

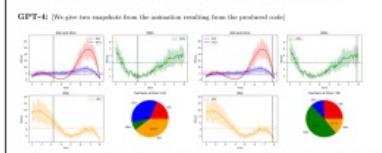
LLM

# Quantization

Xin Yu 2023.11.01

# GPT-4

## Write python codes



## Compose music

## Write poems

**Prompt:** Can you write a proof that there are many primes, with every line that rhymes?

CPT

Yes, I think I can,  
Though it might take a clever plan.  
I'll start by noting Euclid's proof,  
Which shows that primes aren't just aloof.  
Assume we have a finite list,  
Of primes, and that some have been missed.  
Multiply them all together,  
And add one, just to be clever.  
The result will be a number,  
That has no prime factors, oh wonder!  
But every number has a prime divisor,  
So it divides us; it's just a matter of time.  
So we've found a contradiction,  
And our finite list needs eviction.  
There must be infinitely many primes,  
In fact, there are so many, we can't even count them!

## Do math problems

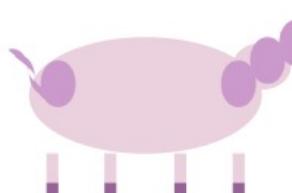
**Prompt:** Suppose  $g(x) = f^{-1}(x)$ ,  $g(0) = 5$ ,  $g(4) = 7$ ,  $g(3) = 2$ ,  $g(7) = 9$ ,  $g(9) = 6$  what is  $f(f(f(6)))$

**GPT-4:** There are different ways to approach this problem, but one possible method is to use the fact that  $g(x) = f^{-1}(x)$  implies that  $f(g(x)) = x$  for any  $x$  in the domain of  $g$ . That is, applying  $f$  to both sides of the equation  $g(x) = f^{-1}(x)$  gives  $f(g(x)) = f(f^{-1}(x)) = x$ . This means that we can undo the effect of  $f$  by applying  $g$ , and vice versa. Using this idea, we can try to find  $f(6)$  by looking for the value of  $x$  such that  $g(x) = 6$ . From the given information, we see that  $g(9) = 6$ , so  $f(6) = f(g(9)) = 9$ . Similarly, we can find  $f(9)$  by looking for the value of  $x$  such that  $g(x) = 9$ . We see that  $g(7) = 9$ , so  $f(g(7)) = f(7)$ . Finally, we can find  $f(7)$  by looking for the value of  $x$  such that  $g(x) = 7$ . We see that  $g(4) = 7$ , so  $f(7) = f(g(4)) = 4$ . Therefore,  $f(f(16)) = f(f(9)) = f(7) = 4$ .

# Write LateX codes

**Prompt:** Draw a unicorn in TiKZ

**GPT-4:** [Produces L<sup>A</sup>T<sub>E</sub>X compiling following picture.]



# Explain jokes

Example of GPT-4 visual insp

User: What is known about this issue? Describe it using key words.



Source: <https://www.reddit.com/r/bmmf/comments/oibabbv/bm>

GPT-4 The image shows a package for a "Lightning Cable" adapter with three panels.

**Panel 1:** A smartphone with a VGA connector (a large, blue, 15-pin connector typically used for computer monitors) plugged into its charging port.

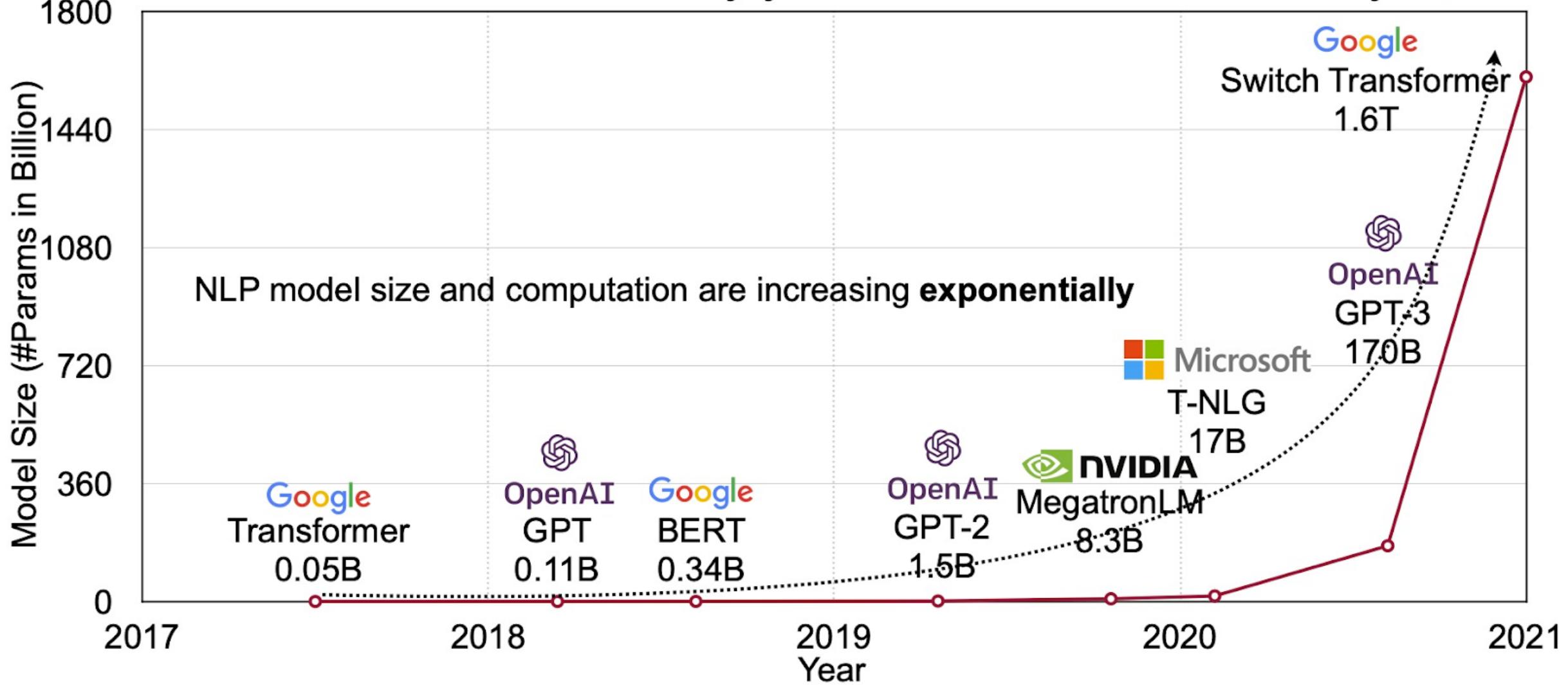
Panel 2: The package for the "Lightning Cable" adapter with a picture of a VGA connector on it.

Panel 3: A close-up of the VGA connector with a small Lightning con-

The humor in this image comes from the absurdity of plugging a large, outdated VGA connector into a small, modern smartphone charging port.

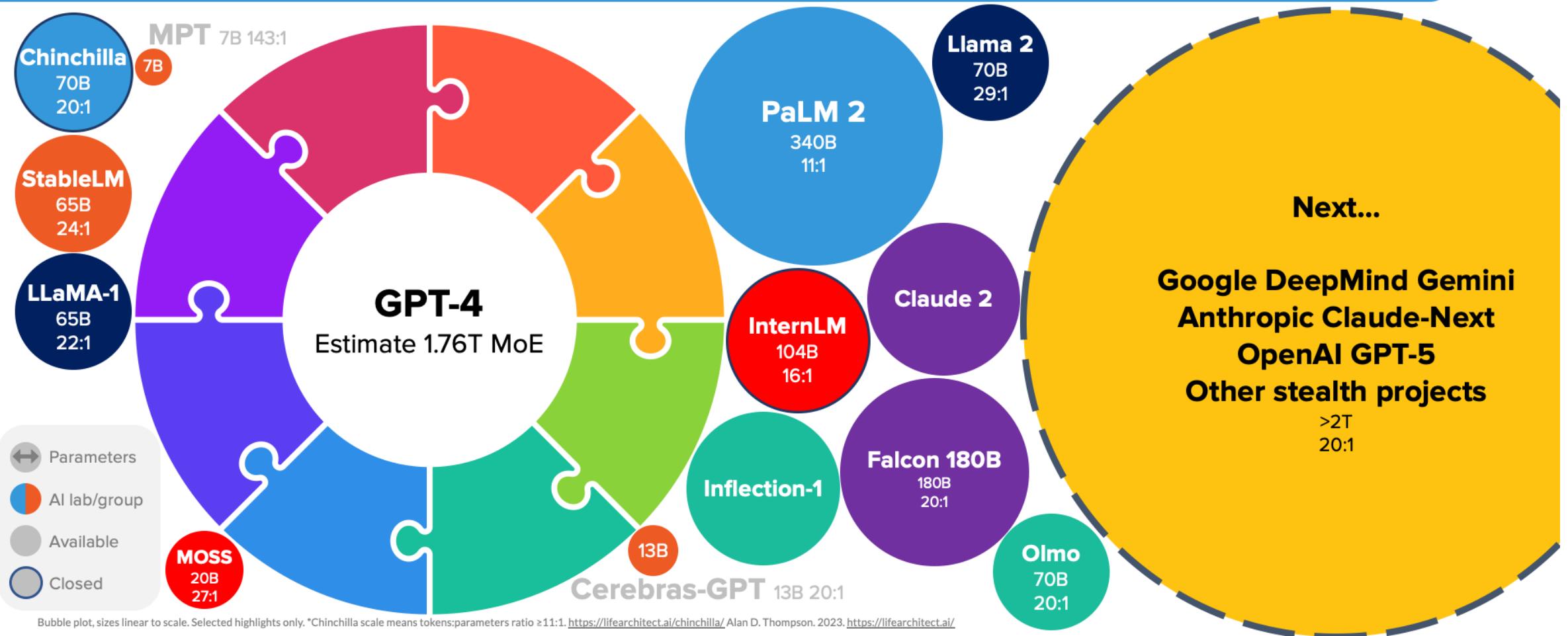
**Table 3.** Example prompt demonstrating GPT-4's visual input capability. The prompt consists of a question about an image with multiple panels which GPT-4 is able to answer.

# NLP's Moore's Law: Every year model size increases by 10x

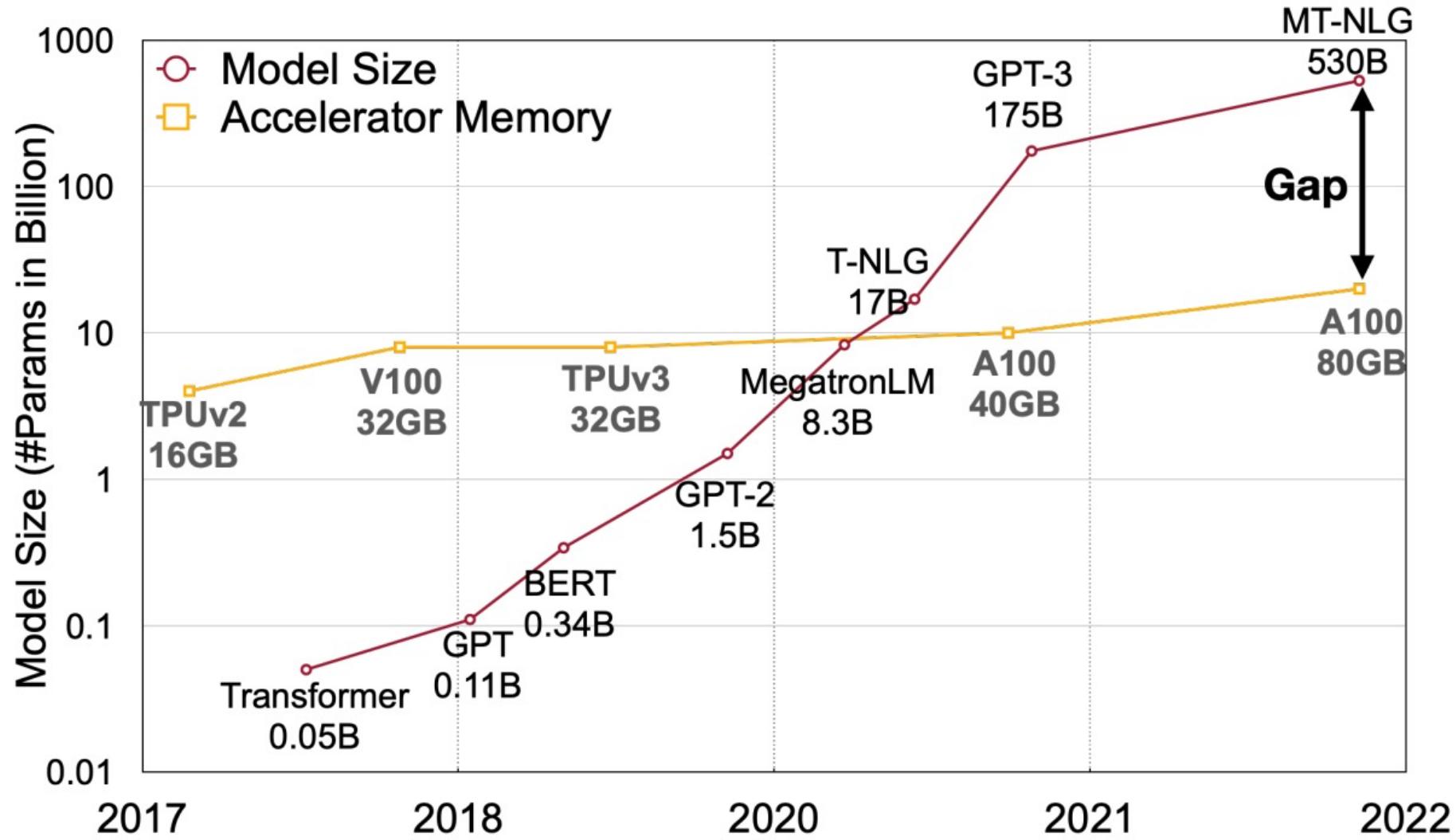


# 2023-2024 OPTIMAL LANGUAGE MODELS

SEP/  
2023

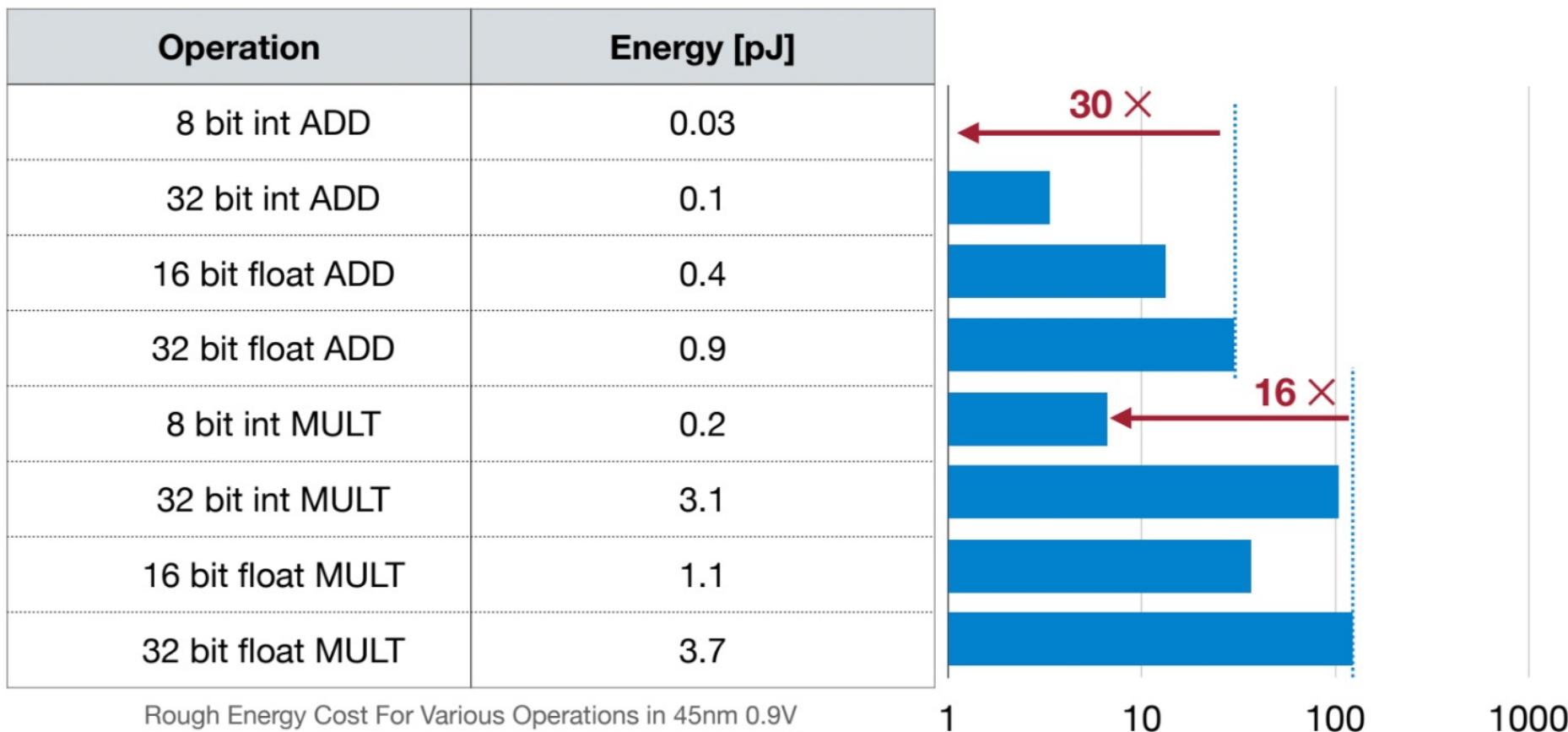


LifeArchitect.ai/models



# Low Bit-Width Operations are Cheap

## **Less Bit-Width → Less Energy**



## Computing's Energy Problem (and What We Can Do About it) [Horowitz, M.. IEEE ISSCC 2014]

This image is in the public domain

# Floating-Point Number

Exponent Width → Range; Fraction Width → Precision

## IEEE 754 Single Precision 32-bit Float (IEEE FP32)



Exponent  
(bits)

8

Fraction  
(bits)

23

Total  
(bits)

32

## IEEE 754 Half Precision 16-bit Float (IEEE FP16)



5

10

16

## Nvidia FP8 (E4M3)



4

3

8

\* FP8 E4M3 does not have INF, and S.1111.111<sub>2</sub> is used for NaN.

\* Largest FP8 E4M3 normal value is S.1111.110<sub>2</sub>=448.

## Nvidia FP8 (E5M2) for gradient in the backward



5

2

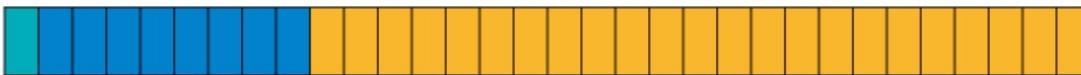
8

\* FP8 E5M2 have INF (S.11111.00<sub>2</sub>) and NaN (S.11111.XX<sub>2</sub>).

\* Largest FP8 E5M2 normal value is S.11110.11<sub>2</sub>=57344.

# Floating-Point Number

Example: 32-bit floating-point number in IEEE 754



Sign 8 bit Exponent

23 bit Fraction

Exponent	Fraction=0	Fraction≠0	Equation
00 <sub>H</sub> = 0	±0	subnormal	(-1) <sup>sign</sup> × <b>Fraction</b> × 2 <sup>-127</sup>
01 <sub>H</sub> ... FE <sub>H</sub> = 1 ... 254		normal	(-1) <sup>sign</sup> × (1 + <b>Fraction</b> ) × 2 <sup>Exponent-127</sup>
FF <sub>H</sub> = 255	±INF	NaN	



	Dynamic Range	Min Positive Value
FP32	$-3.4 \times 10^{38} \sim +3.4 \times 10^{38}$	$1.4 \times 10^{-45}$
FP16	$-65504 \sim +65504$	$5.96 \times 10^{-8}$
INT8	$-128 \sim +127$	1

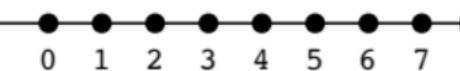
# INT4 and FP4

Exponent Width → Range; Fraction Width → Precision

## INT4

S			
0	0	0	1
0	1	1	1

-1, -2, -3, -4, -5, -6, -7, -8  
0, 1, 2, 3, 4, 5, 6, 7

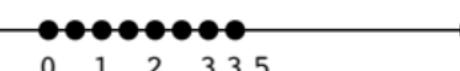


-1, -2, -3, -4, -5, -6, -7, -8  
0, 1, 2, 3, 4, 5, 6, 7

## FP4 (E1M2)

S	E	M	M
0	0	0	1
0	1	1	1

-0, -0.5, -1, -1.5, -2, -2.5, -3, -3.5  
0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5



-0, -1, -2, -3, -4, -5, -6, -7 × 0.5  
0, 1, 2, 3, 4, 5, 6, 7

$$=1$$

$$=7$$

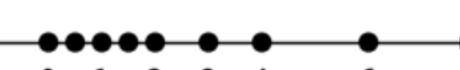
$$=0.25 \times 2^{1-0} = 0.5$$

$$=(1+0.75) \times 2^{1-0} = 3.5$$

## FP4 (E2M1)

S	E	E	M
0	0	0	1
0	1	1	1

-0, -0.5, -1, -1.5, -2, -3, -4, -6  
0, 0.5, 1, 1.5, 2, 3, 4, 6



-0, -1, -2, -3, -4, -6, -8, -12 × 0.5  
0, 1, 2, 3, 4, 6, 8, 12

$$=0.5 \times 2^{1-1} = 0.5$$

$$=(1+0.5) \times 2^{3-1} = 1$$

no inf, no NaN

## FP4 (E3M0)

S	E	E	E
0	0	0	1
0	1	1	1

-0, -0.25, -0.5, -1, -2, -4, -8, -16  
0, 0.25, 0.5, 1, 2, 4, 8, 16



-0, -1, -2, -4, -8, -16, -32, -64 × 0.25  
0, 1, 2, 4, 8, 16, 32, 64

$$=(1+0) \times 2^{1-3} = 0.25$$

$$=(1+0) \times 2^{7-3} = 16$$

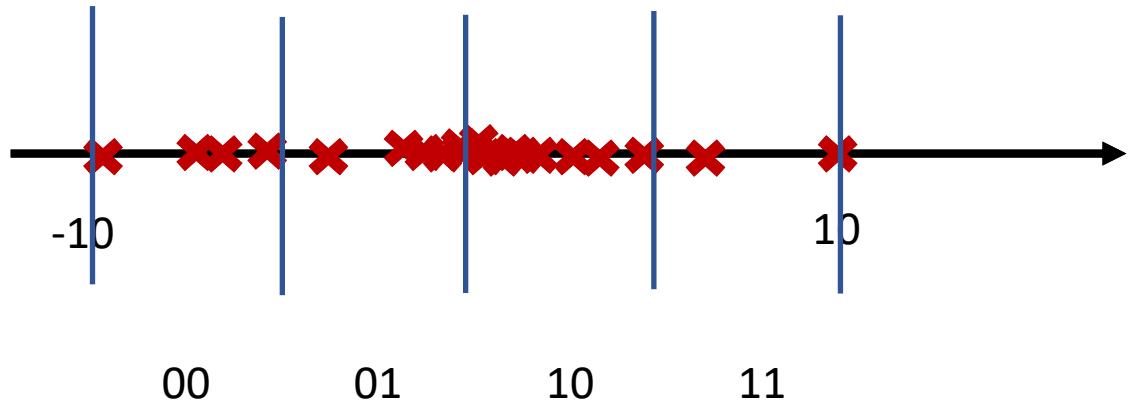
no inf, no NaN

# Quantization: Data with less bits



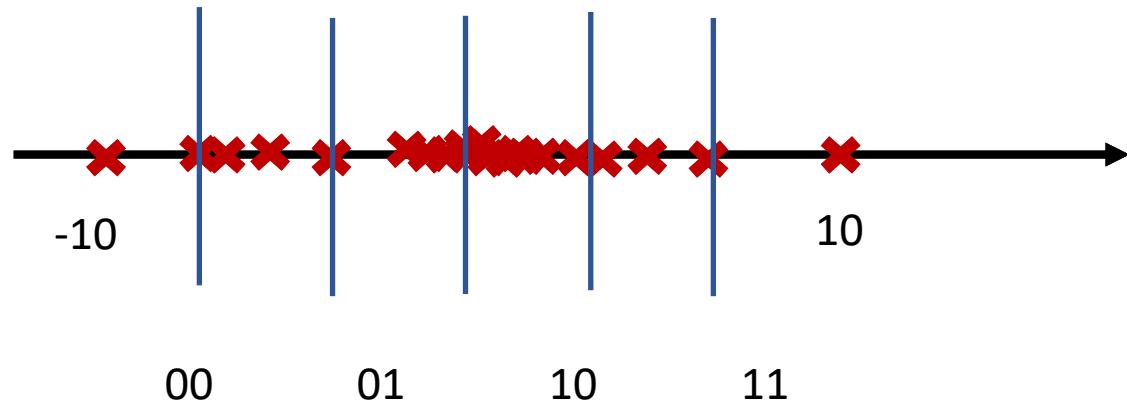
# Quantization: Data with less bits

RTN: round to nearest



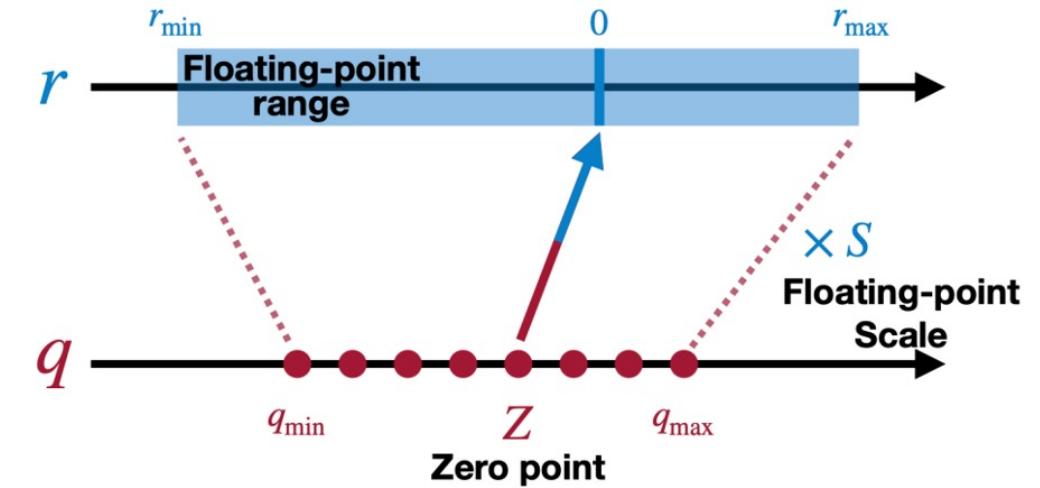
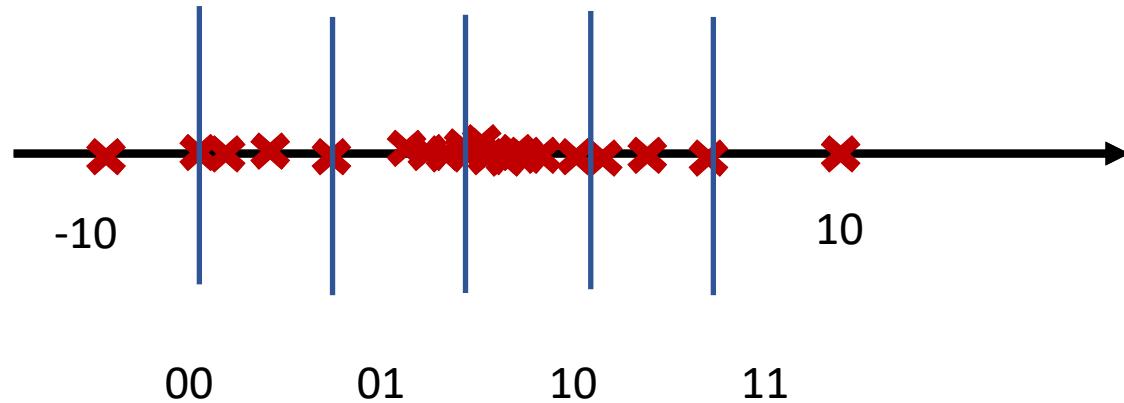
# Quantization: Data with less bits

RTN: round to nearest



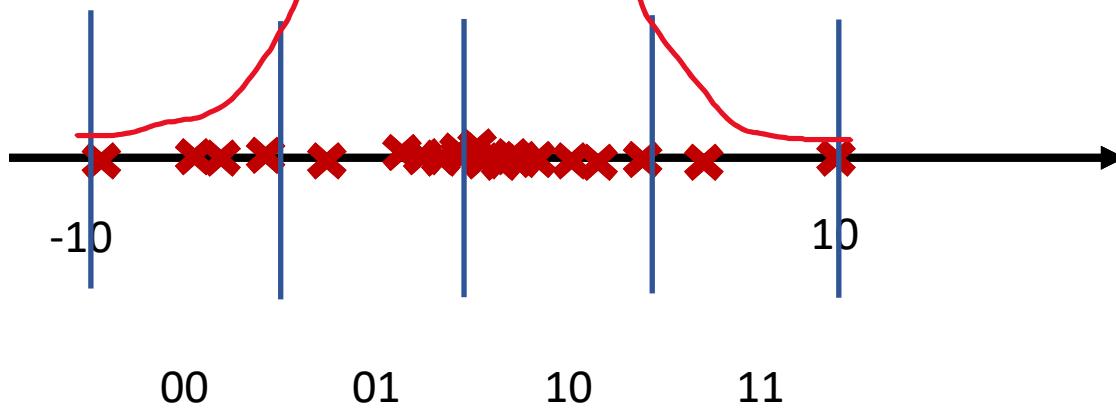
# Quantization: Data with less bits

RTN: round to nearest



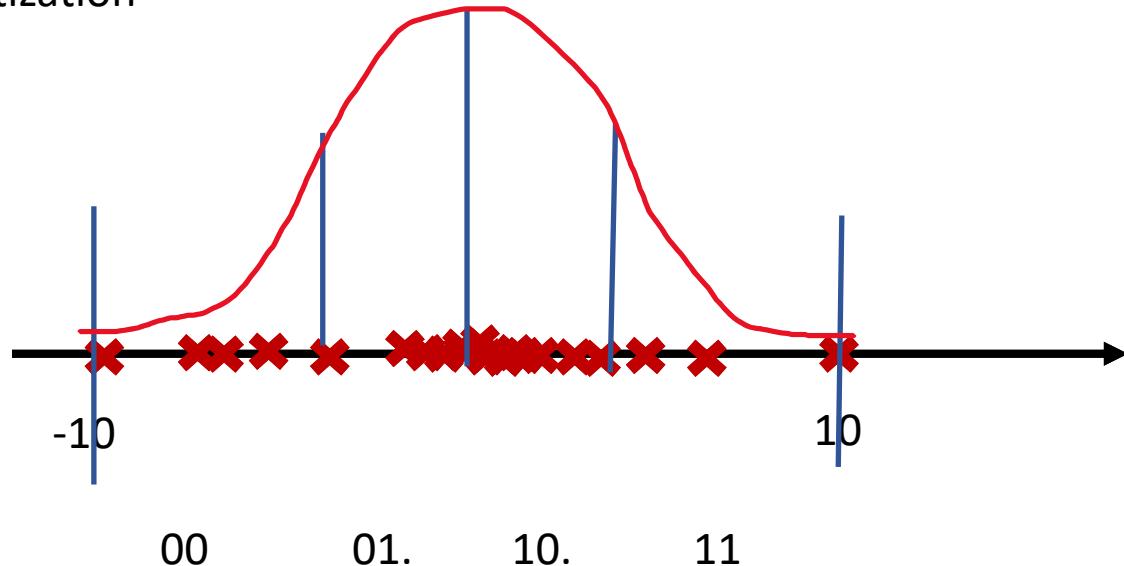
# Quantization: Data with less bits

Normal Float Quantization



# Quantization: Data with less bits

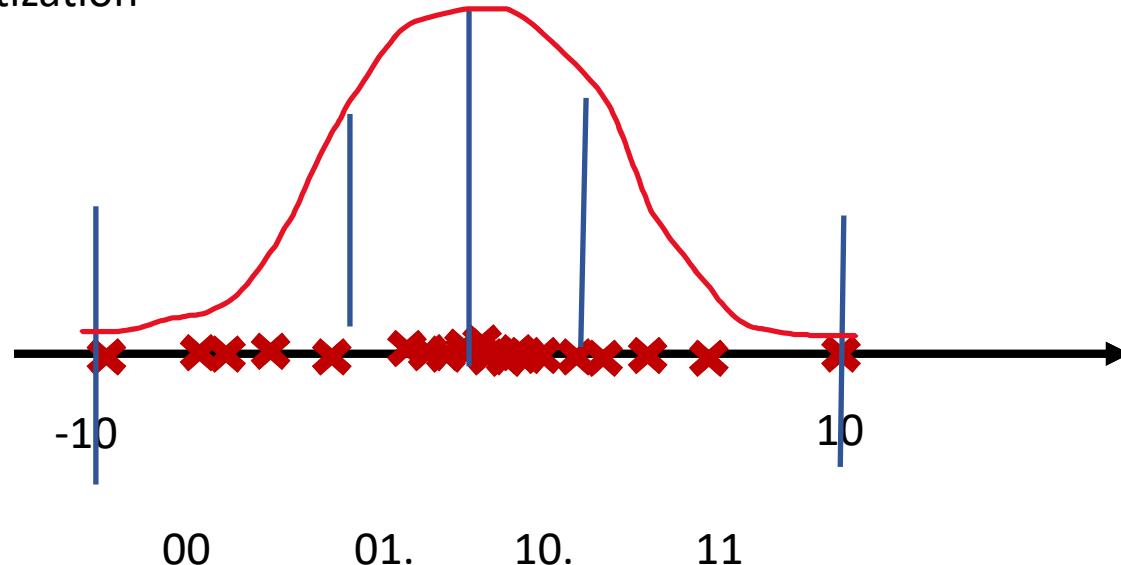
Normal Float Quantization



Quantile Quantization

# Quantization: Data with less bits

Normal Float Quantization



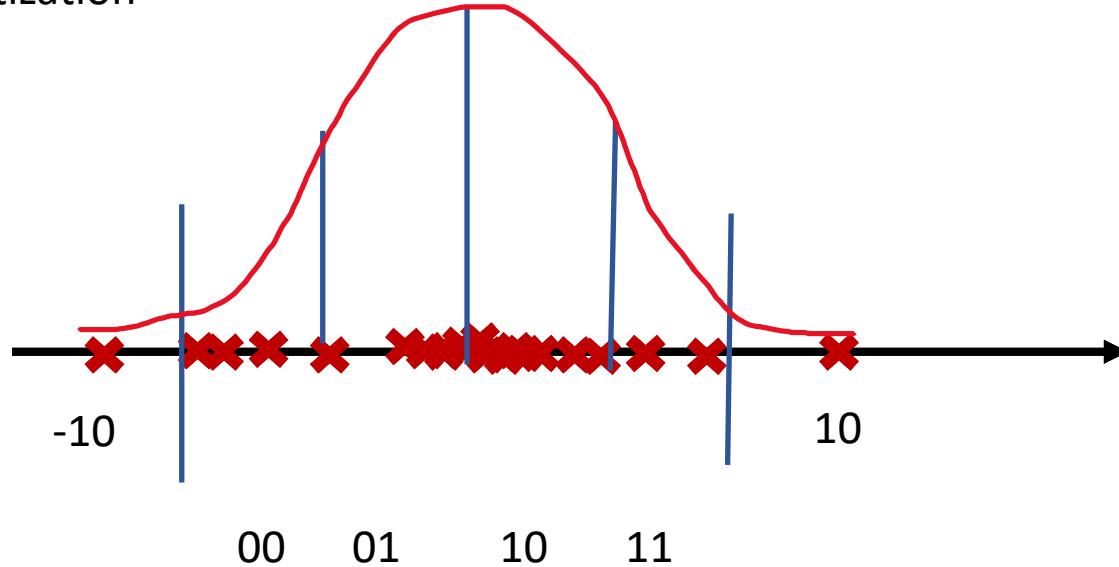
Quantile Quantization

The exact values of the NF4 data type are as follows:

```
[-1.0, -0.6961928009986877, -0.5250730514526367,  
-0.39491748809814453, -0.28444138169288635, -0.18477343022823334,  
-0.09105003625154495, 0.0, 0.07958029955625534, 0.16093020141124725,  
0.24611230194568634, 0.33791524171829224, 0.44070982933044434,  
0.5626170039176941, 0.7229568362236023, 1.0]
```

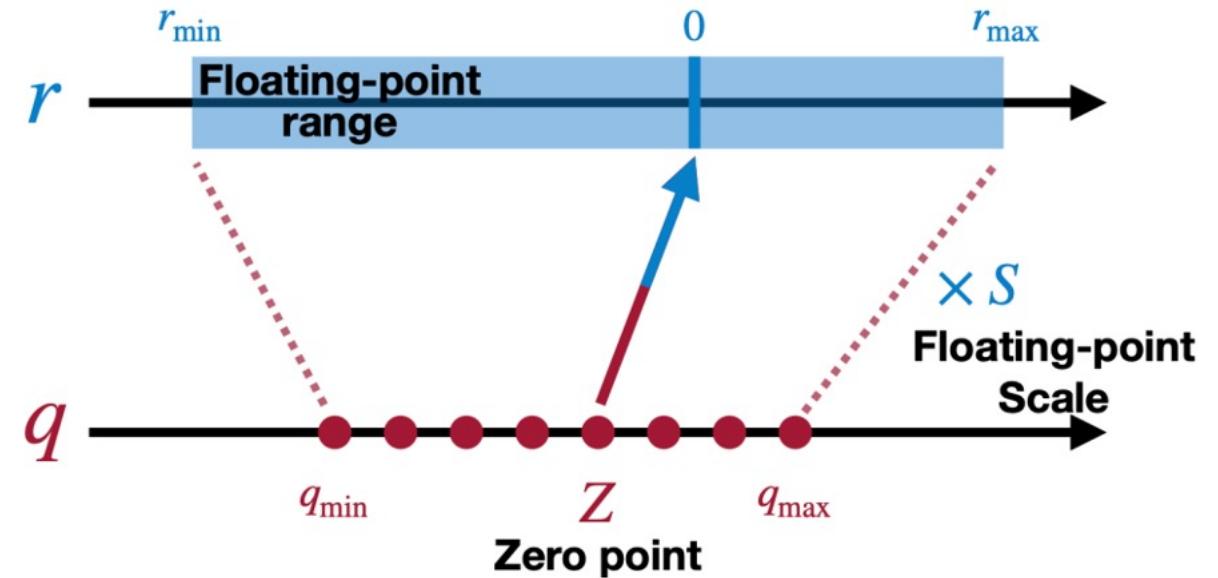
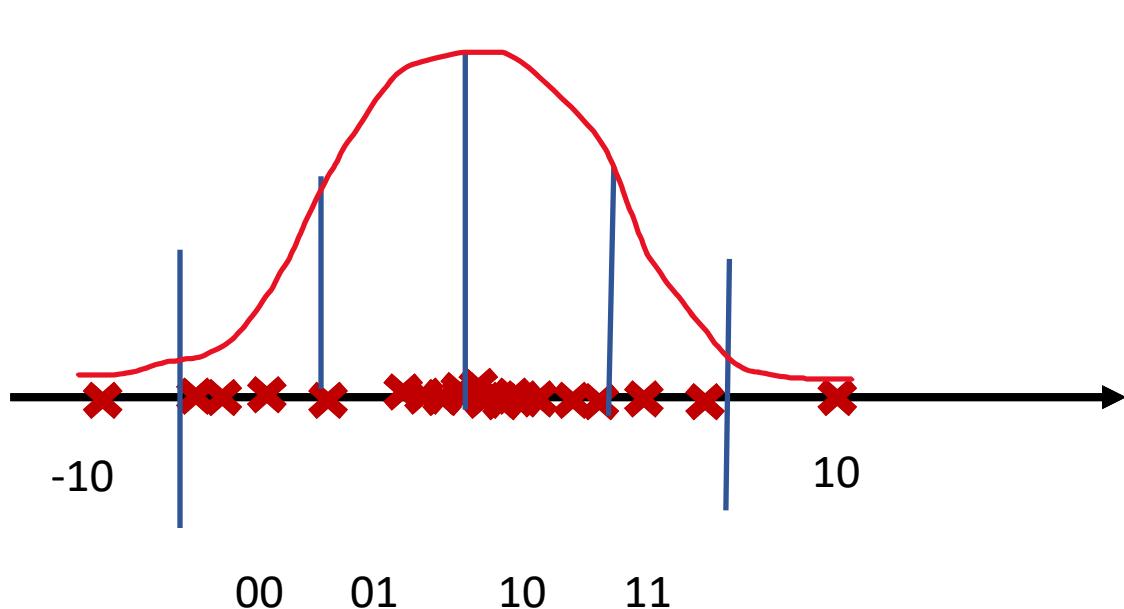
# Quantization: Data with less bits

Normal Float Quantization



Truncate the outliers!

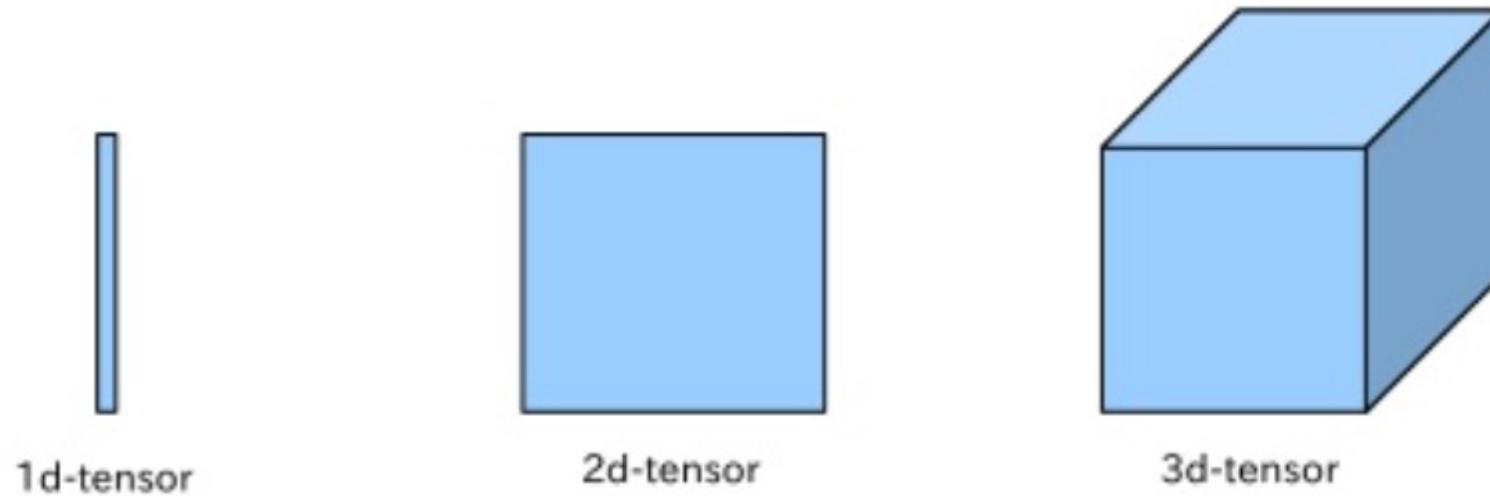
# Quantization: Data with less bits



Binary	Decimal
01	1
00	0
11	-1
10	-2

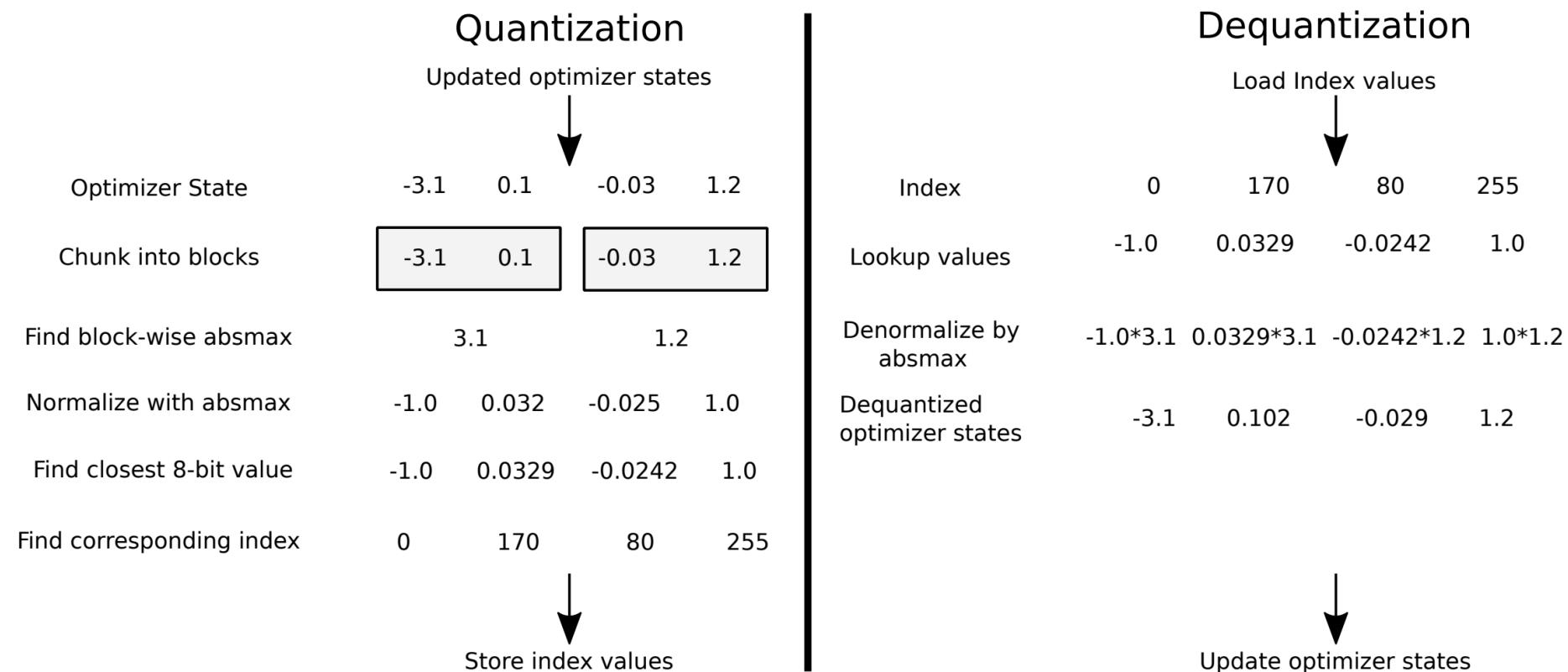
# Quantization: Data with less bits

Block-wise dynamic quantization



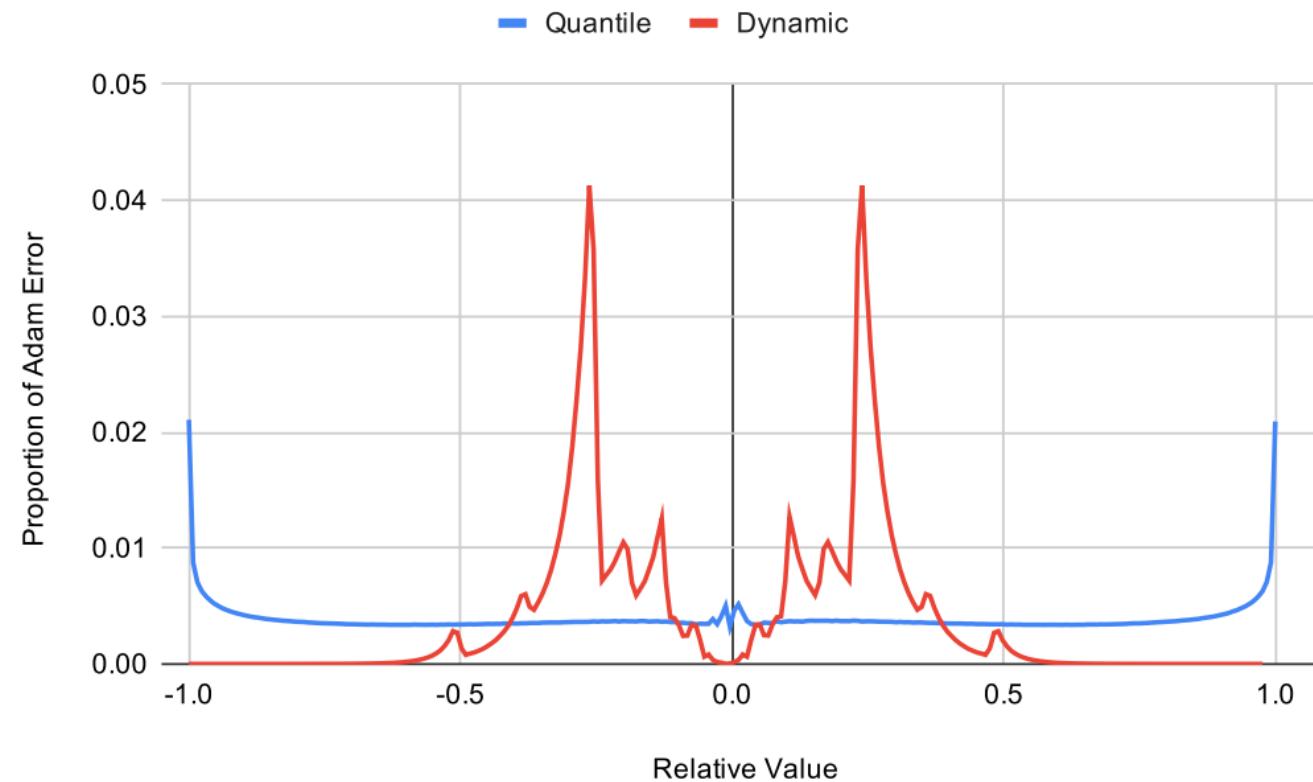
# Quantization: Data with less bits

Block-wise dynamic quantization



# Quantization: Data with less bits

Block-wise dynamic quantization



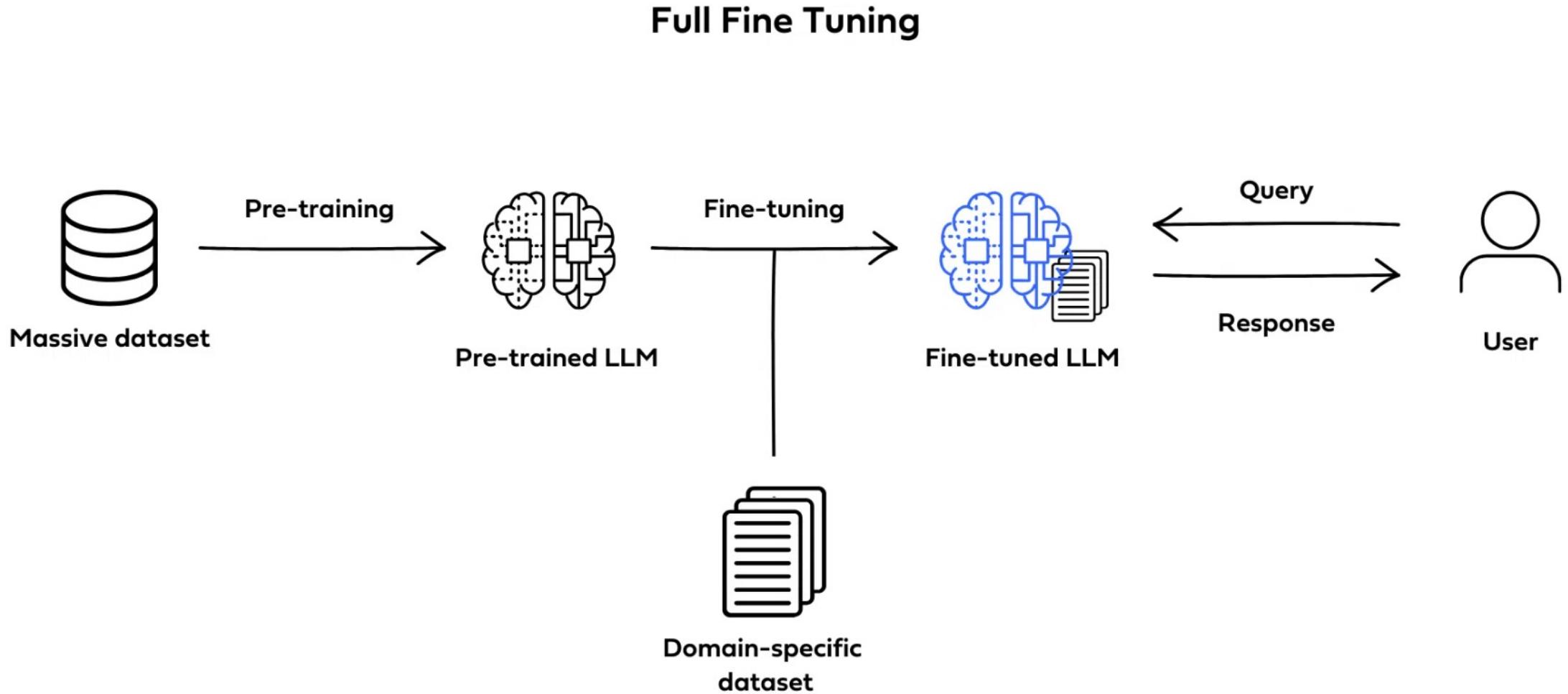
Minimize quantization error of each layer!  
Reduce the accumulated error cross the network

$$\operatorname{argmin}_{\widehat{\mathbf{W}}} \|\mathbf{WX} - \widehat{\mathbf{W}}\mathbf{X}\|_2^2.$$

$$\mathbf{X} \quad \begin{array}{|c|c|c|c|c|} \hline 2 & 45 & -1 & -17 & -1 \\ \hline 0 & 12 & 3 & -63 & 2 \\ \hline -1 & 37 & -1 & -83 & 0 \\ \hline \end{array} \quad \text{FP16} \quad \mathbf{W} \quad \begin{array}{|c|c|} \hline -1 & 0 \\ \hline 2 & 0 \\ \hline 0 & -2 \\ \hline 3 & -2 \\ \hline -1 & 2 \\ \hline \end{array} \quad \text{FP16}$$

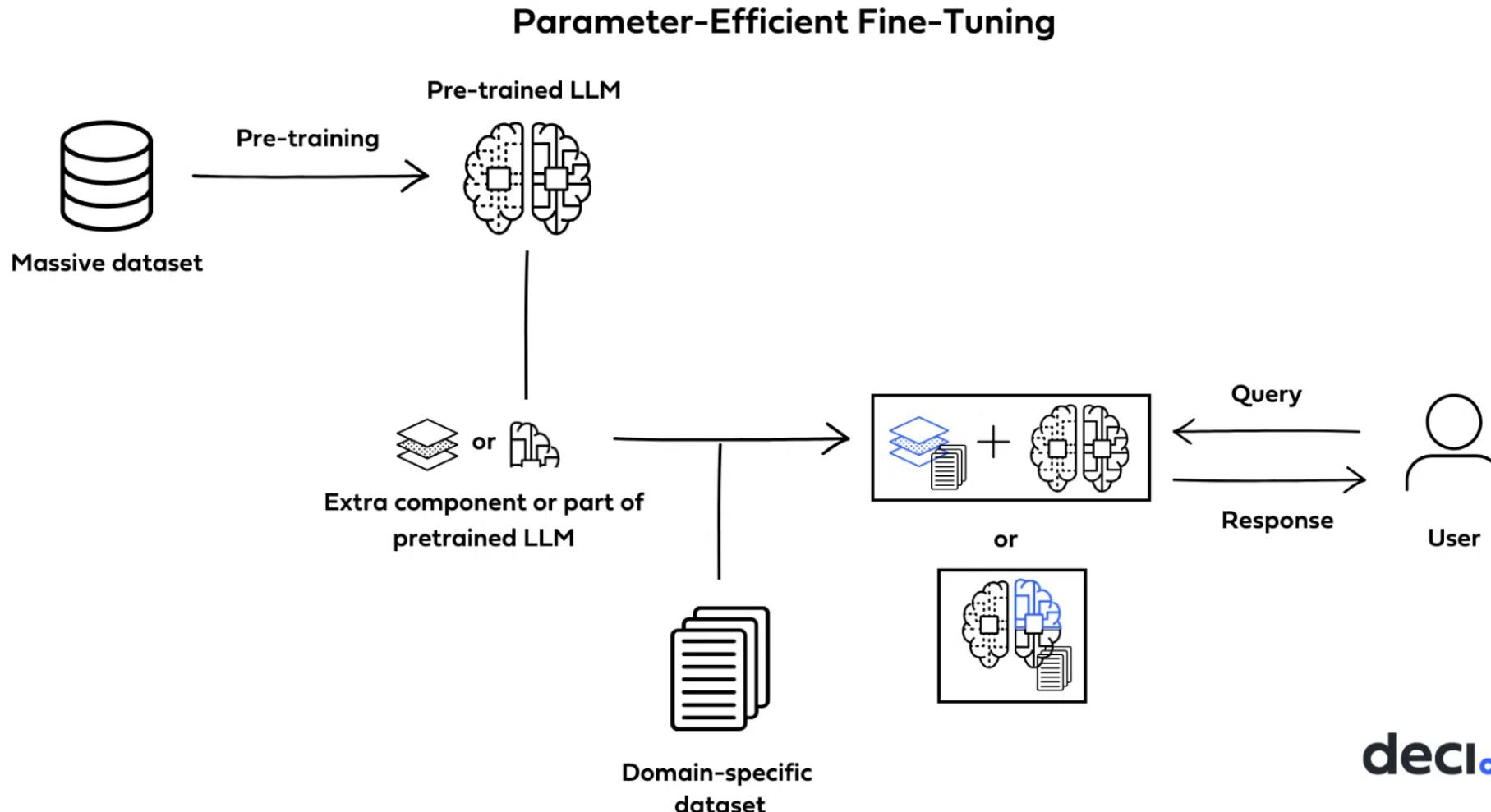
# Method-1 QLORA: Efficient Finetuning of Quantized LLMs

Full fine-tuning vs. Parameter-Efficient finetuning



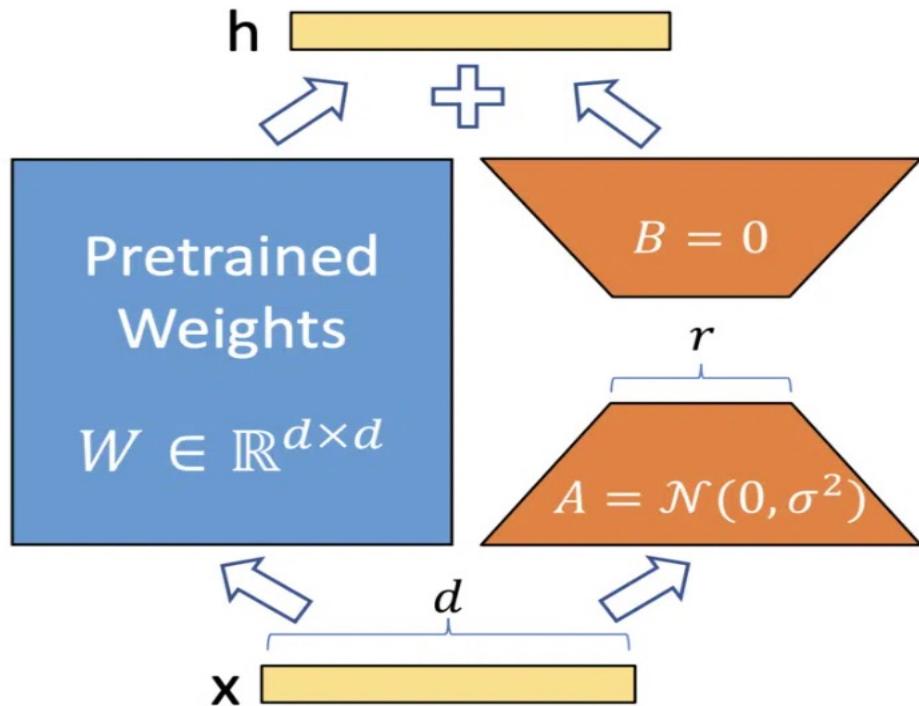
# Method-1 QLORA: Efficient Finetuning of Quantized LLMs

Full fine-tuning vs. Parameter-Efficient finetuning



# Method-1 QLORA: Efficient Finetuning of Quantized LLMs

LoRA: was introduced in early 2023, the most commonly used PEFT method

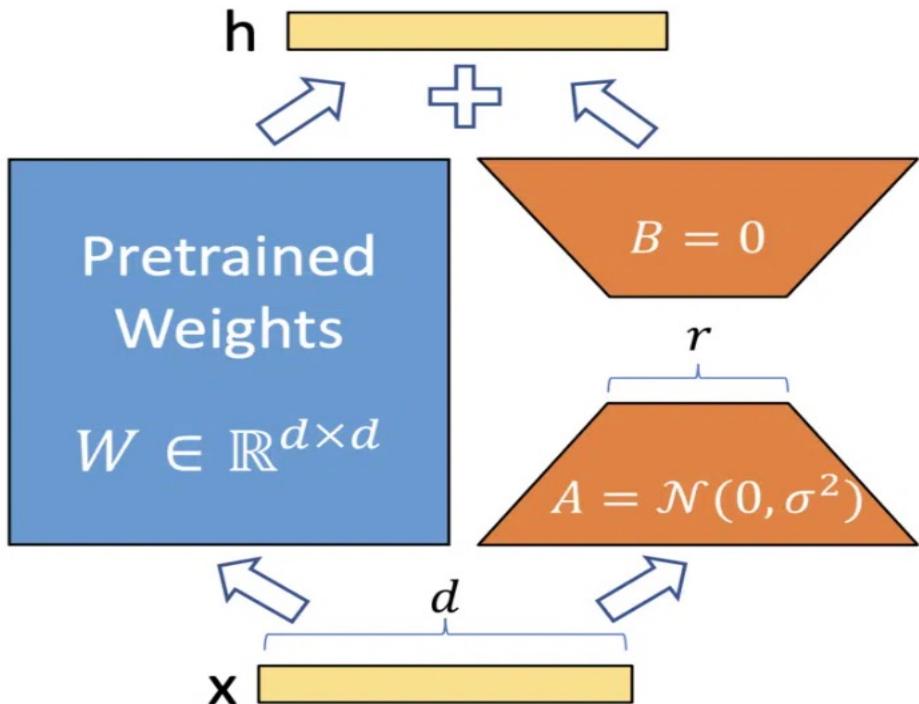


$$W' = W + A @ B$$

*Figure 1: LoRA reparameterization trains  
only A and B. (Credit:  
<https://arxiv.org/abs/2106.09685>)*

# Method-1 QLoRA: Efficient Finetuning of Quantized LLMs

QLoRA: the most commonly used PEFT Quantization



$$W' = \text{quant}(W) + A @ B$$

Figure 1: LoRA reparameterization trains  
only  $A$  and  $B$ . (Credit:  
<https://arxiv.org/abs/2106.09685>)

## *Method-1 QLORA: Efficient Finetuning of Quantized LLMs – Normal Float Quantization*

$$W' = \text{double\_quantize}(W) + A @ B$$

## *Method-1 QLoRA: Efficient Finetuning of Quantized LLMs – Normal Float Quantization*

$$\mathbf{W}' = \text{double\_normal\_float}(\mathbf{W}) + \mathbf{A} @ \mathbf{B}$$

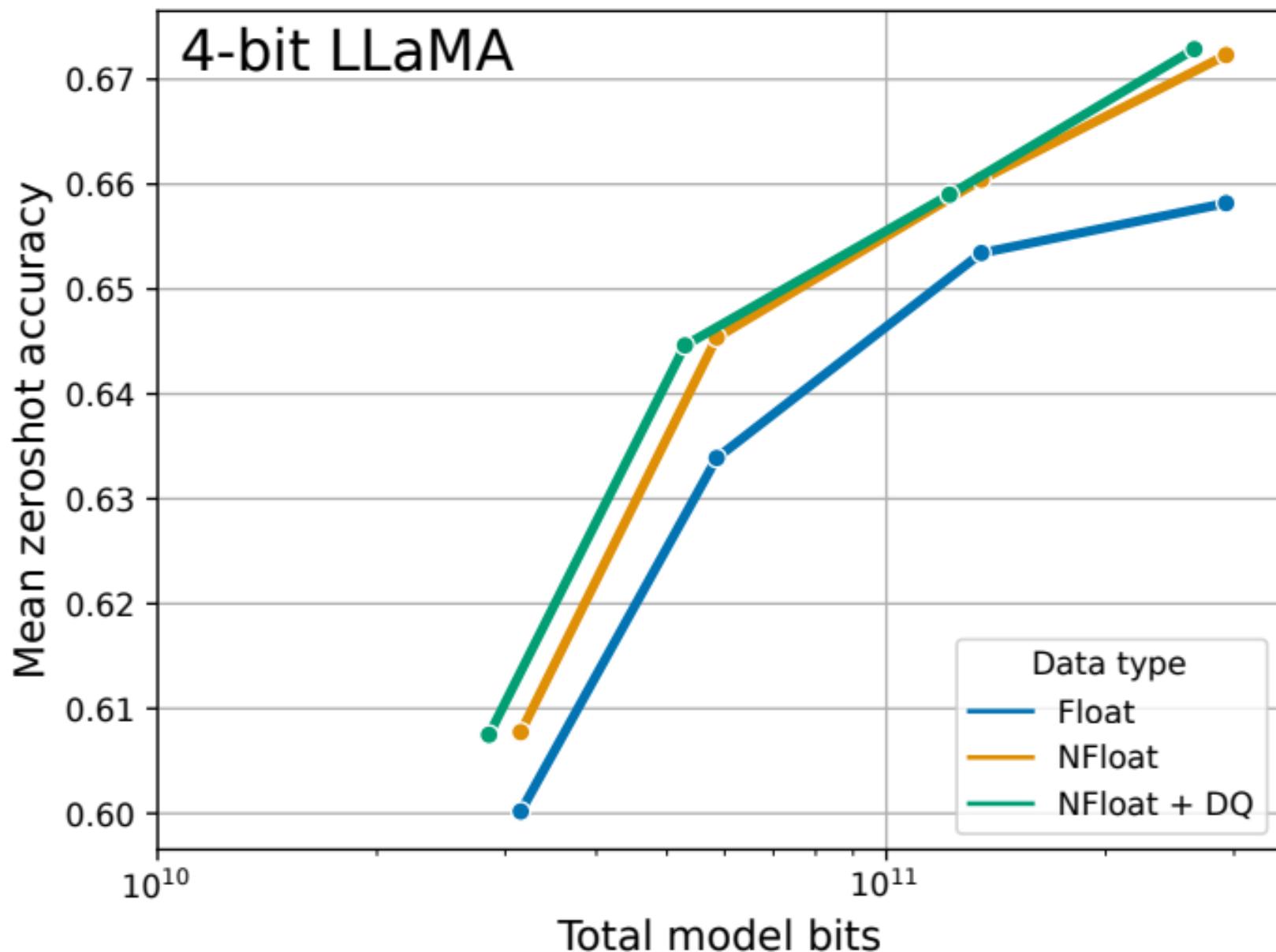
**QLoRA.** Using the components described above, we define QLoRA for a single linear layer in the quantized base model with a single LoRA adapter as follows:

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

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

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

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.

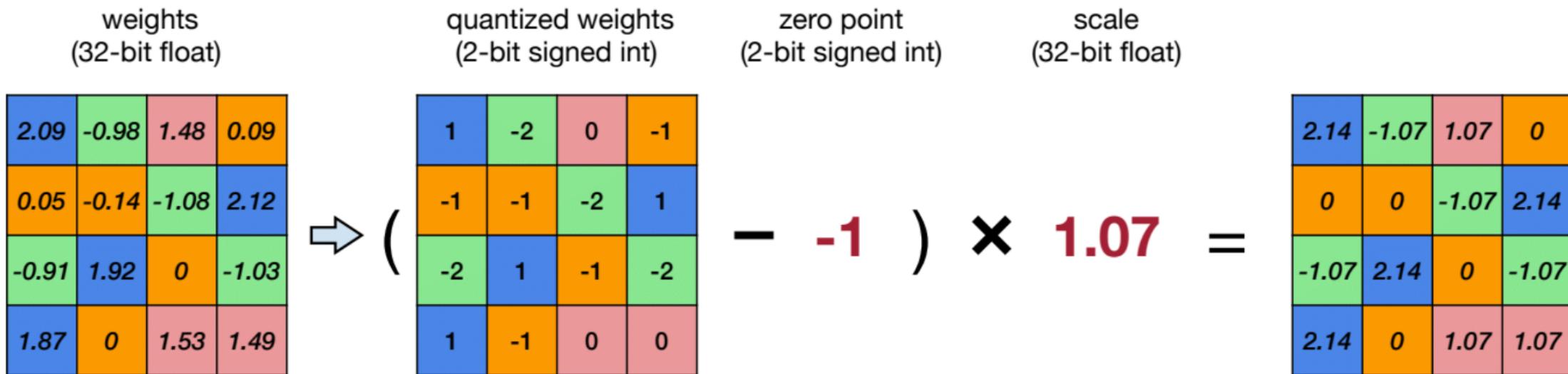


**Table 3:** Experiments comparing 16-bit BrainFloat (BF16), 8-bit Integer (Int8), 4-bit Float (FP4), and 4-bit NormalFloat (NF4) on GLUE and Super-NaturalInstructions. QLoRA replicates 16-bit LoRA and full-finetuning.

Dataset Model	GLUE (Acc.) RoBERTa-large	Super-NaturalInstructions (RougeL)				
		T5-80M	T5-250M	T5-780M	T5-3B	T5-11B
BF16	88.6	40.1	42.1	48.0	54.3	62.0
BF16 replication	88.6	40.0	42.2	47.3	54.9	-
LoRA BF16	88.8	40.5	42.6	47.1	55.4	60.7
QLoRA Int8	88.8	40.4	42.9	45.4	56.5	60.7
QLoRA FP4	88.6	40.3	42.4	47.5	55.6	60.9
QLoRA NF4 + DQ	-	40.4	42.7	47.7	55.3	60.9

# Linear Quantization

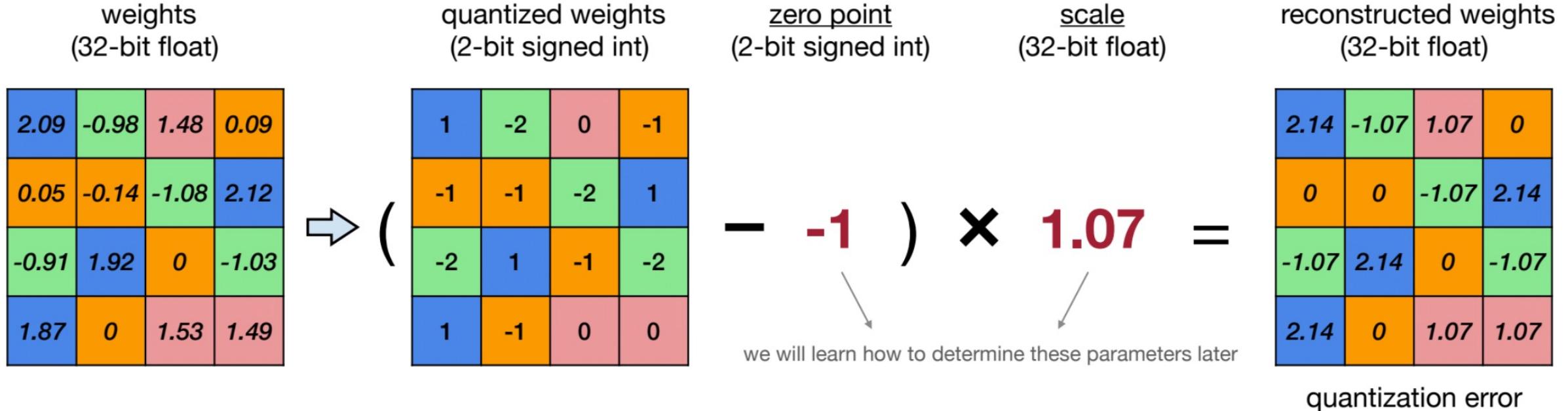
An affine mapping of integers to real numbers  $r = S(q - Z)$



Binary	Decimal
01	1
00	0
11	-1
10	-2

# What is Linear Quantization?

An affine mapping of integers to real numbers

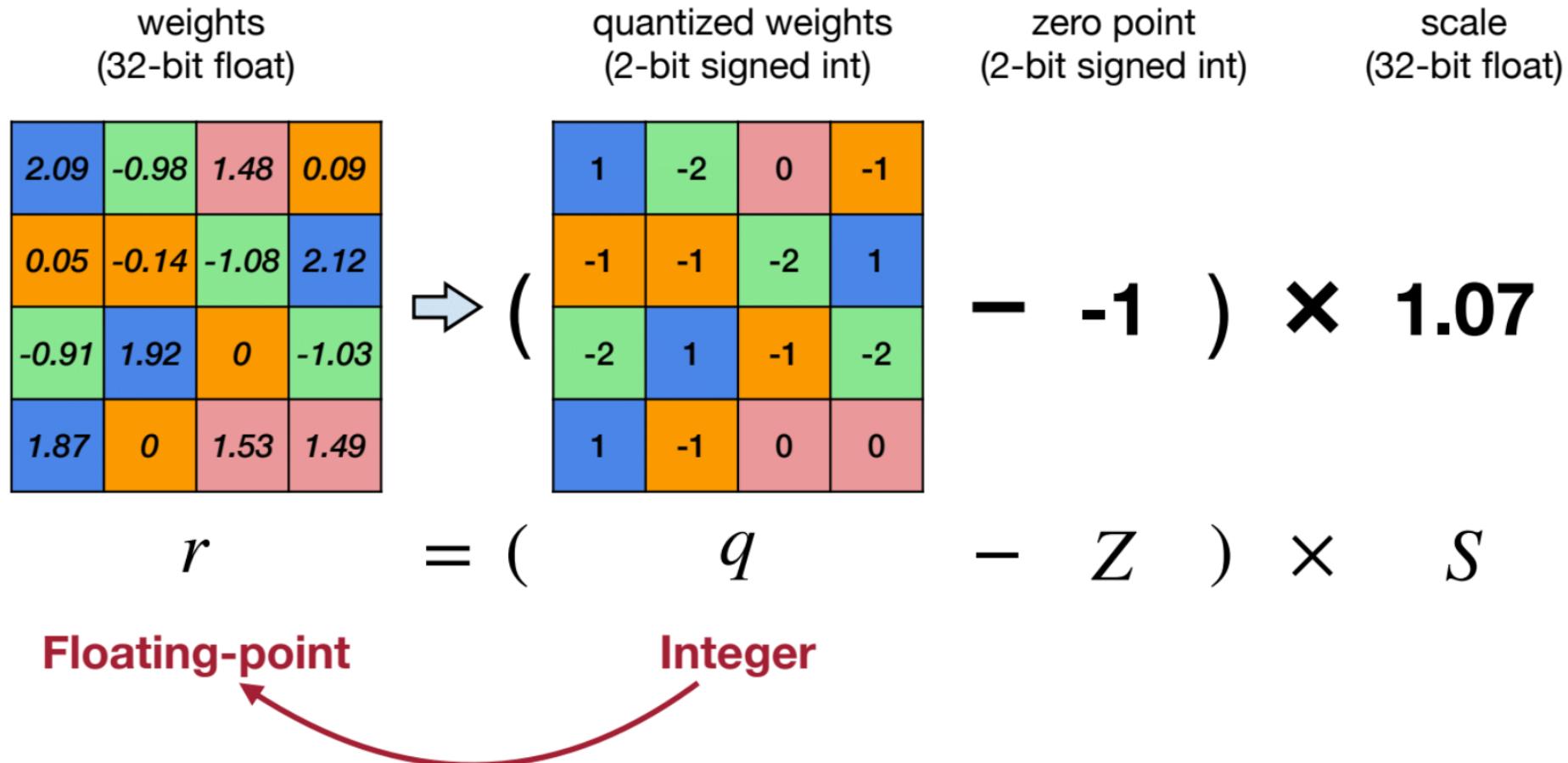


Binary	Decimal
01	1
00	0
11	-1
10	-2

-0.05	0.09	0.41	0.09
0.05	-0.14	-0.01	-0.02
0.16	-0.22	0	0.04
-0.27	0	0.46	0.42

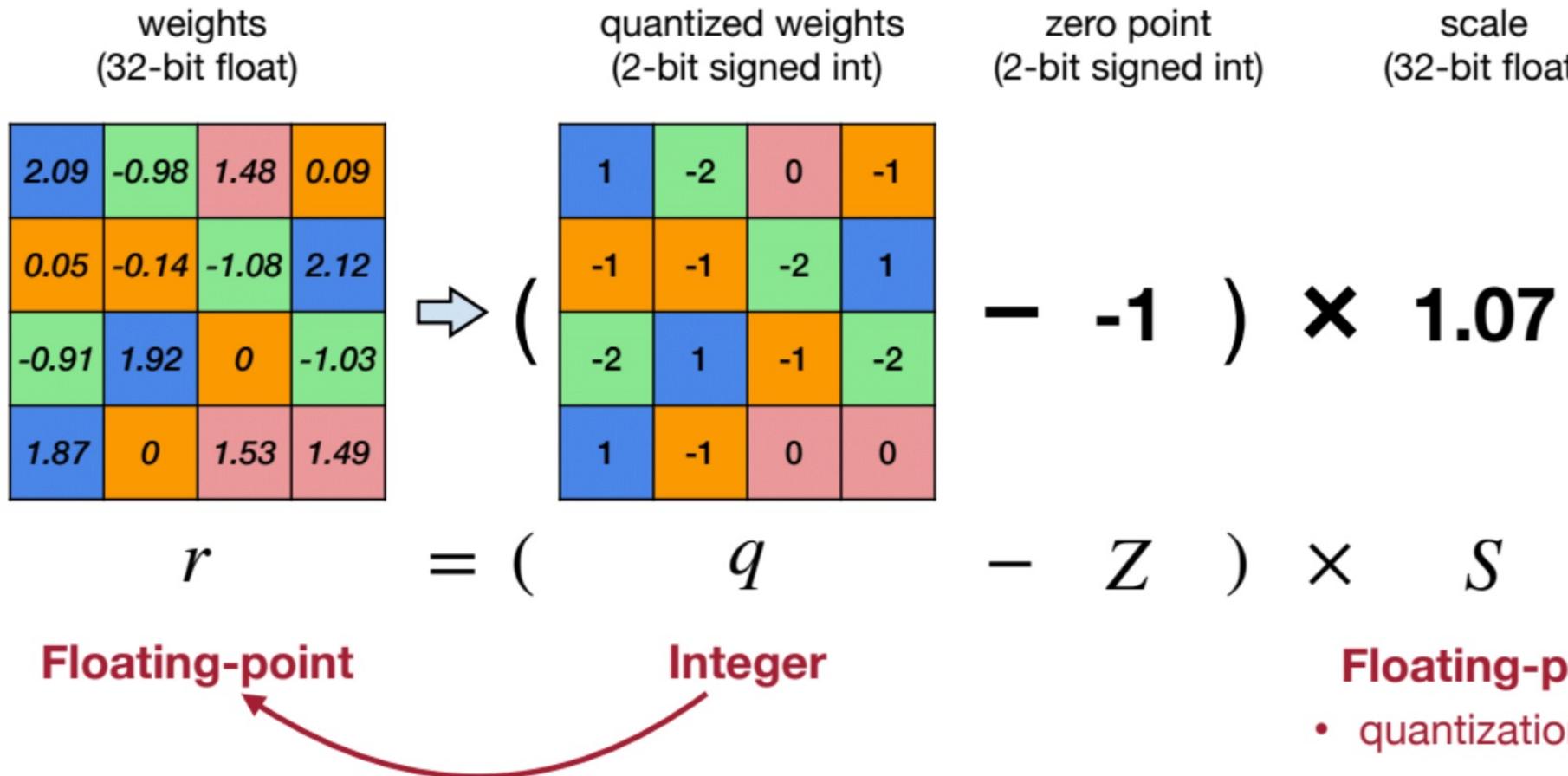
# Linear Quantization

An affine mapping of integers to real numbers  $r = S(q - Z)$



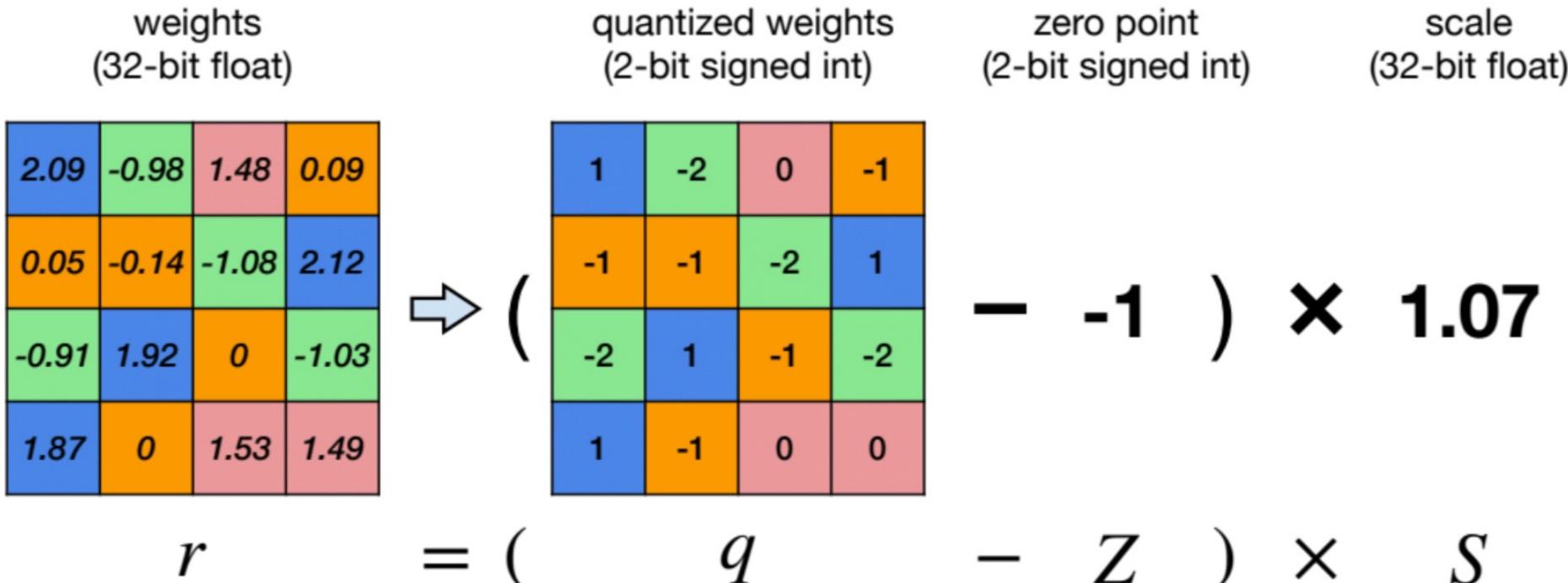
# Linear Quantization

An affine mapping of integers to real numbers  $r = S(q - Z)$



# Linear Quantization

An affine mapping of integers to real numbers  $r = S(q - Z)$



Floating-point

Integer

Integer

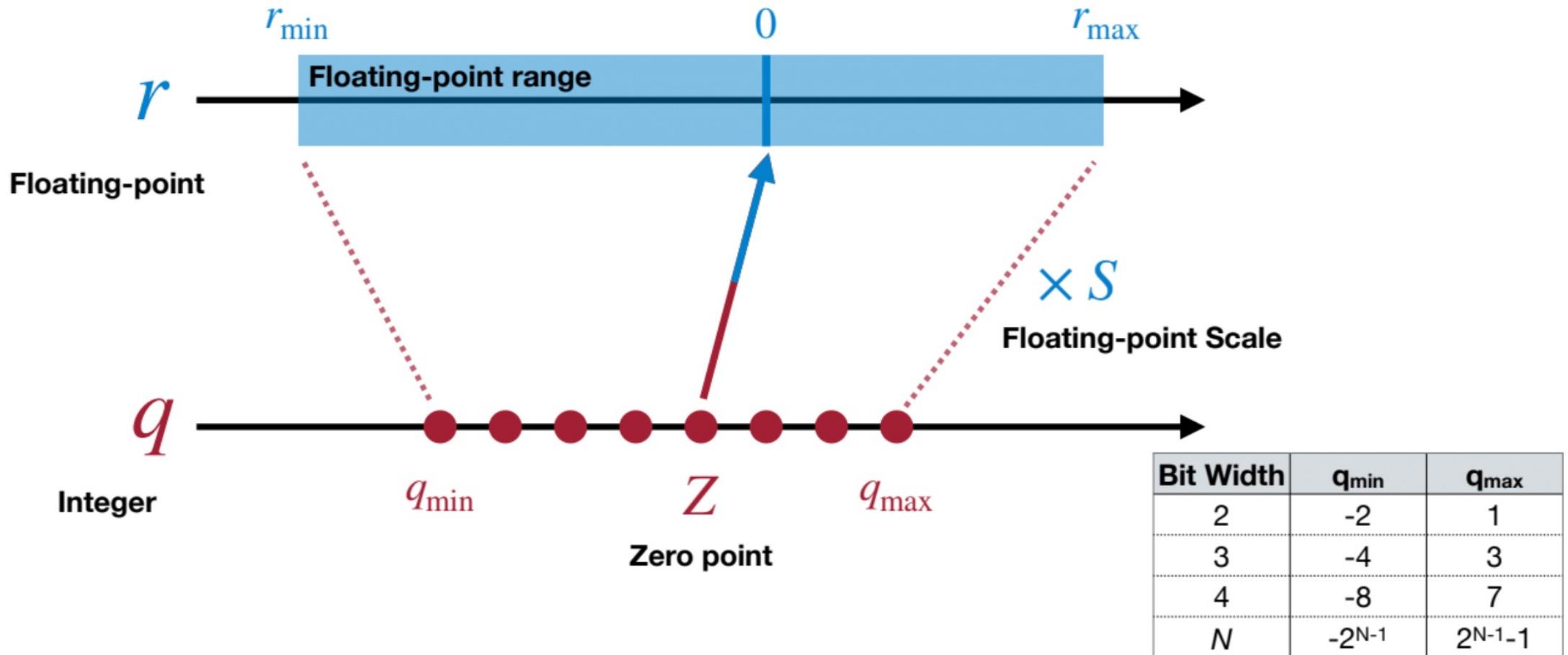
Floating-point

- quantization parameter
- allow real number  $r=0$  be exactly representable by a quantized integer  $Z$

- quantization parameter

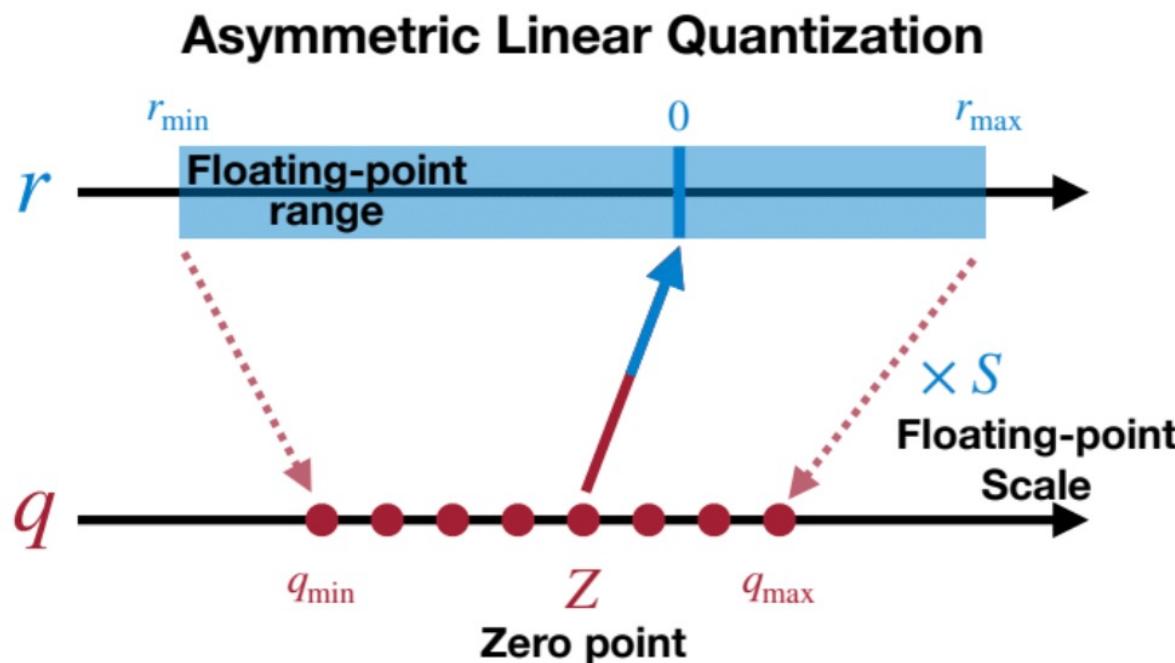
# Linear Quantization

An affine mapping of integers to real numbers  $r = S(q - Z)$



# Scale and Zero Point of Linear Quantization

Linear Quantization is an affine mapping of integers to real numbers  $r = S(q - Z)$

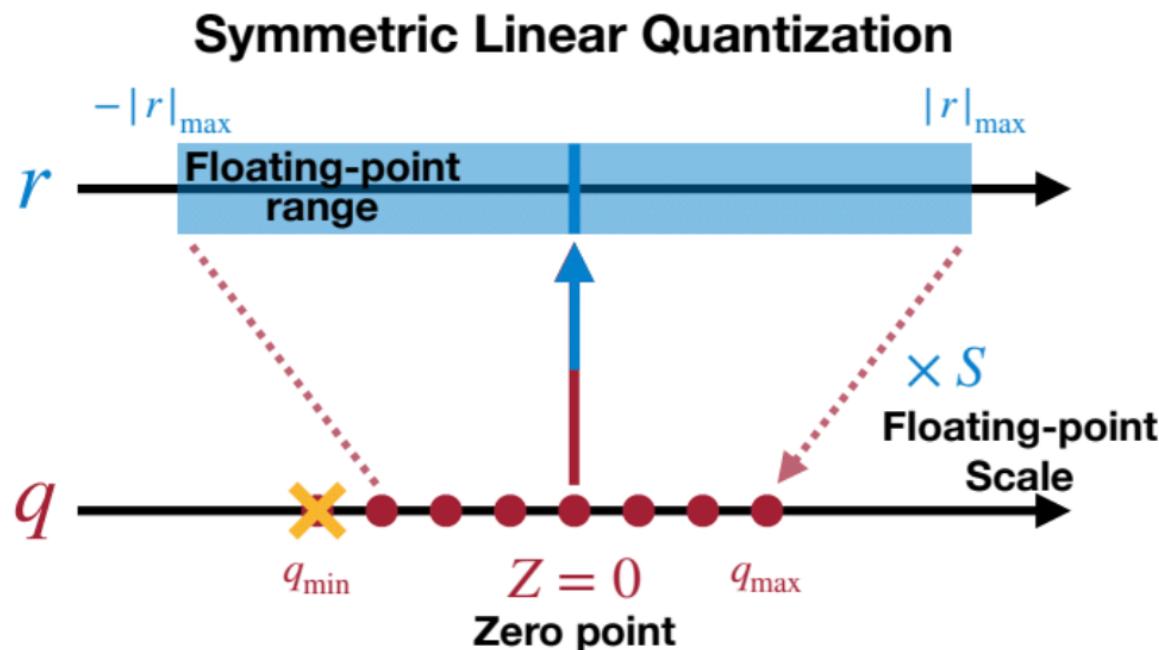


2.09	-0.98	1.48	0.09
0.05	-0.14	-1.08	2.12
-0.91	1.92	0	-1.03
1.87	0	1.53	1.49

$$S = \frac{r_{\max} - r_{\min}}{q_{\max} - q_{\min}}$$
$$Z = q_{\min} - \frac{r_{\min}}{S}$$
$$= \frac{2.12 - (-1.08)}{1 - (-2)}$$
$$= \text{round}\left(-2 - \frac{-1.08}{1.07}\right)$$
$$= -1$$

# Scale and Zero Point of Linear Quantization

Linear Quantization is an affine mapping of integers to real numbers  $r = S(q - Z)$



2.09	-0.98	1.48	0.09
0.05	-0.14	-1.08	2.12
-0.91	1.92	0	-1.03
1.87	0	1.53	1.49

$$S = \frac{|r|_{\max}}{q_{\max}}$$
$$= \frac{2.12}{1}$$
$$= 2.12$$
$$Z = 0$$

# **Post-Training Quantization**

**How should we get the optimal linear quantization parameters ( $S$ ,  $Z$ )?**

**Topic I: Quantization Granularity**

**Topic II: Dynamic Range Clipping**

**Topic III: Rounding**

# Post-Training Quantization

How should we get the optimal linear quantization parameters ( $S, Z$ )?

**Topic I: Quantization Granularity**

Topic II: Dynamic Range Clipping

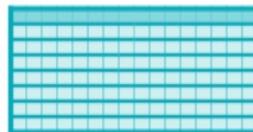
Topic III: Rounding

# Quantization Granularity

- Per-Tensor Quantization

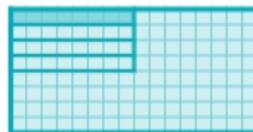


- Per-Channel Quantization



- Group Quantization

- Per-Vector Quantization



- Shared Micro-exponent (MX) data type

# Quantization Granularity

- Per-Tensor Quantization



- Per-Channel Quantization

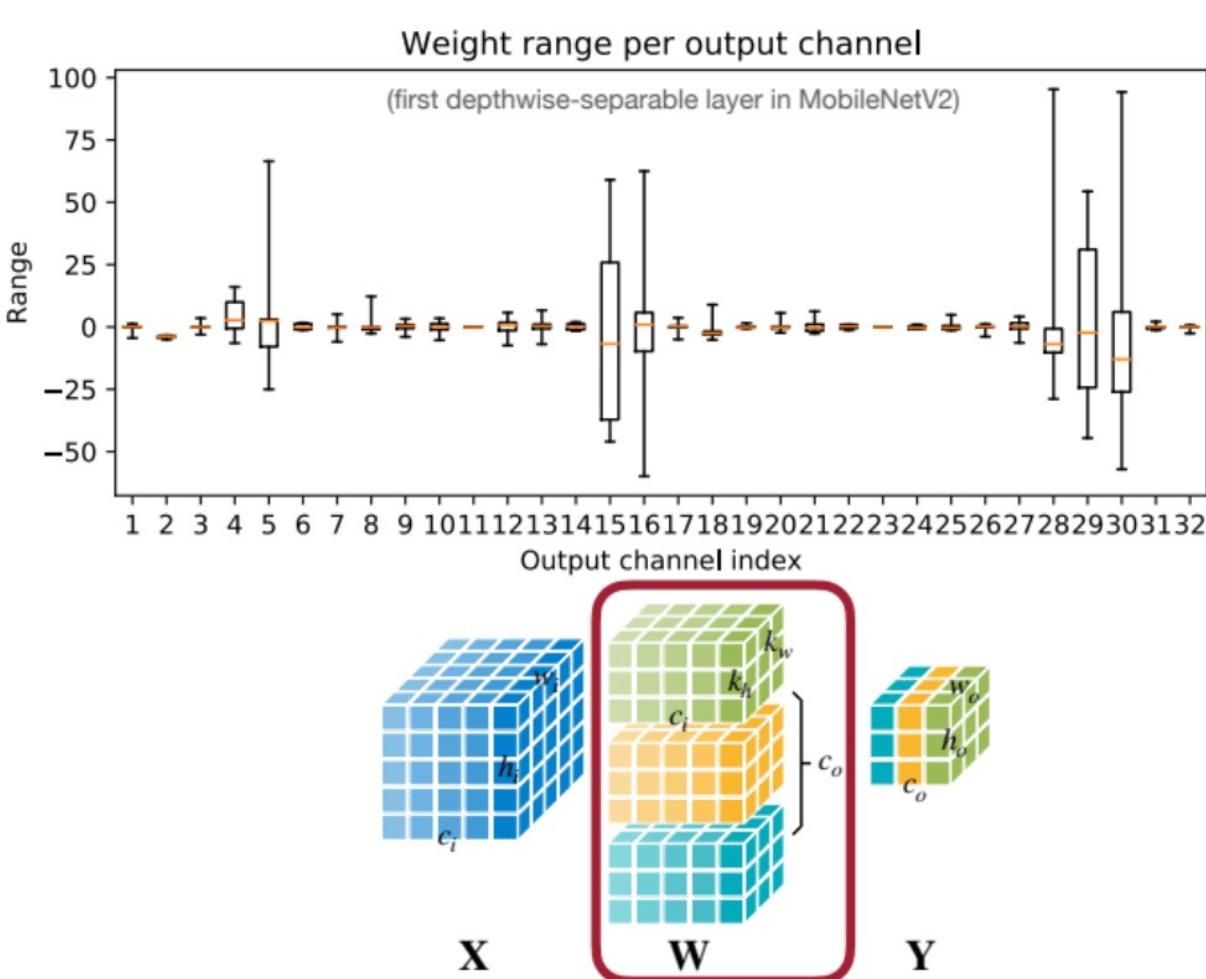


- Group Quantization



- Shared Micro-exponent (MX) data type

# Symmetric Linear Quantization on Weights



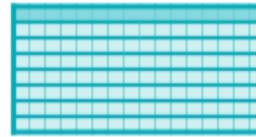
- $|r|_{\max} = |\mathbf{W}|_{\max}$
- Using *single* scale  $S$  for whole weight tensor  
**(Per-Tensor Quantization)**
  - works well for large models
  - accuracy drops for small models
- Common failure results from
  - large differences (more than 100x) in ranges of weights for different output channels — outlier weight
- Solution: **Per-Channel Quantization**

# Quantization Granularity

- Per-Tensor Quantization



- **Per-Channel Quantization**



- Group Quantization



- Per-Vector Quantization

- Shared Micro-exponent (MX) data type

# Per-Channel Weight Quantization

Example: 2-bit linear quantization

<i>ic</i>	Per-Channel Quantization				Per-Tensor Quantization
<i>oc</i>	2.09	-0.98	1.48	0.09	
	0.05	-0.14	-1.08	2.12	
	-0.91	1.92	0	-1.03	
	1.87	0	1.53	1.49	

# Per-Channel Weight Quantization

Example: 2-bit linear quantization

	<i>ic</i>		
<i>oc</i>			
	2.09	-0.98	1.48
	0.05	-0.14	-1.08
			2.12
	-0.91	1.92	0
			-1.03
	1.87	0	1.53
			1.49

## Per-Channel Quantization

## Per-Tensor Quantization

$$|r|_{\max} = 2.12$$

$$S = \frac{|r|_{\max}}{q_{\max}} = \frac{2.12}{2^{2-1} - 1} = 2.12$$

1	0	1	0
0	0	-1	1
0	1	0	0
1	0	1	1

Quantized

Reconstructed

$$\|\mathbf{W} - S\mathbf{q}_{\mathbf{W}}\|_F = 2.28$$

# Per-Channel Weight Quantization

Example: 2-bit linear quantization

ic				Per-Channel Quantization			
oc	2.09	-0.98	1.48	0.09	$ r _{\max} = 2.09$	$S_0 = 2.09$	
	0.05	-0.14	-1.08	2.12	$ r _{\max} = 2.12$	$S_1 = 2.12$	
	-0.91	1.92	0	-1.03	$ r _{\max} = 1.92$	$S_2 = 1.92$	
	1.87	0	1.53	1.49	$ r _{\max} = 1.87$	$S_3 = 1.87$	

Per-Tensor Quantization

$$S = \frac{|r|_{\max}}{q_{\max}} = \frac{2.12}{2^{2-1} - 1} = 2.12$$

Quantized				Reconstructed			
1	0	1	0	2.12	0	2.12	0
0	0	-1	1	0	0	-2.12	2.12
0	1	0	0	0	2.12	0	0
1	0	1	1	2.12	0	2.12	2.12

$$\|\mathbf{W} - S\mathbf{q}_W\|_F = 2.28$$

# Per-Channel Weight Quantization

Example: 2-bit linear quantization

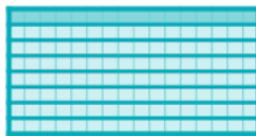
ic				Per-Channel Quantization				Per-Tensor Quantization																																																																											
oc	2.09	-0.98	1.48	0.09	$ r _{\max} = 2.09$	$S_0 = 2.09$																																																																													
	0.05	-0.14	-1.08	2.12	$ r _{\max} = 2.12$	$S_1 = 2.12$																																																																													
	-0.91	1.92	0	-1.03	$ r _{\max} = 1.92$	$S_2 = 1.92$																																																																													
	1.87	0	1.53	1.49	$ r _{\max} = 1.87$	$S_3 = 1.87$																																																																													
				<table border="1"> <tr><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>-1</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>-1</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td></tr> </table>				1	0	1	0	0	0	-1	1	0	1	0	-1	1	0	1	1	<table border="1"> <tr><td>2.09</td><td>0</td><td>2.09</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>-2.12</td><td>2.12</td></tr> <tr><td>0</td><td>1.92</td><td>0</td><td>-1.92</td></tr> <tr><td>1.87</td><td>0</td><td>1.87</td><td>1.87</td></tr> </table>				2.09	0	2.09	0	0	0	-2.12	2.12	0	1.92	0	-1.92	1.87	0	1.87	1.87	<table border="1"> <tr><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>-1</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td></tr> </table>				1	0	1	0	0	0	-1	1	0	1	0	0	1	0	1	1	<table border="1"> <tr><td>2.12</td><td>0</td><td>2.12</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>-2.12</td><td>2.12</td></tr> <tr><td>0</td><td>2.12</td><td>0</td><td>0</td></tr> <tr><td>2.12</td><td>0</td><td>2.12</td><td>2.12</td></tr> </table>				2.12	0	2.12	0	0	0	-2.12	2.12	0	2.12	0	0	2.12	0	2.12	2.12
1	0	1	0																																																																																
0	0	-1	1																																																																																
0	1	0	-1																																																																																
1	0	1	1																																																																																
2.09	0	2.09	0																																																																																
0	0	-2.12	2.12																																																																																
0	1.92	0	-1.92																																																																																
1.87	0	1.87	1.87																																																																																
1	0	1	0																																																																																
0	0	-1	1																																																																																
0	1	0	0																																																																																
1	0	1	1																																																																																
2.12	0	2.12	0																																																																																
0	0	-2.12	2.12																																																																																
0	2.12	0	0																																																																																
2.12	0	2.12	2.12																																																																																
				Quantized				Reconstructed																																																																											
				$\ \mathbf{W} - \mathbf{S} \odot \mathbf{q}_W\ _F = 2.08$				$\ \mathbf{W} - S \mathbf{q}_W\ _F = 2.28$																																																																											

# Quantization Granularity

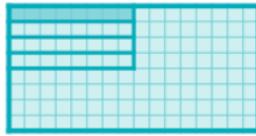
- Per-Tensor Quantization



- Per-Channel Quantization



- **Group Quantization**



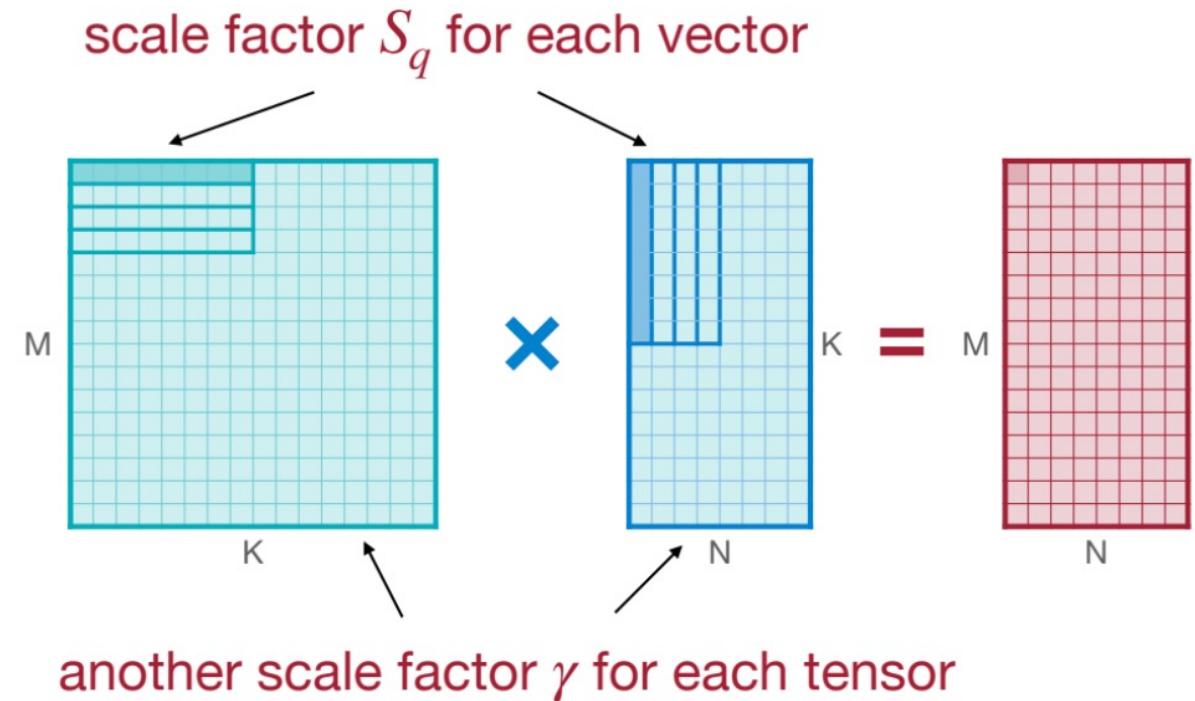
- **Per-Vector Quantization**

- **Shared Micro-exponent (MX) data type**

# VS-Quant: Per-vector Scaled Quantization

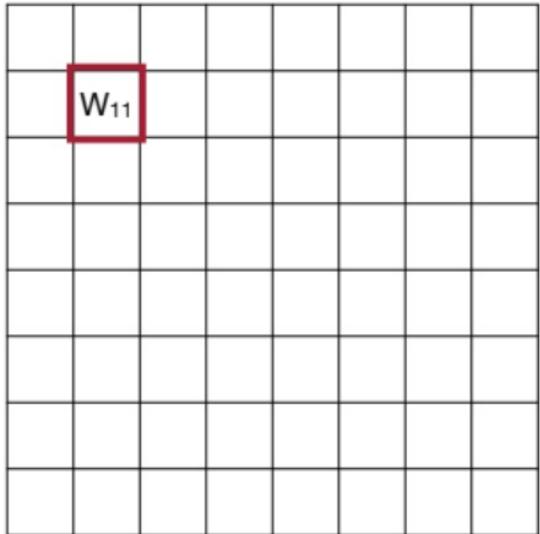
## Hierarchical scaling factor

- $r = S(q - Z) \rightarrow r = \gamma \cdot S_q(q - Z)$ 
  - $\gamma$  is a floating-point coarse grained scale factor
  - $S_q$  is an integer per-vector scale factor
  - achieves a balance between accuracy and hardware efficiency by
    - less expensive integer scale factors at finer granularity
    - more expensive floating-point scale factors at coarser granularity
- Memory Overhead of two-level scaling:
  - Given 4-bit quantization with 4-bit per-vector scale for every 16 elements, the effective bit width is  $4 + 4 / 16 = 4.25$  bits.



# Group Quantization

## Multi-level scaling scheme



$$r = (q - z) \cdot s \rightarrow$$
$$r = (q - z) \cdot s_{l_0} \cdot s_{l_1} \cdot \dots$$

$r$  : real number value

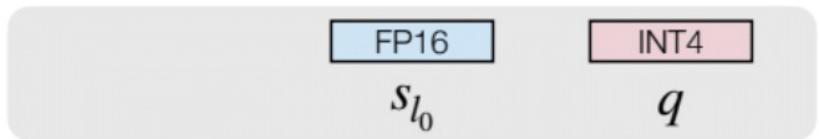
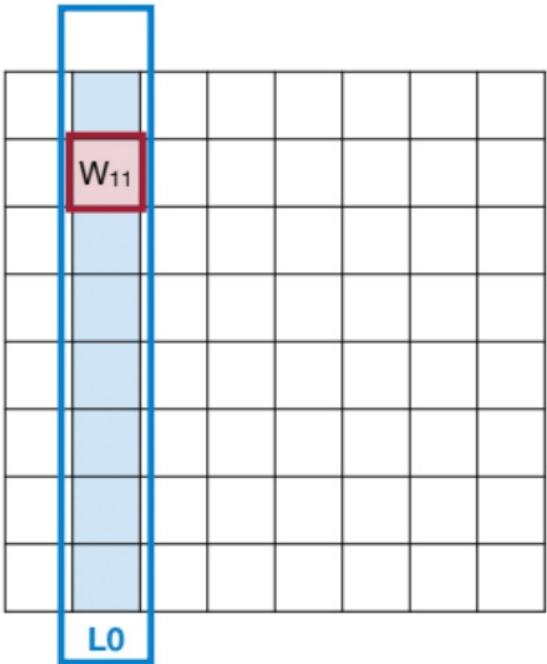
$q$  : quantized value

$z$  : zero point ( $z = 0$  is symmetric quantization)

$s$  : scale factors of different levels

# Group Quantization

## Multi-level scaling scheme



$$r = (q - z) \cdot s_{l_0} \cdot s_{l_1} \cdot \dots$$

$r$  : real number value

$q$  : quantized value

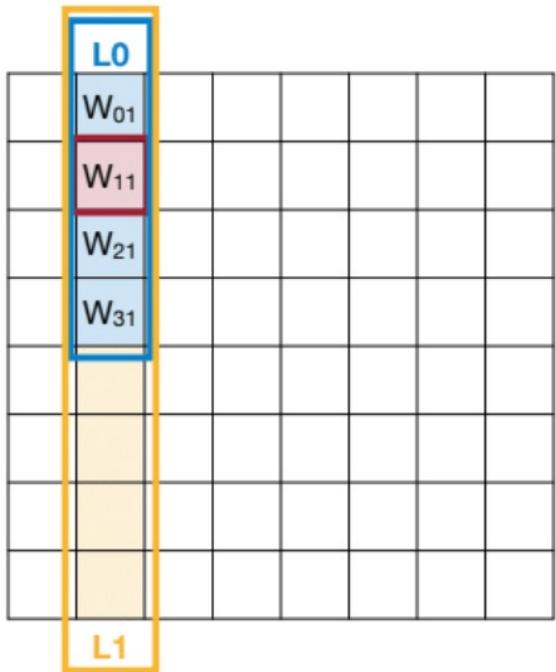
$z$  : zero point ( $z = 0$  is symmetric quantization)

$s$  : scale factors of different levels

Quantization Approach	Data Type	L0 Group Size	L0 Scale Data Type	L1 Group Size	L1 Scale Data Type	Effective Bit Width
Per-Channel Quant	INT4	Per Channel	FP16	-	-	4

# Group Quantization

## Multi-level scaling scheme



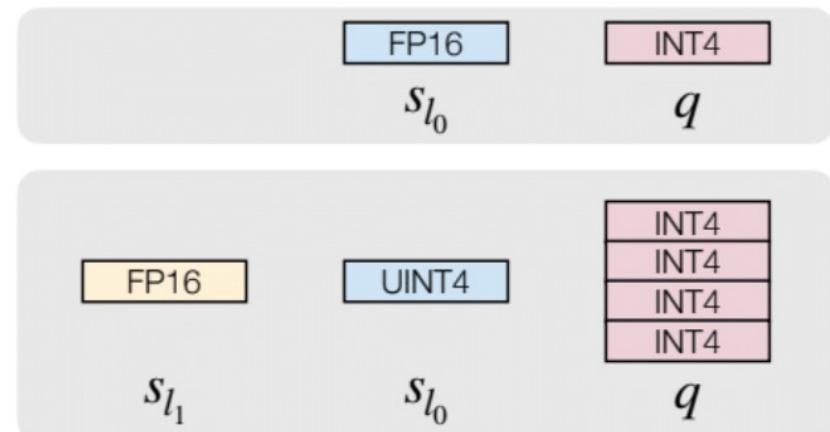
$$r = (q - z) \cdot s_{l_0} \cdot s_{l_1} \cdot \dots$$

$r$  : real number value

$q$  : quantized value

$z$  : zero point ( $z = 0$  is symmetric quantization)

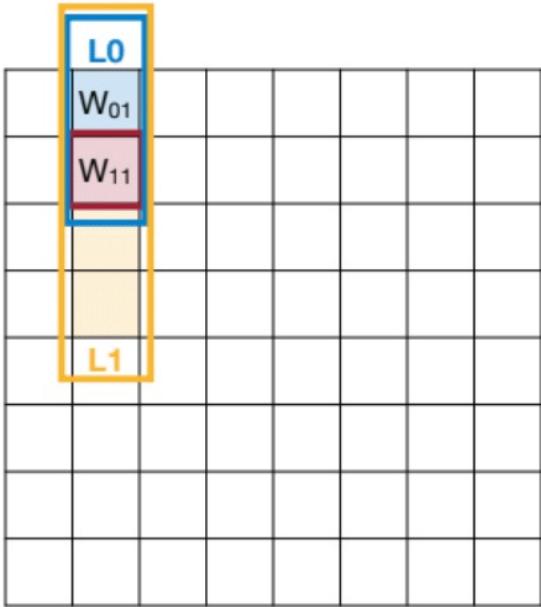
$s$  : scale factors of different levels



Quantization Approach	Data Type	L0 Group Size	L0 Scale Data Type	L1 Group Size	L1 Scale Data Type	Effective Bit Width
Per-Channel Quant	INT4	Per Channel	FP16	-	-	4
VSQ	INT4	16	UINT4	Per Channel	FP16	4+4/16=4.25

# Group Quantization

## Multi-level scaling scheme



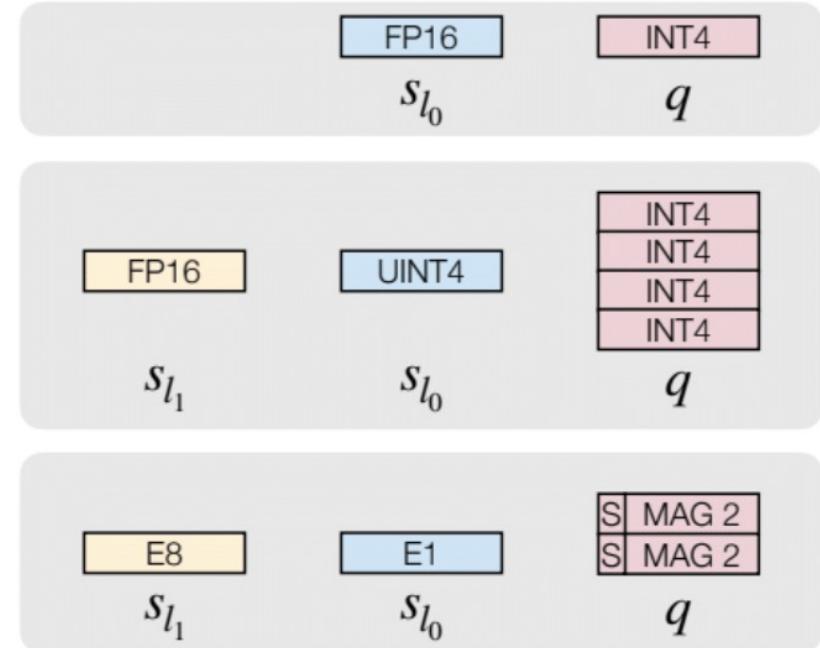
$$r = (q - z) \cdot s_{l_0} \cdot s_{l_1} \cdot \dots$$

$r$  : real number value

$q$  : quantized value

$z$  : zero point ( $z = 0$  is symmetric quantization)

$s$  : scale factors of different levels



Quantization Approach	Data Type	L0 Group Size	L0 Scale Data Type	L1 Group Size	L1 Scale Data Type	Effective Bit Width
Per-Channel Quant	INT4	Per Channel	FP16	-	-	4
VSQ	INT4	16	UINT4	Per Channel	FP16	4+4/16=4.25
MX4	S1M2	2	E1M0	16	E8M0	3+1/2+8/16=4
MX6	S1M4	2	E1M0	16	E8M0	5+1/2+8/16=6
MX9	S1M7	2	E1M0	16	E8M0	8+1/2+8/16=9

# Post-Training Quantization

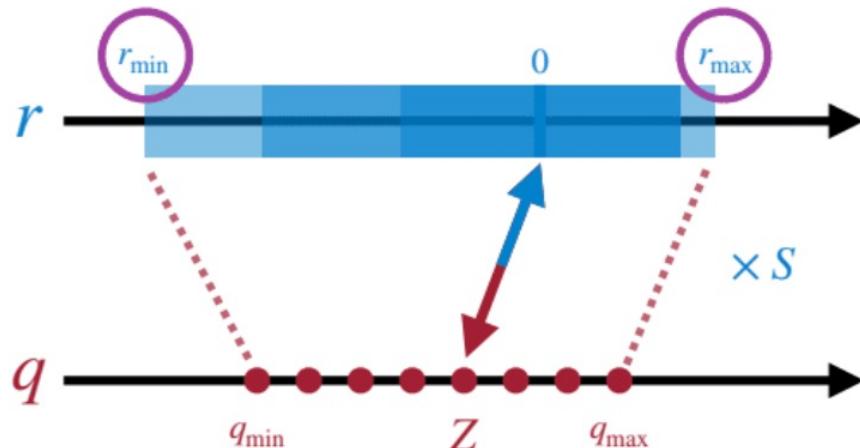
**How should we get the optimal linear quantization parameters ( $S$ ,  $Z$ )?**

Topic I: Quantization Granularity

Topic II: Dynamic Range Clipping

Topic III: Rounding

# Linear Quantization on Activations



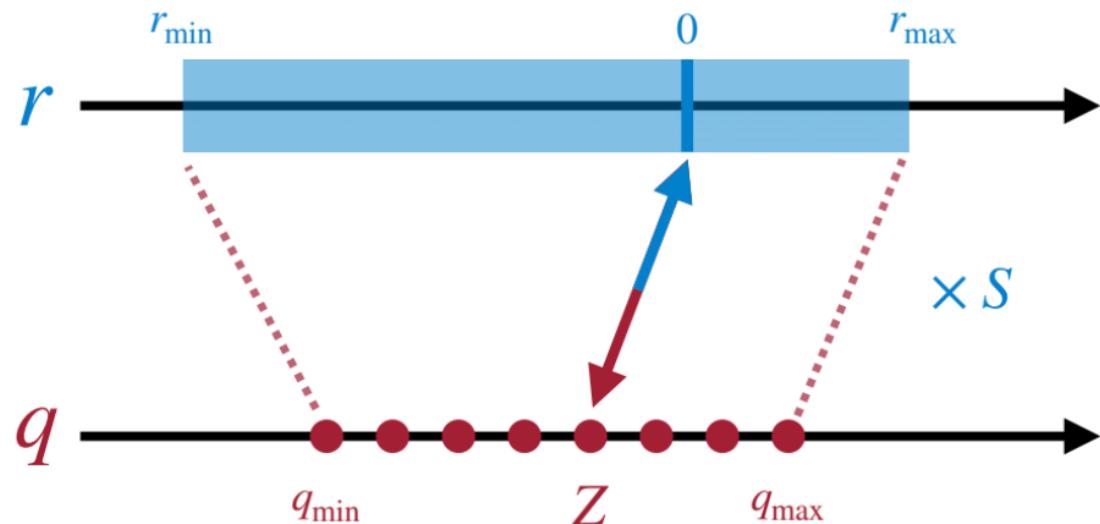
- Unlike weights, the activation range varies across inputs.
- To determine the floating-point range, the activations statistics are gathered **before** deploying the model.



# Dynamic Range for Activation Quantization

Collect activations statistics before deploying the model

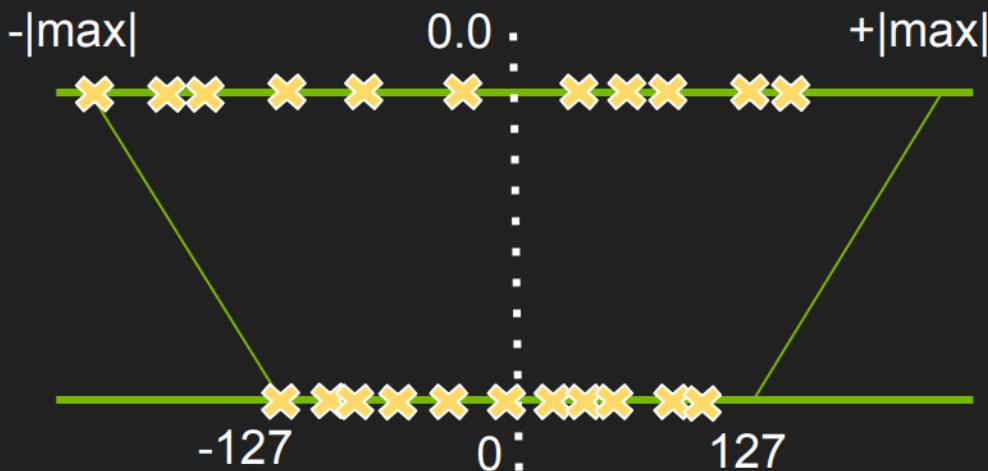
$$\hat{r}_{\max, \min}^{(t)} = \alpha \cdot r_{\max, \min}^{(t)} + (1 - \alpha) \cdot \hat{r}_{\max, \min}^{(t-1)}$$



- Type 1: During training
  - Exponential moving averages (EMA)
  - observed ranges are smoothed across thousands of training steps

# Quantization

- No saturation: map  $|\max|$  to 127



- Saturate above  $|threshold|$  to 127



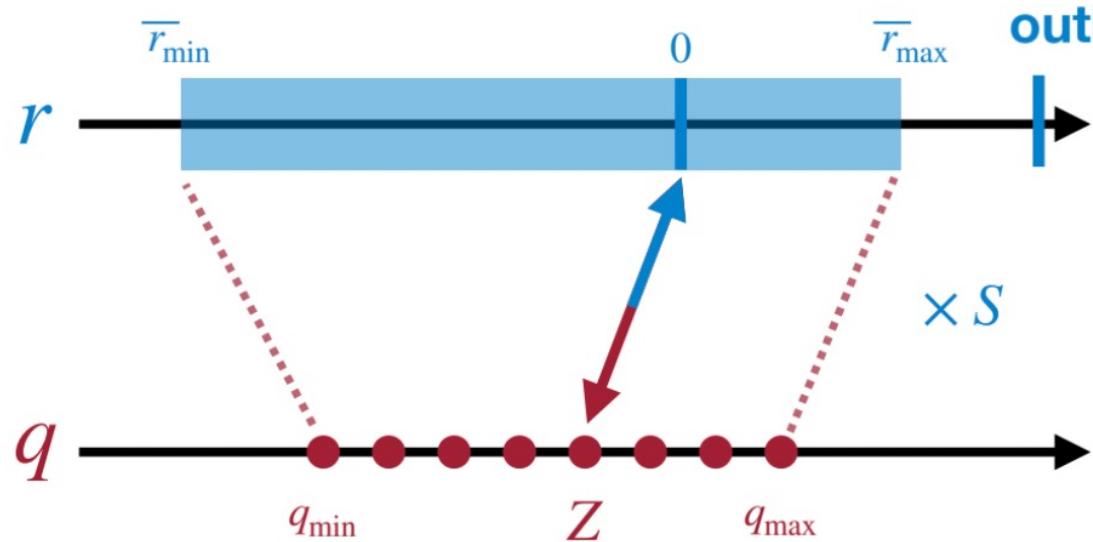
- Significant accuracy loss, in general

- Weights: no accuracy improvement
- Activations: improved accuracy
- Which  $|threshold|$  is optimal?

# Dynamic Range for Activation Quantization

Collect activations statistics before deploying the model

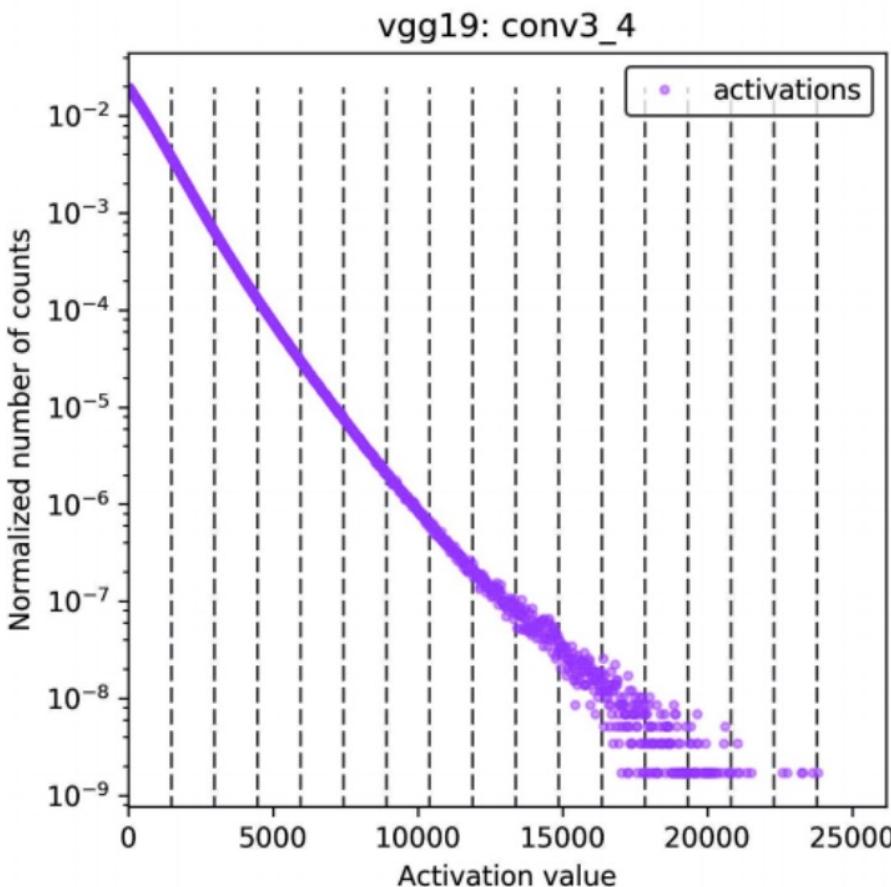
- Type 2: By running a few “calibration” batches of samples on the trained FP32 model
- **outliers**:
  - spending dynamic range on the outliers hurts the representation ability.
  - use *mean* of the min/max of each sample in the batches
  - analytical calculation (see next slide)



Neural Network Distiller

# Dynamic Range for Activation Quantization

Collect activations statistics before deploying the model



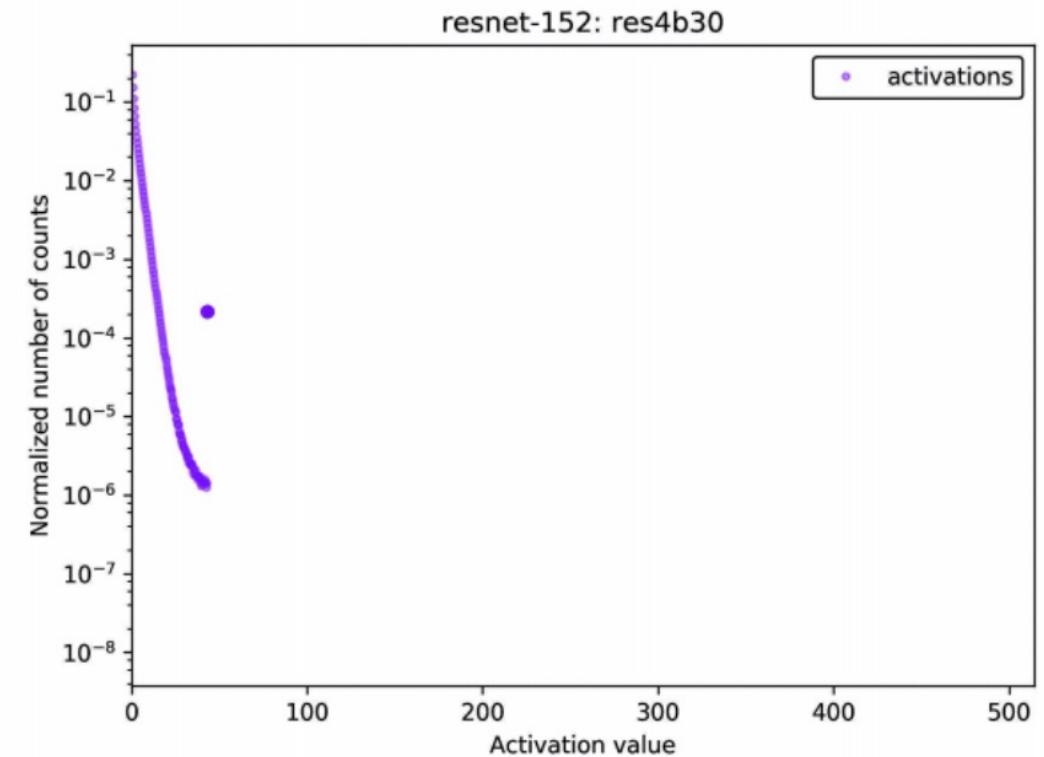
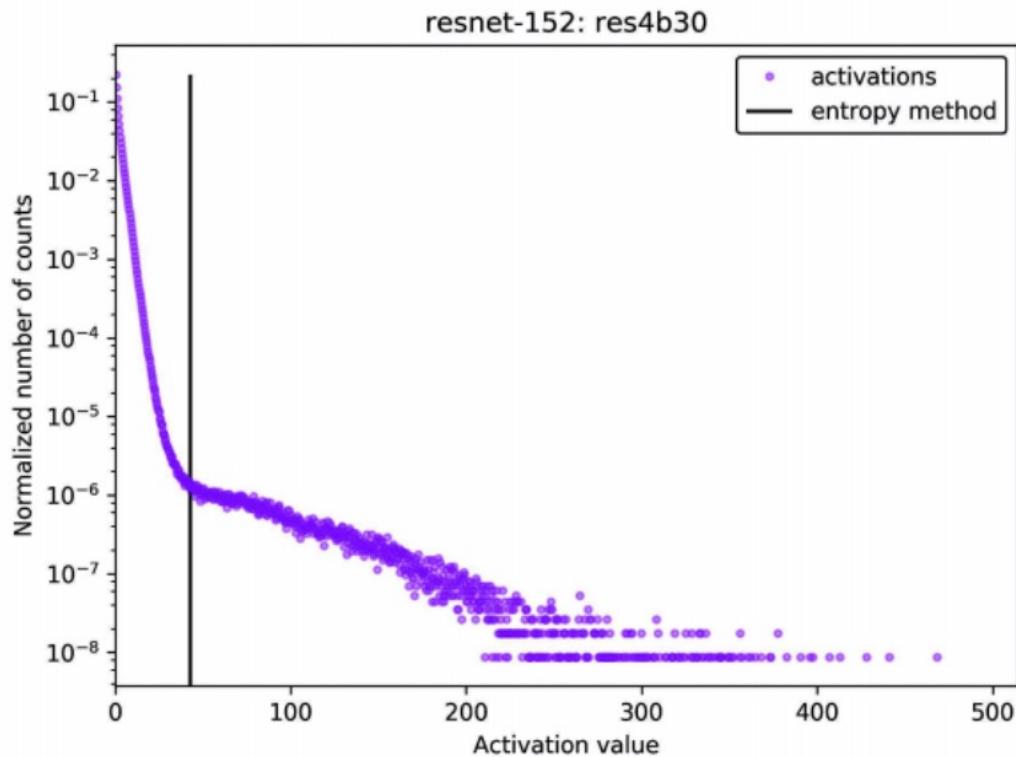
- Type 2: By running a few “calibration” batches of samples on the trained FP32 model
  - minimize loss of information, since integer model encodes the same information as the original floating-point model.
  - loss of information is measured by **Kullback-Leibler divergence** (relative entropy or information divergence):
    - for two discrete probability distributions  $P, Q$ 
$$D_{KL}(P\|Q) = \sum_i^N P(x_i)\log \frac{P(x_i)}{Q(x_i)}$$
    - intuition: KL divergence measures the amount of information lost when approximating a given encoding.

# Typical workflow in TensorRT

- You will need:
  - Model trained in FP32.
  - Calibration dataset.
- TensorRT will:
  - Run inference in FP32 on calibration dataset.
  - Collect required statistics.
  - Run calibration algorithm → optimal scaling factors.
  - Quantize FP32 weights → INT8.
  - Generate “CalibrationTable” and INT8 execution engine.

# Dynamic Range for Activation Quantization

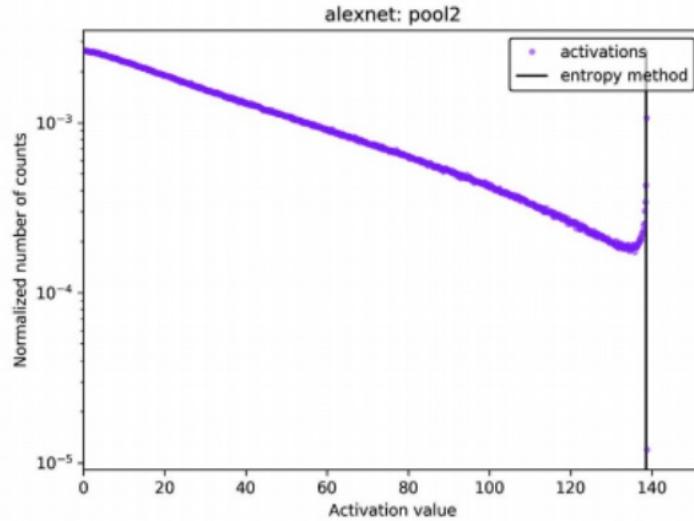
Minimize loss of information by minimizing the KL divergence



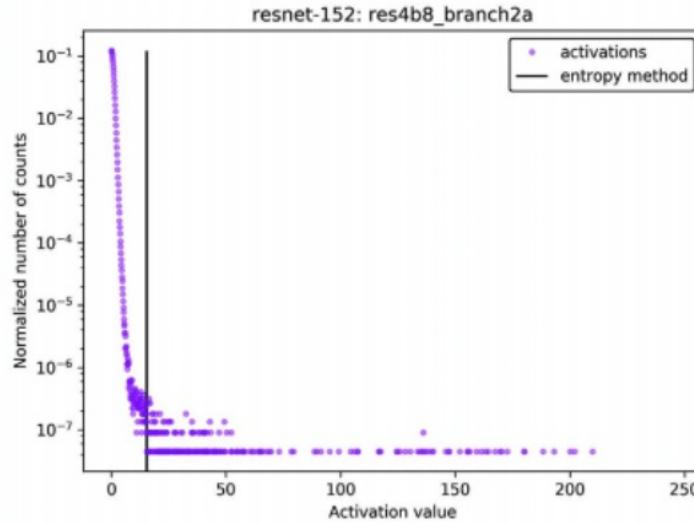
# Dynamic Range for Activation Quantization

Minimize loss of information by minimizing the KL divergence

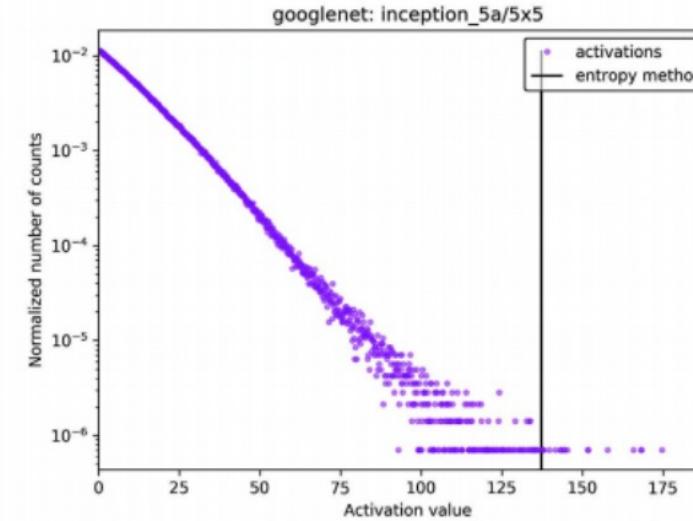
AlexNet: Pool 2



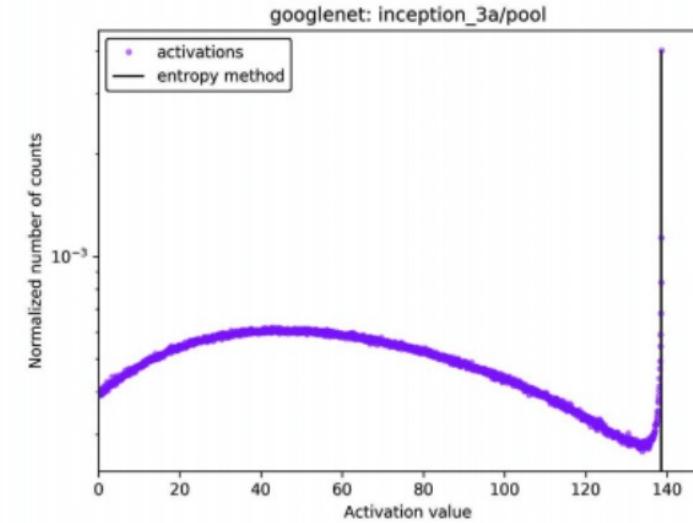
ResNet-152:  
res4b8\_branch2a



GoogleNet:  
inception\_5a/5x5



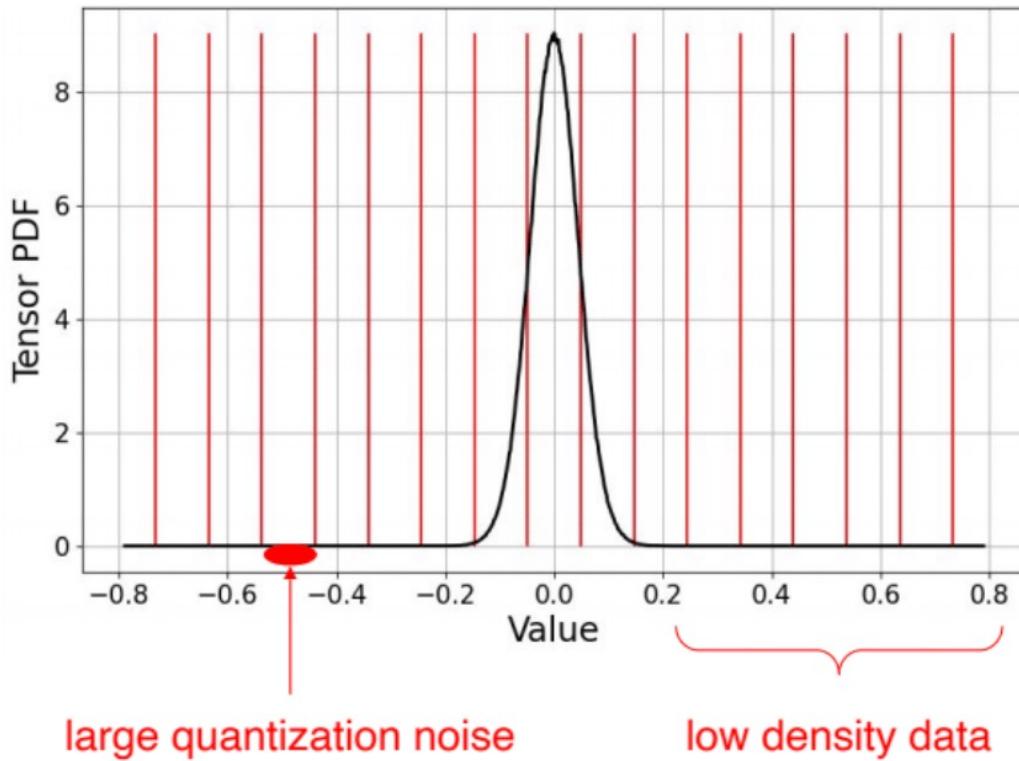
GoogleNet:  
inception\_3a/pool



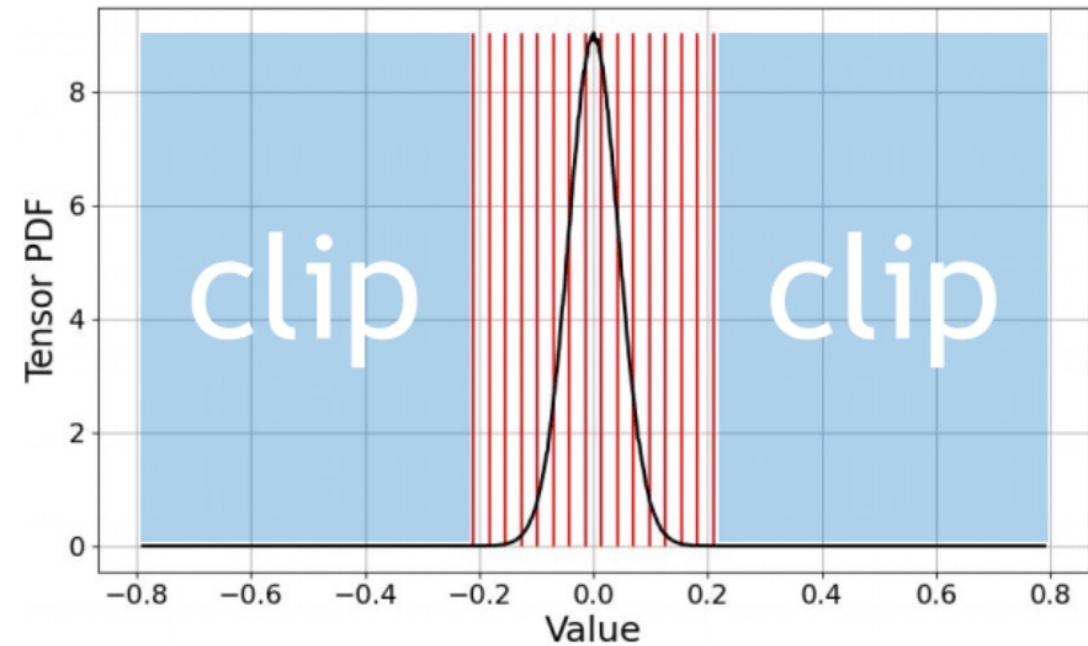
# Dynamic Range for Quantization

Minimize mean-square-error (MSE) using Newton-Raphson method

max-scaled quantization

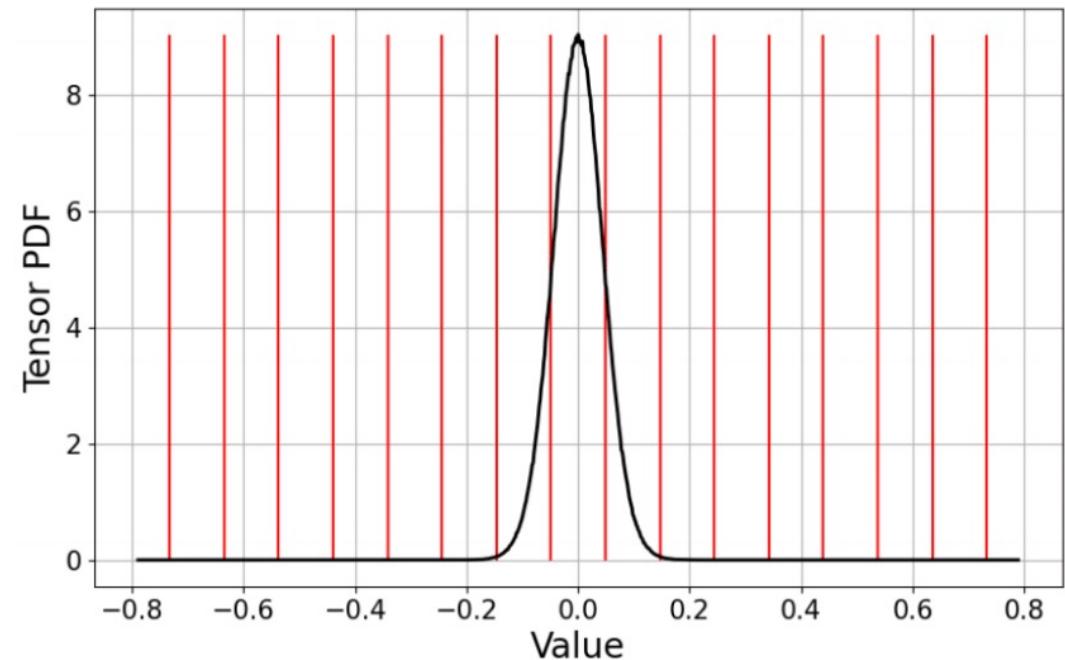
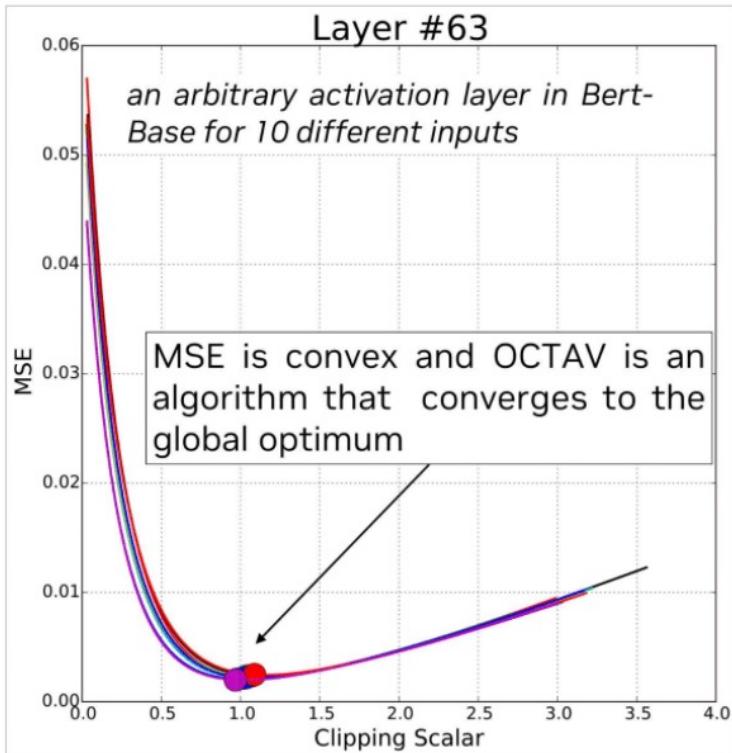


*clipped* quantization



# Dynamic Range for Quantization

Minimize mean-square-error (MSE) using Newton-Raphson method



Network	FP32 Accuracy	OCTAV int4
ResNet-50	76.07	75.84
MobileNet-V2	71.71	70.88
Bert-Large	91.00	87.09

# **Post-Training Quantization**

**How should we get the optimal linear quantization parameters ( $S$ ,  $Z$ )?**

Topic I: Quantization Granularity

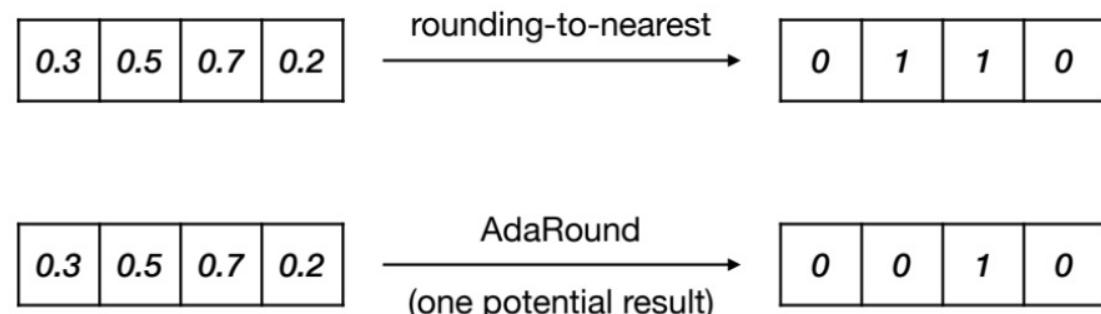
Topic II: Dynamic Range Clipping

**Topic III: Rounding**

# Adaptive Rounding for Weight Quantization

## Rounding-to-nearest is not optimal

- **Philosophy**
  - Rounding-to-nearest is not optimal
  - Weights are correlated with each other. The best rounding for each weight (to nearest) is not the best rounding for the whole tensor



- What is optimal? Rounding that reconstructs the original activation the best, which may be very different
  - For weight quantization only
  - With short-term tuning, (almost) post-training quantization

# Adaptive Rounding for Weight Quantization

Rounding-to-nearest is not optimal

- **Method:**
  - Instead of  $\lfloor w \rfloor$ , we want to choose from  $\{\lfloor w \rfloor, \lceil w \rceil\}$  to get the best reconstruction
  - We took a learning-based method to find quantized value  $\tilde{w} = \lfloor \lfloor w \rfloor + \delta \rceil, \delta \in [0,1]$

# Adaptive Rounding for Weight Quantization

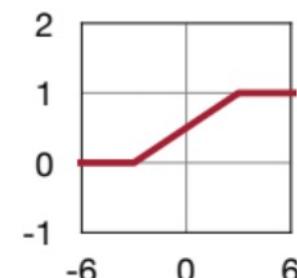
Rounding-to-nearest is not optimal

- Method:
  - Instead of  $\lfloor w \rfloor$ , we want to choose from  $\{ \lfloor w \rfloor, \lceil w \rceil \}$  to get the best reconstruction
  - We took a learning-based method to find quantized value  $\tilde{w} = \lfloor \lfloor w \rfloor + \delta \rceil, \delta \in [0,1]$
  - We optimize the following equation (omit the derivation):

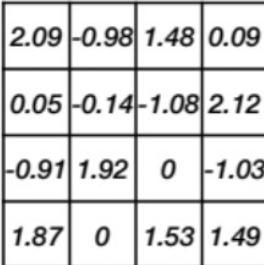
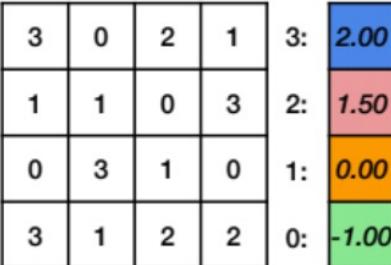
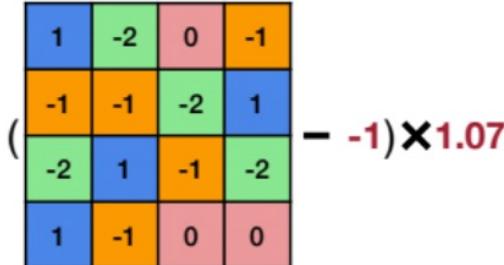
$$\begin{aligned} & \operatorname{argmin}_{\mathbf{V}} \|\mathbf{Wx} - \tilde{\mathbf{Wx}}\|_F^2 + \lambda f_{reg}(\mathbf{V}) \\ \rightarrow & \operatorname{argmin}_{\mathbf{V}} \|\mathbf{Wx} - \lfloor \lfloor \mathbf{W} \rfloor + \mathbf{h}(\mathbf{V}) \rceil \mathbf{x}\|_F^2 + \lambda f_{reg}(\mathbf{V}) \end{aligned}$$

- $\mathbf{x}$  is the input to the layer,  $\mathbf{V}$  is a random variable of the same shape
- $\mathbf{h}()$  is a function to map the range to  $(0,1)$ , such as rectified sigmoid
- $f_{reg}(\mathbf{V})$  is a regularization that encourages  $\mathbf{h}(\mathbf{V})$  to be binary

$$\cdot f_{reg}(\mathbf{V}) = \sum_{i,j} 1 - |2h(\mathbf{V}_{i,j}) - 1|^\beta$$



# Neural Network Quantization

	 <table border="1"><tr><td>2.09</td><td>-0.98</td><td>1.48</td><td>0.09</td></tr><tr><td>0.05</td><td>-0.14</td><td>-1.08</td><td>2.12</td></tr><tr><td>-0.91</td><td>1.92</td><td>0</td><td>-1.03</td></tr><tr><td>1.87</td><td>0</td><td>1.53</td><td>1.49</td></tr></table>	2.09	-0.98	1.48	0.09	0.05	-0.14	-1.08	2.12	-0.91	1.92	0	-1.03	1.87	0	1.53	1.49	 <table border="1"><tr><td>3</td><td>0</td><td>2</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td><td>3</td></tr><tr><td>0</td><td>3</td><td>1</td><td>0</td></tr><tr><td>3</td><td>1</td><td>2</td><td>2</td></tr></table> <p>3: 2.00 2: 1.50 1: 0.00 0: -1.00</p>	3	0	2	1	1	1	0	3	0	3	1	0	3	1	2	2	 <table border="1"><tr><td>1</td><td>-2</td><td>0</td><td>-1</td></tr><tr><td>-1</td><td>-1</td><td>-2</td><td>1</td></tr><tr><td>-2</td><td>1</td><td>-1</td><td>-2</td></tr><tr><td>1</td><td>-1</td><td>0</td><td>0</td></tr></table> <p>( -1) × 1.07</p>	1	-2	0	-1	-1	-1	-2	1	-2	1	-1	-2	1	-1	0	0
2.09	-0.98	1.48	0.09																																																
0.05	-0.14	-1.08	2.12																																																
-0.91	1.92	0	-1.03																																																
1.87	0	1.53	1.49																																																
3	0	2	1																																																
1	1	0	3																																																
0	3	1	0																																																
3	1	2	2																																																
1	-2	0	-1																																																
-1	-1	-2	1																																																
-2	1	-1	-2																																																
1	-1	0	0																																																
Storage	Floating-Point Weights	Integer Weights; Floating-Point Codebook	Integer Weights																																																
Computation	Floating-Point Arithmetic	Floating-Point Arithmetic	Integer Arithmetic																																																

- **Zero Point**
  - Asymmetric
  - Symmetric
- **Scaling Granularity**
  - Per-Tensor
  - Per-Channel
  - Group Quantization
- **Range Clipping**
  - Exponential Moving Average
  - Minimizing KL Divergence
  - Minimizing Mean-Square-Error
- **Rounding**
  - Round-to-Nearest
  - AdaRound

# Post-Training INT8 Linear Quantization

		Symmetric	Asymmetric
		Per-Tensor	Per-Tensor
Activation		Minimize KL-Divergence	Exponential Moving Average (EMA)
Neural Network	Weight	Symmetric	Symmetric
		Per-Tensor	Per-Channel
	GoogleNet	-0.45%	0%
	ResNet-50	-0.13%	-0.6%
	ResNet-152	-0.08%	-1.8%
	MobileNetV1	-	-11.8%
	MobileNetV2	-	-2.1%

Data-Free Quantization Through Weight Equalization and Bias Correction [Markus et al., ICCV 2019]  
Quantizing Deep Convolutional Networks for Efficient Inference: A Whitepaper [Raghuraman Krishnamoorthi, arXiv 2018]  
8-bit Inference with TensorRT [Szymon Migacz, 2017]

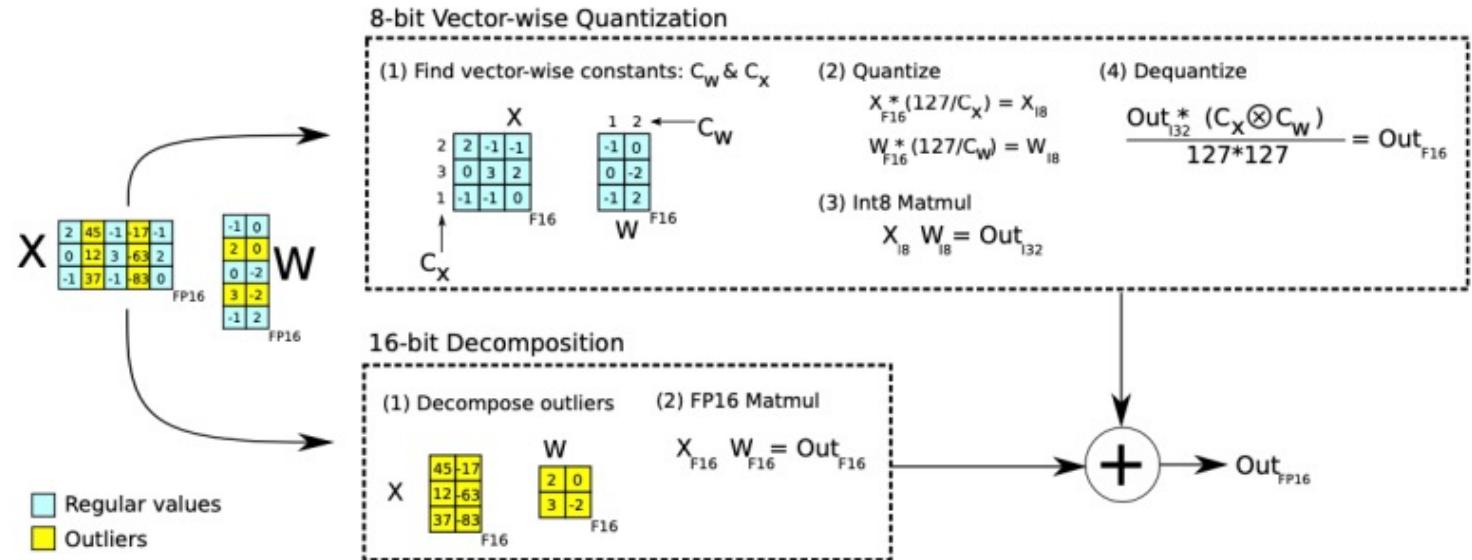
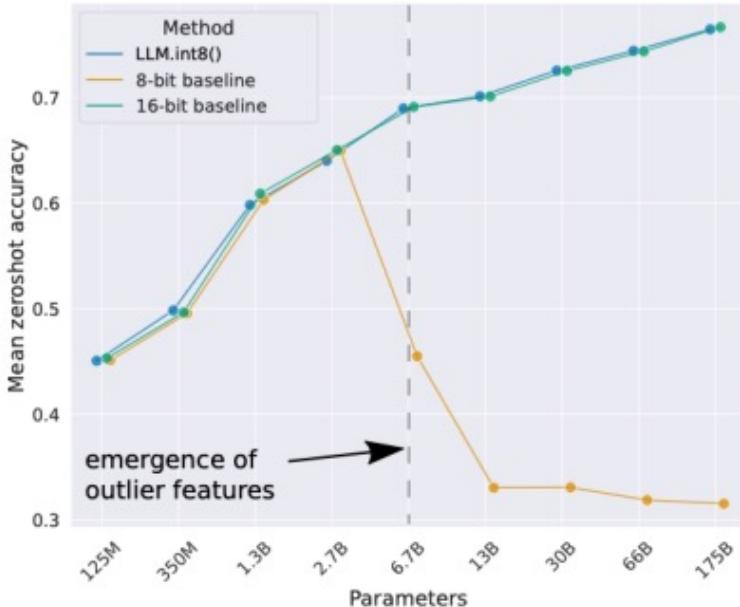
# Post-Training INT8 Linear Quantization

		Symmetric	Asymmetric
		Per-Tensor	Per-Tensor
Activation		Minimize KL-Divergence	Exponential Moving Average (EMA)
Weight	Symmetric	Symmetric	Symmetric
	Per-Tensor	Per-Tensor	Per-Channel
Neural Network	Smaller models seem to not respond as well to post-training quantization, presumably due to their smaller representational capacity.	How should we improve performance of quantized models?	
	MobileNetV1	-	-11.8%
	MobileNetV2	-	-2.1%

Data-Free Quantization Through Weight Equalization and Bias Correction [Markus et al., ICCV 2019]  
Quantizing Deep Convolutional Networks for Efficient Inference: A Whitepaper [Raghuraman Krishnamoorthi, arXiv 2018]  
8-bit Inference with TensorRT [Szymon Migacz, 2017]

# Quantization – LLM.int8()

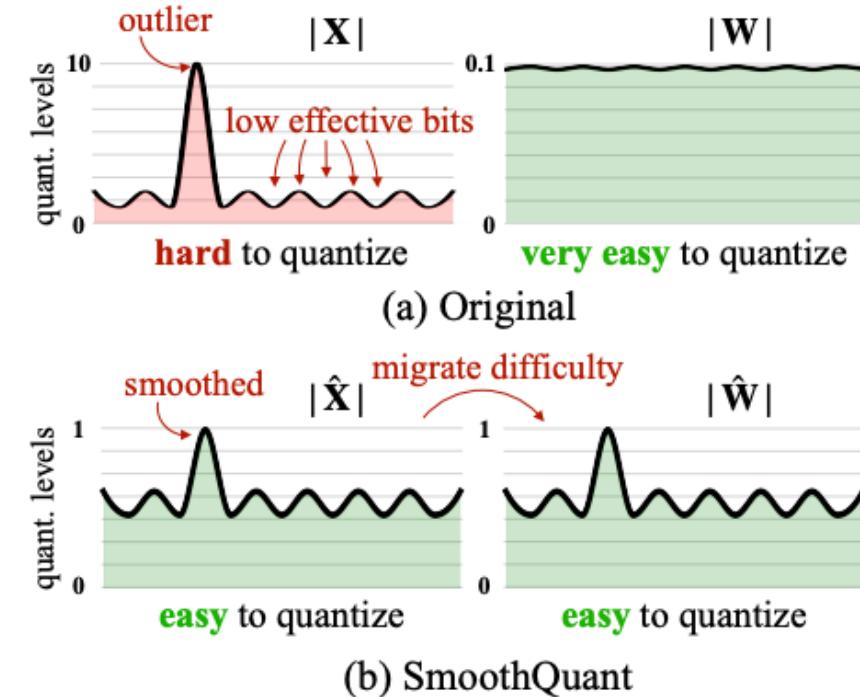
## Mixed-Precision Decomposition



- Motivation:** Transformers have outlier features that have **large values** (especially large models).
  - They occur in particular hidden dimensions, leading to large quantization error.
- Key idea:** Separate outlier features into a **separate FP16 MM**, quantize the other values to Int8.
  - Outlier: At least one feature dimension with a magnitude larger than the threshold (6).
  - Token-wise scale factor (for X) and (output) channel-wise scale factor (for W).

# SmoothQuant: Accurate and Efficient Post-Training Quantization for LLMs

	LLM (100B+) Accuracy	Hardware Efficiency
ZeroQuant	✗	✓
Outlier Suppression	✗	✓
LLM.int8()	✓	✗
<b>SmoothQuant</b>	✓	✓



- We propose SmoothQuant, an **accurate** and **efficient** post-training-quantization (PTQ) method to enable 8-bit weight, 8-bit activation (**W8A8**) quantization for LLMs.
- Since **weights are easy to quantize while activations are not**, SmoothQuant smooths the activation outliers by **migrating the quantization difficulty from activations to weights** with a mathematically equivalent transformation.

