# Quantization

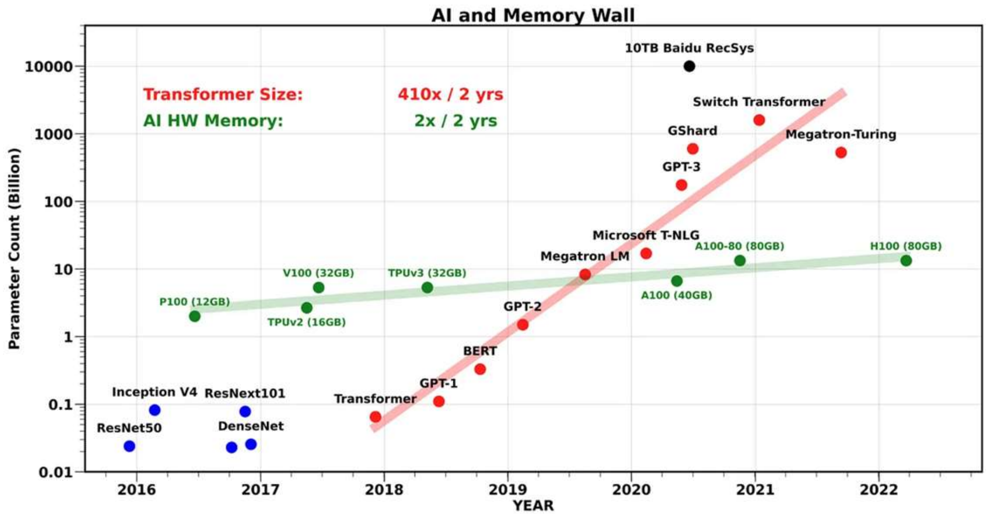
# Contents

# Background

- **Why is Quantization Important?**
  - Imbalance in the improvement of hardware resources

# Background

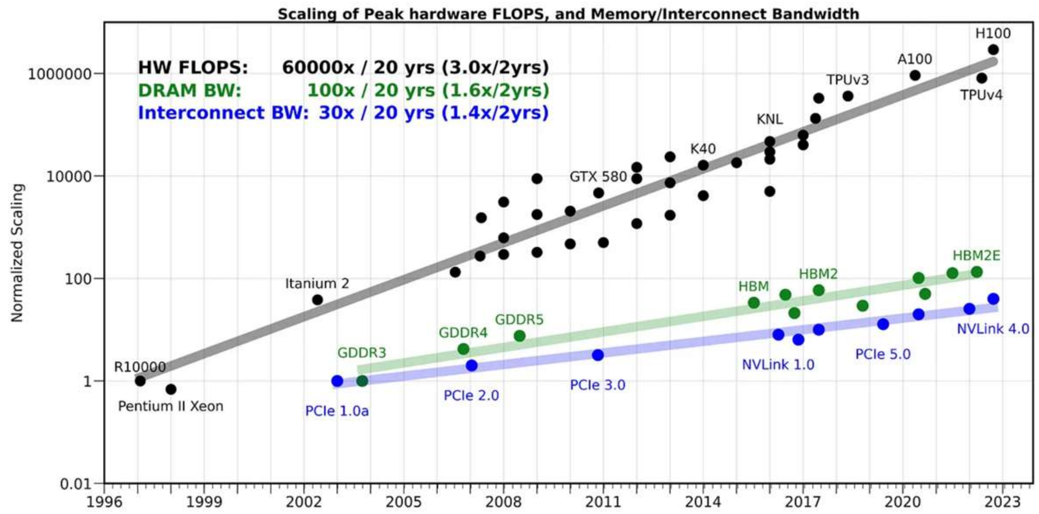
- **Why is Quantization Important?**
  - <u>Imbalance in the improvement of hardware resources</u>




**1. Models are expanding rapidly**, but **hardware resources** (particularly VRAM) **are not** keeping pace, primarily due to cost constraints.

# Background

- **Why is Quantization Important?**
  - <u>Imbalance in the improvement of hardware resources</u>




**2.** Hardware **compute power is increasing fast,** while **memory bandwidth** is growing at a **slower rate**.

# Background

- **Why is Quantization Important?**
  - Imbalance in the improvement of hardware resources

  - To maximize throughput/latency, we need to fully utilize the hardware.
    (We want our model to run faster on newer H/W)

# Background

- **Why is Quantization Important?**
  - Imbalance in the improvement of hardware resources
  - To maximize throughput/latency, we need to fully utilize the hardware.
    (We want our model to run faster on newer H/W)
  - Quantization is one of the most practical method to solve this problem.

# Background – Techniques

- **Optimizing the NN architecture**
  - Designing efficient NN model architecture
  - Co-designing NN architecture and hardware together
- **Lightweighting NN model**
  - Knowledge Distillation
  - Pruning
  - Quantization

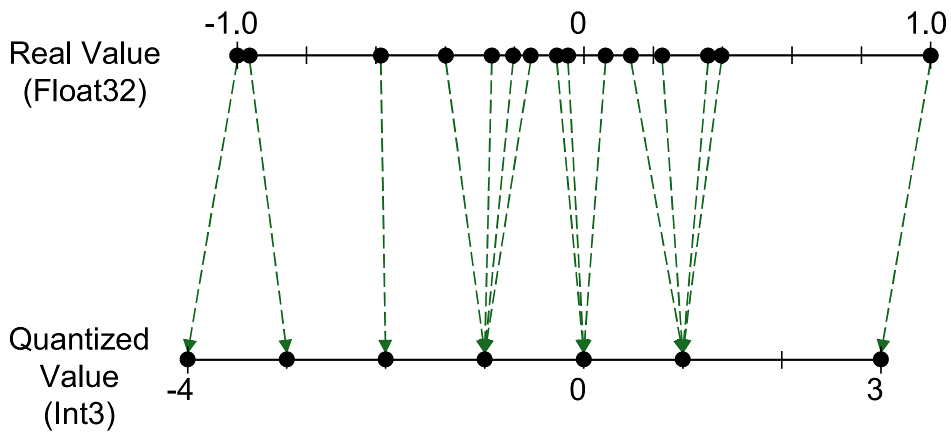
# Background – Techniques

- **Optimizing the NN architecture**
  - Designing efficient NN model architecture
  - Co-designing NN architecture and hardware together
- **Lightweighting NN model**
  - Knowledge Distillation
  - Pruning
  - **Quantization**

# What is Quantization?

Mapping **continuous (Float)** values into smaller set of **discrete (Integer)** values

- Example (3-Bit Uniform Quantization)



# How do we optimize with Quantization?

Quantization inevitably have a **negative impact on accuracy**

How to overcome this problem?

# How do we optimize with Quantization?

Quantization inevitably have a **negative impact on accuracy**

How to overcome this problem?

1. Quantize everything and find way to restore accuracy

# How do we optimize with Quantization?

Quantization inevitably have a **negative impact on accuracy**

How to overcome this problem?

1. Quantize everything and find way to restore accuracy
2. Partially apply quantization to avoid affecting accuracy

# How do we optimize with Quantization?

Quantization inevitably have a **negative impact on accuracy**

How to overcome this problem?

1. Quantize everything and find way to restore accuracy

2. Partially apply quantization to avoid affecting accuracy
   - Quantizing a few operation **would not change** model output **much**
   - **Find expensive operation** with a long latency **and quantize** them

# How do we optimize with Quantization?

- Examples of **expensive** operations in DL
  - **Convolution operation**
  - **Linear operation**

- Examples of inexpensive operations in DL
  - Normalization Layer
  - Element-wise Operation (Residual add/concatenation)

# Preliminaries – Convolution operation

<Input>

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

<Kernel>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

<Output>

| | | |
|---|---|---|
| | | |
| | | |
| | | |

# Preliminaries – Convolution operation

<Input>

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

<Kernel>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

<Output>

| | | |
|---|---|---|
| 3 | | |
| | | |
| | | |

# Preliminaries – Convolution operation

<Input>

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

<Kernel>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

<Output>

| | | |
|---|---|---|
| 3 | 0 | |
| | | |
| | | |

# Preliminaries – Convolution operation

<Input>

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

<Kernel>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

<Output>

| | | |
|---|---|---|
| 3 | 0 | 0 |
| | | |
| | | |

# Preliminaries – Convolution operation

<Input>

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

<Kernel>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

<Output>

| | | |
|---|---|---|
| 3 | 0 | 0 |
| 0 | | |
| | | |

# Preliminaries – Convolution operation

<Input>

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

<Kernel>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

<Output>

| | | |
|---|---|---|
| 3 | 0 | 0 |
| 0 | 3 | |
| | | |

# Preliminaries – Convolution operation

<Input>

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

<Kernel>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

<Output>

| | | |
|---|---|---|
| 3 | 0 | 0 |
| 0 | 3 | 0 |
| 0 | 0 | 3 |

# Preliminaries – Convolution operation

<Input>

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

<Kernel>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

<Output>

| | | |
|---|---|---|
| 3 | 0 | 0 |
| 0 | 3 | 0 |
| 0 | 0 | 3 |

<Kernel>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

<Output>

# Preliminaries – Convolution operation

<Input>

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

<Kernel>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

<Output>

| | | |
|---|---|---|
| 3 | 0 | 0 |
| 0 | 3 | 0 |
| 0 | 0 | 3 |

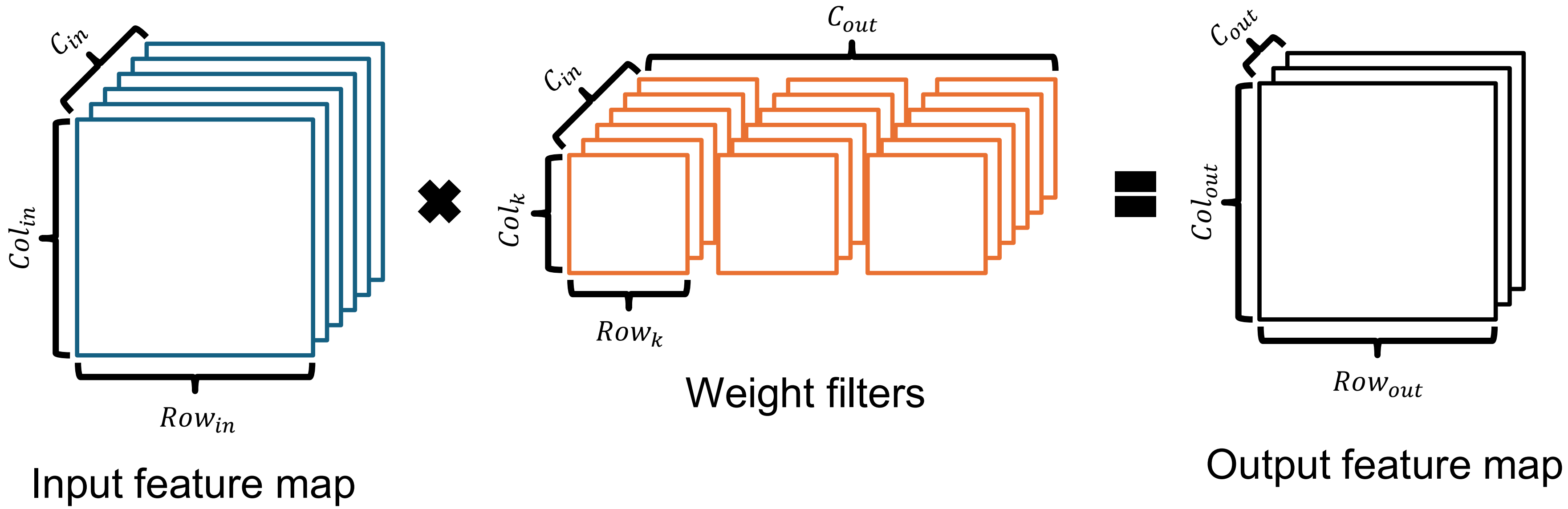
<Kernel>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

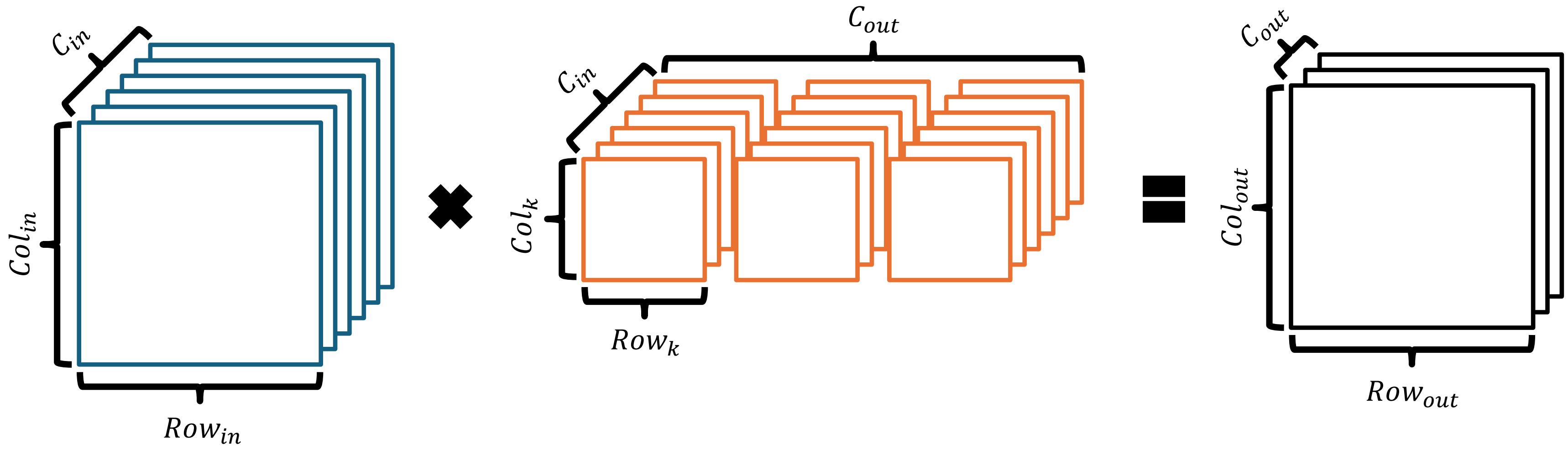
<Output>

| |
|---|
| 9 |

# Preliminaries – Convolution operation

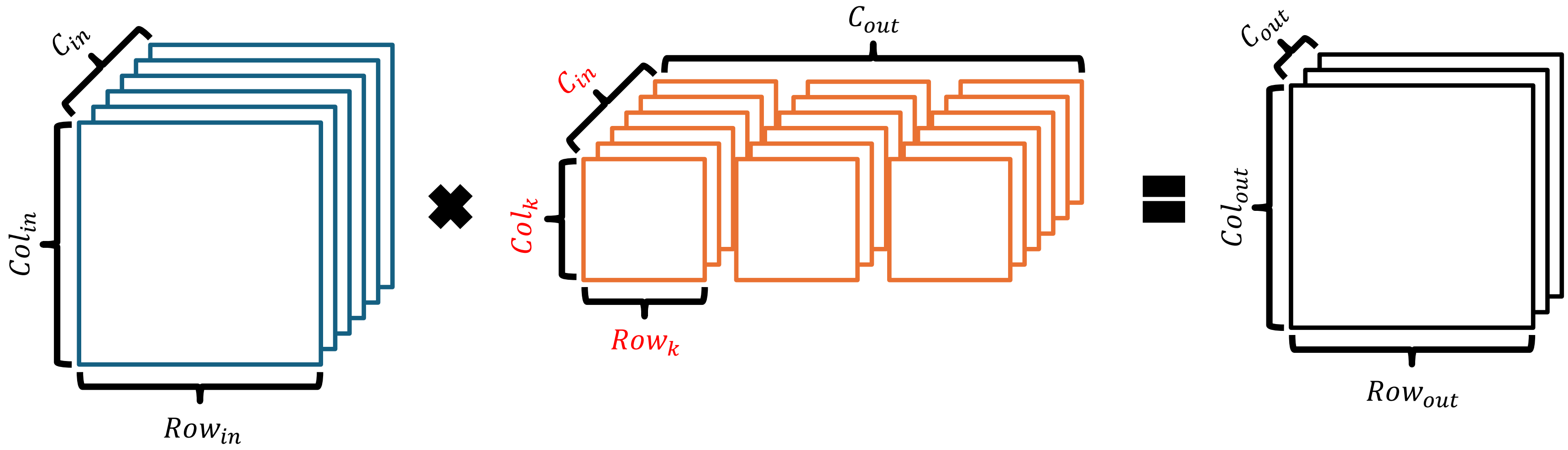


# Preliminaries – Convolution operation



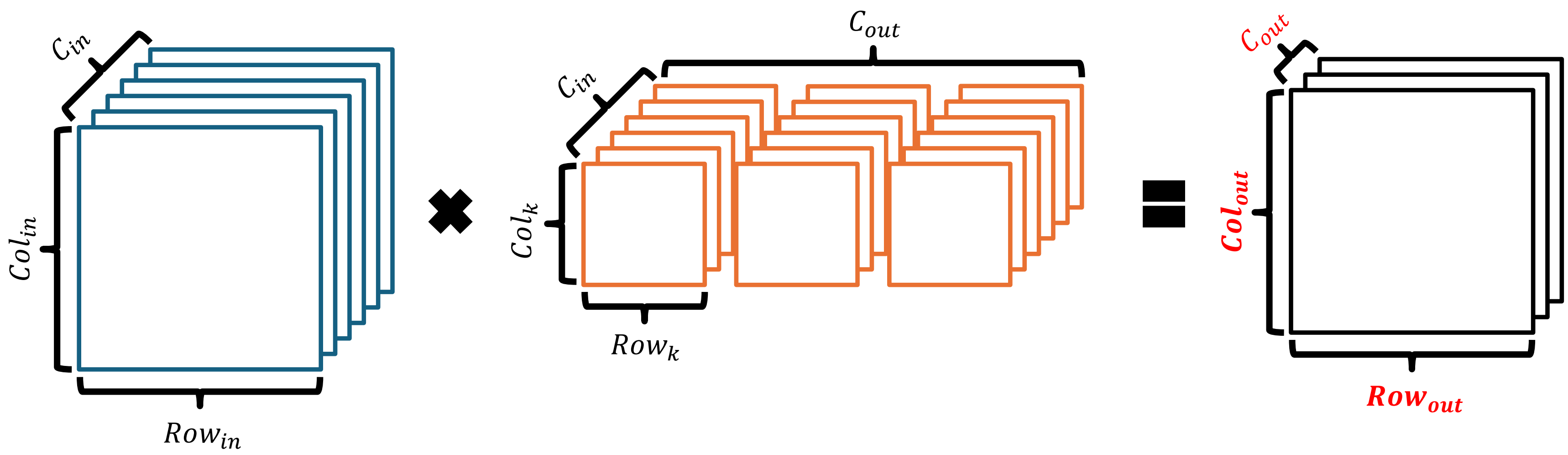

$$FLOPs = Col_k \times Row_k \times C_{in} \times Col_{out} \times Row_{out} \times C_{out}$$

# Preliminaries – Convolution operation



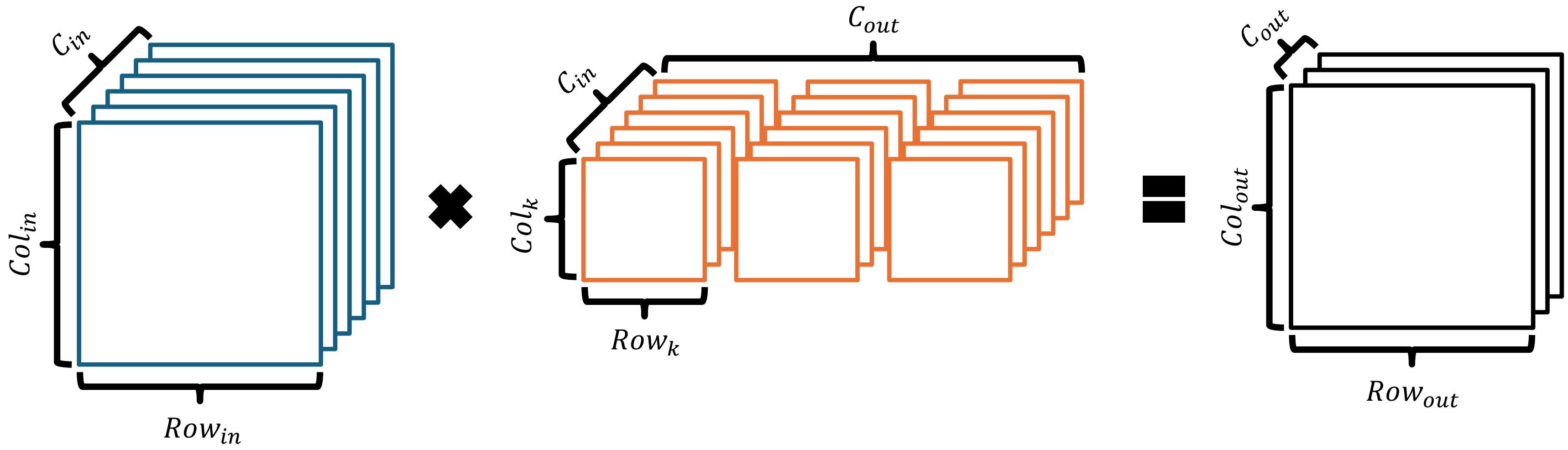

$$FLOPs = Col_k \times Row_k \times C_{in} \times Col_{out} \times Row_{out} \times C_{out}$$

# Preliminaries – Convolution operation



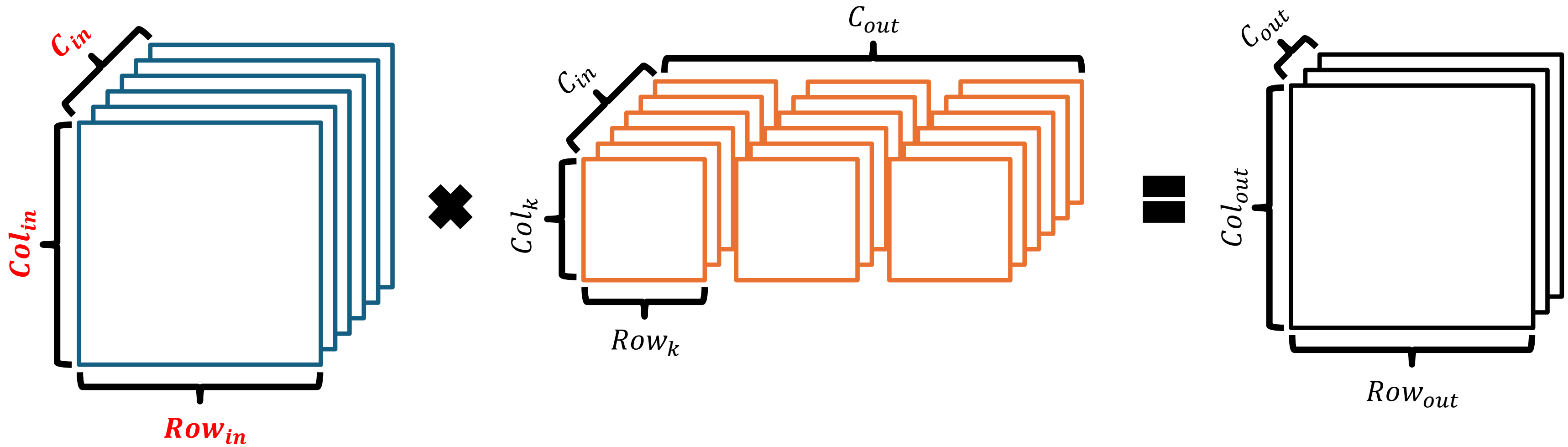

$$FLOPs = Col_k \times Row_k \times C_{in} \times Col_{out} \times Row_{out} \times C_{out}$$

# Preliminaries – Convolution operation



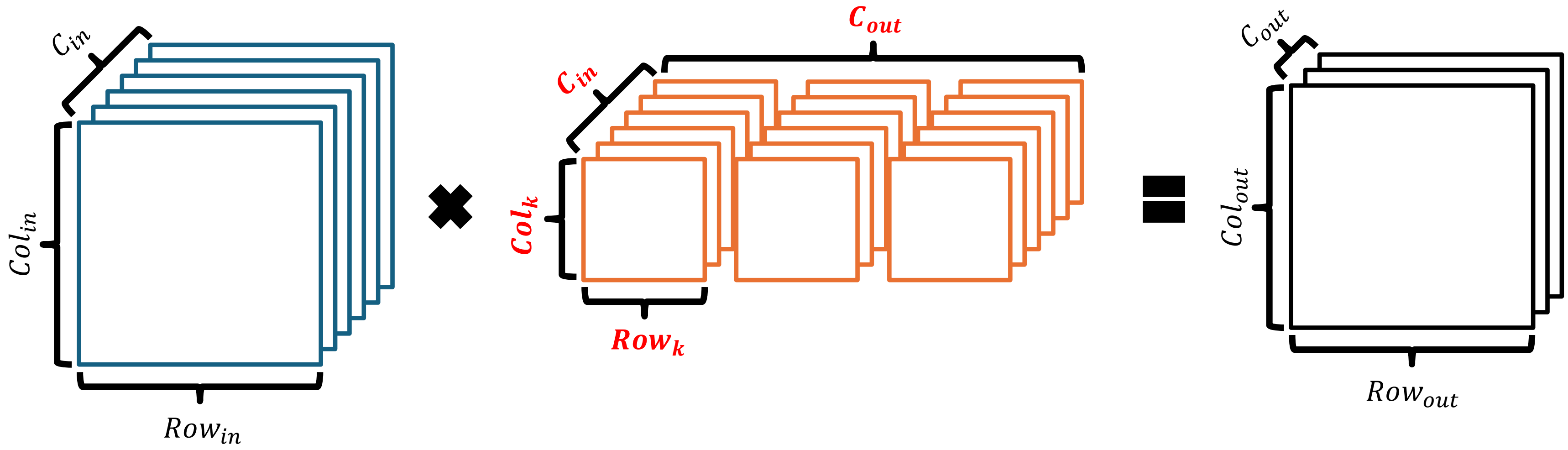

$$MEM = [Col_{in} \times Row_{in} \times C_{in}] + [Col_k \times Row_k \times C_{in} \times C_{out}]$$

# Preliminaries – Convolution operation



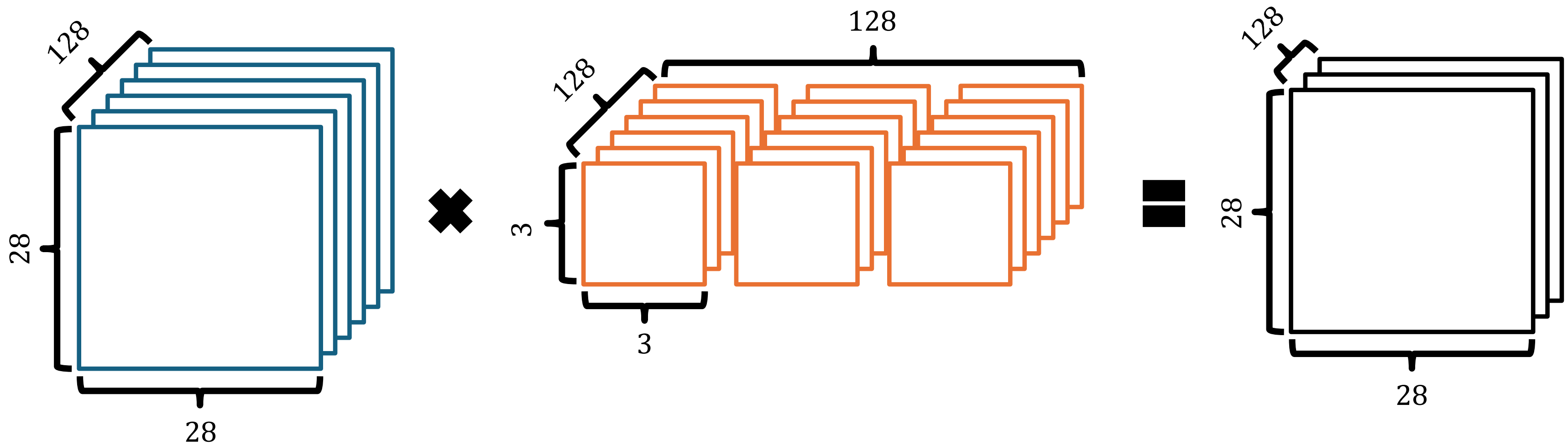

$$MEM = [Col_{in} \times Row_{in} \times C_{in}] + [Col_k \times Row_k \times C_{in} \times C_{out}]$$

# Preliminaries – Convolution operation




$$MEM = [Col_{in} \times Row_{in} \times C_{in}] + [Col_k \times Row_k \times C_{in} \times C_{out}]$$

# Preliminaries – Convolution operation

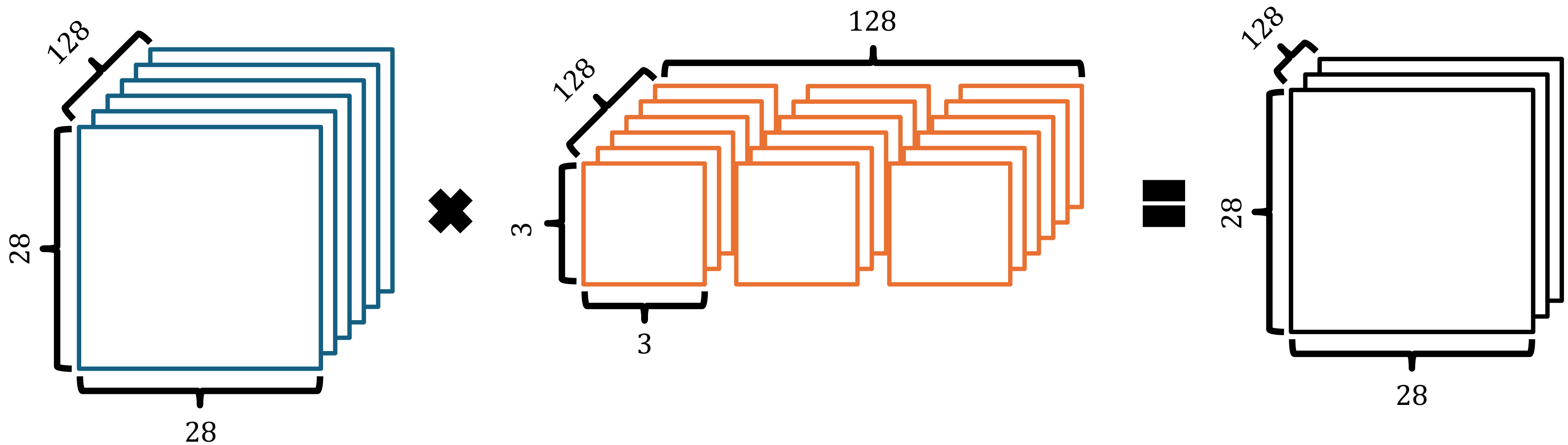



Example with ResNet50 (middle layer)

$$FLOPs = 3 \times 3 \times 28 \times 28 \times 128 \times 128 = 115M$$

$$MEM = [28 \times 28 \times 128] + [3 \times 3 \times 128 \times 128] = 0.10M + 0.15M$$

# Preliminaries – Convolution operation

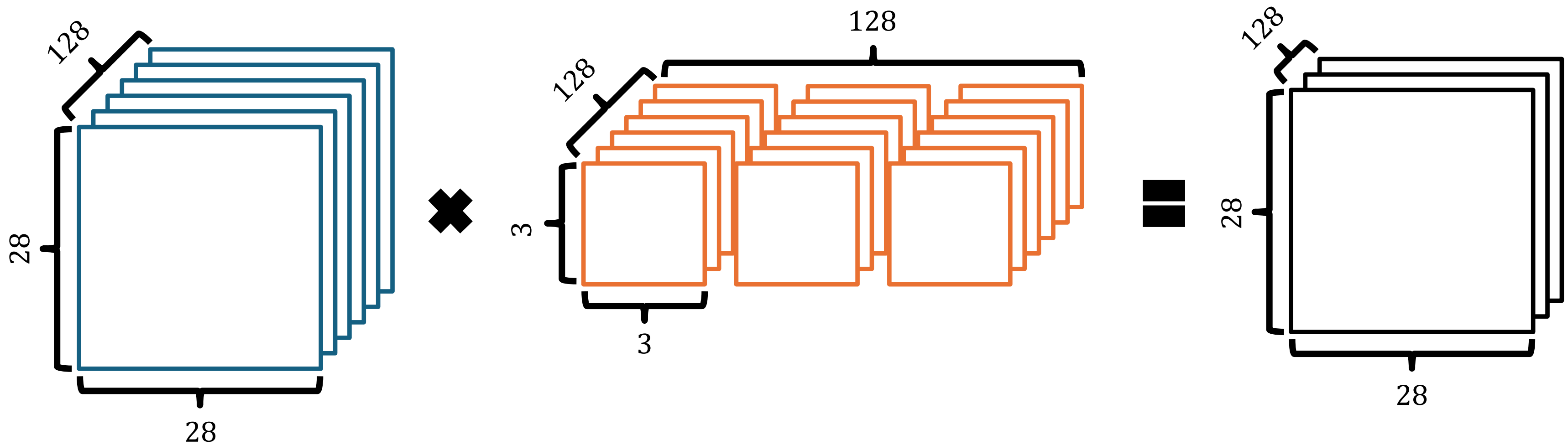



Example with ResNet50 (middle layer)

$$FLOPs = 115M$$

$$MEM = 0.10M + 0.15M$$

# Preliminaries – Convolution operation



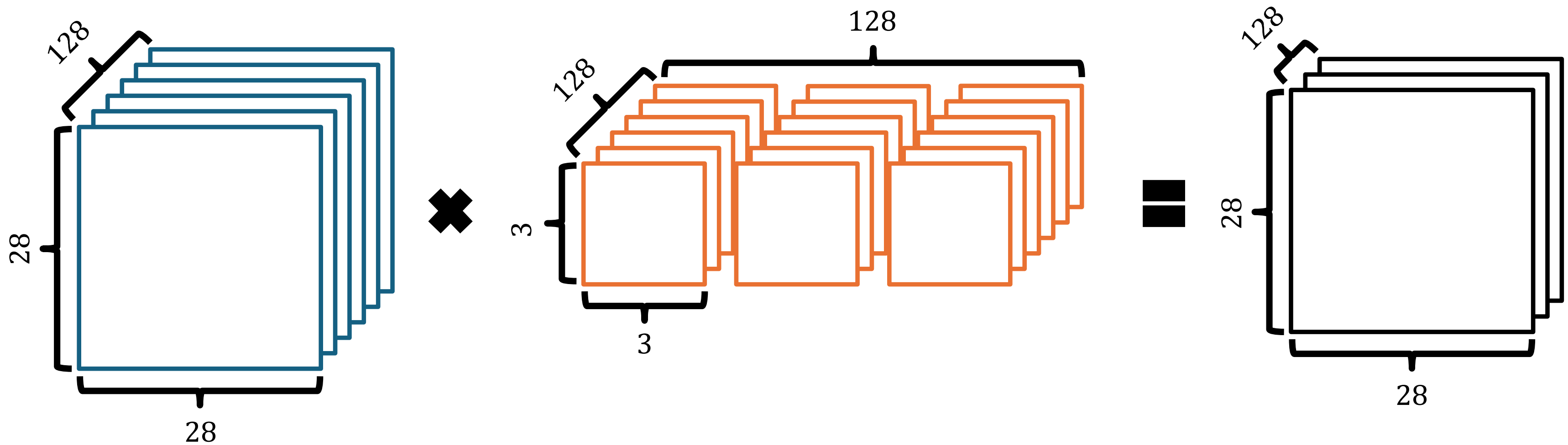

Example with ResNet50 (middle layer)

$FLOPs = 115M$

Suppose we are using FP16..

$MEM = 0.10M + 0.15M = 0.5MB$

# Preliminaries – Convolution operation



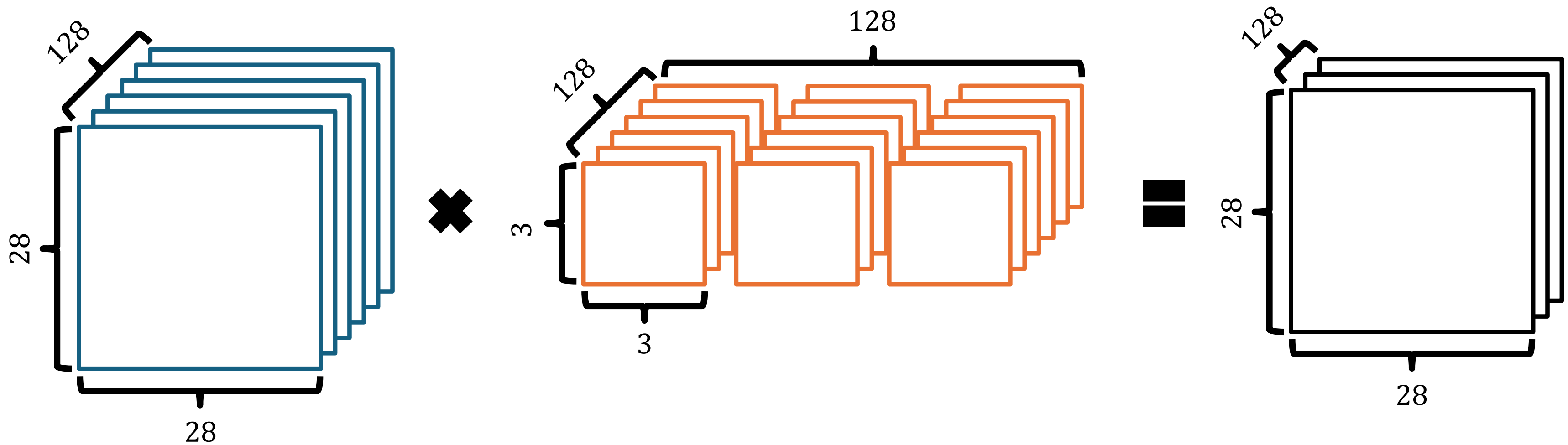

Example with ResNet50 (middle layer)

$$FLOPs = 115M$$

$$MEM = 0.10M + 0.15M = 0.5MB$$

So roughly assuming, our hardware need capability to process 115M computation while reading 0.5MB

# Preliminaries – Convolution operation




Example with ResNet50 (middle layer)

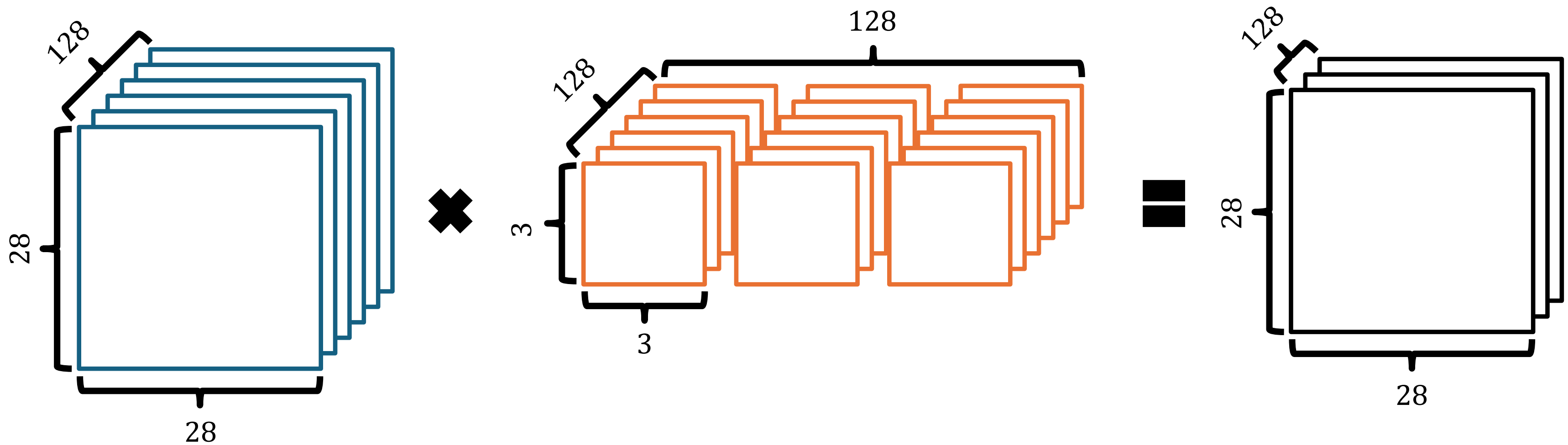
$$FLOPs = 115M$$

$$MEM = 0.10M + 0.15M = 0.5MB$$

So roughly assuming, our hardware need capability to process 115M computation while reading 0.5MB

This is called **Arithmetic intensity** : FLOPs / Bytes

# Preliminaries – Convolution operation



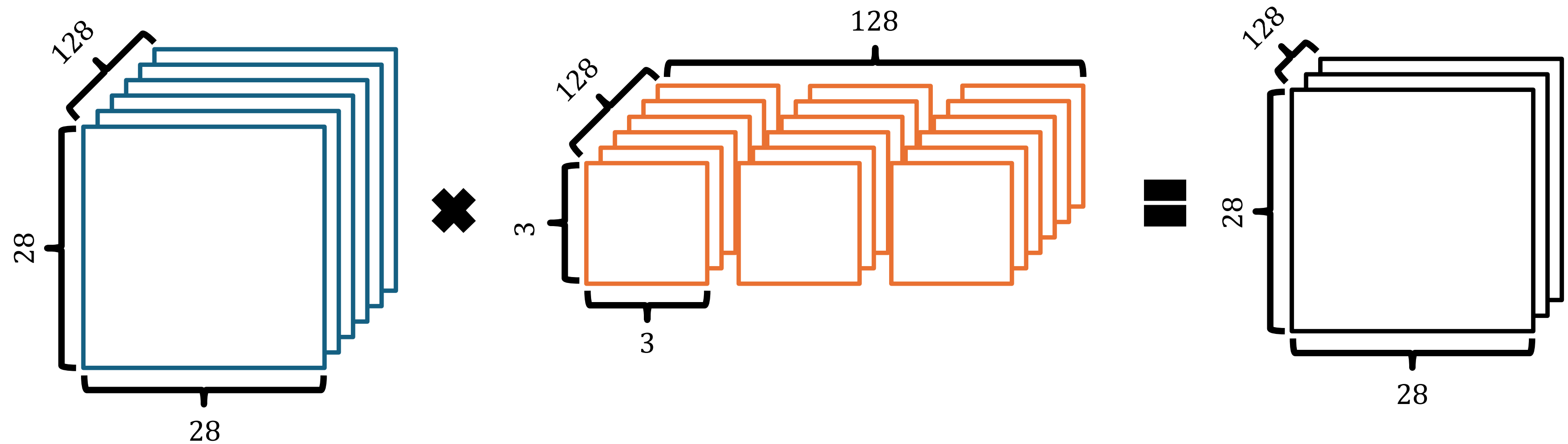

Example with ResNet50 (middle layer)

$$FLOPs = 115M$$

$$MEM = 0.10M + 0.15M = 0.5MB$$

Arithmetic intensity of A100 with FP16 :
**208 FLOPs / Bytes**

# Preliminaries – Convolution operation




Example with ResNet50 (middle layer)

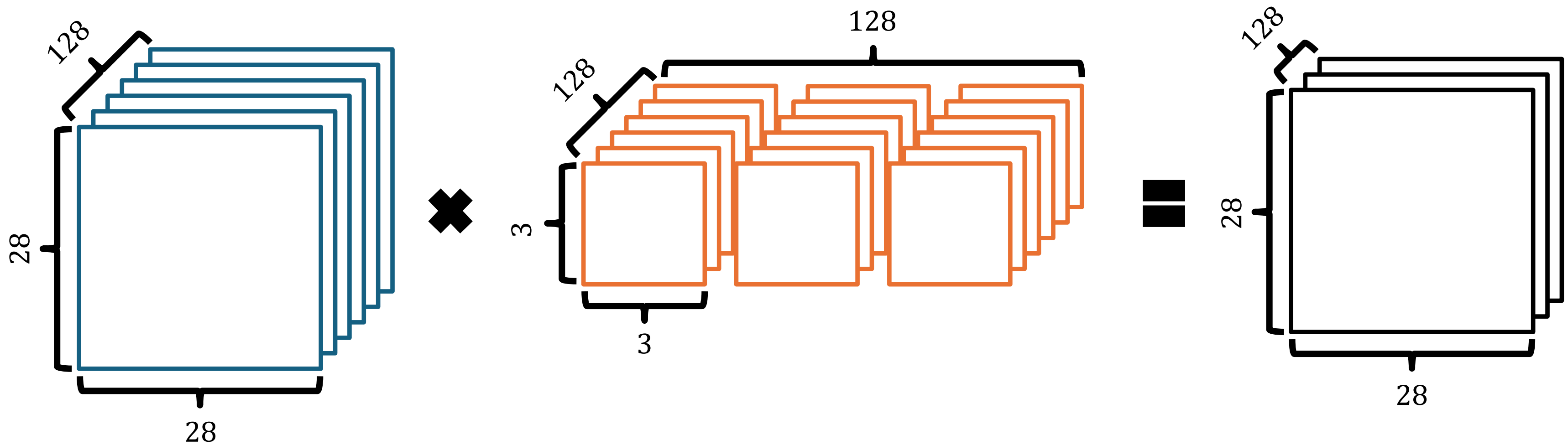
$$FLOPs = 115M$$

$$MEM = 0.10M + 0.15M = 0.5MB$$

Arithmetic intensity of A100 with FP16 :
**208 FLOPs / Bytes**

Can process at most 208 Operation every 1 Byte Read.

# Preliminaries – Convolution operation




Example with ResNet50 (middle layer)

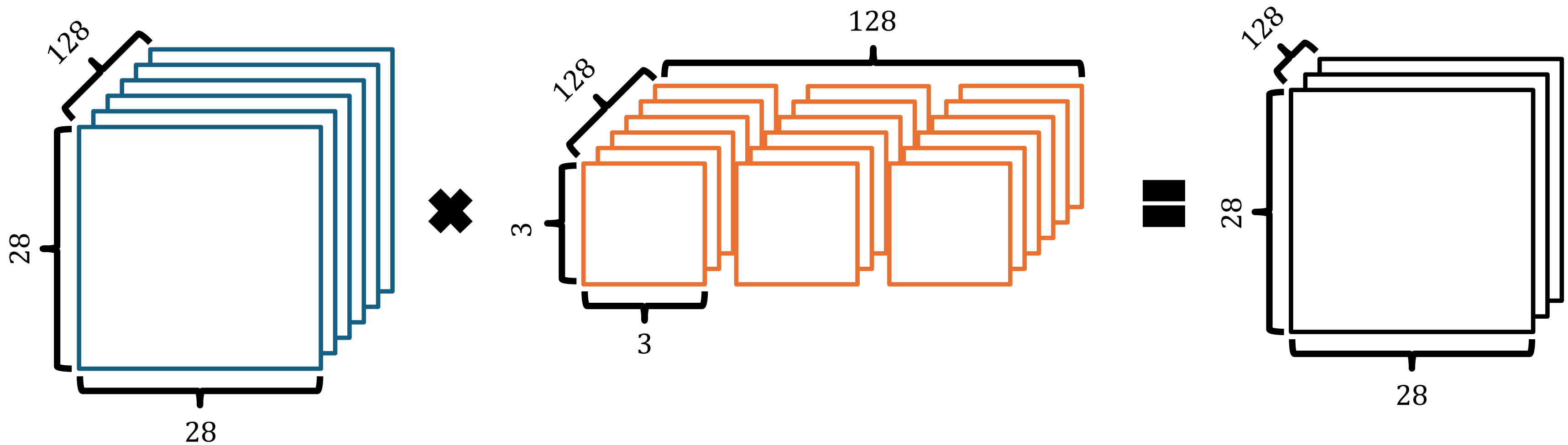
$$FLOPs = 115M$$

$$MEM = 0.10M + 0.15M = 0.5MB$$

Arithmetic intensity of A100 with FP16 :
**208 FLOPs / Bytes**

Arithmetic intensity of example:
**230 FLOPs / Bytes**

# Preliminaries – Convolution operation




Example with ResNet50 (middle layer)

$$FLOPs = 115M$$

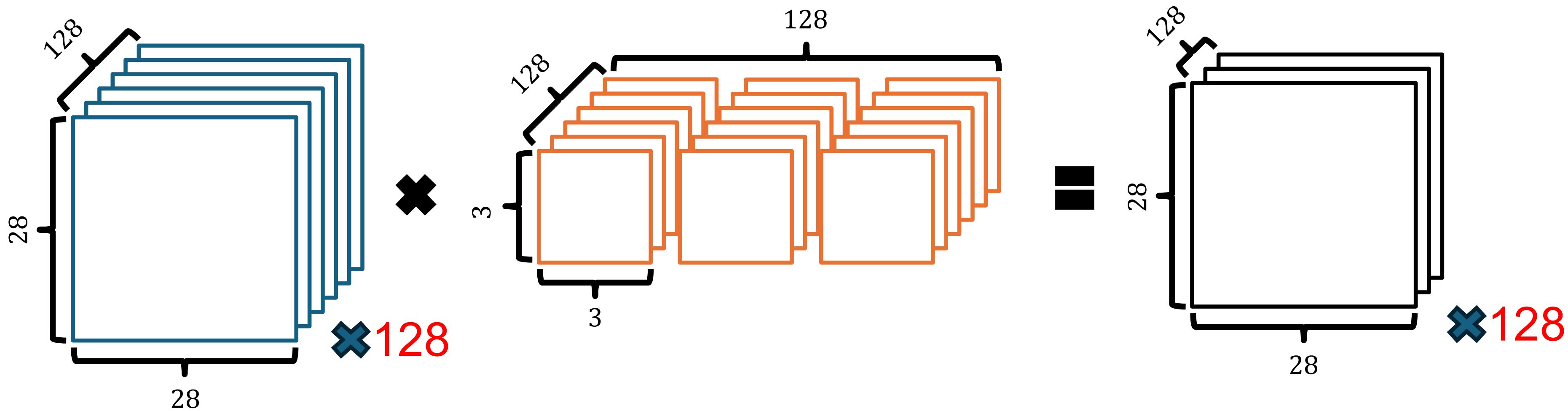
$$MEM = 0.10M + 0.15M = 0.5MB$$

Arithmetic intensity of A100 with FP16 :
**208 FLOPs / Bytes**

Arithmetic intensity of example:
**230 FLOPs / Bytes**

**Going to suffer Compute bound!**

# Preliminaries – Convolution operation




Example with ResNet50 (middle layer)

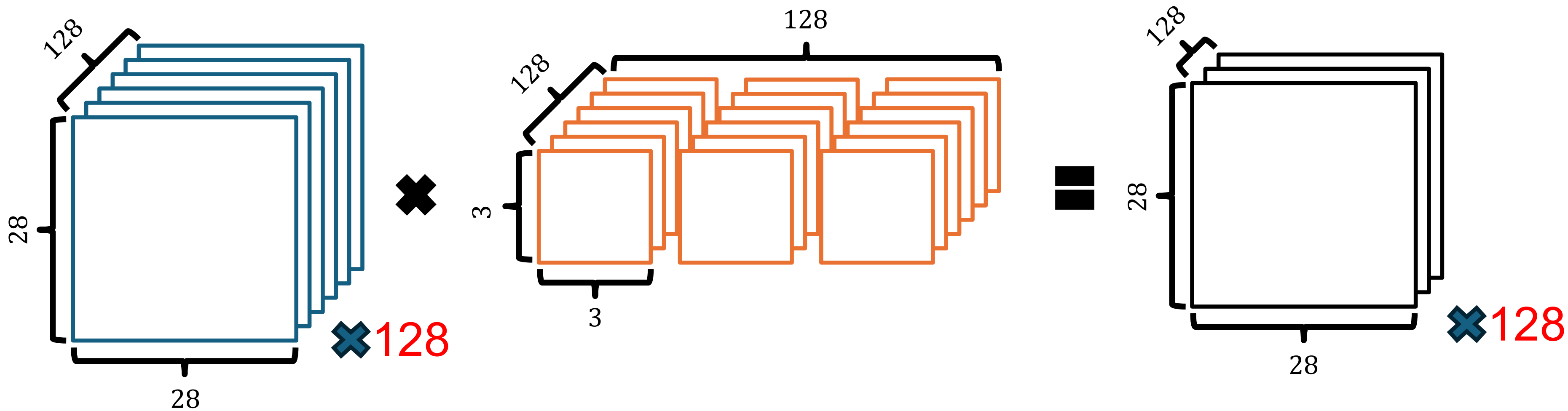
$$FLOPs = 115M \times 128 = 14.8B$$

$$MEM = 12.8M + 0.15M = 26.0MB$$

Arithmetic intensity of A100 with FP16 :
**208 FLOPs / Bytes**

Arithmetic intensity with 128 batch:
**569 FLOPs / Bytes**

# Preliminaries – Convolution operation




Example with ResNet50 (middle layer)

$$FLOPs = 115M \times 128 = 14.8B$$

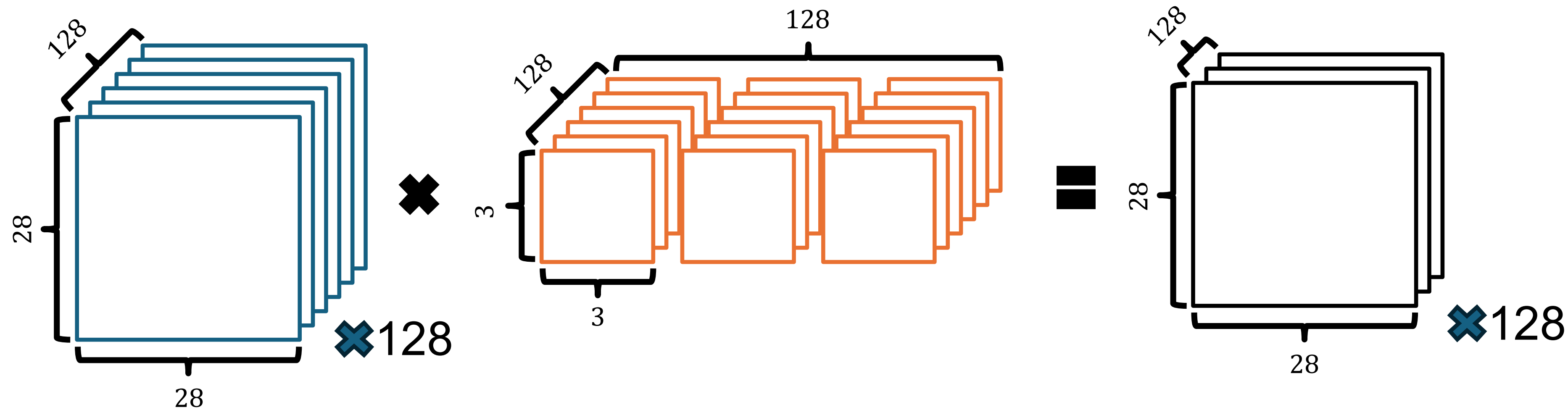
$$MEM = 12.8M + 0.15M = 26.0MB$$

Arithmetic intensity of A100 with FP16 :
**208 FLOPs / Bytes**

Arithmetic intensity with 128 batch:
**569 FLOPs / Bytes**

**Very Compute bound!**

# Preliminaries – Convolution operation




Example with ResNet50 (middle layer)

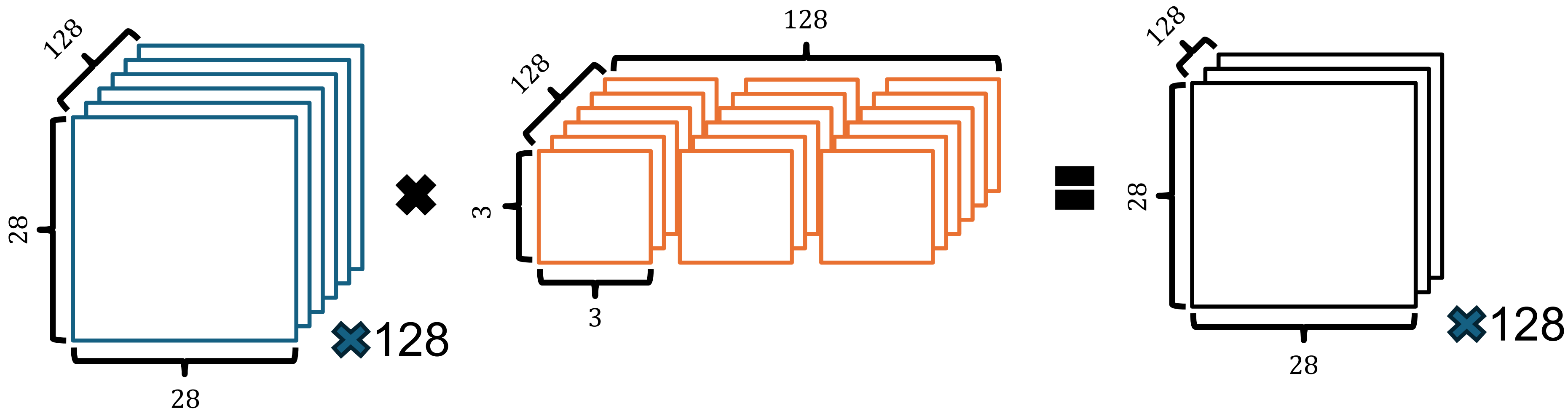
$$FLOPs = 115M \times 128 = 14.8B$$

$$MEM = 12.8M + 0.15M = 26.0MB$$

Arithmetic intensity of A100 with INT8 :
**832 OPs / Bytes**

Arithmetic intensity with 128 batch:
**569 FLOPs / Bytes**

# Preliminaries – Convolution operation




Example with ResNet50 (middle layer)

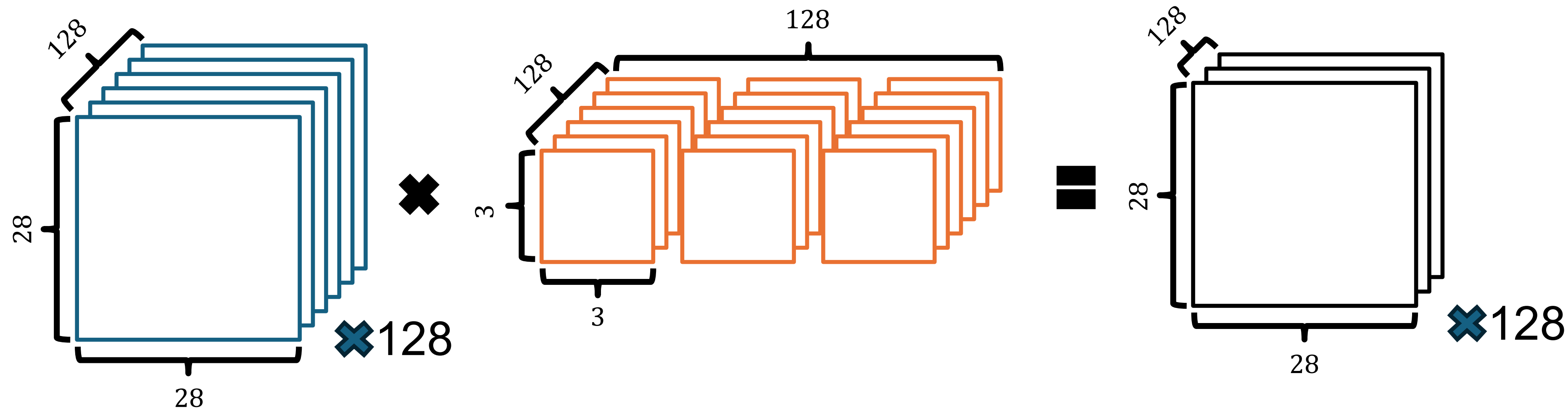
$$FLOPs = 115M \times 128 = 14.8B$$

$$MEM = 12.8M + 0.15M = 13.0MB$$

Arithmetic intensity of A100 with INT8 :
**832 OPs / Bytes**

Arithmetic intensity with 128 batch:
**1139 OPs / Bytes**

# Preliminaries – Convolution operation




Example with ResNet50 (middle layer)

$$FLOPs = 115M \times 128 = 14.8B$$

$$MEM = 12.8M + 0.15M = 13.0MB$$

Arithmetic intensity of A100 with INT8 :
**832 OPs / Bytes**

Arithmetic intensity with 128 batch:
**1139 OPs / Bytes**

**We have chance to accelerate the layer with INT8 quantized computation!**

# Preliminaries – Linear operation

<Input>

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |

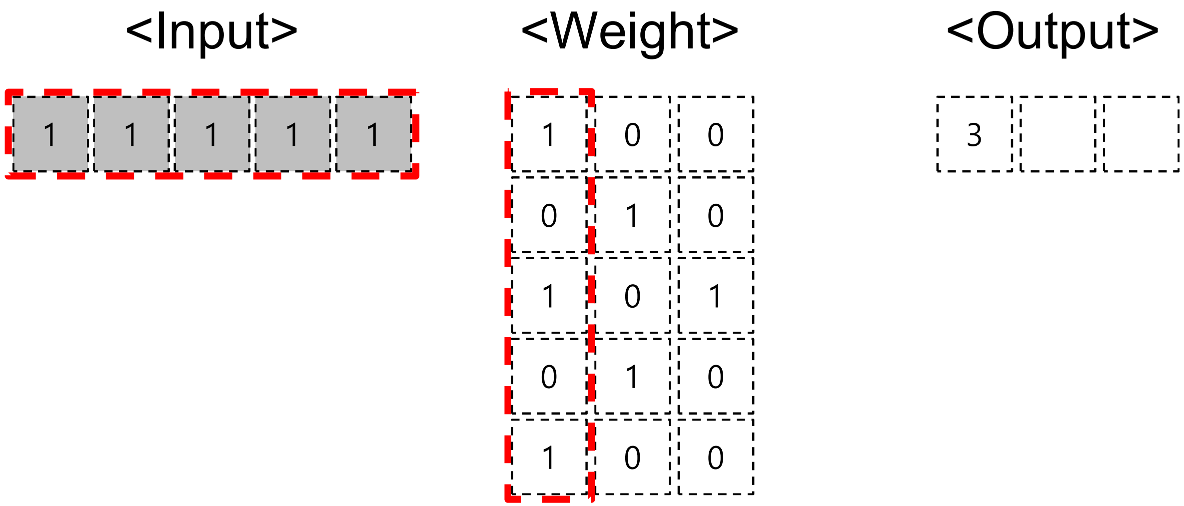
<Weight>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |

<Output>

| | | |
|---|---|---|
| | | |

# Preliminaries – Linear operation



# Preliminaries – Linear operation

<Input>

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |

<Weight>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |

<Output>

| | | |
|---|---|---|
| 3 | 2 | |

# Preliminaries – Linear operation

<Input>

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |

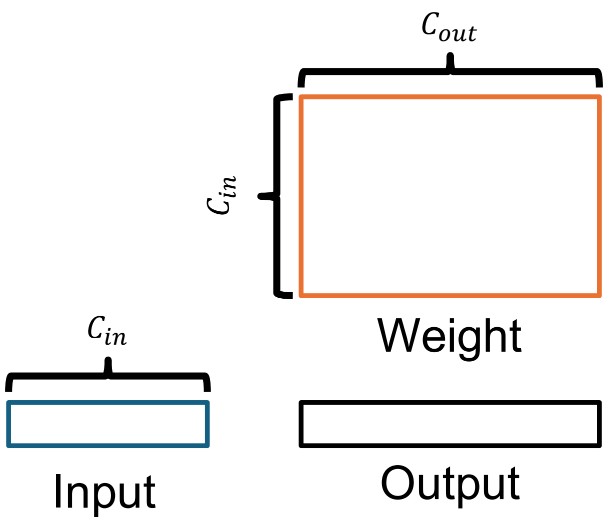
<Weight>

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |

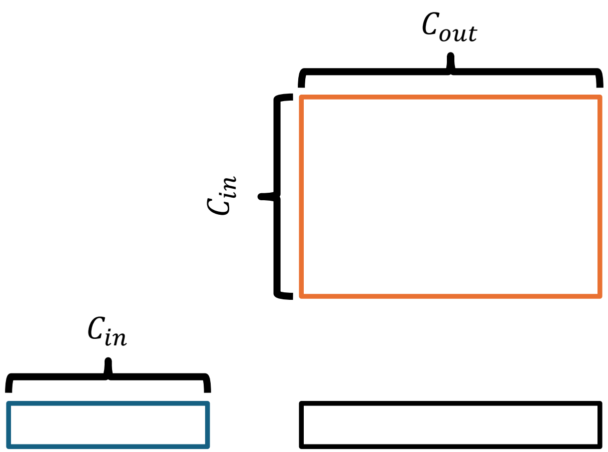
<Output>

| | | |
|---|---|---|
| 3 | 2 | 1 |

# Preliminaries – Linear operation



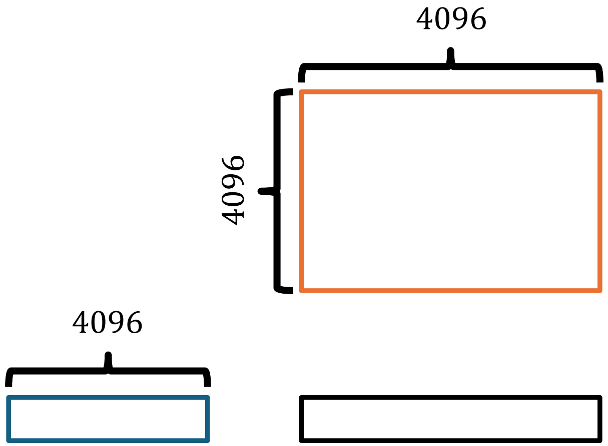
# Preliminaries – Linear operation




$$FLOPs = (2 * C_{in} - 1) \times C_{out}$$

$$MEM = C_{in} + [C_{in} \times C_{out}]$$

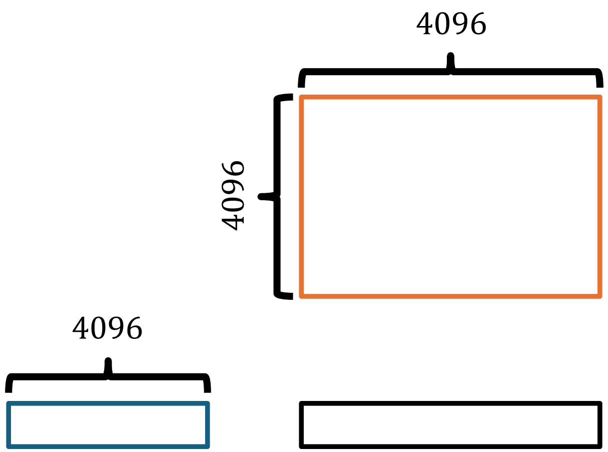
# Preliminaries – Linear operation




Example with AlexNet

$$FLOPs = (2 * 4096 - 1) \times 4096$$
$$MEM = 4096 + [4096 \times 4096]$$

# Preliminaries – Linear operation

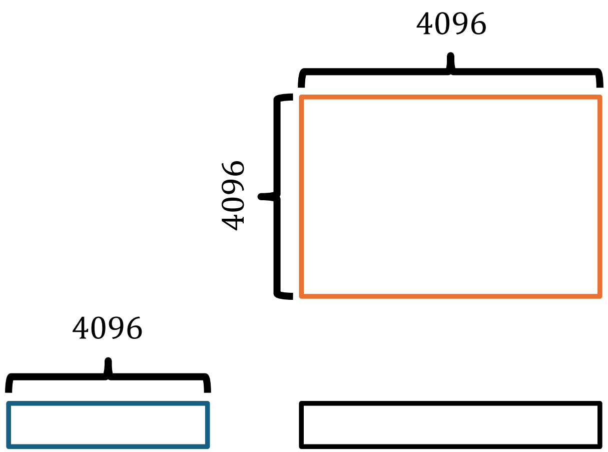



Example with AlexNet

$$FLOPs = 33.55M$$

$$MEM = 4K + 16.7M = 16.78M$$

# Preliminaries – Linear operation



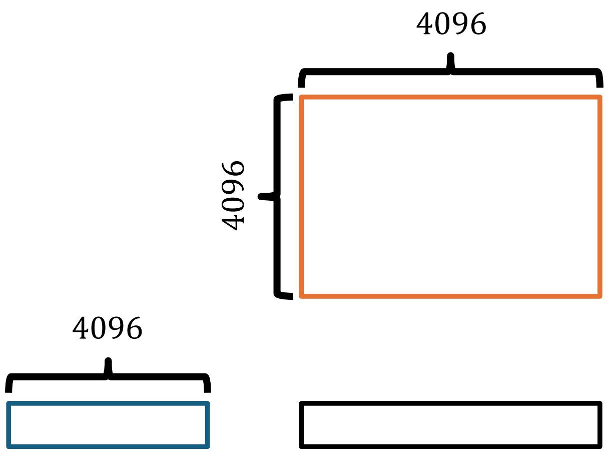

Example with AlexNet

$$FLOPs = 33.55M$$

FP16

$$MEM = 4K + 16.7M = 33.56MB$$

# Preliminaries – Linear operation




Example with AlexNet

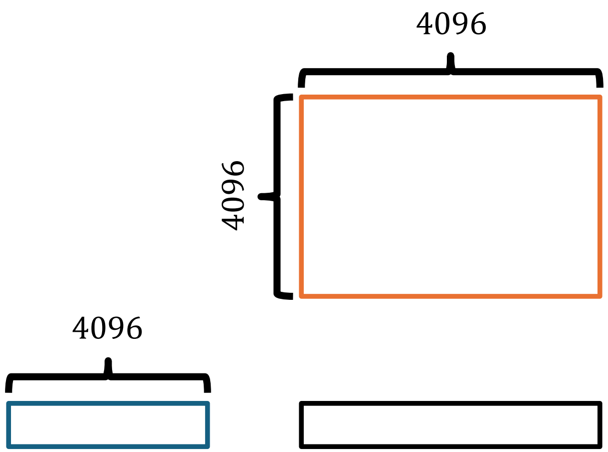
$$FLOPs = 33.55M$$

$$MEM = 4K + 16.7M = 33.56MB$$

Arithmetic intensity of example:
**1.0 FLOPs / Bytes**

Can process 1 Operation every 1 Byte Read.

# Preliminaries – Linear operation




Example with AlexNet

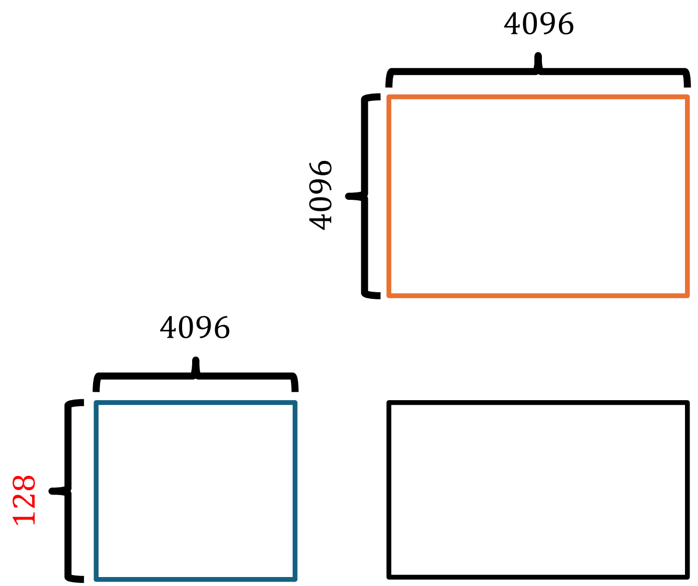
$$FLOPs = 33.55M$$

$$MEM = 4K + 16.7M = 33.56MB$$

Arithmetic intensity of example:
**1.0 FLOPs / Bytes**

**Going to strongly suffer memory bound!**

# Preliminaries – Linear operation



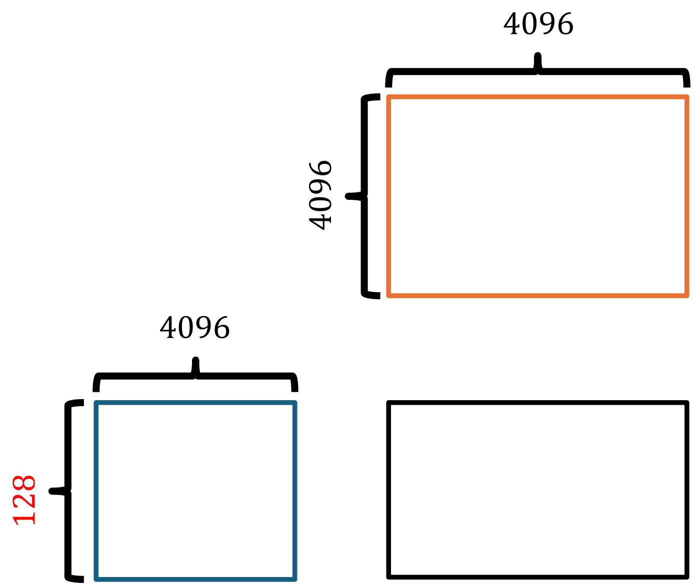

Example with AlexNet

$$FLOPs = 33.55M \times 128 = 4.3B$$

$$MEM = 0.5M + 16.7M = 34.60MB$$

Arithmetic intensity with 128 batch :
**124.1 FLOPs / Bytes**

# Preliminaries – Linear operation




Example with AlexNet

$$FLOPs = 33.55M \times 128 = 4.3B$$

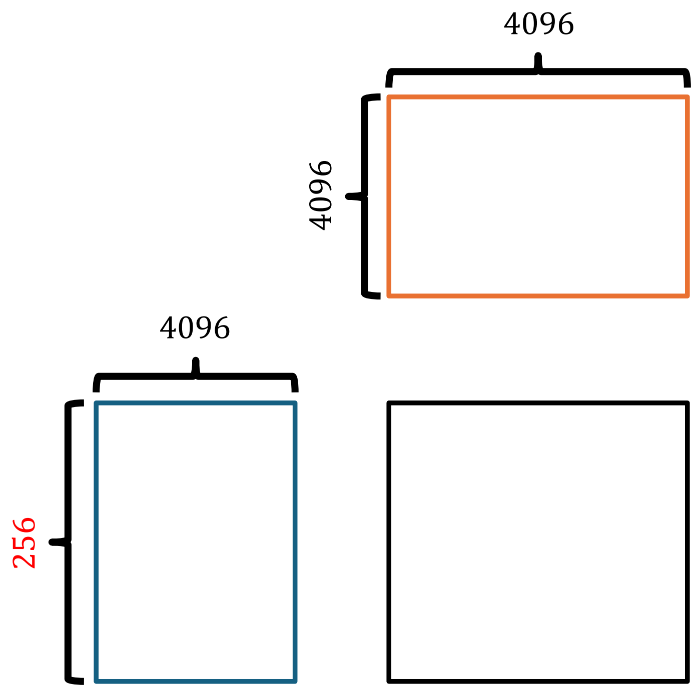
$$MEM = 0.5M + 16.7M = 34.60MB$$

Arithmetic intensity with 128 batch :

**124.1 FLOPs / Bytes**

**Still not enough!**

# Preliminaries – Linear operation



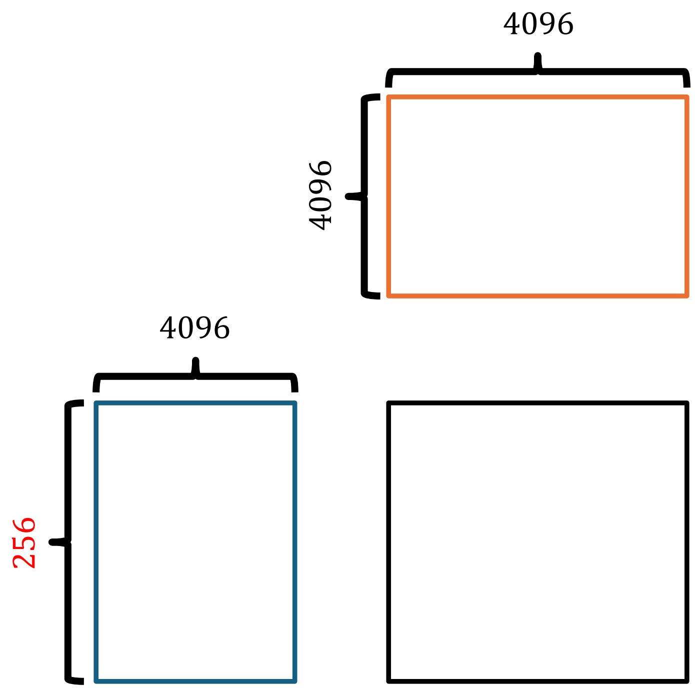

Example with AlexNet

$$FLOPs = 33.55M \times 256 = 8.6B$$

$$MEM = 1.0M + 16.7M = 35.65MB$$

Arithmetic intensity with 256 batch :
**240.91 FLOPs / Bytes**

# Preliminaries – Linear operation




Example with AlexNet

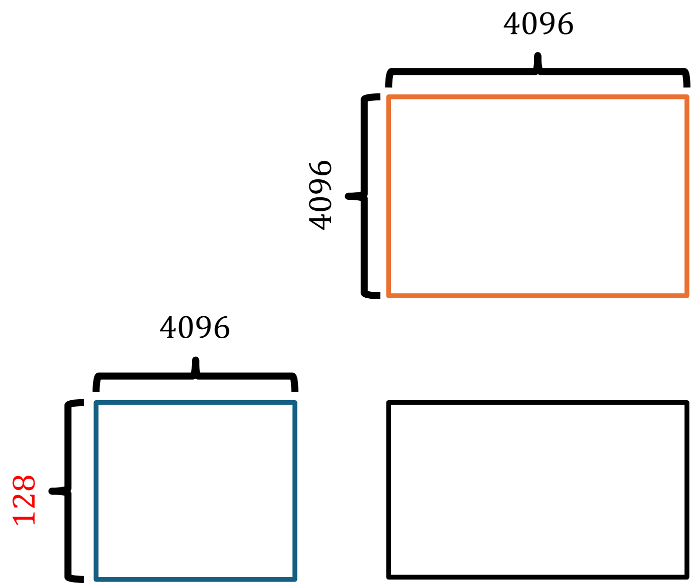
$$FLOPs = 33.55M \times 256 = 8.6B$$

$$MEM = 1.0M + 16.7M = 35.65MB$$

Arithmetic intensity with 256 batch :
**240.91 FLOPs / Bytes**

**Now compute bound!**
**But batch size might be too big for inference.**

# Preliminaries – Linear operation




Example with AlexNet

If we quantize weight to INT8 with 128 batch

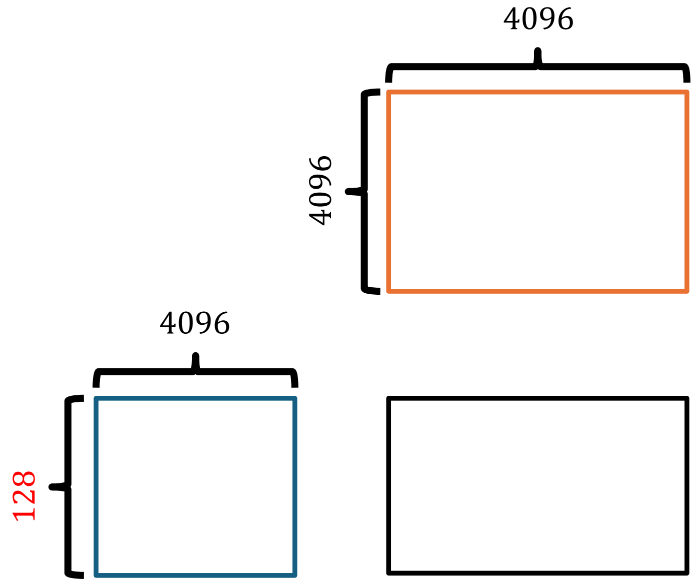
$$FLOPs = 33.55M \times 128 = 4.3B$$

$$MEM = 0.5M + 16.7M = 34.60MB$$

Arithmetic intensity with 128 batch :

**124.1 FLOPs / Bytes**

# Preliminaries – Linear operation




Example with AlexNet

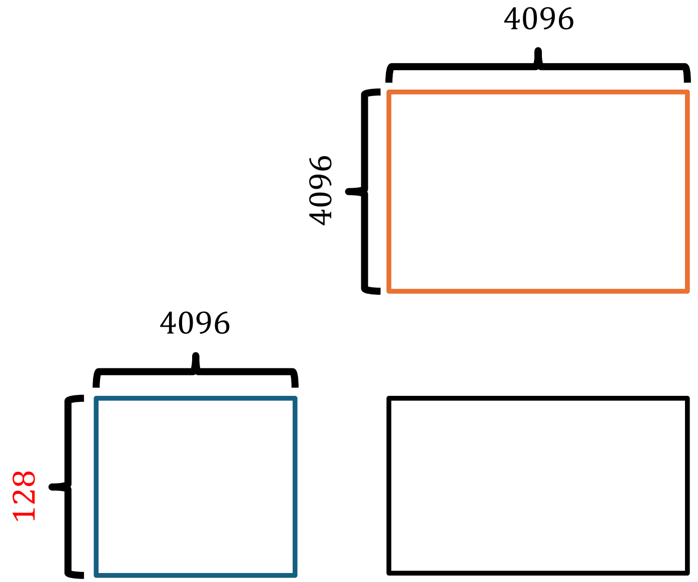
If we quantize weight to INT8 with 128 batch

$$FLOPs = 33.55M \times 128 = 4.3B$$

$$MEM = 0.5M + 16.7M = 17.8MB$$

Arithmetic intensity with 128 batch :
**240.91 FLOPs / Bytes**

# Preliminaries – Linear operation




Example with AlexNet

If we quantize weight to INT8 with 128 batch

$$FLOPs = 33.55M \times 128 = 4.3B$$

$$MEM = 0.5M + 16.7M = 17.8MB$$

Arithmetic intensity with 128 batch :
**240.91 FLOPs / Bytes**

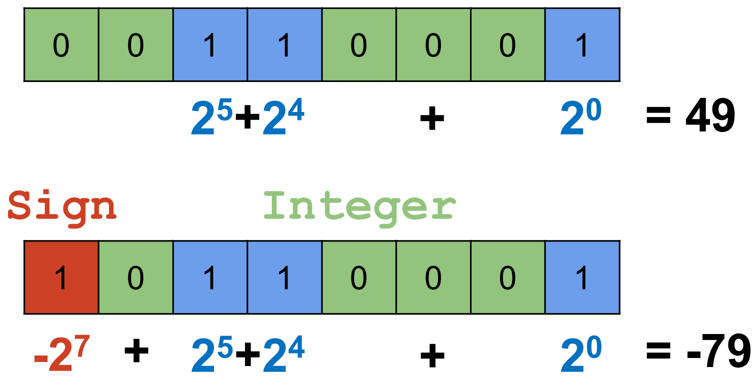
**Lowering memory pressure can increase arithmetic intensity, thus accelerate the layer!**

# Preliminaries

- Major data types used in Deep Learning
  - **Integer**
  - **Floating Point**

# Preliminaries – Data Type

- Major data types used in Deep Learning
  - **Integer**
    - Unsigned Integer: $[0, 2^n - 1]$
    - Signed Integer: $[-2^{n-1}, 2^{n-1} - 1]$



# Preliminaries – Data Type

- Major data types used in Deep Learning
  - **Floating Point**
    - **Numerical Form: $(-1)^s\ M\ 2^E$**
      - **s (Sign bit)** determines whether number is negative or positive
      - **M (Significand)** normally a fractional value in range [1.0,2.0)
      - **E (Exponent)** weights value by power of 2

# Preliminaries – Data Type

- Major data types used in Deep Learning
  - **Floating Point**
    - **Numerical Form: $(-1)^s\ M\ 2^E$**
      - **s (Sign bit)** determines whether number is negative or positive
      - **M (Significand)** normally a fractional value in range [1.0,2.0)
      - **E (Exponent)** weights value by power of 2
    - **Encoding**
      - `s` is sign bit **s**
      - `exp` field encodes **E** (but is not equal to **E**)
      - `frac` field encodes **M** (but is not equal to **M**)

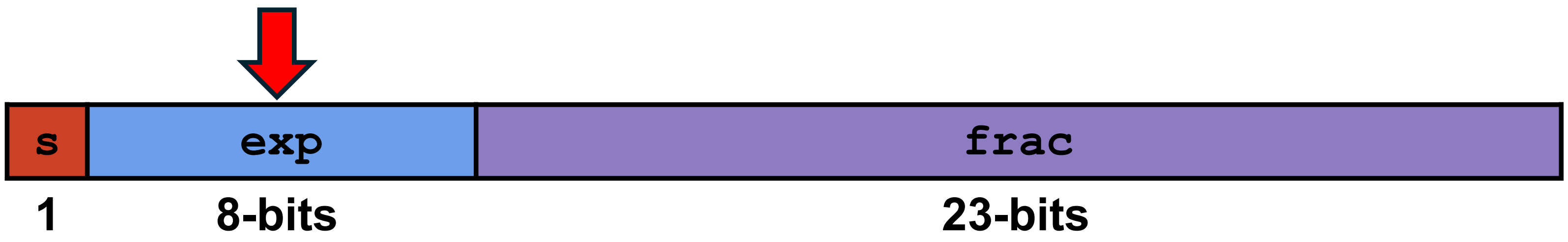
# Preliminaries – Data Type

- Major data types used in Deep Learning
  - **Floating Point**
    - **Single Precision**: 32 bits
      ≈ 7 decimal digits, max/min: $10^{\pm 38}$

# Preliminaries – Data Type

- $V = (-1)^s$ M $2^E$

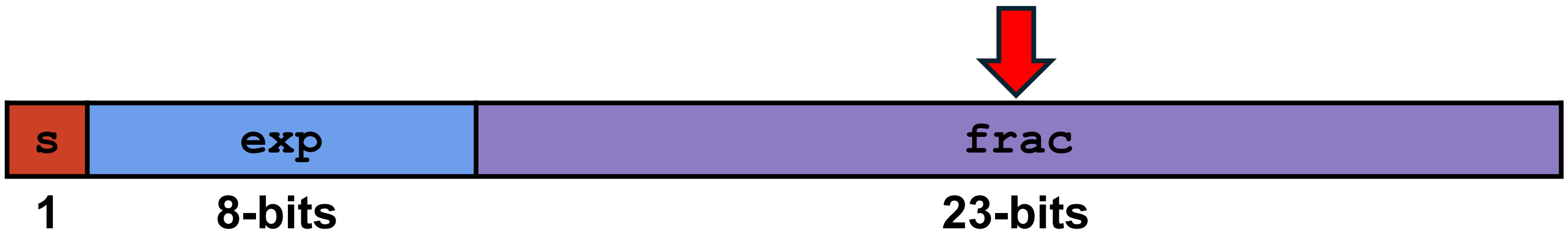
- Major data types used in Deep Learning
  - **Floating Point**
    - **Exponent** coded as a biased value: E = **`exp`** – *Bias*
      - **`exp`**: unsigned value of **`exp`** field
      - *Bias* = $2^{k-1} - 1$, where *k* is number of exponent bits
        - Single precision: 127 (**`exp`**: 1…254, E: -126…127)



# Preliminaries – Data Type

- $V = (-1)^s M\ 2^E$

- Major data types used in Deep Learning
  - **Floating Point**
    - **Significand** coded with implied leading 1: **M = $1.xxx\ldots x_2$**
      - **`xxx...x`**: bits of **`frac`** field
      - Get extra leading bit for "free"



# Preliminaries – Data Type

- $V = (-1)^s\, M\, 2^E$
- $E$ = **`exp`** – *Bias*

- Major data types used in Deep Learning
  - **Floating Point** – Example (float32)
    - **Value** = 15213.0 = $1.1101101101101_2 \times 2^{13}$
    - **Significand**
      - M = $\mathbf{1.\underline{1101101101101}_2}$
      - `frac` = $\mathbf{\underline{1101101101101}0000000000_2}$
    - **Exponent**
      - E = 13
      - *Bias* = 127
      - `exp` = 140 = $\mathbf{10001100_2}$
    - **Result**

| 0 | 1 0 0 0 1 1 0 0 | 1 1 0 1 1 0 1 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 |
|---|---|---|
| **1** | **8-bits** | **23-bits** |

# Preliminaries – Data Type

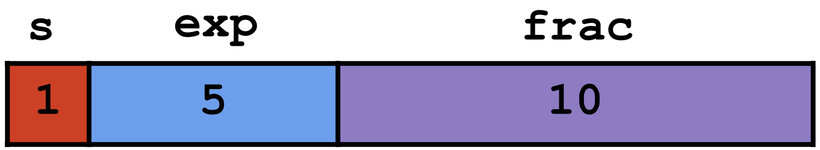
- Major data types used in Deep Learning
  - **Floating Point** – Example
    - IEEE 754 Single Precision 32-bit Float (IEEE FP32)


    - Google Brain Float (BF16)

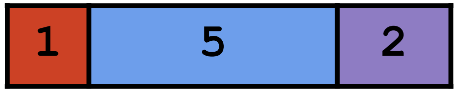

    - IEEE 754 Half Precision 16-bit Float (IEEE FP16)

# Preliminaries – Data Type

- Major data types used in Deep Learning
  - **Floating Point** – Example
    - IEEE 754 Half Precision 16-bit Float (IEEE FP16)




    - NVIDIA FP8 (E5M2)




    - NVIDIA FP8 (E4M3)

# Break

# Quantization Basics

- Fundamentals
- Scheme
- Type

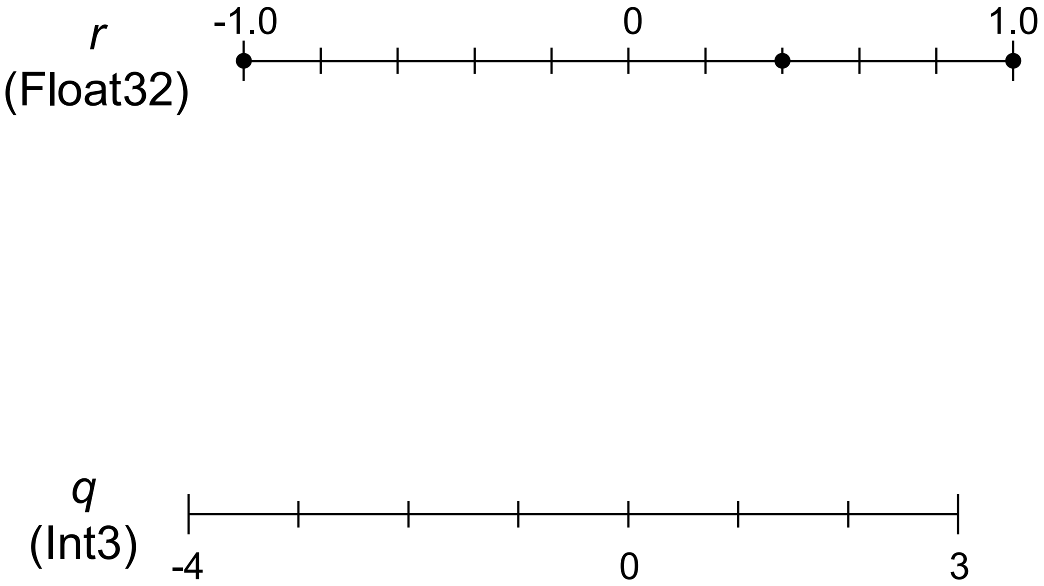
# Quantization Basics – Fundamentals

- **Example** (3-Bit Uniform Quantization)

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

- **Example** (3-Bit Uniform Quantization)

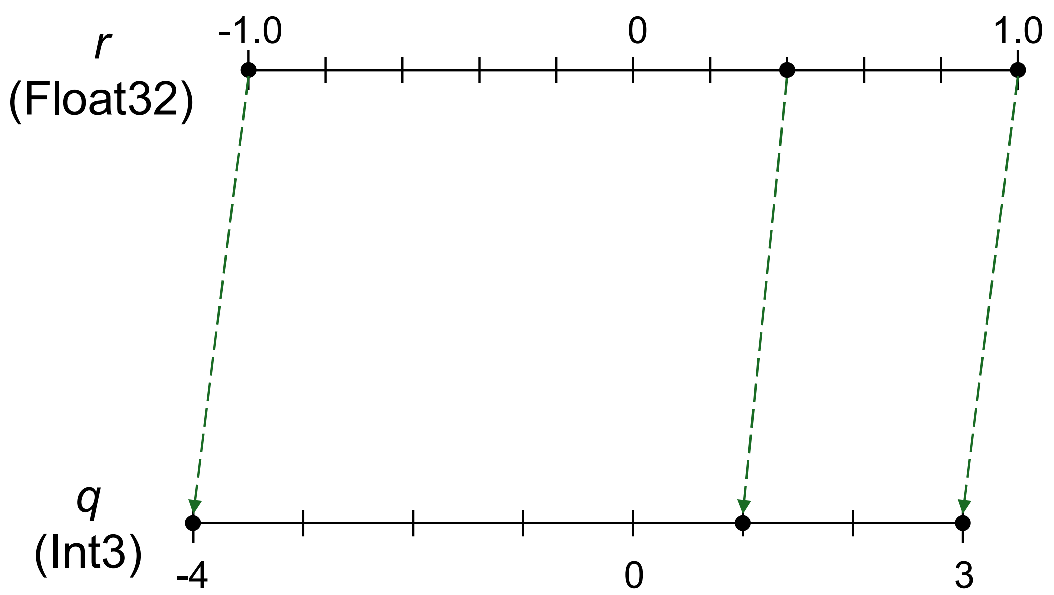



- $S = (Max_r – Min_r) / (2^n - 1)$

- $q = \lfloor r / S \rceil$

# Quantization Basics – Fundamentals

- **Example** (3-Bit Uniform Quantization)



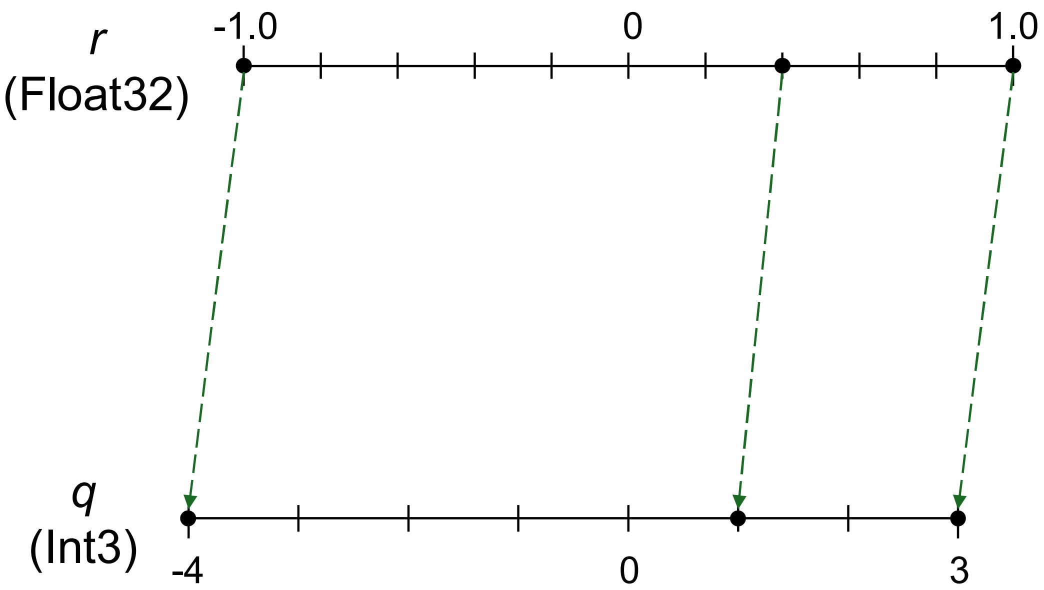

- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

- $S = (Max_r - Min_r) / (2^n - 1)$
- $q = \lfloor r / S \rceil$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

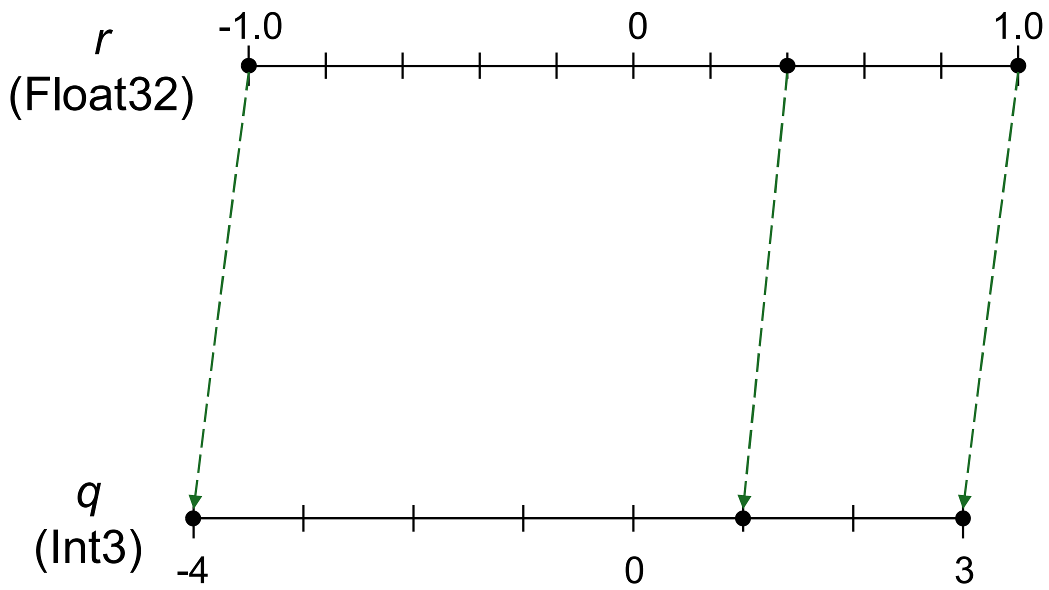
- **Example** (3-Bit Uniform Quantization)




- $S = (1.0 - (-1.0)) / (2^3 - 1)$
- $q = \lfloor r / S \rceil$

# Quantization Basics – Fundamentals

- **Example** (3-Bit Uniform Quantization)




- *r* : real value
- *q* : quantized value
- ⌊⌉ : round
- *S* : scaling factor

- $S = (1.0 - (-1.0)) / (2^3 - 1)$

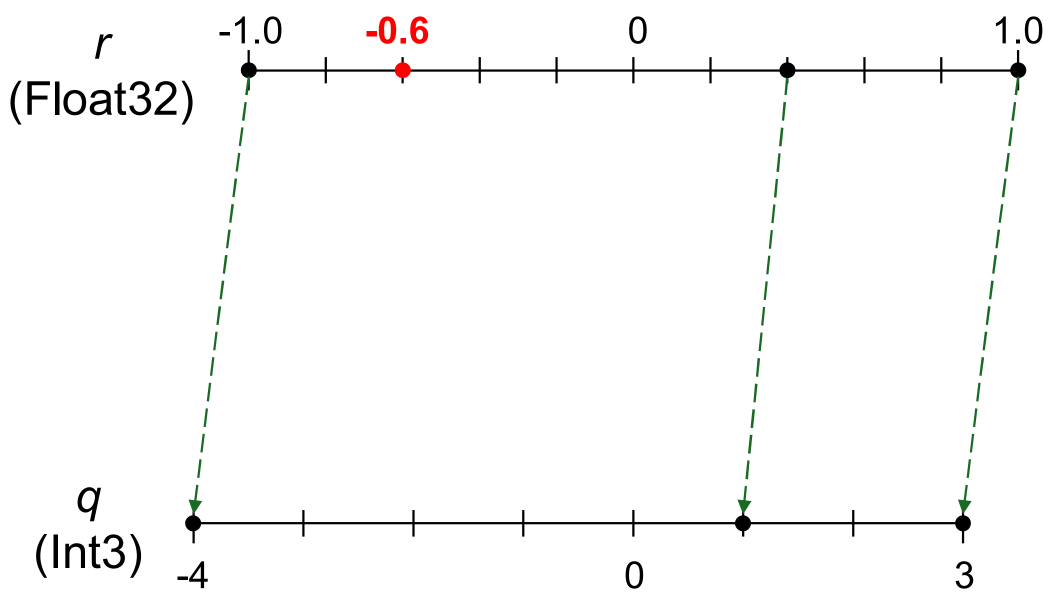
  $= 0.28571$

- $q = \lfloor r / S \rceil$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

- **Example** (3-Bit Uniform Quantization)




- $S = (1.0 - (-1.0)) / (2^3 - 1)$

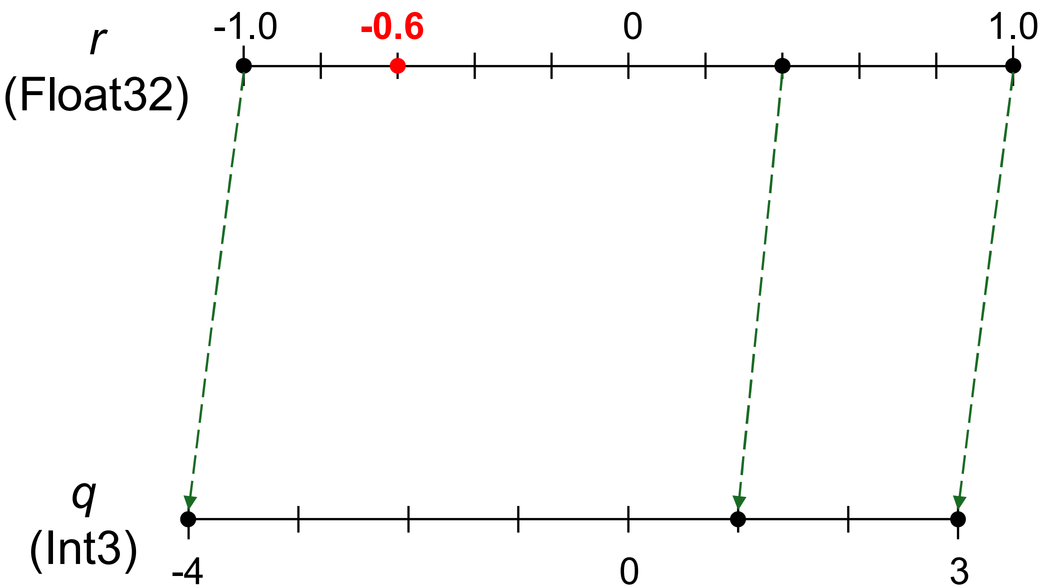
  $= 0.28571$

- $q = \lfloor r / S \rceil$

# Quantization Basics – Fundamentals

- **Example** (3-Bit Uniform Quantization)

- $r$ : real value
- $q$ : quantized value
- $\lfloor \rceil$ : round
- $S$ : scaling factor




- $S = (1.0 - (-1.0)) / (2^3 - 1)$

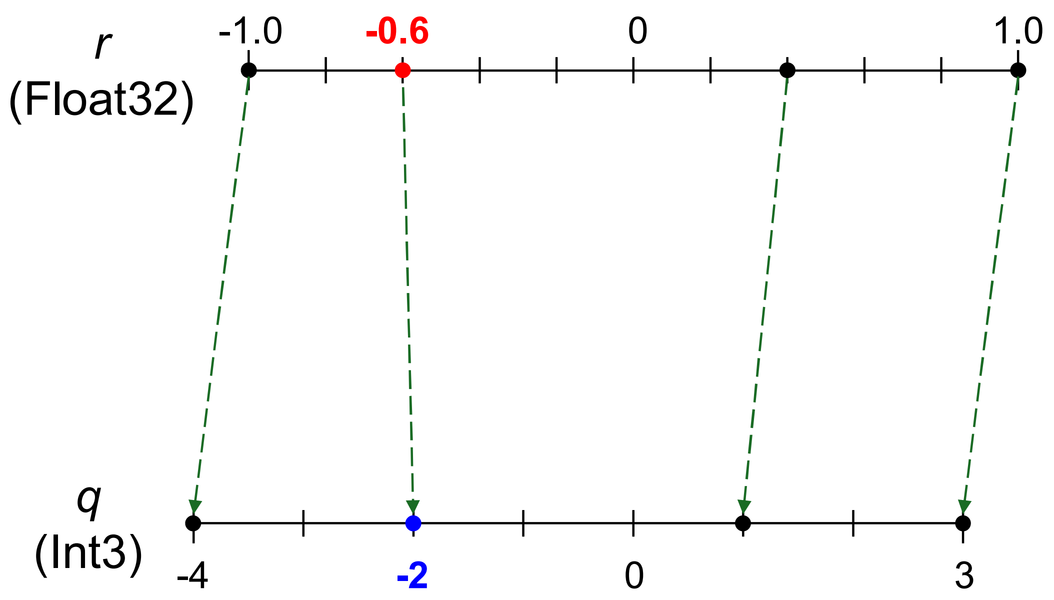
  $= 0.28571$

- $q = \lfloor -0.6 / 0.28571 \rceil$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

- **Example** (3-Bit Uniform Quantization)




- $S = (1.0 - (-1.0)) / (2^3 - 1)$

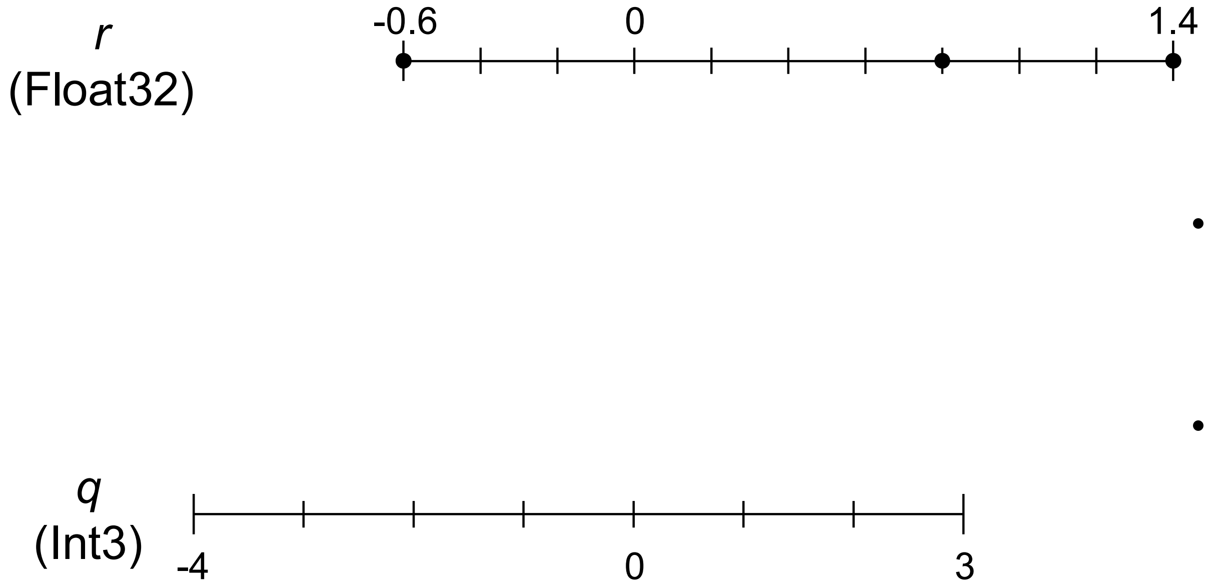
  $= 0.28571$

- $q = \lfloor \color{red}{-0.6} / 0.28571 \rceil$

  $= \color{blue}{-2}$

# Quantization Basics – Fundamentals

- **Example** (3-Bit Uniform Quantization)

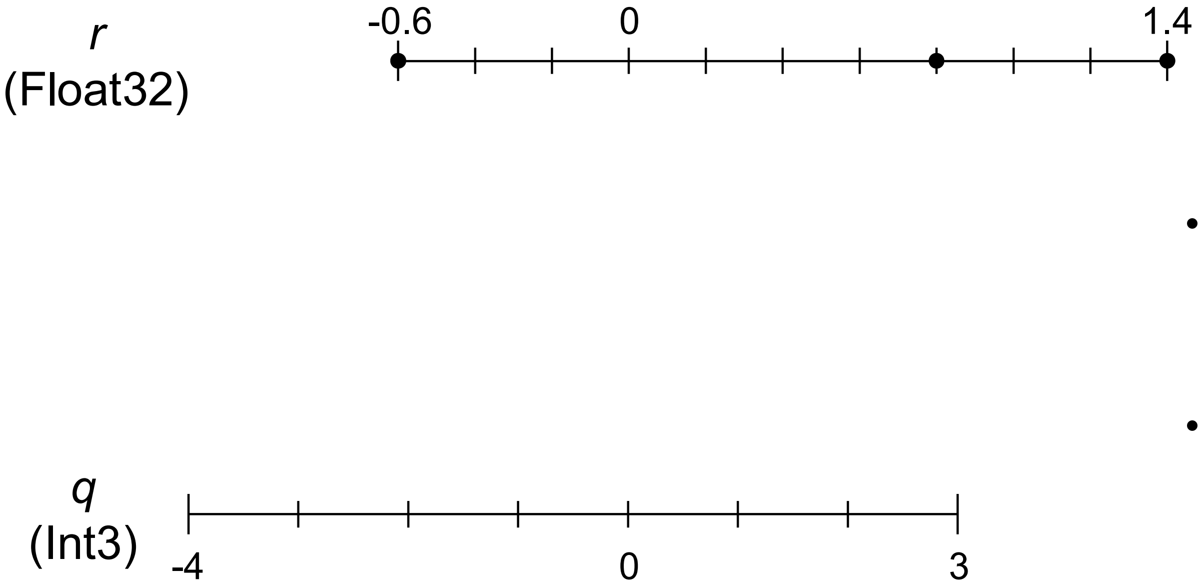



- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

- $S = (Max_r - Min_r) / (2^n - 1)$
- $q = \lfloor r / S \rceil$

# Quantization Basics – Fundamentals

- **Example** (3-Bit Uniform Quantization)



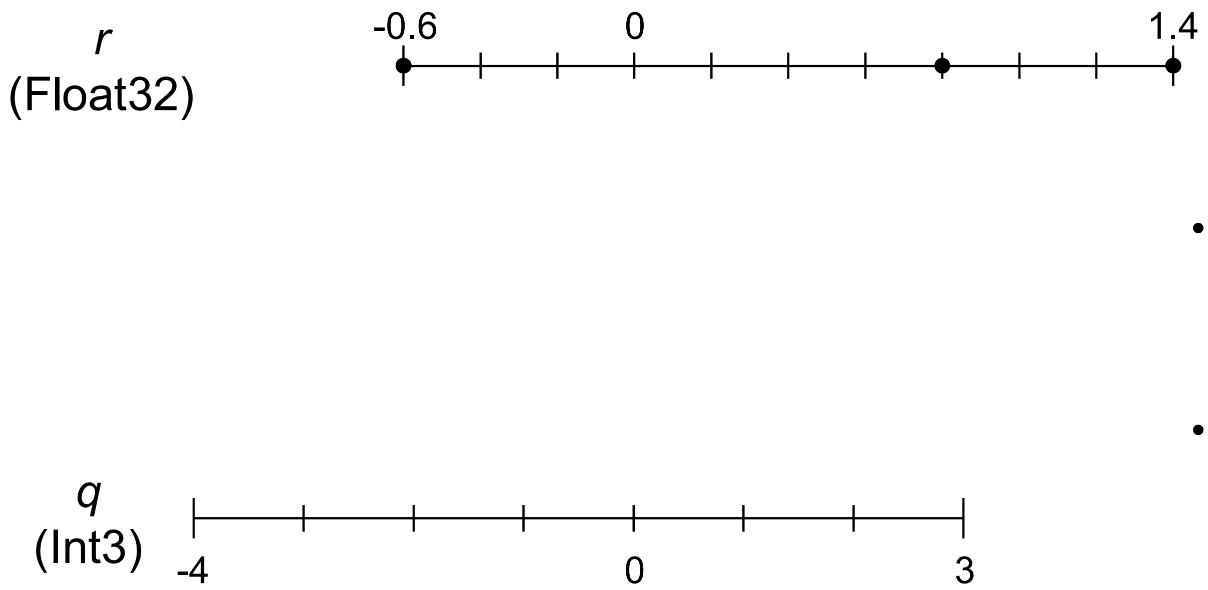

- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

- $S = (1.4 - (-0.6)) / (2^n - 1)$
- $q = \lfloor r / S \rceil$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

- **Example** (3-Bit Uniform Quantization)



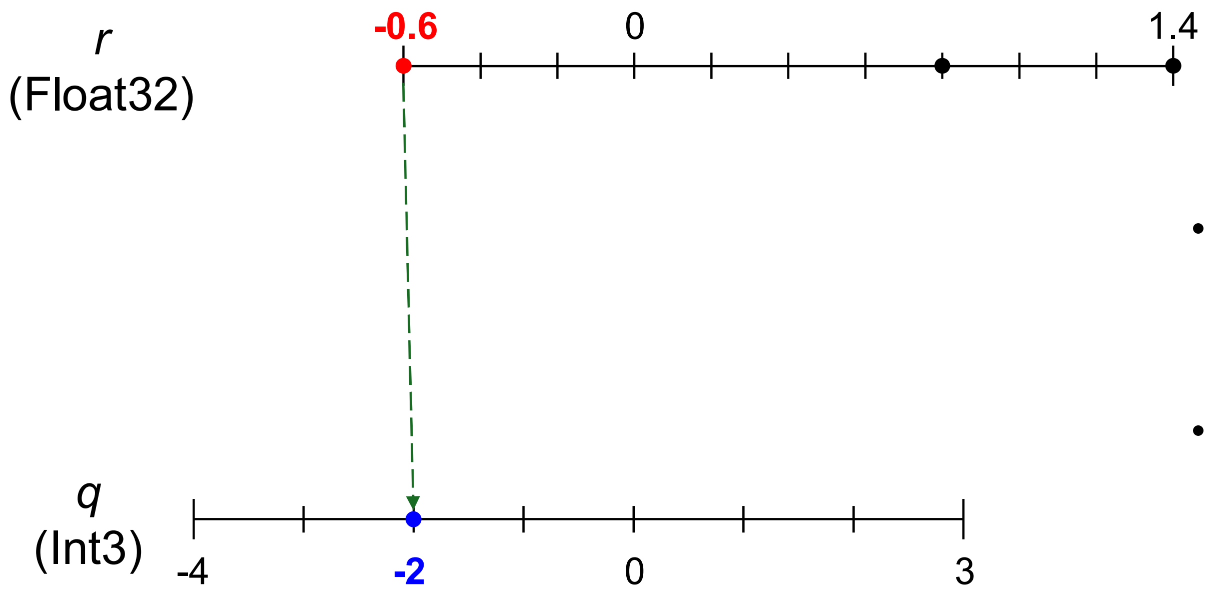

- $S = (1.4 - (-0.6)) / (2^n - 1)$

  $= 0.28571$

- $q = \lfloor r / S \rceil$

# Quantization Basics – Fundamentals

- **Example** (3-Bit Uniform Quantization)




- $r$ : real value
- $q$ : quantized value
- $\lfloor\,\rceil$ : round
- $S$ : scaling factor

- $S = (1.4 - (-0.6)) / (2^n - 1)$

  $= 0.28571$

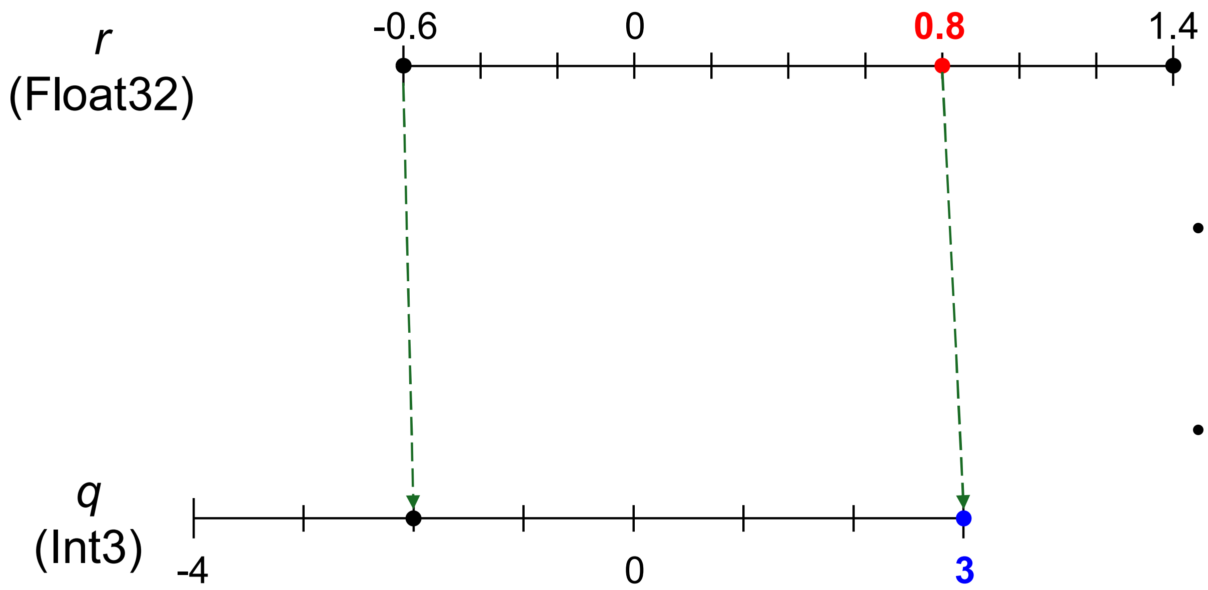
- $q = \lfloor -0.6 / 0.28571 \rceil$

  $= -2$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

- **Example** (3-Bit Uniform Quantization)




- $S = (1.4 - (-0.6)) / (2^n - 1)$

  $= 0.28571$

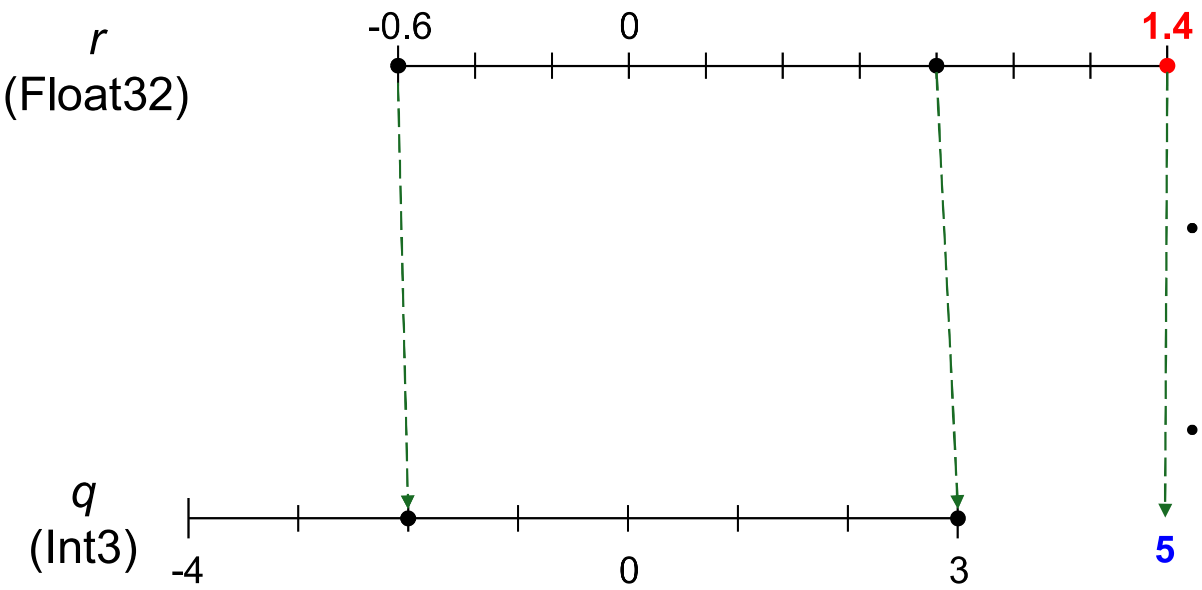
- $q = \lfloor 0.8 / 0.28571 \rceil$

  $= 3$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

- **Example** (3-Bit Uniform Quantization)




- $S = (1.4 - (-0.6)) / (2^n - 1)$

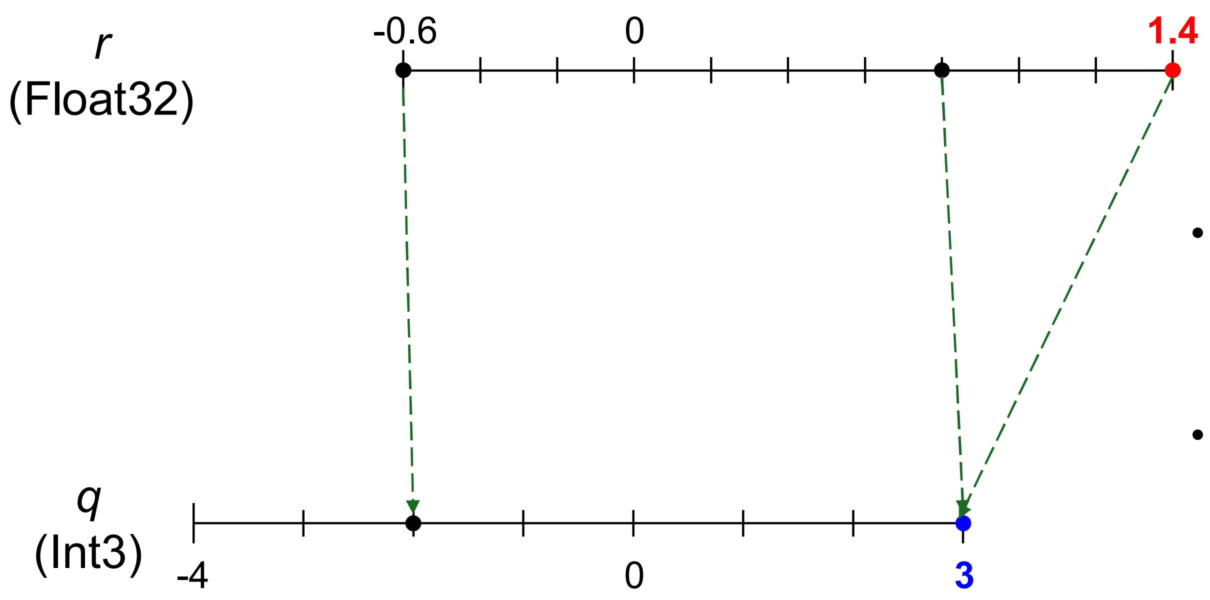
  $= 0.28571$

- $q = \lfloor 1.4 / 0.28571 \rceil$

  $= 5$

# Quantization Basics – Fundamentals

- **Example** (3-Bit Uniform Quantization)




- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

- $S = (1.4 - (-0.6)) / (2^n - 1)$

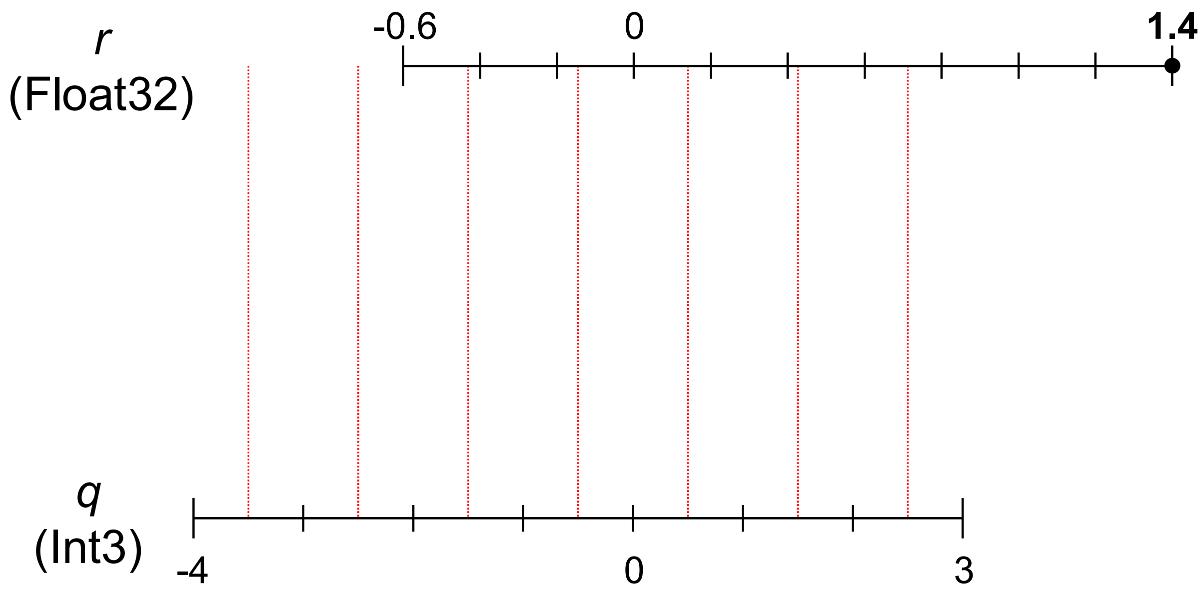
  $= 0.28571$

- $q = clamp(\lfloor 1.4 / 0.28571 \rceil)$

  $= 3$

# Quantization Basics – Fundamentals

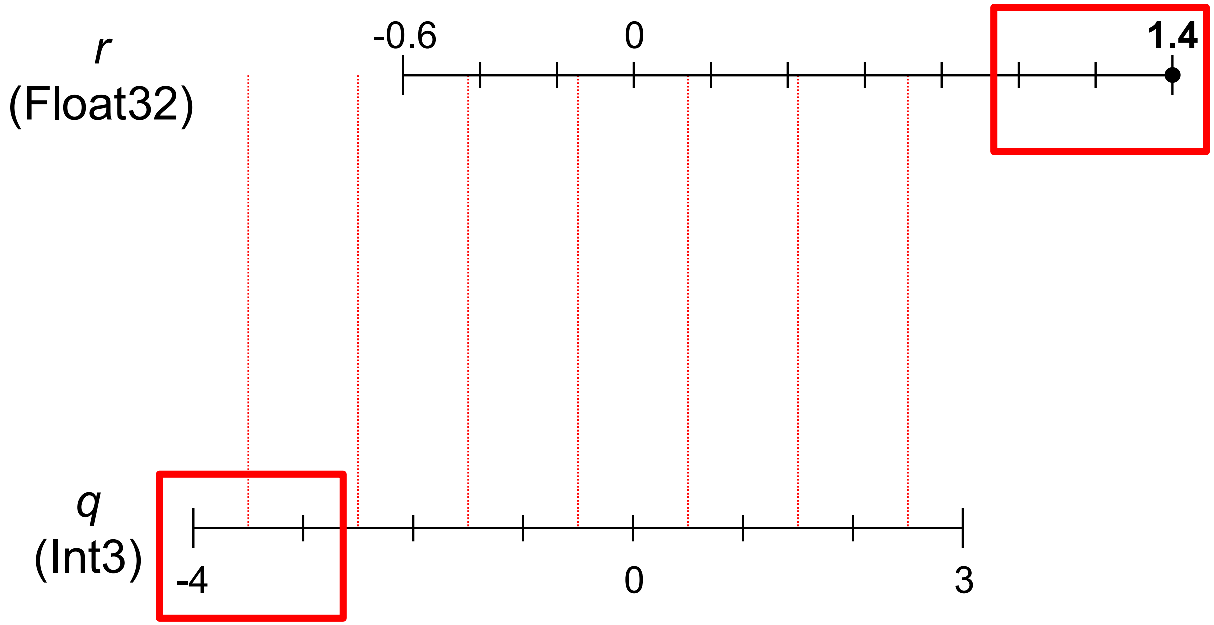
- **Example** (3-Bit Uniform Quantization)




- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

# Quantization Basics – Fundamentals

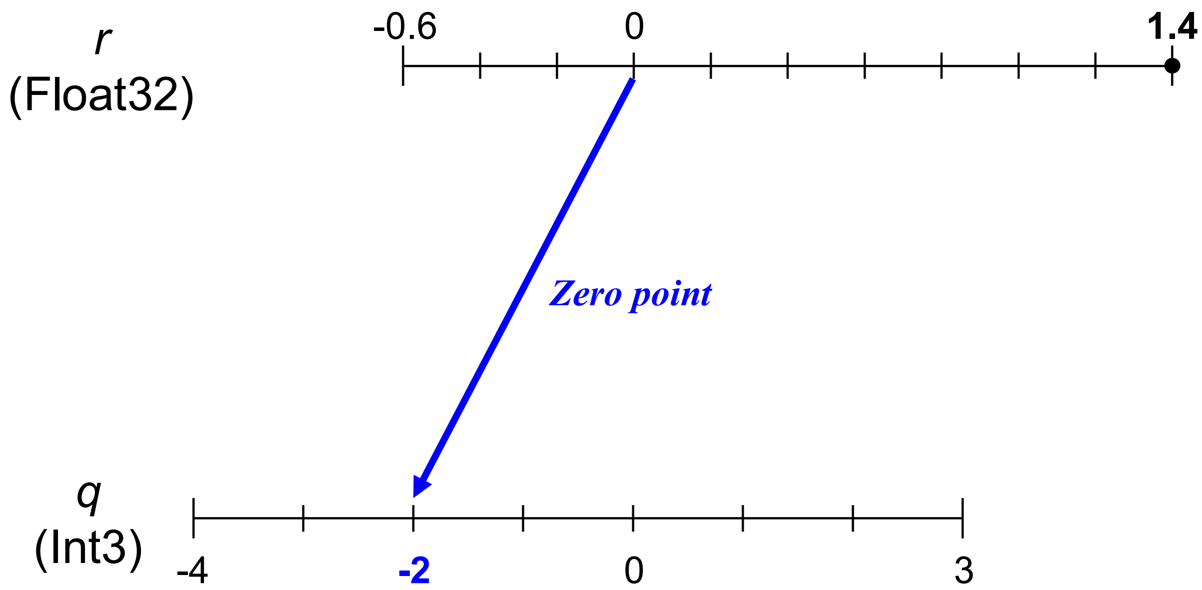
- **Example** (3-Bit Uniform Quantization)




- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

# Quantization Basics – Fundamentals

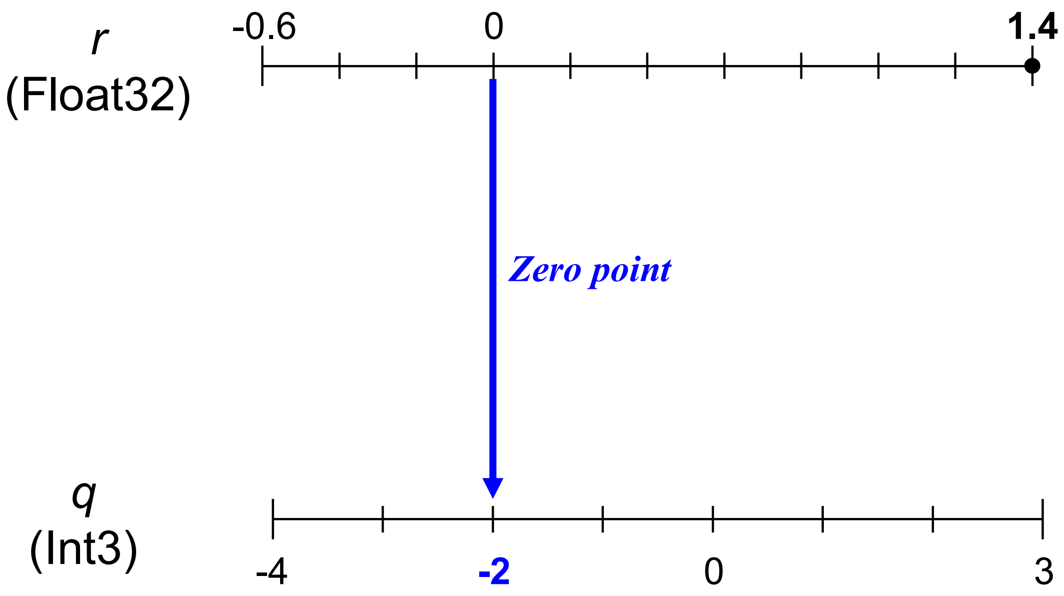
- **Example** (3-Bit Uniform Quantization)




- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

# Quantization Basics – Fundamentals

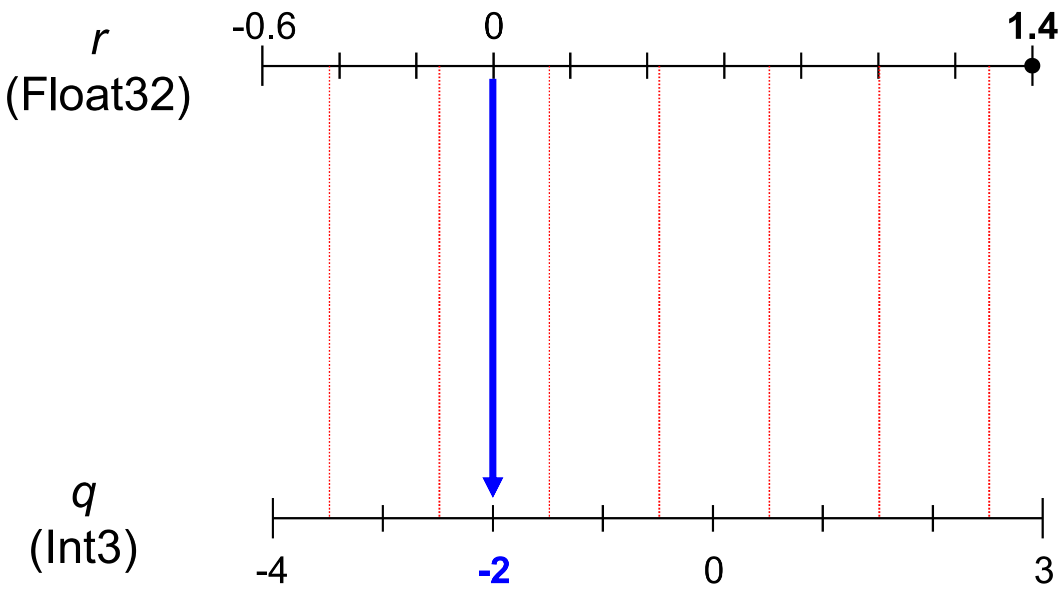
- **Example** (3-Bit Uniform Quantization)




- $r$ : real value
- $q$ : quantized value
- $\lfloor \rceil$ : round
- $S$ : scaling factor

# Quantization Basics – Fundamentals

- **Example** (3-Bit Uniform Quantization)

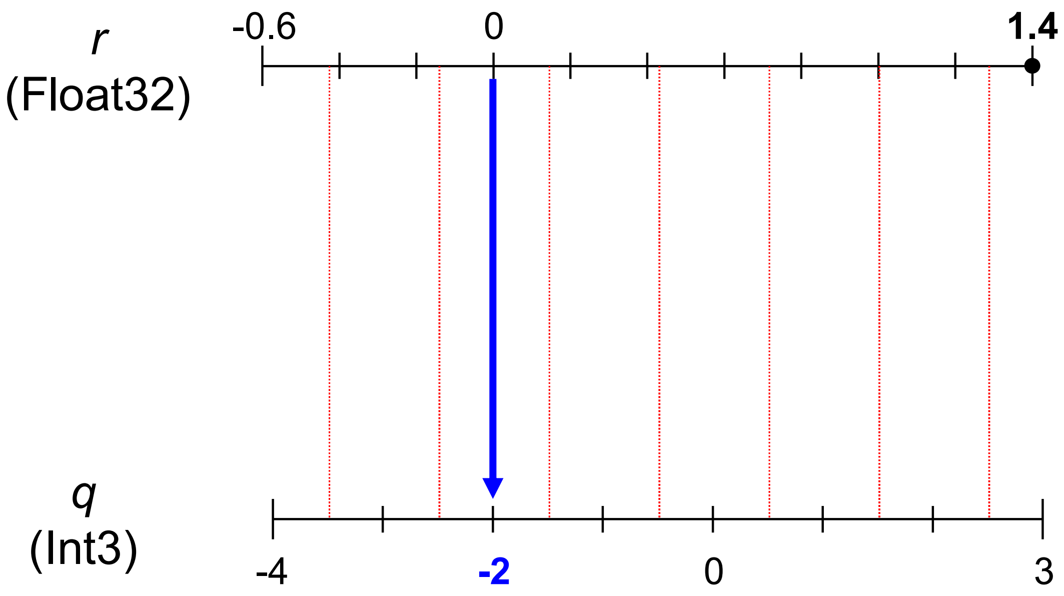



- $r$ : real value
- $q$ : quantized value
- $\lfloor\rceil$ : round
- $S$ : scaling factor

# Quantization Basics – Fundamentals

- *r* : real value
- *q* : quantized value
- ⌊⌉ : round
- *S* : scaling factor
- *Z* : zero-point

- **Example** (3-Bit Uniform Quantization)

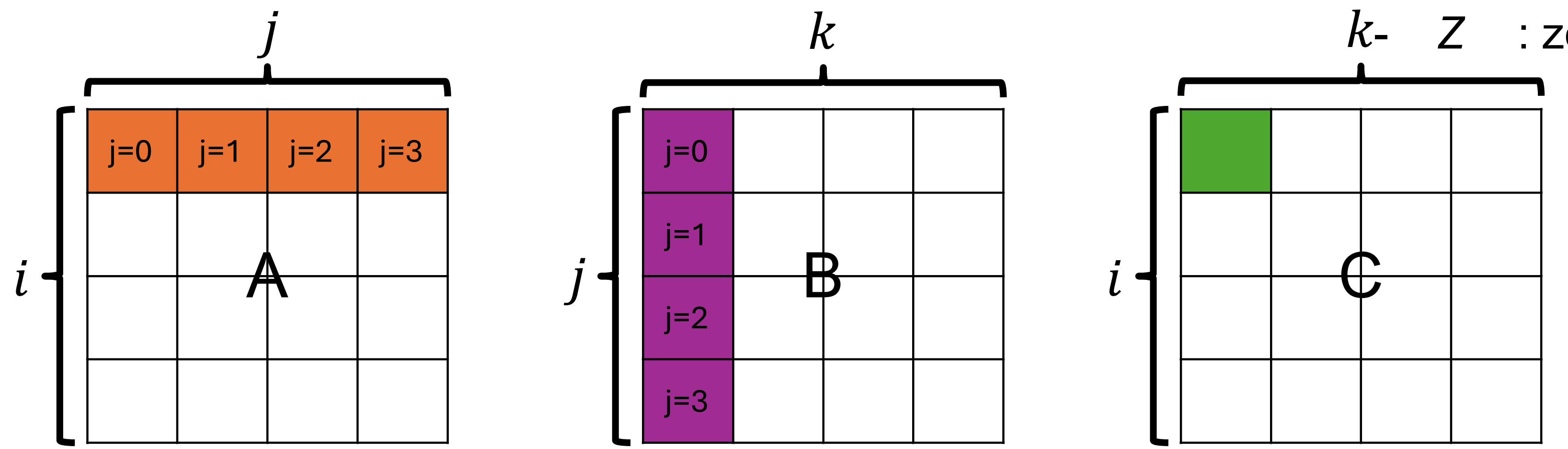



- $S = (Max_r - Min_r) / (2^n - 1)$
- $Z = Min_q - (Min_r / S)$
- $q = \lfloor r / S \rceil + Z$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $S$ : scaling factor
- $Z$ : zero point

- **Quantized Matrix Multiplication**

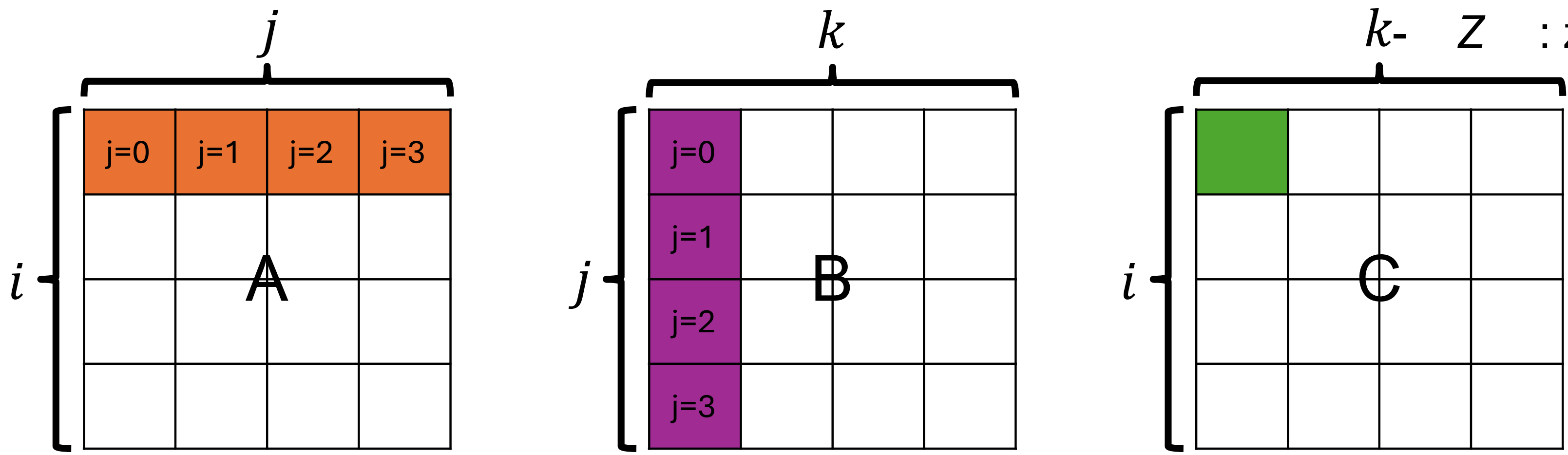



$$r_C^{(i,k)} = \sum_j r_A^{(i,j)} \times r_B^{(j,k)}$$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $S$ : scaling factor
- $Z$ : zero point

- **Quantized Matrix Multiplication**




$$S_C\left(q_C^{(i,k)} - Z_C\right) = \sum_j S_A\left(q_A^{(i,j)} - Z_A\right) S_B\left(q_B^{(j,k)} - Z_B\right)$$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $S$ : scaling factor
- $Z$ : zero point

- **Quantized Matrix Multiplication**

$$S_C\left(q_C^{(i,k)} - Z_C\right) = \sum_j S_A\left(q_A^{(i,j)} - Z_A\right) S_B\left(q_B^{(j,k)} - Z_B\right)$$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $S$ : scaling factor
- $Z$ : zero point

- **Quantized Matrix Multiplication**

$$S_C\left(q_C^{(i,k)} - Z_C\right) = \sum_j S_A\left(q_A^{(i,j)} - Z_A\right) S_B\left(q_B^{(j,k)} - Z_B\right)$$

$$\Downarrow$$

$$q_C^{(i,k)} = Z_C + \frac{S_A S_B}{S_C} \sum_j \left(q_A^{(i,j)} - Z_A\right)\left(q_B^{(j,k)} - Z_B\right)$$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $S$ : scaling factor
- $Z$ : zero point

- **Quantized Matrix Multiplication**

$$q_C^{(i,k)} = Z_C + \frac{S_A S_B}{S_C} \sum_j \left(q_A^{(i,j)} - Z_A\right) \left(q_B^{(j,k)} - Z_B\right)$$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $S$ : scaling factor
- $Z$ : zero point

- **Quantized Matrix Multiplication**

$$q_C^{(i,k)} = Z_C + \frac{S_A S_B}{S_C} \sum_j \left(q_A^{(i,j)} - Z_A\right) \left(q_B^{(j,k)} - Z_B\right)$$

$$\Downarrow$$

$$q_C^{(i,k)} = Z_C + \frac{S_A S_B}{S_C} \left[ N Z_A Z_B - Z_B \sum q_A^{(i,j)} - Z_A \sum q_B^{(j,k)} + \sum q_A^{(i,j)} q_B^{(j,k)} \right]$$

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $S$ : scaling factor
- $Z$ : zero point

- **Quantized Matrix Multiplication**

$$q_C^{(i,k)} = Z_C + \frac{S_A S_B}{S_C} \sum_j \left(q_A^{(i,j)} - Z_A\right) \left(q_B^{(j,k)} - Z_B\right)$$

$$\Downarrow$$

$$q_C^{(i,k)} = Z_C + \frac{S_A S_B}{S_C} \left[ N Z_A Z_B - Z_B \sum q_A^{(i,j)} - Z_A \sum q_B^{(j,k)} + \boxed{\sum q_A^{(i,j)} q_B^{(j,k)}} \right]$$

**Integer Matmul Operation**

# Quantization Basics – Fundamentals

- $r$ : real value
- $q$ : quantized value
- $S$ : scaling factor
- $Z$ : zero point

- **Quantized Matrix Multiplication**

$$q_C^{(i,k)} = Z_C + \frac{S_A S_B}{S_C} \sum_j \left(q_A^{(i,j)} - Z_A\right) \left(q_B^{(j,k)} - Z_B\right)$$

$$\Downarrow$$

$$q_C^{(i,k)} = Z_C + \frac{S_A S_B}{S_C} \left[ \underline{N Z_A Z_B - Z_B \sum q_A^{(i,j)} - Z_A \sum q_B^{(j,k)}} + \sum q_A^{(i,j)} q_B^{(j,k)} \right]$$

- If we don't use **Zero Point**, we can get rid of a lot of computations

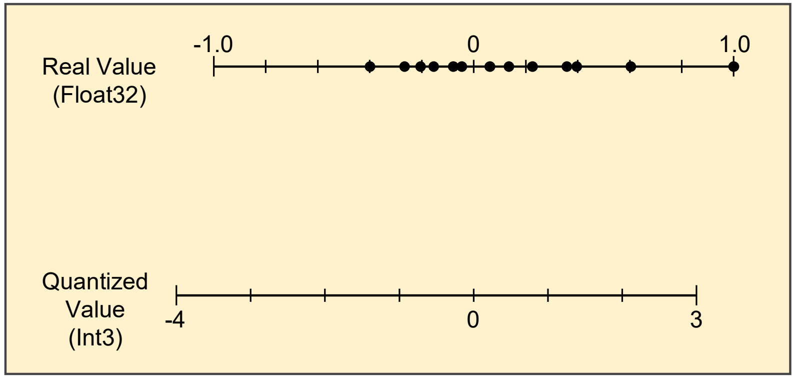
# Quantization Basics – Fundamentals

- **Asymmetric Quantization**
  - Use Zero Point
  - Relatively Slow

- **Symmetric Quantization**
  - Do not use Zero Point
  - Relatively Fast

# Quantization Basics – Fundamentals

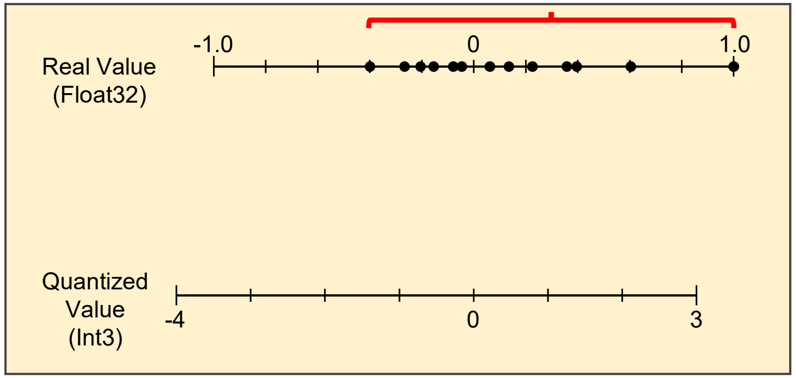
- **Asymmetric Quantization**
  - Use Zero Point
  - Relatively Slow




- **Symmetric Quantization**
  - Do not use Zero Point
  - Relatively Fast

# Quantization Basics – Fundamentals

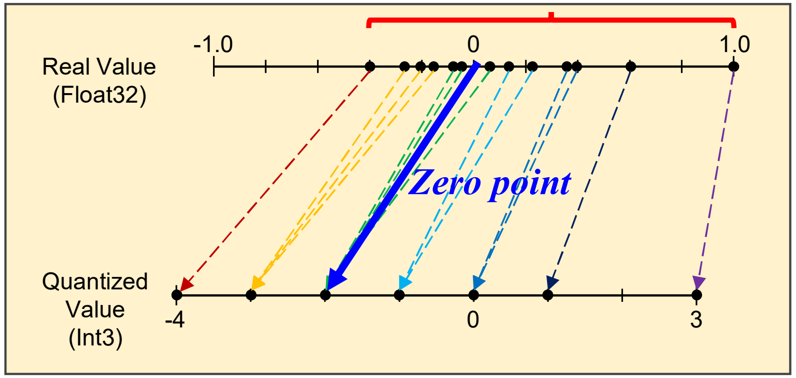
- **Asymmetric Quantization**
  - Use Zero Point
  - Relatively Slow




- **Symmetric Quantization**
  - Do not use Zero Point
  - Relatively Fast

# Quantization Basics – Fundamentals

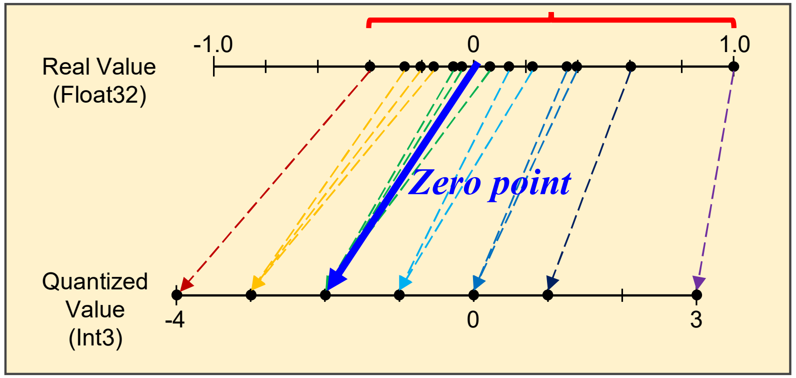
- **Asymmetric Quantization**
  - Use Zero Point
  - Relatively Slow




- **Symmetric Quantization**
  - Do not use Zero Point
  - Relatively Fast

# Quantization Basics – Fundamentals

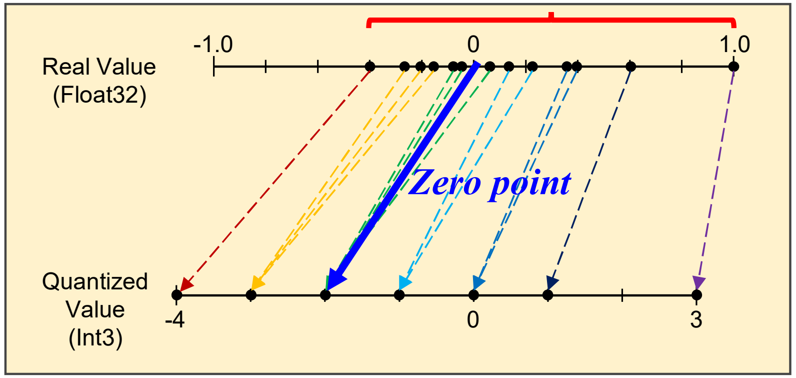
- **Asymmetric Quantization**
  - Use Zero Point
  - Relatively Slow
  - Can utilize any possible range



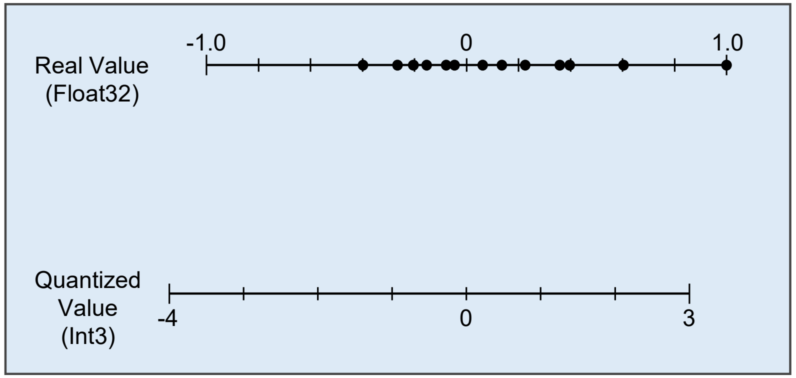

- **Symmetric Quantization**
  - Do not use Zero Point
  - Relatively Fast

# Quantization Basics – Fundamentals

- **Asymmetric Quantization**
  - Use Zero Point
  - Relatively Slow
  - Can utilize any possible range

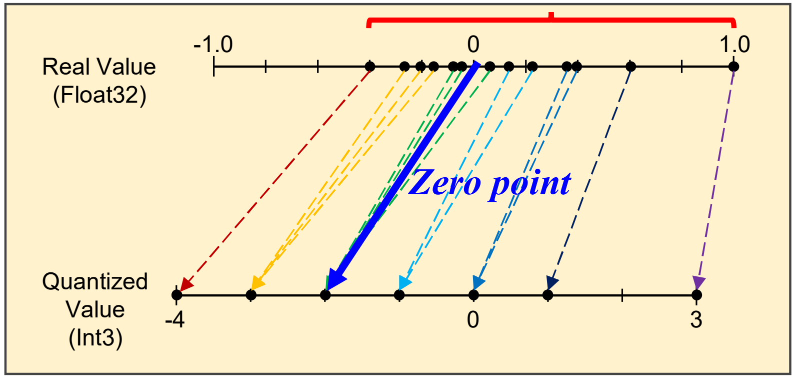



- **Symmetric Quantization**
  - Do not use Zero Point
  - Relatively Fast

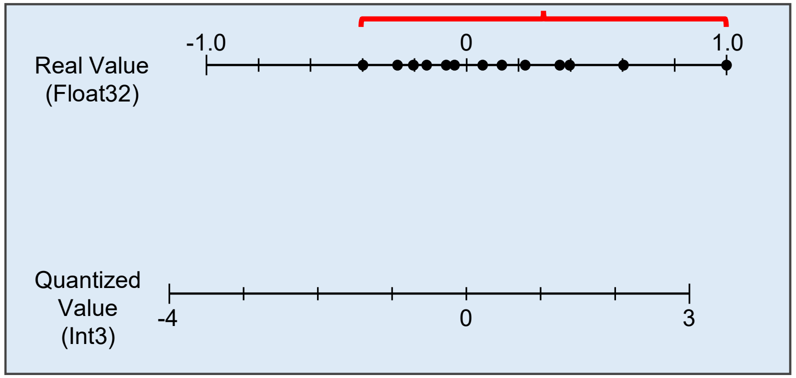


# Quantization Basics – Fundamentals

- **Asymmetric Quantization**
  - Use Zero Point
  - Relatively Slow
  - Can utilize any possible range

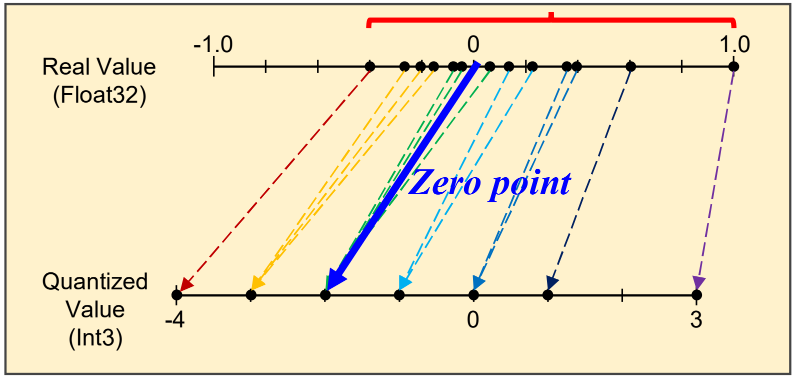



- **Symmetric Quantization**
  - Do not use Zero Point
  - Relatively Fast

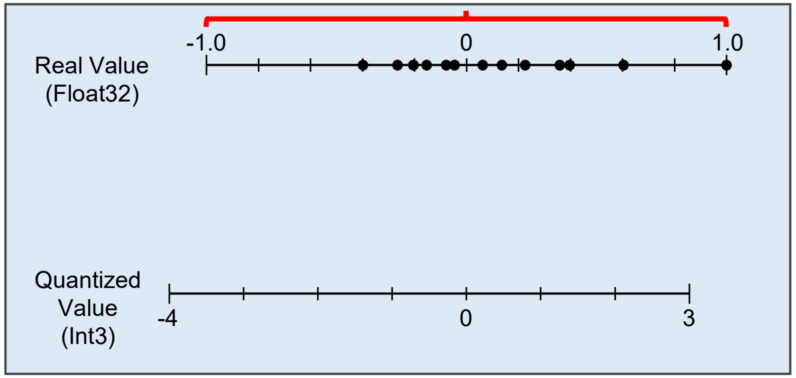


# Quantization Basics – Fundamentals

- **Asymmetric Quantization**
  - Use Zero Point
  - Relatively Slow
  - Can utilize any possible range

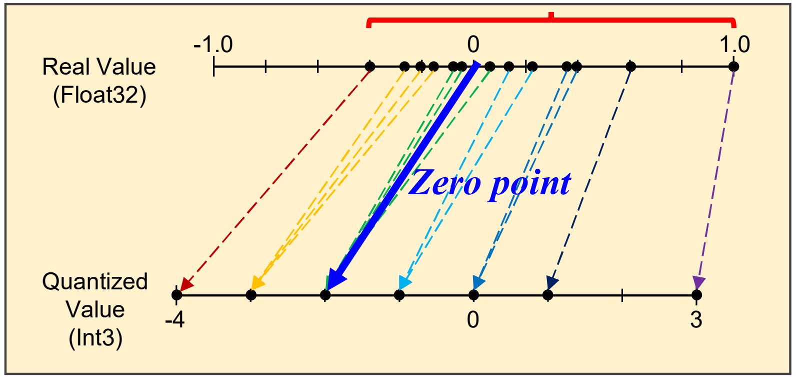



- **Symmetric Quantization**
  - Do not use Zero Point
  - Relatively Fast

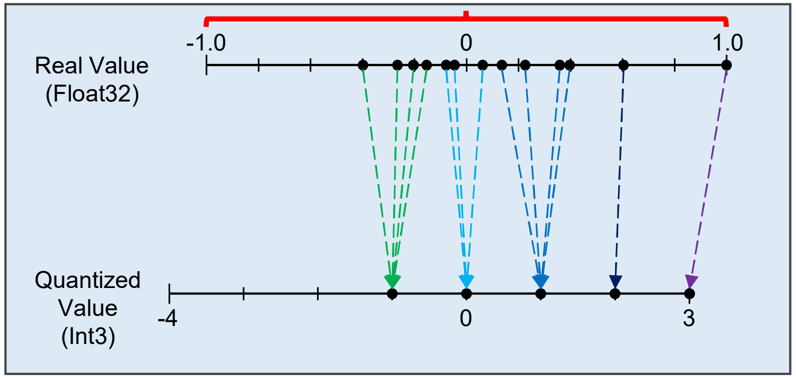


# Quantization Basics – Fundamentals

- **Asymmetric Quantization**
  - Use Zero Point
  - Relatively Slow
  - Can utilize any possible range

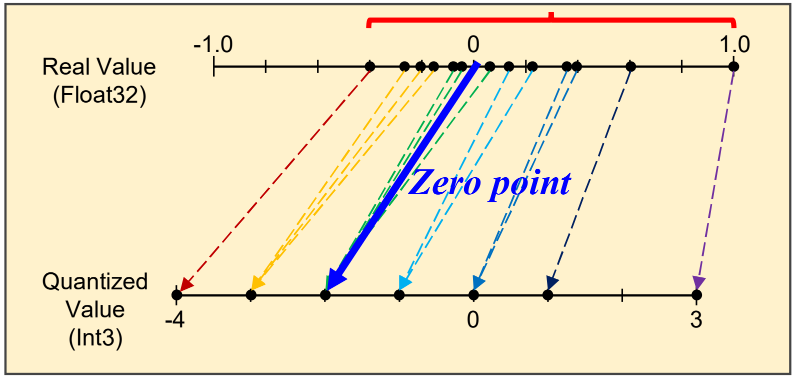



- **Symmetric Quantization**
  - Do not use Zero Point
  - Relatively Fast

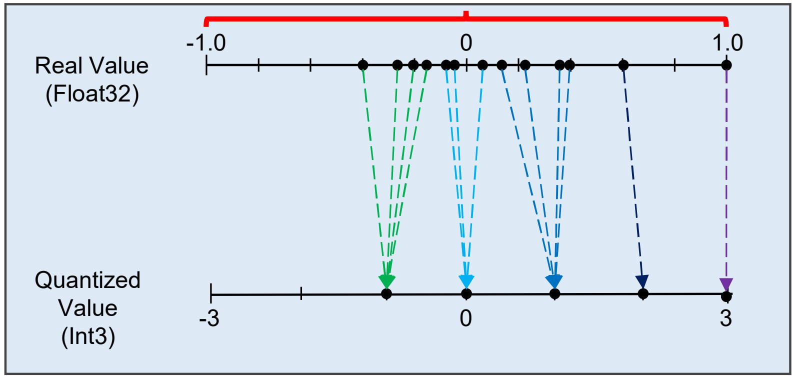


# Quantization Basics – Fundamentals

- **Asymmetric Quantization**
  - Use Zero Point
  - Relatively Slow
  - Can utilize any possible range




- **Symmetric Quantization**
  - Do not use Zero Point
  - Relatively Fast

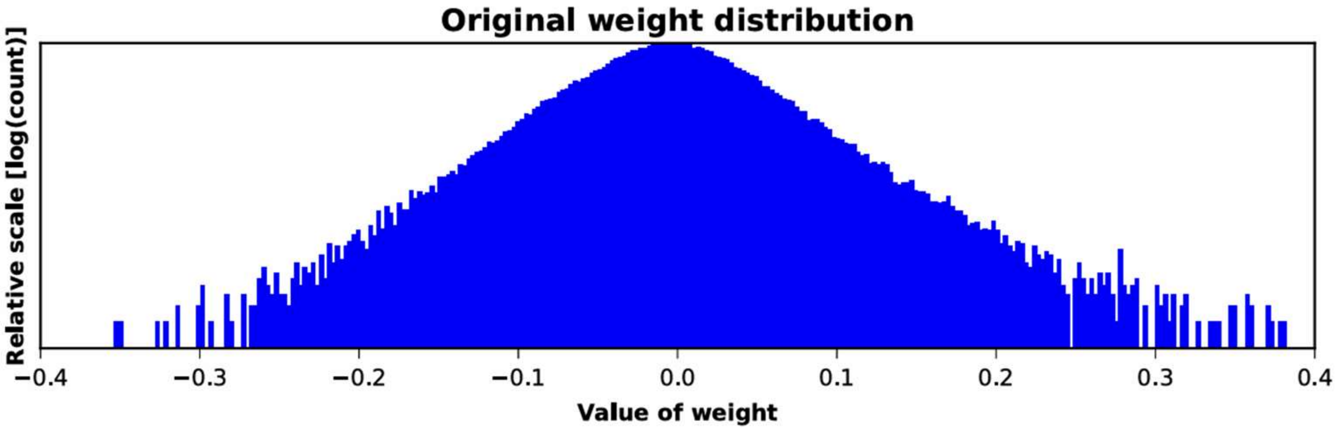


# Quantization Basics – Scheme

- **Uniform Quantization**
- **Non-uniform Quantization**

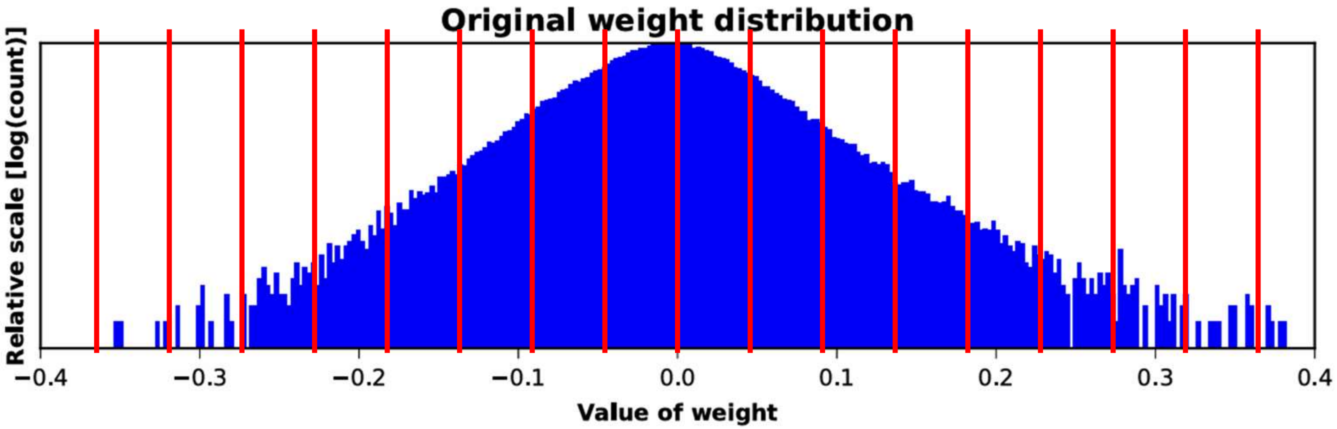
# Quantization Basics – Scheme

- **Uniform Quantization**



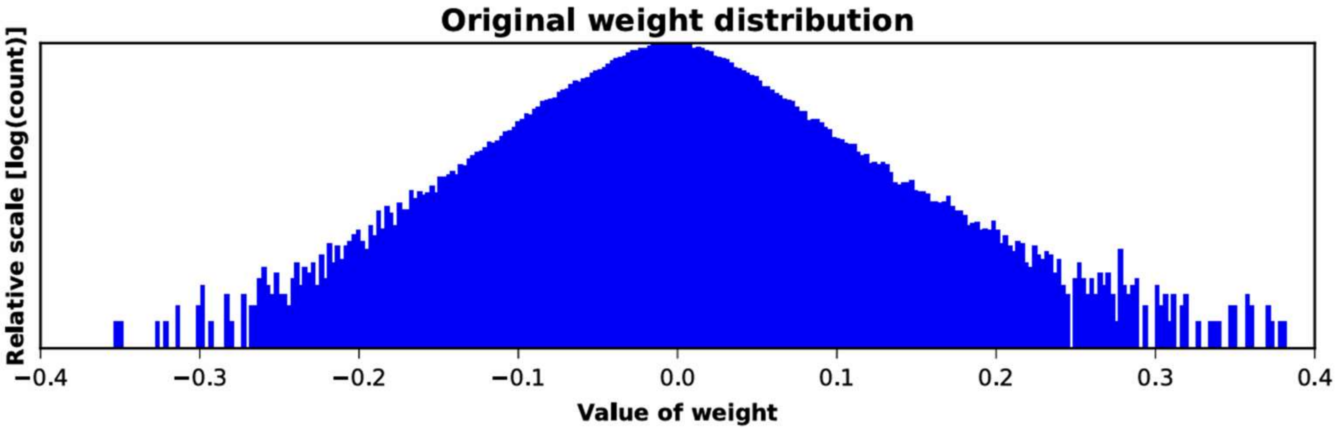
# Quantization Basics – Scheme

- **Uniform Quantization**



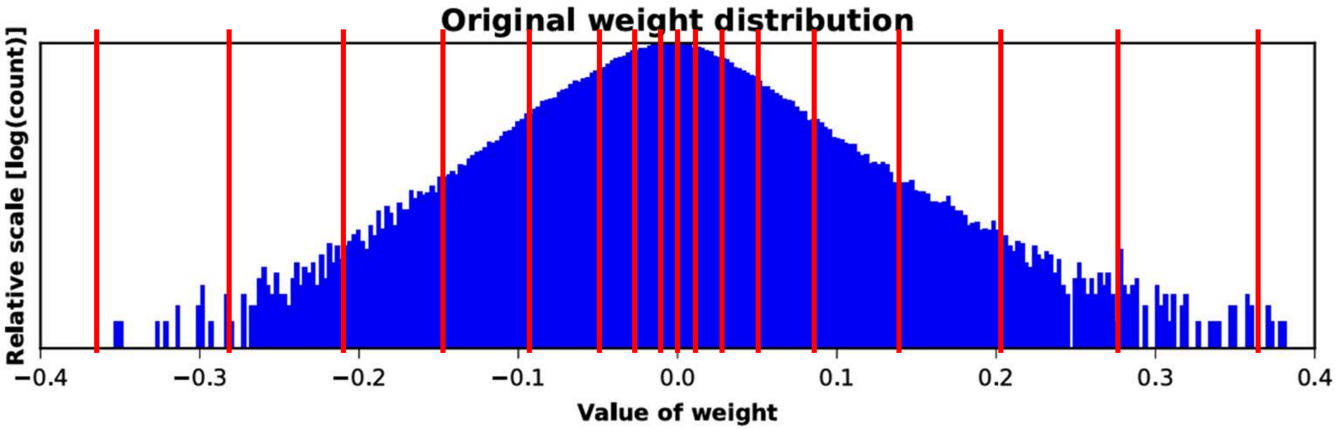
# Quantization Basics – Scheme

- **Non-uniform Quantization**



# Quantization Basics – Scheme

- **Non-uniform Quantization**

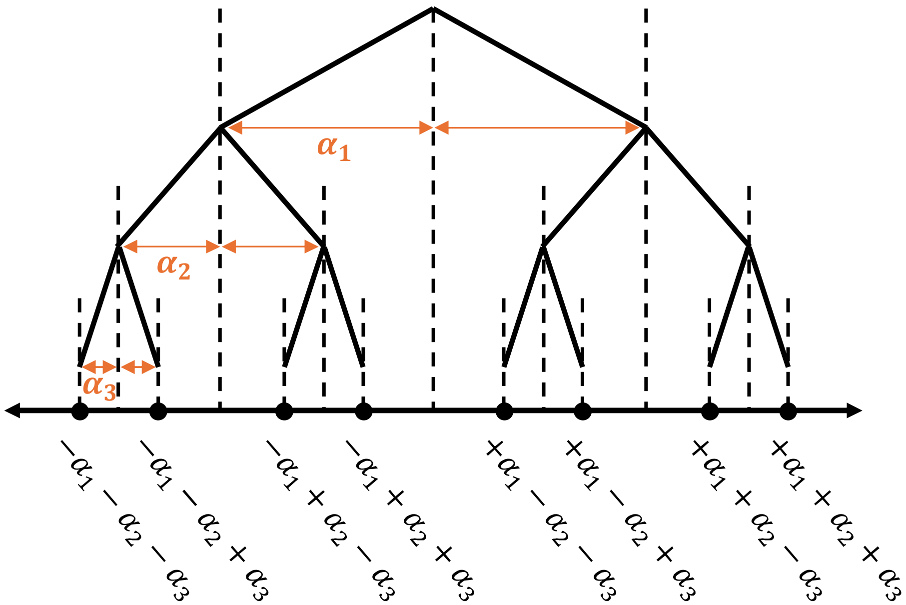



**Logarithmic Quantization (a.k.a Weighted Quantization)**

# Quantization Basics – Scheme

- **Non-uniform Quantization**

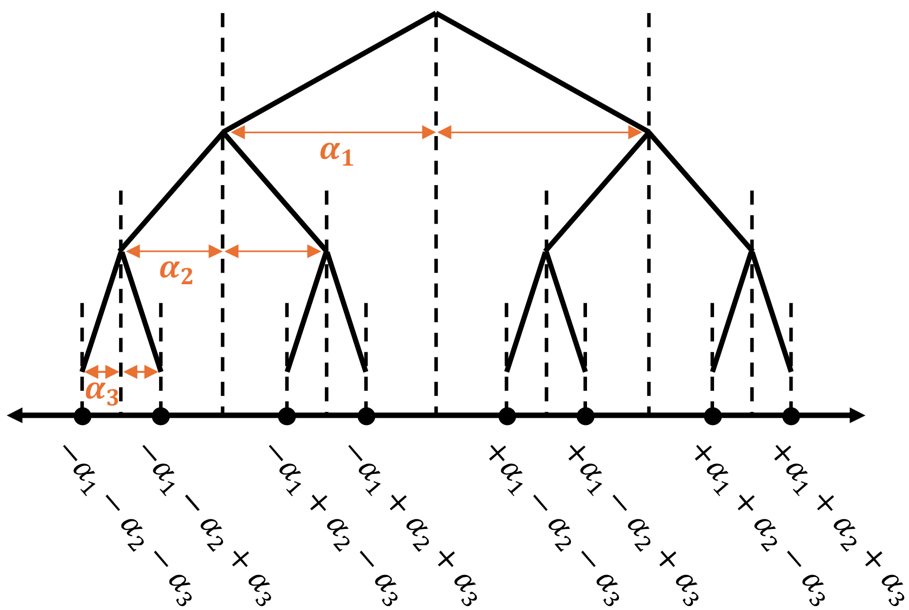
**Binary-Code Quantization**



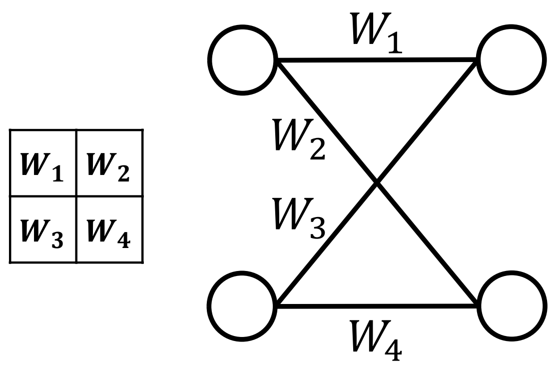
# Quantization Basics – Scheme

- **Non-uniform Quantization**

**Binary-Code Quantization**



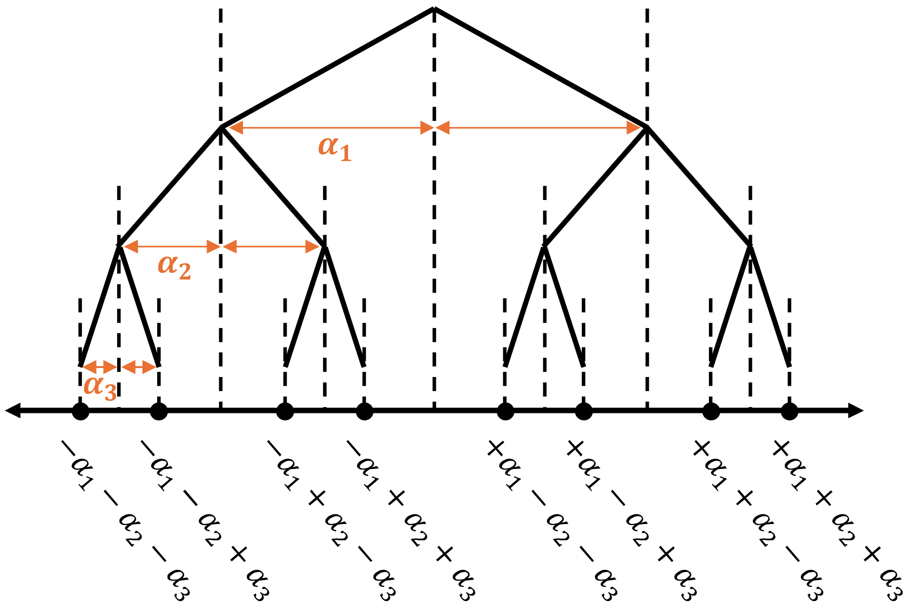

**Example of quantizing 2x2 weight**



# Quantization Basics – Scheme

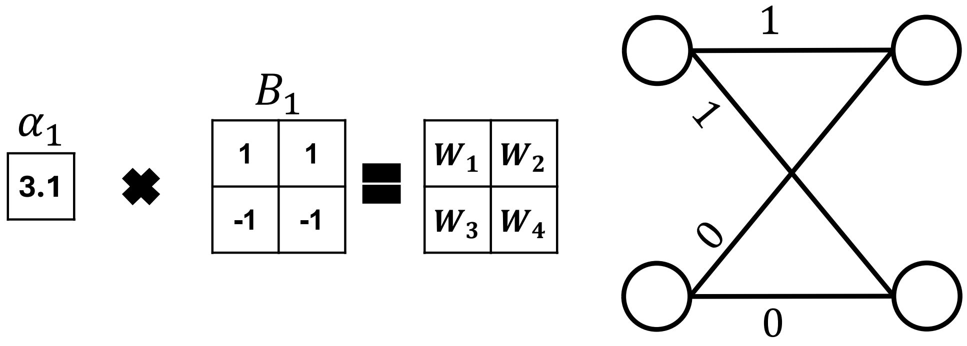
- **Non-uniform Quantization**

**Binary-Code Quantization**




**Example of quantizing 2x2 weight**

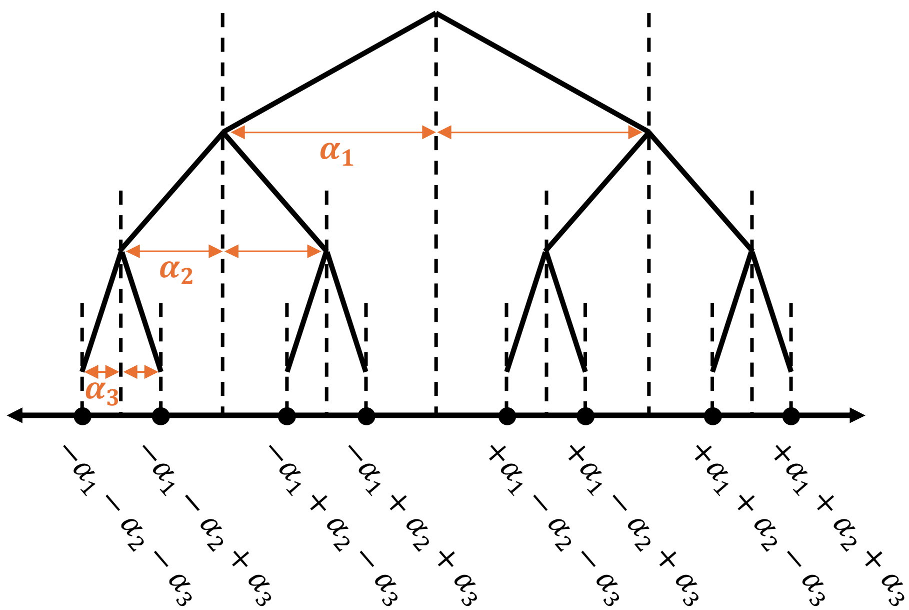
1-bit quantization



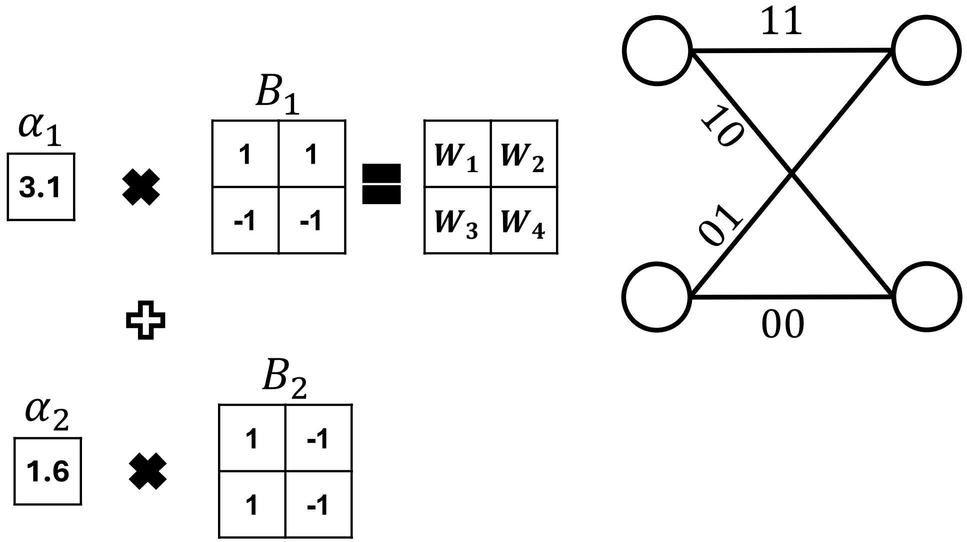
# Quantization Basics – Scheme

- **Non-uniform Quantization**

**Binary-Code Quantization**

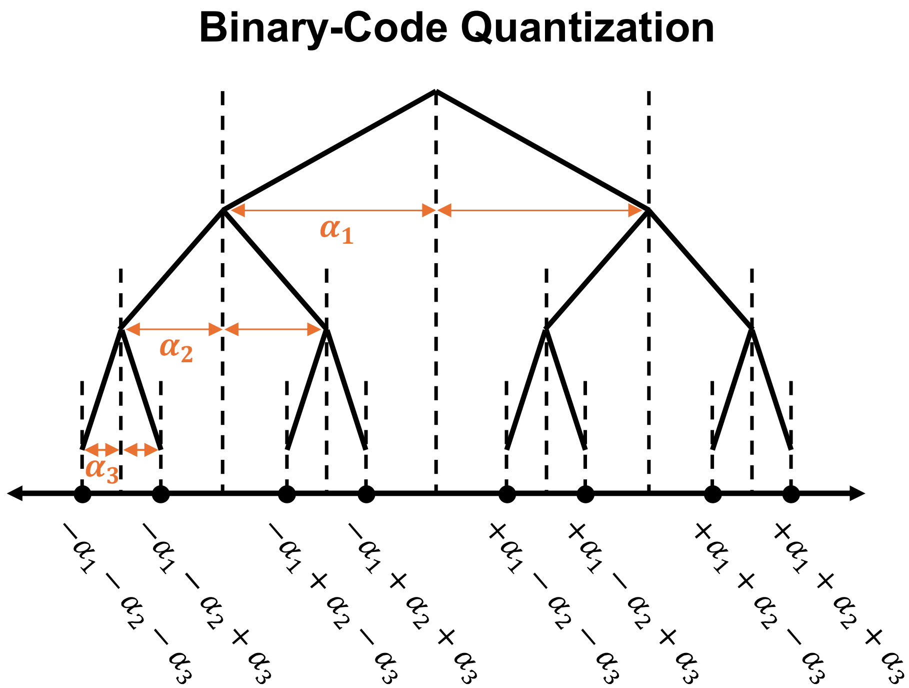


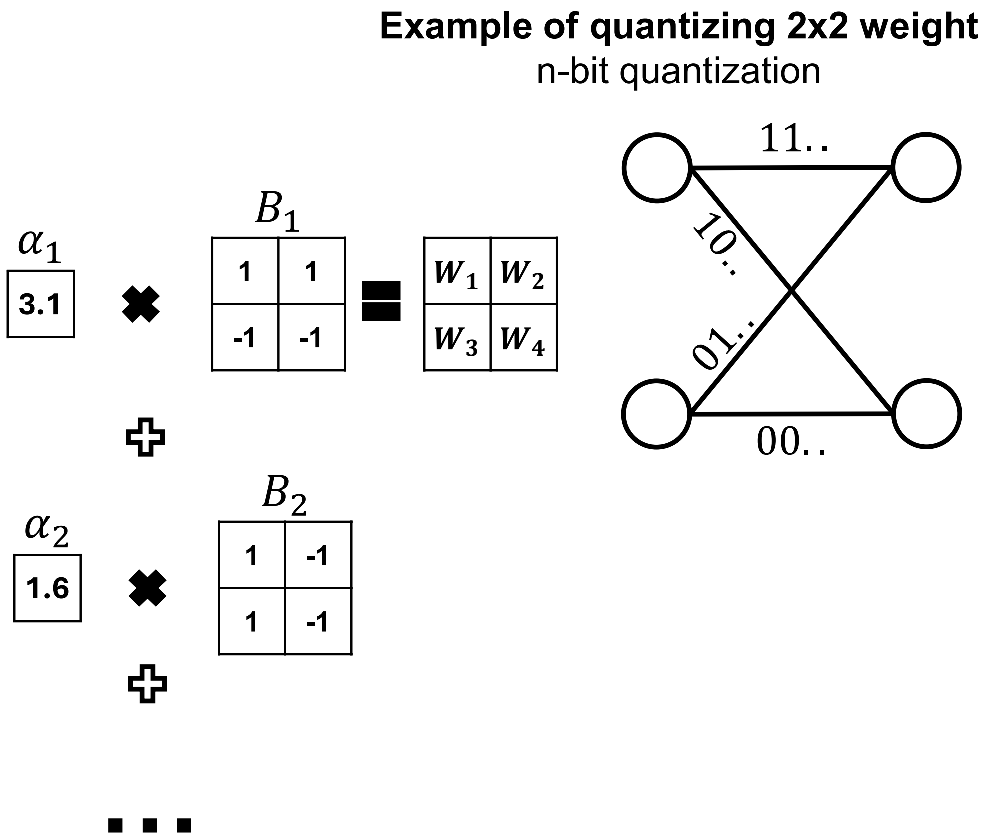

**Example of quantizing 2x2 weight**
2-bit quantization



# Quantization Basics – Scheme

- **Non-uniform Quantization**

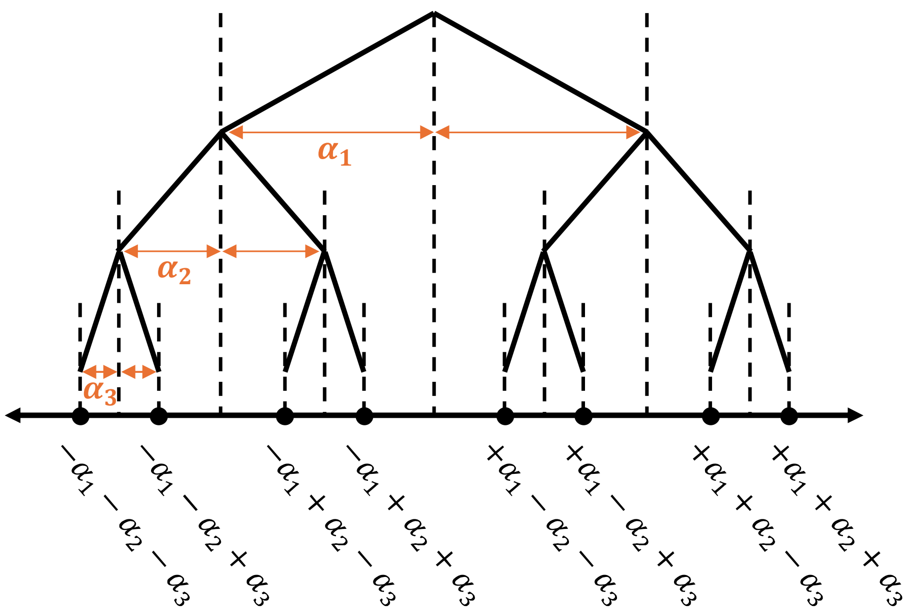


# Quantization Basics – Scheme

- **Non-uniform Quantization**

**Binary-Code Quantization**




**Very hard to optimize this method**

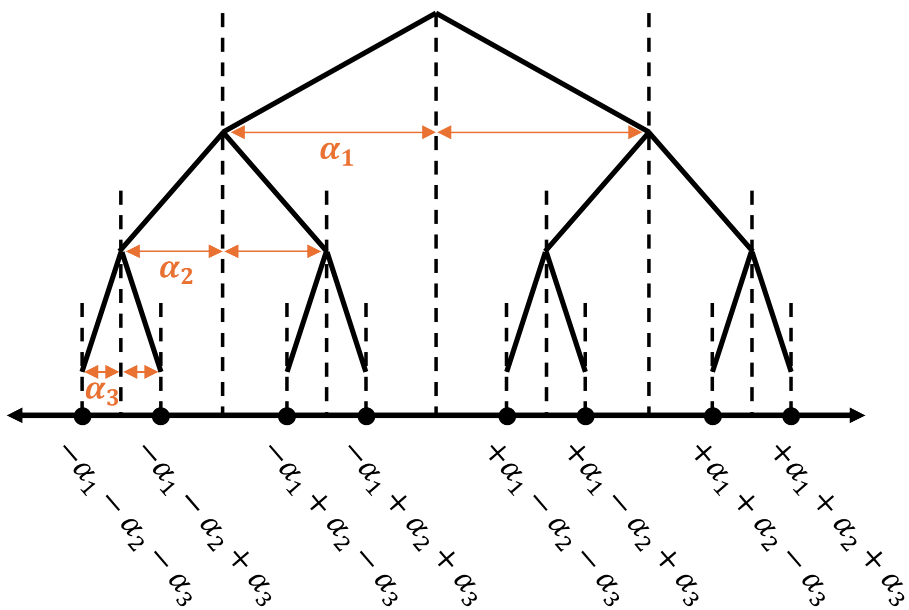
1. Reconstruction is expensive
   - a lot of elementwise operations

2. Need special optimization for integer MM
   - BiQGEMM
   - LUT-GEMM

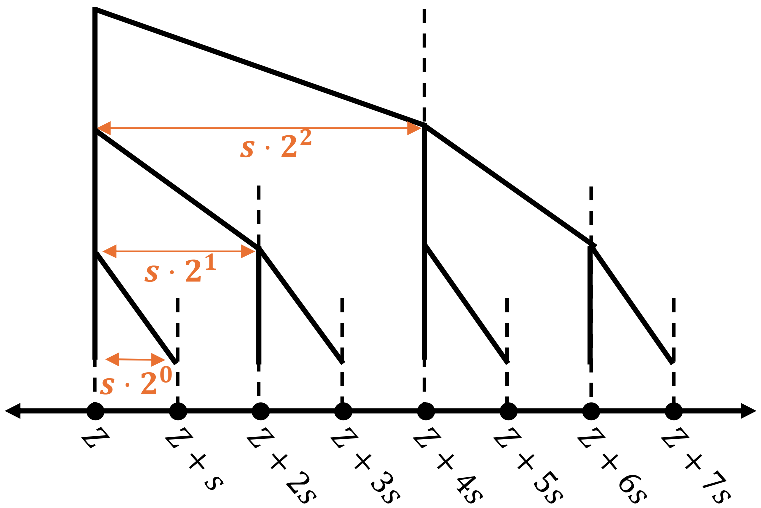
# Quantization Basics – Scheme

- **Non-uniform Quantization**

**Binary-Code Quantization**

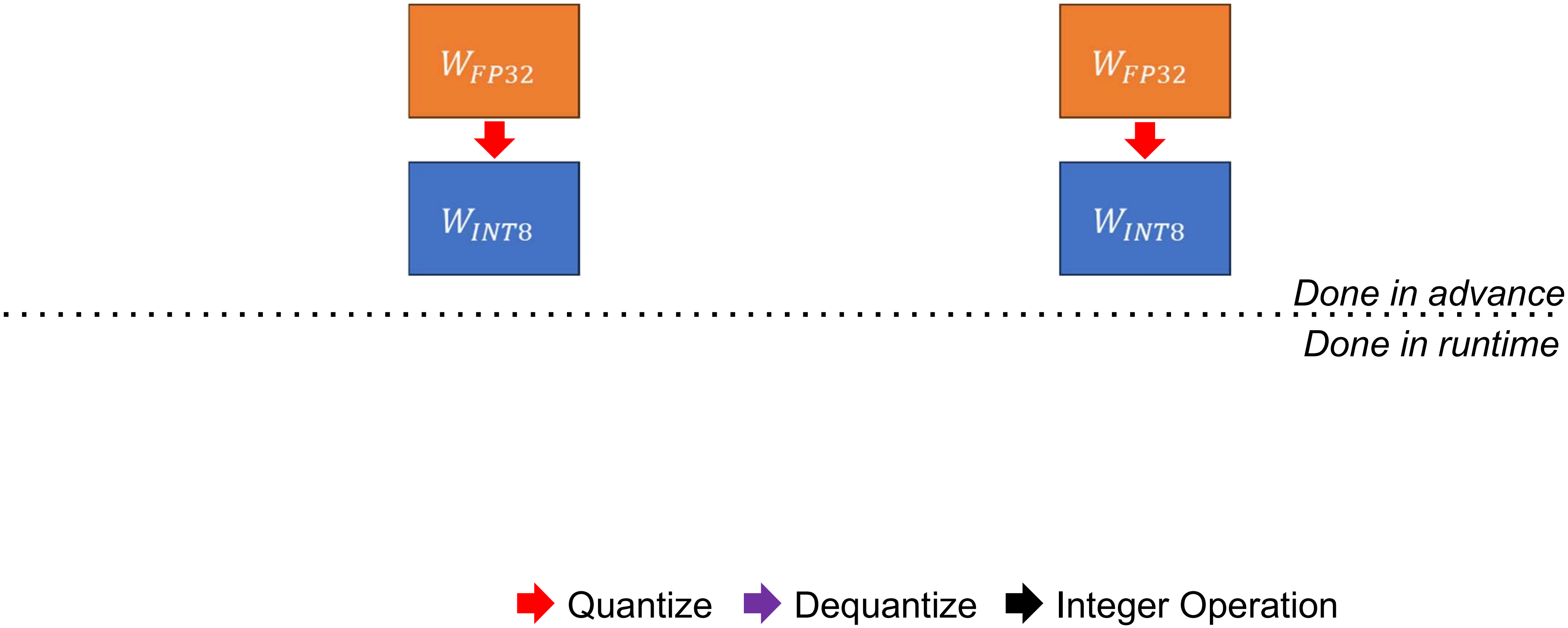



**Uniform Quantization**



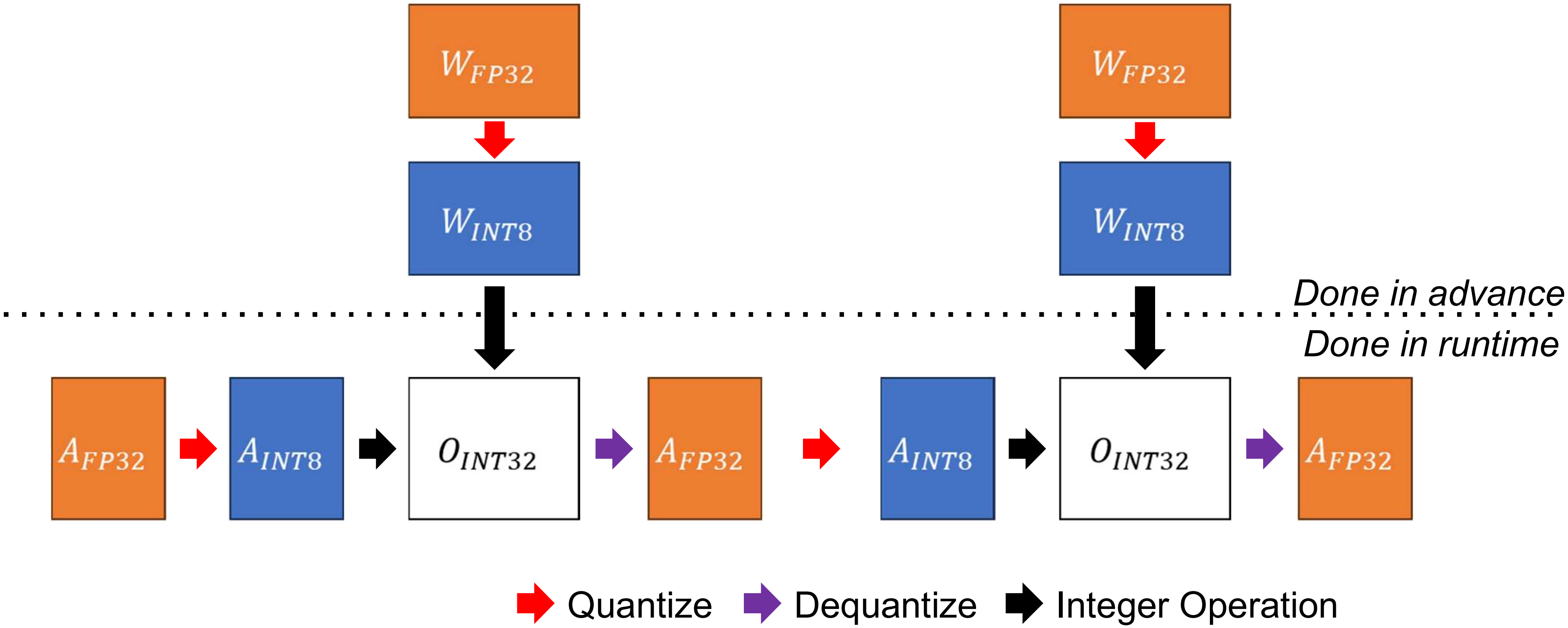
# Quantization Basics – Type

- **Dynamic Quantization**



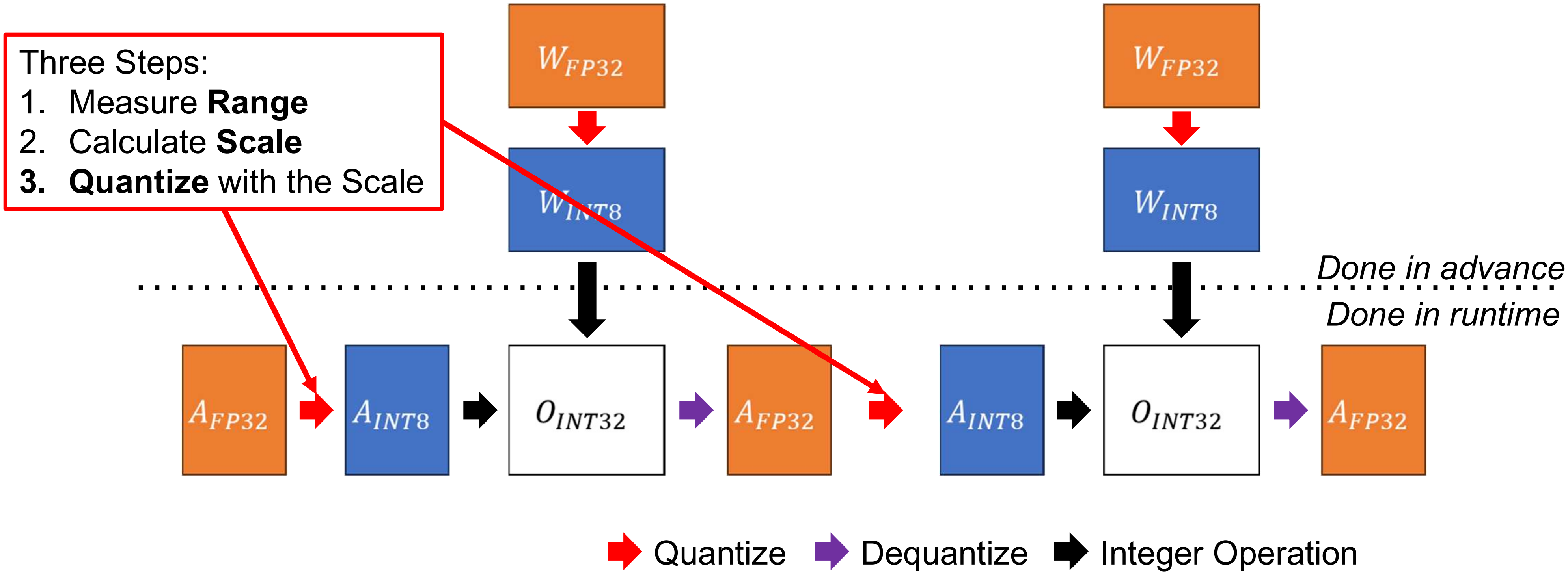
# Quantization Basics – Type

- **Dynamic Quantization**



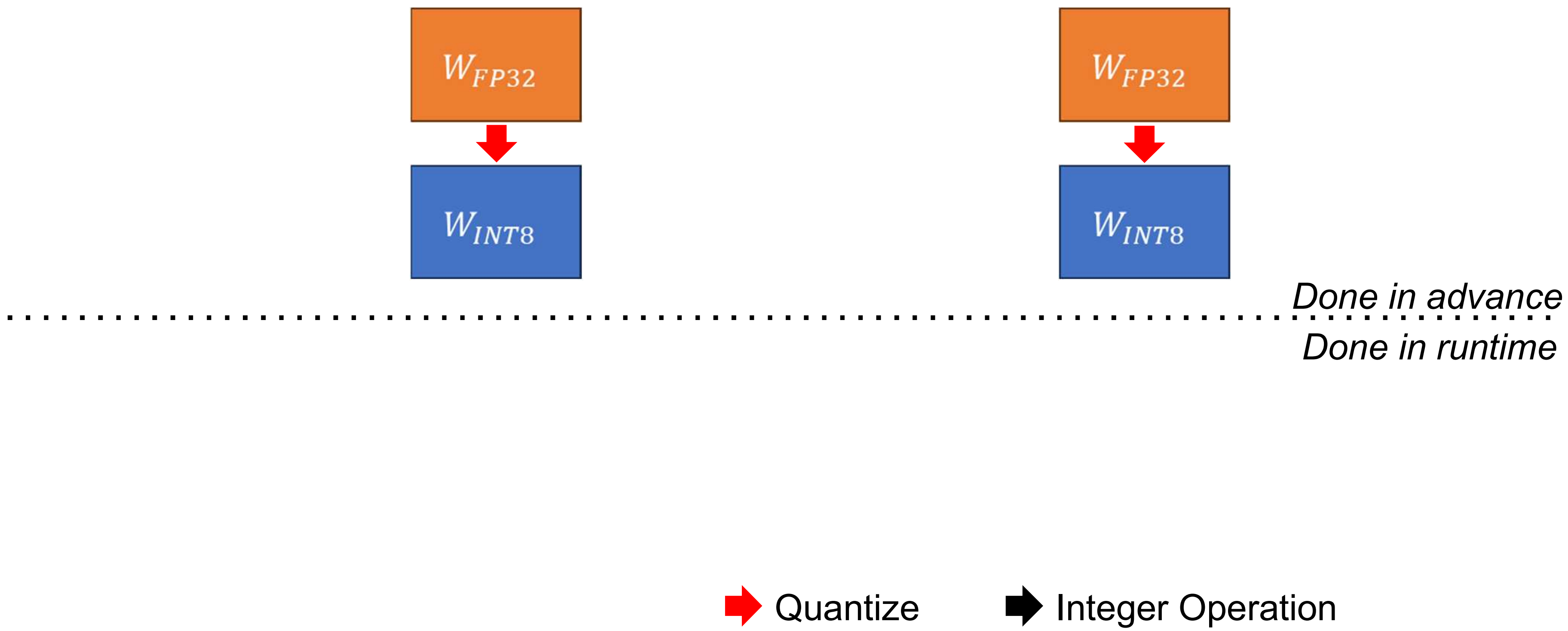
# Quantization Basics – Type

- **Dynamic Quantization**



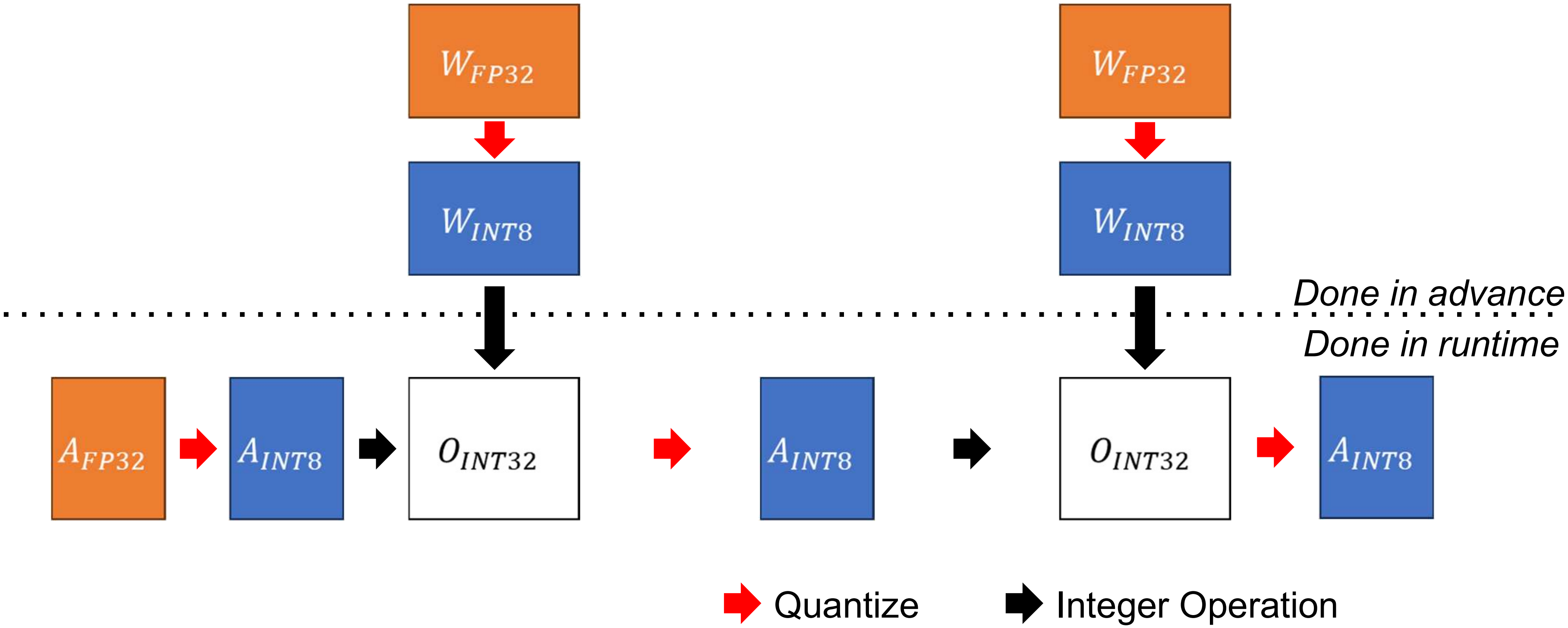
# Quantization Basics – Type

- **Static Quantization**



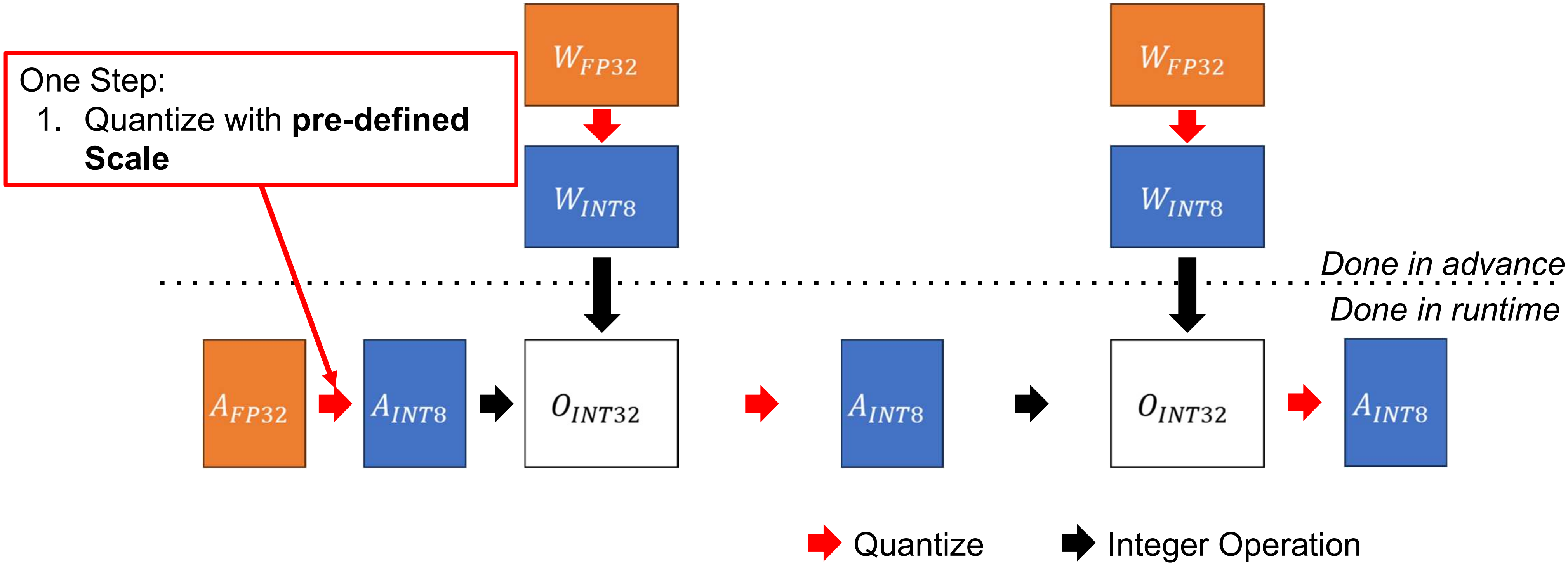
# Quantization Basics – Type

- **Static Quantization**



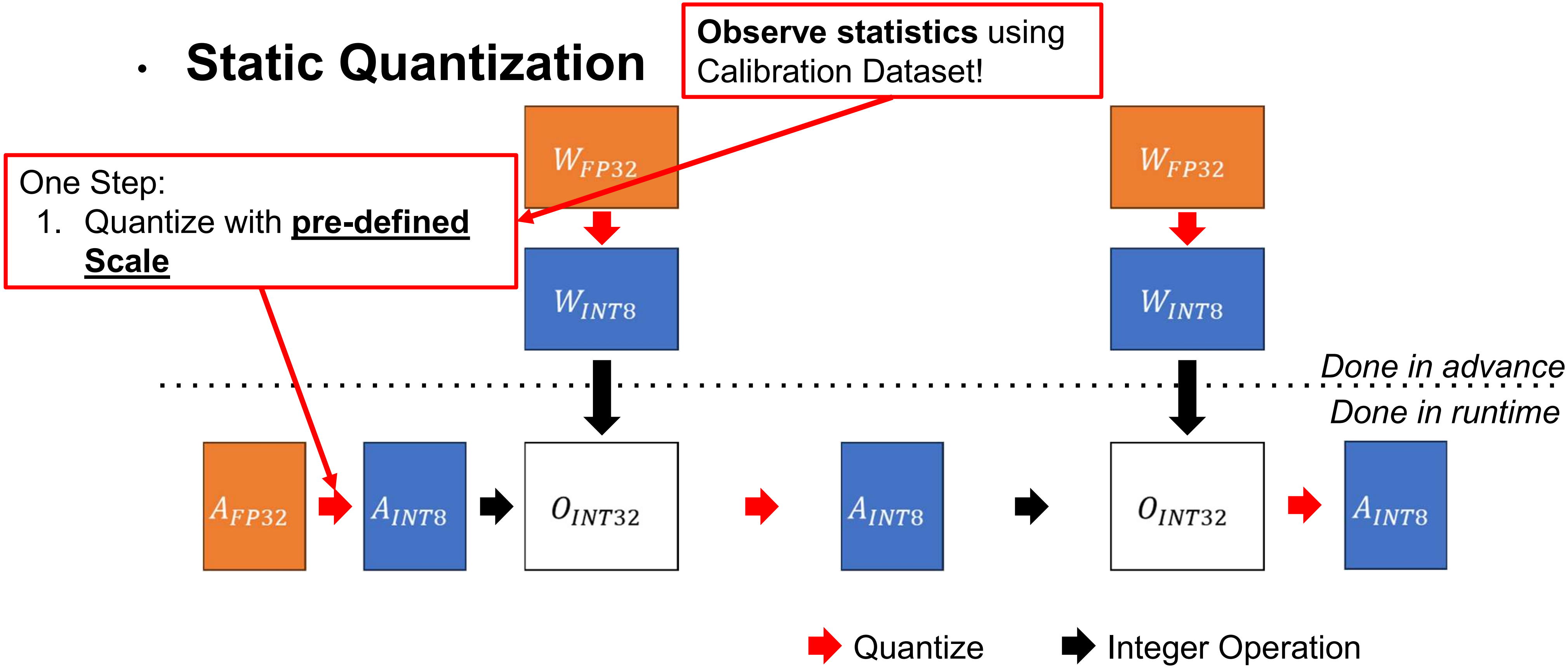
# Quantization Basics – Type

- **Static Quantization**



# Quantization Basics – Type

- **Static Quantization**



# Quantization Basics – Type

- **Static Quantization**
  - Post Training Quantization (PTQ)
  - Quantization Aware Training (QAT)

# Quantization Basics – Type

- **Static Quantization**
  - Post Training Quantization (PTQ)
    - Quantize model **after training**
    - Calibrate range statistics with **only a small subset of data**
  - Quantization Aware Training (QAT)

# Quantization Basics – Type

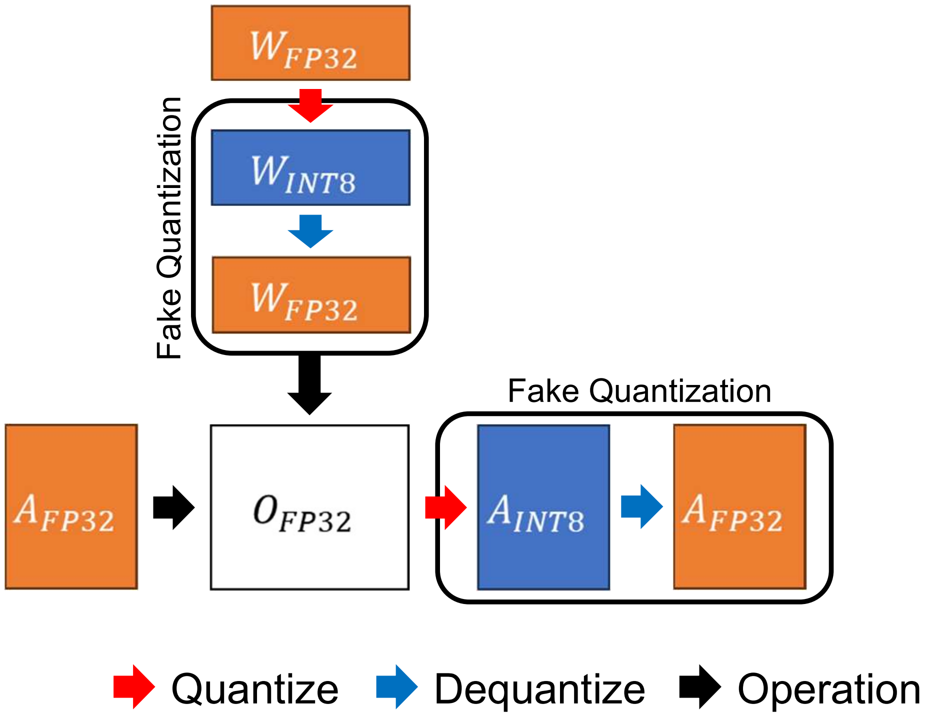
- **Static Quantization**
  - Post Training Quantization (PTQ)
  - Quantization Aware Training (QAT)
    - Quantize **with training/fine-tuning**
    - **Computation is done in FP**, but **fake quantize** to simulate integer operation

# Quantization Basics – Type

- **Static Quantization**
  - Post Training Quantization (PTQ)
  - Quantization Aware Training (QAT)
    - Quantize **with training/fine-tuning**
    - **Computation is done in FP**, but **fake quantize** to simulate integer operation
    - However, we can’t naively implement the fake quantization due to existence of operation (ex. round) that does not have a gradient function

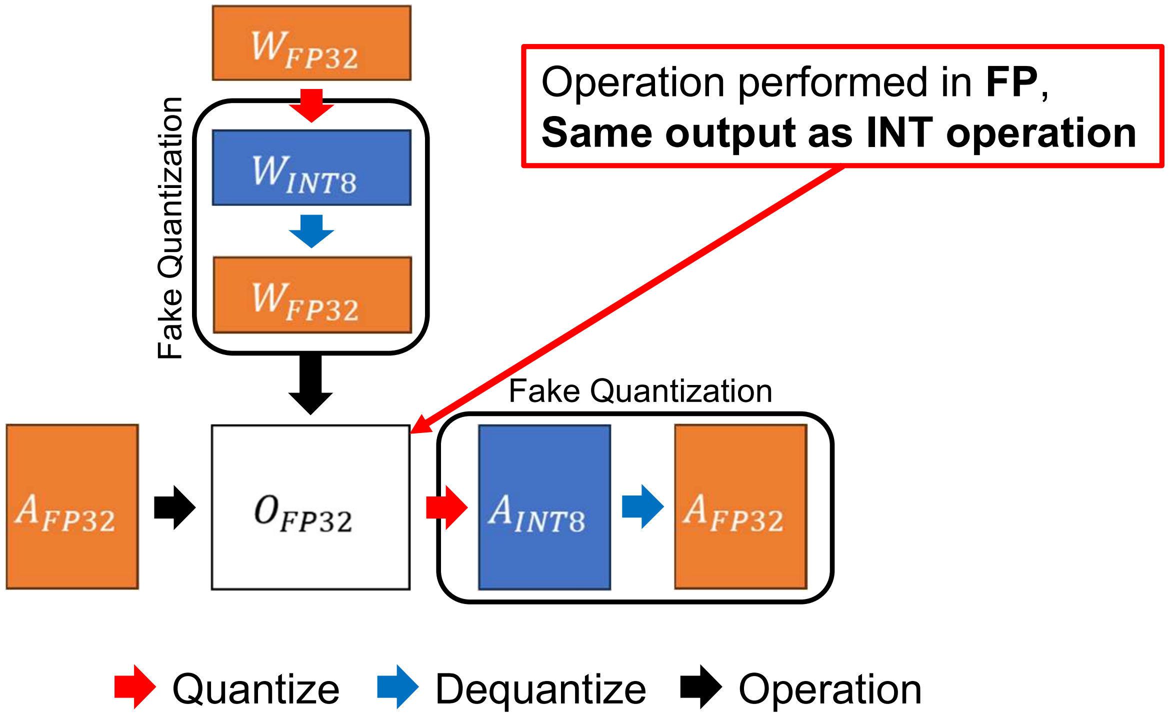
# Quantization Basics – Type

- **Quantization Aware Training :** Fake Quantization



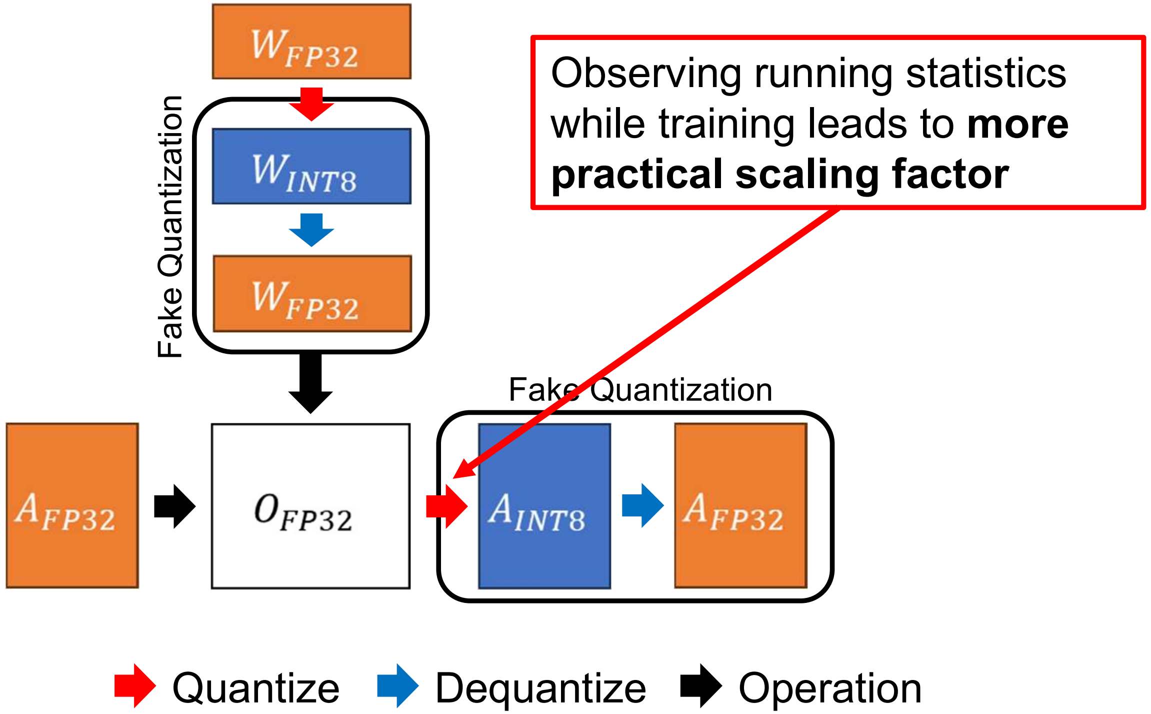
# Quantization Basics – Type

- **Quantization Aware Training :** Fake Quantization



# Quantization Basics – Type

- **Quantization Aware Training :** Fake Quantization



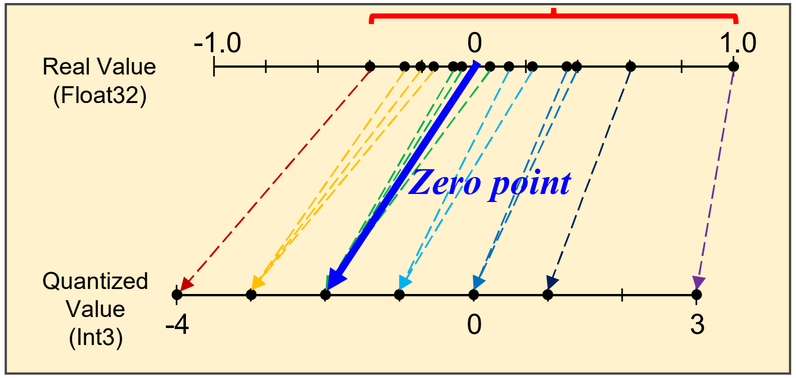
Break

# Quantization Strategies

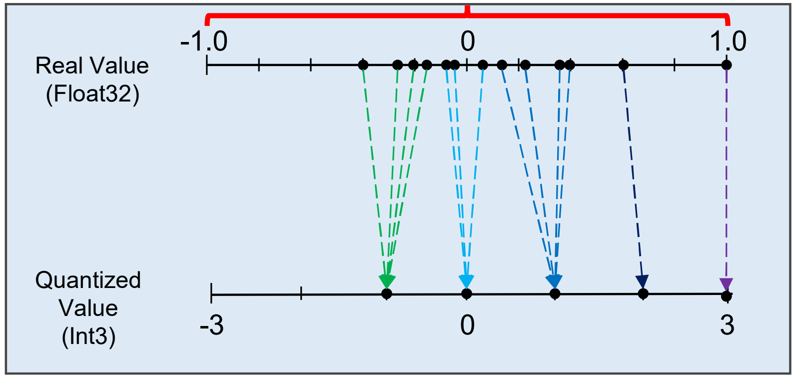
- Quantization Optimization Strategies
  - How can we **calculate** proper scaling factor?
  - What are the **common considerations** when using quantization?

# Quantization Strategies

- How can we **calculate** proper scaling factor?
  1. Simply use **Min/Max value**




**Asymmetric quantization**



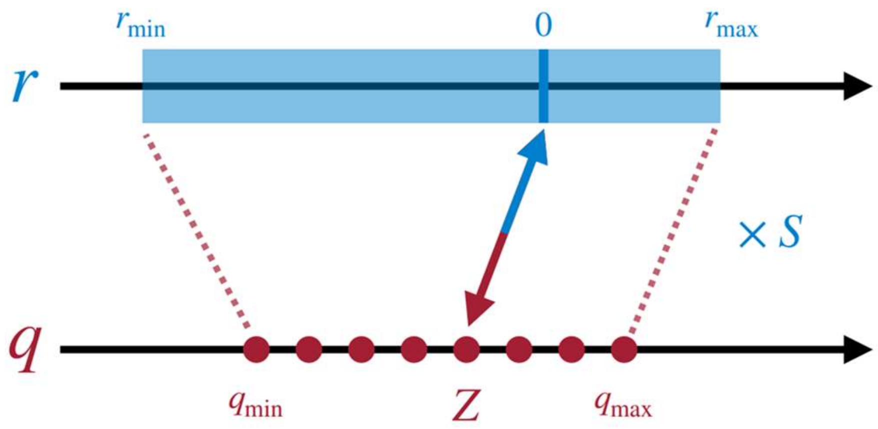

**Symmetric quantization**

# Quantization Strategies

- How can we **calculate** proper scaling factor?
  2. Use **Exponential Moving Averages (EMA)**

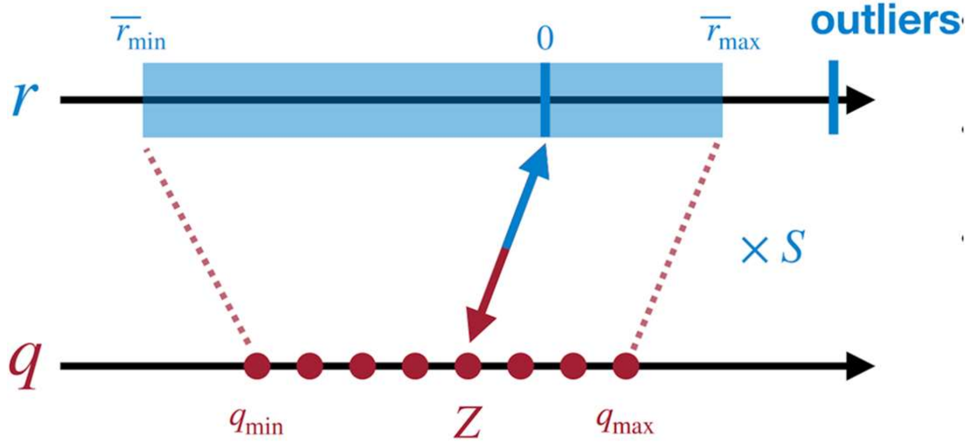
$$r_{min}^{t} = \alpha \cdot \min(x_t) + (1-\alpha) \cdot r_{min}^{t-1}$$
$$r_{max}^{t} = \alpha \cdot \max(x_t) + (1-\alpha) \cdot r_{max}^{t-1}$$

$\alpha = 0.01$



# Quantization Strategies

- How can we **calculate** proper scaling factor?
  3. Remove **Outlier**
     - Observe statistics from **calibration samples** and **clamp the range** properly.

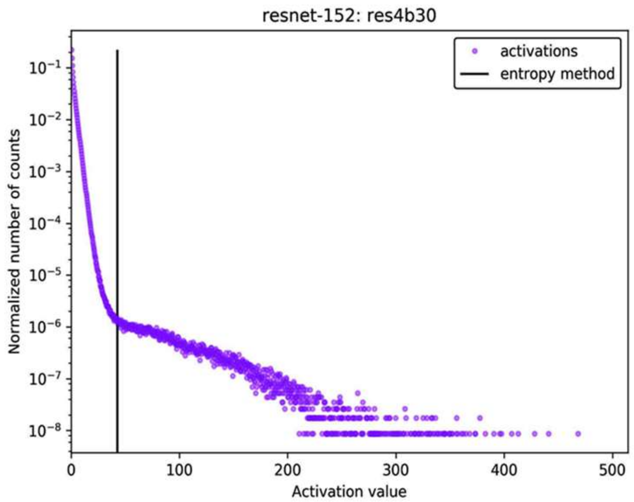


# Quantization Strategies

- How can we **calculate** proper scaling factor?

  3-1. **Entropy Calibration**

  - find proper clamping value which **minimize KL-divergence score**.

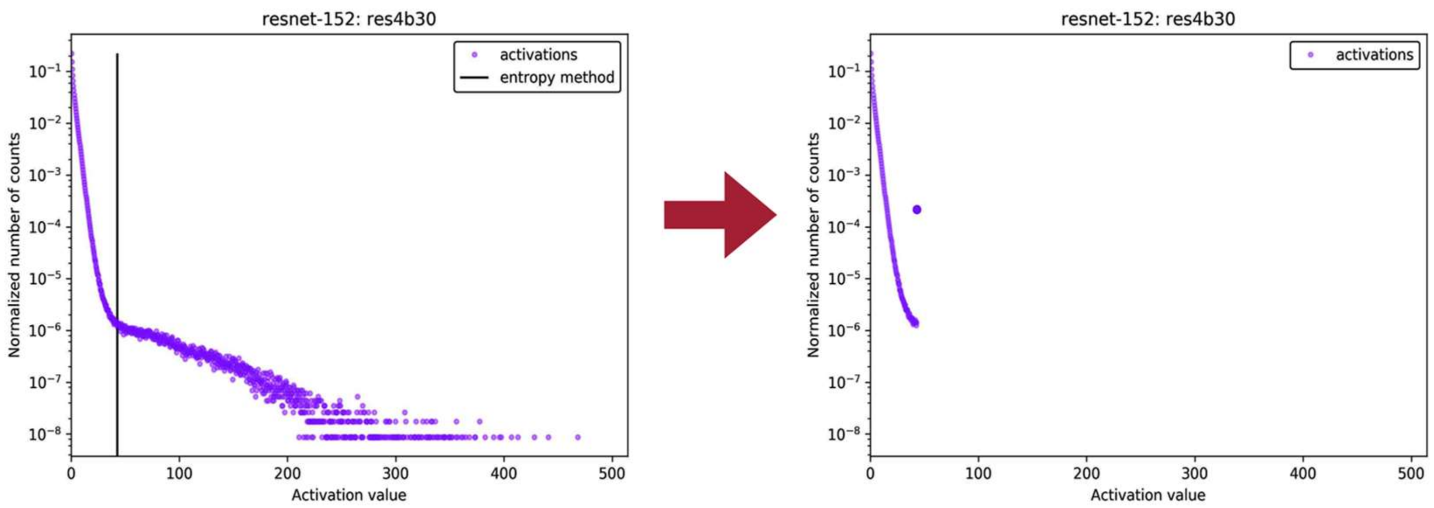


# Quantization Strategies

- How can we **calculate** proper scaling factor?

  3-1. **Entropy Calibration**

  - find proper clamping value which **minimize KL-divergence score**.

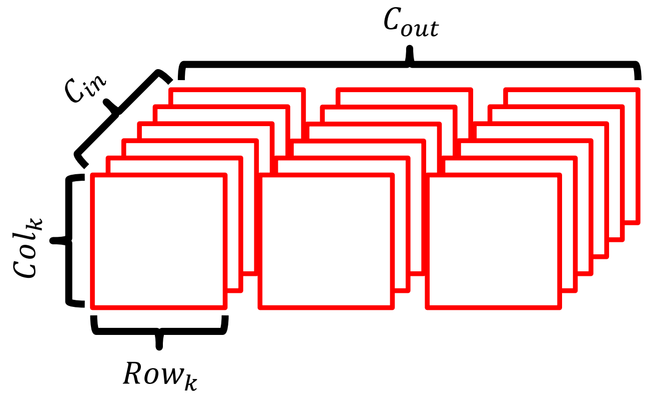


# Quantization Strategies

- How can we **calculate** proper scaling factor?

  4. **Quantization Granularity**

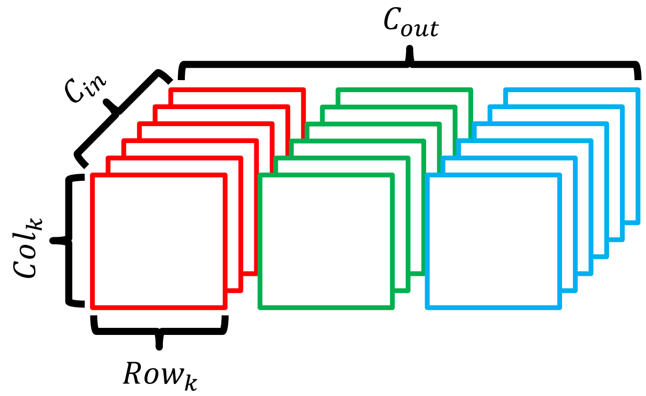
     - **Layer-wise** Quantization
       - Calculate scaling factor **per-tensor**




**Convolution Weight filters**

# Quantization Strategies

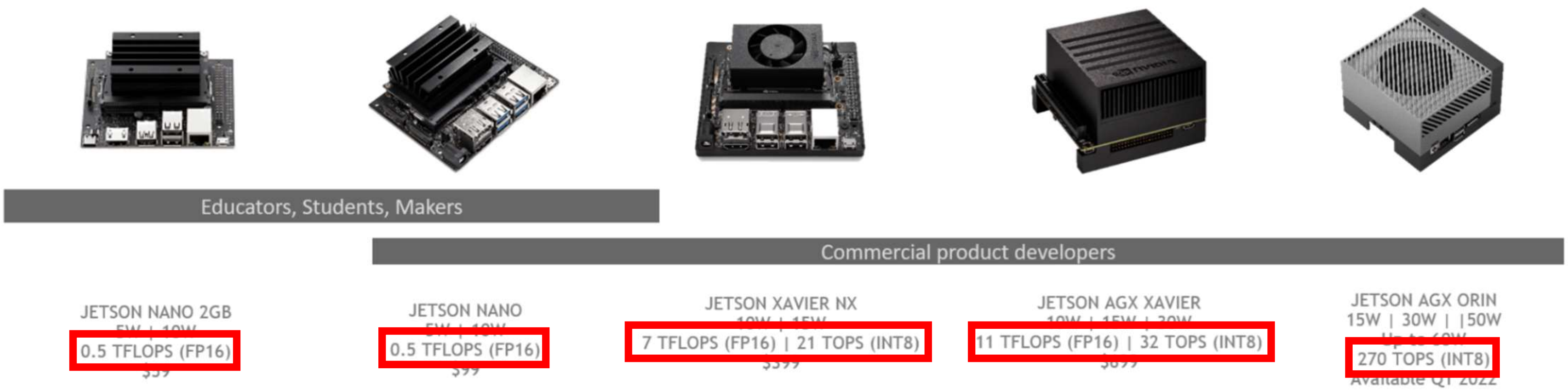
- How can we **calculate** proper scaling factor?
  - **Quantization Granularity**
    - **Layer-wise** Quantization
      - Calculate scaling factor **per-tensor**
    - **Channel-wise** Quantization
      - Calculate scaling factor **per-channel**




**Convolution Weight filters**

# Quantization Strategies

- **Guideline for Quantization**
  - Check the precision supported by hardware acceleration operations

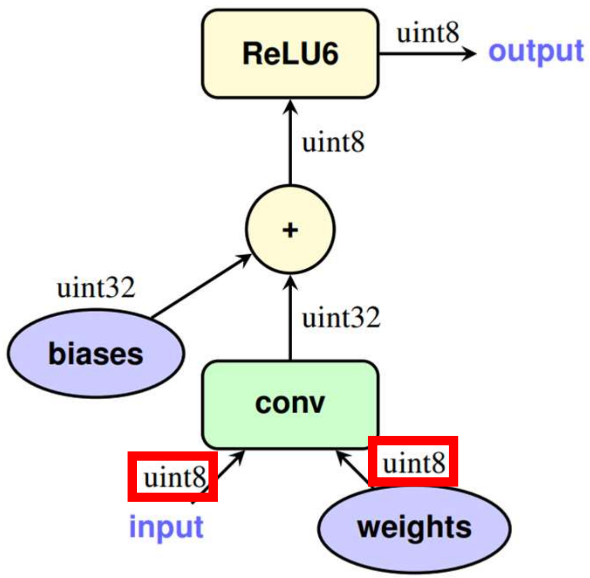


# Quantization Strategies

- **Guideline for Quantization**
  - Quantizing all layers may not be optimal
    - It is necessary to consider the trade-off between **computational cost** and **accuracy loss** of the model.
    - Example : Compute amount only 0.1% of total inference; Accuracy drop significant (Depthwise-Convolution in MobileNetV3)

# Quantization Strategies

- **Guideline for Quantization**
  - Input and weight must have the **same precision** for correct computation



# Quantization Strategies

- **Guideline for Quantization**
  - Specific operations require special handling
    - **Non-Linear Activation** (GeLU, Softmax, etc.): FP16
    - **Normalization Layer**: Minimum 16-bit or FP16 or BN Folding
    - **Residual Connection**: Match scaling factor or use INT16, FP16

# Quantization Strategies

- **Guideline for Quantization**
  - Challenges of Extremely Low-Bitwidth Quantization
    - Changes in Output Distribution (Norm-layer statistics become inaccurate)
    - Necessity of Quantization Aware Training (QAT)

# Break