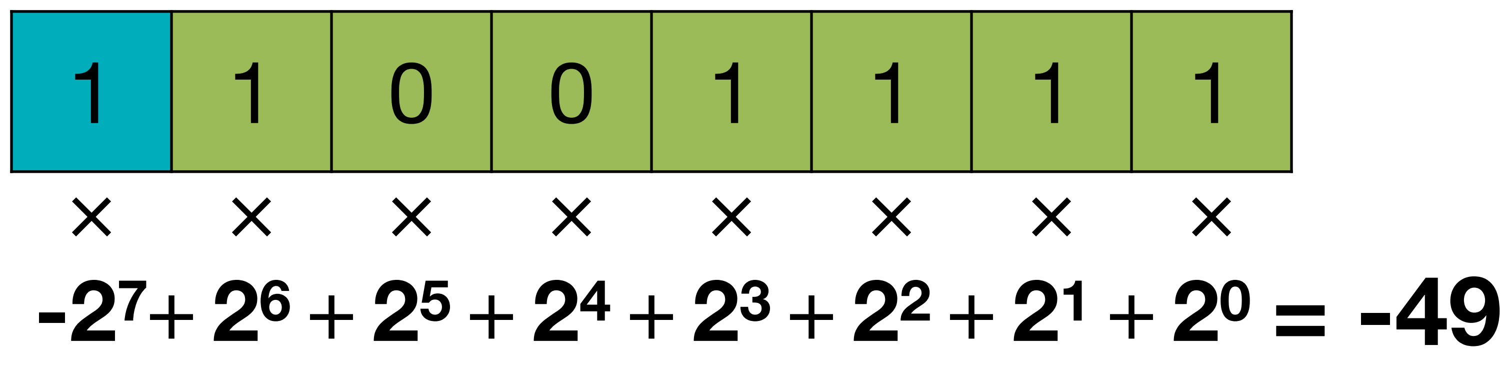
Lecture 05
Quantization
Part I

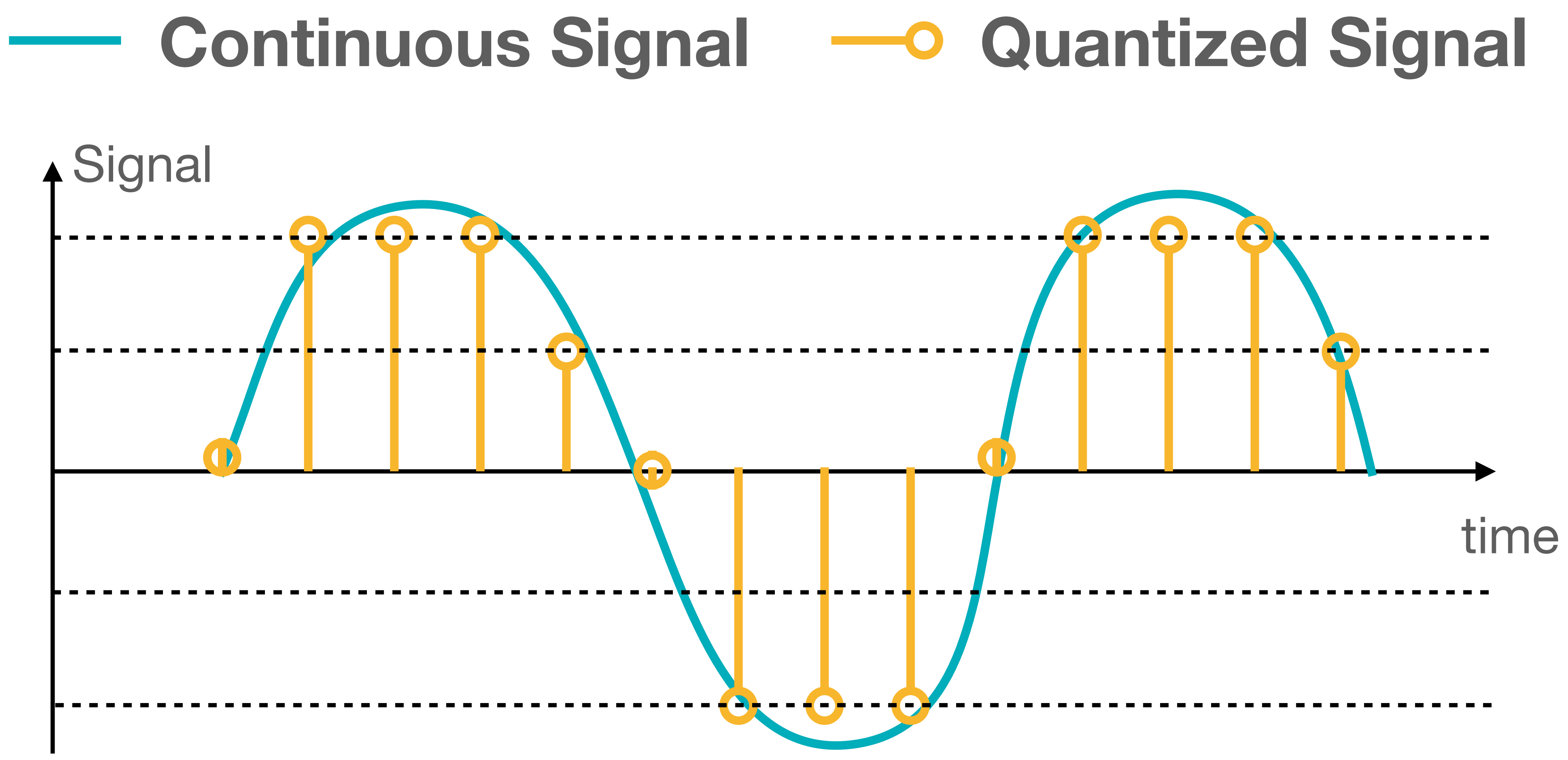
Song Han
songhan@mit.edu

Lecture Plan

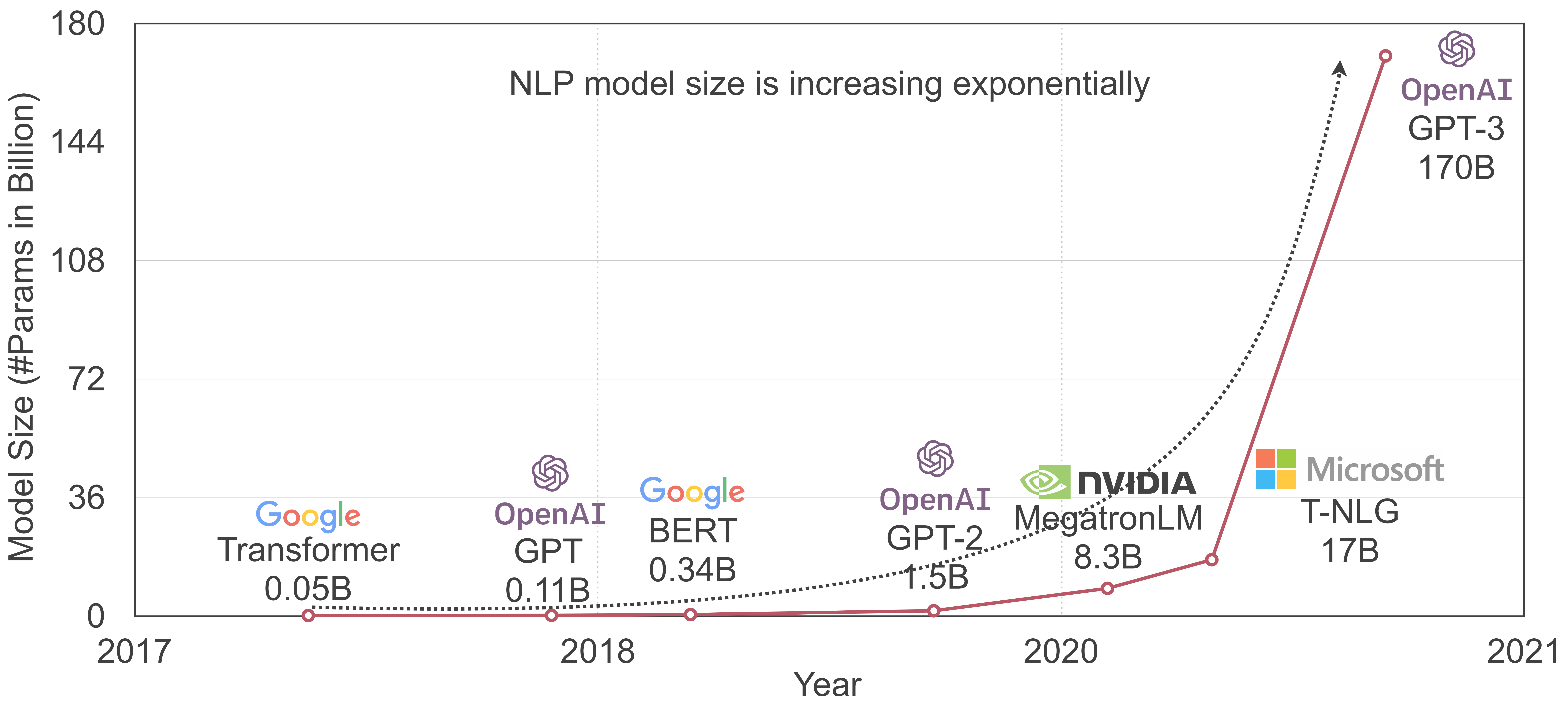
Today we will:

1. Review the numeric ***data types*** used in the modern computing systems, including integers and floating-point numbers.
2. Learn the basic concept of ***neural network quantization***
3. Learn three types of common neural network quantization:
   1. K-Means-based Quantization
   2. Linear Quantization
   3. Binary and Ternary Quantization

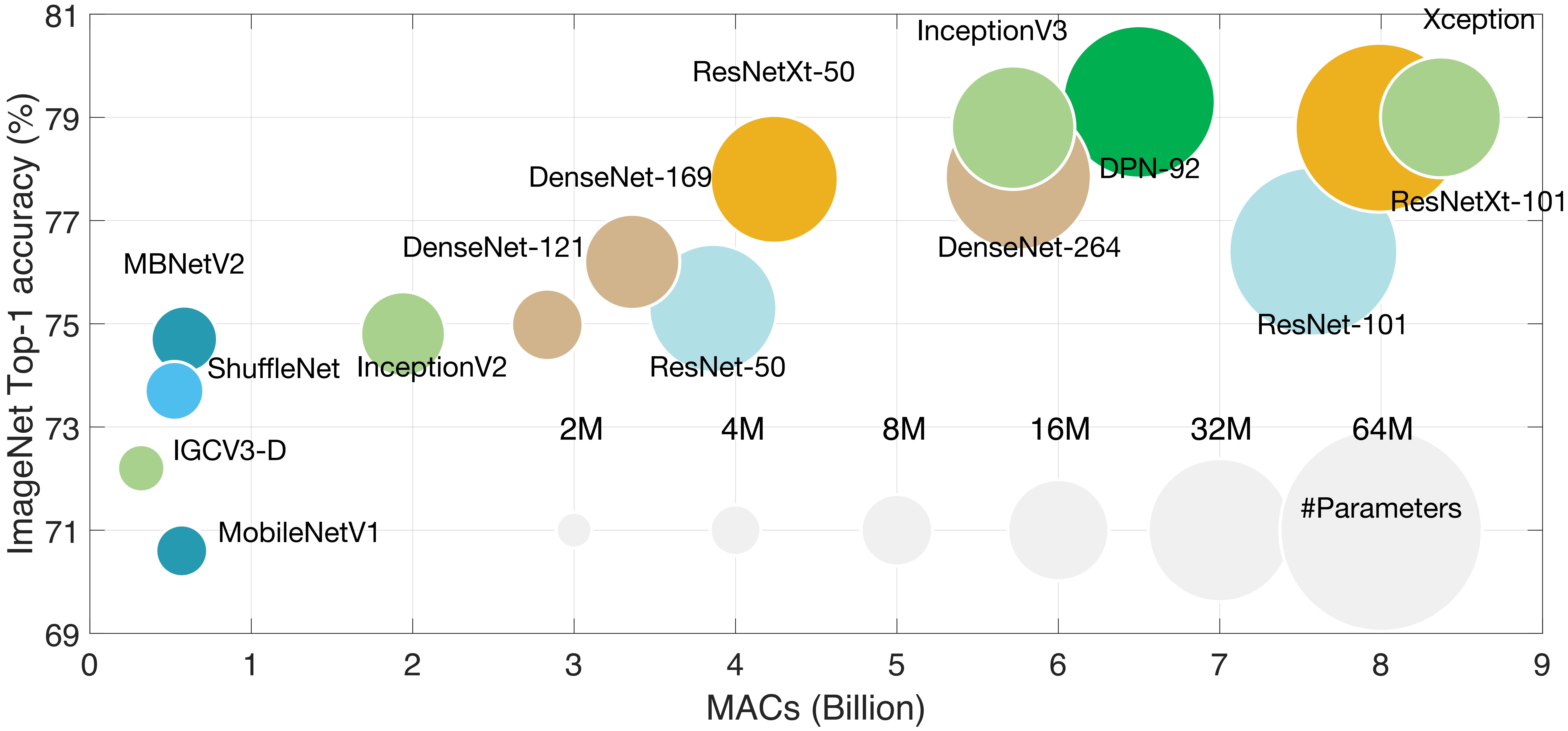




Today's AI is too BIG!



Today's AI is too BIG!



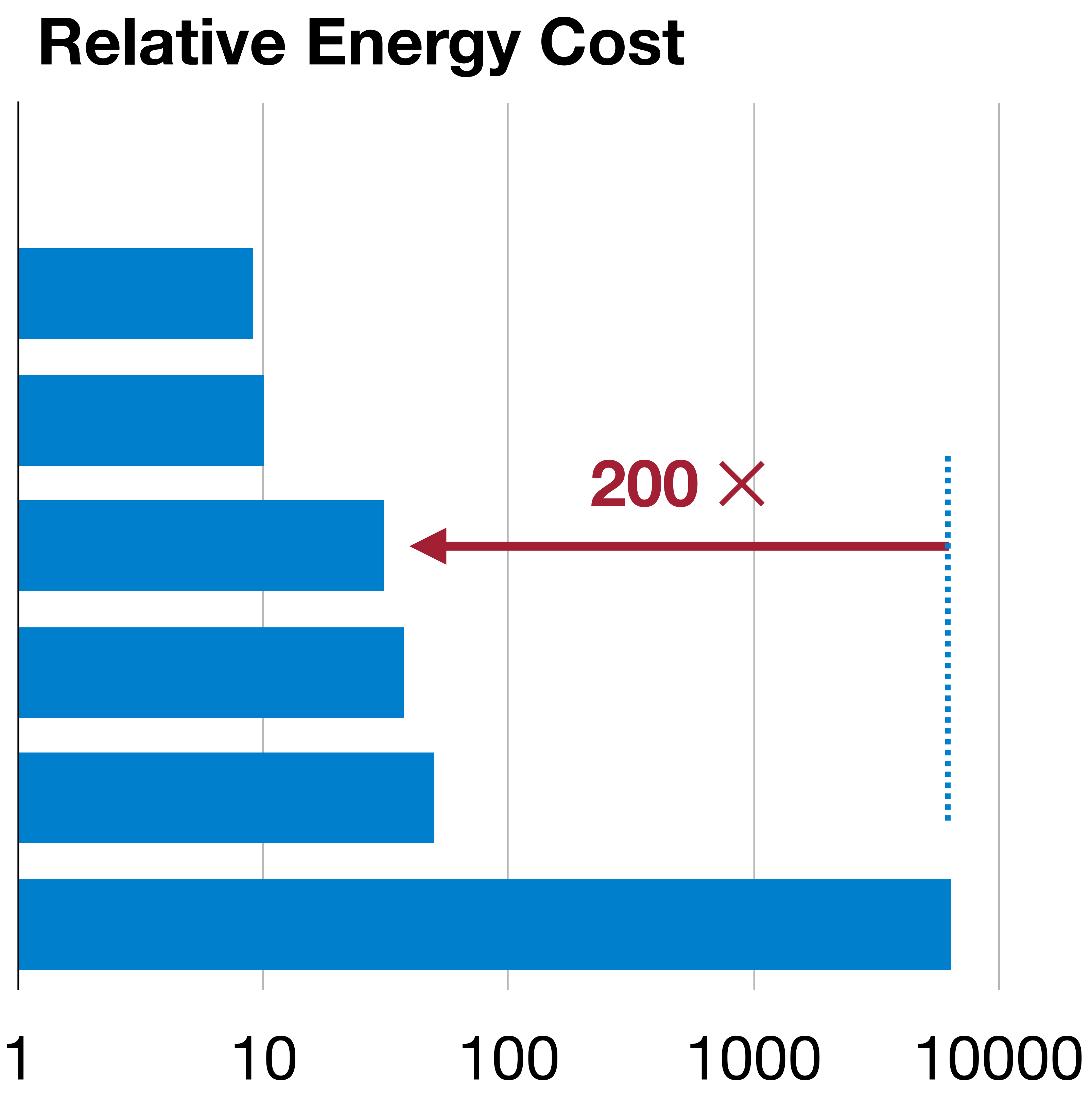
Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey [Deng *et al.*, IEEE 2020]

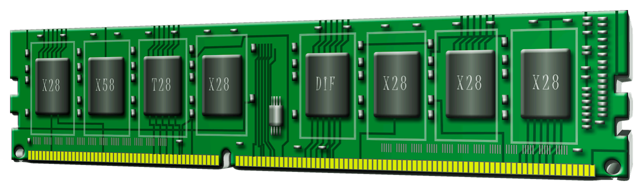
Memory is Expensive

Data Movement → More Memory Reference → More Energy

Operation	Energy [pJ]
32 bit int ADD	0.1
32 bit float ADD	0.9
32 bit Register File	1
32 bit int MULT	3.1
32 bit float MULT	3.7
32 bit SRAM Cache	5
32 bit DRAM Memory	640

Rough Energy Cost For Various Operations in 45nm 0.9V



1  = 200 ×+

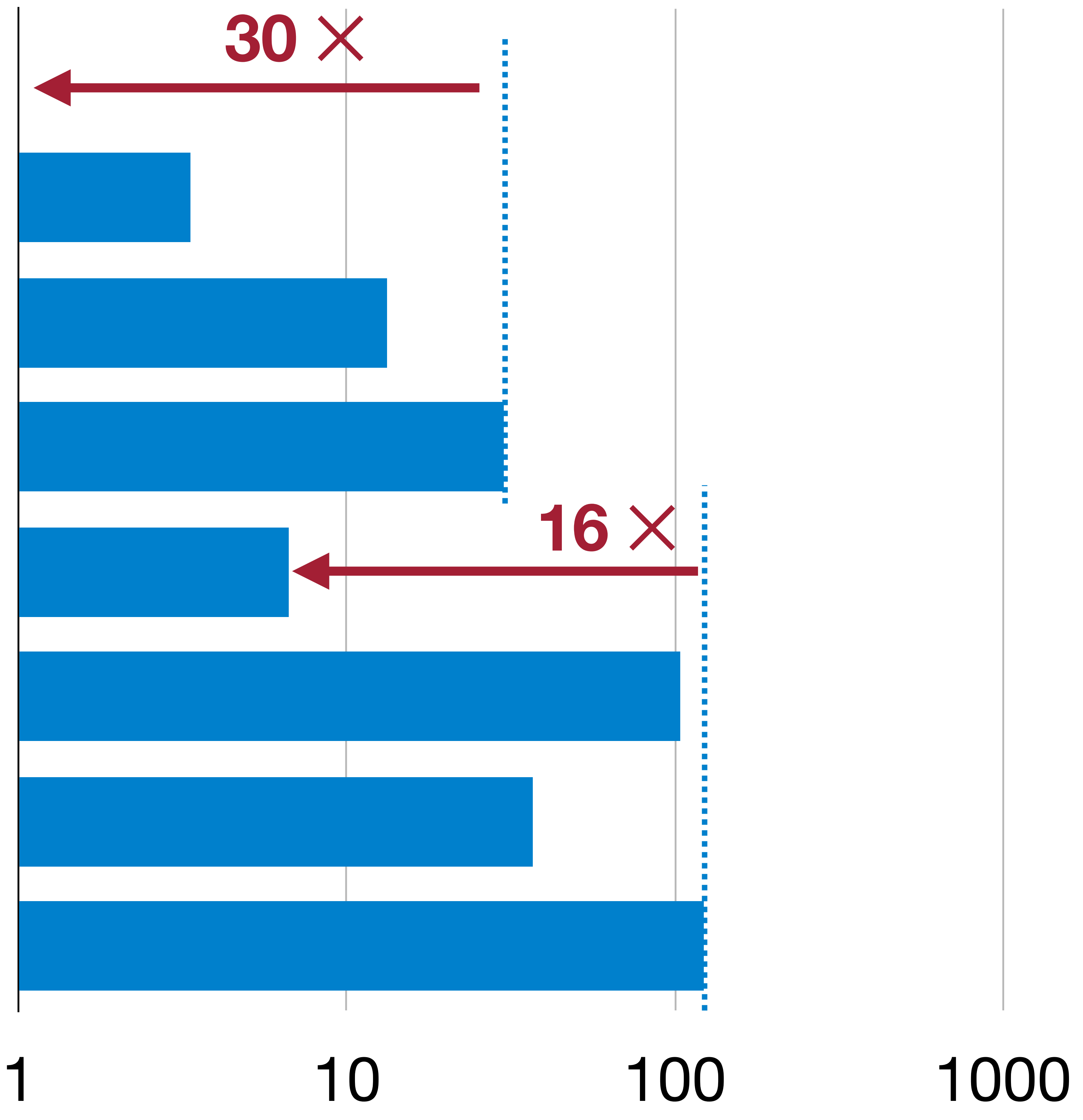
This image is in the public domain

Computing's Energy Problem (and What We Can Do About it) [Horowitz, M., IEEE ISSCC 2014]

Low Bit-Width Operations are Cheap

Less Bit-Width → Less Energy

Operation	Energy [pJ]
8 bit int ADD	0.03
32 bit int ADD	0.1
16 bit float ADD	0.4
32 bit float ADD	0.9
8 bit int MULT	0.2
32 bit int MULT	3.1
16 bit float MULT	1.1
32 bit float MULT	3.7



Rough Energy Cost For Various Operations in 45nm 0.9V

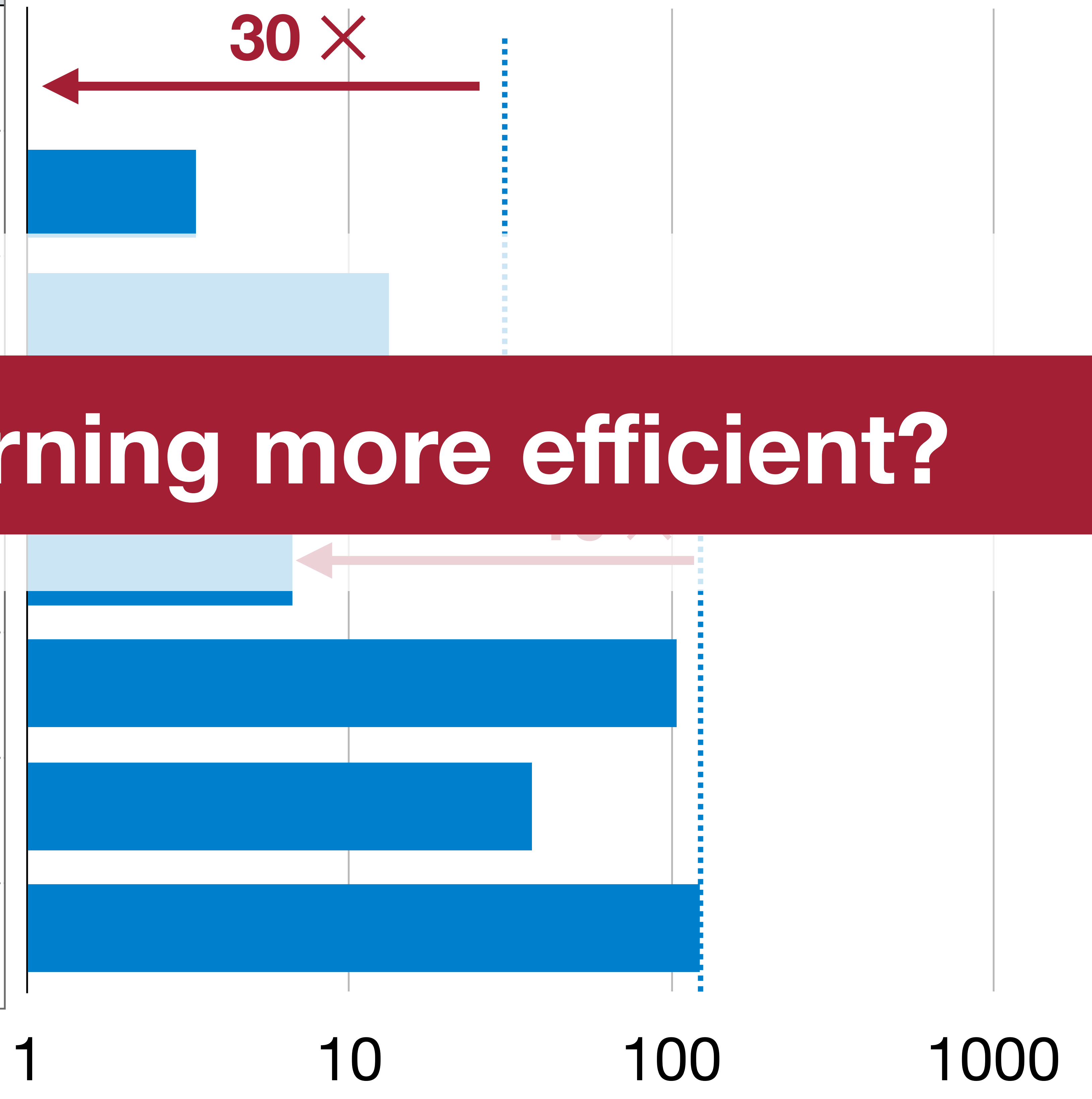
Computing's Energy Problem (and What We Can Do About it) [Horowitz, M., IEEE ISSCC 2014]

Low Bit-Width Operations are Cheap

Less Bit-Width → Less Energy

Operation	Energy [pJ]
8 bit int ADD	0.03
32 bit int ADD	0.1
16 bit float ADD	0.4
8 bit int MULT	0.2
32 bit int MULT	3.1
16 bit float MULT	1.1
32 bit float MULT	3.7

Rough Energy Cost For Various Operations in 45nm 0.9V

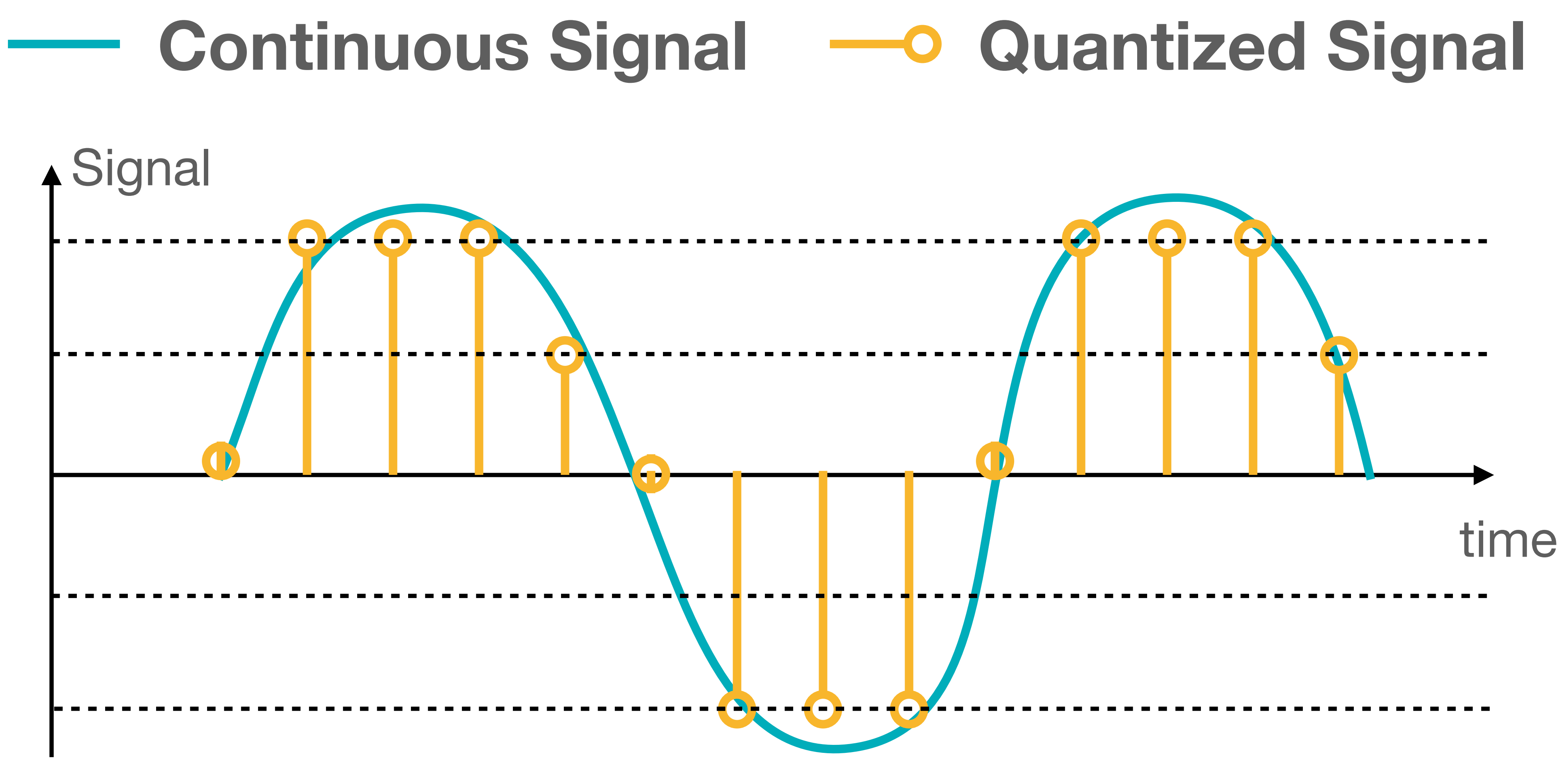


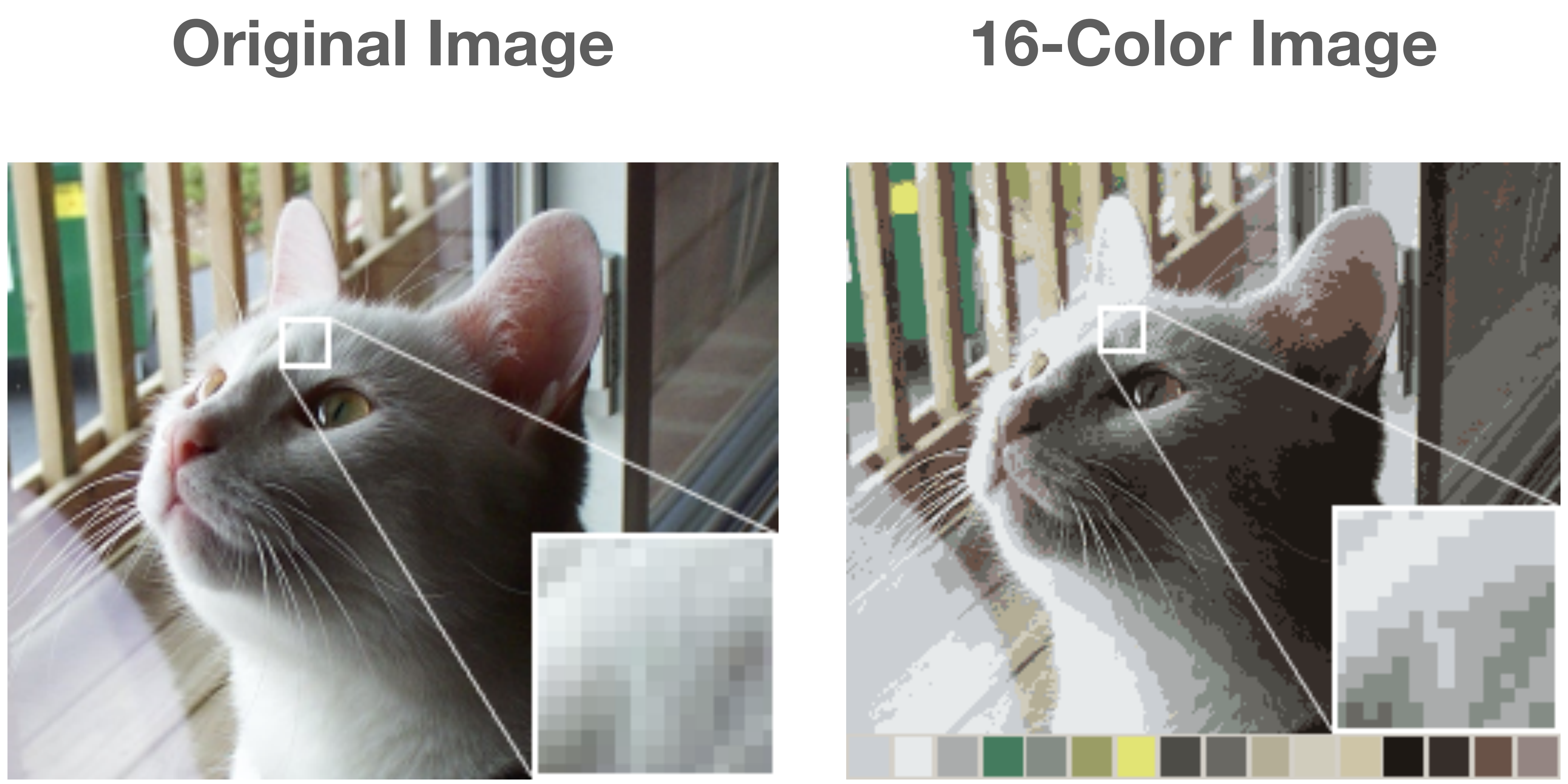
How should we make deep learning more efficient?

Computing's Energy Problem (and What We Can Do About it) [Horowitz, M., IEEE ISSCC 2014]

What is Quantization?

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





Images are in the public domain.

Wikipedia: Quantization

Numeric Data Types

How is numeric data represented in modern computing systems?

Integer

- Unsigned Integer
  - n-bit Range: $[0,\ 2^n - 1]$
- Signed Integer
  - Sign-Magnitude Representation
    - n-bit Range: $[-2^{n-1} - 1,\ 2^{n-1} - 1]$
    - Both 000…00 and 100…00 represent 0
  - Two's Complement Representation
    - n-bit Range: $[-2^{n-1},\ 2^{n-1} - 1]$
    - 000…00 represents 0
    - 100…00 represents -2^{n-1}

0	0	1	1	0	0	0	1
×	×	×	×	×	×	×	×
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

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

Sign Bit

1	0	1	1	0	0	0	1
	×	×	×	×	×	×	×
-	2^6	2^5	2^4	2^3	2^2	2^1	2^0

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

1	1	0	0	1	1	1	1
×	×	×	×	×	×	×	×
-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

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

Fixed-Point Number

Integer . Fraction

"Decimal" Point

0	0	1	1	0	0	0	1
×	×	×	×	×	×	×	×
-2^3	$+2^2$	$+2^1$	$+2^0$	$+2^{-1}$	$+2^{-2}$	$+2^{-3}$	$+2^{-4}$

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

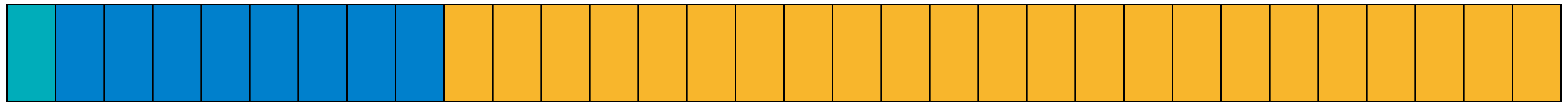
0	0	1	1	0	0	0	1
×	×	×	×	×	×	×	×
-2^7	$+2^6$	$+2^5$	$+2^4$	$+2^3$	$+2^2$	$+2^1$	$+2^0$

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

(using 2's complement representation)

Floating-Point Number

Example: 32-bit floating-point number in IEEE 754



Sign 8 bit Exponent 23 bit Fraction

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

(significant / mantissa)

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

$$0\ \underbrace{0\ 1\ 1\ 1\ 1\ 1\ 0\ 1}_{125}\ \underbrace{0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0}_{0.0625}$$

Floating-Point Number

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 Half Precision 16-bit Float (IEEE FP16)	5	10	16
Brain Float (BF16)	8	7	16
Nvidia TensorFloat (TF32)	8	10	19
AMD 24-bit Float (AMD FP24)	7	16	24

Numeric Data Types

- **Question**: What is the following IEEE half precision (IEEE FP16) number in decimal?

Sign	5 bit Exponent					10 bit Fraction									
1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0

Exponent Bias = 15_{10}

- Sign: -
- Exponent: $10001_2 - 15_{10} = 17_{10} - 15_{10} = 2_{10}$
- Fraction: $1100000000_2 = 0.75_{10}$
- Decimal Answer = $-(1 + 0.75) \times 2^2 = -1.75 \times 2^2 = -7.0_{10}$

Numeric Data Types

- **Question**: What is the decimal 2.5 in Brain Float (BF16)?

$$2.5_{10} = 1.\underline{25}_{10} \times 2^{\underline{1}}$$

Exponent Bias = 127_{10}

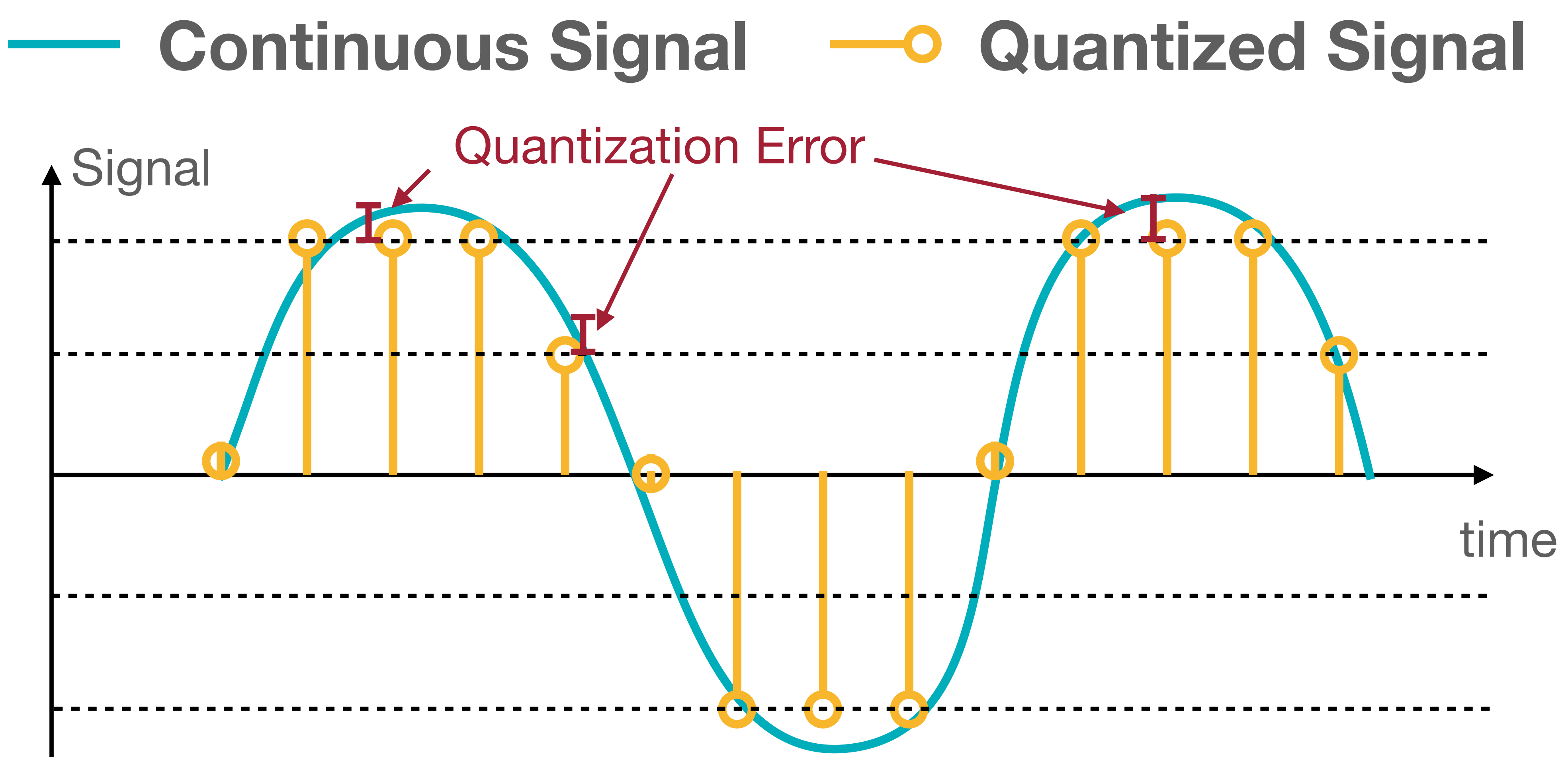
- Sign: +
- Exponent Binary: $1_{10} + 127_{10} = 128_{10} = 10000000_2$
- Fraction Binary: $0.25_{10} = 0100000_2$
- Binary Answer

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

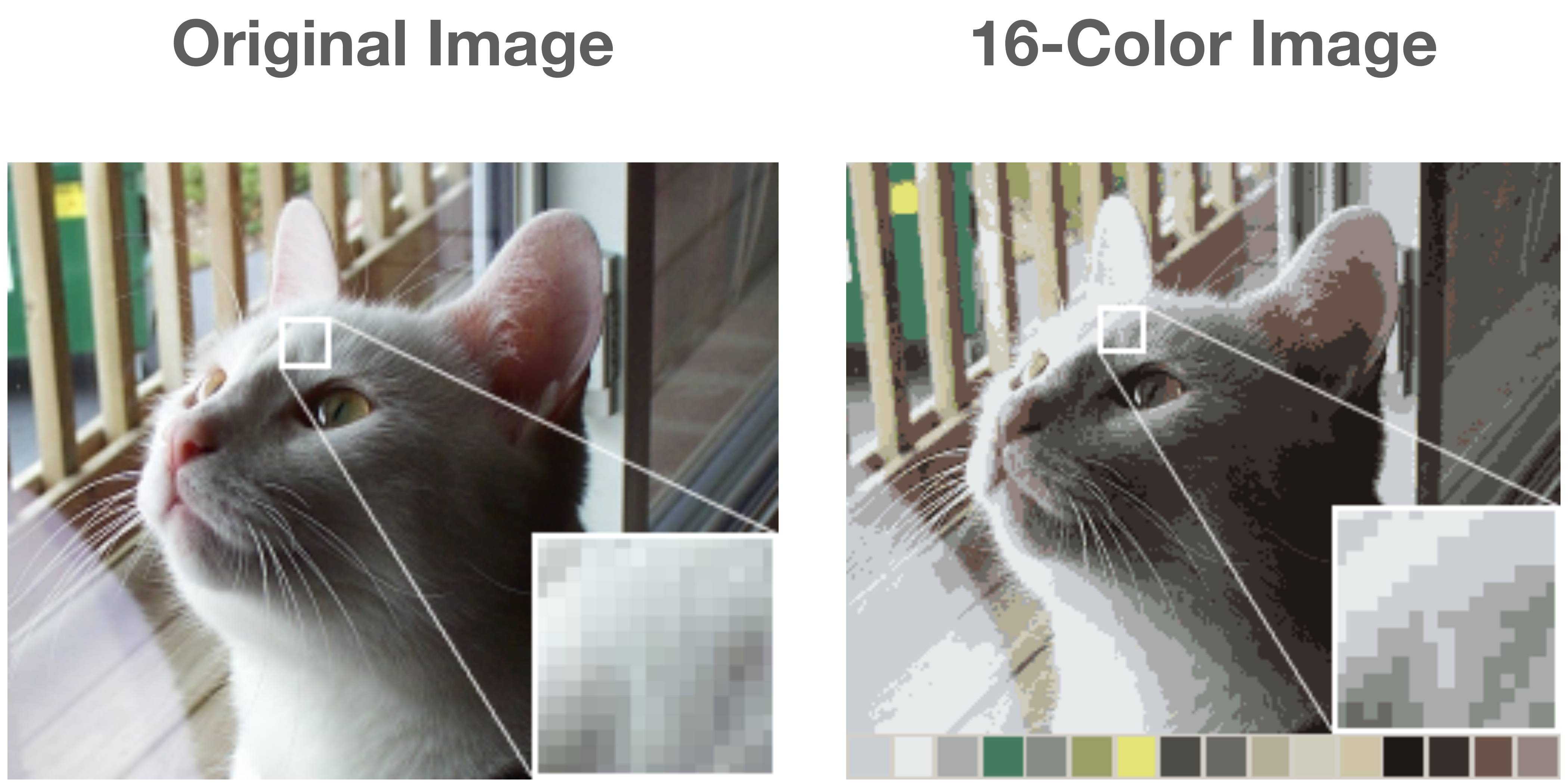
Sign 8 bit Exponent 7 bit Fraction

What is Quantization?

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



The difference between an input value and its quantized value is referred to as quantization error.



Images are in the public domain.

"Palettization"

Quantization [Wikipedia]

Neural Network Quantization: Agenda

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)×1.07

Linear Quantization

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

Binary/Ternary Quantization

Storage	Floating-Point Weights	Integer Weights; Floating-Point Codebook		
Computation	Floating-Point Arithmetic	Floating-Point Arithmetic		

Neural Network Quantization

Weight 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

K-Means-based Weight 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

cluster ⇨

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 32 bit × 16 = 512 bit = 64 B 2 bit × 16 = 32 bit = 4 B **+** 32 bit × 4 = 128 bit = 16 B **=** 20 B

3.2 × smaller

Assume N-bit quantization, and #parameters = M >> 2^N.

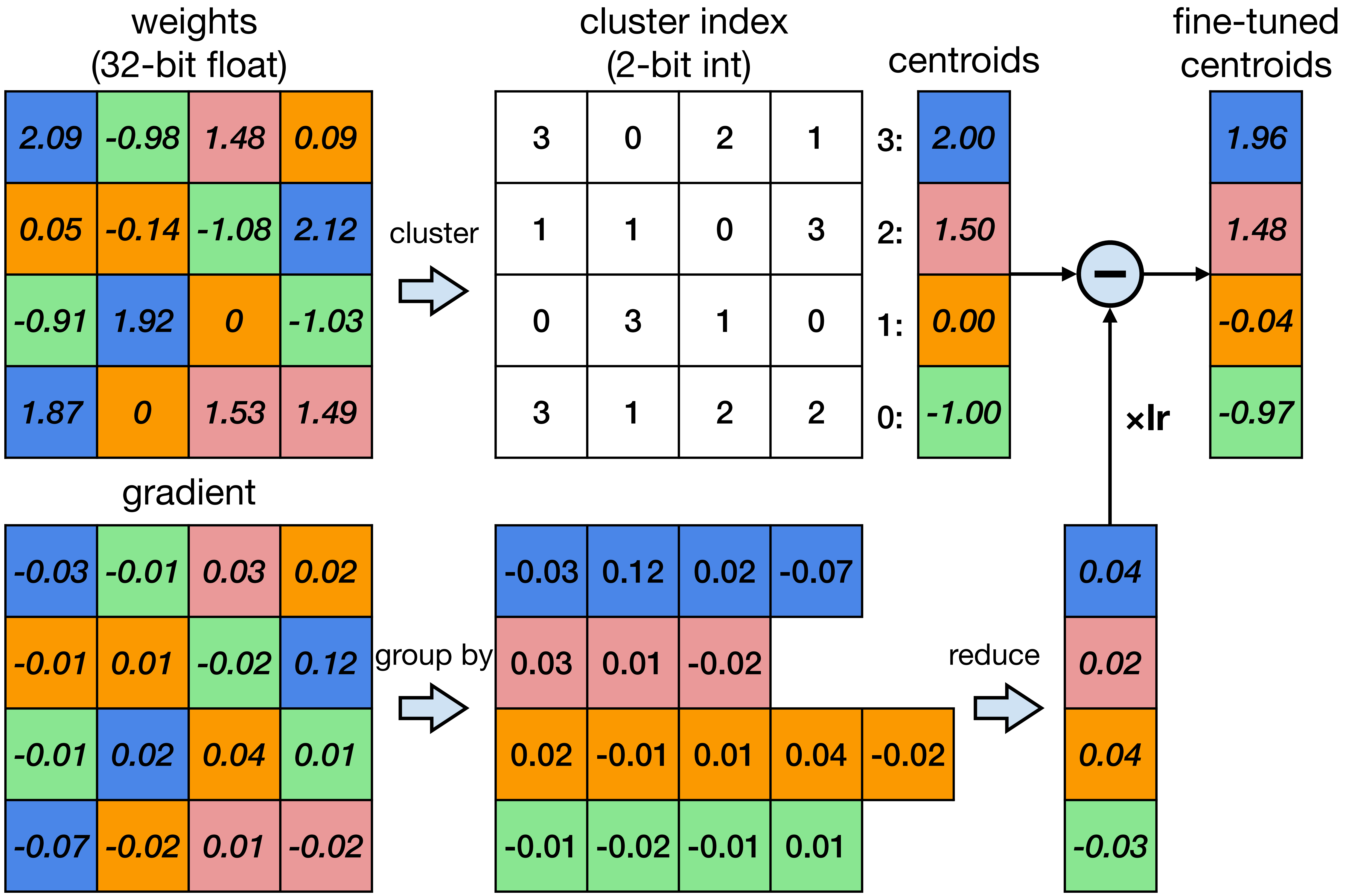
32 bit × M = $32M$ bit N bit × M = NM bit ~~32 bit × 2^N = 2^{N+5} bit~~

$32/N$ × smaller

Deep Compression [Han *et al.*, ICLR 2016]

K-Means-based Weight Quantization

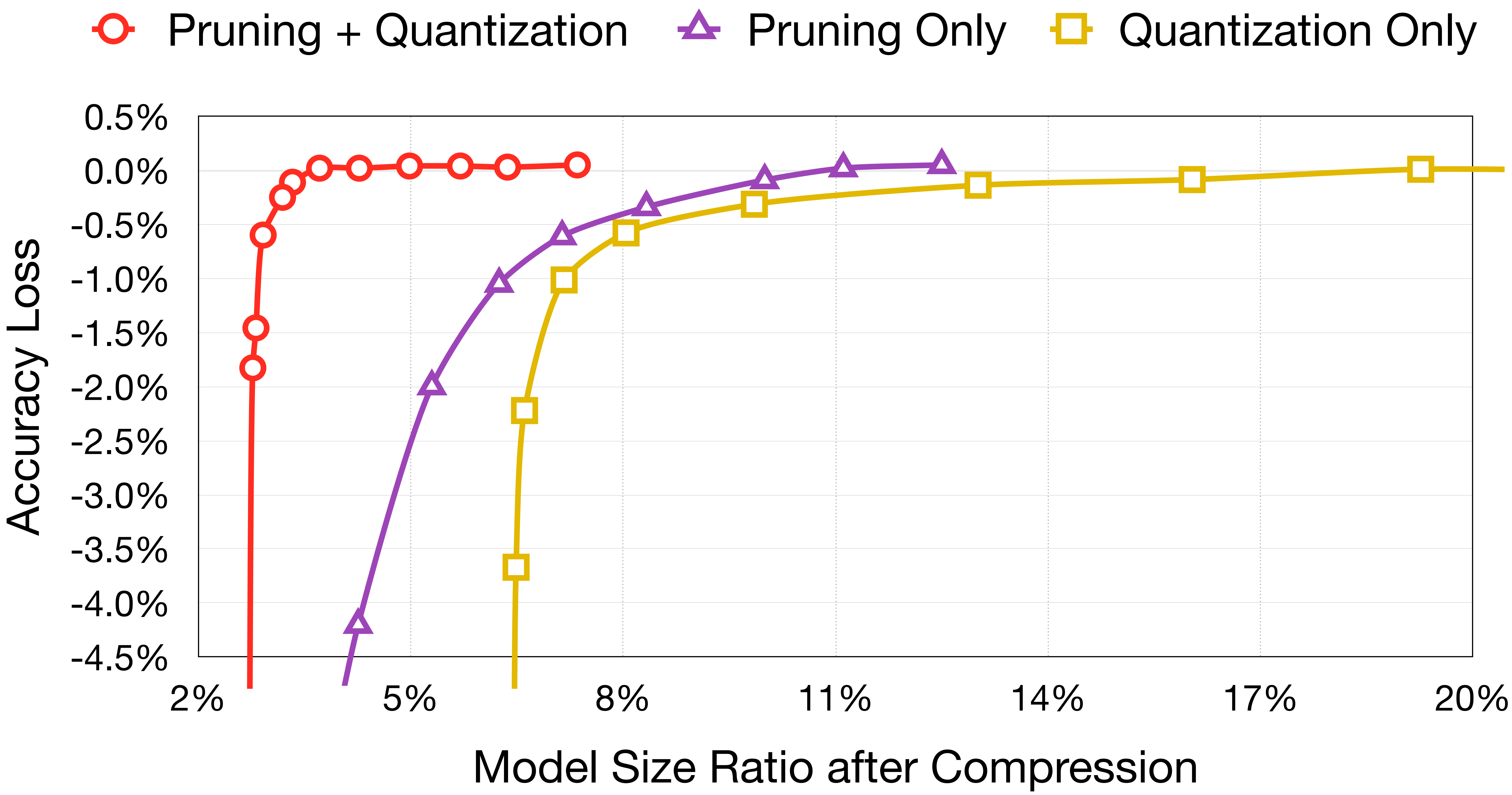
Fine-tuning Quantized Weights



Deep Compression [Han *et al.*, ICLR 2016]

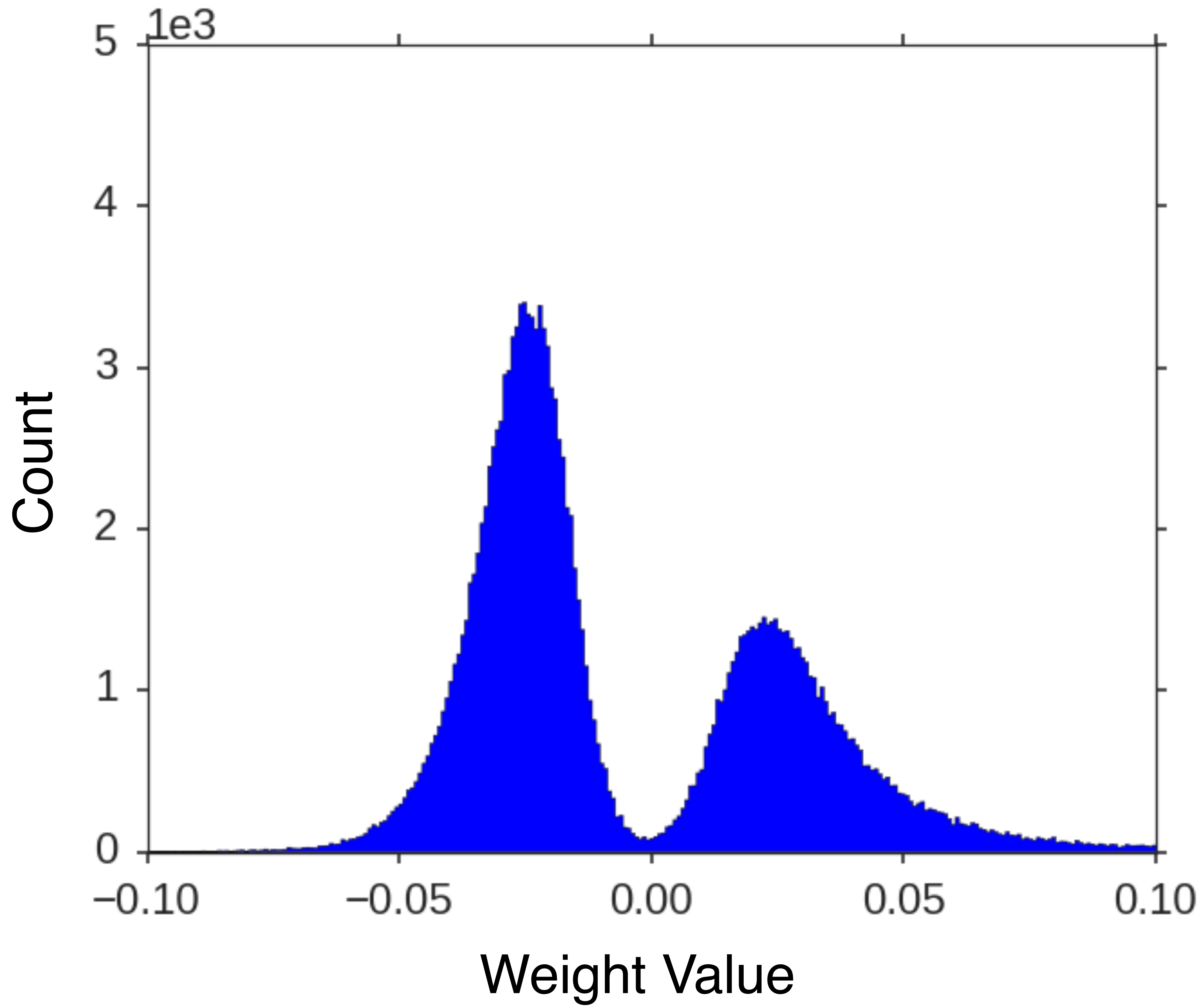
K-Means-based Weight Quantization

Accuracy vs. compression rate for AlexNet on ImageNet dataset



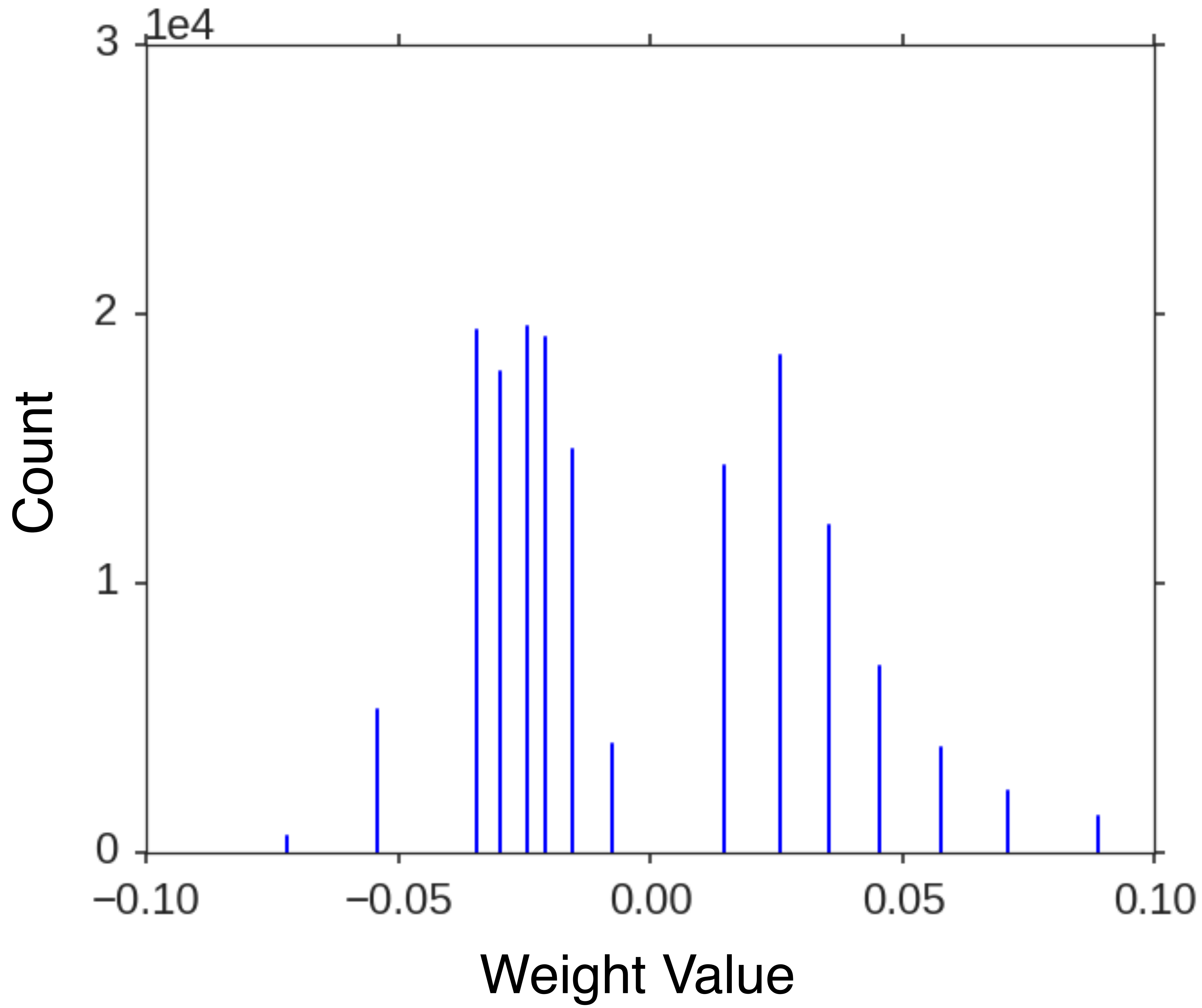
Deep Compression [Han *et al.*, ICLR 2016]

Before Quantization: Continuous Weight



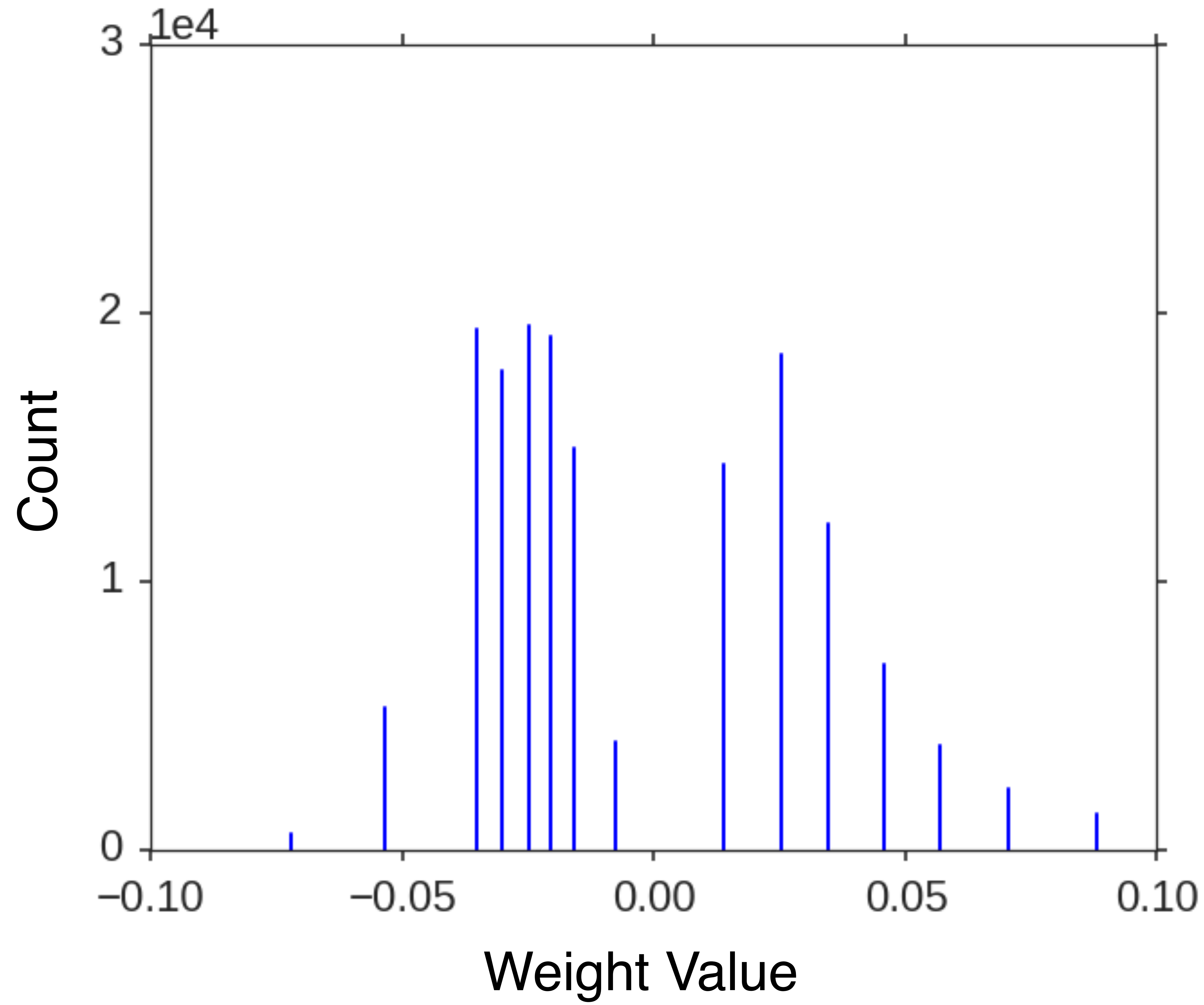
Deep Compression [Han *et al.*, ICLR 2016]

After Quantization: Discrete Weight



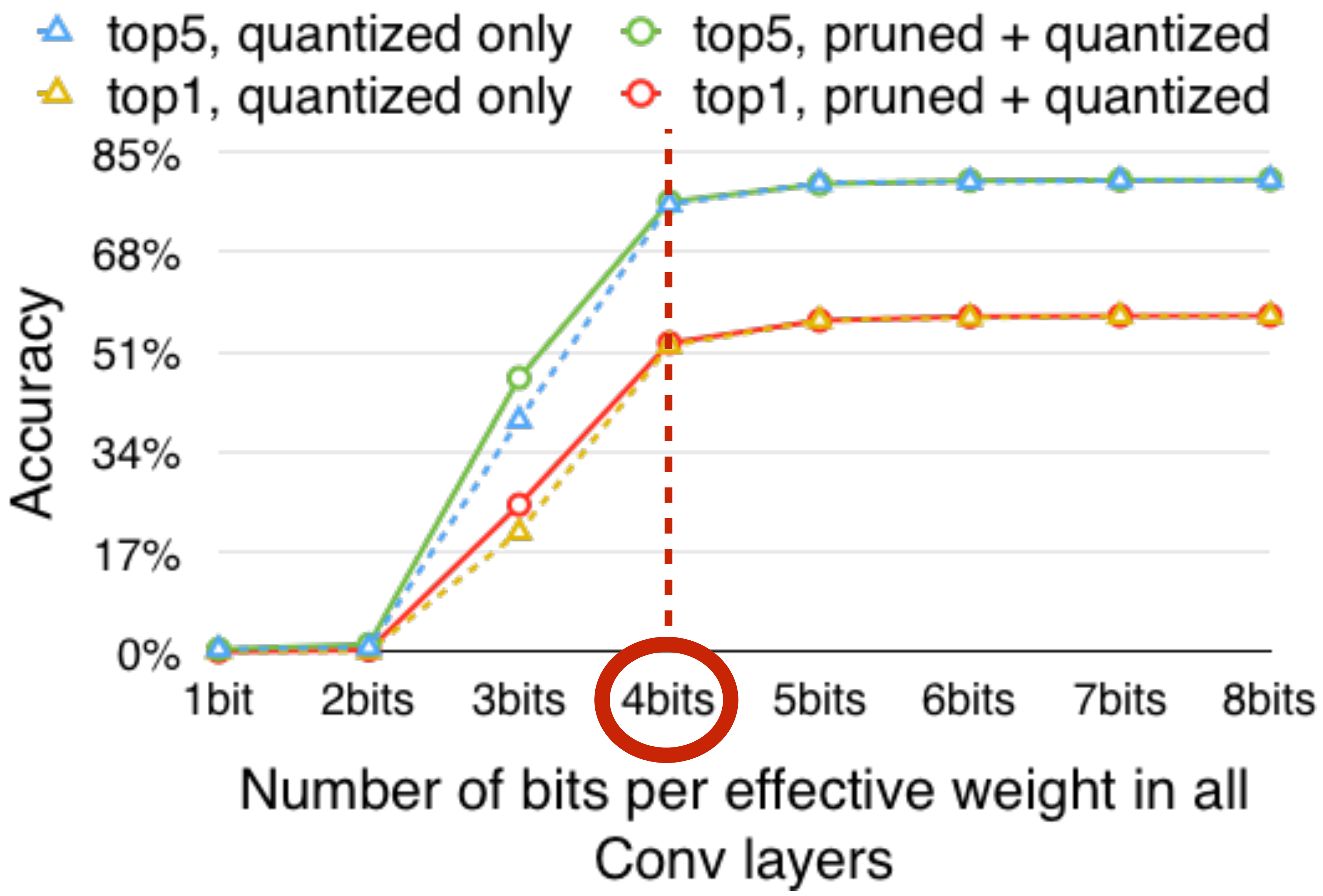
Deep Compression [Han *et al.*, ICLR 2016]

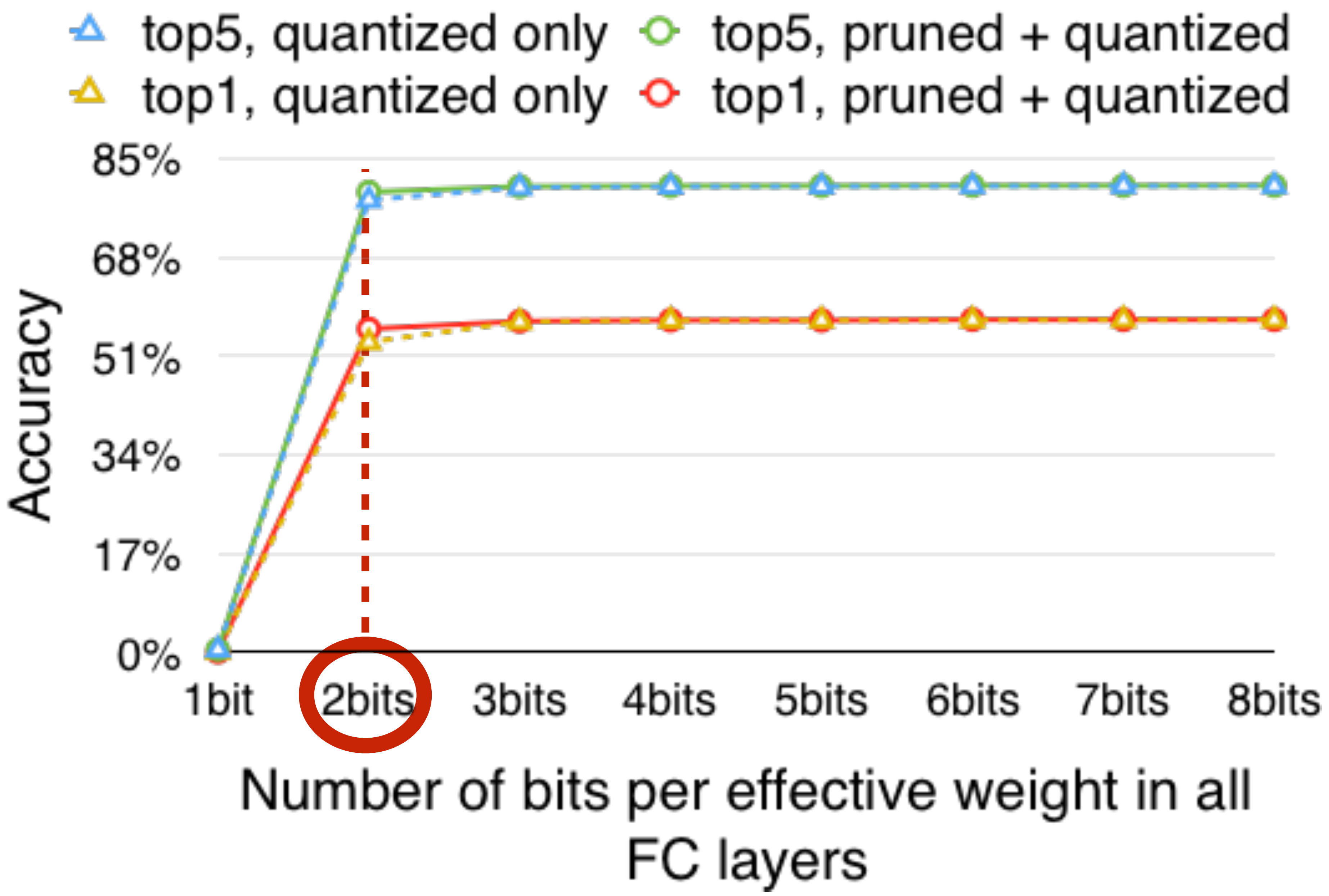
After Quantization: Discrete Weight after Training



Deep Compression [Han *et al.*, ICLR 2016]

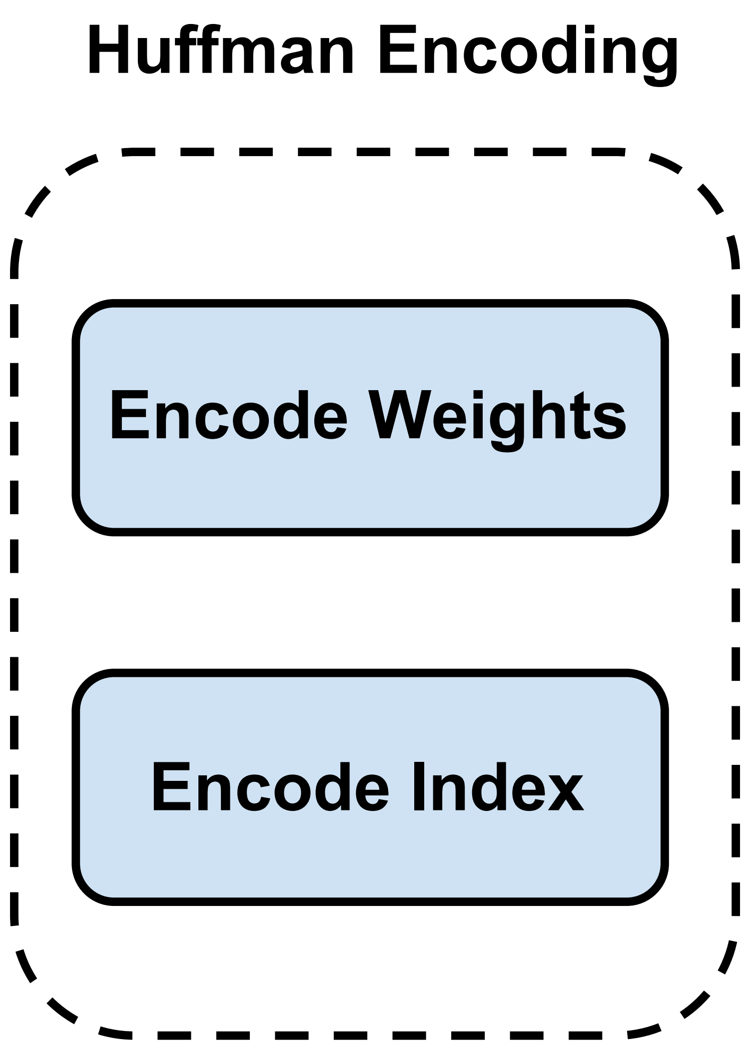
How Many Bits do We Need?

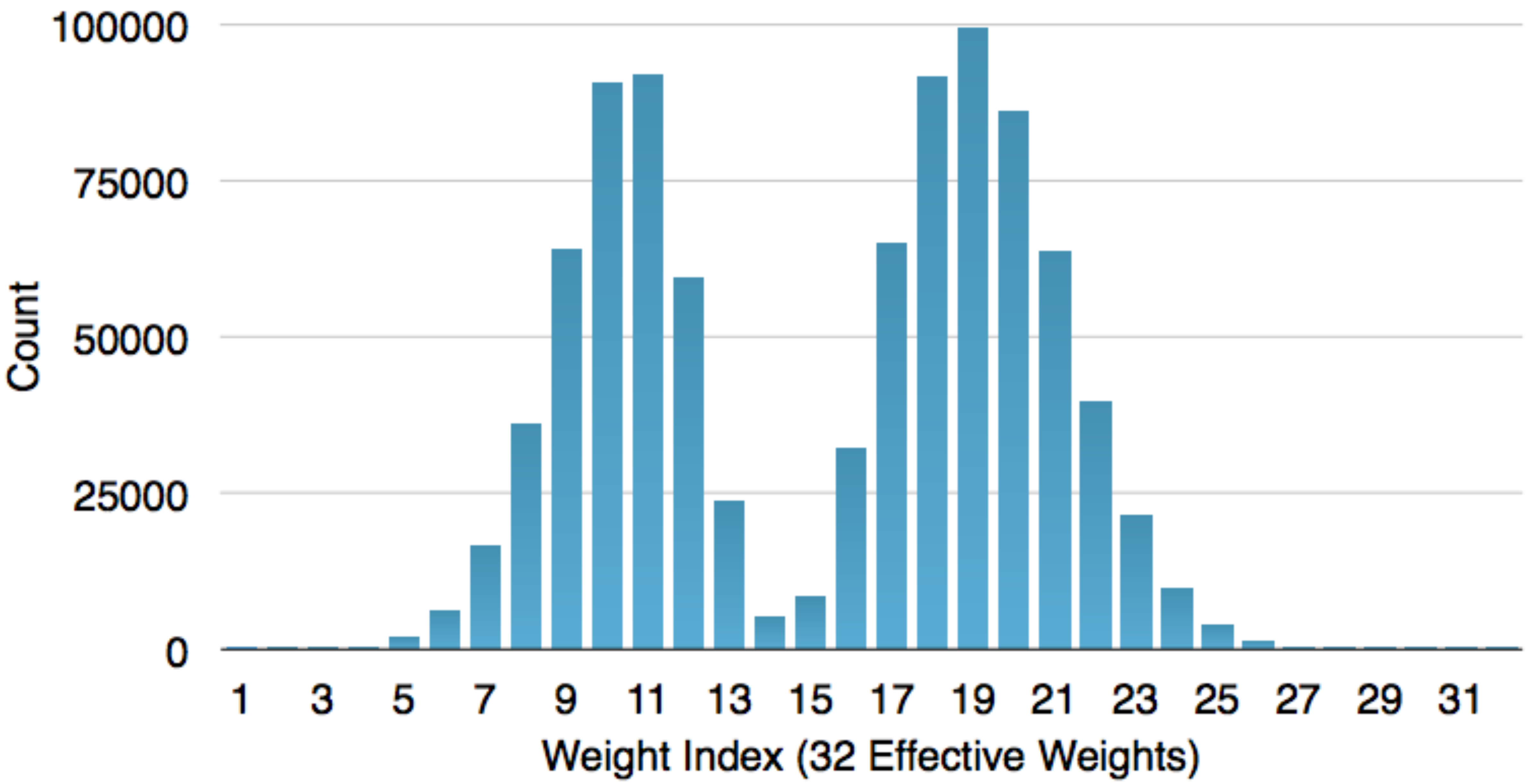




Deep Compression [Han *et al.*, ICLR 2016]

Huffman Coding

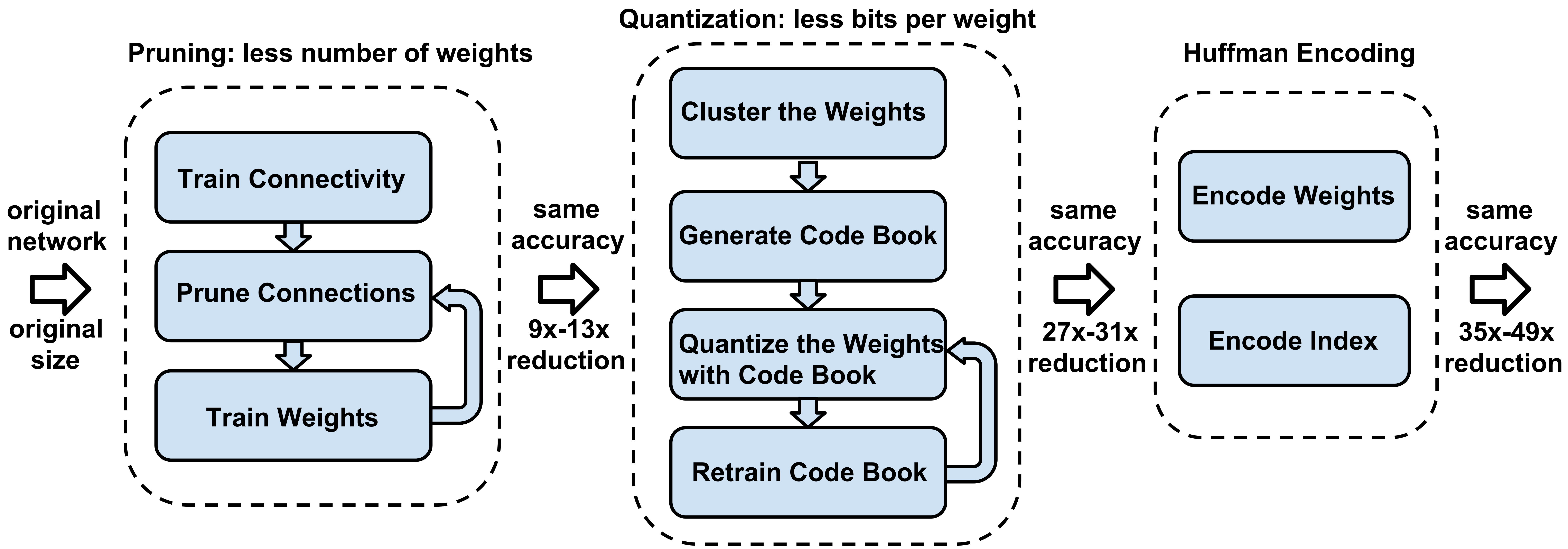




- In-frequent weights: use more bits to represent
- Frequent weights: use less bits to represent

Deep Compression [Han *et al.*, ICLR 2016]

Summary of Deep Compression



Deep Compression [Han *et al.*, ICLR 2016]

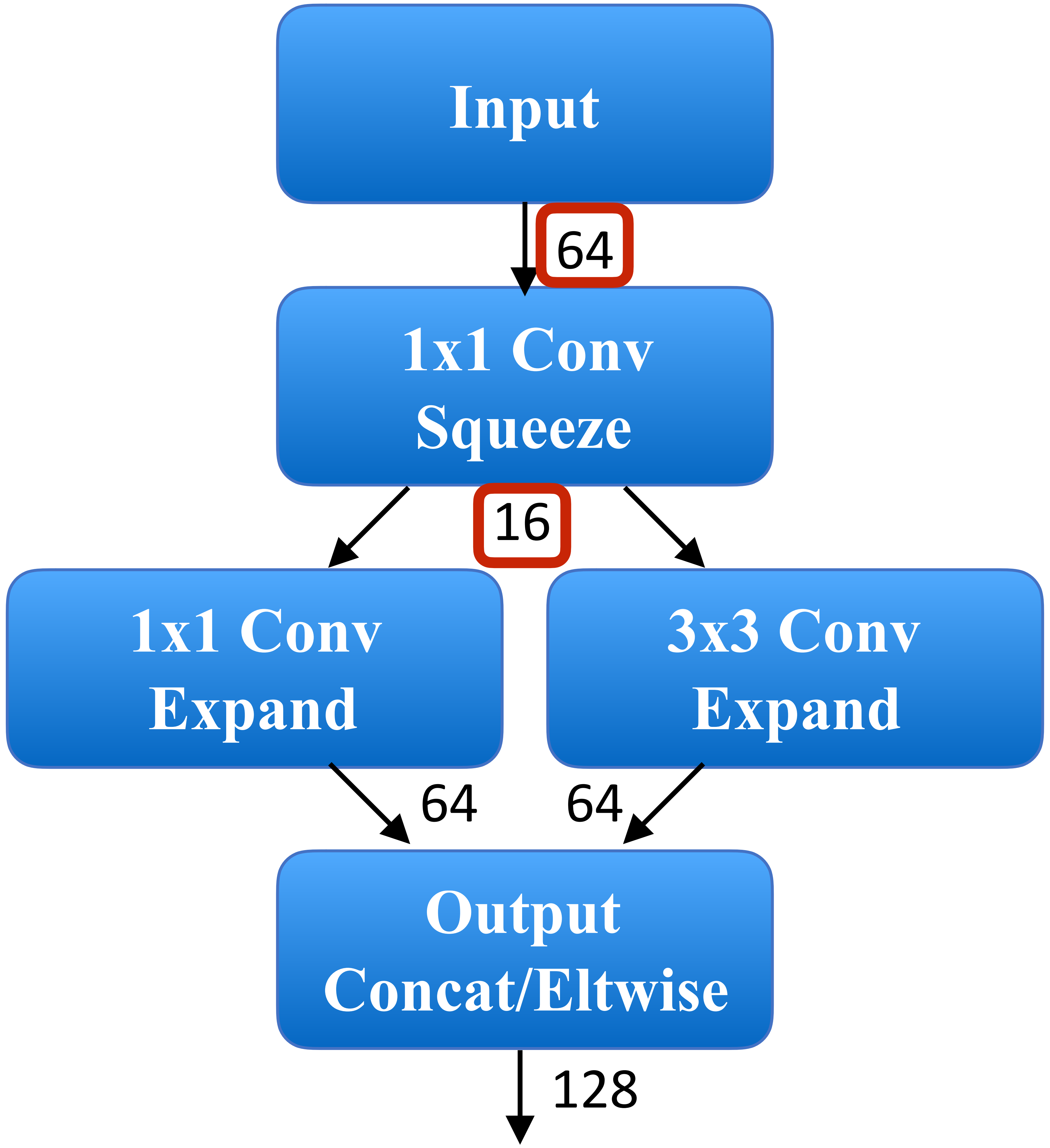
Deep Compression Results

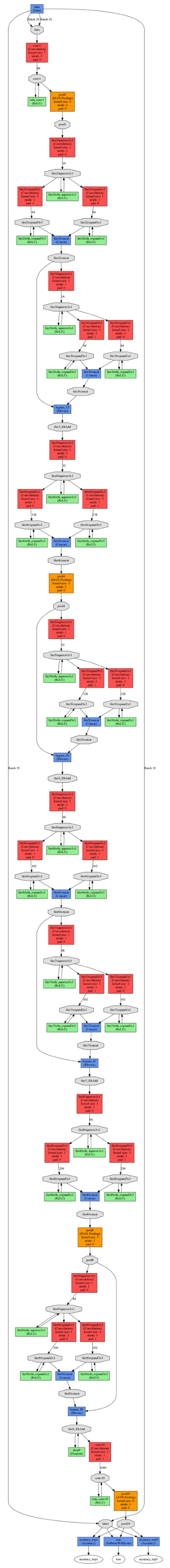
Network	Original Size	Compressed Size	Compression Ratio	Original Accuracy	Compressed Accuracy
LeNet-300	1070KB	27KB	**40x**	98.36%	98.42%
LeNet-5	1720KB	44KB	**39x**	99.20%	99.26%
AlexNet	240MB	6.9MB	**35x**	80.27%	80.30%
VGGNet	550MB	11.3MB	**49x**	88.68%	89.09%
GoogleNet	28MB	2.8MB	**10x**	88.90%	88.92%
ResNet-18	44.6MB	4.0MB	**11x**	89.24%	89.28%

Can we make compact models to begin with?

Deep Compression [Han *et al.*, ICLR 2016]

SqueezeNet





SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size [Iandola *et al.*, arXiv 2016]

Deep Compression on SqueezeNet

Network	Approach	Size	Ratio	Top-1 Accuracy	Top-5 Accuracy
AlexNet	-	240MB	**1x**	57.2%	80.3%
AlexNet	SVD	48MB	**5x**	56.0%	79.4%
AlexNet	Deep Compression	6.9MB	**35x**	57.2%	80.3%
SqueezeNet	-	4.8MB	**50x**	57.5%	80.3%
SqueezeNet	Deep Compression	0.47MB	**510x**	57.5%	80.3%

SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size [Iandola *et al.*, arXiv 2016]

K-Means-based Weight 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*

decode ⇐

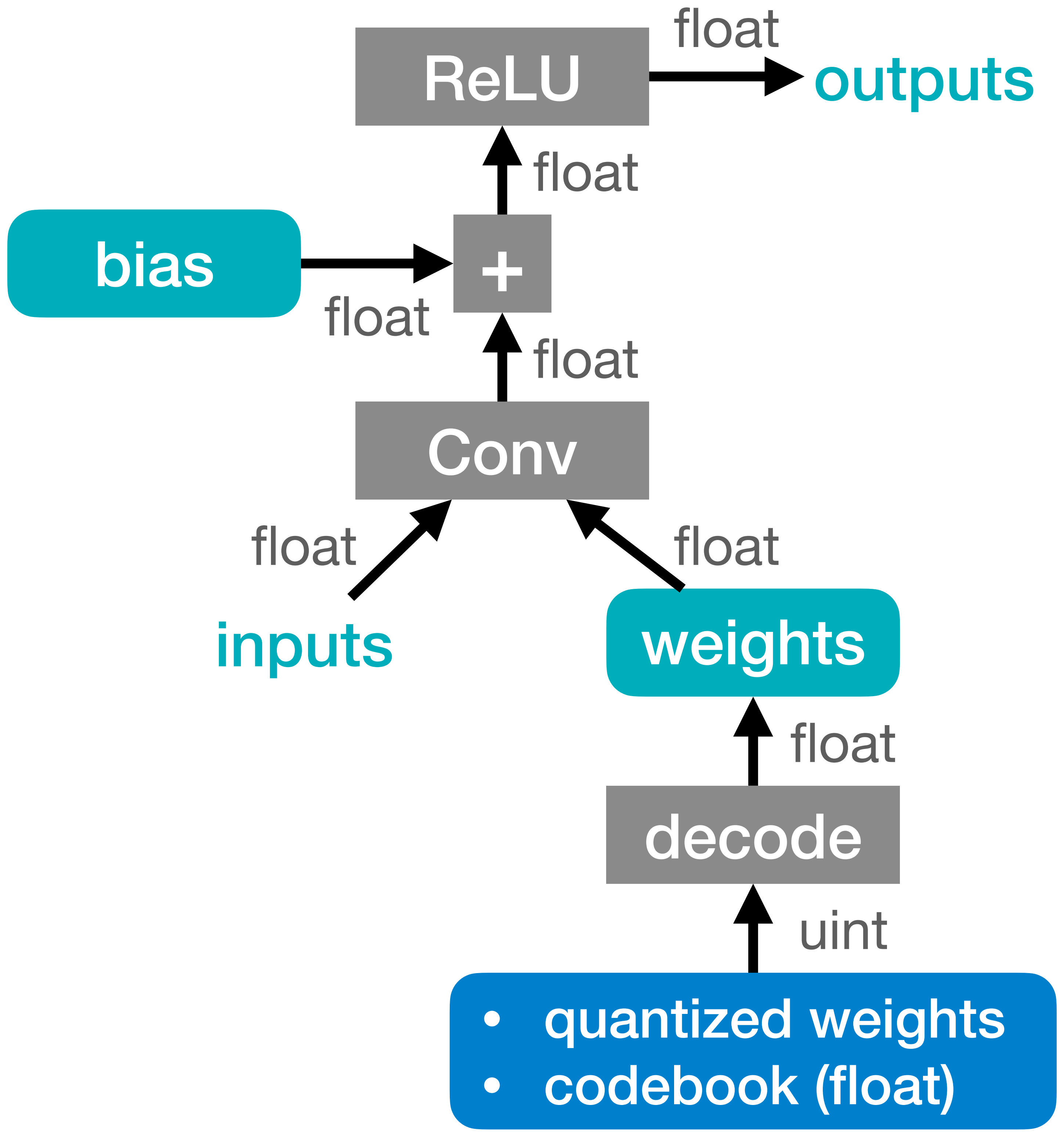
cluster index (2-bit int)

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

centroids

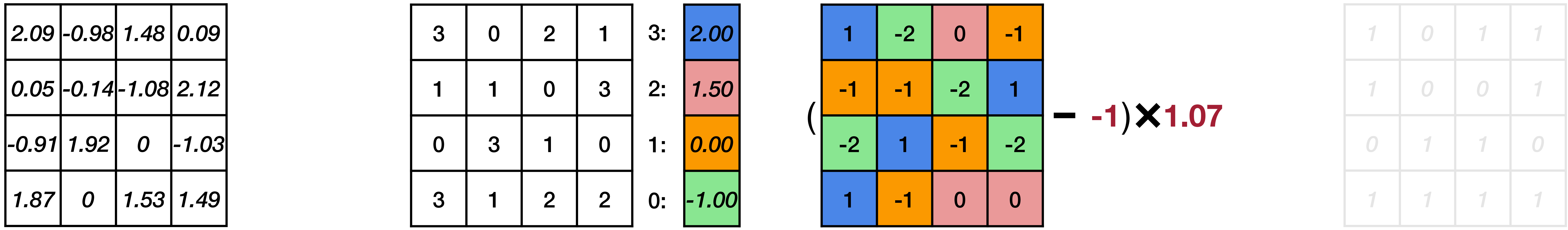
3:	*2.00*
2:	*1.50*
1:	*0.00*
0:	*-1.00*

During Computation **In Storage**



- The weights are decompressed using a lookup table (*i.e.*, codebook) during runtime inference.
- K-Means-based Weight Quantization only saves storage cost of a neural network model.
  - All the computation and memory access are still floating-point.

Neural Network Quantization



		K-Means-based Quantization	Linear Quantization	Binary/Ternary Quantization
Storage	Floating-Point Weights	Integer Weights; Floating-Point Codebook	Integer Weights	
Computation	Floating-Point Arithmetic	Floating-Point Arithmetic	Integer Arithmetic	

Linear Quantization

What is Linear Quantization?

An affine mapping of integers to real numbers

weights (32-bit float) ⇒ (quantized weights (2-bit signed int) − zero point (2-bit signed int)) × scale (32-bit float) = reconstructed weights (32-bit float)

$$
\begin{bmatrix} 2.09 & -0.98 & 1.48 & 0.09 \\ 0.05 & -0.14 & -1.08 & 2.12 \\ -0.91 & 1.92 & 0 & -1.03 \\ 1.87 & 0 & 1.53 & 1.49 \end{bmatrix}
\Rightarrow
\left(
\begin{bmatrix} 1 & -2 & 0 & -1 \\ -1 & -1 & -2 & 1 \\ -2 & 1 & -1 & -2 \\ 1 & -1 & 0 & 0 \end{bmatrix}
- (-1) \right) \times 1.07 =
\begin{bmatrix} 2.14 & -1.07 & 1.07 & 0 \\ 0 & 0 & -1.07 & 2.14 \\ -1.07 & 2.14 & 0 & -1.07 \\ 2.14 & 0 & 1.07 & 1.07 \end{bmatrix}
$$

we will learn how to determine these parameters later

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

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

Linear Quantization

An affine mapping of integers to real numbers $r = S(q - Z)$

$$
\begin{array}{ccccccc}
\text{weights (32-bit float)} & & \text{quantized weights (2-bit signed int)} & & \text{zero point (2-bit signed int)} & & \text{scale (32-bit float)} \\
\begin{bmatrix} 2.09 & -0.98 & 1.48 & 0.09 \\ 0.05 & -0.14 & -1.08 & 2.12 \\ -0.91 & 1.92 & 0 & -1.03 \\ 1.87 & 0 & 1.53 & 1.49 \end{bmatrix} & \Rightarrow (& \begin{bmatrix} 1 & -2 & 0 & -1 \\ -1 & -1 & -2 & 1 \\ -2 & 1 & -1 & -2 \\ 1 & -1 & 0 & 0 \end{bmatrix} & - & -1 &) \times & 1.07 \\
r & = (& q & - & Z &) \times & S
\end{array}
$$

Floating-point — **Integer** — **Integer** — **Floating-point**

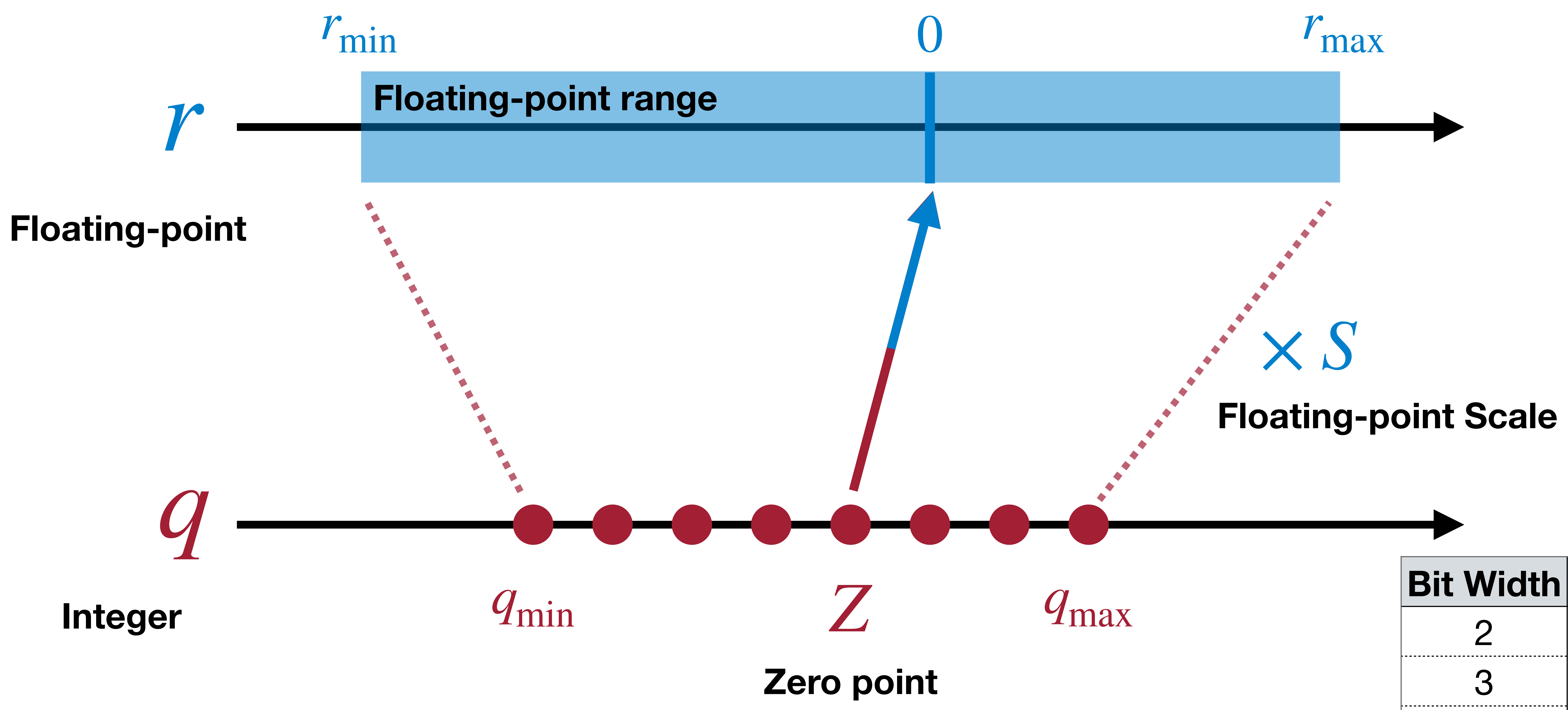
Integer (Z):
- quantization parameter
- allow real number $r=0$ be exactly representable by a quantized integer Z

Floating-point (S):
- quantization parameter

Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [Jacob *et al.*, CVPR 2018]

Linear Quantization

An affine mapping of integers to real numbers $r = S(q - Z)$

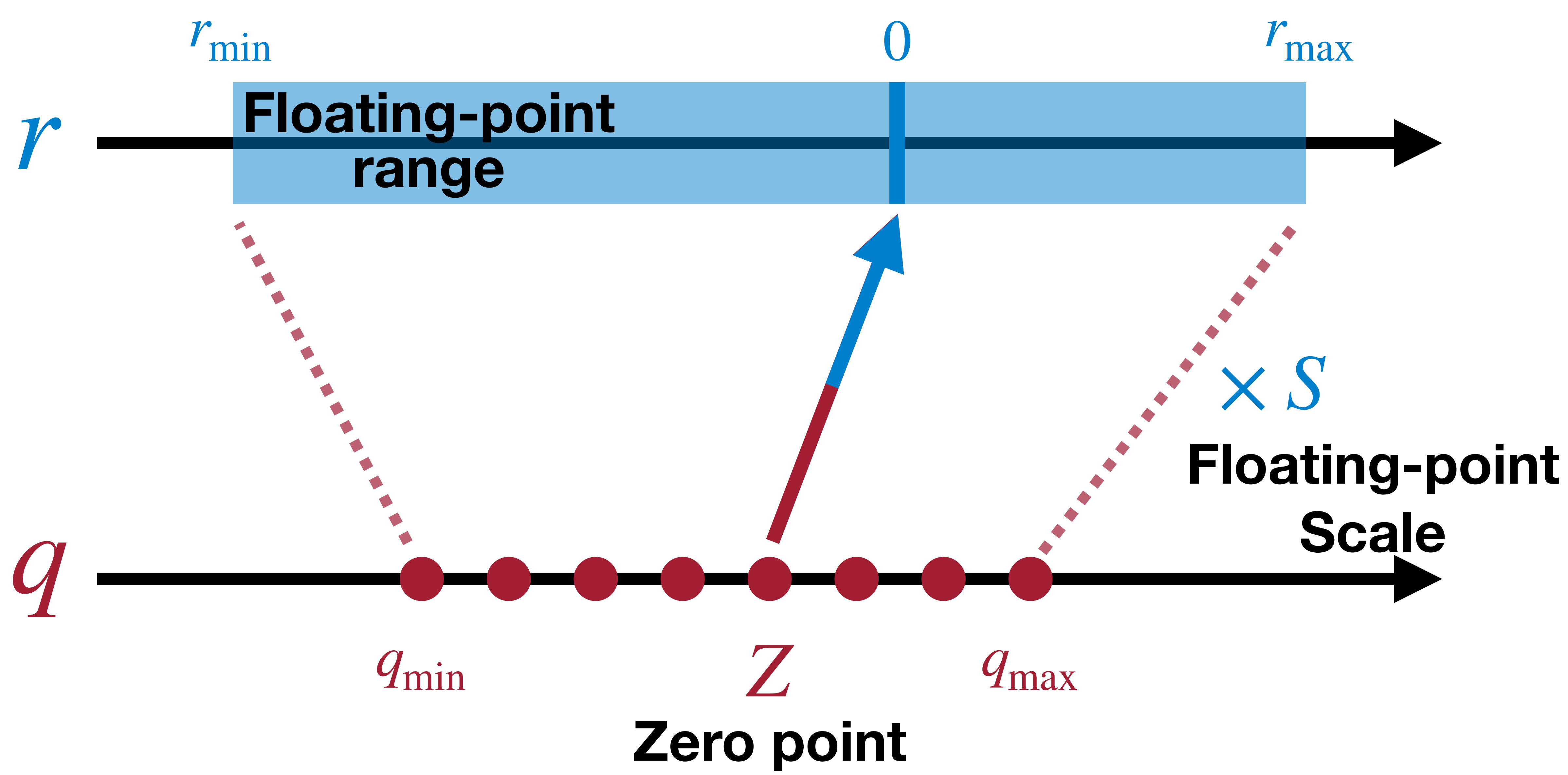


Bit Width	q_{min}	q_{max}
2	-2	1
3	-4	3
4	-8	7
N	-2^{N-1}	$2^{N-1}-1$

Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [Jacob *et al.*, CVPR 2018]

Scale of Linear Quantization

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$



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

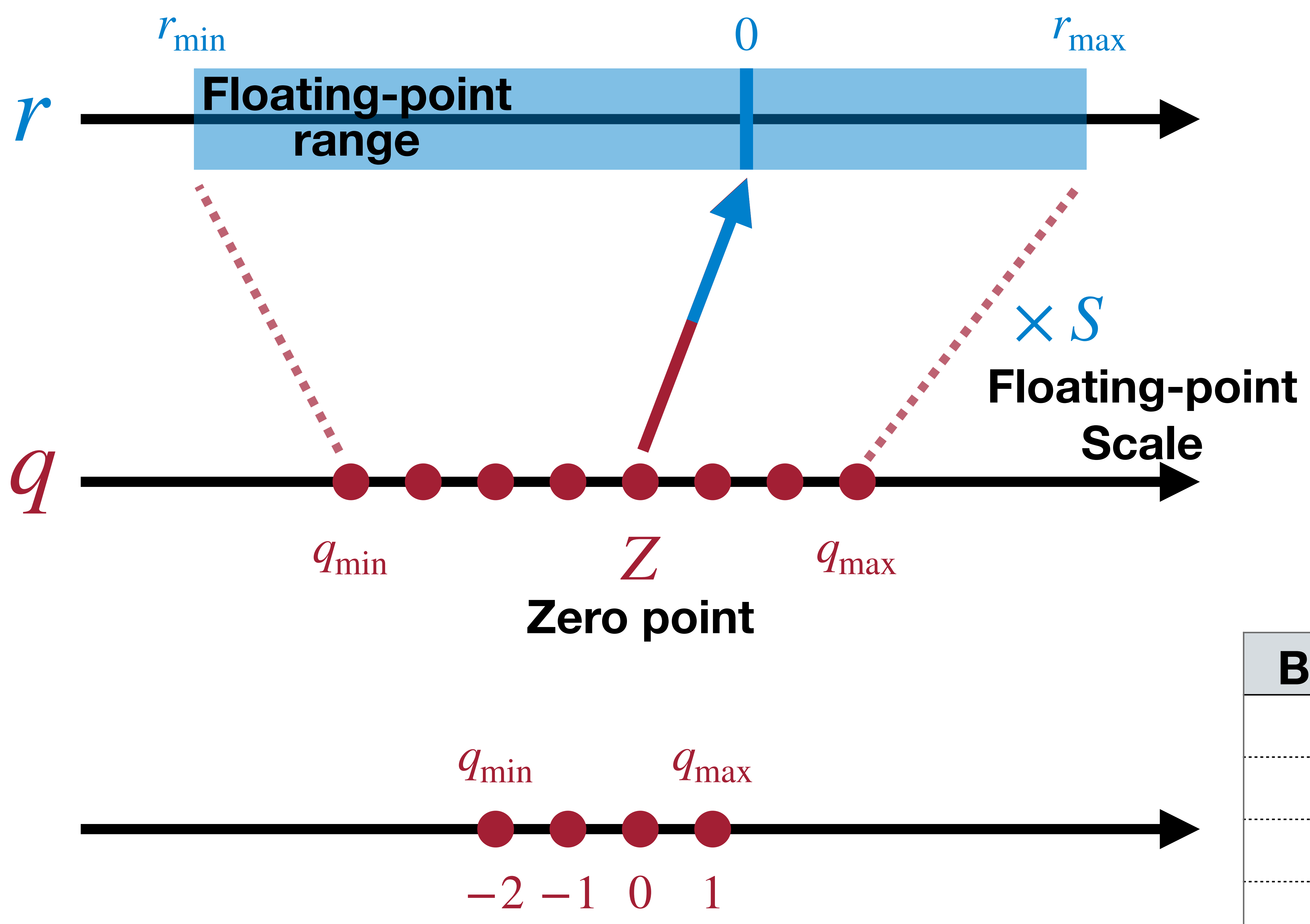
$$r_{\min} = S\left(q_{\min} - Z\right)$$

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

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

Scale of Linear Quantization

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$



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

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

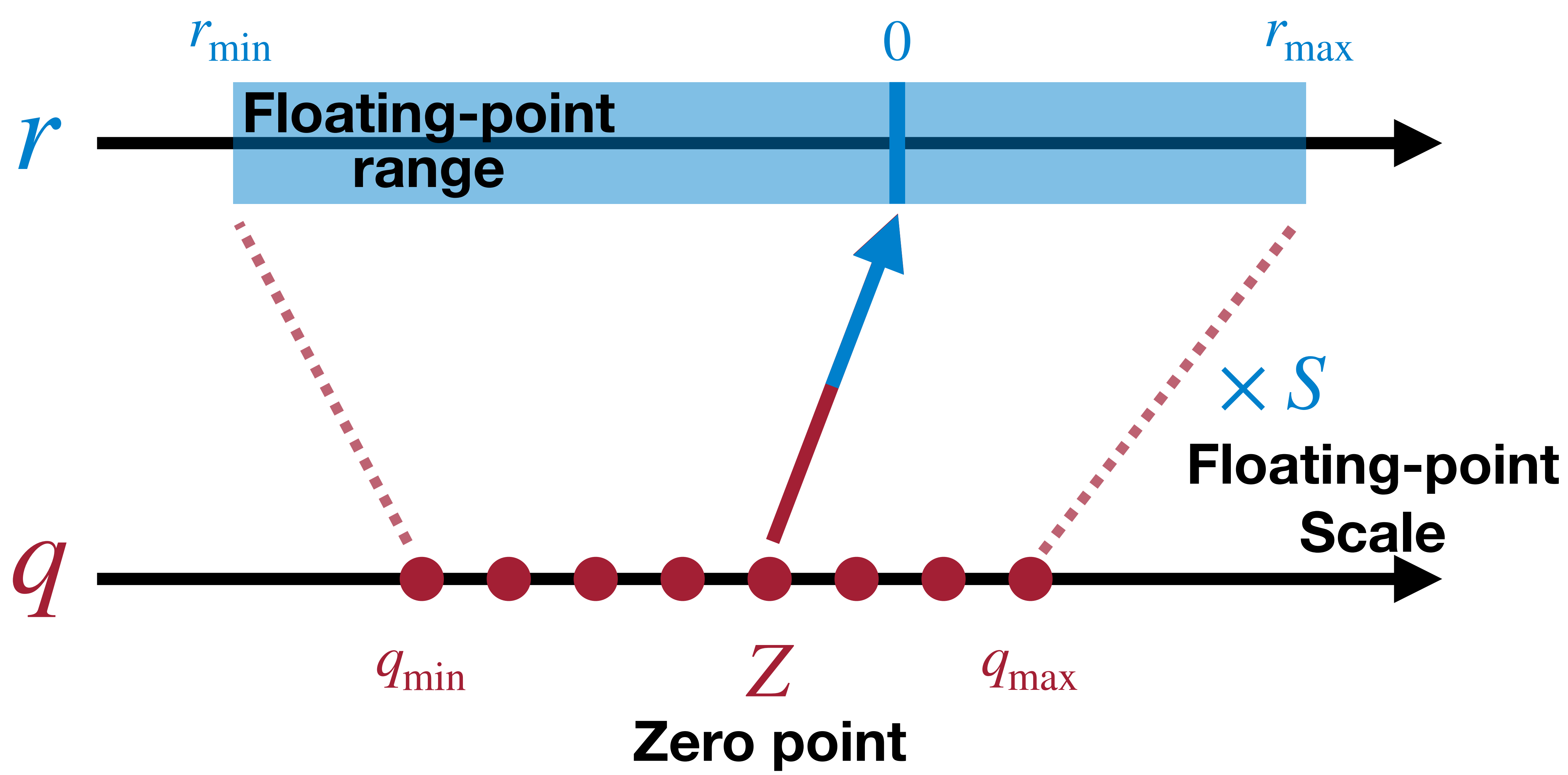
$$= \frac{2.12 - (-1.08)}{1 - (-2)}$$

$$= 1.07$$

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

Zero Point of Linear Quantization

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$



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

$$\downarrow$$

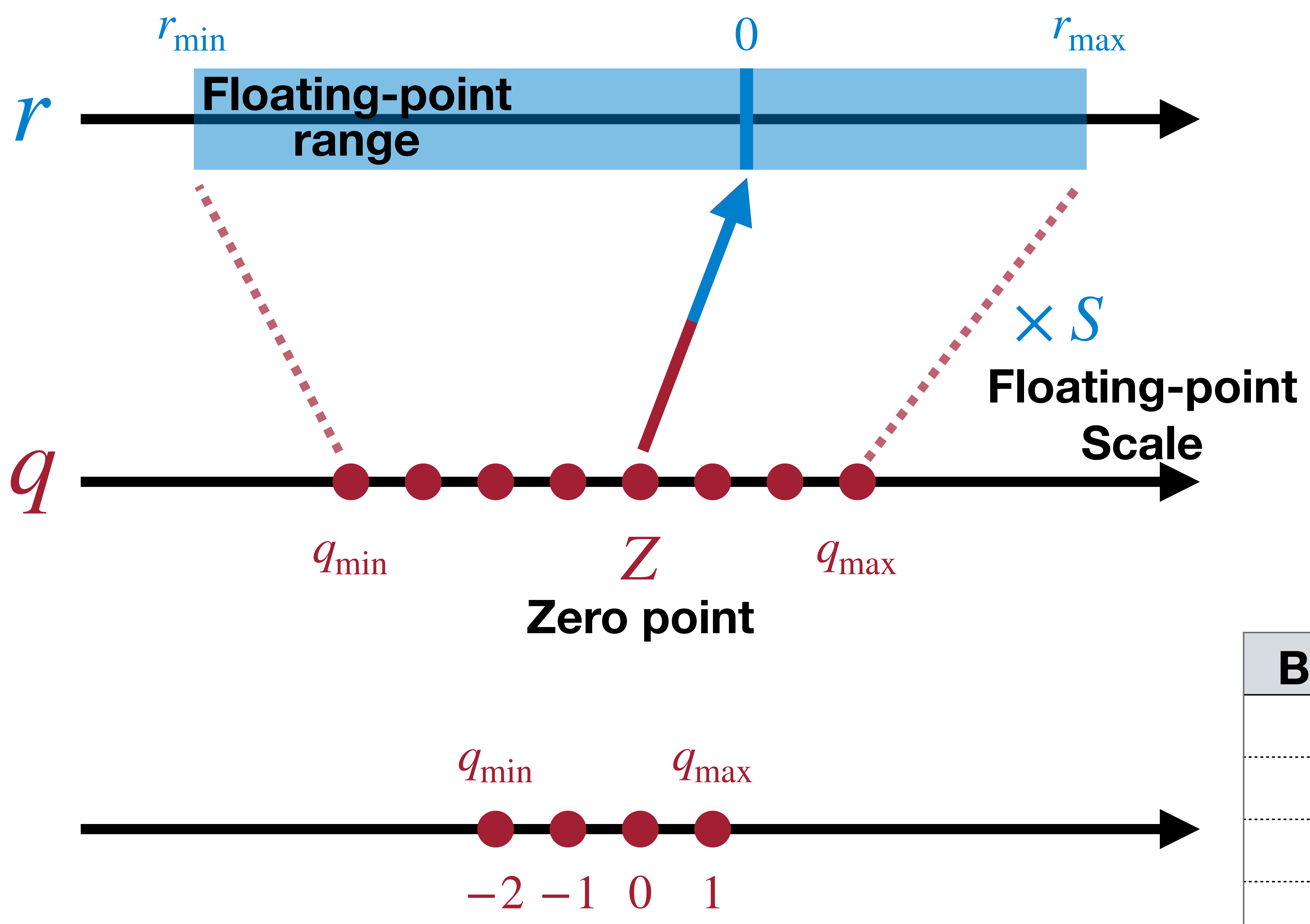
$$Z = q_{\min} - \frac{r_{\min}}{S}$$

$$\downarrow$$

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

Zero Point of Linear Quantization

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$



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

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

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

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

$$= -1$$

Linear Quantized Matrix Multiplication

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$

- Consider the following matrix multiplication.

$$\mathbf{Y} = \mathbf{W}\mathbf{X}$$

$$S_{\mathbf{Y}}\left(\mathbf{q}_{\mathbf{Y}} - Z_{\mathbf{Y}}\right) = S_{\mathbf{W}}\left(\mathbf{q}_{\mathbf{W}} - Z_{\mathbf{W}}\right) \cdot S_{\mathbf{X}}\left(\mathbf{q}_{\mathbf{X}} - Z_{\mathbf{X}}\right)$$

$$\mathbf{q}_{\mathbf{Y}} = \frac{S_{\mathbf{W}} S_{\mathbf{X}}}{S_{\mathbf{Y}}}\left(\mathbf{q}_{\mathbf{W}} - Z_{\mathbf{W}}\right)\left(\mathbf{q}_{\mathbf{X}} - Z_{\mathbf{X}}\right) + Z_{\mathbf{Y}}$$

$$\mathbf{q}_{\mathbf{Y}} = \frac{S_{\mathbf{W}} S_{\mathbf{X}}}{S_{\mathbf{Y}}}\left(\mathbf{q}_{\mathbf{W}}\mathbf{q}_{\mathbf{X}} - Z_{\mathbf{W}}\mathbf{q}_{\mathbf{X}} - Z_{\mathbf{X}}\mathbf{q}_{\mathbf{W}} + Z_{\mathbf{W}}Z_{\mathbf{X}}\right) + Z_{\mathbf{Y}}$$

Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [Jacob *et al.*, CVPR 2018]

Linear Quantized Matrix Multiplication

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$

- Consider the following matrix multiplication.

$$\mathbf{Y} = \mathbf{W}\mathbf{X}$$

$$\mathbf{q_Y} = \boxed{\frac{S_\mathbf{W} S_\mathbf{X}}{S_\mathbf{Y}}} \boxed{\left(\mathbf{q_W}\mathbf{q_X} - Z_\mathbf{W}\mathbf{q_X} \underbrace{- Z_\mathbf{X}\mathbf{q_W} + Z_\mathbf{W}Z_\mathbf{X}}_{\text{Precompute}}\right)} + \boxed{Z_\mathbf{Y}}$$

Rescale to *N*-bit Integer

N-bit Integer Multiplication
32-bit Integer Addition/Subtraction

N-bit Integer Addition

Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [Jacob *et al.*, CVPR 2018]

Linear Quantized Matrix Multiplication

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$

- Consider the following matrix multiplication.

$$\mathbf{Y} = \mathbf{W}\mathbf{X}$$

$$\mathbf{q}_{\mathbf{Y}} = \boxed{\frac{S_{\mathbf{W}} S_{\mathbf{X}}}{S_{\mathbf{Y}}}} \left(\boxed{\mathbf{q}_{\mathbf{W}} \mathbf{q}_{\mathbf{X}} - Z_{\mathbf{W}} \mathbf{q}_{\mathbf{X}} - Z_{\mathbf{X}} \mathbf{q}_{\mathbf{W}} + Z_{\mathbf{W}} Z_{\mathbf{X}}} \right) + \boxed{Z_{\mathbf{Y}}}$$

- Empirically, the scale $\frac{S_{\mathbf{W}} S_{\mathbf{X}}}{S_{\mathbf{Y}}}$ is always in the interval (0, 1).

Fixed-point Multiplication

$$\frac{S_{\mathbf{W}} S_{\mathbf{X}}}{S_{\mathbf{Y}}} = 2^{-n} M_0, \quad \text{where} \quad M_0 \in [0.5, 1)$$

Bit Shift

Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [Jacob *et al.*, CVPR 2018]

Linear Quantized Matrix Multiplication

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$

- Consider the following matrix multiplication.

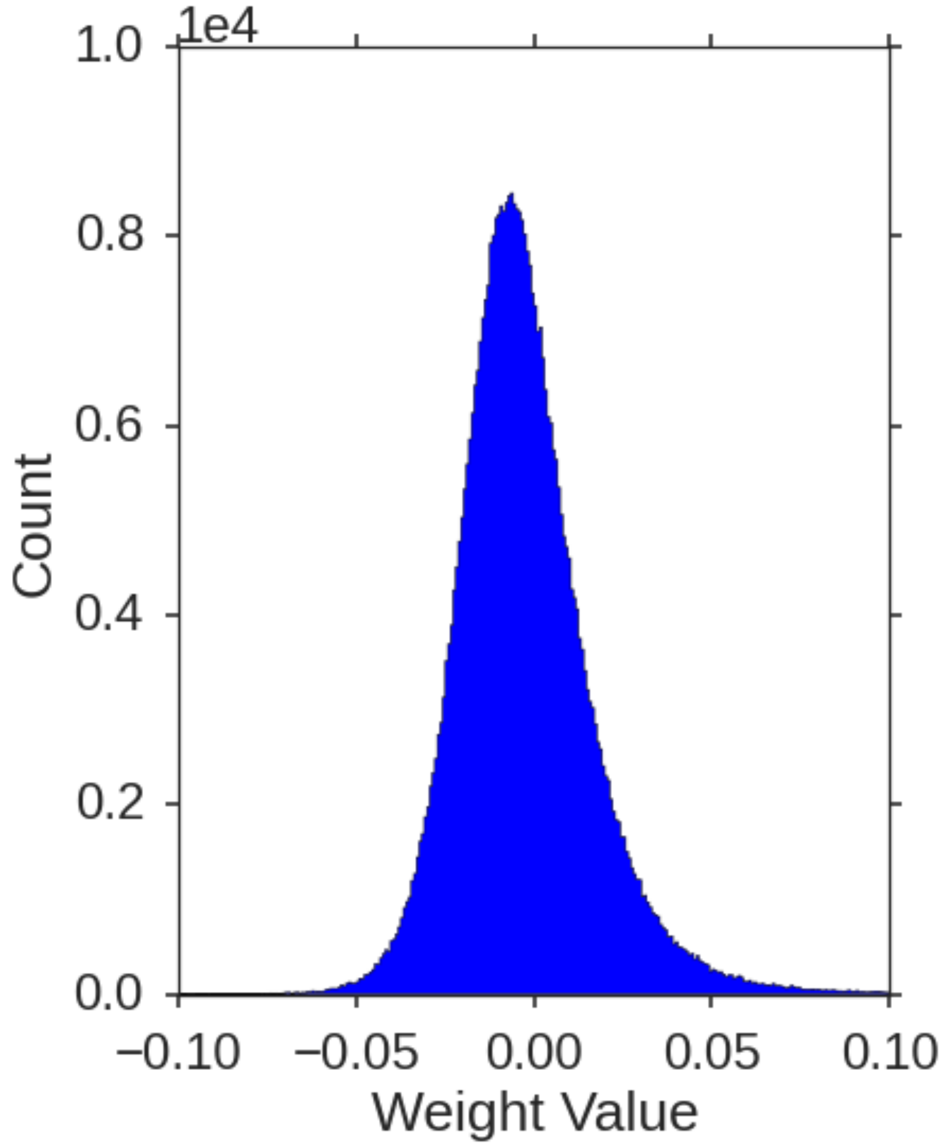
$$\mathbf{Y} = \mathbf{W}\mathbf{X}$$

$$\mathbf{q_Y} = \boxed{\frac{S_\mathbf{W} S_\mathbf{X}}{S_\mathbf{Y}}} \boxed{\left(\mathbf{q_W}\mathbf{q_X} - Z_\mathbf{W}\mathbf{q_X} \underbrace{- Z_\mathbf{X}\mathbf{q_W} + Z_\mathbf{W}Z_\mathbf{X}}_{\text{Precompute}}\right)} + \boxed{Z_\mathbf{Y}}$$

Rescale to *N*-bit Integer

N-bit Integer Multiplication
32-bit Integer Addition/Subtraction

N-bit Integer Addition

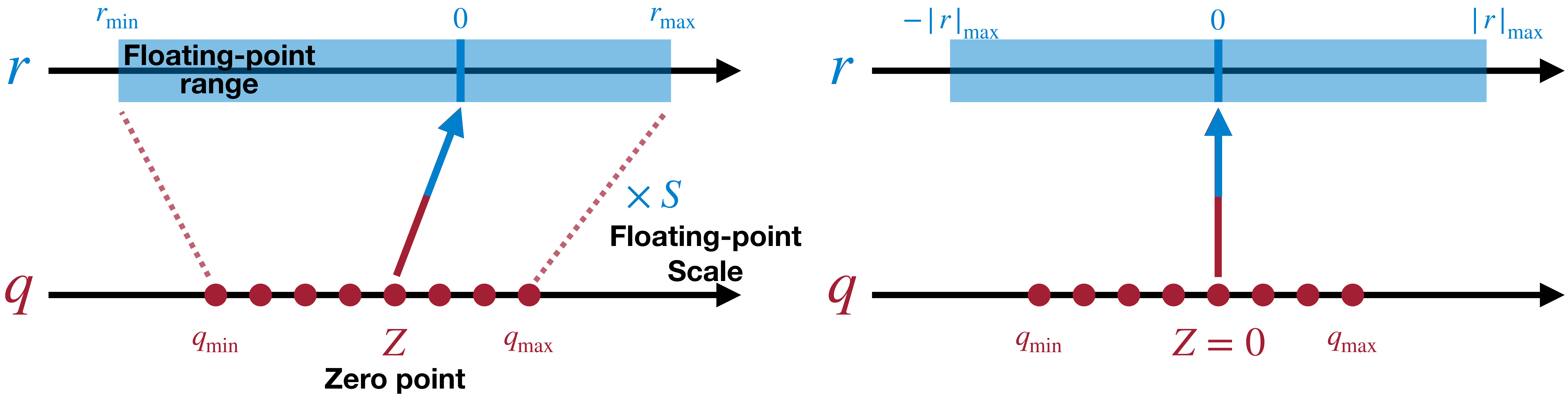


$$Z_\mathbf{W} = 0?$$

Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [Jacob *et al.*, CVPR 2018]

Symmetric Linear Quantization

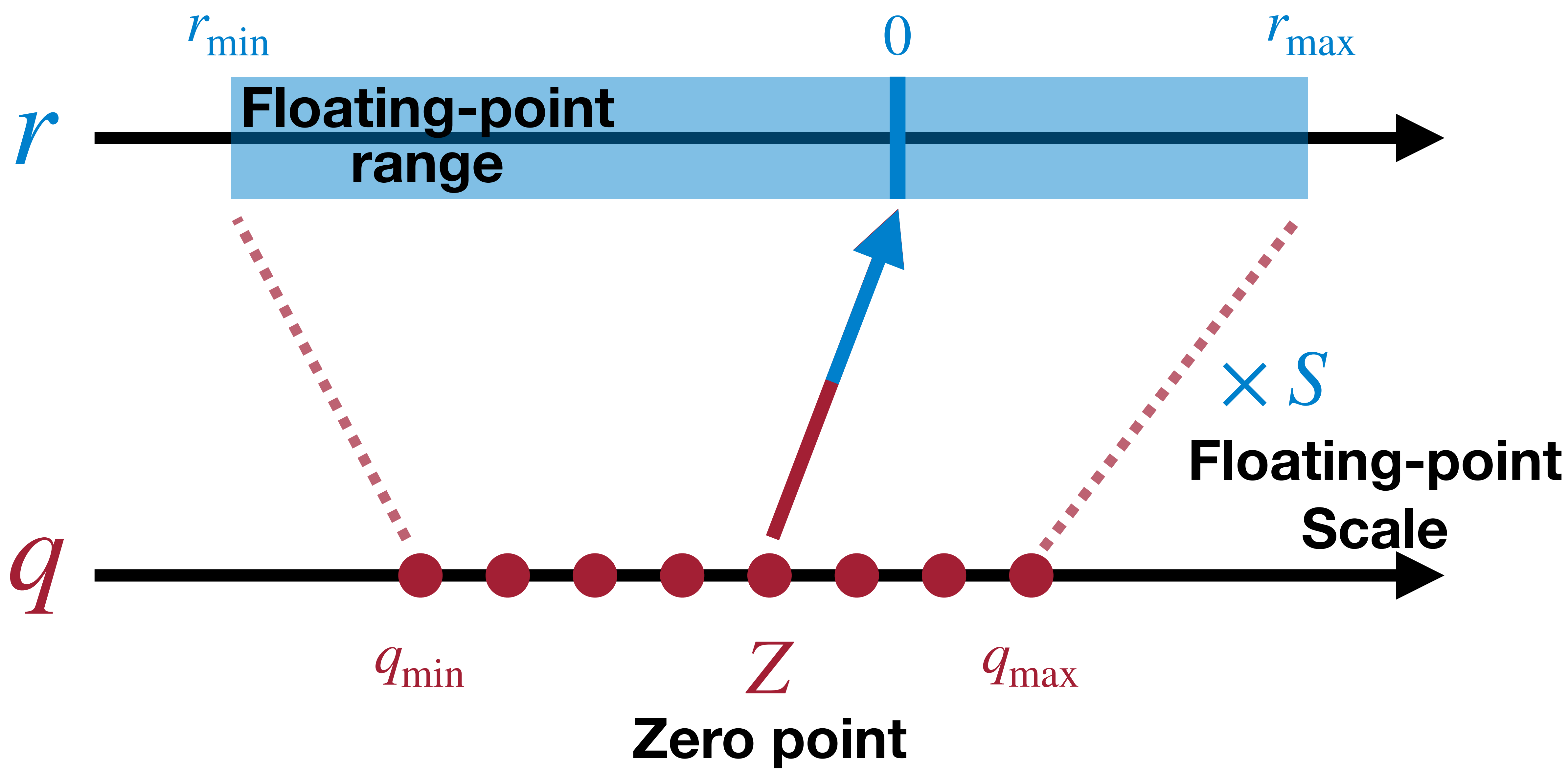
Zero point $Z = 0$ and Symmetric floating-point range

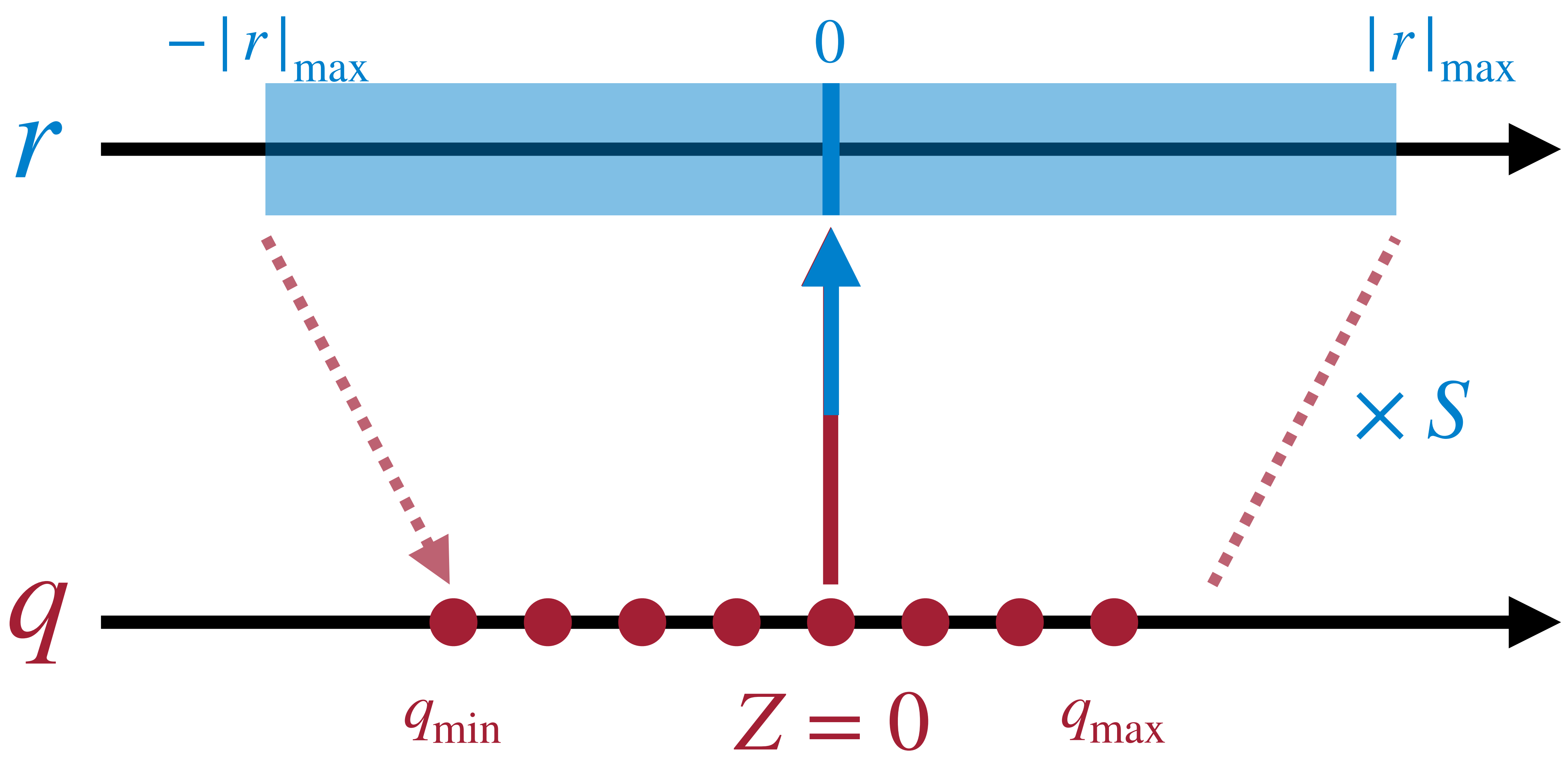


Bit Width	q_{min}	q_{max}
2	-2	1
3	-4	3
4	-8	7
N	-2^{N-1}	$2^{N-1}-1$

Symmetric Linear Quantization

Full range mode





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

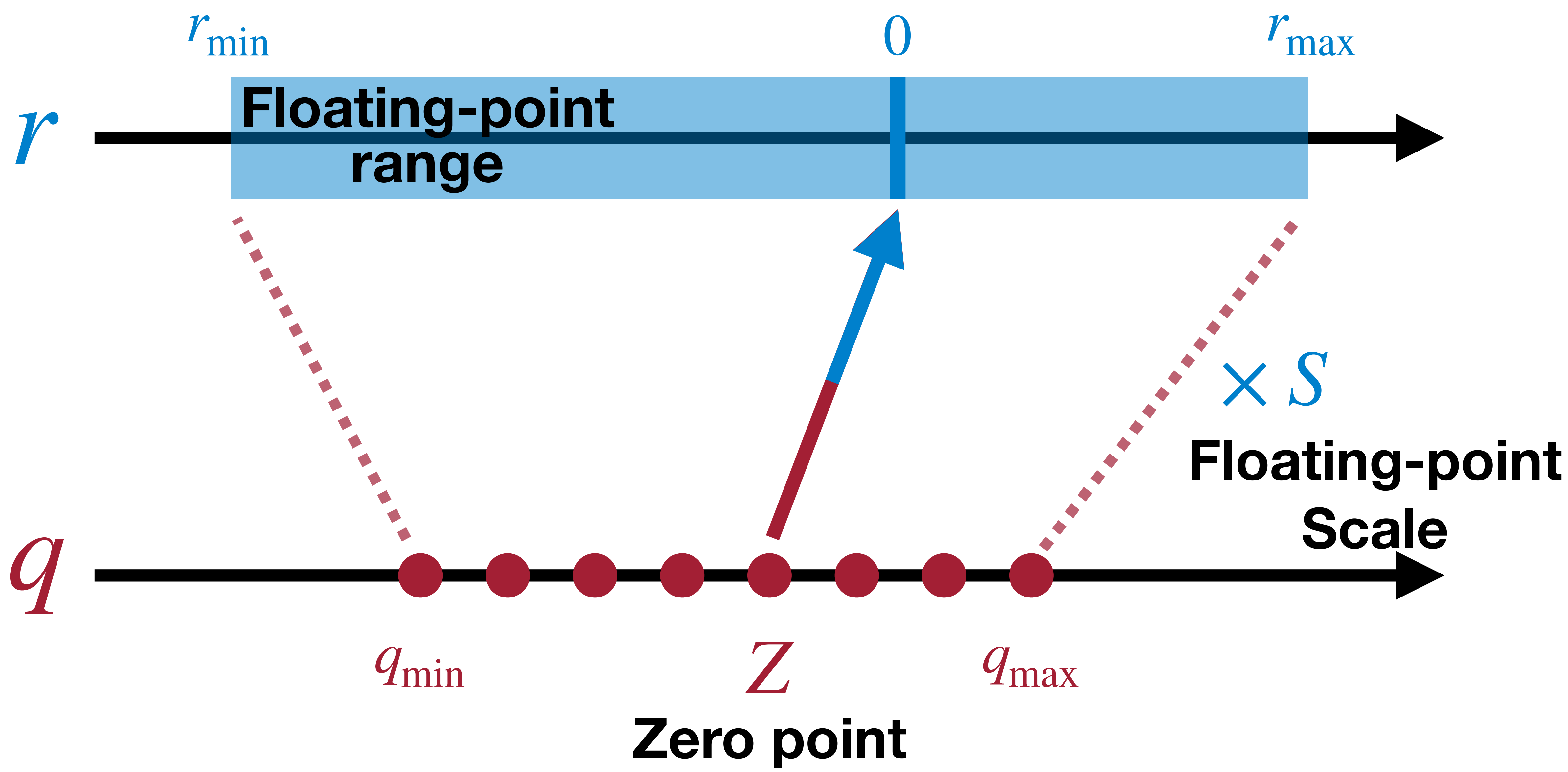
$$S = \frac{r_{min}}{q_{min} - Z} = \frac{-|r|_{max}}{q_{min}} = \frac{|r|_{max}}{2^{N-1}}$$

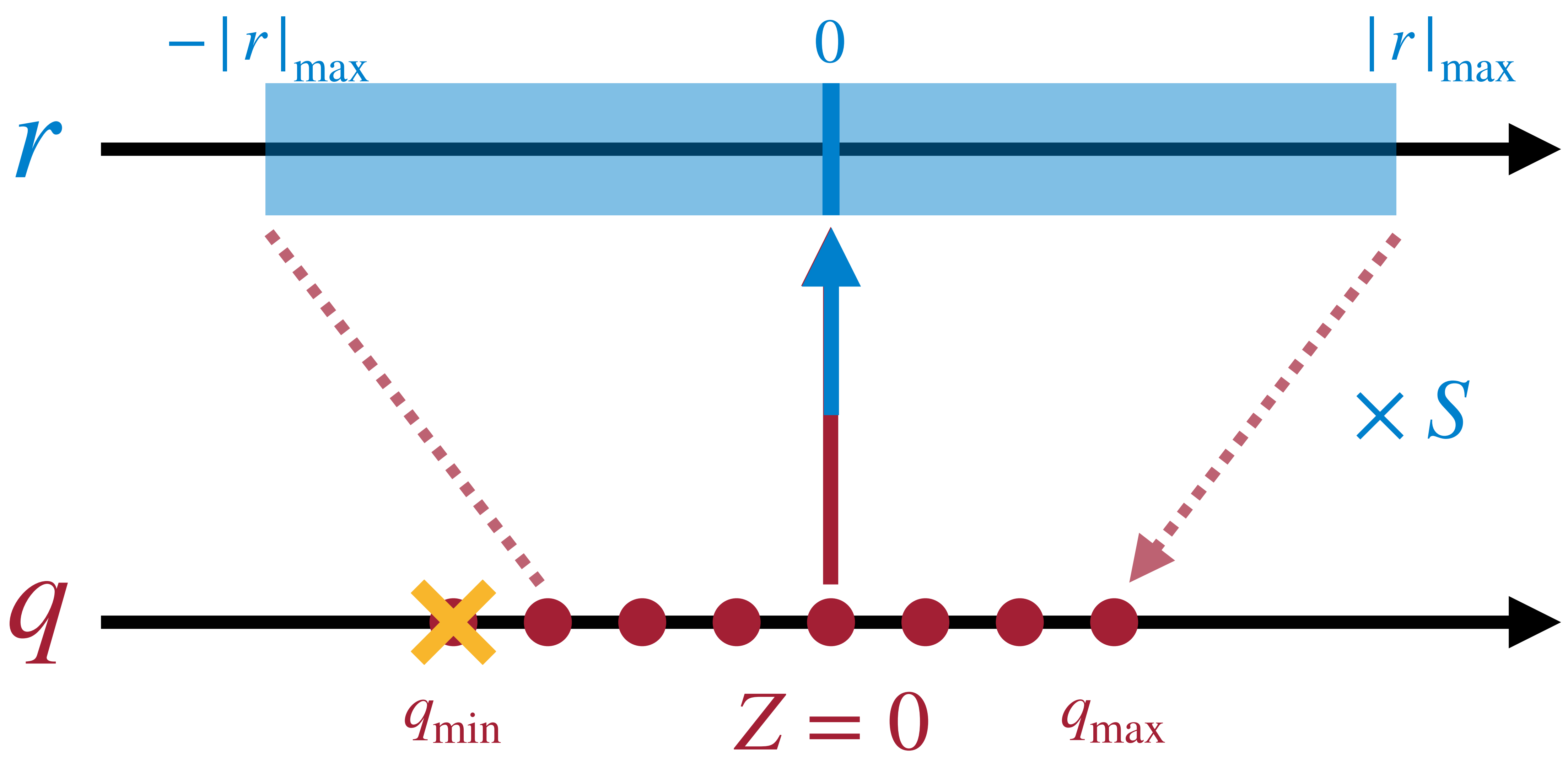
- use full range of quantized integers
- example: PyTorch's native quantization, ONNX

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

Symmetric Linear Quantization

Restricted range mode





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

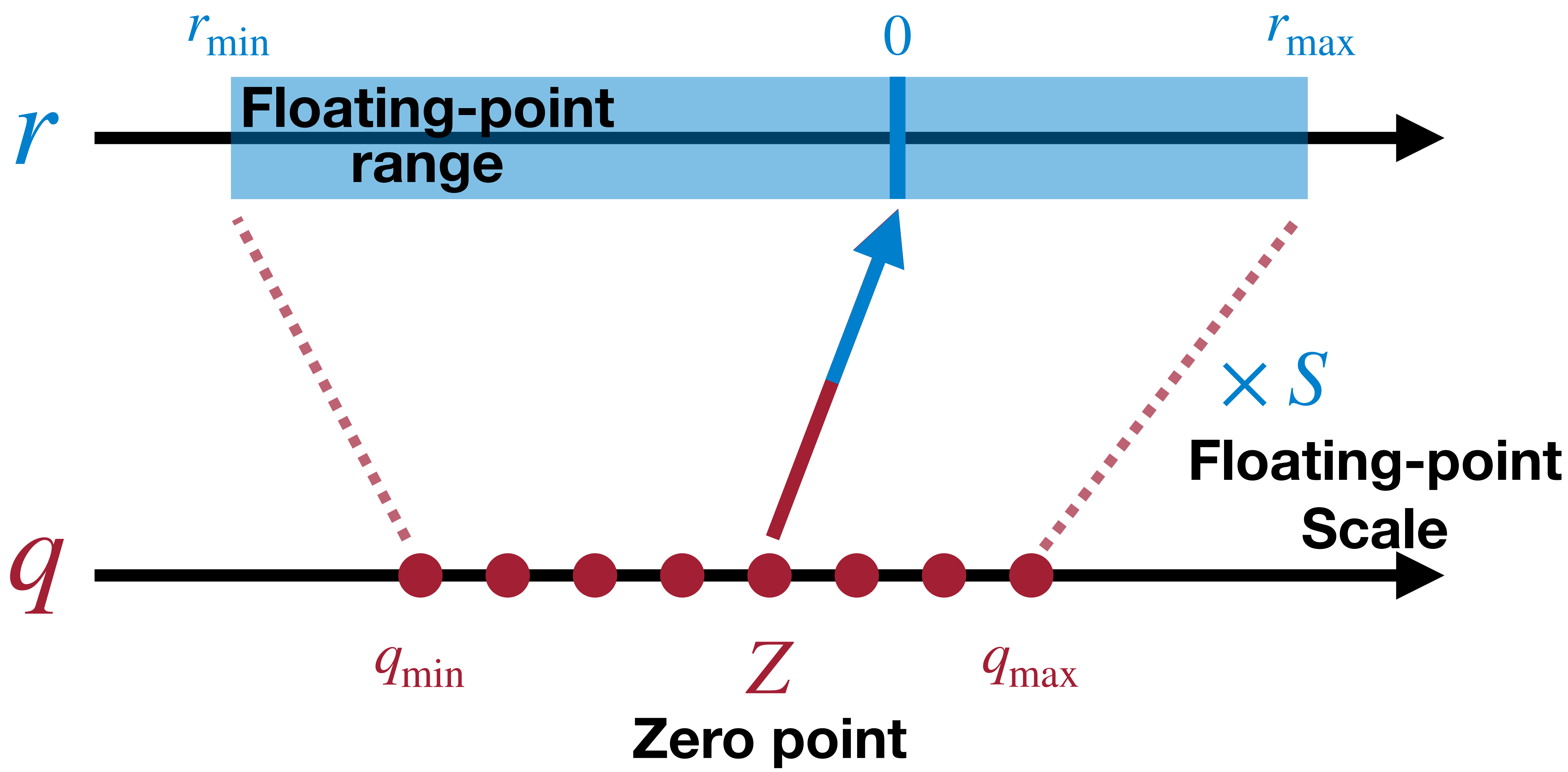
$$S = \frac{r_{\max}}{q_{\max} - Z} = \frac{|r|_{\max}}{q_{\max}} = \frac{|r|_{\max}}{2^{N-1} - 1}$$

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

- example: TensorFlow, NVIDIA TensorRT, Intel DNNL

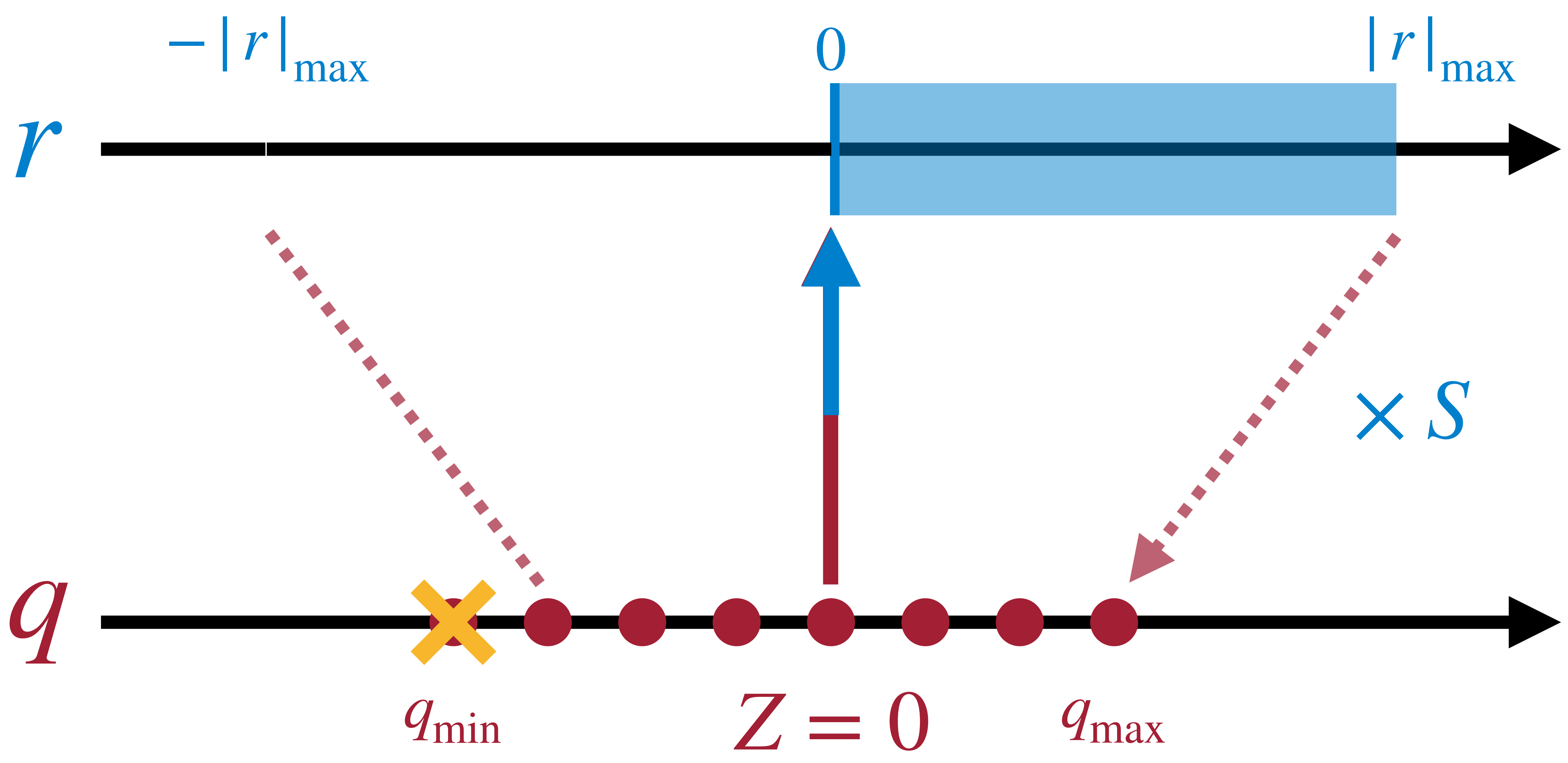
Asymmetric *vs.* Symmetric

Asymmetric Linear Quantization



- The quantized range is fully used.
- The implementation is more complex, and zero points require additional logic in hardware.

Symmetric Linear Quantization



- The quantized range will be wasted for biased float range.
  - Activation tensor is non-negative after ReLU, and thus symmetric quantization will lose 1 bit effectively.
- The implementation is much simpler.

Neural Network Distiller

Linear Quantized Matrix Multiplication

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$

- Consider the following matrix multiplication.

$$\mathbf{Y} = \mathbf{W}\mathbf{X}$$

$$\mathbf{q_Y} = \boxed{\frac{S_\mathbf{W} S_\mathbf{X}}{S_\mathbf{Y}}} \boxed{\left(\mathbf{q_W q_X} - Z_\mathbf{W}\mathbf{q_X} \underbrace{- Z_\mathbf{X}\mathbf{q_W} + Z_\mathbf{W} Z_\mathbf{X}}_{\text{Precompute}}\right)} + \boxed{Z_\mathbf{Y}}$$

Rescale to *N*-bit Integer

N-bit Integer Multiplication
32-bit Integer Addition/Subtraction

N-bit Integer Addition

$$\mathbf{q_Y} = \boxed{\frac{S_\mathbf{W} S_\mathbf{X}}{S_\mathbf{Y}}} \boxed{\left(\mathbf{q_W q_X} - Z_\mathbf{X}\mathbf{q_W}\right)} + \boxed{Z_\mathbf{Y}}$$

$$Z_\mathbf{W} = 0$$

Linear Quantized Fully-Connected Layer

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$

- So far, we ignore bias. Now we consider the following fully-connected layer with bias.

$$\mathbf{Y} = \mathbf{W}\mathbf{X} + \mathbf{b}$$

$$S_{\mathbf{Y}}\left(\mathbf{q}_{\mathbf{Y}} - Z_{\mathbf{Y}}\right) = S_{\mathbf{W}}\left(\mathbf{q}_{\mathbf{W}} - Z_{\mathbf{W}}\right) \cdot S_{\mathbf{X}}\left(\mathbf{q}_{\mathbf{X}} - Z_{\mathbf{X}}\right) + S_{\mathbf{b}}\left(\mathbf{q}_{\mathbf{b}} - Z_{\mathbf{b}}\right)$$

$$\downarrow Z_{\mathbf{W}} = 0$$

$$S_{\mathbf{Y}}\left(\mathbf{q}_{\mathbf{Y}} - Z_{\mathbf{Y}}\right) = \underline{\boxed{S_{\mathbf{W}}S_{\mathbf{X}}}\left(\mathbf{q}_{\mathbf{W}}\mathbf{q}_{\mathbf{X}} - Z_{\mathbf{X}}\mathbf{q}_{\mathbf{W}}\right)} + \underline{\boxed{S_{\mathbf{b}}}\left(\mathbf{q}_{\mathbf{b}} - \boxed{Z_{\mathbf{b}}}\right)}$$

Linear Quantized Fully-Connected Layer

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$

- So far, we ignore bias. Now we consider the following fully-connected layer with bias.

$$\mathbf{Y} = \mathbf{W}\mathbf{X} + \mathbf{b}$$

$$S_{\mathbf{Y}}\left(\mathbf{q}_{\mathbf{Y}} - Z_{\mathbf{Y}}\right) = S_{\mathbf{W}}\left(\mathbf{q}_{\mathbf{W}} - Z_{\mathbf{W}}\right) \cdot S_{\mathbf{X}}\left(\mathbf{q}_{\mathbf{X}} - Z_{\mathbf{X}}\right) + S_{\mathbf{b}}\left(\mathbf{q}_{\mathbf{b}} - Z_{\mathbf{b}}\right)$$

$$\downarrow \; Z_{\mathbf{W}} = 0$$

$$S_{\mathbf{Y}}\left(\mathbf{q}_{\mathbf{Y}} - Z_{\mathbf{Y}}\right) = S_{\mathbf{W}}S_{\mathbf{X}}\left(\mathbf{q}_{\mathbf{W}}\mathbf{q}_{\mathbf{X}} - Z_{\mathbf{X}}\mathbf{q}_{\mathbf{W}}\right) + S_{\mathbf{b}}\left(\mathbf{q}_{\mathbf{b}} - Z_{\mathbf{b}}\right)$$

$$\downarrow \; Z_{\mathbf{b}} = 0, \quad S_{\mathbf{b}} = S_{\mathbf{W}}S_{\mathbf{X}}$$

$$S_{\mathbf{Y}}\left(\mathbf{q}_{\mathbf{Y}} - Z_{\mathbf{Y}}\right) = S_{\mathbf{W}}S_{\mathbf{X}}\left(\mathbf{q}_{\mathbf{W}}\mathbf{q}_{\mathbf{X}} - Z_{\mathbf{X}}\mathbf{q}_{\mathbf{W}} + \mathbf{q}_{\mathbf{b}}\right)$$

Linear Quantized Fully-Connected Layer

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$

- So far, we ignore bias. Now we consider the following fully-connected layer with bias.

$$\mathbf{Y} = \mathbf{W}\mathbf{X} + \mathbf{b}$$

$$\Big\downarrow \; Z_{\mathbf{W}} = 0, \quad Z_{\mathbf{b}} = 0, \quad S_{\mathbf{b}} = S_{\mathbf{W}} S_{\mathbf{X}}$$

$$S_{\mathbf{Y}}\left(\mathbf{q}_{\mathbf{Y}} - Z_{\mathbf{Y}}\right) = S_{\mathbf{W}} S_{\mathbf{X}}\left(\mathbf{q}_{\mathbf{W}} \mathbf{q}_{\mathbf{X}} - Z_{\mathbf{X}} \mathbf{q}_{\mathbf{W}} + \mathbf{q}_{\mathbf{b}}\right)$$

$$\Big\downarrow$$

Precompute

$$\mathbf{q}_{\mathbf{Y}} = \frac{S_{\mathbf{W}} S_{\mathbf{X}}}{S_{\mathbf{Y}}}\left(\mathbf{q}_{\mathbf{W}} \mathbf{q}_{\mathbf{X}} + \boxed{\mathbf{q}_{\mathbf{b}} - Z_{\mathbf{X}} \mathbf{q}_{\mathbf{W}}}\right) + Z_{\mathbf{Y}}$$

$$\Big\downarrow \; \mathbf{q}_{bias} = \mathbf{q}_{\mathbf{b}} - Z_{\mathbf{X}} \mathbf{q}_{\mathbf{W}}$$

$$\mathbf{q}_{\mathbf{Y}} = \frac{S_{\mathbf{W}} S_{\mathbf{X}}}{S_{\mathbf{Y}}}\left(\mathbf{q}_{\mathbf{W}} \mathbf{q}_{\mathbf{X}} + \mathbf{q}_{bias}\right) + Z_{\mathbf{Y}}$$

Linear Quantized Fully-Connected Layer

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$

- So far, we ignore bias. Now we consider the following fully-connected layer with bias.

$$\mathbf{Y} = \mathbf{W}\mathbf{X} + \mathbf{b}$$

$$Z_{\mathbf{W}} = 0$$

$$Z_{\mathbf{b}} = 0, \quad S_{\mathbf{b}} = S_{\mathbf{W}} S_{\mathbf{X}}$$

$$\mathbf{q}_{bias} = \mathbf{q}_{\mathbf{b}} - Z_{\mathbf{X}} \mathbf{q}_{\mathbf{W}}$$

$$\mathbf{q}_{\mathbf{Y}} = \boxed{\frac{S_{\mathbf{W}} S_{\mathbf{X}}}{S_{\mathbf{Y}}}} \left(\boxed{\mathbf{q}_{\mathbf{W}} \mathbf{q}_{\mathbf{X}} + \mathbf{q}_{bias}} \right) + \boxed{Z_{\mathbf{Y}}}$$

Rescale to *N*-bit Int

N-bit Int Mult.
32-bit Int Add.

N-bit Int Add

Note: both $\mathbf{q}_{\mathbf{b}}$ and $\mathbf{q}_{bias}$ are 32 bits.

Linear Quantized Convolution Layer

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$

- Consider the following convolution layer.

$$\mathbf{Y} = \text{Conv}\left(\mathbf{W}, \mathbf{X}\right) + \mathbf{b}$$

$$Z_{\mathbf{W}} = 0$$

$$Z_{\mathbf{b}} = 0, \quad S_{\mathbf{b}} = S_{\mathbf{W}} S_{\mathbf{X}}$$

$$\mathbf{q}_{bias} = \mathbf{q}_{\mathbf{b}} - \text{Conv}\left(\mathbf{q}_{\mathbf{W}}, Z_{\mathbf{X}}\right)$$

$$\mathbf{q}_{\mathbf{Y}} = \boxed{\frac{S_{\mathbf{W}} S_{\mathbf{X}}}{S_{\mathbf{Y}}}} \left(\boxed{\text{Conv}\left(\mathbf{q}_{\mathbf{W}}, \mathbf{q}_{\mathbf{X}}\right) + \mathbf{q}_{bias}} \right) + \boxed{Z_{\mathbf{Y}}}$$

Rescale to *N*-bit Int — **N-bit Int Mult. 32-bit Int Add.** — ***N*-bit Int Add**

Note: both $\mathbf{q}_{\mathbf{b}}$ and $\mathbf{q}_{bias}$ are 32 bits.

Linear Quantized Convolution Layer

Linear Quantization is an affine mapping of integers to real numbers $r = S(q - Z)$

- Consider the following convolution layer.

$$\mathbf{Y} = \text{Conv}(\mathbf{W}, \mathbf{X}) + \mathbf{b}$$

$$Z_{\mathbf{W}} = 0$$

$$Z_{\mathbf{b}} = 0, \quad S_{\mathbf{b}} = S_{\mathbf{W}} S_{\mathbf{X}}$$

$$\mathbf{q}_{bias} = \mathbf{q}_{\mathbf{b}} - \boxed{\text{Conv}(\mathbf{q}_{\mathbf{W}}, Z_{\mathbf{X}})}$$

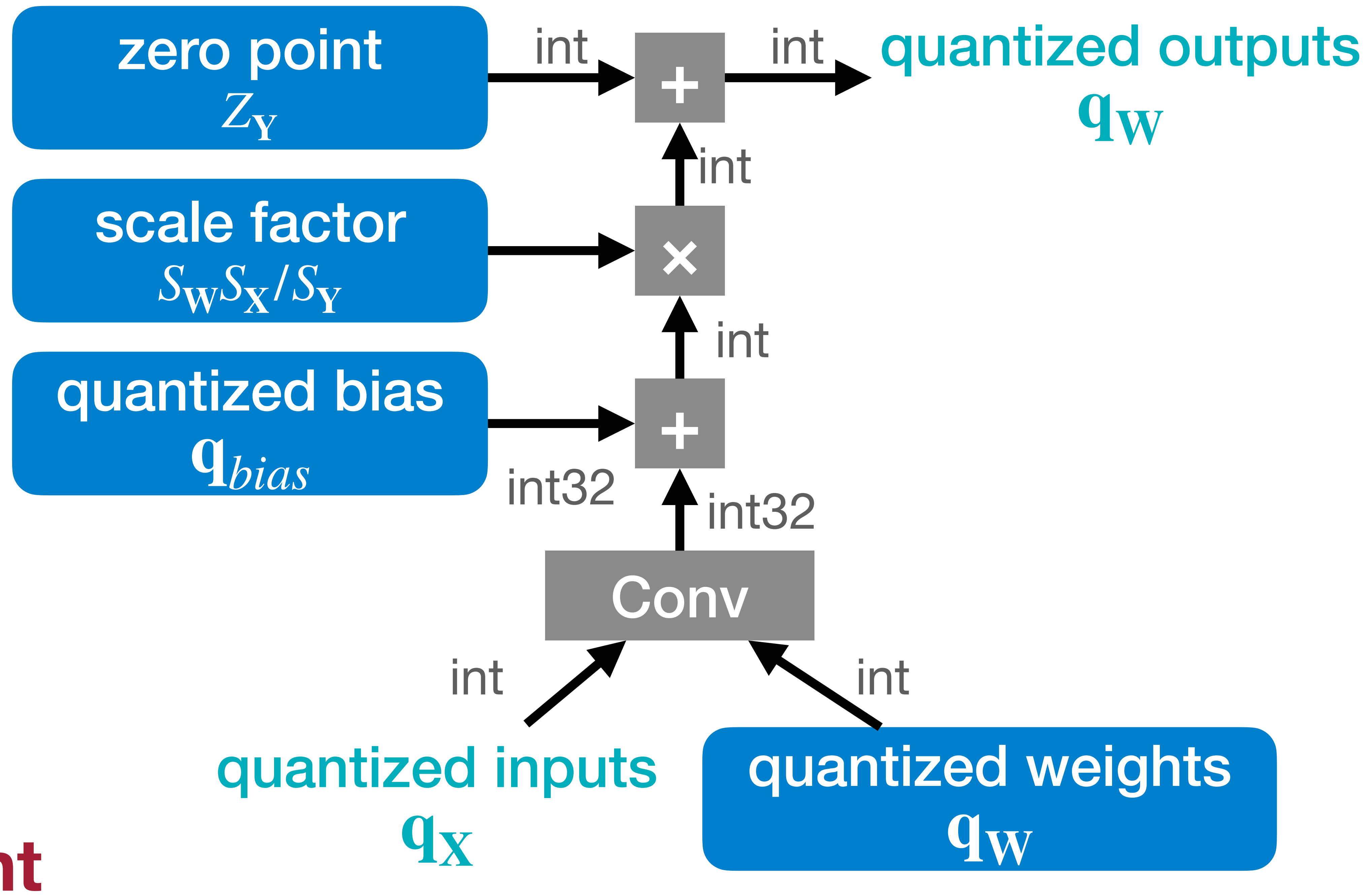
$$\mathbf{q}_{\mathbf{Y}} = \underbrace{\boxed{\frac{S_{\mathbf{W}} S_{\mathbf{X}}}{S_{\mathbf{Y}}}}}_{\text{Rescale to } N\text{-bit Int}} \left(\underbrace{\boxed{\text{Conv}(\mathbf{q}_{\mathbf{W}}, \mathbf{q}_{\mathbf{X}}) + \mathbf{q}_{bias}}}_{N\text{-bit Int Mult. 32-bit Int Add.}} \right) + \underbrace{\boxed{Z_{\mathbf{Y}}}}_{N\text{-bit Int Add}}$$

Rescale to *N*-bit Int

***N*-bit Int Mult.**
32-bit Int Add.

***N*-bit Int Add**

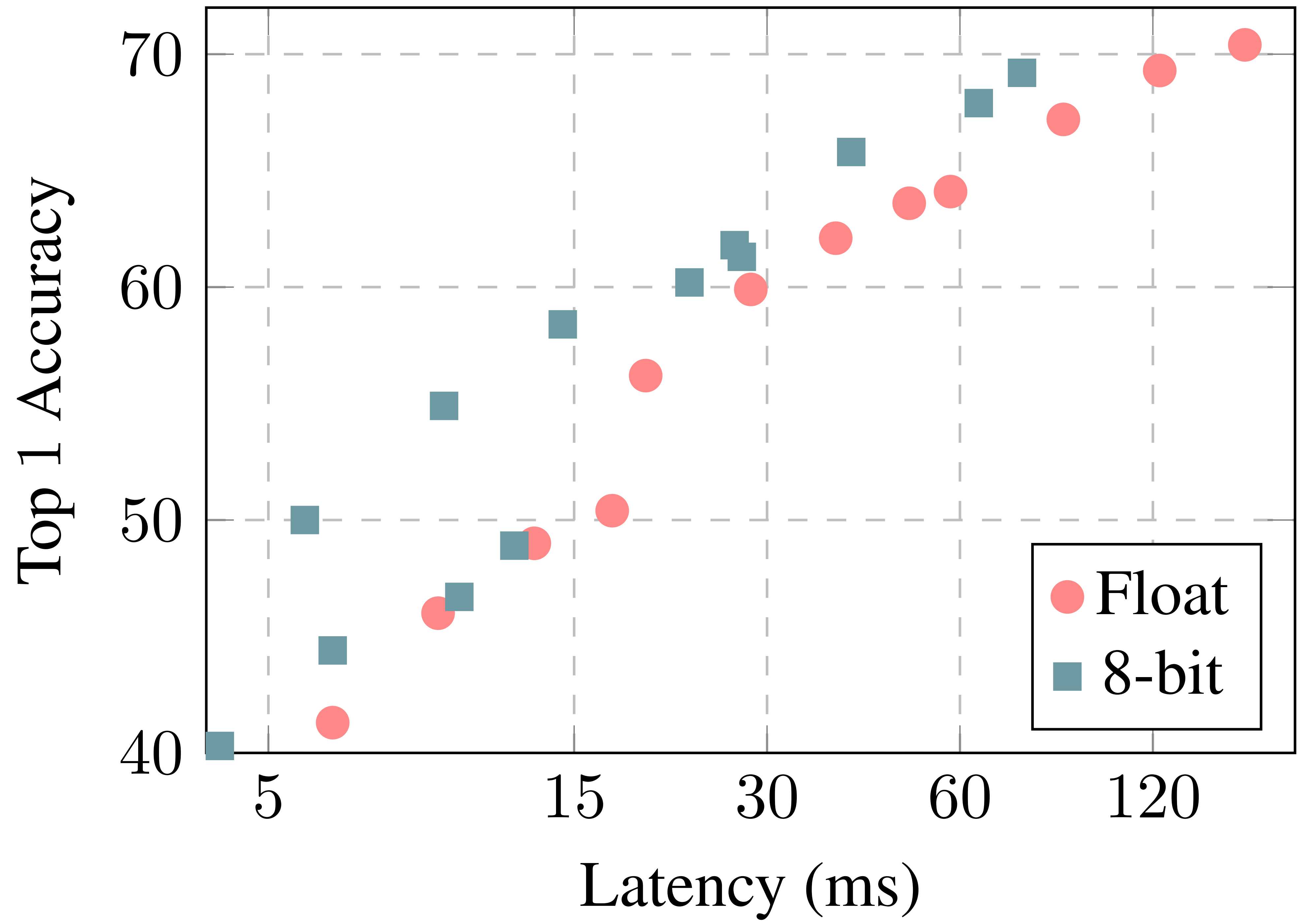
Note: both $\mathbf{q}_{\mathbf{b}}$ and $\mathbf{q}_{bias}$ are 32 bits.



INT8 Linear Quantization

An affine mapping of integers to real numbers $r = S(q - Z)$

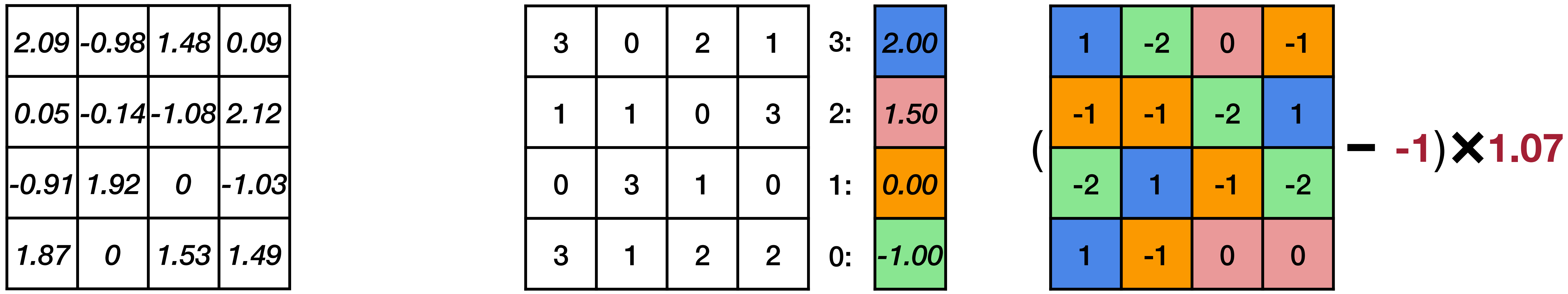
Neural Network	ResNet-50	Inception-V3
Floating-point Accuracy	76.4%	78.4%
8-bit Integer-quantized Acurracy	74.9%	75.4%



Latency-vs-accuracy tradeoff of float vs. integer-only MobileNets on ImageNet using Snapdragon 835 big cores.

Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [Jacob *et al.*, CVPR 2018]

Neural Network Quantization



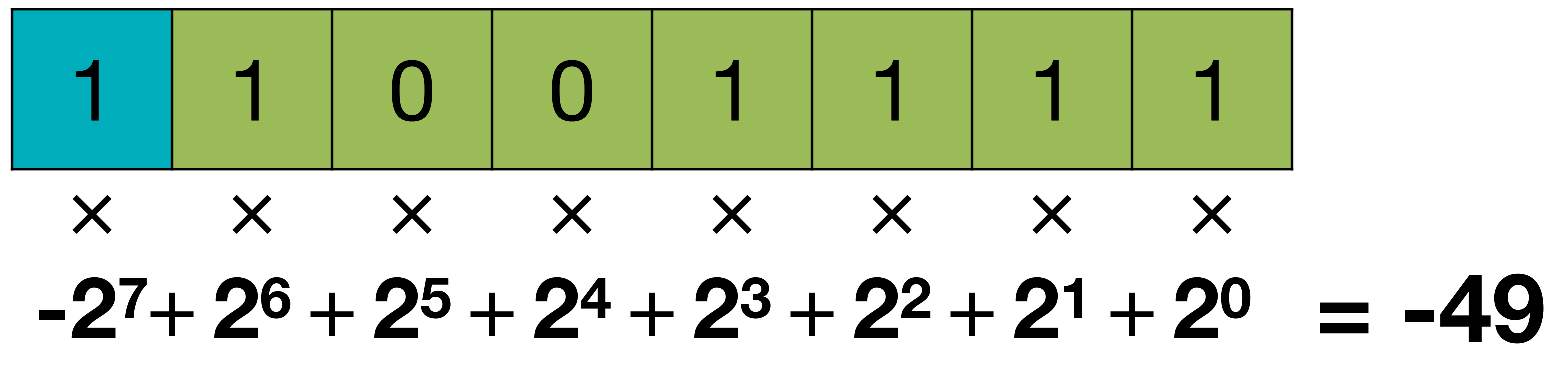
		K-Means-based Quantization	**Linear Quantization**
Storage	Floating-Point Weights	Integer Weights; Floating-Point Codebook	Integer Weights
Computation	Floating-Point Arithmetic	Floating-Point Arithmetic	Integer Arithmetic

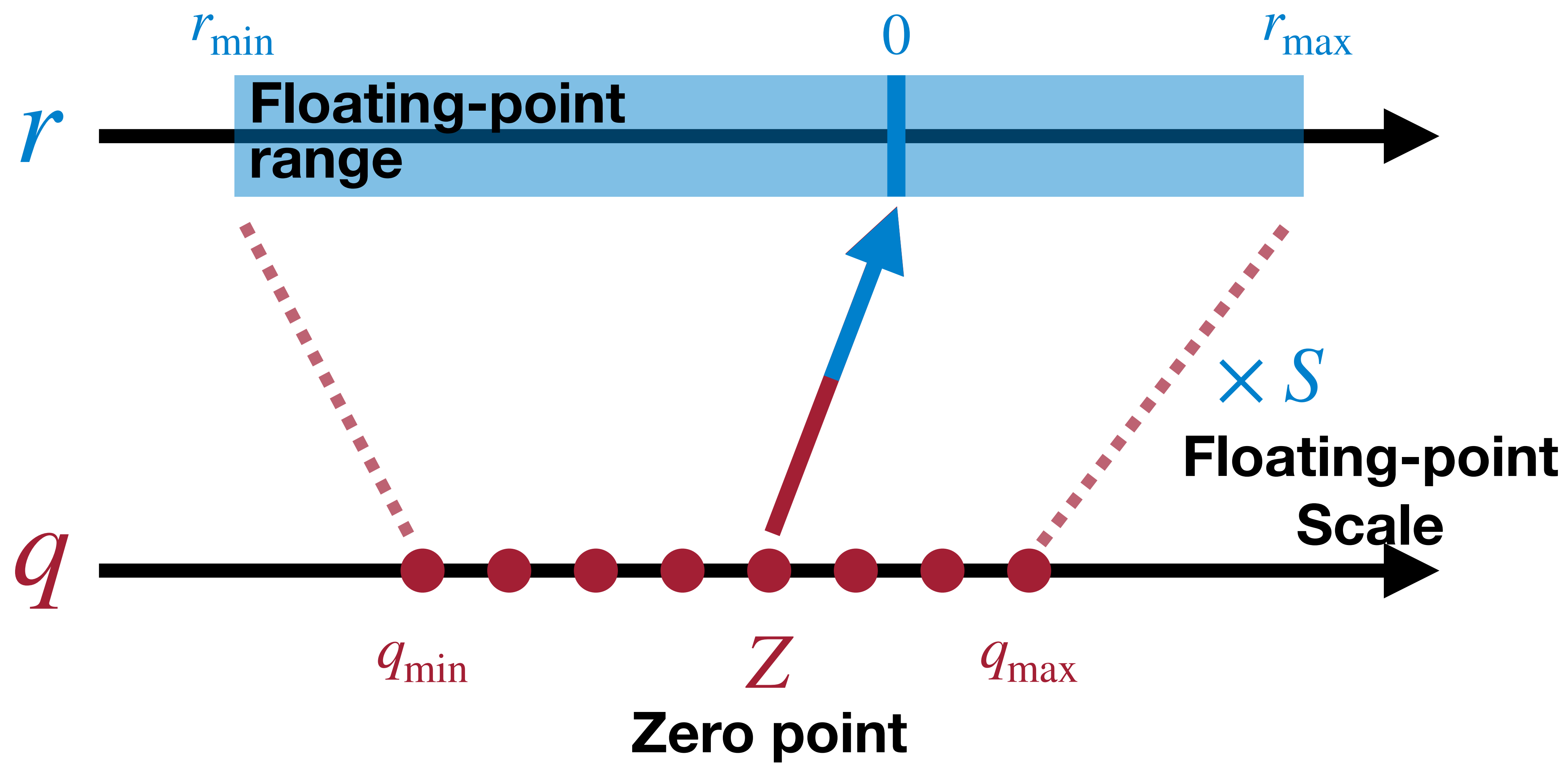
?

Summary of Today's Lecture

Today, we reviewed and learned

- the numeric data types used in the modern computing systems, including integers and floating-point numbers.
- the basic concept of **neural network quantization**: *converting the weights and activations of neural networks into a limited discrete set of numbers.*
- three types of common neural network quantization:
  - K-Means-based Quantization
  - Linear Quantization





References

1. Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey [Deng et al., IEEE 2020]
2. Computing's Energy Problem (and What We Can Do About it) [Horowitz, M., IEEE ISSCC 2014]
3. Deep Compression [Han et al., ICLR 2016]
4. Neural Network Distiller: https://intellabs.github.io/distiller/algo_quantization.html
5. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [Jacob et al., CVPR 2018]
6. BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations [Courbariaux et al., NeurIPS 2015]
7. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or –1. [Courbariaux et al., Arxiv 2016]
8. XNOR-Net: ImageNet Classification using Binary Convolutional Neural Networks [Rastegari et al., ECCV 2016]
9. Ternary Weight Networks [Li et al., Arxiv 2016]
10. Trained Ternary Quantization [Zhu et al., ICLR 2017]