HONGIK UNIVERSITY

2024년 7월 24일

# Quantization (2)

홍익대학교 컴퓨터공학과
C135283 이수현

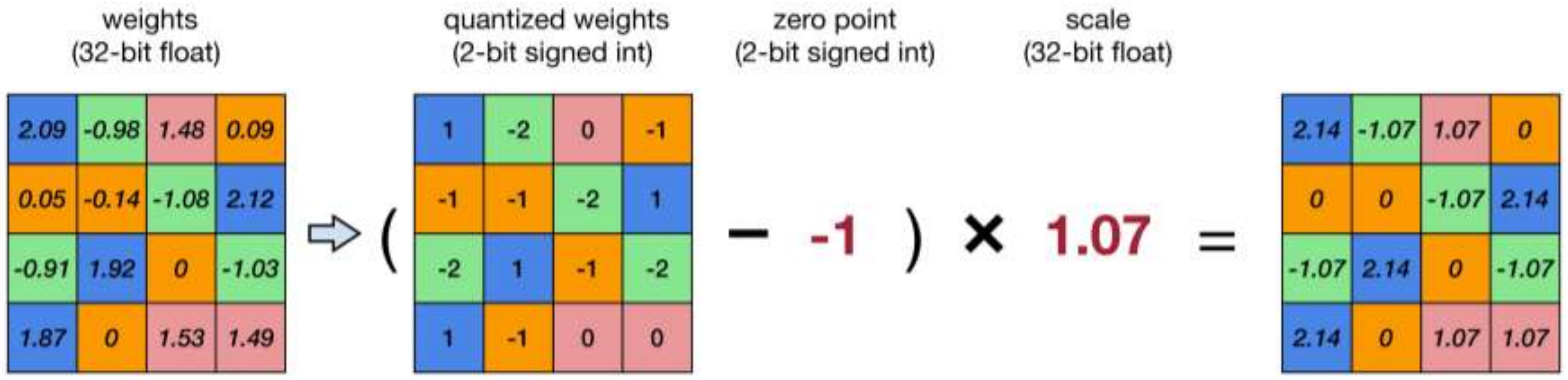
**Source:** Prof. Song Han at MIT, Lecture Slide on TinyML and Efficient Deep Learning Computing

# Contents

- Linear Quantization
- Post-Training Quantization (PTQ)
  - Quantization Granularity
  - Dynamic Range Clipping
  - Rounding
- Quantization-Aware Training (QAT)
- Binary and Ternary Quantization
- Mixed-Precision Quantization

# Linear Quantization

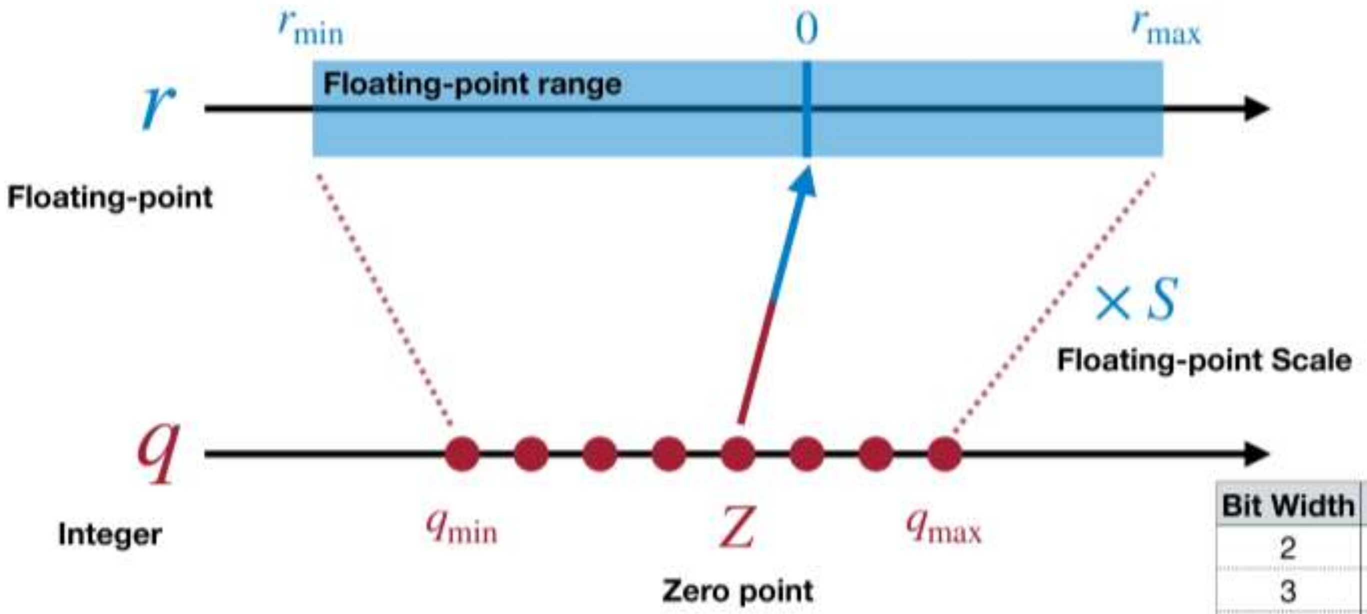
**An affine mapping of integers to real numbers** $r = S(q - Z)$




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

# Linear Quantization

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



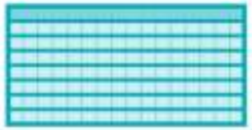

| Bit Width | $q_{min}$ | $q_{max}$ |
|---|---|---|
| 2 | -2 | 1 |
| 3 | -4 | 3 |
| 4 | -8 | 7 |
| $N$ | $-2^{N-1}$ | $2^{N-1}-1$ |

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

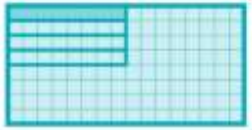
# Quantization Granularity

- Per-Tensor Quantization

- Per-Channel Quantization

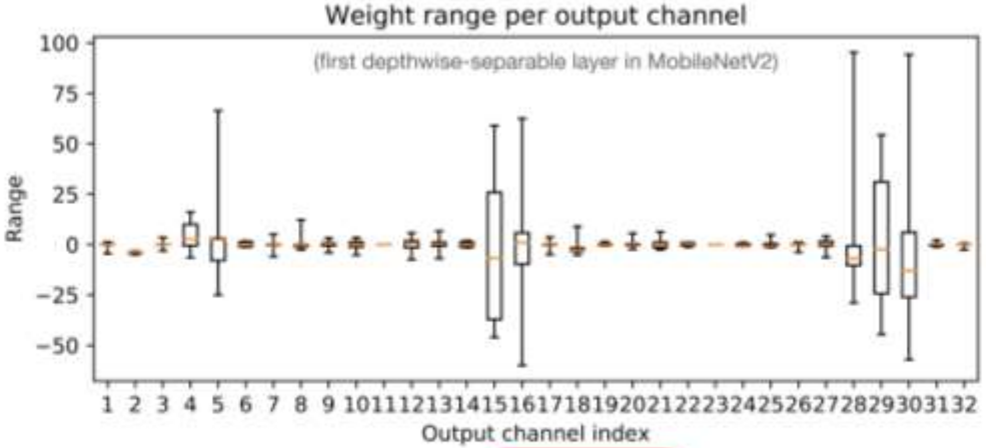


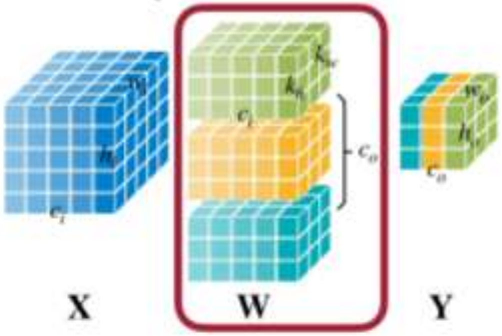
- Group Quantization



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

# Symmetric Linear Quantization on Weights

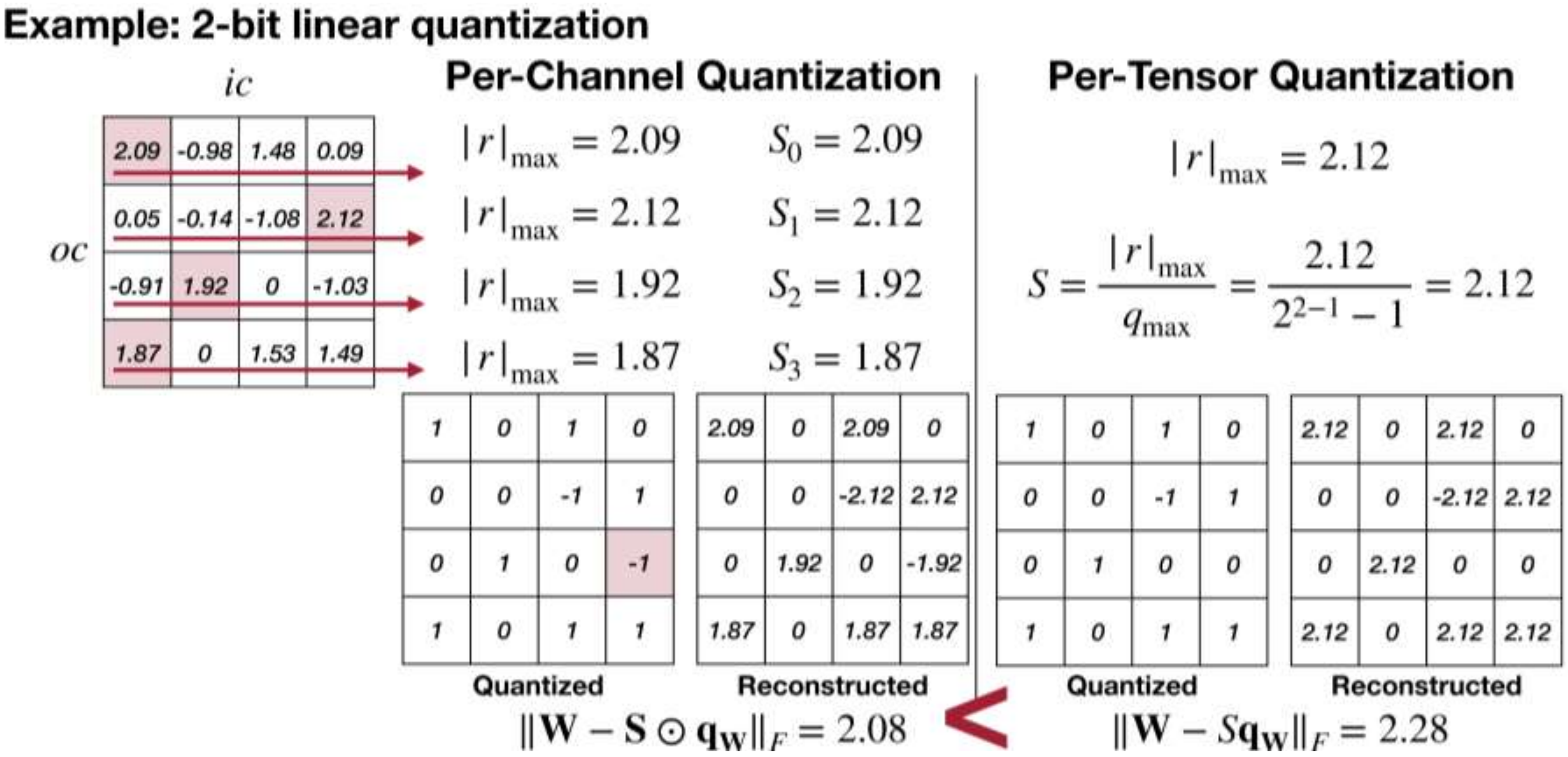



- $|r|_{\max} = |\mathbf{W}|_{\max}$
- Using *single* scale $S$ for whole weight tensor (**Per-Tensor Quantization**)
  - works well for large models
  - accuracy drops for small models
- Common failure results from
  - large differences (more than 100×) in ranges of weights for different output channels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

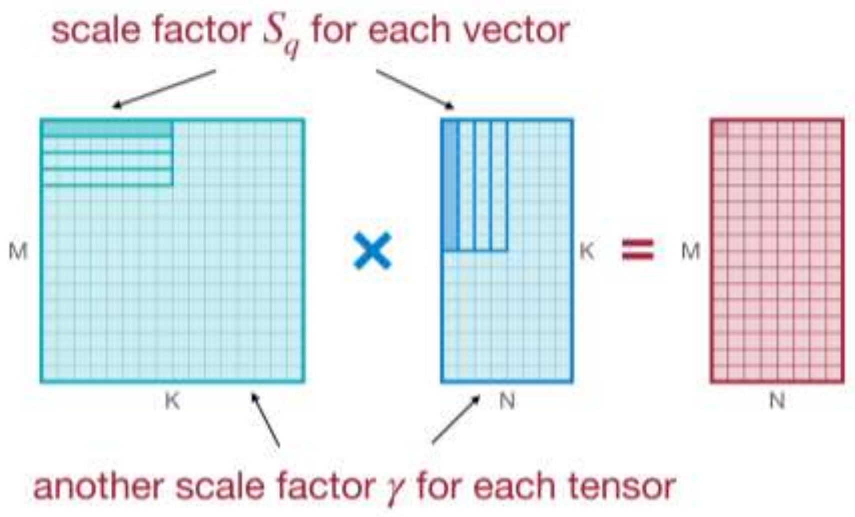
# Per-channel Weight Quantization



# VS-Quant: Per-vector Scaled Quantization

**Hierarchical scaling factor**

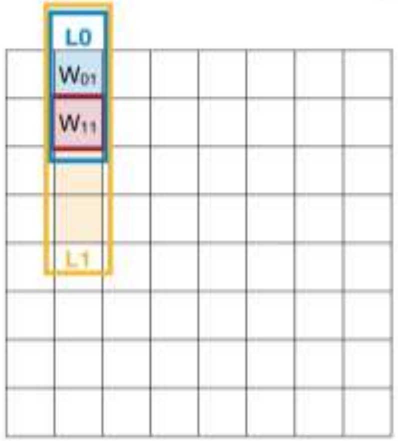
- $r = S(q - Z) \rightarrow r = \gamma \cdot S_q(q - Z)$
  - $\gamma$ is a floating-point coarse grained scale factor
  - $S_q$ is an integer per-vector scale factor
  - achieves a balance between accuracy and hardware efficiency by
    - less expensive integer scale factors at finer granularity
    - 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.




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

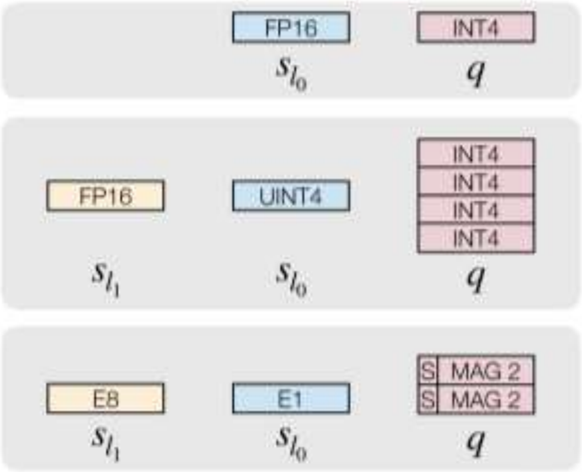
# Group Quantization

## Multi-level scaling scheme




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

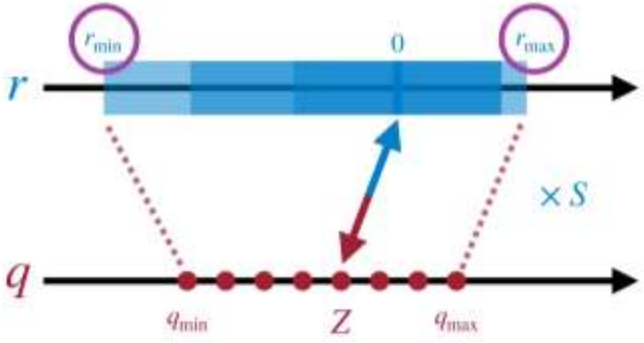
$r$ : real number value
$q$ : quantized value
$z$ : zero point ($z = 0$ is symmetric quantization)
$s$ : scale factors of different levels




| 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.*]

# Linear Quantization on Activations

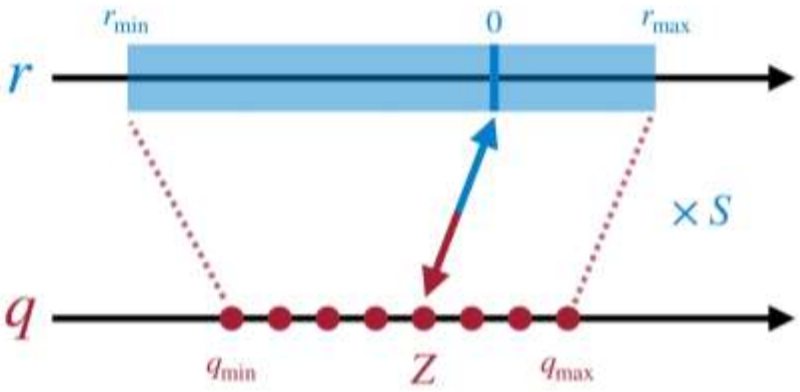



- Unlike weights, the activation range varies across inputs.
- To determine the floating-point range, the activations statistics are gathered ***before*** deploying the model.

# Dynamic Range for Activation Quantization

**Collect activations statistics before deploying the model**

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

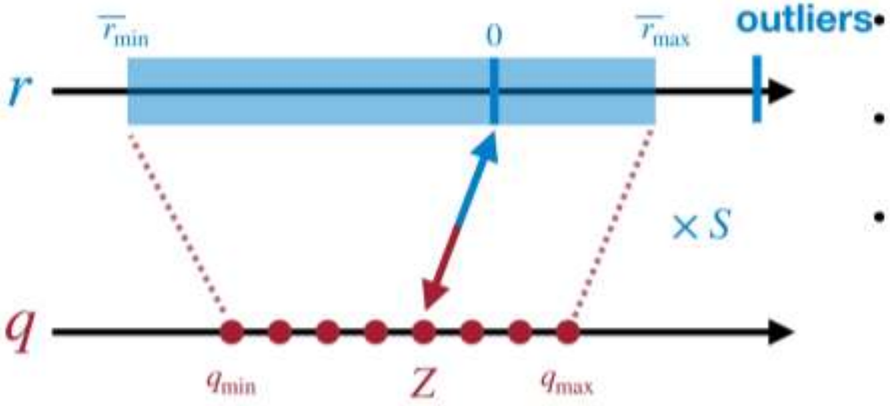



- Type 1: During training
  - Exponential moving averages (EMA)
    - observed ranges are smoothed across thousands of training steps

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**



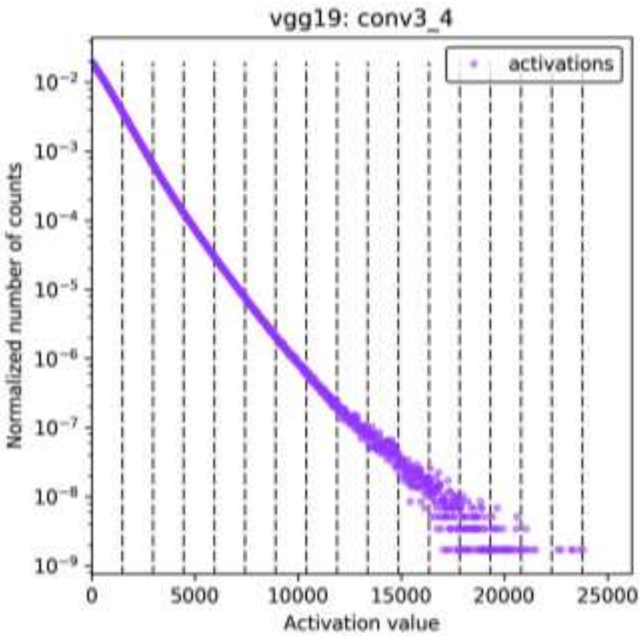

- Type 2: By running a few "calibration" batches of samples on the trained FP32 model
  - spending dynamic range on the outliers hurts the representation ability.
  - use *mean* of the min/max of each sample in the batches
  - analytical calculation (see next slide)

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 $P, Q$

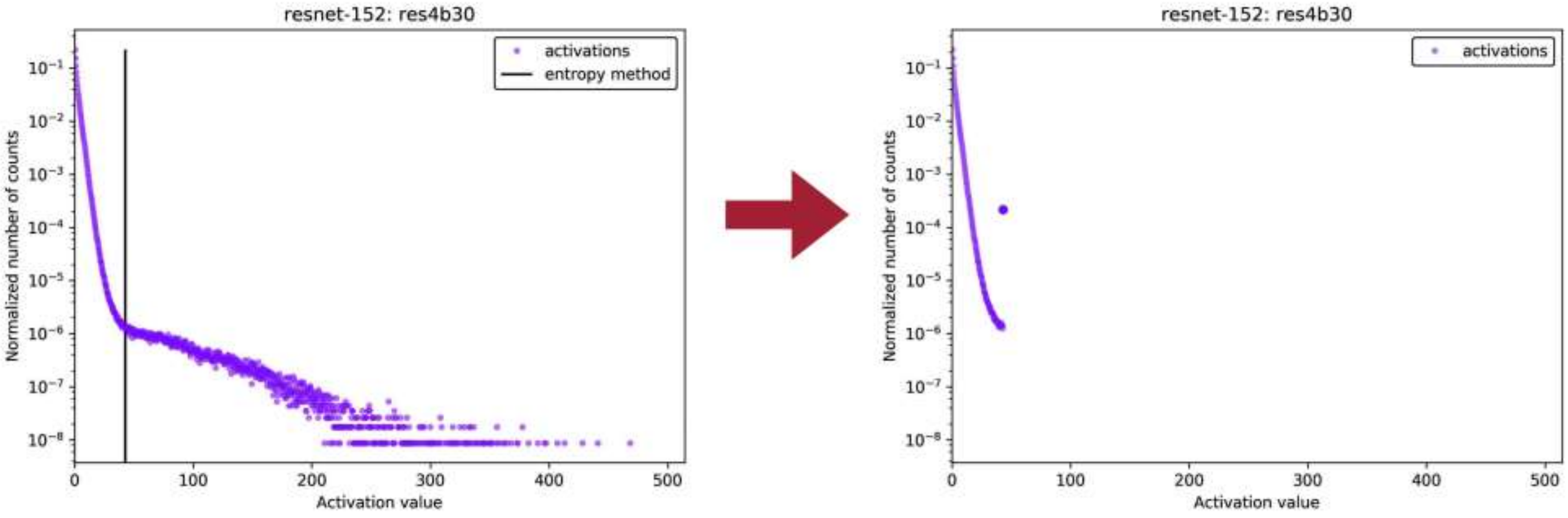
$$D_{KL}(P\|Q) = \sum_{i}^{N} P(x_i)\log\frac{P(x_i)}{Q(x_i)}$$

      - 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

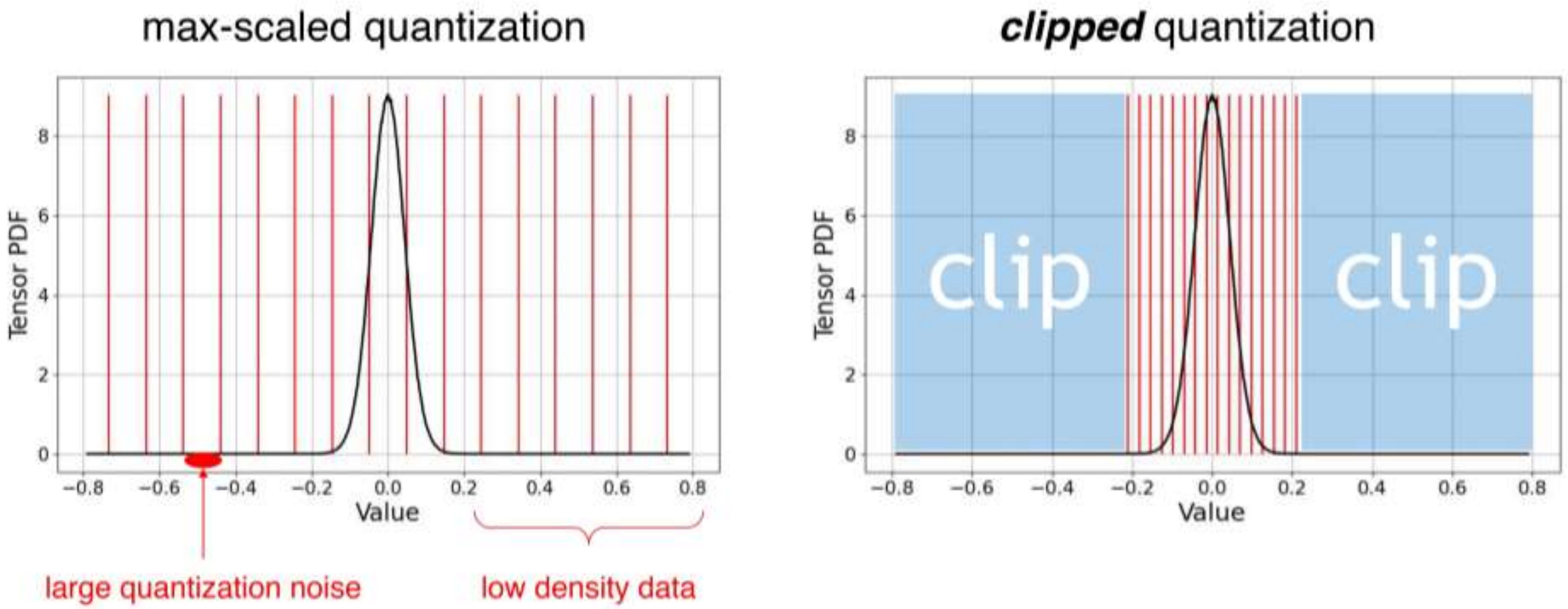
**Minimize loss of information by minimizing the KL divergence**




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

# Dynamic Range for Activation Quantization

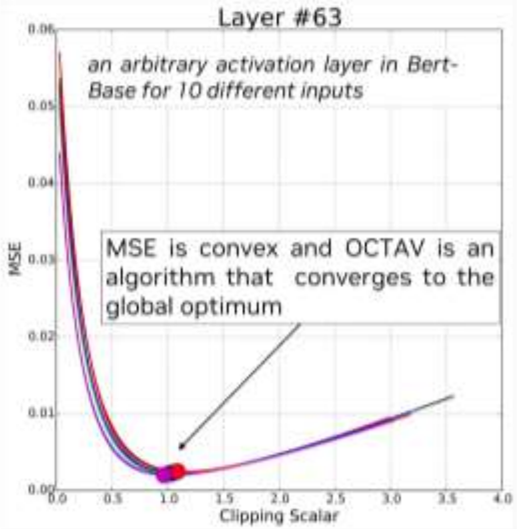
**Minimize mean-square-error (MSE) using Newton-Raphson method**



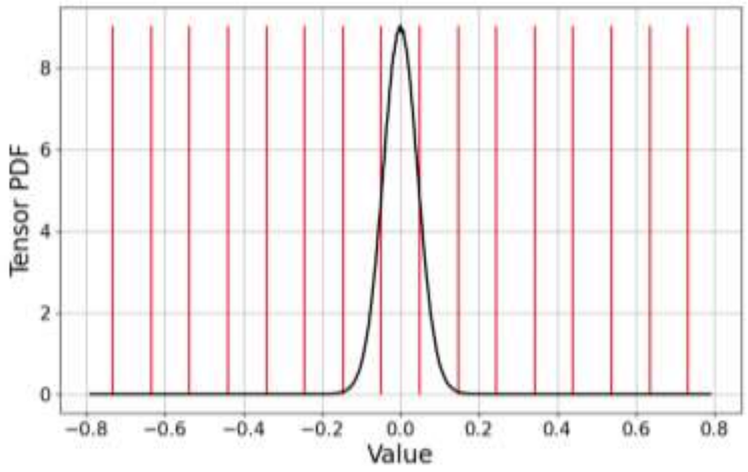

Optimal Clipping and Magnitude-aware Differentiation for Improved Quantization-aware Training [Sakr *et al.*, ICML 2022]

# Dynamic Range for Activation 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]

# 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

$$\begin{array}{|c|c|c|c|}\hline 0.3 & 0.5 & 0.7 & 0.2 \\ \hline\end{array} \xrightarrow{\text{rounding-to-nearest}} \begin{array}{|c|c|c|c|}\hline 0 & 1 & 1 & 0 \\ \hline\end{array}$$

$$\begin{array}{|c|c|c|c|}\hline 0.3 & 0.5 & 0.7 & 0.2 \\ \hline\end{array} \xrightarrow[\text{(one potential result)}]{\text{AdaRound}} \begin{array}{|c|c|c|c|}\hline 0 & 0 & 1 & 0 \\ \hline\end{array}$$

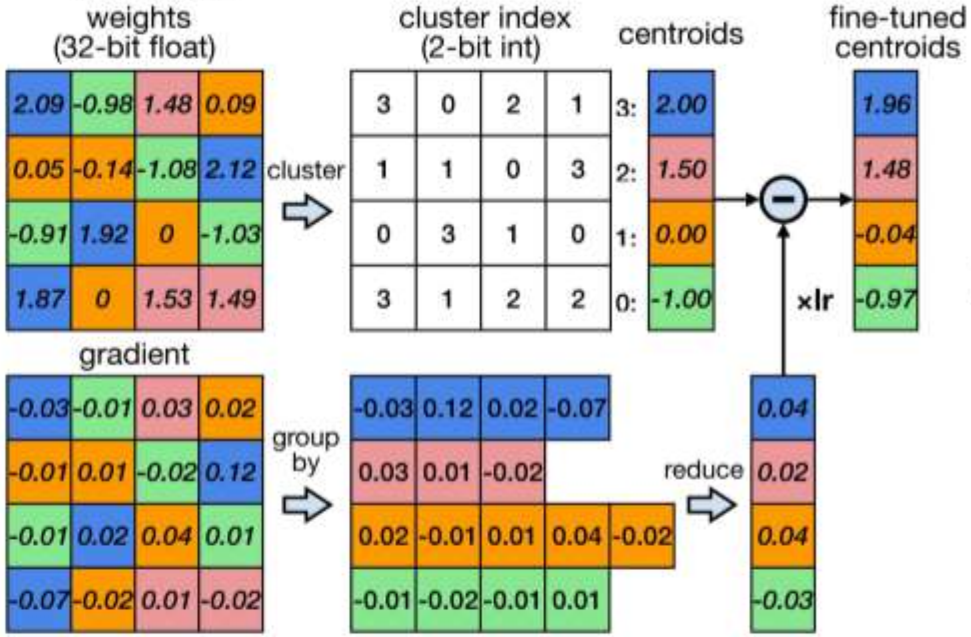
- What is optimal? Rounding that reconstructs the original <u>activation</u> the best, which may be very different
  - For weight quantization only
  - With short-term tuning, (almost) post-training quantization

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

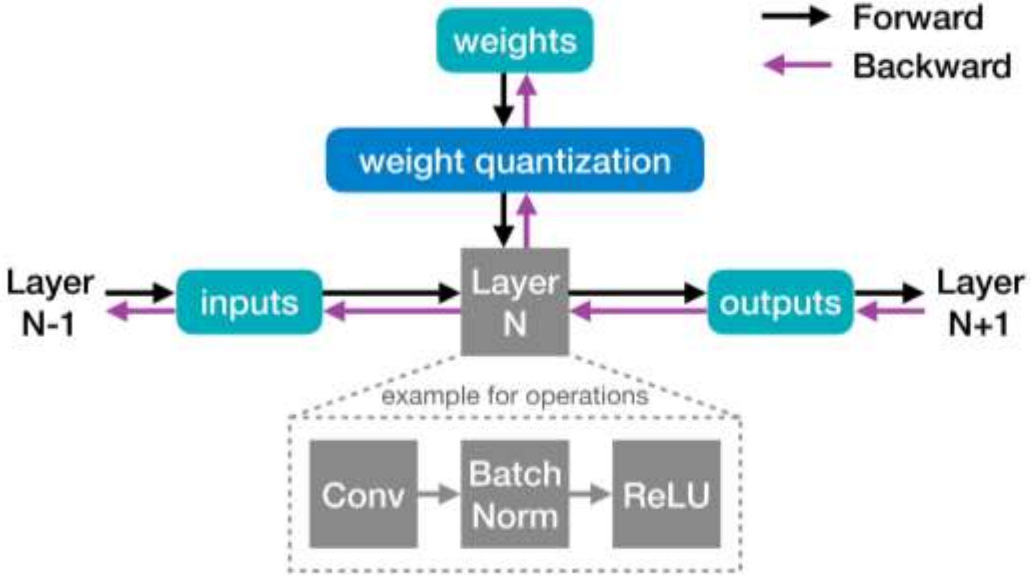
# Quantization-Aware Training

**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.



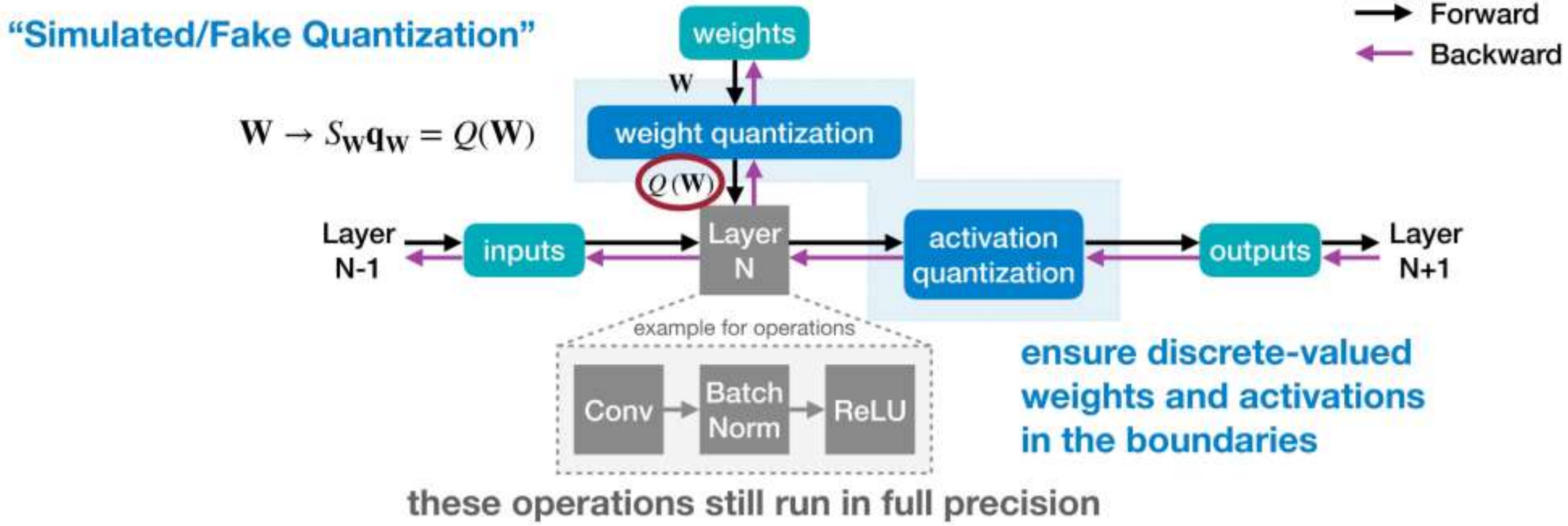

Deep Compression [Han *et al.*, ICLR 2016]



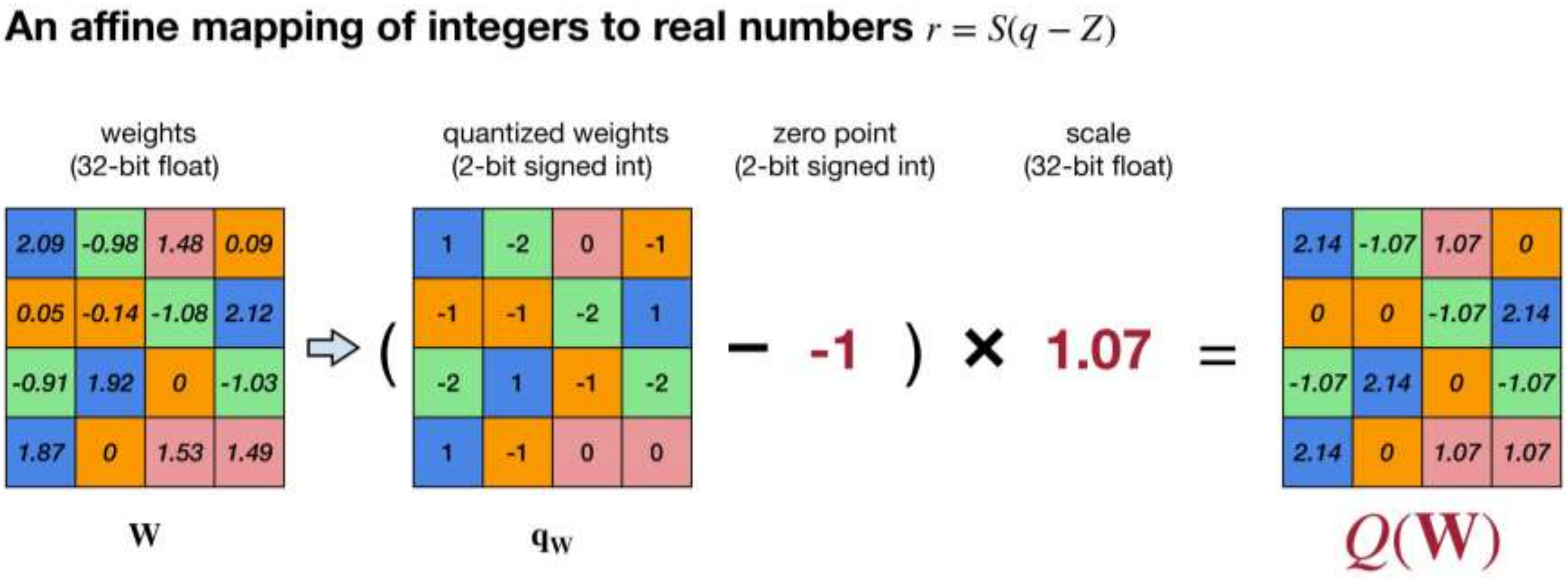
# Quantization-Aware Training

**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.



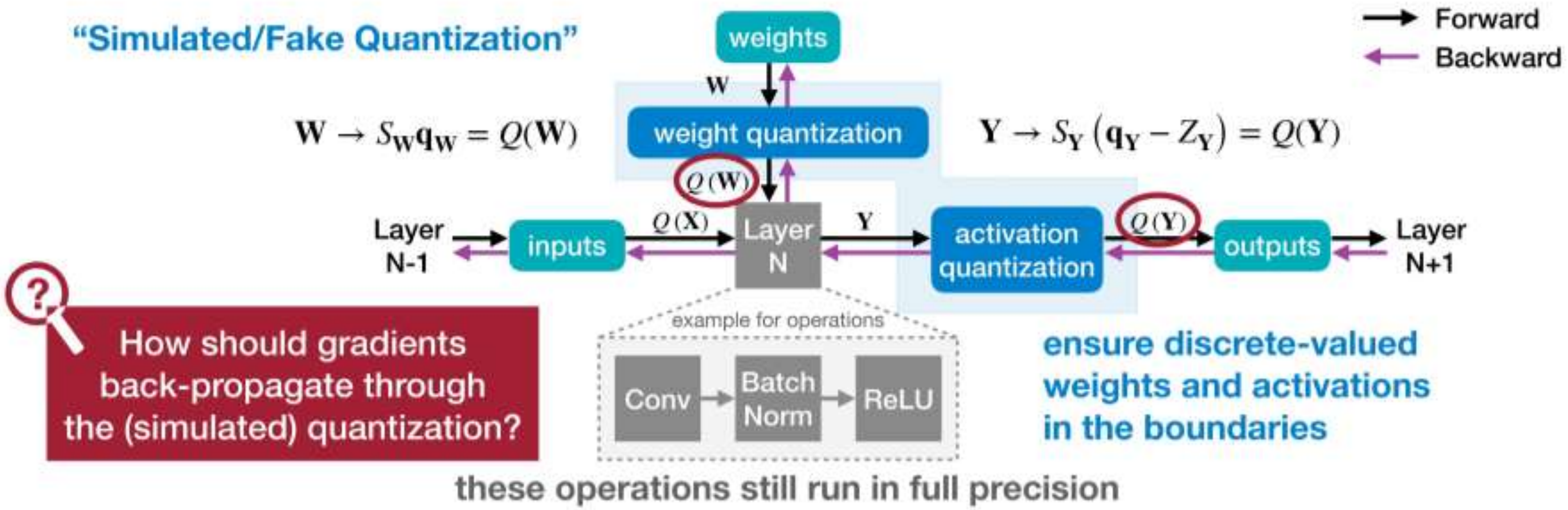
# Linear Quantization



# 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.



# Straight-Through Estimator (STE)

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

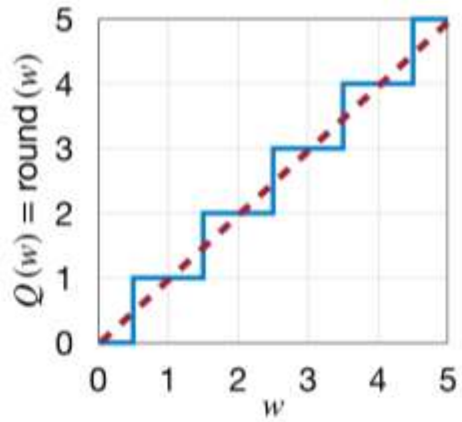
$$\frac{\partial Q(W)}{\partial W} = 0$$

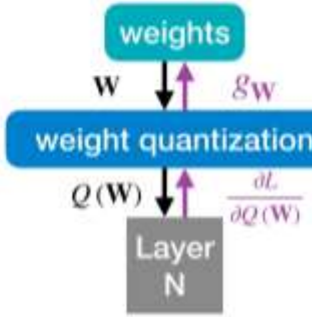
  - The neural network will learn nothing since gradients become 0 and the weights won't get updated.

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

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

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



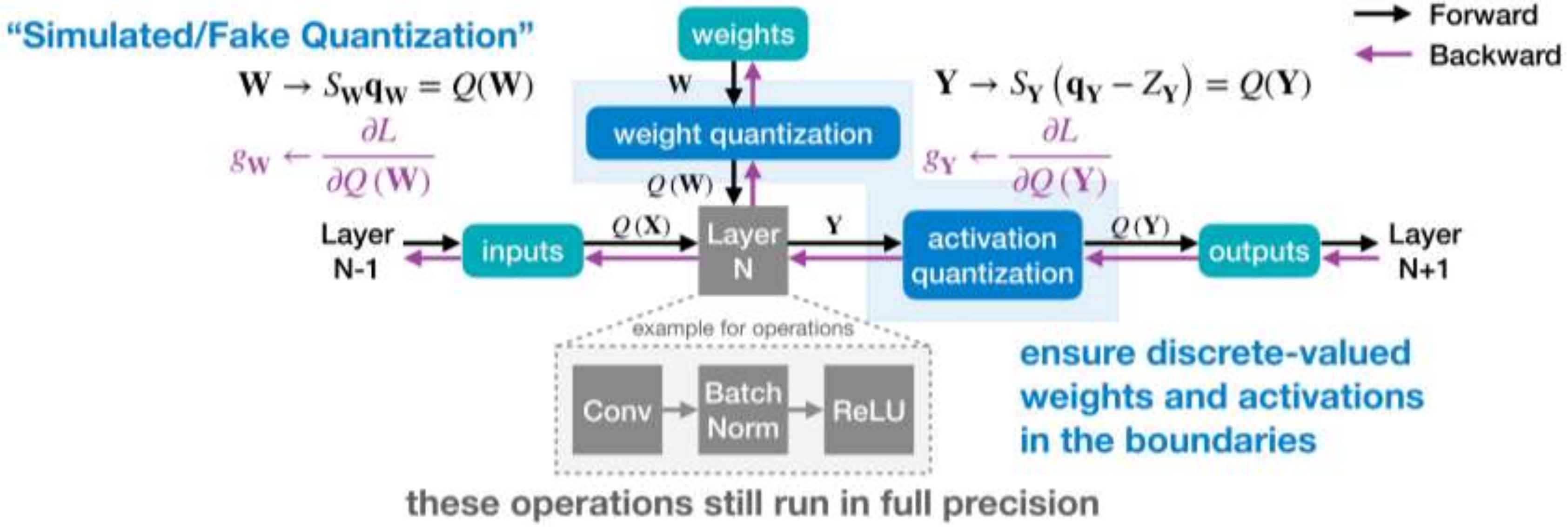




Neural Networks for Machine Learning [Hinton *et al.*, Coursera Video Lecture, 2012]
Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation [Bengio, arXiv 2013]

# 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.

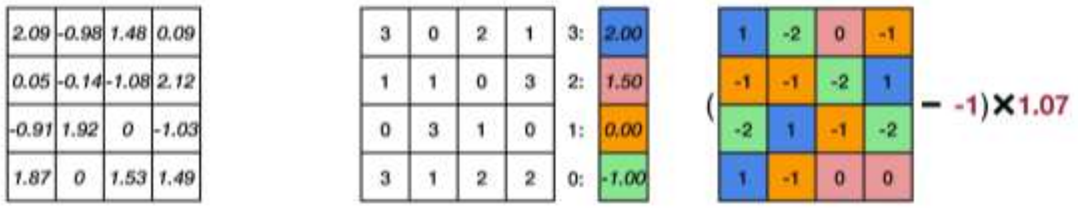


# INT8 Linear Quantization-Aware Training

| 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]

# Binary/Ternary Quantization




| | | 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 |

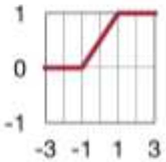
# Binarization

- **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)

$$\mathbf{W} = \begin{bmatrix} 2.09 & -0.98 & 1.48 & 0.09 \\ 0.05 & -0.14 & -1.08 & 2.12 \\ -0.91 & 1.92 & 0 & -1.03 \\ 1.87 & 0 & 1.53 & 1.49 \end{bmatrix}$$

binary weights (1-bit)

$$\mathbf{W}^{\mathbb{B}} = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad \|\mathbf{W} - \mathbf{W}^{\mathbb{B}}\|_F^2 = 9.28$$

$$\alpha\mathbf{W}^{\mathbb{B}} = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \times 1.05$$

scale (32-bit float)

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

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

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

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

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

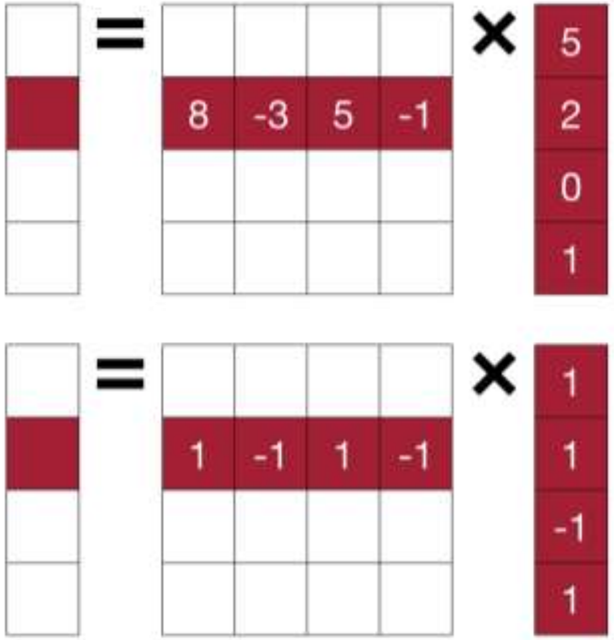
XNOR-Net: ImageNet Classification using Binary Convolutional Neural Networks [Rastegari *et al.*, ECCV 2016]

# XNOR and Popcount operation

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

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

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




| 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**

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

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

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

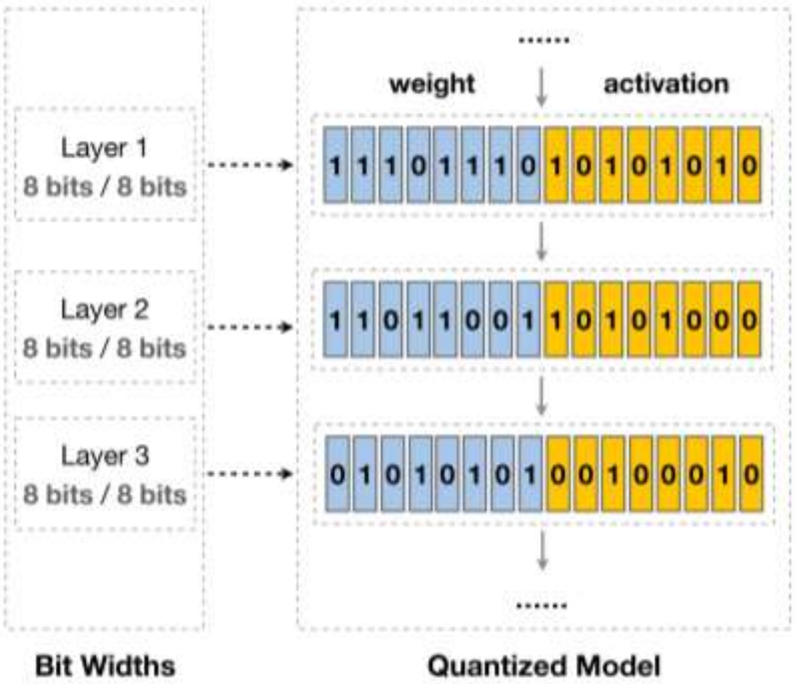
$$\times \quad \mathbf{1.5} = \frac{1}{11}\|\mathbf{W}_{\mathbf{W}^{\mathbb{T}} \neq 0}\|_1$$

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

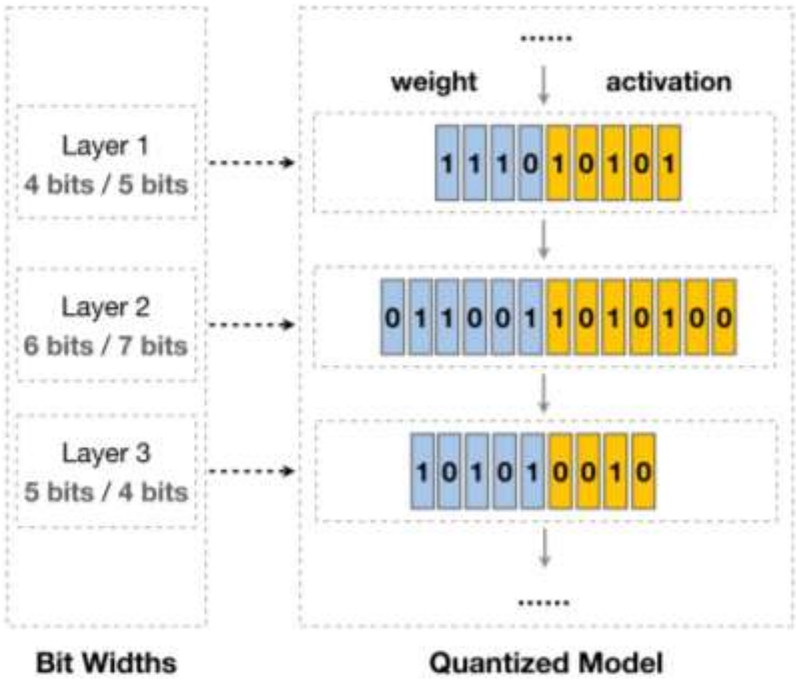
Ternary Weight Networks [Li *et al.*, Arxiv 2016]

# Mixed-Precision Quantization

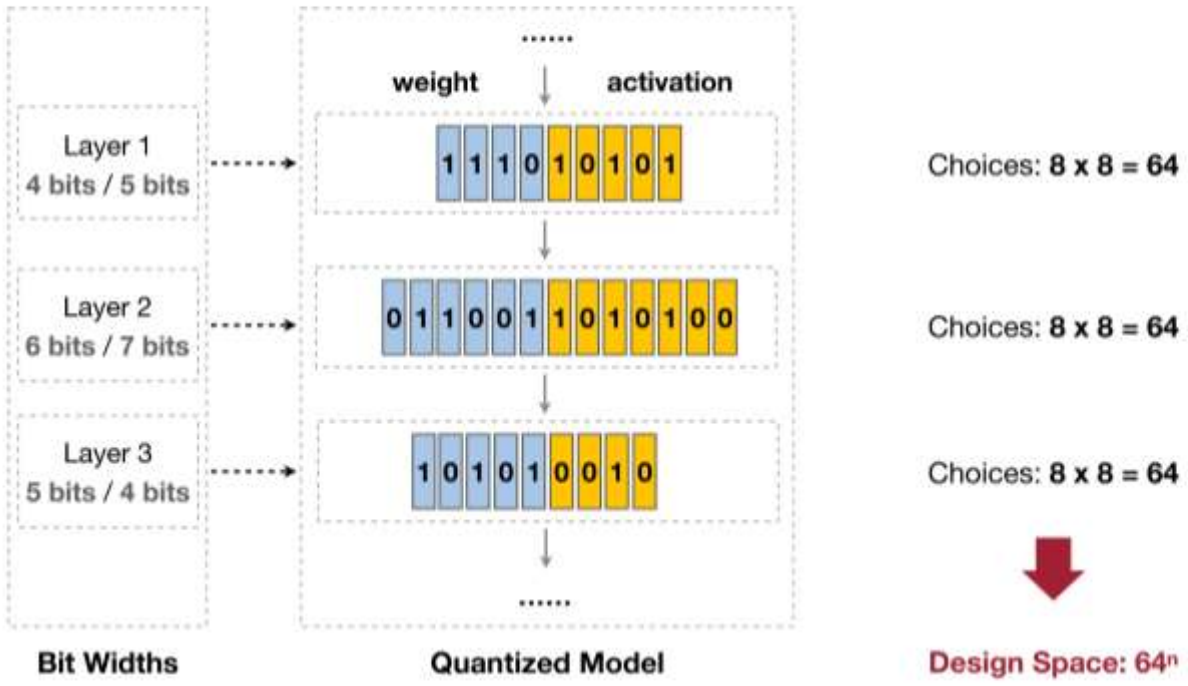
## Uniform Quantization



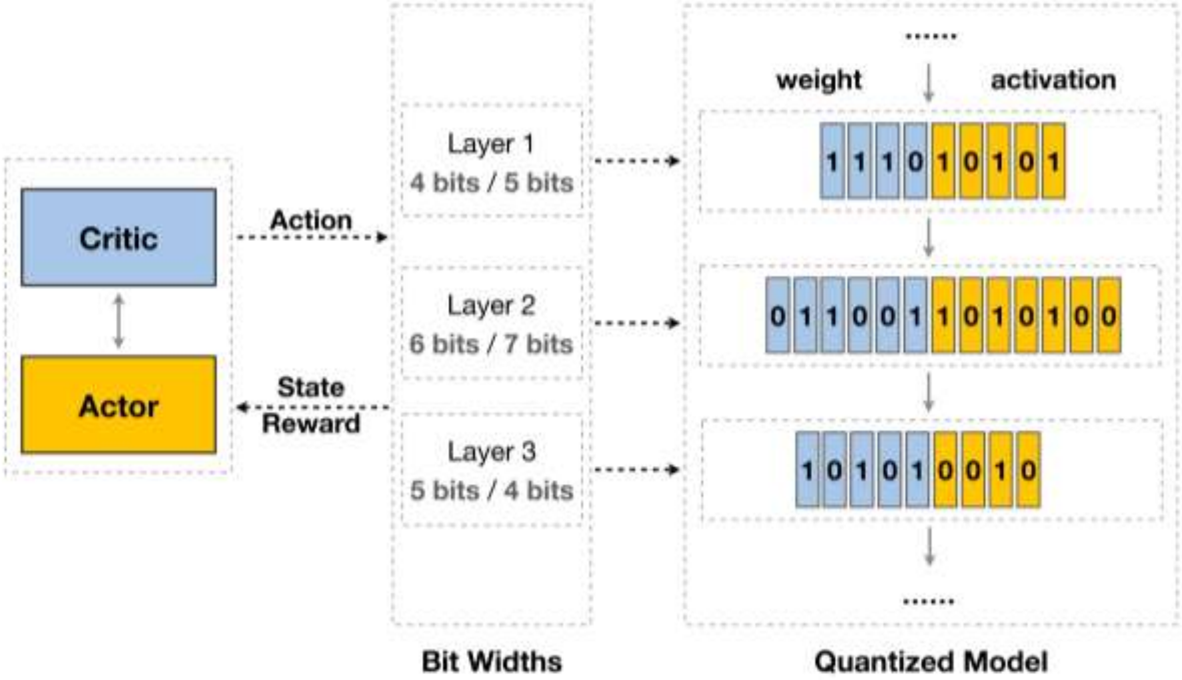

## Mixed-Precision Quantization



# Challenge: Huge Design Space

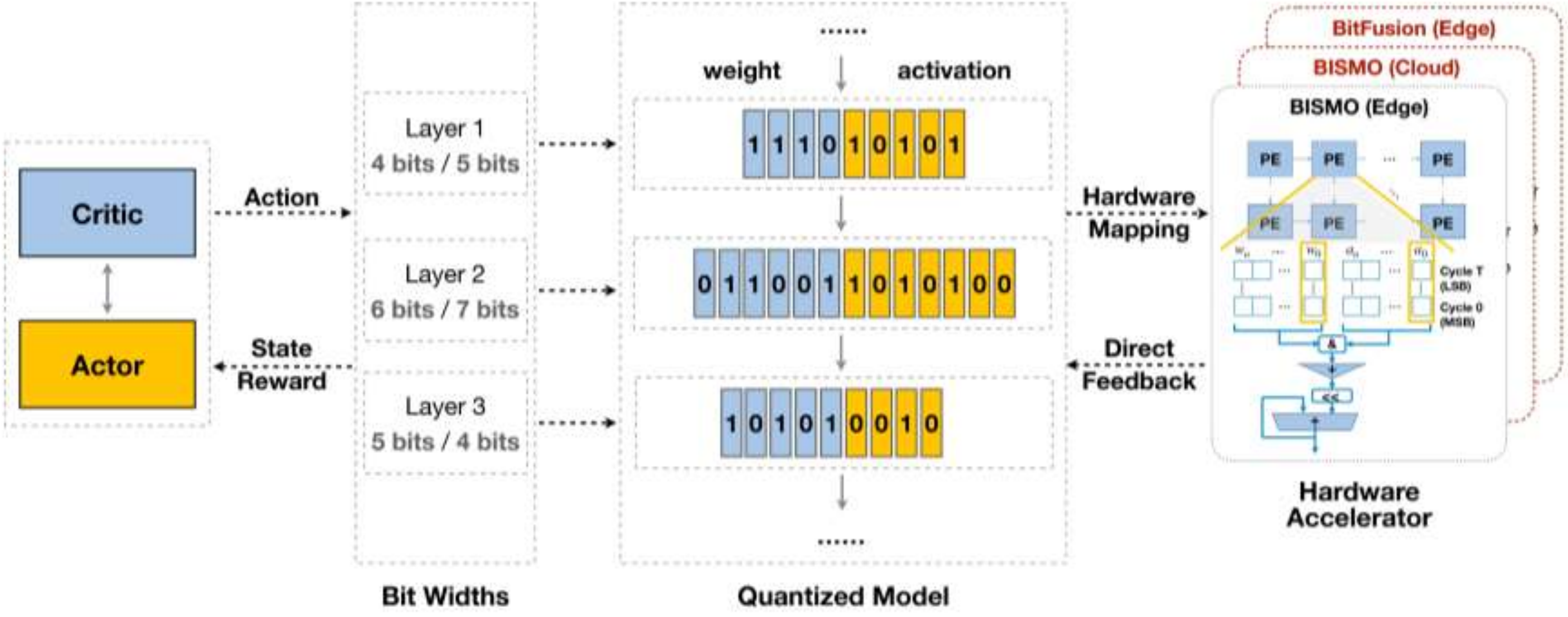


# Solution: Design Automation



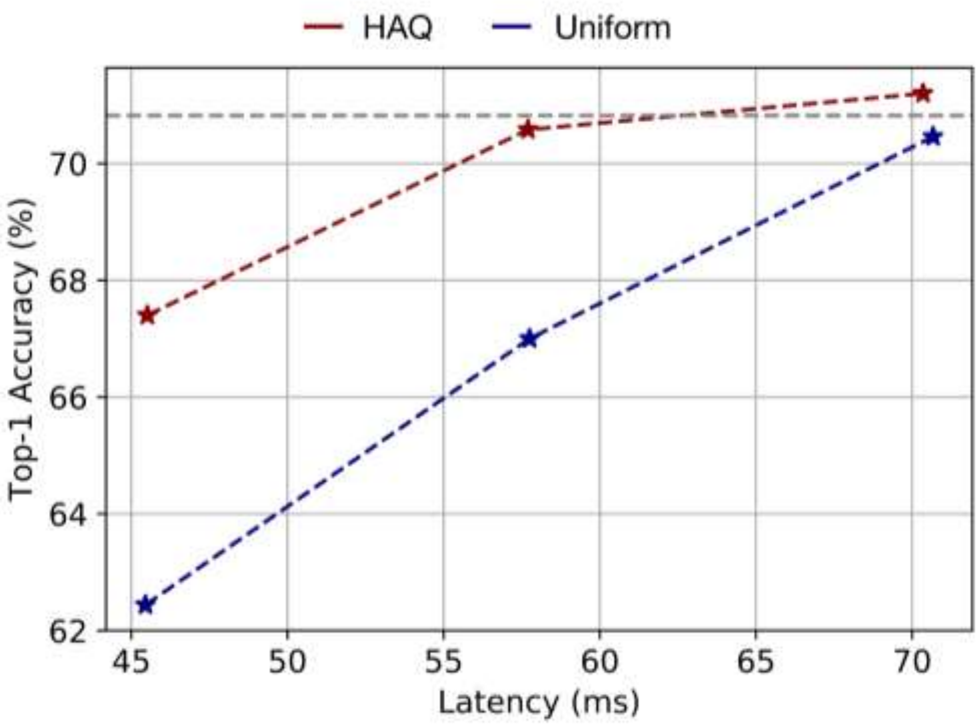

HAQ: Hardware-Aware Automated Quantization with Mixed Precision [Wang *et al.*, CVPR 2019]

# Solution: Design Automation




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

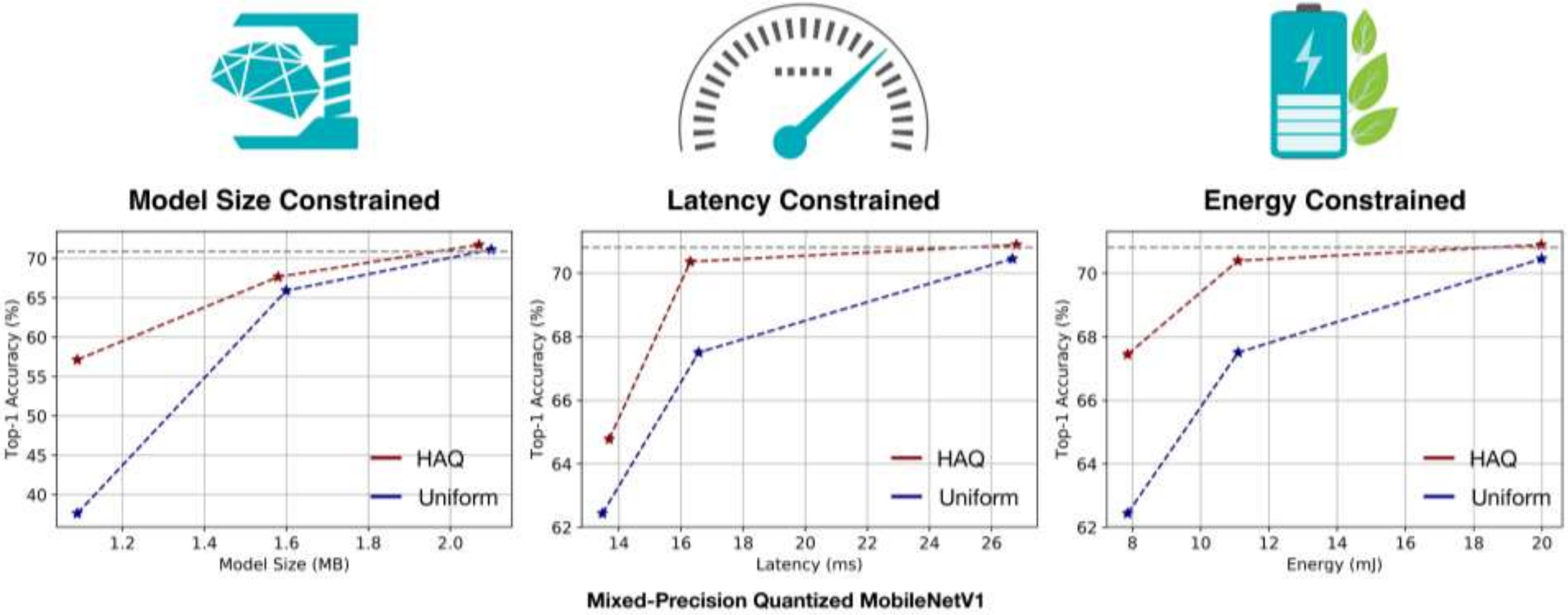
# HAQ Outperforms Uniform Quantization




**Mixed-Precision Quantized MobileNetV1**

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

# HAQ Supports Multiple Objectives




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