LORA & QLORA

PEFT (Parameter Efficient Fine-Tuning):

Training large language models requires a high cost .
These models require updating every single parameter during fine-tuning and it requires lots of GPU memory .
PEFT is an approach to fine tune large pre-trained models without updating all their parameters which saves computational power, time and GPU memory.

LORA(Low Rank Adaptation):

matrix rank (r):

The rank of any matrix (r) is the number of linearly independent rows or columns .
so if we have a matrix with size 4*5 and it has 2 columns linearly independently so the r becomes 2 and the other 3 columns can be expressed as a combination of those 2 independent columns .
We can represent the same information with fewer dimensions.

We will put this two columns in a matrix (A) 4*2 and the other matrix are the weights of the combinations (B) 5*2. when we multiply this together(A*B) we will get back the original matrix 4*5,but we store it in 8+10=18 instead of 20 and for bigger dimensions we will save large numbers of parameters.

adapter:

For example BERT with Adapter:

BERT model is already pretrained without adapters on Masked Language Model (MLM) and Next Sentence Prediction (NSP), while fine tuning only adapter weights are updated during back propagation instead of all parameters of the model.
only these parameters need to be stored and everything else will be frozen.

- pros : reduce number of trainable parameters per task
- cons : increase latency / inference time, adapters added in a sequential manner and increases the time of forward path of transformer layer

We know that each multi head attention layer has weights for query, key and value with shape d*d .
We are going to attach **A**q, **B**q, **A**k, **B**k, **A**v and **B**v weighted tensors for Query, Key and Value to every single multi head attention layer where A with size d*r and B r*d

During fine tuning all A's and B's are learned and the modify matrices of weights for every layer based on **Wq = Wq + Aq * B**
In inference there are no additional weight parameters and no inference latency.

QLORA (Quantization Low Rank Adaptation):

quantization:

Quantization is the process of representing model weights using fewer bits instead of full precision (FP32 / FP16).
For example, instead of storing weights in 16-bit floating point, we store them in 4-bit, which reduces memory usage.The pretrained model weights are quantized to 4-bit and kept frozen during fine-tuning.

In the base model the original weight matrices of the transformer (**Wq, Wk, Wv**) are stored in 4-bit precision.These weights are never updated during training.

This allows loading very large models 7B for example on limited GPU memory.

low-rank adapters:

Just like LoRA we attach low-rank matrices A and B to the attention layers. A has shape d * r **,** B has shape r * d. Only A and B are trainable and are stored in FP16 / BF32

LoRA vs QLoRA

- **LoRA**:
  Base weights [FP16 and frozen]
  Adapters [FP16 and trainable]

- **QLoRA**:
  Base weights [4-bit and frozen]
  Adapters [FP16 and trainable]