**What is fine tuning?**

Fine-tuning a large language model (LLM) involves training the pre-trained model further on a specific dataset to adapt it to a particular task or domain.

Pre-trained-model’s examples: llama3, llama3.2, mistral, gpt3.5, gpt4 etc

**Two Types of fine tuning:**

1. **Full Fine-Tuning:**

In full fine-tuning, all parameters of the model are updated using a task-specific dataset.

It requires more computation power and more expensive.

1. **PEFT (Parameter-Efficient Fine-Tuning):**

Here's how PEFT works:

1. Freeze the parameters of a pre-trained LLM.
2. Add a small number of new parameters.
3. Fine-tune the new parameters on a small training dataset.

It requires less computation power and is less expensive, faster.

**When to choose fine tuning and when to choose RAG?**

When domain specific data is static and not varying too much then fine tuning is best choice.

When Chatbot or question answering or chat with database means input data is keep on changing and new data coming then RAG is best choice.

**How LoRA Fits Into Fine-Tuning:**

When fine-tuning an LLM with LoRA, the overall procedure typically involves:

1. **Starting with a Pre-trained Model:** You start with a pre-trained model (e.g., GPT, BERT, etc.).
2. **Applying LoRA to Layers:** LoRA is applied to the attention layers (or any other layers where adaptation is needed) by introducing low-rank matrices.
3. **Training the Low-Rank Matrices:** During the fine-tuning process, only the low-rank matrices AAA and BBB are updated while keeping the original model parameters frozen.
4. **Using the Adapted Model:** After training, the adapted model is used for inference with the original pre-trained weights combined with the fine-tuned low-rank updates.

How QLORA Works?

1. **Base Model Selection:** Select pre trained llm model GPT or llama (might be stored with high precision (e.g., FP16 or FP32).)
2. **Quantization**: Before fine-tuning, QLoRA quantizes the base model’s weights. The weights are typically quantized to lower precision (e.g., INT8 or FP8). This quantization drastically reduces memory usage, making it possible to load and manipulate large models on less powerful GPUs.
3. **LORA:** The original weights of the model (e.g., the attention matrix) are kept frozen and only the low-rank matrices AAA and BBB are trained, which significantly reduces the number of parameters that need to be updated. The main advantages of LoRA are memory efficiency, faster adaptation to new tasks, and reduced computational costs.
4. **Training**:  
   During fine-tuning, only the low-rank matrices are updated, not the entire model. This makes the process much more efficient because instead of updating all the weights, only a small subset of parameters are changed, and the quantized weights help to keep memory usage low. This allows the model to be adapted to new tasks with a reduced computational burden.
5. **Inference**:  
   After fine-tuning, the model uses the quantized weights combined with the adapted low-rank matrices. During inference, the model can perform efficiently with a relatively smaller memory footprint compared to traditional fine-tuning methods.

**Differences between lora and Qlora:**

Focuses solely on low-rank updates to fine-tune large pre-trained models efficiently.

Qlora Combines **quantization** with LoRA

LORA Requires more memory than QLoRA.

QLORA requires less memory.

**Fine tuning coding Steps:**

Here’s a step-by-step summary for fine-tuning using the PEFT method and unsloth:

1. **Install Required Libraries:** Install the necessary libraries, including unsloth, transformers, datasets, and trl.
2. **Load Pre-trained Model:** Load a pre-trained language model from the unsloth repository or other sources.
3. **Prepare Dataset:** Load and preprocess your dataset using datasets.load\_dataset() or custom preprocessing methods.
4. **Set Chat Template:** Apply a relevant chat template using get\_chat\_template() to format the input correctly.
5. **Apply PEFT with LoRA:** Use FastLanguageModel.get\_peft\_model() to configure the model for PEFT (LoRA).
6. **Configure Trainer:** Set up the SFTTrainer with appropriate training arguments (e.g., batch size, learning rate, epochs).
7. **Train the Model:** Train the model by calling trainer.train() with the training data.
8. **Monitor Training:** Track training statistics and adjust hyperparameters if needed based on the output from trainer.train().
9. **Fine-Tune Responses:** Use train\_on\_responses\_only() to focus training on response generation from the model.
10. **Optimize for Inference:** Use FastLanguageModel.for\_inference() to optimize the model for faster inference after training.
11. **Evaluate and Save Model:** Evaluate the fine-tuned model and save it using .save\_pretrained() for later use.   
      
      
    #################################