Quantization for LLM Inference and Finetuning

Bingrui Li

2023.06.09

Outline

- Quantization Basis
- Quantization for LLM Inference
  - LLM.int8(), SmoothQuant, GPTQ, AWQ
- Quantization for LLM Finetuning
  - QLoRA

Background: Quantization

- Quantization: real2int
- Quantization = Normalization + Mapping
  - Normalization: a real number => a number in unit interval
  - Mapping: a number in unit interval => a int number
- … only keep quantized tensor …
- Dequantization: reverse the quantization, int2real

Normalization

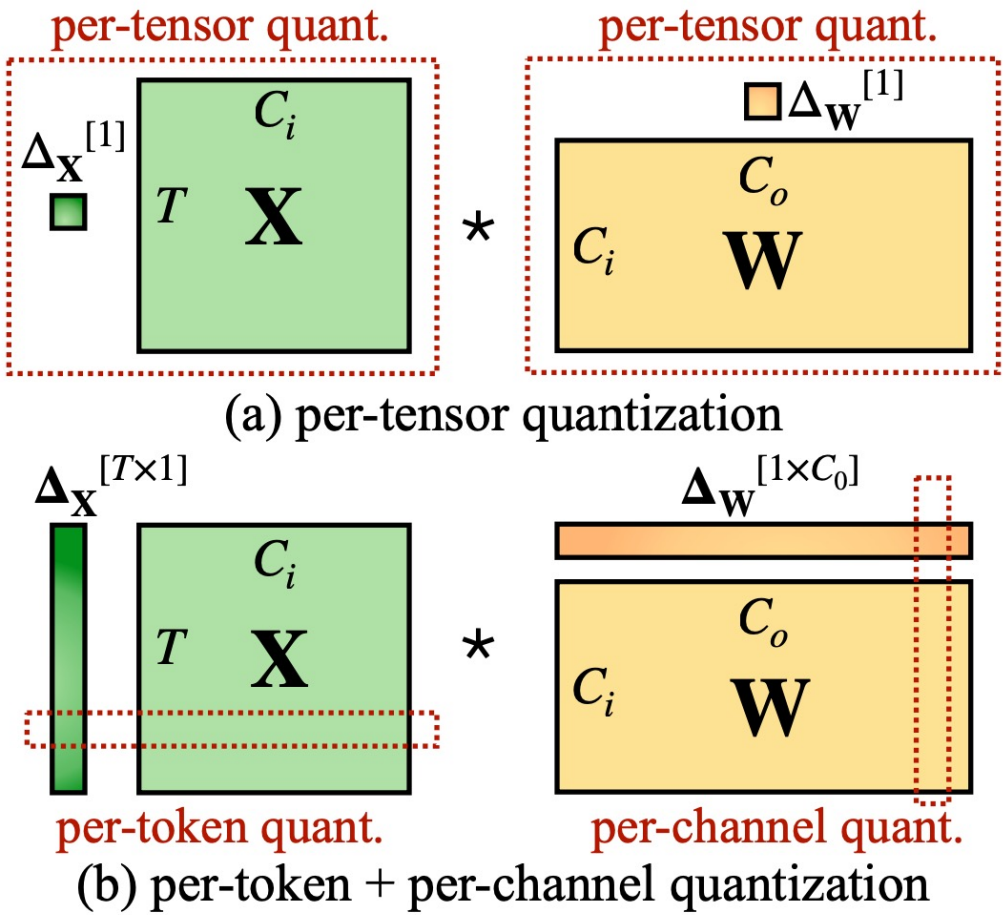
Normalization The normalization operator $\mathbf{N}$ scales each elements of $\mathbf{x}$ into the unit interval, i.e. $[0, 1]$. Normalization can have different granularity, such as per-tensor, per-token (row) [33, 51], per-channel (column) [4], group-wise [33, 51] and block-wise [14]. The per-tensor and block-wise normalization operators are given by

$$n_j := \mathbf{N}_{\text{per-tensor}}(x_j) = x_j / \max_{1 \le i \le p} |x_i| \,,$$

$$n_j := \mathbf{N}_{\text{block-wise}}(x_j) = x_j / \max \left\{ |x_i| : 1 + B \lfloor j/B \rfloor \le i \le B \left(\lfloor j/B \rfloor + 1 \right) \right\},$$

respectively, where the involved scaling factors are called *quantization scale*, which are persistently stored together with quantized tensor until dequantization. The granularity of normalization presents a trade-off of quantization error and memory overhead. Normalization method with low quantization

Per-token and per-channel norm.



Mapping/data type: from linear to nonlinear

Mapping A mapping [14] converts normalized quantities to low-bitwidth integers. Formally, the mapping operator $\mathbf{M} = \mathbf{M}_{\mathbf{T},b}$ is equipped with a bitwidth b and a predefined increasing mapping, named *quantization mapping* $\mathbf{T} : [0, 2^b - 1] \cap \mathbb{Z} \to [0, 1]$. Then $\mathbf{M}$ is defined as

$$q_j := \mathbf{M}(n_j) = \arg \min_{0 \le i < 2^b} |n_j - \mathbf{T}(i)| .$$

The design of $\mathbf{T}$ is critical as it could effectively mitigate quantization error by capturing the distribution information of $\mathbf{n}$. There are two kinds mappings that are of specific interest to optimizer

- Linear (Integer) quantization mapping: T(i) = i / (2^b - 1)
  - Traditional def.: $q = round(n \times (2^b - 1))$
- For signed setting:
  - more flexible in mapping design
  - incompatible with low-bit computation

Quantization Setting

- W8A8
- Weight-only or W4A16

- Difference:
  - choice of mapping
  - computation mechanism

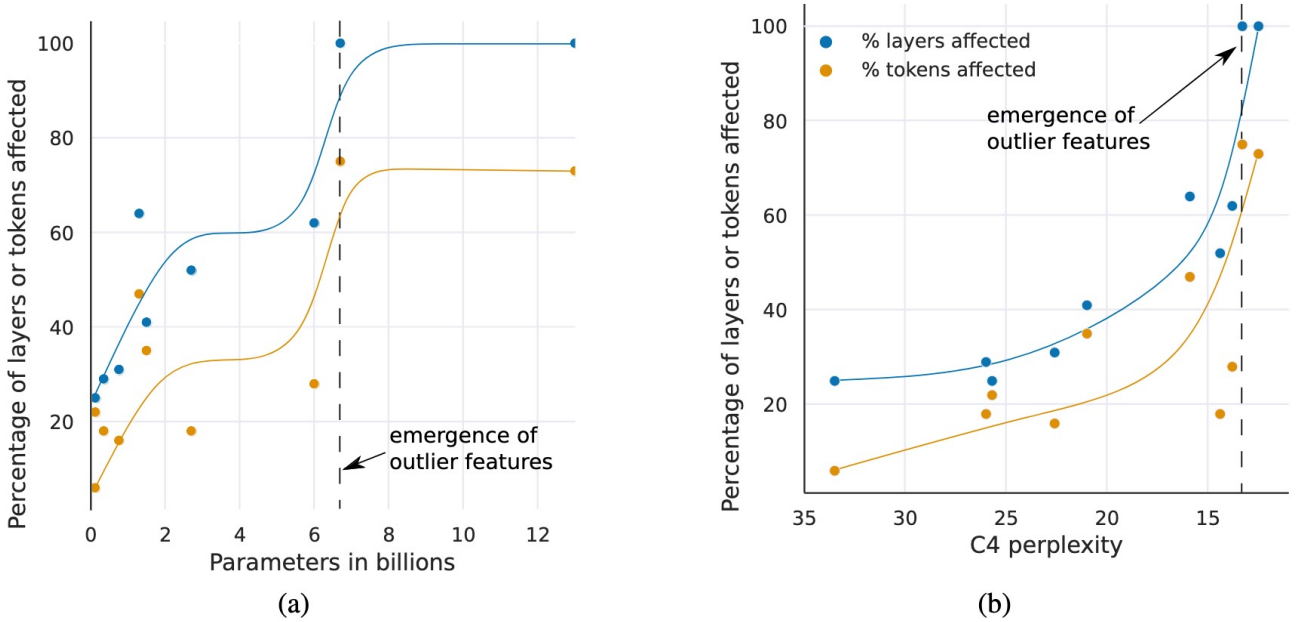
Philosophy of Quantization

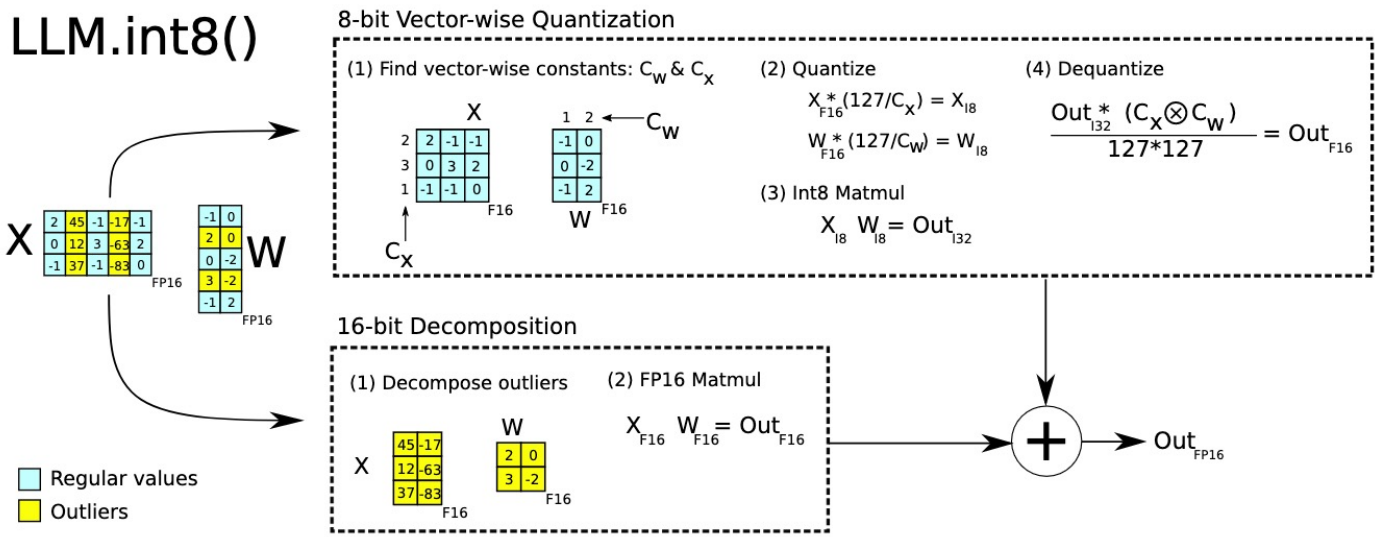
Informally, consider a tensor x to be quantized:

- Large entries has lower relative quantization error
- When there are some outlier (extremely large) entries, small entries will only use low effective quantization bits

LLM.int8() (W8A8)

- Empirical observation:
  - Activations have large magnitude features (outliers) and these outliers occur systematically for almost all sequence/token dimensions s but are limited to specific feature/hidden/channel dimensions h.
    - Criteria: the magnitude of the feature is at least 6.0





- Technique:
  - Per-token norm. for X and per-channel norm. for W
  - Integer data type
  - Mixed-precision decomposition:
    - Use a threshold α to determine the outlier channel dimension.
    - The number of outlier feature dimensions is not larger than 7.

$$\mathbf{C}_{f16} \approx \sum_{h \in O} \mathbf{X}^h_{f16} \mathbf{W}^h_{f16} + \mathbf{S}_{f16} \cdot \sum_{h \notin O} \mathbf{X}^h_{i8} \mathbf{W}^h_{i8}$$

SmoothQuant (W8A8)

Model size (OPT-)	6.7B	13B	30B	66B	175B
FP16	64.9%	65.6%	67.9%	69.5%	71.6%
INT8 per-tensor	39.9%	33.0%	32.8%	33.1%	32.3%
INT8 per-token	42.5%	33.0%	33.1%	32.9%	31.7%
INT8 per-channel	64.8%	65.6%	68.0%	69.4%	71.4%

- claim: Outliers persist in fixed channels

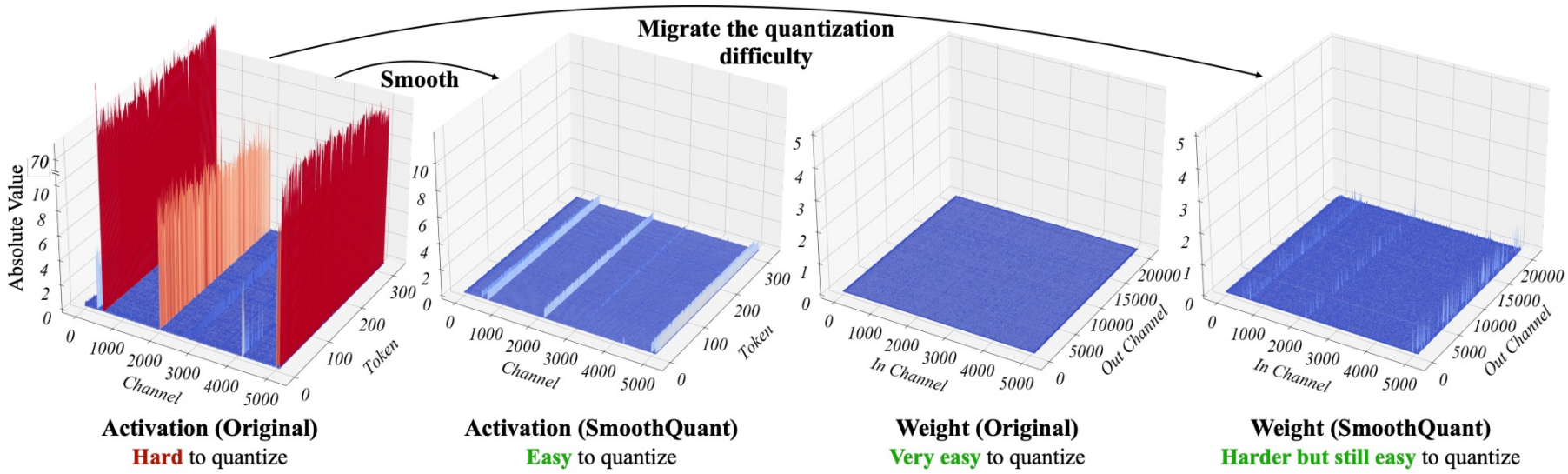


Figure 3: Magnitude of the input activations and weights of a linear layer in OPT-13B before and after SmoothQuant. Observations: (1) there are a few channels in the original activation map whose magnitudes are very large (greater than 70); (2) the variance in one activation channel is small; (3) the original weight distribution is flat and uniform. SmoothQuant migrates the outlier channels from activation to weight. In the end, the outliers in the activation are greatly smoothed while the weight is still pretty smooth and flat.

- Limitation of W8A8

$$\mathbf{Y} = \text{diag}(\boldsymbol{\Delta}_{\mathbf{X}}^{\text{FP16}}) \cdot (\bar{\mathbf{X}}^{\text{INT8}} \cdot \bar{\mathbf{W}}^{\text{INT8}}) \cdot \text{diag}(\boldsymbol{\Delta}_{\mathbf{W}}^{\text{FP16}}) \quad (2)$$

- SmoothQuant

$$\mathbf{Y} = (\mathbf{X}\text{diag}(\mathbf{s})^{-1}) \cdot (\text{diag}(\mathbf{s})\mathbf{W}) = \hat{\mathbf{X}}\hat{\mathbf{W}}$$

  - Set $s_j = \max(|X_j|)$ to ``completely'' migrate the quantization difficulty from activations to weights. In practice, use

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

The role of α

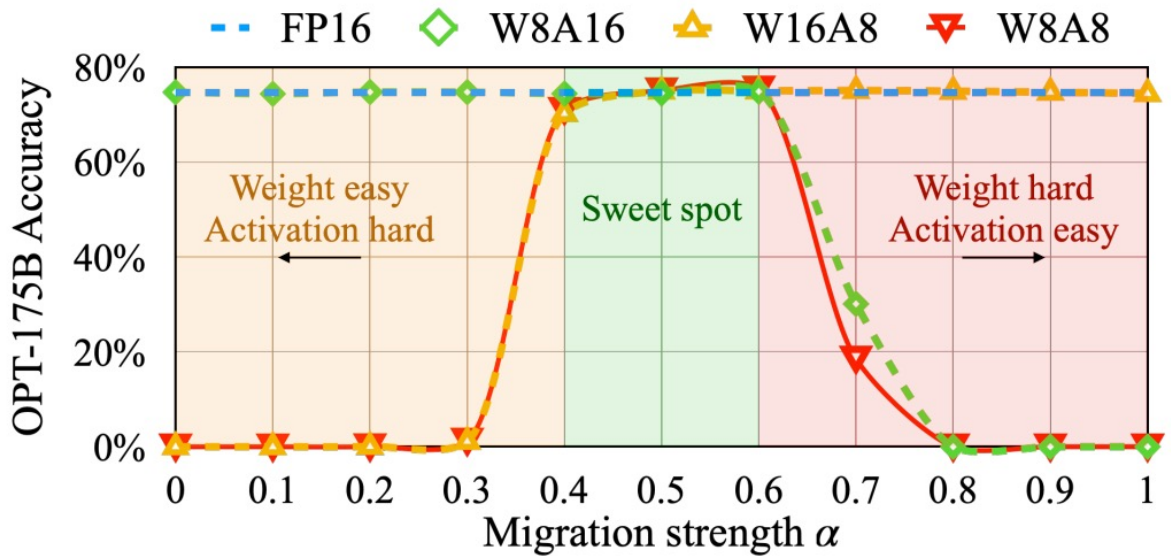
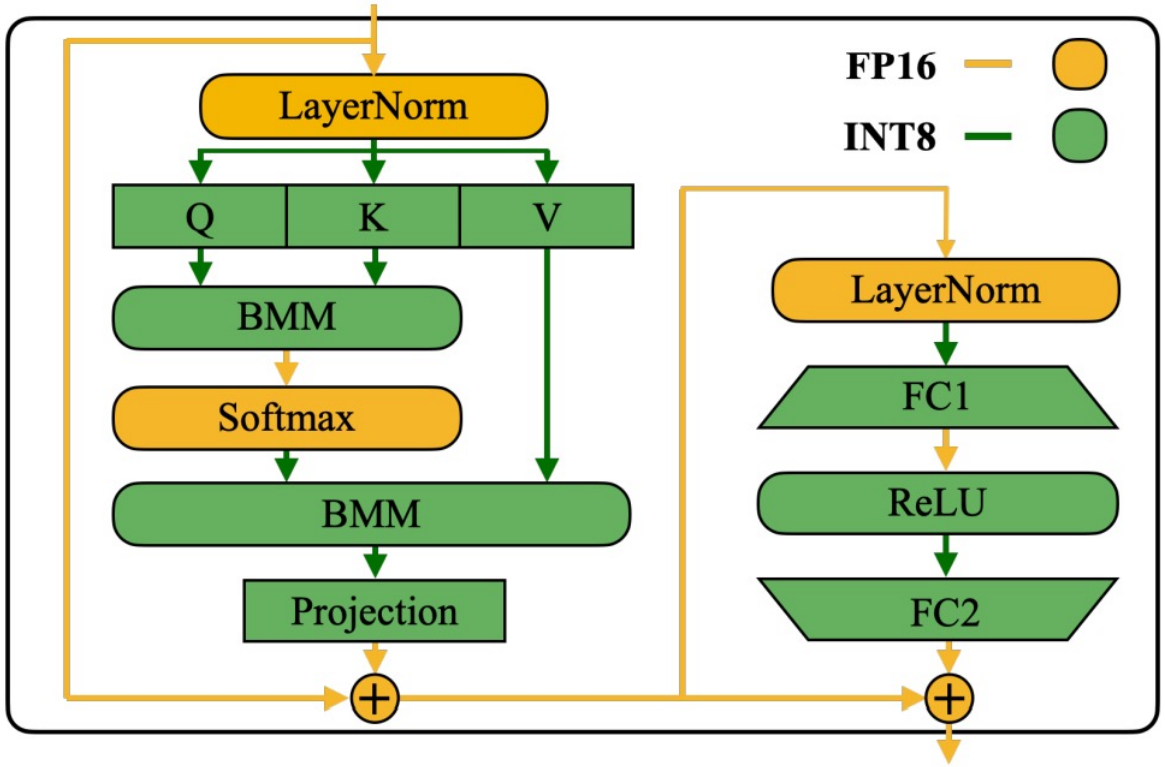


Figure 9: A suitable migration strength α (sweet spot) makes both activations and weights easy to quantize. If the α is too large, weights will be hard to quantize; if too small, activations will be hard to quantize.

Memorize this figure.
Compare with AWQ.

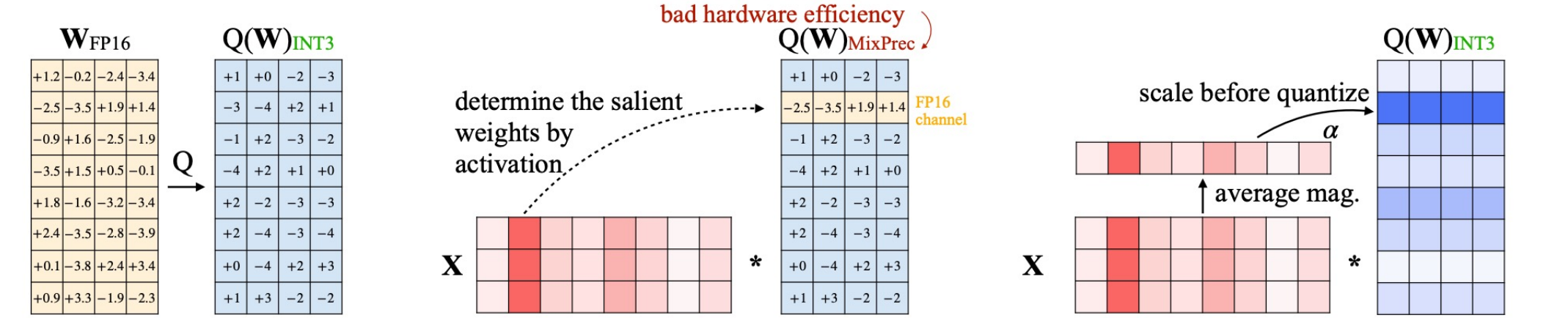
Efficient Implementation (?)



- Reported GPU usage: 1 GPU for OPT-66B, 4 GPUs for OPT-175B

AWQ: Activation-aware Weight Quantization (W4A16)

- claim: Weights of LLMs are not equally important: there is a small fraction of salient weights that are much more important for LLMs' performance compared to others.
- salient= ? (weight magnitude or activation magnitude or …)



(a) RTN quantization (**PPL 43.2**) (b) Keep 1% salient weights in FP16 (**PPL 13.0**) (c) Scale the weights before quantization (**PPL 13.0**)

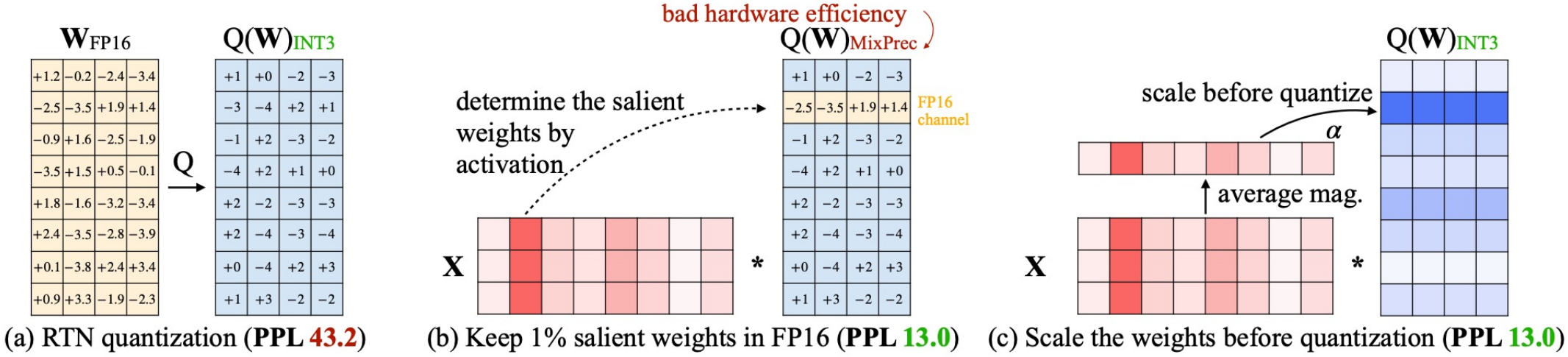
Salient = weight magnitude or activation magnitude ?

PPL ↓	FP16	RTN (w3-g128)	FP16% (based on act.)			FP16% (based on W)			FP16% (random)		
			0.1%	1%	3%	0.1%	1%	3%	0.1%	1%	3%
OPT-1.3B	16.41	206.5	28.00	18.51	18.30	187.1	173.1	165.5	211.2	201.4	88.44
OPT-6.7B	12.29	43.16	13.14	13.02	12.97	43.51	38.59	39.78	42.73	37.83	46.49
OPT-13B	11.50	45.37	12.14	12.04	12.00	46.21	48.07	54.38	45.95	44.47	40.01

Table 1. Keeping a small fraction of weights (0.1%-1%) in FP16 significantly improves the performance of the quantized models over round-to-nearest (RTN). It is only effective when we select the important weights in FP16 by looking at *activation* distribution instead of *weight* distribution. We highlight results with a decent perplexity in green. We used INT3 quantization with a group size of 128 and measured the WikiText perplexity (↓).

Activation matters. How can we improve quantization by leveraging this observation? --- scaling. why?

Replace keeping fp16 with scaled quantization



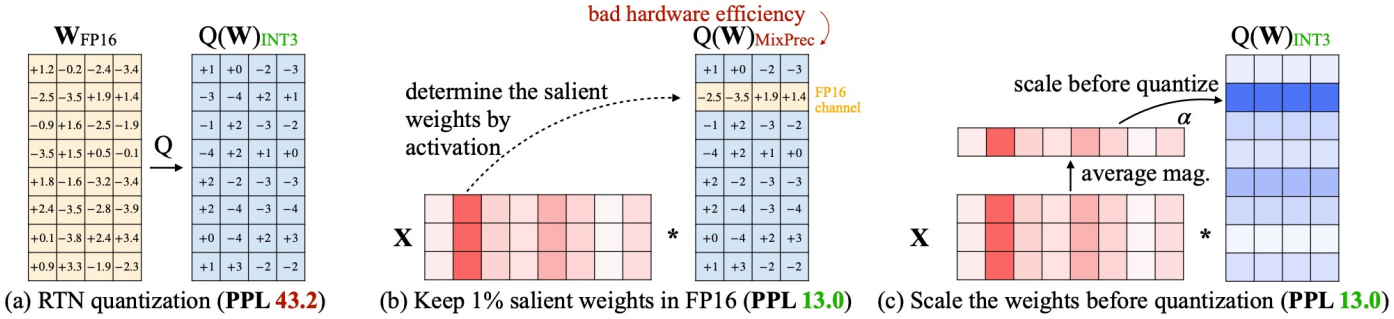
(a) RTN quantization (**PPL 43.2**) (b) Keep 1% salient weights in FP16 (**PPL 13.0**) (c) Scale the weights before quantization (**PPL 13.0**)

Determine scale

Searching to scale. We automatically search for an optimal (per input channel) scaling factor that minimizes the output difference after quantization for a certain layer. Formally, we want to optimize the following objective:

$$\mathbf{s}^* = \arg\min_{\mathbf{s}} \mathcal{L}(\mathbf{s}), \quad \mathcal{L}(\mathbf{s}) = \|Q(\mathbf{W} \cdot \mathbf{s})(\mathbf{s}^{-1} \cdot \mathbf{X}) - \mathbf{W}\mathbf{X}\| \tag{1}$$

Determine scale



(a) RTN quantization (**PPL** 43.2) (b) Keep 1% salient weights in FP16 (**PPL** 13.0) (c) Scale the weights before quantization (**PPL** 13.0)

- $s_X = mean_{Co}|X|$
- computed via calibration set (from pretrained dataset)
- $s_W = mean_{Ci}|W|$ (1.normed. 2. contribute a little. 3. ?)

$$\mathbf{s} = f(\mathbf{s_X}, \mathbf{s_W}) = \mathbf{s_X}^{\alpha} \cdot \mathbf{s_W}^{-\beta}, \quad \alpha^*, \beta^* = \arg\min_{\alpha,\beta} \mathcal{L}(\mathbf{s_X}^{\alpha} \cdot \mathbf{s_W}^{-\beta})$$

- last trick: shrinking ratio / clip

crucial. Furthermore, we find adjusting the clipping range by searching a shrinking ratio (denoted as "+clip") can sometimes further improve the quantized performance. Adjusting the clipping range leads to a nudged scaling factor [15] which may help better protect the salient weights. We combine both scaling and clipping to form AWQ.

- grid search

Evaluation results

LLaMA Faimily		MMLU (5-shot) average ↑				Common Sense QA (0-shot) average ↑			
		7B	13B	30B	65B	7B	13B	30B	65B
FP16	-	38.41%	45.21%	56.84%	60.50%	67.30%	70.65%	72.97%	74.49%
INT3 g128	RTN	33.43%	39.20%	50.58%	57.77%	64.55%	68.63%	72.07%	72.58%
	GPTQ	30.53%	40.90%	52.32%	58.04%	59.66%	68.71%	70.77%	73.03%
	AWQ	**35.43%**	**41.84%**	**53.22%**	**58.83%**	**65.53%**	**69.22%**	**72.10%**	**73.39%**

COCO (CIDEr ↑)		0-shot	4-shot	8-shot	16-shot	32-shot	Δ*(32-shot)*
FP16	-	63.73	72.18	76.95	79.74	81.70	-
INT4 g128	RTN	60.24	68.07	72.46	74.09	77.13	-4.57
	GPTQ	59.72	67.68	72.53	74.98	74.98	-6.72
	AWQ	**62.57**	**71.02**	**74.75**	**78.23**	**80.53**	**-1.17**
INT3 g128	RTN	46.07	55.13	60.46	63.21	64.79	-16.91
	GPTQ	29.84	50.77	56.55	60.54	64.77	-16.93
	AWQ	**56.33**	**64.73**	**68.79**	**72.86**	**74.47**	**-7.23**

Table 6. Quantization results of a visual language model OpenFlamingo-9B [2] on COCO Captioning datasets. AWQ outperforms existing methods under zero-shot and various few-shot settings, demonstrating the generability to different modalities and in-context learning workloads. AWQ reduces the quantization degradation (32-shot) from 4.57 to 1.17 under INT4-g128, providing 4× model size reduction with negligible performance loss.

Comparison with SmoothQuant

- 1. the intuition and mechanism are different. the setting is different.
  - SmoothQuant: balance the smoothness between weight and activation
  - AWQ: scale salient weights with activation information
- 2. the final formulation are similar.
  - $Y = (X \cdot s^{-1}) \cdot Q(s \cdot W)$
  - SmoothQuant: $s = max_{Co}|X|$, + quantize X
  - AWQ: $s = mean_{Co}|X|$
- 3. remarks:
  - difference between max and mean
    - results and quant.
  - reduce the approximation error of output.

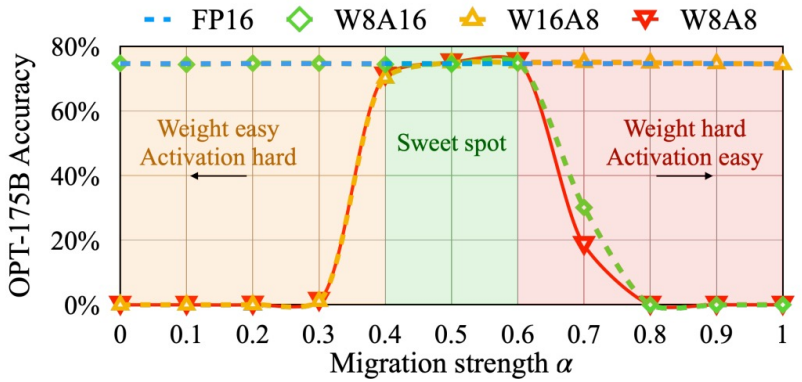


Figure 9: A suitable migration strength α (sweet spot) makes both activations and weights easy to quantize. If the α is too large, weights will be hard to quantize; if too small, activations will be hard to quantize.

Robust to the calibration set distributions

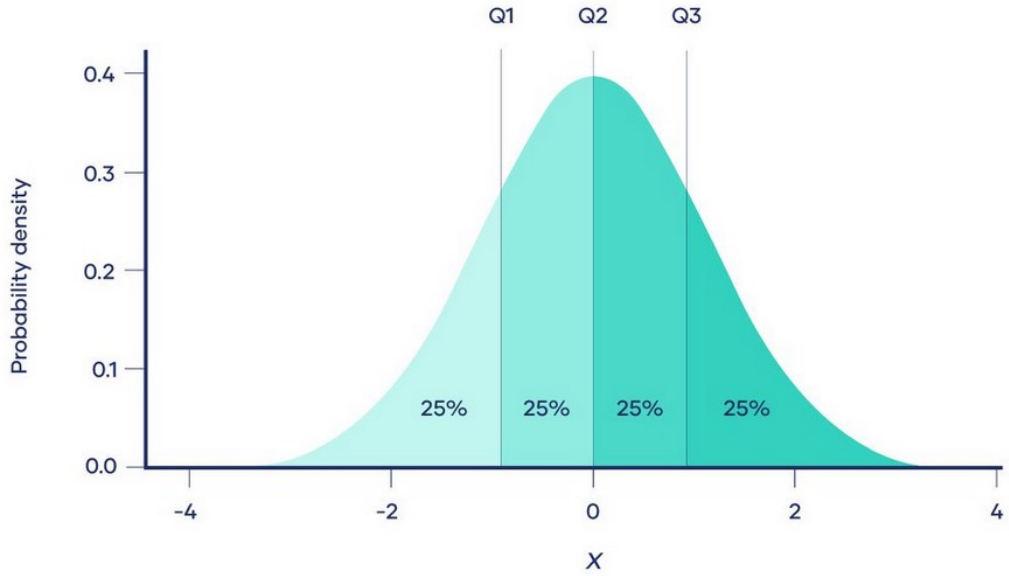
Calib \ Eval	GPTQ		Ours	
	PubMed	Enron	PubMed	Enron
PubMed	32.48	50.41 +4.89	32.56	45.07 +0.50
Enron	+2.33 34.81	45.52	+0.60 33.16	44.57

QLoRA

- Quantization setting: 4-bit weight-only quantization for pretrained weights and 16/32-bit lora weights

$$\mathbf{Y}^{\text{BF16}} = \mathbf{X}^{\text{BF16}}\text{doubleDequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}, \mathbf{W}^{\text{NF4}}) + \mathbf{X}^{\text{BF16}}\mathbf{L}_1^{\text{BF16}}\mathbf{L}_2^{\text{BF16}},$$

- Quantization method: block-wise norm. + NormalFloat type
  - NormalFloat: quantile on normal distribution



$$q_i = \frac{1}{2}\left(Q_X\left(\frac{i}{2^k+1}\right) + Q_X\left(\frac{i+1}{2^k+1}\right)\right),$$

Further memory efficiency

- Double quantization
  - If 32-bit scales and a blocksize of 64 are used for W, quantization scales add 32/64 = 0.5 bits per parameter on average. Double Quantization means quantize the quantization scales, with blocksize of 256 and FP8 data type.
  - overhead: 32/64 = 0.5 bits => 8/64 + 32/(64 · 256) = 0.127 bits
- Paged optimizer: surviving mem spikes
  - similar to optimizer offloading, but more adaptive

=> With QLoRA, one can finetune Guanaco 33B/65B models on a single 24/48GB GPU taking only 12/24h for a finetuning run.

Evaluation

Table 3: Experiments comparing 16-bit BrainFloat (BF16), 8-bit Integer (Int8), 4-bit Float (FP4), and 4-bit NormalFloat (NF4) on GLUE and Super-NaturalInstructions. QLoRA replicates 16-bit LoRA and full-finetuning.

Dataset	GLUE (Acc.)	Super-NaturalInstructions (RougeL)				
Model	RoBERTa-large	T5-80M	T5-250M	T5-780M	T5-3B	T5-11B
BF16	88.6	40.1	42.1	48.0	54.3	62.0
BF16 replication	88.6	40.0	42.2	47.3	54.9	-
LoRA BF16	88.8	40.5	42.6	47.1	55.4	60.7
QLoRA Int8	88.8	40.4	42.9	45.4	56.5	60.7
QLoRA FP4	88.6	40.3	42.4	47.5	55.6	60.9
QLoRA NF4 + DQ	-	40.4	42.7	47.7	55.3	60.9

Table 4: Mean 5-shot MMLU test accuracy for LLaMA 7-65B models finetuned with adapters on Alpaca and FLAN v2 for different data types. Overall, NF4 with double quantization (DQ) matches BFloat16 performance, while FP4 is consistently one percentage point behind both.

	Mean 5-shot MMLU Accuracy								
LLaMA Size	7B		13B		33B		65B		Mean
Dataset	Alpaca	FLAN v2	Alpaca	FLAN v2	Alpaca	FLAN v2	Alpaca	FLAN v2	
BFloat16	38.4	45.6	47.2	50.6	57.7	60.5	61.8	62.5	53.0
Float4	37.2	44.0	47.3	50.0	55.9	58.5	61.3	63.3	52.2
NFloat4 + DQ	39.0	44.5	47.5	50.7	57.3	59.2	61.8	63.9	53.1

1. Match baseline
2. DQ not degrade
3. NF better than FP

Reference

[1] Dettmers, Tim, et al. "Llm. int8 (): 8-bit matrix multiplication for transformers at scale." *arXiv preprint arXiv:2208.07339* (2022).

[2] Xiao, Guangxuan, et al. "Smoothquant: Accurate and efficient post-training quantization for large language models." *arXiv preprint arXiv:2211.10438* (2022).

[3] Lin, Ji, et al. "AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration." *arXiv preprint arXiv:2306.00978* (2023).

[4] Dettmers, Tim, et al. "QLoRA: Efficient Finetuning of Quantized LLMs." *arXiv preprint arXiv:2305.14314* (2023).

Thanks