Deep Learning HW1 - Fakes Detection

Instructions on how to run code:

For FFNN and LSTM: run the FFNN_LSTM.py file.
For Transformer model and Blind test results: run thehw1_transformers.py file.

All requirements and imports are present within the files. The **blind test results** are in the file 'blind_test_predictions.csv'. The data is formatted in two columns: 'bio' containing the biography text, and 'result' containing the prediction made by our best model (The Bert Base Cased Transformer model). The 'target' column contains labels 'REAL' and 'FAKE'.

1. **FFNN Training**

- **Data Preprocessing**: Basic text processing, such as removing stopwords, HTML tags, extra new lines, punctuations and lower casing. Padding is added based on the length of the longest bio in training data. The bios are truncated to the length of 256. The labels "REAL" and "FAKE" are maintained in another array.

- **Data loaders**: We have defined a dictionary which assigns integers using a counter to new words encountered.

- **Model Architecture**:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 256, 128)          15022464

 dense (Dense)               (None, 256, 128)          16512

 dense_1 (Dense)             (None, 256, 64)           8256

 dense_2 (Dense)             (None, 256, 1)            65

 flatten (Flatten)           (None, 256)               0

 dense_3 (Dense)             (None, 1)                 257

=================================================================
Total params: 15,047,554
Trainable params: 15,047,554
Non-trainable params: 0
```

- **Loss function and Hyperparameters**:

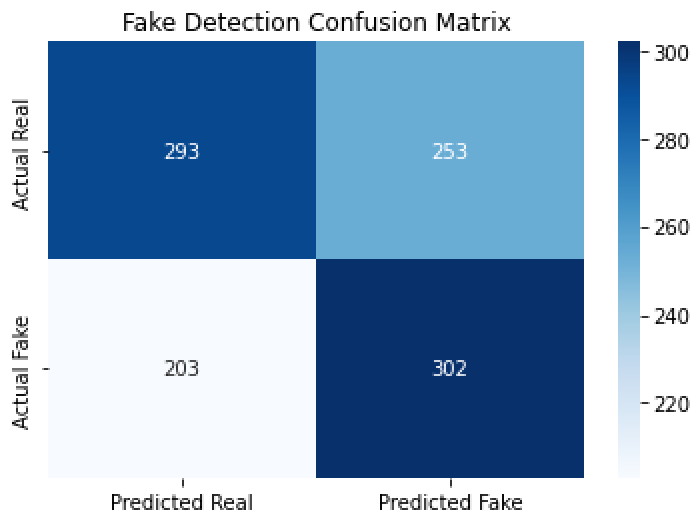
```
optimizer = tf.keras.optimizers.Adam(8e-5)
```

```
loss = tf.keras.losses.BinaryCrossentropy()

metrics = [tf.keras.metrics.BinaryAccuracy()]

batch_size = 32

epochs = 4
```

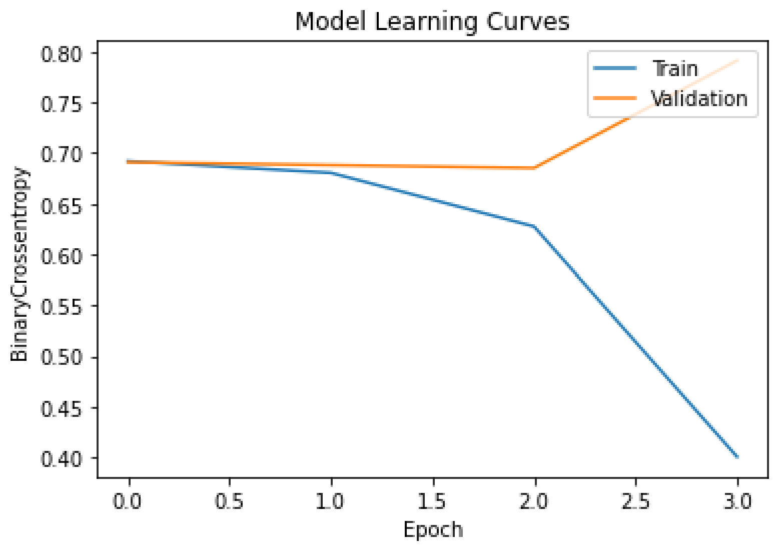
- **Test set performance:**

```
FFNN Accuracy : 0.566127497621313
```

Confusion matrix for the test set



Model Learning Curves for the test set



Limitations and Possible solutions: The model is designed to be simple and basic, but this also means that it may not perform as well as more complex models like Transformers. In fact, with an accuracy rate of just 56.6%, it's possible that a person randomly guessing could outperform the model. Another issue with the model is that it's prone to overfitting, which can lead to poor generalization and lower accuracy on new data. Therefore, it may be necessary to consider using more sophisticated models or regularization techniques to improve performance and prevent overfitting.

2. LSTM Model

- **Data Preprocessing**: Basic text processing, such as removing stopwords, HTML tags, extra new lines, punctuations and lower casing. Padding is added based on the length of the longest bio in training data. The bios are truncated to the length of 256. The labels "REAL" and "FAKE" are maintained in another array.
- **Data loaders**: We have defined a dictionary which assigns integers using a counter to new words encountered.
- **Model Architecture**:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 256, 128)          15022464
 lstm (LSTM)                 (None, 256, 50)           35800
 lstm_1 (LSTM)               (None, 50)                20200
 dense_4 (Dense)             (None, 64)                3264
 dense_5 (Dense)             (None, 1)                 65
=================================================================
Total params: 15,081,793
Trainable params: 15,081,793
Non-trainable params: 0
```

- **Loss function and Hyperparameters**:

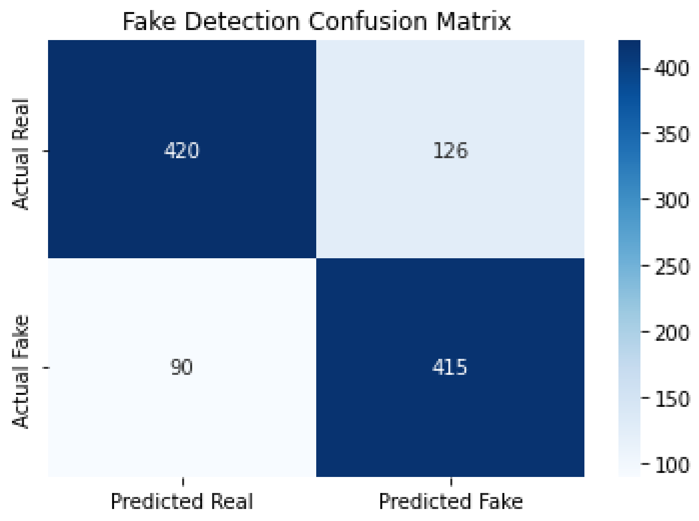
```
optimizer = tf.keras.optimizers.Adam(8e-5)
loss = tf.keras.losses.BinaryCrossentropy()
```

```
metrics = [tf.keras.metrics.BinaryAccuracy()]

batch_size = 32

epochs = 4
```

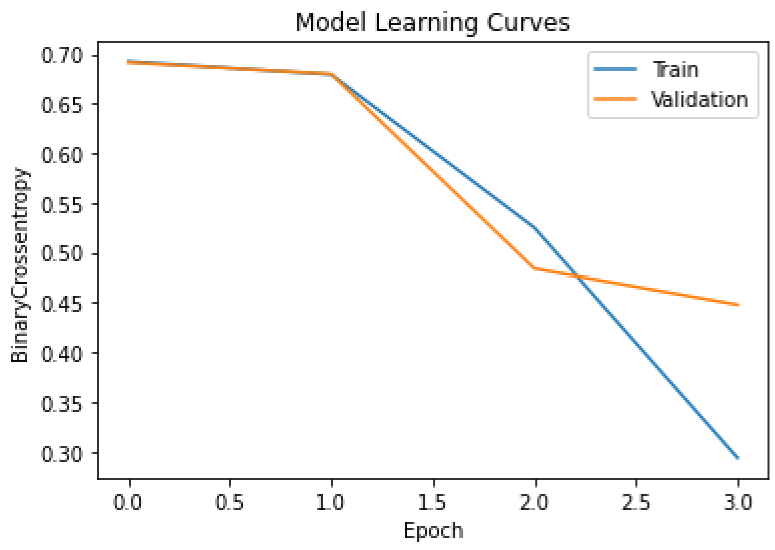
- **Test set performance:**

```
LSTM Accuracy : 0.7944814462416746
```

Confusion matrix for the test set



Model Learning Curves for the test set



- **Limitations and Possible solutions**: To avoid overfitting, it's important to ensure that bios are not truncated to lengths shorter than 100. On the other hand, using longer bios, exceeding 300, can also result in decreased accuracy, which may be attributed to padding. When bios are padded to match the length of the longest bio, it can have a negative impact on accuracy, especially for shorter bios that require excessive padding. Therefore, it's crucial to find a balance between bio length and padding to achieve optimal model accuracy.

3. Transformer Model (The Bert Base Cased Transformer model) + Blind Set Results

- **Data Preprocessing**: We believe the BERT tokenizer is robust enough to handle the noise in the data and hence Data preprocessing is not done.
- **Model Architecture**: **The Bert Base Cased Transformer model**
- **Why this model was chosen:**

  The BERT Base Cased Transformer model is considered to be a powerful model in natural language processing tasks due to several reasons:

  Pre-training: The model is pre-trained on a massive amount of data, allowing it to learn the nuances and patterns of language. This pre-training allows the model to perform well on a wide range of natural language tasks, including text classification, question-answering, and language generation.

  Fine-tuning: After pre-training, the BERT model can be fine-tuned on a specific task by adding an additional layer on top of the pre-trained model. This approach has shown to be effective in improving performance on various NLP tasks, as the pre-trained model already has a good understanding of language.

  Contextual understanding: The model has the ability to understand the context of words in a sentence, allowing it to capture the meaning of words that have multiple meanings depending on the context in which they are used.

  Attention mechanism: The BERT model uses a self-attention mechanism that allows it to focus on the most important parts of a sentence, making it particularly useful for tasks that require understanding longer texts.

  Overall, these factors contribute to the effectiveness of the BERT Base Cased Transformer model in a wide range of NLP tasks and thus it was chosen.

- **Test set performance:**

```
Transformer accuracy: 0.8279467680608364
```

- **Limitations and Possible solutions**:
  The model has an excessively large number of parameters, which significantly increases its computational demands, leading to longer processing times and requiring more powerful computing resources.
  .