Atharva Vaidya
23CS60R41

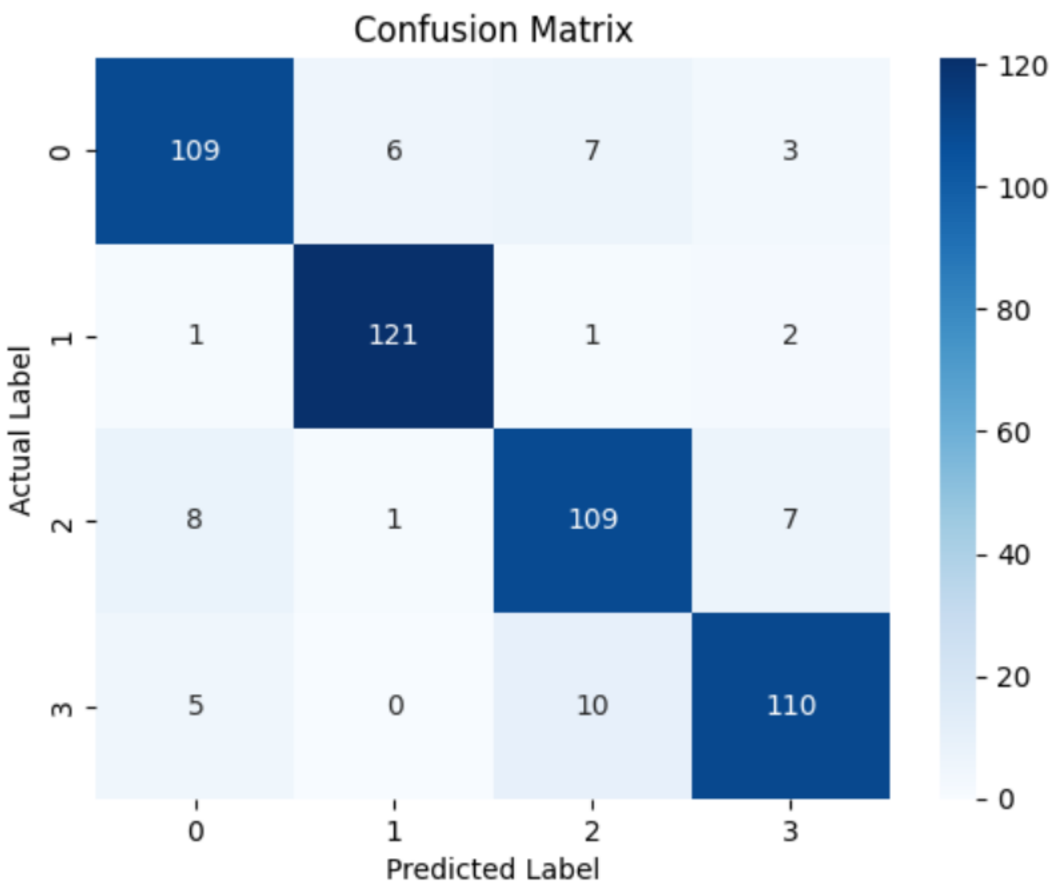
Task 2: Bert report

```
Accuracy: 0.898
Macro F1 Score: 0.897830405345907
```

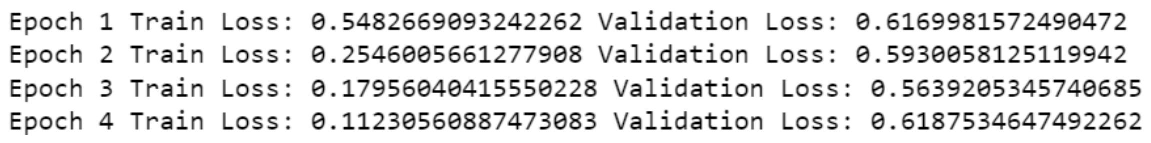
Classification Report:

```
              precision    recall  f1-score   support

           0       0.89      0.87      0.88       125
           1       0.95      0.97      0.96       125
           2       0.86      0.87      0.87       125
           3       0.90      0.88      0.89       125

    accuracy                           0.90       500
   macro avg       0.90      0.90      0.90       500
weighted avg       0.90      0.90      0.90       500
```

Confusion Matrix:



Loss per epoch:



```
Epoch 1 Train Loss: 0.5482669093242262 Validation Loss: 0.6169981572490472
Epoch 2 Train Loss: 0.2546005661277908 Validation Loss: 0.5930058125119942
Epoch 3 Train Loss: 0.17956040415550228 Validation Loss: 0.5639205345740685
Epoch 4 Train Loss: 0.11230560887473083 Validation Loss: 0.6187534647492262
```

HyperParameters used:

1. **max_seq_length**: 256 (Maximum length of the tokenized input sequences)
2. **batch_size**: 16 (Number of samples per batch of computation)
3. **num_labels**: Number of unique labels in the dataset (Dynamically determined from **train_df['label'].unique()**) (= 4 in our dataset).
4. **learning_rate (lr)**: 5e-5 (Learning rate for the AdamW optimizer)
5. **epochs**: 4 (Number of epochs to train the model)
6. **train_size**: 0.9 (Proportion of the dataset to include in the train split)
7. **device**: "cuda" or "cpu" (Whether to use GPU or CPU for training, depending on availability)

Challenges Faced and resolution:

1) One challenge faced while designing the BERT code was managing computational resources, as BERT models are large and require significant memory and processing power. This was addressed by setting an appropriate batch_size to balance between computational load and training efficiency. Using a smaller batch_size helped prevent out-of-memory errors on limited hardware resources.

2) Another challenge was selecting the right max_seq_length for tokenization. Using too large a sequence length could waste computation on padding tokens, while too short a length might truncate important information. This was resolved by analyzing the distribution of the text lengths in the dataset and choosing a max_seq_length that accommodated the majority of the text data without excessive padding.

3) Another challenge encountered was fine-tuning the learning rate for the BERT model. The choice of learning rate significantly impacts the model's training efficiency and final performance. If the learning rate is too high, the model may not converge, while a too low learning rate can lead to slow training. This was addressed by experimenting with different values and using the AdamW optimizer's default learning rate as a starting point, then adjusting based on the training progress and validation performance to find an optimal rate.