LoRA and QLoRA

LoRA vs. QLoRA

LoRA (Low-Rank adaptation) and QLoRA (quantized Low-Rank adaptation) are both techniques for training AI models. More specifically, they are forms of parameter-efficient fine-tuning (PEFT), a fine-tuning technique that has gained popularity because it is more resource-efficient than other methods of training large language models (LLMs).

LoRA and QLoRA both help fine-tune LLMs more efficiently, but differ in how they manipulate the model and utilize storage to reach intended results.

How does LoRA work?

The LoRA technique uses new parameters to train the AI model on new data.

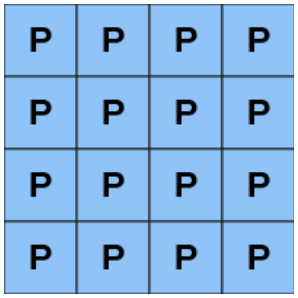
Instead of training the entire model and all of the pre-trained weights, they are set aside or "frozen" and a smaller sample size of parameters is trained instead. These sample sizes are called "low-rank" adaptation matrices, for which LoRA is named.

They are called low-rank because they are matrices with a low number of parameters and weights. Once trained, they are combined with the original parameters, and then act as one single matrix. This allows fine-tuning to be done much more efficiently.

It's easier to think of the LoRA matrix as one row or one column that is added to the matrix.

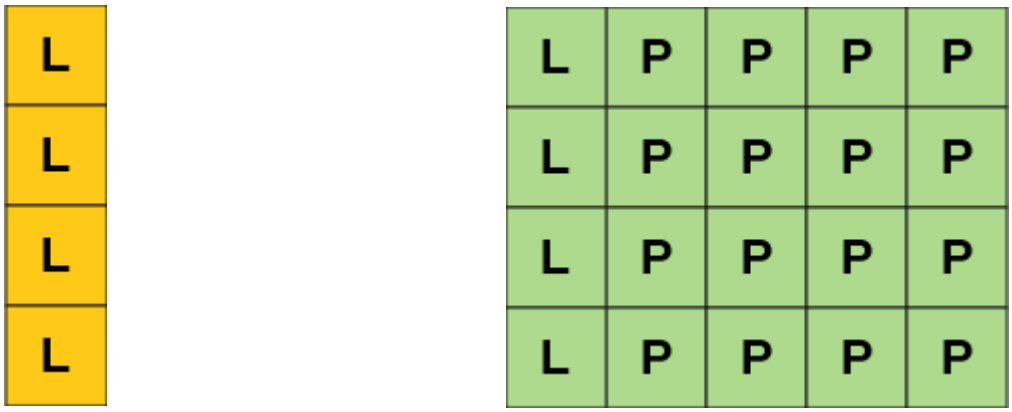
Think of this as the whole parameter that needs to be trained:

Training all of the weights in the parameter takes significant time, money, and memory. When it's done, you may still have more training to do, and wasted a lot of resources along the way.



This column represents a low-rank weight:

When the new low-rank parameters have been trained, the single “row” or “column” is added into the original matrix. This allows it to apply its new training to the whole parameter.



Now the AI model can operate together with the newly fine-tuned weights.

Training the low-rank weight takes less time, memory, and cost. Once the sample size is trained, it can apply what it’s learned within the larger matrix, without taking up any extra memory.

How does QLoRA work?

QLoRA is an extension of LoRA. It is a similar technique with an additional perk: less memory.

The “Q” in “QLoRA” stands for “quantized.” In this context, quantizing the model means compressing very complex, precise parameters (a lot of decimal numbers and a lot of memory) into a smaller, more concise parameter (less decimals and less memory).

Its goal is to fine-tune a portion of the model using the storage and memory of a single graphics processing unit (GPU). It does this using a 4-bit NormalFloat (NF4)---a new data type that is capable of quantizing the matrices with even

less memory than LoRA. By compressing the parameter into smaller, more manageable data, it can decrease the memory footprint required by up to 4 times its original size.

After the model has been quantized, it is much easier to fine-tune because of its small size.

Think of this as the original model's parameters:

Within the 12 parameters, 3 are green, 6 are blue, 2 are yellow, and 1 is pink. When the model is quantized, it is compressed into a representation of the previous model.

After quantization, we are left with a sample size of 1 green, 2 blue, and 1 yellow.

During quantization, there is a risk that some data is so small that it is lost during the compression. For example, the 1 pink parameter is missing because it was such a small fraction of the parameter, it did not represent enough data to carry over into the compressed version.

In this example, we compress the parameters from 12 to 4. But in reality, billions of parameters are being compressed into a finite number that can be manageably fine-tuned on a single GPU.

Ideally, any lost data can be recovered from the original parameter when the newly trained matrix is added back to the original matrices, without losing precision or accuracy along the way. However, this is not guaranteed.

This technique combines high-performance computing with low-maintenance memory storage. This keeps the model extremely accurate while working with limited resources.