

# FlattenQuant: Breaking Through the Inference Compute-bound for Large Language Models with Per-tensor Quantization

LREC-COLING 2024

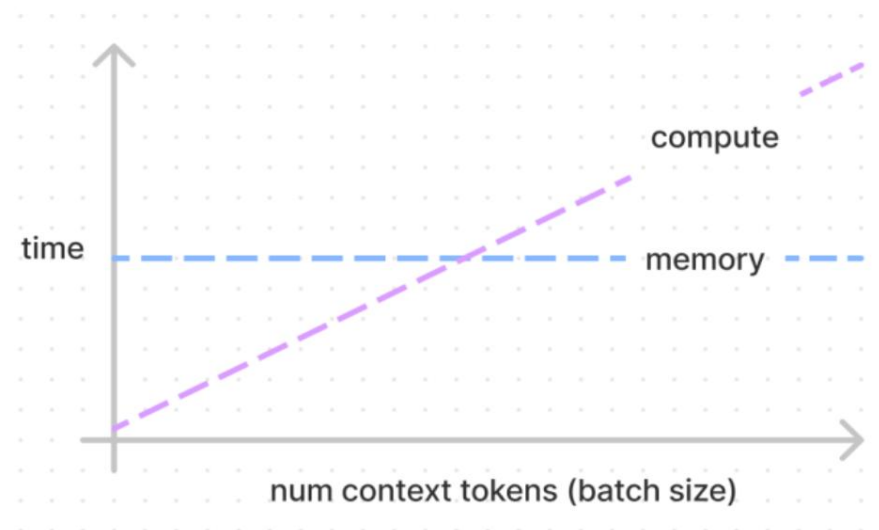
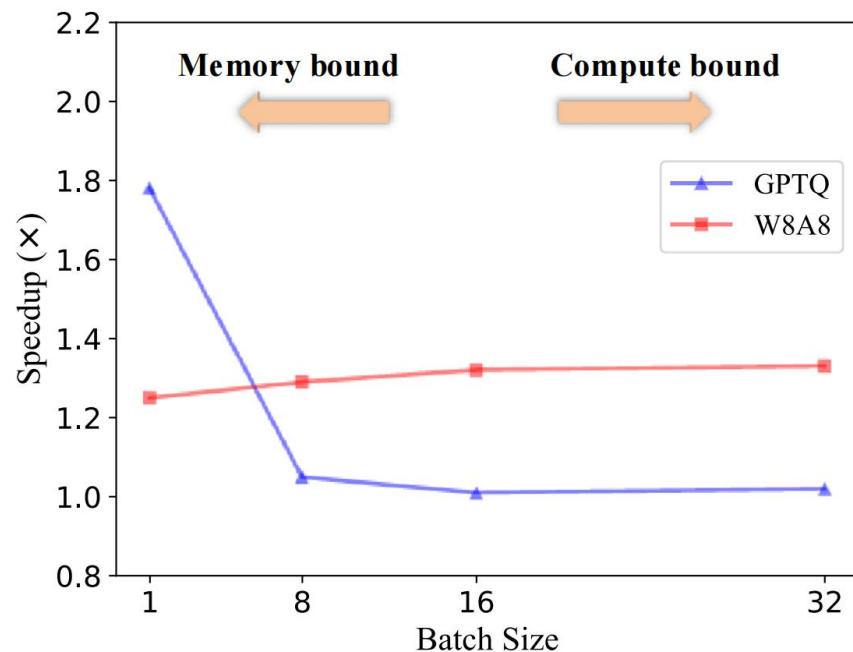
Yi Zhang



# Target scenarios — Compute bound



As the sequence length or batch size increases, the compute-bound replaces the memory-bound as the main factor contributing to latency.



# Target scenarios — Utilizing higher-performance low-bit TensorCore

之江实验室



ZHEJIANG LAB

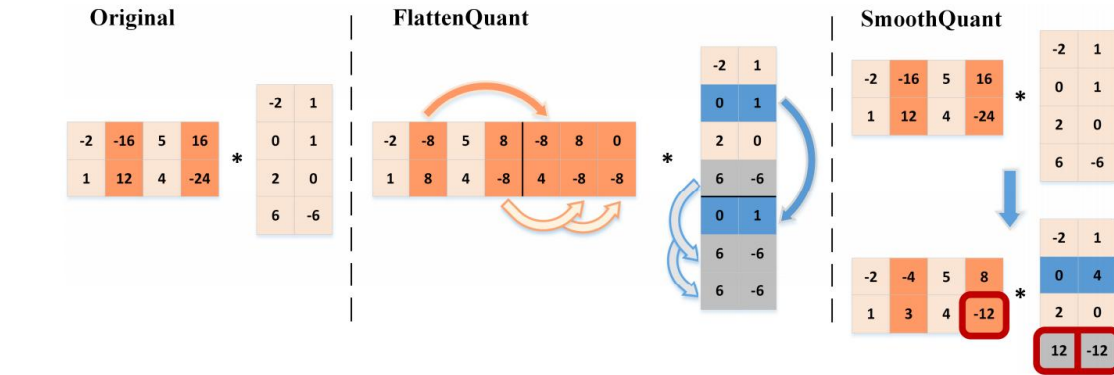
	V100	A100	A100 Sparsity <sup>1</sup>	A100 Speedup	A100 Speedup with Sparsity
A100 FP16 vs V100 FP16	31.4 TFLOPS	78 TFLOPS	NA	2.5x	NA
A100 FP16 TC vs V100 FP16 TC	125 TFLOPS	312 TFLOPS	624 TFLOPS	2.5x	5x
A100 BF16 TC vs V100 FP16 TC	125 TFLOPS	312 TFLOPS	624 TFLOPS	2.5x	5x
A100 FP32 vs V100 FP32	15.7 TFLOPS	19.5 TFLOPS	NA	1.25x	NA
A100 TF32 TC vs V100 FP32	15.7 TFLOPS	156 TFLOPS	312 TFLOPS	10x	20x
A100 FP64 vs V100 FP64	7.8 TFLOPS	9.7 TFLOPS	NA	1.25x	NA
A100 FP64 TC vs V100 FP64	7.8 TFLOPS	19.5 TFLOPS	NA	2.5x	NA
A100 INT8 TC vs V100 INT8	62 TOPS	624 TOPS	1248 TOPS	10x	20x
A100 INT4 TC	NA	1248 TOPS	2496 TOPS	NA	NA
A100 Binary TC	NA	4992 TOPS	NA	NA	NA

Table 1: Setting of the LLMs quantification

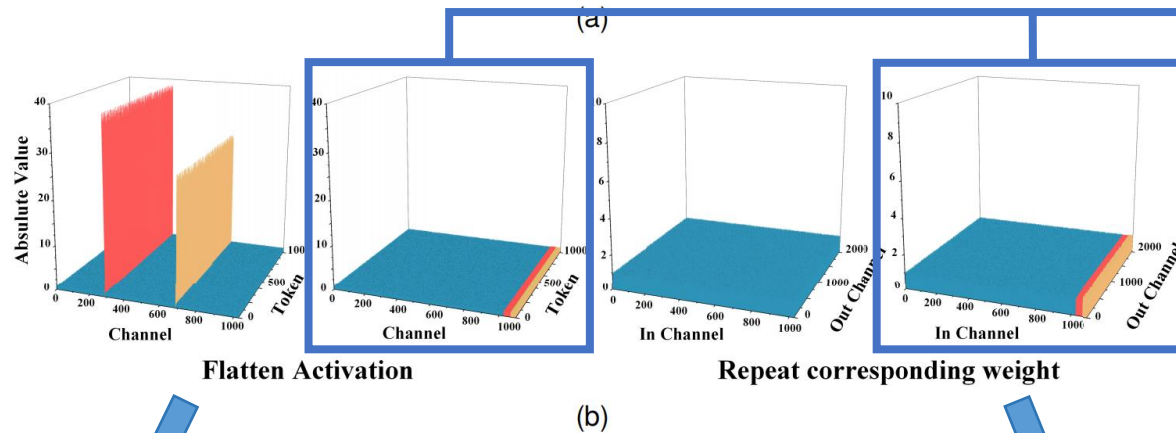
Method	Activation	Weight	Compute data type
W8A8	per-tensor static	per-tensor	INT8
LLM.int8()	per-token dynamic	per-channel	INT8+FP16
SmoothQuant	per-tensor dynamic/static	per-tensor	INT8
RPTQ	group-wise static	group-wise	FP16
GPTQ	not quant	per-channel	FP16

**INT4 per-tensor quantization can maximize the computational power of the A100 TensorCore**

# FlattenQuant — The basic idea



Easy to be per-tensor quantized



$$X_k = \text{round}\left(\frac{X_{fp16}}{s}\right), \quad s = \frac{\max(|X_{fp16}|)}{2^{k-1} - 1}$$

$$\text{Flatten}(X_{ij}) = \vec{X}_{\tilde{j}} = \left[ \underbrace{\left\lfloor \frac{X_{ij}}{T} \right\rfloor}_{T}, \dots, T, X_{ij} \bmod T \right]$$

$$\tilde{j} \in [j \cup (C + \sum_{k=1}^{j-1} E_k, C + \sum_{k=1}^{j-1} E_k + \lfloor \frac{X_{ij}}{T} \rfloor)]$$

$$\text{Repeat}(W_j) = E_j \begin{Bmatrix} 1 \\ 1 \\ \vdots \end{Bmatrix} W_j$$

# Achieving High-precision — ChannelSmooth



**Aim to acquire uniformly distributed of channel values.**

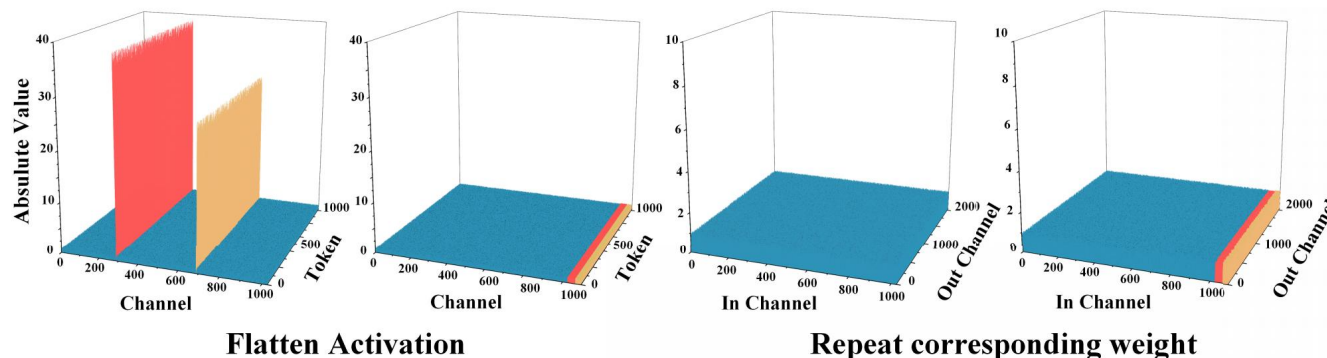
$$\text{Smooth}(X, W) = (X \text{diag}(s)^{-1}, \text{diag}(s)W)$$

$$s_j = \frac{(\text{Sigmoid}((\max(|X_j|) - \mu)/\sigma))^\alpha}{(\text{Sigmoid}((\max(|W_j|) - \mu)/\sigma))^{1-\alpha}}$$

**We define  $\mu$  as**  $\sum_{k=1}^C \max(|X_k|)/C$

**where  $\sigma$  is**  $(\sum_{k=1}^C (\max(|X_k|) - \mu)^2 / C)^{0.5}$

# Achieving High-precision — Flatten + Repeat

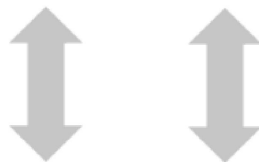


Operator	Flatten	Gemm (FP16)	Gemm (INT8)	Gemm (INT4)
Latency (ms)	0.19	3.12	2.35	1.57

**The latency of tensor flattening operation is very small by comparison**

**weight:**

**Repeat+Flatten**



**activation:**

**Flatten+Repeat**

**Done in advance.  
Quantized to low-bit and saved.**

**Performed in inference.**

# Achieving High-precision — Selection of the truncation threshold

之江实验室



ZHEJIANG LAB

**step 1: Remove outliers by boxplot.**

**step 2: The truncation threshold is the mean of the maximum values of each channel multiplied by coefficient  $\beta$ .**

$$\left| \widetilde{X}_j \right| = \text{clip}(|X_j|, \min = Q_1 - 1.5 \cdot IQR, \\ \max = Q_3 + 1.5 \cdot IQR)$$

$$T = \beta \cdot \frac{\sum_{j=1}^C \max(|\widetilde{X}_j|)}{C}$$

# Achieving High-precision — Quantize some layers to INT4

之江实验室



ZHEJIANG LAB

To maintain accuracy, tensors are quantized separately using INT4 and INT8.

The KL divergence caused by quantization is obtained. Expressed as  $KL(P, Q_{INT4})$  and  $KL(P, Q_{INT8})$

The  $\gamma$  determines whether to use INT4 quantization.

$$\text{QuantBits} = \begin{cases} 4 & KL(P, Q_{INT4})/KL(P, Q_{INT8}) < \gamma \\ 8 & KL(P, Q_{INT4})/KL(P, Q_{INT8}) \geq \gamma \end{cases}$$

Table 4: The proportion of INT4 quantized layers in LLMs, and the average ratio of expanded channels over original channels.

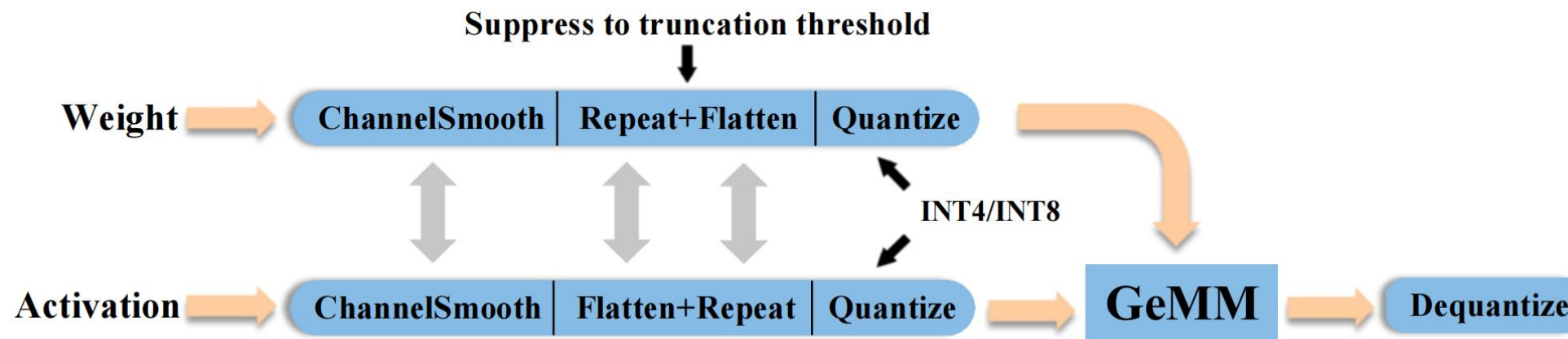
Model	125M	1.3B	6.7B	13B	30B	66B
INT4 layers	35.54%	36.42%	47.25%	45.75%	48.29%	47.37%
Flatten ratio	26.21%	24.62%	24.41%	21.50%	22.42%	21.65%

# FlattenQuant — Optimized entire workflow

之江实验室



ZHEJIANG LAB



- ChannelSmooth
  - Flatten + Repeat
  - Quantize to INT4 by per-tensor as much as possible while ensuring accuracy.
- Easy to be per-tensor quantized

**The baseline we are comparing against is the existing per-tensor quantization method.**

Table 2: Quantization setting of the baselines and FlattenQuant. All weight and activations use per-tensor static quantization. In the case of FlattenQuant-O3, we adopt per-tensor quantization instead of per-channel quantization, during the GPTQ optimization process for weight.

Method	Bits	Additional Optimizations
W8A8	8	N/A
Smoothquant	8	N/A
FlattenQuant-O1	8	N/A
FlattenQuant-O2	4,8 mixed	N/A
FlattenQuant-O3	4,8 mixed	GPTQ

# Experiments — Accuracy

It is evident that the accuracy achieved by FlattenQuant under the O3 configuration is comparable to that of SmoothQuant.

Task	OpenBookQA (↑)						LAMBADA (OpenAI)(↑)					
Model	125M	1.3B	6.7B	13B	30B	66B	125M	1.3B	6.7B	13B	30B	66B
FP16	27.80%	33.40%	37.40%	39.00%	40.20%	40.80%	37.84%	57.90%	67.66%	68.60%	71.39%	73.92%
W8A8	27.60%	34.20%	28.20%	25.60%	27.80%	27.33%	35.53%	54.39%	13.31%	0.05%	0.03%	0.02%
SmoothQuant	<b>27.20%</b>	<b>33.60%</b>	<b>37.40%</b>	<b>38.85%</b>	<b>39.63%</b>	<b>39.82%</b>	<b>37.41%</b>	<b>55.24%</b>	<b>67.03%</b>	<b>68.02%</b>	<b>71.20%</b>	<b>72.79%</b>
FlattenQuant-O1	27.80%	33.00%	37.80%	39.20%	39.27%	39.87%	36.73%	56.47%	66.13%	67.41%	71.22%	73.12%
FlattenQuant-O2	27.11%	31.00%	35.60%	38.11%	37.74%	36.52%	35.88%	55.63%	65.47%	66.03%	68.58%	69.91%
FlattenQuant-O3	<b>27.60%</b>	<b>33.22%</b>	<b>37.01%</b>	<b>38.71%</b>	<b>39.11%</b>	<b>39.46%</b>	<b>36.13%</b>	<b>56.51%</b>	<b>66.41%</b>	<b>67.20%</b>	<b>70.82%</b>	<b>72.63%</b>

Task	PIQA (↑)						HellaSwag (↑)					
Model	125M	1.3B	6.7B	13B	30B	66B	125M	1.3B	6.7B	13B	30B	66B
FP16	61.53%	71.36%	76.49%	76.87%	78.12%	79.76%	31.32%	53.73%	67.18%	69.80%	72.27%	74.88%
W8A8	61.26%	69.85%	59.30%	52.50%	53.15%	53.88%	30.92%	51.89%	39.31%	27.44%	28.34%	28.01%
SmoothQuant	<b>61.75%</b>	<b>71.04%</b>	<b>76.10%</b>	<b>76.12%</b>	<b>77.52%</b>	<b>79.01%</b>	<b>31.42%</b>	<b>53.50%</b>	<b>66.69%</b>	<b>67.54%</b>	<b>70.85%</b>	<b>72.89%</b>
FlattenQuant-O1	61.47%	70.89%	75.78%	76.06%	78.01%	79.34%	31.55%	53.09%	65.97%	68.68%	71.31%	73.62%
FlattenQuant-O2	60.77%	69.04%	74.51%	75.89%	76.07%	76.52%	31.21%	50.09%	63.45%	66.38%	68.51%	70.11%
FlattenQuant-O3	<b>61.20%</b>	<b>71.02%</b>	<b>75.43%</b>	<b>76.31%</b>	<b>77.58%</b>	<b>79.07%</b>	<b>31.56%</b>	<b>53.40%</b>	<b>66.31%</b>	<b>68.70%</b>	<b>71.35%</b>	<b>73.10%</b>

Task	WinoGrande (↑)						WikiText2 (↓)					
Model	125M	1.3B	6.7B	13B	30B	66B	125M	1.3B	6.7B	13B	30B	66B
FP16	50.43%	59.19%	65.19%	65.03%	68.42%	68.90%	27.65	14.62	10.86	10.13	9.56	9.34
W8A8	49.90%	58.40%	49.56%	51.30%	50.74%	49.72%	30.10	15.85	31.08	4551.63	2130.69	3754.28
SmoothQuant	<b>50.03%</b>	<b>59.11%</b>	<b>64.90%</b>	<b>64.21%</b>	<b>68.52%</b>	<b>68.21%</b>	<b>28.88</b>	<b>15.69</b>	<b>11.43</b>	<b>10.99</b>	<b>10.62</b>	<b>10.55</b>
FlattenQuant-O1	49.01%	58.48%	64.56%	64.24%	68.11%	68.33%	28.46	15.22	11.28	10.87	10.34	10.24
FlattenQuant-O2	48.21%	57.85%	62.96%	63.21%	65.47%	66.43%	30.86	16.80	13.30	12.26	11.72	11.37
FlattenQuant-O3	<b>50.35%</b>	<b>58.90%</b>	<b>64.77%</b>	<b>64.41%</b>	<b>68.10%</b>	<b>68.12%</b>	<b>28.75</b>	<b>15.94</b>	<b>11.68</b>	<b>11.21</b>	<b>10.88</b>	<b>10.77</b>

# Experiments — Memory consumption

## Memory consumption (GB) of LLMs on different batch size and sequence length

Sequence Length		512			1024			2048		
Batch Size		1	8	32	1	8	32	1	8	32
OPT-30b	FP16	59.2	60.4	66.2	60.1	62.3	72.3	62.8	73.4	88.6
	SmoothQuant	32.3	33.2	38.2	32.6	33.5	40.5	37.2	42.1	48.9
	FlattenQuant-O3	<b>26.1</b>	<b>26.8</b>	<b>31.7</b>	<b>26.3</b>	<b>27.5</b>	<b>33.5</b>	<b>31.8</b>	<b>36.1</b>	<b>39.8</b>
OPT-66b	FP16	126.3	126.8	133.5	127.2	129.1	140.1	130.6	143.5	162.6
	SmoothQuant	67.4	68.2	73.1	67.3	72.2	75.9	69.6	79.9	91.2
	FlattenQuant-O3	<b>53.7</b>	<b>55.4</b>	<b>60.2</b>	<b>56.2</b>	<b>59.6</b>	<b>60.1</b>	<b>55.4</b>	<b>61.2</b>	<b>70.5</b>

# Experiments — Speedup

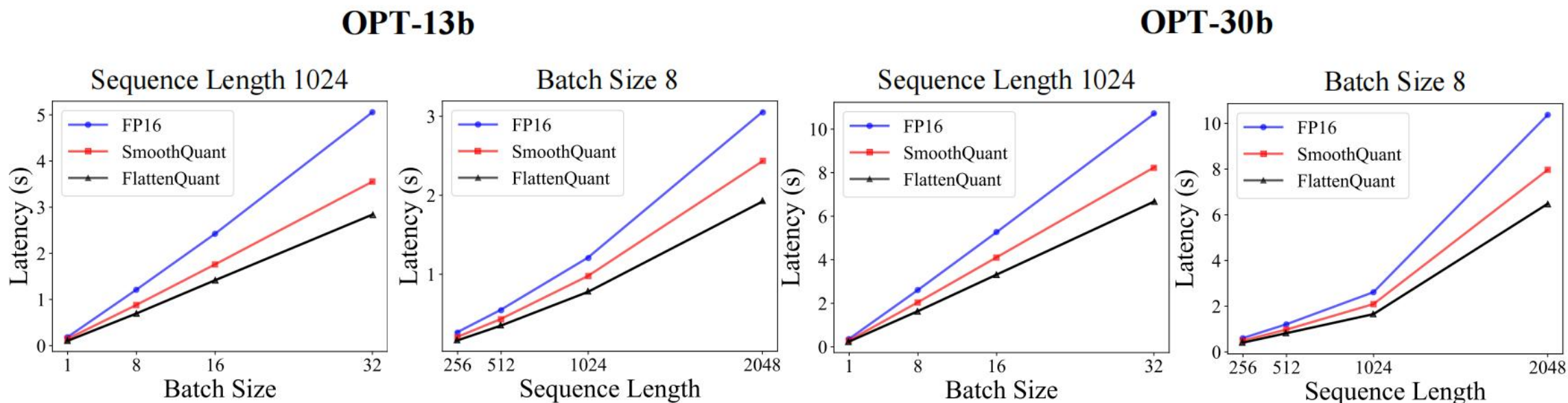


Figure 4: In compute-bound scenarios that involve large batch sizes and long sequences, our per-tensor method replaced FP16 matrix multiplication with INT4 and INT8 alternatives, yielding a considerable improvement in inference speed. This achievement is a direct result of our targeted efforts to overcome compute-bound challenges.



## The performance impact of the channel smoothing

Model	OPT-1.3b		OPT-6.7b		OPT-13b	
	Smooth	Yes	No	Yes	No	Yes
Wiki PPL (↓)	15.94	16.50	11.68	12.16	11.21	11.85

$$\text{Smooth}(X, W) = (X \text{diag}(s)^{-1}, \text{diag}(s)W)$$
$$s_j = \frac{(\text{Sigmoid}((\max(|X_j|) - \mu)/\sigma))^\alpha}{(\text{Sigmoid}((\max(|W_j|) - \mu)/\sigma))^{1-\alpha}}$$

## Impact of suppressing outlier channels during obtaining truncation threshold

Model	OPT-1.3b		OPT-6.7b		OPT-13b	
	Outlier clip	Yes	No	Yes	No	Yes
Wiki PPL (↓)	15.94	16.22	11.68	11.87	11.21	11.74

$$\left| \widetilde{X}_j \right| = \text{clip}(|X_j|, \min = Q_1 - 1.5 \cdot IQR, \max = Q_3 + 1.5 \cdot IQR) \quad (6)$$

$$T = \beta \cdot \frac{\sum_{j=1}^C \max(|\widetilde{X}_j|)}{C} \quad (7)$$

# Experiments — Ablation Study

The parameter  $\beta$  with respect to accuracy, flatten ratio and GPU memory consumption

$\beta$	1.1	1.2	1.3	1.4	1.5
Wiki PPL ( $\downarrow$ )	11.32	11.38	11.68	11.89	12.08
Flatten ratio	32.25%	26.85%	24.41%	22.01%	20.56%
Memory (GB)	7.12	6.89	6.75	6.66	6.72

$$|\widetilde{X}_j| = \text{clip}(|X_j|, \min = Q_1 - 1.5 \cdot IQR, \max = Q_3 + 1.5 \cdot IQR) \quad (6)$$

$$T = \boxed{\beta} \cdot \frac{\sum_{j=1}^C \max(|\widetilde{X}_j|)}{C} \quad (7)$$

$\gamma$  determines the tolerance of the INT4 quantization that affects KL divergence

$$\text{QuantBits} = \begin{cases} 4 & \text{KL}(P, Q_{\text{INT4}})/\text{KL}(P, Q_{\text{INT8}}) < \boxed{\gamma} \\ 8 & \text{KL}(P, Q_{\text{INT4}})/\text{KL}(P, Q_{\text{INT8}}) \geq \boxed{\gamma} \end{cases}$$

$\gamma$	1.82	1.84	1.86	1.88	1.90
INT4 layers	31.21%	40.85%	47.25%	51.67%	55.85%
Wiki PPL ( $\downarrow$ )	11.35	11.37	11.68	11.95	12.47
Memory (GB)	7.32	7.02	6.75	6.58	6.29

# Thanks

