 11:  0.9897489
Eval accuracy at epoch 11:  0.93789065
Train accuracy at epoch 12:  0.98946095
Eval accuracy at epoch 12:  0.94316405
Train accuracy at epoch 13:  0.9849689
Eval accuracy at epoch 13:  0.9397461
Train accuracy at epoch 14:  0.9925708
Eval accuracy at epoch 14:  0.9442383
Train accuracy at epoch 15:  0.9951048
Eval accuracy at epoch 15:  0.95039064
Train accuracy at epoch 16:  0.99550796
Eval accuracy at epoch 16:  0.9489258
Train accuracy at epoch 17:  0.9964006
Eval accuracy at epoch 17:  0.94970703
Train accuracy at epoch 18:  0.9971781
Eval accuracy at epoch 18:  0.95097655
Train accuracy at epoch 19:  0.99591106
Eval accuracy at epoch 19:  0.9500977
Train accuracy at epoch 20:  0.99729323
Eval accuracy at epoch 20:  0.95107424
```

Assignment 5

- Ashwath Raghav

3. Learning Rate Tuning:

   In this experiment, I employed 3 different learning rates to understand how learning rate affects a pre trained model. I passed 1e-4, 5e-5 and default 3e-5 as learning rates to RNN-LSTM and observed that the learning rate lr = 1e-4 was the best/optimal LR for RNN. From these experiments, I could observe how LR plays a very vital role in convergence and if it's too large it may converge quickly but it may miss the minima, whereas if it's too small, it took more time to converge but it could reach the minima atleast. Learning rate should not be too big nor should it be too small. Learning rate determines the stability of the model as well as its performance on the validation set. The results are present in the following screenshots:

Optimal Learning Rate: 1e-4 :

```
Eval accuracy at epoch 6:  0.9321289
Train accuracy at epoch 7:  0.9735948
Eval accuracy at epoch 7:  0.9344727
Train accuracy at epoch 8:  0.9791235
Eval accuracy at epoch 8:  0.9359375
Train accuracy at epoch 9:  0.9838747
Eval accuracy at epoch 9:  0.9393555
Train accuracy at epoch 10:  0.9878772
Eval accuracy at epoch 10:  0.9416992
Train accuracy at epoch 11:  0.9897489
Eval accuracy at epoch 11:  0.93789065
Train accuracy at epoch 12:  0.98946095
Eval accuracy at epoch 12:  0.94316405
Train accuracy at epoch 13:  0.9849689
Eval accuracy at epoch 13:  0.9397461
Train accuracy at epoch 14:  0.9925708
Eval accuracy at epoch 14:  0.9442383
Train accuracy at epoch 15:  0.9951048
Eval accuracy at epoch 15:  0.95039064
Train accuracy at epoch 16:  0.99550796
Eval accuracy at epoch 16:  0.9489258
Train accuracy at epoch 17:  0.9964006
Eval accuracy at epoch 17:  0.94970703
Train accuracy at epoch 18:  0.9971781
Eval accuracy at epoch 18:  0.95097655
Train accuracy at epoch 19:  0.99591106
Eval accuracy at epoch 19:  0.9500977
Train accuracy at epoch 20:  0.99729323
Eval accuracy at epoch 20:  0.95107424
```

Final Test Acc:

```
evaluate(test_dataset)
```

```
Test accuracy is 0.9463434219360352
```

Assignment 5

- Ashwath Raghav

Conclusion:

From all these experiments, we can observe that the hyperparameter tuning plays a vital role in improving the model's performance and test accuracy. The test accuracy achieved by optimal BERT was around 88% and optimal Electra achieved 91% but optimal RNN-LSTM achieved 94%. This shows that for our dataset(IMDB) and for the selection of hyperparameters, RNN performs better than pretrained models like BERT and Electra. By performing these experiments, I could understand that the overall performance depends on the complexity of the model and the selection of the hyper parameters. When we have a business problem, we need to first perform exploratory data analysis to understand the data and the anomalies. After performing feature engineering and selecting the correlated features, we need to look into the models that are available to employ. Now, we start with simple models and if they can't handle it, only then should we move to complex models. From what I understood, we need to select a model that has optimal overall performance, takes less inference time and is easily interpretable. But as we know, there is usually a tradeoff between these factors. We need to choose a model in such a way that it performs well on the validation test as this will help us predict its performance on out of sample data points, aka test dataset. We perform hyperparameter tuning to reduce the overall time taken to train the model and to improve the overall performance of the model. Thus, we need to select a model that has optimal performance and at the same time take less time to train and test. Lower the complexity, the better. This is to avoid overfitting/memorization of data.

Problems Faced: The models took a long time to run, with each epoch taking more than 20 mins to run and each training run iteration of the model took 2 hours to run. But, this is due to the batch size and the size of the dataset. LSTM and BERT are complex models which might also contribute to this. But the model was stable with uniform convergence and I did not notice any jumps in validation loss which means it converged to the minima without a problem.