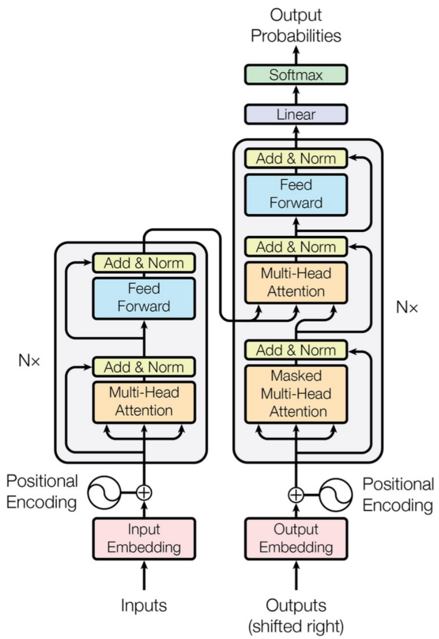
QLLM: Accurate and Efficient Low-Bitwidth Quantization for Large Language Models

Jing Liu[1,2], Ruihao Gong[2,3], Xiuying Wei[2,4], Zhiwei Dong[2,5], Jianfei Cai[1], Bohan Zhuang[1†]

[1]Zip Lab, Monash University [2]SenseTime Research [3]Beihang University
[4]School of Computer and Communication Sciences, EPFL,
[5]University of Science and Technology Beijing

Large Language Models



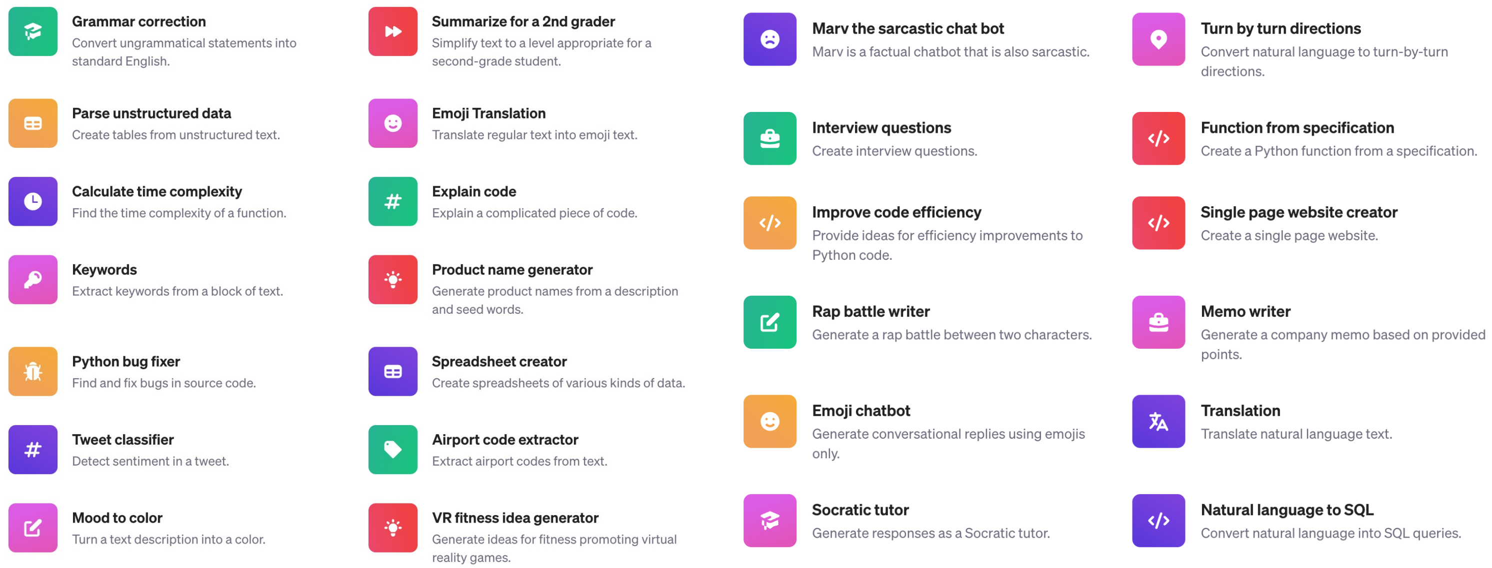


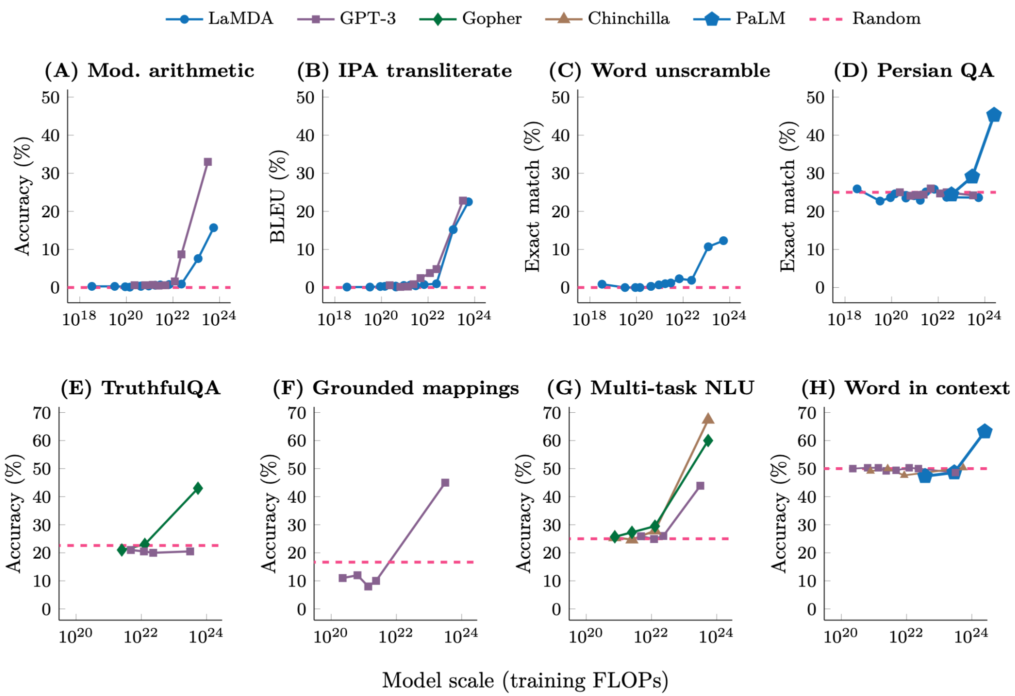
Image Source: OpenAI Admin Panel

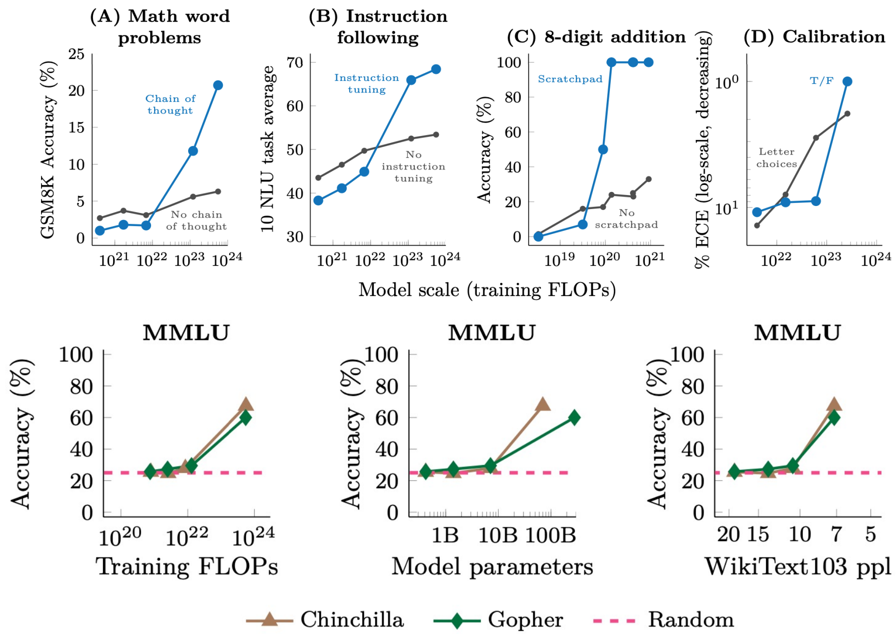
Large Language Models (LLMs) have achieved unprecedented advancements in NLP.

Vaswani *et al*. Attention Is All You Need. NeurIPS 2017.

Liu et al. QLLM – ICLR 2024

Emergent Abilities of LLM

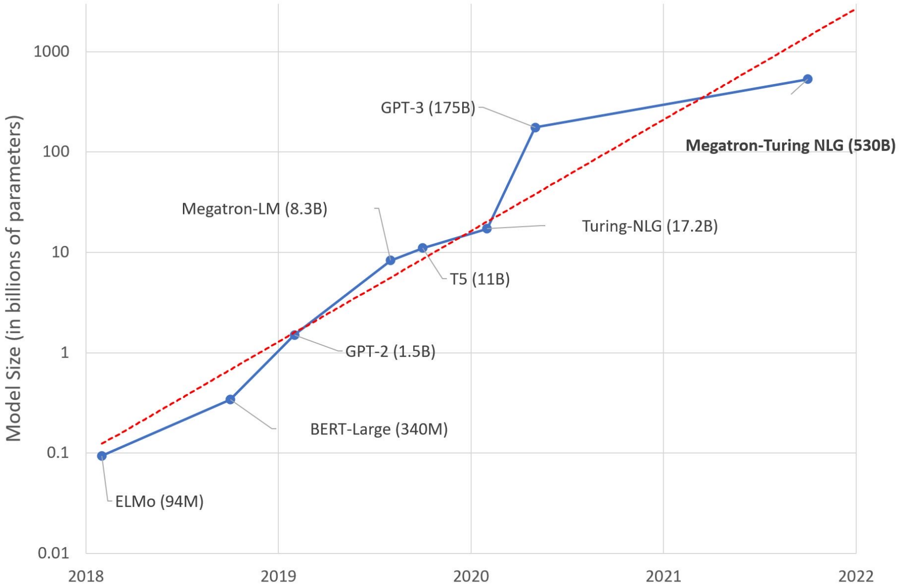




LLMs suddenly gain new emergent abilities as they grow.

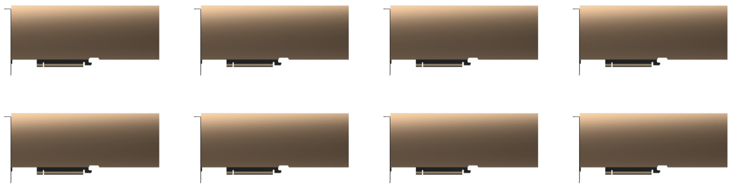
Wei *et al*. Emergent Abilities of Large Language Models. TMLR 2022.

LLM Deployment Requirements



- Serving a 175B GPT-3 model at least requires
  - 80GB NVIDIA A100 GPUs

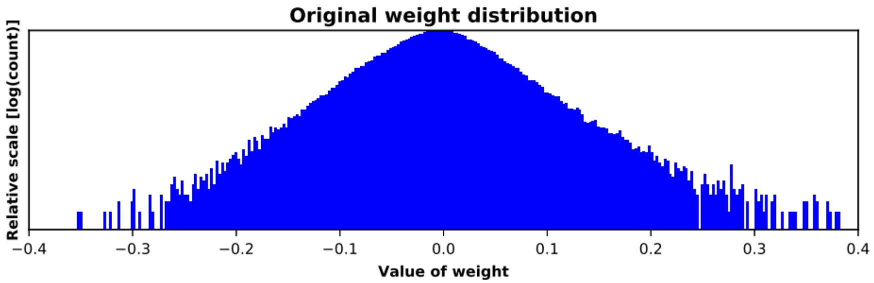
  - 40GB NVIDIA A40 GPUs

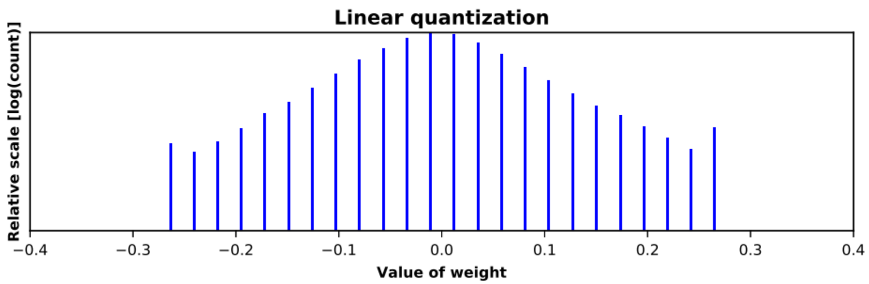


The substantial computational demands and vast model sizes of LLM necessitates lot of GPUs during inference.

[Large Language Models: A New Moore's Law? (huggingface.co)]

Network Quantization





- Linear quantization:

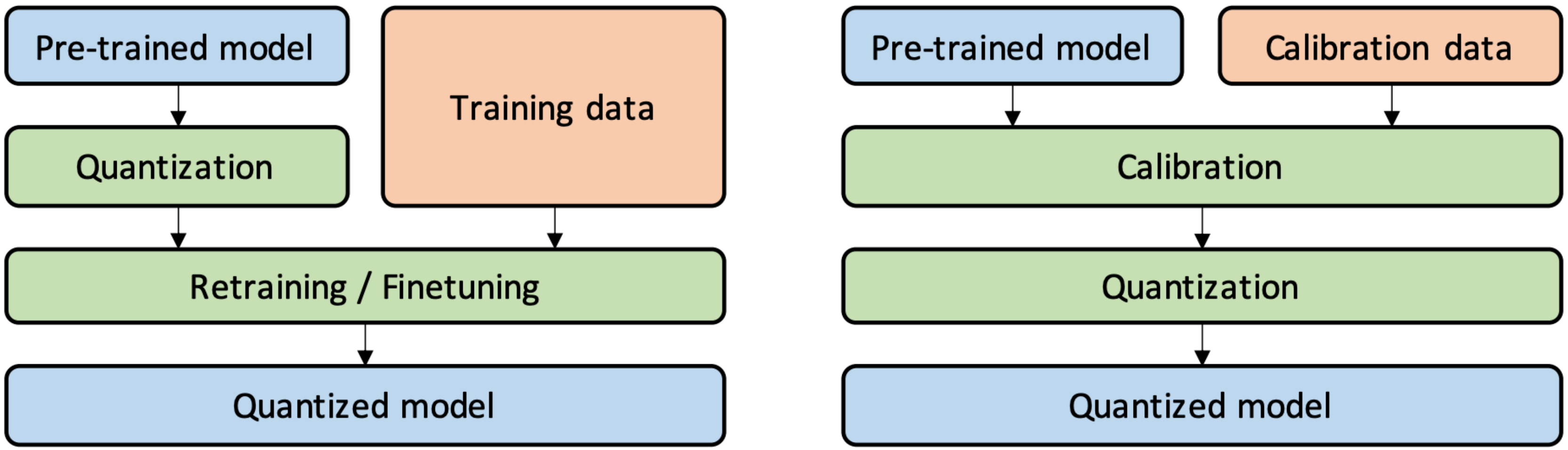
$$\mathbf{X}_q = \mathrm{quant}(\mathbf{X}) = \mathrm{clamp}\left(\left\lfloor \frac{\mathbf{X}}{\alpha} \right\rceil + \beta, 0, 2^b - 1\right),$$

$$\text{where } \alpha = \frac{\max(\mathbf{X}) - \min(\mathbf{X})}{2^b - 1}, \beta = -\left\lfloor \frac{\min(\mathbf{X})}{\alpha} \right\rceil$$

Network quantization represents the weights and activations with low-precision, resulting in lower memory footprint and faster inference.

Park, *et al*. Weighted-Entropy-Based Quantization for Deep Neural Networks. CVPR 2017.

Quantization-aware Training vs Post-training Quantization

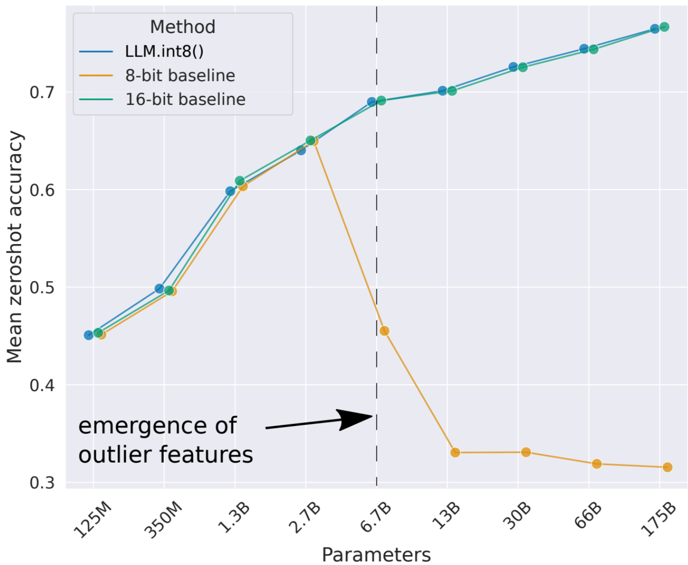


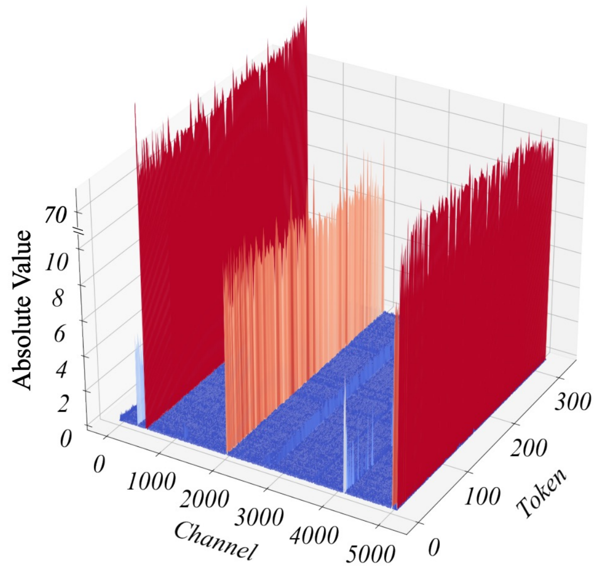
Training Cost: High 😫 Training Cost: Low 😁

QAT suffers from unbearable training costs, rendering it impractical for the efficient deployment of LLMs.

Gholami, *et al*. A Survey of Quantization Methods for Efficient Neural Network Inference. Low-Power Computer Vision 2022.

Outlier issues in Post-training Quantization

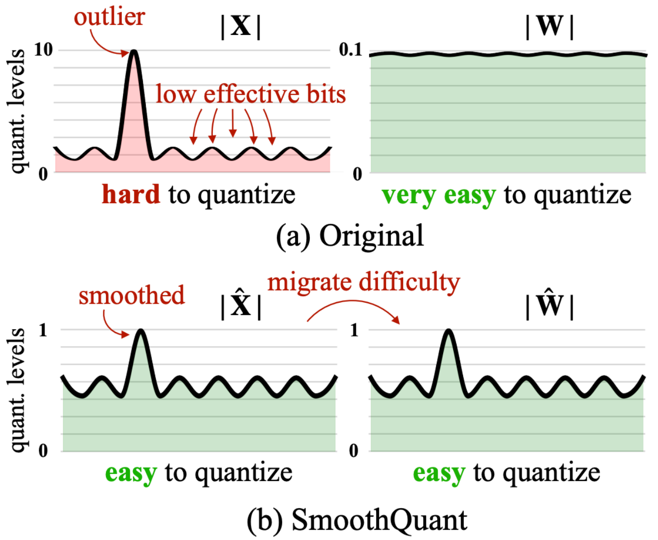


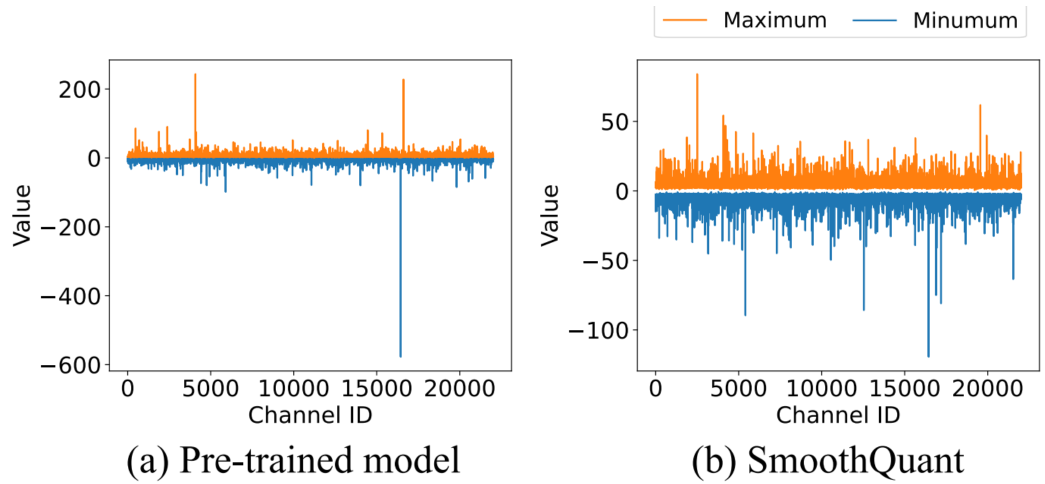


- ❑ Activation outliers emerge when scaling up beyond 6.7B parameters.
- ❑ The activation outliers in specific channels make existing quantization methods less effective.

Tim, *et al*. LLM.int8 (): 8-bit matrix multiplication for transformers at scale. NeurIPS 2022.
Xiao, *et al*. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. ICML 2023.

Existing Quantization Methods

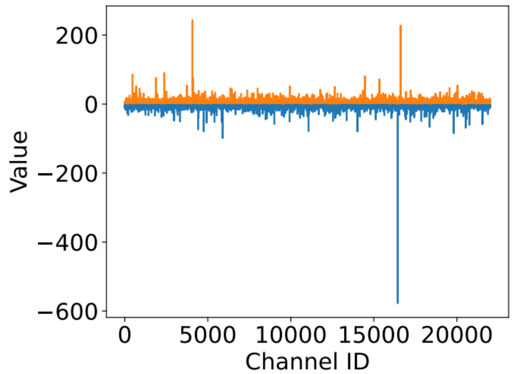




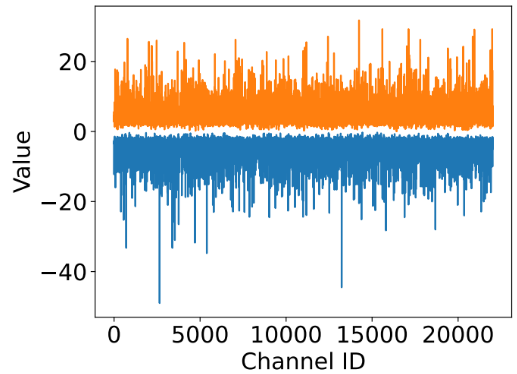
- Existing quantization methods smoothing activation outliers by **transitioning the magnitudes from activations to weights** through a mathematically equivalent transformation.
- For exceeding pronounced activation outliers, existing methods offers only limited alleviation.

Xiao, *et al*. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. ICML 2023.

QLLM: Accurate and Efficient Low-bitwidth Quantization for LLMs



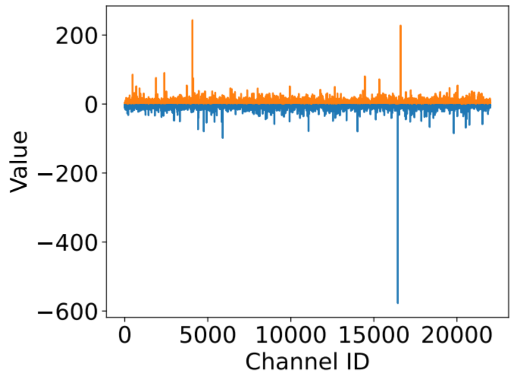
(a) Pre-trained model



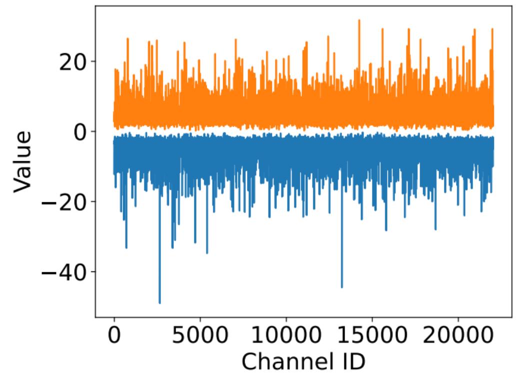
(c) QLLM

- QLLM handle the outlier issue by employing **a gradient-free channel reassembly** that redistributes the large activation magnitude of the outlier channels across the channel.
- QLLM further improve the performance of the quantized LLM through **a efficient gradient-based error correction**, which learns low-rank parameters to counteract quantization error.

Channel Reassembly



(a) Pre-trained model



(c) QLLM

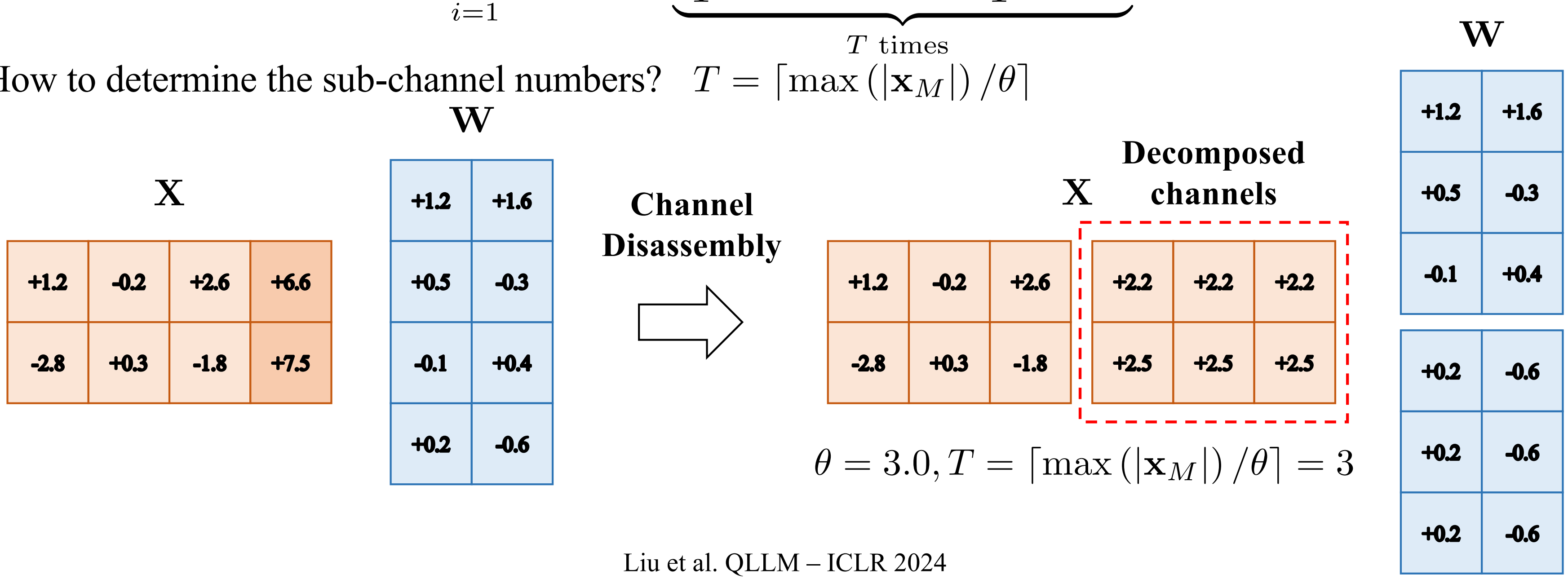
- Channel disassembly: **decompose the input outlier channels** into several sub-channels, which reduces outlier magnitude and make the activations more quantization-friendly.
- Channel assembly: **merge similar input channels** to keep the original channel count.
- Adaptive reassembly: adaptively **determine the appropriate reassembly ratio** for each layer.

Channel Disassembly

- ❑ Channel disassembly: decompose outlier channel channel M into $\frac{\mathbf{x}_M}{T}$ and replicate this channel T times

$$\mathbf{y}_k = \sum_{i=1}^{M-1} \mathbf{x}_i \mathbf{W}_{ik} + \underbrace{\frac{\mathbf{x}_M}{T}\mathbf{W}_{Mk} + \cdots + \frac{\mathbf{x}_M}{T}\mathbf{W}_{Mk}}_{T \text{ times}}$$

- ❑ How to determine the sub-channel numbers? $T = \lceil \max\left(|\mathbf{x}_M|\right)/\theta \rceil$



$\theta = 3.0, T = \lceil \max\left(|\mathbf{x}_M|\right)/\theta \rceil = 3$

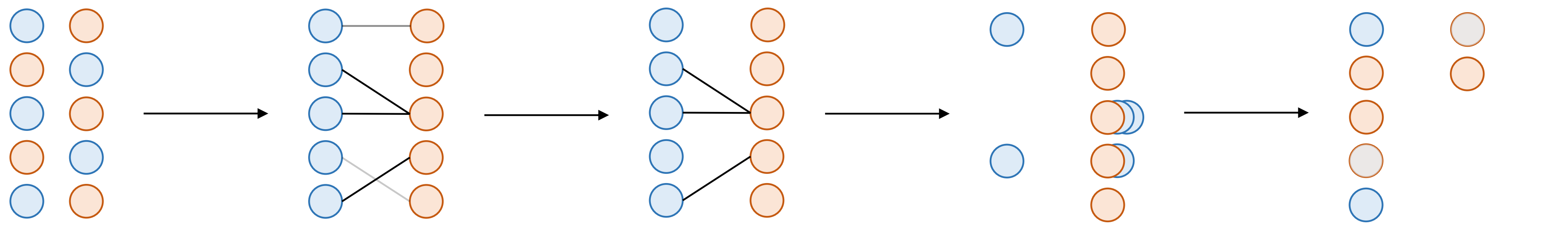
Liu et al. QLLM – ICLR 2024

Channel Assembly

- Merge similar channels (ignore outlier channels) with the aim of reducing the information loss.

$$D(i,j) = \left\| \mathbf{x}_i \mathbf{W}_{ik} + \mathbf{x}_j \mathbf{W}_{jk} - \underbrace{\boxed{\frac{\mathbf{x}_i + \mathbf{x}_j}{2}}}_{\text{Aggregated channels}} (\mathbf{W}_{ik} + \mathbf{W}_{jk}) \right\|_2^2 = \left\| \frac{\mathbf{x}_i(\mathbf{W}_{ik} - \mathbf{W}_{jk})}{2} + \frac{\mathbf{x}_j(\mathbf{W}_{jk} - \mathbf{W}_{ik})}{2} \right\|_2^2,$$

- How to determine which channels to aggregate to reduce the total number by $T-1$? Bipartite soft matching.



Step 1: Assign Channels to Set A or Set B

Step 2: For each channel in Set A, construct an edge to its most similar counterpart in Set B

Step 3: Select the $T-1$ most similar edges.

Step 4: Aggregate the channels that remain connected.

Step 5: Concatenate the two sets to form the assembled channel set.

Bolya, *et al*. Token Merging: Your ViT But Faster. ICLR 2023.

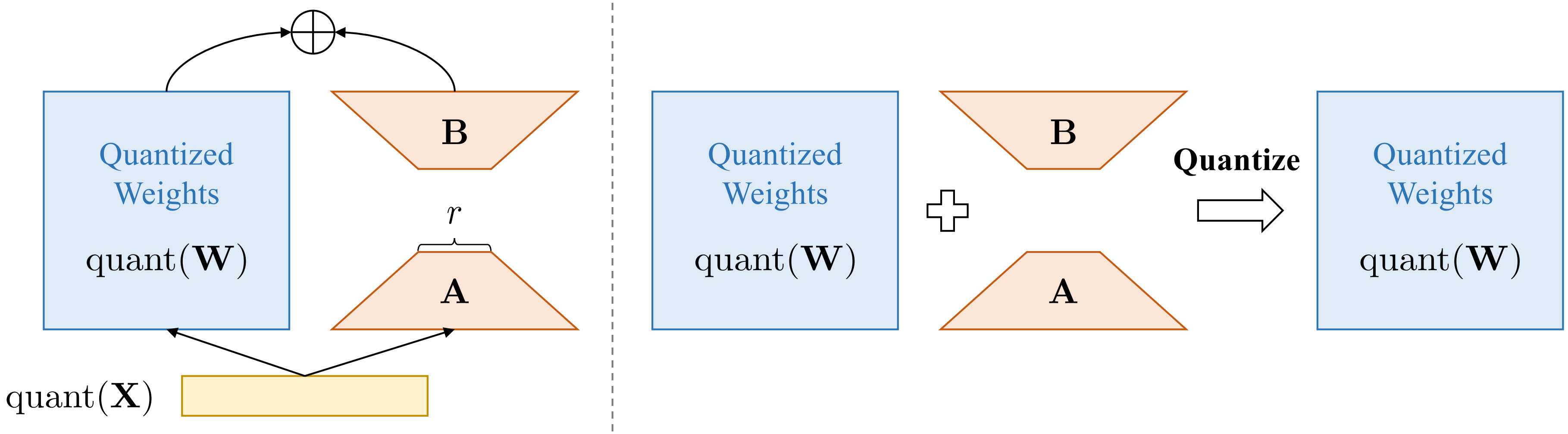
Liu et al. QLLM – ICLR 2024

Adaptive Reassembly

- Question: how to determine reassembly ratio?
  - Selecting a high value for T with a small θ substantially reduces outlier magnitudes and benefits quantization, while resulting a large increase in channel merging error.
  - Choosing a small T with a large θ will not increase too much channel merging loss but may cause significant quantization errors due to the remaining outliers.
- Solution: using grid search to find the optimal θ by minimizing the reassembly error between the original output activations and their counterparts generated with the reassembled input activations.

Efficient Error Correction

- ❑ Learns two low-rank parameters $\mathbf{A} \in \mathbb{R}^{M \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times N}$ for each projection layers by minimizing the block-wise reconstruction error.
- ❑ Perform block-wise reconstruction sequentially rather than parallel to mitigate accumulated error.



Training **Inference**

LLaMA-1 Results

Table. Performance comparisons of different methods for weights and activations quantization on LLaMA-1 model family.

Model	#Bits	Method	PPL ↓			Accuracy (%) ↑					
			WikiText2	C4	Avg.	PIQA	ARC-e	ARC-c	HellaSwag	Winogrande	Avg.
LLaMA-1-7B	W16A16	-	5.68	7.08	6.38	77.37	52.48	41.38	72.99	66.93	62.23
	W6A6	SQ	6.15	7.61	6.88	76.65	53.11	40.10	71.52	61.88	60.65
	W6A6	OS+	5.90	-	-	76.82	51.35	41.13	71.42	65.98	61.34
	W6A6	OmniQuant	5.96	7.43	6.70	77.09	51.89	40.87	71.61	65.03	61.30
	W6A6	QLLM	5.89	7.34	**6.62**	77.26	52.02	41.04	71.40	65.19	**61.38**
	W4A8	QLLM	5.96	7.49	6.73	76.17	50.84	40.02	70.75	66.22	60.80
	W4A4	SQ	52.85	104.35	78.60	49.80	30.40	25.80	27.40	48.00	36.28
	W4A4	LLM-QAT	-	-	-	51.50	27.90	23.90	31.10	51.90	37.26
	W4A4	LLM-QAT+SQ	-	-	-	55.90	35.50	26.40	47.80	50.60	43.24
	W4A4	OS+	40.32	-	-	62.73	39.98	30.29	44.39	52.96	46.07
	W4A4	OmniQuant	11.26	14.51	12.89	66.15	45.20	31.14	56.44	53.43	50.47
	W4A4	QLLM	9.65	12.29	**10.97**	68.77	45.20	31.14	57.43	56.67	**51.84**
LLaMA-1-65B	W16A16	-	3.56	5.62	4.59	80.85	58.75	46.25	80.73	77.11	68.74
	W6A6	SQ	4.00	6.08	5.04	77.97	54.67	44.62	77.51	72.61	65.48
	W6A6	OS+	-	-	-	79.67	55.68	45.22	78.03	73.95	66.51
	W6A6	OmniQuant	3.75	5.82	4.79	81.01	58.12	46.33	79.91	75.69	**68.21**
	W6A6	QLLM	3.73	5.80	**4.77**	80.14	57.79	45.05	79.74	74.59	67.46
	W4A8	QLLM	3.78	8.82	6.30	80.14	58.59	46.42	79.71	74.66	67.90
	W4A4	SQ	112.02	118.96	115.49	61.81	40.15	32.08	46.19	50.83	46.21
	W4A4	OS+	32.60	-	-	68.06	43.98	35.32	50.73	54.30	50.48
	W4A4	OmniQuant	9.17	11.28	10.23	71.81	48.02	35.92	66.81	59.51	56.41
	W4A4	QLLM	6.87	8.98	**7.93**	73.56	52.06	39.68	70.94	62.9	**59.83**

- QLLM achieves significantly **higher zero-shot accuracy** and **much lower perplexity** than the contenders.

LLaMA-2 Results

Table. Performance comparisons of different methods for weights and activations quantization on LLaMA-2 model family.

Model	#Bits	Method	PPL ↓			Accuracy (%) ↑					
			WikiText2	C4	Avg.	PIQA	ARC-e	ARC-c	HellaSwag	Winogrande	Avg.
LLaMA-2-7B	W16A16	-	5.47	6.97	6.22	76.82	53.62	40.53	72.87	67.25	62.22
	W6A6	SQ	6.37	7.84	7.11	75.57	53.62	39.93	71.76	66.14	61.40
	W6A6	OS+	-	-	-	76.22	52.74	40.70	71.89	65.19	61.35
	W6A6	OmniQuant	5.87	7.48	6.68	76.77	52.90	40.61	71.86	64.09	61.25
	W6A6	QLLM	5.72	7.31	**6.52**	77.48	52.99	39.33	71.38	65.98	**61.43**
	W4A8	QLLM	5.91	7.50	6.71	76.11	51.73	39.33	71.27	65.59	60.81
	W4A4	SQ	101.77	93.21	97.49	60.17	35.23	27.13	37.08	49.57	41.84
	W4A4	OS+	-	-	-	63.11	39.10	28.84	47.31	51.3	45.93
	W4A4	OmniQuant	14.61	18.39	16.50	65.94	43.94	30.80	53.53	55.09	49.86
	W4A4	QLLM	11.75	13.26	**12.51**	67.68	44.40	30.89	58.45	56.59	**51.60**
LLaMA-2-70B	W16A16	-	3.32	5.52	4.42	81.01	59.68	47.95	80.87	76.95	69.29
	W6A6	SQ	3.69	5.88	4.79	79.87	57.32	45.65	79.01	74.03	67.18
	W6A6	OS+	-	-	-	79.33	59.09	47.18	79.46	75.06	68.02
	W6A6	OmniQuant*	3.71	5.91	4.81	80.20	60.27	46.84	80.55	76.01	**68.77**
	W6A6	QLLM	3.55	5.76	**4.66**	80.63	59.01	45.99	79.64	75.37	68.13
	W4A8	QLLM	3.6	5.76	4.68	80.79	58.59	47.44	79.42	75.77	68.40
	W4A4	SQ	26.01	34.61	30.31	64.09	41.84	32.00	54.21	51.07	48.64
	W4A4	OS+	-	-	-	66.16	42.72	34.90	56.93	52.96	50.73
	W4A4	OmniQuant*	41.10	54.33	47.72	52.99	31.14	23.89	33.88	52.01	38.78
	W4A4	QLLM	7.00	8.89	**7.95**	74.27	50.59	37.2	71.62	59.43	**58.62**

❑ QLLM significantly outperforms the state-of-the-art post-training quantization (PTQ) methods, demonstrating **a substantial margin of improvement** in 4-bit quantization.

Effect of different components in channel reassembly

Table. Perplexity results of different components in channel reassembly.

CD	CA	CP	Adaptive	γ	LLaMA-1-13B			
					WikiText2	PTB	C4	Avg.
✓				0.00	189.35	539.59	303.45	344.13
✓				0.01	8.31	14.44	10.74	11.16
✓				0.03	8.01	13.52	10.27	10.60
✓				0.05	7.85	13.38	10.13	10.45
✓				0.07	7.81	13.35	10.11	**10.42**
✓	✓			0.01	8.68	15.16	11.12	11.65
✓	✓			0.03	8.72	14.99	11.03	**11.58**
✓	✓			0.05	8.95	15.34	11.29	11.86
✓	✓			0.07	9.39	15.98	11.84	12.40
✓		✓		0.01	8.98	16.34	11.37	**12.23**
✓		✓		0.03	9.51	18.29	12.7	13.50
✓		✓		0.05	9.60	18.11	13.4	13.70
✓		✓		0.07	11.23	21.61	19.79	17.54
✓	✓	-	✓	-	8.41	14.38	10.58	**11.12**

- Notation:
  - CD: channel disassembly
  - CA: channel assembly
  - CP: channel pruning
  - Adaptive: Adaptive strategy
  - γ: channel expansion ratio

- CD makes activations more quantization-friendly by **decomposing the outlier channels**.
- CA retains the original channel count **with a less performance drop** compared to channel pruning.
- Our adaptive strategy is able to find optimal θ with **near-lossless performance**.

Channel Reassembly vs. Other Outlier Handling Methods.

Table. Performance comparisons of our channel reassembly (CR) with previous outlier handling methods.

Model	#Bits	Method	PIQA	ARC-e	ARC-c	HellaSwag	Winogrande	Avg.
LLaMA-1-7B	W6A6	SQ	76.65	53.11	40.10	71.52	61.88	60.65
	W6A6	OS+	76.82	51.35	41.13	71.42	65.98	**61.34**
	W6A6	CR	76.88	52.31	40.87	71.37	64.33	61.15
	W4A4	SQ	49.80	30.40	25.80	27.40	48.00	36.28
	W4A4	OS+	62.73	39.98	30.29	44.39	52.96	46.07
	W4A4	CR	66.92	42.55	32.34	54.31	50.04	**49.23**
LLaMA-1-13B	W6A6	SQ	77.80	56.36	42.58	75.11	68.11	63.99
	W6A6	OS+	78.29	56.90	43.09	75.09	69.22	**64.52**
	W6A6	CR	78.02	56.69	42.41	74.70	70.01	64.37
	W4A4	SQ	55.55	34.51	26.71	41.56	48.70	41.41
	W4A4	OS+	63.00	40.32	30.38	53.61	51.54	47.77
	W4A4	CR	67.57	43.77	31.48	60.78	56.04	**51.93**

- All methods exhibit **comparable performance** at 6-bit quantization.
- Channel reassembly **significantly surpasses** other methods by a large margin at 4-bit quantization.

Effect of Efficient Gradient-based Error Correction

Table. Comparisons between efficient error correction (EEC) and tuning quantized weights directly (TQW) for 4-bit LLaMA-1-65B.

#Attn-FFN Block	Method	WikiText2	PTB	C4	Avg.	Training Time (GPU Hours)	GPU Memory (GB)
1	TQW	6.34	17.61	9.56	11.17	12.16	30.84
1	EEC	8.31	13.77	10.76	10.95	**7.79**	**19.00**
2	TQW	6.25	11.18	8.56	8.66	12.13	52.45
2	EEC	7.62	11.47	9.39	9.49	**7.79**	**28.60**
4	TQW	-	-	-	-	-	OOM
4	EEC	6.87	11.36	8.98	9.07	**7.77**	**47.71**

- Compared with TQW, EEC **significantly reduces training costs** and **GPU memory usage** while delivering comparable performance.
- The reduced GPU memory demand allows EEC to quantize LLaMA-1-65B on **a single 24GB consumer-grade GPU**, such as the NVIDIA RTX 4090.

Inference Efficiency

Table. Inference throughput comparisons using a 2048-token segment on RTX 3090 GPUs: 1x GPU for LLaMA-1-7B and 2x GPUs for LLaMA-1-13B.

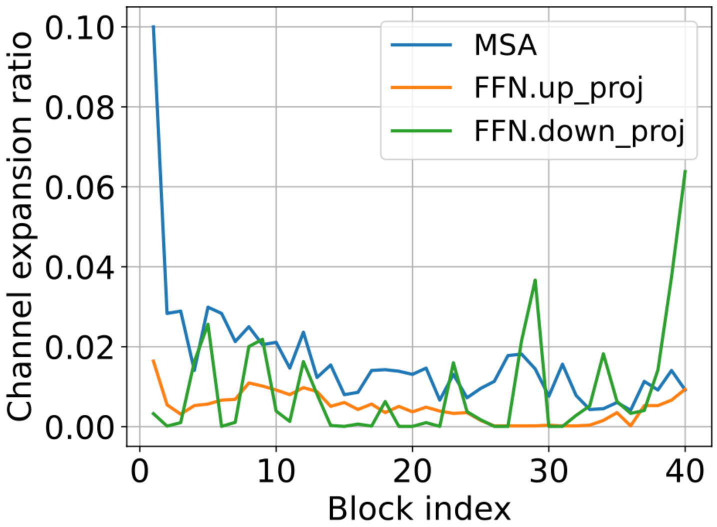
Model	Method	Throughput (tokens/s)
LLaMA-1-7B	FP16	3252
	W8A8	5676
	W4A16	5708
	W4A4	6667
	QLLM	6385
LLaMA-1-13B	FP16	1910
	W8A8	3179
	W4A16	2026
	W4A4	3873
	QLLM	3730

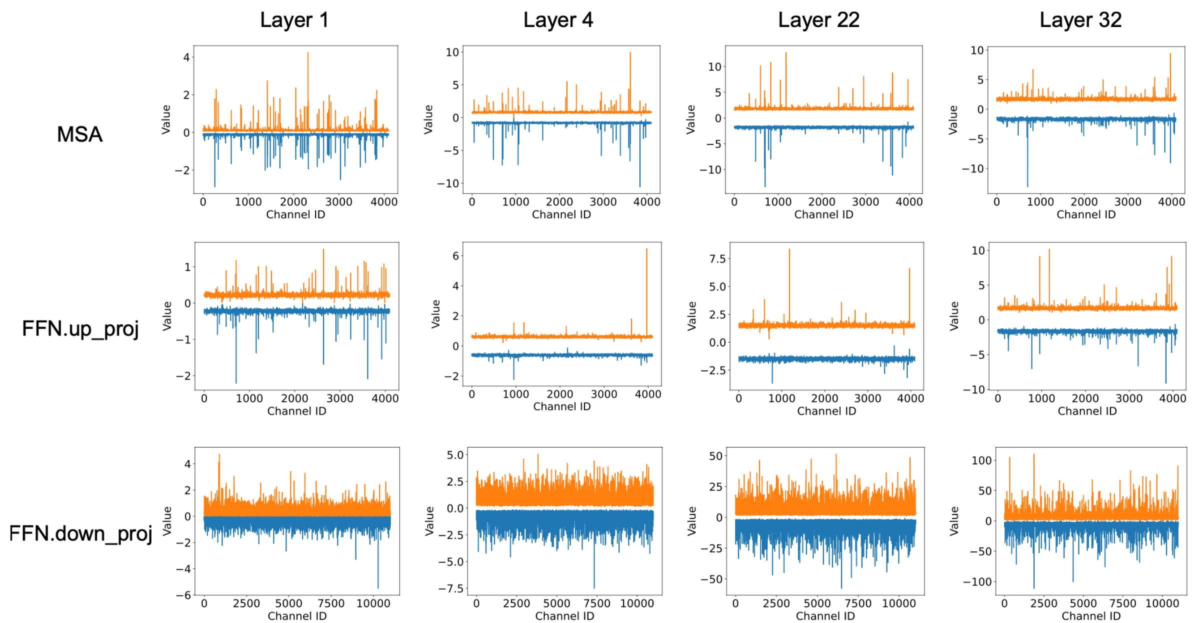
Table. Inference throughput (tokens/s) comparisons of different models. The throughput is measured with a 2048-token segment on a NVIDIA RTX 3090 GPUs

Model	Method	CD	CA	Adaptive	γ	Inference Throughput (tokens/s)
LLaMA-1-7B	FP16				-	3252
	W8A8				-	5676
	W4A16				-	5708
	W4A4				-	6667
	W4A4	✓			0.01	6322
	W4A4	✓			0.05	6315
	W4A4	✓			0.1	6310
	W4A4	✓	✓		0.01	6365
	W4A4	✓	✓		0.05	6334
	W4A4	✓	✓		0.1	6318
	W4A4	✓	✓	✓	-	6385

- With efficient CUDA and Triton kernels, our 4-bit QLLM only incurs **4% additional cost** relative to W4A4 but achieves a notable **1.96× speedup** over FP16.
- Channel disassembly results in **additional costs** due to the extra channels (not a multiples like 32 or 64).
- Channel assembly **maintains original channel count** and **mitigating the extra costs** from disassembly.

The Expansion Ratio Results of 4-bit LLaMA-1-13B





- ❑ Our adaptive strategy allocates **higher expansion ratios to the shallower MSA layers** and to the **deeper down projection layer** in the FFN, which indicates that these layers possess a greater number of outliers.

Liu et al. QLLM – ICLR 2024

Thanks for Watching

Please refer to our paper and code for more details

Paper

Q & A

Code