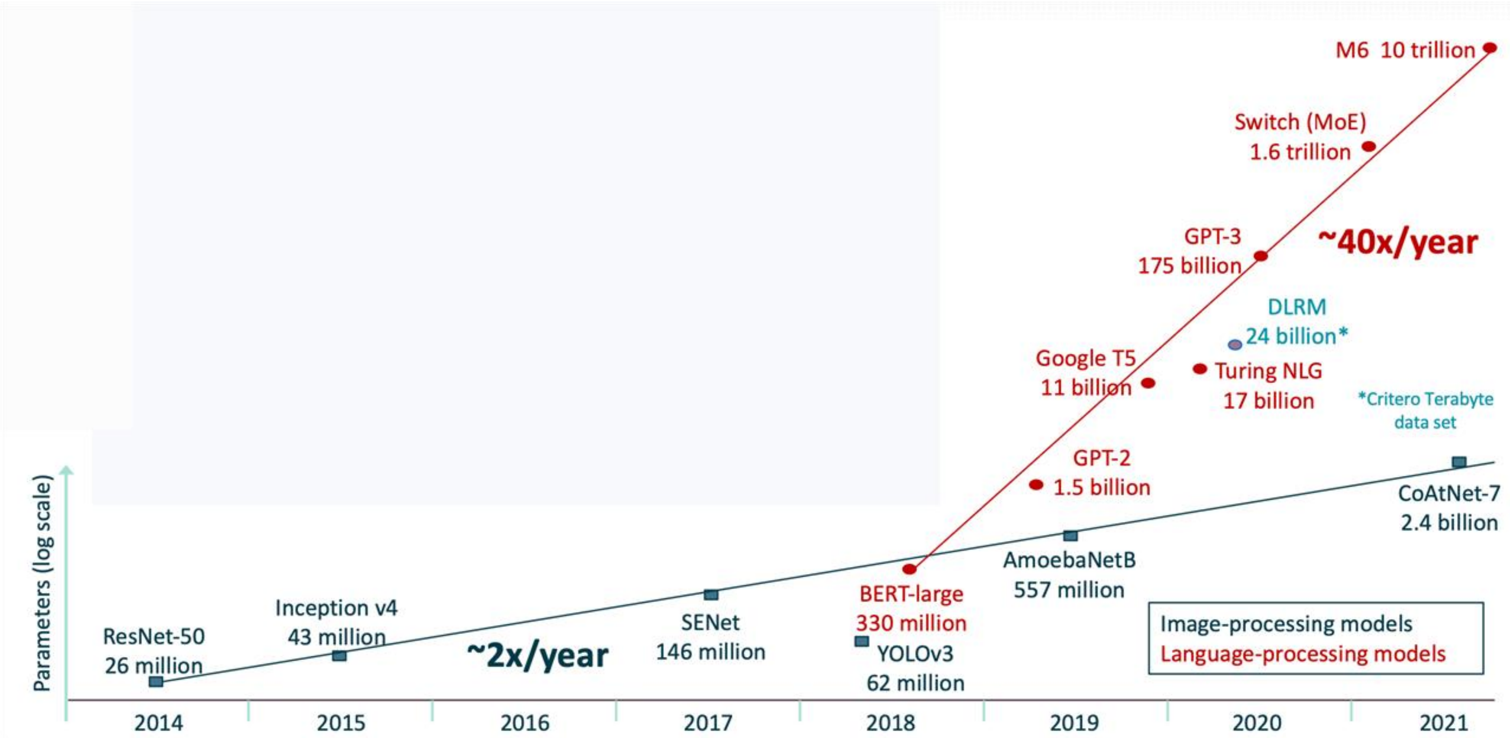
AI VIETNAM
All-in-One Course

Model Quantization

Bach-Hoang Ngo

Year 2024

Motivation


M6 10 trillion
Switch (MoE)
1.6 trillion
GPT-3
175 billion
~40x/year
DLRM
24 billion*
Google T5
11 billion
Turing NLG
17 billion
*Critero Terabyte
data set
GPT-2
1.5 billion
CoAtNet-7
2.4 billion
AmoebaNetB
557 million
BERT-large
330 million
YOLOv3
62 million
SENet
146 million
Inception v4
43 million
ResNet-50
26 million
~2x/year
Parameters (log scale)
Image-processing models
Language-processing models
2014
2015
2016
2017
2018
2019
2020
2021

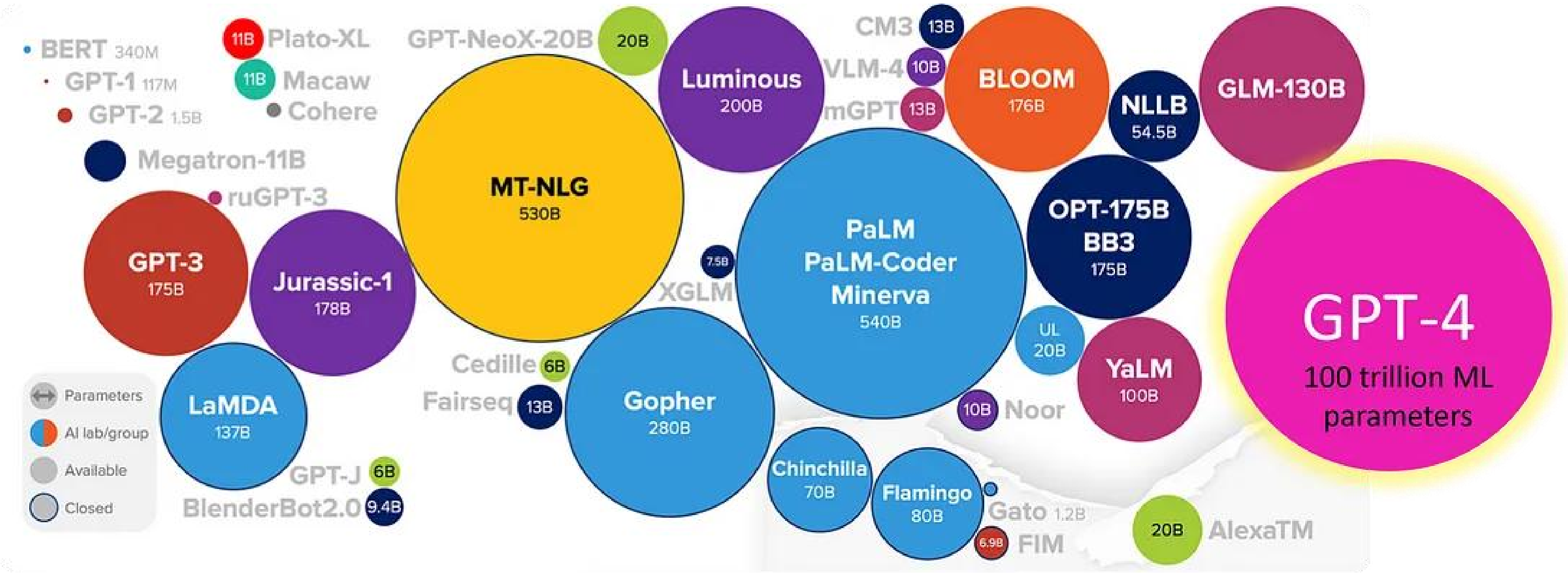
Outline

- Motivation
- Floating Point
- Uniform Quantization
- Non-Uniform Quantization

Outline

- Motivation
- Floating Point
- Uniform Quantization
- Non-Uniform Quantization

Motivation

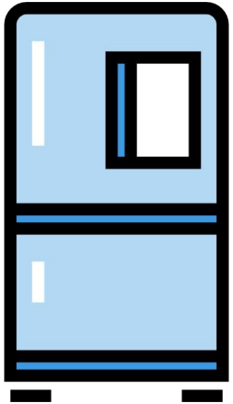

BERT 340M
GPT-1 117M
GPT-2 1.5B
Megatron-11B
11B Plato-XL
11B Macaw
Cohere
ruGPT-3
GPT-NeoX-20B 20B
MT-NLG 530B
Luminous 200B
CM3 13B
VLM-4 10B
mGPT 13B
BLOOM 176B
NLLB 54.5B
GLM-130B
GPT-3 175B
Jurassic-1 178B
7.5B XGLM
PaLM
PaLM-Coder
Minerva
540B
OPT-175B
BB3
175B
UL 20B
YaLM 100B
GPT-4
100 trillion ML parameters
Parameters
AI lab/group
Available
Closed
LaMDA 137B
Cedille 6B
Fairseq 13B
Gopher 280B
10B Noor
GPT-J 6B
BlenderBot2.0 9.4B
Chinchilla 70B
Flamingo 80B
Gato 1.2B
6.9B FIM
20B AlexaTM

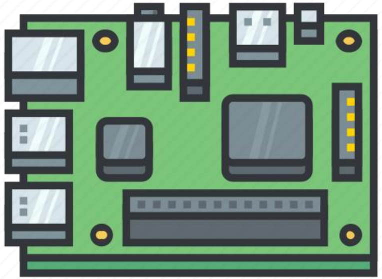
Motivation

VRAM Usage: Model Size * 2 (Loading with FP16)

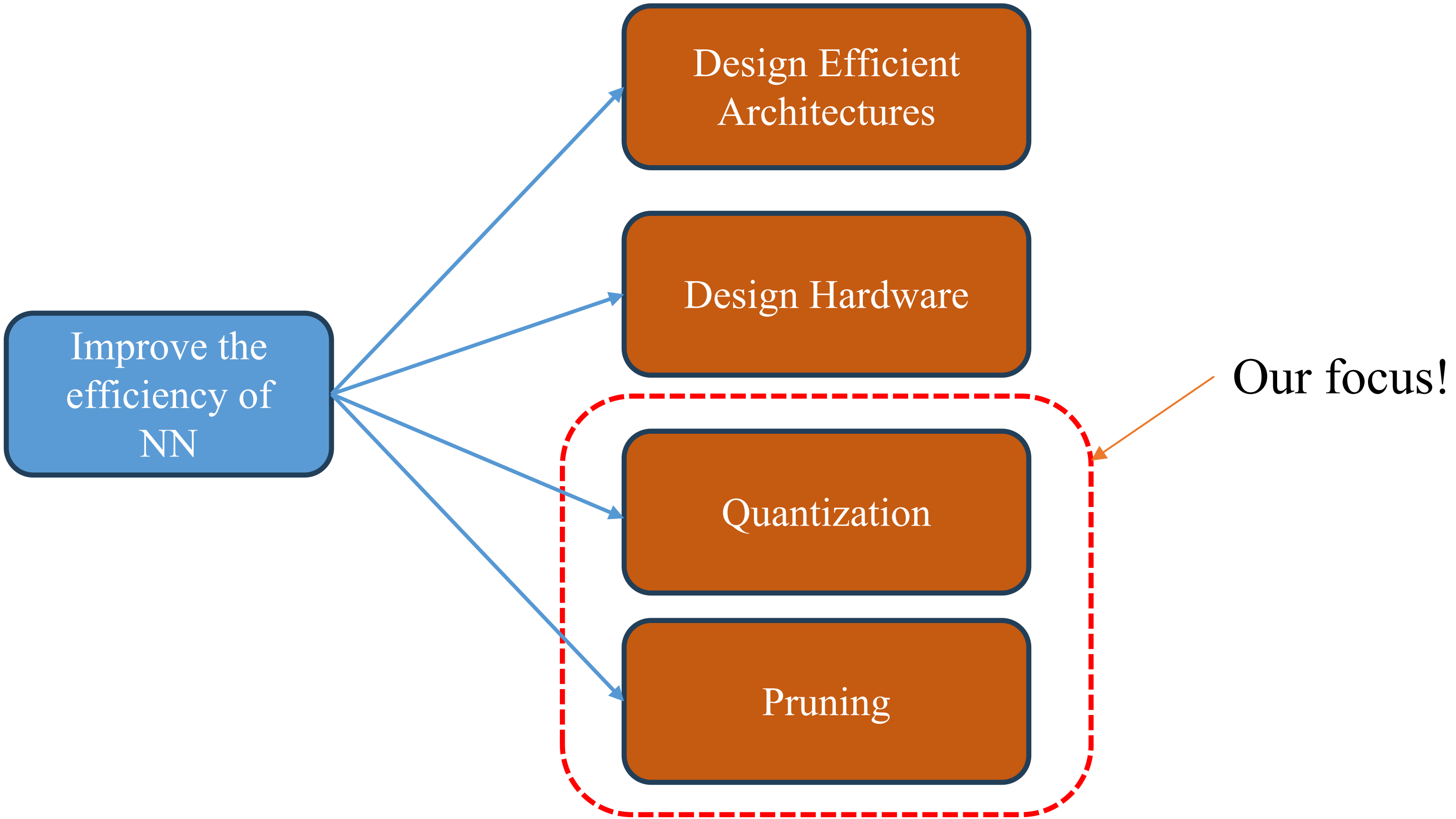
Model	Size	VRAM Usage
GPT3	175B	350 GB
Bloom	176B	352GB
Llama-2-70B	70B	140GB
Mistral	7B	14GB

Motivation





Motivation


Improve the efficiency of NN
Design Efficient Architectures
Design Hardware
Quantization
Pruning
Our focus!

Quantization

Floating Point

-0.2	1	0.3
0.1	-0.6	-0.7
1.2	0.4	0

32 bit

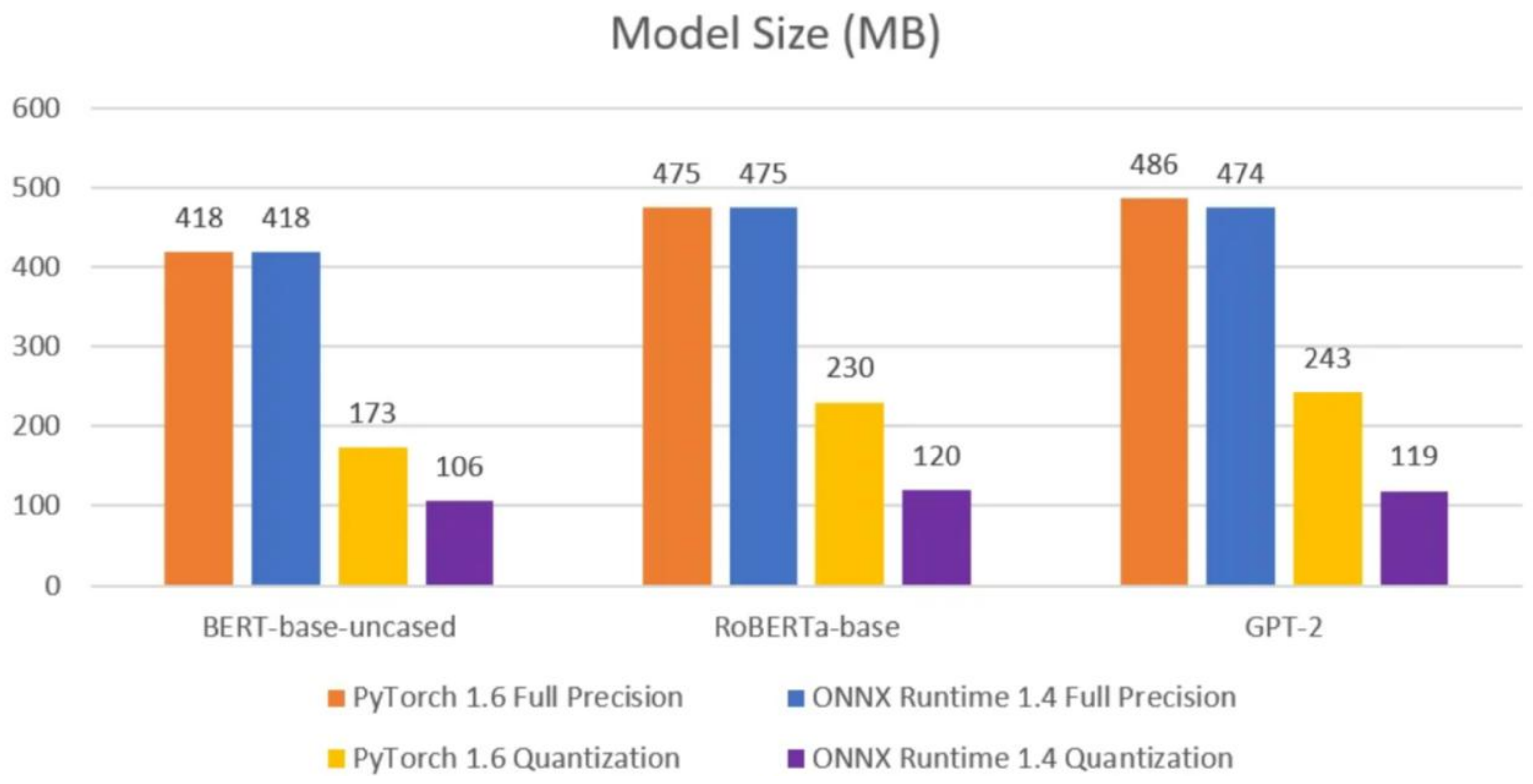
Quantization

Integer

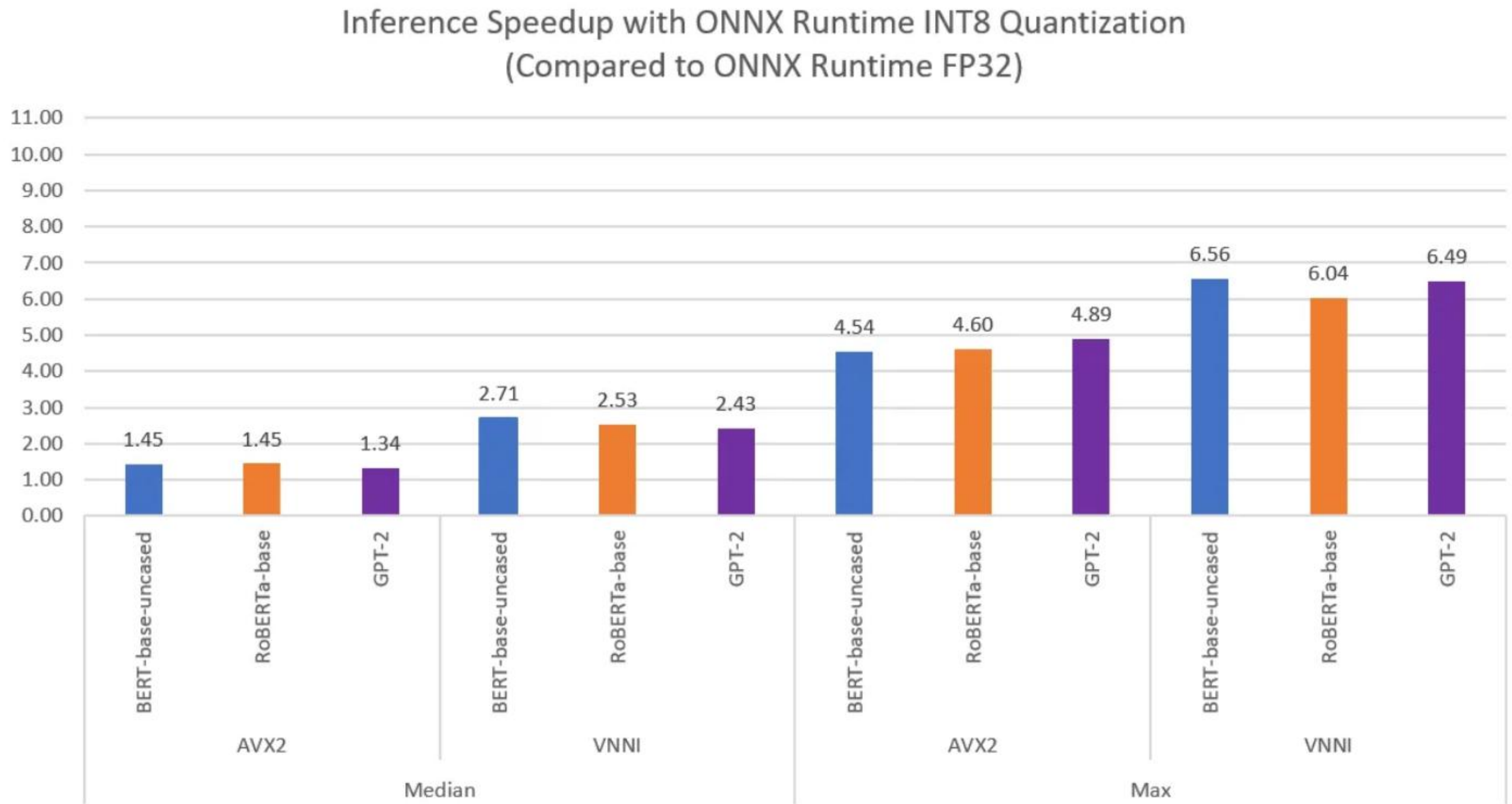
1	3	2
1	0	0
3	2	1

8 bit

Quantization



Quantization



Quantization

- Motivation
- **Floating Point**
- Uniform Quantization
- Non-Uniform Quantization

Representing numbers

Representing Real Numbers: Decimal

10^2	10^1	10^0	.	10^{-1}	10^{-2}	10^{-3}
d_2	d_1	d_0	.	d_{-1}	d_{-2}	d_{-3}
1	2	3	.	4	5	6

Representing numbers

Representing Real Numbers: Decimal

10^m	10^{m-1}	$\ldots$	10^0	.	10^{-1}	10^{-2}	$\ldots$	10^{-n}
d_m	d_n	$\ldots$	d_0	.	d_{-1}	d_{-2}	$\ldots$	d_{-3}

$$d = \sum_{i=-n}^{m} 10^i * d_i$$

Representing numbers

Representing Real Numbers: Binary

2^m	2^{m-1}	…	2^0	.	2^{-1}	2^{-2}	…	2^{-n}
b_m	b_n	…	b_0	.	b_{-1}	b_{-2}	…	b_{-3}

$$b = \sum_{i=-n}^{m} 2^i * b_i$$

Fixed-point

Where to put the point?

0 0 0 0 0 1 0 1 0 1 1 1 1 1 0 0 1 1 0 1 0 0 0 1 1

More Precision

Better Range

12345678.9101112

redundant → 00000000.0000121

12300000.0000000 ← redundant

Any way to make the point moving?

Fixed-point

Where to put the point?

12300000.0000000 → $1.23 * 10^7$

00000000.0000121 → $1.21 * 10^{-5}$

For Binary

11000000.0000000 → $1.1 * 2^7$

00000000.0000011 → $1.1 * 2^{-6}$

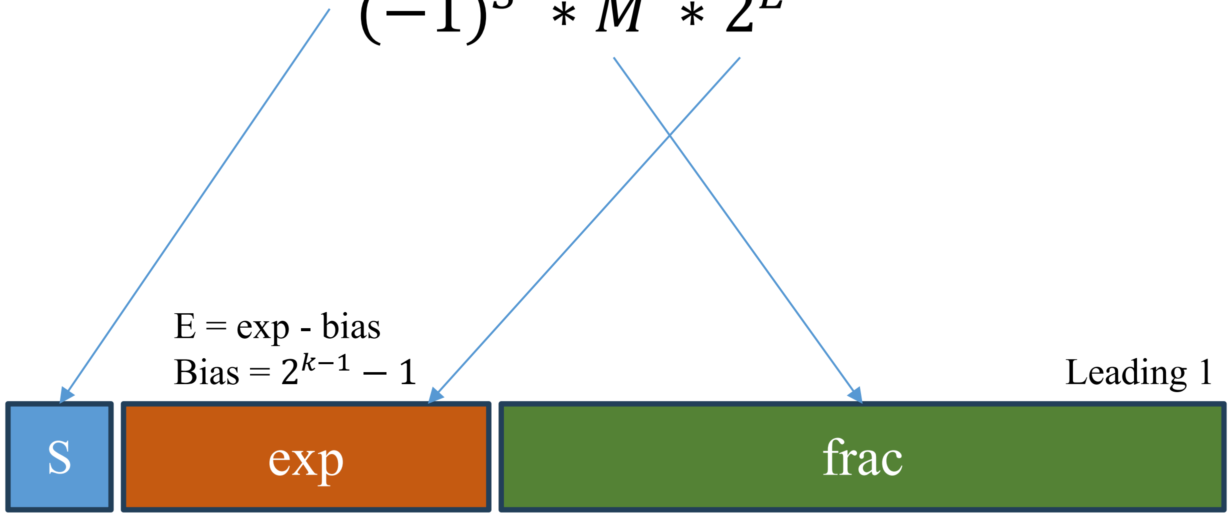
Indicates where to put the point

$$(-1)^S * M * 2^E$$

AI VIETNAM
All-in-One Course

Floating Point

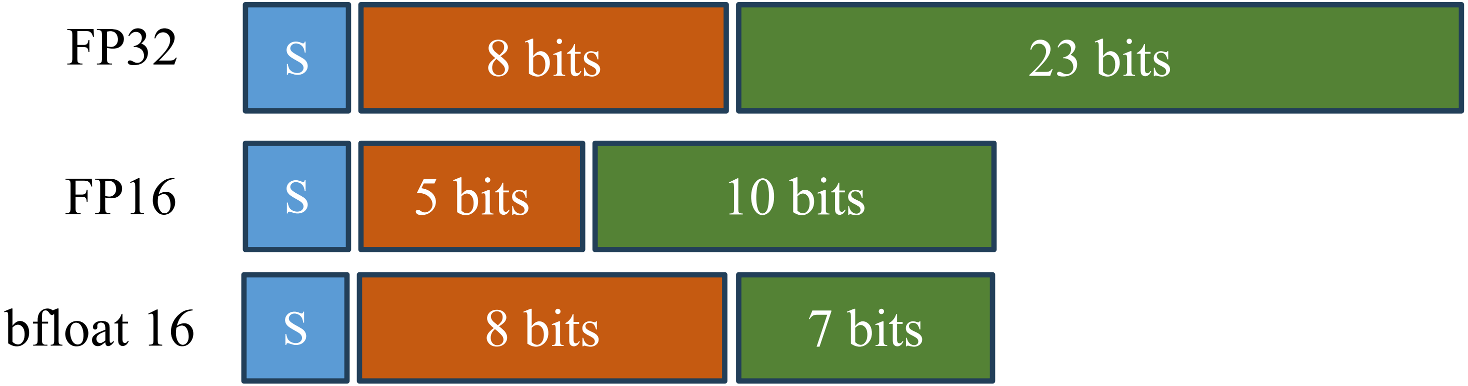
$$(-1)^S * M * 2^E$$



Sign bit

Floating Point

$$(-1)^S * M * 2^E$$



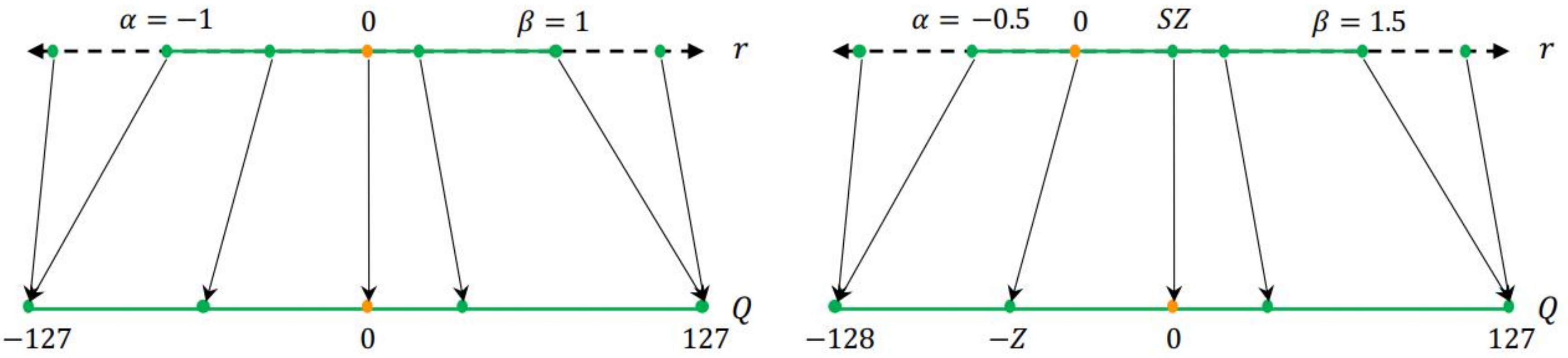
Floating Point

Data type	Range	Precision
FP32	1.18 x 10^-38 to 3.40 x 10^38	6 – 9 decimal digits
FP16	4.88 x 10^-4 to 6.55 x 10^4	5 – 6 decimal digits
Bfloat16	1.18 x 10^-38 to 3.40 x 10^38	3 decimal digits

Quantization

- Motivation
- Floating Point
- **Uniform Quantization**
- Non-Uniform Quantization

Uniform Quantization



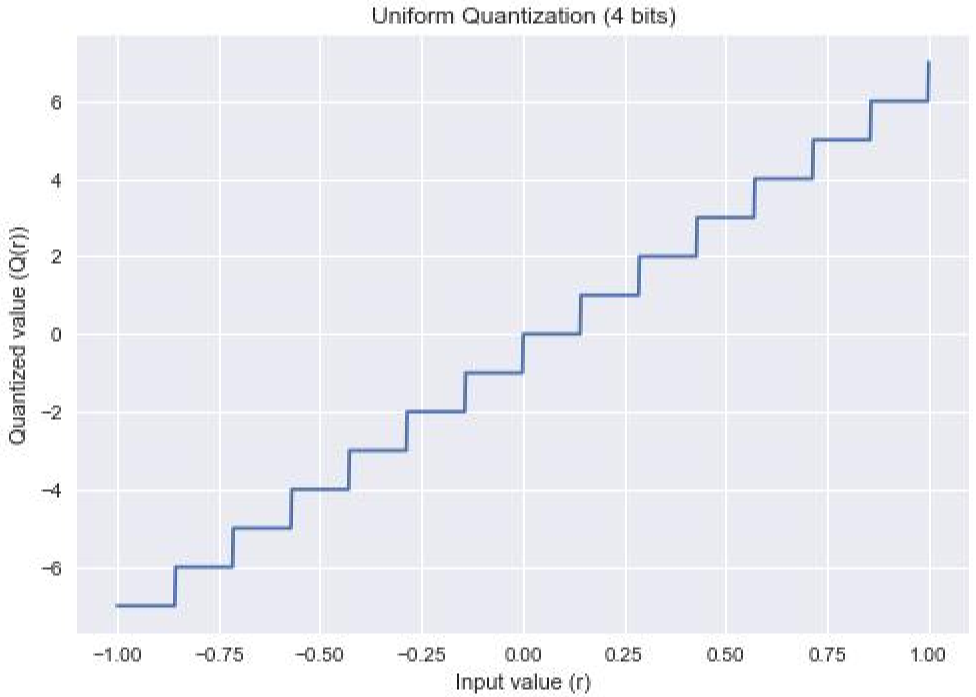
$$Q(r) = Int\left(\frac{r}{S}\right) - Z$$

Zero Value

Scaling Factor

Uniform Quantization

Example $S = 1/127, \ Z = 0$

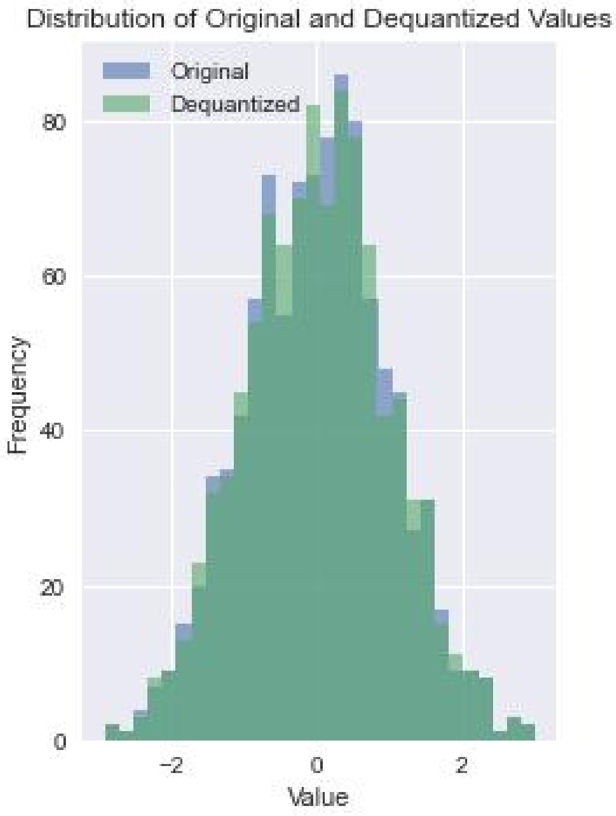


Dequantization

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

$$\rightarrow Q(r) + Z = Int\left(\frac{r}{S}\right)$$

$$\rightarrow r \approx S * (Q(r) + Z)$$



Scaling Factor

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

S: $scaling\ factor$
$[\alpha,\ \beta]$: $clippping\ rage$
b: $bit\ width$

For 8-bit quantization

$$S = \frac{\beta - \alpha}{255}$$

How to find $\beta\ and\ \alpha$?

Calibration

Calibration: the process of choosing the clipping range

Simple approach: Choosing the min/max of the signal

Assymetric Quantization

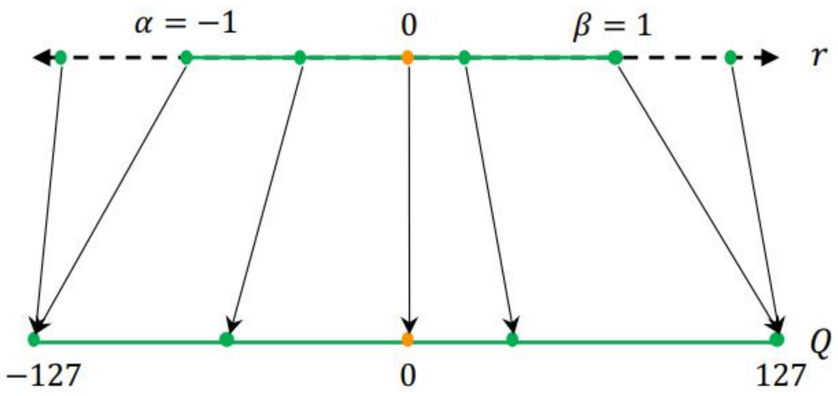
$$\alpha = r_{min}$$
$$\beta = r_{max}$$
$$Z \,! = 0$$

Symetric Quantization

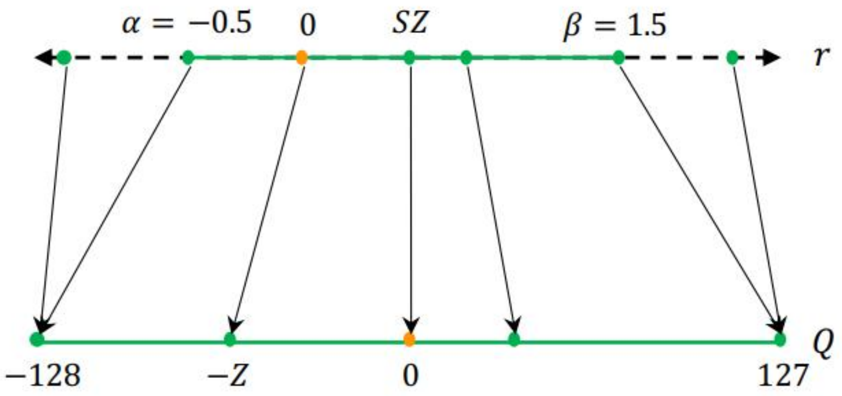
$$-\alpha = \beta$$
$$= \max(|r_{max}|, |r_{min}|)$$
$$Z = 0$$

Uniform Quantization

Symmetric Quantization



Asymmetric Quantization



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

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

Symmetric Quantization

```python
import torch

def absmax_quantize(X):
    # Calculate scale
    scale = 127 / torch.max(torch.abs(X))

    # Quantize
    X_quant = (scale * X).round()

    # Dequantize
    X_dequant = X_quant / scale

    return X_quant.to(torch.int8), X_dequant
```

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

For symmetry S = absmax(r)/127

Symmetric Quantization

Quantize

0.3	0.6	-0.3	-0.5	0.0	0.7	-0.8	0.2

Absolute max: 0.8

Scale: $\frac{1}{S} = \frac{127}{0.8} = \ 158.75$

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

48	95	-48	-79	0	111	-127	32

Symmetric Quantization

Dequantize

48	95	-48	-79	0	111	-127	32

Absolute max: 0.8

Scale: $\frac{1}{S} = \frac{127}{0.8} = 158.75$

$$\hat{r} = Q(r) * S$$

0.30	0.49	-0.30	-0.49	0.0	0.69	-0.8	0.20

Asymmetric Quantization

$$Q(r) = Int\left(\frac{r}{S}\right) - Z \qquad S = \frac{\beta - \alpha}{255}$$

Calculate zero point

$$r_{min} \rightarrow -128$$

$$-128 = Int\left(\frac{r_{min}}{S}\right) - Z$$

$$Z = Int\left(\frac{r_{min}}{S}\right) + 128$$

Asymmetric Quantization

```python
def zeropoint_quantize(X):
    # Calculate value range (denominator)
    x_range = torch.max(X) - torch.min(X)
    x_range = 1 if x_range == 0 else x_range

    # Calculate scale
    scale = x_range / 255

    # Shift by zero-point
    zeropoint = (torch.min(X) / scale).round() + 128

    # Scale and round the inputs
    X_quant = torch.clip((X / scale - zeropoint).round(), -128, 127)

    # Dequantize
    X_dequant = (X_quant + zeropoint) * scale

    return X_quant.to(torch.int8), X_dequant
```

Symmetric Quantization

Quantize

0.3	0.6	-0.3	-0.5	0.0	0.7	-0.8	0.2

max: 0.7, min: -0.8

$$\text{Scale: } \frac{1}{S} = \frac{255}{0.7-(-0.8)} = 170$$

$$\text{Zero point: } Z = Int\left(\frac{r_{min}}{scale}\right) + 128 =- 8$$

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

59	110	-43	-77	8	127	-128	42

Symmetric Quantization

Dequantize

59	110	-43	-77	8	127	-128	42

max: 0.7, min: -0.8

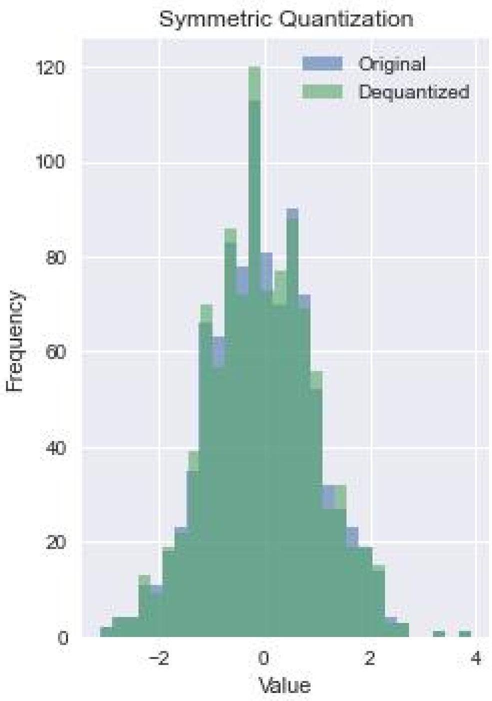
Scale: $\frac{1}{S} = \frac{255}{0.7-(-0.8)} = 170$

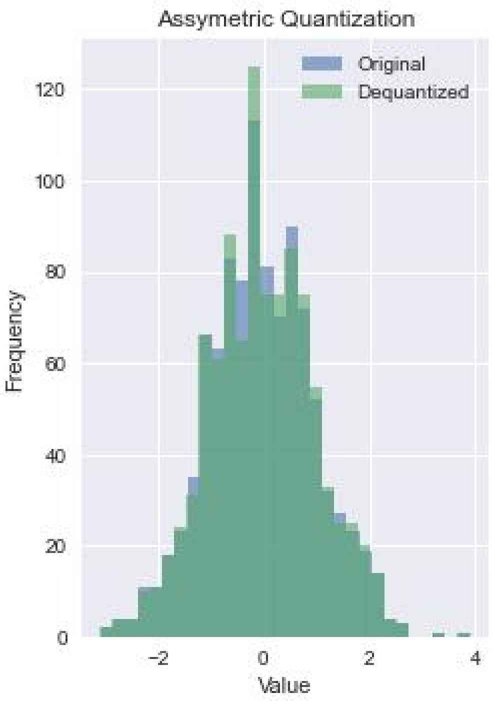
Zero point: $Z = Int\left(\frac{r_{min}}{scale}\right) + 128 =- 8$

$\hat{r} = S * (Q(r) + Z)$

0.30	0.60	-0.30	-0.50	0.0	0.70	-0.80	0.20

Asymmetric vs. Symmetric



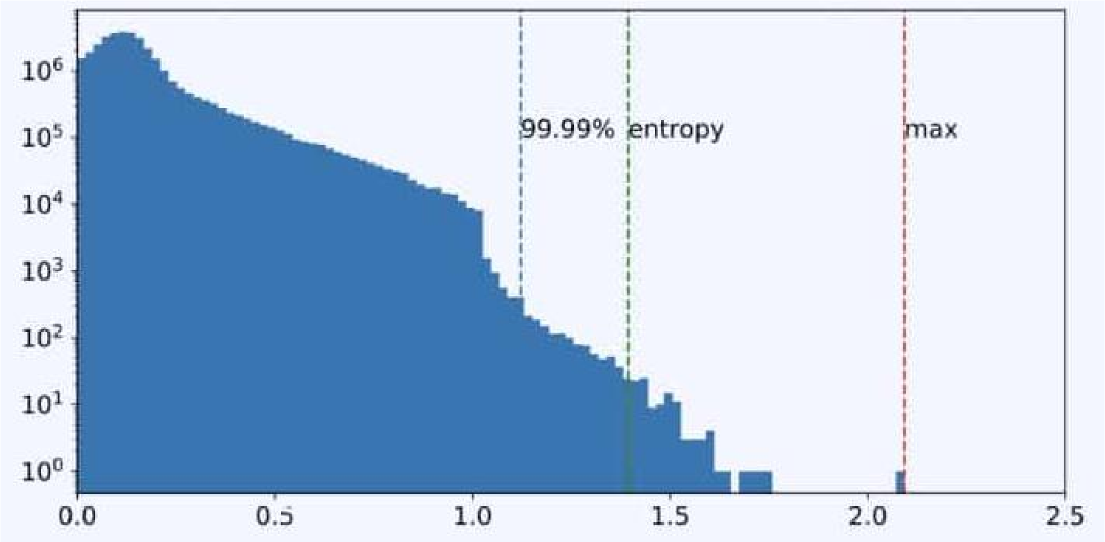


Asymmetric vs. Symmetric

Feature	Symmetric Quantization	Asymmetric Quantization
Implementation	Simple Implementation	More complex
Computational cost	Simpler calculation	Slightly higher
Efficiency for non-uniform distributions	Less efficient	More efficient

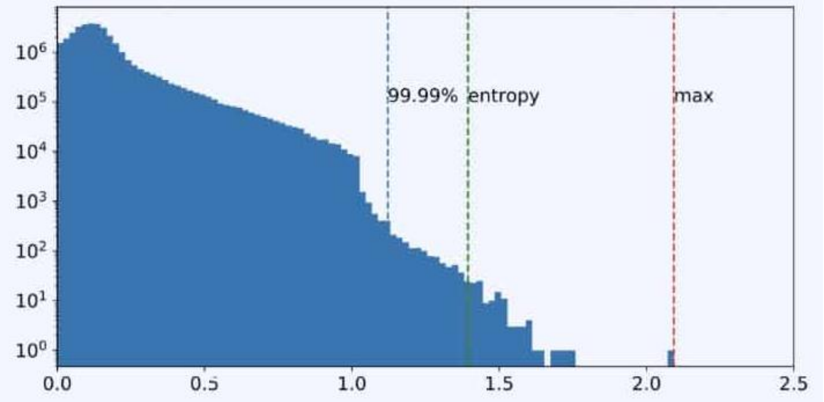
Calibration

Is choosing max/min values a good option?



Calibration

Choose max absolute value -> vulnerable to outliers.
- Entropy: KL divergence to minimize information loss.
- Percentile: Set a range to a percentile of the distribution.



Calibration

Outlier

0.3	0.6	-0.3	-0.5	0.0	0.7	-99	0.2

Quantization

0	1	0	-1	0	1	-127	0

Dequantization

0.0	0.77	0.0	-0.77	0.00	0.77	-99	0.00

Calibration

Percentile Quantization

```python
def percentile_quantize(X, percentile=70):
    # Calculate the 99th percentile of the absolute values in X
    scale = torch.quantile(torch.abs(X), percentile/100.) / 127

    # Quantize
    X_quant = (X / scale).round()

    # Clamp the values to ensure they are within the int8 range after rounding
    X_quant = torch.clamp(X_quant, min=-128, max=127)

    # Dequantize
    X_dequant = X_quant * scale

    return X_quant.to(torch.int8), X_dequant
```

Calibration

Outlier

0.3	0.6	-0.3	-0.5	0.0	0.7	-99	0.2

Quantization

65	127	-65	-108	0	127	-128	43

Dequantization

0.30	0.59	-0.30	0.50	0.00	0.59	-0.59	0.19

QUIZ TIME!!!

QUIZ TIME!!!

Câu 1: Quá trình quantization là gì?
A) Chuyển đổi giá trị từ dạng phân số sang số nguyên
B) Chuyển đổi giá trị liên tục thành giá trị rời rạc / giảm số bit để biểu diễn 1 giá trị.
C) Tăng độ chính xác của dữ liệu số
D) Giảm tốc độ xử lý của hệ thống số

Câu 2 Trong machine learning, quantization thường được sử dụng để:
A) Tăng độ phức tạp của mô hình
B) Tăng độ chính xác của mô hình
C) Giảm kích thước và yêu cầu tính toán của mô hình
D) Tăng số lượng lớp trong mạng neural

QUIZ TIME!!!

4) Phương pháp quantization nào cung cấp các đặc tính sau:
- **Dễ dàng triển khai (easy to implement) hơn.**
- **Khoảng cách bằng nhau (equal spacing) giữa các mức lượng hóa**.

A) Lượng hóa đồng nhất (Uniform quantization)
B) Lượng hóa không đồng nhất (Non-uniform quantization)
C) Cả hai lượng hóa đồng nhất và không đồng nhất
D) Không phải lượng hóa đồng nhất cũng không phải không đồng nhất

5) Loại lượng hóa nào phù hợp hơn cho các tín hiệu có hầu hết các giá trị tập trung ở phía dương, xét về cả tính đơn giản và hiệu quả?

a) Lượng hóa đối xứng (Symmetric quantization)
b) Lượng hóa không đối xứng (Non-symmetric quantization)
c) Cả hai đều hoạt động tốt như nhau.
d) Không phương pháp nào phù hợp.

QUIZ TIME!!!

Câu 5: Quá trình hiệu chỉnh (calibration) trong quantization được sử dụng để làm gì?

a) Tăng độ chính xác của mô hình sau quantization

b) Giảm kích thước của mô hình mà không cần quantization

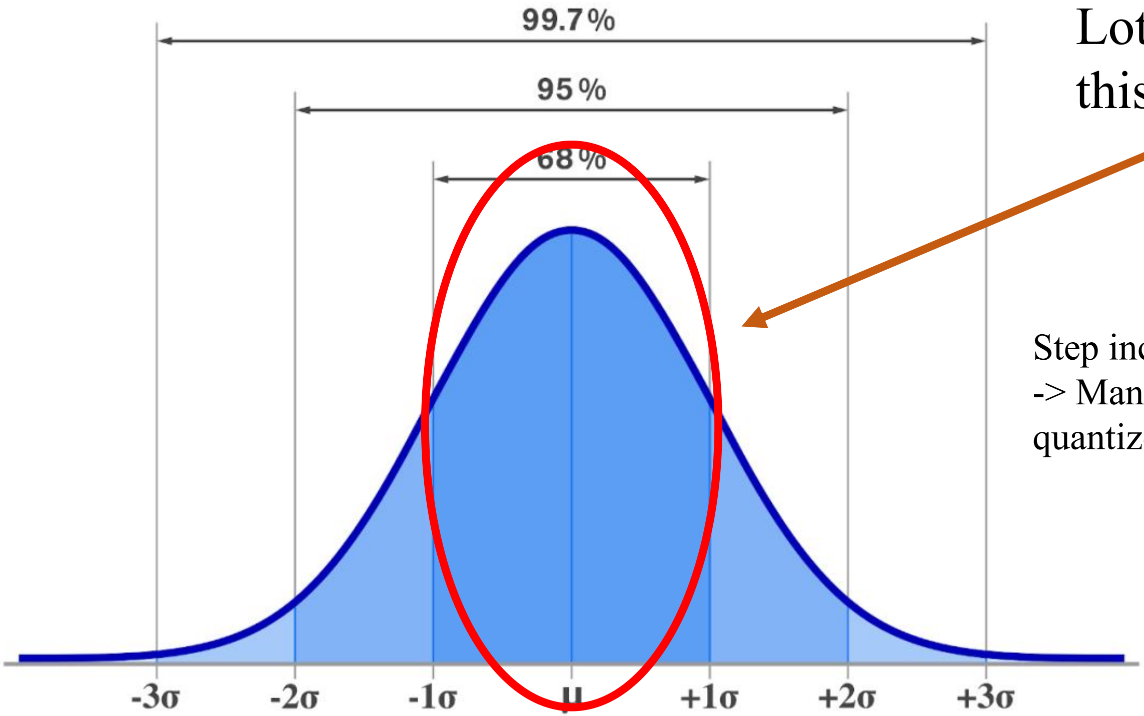
c) Xác định phạm vi giá trị tối ưu cho các tham số để giảm thiểu lỗi quantization

d) Tăng tốc độ tính toán của mô hình mà không ảnh hưởng đến kích thước mô hình

Quantization

- Motivation
- Floating Point
- Uniform Quantization
- **Non-Uniform Quantization**

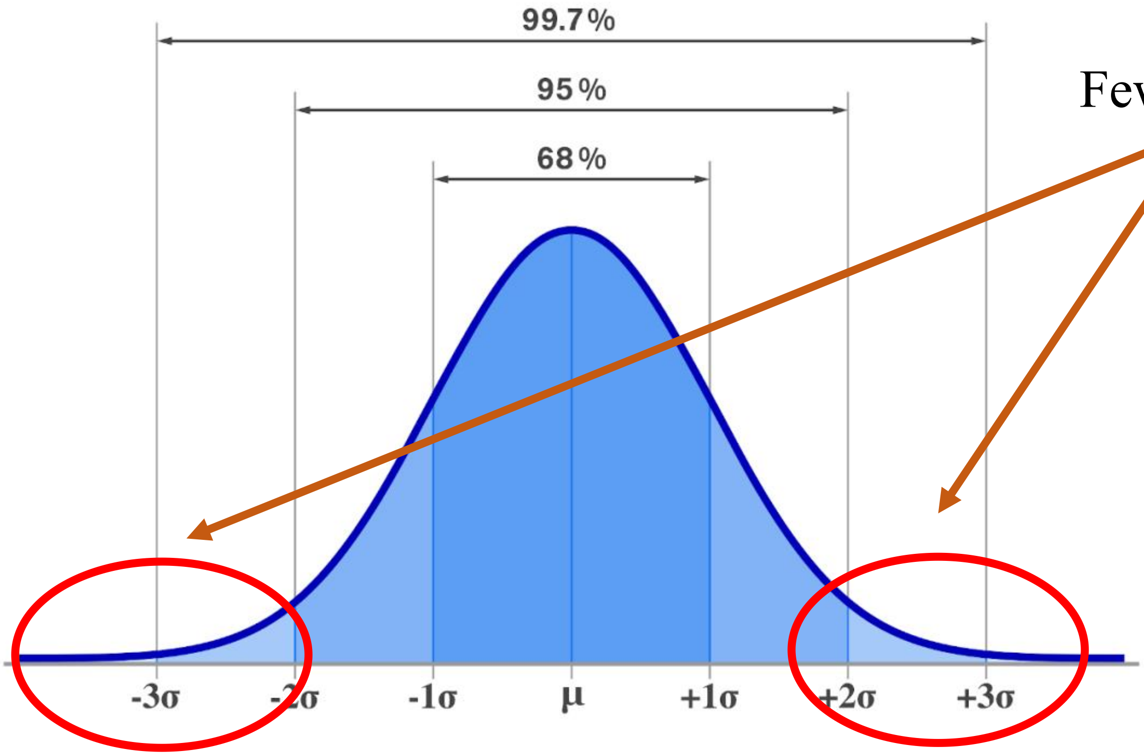
Non-uniform Quantization



Lots of data in this area

Step increase linearly
-> Many data points get quantized into 1 bin

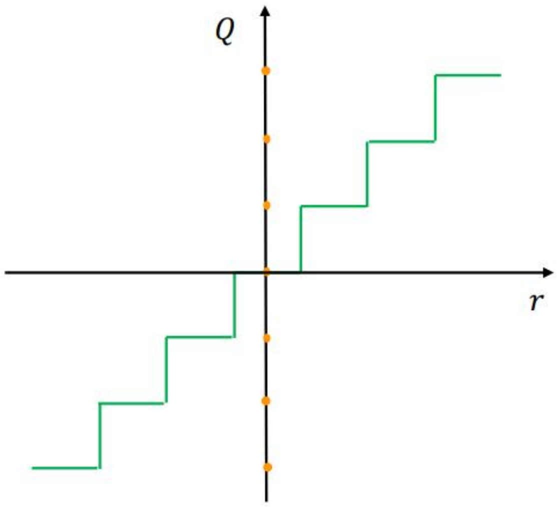
Non-uniform Quantization

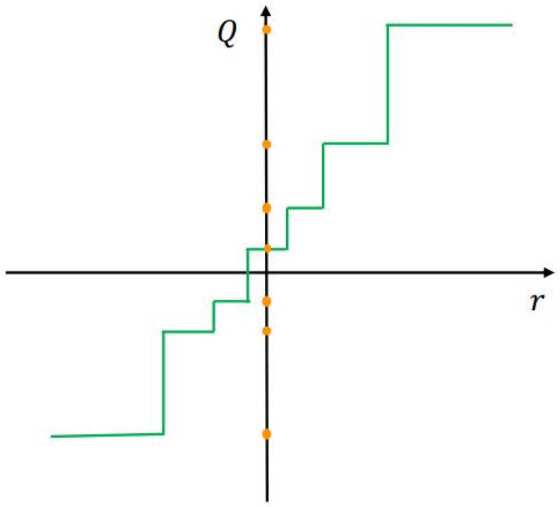


Fewer values

Still the same
number of bins

Non-uniform Quantization



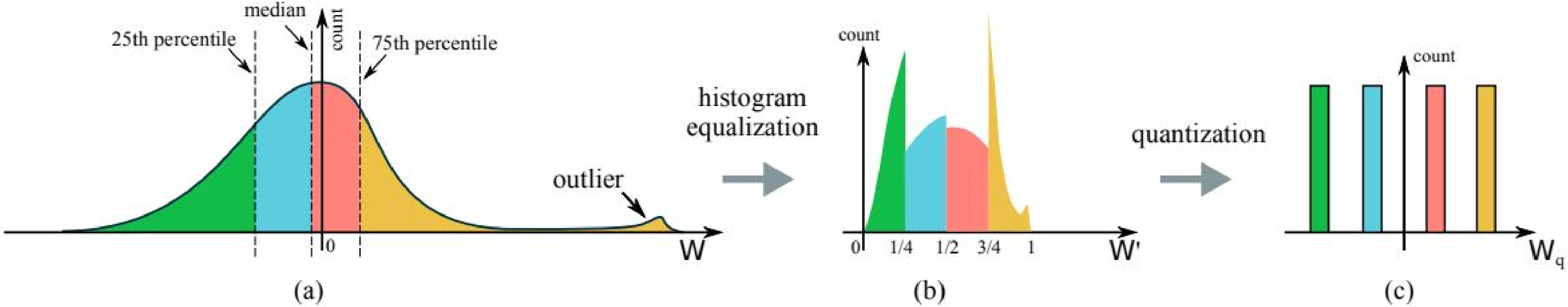


$$Q(r) = X_i,\ if\ r \in [\Delta_i,\ \Delta_{\mathrm{i+1}})$$

Non-uniform Quantization

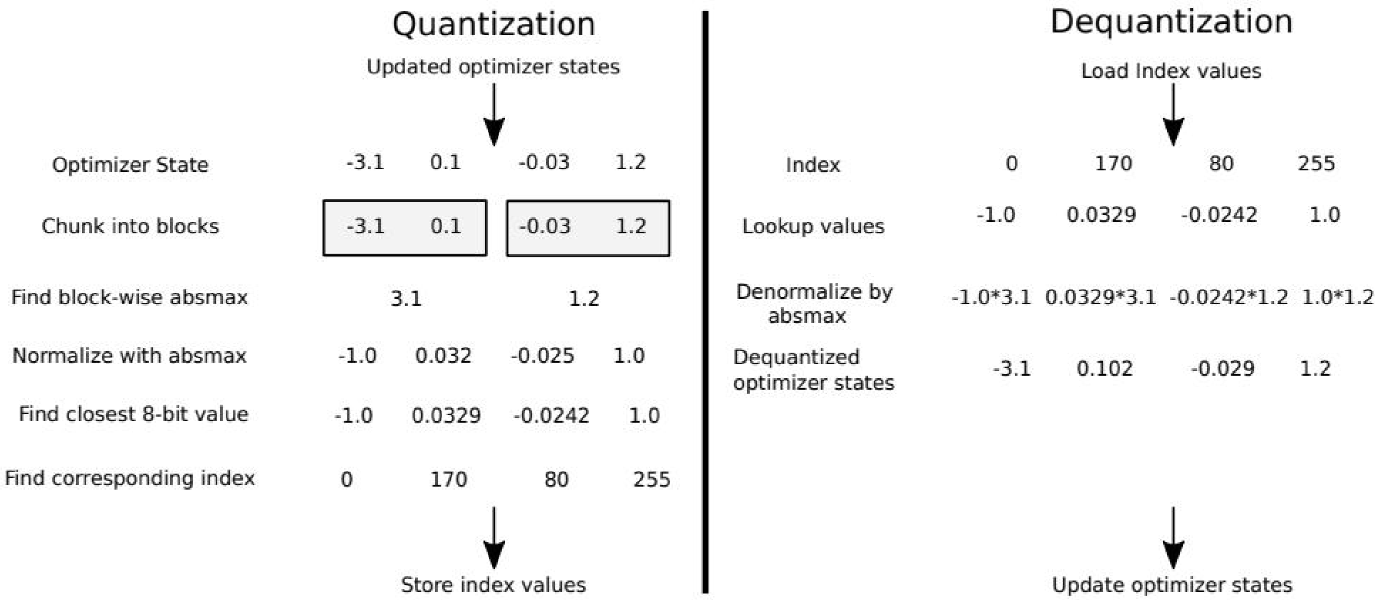
$$Q(r) = X_i,\ if\ r \in [\Delta_i,\ \Delta_{\mathrm{i+1}})$$

Balance Quantization



Non-uniform Quantization

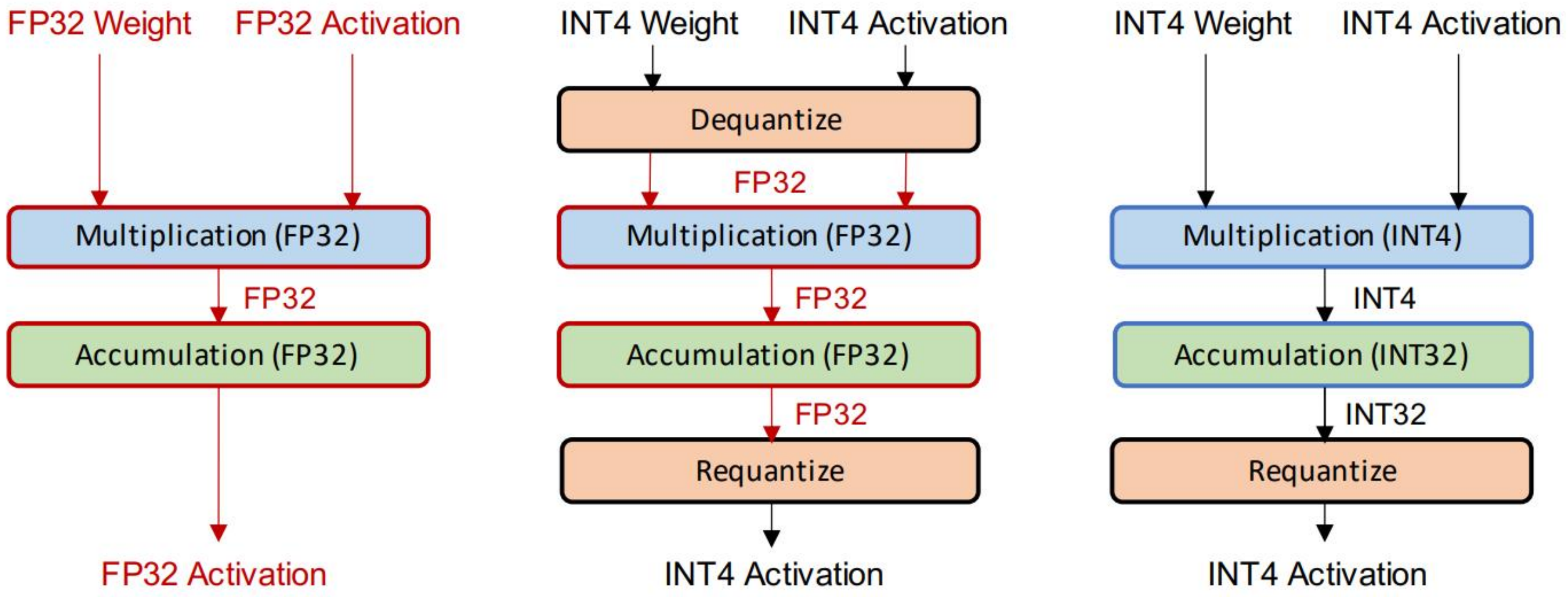
Block-wise Quantization



Non-uniform Quantization

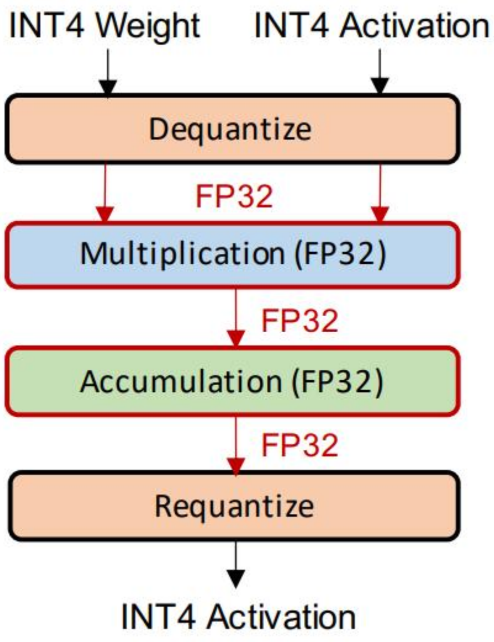
Feature	Uniform Quantization	Non-uniform Quantization
Spacing	Equally spaced levels	Unequally spaced levels (based on signal characteristics)
Simplicity	Simple and straightforward	More complex
Efficiency (uniform distribution)	Efficient	Less efficient for uniform distributions
Efficiency (non-uniform distribution)	May waste bits, poor accuracy for non-uniform distributions	More efficient and accurate for non-uniform distributions
Complexity	Low	High

Inferencing Quantized Models



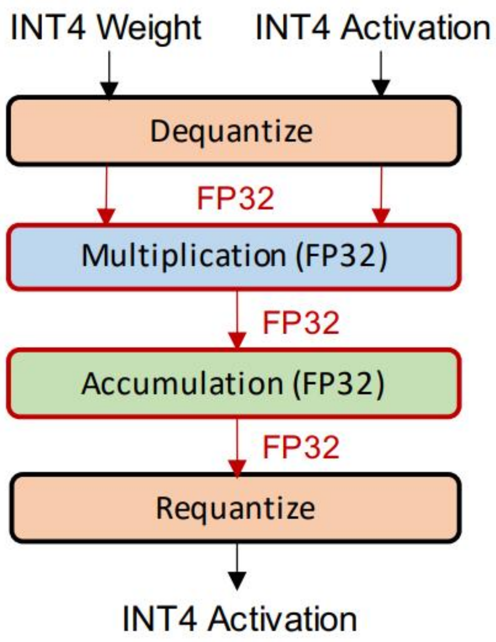
Simulated Quantization

Simulated Quantization



Simulated Quantization

Simulated Quantization



-64	32	-16
96	-48	112
-12	60	-56

→

-0.5	0.25	0.12
0.75	-0.37	0.88
-0.09	0.47	-0.44

32	50	16
16	32	44

→

0.25	0.39	0.12
0.12	0.25	0.34

Range [-1, 1], S = 1/127

Simulated Quantization

-0.5	0.25	0.12
0.75	-0.37	0.88
-0.09	0.47	-0.44

0.25	0.39	0.12
0.12	0.25	0.34

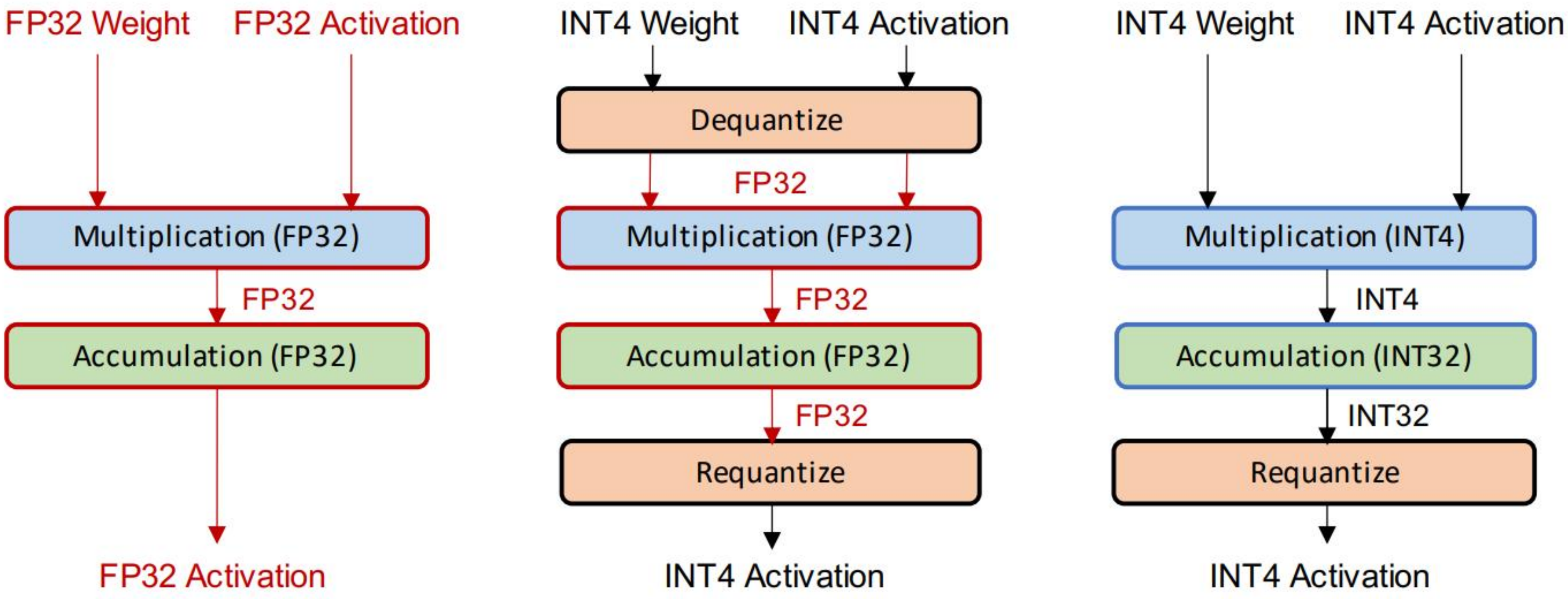
$\xrightarrow{\text{MatMul}}$

-0.01	0.04
0.15	0.30
0.10	-0.04

- Reduce Model Size
- Slow inference time

Range [-1, 1], S = 1/127

Inferencing Quantized Models



Integer Only

Int8

-64	32	-16
96	-48	112
-12	60	-56

$S_1 = 1/127$

Int8

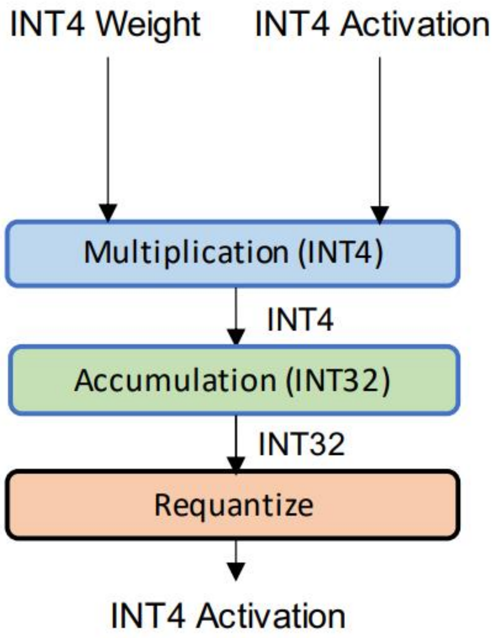
32	50	16
16	32	44

$S_2 = 1/127$

MatMul $\rightarrow$

Int32

-192	704
2464	4928
1720	-736



Integer Only

Int32

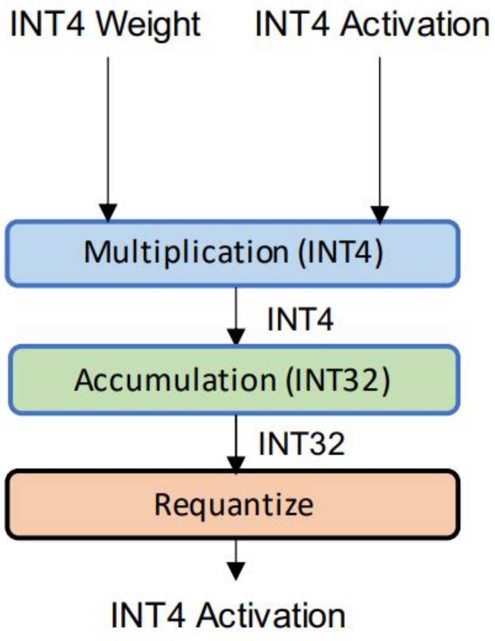
-192	704
2464	4928
1720	-736

$$Out_{Int32} * S_1 * S_2 = Out_{F16}$$

Dequantize

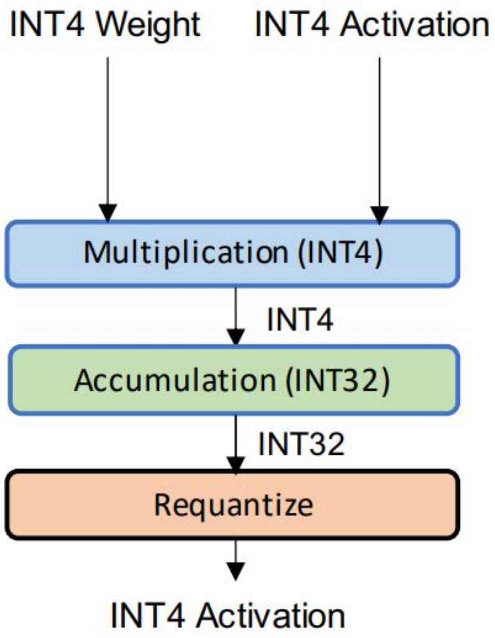
FP16

-0.01	0.04
0.15	0.30
0.10	-0.04

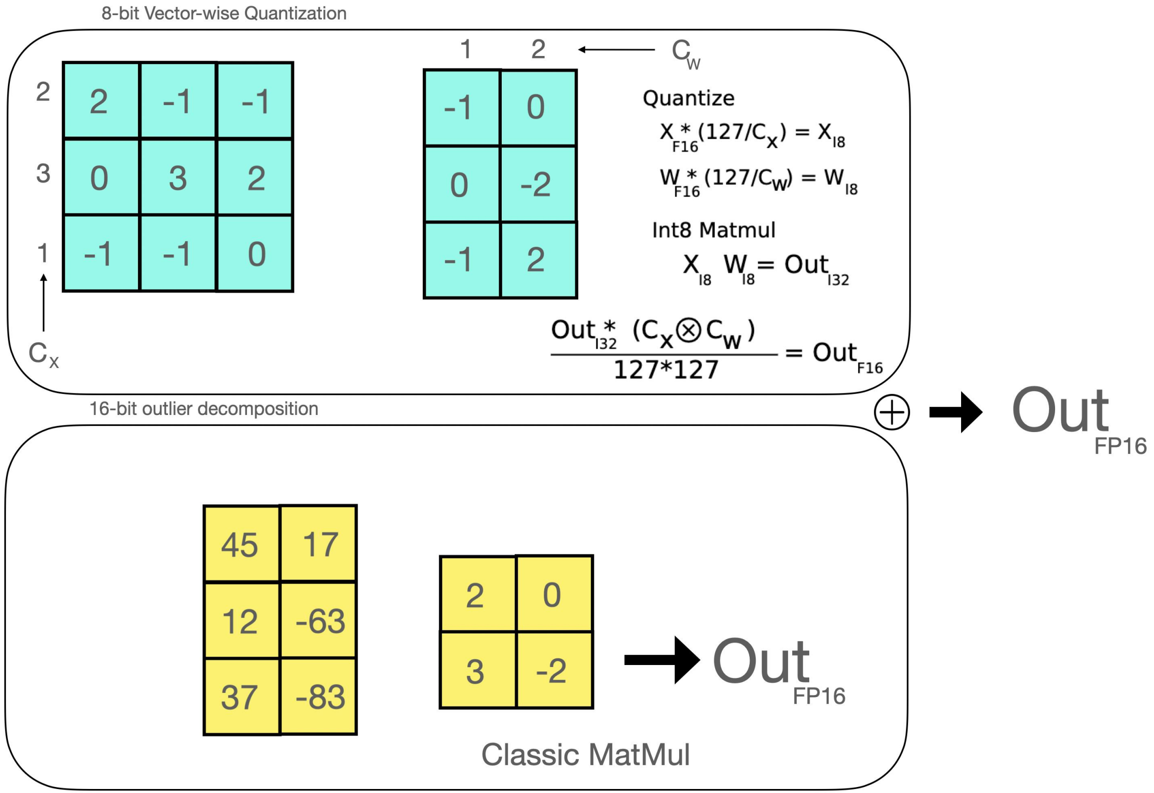


Integer Only

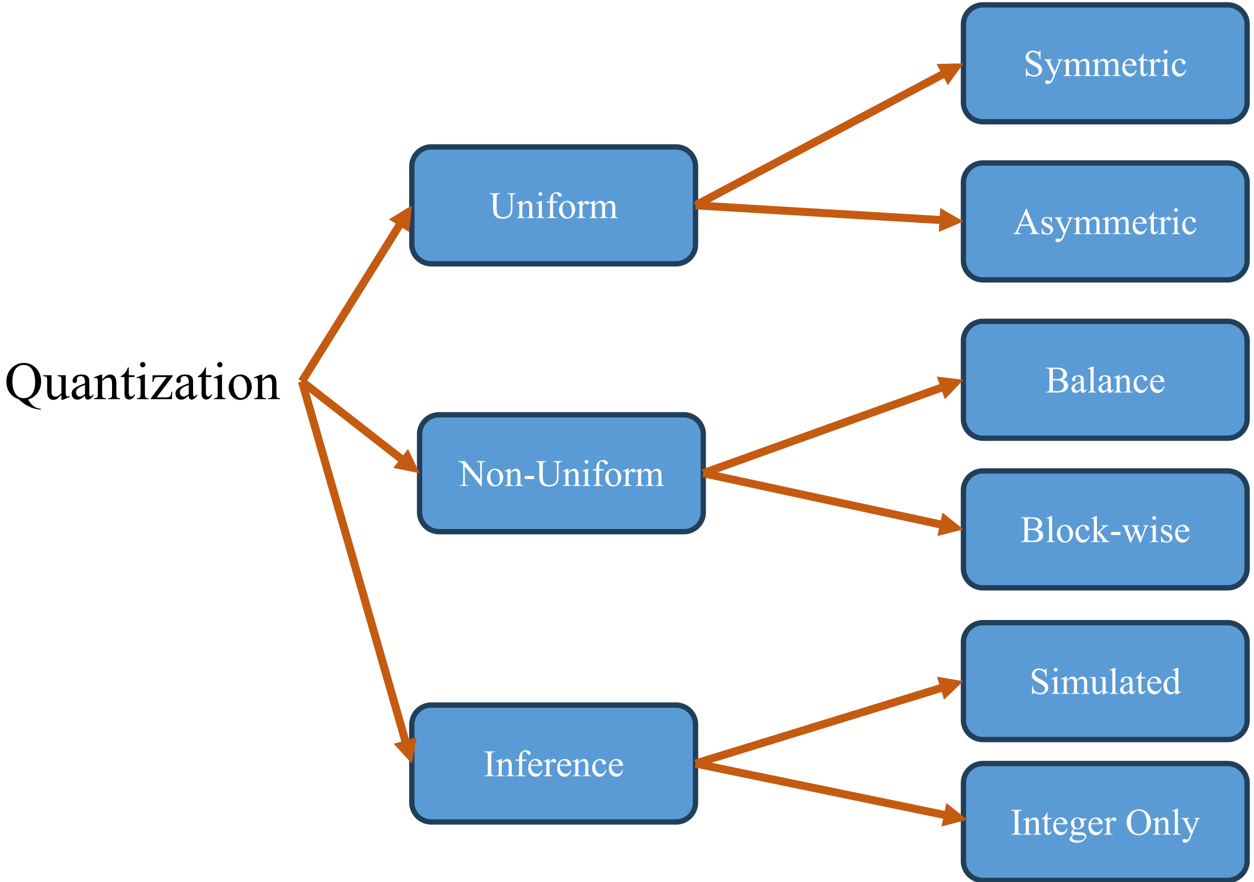
All operations -> Using low-precision integer arithmetic
- Latency
- Power consumption
- Area efficiency



Inferencing Quantized Models



Summary


Quantization
Uniform
Non-Uniform
Inference
Symmetric
Asymmetric
Balance
Block-wise
Simulated
Integer Only

Quantization