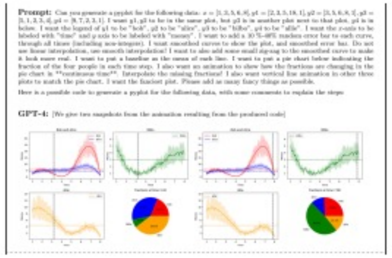
LLM Quantization

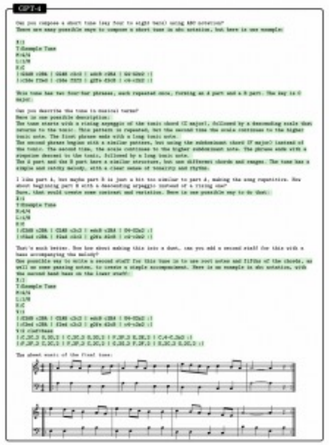
Xin Yu 2023.11.01

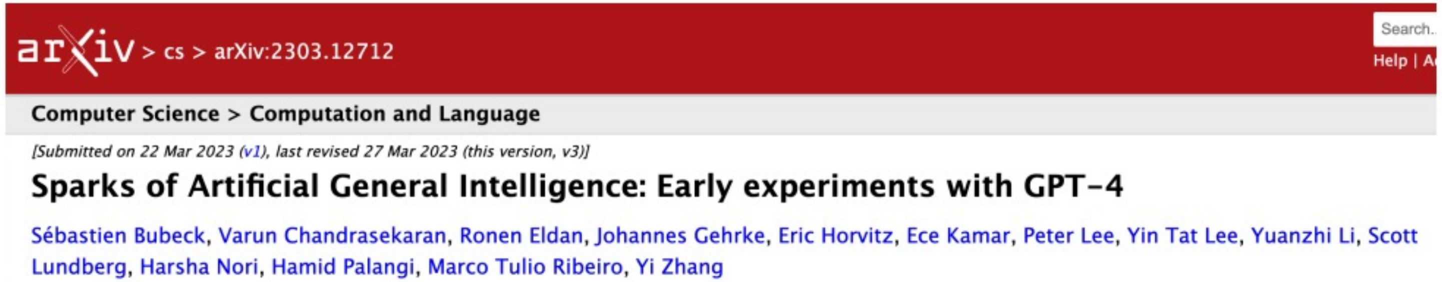
GPT-4

Write python codes



Compose music





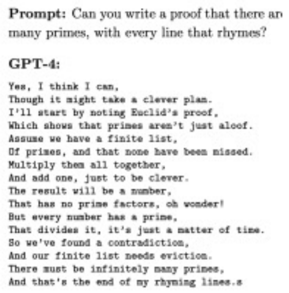
Computer Science > Computation and Language

[Submitted on 22 Mar 2023 (v1), last revised 27 Mar 2023 (this version, v3)]

Sparks of Artificial General Intelligence: Early experiments with GPT-4

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, Yi Zhang

Write poems

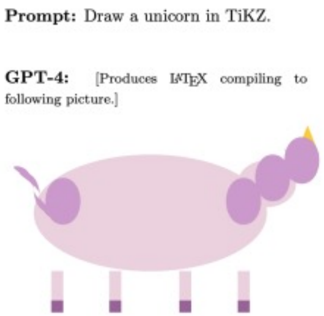


Prompt: Can you write a proof that there ar many primes, with every line that rhymes?

GPT-4:

```
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 none 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,
That divides it, 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,
And that's the end of my rhyming lines.s
```

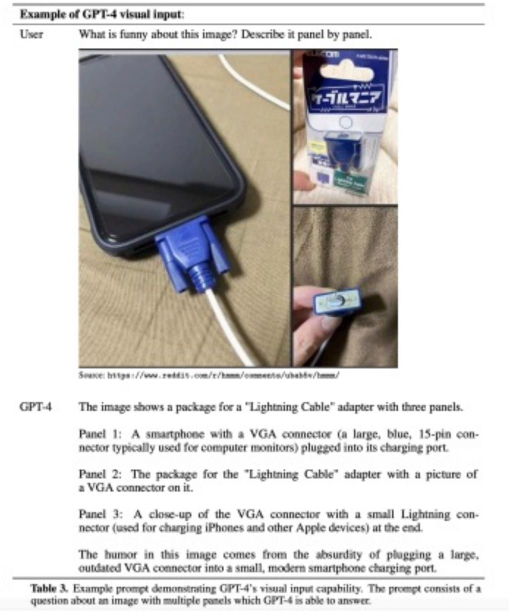
Write LateX codes



Prompt: Draw a unicorn in TiKZ.

GPT-4: [Produces LaTeX compiling to following picture.]

Explain jokes



Example of GPT-4 visual input:

User What is funny about this image? Describe it panel by panel.

Source: https://www.reddit.com/r/hmmm/comments/ubab5v/hmmm/

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.

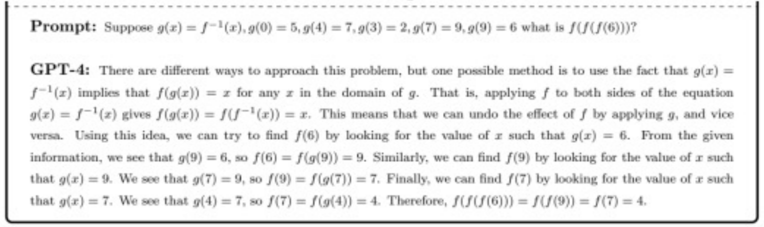
Panel 2: The package for the "Lightning Cable" adapter with a picture of a VGA connector on it.

Panel 3: A close-up of the VGA connector with a small Lightning connector (used for charging iPhones and other Apple devices) at the end.

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.

Do math problems

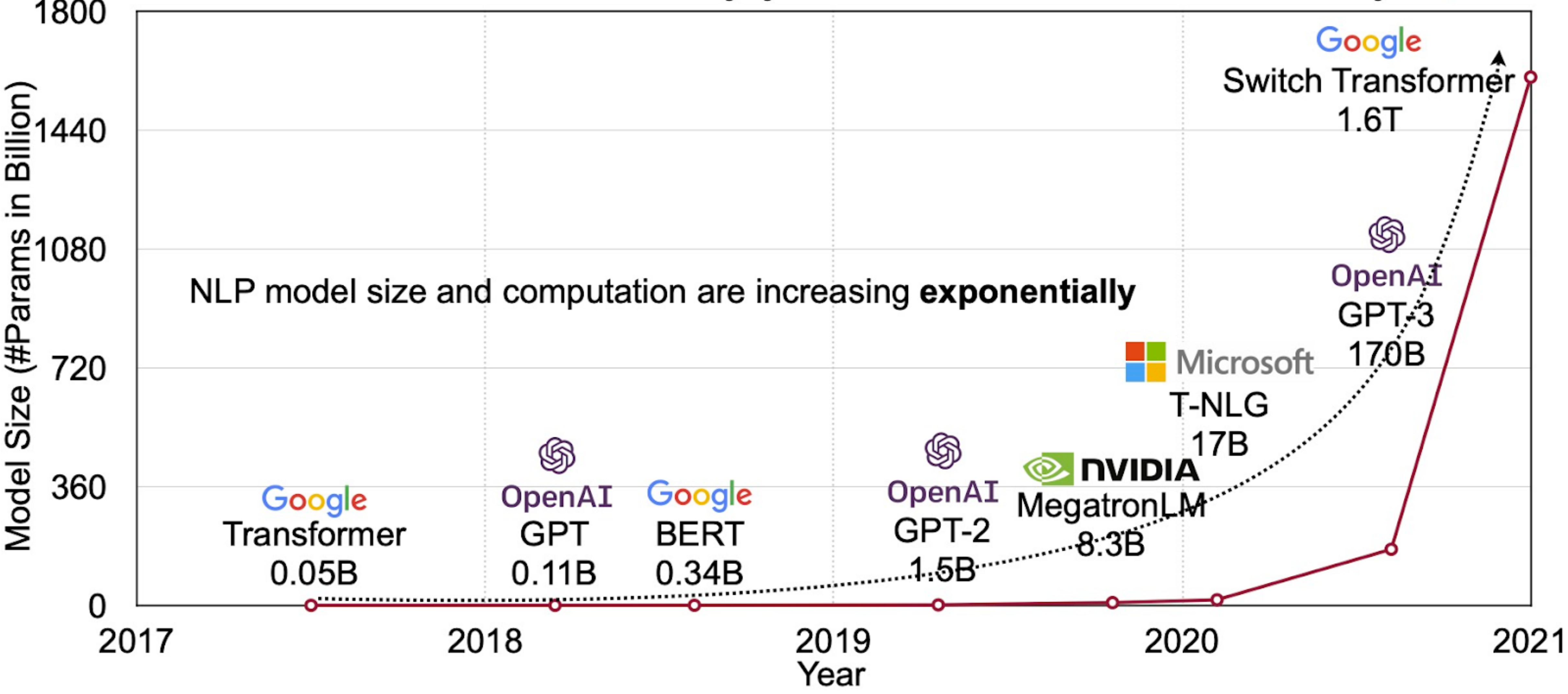


Prompt: Suppose $g(x) = f^{-1}(x), g(0) = 5, g(4) = 7, g(3) = 2, g(7) = 9, g(9) = 6$ what is $f(f(f(6)))$?

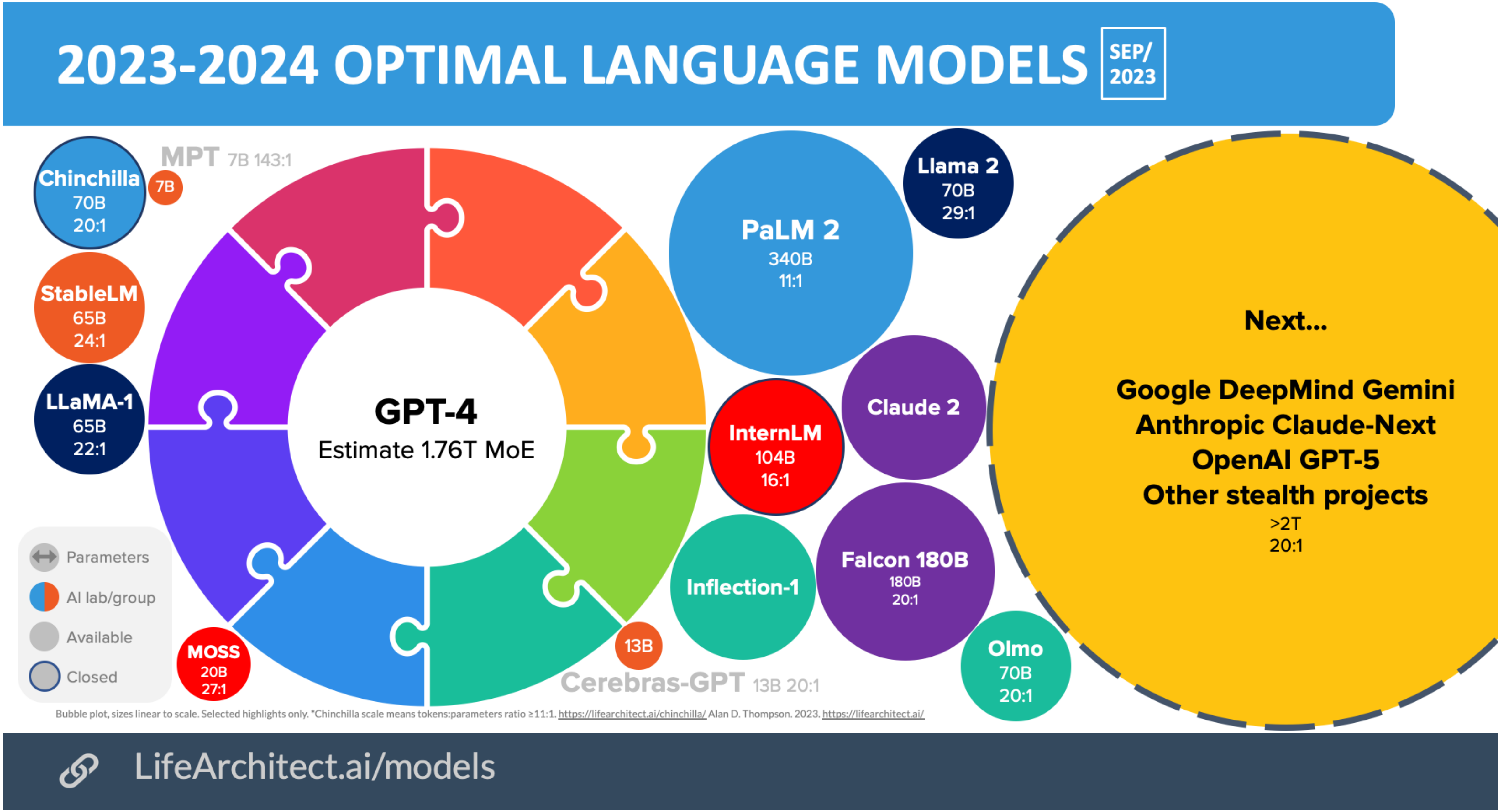
GPT-4: There are different ways to approach this problem, but one possible method is to use the fact that $g(x) = f^{-1}(x)$ implies that $f(g(x)) = x$ for any x in the domain of g. That is, applying f to both sides of the equation $g(x) = f^{-1}(x)$ gives $f(g(x)) = f(f^{-1}(x)) = x$. This means that 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 $f(9) = f(g(7)) = 7$. 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, $f(f(f(6))) = f(f(9)) = f(7) = 4$.

Ref: https://qiniu.pattern.swarma.org/attachment/LLM_science_swarm.pdf

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

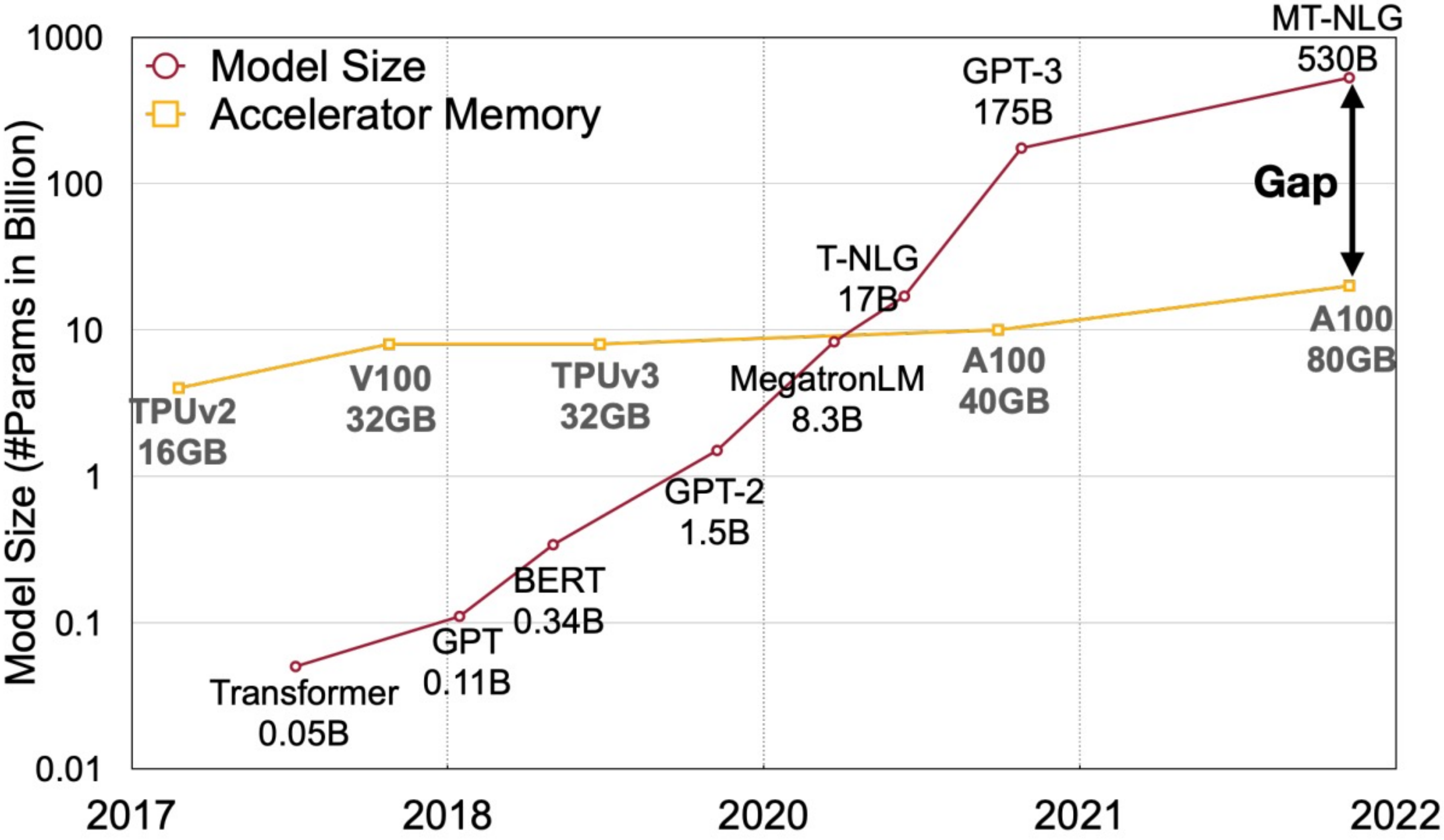


Ref: https://indiaai.gov.in/article/the-future-of-large-language-models-llms-strategy-opportunities-and-challenges



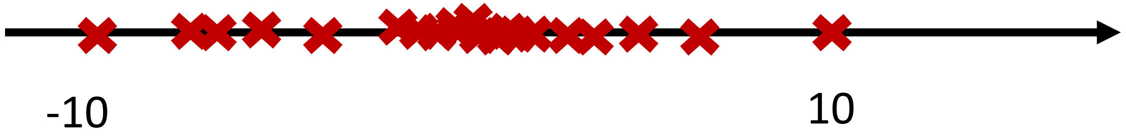
Bubble plot, sizes linear to scale. Selected highlights only. *Chinchilla scale means tokens:parameters ratio ≥11:1. https://lifearchitect.ai/chinchilla/ Alan D. Thompson. 2023. https://lifearchitect.ai/

Ref: https://lifearchitect.ai/models/



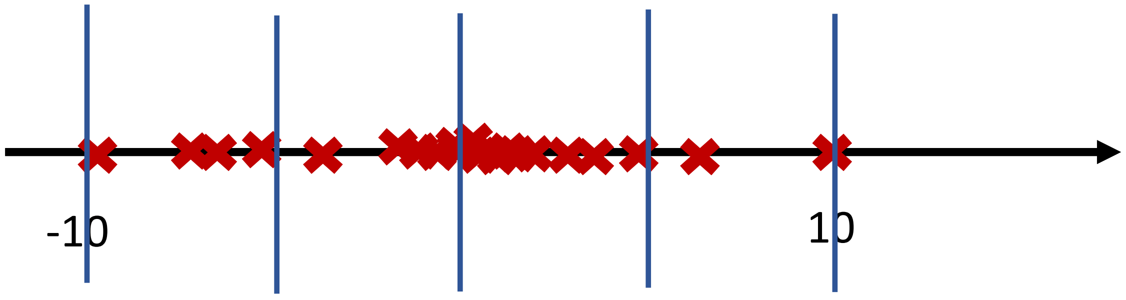
Ref: https://lifearchitect.ai/models/

Quantization: Data with less bits



Quantization: Data with less bits

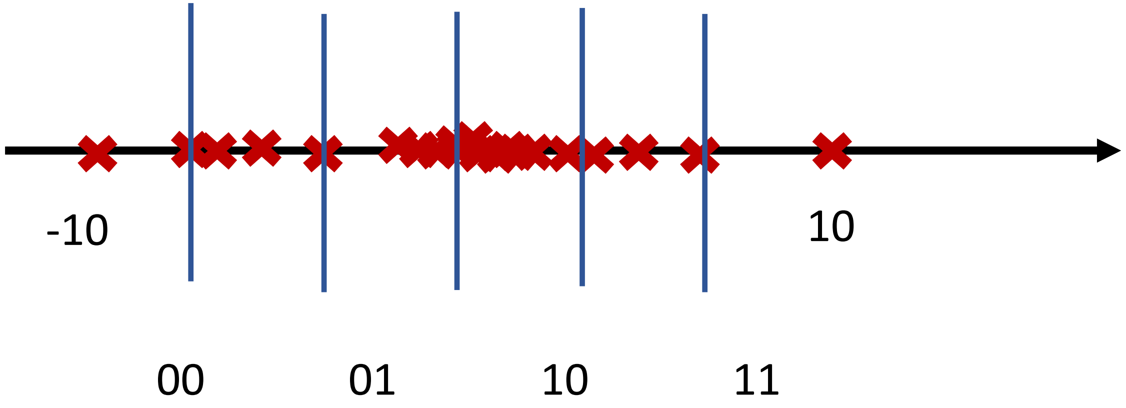
RTN: round to nearest



00 01 10 11

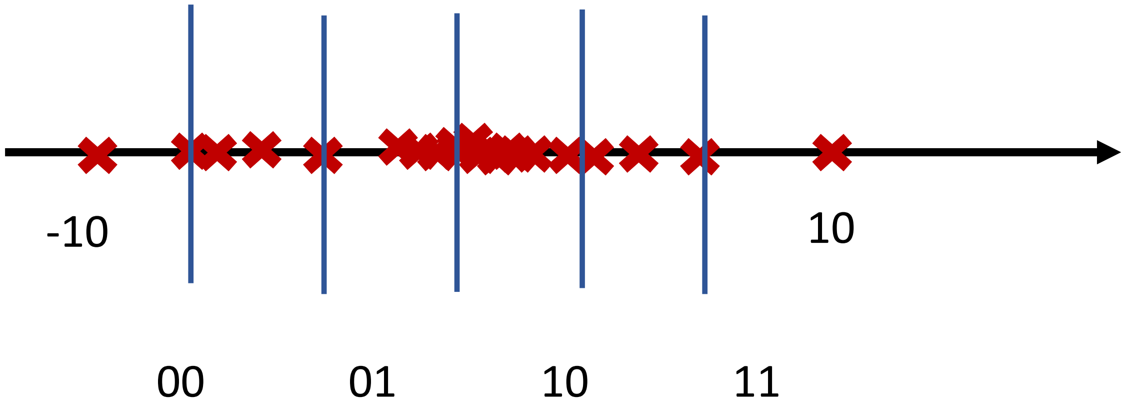
Quantization: Data with less bits

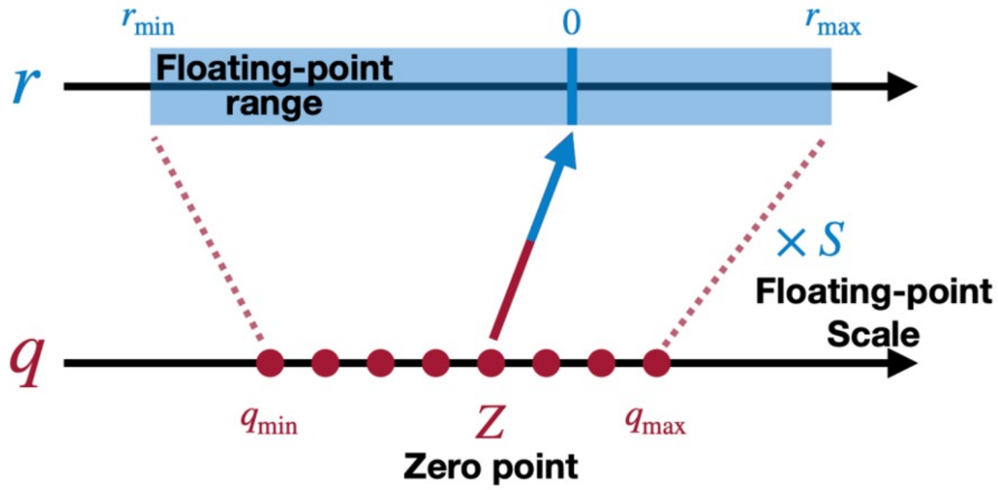
RTN: round to nearest



Quantization: Data with less bits

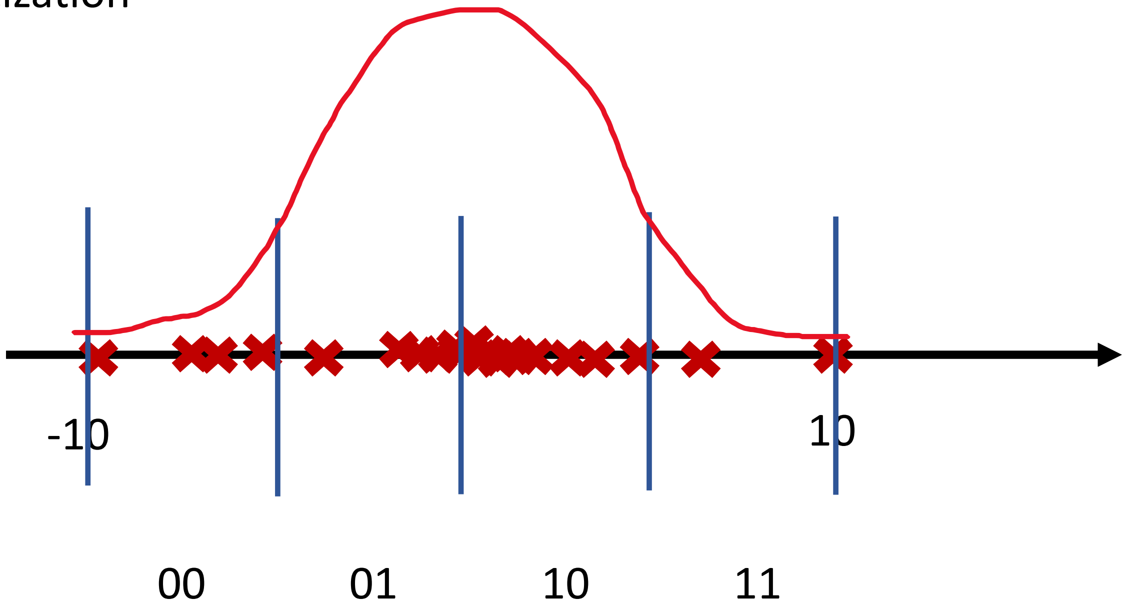
RTN: round to nearest





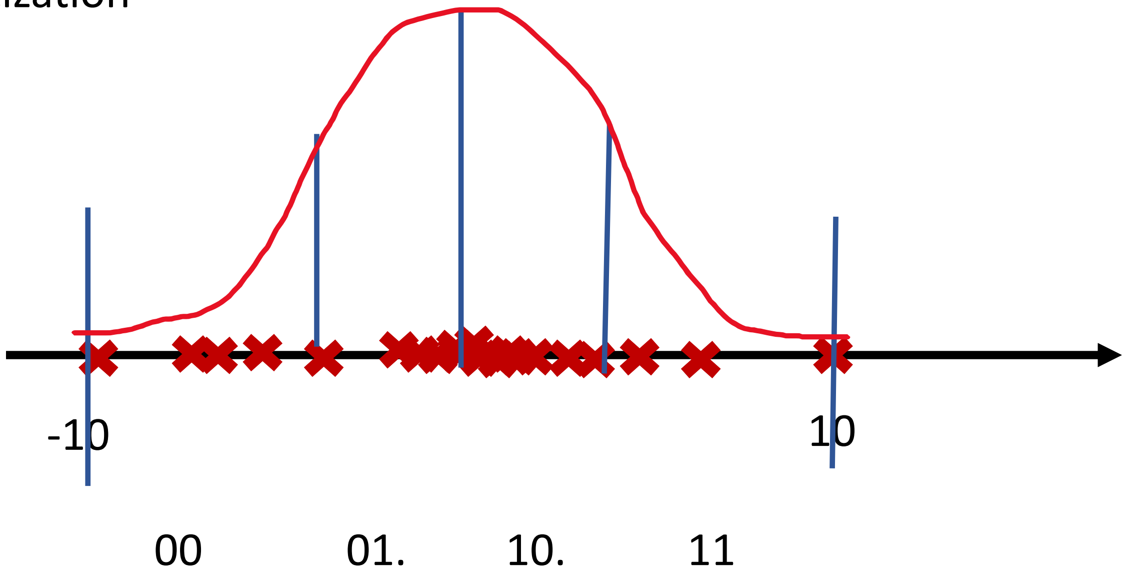
Quantization: Data with less bits

Normal Float Quantization



Quantization: Data with less bits

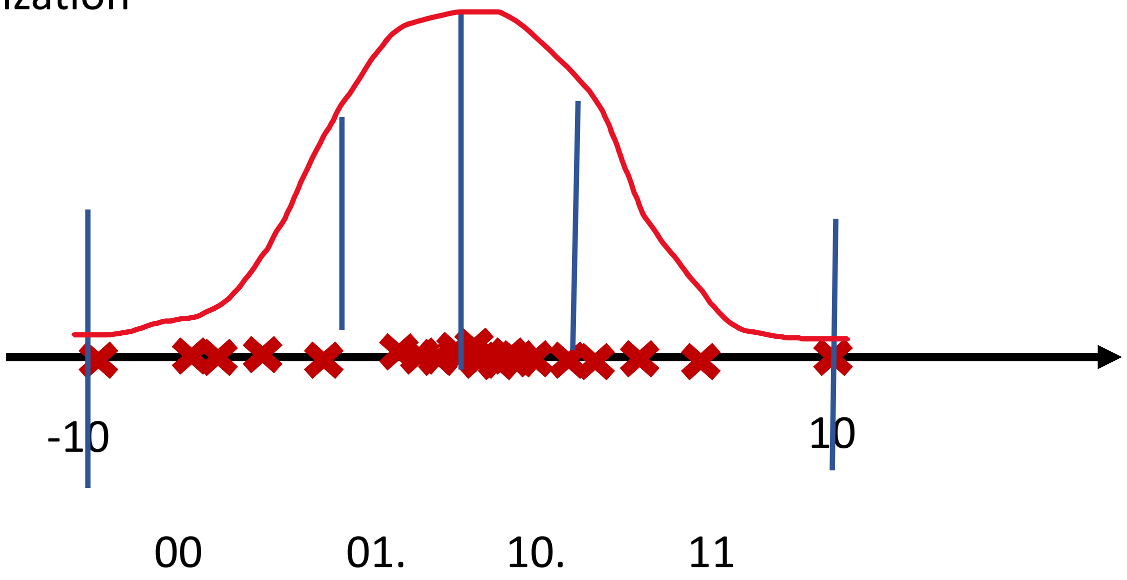
Normal Float Quantization



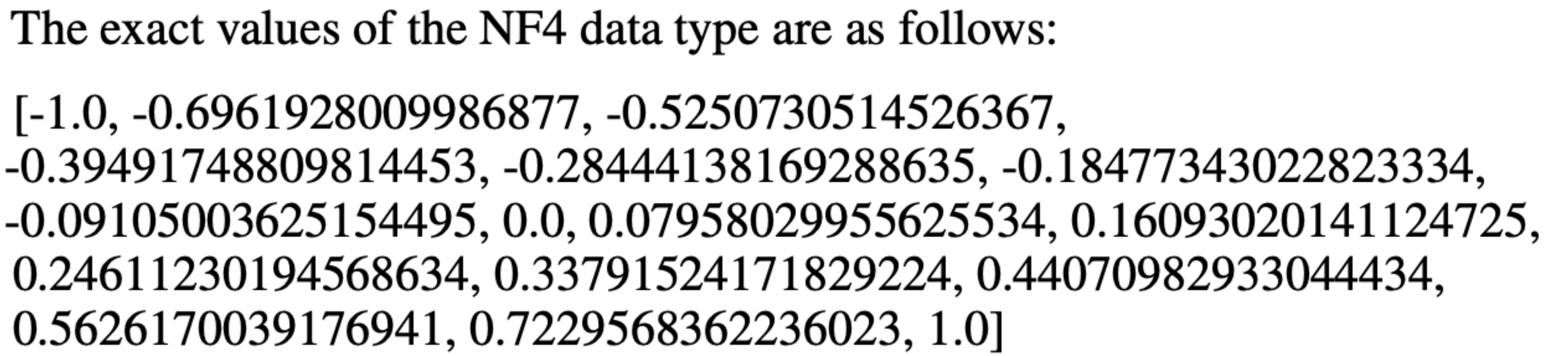
Quantile Quantization

Quantization: Data with less bits

Normal Float Quantization



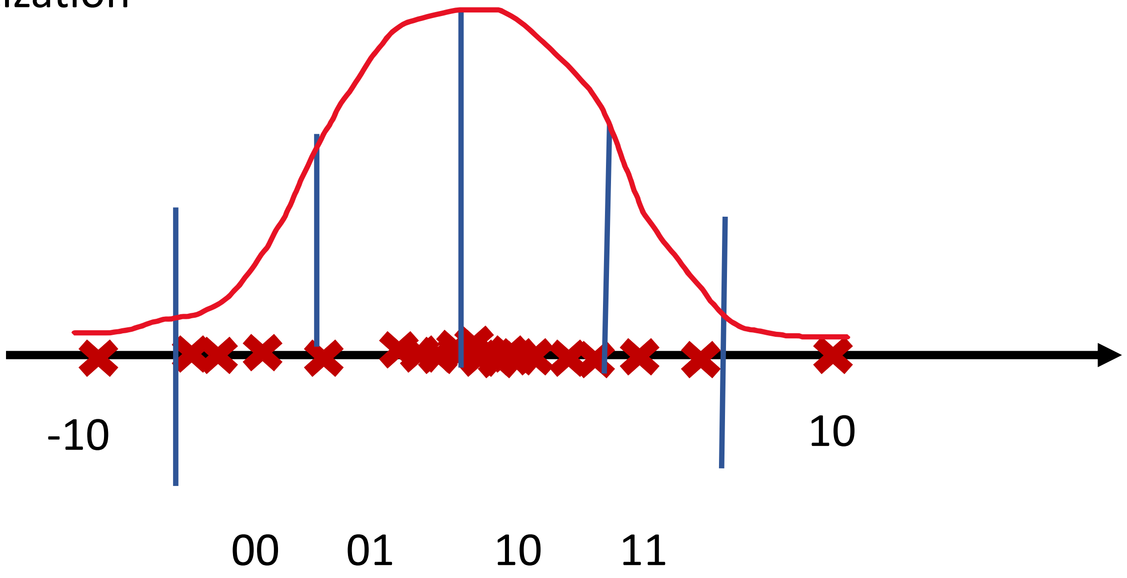
Quantile Quantization


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]

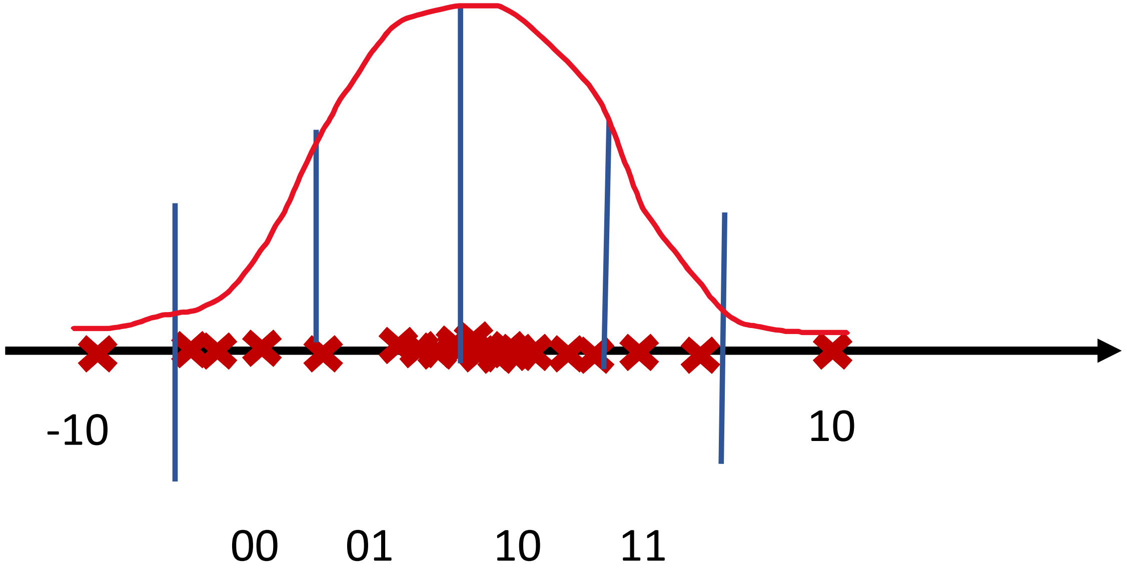
Quantization: Data with less bits

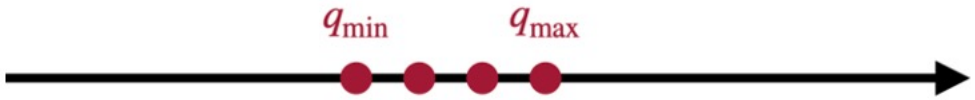
Normal Float Quantization

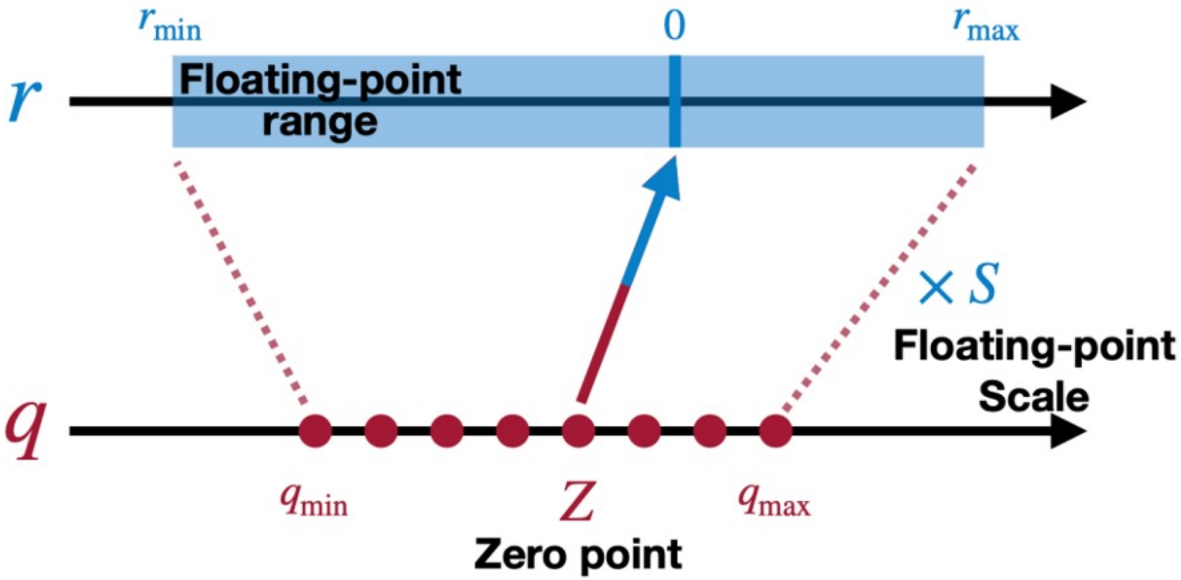


Truncate the outliers!

Quantization: Data with less bits



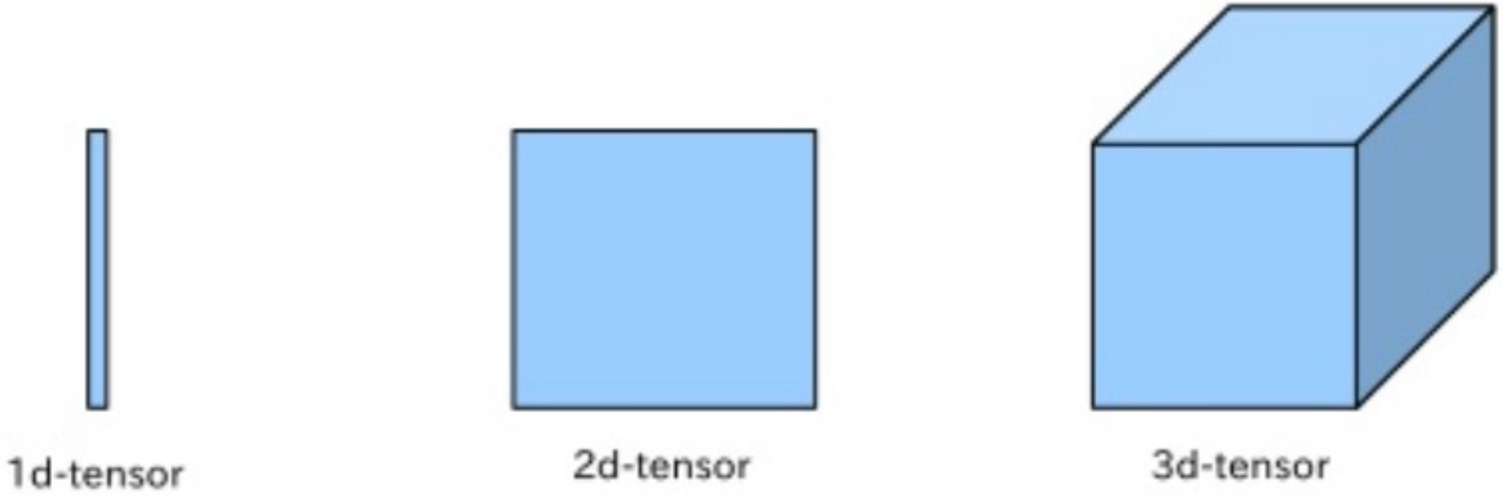




Binary	Decimal
01	1
00	0
11	-1
10	-2

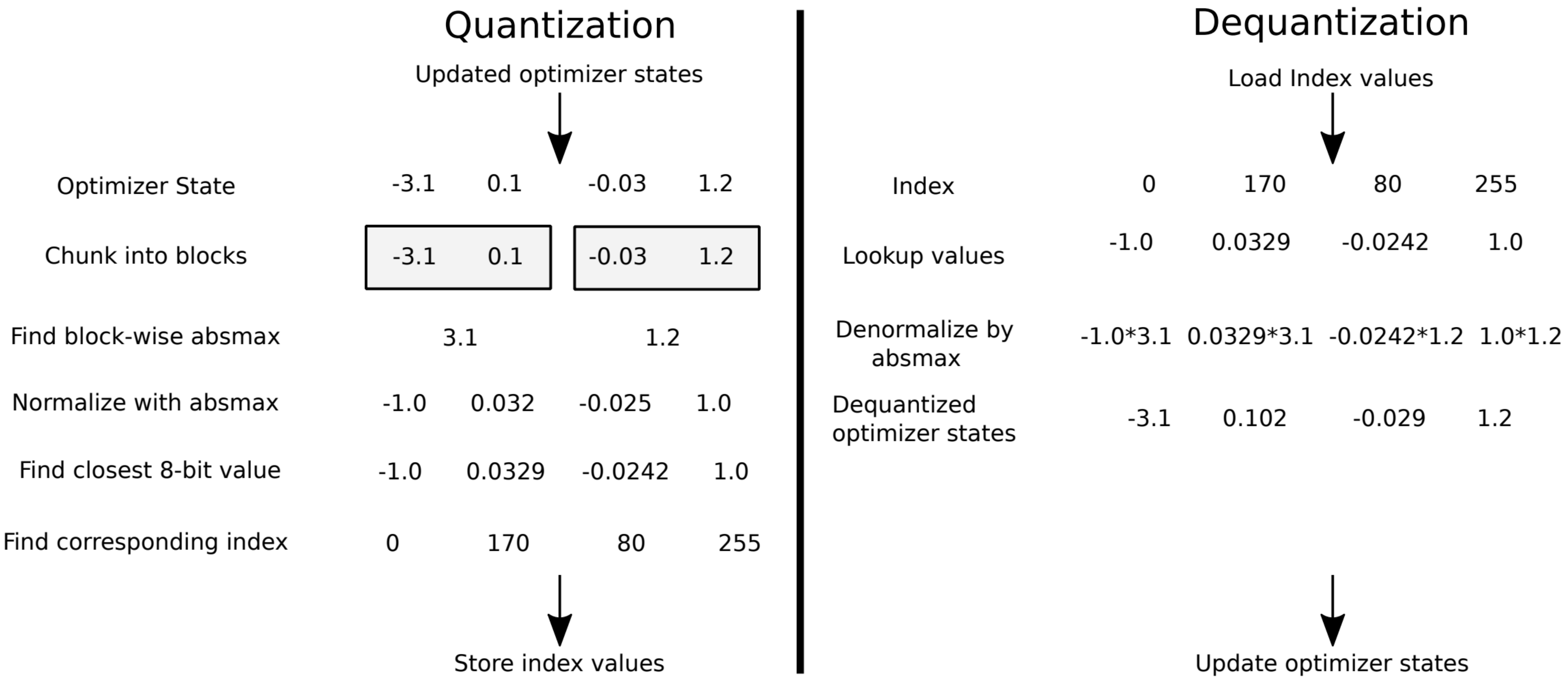
Quantization: Data with less bits

Block-wise dynamic quantization



Quantization: Data with less bits

Block-wise dynamic quantization



Quantization

Updated optimizer states

Optimizer State	-3.1	0.1	-0.03	1.2
Chunk into blocks	-3.1	0.1	-0.03	1.2
Find block-wise absmax	3.1		1.2	
Normalize with absmax	-1.0	0.032	-0.025	1.0
Find closest 8-bit value	-1.0	0.0329	-0.0242	1.0
Find corresponding index	0	170	80	255

Store index values

Dequantization

Load Index values

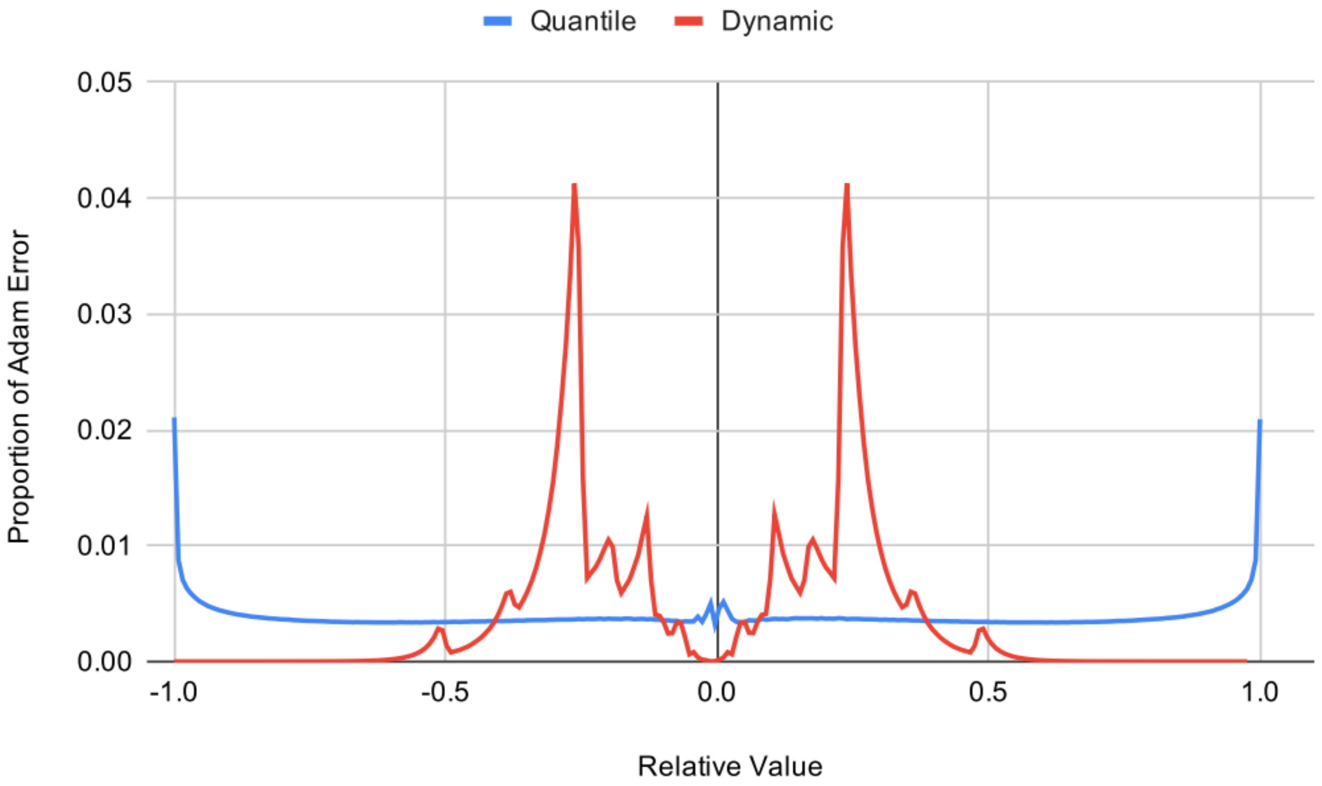
Index	0	170	80	255
Lookup values	-1.0	0.0329	-0.0242	1.0
Denormalize by absmax	-1.0*3.1	0.0329*3.1	-0.0242*1.2	1.0*1.2
Dequantized optimizer states	-3.1	0.102	-0.029	1.2

Update optimizer states

Ref: 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION

Quantization: Data with less bits

Block-wise dynamic quantization

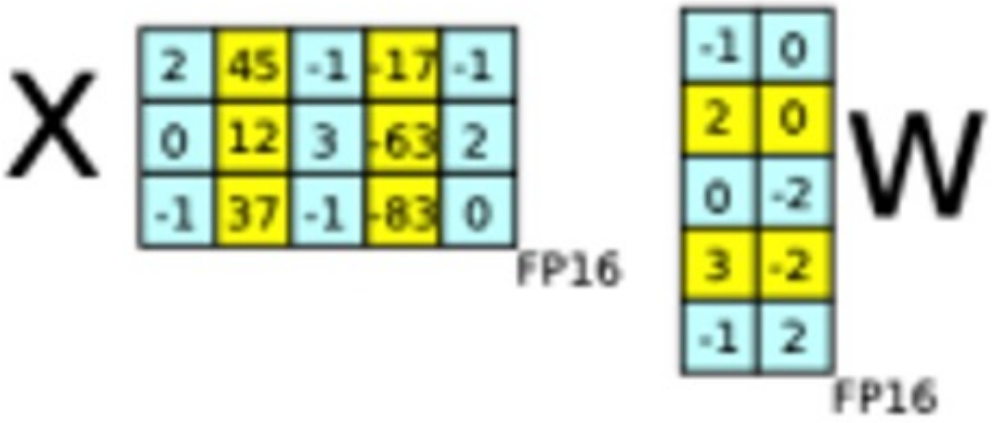


Ref: 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION

Minimize quantization error of each layer!

Reduce the accumulated error cross the network

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



X

2	45	-1	-17	-1
0	12	3	-63	2
-1	37	-1	-83	0

FP16

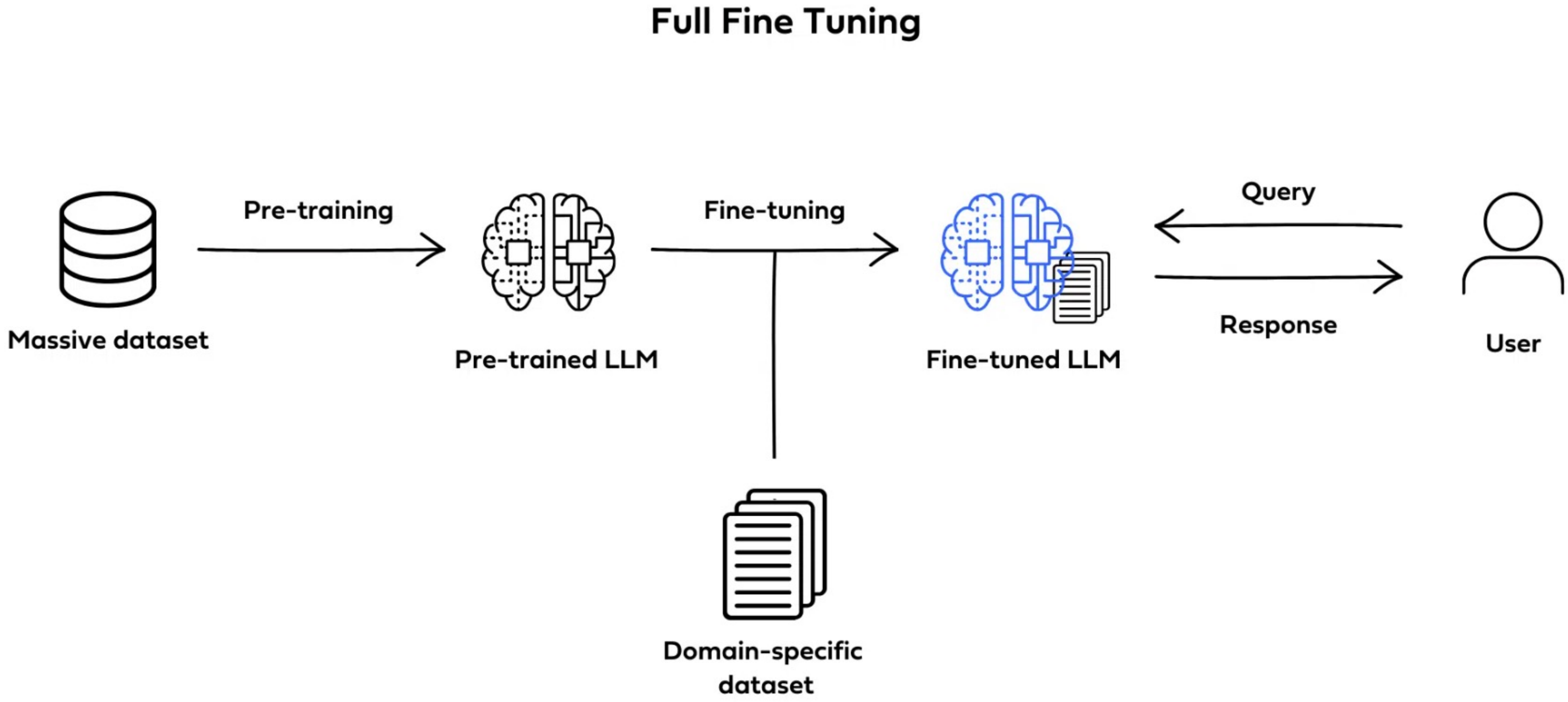
W

-1	0
2	0
0	-2
3	-2
-1	2

FP16

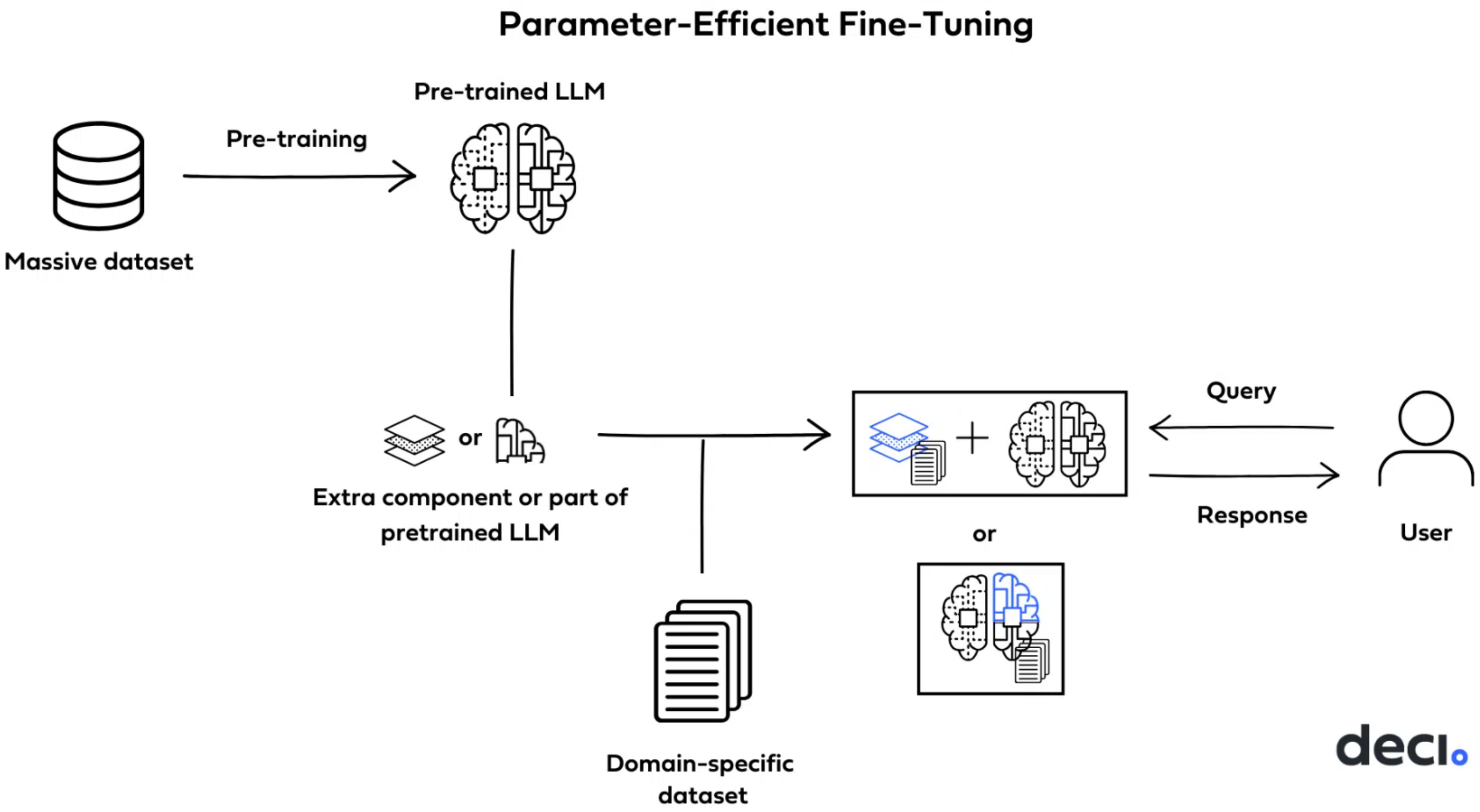
Method-1 **QLORA: Efficient Finetuning of Quantized LLMs**

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

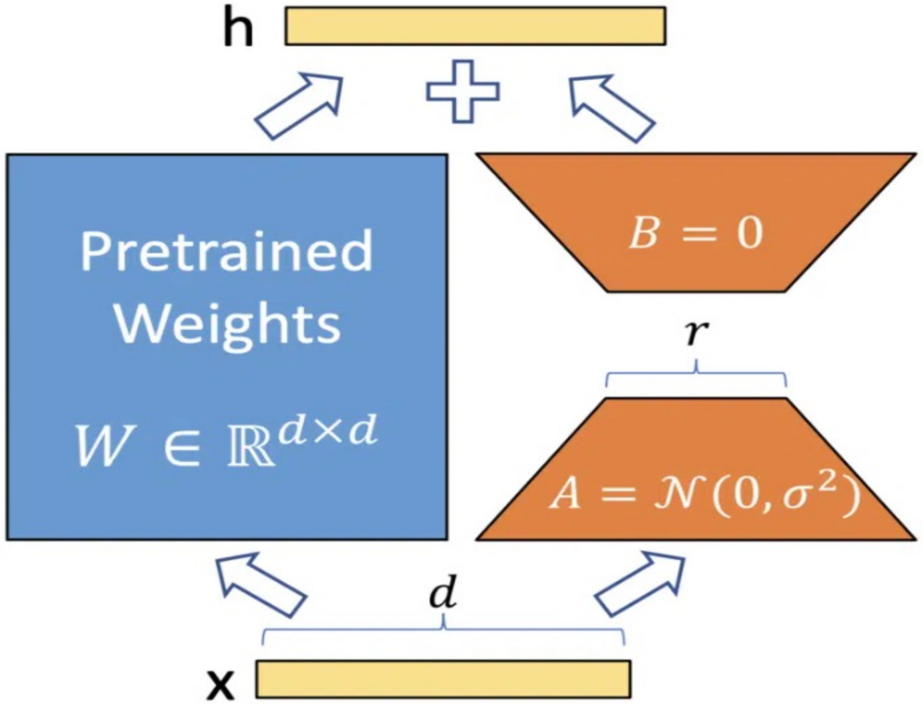

h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

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

W’ = W + **A @ B**

Method-1 **QLORA: Efficient Finetuning of Quantized LLMs**

QLoRA: the most commonly used PEFT Quantization

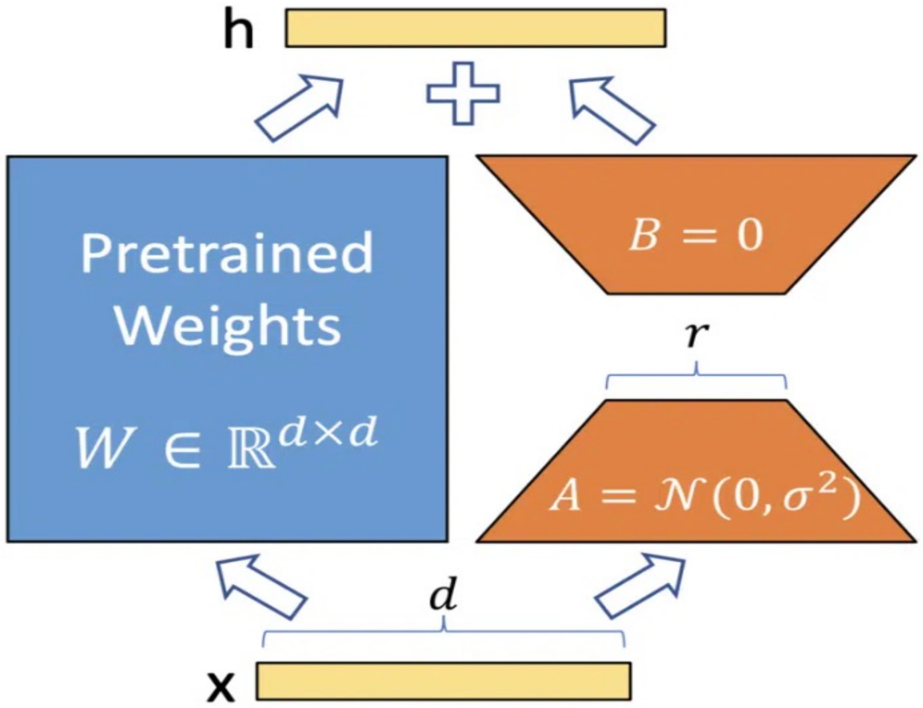

h
Pretrained
Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

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

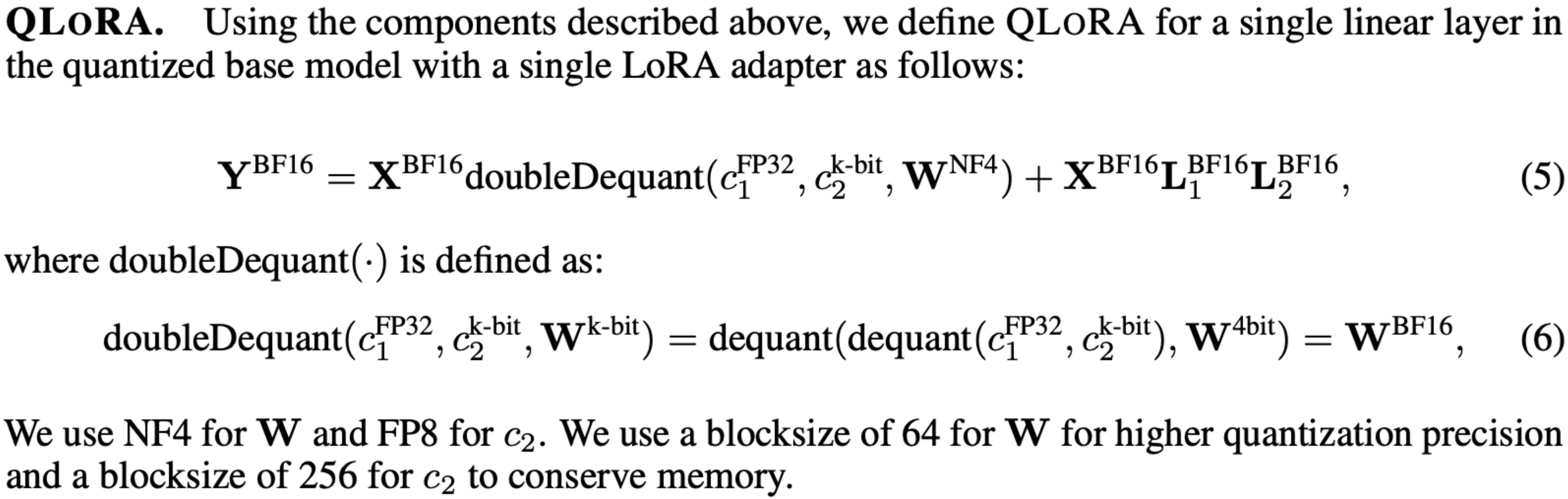
$$W' = \text{quant}(W) + \mathbf{A} \mathbin{@} \mathbf{B}$$

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

W’ = double_quantize(W) + **A @ B**

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

W’ = double_normal_float(W) + **A @ 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:

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

where 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.

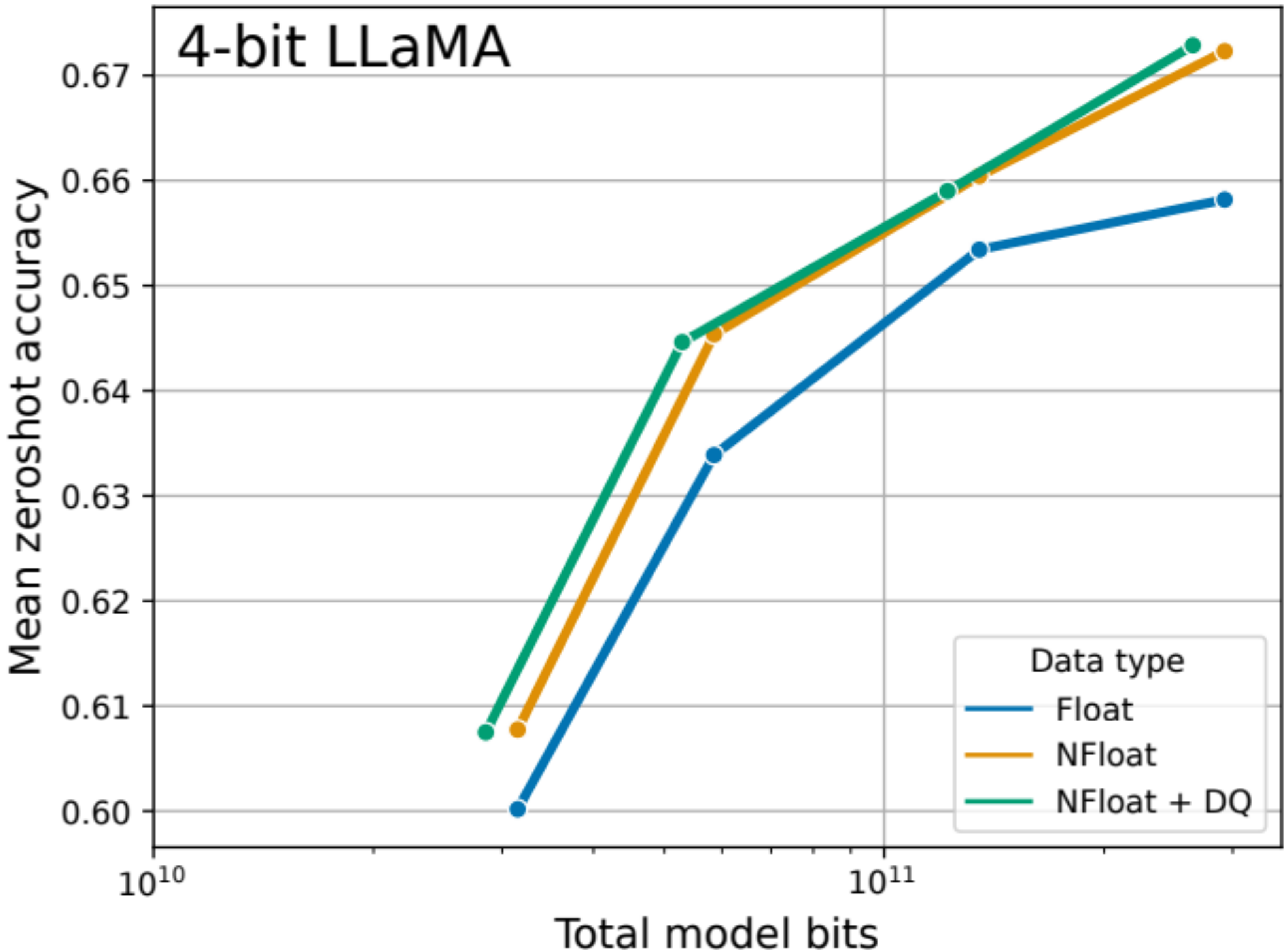
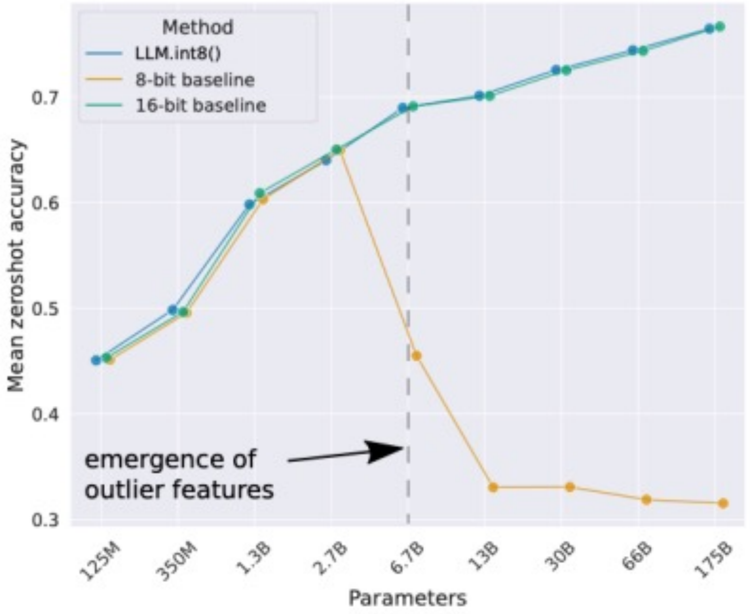

4-bit LLaMA
Mean zeroshot accuracy
0.60
0.61
0.62
0.63
0.64
0.65
0.66
0.67
Total model bits
10^{10}
10^{11}
Data type
Float
NFloat
NFloat + DQ

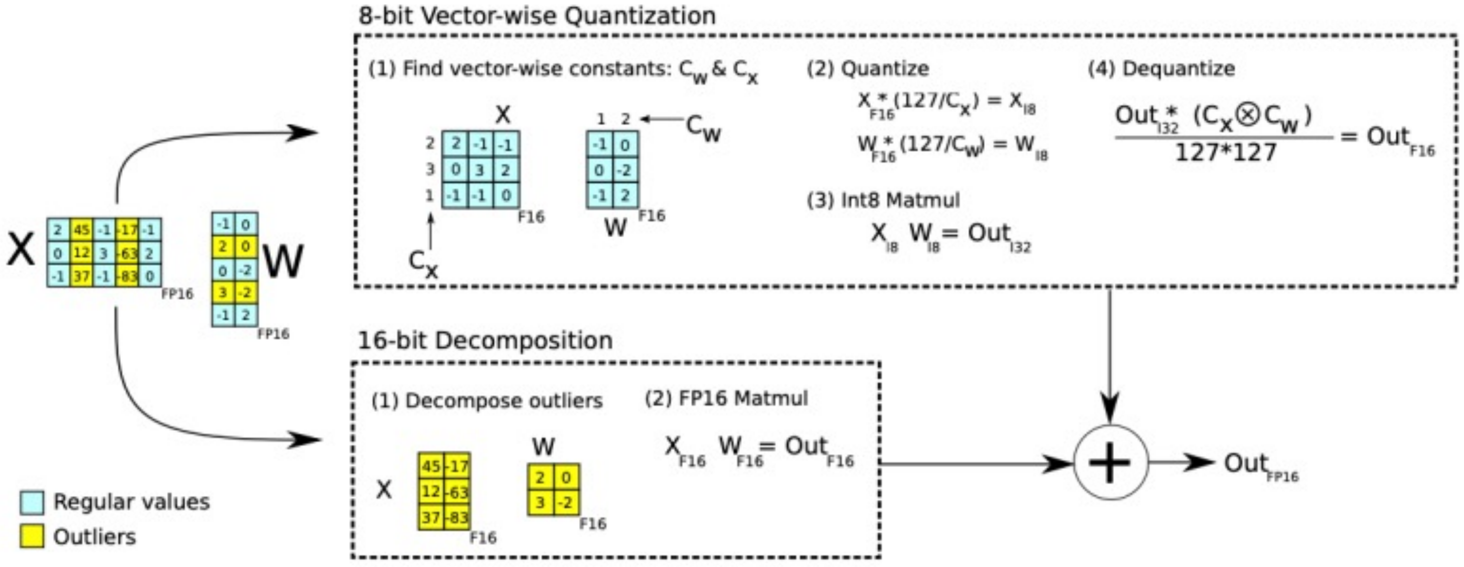
Table 3: Experiments comparing 16-bit BrainFloat (BF16), 8-bit Integer (Int8), 4-bit Float (FP4), and 4-bit NormalFloat (NF4) on GLUE and Super-NaturalInstructions. QLORA replicates 16-bit LoRA and full-finetuning.

Dataset	GLUE (Acc.)	Super-NaturalInstructions (RougeL)				
Model	RoBERTa-large	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

Quantization — LLM.int8()

Mixed-Precision Decomposition



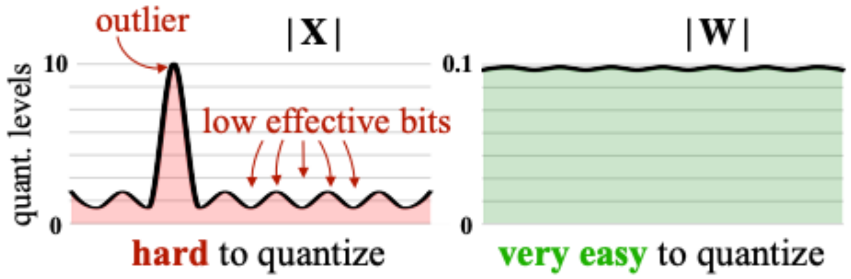


- **Motivation**: Transformers have outlier features that have **large values** (especially large models).
  - They occur in particular hidden dimensions, leading to large quantization error.
- **Key idea**: Separate outlier features into a **separate FP16 MM**, quantize the other values to Int8.
  - 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).

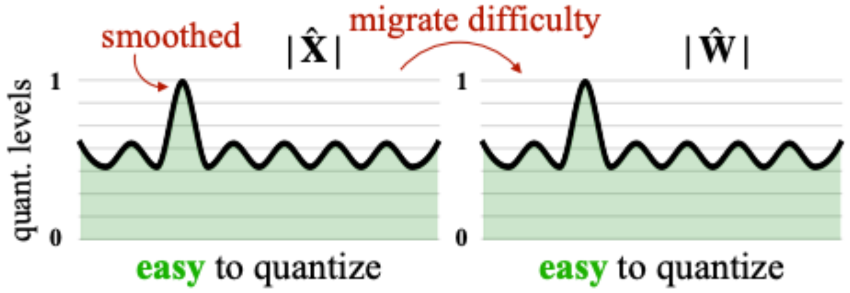
LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale [Dettmers *et al.*, 2022]

SmoothQuant: Accurate and Efficient Post-Training Quantization for LLMs

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

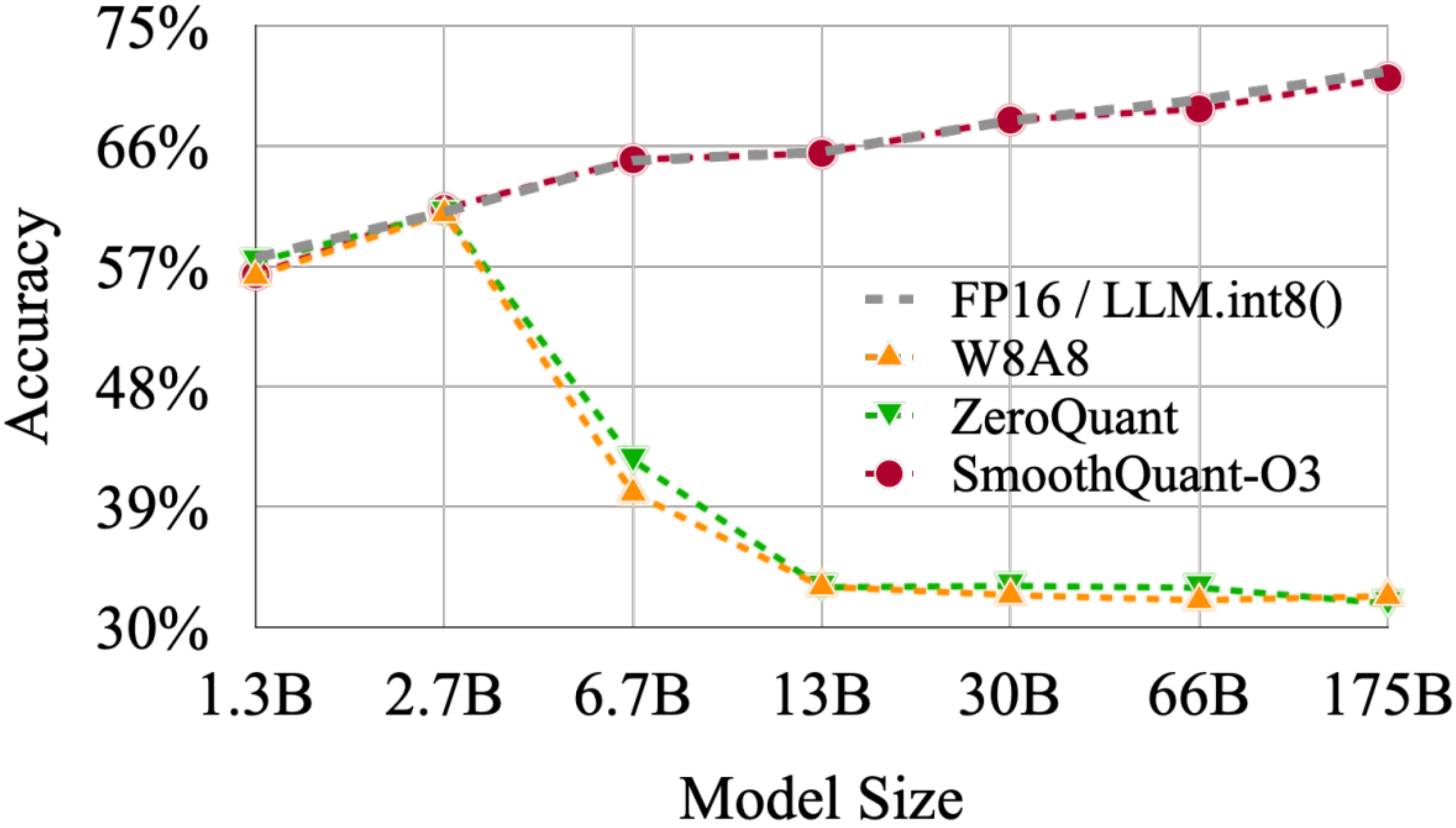


(a) Original



(b) SmoothQuant

- 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.


75%
66%
57%
48%
39%
30%
Accuracy
1.3B
2.7B
6.7B
13B
30B
66B
175B
Model Size
FP16 / LLM.int8()
W8A8
ZeroQuant
SmoothQuant-O3