UNIVERSITY OF TORONTO

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models

@ MIT HAN lab

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, Song Han

Present by: Kecen Yao & Zhichen Ren

3/21/2025

Background: What is Quantization?

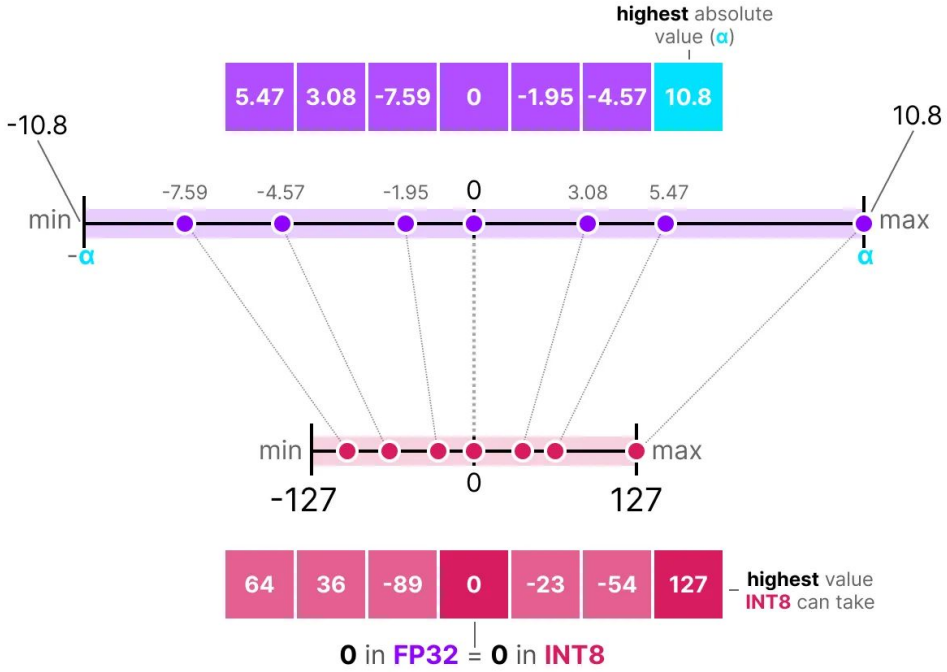
Lower the bit-width and increase the efficiency

$$\bar{\mathbf{X}}^{\mathrm{INT8}} = \left\lceil \frac{\mathbf{X}^{\mathrm{FP16}}}{\Delta} \right\rfloor, \quad \Delta = \frac{\max(|\mathbf{X}|)}{2^{N-1} - 1}$$

Scaling factor

Background: What is Quantization?

Lower the bit-width and increase the efficiency



$$\bar{\mathbf{X}}^{\text{INT8}} = \lceil \frac{\mathbf{X}^{\text{FP16}}}{\Delta} \rfloor, \quad \Delta = \frac{\max(|\mathbf{X}|)}{2^{N-1}-1}$$

INT8 Quantization -> N=8 ,

X(FP16) = [5.47,3.08,-7.59,0,-1.95,-4.57,**10.8**]

max(|X|) = 10.8, Δ = 10.8 / 127 = 0.085

X(INT8) = [64,36,-89,0,-23,-54,**127**]

- preserves relative values
- sensitive to outliers

Background: What is Quantization?

Lower the bit-width and increase the efficiency

$$Y^{FP16} = X^{FP16} * W^{FP16}$$

$$= \quad \mathrm{diag}(\Delta_X^{FP16}) \cdot (\bar{\mathbf{X}}^{INT8} \cdot \bar{\mathbf{W}}^{INT8}) \cdot \mathrm{diag}(\Delta_W^{FP16})$$

⬆️
Smaller Storage (16 bit to 8 bit)
& Compute faster
(integer kernels INT8 GEMM)

Traditional Method for CNN: W8A8

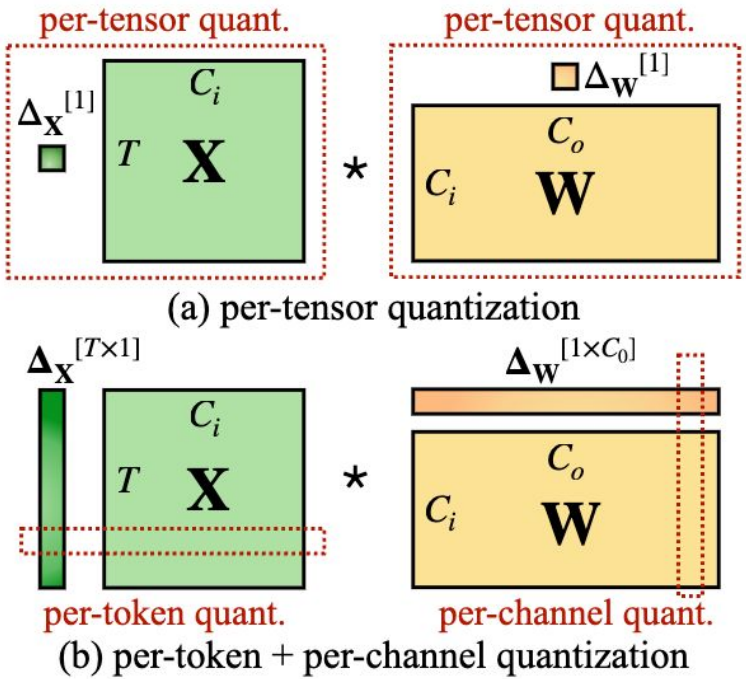
Background: Type of Quantization

Lower the bit-width and increase the efficiency

- **Post-Training Quantization (PTQ): quantization during inference**
  - **static quantization** (precomputed scaling factors)
  - **dynamic quantization** (scaling factors computed at runtime).
- Quantization-Aware Training (QAT)**:** Simulates quantization effects during training.

Background: Different granularity level

Lower the bit-width and increase the efficiency

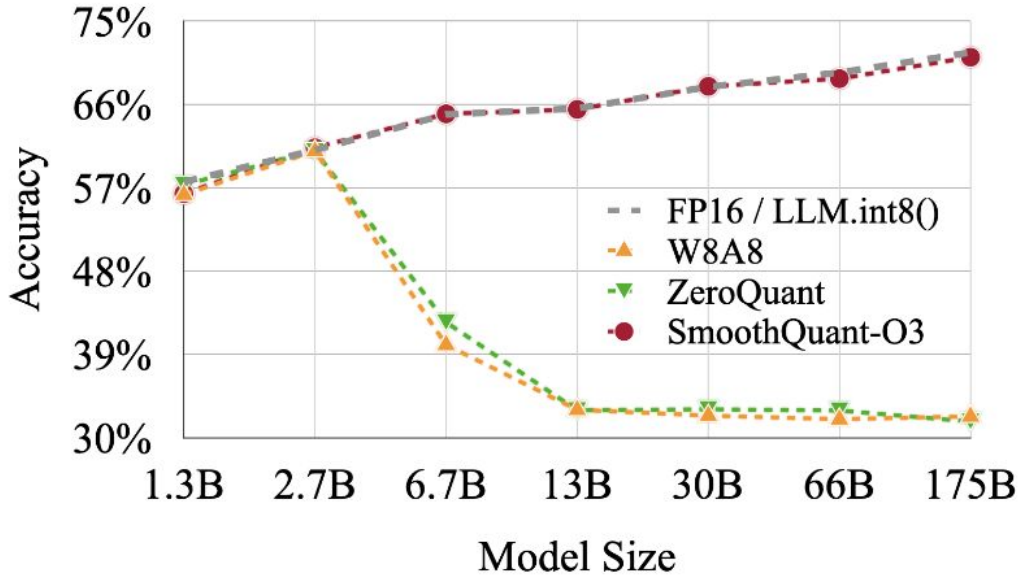


- Per-tensor
- Per-token (typically for activation)
- Per-channel (typically for weight)

Background: Quantization for LLM

So… How about Quantization in LLM?

Background: Quantization for LLM

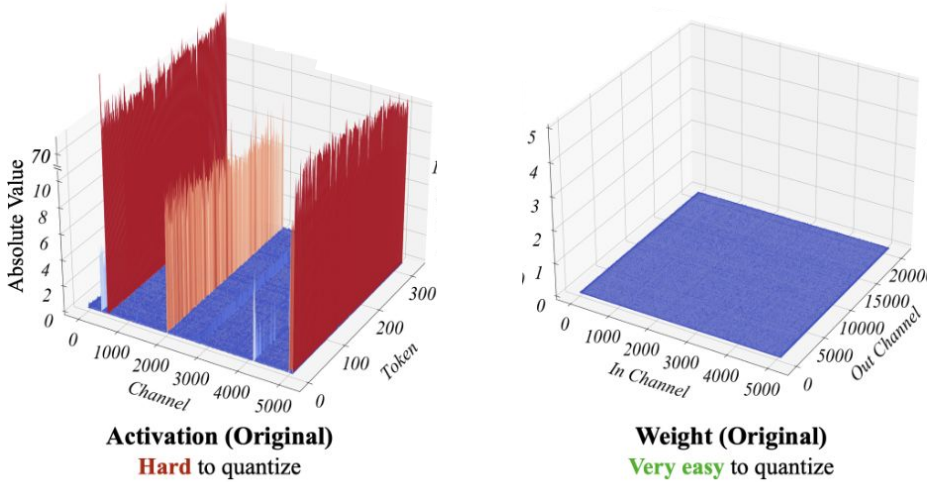


Traditional W8A8 in LLM ❌ WHY?

- systematic outliers with large magnitude will emerge in activations when LLM > 6.7B

Background: Quantization for LLM

Smoothing activation to reduce quantization error



- Weight are easy to quanize, but activation is hard due to outliers
- Luckily, outliers persist in fixed channels

Related Work

How other research address with this issue?

- **LLM.int8:** a mixed-precision decomposition (outliers in FP16 and INT8 for other)

  ❌ large latency overhead, which can be even slower than FP16 inference

- **ZeroQuant:** dynamic per-token activation quantization and group-wise weight quantization

  ❌ requires customized CUDA kernels and not maintain the accuracy for the large model

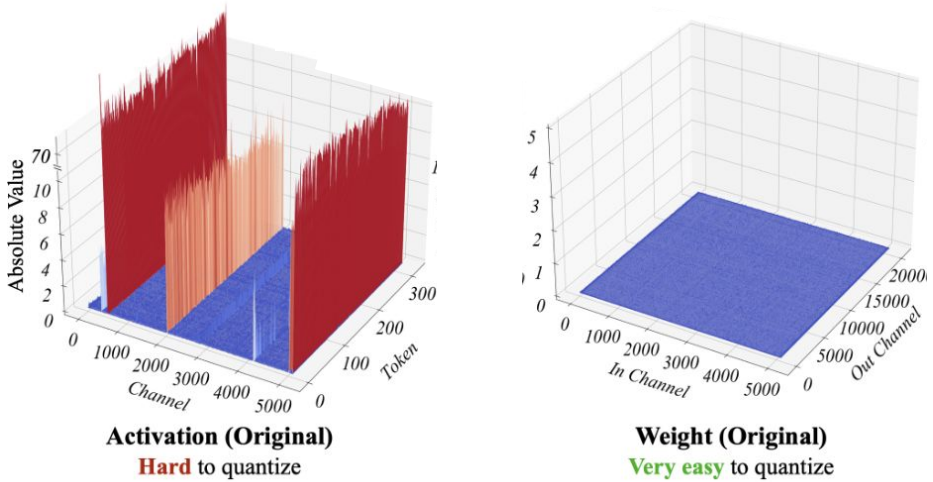
- **Outlier Suppression:** non-scaling LayerNorm and token-wise clipping to deal with the activation Outliers

  ❌ only succeeds on small language models

- What else…?

Background: Quantization for LLM

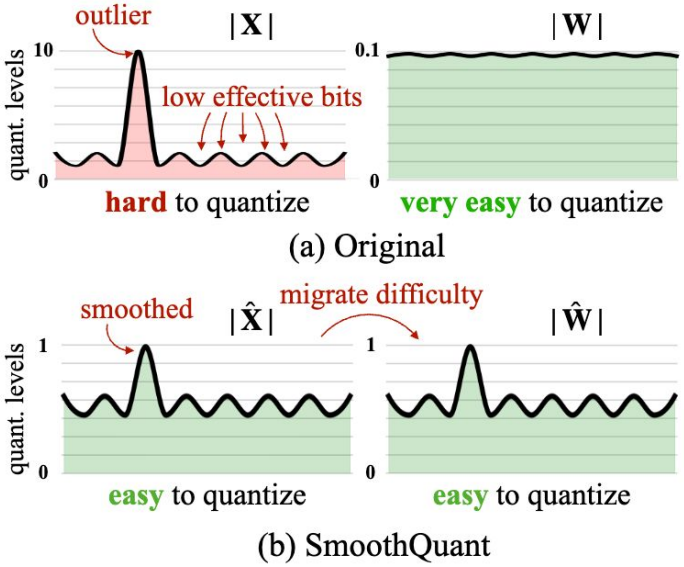
Smoothing activation to reduce quantization error



- Migrate the quantization difficulty from activation to weights, so both are easy to quantize

Motivation

Key Idea: weights are easy to quantize while activations are hard.



Methods

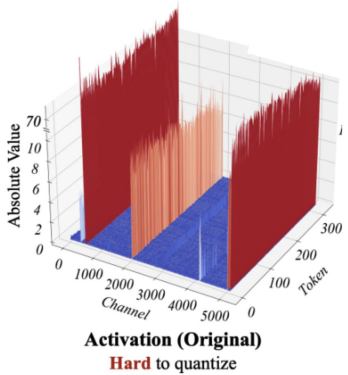
$$\mathbf{Y} = \mathbf{X} \cdot \mathbf{W}$$

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

Objective: Find a diagonal matrix s to migrate difficulty

Methods: Migration Matrix Selection

Observation: Variance in one activation channel is small



Idea: Convert each channel of activations into similar distribution

$$\mathbf{s}_j = \max(|\mathbf{X}_j|), j = 1, 2, ..., C_i$$

Good Enough?

Methods: Difficulty Migration

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

Difficulty are migrated from activations to weights!

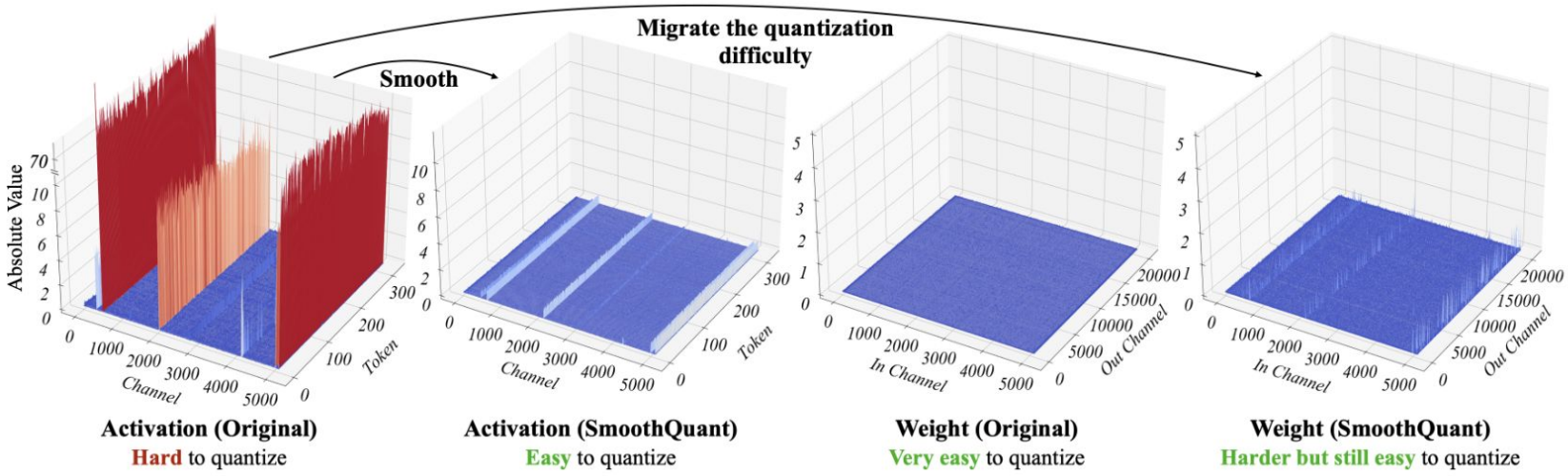
Methods: Difficulty Migration

$$\mathbf{s}_j = \max(|\mathbf{X}_j|), j = 1, 2, ..., C_i$$

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

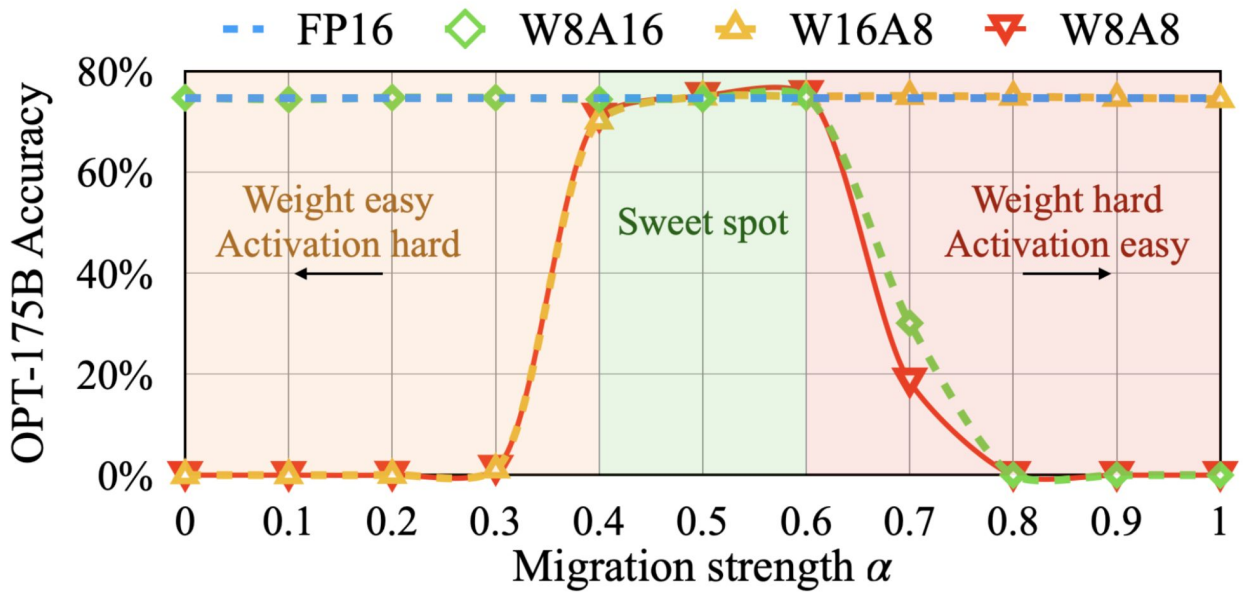
Idea: migrate difficulty **"smoothly"** so both activation and weights are easy to quantize. (α is a hyper parameter from 0 to 1)

Methods: Difficulty Migration



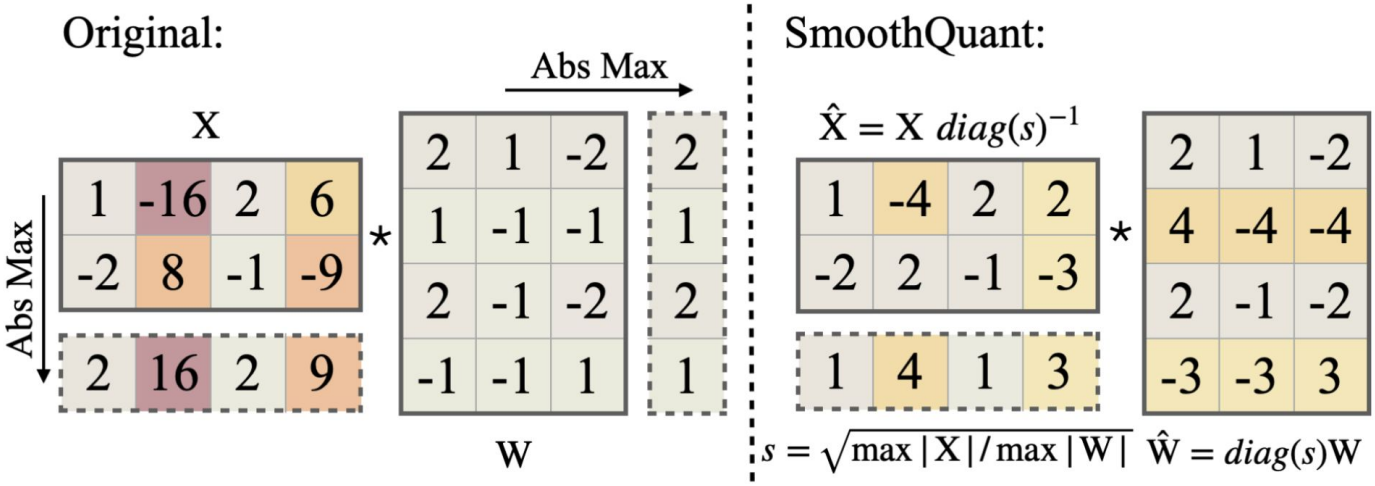
Idea: migrate difficulty **"smoothly"** so both activation and weights are easy to quantize.

Methods: Migration Strength



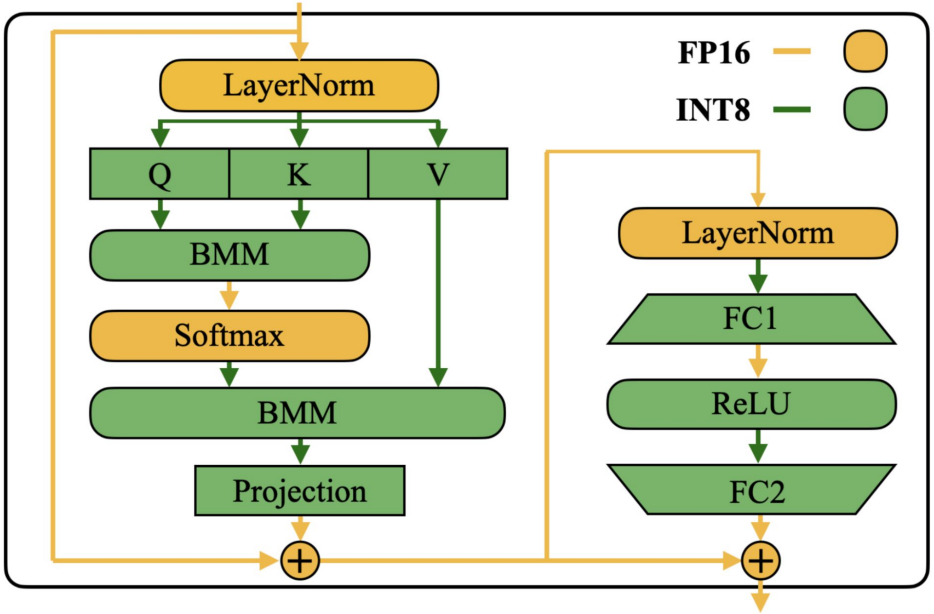
Suitable migration strength has good performance

SmoothQuant Example



Example of SmoothQuant when α is 0.5

SmoothQuant in Transformers



SmoothQuant is applied in the attention and linear calculation

Experiments: Experiments Setups

Baselines

- W8A8 Naive Quantization
- ZeroQuant
- LLM.int8()
- Outlier Suppression

Models

- OPT
- BLOOM
- GLM

Evaluation Dataset

- LAMBADA
- HellaSwag
- PIQA
- ……

Migration Strength

- OPT $\alpha = 0.5$
- BLOOM $\alpha = 0.5$
- GLM $\alpha = 0.75$

Experiments: Quantization Setups

Method	Weight	Activation
W8A8	per-tensor	per-tensor dynamic
ZeroQuant	group-wise	per-token dynamic
`LLM.int8()`	per-channel	per-token dynamic+FP16
Outlier Suppression	per-tensor	per-tensor static
SmoothQuant-O1	per-tensor	per-token dynamic
SmoothQuant-O2	per-tensor	per-tensor dynamic
SmoothQuant-O3	per-tensor	per-tensor static

Quantization setting of baselines and SmoothQuant

Experiments: Accuracy Evaluation

OPT-175B	LAMBADA	HellaSwag	PIQA	WinoGrande	OpenBookQA	RTE	COPA	**Average↑**	**WikiText↓**
FP16	74.7%	59.3%	79.7%	72.6%	34.0%	59.9%	88.0%	66.9%	10.99
W8A8	0.0%	25.6%	53.4%	50.3%	14.0%	49.5%	56.0%	35.5%	93080
ZeroQuant	0.0%*	26.0%	51.7%	49.3%	17.8%	50.9%	55.0%	35.8%	84648
`LLM.int8()`	74.7%	59.2%	79.7%	72.1%	34.2%	60.3%	87.0%	66.7%	11.10
Outlier Suppression	0.00%	25.8%	52.5%	48.6%	16.6%	53.4%	55.0%	36.0%	96151
SmoothQuant-O1	74.7%	59.2%	79.7%	71.2%	33.4%	58.1%	89.0%	66.5%	11.11
SmoothQuant-O2	75.0%	59.0%	79.2%	71.2%	33.0%	59.6%	88.0%	66.4%	11.14
SmoothQuant-O3	74.6%	58.9%	79.7%	71.2%	33.4%	59.9%	90.0%	66.8%	11.17

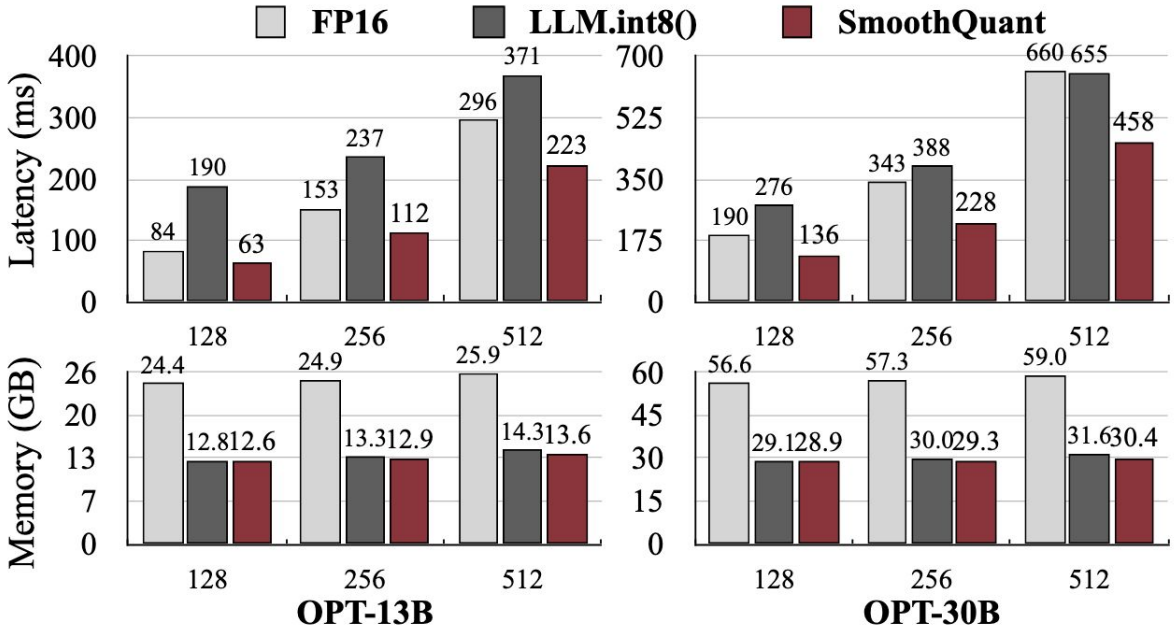
Accuracy of OPT-175B using different quantization methods on different dataset

Experiments: Accuracy Evaluation

Method	OPT-175B	BLOOM-176B	GLM-130B*
FP16	71.6%	68.2%	73.8%
W8A8	32.3%	64.2%	26.9%
ZeroQuant	31.7%	67.4%	26.7%
`LLM.int8()`	71.4%	68.0%	73.8%
Outlier Suppression	31.7%	54.1%	63.5%
SmoothQuant-O1	**71.2%**	68.3%	**73.7%**
SmoothQuant-O2	71.1%	**68.4%**	72.5%
SmoothQuant-O3	71.1%	67.4%	72.8%

Average accuracy of OPT, BLOOM, GLM on different dataset

Experiments: Speedup and Memory Save



Latency and memory cost of OPT model at different sequence length

Limitation

- Extra resources are required to find optimal hyperparameter.
- Migrate difficulty, but not decrease overall difficulty.
- Focus on W8A8 quantization, which is not efficient enough for LLMs now.

Conclusion

- SmoothQuant balances quantization difficulty between activations and weights.
- SmoothQuant enables almost lossless 8-bit weight and activation quantization for different LLMs up to 530B.
- With comparable accuracy, SmoothQuant gets up to 1.51x inference acceleration and 1.91x memory saving.