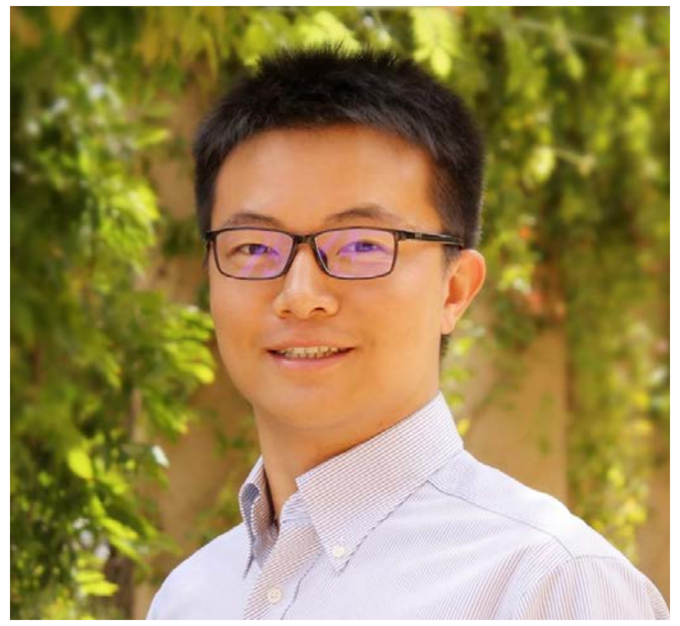
MIT HAN LAB

EfficientML.ai Lecture 06 Quantization

Part II



Song Han

Associate Professor, MIT
Distinguished Scientist, NVIDIA

@SongHan_MIT

Lecture Plan

Today we will:

1. Review Linear Quantization.

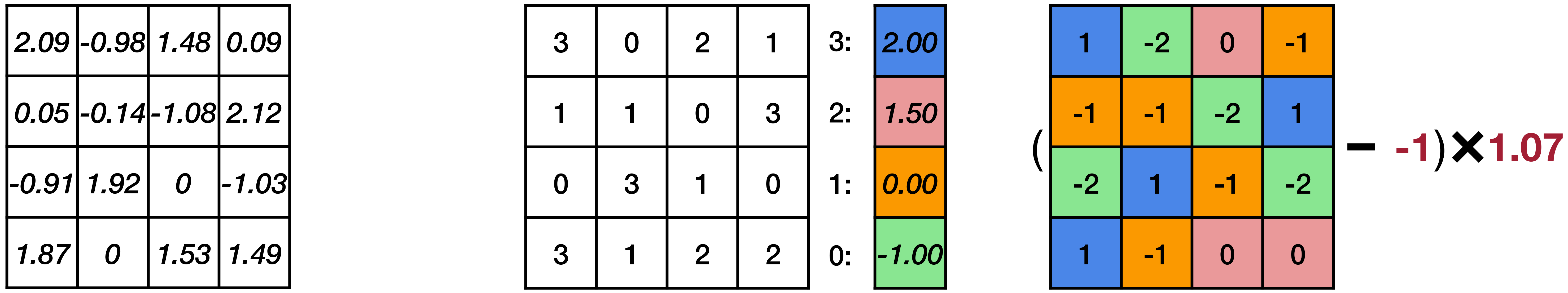
2. Introduce **Post-Training Quantization (PTQ)** that quantizes a floating-point neural network model, including: channel quantization, group quantization, and range clipping.

3. Introduce **Quantization-Aware Training (QAT)** that emulates inference-time quantization during the training/fine-tuning and recover the accuracy.

4. Introduce **binary and ternary** quantization.

5. Introduce automatic **mixed-precision** quantization.

Neural Network Quantization



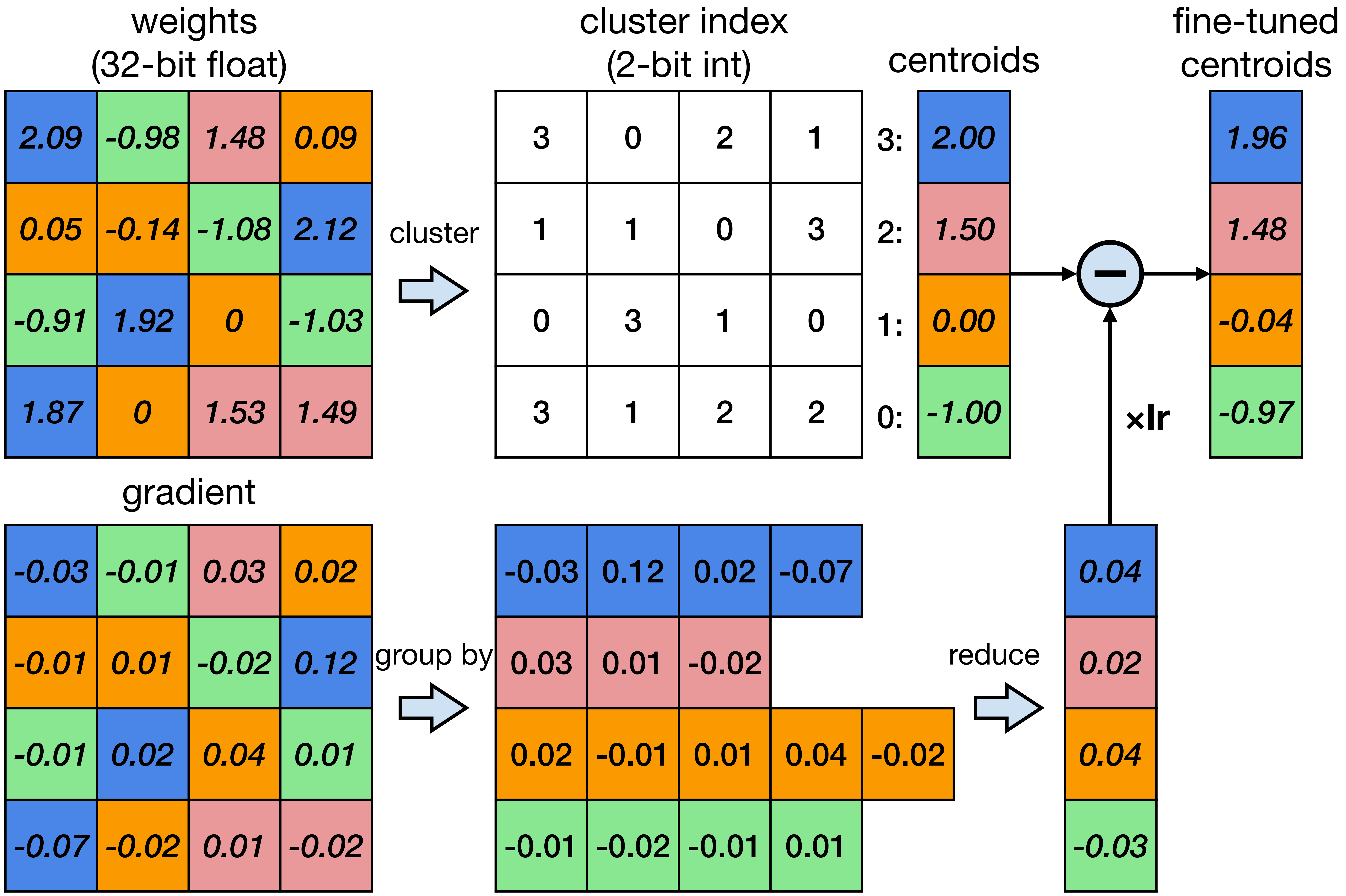
2.09	-0.98	1.48	0.09
0.05	-0.14	-1.08	2.12
-0.91	1.92	0	-1.03
1.87	0	1.53	1.49

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

$$\left(\begin{matrix}1 & -2 & 0 & -1\\ -1 & -1 & -2 & 1\\ -2 & 1 & -1 & -2\\ 1 & -1 & 0 & 0\end{matrix} - -1\right)\times 1.07$$

		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

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 index (2-bit int)

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

	centroids	fine-tuned centroids
3:	2.00	1.96
2:	1.50	1.48
1:	0.00	-0.04
0:	-1.00	-0.97

gradient

-0.03	-0.01	0.03	0.02
-0.01	0.01	-0.02	0.12
-0.01	0.02	0.04	0.01
-0.07	-0.02	0.01	-0.02

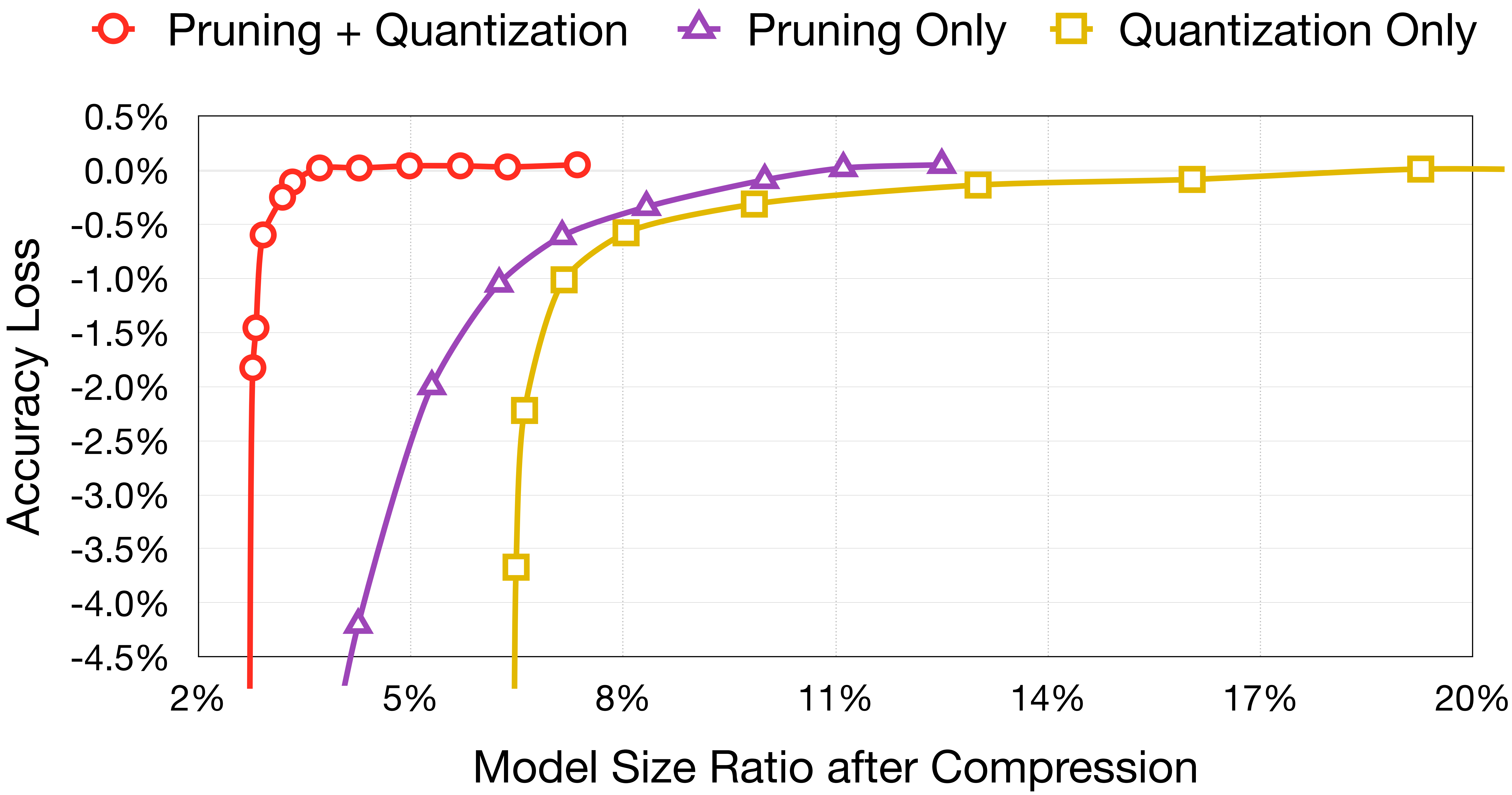
-0.03	0.12	0.02	-0.07	
0.03	0.01	-0.02		
0.02	-0.01	0.01	0.04	-0.02
-0.01	-0.02	-0.01	0.01	

0.04
0.02
0.04
-0.03

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

K-Means-based Weight Quantization

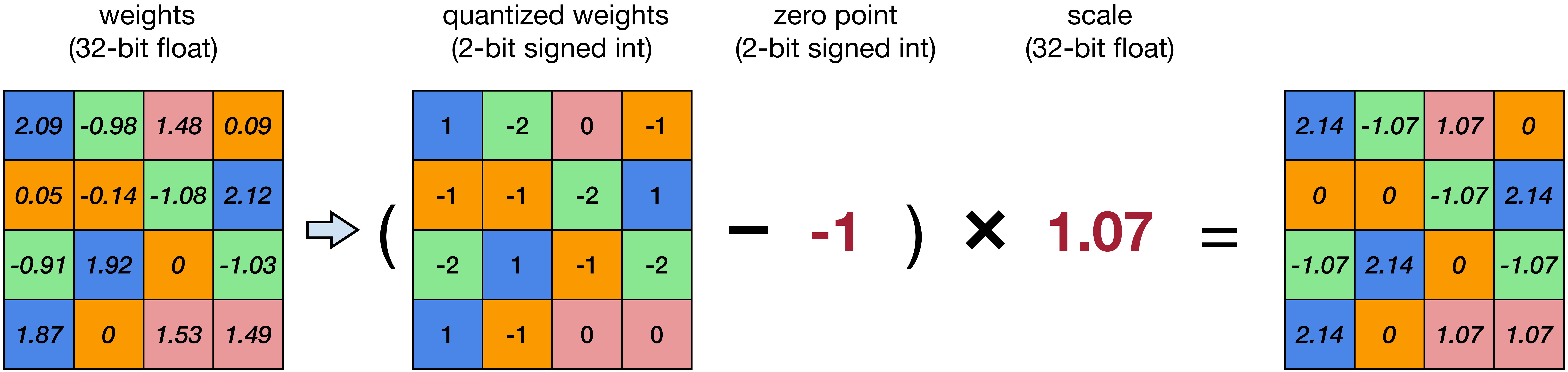
Accuracy vs. compression rate for AlexNet on ImageNet dataset



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

Linear Quantization

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

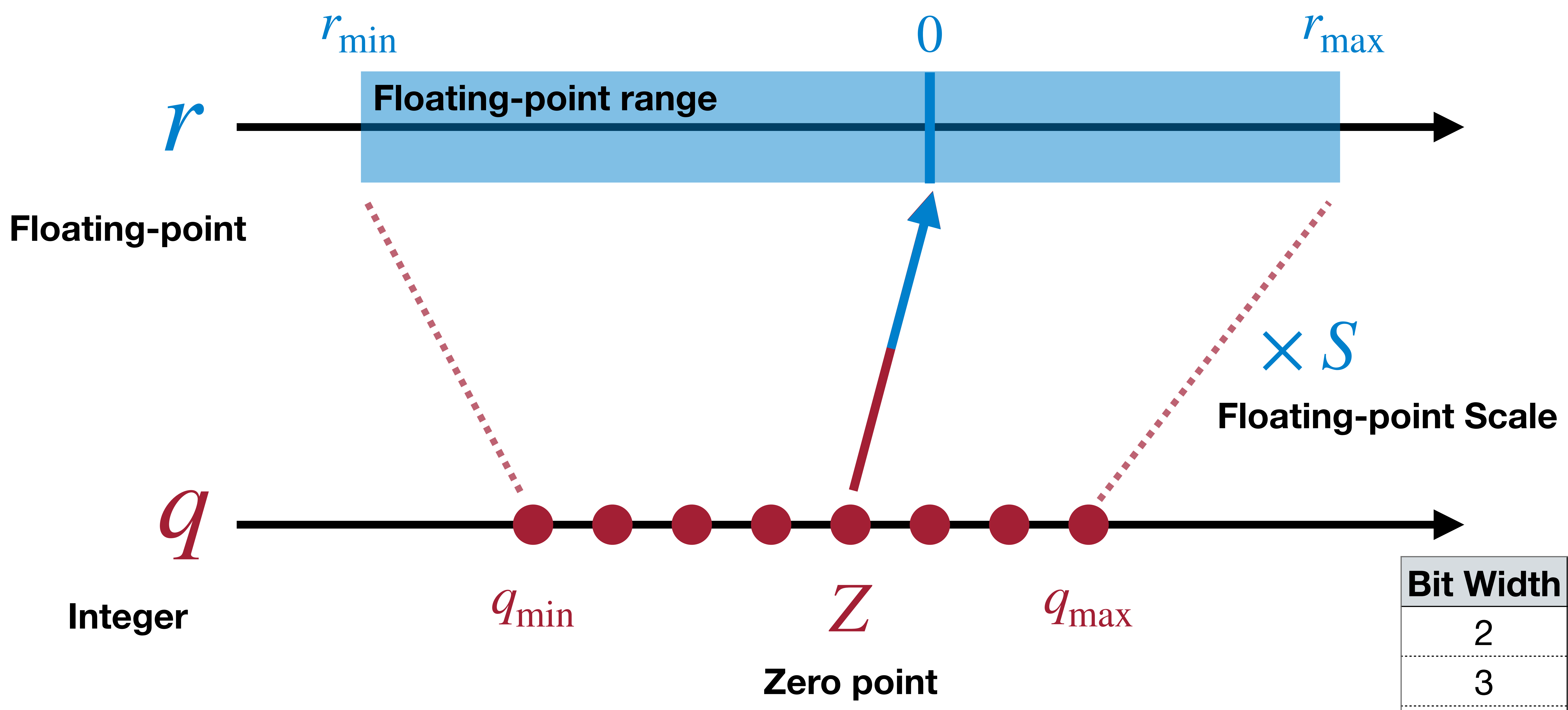


$$\underset{\text{weights (32-bit float)}}{\begin{bmatrix}2.09 & -0.98 & 1.48 & 0.09\\ 0.05 & -0.14 & -1.08 & 2.12\\ -0.91 & 1.92 & 0 & -1.03\\ 1.87 & 0 & 1.53 & 1.49\end{bmatrix}} \Rightarrow \left(\underset{\text{quantized weights (2-bit signed int)}}{\begin{bmatrix}1 & -2 & 0 & -1\\ -1 & -1 & -2 & 1\\ -2 & 1 & -1 & -2\\ 1 & -1 & 0 & 0\end{bmatrix}} - \underset{\text{zero point (2-bit signed int)}}{-1} \right) \times \underset{\text{scale (32-bit float)}}{1.07} = \begin{bmatrix}2.14 & -1.07 & 1.07 & 0\\ 0 & 0 & -1.07 & 2.14\\ -1.07 & 2.14 & 0 & -1.07\\ 2.14 & 0 & 1.07 & 1.07\end{bmatrix}$$

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

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]

Linear Quantized Fully-Connected Layer

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

- Consider the following fully-connected layer.

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

$$\Big\downarrow \quad \begin{aligned} &Z_{\mathbf{W}} = 0 \\ &Z_{\mathbf{b}} = 0, \quad S_{\mathbf{b}} = S_{\mathbf{W}} S_{\mathbf{X}} \\ &\mathbf{q}_{bias} = \mathbf{q}_{\mathbf{b}} - Z_{\mathbf{X}} \mathbf{q}_{\mathbf{W}} \end{aligned}$$

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

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.

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

$$\Big\downarrow \quad \begin{aligned} & Z_{\mathbf{W}} = 0 \\ & Z_{\mathbf{b}} = 0, \quad S_{\mathbf{b}} = S_{\mathbf{W}} S_{\mathbf{X}} \\ & \mathbf{q}_{bias} = \mathbf{q}_{\mathbf{b}} - \boxed{\text{Conv}\left(\mathbf{q}_{\mathbf{W}}, Z_{\mathbf{X}}\right)} \end{aligned}$$

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

Rescale to *N*-bit Int

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

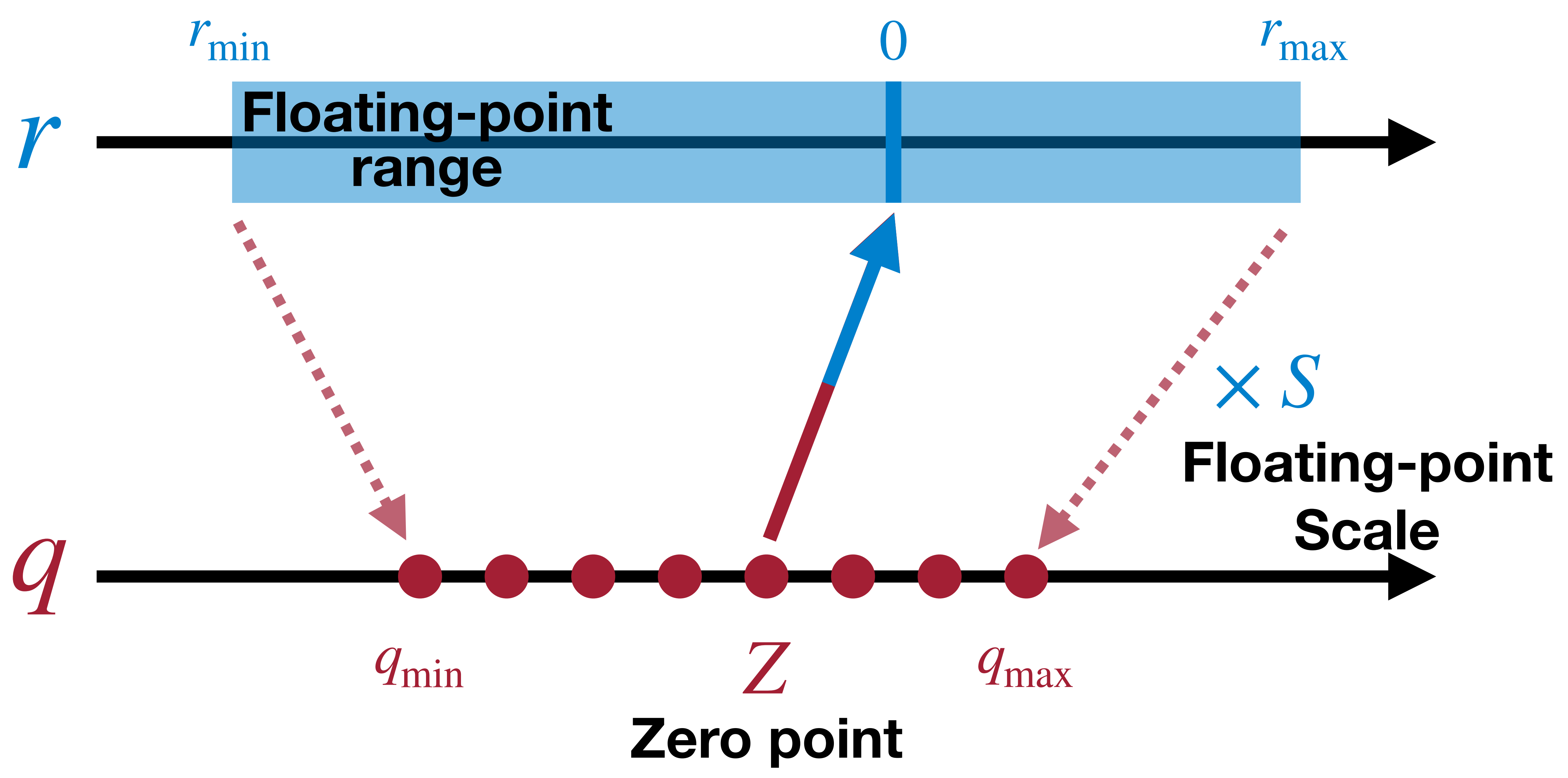
***N*-bit Int Add**

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

Scale and Zero Point of Linear Quantization

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

Asymmetric Linear Quantization



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

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

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

$$= 1.07$$

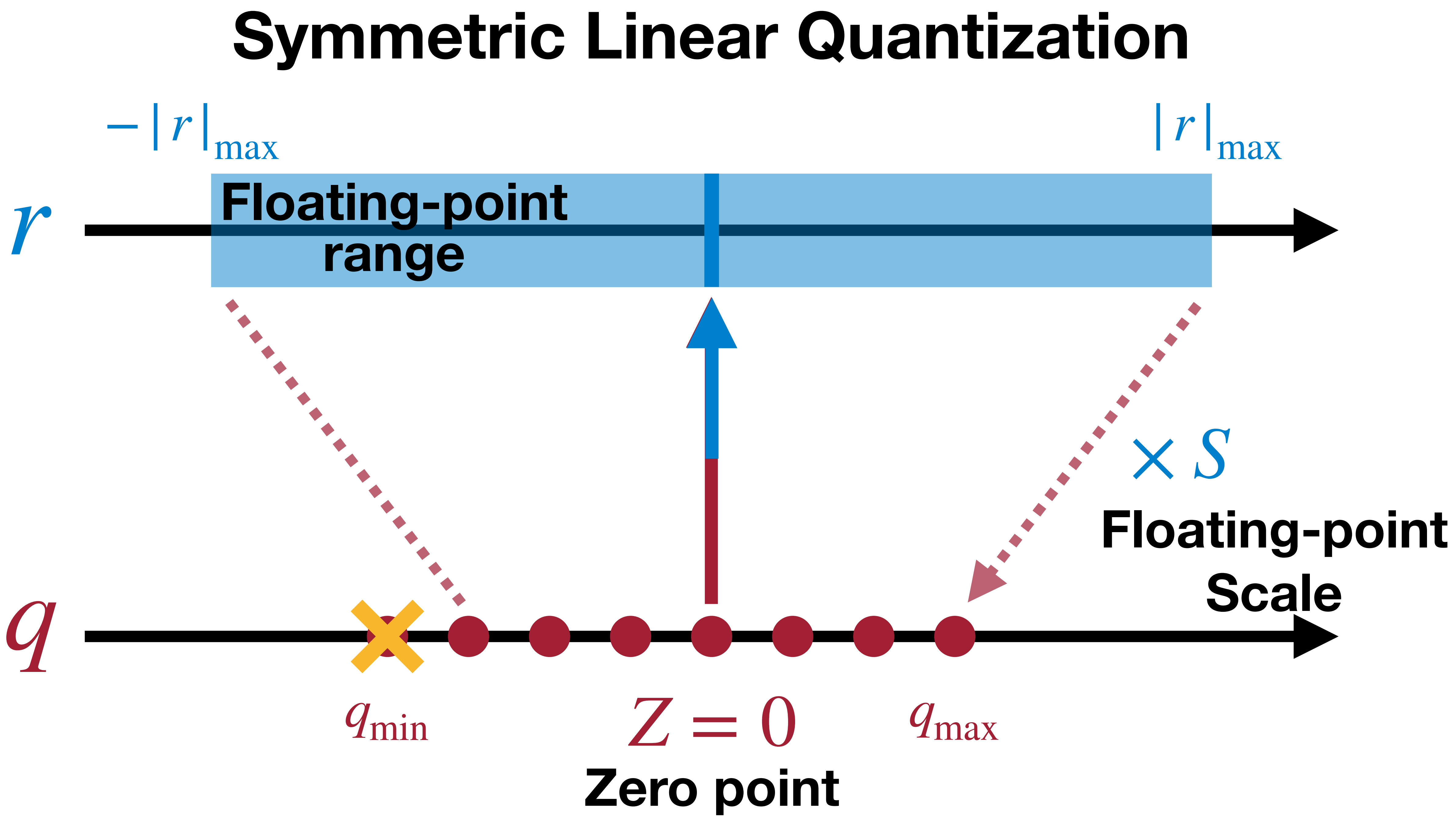
$$Z = q_{\min} - \frac{r_{\min}}{S}$$

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

$$= -1$$

Scale and 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

$$S = \frac{|r|_{\max}}{q_{\max}} = \frac{2.12}{1} = 2.12$$

$$Z = 0$$

Post-Training Quantization

How should we get the optimal linear quantization parameters (S, Z)?

Topic I: Quantization Granularity
Topic II: Dynamic Range Clipping
Topic III: Rounding

Post-Training Quantization

How should we get the optimal linear quantization parameters (S, Z)?

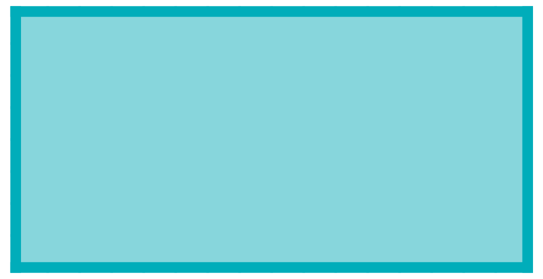
Topic I: Quantization Granularity

Topic II: Dynamic Range Clipping

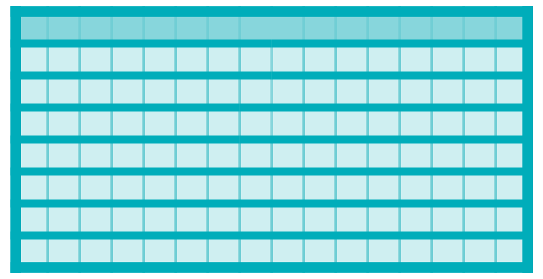
Topic III: Rounding

Quantization Granularity

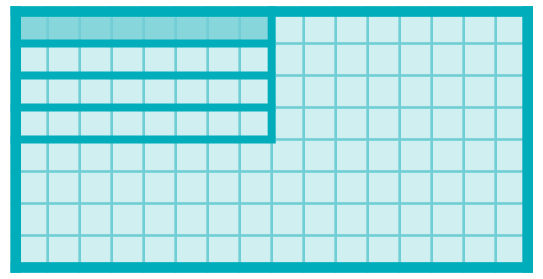
- Per-Tensor Quantization



- Per-Channel Quantization



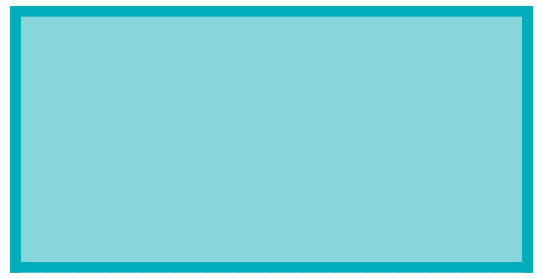
- Group Quantization



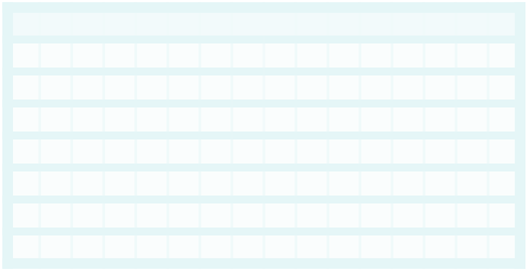
    - Per-Vector Quantization
    - Shared Micro-exponent (MX) data type

Quantization Granularity

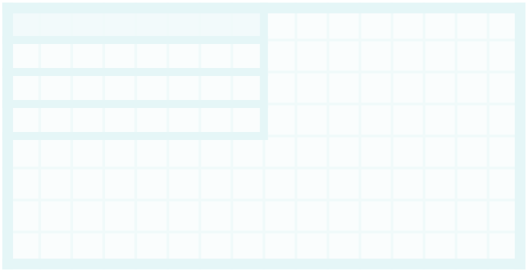
- **Per-Tensor Quantization**



- Per-Channel Quantization

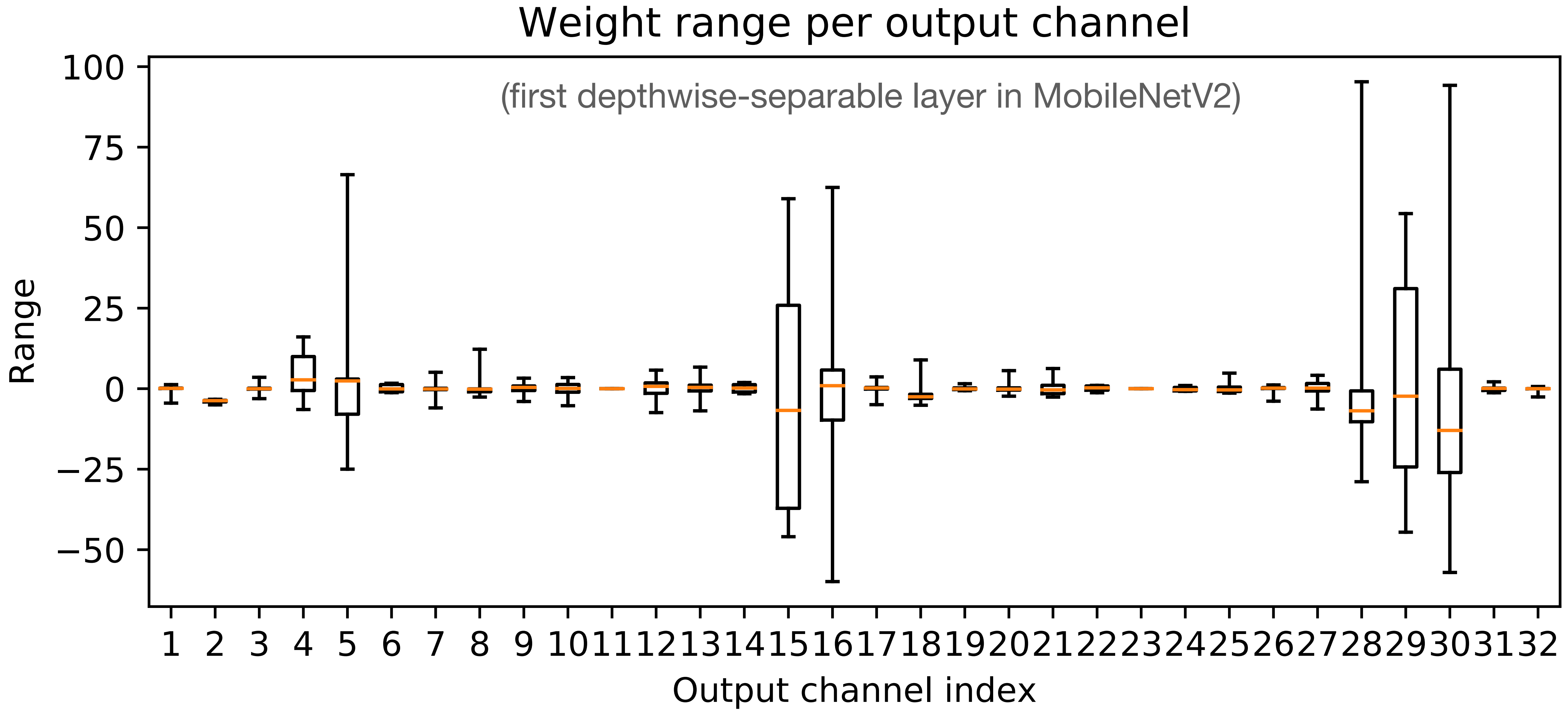


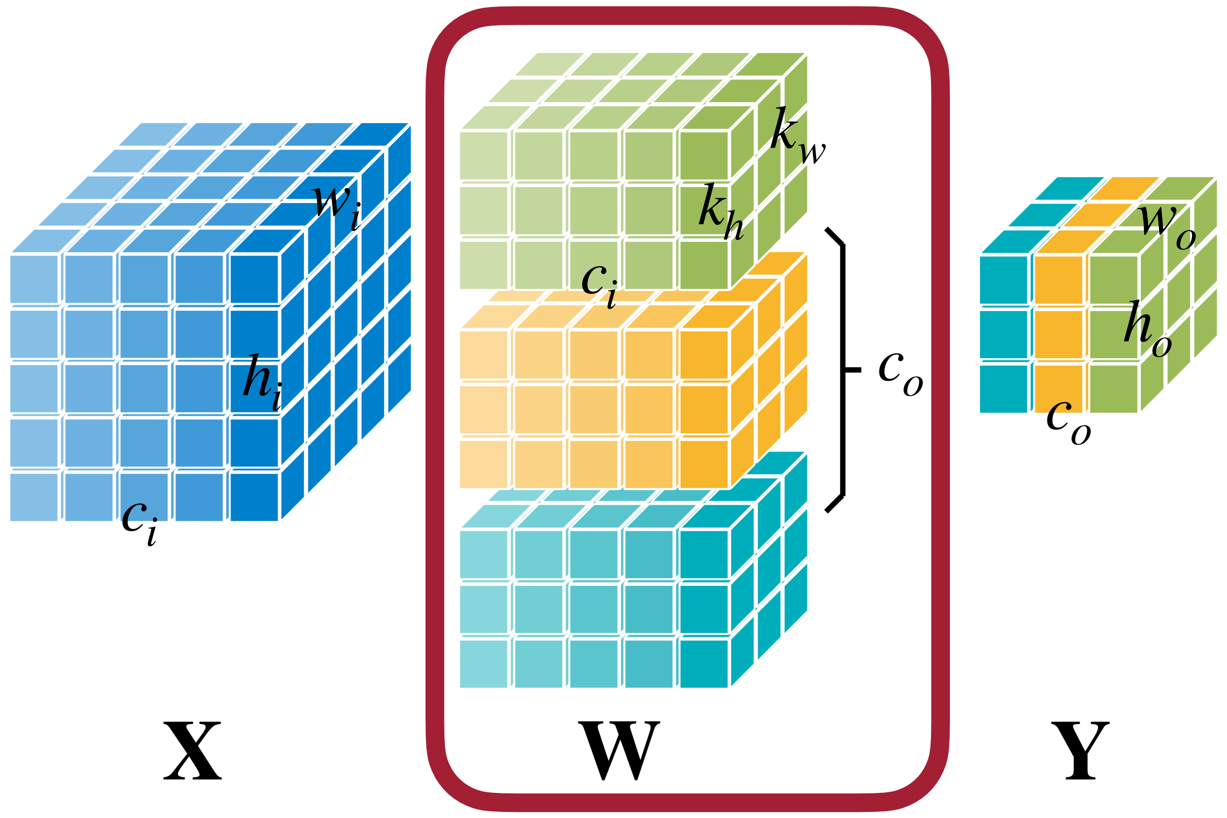
- Group Quantization



  - Per-Vector Quantization
  - Shared Micro-exponent (MX) data type

Symmetric Linear Quantization on Weights



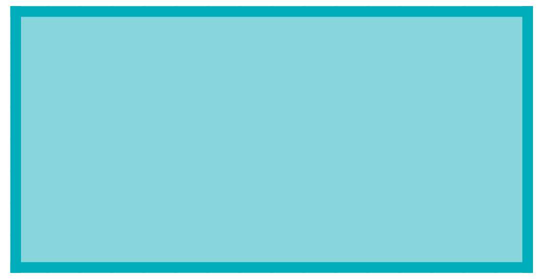


- $|r|_{\max} = |\mathbf{W}|_{\max}$
- Using *single* scale S for whole weight tensor (**Per-Tensor Quantization**)
  - works well for large models
  - accuracy drops for small models
- Common failure results from
  - large differences (more than 100×) in ranges of weights for different output channels — outlier weight
- Solution: **Per-Channel Quantization**

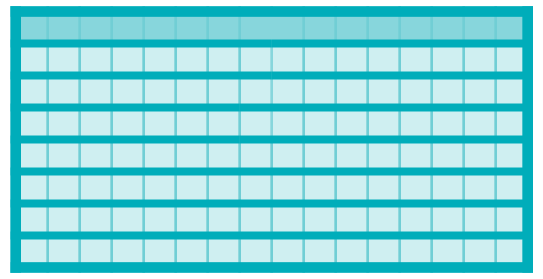
Data-Free Quantization Through Weight Equalization and Bias Correction [Markus *et al.*, ICCV 2019]
Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [Jacob *et al.*, CVPR 2018]

Quantization Granularity

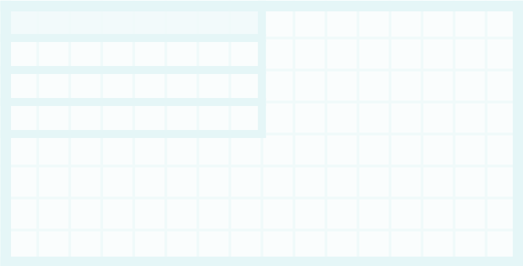
- Per-Tensor Quantization



- **Per-Channel Quantization**



- Group Quantization



  - Per-Vector Quantization
  - Shared Micro-exponent (MX) data type

Per-Channel Weight Quantization

Example: 2-bit linear quantization

$$\begin{array}{c|c|c|c|c|}
 & \multicolumn{4}{c}{ic} \\
\hline
 & 2.09 & -0.98 & 1.48 & 0.09 \\
\hline
 & 0.05 & -0.14 & -1.08 & 2.12 \\
\hline
oc & -0.91 & 1.92 & 0 & -1.03 \\
\hline
 & 1.87 & 0 & 1.53 & 1.49 \\
\hline
\end{array}$$

Per-Channel Quantization

Per-Tensor Quantization

Per-Channel Weight Quantization

Example: 2-bit linear quantization

ic

oc

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

Per-Channel Quantization

Per-Tensor Quantization

$$|r|_{\max} = 2.12$$

$$S = \frac{|r|_{\max}}{q_{\max}} = \frac{2.12}{2^{2-1}-1} = 2.12$$

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

Quantized

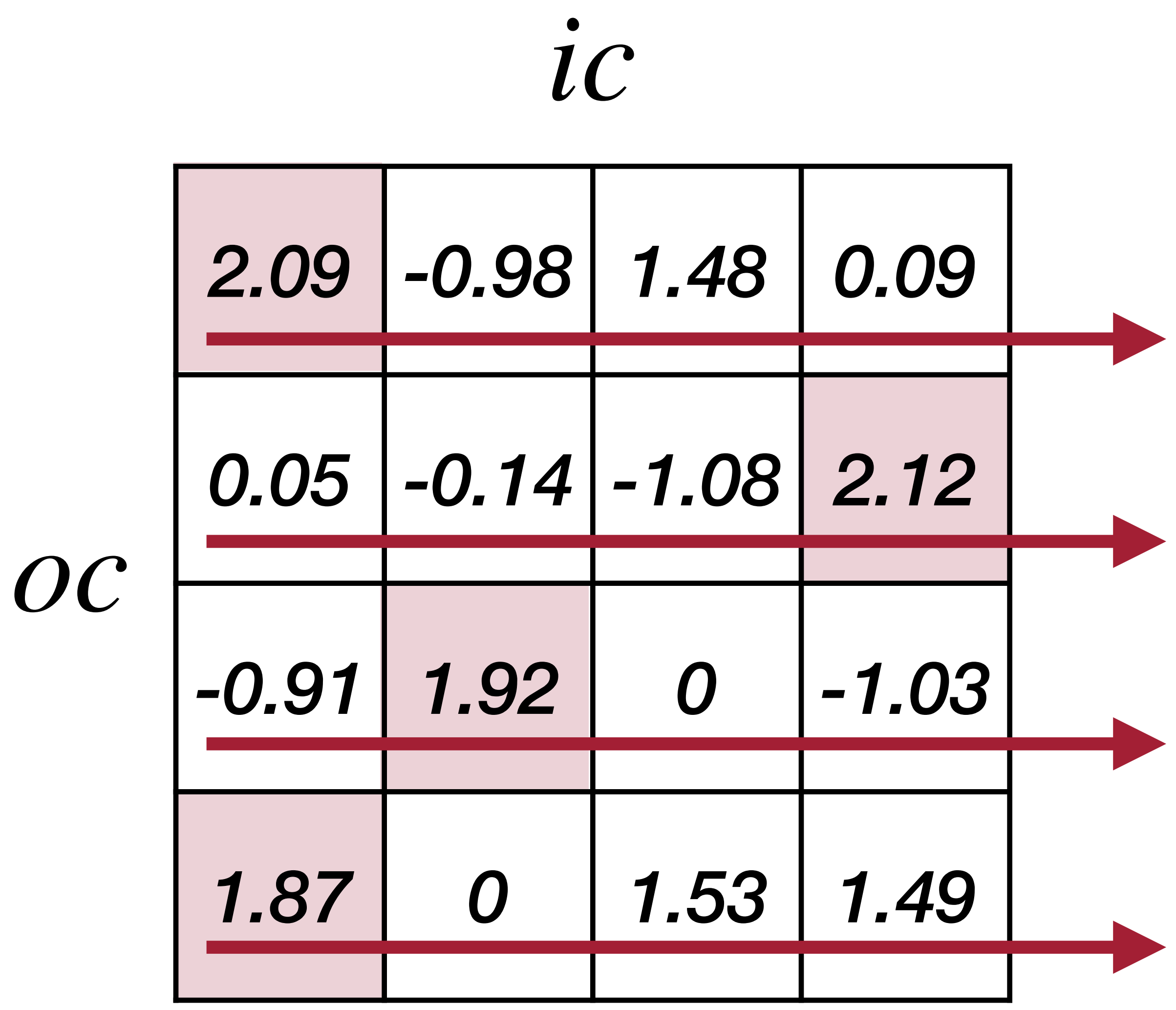
2.12	0	2.12	0
0	0	-2.12	2.12
0	2.12	0	0
2.12	0	2.12	2.12

Reconstructed

$$\|\mathbf{W} - S\mathbf{q}_{\mathbf{W}}\|_F = 2.28$$

Per-Channel Weight Quantization

Example: 2-bit linear quantization



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

Per-Channel Quantization

$$|r|_{\max} = 2.09 \qquad S_0 = 2.09$$

$$|r|_{\max} = 2.12 \qquad S_1 = 2.12$$

$$|r|_{\max} = 1.92 \qquad S_2 = 1.92$$

$$|r|_{\max} = 1.87 \qquad S_3 = 1.87$$

Per-Tensor Quantization

$$|r|_{\max} = 2.12$$

$$S = \frac{|r|_{\max}}{q_{\max}} = \frac{2.12}{2^{2-1} - 1} = 2.12$$

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

Quantized

2.12	0	2.12	0
0	0	-2.12	2.12
0	2.12	0	0
2.12	0	2.12	2.12

Reconstructed

$$\|\mathbf{W} - S\mathbf{q_W}\|_F = 2.28$$

Per-Channel Weight Quantization

Example: 2-bit linear quantization

	ic			
oc	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

Per-Channel Quantization

$$|r|_{\max} = 2.09 \qquad S_0 = 2.09$$

$$|r|_{\max} = 2.12 \qquad S_1 = 2.12$$

$$|r|_{\max} = 1.92 \qquad S_2 = 1.92$$

$$|r|_{\max} = 1.87 \qquad S_3 = 1.87$$

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

Quantized

2.09	0	2.09	0
0	0	-2.12	2.12
0	1.92	0	-1.92
1.87	0	1.87	1.87

Reconstructed

$$\|\mathbf{W} - \mathbf{S} \odot \mathbf{q}_{\mathbf{W}}\|_F = 2.08$$

$$<$$

Per-Tensor Quantization

$$|r|_{\max} = 2.12$$

$$S = \frac{|r|_{\max}}{q_{\max}} = \frac{2.12}{2^{2-1} - 1} = 2.12$$

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

Quantized

2.12	0	2.12	0
0	0	-2.12	2.12
0	2.12	0	0
2.12	0	2.12	2.12

Reconstructed

$$\|\mathbf{W} - S\mathbf{q}_{\mathbf{W}}\|_F = 2.28$$

Per-Channel Weight Quantization

Example: 2-bit linear quantization

$$\begin{array}{c|cccc} & & ic & & \\ \hline & 2.09 & -0.98 & 1.48 & 0.09 \\ & 0.05 & -0.14 & -1.08 & 2.12 \\ oc & -0.91 & 1.92 & 0 & -1.03 \\ & 1.87 & 0 & 1.53 & 1.49 \end{array}$$

Per-Channel Quantization

$$|r|_{\max} = 2.09 \qquad S_0 = 2.09$$

$$|r|_{\max} = 2.12 \qquad S_1 = 2.12$$

$$|r|_{\max} = 1.92 \qquad S_2 = 1.92$$

$$|r|_{\max} = 1.87 \qquad S_3 = 1.87$$

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

Quantized

2.09	0	2.09	0
0	0	-2.12	2.12
0	1.92	0	-1.92
1.87	0	1.87	1.87

Reconstructed

$$\|\mathbf{W} - \mathbf{S} \odot \mathbf{q_W}\|_F = 2.08$$

<

Per-Tensor Quantization

$$|r|_{\max} = 2.12$$

$$S = \frac{|r|_{\max}}{q_{\max}} = \frac{2.12}{2^{2-1} - 1} = 2.12$$

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

Quantized

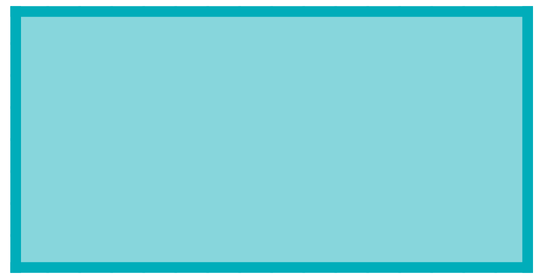
2.12	0	2.12	0
0	0	-2.12	2.12
0	2.12	0	0
2.12	0	2.12	2.12

Reconstructed

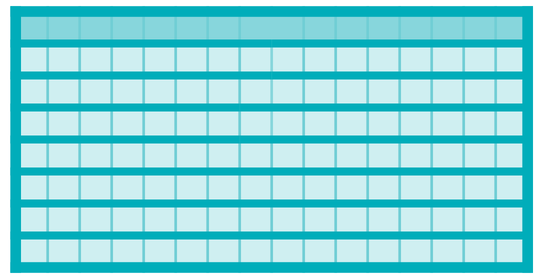
$$\|\mathbf{W} - S\mathbf{q_W}\|_F = 2.28$$

Quantization Granularity

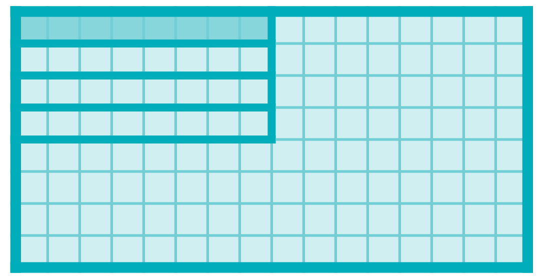
- Per-Tensor Quantization



- Per-Channel Quantization



- **Group Quantization**



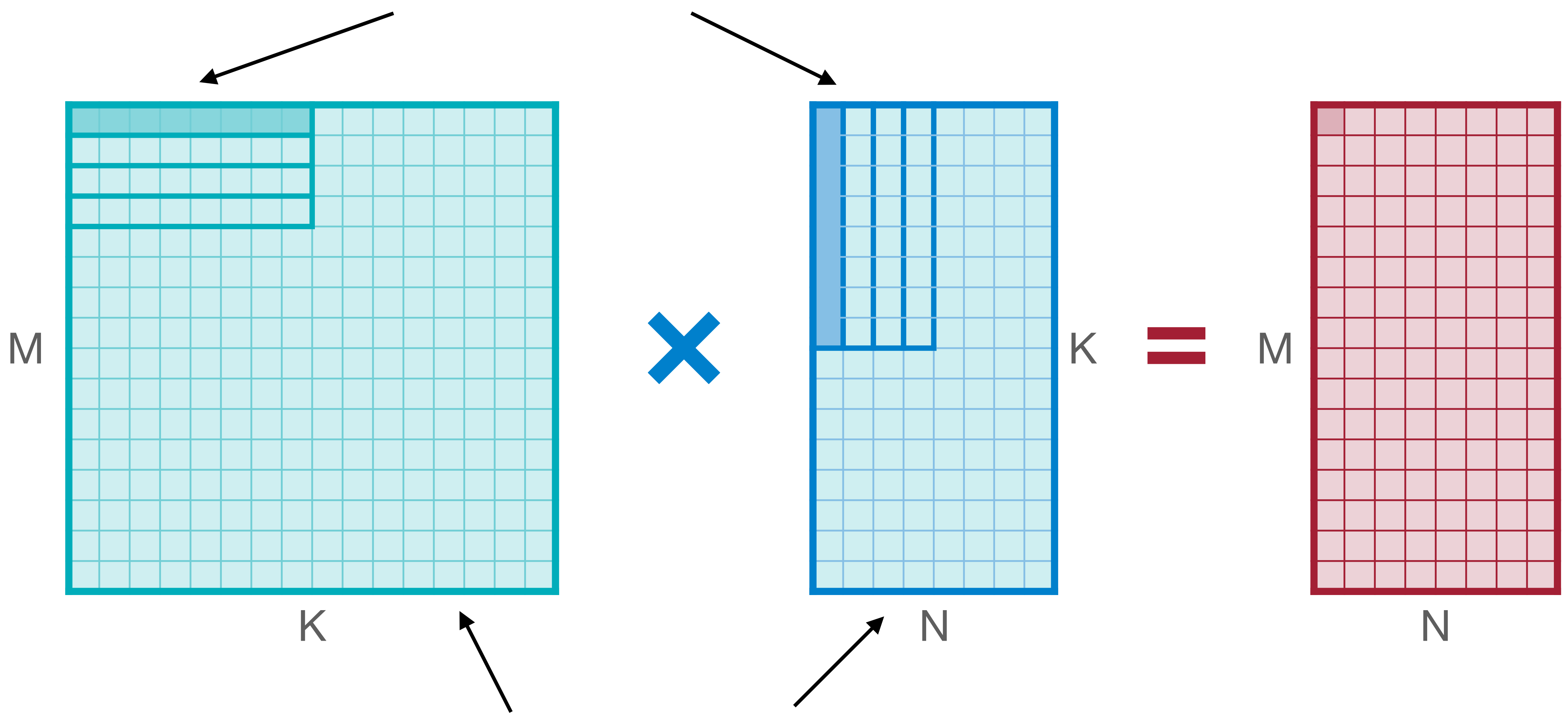
  - **Per-Vector Quantization**
  - **Shared Micro-exponent (MX) data type**

VS-Quant: Per-vector Scaled Quantization

Hierarchical scaling factor

- $r = S(q - Z) \rightarrow r = \gamma \cdot S_q(q - Z)$
  - γ is a floating-point coarse grained scale factor
  - S_q is an integer per-vector scale factor
  - achieves a balance between accuracy and hardware efficiency by
    - less expensive integer scale factors at finer granularity
    - more expensive floating-point scale factors at coarser granularity
- Memory Overhead of two-level scaling:
  - Given 4-bit quantization with 4-bit per-vector scale for every 16 elements, the effective bit width is 4 + 4 / 16 = 4.25 bits.

scale factor S_q for each vector

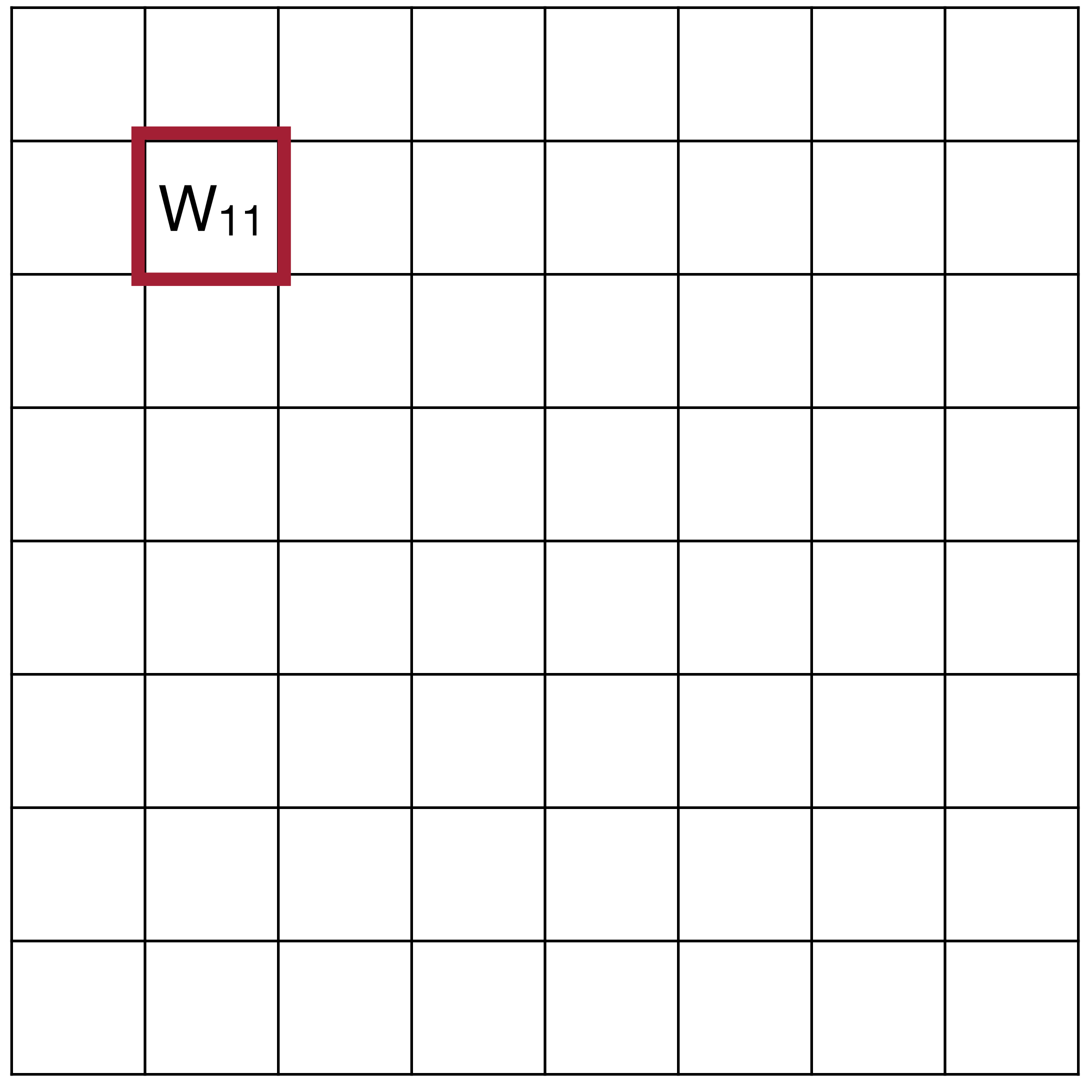


another scale factor γ for each tensor

VS-Quant: Per-Vector Scaled Quantization for Accurate Low-Precision Neural Network Inference [Steve Dai, *et al.*]

Group Quantization

Multi-level scaling scheme



$$r = (q - z) \cdot s \rightarrow$$

$$r = (q - z) \cdot s_{l_0} \cdot s_{l_1} \cdot \cdots$$

r : real number value

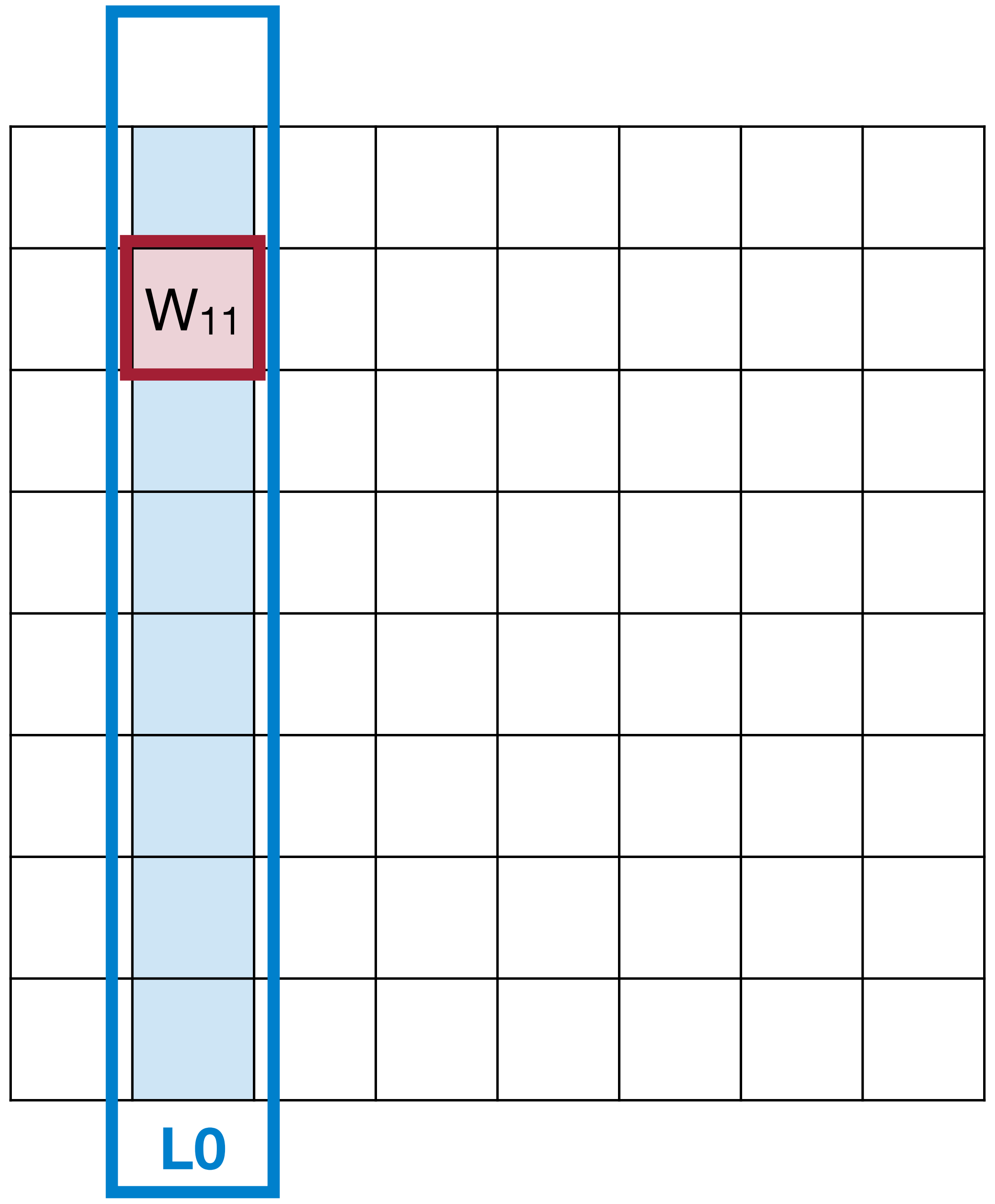
q : quantized value

z : zero point ($z = 0$ is symmetric quantization)

s : scale factors of different levels

Group Quantization

Multi-level scaling scheme



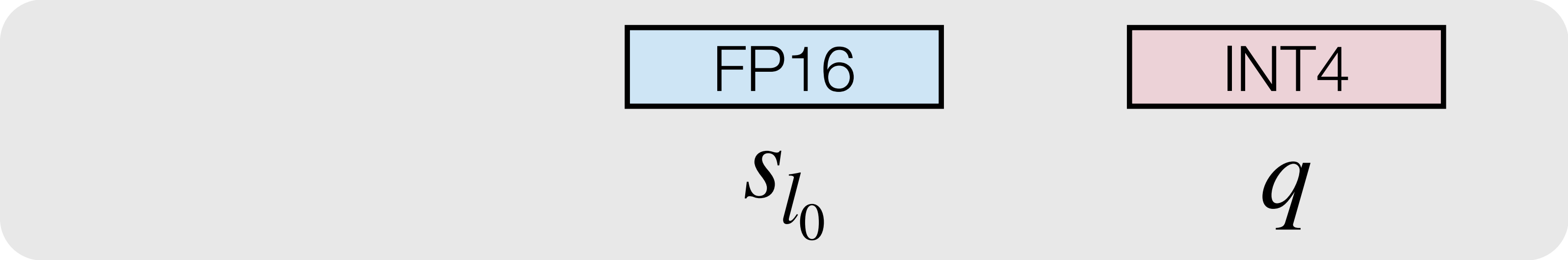
$$r = (q - z) \cdot s_{l_0} \cdot s_{l_1} \cdot \cdots$$

r : real number value

q : quantized value

z : zero point ($z = 0$ is symmetric quantization)

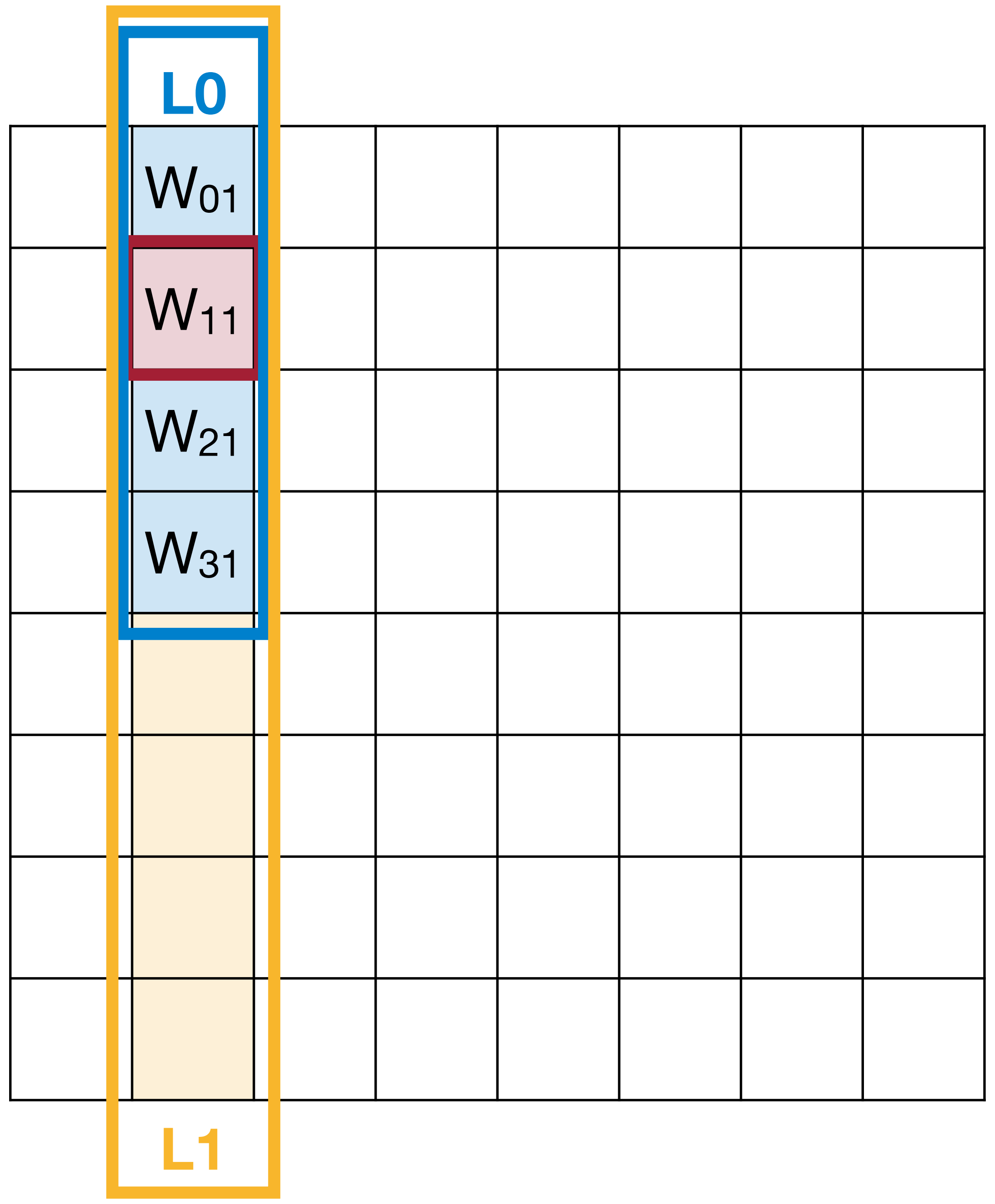
s : scale factors of different levels



Quantization Approach	Data Type	L0 Group Size	L0 Scale Data Type	L1 Group Size	L1 Scale Data Type	Effective Bit Width
Per-Channel Quant	INT4	Per Channel	FP16	-	-	4

Group Quantization

Multi-level scaling scheme



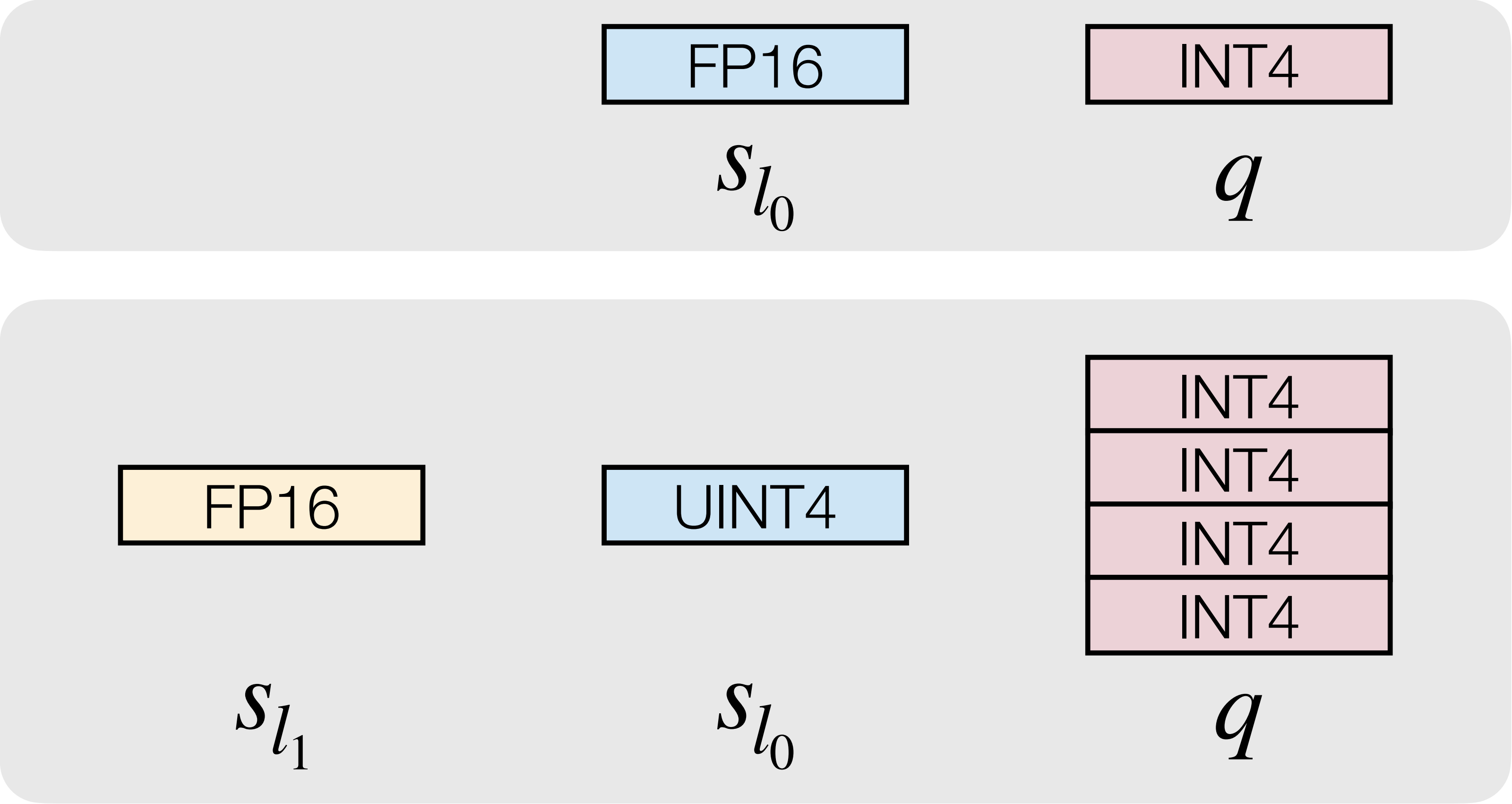
$$r = (q - z) \cdot s_{l_0} \cdot s_{l_1} \cdot \cdots$$

r : real number value

q : quantized value

z : zero point ($z = 0$ is symmetric quantization)

s : scale factors of different levels

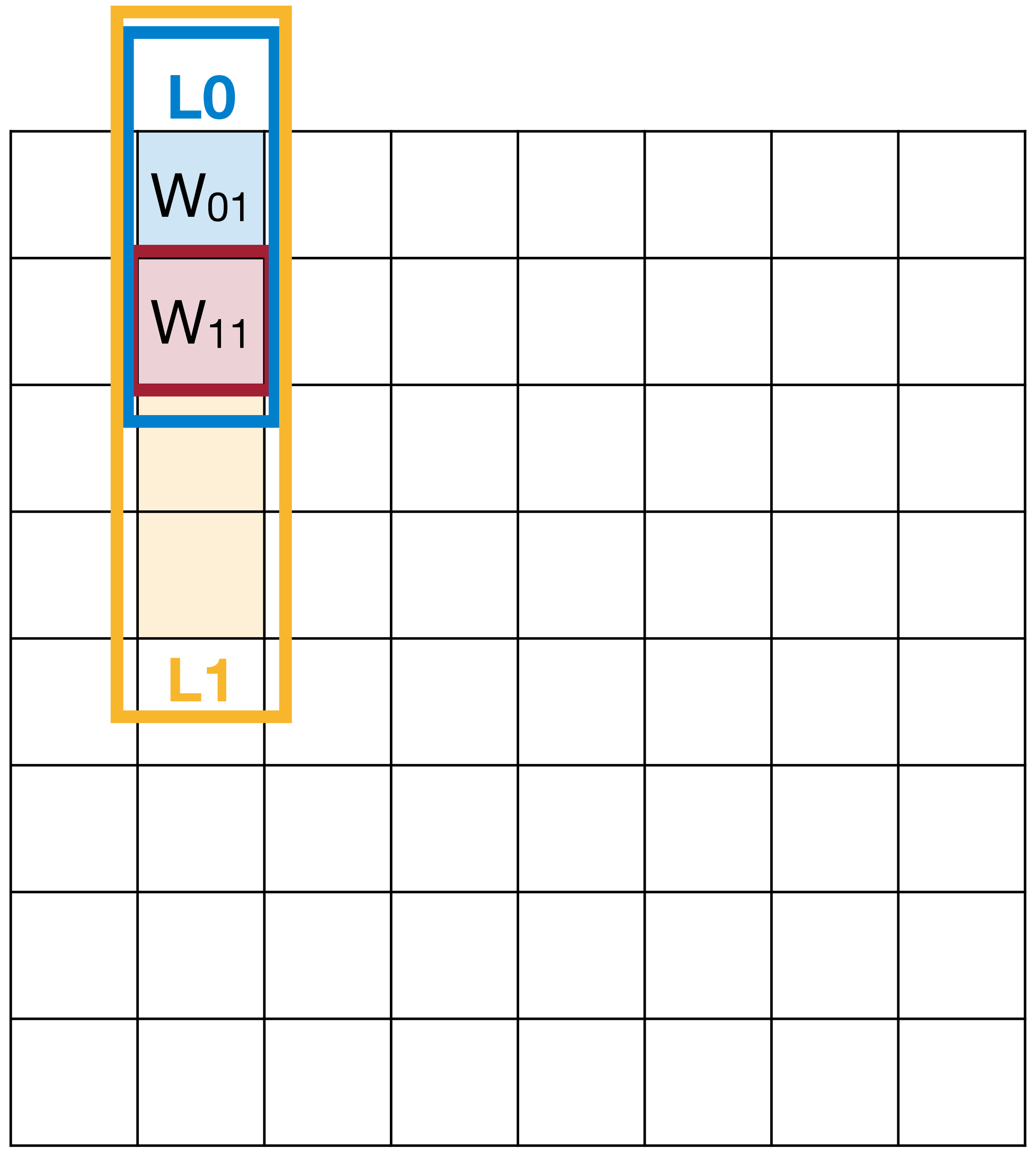


Quantization Approach	Data Type	L0 Group Size	L0 Scale Data Type	L1 Group Size	L1 Scale Data Type	Effective Bit Width
Per-Channel Quant	INT4	Per Channel	FP16	-	-	4
VSQ	INT4	16	UINT4	Per Channel	FP16	4+4/16=4.25

VS-Quant: Per-Vector Scaled Quantization for Accurate Low-Precision Neural Network Inference [Steve Dai, *et al.*]

Group Quantization

Multi-level scaling scheme



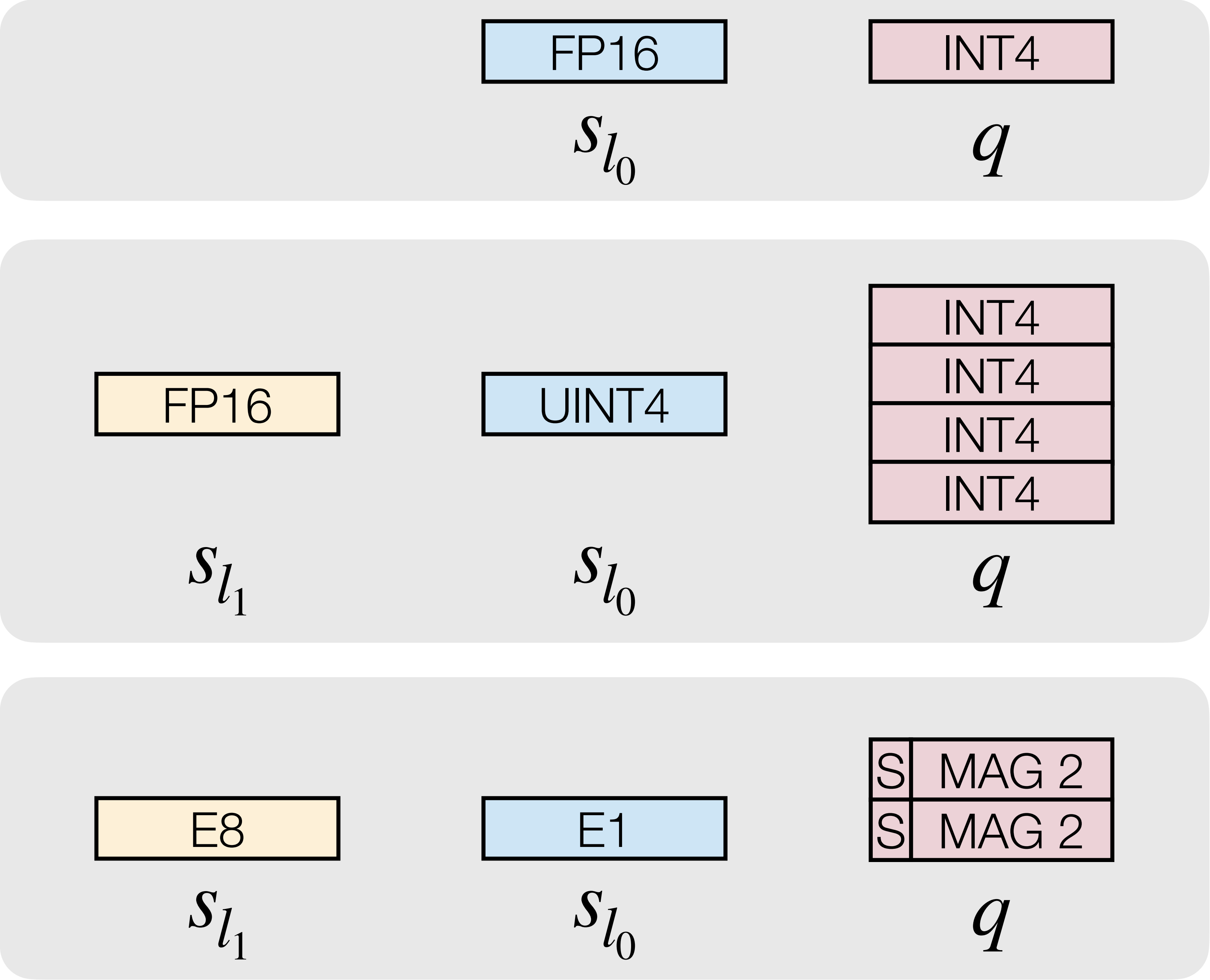
$$r = (q - z) \cdot s_{l_0} \cdot s_{l_1} \cdot \cdots$$

r : real number value

q : quantized value

z : zero point ($z = 0$ is symmetric quantization)

s : scale factors of different levels



Quantization Approach	Data Type	L0 Group Size	L0 Scale Data Type	L1 Group Size	L1 Scale Data Type	Effective Bit Width
Per-Channel Quant	INT4	Per Channel	FP16	-	-	4
VSQ	INT4	16	UINT4	Per Channel	FP16	4+4/16=4.25
MX4	S1M2	2	E1M0	16	E8M0	3+1/2+8/16=4
MX6	S1M4	2	E1M0	16	E8M0	5+1/2+8/16=6
MX9	S1M7	2	E1M0	16	E8M0	8+1/2+8/16=9

With Shared Microexponents, A Little Shifting Goes a Long Way [Bita Rouhani *et al.*]

Post-Training Quantization

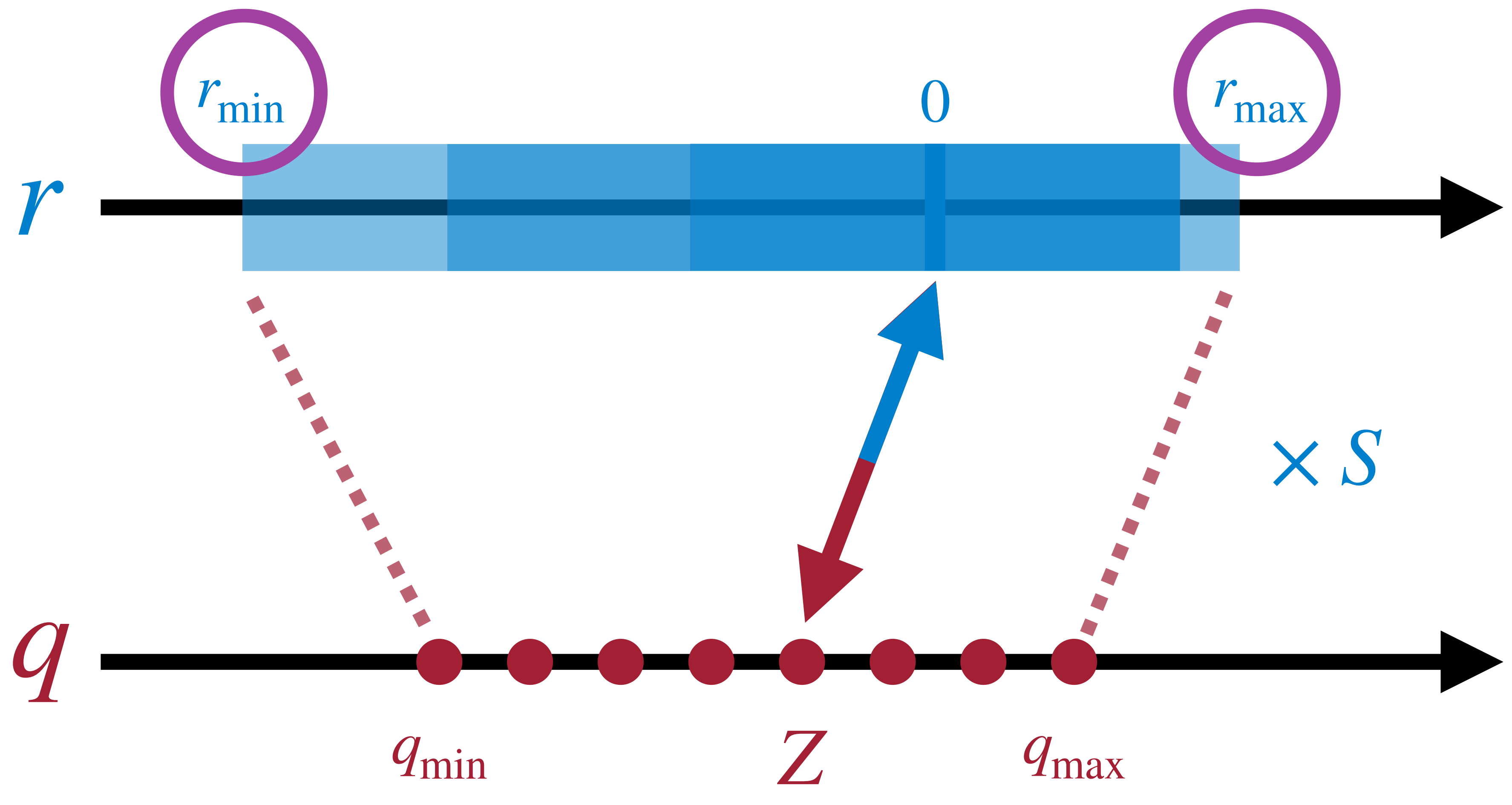
How should we get the optimal linear quantization parameters (S, Z)?

Topic I: Quantization Granularity

Topic II: Dynamic Range Clipping

Topic III: Rounding

Linear Quantization on Activations

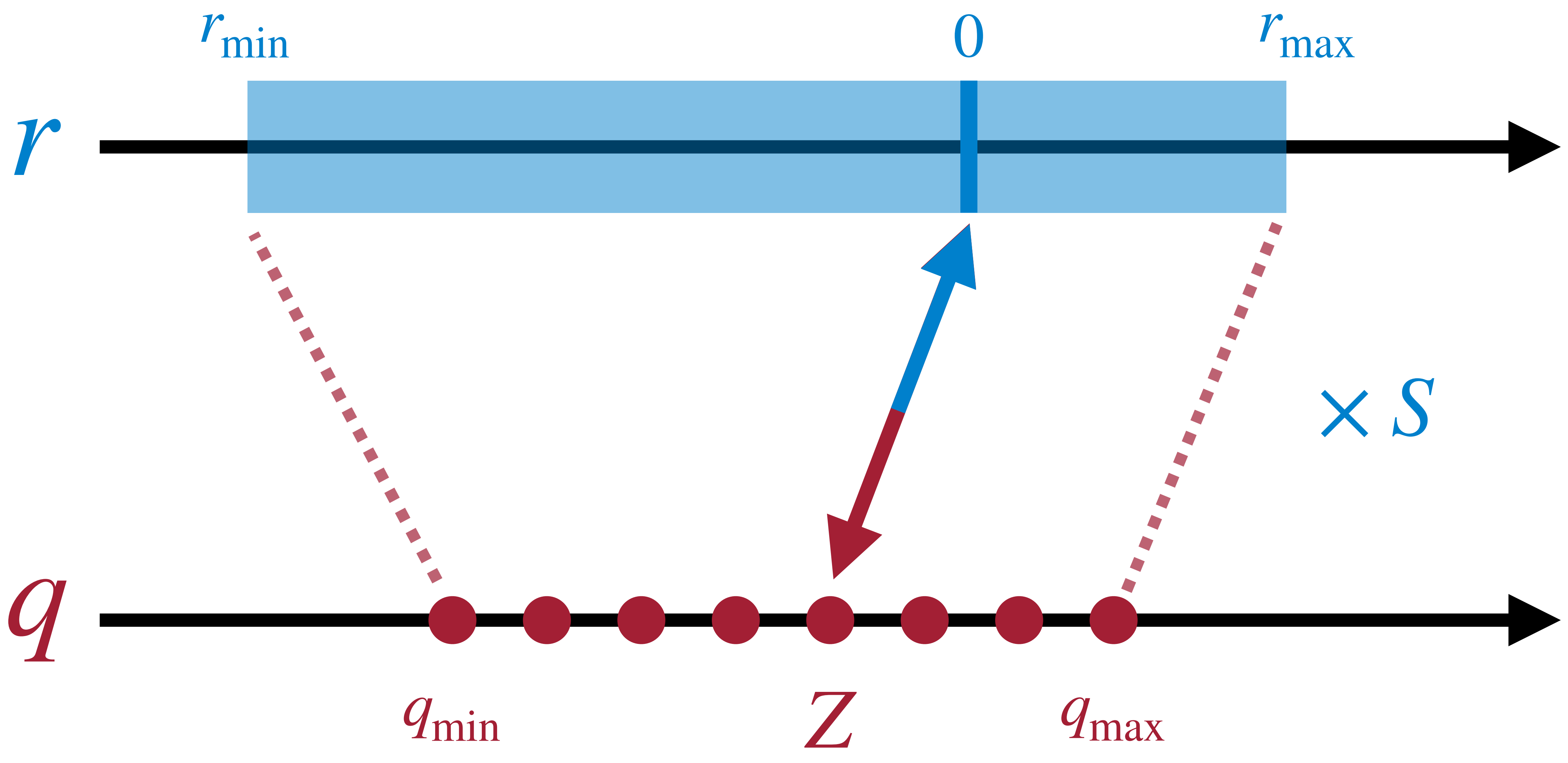


- Unlike weights, the activation range varies across inputs.
- To determine the floating-point range, the activations statistics are gathered ***before*** deploying the model.

Dynamic Range for Activation Quantization

Collect activations statistics before deploying the model

$$\hat{r}^{(t)}_{\max,\min} = \alpha \cdot r^{(t)}_{\max,\min} + (1-\alpha)\cdot \hat{r}^{(t-1)}_{\max,\min}$$

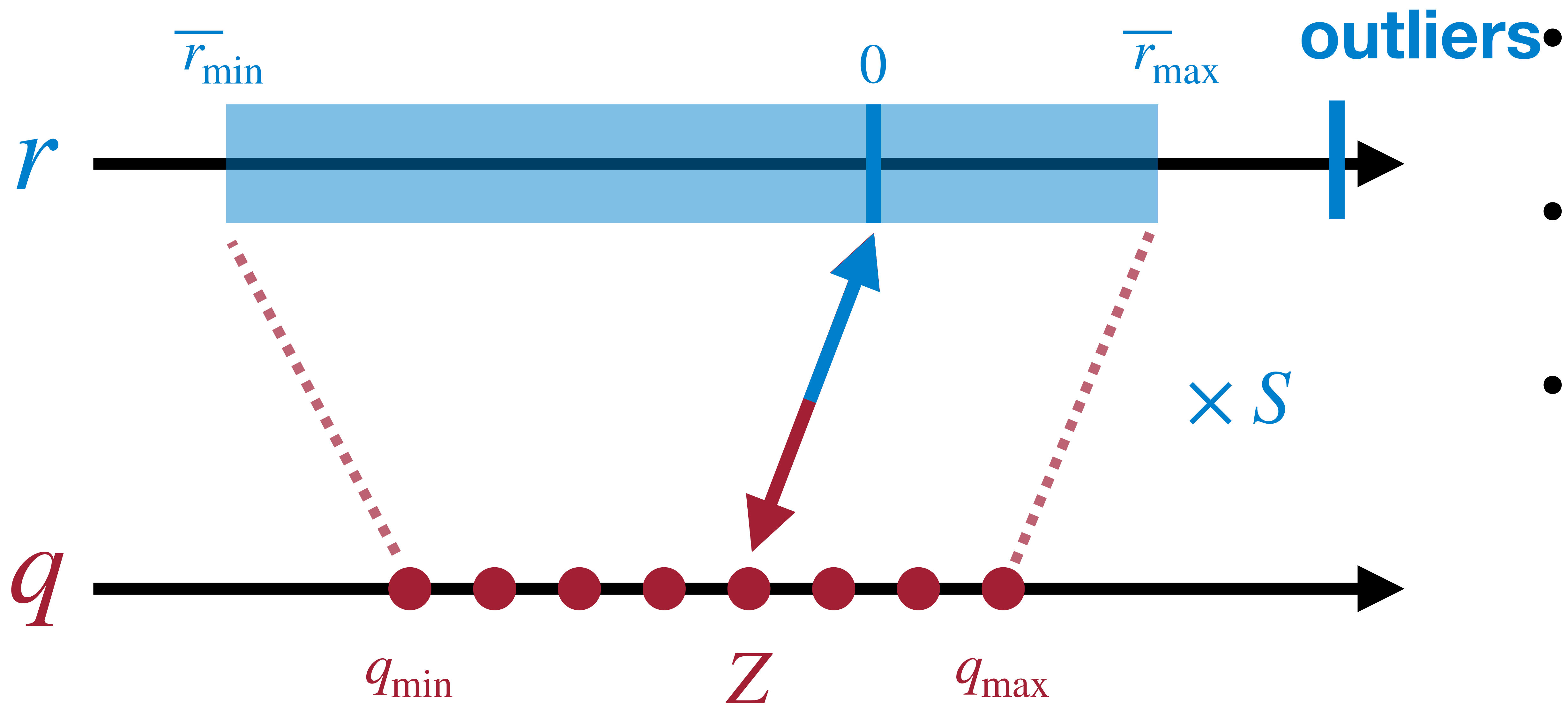


- Type 1: During training
  - Exponential moving averages (EMA)
    - observed ranges are smoothed across thousands of training steps

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

Dynamic Range for Activation Quantization

Collect activations statistics before deploying the model



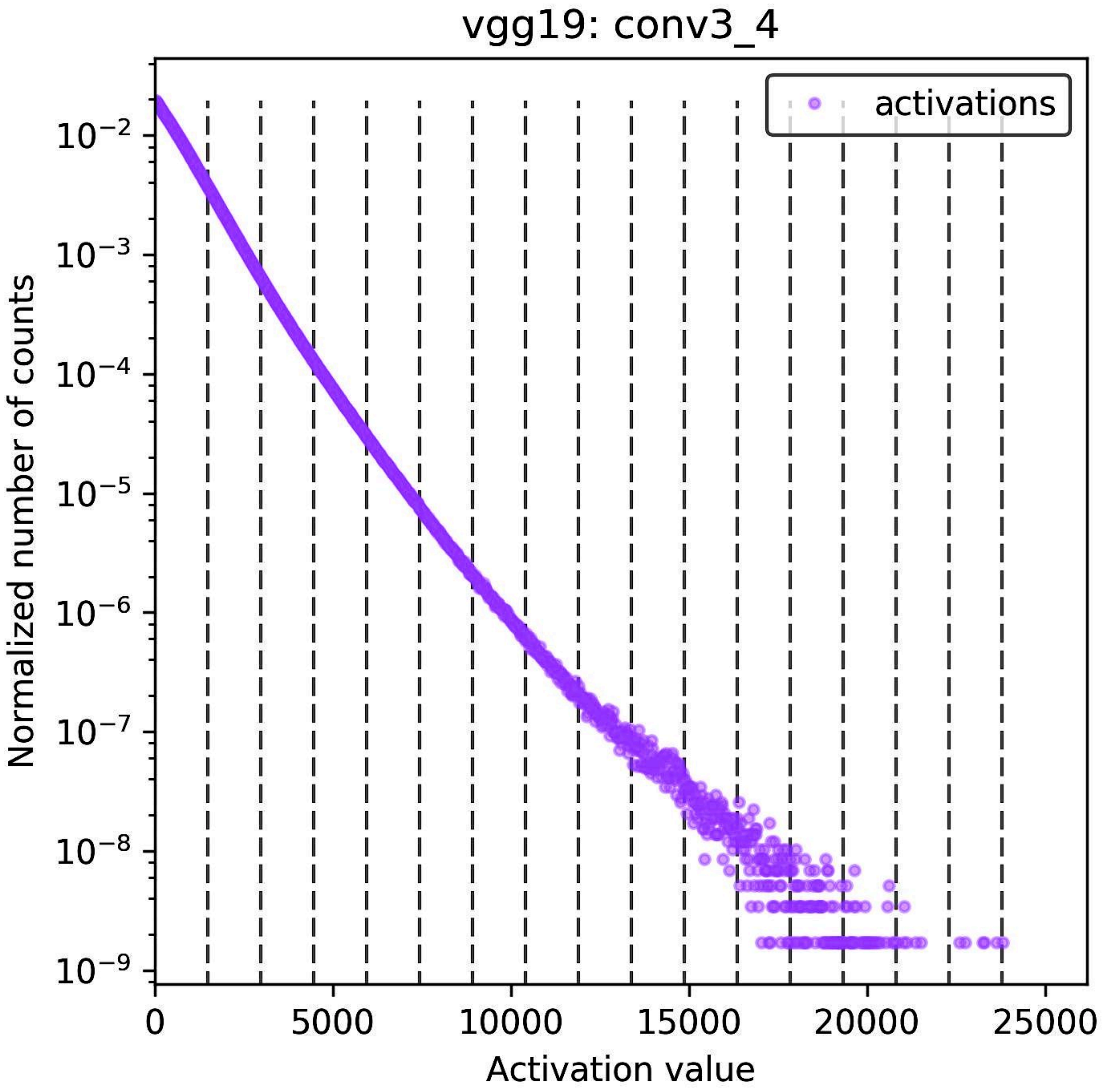
- Type 2: By running a few “calibration” batches of samples on the trained FP32 model
  - spending dynamic range on the outliers hurts the representation ability.
  - use *mean* of the min/max of each sample in the batches
  - analytical calculation (see next slide)

Neural Network Distiller

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

Dynamic Range for Activation Quantization

Collect activations statistics before deploying the model



- Type 2: By running a few "calibration" batches of samples on the trained FP32 model
  - *minimize loss of information*, since integer model encodes the same information as the original floating-point model.
    - loss of information is measured by Kullback-Leibler divergence (relative entropy or information divergence):
      - for two discrete probability distributions P, Q

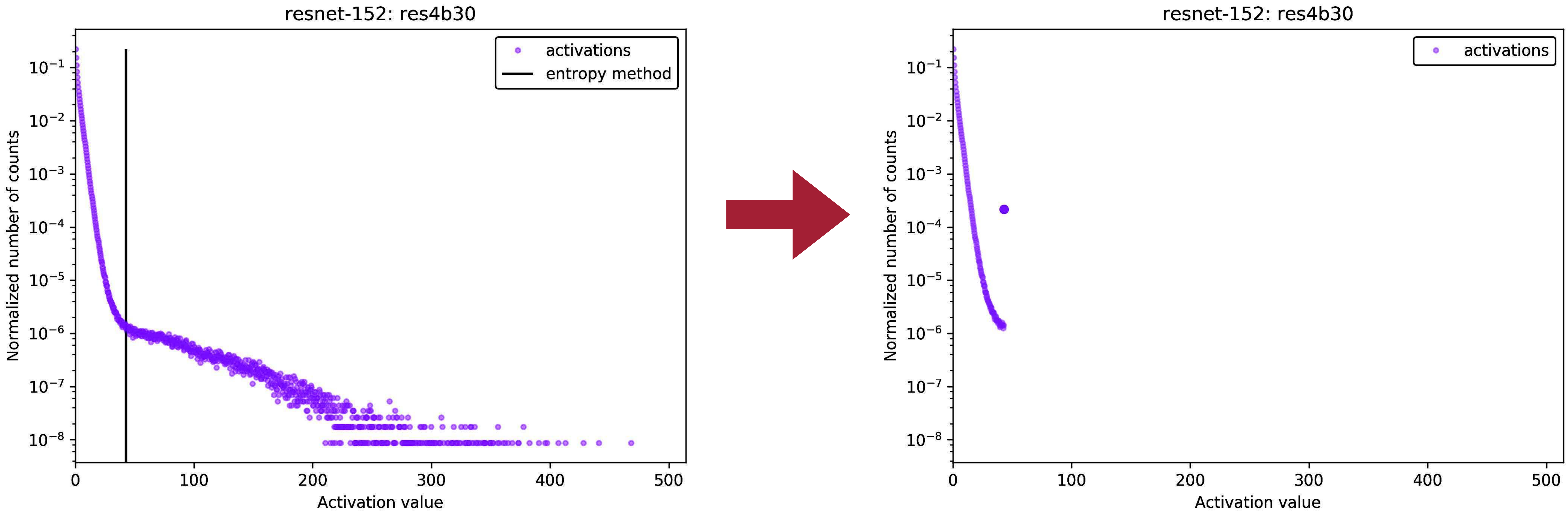
$$D_{KL}(P\|Q) = \sum_{i}^{N} P(x_i)\log\frac{P(x_i)}{Q(x_i)}$$

      - intuition: KL divergence measures the amount of information lost when approximating a given encoding.

8-bit Inference with TensorRT [Szymon Migacz, 2017]

Dynamic Range for Activation Quantization

Minimize loss of information by minimizing the KL divergence

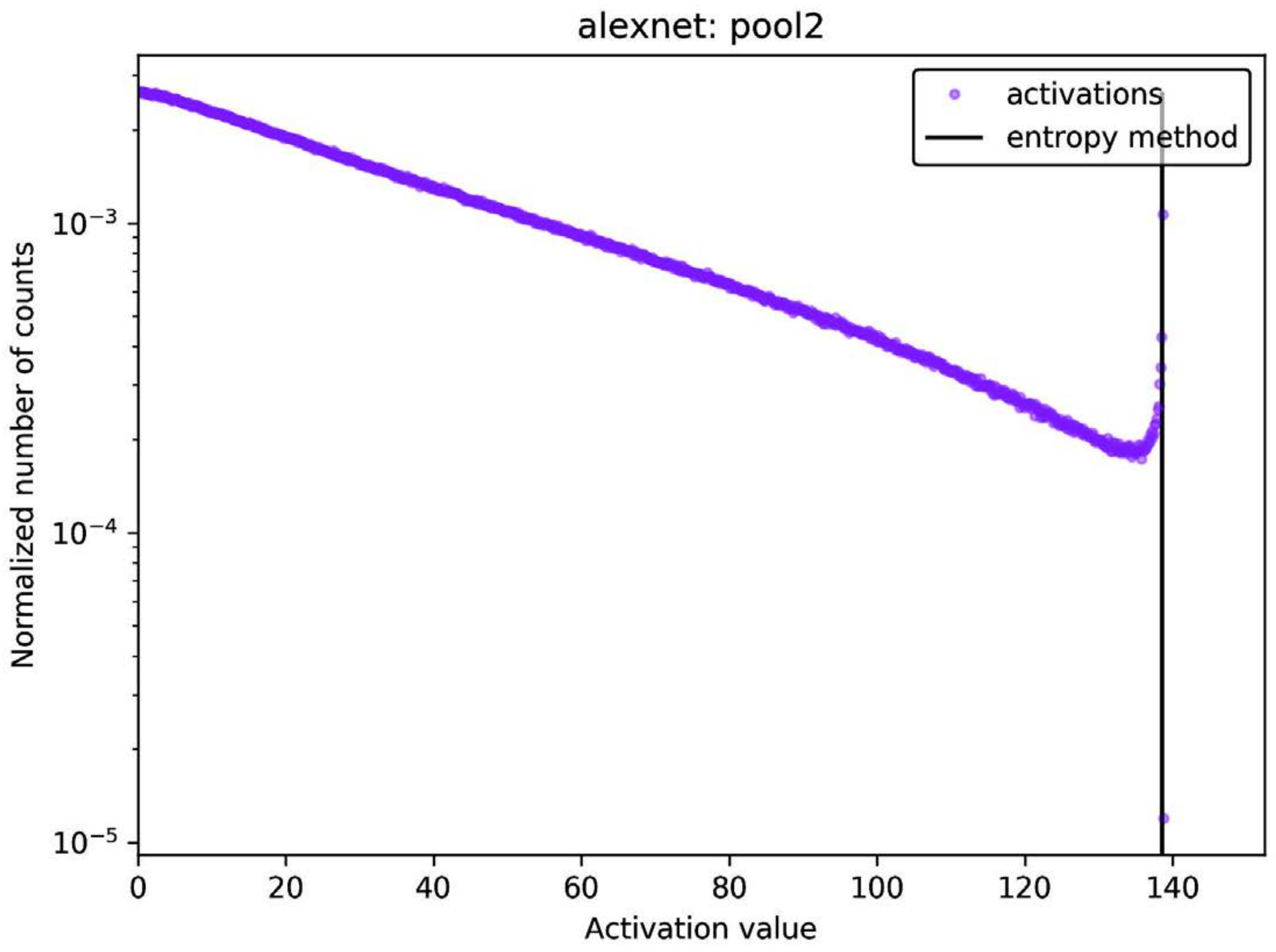


8-bit Inference with TensorRT [Szymon Migacz, 2017]

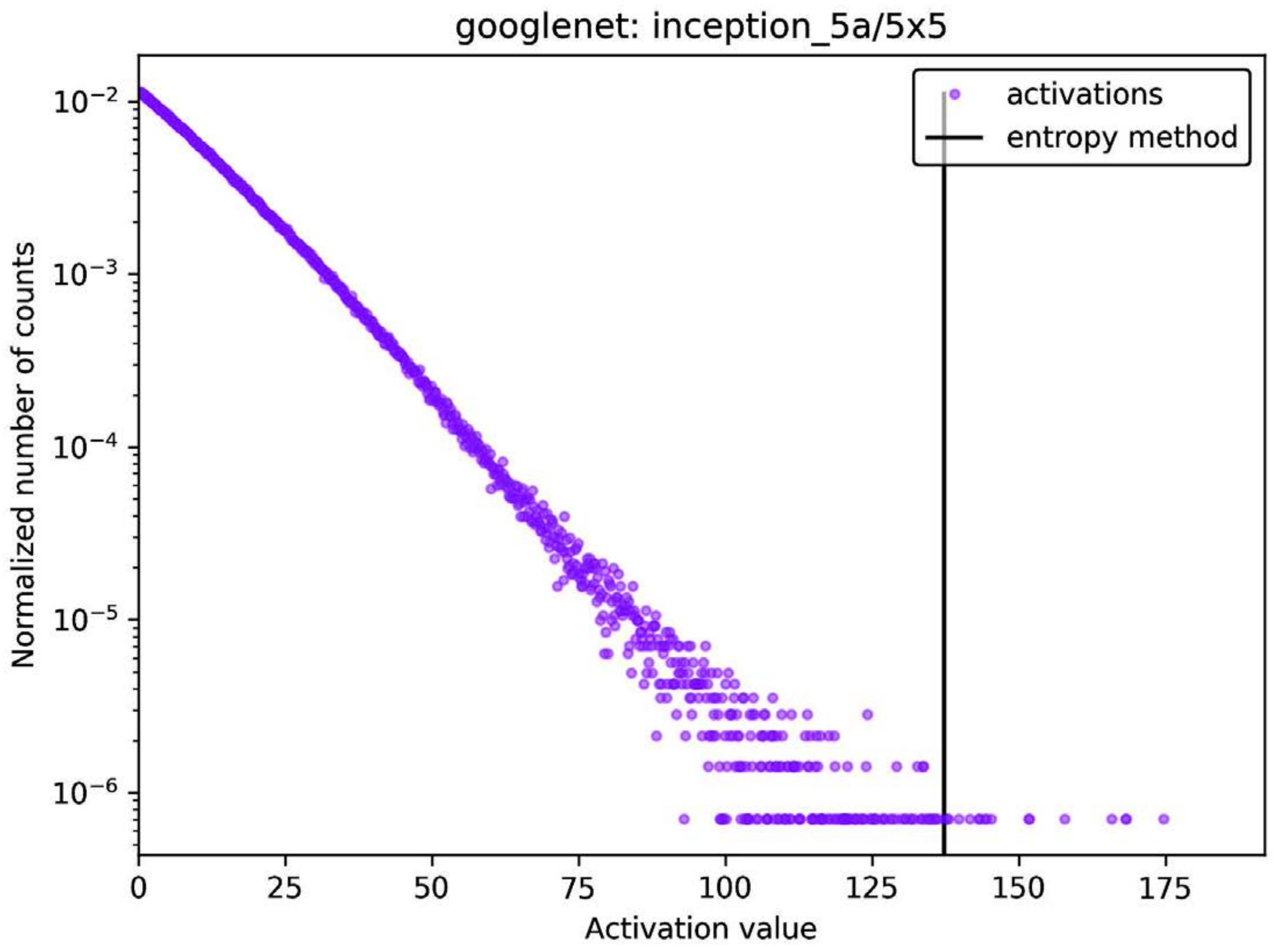
Dynamic Range for Activation Quantization

Minimize loss of information by minimizing the KL divergence

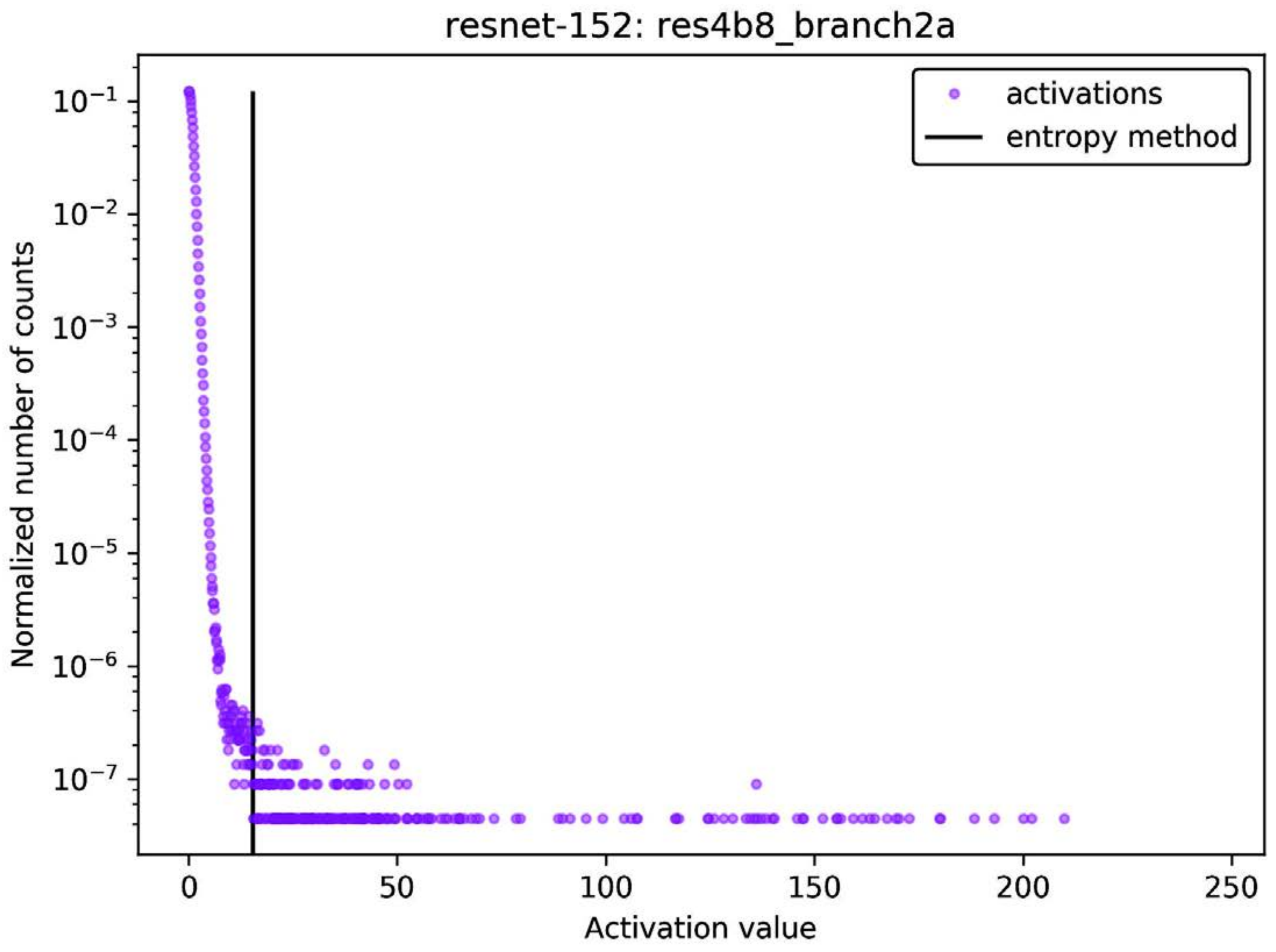
AlexNet: Pool 2



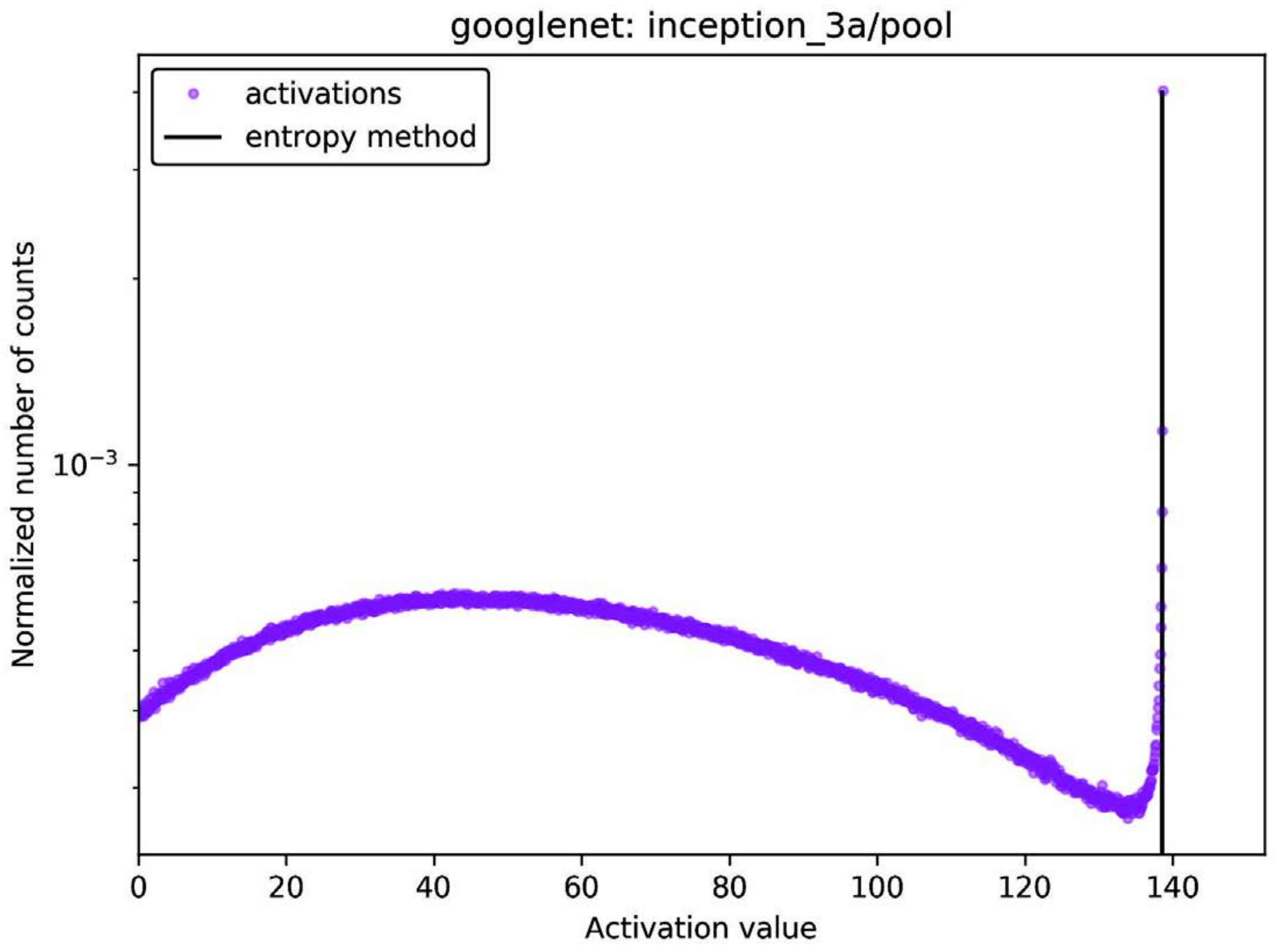
GoogleNet: incpetion_5a/5x5



ResNet-152: res4b8_branch2a



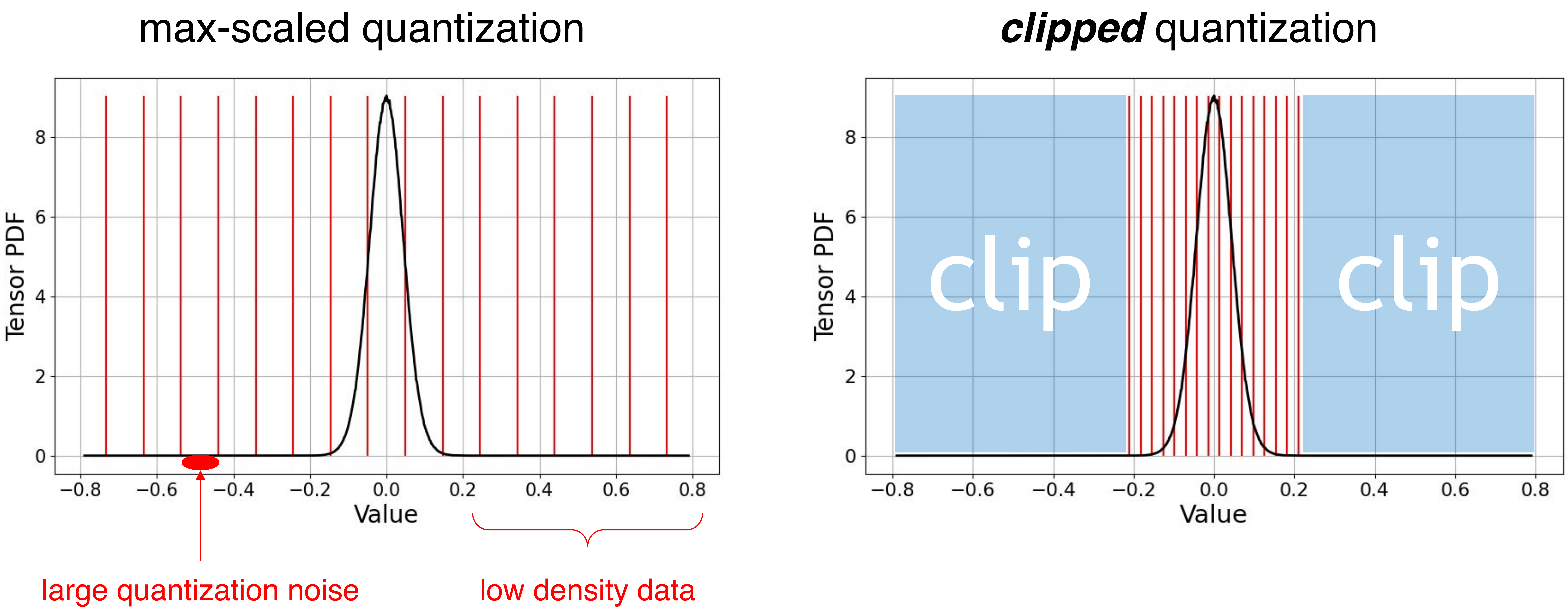
GoogleNet: incpetion_3a/pool



8-bit Inference with TensorRT [Szymon Migacz, 2017]

Dynamic Range for Quantization

Minimize mean-square-error (MSE) using Newton-Raphson method



max-scaled quantization

clipped quantization

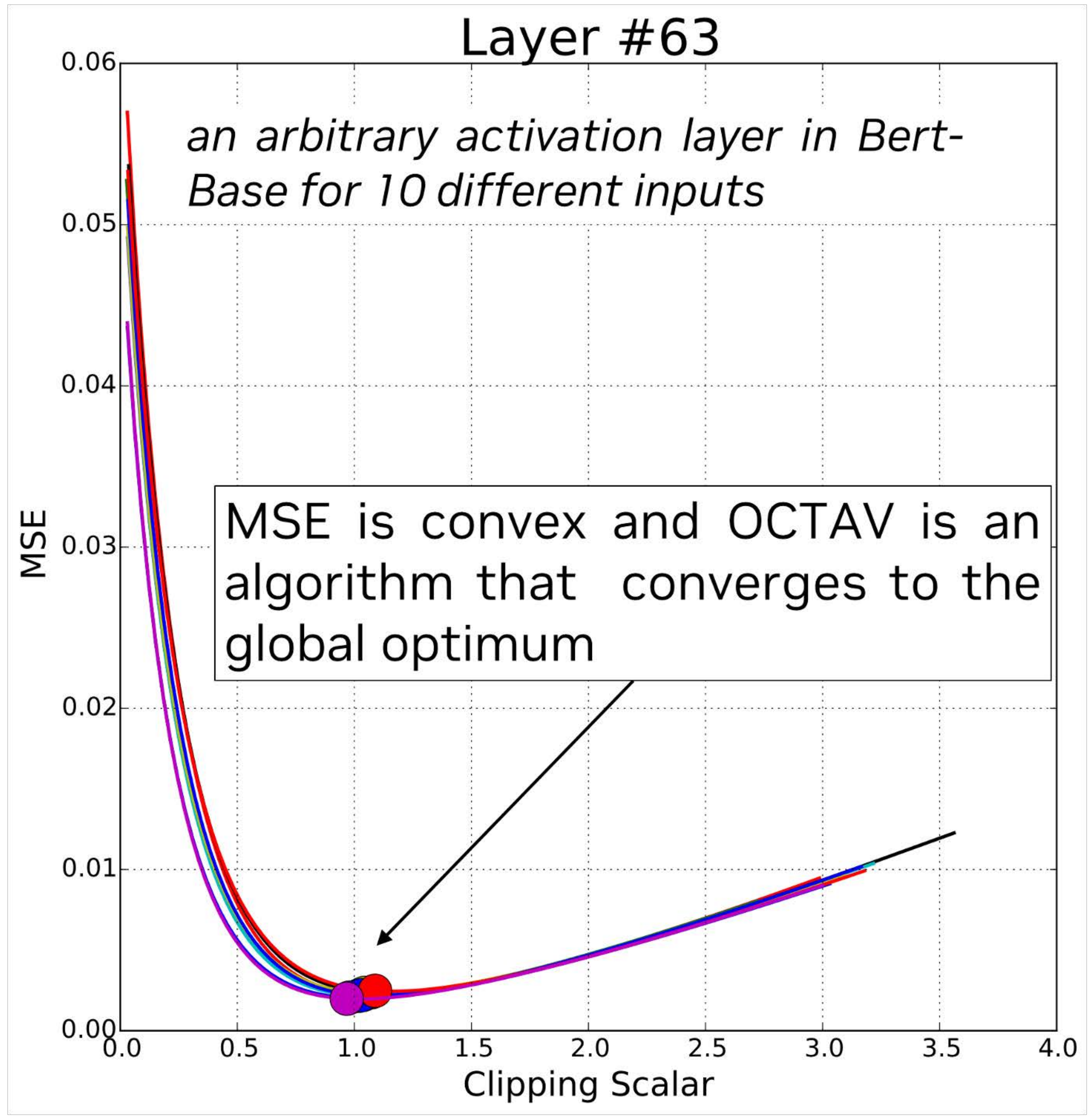
large quantization noise

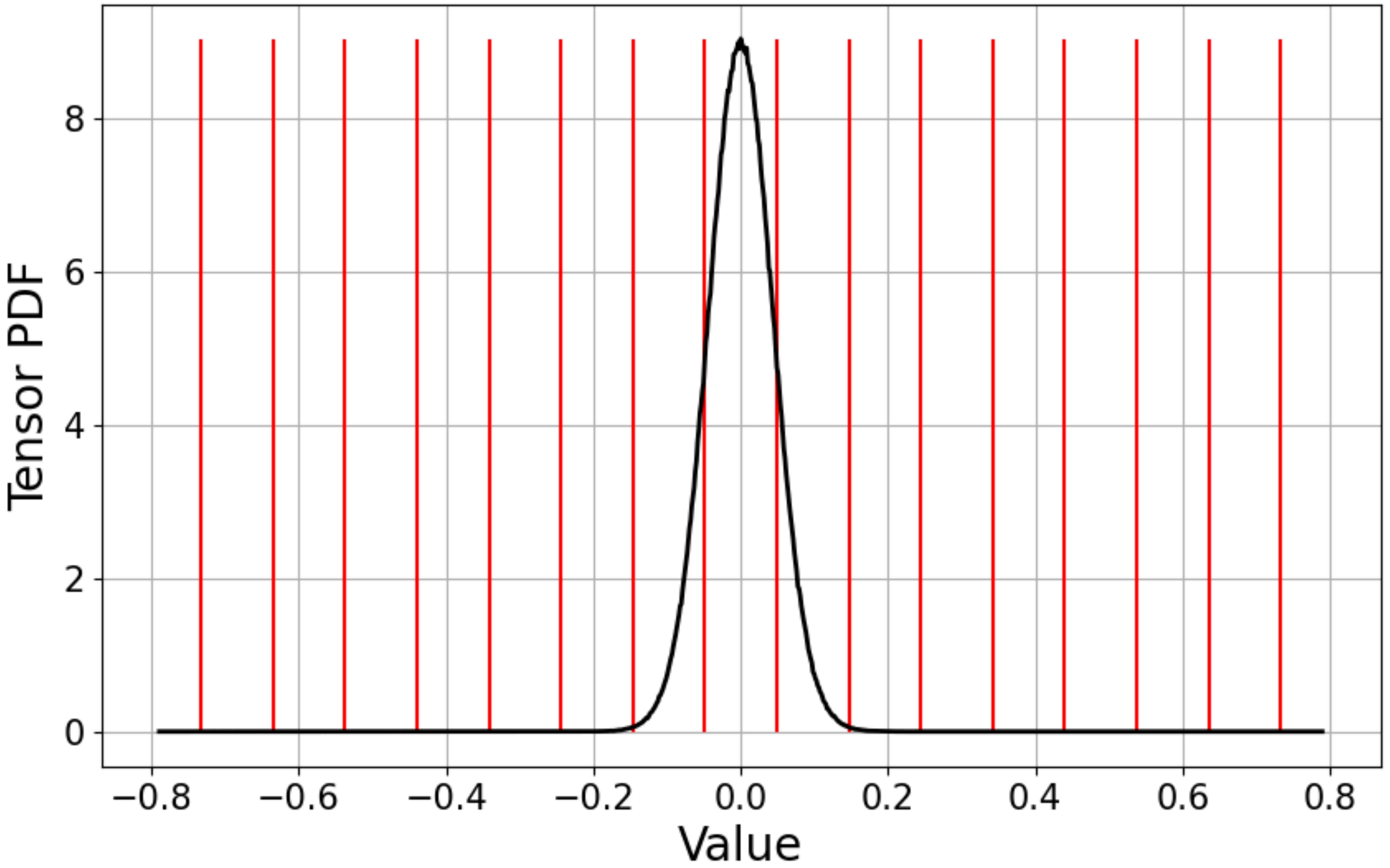
low density data

Optimal Clipping and Magnitude-aware Differentiation for Improved Quantization-aware Training [Sakr *et al.*, ICML 2022]

Dynamic Range for Quantization

Minimize mean-square-error (MSE) using Newton-Raphson method





Network	FP32 Accuracy	OCTAV int4
ResNet-50	76.07	75.84
MobileNet-V2	71.71	70.88
Bert-Large	91.00	87.09

Optimal Clipping and Magnitude-aware Differentiation for Improved Quantization-aware Training [Sakr *et al.*, ICML 2022]

Post-Training Quantization

How should we get the optimal linear quantization parameters (S, Z)?

Topic I: Quantization Granularity

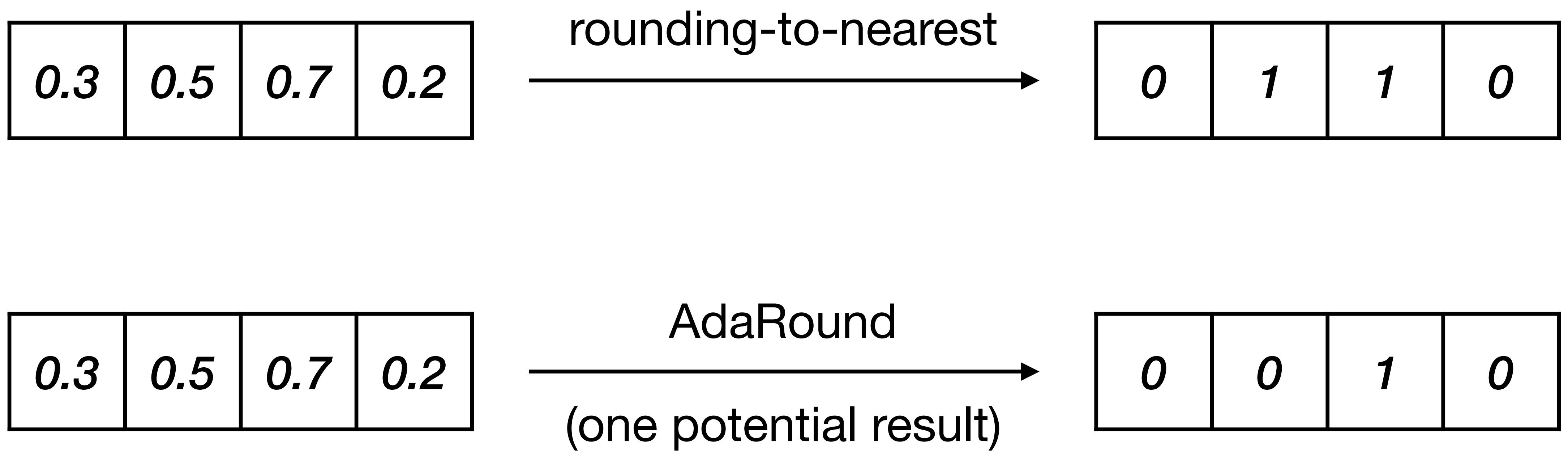
Topic II: Dynamic Range Clipping

Topic III: Rounding

Adaptive Rounding for Weight Quantization

Rounding-to-nearest is not optimal

- **Philosophy**
  - Rounding-to-nearest is not optimal
  - Weights are correlated with each other. The best rounding for each weight (to nearest) is not the best rounding for the whole tensor



$$[0.3 \; 0.5 \; 0.7 \; 0.2] \xrightarrow{\text{rounding-to-nearest}} [0 \; 1 \; 1 \; 0]$$

$$[0.3 \; 0.5 \; 0.7 \; 0.2] \xrightarrow[\text{(one potential result)}]{\text{AdaRound}} [0 \; 0 \; 1 \; 0]$$

- What is optimal? Rounding that reconstructs the original <u>activation</u> the best, which may be very different
  - For weight quantization only
  - With short-term tuning, (almost) post-training quantization

Up or Down? Adaptive Rounding for Post-Training Quantization [Nagel *et al.*, PMLR 2020]

Adaptive Rounding for Weight Quantization

Rounding-to-nearest is not optimal

- **Method**:
  - Instead of $\lfloor w \rceil$, we want to choose from $\{\lfloor w \rfloor, \lceil w \rceil\}$ to get the best reconstruction
  - We took a learning-based method to find quantized value $\tilde{w} = \lfloor \lfloor w \rfloor + \delta \rceil, \delta \in [0,1]$

Up or Down? Adaptive Rounding for Post-Training Quantization [Nagel *et al.*, PMLR 2020]

Adaptive Rounding for Weight Quantization

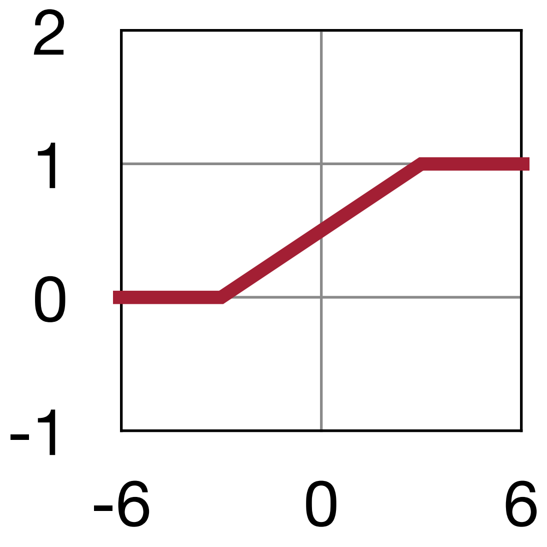
Rounding-to-nearest is not optimal

- **Method**:
  - Instead of $\lfloor w \rceil$, we want to choose from $\{\lfloor w \rfloor, \lceil w \rceil\}$ to get the best reconstruction
  - We took a learning-based method to find quantized value $\tilde{w} = \lfloor \lfloor w \rfloor + \delta \rceil, \delta \in [0,1]$
  - We optimize the following equation (omit the derivation):

$$\operatorname{argmin}_{\mathbf{V}} \|\mathbf{W}\mathbf{x} - \boxed{\tilde{\mathbf{W}}}\mathbf{x}\|_F^2 + \lambda f_{reg}(\mathbf{V})$$

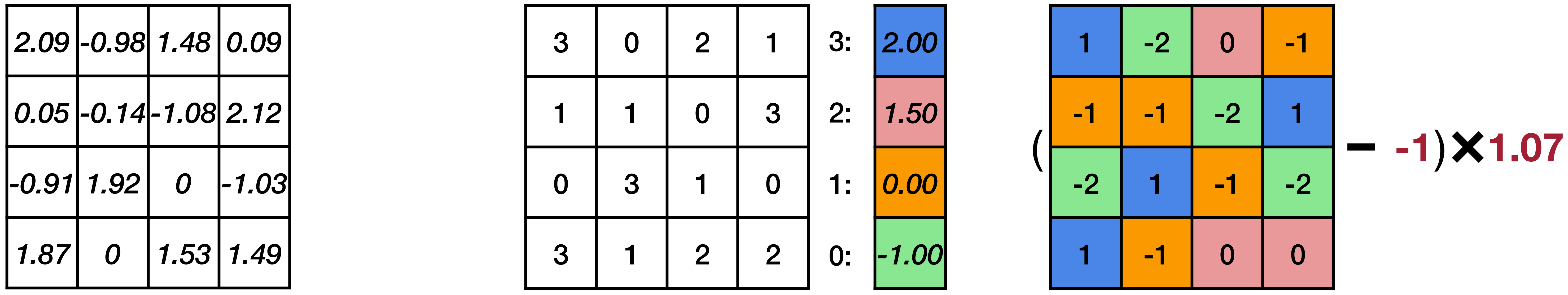
$$\rightarrow \operatorname{argmin}_{\mathbf{V}} \|\mathbf{W}\mathbf{x} - \boxed{\lfloor \lfloor \mathbf{W} \rfloor + \mathbf{h}(\mathbf{V}) \rceil}\mathbf{x}\|_F^2 + \lambda f_{reg}(\mathbf{V})$$

  - $\mathbf{x}$ is the input to the layer, $\mathbf{V}$ is a random variable of the same shape
  - $\mathbf{h}()$ is a function to map the range to $(0,1)$, such as rectified sigmoid
  - $f_{reg}(\mathbf{V})$ is a regularization that encourages $\mathbf{h}(\mathbf{V})$ to be binary
    - $f_{reg}(\mathbf{V}) = \sum_{i,j} 1 - |2h(\mathbf{V}_{i,j}) - 1|^{\beta}$



Up or Down? Adaptive Rounding for Post-Training Quantization [Nagel *et al.*, PMLR 2020]

Neural Network Quantization



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

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

(

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

− -1)×1.07

		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

- **Zero Point**
  - Asymmetric
  - Symmetric
- **Scaling Granularity**
  - Per-Tensor
  - Per-Channel
  - Group Quantization
- **Range Clipping**
  - Exponential Moving Average
  - Minimizing KL Divergence
  - Minimizing Mean-Square-Error
- **Rounding**
  - Round-to-Nearest
  - AdaRound

Post-Training INT8 Linear Quantization

Activation		Symmetric	Asymmertric
		Per-Tensor	Per-Tensor
		Minimize KL-Divergence	Exponential Moving Average (EMA)
Weight		Symmetric	Symmetric
		Per-Tensor	Per-Channel
Neural Network	GoogleNet	-0.45%	0%
	ResNet-50	-0.13%	-0.6%
	ResNet-152	-0.08%	-1.8%
	MobileNetV1	-	-11.8%
	MobileNetV2	-	-2.1%

Data-Free Quantization Through Weight Equalization and Bias Correction [Markus *et al.*, ICCV 2019]
Quantizing Deep Convolutional Networks for Efficient Inference: A Whitepaper [Raghuraman Krishnamoorthi, arXiv 2018]
8-bit Inference with TensorRT [Szymon Migacz, 2017]

Post-Training INT8 Linear Quantization

Activation		Symmetric	Asymmertric
		Per-Tensor	Per-Tensor
		Minimize KL-Divergence	Exponential Moving Average (EMA)
Weight		Symmetric	Symmetric
		Per-Tensor	Per-Channel
Neural Network			
	MobileNetV1	-	-11.8%
	MobileNetV2	-	-2.1%

Smaller models seem to not respond as well to post-training quantization, presumabley due to their smaller representational capacity.

How should we improve performance of quantized models?

Data-Free Quantization Through Weight Equalization and Bias Correction [Markus *et al.*, ICCV 2019]
Quantizing Deep Convolutional Networks for Efficient Inference: A Whitepaper [Raghuraman Krishnamoorthi, arXiv 2018]
8-bit Inference with TensorRT [Szymon Migacz, 2017]

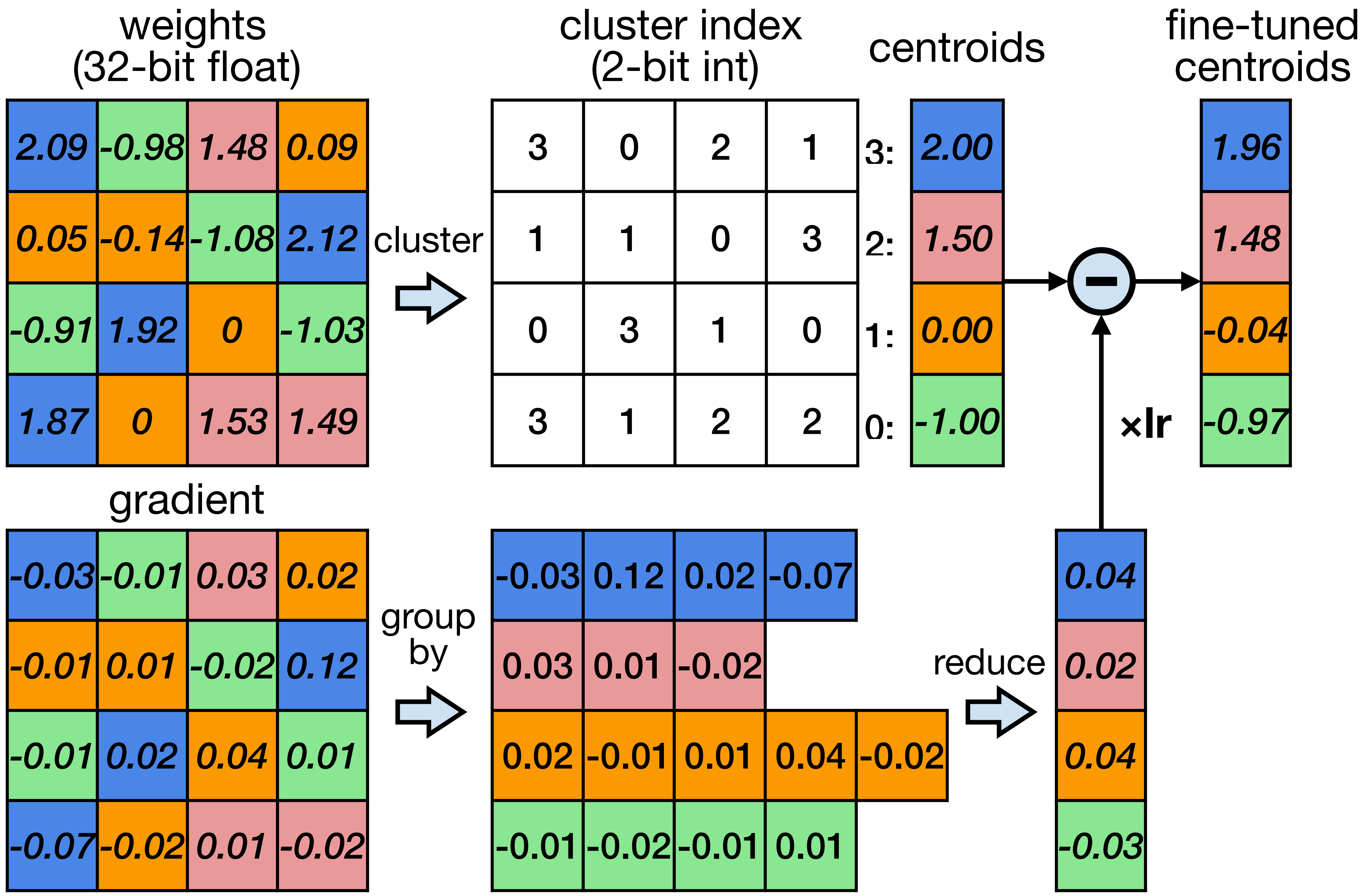
Quantization-Aware Training

How should we improve performance of quantized models?

Quantization-Aware Training

Train the model taking quantization into consideration

- To minimize the loss of accuracy, especially aggressive quantization with 4 bits and lower bit width, neural network will be trained/fine-tuned with quantized weights and activations.
- Usually, fine-tuning a pre-trained floating point model provides better accuracy than training from scratch.



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

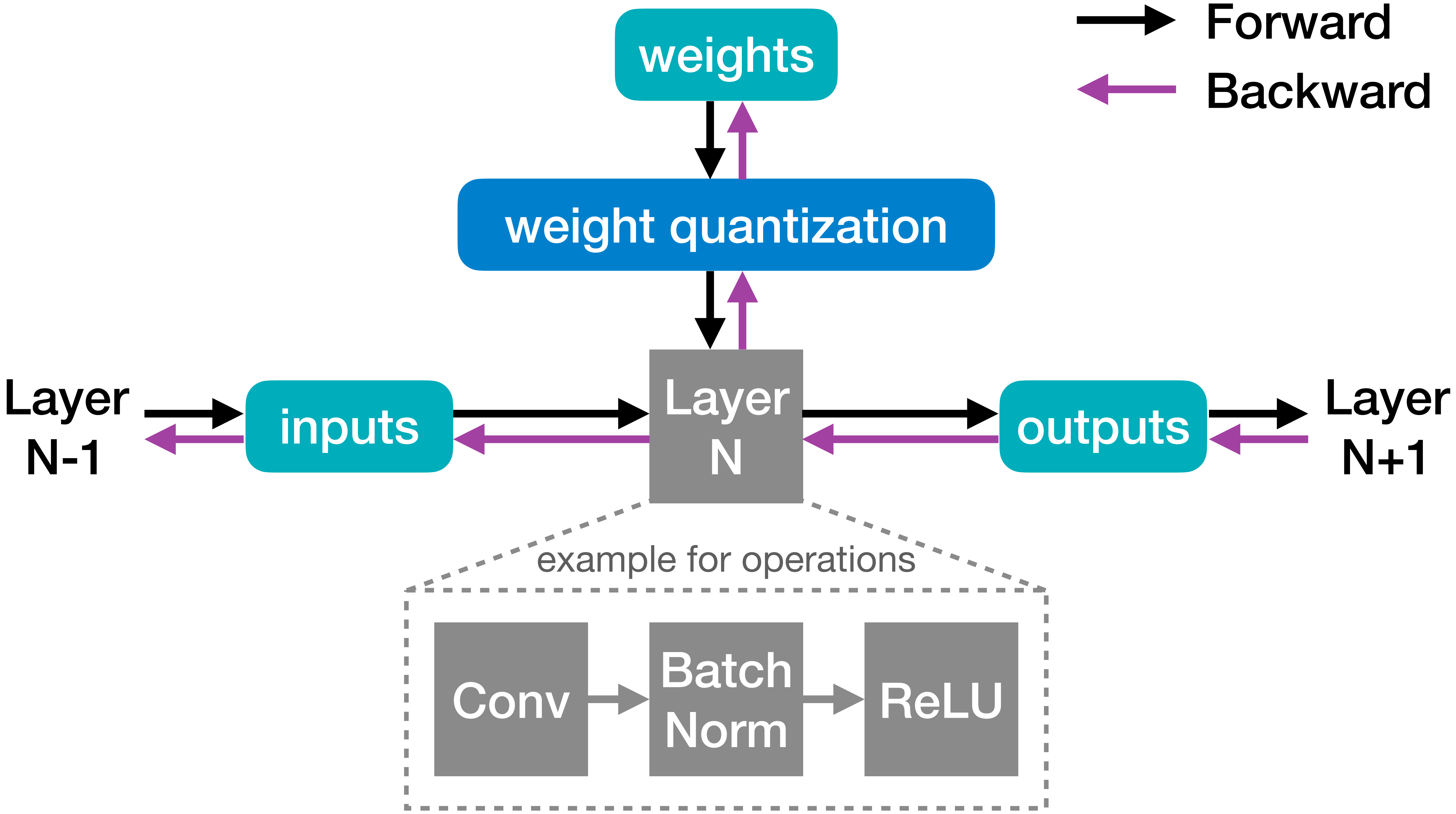
	centroids	fine-tuned centroids
3:	2.00	1.96
2:	1.50	1.48
1:	0.00	-0.04
0:	-1.00	-0.97

gradient

-0.03	-0.01	0.03	0.02
-0.01	0.01	-0.02	0.12
-0.01	0.02	0.04	0.01
-0.07	-0.02	0.01	-0.02

group by

					reduce
-0.03	0.12	0.02	-0.07		0.04
0.03	0.01	-0.02			0.02
0.02	-0.01	0.01	0.04	-0.02	0.04
-0.01	-0.02	-0.01	0.01		-0.03

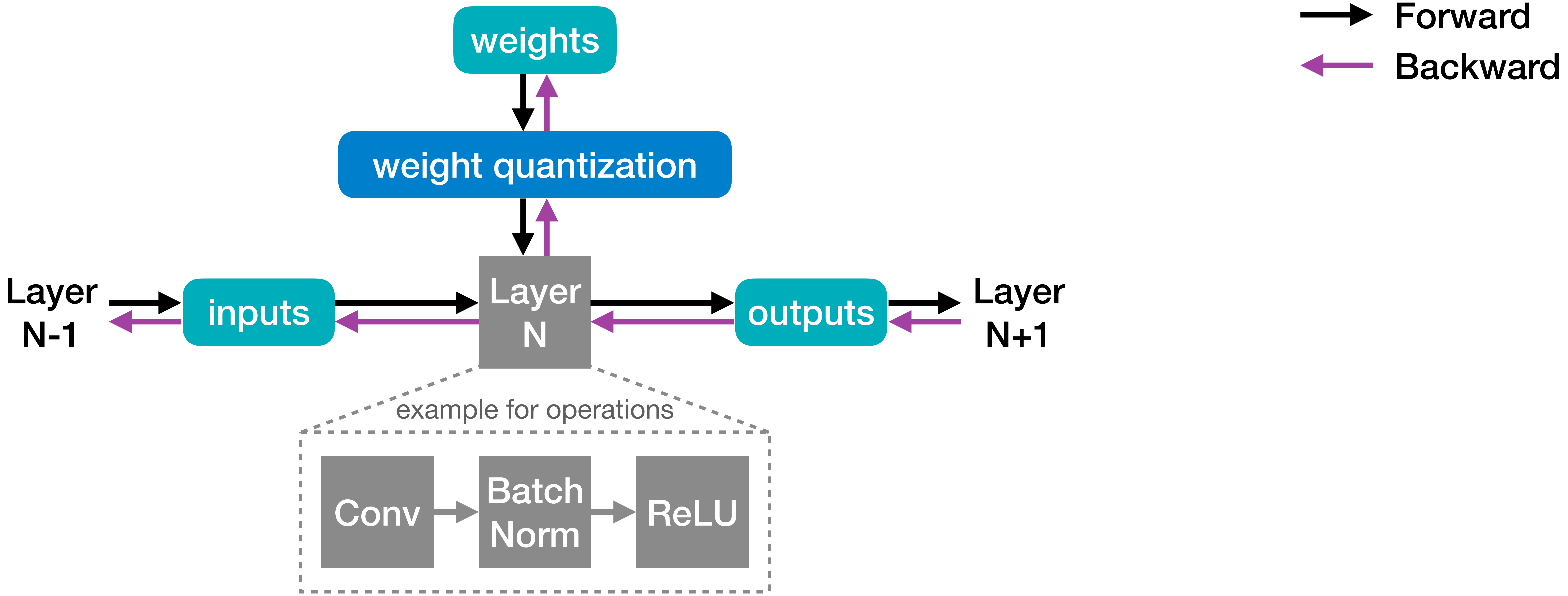


Deep Compression [Han *et al.*, ICLR 2016]

Quantization-Aware Training

Train the model taking quantization into consideration

- A full precision copy of the weights W is maintained throughout the training.
- The small gradients are accumulated without loss of precision.
- Once the model is trained, only the quantized weights are used for inference.



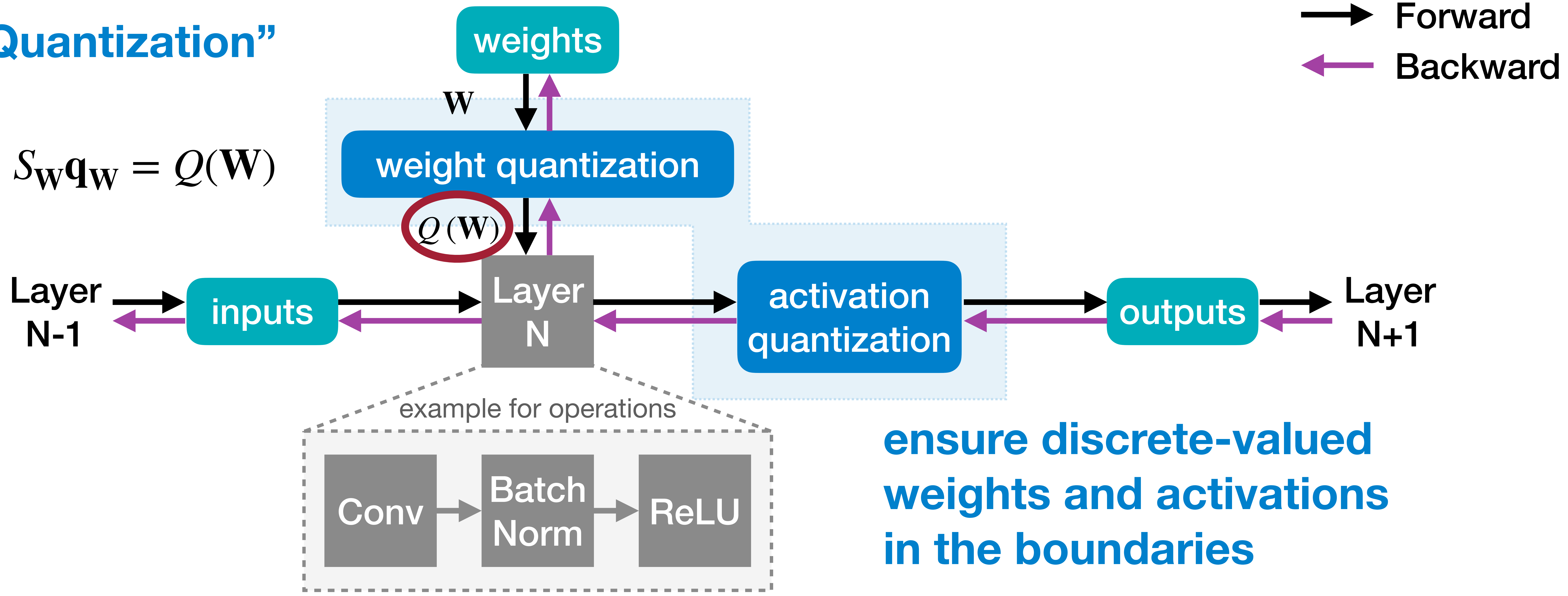
Quantization-Aware Training

Train the model taking quantization into consideration

- A full precision copy of the weights W is maintained throughout the training.
- The small gradients are accumulated without loss of precision.
- Once the model is trained, only the quantized weights are used for inference.

“Simulated/Fake Quantization”

$$\mathbf{W} \rightarrow S_{\mathbf{W}}\mathbf{q}_{\mathbf{W}} = Q(\mathbf{W})$$



ensure discrete-valued weights and activations in the boundaries

these operations still run in full precision

Linear Quantization

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

weights (32-bit float)

quantized weights (2-bit signed int)

zero point (2-bit signed int)

scale (32-bit float)

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

Quantization-Aware Training

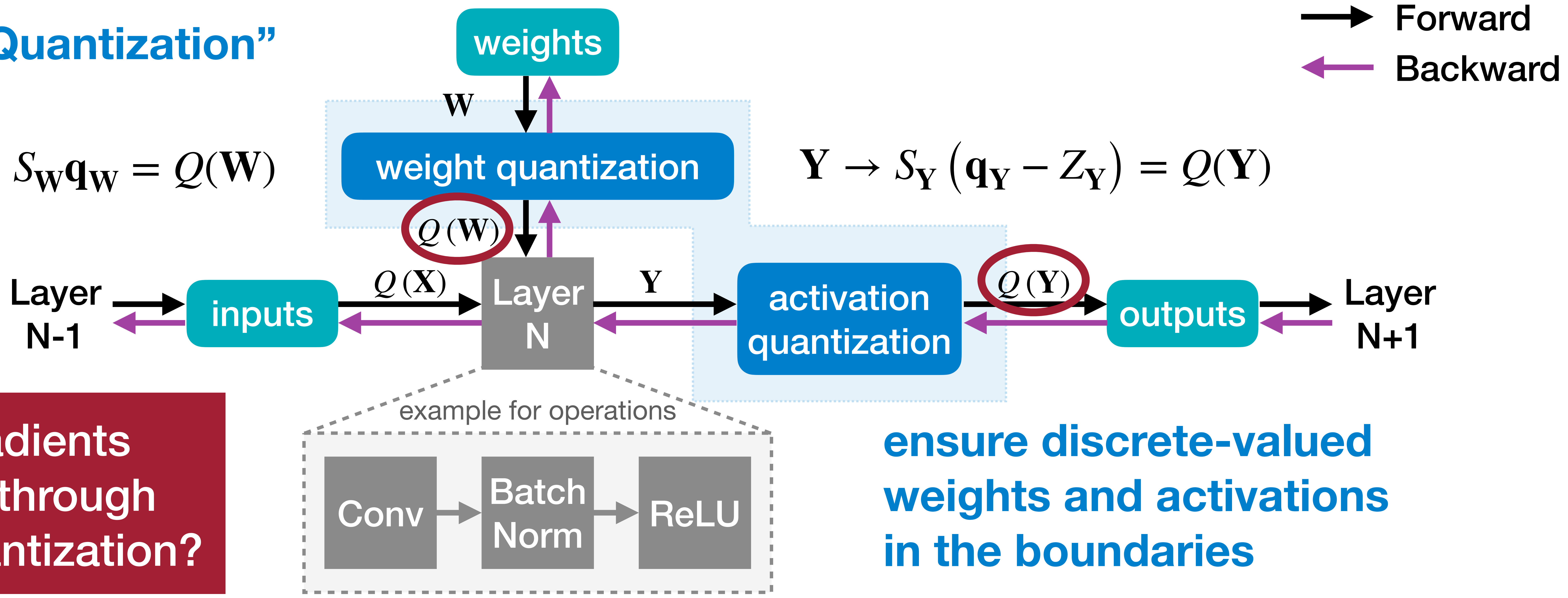
Train the model taking quantization into consideration

- A full precision copy of the weights W is maintained throughout the training.
- The small gradients are accumulated without loss of precision.
- Once the model is trained, only the quantized weights are used for inference.

"Simulated/Fake Quantization"

$$\mathbf{W} \rightarrow S_{\mathbf{W}}\mathbf{q}_{\mathbf{W}} = Q(\mathbf{W})$$

$$\mathbf{Y} \rightarrow S_{\mathbf{Y}}\left(\mathbf{q}_{\mathbf{Y}} - Z_{\mathbf{Y}}\right) = Q(\mathbf{Y})$$



ensure discrete-valued weights and activations in the boundaries

How should gradients back-propagate through the (simulated) quantization?

these operations still run in full precision

Straight-Through Estimator (STE)

- Quantization is discrete-valued, and thus the derivative is 0 almost everywhere.

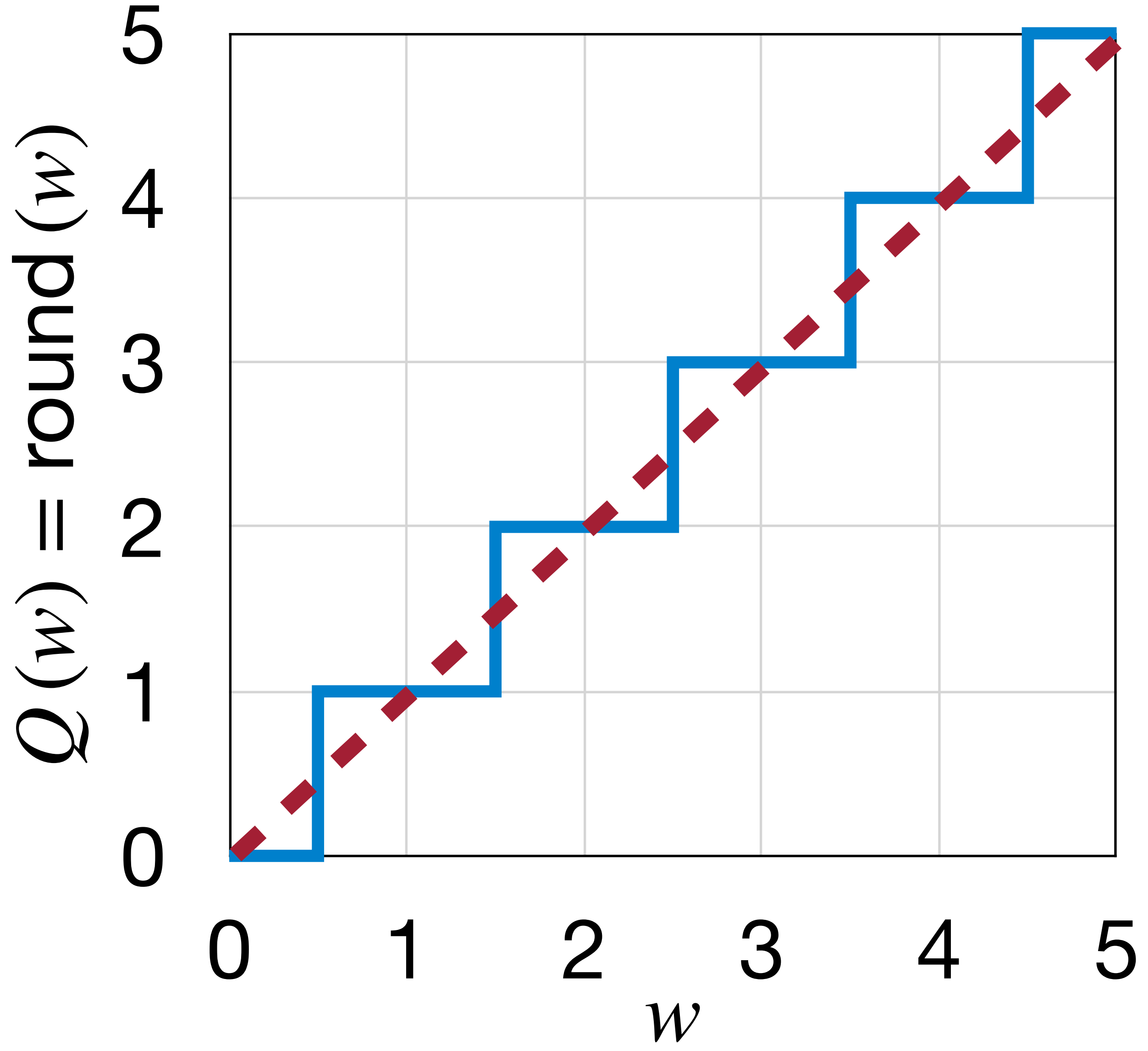
$$\frac{\partial Q(W)}{\partial W} = 0$$

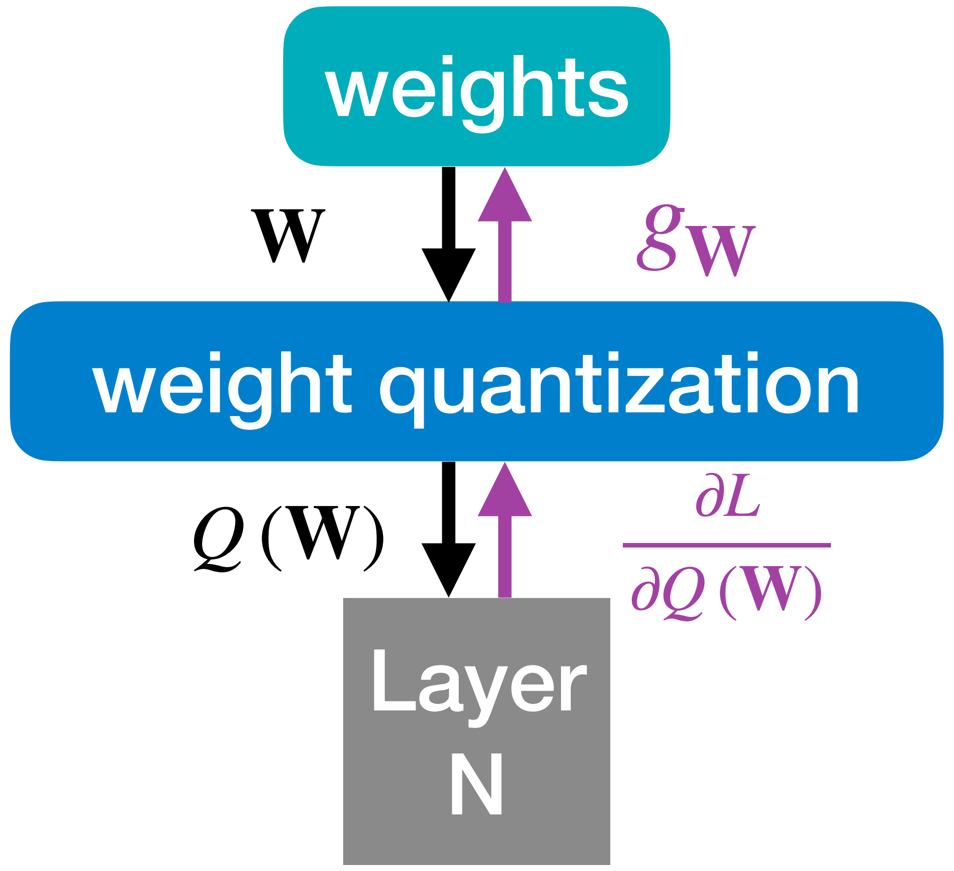
  - The neural network will learn nothing since gradients become 0 and the weights won't get updated.

$$g_{\mathbf{W}} = \frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial Q(\mathbf{W})} \cdot \frac{\partial Q(\mathbf{W})}{\partial \mathbf{W}} = 0$$

- Straight-Through Estimator (STE) simply passes the gradients through the quantization as if it had been the *identity* function.

$$g_{\mathbf{W}} = \frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial Q(\mathbf{W})}$$





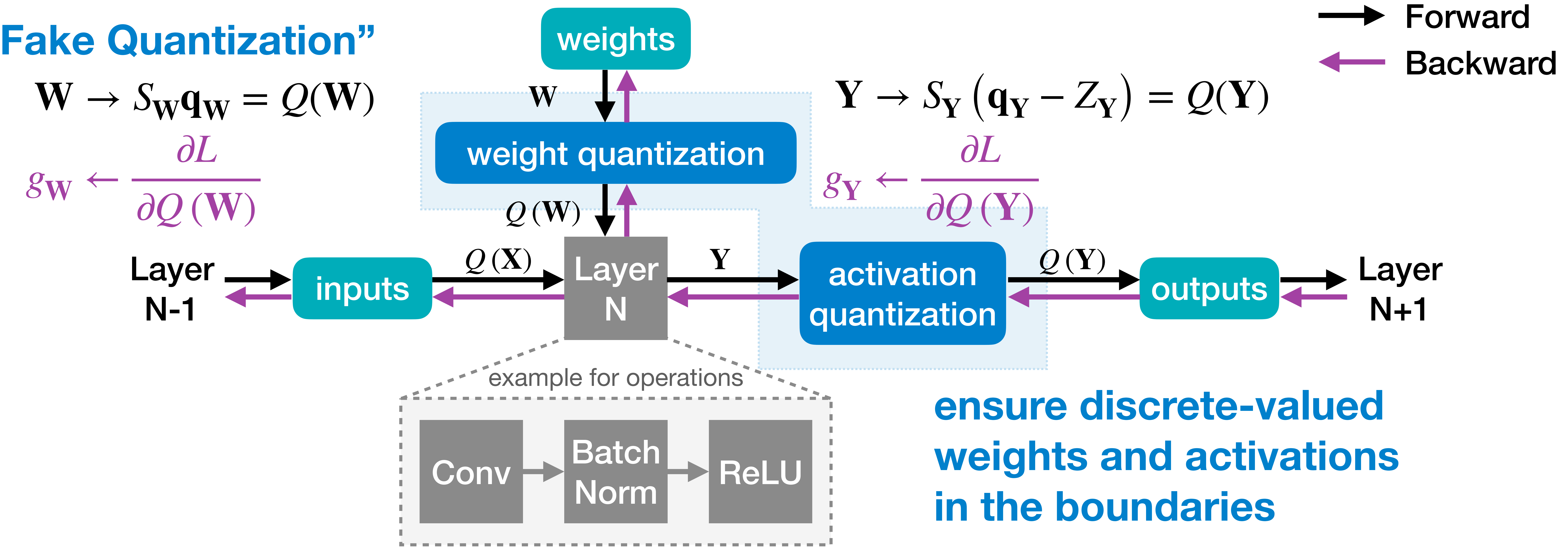
Neural Networks for Machine Learning [Hinton *et al.*, Coursera Video Lecture, 2012]
Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation [Bengio, arXiv 2013]

Quantization-Aware Training

Train the model taking quantization into consideration

- A full precision copy of the weights is maintained throughout the training.
- The small gradients are accumulated without loss of precision.
- Once the model is trained, only the quantized weights are used for inference.

"Simulated/Fake Quantization"



$$\mathbf{W} \rightarrow S_{\mathbf{W}}\mathbf{q}_{\mathbf{W}} = Q(\mathbf{W})$$

$$g_{\mathbf{W}} \leftarrow \frac{\partial L}{\partial Q(\mathbf{W})}$$

$$\mathbf{Y} \rightarrow S_{\mathbf{Y}}\left(\mathbf{q}_{\mathbf{Y}} - Z_{\mathbf{Y}}\right) = Q(\mathbf{Y})$$

$$g_{\mathbf{Y}} \leftarrow \frac{\partial L}{\partial Q(\mathbf{Y})}$$

ensure discrete-valued weights and activations in the boundaries

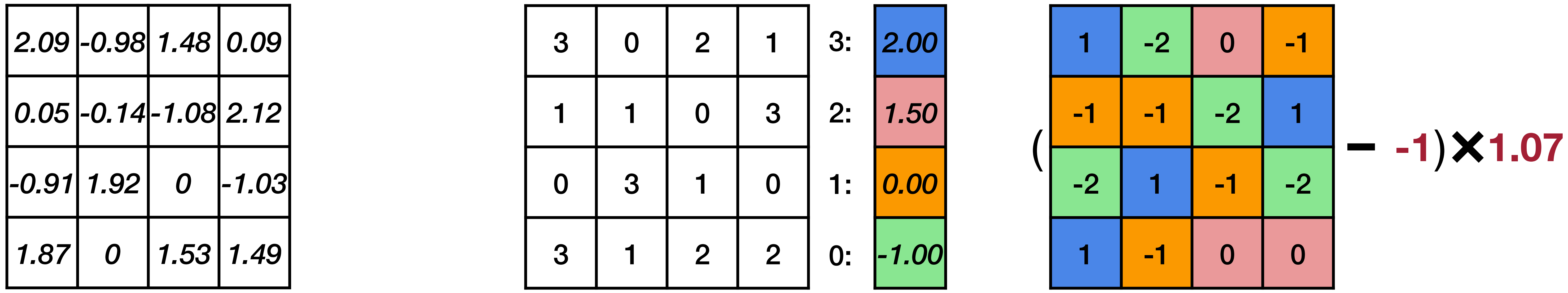
these operations still run in full precision

INT8 Linear Quantization-Aware Training

Neural Network	Floating-Point	Post-Training Quantization		Quantization-Aware Training	
		Asymmetric	Symmetric	Asymmetric	Symmetric
		Per-Tensor	Per-Channel	Per-Tensor	Per-Channel
MobileNetV1	70.9%	0.1%	59.1%	70.0%	70.7%
MobileNetV2	71.9%	0.1%	69.8%	70.9%	71.1%
NASNet-Mobile	74.9%	72.2%	72.1%	73.0%	73.0%

Quantizing Deep Convolutional Networks for Efficient Inference: A Whitepaper [Raghuraman Krishnamoorthi, arXiv 2018]

Neural Network Quantization



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

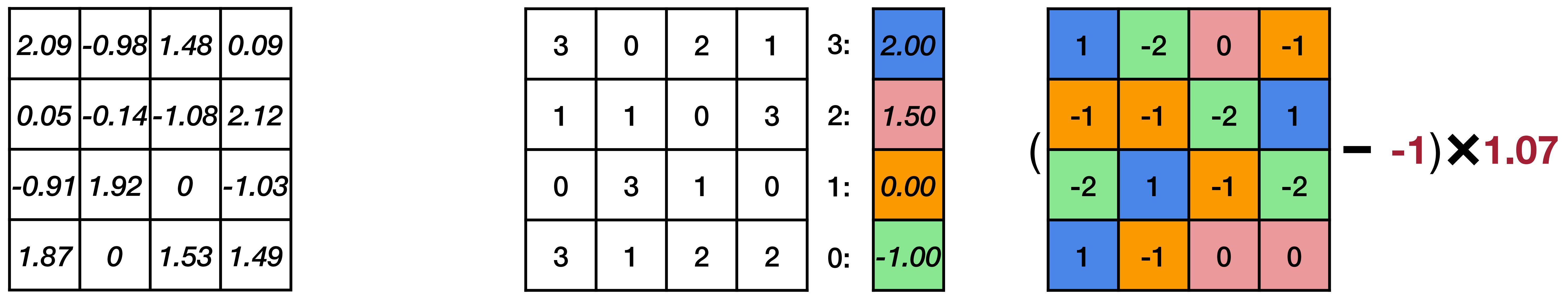
3:	*2.00*
2:	*1.50*
1:	*0.00*
0:	*-1.00*

$$\left(\begin{array}{cccc} 1 & -2 & 0 & -1 \\ -1 & -1 & -2 & 1 \\ -2 & 1 & -1 & -2 \\ 1 & -1 & 0 & 0 \end{array} - -1\right) \times 1.07$$

		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

→ **?**

Neural Network Quantization



$$\begin{array}{|c|c|c|c|}\hline 2.09 & -0.98 & 1.48 & 0.09 \\ \hline 0.05 & -0.14 & -1.08 & 2.12 \\ \hline -0.91 & 1.92 & 0 & -1.03 \\ \hline 1.87 & 0 & 1.53 & 1.49 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|c|}\hline 3 & 0 & 2 & 1 \\ \hline 1 & 1 & 0 & 3 \\ \hline 0 & 3 & 1 & 0 \\ \hline 3 & 1 & 2 & 2 \\ \hline \end{array} \quad \begin{array}{r|c|}\hline 3: & 2.00 \\ \hline 2: & 1.50 \\ \hline 1: & 0.00 \\ \hline 0: & -1.00 \\ \hline \end{array}$$

$$\left(\begin{array}{|c|c|c|c|}\hline 1 & -2 & 0 & -1 \\ \hline -1 & -1 & -2 & 1 \\ \hline -2 & 1 & -1 & -2 \\ \hline 1 & -1 & 0 & 0 \\ \hline \end{array} - -1 \right) \times 1.07$$

$$\begin{array}{|c|c|c|c|}\hline 1 & 0 & 1 & 1 \\ \hline 1 & 0 & 0 & 1 \\ \hline 0 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 \\ \hline \end{array}$$

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

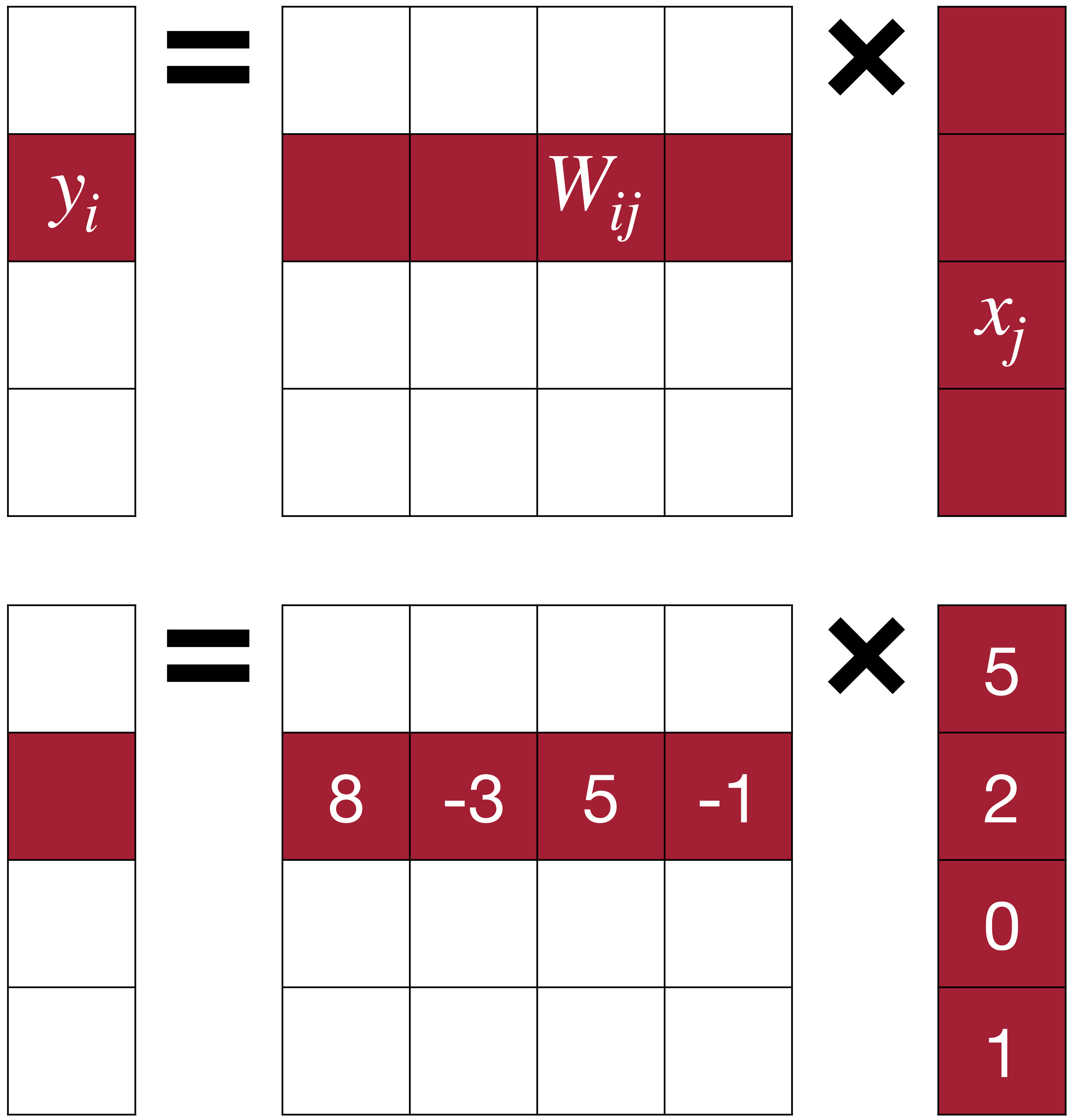
Binary/Ternary Quantization

Can we push the quantization precision to 1 bit?

Can quantization bit width go even lower?

$$y_i = \sum_j W_{ij} \cdot x_j$$

$$= 8×5 + (-3)×2 + 5×0 + (-1)×1$$

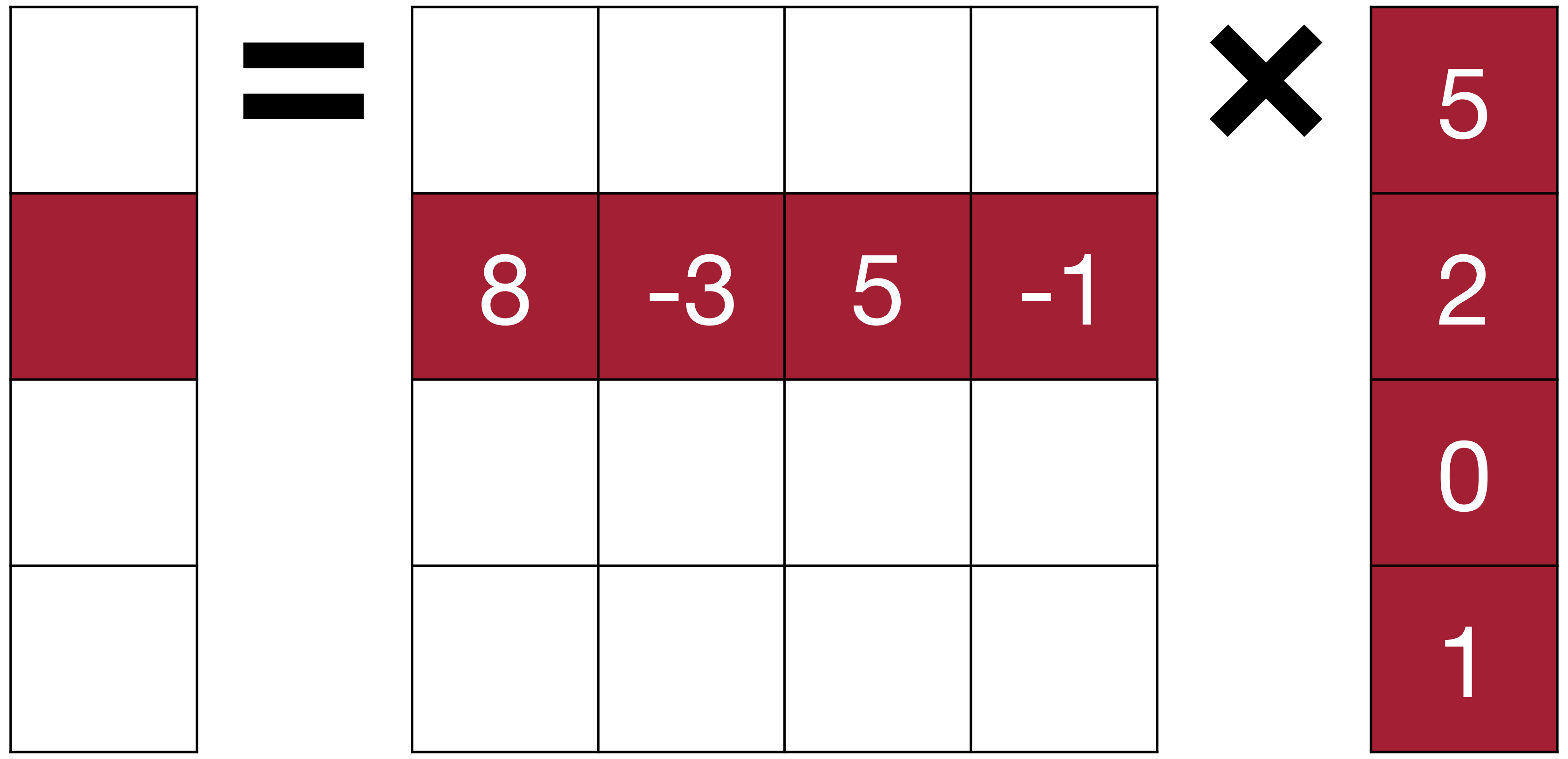


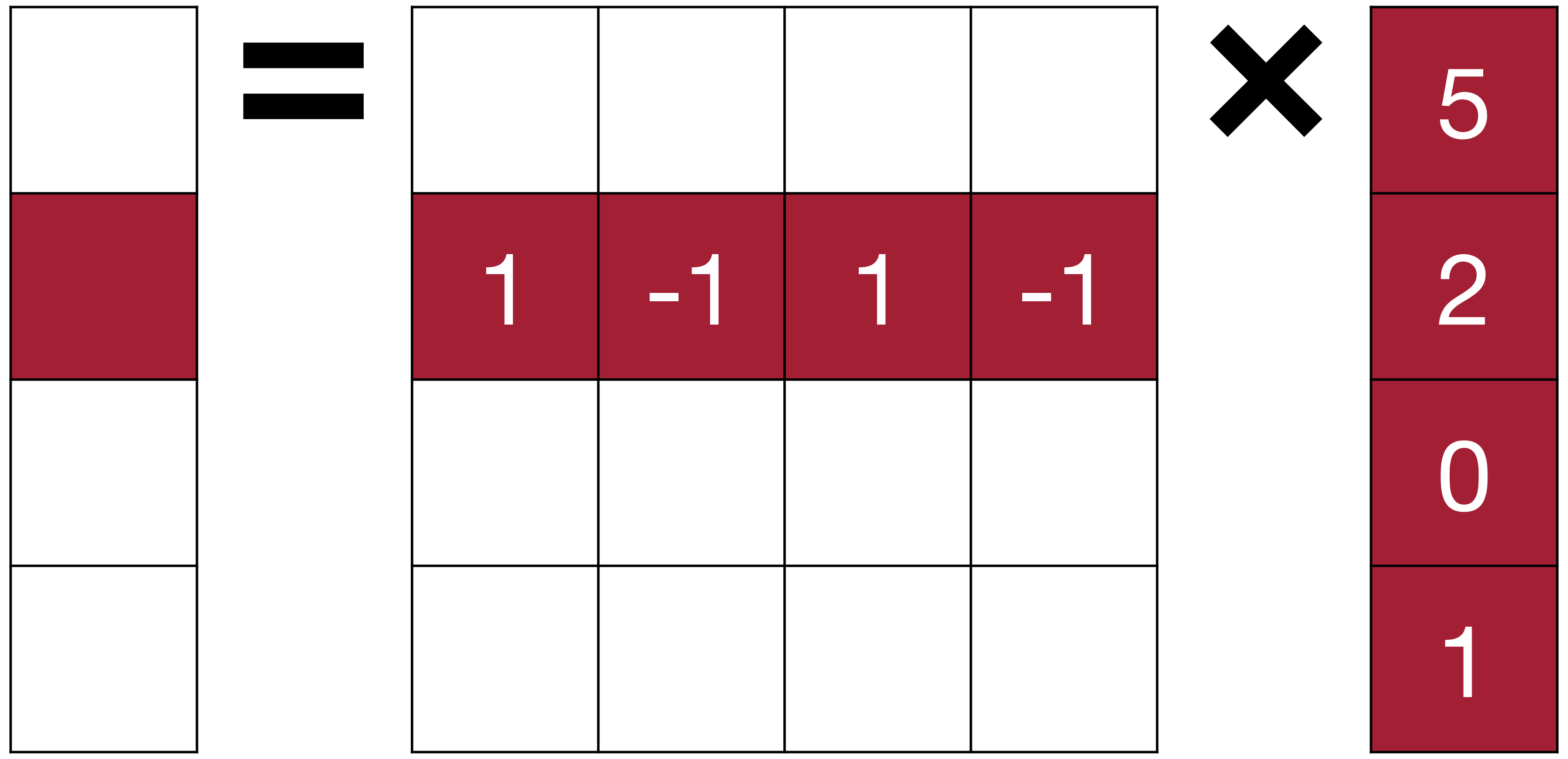
input	weight	operations	memory	computation
$\mathbb{R}$	$\mathbb{R}$	+ ×	1×	1×

If weights are quantized to +1 and -1

$$y_i = \sum_j W_{ij} \cdot x_j$$

$= 5 - 2 + 0 - 1$





input	weight	operations	memory	computation
$\mathbb{R}$	$\mathbb{R}$	+ ×	1×	1×
$\mathbb{R}$	$\mathbb{B}$	+ -	~32× less	~2× less

BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations [Courbariaux *et al.*, NeurIPS 2015]
XNOR-Net: ImageNet Classification using Binary Convolutional Neural Networks [Rastegari *et al.*, ECCV 2016]

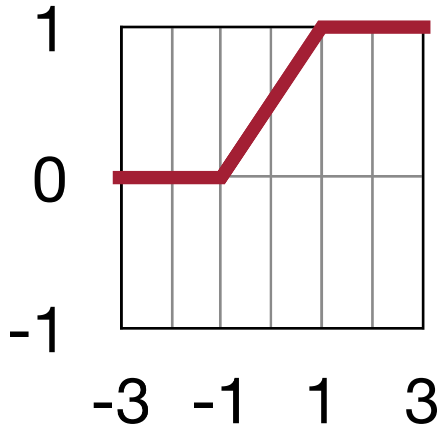
Binarization

- **Deterministic Binarization**
  - directly computes the bit value based on a threshold, usually 0, resulting in a sign function.

$$q = \text{sign}(r) = \begin{cases} +1, & r \geq 0 \\ -1, & r < 0 \end{cases}$$

- **Stochastic Binarization**
  - use global statistics or the value of input data to determine the probability of being -1 or +1
    - *e.g.*, in Binary Connect (BC), probability is determined by hard sigmoid function $\sigma(r)$

$$q = \begin{cases} +1, & \text{with probability } p = \sigma(r) \\ -1, & \text{with probability } 1 - p \end{cases}, \quad \text{where } \sigma(r) = \min(\max(\frac{r+1}{2}, 0), 1)$$



    - harder to implement as it requires the hardware to generate random bits when quantizing.

BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations [Courbariaux *et al.*, NeurIPS 2015]
BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or –1. [Courbariaux *et al.*, Arxiv 2016]

Minimizing Quantization Error in Binarization

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*

$$\begin{tikzcd} \mathbf{W} \arrow[r] \arrow[rd] & \mathbf{W}^{\mathbb{B}} \\ & \alpha\mathbf{W}^{\mathbb{B}} \end{tikzcd}$$

binary weights
(1-bit)

1	-1	1	1
1	-1	-1	1
-1	1	1	-1
1	1	1	1

$$\|\mathbf{W} - \mathbf{W}^{\mathbb{B}}\|_F^2 = 9.28$$

AlexNet-based Network	ImageNet Top-1 Accuracy Delta
BinaryConnect	-21.2%
Binary Weight Network (BWN)	0.2%

$$\mathbf{W}^{\mathbb{B}} = \mathrm{sign}\left(\mathbf{W}\right)$$

$$\alpha = \frac{1}{n}\|\mathbf{W}\|_1$$

1	-1	1	1
1	-1	-1	1
-1	1	1	-1
1	1	1	1

scale
(32-bit float)

$$\times \ \mathbf{1.05} = \frac{1}{16}\|\mathbf{W}\|_1$$

$$\|\mathbf{W} - \alpha\mathbf{W}^{\mathbb{B}}\|_F^2 = 9.24$$

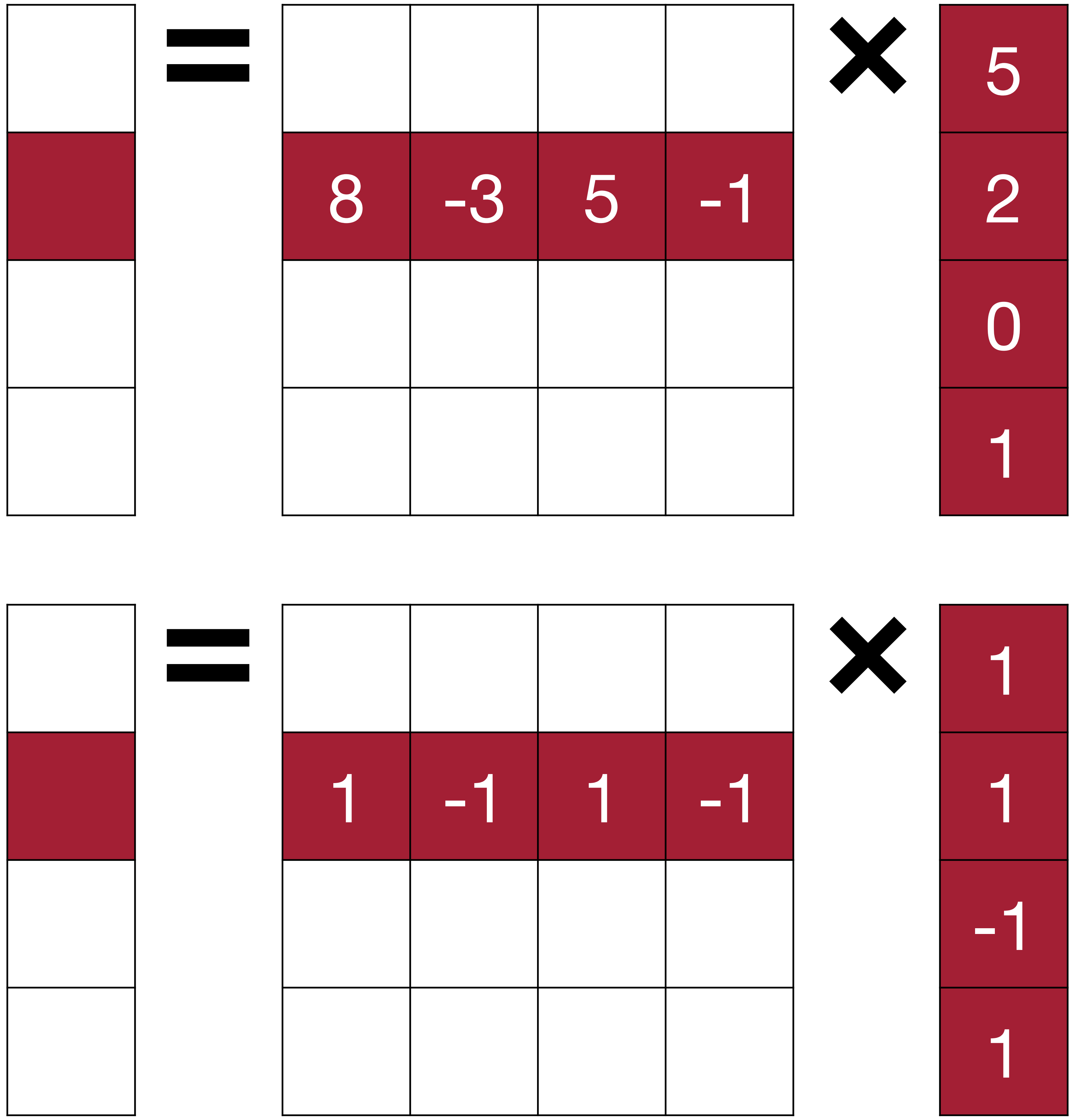
XNOR-Net: ImageNet Classification using Binary Convolutional Neural Networks [Rastegari *et al.*, ECCV 2016]

If both activations and weights are binarized

$$y_i = \sum_j W_{ij} \cdot x_j$$

$$= 1\times1 + (-1)\times1 + 1\times(-1) + (-1)\times1$$

$$= 1 + (-1) + (-1) + (-1) = -2$$



$$\begin{bmatrix} \ \\ \ \\ \ \\ \ \end{bmatrix} = \begin{bmatrix} & & & \\ 8 & -3 & 5 & -1 \\ & & & \\ & & & \end{bmatrix} \times \begin{bmatrix} 5 \\ 2 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \ \\ \ \\ \ \\ \ \end{bmatrix} = \begin{bmatrix} & & & \\ 1 & -1 & 1 & -1 \\ & & & \\ & & & \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

XNOR-Net: ImageNet Classification using Binary Convolutional Neural Networks [Rastegari *et al.*, ECCV 2016]

If both activations and weights are binarized

$$y_i = \sum_j W_{ij} \cdot x_j$$

$$= \boxed{1 \times 1 + (-1) \times 1 + 1 \times (-1) + (-1) \times 1}$$

$$= 1 + (-1) + (-1) + (-1) = -2$$

W	X	Y=WX
1	1	1
1	-1	-1
-1	-1	1
-1	1	-1

b_W	b_X	XNOR(b_W, b_X)
1	1	1
1	0	0
0	0	1
0	1	0

XNOR-Net: ImageNet Classification using Binary Convolutional Neural Networks [Rastegari *et al.*, ECCV 2016]

If both activations and weights are binarized

$$y_i = \sum_j W_{ij} \cdot x_j$$

$$= 1\times1 + (-1)\times1 + 1\times(-1) + (-1)\times1 \qquad = \boxed{1 \text{ xnor } 1 + 0 \text{ xnor } 1 + 1 \text{ xnor } 0 + 0 \text{ xnor } 1}$$

$$= 1 + (-1) + (-1) + (-1) = -2 \qquad = 1 + 0 + 0 + 0 \ = \ 1$$

?

W	X	Y=WX
1	1	1
1	-1	-1
-1	-1	1
-1	1	-1

b_W	b_X	XNOR(b_W, b_X)
1	1	1
1	0	0
0	0	1
0	1	0

XNOR-Net: ImageNet Classification using Binary Convolutional Neural Networks [Rastegari *et al.*, ECCV 2016]

If both activations and weights are binarized

$$y_i = \sum_j W_{ij} \cdot x_j$$

$$= 1 \times 1 + (-1) \times 1 + 1 \times (-1) + (-1) \times 1$$

$$= 1 + (-1) + (-1) + (-1) = -2$$

$$y_i = -n + 2 \cdot \sum_j W_{ij} \text{ xnor } x_j$$

$$= 1 \text{ xnor } 1 + 0 \text{ xnor } 1 + 1 \text{ xnor } 0 + 0 \text{ xnor } 1$$

$$= \underset{\uparrow +2}{1} + 0 + 0 + 0 = \boxed{\begin{array}{c} 1 \times 2 \\ + \\ -4 \end{array}} = -2$$

Assuming -1 -1 -1 -1 → -4

W	X	Y=WX
1	1	1
1	-1	-1
-1	-1	1
-1	1	-1

b_W	b_X	XNOR(b_W, b_X)
1	1	1
1	0	0
0	0	1
0	1	0

XNOR-Net: ImageNet Classification using Binary Convolutional Neural Networks [Rastegari *et al.*, ECCV 2016]

If both activations and weights are binarized

$$y_i = -n + 2 \cdot \boxed{\sum_j} W_{ij} \text{ xnor } x_j \quad \rightarrow \quad y_i = -n + \text{popcount}\left(W_i \text{ xnor } x\right) \ll 1$$

$$= -4 + 2 \times (1 \text{ xnor } 1 + 0 \text{ xnor } 1 + 1 \text{ xnor } 0 + 0 \text{ xnor } 1)$$

$$= -4 + 2 \times \boxed{(1 + 0 + 0 + 0)} = -2$$

→ popcount: return the number of 1

W	X	Y=WX
1	1	1
1	-1	-1
-1	-1	1
-1	1	-1

b_W	b_X	XNOR(b_W, b_X)
1	1	1
1	0	0
0	0	1
0	1	0

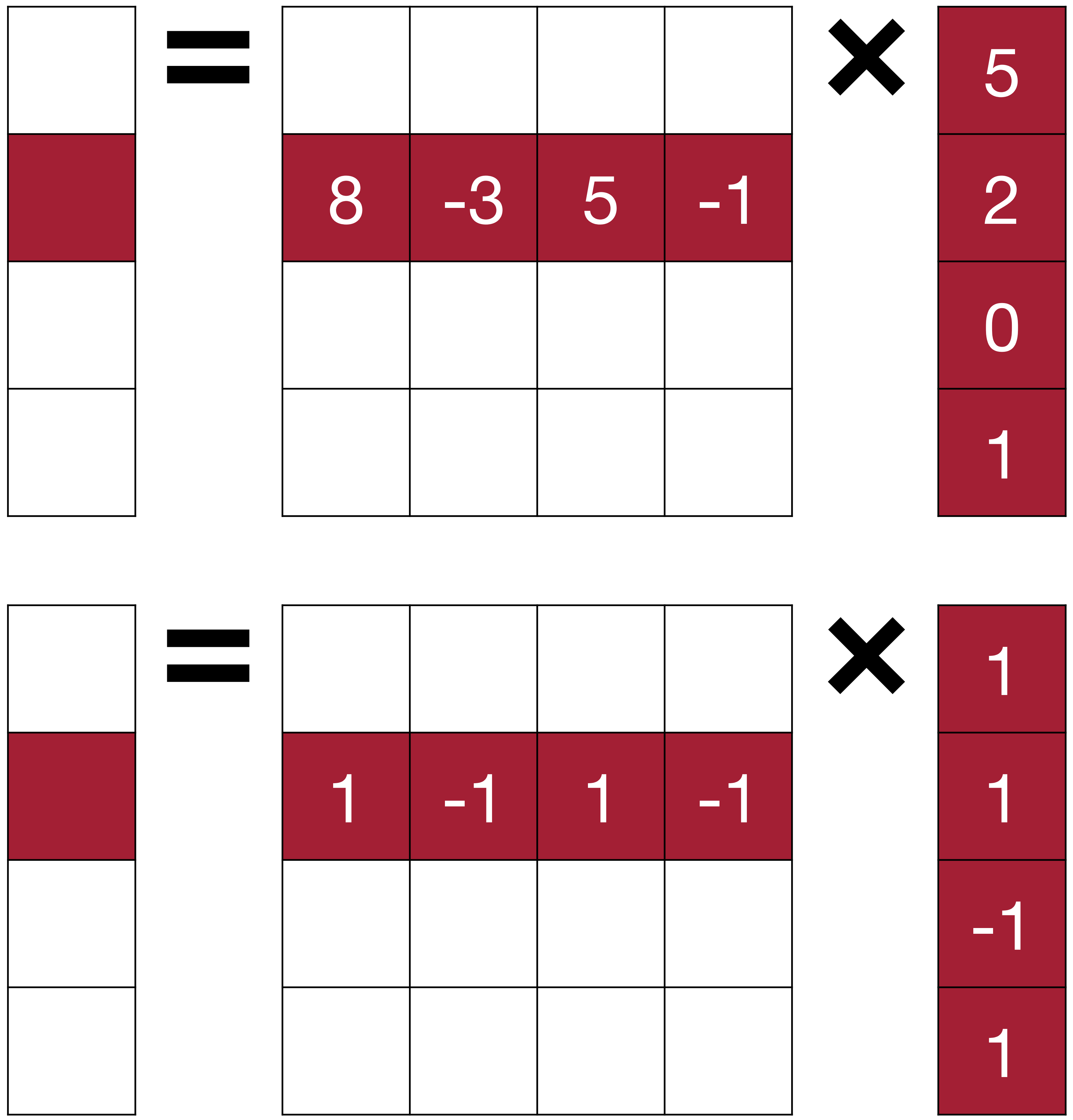
XNOR-Net: ImageNet Classification using Binary Convolutional Neural Networks [Rastegari *et al.*, ECCV 2016]

If both activations and weights are binarized

$$y_i = -n + \text{popcount}\left(W_i \text{ xnor } x\right) \ll 1$$

$$= -4 + \text{popcount}(1010 \text{ xnor } 1101) \ll 1$$

$$= -4 + \text{popcount}(1000) \ll 1 = -4 + 2 = -2$$



input	weight	operations	memory	computation
$\mathbb{R}$	$\mathbb{R}$	+ ×	1×	1×
$\mathbb{R}$	$\mathbb{B}$	+ -	~32× less	~2× less
$\mathbb{B}$	$\mathbb{B}$	xnor, popcount	~32× less	~58× less

XNOR-Net: ImageNet Classification using Binary Convolutional Neural Networks [Rastegari *et al.*, ECCV 2016]

Accuracy Degradation of Binarization

Neural Network	Quantization	Bit-Width		ImageNet Top-1 Accuracy Delta
		W	A	
AlexNet	BWN	1	32	0.2%
	BNN	1	1	-28.7%
	XNOR-Net	1	1	-12.4%
GoogleNet	BWN	1	32	-5.80%
	BNN	1	1	-24.20%
ResNet-18	BWN	1	32	-8.5%
	XNOR-Net	1	1	-18.1%

* BWN: Binary Weight Network with scale for weight binarization
* BNN: Binarized Neural Network without scale factors
* XNOR-Net: scale factors for both activation and weight binarization

Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or −1. [Courbariaux *et al.*, Arxiv 2016]
XNOR-Net: ImageNet Classification using Binary Convolutional Neural Networks [Rastegari *et al.*, ECCV 2016]

Ternary Weight Networks (TWN)

Weights are quantized to +1, -1 and 0

$$q = \begin{cases} r_t, & r > \Delta \\ 0, & |r| \leq \Delta, \\ -r_t, & r < -\Delta \end{cases} \quad \text{where } \Delta = 0.7 \times \mathbb{E}\left(|r|\right), r_t = \mathbb{E}_{|r|>\Delta}\left(|r|\right)$$

$$\begin{array}{c} \text{weights } \mathbf{W} \\ \text{(32-bit float)} \\ \begin{array}{|c|c|c|c|} \hline 2.09 & -0.98 & 1.48 & 0.09 \\ \hline 0.05 & -0.14 & -1.08 & 2.12 \\ \hline -0.91 & 1.92 & 0 & -1.03 \\ \hline 1.87 & 0 & 1.53 & 1.49 \\ \hline \end{array} \end{array} \longrightarrow \begin{array}{c} \text{ternary weights } \mathbf{W}^{\mathbb{T}} \\ \text{(2-bit)} \\ \begin{array}{|c|c|c|c|} \hline 1 & -1 & 1 & 0 \\ \hline 0 & 0 & -1 & 1 \\ \hline -1 & 1 & 0 & -1 \\ \hline 1 & 0 & 1 & 1 \\ \hline \end{array} \end{array}$$

$$\Delta = 0.7 \times \frac{1}{16}\|\mathbf{W}\|_1 = 0.73$$

$$\times \quad \mathbf{1.5} = \frac{1}{11}\|\mathbf{W}_{\mathbf{W}^{\mathbb{T}} \neq 0}\|_1$$

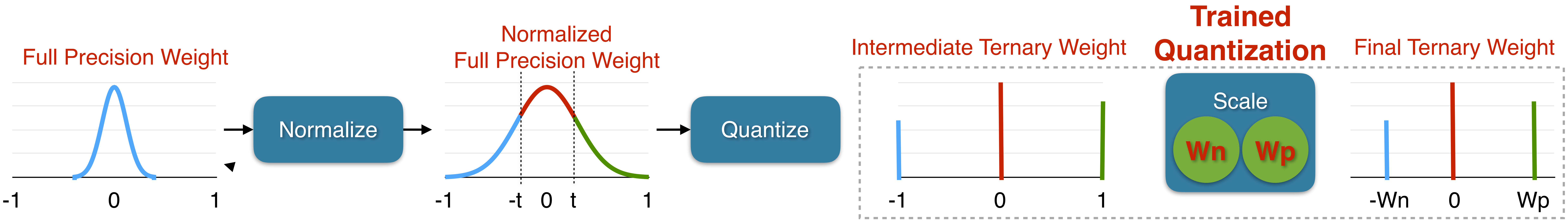
ImageNet Top-1 Accuracy	Full Precision	1 bit (BWN)	2 bit (TWN)
ResNet-18	69.6	60.8	65.3

Ternary Weight Networks [Li *et al.*, Arxiv 2016]

Trained Ternary Quantization (TTQ)

- Instead of using fixed scale r_t, TTQ introduces two *trainable* parameters w_p and w_n to represent the positive and negative scales in the quantization.

$$q = \begin{cases} w_p, & r > \Delta \\ 0, & |r| \leq \Delta \\ -w_n, & r < -\Delta \end{cases}$$



ImageNet Top-1 Accuracy	Full Precision	1 bit (BWN)	2 bit (TWN)	TTQ
ResNet-18	69.6	60.8	65.3	66.6

Trained Ternary Quantization [Zhu *et al.*, ICLR 2017]

Neural Network Quantization

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

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

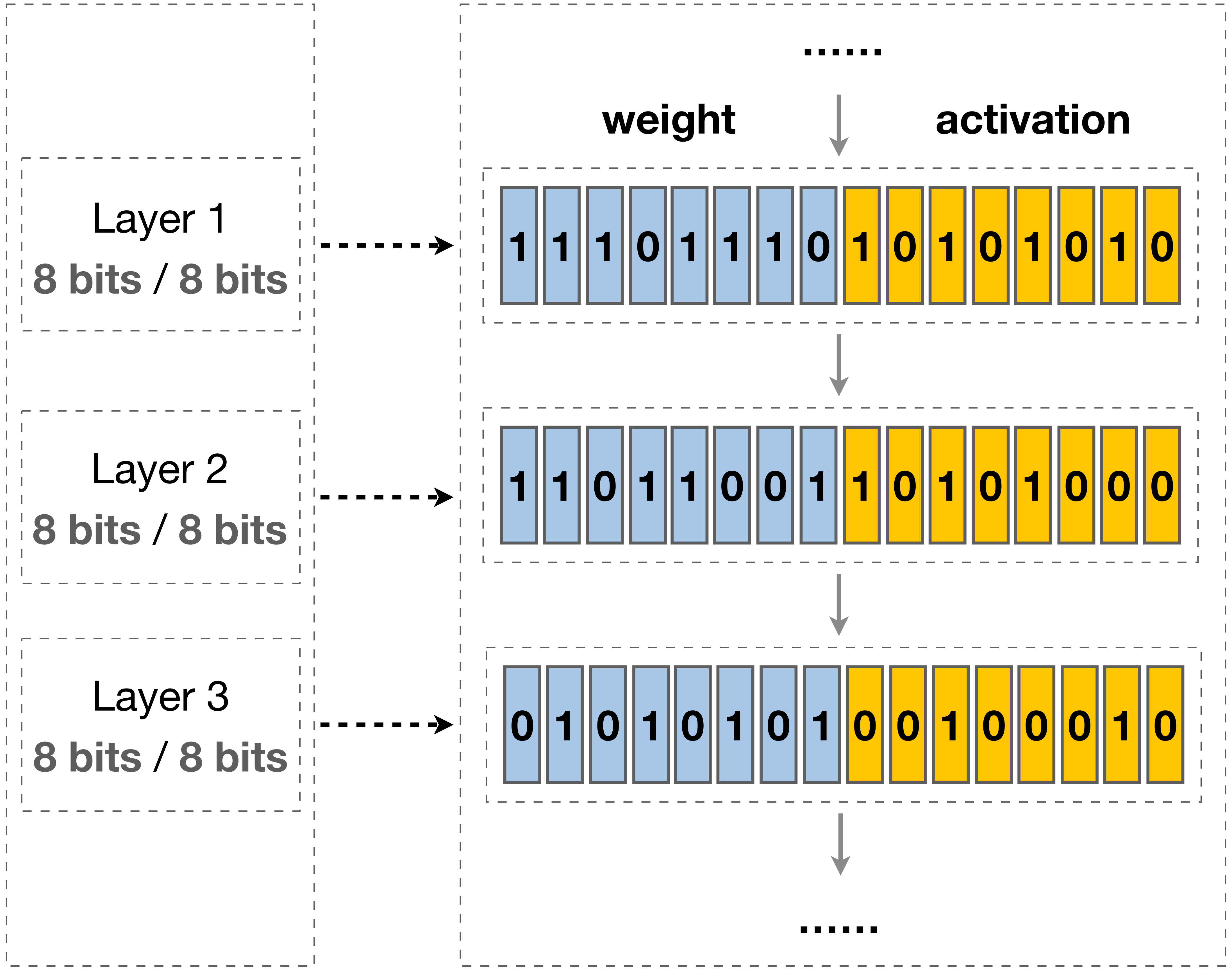
$$\left(\begin{array}{|c|c|c|c|}\hline 1 & -2 & 0 & -1 \\ \hline -1 & -1 & -2 & 1 \\ \hline -2 & 1 & -1 & -2 \\ \hline 1 & -1 & 0 & 0 \\ \hline \end{array} - -1\right)\times 1.07$$

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

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

Mixed-Precision Quantization

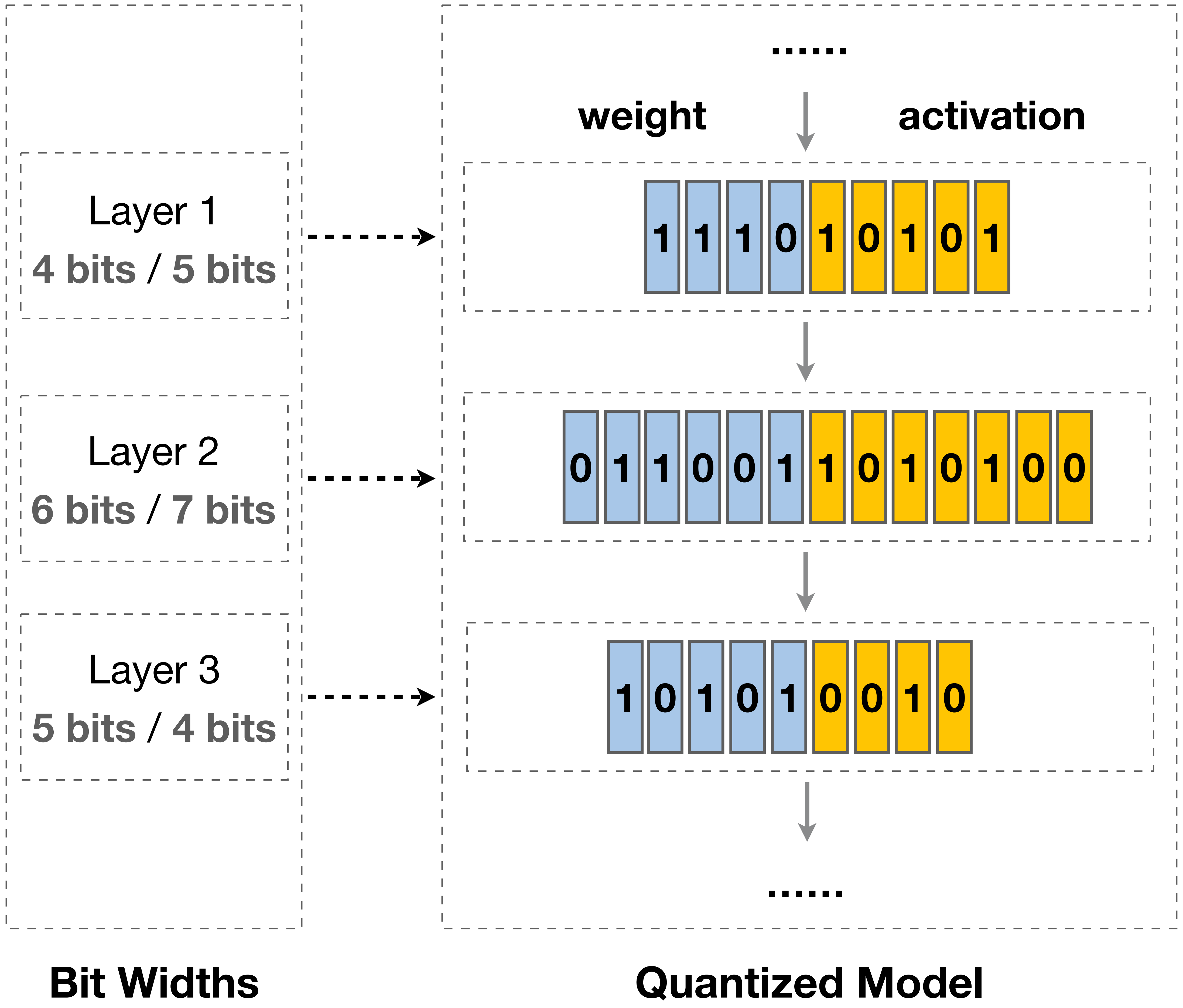
Uniform Quantization



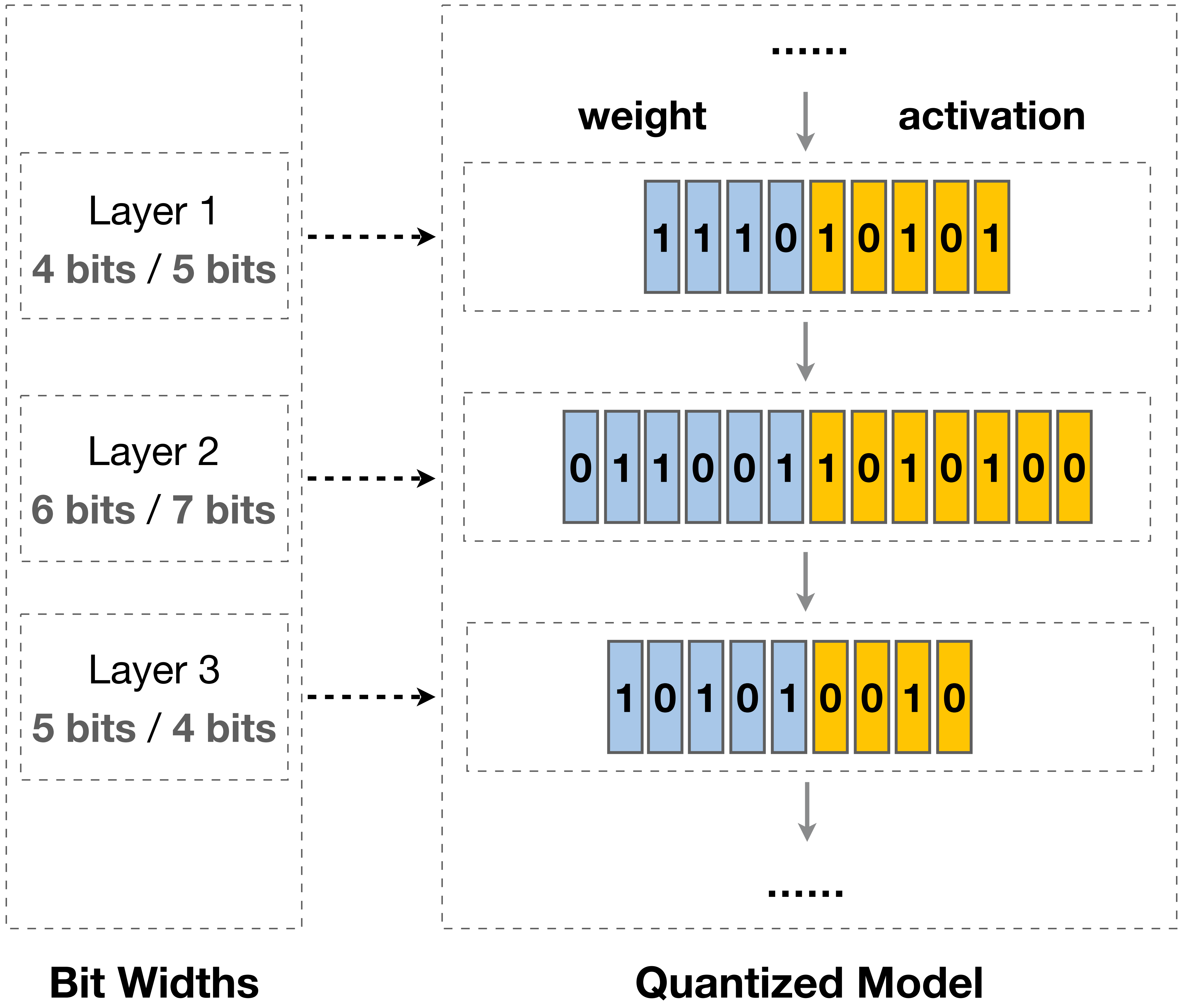
Bit Widths

Quantized Model

Mixed-Precision Quantization



Challenge: Huge Design Space



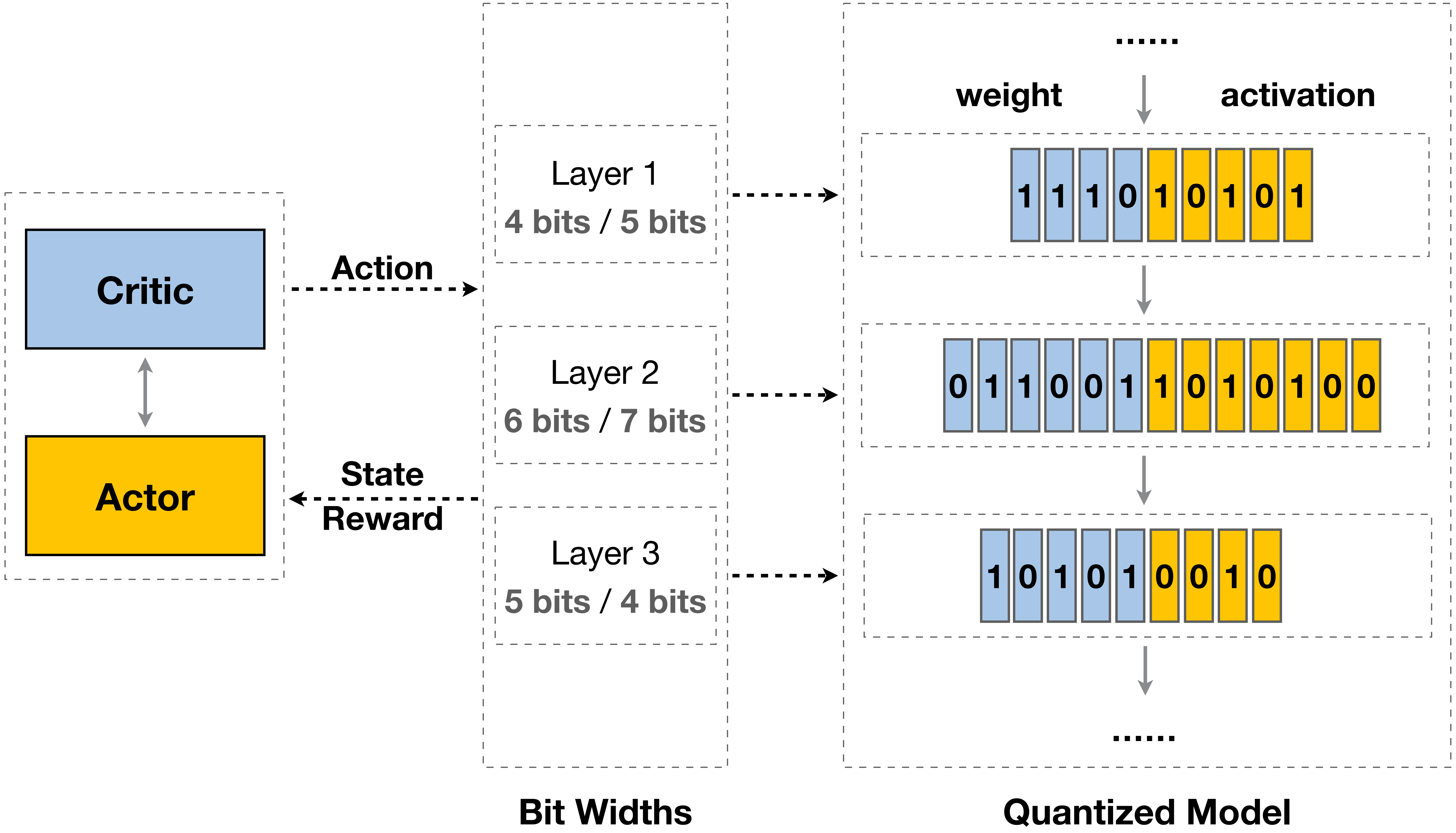
Choices: **8 x 8 = 64**

Choices: **8 x 8 = 64**

Choices: **8 x 8 = 64**

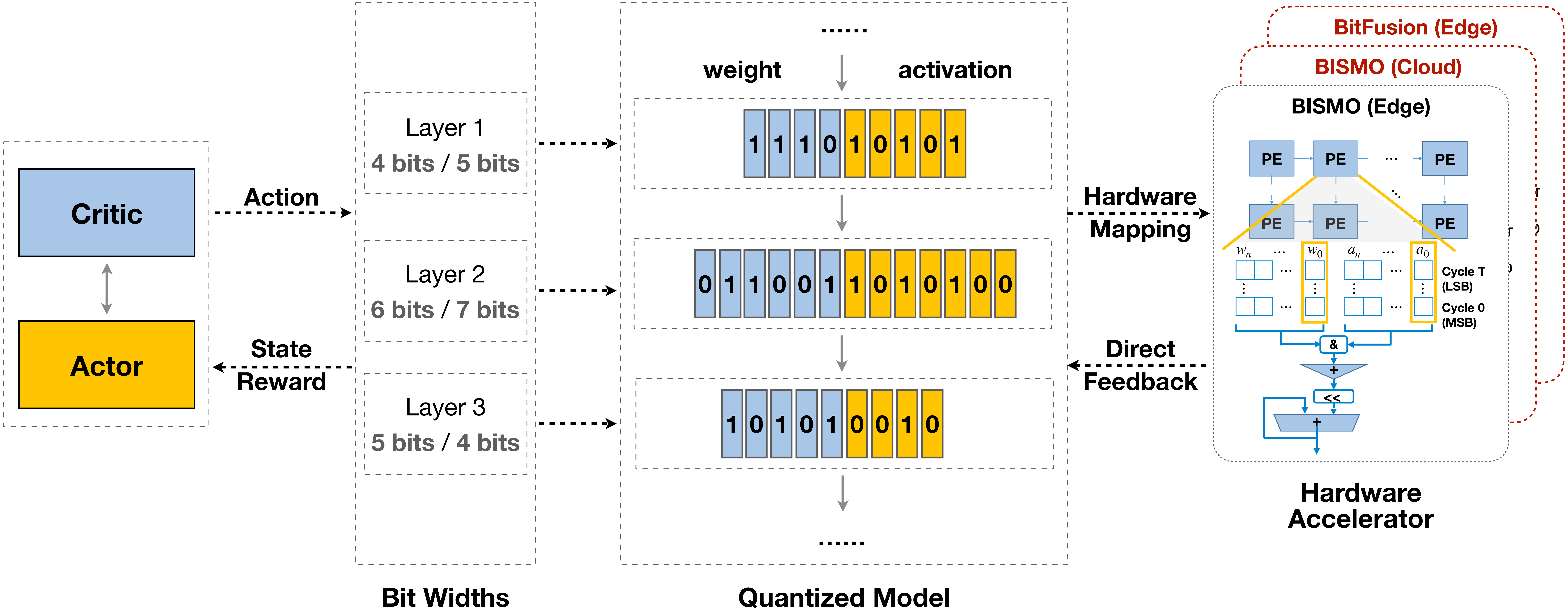
Design Space: 64^n

Solution: Design Automation



HAQ: Hardware-Aware Automated Quantization with Mixed Precision [Wang *et al.*, CVPR 2019]

Solution: Design Automation



Critic

Actor

Action

State

Reward

Layer 1

4 bits / 5 bits

Layer 2

6 bits / 7 bits

Layer 3

5 bits / 4 bits

Bit Widths

......

weight

activation

1 1 1 0 1 0 1 0 1

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

1 0 1 0 1 0 0 1 0

......

Quantized Model

Hardware Mapping

Direct Feedback

BitFusion (Edge)

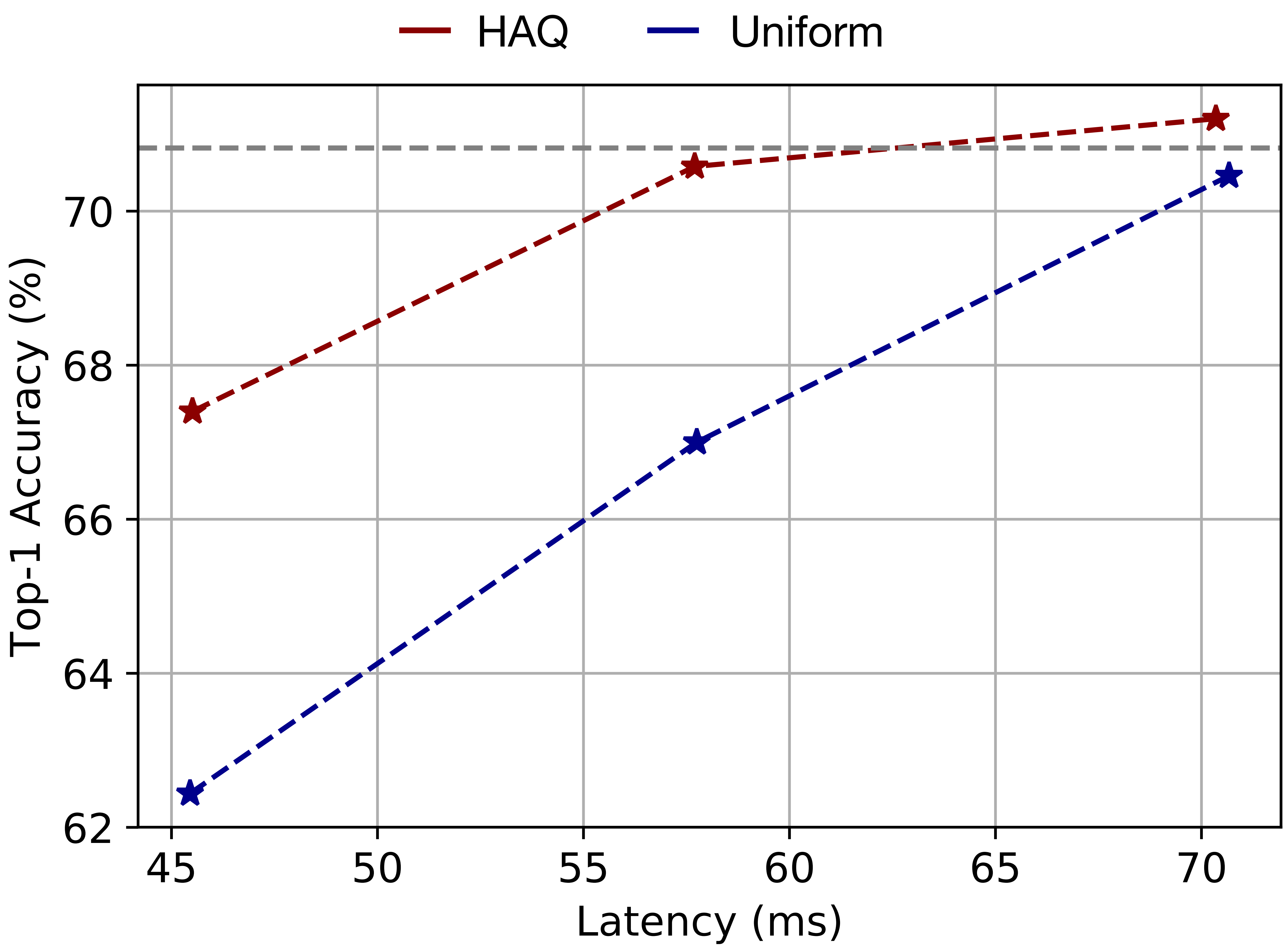
BISMO (Cloud)

BISMO (Edge)

Hardware Accelerator

HAQ: Hardware-Aware Automated Quantization with Mixed Precision [Wang *et al.*, CVPR 2019]

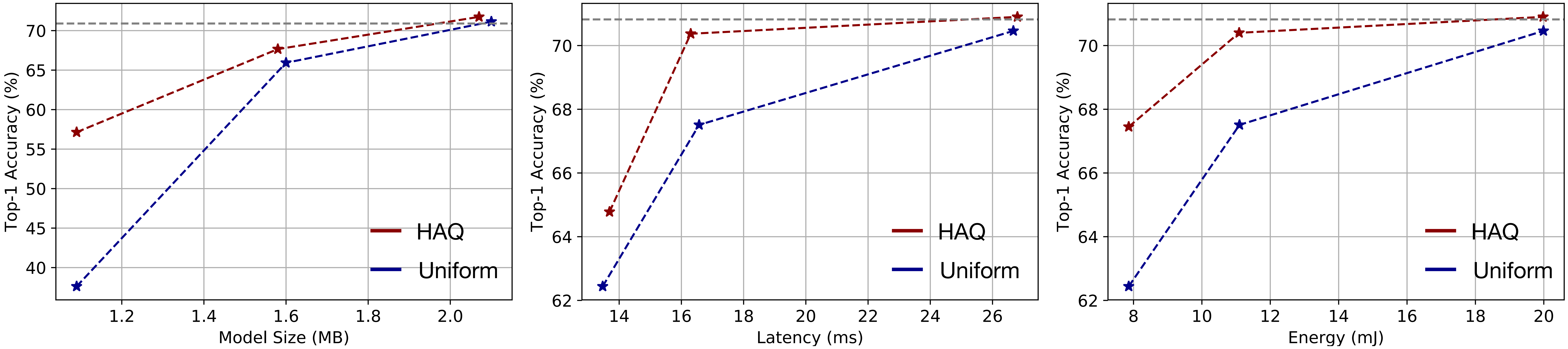
HAQ Outperforms Uniform Quantization



Mixed-Precision Quantized MobileNetV1

HAQ: Hardware-Aware Automated Quantization with Mixed Precision [Wang *et al.*, CVPR 2019]

HAQ Supports Multiple Objectives



Model Size Constrained

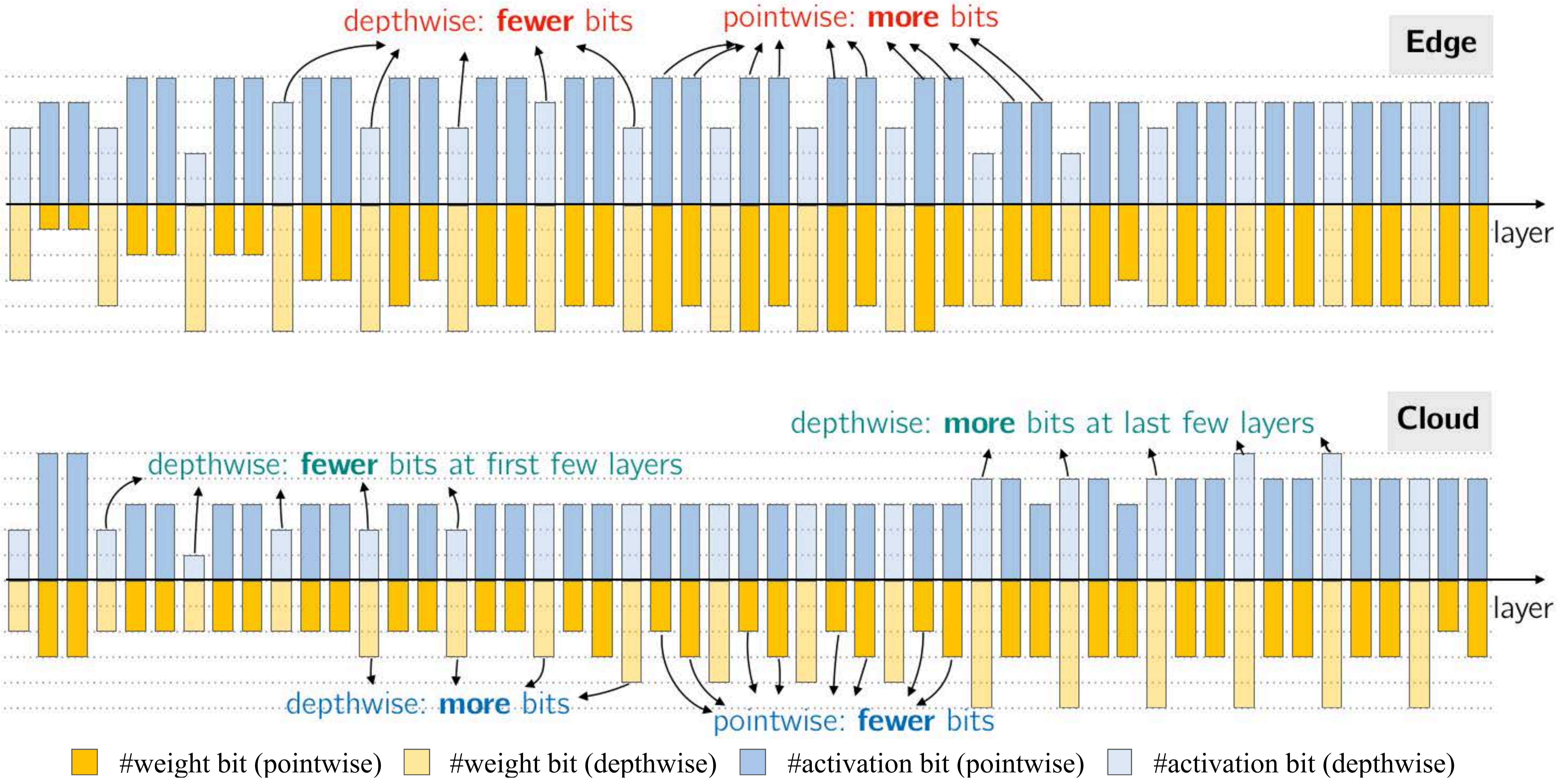
Latency Constrained

Energy Constrained

Mixed-Precision Quantized MobileNetV1

HAQ: Hardware-Aware Automated Quantization with Mixed Precision [Wang *et al.*, CVPR 2019]

Quantization Policy for Edge and Cloud



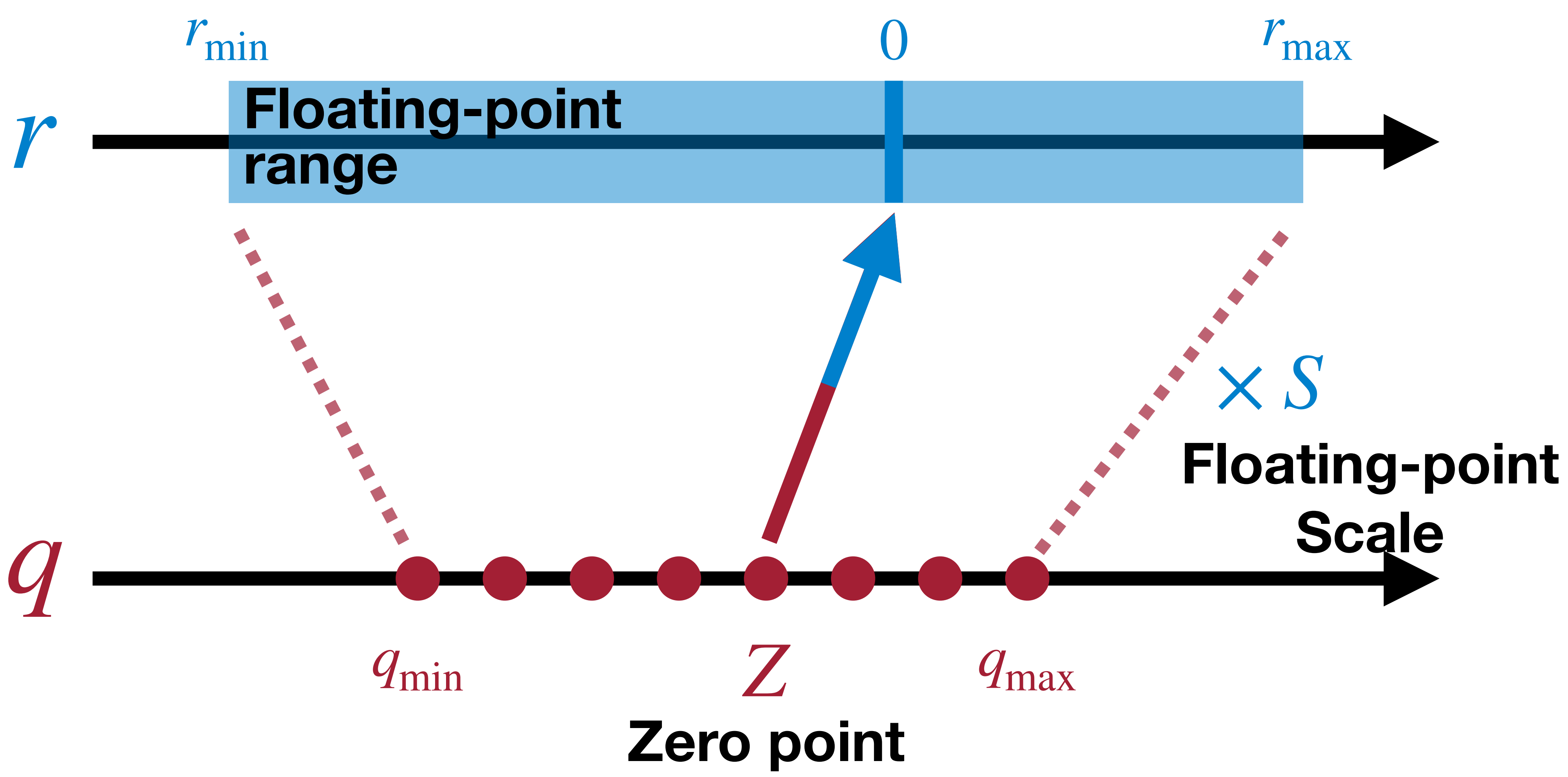
Mixed-Precision Quantized MobileNetV2

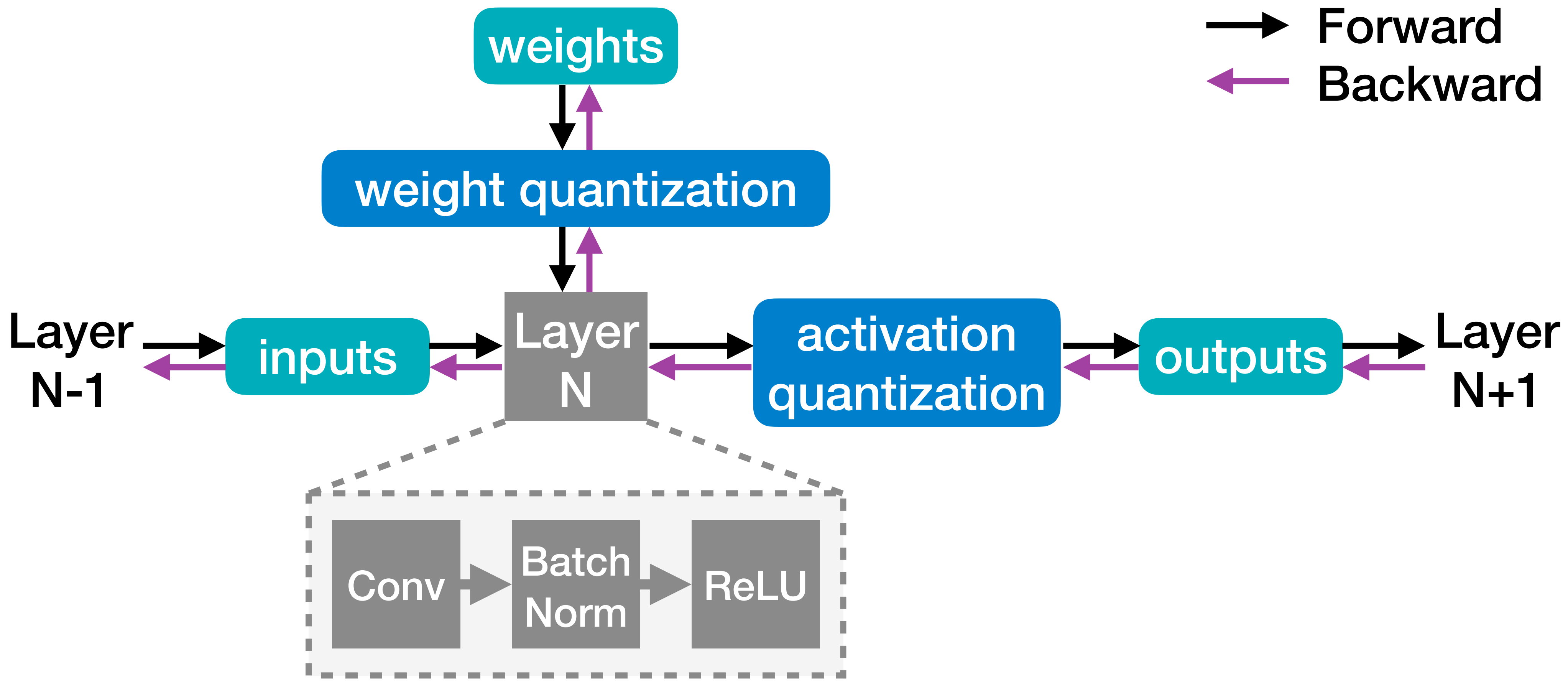
HAQ: Hardware-Aware Automated Quantization with Mixed Precision [Wang *et al.*, CVPR 2019]

Summary of Today’s Lecture

In this lecture, we

1. Reviewed Linear Quantization.
2. Introduced **Post-Training Quantization (PTQ)** that quantizes an already-trained floating-point neural network model.
    - Per-tensor *vs.* per-channel *vs.* group quantization
    - How to determine dynamic range for quantization
3. Introduced **Quantization-Aware Training (QAT)** that emulates inference-time quantization during the training/fine-tuning.
    - Straight-Through Estimator (STE)
4. Introduced **binary and ternary** quantization.
5. Introduced automatic **mixed-precision** quantization.





References

1. Deep Compression [Han et al., ICLR 2016]
2. Neural Network Distiller: https://intellabs.github.io/distiller/algo_quantization.html
3. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [Jacob et al., CVPR 2018]
4. Data-Free Quantization Through Weight Equalization and Bias Correction [Markus et al., ICCV 2019]
5. Post-Training 4-Bit Quantization of Convolution Networks for Rapid-Deployment [Banner et al., NeurIPS 2019]
6. 8-bit Inference with TensorRT [Szymon Migacz, 2017]
7. Quantizing Deep Convolutional Networks for Efficient Inference: A Whitepaper [Raghuraman Krishnamoorthi, arXiv 2018]
8. Neural Networks for Machine Learning [Hinton et al., Coursera Video Lecture, 2012]
9. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation [Bengio, arXiv 2013]
10. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or –1. [Courbariaux et al., Arxiv 2016]
11. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients [Zhou et al., arXiv 2016]
12. PACT: Parameterized Clipping Activation for Quantized Neural Networks [Choi et al., arXiv 2018]
13. WRPN: Wide Reduced-Precision Networks [Mishra et al., ICLR 2018]
14. Towards Accurate Binary Convolutional Neural Network [Lin et al., NeurIPS 2017]
15. Incremental Network Quantization: Towards Lossless CNNs with Low-precision Weights [Zhou et al., ICLR 2017]
16. HAQ: Hardware-Aware Automated Quantization with Mixed Precision [Wang et al., CVPR 2019]