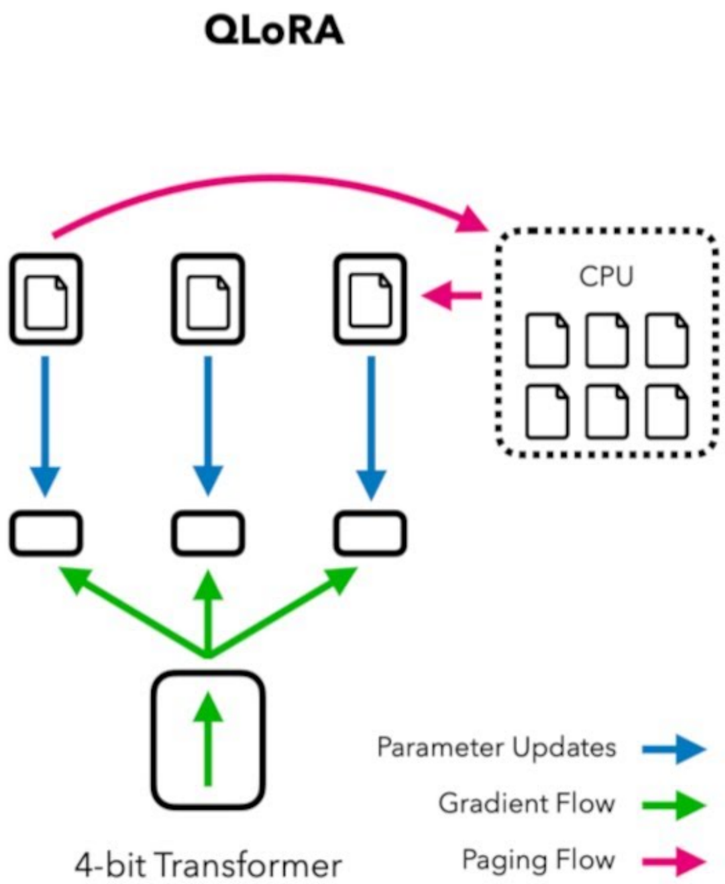


QLoRA

Efficient Finetuning of Quantized LLMs

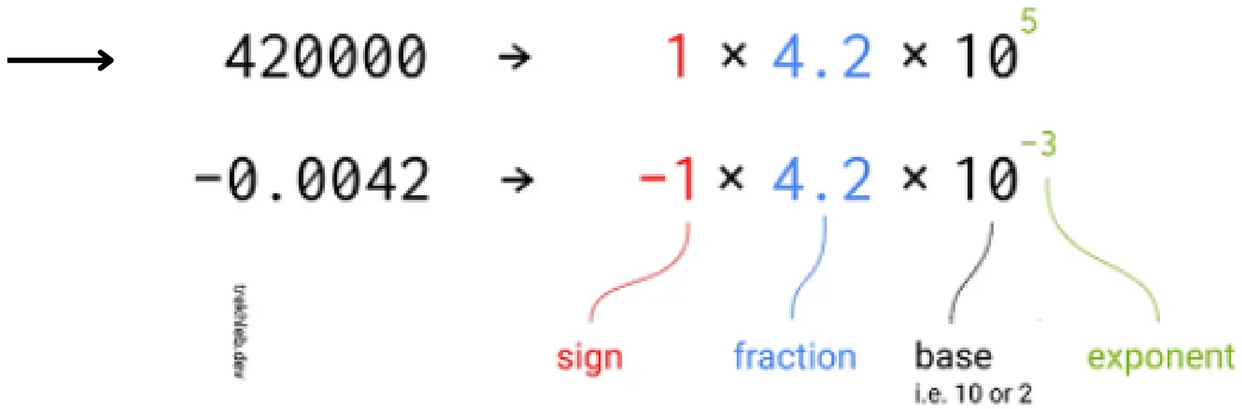
Key Idea

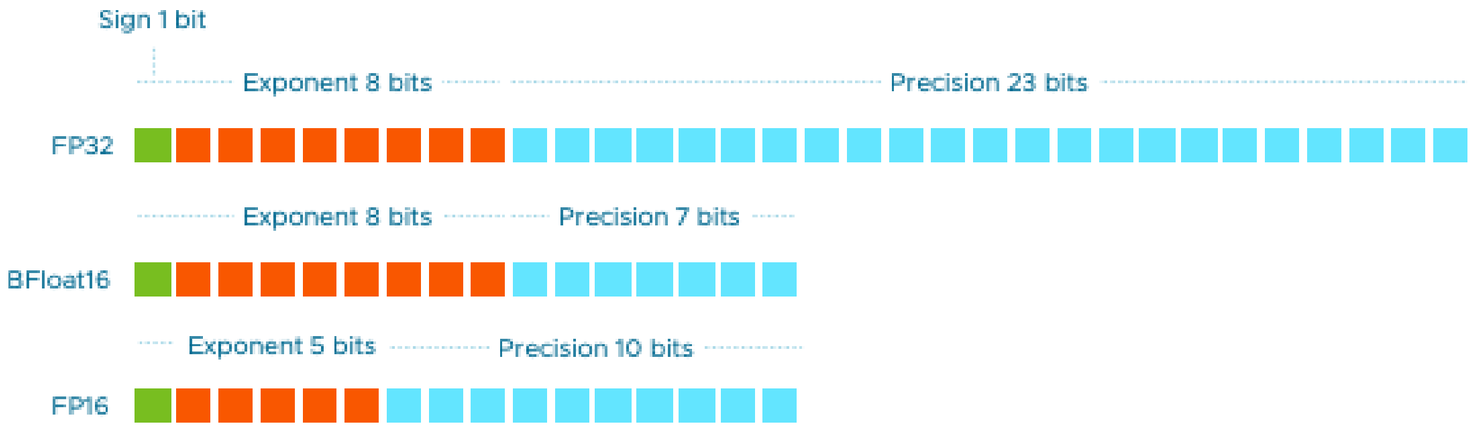
- Approach: Adding quantization on LoRA
- Key word on Solution:
  - 4-bit NormalFloat Quantization
  - Double Quantization
  - Paged Optimizers

Quantization

Datatype - Dtype

Recall: How's number save in computer?





Quantization

2 main steps:

- Normalize X into the rage [-1.0, 1.0] by devide by ansmax(X)
- Find the closest value in the datatype
  (rounding for integers; in general binary search)

$$\mathbf{X}^{\text{In 8}} = \text{round}\left(\frac{127}{\text{absmax}\left(\mathbf{X}^{\text{FP32}}\right)}\mathbf{X}^{\text{FP32}}\right) = \text{round}\left(c^{\text{FP32}} \cdot \mathbf{X}^{\text{FP32}}\right),$$

$\mathbf{X}^{\text{FP32}}$ is the original input tensor in FP32

$\frac{127}{\text{absmax}\left(\mathbf{X}^{\text{FP32}}\right)} = c^{\text{FP32}}$ the **quantization constant**

Quantization

Map: {Index: 0, 1, 2, 3 -> Values: -1.0, 0.3, 0.5, 1.0}

Input tensor: [10, -3, 5, 4]

1. Normalize with absmax: [10, -3, 5, 4] -> [1, -0.3, 0.5, 0.4]
2. Find closest value: [1, -0.3, 0.5, 0.4] -> [1.0, 0.3, 0.5, 0.5]
3. Find the associated index: [1.0, 0.3, 0.5, 0.5] -> [3, 1, 2, 2] -> store
4. Dequantization: load -> [3, 1, 2, 2] -> lookup -> [1.0, 0.3, 0.5, 0.5] -> denormalize -> [10, 3, 5, 5]

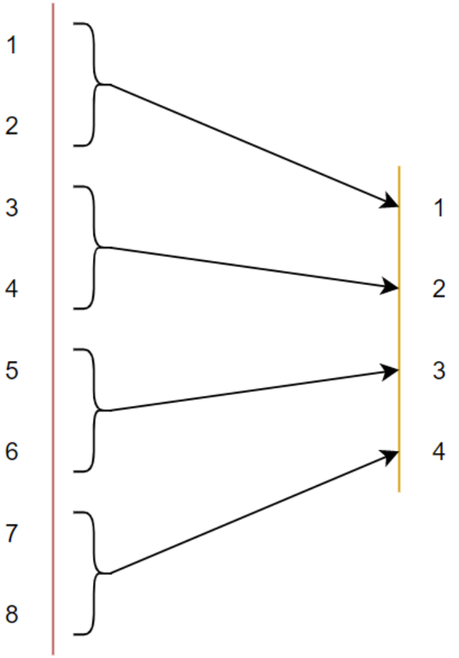
Dequantization

To perform dequantization, we can use the following equation:

$$\text{dequant}(c^{\text{FP32}}, \mathbf{X}^{\text{In18}}) = \frac{\mathbf{X}^{\text{In18}}}{c^{\text{FP32}}} = \mathbf{X}^{\text{FP32}}$$

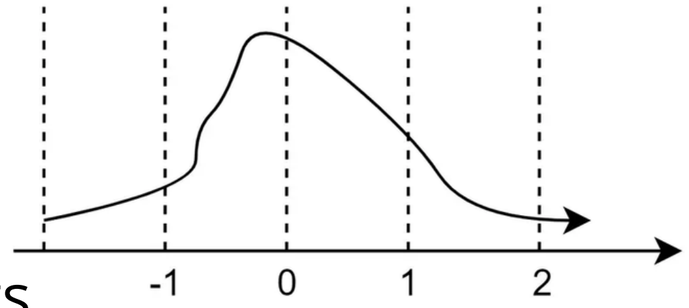
```
def dequantize(c_FP32, X_Int8):
    X_FP32_dequant = X_Int8 / c_FP32
    return X_FP32_dequant
```

Problems in Quantization

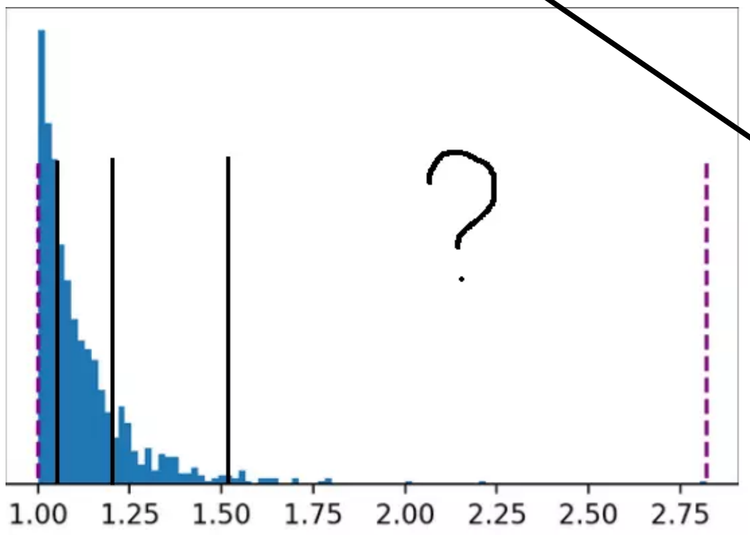


Information is lost due to quantization

The distribution of the weight tensor is not emphasized



Outliers appear in tensor weights

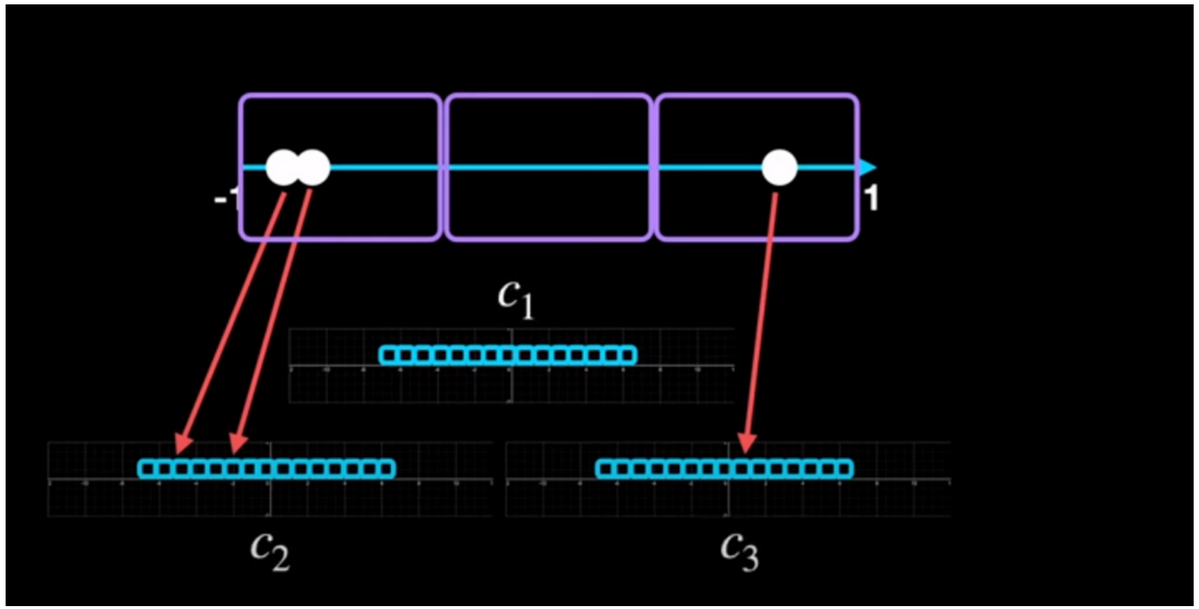


Problems in Quantization

Information is lost due to quantization + Outliers appear in tensor weights

⟶ Using Block-wise Quantize (or Chunk Quantize)

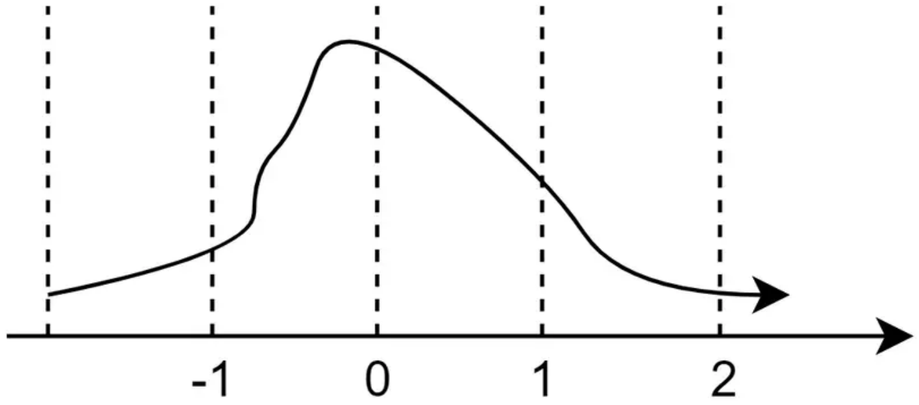
Idea: Divide the tensor into several chunks, and quantize each chunk of that tensor separately



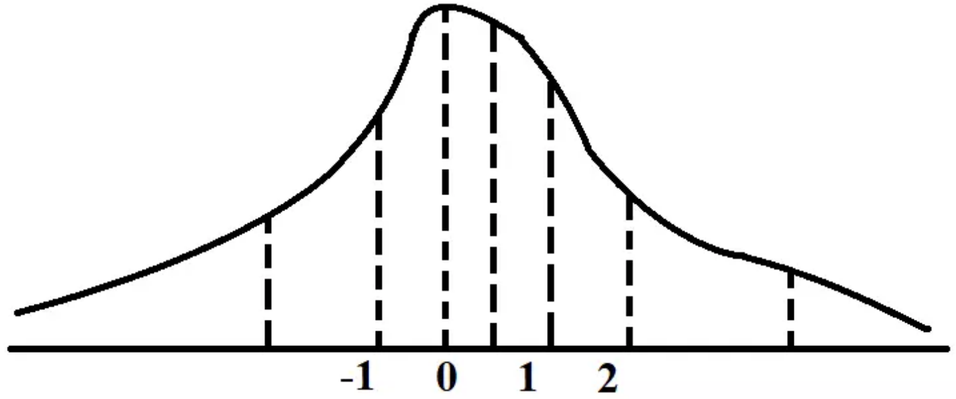
Problems in Quantization

The distribution of the weight tensor is not emphasized

⟶ Using Quantile Quantization ⟶ Datatype: NF4



Without quantil Quantization



With quantil Quantization

Normal Float 4

Since pretrained neural network weights usually have a zero-centered normal distribution with standard deviation σ (see Appendix F), we can transform all weights to a single fixed distribution by scaling σ such that the distribution fits exactly into the range of our data type. For our data type, we set the arbitrary range $[-1, 1]$. As such, both the quantiles for the data type and the neural network weights need to be normalized into this range.

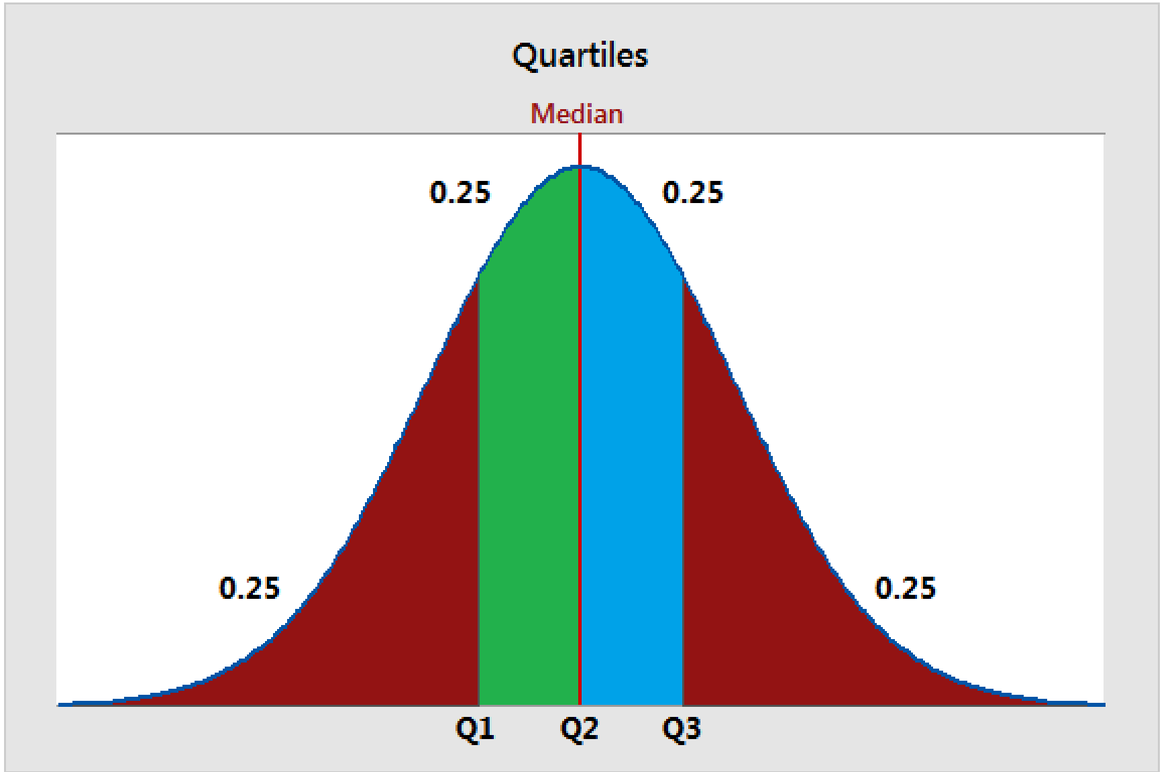
⟶ **Target:** QLoRA converts all weights to a fixed distribution

Once the weight range and data type range match, we can quantize as usual. Step (3) is equivalent to rescaling the standard deviation of the weight tensor to match the standard deviation of the k-bit data type. More formally, we estimate the 2^k values q_i of the data type as follows:

$$q_i = \frac{1}{2}\left(Q_X\left(\frac{i}{2^k+1}\right) + Q_X\left(\frac{i+1}{2^k+1}\right)\right), \tag{4}$$

where $Q_X(\cdot)$ is the quantile function of the standard normal distribution $N(0, 1)$. A problem for a symmetric k-bit quantization is that this approach does not have an exact representation of zero, which is an important property to quantize padding and other zero-valued elements with no error. To

Normal Float 4



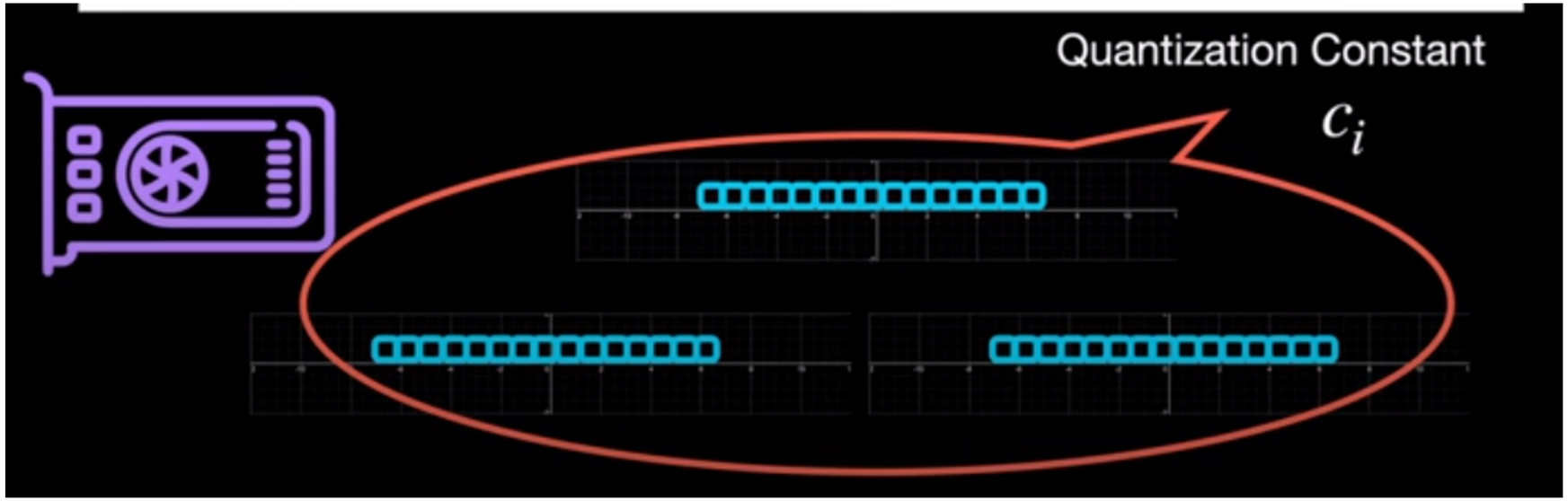
- Use 4 bits of performance
- Ranges from [−1,1]
- Asymmetry, there is a representation of the value 0
- Created to apply to tensors that conform to the mean 0 and std 1 standard distribution

Double Quantization

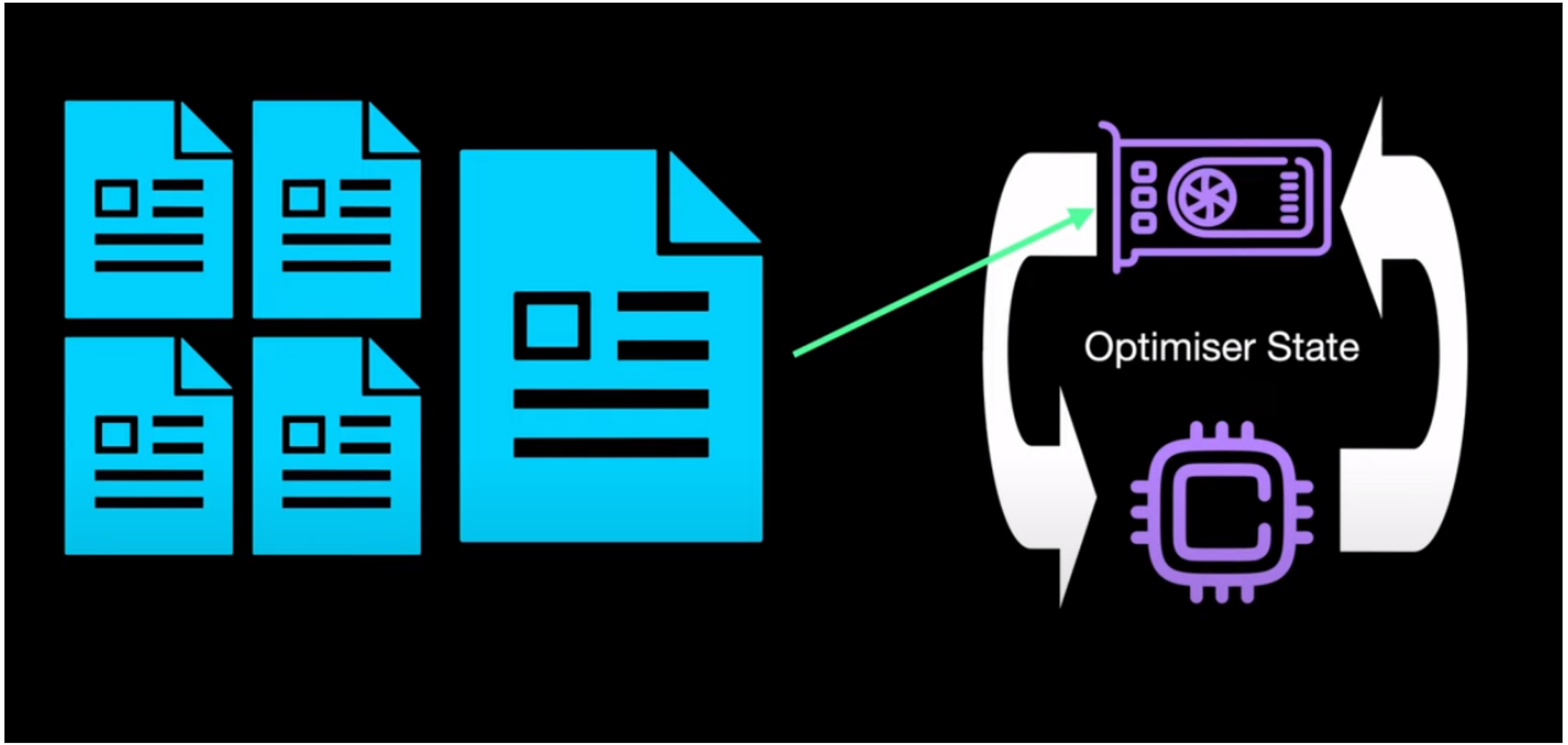
Double Quantization We introduce *Double Quantization* (DQ), the process of quantizing the quantization constants for additional memory savings. While a small blocksize is required for precise 4-bit quantization [13], it also has a considerable memory overhead. For example, using 32-bit constants and a blocksize of 64 for $\mathbf{W}$, quantization constants add $32/64 = 0.5$ bits per parameter on average. Double Quantization helps reduce the memory footprint of quantization constants.

More specifically, Double Quantization treats quantization constants c_2^{FP32} of the first quantization as inputs to a second quantization. This second step yields the quantized quantization constants c_2^{FP8} and the second level of quantization constants c_1^{FP32}. We use 8-bit Floats with a blocksize of 256 for the second quantization as no performance degradation is observed for 8-bit quantization, in line with results from Dettmers and Zettlemoyer [13]. Since the c_2^{FP32} are positive, we subtract the mean from c_2 before quantization to center the values around zero and make use of symmetric quantization. On average, for a blocksize of 64, this quantization reduces the memory footprint per parameter from $32/64 = 0.5$ bits, to $8/64 + 32/(64 \cdot 256) = 0.127$ bits, a reduction of 0.373 bits per parameter.

Double Quantization



Paged Optimizers



QLoRA

QLoRA. Using the components described above, we define QLoRA for a single linear layer in the quantized base model with a single LoRA adapter as follows:

$$\mathbf{Y}^{\text{BF16}} = \mathbf{X}^{\text{BF16}}\text{doubleDequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}, \mathbf{W}^{\text{NF4}}) + \mathbf{X}^{\text{BF16}}\mathbf{L}_1^{\text{BF16}}\mathbf{L}_2^{\text{BF16}}, \tag{5}$$

where $\text{doubleDequant}(\cdot)$ is defined as:

$$\text{doubleDequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}, \mathbf{W}^{\text{k-bit}}) = \text{dequant}(\text{dequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}), \mathbf{W}^{\text{4bit}}) = \mathbf{W}^{\text{BF16}}, \tag{6}$$

We use NF4 for $\mathbf{W}$ and FP8 for c_2. We use a blocksize of 64 for $\mathbf{W}$ for higher quantization precision and a blocksize of 256 for c_2 to conserve memory.

Reference

- https://www.youtube.com/watch?v=y9PHWGOa8HA
- https://brev.dev/blog/how-qlora-works
- In-depth guide to fine-tuning LLMs with LoRA and QLoRA
- [Vinh danh Paper] QLoRA: Quantize để training mô hình hàng tỷ tham số trên Google Colab
- QLoRA paper explained (Efficient Finetuning of Quantized LLMs)

Thanks for watching!