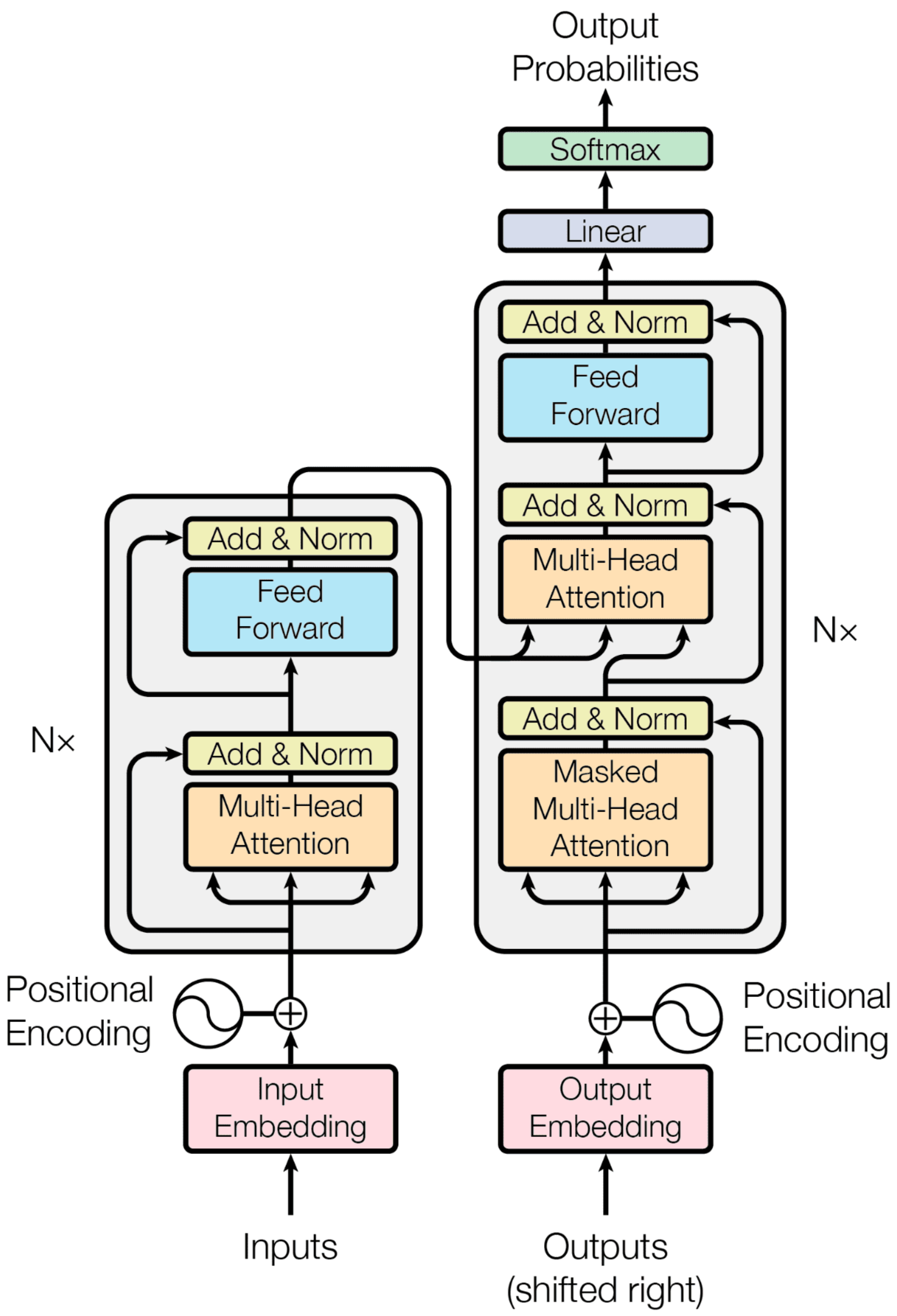
SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models

Guangxuan Xiao[*,1], Ji Lin[*,1],
Mickael Seznec[2], Hao Wu[2], Julien Demouth[2], Song Han[1]

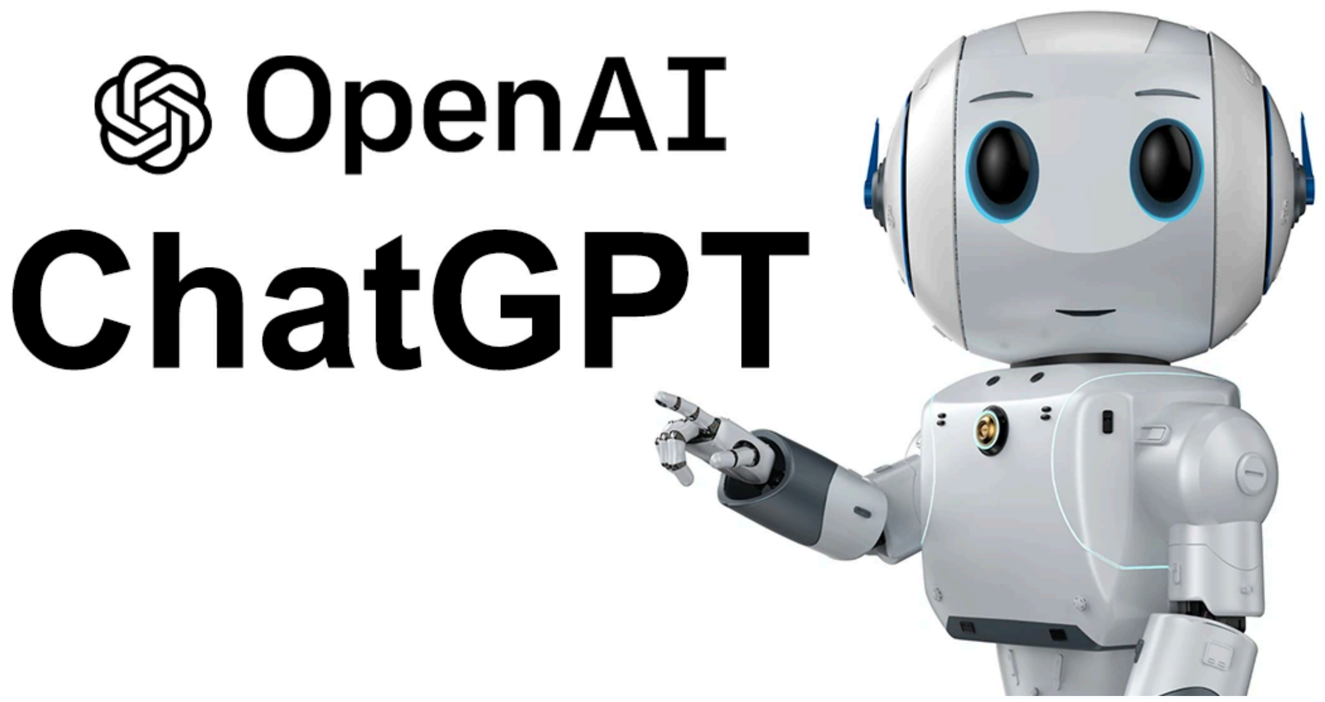
[1] MIT [2] NVIDIA

[*] Equal contribution

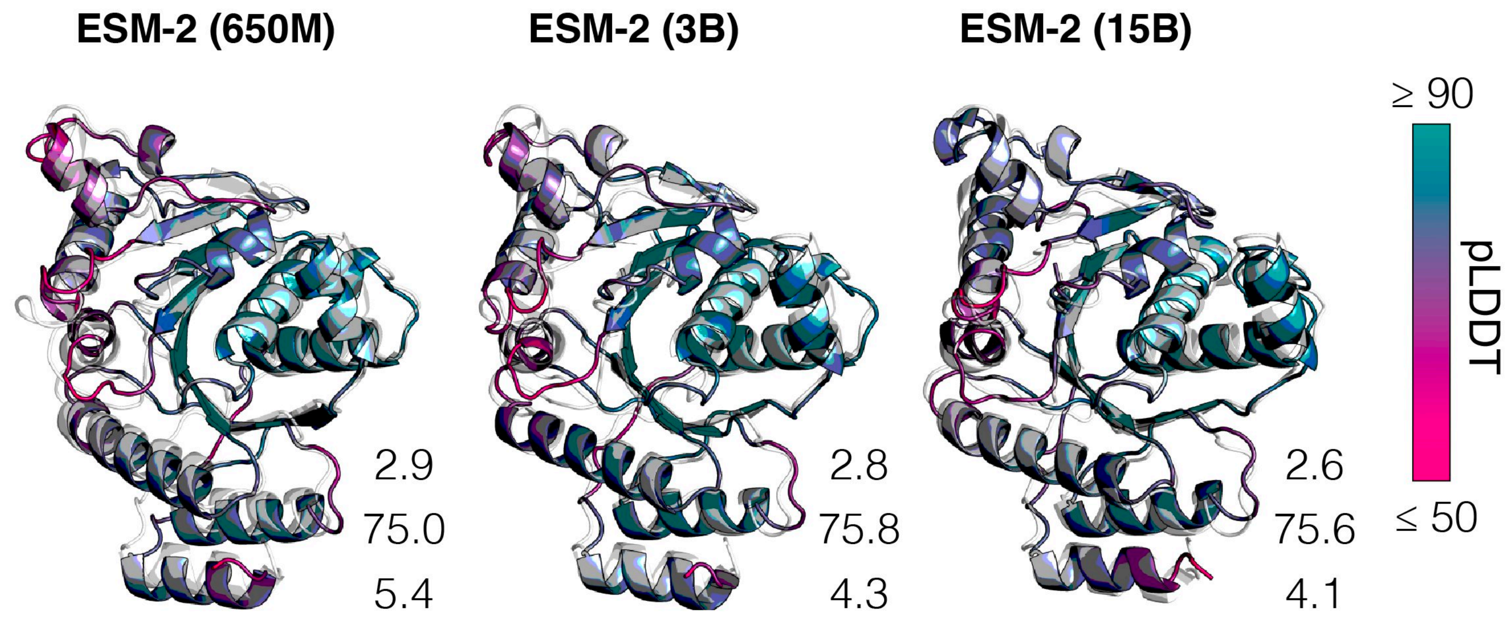
Large Language Models (LLMs) Are Powerful



Transformer Architecture



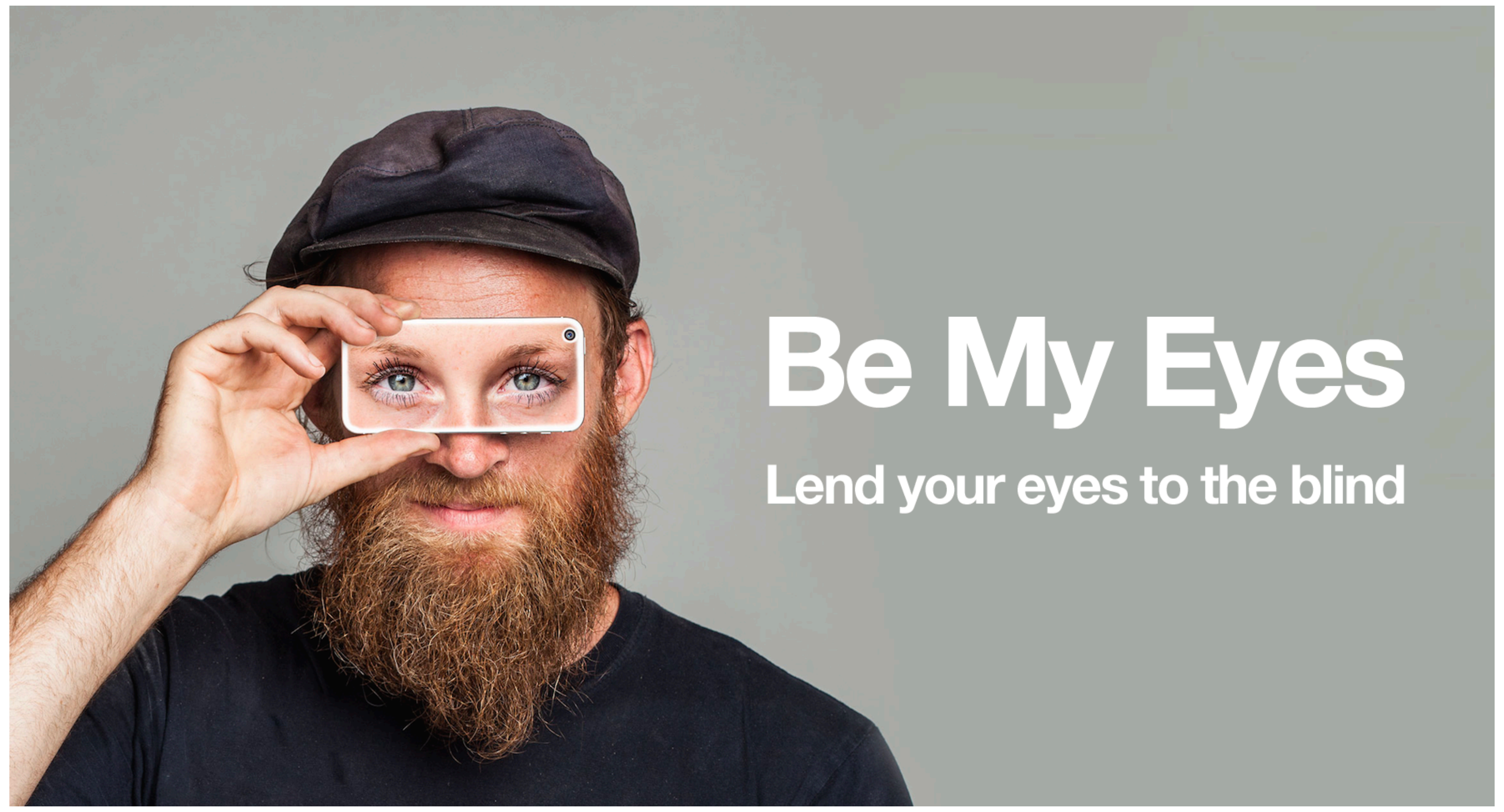
ChatBots



Scientific Discovery

GitHub Copilot

Software Development

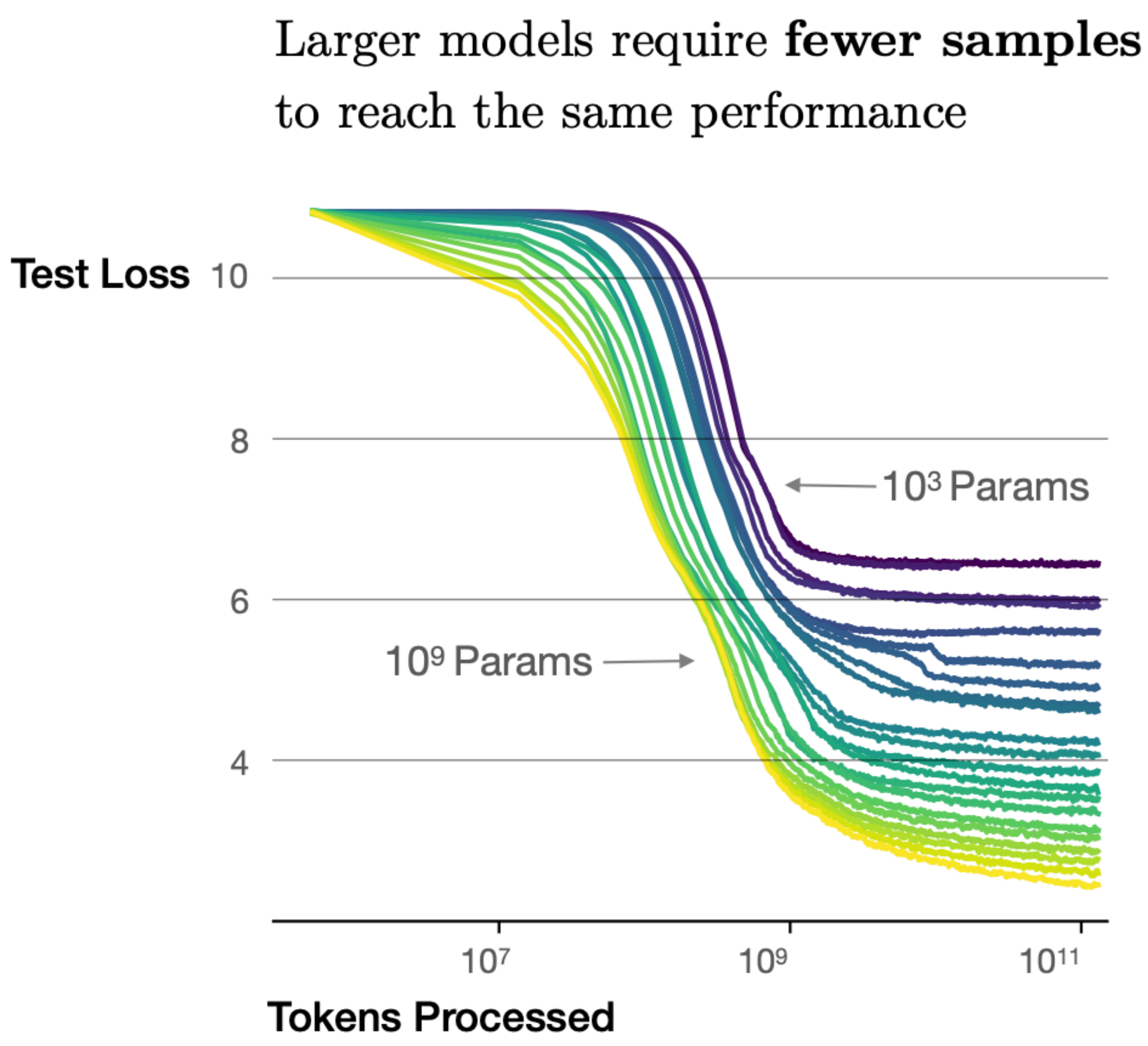


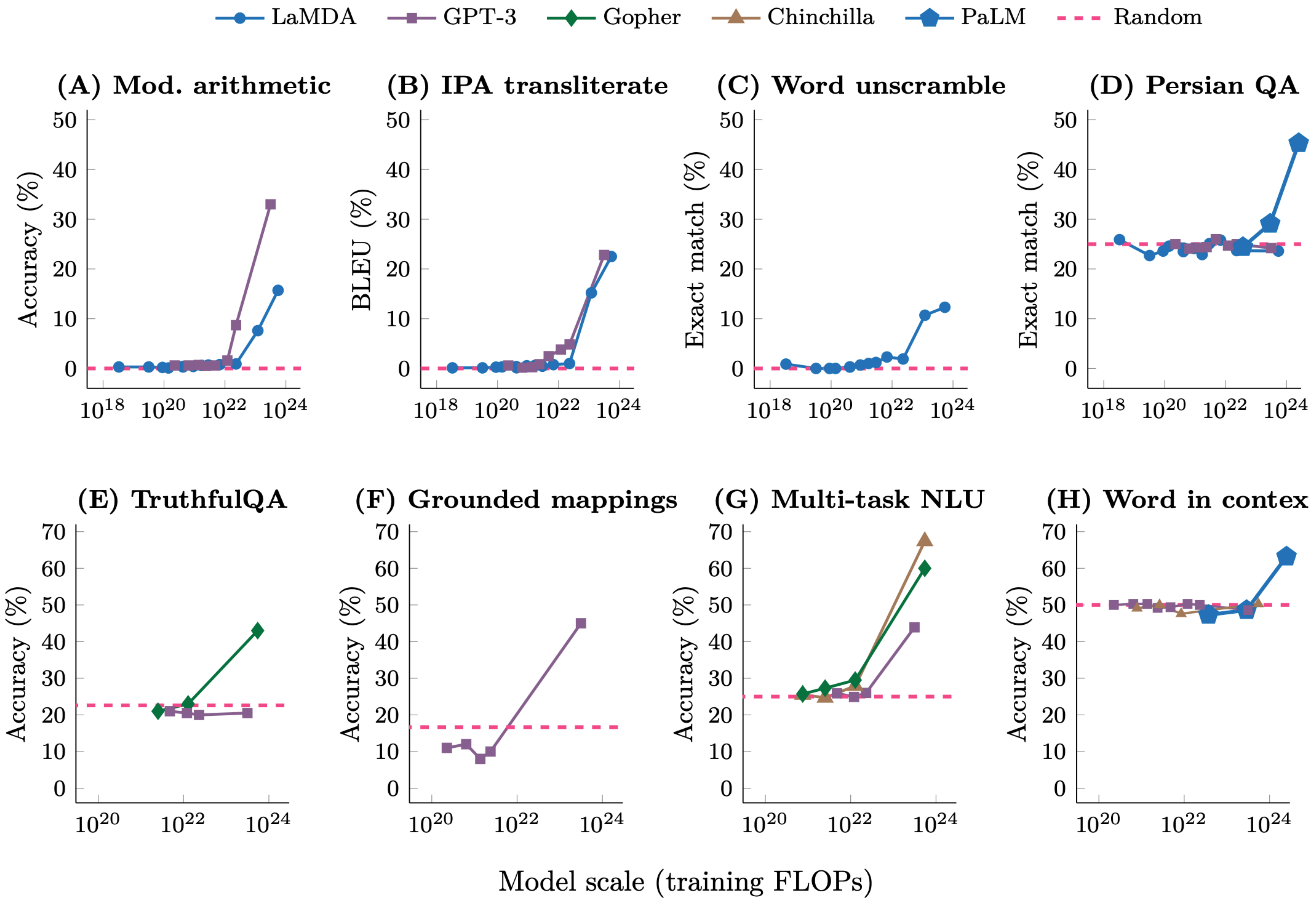
Disability Aid

and more…

Attention Is All You Need (Vaswani *et al.*, 2017)

The Scaling Law and Emergent Abilities



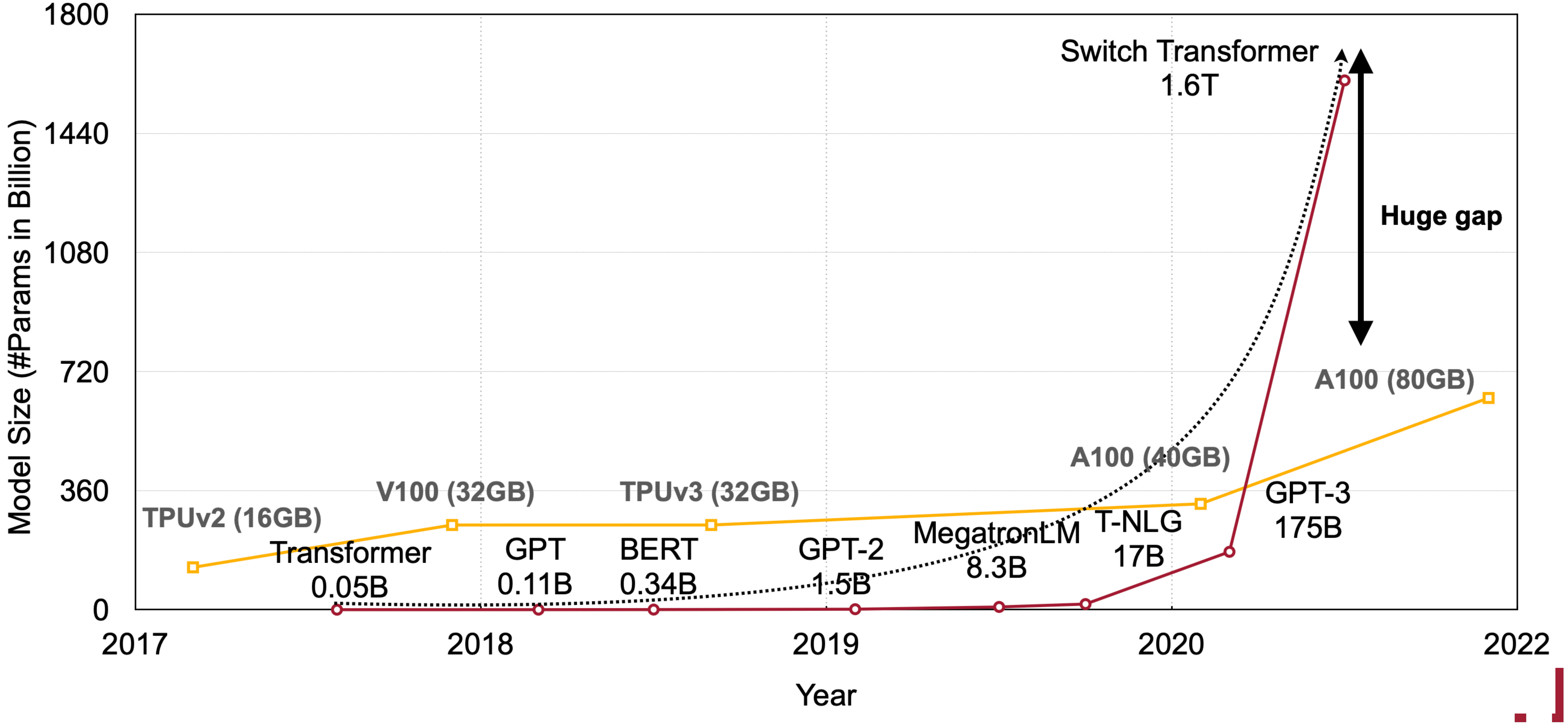


- Scaling up language models can give us unpredictable capabilities (emergent abilities).

Scaling Laws for Neural Language Models (Kaplan *et al.*, 2020)
Emergent Abilities of Large Language Models (Wei *et al.*, 2022)

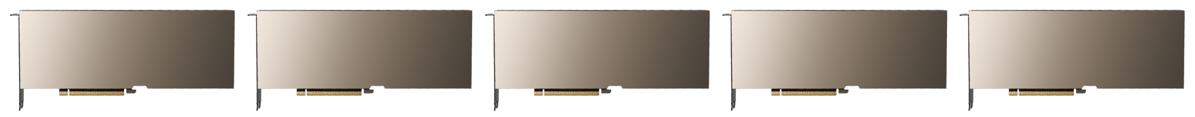
Model Compression for LLMs is Important

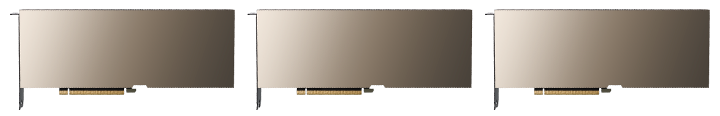
- LLM sizes and computation are increasing exponentially. Model Compression with:
  - Quantization (SmoothQuant) <= today's focus: **training-free**, **model-in & model-out**.
  - Token pruning (SpAtten)
  - Neural architecture search (HAT, Lite-Transformer)

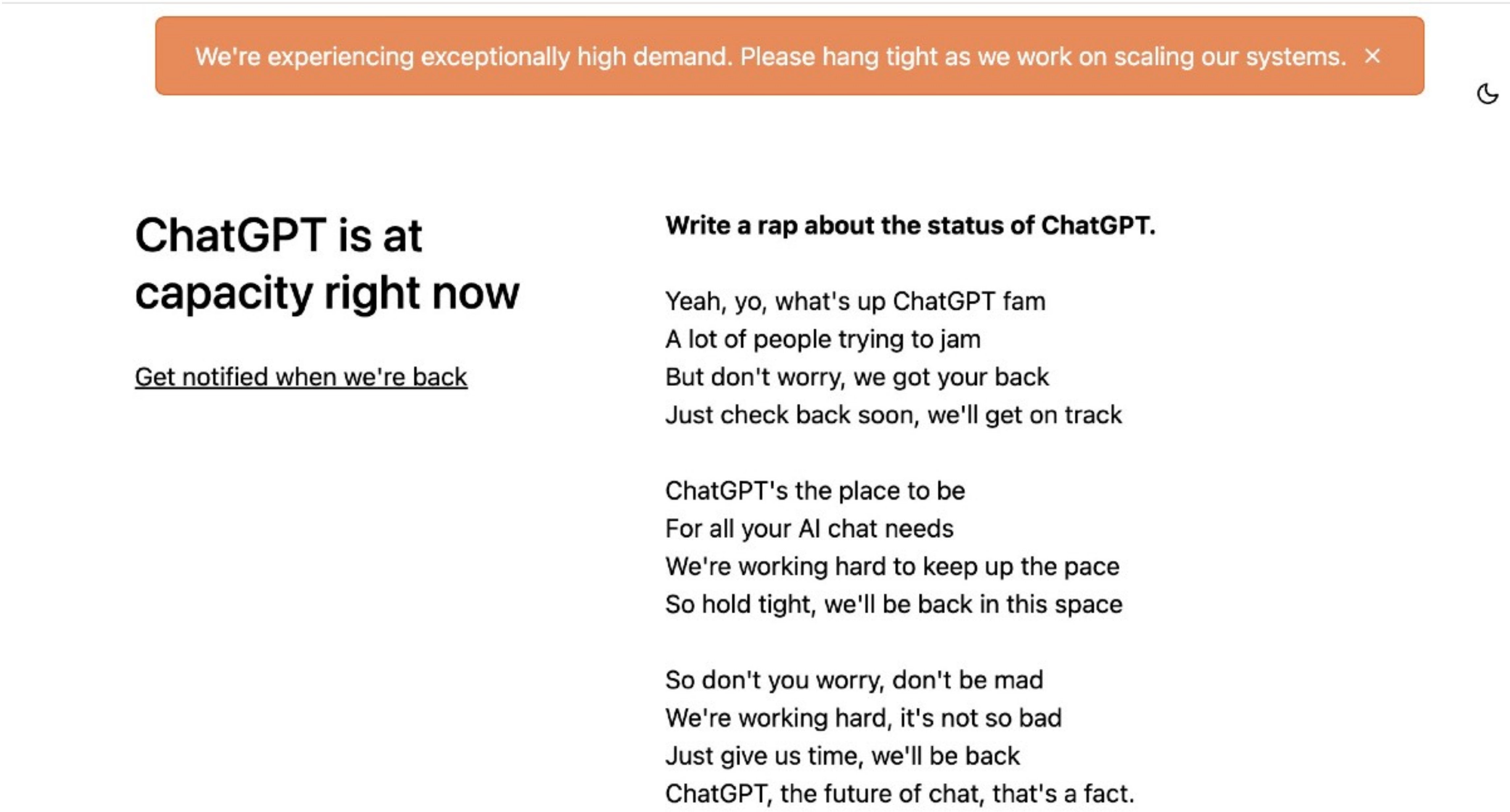


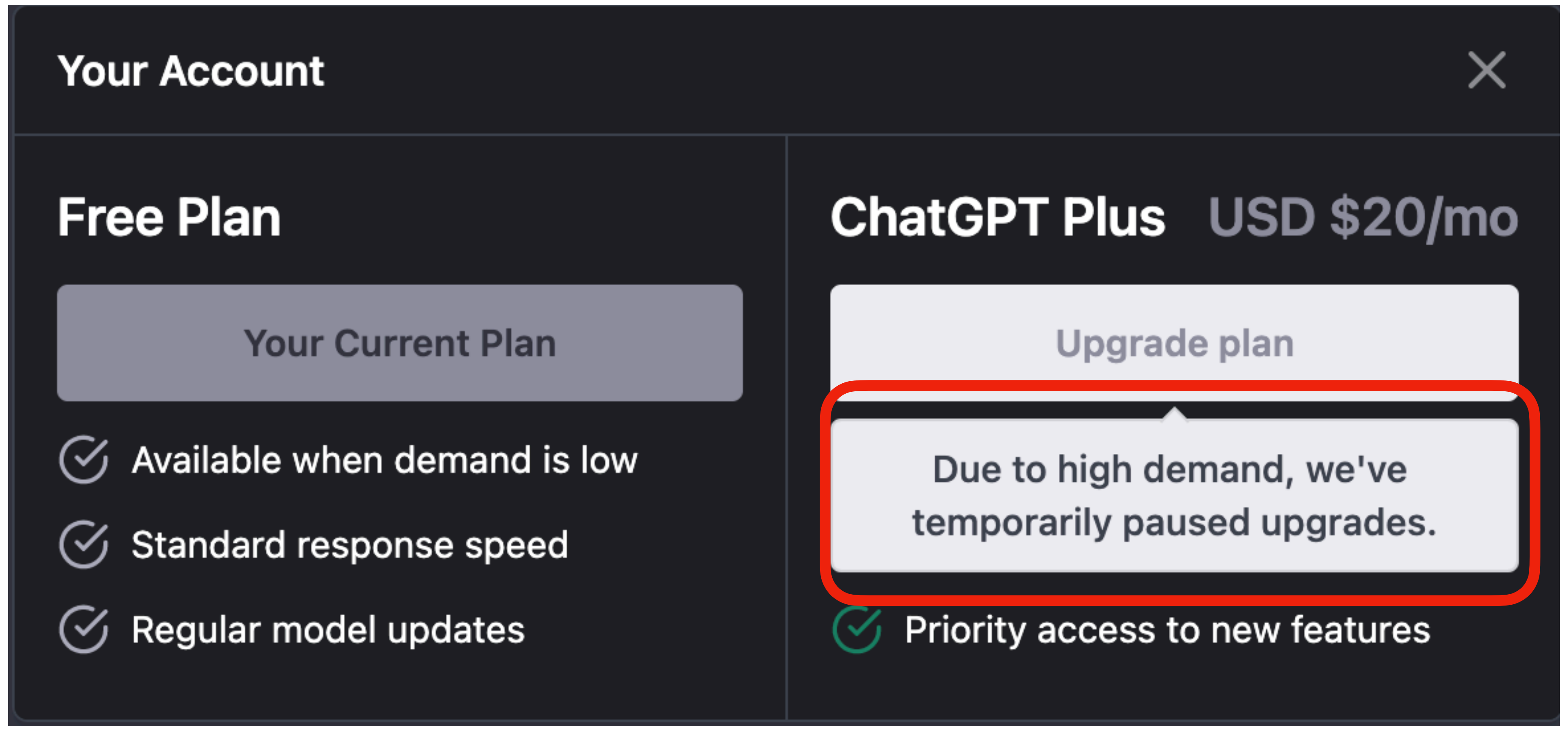
Quantization Can Reduce Deployment Costs

- Serving LLMs is extremely expensive.
- E.g., serving a 175B GPT-3 model at least requires:
  - FP16: 350GB memory ➡️ 5 x 80GB A100 GPUs
  - **INT8:** 175GB memory ➡️ **3 x 80GB A100 GPUs**



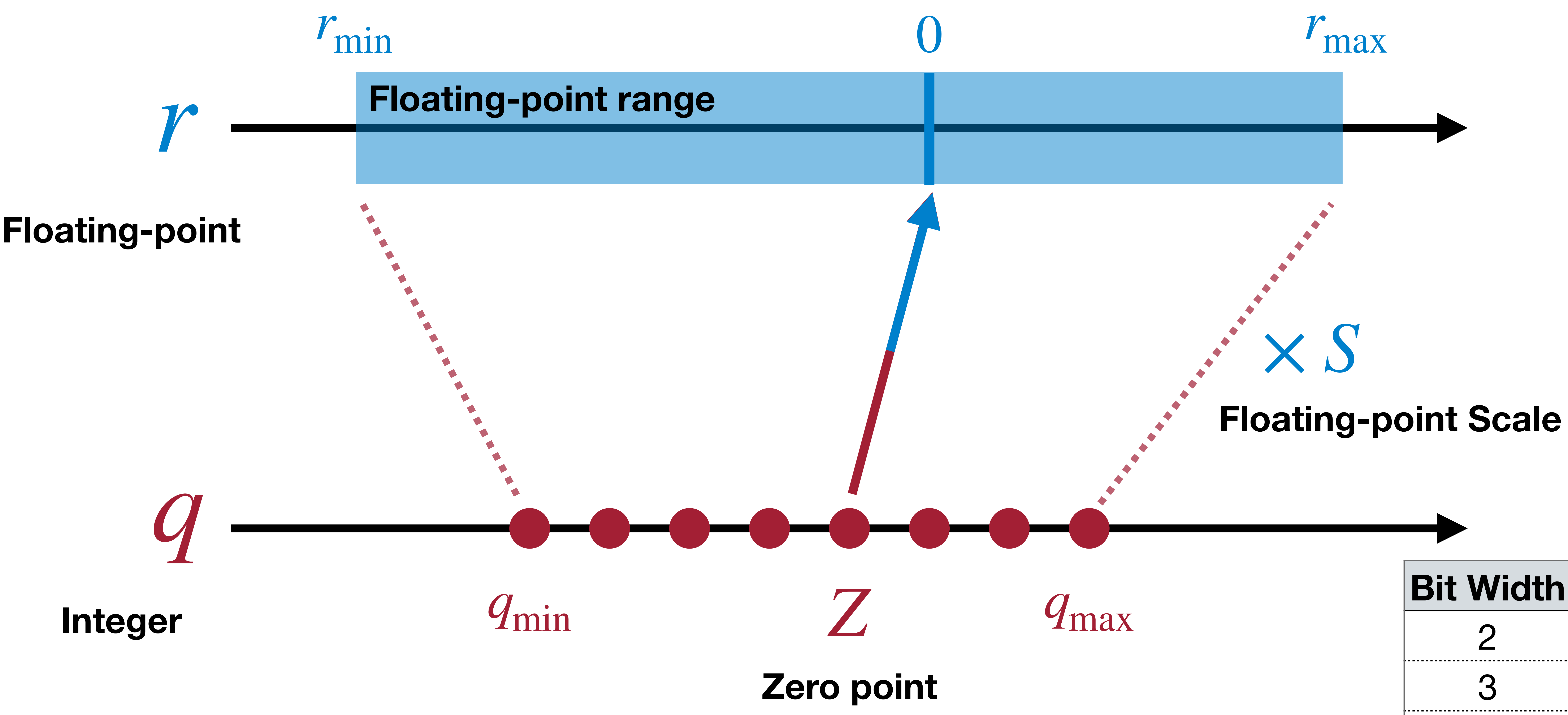






Linear Quantization

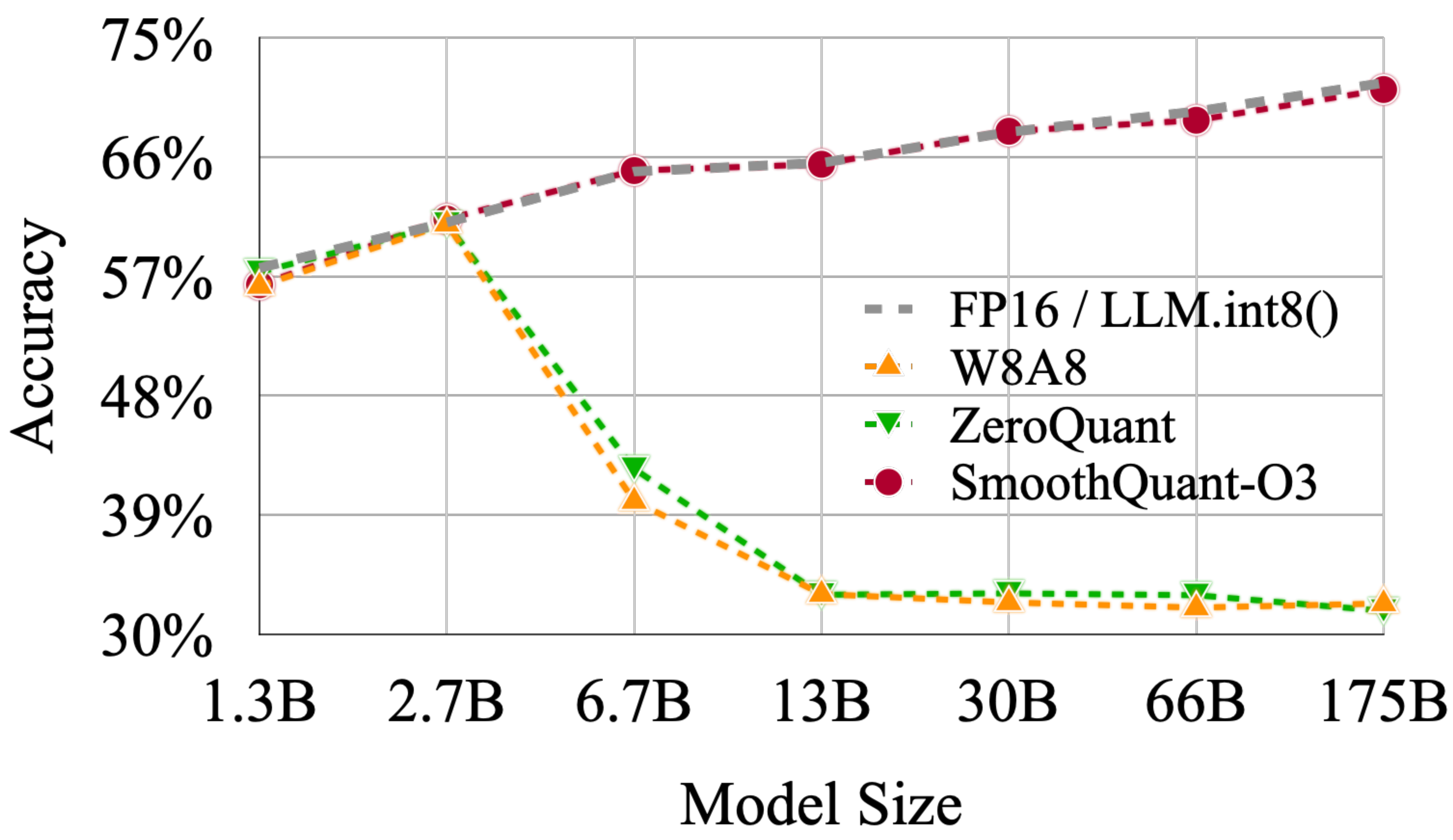
An affine mapping of integers to real numbers $r = S(q - Z)$

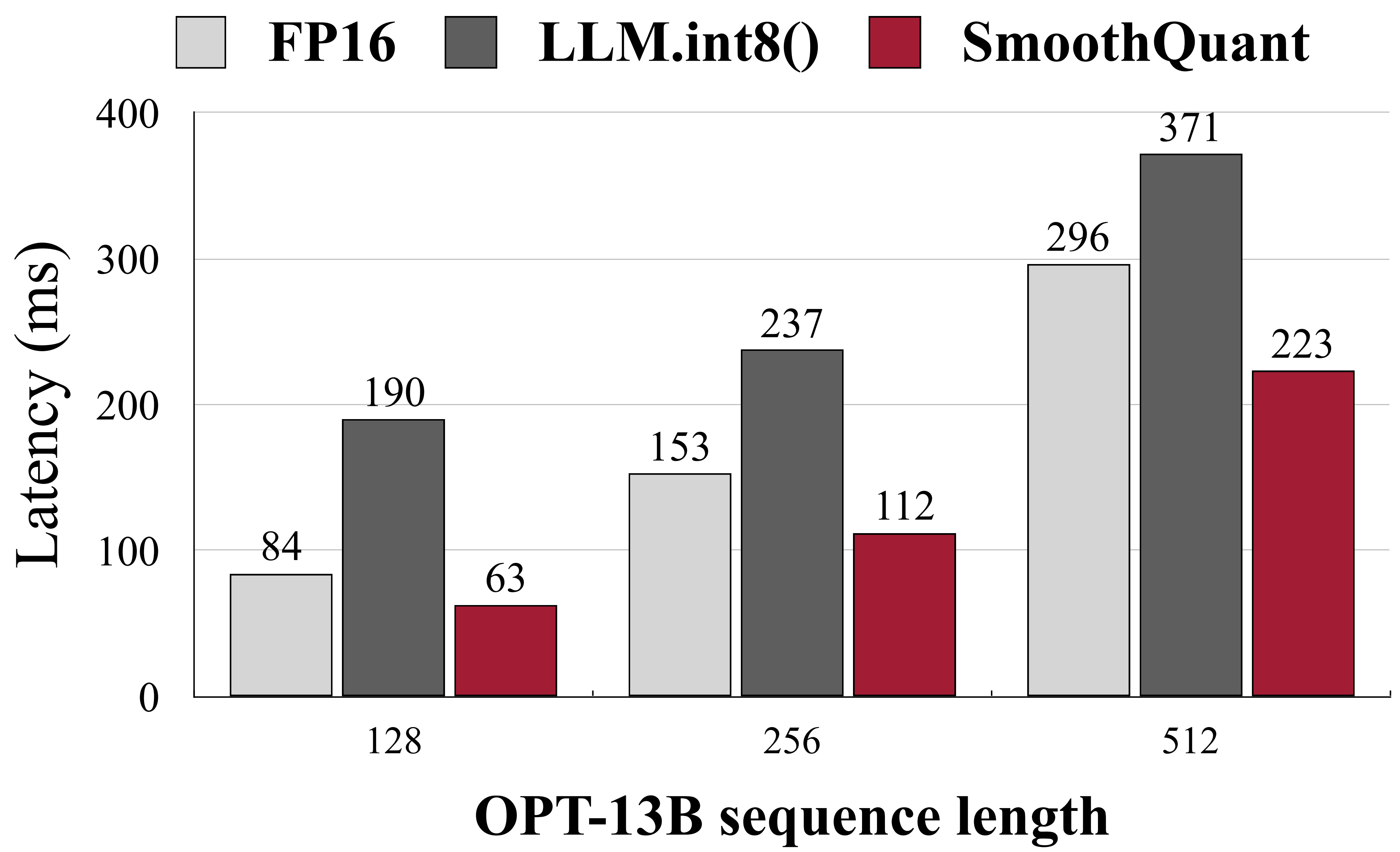


Bit Width	q_{min}	q_{max}
2	-2	1
3	-4	3
4	-8	7
N	-2^{N-1}	$2^{N-1}-1$

Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [Jacob *et al.*, CVPR 2018]

Existing Quantization Method is Slow or Inaccurate



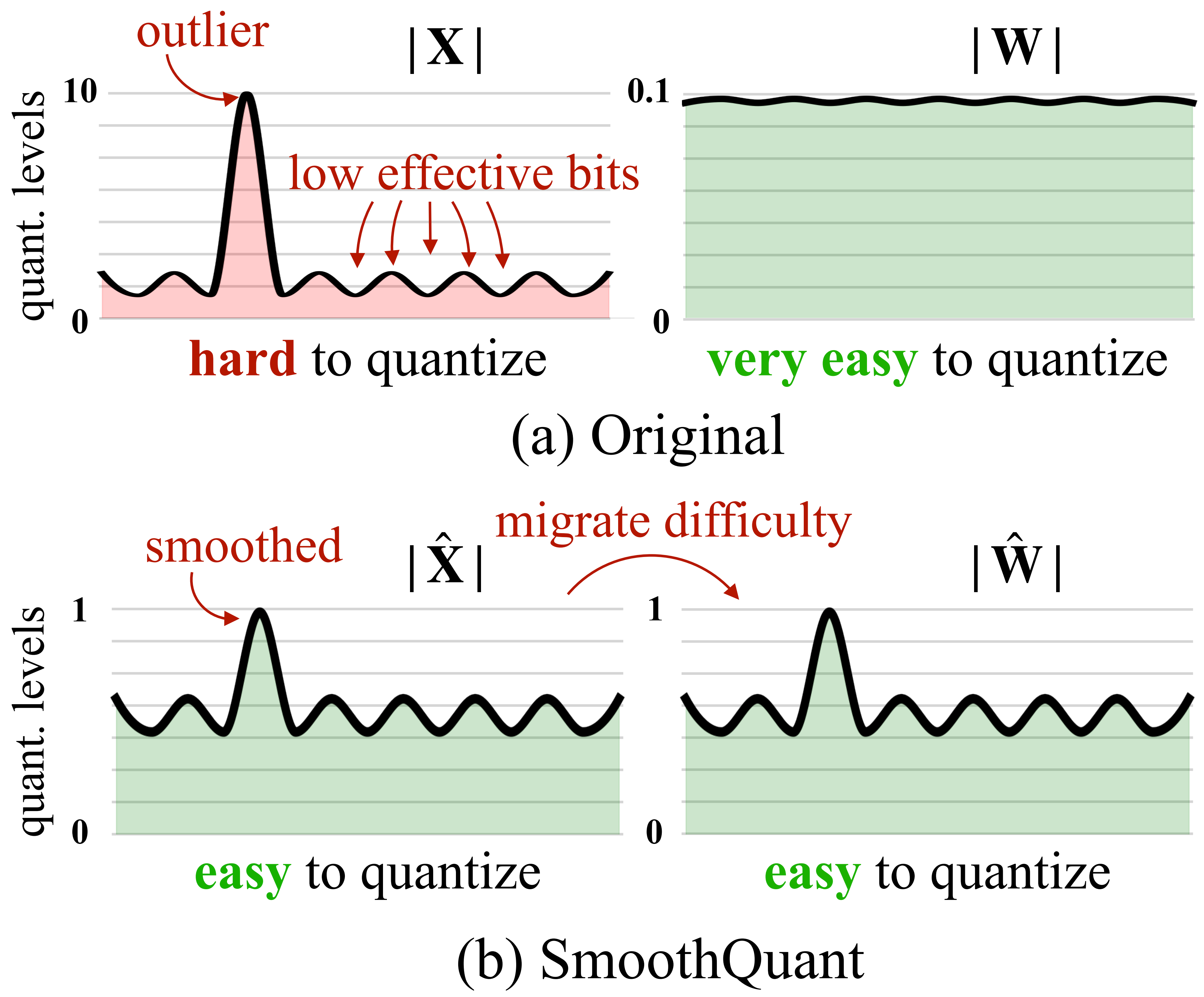


- Systematic outliers emerge in activations when we scale up LLMs beyond 6.7B. Naive but efficient quantization methods will destroy the accuracy.
- The accuracy-preserving baseline, LLM.int8() uses FP16 to represent outliers, which needs runtime outlier detection, scattering and gathering. It is slower than FP16 inference.

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

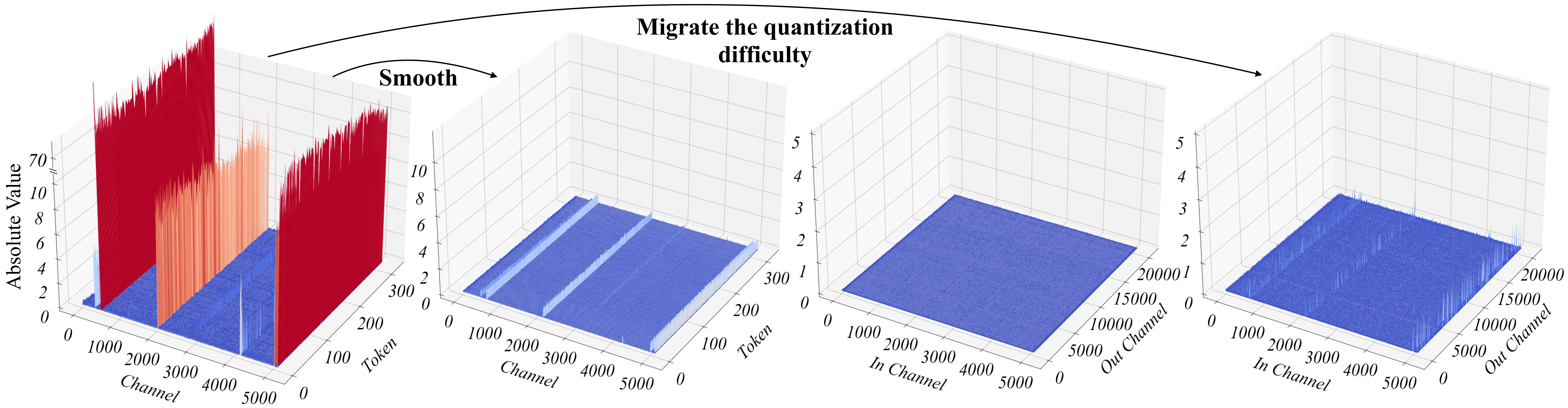
SmoothQuant: Accurate and Efficient Post-Training Quantization for LLMs

	LLM (100B+) Accuracy	Hardware Efficiency
ZeroQuant	✘	✔
Outlier Suppression	✘	✔
LLM.int8()	✔	✘
SmoothQuant	✔	✔



- 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.

Review the Quantization Difficulty of LLMs



Activation (Original)
Hard to quantize

Activation (SmoothQuant)
Easy to quantize

Weight (Original)
Very easy to quantize

Weight (SmoothQuant)
Harder but still easy to quantize

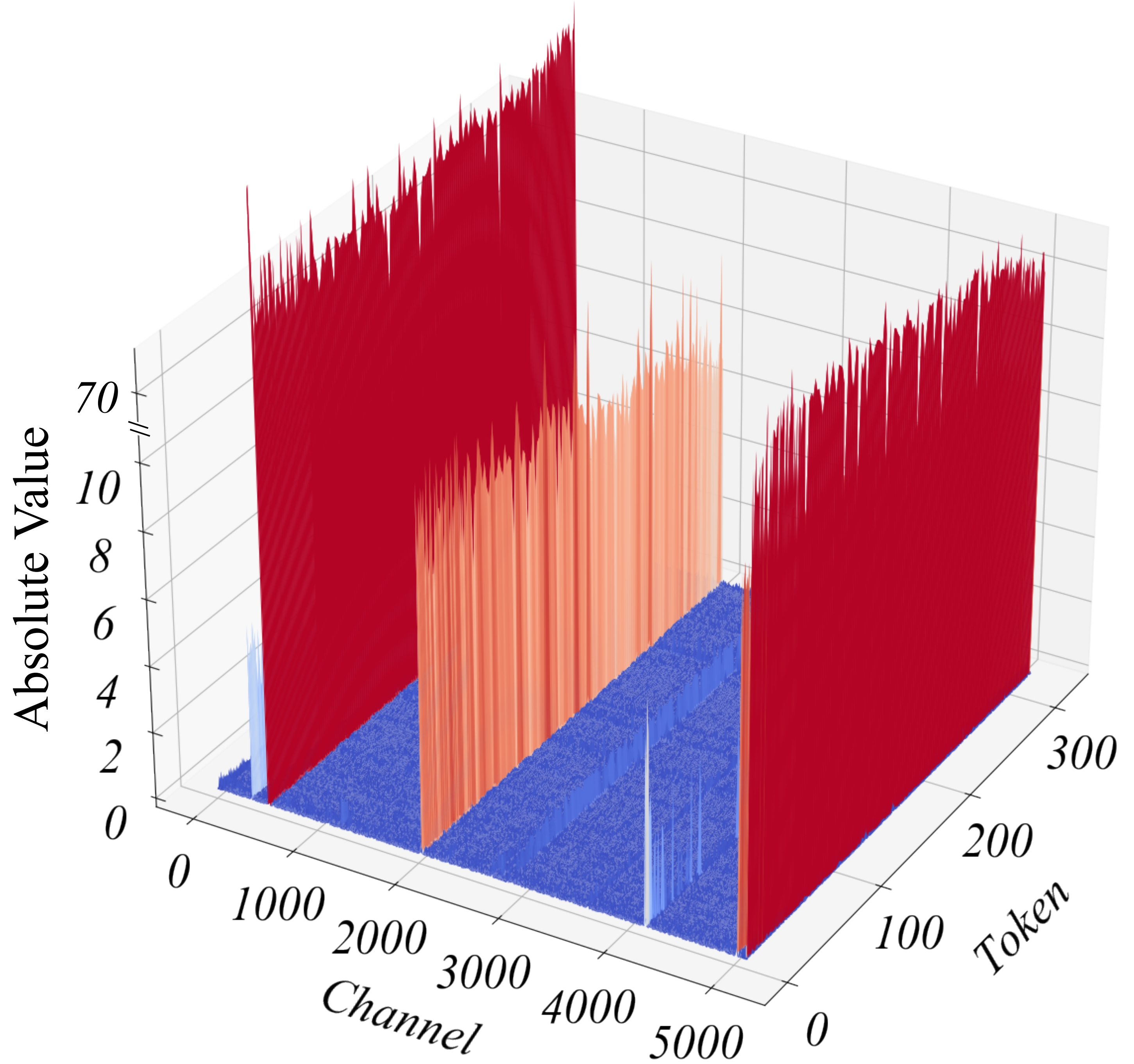
LLMs are difficult to quantize because:

- Activations are harder to quantize than weights
- Outliers make activation quantization difficult
- Outliers persist in *fixed* channels

Review the Quantization Difficulty of LLMs

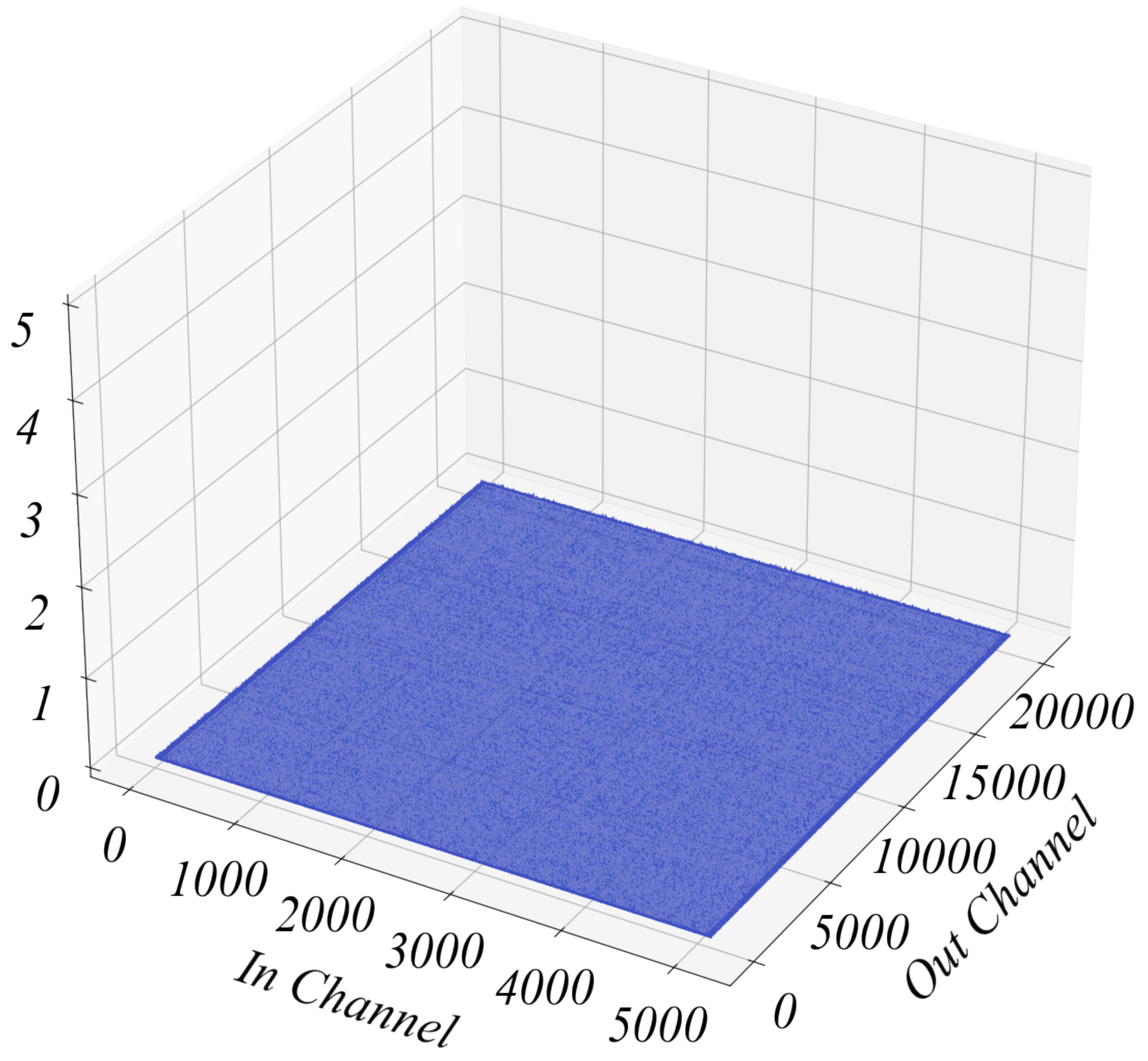
- Activations are harder to quantize than weights

  Previous work has shown quantizing the weights of LLMs with INT8 or even INT4 doesn't degrade accuracy.



Activation (Original)

Hard to quantize



Weight (Original)

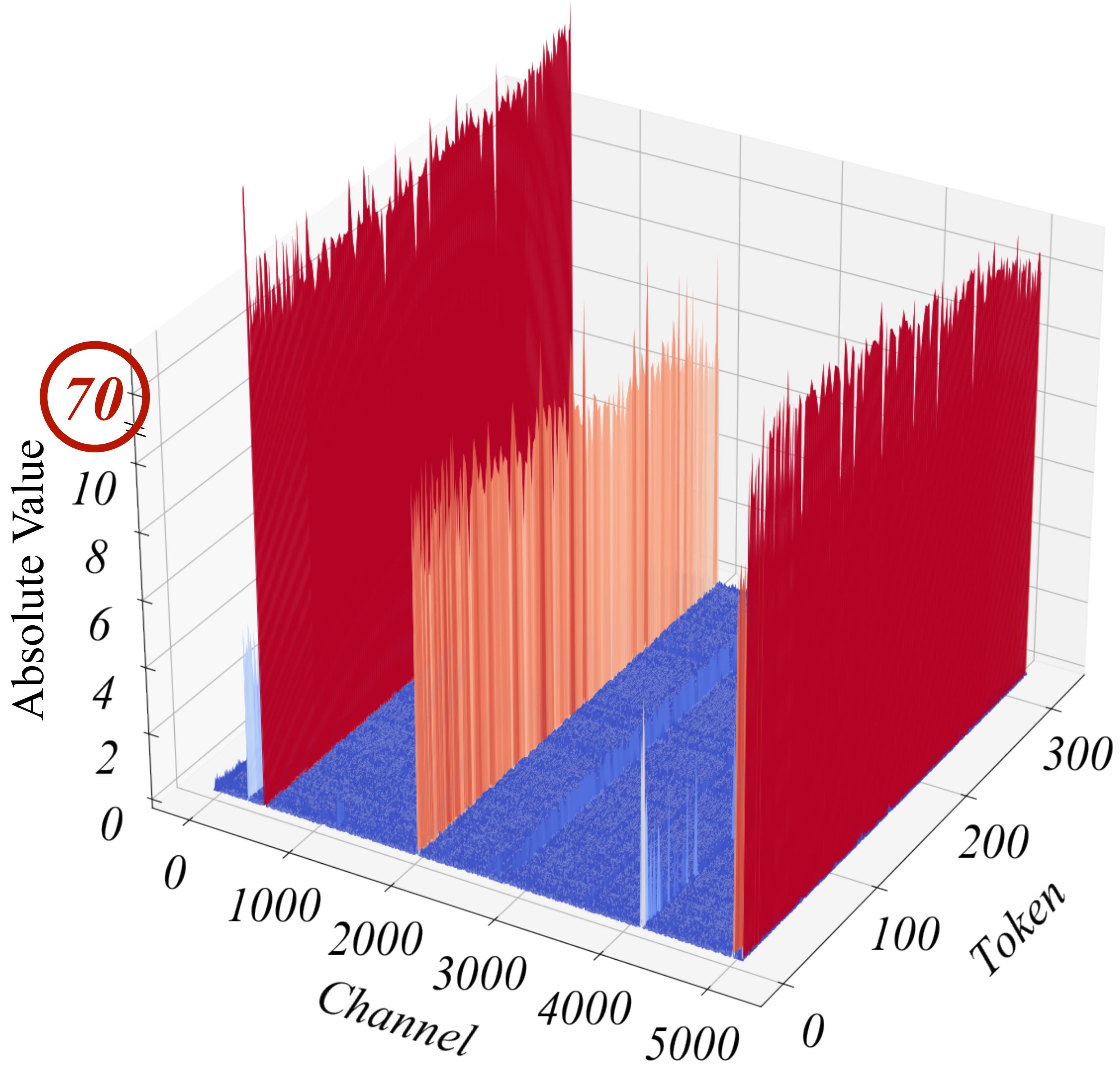
Very easy to quantize

LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale (Dettmers *et al.*, 2022)
GLM-130b: An open bilingual pre-trained model (Zeng *et al.*, 2022)

Review the Quantization Difficulty of LLMs

- Outliers make activation quantization difficult

  The scale of outliers is ~100x larger than most of the activation values.
  If we use INT8 quantization, most values will be zeroed out.



Activation (Original)

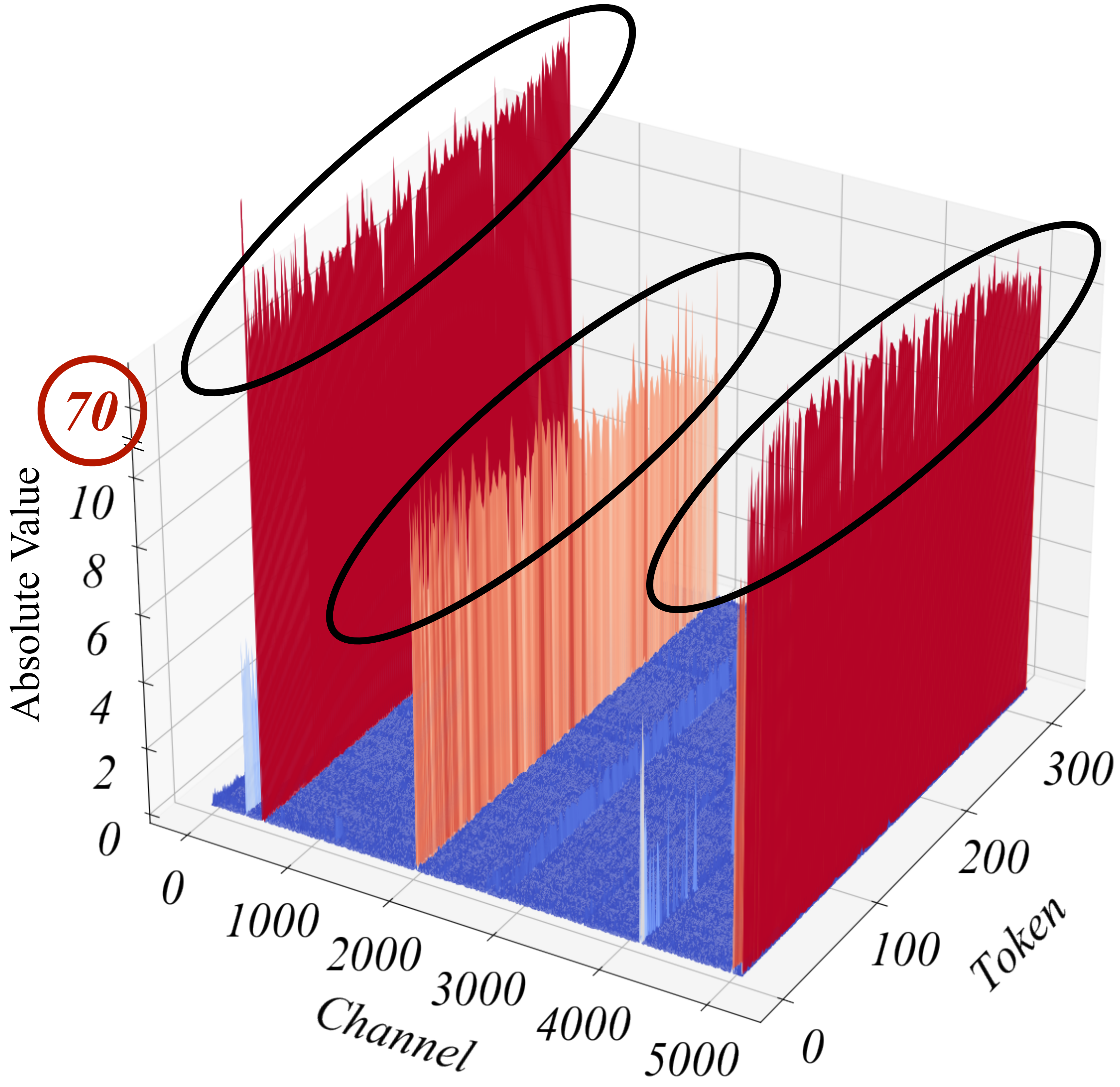
Hard to quantize

Understanding and overcoming the challenges of efficient transformer quantization (Bondarenko *et al.*, 2021)
LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale (Dettmers *et al.*, 2022)

Review the Quantization Difficulty of LLMs

- Outliers persist in *fixed* channels

  Fixed channels have outliers, and the outlier channels are persistently large.

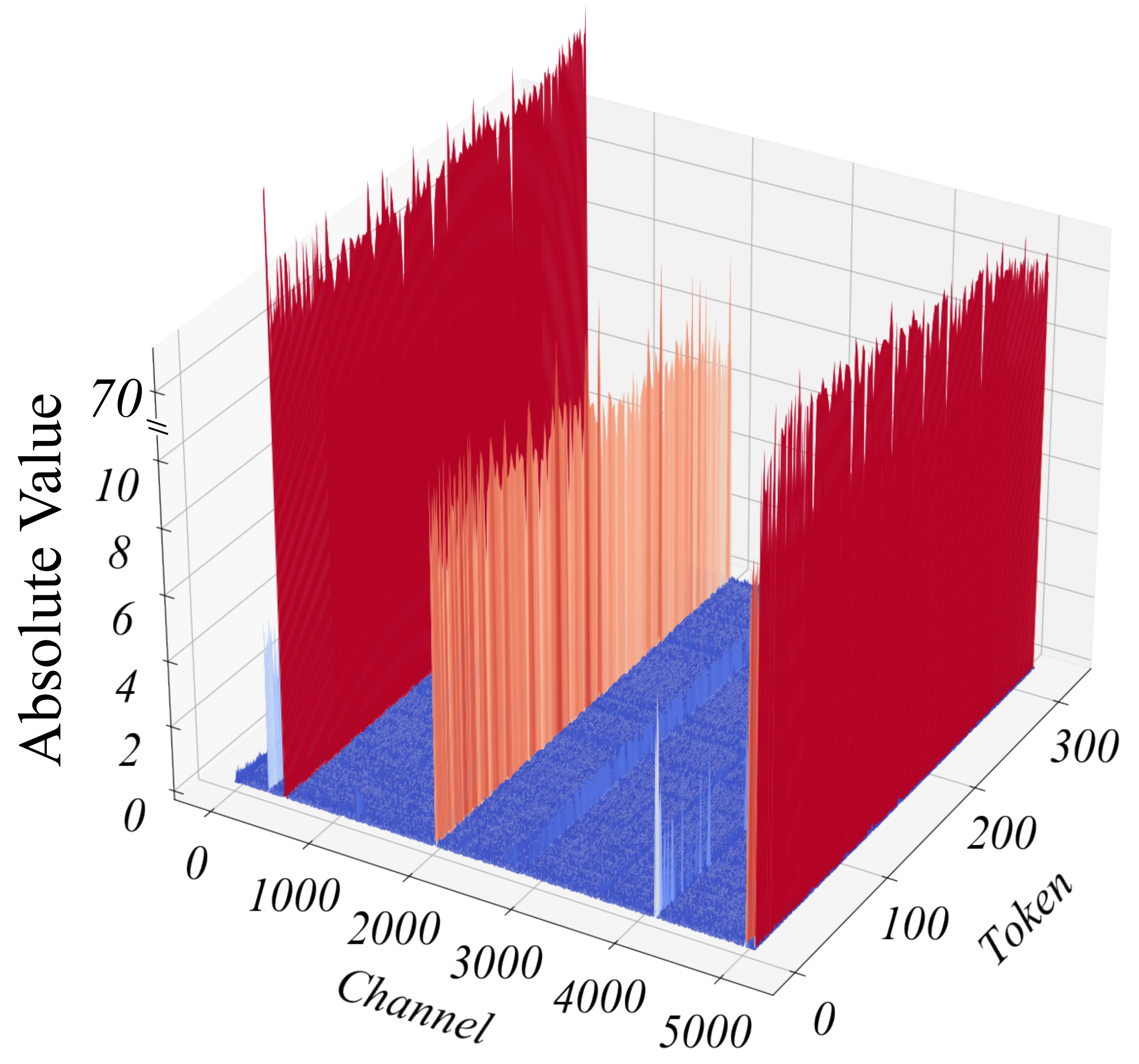


Activation (Original)

Hard to quantize

Understanding and overcoming the challenges of efficient transformer quantization (Bondarenko *et al.*, 2021)

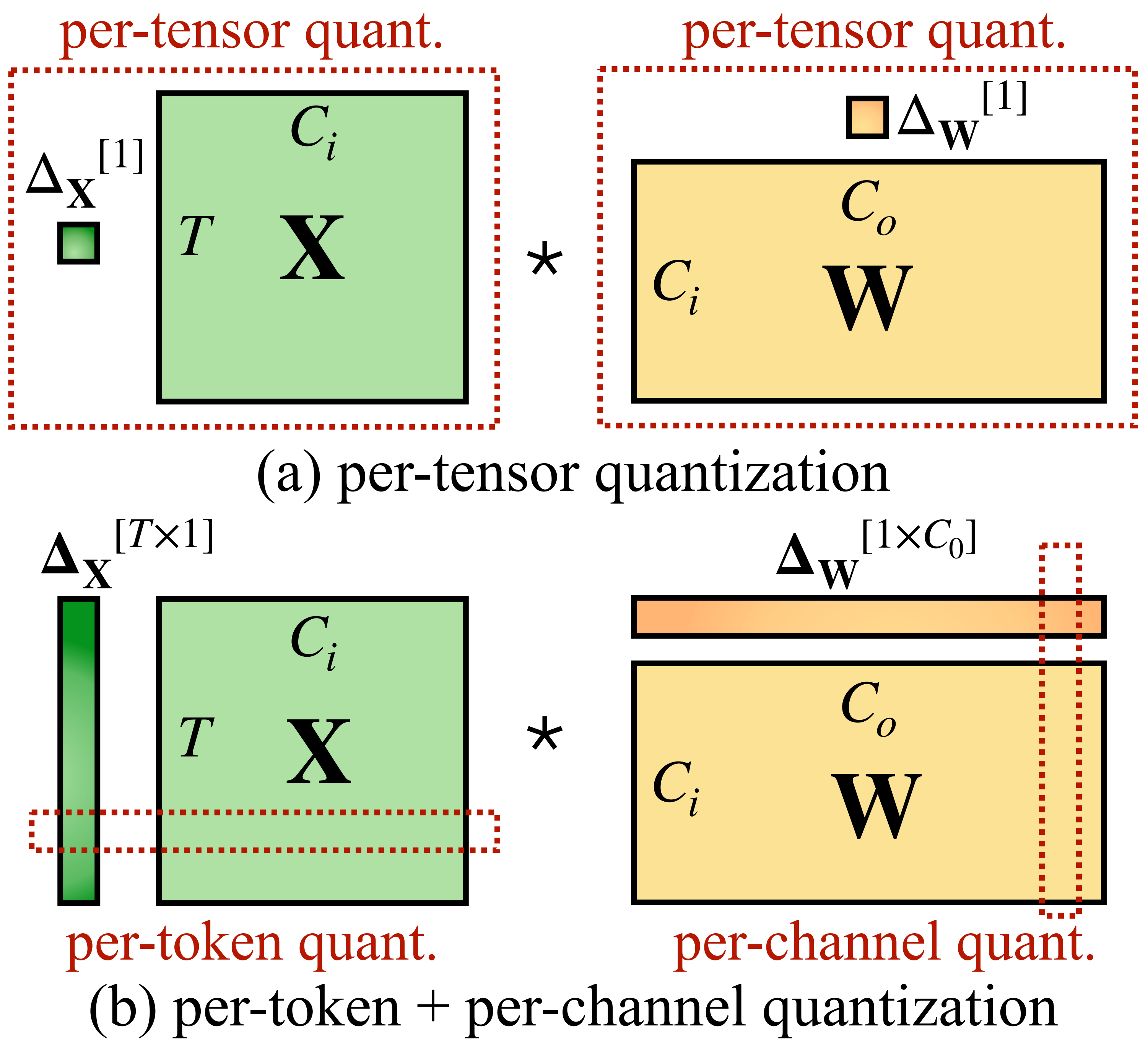
Quantization Schemes



Activation (Original)
Hard to quantize

Model size	6.7B	13B	30B	66B	175B
FP16	64.9%	65.6%	67.9%	69.5%	71.6%
INT8 per-tensor	39.9%	33.0%	32.8%	33.1%	32.3%
INT8 per-token	42.5%	33.0%	33.1%	32.9%	31.7%
INT8 per-channel	64.8%	65.6%	68.0%	69.4%	71.4%

Among different activation quantization schemes, only per-channel quantization preserves the accuracy, but it is not compatible with INT8 GEMM kernels.

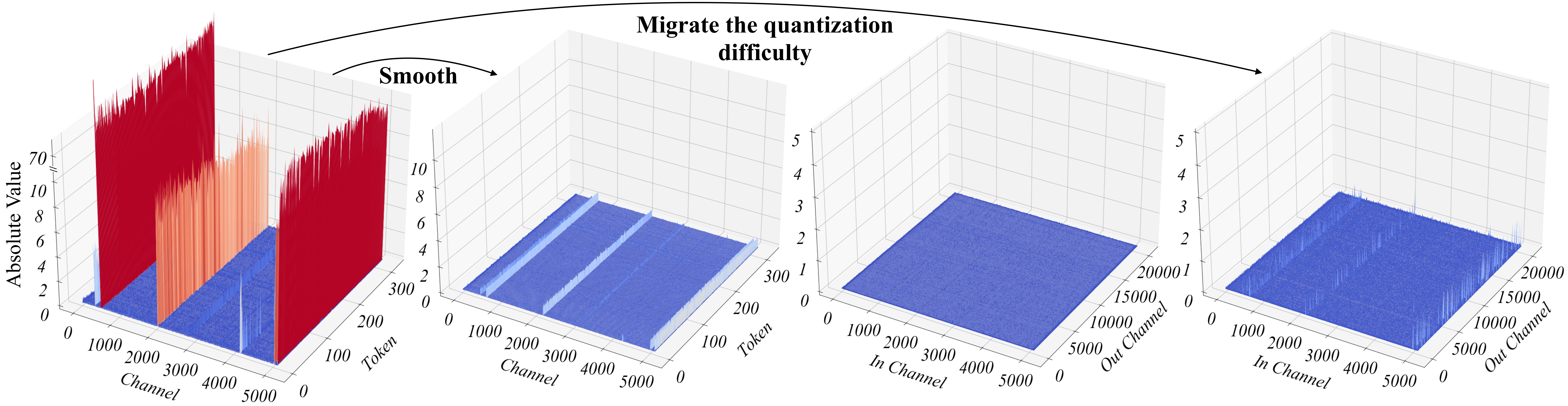


$$\bar{\mathbf{X}}^{\text{INT8}} = \left\lceil \frac{\mathbf{X}^{\text{FP16}}}{\Delta} \right\rfloor, \quad \Delta = \frac{\max(|\mathbf{X}|)}{2^{N-1}-1}$$

$$\mathbf{Y} = diag(\boldsymbol{\Delta}_{\mathbf{X}}^{\text{FP16}}) \cdot (\bar{\mathbf{X}}^{\text{INT8}} \cdot \bar{\mathbf{W}}^{\text{INT8}}) \cdot diag(\boldsymbol{\Delta}_{\mathbf{W}}^{\text{FP16}})$$

Understanding and overcoming the challenges of efficient transformer quantization (Bondarenko *et al.*, 2021)

Review the Quantization Difficulty of LLMs



Activation (Original)
Hard to quantize

Activation (SmoothQuant)
Easy to quantize

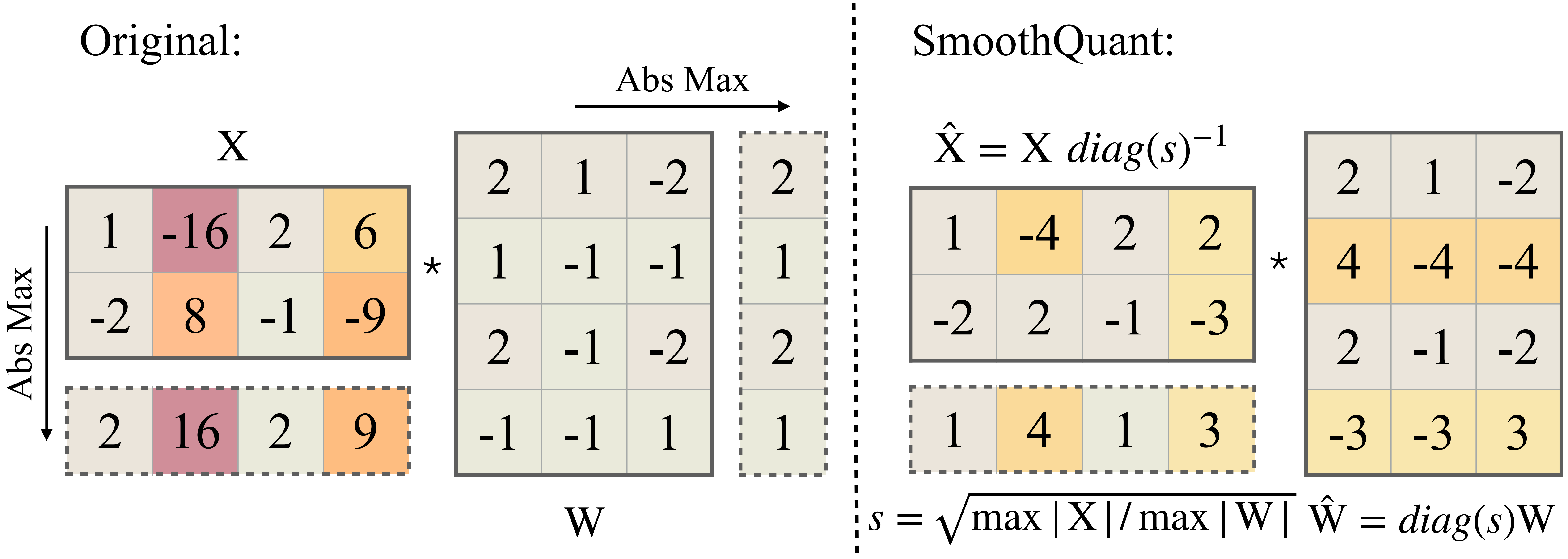
Weight (Original)
Very easy to quantize

Weight (SmoothQuant)
Harder but still easy to quantize

- Activations are harder to quantize than weights
- Outliers make activation quantization difficult
- Outliers persist in *fixed* channels

➡ **We can smooth the outlier channels in activations by migrating their magnitudes into the following weights!**

Activation Smoothing



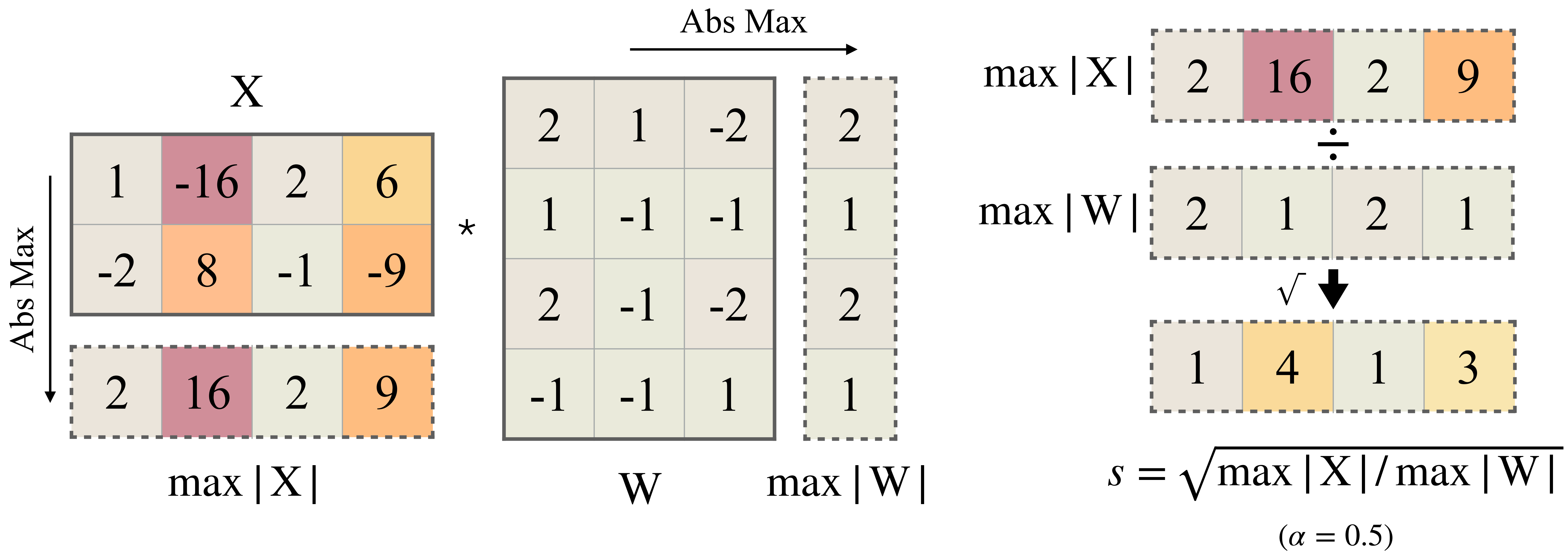
$$\mathbf{s}_j = \max(|\mathbf{X}_j|)^{\alpha} / \max(|\mathbf{W}_j|)^{1-\alpha},\ j = 1,2,\ldots, C_i$$

$$\mathbf{Y} = (\mathbf{X} diag(\mathbf{s})^{-1}) \cdot (diag(\mathbf{s})\mathbf{W}) = \hat{\mathbf{X}}\hat{\mathbf{W}}$$

α: Migration Strength

Activation Smoothing

1.Calibration Stage (Offline):



$$\mathbf{s}_j = \max(|\mathbf{X}_j|)^{\alpha} / \max(|\mathbf{W}_j|)^{1-\alpha},\ j = 1,2,\ldots,C_i$$

α: Migration Strength

Activation Smoothing

2. Smoothing Stage (Offline):

$$X = \begin{bmatrix} 1 & -16 & 2 & 6 \\ -2 & 8 & -1 & -9 \end{bmatrix} \div s = \begin{bmatrix} 1 & 4 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & -4 & 2 & 2 \\ -2 & 2 & -1 & -3 \end{bmatrix}$$

$\hat{X} = X \, diag(s)^{-1}$

divide the output channel of the previous layer by s

multiply the input channel of the following weight by s

$$W = \begin{bmatrix} 2 & 1 & -2 \\ 1 & -1 & -1 \\ 2 & -1 & -2 \\ -1 & -1 & 1 \end{bmatrix} \times s = \begin{bmatrix} 1 \\ 4 \\ 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 & 1 & -2 \\ 4 & -4 & -4 \\ 2 & -1 & -2 \\ -3 & -3 & 3 \end{bmatrix}$$

$\hat{W} = diag(s)W$

$$\mathbf{s}_j = \max(|\mathbf{X}_j|)^{\alpha} / \max(|\mathbf{W}_j|)^{1-\alpha}, \; j = 1,2,\ldots, C_i$$

$$\mathbf{Y} = (\mathbf{X}diag(\mathbf{s})^{-1}) \cdot (diag(\mathbf{s})\mathbf{W}) = \hat{\mathbf{X}}\hat{\mathbf{W}}$$

α: Migration Strength

Activation Smoothing

3. Inference (deployed model):

$\hat{X}$

1	-4	2	2
-2	2	-1	-3

At runtime, the activations are smooth and easy to quantize

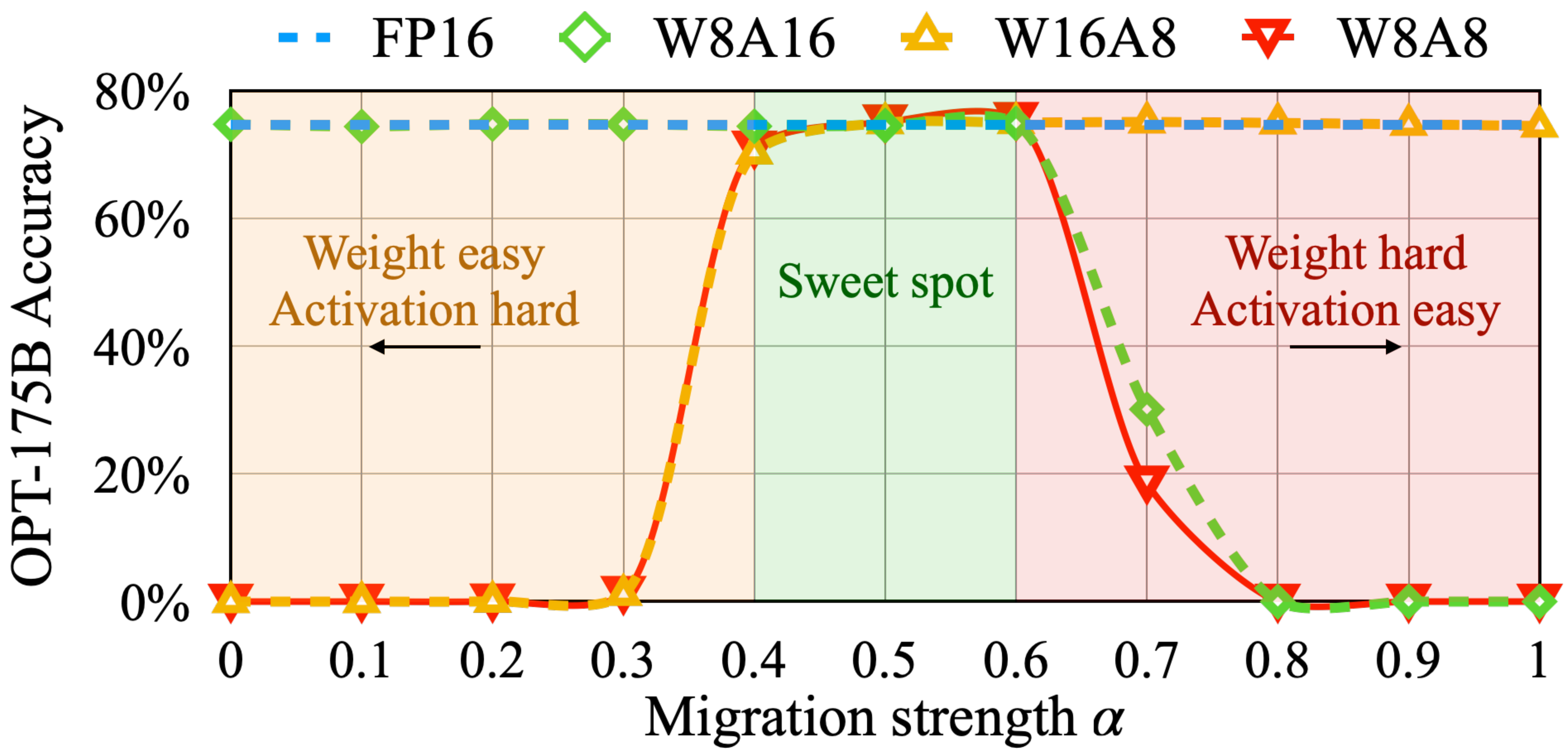
*

2	1	-2
4	-4	-4
2	-1	-2
-3	-3	3

$\hat{W}$

$$\mathbf{Y} = \hat{\mathbf{X}}\hat{\mathbf{W}}$$

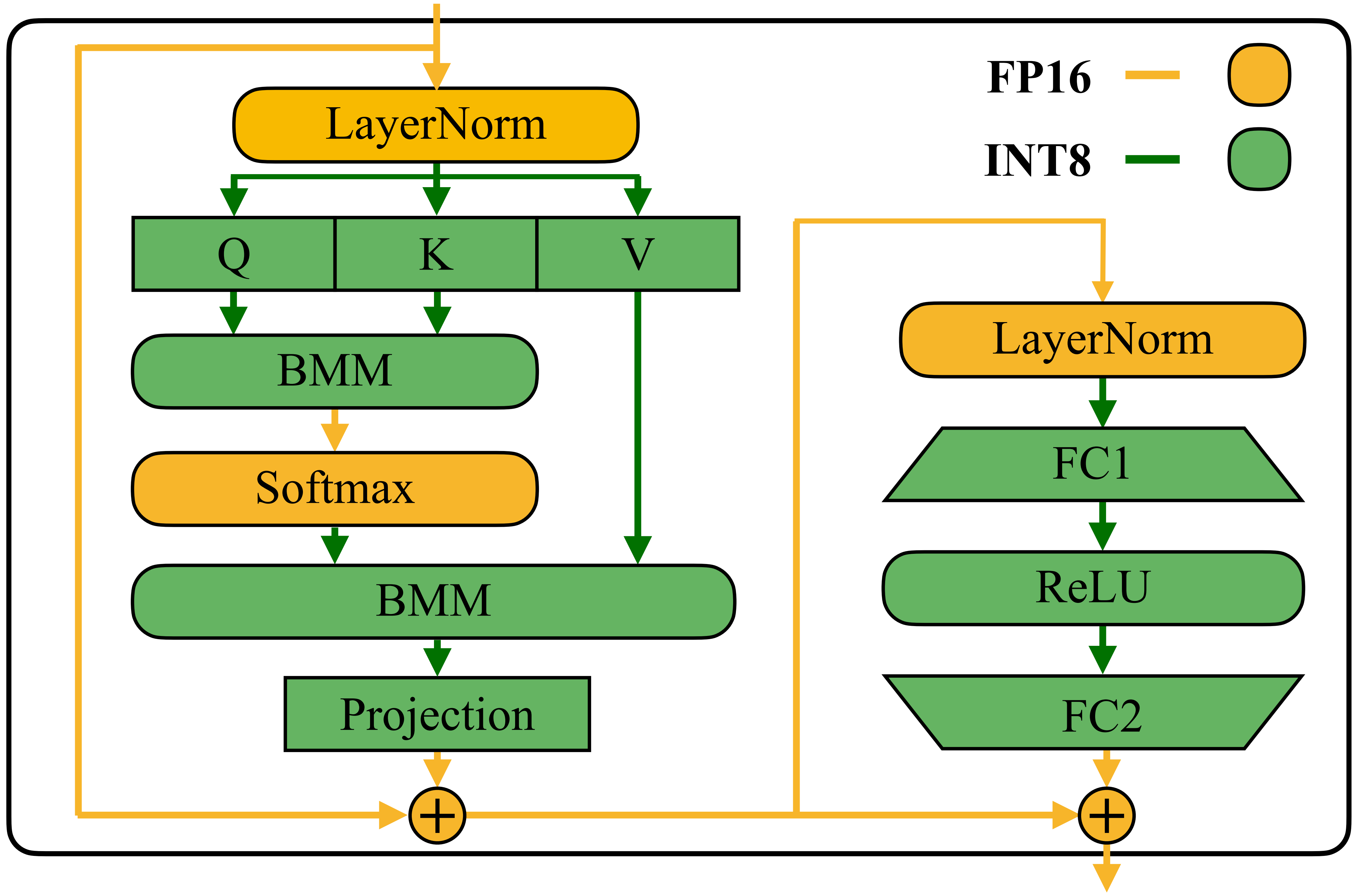
Ablation Study on the Migration Strength α



$\mathbf{s}_j = \max(|\mathbf{X}_j|)^{\alpha} / \max(|\mathbf{W}_j|)^{1-\alpha}, \ j = 1,2,\ldots, C_i \qquad \mathbf{Y} = (\mathbf{X} diag(\mathbf{s})^{-1}) \cdot (diag(\mathbf{s})\mathbf{W}) = \hat{\mathbf{X}}\hat{\mathbf{W}}$

- Migration strength α controls the amount of quantization difficulty migrated from activations to weights.
- A suitable migration strength α (sweet spot) makes both activations and weights easy to quantize.
- If the α is too large, weights will be hard to quantize; if too small, activations will be hard to quantize.

System Implementation



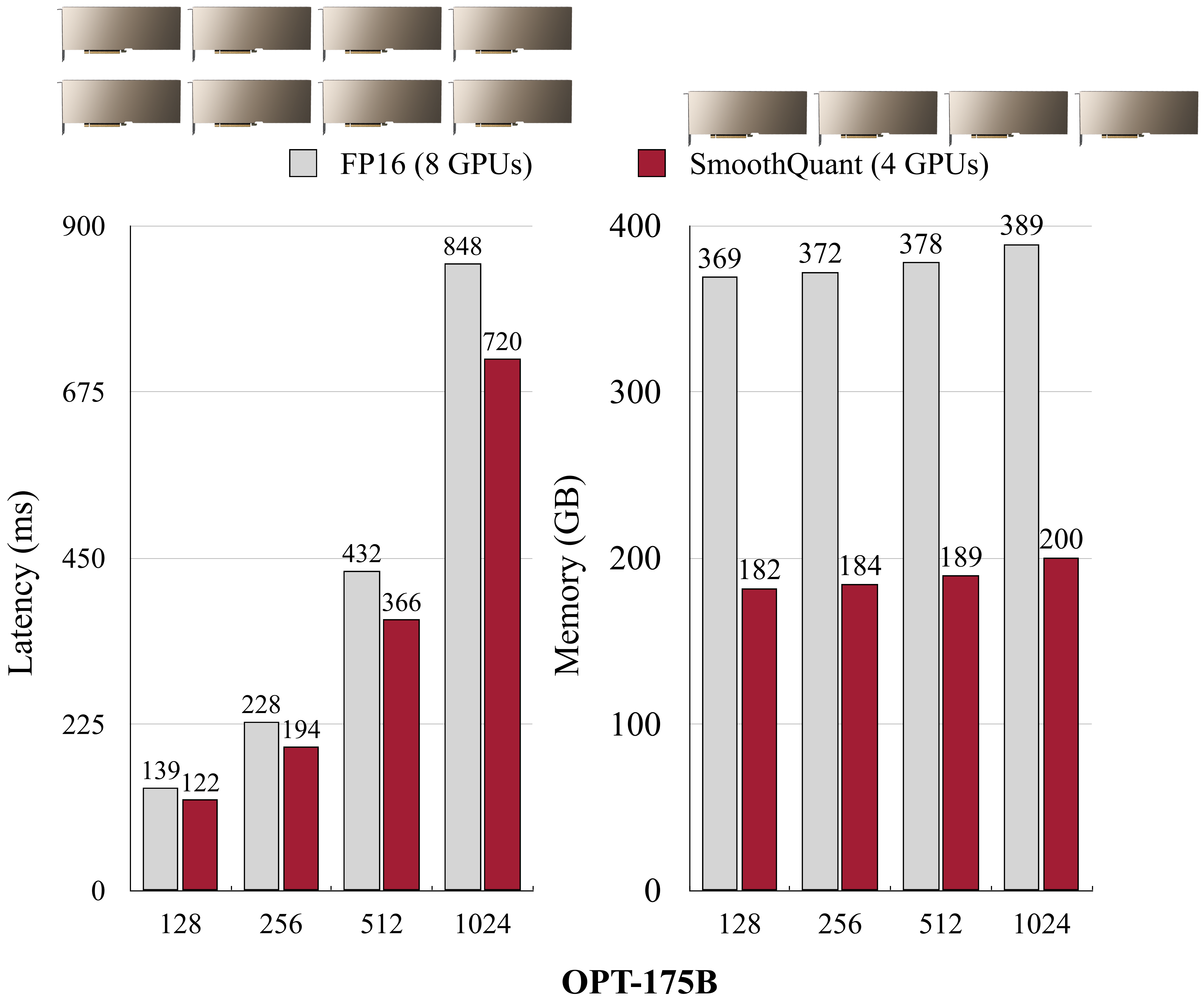
Method	Weight	Activation
W8A8	per-tensor	per-tensor dynamic
ZeroQuant	group-wise	per-token dynamic
`LLM.int8()`	per-channel	per-token dynamic+FP16
Outlier Suppression	per-tensor	per-tensor static
SmoothQuant-O1	per-tensor	per-token dynamic
SmoothQuant-O2	per-tensor	per-tensor dynamic
SmoothQuant-O3	per-tensor	per-tensor static

- SmoothQuant's precision mapping for a Transformer block.
- All compute-intensive operators, such as linear layers and batched matrix multiplications (BMMs) use INT8 arithmetic.

- Quantization setting of the baselines and SmoothQuant. All weight and activations use INT8 representations unless specified.
- We implement three efficiency levels of quantization settings for SmoothQuant. The efficiency improves from O1 to O3.

SmoothQuant is Accurate and Efficient

Method	OPT-175B	BLOOM-176B	GLM-130B*
FP16	71.6%	68.2%	73.8%
W8A8	32.3%	64.2%	26.9%
ZeroQuant	31.7%	67.4%	26.7%
`LLM.int8()`	71.4%	68.0%	73.8%
Outlier Suppression	31.7%	54.1%	63.5%
SmoothQuant-O1	**71.2%**	68.3%	**73.7%**
SmoothQuant-O2	71.1%	**68.4%**	72.5%
SmoothQuant-O3	71.1%	67.4%	72.8%



- SmoothQuant well maintains the accuracy without finetuning.
- SmoothQuant can both accelerate inference and halve the memory footprint.

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models (Xiao et al., 2022)

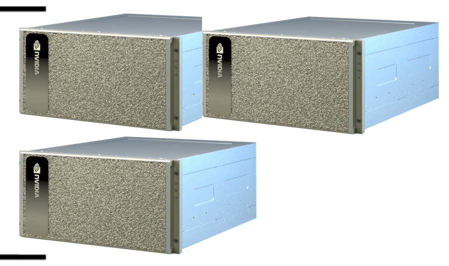
Scaling Up: 530B Model Within a Single Node

MT-NLG 530B Accuracy

	LAMBADA	HellaSwag	PIQA	WinoGrande	Average
FP16	76.6%	62.1%	81.0%	72.9%	73.1%
INT8	77.2%	60.4%	80.7%	74.1%	73.1%

MT-NLG 530B Efficiency

SeqLen	Prec.	#GPUs	Latency	Memory
512	FP16	16	838ms	1068GB
	INT8	8	839ms	545GB
1024	FP16	16	1707ms	1095GB
	INT8	8	1689ms	570GB



SmoothQuant can accurately quantize MT-NLG 530B model and reduce the serving GPU numbers by half at a similar latency, which allows serving the 530B model within a single node.

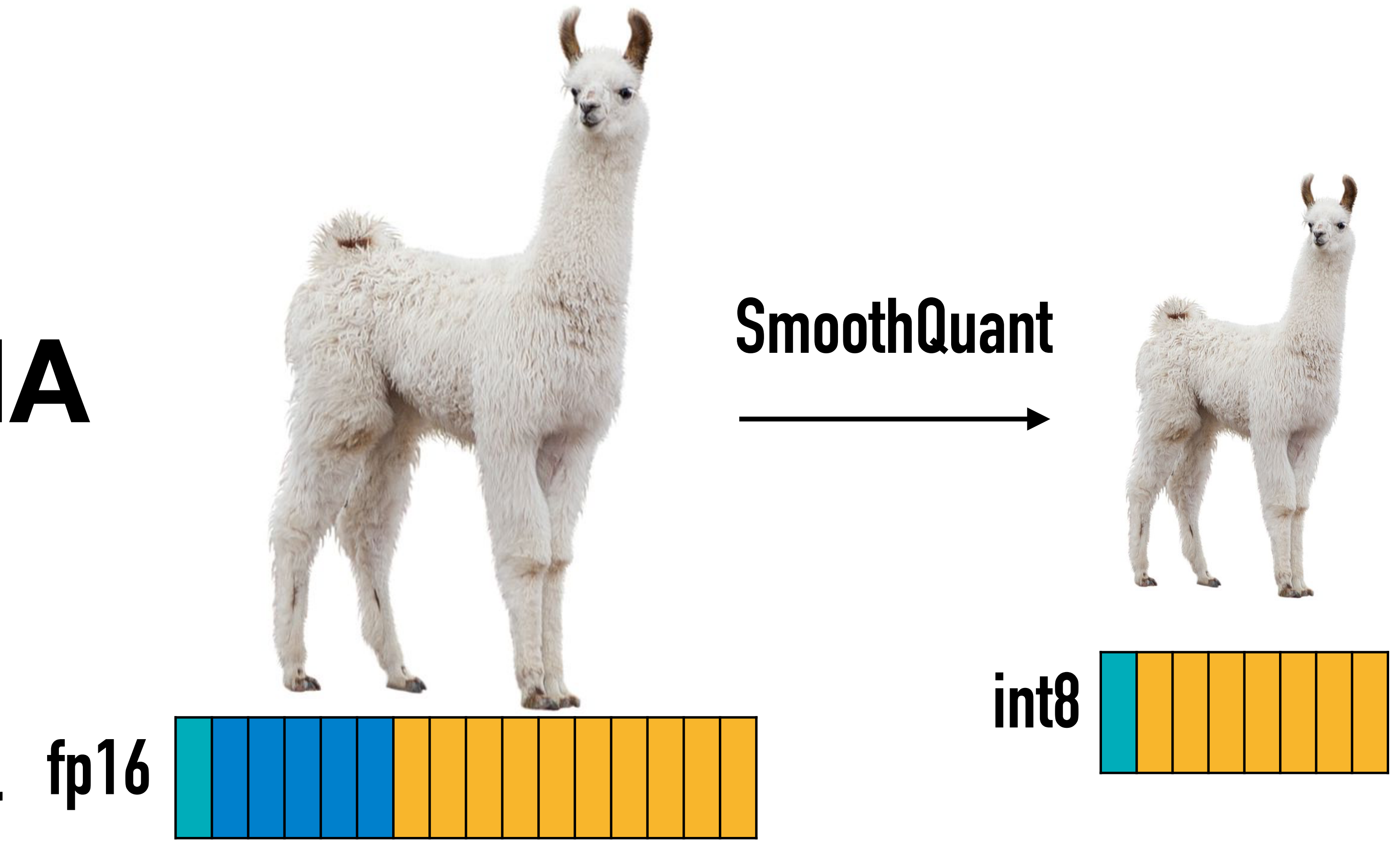
SmoothQuant on Instruction-Tuned LLMs

OPT-IML-30B	LAMBADA ↑	WikiText ↓
FP16	69.12%	14.26
W8A8	4.21%	576.53
ZeroQuant	5.12%	455.12
LLM.int8()	69.14%	14.27
Outlier Sppression	0.00%	9485.62
SmoothQuant-O1	69.94%	14.33
SmoothQuant-O2	69.51%	14.35
SmoothQuant-O3	69.77%	14.37

SmoothQuant works well on instruction-tuned LLM, the backbones of recent chat bots.

SmoothQuant

Advancing new efficient open model LLaMA



- **LLaMA** (and its successors like Alpaca) are popular open-source LLMs, which introduced SwishGLU, making activation quantization even harder
- SmoothQuant can losslessly quantize LLaMA families, further lowering the hardware barrier

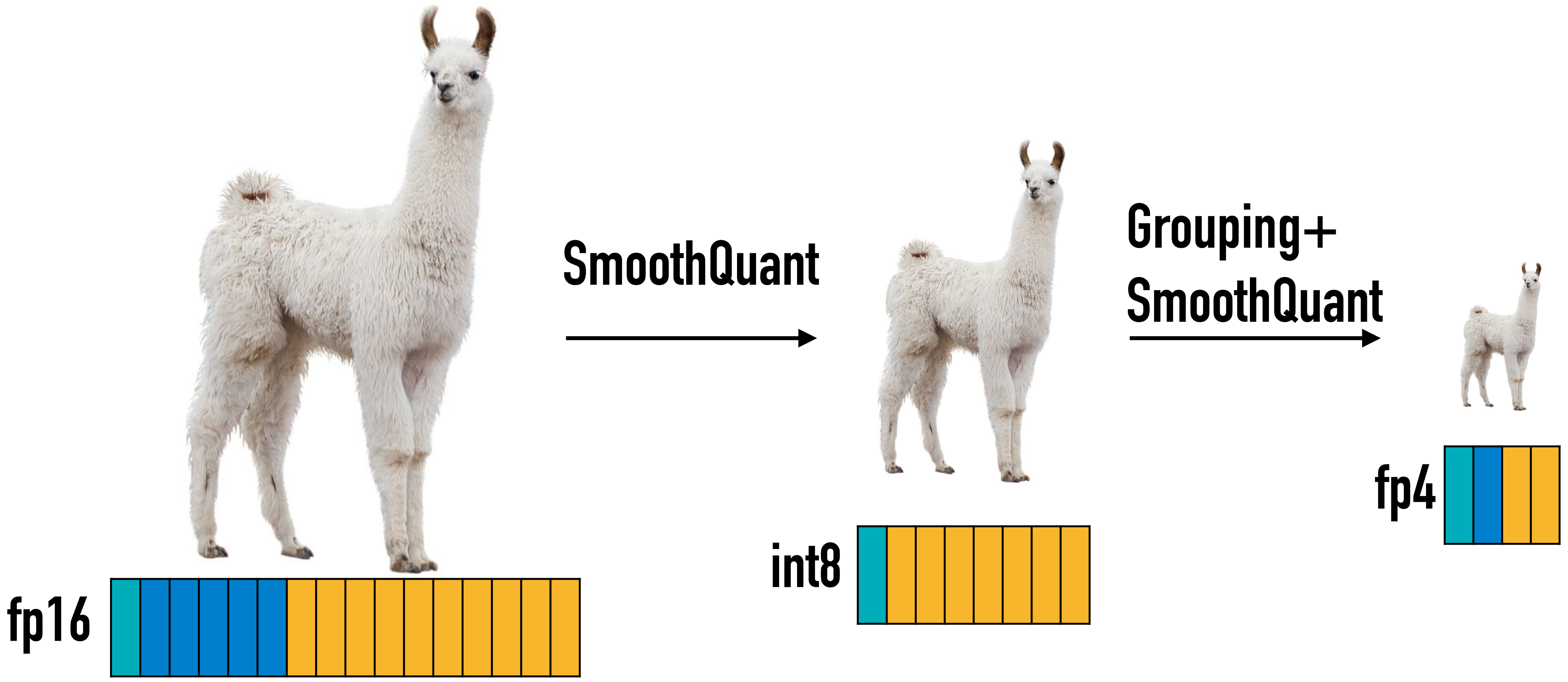
PIQA↑	LLaMA 7B	LLaMA 13B	LLaMA 30B	LLaMA 65B
FP16	78.24%	79.05%	80.96%	81.72%
SmoothQuant	78.24%	78.84%	80.74%	81.50%

Wikitext↓	LLaMA 7B	LLaMA 13B	LLaMA 30B	LLaMA 65B
FP16	11.51	10.05	7.53	6.17
SmoothQuant	11.69	10.31	7.71	6.68

W8A8 per token

SmoothQuant

Going smaller: W4A4 (FP4)



- Can we further push the frontier?
- We evaluate the W4A4 quantization
  - Setting: FP4 data type with a group size of 64; FP16 accumulator and FP16 scaling factor
- Red: ppl degrade > 0.5, Green: ppl degrade < 0.5. SmoothQuant helps most of the time.

wikitext2	llama-7b	llama-30b	llama-65b
fp16	9.49	6.91	4.96
w4a4-m1e2-g64	10.2676	8.1453	5.4746
w4a4-m1e2-g64-sm0.25	10.1437	7.0089	5.4336
	opt-6.7b	opt-13b	opt-30b
fp16	15.12	14.13	13.09
w4a4-m1e2-g64	16.2289	14.7355	13.6172
w4a4-m1e2-g64-sm0.25	15.5899	14.5469	13.3931

Conclusion

- We propose SmoothQuant, a turn-key solution to enable accurate W8A8 quantization for large language models.
- SmoothQuant is accurate and efficient on existing hardware. We can implement SmoothQuant with off-the-shelf kernels to achieve high speedup and memory saving.
- Integration
  - NVIDIA: FasterTransformer
  - Intel: Neural Compressor
  - OpenNMT: CTranslate2

- Paper: https://arxiv.org/abs/2211.10438
- Code: https://github.com/mit-han-lab/smoothquant