Assignment 3

Exercise 1

1) Recurrent Neural Networks (RNNs) are a class of neural networks designed to handle sequential data, such as time series, text, or audio, by capturing temporal dependencies. Unlike traditional feedforward neural networks (FNNs), where information flows only in one direction from input to output, RNNs have loops that allow information to persist across time steps. In contrast, Feedforward Neural Networks (FNNs) process data in a unidirectional manner. The information flows from input layers through hidden layers to the output layer, with no internal memory or state maintained between inputs. FNNs are good for tasks like image classification, where each input can be treated independently and doesn't depend on past data. RNNs can maintain a memory of past inputs by leveraging recurrent connections, allowing them to handle sequential data. FNNs, on the other hand, treat each input independently, making them unsuitable for tasks requiring a memory of previous states or temporal dependencies.

2) The vanishing gradient problem is a common issue encountered during the training of deep neural networks, especially RNNs. It occurs when the gradients used to update the network's weights become extremely small as they are backpropagated through time (in RNNs) or layers (in deep FNNs). This causes the network to stop learning, as weight updates become insignificant. This is a problem in RNNs because they rely on backpropagation through time (BPTT) to compute gradients for each time step and update the weights accordingly. In long sequences, the gradients can shrink exponentially as they are propagated through each time step. As a result: Learning Long-Term Dependencies Becomes Difficult. Because the gradients diminish over time, the model fails to capture long-term dependencies and focuses primarily on recent inputs. This means that the network effectively "forgets" earlier information in the sequence. It can also lead to slow convergence. With very small gradients, the learning process slows down significantly because the weight updates are minimal, making it harder for the network to converge. So the vanishing gradient problem in RNNs can affect training by limiting the network's ability to learn long-term dependencies, slowing down the convergence rate, and leading to suboptimal performance, especially on tasks that require learning from long sequences of data. It can cause the network to focus on short-term patterns and make it difficult to train deep or large RNN models effectively.

3) RNNs are particularly useful in tasks that involve sequential data, as they can model temporal dependencies and maintain context over time. Here are two application areas where RNNs have been highly effective:

   1. Natural Language Processing (NLP)
   RNNs are widely used in NLP applications due to their ability to capture context and word dependencies in sentences and longer texts. Common NLP tasks where RNNs excel include:

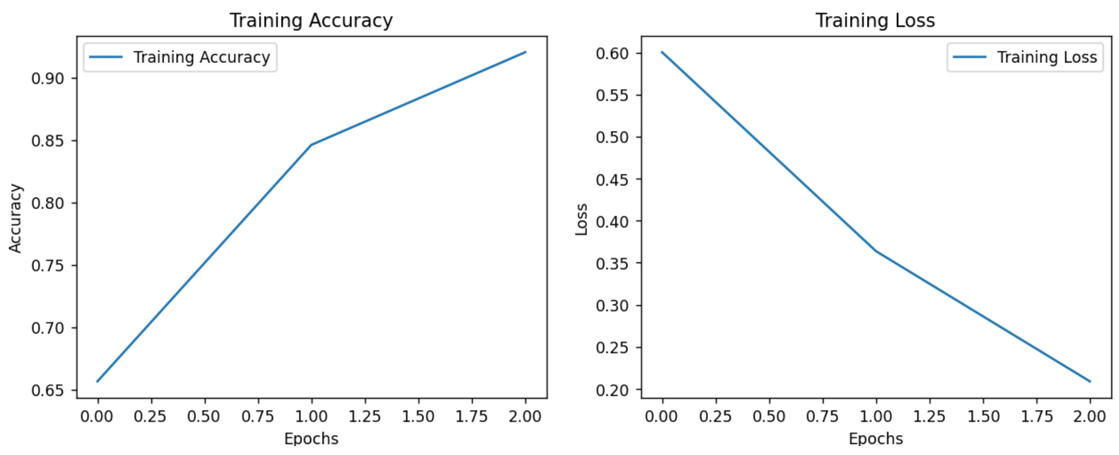
Language Modelling: Predicting the next word in a sentence based on the previous words (e.g., used in predictive text).
Text Generation: Generating coherent text by learning patterns from large text corpora.
Machine Translation: Translating text from one language to another by processing sequences of words and generating corresponding sequences in another language.
Speech Recognition: RNNs can process audio signals, which are sequences of sound frames, and convert them into text or transcriptions.

2. Time Series Prediction

RNNs are also highly effective for time-series forecasting, where the goal is to predict future values based on historical data. This is common in domains such as:
Financial Data Forecasting: RNNs can be used to predict stock prices or market trends by modelling the temporal patterns in the financial data.
Weather Forecasting: Using RNNs, meteorologists can predict future weather conditions based on previous observations, which helps in providing accurate forecasts.

Both these application areas benefit from RNNs' ability to process and model sequential data, making them suitable for tasks where understanding past data is crucial for predicting future outcomes.

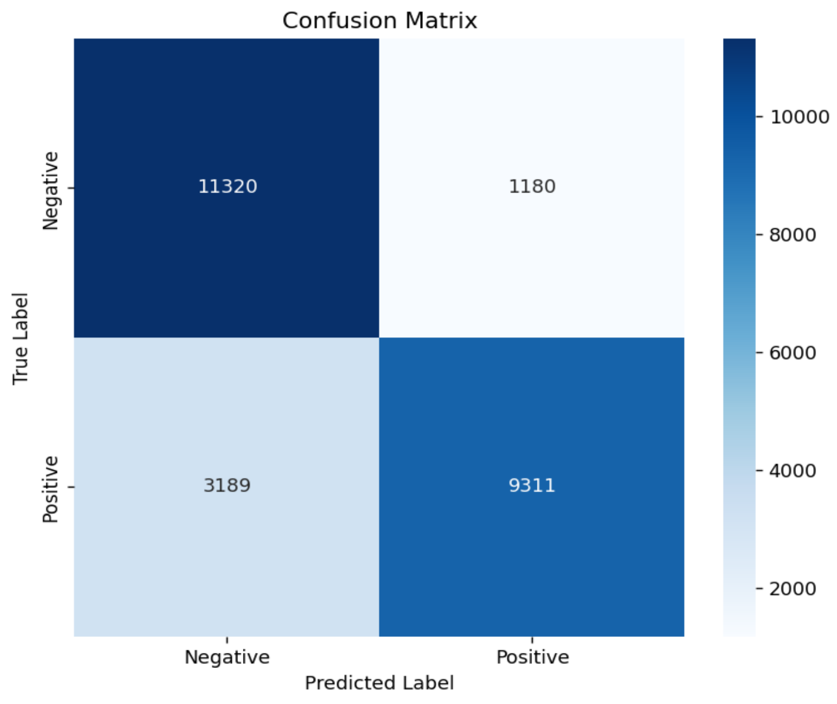
4) For this task, my model consists of an Embedding Layer, a Simple RNN Layer and a Dense Output Layer. The results are the following:



Training Accuracy and Loss Plot:

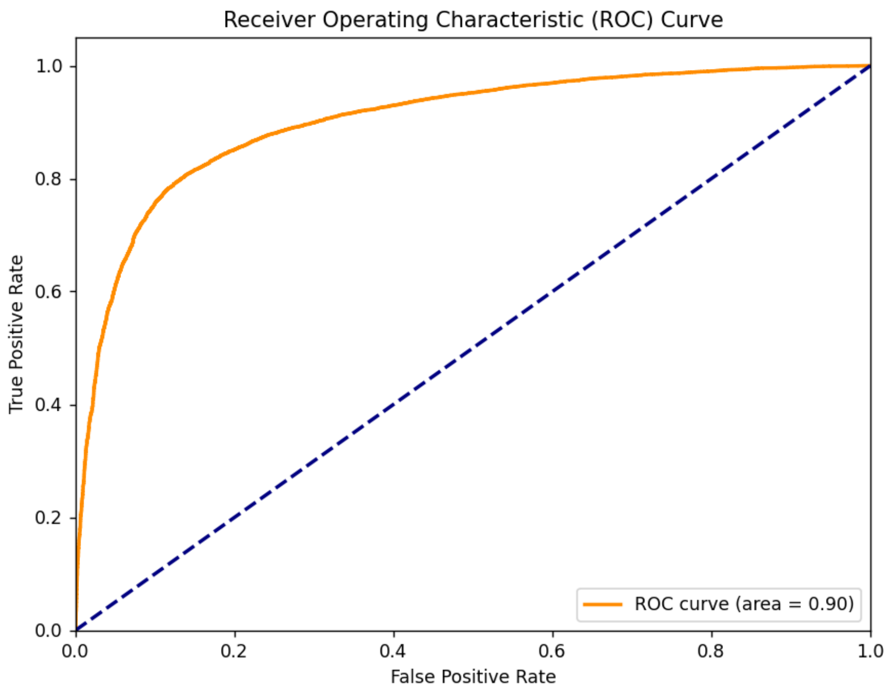
Training Accuracy: The accuracy improves across the epochs, starting from around 66% and reaching above 90%. This indicates that the model is learning effectively during training.

Training Loss: The loss decreases progressively, suggesting that the model is minimizing errors as it is trained.



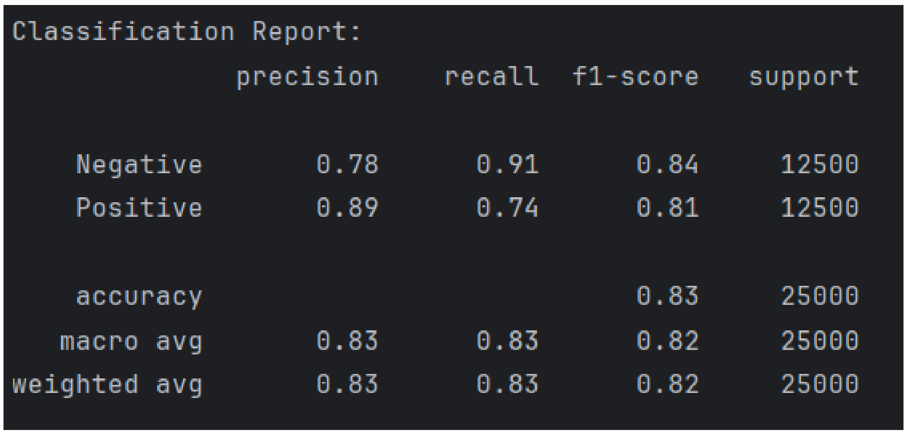
Confusion Matrix

The model performs well having predicted correctly 11320 negatives and 9311 positives, while 3189 positives were predicted as negatives and 1180 negatives were predicted as positives.



ROC Curve

The model has a very good performance, as reflected by the high AUC score of 0.90.


```
Classification Report:
              precision    recall  f1-score   support

    Negative       0.78      0.91      0.84     12500
    Positive       0.89      0.74      0.81     12500

    accuracy                           0.83     25000
   macro avg       0.83      0.83      0.82     25000
weighted avg       0.83      0.83      0.82     25000
```

Classification Report

Overall Accuracy: The model achieved an accuracy of 0.83, indicating that 83% of the predictions were correct.

Precision, Recall, and F1-Score: These metrics (avg) are consistently high (around 0.82-0.83) suggesting a relatively balanced performance in terms of precision (correct positive predictions), recall (correctly identified actual positives), and the harmonic mean of both (F1-score).

Support: Each category has a support value of 125000, meaning the model was tested on 12500 instances per category.

Exercise 2

1) Long Short-Term Memory (LSTM) networks are a special kind of Recurrent Neural Networks (RNNs) that are designed to effectively handle the vanishing gradient problem, which often affects standard RNNs. The key to LSTMs' ability to address this issue lies in their unique architecture, which allows them to maintain and update information over long sequences of data without the gradients vanishing during training. LSTMs consist of memory cells and a series of gates that control the flow of information. These gates enable the network to selectively remember or forget information over time, making it easier to capture long-term dependencies. The three main gates are:
   Forget Gate: Decides what information from the cell state should be forgotten or retained.
   Input Gate: Controls what new information from the current input should be added to the cell state. It works with the input and hidden states to update the cell's memory with new relevant information.
   Output Gate: Determines what information from the cell state should be output at the current time step to influence the hidden state. This hidden state is then passed to the next time step, preserving long-term dependencies.

How LSTMs prevent the vanishing gradient problem:
Constant Error Flow through Cell State: The central mechanism of LSTMs is the cell state, which acts as a "memory" or conveyor belt that flows through the entire sequence, carrying information forward unchanged unless modified by the forget or input gates. This allows LSTMs to retain information over long sequences without suffering from vanishing gradients, as the error gradients flow back largely unchanged through the cell state.

Gating Mechanism for Controlled Learning: The input, forget, and output gates provide a way for the network to regulate which parts of the input are remembered, forgotten, or passed on. This helps prevent irrelevant or noisy information from dominating the learning process, thereby ensuring that important information persists over time.

Better Gradient Propagation: Since the LSTM allows the network to preserve long-range dependencies without the gradients decaying to near-zero values, the error signals can propagate back through many time steps effectively. This leads to better gradient propagation during backpropagation through time (BPTT), avoiding the vanishing gradient problem that occurs in standard RNNs.

2) LSTMs have shown superior performance over standard RNNs in various application areas, particularly those that involve long-term dependencies in sequential data. Here are some key examples:

1. Natural Language Processing (NLP)
In many NLP tasks, understanding long-term dependencies in sentences is crucial. LSTMs outperform standard RNNs because they can capture the relationships between words that are far apart in the text.
Machine Translation: LSTMs are widely used in machine translation systems like Google Translate because they can remember long sentences and accurately translate them into another language by understanding the context of earlier words in the sentence.
Text Summarization: LSTMs are used in summarization tasks to generate coherent summaries by remembering key details from the original text, which standard RNNs often fail to do when the text is long.
Language Modelling and Text Generation: LSTMs are effective in generating text sequences, such as writing entire paragraphs or stories, because they maintain memory of the text generated so far, leading to more coherent and contextually relevant output.

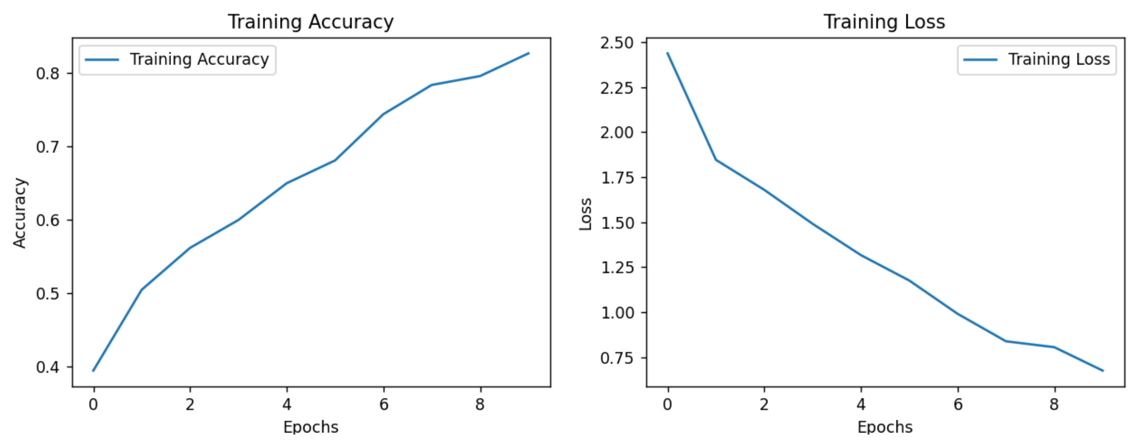
2. Speech Recognition
In speech recognition tasks, LSTMs outperform RNNs by processing sequences of sound waves or audio data over time and retaining important information across long stretches of audio input. Standard RNNs, due to the vanishing gradient problem, struggle with longer audio segments.
End-to-End Speech Recognition: LSTMs have become standard in speech recognition systems, allowing models to map spoken words to text more accurately by handling long utterances and sentences, even when pauses or noise occur in between.

3. Time Series Forecasting

LSTMs are highly effective in predicting future values in time series data, such as financial market trends, weather forecasting, and energy consumption predictions. These tasks require modelling both short-term fluctuations and long-term trends.
Stock Market Prediction: LSTMs perform better than RNNs in predicting stock prices by remembering trends over long periods, allowing them to make more informed predictions based on past patterns.
Weather Prediction: LSTMs can forecast weather conditions by processing historical weather data and retaining important seasonal or long-term patterns that affect future weather conditions, which is harder for standard RNNs.

4. Video Analysis
In video analysis, each frame in the video is part of a sequence, and capturing the temporal dependencies between frames is crucial for tasks such as video classification, action recognition, or video captioning.
Action Recognition: LSTMs can track long sequences of frames and maintain the necessary context to correctly identify actions that unfold over several seconds, such as running, jumping, or speaking, outperforming RNNs that might forget earlier frames.
Video Captioning: In tasks where the goal is to generate a textual description of what is happening in a video, LSTMs have demonstrated superior performance because they maintain contextual information across a series of frames.

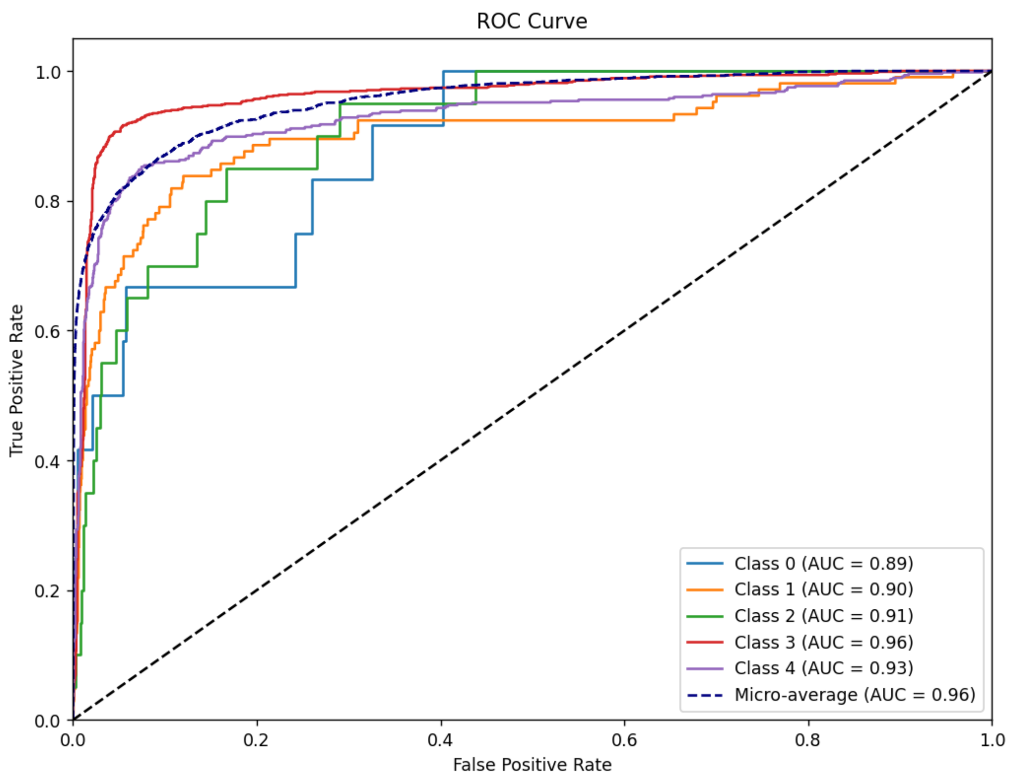
3) For this task, my model consists of an Embedding Layer, and LSTM Layer and a Dense Output Layer. The results are the following:
(A confusion matrix was not included in the results as there are many classes and the information from it was not clearly visible and confusing).



Training Accuracy and Loss Plot:

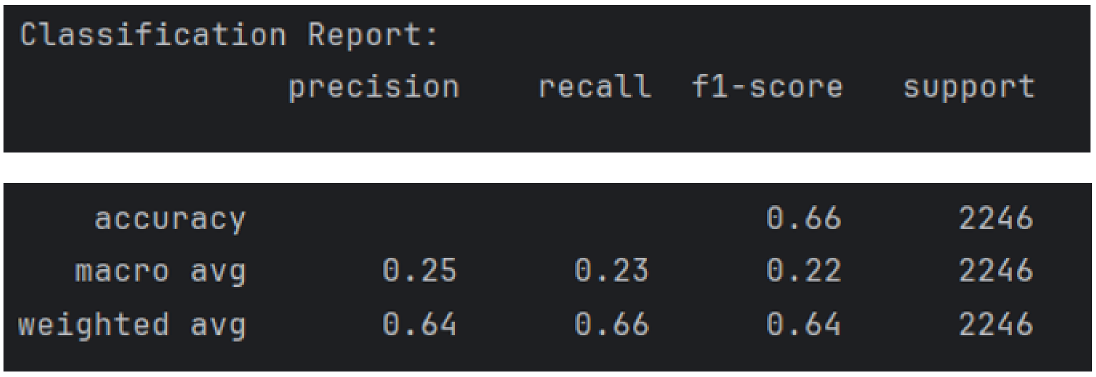
Training Accuracy: The accuracy improves across the epochs, starting from around 40% and reaching above 80%. This indicates that the model is learning effectively during training.

Training Loss: The loss decreases progressively, suggesting that the model is minimizing errors as it is trained.



ROC Curve

Here we can see the ROC curves of the first five classes and the micro-average ROC curve (all classes aggregated). The AUC score of the first five classes is between 0.89 and 0.96, with the aggregated AUC score of all classes being 0.96.



```
Classification Report:
              precision    recall  f1-score   support

    accuracy                           0.66      2246
   macro avg       0.25      0.23      0.22      2246
weighted avg       0.64      0.66      0.64      2246
```

Classification Report

Overall Accuracy: The model achieved an accuracy of 0.66, indicating that 66% of the predictions were correct.

Precision, Recall, and F1-Score: These metrics (avg) are around 0.64-0.66) suggesting a relatively balanced performance in terms of precision (correct positive predictions), recall (correctly identified actual positives), and the harmonic mean of both (F1-score).