



<https://haoailab.com/cse291-s26/>

CSE/DSC 291: Deep Learning Systems Spring 2026

LLM, diffusion, and case studies

Optimizations and Parallelization

Basics

Let's Move On

- Memory and Scheduling
 - Checkpointing and rematerialization
 - Swapping
- Memory and Compute
 - **Quantization**
 - Mixed precision

Dataflow Graph

Autodiff

Graph Optimization

Parallelization

Runtime

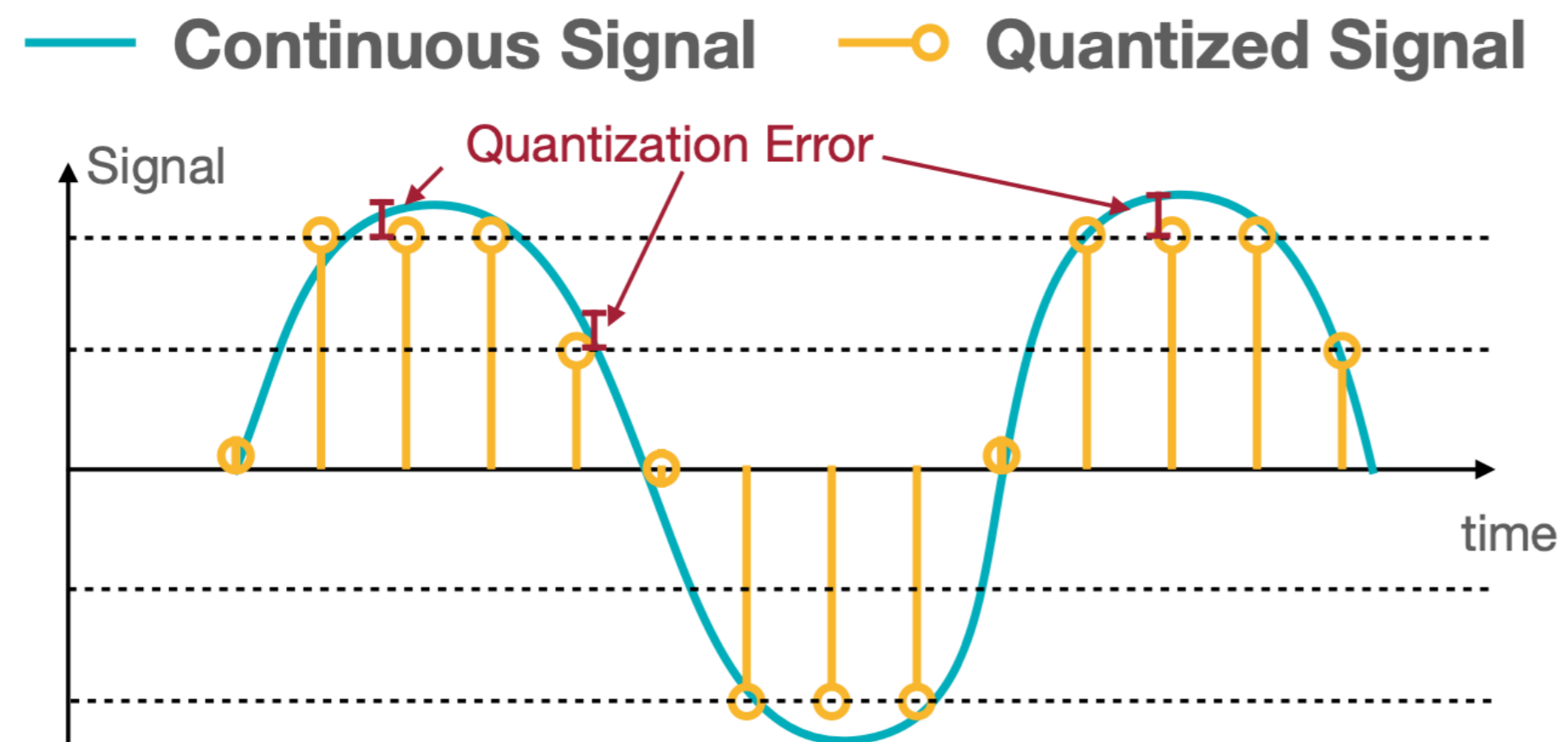
Operator

Reduce Memory of Parameters: Quantization

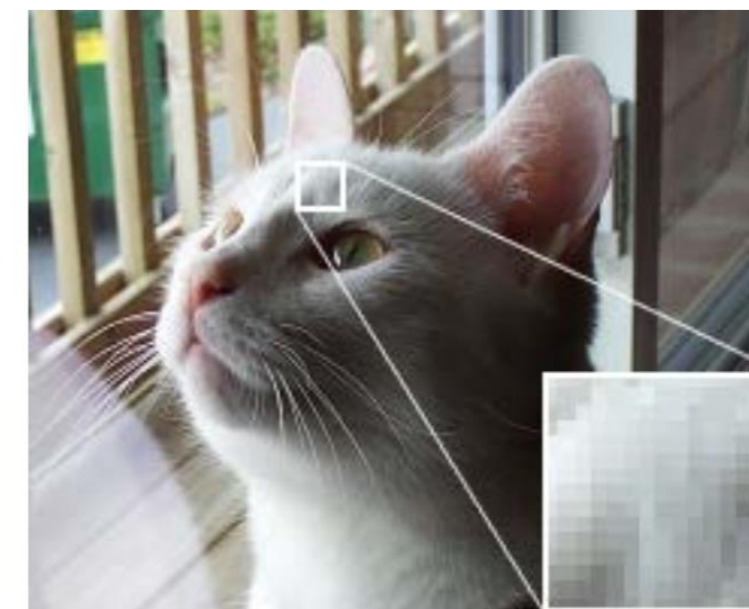
- Observation: all our memory usage (weights, activation, optimizer states) is a multiple of `sizeof(element)`
- 💡 Can we reduce `sizeof(element)`?

What is Quantization

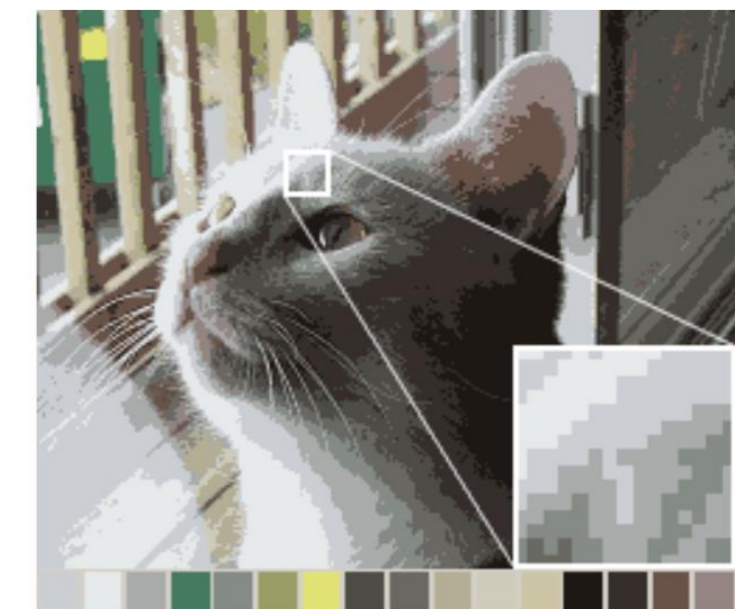
Quantization is the process of constraining an input from a continuous or otherwise large set of values to a discrete set.



Original Image



16-Color Image



Images are in the public domain.

“Palettization”

Quantization in ML

High-level Idea: Use a lower-precision representation for data, while:

- (Almost) preserve ML performance, e.g., accuracy
- Accelerate compute
- Reduce memory
- Save energy
- etc.

Quantization

- Digital representation of data
- Basics of quantization
- Quantization in ML
- Mixed precision training

Dataflow Graph

Autodiff

Graph Optimization

Parallelization

Runtime

Operator

Representation of Data: Integer

Unsigned Integer

- n-bit range: $[0, 2^n - 1]$

0	0	1	1	0	0	0	1
x	x	x	x	x	x	x	x

$$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 49$$

Signed Integer

- Sign-Magnitude Representation
- n-bit range: $[-2^{n-1} - 1, 2^{n-1} - 1]$
- Problem: Both 000...00 and 100...00 represent 0

Sign Bit

1	0	1	1	0	0	0	1
	x	x	x	x	x	x	x

$$- 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = -49$$

Two's complement representation

- n-bit range: $[-2^{n-1} - 1, 2^{n-1} - 1]$

1	1	0	0	1	1	1	1
x	x	x	x	x	x	x	x

$$-2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = -49$$

Fixed-point Number



Integer . Fraction

“Decimal” Point



x x x x x x x x

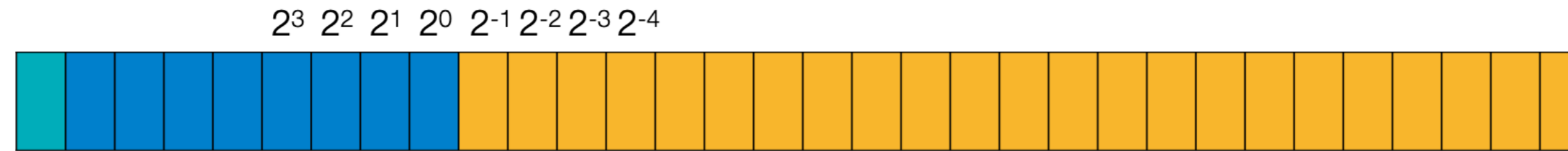
$$-2^3 + 2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} = 3.0625$$



x x x x x x x x

$$(-2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0) \times 2^{-4} = 49 \times 0.0625 = 3.0625$$

Floating-point Representation



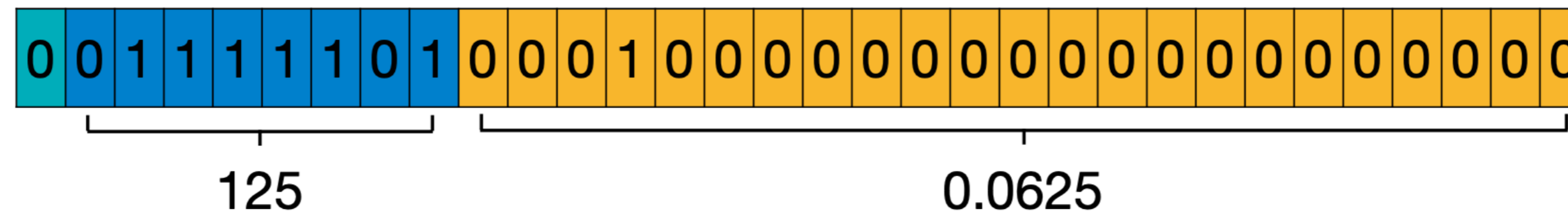
Sign 8 bit Exponent

23 bit Fraction

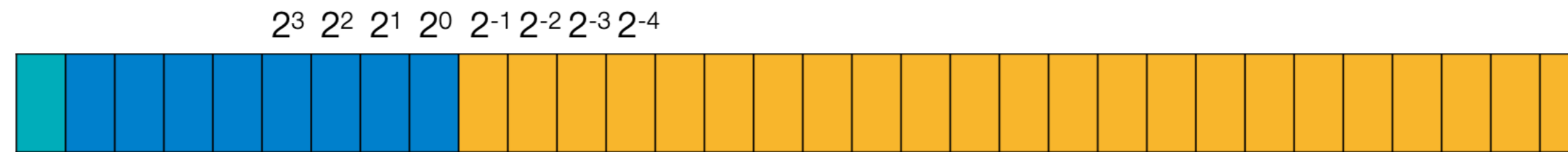
$$(-1)^{\text{sign}} \times (1 + \text{Fraction}) \times 2^{\text{Exponent}-127} \quad \leftarrow \quad \text{Exponent Bias} = 127 = 2^{8-1}-1$$

(significant / mantissa)

$$0.265625 = 1.0625 \times 2^{-2} = (1 + \underline{0.0625}) \times 2^{\underline{125}-127}$$



Floating-point Representation



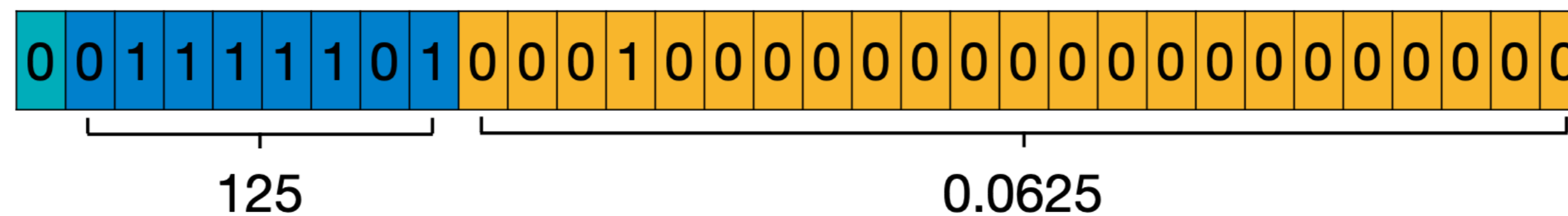
Sign 8 bit Exponent

23 bit Fraction

$$(-1)^{\text{sign}} \times (1 + \text{Fraction}) \times 2^{\text{Exponent}-127} \quad \leftarrow \quad \text{Exponent Bias} = 127 = 2^{8-1}-1$$

(significant / mantissa)

$$0.265625 = 1.0625 \times 2^{-2} = (1 + \underline{0.0625}) \times 2^{\underline{125}-127}$$



Q: How to represent 0?

Floating-point Number: normal vs. subnormal




Sign 8 bit Exponent

23 bit Fraction

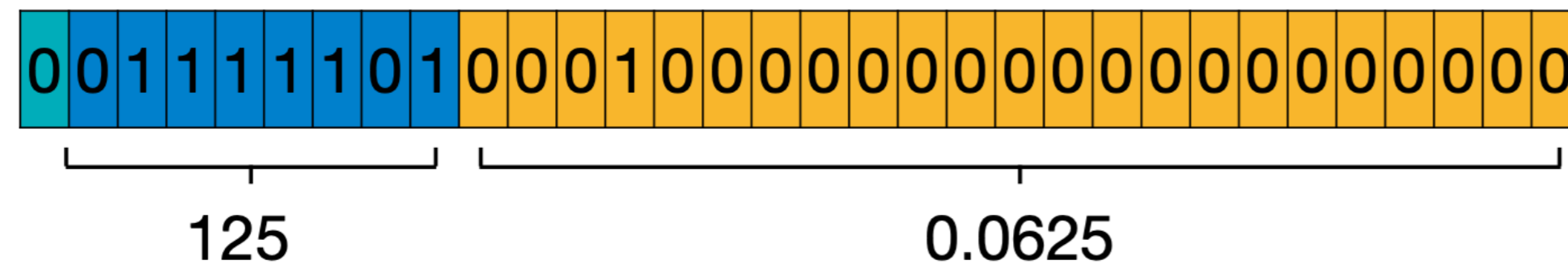
$$(-1)^{\text{sign}} \times (1 + \mathbf{\text{Fraction}}) \times 2^{\mathbf{\text{Exponent}}-127}$$

(Normal Numbers, Exponent≠0)

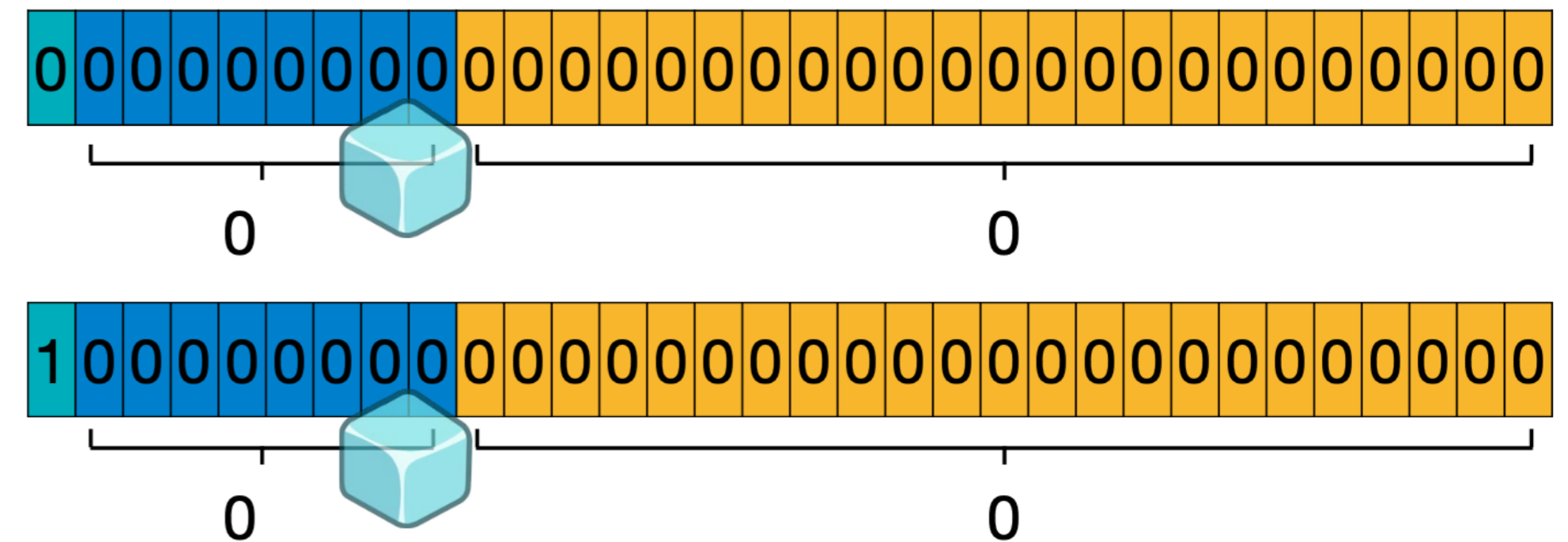
Should have been $(-1)^{\text{sign}} \times (1 + \mathbf{\text{Fraction}}) \times 2^{0-127}$

But we force to be $(-1)^{\text{sign}} \times \mathbf{\text{Fraction}} \times 2^{1-127}$ 

(Subnormal Numbers, Exponent=0)



$$0.265625 = 1.0625 \times 2^{-2} = (1 + 0.0625) \times 2^{125-127}$$



$$0 = 0 \times 2^{-126}$$

Q: What is the representation power of fp32?

What is the minimum positive value?



Sign 8 bit Exponent

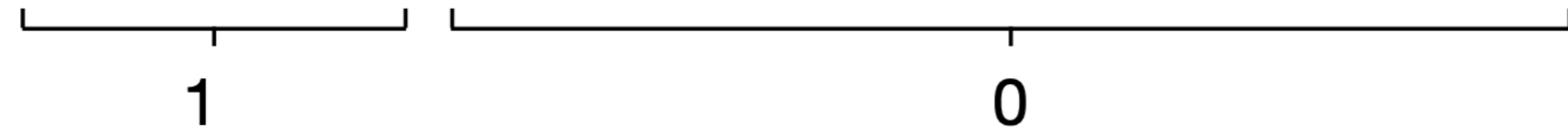
23 bit Fraction

$$(-1)^{\text{sign}} \times (1 + \mathbf{\text{Fraction}}) \times 2^{\mathbf{\text{Exponent}}-127}$$

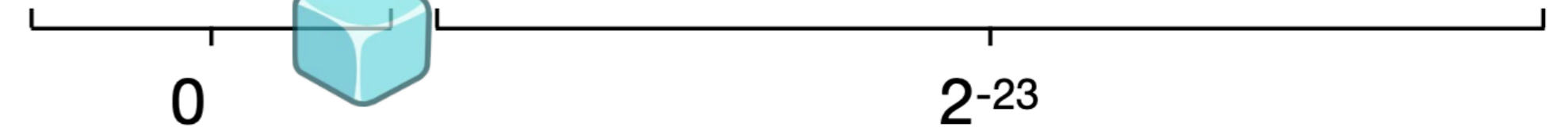
(Normal Numbers, Exponent≠0)

$$(-1)^{\text{sign}} \times \mathbf{\text{Fraction}} \times 2^{1-127}$$

(Subnormal Numbers, Exponent=0)



$$2^{-126} = (1 + \underline{0}) \times 2^{1-127}$$



$$2^{-149} = 2^{-23} \times 2^{-126}$$

Some Special Values



Sign 8 bit Exponent

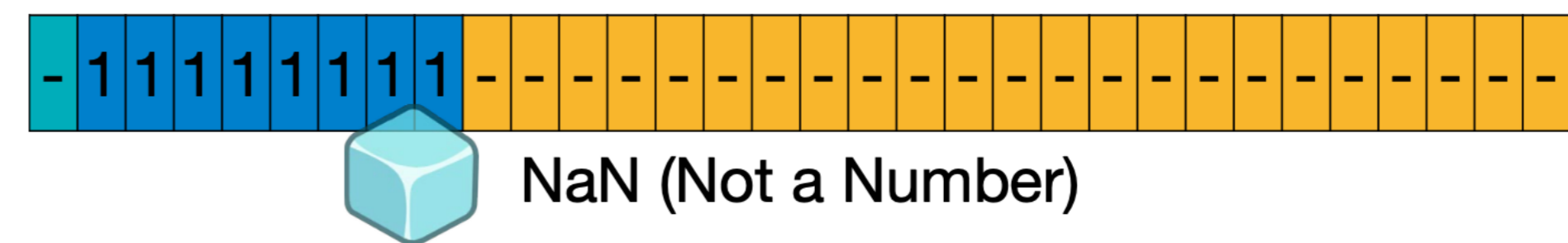
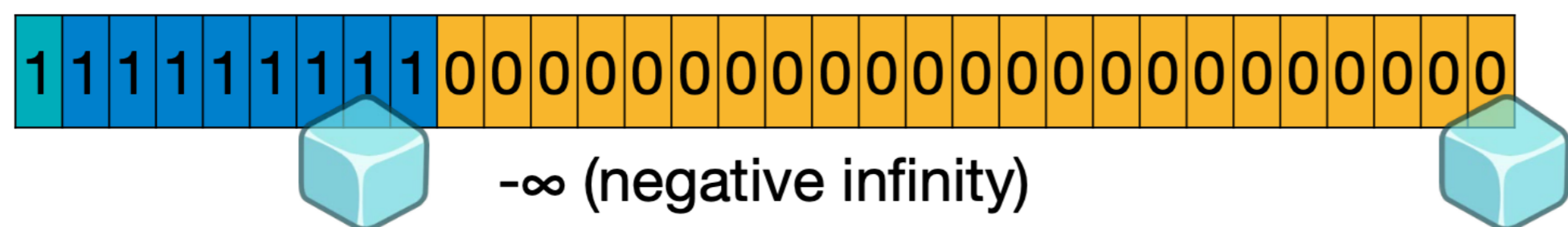
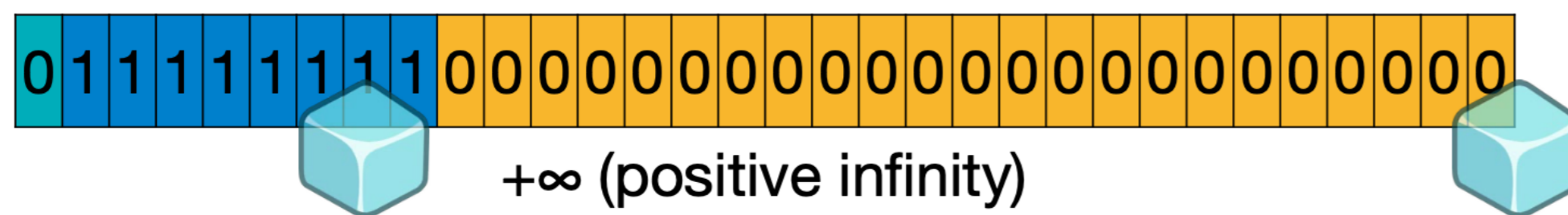
23 bit Fraction

$$(-1)^{\text{sign}} \times (1 + \mathbf{\text{Fraction}}) \times 2^{\text{Exponent}-127}$$

(Normal Numbers, Exponent \neq 0)

$$(-1)^{\text{sign}} \times \mathbf{\text{Fraction}} \times 2^{1-127}$$

(Subnormal Numbers, Exponent=0)






much waste. Revisit in fp8.

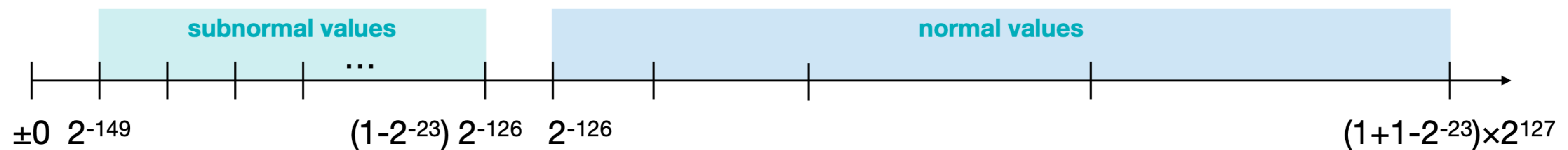
Summary of fp32



Sign 8 bit **Exponent**

23 bit Fraction

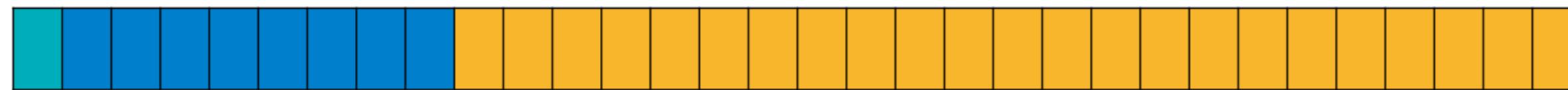
Exponent	Fraction=0	Fraction≠0	Equation
00 _H = 0 	±0	subnormal	$(-1)^{\text{sign}} \times \mathbf{Fraction} \times 2^{1-127}$
01 _H ... FE _H = 1 ... 254	normal		$(-1)^{\text{sign}} \times (1 + \mathbf{Fraction}) \times 2^{\text{Exponent}-127}$
FF _H = 255 	±INF 	NaN	



FP32 vs. FP16 vs. BF16

- Exponent width -> Range; Fraction width -> precision

[IEEE 754](#) Single Precision 32-bit Float (IEEE FP32)



Exponent
(bits)

8

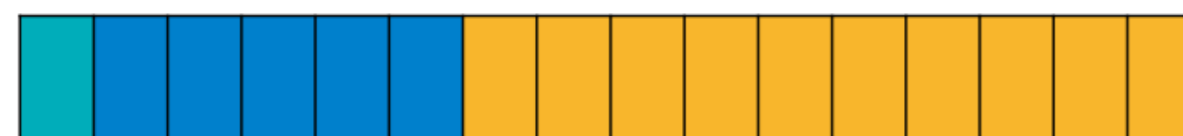
Fraction
(bits)

23

Total
(bits)

32

[IEEE 754](#) Half Precision 16-bit Float (IEEE FP16)



5

10

16

[Google](#) Brain Float (BF16)

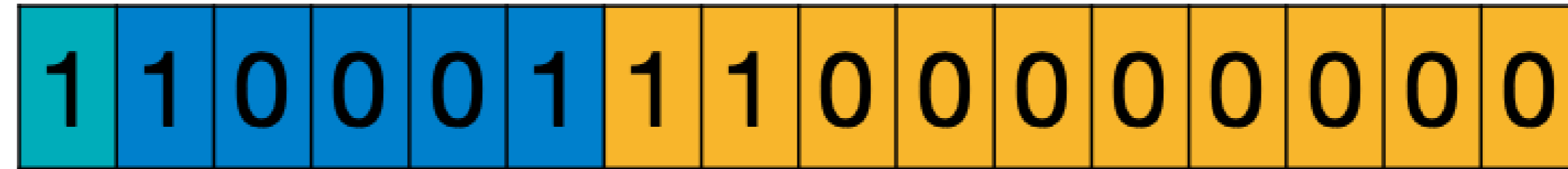


8

7

16

Exercise



Sign 5 bit Exponent

10 bit Fraction

- Sign: -
- Exponent
 - Bias: $2^4 - 1 = 15_{10}$
 - $10001_2 - 15_{10} = 17_{10} - 15_{10} = 2_{10}$
- Fraction
 - $1100000000_2 = 0.75_{10}$
- Answer: $-(1 + 0.75) \times 2^2 = -7.10_{10}$

$$(-1)^{\text{sign}} \times (1 + \text{Fraction}) \times 2^{\text{Exponent}-127}$$

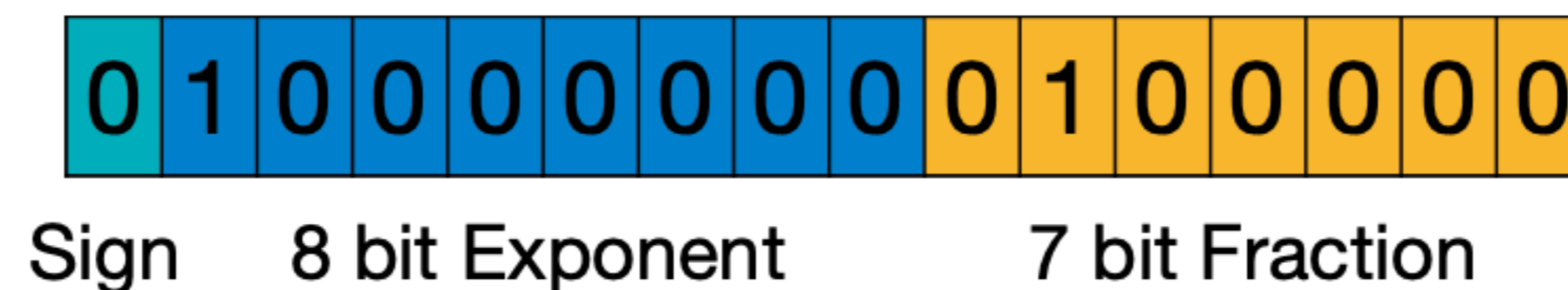
Exercise

Google Brain Float (BF16)







What is Decimal 2.5 in BF16?

- $2.5 = 1.25 \times 2^1$
- Sign: +
- Exponent: bias is $2^7 - 1 = 127$
 - $x - 127 = 1; x = 128_{10} = 10000000_2$
- Fraction: 7-bit fraction
 - $0.25 = 0100000_2$



Latest FP8

- Exponent width -> Range; Fraction width -> precision

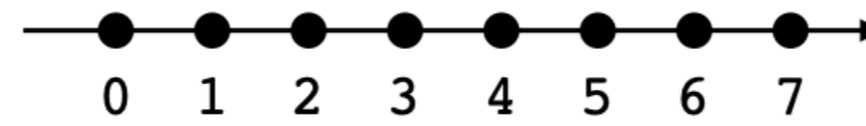
	Exponent (bits)	Fraction (bits)	Total (bits)
IEEE 754 Single Precision 32-bit Float (IEEE FP32) 	8	23	32
IEEE 754 Half Precision 16-bit Float (IEEE FP16) 	5	10	16
Nvidia FP8 (E4M3)  <ul style="list-style-type: none">* FP8 E4M3 does not have INF, and $S.1111.111_2$ is used for NaN.* Largest FP8 E4M3 normal value is $S.1111.110_2 = 448$.	4	3	8
Nvidia FP8 (E5M2) for gradient in the backward  <ul style="list-style-type: none">* FP8 E5M2 have INF ($S.11111.00_2$) and NaN ($S.11111.XX_2$).* Largest FP8 E5M2 normal value is $S.11110.11_2 = 57344$.	5	2	8

INT4 and FP8

INT4

S			
---	--	--	--

-1, -2, -3, -4, -5, -6, -7, -8
0, 1, 2, 3, 4, 5, 6, 7



-1, -2, -3, -4, -5, -6, -7, -8
0, 1, 2, 3, 4, 5, 6, 7

0	0	0	1
---	---	---	---

=1

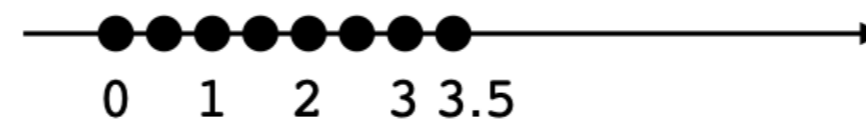
0	1	1	1
---	---	---	---

=7

FP4 (E1M2)

S	E	M	M
---	---	---	---

-0, -0.5, -1, -1.5, -2, -2.5, -3, -3.5
0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5



-0, -1, -2, -3, -4, -5, -6, -7 $\times 0.5$
0, 1, 2, 3, 4, 5, 6, 7 $\times 0.5$

0	0	0	1
---	---	---	---

$=0.25 \times 2^{1-0} = 0.5$

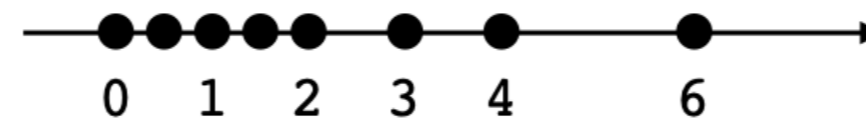
0	1	1	1
---	---	---	---

$=(1+0.75) \times 2^{1-0} = 3.5$

FP4 (E2M1)

S	E	E	M
---	---	---	---

-0, -0.5, -1, -1.5, -2, -3, -4, -6
0, 0.5, 1, 1.5, 2, 3, 4, 6



-0, -1, -2, -3, -4, -6, -8, -12 $\times 0.5$
0, 1, 2, 3, 4, 6, 8, 12 $\times 0.5$

0	0	0	1
---	---	---	---

$=0.5 \times 2^{1-1} = 0.5$

0	1	1	1
---	---	---	---

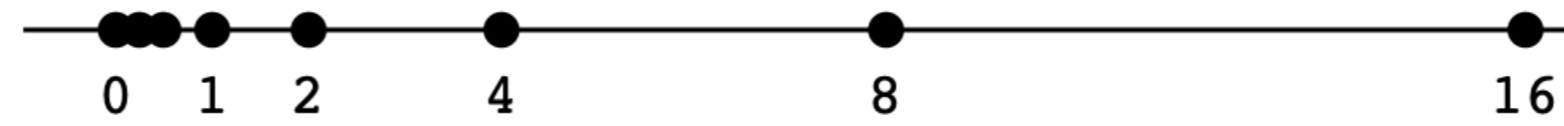
$=(1+0.5) \times 2^{3-1} = 1$

no inf, no NaN

FP4 (E3M0)

S	E	E	E
---	---	---	---

-0, -0.25, -0.5, -1, -2, -4, -8, -16
0, 0.25, 0.5, 1, 2, 4, 8, 16



-0, -1, -2, -4, -8, -16, -32, -64 $\times 0.25$
0, 1, 2, 4, 8, 16, 32, 64 $\times 0.25$

0	0	0	1
---	---	---	---

$=(1+0) \times 2^{1-3} = 0.25$

0	1	1	1
---	---	---	---

$=(1+0) \times 2^{7-3} = 16$

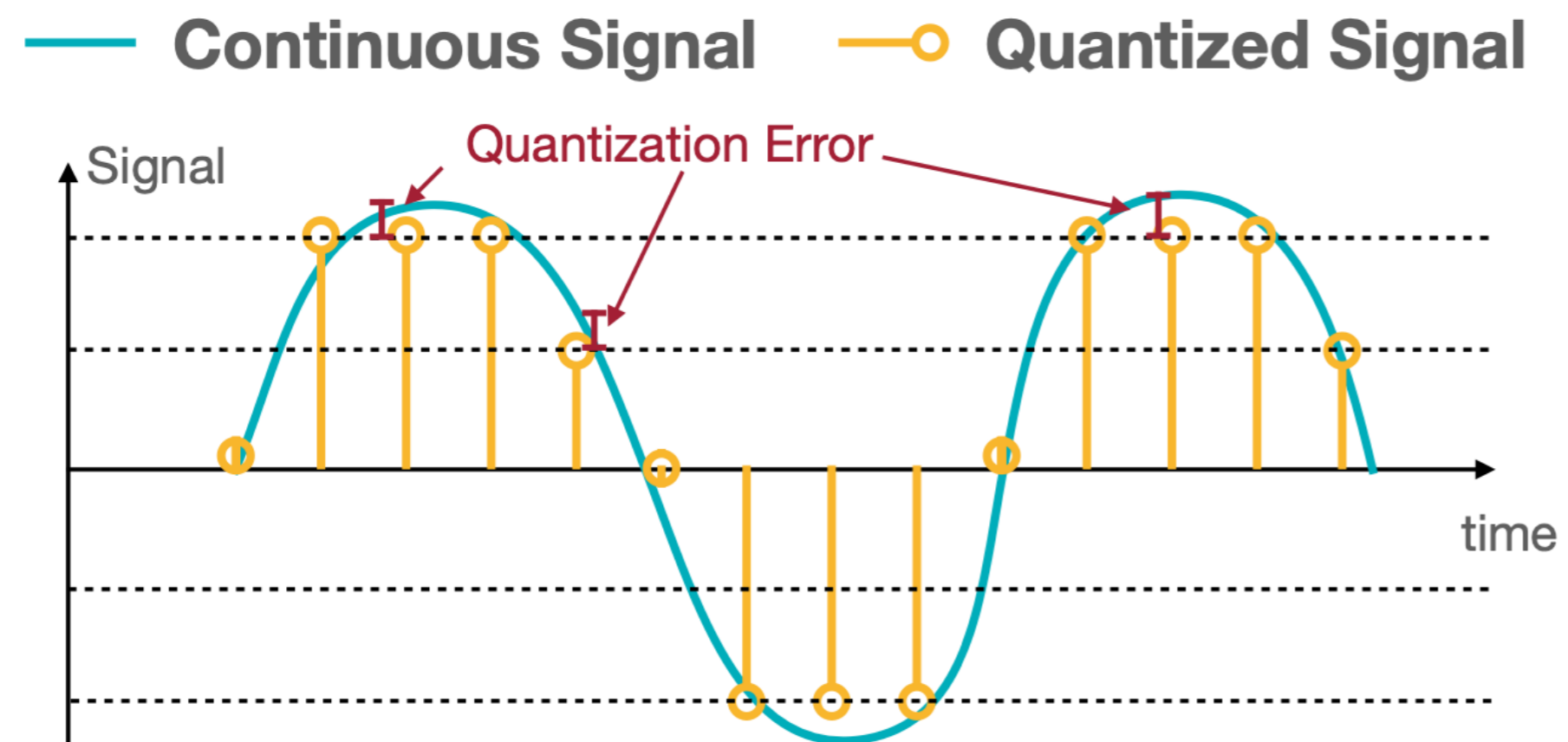
no inf, no NaN

Quantization

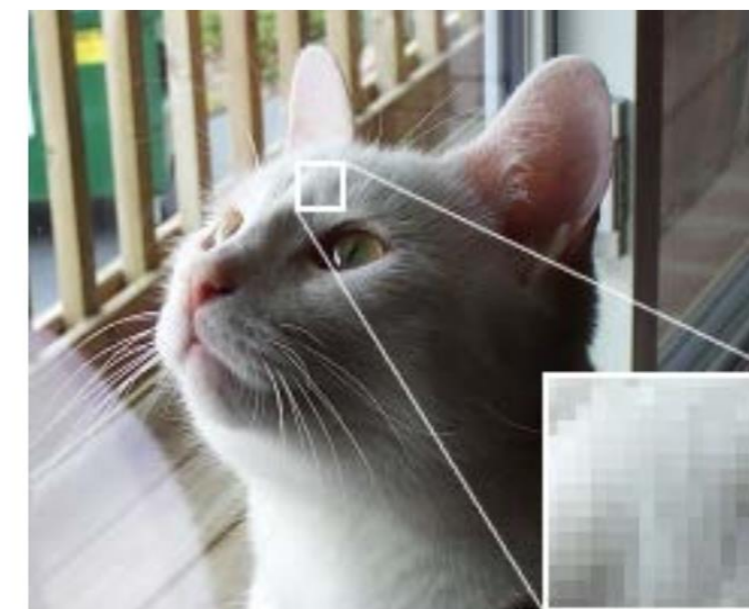
- Digital representation of data
- **Basics of quantization**
- Quantization in ML
- Mixed precision training

What is Quantization

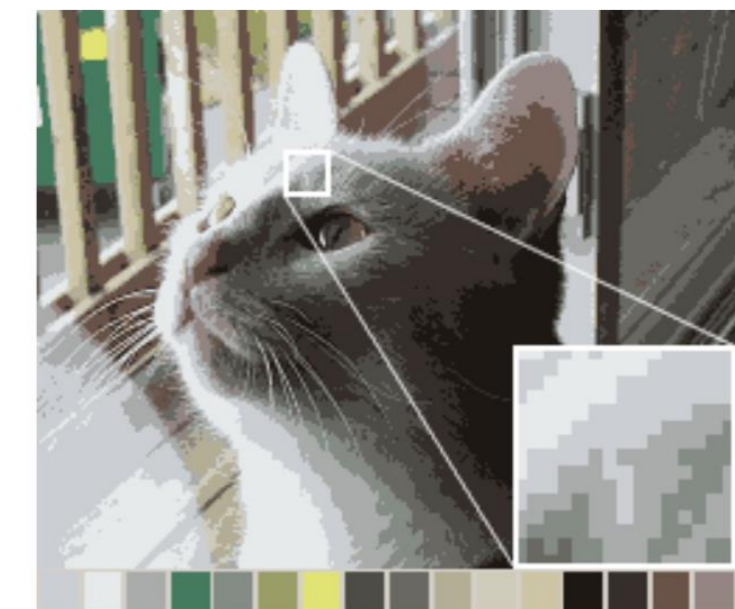
Quantization is the process of constraining an input from a continuous or otherwise large set of values to a discrete set.



Original Image



16-Color Image



Images are in the public domain.

“Palettization”

Quantization Basics

2.09	-0.98	1.48	0.09
0.05	-0.14	-1.08	2.12
-0.91	1.92	0	-1.03
1.87	0	1.53	1.49

3	0	2	1	3:	2.00
1	1	0	3	2:	1.50
0	3	1	0	1:	0.00
3	1	2	2	0:	-1.00

**K-Means-based
Quantization**

1	-2	0	-1
-1	-1	-2	1
-2	1	-1	-2
1	-1	0	0

$(-1) \times 1.07$

**Linear
Quantization**

Storage

Floating point
weights

Compute

Floating point
arithmetic

K-means Quantization

weights
(32-bit float)

2.09	-0.98	1.48	0.09
0.05	-0.14	-1.08	2.12
-0.91	1.92	0	-1.03
1.87	0	1.53	1.49

2.09, 2.12, 1.92, 1.87



2.0

quantization
error:

0.09, 0.12, -0.08, -0.13

K-means Quantization: Clustering

weights
(32-bit float)

2.09	-0.98	1.48	0.09
0.05	-0.14	-1.08	2.12
-0.91	1.92	0	-1.03
1.87	0	1.53	1.49



cluster index
(2-bit int)

3	0	2	1
1	1	0	3
0	3	1	0
3	1	2	2

indexes

centroids

3:	2.00
2:	1.50
1:	0.00
0:	-1.00

codebook

reconstructed weights
(32-bit float)

2.00	-1.00	1.50	0.00
0.00	0.00	-1.00	2.00
-1.00	2.00	0.00	-1.00
2.00	0.00	1.50	1.50

quantization error

0.09	0.02	-0.02	0.09
0.05	-0.14	-0.08	0.12
0.09	-0.08	0	-0.03
-0.13	0	0.03	-0.01

storage

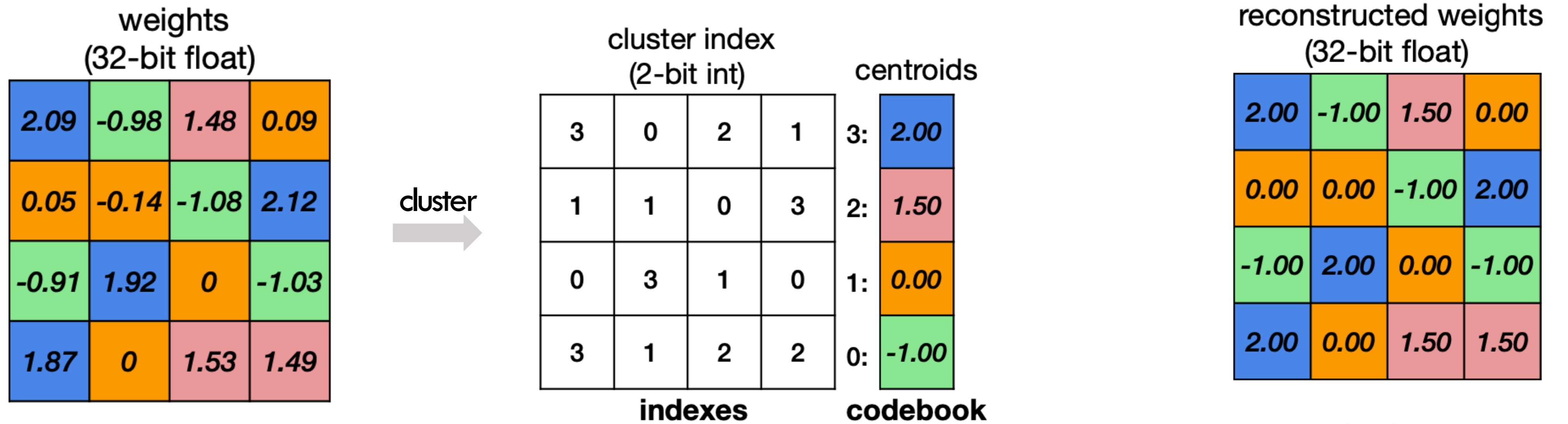
32bits x 16 = 512 bit =
64bytes

2 bit x 16 = 32bit
= 4 bytes

32 bit x 4 = 128bit
= 16 bytes

3.2x reduction

K-means Quantization: Clustering



Assume: N-bit quantization, and #entries = $M \gg 2^N$

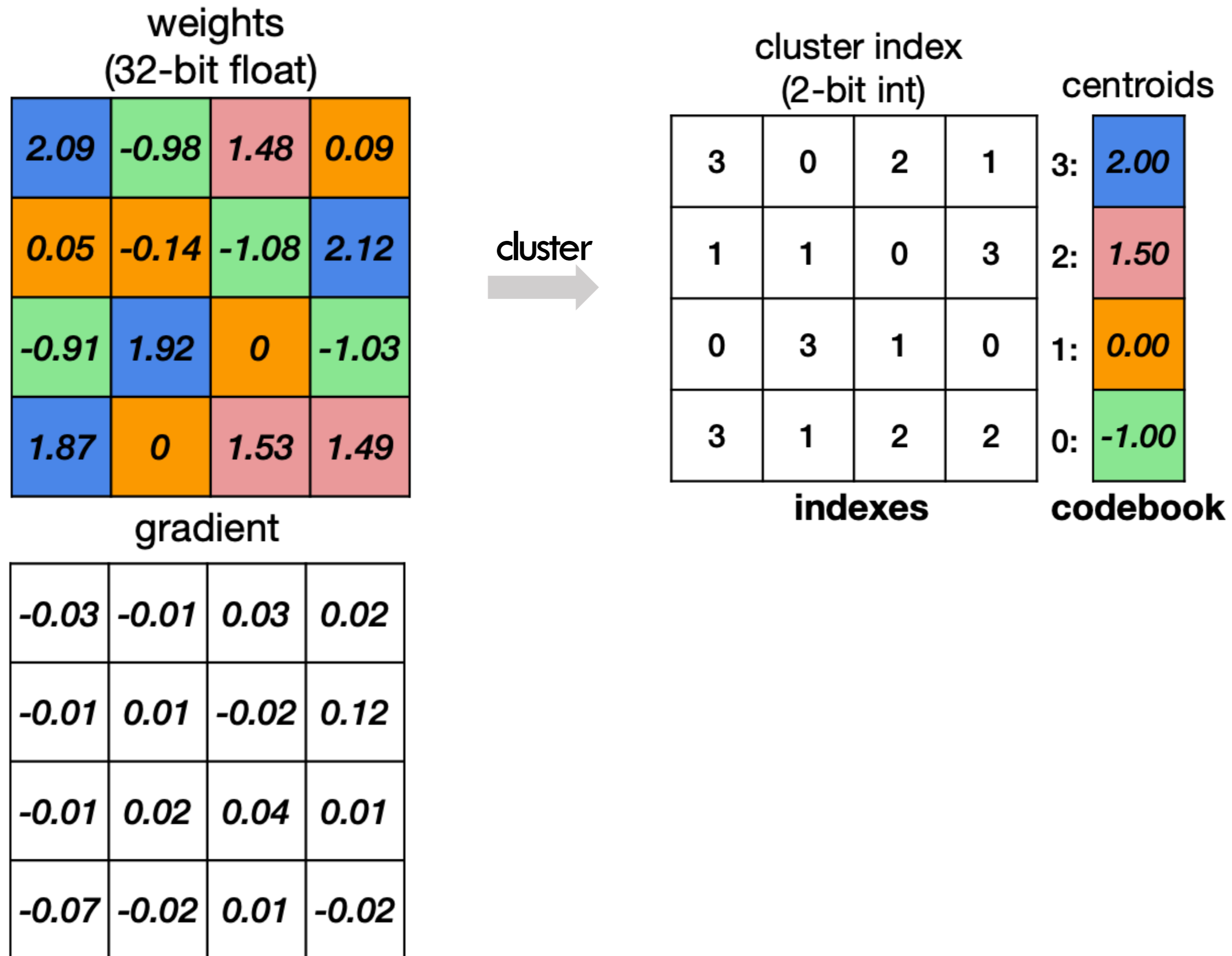
storage 32bits x $M = 32M$ bit N bit x $M = NM$ bit 32 bit x $2^N = 2^{(N+5)}$ bits

$$32M / NM = 32/N \times \text{reduction}$$

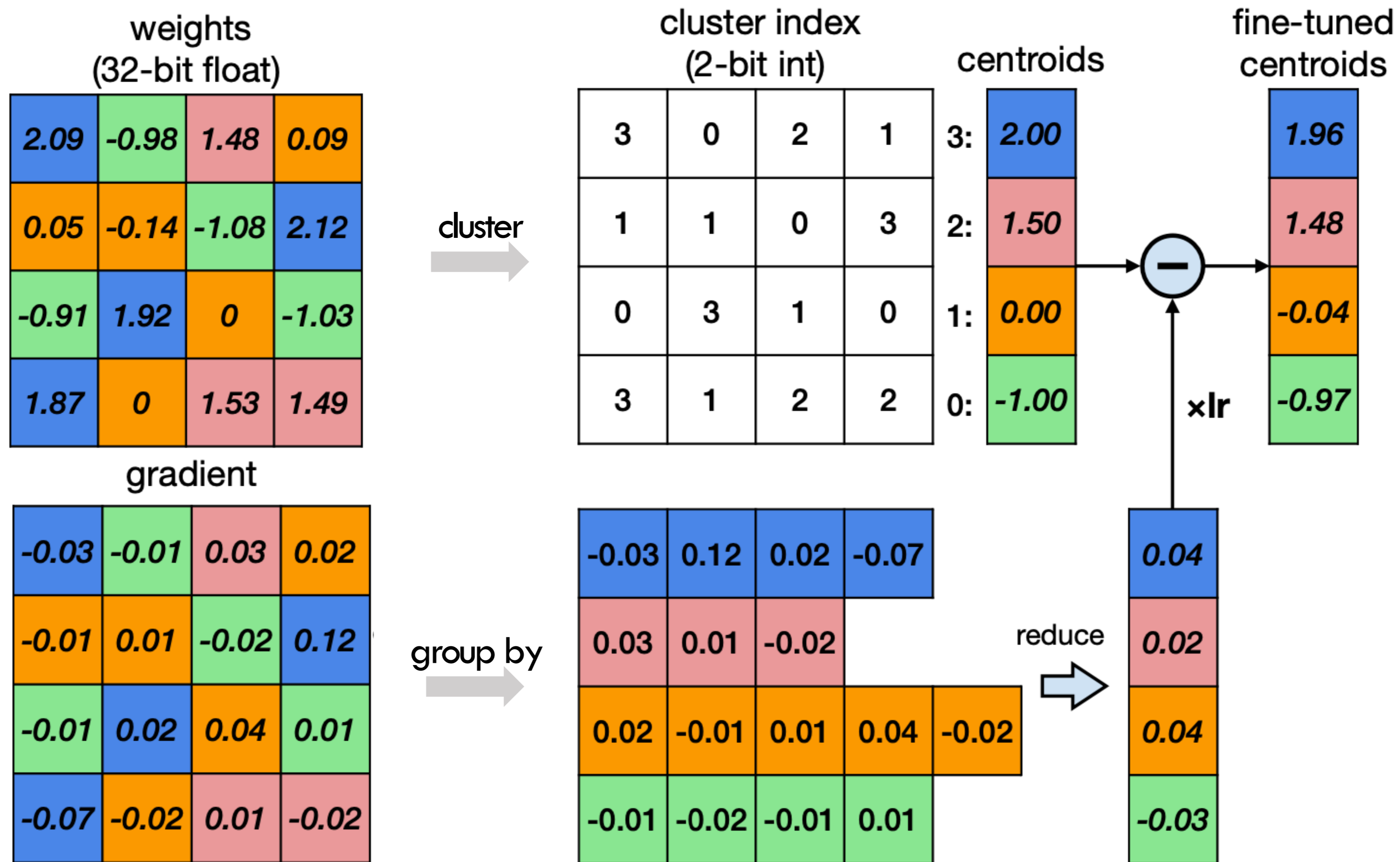
quantization error

0.09	0.02	-0.02	0.09
0.05	-0.14	-0.08	0.12
0.09	-0.08	0	-0.03
-0.13	0	0.03	-0.01

K-means Quantization: Backward

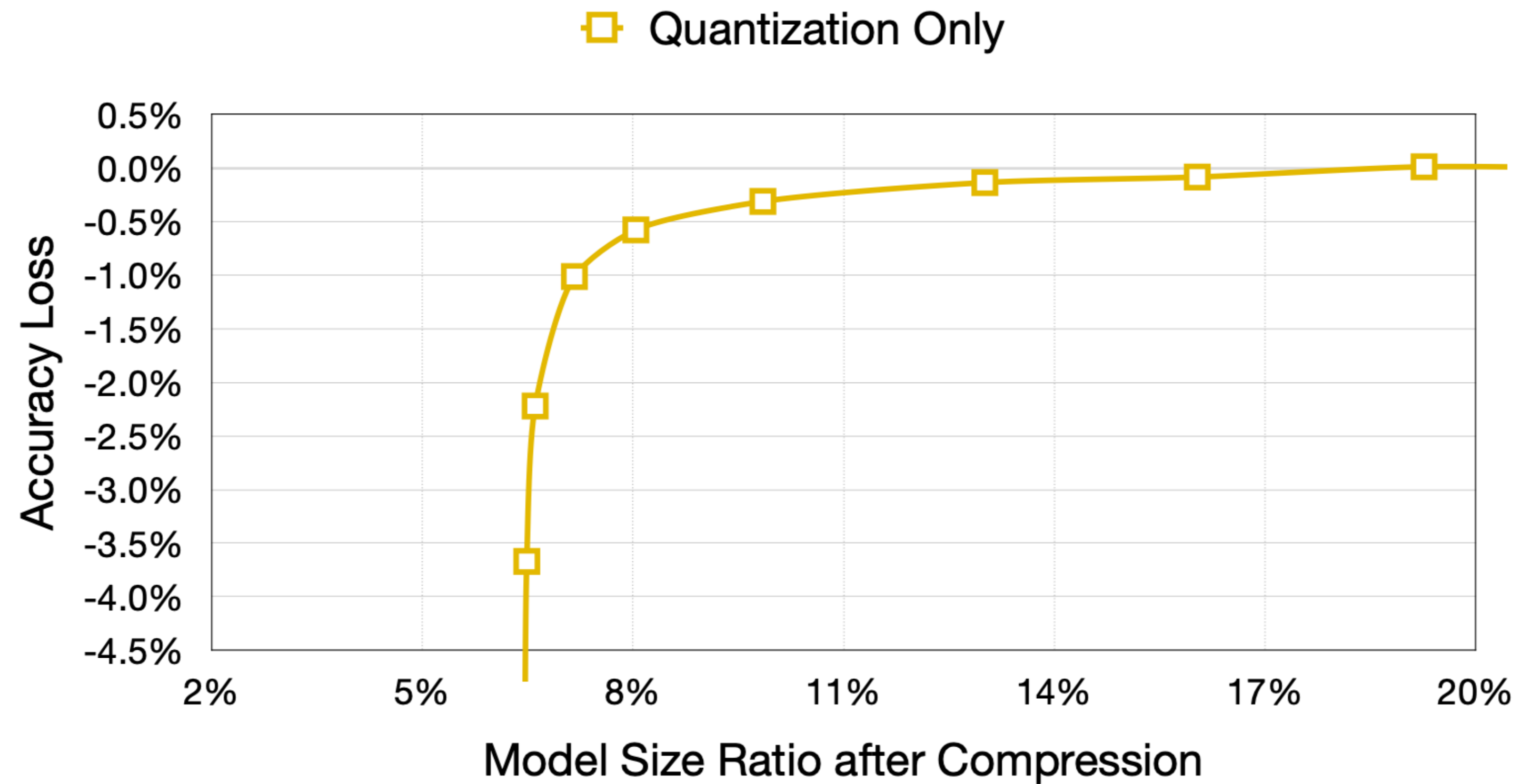


K-means Quantization: Backward

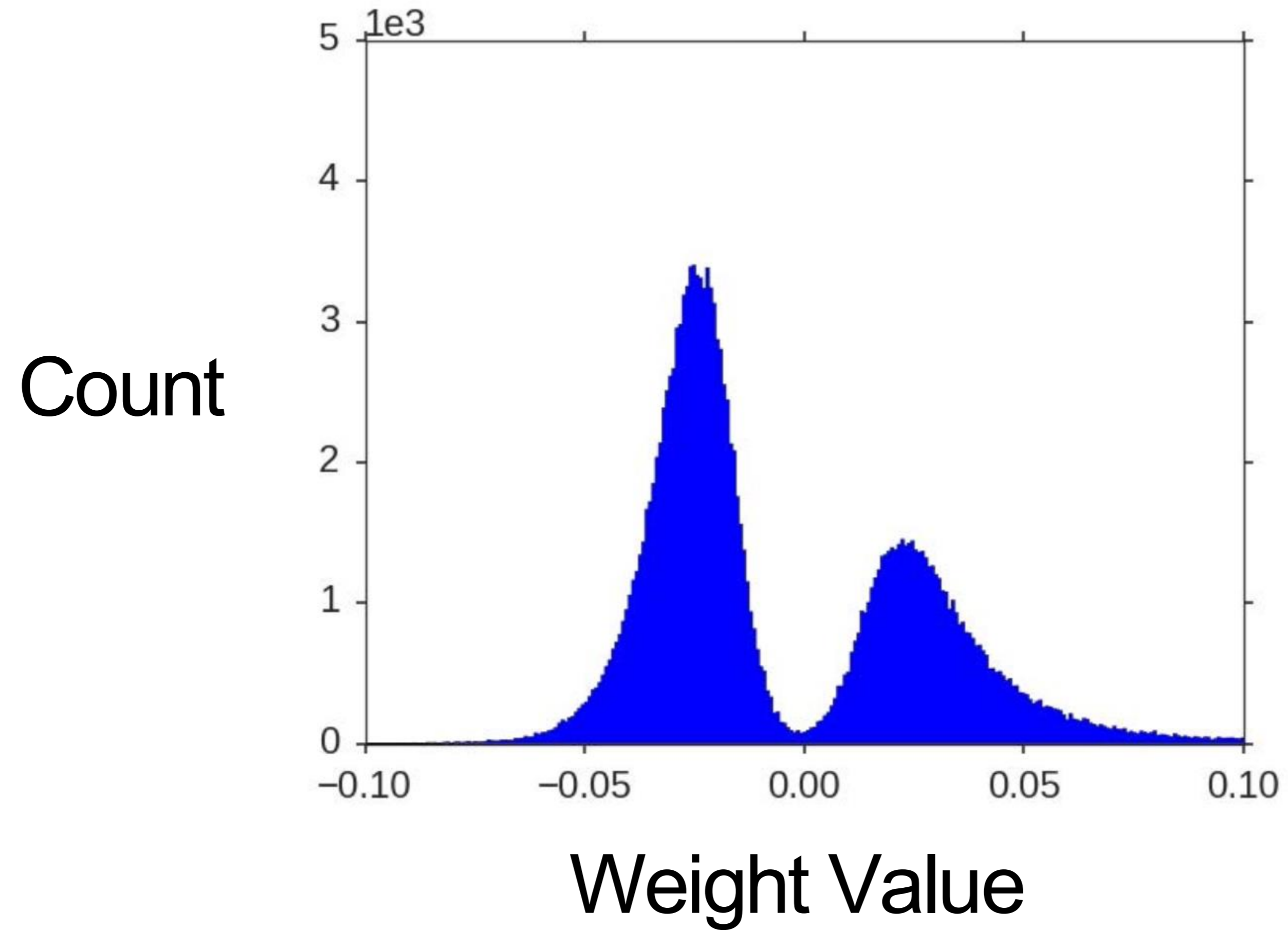


K-means Quantization:

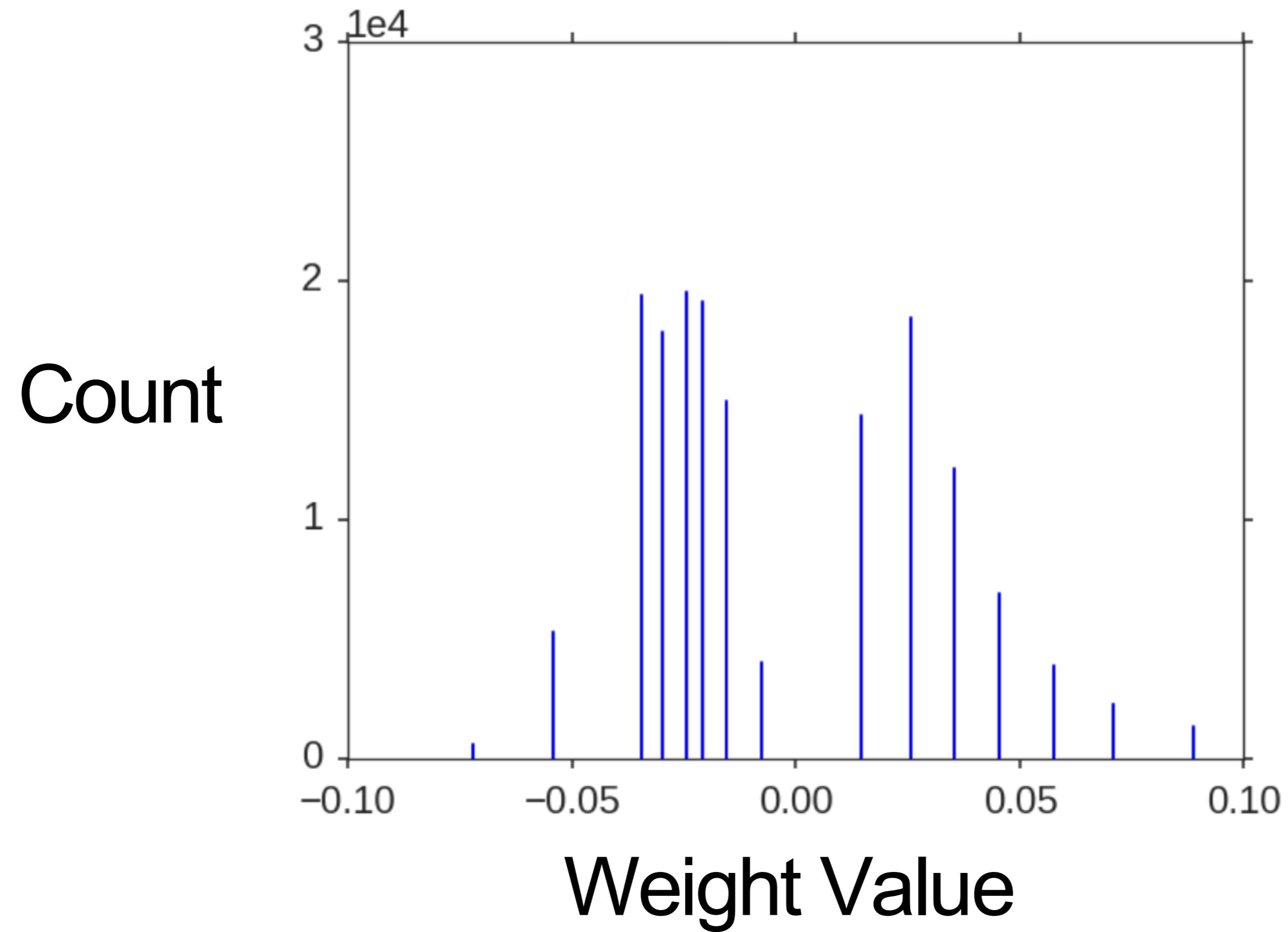
- **Accuracy vs. compression rate for AlexNet on ImageNet dataset**



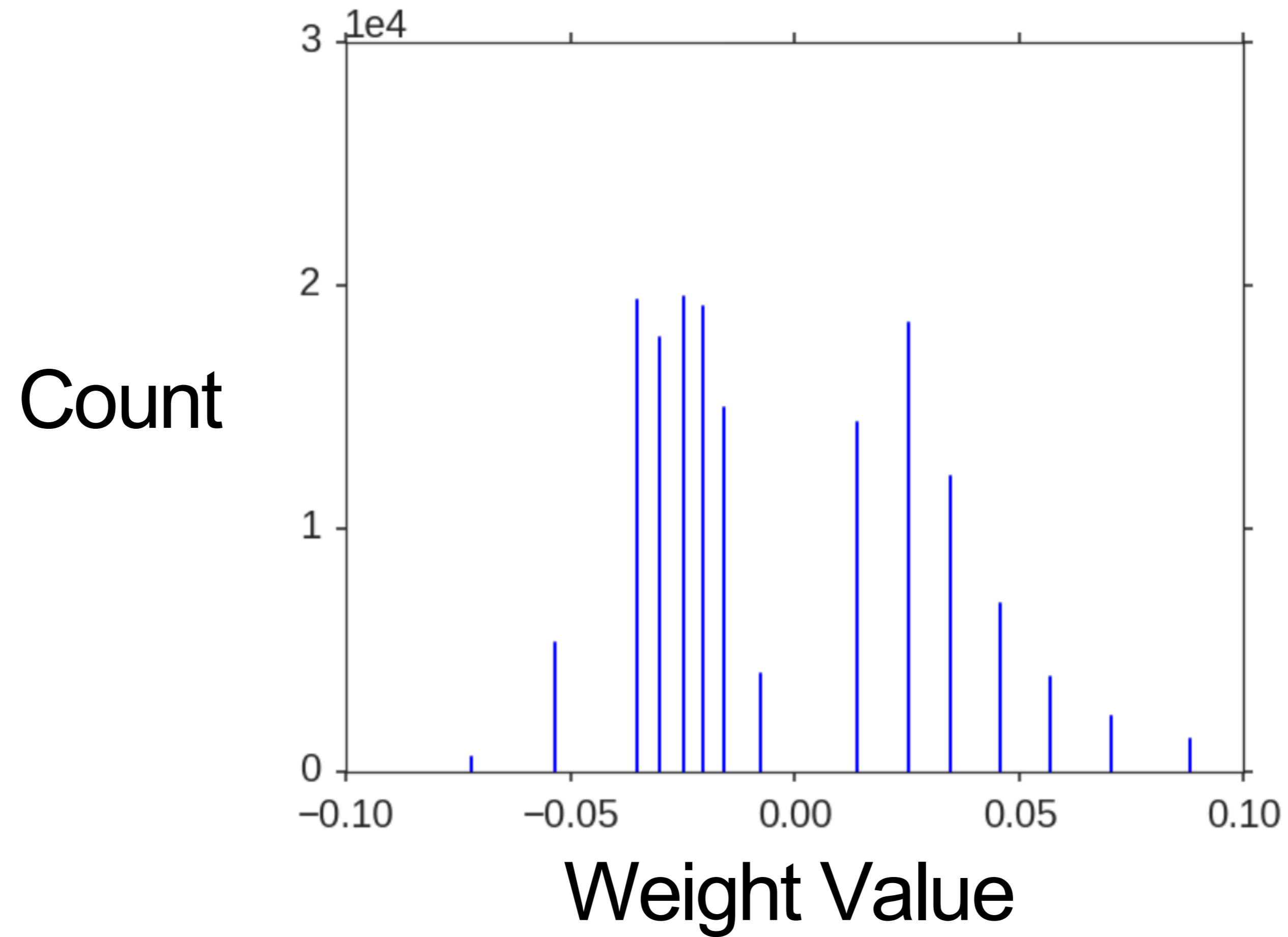
Before Quantization: Continuous Weights



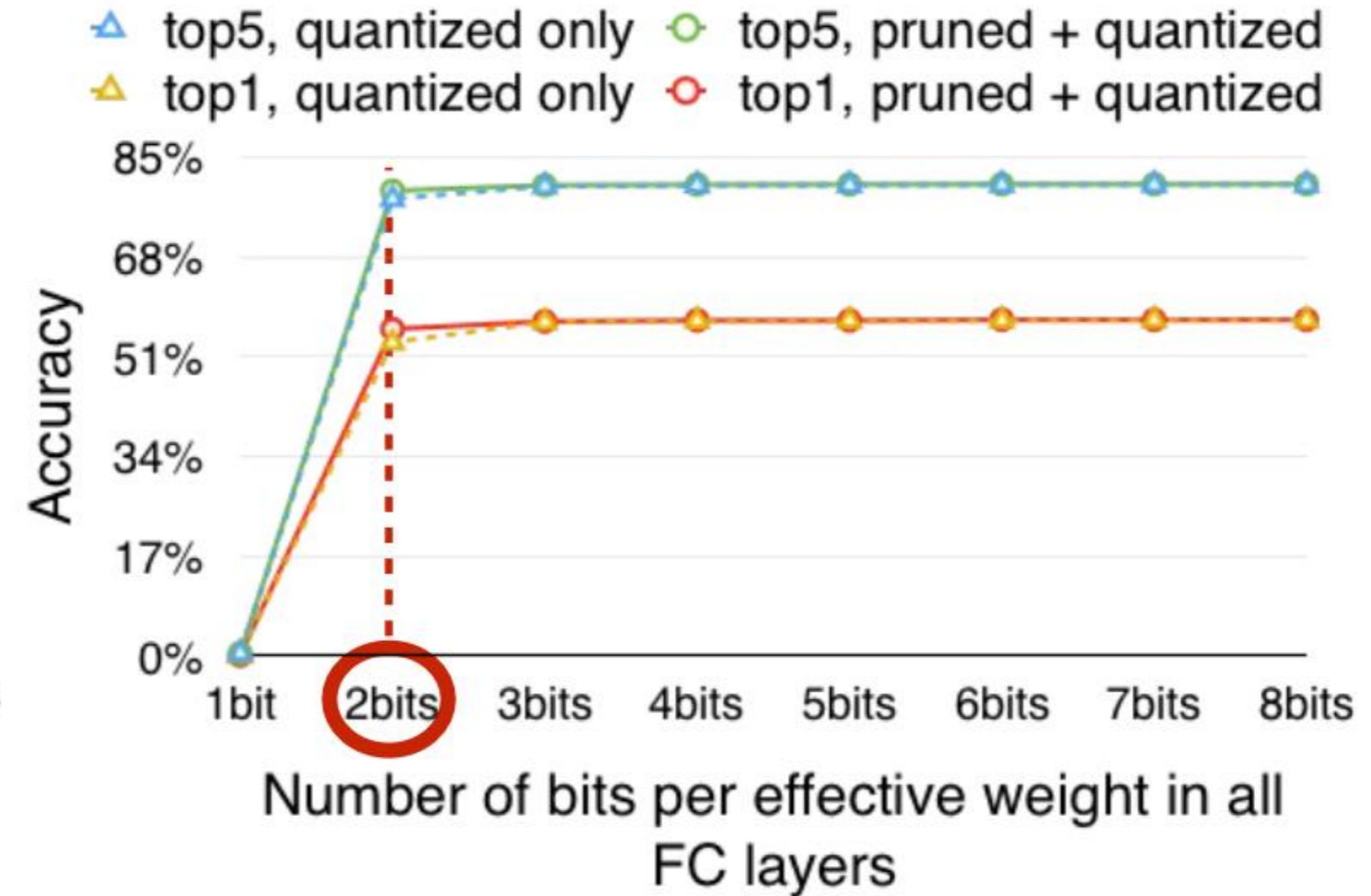
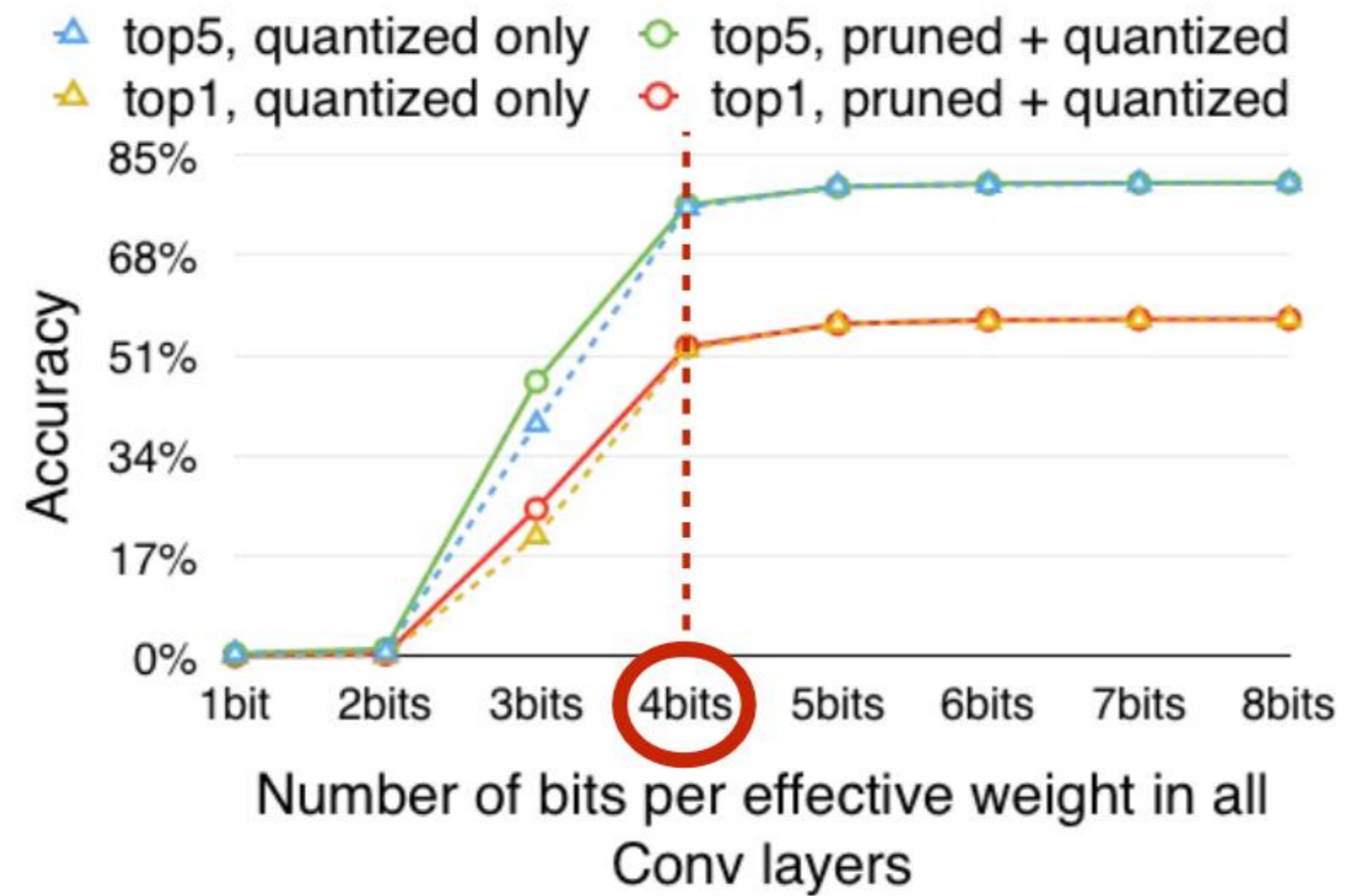
After Quantization: Discrete Weights



After Quantization: Weights Shift after training

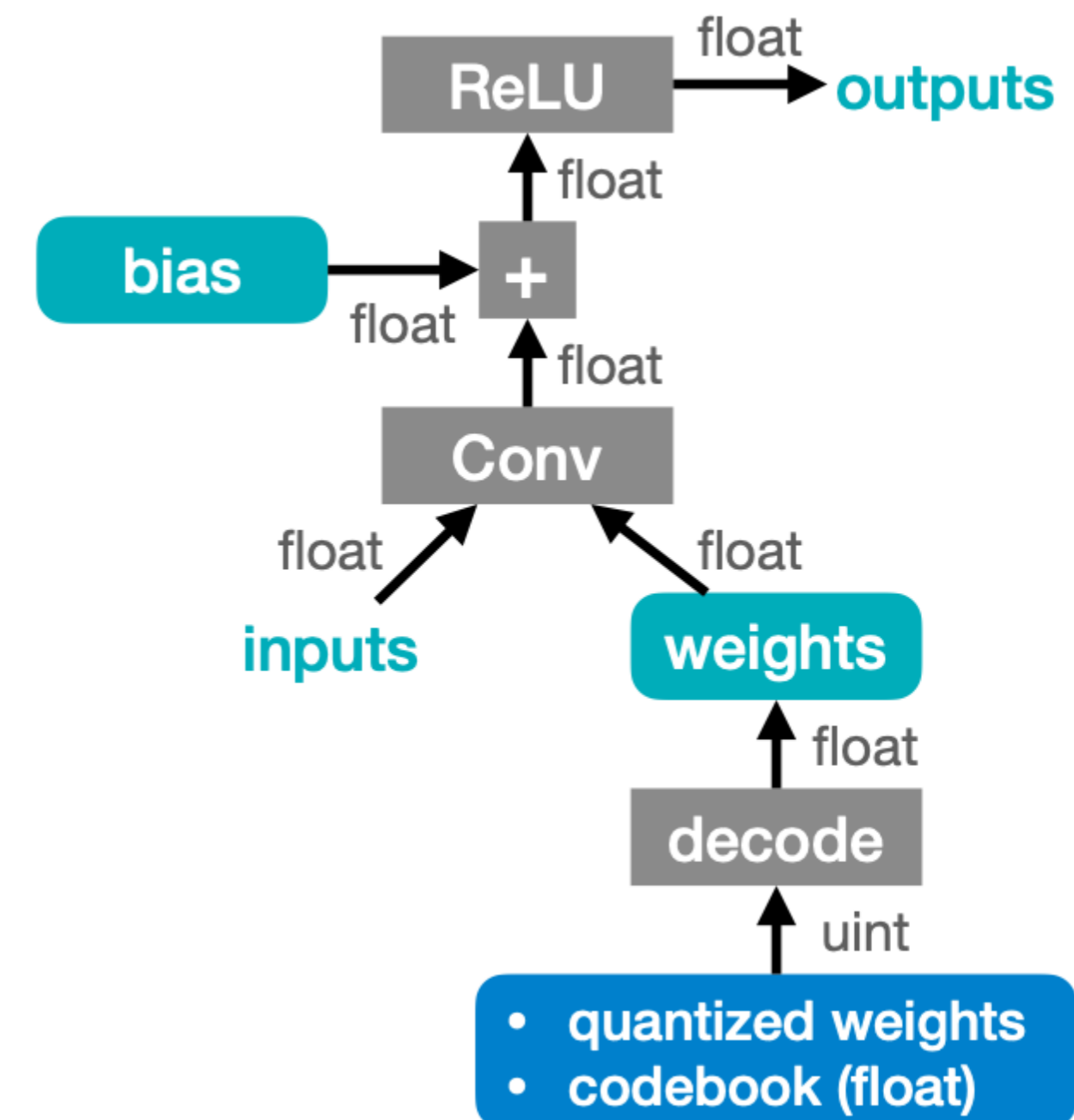
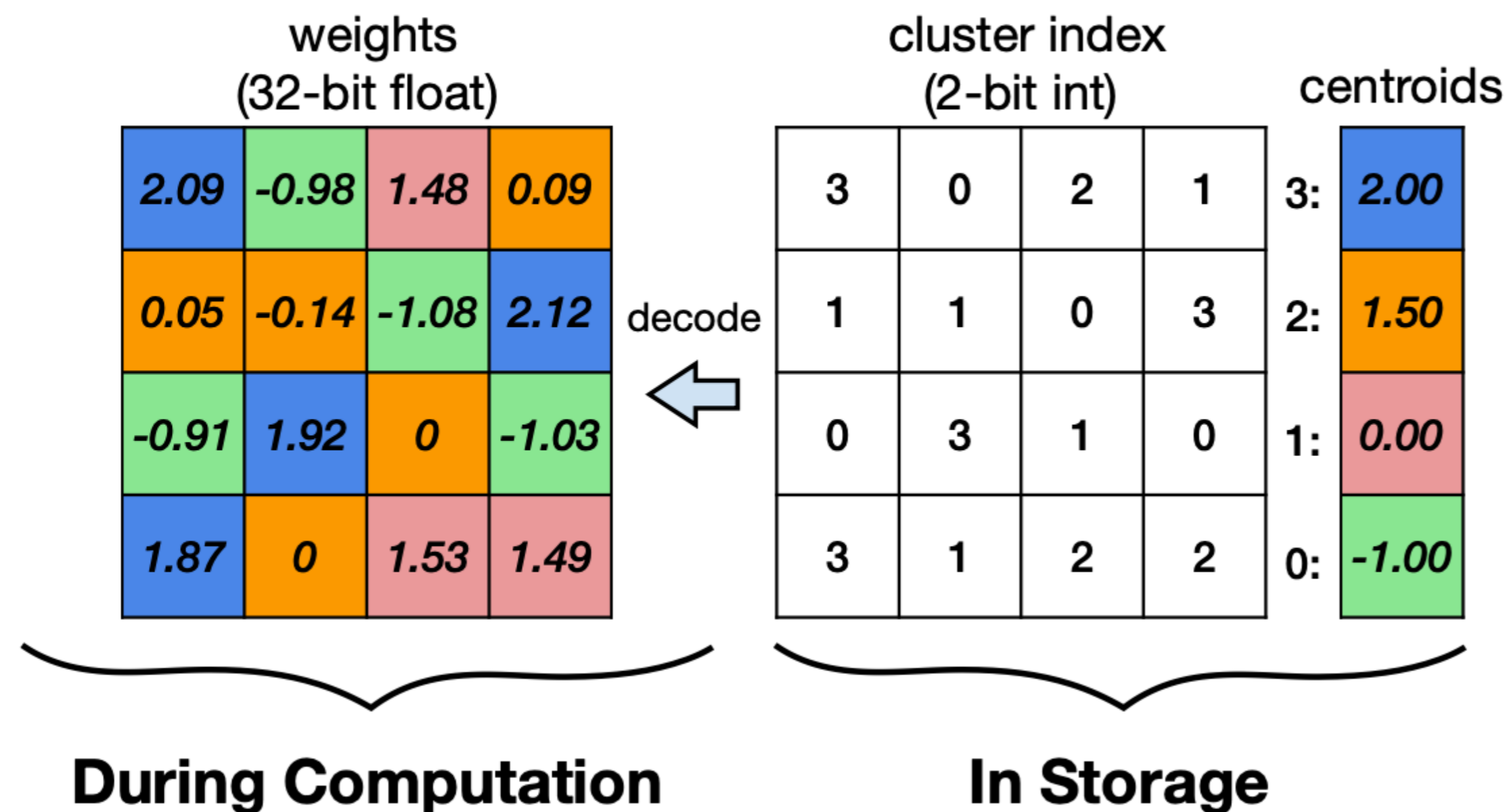


How Many Bits do We Need?



K-Means Quantization: Runtime

- Weights are decompressed using a lookup table (*i.e.*, codebook) at runtime. inference.
- Storage: quantized
- Compute: still float-point arithmetic



Quantization Basics

2.09	-0.98	1.48	0.09
0.05	-0.14	-1.08	2.12
-0.91	1.92	0	-1.03
1.87	0	1.53	1.49

3	0	2	1	3:	2.00
1	1	0	3	2:	1.50
0	3	1	0	1:	0.00
3	1	2	2	0:	-1.00

**K-Means-based
Quantization**

1	-2	0	-1
-1	-1	-2	1
-2	1	-1	-2
1	-1	0	0

$(-1) \times 1.07$

**Linear
Quantization**

Storage

Floating point
weights

integer weights;
floating-point
codebook

Compute

Floating point
arithmetic

Floating point
arithmetic

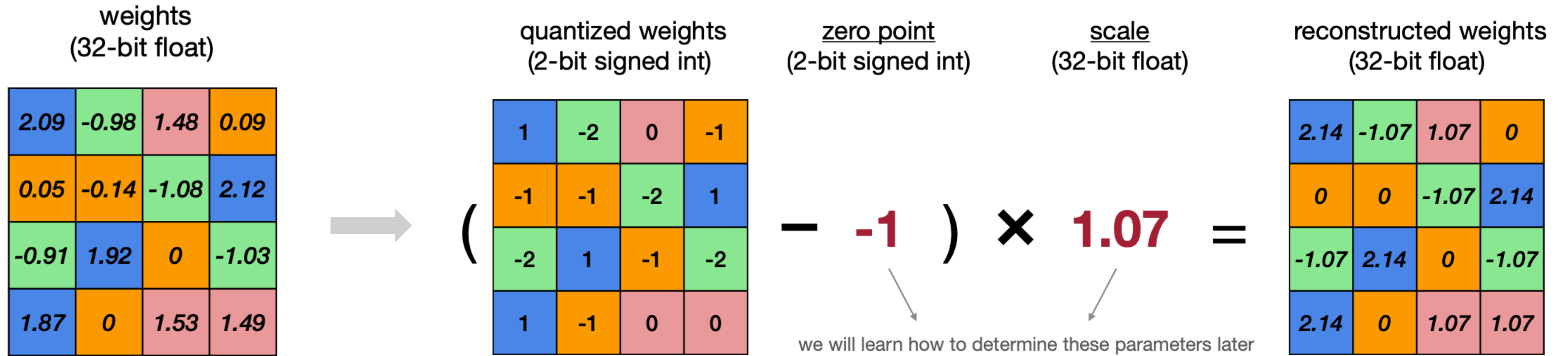
Linear Quantization

weights
(32-bit float)

<i>2.09</i>	<i>-0.98</i>	<i>1.48</i>	<i>0.09</i>
<i>0.05</i>	<i>-0.14</i>	<i>-1.08</i>	<i>2.12</i>
<i>-0.91</i>	<i>1.92</i>	<i>0</i>	<i>-1.03</i>
<i>1.87</i>	<i>0</i>	<i>1.53</i>	<i>1.49</i>

Linear Quantization

- A linear mapping of integers to real numbers



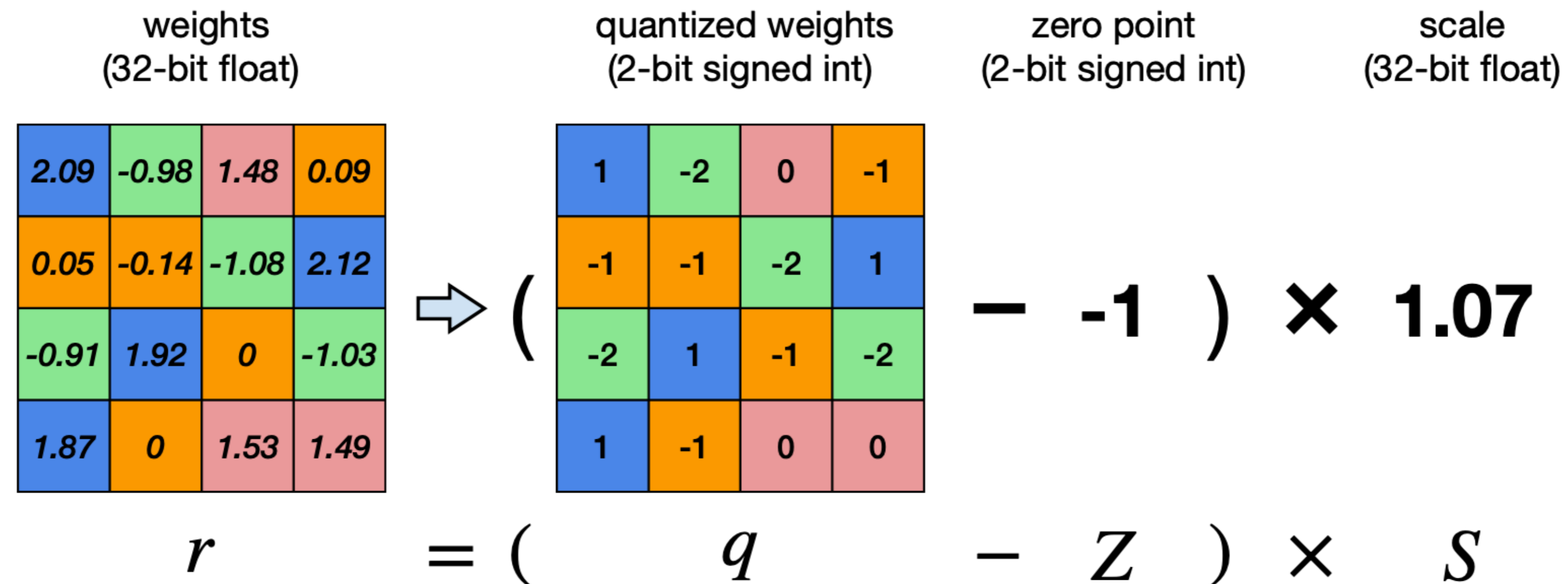
Binary	Decimal
01	1
00	0
11	-1
10	-2

quantization error

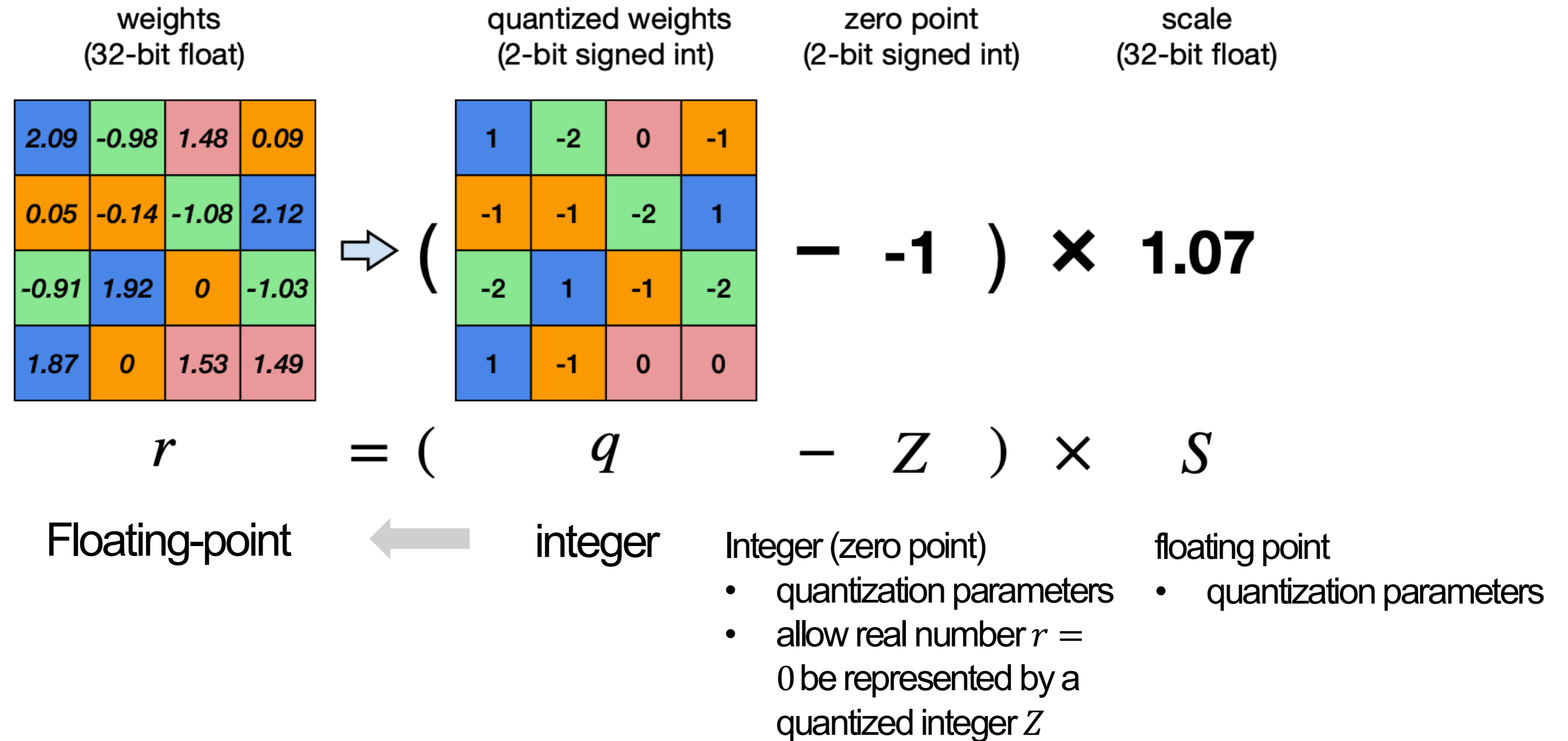
-0.05	0.09	0.41	0.09
0.05	-0.14	-0.01	-0.02
0.16	-0.22	0	0.04
-0.27	0	0.46	0.42

Linear Quantization

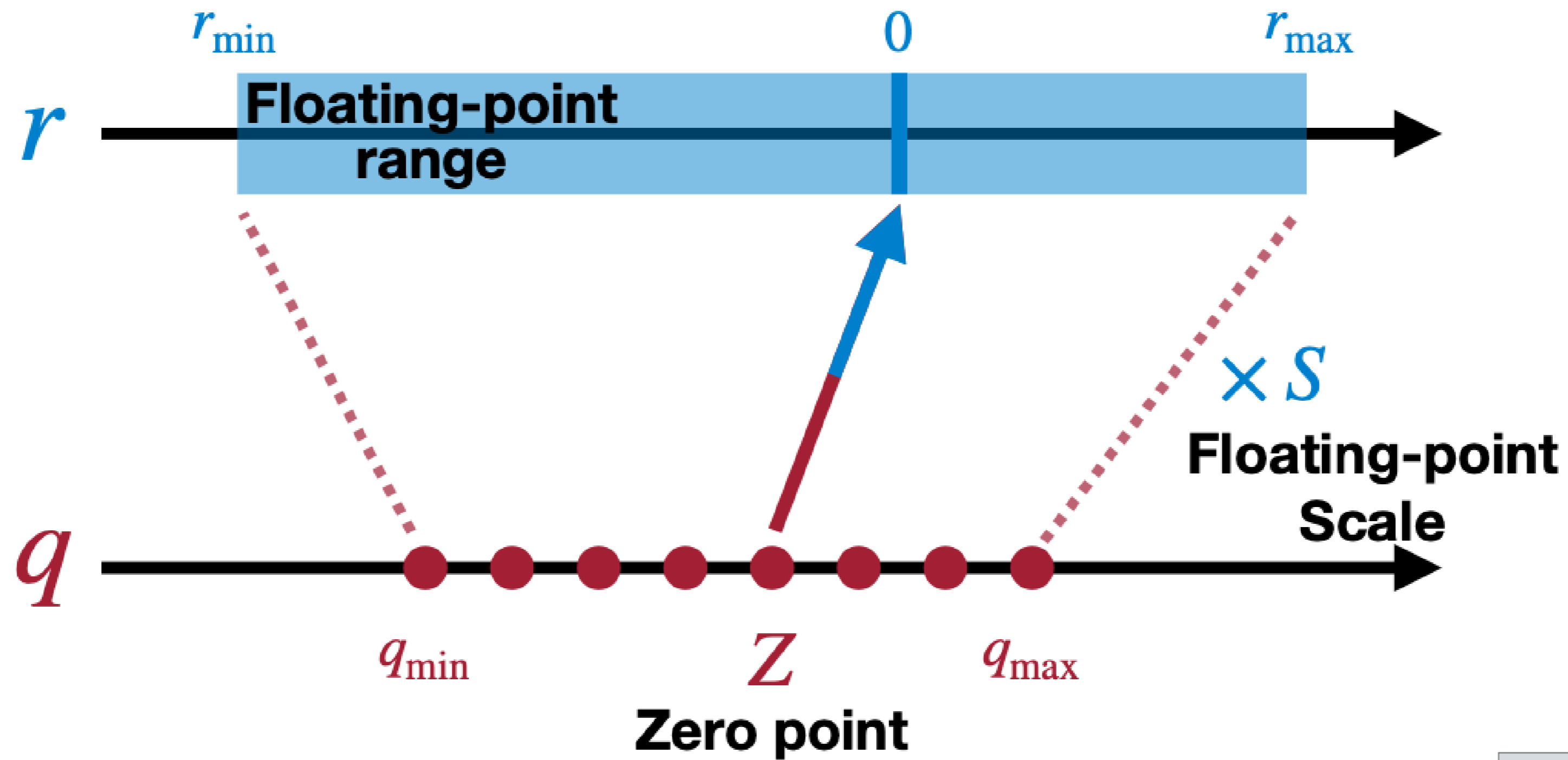
- Critical parameters to determine:
 - Zero point: Z
 - Scale: S



Linear Quantization: $r = S(q - Z)$



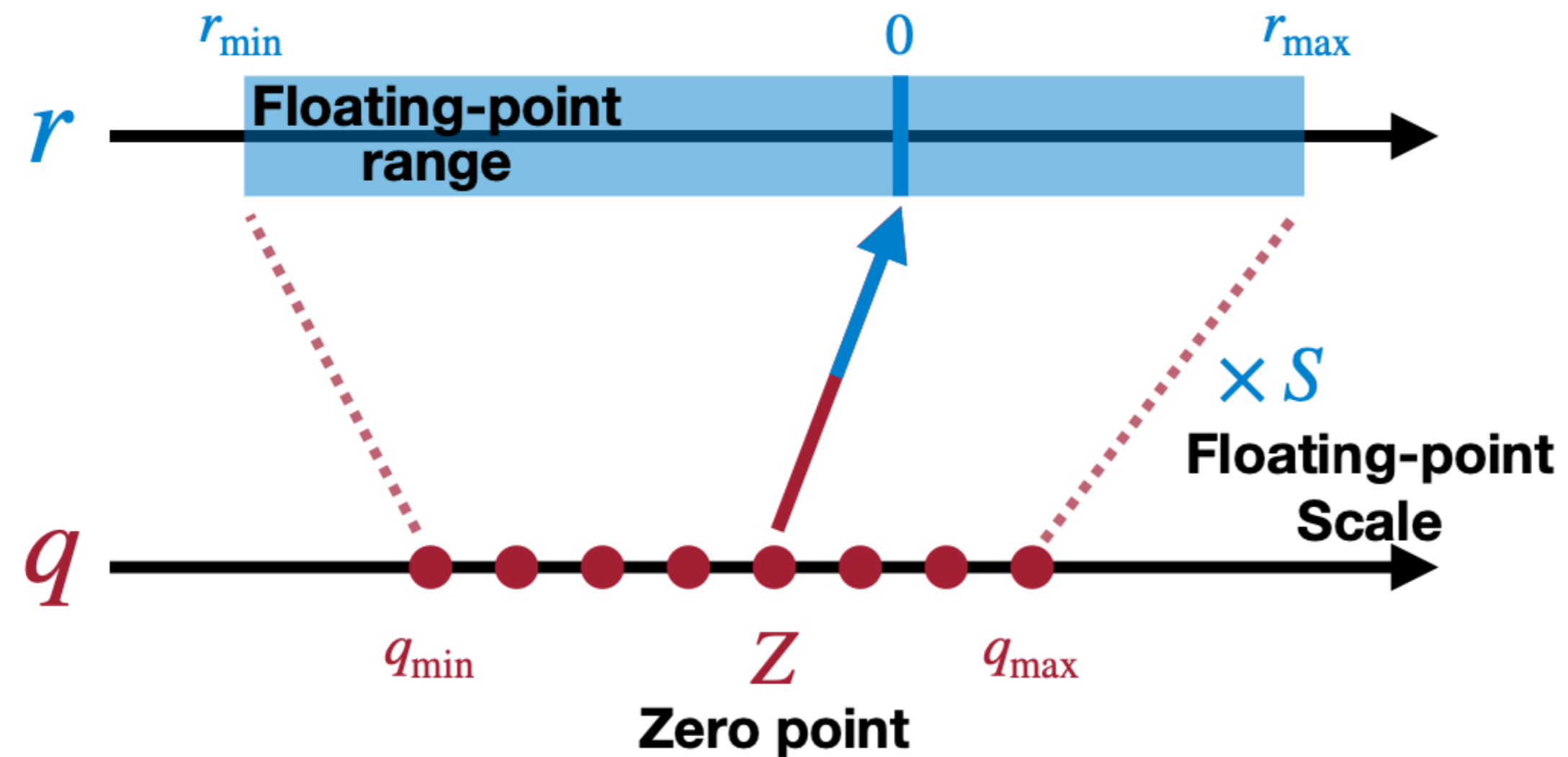
$r = S(q - Z)$: Geometric Interpretation



Q: How to determine S and Z ?

Bit Width	q_{\min}	q_{\max}
2	-2	1
3	-4	3
4	-8	7
N	-2^{N-1}	$2^{N-1}-1$

$r = S(q - Z)$: Determine S and Z



$$r_{\max} = S(q_{\max} - Z)$$

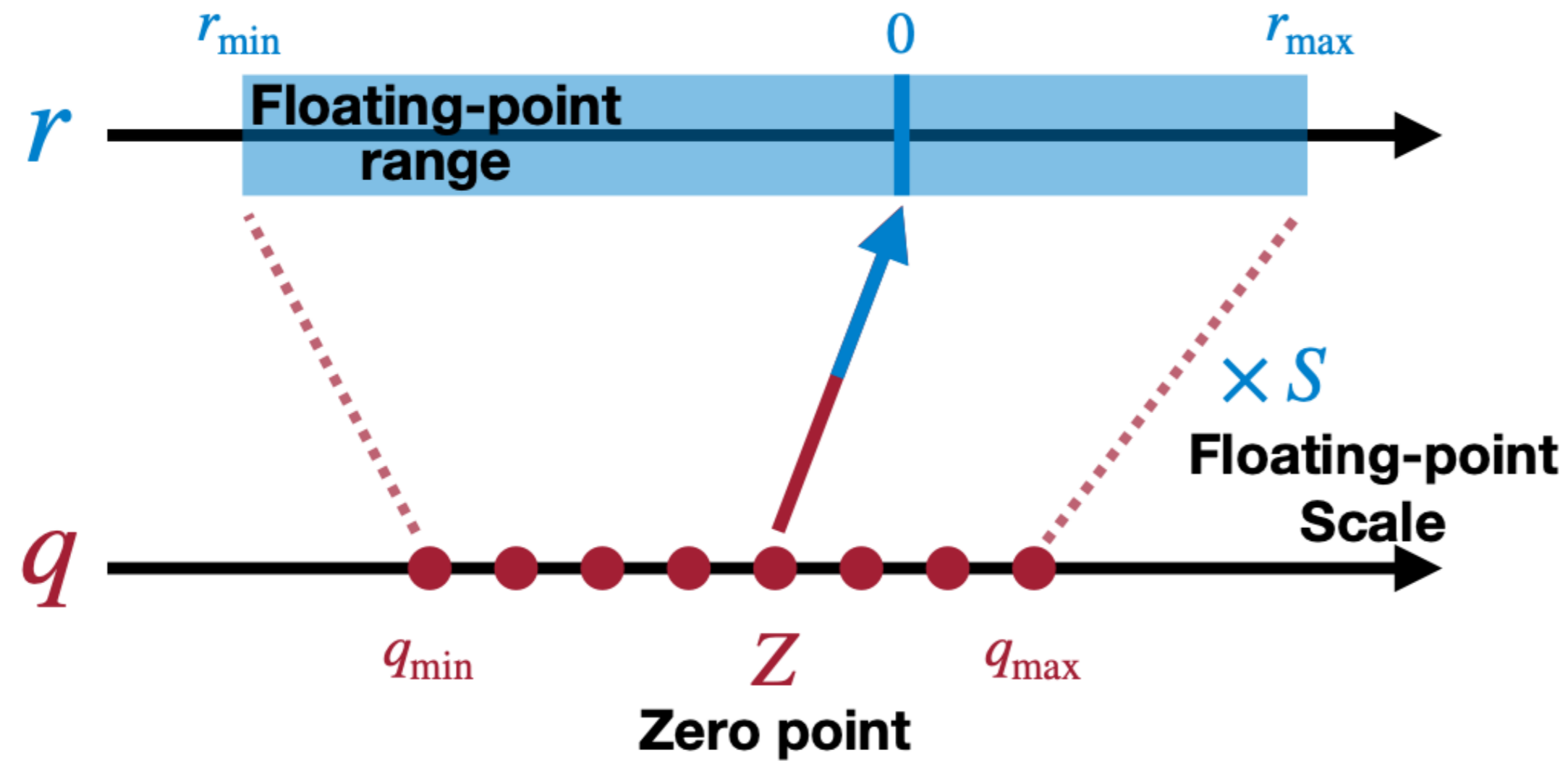
$$r_{\min} = S(q_{\min} - Z)$$



$$r_{\max} - r_{\min} = S(q_{\max} - q_{\min})$$

$$S = \frac{r_{\max} - r_{\min}}{q_{\max} - q_{\min}}$$

$r = S(q - Z)$: Determine S and Z

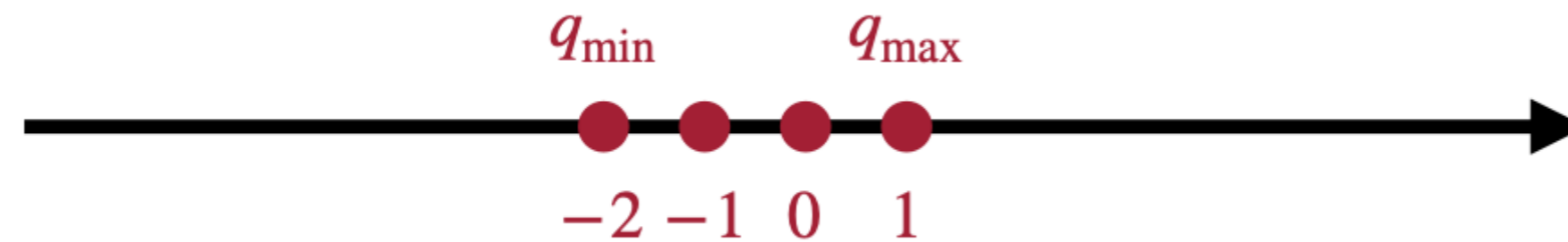


2.09	-0.98	1.48	0.09
0.05	-0.14	-1.08	2.12
-0.91	1.92	0	-1.03
1.87	0	1.53	1.49

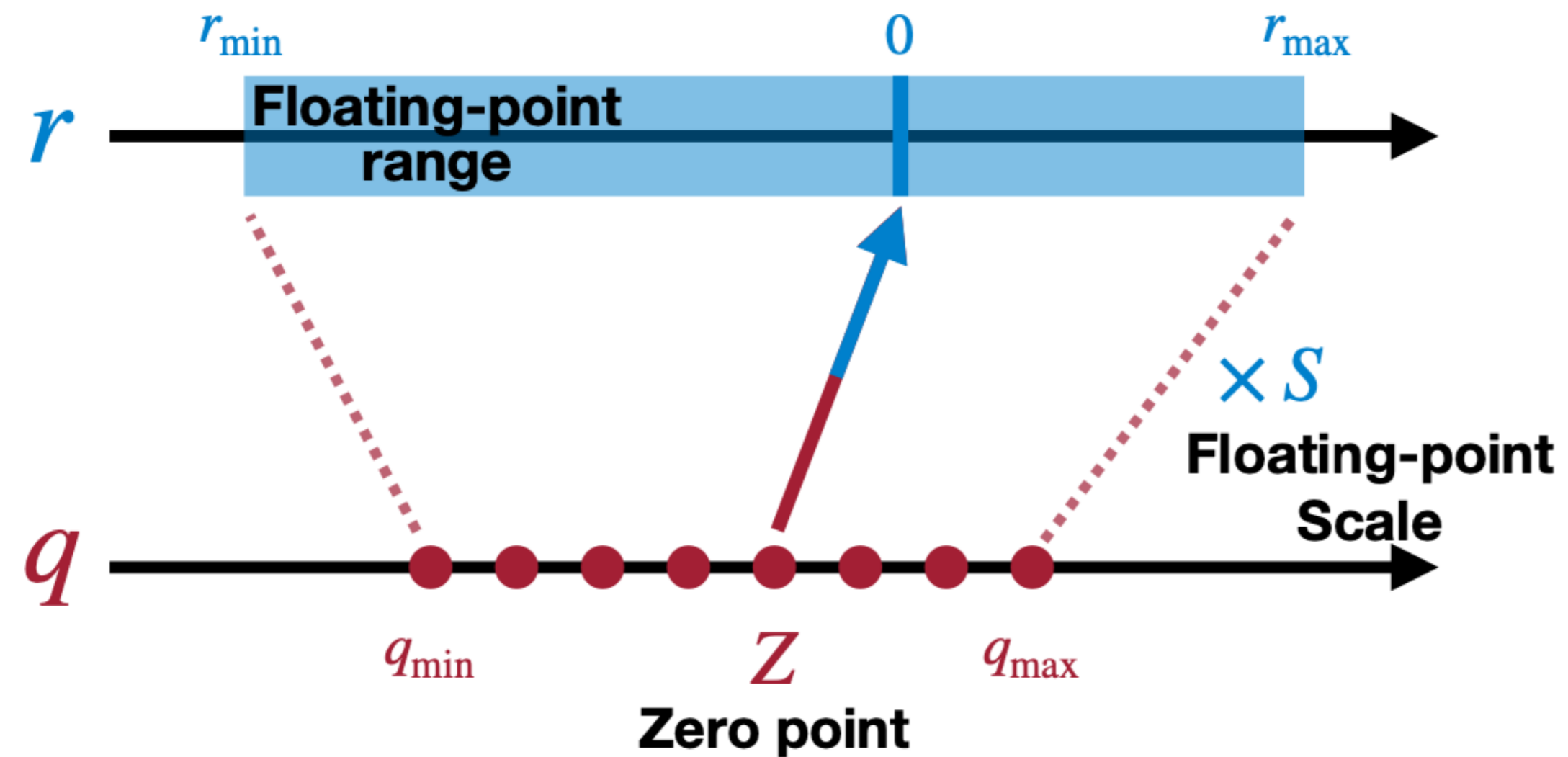
$$S = \frac{r_{\max} - r_{\min}}{q_{\max} - q_{\min}}$$

$$S = \frac{2.12 - (-1.08)}{1 - (-2)} = 1.07$$

Binary	Decimal
01	1
00	0
11	-1
10	-2



$r = S(q - Z)$: Determine S and Z

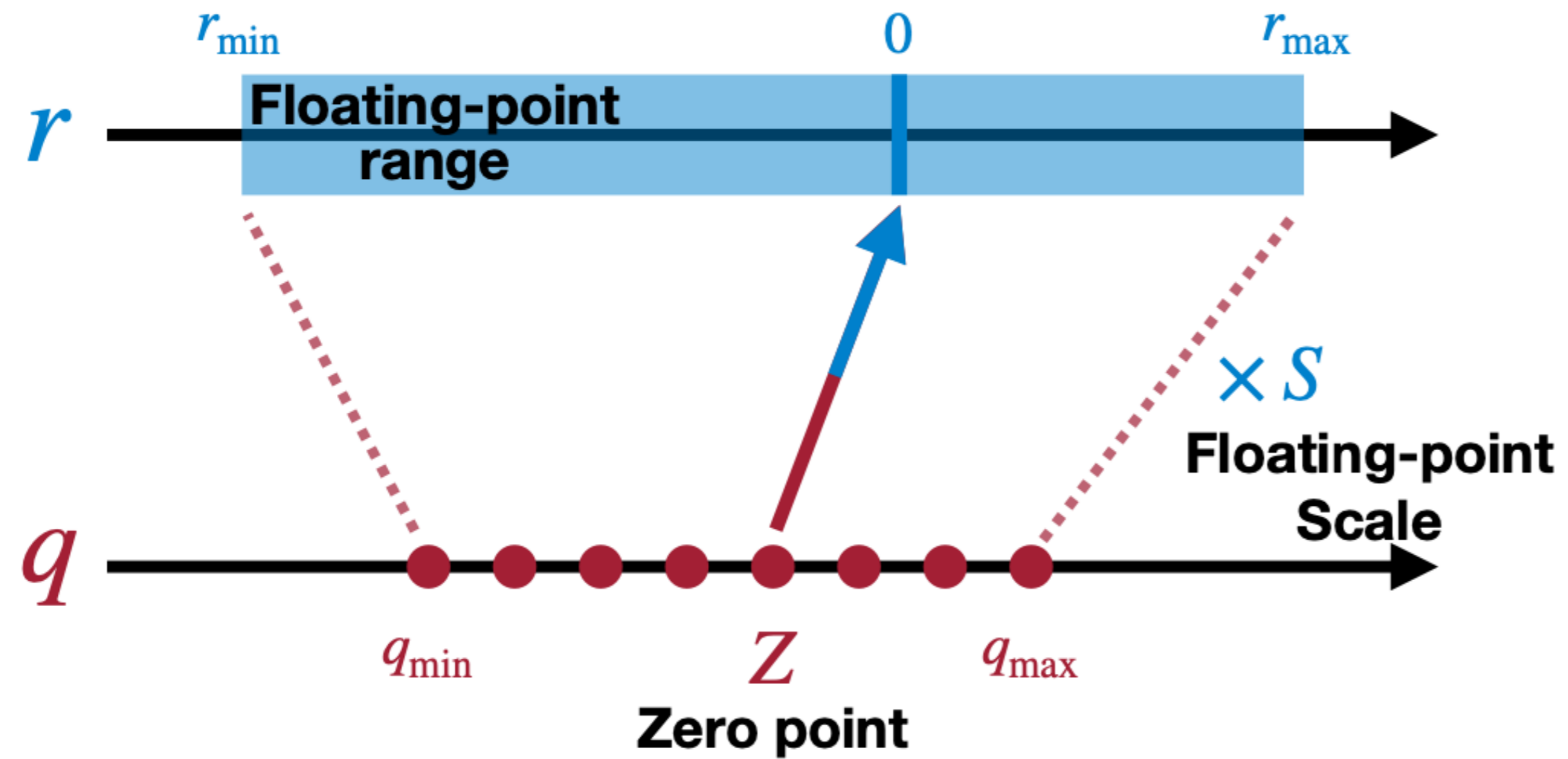


$$r_{\min} = S(q_{\min} - Z)$$

$$Z = q_{\min} - \frac{r_{\min}}{S}$$

$$Z = \text{round}\left(q_{\min} - \frac{r_{\min}}{S}\right)$$

$r = S(q - Z)$: Determine S and Z



2.09	-0.98	1.48	0.09
0.05	-0.14	-1.08	2.12
-0.91	1.92	0	-1.03
1.87	0	1.53	1.49

$$Z = \text{round}\left(q_{min} - \frac{r_{min}}{S}\right)$$

$$= \text{round}\left(-2 - \frac{-1.08}{1.07}\right) = -1$$

Binary	Decimal
01	1
00	0
11	-1
10	-2

