Low Rank Adaptation (LoRA):-

Good Resources - https://www.philschmid.de/fine-tune-flan-t5-peft
https://lightning.ai/lightning-ai/studios/code-lora-from-scratch

It is a popular method for Parameter efficient fine tuning (PeFT).
Problems which it solves - **Lot of time and GPU memory** required in fine tuning normal models.

Paper description:-

Rank of matrix – Minimum number of rows and columns in a matrix.

Large models have an inherent 'intrinsic dimension', i.e. a lot of their values are redundant and one can get very similar results will lesser dimensions also.
During fine tuning, the weights of the pretrained LLM are **frozen**. We assume that the change in weights also has a low intrinsic dimension.

Del W = BA
They are initialized with B = 0 and A = Weights from a normal distribution.
In transformers they are applied to the attention weights.
(100, 100) = (100, 5) (5, 100) so much lesser parameters.
In the paper, they take r = 1.

Scaling factor alpha – To stabilise hyperparameter changes, i.e., if we change r slightly it doesn't affect the model much.

LoRA doesn't impose a significant inference penalty on us.

Some other PeFT techniques – Fine tune last layer, Adapter Layer, Prefix Tuning.

Efficient Adaptation of Quantized LLMs (QLoRA):-

https://www.youtube.com/watch?v=C33SwN3Ynp4
https://dassum.medium.com/fine-tune-large-language-model-llm-on-a-custom-dataset-with-qlora-fb60abdeba07

BitsAndBytes Library for implementation.

The paper introduces 3 main techniques:-

1. **4-bit Normal Float**
2. **Double Quantization**
3. **Paged Optimizers**

Float32 (4 bytes) -> Int4 (0.5 bytes) on quantization. Allows us to finetune models on smaller GPUs.
To convert Float32 to Int8 we first calculate the quantization constant c:-

C = (127 / absmax (Vector X)) * Xi and then we round these to get our values in the required range.
We can dequant the values by dividing each value with c to get back the original values. There is some loss of precision. When we have values over a huge range then it is possible that 2 values map to the same int which we don't want as there is a lot of information loss.
Normal quantization has issues with outliers.

Pretrained Neural Nets – Usually have a zero centred normal distribution of weights. We can break this into equal buckets. Same number of values in each bucket.

Double Quantization – Instead of having one global c value, if we need one for each bucket, we can use it. The c values can be quantized from Float32 to Float8 and get another quantization constant for this quantization.
Then it is double dequantized also.

Paged Optimisers – Prevent memory spikes when we really get a really long input, thus, preventing crashing of sessions. State of the optimiser is moved from the GPU memory to the CPU memory while the long sequence is being read, then it goes back to the GPU memory.