



Lecture 07: Efficiency Strategies for Large Language Models

Notes

- Lab 2 is due this Sunday.
- Think about the project, discuss with me during office hours or after class.
- Midterm
 - Nov 4, in class.
 - Will cover materials up to lecture 8 (Oct 28)
 - More information on the midterm questions next lecture.
 - Will send out a coverage this week.
 - Will post some sample questions next week.

Recap

- Distillation
- Neural architecture search (NAS)
- Low-rank factorization
- Dynamic / Conditional Computing

Topics

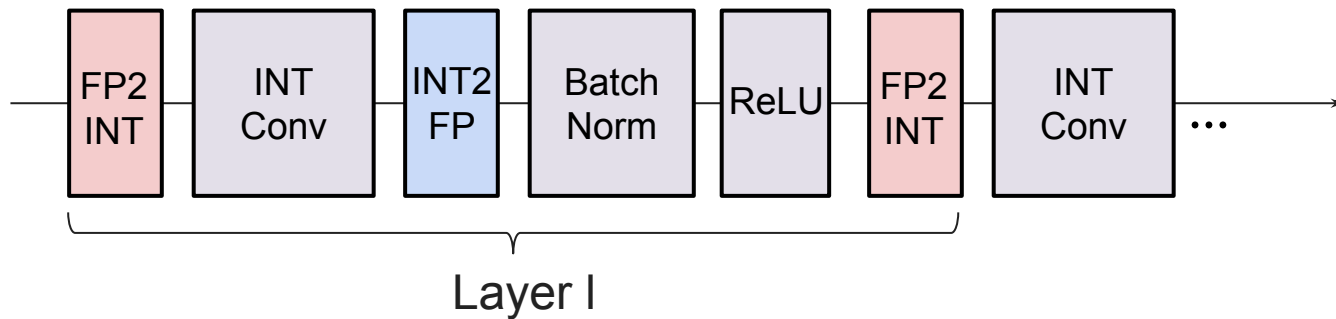
- Large Model Data Distribution Analysis
- Large Model Quantization Strategies
- Large Model Pruning Strategies
- Low-rank Decomposition for LLM

Topics

- Large Model Data Distribution Analysis
- Large Model Quantization Strategies
- Large Model Pruning Strategies
- Low-rank Decomposition for LLM

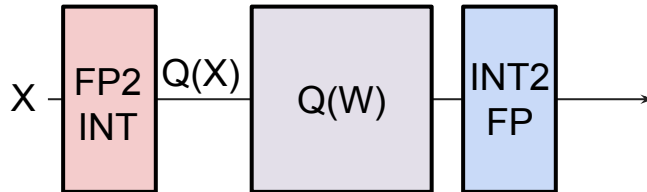
Quantization

- Quantization on activation needs to be performed dynamically. This will introduce additional compute overhead.
- Also the activation will pass the nonlinear functions, which are usually very sensitive to quantization error, so dequantization is required to convert back to FP 16/32.
- For large models with substantial computational demand, even when input activations are dynamically quantized, this approach can still significantly reduce overall processing latency.

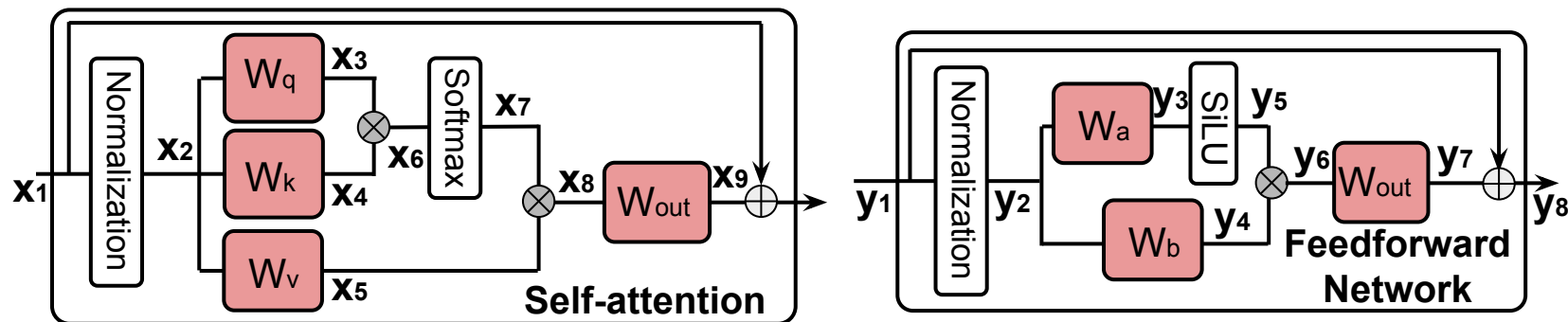


Quantization

- Quantization on activation needs to be performed dynamically. This will introduce additional compute overhead.
- Also the activation will pass the nonlinear functions, which are usually very sensitive to quantization error, so dequantization is required to convert back to FP 16/32.
- For large models with substantial computational demand, even when input activations are dynamically quantized, this approach can still significantly reduce overall processing latency.



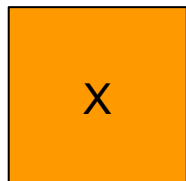
Large Model Quantization



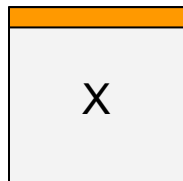
- Quantization is applied to all multiplications highlighted in red.
- Some prior works also quantize the multiplications highlighted in grey. However, these operations typically require more dynamic quantization schemes.

Granularity of Quantization

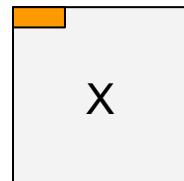
- The activation and weight can be quantized with different granularity:
 - Tensor-based quantization
 - Vector-based quantization
 - Group-based quantization
- A higher quantization granularity will lead to a lower quantization error and a higher hardware implementation cost.



Tensor-based
quantization

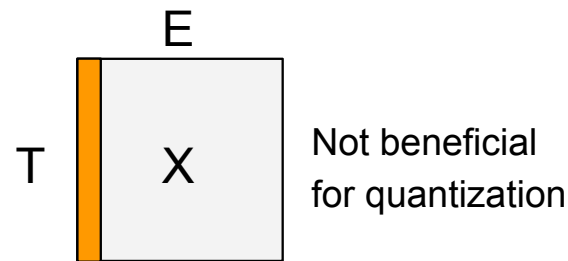
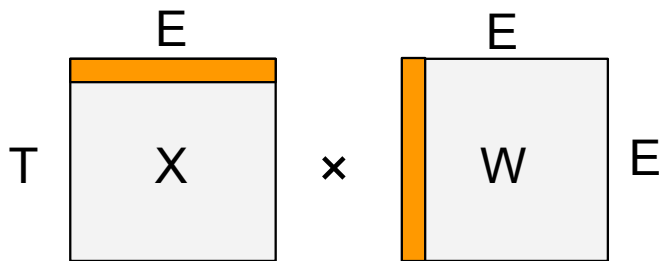


Vector-based
quantization



Group-based
quantization

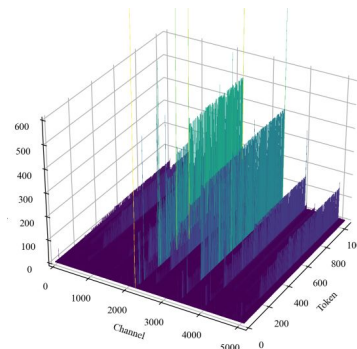
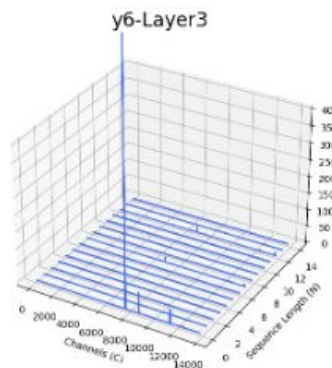
Channelwise Quantization



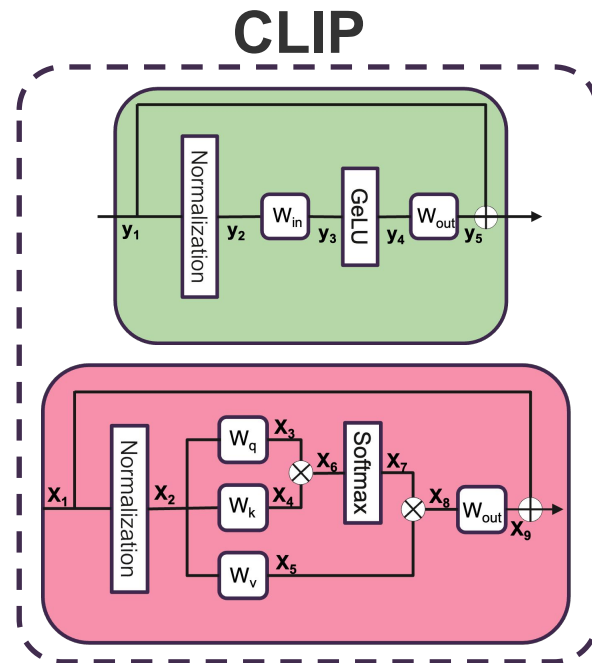
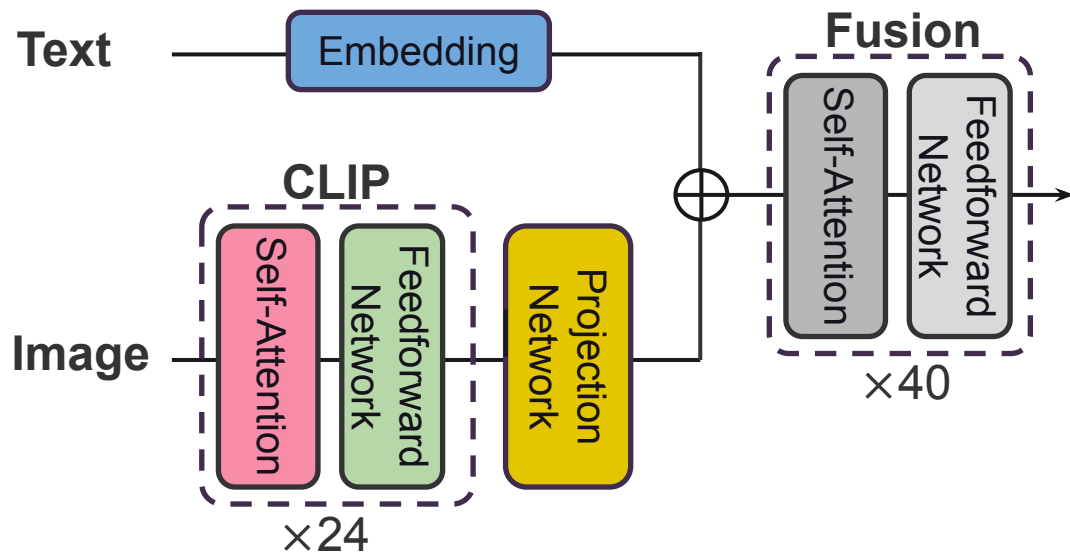
- Tensor-wise and Vector-wise quantization is commonly applied to activations and weights to accelerate LM inference.
- However, the orientation along which quantization is performed is important and needs to be carefully considered.
- Quantizing the input activations is challenging due to the presence of multiple types of outliers.

Types of Outlier

- Massive Activation:
 - For an activation matrix A , an massive activation is an element A_{ij} within it that satisfies:
 - $A_{ij} > \eta \times \text{mean}(|A|)$
 - $A_{ij} > \gamma$
 - For example, $\eta=300$, $\gamma=50$
- Channelwise Outlier:
 - $\text{mean}(A_i) > \eta \times \text{std}(A) + \text{mean}(|A|)$
 - $\text{std}(A_i) < \beta$
 - For example, $\eta=3$, $\beta=0.6$

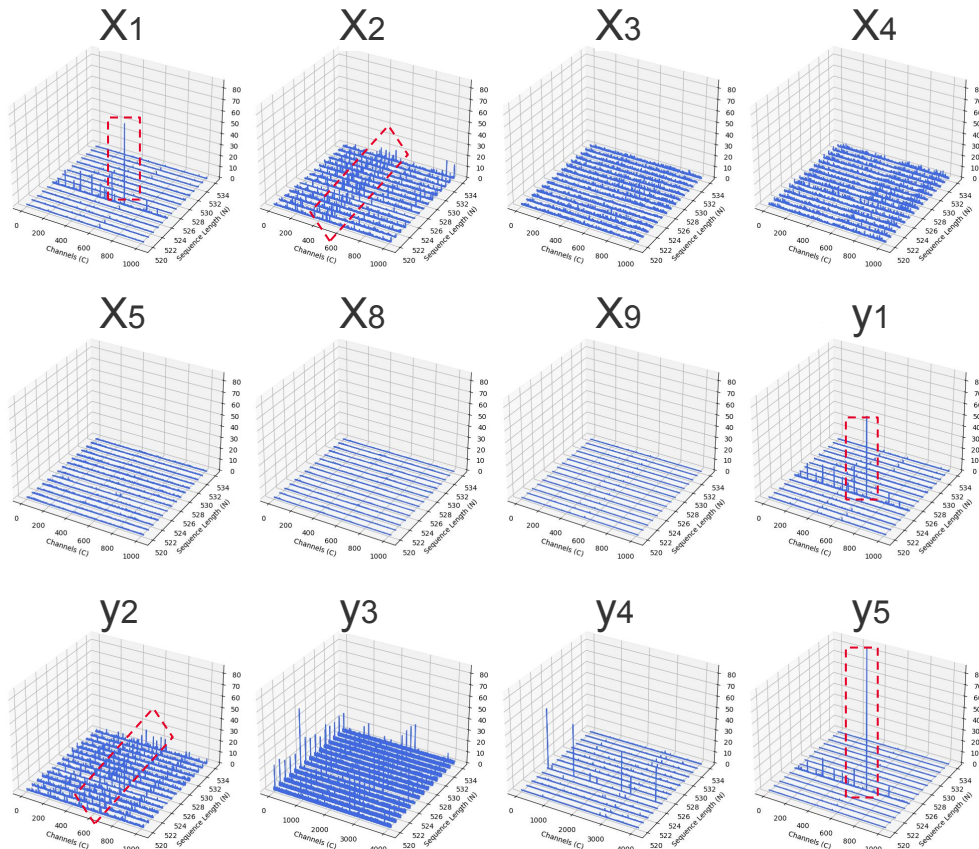
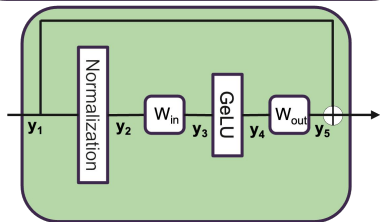
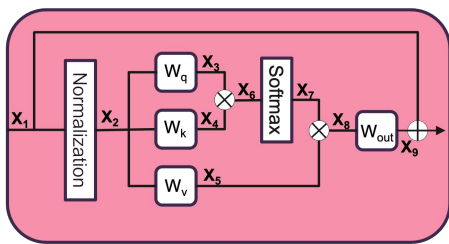


CLIP Model



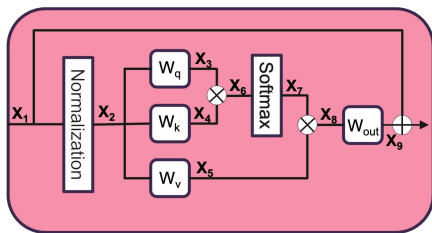
Outlier Study: CLIP Activations

- Outliers with large magnitudes appear at positions x_1 , y_1 , and y_5 , referred to as **massive activations**.



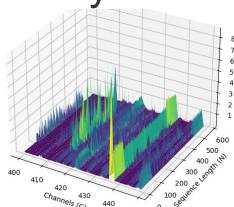
Outlier Study: CLIP Activations

- 3D plots of x_2 across layers.

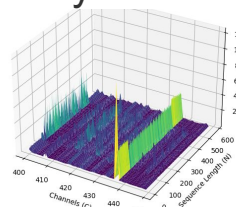


- x_2 exhibits channel wise outlier

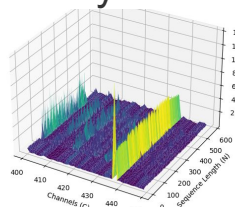
Layer 1



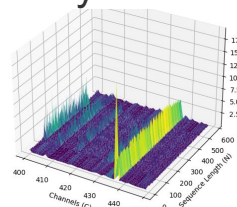
Layer 2



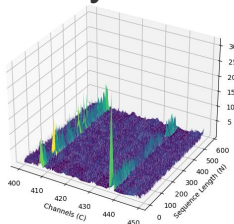
Layer 3



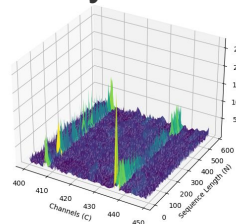
Layer 4



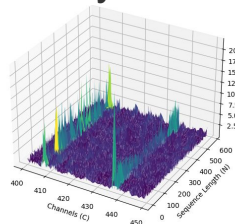
Layer 11



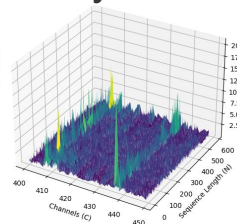
Layer 12



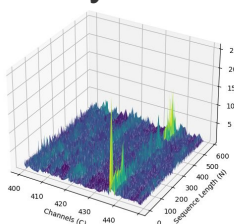
Layer 13



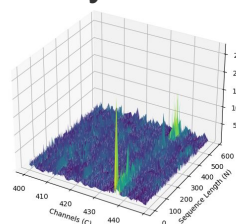
Layer 14



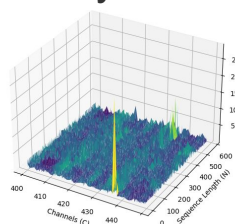
Layer 19



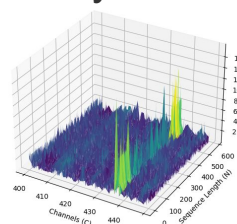
Layer 20



Layer 21

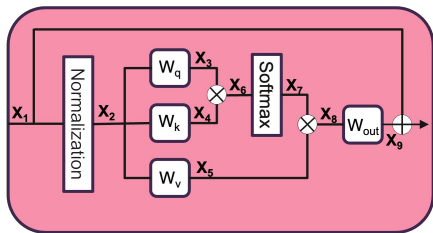


Layer 23

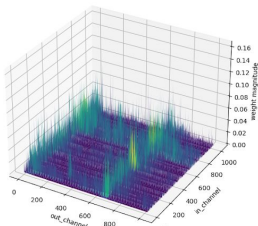


Outlier Study: CLIP Weights

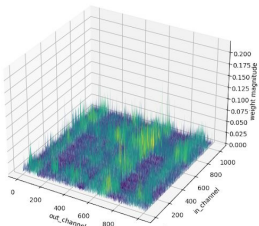
- W_q across CLIP layers.



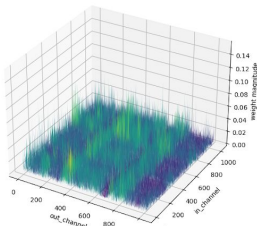
q-weight Layer0



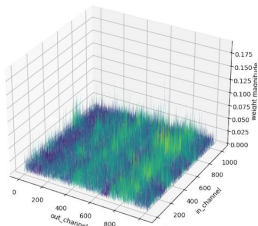
q-weight Layer1



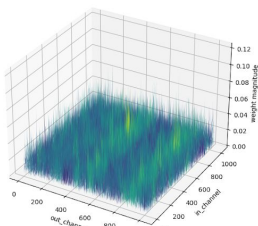
q-weight Layer2



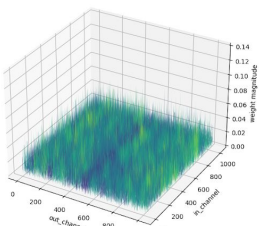
q-weight Layer3



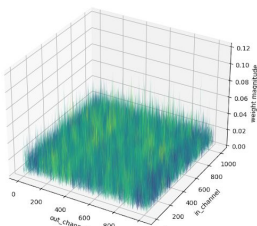
q-weight Layer11



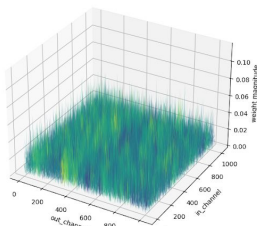
q-weight Layer12



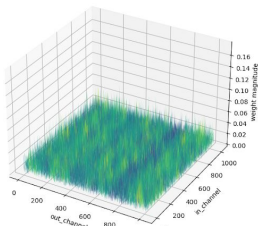
q-weight Layer13



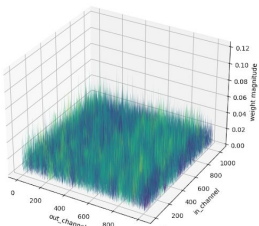
q-weight Layer14



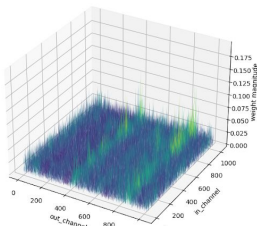
q-weight Layer19



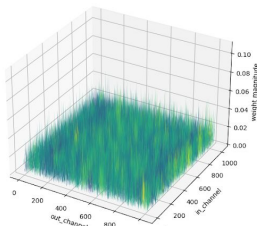
q-weight Layer20



q-weight Layer21

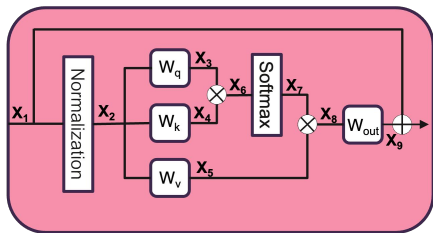


q-weight Layer23

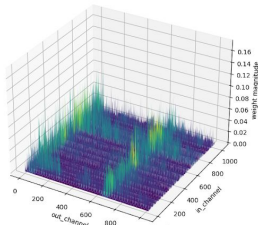


Outlier Study: CLIP Weights

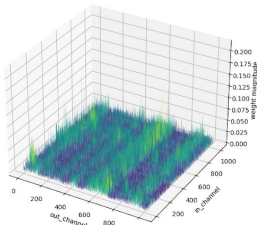
- W_k across CLIP layers.



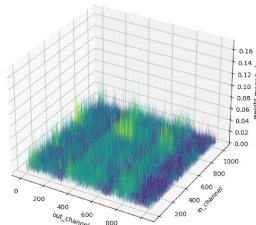
k-weight Layer0



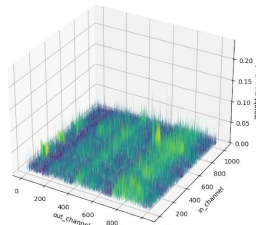
k-weight Layer1



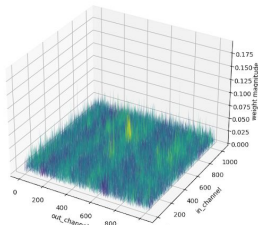
k-weight Layer2



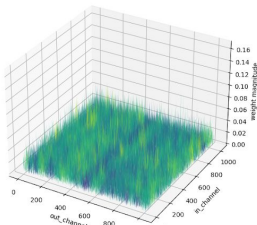
k-weight Layer3



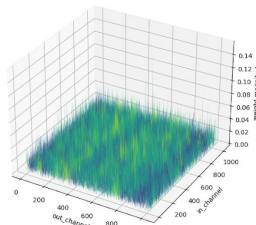
k-weight Layer11



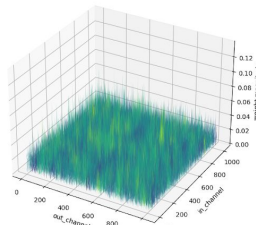
k-weight Layer12



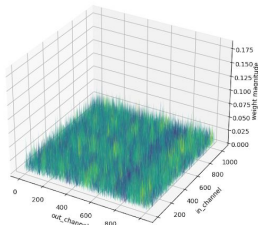
k-weight Layer13



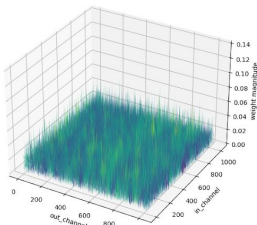
k-weight Layer14



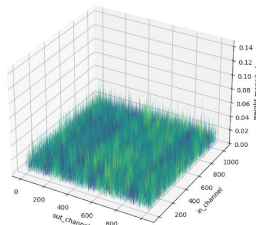
k-weight Layer19



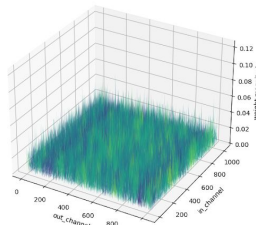
k-weight Layer20



k-weight Layer21

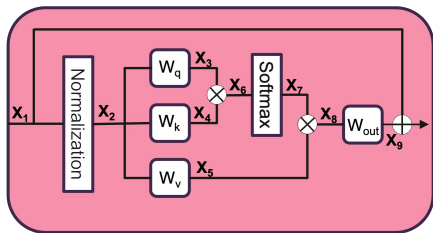


k-weight Layer23

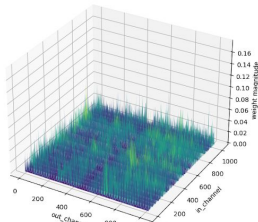


Outlier Study: CLIP Weights

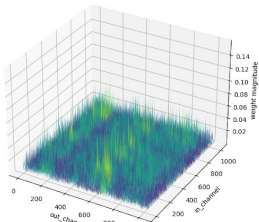
- W_v across CLIP layers.



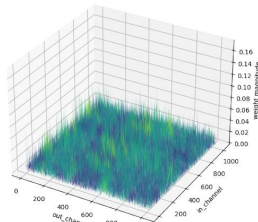
v-weight Layer0



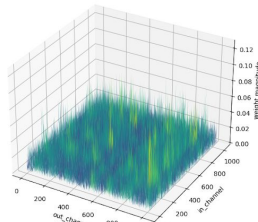
v-weight Layer1



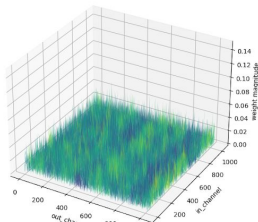
v-weight Layer2



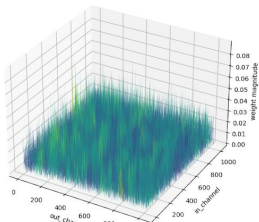
v-weight Layer3



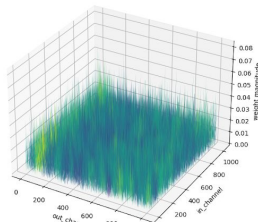
v-weight Layer11



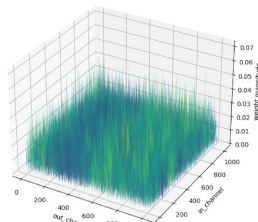
v-weight Layer12



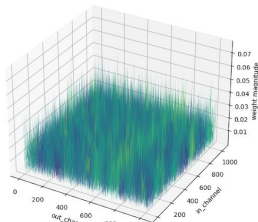
v-weight Layer13



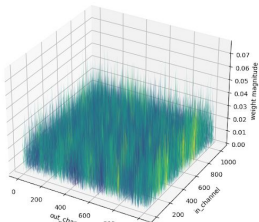
v-weight Layer14



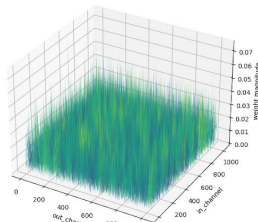
v-weight Layer19



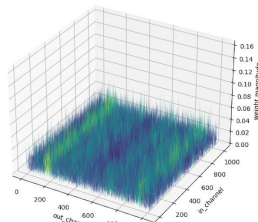
v-weight Layer20



v-weight Layer21

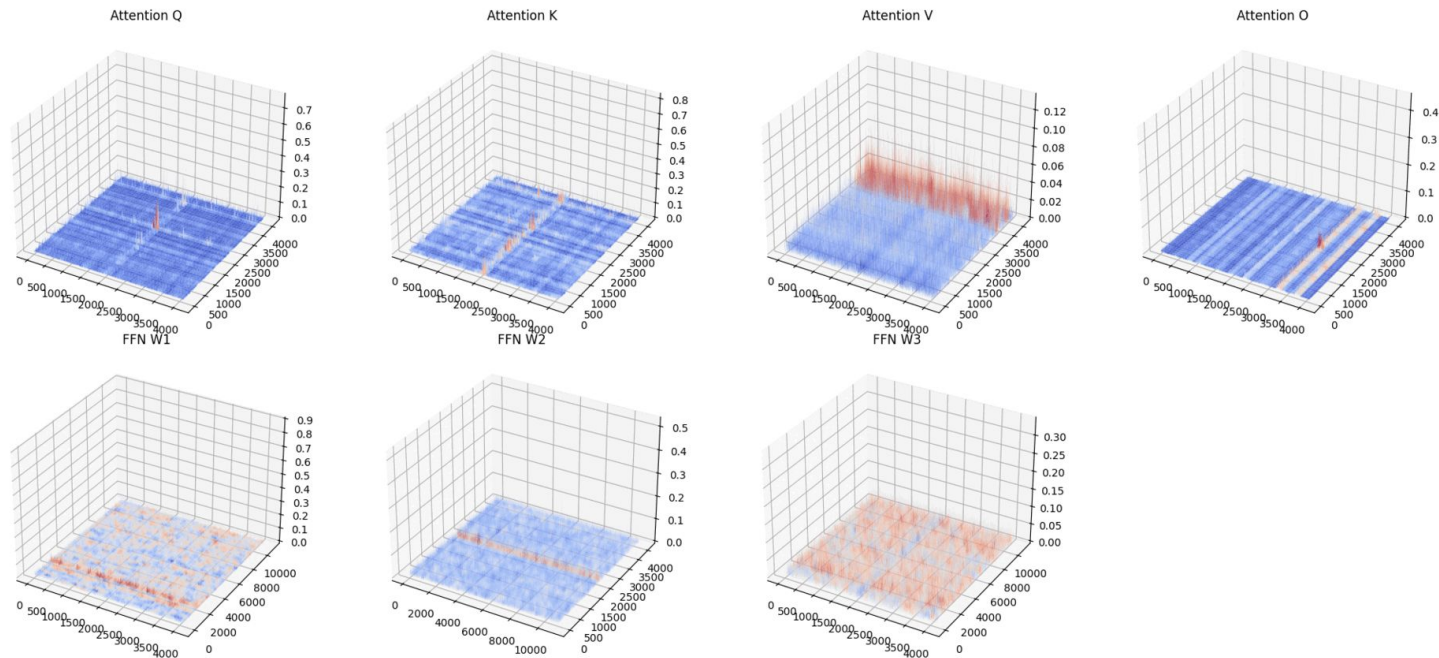


v-weight Layer23



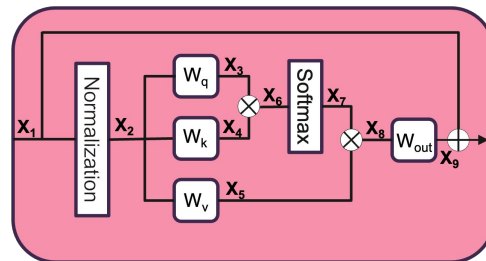
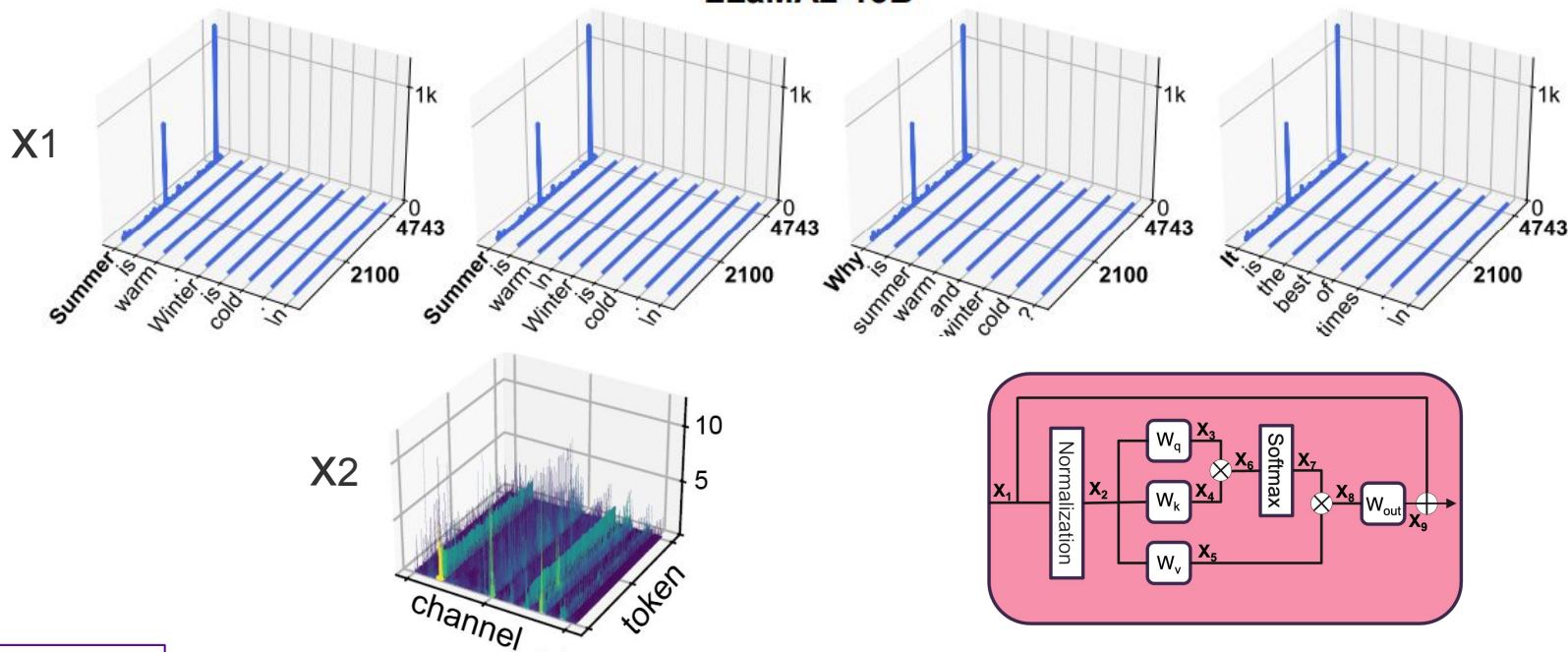
Outlier Study: CLIP Weights

Layer 0 Weights



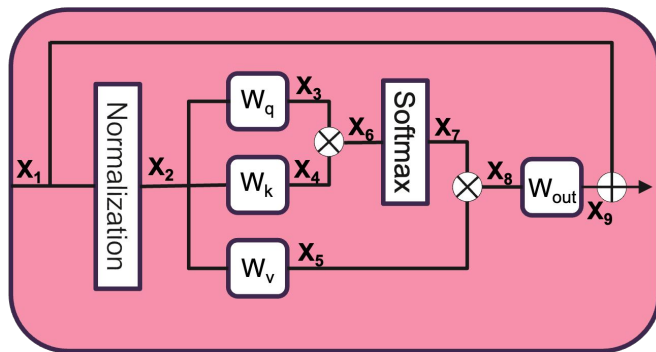
Outlier Study: LLaMA Activations

LLaMA2-13B



Why Massive Activations Exist?

- These massive activations are not random spikes or errors, but rather learned, input-agnostic constants that function as implicit bias terms within the model.
- The paper argues that LLMs use massive activations to inject bias into the self-attention computation. These large, fixed activations essentially allow certain tokens to always attract disproportionately high attention.



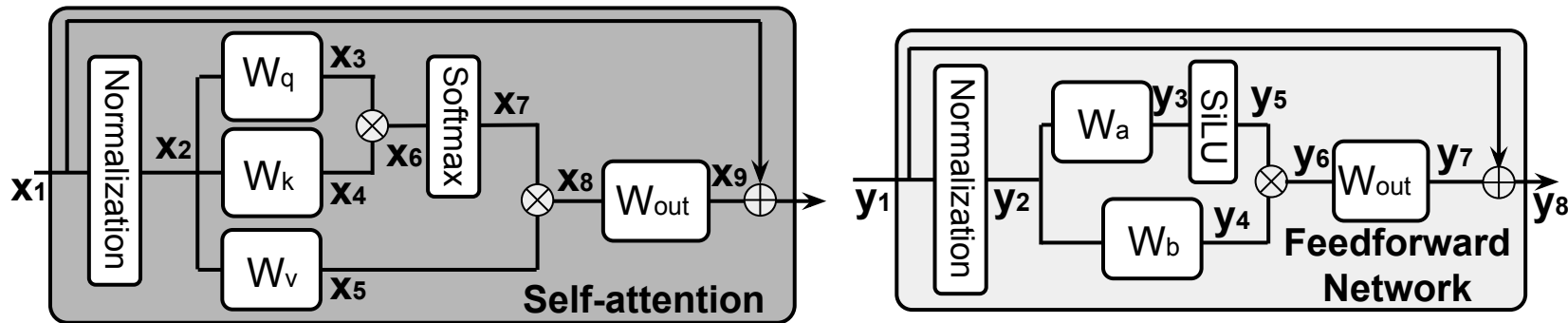
Impact of Massive Activation

Intervention	LLaMA3.2-3B		LLaMA3.1-8B		LLaMA2-13B		GPT-2		Qwen2.5-7B	
	WikiText	C4	WikiText	C4	WikiText	C4	WikiText	C4	WikiText	C4
<i>Original</i>	5.567	10.790	6.941	9.046	4.355	6.405	14.795	19.460	6.520	11.773
<i>TMA to mean at y_7</i>	1124111.75	21046.82	21281.49	1301562.25	1301562.25	6469.42	14.841	19.560	71216.17	66588.86
<i>TMA to zeroes at y_7</i>	1138151.23	21951.41	21601.10	1302018.53	1309211.61	7128.32	14.911	19.928	71835.61	67518.35

- The truncation of massive activation will cause the significant accuracy degradation of the LLM.
- Massive activations also occur in other types of foundation models that utilize attention-based architectures.

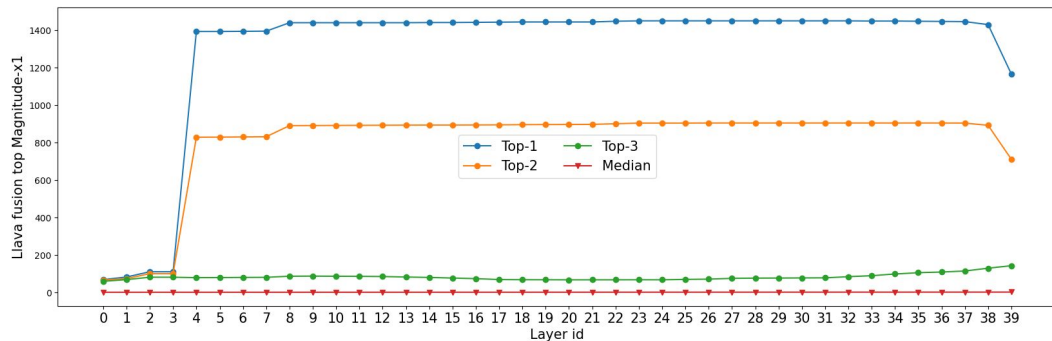
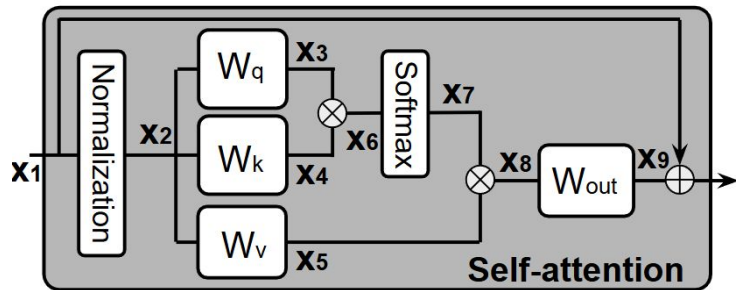
Detailed Study of Massive Activation

- y_7 of layer 3 first produce massive activation (MA), the outlier exists within the first text token.
- x_1 and y_1 in the following layers contain MA, which may reach into the thousands.
- The MA then propagates through residual link across layers.
- Channel wise outliers exists within x_2 , y_2 .



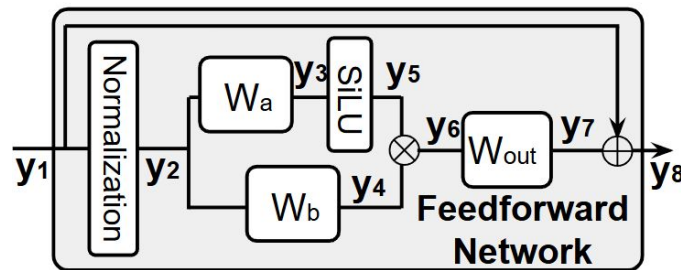
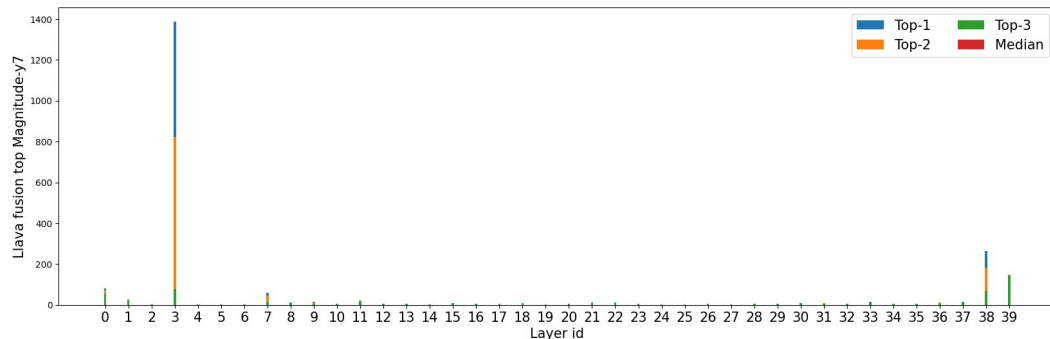
Detailed Study of Massive Activation

- Top magnitude of x1 across layers.



Detailed Study of Massive Activation

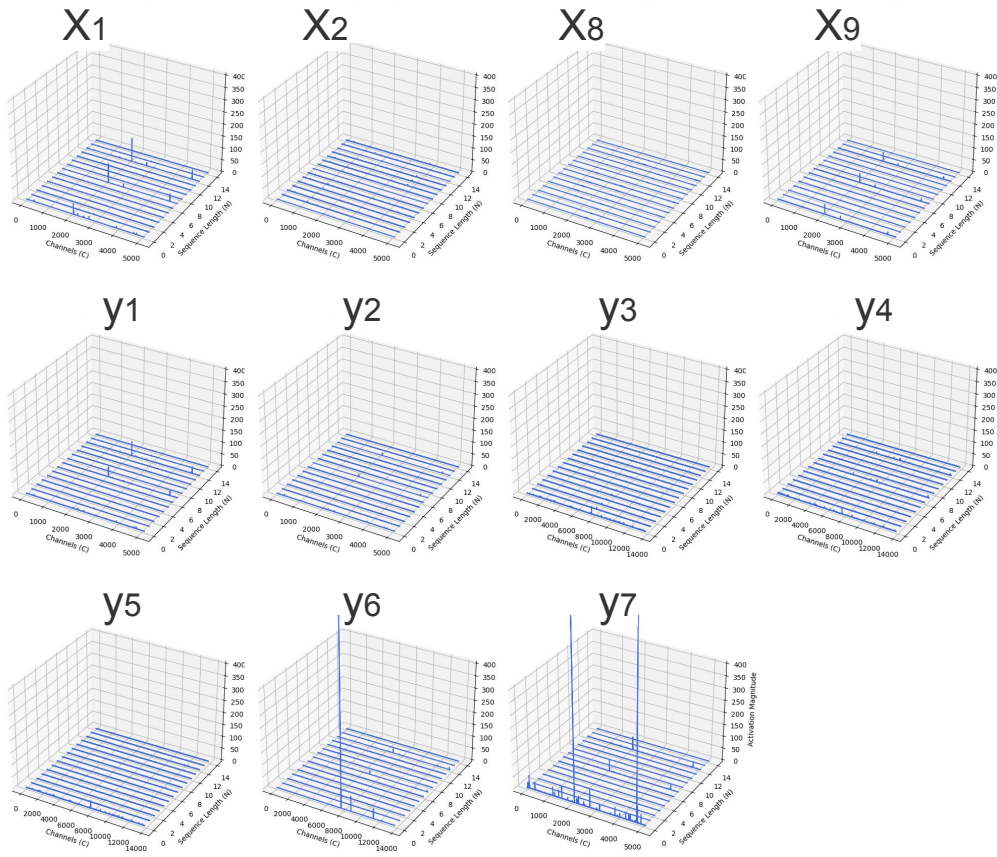
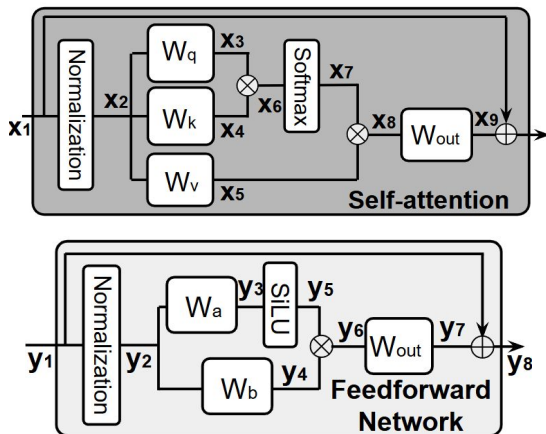
- Top magnitude of y_7 across layers.



- Truncating the massive activation that caused by residual connection will not lead to large accuracy drop.

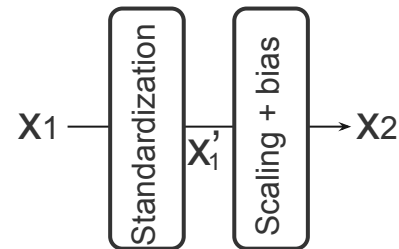
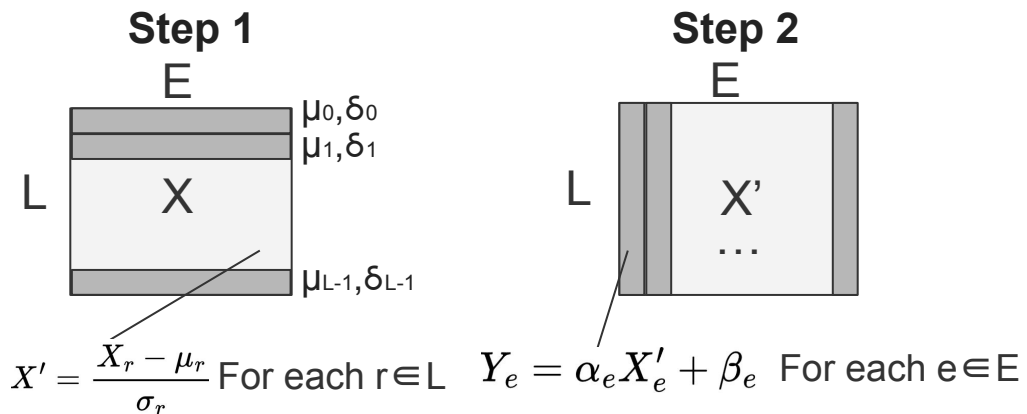
Detailed Study of Massive Activation

- Distribution of activation within layer 3.
- MA first appears in y_6 of token=0.

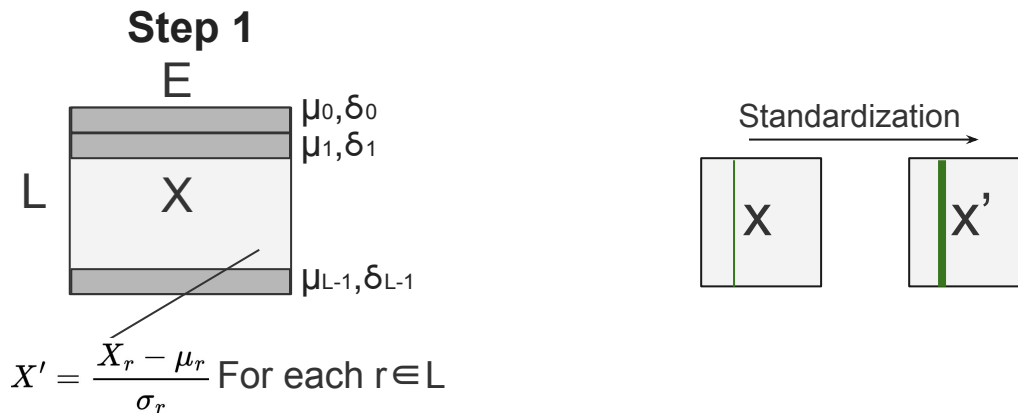


How Channelwise Outlier Forms?

- X1 already has some channelwise outlier, but not significant enough.
- Partial channelwise outlier forms after standardization.
- The scaling process greatly contributes to part of the channelwise outliers.

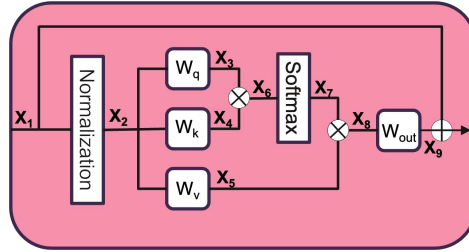


How Channelwise Outlier Forms?

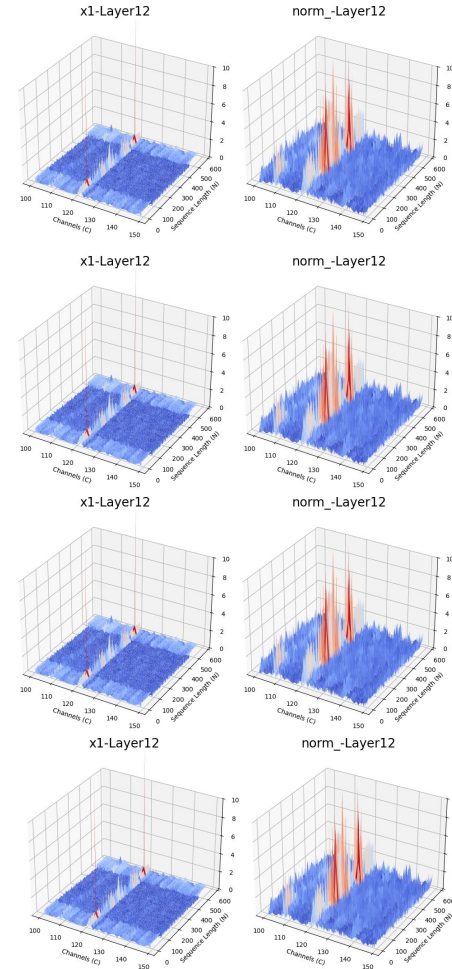


- When the standard deviation is low, channel-wise outliers become more pronounced.

How Channelwise Outlier Forms?

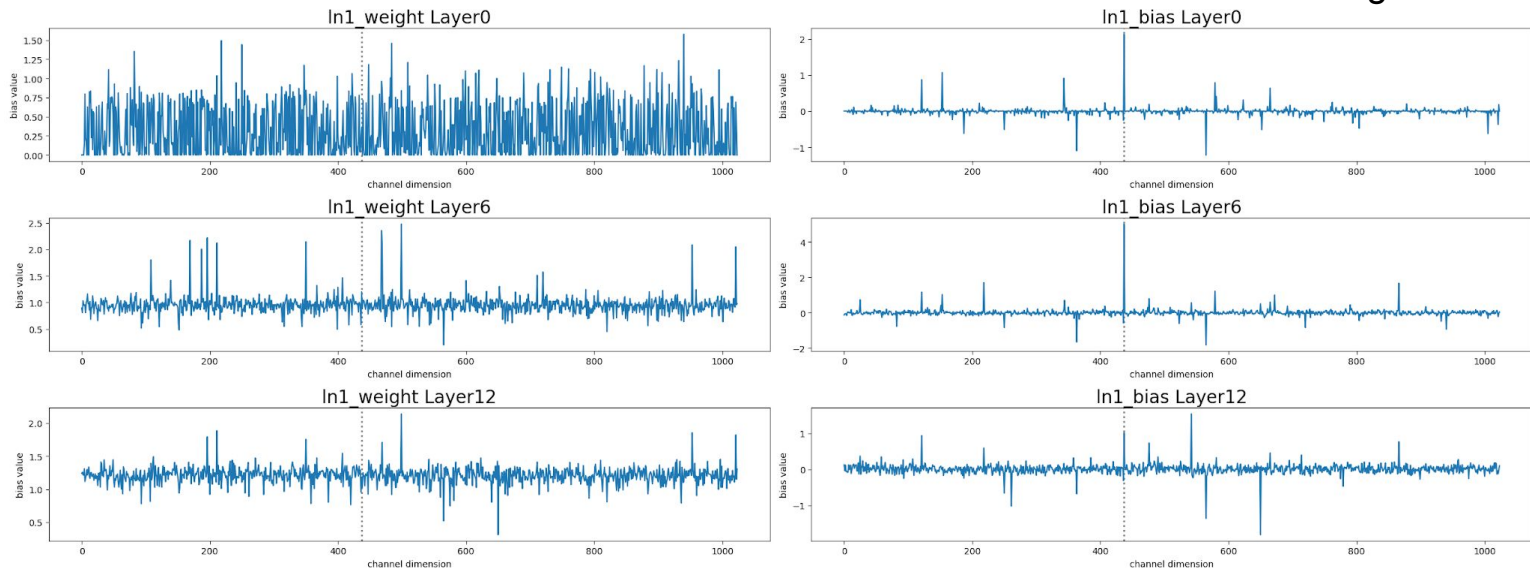


- x_1 typically already contains some channelwise outliers.
- Following standardization, the outlier's magnitude becomes more pronounced.



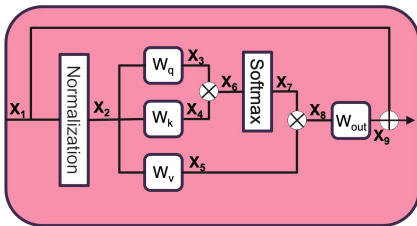
How Channelwise Outlier Forms?

Profiling on CLIP

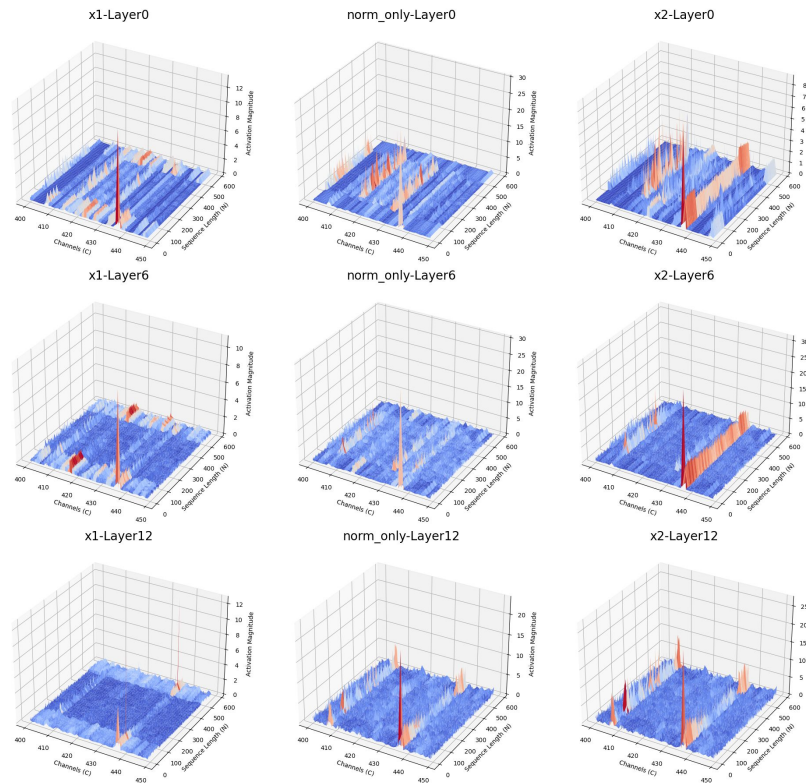


- The scaling and bias factors within the normalization layer also contribute to the channelwise outlier.

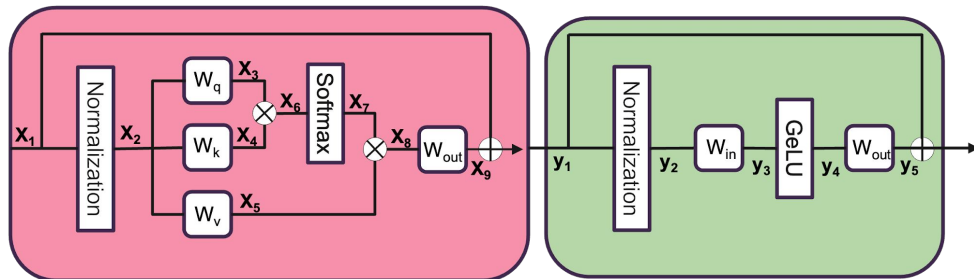
How Channelwise Outlier Forms?



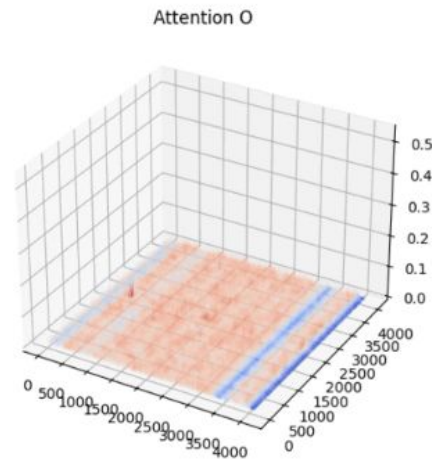
- The scaling and bias factors within the normalization layer also contribute to the channelwise outlier.



How Channelwise Outlier Forms?



- The question then arises: why does x_1 already exhibit some channelwise outliers?
- This is attributed to W_{out} , which results in y_5 having some minor channelwise outliers.
- Additionally, the residual link carries intermediate activations with channelwise outliers, which are also generated by the W_{out} from preceding layers.

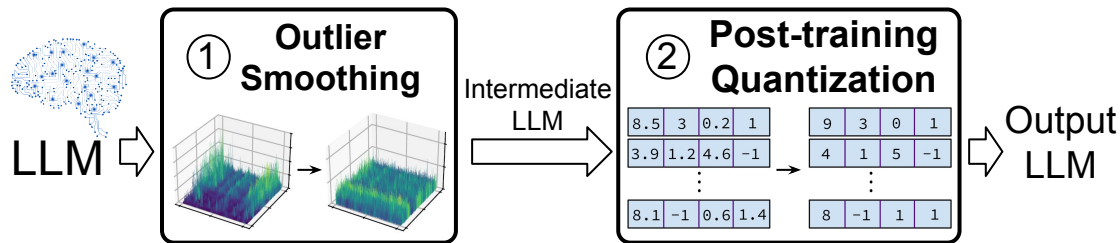


Impact of Channelwise Quantization



- The presence of channel-wise outliers can significantly increase quantization error and substantially degrade the accuracy of the quantized model.
- This effect becomes more pronounced as the bit width decreases.

Quantization Strategy for LMs



- When performing post-training quantization on a LLM, it's common to include a step of outlier smoothing prior to the quantization process.

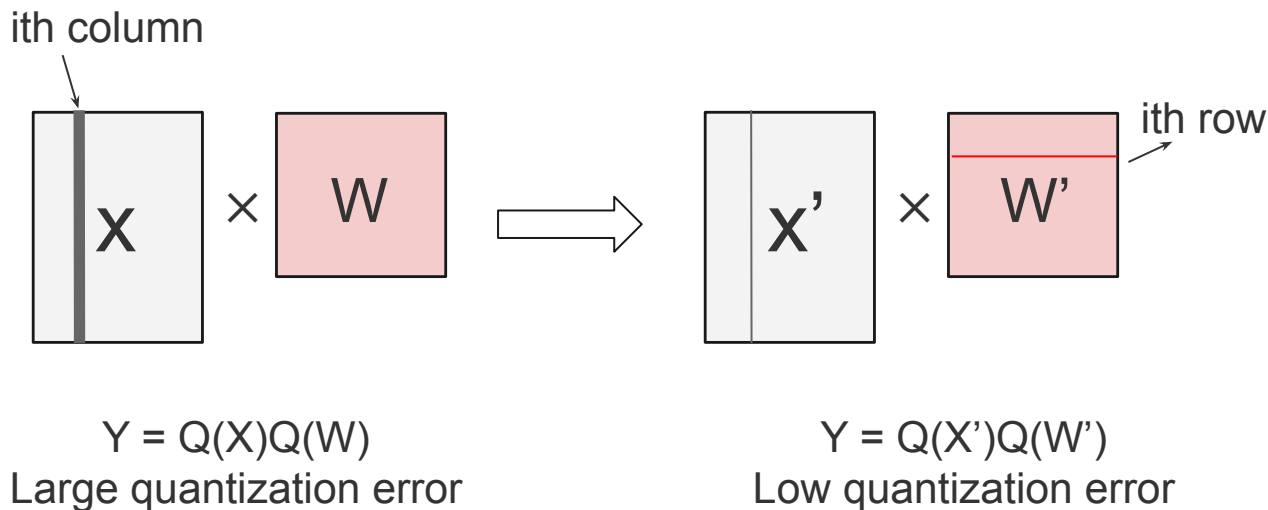
Topics

- Large Model Data Distribution Analysis
- Large Model Quantization Strategies
- Large Model Pruning Strategies
- Low-rank Decomposition for LLM

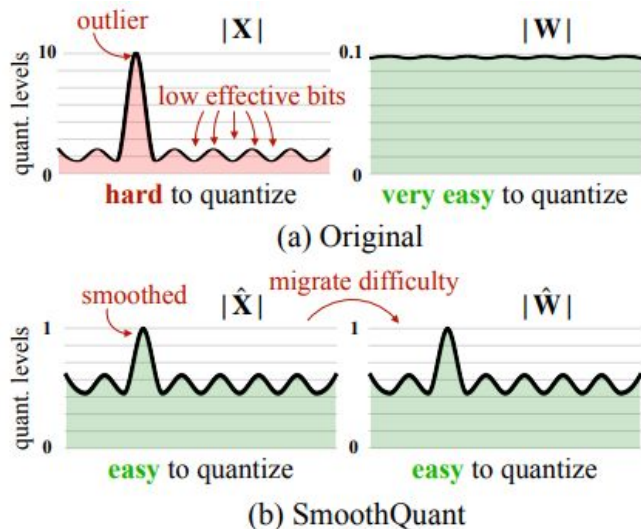
Topics

- Large Model Data Distribution Analysis
- Large Model Quantization Strategies
 - Smoothing Techniques
 - Quantization Techniques
- Large Model Pruning Strategies
- Low-rank Decomposition for LLM

SmoothQuant



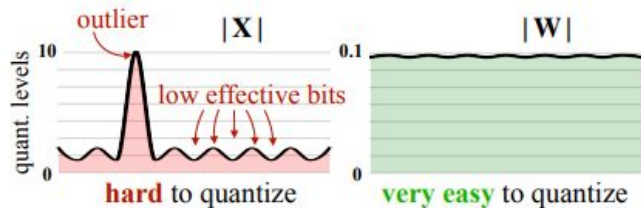
SmoothQuant



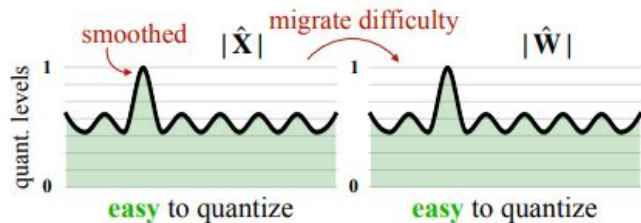
- The intermediate results within LLM usually have a lot of outliers.
- SmoothQuant smooths the activation outliers by offline migrating the quantization difficulty from activations to weights with a mathematically equivalent transformation.

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

SmoothQuant



(a) Original



(b) SmoothQuant

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

- α is a hyperparameter.

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

- α is determined from the calibration dataset.

Activation-Aware Weight Quantization (AWQ)

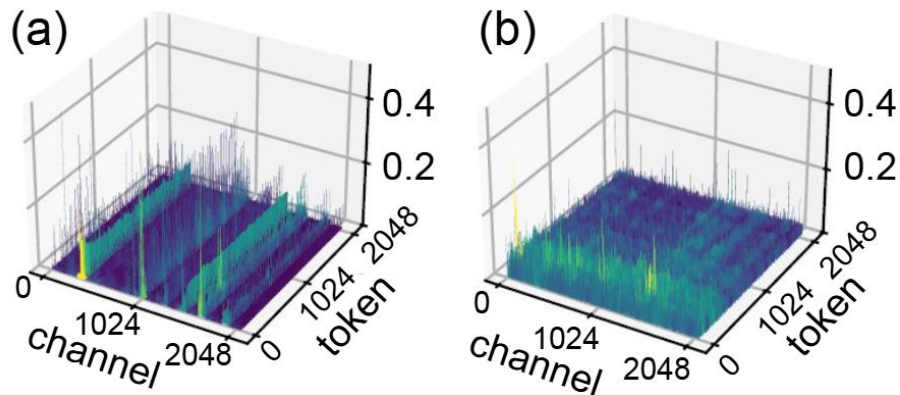
$$\mathbf{s}^* = \arg \min_{\mathbf{s}} \mathcal{L}(\mathbf{s})$$

$$\mathcal{L}(\mathbf{s}) = \|Q(\mathbf{W} \cdot \text{diag}(\mathbf{s}))(\text{diag}(\mathbf{s})^{-1} \cdot \mathbf{X}) - \mathbf{W}\mathbf{X}\|$$

- AWQ improves the performance of smoothquant by making “s” learnable.

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

QuaRot



- QuaRot introduces a novel methods to convert the weights and activation of LLM.
- After conversion, most of the outliers within the activation and weights are removed.
- This conversion introduces almost no additional cost during the inference.

QuaRot

- Assume $Y = AW$, where A may have outliers, quantizing A and W as $Q(A)$ and $Q(W)$ could result in increased quantization error. Consequently, $Q(A)Q(W)$ may differ significantly from AW .
- With QuaRot, a orthogonal matrix is applied to eliminate the outliers within A .

$$A \longrightarrow \boxed{W} \longrightarrow AW$$

$$AR \longrightarrow \boxed{R^T W} \longrightarrow AW$$

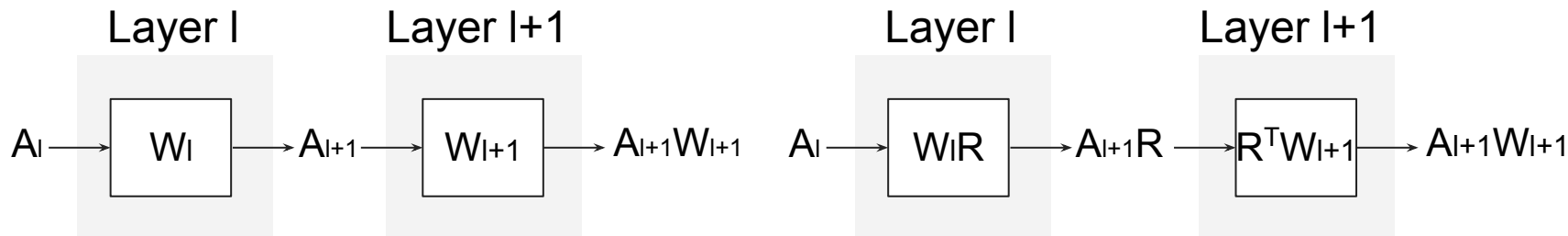
$$R^T R = R R^T = I$$

$$Q(A) \longrightarrow \boxed{Q(W)} \longrightarrow Q(A)Q(W)$$

$$Q(AR) \longrightarrow \boxed{Q(R^T W)} \longrightarrow Q(AR)Q(R^T W)$$

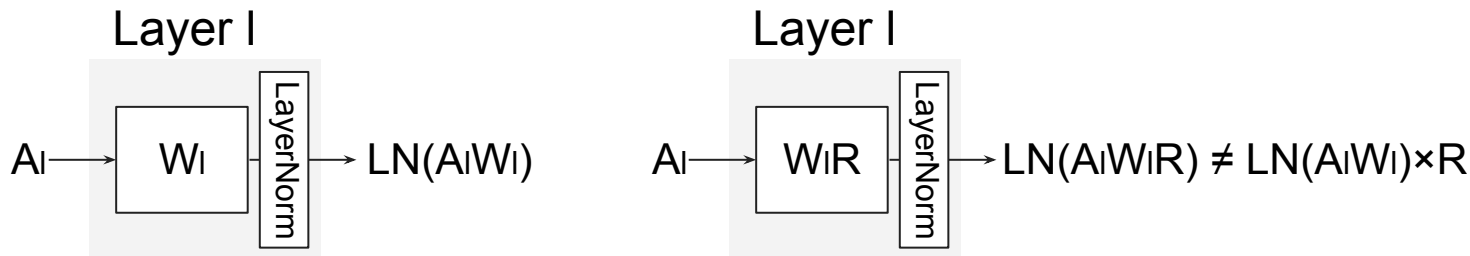
- $R^T W$ can be computed offline, AR can be generated by modifying the weight matrices of the last layer.

QuaRot



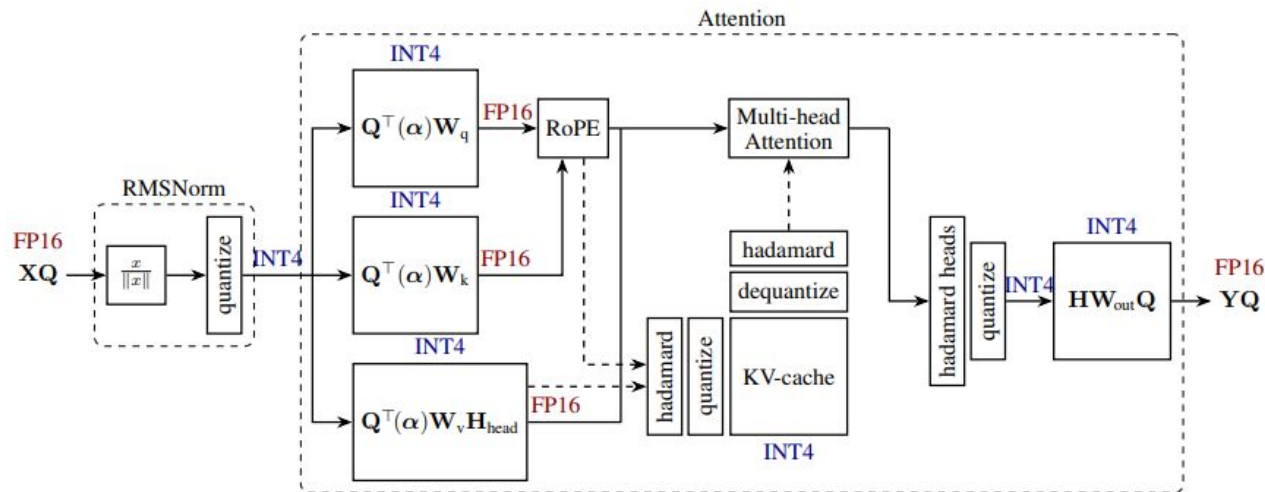
- $R^T W$ can be computed offline, AR can be generated by modifying the weight matrices of the last layer.

QuaRot



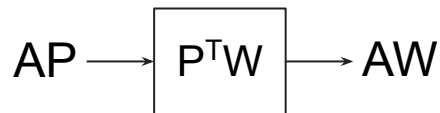
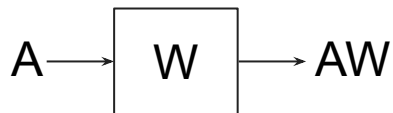
- However, if a nonlinear function follows the linear layers, the Hadamard transform must be computed dynamically.

QuaRot



- For some of the layers, the conversion needs to be performed online.
- We can use Hadamard matrix, which consists of only 1 and -1 to facilitate the matrix multiplications.

DuQuant



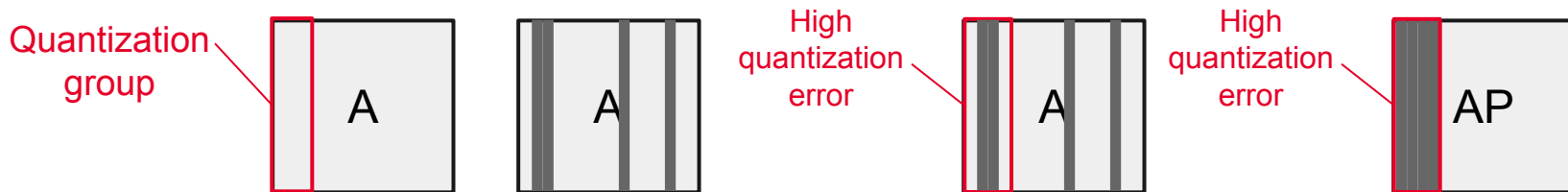
$$P^T P = P P^T = I$$

- Permutation matrix is another types of matrix which cause no change on the output.
- Duquant combines three types of computational invariance operation, including the scalar-based, rotation-based and permutation-based operations for mitigating the outliers.

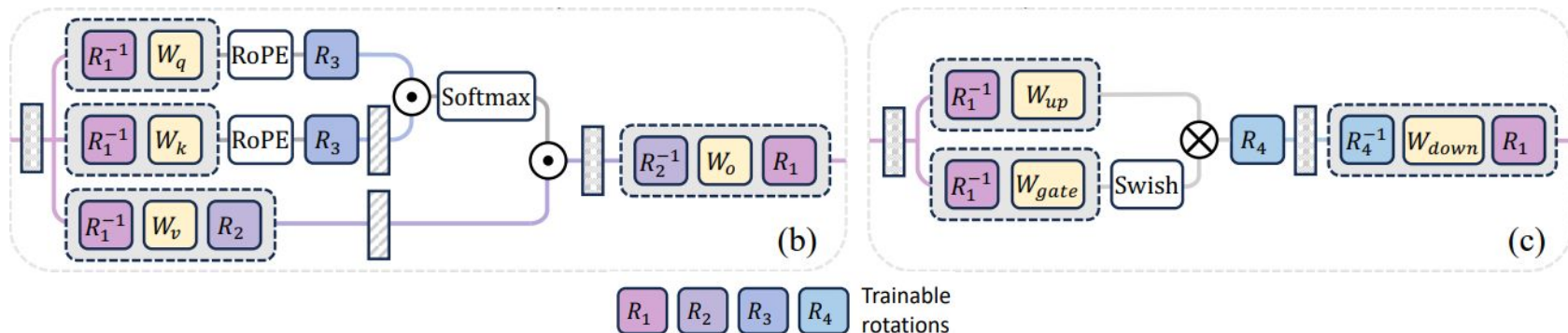
DuQuant

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad A' = AP = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{bmatrix}$$

$$P^T = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad A_{\text{recovered}} = A'P^T = \begin{bmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$



SpinQuant



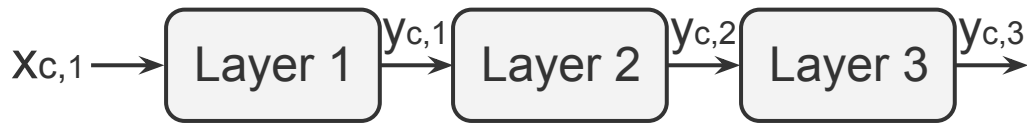
$$\arg \min_{R \in \mathcal{M}} \mathcal{L}_Q(R_1, R_2 \mid W, X)$$

- SpinQuant optimizes (or learns) the rotation matrices to obtain the minimal changes on the training loss.
- We have to ensure the rotational matrix still satisfies the orthogonal property $\rightarrow$ Cayley Optimization.

Topics

- Large Model Data Distribution Analysis
- Large Model Quantization Strategies
 - Smoothing Techniques
 - Quantization Techniques
- Large Model Pruning Strategies
- Low-rank Decomposition for LLM

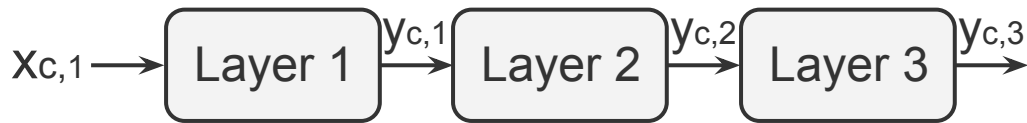
Parallel Update



$$\min_{\theta_1} \sum_{x_{c,1} \in D} \|y_{c,1} - F_{\theta_1}^1(x_{c,1})\|^2$$

- We use a calibration dataset and profile some data $x_c, y_{c,1}, y_{c,2}, y_{c,3}$.
- After that, we use these data to train the optimal quantized parameters.

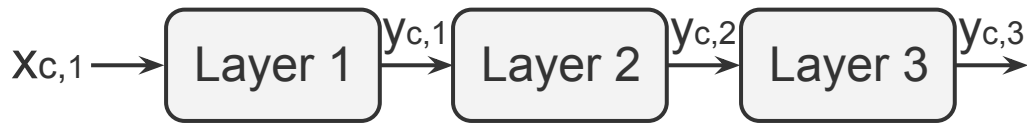
Sequential Update



$$\min_{\theta_1} \sum_{x_{c,1} \in D} \|y_{c,1} - F_{\theta_1}^1(x_{c,1})\|^2$$

- We use a calibration dataset and profile some data $x_c, y_{c,1}$ on the first layer.
- After that, we use these data to train the optimal quantized parameters for the first layer.

Sequential Update



$$\min_{\theta_2} \sum_{x_{c,2} \in D} \|y_{c,2} - F_{\theta_2}^2(x_{c,2})\|^2$$

$$x_{c,2} = F_{\theta_1^*}^1(x_{c,1})$$

- After that, we regenerate the calibration dataset using the new weight values in layer 1, results in $(x_{c,2}, y_{c,2})$.
- Following this, we use these data to train the optimal quantized parameters for the second layer.
- Keep doing this until the final layer.

AdaQuant

$$\begin{aligned} & \left(\hat{\Delta}_w, \hat{\Delta}_x, \hat{V} \right) = \\ & \arg \min_{\Delta_w, \Delta_x, V} \|WX - Q_{\Delta_w}(W')Q_{\Delta_x}(X)\|^2 \end{aligned}$$

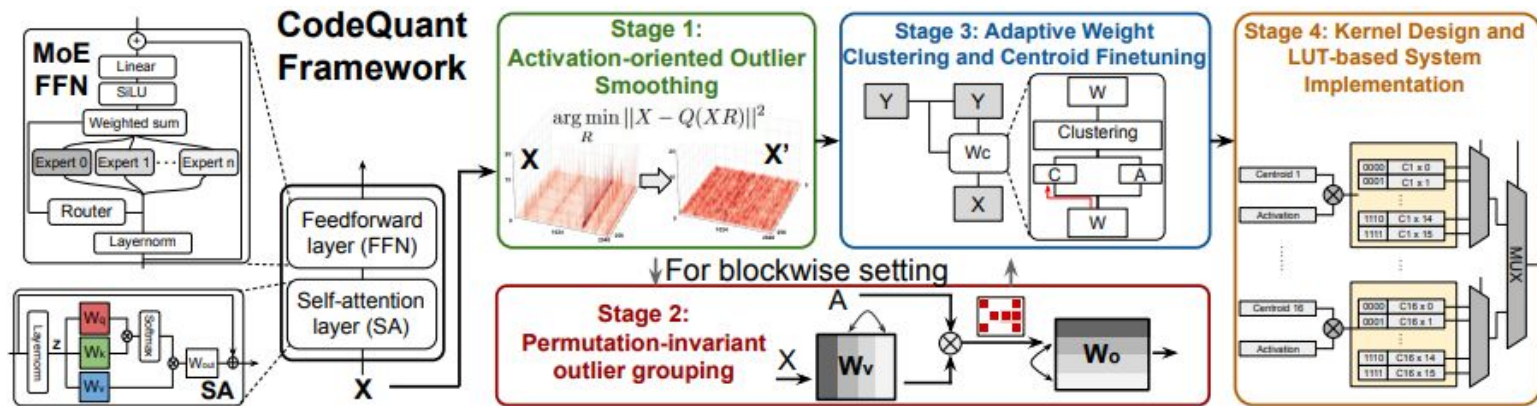
AdaQuant

- A small calibration dataset is used to find the optimal scale and V.

$$\begin{aligned} & \left(\hat{\Delta}_{w_l}, \hat{\Delta}_{x_l}, \hat{V}_l \right) \\ & = \arg \min_{\Delta_{w_l}, \Delta_{x_l}, V_l} \|W_l X_l - Q_{\Delta_{w_l}}(W'_l) \cdot Q_{\Delta_{x_l}}(X_l^q)\|^2 \\ & X_q = \sigma(Q_{\Delta_{w_{l-1}}}(W_{l-1} + V_{l-1}) \cdot Q_{\Delta_{x_l}}(X_{l-1}^q)), \end{aligned}$$

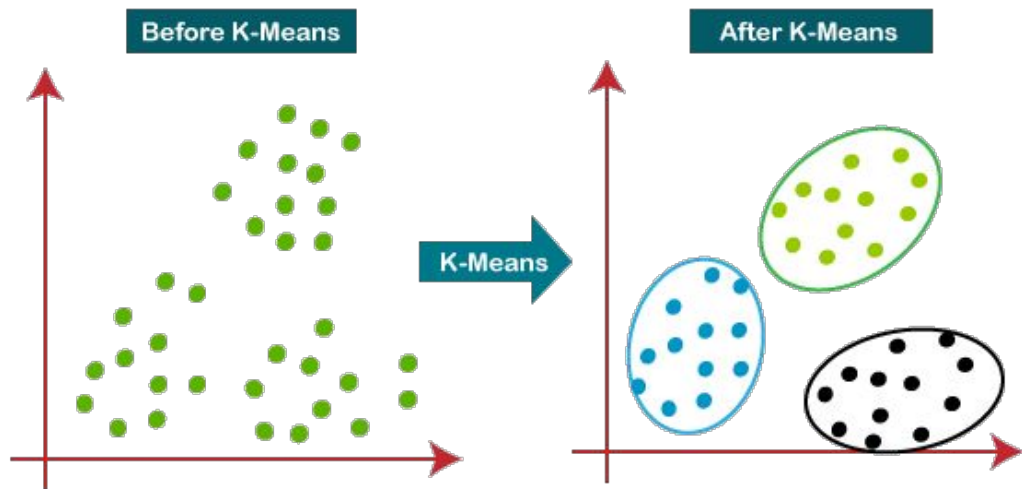
Sequential AdaQuant

CodeQuant



- Compared to quantization, the clustering operation exhibits greater tolerance to outliers.

LUT-based LLM Inference



- Input of 4 bits, weight with K centroids.

Input centroid	Weight centroid	Results
a1	w1	0011
a2	w1	0010
...	...	...
ak	wk	1010

- The multiplication and addition operations can be performed using lookup table (LUT).

Advantage of Clustering

Clustering

Original

10.4	1.2	8.7	0.6	9.9	2.2	1.8	12.0
------	-----	-----	-----	-----	-----	-----	------

Clustered

10.25	1.45	10.25	1.45	10.25	1.45	1.45	10.25
-------	------	-------	------	-------	------	------	-------

😊 Error = 0.75 (Low)

Quantization

Original

10.4	1.2	8.7	0.6	9.9	2.2	1.8	12.0
------	-----	-----	-----	-----	-----	-----	------

Quantized

12.0	0.6	12.0	0.6	12.0	0.6	0.6	12.0
------	-----	------	-----	------	-----	-----	------

😞 Error = 1.3 (High)

- Clustering can be used to handle excessive outliers.

CodeQuant

- How can we find the optimal assignment and centroids to minimize the output change?

Clustering

Original

10.4	1.2	8.7	0.6	9.9	2.2	1.8	12.0
------	-----	-----	-----	-----	-----	-----	------

 W

Clustered

10.25	1.45	10.25	1.45	10.25	1.45	1.45	10.25
-------	------	-------	------	-------	------	------	-------

 W'

$$W' = AC$$

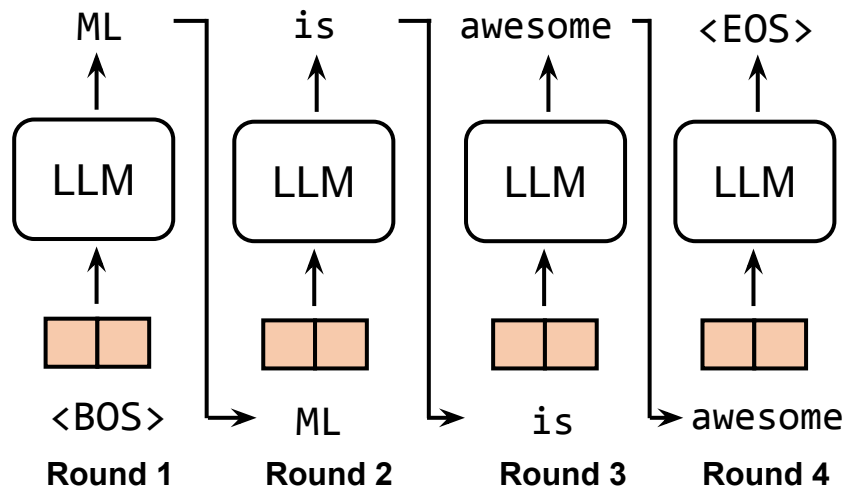
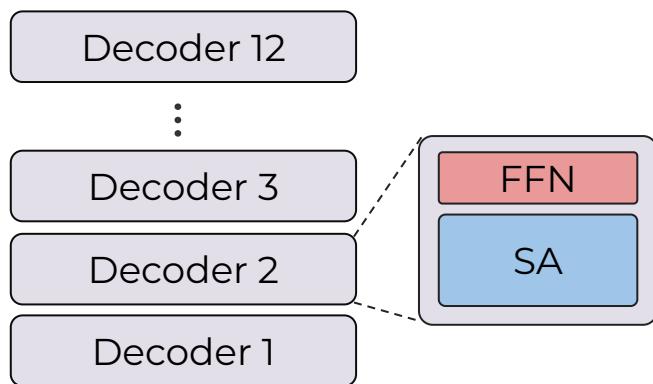
$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$C = [10.25 \ 1.45]$$

Topics

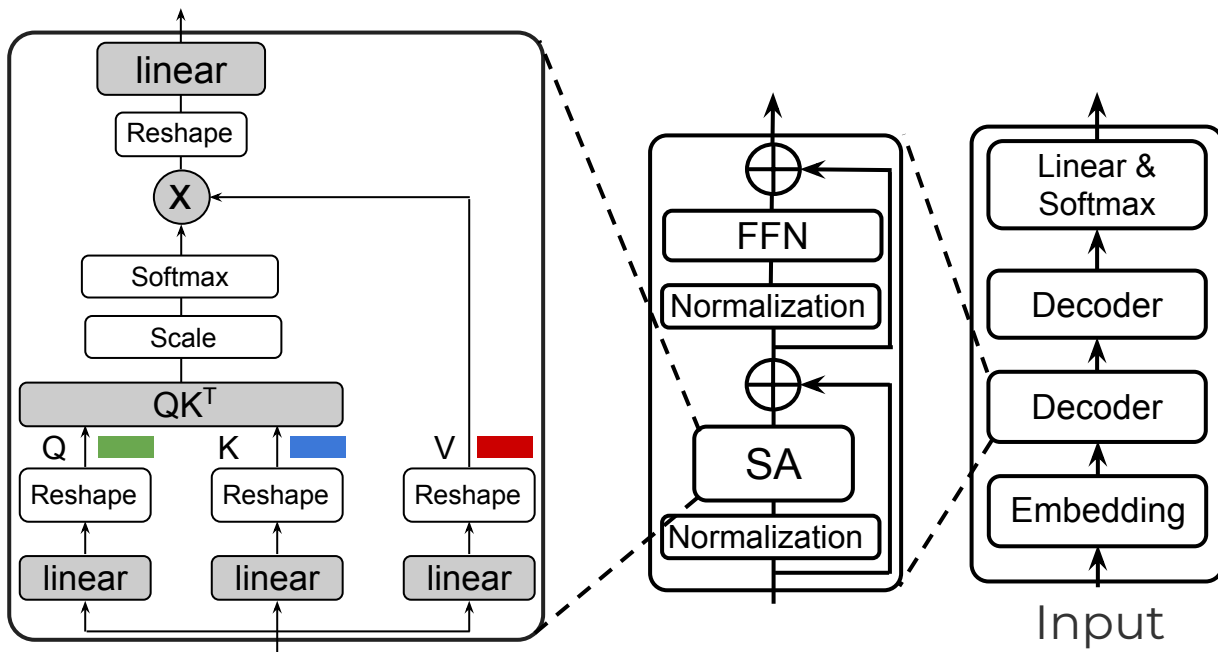
- Large Model Data Distribution Analysis
- Large Model Quantization Strategies
 - Smoothing Techniques
 - Quantization Techniques
- Large Model Pruning Strategies
- Low-rank Decomposition for LLM

Transformers as a Generative AI Tool



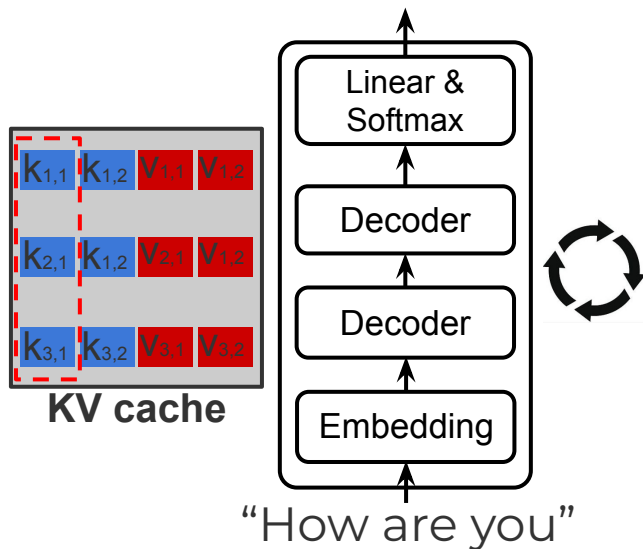
- Each token is generated in an autoregressive manner.

Transformers as a Generative AI Tool



- We need to buffer the v and k for later usage.

LLM: Prefilling

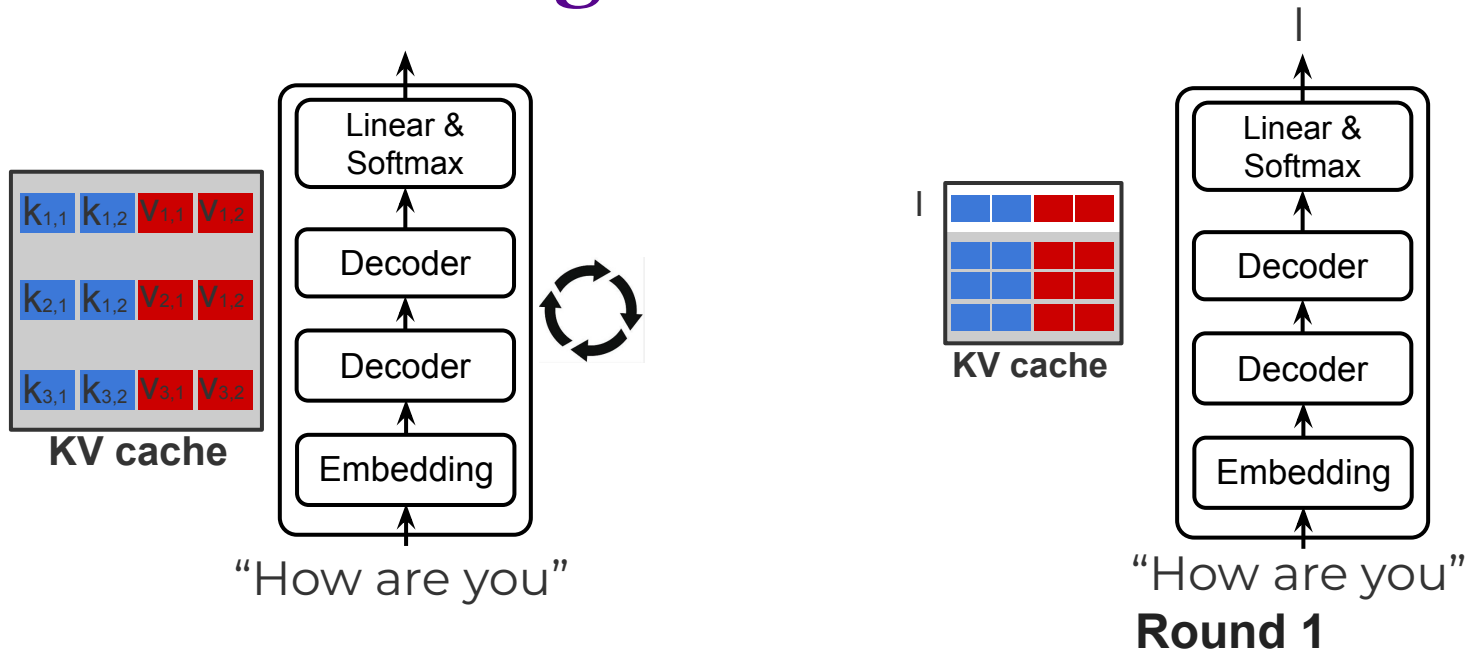


$k_{i,j}$ Key vector for i th token in j th layer
($1 \times E$)

$v_{i,j}$ Value vector for i th token in j th layer
($1 \times E$)

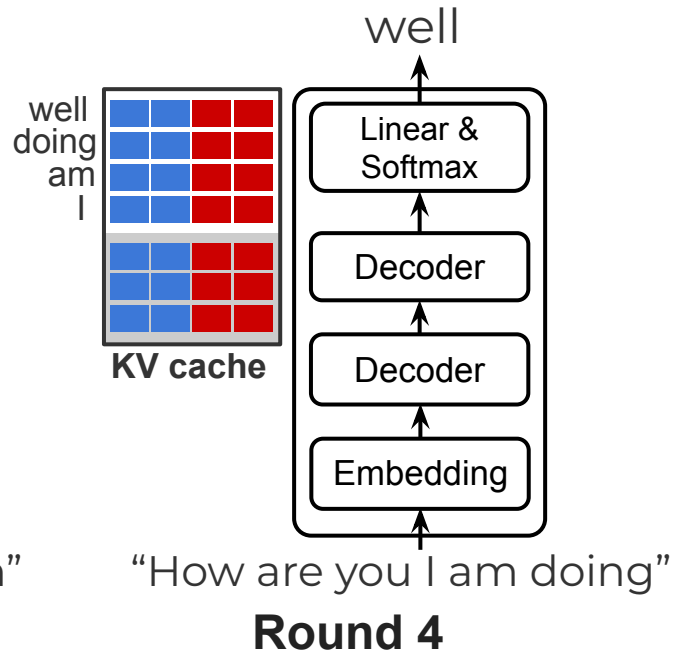
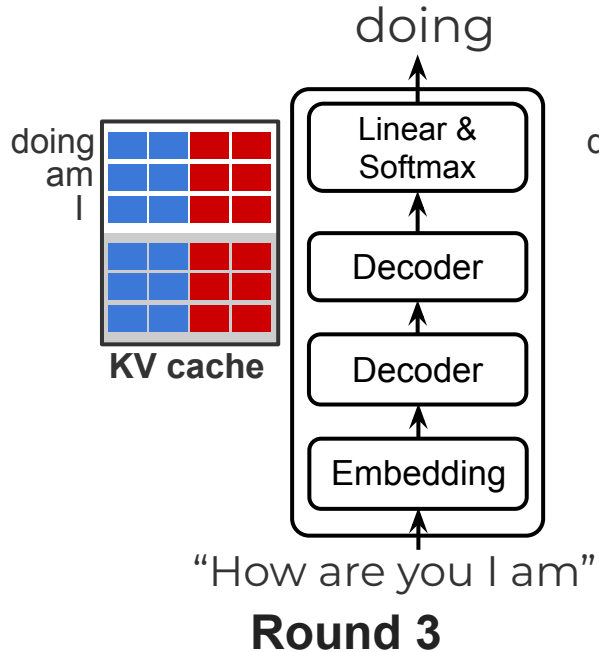
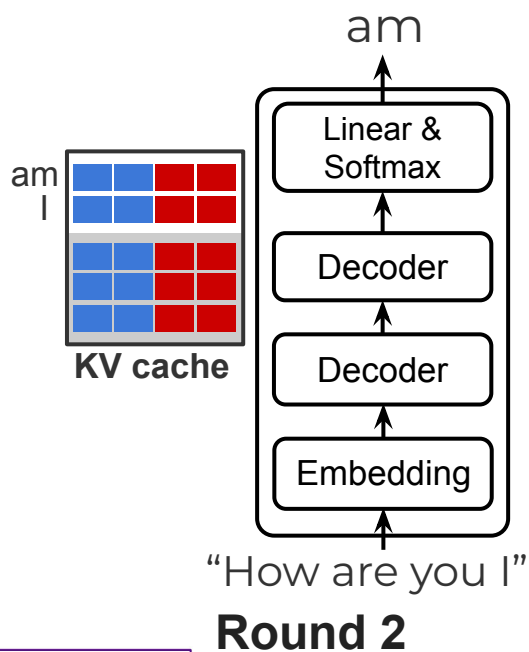
- During the prefilling stage, LLM processes the entire prompt, or context tokens jointly, saving the KV vectors into the memory.

LLM: Decoding

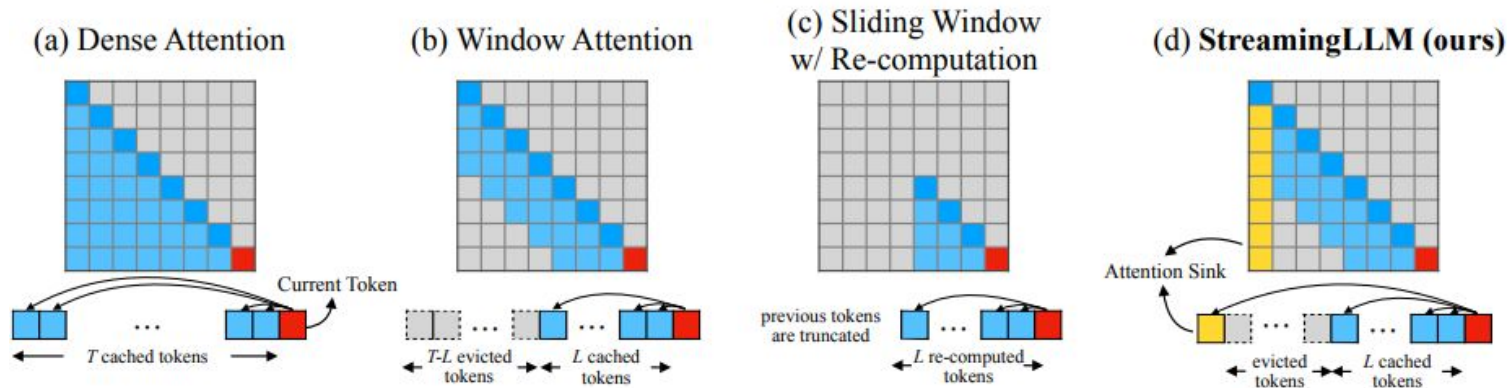


- During the decoding stage, LLM generates the responses in an autoregressive way.

LLM: Decoding

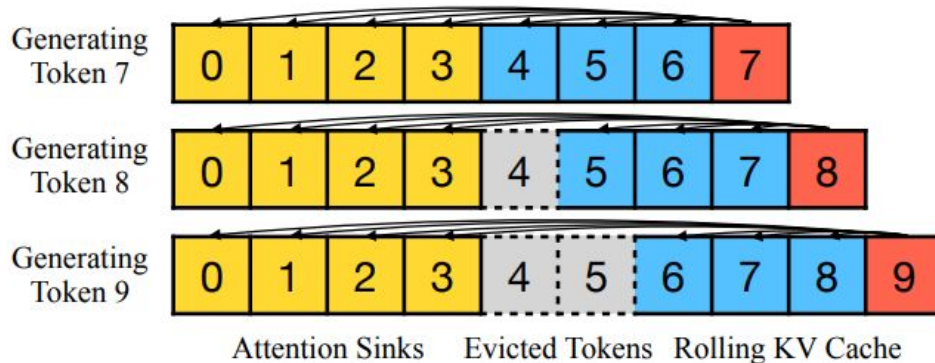


Streaming-LLM



- We observe an interesting phenomenon, namely attention sink, that keeping the KV of initial tokens will largely recover the performance of window attention.
- There are strong attention scores towards initial tokens as a “sink” even if they are not semantically important.

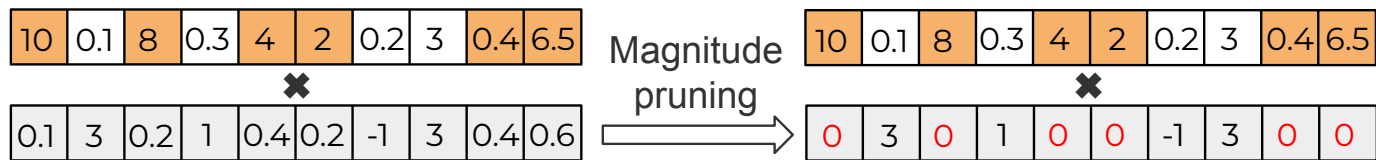
Streaming-LLM



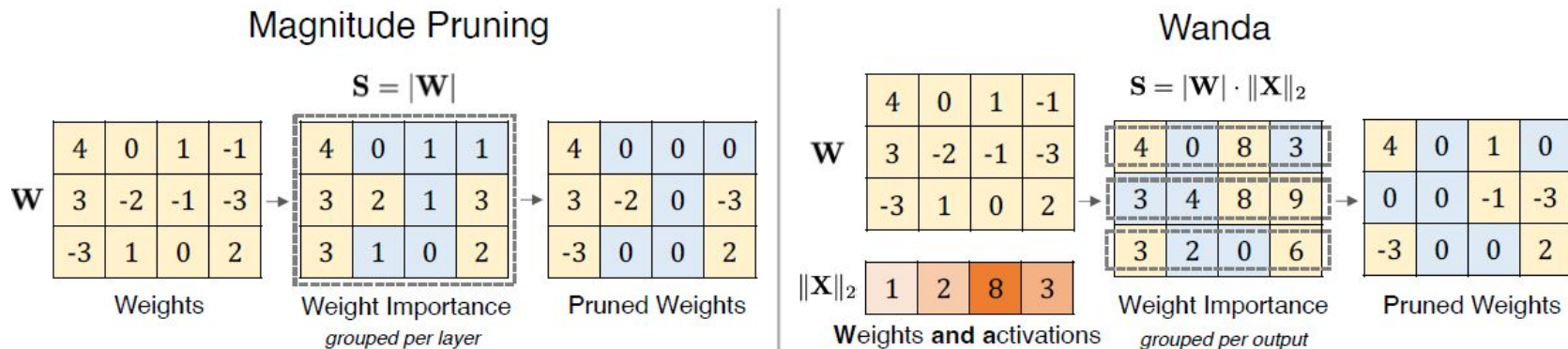
- Assume a fixed cache size, how to accommodate growing KV cache size as the decoding length increases.
- First m sink tokens + most recent K tokens
- Usually m ranges from 1-8

Drawbacks of Magnitude Pruning

- The major drawback of magnitude based pruning is that it does not consider the impact of the input when making the pruning decision.



LLM Pruning: Wanda



- Prune the weights by considering the input statistics.
- For each weight, if the corresponding input's magnitude is large, the output will also be large.
- Need some samples for calibration.

LLM Pruning: Wanda

Method	Weight Update	Sparsity	LLaMA				LLaMA-2		
			7B	13B	30B	65B	7B	13B	70B
Dense	-	0%	5.68	5.09	4.77	3.56	5.12	4.57	3.12
Magnitude	✗	50%	17.29	20.21	7.54	5.90	14.89	6.37	4.98
SparseGPT	✓	50%	7.22	6.21	5.31	4.57	6.51	5.63	3.98
Wanda	✗	50%	7.26	6.15	5.24	4.57	6.42	5.56	3.98
Magnitude	✗	4:8	16.84	13.84	7.62	6.36	16.48	6.76	5.54
SparseGPT	✓	4:8	8.61	7.40	6.17	5.38	8.12	6.60	4.59
Wanda	✗	4:8	8.57	7.40	5.97	5.30	7.97	6.55	4.47
Magnitude	✗	2:4	42.13	18.37	9.10	7.11	54.59	8.33	6.33
SparseGPT	✓	2:4	11.00	9.11	7.16	6.28	10.17	8.32	5.40
Wanda	✗	2:4	11.53	9.58	6.90	6.25	11.02	8.27	5.16

Table 3: WikiText perplexity of pruned LLaMA and LLaMA-2 models. Wanda performs competitively against prior best method SparseGPT, without introducing any weight update.

Topics

- Large Model Data Distribution Analysis
- Large Model Quantization Strategies
 - Smoothing Techniques
 - Quantization Techniques
- Large Model Pruning Strategies
- Low-rank Decomposition for LLM

Low Rank Optimization for DNN Efficiency

- Weight tensors can be decomposed into:

$$\begin{array}{c} \begin{array}{ccccc} & n & & r & \\ & \boxed{W} & = & \begin{array}{c} m \\ \boxed{} \end{array} & \times & \begin{array}{c} r \\ \boxed{} \end{array} & \times & \begin{array}{c} r \\ \boxed{} \end{array} & \times & \begin{array}{c} n \\ \boxed{} \end{array} \\ m & & & & & & & & & \end{array} \\ \\ \begin{array}{ccccc} & & & r & \\ & & = & \begin{array}{c} m \\ \boxed{W_1} \end{array} & \times & \begin{array}{c} n \\ \boxed{W_2} \end{array} \\ & & & & & & & & & \end{array} \end{array} \quad r \leq \min(m, n)$$

Singular Value Decomposition

$$\begin{matrix} n \\ m \end{matrix} \boxed{W} = \begin{matrix} r \\ m \end{matrix} \boxed{U} \times \begin{matrix} r \\ r \end{matrix} \boxed{R} \times \begin{matrix} n \\ r \end{matrix} \boxed{V^T} = \begin{matrix} r \\ m \end{matrix} \boxed{W_1} \times \begin{matrix} n \\ r \end{matrix} \boxed{W_2}$$

- W: Input matrix
 - $m \times n$ matrix
- V ($n \times r$ matrix) contains the orthonormal eigenvectors of $W^T W$
- U ($m \times r$ matrix) contains the orthonormal eigenvectors of $W W^T$
- R: Singular value matrix
 - $r \times r$ diagonal matrix, r is the rank of W

Before: mn

After: $mr + rn = r(m+n) = \min(m,n)(m+n)$
assume W is full rank

- Without rank truncation, the number of parameters increases.

Singular Value Decomposition

The diagram illustrates the Singular Value Decomposition (SVD) of a matrix W . On the left, a matrix X of size $s \times m$ is multiplied by a matrix W of size $m \times n$. This is equated to the product of three matrices: X (size $s \times m$), W_1 (size $m \times r$), and W_2 (size $r \times n$). The matrix X is represented by a gray box, W by a white box, W_1 by a red box, and W_2 by a blue box. Dimensions are labeled above and to the left of each matrix.

Computational cost = smn

Computational cost = $smr + snr$
 $= s(m+n)r$
 $= s(m+n)\min(m,n)$

- Without rank truncation, the computational cost will increase.

QSVD (NeurIPS'25 Spotlight)

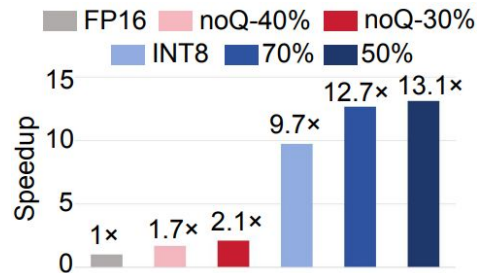
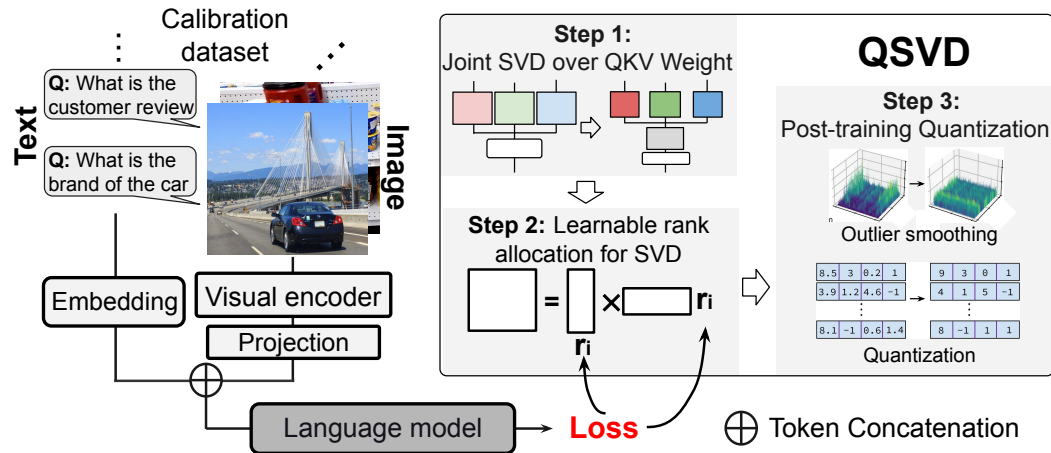
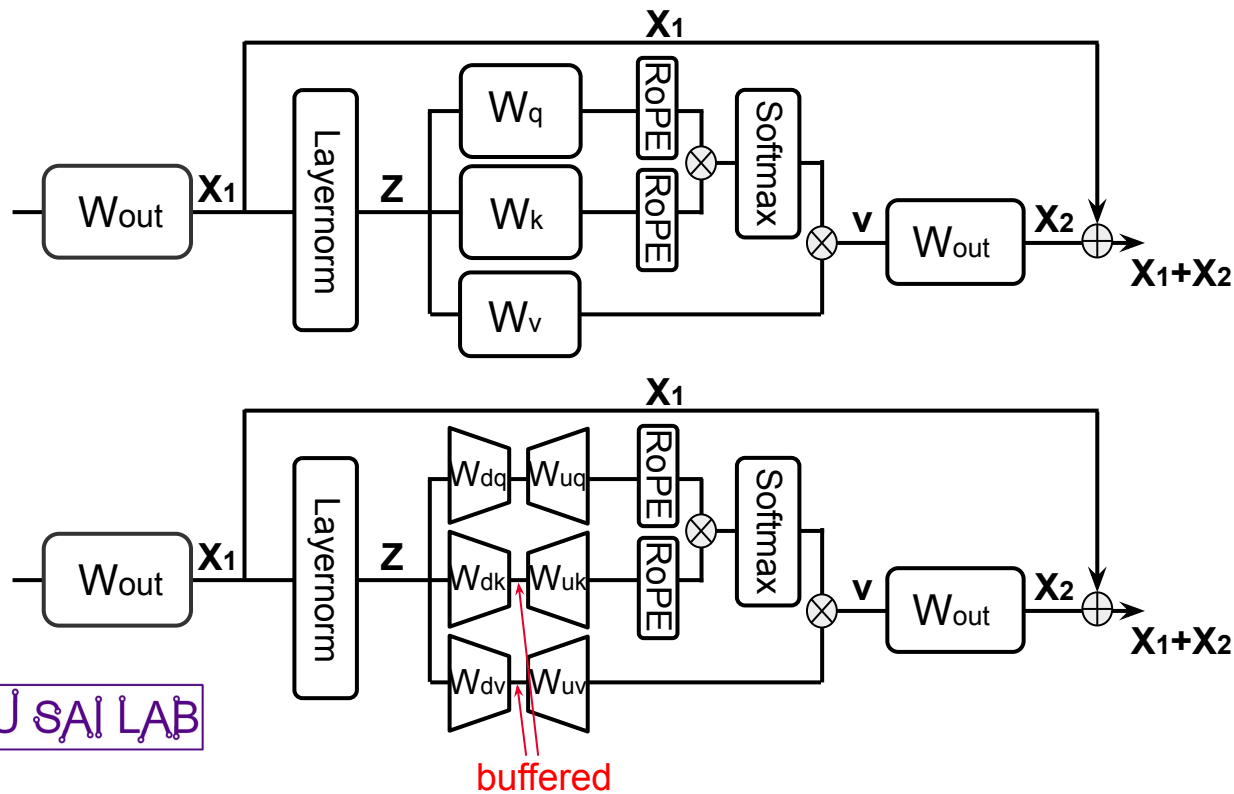


Figure 4: Normalized speedup of QSVD-noQ and QSVD W8A8 on low-end GPU.

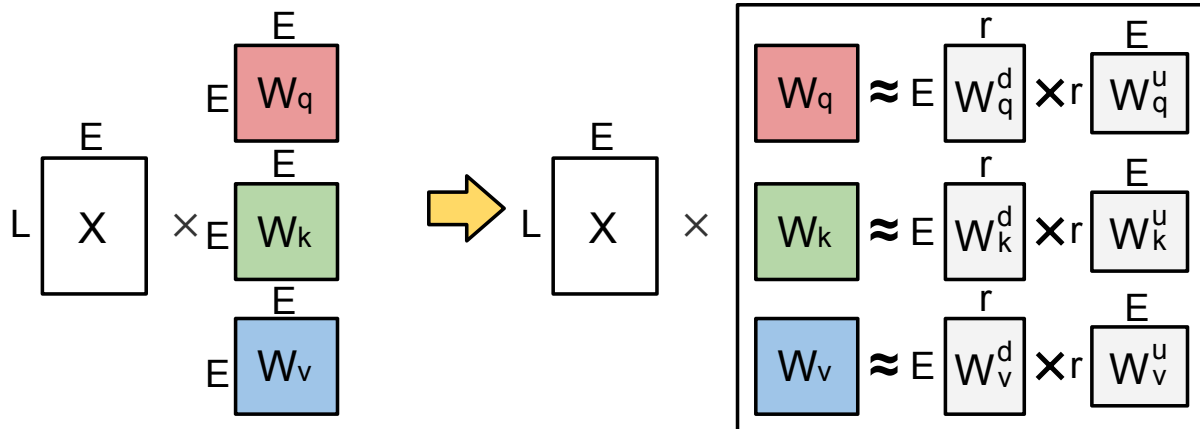
- We propose a method that applies SVD to reduce computation and KV-cache traffic, coupled with custom-designed kernels to achieve substantial latency reduction.

Applying SVD within the Self-Attention Block



SVD decomposition can potentially save the MAC operations, memory storage and KV cache size.

Applying SVD within the Self-Attention Block



Total MACs: $3LE^2$

Total weight parameters: $3E^2$

Total cache size: $2LE$

Total MACs: $6LrE$

Total weight parameters: $6rE$

Total cache size: $2rL$

- If $r < E$, SVD can reduce KV cache size.
- If $r < 0.5E$, SVD can result in computation and weight storage reduction.

ASVD

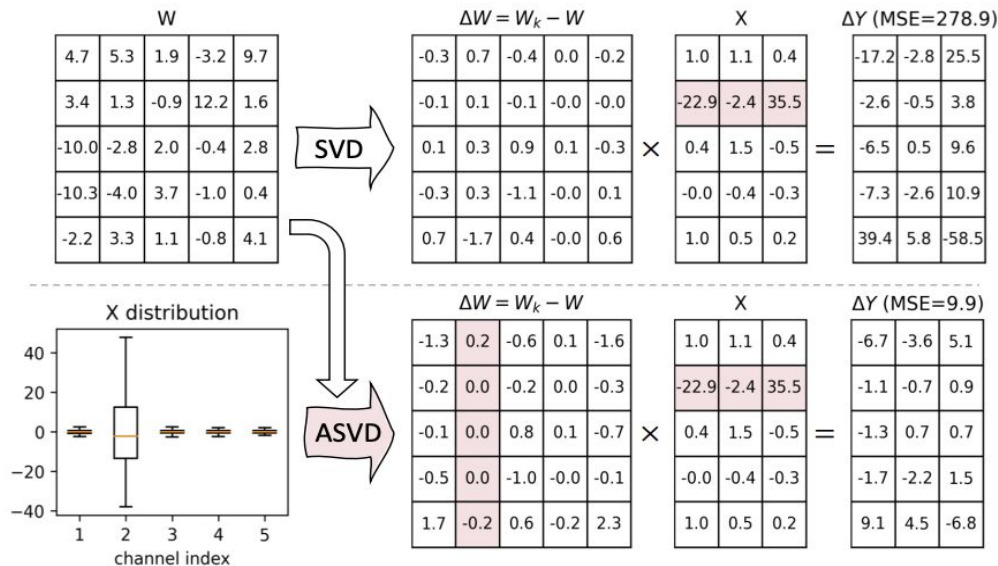
- ASVD changes the objective considering impact of activation x

$$L = ||WX - SVD(W)X||$$

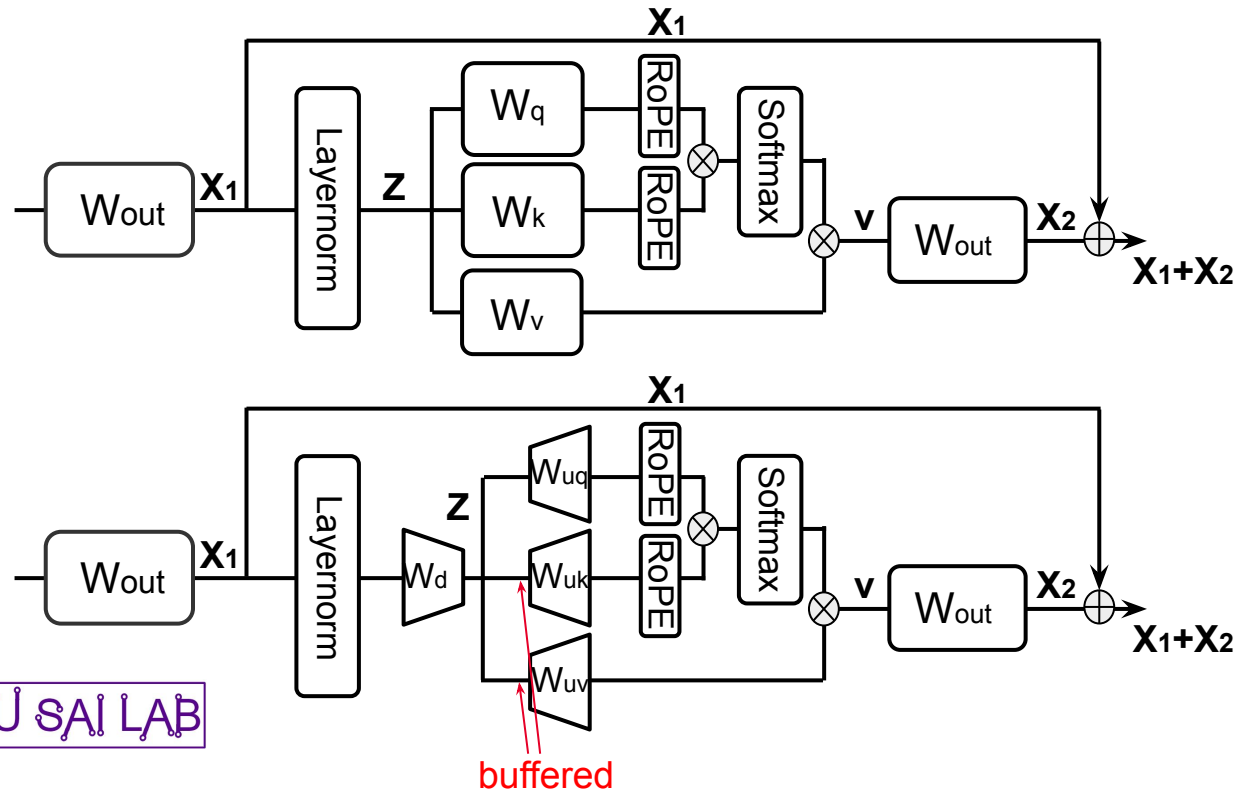
$$L = ||WX - SVD(WS)S^{-1}X||$$

S is a diagonal matrix

- To mitigate impact of channel wise outlier within X , we scale the input activation to mitigate the outliers.

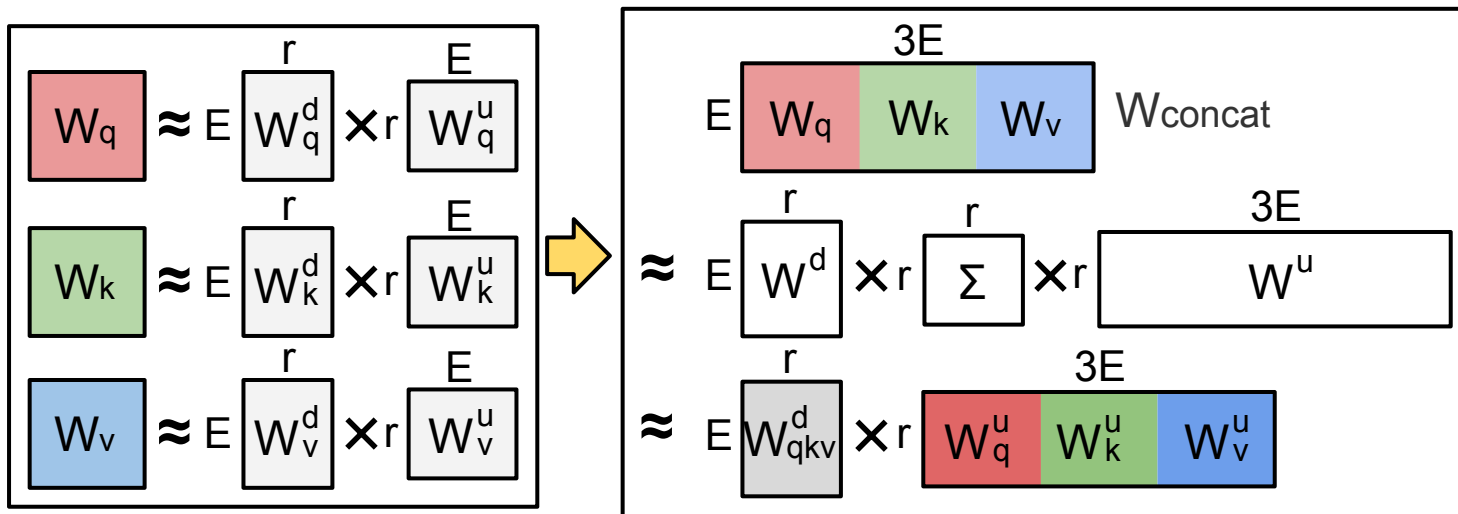


Applying SVD within the Self-Attention Block



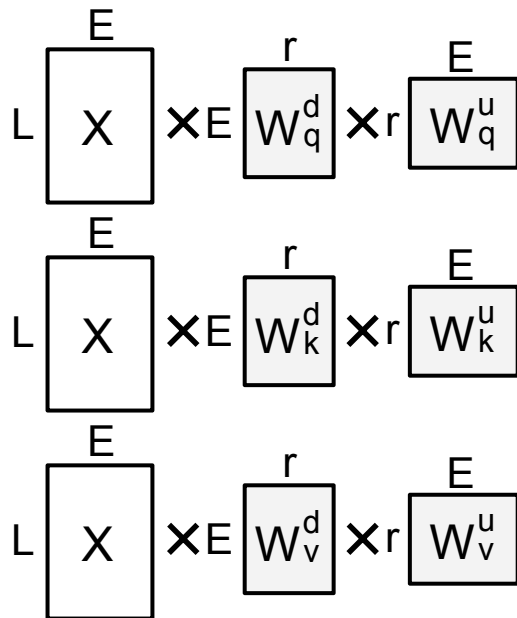
SVD can be further applied to the joint QKV matrix to reduce the KV-cache size.

Applying SVD within the Self-Attention Block



- Concat and SVD

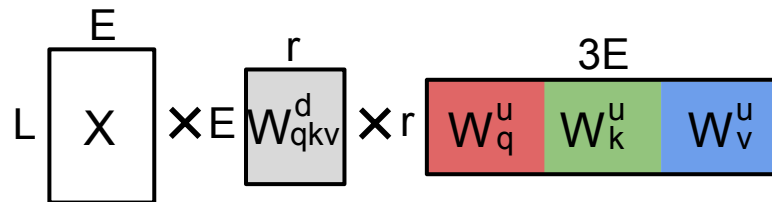
Applying SVD within the Self-Attention Block



Total MACs: $6LrE$

Total weight parameters: $6rE$

Total cache size: $2rL$



Total MACs: $4LrE$

Total weight parameters: $4rE$

Total cache size: rL

Rank Allocation

- How to assign the rank r to each layer?

$$\frac{L(\sigma + \Delta\sigma) - L(\sigma)}{\Delta\sigma} \approx \frac{dL}{d\sigma} \Rightarrow \mathbb{E}_D(|\frac{dL}{d\sigma}|)$$

- The importance of each layer can be expressed as:

$$\sum_i \mathbb{E}_D(|\frac{dL}{d\sigma_i}|)$$

- Given a total rank budget R , we can allocate the rank for each layer in proportion to the importance score.

QSVD Performance

	Method	ScienceQA-IMG $\uparrow$							
		Acc.	Hw cost	Acc.	Hw cost	Acc.	Hw cost	Acc.	Hw cost
SmolVLM 2B	ASVD	53.84%	$R_1 : 100\%$	7.88%	$R_1 : 90.0\%$	0.69%	$R_1 : 80.0\%$	0.10%	$R_1 : 70.0\%$
	SVDLLM	65.89%	$R_2 : 50.0\%$	34.61%	$R_2 : 42.5\%$	9.07%	$R_2 : 35.0\%$	3.02%	$R_2 : 27.5\%$
	QSVD-noQ	83.78%	$R_1 : 100\%$ $R_2 : 37.5\%$	81.70%	$R_1 : 90.0\%$ $R_2 : 33.75\%$	79.57%	$R_1 : 80.0\%$ $R_2 : 30.0\%$	77.64%	$R_1 : 70.0\%$ $R_2 : 26.25\%$
	FP16	Accuracy: 84.53%							
LLaVA-Next 7B	ASVD	50.72%	$R_1 : 63.3\%$	47.15%	$R_1 : 60.0\%$	40.26%	$R_1 : 56.7\%$	25.73%	$R_1 : 53.3\%$
	SVDLLM	65.94%	$R_2 : 22.5\%$	66.14%	$R_2 : 20.0\%$	64.90%	$R_2 : 17.5\%$	62.87%	$R_2 : 15.0\%$
	QSVD-noQ	69.91%	$R_1 : 60.0\%$ $R_2 : 22.5\%$	68.22%	$R_1 : 53.3\%$ $R_2 : 20.0\%$	67.03%	$R_1 : 46.7\%$ $R_2 : 17.5\%$	65.15%	$R_1 : 40.0\%$ $R_2 : 15.0\%$
	FP16	Accuracy: 69.51%							
LLaVA-v1.5 13B	ASVD	64.70%	$R_1 : 63.3\%$	56.92%	$R_1 : 60.0\%$	46.50%	$R_1 : 56.7\%$	42.79%	$R_1 : 53.3\%$
	SVDLLM	71.44%	$R_2 : 22.5\%$	71.44%	$R_2 : 20.0\%$	71.29%	$R_2 : 17.5\%$	70.50%	$R_2 : 15.0\%$
	QSVD-noQ	71.79%	$R_1 : 60.0\%$ $R_2 : 22.5\%$	71.74%	$R_1 : 53.3\%$ $R_2 : 20.0\%$	71.74%	$R_1 : 46.7\%$ $R_2 : 17.5\%$	70.80%	$R_1 : 40.0\%$ $R_2 : 15.0\%$
	FP16	Accuracy: 71.78%							

- QSVD achieves a comparable and even an better performance than the original model.

Problem of SVD

- The problem we have to solve can be formulated as this:

$$\underset{W'}{\operatorname{argmin}} ||W - W'||^2$$

$$W' = DU$$

$$D \in \mathcal{R}^{E \times r}$$

$$U \in \mathcal{R}^{r \times E}$$

- However, we know that each element of W has different impact on the final accuracy.

WSVD (ICLR'26)

- The problem we have to solve can be formulated as this:

$$\underset{W'}{\operatorname{argmin}} \sum_{ij} \|a_{ij}W_{ij} - a_{ij}W'_{ij}\|$$

$$W' = DU$$

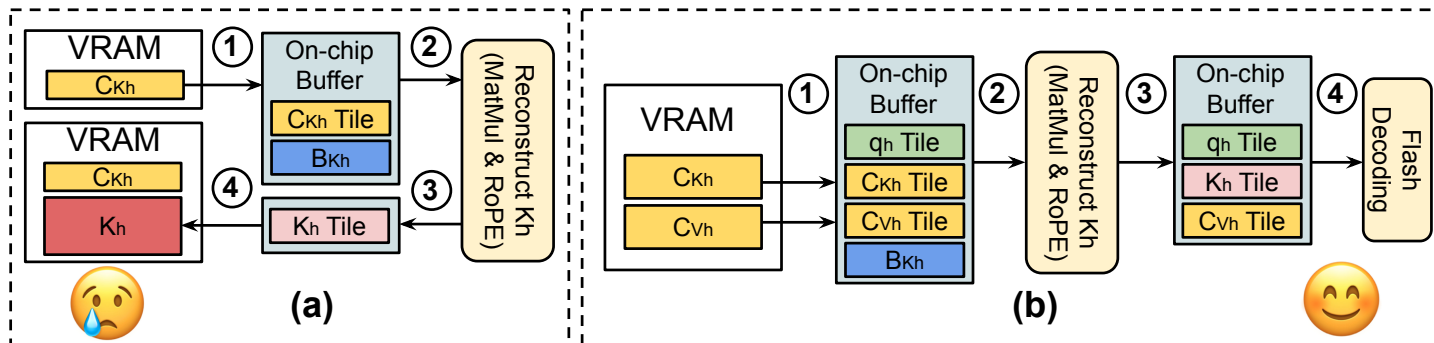
$$D \in \mathcal{R}^{E \times r}$$

$$U \in \mathcal{R}^{r \times E}$$

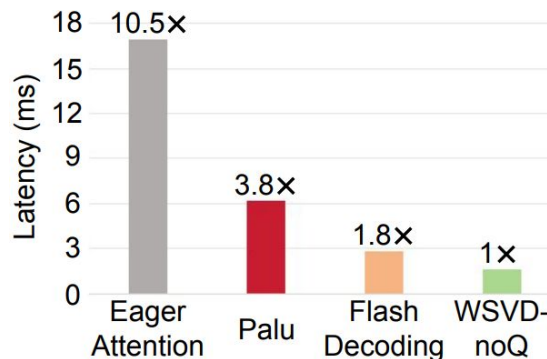
- This problem has no closed-form solution, therefore we can use the deep-learning method to solve this, train D and U such that the objective function is minimized.
- For importance score, we can use the following formula to evaluate.

$$L(w) - L(w') \approx \frac{dL}{dw}(w - w')$$

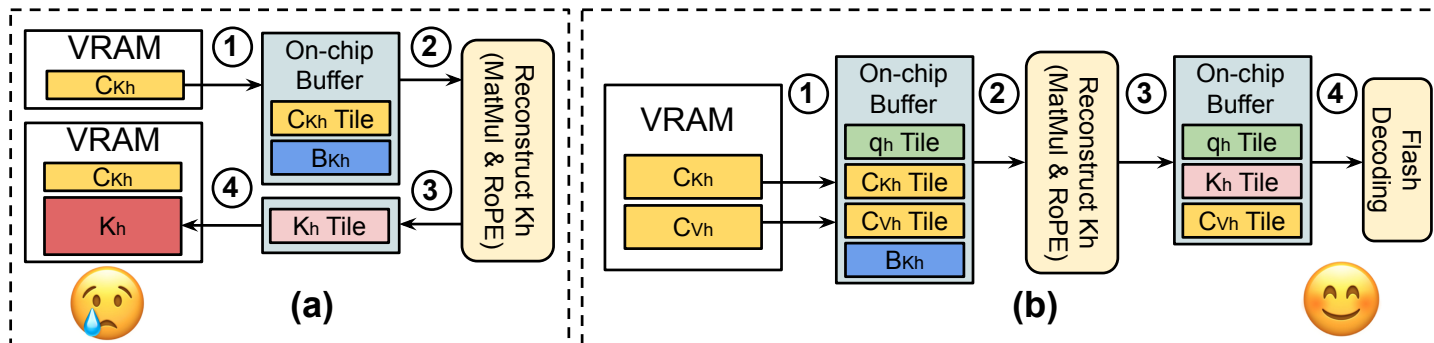
WSVD (ICLR'26)



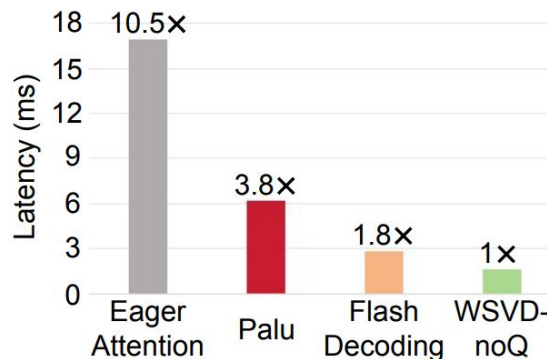
Acc.	Method	ScienceQA-IMG $\uparrow$				
		$\rho_1 : 90\%$	$\rho_1 : 80\%$	$\rho_1 : 70\%$	$\rho_1 : 60\%$	$\rho_1 : 50\%$
LLaVA-v1.5 7B	ASVD	49.93%	50.12%	47.10%	36.69%	19.19%
	SVD-LLM	65.44%	63.71%	61.92%	57.41%	55.53%
	QSVD-noQ	67.72%	68.22%	67.08%	65.05%	62.37%
	WSVD-noQ	68.17%	67.72%	67.28%	65.89%	65.49%
	FP16	Accuracy: 68.01%				
LLaVA-Next 13B	ASVD	71.24%	70.60%	71.44%	71.38%	69.81%
	SVD-LLM	72.53%	72.24%	71.74%	71.15%	70.55%
	QSVD-noQ	71.94%	72.14%	71.74%	72.14%	71.79%
	WSVD-noQ	72.88%	72.98%	73.57%	73.48%	73.28%
	FP16	Accuracy: 73.23%				



WSVD (ICLR'26)



Acc.	Method	ScienceQA-IMG $\uparrow$				
		$\rho_1 : 90\%$	$\rho_1 : 80\%$	$\rho_1 : 70\%$	$\rho_1 : 60\%$	$\rho_1 : 50\%$
LLaVA-v1.5 7B	ASVD	49.93%	50.12%	47.10%	36.69%	19.19%
	SVD-LLM	65.44%	63.71%	61.92%	57.41%	55.53%
	QSVD-noQ	67.72%	68.22%	67.08%	65.05%	62.37%
	WSVD-noQ	68.17%	67.72%	67.28%	65.89%	65.49%
	FP16	Accuracy: 68.01%				
LLaVA-Next 13B	ASVD	71.24%	70.60%	71.44%	71.38%	69.81%
	SVD-LLM	72.53%	72.24%	71.74%	71.15%	70.55%
	QSVD-noQ	71.94%	72.14%	71.74%	72.14%	71.79%
	WSVD-noQ	72.88%	72.98%	73.57%	73.48%	73.28%
	FP16	Accuracy: 73.23%				



LSVD (NeurIPS'26)

