Foundations of Edge AI

Lecture17

Efficient LLM Deployment

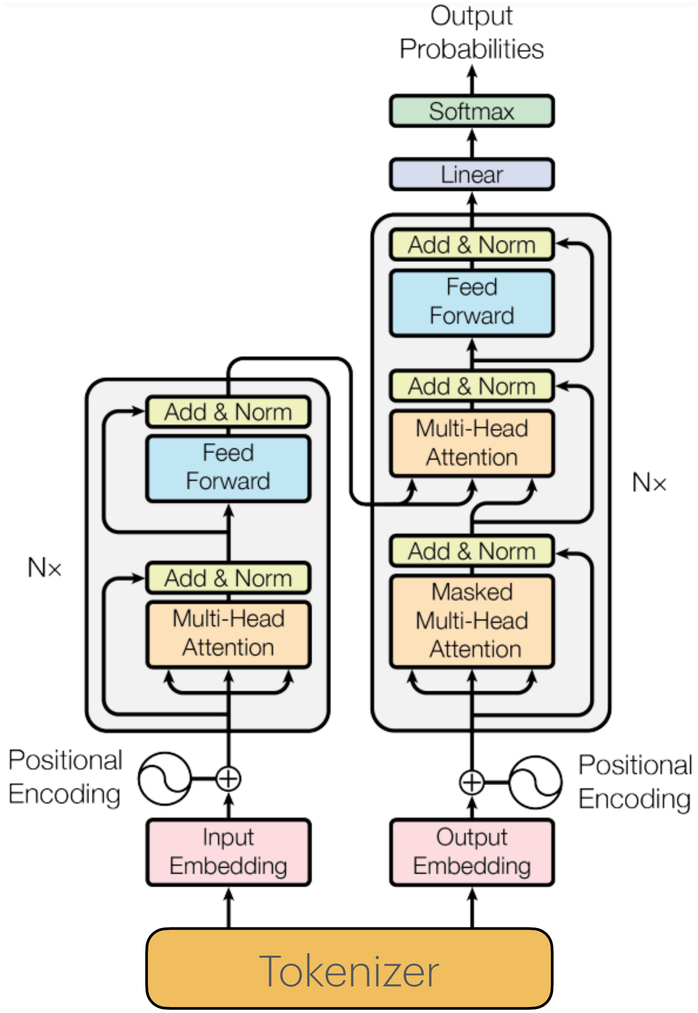
Lanyu (Lori) Xu

Email: lxu@oakland.edu
Homepage: https://lori930.github.io/
Office: EC 524

Recap: Transformer and LLM

- Transformer basics
- Transformer design variants
- Large Language Models (LLMs)

Transformer



A step-by-step Transformer model tutorial

- Tokenize words (word → tokens)
- Map tokens into embeddings
- Embeddings go through Transformer blocks
  - Multi-Head Attention (MHA)
  - Feed-Forward Network (FFN)
  - LayerNorm
  - Residual connection
- Positional encoding
- Final prediction with linear head

Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

Transformer Design Variants

Improved designs after the initial transformer paper

- Encoder-decoder (T5), encoder-only (BERT), decoder-only (GPT)
- Absolute positional encoding → Relative positional encoding
- KV cache optimizations:
  - Multi-Head Attention (MHA) → Multi-Query Attention (MQA) → Grouped-Query Attention (GQA)
- Feed-Forward Network (FFN) → Gated Linear Unit (GLU)

Large Language Models (LLMs)

Scaling up unlocks more capabilities

- **GPT-3:** scaling up Transformers to be few-shot learners
  - Scaled-up LLM (175B) can generalize to new tasks **w/o fine-tuning** by either zero-shot or few-shot
- **OPT:** Open-source Pre-trained Transformer Language Models from Meta
  - Design choices: Decoder-only, pre-norm, SwiGLU (Swish, gated linear unit), rotary positional embedding (RoPE)
- **LLaMA**: significantly better opensource LLMs
- **Llama2 & Llama3:** better performance from a larger training corpus
  - Grouped-query attention to fit longer context (2k -> 4k -> 8k)
- **Mistral-7B:** small model with superior performance

How to Scale Up?

The Chinchilla Law

- Scale up **both the model size and data size** for training to have the best **training** computation vs. accuracy trade-off

- Note: the trade-off is different if we consider the **inference** computation trade-off

  - You want to train a smaller model longer to save inference costs (e.g., LLaMA)

Parameters	FLOPs	FLOPs (in *Gopher* unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4.761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

Llama-2 model goes beyond the #tokens by a lot (7B, 2T tokens)

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., ... & Sifre, L. (2022). Training compute-optimal large language models. arXiv preprint arXiv:2203.15556.

Lecture Plan

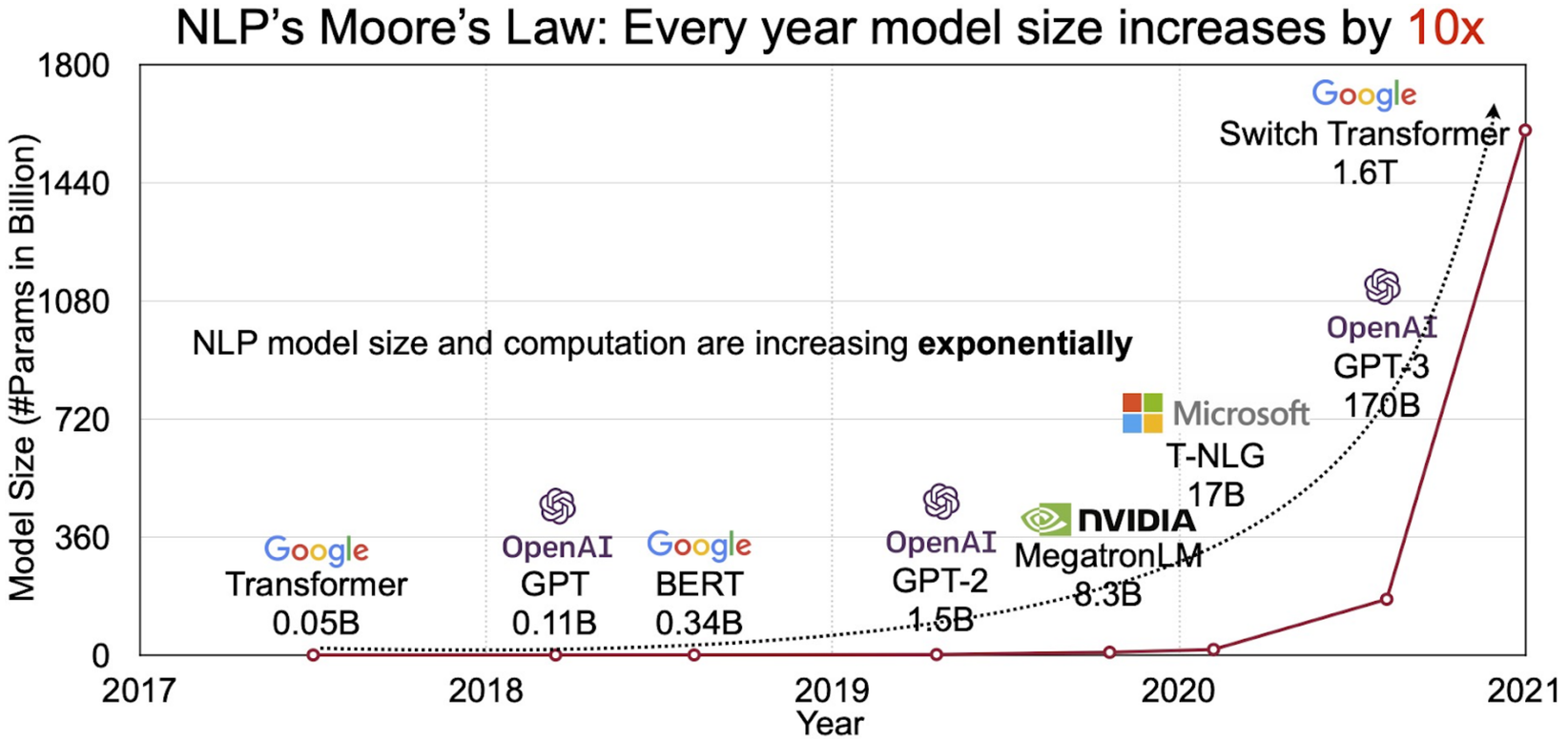
Today we will...

- Introduce quantization in LLMs
  - Weight-activation quantization: SmoothQuant
  - Weight-only quantization: AWQ
- Introduce pruning and sparsity in LLMs
  - Weight sparsity: Wanda
  - Contextual sparsity: DejaVu, MoE
  - Attention sparsity: SpAtten, H2O
- LLM Serving Systems
  - Metrics for LLM Serving
  - vLLM and Paged Attention
  - FlashAttention
  - Speculative Decoding
  - Batching

Large Language Models (LLMs)

The growing trend of scaling up

- LLMs are scaled-up transformer models trained on large language corpus (natural language, code, etc.)

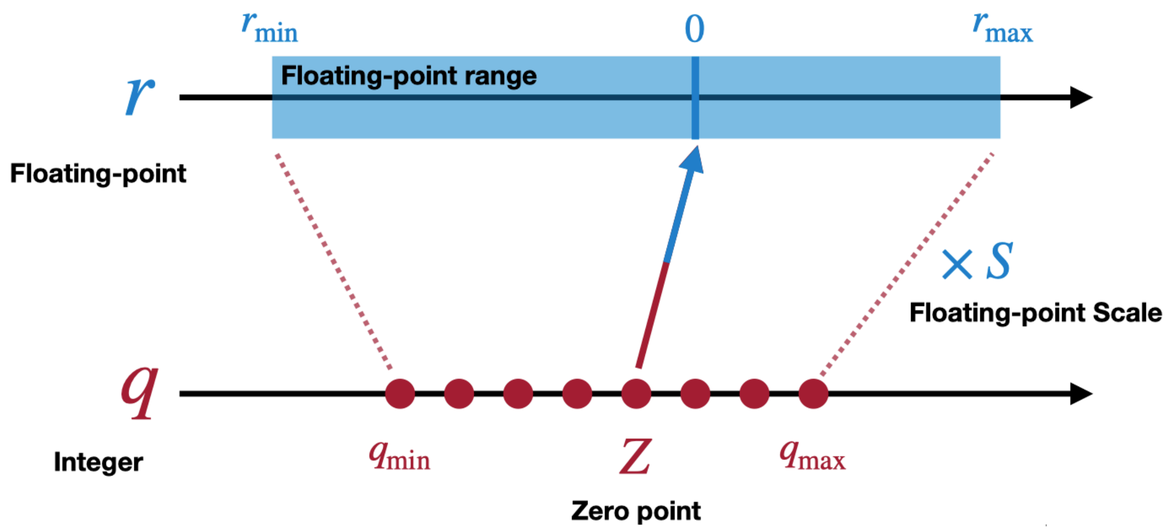


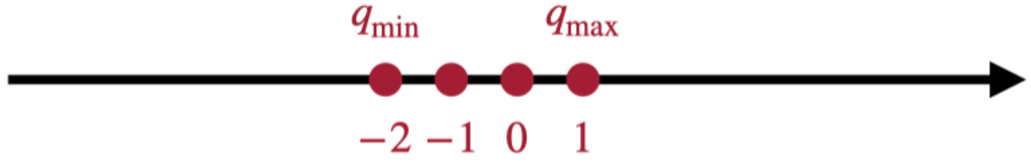
Source

Quantization

Quantization

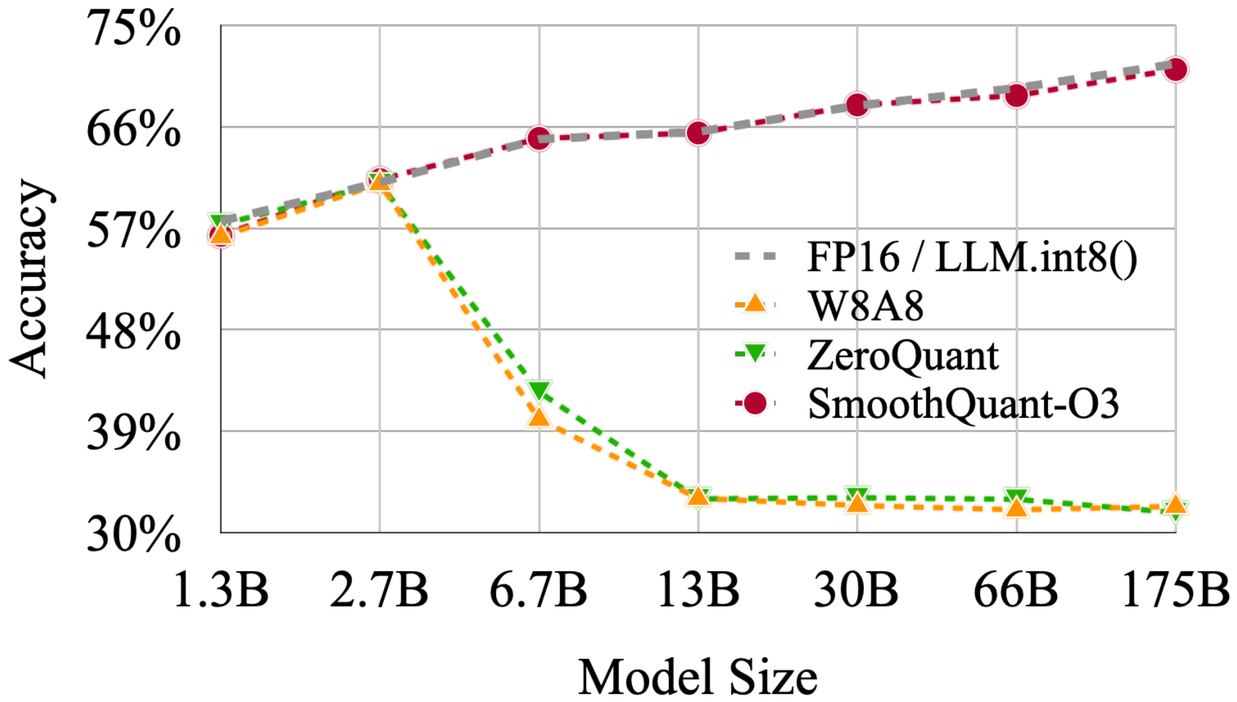
Lowers the bid-width and improves efficiency

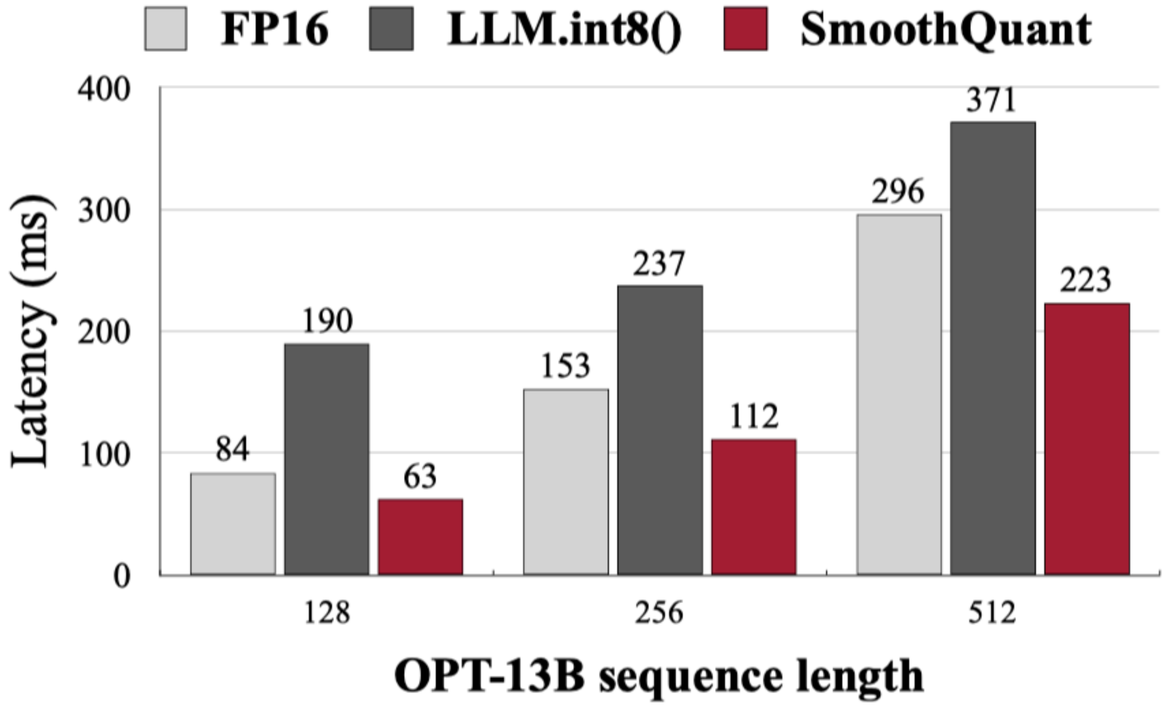




Binary	Decimal
01	1
00	0
11	-1
10	-2

Existing Quantization Methods is Slow or Inaccurate



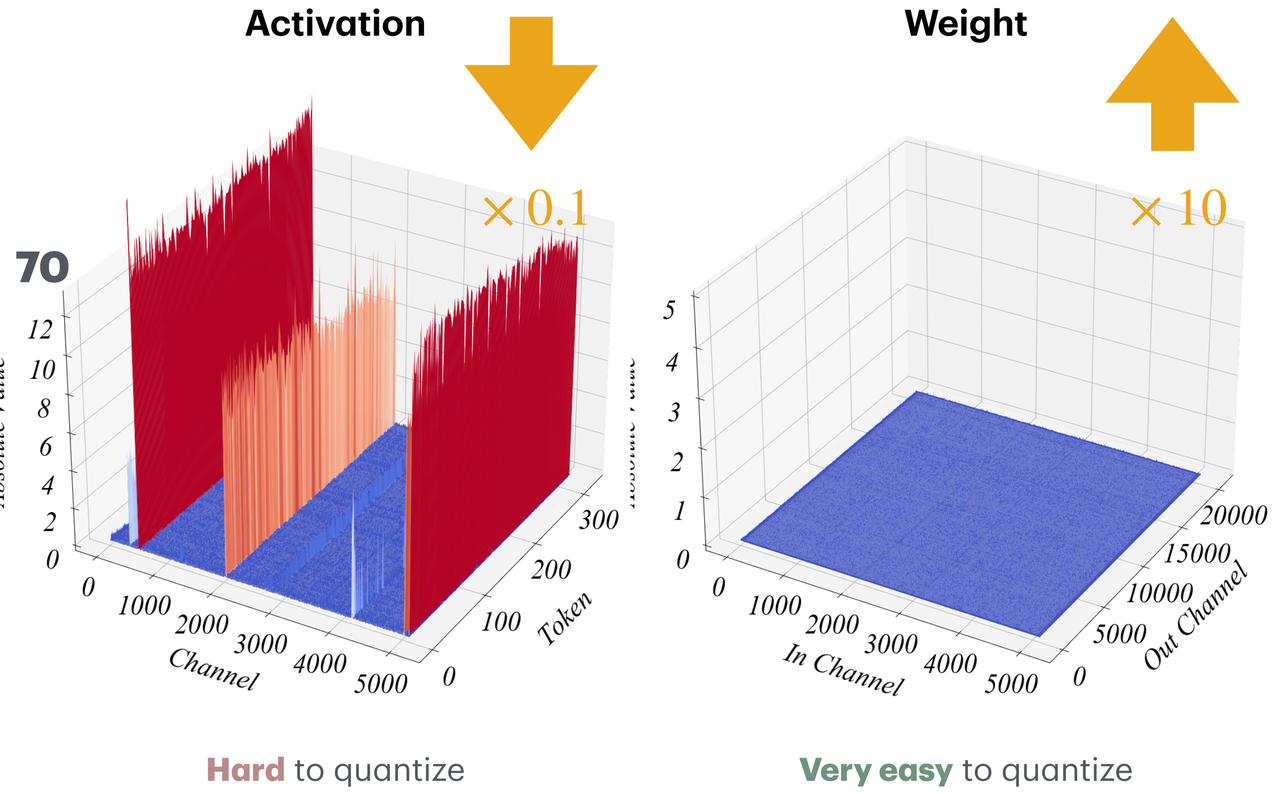


- **W8A8 quantization has been an industry standard for CNNs**, but not LLM. Why?
- Systematic outliers emerge in **activations** when LLMs are scaled up beyond 6.7B. Traditional CNN quantization methods will destroy the accuracy.
- The accuracy-preserving baseline, LLM.int8() uses FP16 to represent outliers, which needs runtime outlier detection, scattering and gathering, slowing the inference.

Dettmers, T., Lewis, M., Belkada, Y., & Zettlemoyer, L. (2022). Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. Advances in Neural Information Processing Systems, 35, 30318-30332.

The Quantization Difficulty of LLMs

Activation outliers destroy quantized performance



Hard to quantize

Very easy to quantize

Can we **balance** the quantization difficulty between activation and quantization?

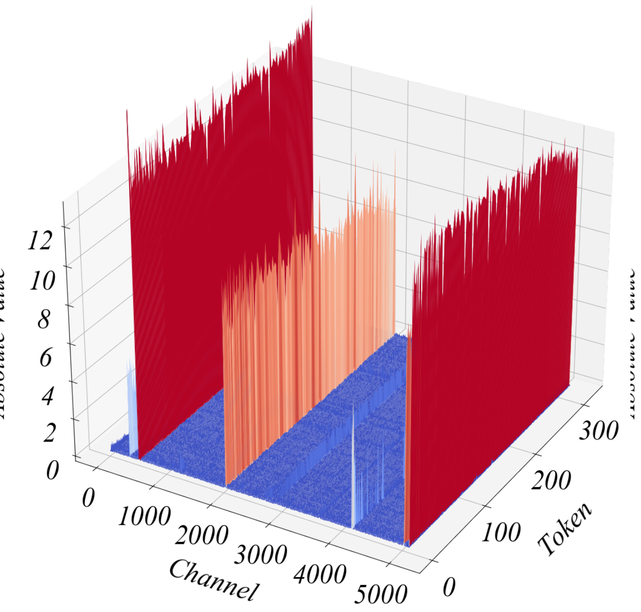
- LLMs are notoriously difficult to quantize because
  - **Activations are harder to quantize** than weights
    - While quantizing the weights with INT8 or INT4 doesn't degrade accuracy
  - **Outliers** make activation quantization difficult
    - ~100x larger than most of the activation values, therefore most values will be zeroed out in INT8 quantization
  - Outliers persist in **fixed channels**
    - And the outlier channels are persistently large

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. (2023, July). Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning (pp. 38087-38099). PMLR.

The Quantization Difficulty of LLMs

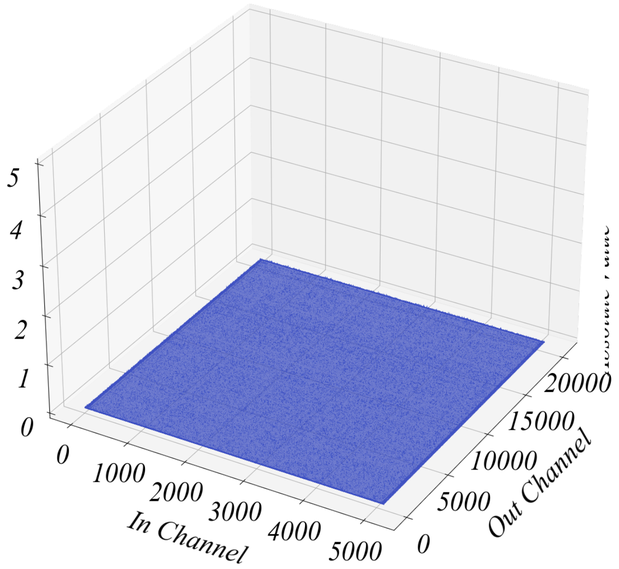
Activation outliers destroy quantized performance

Activation



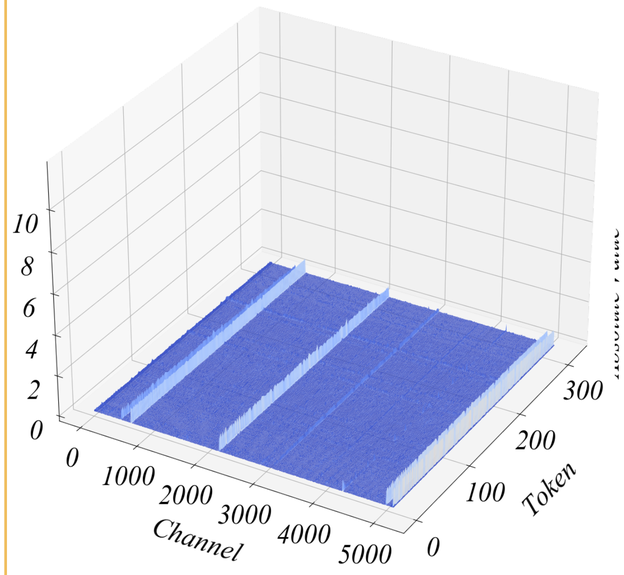
Hard to quantize

Weight



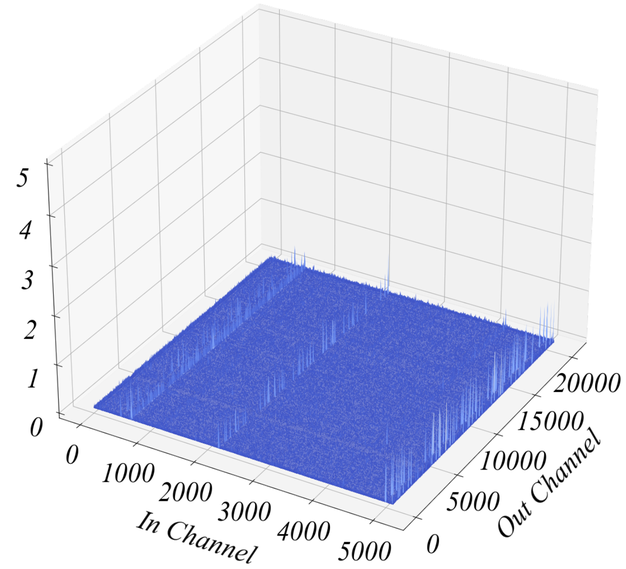
Very easy to quantize

Activation (after smoothed)



Easy to quantize

Weight (after smoothed)



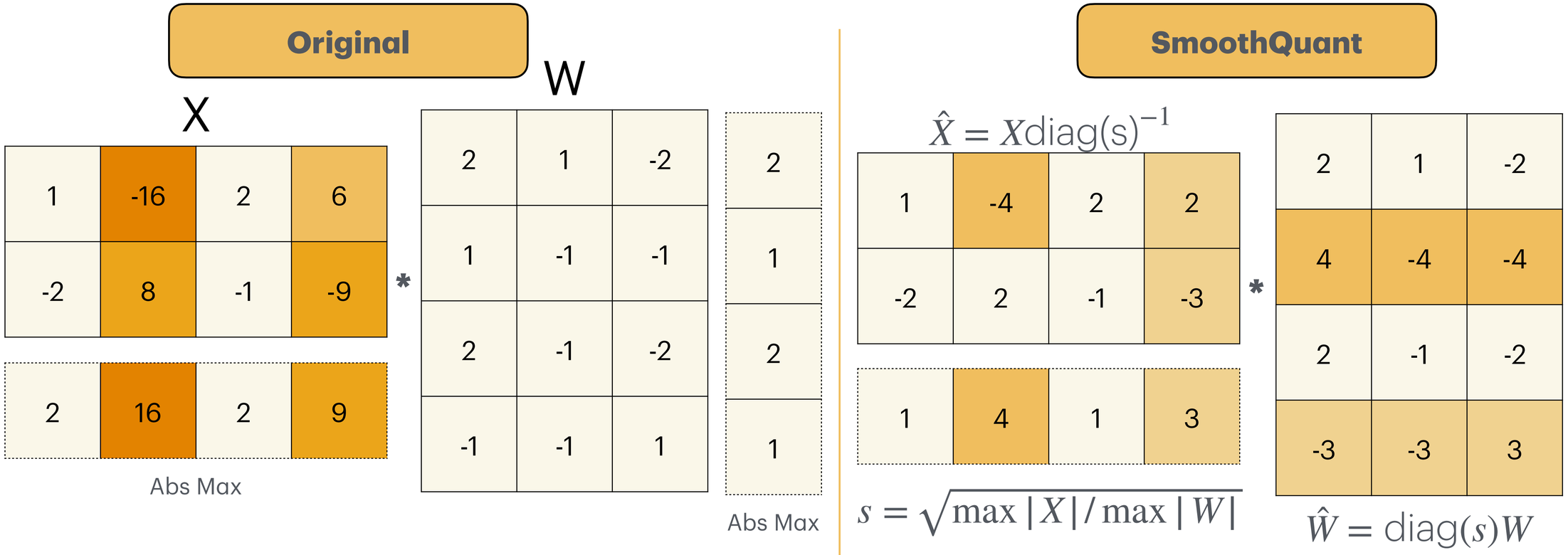
Harder but still easy to quantize

We can smooth the outlier channels in activations by **migrating their magnitudes into the following weights,** so both are easy to quantize.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. (2023, July). Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning (pp. 38087-38099). PMLR.

Activation Smoothing

Overview



$$s_j = \max(|X_j|)^{\alpha} / \max(|W_j|)^{1-\alpha}, j = 1,2,...,C_i$$

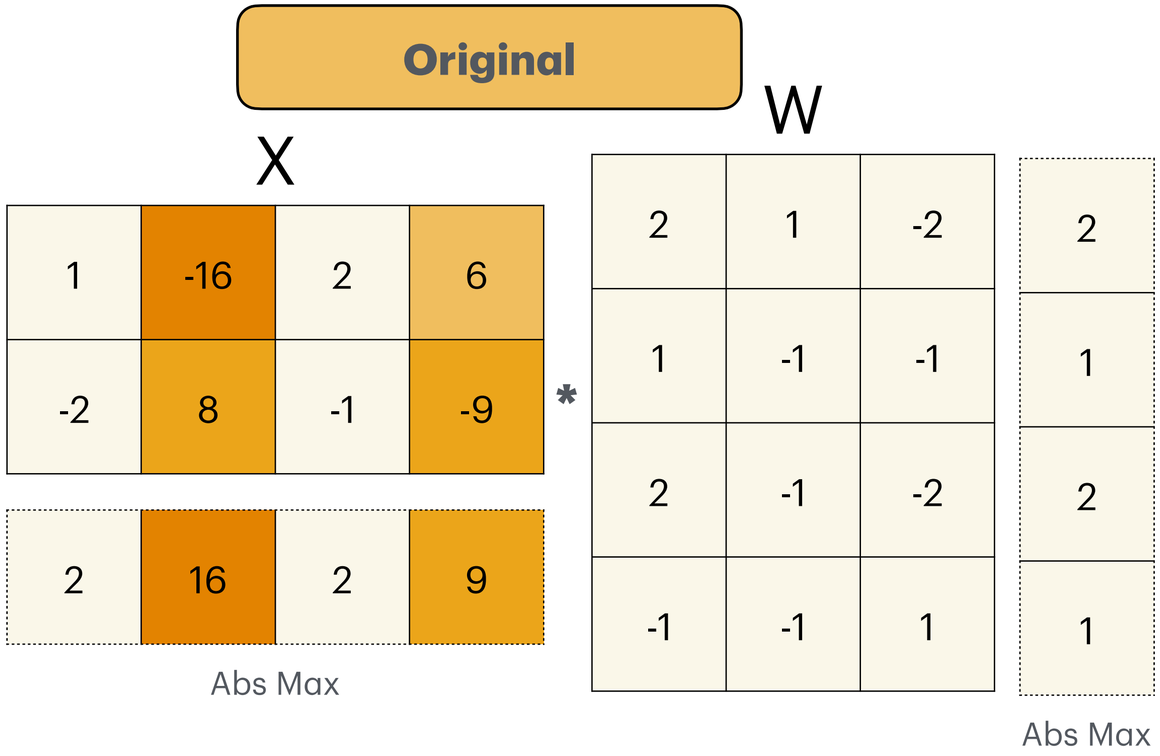
$$Y = (X\text{diag}(s)^{-1}) \cdot (\text{diag}(s)W) = \hat{X}\hat{W}$$

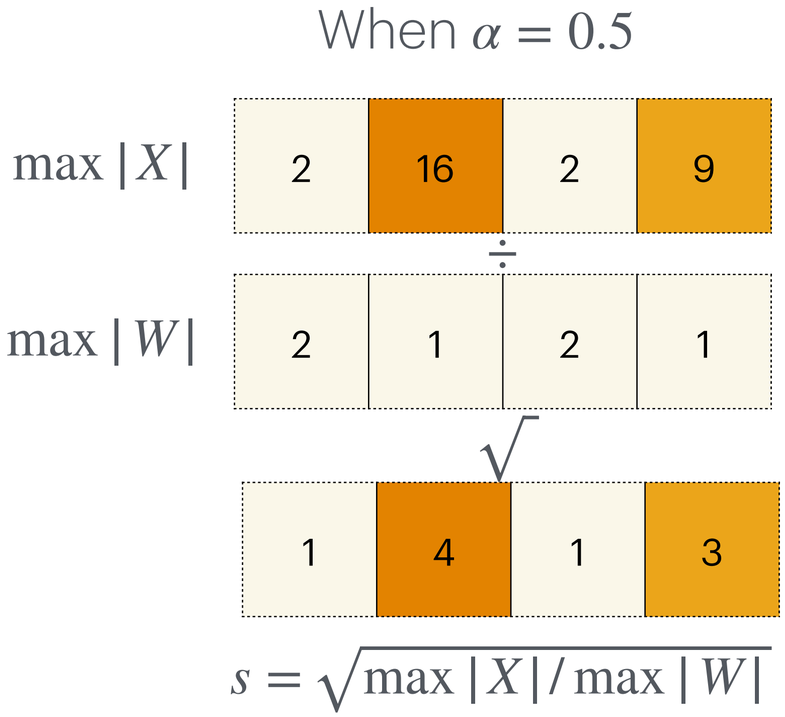
α : Migration Strength

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. (2023, July). Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning (pp. 38087-38099). PMLR.

Activation Smoothing

1. Calibration Stage (Offline)





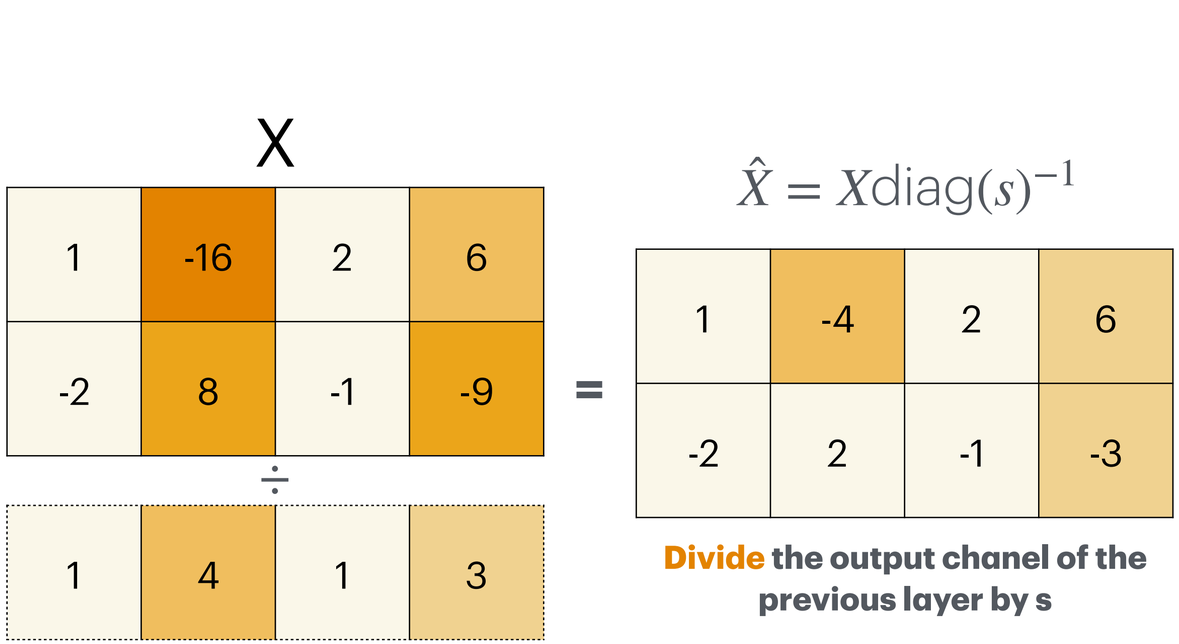
$$s_j = \max(|X_j|)^{\alpha} / \max(|W_j|)^{1-\alpha}, j = 1,2,...,C_i$$

α : Migration Strength

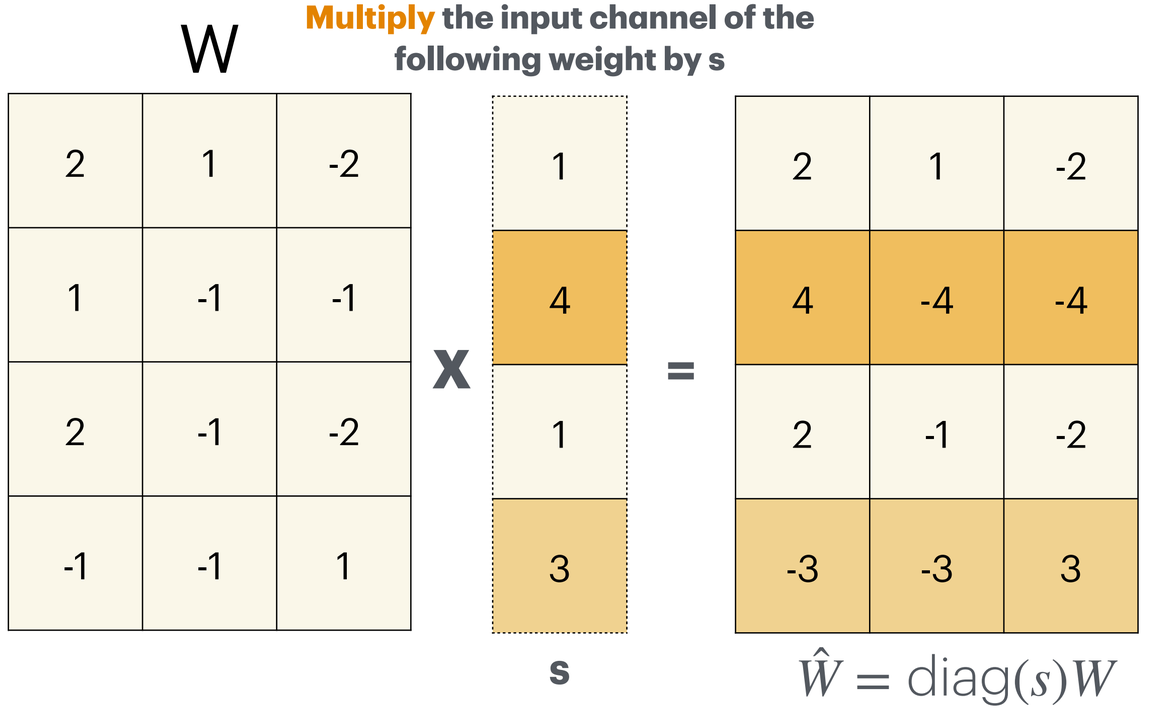
Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. (2023, July). Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning (pp. 38087-38099). PMLR.

Activation Smoothing

2. Smoothing Stage (Offline)



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



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

$$s_j = \max(|X_j|)^{\alpha} / \max(|W_j|)^{1-\alpha}, j = 1,2,...,C_i$$

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

α : Migration Strength

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. (2023, July). Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning (pp. 38087-38099). PMLR.

Activation Smoothing

3. Inference (deployed model)

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

1	-4	2	6
-2	2	-1	-3

At runtime, the activations are smooth and easy to quantize

*

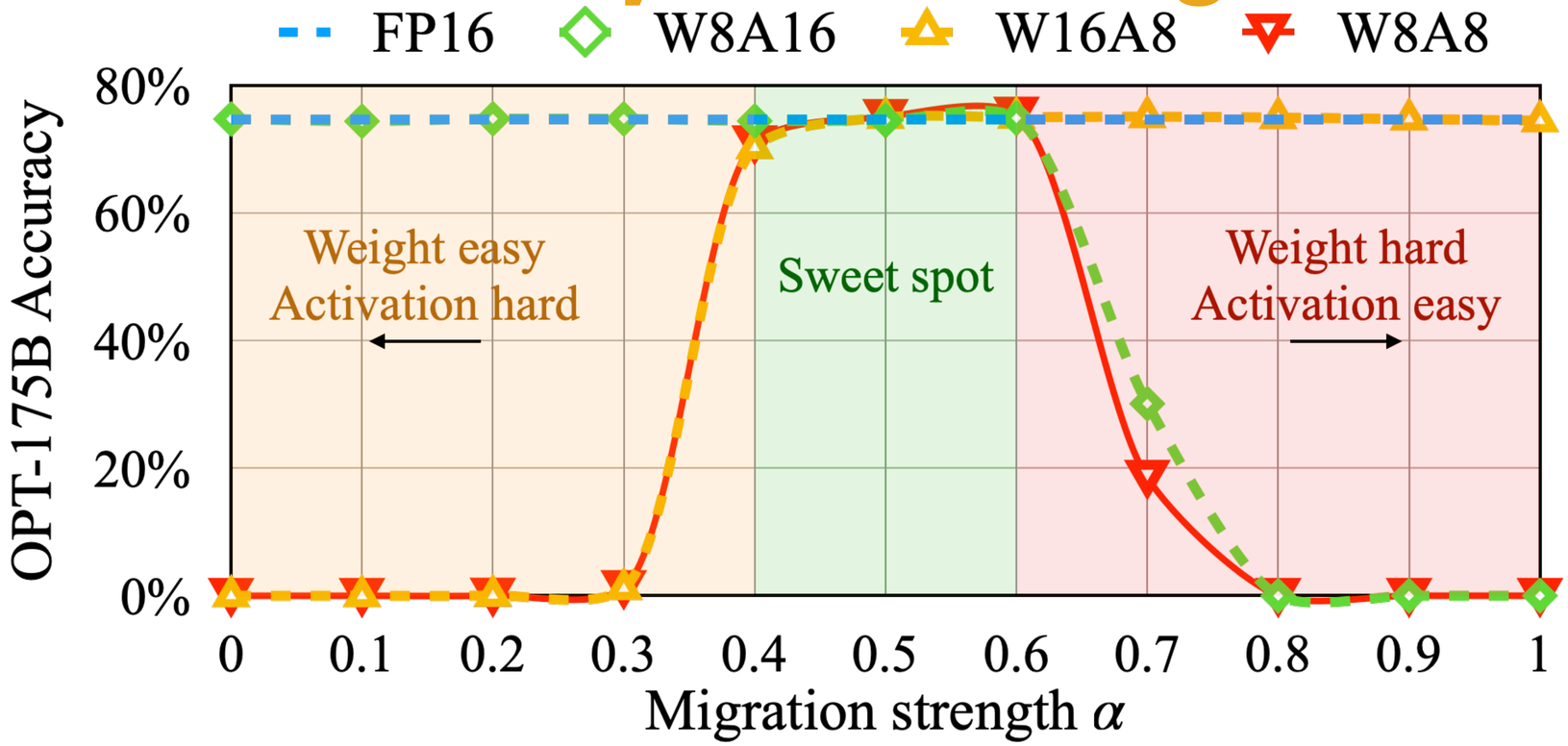
2	1	-2
4	-4	-4
2	-1	-2
-3	-3	3

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

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

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. (2023, July). Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning (pp. 38087-38099). PMLR.

Ablation Study on the Migration Strength α



🤩 **A good practice:** $\alpha = 0.5$

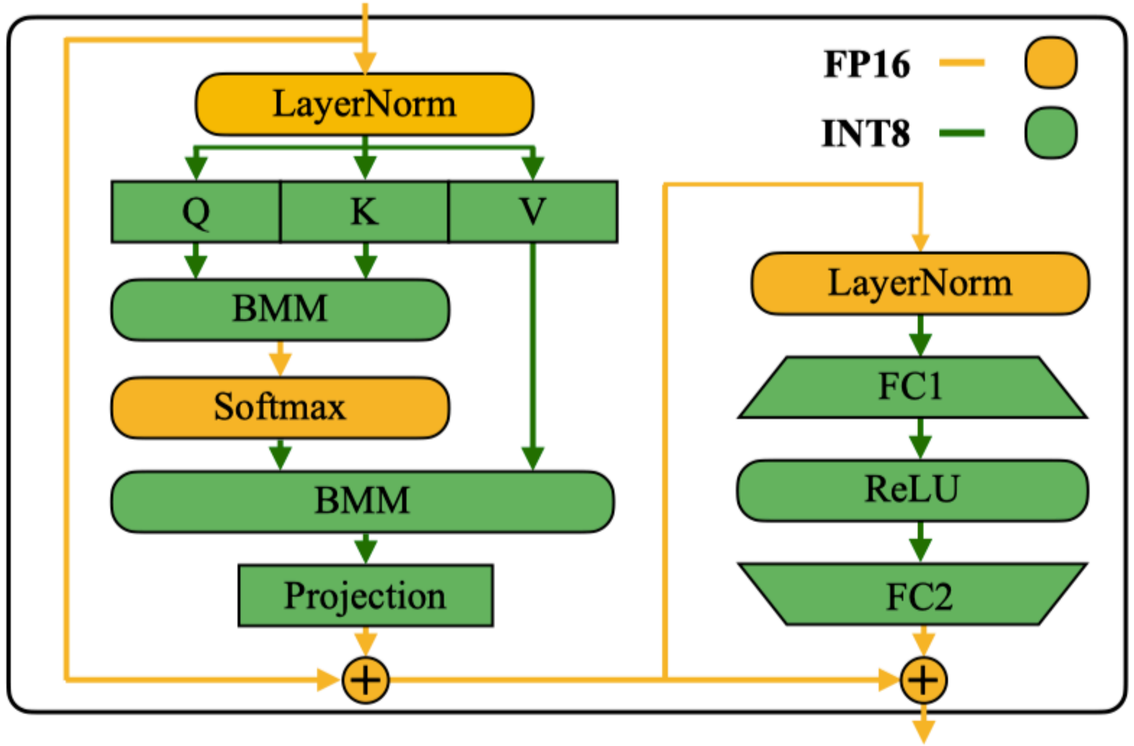
$$s_j = \max(|X_j|)^{\alpha} / \max(|W_j|)^{1-\alpha}, j = 1,2,...,C_i \qquad Y = (X\text{diag}(s)^{-1}) \cdot (\text{diag}(s)W) = \hat{X}\hat{W}$$

- Migration strength α controls the amount of quantization difficulty **migrated from activations to weights**
- A suitable migration strength α (**sweet spot**) makes both activation 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

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. (2023, July). Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning (pp. 38087-38099). PMLR.

SmoothQuant System Implementation

SmothQuant's precision mapping for a Transformer block



Integrated into FasterTransformer, a SOTA Transformer serving framework

- **All compute-intensive operators**, such as linear layers and batched matrix multiplications (BMMs) use **INT8** arithmetic

Quantization settings of the baseline and SmoothQuant (INT8)

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

- Three efficiency levels of quantization settings for SmoothQuant
  - Efficiency: O1 < O2 < O3

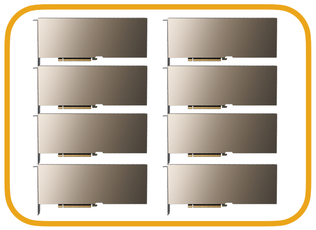
Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. (2023, July). Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning (pp. 38087-38099). PMLR.

SmoothQuant is Accurate and Efficient

- Well **maintains the accuracy** without fine-tuning

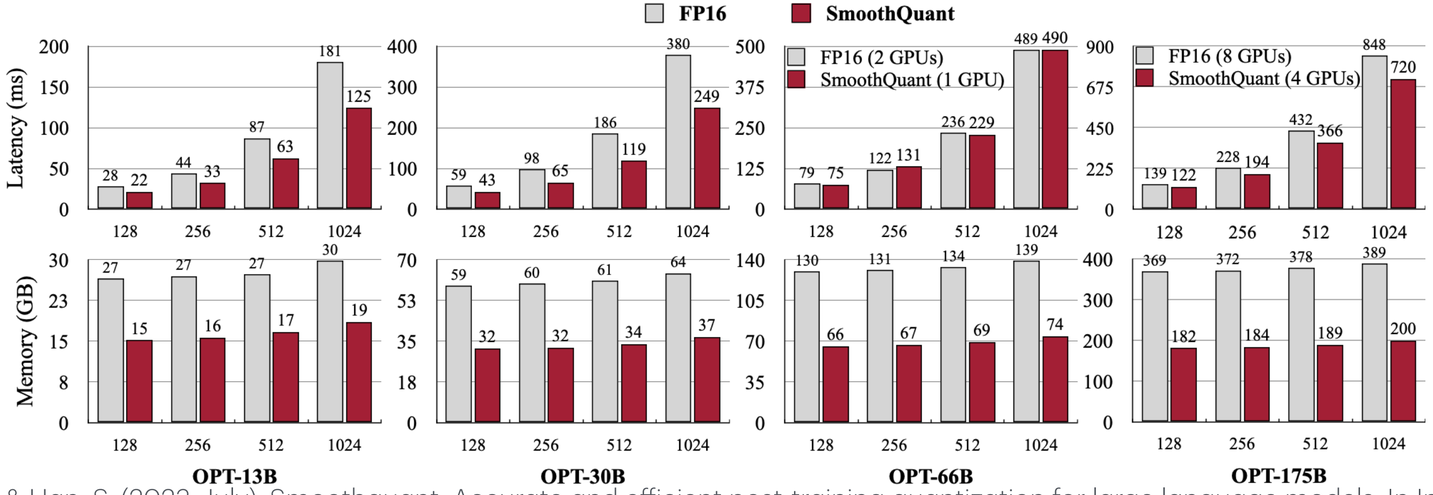
Method	OPT-175B	BLOOM-176B	GLM-130B*
FP16	71.6%	68.2%	73.8%
W8A8	32.3%	64.2%	26.9%
ZeroQuant	31.7%	67.4%	26.7%
`LLM.int8()`	71.4%	68.0%	73.8%
Outlier Suppression	31.7%	54.1%	63.5%
SmoothQuant-O1	**71.2%**	68.3%	**73.7%**
SmoothQuant-O2	71.1%	**68.4%**	72.5%
SmoothQuant-O3	71.1%	67.4%	72.8%

- **Accelerate** inference and **halve the memory footprint**



FP16: 8 GPUs

SmoothQuant: 4 GPUs



Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. (2023, July). Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning (pp. 38087-38099). PMLR.

Scaling Up: 530B Model Within a Single Node

- Quantize MT-NLG 530B to W8A8 with negligible accuracy loss

	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%

- SmoothQuant can reduce the memory by half at a similar latency using **half the number** of GPUs, which allows serving the 530B model within a single node

SeqLen	Prec.	#GPUs	Latency	Memory
128	FP16	16	232ms	1040GB
	INT8	8	253ms	527GB
256	FP16	16	451ms	1054GB
	INT8	8	434ms	533GB
512	FP16	16	838ms	1068GB
	INT8	8	839ms	545GB
1024	FP16	16	1707ms	1095GB
	INT8	8	1689ms	570GB

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. (2023, July). Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning (pp. 38087-38099). PMLR.

SmoothQuant on LLaMA Families

- LLaMA and its variants 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

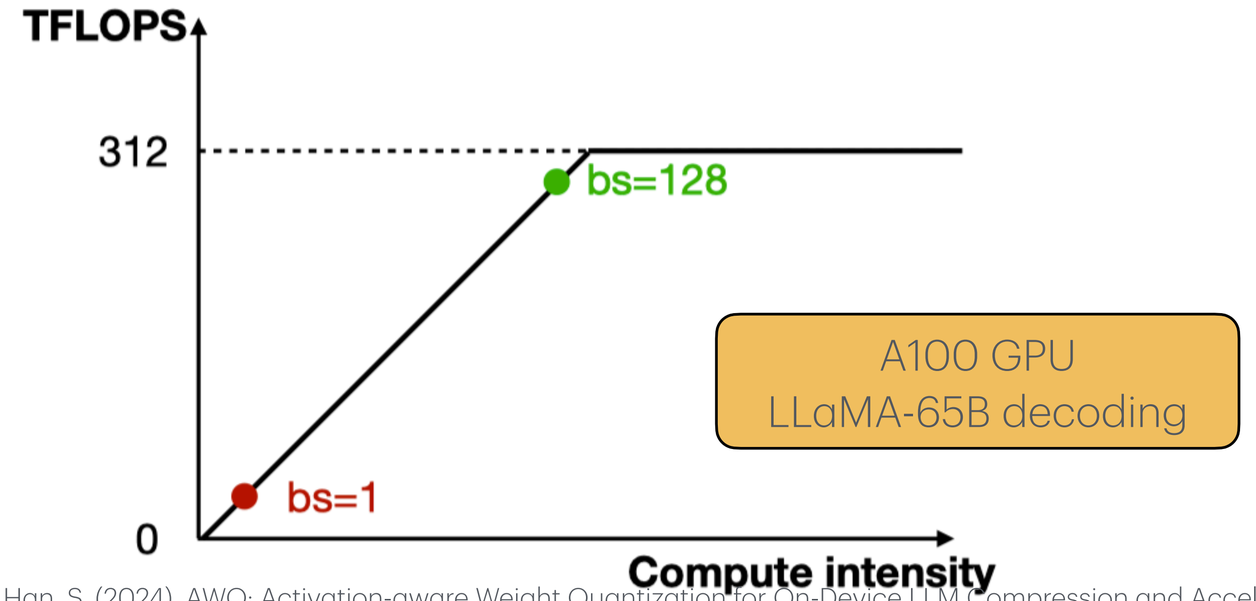
Wiki PPL↓	7B	13B	30B	65B
FP16	11.51	10.05	7.53	6.17
W8A8 SmoothQuant	11.56	10.08	7.56	6.20

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Lample, G. (2023). Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.

Is W8A8 Quantization Enough?

Edge LLMs are memory bounded and requires low-bit weight quantization

- W8A8 quantization is good for batch serving (e.g., batch size 128)
- But single-query LLM inferences (e.g., local) are still highly **memory-bounded**
- We need **low-bit weight-only** quantization (e.g., **W4A16**) for this setting

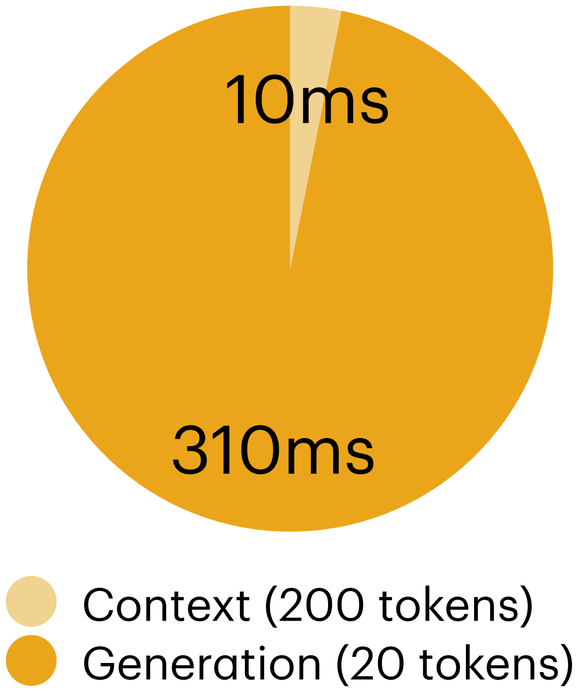


Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.

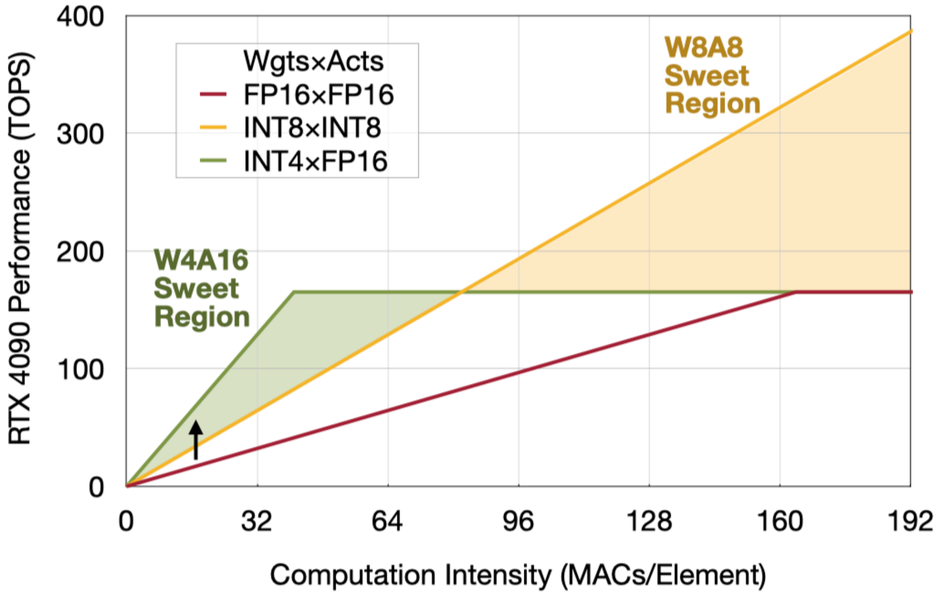
Low-bit Weight Quantization Brings Speedup

Weight loading is the efficiency bottleneck for edge LLM inference

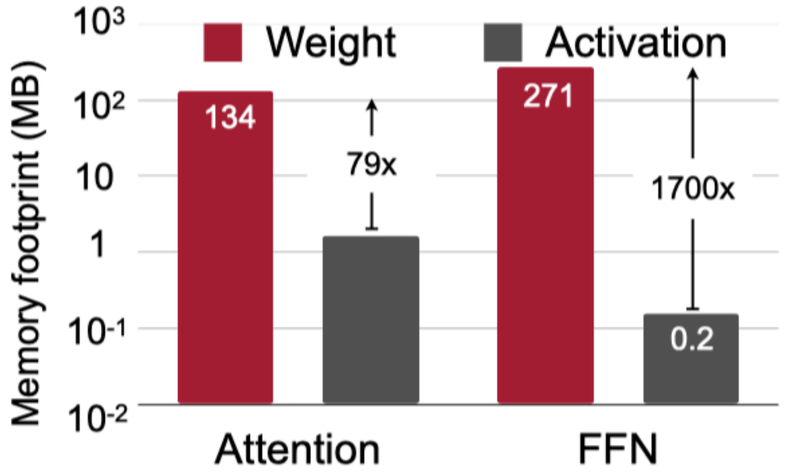
Generation stage is slower



Generation stage is bounded by memory bandwidth



Weight loading is more expensive



Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.

AWQ: Activation-aware Weight Quantization for Efficient Edge LLMs

Weight-only quantization

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.

AWQ: Activation-aware Weight Quantization

Targeting group-wise low-bit weight-only quantization (W4A16)

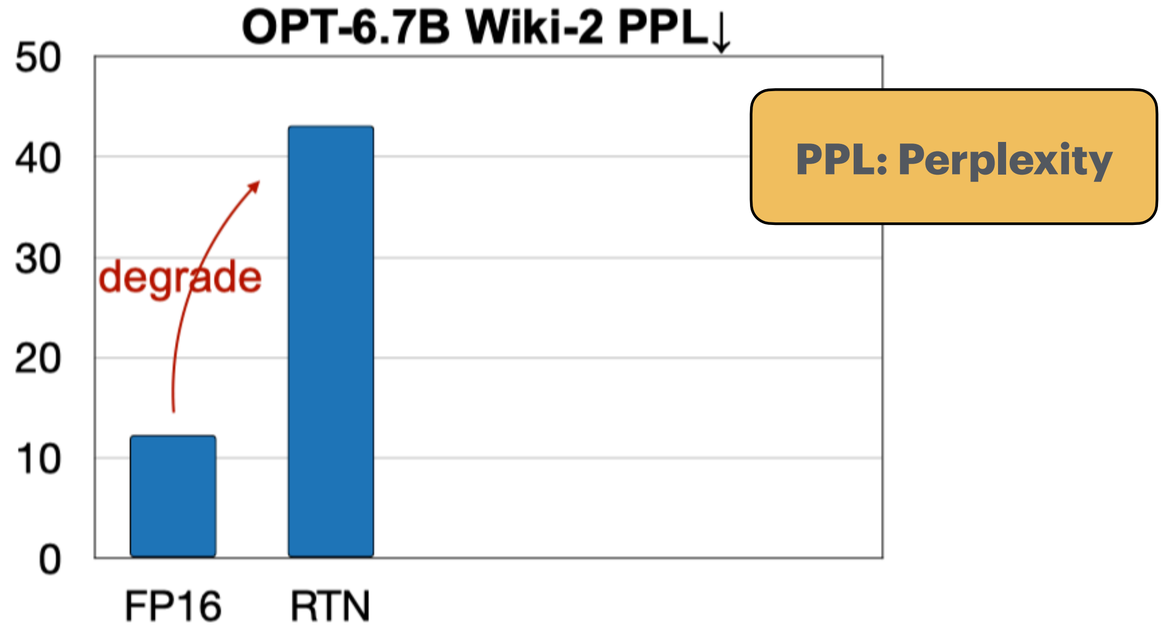
W **FP16**

1.2	-0.2	-2.4	-3.4
-2.5	-3.5	1.9	1.4
-0.9	1.6	-2.5	-1.9
-3.5	1.5	0.5	-0.1
1.8	-1.6	-3.2	-3.4
2.4	-3.5	-2.8	-3.9
0.1	-3.8	2.4	3.4
0.9	3.3	-1.9	-2.3

RTN →

Q(W) **INT3**

1	0	-2	-3
-3	-4	2	1
-1	2	-3	-2
-4	2	1	0
2	-2	-3	-3
2	-4	-3	-4
0	-4	2	3
1	3	-2	-2



- Weight-only quantization reduces the memory requirements and accelerates token generation by alleviating the memory bottleneck
- Group-wise/block-wise quantization (e.g., 64/128/256) offers a better accuracy-model size trade-off
- But there is still a performance gap with round-to-nearest (RTN) quantization (INT3-g128)

Perplexity (PPL)

A common metric used to evaluate the performance of language models

- PPL measures how well a language model predicts a given sequence of text.
- For a sequence of tokens $X = (x_1, x_2, \ldots, x_N)$, the PPL is defined as

$$PPL = e^{-\frac{1}{N}\sum_{i=1}^{N} \log_2 P(x_i|x_{<i})}$$

  - Where N is the total number of tokens in the sequence, $P(x_i \mid x_{<i})$ is the probability assigned by the model to the token x_i, given all previous tokens $x_{<i}$.
- A lower PPL means the model is better at predicting the correct tokens

AWQ: Activation-aware Weight Quantization

Observation: Weights are not equally important

W **FP16**

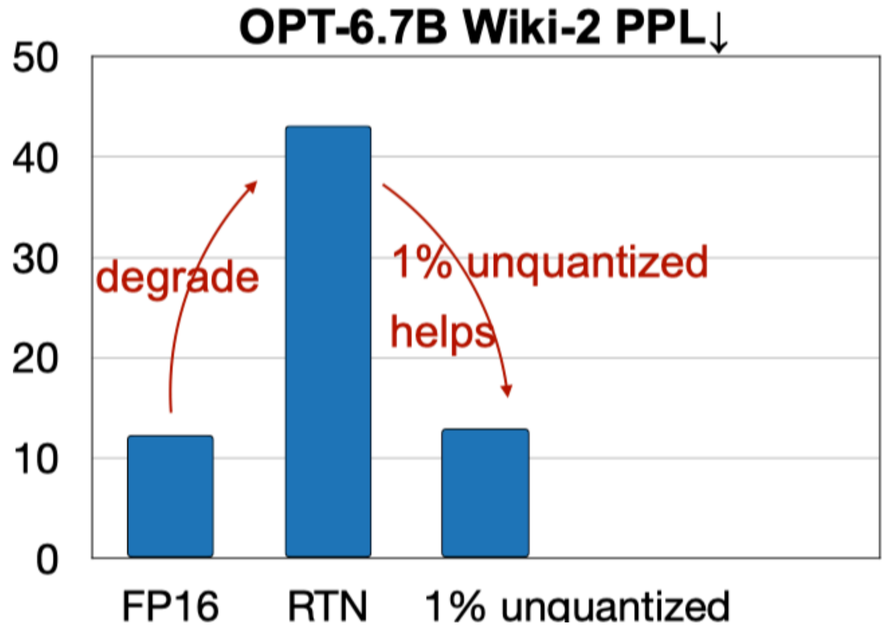
1.2	-0.2	-2.4	-3.4
-2.5	-3.5	1.9	1.4
-0.9	1.6	-2.5	-1.9
-3.5	1.5	0.5	-0.1
1.8	-1.6	-3.2	-3.4
2.4	-3.5	-2.8	-3.9
0.1	-3.8	2.4	3.4
0.9	3.3	-1.9	-2.3

RTN →

Q(W) **MixPrec**

1	0	-2	-3
-2.5	-3.5	1.9	1.4
-1	2	-3	-2
-4	2	1	0
2	-2	-3	-3
2	-4	-3	-4
0	-4	2	3
1	3	-2	-2

Unquantized channel



- Weights are not equally important, keeping **only 1%** of salient weight channels unquantized can greatly improve perplexity

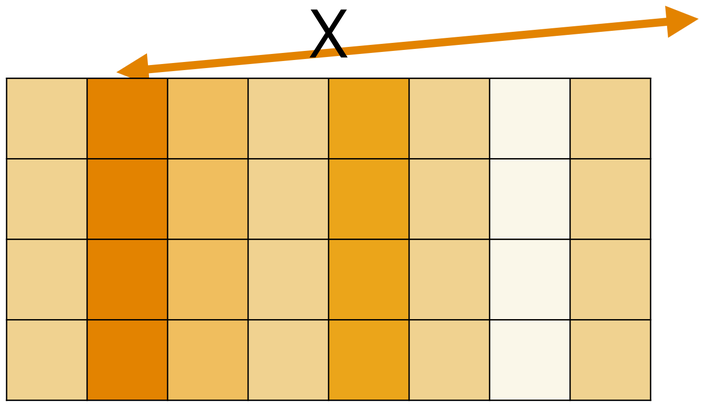
🤔 **How do we select salient channels? Based on weight magnitude?**

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.

AWQ: Activation-aware Weight Quantization

Salient weights are determined by activation distribution, not weight

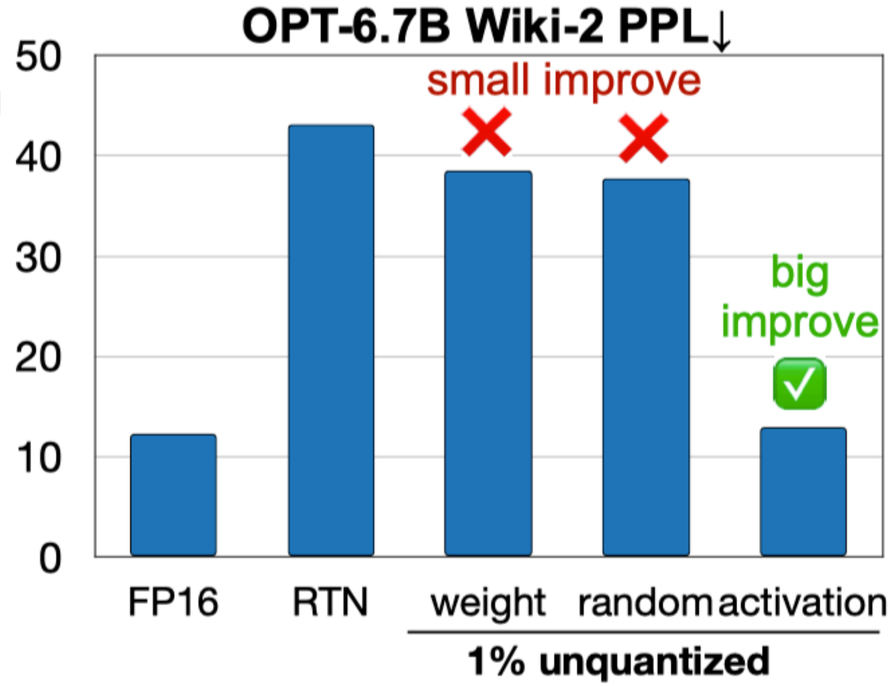
Activation-awareness



Q(W) **MixPrec**

1	0	-2	-3
-2.5	-3.5	1.9	1.4
-1	2	-3	-2
-4	2	1	0
2	-2	-3	-3
2	-4	-3	-4
0	-4	2	3
1	3	-2	-2

Unquantized channel



- How do we select salient channels? → **Activation** has outliers, we should look for activation distribution, not weight.

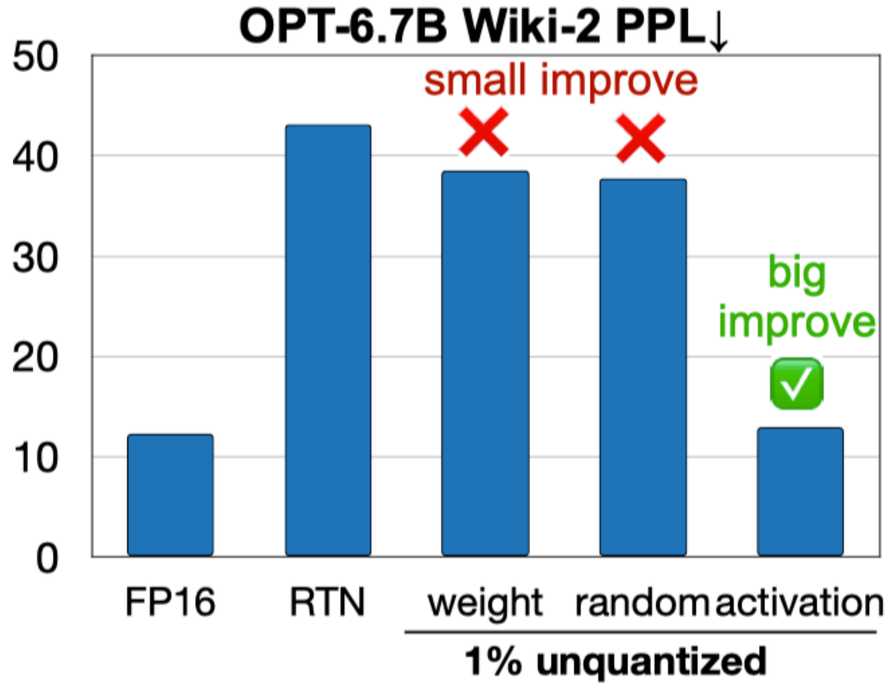
🤔 **Wait, is it really necessary to introduce mixed precision?**

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.

AWQ for Low-bit Weight-only Quantization

Protecting salient weights by scaling up (without mixed precision)

- Inherit the intuition from SmoothQuant:
  $Y = (X\text{diag}(s)^{-1}) \cdot (\text{diag}(s)W) = \hat{X}\hat{W}$
- In AWQ: $WX \xrightarrow{\text{Quantize}} Q(W \cdot s)(s^{-1} \cdot X)$
  - Let $Q(w) = \Delta \cdot \text{round}(\frac{W}{\Delta})$
    - Where Δ is the quantization scaler,
      absmax within the group, $\Delta = \frac{\max(|W|)}{2^{N-1}}$

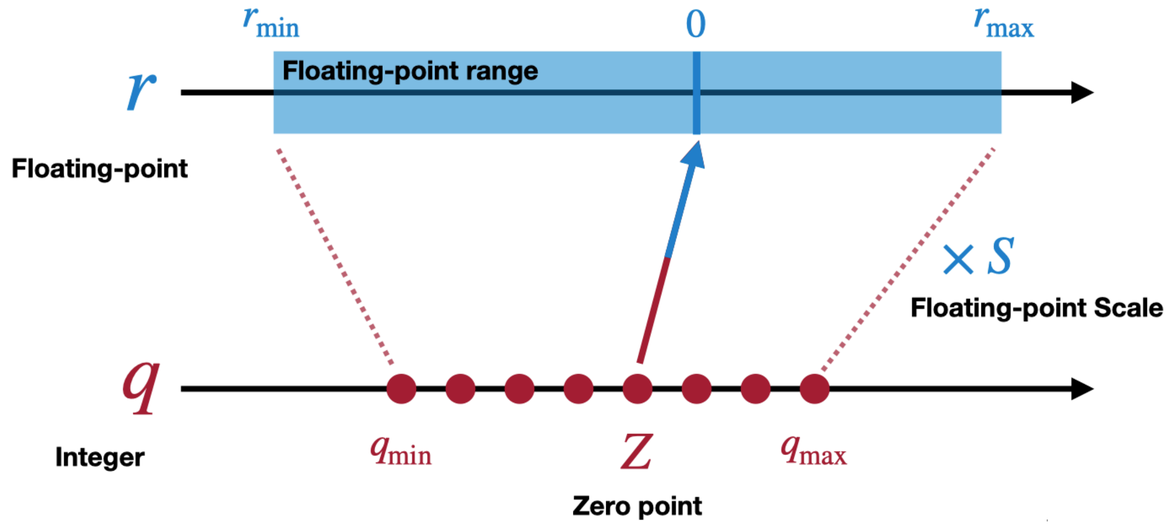


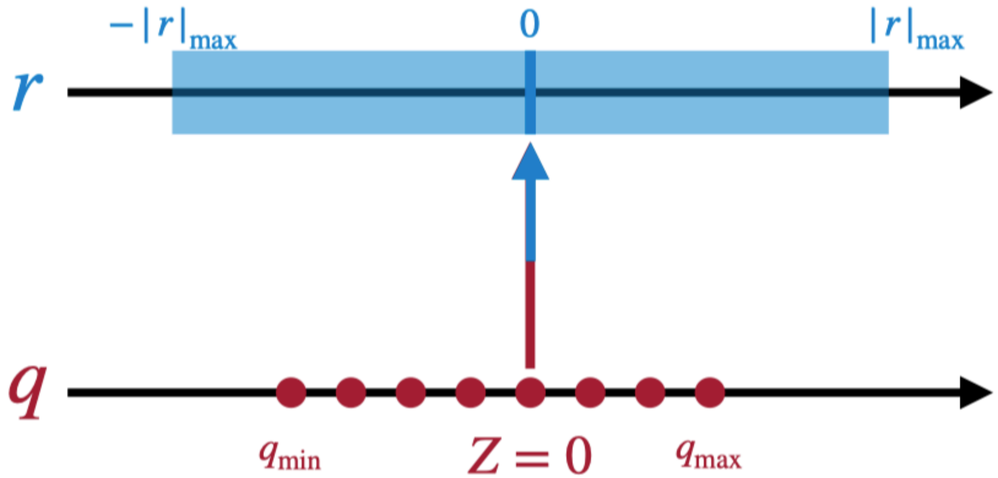
For the weight of salient channels, $\times s$ (the scaling factor), instead of storing as FP16

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.

Symmetric Linear Quantization

Zero point $Z = 0$ and Symmetric floating-point range





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

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

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

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

Why can s Protect the Salient Weight?

- Example: if one channel has two values: 0.5, 1.
  - $s = 1 : 0.5 \times s \approx 1{,}1 \times s = 1$
  - $s = 2 : 0.5 \times s = 1{,}1 \times s = 2$
- Intuitively, if scaling only one channel by 2 ($s = 2$), other channels remain the same → adding one more bit; if scaling by 4 → you are adding 2 more bits.

For the weight of salient channels, $\times s$ (the scaling factor), instead of storing as FP16

Why AWQ Reduces Quantization Error?

When scale s is not too large, quantization error is inversely proportional to s

Quantize

In AWQ: $WX \rightarrow Q(W \cdot s)(s^{-1} \cdot X)$

$$Q(W) = \Delta \cdot \text{round}(\frac{W}{\Delta}), \Delta = \frac{\max(|W|)}{2^{N-1}}$$

$$Q(w) \cdot x = \Delta \cdot \text{round}(\frac{w}{\Delta}) \cdot x$$

$$Q(w \cdot s) \cdot \frac{x}{s} = \Delta' \cdot \text{round}(\frac{w \cdot s}{\Delta'}) \cdot x \cdot \frac{1}{s}$$

- Assume weights are quantized to **signed INT4** numbers in this analysis
- $\Delta \approx \Delta'$, since scaling up a single channel will usually not change the global absmax

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.

Why AWQ Reduces Quantization Error?

When scale s is not too large, quantization error is inversely proportional to s

$\Delta \approx \Delta'$: Quantization scaler, Absmax within the group

$$Q(w) \cdot x = \Delta \cdot \text{round}(\frac{w}{\Delta}) \cdot x$$

$$Q(w \cdot s) \cdot \frac{x}{s} = \Delta' \cdot \text{round}(\frac{w \cdot s}{\Delta'}) \cdot x \cdot \frac{1}{s}$$

$$\rightarrow$$

$$\text{Err}(Q(w) \cdot x) = \Delta \cdot \text{Err}(\text{round}(\frac{w}{\Delta})) \cdot x$$

$$\text{Err}(Q(w \cdot s) \cdot \frac{x}{s}) = \Delta' \cdot \text{Err}(\text{round}(\frac{w \cdot s}{\Delta'})) \cdot x \cdot \frac{1}{s}$$

round(): 0-0.5. The error of the round() has an expectation of 0.25.

- Assume weights are quantized to signed INT4 numbers in this analysis
- As long as AWQ scale s is not too large (maintaining $\Delta \approx \Delta'$, since scaling up a single channel will usually not change the global absmax), the quantization error is inversely proportional to s

$s > 1$ as it's a scaling-up → reduce the quantization error

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.

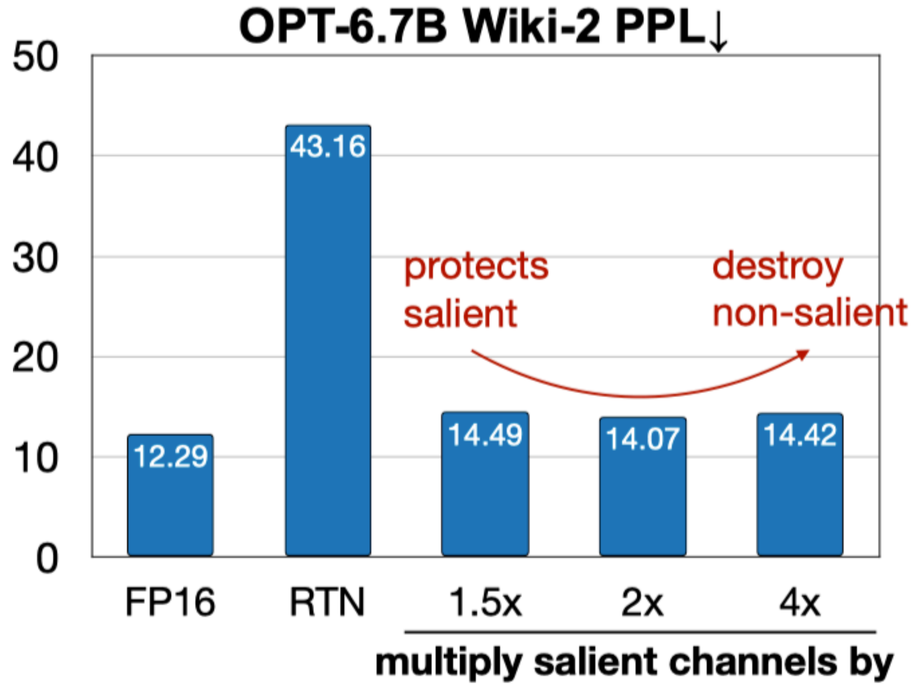
AWQ for Low-bit Weight-only Quantization

Protecting salient weights by scaling up (no mixed precision)

$$WX \rightarrow Q(W \cdot s)(s^{-1} \cdot X)$$

- Consider **activation-awareness** for salient channels
- Scaling up salient channels does not always bring about better performance

🤔 **How to determine s?**



Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.

AWQ for Low-bit Weight-only Quantization

Activation-aware optimal scaling searching

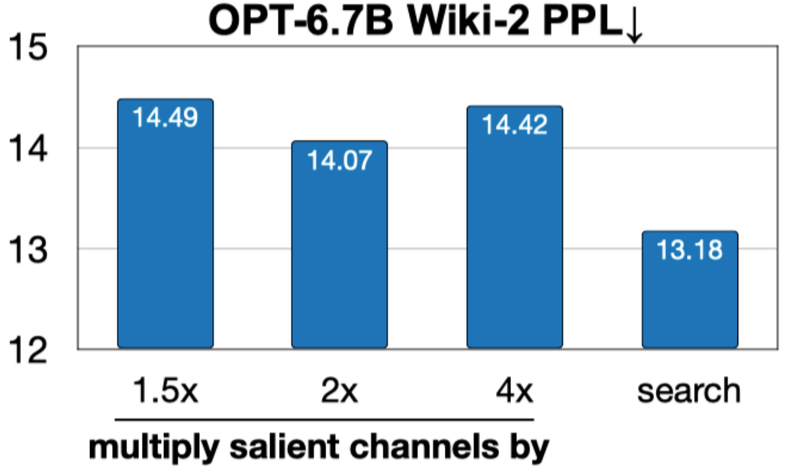
Quantize Calculate weights & Fuse into LN in Pre-norm (offline)

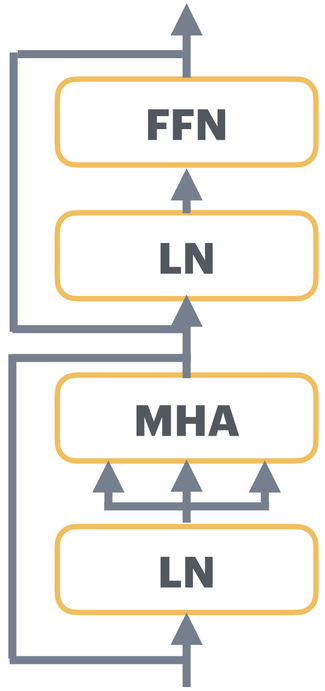
$$WX \rightarrow Q(W \cdot s)(s^{-1} \cdot X)$$

$$\mathscr{L}(s) = \left\| Q(W \cdot s)(s^{-1} \cdot X) - WX \right\|$$

$$s = s_X^{\alpha}, \alpha^* = \arg\min_{a} \mathscr{L}(s_X^{\alpha})$$

s_X^{α}: Average activation magnitude



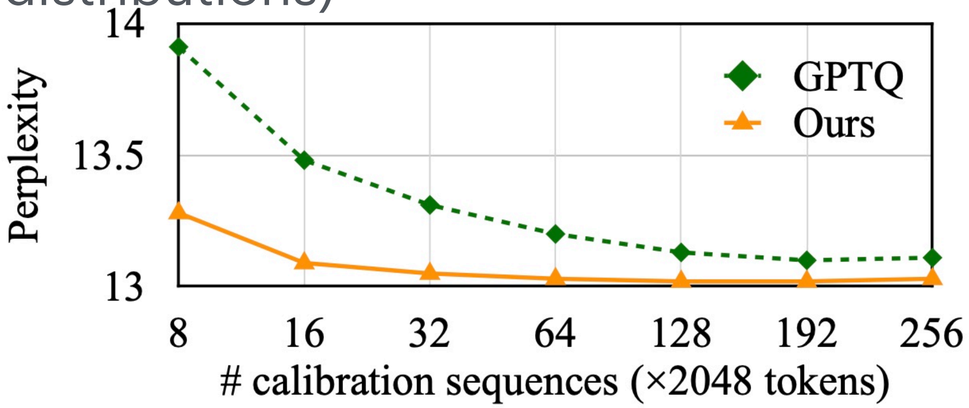


- Scale s is only determined by the **average activation magnitude** (activation awareness)
- α is a hyperparameter we can search from 0 to 1
- Take a data-driven approach with a fast **grid search**
  - The search objective is the mean square root error for the activation, not the weights themselves

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.

Advantages of AWQ

- Accurate, simple and easy to implement
- Hard hardware efficiency
- Less dependency on the calibration set compared to the regression-based method
  - Better data efficiency and distribution robustness
  - Generalize to instruction-tuned model and multi-modal language models (different distributions)



(a) Our method needs a smaller calibration set

Calib \ Eval	GPTQ PubMed	GPTQ Enron	Ours PubMed	Ours Enron
PubMed	32.48	50.41 +4.89	32.56	45.07 +0.50
Enron	+2.33 34.81	45.52	+0.60 33.16	44.57

(b) Our method is more robust to calibration set distribution

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.

AWQ Results

Improving general LLM quantization

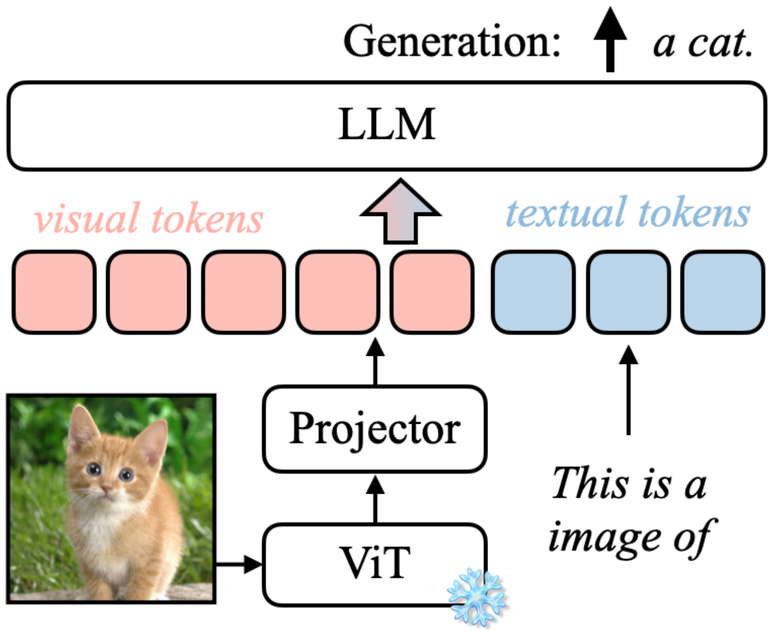
- Better PPL under low-bit weight-only quantization

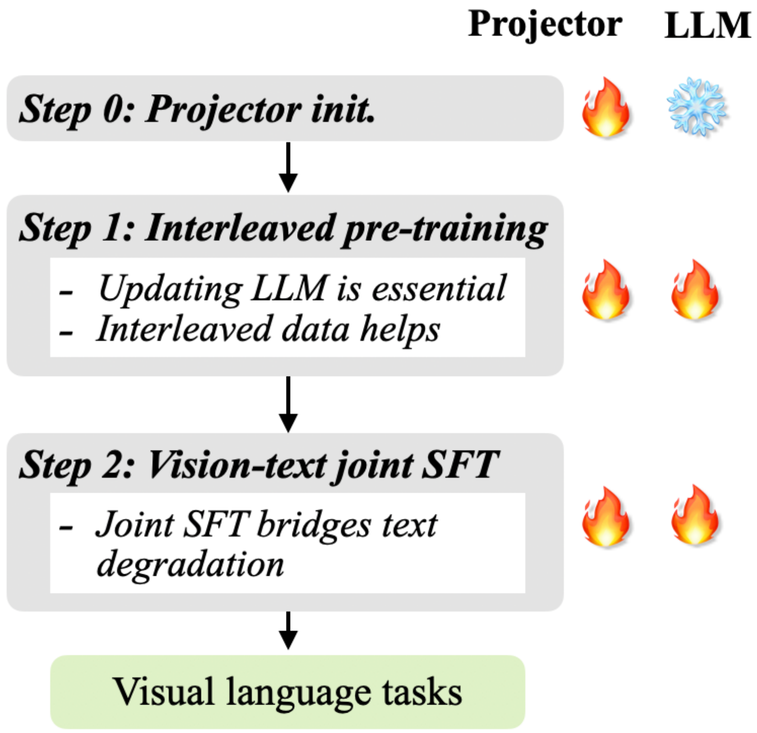
PPL↓		Llama-2			LLaMA			
		7B	13B	70B	7B	13B	30B	65B
FP16	-	5.47	4.88	3.32	5.68	5.09	4.10	3.53
INT3 g128	RTN	6.66	5.52	3.98	7.01	5.88	4.88	4.24
	GPTQ	6.43	5.48	3.88	8.81	5.66	4.88	4.17
	GPTQ-R	6.42	5.41	3.86	6.53	5.64	4.74	4.21
	AWQ	**6.24**	**5.32**	**3.74**	**6.35**	**5.52**	**4.61**	**3.95**
INT4 g128	RTN	5.73	4.98	3.46	5.96	5.25	4.23	3.67
	GPTQ	5.69	4.98	3.42	6.22	5.23	4.24	3.66
	GPTQ-R	5.63	4.99	3.43	5.83	5.20	4.22	3.66
	AWQ	**5.60**	**4.97**	**3.41**	**5.78**	**5.19**	**4.21**	**3.62**

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.

AWQ for Visual Language Model

- Goal: multi-modal LLM with excellent visual-language performance and text-only performance
- PaLM-E alike architecture due to its flexibility for multi-modal input/output
- Augment LLM with tokenized visual input





Lin, J., Yin, H., Ping, W., Molchanov, P., Shoeybi, M., & Han, S. (2024). Vila: On pre-training for visual language models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 26689-26699).

AWQ for Visual Language Model

AWQ well preserves the accuracy of GPT-4o-like multi-modal LLM

More demo

- Q: Elaborate on the visual and narrative elements of the video in detail

- A: The video shows a person's hands working on a white surface. They are folding a piece of fabric with a checkered pattern in shades of blue and white. The fabric is being folded into a smaller, more compact shape. The person's fingernails are painted red, and they are wearing a black and red garment. There are also a ruler and a pencil on the surface, suggesting that measurements and precision are involved in the process.

Lin, J., Yin, H., Ping, W., Molchanov, P., Shoeybi, M., & Han, S. (2024). Vila: On pre-training for visual language models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 26689-26699).

AWQ Results for Visual Langauge Models

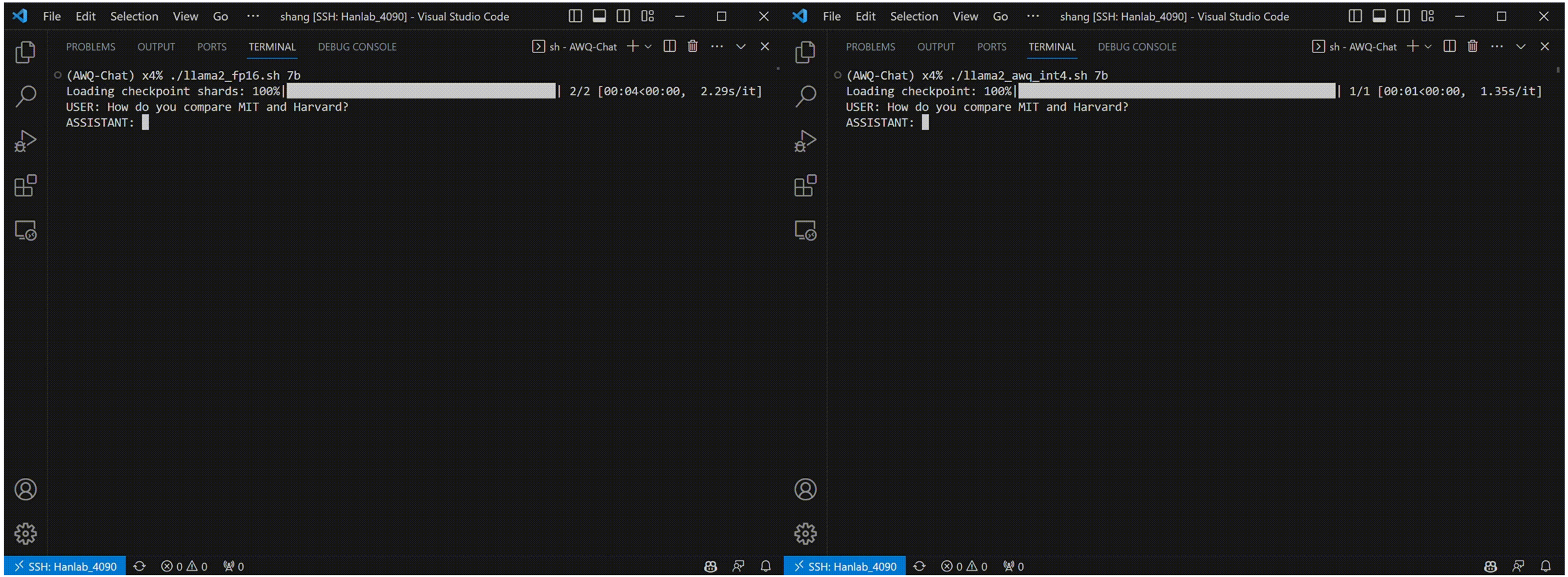
Quantization of multi-model Lms (VILA)

Model	Prec.	VQAv2	GQA	VizWiz	SQA-I	VQA-T	POPE	MME	MMB	MMB-CN	SEED	llava-bench	MMVet	Average
VILA-7B	FP16	80.3	63.1	59.6	68.0	62.6	86.3	1489.4	69.8	61.0	61.7	75.2	35.1	**65.7**
VILA-7B-AWQ	**INT4**	80.1	63.0	57.8	68.0	61.9	85.3	1486.3	68.8	59.0	61.3	75.8	35.9	**65.2**
VILA-13B	FP16	80.5	63.6	63.1	70.5	64.0	86.3	1553.6	73.8	66.7	62.8	78.3	42.6	**68.4**
VILA-13B-AWQ	**INT4**	80.4	63.6	63.0	71.2	63.5	87.0	1552.9	73.6	66.3	62.2	77.6	42.0	**68.2**

Lin, J., Yin, H., Ping, W., Molchanov, P., Shoeybi, M., & Han, S. (2024). Vila: On pre-training for visual language models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 26689-26699).

TinyChat

A lightweight LLM inference engine



Pruning and Sparsity

Weight Sparsity: Wanda

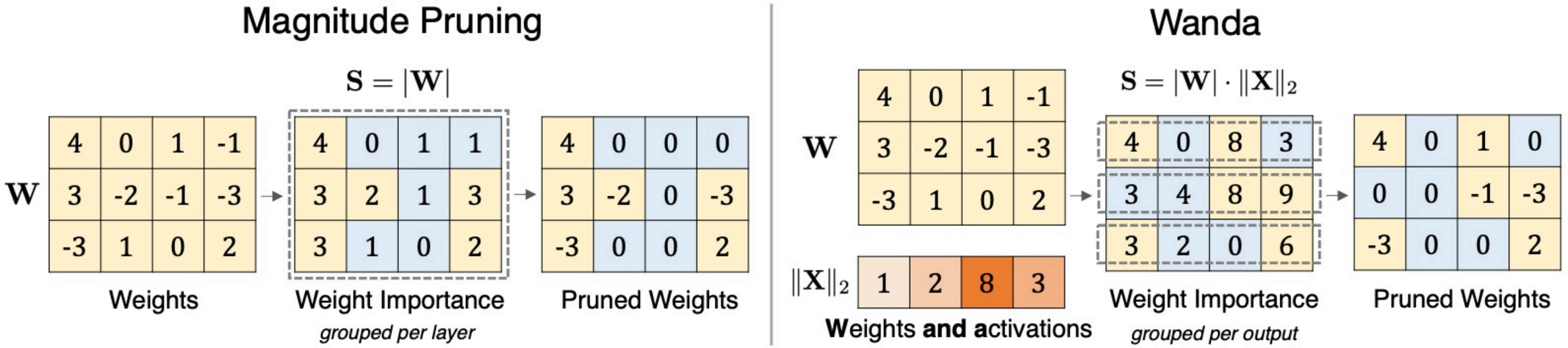
Contextual Sparsity: DejaVu, MoE

Attention Sparsity: SpAtten, H2O

Wanda: Pruning by Considering Weights and Activations

Weight sparsity

- Similar idea to AWQ: consider **activation distribution** when pruning weights
- Pruning criteria: |weight| * ||activation||



Sun, M., Liu, Z., Bair, A., & Kolter, J. Z. (2023). A simple and effective pruning approach for large language models. arXiv preprint arXiv:2306.11695.

Wanda: Pruning by Considering Weights and Activations

Weight sparsity

- Similar idea to AWQ: consider **activation distribution** when pruning weights
- Pruning criteria: |weight| * ||activation||
- Consistently outperforms magnitude-based pruning

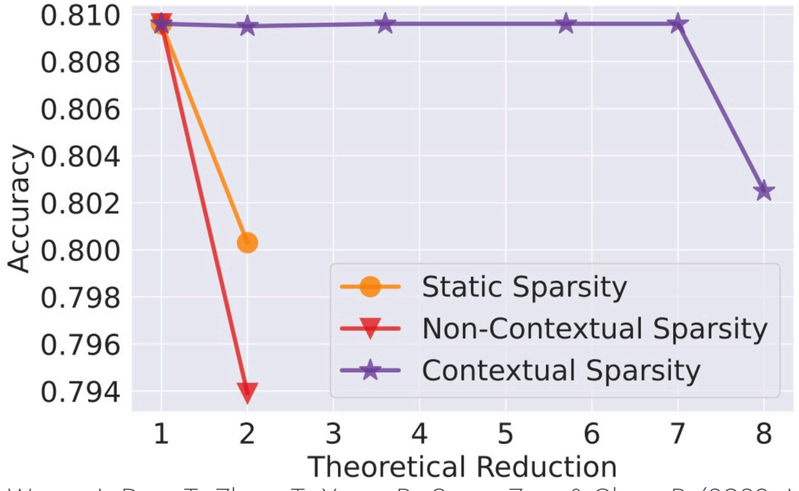
			LLaMA				LLaMA-2		
Method	Weight Update	Sparsity	7B	13B	30B	65B	7B	13B	70B
Dense	-	0%	59.99	62.59	65.38	66.97	59.71	63.03	67.08
Magnitude	✗	50%	46.94	47.61	53.83	62.74	51.14	52.85	60.93
SparseGPT	✓	50%	**54.94**	58.61	63.09	66.30	**56.24**	60.72	**67.28**
Wanda	✗	50%	54.21	**59.33**	**63.60**	**66.67**	**56.24**	**60.83**	67.03
Magnitude	✗	4:8	46.03	50.53	53.53	62.17	50.64	52.81	60.28
SparseGPT	✓	4:8	**52.80**	55.99	60.79	64.87	**53.80**	**59.15**	65.84
Wanda	✗	4:8	52.76	**56.09**	**61.00**	**64.97**	52.49	58.75	**66.06**
Magnitude	✗	2:4	44.73	48.00	53.16	61.28	45.58	49.89	59.95
SparseGPT	✓	2:4	**50.60**	**53.22**	58.91	62.57	**50.94**	54.86	63.89
Wanda	✗	2:4	48.53	52.30	**59.21**	**62.84**	48.75	**55.03**	**64.14**

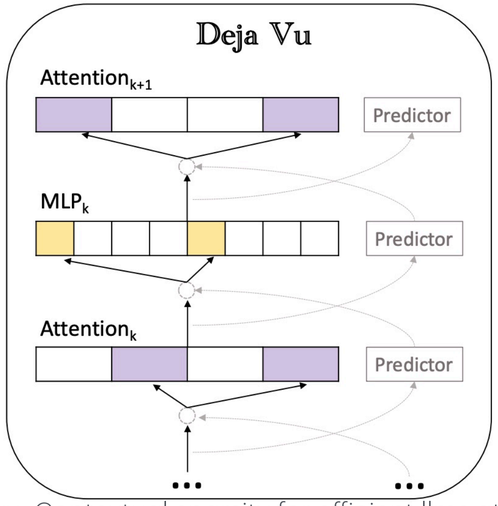
Sun, M., Liu, Z., Bair, A., & Kolter, J. Z. (2023). A simple and effective pruning approach for large language models. arXiv preprint arXiv:2306.11695.

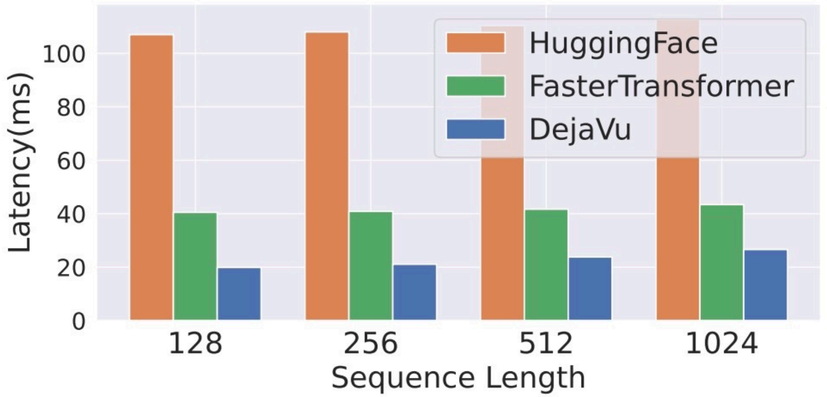
DejaVu: Input Dependent Sparsity

Contextual sparsity

- Static sparsity: hurts the accuracy with a medium-high sparsity
- Contextual sparsity: small, input-dependent sets of redundant heads and features
- Context sparsity exists and can be **predicted** using an asynchronous predictor head
- Accelerate inference without hurting model quality





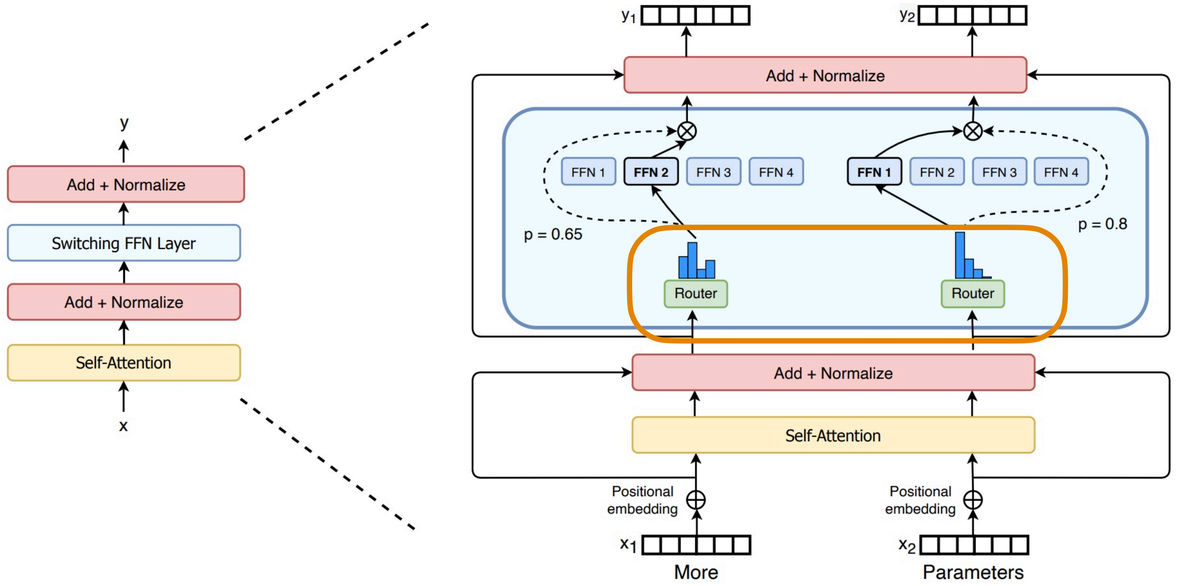


Liu, Z., Wang, J., Dao, T., Zhou, T., Yuan, B., Song, Z., ... & Chen, B. (2023, July). Deja vu: Contextual sparsity for efficient llms at inference time. In International Conference on Machine Learning (pp. 22137-22176). PMLR.

Mixture-of-Experts (MoE)

Contextual sparsity: sparsely activated for each token

- MoE allows sparse use of part of the parameters for each token during inference
- Increase the **total** amount of parameters without increasing inference costs **per token**
- A router will distribute the workload among different experts



Fedus, W., Zoph, B., & Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. Journal of Machine Learning Research, 23(120), 1-39.

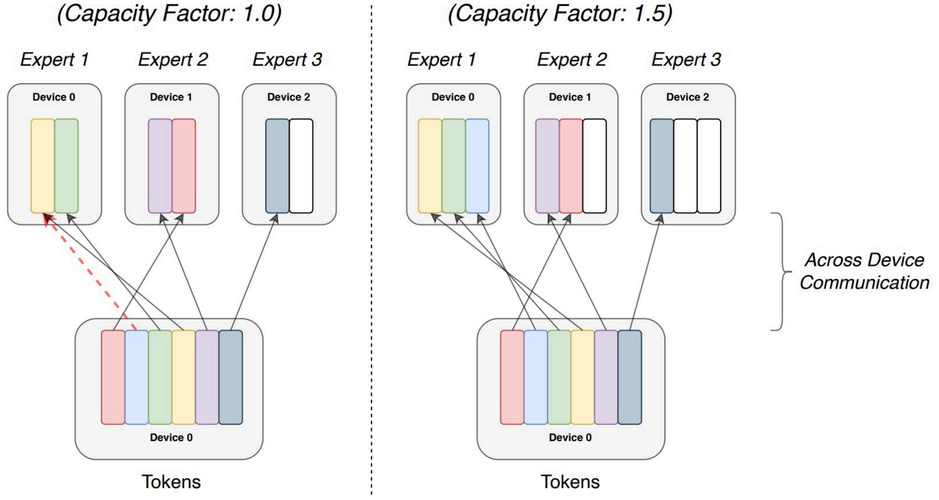
Mixture-of-Experts (MoE)

Contextual sparsity: sparsely activated for each token

- A key component of MoE is the **routing function**
- Expert Capacity $= (\frac{\text{tokens per batch}}{\text{number of experts}}) \times \text{capacity factor}(C)$
- $C = 1$, every expert can process at most $6/3 * 1 = 2$ tokens; one token is skipped
- $C = 1.5$, every expert can process at most $6/3 * 1.5 = 3$ tokens; slack capacity for expert 2&3

Terminology

- **Experts:** Split across devices, each having their own unique parameters. Perform standard feed-forward computation.
- **Expert Capacity:** Batch size of each expert. Calculated as (tokens_per_batch / num_experts) * capacity_factor
- **Capacity Factor:** Used when calculating expert capacity. Expert capacity allows more buffer to help mitigate token overflow during routing.

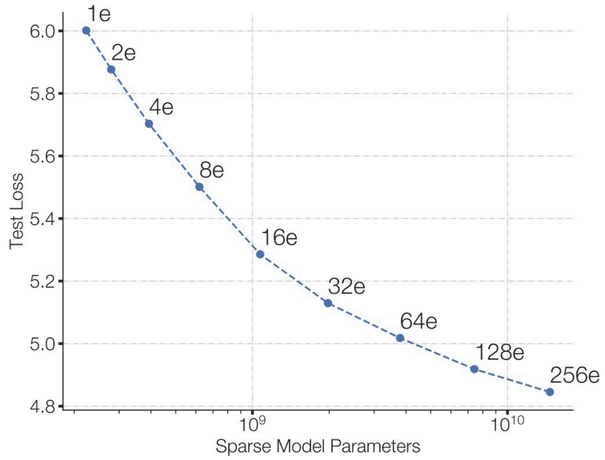


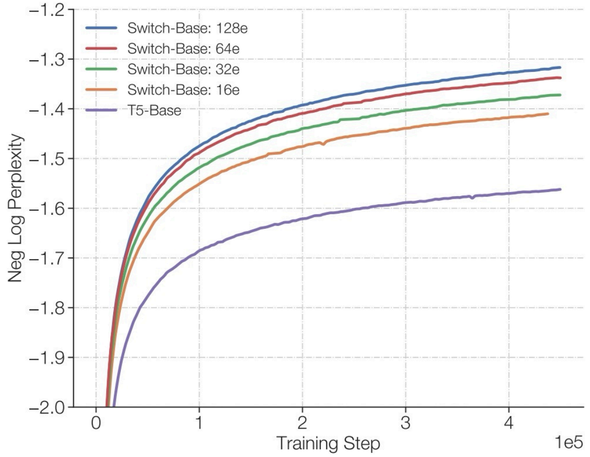
Fedus, W., Zoph, B., & Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. Journal of Machine Learning Research, 23(120), 1-39.

Mixture-of-Experts (MoE)

Contextual sparsity: sparsely activated for each token

- MoE allows sparsely use part of the parameters for each token during inference
- Increase the **total** amount of parameters without increasing inference costs **per token**
- A router will distribute the workload among different experts
- More experts → larger total model size → lower loss/better perplexity



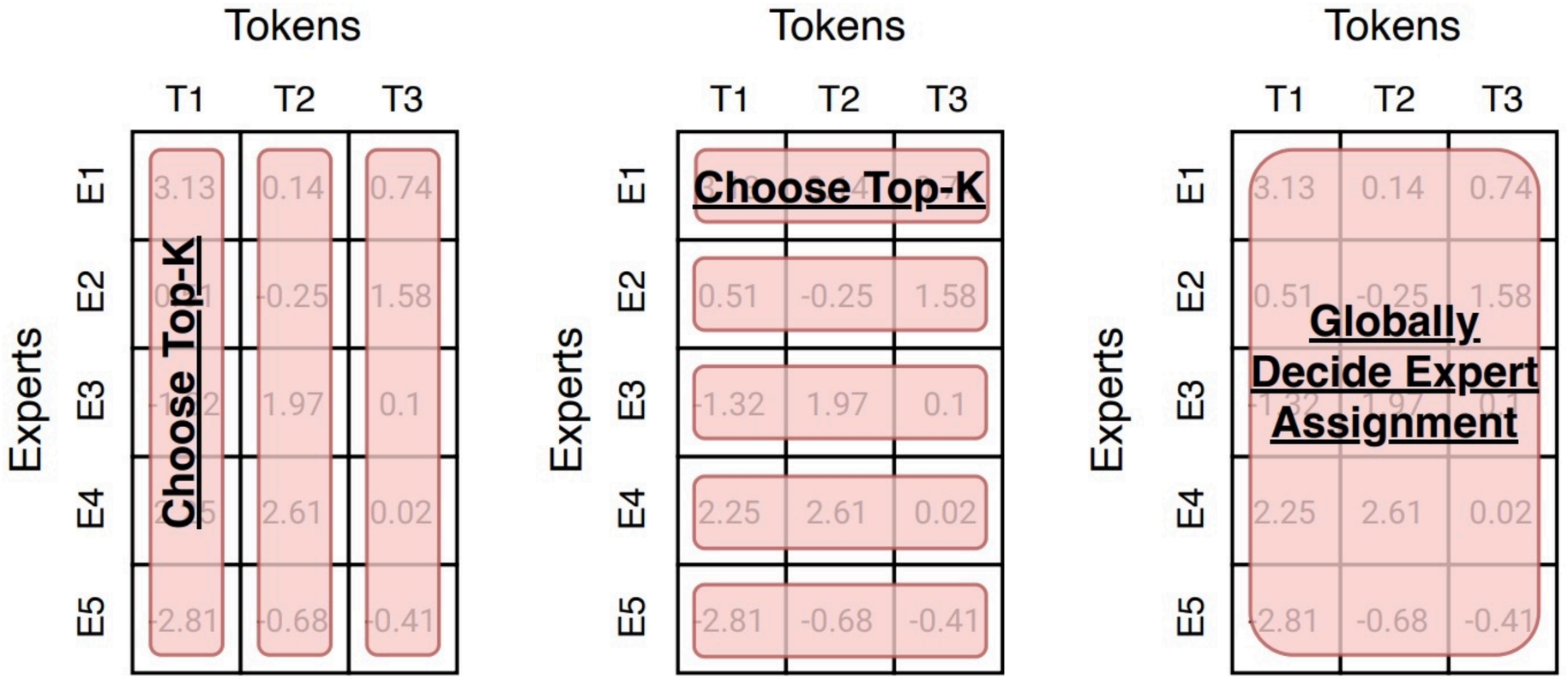


Fedus, W., Zoph, B., & Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. Journal of Machine Learning Research, 23(120), 1-39.

Mixture-of-Experts (MoE)

Contextual sparsity: sparsely activated for each token

- Different routing mechanisms

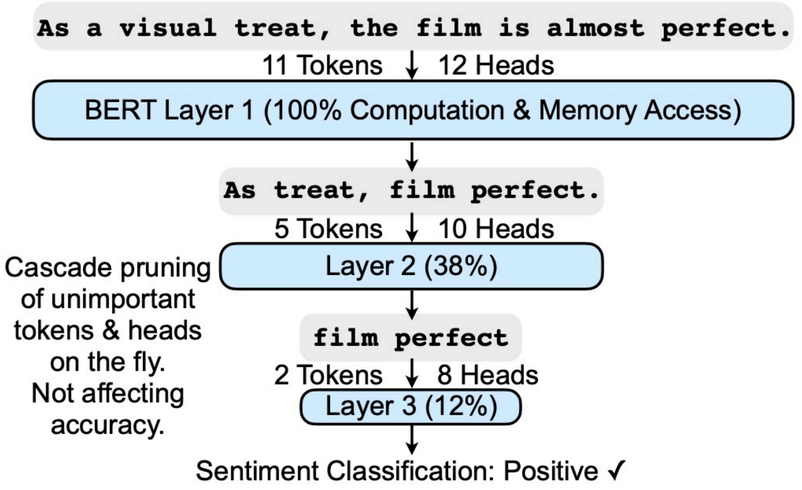


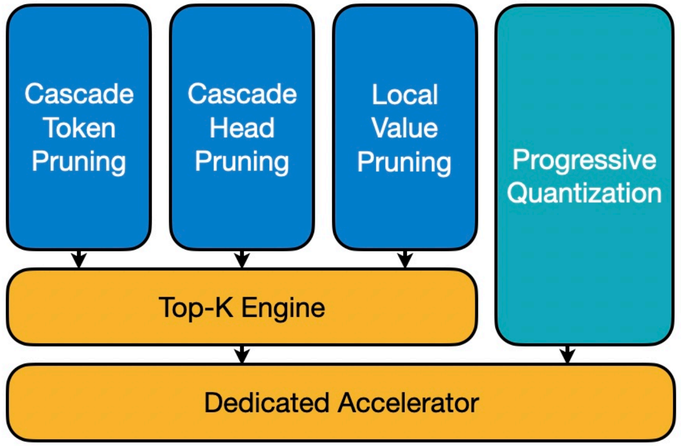
Fedus, W., Dean, J., & Zoph, B. (2022). A review of sparse expert models in deep learning. arXiv preprint arXiv:2209.01667.

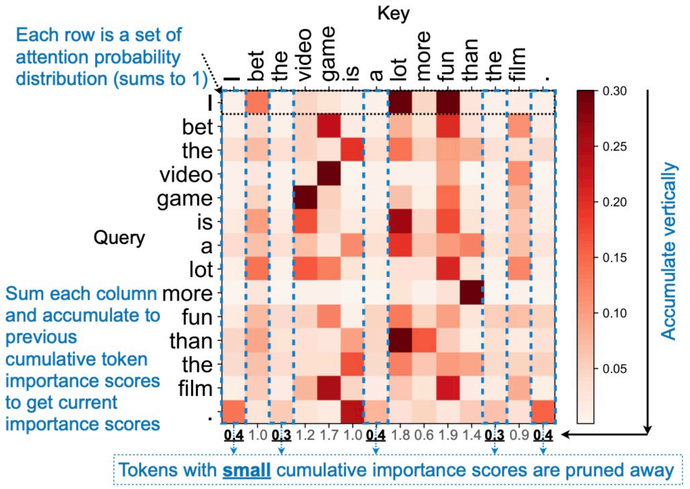
SpAtten: Token Pruning & Head Pruning

Attention sparsity

- Cascade pruning of unimportant tokens and heads
- Tokens with a small cumulative attention are pruned away
- V pruning: don't fetch V if QK is small
- Progressive quantization: low precision first, if not confident → high precision





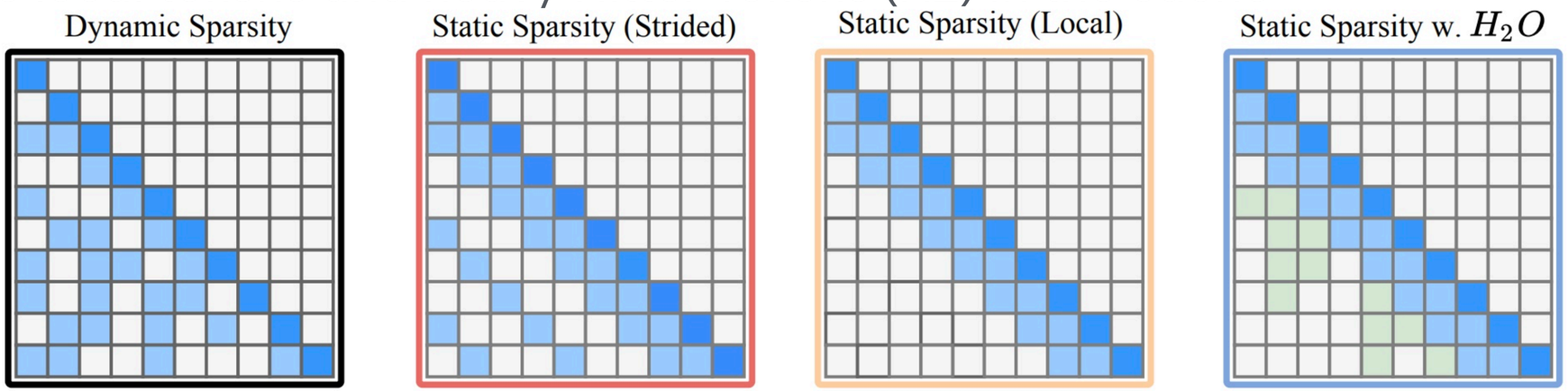


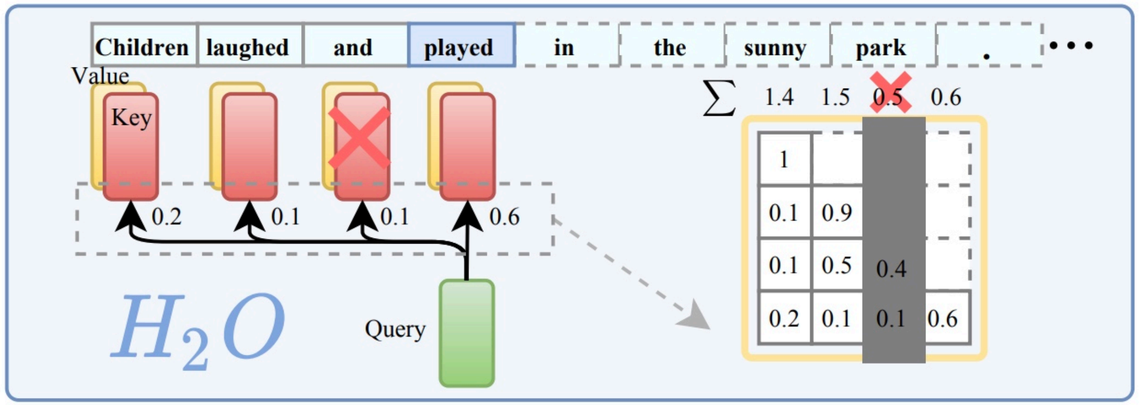
Wang, H., Zhang, Z., & Han, S. (2021, February). Spatten: Efficient sparse attention architecture with cascade token and head pruning. In 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 97-110). IEEE.

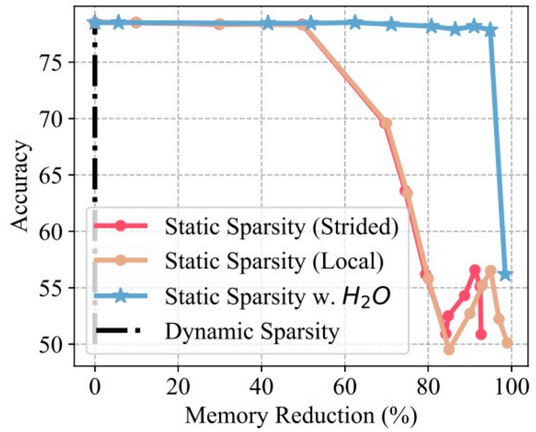
H2O: Token Pruning in KV Cache

Attention sparsity

- Kepp the local tokens and heavy Hitter Tokens (H2) in the cache







Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., ... & Chen, B. (2023). H2o: Heavy-hitter oracle for efficient generative inference of large language models. Advances in Neural Information Processing Systems, 36, 34661-34710.

LLM Serving Systems

Important Metrics for LLM Serving

- **Time to First Token (TTFT)**: measure how quickly users begin to see model output after submitting a query
  - Crucial for real-time inference
  - Driven by prompt processing time and the generation of the first token
- **Time Per Output Token (TPOT)**: time taken to generate each output token
  - Impact user perception of speed (e.g., 100ms/token = 10 tokens/second, ~450 words/minute)
- **Latency = (TTFT) + (TPOT * the number of tokens to be generated)**
  - Total time to generate the complete response
- **Throughput**: number of tokens generated per second across all requests by the inference server

LLM Inference Performance Engineering: Best Practices

Optimizing LLM Serving: Goals and Tradeoffs

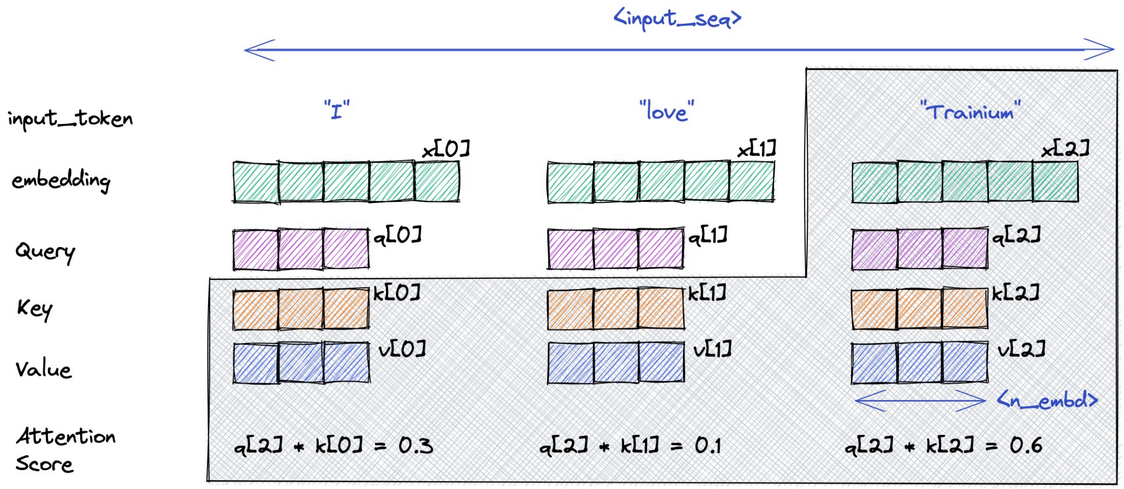
- **Optimization Goals & Tradeoffs**
  - **Goal**: Minimize TTFT, maximize throughput, and reduce TPOT
  - **Throughput vs. TPOT Tradeoff**: Processing multiple queries concurrently increases throughput but extends TPOT for each user
- **Key Heuristics for Model Evaluation**
  - **Output Length**: dominates latency
  - **Input Length**: minimal impact on performance but significant for hardware
  - **Model Size:** Larger models have higher latency, but are not proportional to their size
    - Example" Llama-70B is 2x Llama-13B

LLM Inference Performance Engineering: Best Practices

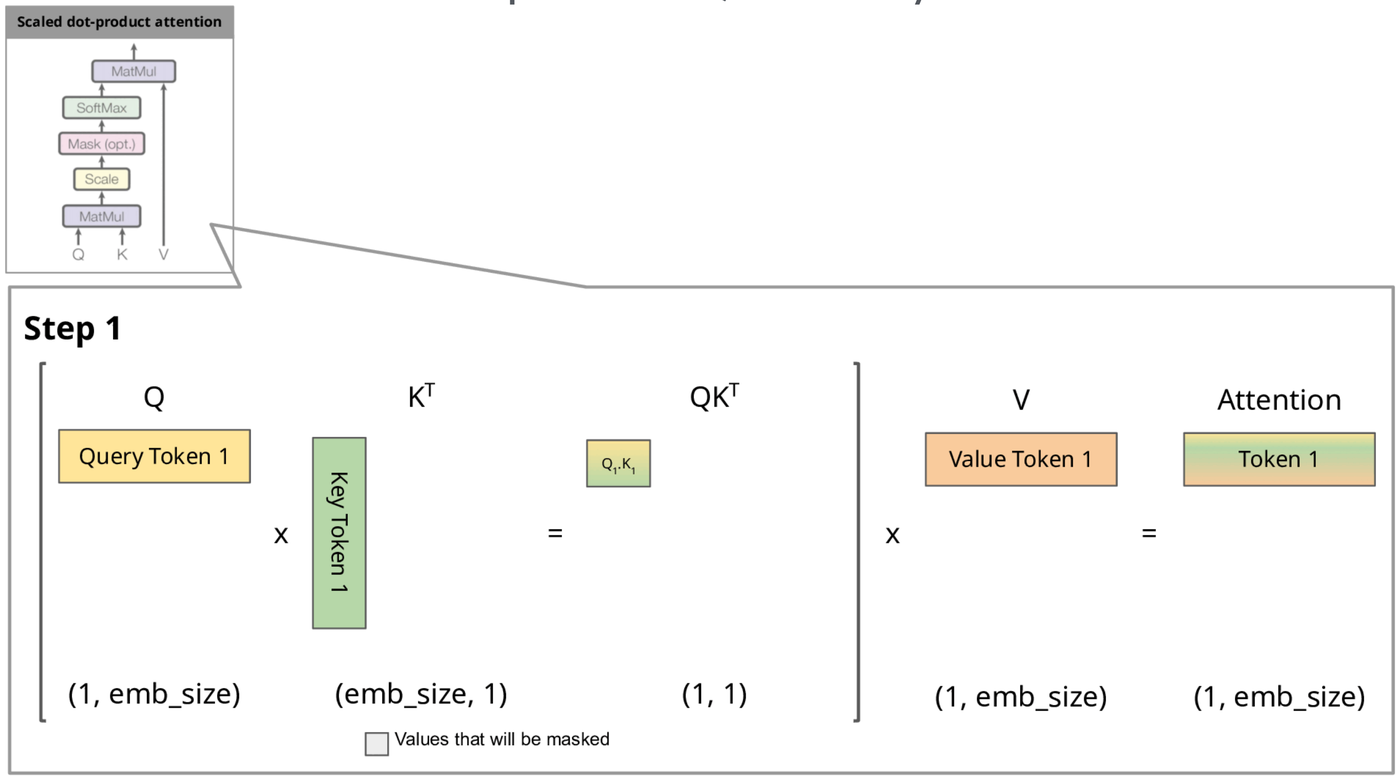
KV Cache Optimizations

The KV cache could be larger with long context

- During Transformder decoding (GPT-style), we need to store the **K**eys and **V**alues of **all previous** tokens so that we can perform the attention computation, namely the **KV cache**
  - Only need the **current** query token



Source

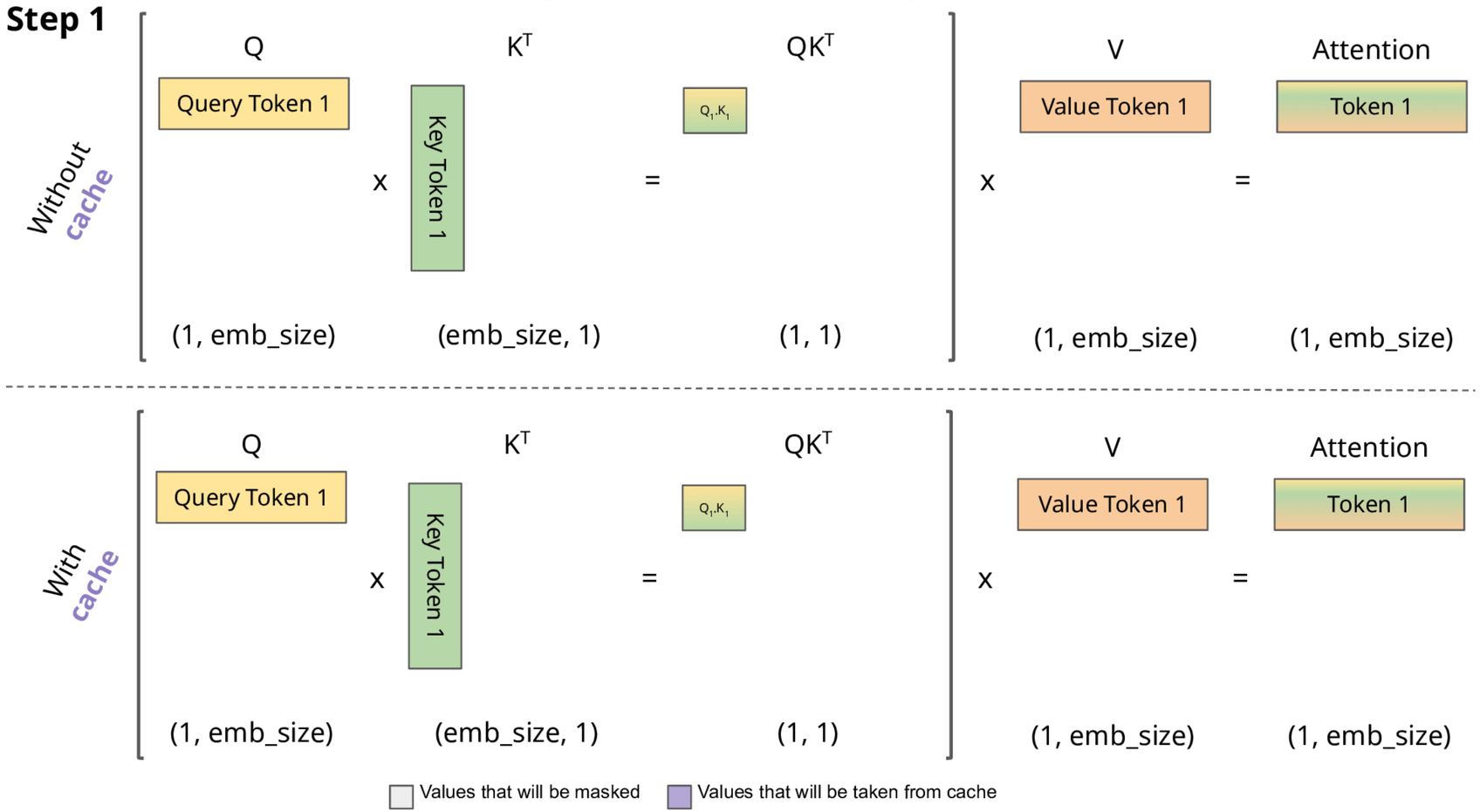


Zoom-in! (simplified without Scale and Softmax)

Step-by-step visualization of the scaled dot-product attention in the decoder

KV Cache Optimizations

The KV cache could be larger with long context



Comparison of scaled dot-product attention with and without KV caching.

KV Cache Optimizations

The KV cache could be larger with long context

- We can calculate the memory required to store the KV cache

- Llama-2-7B, KV cache size =

$$\underbrace{BS}_{minibatch} * \underbrace{32}_{layers} * \underbrace{32}_{heads} * \underbrace{128}_{n_{emd}} * \underbrace{N}_{length} * \underbrace{2}_{K\&V} * \underbrace{2 \text{ bytes}}_{fp16} = 512 \text{ KB} \times BS \times N$$

- Llama-2-13B, KV cache size =

$$\underbrace{BS}_{minibatch} * \underbrace{40}_{layers} * \underbrace{40}_{heads} * \underbrace{128}_{n_{emd}} * \underbrace{N}_{length} * \underbrace{2}_{K\&V} * \underbrace{2 \text{ bytes}}_{fp16} = 800 \text{ KB} \times BS \times N$$

- Llama-2-70B (if using MHA), KV cache size =

$$\underbrace{BS}_{minibatch} * \underbrace{80}_{layers} * \underbrace{64}_{heads} * \underbrace{128}_{n_{emd}} * \underbrace{N}_{length} * \underbrace{2}_{K\&V} * \underbrace{2 \text{ bytes}}_{fp16} = 2.5 \text{ MB} \times BS \times N$$

LLM Inference Performance Engineering: Best Practices

KV Cache Optimizations

The KV cache could be larger with long context

- We can calculate the memory required to store the KV cache

- Llama-2-70B (if using MHA), KV cache size =

$$\underbrace{BS}_{minibatch} * \underbrace{80}_{layers} * \underbrace{64}_{heads} * \underbrace{128}_{n_{emd}} * \underbrace{N}_{length} * \underbrace{2}_{K\&V} * \underbrace{2 \text{ bytes}}_{fp16} = 2.5 \text{ MB} \times BS \times N$$

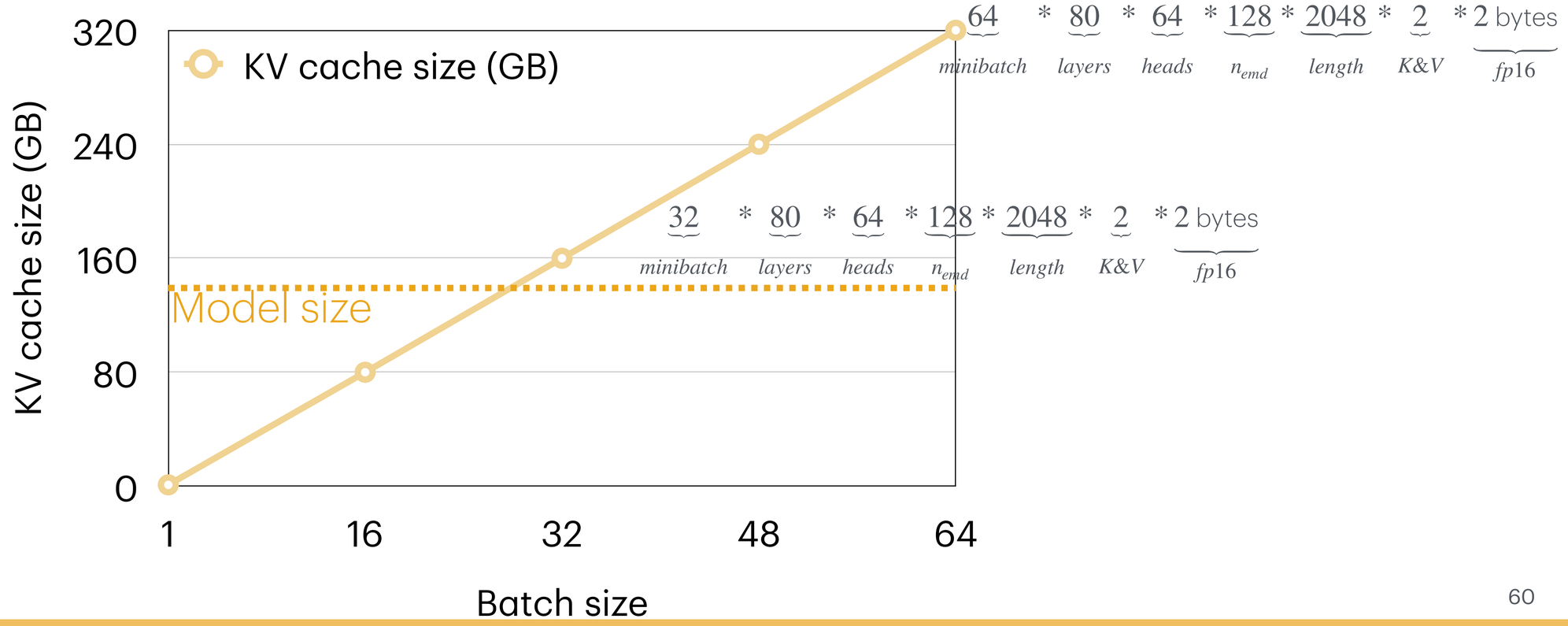
  - $BS = 1, N = 512 : 1.25GB$
  - $BS = 1, N = 4096 : 10GB$ (~ a paper)
  - $BS = 16, N = 4096 : 160GB$ (required two A100 GPUs)

LLM Inference Performance Engineering: Best Practices

KV Cache Optimizations

The KV cache could be larger with long context

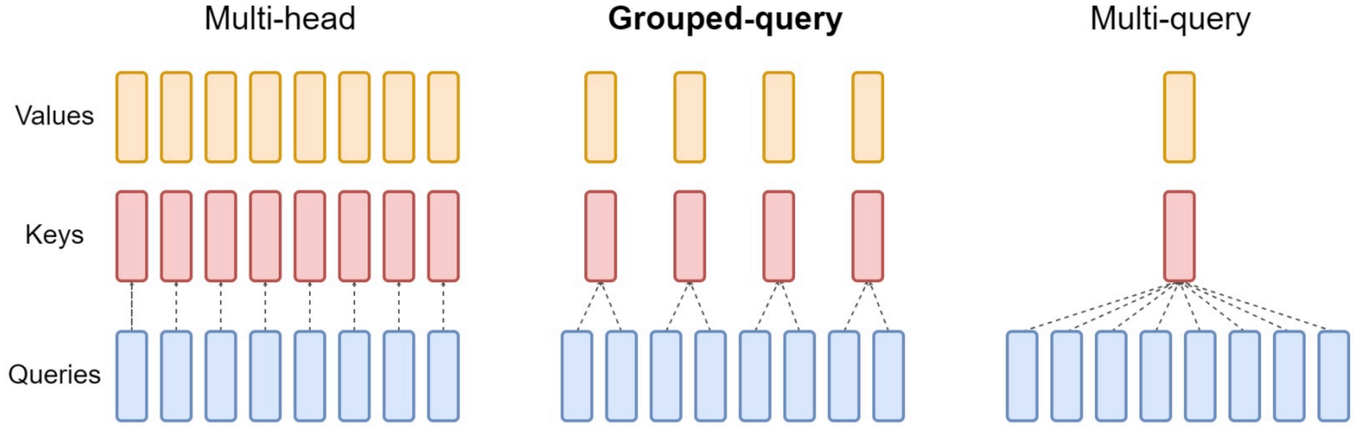
- Now, we calculate the KV cache size under $N = 2048$ different batch sizes
  - The KV cache size goes quickly larger than the model weights



Multi-Query Attention

Reduce the KV cache memory usage with MQA/GQA

- Reducing the KV cache size by reducing the number of KV-heads
  - Multi-head attention (MHA): N heads for query, N heads for key/value
  - Multi-query attention (MQA): N heads for query, 1 head for key/value
  - Grouped-query attention (GQA): N heads for query, G heads for key/value *typically $G = N/8$)

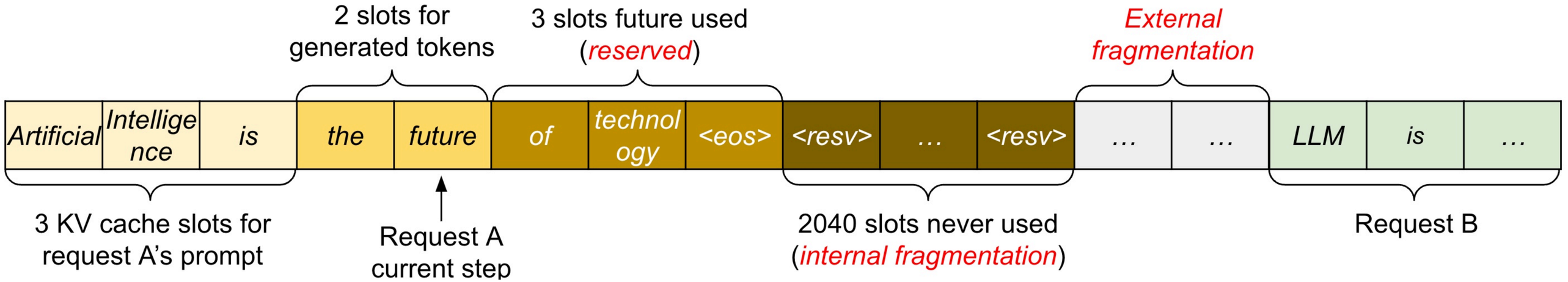


Shazeer, N. (2019). Fast transformer decoding: One write-head is all you need. arXiv preprint arXiv:1911.02150.
Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., & Sanghai, S. (2023). Gqa: Training generalized multi-query transformer models from multi-head checkpoints. arXiv preprint arXiv:2305.13245.

vLLM and Paged Attention

High-throughput and memory-efficient serving

- Analyzing the waste in KV Cache usage
  - Internal fragmentation: over-allocated due to the unknown output length
  - Reservation: not used at the current step, but used in the future
  - External fragmentation: due to different sequence lengths

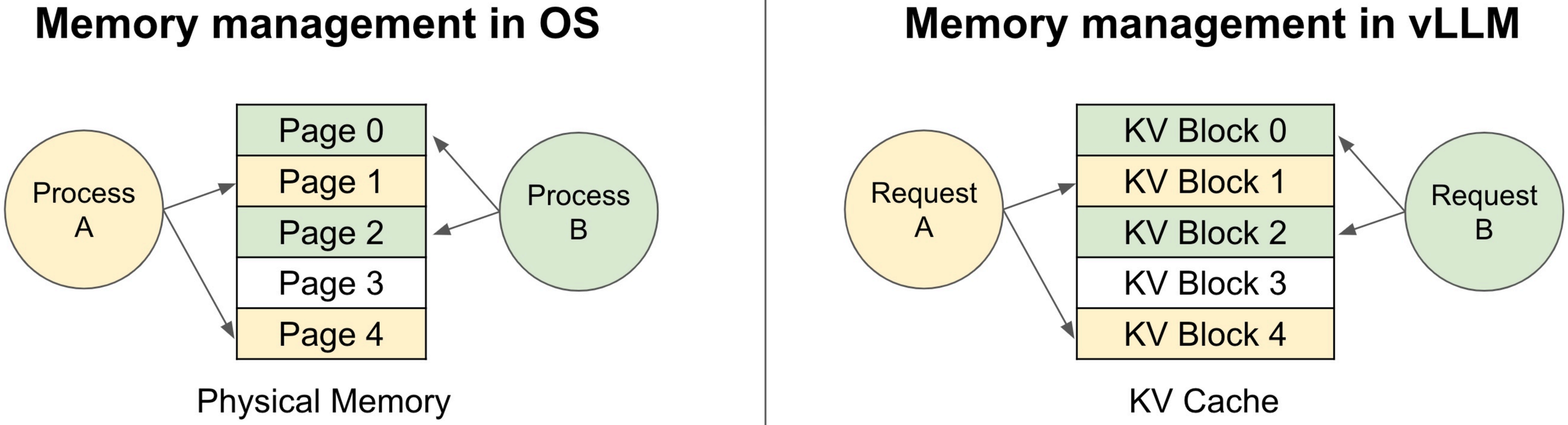


Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., ... & Stoica, I. (2023, October). Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th Symposium on Operating Systems Principles (pp. 611-626).

vLLM and Paged Attention

High-throughput and memory-efficient serving

- **Inspiration from operating systems (OS): virtual memory and paging**



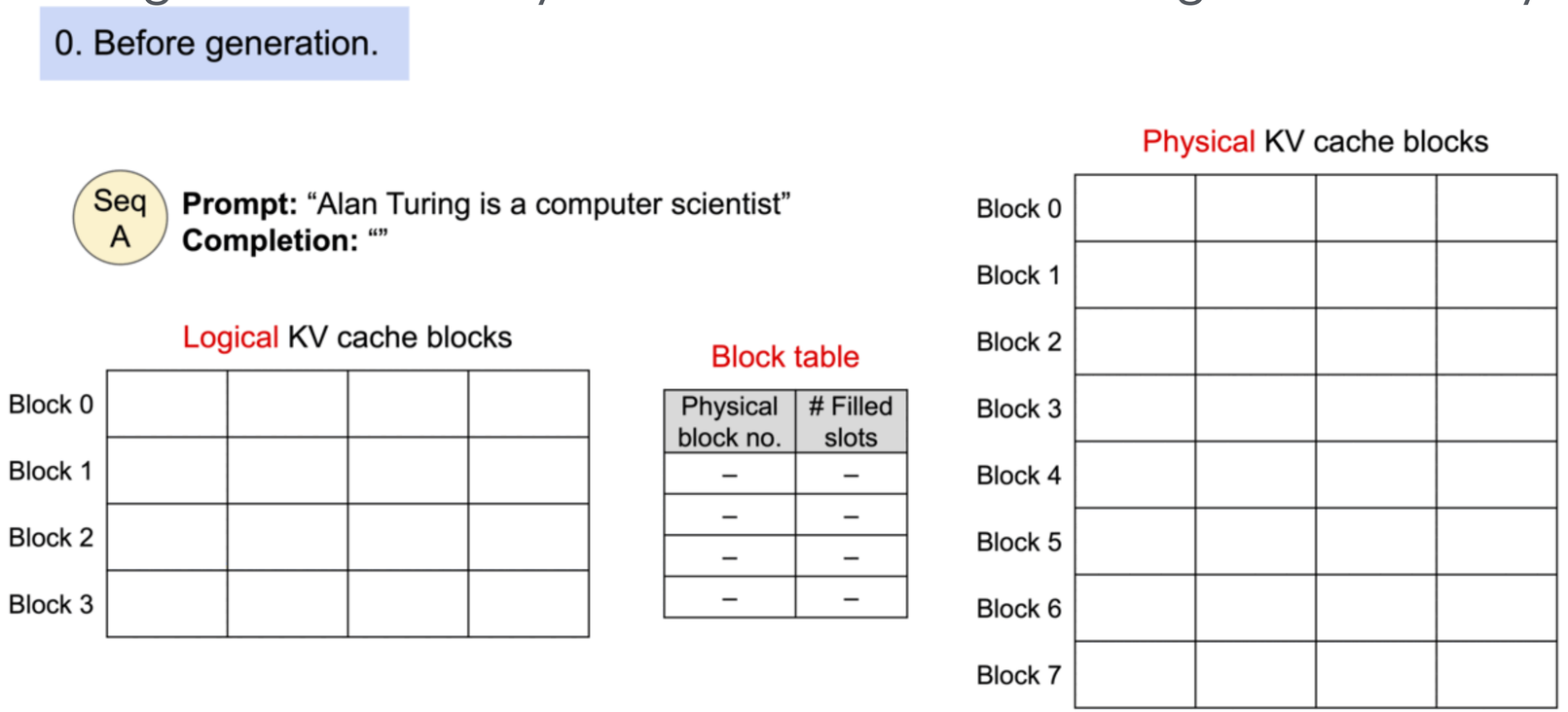
Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., ... & Stoica, I. (2023, October). Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th Symposium on Operating Systems Principles (pp. 611-626).

vLLM and Paged Attention

PagedAttention

- PagedAttention addresses the KV-cache memory fragmentation

- It allows for storing continuous keys and values in non-contiguous memory space

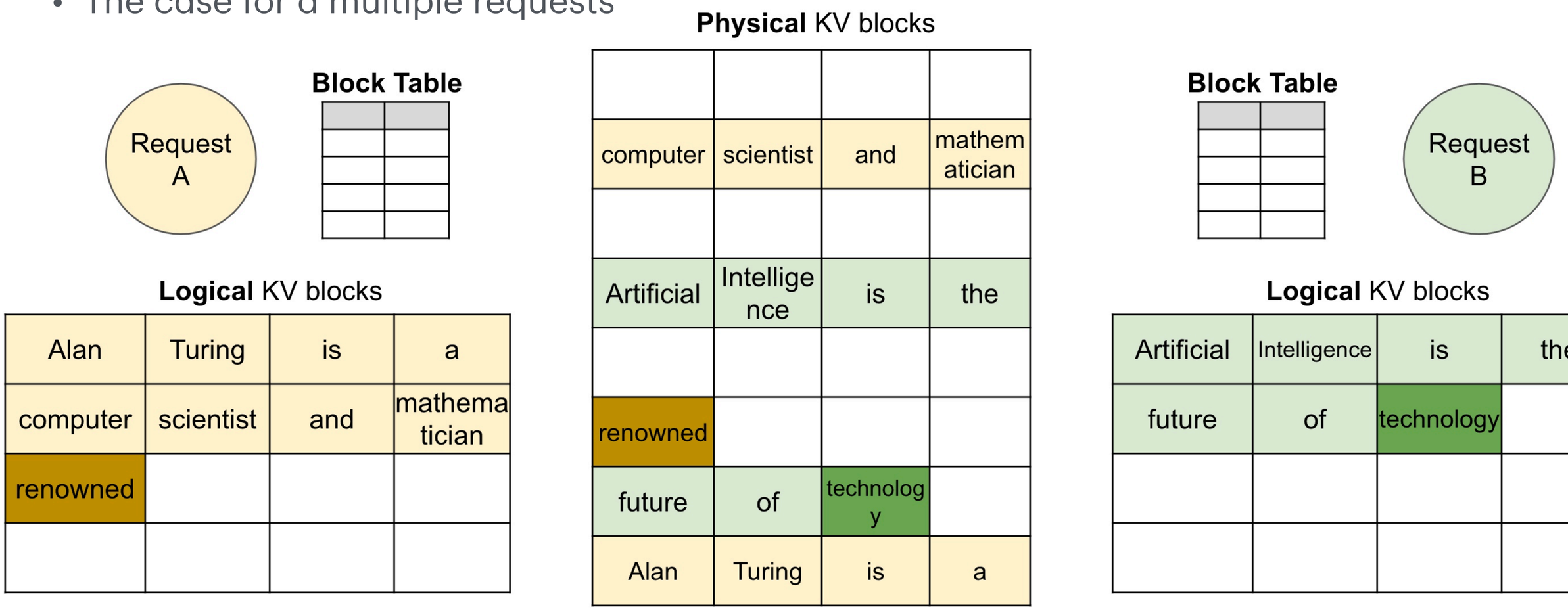


Example generation process for a request with PagedAttention

vLLM and Paged Attention

PagedAttention

- The case for a multiple requests

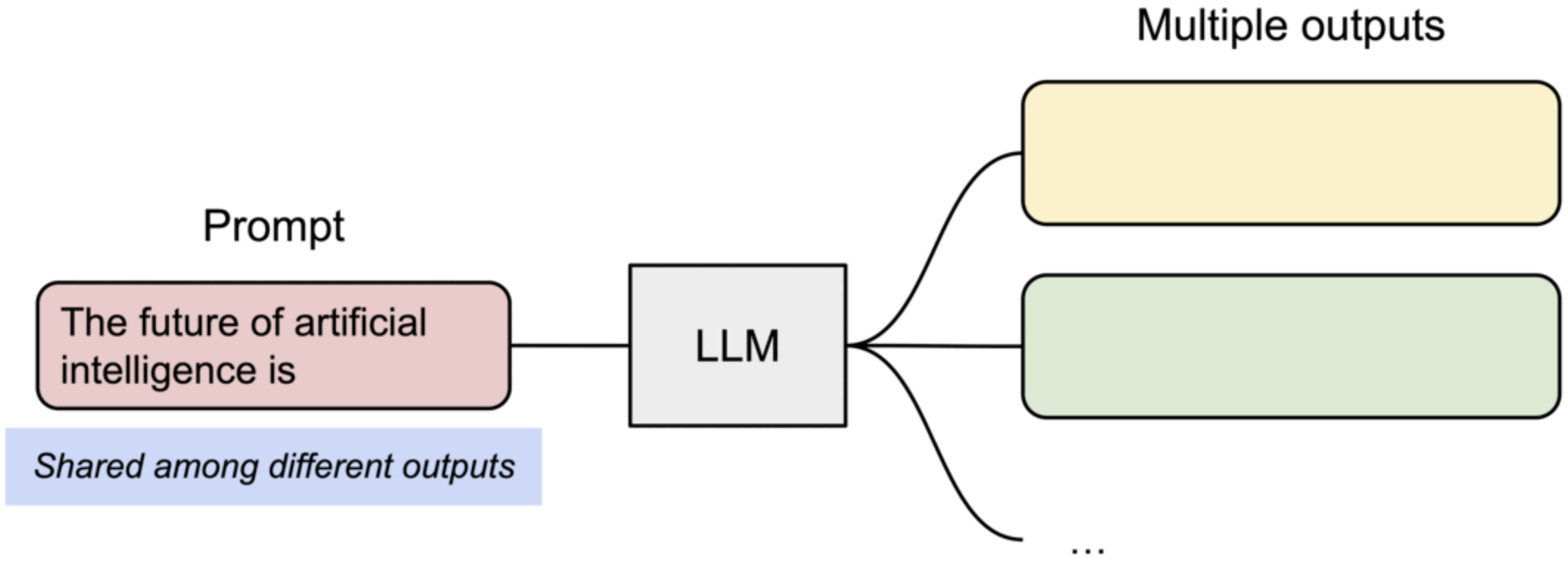


Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., ... & Stoica, I. (2023, October). Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th Symposium on Operating Systems Principles (pp. 611-626).

vLLM and Paged Attention

PagedAttention

- Dynamic block mapping enables prompt sharing in parallel sampling



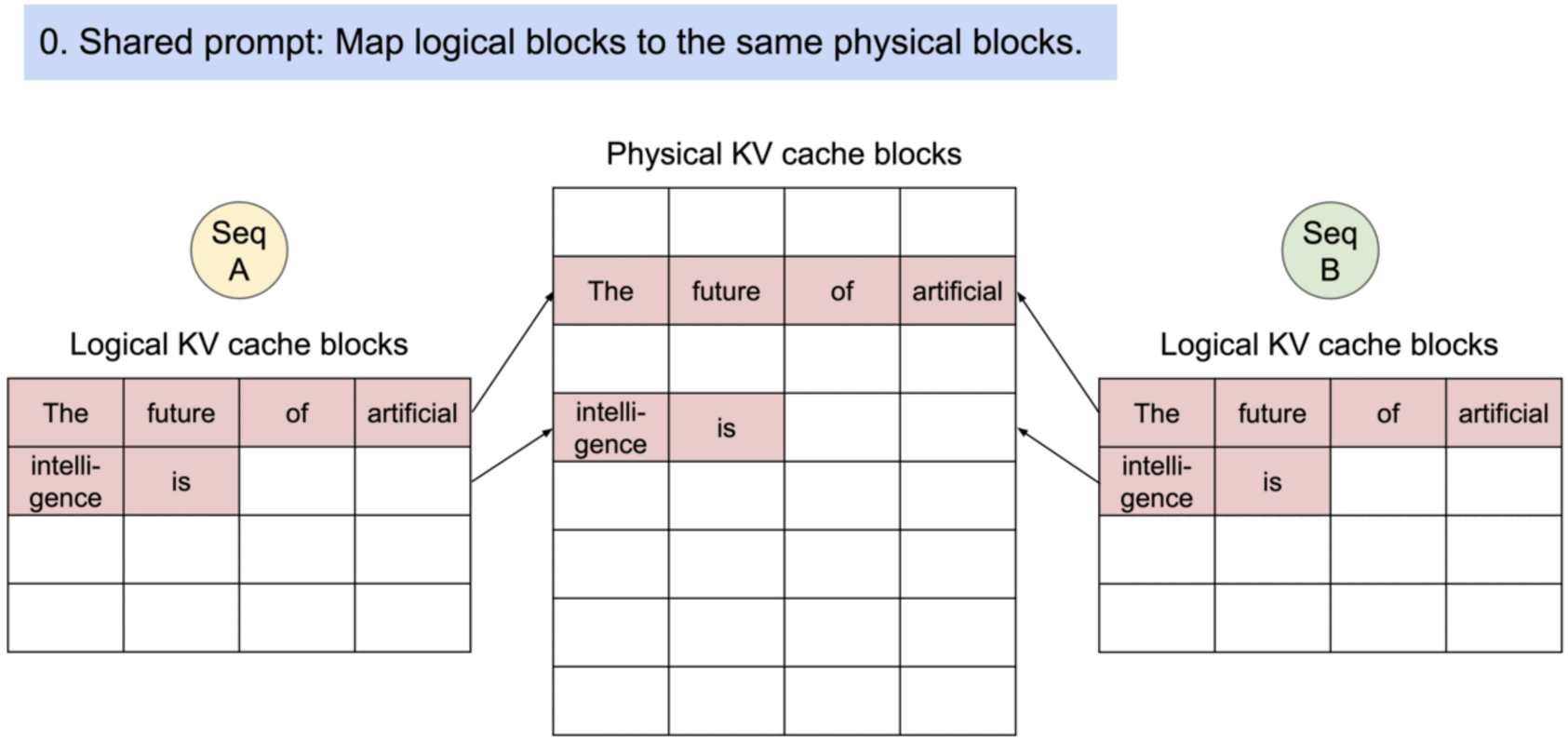
Example of parallel sampling

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., ... & Stoica, I. (2023, October). Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th Symposium on Operating Systems Principles (pp. 611-626).

vLLM and Paged Attention

PagedAttention

- The case for a request samples multiple outputs

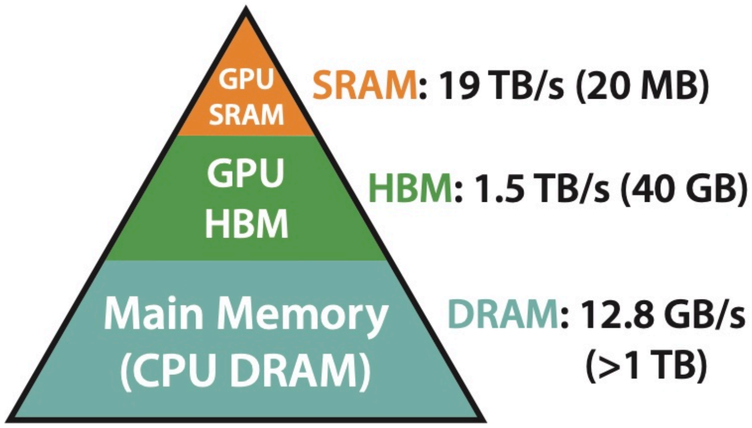


Example generation process for a request that samples multiple outputs

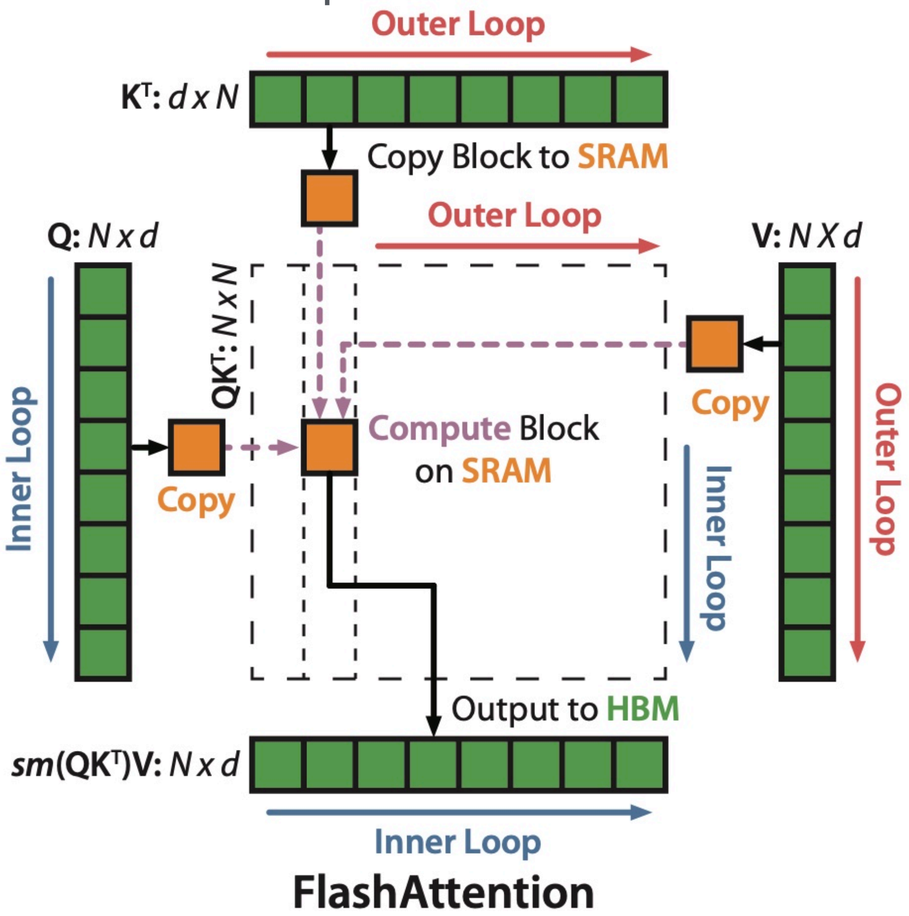
Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., ... & Stoica, I. (2023, October). Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th Symposium on Operating Systems Principles (pp. 611-626).

FlashAttention

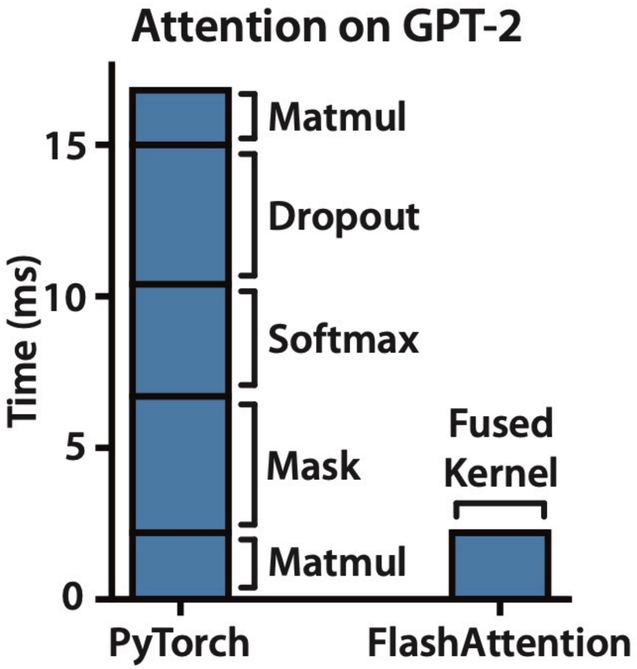
- Use tiling to prevent the materialization of the large $N \times N$ attention matrix, thus avoid using the slow HBM; kernel fusion technique



Memory Hierarchy with Bandwidth & Memory Size



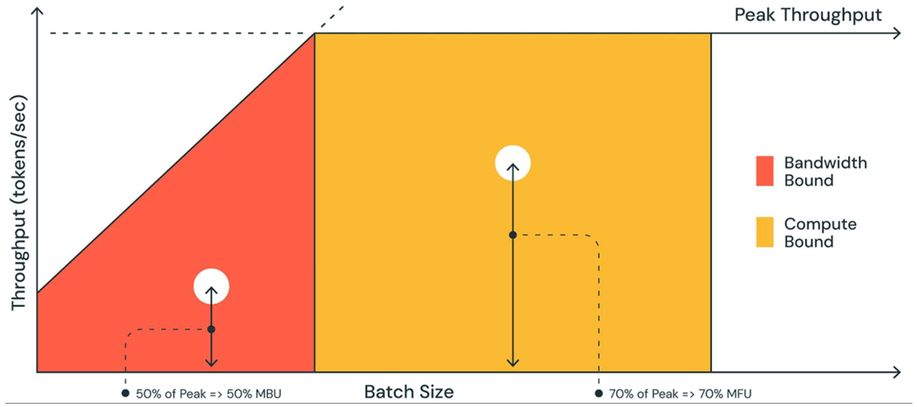
FlashAttention



Dao, T., Fu, D., Ermon, S., Rudra, A., & Ré, C. (2022). Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in Neural Information Processing Systems, 35, 16344-16359.

Speculative Decoding

Accelerating memory-bounded generation



- The decoding phase of LLM generates outputs token by token, which is highly memory-bounded (especially at a small batch size)
- There are two models in speculative decoding
  - Draft model: a small LLM (e.g., 7B)
  - Target model: a large LLM (e.g., 175B, the one to be accelerated)
- Procedure
  - The draft model decodes K tokens autoregressively
  - Feed the K generated tokens in parallel into the target model and get the predicted probabilities on each location
  - Decide if we want to keep the K tokens or reject them

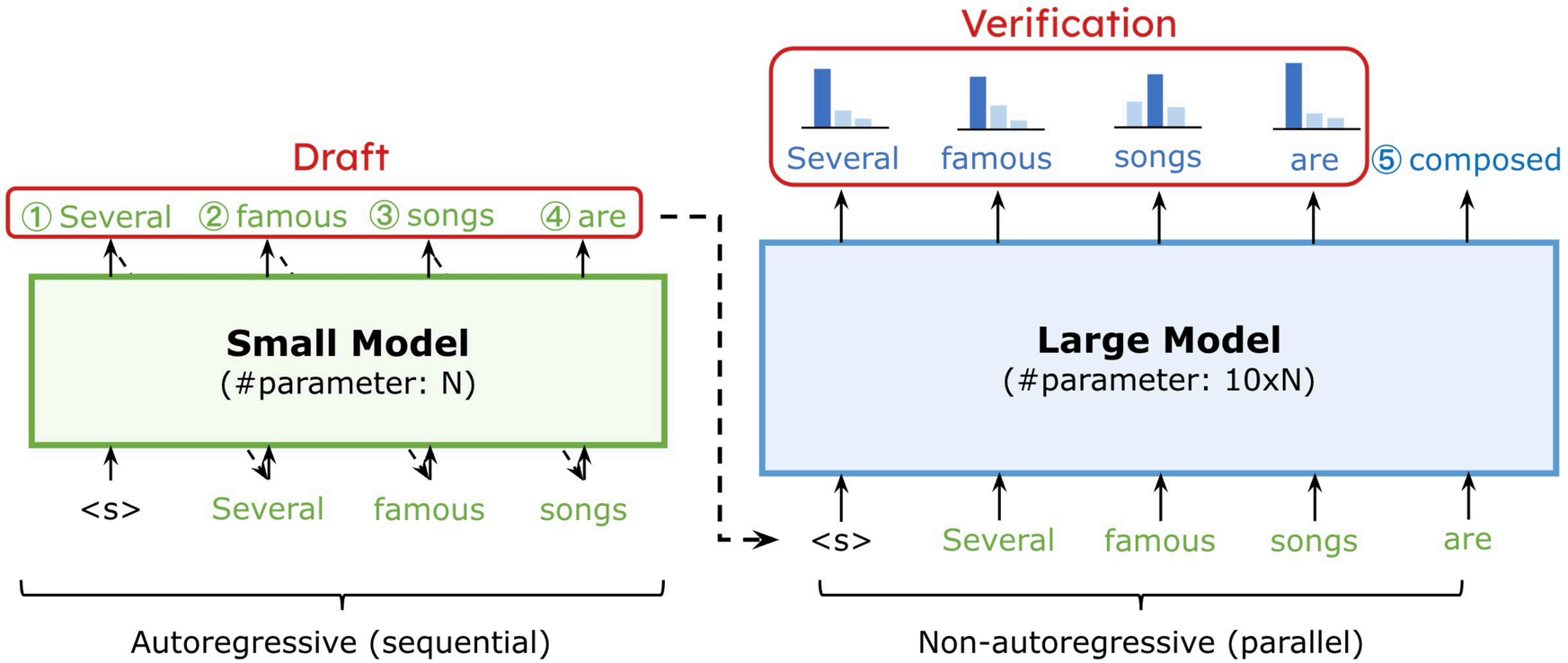
Leviathan, Y., Kalman, M., & Matias, Y. (2023, July). Fast inference from transformers via speculative decoding. In International Conference on Machine Learning (pp. 19274-19286). PMLR.

Speculative Decoding

Accelerating memory-bounded generation

- From generator to verifier

Small model writes a draft → Large model verifies it

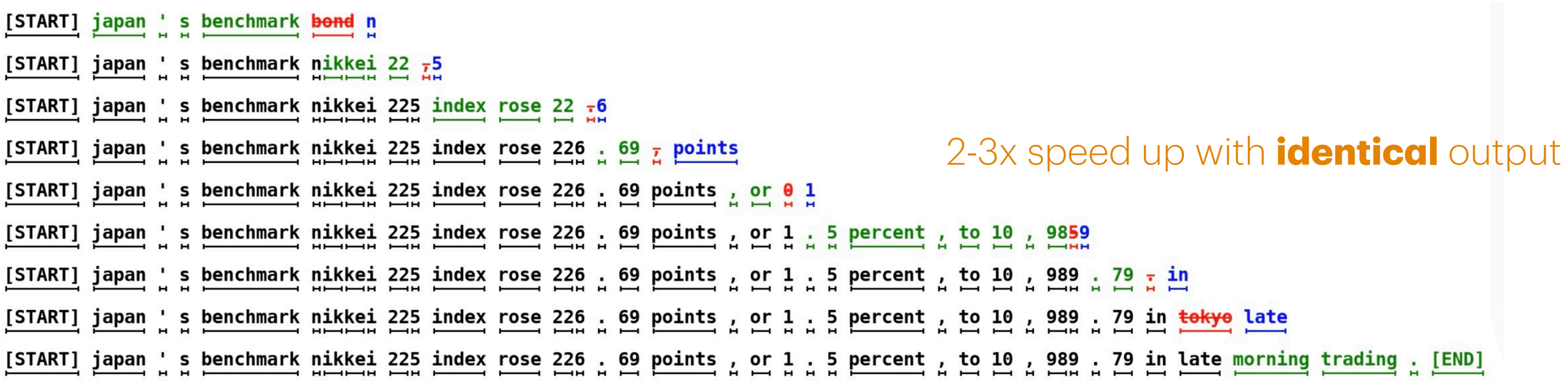


Leviathan, Y., Kalman, M., & Matias, Y. (2023, July). Fast inference from transformers via speculative decoding. In International Conference on Machine Learning (pp. 19274-19286). PMLR.

Speculative Decoding

Accelerating memory-bounded generation

- The decoding phase of LLM generates outputs token by token, which is highly memory-bounded (especially at a small batch size)
- Since multiple tokens are fed to the target model in parallel, it lifts the memory bottleneck
- An example from speculative decoding (green: accepted; red: rejected; blue: correction)



2-3x speed up with **identical** output

Leviathan, Y., Kalman, M., & Matias, Y. (2023, July). Fast inference from transformers via speculative decoding. In International Conference on Machine Learning (pp. 19274-19286). PMLR.

Batching

Maximizing throughput of LLM serving

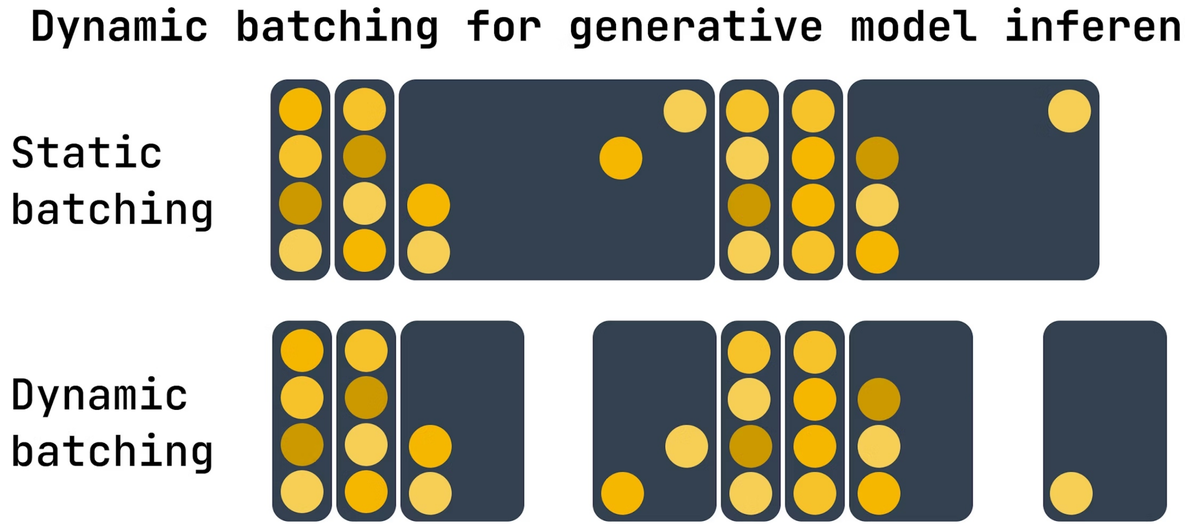
- **Why Batching?**
  - Running multiple inputs simultaneously maximizes throughput by fully utilizing the GPU resources, avoiding idle time
- **Batching methods**
  - **No Batching:** each request is processed individually, leading to underutilization of GPU resources
  - **Static Batching:** waits for a full batch of requests before processing. Good for scheduled tasks (can be processed offline); increases the latency for online tasks
  - **Dynamic Batching:** batches are processed when full or after a set time delay to balance throughput and latency
  - **Continuous Batching (aka., In-Flight Batching):** processes requests token-by-token, ideal for LLMs, improving GPU utilization by eliminating idle time waiting for the longest response

Continuous vs dynamic batching for AI inference

Dynamic Batching

Flexibility in high traffic

- Requests are collected and processed once the **batch is full** or **the maximum time has elapsed**
- Suitable for models like Stable Diffusion where **latency matters**
- **Analogy**: A bus that leaves when full or after a timer expires, ensuring minimal wait time
- Ideal for: Generative models with **uniform inference latency**, balancing throughput and latency



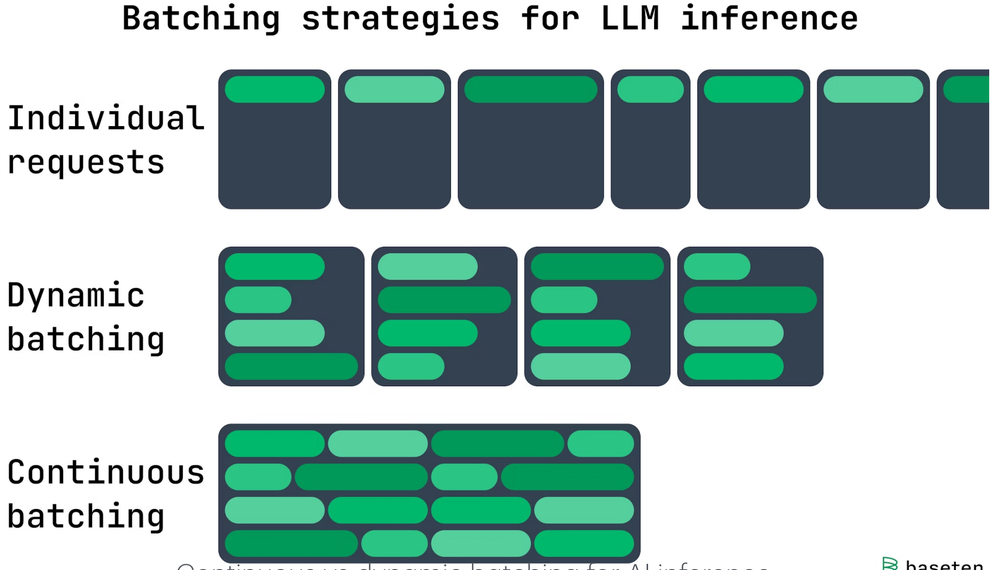
Continuous vs dynamic batching for AI inference

baseten

Continuous Batching (In-Flight Batching)

Optimized for LLM (token-by-token generative model) serving

- Works token-by-token, **processing new requests as previous ones finish**
- Maximizes GPU efficiency by **avoiding idle time** while waiting for the longest response
- **Analogy**: A bus where passengers get off at different stops, making space for new riders
- **Ideal** for: LLMs with varying output lengths, optimizing next-token generation



Continuous vs dynamic batching for AI inference

baseten

Lab 5 is out

Due in two weeks.

Reference

- Dettmers, T., Lewis, M., Belkada, Y., & Zettlemoyer, L. (2022). Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. Advances in Neural Information Processing Systems, 35, 30318-30332.
- Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. (2023, July). Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning (pp. 38087-38099). PMLR.
- Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., ... & Han, S. (2024). AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, 6, 87-100.
- Lin, J., Yin, H., Ping, W., Molchanov, P., Shoeybi, M., & Han, S. (2024). Vila: On pre-training for visual language models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 26689-26699).
- TinyChat
- Sun, M., Liu, Z., Bair, A., & Kolter, J. Z. (2023). A simple and effective pruning approach for large language models. arXiv preprint arXiv:2306.11695.
- Liu, Z., Wang, J., Dao, T., Zhou, T., Yuan, B., Song, Z., ... & Chen, B. (2023, July). Deja vu: Contextual sparsity for efficient llms at inference time. In International Conference on Machine Learning (pp. 22137-22176). PMLR.
- Fedus, W., Zoph, B., & Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. Journal of Machine Learning Research, 23(120), 1-39.
- Fedus, W., Dean, J., & Zoph, B. (2022). A review of sparse expert models in deep learning. arXiv preprint arXiv:2209.01667.
- Wang, H., Zhang, Z., & Han, S. (2021, February). Spatten: Efficient sparse attention architecture with cascade token and head pruning. In 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 97-110). IEEE.
- Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., ... & Chen, B. (2023). H2o: Heavy-hitter oracle for efficient generative inference of large language models. Advances in Neural Information Processing Systems, 36, 34661-34710.
- LLM Inference Performance Engineering: Best Practices
- Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., ... & Stoica, I. (2023, October). Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th Symposium on Operating Systems Principles (pp. 611-626).
- Dao, T., Fu, D., Ermon, S., Rudra, A., & Ré, C. (2022). Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in Neural Information Processing Systems, 35, 16344-16359.
- Leviathan, Y., Kalman, M., & Matias, Y. (2023, July). Fast inference from transformers via speculative decoding. In International Conference on Machine Learning (pp. 19274-19286). PMLR.
- LLM Deployment Techniques [MIT 6.5940]