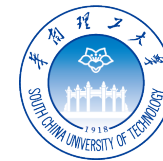




Rotated Runtime Smooth: Training-Free Activation Smoother for Accurate INT4 Inference

Ke Yi, Zengke Liu, Jianwei Zhang

Chengyuan Li, Tong Zhang, Junyan Lin, Jingren Zhou



华南理工大学
South China University of Technology

INT4 Inference for Large Language Models

Growth drivers

Application of LLM

The **Scale** of model size increase rapidly, from 7B to 671B
Models are employed across **servers** and **mobile devices**

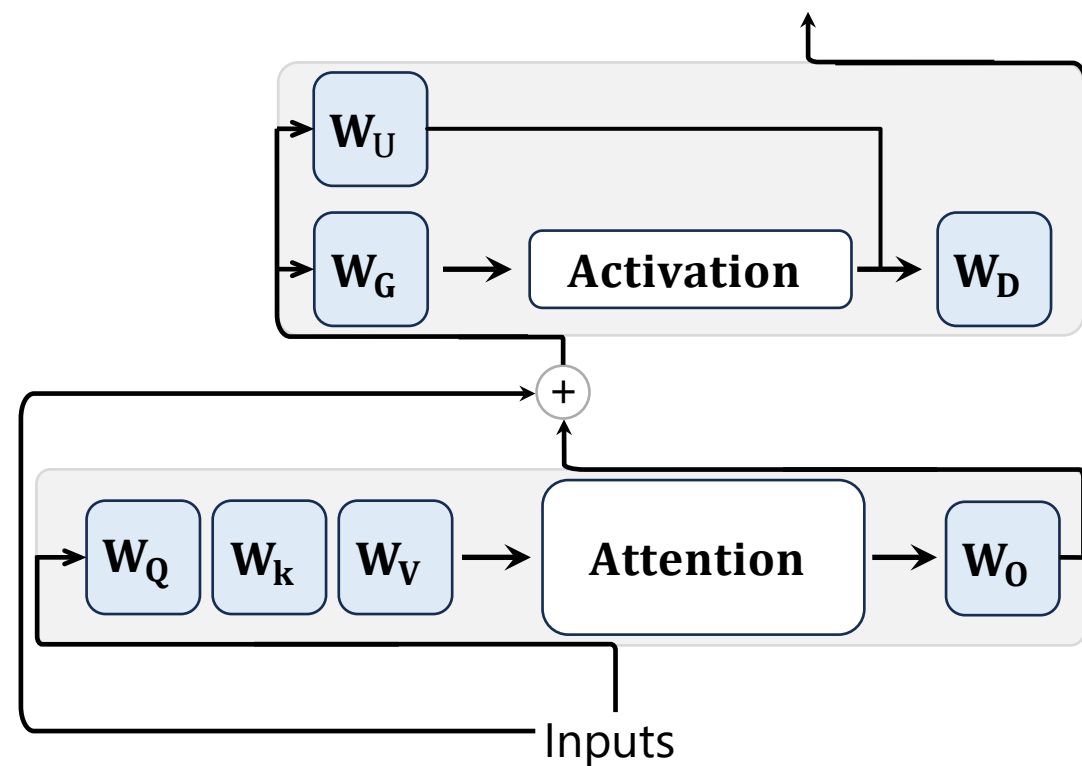
Efficiency Challenge

Matrix Multiplication is at the core of LLM

Current limitations

Quantization Loss Quant FP16 Weight/Activation to INT4 leads to severe loss

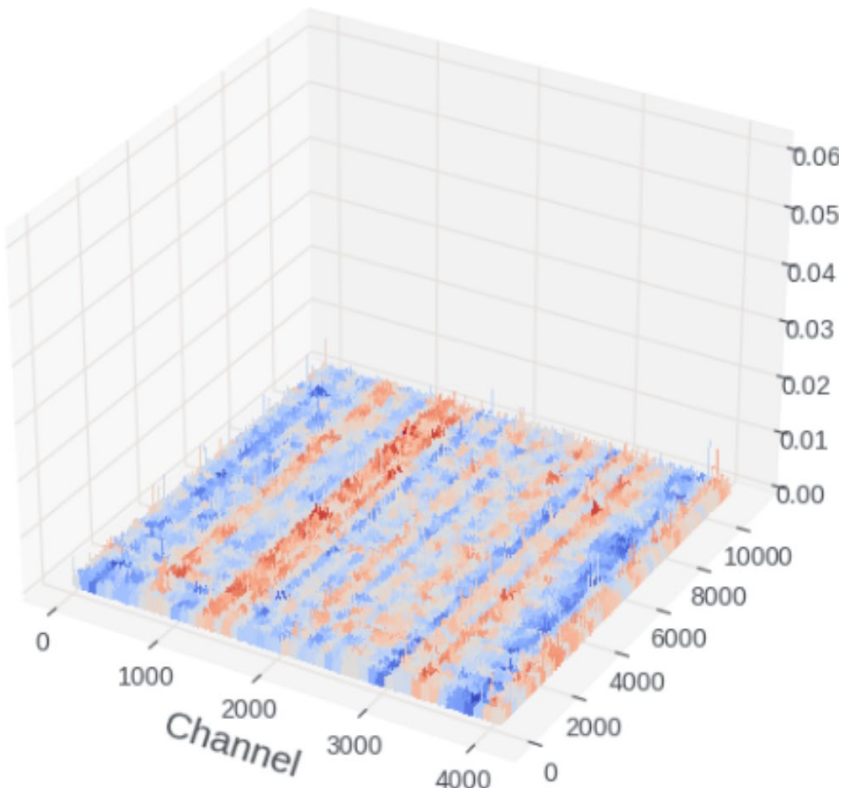
Overfitting from Calibration Calibration based on validation data leads to overfitting



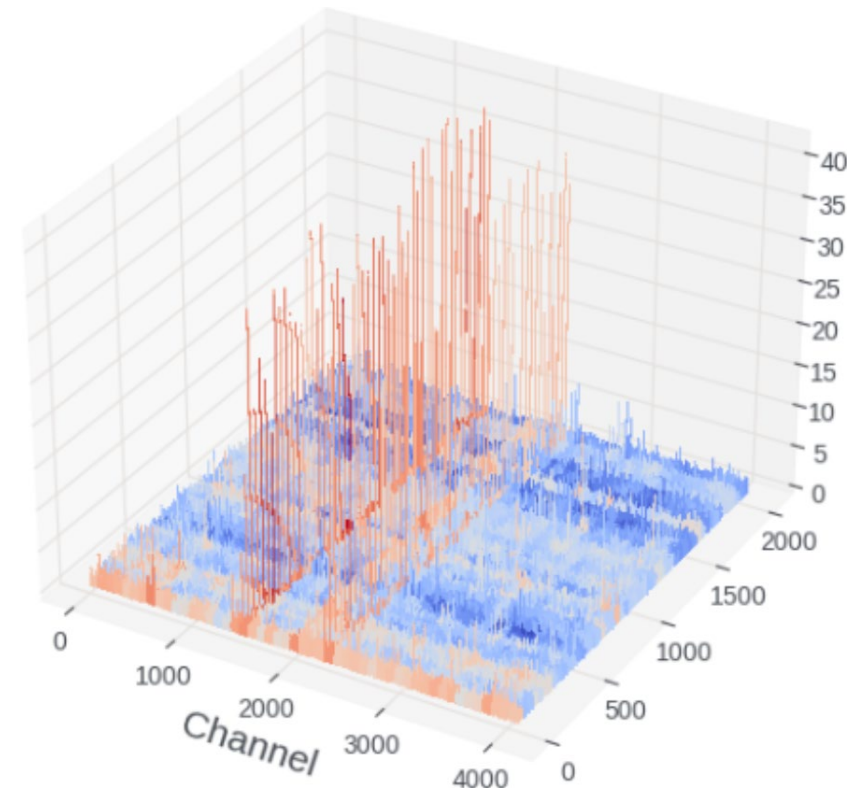
Background

Quantization Challenges from Outliers

- INT4 inference quantizing both inputs and weights of the linear layer.
 - Both have outliers, while inputs have more outstanding outliers



Layer-0-Down_proj of LLaMA-2 7B

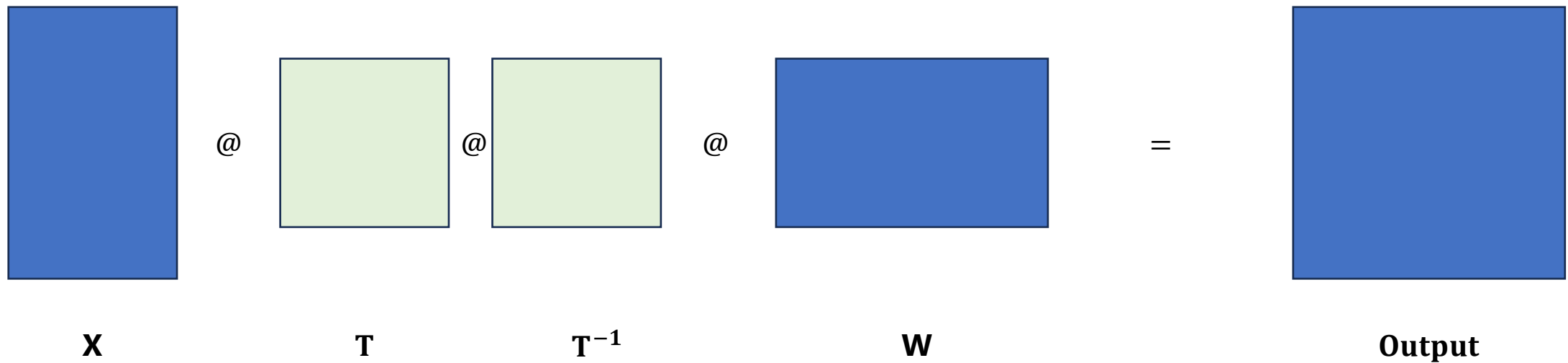


Inputs for this Linear

Background

Solution by Computational Invariance trick

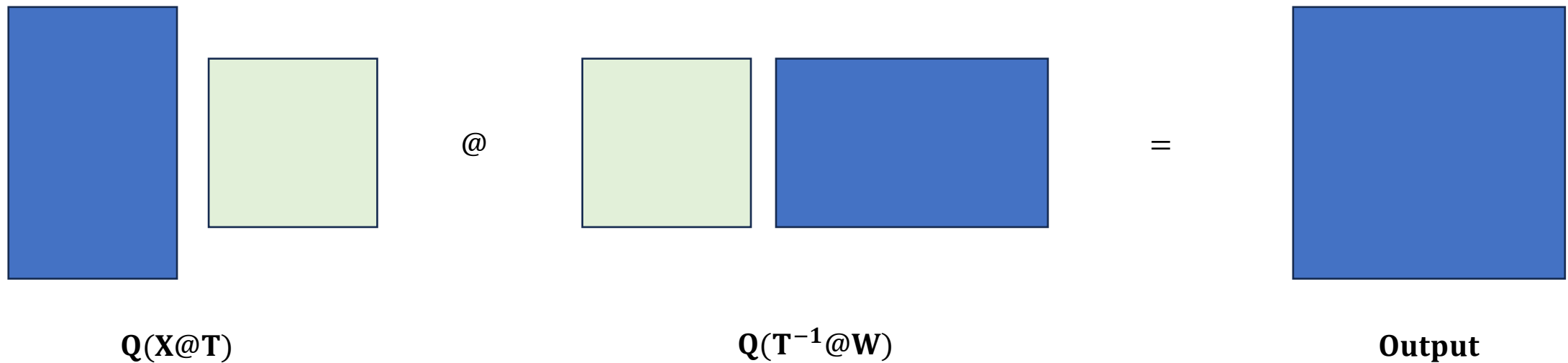
- It will benefit if we can find a transformation to make the quantization easier
 - While keeping the computation invariance



Background

Solution by Computational Invariance trick

- It will benefit if we can find a transformation to make the quantization easier
 - While keeping the computation invariance
 - And Fusing the transformation priorly to avoid



Background

Form1: 1D Transformation

- Based on the assumption that outliers are more outstanding at input matrix
- It would benefit from 'sharing' the level of outliers from input to weight
- The 'sharing' is based on 1D Transformation,
 - where $T = \text{Diag}(S), S \in \mathbb{R}^{1 \times K}, X \in \mathbb{R}^{N \times K}, W \in \mathbb{R}^{K \times M}$
 - To uniformly share the outlier, $S = \sqrt{\frac{\text{ChannelWise_MAX}(T)}{\text{ChannelWise_MAX}(X)}}$

2	8	1
2	8	-1

X

@

2	2
2	2
1	-1

W

Background

Form1: 1D Transformation

- Based on the assumption that outliers are more outstanding at input matrix
- It would benefit from 'sharing' the level of outliers from input to weight
- The 'sharing' is based on 1D Transformation,
 - where $T = \text{Diag}(S)$, $S \in \mathbb{R}^{1 \times K}$, $X \in \mathbb{R}^{N \times K}$, $W \in \mathbb{R}^{K \times M}$
 - To uniformly share the outlier, $S = \sqrt{\frac{\text{ChannelWise_MAX}(T)}{\text{ChannelWise_MAX}(X)}}$

2	8	1
2	8	-1

1		
	$\frac{1}{2}$	
		1

@

1		
	2	
		1

2	2
2	2
1	-1

$(X@T)$

$(T^{-1}@W)$

Background

Form1: 1D Transformation

- Based on the assumption that outliers are more outstanding at input matrix
- It would benefit from 'sharing' the level of outliers from input to weight
- The 'sharing' is based on 1D Transformation,
 - where $T = \text{Diag}(S), S \in \mathbb{R}^{1 \times K}, X \in \mathbb{R}^{N \times K}, W \in \mathbb{R}^{K \times M}$
 - To uniformly share the outlier, $S = \sqrt{\frac{\text{ChannelWise_MAX}(T)}{\text{ChannelWise_MAX}(X)}}$

2	4	1
2	4	-1

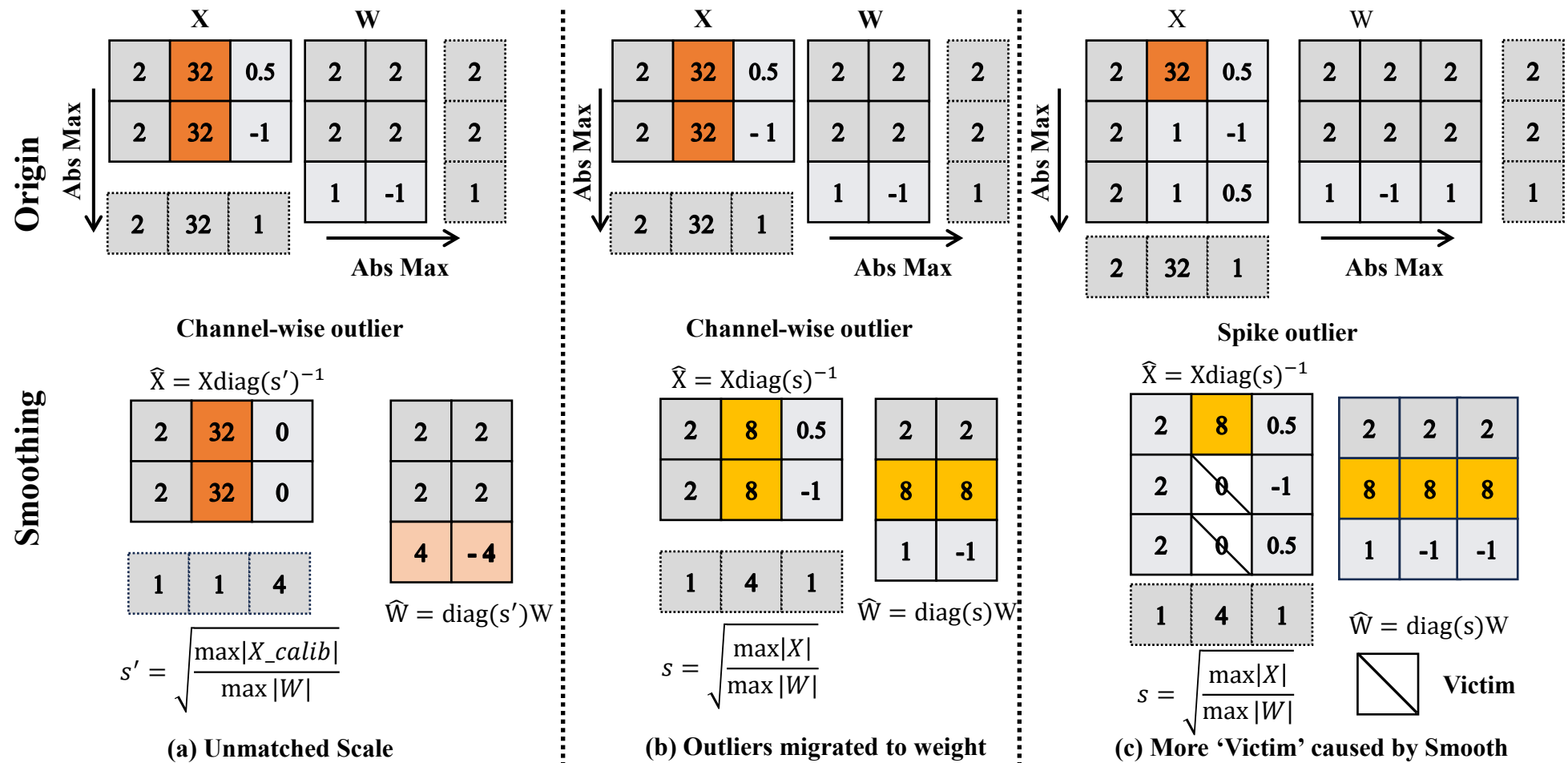
\mathbf{X}'

@

2	2
4	4
1	-1

\mathbf{W}'

Problem of 1D Transformation



Form2: 2D Transformation

- Is it possible to not migrate outliers but eliminate outliers?
- Here we introduce the Rotation transformation

2	8	1	1
2	8	-1	-1

X

@

2	2
1	1
4	-4
1	-1

W

Form2: 2D Transformation

- Is it possible to not migrate outliers but eliminate outliers?
- Here we introduce the Rotation transformation

2	8	1	1
2	8	-1	-1

$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$

@

$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$

2	2
1	1
4	-4
1	-1

$(X@R)$

$(R^{-1}@W)$

Form2: 2D Transformation

- Is it possible to not migrate outliers but eliminate outliers?
- Here we introduce the Rotation transformation

6	-3	4	-3
5	-2	5	-4

X'

@

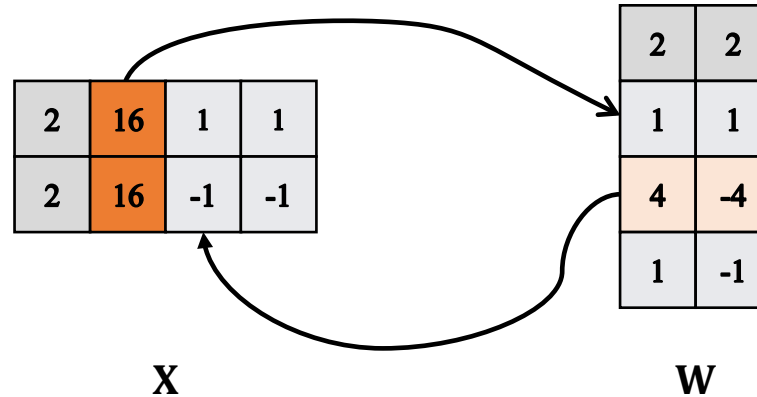
4	-1
2	-1
-1	4
-1	2

W'

Background

Why 2D transformation is better than 1D transformation

- The key lies into the 'flow' of outliers



Background

Why 2D transformation is better than 1D transformation

- The key lies into the 'flow' of outliers

2	8	2	1
2	8	-2	-1

X'

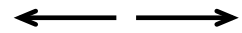
2	2
8	8
2	-2
1	-1

W'

Background

Why 2D transformation is better than 1D transformation


- The key lies into the 'flow' of outliers



2	16	1	1
2	16	-1	-1

X

2	2
1	1
4	-4
1	-1



W

Background

Why 2D transformation is better than 1D transformation

- The key lies into the 'flow' of outliers

10	-7	8	-7
8	7	10	-7

\mathbf{X}'

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

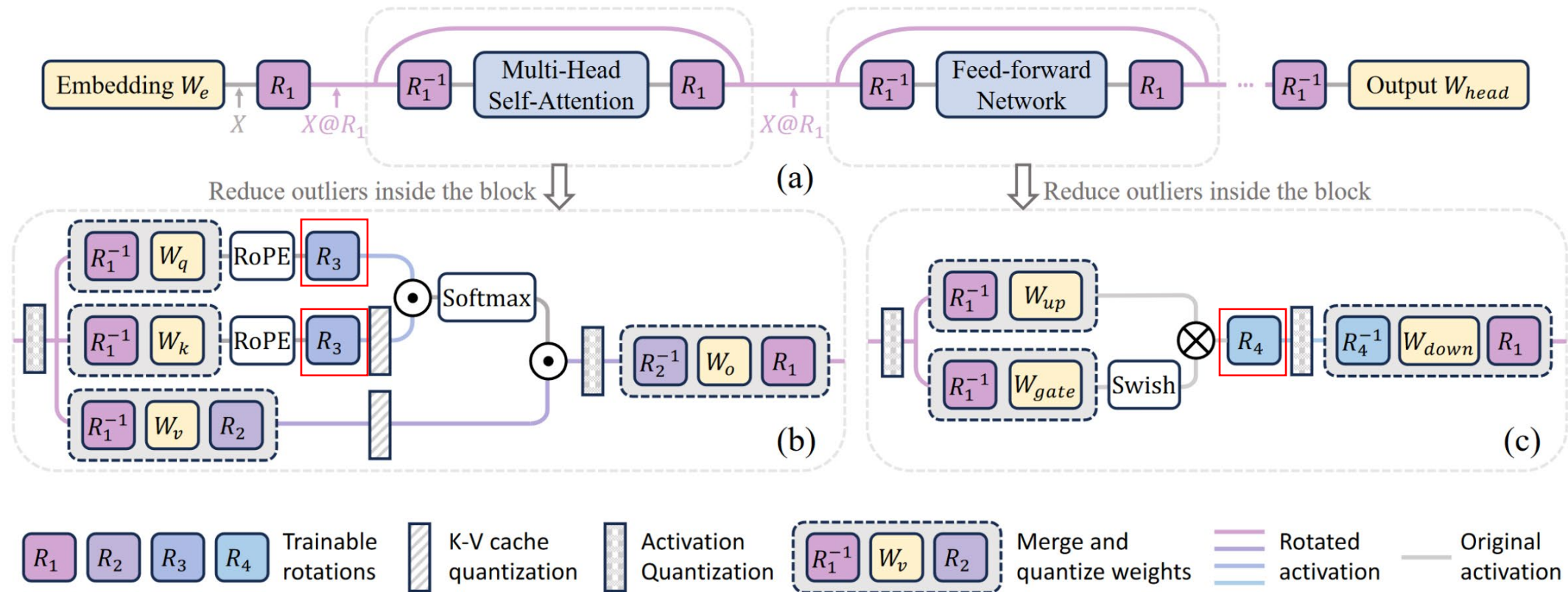
\mathbf{W}'

Method	PPL
FP16	5.0
INT4	7e3
INT4 w/ 1D	34.5
INT4 w/ 2D	5.39

Background

No Free lunch for better performance

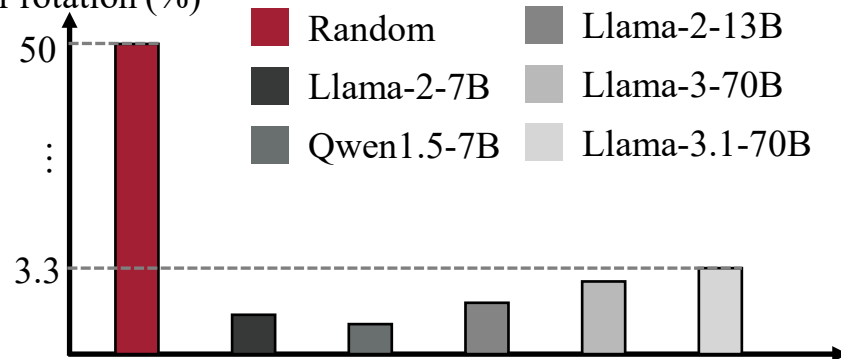
- Some 'rotation' must apply online



Review the nature of 'Rotation'

- The rotation operation is revertible
 - i.e., it can transform rough matrix to smooth one, also it would transform a smooth one to rough one.
- The current success depends on the distribution of current LLMs. (Figure a)
- Moreover, the rotated matrix can be sub-smooth. (Figure b)

Probability to generate more outliers
after rotation (%)



(a)

Rotate →

(b)

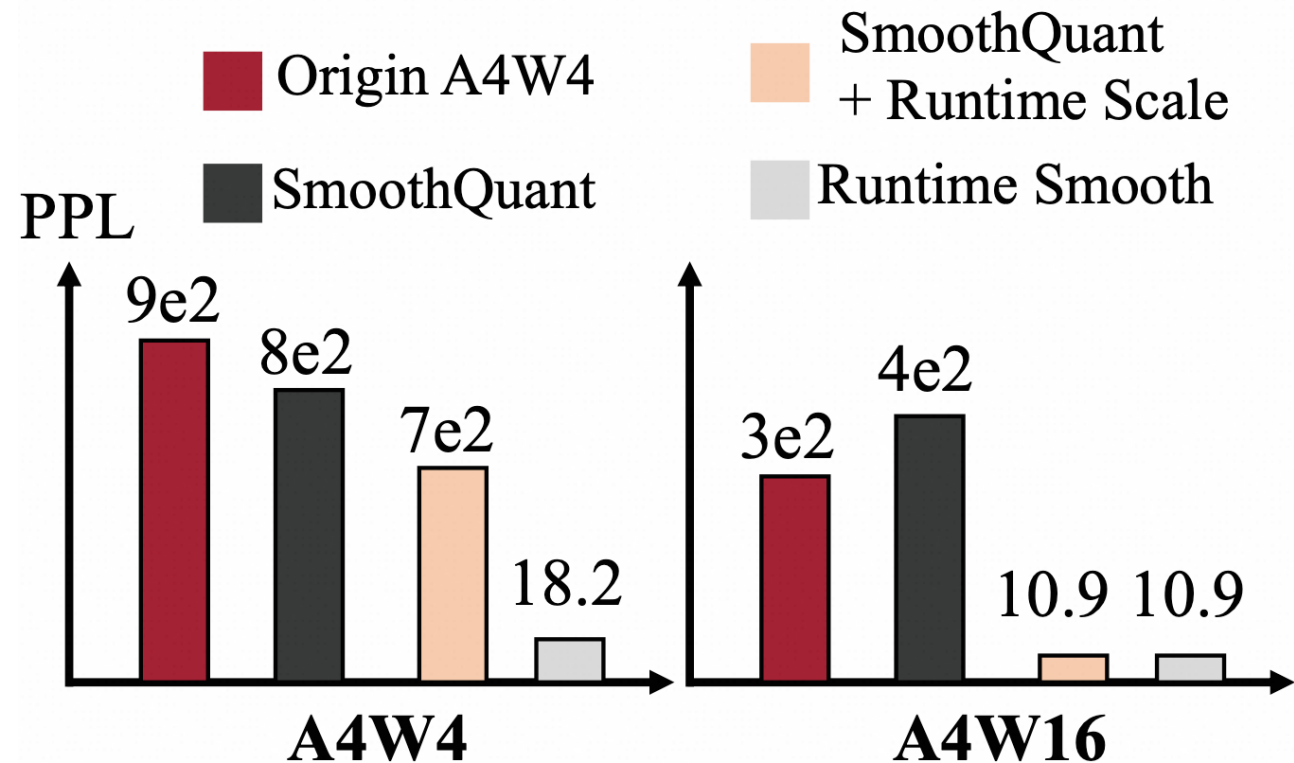
Contribution 1

- Propose Runtime Smooth, which obtain transformation factor online
 - Avoid the unmatched problem
 - Instead of migrating the outliers, it eliminate outliers

$$s_j = \max(|\mathbf{X}_j|), j = 1, 2, \dots, K \quad (1)$$

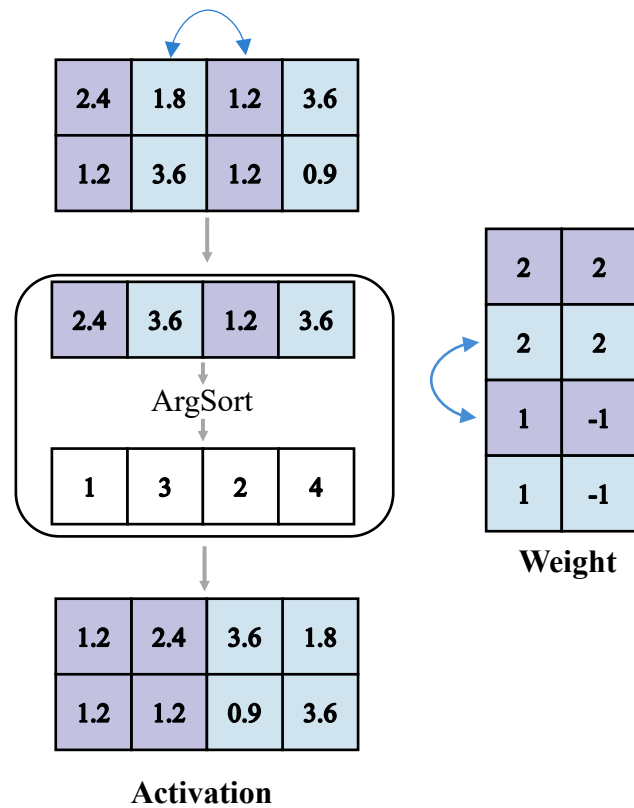
$$\hat{\mathbf{X}} = \text{Quantize}(\mathbf{X}/s), \hat{\mathbf{W}} = \text{Quantize}(\mathbf{W}), \quad (2)$$

$$\mathbf{Y} = \sum_{j=1}^K \hat{\mathbf{X}}_j \cdot \hat{\mathbf{W}}_j^T \cdot s_j, \quad (3)$$

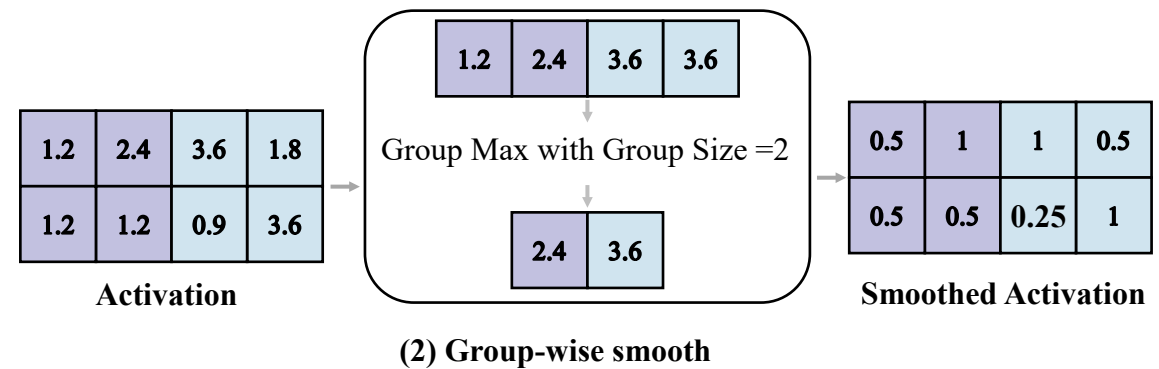


Contribution 2

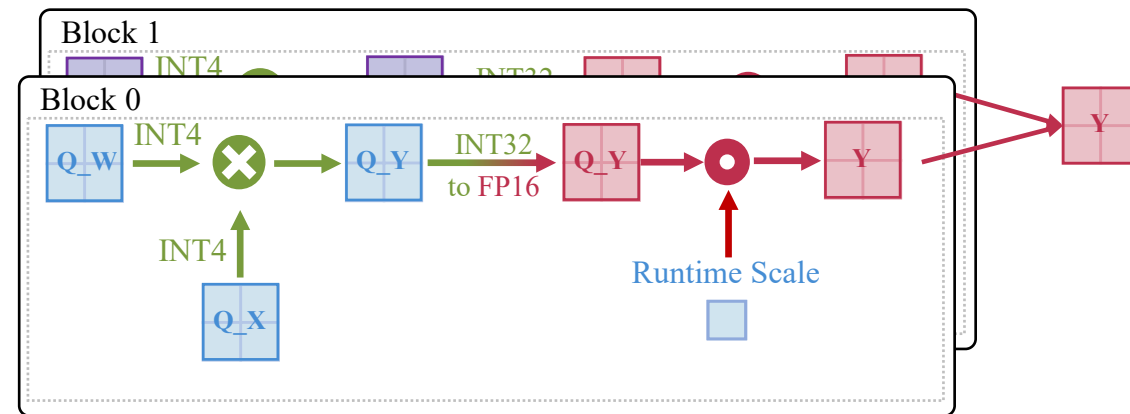
- Fuse the Runtime Smooth factor into GEMM kernel
 - Maintain the computation invariance
 - Bring negligible overhead



(1) Reorder activations and weights

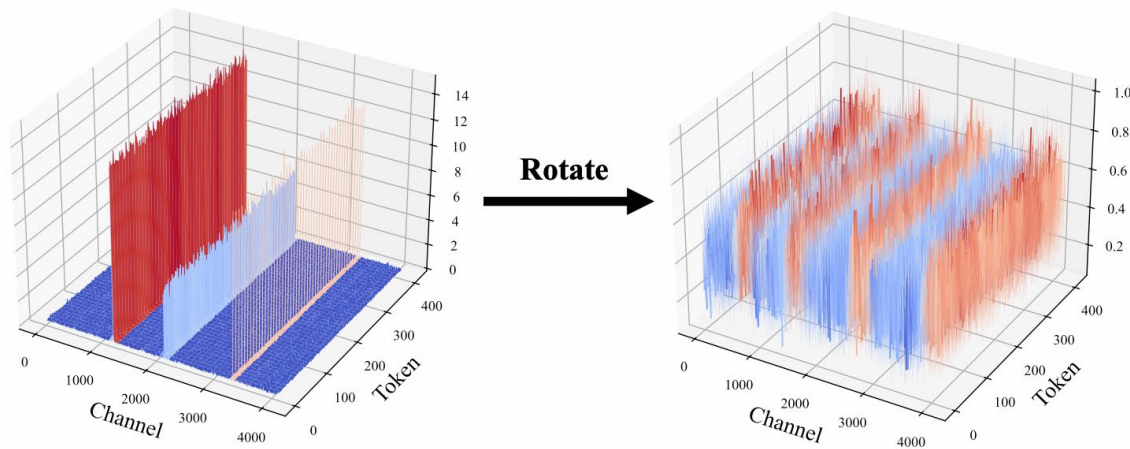


(3) Kernel fusion for Runtime Smooth

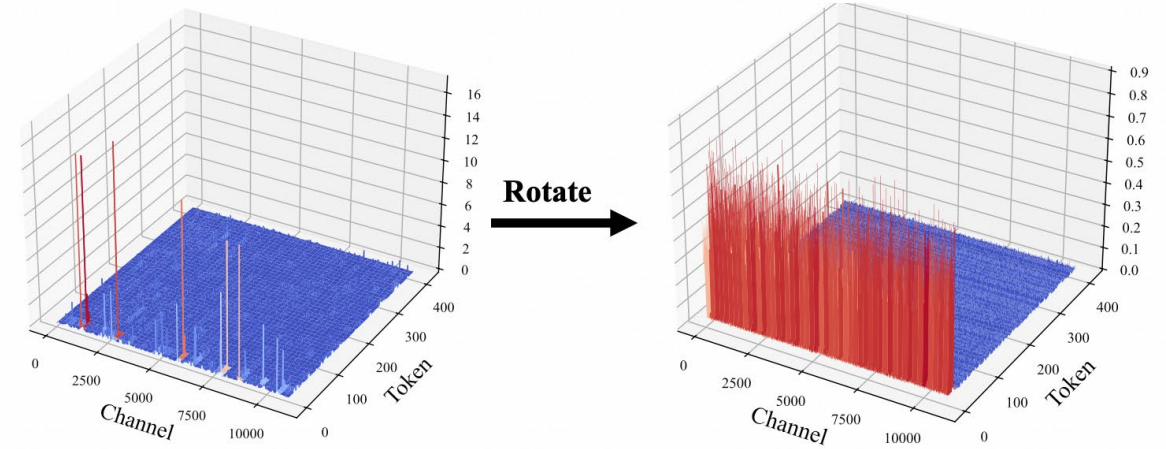


Contribution 3

- Integrate 'Rotation' with Runtime Smooth
 - Further Smooth the rotated Matrix (Figure a)
 - Rotation alleviate the effect of victim (Figure b)



(a)



(b)

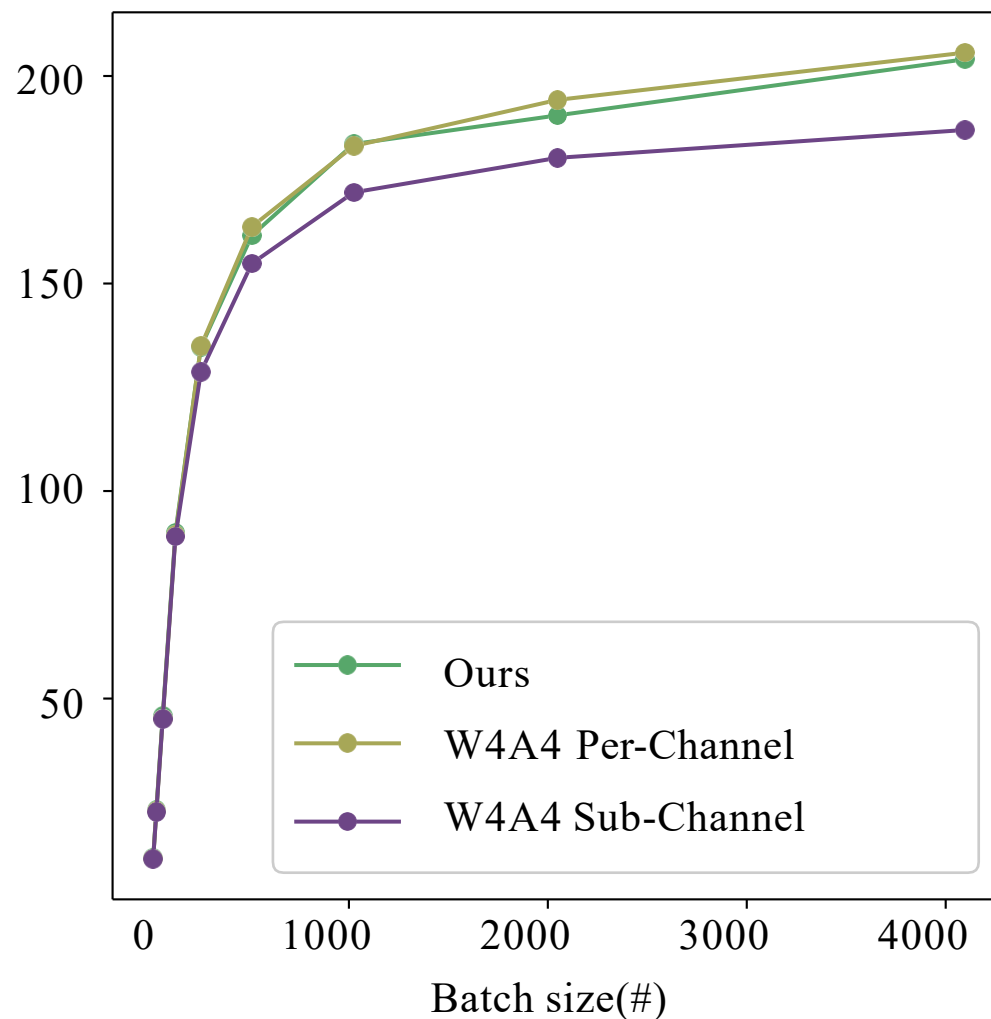
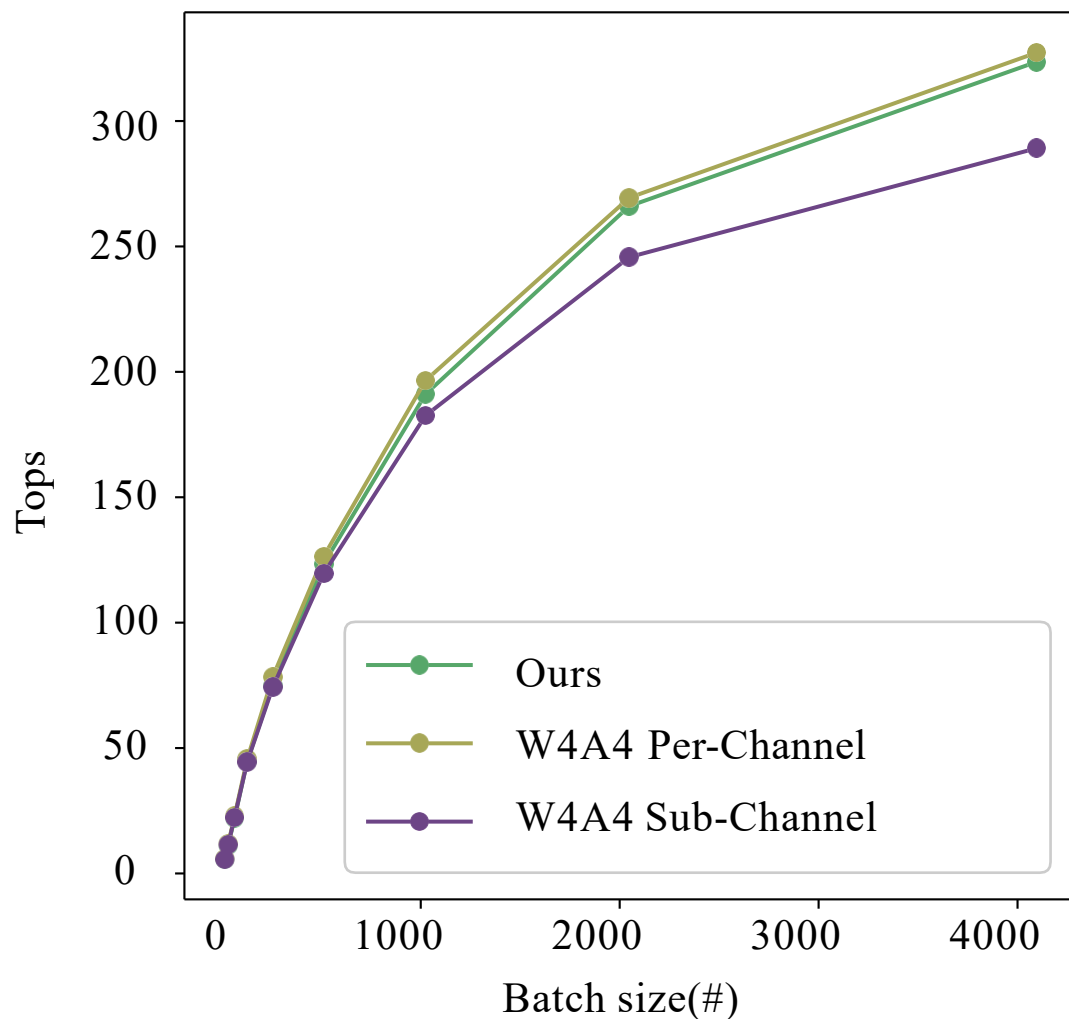
Result

- Runtime Smooth (RS) already attain **most of the accuracy**.
- Rotated Runtime Smooth (RRS) achieve the **best** performance across models.
- For LLama3-70B, RRS suppress PPL from **57.3 to 6.67** compared with previous SOTA

#Bits W-A-KV	Method	LLaMA 2-13B	LLaMA 2-70B	LLaMA 3-8B	LLaMA 3.1-8B	LLaMA 3-70B	LLaMA 3.1-70B	Mixtral	Mistral	Qwen 1.5-7B	Qwen 1.5-14B
16-16-16	FP16	5.00	3.30	6.13	6.24	2.80	2.81	3.84	5.94	7.95	7.44
16-4-16	RTN	5e3	Nan	4e2	2e2	3e4	1e4	8e2	7e2	6e3	2e4
	SmoothQuant	97.04	Nan	5e2	2e2	74.23	48.59	88.68	47.92	2e2	1e2
	RS	6.40	4.52	11.44	9.71	11.31	7.27	5.72	7.39	10.95	9.74
	QuaRot	5.24	3.72	7.77	7.81	1e2	5.67	4.68	6.29	9.07	8.18
	RRS	5.22	3.74	7.55	7.60	6.32	5.03	4.50	6.21	8.91	8.16
4-4-16	RTN	7e3	2e5	9e2	5e2	1e5	2e4	8e2	7e2	9e3	3e4
	SmoothQuant	34.50	1e2	8e2	4e2	5e2	2e2	3e2	76.25	3e2	1e2
	GPTQ	5e3	2e6	9e2	4e2	1e5	2e4	1e3	6e2	1e4	3e4
	RS	8.79	6.95	10.47	10.39	7e3	1e4	7.37	8.16	13.32	10.38
	QuaRot	5.39	3.85	8.38	8.38	57.33	6.26	4.80	6.38	9.34	8.32
	RRS	5.36	3.86	8.11	8.12	6.66	5.56	4.63	6.31	9.17	8.29
4-4-4	RTN	7e3	2e5	1e3	6e2	1e5	2e4	9e2	7e2	1e4	2e4
	SmoothQuant	56.60	1e2	1e3	6e2	5e2	3e2	2e2	78.39	3e2	2e2
	GPTQ	5e3	1e6	1e3	5e2	1e5	2e4	1e3	6e2	1e4	3e4
	RS	9.15	7.08	11.52	11.19	8e3	1e4	7.98	8.64	13.97	10.62
	QuaRot	5.51	3.89	8.76	8.80	49.73	6.46	4.93	6.45	9.55	8.43
	RRS	5.45	3.89	8.42	8.49	6.87	5.69	4.74	6.35	9.37	8.35

Result

- Runtime Smooth (RS) only bring negligible overhead.





Take-away message

- Runtime Smooth: a plug-and-play operation that smooth activation for accurate INT4 inference.
- Rotated Runtime Smooth: a framework that integrate Rotation and Runtime Smooth, which gain the SOTA accuracy for INT4 inference.
- Runtime Smooth would bring negligible overhead.

Thank you for listening!

