Quantization Methods for Efficient ML Inference

Amir Gholami, in collaboration with

Sehoon Kim, Coleman Hooper, Zhen Dong, Xiuyu Li, Sheng Shen, Michael Mahoney, Kurt Keutzer

University of California Berkeley

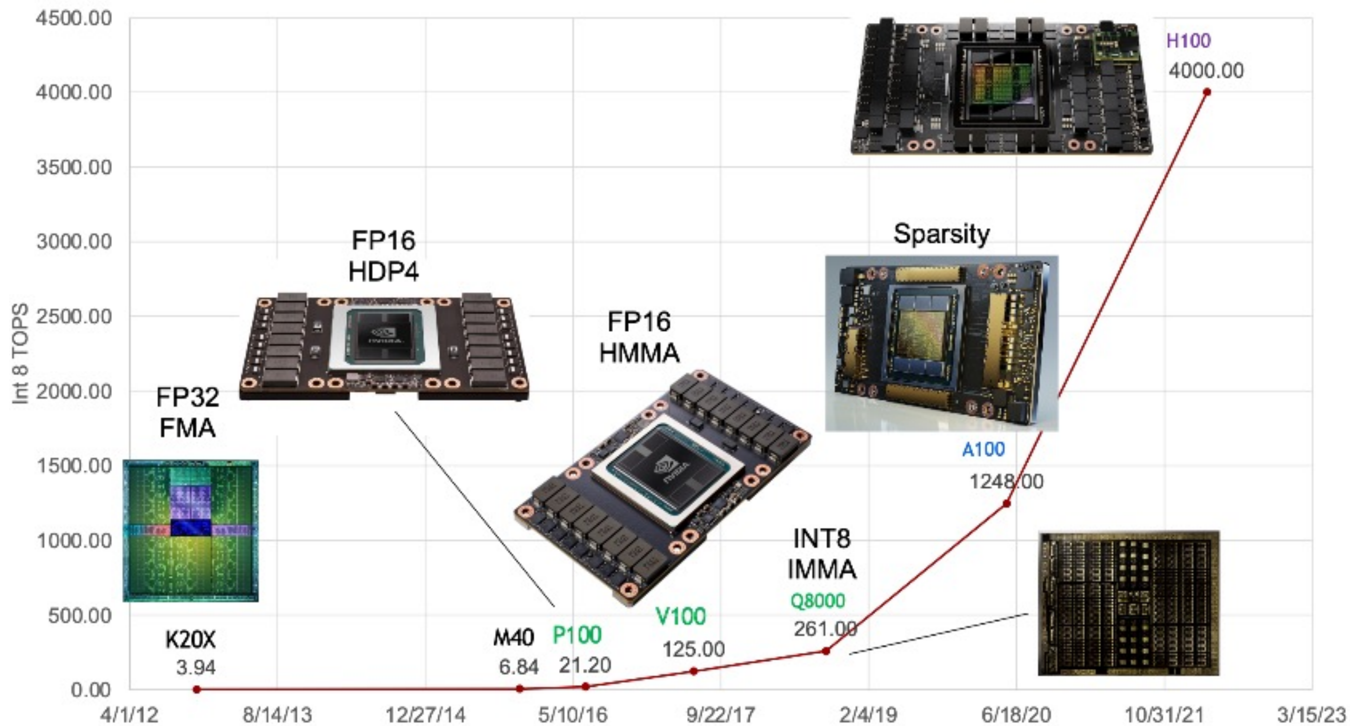
HotChips 2023

Outline

- Why Quantization?
- Basic Concepts of Quantization
- Advanced Concepts of Quantization

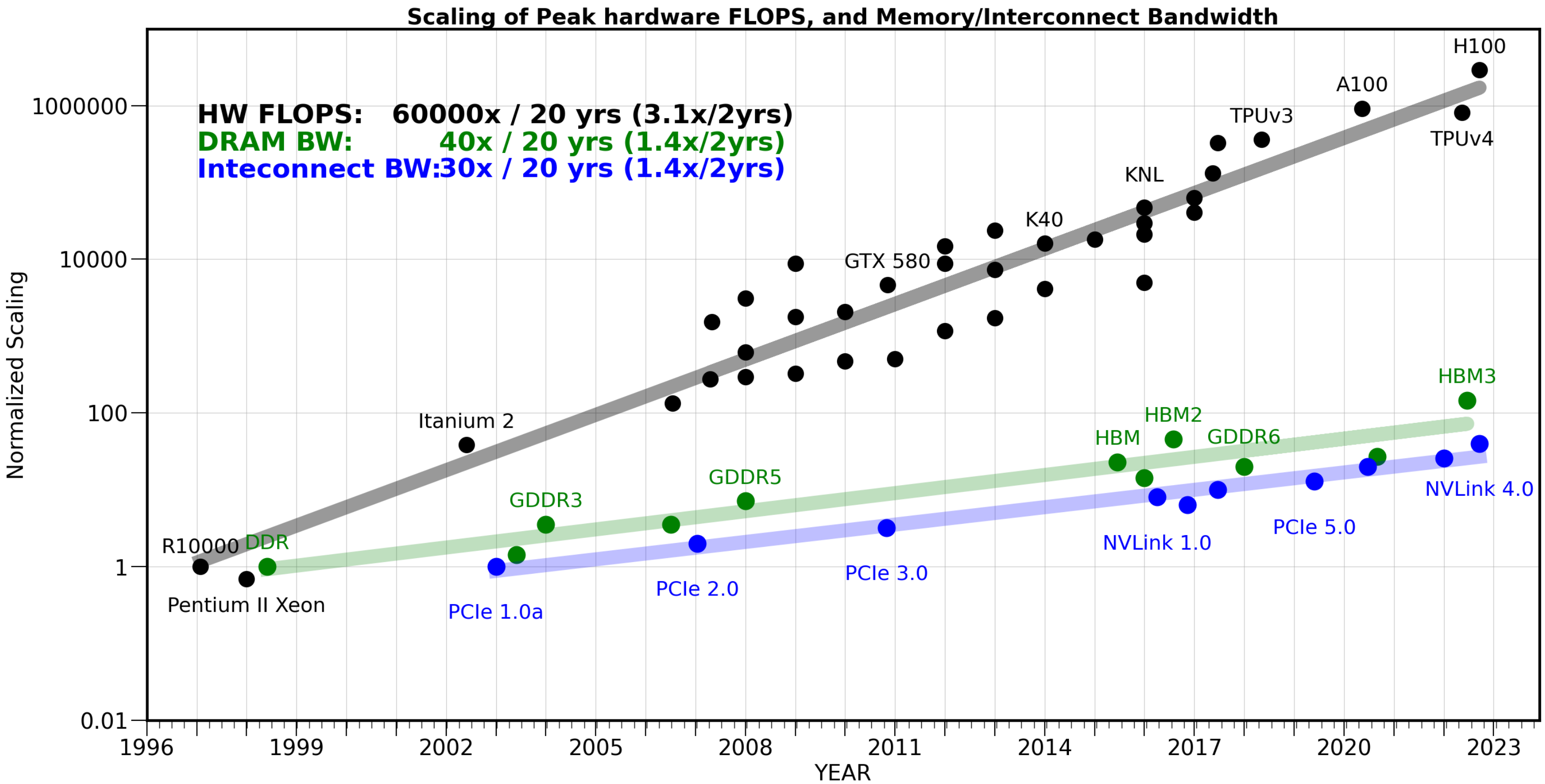
Single-Chip GPU Inference Performance
1000X in 10 years!

Gains from
- Number representation
  - FP32, FP16, Int8, FP8
- Complex instructions
  - DP4, HMMA, IMMA
- Process
  - 28nm, 16nm, 7nm, 5nm



Slide Credit: Bill Dally Keynote at Berkeley Deep Drive, Deep Learning and Autonomous Vehicles, 2023.

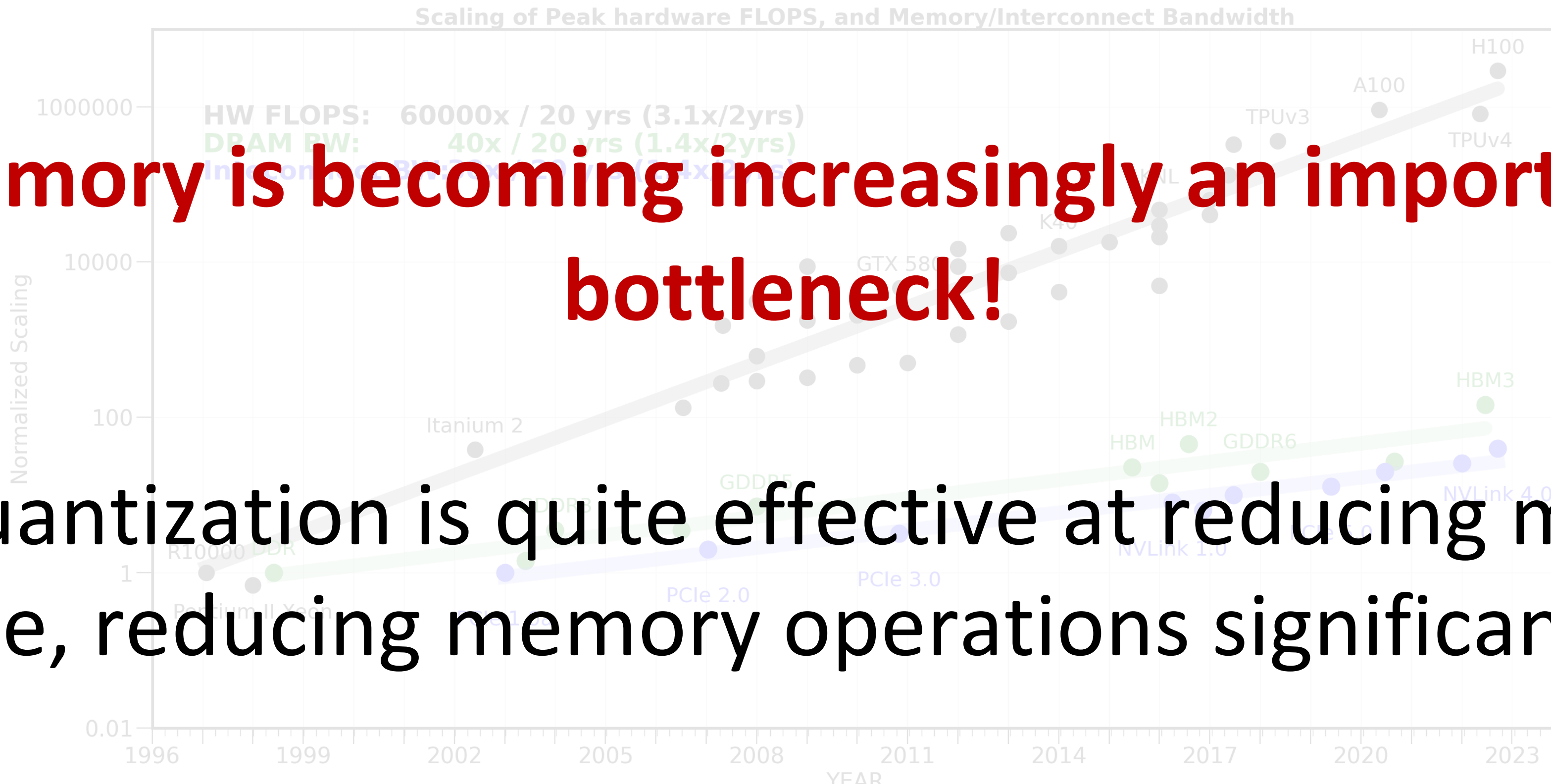
Memory Wall: Main Bottleneck is Memory Bandwidth



Memory is developing much slower than computes

Amir Gholami, Zhewei Yao, Sehoon Kim, Michael W. Mahoney, Kurt Keutzer, AI and Memory Wall, Riselab Medium Blogpost, 2021.

Memory Wall: Main Bottleneck is Memory Bandwidth



Memory is developing much slower than computes

Memory is becoming increasingly an important bottleneck!

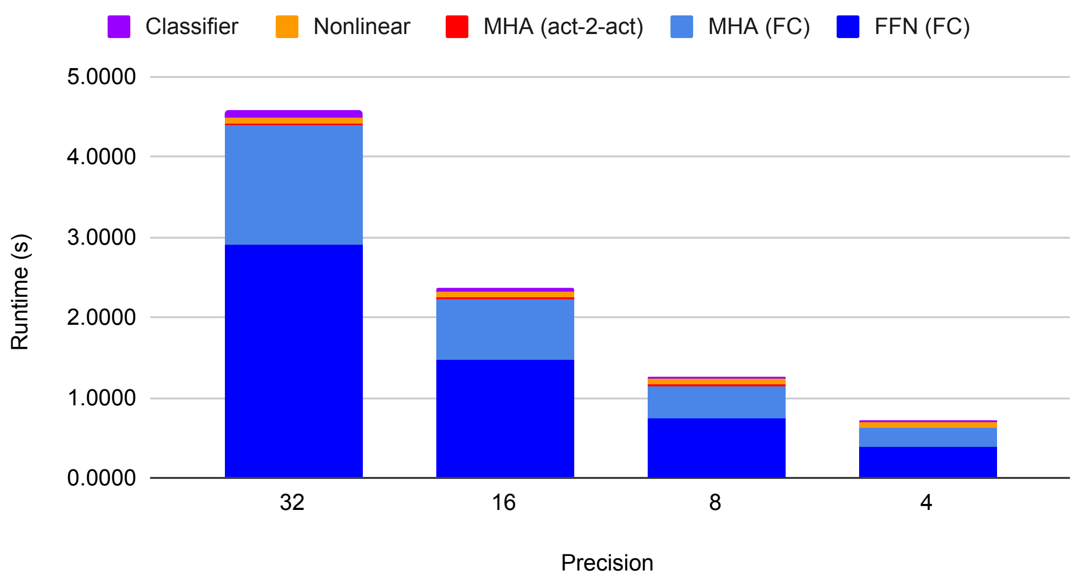
✅ Quantization is quite effective at reducing model size, reducing memory operations significantly

Amir Gholami, Zhewei Yao, Sehoon Kim, Michael W. Mahoney, Kurt Keutzer, AI and Memory Wall, Riselab Medium Blogpost, 2021.

Memory Wall for LLM Inference!

The dominant contributor to runtime is the time for **memory bandwidth not compute**

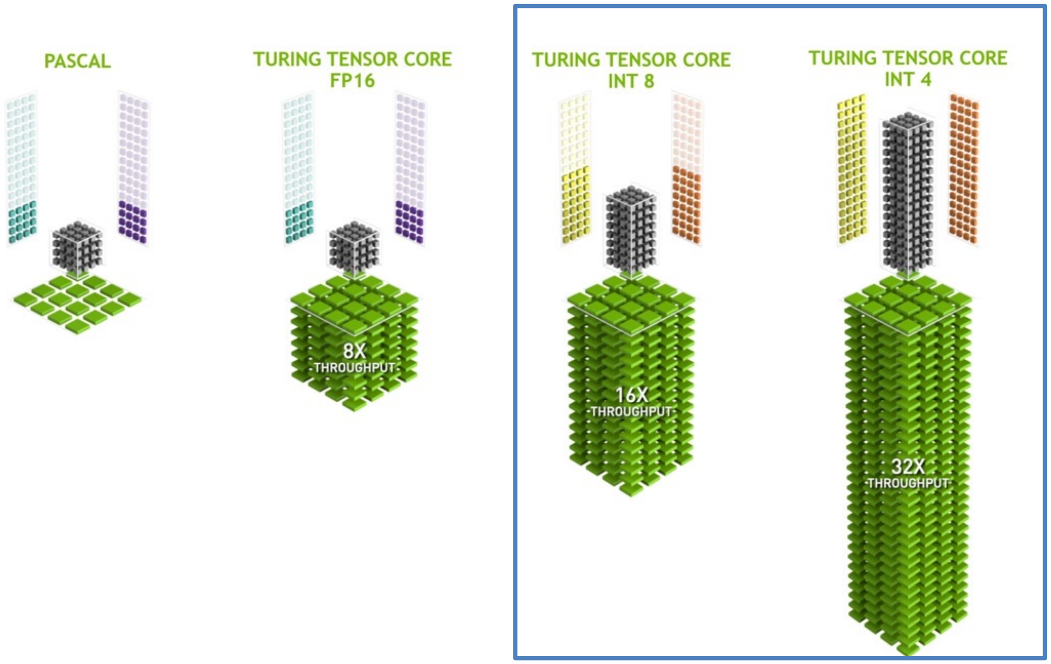
Breakdown of LLaMA 7B model with Seq Len of 128 and batch of 1



S. Kim*, C. Hooper*, A. Gholami*, Z. Dong, X. Li, S. Sheng, M. Mahoney, K. Keutzer, SqueezeLLM: Dense-and-Sparse Quantization, arxiv: :2306.07629.

Quantization enables low precision arithmetic

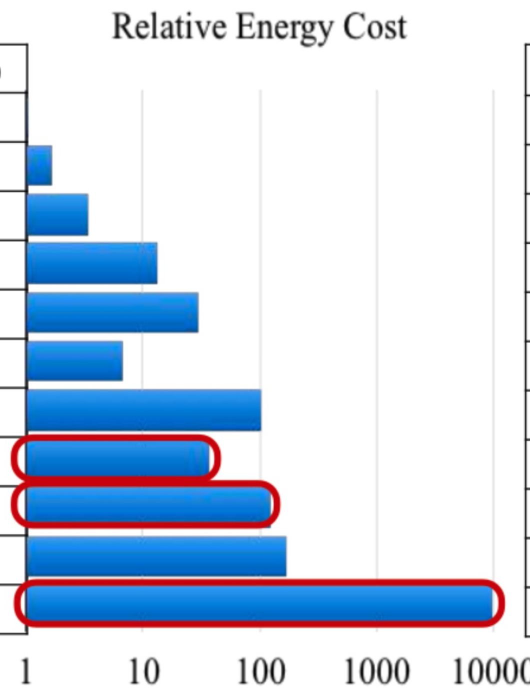
- Lower precision weights mean less energy per Multiply-Accumulate
- Also enables putting more MAC units per unit of silicon

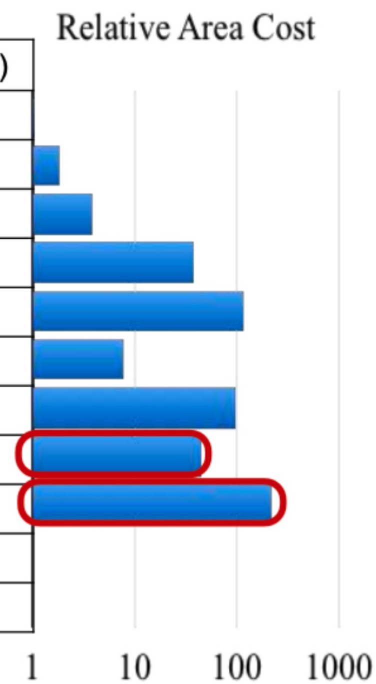


✅ Quantization is great for compute bound inference problems as it allows us to utilize lower precision ALUs

Energy Consumption

Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A





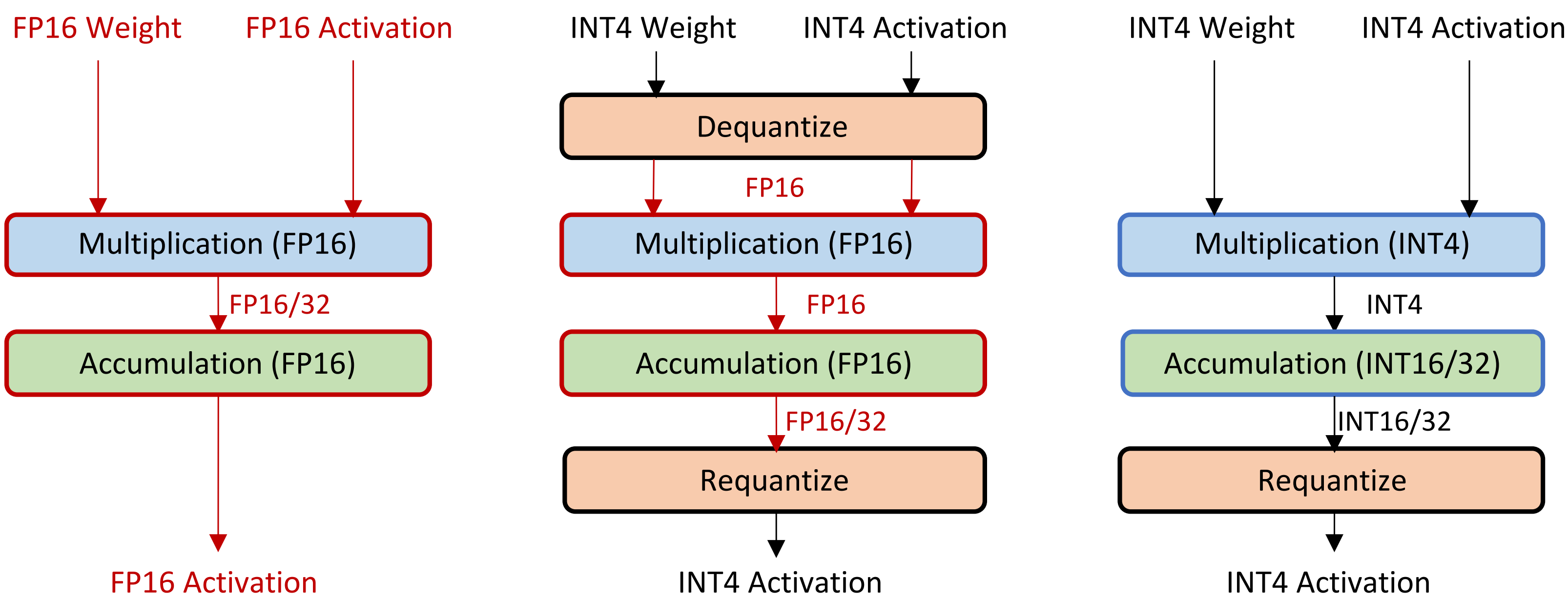
✅ Reducing memory movement directly impacts power consumption

"computing's Energy Problem, M. Horowitz, ISSCC, 2014 (Numbers are rough approximations for 45nm)
Slide: Courtesy of Prof. Shao

Outline

- **Basic Concepts of Quantization**
  - Uniform vs Non-Uniform Quantization
  - Symmetric vs Asymmetric Quantization
  - Quantization Granularity: Layer-wise vs Channel-wise
  - Dynamic vs Static Quantization
  - Post Training Quantization vs Quantization Aware Training

Quantized Inference



FP16
(Before Quantization)

INT4 Simulated Quantization
(aka fake quantization)

Integer Only Quantization

Quantization:
Workhorse for Efficient Inference

- Uniform quantization is a linear mapping from floating point values to quantized integer values

0.34	3.75	5.64
1.12	2.7	-0.9
-4.7	0.68	1.43

FP32
(Before Quantization)

Quantize

-3	81	127
16	55	-34
-128	5	23

INT8
(After Quantization)

Quantization:
Workhorse for Efficient Inference

r

0.34	3.75	5.64
1.12	2.7	-0.9
-5.64	0.68	1.43

FP32
(Before Quantization)

$$Q(r) = \text{Int}\left(\frac{r}{S}\right)$$

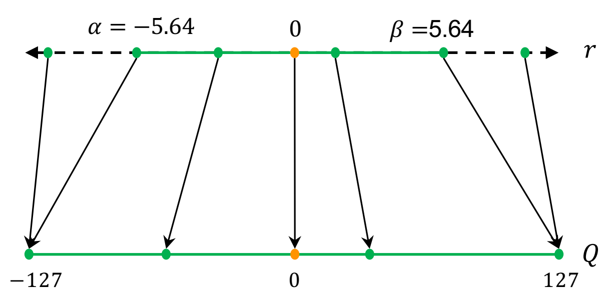
$$S = \frac{\beta - \alpha}{2\text{\textasciicircum}B - 1}$$

Quantize

$Q(r)$

-3	81	127
16	55	-34
-127	5	23

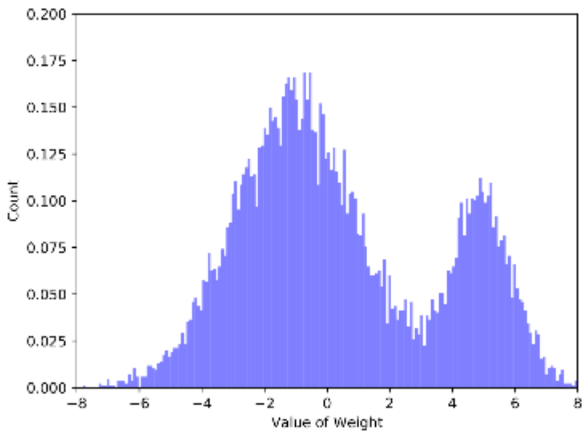
INT8
(After Quantization)

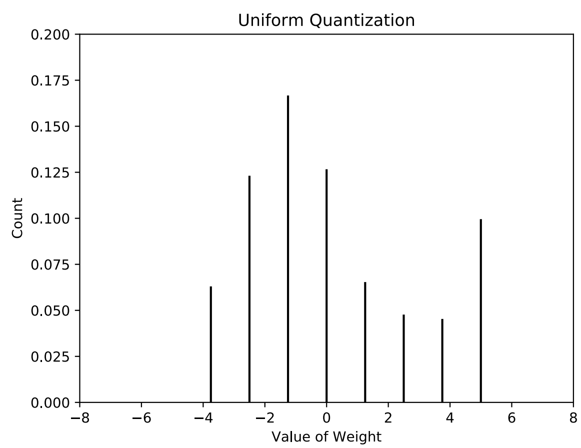


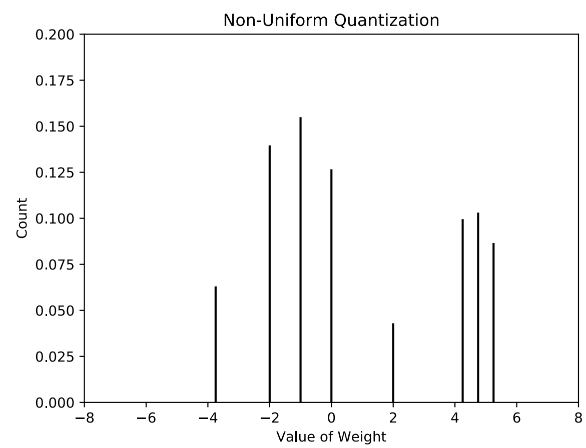
Using uniform, symmetric quantization method

Uniform vs Non-Uniform Quantization

- **Uniform Quantization**: Split range of weight values evenly
- **Non-uniform quantization**: No constraint on how the weight values are quantized

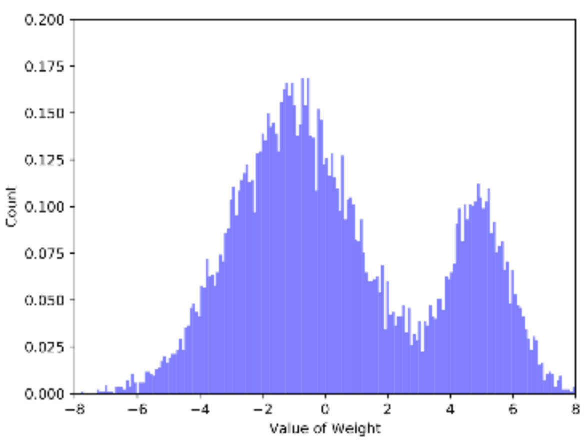


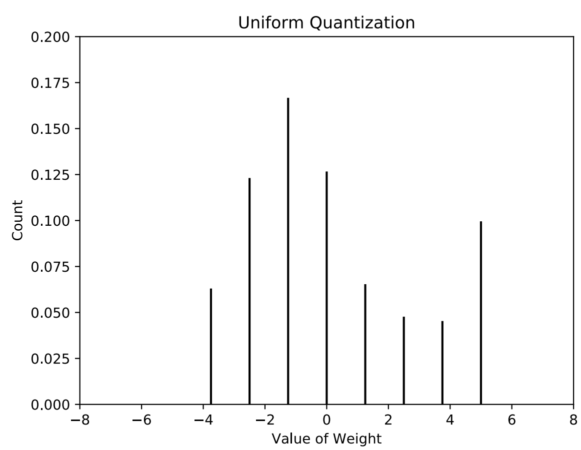


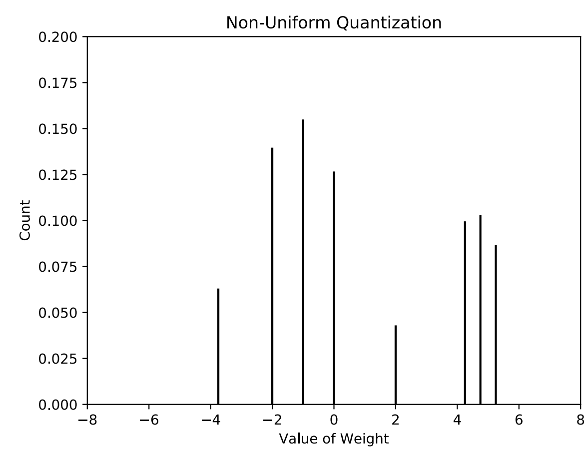


Uniform vs Non-Uniform Quantization

Uniform Quantization	Non-Uniform Quantization
Easy to utilize reduced precision ALUs	Typically requires inference arithmetic at higher precision (for example FP16)
Just requires loading scale values and Zero point	Requires a Look Up Table
Higher quantization error	Lower quantization error
Easy to implement	Typically more involved to implement/quantize

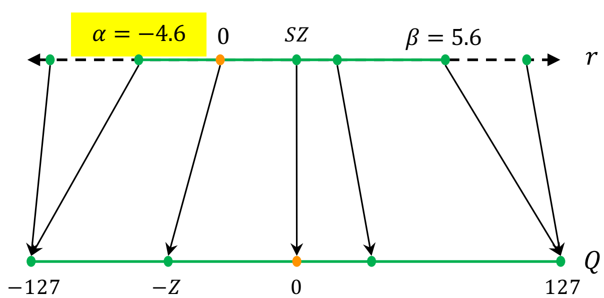






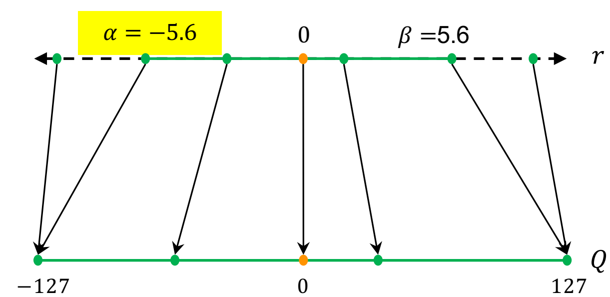
Asymmetric vs Symmetric Quantization

Asymmetric Quantization	Symmetric Quantization
Suitable for cases where min/max values are very different (e.g. activations after ReLu)	Suitable when min/max values are similar/symmetric around zero point
Typically used for activation quantization	Typically used for weight quantization
Requires storing a zero point (Z)	No zero point required (simpler to implement)



Asymmetric Quantization

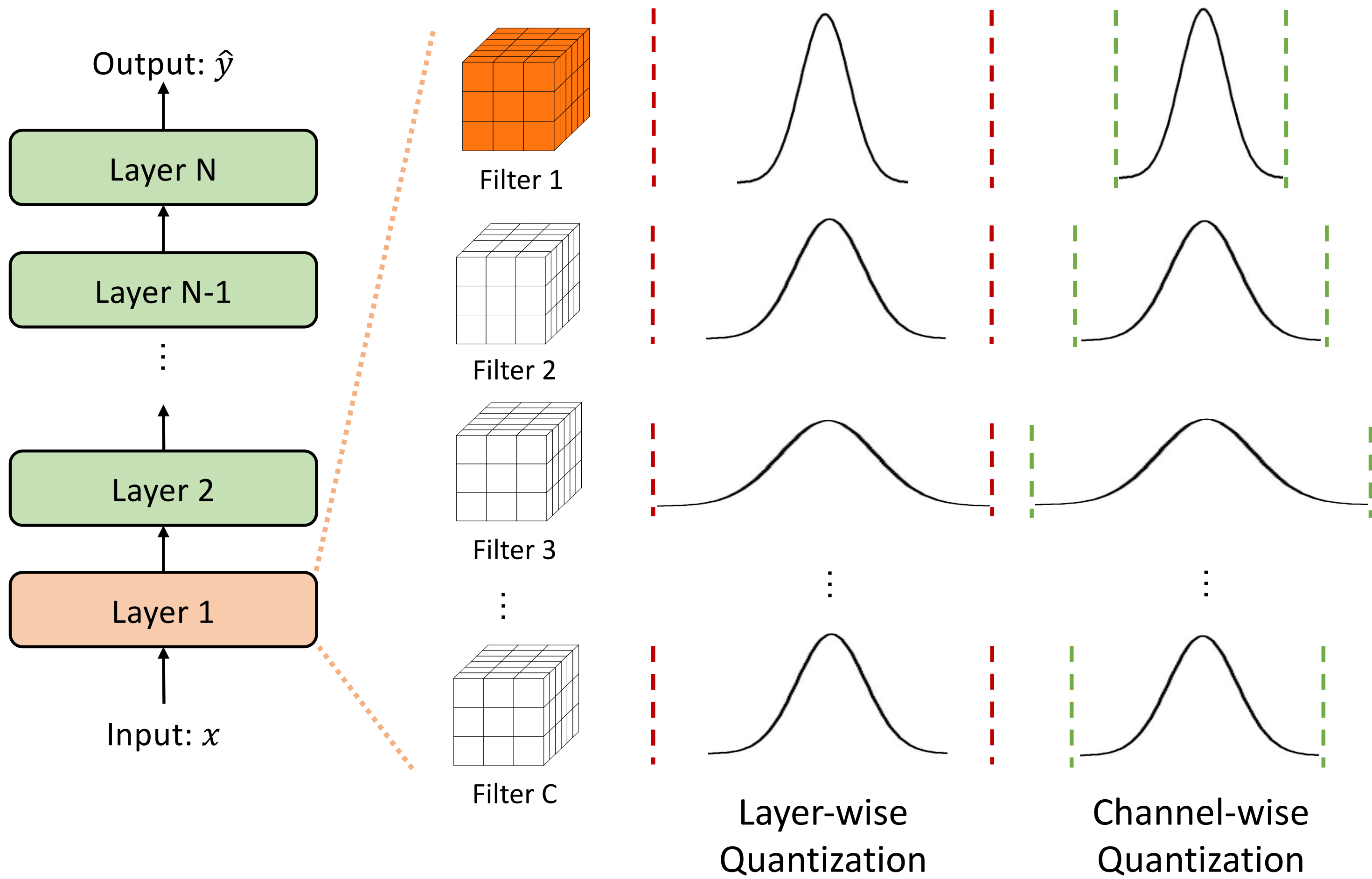
$$Q(r) = \text{Int}\big(r/S\big) - Z$$



Symmetric Quantization

$$Q(r) = \text{Int}\left(\frac{r}{S}\right)$$

Layer-Wise vs Channel-Wise Quantization

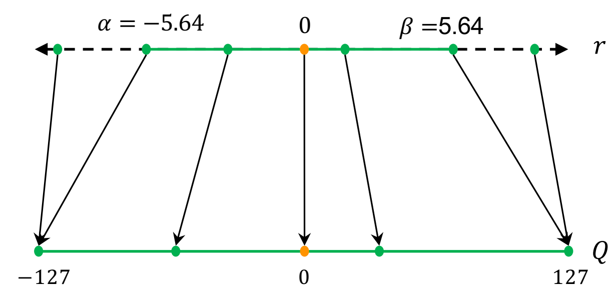


Static vs Dynamic Quantization

- How do we choose the range $[\beta, \alpha]$?
  - For weights, we know the values **statically,** since weights are fixed during inference
  - **But what about activations? We can either use static or dynamic quantization:**

$$Q(r) = \text{Int}\left(\frac{r}{S}\right)$$

$$S = \frac{\beta - \alpha}{2^B - 1}$$

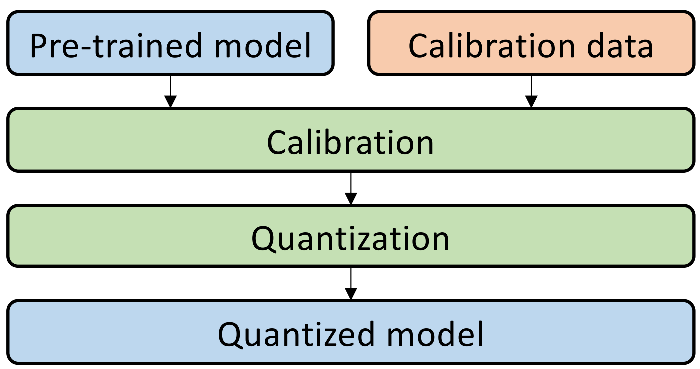


- **Static Quantization:** Choose pre-determined static range for activations independent of input
  - Very fast, low overhead, but typically not accurate since each input can have a different range
- **Dynamic Quantization:** Determine range for each activation separately during the runtime
  - Typically very slow due to the cost of computing mix/max or percentile
  - But very accurate as it exactly detects the correct range for quantization

Model Quantization Methods

The quantization schemes we talked about so far assume that we have the model parameters given to us. There are generally two approaches for getting these values:

- **Post Training Quantization** (aka training-free quantization):
  - Typically just uses the weights after normal training is finished without any extra training.
  - Variants of this approach exist where a small amount of calibration data is used to determine the network behaviour (e.g. to compute range of activations, adjusting normalization constants, and possibly even adjusting the weights without training).

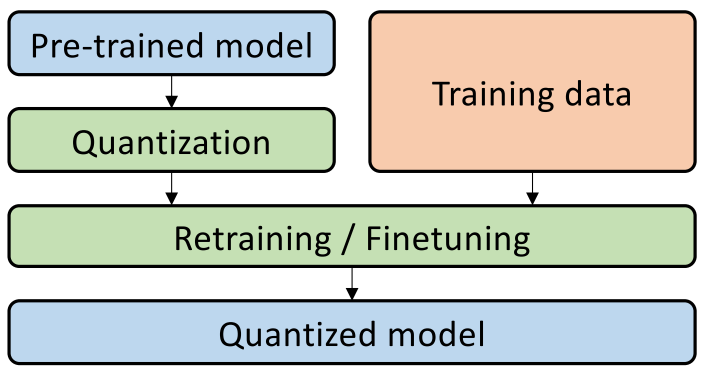


Post Training Quantization

Model Quantization Methods

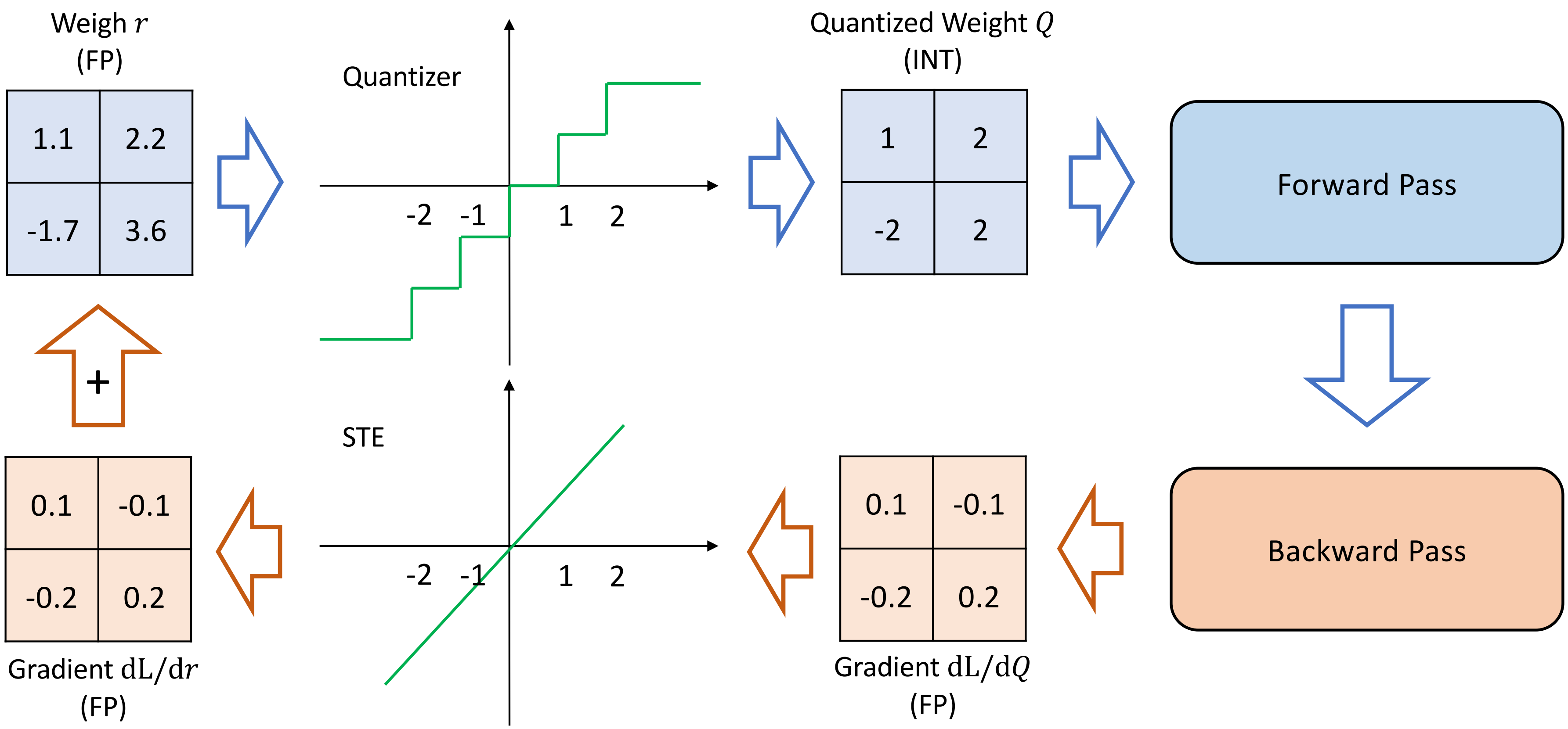
The quantization schemes we talked about so far assume that we have the model parameters given to us. There are generally two approaches for getting these values:

- **Quantization Aware Training**
  - In this approach, training is performed to adjust the weights by backpropagating the loss through the quantization operators.
  - Performing backprop requires simulated quantization along with Straight Through Estimator for rounding functions



Quantization Aware Training

Quantization Aware Training



Post Training Quantization (PTQ) vs Quantization Aware Training (QAT)

Post Training Quantization	Quantization Aware Training
Usually very fast (1-3 min)	Slow (may require hundreds of epochs)
No re-training required	Model must be retrained
Less accurate at low precisions	Typically more accurate than PTQ

Review

- **Basic Concepts of Quantization**
  - Uniform vs Non-Uniform Quantization
  - Symmetric vs Asymmetric Quantization
  - Quantization Granularity: Layer-wise vs Channel-wise
  - Dynamic vs Static Quantization
  - Post Training Quantization vs Quantization Aware Training

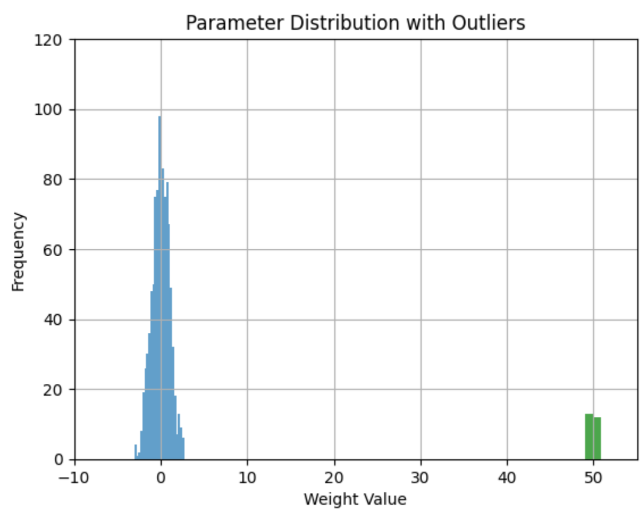
Outline

- Basic Concepts of Quantization
- **Advanced Concepts of Quantization**
  - **Dense and Sparse Quantization**
  - Mixed-Precision Quantization

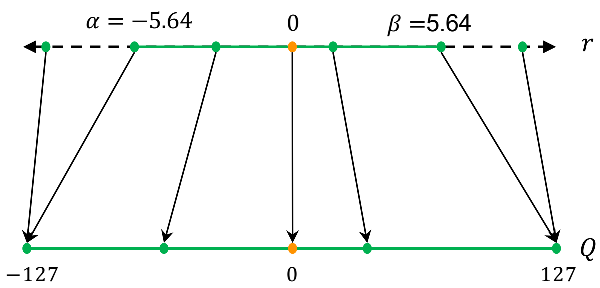
New LLMs have Significant Outliers

- **Weight distribution analysis of LLaMA-7B Model**
  - **Range of the weight values** in the Output (MHA) and Down (FFN) projection layers
  - Around **99.99%** of the values are in the **10-20%** of the overall range

- **Outliers over-exaggerate the quantization range**



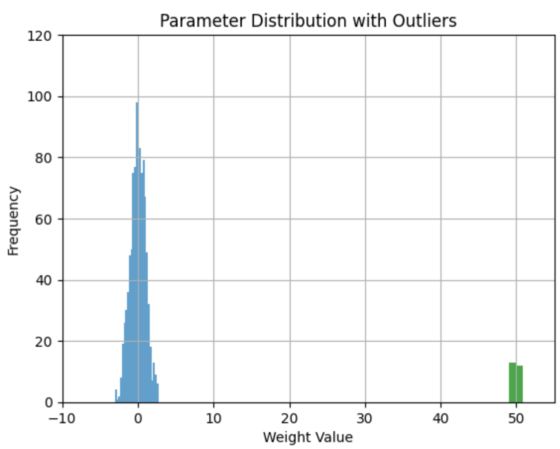
$$S = \frac{\beta - \alpha}{2\text{^}B - 1} \qquad Q(r) = \mathrm{Int}\left(\frac{r}{S}\right)$$



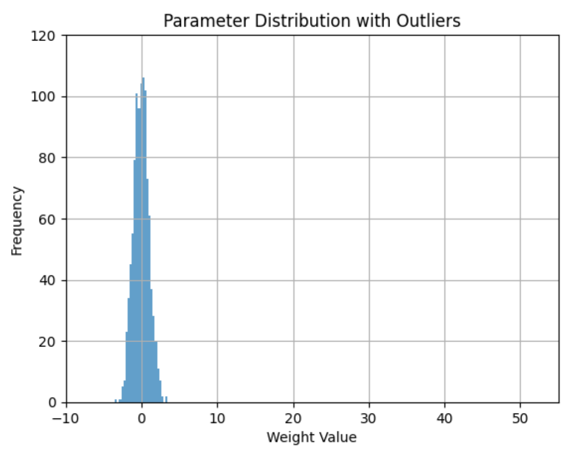
Dense-and-Sparse Quantization

- Decompose a matrix into a **dense matrix** and a **sparse matrix**

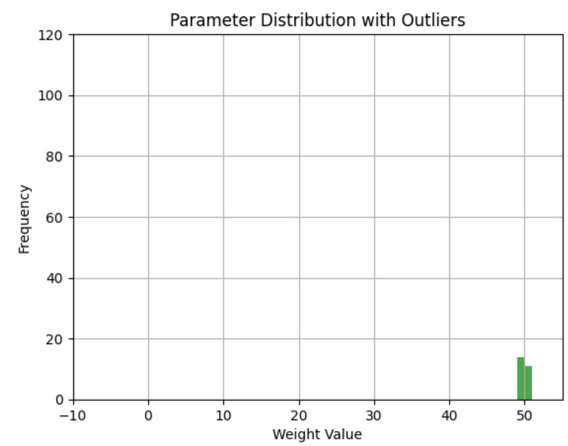
$$W = (D + S)$$



=



+



S. Kim*, C. Hooper*, A. Gholami*, Z. Dong, X. Li, S. Sheng, M. Mahoney, K. Keutzer, SqueezeLLM: Dense-and-Sparse Quantization, arxiv: :2306.07629.

Dense-and-Sparse Decomposition

- Decompose a matrix into a **dense matrix** and a **sparse matrix**

$$W = \boxed{D} + \boxed{S}$$

Dense matrix: reduced range
→ smaller quantization error

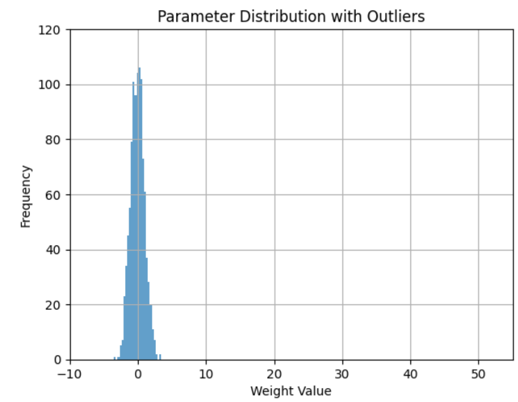
Sparse matrix: ~0.1% outliers

$$A = \begin{pmatrix} 7.5 & 2.9 & 2.8 & 2.7 & 0 & 0 \\ 6.8 & 5.7 & 3.8 & 0 & 0 & 0 \\ 2.4 & 6.2 & 3.2 & 0 & 0 & 0 \\ 9.7 & 0 & 0 & 2.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5.8 & 5.0 \\ 0 & 0 & 0 & 0 & 6.6 & 8.1 \end{pmatrix}$$

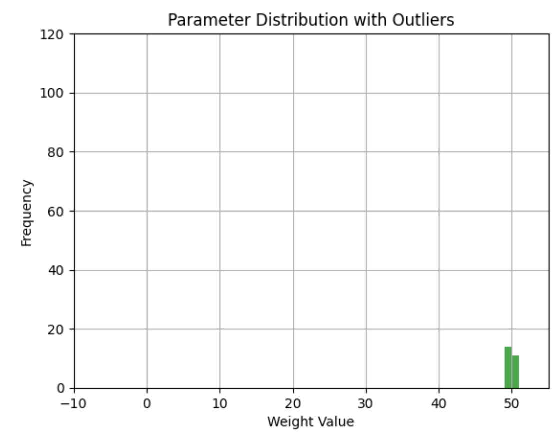
```
rowptr:  ( 0   4   7   10  12  14  16 )
colind:  ( 0   1   2   3   0   1   2   0   1   2   0   3   4   5   4   5 )
val:     ( 7.5 2.9 2.8 2.7 6.8 5.7 3.8 2.4 6.2 3.2 9.7 2.3 5.8 5.0 6.6 8.1 )
```

Sparse matrix representation using the compressed row storage (CSR) format

D



S



Dense-and-Sparse Decomposition

- Decompose a matrix into a **dense matrix** and a **sparse matrix**

$$Wx = (\boxed{D} + \boxed{S})x = Dx + Sx \approx \boxed{Qx} + \boxed{Sx}$$

Dense matrix: reduced range
→ smaller quantization error

Sparse matrix: ~0.1% outliers

Sparse matrix multiplication
(e.g. CuSparse)

FP16 **dense matrix multiplication**
After dequantization

$$A = \begin{pmatrix} 7.5 & 2.9 & 2.8 & 2.7 & 0 & 0 \\ 6.8 & 5.7 & 3.8 & 0 & 0 & 0 \\ 2.4 & 6.2 & 3.2 & 0 & 0 & 0 \\ 9.7 & 0 & 0 & 2.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5.8 & 5.0 \\ 0 & 0 & 0 & 0 & 6.6 & 8.1 \end{pmatrix}$$

```
rowptr:            (  0  4  7  10  12  14  16  )
colind: (  0    1    2    3    0    1    2    0    1    2    0    3    4    5    4    5   )
val:    ( 7.5  2.9  2.8  2.7  6.8  5.7  3.8  2.4  6.2  3.2  9.7  2.3  5.8  5.0  6.6  8.1 )
```

Sparse matrix representation using the compressed row storage (CSR) format

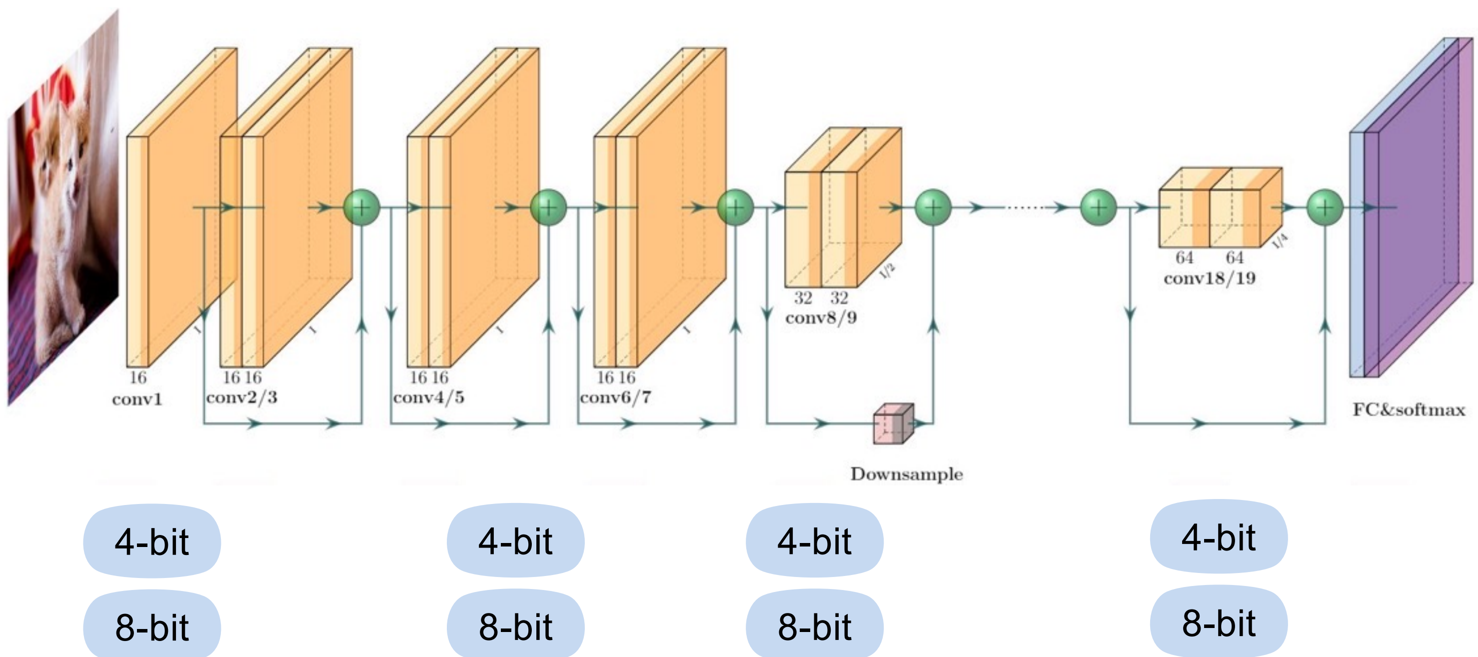
S. Kim*, C. Hooper*, A. Gholami*, Z. Dong, X. Li, S. Sheng, M. Mahoney, K. Keutzer, SqueezeLLM: Dense-and-Sparse Quantization, arxiv: :2306.07629.

Outline

- Basic Concepts of Quantization
- **Advanced Concepts of Quantization**
  - Dense and Sparse Quantization
  - **Mixed-Precision Quantization**

Mixed Precision Quantization

How can we perform low precision quantization with minimal generalization loss?

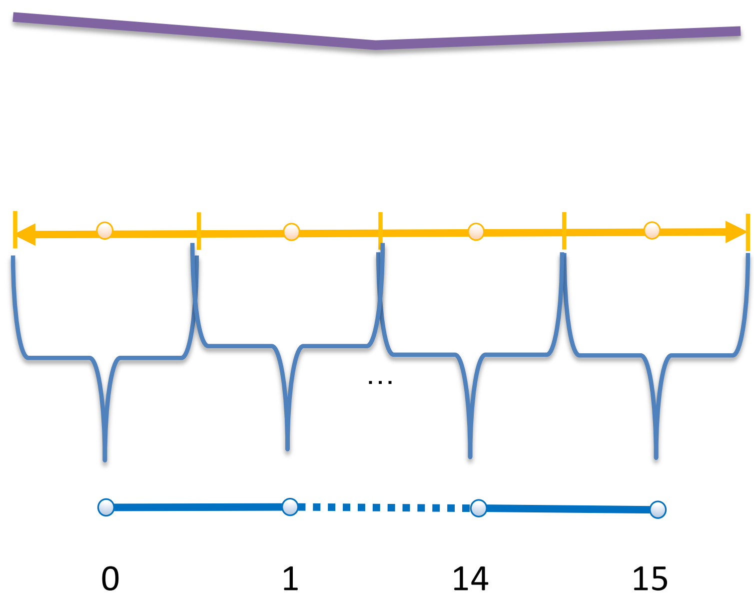


Uniform low precision does not work as it can significantly degrade accuracy

➢ **Use mixed-precision ==> How to determine mixed precision? Exponential search space**

Flat Loss Landscape → Low Bit Precision

- Uniform quantization is a linear mapping from floating point values to quantized integer values



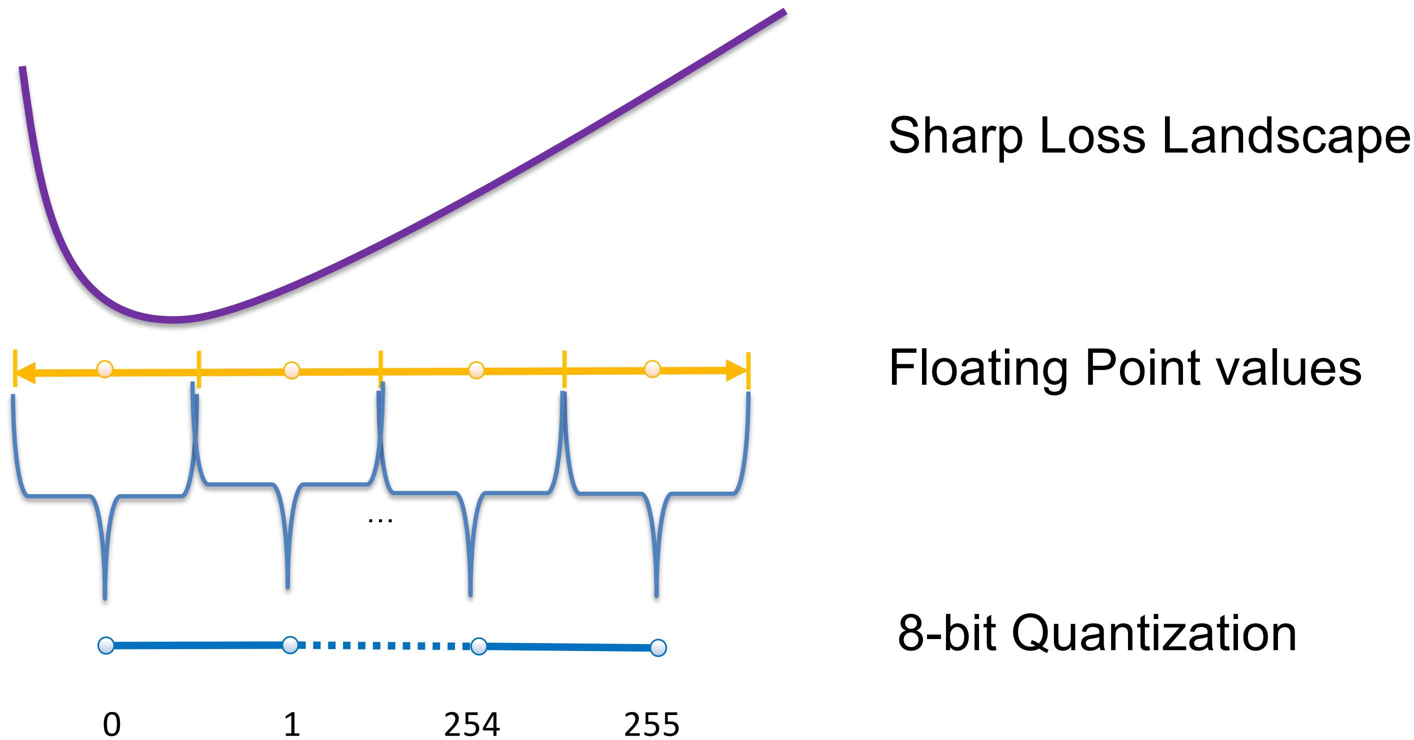
Flat Loss Landscape

Floating Point values

4-bit Quantization

Sharp Loss Landscape → High Bit Precision Needed

- Uniform quantization is a linear mapping from floating point values to quantized integer values



Hessian Aware Quantization

This is somewhat similar to the **Jenga** game. We only remove blocks that are not sensitive.

- Only use low precision quantization for insensitive parameters (flat loss landscape)
- Use high precision quantization for sensitive parameters (sharp loss landscape)

This sensitivity can be calculated through Hessian which quantifies the relative sharpness/flatness of the loss landscape.

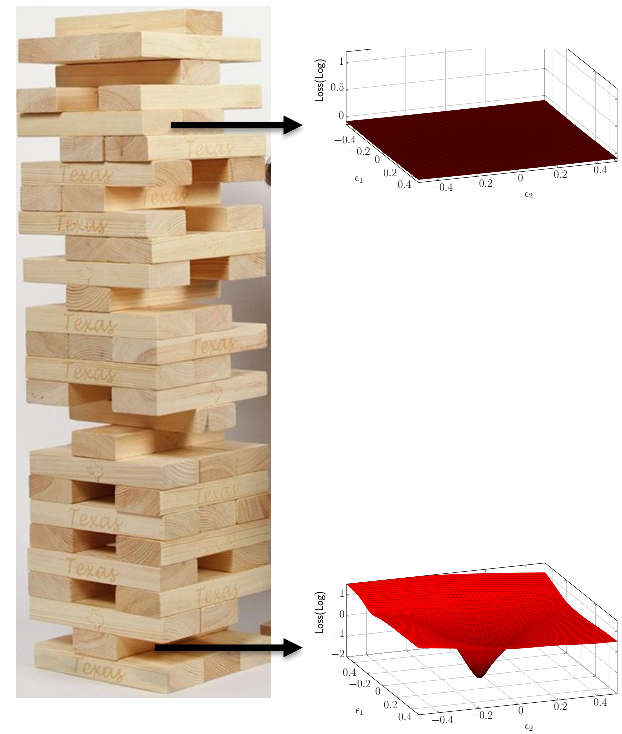
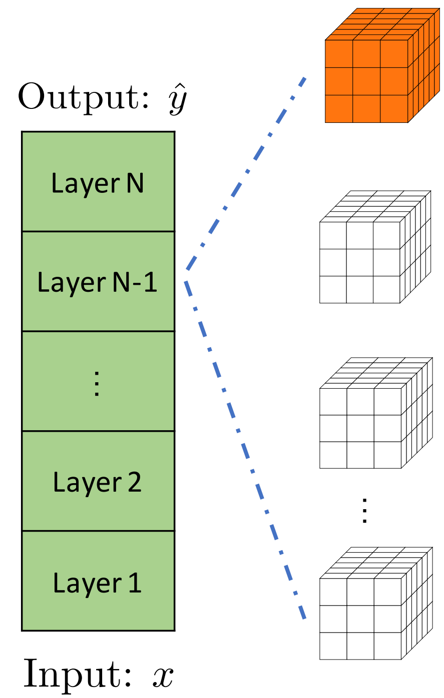
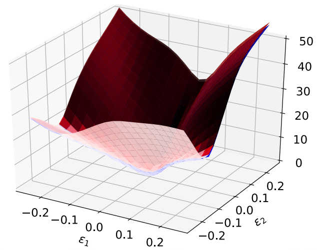


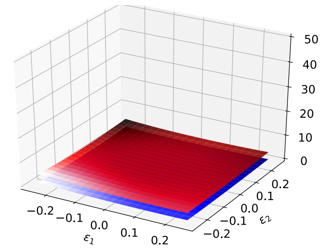
Image from UniversityCoop

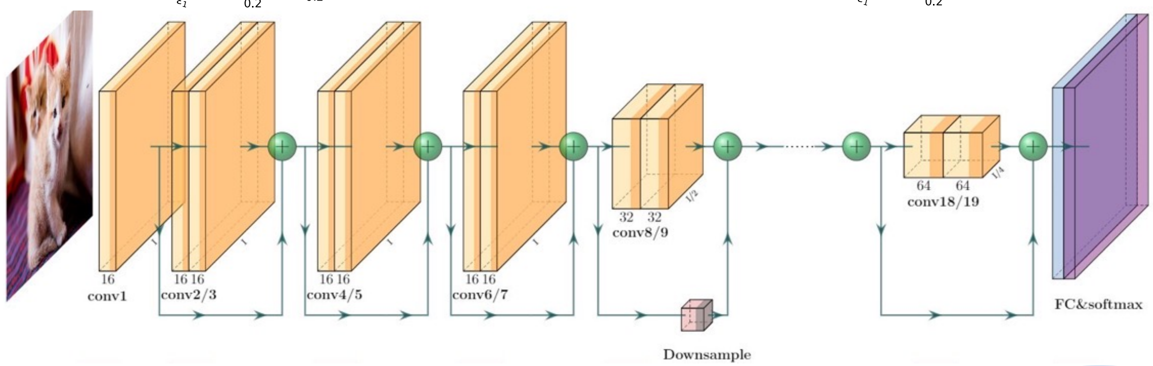


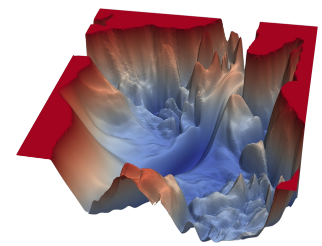
Dong Z, Yao Z, Arfeen D, **Gholami A**, Mahoney MW, Keutzer K. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. **NeurIPS**, 2020.
Yu S*, **Gholami A***, Yao Z*, Dong Z*, Mahoney MW, Keutzer K. Hessian-Aware Pruning and Optimal Neural Implant. **WACV**, 2022.

Using Hessian to Guide Choice of Bit Precision Layer by Layer









Training Loss

4-bit | 4-bit | 4-bit | 4-bit

8-bit | 8-bit | 8-bit | 8-bit

Z. Yao*, Z. Dong*, Z. Zheng*, **A. Gholami***, E. Tan, J. Li, L. Yuan, Q. Huang, Y. Wang, M. W. Mahoney, K. Keutzer, HAWQ-V3: Dyadic Neural Network Quantization in Mixed Precision, **ICML, 2021**.

Dong Z, Yao Z, Arfeen D, **Gholami A**, Mahoney MW, Keutzer K. Hawq-V2: Hessian aware trace-weighted quantization of neural networks. **NeurIPS, 2020.**

Dong Z*, Yao Z*, **Gholami A***, Mahoney MW, Keutzer K. HAWQ: Hessian AWare Quantization of neural networks with mixed-precision. **ICCV, 2019.**

Full Stack Approach for Efficient Conversational AI

Data

- **PowerNorm:** ICML’20
- **SqueezeFormer: NeurIPS’22**
- **BiLD: In Review**

NN Architecture

NN Compression

- **HAWQ-V3, I-BERT:** ICML’21 (Oral)
- **Optimal Neural Implant:** WACV’21
- **Zero-shot/integer-only ASR:** ICASSP’22
- **Learned Token Pruning:** KDD’22
- **Post-training Pruning:** NeurIPS’22
- **SqueezeLLM**: This Work

HW

- **SqueezeNext**: CVPR’18
- **Genisys**: ISCA’23 Workshop

Thanks for Listening

Please reach out if you had any feedback/questions:

amirgh@berkeley.edu

Further Reading:

- Gholami A, Kim S, Dong Z, Yao Z, Mahoney MW, Keutzer K. **A survey of quantization methods for efficient neural network inference**. In Low-Power Computer Vision 2022.
- Kim S, Hooper C, Wattanawong T, Kang M, Yan R, Genc H, Dinh G, Huang Q, Keutzer K, Mahoney MW, Shao YS. **Full stack optimization of transformer inference: a survey**. Workshop on Architecture and System Support for Transformer Models (ASSYST) at ISCA 2023.

HOT CHIPS

Berkeley UNIVERSITY OF CALIFORNIA