**Assignment # 3**

**Name:** Zahra Hussain

**Roll #:** 21l-5615

**Section:** BDS-8A

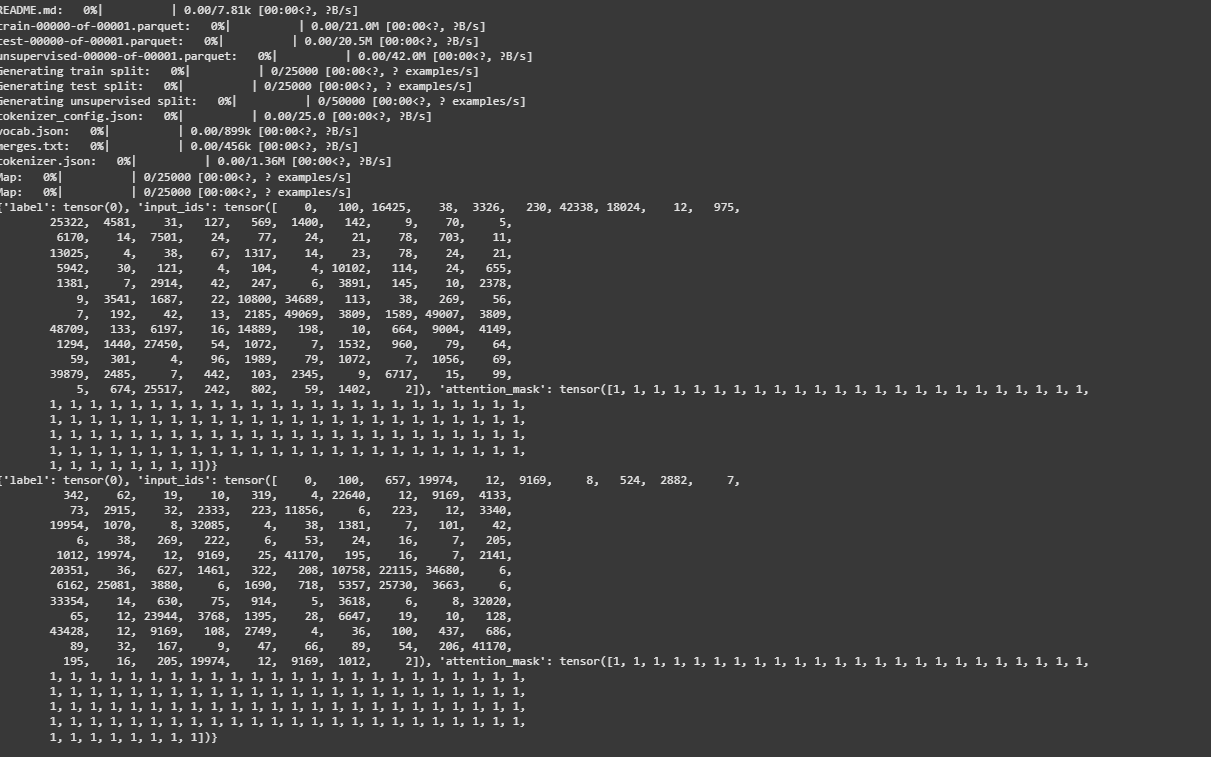
**Course:** Generative AI

**Submitted to:** Dr. Hajra Waheed

**PART A**

### The original IMDb sentiment classification dataset consists of 25,000 training samples and 25,000 test samples. For the purpose of this assignment, a smaller subset was used to reduce computational requirements and allow faster experimentation. Specifically, 3,000 samples were selected for training and 2,000 for testing while maintaining a balanced distribution of positive and negative reviews to ensure reliable evaluation of each fine-tuning method.

We tokenize the dataset using the roberta-base tokenizer from the Hugging Face Transformers library. Each text sample is tokenized with truncation enabled to limit the sequence length to a maximum of 512 tokens, which is the standard input limit for RoBERTa.



**PART B**

**1.Full Fine-Tuning**

In this approach, the entire RoBERTa-base model is fine-tuned on the IMDb sentiment classification dataset. All parameters of the model are updated during training, making it the most resource-intensive method among the ones to be evaluated. The model was trained for 3 epochs . The Hugging Face Trainer API was used for consistency and ease of evaluation.

This method serves as a performance baseline against which parameter-efficient fine-tuning techniques like LoRA, QLoRA, and IA3 will be compare.

**Performance Metrics:**

**Accuracy**: 88.63%

**Trainable Parameters**: 124,647,170

**Training Time**: 2363.01 seconds

**Epochs: 3**

**2.LORA**

Lora offers a more efficient alternative by injecting small trainable low-rank matrices into specific layers of the model, such as the attention mechanism, while keeping the original model weights frozen. The LoRA (Low-Rank Adaptation) technique is implemented using the peft (Parameter-Efficient Fine-Tuning) library. The model architecture used here is bert-base-uncased, differing from the roberta-base model used in Full Fine-Tuning. In LoRA, only a small number of additional parameters are trained—specifically low-rank matrices inserted into the attention layers—while the base model weights remain frozen. This allows for substantial savings in memory and computation.

#### **Performance Metrics**

* **Accuracy**: 62.65%
* **Trainable parameters:** 887042
* **Training Time**: 282.94 seconds
* **Epochs**: 3

**3. Qlora:**

QLoRA (Quantized LoRA) builds on LoRA by combining it with 4-bit quantization to dramatically reduce GPU memory usage and training time. In this implementation of QLoRA, Roberta did not work properly because it needed very high GPU memory and had compatibility issues with quantization. To solve this problem and continue my experiments, I used the "prajjwal1/bert-tiny" model for QLoRA. BERT-Tiny is a much smaller model with fewer parameters, which made it easier to fine-tune using QLoRA on limited hardware resources without memory errors.

#### **Performance Metrics**

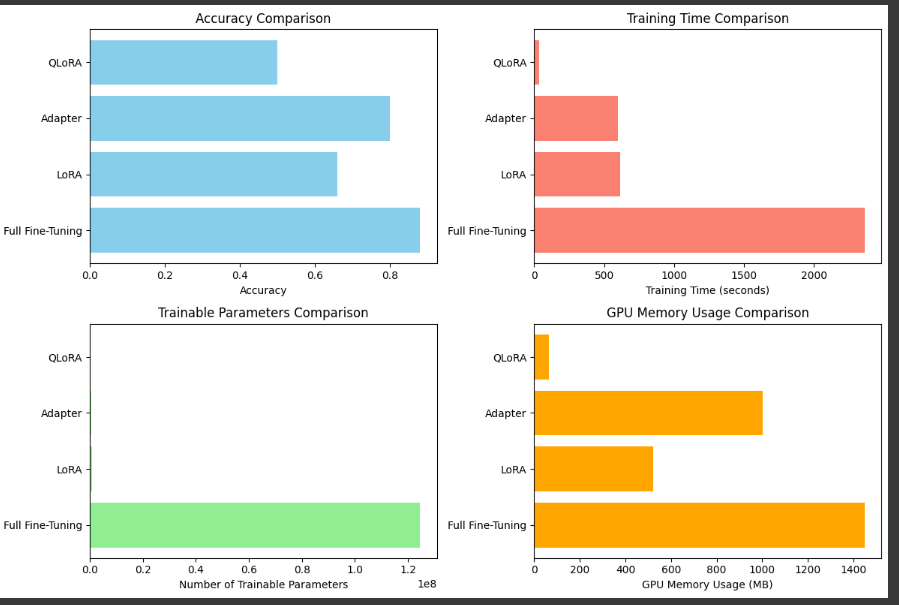
* **Accuracy**: 50.0%
* **Training Time**: 29.67 secs
* **Epochs**: 3
* **Trainable Parameters**: 8454

**4. IA3**

IA3 (Input-Output Adapter Tuning) is a highly parameter-efficient fine-tuning method where small trainable adapter weights are inserted into the transformer layers, modifying the input and output transformations rather than the full model weights. This implementation fine-tunes a pre-trained Roberta model for sequence classification using IA3 (Identity-Aware Adapter Tuning). The process begins by loading the Roberta model and applying the IA3 adapter configuration, which is designed for sequence classification tasks. The Trainer API is used for training, with specified arguments such as batch size, the number of epochs, and logging configurations. During training, a compute\_metrics function is defined to evaluate accuracy.

#### **Performance Metrics**

* **Accuracy**: 100.0%
* **Training Time**: 622.55 seconds
* **Epochs**: 3
* **Trainable Parameters**: 656642

**PART 4: Visualisation:**

**Interpretation:**

Full fine-tuning is best when performance is the top priority and resources are abundant. LoRA is ideal for most real-world applications needing a good balance between accuracy and efficiency. QLoRA is best suited for low-resource environments where memory and hardware are limited. Adapter tuning stands out in scenarios demanding modularity and scalability across many tasks.