NEURAL INFORMATION PROCESSING SYSTEMS

🎤 DuQuant: Distributing Outliers via Dual Transformation Makes Stronger Quantized LLMs

NeurIPS 2024 Oral

Haokun Lin[*1,3,4], Haobo Xu[*2], Yichen Wu[*4], Jingzhi Cui[2], Yingtao Zhang[2], Linzhan Mou[5], Linqi Song[4], Zhenan Sun[^1,3], Ying Wei[^5]

[*]Equal Contribution [^]Corresponding Authors

[1]School of Artificial Intelligence, University of Chinese Academy of Sciences

[2]Tsinghua University [3]NLPR & MAIS, Institute of Automation, Chinese Academy of Sciences

[4]City University of Hong Kong [5]Zhejiang University

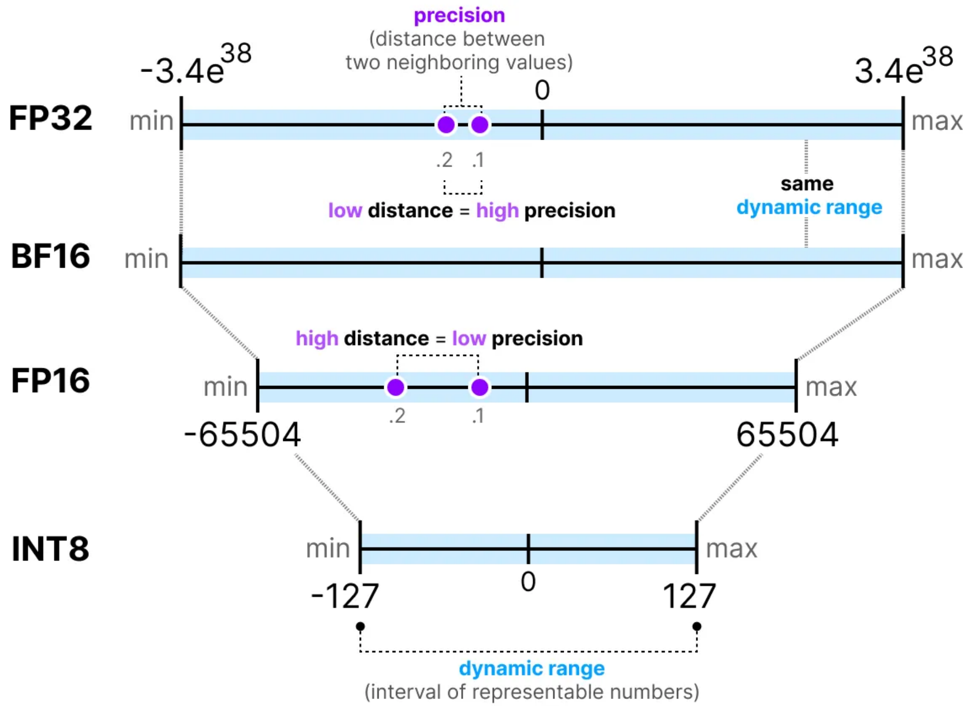
Code

Paper

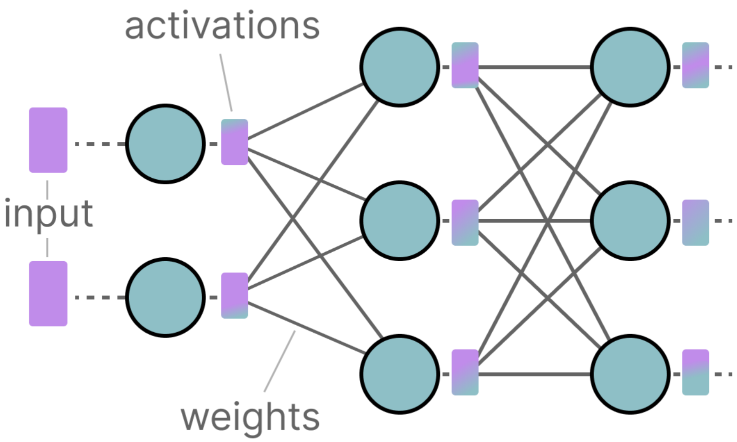
Project: https://duquant.github.io/

Network Quantization

- Network Quantization
  - Reduce redundancy in network representation
  - FP16 --- Low bits storage



- What to quantize?
  - Weights W
  - Activations X
  - Gradients



- The basic Quantization formulation
  - X: Float format
  - X_q: Int format
  - Δ: Scaling factor
  - z: Zero point

$$Quant: \mathbf{X}_q = \text{clamp}\left(\left\lfloor \frac{\mathbf{X}}{\Delta} \right\rceil + z, 0, 2^b - 1\right) \quad De\text{-}quant: \hat{\boldsymbol{X}} = s\left(\mathbf{X}_q - z\right) \approx \boldsymbol{X}$$

Rounding Function
Nearest Rounding

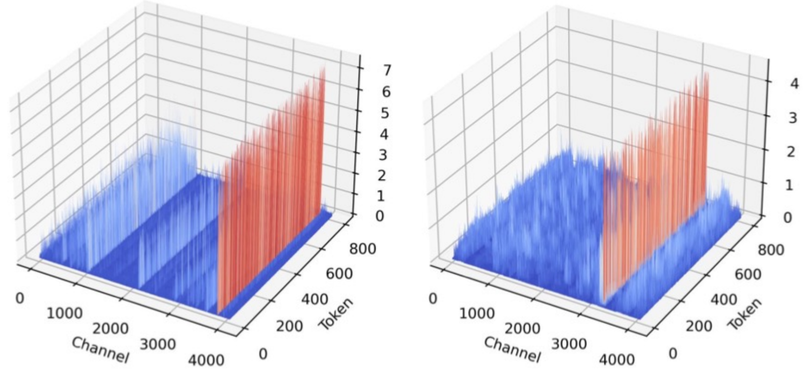
$$\Delta = \frac{\max(\mathbf{X}) - \min(\mathbf{X})}{2^b - 1}, z = -\left\lfloor \frac{\min(\mathbf{X})}{\Delta} \right\rceil$$

Motivation

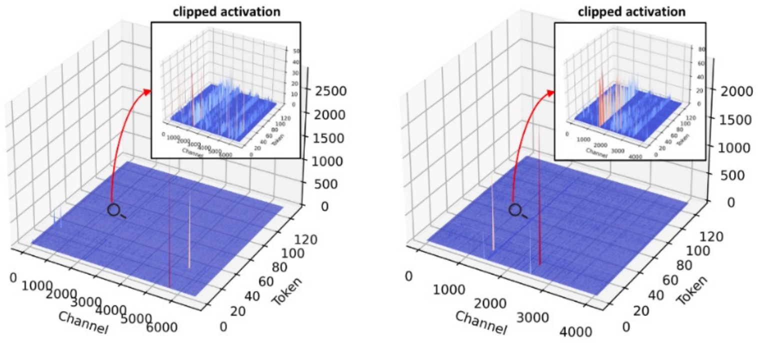
➢ Massive Outliers vs Normal Outliers

- Normal: large values across specific feature dimensions and present in all token sequences
- Massive: exceedingly high values and occur in a subset of tokens

Normal



(a) LLaMA7B_Layer2_Attn_K_Proj (b) LLaMA7B_Layer15_Attn_O_Proj

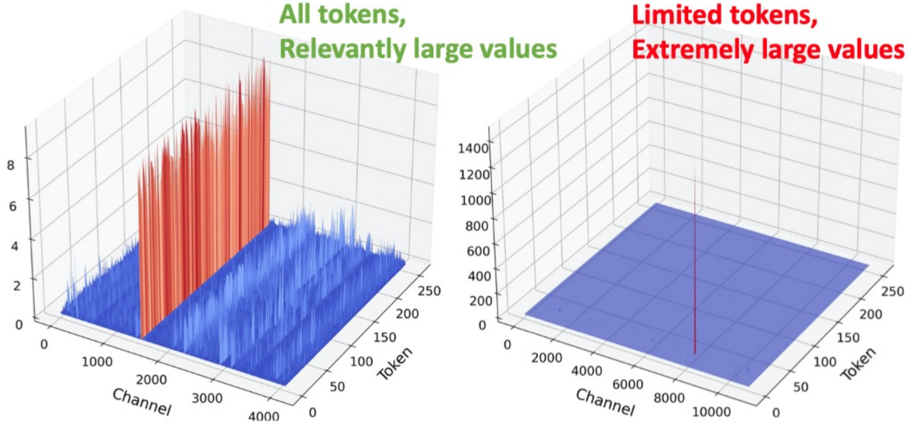


(a) Output activations of LLaMA-30B Layer 24 (b) Output activations of LLaMA-2-7B Layer 24

Massive

➢ Our Observations

- Massive Outliers Exist at the Second Linear Layer (Down Projection) of FFN Module
- We are the first to discover this phenomenon, while previous works only focus on layer outputs



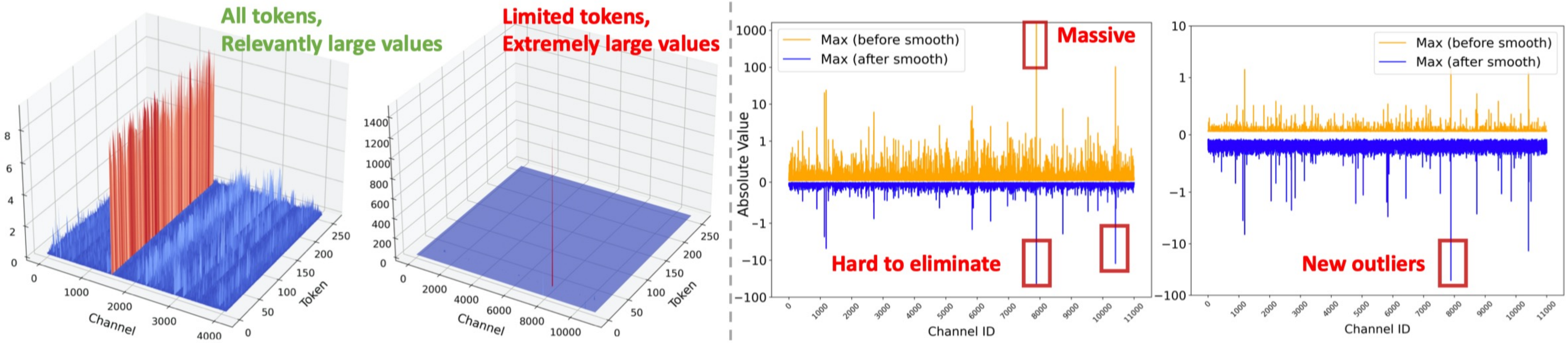
(a) LLaMA2_7B_Layer1_Attn_K_Proj (b) LLaMA2_7B_Layer1_FFN_Down_Proj

Normal Outliers **Massive Outliers**

Motivation

➢ Our Observations

- Massive Outliers Exist at the Second Linear Layer (Down Proj) of FFN Module
- Traditional Methods fail to eliminate these massive outliers
  - SmoothQuant[1]: cause the weights of the down-projection to display noticeable outliers
  - OmniQuant[2] and AffineQuant[3]: optimization-based methods to encounter problems with loss explosion



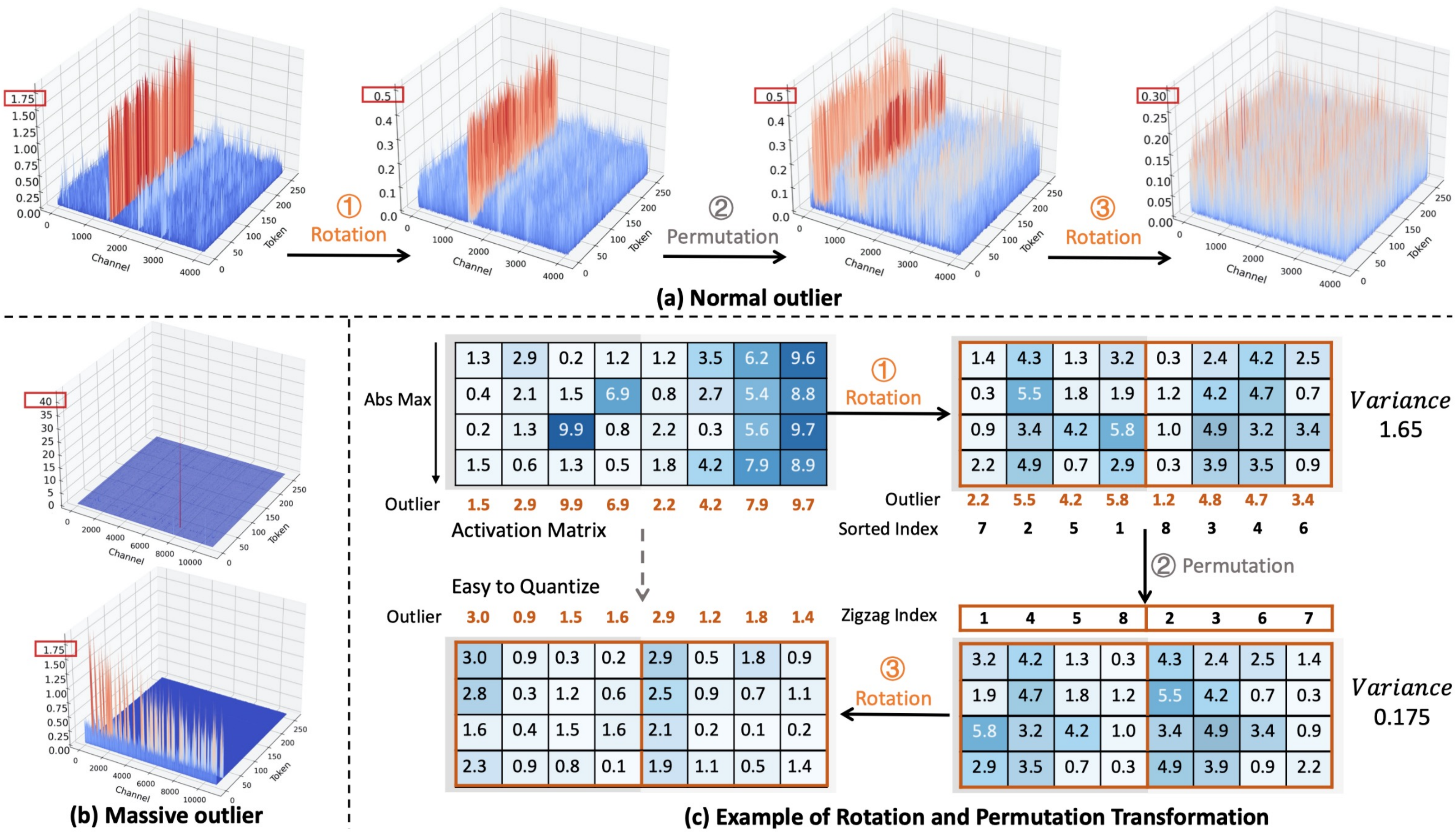
(a) LLaMA2_7B_Layer1_Attn_K_Proj (b) LLaMA2_7B_Layer1_FFN_Down_Proj (c) Activation Change with SmoothQuant (d) Weight Change with SmoothQuant

Normal Outliers **Massive Outliers** **SmoothQuant Fails to Eliminate Massive Outliers**

How to eliminate both Normal and Massive outliers?

[1]. Xiao, Guangxuan, et al. "Smoothquant: Accurate and efficient post-training quantization for large language models." *ICML*, 2023.
[2]. Shao, Wenqi, et al. "Omniquant: Omnidirectionally calibrated quantization for large language models." *ICLR,* 2024.
[3]. Ma, Yuexiao, et al. "Affinequant: Affine transformation quantization for large language models." *ICLR,* 2024.

DuQuant



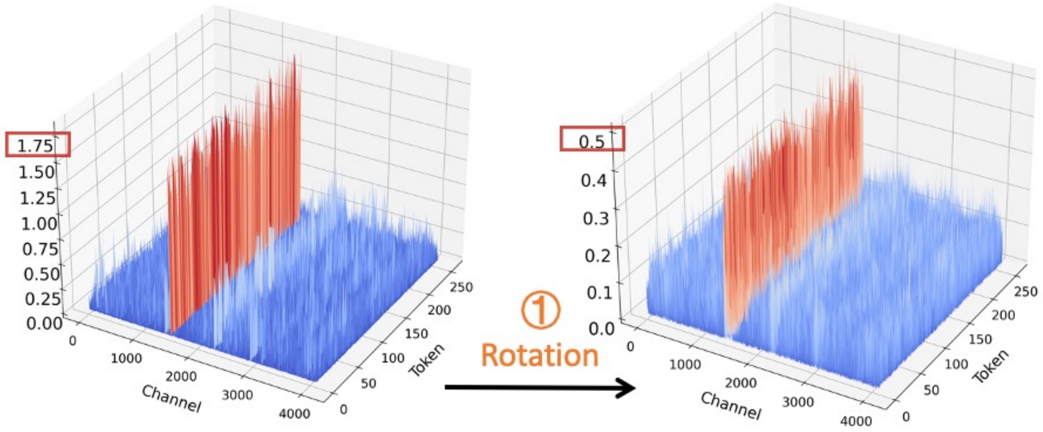
1.75
① Rotation
0.5
② Permutation
0.5
③ Rotation
0.30
Channel
Token
(a) Normal outlier
40
1.75
(b) Massive outlier
Abs Max
1.3 2.9 0.2 1.2 1.2 3.5 6.2 9.6
0.4 2.1 1.5 6.9 0.8 2.7 5.4 8.8
0.2 1.3 9.9 0.8 2.2 0.3 5.6 9.7
1.5 0.6 1.3 0.5 1.8 4.2 7.9 8.9
Outlier 1.5 2.9 9.9 6.9 2.2 4.2 7.9 9.7
Activation Matrix
① Rotation
1.4 4.3 1.3 3.2 0.3 2.4 4.2 2.5
0.3 5.5 1.8 1.9 1.2 4.2 4.7 0.7
0.9 3.4 4.2 5.8 1.0 4.9 3.2 3.4
2.2 4.9 0.7 2.9 0.3 3.9 3.5 0.9
Variance 1.65
Outlier 2.2 5.5 4.2 5.8 1.2 4.8 4.7 3.4
Sorted Index 7 2 5 1 8 3 4 6
② Permutation
Zigzag Index 1 4 5 8 2 3 6 7
3.2 4.2 1.3 0.3 4.3 2.4 2.5 1.4
1.9 4.7 1.8 1.2 5.5 4.2 0.7 0.3
5.8 3.2 4.2 1.0 3.4 4.9 3.4 0.9
2.9 3.5 0.7 0.3 4.9 3.9 0.9 2.2
Variance 0.175
③ Rotation
Easy to Quantize
Outlier 3.0 0.9 1.5 1.6 2.9 1.2 1.8 1.4
3.0 0.9 0.3 0.2 2.9 0.5 1.8 0.9
2.8 0.3 1.2 0.6 2.5 0.9 0.7 1.1
1.6 0.4 1.5 1.6 2.1 0.2 0.1 0.2
2.3 0.9 0.8 0.1 1.9 1.1 0.5 1.4
(c) Example of Rotation and Permutation Transformation

Rotation

➢ Method

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

- We first use Smooth technique [1] to balance the quantization difficulty $\mathbf{Y} = \mathbf{X} \cdot \mathbf{W} = (\mathbf{X} \cdot \Lambda^{-1})(\Lambda \cdot \mathbf{W})$
- Use rotation matrix to distribute the outliers to adjacent channels
- Ideal rotation matrix $\boldsymbol{R}$
  - Orthogonal $\mathbf{R}\mathbf{R}^{\top} = \mathbf{I}$ $|\mathbf{R}| = \pm 1$
  - Target the positions of outliers and mitigate them through matrix multiplication



➢ Rotation with prior knowledge

- Use greedy search with prior knowledge (the feature dimension of outlier) to compute a rotation matrix $\hat{\mathbf{R}}$

[1]. Xiao, Guangxuan, et al. "Smoothquant: Accurate and efficient post-training quantization for large language models." *ICML*, 2023.

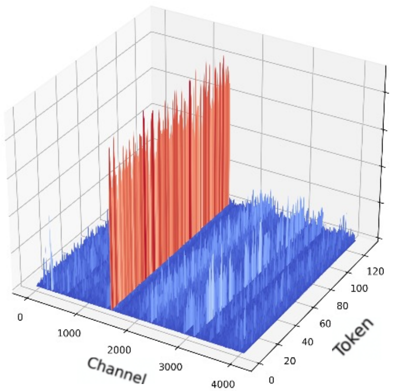
Rotation

➢ Rotation with prior knowledge

- Use greedy search with prior knowledge (the feature dimension of outlier) to compute a rotation matrix $\hat{\mathbf{R}}$
- The feature dimension $d^{(1)} = \arg\max_j (\max_i |\mathbf{X}_{ij}|)$
- Construct the rotation matrix by:

$$\mathbf{R}^1 = \mathbf{E}_{d^{(1)}} \tilde{\mathbf{R}} \mathbf{Q} \mathbf{E}_{d^{(1)}}, \qquad \mathbf{Q} = \begin{bmatrix} 1 & \mathbf{O} \\ \mathbf{O} & \mathbf{Q}' \end{bmatrix}$$

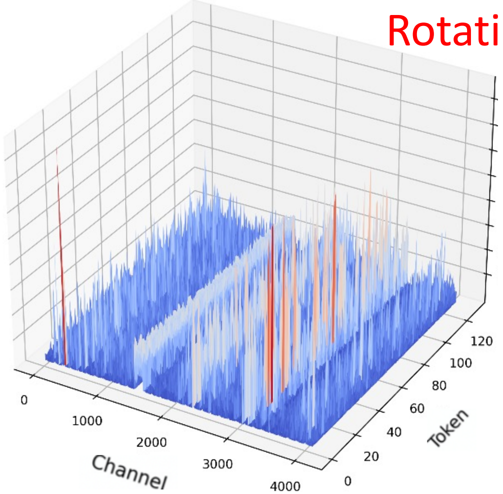
Original



- $\tilde{\mathbf{R}}$: an orthogonal initialized rotation matrix, first row is specifically uniformly distributed
- $\mathbf{E}_{d^{(1)}}$: switching matrix used to swap the first and the $d^{(1)}$ column of the activation
- $\tilde{\mathbf{R}}$ can mitigate outliers in the first column after the transformation by $\mathbf{E}_{d^{(1)}}$
- $\mathbf{Q}$: further increase the randomness of the rotation operation, $\mathbf{Q}'$ is a random orthogonal matrix
- Greedy search for N steps (once rotation may induce new outliers)

$$\hat{\mathbf{R}} = \mathbf{R}^1 \mathbf{R}^2 \cdots \mathbf{R}^n \qquad n = \arg\min_{k \in [1:N]} \left(\max_{i,j} |(\mathbf{X}\mathbf{R}^1 \cdots \mathbf{R}^k)_{ij}|\right)$$

Rotation once



➢ Block-wise rotation

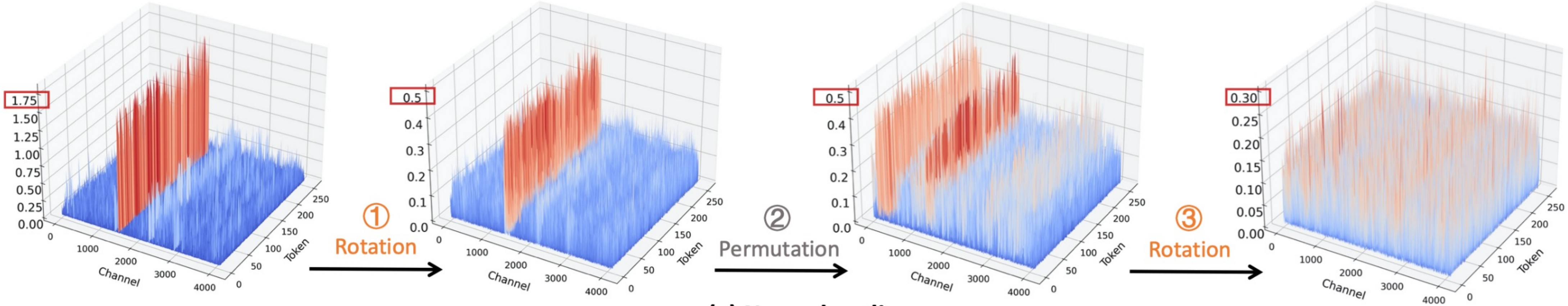
- For time and memory efficiency, we use block-wise rotation matrix

$$\hat{\mathbf{R}} = \mathrm{BlockDiag}(\hat{\mathbf{R}}_{b_1}, \ldots, \hat{\mathbf{R}}_{b_K}) \qquad \hat{\mathbf{R}} \in \mathbb{R}^{C_{in} \times C_{in}} \qquad \hat{\mathbf{R}}_{b_i} \in \mathbb{R}^{2^n \times 2^n}$$

Permutation

➢Limitation of Rotation

- Block-wise rotation: uneven outlier magnitudes across different blocks
- Measurement: Compute the variance of different blocks $\mathrm{Var}([M_{b_1}, M_{b_2}, ..., M_{b_K}])$
  - For i block, the M_{b_i} represents the mean values of all O_j, O_j is the largest outlier in dimension d_j



(a) Normal outlier

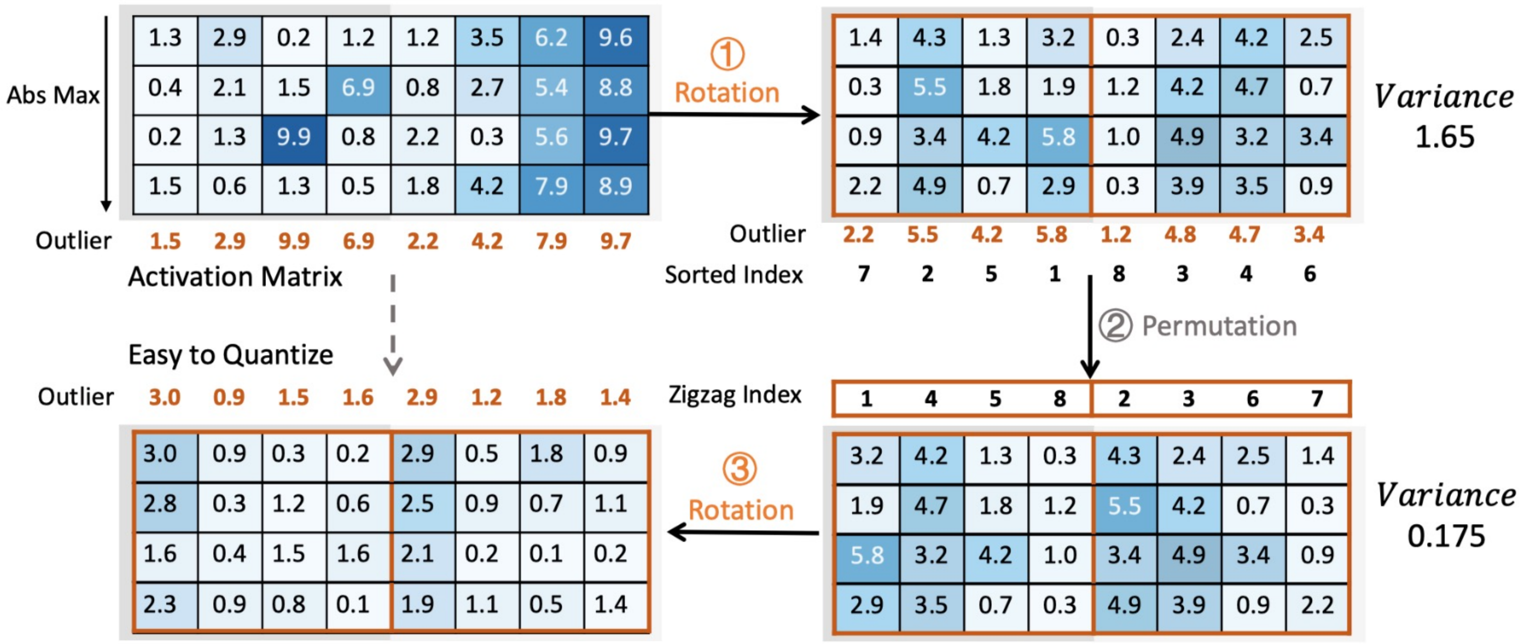
➢Solution

- Channel permutation to balance the distribution of outliers across blocks
- Permutation transformation is also orthogonal, denote as $\mathbf{P}$
- After permutation, employ another rotation transformation to further smooth the activations

Zigzag Permutation

➢Zigzag Order

- Distribute the channels with the highest activations across the blocks in a back-and-forth pattern
- Fast with strong performance



	LLaMA2-7B				LLaMA2-13B			
Permutation Method	WikiText2 ↓	C4 ↓	Variance	Time/s	WikiText2 ↓	C4 ↓	Variance	Time/s
w.o. Permutation	7.92	10.64	3.9e-2	27.5	5.96	7.94	3.1e-2	44.7
Random	6.40	8.08	4.9e-3	89.5	5.43	7.07	3.9e-3	148.6
Simulated Annealing	6.26	7.89	1.7e-4	769.6	5.42	7.06	1.5e-4	1257.8
Zigzag	6.28	7.90	3.0e-4	48.6	5.42	7.05	2.5e-4	74.0

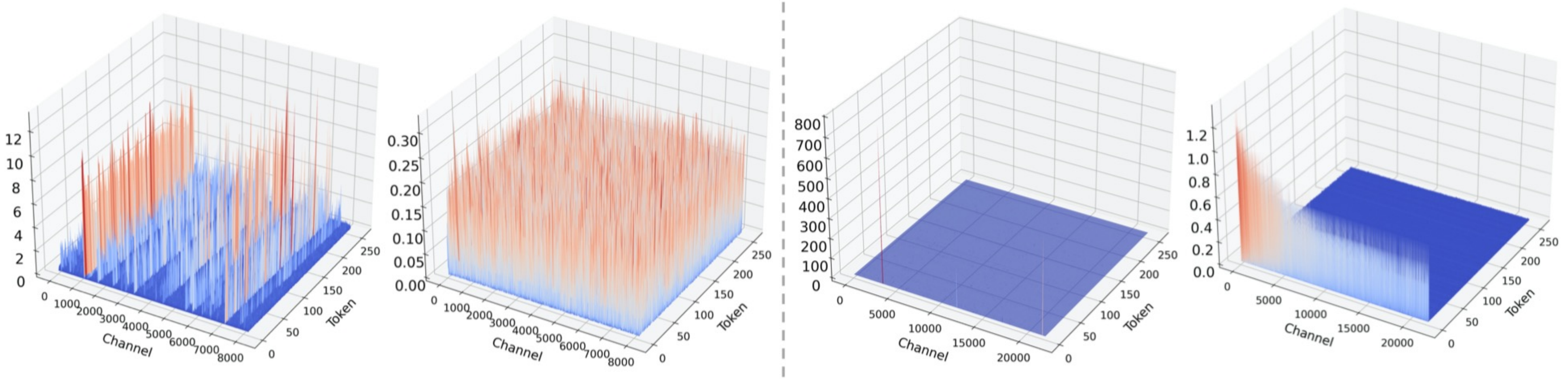
DuQuant

➢Linear Layer

- Smooth techniques (SmoothQuant)
- Block-wise Rotation (block size: 128)
- Permutation along with second Rotation

$$\mathbf{Y} = \mathbf{X} \cdot \mathbf{W} = [(\mathbf{X} \cdot \underbrace{\mathbf{\Lambda}^{-1})\hat{\mathbf{R}}_{(1)} \cdot \mathbf{P} \cdot \hat{\mathbf{R}}_{(2)}}_{\mathbf{G}}] \cdot [\underbrace{\hat{\mathbf{R}}_{(2)}^\top \cdot \mathbf{P}^\top \cdot \hat{\mathbf{R}}_{(1)}^\top (\mathbf{\Lambda}}_{\mathbf{G}^{-1}} \cdot \mathbf{W})]$$

➢Visualization



(a) LLaMA1_65B_Layer11_Attn_K_Proj (b) After Rotation and Permutation

Normal Outliers

(c) LLaMA1_65B_Layer2_FFN_Down_Proj (d) After Rotation and Permutation

Massive Outliers

DuQuant

➢Theoretical Analysis

- Within Each block, the constructed rotation matrix effectively mitigates the maximum outlier

$$Eqn.\ (3)\quad \hat{\mathbf{R}} = \mathrm{BlockDiag}(\hat{\mathbf{R}}_{b_1}, ..., \hat{\mathbf{R}}_{b_K})\quad \hat{\mathbf{R}} \in \mathbb{R}^{C_{in}\times C_{in}}\quad \hat{\mathbf{R}}_{b_i} \in \mathbb{R}^{2^n\times 2^n}$$

Theorem 1 (Rotation). *For the activation input $\mathbf{X} \in \mathbb{R}^{T\times C_{in}}$, $\hat{\mathbf{R}} \in \mathbb{R}^{2^n\times 2^n}$ is a diagonal block matrix constructed as per Eqn. (3). For a specific block b_i, let $O_j(\cdot)$ represent the maximum outlier of the j-th dimension d_j within the input. Then, we can deduce that,*

$$\max_{1\le j\le 2^n} O_j(\mathbf{X}_{b_i}\hat{\mathbf{R}}_{b_i}) \le \max_{1\le j\le 2^n} O_j(\mathbf{X}_{b_i}). \tag{6}$$

- Zigzag permutation ensures a balanced upper bound shared among different blocks

Theorem 2 (Zigzag Permutation). *For the activation input $\mathbf{X} \in \mathbb{R}^{T\times C_{in}}$, it can be divided into K blocks, where $K = C_{in}/2^n$. Let O_j denote the max outlier of the dimension d_j in $\mathbf{X}$, the reordered outliers from large to small is expressed as $O^{(1)}, O^{(2)}, ..., O^{(C_{in})}$. Moreover, the M_{b_i} represents the mean value of all O_j in the i-th block, $i = 1, 2, ..., K$. Let $\delta := \max\{|O^{(i+1)} - O^{(i)}|\}, i = 1, 2, ..., C_{in}-1$. Then, following the zigzag permutation, the mean value M_{b_i} within each i-th block consistently satisfies,*

$$M_{b_i} \le O^{(1)} + \frac{(2^n K - 1)(2^{n-1} - 1)}{2^n}\delta, \qquad i = 1, 2, 3, ..., K. \tag{7}$$

Experiments

- **DuQuant: Rotation – Permutation – Rotation**
  - **LWC[1]**: adjusts weights by training parameters $\gamma, \beta \in [0,1]$ to compute the step size $\Delta = \frac{\gamma \max(\mathbf{X}) - \beta \min(\mathbf{X})}{2^b - 1}$
- **Models: LLaMA1, LLaMA2, LLaMA3, Vicuna, Mistral**
- **Tasks: Language generation (PPL), Commonsense QA, MMLU, MT-Bench, LongBench**

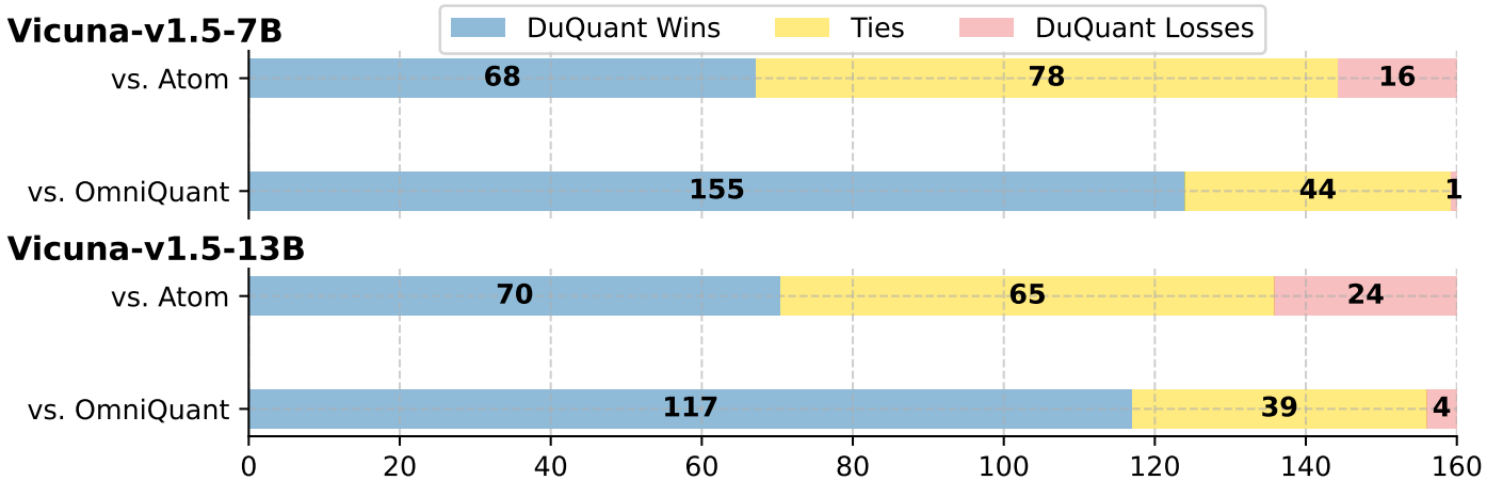
Dataset	#Bit	Method	1-7B	1-13B	1-30B	1-65B	2-7B	2-13B	2-70B
WikiText2	FP16	-	5.68	5.09	4.10	3.53	5.47	4.88	3.31
	W4A4	SmoothQuant	25.25	40.05	192.40	275.53	83.12	35.88	26.01
		OmniQuant	11.26	10.87	10.33	9.17	14.26	12.30	NaN
		AffineQuant	10.28	10.32	9.35	-	12.69	11.45	-
		QLLM	9.65	8.41	8.37	6.87	11.75	9.09	7.00
		Atom	8.15	7.43	6.52	5.14	8.40	6.96	NaN
		DuQuant	6.40	5.65	4.72	4.13	6.28	5.42	3.79
		DuQuant+LWC	**6.18**	**5.47**	**4.55**	**3.93**	**6.08**	**5.33**	**3.76**
C4	FP16		7.08	6.61	5.98	5.62	6.97	6.46	5.52
	W4A4	SmoothQuant	32.32	47.18	122.38	244.35	77.27	43.19	34.61
		OmniQuant	14.51	13.78	12.49	11.28	18.02	14.55	NaN
		AffineQuant	13.64	13.44	11.58	-	15.76	13.97	-
		QLLM	12.29	10.58	11.51	8.98	13.26	11.13	8.89
		Atom	10.34	9.57	8.56	8.17	10.96	9.12	NaN
		DuQuant	7.84	7.16	6.45	6.03	7.90	7.05	5.87
		DuQuant+LWC	**7.73**	**7.07**	**6.37**	**5.93**	**7.79**	**7.02**	**5.85**

[1]. Shao, Wenqi, et al. "Omniquant: Omnidirectionally calibrated quantization for large language models." *ICLR,* 2024.

Experiments

- Models: LLaMA1, LLaMA2, LLaMA3, Vicuna, Mistral
- Tasks: Language generation (PPL), Commonsense QA, MMLU, MT-Bench, LongBench

Vicuna	Setting	Qasper	QMSum	MultiNews	TREC	TriviaQA	SAMSum	DuReader	RepoBench-P	Avg
Vicuna-v1.5-7B W4A4	FP16	23.27	21.07	26.91	66.00	82.59	41.06	25.53	48.23	41.83
	SmoothQuant	4.11	2.00	6.05	15.00	1.62	1.55	4.24	25.92	7.56
	OmniQuant	1.62	3.93	2.64	1.00	0.81	0.61	1.87	14.97	3.43
	Atom	17.97	20.24	24.60	58.00	67.20	37.94	19.41	29.34	34.34
	DuQuant	**19.98**	**21.15**	**25.85**	**64.00**	**78.91**	**42.24**	**23.15**	**47.66**	**40.37**
Vicuna-v1.5-13B W4A4	FP16	24.41	21.24	26.53	68.00	86.81	41.97	27.57	43.08	42.45
	SmoothQuant	2.18	2.95	3.54	1.50	1.83	0.35	6.71	11.57	3.83
	OmniQuant	0.68	1.78	2.83	9.00	1.13	0.45	13.83	8.46	4.77
	Atom	17.67	20.23	23.39	59.00	80.75	38.72	21.79	37.31	37.36
	DuQuant	**18.93**	**20.72**	**26.59**	**66.50**	**83.04**	**42.67**	**26.02**	**38.09**	**40.32**



DuQuant v.s. FP16	Former Win	Tie	Former Loss
Vicuna-v1.5-7B	36	56	68
Vicuna-v1.5-13B	43	53	64

Experiments

➢ **Settings: LLaMA2-7B, Measure on RTX 3090, Input seq --- 2048, Decoding --- 128 steps**
- Pre-filling stage --- computational bound, measure the speedup
- Decoding stage --- memory bound, measure the memory usage

Peak memory usage with a batch size of 1.

LLaMA2-7B	Pre-filling (GB)	Saving	Decoding (GB)	Saving
FP16	15.282	-	13.638	-
SmoothQuant	4.782	3.196×	3.890	3.506×
QLLM	5.349	2.857×	3.894	3.502×
QuaRot	4.784	3.194×	3.891	3.505×
DuQuant	4.786	3.193×	3.893	3.503×

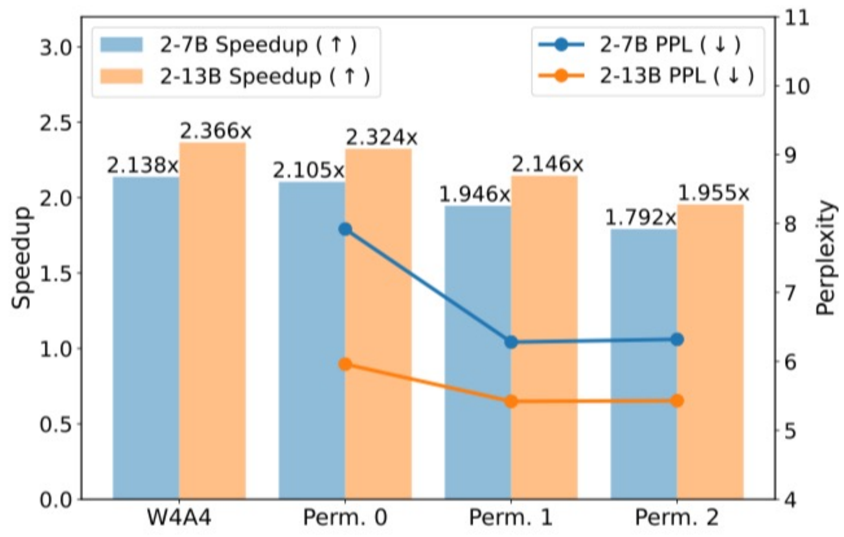
End-to-end pre-filling speedup on LLaMA2-7B model.

Batch Size	FP16 Time	DuQuant Time	Speedup
1	568ms	294ms	1.93×
2	1003ms	509ms	1.97×
3	1449ms	720ms	2.01×

Decoding phase results of one LLaMA2-7B layer with a batch size of 64.

Method	Time (ms)	Saving Factor	Memory (GB)	Saving Factor
FP16	659	-	3.550	-
SmoothQuant	437	1.508x	1.669	2.127x
QLLM	OOM	-	OOM	-
QuaRot	457	1.442x	1.678	2.116x
DuQuant	499	1.321x	1.677	2.117x

Computational overhead analysis.



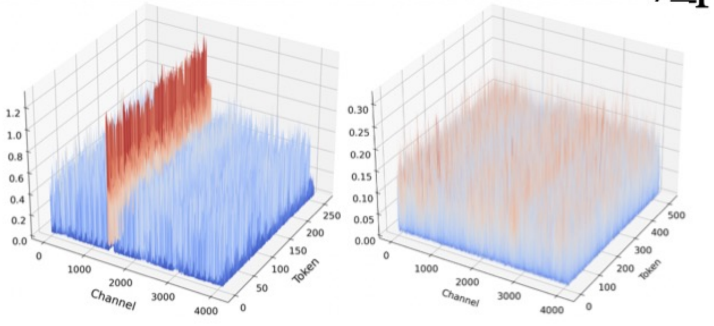
About 10% compared to W4A4

Experiments

➢ Comparison with QuaRot [1]

- Better rotation --- utilize prior knowledge
- Permutation transformation --- further smooth activation landscape, better performance
- Jointly smooth weight and activations --- no need for GPTQ, faster

Figure 4: LLaMA2-7B Attention key_proj.



(a) Hadamard (b) Ours

Table F22: Evaluation results between QuaRot and DuQuant under QuaRot settings.

Model	Method	WikiText2 ↓	C4 ↓	PIQA	WinoGrande	HellaSwag	ARC-E	ARC-C	LAMBADA	Avg ↑
LLaMA2-7B W4A4 QuaRot Setting	FP16	5.47	6.97	79.11	69.06	75.99	74.58	46.25	73.90	69.82
	QuaRot-RTN	8.37	-	72.09	60.69	65.4	58.88	35.24	57.27	58.26
	QuaRot-GPTQ	6.1	-	76.77	63.77	72.16	69.87	40.87	**70.39**	65.64
	DuQuant	6.23	7.91	76.28	66.93	72.96	69.99	40.53	69.61	66.05
	DuQuant+LWC	**6.01**	**7.67**	**77.64**	**67.8**	**72.97**	**70.37**	**41.81**	69.53	**66.69**
LLaMA2-13B W4A4 QuaRot Setting	FP16	4.88	6.46	80.47	72.22	79.39	77.48	49.23	76.75	72.59
	QuaRot-RTN	6.09	-	77.37	67.32	73.11	70.83	43.69	70.66	67.16
	QuaRot-GPTQ	5.4	-	78.89	70.24	76.37	72.98	46.59	73.67	69.79
	DuQuant	5.39	7.05	78.51	**70.88**	76.80	**74.62**	**48.21**	73.92	**70.49**
	DuQuant+LWC	**5.27**	**6.93**	**78.73**	**70.88**	**77.20**	74.07	47.27	**73.96**	70.35

Table F24: Quantization runtime comparison on a single NVIDIA A100 80G GPU.

Model	LLaMA2-7B	LLaMA2-13B	LLaMA2-70B
QuaRot	20min	36min	5.1h
DuQuant	50s	71s	270s

Table 8: PPL (↓) comparison under W4A4 setting.

Method	**1-7B**	**1-13B**	**1-30B**	**2-7B**	**2-13B**
FP16	5.68	5.09	4.10	5.47	4.88
QuaRot-RTN	7.08	6.57	5.44	9.66	6.73
QuaRot-GPTQ	6.44	5.63	4.73	6.39	5.75
DuQuant	6.40	5.65	4.72	6.28	5.42
DuQuant+LWC	**6.18**	**5.47**	**4.55**	**6.08**	**5.33**

[1]. Ashkboos, Saleh, et al. "Quarot: Outlier-free 4-bit inference in rotated llms." *NeurIPS* 2024.

NEURAL INFORMATION PROCESSING SYSTEMS

DuQuant: Distributing Outliers via Dual Transformation Makes Stronger Quantized LLMs

NeurIPS 2024 Oral

Haokun Lin[*1,3,4], Haobo Xu[*2], Yichen Wu[*4], Jingzhi Cui[2], Yingtao Zhang[2], Linzhan Mou[5], Linqi Song[4], Zhenan Sun[^1,3], Ying Wei[^5]

[*]Equal Contribution [^]Corresponding Authors

[1]School of Artificial Intelligence, University of Chinese Academy of Sciences

[2]Tsinghua University [3]NLPR & MAIS, Institute of Automation, Chinese Academy of Sciences

[4]City University of Hong Kong [5]Zhejiang University

Thanks

Code

Paper

Haokun Lin and *Yichen Wu* are actively seeking postdoctoral opportunities,

while *Haobo Xu* is exploring potential PhD positions.

Please feel free to reach out to us!