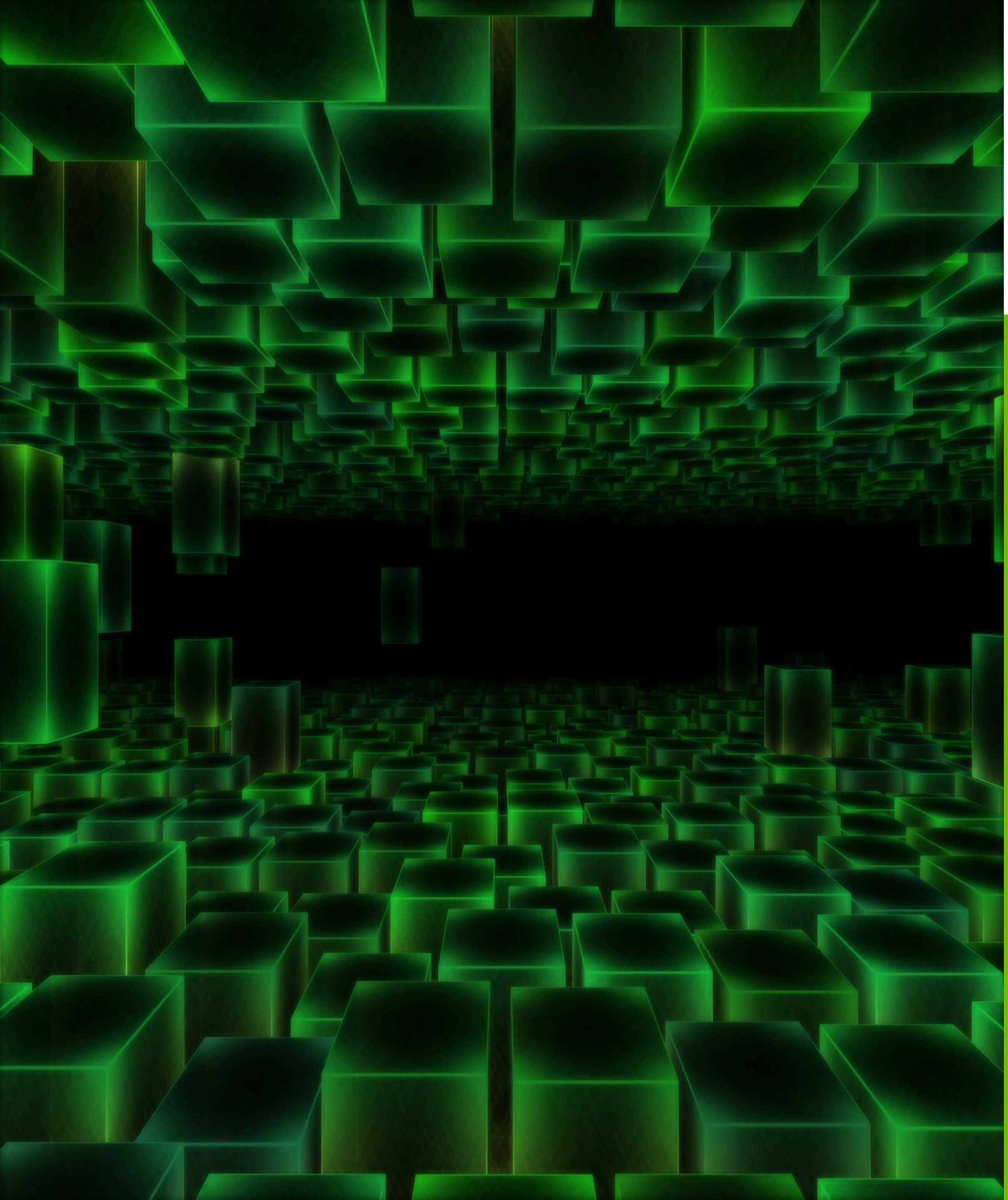
ACCELERATING LARGE LANGUAGE MODELS VIA LOW-BIT QUANTIZATION

YOUNG JIN KIM – PRINCIPAL RESEARCHER, MICROSOFT

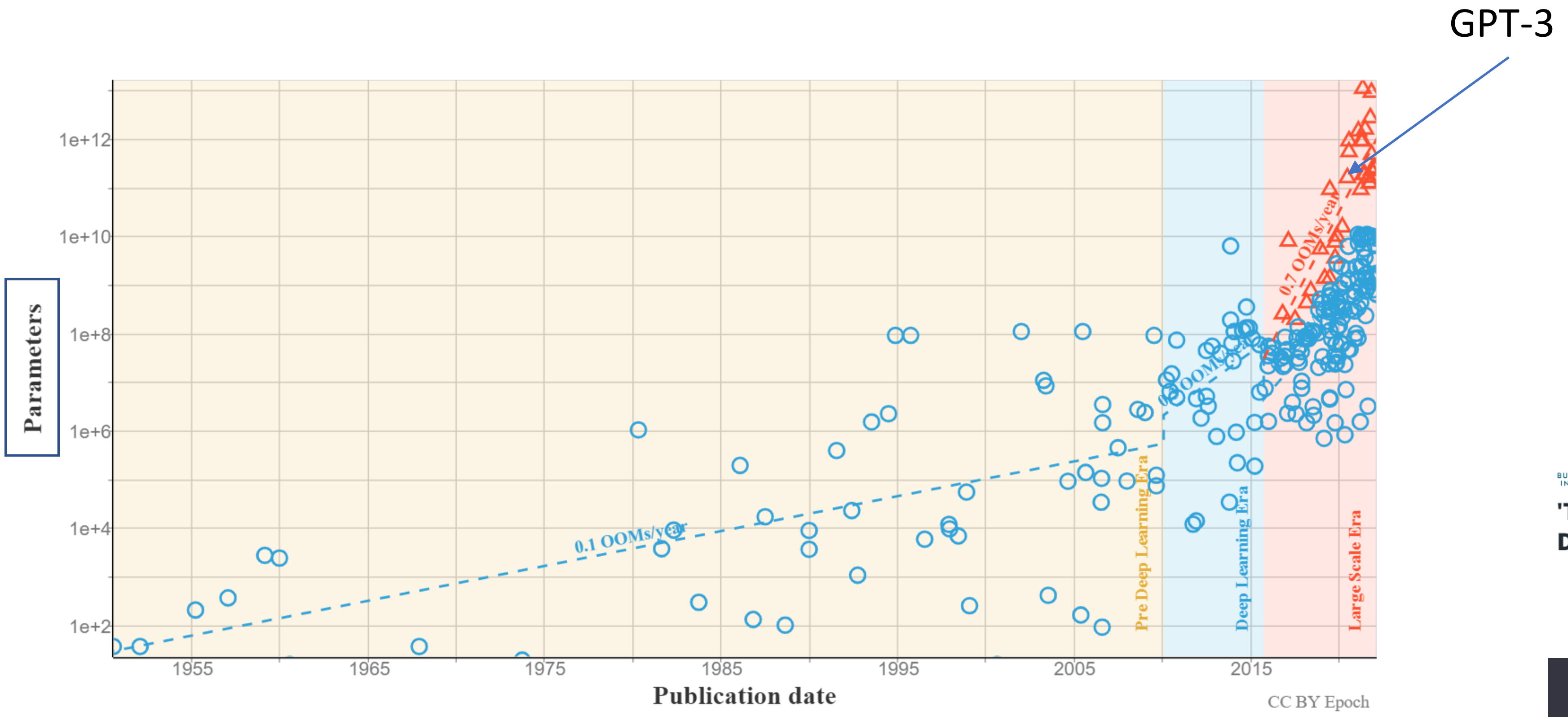
RAWN HENRY – SENIOR AI DEVELOPER TECHNOLOGY ENGINEER, NVIDIA



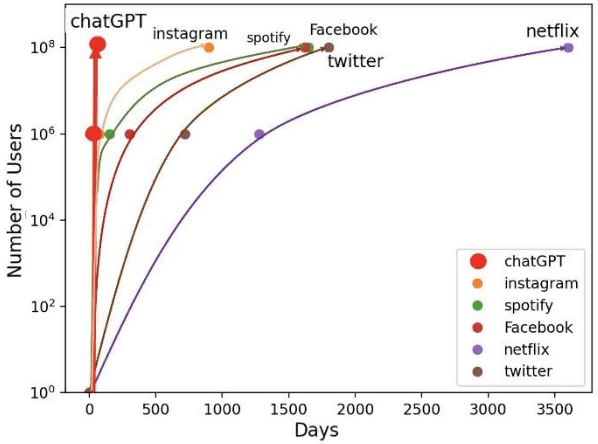
Agenda

- **Background on Quantization**
- Introduction to the model architecture (Mixture-of-Experts)
- Quantization properties on MoE LLM
- Custom kernels for Weight-Only Quantization
- Performance benchmarks
- Accuracy benchmarks

Era of large language models



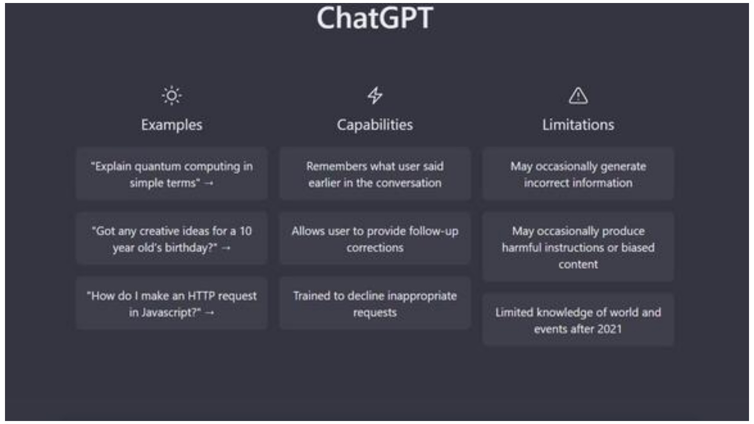
Sevilla, Jaime, et al. "Compute trends across three eras of machine learning." *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022.



BUSINESS INSIDER

'Transformative, fascinating': Here's what top executives at Davos are saying about ChatGPT

OpenAI | Microsoft



Challenges of inferencing large language models

Increased model size

GPT3 – 175B
OPT – 175B
BLOOM – 176B …

Decoder-heavy model architectures

-> Autoregressive decoding

- Model loading
  - Model weights takes up to 350 GB in FP16
  - Requires multiple GPUs (5+ A100 with 80GB memory)
    - 8 A100 with 40GB could not fit one model
  - Communication overheads between different GPUs and different nodes

- Memory bandwidth bound
  - Decoder processes 1 token at a time
  - One huge weight matrix is used for a very small activation (1 token) vector or tensor.
  - Loading large weight matrices creates a memory bandwidth bottleneck

Methods to accelerate large language models

Quantization

1. General approach to approximate the numerical values with a lower precision.
2. Less dependency on model architectures and hardware.

Pruning

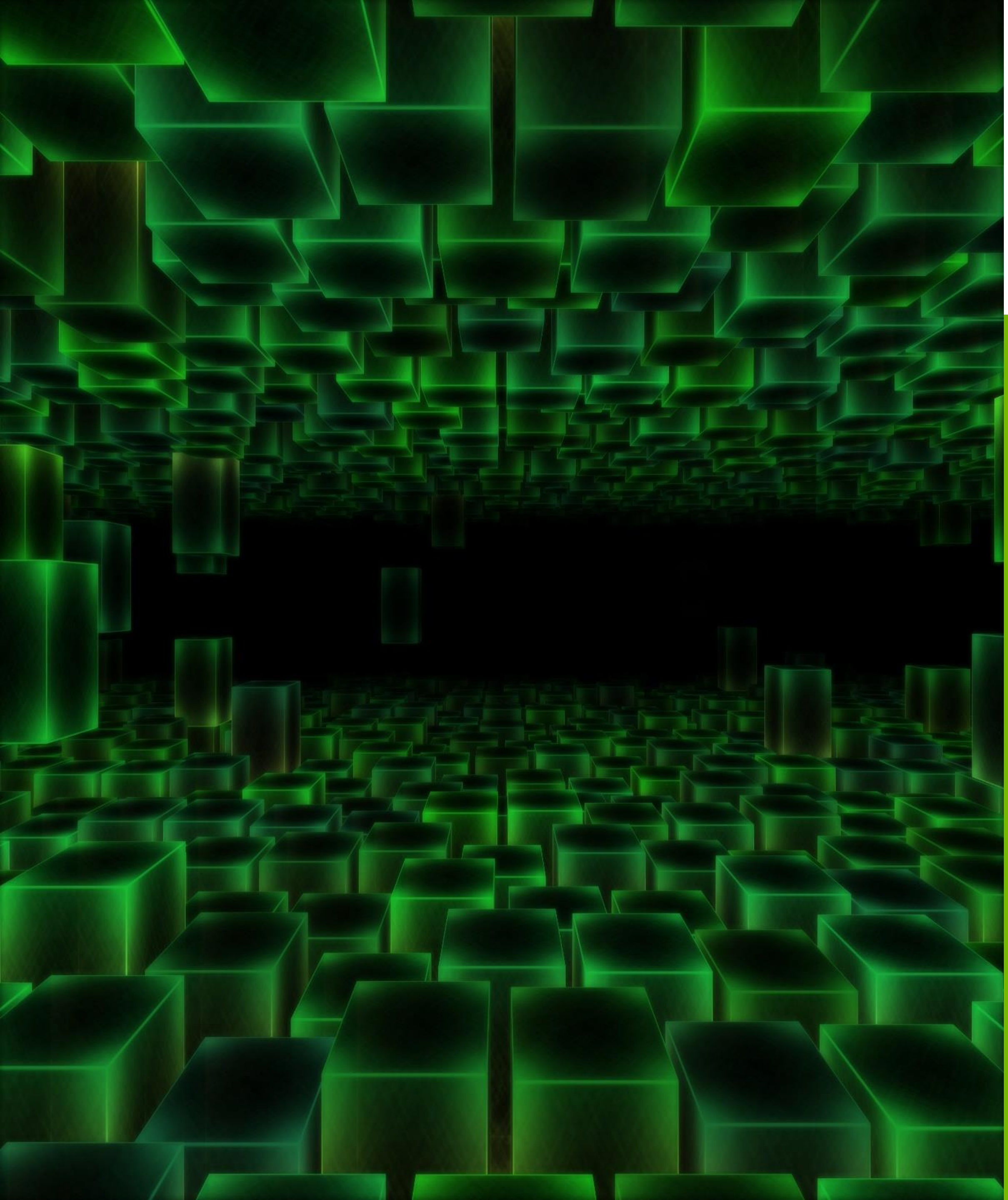
1. Making less important weights/activations zero.
2. Not easy to get actual latency/throughput improvements with unstructured pruning on modern hardware.

Knowledge distillation

1. No recipes found effective for large language models, yet.
2. Complex dynamics with teacher – student training – student architecture, size and etc.

Conditional computation (such as Mixture-of-Experts)

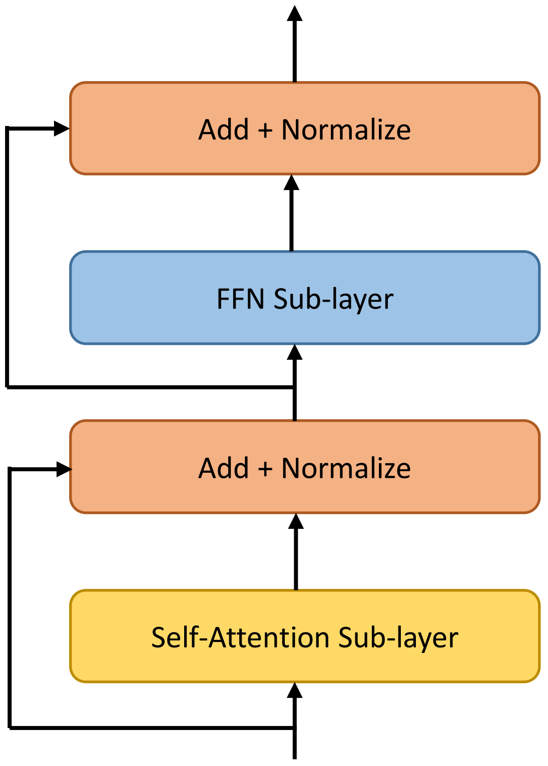
1. Using subset of parameters at a given time.
2. Can be complementary with any of the other approaches.



Agenda

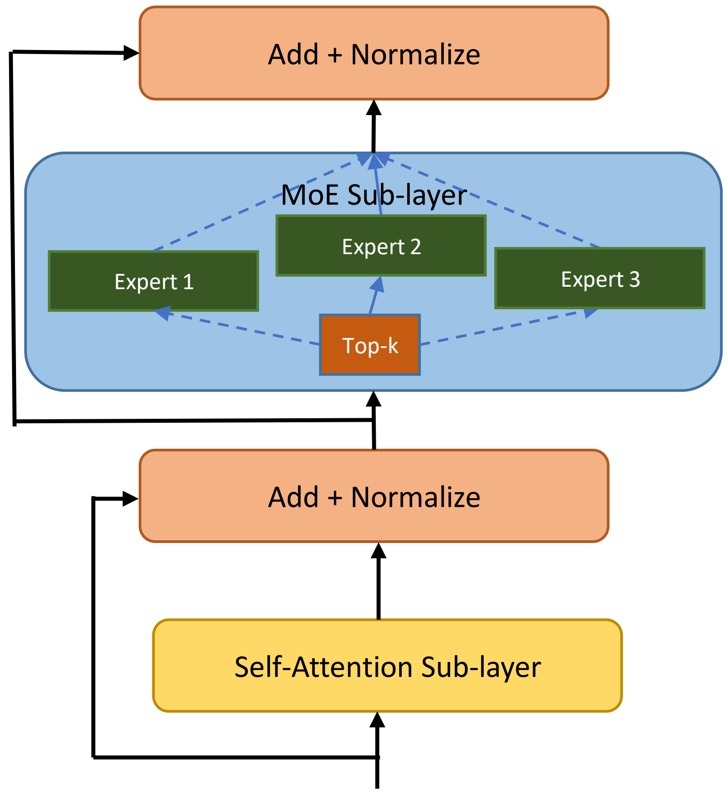
- Background on Quantization
- **Introduction to the model architecture (Mixture-of-Experts)**
- Quantization properties on MoE LLM
- Custom kernels for Weight-Only Quantization
- Performance benchmarks
- Accuracy benchmarks

Sparse Mixture-of-Experts Transformer



Dense transformer

replace the FFN sublayer with MoE sub-layer (*Mixture of Experts*, i.e., a set of FFN sublayers residing at different machines)



MoE Transformer

Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer **(Shazeer et al. 2017)**
GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding **(Lepikhin et al. 2020)**
Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity **(Fedus et al. 2021)**

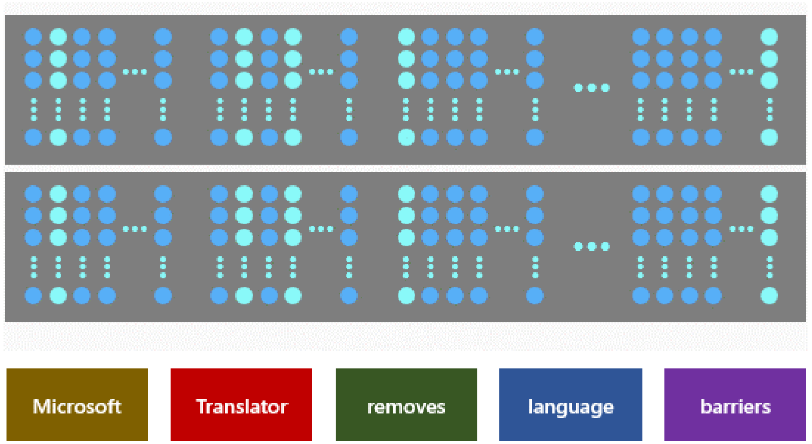
Sparse Models for Efficient Scaling

Dense Models:

- All parameters are used in forward and backward paths
- Increasing model capacity needs more computation
- Optimized for dense computation
- Larger models requires model parallelism with heavy communication across devices
- **Larger model size → Higher compute requirements (FLOPs)**

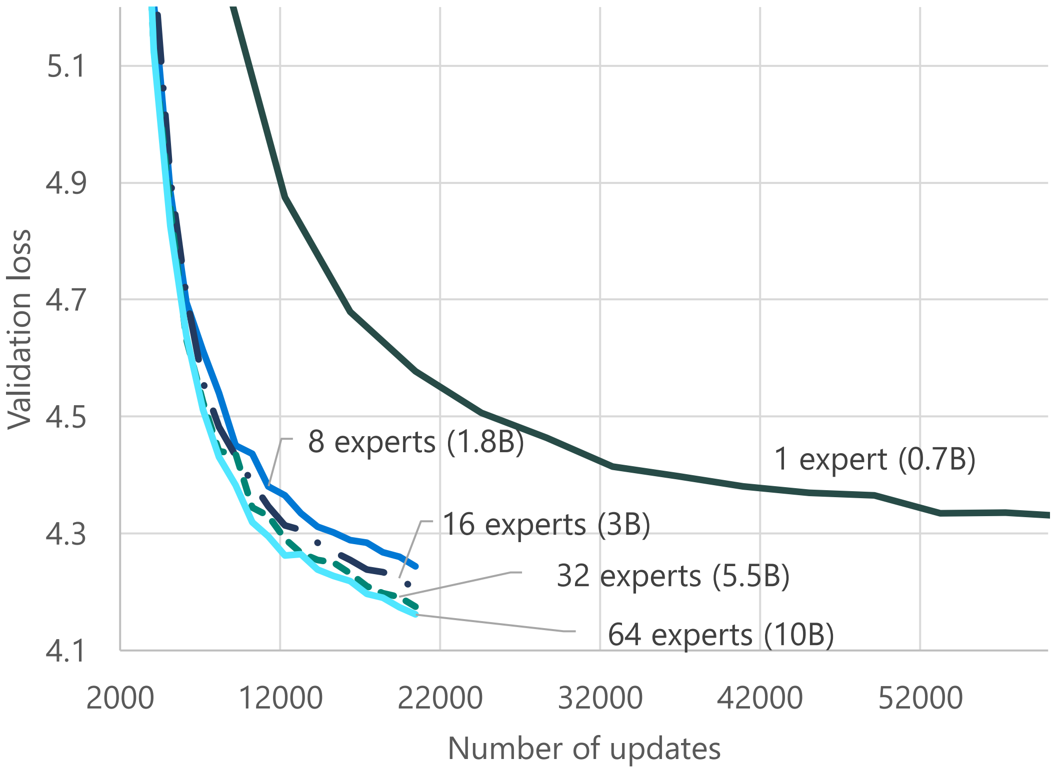
Sparse MoE models

- Sparse utilization of subset of parameters based on input
- Same computation is needed regardless of the model size (top-1 gating)
- Structured sparsity
- Requires more All-to-All communication
- Larger models can be achieved by expert parallelism with much less communication requirements
- Natural fit for multitask and multilingual modeling
- **Larger model size → Similar/Same Compute requirements**

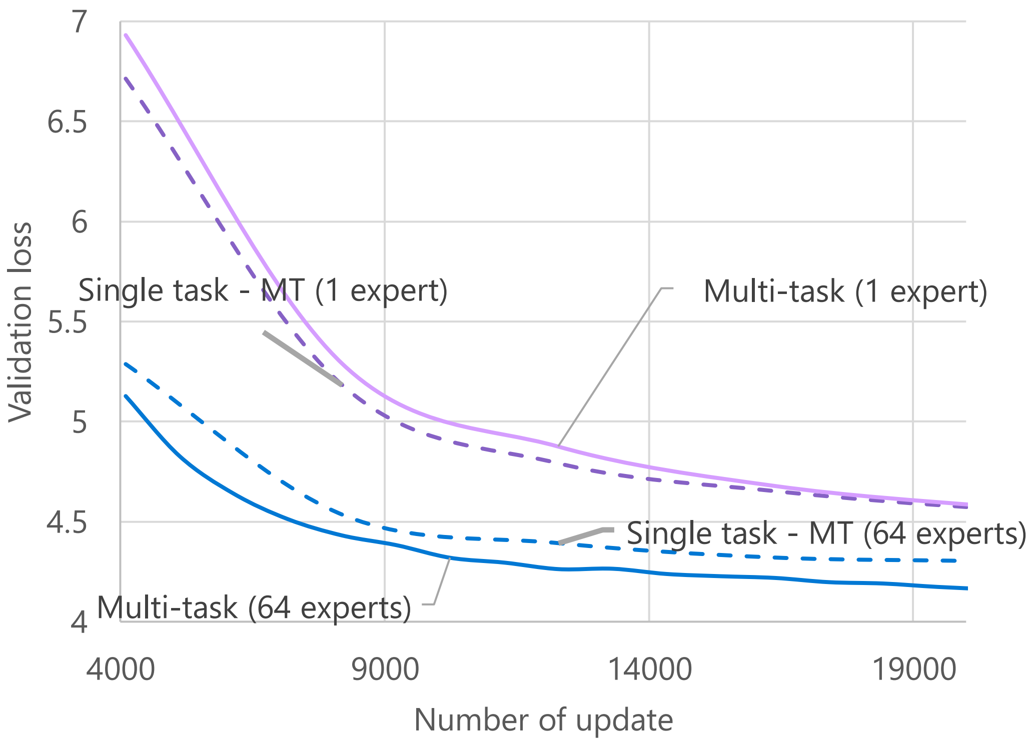


Z-Code MoE: Accuracy and Training Efficiency

More experts -> Better accuracy and faster convergence on **validation loss**



MoE is a better multitask learner!



MoE Inference Challenge

- Constant FLOPs $\neq$ Constant inference speed

With 32 experts, top-1 gating

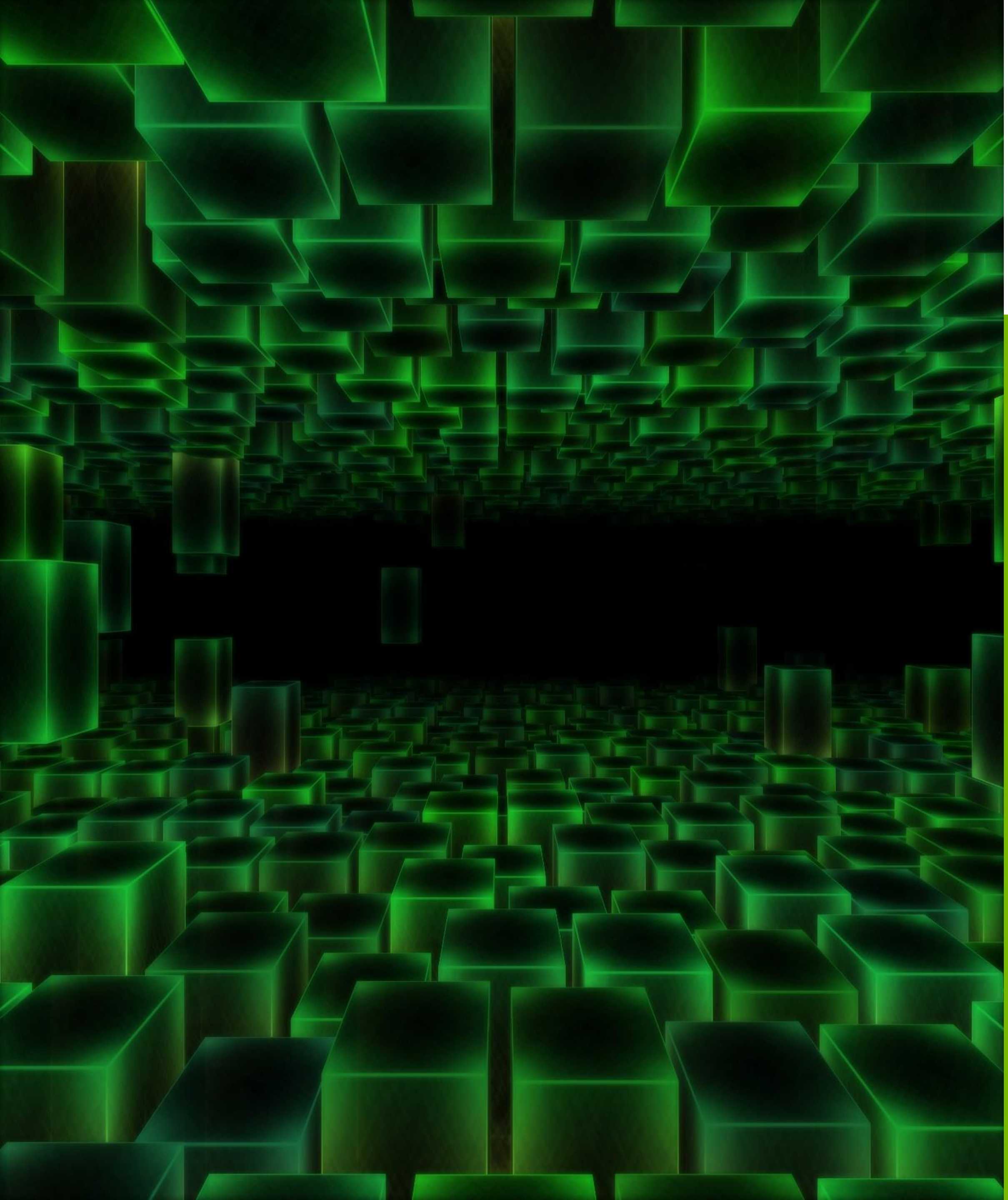
Model parameters: 5B

Hidden dim: 1024
Number of heads: 16
Number of encoders: 24
Number of decoders: 12
MoE: **32 experts per every other layers**

Model	Throughput (sentences/second)	Model size (*fp16*) in GB	% of MoE weights
Dense	14.02	1.18	-
MoE (32 experts)	5.37	9.91	**92.8 %**
Difference	0.38X	8.38X	-

Only less than 40% of throughput.

Memory consumption increases more than 8 times.
-> Prevent larger batch size.



Agenda

- Background on Quantization
- Introduction to the model architecture (Mixture-of-Experts)
- **Quantization properties on MoE LLM**
- Custom kernels for Weight-Only Quantization
- Performance benchmarks
- Accuracy benchmarks

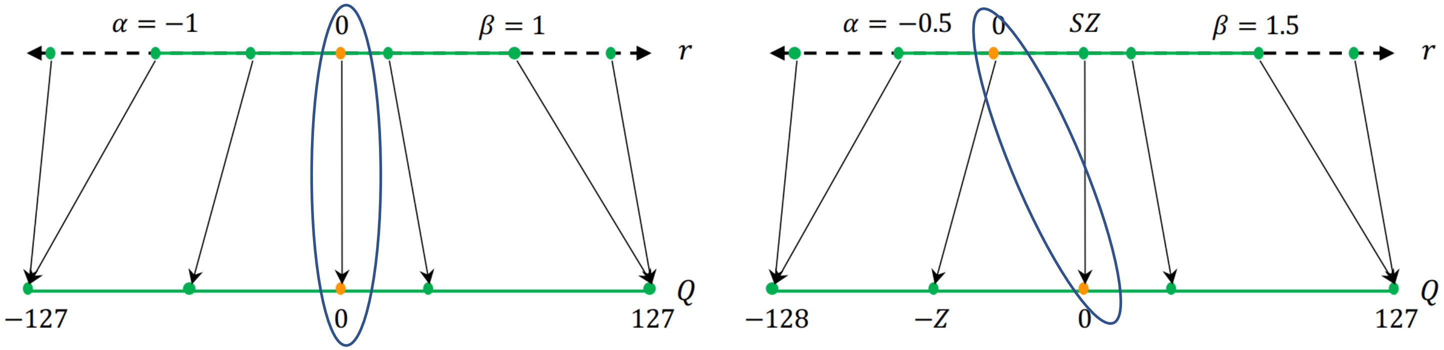
What to quantize?

	Weight-only	Weight & Activation
Memory consumption	Reduced	Reduced
Computation	De-quantization step needed to utilize existing floating point GEMMs	- Potential to use low-bit quantized GEMMs - Potential to get more speed-up
Hardware	**Hardware agnostic**	- Requires integer arithmetic instructions
Accuracy	**Smaller accuracy impact is expected by using floating point arithmetic → but not verified with LLM and low-bits**	Depending on models and tasks, there could be big accuracy loss

Best fit if the computation disadvantage can be overcome and good accuracy can be achieved

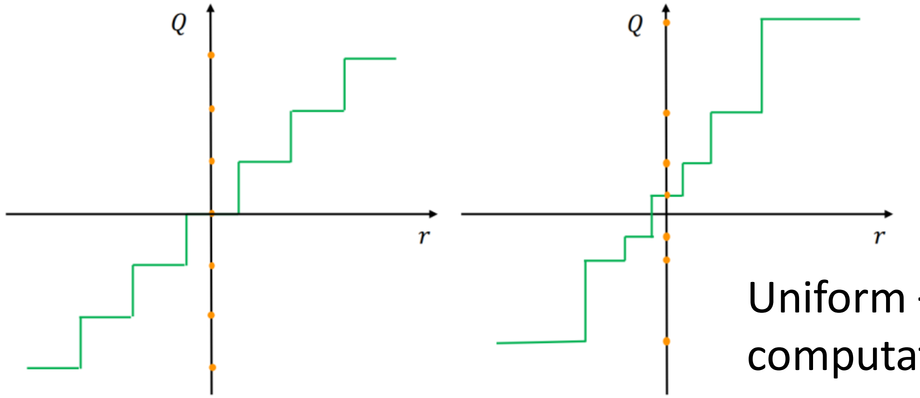
How to quantize model weights?

- Symmetric vs. non-symmetric



Symmetric – less computation,
potentially less accurate

- Uniform (linear) vs. non-uniform (log-linear)



Uniform – less computation

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W. and Keutzer, K., 2021. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*.

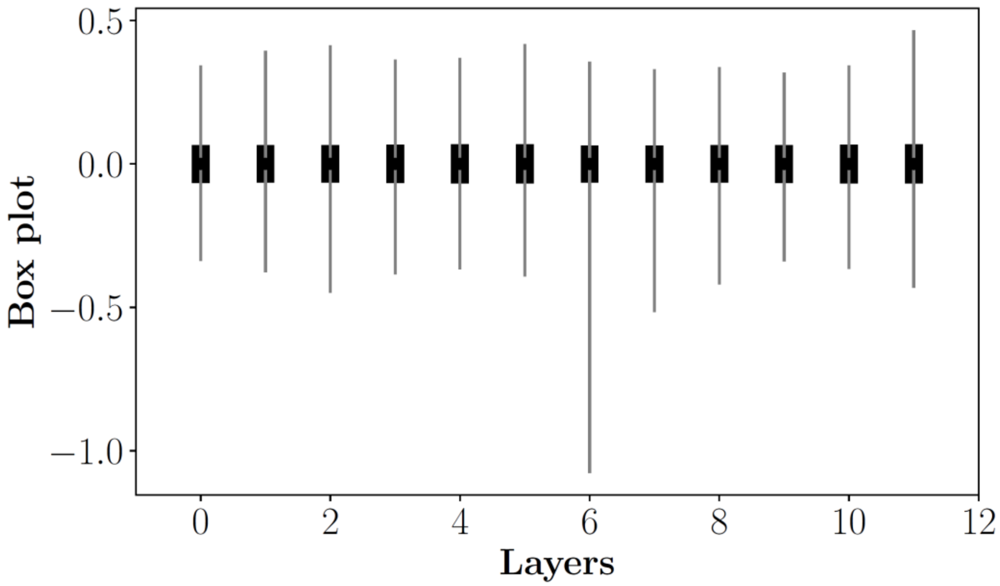
How to quantize model weights?

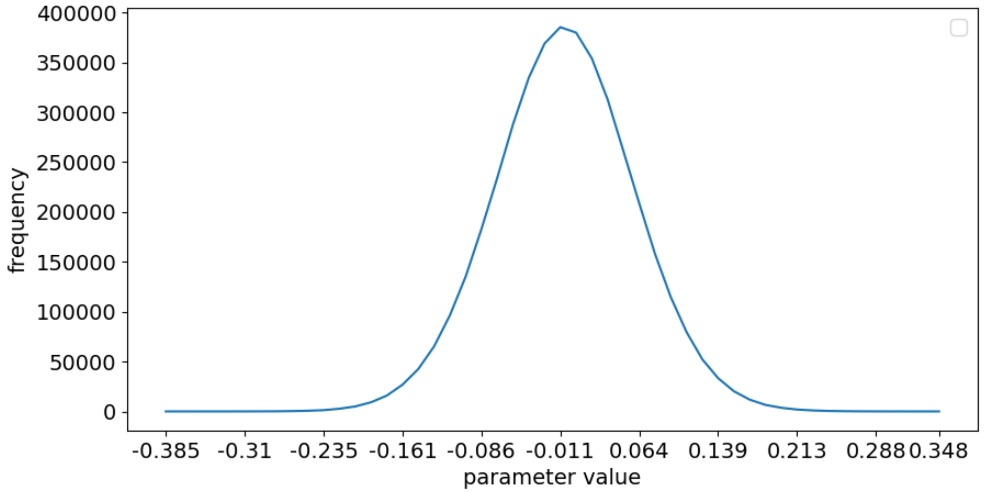
- Quantization granularity – matrix, channel and group
  - The same quantization scaler and bias can be used across different granularity
  - **Finer granularity** potentially gives **better accuracy**, but makes the algorithm **complex**

- Dynamic range vs. static range

  → Doesn’t matter for the weight-only quantization, only meaningful for dynamically changing activations

MoE Transformer weight distribution – MoE layers



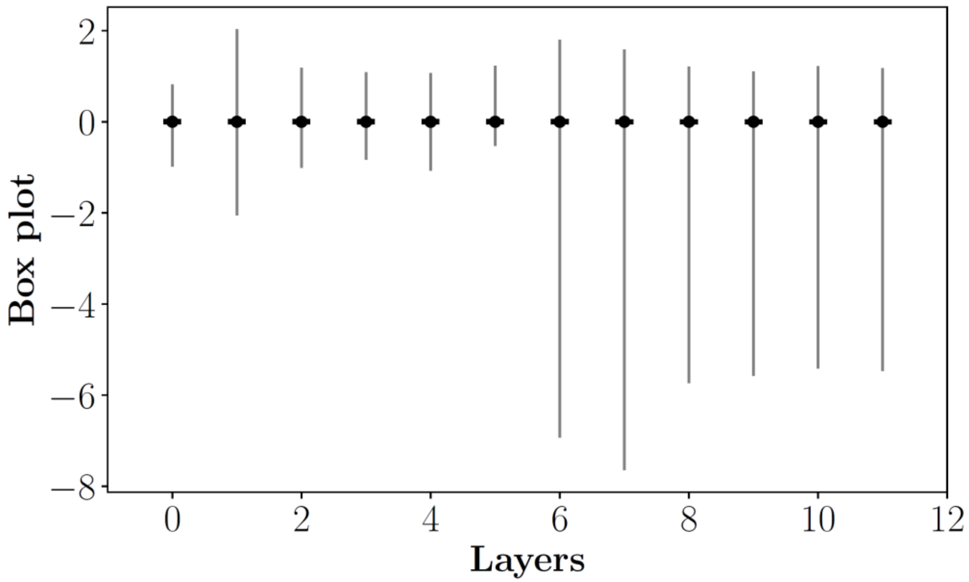


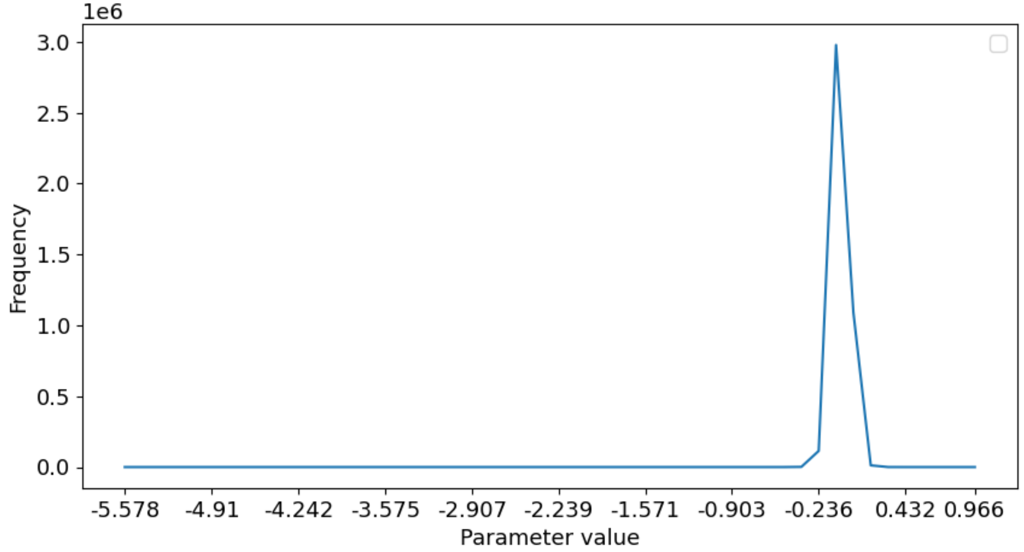
MoE layers' weights are symmetrically distributed

MoE layers' weight distribution is almost a normal distribution

Symmetric and **uniform** quantization!

MoE Transformer weight distribution – dense FFN layers

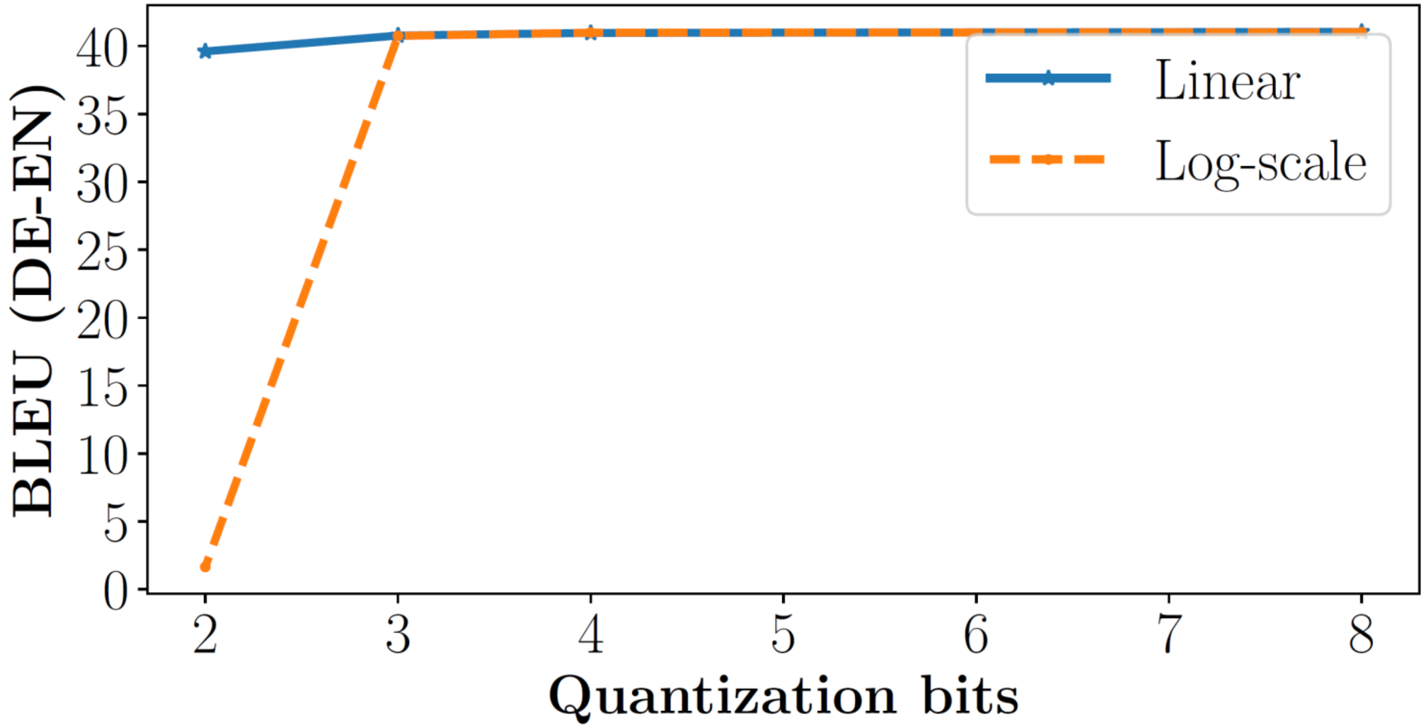




Dense FFN layers have outliers and distributed in wider ranges (-8.0 ~ 2.0)

Dense FFN layers have long tails with outliers

Uniform quantization

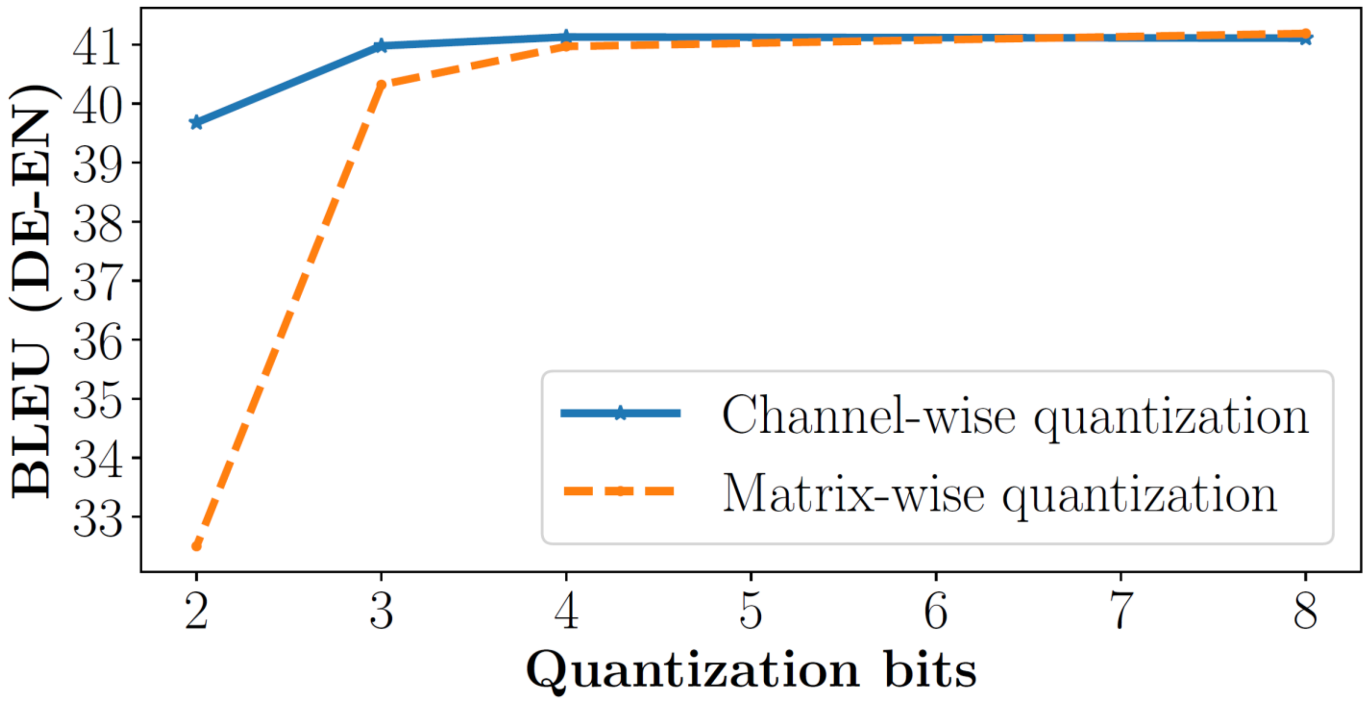


Both uniform and non-uniform quantization work reasonably well down to 3-bit. For 2-bit, uniform quantization outperforms

→ Considering the computational complexity, use **uniform** quantization

Translation BLEU score – higher is better

Quantization granularity



With matrix-wise quantization, the model loses the accuracy as the bits get lower.

→ User **Channel-wise** quantization

Translation BLEU score – higher is better

Uniform, symmetric and channel-wise quantization

Channel-wise absolute maximum quantization

Quantization

$$s_j = \frac{2 \times \max(|A_{:,j}|)}{2^b - 1}$$

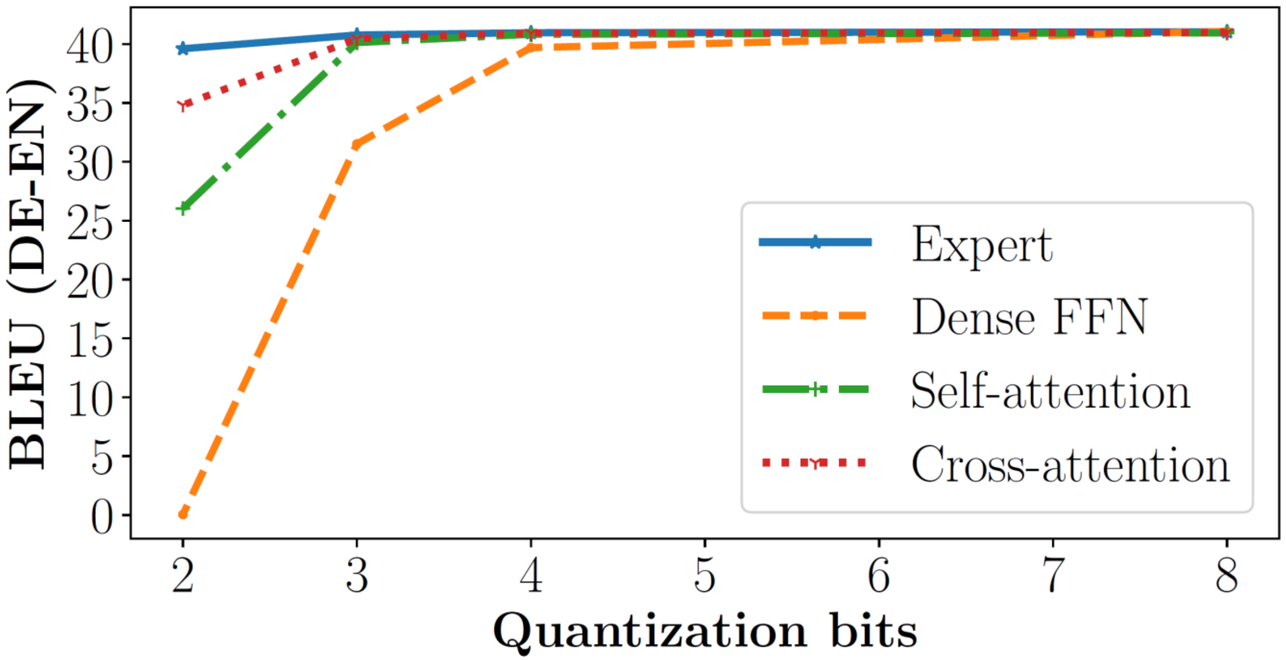
$$Q_{:,j} = \mathrm{int}(\frac{A_{:,j}}{s_j})$$

De-quantization

$$A'_{:,j} = Q_{:,j} \times s_j$$

A: matrix
j: channel index
b: bits to quantize
s: scaler
Q: quantized value
A’: de-quantized matrix

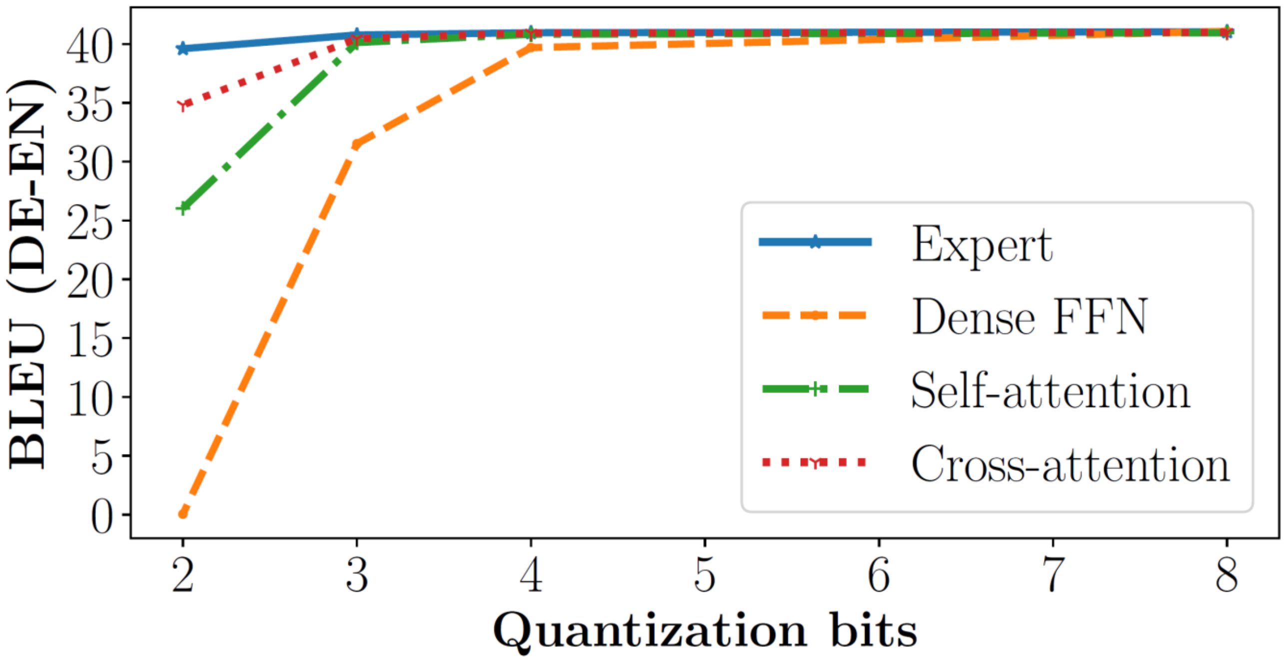
Accuracy impacts of quantization on different types of layers – inside 1 model



Translation BLEU score – higher is better

- Thanks to MoE layers' weight distribution, the proposed quantization method works very well to quantize MoE layers without losing any accuracy down to 3-bits.
- On the other hand, the other layers lose significant accuracy with low-bit quantization.

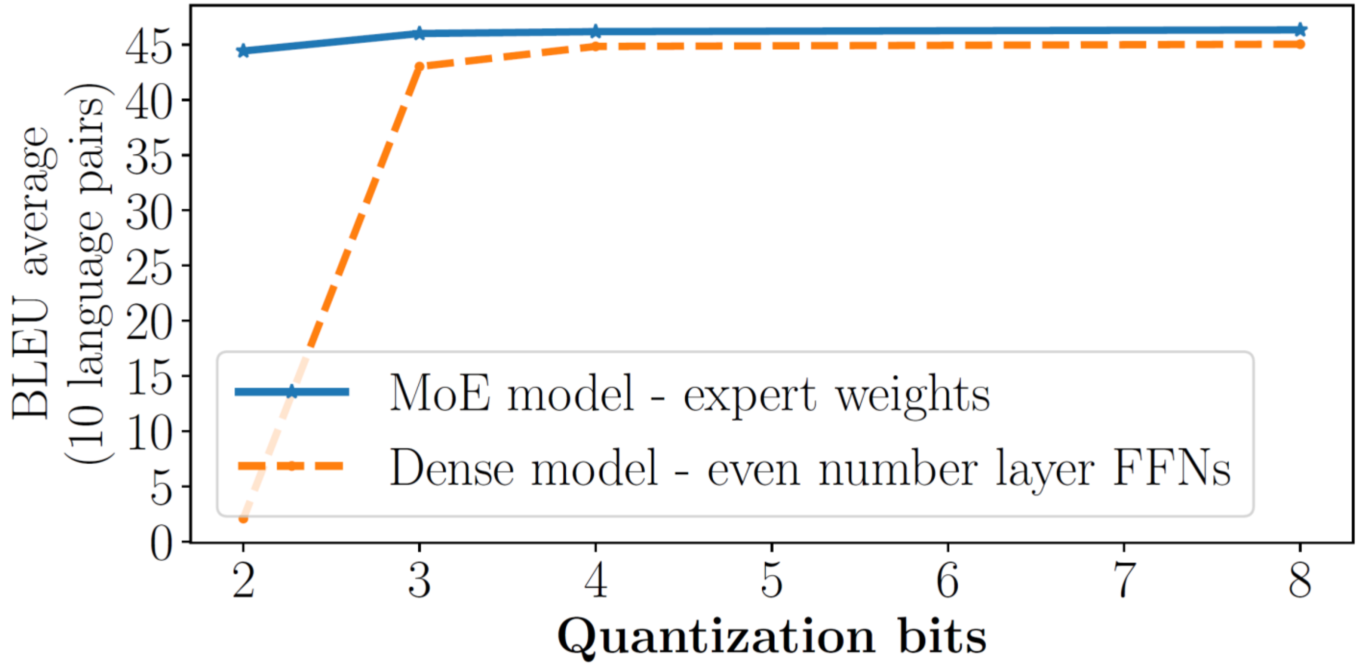
Accuracy impacts of quantization on different types of layers – inside 1 model



Translation BLEU score – higher is better

- Thanks to MoE layers' weight distribution, the proposed quantization method works very well to quantize MoE layers without losing any accuracy down to 3-bits.
- On the other hand, the other layers lose significant accuracy with low-bit quantization.

Accuracy impacts of quantization on MoE vs. Dense models – 2 different models



Translation BLEU score – higher is better

- Thanks to MoE layers' weight distribution, the proposed quantization method works very well to quantize an MoE model without losing any accuracy down to 3-bits.
- On the other hand, a dense model with dense FFN layers loses significant accuracy with low-bit quantization with uniform channel-wise quantization.

Quantization results

Translation BLEU score – higher is better

Model type	Precision	Average BLEU (difference %)	Size (X times)	Sparsity %
Dense (baseline)	fp16	45.06 (0)	1X	3.8e-5

	fp16	**46.35 (+2.87)**	8.38X	3.8e-5
MoE 5.3B (32 experts)	int8	**46.34 (+2.85)**	4.57X	1.24
	int4	46.18 (+2.49)	2.67X	20.68
	int3	46.01 (+2.11)	2.19X	42.15
	int2 QAT	45.90 (+1.88)	1.71X	79.10

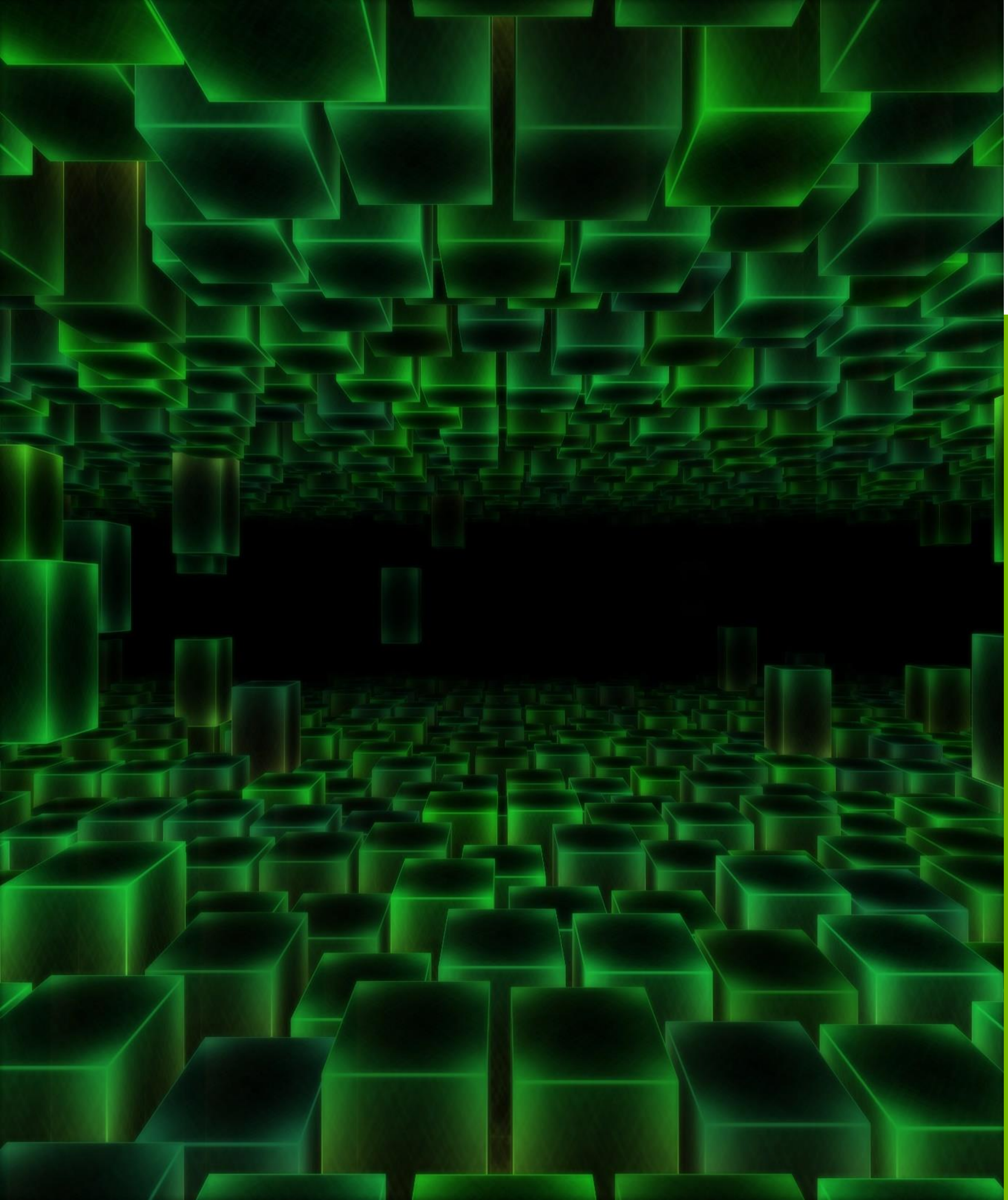
- Dense -> (MoE with **4-bit** quantization)

→ 2.49 % accuracy improvement while model size is only 2.67X (compared to 8.38X)

What to quantize?

	Weight-only	Weight & Activation
Memory consumption	Reduced	Reduced
Computation	De-quantization step needed to utilize existing floating point GEMMs	- Potential to use low-bit quantized GEMMs - Potential to get more speed-up
Hardware	**Hardware agnostic**	- Requires integer arithmetic instructions
Accuracy	**Smaller accuracy impact is expected by using floating point arithmetic**	Depending on models and tasks, there could be big accuracy loss

Best fit if the computation disadvantage can be overcome

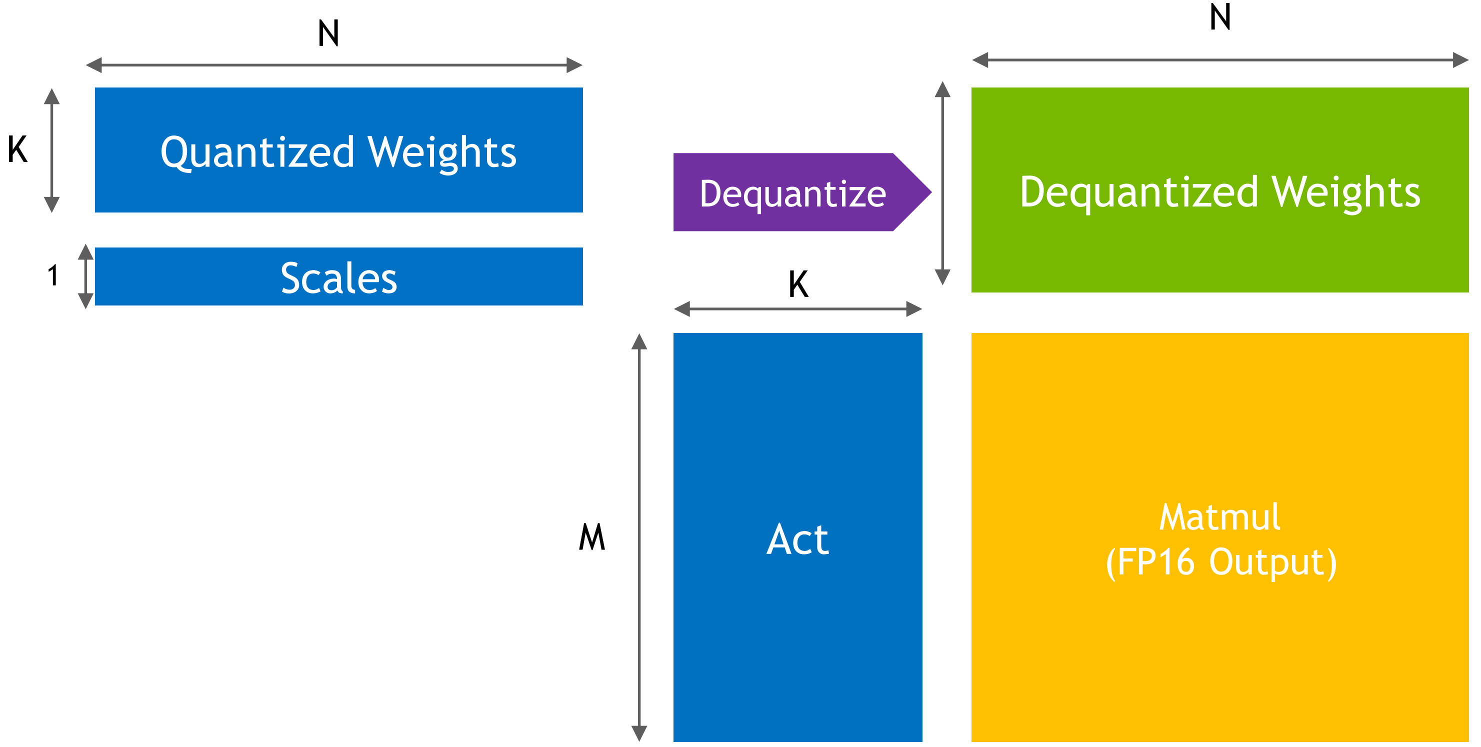


Agenda

- Background on Quantization
- Introduction to the model architecture (Mixture-of-Experts)
- Quantization properties on MoE LLM
- **Custom kernels for Weight-Only Quantization**
- Performance benchmarks
- Accuracy benchmarks

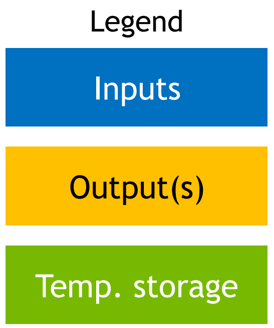
MOTIVATION

Extra Memory Traffic from Unfused Implementation



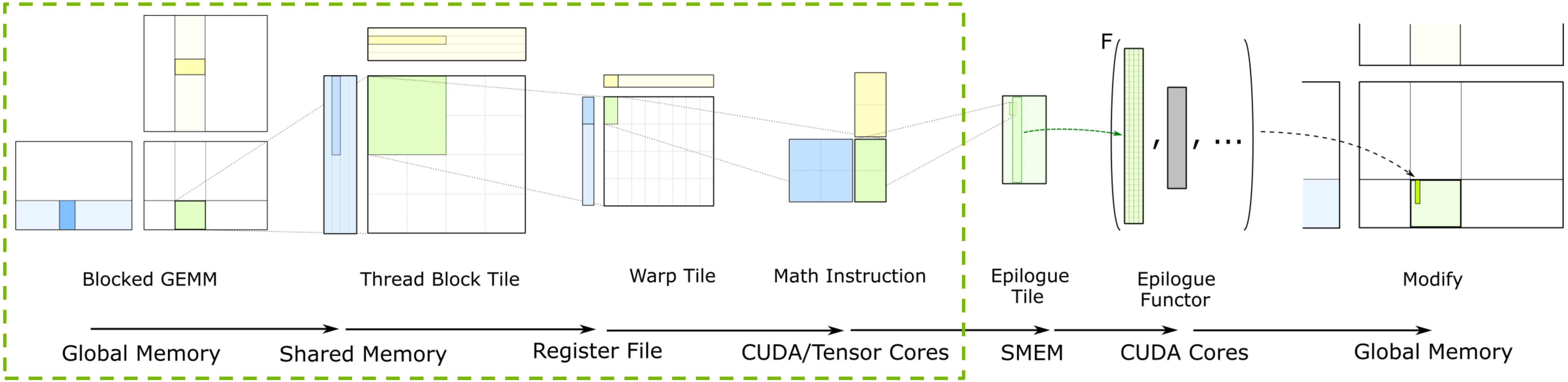
Can reduce model size but must read K x N extra bytes and store 2 x K x N extra bytes.

So, we move an extra 3 x K x N bytes



Approach - CUTLASS

Efficient storing and loading through Shared Memory



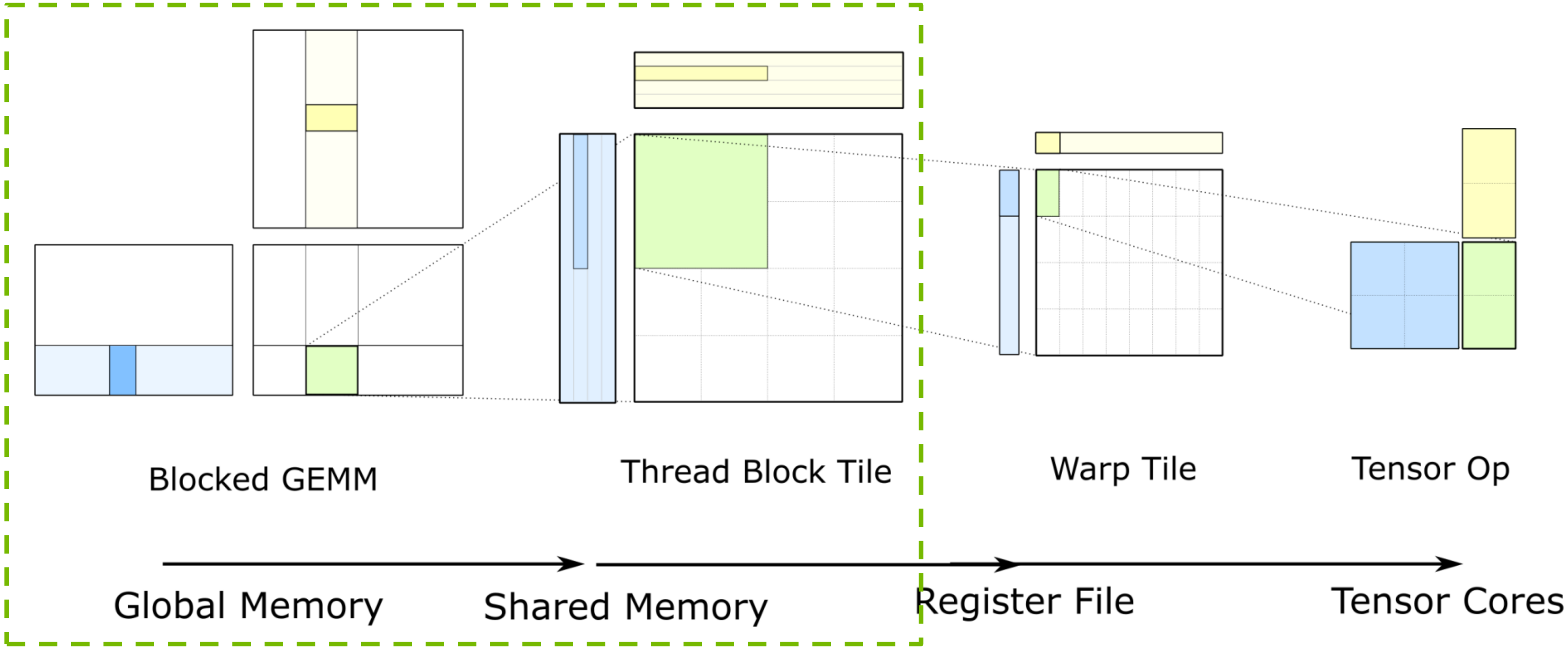
Tiled, hierarchical model: reuse data in Shared Memory and in Registers

See CUTLASS GTC 2020 talk for more details about this model. See S51414 for Hopper model of CUTLASS

Approach - CUTLASS

Move data from Global Memory to Tensor Cores as efficiently as possible

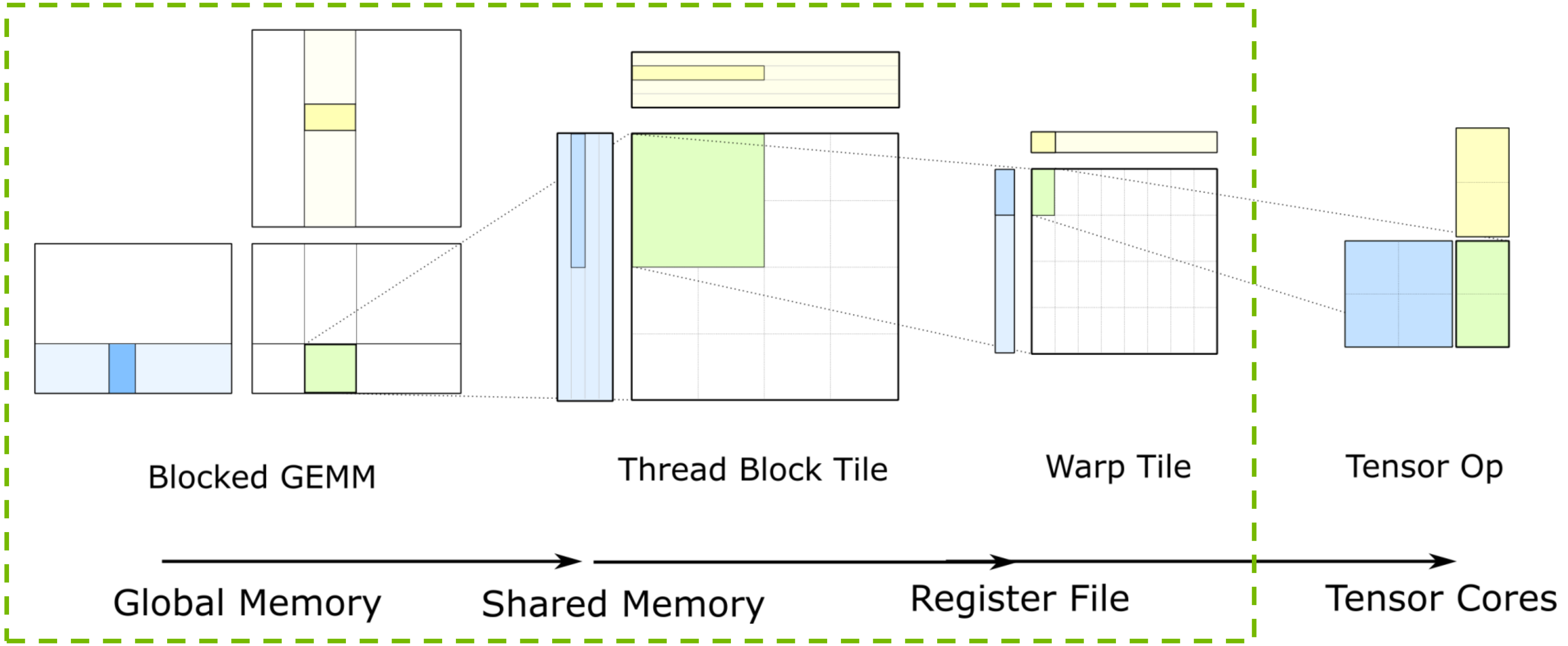
- **Latency-tolerant pipeline from Global Memory**
- Conflict-free Shared Memory stores
- Conflict-free Shared Memory loads



Approach - CUTLASS

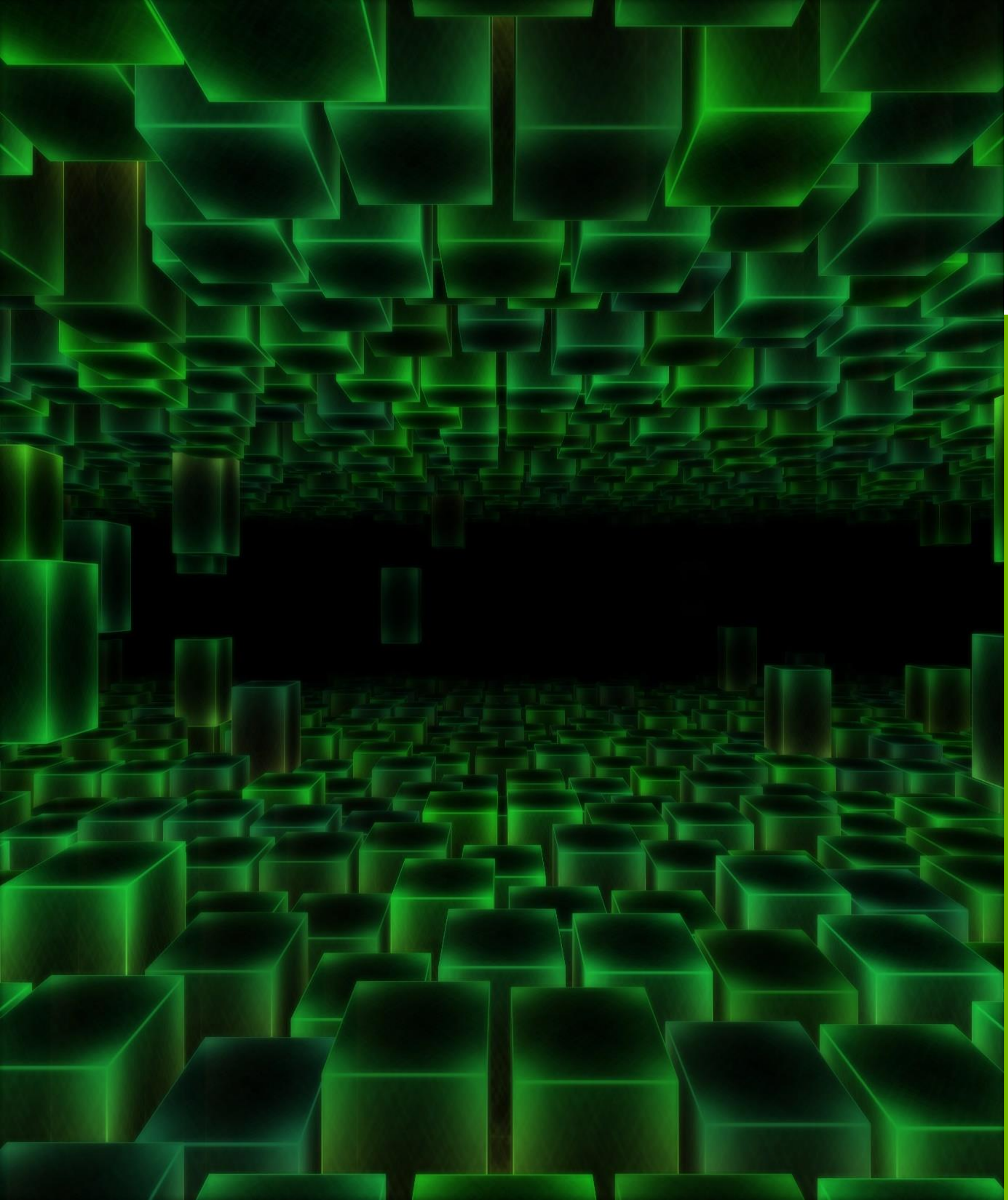
Move data from Global Memory to Tensor Cores as efficiently as possible

- Latency-tolerant pipeline from Global Memory
- **Conflict-free Shared Memory stores**
- **Conflict-free Shared Memory loads**



Overview of Implementation

- Use CUTLASS to efficiently move FP16 activations from global memory to tensor cores for HMMA
- Use CUTLASS to efficiently move INT8 weights from global memory to tensor cores for IMMA
- Convert INT8 weights to FP16 and apply scaling factor
- Issue HMMA and produce FP16 outputs
- Fused kernels reduced DRAM bandwidth as weights are loaded in narrow bit type and expanded on chip.
- Kernels are open sourced. See FasterTransformer for details. See talk S51196 for more details.
- For very large LLMs, we quantize using multiple scales per column. Each block of 64 elements in a column gets its own scaling factor. This is referred to as “fine-grained int4” in subsequent slides. This kernel is not currently open sourced.

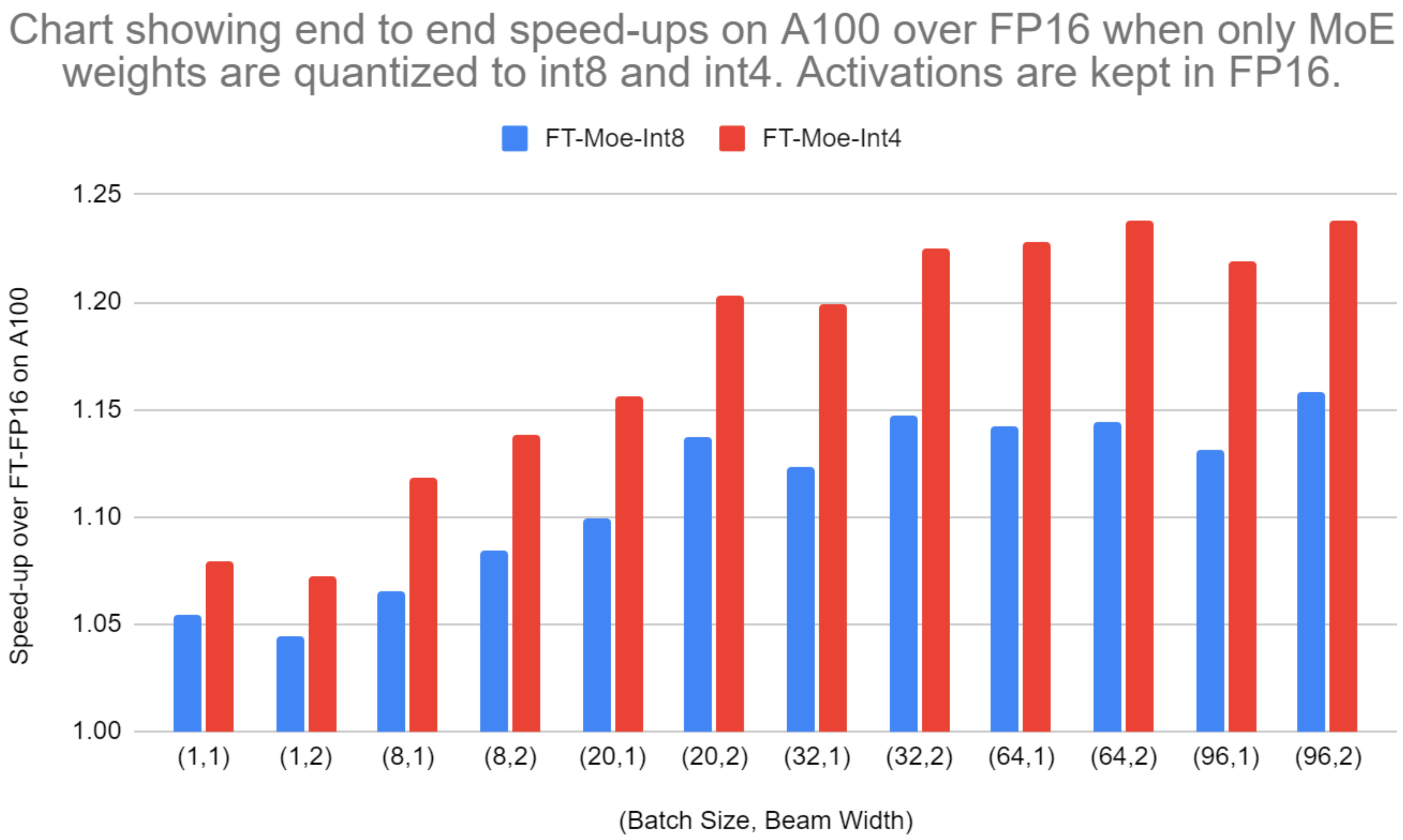


Agenda

- Background on Quantization
- Introduction to the model architecture (Mixture-of-Experts)
- Quantization properties on MoE LLM
- Custom kernels for Weight-Only Quantization
- **Performance benchmarks**
- Accuracy benchmarks

A100 Performance Benchmarks

End to end improvements – Microsoft MoE



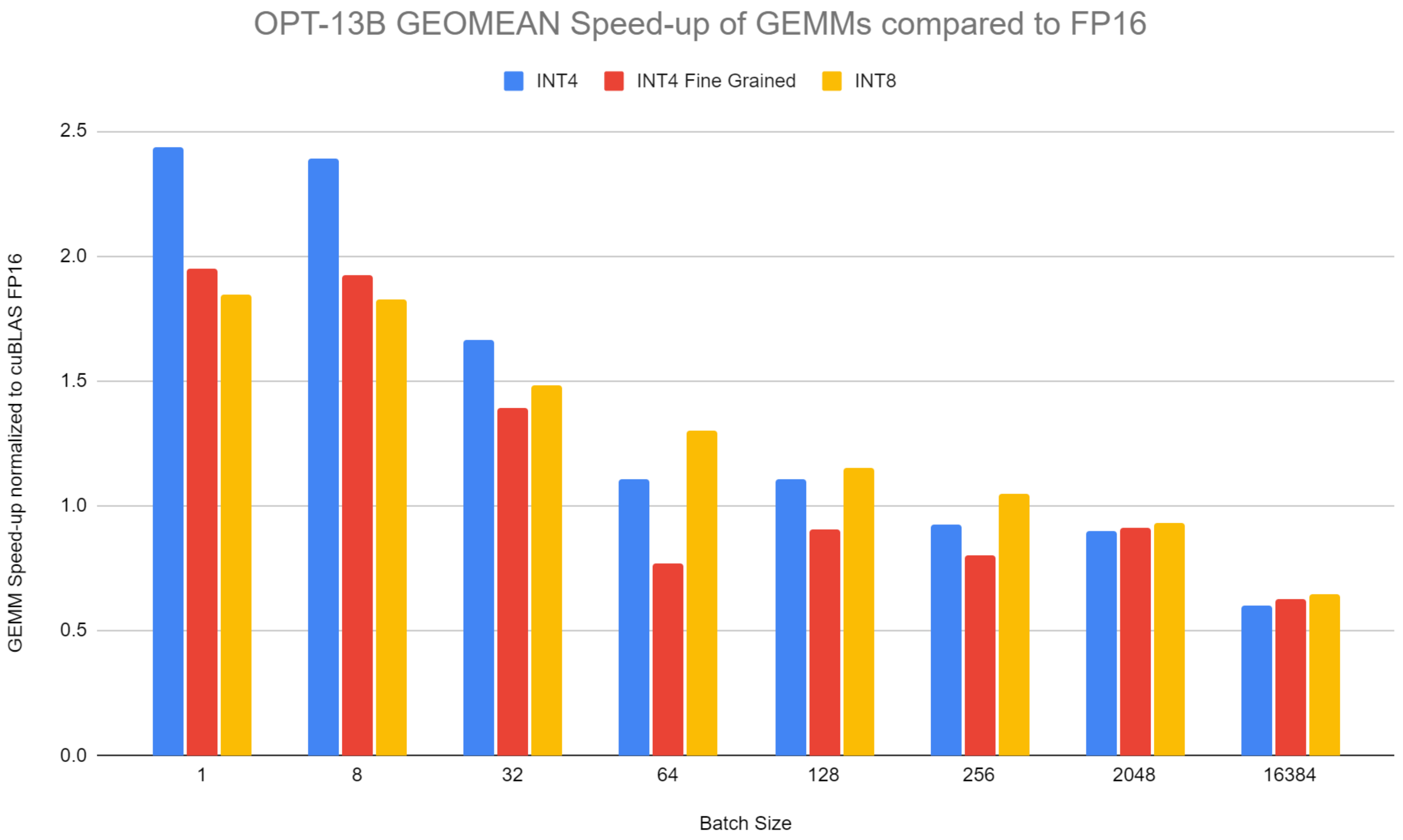
A100 Performance Benchmarks

What gets Quantized?

- QKV input projection Weight [Hidden, 3 x Hidden]
- QKV output projection Weight [Hidden x Hidden]
- FFN 1 Weight [Hidden x InterSize] – Note: InterSize is usually 4 x Hidden
- FFN 2 Weight [InterSize x Hidden]

A100 Performance Benchmarks

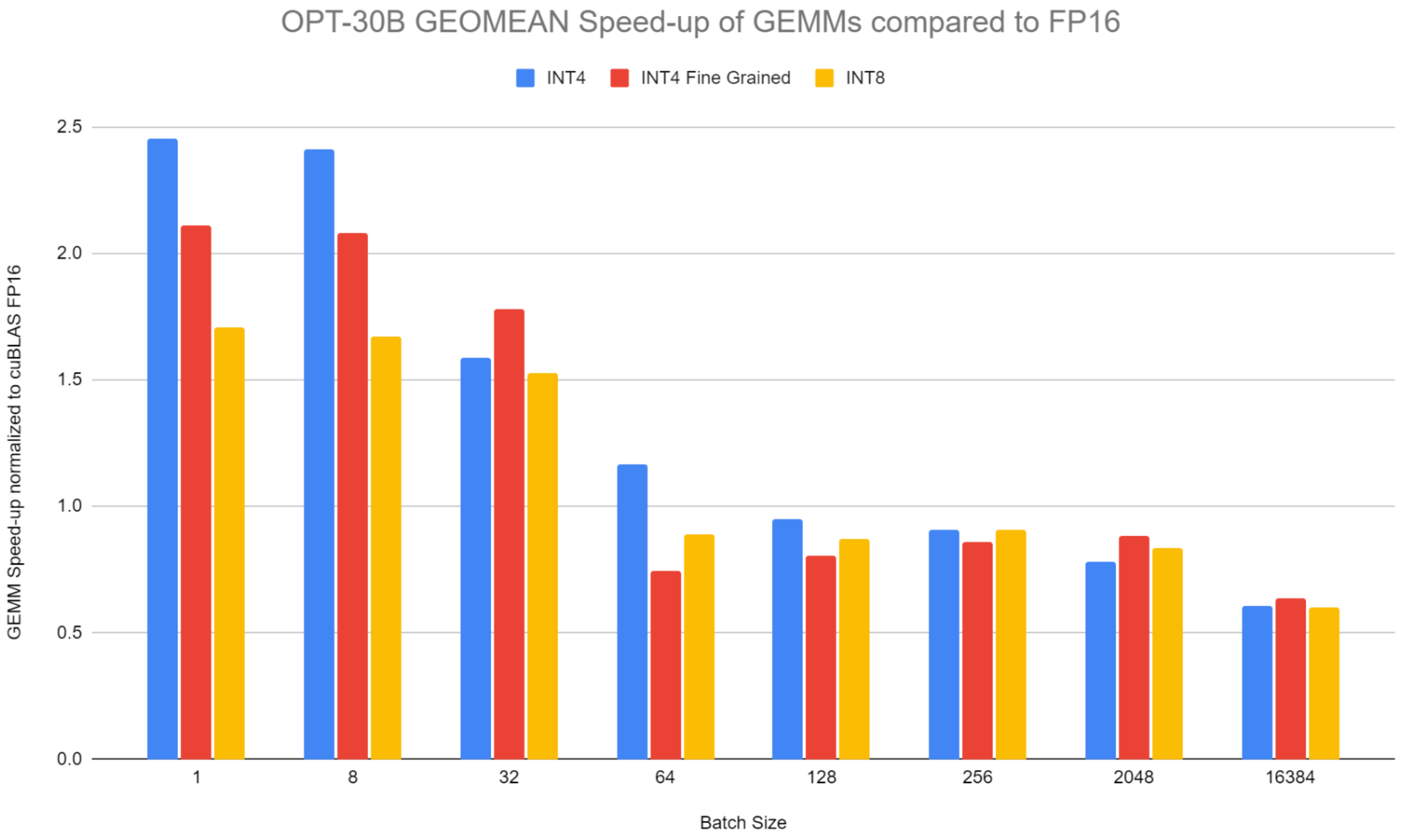
Micro benchmark - OPT-13b GEOMEAN GEMM Speedup for different batch sizes



- QKV input projection GEMM [5120, 15360]
- QKV output projection GEMM [5120 x 5120]
- FFN 1 GEMM [5120 x 20480]
- FFN 2 GEMM [20480 x 5120]
- Geomean across these 4 shapes, varying the batch size
- Clocks locked to MEM=1593 MHz, SM=1410Mhz

A100 Performance Benchmarks

Micro benchmark - OPT-30b GEOMEAN GEMM Speedup for different batch sizes

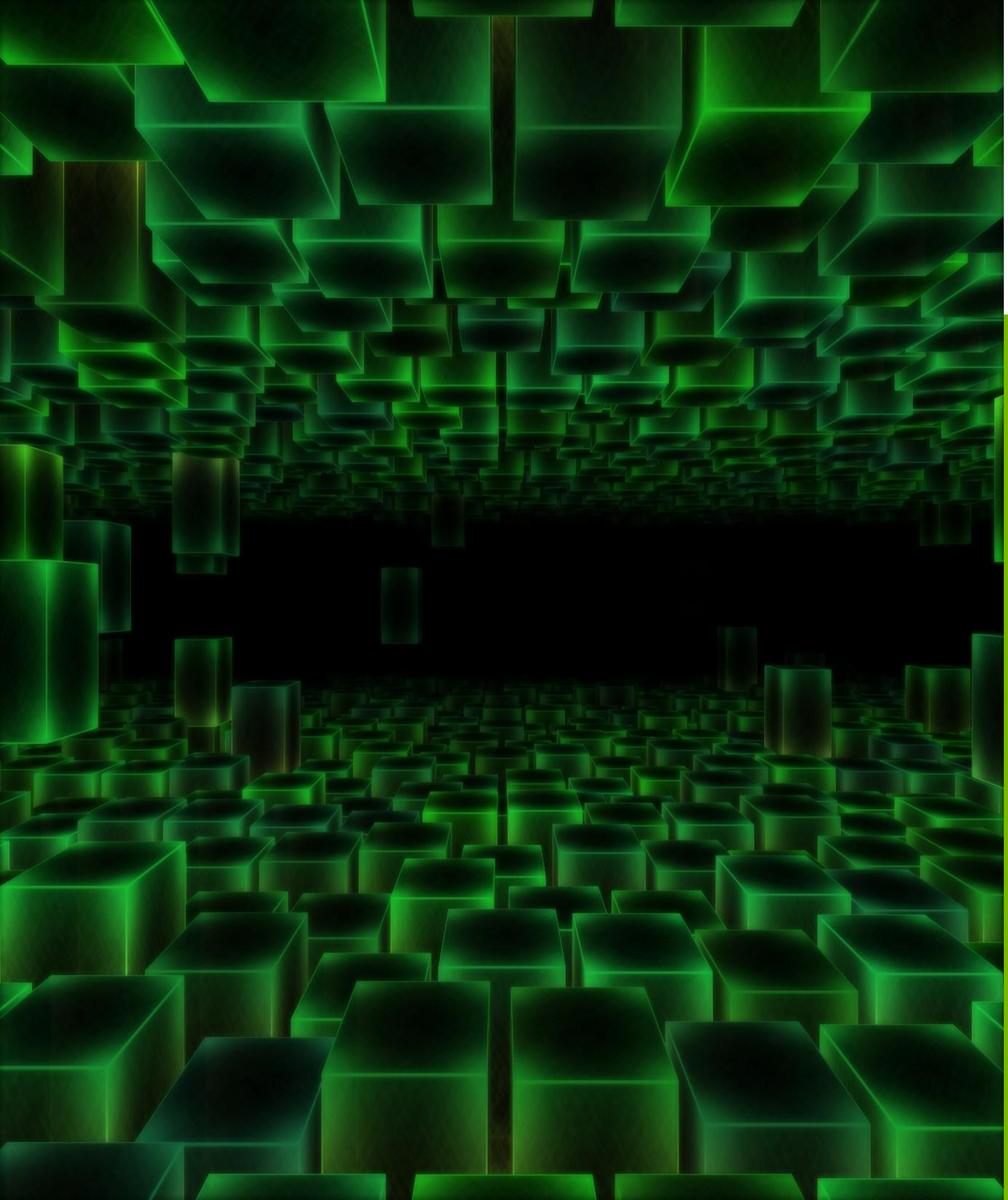


- QKV input projection GEMM [7168, 21504]
- QKV output projection GEMM [7168 x 7168]
- FFN 1 GEMM [7168 x 28672]
- FFN 2 GEMM [28672 x 7168]
- Geomean across these 4 shapes, varying the batch size
- Clocks locked to MEM=1593 MHz, SM=1410Mhz

A100 Performance Benchmarks

When do we start seeing perf gains?

- On A100, started seeing perf gains on small batch from OPT 2.7B.
- In general, it depends on how much memory bandwidth GPU has. Cards will lower memory bandwidth will see performance improvements on smaller models, since this kernel only reduces traffic from GPU HBM.
- In addition, cards with lower memory bandwidth will see more performance upside overall.



Agenda

- Background on Quantization
- Introduction to the model architecture (Mixture-of-Experts)
- Quantization properties on MoE LLM
- Custom kernels for Weight-Only Quantization
- Performance benchmarks
- **Accuracy benchmarks**

Accuracy Benchmarks

GPT2-XL (1.5B params)

	LAMBADA OPENAI	HellaSwag	PIQA	WinoGrande	OpenBookQA	RTE	COPA	Average	Wikitext [Perplexity]
FP16	51.1%	40.0%	70.7%	58.2%	22.4%	52.3%	73.0%	**52.5%**	20.38
INT8	51.1%	40.0%	70.7%	58.3%	22.6%	52.7%	73.0%	**52.6%**	20.39
INT4-Fine grained	49.3%	39.6%	70.7%	58.4%	20.6%	50.9%	74.0%	**51.9%**	20.87
Int4	47.5%	37.4%	69.4%	57.1%	19.4%	51.9%	73.0%	**50.8%**	21.7

Accuracy benchmarks were done using LM Evaluation Harness

Accuracy Benchmarks

OPT 13b

	LAMBADA OPENAI	HellaSwag	PIQA	WinoGrande	OpenBookQA	RTE	COPA	Average	Wikitext [Perplexity]
FP16	68.6%	52.5%	75.9%	65.0%	26.6%	58.1%	86.0%	**61.8%**	11.5
INT8	68.5%	52.4%	76.0%	65.4%	27.2%	57.0%	86.0%	**61.8%**	11.5
INT4-Fine grained	67.4%	50.7%	75.6%	65.4%	25.8%	59.2%	84.0%	**61.2%**	12.0
Int4	65.5%	50.2%	75.5%	64.8%	26.4%	56.0%	85.0%	**60.5%**	12.8

Accuracy benchmarks were done using LM Evaluation Harness

Accuracy Benchmarks

OPT 30b

	LAMBADA OPENAI	HellaSwag	PIQA	WinoGrande	OpenBookQA	RTE	COPA	Average	Wikitext [Perplexity]
FP16	71.5%	54.3%	77.6%	68.2%	30.2%	57.4%	82.0%	**63.0%**	10.7
INT8	71.4%	54.3%	77.6%	67.9%	30.2%	58.1%	82.0%	**63.0%**	10.7
INT4-Fine grained	69.9%	53.4%	77.5%	67.3%	30.0%	56.0%	83.0%	**62.4%**	11.1
Int4	69.5%	51.9%	75.8%	66.3%	26.8%	54.9%	79.0%	**60.1%**	11.6

Accuracy benchmarks were done using LM Evaluation Harness

Note that OPT 30B with per-column int4 quantization is worse than OPT 13B in FP16!
We recover some of this accuracy with fine grained quantization

Accuracy Benchmarks

BLOOM-176B

	LAMBADA
FP16 (8 GPUs)	67.79%
INT8 (4 GPUs)	67.86%
INT4-Fine grained (2 GPUs)	67.44%

Accuracy benchmarks were done using example FT script.

THANK YOU!