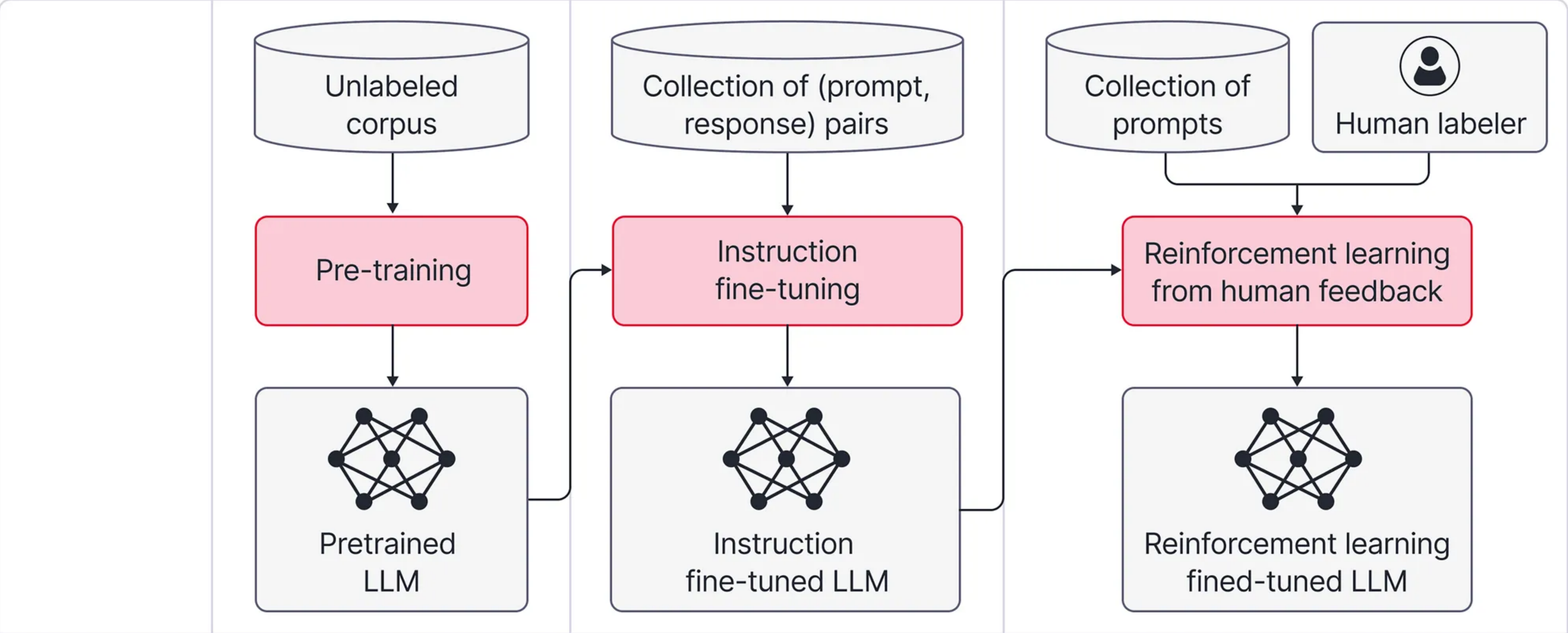
# LLM compression

## EECE695D: Efficient ML Systems

**Spring 2025**

# Compressing LLMs

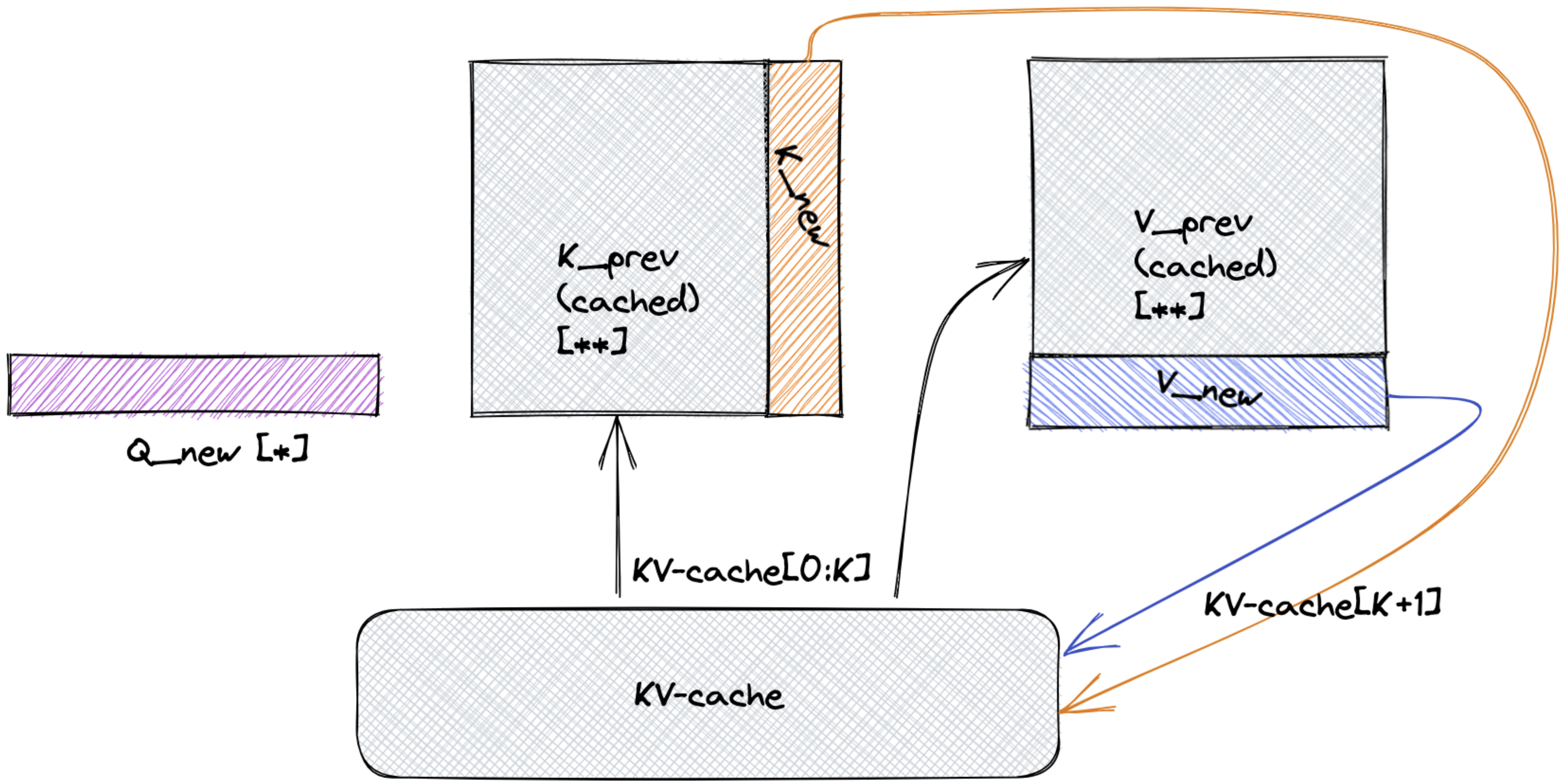
- **Post-training compression** has been the mainstream
  - Retraining cost is too large, including alignment
  - **2025.** Shifting toward methods involving retraining (e.g., Gemma QAT)




| | | | |
|---|---|---|---|
| Dataset size | x00 billions to 1.x trillion tokens | ~xk to x0k (prompt, response) | ~x0k prompt |
| Example of models | GPT-3, LLaMA, Falcon, BLOOM | Dolly-v2, Falcon-instruct | Claude, GPT-4, ChatGPT |
| | (a)Pre-training | (b) Instruction fine-tuning | (c) Reinforcement learning from human feedback |

# Compressing LLMs

- Also, more efforts on resolving **memory bottleneck**
  - e.g., weight quantization > weight & activation quantization
  - **2025.** Activation quantization as well, especially the KV cache



# Popular ideas

- Thus, much emphasis on finding a good **approximation** of original model

$$\min_{\hat{\theta}:\text{compressed}} \|f(x; w) - f(x; \hat{w})\|^2$$

- Two mainstream approaches:
  - **Hessian-based**
    "Minimize the compression error very carefully, using Hessians"
  - **Outlier-driven**
    "Identify outliers, and use these to keep compression error small"

# Hessian-based Approach

# Basic idea

- Already discussed Hessian-based loss approximation in sparsity
  - **Optimal Brain Damage**
    - Approximate the loss of removing a weight as:

$$f(x; w + \delta) - f(x; w) \approx \delta^\top \mathbf{H} \delta$$

  - **Optimal Brain Surgeon**
    - Prune the weight with minimal score: $w_i^2 / 2[\mathbf{H}^{-1}]_{ii}$
    - Update other weights by $-w_i \mathbf{H}^{-1} \mathbf{e}_i / [\mathbf{H}^{-1}]_{ii}$
- **Idea.** Perform OBS for LLMs

# Basic idea

- **Challenge.** Computing Hessian inverse for LLMs, multiple times
  - Very heavy: trillion x trillion matrix
  - **Idea.** Approximate by the layerwise subproblem

$$\min_{\hat{\mathbf{W}}} \|\mathbf{W}\mathbf{X} - \hat{\mathbf{W}}\mathbf{X}\|_2^2$$

    - Further approximate it by the rowwise subproblem

$$\min_{\hat{\mathbf{w}}} \|\mathbf{w}_{i,:}^\top \mathbf{X} - \hat{\mathbf{w}}^\top \mathbf{X}\|_2^2$$

    - Then the Hessian becomes: $\mathbf{H} = 2\mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{d_{\mathrm{col}} \times d_{\mathrm{col}}}$
(same for all rows)

Frantar et al., "Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning," NeurIPS 2022

# Basic idea

- Again, computing the Hessian inverse can be done recursively by using the matrix inversion formula:

$$\mathbf{H}_m^{-1} = \mathbf{H}_{m-1}^{-1} - \frac{\mathbf{H}_{m-1}^{-1}\mathbf{x}_m\mathbf{x}_m^\top\mathbf{H}_{m-1}^{-1}}{N + \mathbf{x}_m\mathbf{H}_{m-1}^{-1}\mathbf{x}_m}$$

- **Problem.** Need to compute Hessian after removing each weight
    - LLMs are very large, so a lot of repetitions!

Frantar et al., "Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning," NeurIPS 2022

# Basic idea

- Fortunately, we have the following lemma:
- **Lemma.** Given an invertible $d_{\text{col}} \times d_{\text{col}}$ matrix $\mathbf{H}$ and its inverse $\mathbf{H}^{-1}$, let $\mathbf{H}_{-i}$ denote the Hessian with row and column $i$ removed. Then, we have:

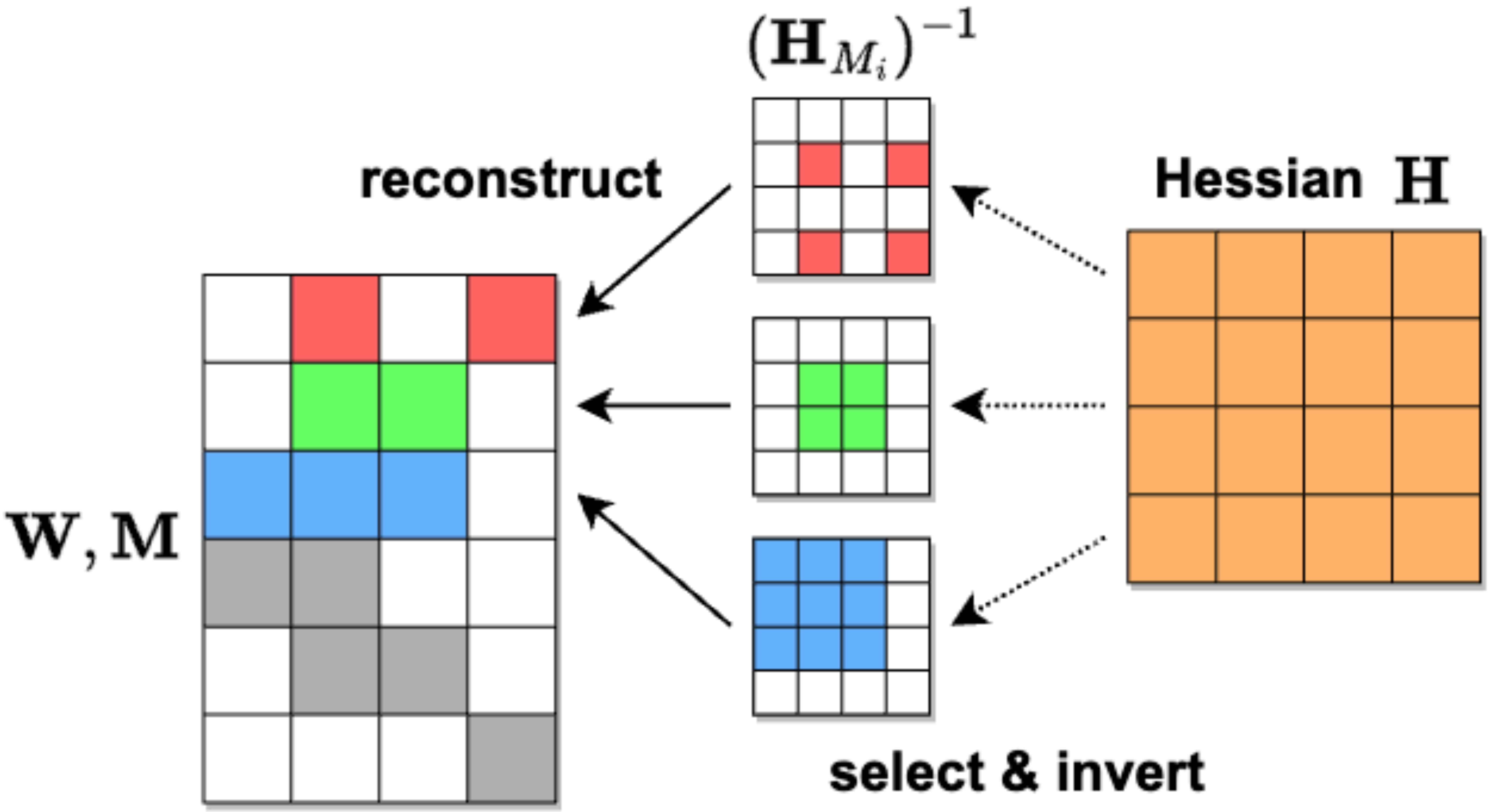
$$\mathbf{H}_{-i}^{-1} = \left( \mathbf{H}^{-1} - \frac{1}{[\mathbf{H}^{-1}]_{ii}} \mathbf{H}_{:,i}^{-1} \mathbf{H}_{i,:}^{-1} \right)_{-i}$$

  - Gaussian elimination of row/col $i$ in $\mathbf{H}^{-1}$, then remove them completely
  - Complexity of $\Theta(d_{\text{col}}^2)$
- Allows parallel processing of rows

Frantar et al., "Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning," NeurIPS 2022

# SparseGPT

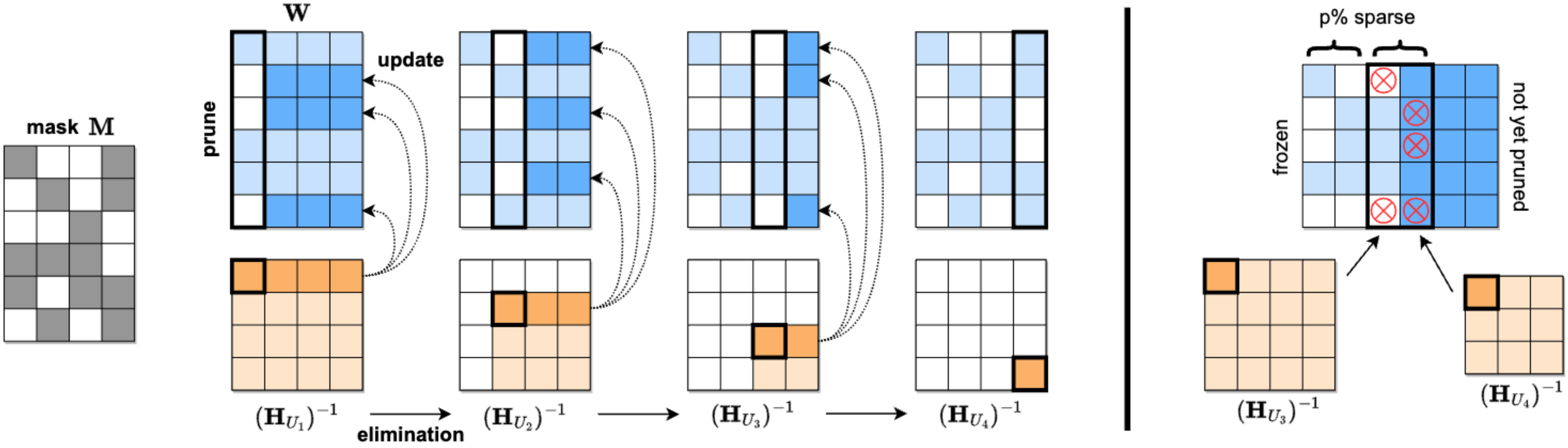
- **Problem.** Keeping track of row-wise Hessian inverse is memory-heavy




Frantar et al., "SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot," ICML 2023

# SparseGPT

- **Idea.** Iterate over columns, removing certain fraction at a time and freezing the surviving weights




Frantar et al., "SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot," ICML 2023

# Quantization

- Similarly, the weight quantization can be done using Hessians
  - For simplicity, assume that we have a fixed grid, and put weights on grid one-by-one.
  - Similar derivation gives that the weight updates needed to compensate for quantizing $w_q$ is:

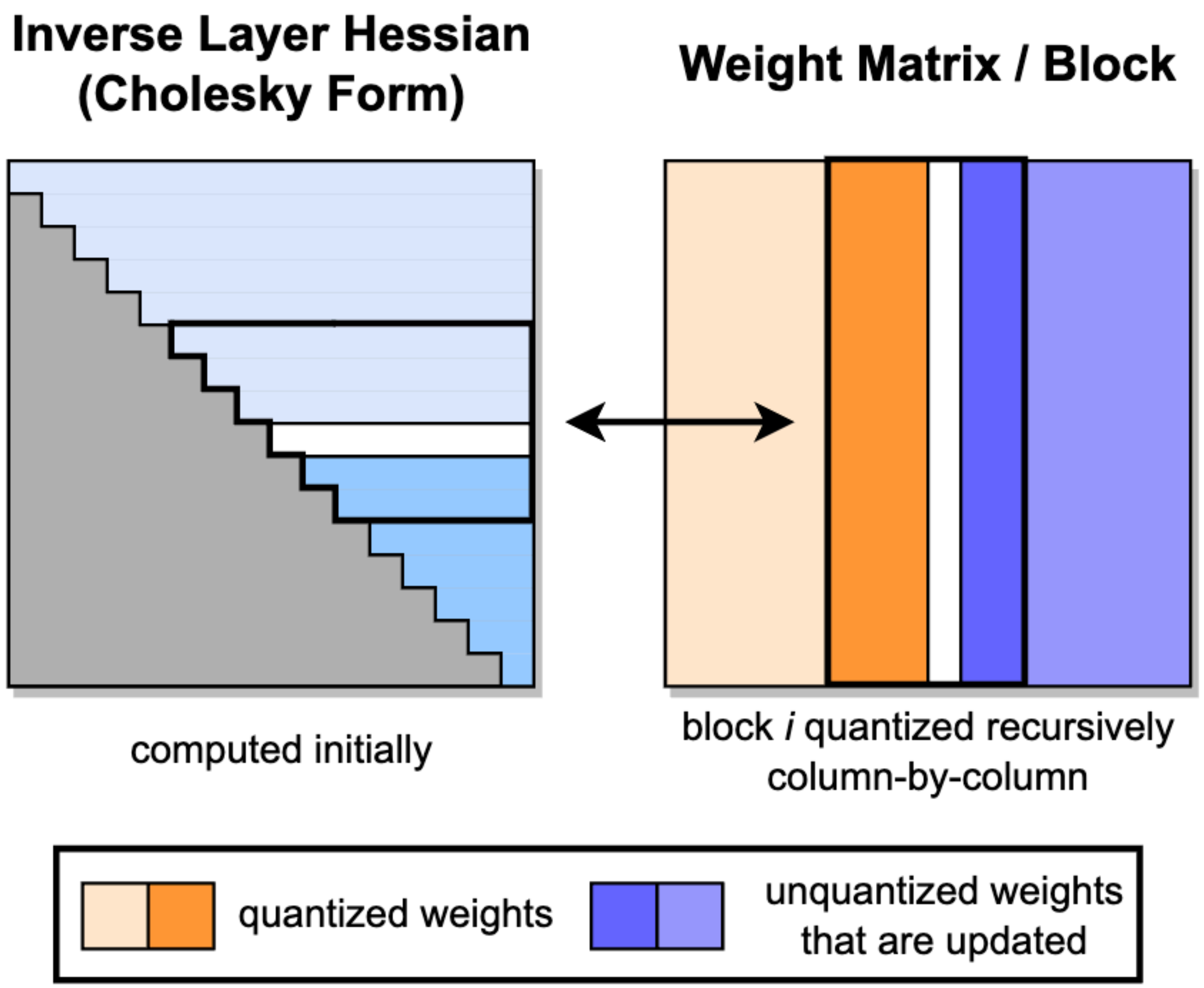
$$\delta = \frac{\mathrm{quant}(w_q) - w_q}{[\mathbf{H}^{-1}]_{qq}} \mathbf{H}^{-1} e_q$$

    - Weights quantized later are likelier to change more
    - **Heuristic.** Quantize the outliers, as soon as they appear

Frantar et al., "Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning," NeurIPS 2022

# GPTQ

- **Order.** Later, people discovered that the order does not matter much
  - Simple quantized in an order, in parallel, save time
- **More tricks** (see the paper)
  - <u>Lazy batching</u>: Update later columns a bit slowly to prevent frequent memory access
  - <u>Cholesky reformulation</u>: Improved numerical stability, avoiding the accumulation of errors from repeated updates

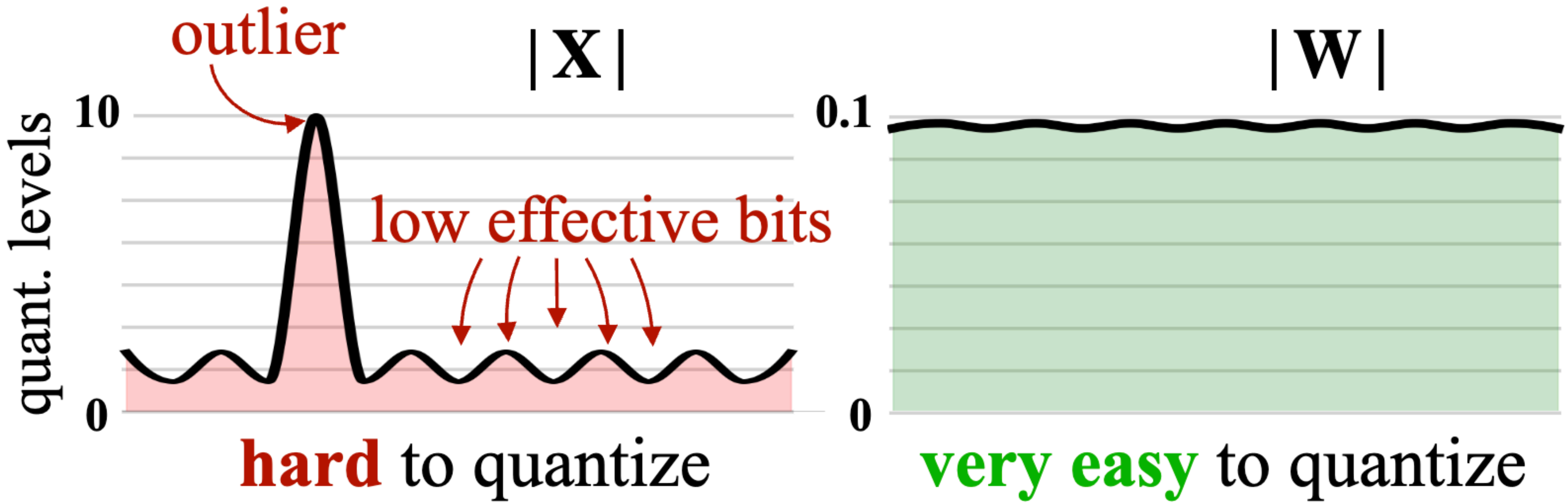



Frantar et al., "GPTQ: Accurate Post-Training Quantization for Generated Pre-Trained Transformers," ICLR 2023

# Outlier-driven Approach

# Outliers?

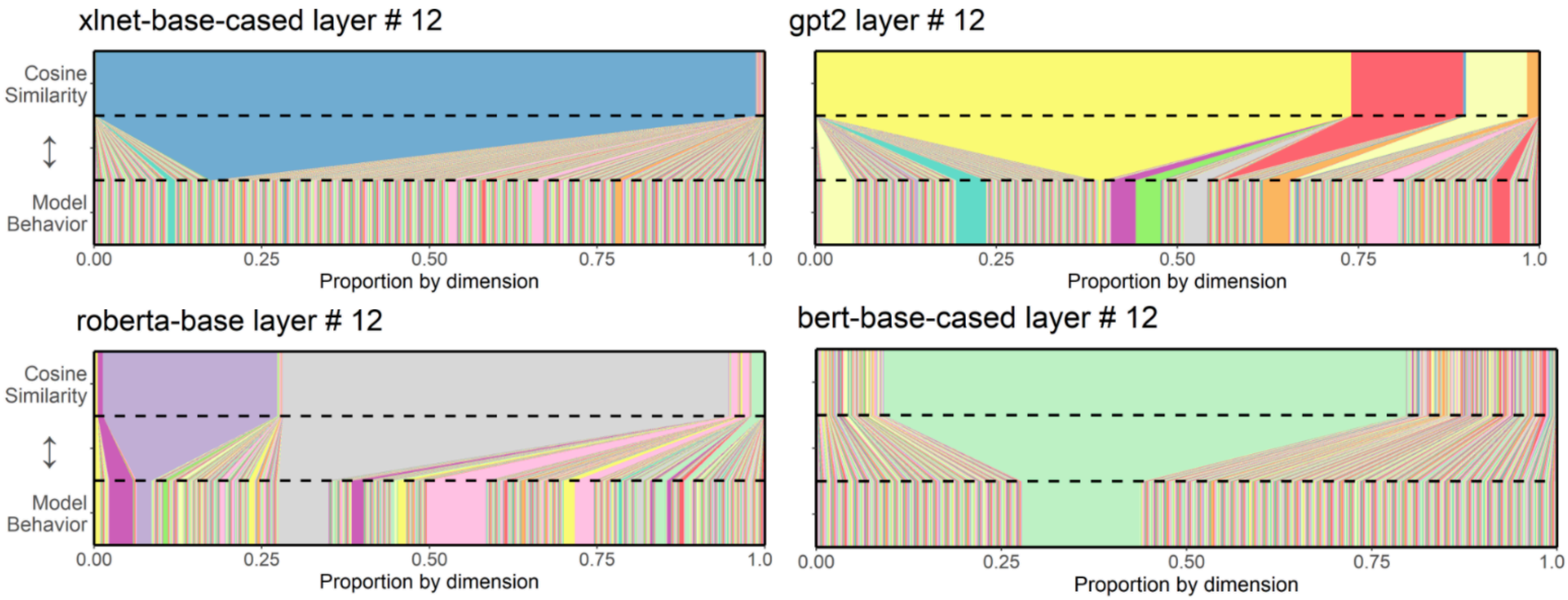
- Transformer activations of very large magnitude
  - Happens in a small number of **channels** / tokens / layers




Xiao et al., "SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models," ICML 2023

# Timkey & van Schijndel (EMNLP 2021)

- Observes that 1–3 channels in later layers are outliers
  - Dominates the cosine similarity computation

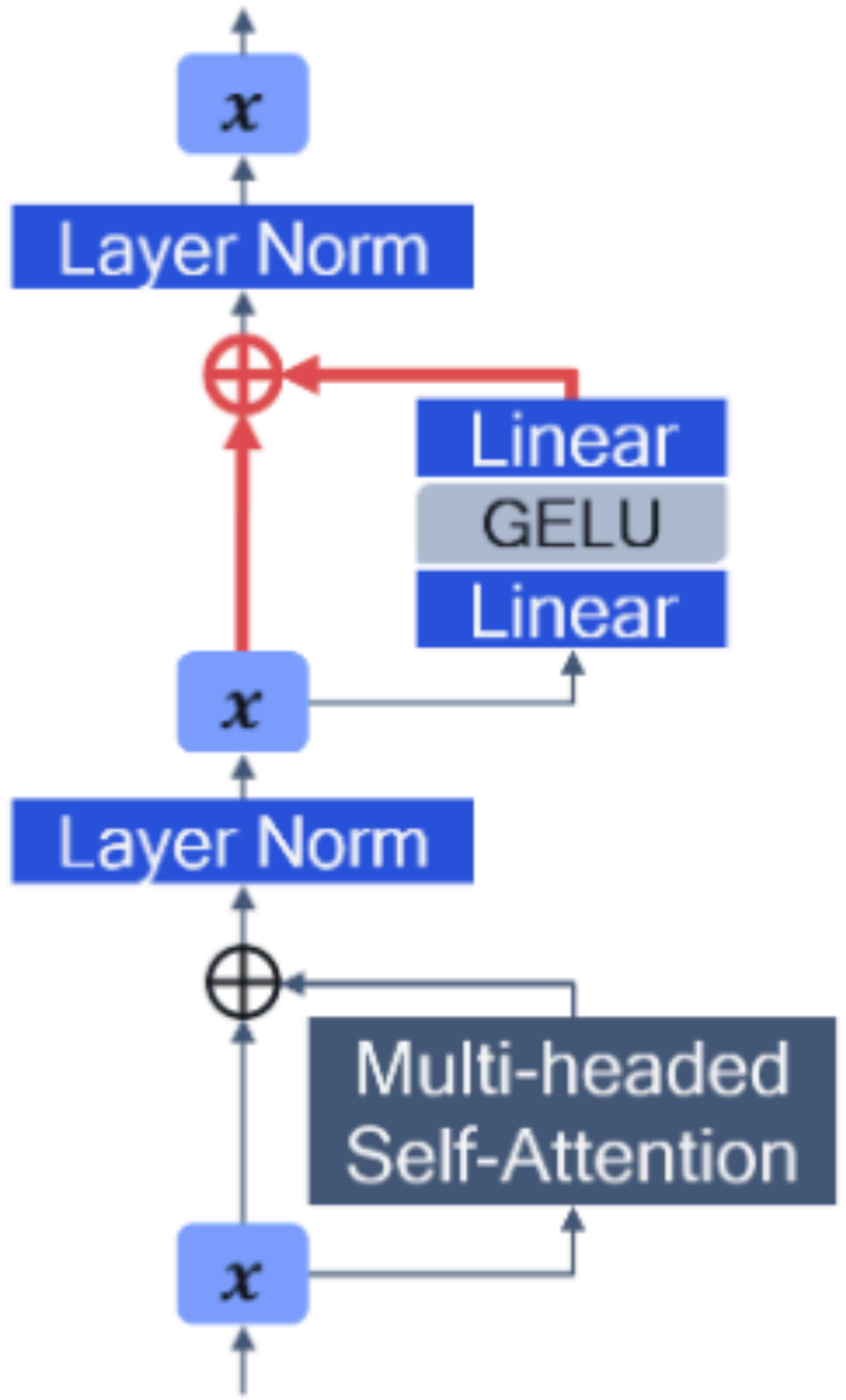



Timkey and van Schijndel., "All Bark and No Bite: Rogue Dimensions in Transformer Language Models Obscure Representational Quality," EMNLP 2021

# Bondarenko et al. (EMNLP 2021)

- Outliers are the residual sum after FFN of layer 10 & 11
  - Large accuracy drop of W32A8 / W8A8 performance on BERT

| Quantized activations | STS-B | MNLI | QNLI | RTE |
|---|---|---|---|---|
| none (FP32 model) | 89.09 | 84.91 | 91.58 | 70.40 |
| all | 62.64 | 42.67 | 50.74 | 48.74 |
| all, except softmax input | 70.92 | 42.54 | 51.84 | 48.74 |
| all, except sum of embeddings | 67.57 | 46.82 | 51.22 | 51.26 |
| all, except self-attention output | 70.47 | 46.57 | 50.98 | 50.90 |
| all, except softmax output | 72.83 | 50.35 | 50.23 | 49.46 |
| all, except residual connections after FFN | **81.57** | **82.56** | **89.73** | **67.15** |
| same as above, but for layers 10, 11 only | **79.40** | **81.24** | **88.03** | **63.90** |




Bondarenko et al., "Understanding and Overcoming the Challenges of Efficient Transformer Quantization," EMNLP 2021

# Bondarenko et al. (EMNLP 2021)

- Happens for [SEP] token, and small number of channels
    - Outliers have small variance, though

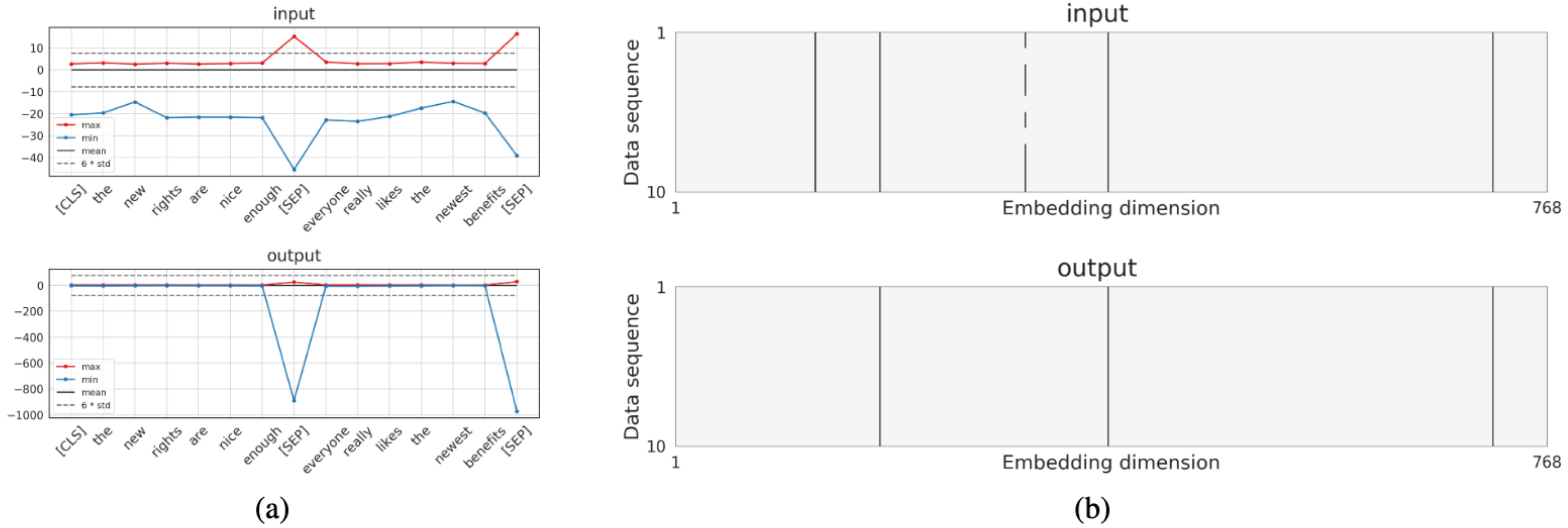
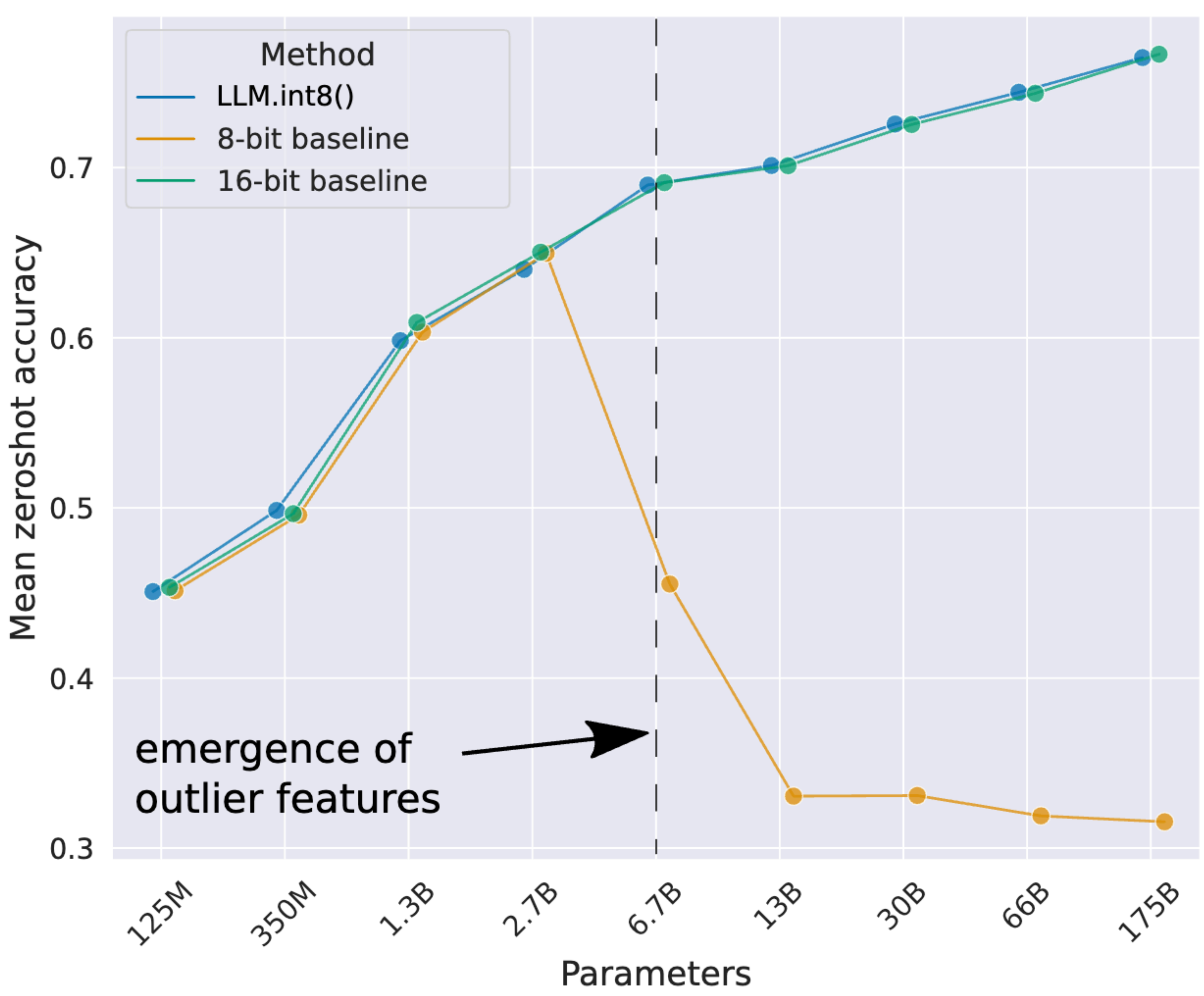


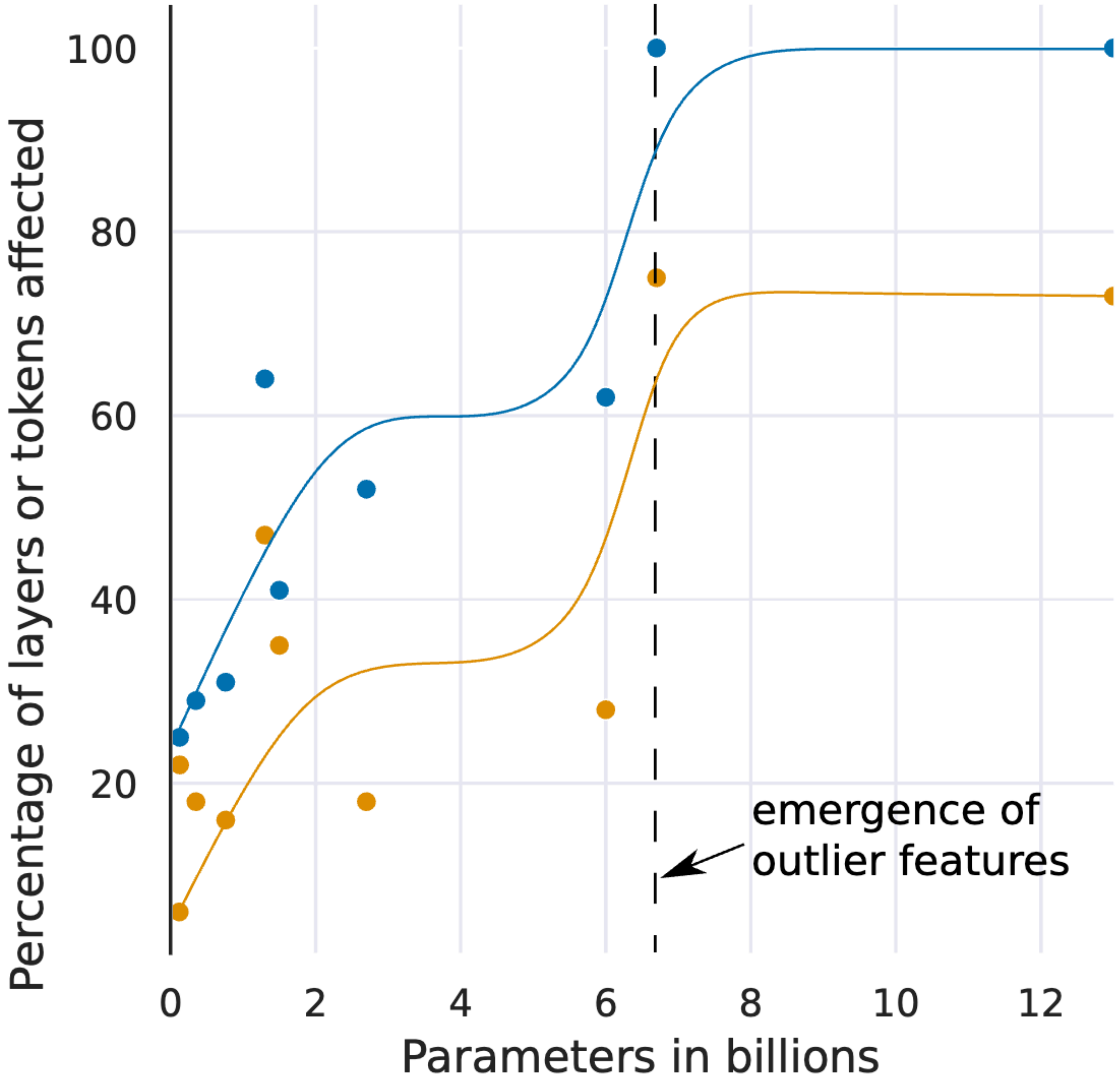

Figure 2: Full-precision FFN input (top row) and output (bottom row) in 11th layer of BERT. (a) Per-token ranges for first data sequence in the MNLI development set. (b) Visualization of outliers across embedding dimension for the first ten data sequences in the MNLI development set. Dark grey color indicates values that exceed six standard deviations from the mean of the activation tensor.

Bondarenko et al., "Understanding and Overcoming the Challenges of Efficient Transformer Quantization," EMNLP 2021

# Dettmers et al. (NeurIPS 2022)

- More significant in **>6.7B models** (attention projection & 1st FFN output)
  - At 6.7B scale, outliers occur in all layers and 75% of all tokens

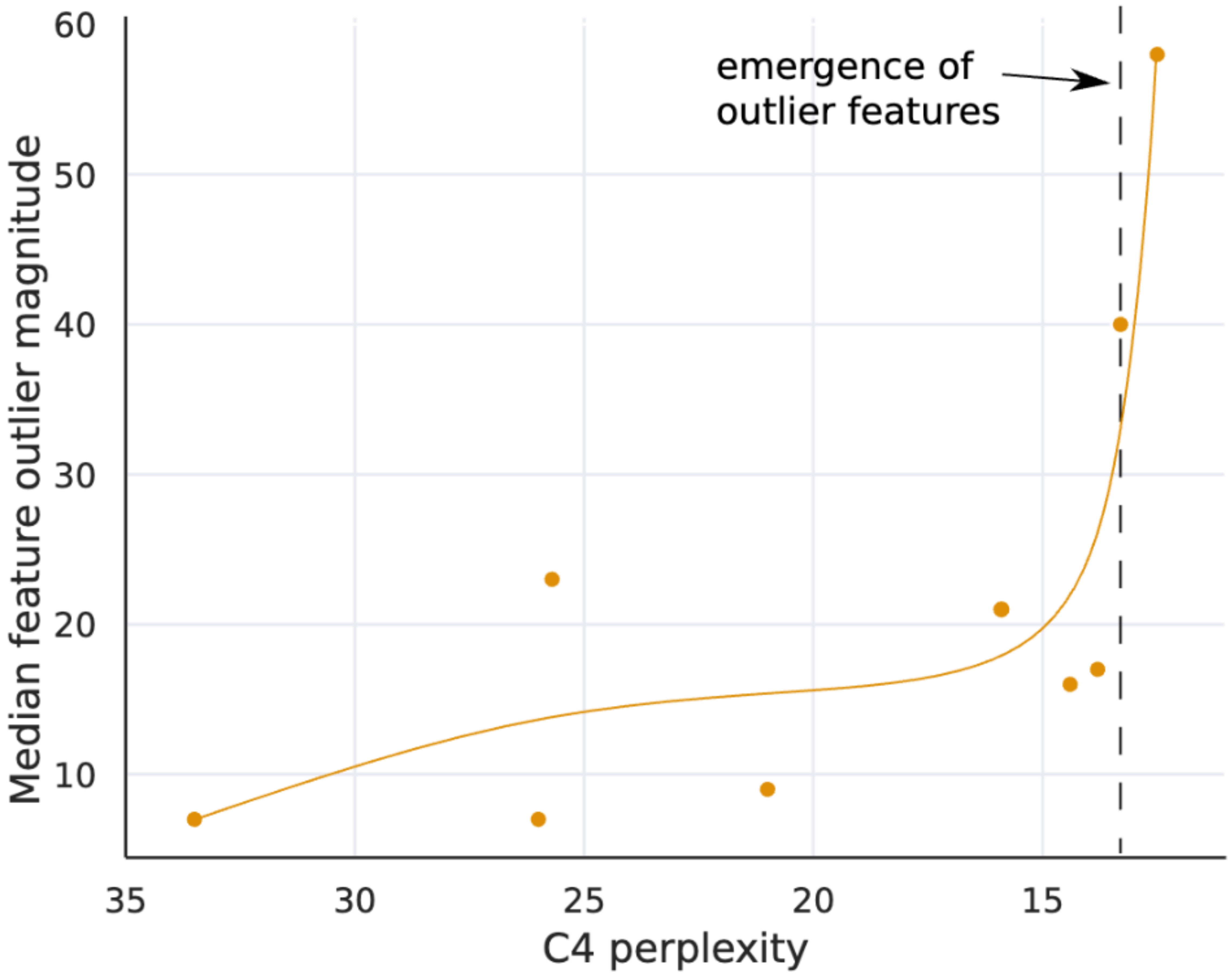


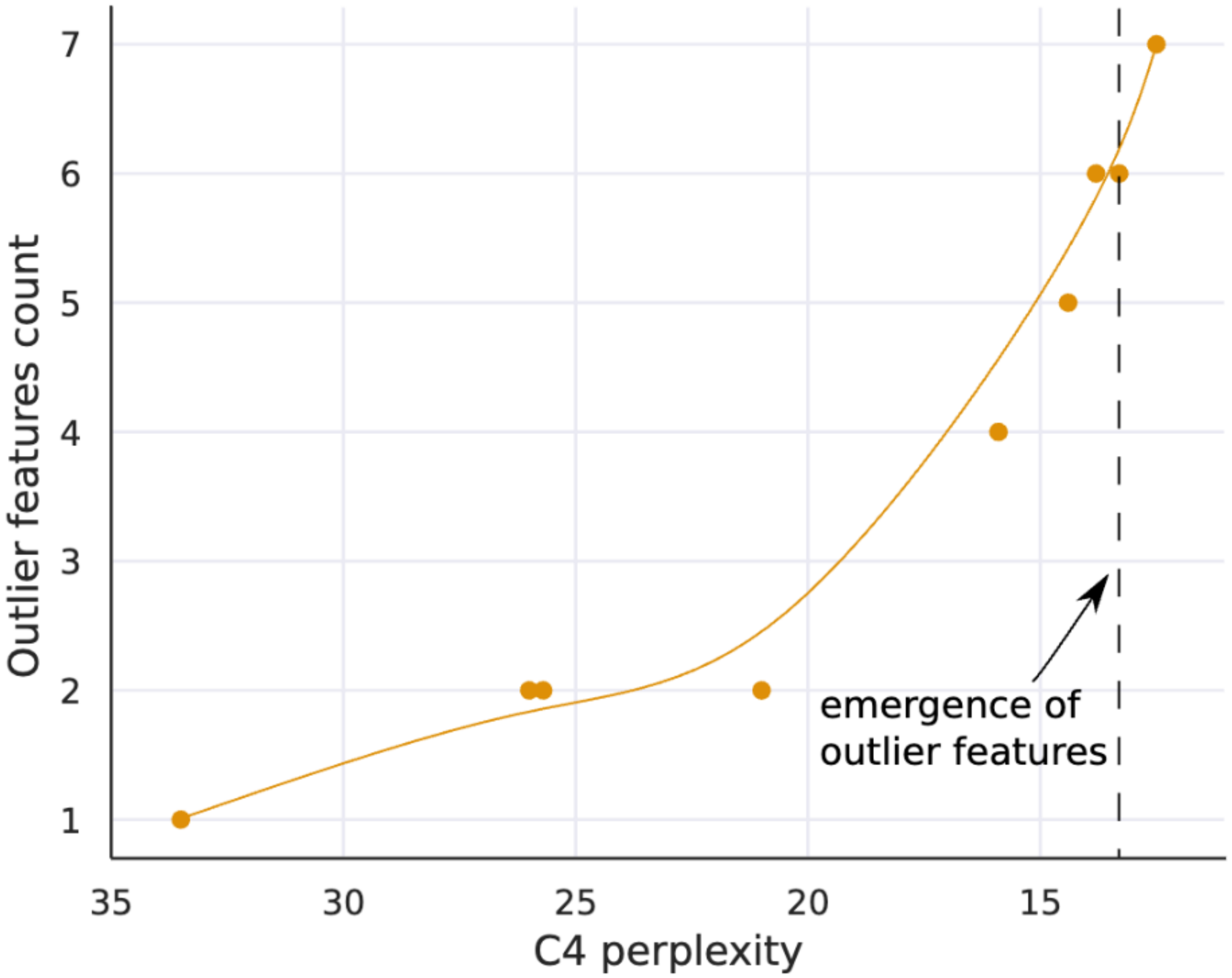




Dettmers et al., "LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale," NeurIPS 2022

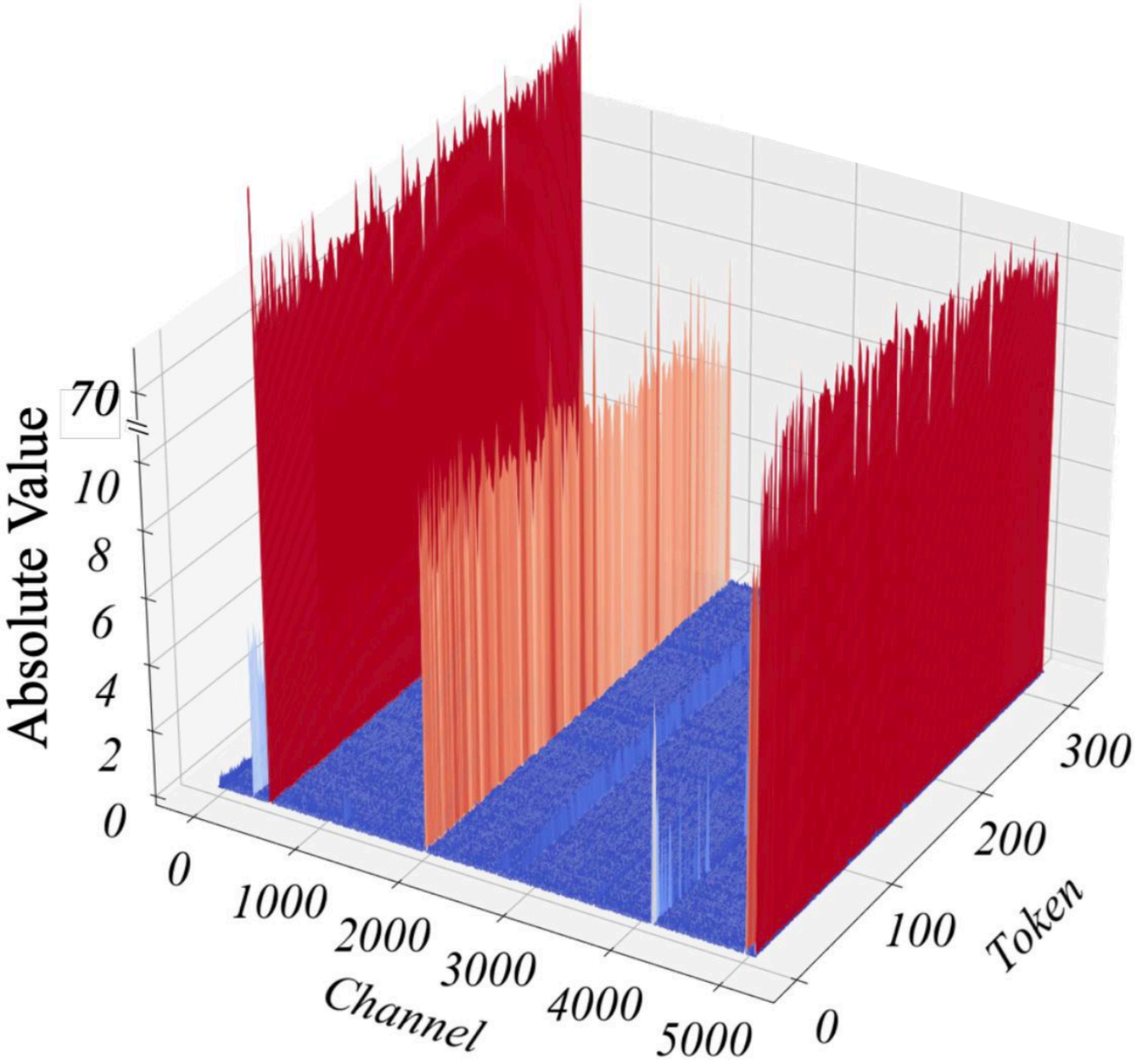
# Dettmers et al. (NeurIPS 2022)

- Larger model => Larger outlier magnitude, greater number of outlier features




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




**Activation (Original)**

Xiao et al., "SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models," ICML 2023

# Basic idea

- **Idea.** If certain channels are likelier to have larger input:
    - pruning / quantizing the weights connected to the channel is likelier to hurt the model accuracy more

$$\|\mathbf{W}\mathbf{X} - \hat{\mathbf{W}}\mathbf{X}\|^2$$

    - selecting quantization range can be problematic: all non-outliers are likely to be quantized to zero

- **Question.** If we know certain channels are outlier-prone, what can we do?

# Activation Quantization

- (1) Divide-and-Quantize
  - Assign different quantization range to different groups of channels

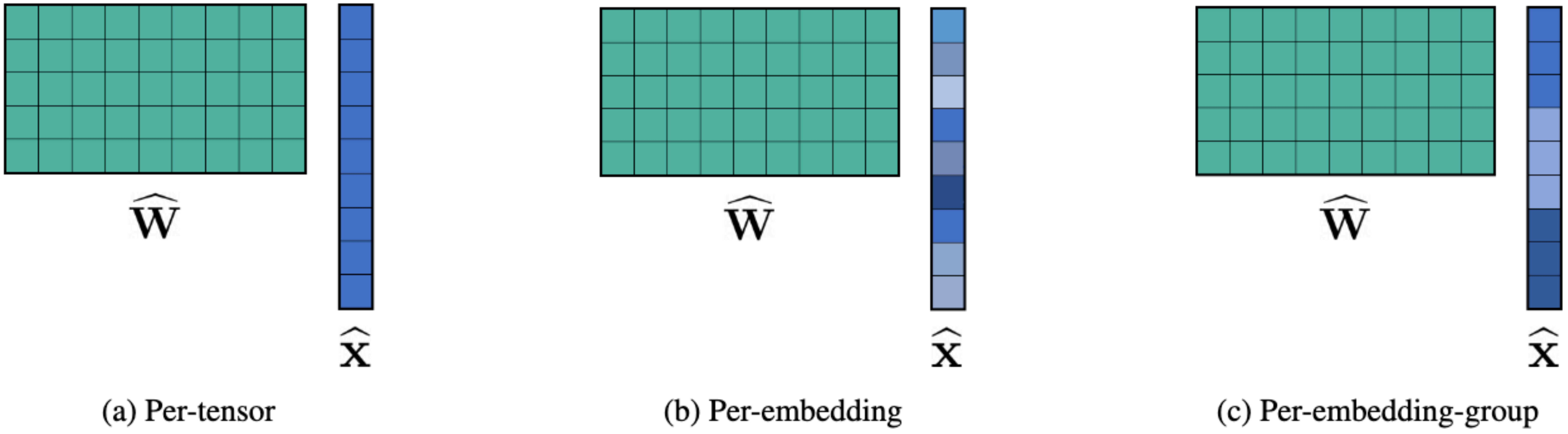
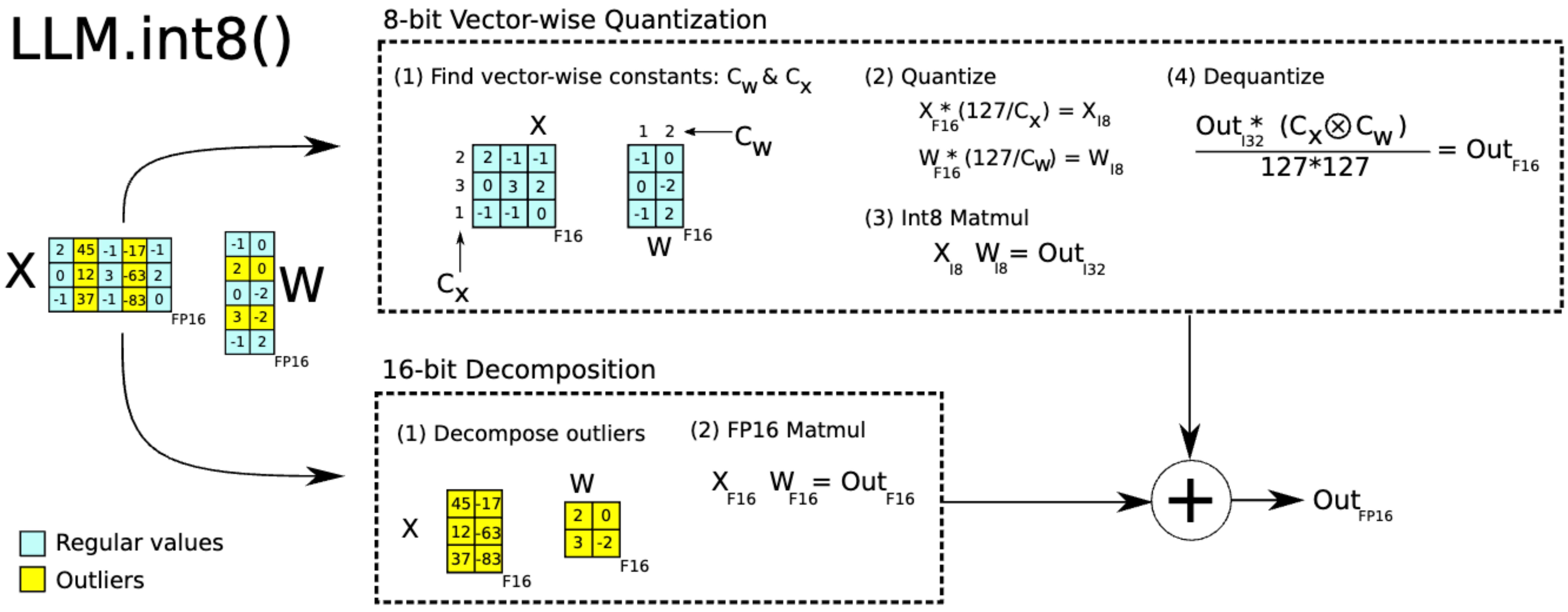



Figure 3: An overview for several choices of activation quantization granularity. The color indicates quantization parameter sharing. In all cases we assume per-tensor weight quantization.

Bondarenko et al., "Understanding and Overcoming the Challenges of Efficient Transformer Quantization," EMNLP 2021

# Activation Quantization

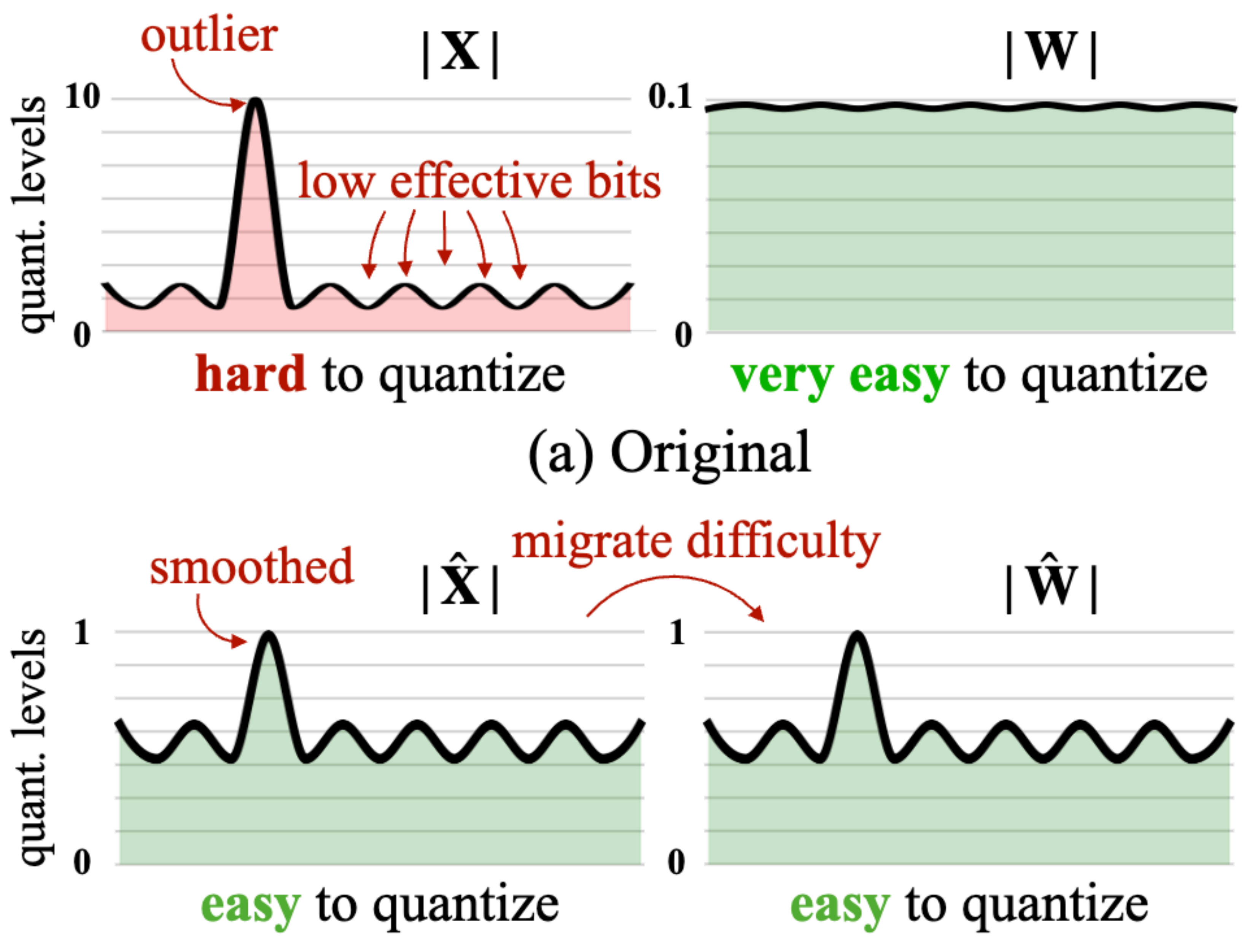
- (1) Divide-and-Quantize
  - Assign different precision to different channels



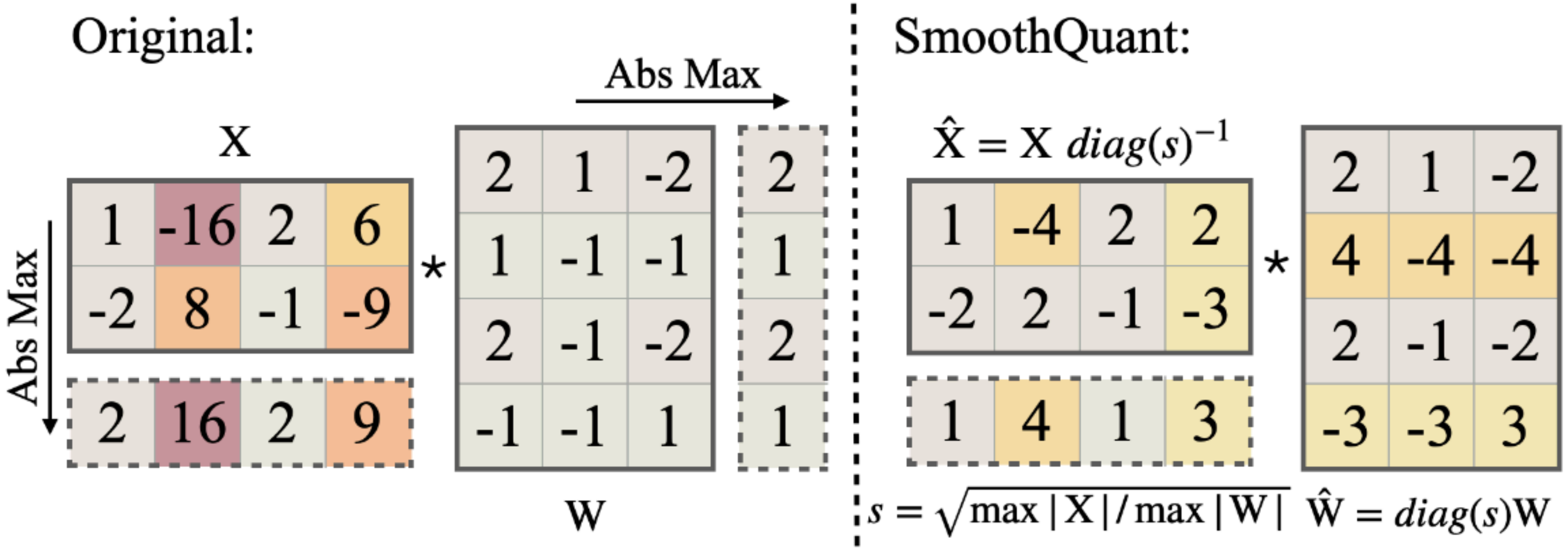

Dettmers et al., "LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale," NeurIPS 2022

# Activation Quantization

- (2) Migrate the difficulty to weight
  - Defer scaling factors in the weights $\mathbf{W}\mathbf{X} = (\mathbf{W}\mathbf{\Lambda}^{-1})(\mathbf{\Lambda}\mathbf{X})$
    - $\mathbf{\Lambda}_i = (\max(|\mathbf{X}_i|)/\max(|\mathbf{W}_i|))^{\alpha}$



(a) Original
(b) SmoothQuant




Wei et al., "Outlier Suppression: Pushing the Limit of Low-bit Transformer Language Models,"NeurIPS 2022
Xiao et al., "SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models," ICML 2023

# Activation Quantization

- (2) Migrate the difficulty to weight
  - Can be scaled using the scaling factors in the LayerNorm

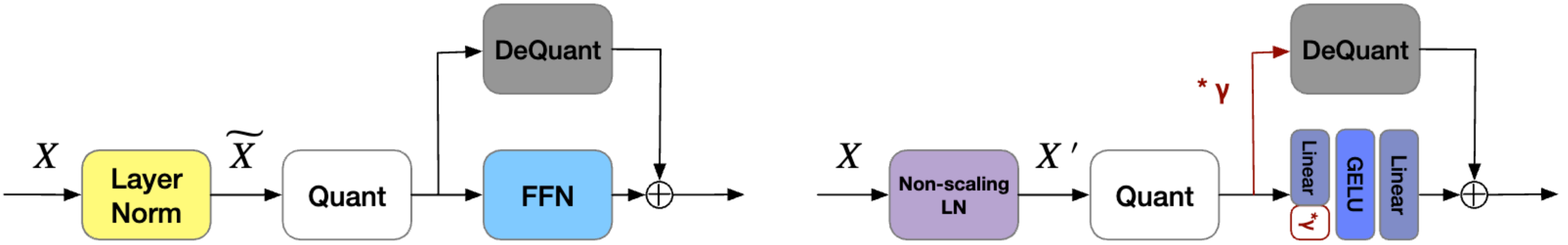
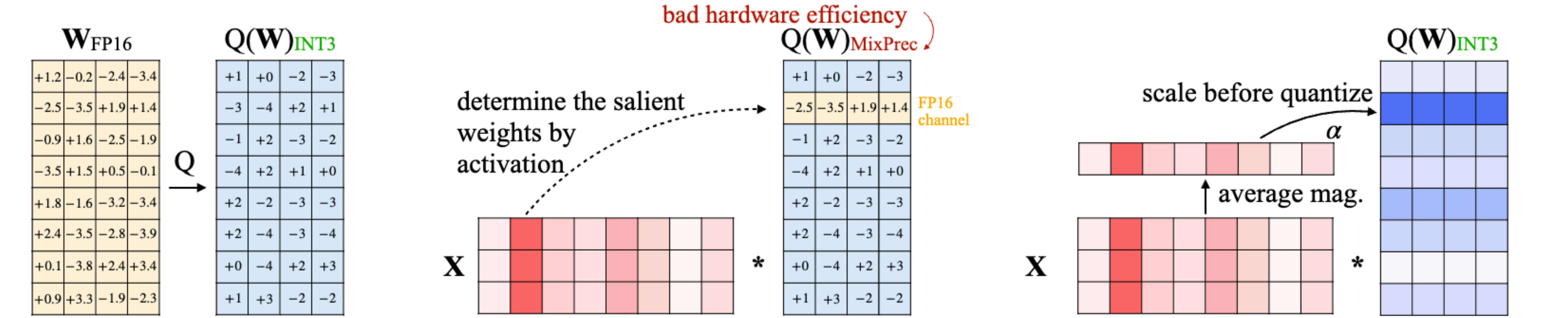



Figure 3: Comparison of the quantization flow before (left) and after (right) Gamma Migration. The original LayerNorm = the Non-scaling LayerNorm * $\gamma$. For other detailed applications such as LayerNorm in encoder-decoder structure, see Fig. 6, Fig. 7.


Wei et al., "Outlier Suppression: Pushing the Limit of Low-bit Transformer Language Models,"NeurIPS 2022
Xiao et al., "SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models," ICML 2023

# Weight Quantization

- Activation distribution helps discover important weights



(a) RTN quantization (**PPL** 43.2) (b) Keep 1% salient weights in FP16 (**PPL** 13.0) (c) Scale the weights before quantization (**PPL** 13.0)

# Weight Pruning

- Activation distribution helps discover important weights

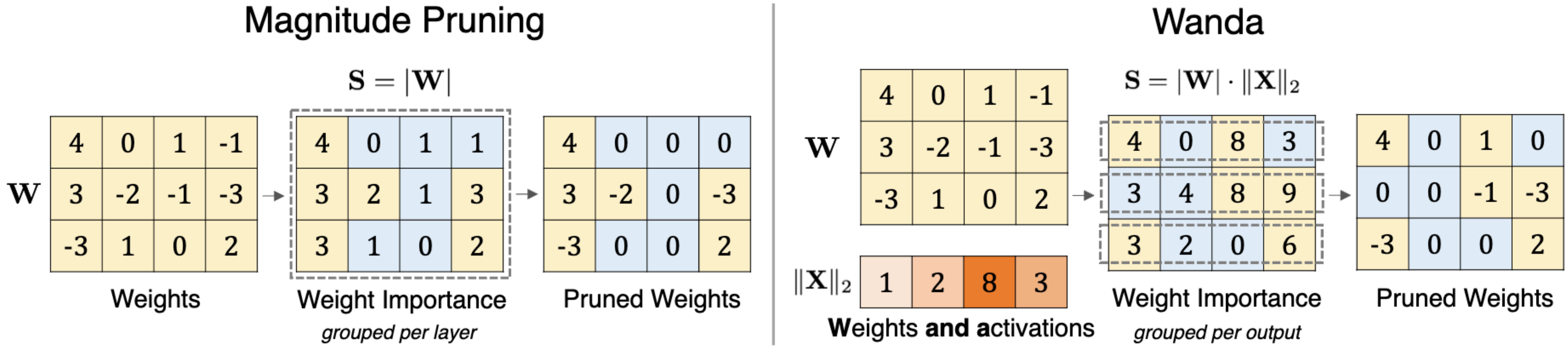
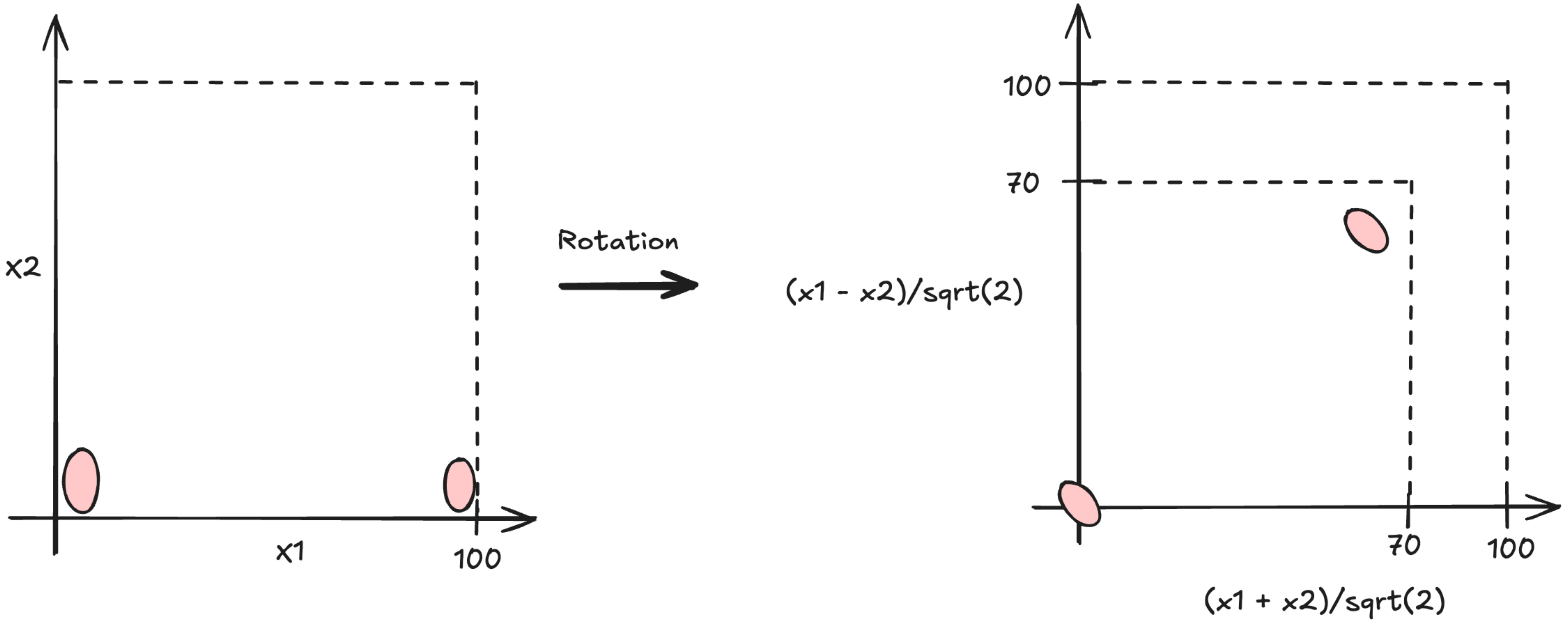



Figure 1: Illustration of our proposed method Wanda (Pruning by **W**eights **and a**ctivations), compared with the magnitude pruning approach. Given a weight matrix $\mathbf{W}$ and input feature activations $\mathbf{X}$, we compute the weight importance as the product between the weight magnitude and the norm of the corrsponding input activations ($|\mathbf{W}| \cdot \|\mathbf{X}\|_2$). Weight importance scores are compared on a *per-output* basis (within each row in $\mathbf{W}$), rather than globally across the entire matrix.

# Rotation

- **Idea.** Mitigate the outliers by multiplying the rotation matrix
  - For some orthogonal rotation matrix $\mathbf{R}$ $(\mathbf{R}^\top\mathbf{R} = \mathbf{R}\mathbf{R}^\top = \mathbf{I},\ \mathbf{R}_{ij} \in \{\pm 1\})$

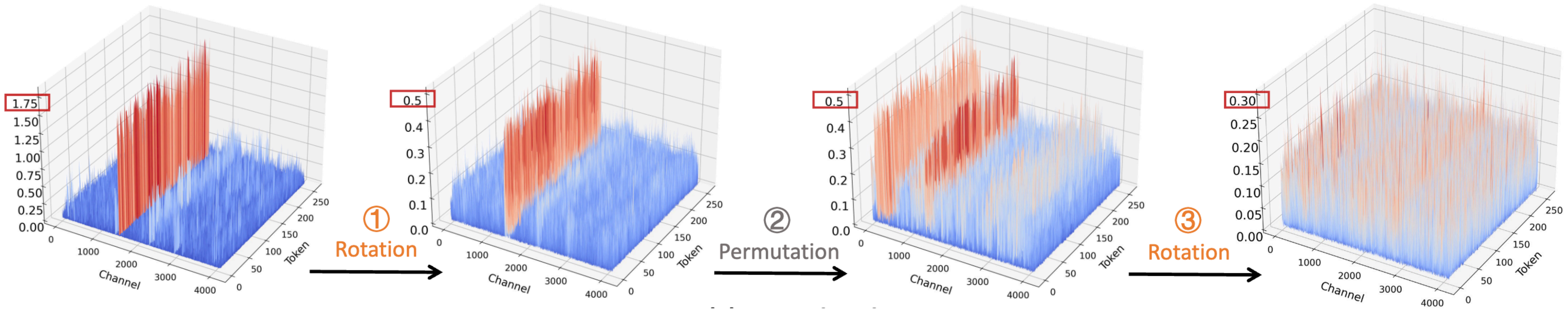
$$\mathbf{WX} = (\mathbf{WR}^\top)(\mathbf{RX})$$




Ashkboos et al., "QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs," ICLR 2024

# Rotation

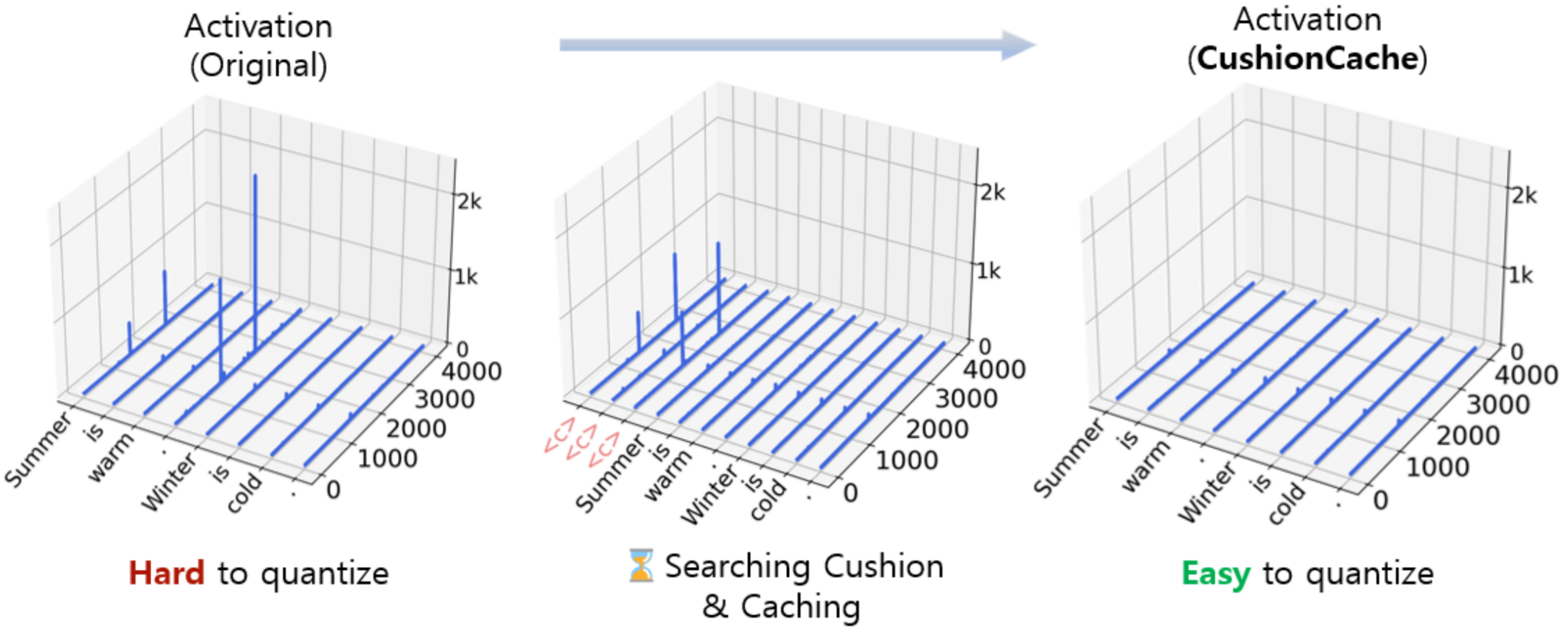
- To get the rotation matrix:
  - Random Hadamard rotation
  - Learned rotation (e.g., SpinQuant, via Cayley SGD)
- Often represented as blockwise rotation + permutation




Ashkboos et al., "QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs," ICLR 2024

# Prefix Tuning

- **Idea.** Add some special "attention sink" tokens as the prompt token
  - These suck up all attention, and mitigates outliers in later tokens
  - Apply further fine-tuning




Son et al., "Prefixing Attention Sinks can Mitigate Activation Outliers for Large Language Model Quantization," EMNLP 2024

# Wrapping up

- LLM compression = Model compression + LLM-specific constraints
    - Much focus on practicality

That’s it for today 🙌