Institute of Electronics
National Yang Ming Chiao Tung University
Hsinchu, Taiwan

AI Training Course Series

Model Compression: Quantization + Pruning + Weight-Sharing

Lecture 12

Architecture Development & Algorithm Refinement for Intelligent Computing

Student: Ting-Wei Hu
Advisor: Juinn-Dar Huang, Ph.D.

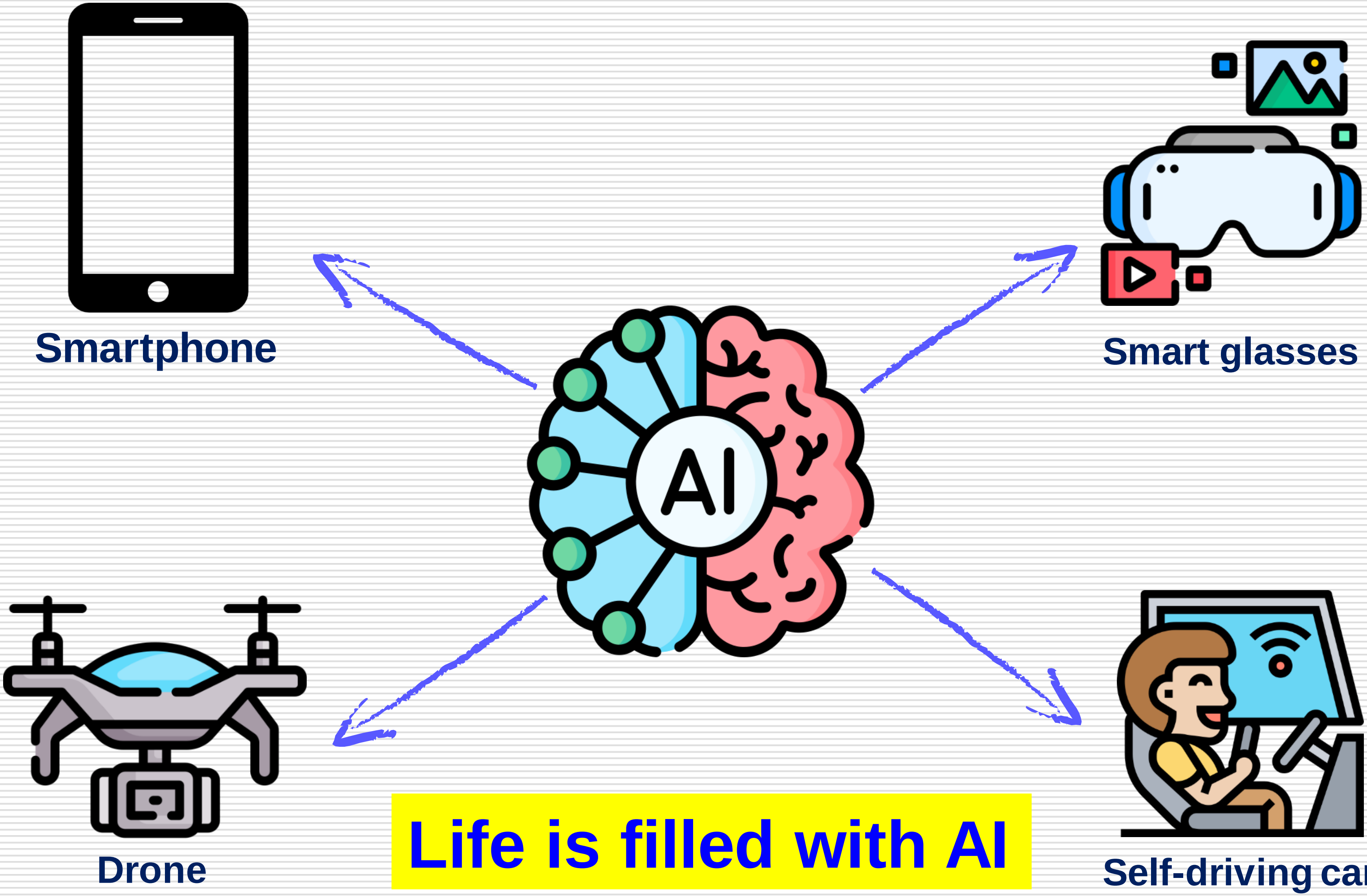
Aug 22, 2024

Outline

- Necessity of model compression
- Pruning
- Weight-sharing
- Low-rank approximation
- Quantization concepts
- Uniform integer quantization
- Survey of LLM quantization
- PFPQuant
- Homework

Necessity of Model Compression

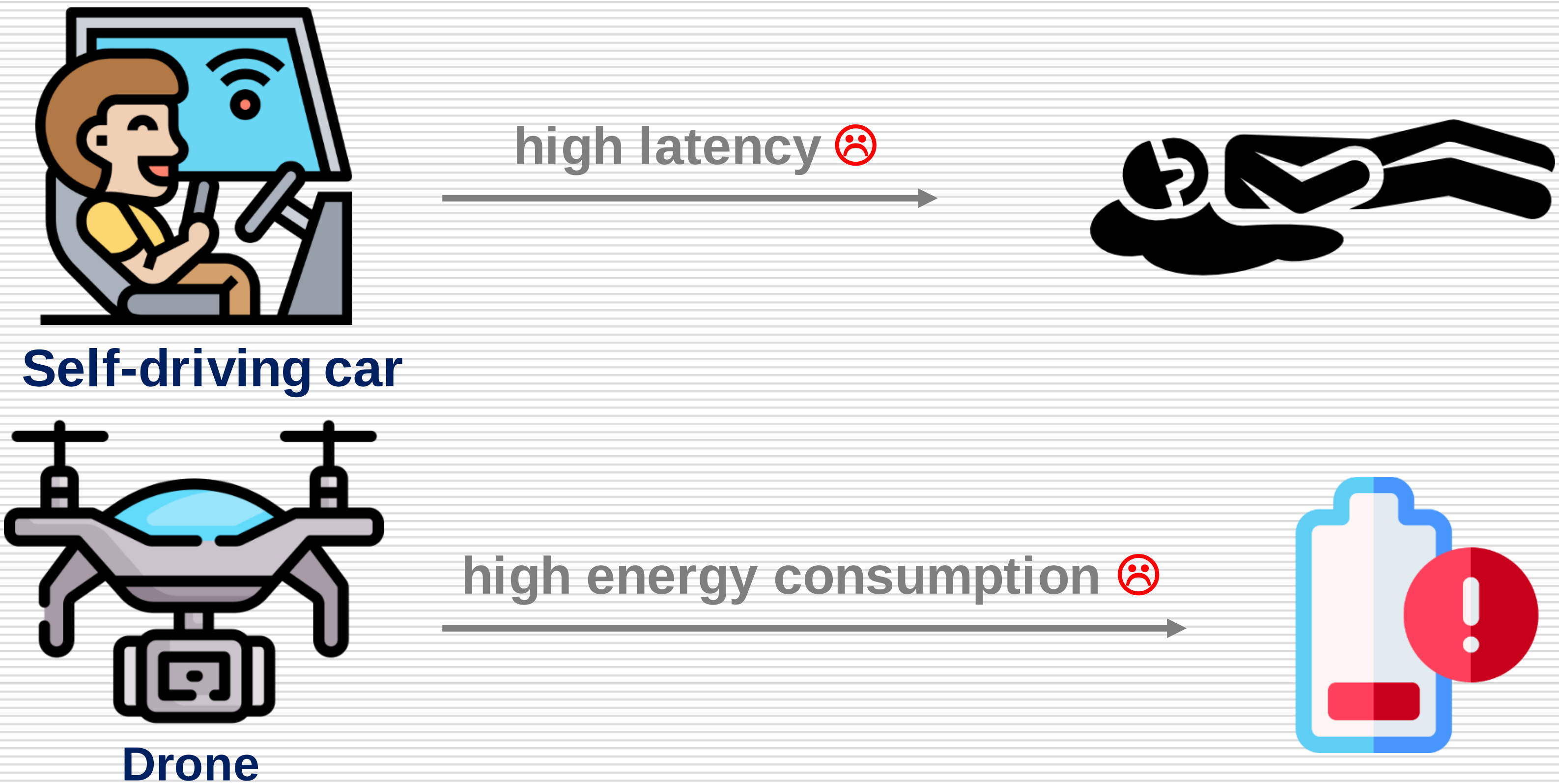
AI Applications (1/2)



Life is filled with AI

AI Applications (2/2)

- Requirements for the common applications
  - Low latency
  - Energy efficient

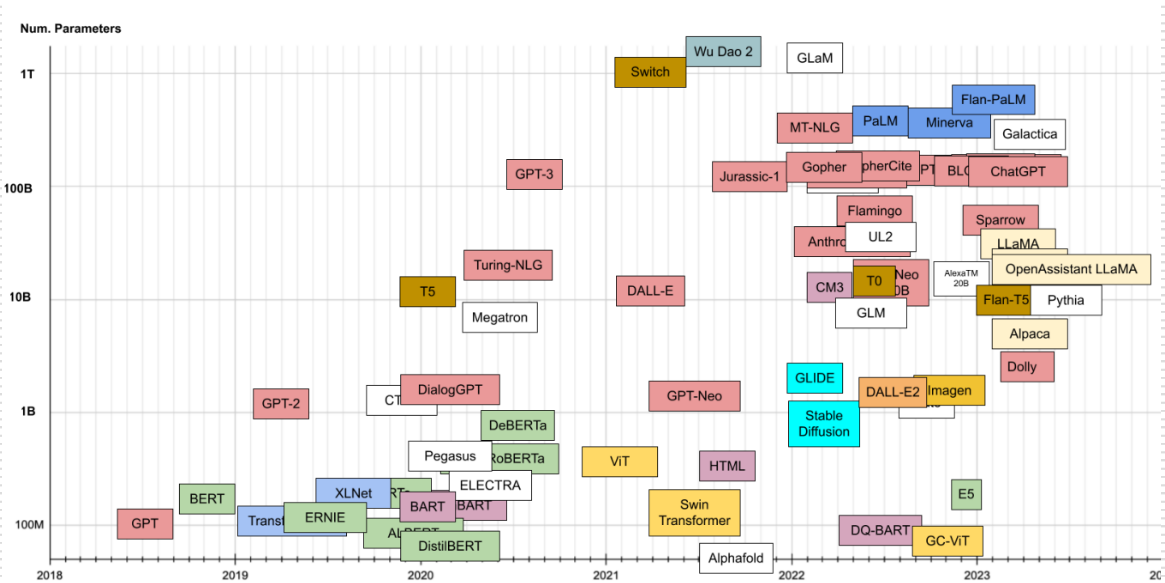


Booming AI Models

- Recent deep learning models are **complex**
  - LeNet-5: 1M parametes (1998)
  - VGG-16: 133M parameters (2014)
  - CoCa: 2100M parameters (2022)

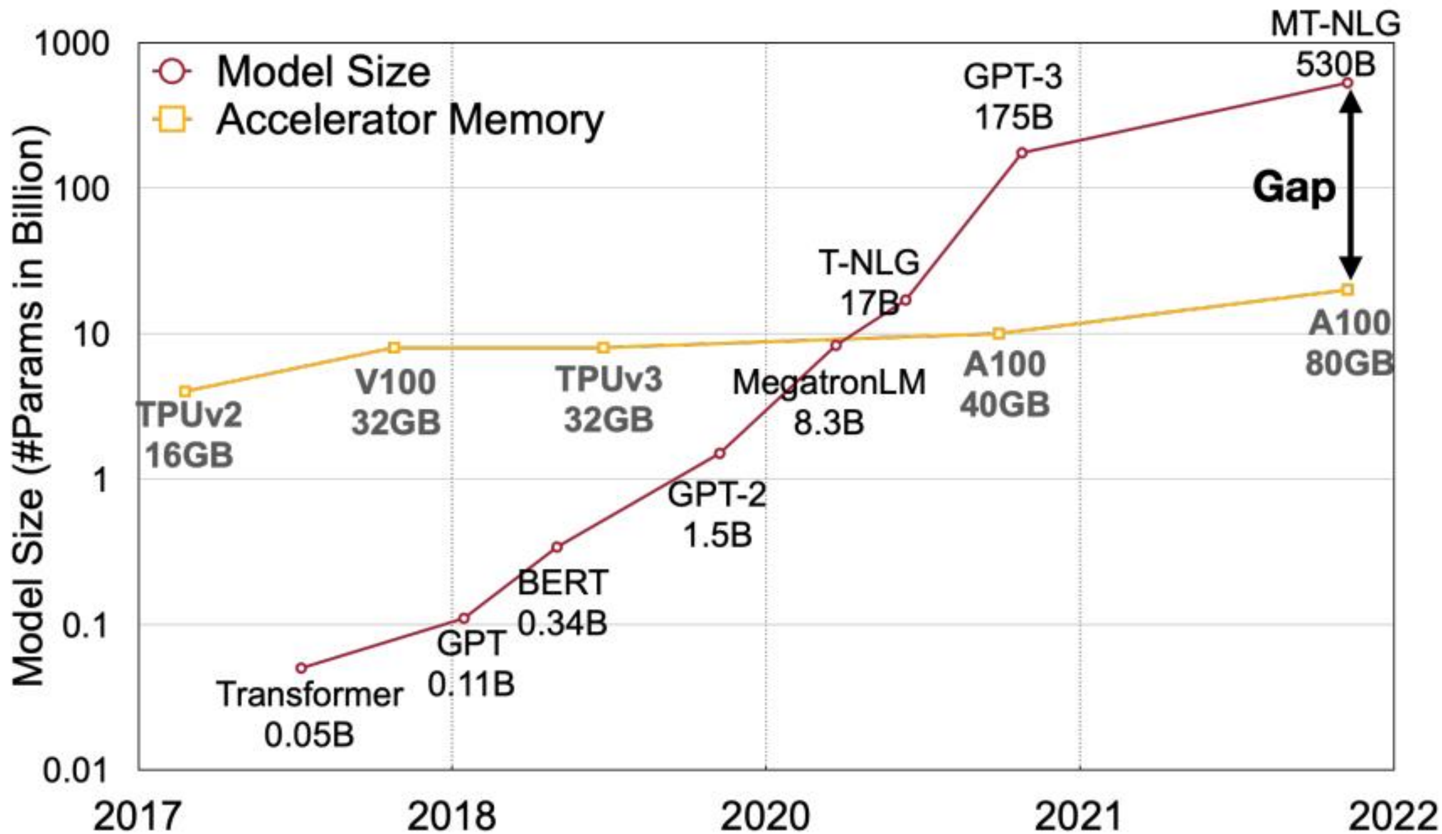
83× ↓ **Emergence of LLMs**

  - ChatGPT: **175B** parameters (2023) extremely huge!



Computational Issue (1/2)

- The model size is developing at a faster pace than the GPU memory in recent years



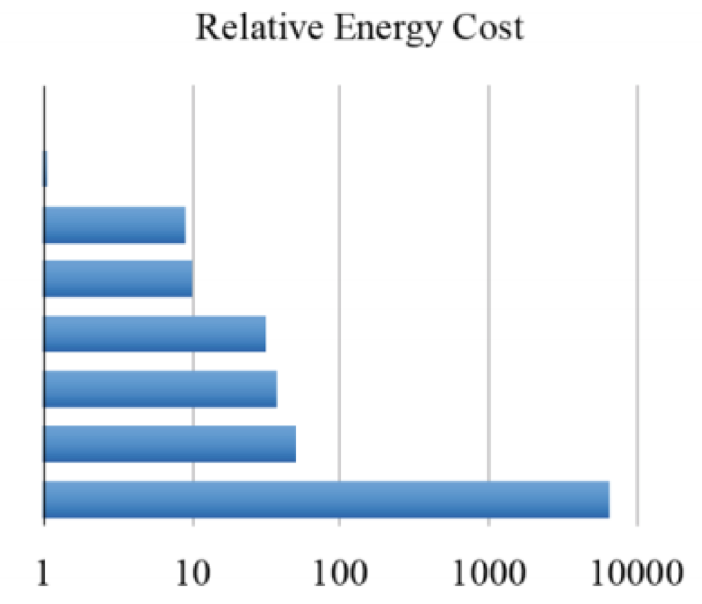
Computational Issue (2/2)

- LLMs are compute and memory-intensive
- Swin Transformer model in computer vision ➔ Swin-L **197M** parameters
- ChatGPT ➔ **175B** parameters consuming at least 350GB memory to store in FP16
  – 5 X 80GB A100 GPUs for inference
  – Huge computation and communication overhead
  – Almost **1000**× increase compared with Swin-L

Energy Issue

- DRAM results in significant energy consumption
- Huge models usually access data from DRAM
  - More data transfer, more energy consumption

Operation	Energy [pJ]	Relative Cost
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit Register File	1	10
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit SRAM Cache	5	50
32 bit DRAM Memory	**640**	**6400**



Energy consumption increases dramatically

Why Model Compression

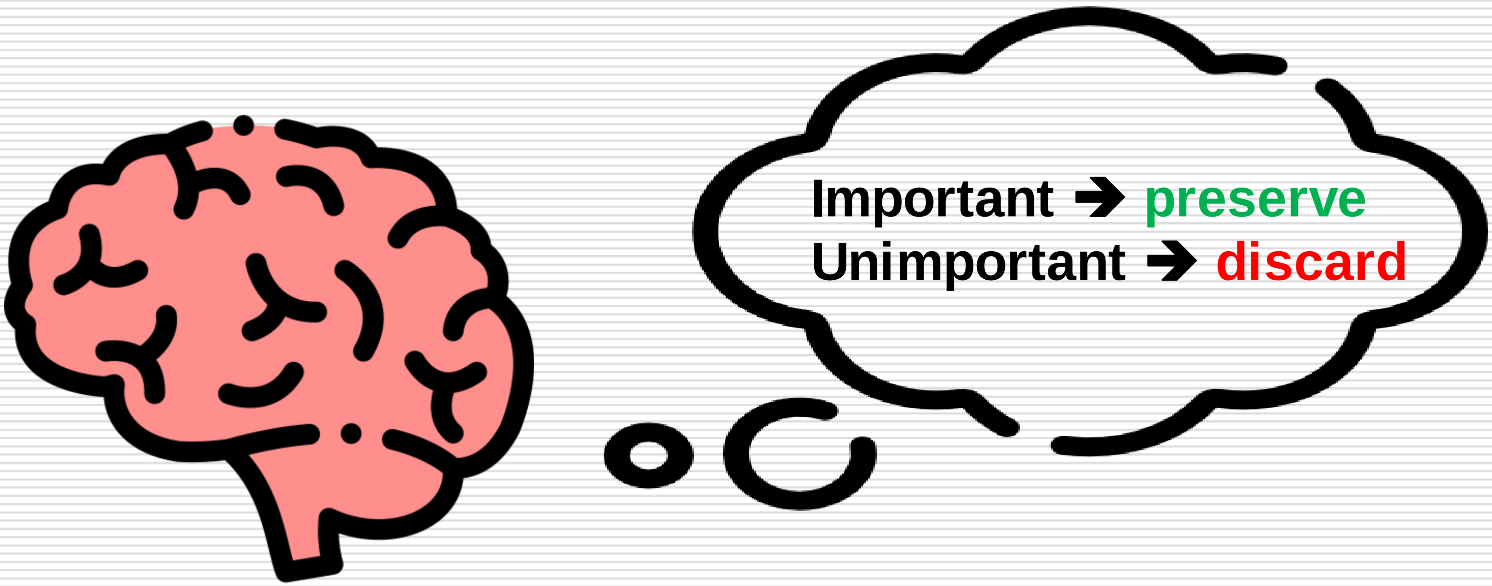
- Challenges of complex deep models
  - Huge storage (memory, disk) requirement
  - Computationally expensive
  - Consume lots of energy
  - Hard to deploy models on small edge devices

Compression becomes more significant !

Pruning

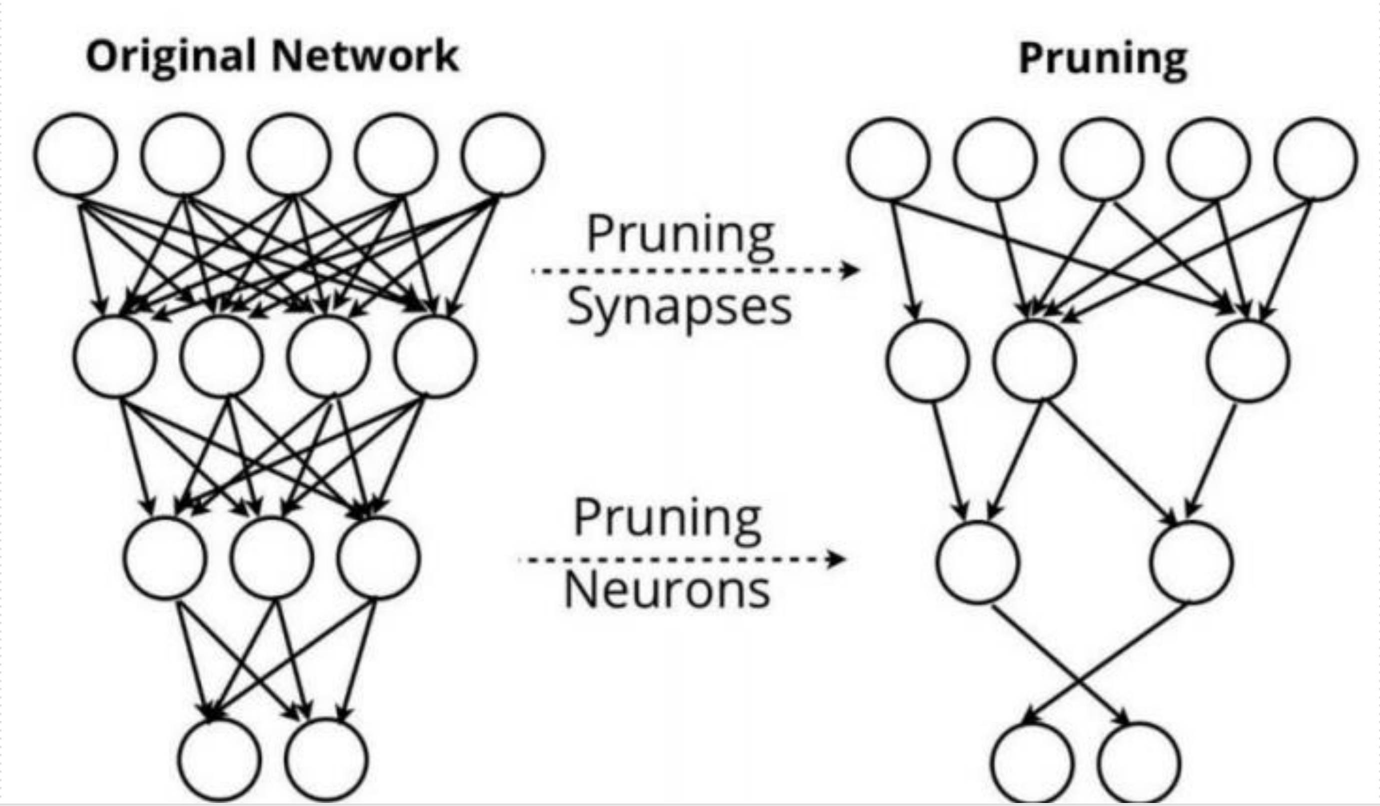
Introduction

- Motivated by how real brain learns
  - Usefulness in, waste out
- Propose **pruning** technique discarding relatively unimportant weights to ahieve sparse model
  - Remove redundant connections



How Pruning Works (1/2)

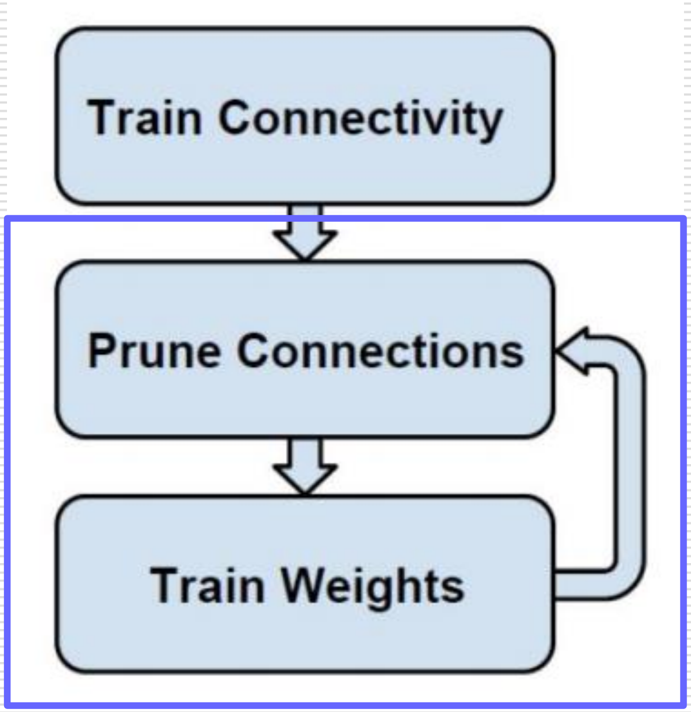
- Threshold pruning
  - Set a parameter as threshold
  - Remove weights whose abs(weight) < threshold



dense sparse

How Pruning Works (2/2)

- Retrain after threshold pruning
  - Reduce accuracy drop
  - Learn effective connections by iterative pruning



repeat the two steps iteratively
To achieve gradual pruning

Retraining Strategies

- After pruning, the performance loss should be compensated by retraining
  - Prune weights of all layers once then retrain few epochs, then repeat this procedure until certain degree of accuracy is restored
    - Faster
    - Lower accuracy
  - Prune weights layer by layer then retrain iteratively, the model is retrained before pruning the next layer
    - Slower
    - Higher accuracy

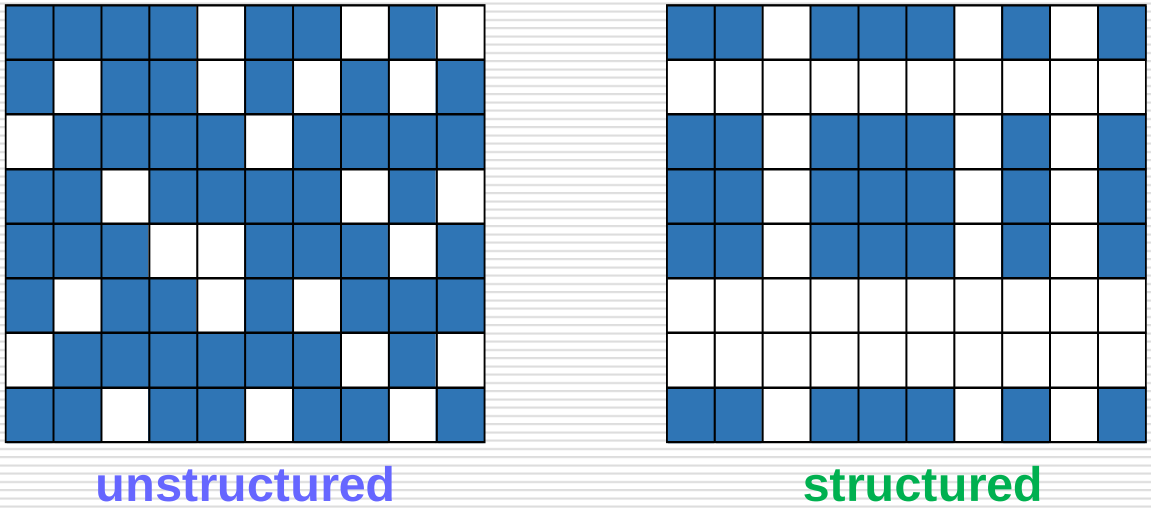
Model Compression Rate

- Result of threshold pruning
  - Experimented with LeNet, AlexNet, VGG (ImageNet)
  - About **10x** smaller compared to origin network
  - Mostly contributed by FC layers
  - Less than 1% accuracy loss

Network	Top-1 Error	Top-5 Error	Parameters	Compression Rate
LeNet-300-100 Ref	1.64%	-	267K	
LeNet-300-100 Pruned	1.59% 3.0%↓	-	**22K**	**12**×
LeNet-5 Ref	0.80%	-	431K	
LeNet-5 Pruned	0.77% 3.7%↓	-	**36K**	**12**×
AlexNet Ref	42.78%	19.73%	61M	
AlexNet Pruned	42.77% 0.2%↓	19.67%	**6.7M**	**9**×
VGG-16 Ref	31.50%	11.32%	138M	
VGG-16 Pruned	31.34% 5.0%↓	10.88%	**10.3M**	**13**×

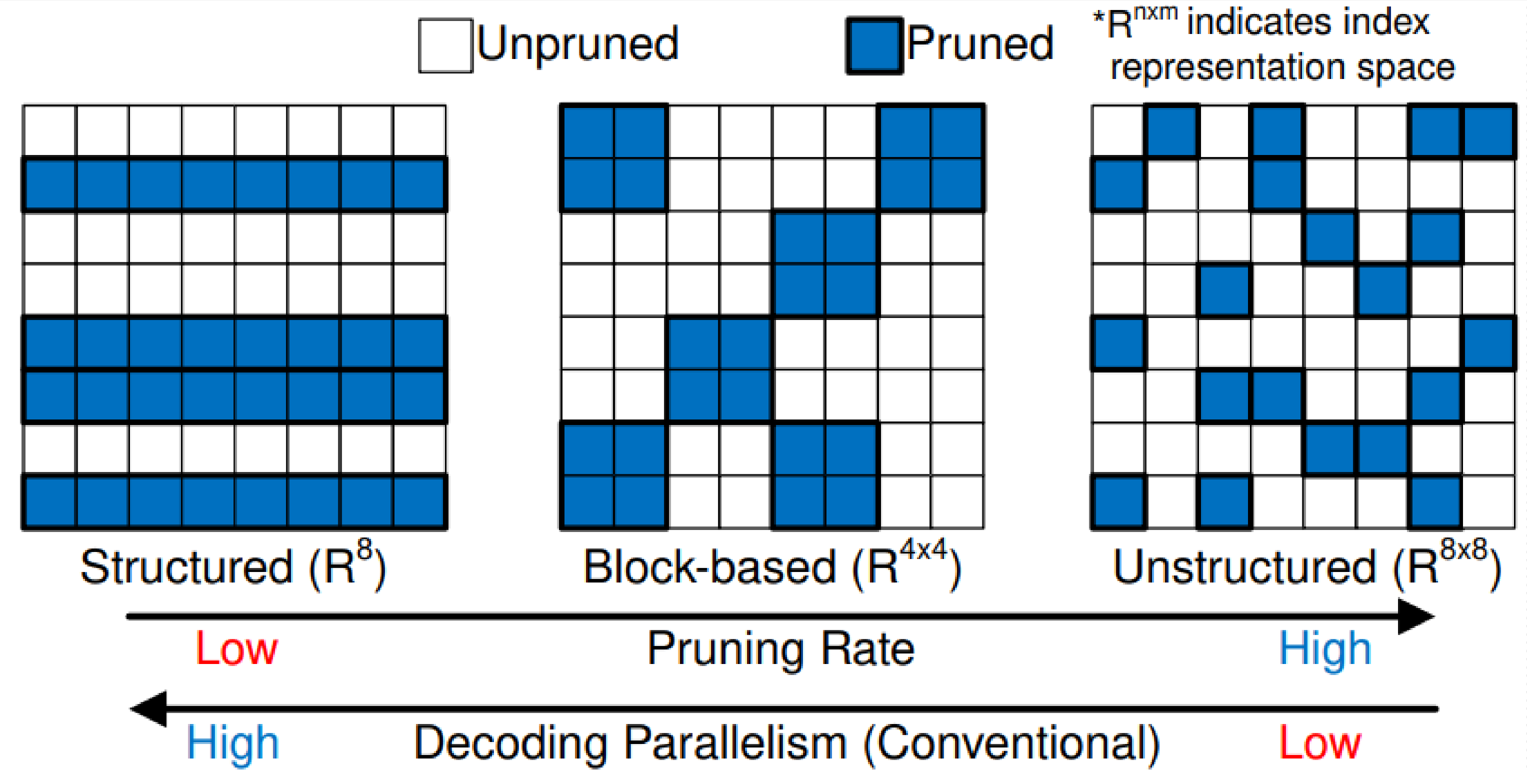
Unstructured Pruning

- Unstructured pruning
  - Better accuracy
  - Needs extra memory to store index
  - Extremely hard to keep parallelism in hardware acceleration (i.e., you'd better forget about it!)



Structured Pruning

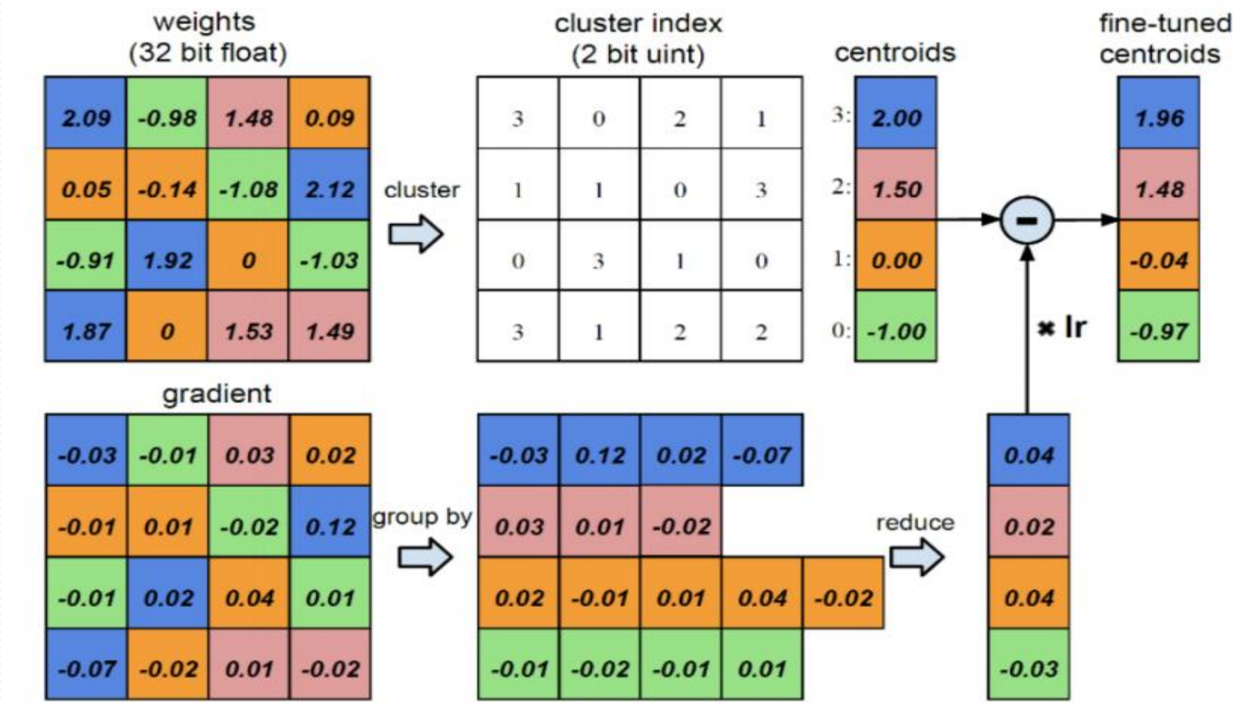
- Structured pruning
  - Maintains the regularity of the weight matrix
  - Accuracy degradation due to larger information loss
  - Eliminates the overhead and accelerates computation



Weight-Sharing

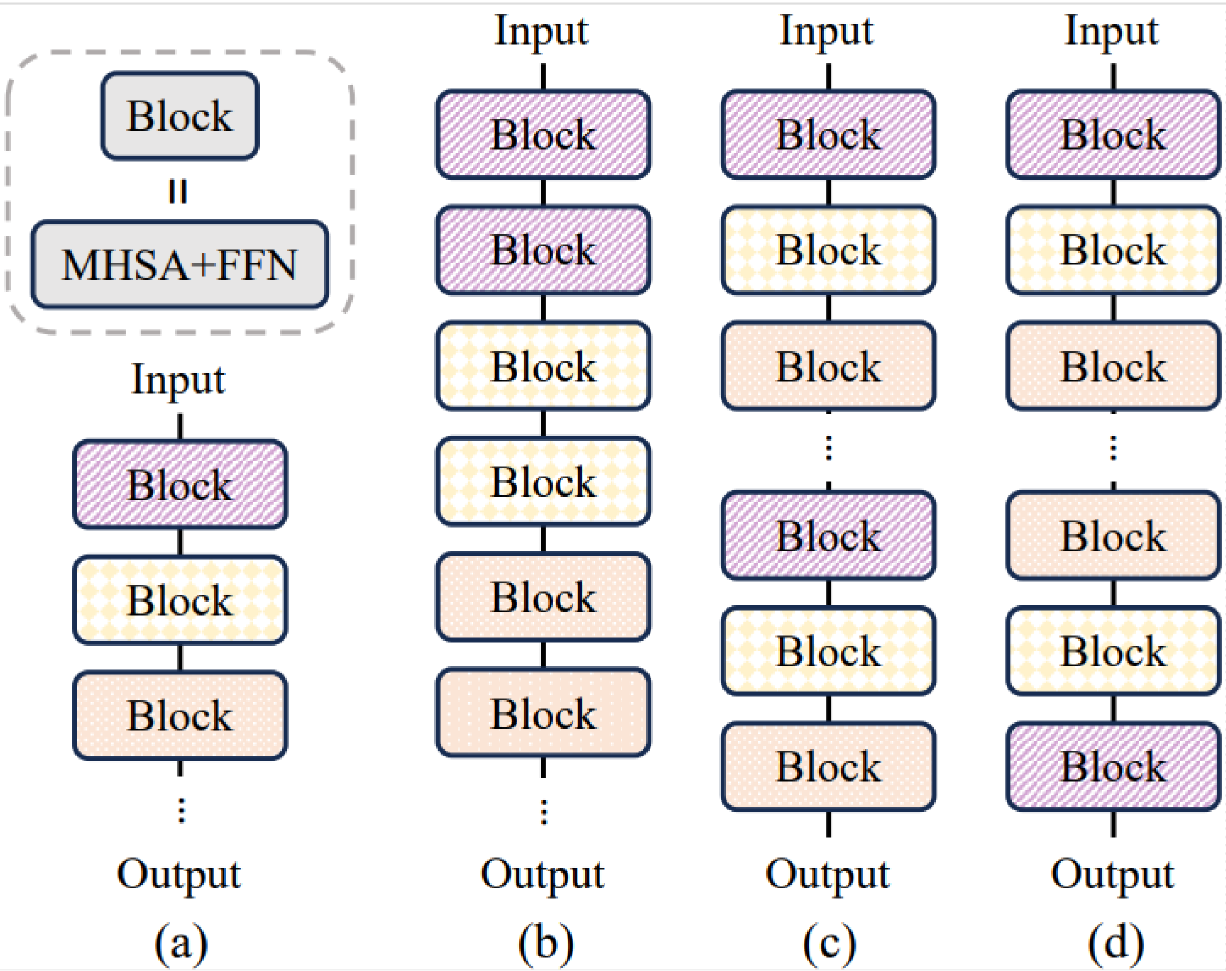
Weight-Sharing By Clustering

- Weight-sharing by clustering
  - Cluster weights and use centroid in the indexed array
    - E.g., k-means
  - Fine-tune the centroid weights with a summation of corresponding gradients



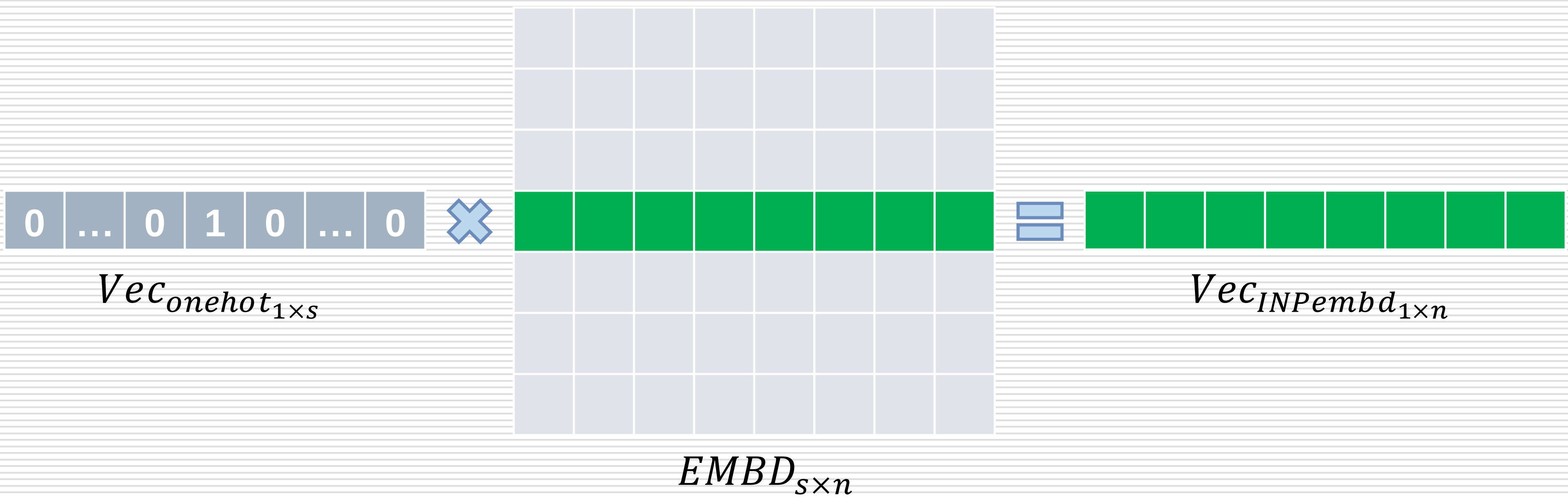
Weight-Sharing in Language Models (1/3)

- Scaling LMs to achieve better performance
  - Layer-sharing
    - MobileLLM



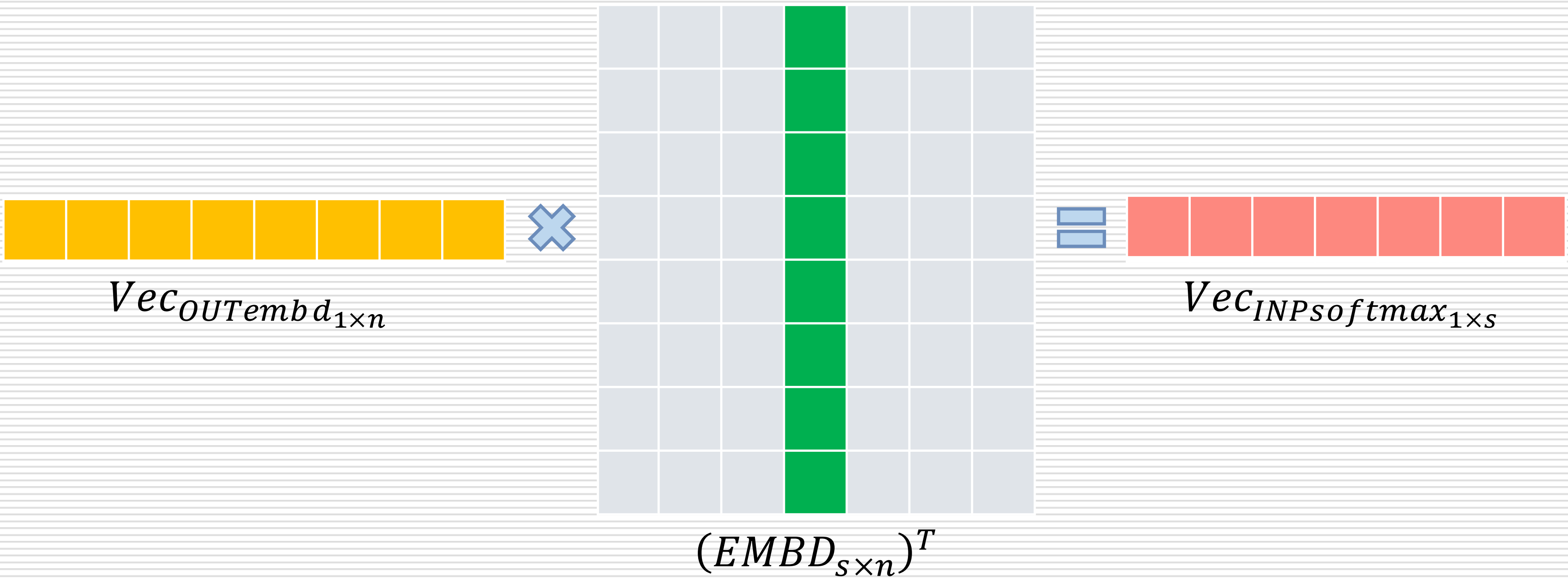
Weight-Sharing in Language Models (2/3)

- Embedding-sharing



Weight-Sharing in Language Models (3/3)

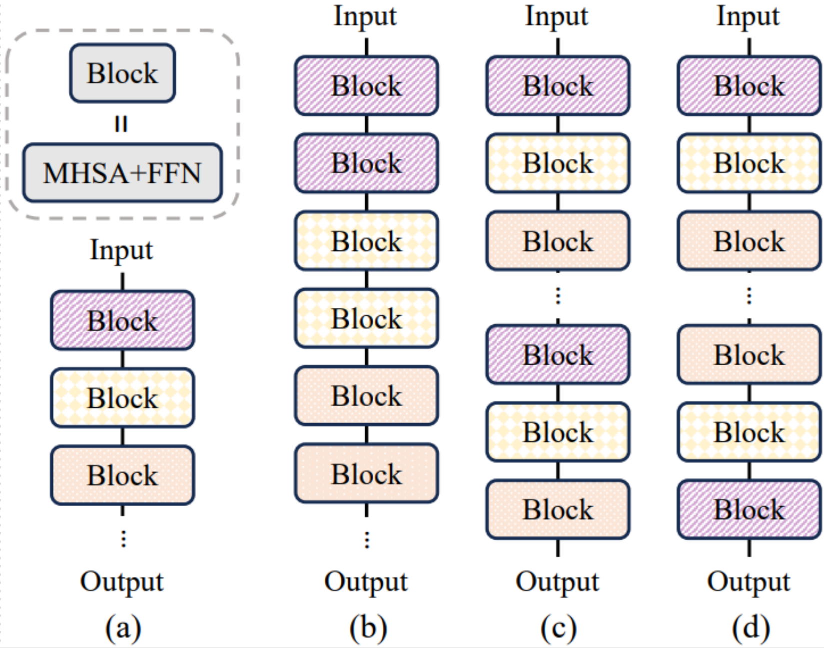
- Embedding-sharing



Experiment Results in MobileLLM

Table 2: Ablation study of layer-sharing strategy on zero-shot common sense reasoning tasks.

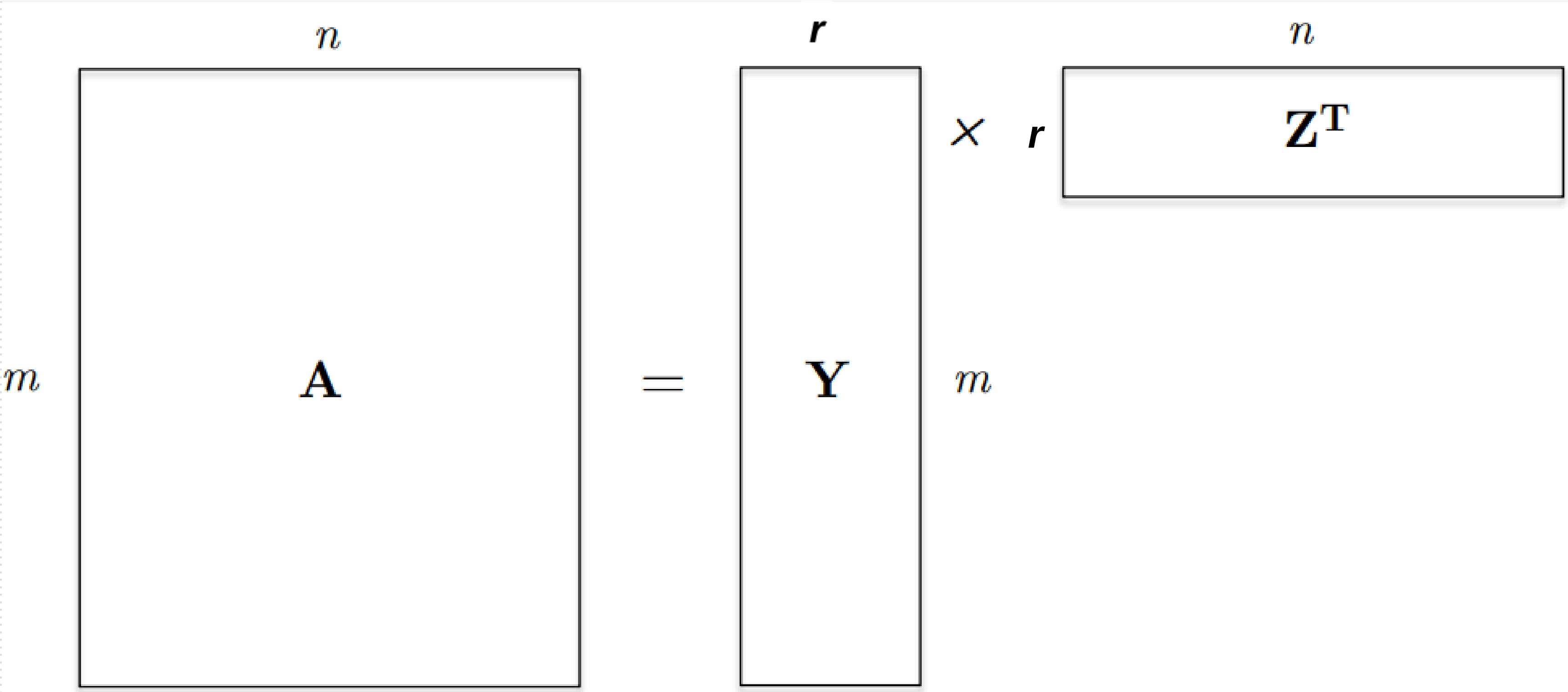
Model	Sharing method	ARC-e	ARC-c	BoolQ	PIQA	SIQA	HellaSwag	OBQA	WinoGrande	Avg.
125M	baseline	41.6	25.7	61.1	62.4	43.1	34.4	36.9	51.6	44.6
	Immediate block-wise share	43.9	27.9	61.5	64.3	41.5	35.5	35.1	50.2	45.0
	Repeat-all-over share	43.6	27.1	60.7	63.4	42.6	35.5	36.9	51.7	**45.2**
	Reverse share	43.8	26.0	58.9	62.9	42.2	35.2	36.8	52.2	44.8
350M	baseline	50.8	30.6	62.3	68.6	43.5	45.1	43.8	52.4	49.6
	Immediate block-wise share	51.5	30.8	59.6	68.2	43.9	47.7	44.7	55.0	50.2
	Repeat-all-over share	53.5	33.0	61.2	69.4	43.2	48.3	42.2	54.6	**50.7**
	Reverse share	50.7	32.2	61.0	68.8	43.8	47.4	43.1	53.8	50.1



Low-Rank Approximation

Matrix Decomposition

- Any matrix A of rank-r can be decomposed into a long and skinny matrix times a short and long one



Singular Value Decomposition (SVD)

- ***A*** : an m × n matrix of rank-r
- ***U*** : an m × m orthogonal matrix
- ***V*** : an n × n orthogonal matrix
- ***S*** : an m × n diagonal matrix with nonnegative entries, and with the diagonal entries sorted from high to low

$$A = USV^T$$

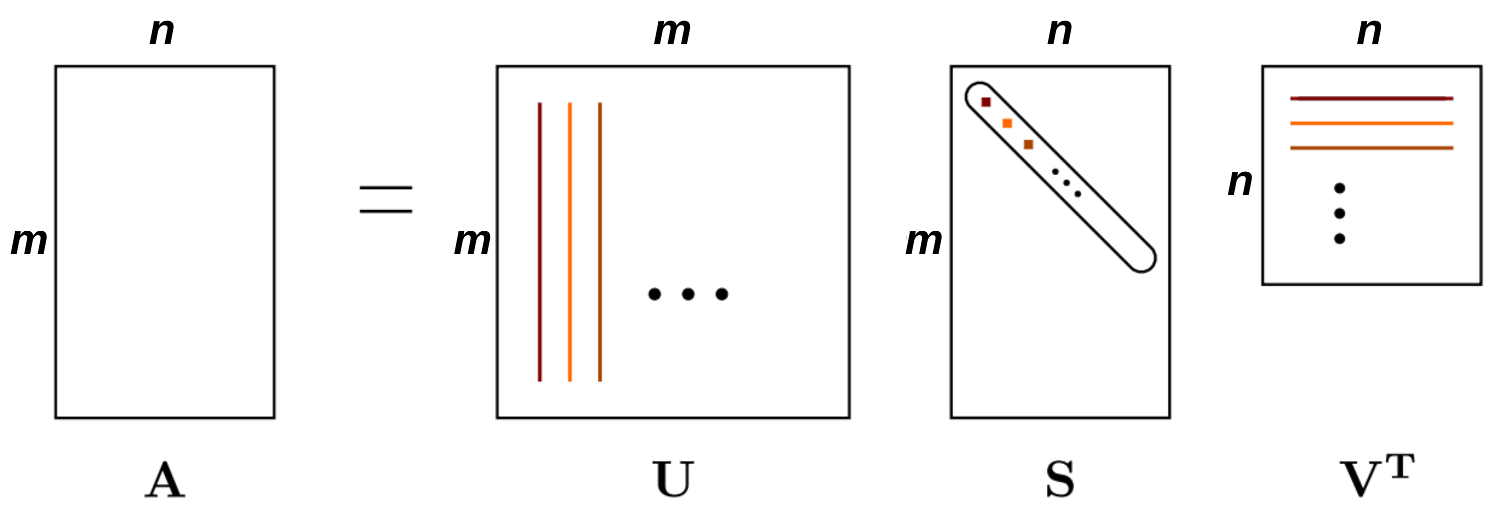


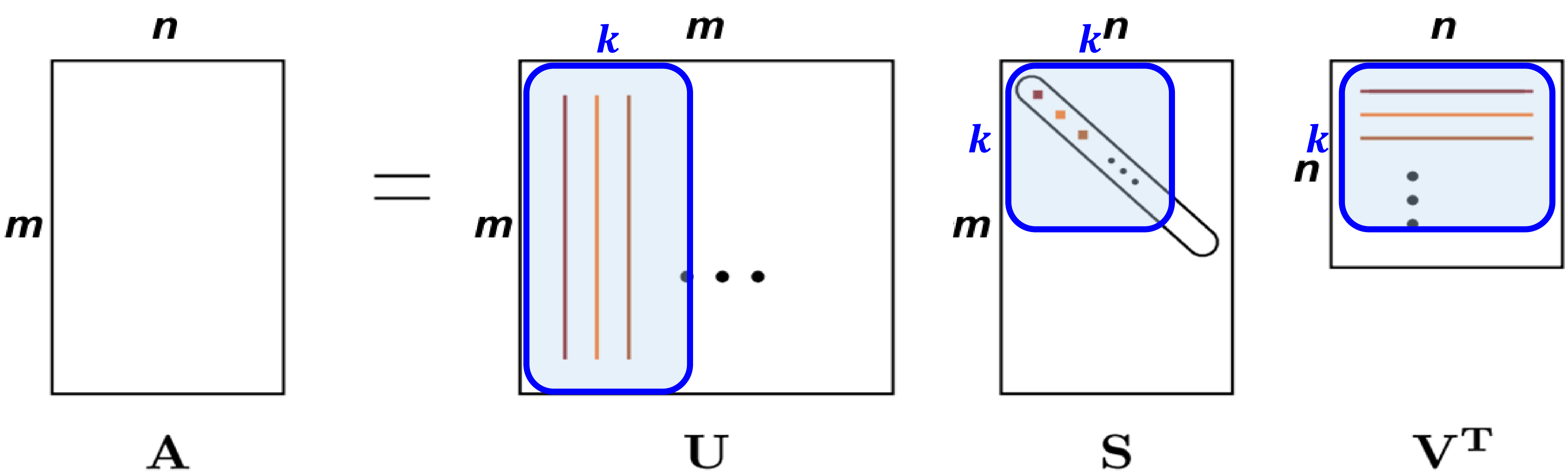
Figure 2: The singular value decomposition (SVD). Each singular value in **S** has an associated left singular vector in **U**, and right singular vector in **V**.

Low-Rank Approximation with SVD (1/2)

- Rank-k approximation, $k \leq r$

$$A_k = U_k S_k V_k^T$$

  - Compute the SVD
  - Set U_k equal to the first k columns of U (m $\times$ k)
  - Set S_k equal to the first k rows and columns of S (k $\times$ k)
  - Set V_k^T equal to the first k rows of V^T (k $\times$ n)



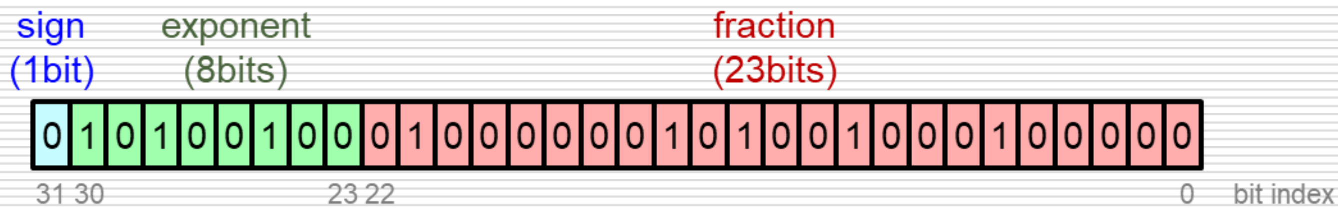
Low-Rank Approximation with SVD (2/2)

- Get $A_k = U_k S_k V_k^T$
- Fuse S_k in U_k or V_k^T
  - $A_k = U_k' V_k^T \ or \ U_k V_k^{T\prime}$
- # of parameters
  - Original
    - $m \times n$
  - After low-rank approximation
    - $m \times k + k \times n$

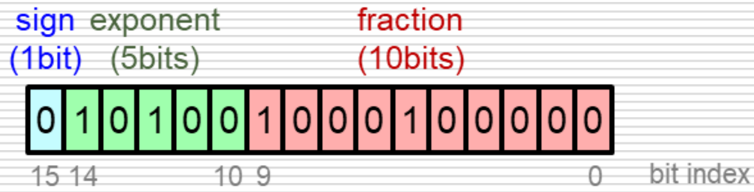
Quantization Concepts

Common Data Types (1/2)

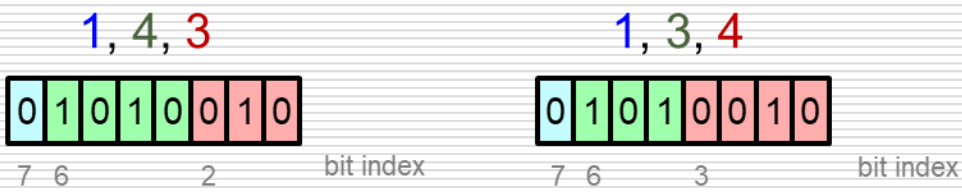
- In neural network
  - Parameters are usually stored in 16-bit or 32-bit precision
  - Single-precision IEEE 754 (FP32)



  - Floating point format with lower bits
    - FP16

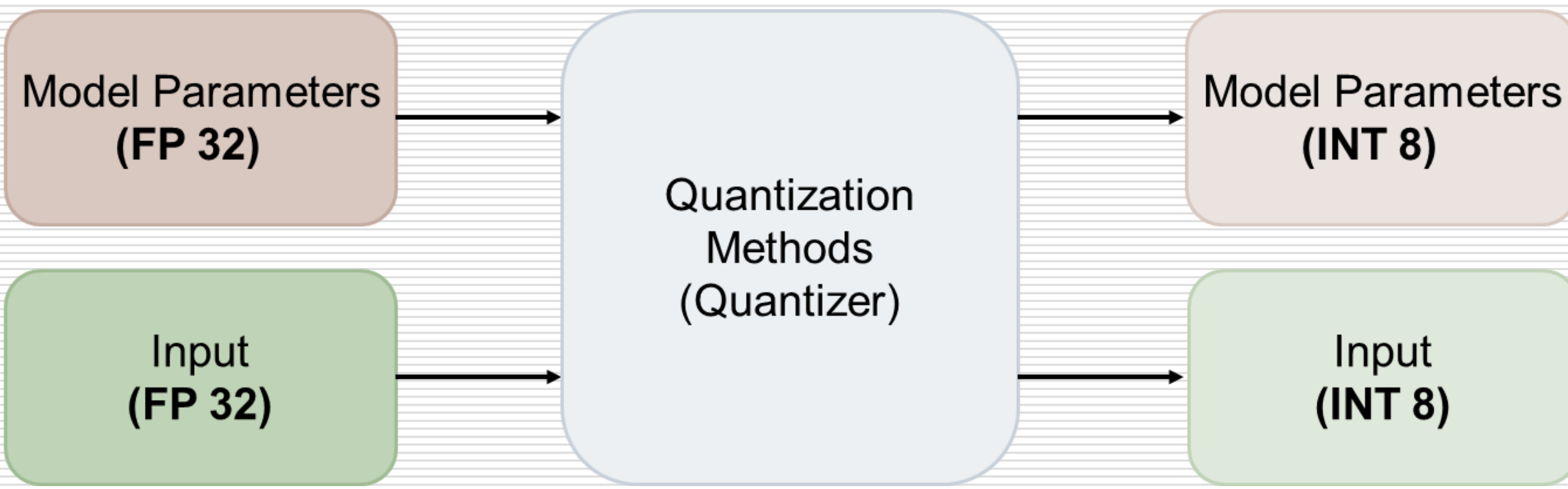


    - FP8



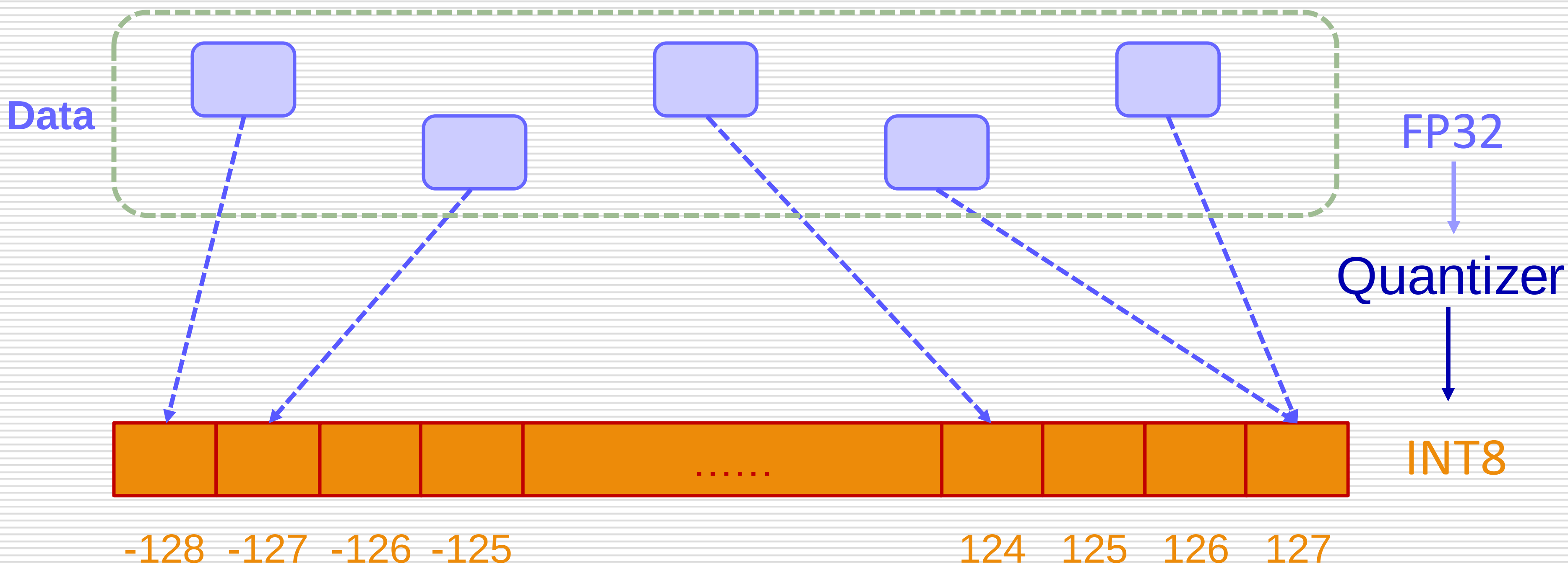
Common Data Types (2/2)

- When quantize higher bits to lower bits
  - Lower computation overhead ☺
  - Reduce memory usage ☺
  - Lower power consumption ☺
  - Quantization error→ loss of precision ☹
- FP32→INT8



Example: FP32 to INT8

- Aim to move values from **FP32** to **INT8**
- Quantizer needs to allocate all FP32 values into 256 states
  - 8 bits can represent 256 states



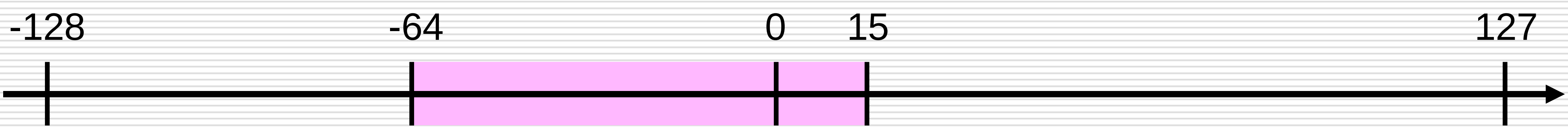
Uniform Integer Quantization

Basics of N-Bit Uniform Integer Quantization

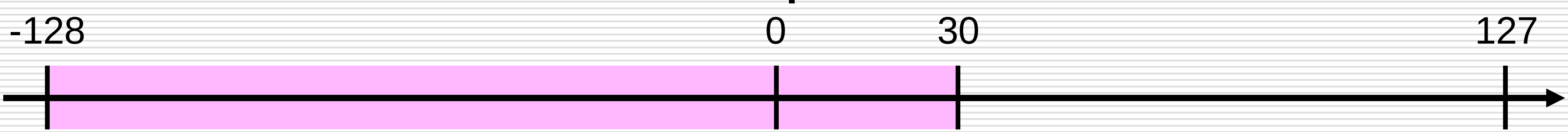
- Formula

$$X^{INT_N} = \left\lfloor \frac{X^{FP}}{s} \right\rceil + z, \qquad X^{FP} \approx s(X^{INT_N} - z)$$

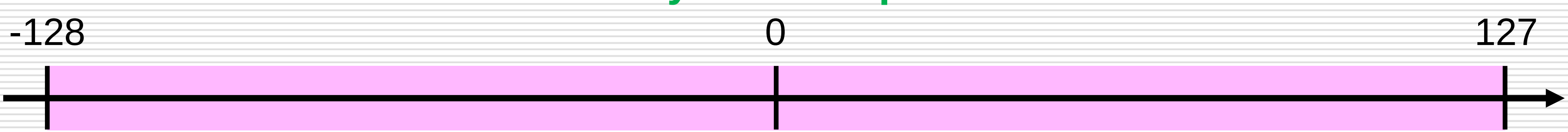
- Example of 8-bit quantization



Data before quantization



Data after symmetric quantization

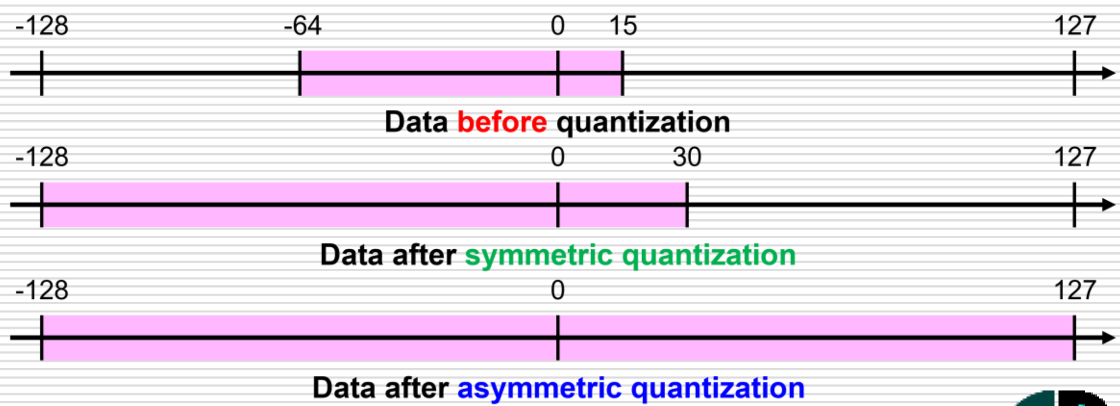


Data after asymmetric quantization

Symmetric & Asymmetric Quantization

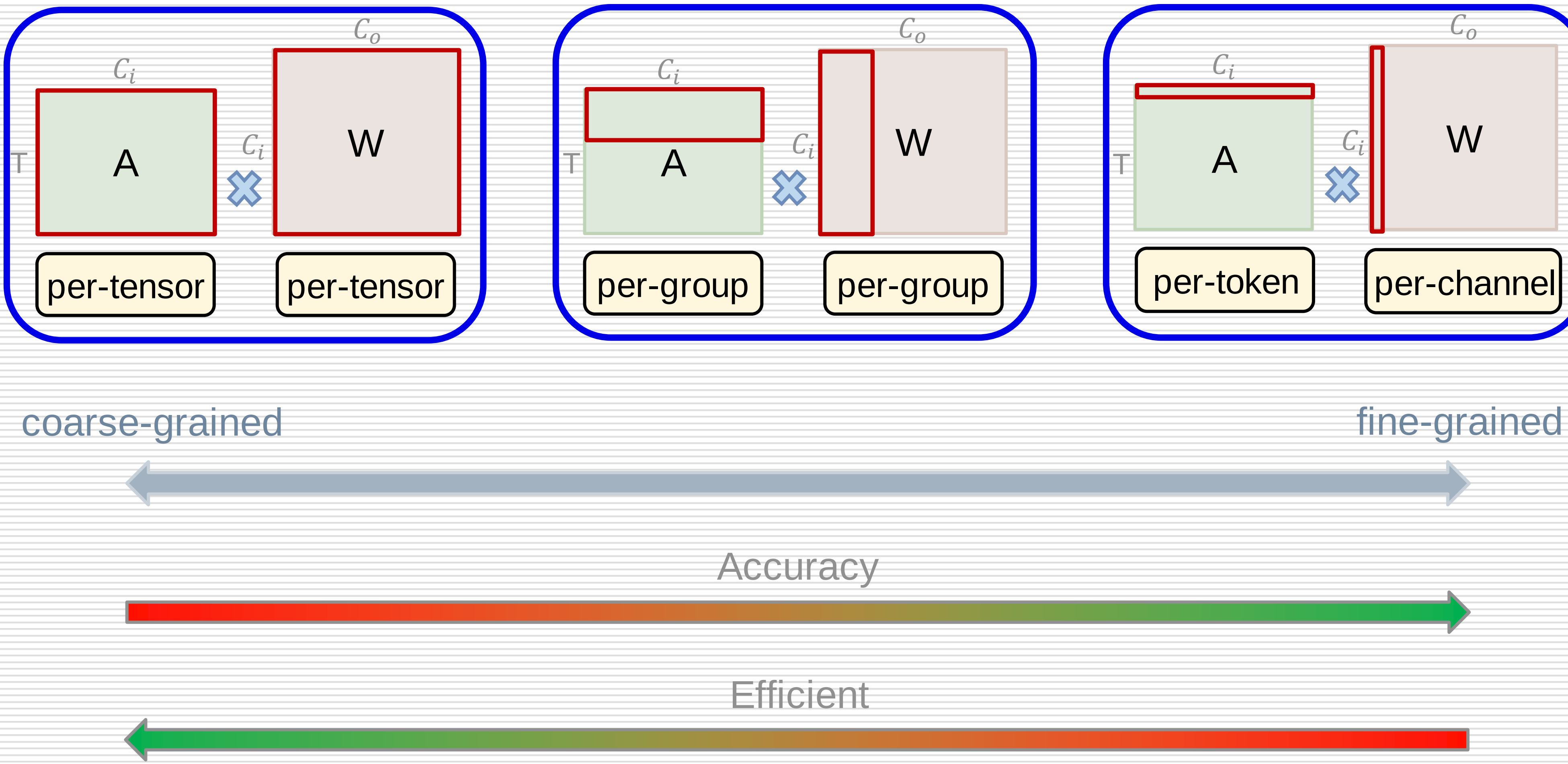
s: scale
z: zero point
N: target number of bits

- Formula
  - $X^{INT_N} = \left\lfloor \frac{X^{FP}}{s} \right\rceil + z$
  - $X^{FP} \approx s(X^{INT_N} - z)$ (dequantize)
  - $a = -2^{N-1},\ b = 2^{N-1} - 1$
- **Symmetric** quantization
  - $s_{sym} = \frac{max(|X^{FP}|)}{b-a}, \qquad z_{sym} = 0$
- **Asymmetric** quantization
  - $s_{asym} = \frac{\lfloor max(X^{FP}) - min(X^{FP}) \rceil}{b-a}$
  - $z_{tmp} = a - \frac{min(X^{FP})}{s_{asym}}$
  - $z_{asym} = clamp\{z_{tmp},\ a,\ b\}$



Quantization Granularity

Scale Calibration Range



Per-tensor Quantization for Matrix Operation

$$I^{INT} = \left\lfloor \frac{I^{FP}}{s_i} \right\rceil + z_i, \qquad \overline{I^{FP}} = s_i\left(I^{INT} - z_i\right)$$

$$W^{INT} = \left\lfloor \frac{W^{FP}}{s_w} \right\rceil + z_w, \qquad \overline{W^{FP}} = s_w\left(W^{INT} - z_w\right)$$

$$B^{INT} = \left\lfloor \frac{B^{FP}}{s_i \cdot s_w} \right\rceil, \qquad \overline{B^{FP}} = s_i s_w \cdot B^{INT}$$

$$O = I \cdot W + B \approx \overline{O^{FP}} = \overline{I^{FP}} \cdot \overline{W^{FP}} + \overline{B^{FP}}, \qquad O^{INT} = \left\lfloor \frac{\overline{O^{FP}}}{s_o} \right\rceil + z_o$$

$$O^{INT} = \left\lfloor \frac{\sum s_i\left(I^{INT} - z_i\right) s_w\left(W^{INT} - z_w\right) + s_i s_w \cdot B^{INT}}{s_o} \right\rceil + z_o = \left\lfloor \frac{s_i s_w}{s_o}\left\lfloor \sum \left(I^{INT} - z_i\right)\left(W^{INT} - z_w\right) + B^{INT} \right\rceil \right\rceil + z_o$$

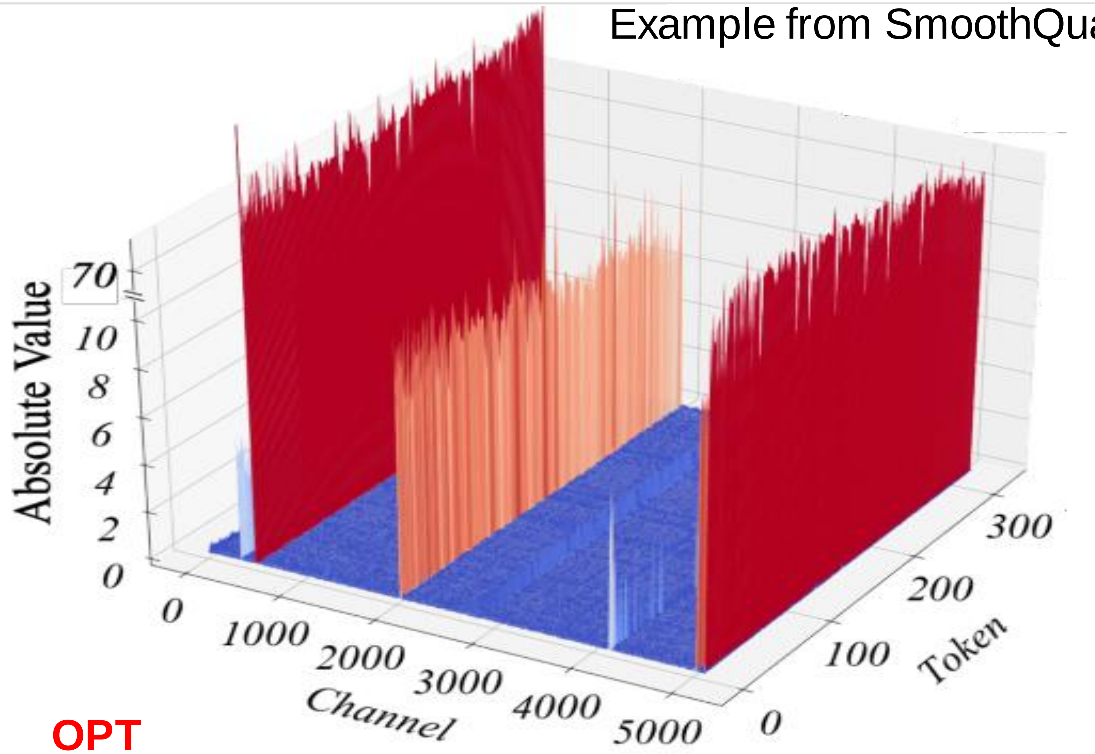
$$\frac{s_i s_w}{s_o} \approx M^{INT} \cdot 2^{shift}$$

Survey of LLM Quantization

Quantization Difficulty of LLMs (1/2)

- Outliers in OPT (SmoothQuant)

Example from SmoothQuant

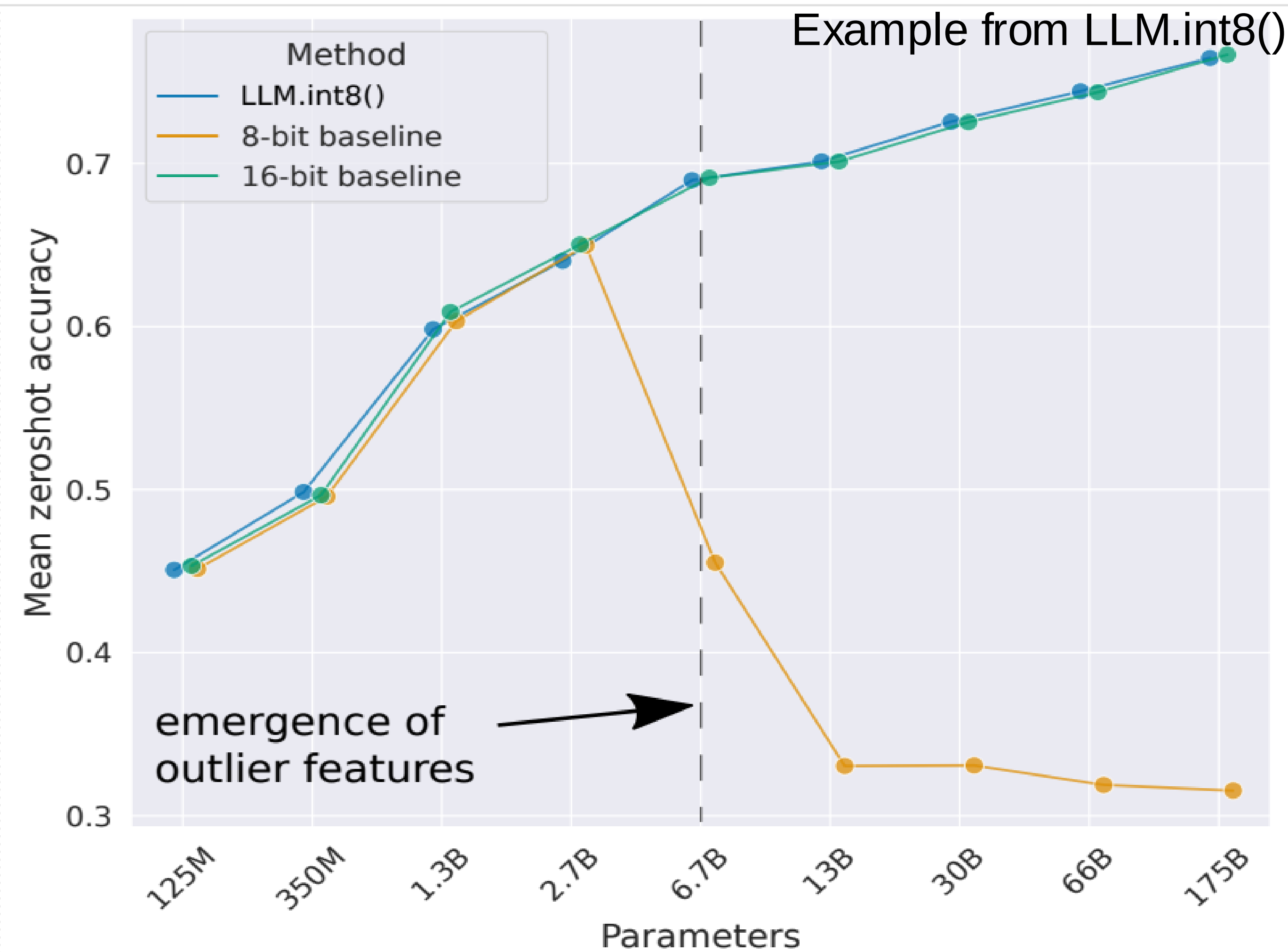


OPT

Model	#L	#H	d_{model}	LR	Batch
13B	40	40	5120	$1.0e{-}4$	4M

Quantization Difficulty of LLMs (2/2)

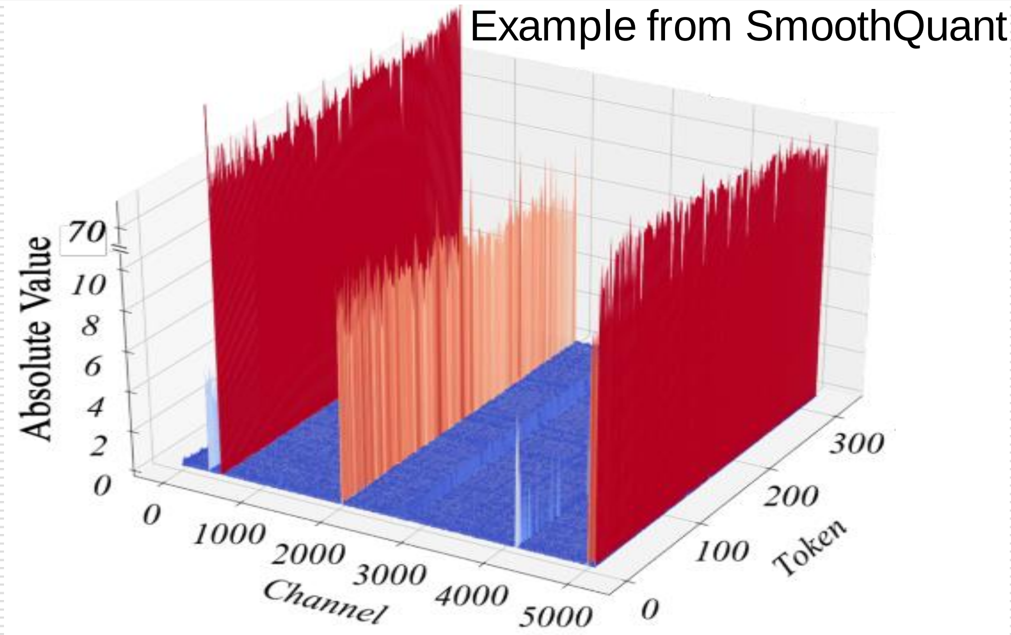
- Apply the vanilla 8-bit quantization on LLM



LLM.int8(): The Fact of Outliers (1/3)

- These outlier features are highly systematic after the emergence of outlier features occurs
  - EX：6.7B transformer with a sequence length of 2048
    - About 150k outlier features per sequence for the entire transformer, but these features are concentrated in only 7 different hidden dimensions

Example from SmoothQuant



LLM.int8(): The Fact of Outliers (2/3)

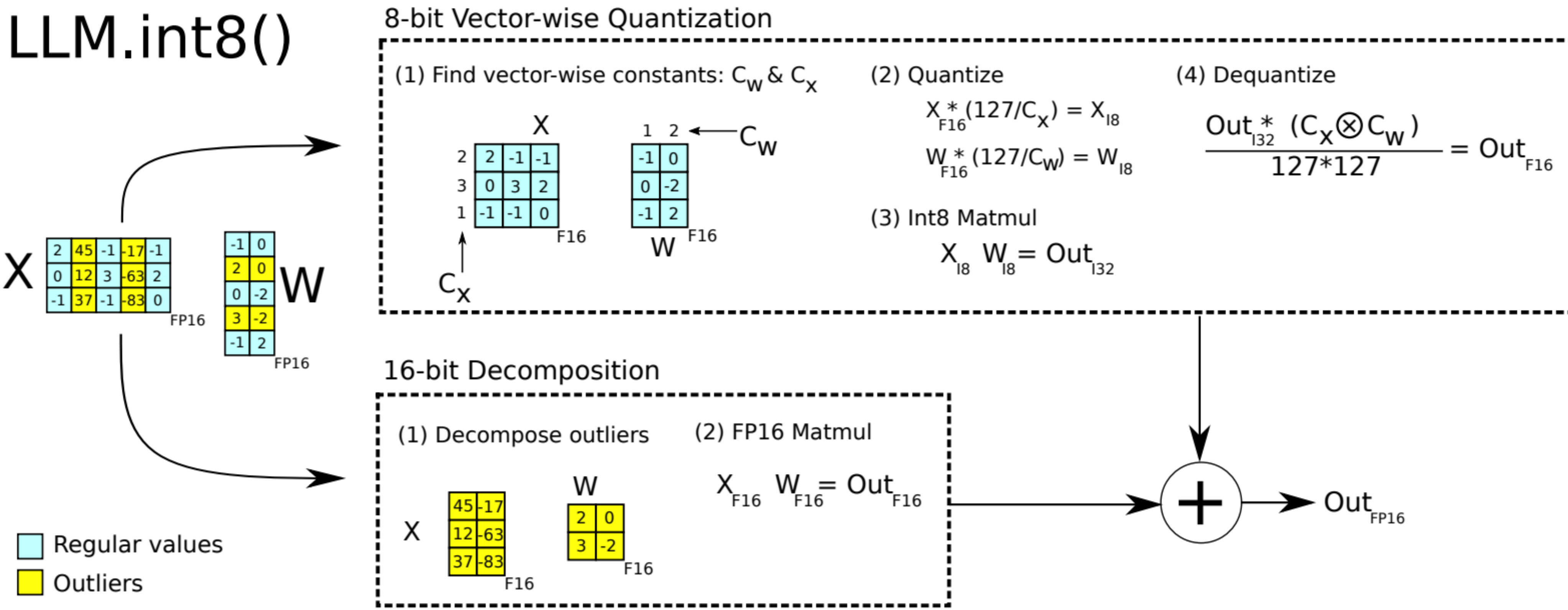
- The effect of outlier features
  - Set all outlier features (at most 7 hidden dimensions) of a layer to zero
    - 為了不要讓error累積，一次只做一層，其他層保持原樣
    - Accuracy會掉20~40%
    - Perplexity increases by 600~1000%
  - Set 7 random feature dimensions of a layer to zero
    - 為了不要讓error累積，一次只做一層，其他層保持原樣
    - Accuracy只掉0.02~0.3%
    - Perplexity increases by 0.1%

LLM.int8(): The Fact of Outliers (3/3)

- The effect of outlier features
  - These outliers are critical for transformer performance
  - Quantization precision for these outlier features is paramount as even tiny errors greatly impact model performance

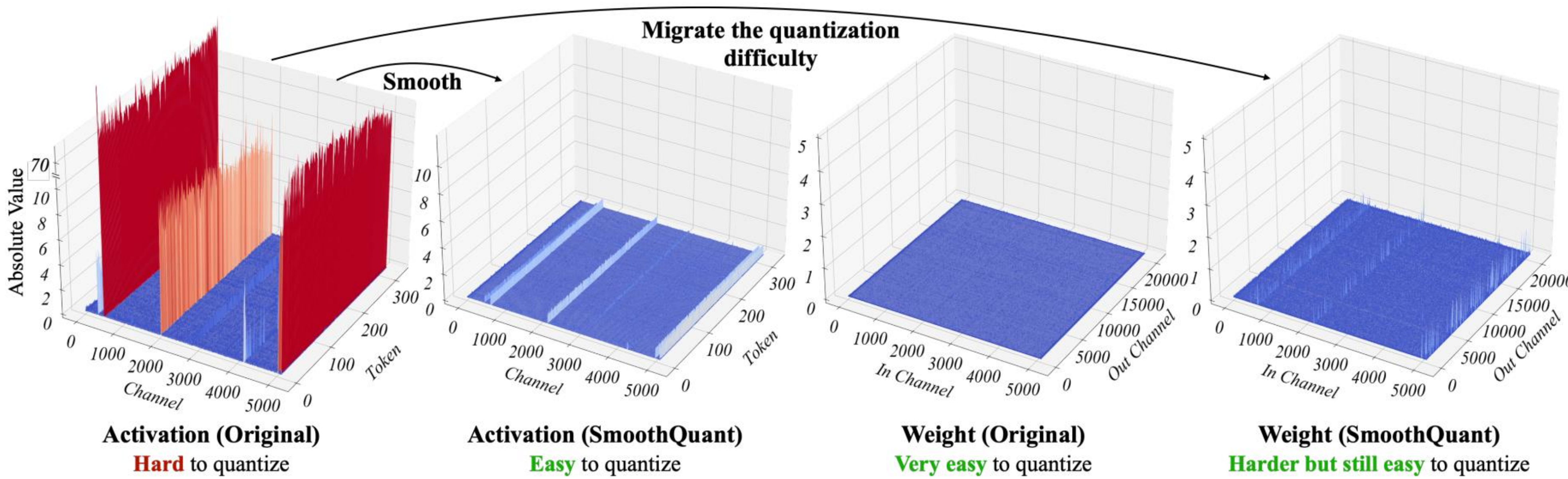
LLM.int8(): Methodology

- Structured outliers



SmoothQuant (1/3)

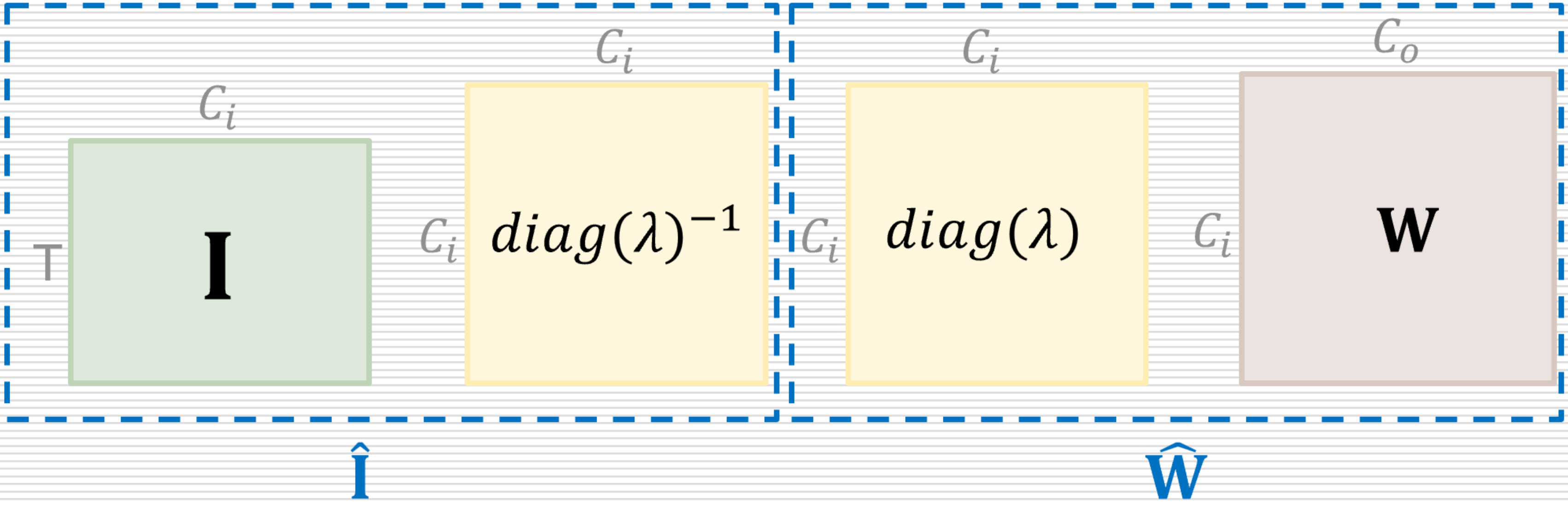
- Idea
  - Migrate the quantization difficulty from activations to weights



SmoothQuant (2/3)

- Channel-wise smoothing

$$\mathbf{O} = (\mathbf{I} \cdot \mathrm{diag}(\lambda)^{-1}) \cdot (\mathrm{diag}(\lambda) \cdot \mathbf{W}) = \hat{\mathbf{I}}\hat{\mathbf{W}}$$

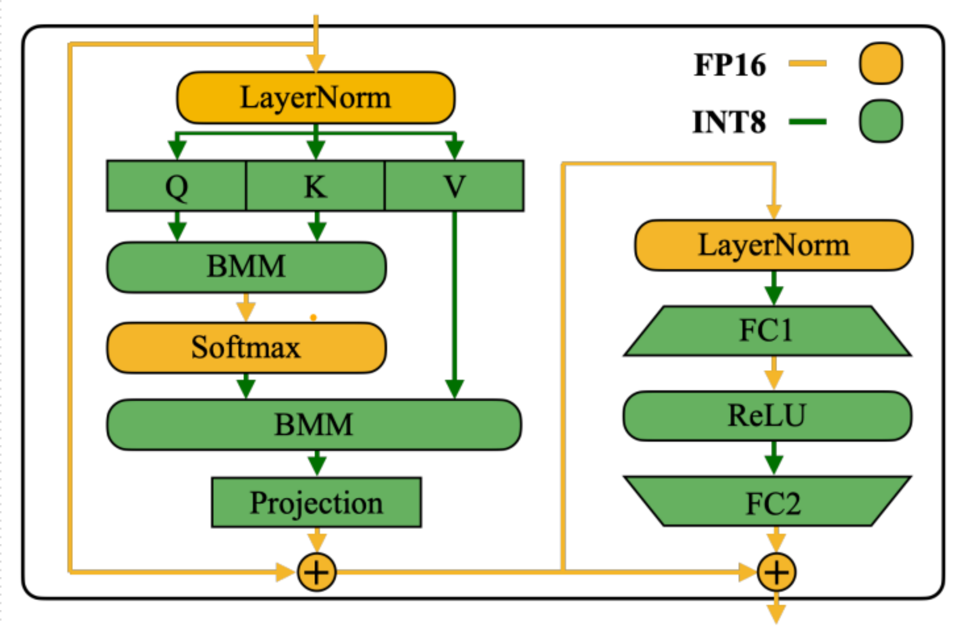


SmoothQuant (3/3)

- Propose smoothing factor $\boldsymbol{\lambda} \in \mathbb{R}^{c_i}$
  - Splits some quantization difficulties to the weights

$$\boldsymbol{\lambda}_\rho = \frac{maxAbs\left(\boldsymbol{I}_\rho\right)^{\alpha}}{maxAbs\left(\boldsymbol{W}_\rho\right)^{1-\alpha}}, \qquad \rho \in 1, 2, \ldots, C_{embd}$$

    - $\boldsymbol{\alpha}$ control how much difficulty is migrated



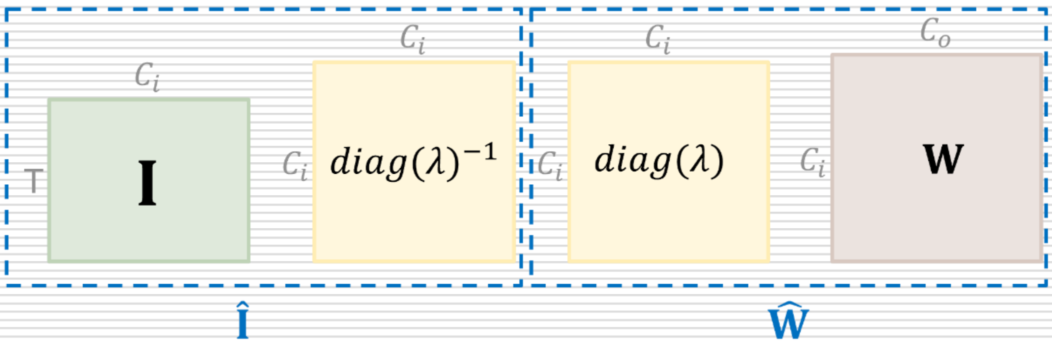
FPTQ

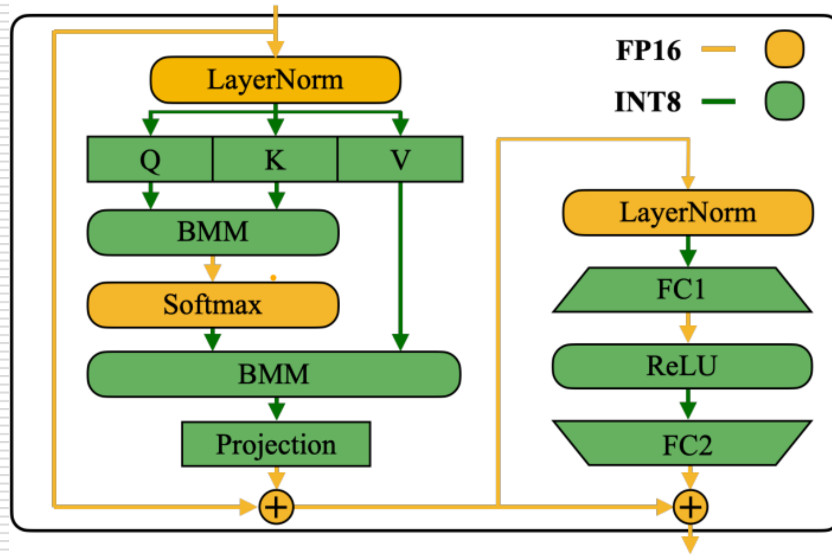
- **L**ogarithmic **A**ctivation **E**qualization
  - Weight-independent

$$\lambda_\rho = \frac{maxAbs(\boldsymbol{I}_\rho)}{\log_2(2 + maxAbs(\boldsymbol{I}_\rho))}, \qquad \rho \in 1, 2, \ldots \ldots, C_{embd}$$

- If outlier exceed a preset threshold
  - **Dynamic** quantization

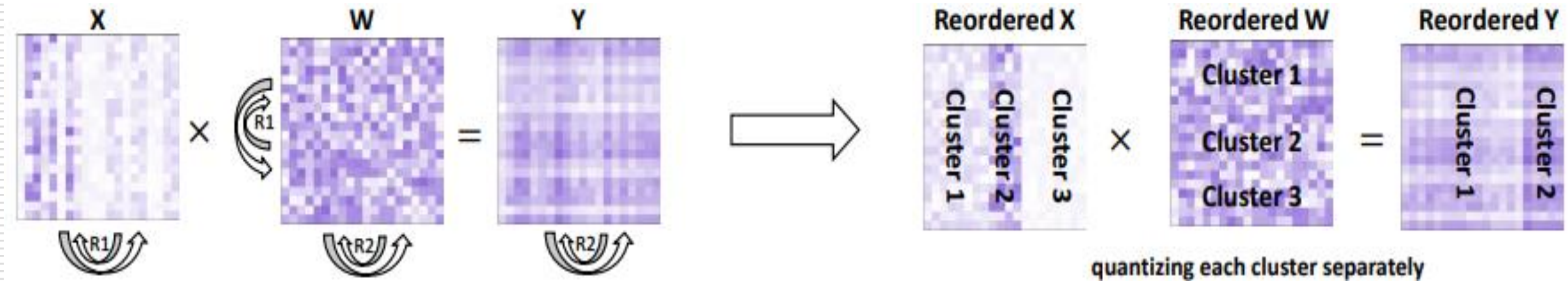
$$\mathbf{O} = (\mathbf{I} \cdot \mathrm{diag}(\lambda)^{-1}) \cdot (\mathrm{diag}(\lambda) \cdot \mathbf{W}) = \hat{\mathbf{I}}\hat{\mathbf{W}}$$





RPTQ (1/2)

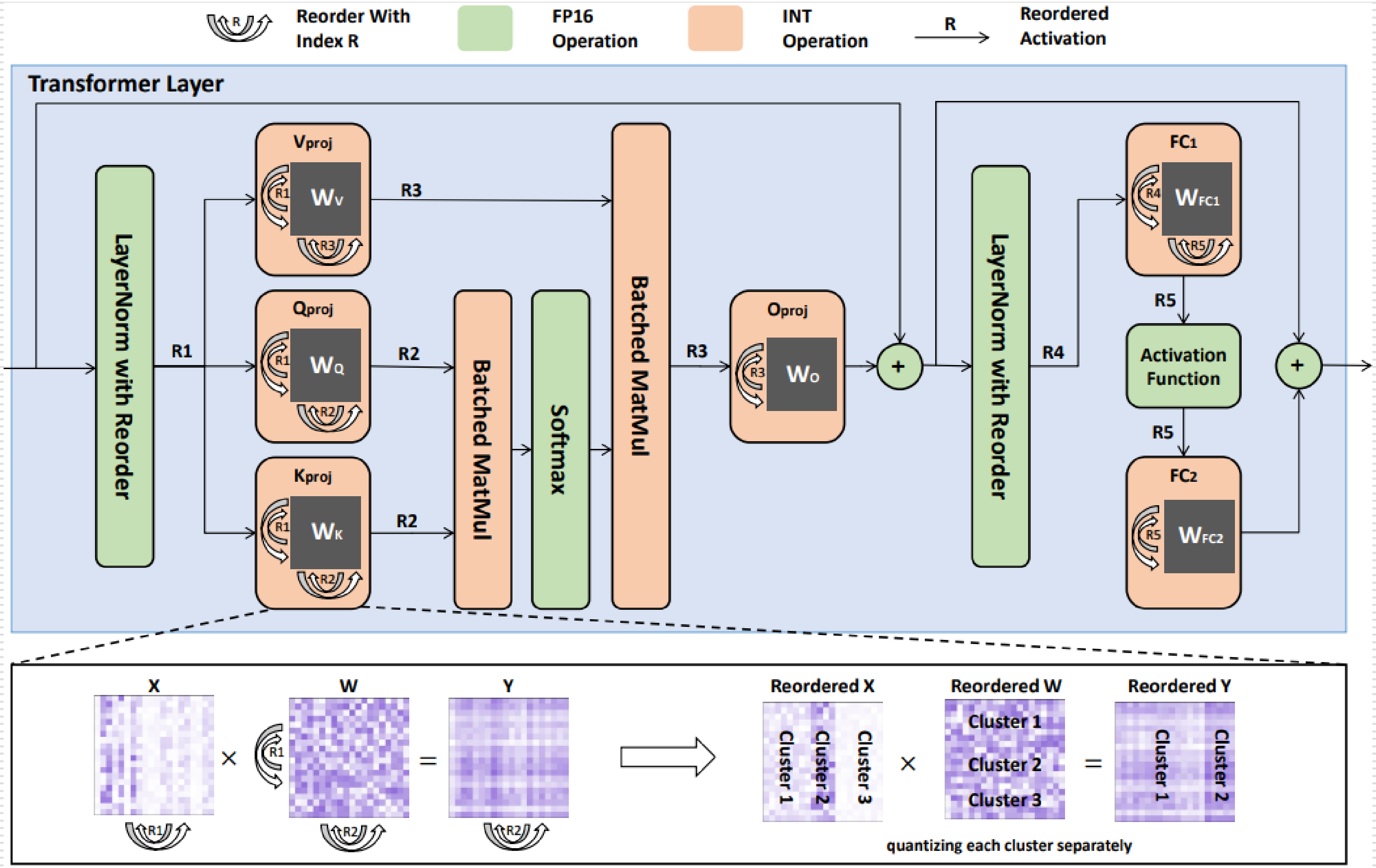
- Per-group quantization
  - Reorder activation & weight by clustering algorithm



- Flow
  - Representative vector
    - Each activation channel's (min, max) values after calibration
  - Apply k-means, then get the result ***R***
  - Activations and their corresponding weights are reordered by ***R*** to maintain computational equivalence
  - Different scales are calculated for different clusters

RPTQ (2/2)

- Per-group quantization
  - Reorder activation & weight by clustering algorithm

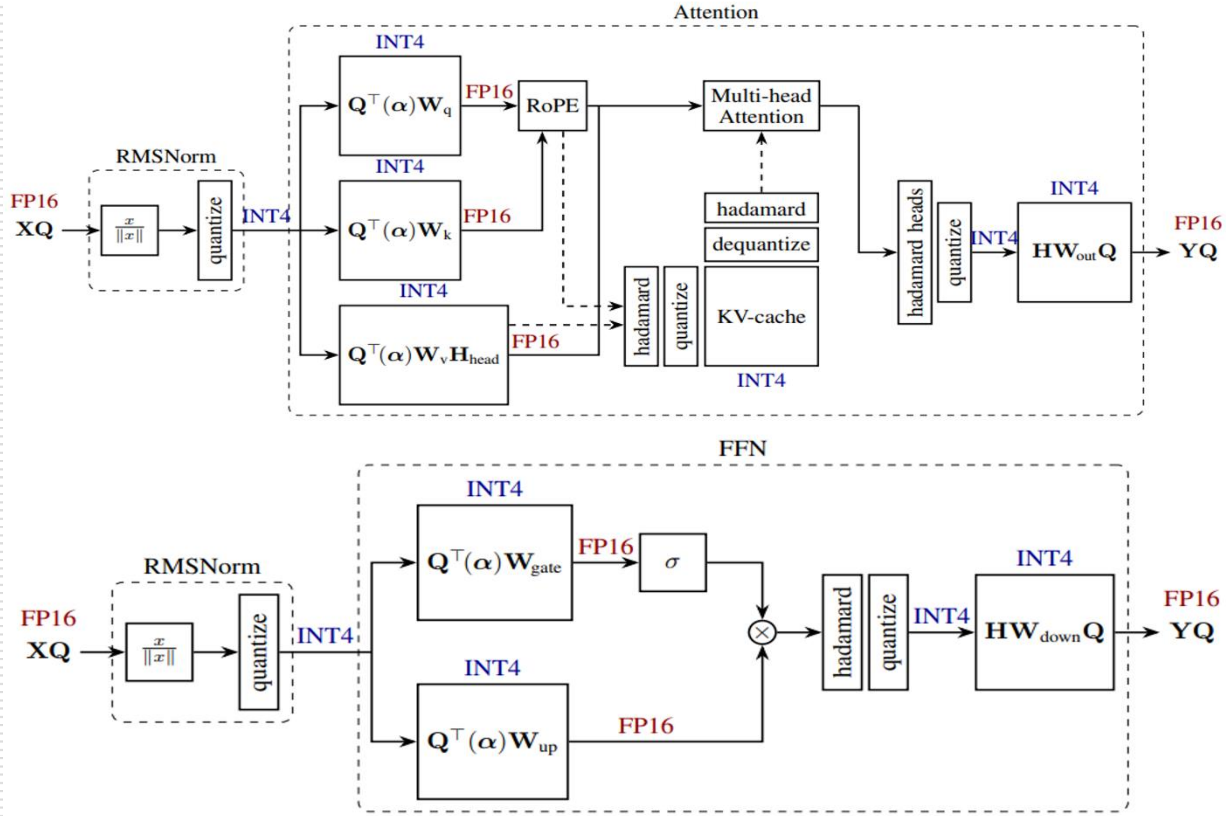


QuaRot (1/2)

- Rotation-based

QuaRot (2/2)

- Retains floating-point operations for
  - Layer normalization, residual link additions
  - Rotation of input activations
  - Multi-head Attention
  - Gating in the FFN



PFPQuant:
A Pure Fixed-Point Quantization Framework for Edge Devices without Floating-Point Units

LLM Quantization w/o FPUs

- FPUs
  - Floating-point units
- w/o FPUs
  - Means all computation should be
    - Integer operation & static shifts
- Difficulty
  - Most of uniform integer quantization is mixed precision
    - **Layer normalization** (LayerNorm、RMSNorm)
      - All
    - Residual-link addition
      - All
    - Outlier-related computation
      - LLM.int8()

Main Components of PFPQuant

- **G**eneralized **O**utlier **M**itigation
  - **GOM**
- **O**utlier-**R**educed **I**nteger **L**ayer**N**orm
  - **ORI-LN**
- **R**esidual-**L**ink with **S**pace-**A**ccuracy **T**radeoff
  - **RL-SAT**

GOM: Outlier Mitigation (1/2)

- Equivalence

$$O = I \cdot W = \underbrace{\boxed{\{I \cdot diag(\lambda)^{-1}\}}}_{\boldsymbol{I_{new}}} \cdot \underbrace{\boxed{\{diag(\lambda) \cdot W\}}}_{\boldsymbol{W_{new}}}$$

- **E**nhanced **L**ogarithmic **A**ctivation **E**qualization
  - E-LAE
  - **LAE** proposed by FPTQ lacks the flexibility to adjust various quantization nodes
  - We introduced an $\boldsymbol{\alpha}$ exponent term to the original formula

$$\lambda_\rho = \left[\frac{maxAbs(\boldsymbol{I}_\rho)}{\log_2(2 + maxAbs(\boldsymbol{I}_\rho))}\right]^{\boldsymbol{\alpha}}, \qquad \rho \in 1, 2, \dots, C_{embd}$$

GOM: Outlier Mitigation (2/2)

- **M**edian-**C**onvergent **O**utlier **M**itigation
  - MCOM
  - Goal
    - Static per-tensor quantization for input activations
      - Extremely hardware-friendly
  - Idea
    - Based on **E-LAE**
    - Concentrate the elements of the same tensor around the median

$$\boldsymbol{V_{maxAbs}} = \{maxAbs(I_1), \quad maxAbs(I_2), \quad \dots, \quad maxAbs(I_{C_{embd}})\}$$

$$\lambda_\rho = \left[\frac{\boldsymbol{\beta} + maxAbs(\boldsymbol{I_\rho})}{\log_2(2 + \boldsymbol{median(V_{maxAbs})})}\right]^{\boldsymbol{\alpha}}, \quad \rho \in 1, 2, \dots, C_{embd}$$

GOM: Combined with Data-Shifting

- Data-shifting vector

$$\boldsymbol{\delta_\rho} = \frac{-\mathbf{1}}{\mathbf{2}}\left[\boldsymbol{max}(\boldsymbol{I_\rho}) + \boldsymbol{min}(\boldsymbol{I_\rho})\right], \qquad \boldsymbol{\rho} \in \mathbf{1}, \mathbf{2}, \dots, \boldsymbol{C_{embd}}$$

- Final formula of **GOM**

$$O = \underbrace{\boxed{\{(I+\delta)\cdot diag(\lambda)^{-1}\}}}_{I_{easyQ}} \cdot \underbrace{\boxed{\{diag(\lambda)\cdot W\}}}_{W_{new}} + \overbrace{\boxed{\{B-\delta\cdot W\}}}^{B_{new}}$$

ORI-LN: Outlier Score (1/2)

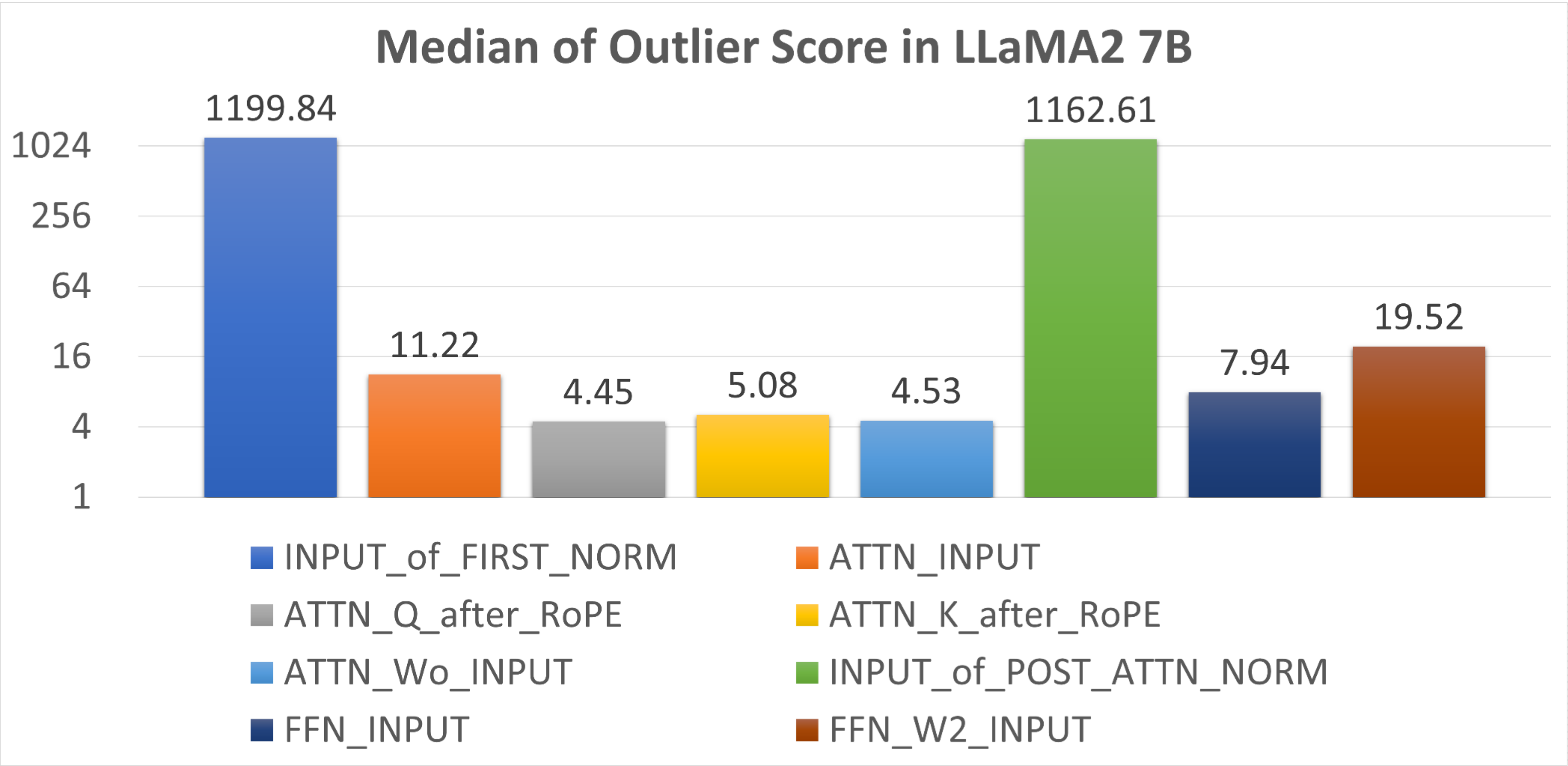
- Previous quantization research has not performed LayerNorm under quantized precision

- To elucidate the difficulty of LayerNorm quantization
  - We define an outlier score $\boldsymbol{OS}$
    - After calibration, $\boldsymbol{V_{maxAbs}}$ is a static vector

$$\boldsymbol{V_{maxAbs}} = \{maxAbs(I_1), \quad maxAbs(I_2), \quad \dots, \quad maxAbs(I_{C_{embd}})\}$$

$$\boldsymbol{OS} = \frac{\boldsymbol{max}(V_{maxAbs})}{\boldsymbol{median}(V_{maxAbs})}$$

ORI-LN: Outlier Score (2/2)



ORI-LN: Vanilla Integer LayerNorm (1/2)

- Static per-tensor quantization for input activations
  - For simplicity, we use the RMSNorm formula, a variant of LayerNorm, to illustrate our approach

$$O_\rho = \frac{I_\rho \cdot \gamma_\rho}{\sqrt{\frac{1}{C_{embd}} \cdot \sum_{m=1}^{C_{embd}} (I_m)^2}}, \qquad \rho \in 1, 2, \ldots, C_{embd}$$

$$I_\rho^{INT} = \left\lfloor \frac{I_\rho}{s_i} \right\rceil + z_i, \qquad I_\rho \approx s_i\left(I_\rho^{INT} - z_i\right) \qquad \gamma_\rho \approx \gamma_\rho^{INT} \cdot s_{\gamma_\rho}, \qquad s_{\gamma_\rho} = 2^{shift_{\gamma_\rho}}$$

$$O_\rho \approx \frac{s_i\left(I_\rho^{INT} - z_i\right) \cdot s_{\gamma_\rho} \cdot \gamma_\rho^{INT}}{\sqrt{\frac{1}{C_{embd}} \cdot \sum_{m=1}^{C_{embd}} \left(s_i\left(I_m^{INT} - z_i\right)\right)^2}} = \cancel{\frac{s_i}{\sqrt{s_i^2}}} \cdot \left(s_{\gamma_\rho} \cdot \sqrt{C_{embd}}\right) \cdot \frac{\left(I_\rho^{INT} - z_i\right) \cdot \gamma_\rho^{INT}}{\sqrt{\sum_{m=1}^{C_{embd}} \left(\left(I_m^{INT} - z_i\right)\right)^2}}$$

$$s_{\gamma_\rho} \cdot \sqrt{C_{embd}} \approx M_\rho^{INT} \cdot 2^{shift_{M_\rho}}$$

ORI-LN: Vanilla Integer LayerNorm (2/2)

- SQ represents the outlier mitigation formula proposed by SmoothQuant
- Using the LLaMA-2 7B chat model
  - The average accuracy of each method on LAMBADA, HellaSwag, PIQA, WinoGrande, and ARCe
  - The perplexity on Wikitext-2

Method	Average↑	Wikitext↓
FP	0.7226	12.28
SQ w VILN	0.3106	51036.37
E-LAE w VILN	0.3092	53769.12

ORI-LN

- Introducing outlier mitigation into LayerNorm

$$O_\rho = \frac{I_\rho \cdot \gamma_\rho}{\sqrt{\frac{1}{C_{embd}} \cdot \sum_{m=1}^{C_{embd}} (I_m)^2}} = \frac{\sqrt{C_{embd}}}{\sqrt{\sum_{m=1}^{C_{embd}} \left(\frac{I_m}{\lambda_m} \cdot \lambda_m\right)^2}} \cdot \left(\frac{I_\rho}{\lambda_\rho}\right) \cdot (\lambda_\rho \cdot \gamma_\rho), \qquad \rho \in 1, 2, \ldots, C_{embd}$$

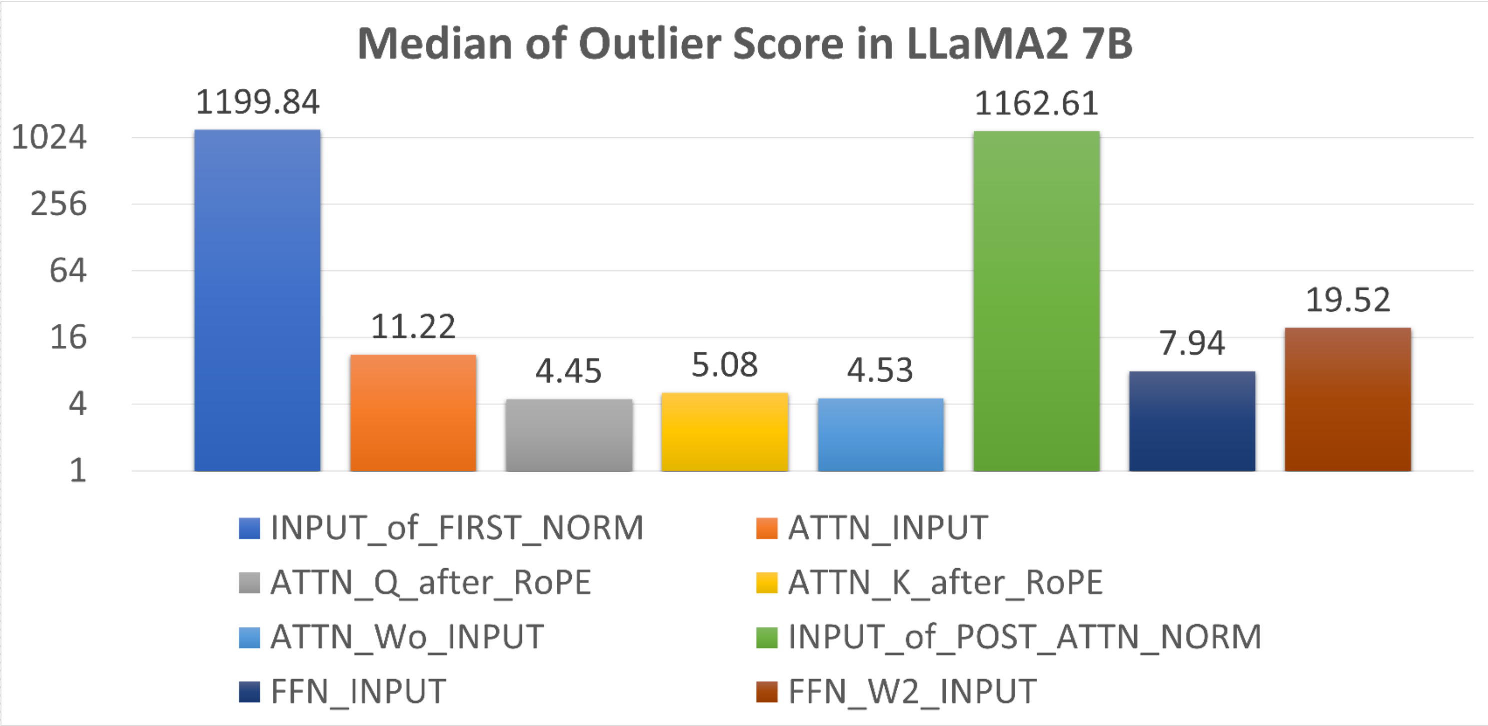
$$O_\rho = \frac{\sqrt{C_{embd}}}{\sqrt{\sum_{m=1}^{C_{embd}} \left(\frac{I_m}{\lambda_m}\right)^2 \cdot (\lambda_m)^2}} \cdot \left(\frac{I_\rho}{\lambda_\rho}\right) \cdot (\lambda_\rho \cdot \gamma_\rho) \qquad I'_\rho = \left(\frac{I_\rho}{\lambda_\rho}\right), \qquad \gamma'_\rho = (\lambda_\rho \cdot \gamma_\rho)$$

$$O_\rho = \frac{I'_\rho \cdot \gamma'_\rho \cdot \sqrt{C_{embd}}}{\sqrt{\sum_{m=1}^{C_{embd}} (I'_m)^2 \cdot (\lambda_m)^2}} \approx \frac{s_{i'}\left({I'_\rho}^{INT} - z_{i'}\right) \cdot s_{\gamma'_\rho} \cdot {\gamma'_\rho}^{INT} \cdot \sqrt{C_{embd}}}{\sqrt{\sum_{m=1}^{C_{embd}} \left(s_{i'}\left({I'_m}^{INT} - z_{i'}\right)\right)^2 \cdot (\lambda_m)^2}} = \left(s_{\gamma'_\rho} \cdot \sqrt{C_{embd}}\right) \cdot \frac{\left({I'_\rho}^{INT} - z_{i'}\right) \cdot {\gamma'_\rho}^{INT}}{\sqrt{\sum_{m=1}^{C_{embd}} \left({I'_m}^{INT} - z_{i'}\right)^2 \cdot (\lambda_m)^2}}$$

$$s_{\gamma'_\rho} \cdot \sqrt{C_{embd}} \approx M_\rho^{INT} \cdot 2^{shift_{M_\rho}}, \quad (\lambda_m)^2 \approx M_{\lambda_m}^{INT} \cdot 2^{shift_{M_{\lambda_m}}}$$

RL-SAT: Recall (1/2)

- Since the residual-link originates from the input of the previous LayerNorm, it inherits its quantization difficulty



RL-SAT: Recall (2/2)

- Per-tensor quantization for matrix operation

$$I^{INT} = \left\lfloor \frac{I^{FP}}{s_i} \right\rceil + z_i, \qquad \overline{I^{FP}} = s_i\left(I^{INT} - z_i\right)$$

$$W^{INT} = \left\lfloor \frac{W^{FP}}{s_w} \right\rceil + z_w, \qquad \overline{W^{FP}} = s_w\left(W^{INT} - z_w\right)$$

$$B^{INT} = \left\lfloor \frac{B^{FP}}{s_i \cdot s_w} \right\rceil, \qquad \overline{B^{FP}} = \boxed{s_i s_w} \cdot B^{INT}$$

$$O = I \cdot W + B \approx \overline{O^{FP}} = \overline{I^{FP}} \cdot \overline{W^{FP}} + \overline{B^{FP}}, \qquad O^{INT} = \left\lfloor \frac{\overline{O^{FP}}}{s_o} \right\rceil + z_o$$

$$O^{INT} = \left\lfloor \frac{\sum s_i\left(I^{INT} - z_i\right) s_w\left(W^{INT} - z_w\right) + s_i s_w \cdot B^{INT}}{s_o} \right\rceil + z_o = \left\lfloor \frac{\boxed{s_i s_w}}{s_o} \left[\sum \left(I^{INT} - z_i\right)\left(W^{INT} - z_w\right) + B^{INT} \right] \right\rceil + z_o$$

$$\frac{s_i s_w}{s_o} \approx M^{INT} \cdot 2^{shift}$$

RL-SAT: Vanilla Residual Quantization

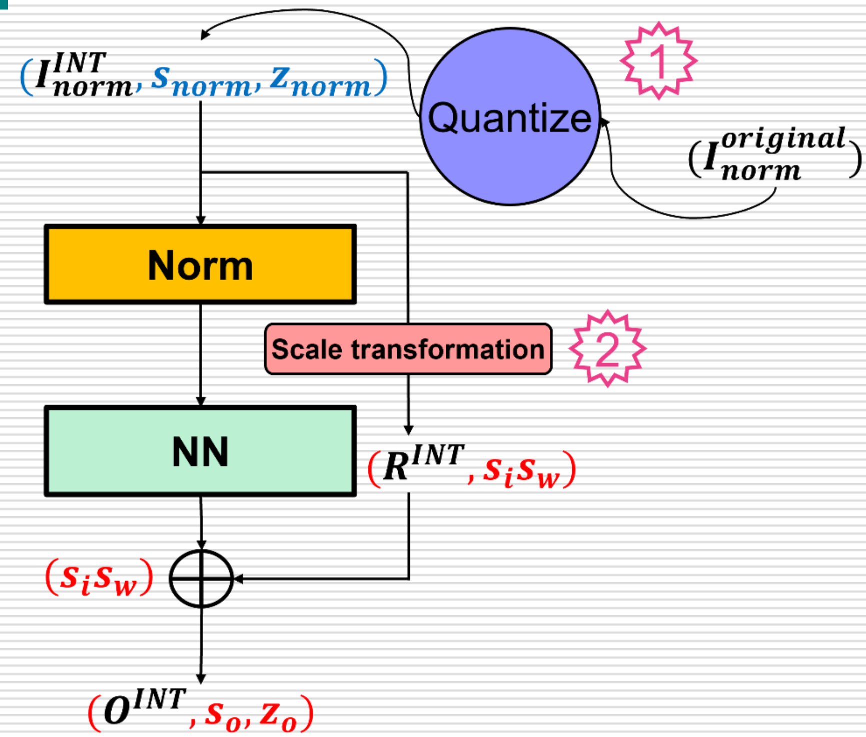
$$I_{norm}^{INT} = \left\lfloor \frac{I_{norm}^{original}}{s_{norm}} \right\rceil + z_{norm}$$

$$\overline{R_0^{FP}} = \overline{I_{norm}^{FP}} = s_{norm} \cdot (I_{norm}^{INT} - z_{norm}) \approx I_{norm}^{original}$$

$$R^{INT} = \left\lfloor \frac{\overline{R_0^{FP}}}{\boxed{s_i s_w}} \right\rceil = \left\lfloor \frac{s_{norm}}{s_i s_w} \cdot (I_{norm}^{INT} - z_{norm}) \right\rceil$$

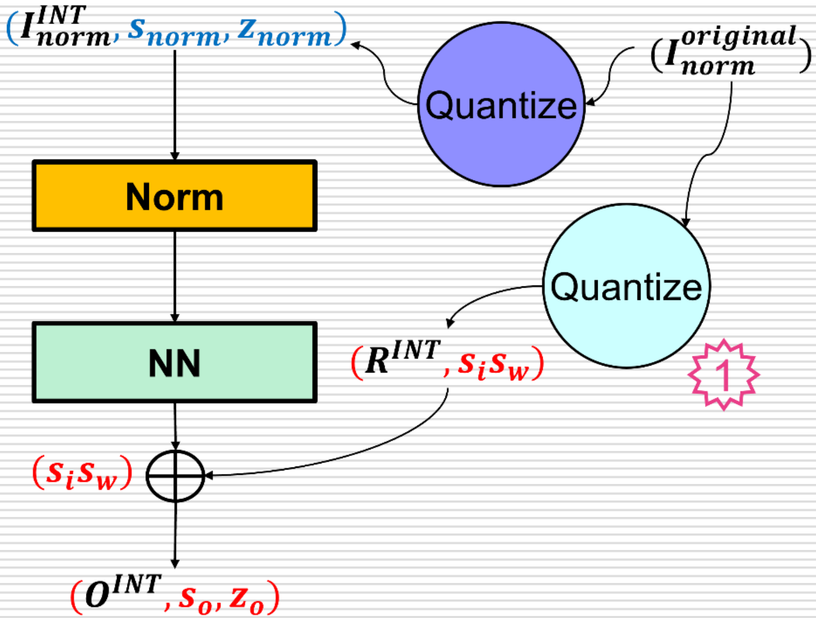
$$\overline{R^{FP}} = s_i s_w \cdot \left\lfloor \boxed{\frac{s_{norm}}{s_i s_w}} \cdot (I_{norm}^{INT} - z_{norm}) \right\rceil = s_i s_w \cdot R^{INT}$$

$$\frac{s_{norm}}{s_i s_w} \approx M_{sc_{Transform}}^{INT} \cdot 2^{shift_{sc_{Transform}}} = M_{sc_{Transform}}$$



RL-SAT

- Directly quantizes $I_{norm}^{original}$ to a scale that allows correct accumulation
- While this approach uses slightly more memory space, it reduces one instance of quantization error (due to rounding)
- Since the additional memory consumption is negligible compared to the weights of LLMs, PFPQuant adopts this method (**RL-SAT**)

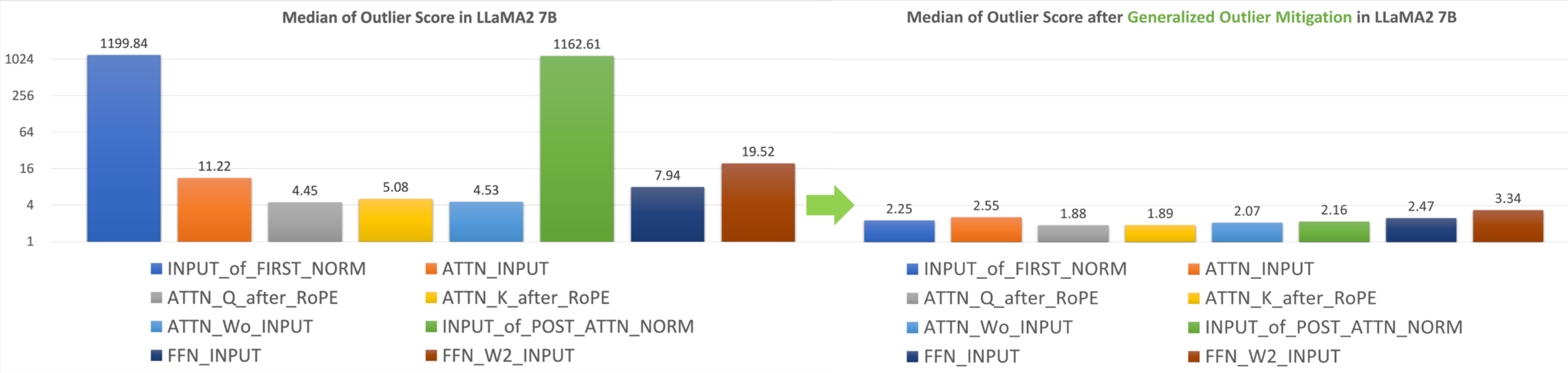


$$R^{INT} = \left\lfloor \frac{I_{norm}^{original}}{s_i \cdot s_w} \right\rceil, \quad \overline{R^{FP}} = s_i s_w \cdot R^{INT} \approx R^{FP} = I_{norm}^{original}$$

1

Experiment Results: Effectiveness of GOM

- GOM
  - **G**eneralized **O**utlier **M**itigation



Experiment Result: Quantized LLaMA 2

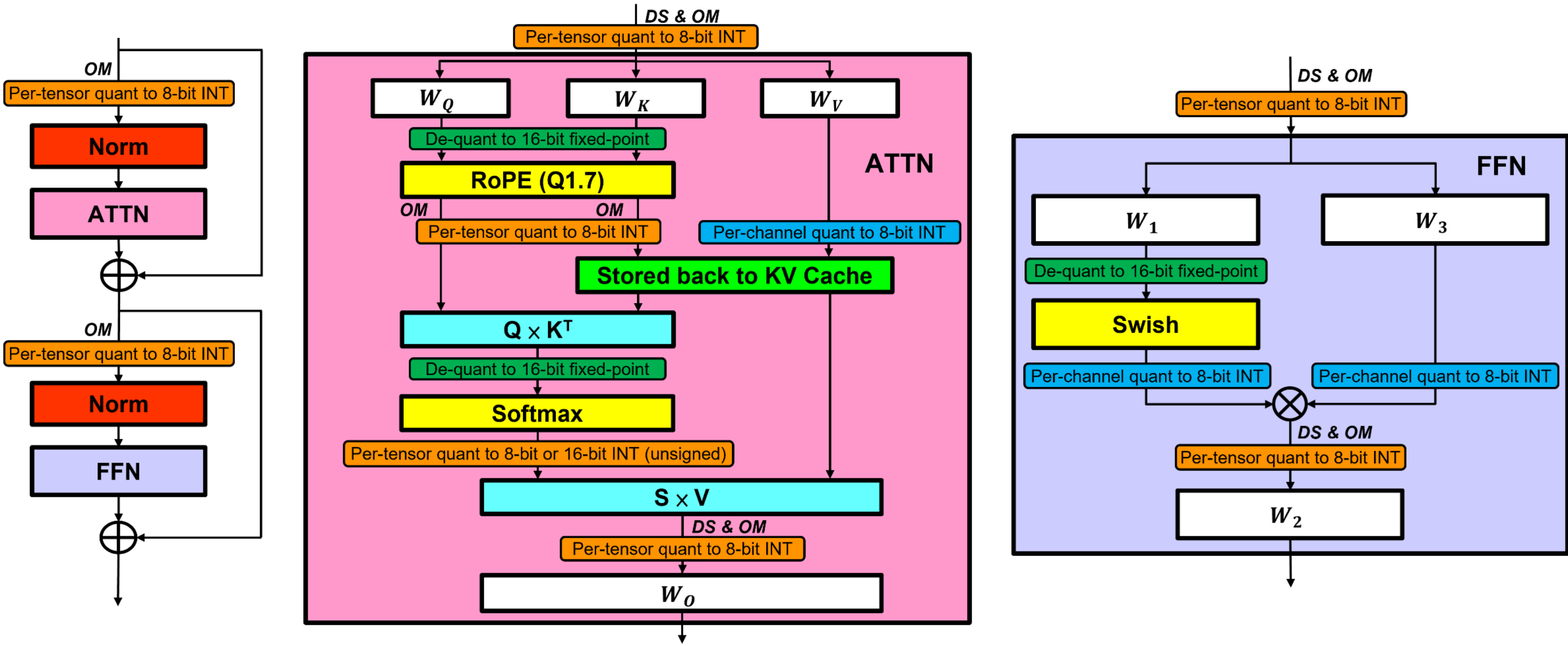
- We tested PFPQuant on various common tasks using the LLaMA-2 7B chat model

Method	LAMBDA	HellaSwag	PIQA	WinoGrande	ARCe	**Average↑**	**Wikitext↓**
FP	0.6850	0.7530	0.7666	0.6646	0.7441	0.7226	12.28
SQ w R8-S8	0.6238	0.7266	0.7568	0.6464	0.7033	0.6914	41.52
SQ w R16-S8	0.6286	0.7269	**0.7579**	0.6409	0.7050	0.6919	41.46
SQ w R8-S16	0.6173	0.7182	0.7546	0.6346	0.7066	0.6863	22.55
SQ w R16-S16	0.6282	0.7220	0.7541	0.6472	0.7104	0.6924	22.39
E-LAE w R8-S8	0.6255	0.7182	0.7541	0.6251	0.7037	0.6853	36.08
E-LAE w R16-S8	0.6271	0.7192	0.7530	0.6172	0.7066	0.6846	36.15
E-LAE w R8-S16	0.6112	0.7111	0.7552	0.6227	0.7075	0.6815	19.34
E-LAE w R16-S16	0.6151	0.7125	0.7519	0.6283	0.7079	0.6831	19.25
MCOM w R8-S8	0.6275	0.7380	0.7519	0.6527	0.7189	0.6978	27.86
MCOM w R16-S8	0.6295	0.7378	0.7486	**0.6701**	0.7180	0.7008	26.73
MCOM w R8-S16	0.6327	0.7383	0.7552	0.6559	**0.7243**	0.7013	14.80
MCOM w R16-S16	**0.6339**	**0.7410**	**0.7579**	0.6535	0.7218	**0.7016**	**14.77**

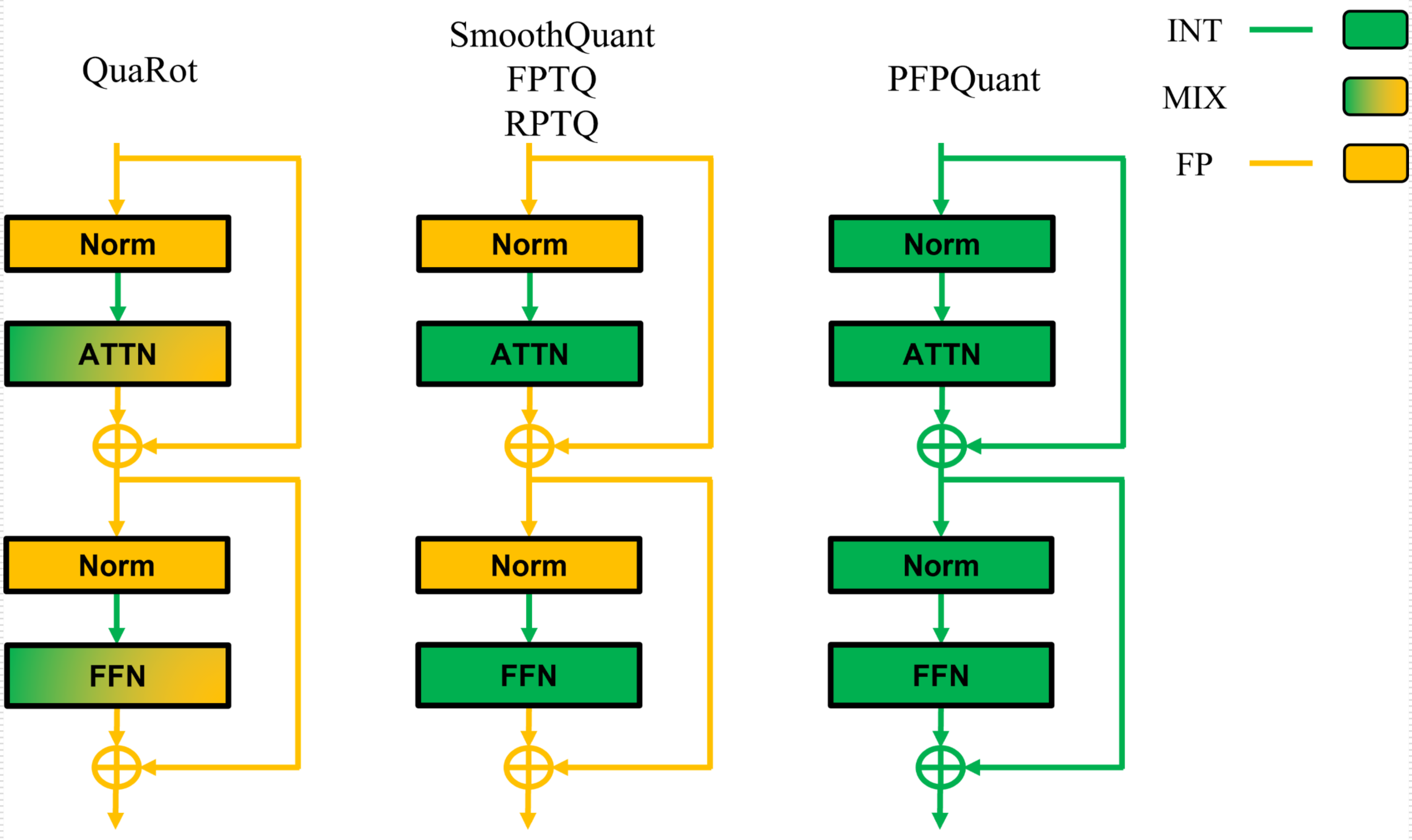
Table 2: **R** indicates the bit-width of RoPE sine and cosine tables, and **S** indicates the softmax output bit-width. For example, **R8-S8** uses 8-bit RoPE tables and clamps the softmax output to unsigned 8-bit (Q0.8). The data shows results after applying GOM, ORI-LN, and RL-SAT; omitting any of these strategies leads to pure fixed-point quantization failure.

Implementation Overview

- LLaMA-2 7B



Comparison



QuaRot
SmoothQuant
FPTQ
RPTQ
PFPQuant
INT
MIX
FP
Norm
ATTN
Norm
FFN
Norm
ATTN
Norm
FFN
Norm
ATTN
Norm
FFN

Homework

HW1 & HW2

- HW1 (35%)
  - 詳細介紹k-means分群演算法，並說明你覺得該演算法要怎麼用來壓縮NN模型
    - 請善用圖示說明：圖最好自己畫，少用截圖
    - 以**簡報**形式呈現
- HW2 (65%)
  - 請從下列論文擇一閱讀，並製作圖文並茂的**簡報**，圖例為主，文字需簡潔扼要：圖最好自己畫，少用截圖
    - AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration
    - QuaRot: Outlier-free 4-bit Inference in Rotated LLMs
    - A White Paper on Neural Network Quantization
    - Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference

Format

- 將HW1 & HW2**融合成一份**PowerPoint檔案(**不超過25頁投影片**)，繳交該檔案，檔名為:
  - 中文姓名_Lab12
- 將該份PowerPoint檔案轉成PDF檔案，一併繳交，檔名為:
  - 中文姓名_Lab12
- 總共要繳交**兩份**檔案，注意，**不要壓縮**
- 別直接傳兩個檔案，請至少附上姓名、主旨&禮貌
- 繳交至以下信箱：
  - anson.twhu.ee11@nycu.edu.tw

Deadline

- 23:59, Aug 26, 2024

Thank you