Intriguing Properties of Quantization at Scale

Arash Ahmadian*
Cohere For AI
University of Toronto

Saurabh Dash*
Cohere

Hongyu Chen*
Cohere

Bharat Venkitesh
Cohere

Stephen Gou
Cohere

Phil Blunsom
Cohere

Ahmet Üstün
Cohere For AI

Sara Hooker
Cohere For AI

Cohere For AI

Massive Interest in Quantizing LLMs

A Survey of Quantization Methods for Efficient Neural Net

AINING QUANTIZATION

NED TRANSFORMERS

Transformer Quantization

Outlier Suppression: Pushing the Limit of Low-bit Transformer Language Models

`LLM.int8()`: **8-bit Matri for Transformers**

INTEGER QUANTIZATION FOR DEEP LEARNING INFERENCE: PRINCIPLES AND EMPIRICAL EVALUATION

LLMs Burn through GPU Credits Faster than Fire

	What people talk about	What really matters
2016	Number of layers	Performance, latency, cost
2018	Papers @ NeurIPS	Performance, latency, cost
2019	State-of-the-art GLUE score	Performance, latency, cost
2020	Amount of compute	Performance, latency, cost
2022	Number of parameters	Performance, latency, cost
2023	Context length	Performance, latency, cost

Spotted by Patrick Lewis at UCL; source unknown

Emergent Properties in Large Models

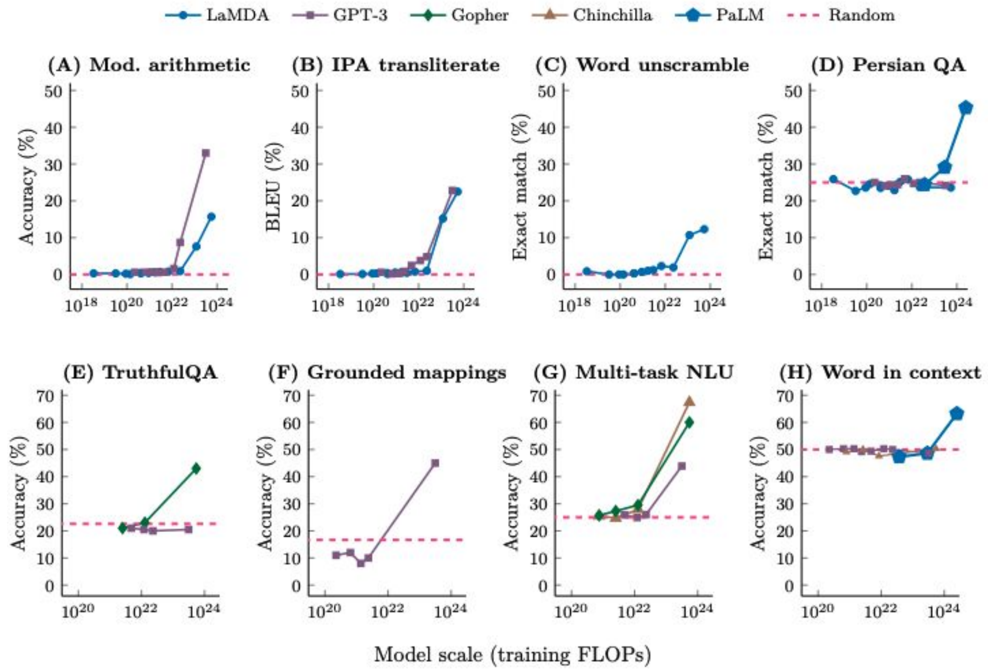
Cohere For AI

Emergent Properties

- Properties/abilities that are "present in larger language models but not in smaller ones"(Wei et. al., 2022)

- Larger models → higher inference cost (latency, memory)

- **Quantization** can help remedy this cost



Few-shot performance (credits : Wei et. al., 2022)

Cohere For AI

Quantization: A ray of hope for efficiency

Cohere For AI

Quantization 101

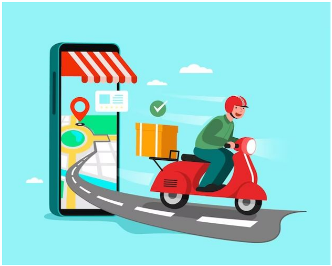
CPU/GPU memory

CPU/GPU computation

I/O - data movement

Quantization 101

Quantization: reduce the amount of data to store, compute and move



Background: Quantization

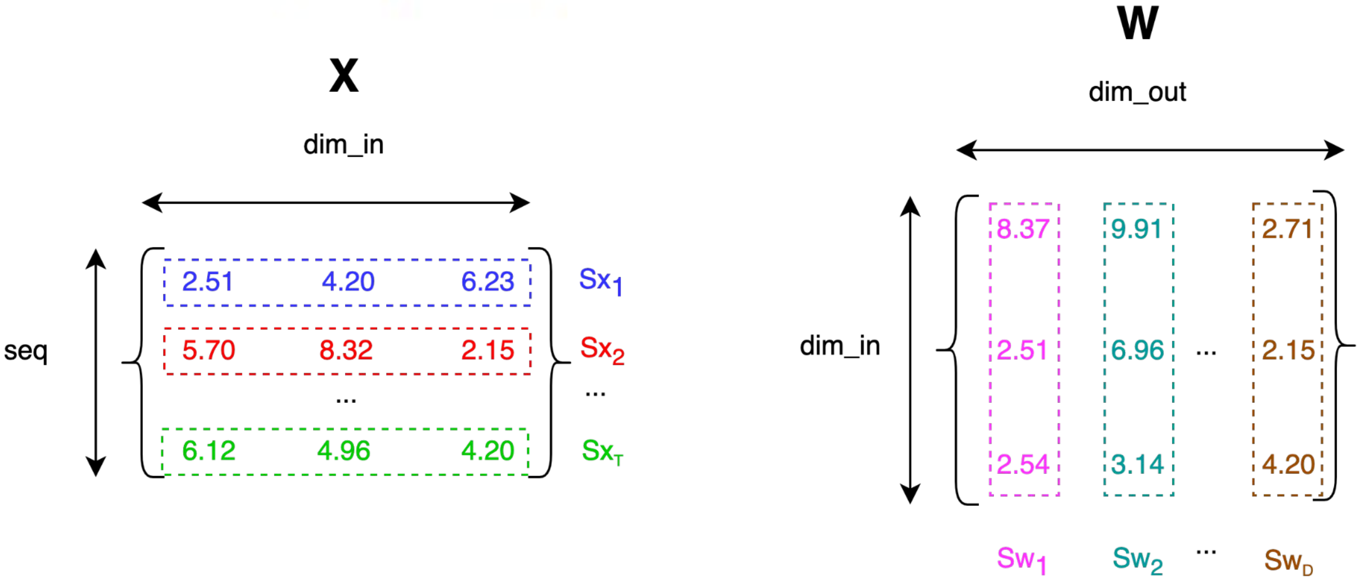
Traditional absmax-int8:

$$\mathbf{X}_{i8} = \left\lfloor \frac{127 \cdot \mathbf{X}_{f16}}{\max\limits_{ij}(|\mathbf{X}_{f16_{ij}}|)} \right\rceil = \left\lfloor \frac{127}{\|\mathbf{X}_{f16}\|_\infty} \mathbf{X}_{f16} \right\rceil = \left\lfloor s_{x_{f16}} \mathbf{X}_{f16} \right\rceil,$$

Pros: Straight forward
Cons: Not enough granularity in most cases

Vectorwise INT8 MatMul

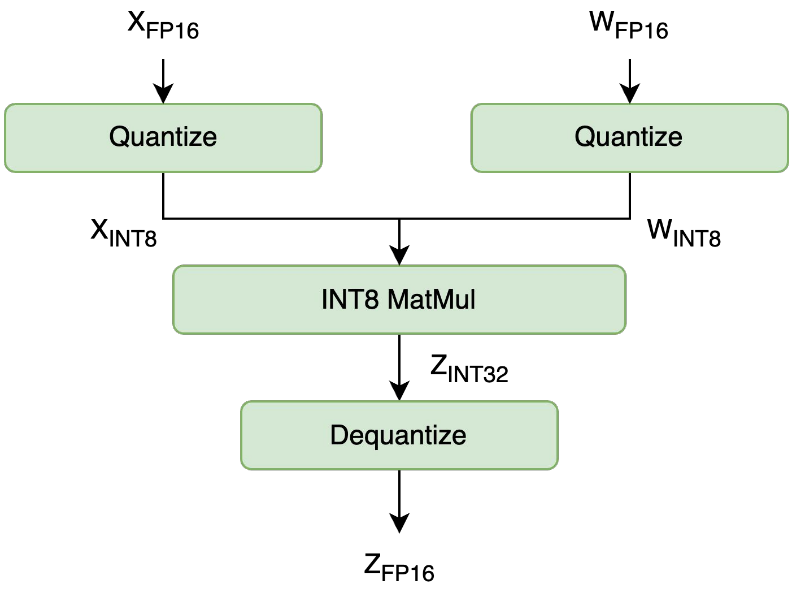
$$\mathbf{XW} \approx \mathbf{s_x} \odot (\mathbf{X_Q W_Q}) \odot \mathbf{s_w}$$



$\mathbf{s_x} \in \mathbb{R}^t$

$\mathbf{s_w} \in \mathbb{R}^o$

Quantization 101

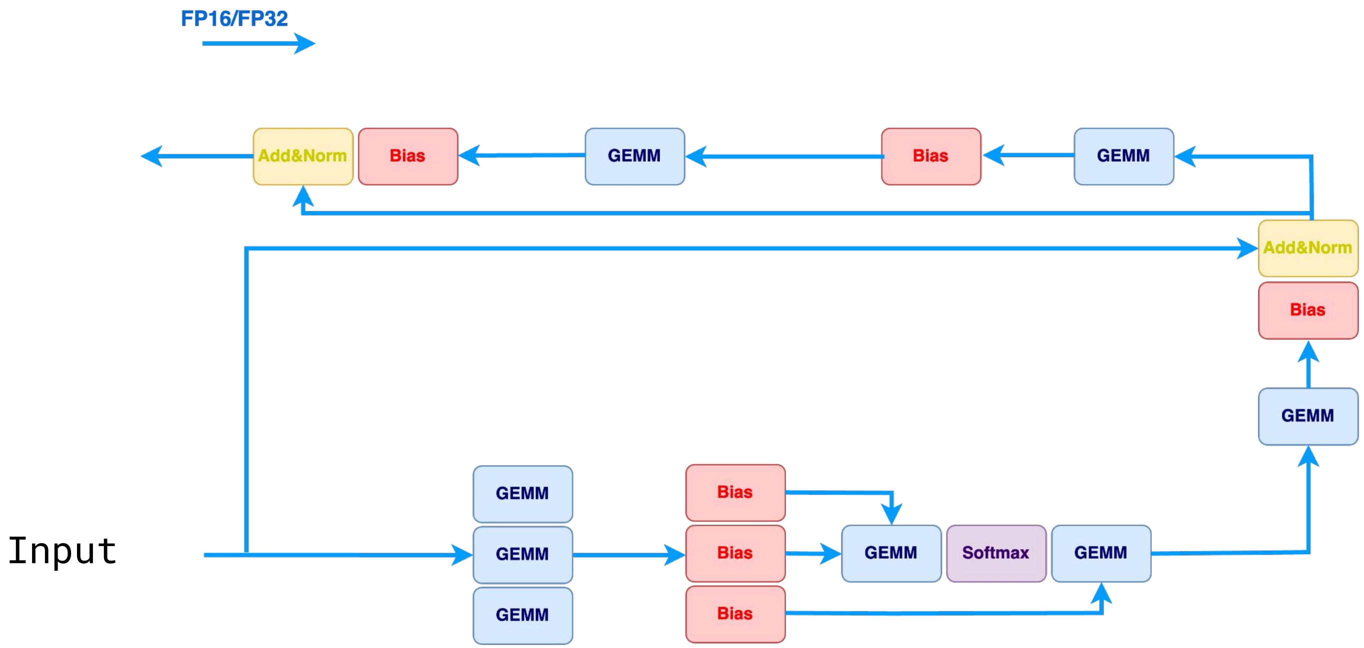


Weight quantization

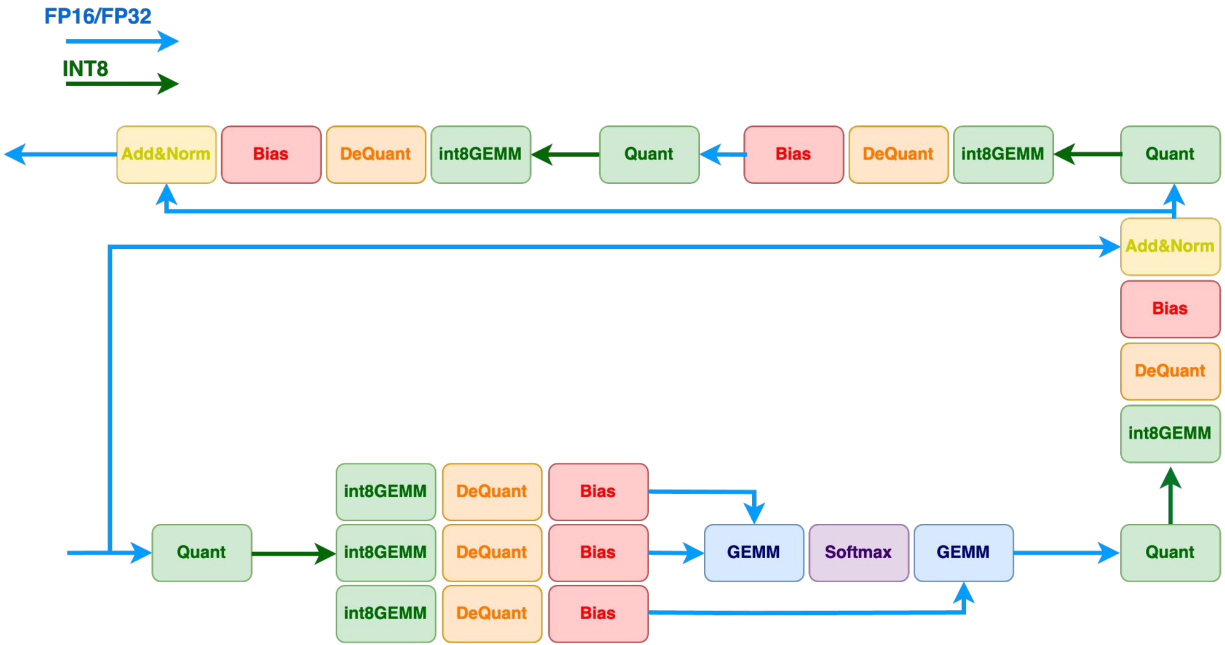
Activation quantization

INT8 MatMul

Quantization 101

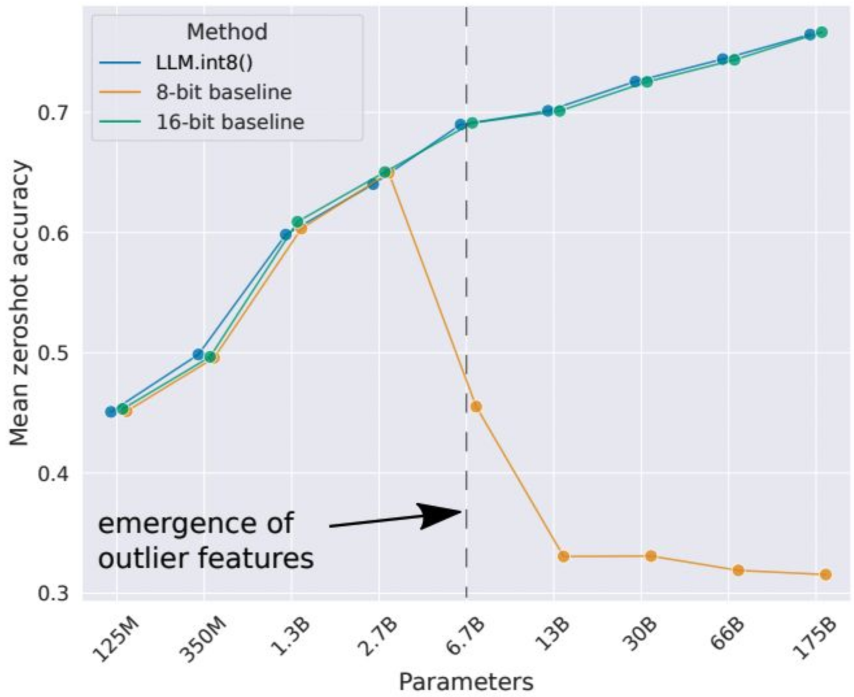


Quantization 101



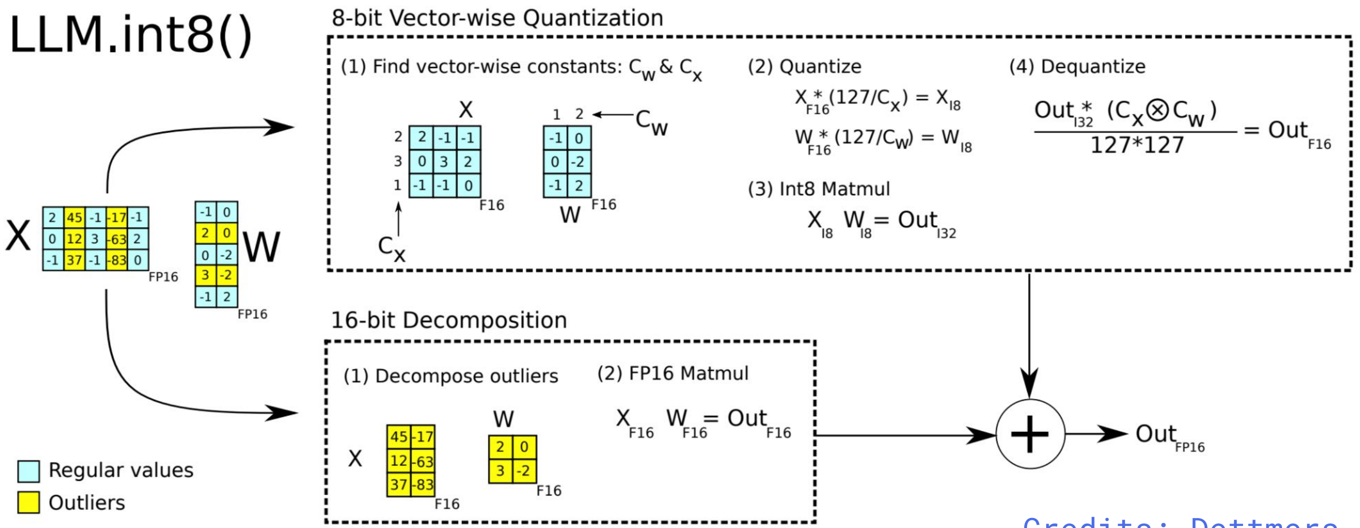
Cohere For AI

Vectorwise INT8 Methods FAIL at Scale?



Credits: Dettmers et al., 2022

Vectorwise INT8 Methods FAIL at Scale?



Credits: Dettmers et al., 2022

Better for memory, but adds computation overhead, no improvement on latency

Vectorwise INT8 Methods FAIL at Scale?

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%

Credits: Xiao et al., 2022

“

Are Emergent Outliers due to Nature or Nurture?

Axes of Experimentation

01 Weight Decay

02 Gradient Clipping

03 Dropout

04 Data-type Precision

Methodology

- Isolate effects of each optimization choice -> controlled setup
- High cost of training at scale -> 6B early checkpoint (75k steps)

Experimental Axes	Choices
Weight decay	0.001, 0.01, 0.1
Gradient clipping	None, 1
Dropout	0, 0.1, 0.4, 0.8
Half-precision	`bf16`, `fp16`

Model and Dataset Details

- GPT based models trained with C4

Benchmark	Task Type	Evaluation Metric
Copa (test set)(Wang et al., 2019)	MC Completion	MC Accuracy
Copa100 (dev set) (Wang et al., 2019)	MC Completion	MC Accuracy
HellaSwag (Zellers et al., 2019)	MC Completion	MC Accuracy
PIQAValidation (Bisk et al., 2020)	MC Completion	MC Accuracy
StoryCloze (Mostafazadeh et al., 2016)	MC Completion	MC Accuracy
WinoGrande (Sakaguchi et al., 2019)	MC Co-referencing	MC Accuracy
Paralex (Fader et al., 2013)	Generation	Likelihood (bytes)
LAMBADA (Paperno et al., 2016)	Generation	Exact String Match Accuracy

Table 3: An Overview of the 8 tasks we benchmark the zero-shot downstream performance of trained models. QA and MC denotes Question Answering, and Multiple-choice respectively.

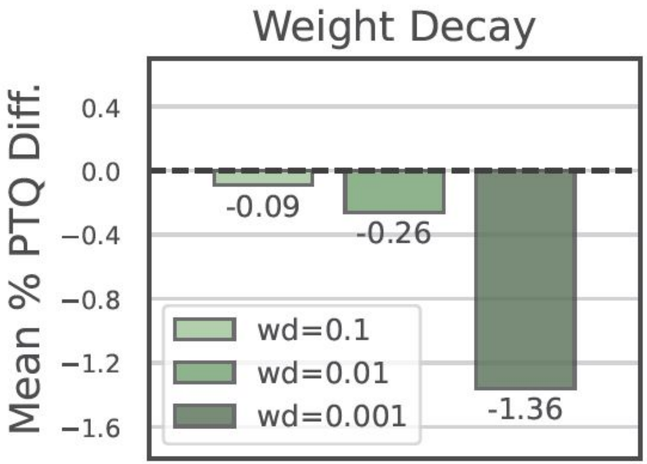
01

Weight Decay

Cohere For AI

Weight Decay

- Vary weight decay with gradient-clipping turned off
- Want to decouple their effects
- Higher weight decay → better PTQ

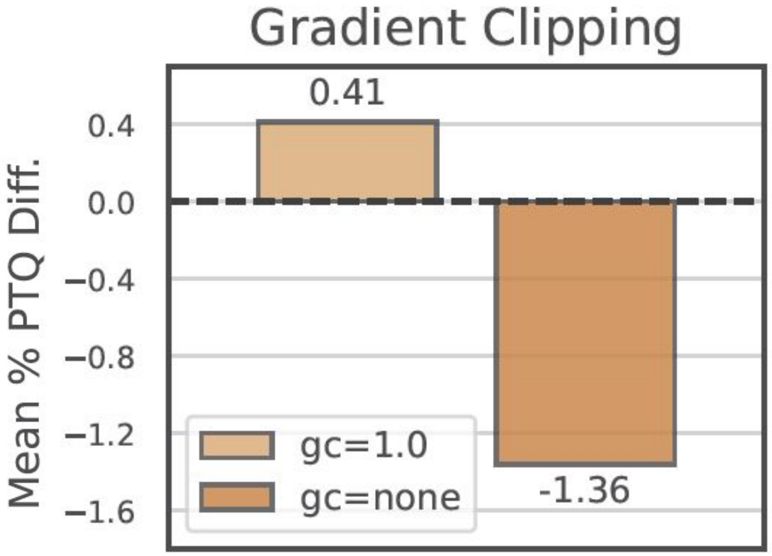


02

Gradient Clipping

Gradient Clipping

- Vary gradient-clipping with weight decay = 0.001
- Want to decouple the effects of two
- Gradient Clipping → better PTQ

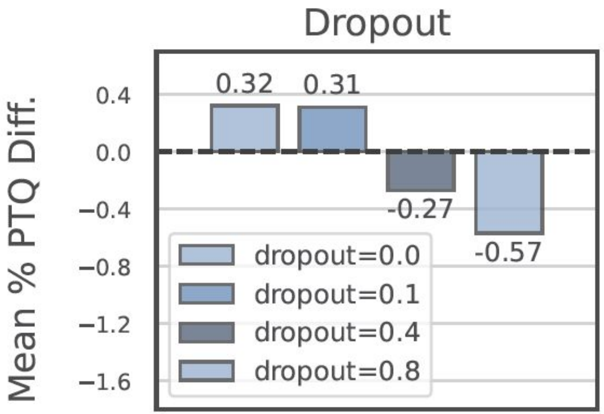


03

Dropout

Dropout

- Only applied to the activations right before a residual connection
- Not applied to embeddings
- dropout=0.8 has significantly worse fp16 performance (expected)
- Smaller dropout → better PTQ

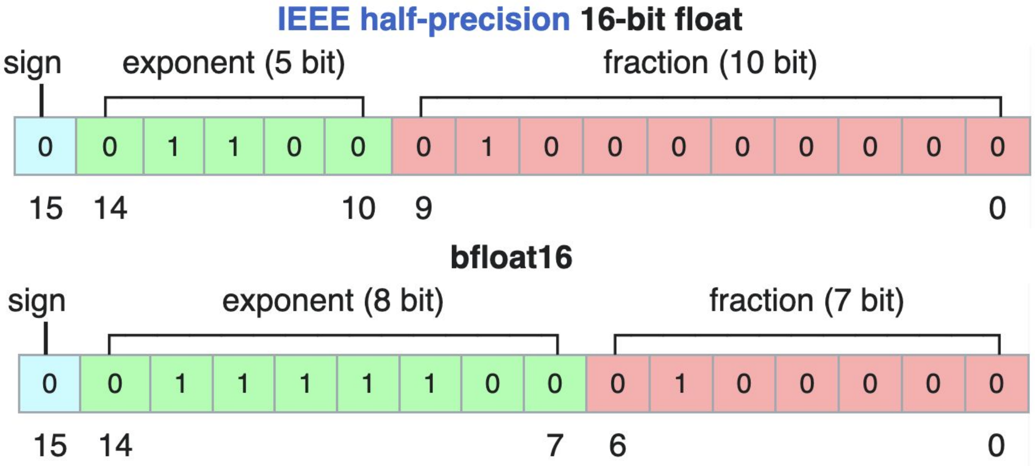


04

Data-type Precision

Data-type Precision

- FP16 has a smaller dynamic range → higher precision
- BF16 has higher dynamic range (same as fp32) → less precision
- FP16 training is very-hacky (loss-scaling, rewinding, etc.) while BF16 training is not

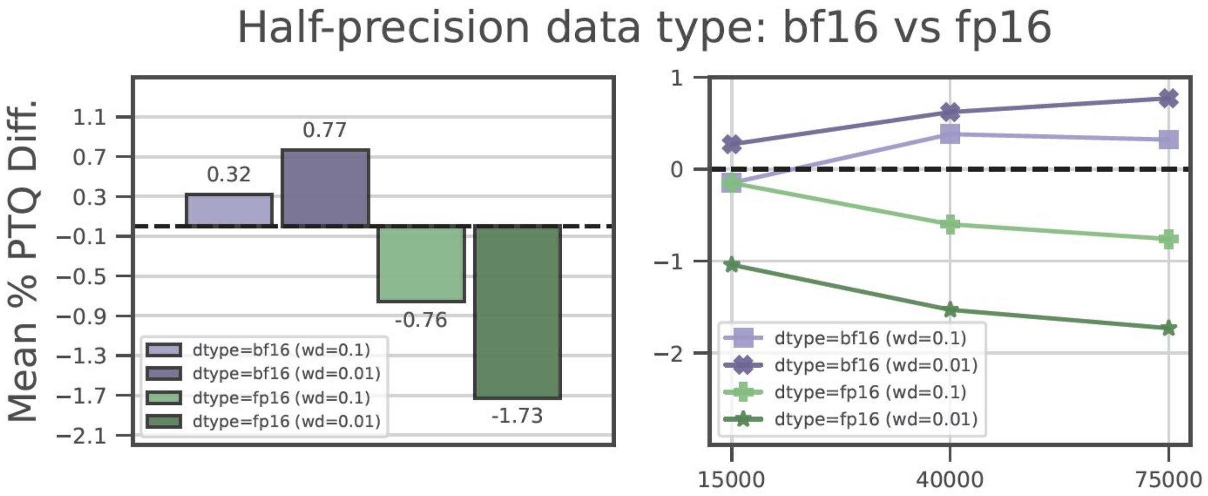


Cohere For AI

Source: Wikipedia

BF16 vs FP16

- Note: layernorm in fp32 since fp16 layernorm lead to loss divergence
- Fp16 → worse PTQ (most significant out of all experimental axis)
- Degradation trend consistent over time



Analysis

Outliers

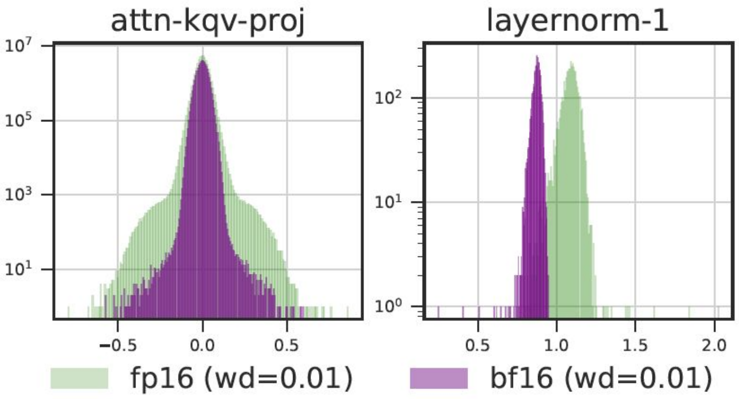
Table 5: Outlier statistic using different thresholding rules **Top Table:** constant threshold $\alpha > 4.2$
Bottom Table: adaptive threshold $\alpha > 4\sigma_{token} + \mu_{token}$

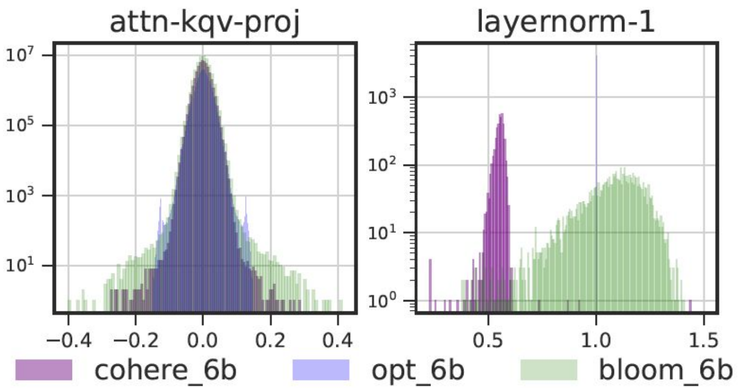
Variant	#Outliers	%Seq Affected	%Layers Affected
{all other variants}	0	0	0
dtype=fp16 (wd=0.01)	6	68.4	65.5
cohere_410M	1	25.0	26.3
cohere_6B	0	0	0

dropout=0.1	2	44.0	40.5
dropout=0.4	2	44.9	42.9
dropout=0.8	1	19.2	25
wd=0.1 (gc=none)	2	28.0	39.3
wd=0.01 (gc=none)	0	0	0
wd=0.001 (gc=none)	2	34.3	34.5
dtype=bf16 (wd=0.1)	2	33.0	36.9
dtype=fp16 (wd=0.1)	2	41.1	42.9
dtype=bf16 (wd=0.01)	0	0	0
dtype=fp16 (wd=0.01)	8	66.2	64.3
cohere_410M	55	88.2	86.2
cohere_6B	7	65.5	56

Weight Distribution

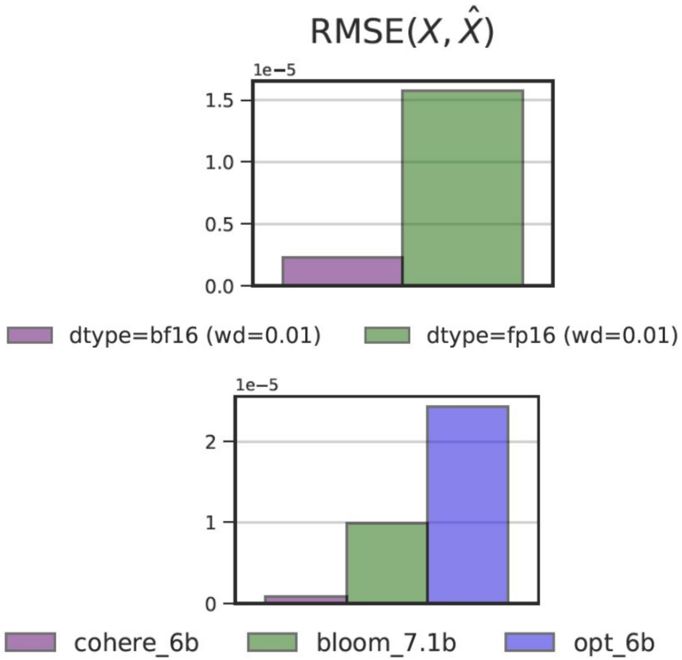
- Attn-kqv-proj exhibited the highest change between bf16 and fp16 variants
- Layernorm scales directly impact spread of activation values





Reconstruction Loss

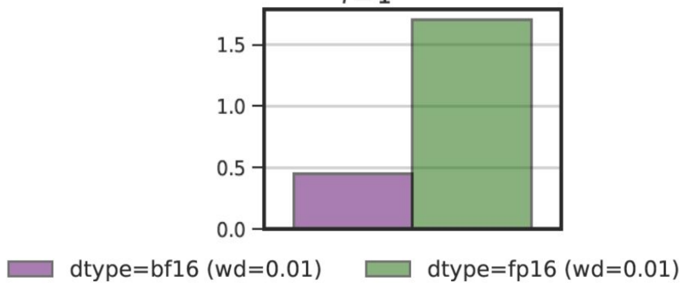
- Directly relates to quantization error
- Generally, variants with higher degradation have higher loss

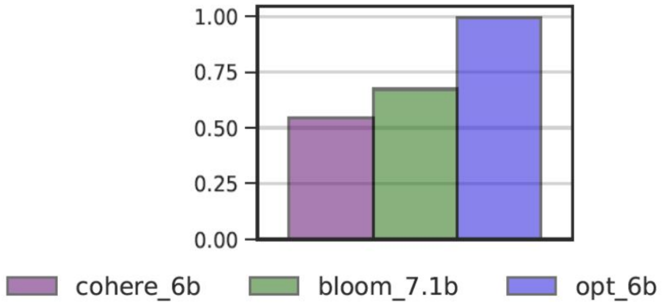


Activation Token Standard Deviation

- Measures spread of token activations → directly relates to expected quantization error for a Gaussian (Kuzmin et. al)
- Generally variants with higher std show higher PTQ degradation

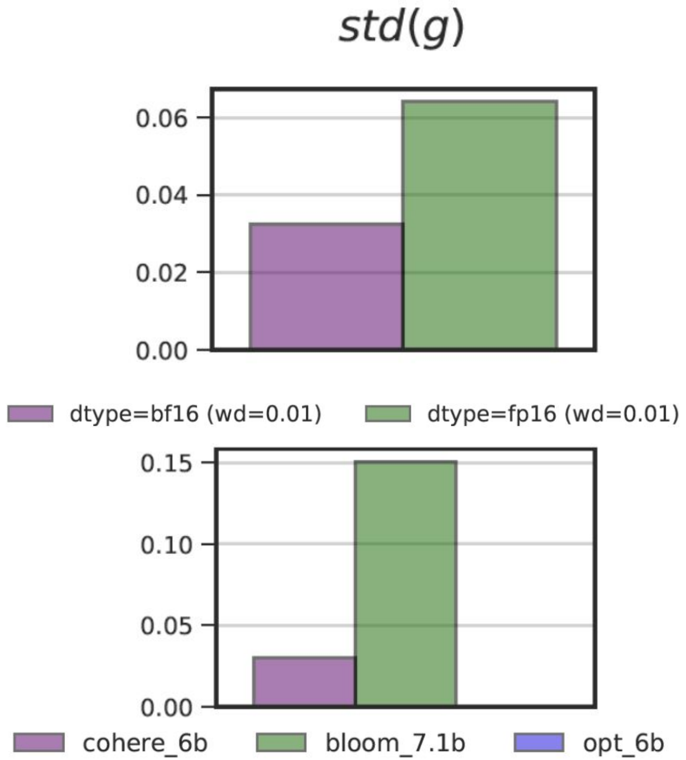
$$\frac{1}{t}\sum_{i=1}^{t} std(X_i)$$





Layernorm-gain Standard Deviation

- Layernorm layers can act as activation outlier amplifiers [or suppressors] (Wei et. al)
- Determined by layernorm gain parameters
- Layernorms in variants with higher degradation generally have larger gain



Corresponding to First layernorm in the attention block

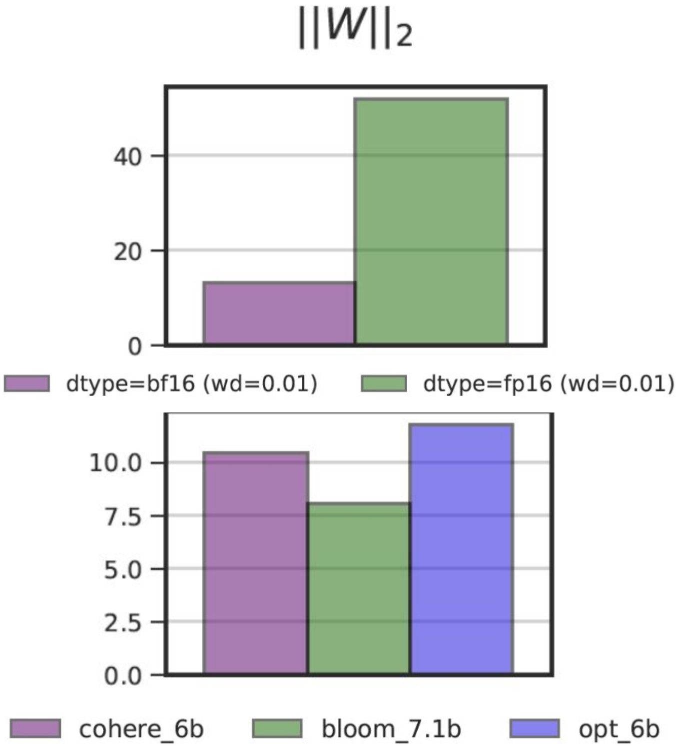
Spectral Norm

- Measures maximum degree of input activation noise amplification due to the weights

- Has been previously used in quantization coupled with robustness (Lin et al)

$$\|\mathbf{W}\|_2 = \sup_{\mathbf{x}_\delta \neq 0} \frac{\|\mathbf{W}\mathbf{x}_\delta\|_2}{\|\mathbf{x}_\delta\|_2} = \sigma_{max}$$

- This is also the Lipschitz constant for the MatMul function



Cohere For AI

Are Emergent Outliers due to Nature or Nurture?

Cohere For AI

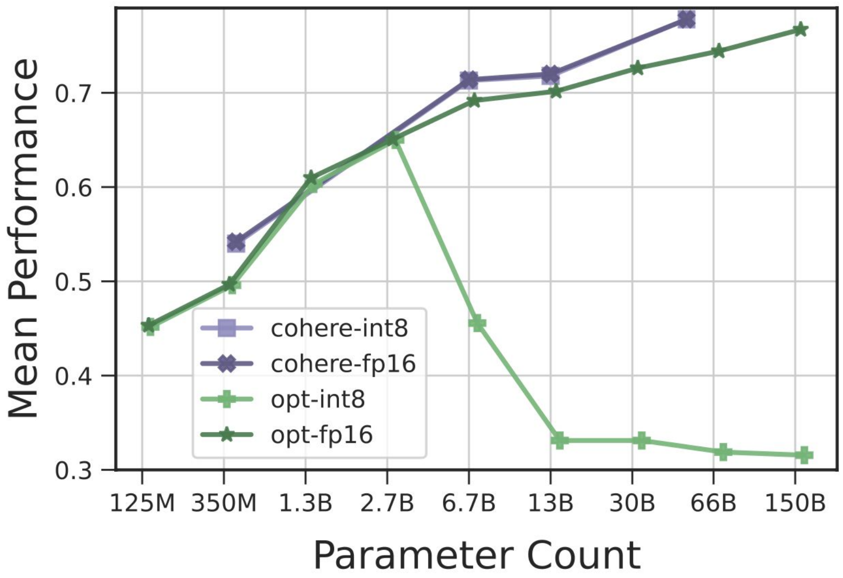
Our Perspective: Nurture

A.1 Optimal Hyper-parameters

Weight decay	Gradient clipping	Dropout	Half-precision datatype
0.1	1.0	0	bf16

Table 2: Optimal hyper-parameters for PTQ based on results in Section 4

Nurture LLMs to be Quantization Friendly



Nurture LLMs to be Quantization Friendly

Model Size	Data type	PIQA	HellaSwag	WinoGrande	LAMBADA	Copa	Copa100	StoryCloze	Paralex	Average
52B	FP16	83.19	82.48	70.01	75.47	79.40	81.00	85.87	61.05	77.31
	W8A8	83.20	82.40	70.00	75.50	79.40	82.00	85.50	61.10	**77.39**
	W4	80.20	72.22	66.30	66.85	78.20	83.00	82.56	60.43	73.72
13B	FP16	79.54	75.26	62.27	70.81	76.00	76.00	82.11	60.68	72.83
	W8A8	79.20	74.60	62.90	69.90	76.00	75.00	82.20	60.90	**72.59**
	W4	76.66	60.59	57.30	46.05	73.60	76.00	77.40	59.74	65.92
6B	FP16	79.50	74.20	61.20	70.50	75.40	79.00	81.50	60.10	72.67
	W8A8	79.50	73.70	61.40	70.00	74.60	77.00	81.00	60.20	**72.18**
	W4	76.93	62.92	56.43	55.40	74.00	72.00	77.21	59.01	66.74
410M	FP16	70.40	46.90	50.80	48.80	65.40	65.00	70.50	57.10	59.36
	W8A8	70.00	46.80	51.50	47.80	64.00	64.00	69.70	57.00	**58.85**
	W4	67.19	43.08	50.59	37.71	62.80	64.00	67.47	54.60	55.93

Table 4: Our fully trained models with hyper-parameters outlined in Table 2 show minimal PTQ degradation.

Final Takeaways

- Outliers at scale are due to nurture rather than nature
- Train with **bf16**, gradient clipping, higher weight decay, and low dropout
- Vectorwise INT8 quantization at scale is feasible

Contact:

saurabh@cohere.com

Scan for Paper

Cohere For AI

Exploring the unknown, together

Web: cohere.for.ai

Twitter: @forai_ml