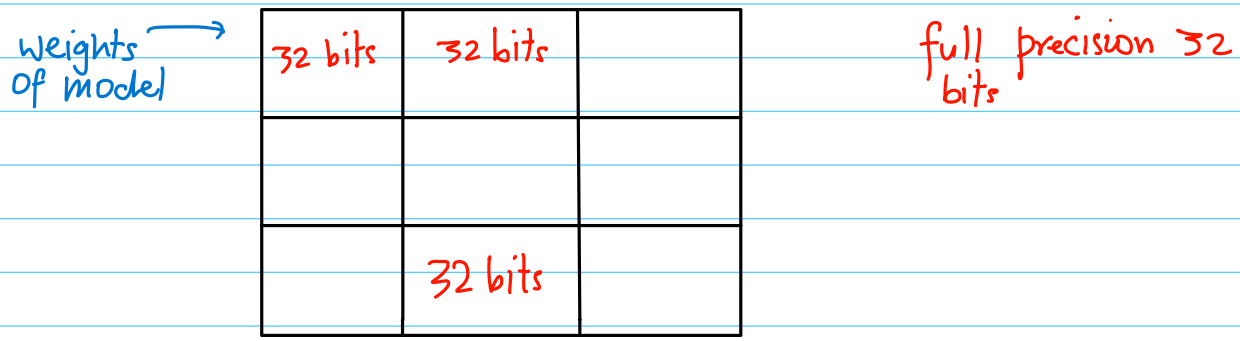
Quantization : Conversion from higher memory format to lower memory format.



We can convert this 32 bits into 8 bits.

int 32 → int 8 > Help w easier inference

How to perform Quantization

① Symmetric Quantization ② Asymmetric Quantization

(all the data is evenly distributed)

Symmetric Uint8 :

[0 ... 1000] → [0, ... 255]

x_{min} x_{max} q_{min} q_{max}

Min/ Max Scalar

0 1000

↓ ↓

0 255

$$Scale = \frac{X_{max} - X_{min}}{q_{max} - q_{min}} = 3.92$$

convert 250 → 8 bit

$\Rightarrow \left(\frac{250}{3.92}\right) \Rightarrow 64$

Asymmetric Quantization (non uniform distribution of data)

[-20.0 . . . . . 1000.0]

↓

[0 . . . . . 255]

$\text{Scale} = \frac{1000+20}{255} = 4.0$ Scale factor

Conversion :

$\frac{-20}{4} = -5.0 + \boxed{5.0}$

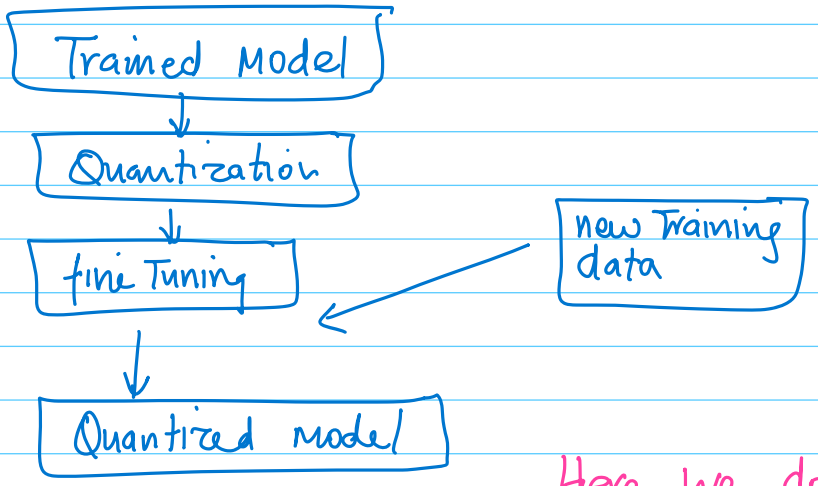
add same # in +ve sign (zero factor)

① Post Training Quantization (we don't use this technique)

Pretrained model → Calibration → Quantized Model

Weights data

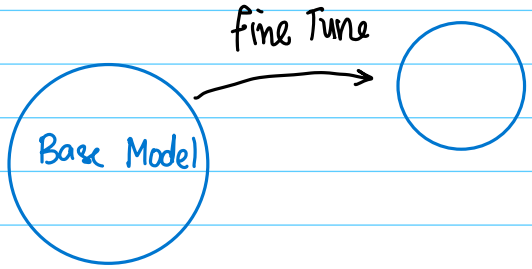
② Quantization Aware training (QAT) (This is what we use)



Here we don't lose any accuracy after this

LoRA, QLoRA intuition

Low Rank Adaptation of LLMS



① full Parameter fine tuning : Training all weights

② Domain fine tuning

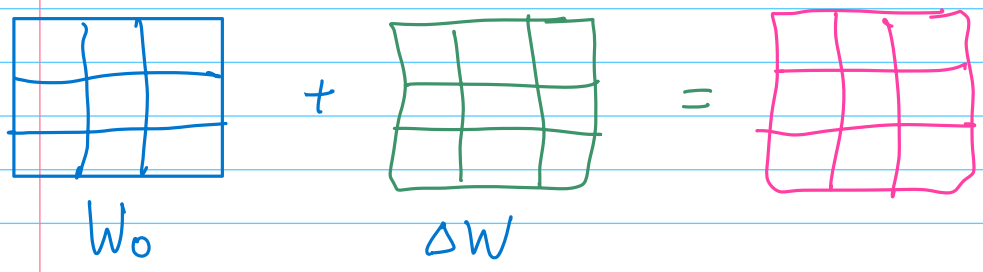
③ Specific task fine tuning

What does LoRA do?

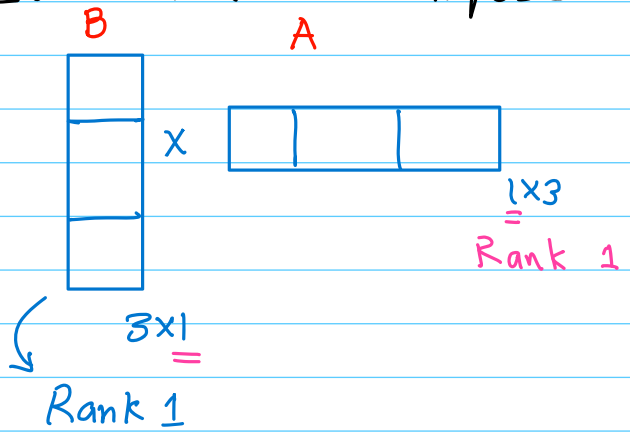
Decomposed matrices

$W_0 + \Delta W = W_0 + BA$

fine tuned weights



This ΔW matrix is decomposed like:



with Increasing R ↑ the parameters ↑ and we use higher ranks when model wants to learn complex things.

QLoRA → Quantized LoRA. ⇒ 16bit ⇅ 4bit