**LoRA Fine-Tuning Experiment on BLOOM**

**Objective**

The goal of this lab was to explore efficient fine-tuning of a pre-trained language model using the LoRA method from the PEFT library. The aim was to adjust lora\_config parameters and evaluate whether strong results could be achieved in fewer epochs, saving time and compute.

**Experimental Setup**

* **Model used:** bigscience/bloom-560m
* **Dataset:** fka/awesome-chatgpt-prompts
* **LoRA Configuration:**
  + r = 8
  + lora\_alpha = 16
  + lora\_dropout = 0.05
  + target\_modules = ["query\_key\_value"]
* **Training parameters:**
  + epochs = 3
  + batch\_size = 4
  + output\_dir = ./results-lora-r8-alpha16

**Results**

* **Training Loss** steadily decreased:
  + Step 10: 2.75
  + Step 20: 2.72
  + Step 30: 2.72
* **Final Training Loss:** 2.7246
* **Total Trainable Parameters:** ~860K (~0.15% of the model)

Training was completed in a short amount of time with minimal GPU memory usage, and the loss showed signs of convergence within 3 epochs.

**What I Learned**

* LoRA allows you to fine-tune large models efficiently by updating only a small portion of the parameters.
* With the right lora\_config, it's possible to get meaningful results in just a few epochs.
* This setup is highly cost-effective for companies or researchers working in low-resource environments.
* Even a small model like bloom-560m can produce strong results when fine-tuned properly.

**Conclusion**

One carefully chosen configuration was enough to validate LoRA’s effectiveness. The results show that with minimal compute, training time, and cost, strong performance can be achieved. This makes LoRA a powerful strategy for scaling AI applications efficiently.