NYU

Lecture 07: Efficiency Strategies for Large Language Models

Notes

- Lab 2 is due next Sunday.
- Think about the project, discuss with me during office hours or after class.
- In-course quiz today.
- Midterm
  - March 27, in class.
  - Will cover materials up to lecture 8 (Mar 13)
  - Will send out a coverage by this weekend
  - Will send out some sample questions next week

Recap

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

NYU SAI LAB

Topics

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

NYU SAI LAB

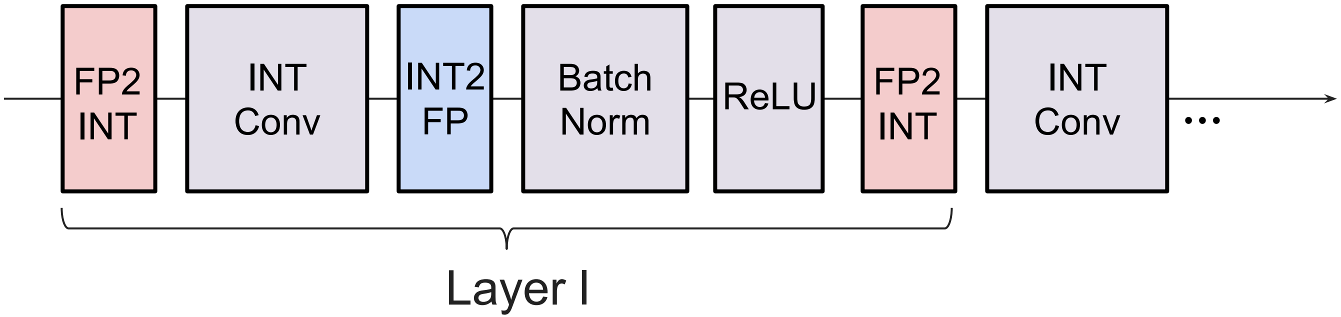
Topics

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

NYU SAI LAB

Quantization

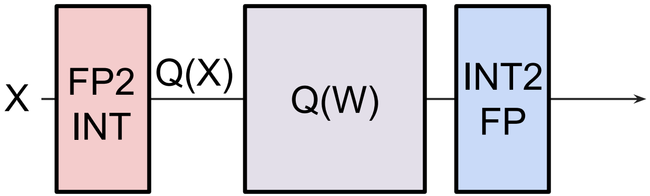
- Quantization on activation needs to be performed dynamically. This will introduce additional compute overhead.
- Also the activation will pass the nonlinear functions, which are usually very sensitive to quantization error, so dequantization is required to convert back to FP 16/32.
- For large models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



NYU SAI LAB

Quantization

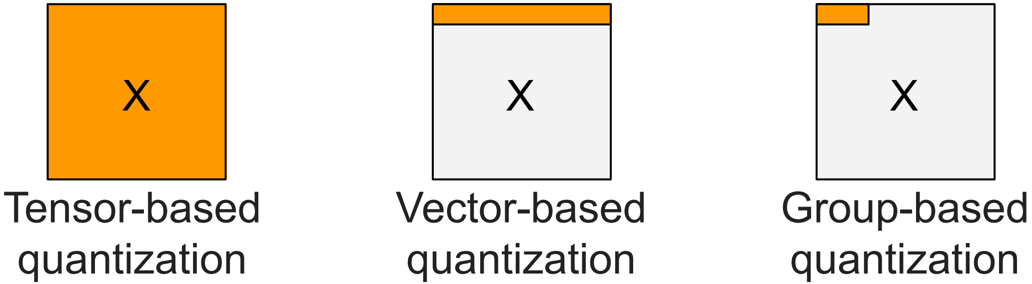
- Quantization on activation needs to be performed dynamically. This will introduce additional compute overhead.
- Also the activation will pass the nonlinear functions, which are usually very sensitive to quantization error, so dequantization is required to convert back to FP 16/32.
- For large models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



NYU SAI LAB

Granularity of Quantization

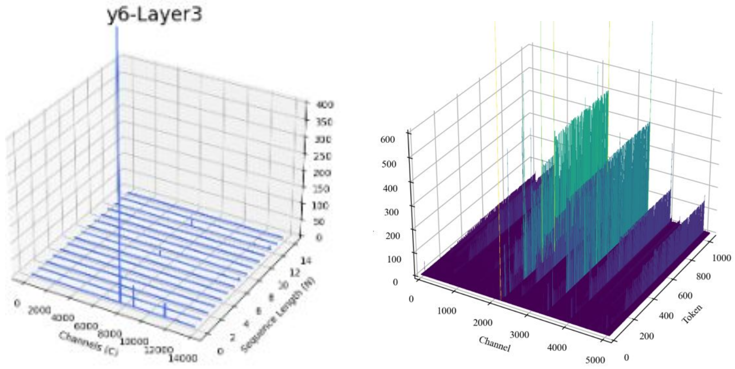
- The activation and weight can be quantized with different granularity:
  - Tensor-based quantization
  - Vector-based quantization
  - Group-based quantization
- A higher quantization granularity will lead to a lower quantization error and a higher hardware implementation cost.



NYU SAI LAB

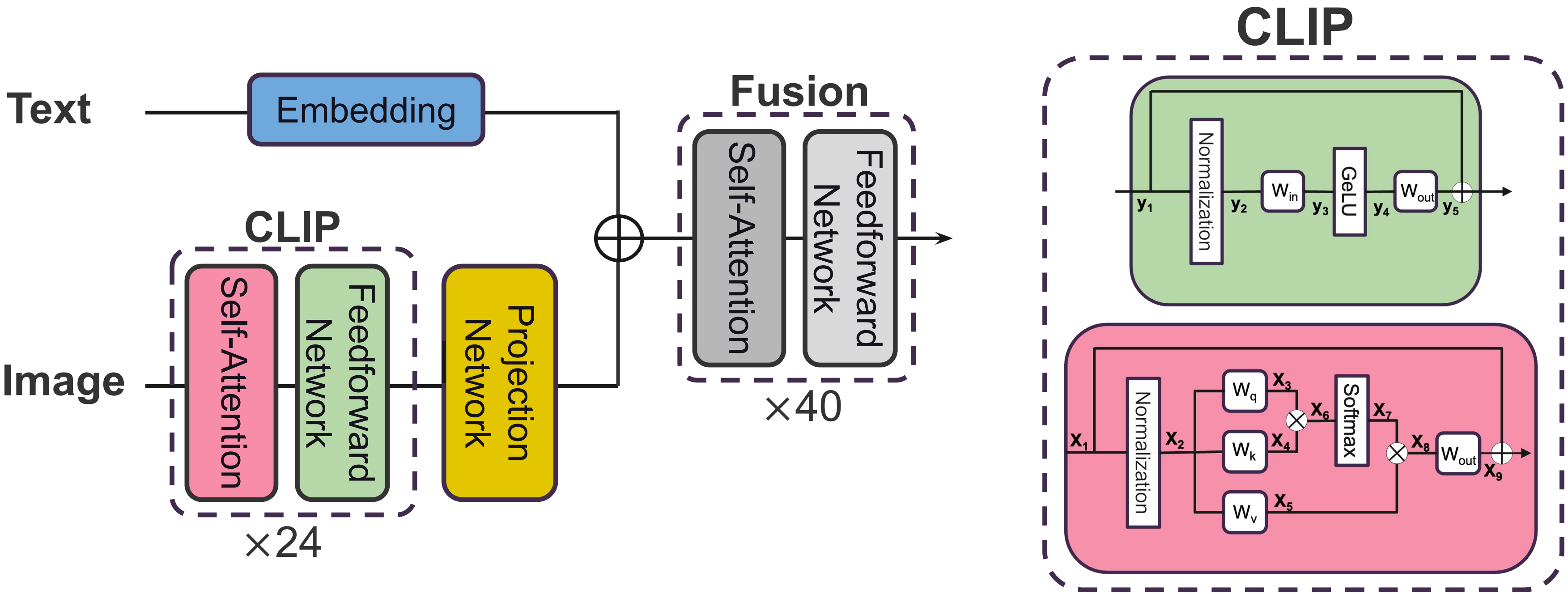
Types of Outlier

- Massive Activation:
  - For an activation matrix A, an massive activation is an element Aij 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, $\eta=3$, $\beta=0.6$



NYU SAI LAB

CLIP Model

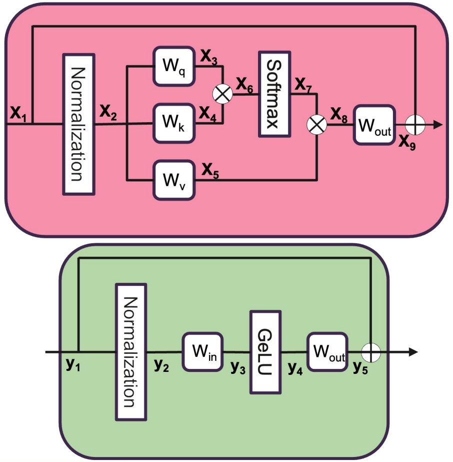


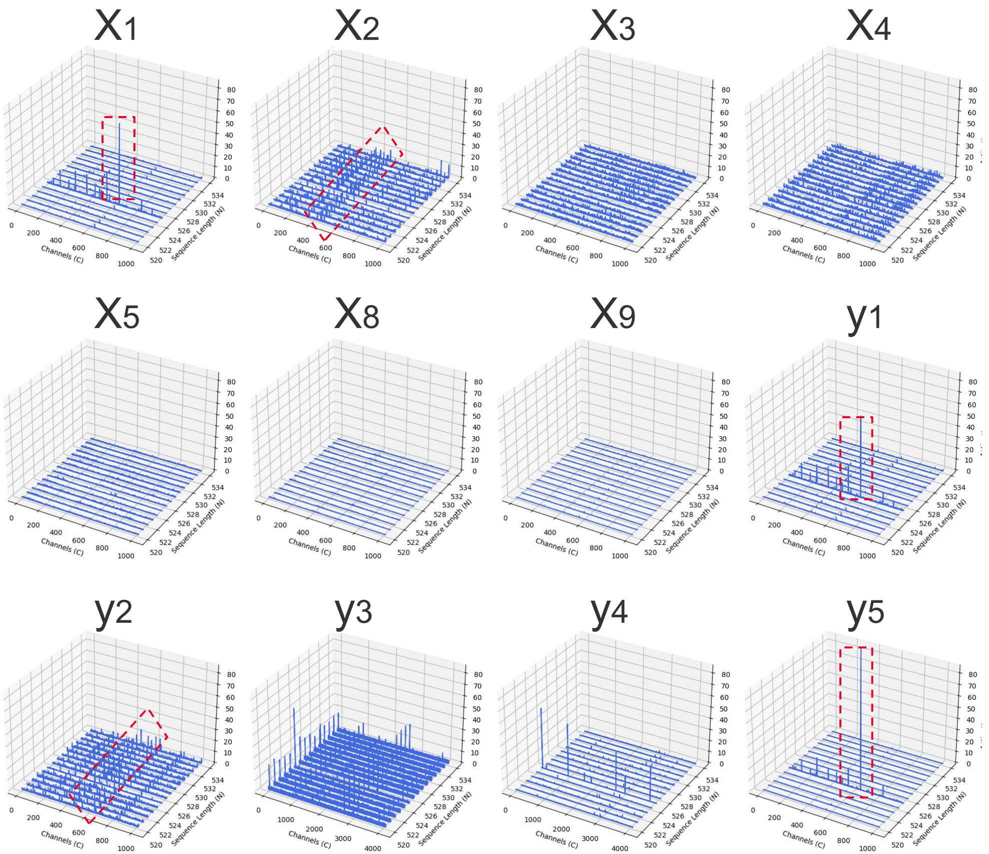
NYU SAI LAB

Radford, Alec, et al. "Learning transferable visual models from natural language supervision." *International conference on machine learning*. PmLR, 2021.

Outlier Study: CLIP Activations

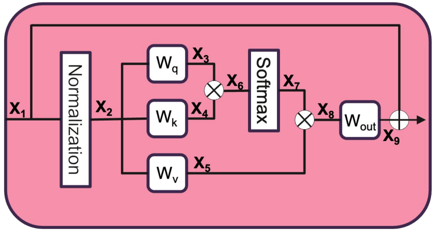
- Outliers with large magnitudes appear at positions x1, y1, and y5, referred to as **massive activations**.



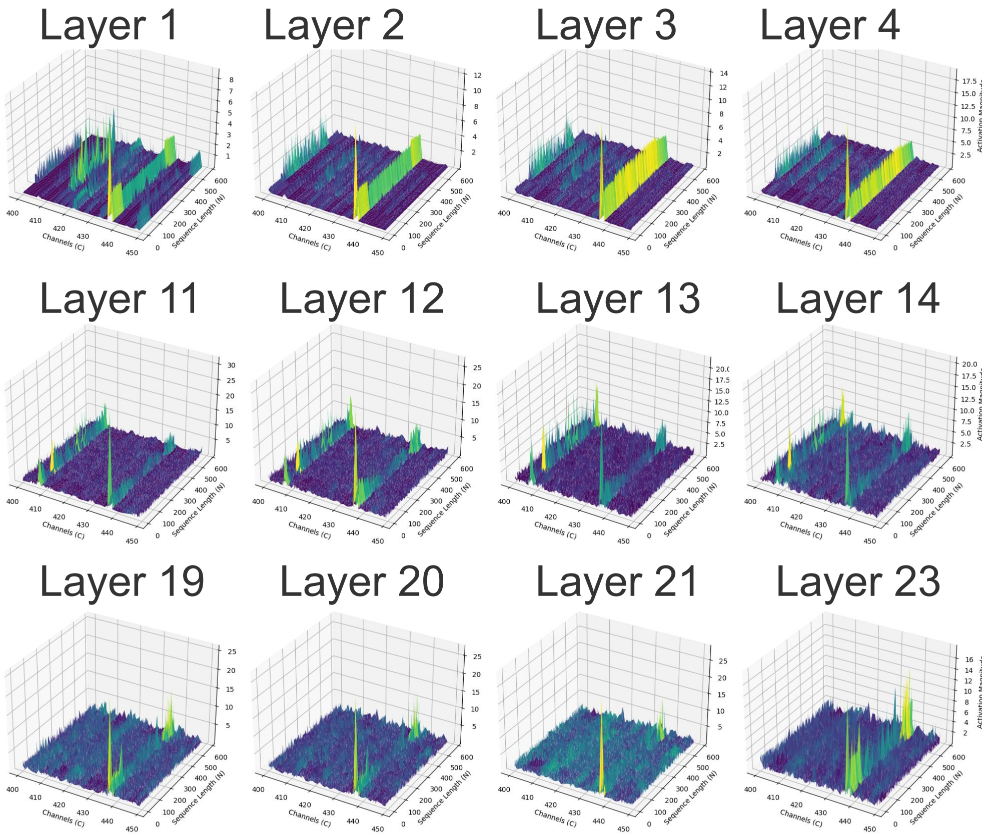


Outlier Study: CLIP Activations

- 3D plots of x2 across layers.



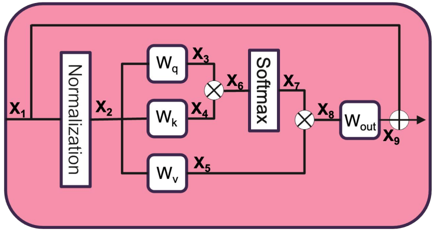
- x2 exhibits channel wise outlier

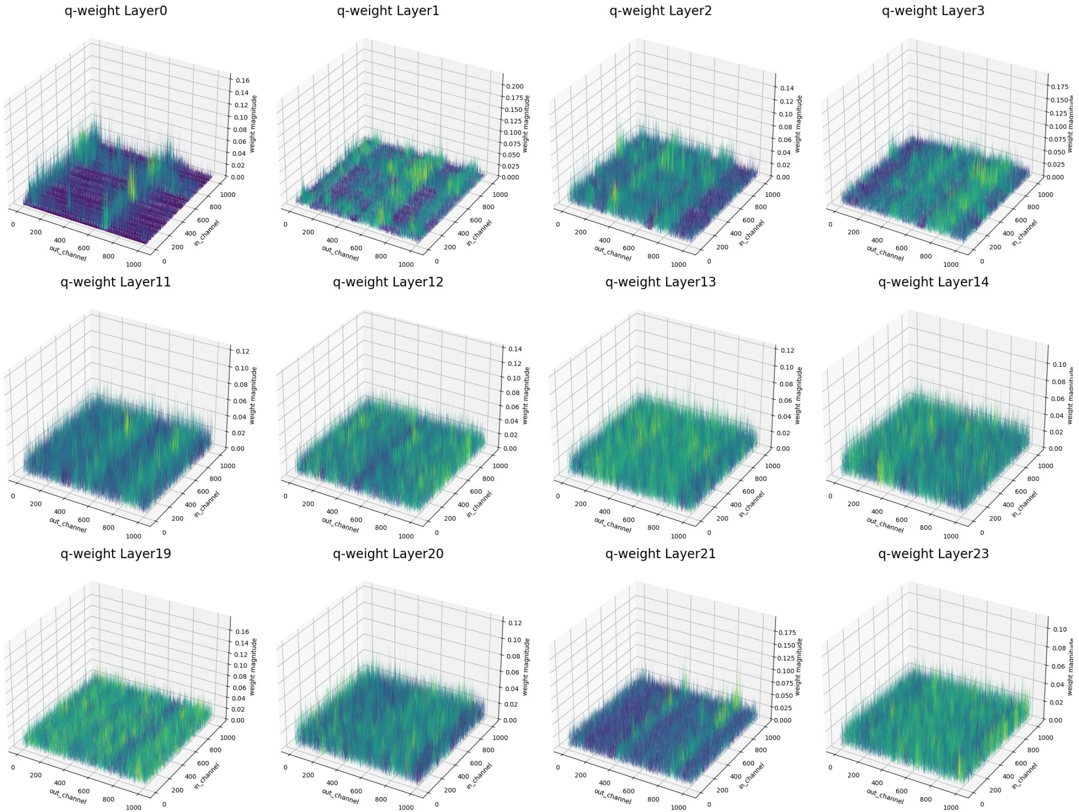


NYU SAI LAB

Outlier Study: CLIP Weights

- W_q across CLIP layers.

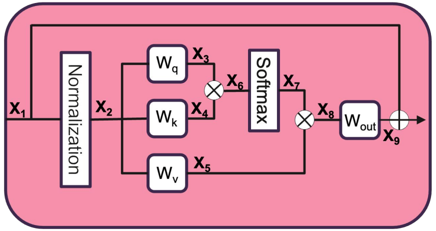


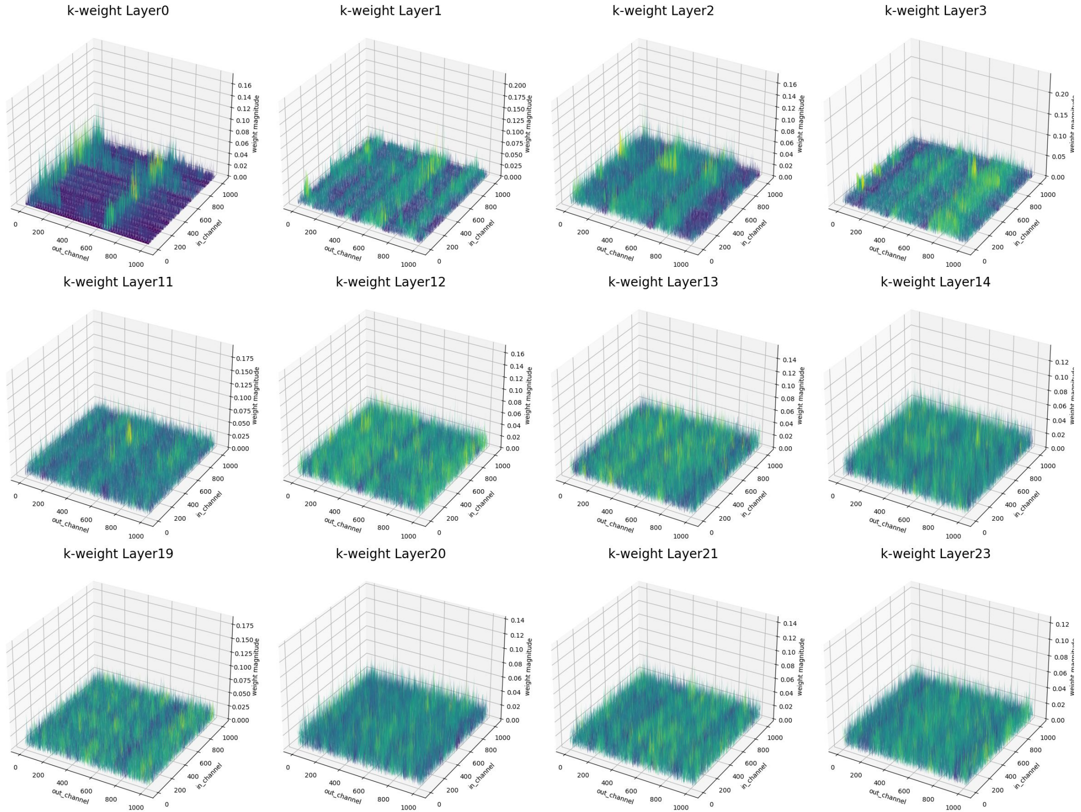


NYU SAI LAB

Outlier Study: CLIP Weights

- W_k across CLIP layers.

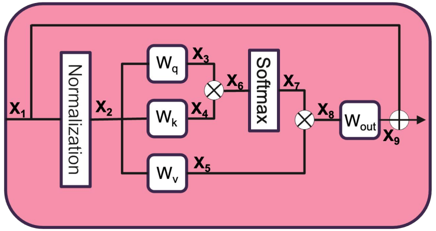


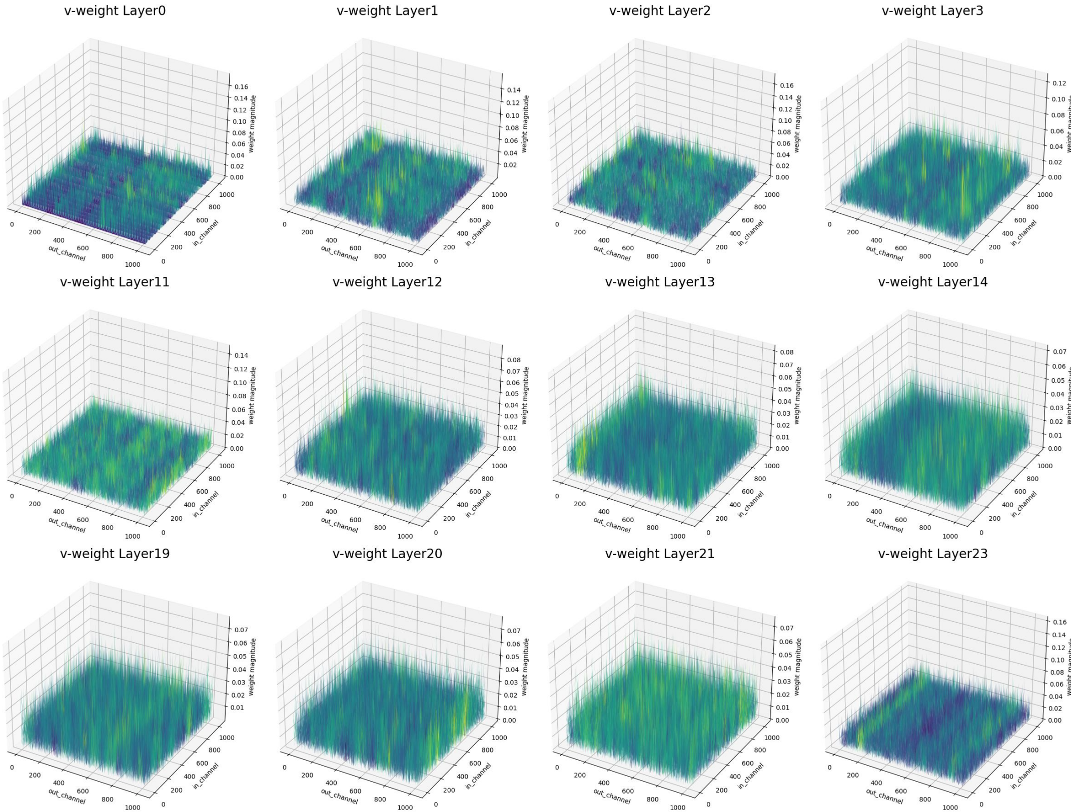


NYU SAI LAB

Outlier Study: CLIP Weights

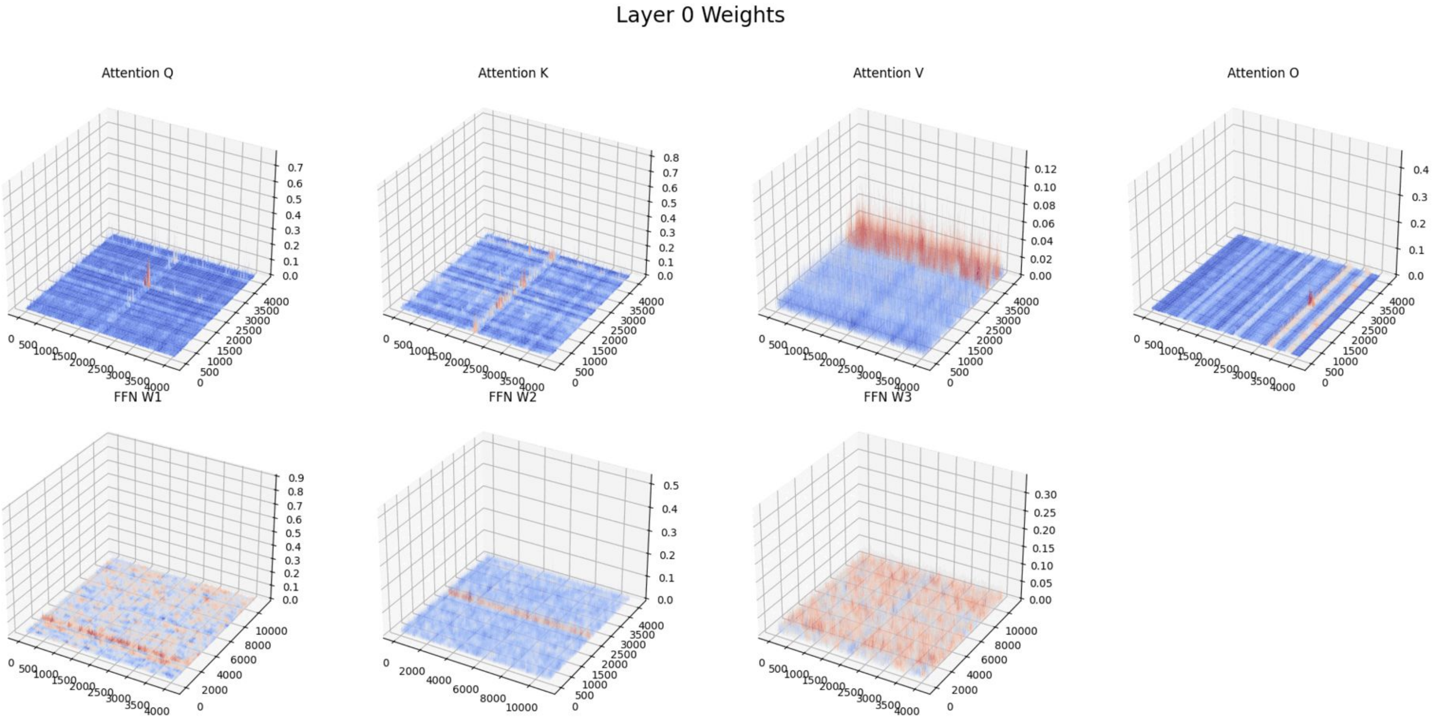
- W_v across CLIP layers.



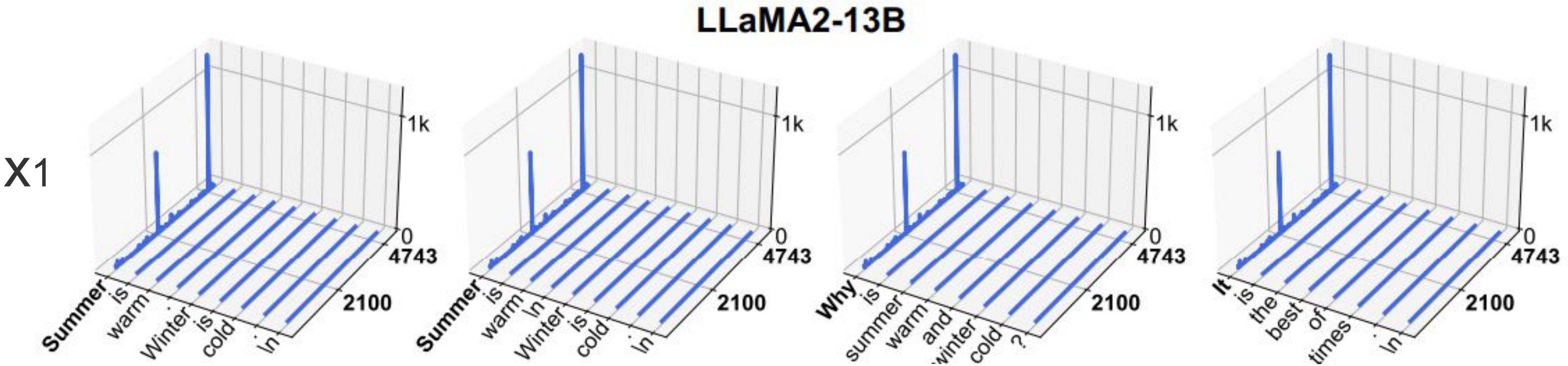


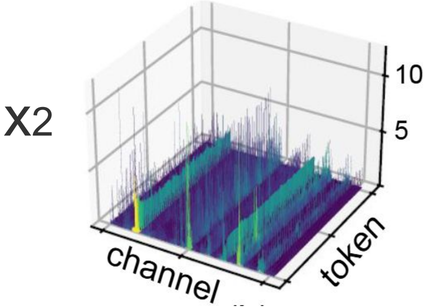
NYU SAI LAB

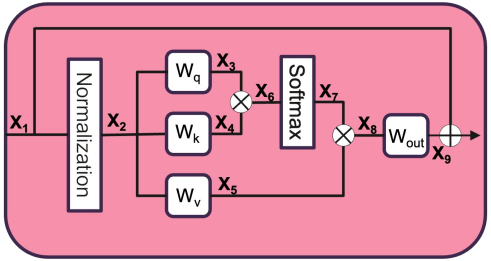
Outlier Study: CLIP Weights



Outlier Study: LLaMA Activations



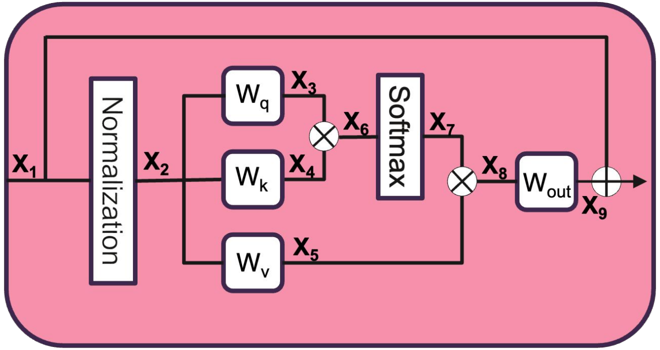




Sun, Mingjie, et al. "Massive activations in large language models." *arXiv preprint arXiv:2402.17762* (2024).

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.
- No clear conclusion has been reached.



NYU SAI LAB

Sun, Mingjie, et al. "Massive activations in large language models." *arXiv preprint arXiv:2402.17762* (2024).

Impact of Massive Activation

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

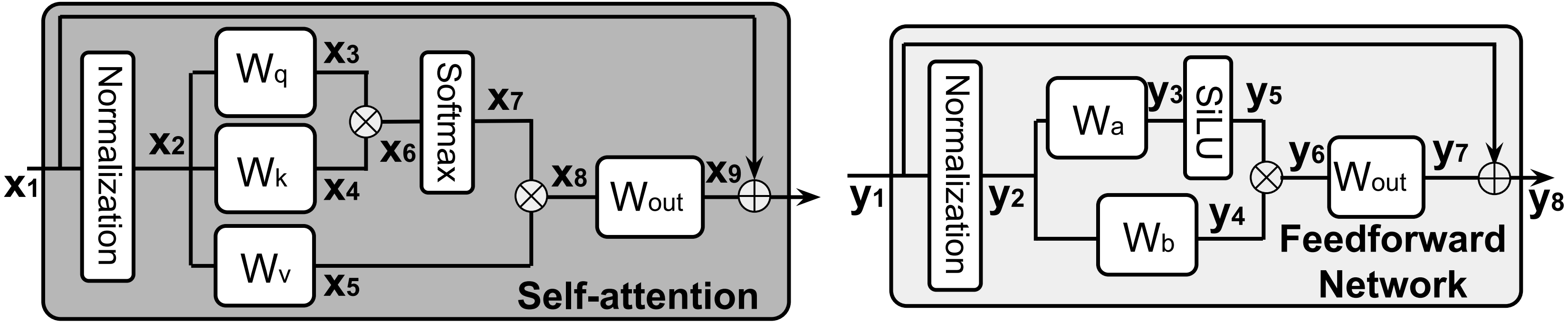
- The truncation of massive activation will cause the significant accuracy degradation of the LLM.
- Massive activations also occur in other types of foundation models that utilize attention-based architectures.

NYU SAI LAB

Raman, Rahul, Khushi Sharma, and Sai Qian Zhang. "Rethinking the Outlier Distribution in Large Language Models: An In-depth Study." *arXiv preprint arXiv:2505.21670* (2025).

Detailed Study of Massive Activation

- y7 of layer 3 first produce massive activation (MA), the outlier exists within the first text token.
- x1 and y1 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 x2, y2.

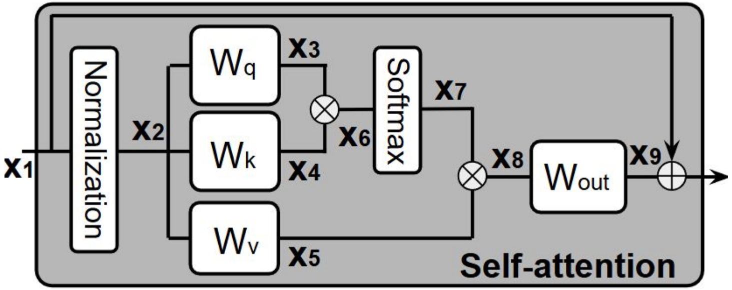


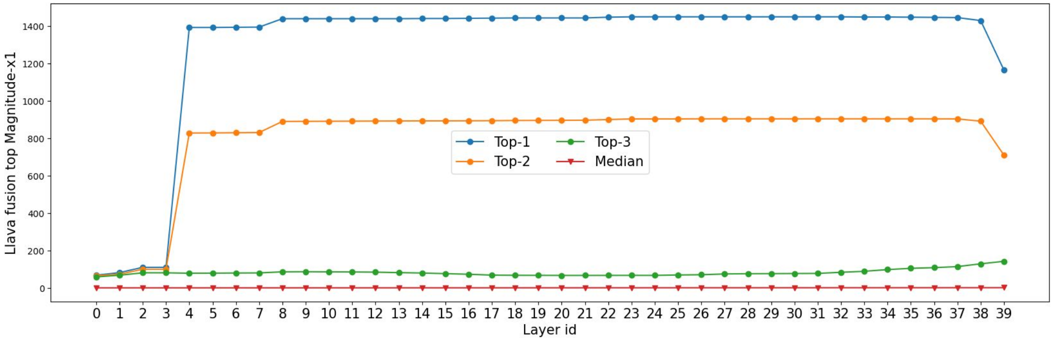
Study on Llama-2 7B

NYU SAI LAB

Detailed Study of Massive Activation

- Top magnitude of x1 across layers.

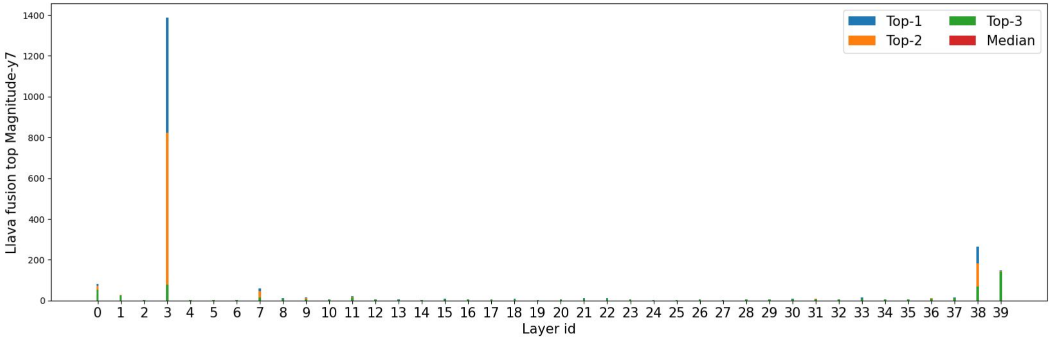


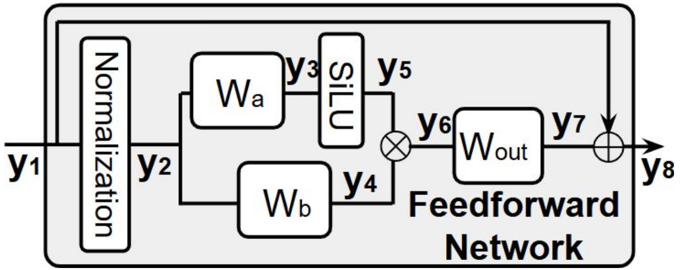


NYU SAI LAB

Detailed Study of Massive Activation

- Top magnitude of y7 across layers.



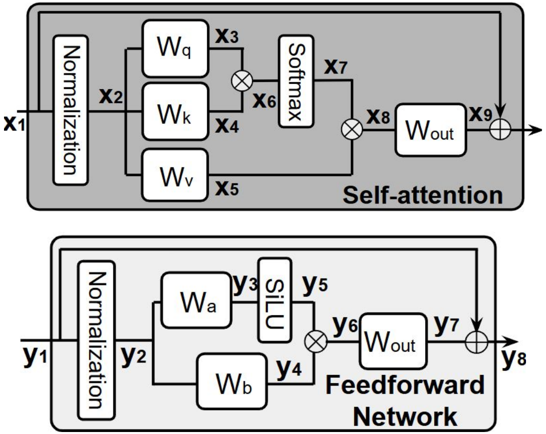


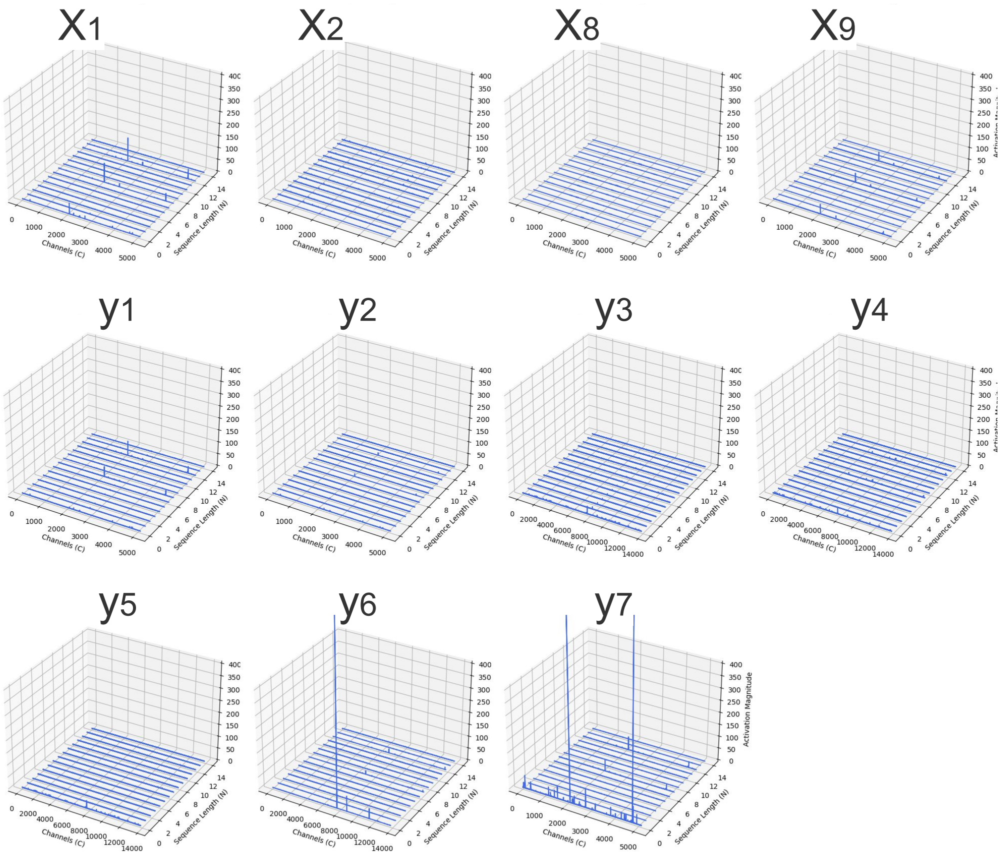
- Truncating the massive activation that caused by residual connection will not lead to large accuracy drop.

NYU SAI LAB

Detailed Study of Massive Activation

- Distribution of activation within layer 3.
- MA first appears in y6 of token=0.



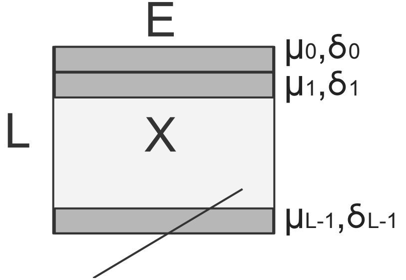


NYU SAI LAB

How Channelwise Outlier Forms?

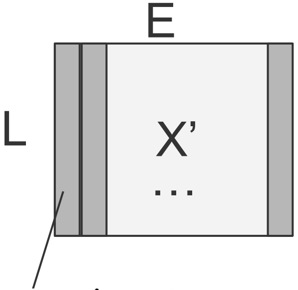
- X1 already has some channelwise outlier, but not significant enough.
- Partial channelwise outlier forms after standardization.
- The scaling process greatly contributes to part of the channelwise outliers.

Step 1

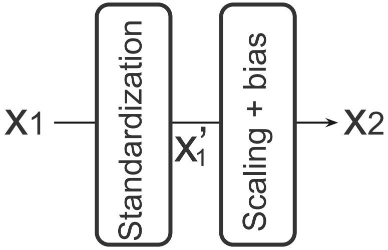


$X' = \frac{X_r - \mu_r}{\sigma_r}$ For each r∈L

Step 2

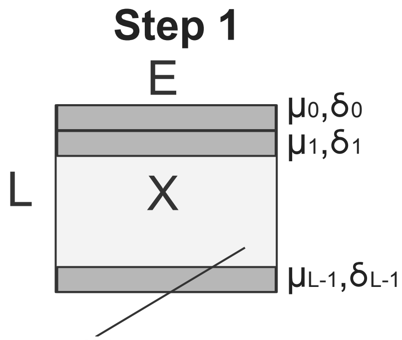


$Y_e = \alpha_e X'_e + \beta_e$ For each e∈E

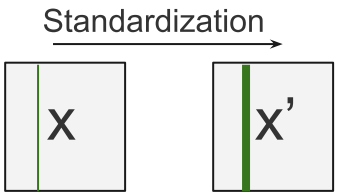


NYU SAI LAB

How Channelwise Outlier Forms?



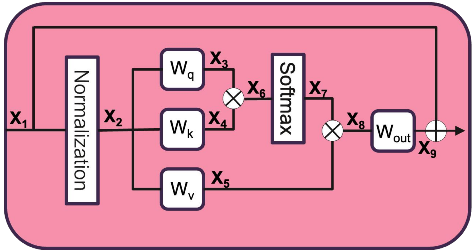
$$X' = \frac{X_r - \mu_r}{\sigma_r}$$ For each r∈L

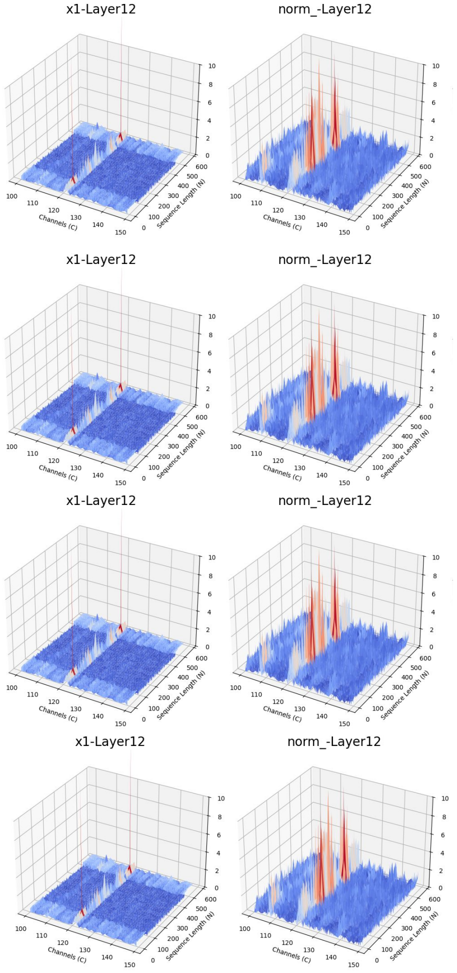


- When the standard deviation is low, channel-wise outliers become more pronounced.

NYU SAI LAB

How Channelwise Outlier Forms?



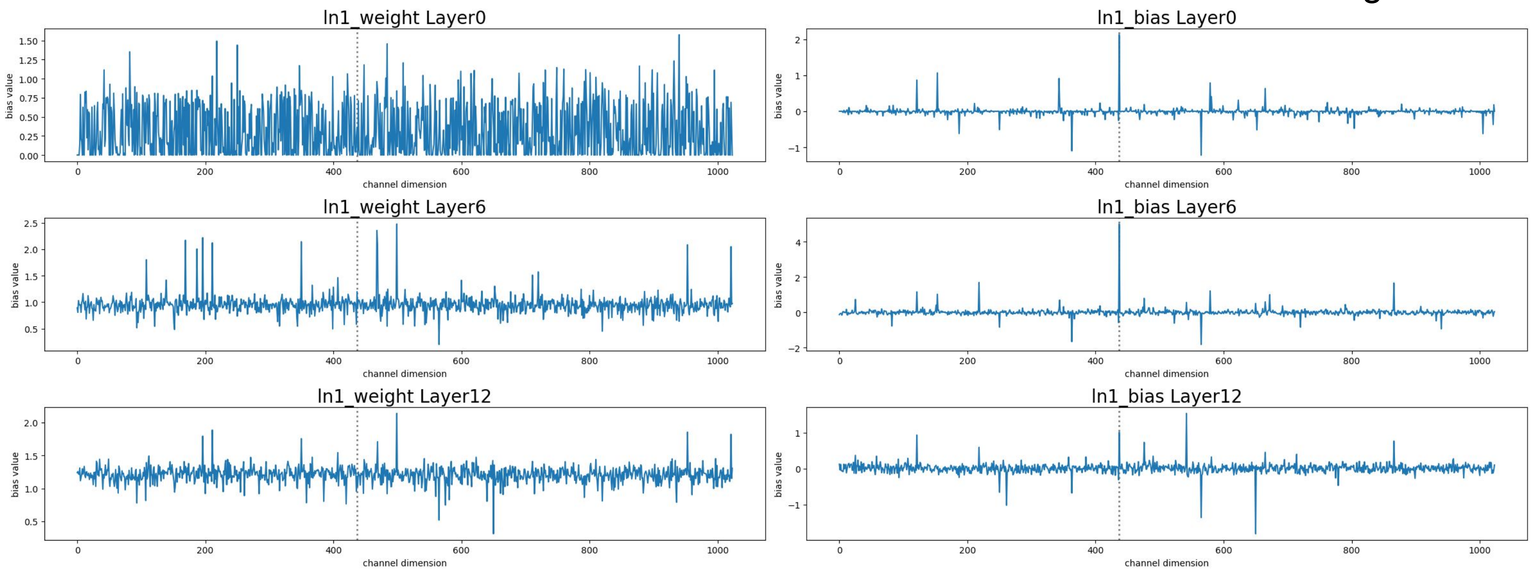


- X1 typically already contains some channelwise outliers.
- Following standardization, the outlier's magnitude becomes more pronounced.

NYU SAI LAB

How Channelwise Outlier Forms?

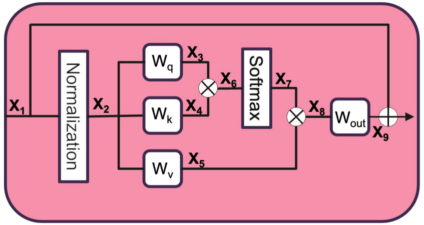
Profiling on CLIP

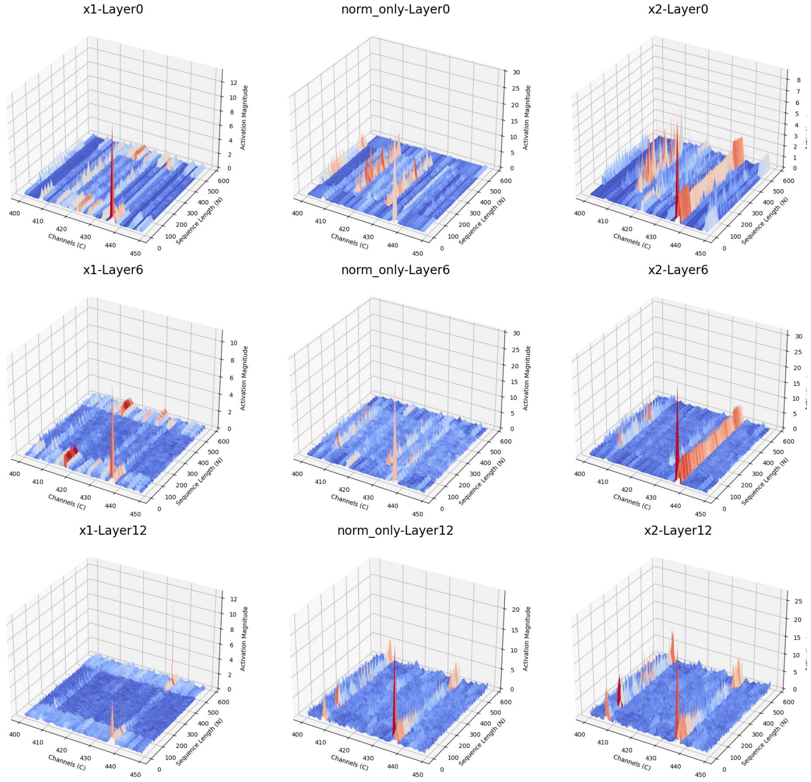


- The scaling and bias factors within the normalization layer also contribute to the channelwise outlier.

NYU SAI LAB

How Channelwise Outlier Forms?

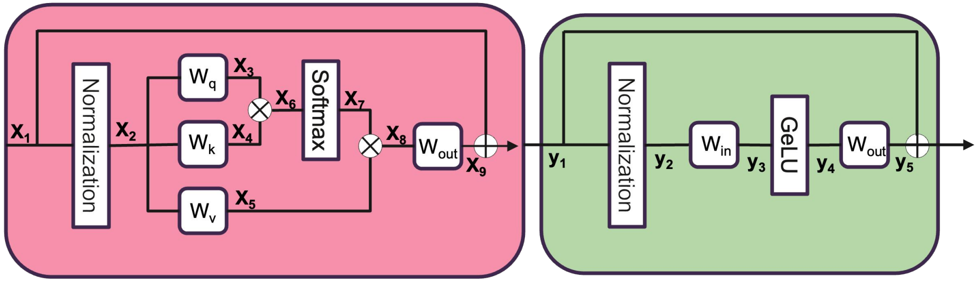


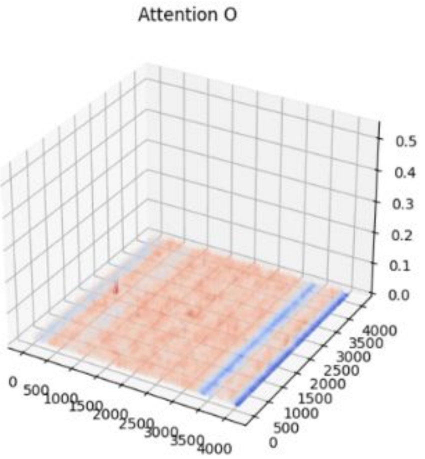


- The scaling and bias factors within the normalization layer also contribute to the channelwise outlier.

NYU SAI LAB

How Channelwise Outlier Forms?

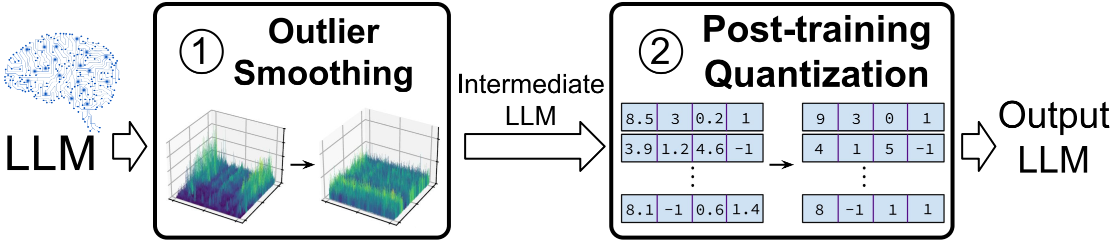




- The question then arises: why does x1 already exhibit some channelwise outliers?
- This is attributed to Wout, which results in y5 having some minor channelwise outliers.
- Additionally, the residual link carries intermediate activations with channelwise outliers, which are also generated by the Wout from preceding layers.

NYU SAI LAB

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.

NYU SAI LAB

Topics

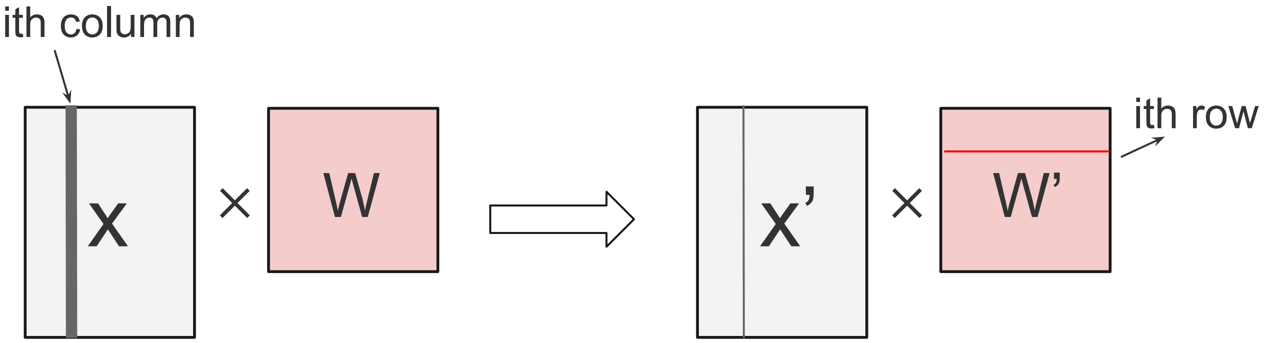
- Large Model Data Distribution
- Large Model Quantization
- Large Model Pruning
- Low-rank Decomposition for LLM

NYU SAI LAB

Topics

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

SmoothQuant



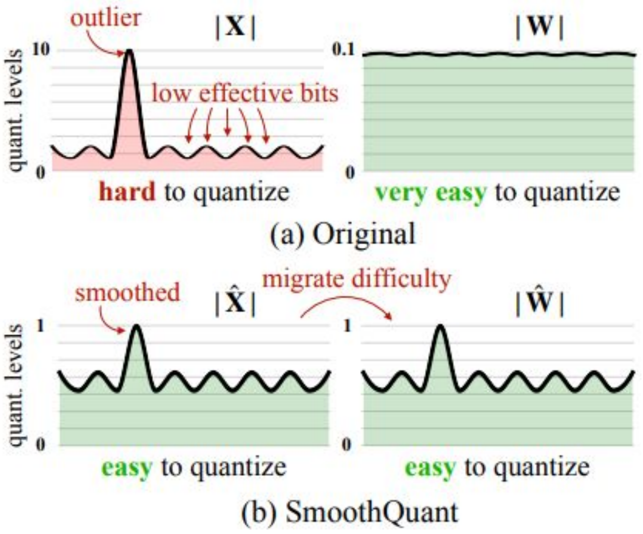
Y = Q(X)Q(W)
Large quantization error

Y = Q(X')Q(W')
Low quantization error

NYU SAI LAB

Xiao, Guangxuan, et al. "Smoothquant: Accurate and efficient post-training quantization for large language models." *International Conference on Machine Learning*. PMLR, 2023.

SmoothQuant

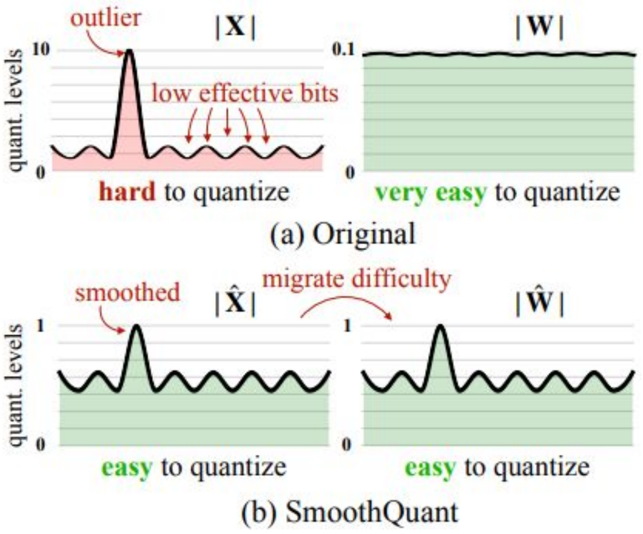


- The intermediate results within LLM usually have a lot of outliers.
- SmoothQuant smooths the activation outliers by offline migrating the quantization difficulty from activations to weights with a mathematically equivalent transformation.

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

Xiao, Guangxuan, et al. "Smoothquant: Accurate and efficient post-training quantization for large language models." *International Conference on Machine Learning*. PMLR, 2023.

SmoothQuant



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

- α is a hyperparameter.

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

- α is determined from the calibration dataset.

NYU SAI LAB

Xiao, Guangxuan, et al. "Smoothquant: Accurate and efficient post-training quantization for large language models." *International Conference on Machine Learning*. PMLR, 2023.

Activation-Aware Weight Quantization (AWQ)

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

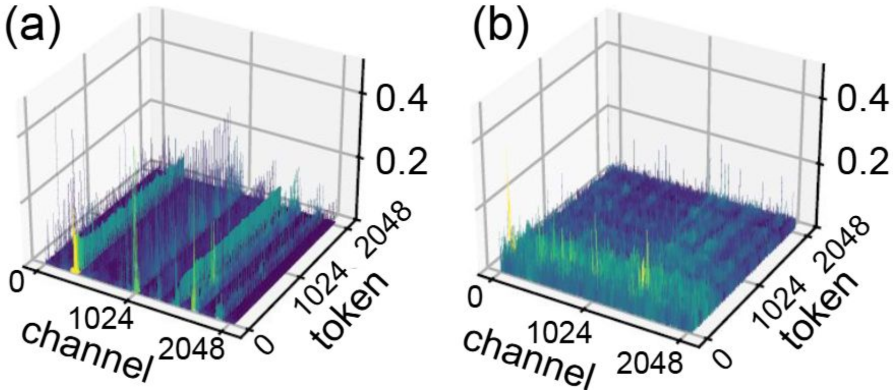
$$\mathcal{L}(\mathbf{s}) = \|Q(\mathbf{W} \cdot \mathrm{diag}(\mathbf{s}))(\mathrm{diag}(\mathbf{s})^{-\mathbf{1}} \cdot \mathbf{X}) - \mathbf{W}\mathbf{X}\|$$

- 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**

Lin, Ji, et al. "AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration." *Proceedings of Machine Learning and Systems* 6 (2024): 87-100.

QuaRot



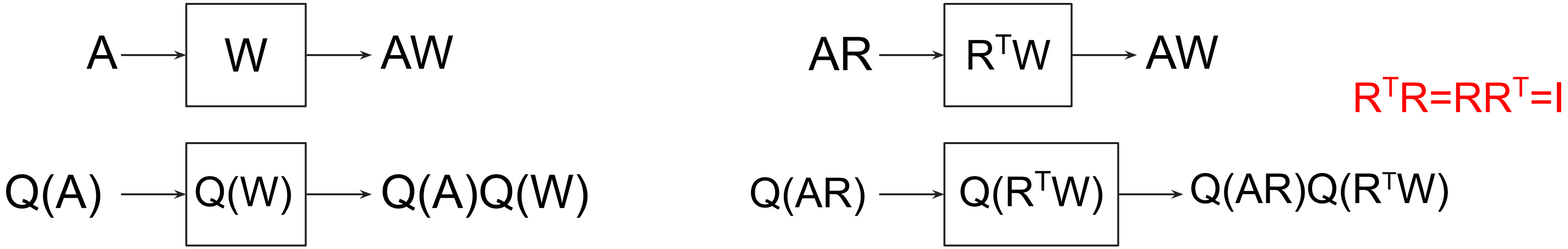
- QuaRot introduces a novel methods to convert the weights and activation of LLM.
- After conversion, most of the outliers within the activation and weights are removed.
- This conversion introduces almost no additional cost during the inference.

NYU SAI LAB

Ashkboos, Saleh, et al. "Quarot: Outlier-free 4-bit inference in rotated llms." *arXiv preprint arXiv:2404.00456* (2024).

QuaRot

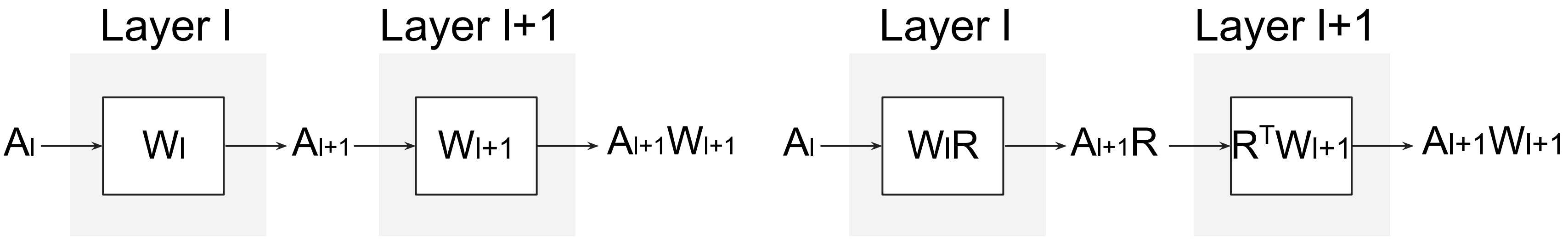
- Assume Y = AW, where A may have outliers, quantizing A and W as Q(A) and Q(W) could result in increased quantization error. Consequently, Q(A)Q(W) may differ significantly from AW.
- With QuaRot, a orthogonal matrix is applied to eliminate the outliers within A.



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

Ashkboos, Saleh, et al. "Quarot: Outlier-free 4-bit inference in rotated llms." *arXiv preprint arXiv:2404.00456* (2024).

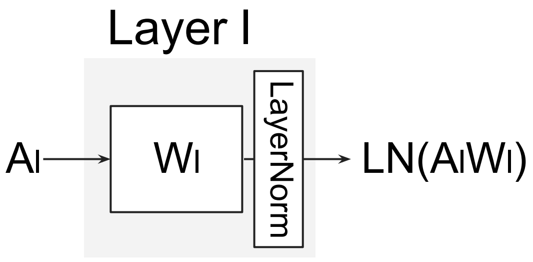
QuaRot

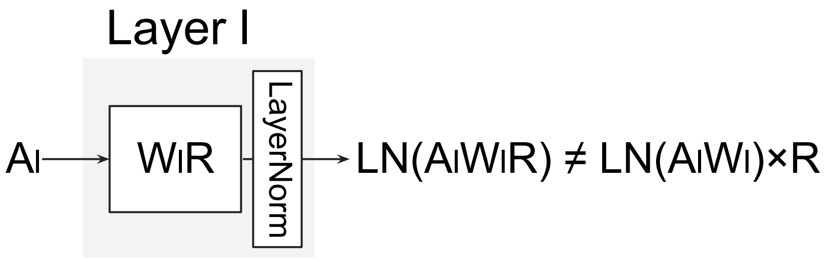


- R^TW can be computed offline, AR can be generated by modifying the weight matrices of the last layer.

Ashkboos, Saleh, et al. "Quarot: Outlier-free 4-bit inference in rotated llms." *arXiv preprint arXiv:2404.00456* (2024).

QuaRot

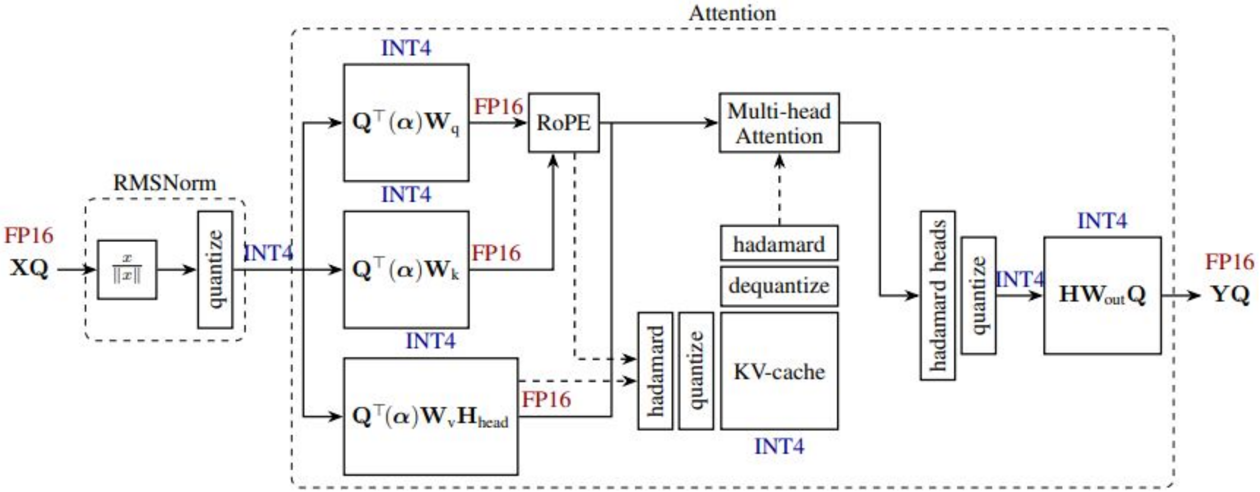




- However, if a nonlinear function follows the linear layers, the Hadamard transform must be computed dynamically.

Ashkboos, Saleh, et al. "Quarot: Outlier-free 4-bit inference in rotated llms." *arXiv preprint arXiv:2404.00456* (2024).

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.

NYU SAI LAB

Ashkboos, Saleh, et al. "Quarot: Outlier-free 4-bit inference in rotated llms." *arXiv preprint arXiv:2404.00456* (2024).

DuQuant

$P^TP=PP^T=I$

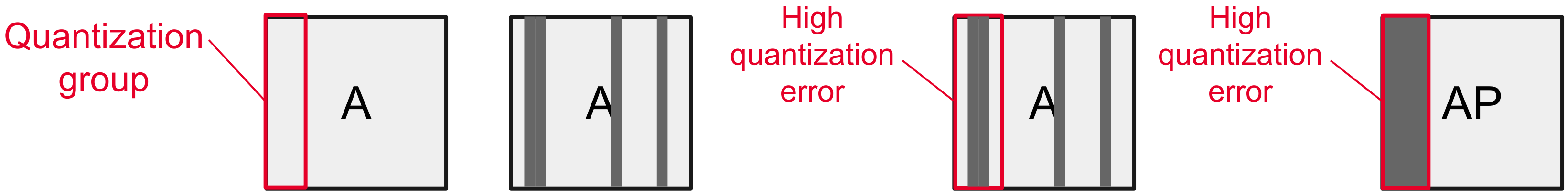
- Permutation matrix is another types of matrix which cause no change on the output.
- Duquant combines three types of computational invariance operation, including the scalar-based, rotation-based and permutation-based operations for mitigating the outliers.

Lin, Haokun, et al. "Duquant: Distributing outliers via dual transformation makes stronger quantized llms." *Advances in Neural Information Processing Systems* 37 (2024): 87766-87800.

DuQuant

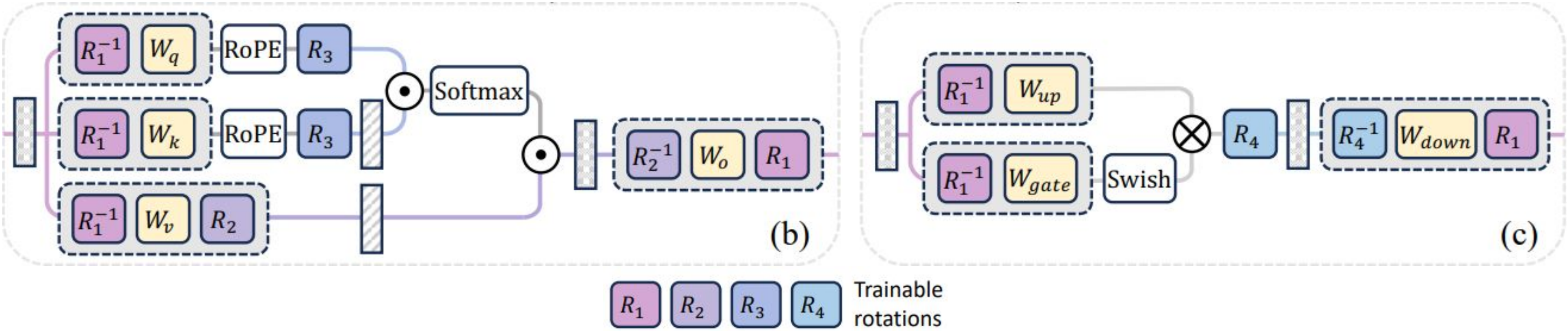
$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad A' = AP = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix}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

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



Lin, Haokun, et al. "Duquant: Distributing outliers via dual transformation makes stronger quantized llms." *Advances in Neural Information Processing Systems* 37 (2024): 87766-87800.

SpinQuant



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

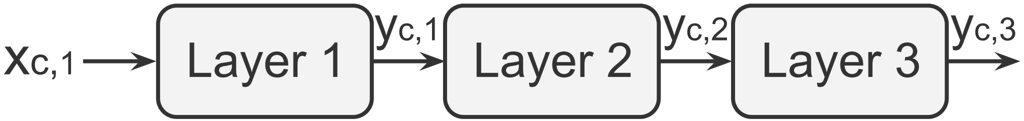
- SpinQuant optimizes (or learns) the rotation matrices to obtain the minimal changes on the training loss.
- We have to ensure the rotational matrix still satisfies the orthogonal property → Cayley Optimization.

Liu, Zechun, et al. "SpinQuant--LLM quantization with learned rotations." *arXiv preprint arXiv:2405.16406* (2024).

Topics

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

Parallel Update

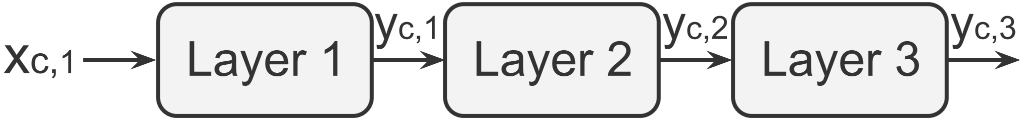


$$\min_{\theta_1} \sum_{x_{c,1} \in D} ||y_{c,1} - F^1_{\theta_1}(x_{c,1})||^2$$

- 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.

NYU SAI LAB

Sequential Update

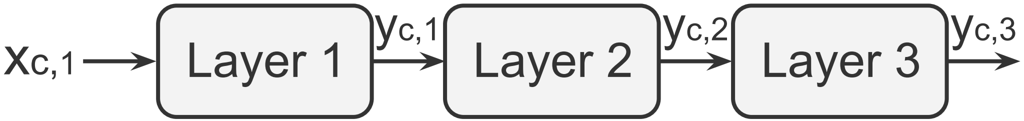


$$\min_{\theta_1} \sum_{x_{c,1} \in D} ||y_{c,1} - F^1_{\theta_1}(x_{c,1})||^2$$

- 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.

NYU SAI LAB

Sequential Update



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

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

- 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.

NYU SAI LAB

AdaQuant

$$\left(\hat{\Delta}_w, \hat{\Delta}_x, \hat{V}\right) = \underset{\Delta_w, \Delta_x, V}{\arg\min} ||WX - Q_{\Delta_w}(W')Q_{\Delta_x}(X)||^2$$

AdaQuant

$$\left(\hat{\Delta}_{w_l}, \hat{\Delta}_{x_l}, \hat{V_l}\right) = \underset{\Delta_{w_l}, \Delta_{x_l}, V_l}{\arg\min} ||W_l X_l - Q_{\Delta_{w_l}}(W_l') \cdot Q_{\Delta_{x_l}}(X_l^q)||^2$$

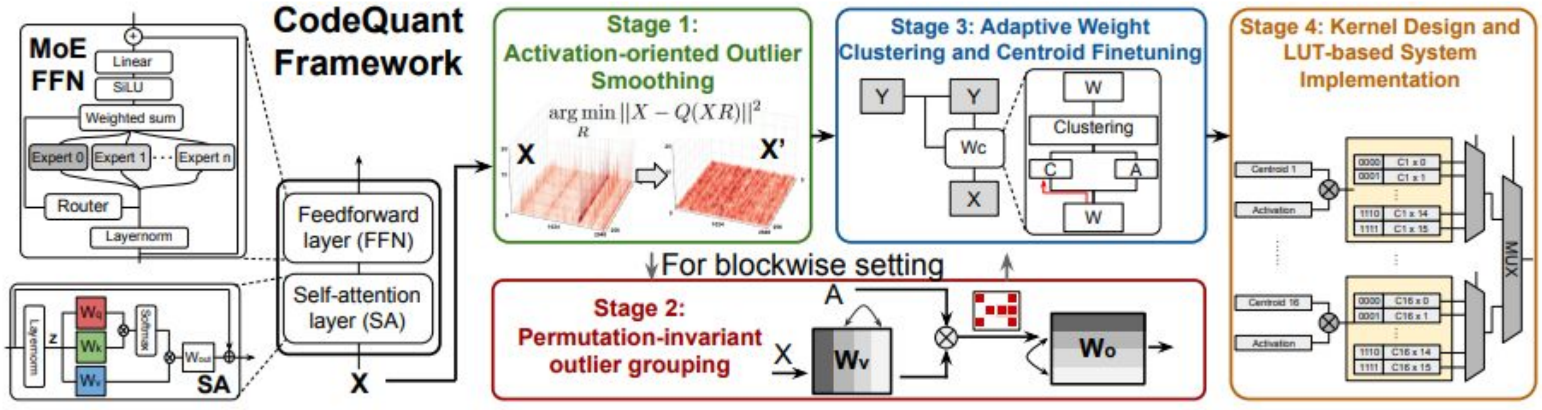
$$X_q = \sigma(Q_{\Delta_{w_{l-1}}}(W_{l-1} + V_{l-1}) \cdot Q_{\Delta_{x_l}}(X_{l-1}^q)),$$

Sequential AdaQuant

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

Hubara, Itay, et al. "Accurate post training quantization with small calibration sets." *International Conference on Machine Learning*. PMLR, 2021.

CodeQuant

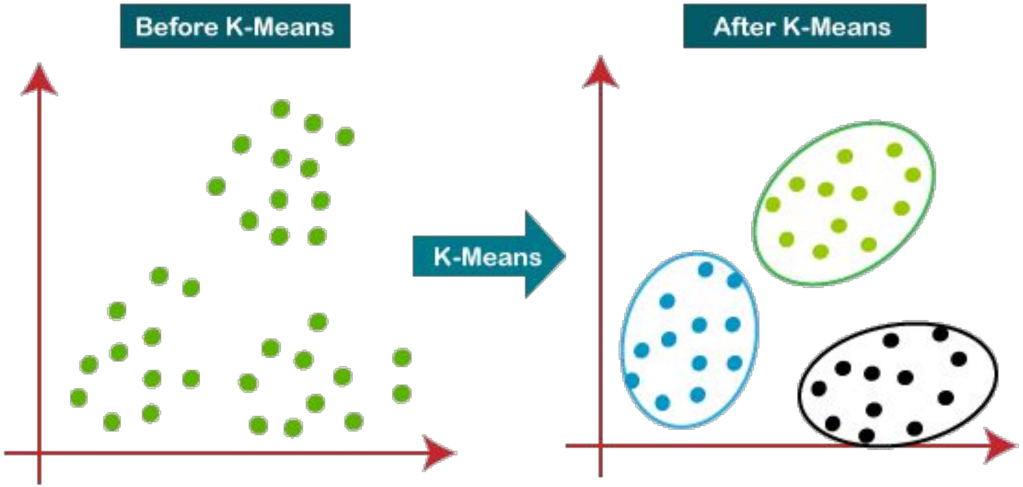


- Compared to quantization, the clustering operation exhibits greater tolerance to outliers.

NYU SAI LAB

Xiangyang Yin, et al. "CodeQuant: Unified Clustering and Quantization for Enhanced Outlier Smoothing in Low-Precision Mixture-of-Experts." *ICLR 2026*.

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).

NYU SAI LAB

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)**

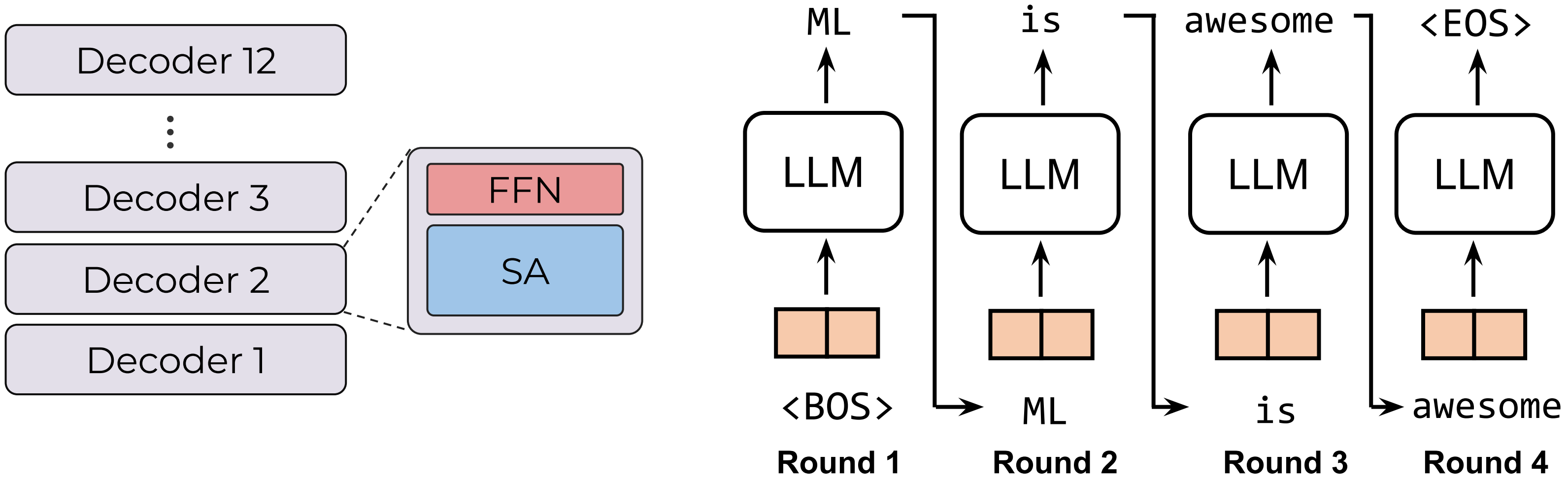
- Clustering can be used to hand excessive outlier.

NYU SAI LAB

Topics

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

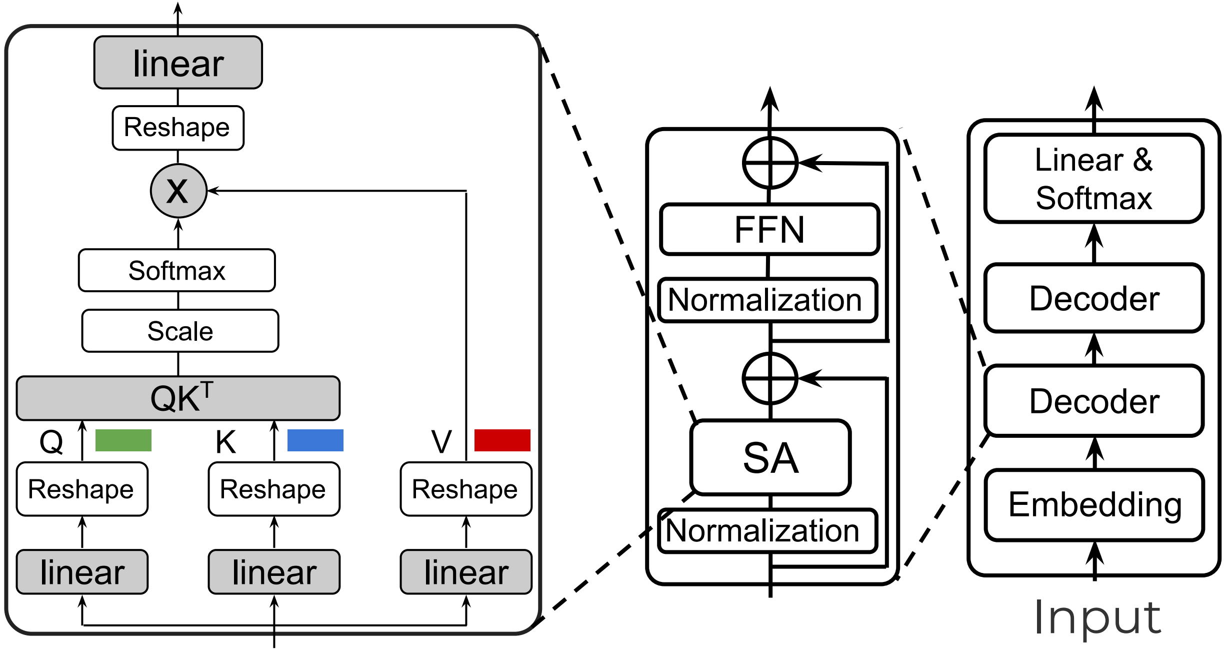
Transformers as a Generative AI Tool



- Each token is generated in an autoregressive manner.

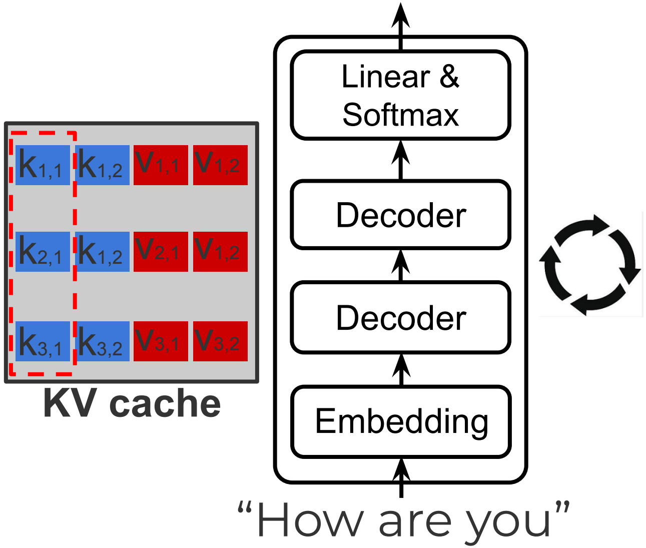
Radford, Alec, et al. "Language models are unsupervised multitask learners." *OpenAI blog* 1.8 (2019): 9.

Transformers as a Generative AI Tool



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

LLM: Prefilling

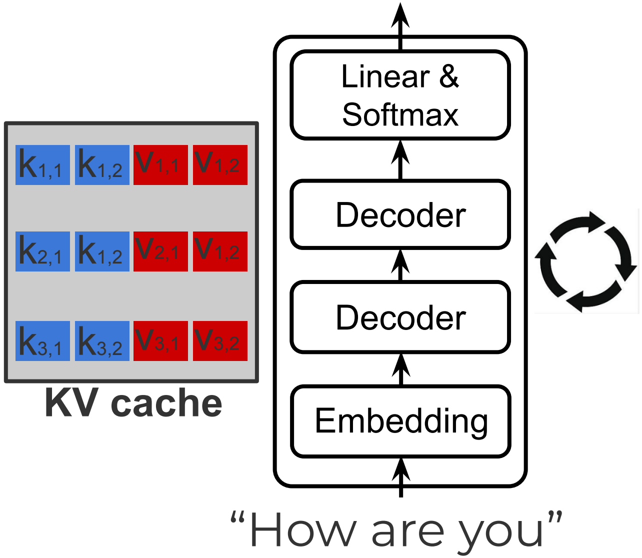


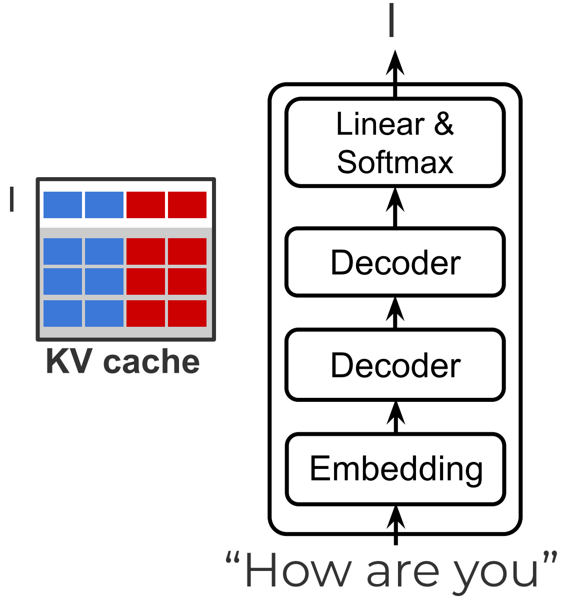
$k_{i,j}$ Key vector for ith token in jth layer (1×E)

$V_{i,j}$ Value vector for ith token in jth layer (1×E)

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

LLM: Decoding

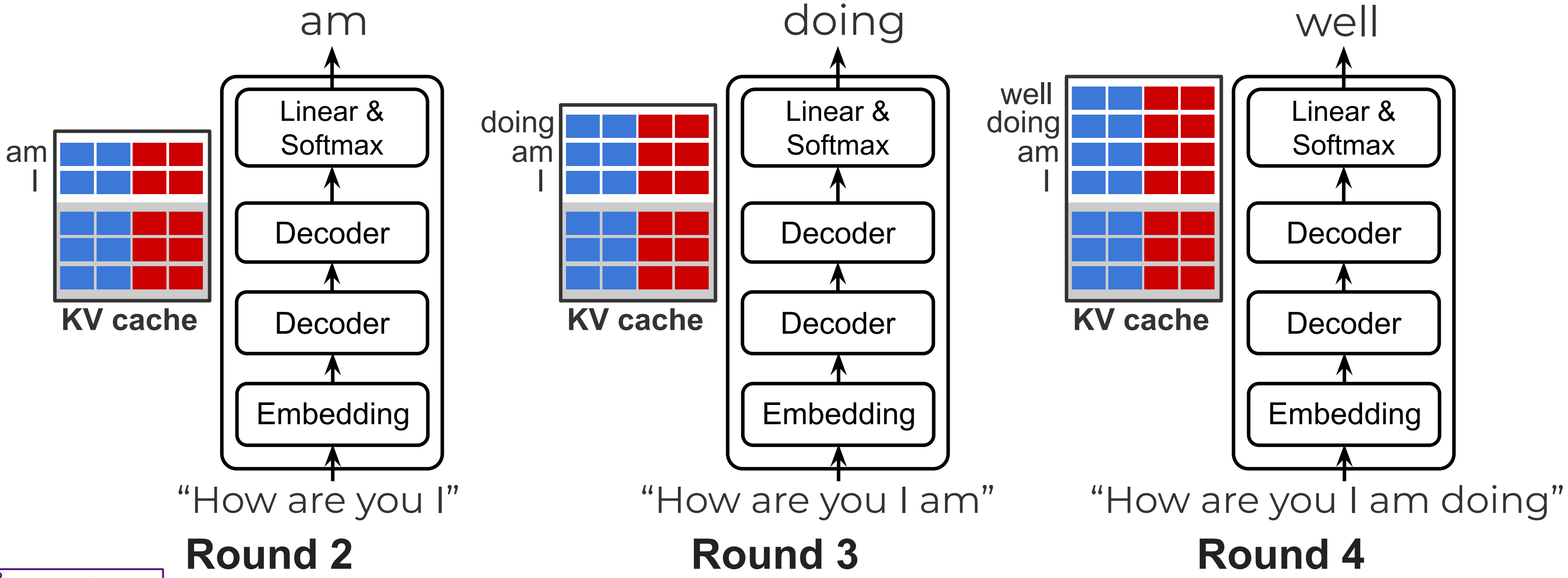




Round 1

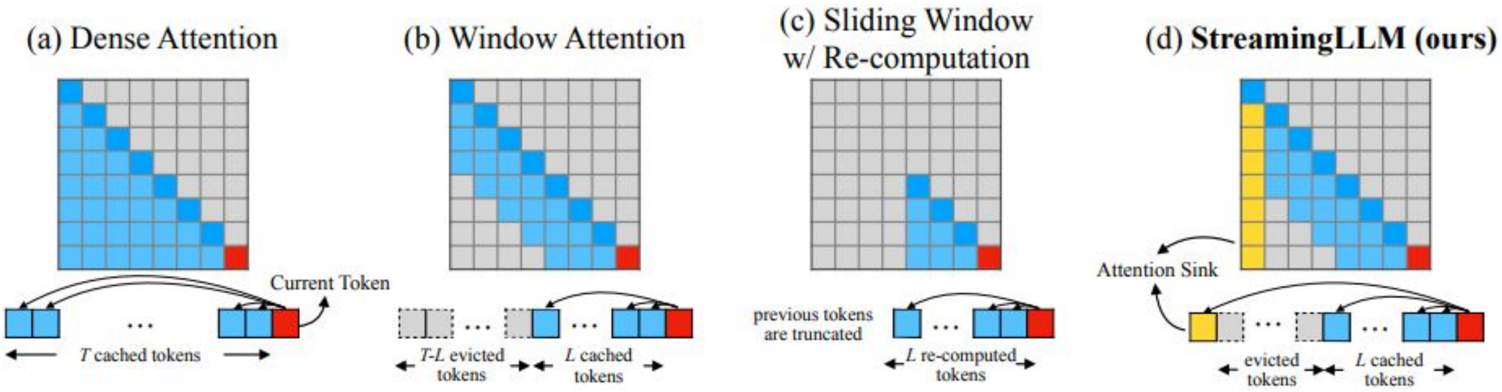
- During the decoding stage, LLM generates the responses in an autoregressive way.

LLM: Decoding



NYU SAI LAB

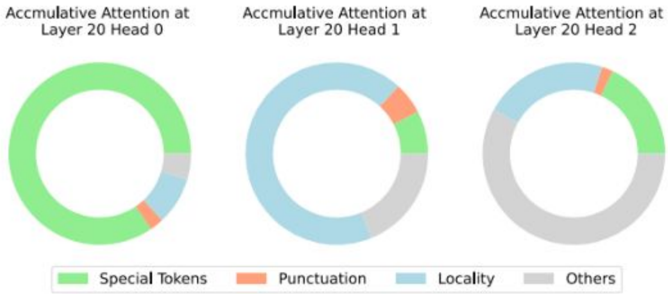
Streaming-LLM

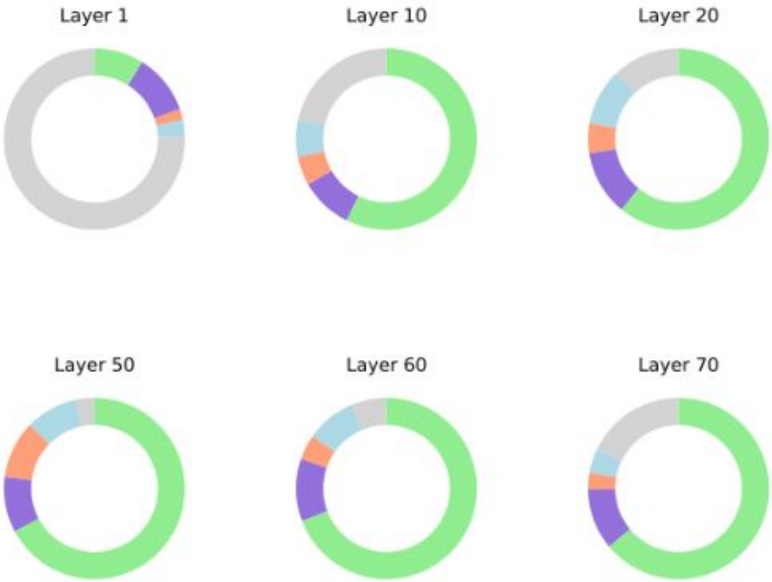


- We observe an interesting phenomenon, namely attention sink, that keeping the KV of initial tokens will largely recover the performance of window attention.
- There are strong attention scores towards initial tokens as a "sink" even if they are not semantically important.

Xiao, Guangxuan, et al. "Efficient streaming language models with attention sinks." *arXiv preprint arXiv:2309.17453* (2023).

Pruning on Large Models: KV Cache Pruning



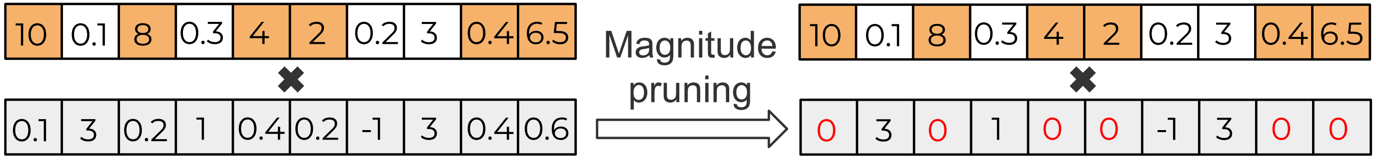


- We show the attention score of each token.
- Different attention heads usually have different importance scores on KV vectors.
- The importance of KV vectors also varies across layers.

Ge, Suyu, et al. "Model tells you what to discard: Adaptive kv cache compression for llms." *arXiv preprint arXiv:2310.01801* (2023).

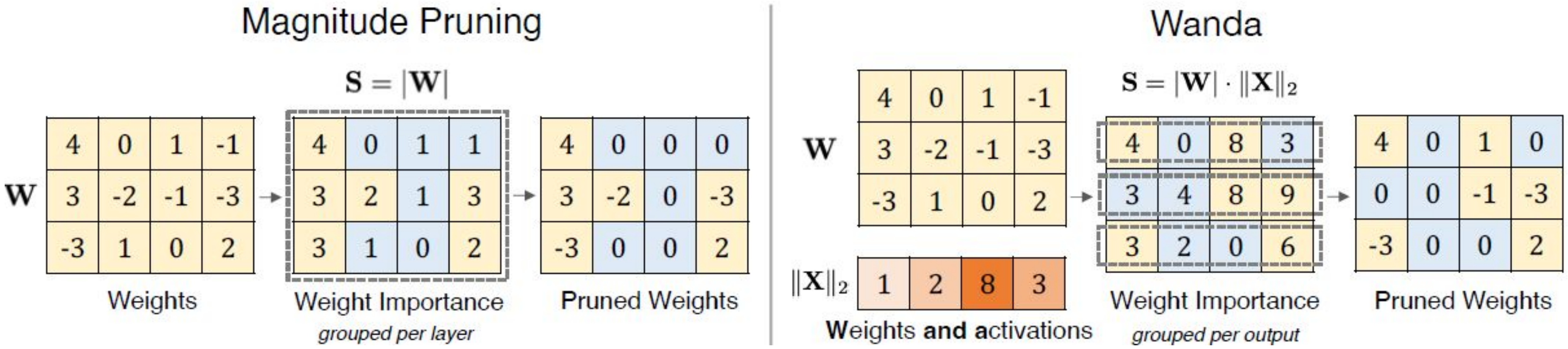
Drawbacks of Magnitude Pruning

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



NYU SAI LAB

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.

Sun, Mingjie, et al. "A simple and effective pruning approach for large language models." *arXiv preprint arXiv:2306.11695* (2023).

LLM Pruning: Wanda

			LLaMA				LLaMA-2		
Method	Weight Update	Sparsity	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.

NYU SAI LAB

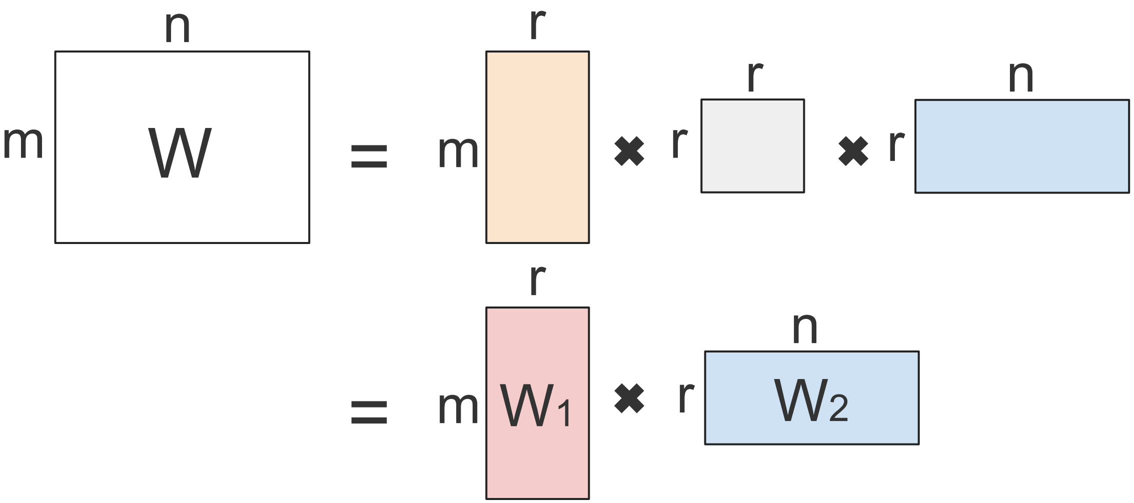
Topics

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

NYU SAI LAB

Low Rank Optimization for DNN Efficiency

- Weight tensors can be decomposed into:

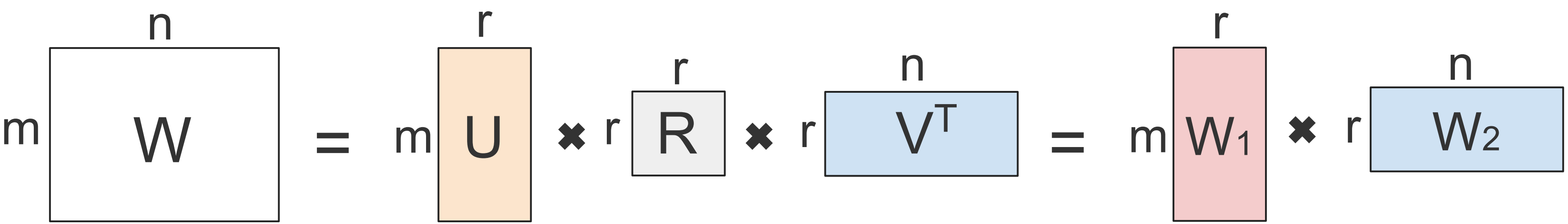


r<=min(m,n)

- We can train the W_1 and W_2 in the DNN instead of W.
- Less storage is required.

NYU SAI LAB

Singular Value Decomposition

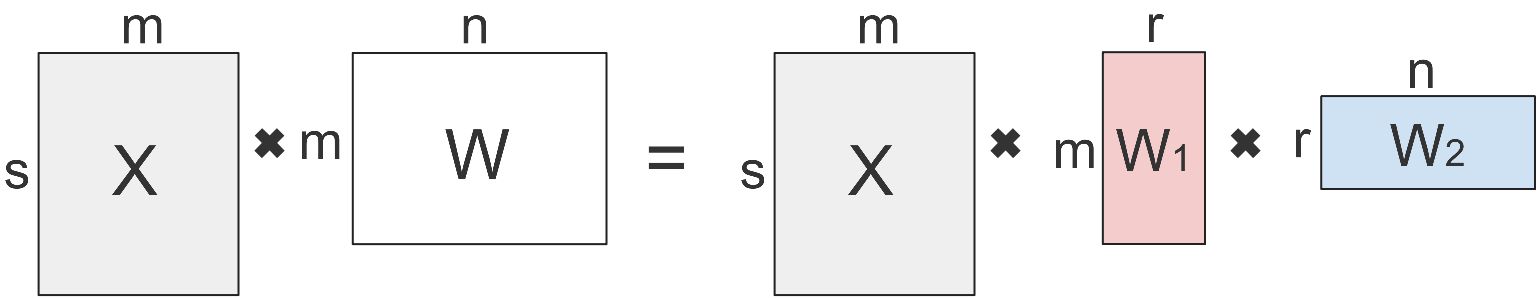


- W: Input matrix
  - m✖n matrix
- V (n✖r matrix) contains the orthonormal eigenvectors of W^TW
- U (m✖r matrix) contains the orthonormal eigenvectors of WW^T
- R: Singular value matrix
  - r✖r diagonal matrix, r is the rank of W

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



Computational cost=smn

Computational cost = smr + snr

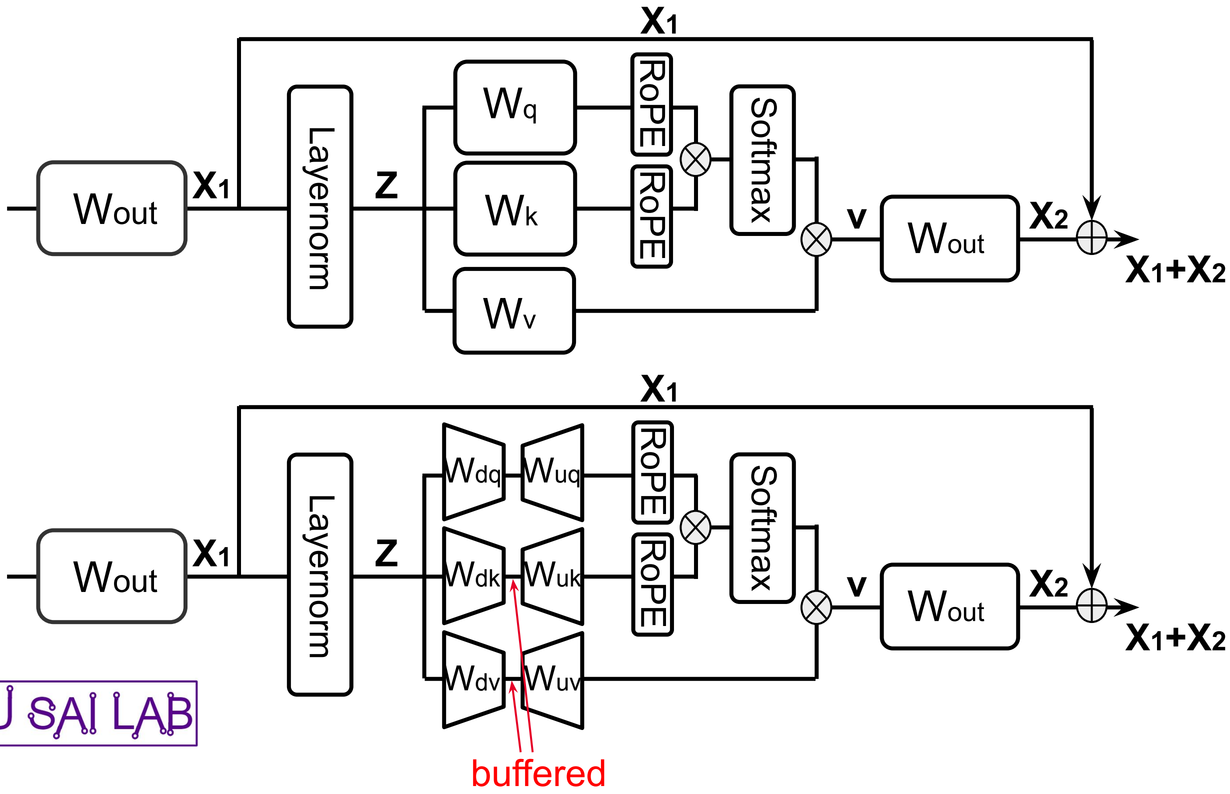
= s(m+n)r

= s(m+n)min(m,n)

- Without rank truncation, the computational cost will increase.

NYU SAI LAB

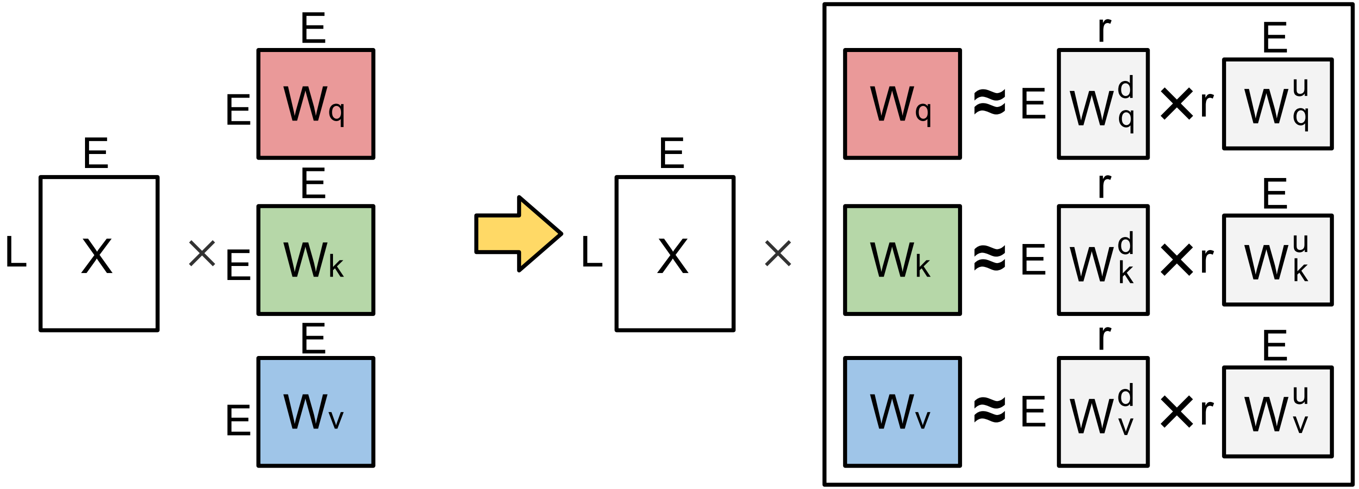
Singular Value Decomposition



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

NYU SAI LAB

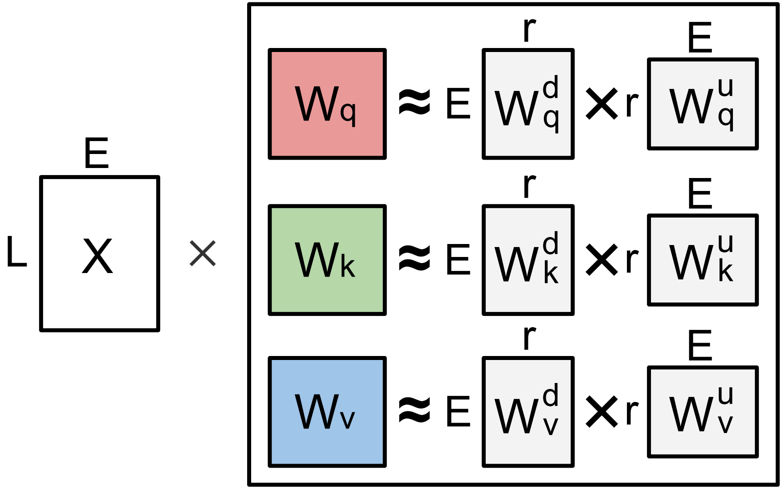
QSVD



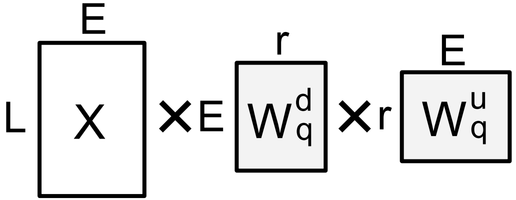
Total MACs: $3LE^2$
Total weight parameters: $3E^2$
Total cache size: 2LE

Total MACs: 6LrE
Total weight parameters: 6rE
Total cache size: 2rL

QSVD



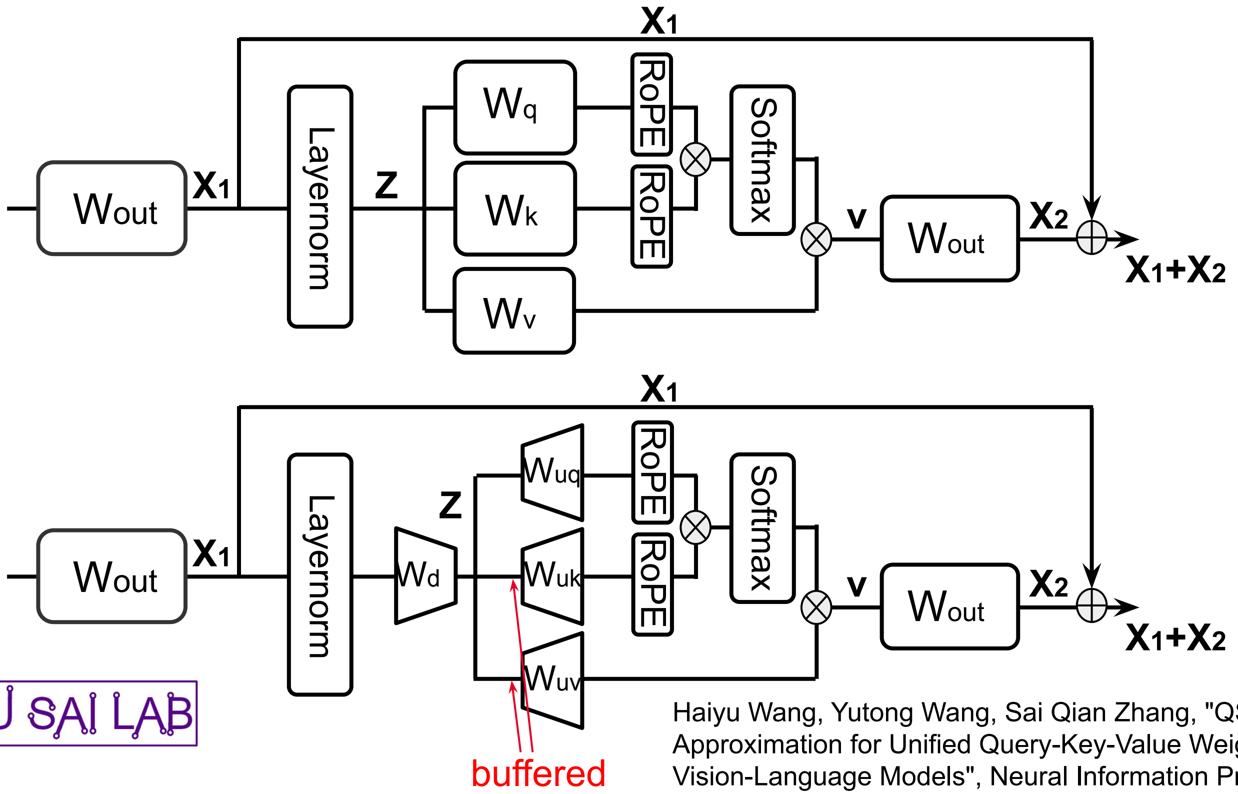
Total MACs: 6LrE
Total weight parameters: 6rE
Total cache size: 2rL



Total MACs: 6LrE
Total weight parameters: 6rE
Total cache size: 2rL

NYU SAI LAB

QSVD

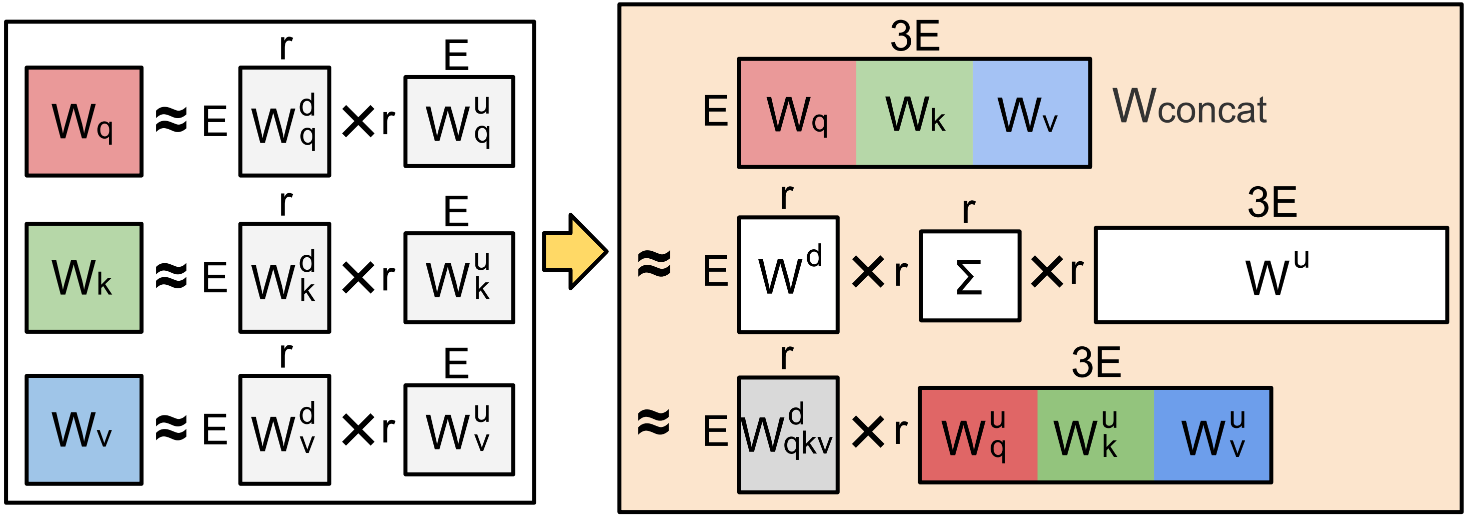


SVD decomposition can potentially save the MAC operations, memory storage and KV cache size.

NYU SAI LAB

Haiyu Wang, Yutong Wang, Sai Qian Zhang, "QSVD: Efficient Low-rank Approximation for Unified Query-Key-Value Weight Compression in Low-Precision Vision-Language Models", Neural Information Processing Systems (NeurIPS), 2025.

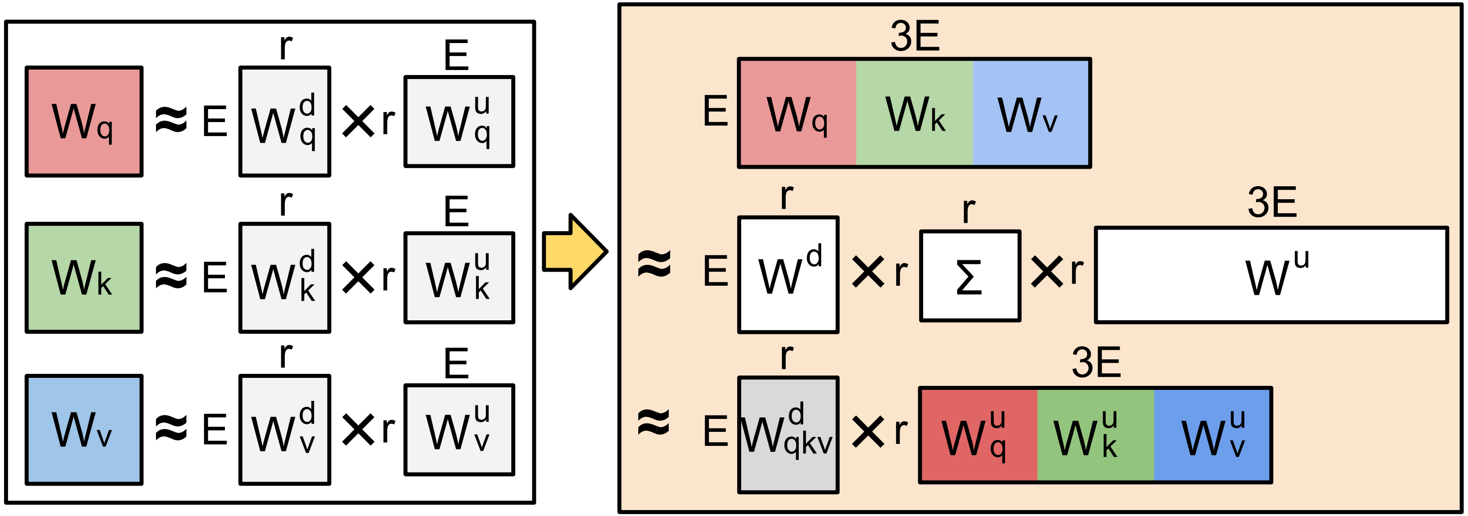
QSVD



- Concat and SVD

$$[W_q,\ W_k,\ W_v] = W_{concat} \approx W^d \times \Sigma \ \times W^u$$

QSVD



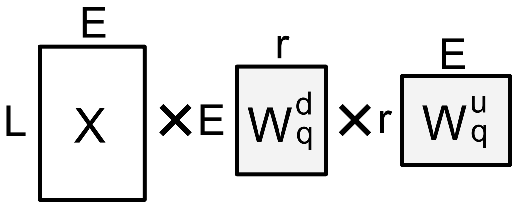
- Down/up projection

$$W^d_{qkv} = W^d_r \Sigma_r^{1/2},\ [W^u_q,\ W^u_k,\ W^u_v] = \Sigma_r^{1/2} W^u_r$$

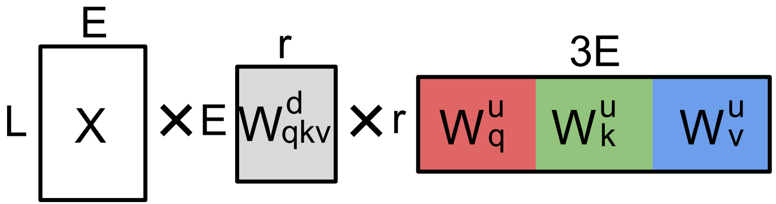
$$[W_q,\ W_k,\ W_v] = W^d_{qkv} \times [W^u_q,\ W^u_k,\ W^u_v]$$

NYU SAI LAB

QSVD

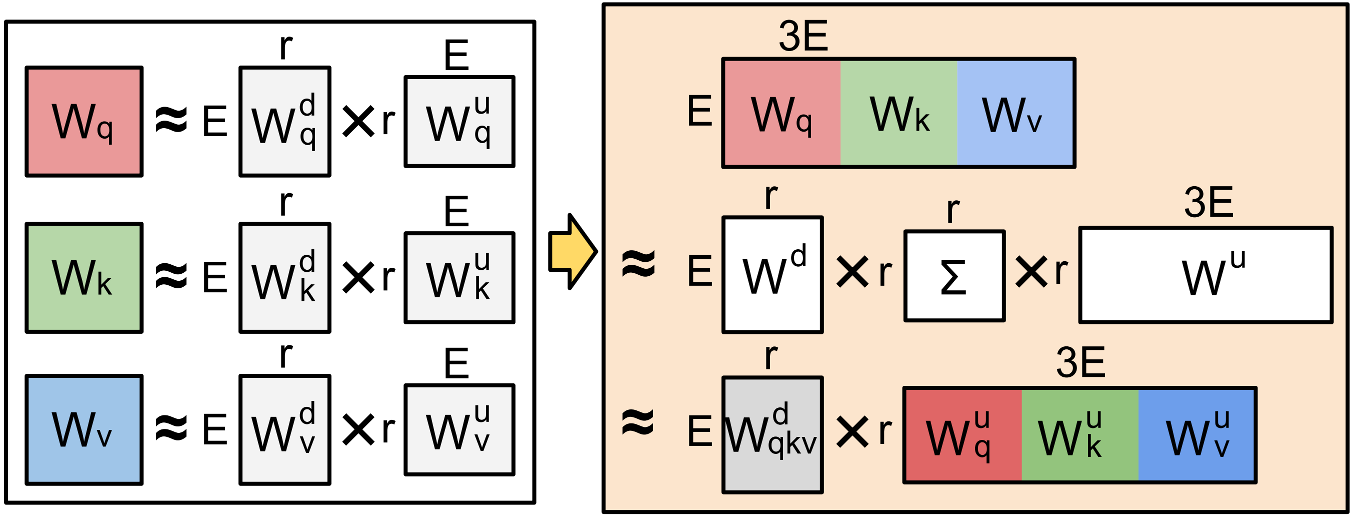


Total MACs: 6LrE
Total weight parameters: 6rE
Total cache size: 2rL



Total MACs: 4LrE
Total weight parameters: 4rE
Total cache size: rL

QSVD

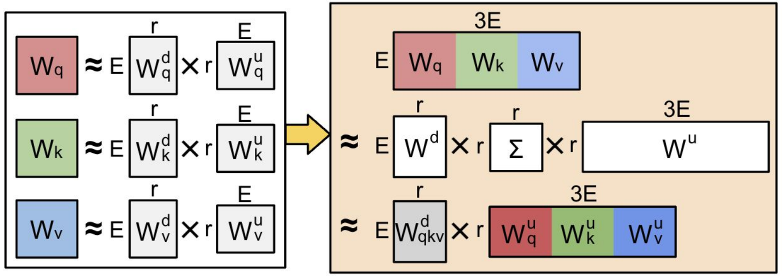


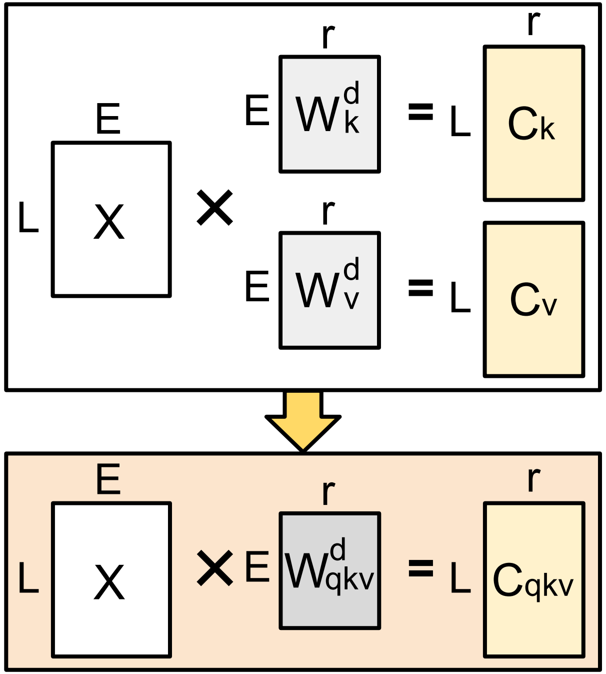
Total MACs: 6LrE
Total weight parameters: 6rE
Total KV cache size: 2rL

Total MACs: 4LrE
Total weight parameters: 4rE
Total KV cache size: rL

NYU SAI LAB

QSVD



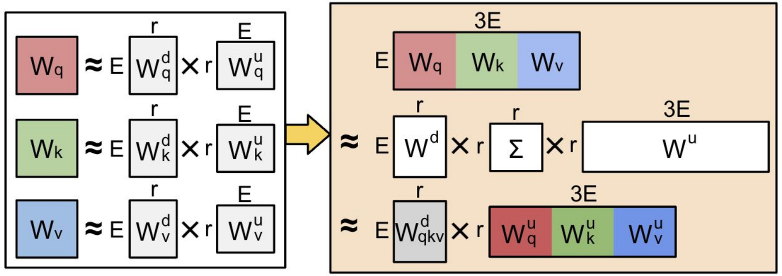


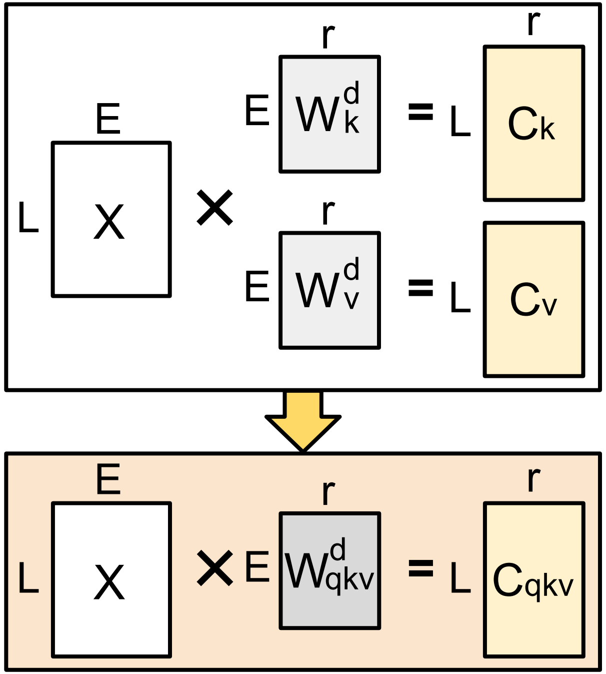
- Shared latent

$$[Q,\ K,\ V] = XW^d_{qkv} \times [W^u_q,\ W^u_k,\ W^u_v] = C_{qkv} \times [W^u_q,\ W^u_k,\ W^u_v]$$

NYU SAI LAB

QSVD





- Reconstruction

$$K = C_{qkv} W_k^u,\ V = C_{qkv} W_v^u$$

QSVD

- 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.

NYU SAI LAB

QSVD Performance

	Method	ScienceQA-IMG ↑							
		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.

NYU SAI LAB

Problem of SVD

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

$$\underset{W'}{argmin}||W - W'||^2$$
$$W' = DU$$
$$D \in \mathcal{R}^{E\times r}$$
$$U \in \mathcal{R}^{r\times E}$$

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

NYU SAI LAB

WSVD (ICLR'26)

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

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

$$W' = DU$$

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

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

- 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.

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

NYU SAI LAB

Haiyu Wang, Yutong Wang, Sai Qian Zhang, "WSVD: Weighted Low-Rank Approximation for Fast and Efficient Execution of Low-Precision Vision-Language Models", in submission.

Presentation

[MiniLLM: On-Policy Distillation of Large Language Models]
Daniel Enriquez, Nikhil Arora, Man Sharma

NYU SAI LAB