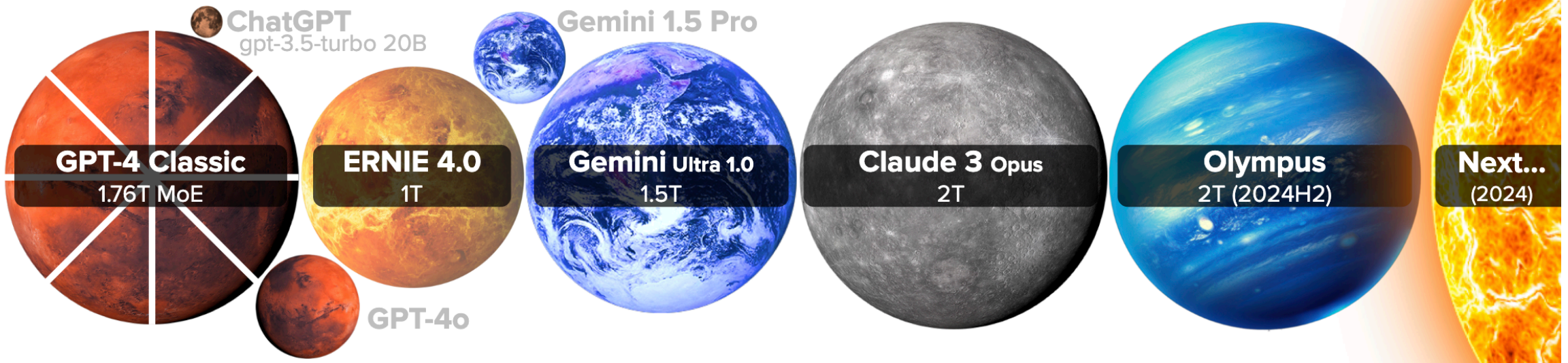
하계세미나

발표자: 임정우

LARGE LANGUAGE MODEL HIGHLIGHTS (JUN/2024)



- **Nano**
  Gemini-Nano-1 1.8B
  Mamba-2 2.7B
  Phi-3-mini 3.8B
- **XS**
  Falcon 2 11B
  Gemma 7B
  Mistral 7B
- 30B **Small**
  Command-R 35B
  Mixtral 8x7B
  Yi 1.5 34B
- 70B **Medium**
  K2-65B
  Llama 3 70B
  Luminous Supreme
- 180B **Large**
  Command R+ 104B
  Qwen-1.5 110B
  Titan 200B
- 300B **XL**
  Grok-1.5 314B
  Inflection-2.5
  Llama 3 405B

Parameters
AI lab/group

+ more than 350 documented models at LifeArchitect.ai/models-table

Sizes linear to scale. Selected highlights only. All 350+ models: https://lifearchitect.ai/models-table/ Alan D. Thompson. 2021-2024.

Preliminaries

- 두 가지의 다른 LLM Quantization Methods: QAT, PTQ
  1. QAT (Quantization-aware training)
     - Quantization 을 위해 전체 모델과 quantization 파라미터를 위해서 fine-tuning 시켜야함

     ex) LLM-QAT

  2. PTQ (Post-training quantization)
     - 전체 학습 후 모델의 파라미터를 quantization. 하지만 성능의 저하가 뚜렷함

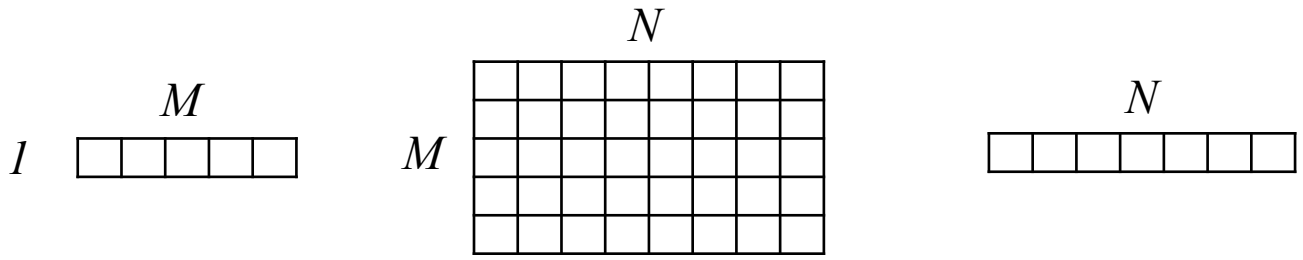
     ex) GPTQ, SmoothQuant, LLM.int8()

Preliminaries

- 파라미터는 어떻게 Quantization을 할까?
  - LLM이 가지고 있는 두 가지 중요 파트 :
    1) MSA (Multi-head Self-Attention)
    2) FFN (Feed-Forward Network)

- 이 구조 안에 들어있는 linear layer 들 내부의 Notation은 다음과 같음

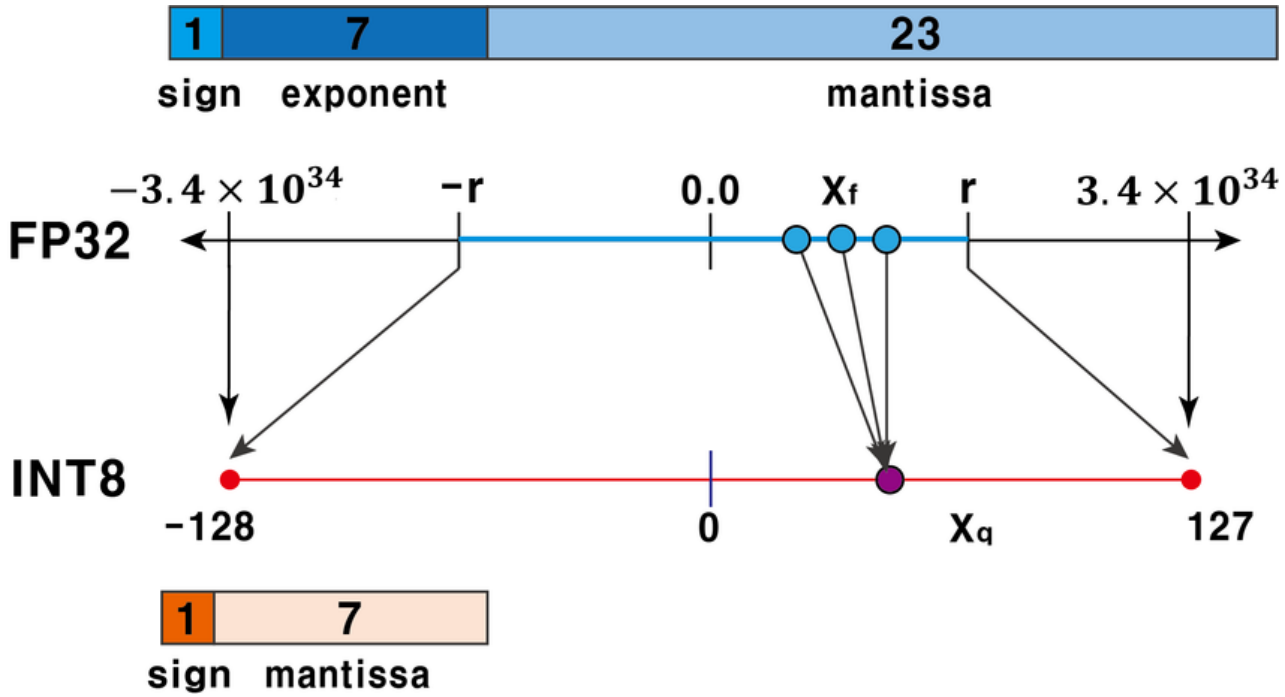
$$\mathbf{y}_k = \sum_{i=1}^{M} \mathbf{x}_i \mathbf{W}_{ik},$$



$\mathbf{x} \in \mathbb{R}^M$ $\mathbf{W} \in \mathbb{R}^{M \times N}$ $\mathbf{y} \in \mathbb{R}^N$

Preliminaries

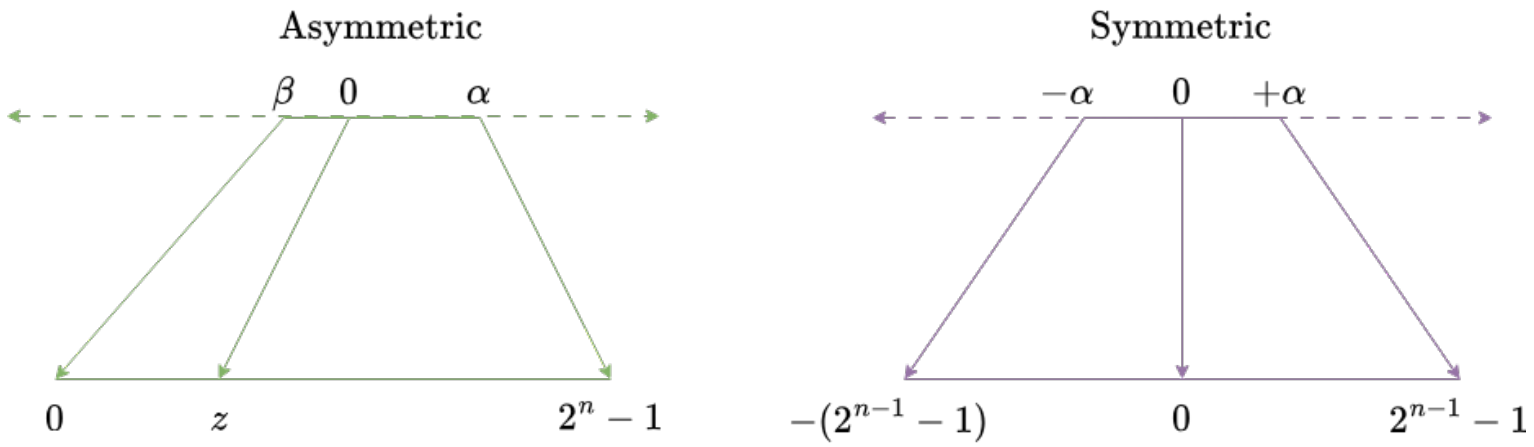
- X, Y, W값 >> FP32, FP16의 Real value 들을 INT8로 Quantization 진행한다면,
  = 수들을 다음의 형태로 표현한다는 것
  = Dequantize 하게 되면 당연히 원래의 값과 차이가 날 수밖에 없음



Preliminaries

- 0의 좌우범위에 따라 Assymmetric, Symmetric으로 나뉨

Original	49.7	-13.14	0	-6.66	48.7	-12.14	-7.41



Asymmetric	255	0	53	26	251	4	23

Symmetric	127	-34	0	-17	124	-31	-19

Preliminaries

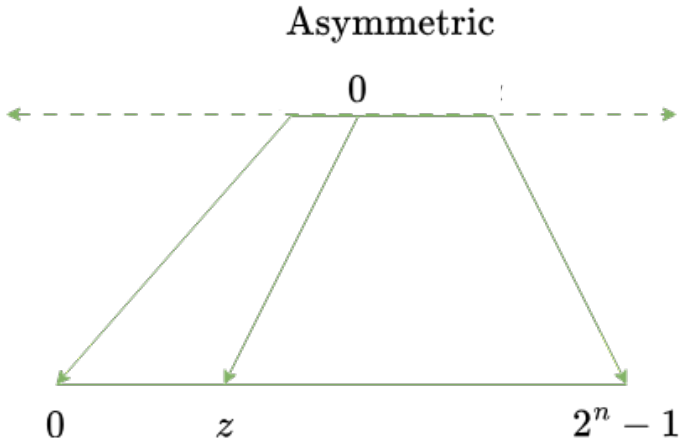
- 이걸 수식적으로 나타내보자

b는 quantization 비트 수
α 는 scale factor
β 는 zero-point value
[] 는 round

$$\alpha = \frac{\max(\mathbf{X}) - \min(\mathbf{X})}{2^b - 1}$$

$$\beta = -\left\lfloor \frac{\min(\mathbf{X})}{\alpha} \right\rceil$$

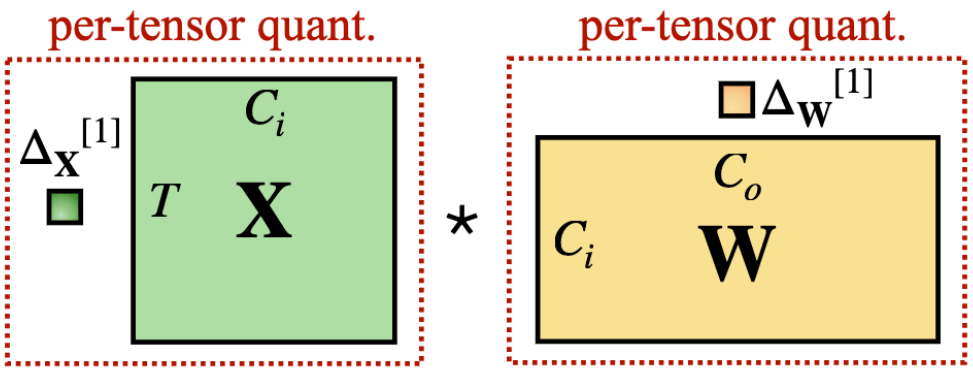
$$\mathbf{X}_q = \text{quant}(\mathbf{X}) = \text{clamp}\left(\lfloor \frac{\mathbf{X}}{\alpha} \rceil + \beta, 0, 2^b - 1\right)$$



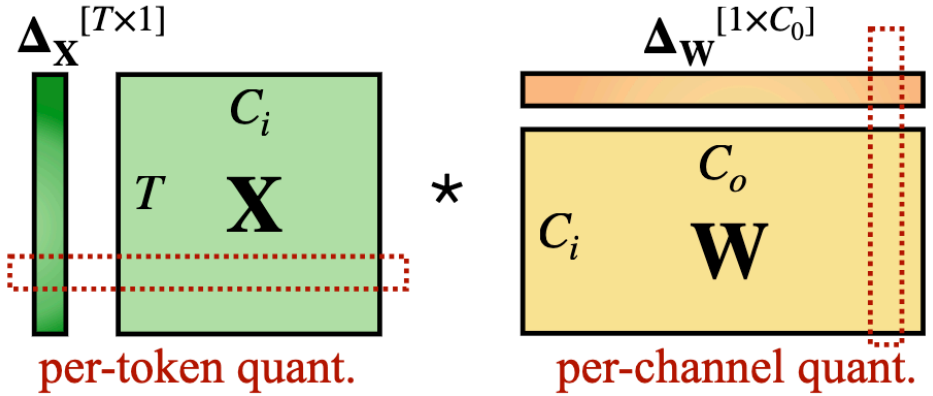
- 그림의 n은 수식의 b
- 그림의 z는 수식의 β

Preliminaries

- 저 min, max를 어떻게 설정하느냐에 따라 ..



(a) per-tensor quantization



(b) per-token + per-channel quantization

OMNIQUANT: OMNIDIRECTIONALLY CALIBRATED QUANTIZATION FOR LARGE LANGUAGE MODELS

Wenqi Shao[†1], **Mengzhao Chen**[†1], **Zhaoyang Zhang**[3], **Peng Xu**[1,2], **Lirui Zhao**[1], **Zhiqian Li**[2], **Kaipeng Zhang**[1], **Peng Gao**[1], **Yu Qiao**[1], **Ping Luo**[*1,2]
[1]OpenGVLab, Shanghai AI Laboratory [2]The University of Hong Kong
[3]The Chinese University of Hong Kong

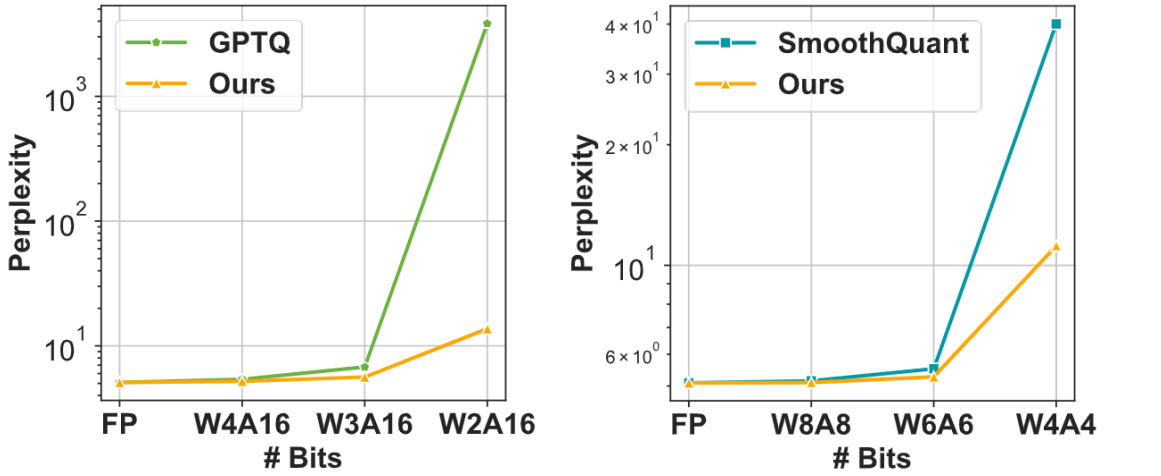
ICLR 2024 (Spotlight)

Introduction

- QAT 는 전체 모델을 quantization 을 위해 training 시키는 것은 너무나 비용이 많이 듬
- 그래서 PTQ가 많이 사용되는데, 그 중에서도 weight만 quantization 시키고, activation은 full-precision으로 두는 경우가 있었음 (GPTQ, SPQR)
    - 일환으로, computational overhead를 줄이기위해 Smoothquant나 OS(Outlier Supression) 등의 모델등은 weight-activation 모두를 quantization하여 low-bit matrix multiplication을 구현함

Introduction

- 이렇게 기존의 quantization 모델들은 W4A16 (i.e. 4-bit weight and 16-bit activation), W8A8 등에서 높은 성능을 보였지만, 더 낮은 비트 수로 quantization하면 성능이 급격하게 떨어짐



(b) weight-only quantization (c) weight-activation quantization

- 이러한 이유는 기존의 method가 handcrafted quantization 파라미터 (migration strength, scaling parameter) 등을 가지고 있어서 그렇다고 함 (optimal하지 않음)

Introduction

- 그렇다고 QAT를 적용하기에는, training과 data efficiency가 확보되지 않음
  (실제로 GPTQ는 A100으로 LLaMA-13B quantization을 128개의 sample로 1시간만에 실행 가능
  하지만 LLM-QAT로는 100k 샘플이 필요하고 100시간 이상이 필요함)

- 저자의 Motivation

  : Can we attain the performance of QAT, while maintaining
  the time and data efficiency of PTQ?

Introduction

- OmniQuant
  : low-bit setting에서 굉장히 좋은 성능을 보였고, time, data efficiency를 동시에 확보하였음
  : OmniQuant 는 기존의 full-precision weight를 얼리고 quantization parameter를 학습시킴

  1) LWC (Learnable Weight Clipping)
  : modulates the extreme values of weights by optimizing the clipping threshold

  2) LET (Learnable Equivalent Transformation)
  : tackles activation outliers by learning mathematically equivalent transformations in a transformer encoder.

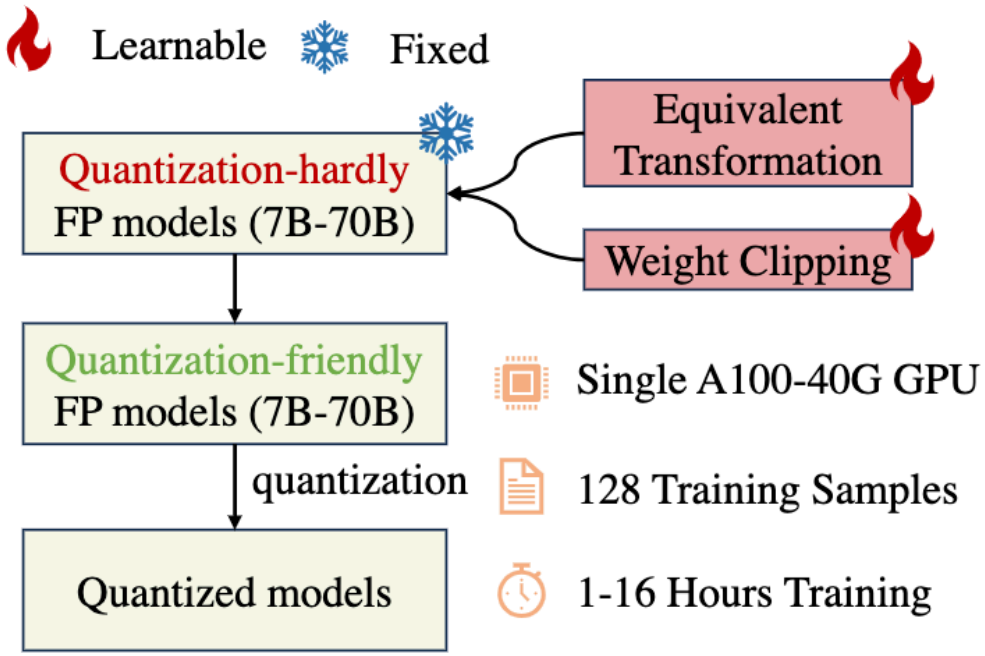


Figure 2: Characteristics of OmniQuant on LLaMA family.

Method

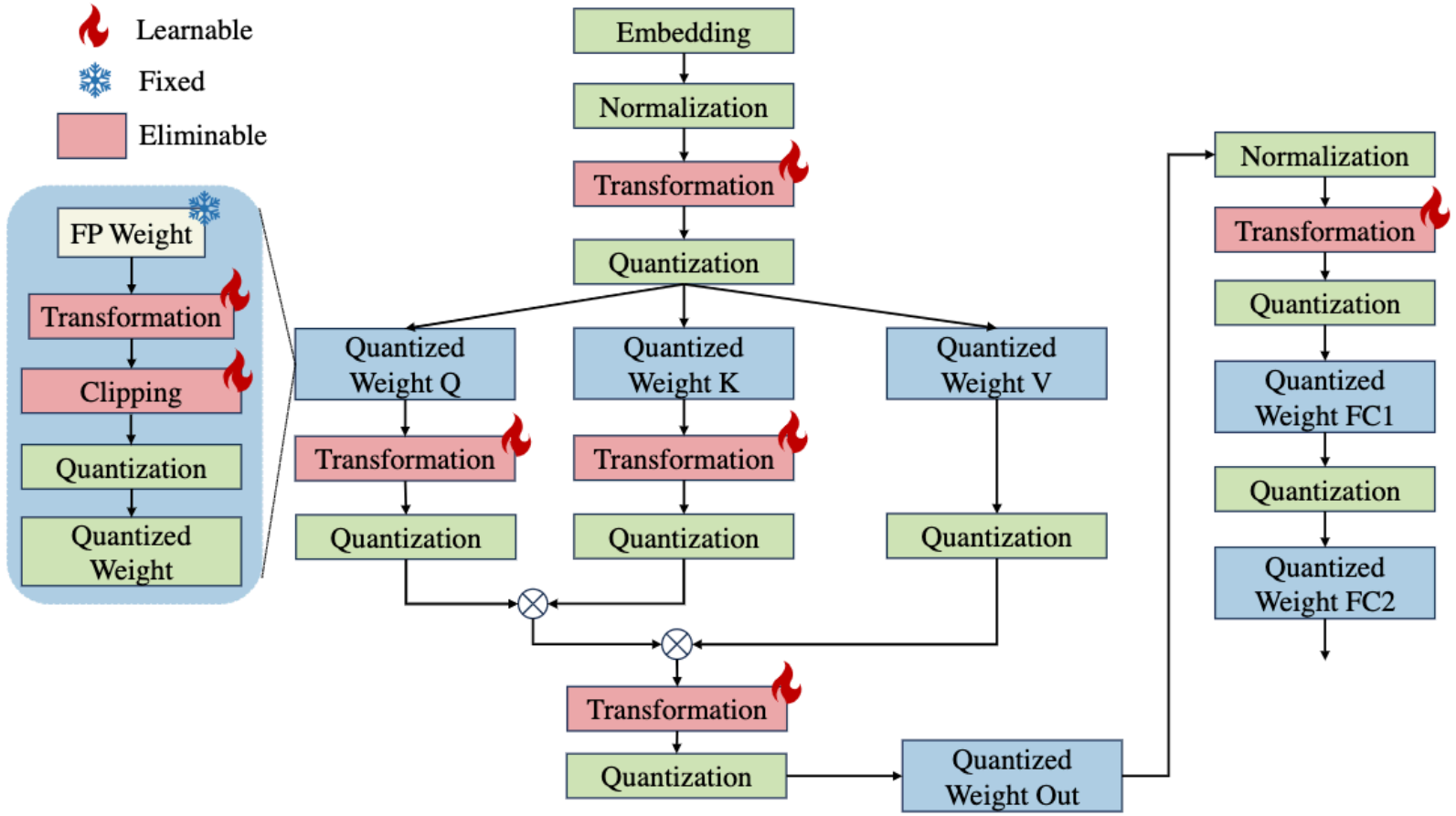


Figure 3: **Details of OmniQuant** in a transformer block. Note that all learnable parameters can be eliminated after quantization.

Method

- Challenge of LLM quantization

  1) Outlier 채널 때문에 activation은 quantization하기 어려움

  2) Activation의 중요도에 따라서 Weight의 quantization error는 performance에 직접적인 영향을

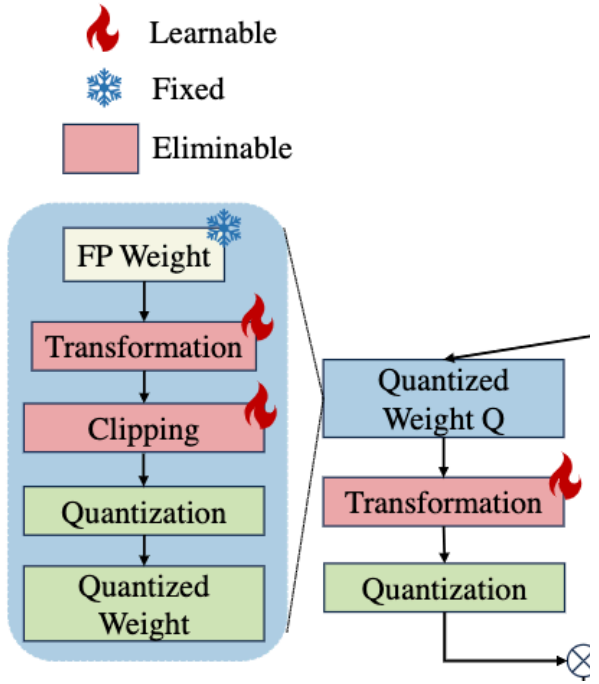
Method

- Challenge of LLM quantization

  1) Outlier 채널 때문에 activation은 quantization하기 어려움

  2) Activation의 중요도에 따라서 Weight의 quantization error는 performance에 직접적인 영향을 줌

- OmniQuant
  a) Block-wise quantization error minimization framework 제안
     - **Learnable weight clipping (LWC)**
       : to mitigate the difficulty in quantizing weights
     - **Learnable equivalent transformation (LET)**
       : shift the challenge of quantization from activations to weights



Method

- Block-wise Quantization Error Minimization

$$\arg\min_{\Theta_1,\Theta_2} ||\mathcal{F}(\mathbf{W},\mathbf{X}) - \mathcal{F}\big(Q_w(\mathbf{W};\Theta_1,\Theta_2), Q_a(\mathbf{X},\Theta_2)\big)||,$$

$\mathcal{F}$: Transformer block

$\mathbf{X}$: activation (FP)

$\mathbf{W}$: weight (FP)

$Q_w(\cdot)$: weight quantizer

$Q_a(\cdot)$: activation quantizer

Θ_1 : LWC 의 파라미터

Θ_2 : LET의 파라미터

1) OmniQuant에서는 LWC와 LET를 동시에 optimizing 하므로 weight-only와 weight-activation quantization을 모두 포괄한다고 함

2) 또한, block-wise minimization 는 quantization 파라미터만 최적화 하기 때문에 minimal resource가 있어도 최적화에 용이함

Method

- Learnable Weight Clipping (LWC)
  : 기존처럼 weight의 clipping threshold를 dynamic하게 정해주는 건 유사하지만, direct하게 파라미터를 튜닝하지 않는다는 점에서 그 차이가 있음

$$h = \frac{\gamma \max(\mathbf{W}) - \beta \min(\mathbf{W})}{2^N - 1}$$

$$z = -\lfloor \frac{\beta \min(\mathbf{W})}{h} \rceil$$

$$\mathbf{W_q} = \text{clamp}(\lfloor \frac{\mathbf{W}}{h} \rceil + z, 0, 2^N - 1)$$

N는 quantization 비트 수
h 는 normalization factor
z 는 zero-point value
[] 는 round
$\alpha, \beta \in [0,1]$ 는 파라미터, learnable clipping strengths

Method

- Learnable Equivalent Transformation (LET) – Linear Layer
  : weight-activation quantization의 어려움을 줄이기 위해 제안됨
  : 특정 채널에서 outlier가 발견되는 점을 고려하여, 기존의 hand-crafted 파라미터는 optimal 하지 않음
  : LET는 channel-wise scaling, channel-wise shifting을 이용하여 activation을 변경하여 outlier를 완화함

$$\mathbf{Y} = \mathbf{XW} + \mathbf{B} = \underbrace{[(\mathbf{X} - \delta) \oslash s]}_{\tilde{\mathbf{X}}} \cdot \underbrace{[s \odot \mathbf{W}]}_{\tilde{\mathbf{W}}} + \underbrace{[\mathbf{B} + \delta\mathbf{W}]}_{\tilde{\mathbf{B}}}$$

T: sequence length

$\mathbf{X} \in \mathbb{R}^{T \times C_{in}}$: input

$\mathbf{W} \in \mathbb{R}^{C_{in} \times C_{out}}$

$\delta \in \mathbb{R}^{1 \times C_{in}}$: channel-wise shifting 파라미터

$\mathbf{s} \in \mathbb{R}^{1 \times C_{in}}$: channel-wise scaling 파라미터

$\oslash$: elementwise division

$\odot$: elementwise multiplication

Method

- Learnable Equivalent Transformation (LET) – Linear Layer
  : weight-activation quantization의 어려움을 줄이기 위해 제안됨
  : 특정 채널에서 outlier가 발견되는 점을 고려하여, 기존의 hand-crafted 파라미터는 optimal 하지 않음
  : LET는 channel-wise scaling, channel-wise shifting을 이용하여 activation을 변경하여 outlier를 완화함

$$\mathbf{Y} = Q_a(\tilde{\mathbf{X}})Q_w(\tilde{\mathbf{W}}) + \tilde{\mathbf{B}}$$

$Q_a(\cdot)$: vanilla MinMax quantizer

$Q_w(\cdot)$: vanilla MinMax quantizer w/ LWC

: 이러한 transformation을 FFN의 두번째 레이어만 빼고 적용함
(왜냐하면 non-linear layer 다음의 high sparsity 때문에 gradient가 unstable할 수 있기 때문)

Method

- Learnable Equivalent Transformation (LET) – Attention operation
  : Attention에서는 엄청난 양의 computation이 필요하기에, 저자들은 Q/K/V matrix도 low-bit로 만들고자 함 (weight-activation setting)

$$\mathbf{P} = \text{Softmax}(\mathbf{Q}\mathbf{K}^T) = \text{Softmax}((\underbrace{\mathbf{Q} \oslash s_a}_{\tilde{\mathbf{Q}}})(\underbrace{s_a \odot \mathbf{K}^T}_{\tilde{\mathbf{K}}^T}))$$

$s_a \in \mathbb{R}^{1 \times C_{out}}$

$\tilde{\mathbf{Q}}$ $\tilde{\mathbf{K}}$ 는 Q_a 와 같이 MinMax Quantizer를 통과한 값

Method

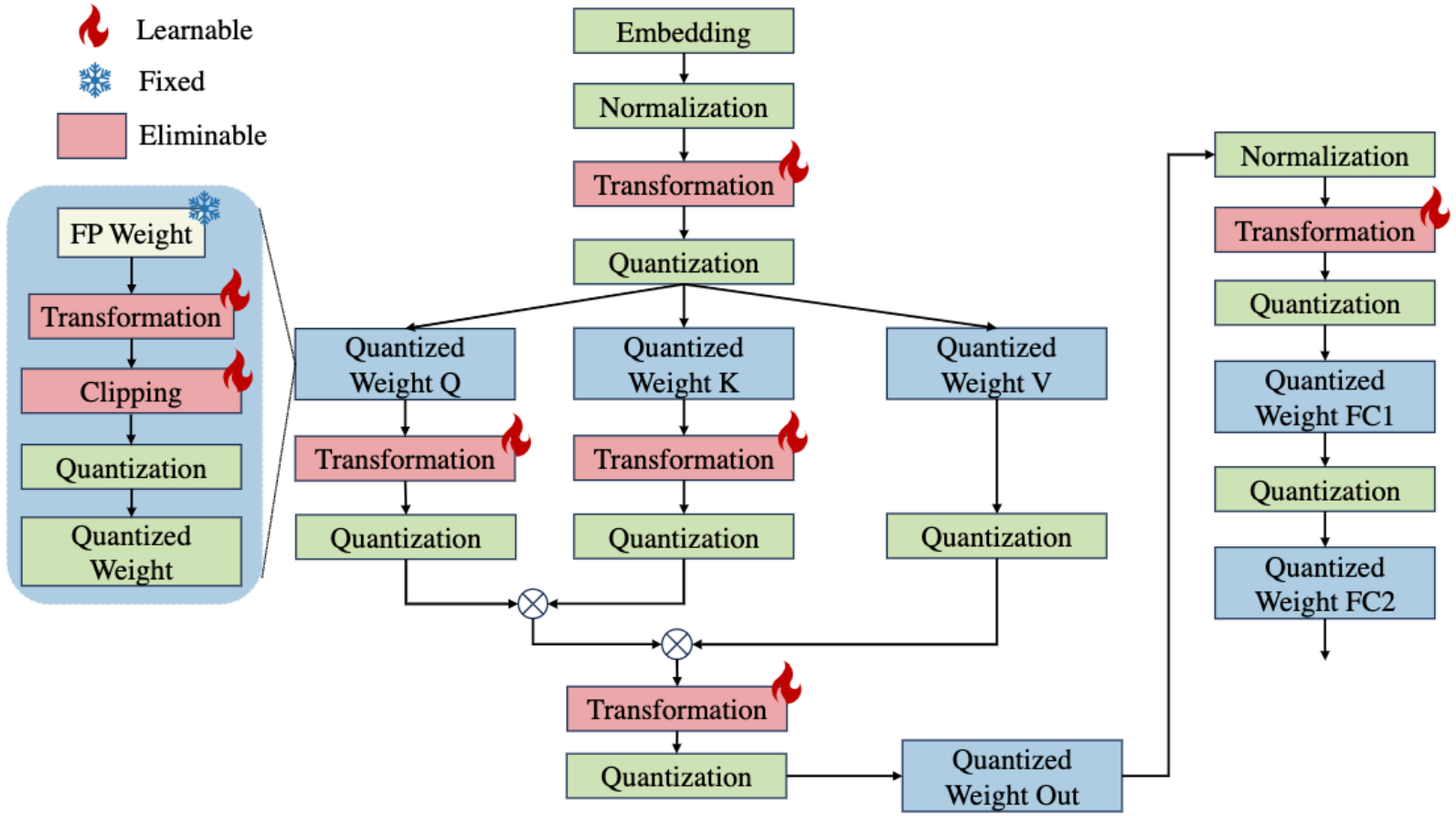


Figure 3: **Details of OmniQuant** in a transformer block. Note that all learnable parameters can be eliminated after quantization.

Experiments

Table 1: **Weight-only quantization Results of LLaMA-1 and LLaMA-2 Models**. We report WikiText2 perplexity in this table, C4 perplexity can be found in Table A19 in Appendix.

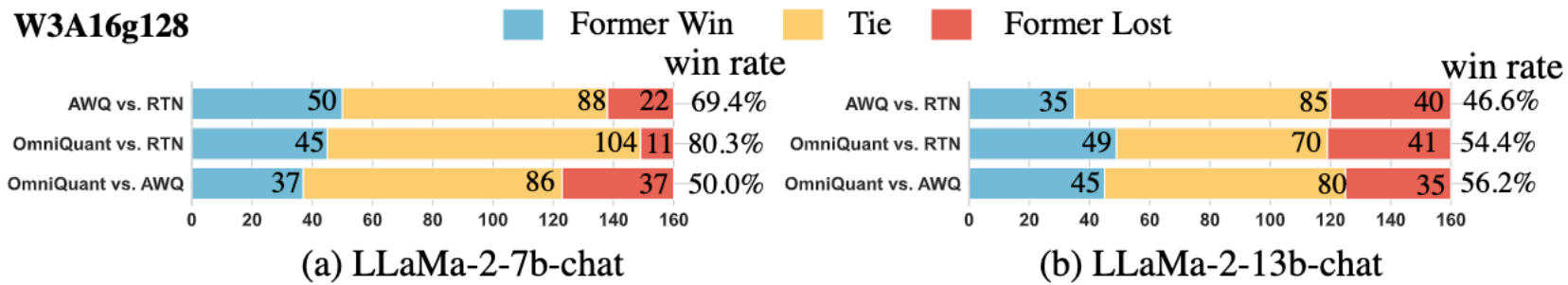
LLaMA1&2 / PPL↓		1-7B	1-13B	1-30B	1-65B	2-7B	2-13B	2-70B
FP16	-	5.68	5.09	4.10	3.53	5.47	4.88	3.31
W2A16	RTN	1.1e5	6.8e4	2.4e4	2.2e4	3.8e4	5.6e4	2.0e4
	GPTQ	2.1e3	5.5e3	499.75	55.91	7.7e3	2.1e3	77.95
	OmniQuant	**15.47**	**13.21**	**8.71**	**7.58**	**37.37**	**17.21**	**7.81**
W2A16 g128	RTN	1.9e3	781.20	68.04	15.08	4.2e3	122.08	27.27
	GPTQ	44.01	15.60	10.92	9.51	36.77	28.14	NAN
	AWQ	2.6e5	2.8e5	2.4e5	7.4e4	2.2e5	1.2e5	-
	OmniQuant	**9.72**	**7.93**	**7.12**	**5.95**	**11.06**	**8.26**	**6.55**
W2A16 g64	RTN	188.32	101.87	19.20	9.39	431.97	26.22	10.31
	GPTQ	22.10	10.06	8.54	8.31	20.85	22.44	NAN
	AWQ	2.5e5	2.7e5	2.3e5	7.4e4	2.1e5	1.2e5	-
	OmniQuant	**8.90**	**7.34**	**6.59**	**5.65**	**9.62**	**7.56**	**6.11**
W3A16	RTN	25.73	11.39	14.95	10.68	539.48	10.68	7.52
	GPTQ	8.06	6.76	5.84	5.06	8.37	6.44	4.82
	AWQ	11.88	7.45	10.07	5.21	24.00	10.45	-
	OmniQuant	**6.49**	**5.68**	**4.74**	**4.04**	**6.58**	**5.58**	**3.92**
W3A16 g128	RTN	7.01	5.88	4.87	4.24	6.66	5.51	3.97
	GPTQ	6.55	5.62	4.80	4.17	6.29	5.42	3.85
	AWQ	6.46	5.51	4.63	3.99	6.24	5.32	-
	OmniQuant	**6.15**	**5.44**	**4.56**	**3.94**	**6.03**	**5.28**	**3.78**
W4A16	RTN	6.43	5.55	4.57	3.87	6.11	5.20	3.67
	GPTQ	6.13	5.40	4.48	3.83	5.83	5.13	3.58
	AWQ	6.08	5.34	4.39	3.76	6.15	5.12	-
	OmniQuant	**5.86**	**5.21**	**4.25**	**3.71**	**5.74**	**5.02**	**3.47**
W4A16 g128	RTN	5.96	5.25	4.23	3.67	5.72	4.98	3.46
	GPTQ	5.85	5.20	4.23	3.65	5.61	4.98	3.42
	AWQ	5.81	5.20	4.21	3.62	5.62	4.97	-
	OmniQuant	**5.77**	**5.17**	**4.19**	**3.62**	**5.58**	**4.95**	**3.40**

Experiments

Table 2: **Weight-activation quantization results of LLaMA Models.** This table reports the accuracy of 6 zero-shot tasks. Perplexity results can be found in Table A23 & A24 at Appendix.

LLaMA / Acc↑	#Bits	Method	PIQA	ARC-e	Arc-c	BoolQ	HellaSwag	Winogrande	Avg.
LLaMA-1-7B	FP16	-	77.47	52.48	41.46	73.08	73.00	67.07	64.09
	W6A6	SmoothQuant	76.75	51.64	39.88	71.75	71.67	65.03	62.81
	W6A6	OS+	76.82	51.35	41.13	72.08	71.42	65.98	61.13
	W6A6	**OmniQuant**	77.09	51.89	40.87	72.53	71.61	65.03	**63.17**
	W4A4	SmoothQuant	49.80	30.40	25.80	49.10	27.40	48.00	38.41
	W4A4	LLM-QAT	51.50	27.90	23.90	61.30	31.10	51.90	41.27
	W4A4	LLM-QAT+SQ	55.90	35.50	26.40	62.40	47.80	50.60	46.43
	W4A4	OS+	62.73	39.98	30.29	60.21	44.39	52.96	48.43
	W4A4	**OmniQuant**	66.15	45.20	31.14	63.51	56.44	53.43	**52.65**
LLaMA-1-13B	FP16	-	79.10	59.89	44.45	68.01	76.21	70.31	66.33
	W6A6	SmoothQuant	77.91	56.60	42.40	64.95	75.36	69.36	64.43
	W6A6	OS+	78.29	56.90	43.09	66.98	75.09	69.22	64.92
	W6A6	**OmniQuant**	78.40	57.28	42.91	67.00	75.82	68.27	**64.95**
	W4A4	SmoothQuant	61.04	39.18	30.80	61.80	52.29	51.06	49.36
	W4A4	OS+	63.00	40.32	30.38	60.34	53.61	51.54	49.86
	W4A4	**OmniQuant**	69.69	47.39	33.10	62.84	58.96	55.80	**54.37**
LLaMA-1-30B	FP16	-	80.08	58.92	45.47	68.44	79.21	72.53	67.44
	W6A6	SmoothQuant	77.14	57.61	42.91	65.56	78.07	69.92	65.20
	W6A6	OS+	80.14	58.92	45.05	68.02	77.96	71.98	67.01
	W6A6	**OmniQuant**	79.81	58.79	45.22	68.38	78.95	72.21	**67.23**
	W4A4	SmoothQuant	58.65	35.53	27.73	60.42	35.56	48.06	44.83
	W4A4	OS+	67.63	46.17	34.40	60.70	54.32	52.64	52.62
	W4A4	**OmniQuant**	71.21	49.45	34.47	65.33	64.65	59.19	**56.63**
LLaMA-1-65B	FP16	-	80.79	58.71	46.24	82.29	80.72	77.50	71.04
	W6A6	SmoothQuant	80.25	57.92	45.50	80.22	80.18	74.76	69.80
	W6A6	OS+	79.67	55.68	45.22	80.02	78.03	73.95	68.76
	W6A6	**OmniQuant**	81.01	58.12	46.33	80.64	79.91	75.69	**70.28**
	W4A4	SmoothQuant	64.47	40.44	29.82	59.38	39.90	52.24	47.71
	W4A4	OS+	68.06	43.98	35.32	62.75	50.73	54.30	52.52
	W4A4	**OmniQuant**	71.81	48.02	35.92	73.27	66.81	59.51	**59.22**

Experiments



(a) LLaMa-2-7b-chat (b) LLaMa-2-13b-chat

Figure 4: Comparing W3A16g128 quantization among RTN, AWQ (Lin et al., 2023), and OmniQuant under Vicuna-Bench (Chiang et al., 2023). Win rates are calculated without considering tie samples. A higher win rate indicates the better performance of the former of *vs.* pairs.

Experiments

Table 3: Deployment of weight-only quantization through MLC-LLM. We report the memory size of quantized weights (denoted as 'WM') and the running memory (denoted as 'RM') and speed in NVIDIA A100-80G.

LLaMA	**7B**			**13B**			**30B**			**65B**		
	WM	RM	token/s	WM	RM	token/s	WM	RM	token/s	WM	RM	token/s
FP	12.6G	14.4G	69.2	24.3G	27.1G	52.5	60.6G	66.1G	23.9	OOM	-	-
W4A16g128	3.8G	5.7G	134.2	7.0G	10.0G	91.3	16.7G	21.7G	43.6	33.0G	41.0G	24.3
W3A16g128	3.2G	5.1G	83.4	5.8G	8.7G	57.6	13.7G	18.7G	29.0	27.0G	35.1G	15.2
W2A16g128	2.2G	4.1G	83.9	4.0G	7.5G	92.6	9.2G	14.1G	36.7	18.0G	25.6G	24.8

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

ICLR 2024

Introduction

- 기존 연구에서는 LLM의 activation 쪽에서 매우 큰값을 가지는 특정한 outlier channel 이 있다는 것을 보였음
- 이는 outlier들이 quantization range를 매우 크게 만들기 때문에, 정확한 숫자를 나타내는 데에 오류를 많이 포함시키고, 결국엔 모델의 성능 저하를 야기함

- 이를 해결하기 위해, 다양한 연구들이 activation 쪽의 outlier 크기를 weight로 전이하거나 하는 노력들을 하였음 (e.g. SmoothQuant)

Introduction

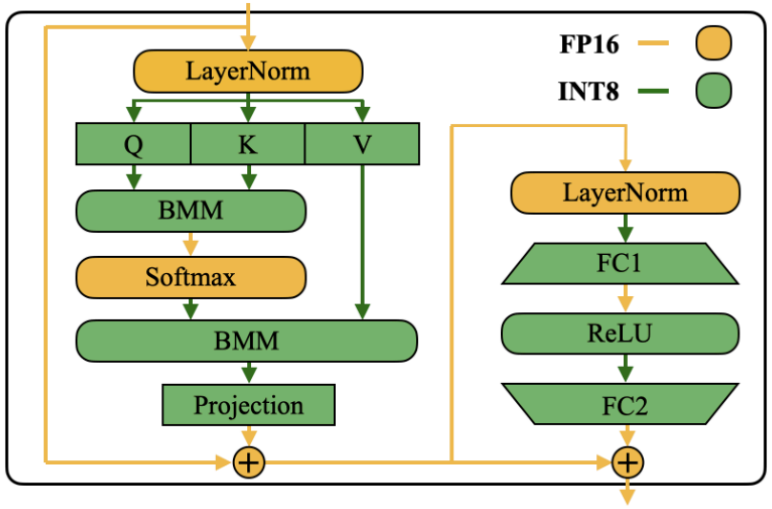
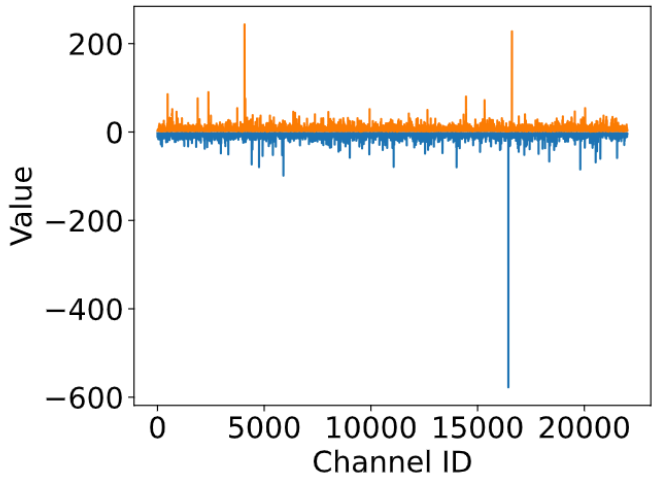
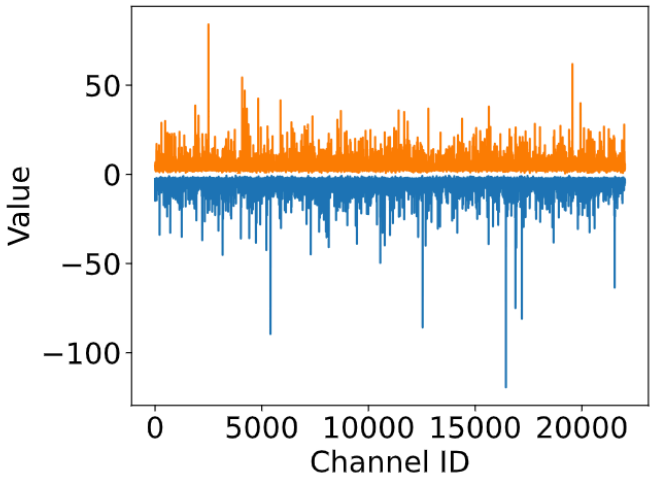


Figure 6: SmoothQuant's precision mapping for a Transformer block. All compute-intensive operators like linear layers and batched matmul (BMMs) use INT8 arithmetic.



(a) Pre-trained model



(b) SmoothQuant

- 하지만, FFN 내의 activation내의 outlier들이 이 unstable한 gradient를 만들게 되고, 결국 성능이 엄청나게 떨어지게 됨
- 이 논문에서는, QLLM이라는 low-bit width post-training quantization을 제안한다고 함
  > outlier들을 gradient-free channel reassembly 테크닉으로 큰 크기의 activation value를 channel들 전체에 재배치하여 이를 해결하고자 함

Proposed Method

- ✓ Adaptive channel reassembly framework 제안
  : to redistribute input activation outliers across multiple channels

  1) Channel disassembly
     : for decomposing the outlier channel

  2) Channel assembly
     : for balancing the efficiency

  3) Adaptive strategy
     : to find the suitable reassembly ratio for each layer

  Channel reassembly technique은 gradient-free이며, 효율적이라는 장점이 있음

Proposed Method

1) Channel Disassembly
: input의 outlier channel들을 분해해서 sub-channel들로 만들기

Outlier들이 특정 채널들에게만 있으니까, 이러한 채널들을 sub-channel들로 만들어서 큰 value들을 분산시키는 원리 !

M번째 채널이 outlier 채널이라 하면, 다른 채널들 계산 완료된 것에 x를 T로 나누고 M번째 계산을 T번 더해줌

$$\mathbf{y}_k = \sum_{i=1}^{M-1} \mathbf{x}_i \mathbf{W}_{ik} + \underbrace{\frac{\mathbf{x}_M}{T}\mathbf{W}_{Mk} + \cdots + \frac{\mathbf{x}_M}{T}\mathbf{W}_{Mk}}_{T \text{ times}}. \qquad T = \lceil \max(|\mathbf{x}_M|)/\theta \rceil$$

* θ는 outlier threshold라고 해서 하이퍼 파라미터

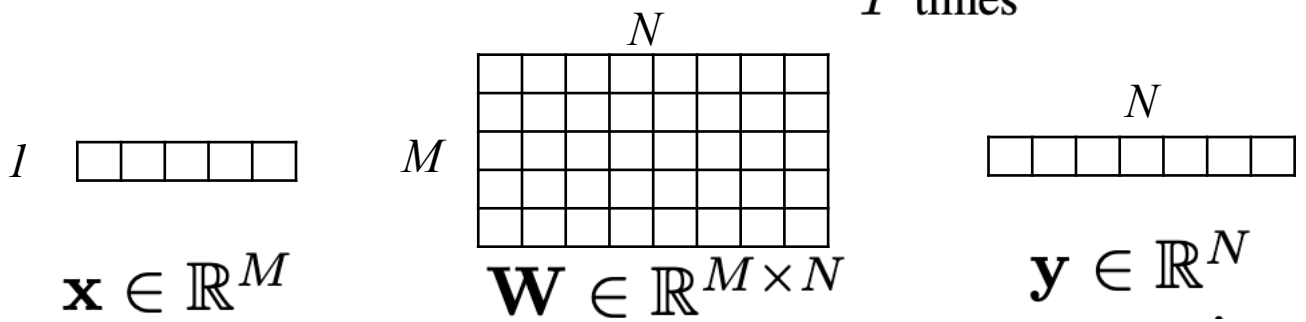
Proposed Method

1) Channel Disassembly
: input의 outlier channel들을 분해해서 sub-channel들로 만들기

Outlier들이 특정 채널들에게만 있으니까, 이러한 채널들을 sub-channel들로 만들어서 큰 value들을 분산시키는 원리 !

M번째 채널이 outlier 채널이라 하면, 다른 채널들 계산 완료된 것에 x를 T로 나누고 M번째 계산을 T번 더해줌

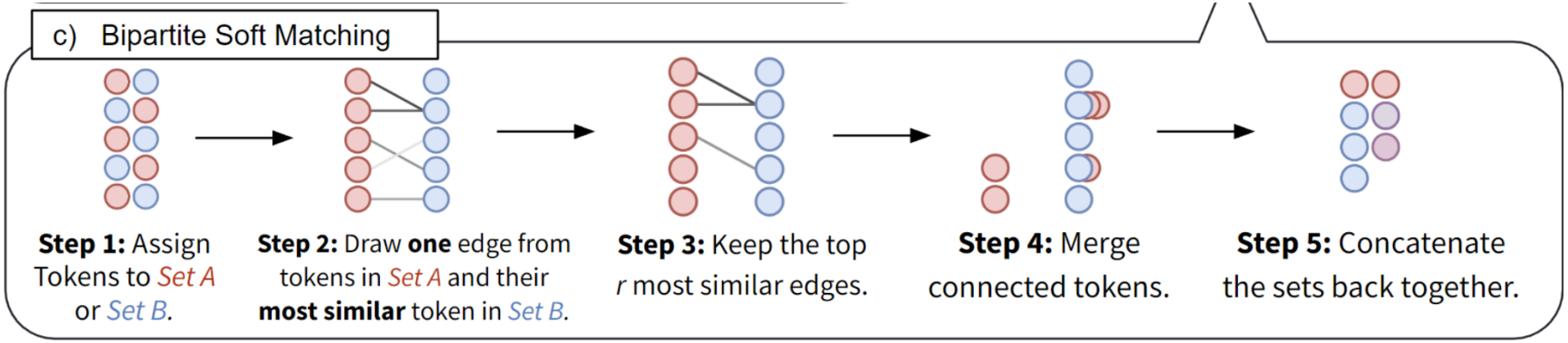
$$\mathbf{y}_k = \sum_{i=1}^{M-1} \mathbf{x}_i \mathbf{W}_{ik} + \underbrace{\frac{\mathbf{x}_M}{T}\mathbf{W}_{Mk} + \cdots + \frac{\mathbf{x}_M}{T}\mathbf{W}_{Mk}}_{T \text{ times}}.$$



Proposed Method

2) Channel Assembly
: input 채널이 개수 M+T-1개로 늘어났기에 중요하지 않은 채널은 지우거나 비슷한 채널은 합친 후 채널 개수를 동일하게 맞추고자 함
(Channel Pruning + Chanel Merging)

* token Merging이라고하는 아이디어 차용



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

Proposed Method

2) Channel Assembly
: input 채널이 개수 M+T-1개로 늘어났기에 중요하지 않은 채널은 지우거나 비슷한 채널은 합친 후 채널 개수를 동일하게 맞추고자 함
(Channel Pruning + Chanel Merging)

i, j 채널 merging은 다음과 같이 진행됨

$$\mathbf{x}_i\mathbf{W}_{ik} + \mathbf{x}_j\mathbf{W}_{jk} \approx \frac{\mathbf{x}_i+\mathbf{x}_j}{2}\left(\mathbf{W}_{ik} + \mathbf{W}_{jk}\right)$$

* 이때, outlier 채널과 먼 거리의 channel은 merge 하지 않음

Proposed Method

3) Adaptive Reassembly
: 각 레이어마다 위의 Reassembly 비율을 얼마나 해야할지 결정함

- Channel Disassembly:

$$T = \lceil \max(|\mathbf{x}_M|)/\theta \rceil$$

- high value for T with a small θ 일 경우:
  a) substantially reduces outlier magnitudes and benefits quantization
  b) resulting in a larger increase in channel merging error due to a higher merging ratio

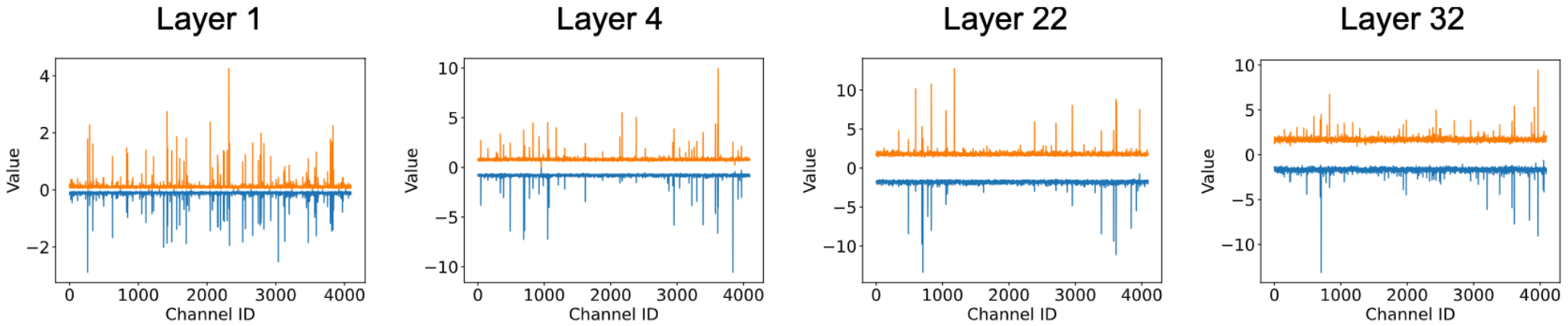
- small T with a large θ 일 경우:
  a) will not increase the channel count much making it easier for the assembly stage to keep the information
  b) likely still retaining outliers, causing significant quantization errors

Proposed Method

3) Adaptive Reassembly
: 각 레이어마다 위의 Reassembly 비율을 얼마나 해야할지 결정함

- 하지만 그걸 만족하는 θ 를 찾기는 어려움. 왜냐하면 각 레이어마다 다른 패턴을 가지고 있기 때문

MSA



Proposed Method

3) Adaptive Reassembly
: 각 레이어마다 위의 Reassembly 비율을 얼마나 해야할지 결정함
각 레이어의 original output activations과 reassembled input activation 에러를 가장 줄이는 θ를 고르는 문제

$\hat{\mathbf{X}} \in \mathbb{R}^{L \times M}$: Reassembled activation

$\boldsymbol{L}$: Reassembled activation

$$\tilde{\mathbf{Q}} = \text{quant}(\hat{\mathbf{X}})\text{quant}(\hat{\mathbf{W}}_Q)$$
$$\tilde{\mathbf{K}} = \text{quant}(\hat{\mathbf{X}})\text{quant}(\hat{\mathbf{W}}_K)$$
$$\tilde{\mathbf{V}} = \text{quant}(\hat{\mathbf{X}})\text{quant}(\hat{\mathbf{W}}_V)$$

$$\arg\min_\theta \left\| \text{Softmax}(\mathbf{Q}\mathbf{K}^\top)\mathbf{V} - \text{Softmax}(\tilde{\mathbf{Q}}\tilde{\mathbf{K}}^\top)\hat{\mathbf{V}} \right\|_F^2,$$

Proposed Method

✓ Efficient Gradient-Based Error Correction
: 이전 방법이 gradient-free였다면, 이제 gradient-based는 작은 calibration set을 이용해서 튜닝하는 법을 말함. 이때 학습시키는 파라미터는 Low-rank 로 구성하고 기존 W는 freeze

$$\mathbf{Y} = \text{quant}(\mathbf{X})\text{quant}(\mathbf{W}) + \text{quant}(\mathbf{X})\mathbf{AB}$$

$$\mathbf{A} \in \mathbb{R}^{M \times r}$$

$$\mathbf{B} \in \mathbb{R}^{r \times N}$$

- low-rank 파라미터는 reconstruction error (original과 quantized output)을 줄이는 목적으로 학습되고 Attention-FFN 블럭들을 자동으로 조정하게됨

$\text{quant}(\mathbf{W} + \mathbf{AB})$ 이 값이 결국에 보존되는 값!

Experiments

- ✓ Model Setup
  : LLaMA 1, LLaMA2 (Zero-shot)

- ✓ Quantization Setup
  : per-channel weight quantization + per-token activation quantization

- ✓ Calibration Set
  : 128 random samples (from WikiText2)

Experiments

Table 1: Performance comparisons of different methods for weights and activations quantization on LLaMA-1 model family. PPL denotes the perplexity.

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-1-13B	W16A16	-	5.09	6.61	5.85	79.05	59.84	44.62	76.22	70.09	65.96
	W6A6	SQ	5.50	7.03	6.27	77.80	56.36	42.58	75.11	68.11	63.99
	W6A6	OS+	5.37	-	-	78.29	56.90	43.09	75.09	69.22	64.52
	W6A6	OmniQuant	5.28	6.84	6.06	78.40	57.28	42.91	75.82	68.27	**64.54**
	W6A6	QLLM	5.28	6.82	**6.05**	77.91	57.70	42.92	75.02	69.14	**64.54**
	W4A8	QLLM	5.33	6.91	6.12	78.29	57.03	42.75	74.46	68.35	64.18
	W4A4	SQ	79.35	120.24	99.80	55.55	34.51	26.71	41.56	48.70	41.41
	W4A4	OS+	53.64	-	-	63.00	40.32	30.38	53.61	51.54	47.77
	W4A4	OmniQuant	10.87	13.78	12.33	69.69	47.39	33.10	58.96	55.80	52.99
	W4A4	QLLM	8.41	10.58	**9.50**	71.38	47.60	34.30	63.70	59.43	**55.28**

Experiments

LLaMA-1-30B	W16A16	-	4.10	5.98	5.04	80.09	58.92	45.39	79.21	72.77	67.28
	W6A6	SQ	5.37	-	-	77.14	57.61	42.91	78.07	69.92	65.13
	W6A6	OS+	4.48	-	-	80.14	58.92	45.05	77.96	71.98	66.81
	W6A6	OmniQuant	4.38	6.22	5.30	79.81	58.79	45.22	78.95	72.21	**67.00**
	W6A6	QLLM	4.30	6.17	**5.24**	79.65	58.08	44.11	78.38	73.24	66.69
	W4A8	QLLM	4.40	6.22	5.31	79.11	57.87	44.62	78.03	72.22	66.37
	W4A4	SQ	399.65	245.87	322.76	50.16	28.11	26.71	31.97	51.14	37.62
	W4A4	OS+	112.33	-	-	67.63	46.17	34.30	54.32	52.64	51.01
	W4A4	OmniQuant	10.33	12.49	11.41	71.21	49.45	34.47	64.65	59.19	55.79
	W4A4	QLLM	8.37	11.51	**9.94**	73.83	50.67	38.40	67.91	58.56	**57.87**
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**

Experiments

Table 2: Perplexity results of different components in channel reassembly. "CD" stands for channel disassembly. "CA" represents channel assembly. "CP" indicates channel pruning. "Adaptive" refers to the adaptive strategy. "γ" is the channel expansion ratio.

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**

Experiments

Table 4: Comparisons between efficient error correction (EEC) and tuning quantized weights directly (TQW) for 4-bit LLaMA-1-65B. "OOM" indicates out of memory.

#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**

Thanks
Q & A