**HW7 – Parametric Efficient Fine Tuning (PEFT)**

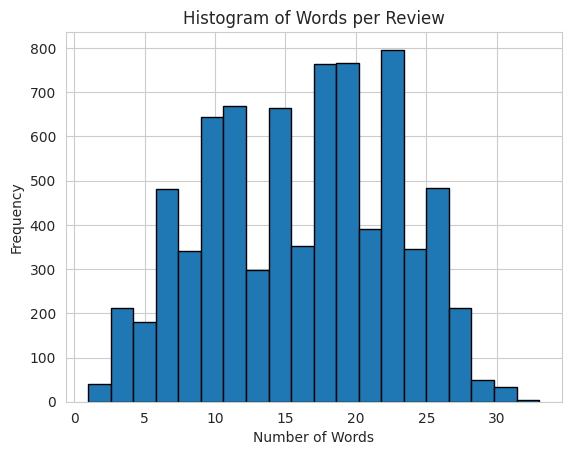
**Objective:** This main aim of this assignment is to explore different parameter efficient fine-tuning (PEFT) methods on different Large Language models (LLMs) and evaluate the performance of them for the Tweet Detection task. This task is a classification task with an objective to predict multiple emotions conveyed in a single tweet using different models.

**Dataset Explanation:** The dataset used for various tasks in this assignment is taken from the on-going competition and it is a Emotion Detection Dataset with unique tweets and each tweet is labeled with a range of emotions such as anger, joy, optimism, sadness, and more. The dataset has two files ‘train.csv’ and ‘test.csv’. The ‘train.csv’ file is used to train the models as well as the evaluate the performance of the models. This train dataset has 7724 tweets with 11 emotions namely anger, anticipation, disgust, fear, joy, love, optimism, pessimism, sadness, surprise, trust.

**Data Preparation:** Since the models that I used in this assignment does internal preprocessing of data during tokenization, no preprocessing on the data was done.

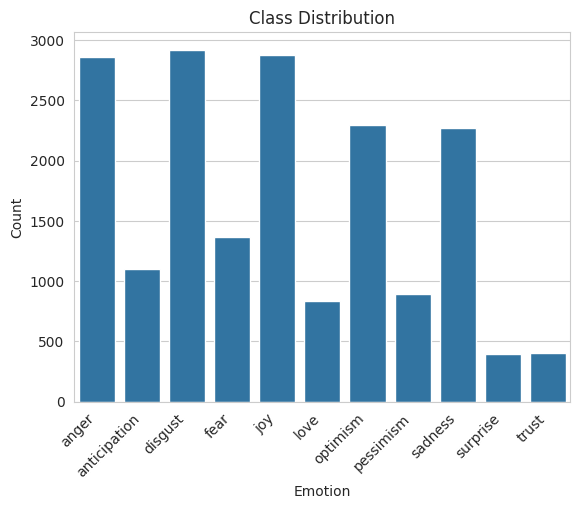
**Exploratory Data Analysis:**

Analysis of the length of the text:



From the above histogram, we can see that the max length of tweets is 35 which is less than the maximum tokens of the models used in this assignment. Therefore, there is no need for truncation of any tweet to fit the capabilities of the proposed models.

Class Distribution Analysis:



From the above bar graph, we can see that there is class imbalance among the distribution of labels across the tweet data. It can be observed that that the emotions like anger, disgust, joy are more occurring i.e. more than 2500 times as compared to the emotions like surprise and trust which are occurring less than 500 times. This highlights the class imbalance present in the data and it is important to take the class imbalance into consideration for better performance of the model.

This class imbalance was handled by cost sensitive learning by assigning different weights to the instances of the class labels during training. Higher weights are given to the minority classes making them more influential during training. This step is common for all the tasks accomplished in this assignment.

**Task 1 and its Implementation:**

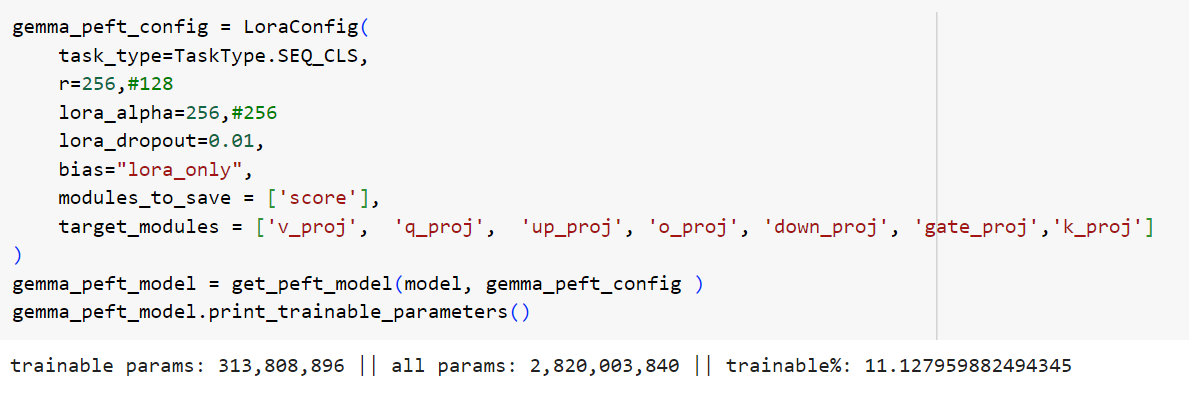
This task has two sub-tasks to implement:

**Task 1 Part A: *Use the `google/gemma-1.1-2b-it` model and Fine-tune using LoRA***

The main goal of this task it to fine-tune a decoder-only ‘gemma-1.1-2b-it’ model using LoRA. This model is a latest instruct version of Gemma model by Google with 2 billion parameters. I used this model to do sentiment analysis on the emotion detection dataset to do the multi-label classification of the emotions. Since fine-tuning requires updating all the parameters of the decoder model, I fine-tuned this model using one of the PEFT methods called LoRA (Low-Rank Adapation) in order to adapt my model for the classification task.

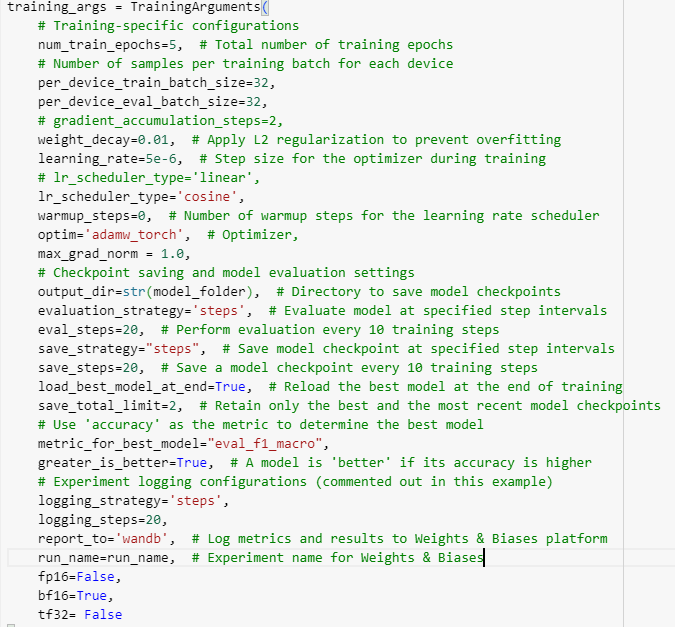
LoRA freezes the parameters of the original model and decomposes the weights updates into two low-rank matrices A and B thus reducing number of trainable parameters. This increases the efficiency of the fine-tuning for down-stream tasks like classification and reduces the memory required to train this model.

In the model, LoRA is applied on the linear layers of the Gemma model. For this task, I have applied LoRA on all the linear layers including the attention layers and did not apply LoRA on the ‘score’ module which is the classification head.



I took the rank of the decomposed matrices to be 256 and the value of alpha to be 256. This reduced the number of updatable parameters to 11.127% out of 2 billion parameters. Changing the rank changed the number of updatable parameters. Increasing it increased the parameters to update and vice versa. After trying different ranks, the above values gave me better results.

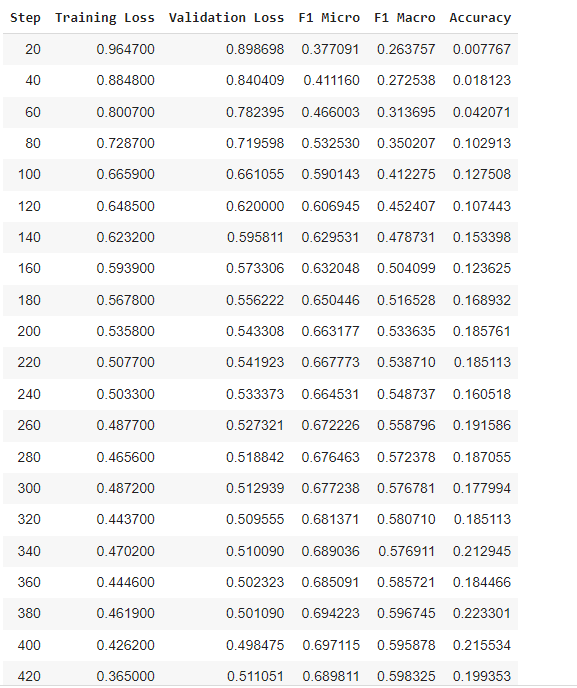
Training: Since this is a decoder-only model, the input to the classification head is the last hidden state which is the <EOS> token. For training, following hyperparameters are chosen:

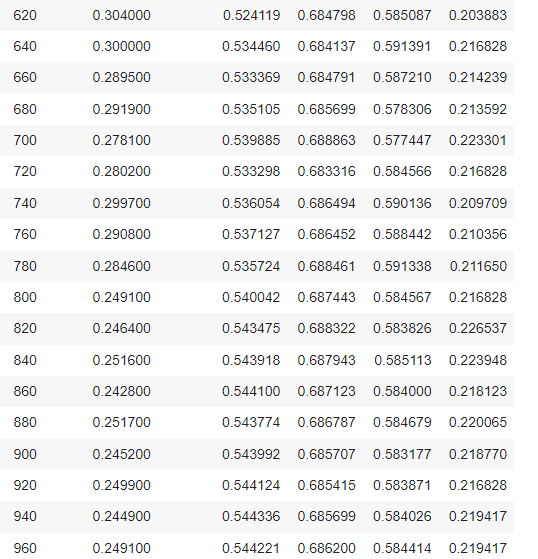


Notable hyperparameters in the model are learning scheduler type which is chosen as ‘cosine’ instead of ‘Linear’ to reduce the learning rate smoothly instead of decreasing it in a linear fashion.

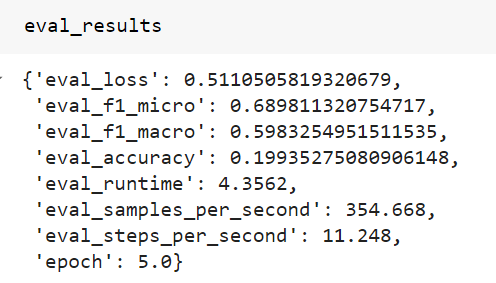
Next is the learning rate, among various learning rates tried, 5e-6 gave me the optimal results. Finally in this model, mixed precision is being used for model training. Generally, weight matrices are stored as floating point-32 in the memory. In order to make the training efficient by reducing the memory usage and computational requirements for the LLMs like Gemma, mixed precision is used where bf16=’True’ for training and evaluation. It has less precision than FP32 but same range as FP32.

Using the above hyperparameters, following results are obtained:





Validation Results:

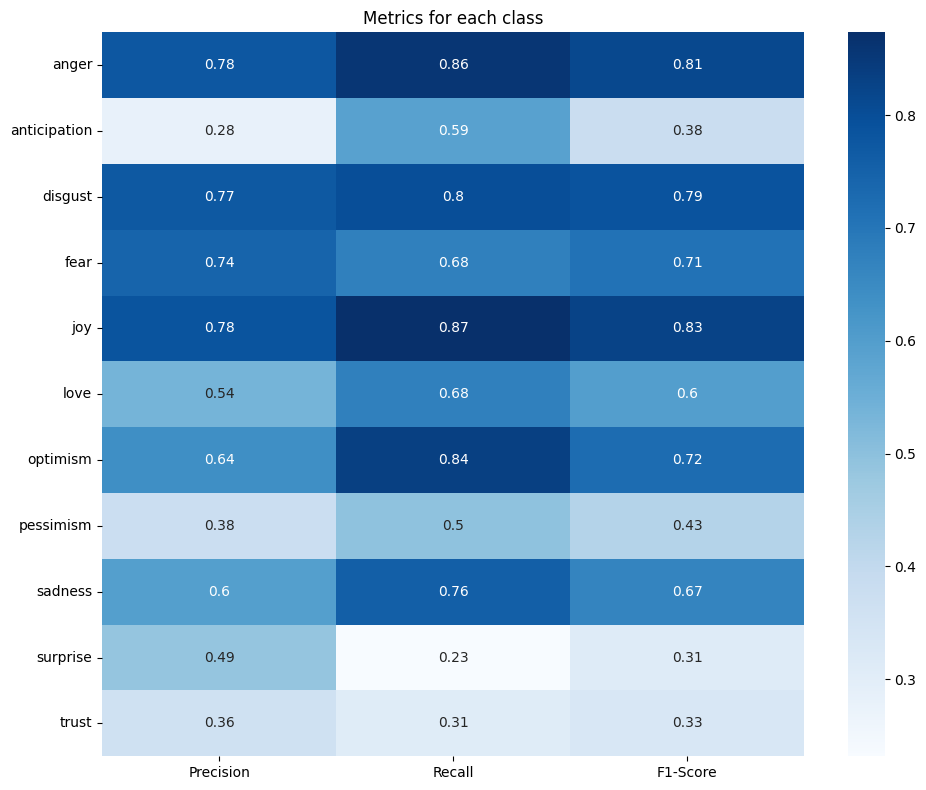


Based on the above results, the best model is found at step 420 where the F1-score is 0.598. This score is close to the F1-score obtained by Roberta which is 0.6167. Thus this shows that by updating few parameters, we can obtain performance close to the performance obtained through full fine-tuning.

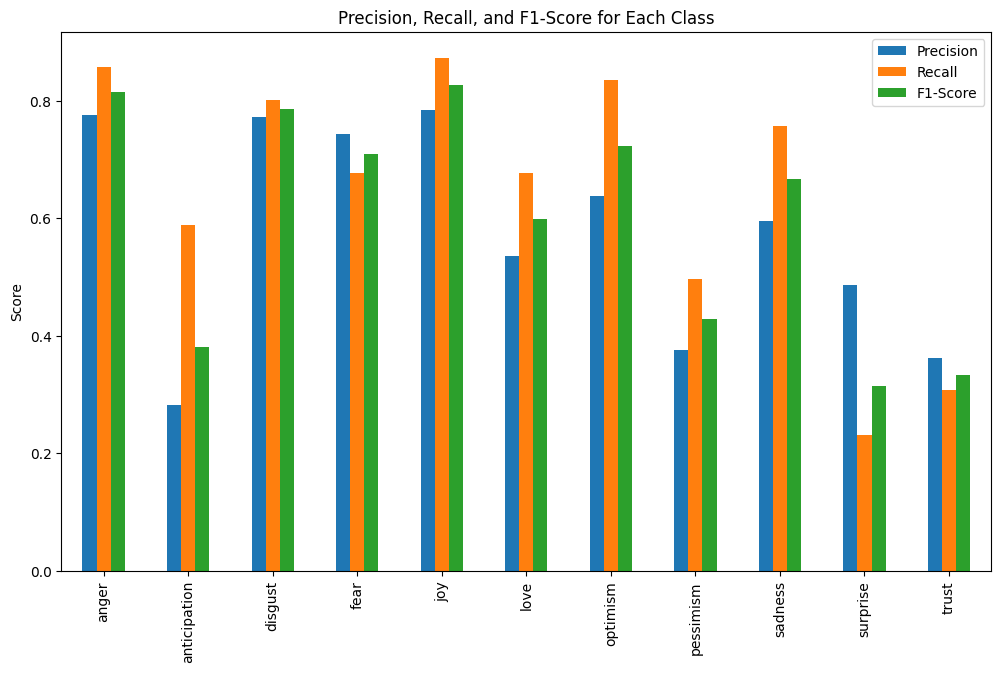
Confusion Matrix:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Heatmap of the confusion matrix:



The heatmap shows and improved and balance scores across precision and recall and F1-scores for all the labels after taking class imbalance into consideration.



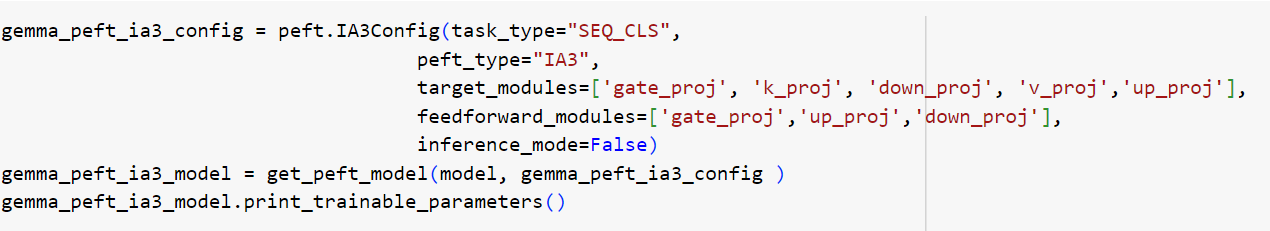


**Task 1 Part B: *Use the `google/gemma-1.1-2b-it` model and Fine-tune using IA3***

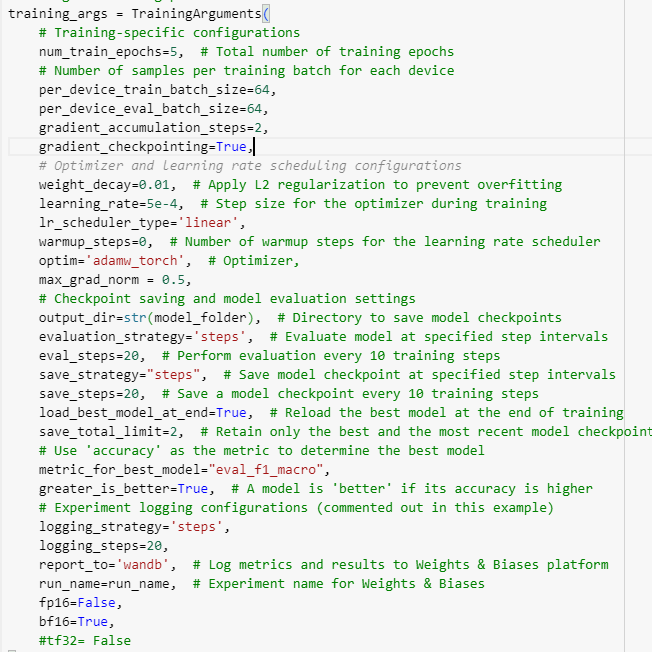
In the part B of the task 1, a different PEFT method is used on gemma model called IA3 where the keys and values of the attention layer as well as the hidden feed forward activation layers are rescaled. Unlike LoRA, IA3 modifies the weights of these layers during fine-tuning. This is a relatively simple technique as compared to LoRA which introduces two additional matrices.

Training:

Following configuration is used for IA3:

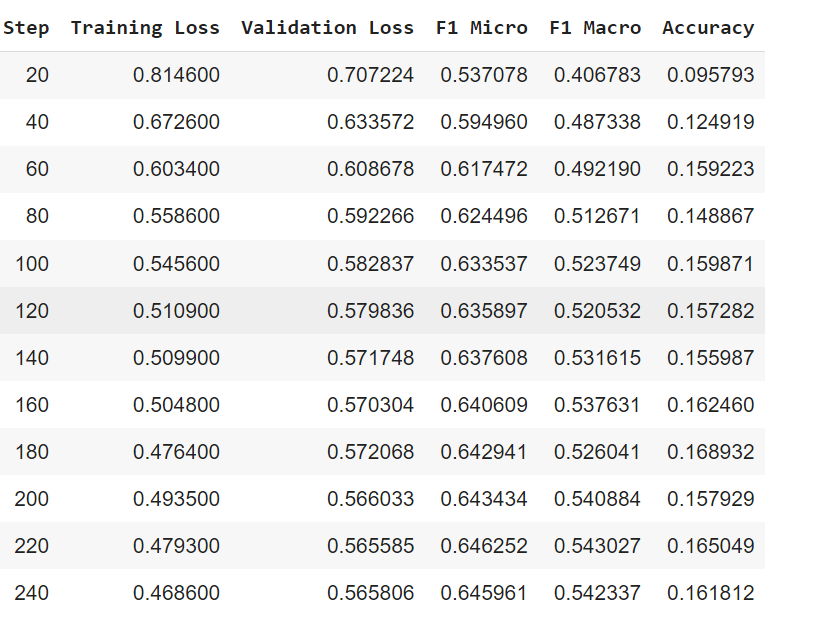


Here keys and values of the attention layer as well as the hidden feed forward activation layers are being modified by applying IA3.

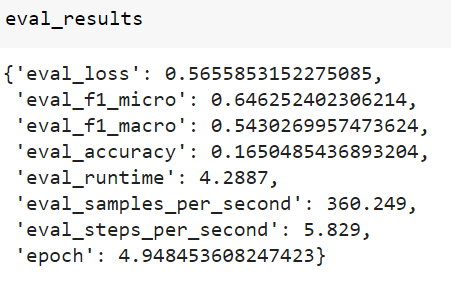


Similar hyperparameters as Part A are chosen for Part B task. Except the learning rate of 5e-4 is giving optimal performance of this model with IA3 implementation.

The trainer results are found to be:



Validation Output:



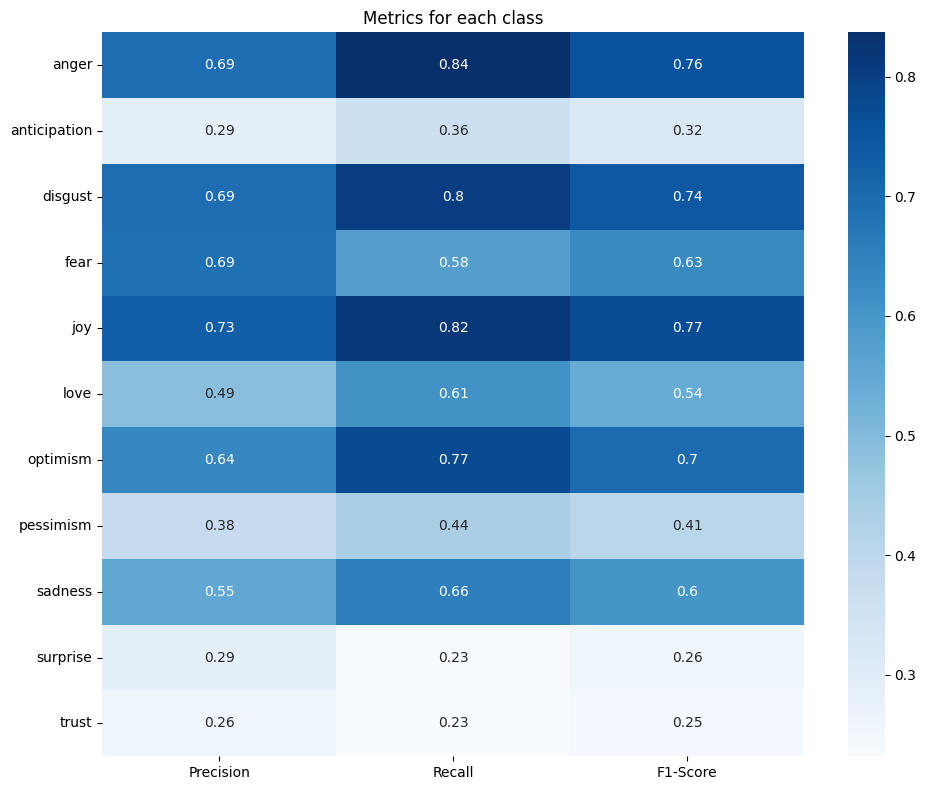
Based on the above results, the best model is found at step 220 where the F1-score is 0.543.

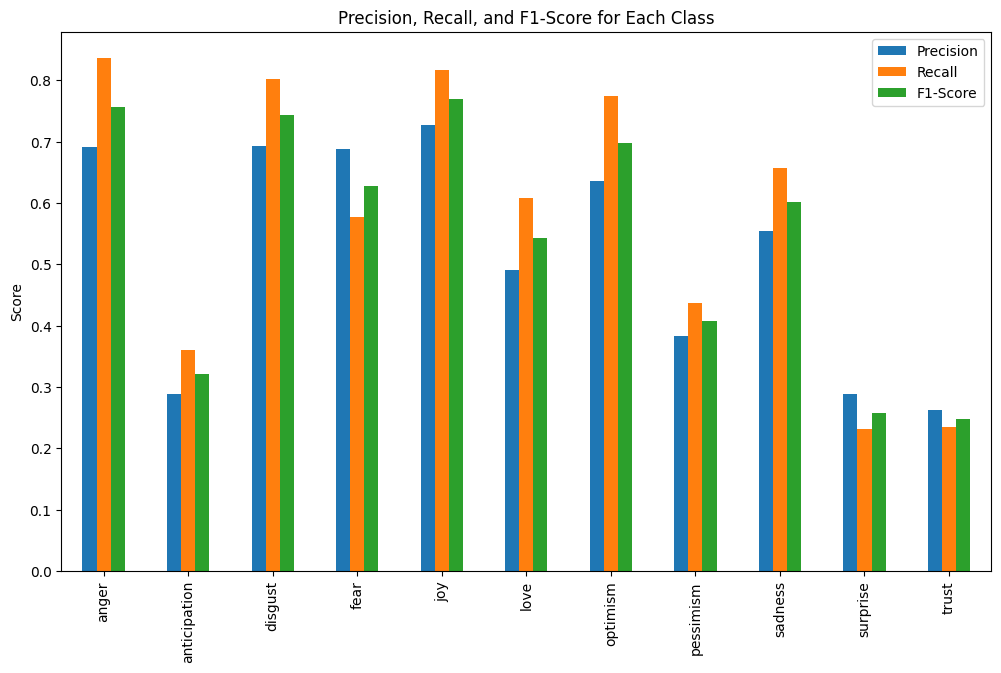


Confusion Matrix:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Heat Map of Confusion Matrix:



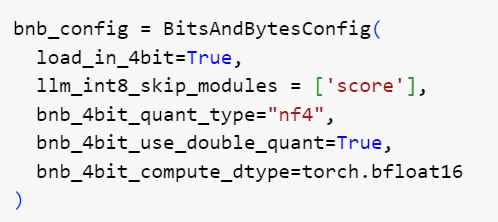


**Task 2 and its Implementation:**

**Task 2: *Select a model from the MTEB benchmark and fine-tune it using QLoRA.***

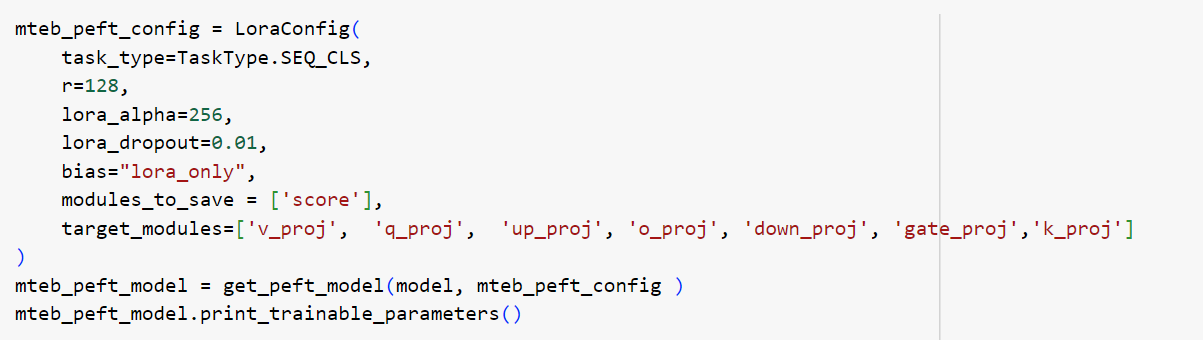
For this task I chose ‘intfloat/e5-mistral-7b-instruct’ from the MTEB benchmark. This model is a 7 billion parameter model which is an instruct tuned version of the ‘mistral’ model which is also a decoder-only transformer. For this task I chose this model and fine-tuned it using QLoRA. The training of mistral for a classification task is similar to that of gemma where the last hidden state is inputted to the classification head to do the multi-label classification.

Since, e5-mistral-7b-instruct is a 7 billion parameter model, in order to fine-tune it for the classification task, I fined-tuned it using QLoRA (Quantized LoRA) where it quantizes the decomposed low-rank matrices of LoRA to low precision representations i.e. 4-bit integers.

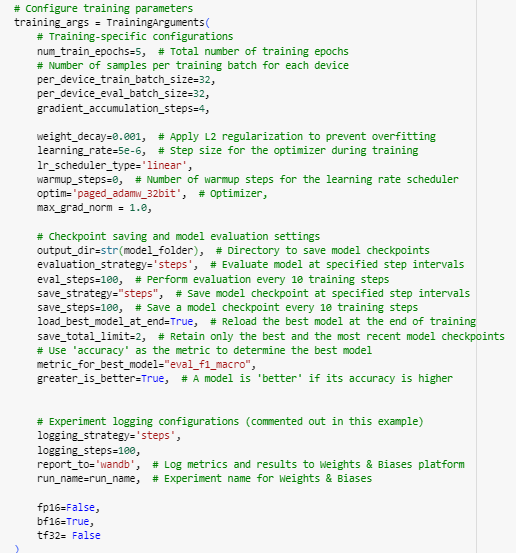


Here except for the classification layer, all the weights are quantized to low-bit precision i.e. 4-bit.

Following is configuration of QLoRA where QLoRA is applied on all linear layers except the classification layer.

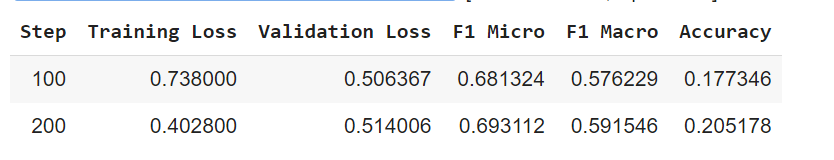


Training: Following hyperparameters are chosen for this model:

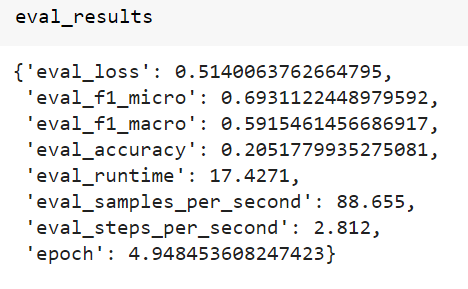
**

A notable hyperparameter used in the training of this model is the optimizer. Instead of Adam-W optimizer, a paged-adam-32bit optimizer is used where parts of the optimizer state or gradients is stored in the CPU in order to manage memory spikes during training. QLoRA quantizes the base model to 4-bit precision effectively reducing the memory utilized for storing the parameters.

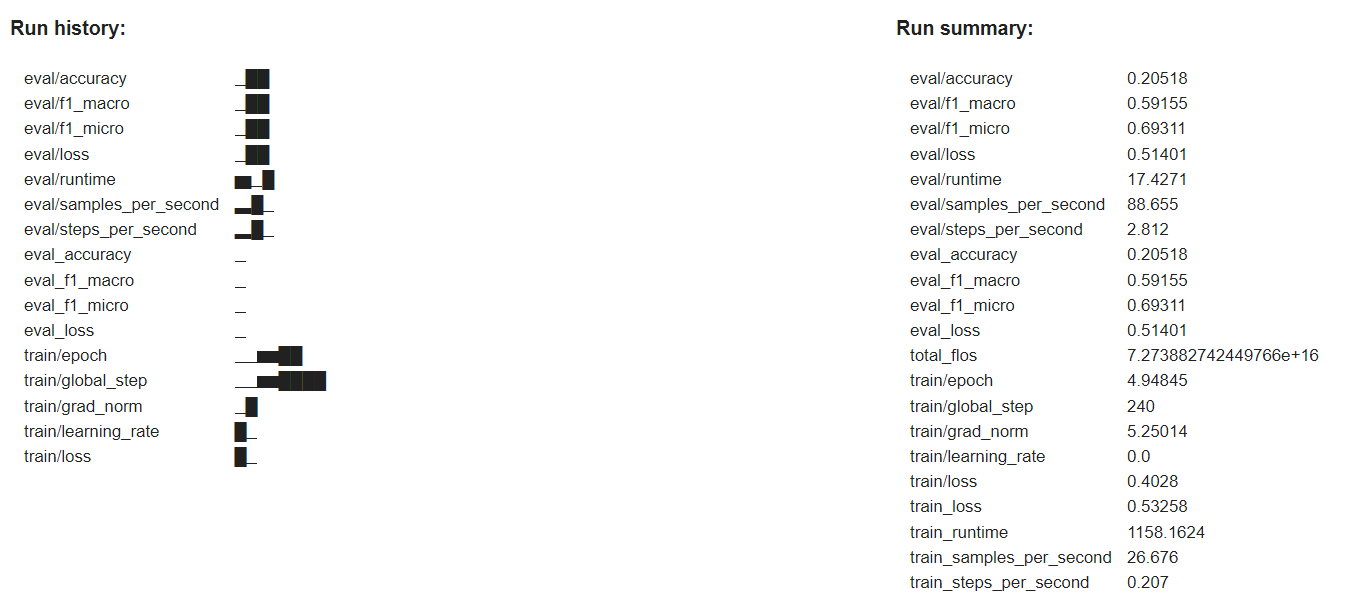
Trainer gave the following results with logging steps of 100:



Validation Output:



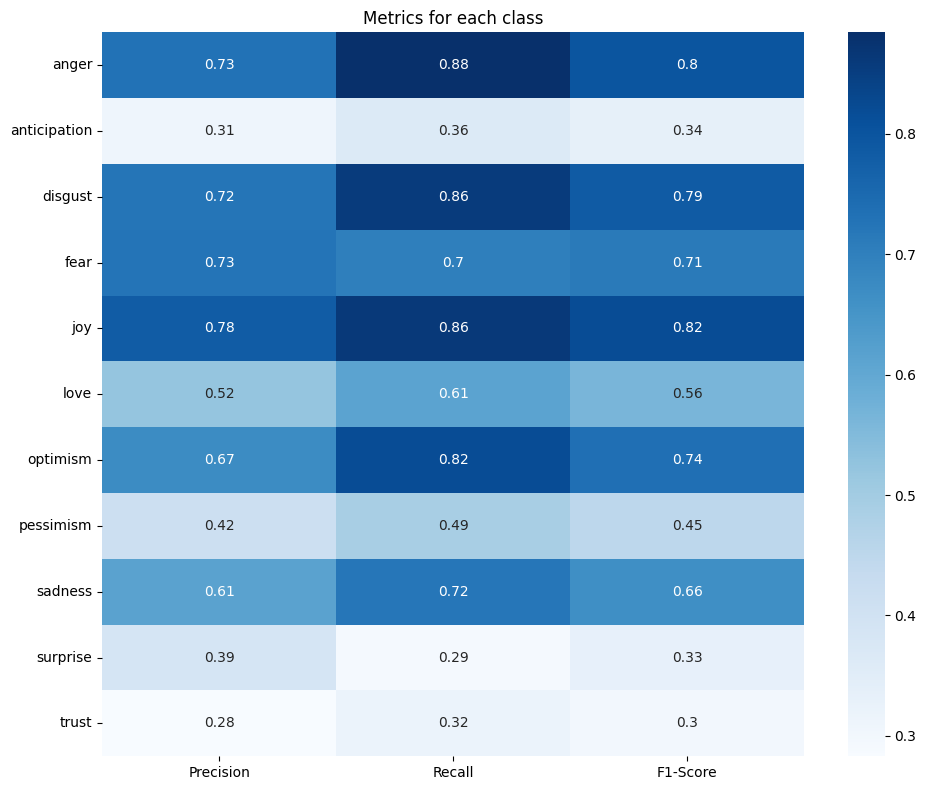
Based on the above results, the best model is found at step 200 with an F1-score of 0.5915. This gave similar performance to the of the Gemma model used in the first task.

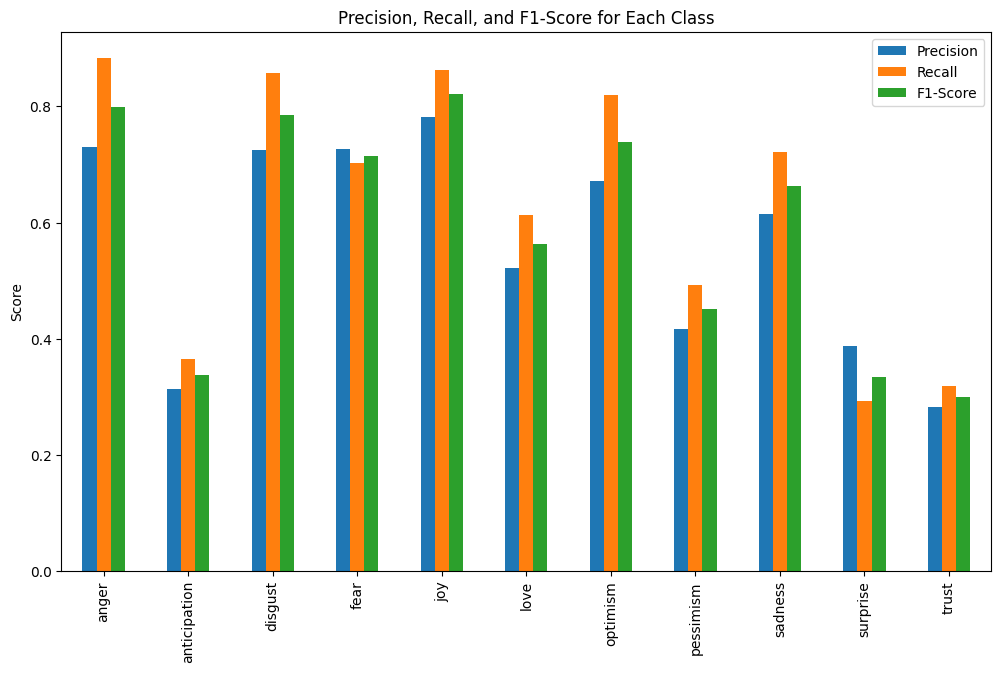


Confusion Matrix:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

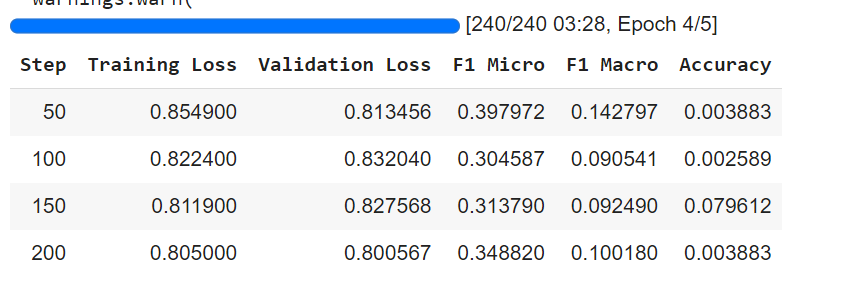
Heat map of the Confusion Matrix:



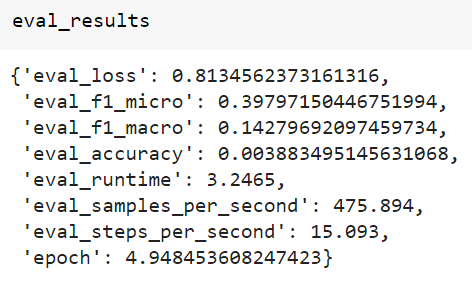


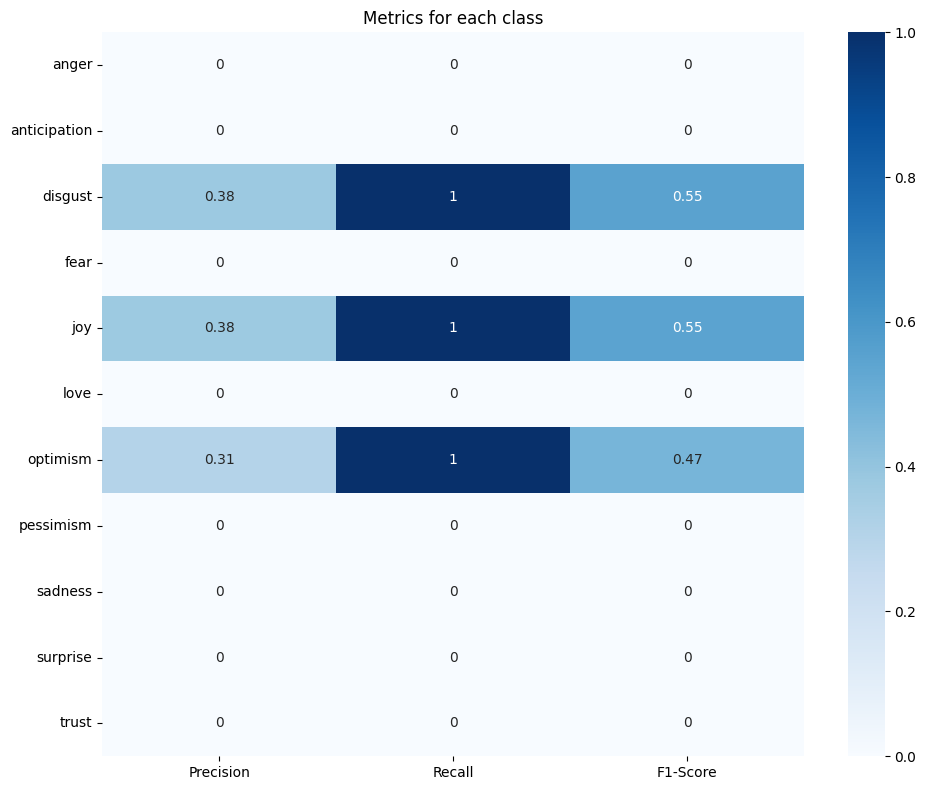
**Task 2: Implemented another model from MTEB benchmark called ‘intfloat/e5-base-v2’**

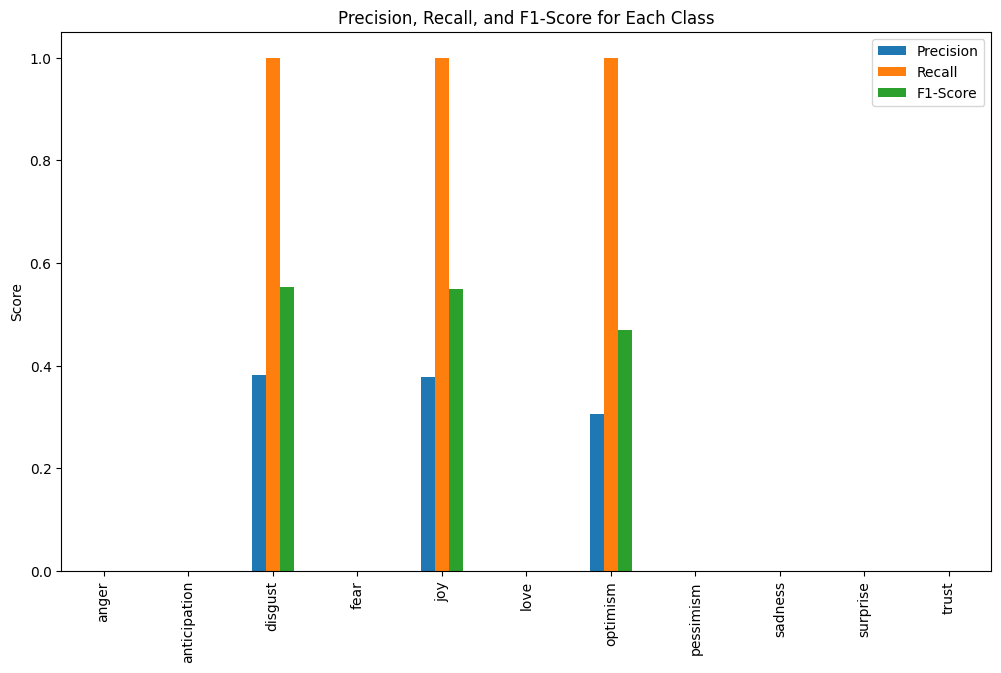
This model is a sentence transformer which takes the pooling of its word embeddings and input to the classifier head. On training this model, it performed poorly in classifying the tweets based on its pooled document-level embeddings. Trainer output is as following:



Validation Output:







It can be seen that this sentence transformer performed very poorly in doing the classification task.



**Metrics to Consider:**

For all the models, F1-score is taken as the main metric to evaluate the performance of the models on the sentiment analysis classification task.

|  |  |  |
| --- | --- | --- |
| **Model** | **Macro-F1 Score** | **Accuracy** |
| Gemma-1.1-2b-it using LoRA | 0.59833 | 0.19935 |
| Gemma-1.1-2b-it using IA3 | 0.54303 | 0.16505 |
| E5-mistral-7b-instruct using QLoRA | 0.59155 | 0.20518 |
| E5-base-v2 using QLORA | 0.1428 | 0.00388 |

**Comparison of the models and Performance evaluation:**

Based on the above metrics table, it can be seen that Gemma model using LoRA outperformed other models and tasks with an F1-score of 0.59833. It can also be seen that Gemma using IA3 did not perform as better as LoRA since LoRA better captures the knowledge and finetuned it for the classification task through its greater capacity to capture changes required for downstream task as compared to IA3 which is a simple rescaling of weights.

The mistral-7b-instruct model also performed as better as the Gemma model using QLoRA highlighting that QLoRA can give performance comparable to a full fine-tuned model. However, Gemma is a 2 billion parameter model and mistral-7b-instruct is a 7 billion parameter model, despite being a bigger model than gemma, Gemma outperformed it using LoRA demonstrating the potential of LoRA as powerful PEFT technique to train LLMs.

It can also be seen that the sentence transformer e5-base-v2 which is BERT model using mean pooling to create the document-level embedding performed very poorly in doing the classification task.

For all the three models, during inference, the updated weights in LoRA and QLoRA are combined with the base model to do the inference on test data.

Challenges Faced and Learnings:

* Computational capacity required to run these models is one of the major challenges that I faced during this assignment. My Google Colab GPUs exhausted their compute units while running these large models.
* Major learning during this assignment is that despite the models getting bigger with billions of parameters, I learnt that most these models have low intrinsic dimension for the downstream tasks making them more easily adaptable to the downstream tasks with the help of PEFT techniques.
* I tried the sentence transformers to check how well it performs the classification tasks but got very poor results despite hyperparameter tuning.

**WandB link for this project:** <https://wandb.ai/vyshnavi-utd/emotions_kaggle_gemma_S2024>