CS 498: Machine Learning System Spring 2026

Minjia Zhang

The Grainger College of Engineering

Recall: Adaptive KV Cache

- Heavy Hitter (H2O): not all KV cache (tokens) are equally important
  - Keep the ones with high attention score to cap memory usage

- Attention Sink:
  - Keep the first several tokens and recent tokens
  - Highly effective yet hard to interpret

Agenda: Quantization

Inference Optimization (Part 8)

- LLM Quantization

Learning Objective:

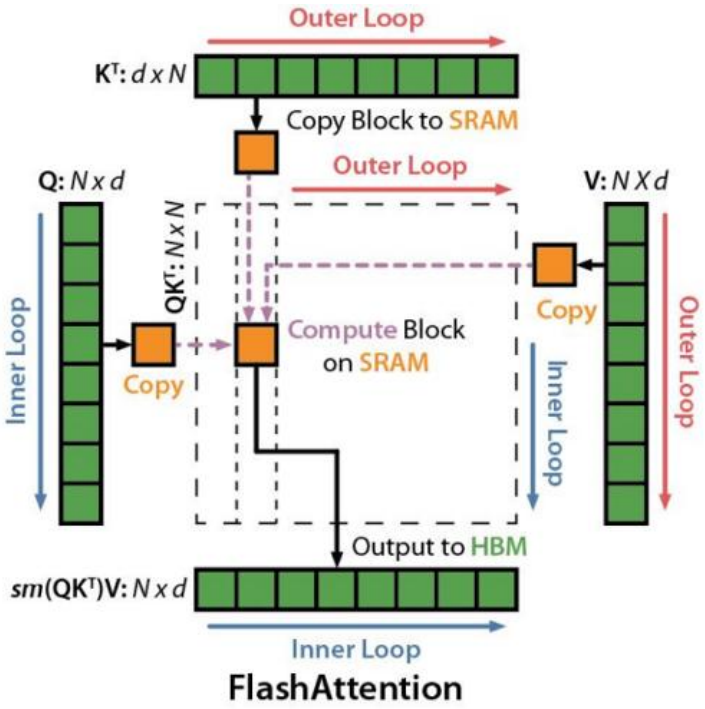
- Understand how LLM quantization reduces memory and compute cost
- Learn the core ideas behind SmoothQuant
- Analyze the trade-offs in accuracy and efficiency introduce by quantization

Reminder: Lab Assignment 3 due on April 26th

Gap Between the Supply and Demand of LLMs

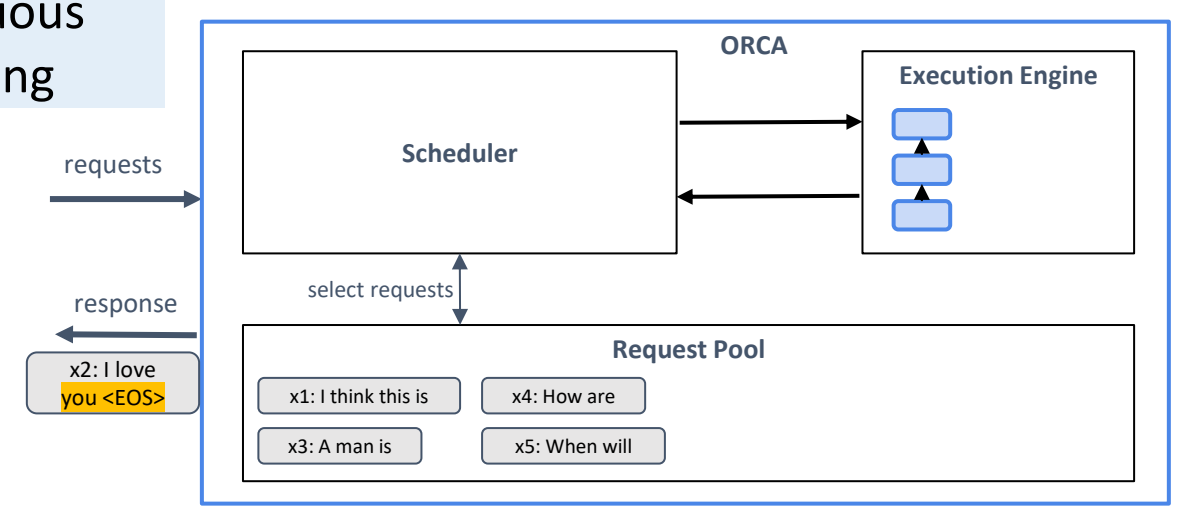
- Size of LLMs is increasing faster than GPU memory: **exponential growth in model size** vs. linear increase in GPU memory
- **Memory challenges:** 175B parameters in GPT-3 requires 350GB of memory to store just weights
- **Deployment difficulties:** Requires multiple GPUs, high latency, and resource intensive setups

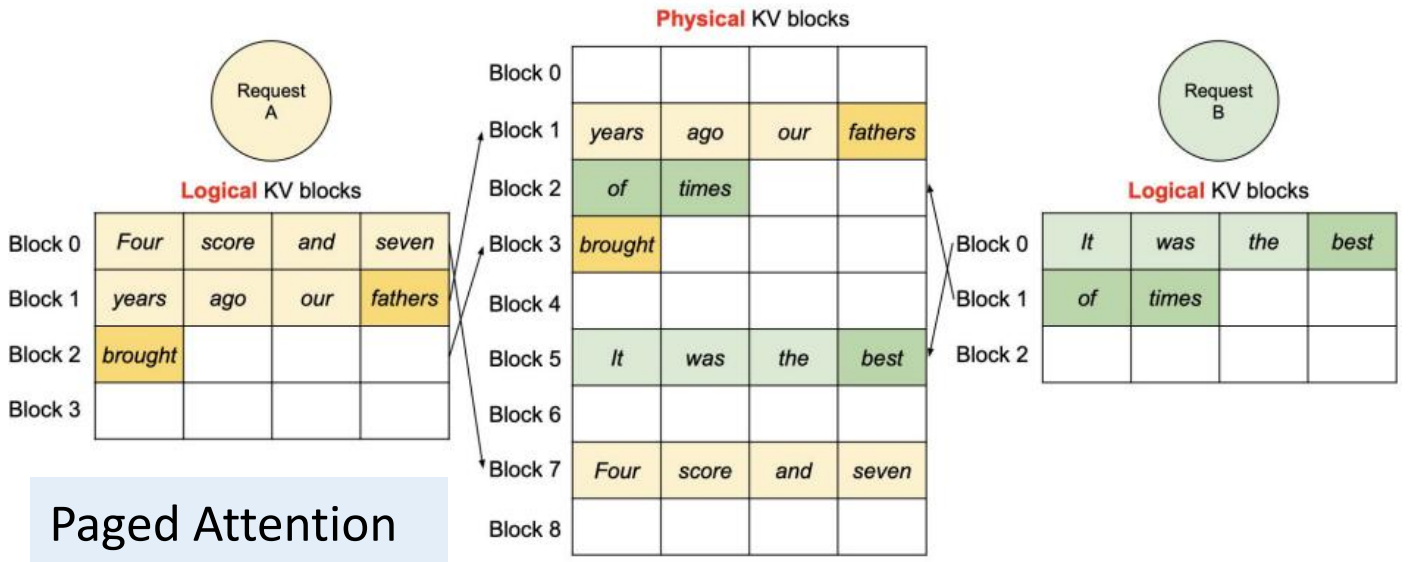
Do Previously Introduced Techniques Reduce LLM Memory?



Flash Attention

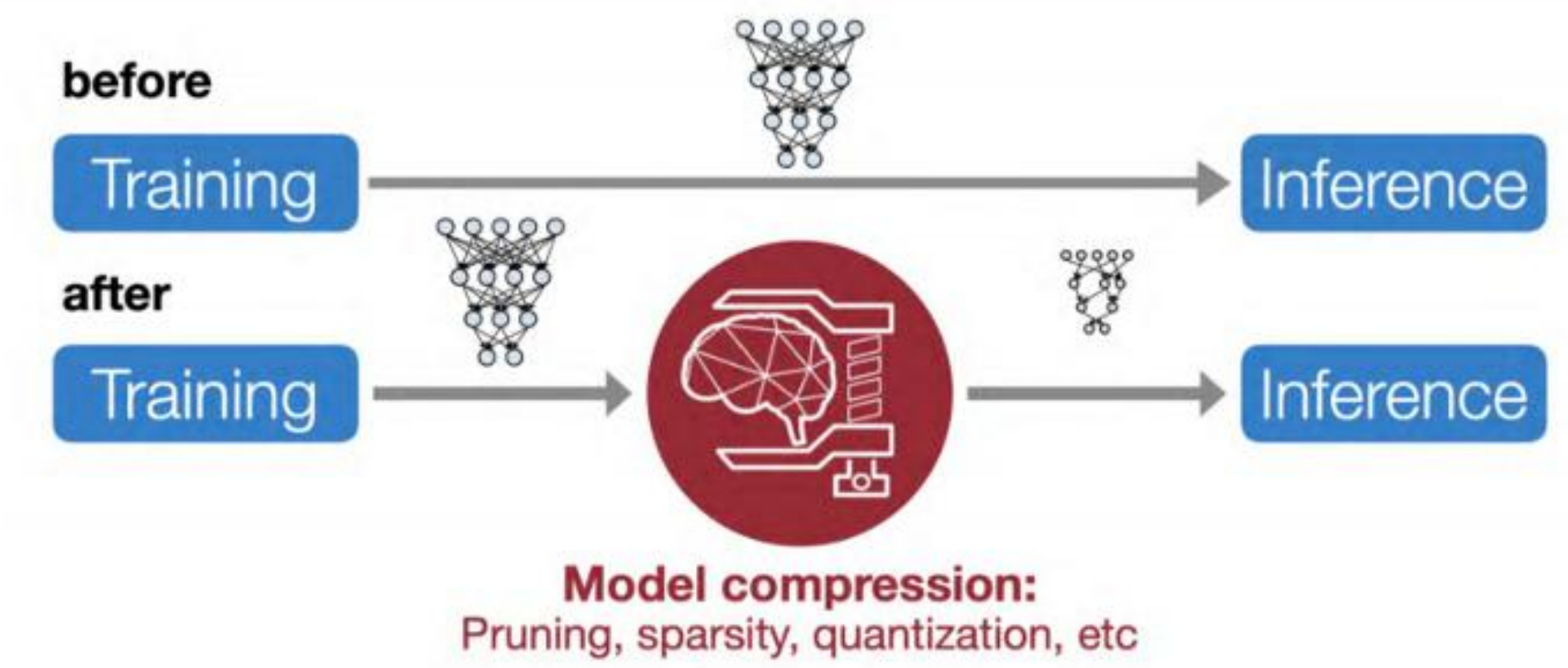
Continuous Batching





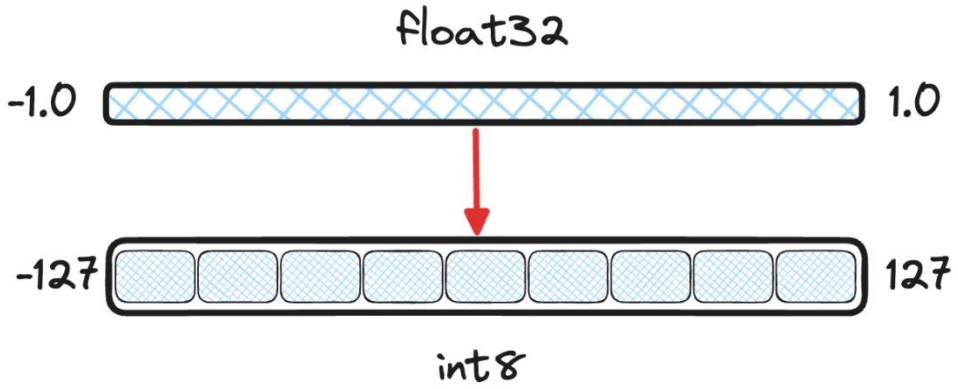
Paged Attention

Bridge the Gap through Model Compression



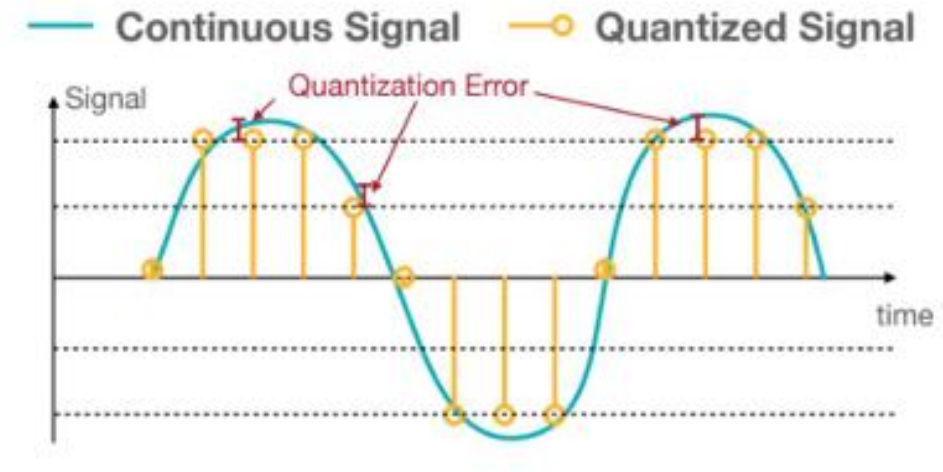
Quantization

- *Quantization is the process of mapping a large set of continuous or high-precision values (typically floating-point numbers) to a smaller set of discrete values (typically lower-precision integers) in order to reduce the computational and memory requirements of a model.*



Quantization

- *Quantization is the process of mapping a large set of continuous or high-precision values (typically floating-point numbers) to a smaller set of discrete values (typically lower-precision integers) in order to reduce the computational and memory requirements of a model.*

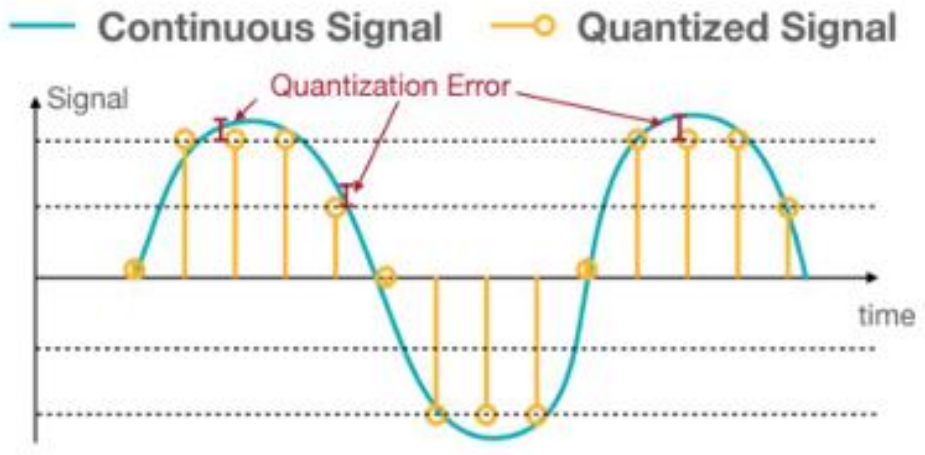


The difference between an input value and its quantized value is referred to as quantization error.

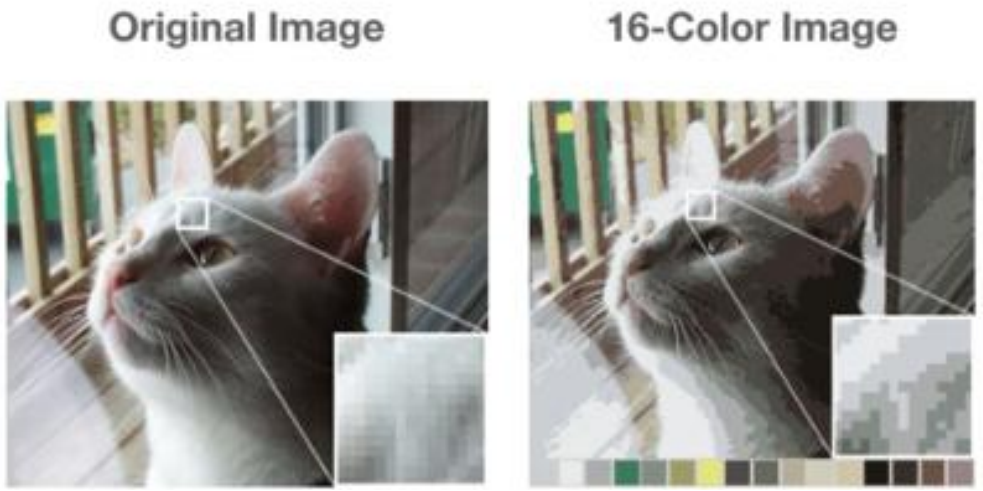
Originate from signal processing and information theory

Quantization

- *Quantization is the process of mapping a large set of continuous or high-precision values (typically floating-point numbers) to a smaller set of discrete values (typically lower-precision integers) in order to reduce the computational and memory requirements of a model.*



The difference between an input value and its quantized value is referred to as quantization error.



Originate from signal processing and information theory

Adapted to computer vision models in ML

Motivation for LLM Quantization

- LLMs often come in high precision formats such as FP16/BF16/FP32 and 10B → 1T+ parameters
  - Significant GPU memory requirements (memory capacity and bandwidth)

Motivation for LLM Quantization

- LLMs often come in high precision formats such as FP16/BF16/FP32 and 10B → 1T+ parameters
  - Significant GPU memory requirements (memory capacity and bandwidth)
- Discussion: Can we train LLMs directly in lower precision (8-bit/4-bit)? If so, how are you going to do it, and why?

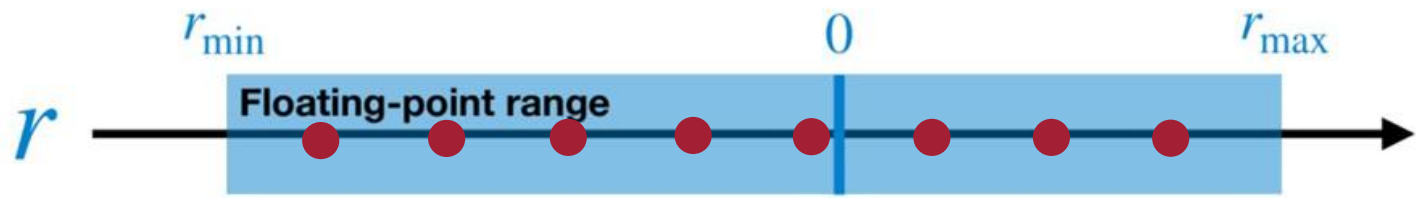
Motivation for LLM Quantization

- LLMs often come in high precision formats such as FP16/BF16/FP32 and 10B → 1T+ parameters
  - Significant GPU memory requirements (memory capacity and bandwidth)
- Discussion: Can we train LLMs directly in lower precision (8-bit/4-bit)? If so, how are you going to do it, and why?

Mixed-Precision Training (Training Series, Part 6)

Quantization: Problem

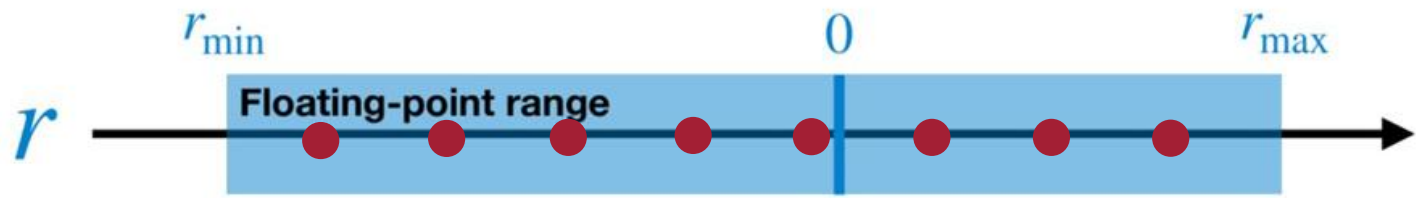
Quantization: Constraints



Question: Can we just round to integers?

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

Quantization: Constraints



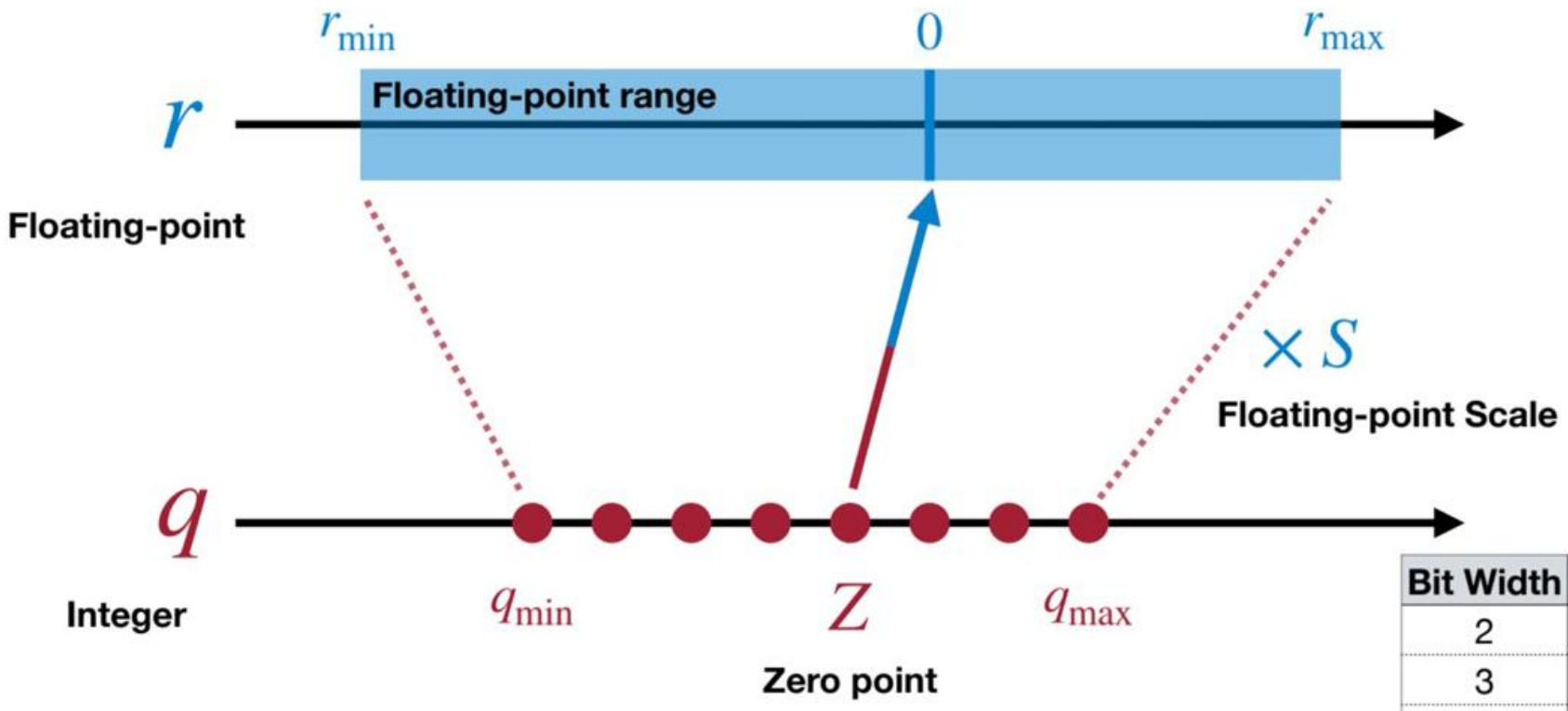
Question: Can we just round to integers?

Massive information loss

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

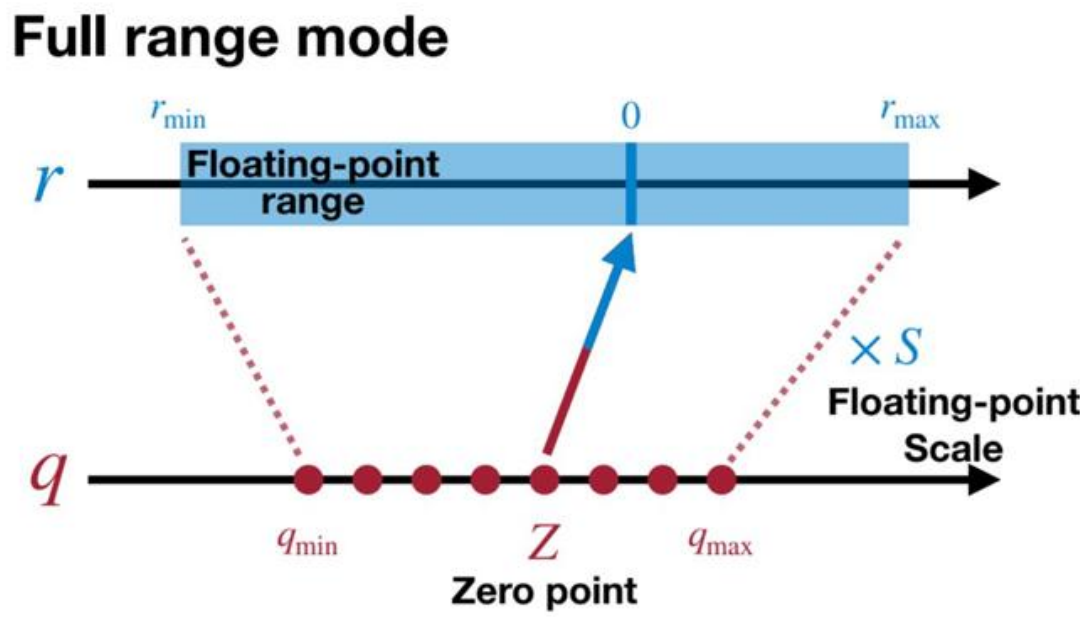
Quantization: Linear Quantization + Round-to-Nearest

- 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}	$2^{N-1}-1$

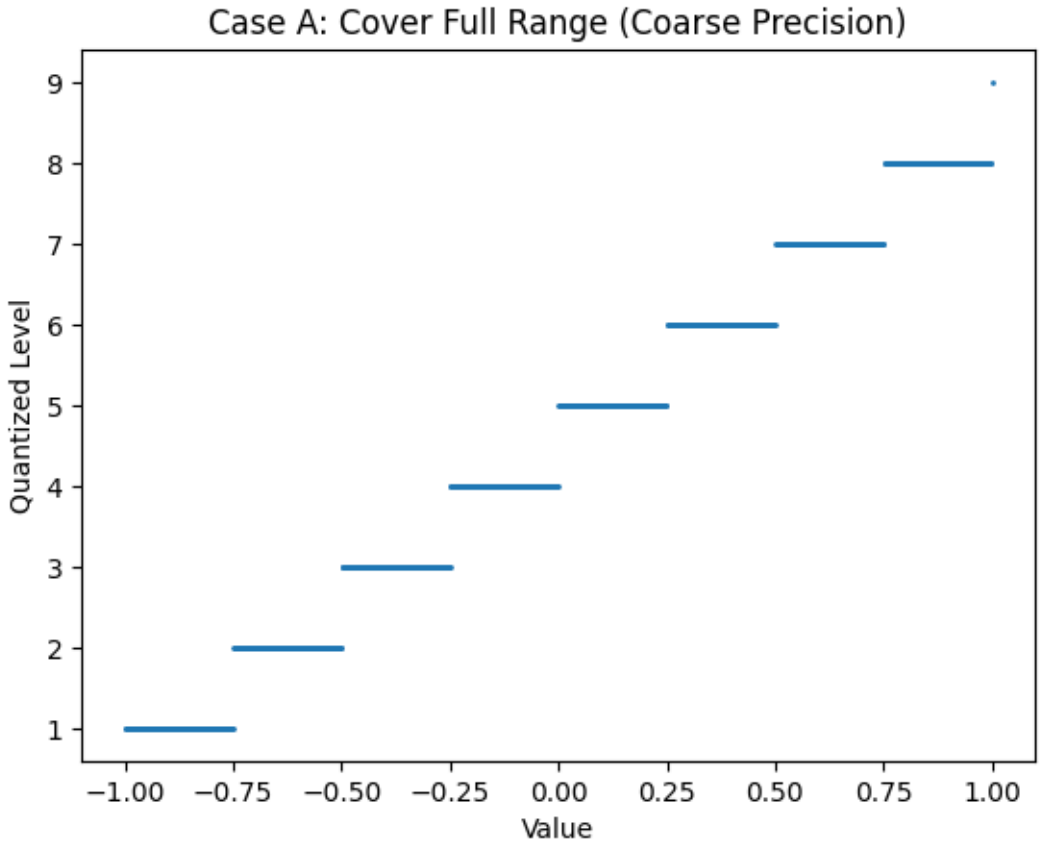
Quantization: Linear Quantization + Round-to-Nearest



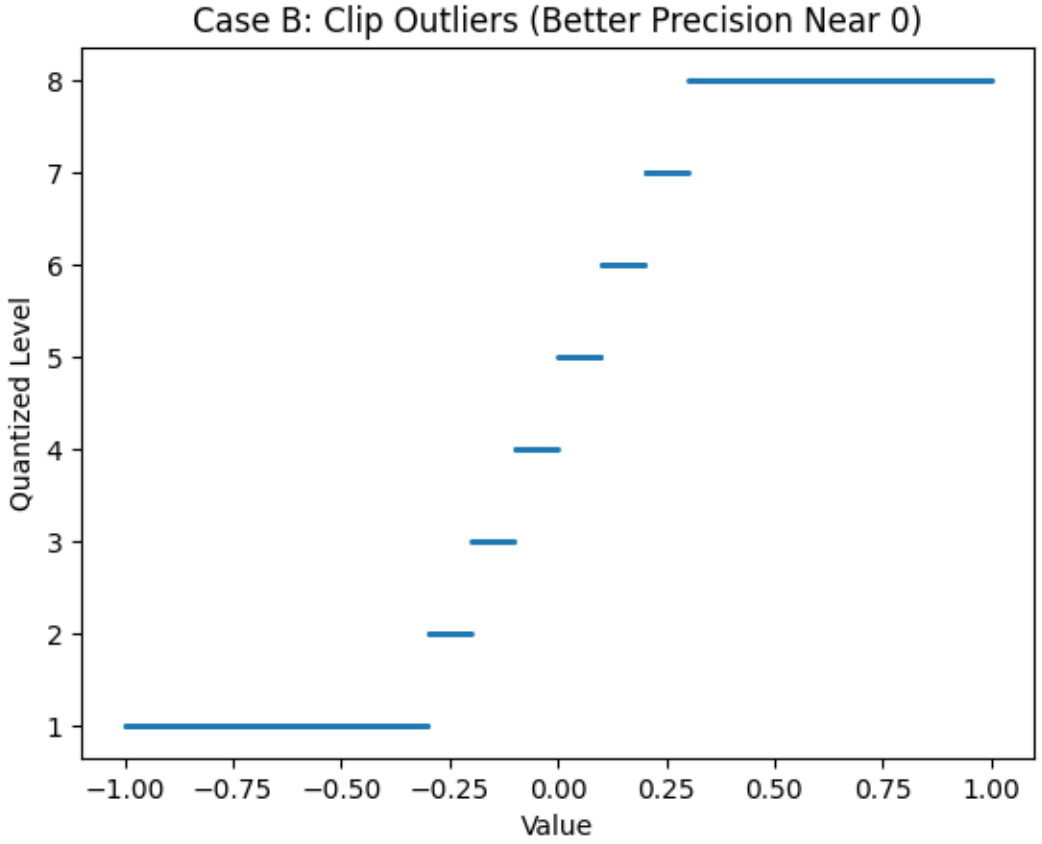
Bit Width	q_{min}	q_{max}
2	-2	1
3	-4	3
4	-8	7
N	-2^{N-1}	$2^{N-1}-1$

Without scaling, most values would collapse to 0, so scaling is necessary, but how?

Quantization: Range vs. Precision Tradeoff



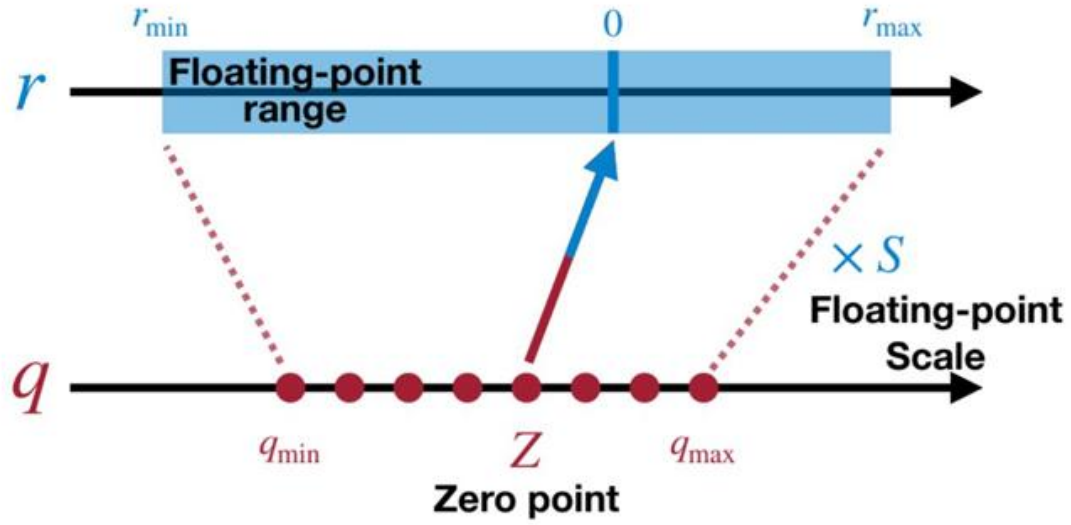
Full range coverage
Large bucket size
Low precision near 0

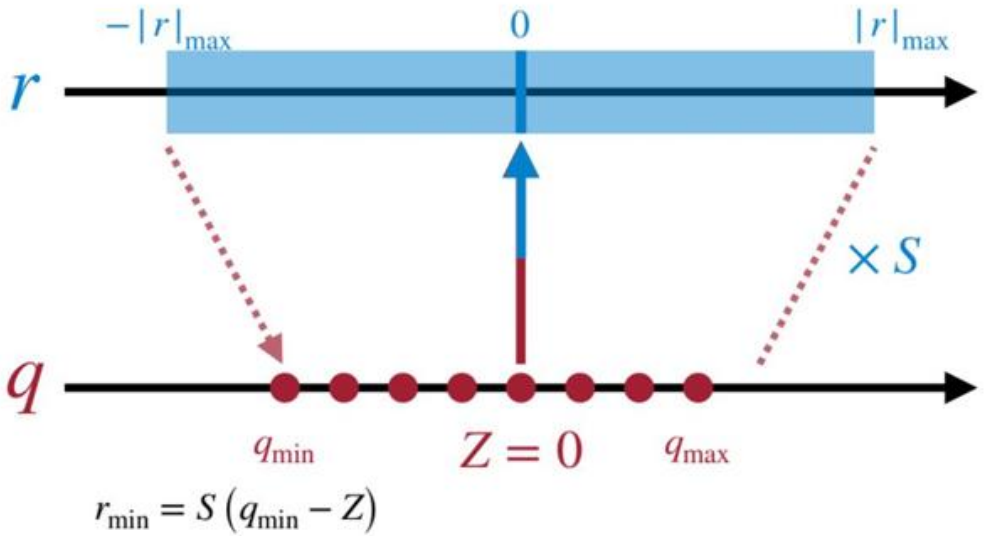


Partial range (outliers clipped)
Small bucket size
Better precision near 0

Quantization: Asymmetric vs. Symmetric

Full range mode





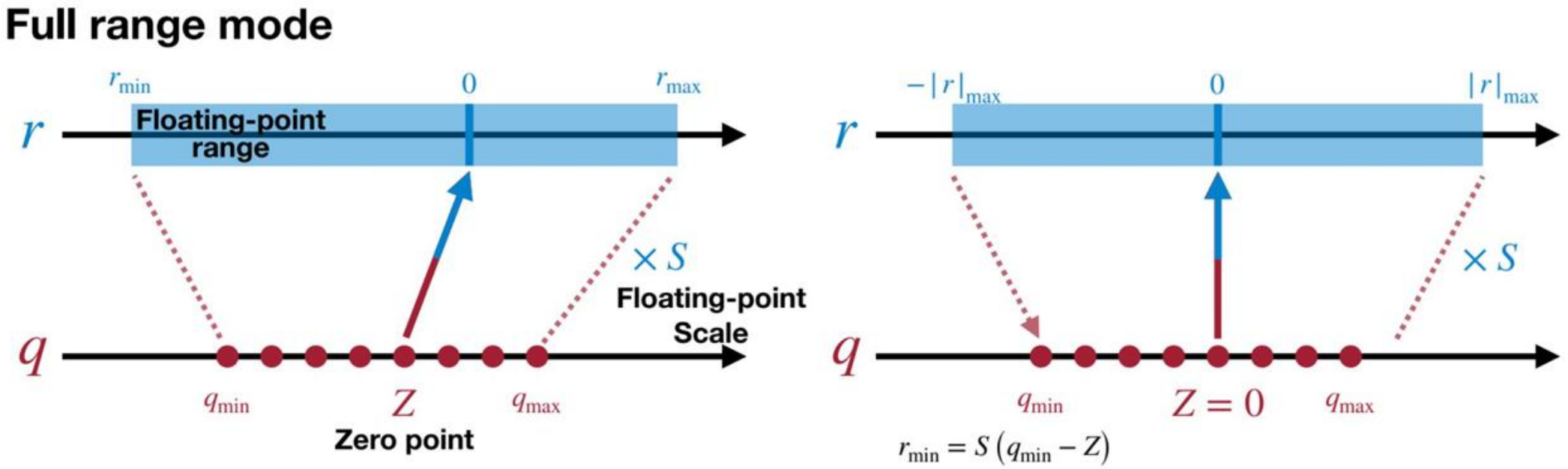
$$r_{min} = S\,(q_{min} - Z)$$

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

$$S = \frac{r_{max} - r_{min}}{q_{max} - q_{min}}$$

$$S = \frac{r_{min}}{q_{min} - Z} = \frac{-|r|_{max}}{q_{min}} = \frac{|r|_{max}}{2^{N-1}}$$

Quantization: Asymmetric vs. Symmetric



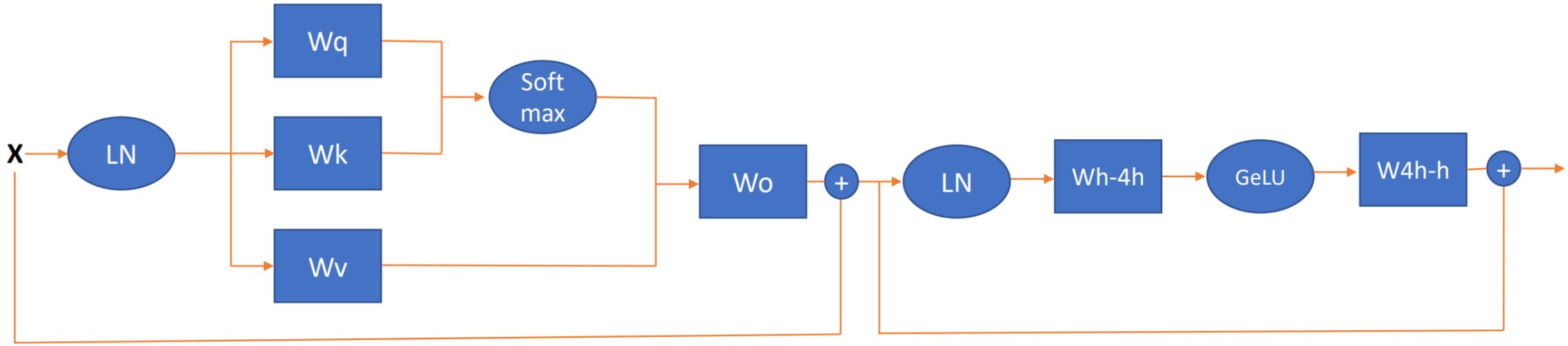
Bit Width	q_{min}	q_{max}
2	-2	1
3	-4	3
4	-8	7
N	-2^{N-1}	$2^{N-1}-1$

$$S = \frac{r_{max} - r_{min}}{q_{max} - q_{min}}$$

$$S = \frac{r_{min}}{q_{min} - Z} = \frac{-|r|_{max}}{q_{min}} = \frac{|r|_{max}}{2^{N-1}}$$

Next: How to apply quantization to LLMs

LLM Quantization

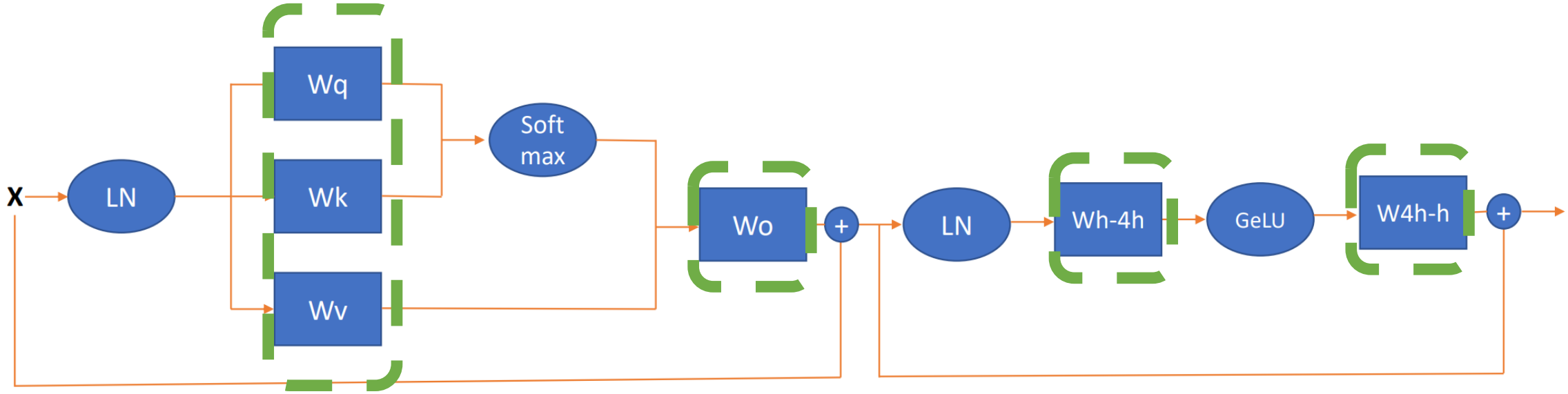


- 8-bit quantization

$$\mathbf{x}_{quantize} = round\left(clamp(\frac{\mathbf{x}}{S}, -2^{bit-1}, 2^{bit-1}-1)\right)$$

Where can we quantize without breaking the model?

LLM 8-bit Quantization: Weights



- 8-bit weight quantization

$$\mathbf{x}_{quantize} = round\left(clamp(\frac{\mathbf{x}}{S}, -2^{bit-1}, 2^{bit-1}-1)\right)$$

FP32 weight matrix

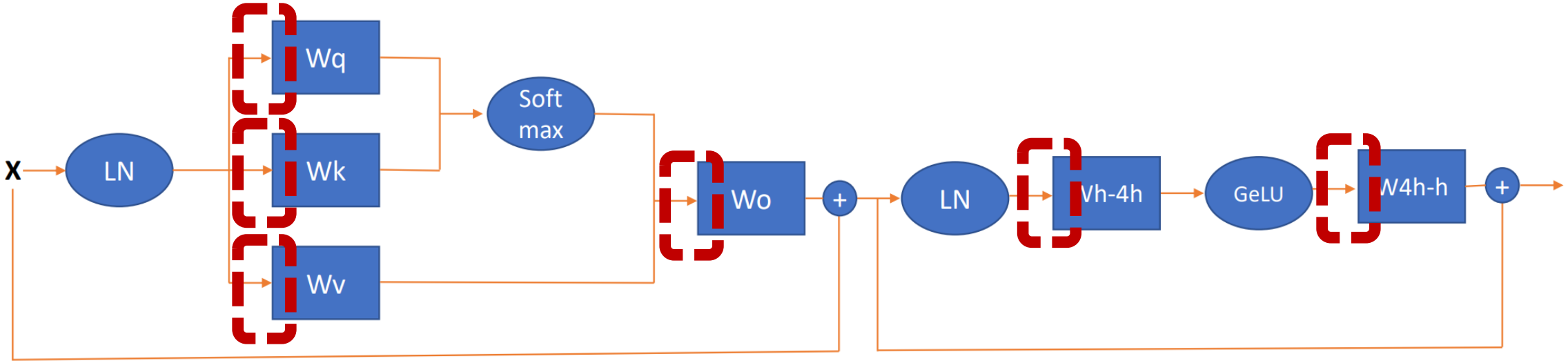
1.1	2.2	0.1	-0.1	-5.5	-6.6
…					
…					
…					
…					
1.1	2.1	0.1	-0.1	-4.8	-6.6

Scaling Factor $1/S$ ≈ 0.05 *

8-bit quantization

21	42	2	-2	-106	-127
…					
…					
…					
…					
21	40	2	-2	-92	-127

LLM 8-bit Quantization: Activation



- 8-bit activation
  (Input to the linear layer)

$$\mathbf{x}_{quantize} = round\left(clamp(\frac{\mathbf{x}}{S}, -2^{bit-1}, 2^{bit-1}-1)\right)$$

FP32 activation matrix

1.1	2.2	0.1	-0.1	-5.5	-6.6
…					
…					
…					
…					
1.1	2.1	0.1	-0.1	-4.8	-6.6

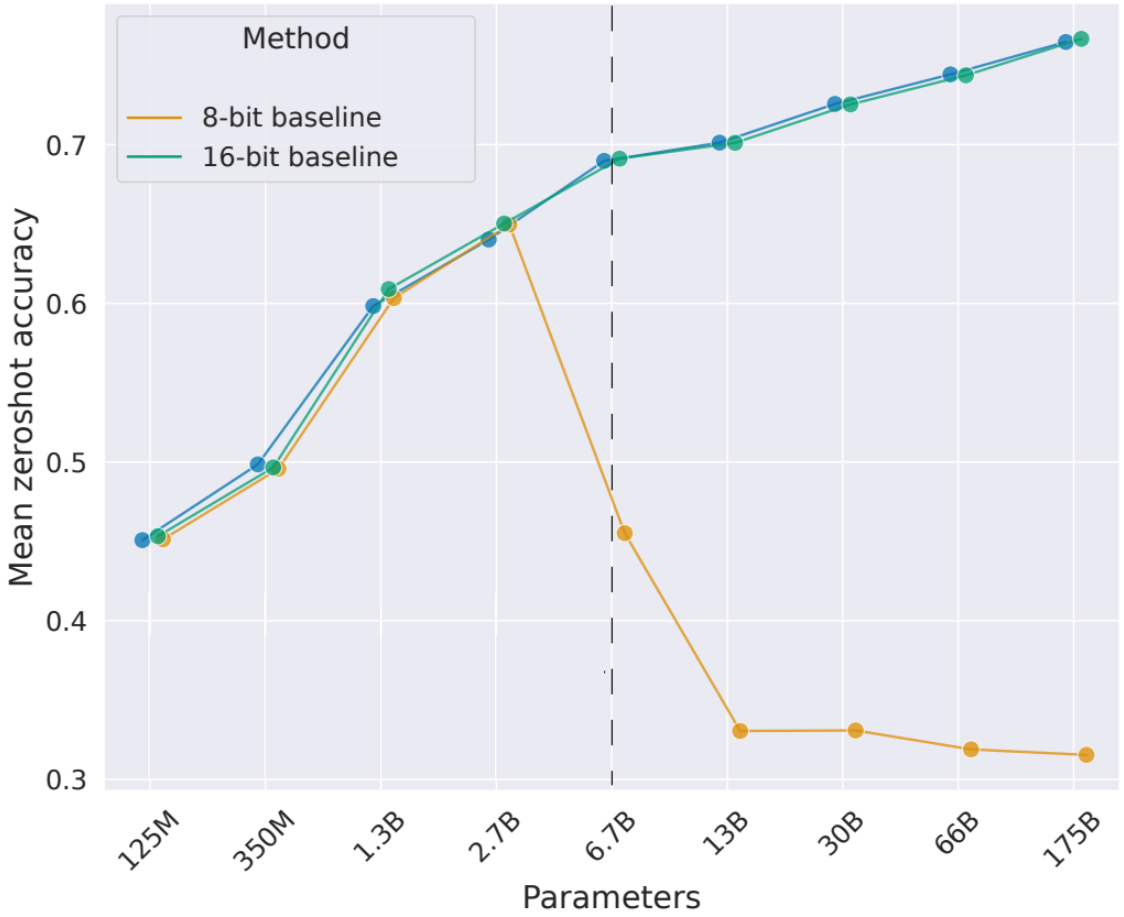
Scaling Factor $1/S$ $\approx$ 0.05*

8-bit quantization

21	42	2	-2	-106	-127
…					
…					
…					
…					
21	40	2	-2	-92	-127

LLM Quantization Challenges

- Standard quantization strategy leads to catastrophic accuracy drop

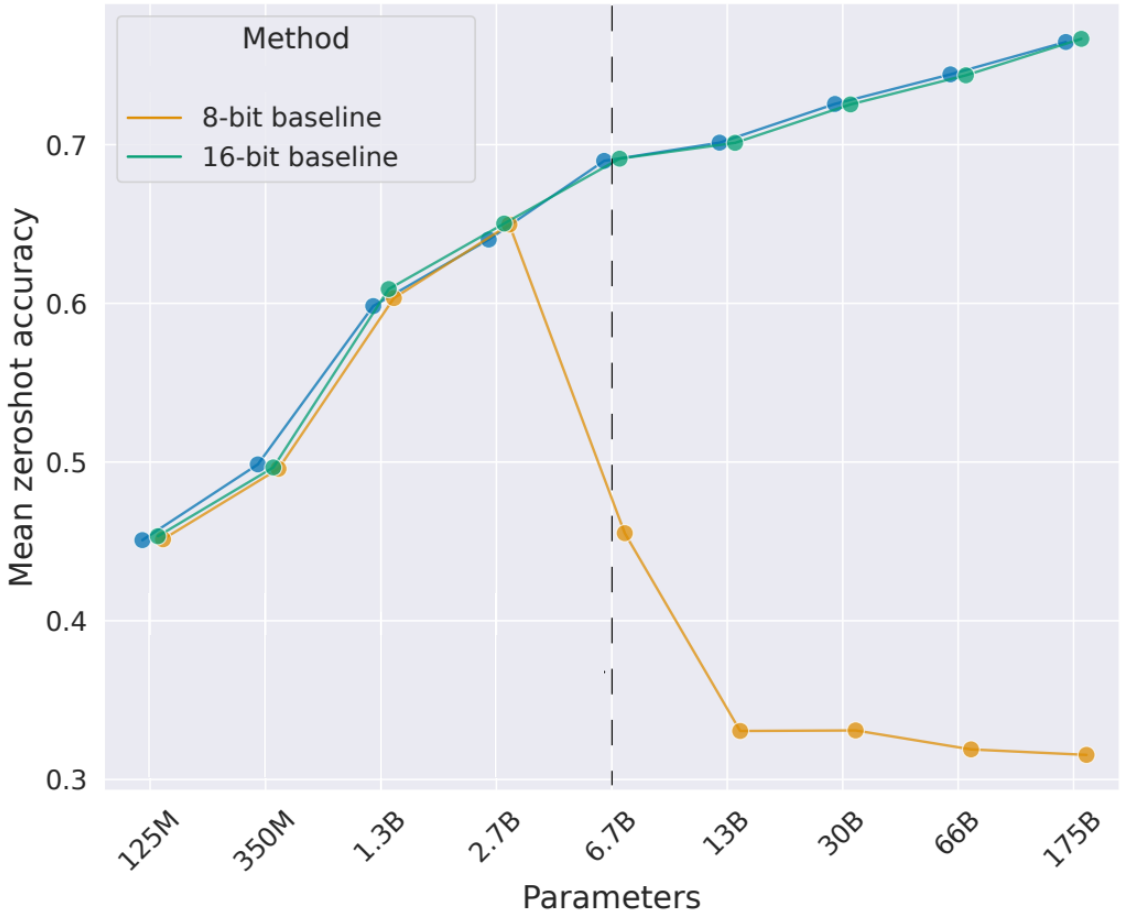


LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale, 2023

LLM Quantization Challenges

- Standard quantization strategy leads to catastrophic accuracy drop

Discussion: Something fundamentally changs in large LLMs. What changes as models get larger?



LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale, 2023

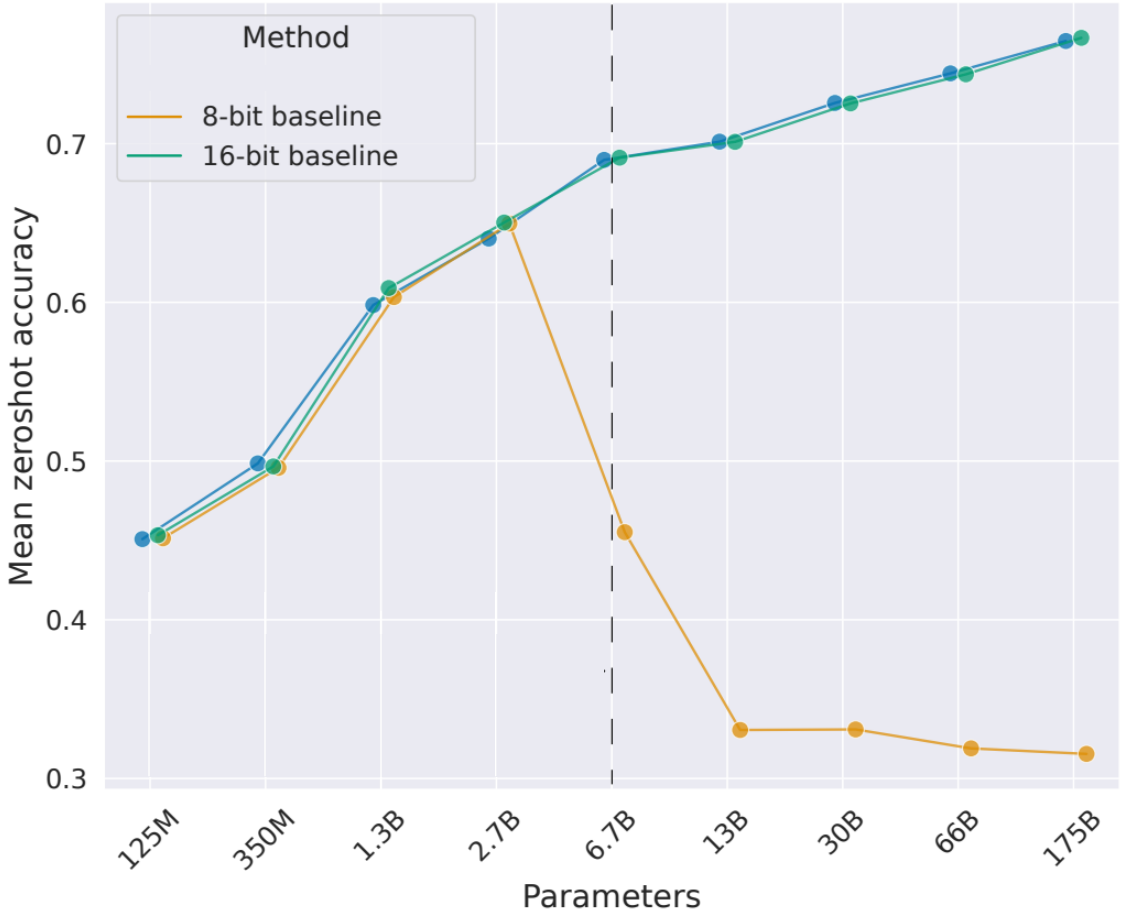
LLM Quantization Challenges

- Standard quantization strategy leads to catastrophic accuracy drop

Output = Activation X Weight

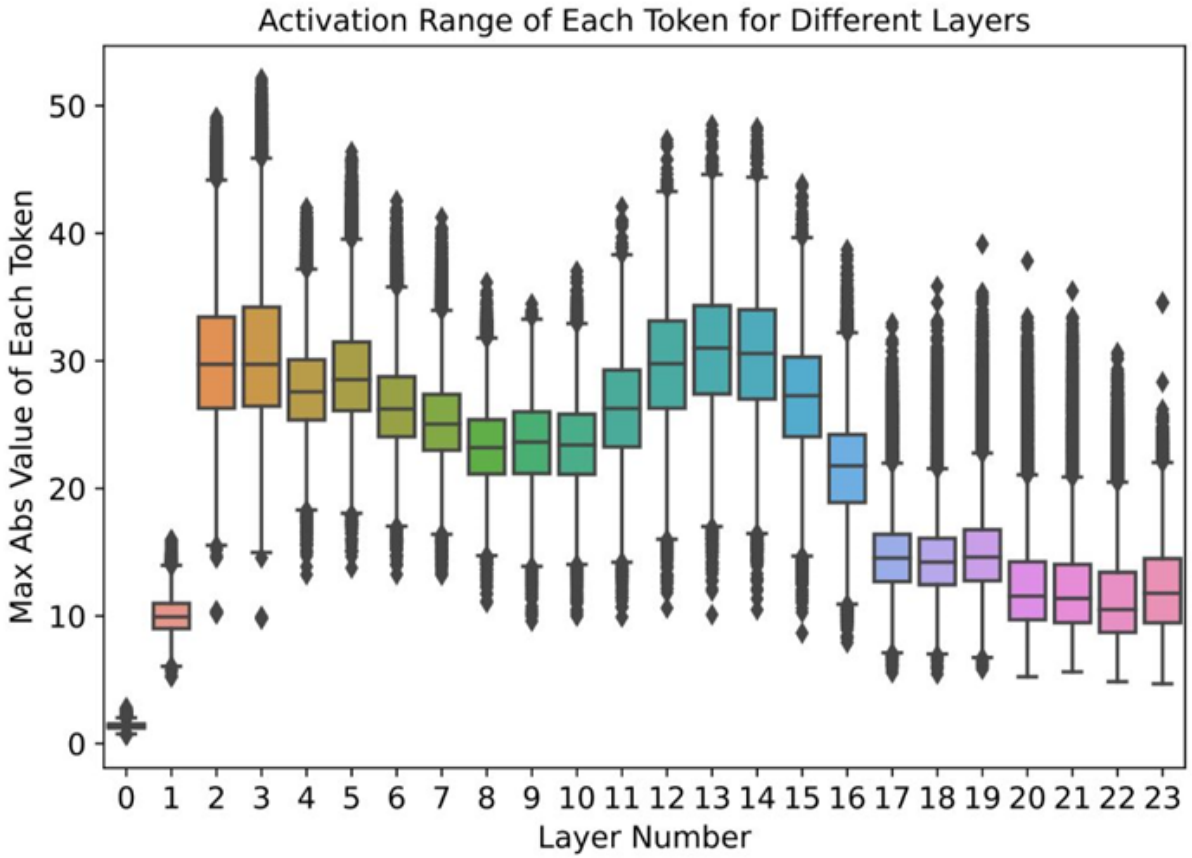
Precision	Lambada (↑)
W16A16	49.3
W8A16	49.3
W16A8	44.7
W8A8	42.6

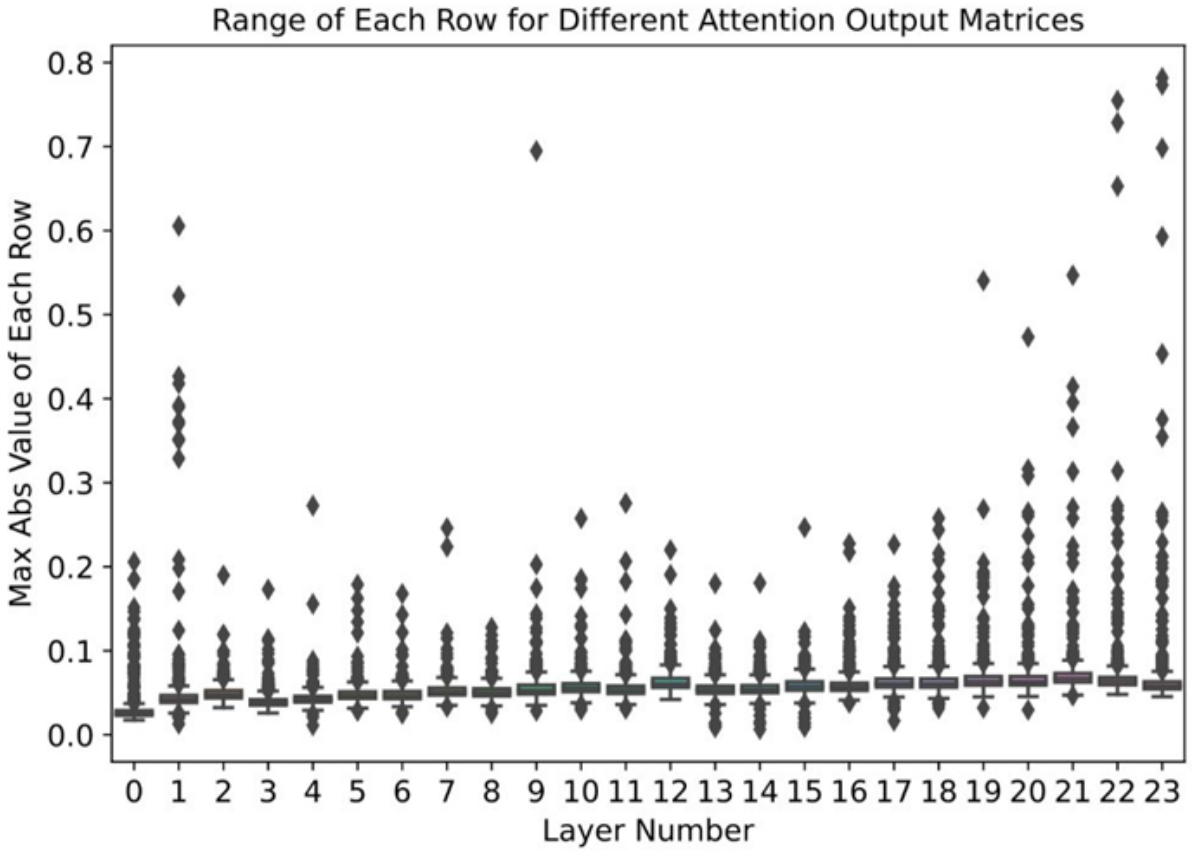
WxAy means x-/y-bit for weight/activation.



INT8 activation quantization causes the primary accuracy loss

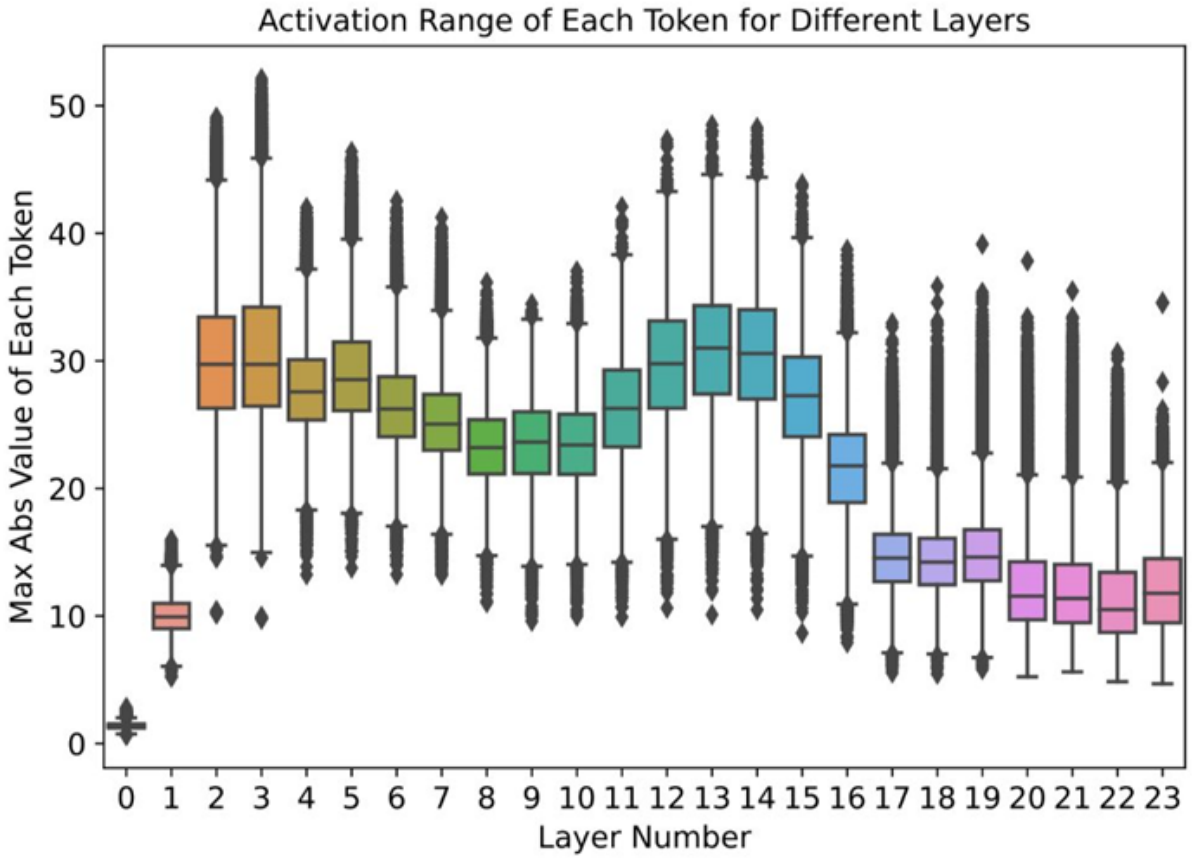
LLM Quantization Challenges

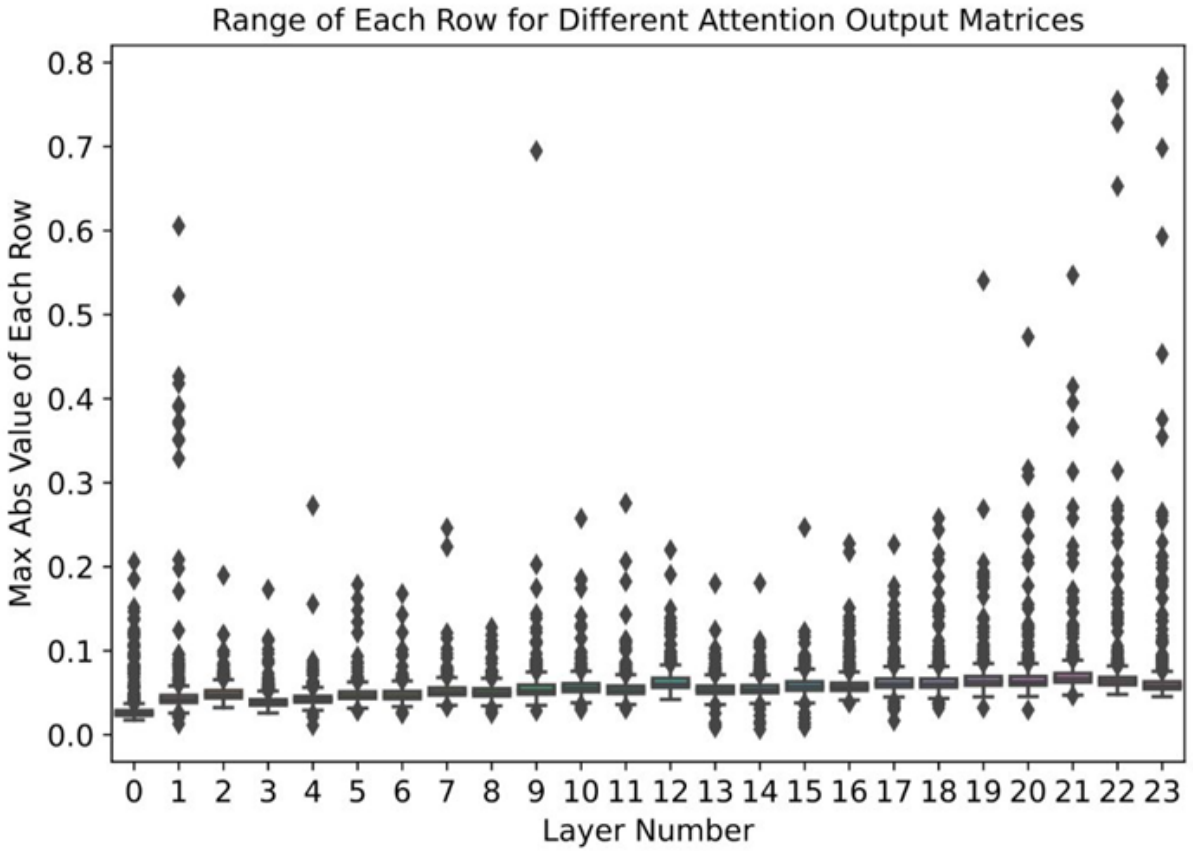




ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers, 2023

LLM Quantization Challenges



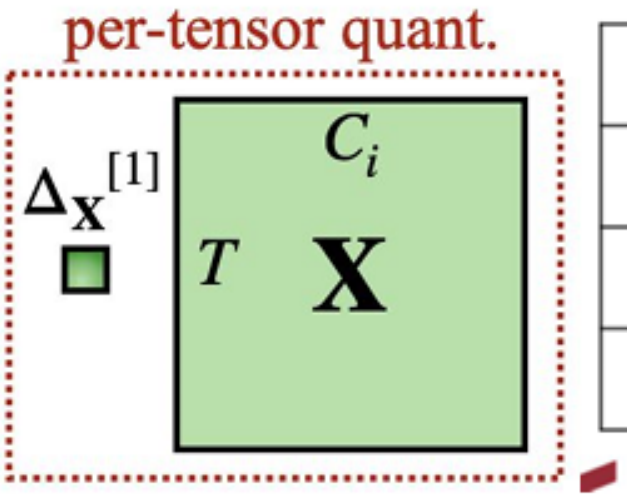


Questions: How do we mitigate accuracy loss from large dynamic range?

Quantization Can Have Different Granularities

- Per-Tensor Quantization
  - Uses a single scaling factor for the entire matrix

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



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

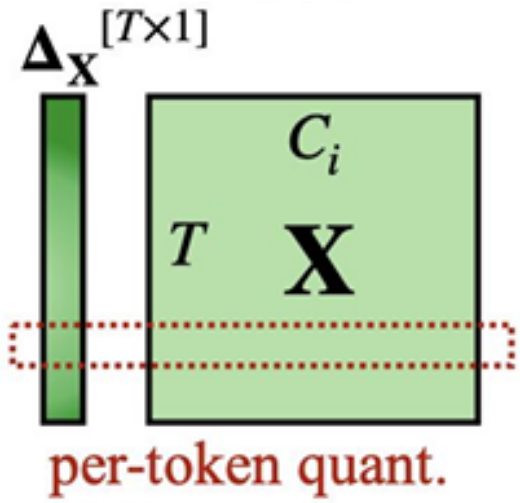
Quantized

2.12	0	2.12	0
0	0	-2.12	2.12
0	2.12	0	0
2.12	0	2.12	2.12

Reconstructed

Quantization Granularity

- Per-Tensor Quantization
- Per-Token Quantization
  - Uses different scaling factor for activations associated with each token
- Per-Channel (e.g., weight column) Quantization
- Group-wise Quantization



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

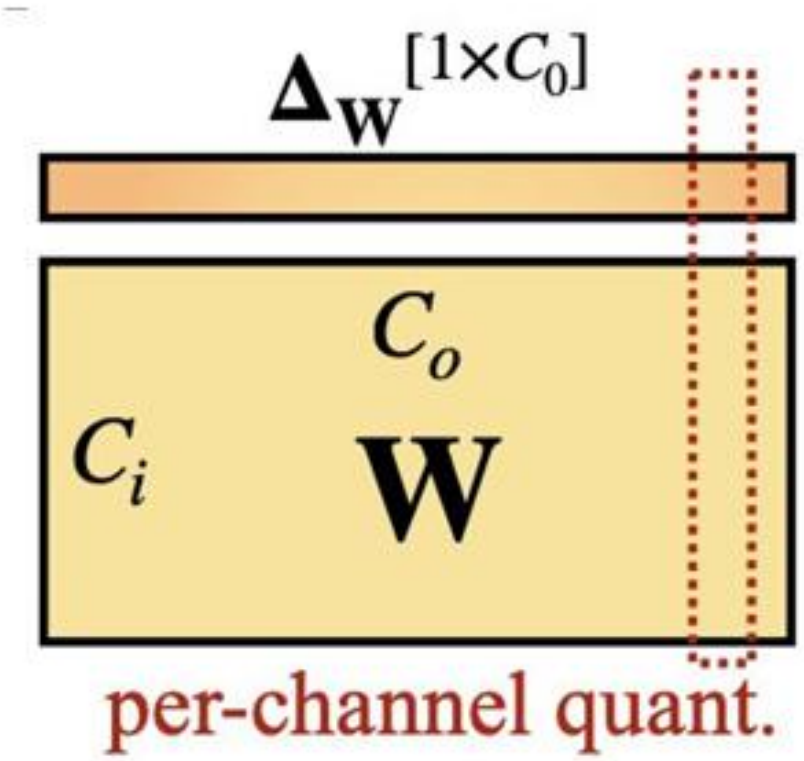
Quantized

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

Reconstructed

Quantization Granularity

- Per-Tensor Quantization
- Per-Token Quantization
- Per-Channel (e.g., weight column) Quantization
  - Uses different scaling factors for each weight channels
- Group-wise Quantization



Quantization Granularity

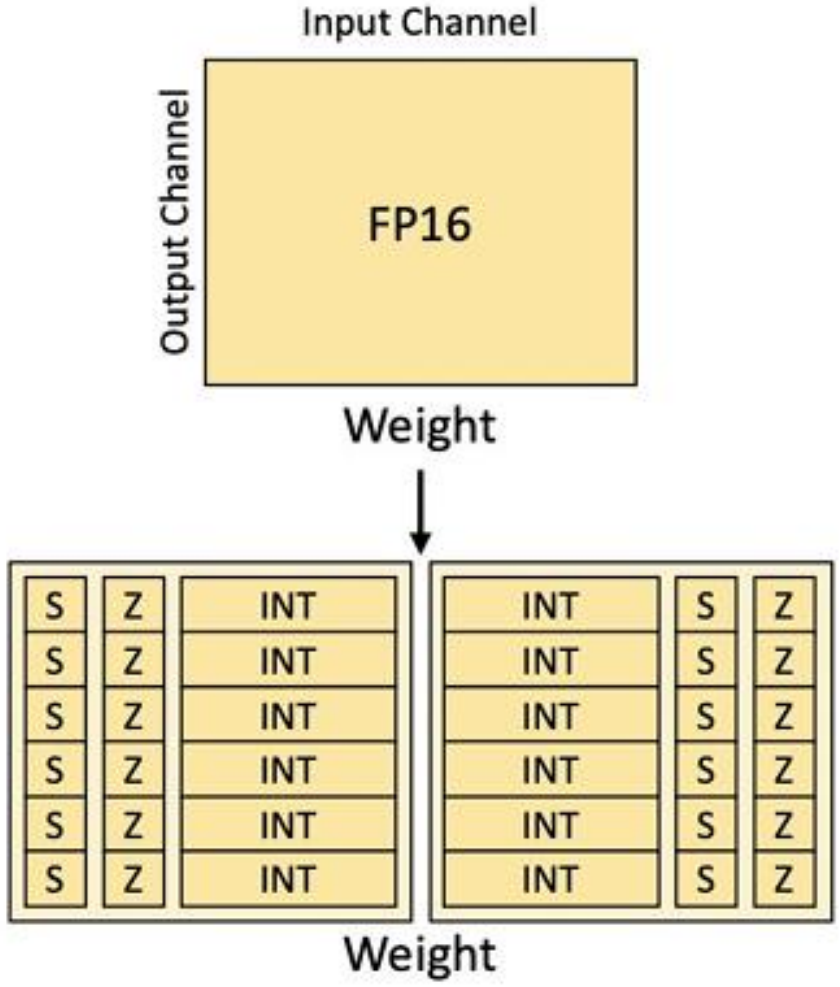
- Per-Tensor Quantization
- Per-Token Quantization
- Per-Channel (e.g., weight column) Quantization
  - Uses different scaling factors for each weight channels
- Group-wise Quantization

Model size (OPT-)	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%

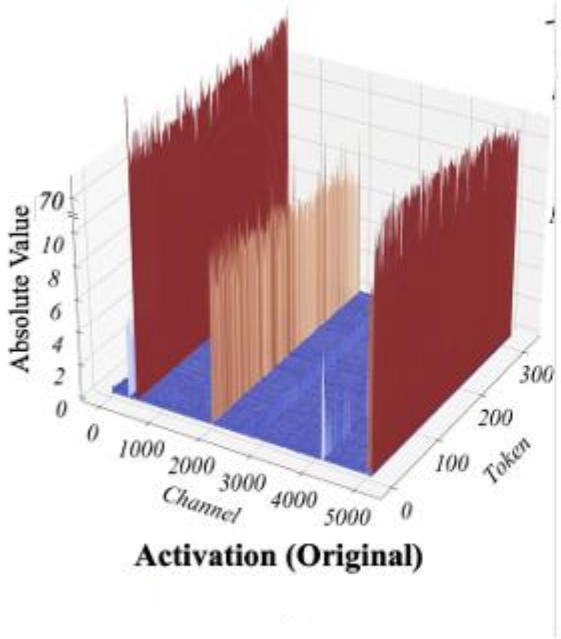
Per-channel QAT achieves best acc (tokens often demonstrate similar range patterns in the same channel) but hard to support in hardware

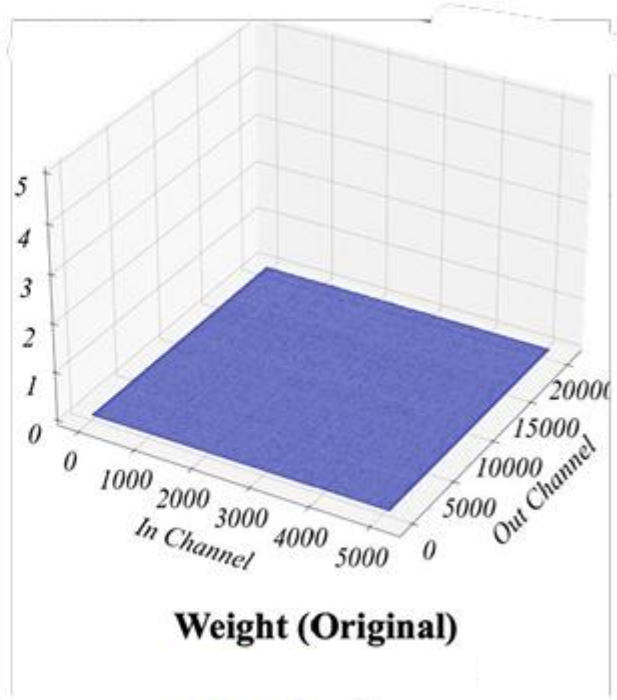
Quantization Granularity

- Per-Tensor Quantization
- Per-Token Quantization
- Per-Channel (e.g., weight column) Quantization
- Group-wise Quantization
  - Different scaling factors for different channels groups



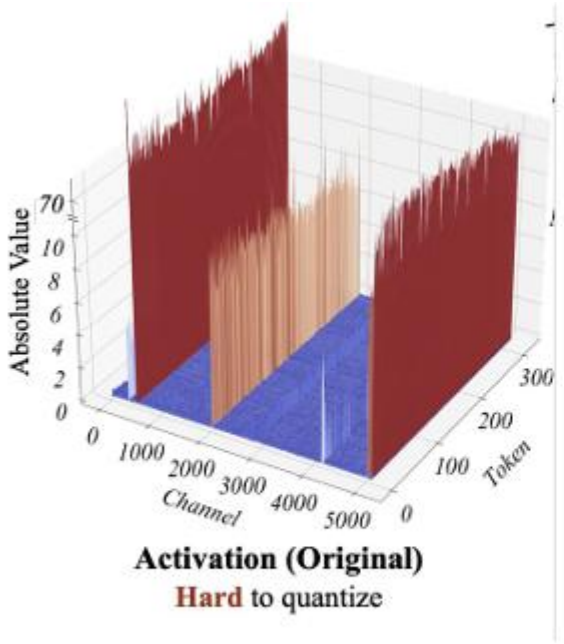
More Observations

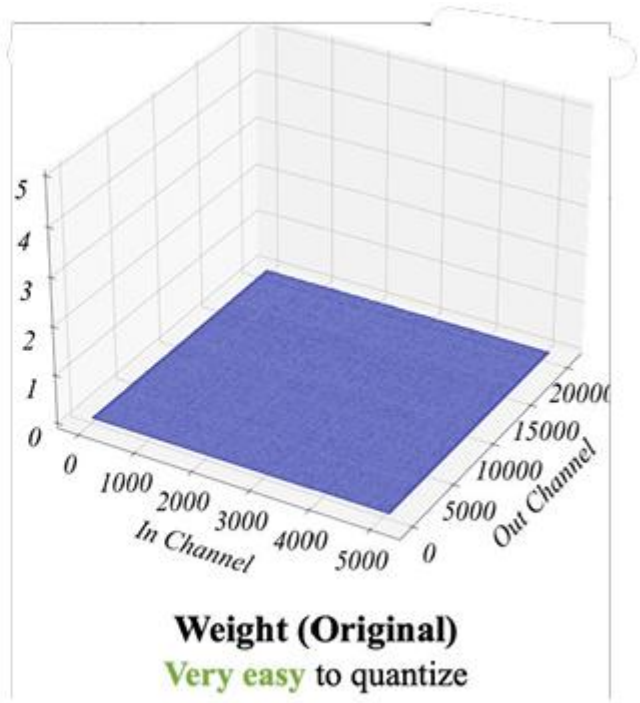




- Output = Activation X Weight, so we want to quantize both weights and activations into INT8 (i.e., W8A8) to utilize the integer kernels (e.g., INT8 GEMM)
- Activation has many outliers (10-70x larger) while the weight value is smooth (flat)
  - Outliers leave few effective bits for most values

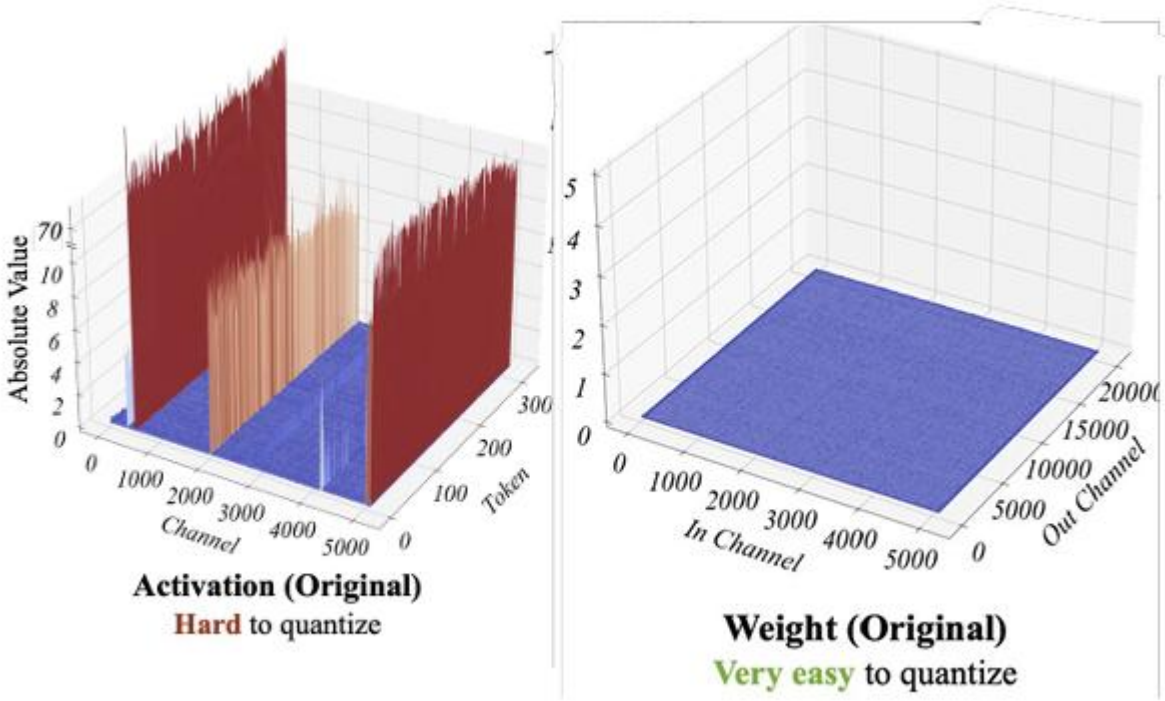
Activations are Intrinsically Challenging to Compress



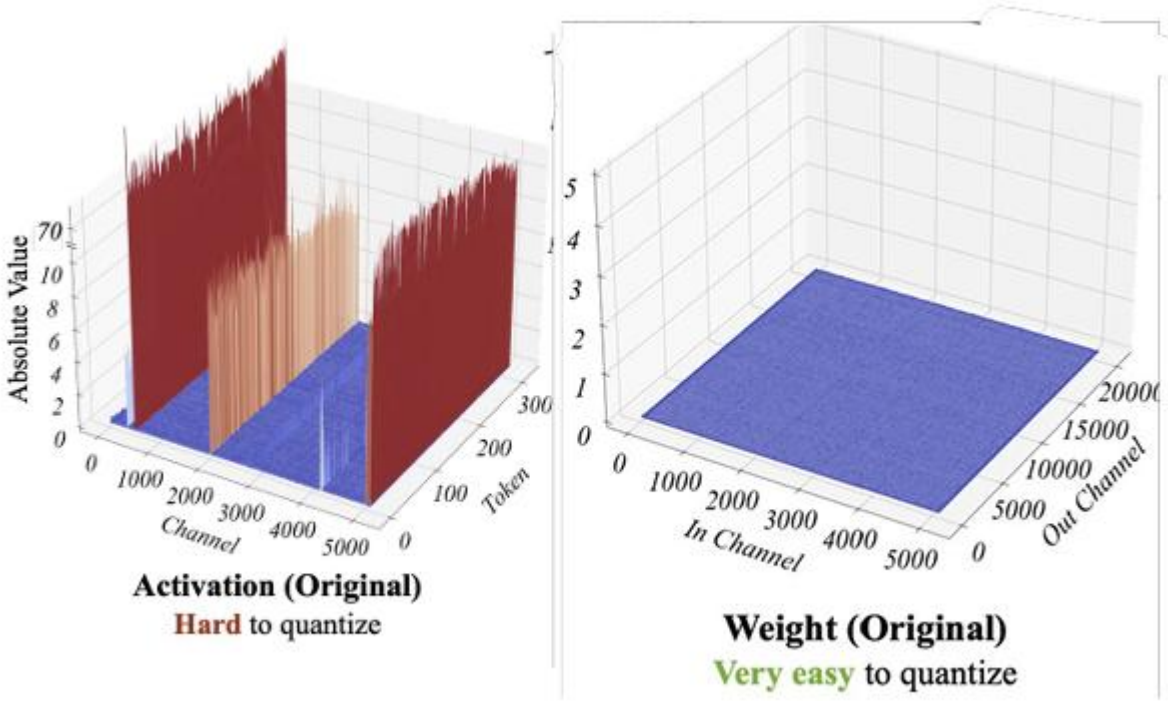


- Output = Activation X Weight, so we want to quantize both weights and activations into INT8 (i.e., W8A8) to utilize the integer kernels (e.g., INT8 GEMM)
- Activation has many outliers (~70x larger) while the weight value is smooth (flat)
  - Outliers leave few effective bits for most values

SmoothQuant: Difficulty Transformation



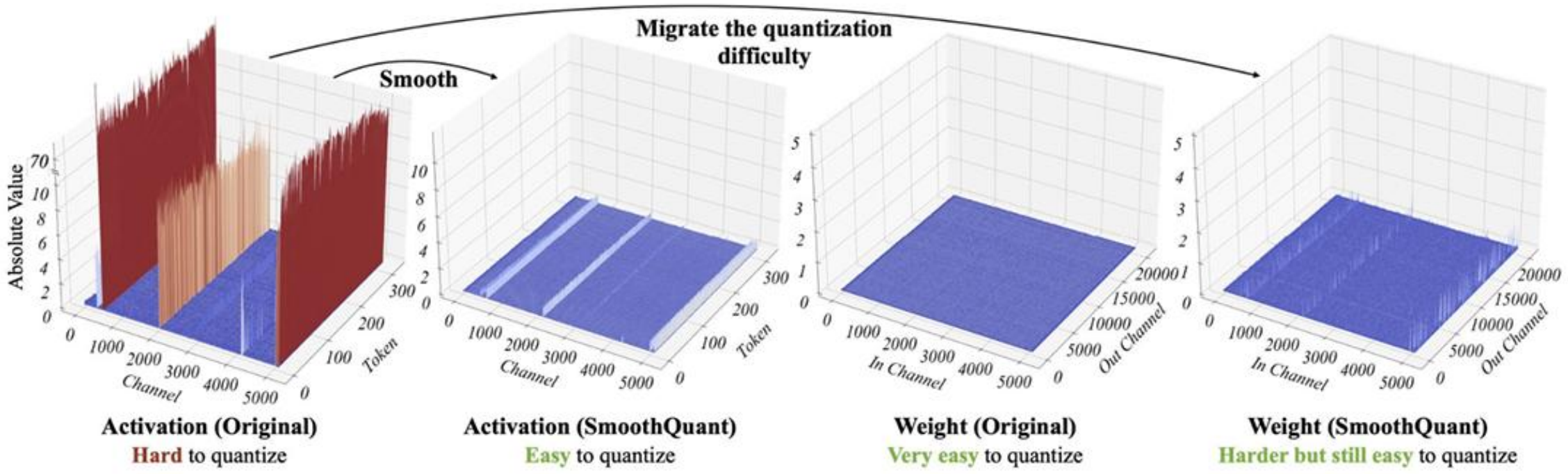
SmoothQuant: Difficulty Transformation



$$\mathbf{Y} = \mathbf{XW} = (\mathbf{0.01X})(\mathbf{100W})$$

- **Key idea: Migrating the quantization difficulty from activation to weight**

SmoothQuant: Difficulty Transformation



- **Key idea: Migrating the quantization difficulty from activation to weight**

SmoothQuant: Per-Channel Smoothing Factor

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

- Push all quantization difficulty from activations to weights (all channels have same maximum magnitude):

$$\mathbf{s}_j = \max(|\mathbf{X}_j|), j = 1, 2, ..., C_i,$$

SmoothQuant: Per-Channel Smoothing Factor

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

- Push all quantization difficulty from activations to weights (all channels have same maximum magnitude):

$$\mathbf{s}_j = \max(|\mathbf{X}_j|), j = 1, 2, ..., C_i,$$

- Push all quantization difficulty from weights to activations:

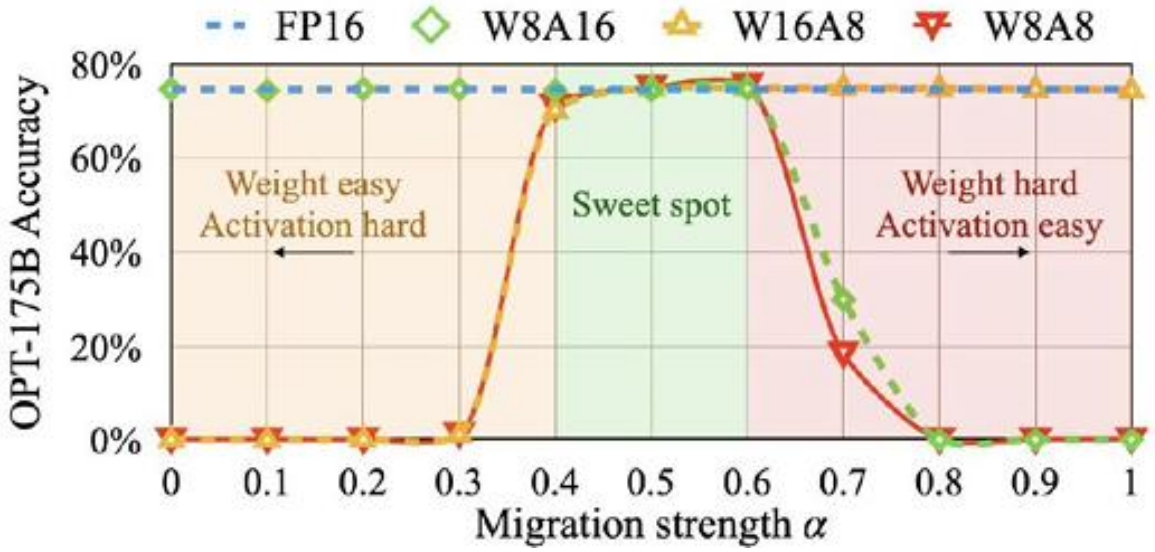
$$\mathbf{s}_j = 1/\max(|\mathbf{W}_j|)$$

- Balance difficulty according to α:

$$\mathbf{s}_j = \max(|\mathbf{X}_j|)^{\alpha} / \max(|\mathbf{W}_j|)^{1-\alpha}$$

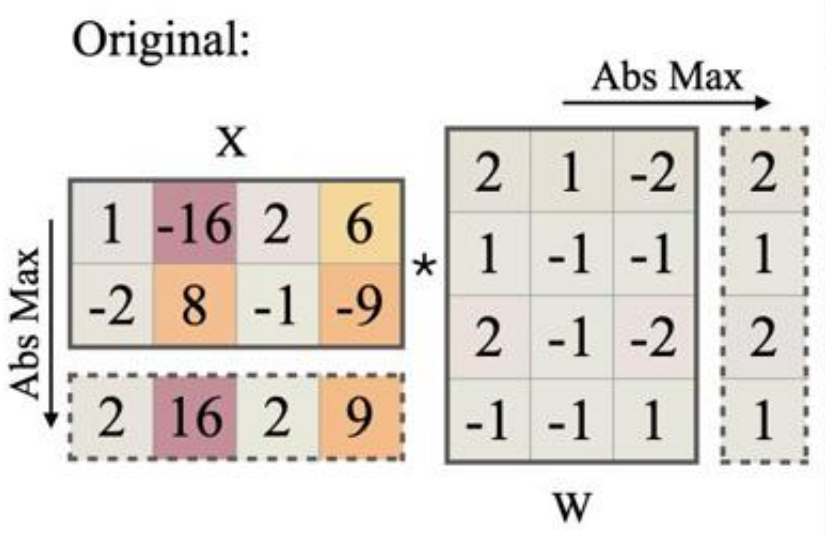
Choosing α

- Case-by-case decision
- If α is too large, weights will be hard to quantize. If too small, activations will be hard to quantize.
- Goal: Make activations and weights both easy to quantize.



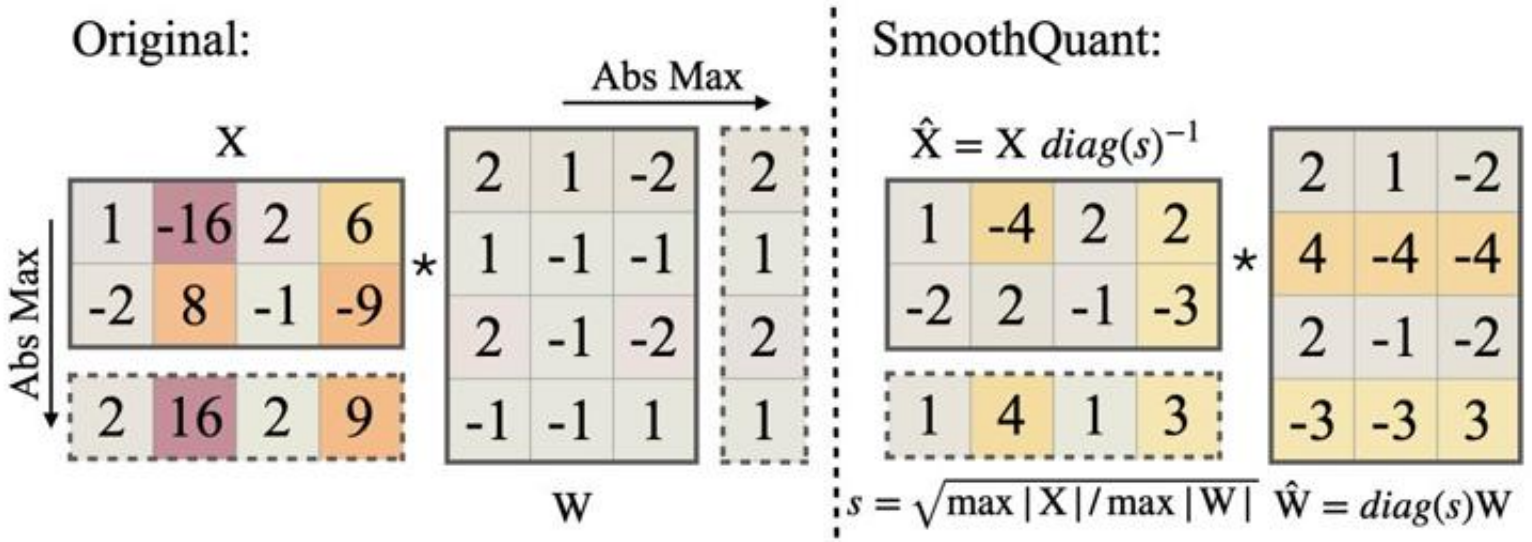
Example of SmoothQuant

- 1) Applying Smoothing Factor
- 2) Quantize (constant step size)



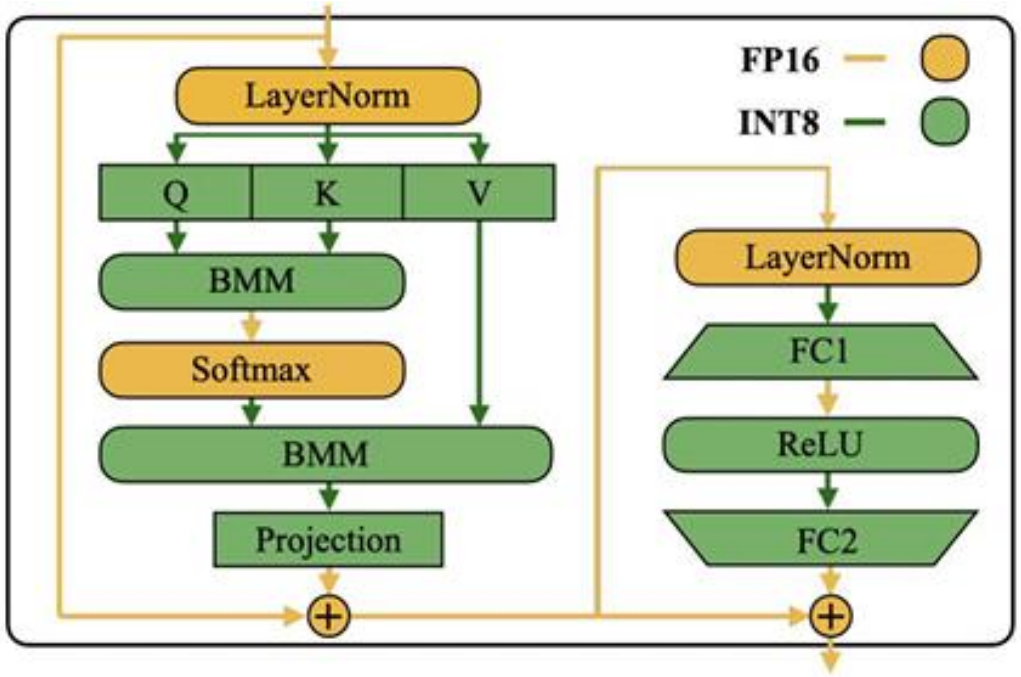
Example of SmoothQuant

- 1) Applying Smoothing Factor
- 2) Quantize (constant step size)

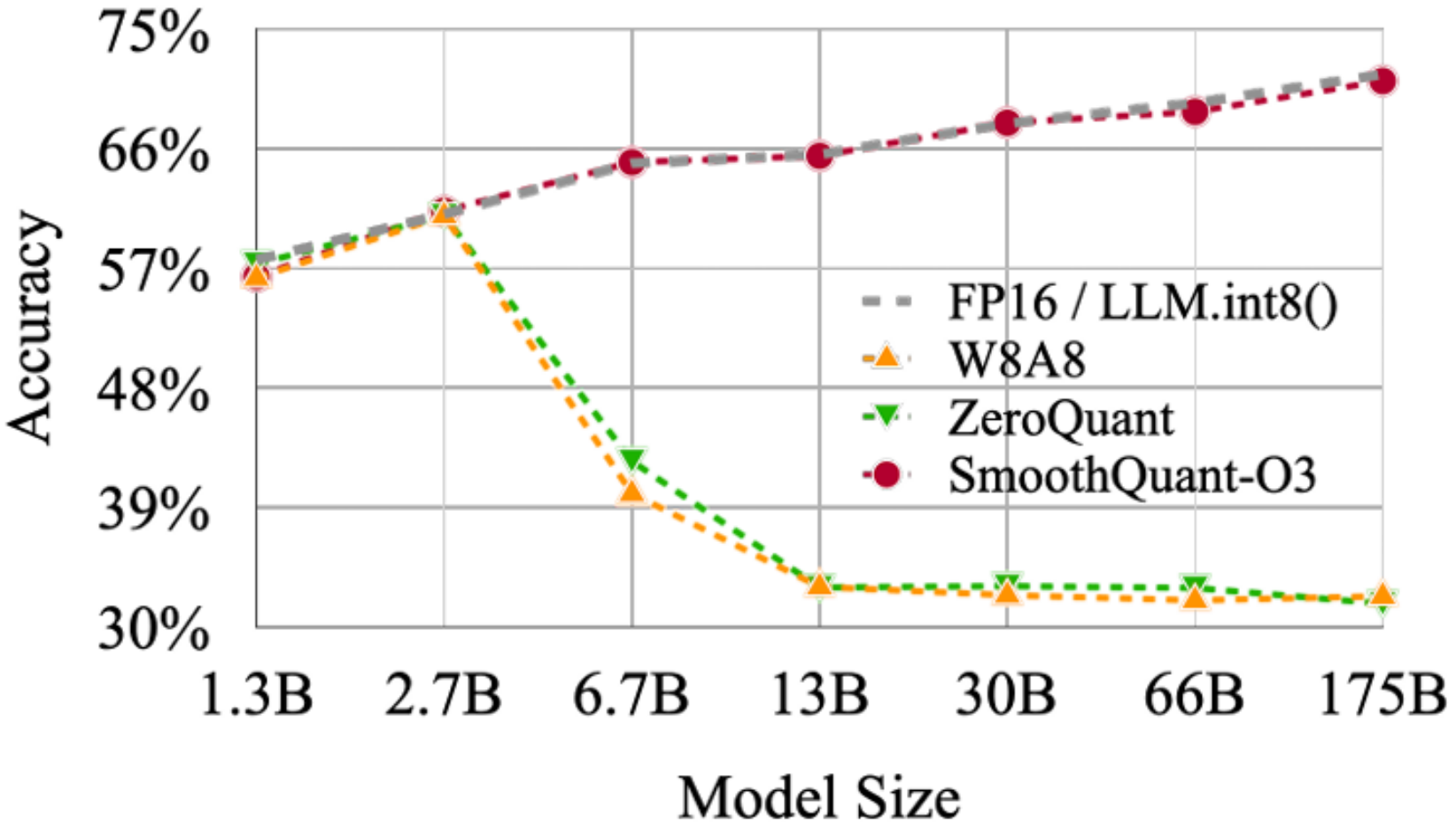


SmoothQuant Hardware Efficiency

- Applying SmoothQuant to transfer blocks
  - Linear layers take up most of the parameter and computation
  - Smoothing factor can be fused into previous layers' parameters offline
  - All linear layers are quantized with W8A8, as well as BMM operators in attention computation

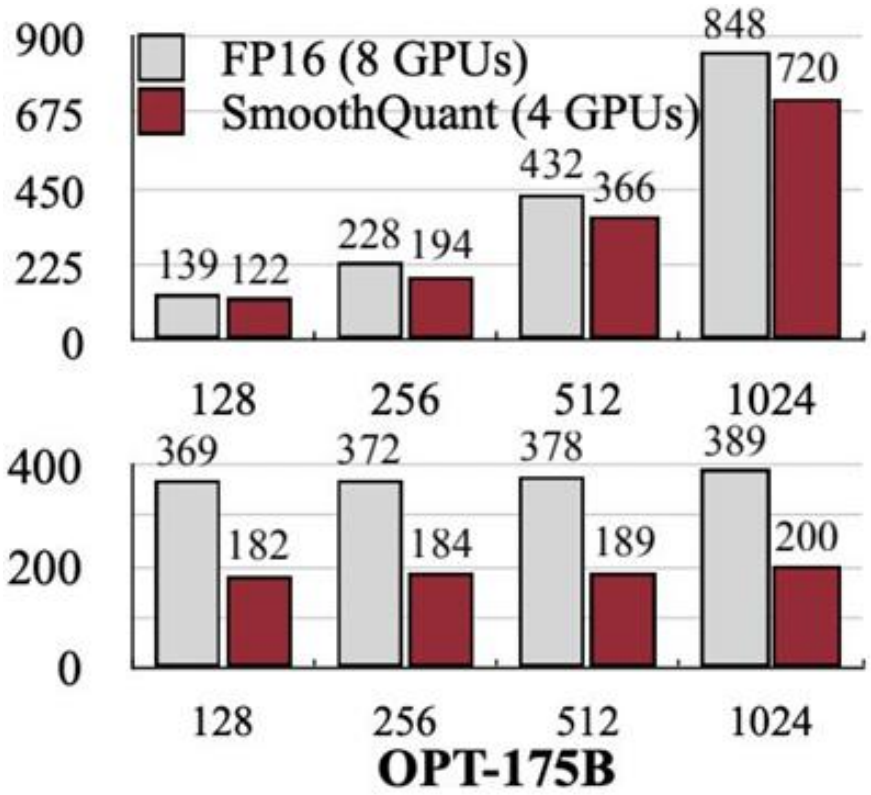


Evaluation: OPT 1.3B to 175B



Hardware Efficiency

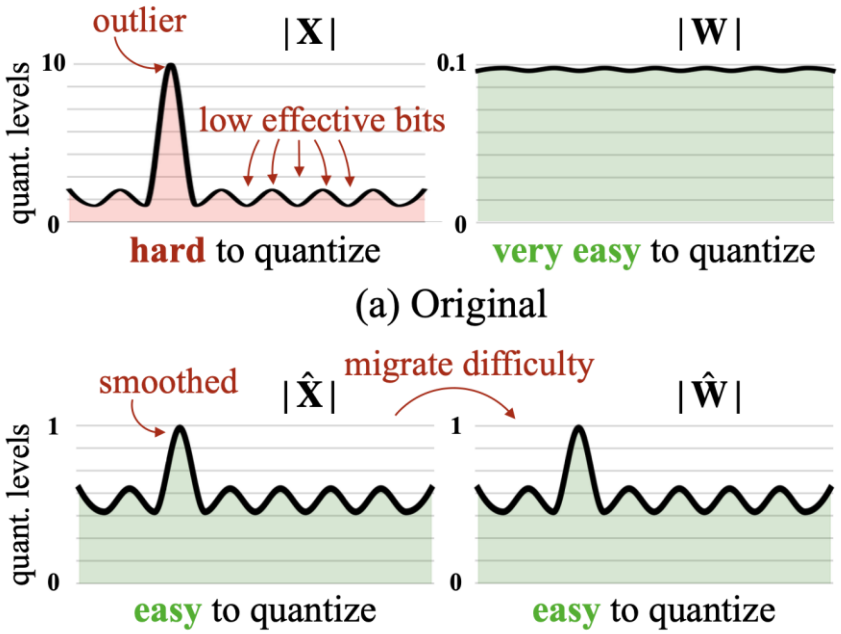
- Similar or faster latency with half #GPUs



Integrated into TensorRT and vLLM

Limitations

Limitations



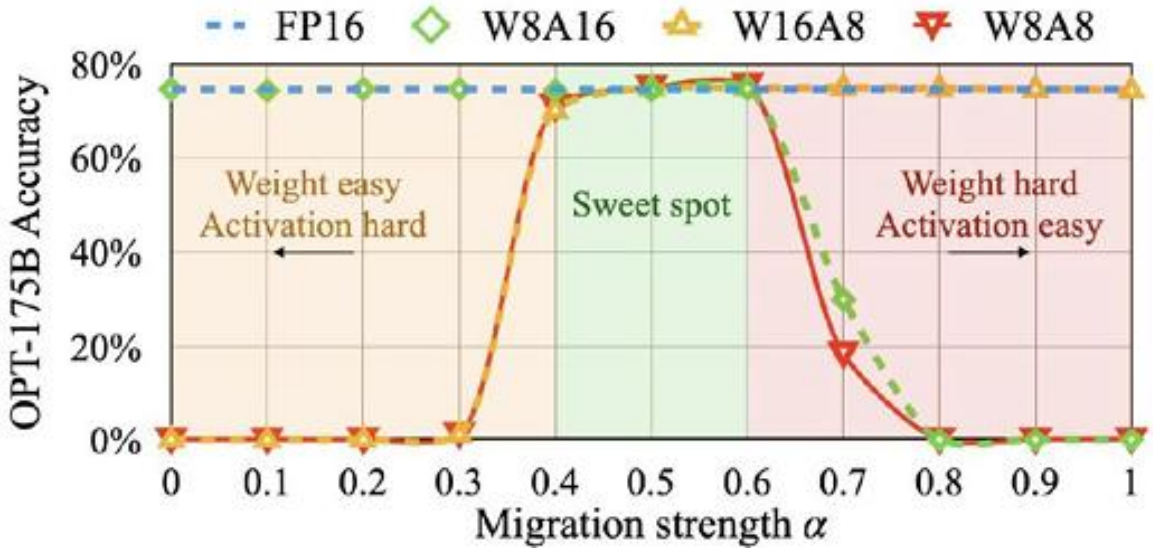
- Smoothing tradeoff
- Calibration dependency (e.g., smoothing factor)

Questions?

SmoothQuant: Alpha (α)

- Hyperparameter which controls the extent to which quantization difficulty is shifted from activations to weights

- α is between 0.4 to 0.6, though larger models or models with more significant activation outliers may require higher values.



Evaluation: SmoothQuant O1 to O3

- Gradually aggressive and efficient (lower latency) quantization levels

Method	Weight	Activation
SmoothQuant-O1	per-tensor	per-token dynamic
SmoothQuant-O2	per-tensor	per-tensor dynamic
SmoothQuant-O3	per-tensor	per-tensor static

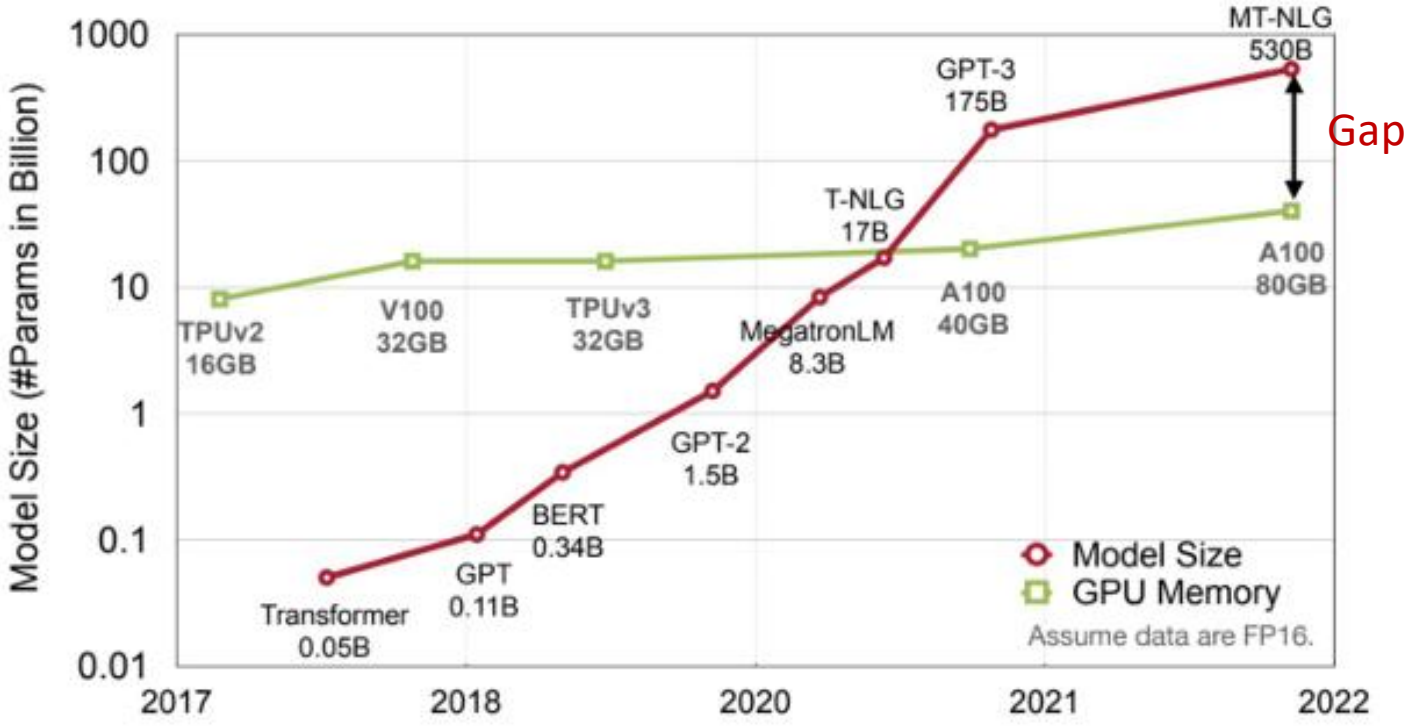
Serving Large Language Models Is Expensive

- Large language models (LLMs) are taking over every field.

- As the models get larger, serving such models for inference becomes **expensive** and **challenging**!

Gap Between the Supply and Demand of LLMs

- Size of LLMs is increasing faster than GPU memory: **exponential growth in model size** vs. linear increase in GPU memory



Gap Between the Supply and Demand of LLMs

- Size of LLMs is increasing faster than GPU memory: **exponential growth in model size** vs. linear increase in GPU memory
- **Memory challenges:** 175B parameters in GPT-3 requires 350GB of memory to store just weights