Foundations of Edge AI

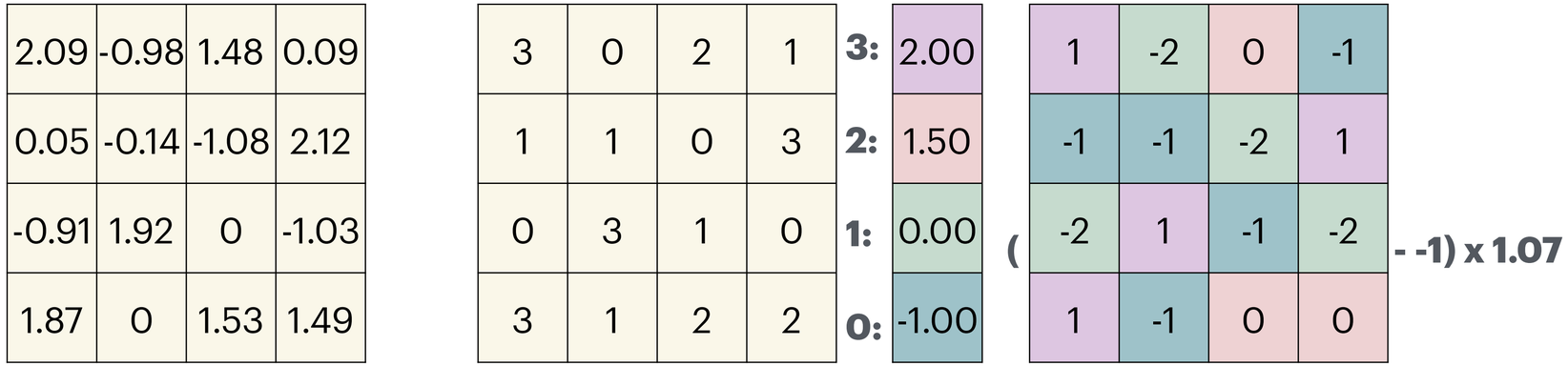
Lecture10

Neural Network Quantization (Cont'd)

Lanyu (Lori) Xu

Email: lxu@oakland.edu
Homepage: https://lori930.github.io/
Office: EC 524

Neural Network Quantization



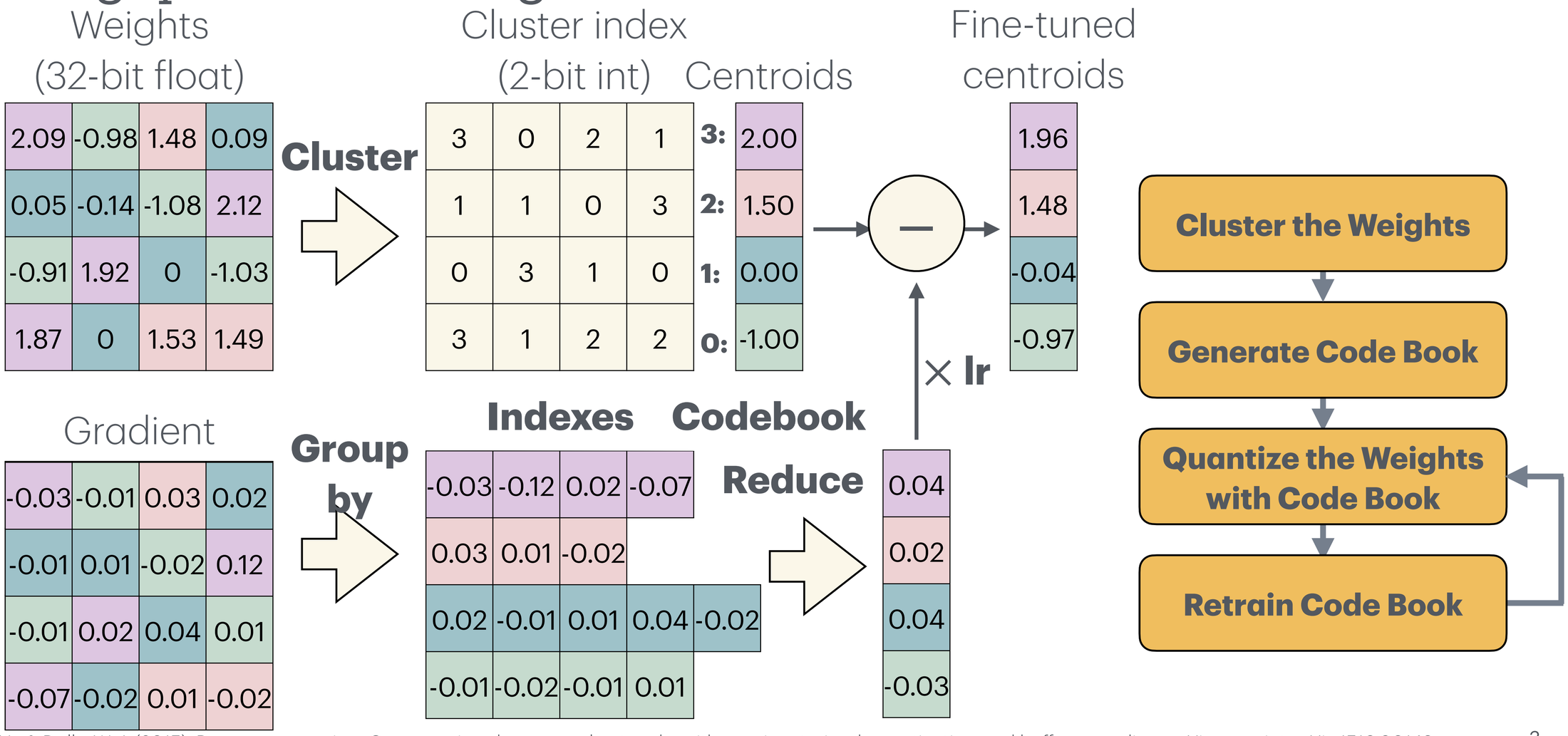
K-Means-based Quantization

Linear Quantization

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

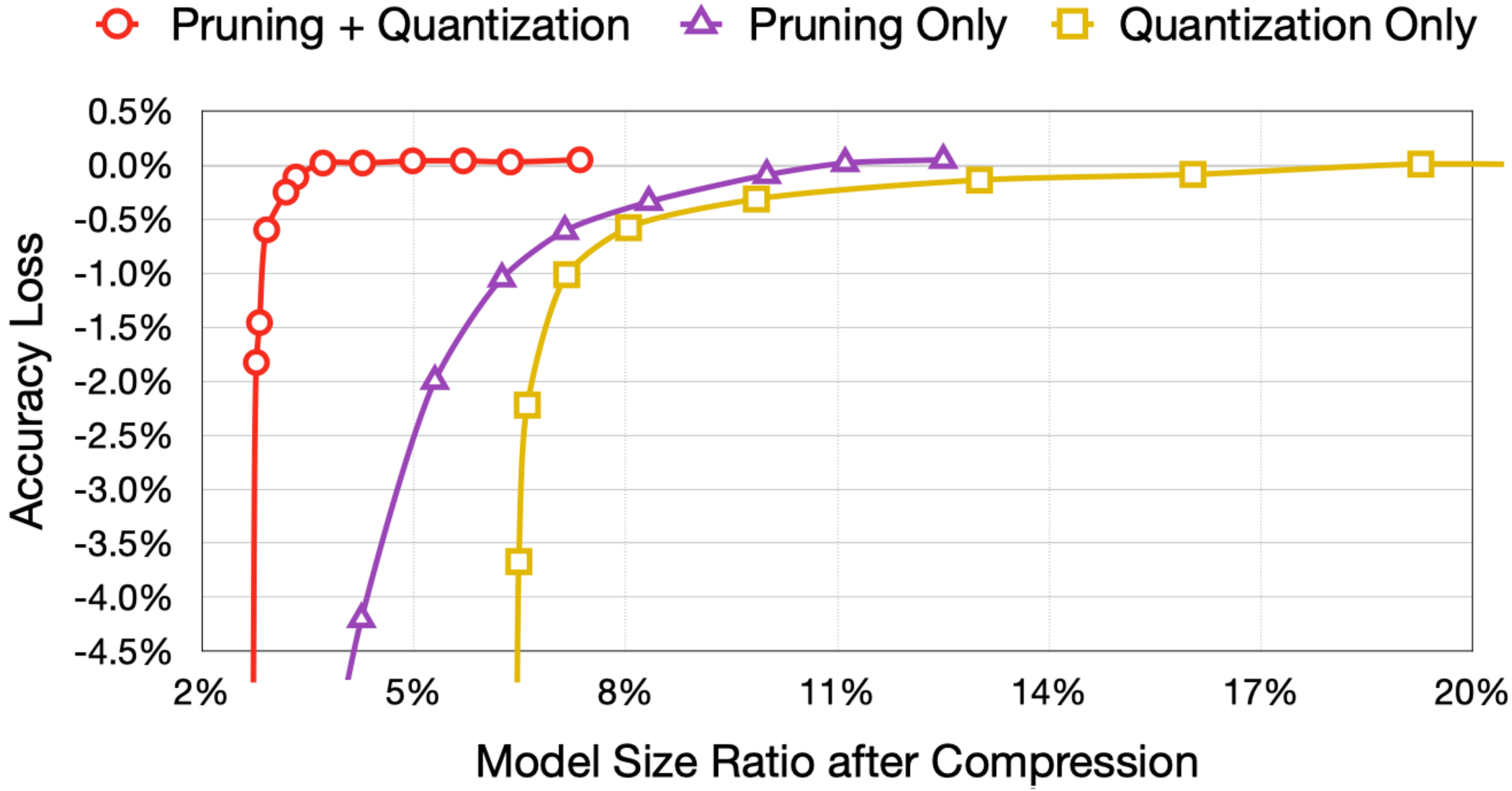
K-Means-based Weight Quantization

Fine-tuning quantizaed weights



Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

K-Means-based Weight Quantization



Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

Linear Quantization

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

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

Quantized weights (2-bit signed int)

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

Zero point (2-bit signed int)

Scale (32-bit float)

$$(\quad - \quad -1 \quad) \times \quad 1.07 =$$

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

Quantization parameters

$$r = (\quad q \quad - \quad Z \quad) \times \quad S$$

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

Allow real number $r = 0$ be exactly representable by a quantized integer Z

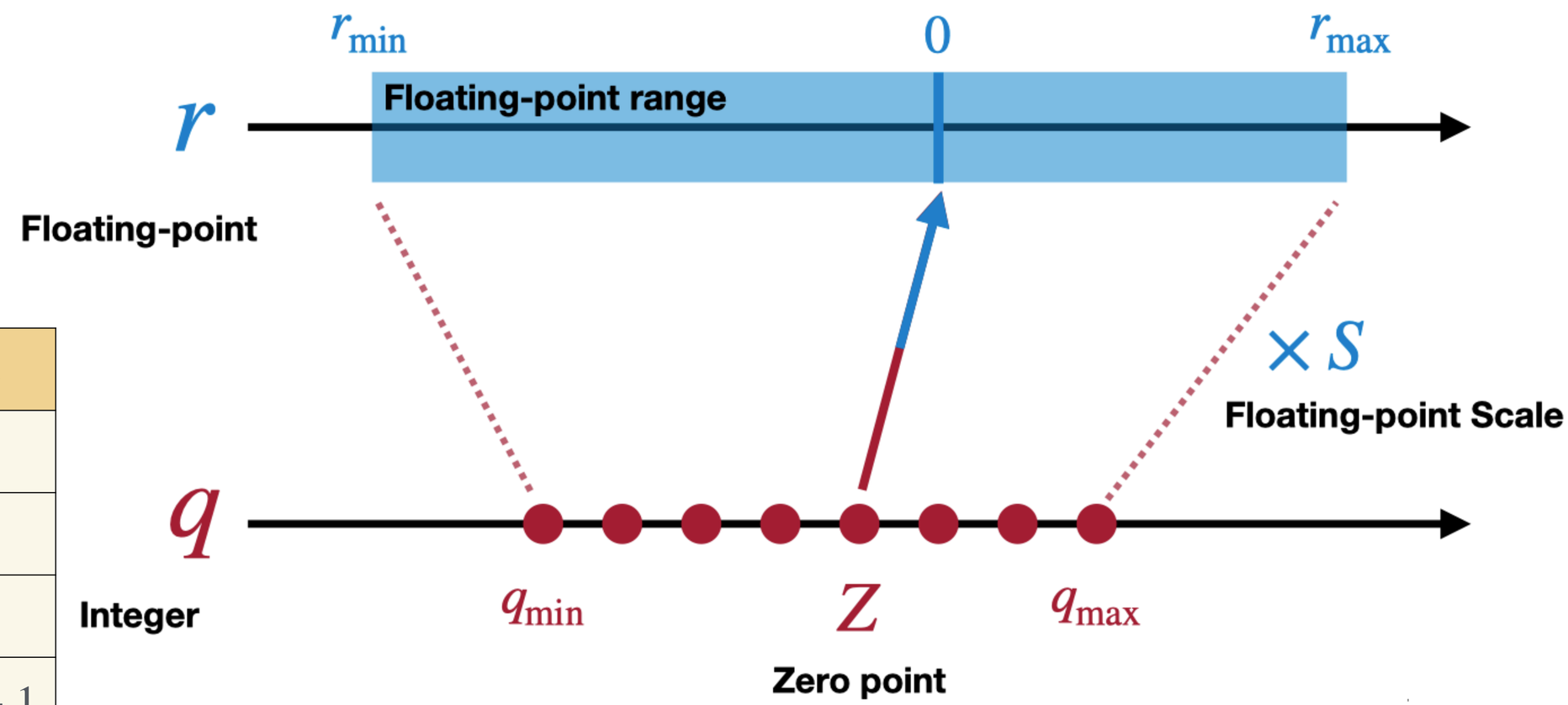
Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2704-2713).

Linear Quantization

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

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

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



Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2704-2713).

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 with bias (single layer MLP).

$$Y = WX + b$$

$$Z_W = 0$$

$$Z_b = 0,\ S_b = S_W S_X$$

$$q_{bias} = q_b - Z_X q_W$$

$$q_Y = \frac{S_W S_X}{S_Y}(q_W q_X + q_{bias}) + Z_Y$$

Rescale to N-bit int

N-bit int Mult
32-bit int Add

N-bit int add

Note: both q_b and 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

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

$$Z_W = 0$$

$$Z_b = 0,\ S_b = S_W S_X$$

$$q_{bias} = q_b - \textbf{Conv}(q_W, Z_X)$$

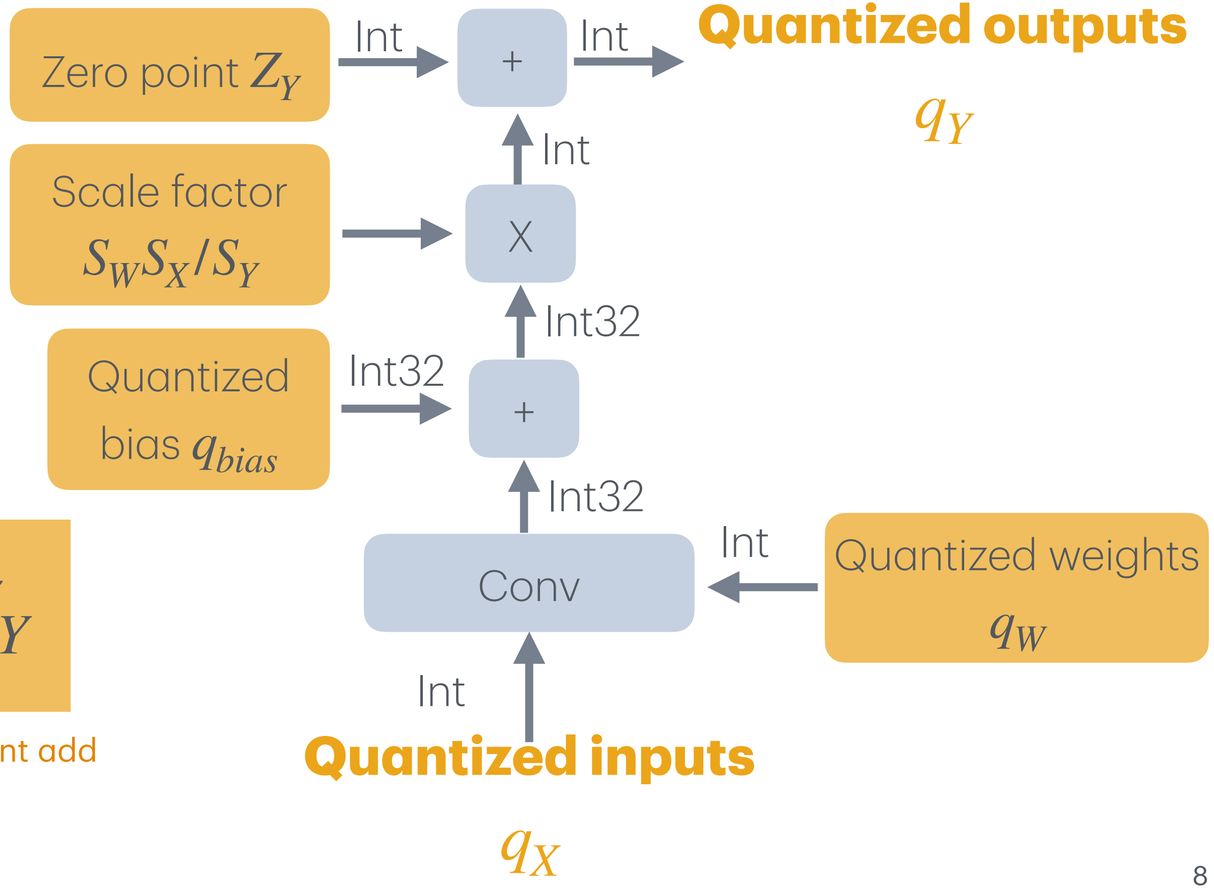
$$q_Y = \frac{S_W S_X}{S_Y}(\text{Conv}(q_W, q_X) + q_{bias}) + Z_Y$$

Rescale to N-bit int

N-bit int Mult
32-bit int Add

N-bit int add

Note: both q_b and q_{bias} are 32 bits



Lecture Plan

Today we will:

- Introduce **Post-Training Quantization (PTQ)**
  - Quantization granularity
  - Dynamic Range clipping
  - Rounding
- Introduce **Quantization-Aware Training (QAT)**
- Introduce binary and ternary quantization
- Introduce automatic mixed-precision quantization
- Lab 3 is out, mid-term survey is out

Post-Training Quantization

Quantize a floating-point neural network model,
to reduce the model size, improve inference speed, and lower power consumption

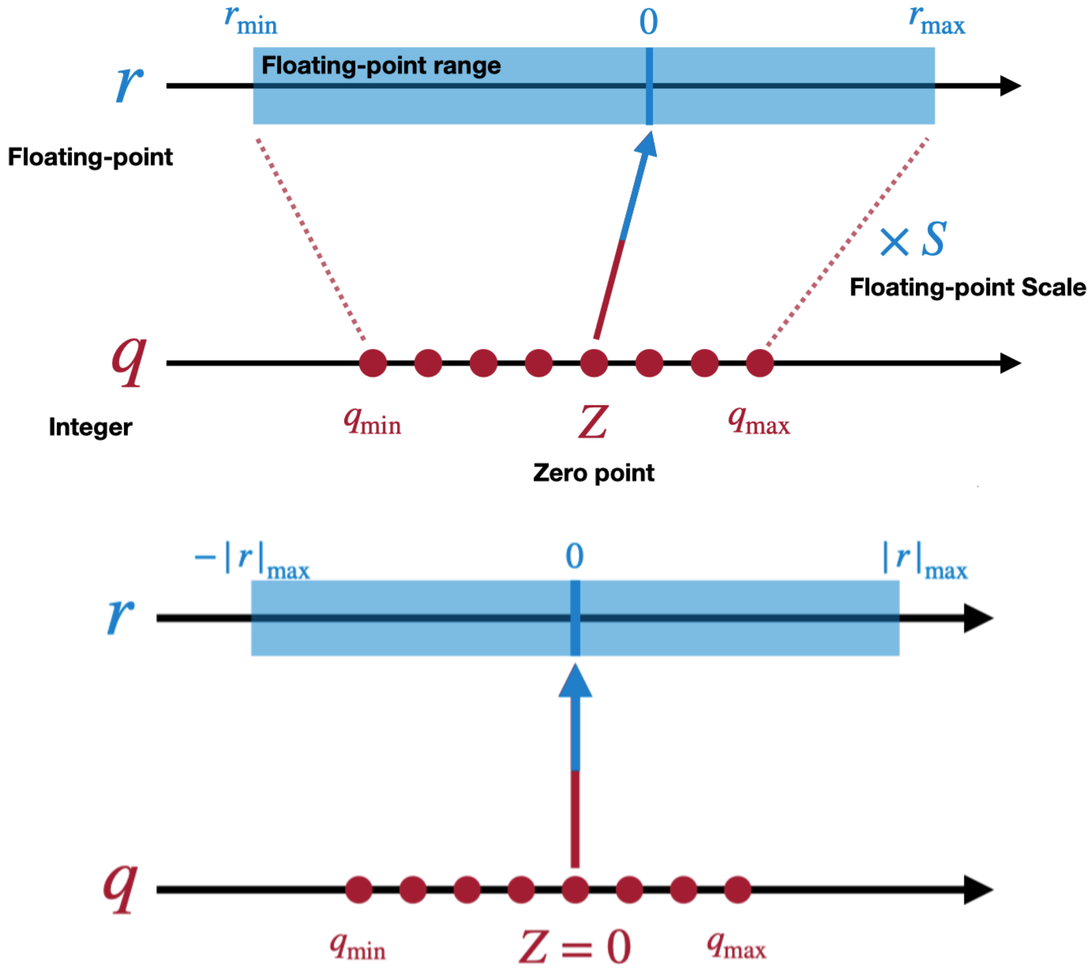
Linear Quantization is an affine mapping of integers to real numbers

$$r = S(q - Z)$$

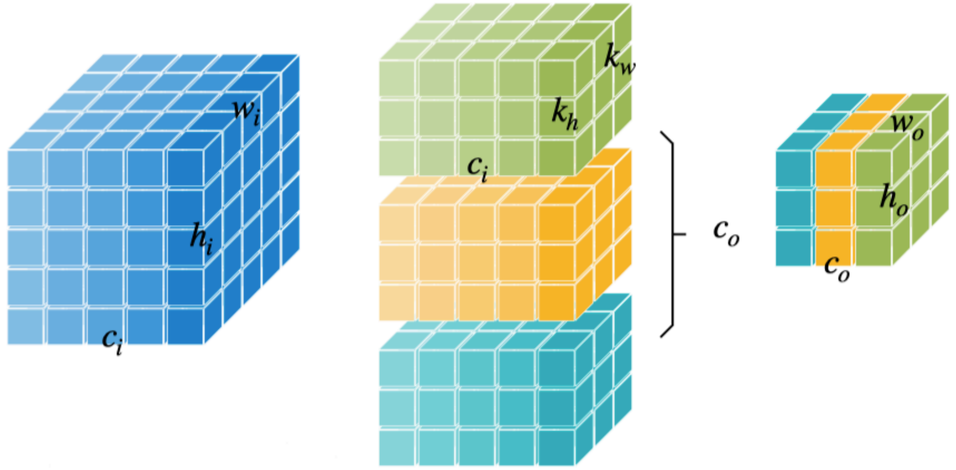
Post-Training Quantization

How to get the optimal linear quantization parameters S and Z?

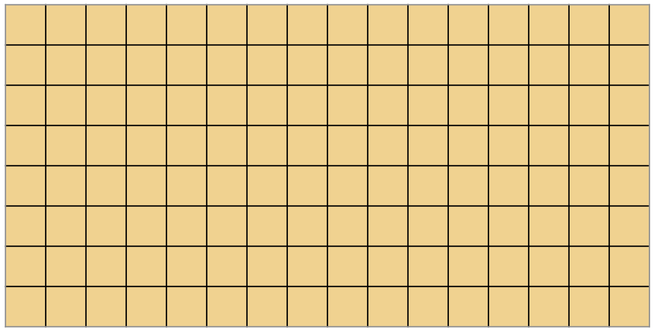
- Recall: Pruning can be performed at different granularities, from structured to non-structured.
- So can PTQ!
  - Quantization can be performed at different granularities, from structured to non-structured.



Quantization Granularity

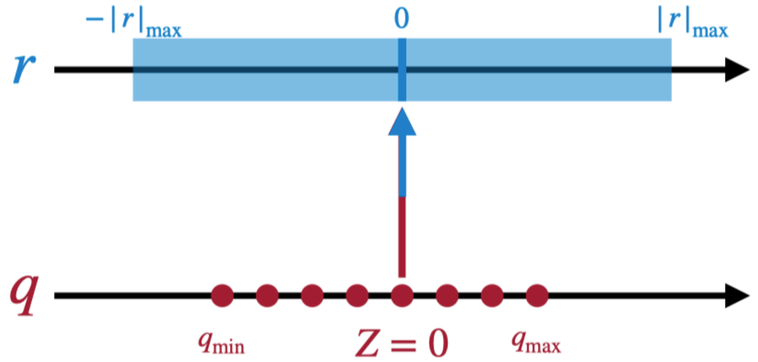


Per-Tensor Quantization



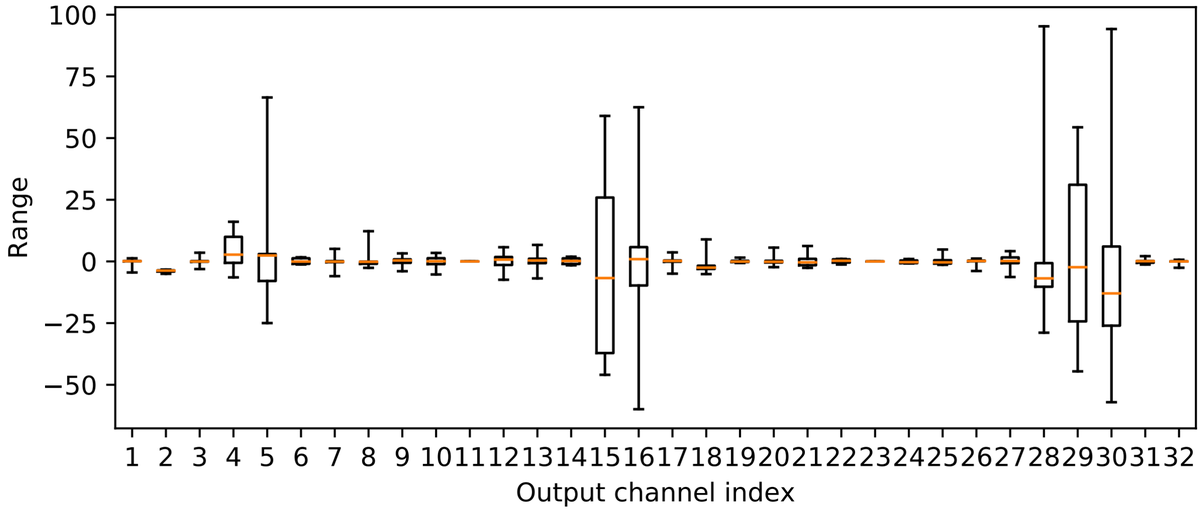
One scaling factor to scale the whole tensor

Per-Tensor Quantization



Symmetric linear quantization on weights

- A single scale S for the whole tensor
- $|r|_{max} = |W|_{max}$
- Works well for large models, but accuracy drop for small models
- Common failure because of **outlier weight** 👇



Large differences (more than 100x) in ranges of weights for different output channels.

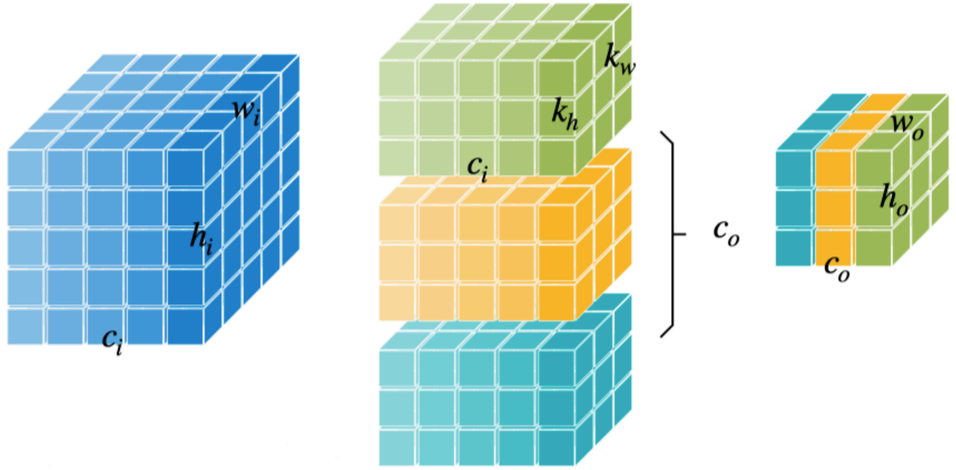
How to address outlier weight issue?

Per-Channel Quantization

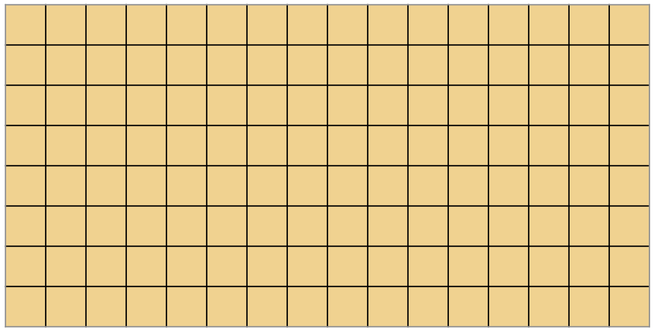
Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2704-2713).

Nagel, M., Baalen, M. V., Blankevoort, T., & Welling, M. (2019). Data-free quantization through weight equalization and bias correction. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1325-1334).

Quantization Granularity

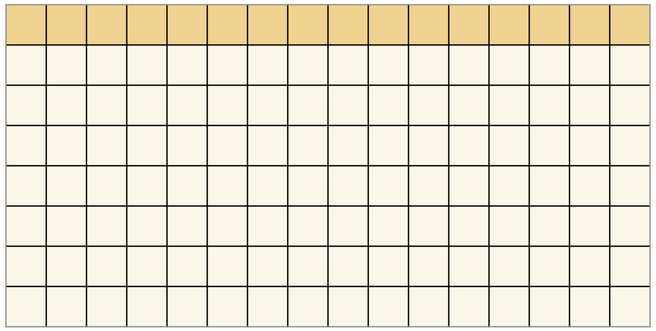


Per-Tensor Quantization



One scaling factor to scale the whole tensor

Per-Channel Quantization



One scaling factor to scale one channel

Per-Channel Weight Quantization

$$r = S(q - Z),\ S = \frac{|r|_{max}}{q_{max}}$$

Per-Tensor Quantization

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

$$q_{max} = 2^{2-1} - 1 = 1$$

$$S = \frac{2.12}{1} = 2.12$$

Round $q = \frac{r}{S}$

1	0	1	0	2.12	0	2.12	0
0	0	-1	1	0	0	-2.12	2.12
0	1	0	0	0	2.12	0	0
1	0	1	1	2.12	0	2.12	2.12

Quantized | Reconstructed

Input channel

Output channel

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

$$\| W - Sq_W \|_F = 2.28 > \| W - Sq_W \|_F = 2.08$$

$\| \cdot \|_F$ Frobenius norm: the square root of the sum of the squares of the elements of the matrix (see next slide)

Per-Channel Quantization

Channel 1: $|r|_{max} = 2.09,\ S_0 = 2.09$

Channel 2: $|r|_{max} = 2.12,\ S_1 = 2.12$

Channel 3: $|r|_{max} = 1.92,\ S_2 = 1.92$

Channel 4: $|r|_{max} = 1.87,\ S_3 = 1.87$

1	0	1	0	2.09	0	2.09	0
0	0	-1	1	0	0	-2.12	2.12
0	1	0	-1	0	1.92	0	-1.92
1	0	1	1	1.87	0	1.87	1.87

Quantized | Reconstructed

Compute the Frobenius Norm $\| \cdot \|_F$

A common way to measure the difference between two matrices

- The square root of the sum of the squared differences between corresponding elements in the two matrices: $\| A \|_F = \sqrt{\sum_{i,j} |A_{i,j}|^2}$

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

Original

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

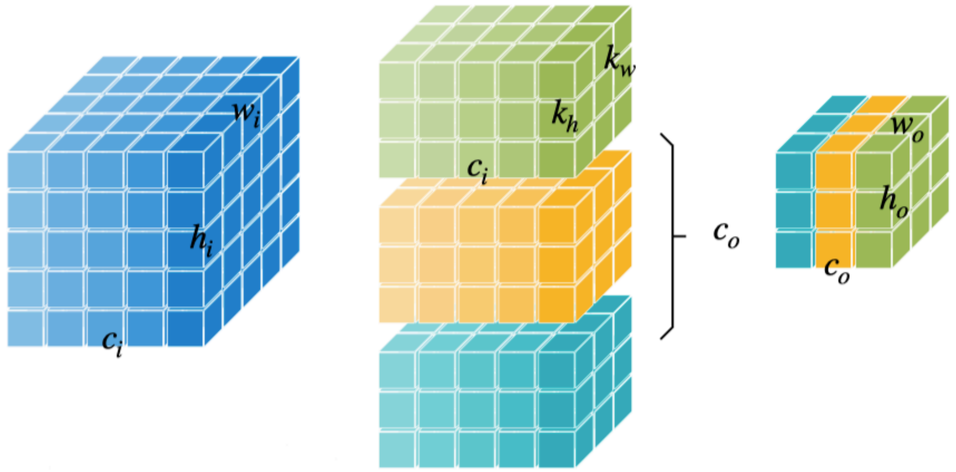
Reconstructed

$$W - Sq_W = \begin{vmatrix} -0.03 & -0.98 & -0.64 & 0.09 \\ 0.05 & -0.14 & 1.04 & 0 \\ -0.91 & -0.2 & 0 & -1.03 \\ -0.25 & 0 & -0.59 & -0.63 \end{vmatrix}$$

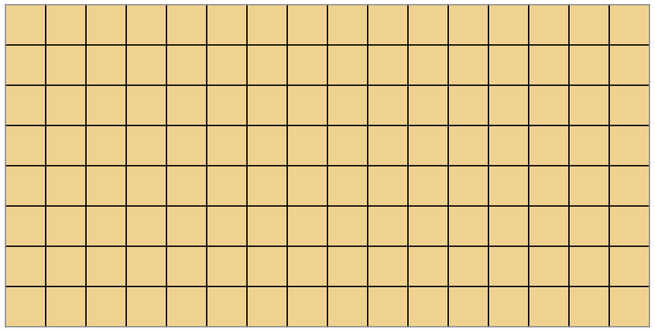
$$\| W - Sq_W \|_F = \sqrt{(-0.03)^2 + (-0.98)^2 + (-0.64)^2 + (0.09)^2 + (0.05)^2 + (-0.14)^2 + (1.04)^2 + (0)^2 + (-0.91)^2 + (-0.2)^2 + (0)^2 + (-1.03)^2 + (-0.25)^2 + (0)^2 + (-0.59)^2 + (-0.63)^2}$$

$$= \sqrt{5.2182} \approx 2.28$$

Quantization Granularity

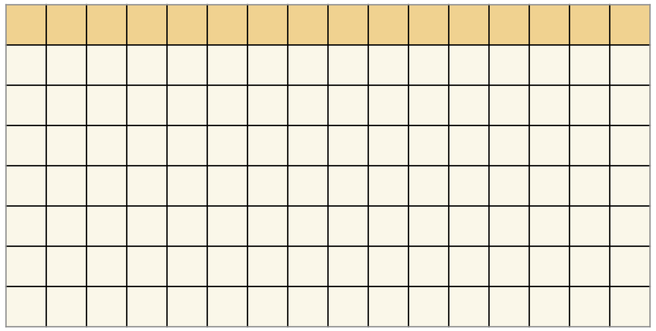


Per-Tensor Quantization



One scaling factor to scale the whole tensor

Per-Channel Quantization

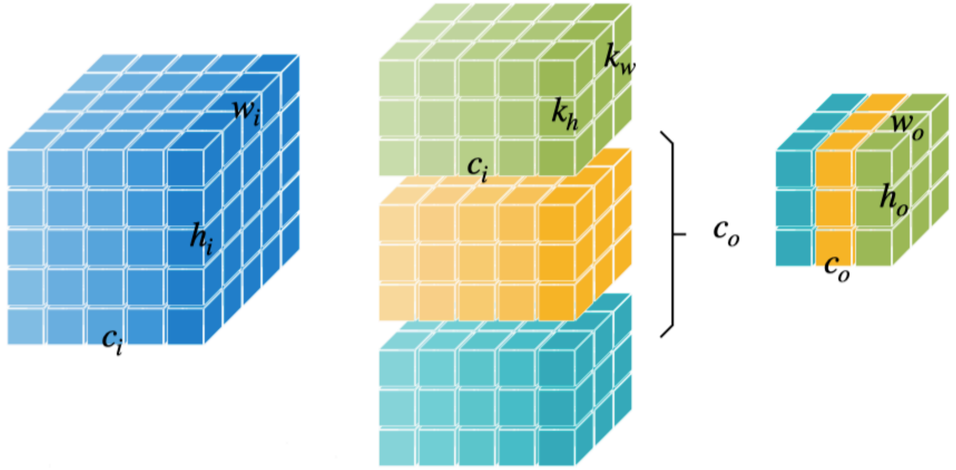


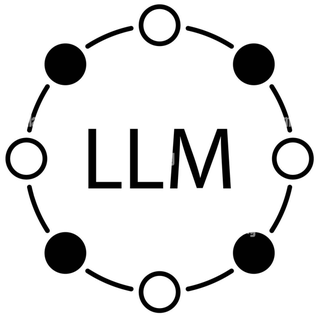
One scaling factor to scale one channel

→ More fine-grained scaling factor,
Better accurate representation,
Less quantization error.

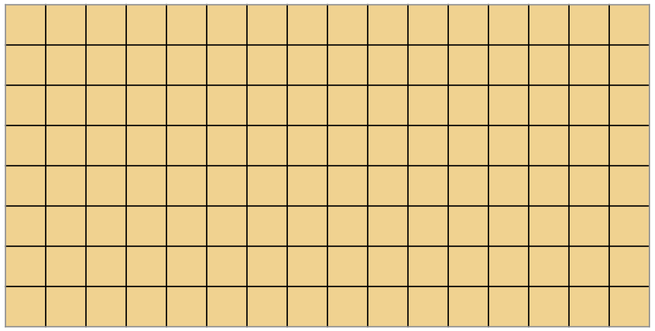
Trade-off?
More storage request.

Quantization Granularity



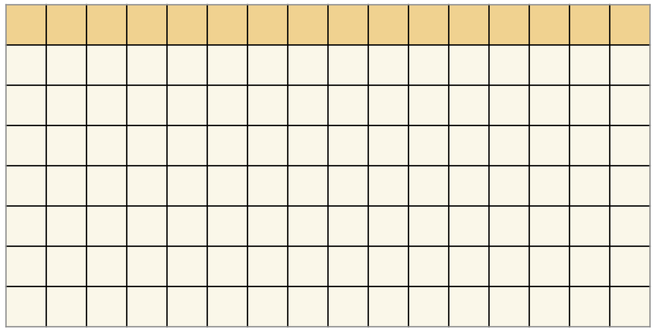


Per-Tensor Quantization



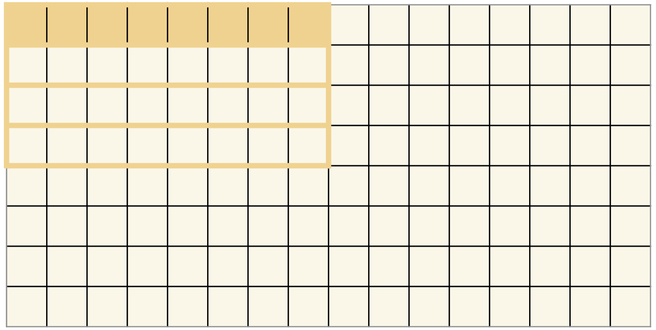
One scaling factor to scale the whole tensor

Per-Channel Quantization



One scaling factor to scale one channel

Group Quantization

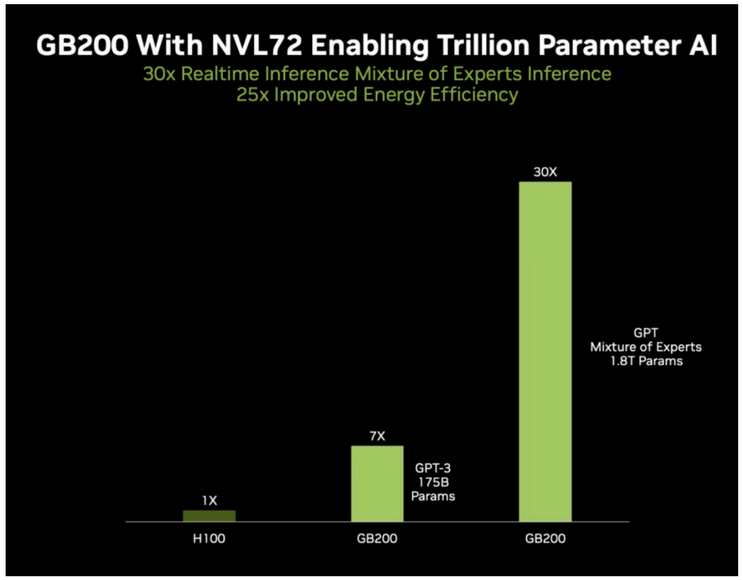


Group Quantization

Achieve a balance between quantization accuracy and hardware efficiency

- Blackwell GPUs support "**micro-tensor scaling**" to optimize accuracy for FP4 AI

- FP4 tensor core provides 2x higher theoretical throughputs than FP8/FP16/INT8 tensor core

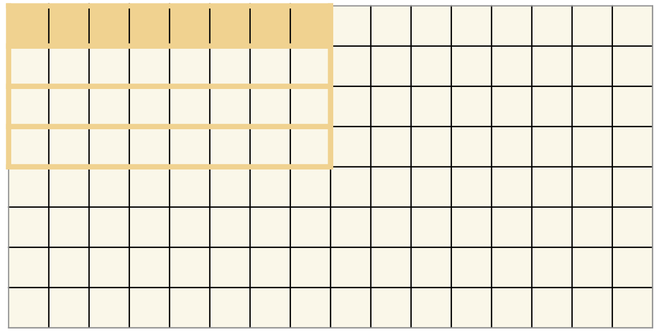


	HGX B200	HGX B100
Blackwell GPUs	8	8
FP4 Tensor Core	144 PetaFLOPS	112 PetaFLOPS
FP8/FP6/INT8	72 PetaFLOPS	56 PetaFLOPS
Fast Memory	Up to 1.5 TB	Up to 1.5TB
Aggregate Memory Bandwidth	Up to 64 TB/s	Up to 64 TB/s
Aggregate NVLink Bandwidth	14.4 TB/s	14.4 TB/s
Per Blackwell GPU Specifications		
FP4 Tensor Core	18 petaFLOPS	14 petaFLOPS
FP8/FP6 Tensor Core	9 petaFLOPS	7 petaFLOPS
INT8 Tensor Core	9 petaOPS	7 petaOPs
FP16/BF16 Tensor Core	4.5 petaFLOPS	3.5 petaFLOPS
TF32 Tensor Core	2.2 petaFLOPS	1.8 petaFLOPS
FP32	80 teraFLOPS	60 teraFLOPS
FP64 Tensor Core	40 teraFLOPS	30 teraFLOPS
FP64	40 teraFLOPS	30 teraFLOPS
GPU memory | Bandwidth	Up to 192 GB HBM3e | Up to 8 TB/s	
Max thermal design power (TDP)	1,000W	700W
Interconnect	NVLink: 1.8TB/s PCIe Gen6: 256GB/s	NVLink: 1.8TB/s PCIe Gen6: 256GB/s
Server options	NVIDIA HGX B200 partner and NVIDIA-Certified Systems with 8 GPUs	NVIDIA HGX B100 partner and NVIDIA-Certified Systems with 8 GPUs

System Specifications for HGX B200 and HGX B100

Blackwell Architecture for Generative AI | NVIDIA

Group Quantization

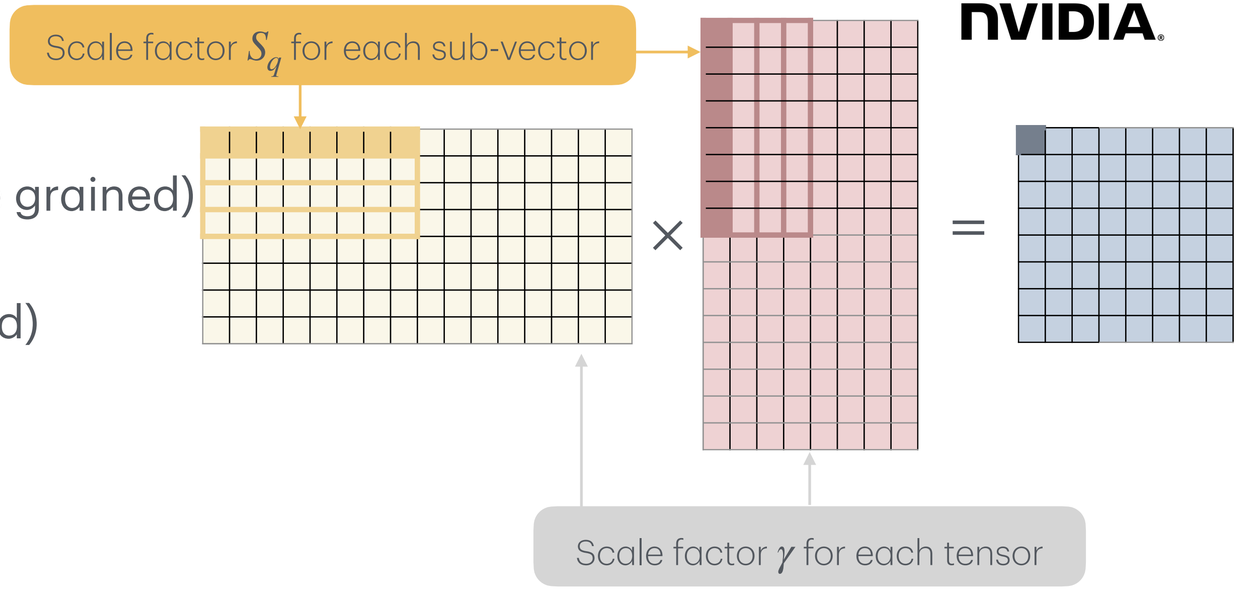


- Per-vector quantization
  - VS-Quant
- Multi-level scaling scheme
  - Shared Micro-exponent (MX) data type

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 per-tensor scale factor (coarse grained)
  - S_q is an integer per-vector scale factor (fine grained)
- Achieve a balance between accuracy and hardware efficiency by
  - Integer scale factor at finer granularity → less expensive
  - Floating-point scale factors at coarser granularity → more expensive

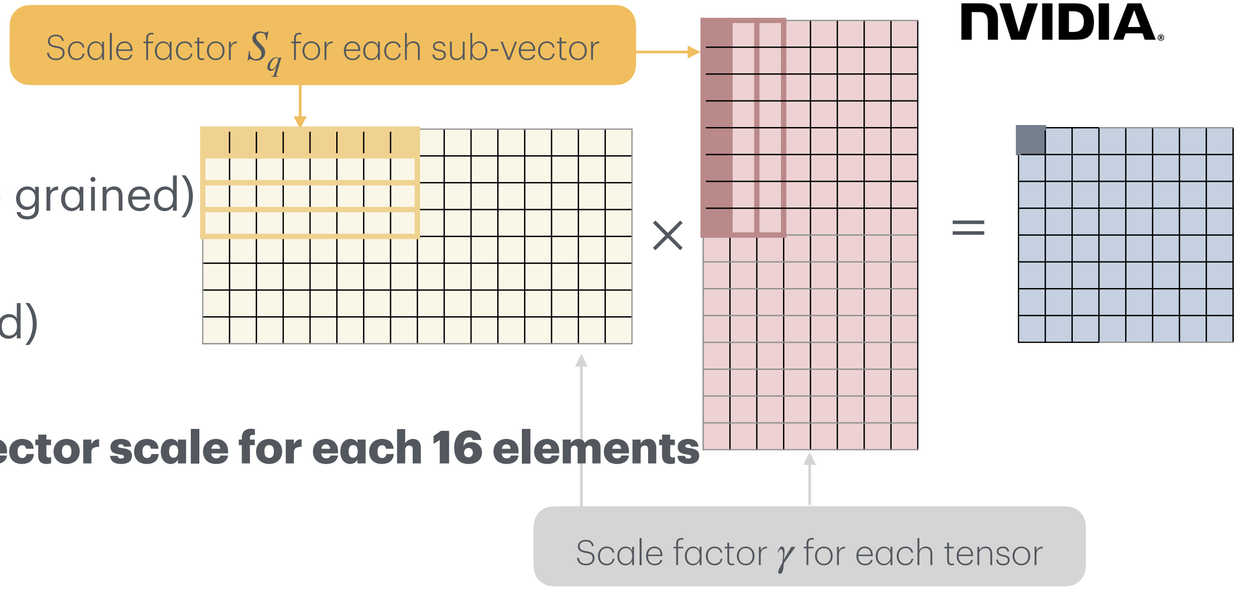


🤔 **What is the effective bit width of this two-level scaling?**

Dai, S., Venkatesan, R., Ren, M., Zimmer, B., Dally, W., & Khailany, B. (2021). Vs-quant: Per-vector scaled quantization for accurate low-precision neural network inference. Proceedings of Machine Learning and Systems, 3, 873-884.

VS-Quant: Per-Vector Scaled Quantization

Memory overhead of the hierarchical scaling factor

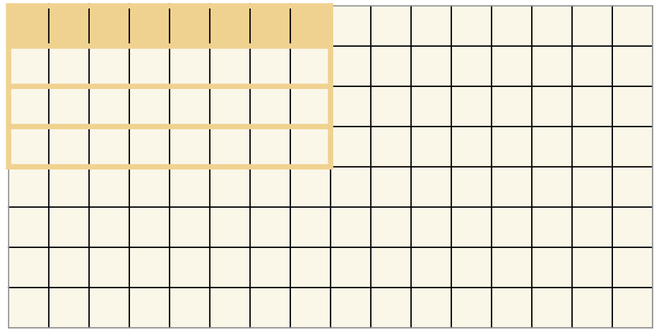


- $r = S(q - Z) \rightarrow r = \gamma \cdot S_q(q - Z)$
  - γ is a floating-point per-tensor scale factor (coarse grained)
  - S_q is an integer per-vector scale factor (fine grained)
- **Example: Given 4-bit quantization with 4-bit per-vector scale for each 16 elements**
  - 4-bit quantization → Each element has 4 bits
  - Every 16 elements share a 4-bit scaling factor → Equivaltently, each element has $\frac{4}{16}$ bit
  - The effective bit width is $4 + \frac{4}{16} = 4.25$ bits

Dai, S., Venkatesan, R., Ren, M., Zimmer, B., Dally, W., & Khailany, B. (2021). Vs-quant: Per-vector scaled quantization for accurate low-precision neural network inference. Proceedings of Machine Learning and Systems, 3, 873-884.

Group Quantization

Multi-level scaling scheme

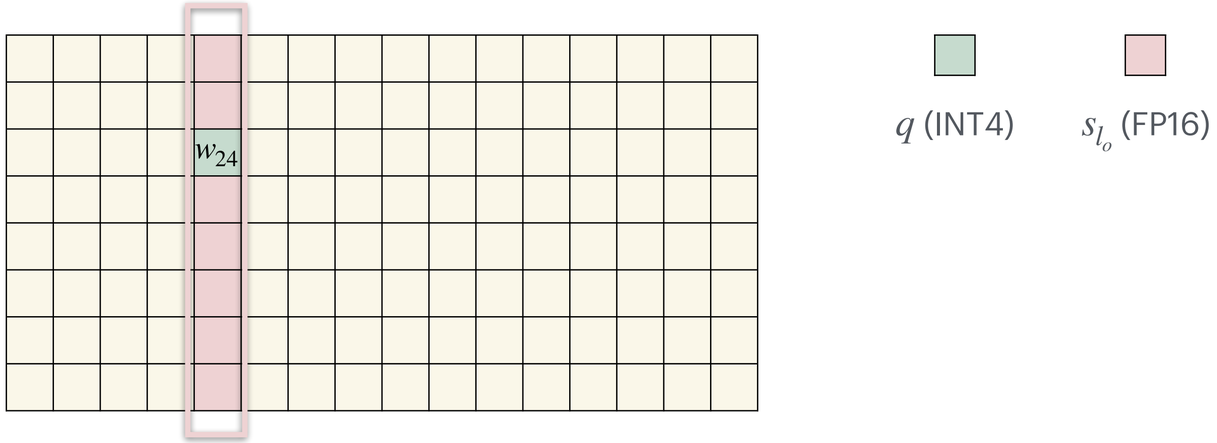


- $r = S(q - Z) \rightarrow r = \gamma \cdot S_q(q - Z)$ is a two-level scaling scheme
- $r = s_{l_0} \cdot s_{l_1} \cdot \ldots \cdot (q - z)$
  - r: real number value
  - q: quantized value
  - z: zero point ($z = 0$ is symmetric quantization)
  - $s_{l_0}, s_{l_1}, \ldots$ **: scale factors of different levels (fine → coarse)**

Group Quantization

Multi-level scaling scheme

- $r = s_{l_0} \cdot s_{l_1} \cdot \ldots \cdot (q - z)$
  - r: real number value
  - q: quantized value
  - z: zero point ($z = 0$ is symmetric quantization)
  - $s_{l_0}, s_{l_1}, \ldots$: scale factors of different levels (fine → coarse)

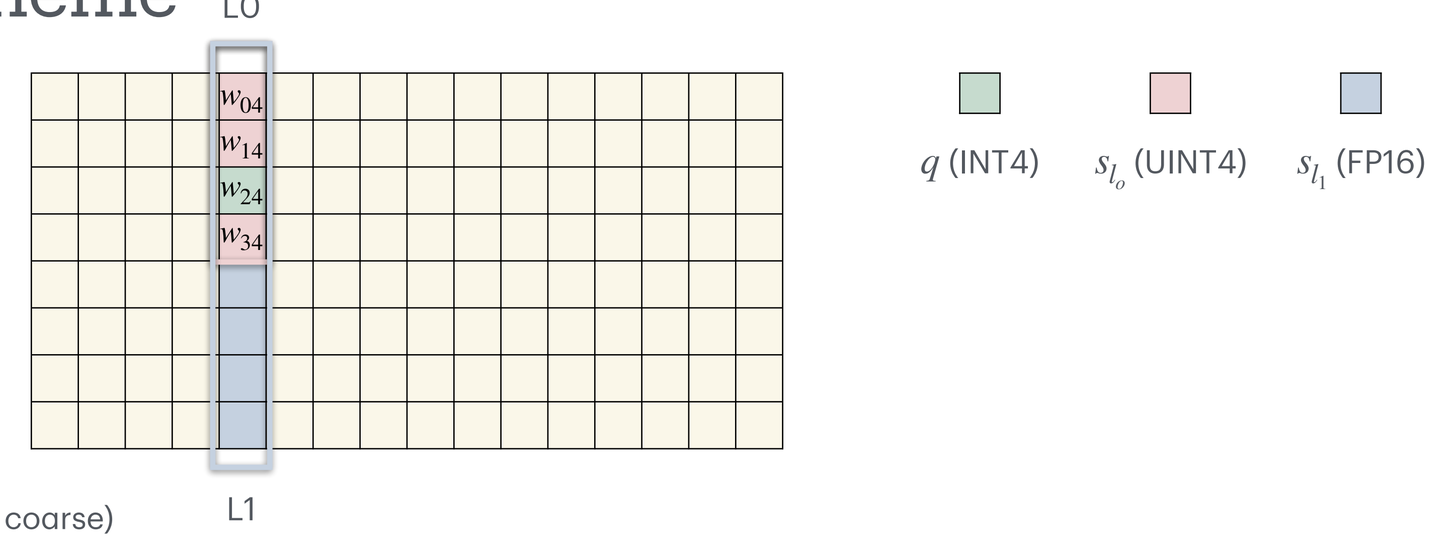


Approach	Quantized Data Type	L0 Group Size	L0 Scale Data Type	L1 Gourp Size	L1 Scale Data Type	Effective Bit Width
Per-Channel Quant	INT4	Per Channel	FP16	-	-	$4 + \frac{16}{N} \approx 4 (N \gg 16)$

Group Quantization

Multi-level scaling scheme

- $r = s_{l_0} \cdot s_{l_1} \cdot \ldots \cdot (q - z)$
  - r: real number value
  - q: quantized value
  - z: zero point ($z = 0$ is symmetric quantization)
  - $s_{l_0}, s_{l_1}, \ldots$: scale factors of different levels (fine → coarse)

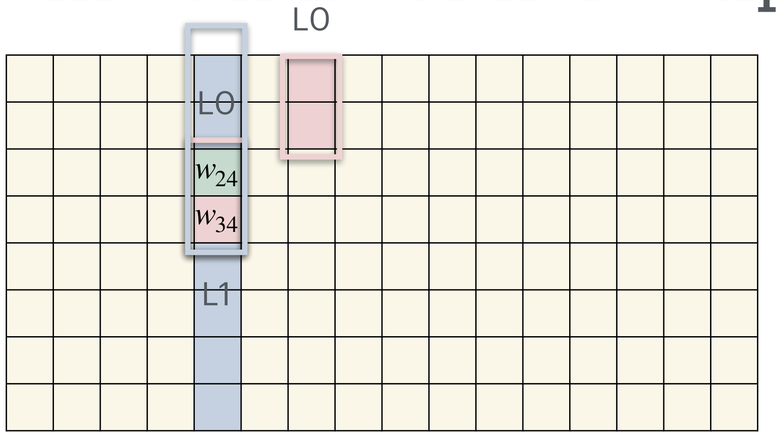


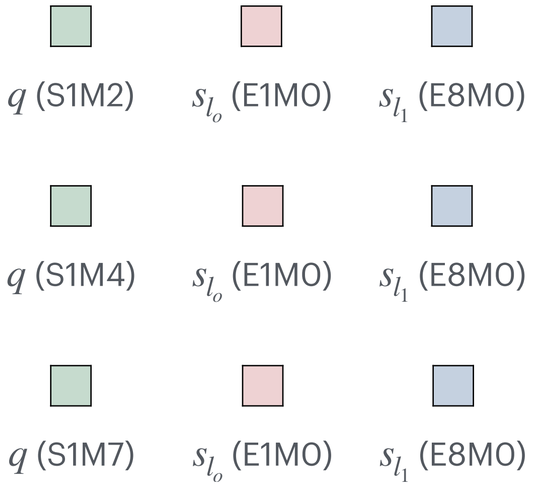
Approach	Quantized Data Type	L0 Group Size	L0 Scale Data Type	L1 Gourp Size	L1 Scale Data Type	Effective Bit Width
Per-Channel Quant	INT4	Per Channel	FP16	-	-	$4 + \frac{16}{N} \approx 4 (N \gg 16)$
VS-Quant	INT4	16	UINT4	Per Channel	FP16	$4 + \frac{4}{16} + \frac{16}{N} \approx 4.25 (N \gg 16)$

Group Quantization

Multi-level scaling scheme: shared Micro-exponent (MX) data type

- $r = s_{l_0} \cdot s_{l_1} \cdot \ldots \cdot (q - z)$
  - r: real number value
  - q: quantized value
  - z: zero point ($z = 0$ is symmetric quantization)
  - $s_{l_0}, s_{l_1}, \ldots$: scale factors of different levels (fine → coarse)





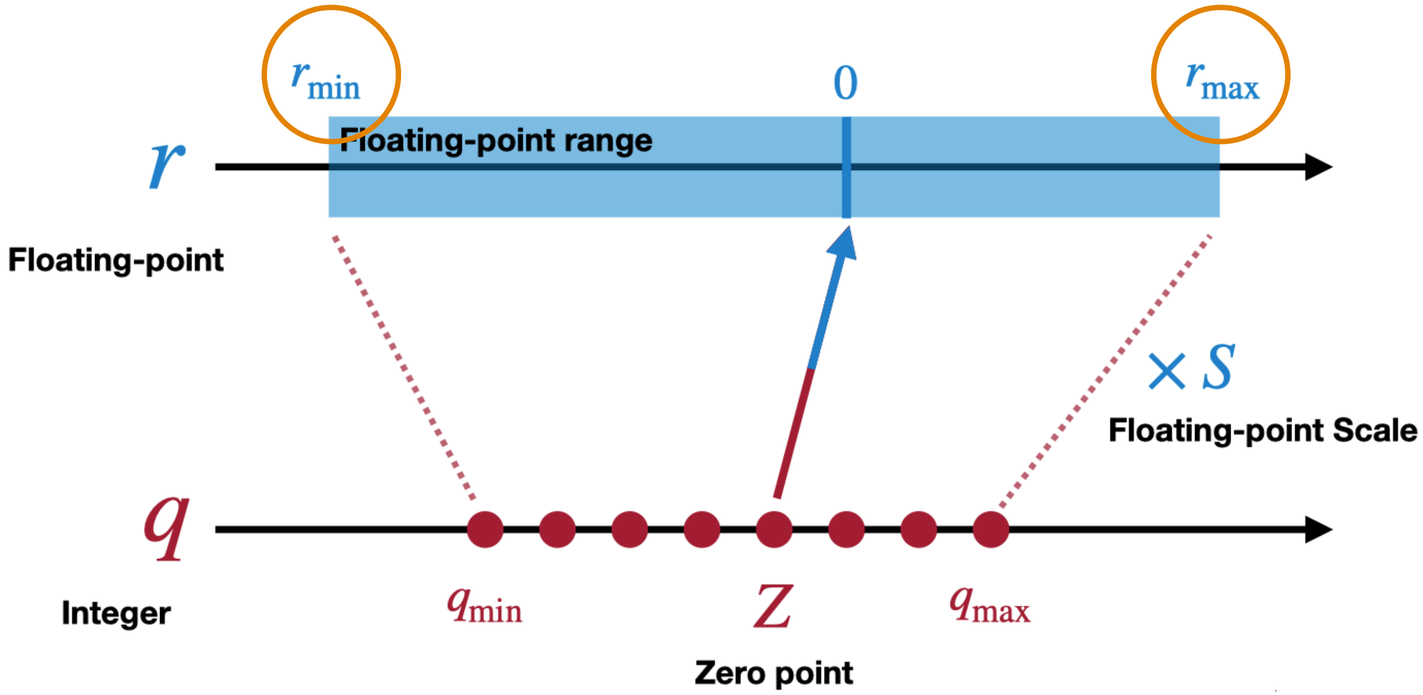
Approach	Quantized Data Type	L0 Group Size	L0 Scale Data Type	L1 Gourp Size	L1 Scale Data Type	Effective Bit Width
Per-Channel Quant	INT4	Per Channel	FP16	-	-	$4 + \frac{16}{N} \approx 4 (N \gg 16)$
VS-Quant	INT4	16	UINT4	Per Channel	FP16	$4 + \frac{4}{16} + \frac{16}{N} \approx 4.25 (N \gg 16)$
MX4	S1M2	2	E1M0	16	E8M0	$3 + \frac{1}{2} + \frac{8}{16} = 4$
MX6	S1M4	2	E1M0	16	E8M0	$5 + \frac{1}{2} + \frac{8}{16} = 6$
MX9	S1M7	2	E1M0	16	E8M0	$8 + \frac{1}{2} + \frac{8}{16} = 9$

Darvish Rouhani, B., Zhao, R., Elango, V., Shafipour, R., Hall, M., Mesmakhosroshahi, M., ... & Naumov, M. (2023, June). With shared microexponents, a little shifting goes a long way. In Proceedings of the 50th Annual International Symposium on Computer Architecture (pp. 1-13).

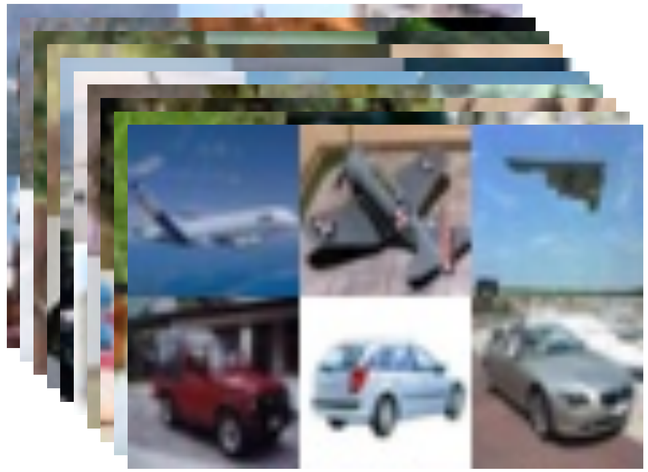
Post-Training Quantization

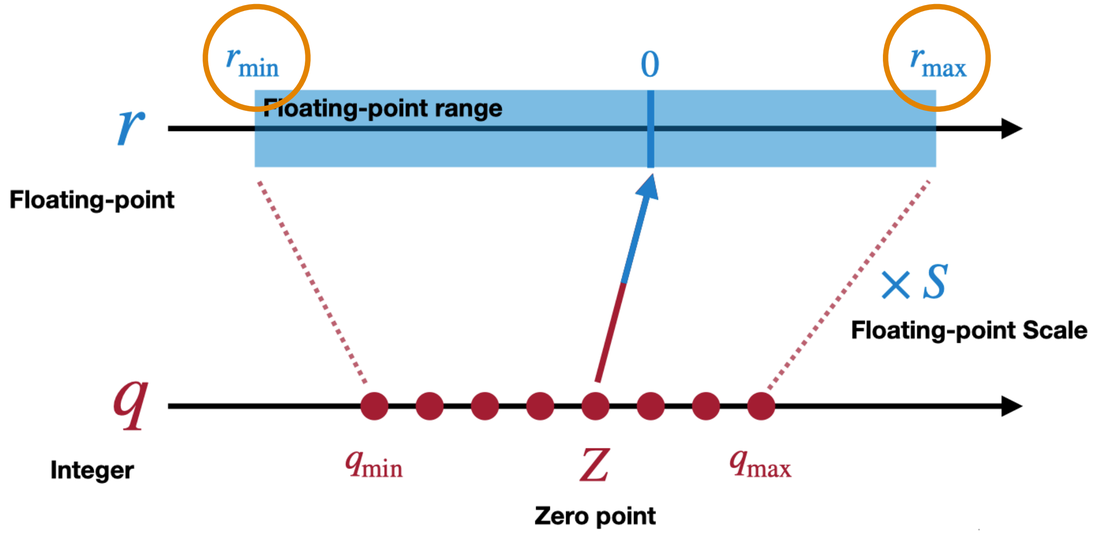
How to get the optimal linear quantization parameters S and Z?

- Quantization Granularity
- Dynamic Range Clipping



Linear Quantization on Activations



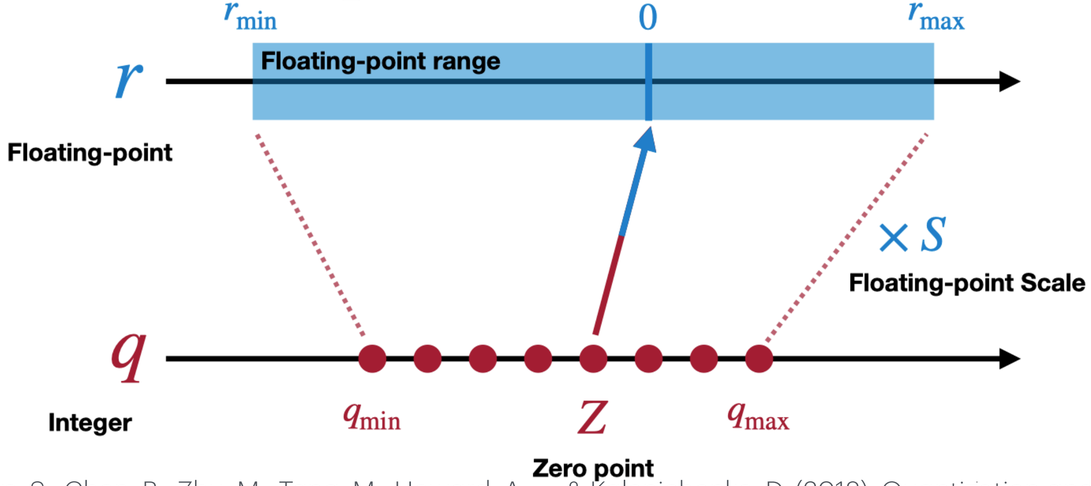


- Unlike weights, the activation range varies across inputs
- Collect activations statistics **before** deploying the model, to determine the dynamic range r_{max}, r_{min}
  - Also known as static quantization

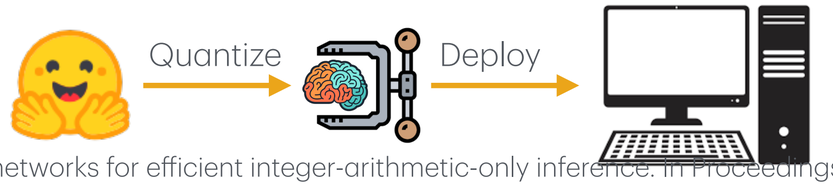
Dynamic Range for Activation Quantization

Collect activations statistics before deploying the model

- If we have access to the training process, we can calculate the dynamic range during training
- **Exponential moving averages (EMA)**
  - $\hat{r}^{(t)}_{max,min} = \alpha \cdot r^{t}_{max,min} + (1-\alpha) \cdot \hat{r}^{(t-1)}_{max,min}$, α is the discount factor
  - Observed ranges are smoothed across thousands of training steps



🤔 **What if we don’e have access to the trainng process?**

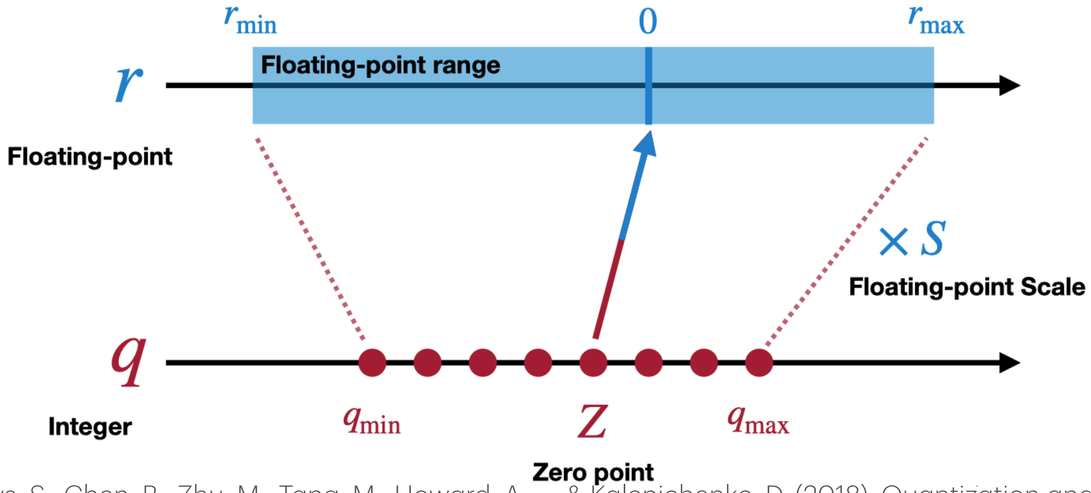


Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2704-2713).

Dynamic Range for Activation Quantization

Collect activations statistics before deploying the model

- If we don't have access to the training process, we can run **"calibration" batches (dataset)** on the trained FP32 model, to collect r_{min}, r_{max} at the runtime.
  - The calibration dataset is separate from test, representative for the input images
  - Use mean of the min/max of each sample in the batches



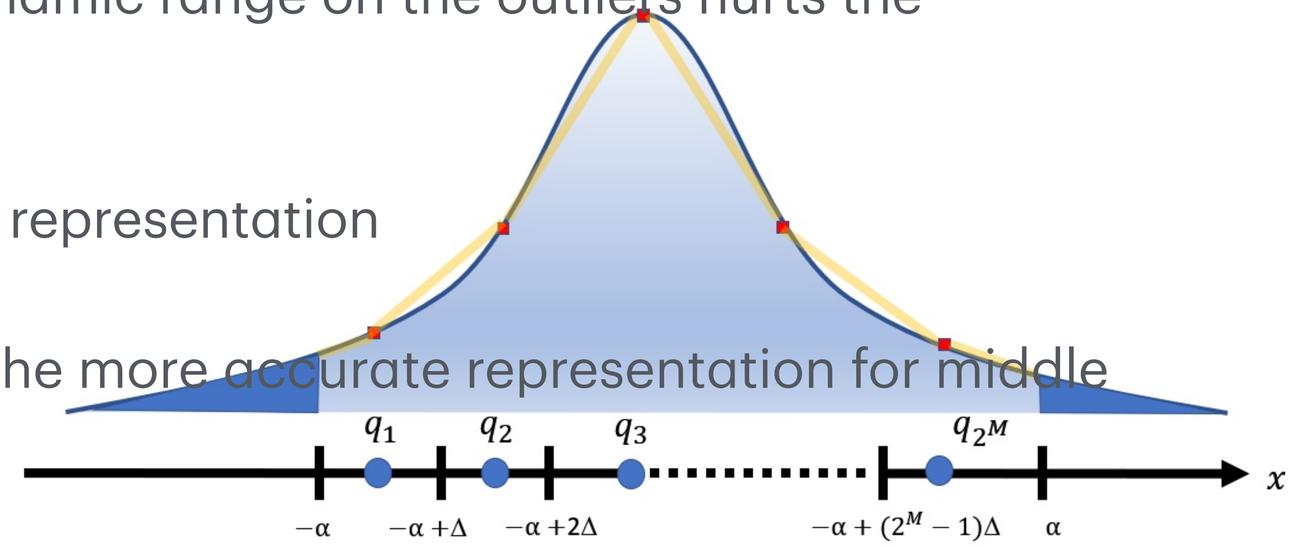
Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2704-2713).

Dynamic Range for Activation Quantization

Collect activations statistics before deploying the model

- If we don't have access to the training process, we can run **"calibration" batches (dataset)** on the trained FP32 model, to collect r_{min}, r_{max} at the runtime.

- Clipping value α denotes r_{max}, spending dynamic range on the outliers hurts the representation ability
  - If α is the exact max value, a big waste of representation
  - If closer α, $-\alpha$, the closer the boundary, the more accurate representation for middle values
  - How to find the sweet spot?



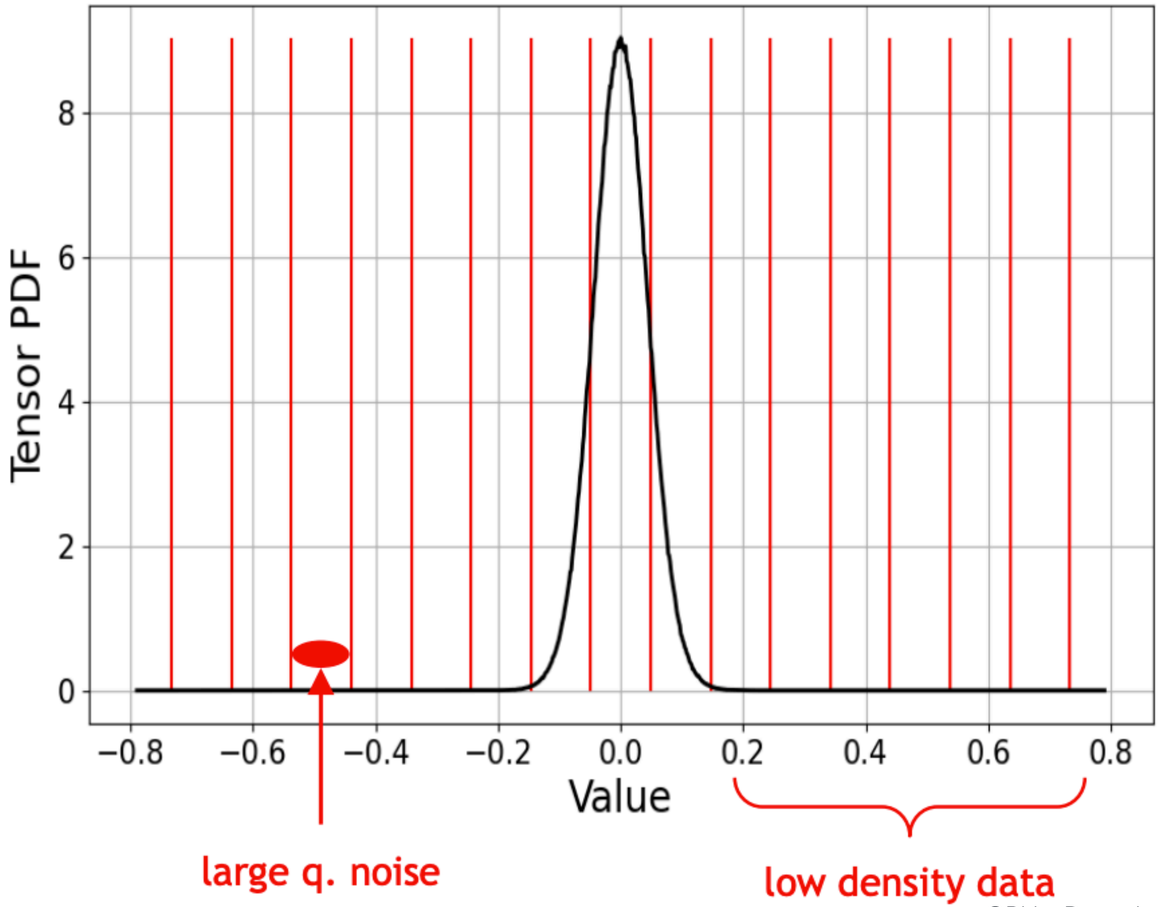
An activation distribution quantized uniformly in the range $[-\alpha, \alpha]$ with 2^M equal quantization intervals (bins)

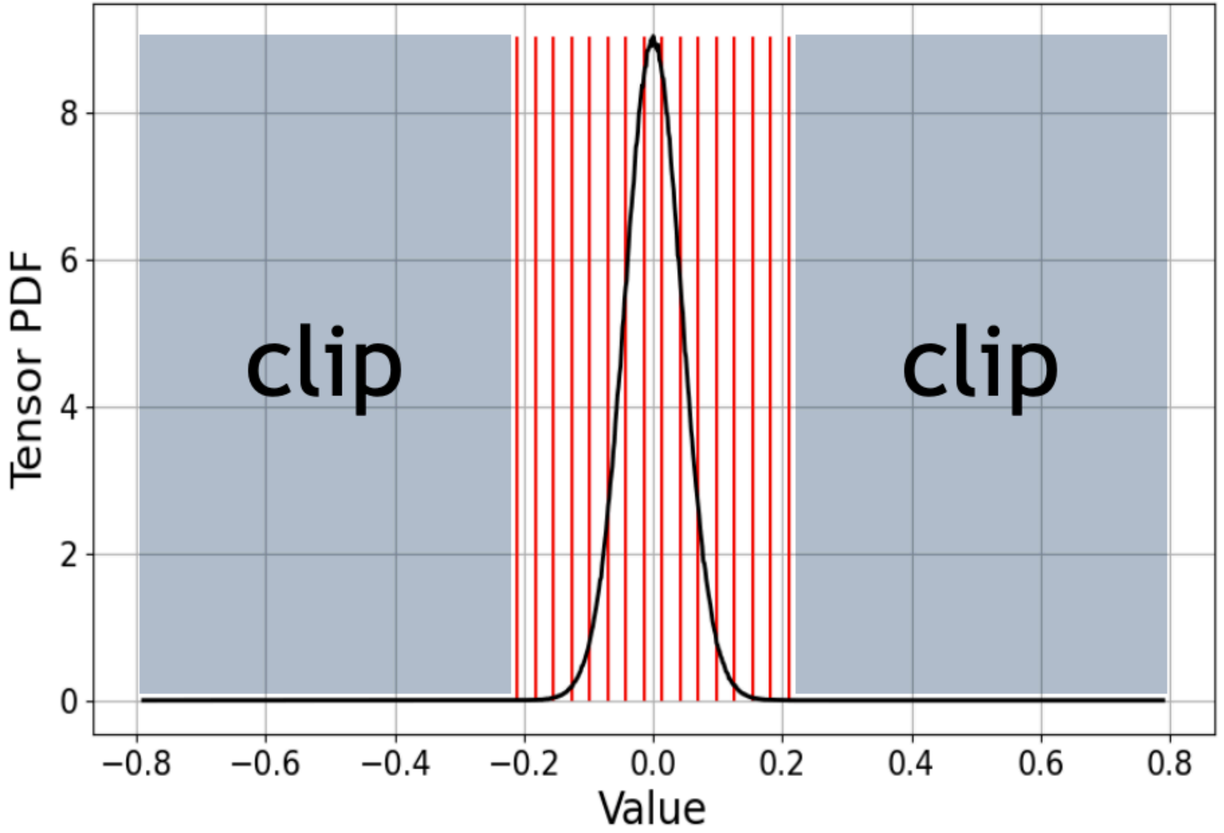
Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2704-2713).

Dynamic Range for Quantization

Optimal Clipping

Max-scaled quantization vs. Clipped quantization





Dynamic Range for Activation Quantization

Find r_{max} with the calibration batches

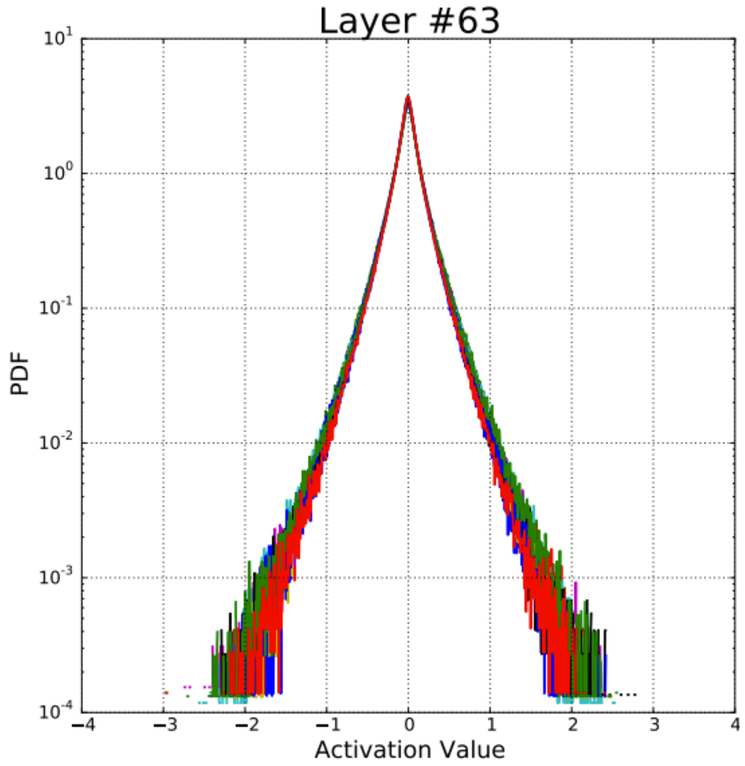
- **Minimize the mean-square-error (MSE)** between inputs X and (reconstructed) quantized inputs $Q(X)$: $\min_{|r|_{min}} \mathbb{E}[(X - Q(X))^2]$
  - Assume the inputs (activation) follows a **Gaussian or Laplace distribution**
  - For Laplace $(0,b)$ distribution, optimal clipping values can be solved numerically as $|r|_{max} = 2.83b, 3.89b, 5.03b$ for 2, 3, 4 bits
  - The Laplace parameter b can be estimated from calibration input distribution
  - But it's a very RARE case!

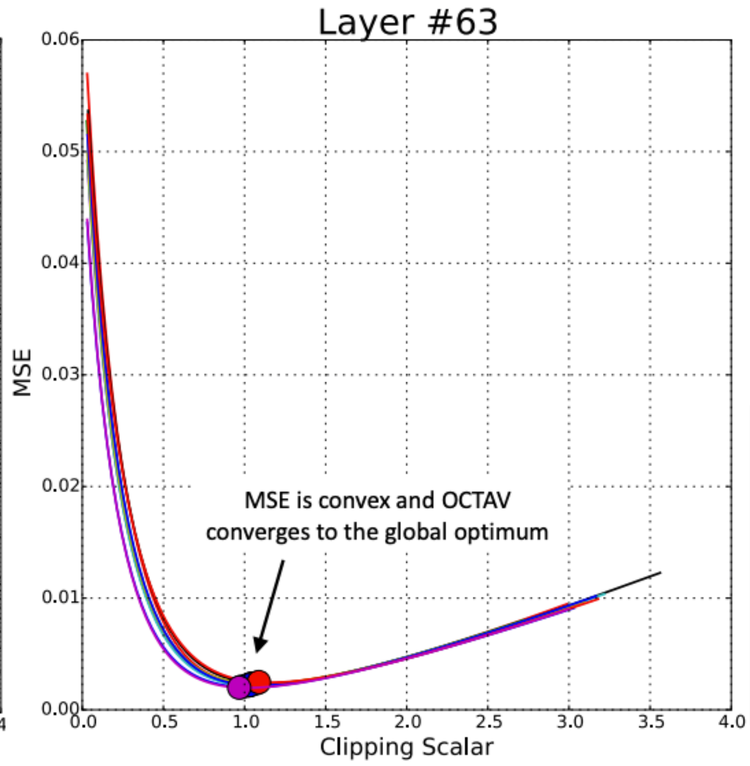
Banner, R., Nahshan, Y., & Soudry, D. (2019). Post training 4-bit quantization of convolutional networks for rapid-deployment. Advances in Neural Information Processing Systems, 32.

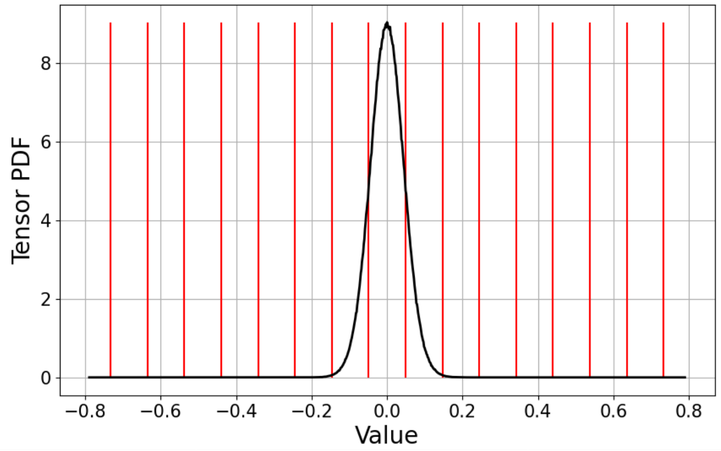
Dynamic Range for Quantization

Minimize MSE using Newton-Raphson method

- An arbitrary activation layer in BERT-Base for 10 different inputs







	FP32 Accuracy	OCTAV INT4
ResNet-50	76.07	75.84
MobileNet-V2	71.71	70.88
BERT-Large	91.00	87.09

Sakr, C., Dai, S., Venkatesan, R., Zimmer, B., Dally, W., & Khailany, B. (2022, June). Optimal clipping and magnitude-aware differentiation for improved quantization-aware training. In International Conference on Machine Learning (pp. 19123-19138). PMLR.

Dynamic Range for Activation Quantization

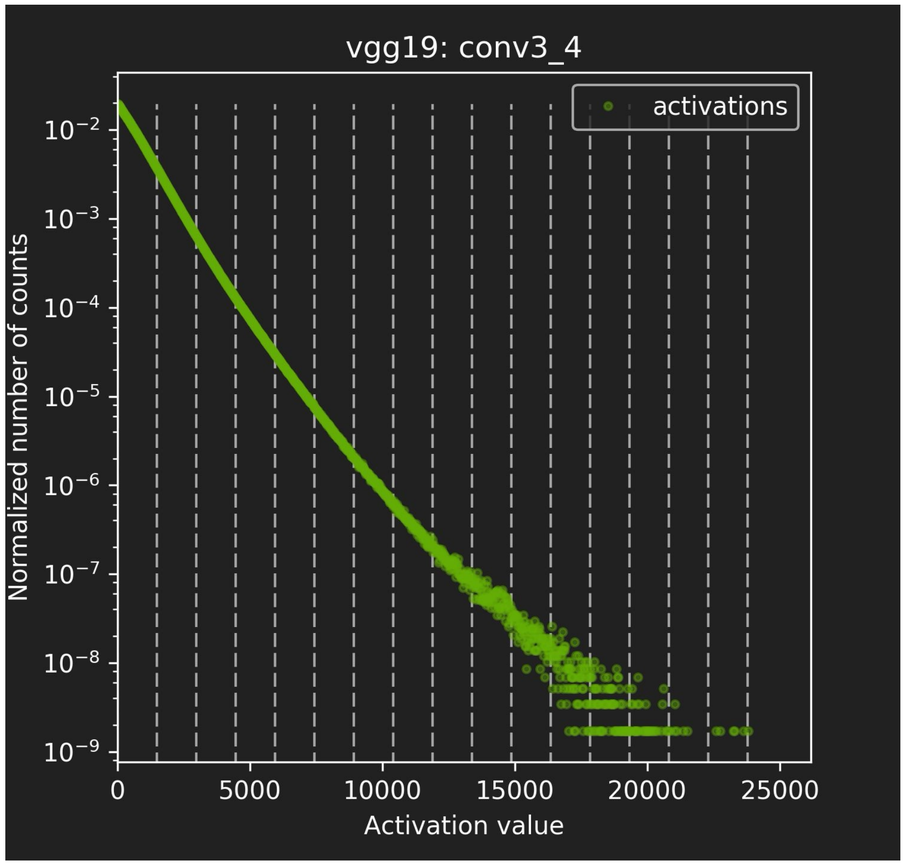
Find r_{max} with the calibration batches

- **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

Banner, R., Nahshan, Y., & Soudry, D. (2019). Post training 4-bit quantization of convolutional networks for rapid-deployment. Advances in Neural Information Processing Systems, 32.

Dynamic Range for Activation Quantization

Minimize loss of information by minimizing the KL divergence



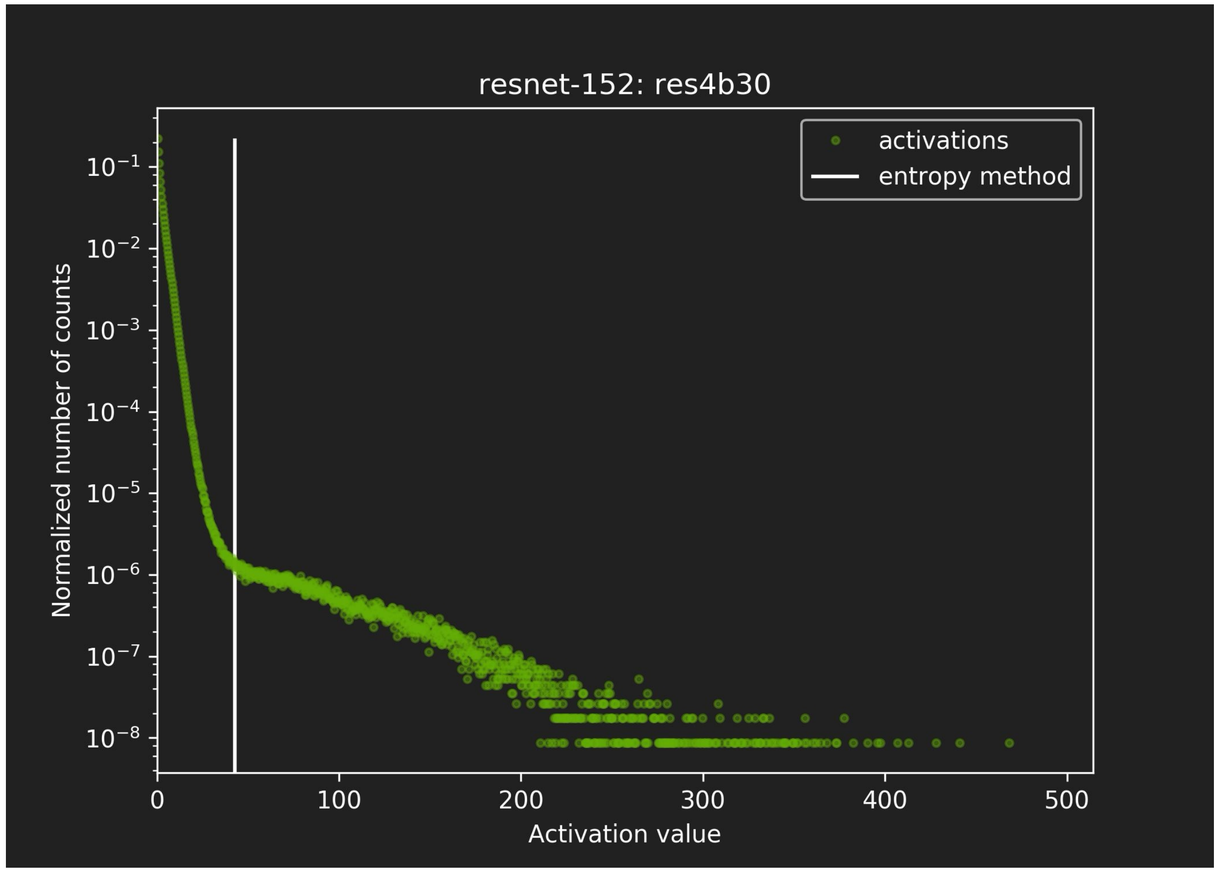
8-bit Inference with TensorRT

- Run FP32 inference on Calibration dataset
  - **Representative**, diverse, ideally a subset of val dataset, 1000s of samples
- For eacy layer
  - Collect histograms of activations
  - Generate many quantized distributions with different saturation thresholds T
  - Pick threshold with minimize KL divergence

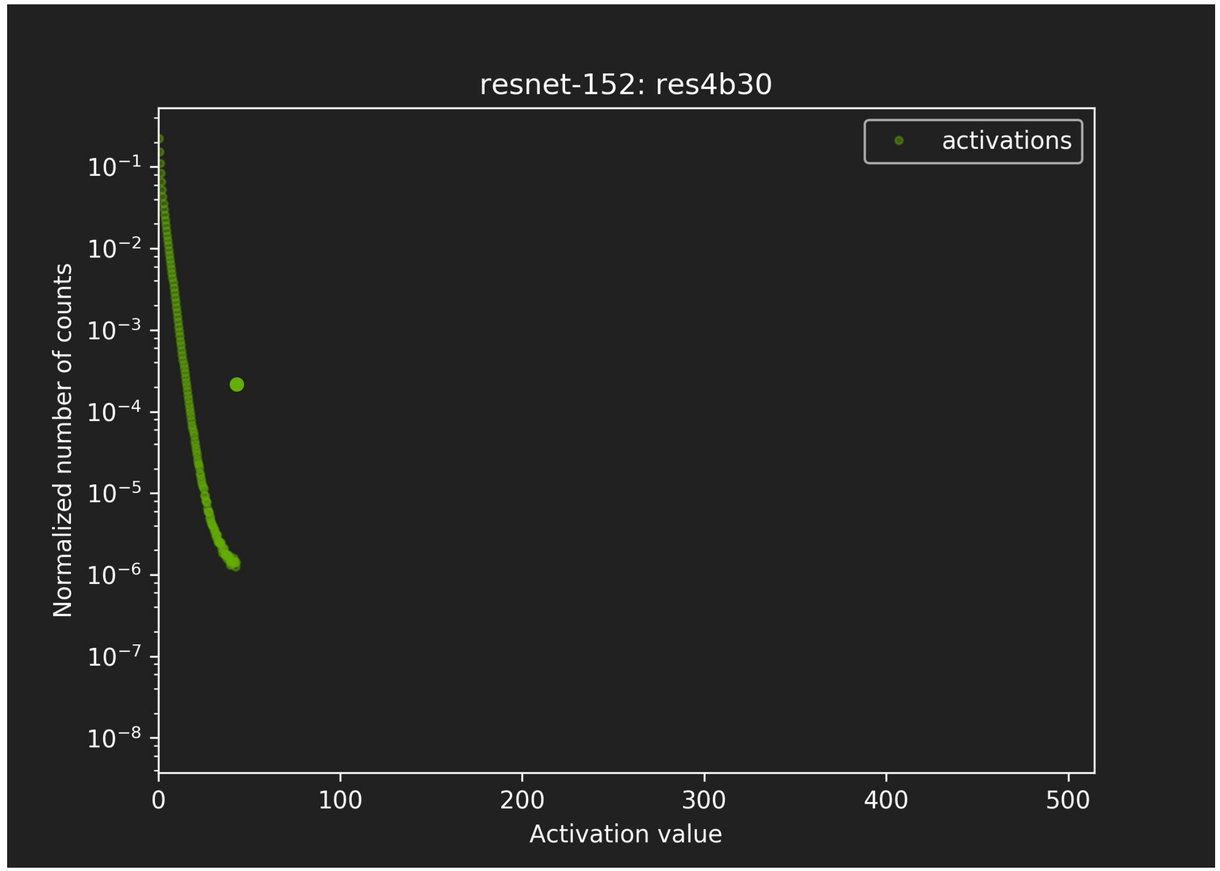
$$arg \min_{t} D_{KL}(P||Q), t \in T$$

Dynamic Range for Activation Quantization

Minimize loss of information by minimizing the KL divergence



Before saturation

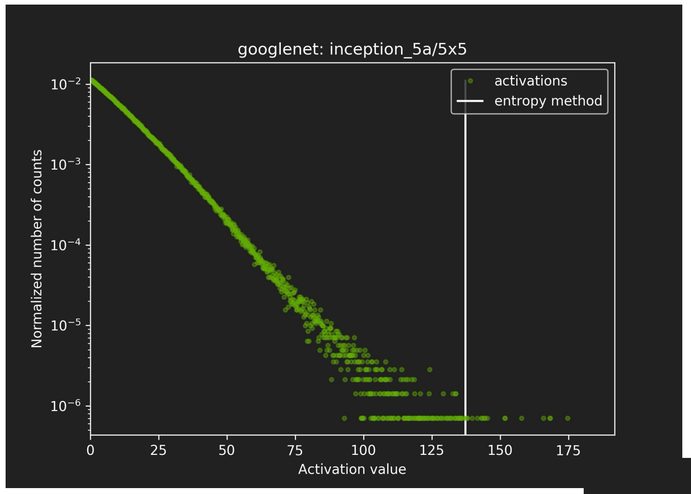


After saturation

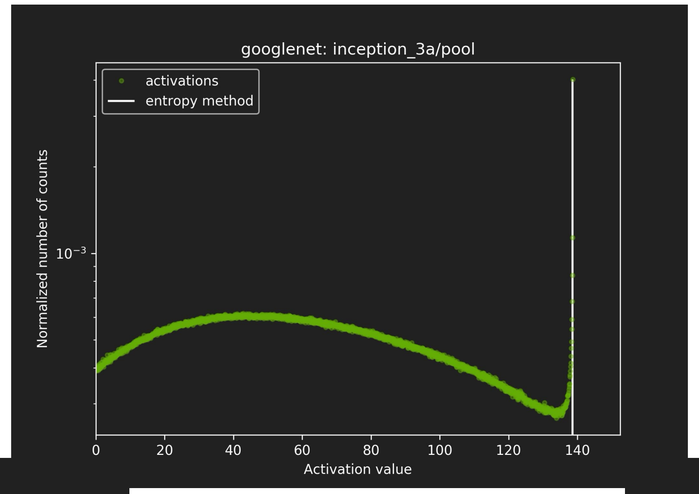
Dynamic Range for Activation Quantization

Minimize loss of information by minimizing the KL divergence

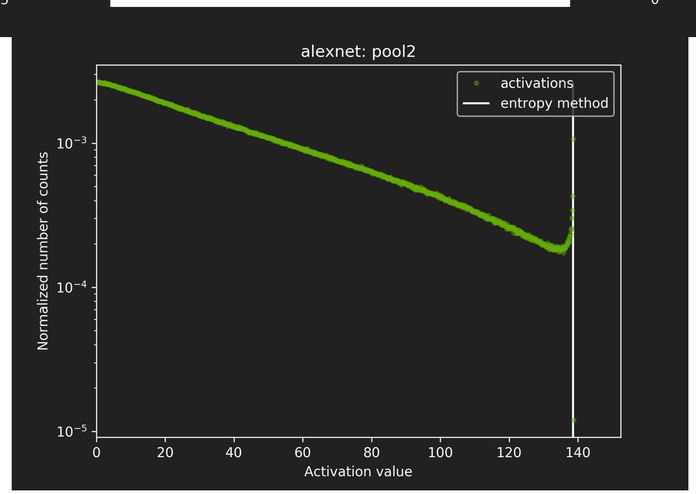
GoogleNet: inception_5a/5x5



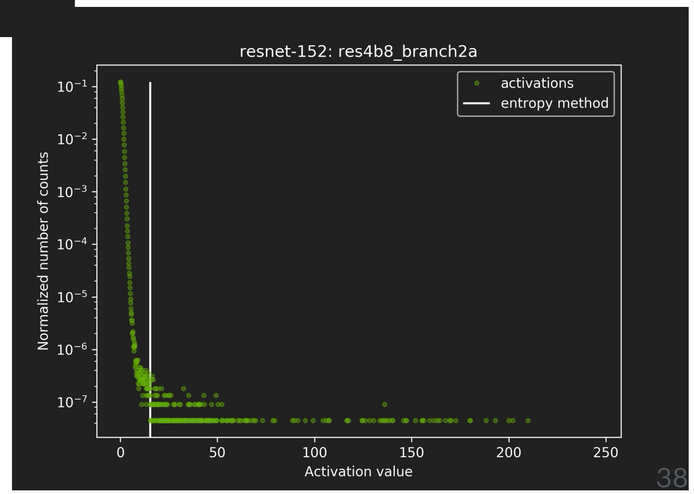
GoogleNet: Inception_3a/pool



AlexNet: Pool2



ResNet-152: res4b8_branch2a

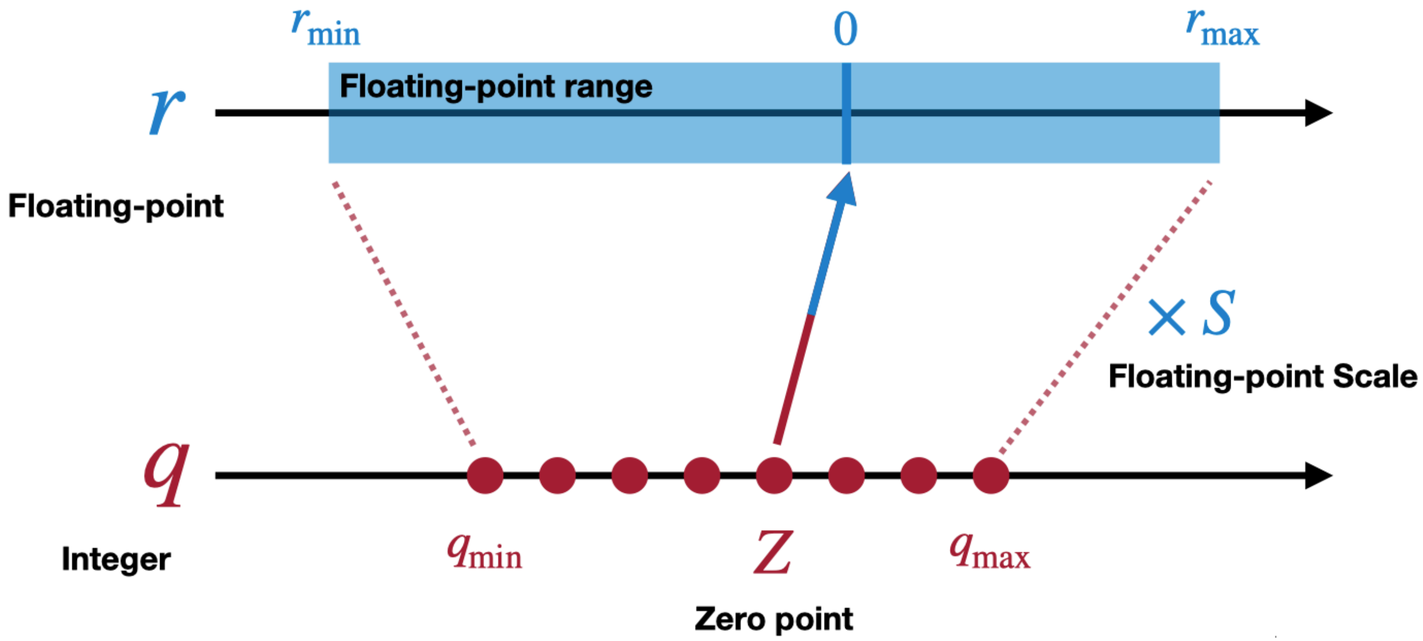


Post-Training Quantization

How to get the optimal linear quantization parameters S and Z?

- Quantization Granularity
- Dynamic Range Clipping
- Rounding

🤔 **What is the simpliest rounding?**



Adaptive Rounding for Weight Quantization

Rounding-to-nearest is not always optimal

- Weights are correlated with each other. The best rounding for each weight (to nearest) is not the best rounding for the whole tensor.
- What is optimal?
  - Consider the whole tensor's effect by trying to match the activation before and after rounding

Adaptive Rounding for Weight Quantization

Qualcomm AI research

AdaRound method

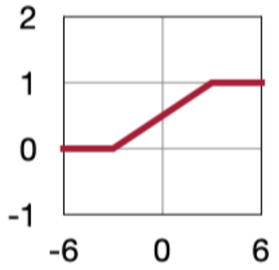
- Instead of $\lfloor x \rceil$, we want to choose from $\{\lfloor x \rfloor, \lceil x \rceil\}$ to get the best reconstruction
- A learning-based method to find quantized value $\tilde{w} = \lfloor \lfloor w \rfloor + \delta \rceil, \delta \in [0,1]$
- We optimize the following equation (omit the derivation)

$$\arg\min_{v} \|W_X - \tilde{W}X\|_F^2 + \lambda f_{reg}(V)$$

$$\rightarrow \arg\min_{v} \|W_X - \lfloor \lfloor W \rfloor + h(V) \rceil X\|_F^2 + \lambda f_{reg}(V)$$

Takeaway:
Instead of rounding to the nearest, add some randomness by applying ***stochastic rounding***.

- X is the input of the layer, V is a random variable of the same shape
- $h()$ is a function to map V to $(0,1)$, such as rectified sigmoid
- $f_{reg}(V)$ is a regularization that encourages $h(V)$ to be binary (1 or 0)



Nagel, M., Amjad, R. A., Van Baalen, M., Louizos, C., & Blankevoort, T. (2020, November). Up or down? adaptive rounding for post-training quantization. In International Conference on Machine Learning (pp. 7197-7206). PMLR.

Post-Training Quantization Summary

- **Zero Point**
  - Asymmetric
  - Symmetric
- **Scaling granularity**
  - Per-tensor
  - Per-channel
  - Group Quantization
- **Dynamic Range Clipping**
  - Exponential Moving Average
  - Minimizing Mean-square-error
  - Minimizing KL divergence
- **Rounding**
  - Round-to-nearest
  - AdaRound

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

Quantized weights (2-bit signed int)

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

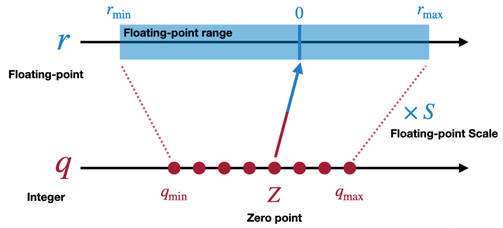
Zero point (2-bit signed int) **Scale (32-bit float)**

$$(\quad - \quad -1) \times \quad 1.07$$

Quantization parameters

$$r = (\quad q \quad - \quad Z) \times \quad S$$

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



Post-Training Quantization

Quantize a floating-point neural network model,
to reduce the model size, improve inference speed, and lower power consumption

🤩 **Quantize on a trained model, no fine-tuning needed**

😣 **Accuracy drop**

Post-Training INT8 Linear Quantization

Activation		Symmetric	Asymmetric
		Per-Tensor	Per-Tensor
		Minimize KL-Divergence	Exponential Moving Average
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%

Post-Training INT8 Linear Quantization

Activation		Symmetric	Asymmetric
		Per-Tensor	Per-Tensor
		Minimize KL-Divergence	Exponential Moving Average
Weight		Symmetric	Symmetric
		Per-Tensor	Per-Channel
Neural Network			0%
			-0.6%
			-1.8%
	MobileNetV1	-	-11.8%
	MobileNetV2	-	-2.1%

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

Quantization-Aware Training

During the training/fine-tuning,
emulate inference-time quantization and recover the accuracy

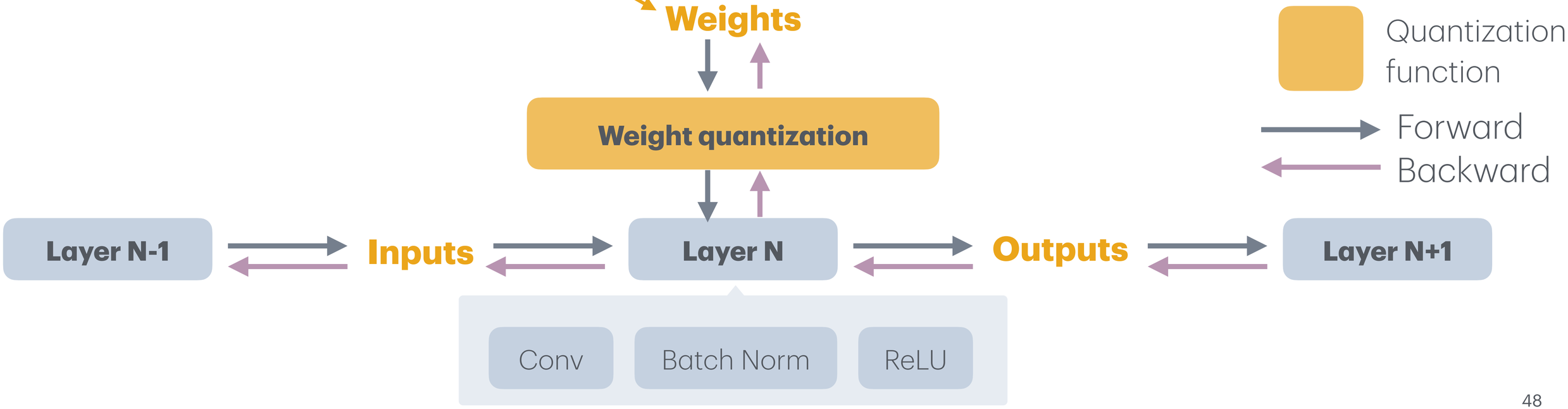
Quantization-Aware Training

- 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

Quantization-Aware Training

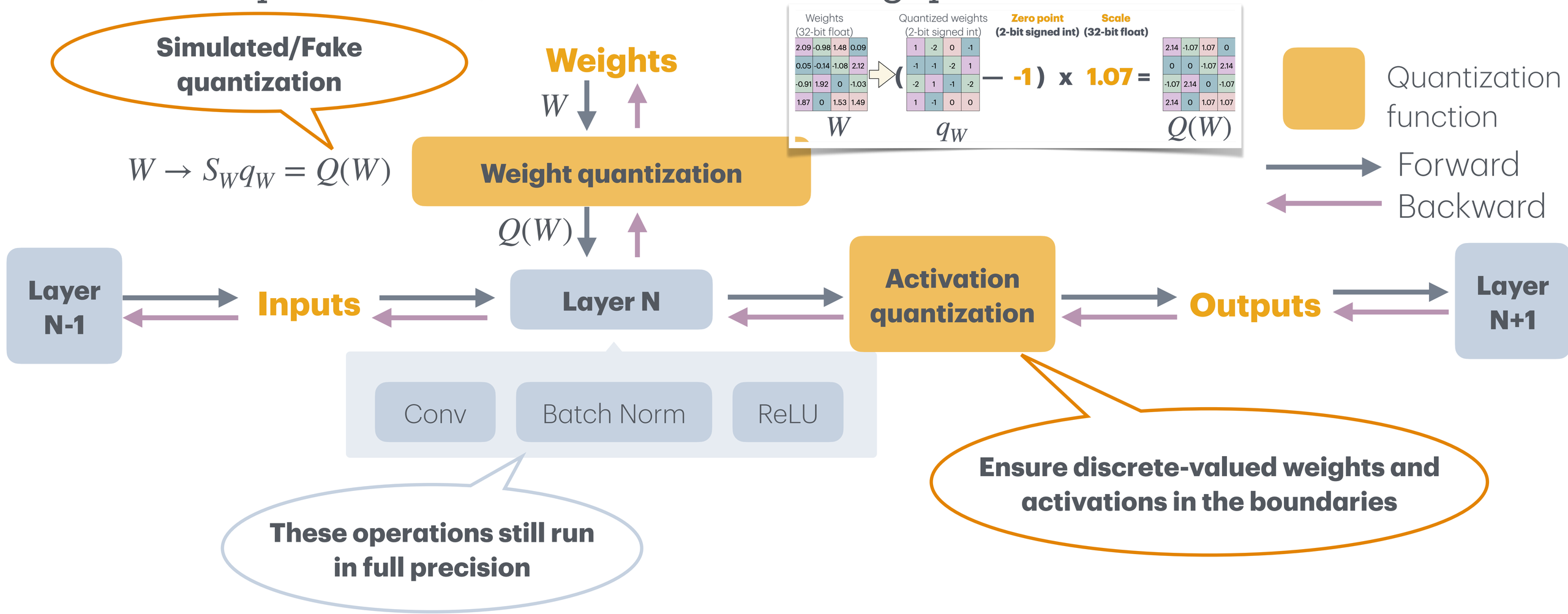
In linear quantization, train the model taking quantization into consideration

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



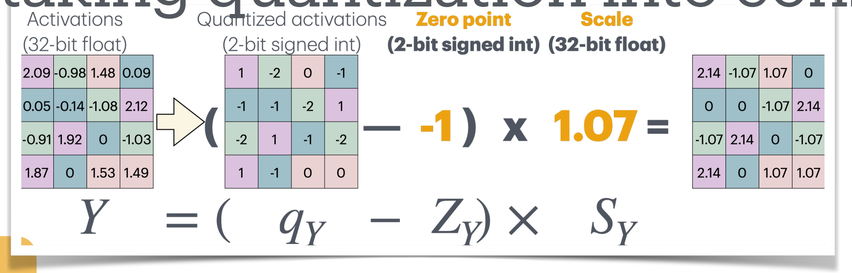
Quantization-Aware Training

In linear quantization, train the model taking quantization into consideration



Quantization-Aware Training

In linear quantization, train the model taking quantization into consideration



Simulated/Fake quantization

$W \rightarrow S_W q_W = Q(W)$

Weights

W

Weight quantization

$Q(W)$

$Y \rightarrow S_Y(q_Y - Z_Y) = Q(Y)$

Forward

Backward

Layer N-1

Inputs

Layer N

Activation quantization

$Q(Y)$

Outputs

Layer N+1

Conv

Batch Norm

ReLU

These operations still run in full precision

Ensure discrete-valued weights and activations in the boundaries

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

Straight-Through Estimator (STE)

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

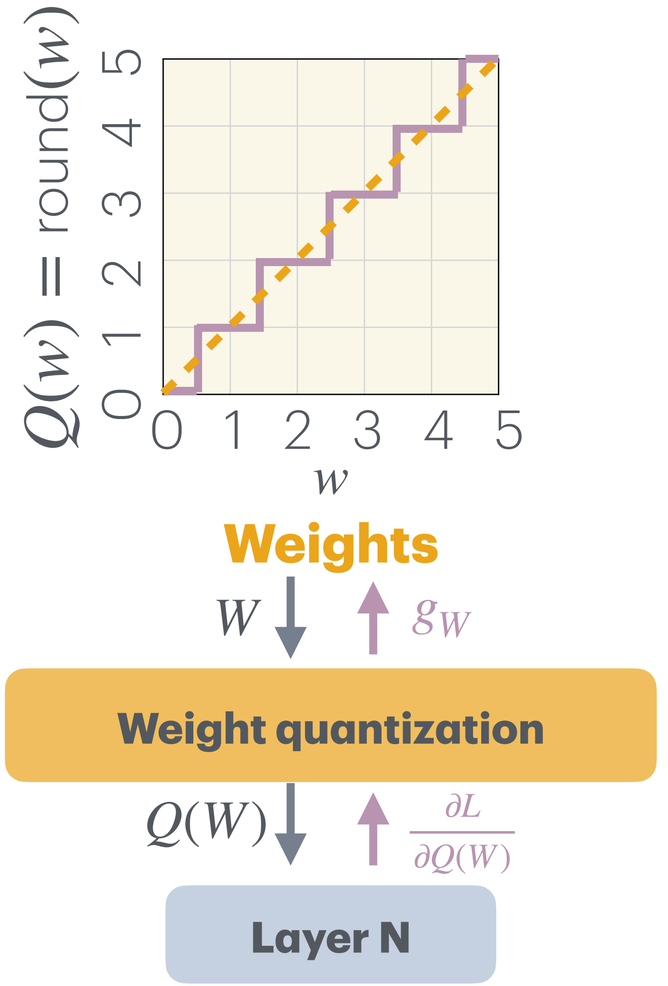
$$\frac{\partial Q(W)}{\partial W} = 0$$

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

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

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

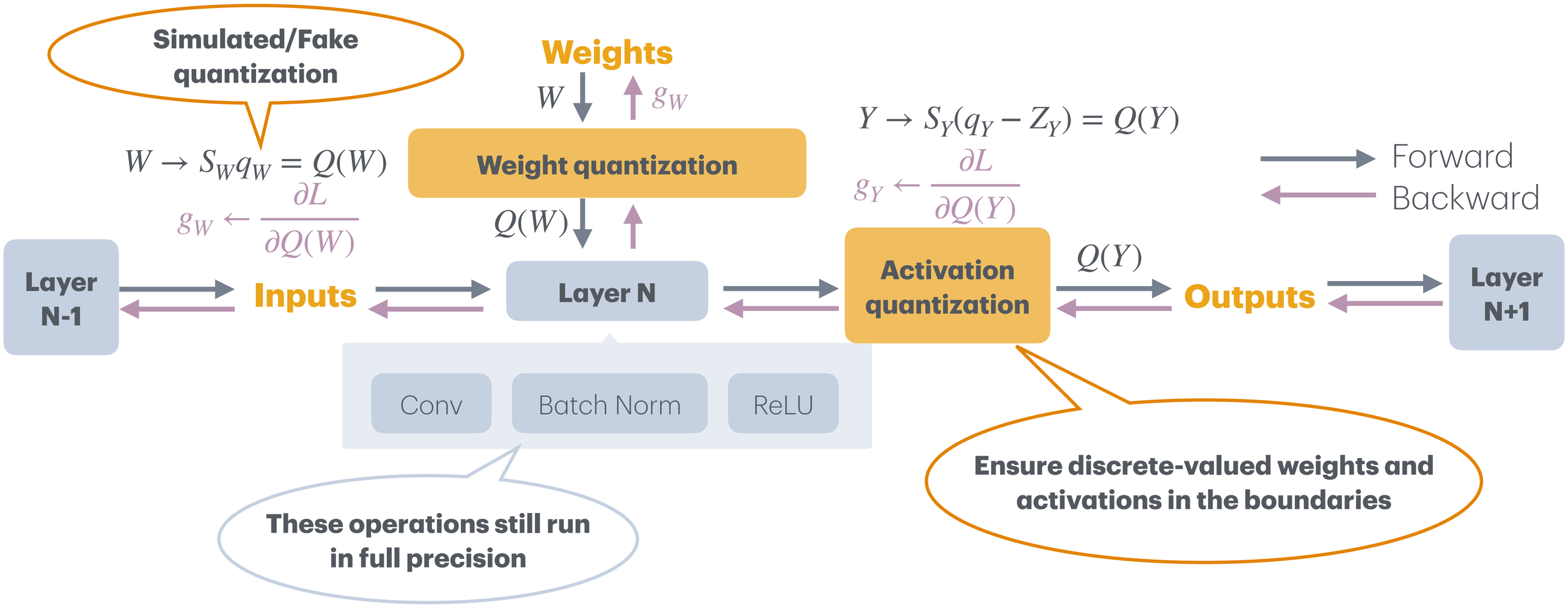
$$g_w = \frac{\partial L}{\partial W} = \frac{\partial L}{\partial Q(W)}$$



Hinton, G., Srivastava, N., & Swersky, K. (2012). Neural networks for machine learning. Coursera, video lectures, 264(1), 2146-2153.
Bengio, Y., Léonard, N., & Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432.

Quantization-Aware Training

In linear quantization, train the model taking quantization into consideration



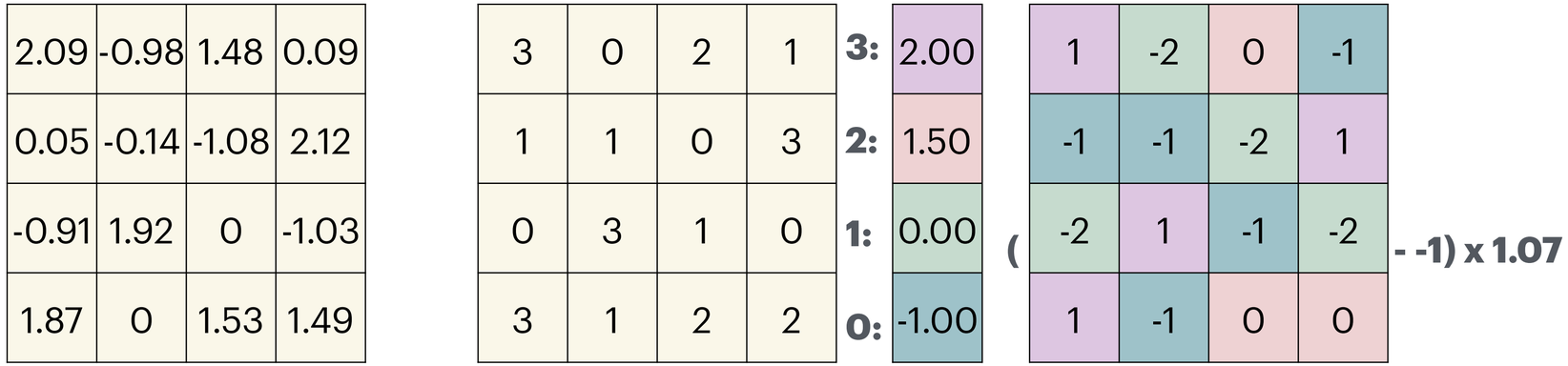
INT8 Linear Quantization-Aware Training

Quantization-aware training provides the best accuracy and allows for simpler quantization scehemes

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%**

Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342.

Neural Network Quantization

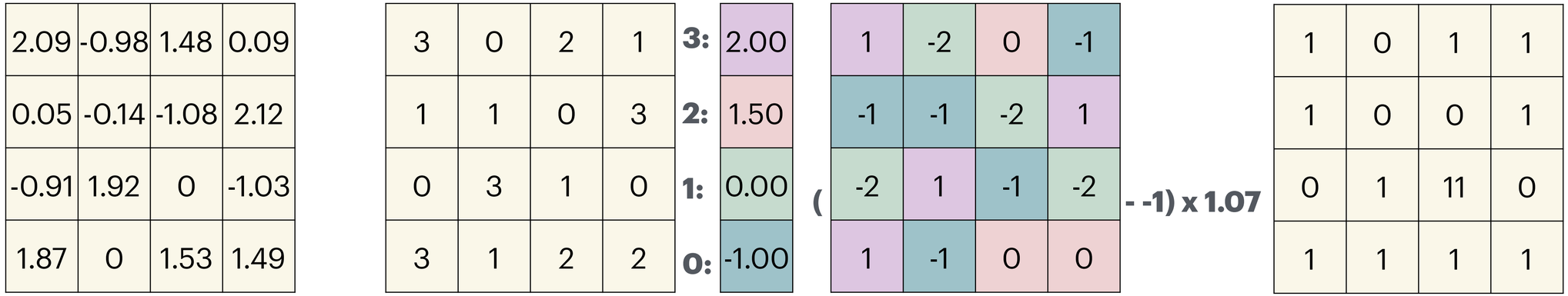


K-Means-based Quantization

Linear Quantization

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

Neural Network 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

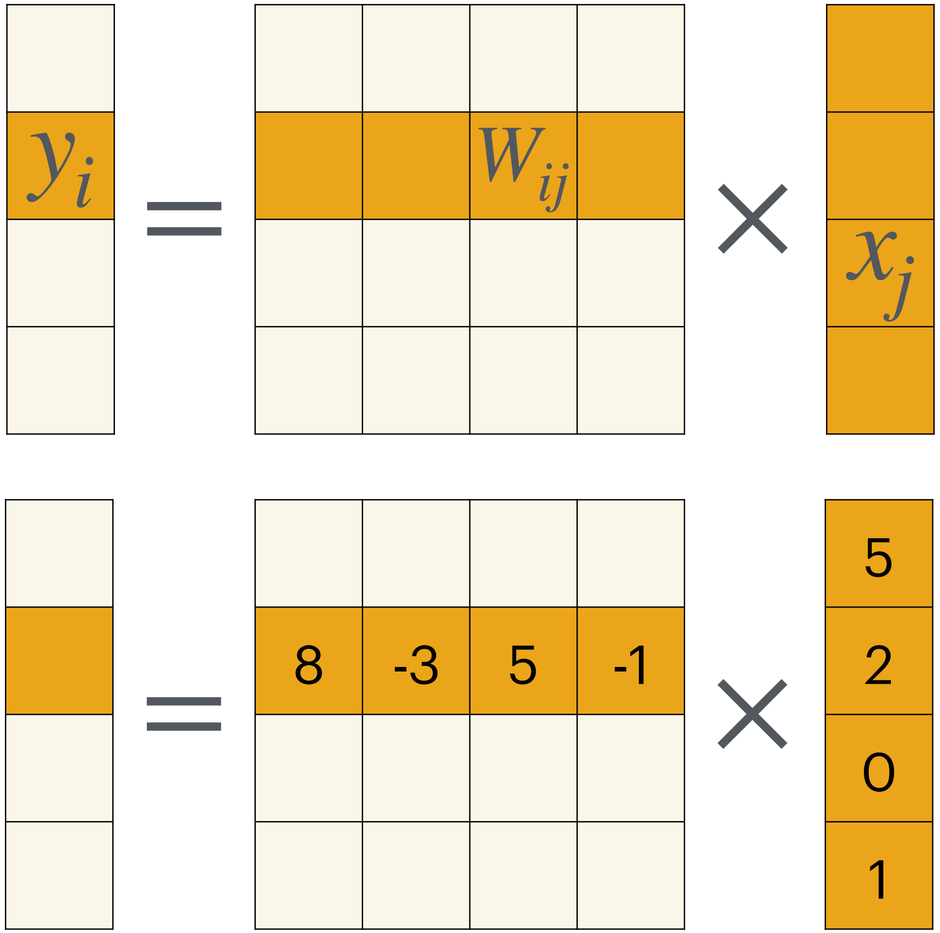
Binary/Ternary Quantization

Push the quantization precision to 1 bit.

Can Quantization Bit Width Go Even Lower?

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

$$= 8 \times 5 + (-3) \times 2 + 5 \times 0 + (-1) \times 1$$

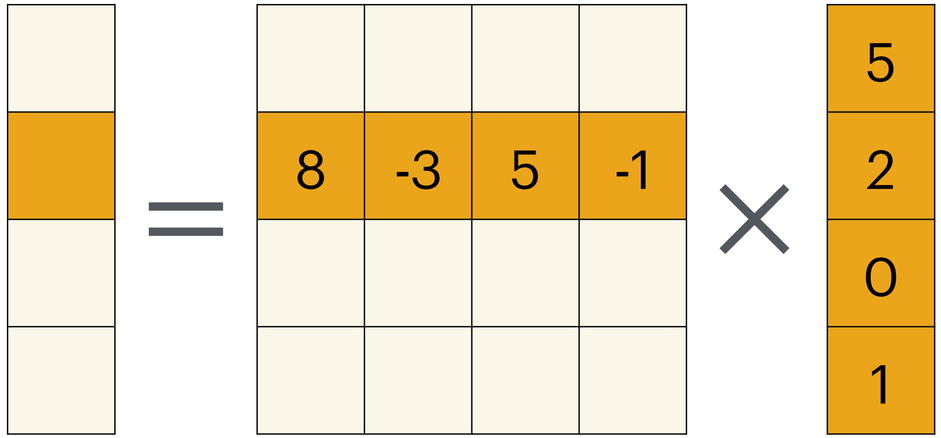


Input	Weight	Operations	Memory	Computation
$\mathbb{R}$	$\mathbb{R}$	$+\times$	$1\times$	$1\times$

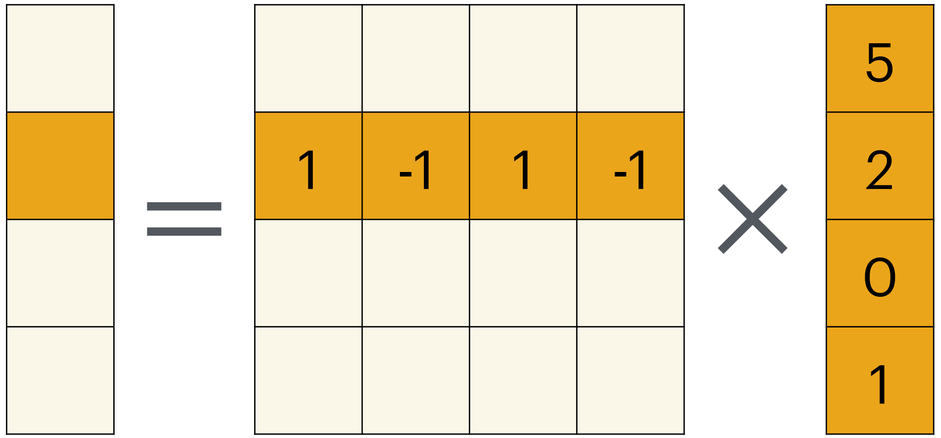
If Weights are Quantized to +1 and -1

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

$$= 5 - 2 + 0 - 1$$



Input	Weight	Operations	Memory	Computation
$\mathbb{R}$	$\mathbb{R}$	$+\times$	$1\times$	$1\times$
$\mathbb{R}$	$\mathbb{B}$	$+-$	~32x less	~2x less



Binarization

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

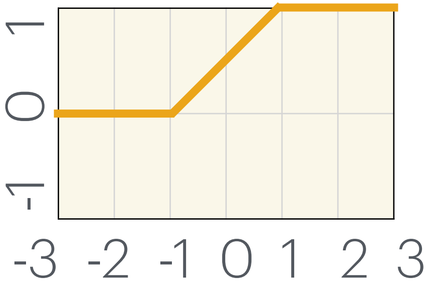
$$q = \text{sign}(r) = \begin{cases} +1, r \leq 0 \\ -1, r < 0 \end{cases}$$

😣 **Accuracy degradation**

- **Stochastic binarization** uses global statistics or the value of input data to determine the probability of being -1 or +1

  - For example, in BinaryConnect, probability is determined by hard sigmoid function $\sigma(r)$

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



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

BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations

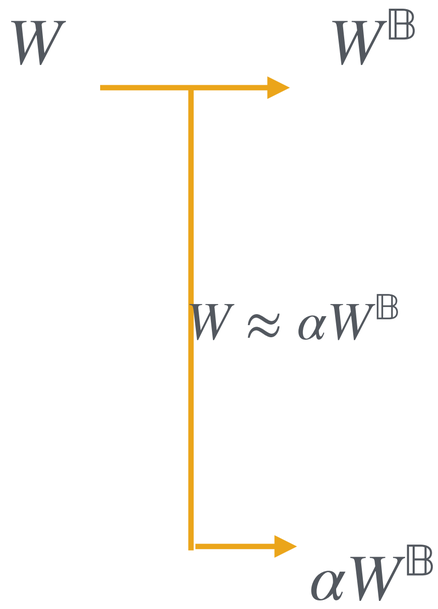
Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830.

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

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



Binary weights (1-bit)

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

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

AlexNet-based Network	ImageNet Top-1 △ Accuracy
Binary Connect (BC)	-21.2%
Binary Weight Network (BWN)	**0.2%**

Scale (32-bit float)

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

$$= \frac{1}{16} \| W \|_1 = \frac{1}{16} \times 16.78 \approx 1.05$$

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

Scale (32-bit float)

X 1.05

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

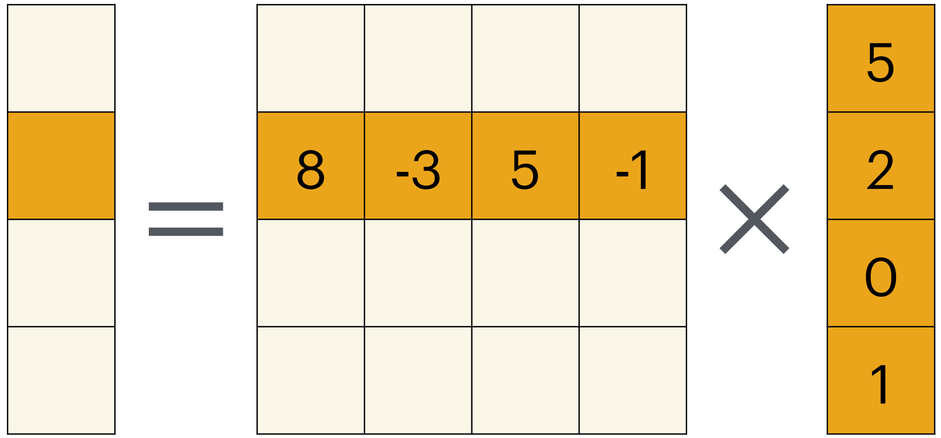
Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, September). Xnor-net: Imagenet classification using binary convolutional neural networks. In European conference on computer vision (pp. 525-542). Cham: Springer International Publishing.

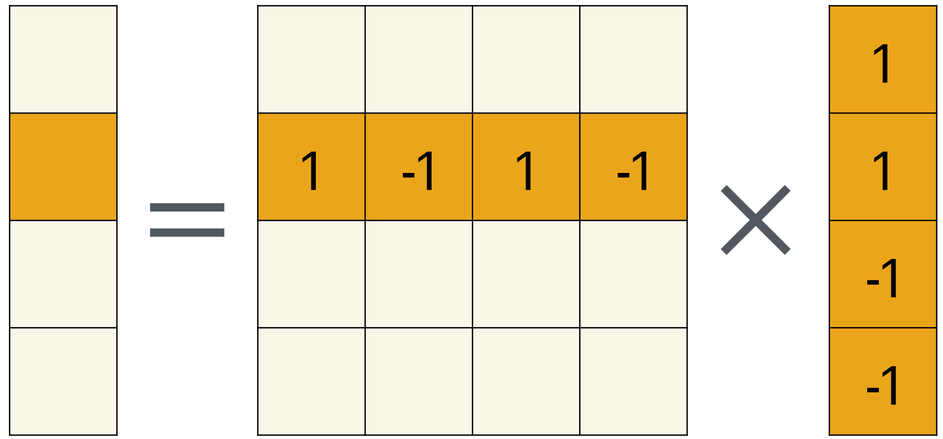
If Both Activations and Weights are Binarized

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

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

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





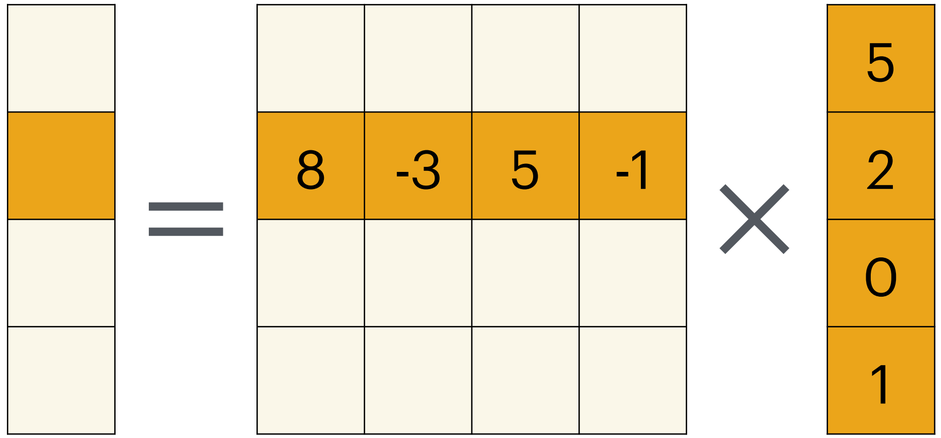
Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, September). Xnor-net: Imagenet classification using binary convolutional neural networks. In European conference on computer vision (pp. 525-542). Cham: Springer International Publishing.

If Both Activations and Weights are Binarized

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

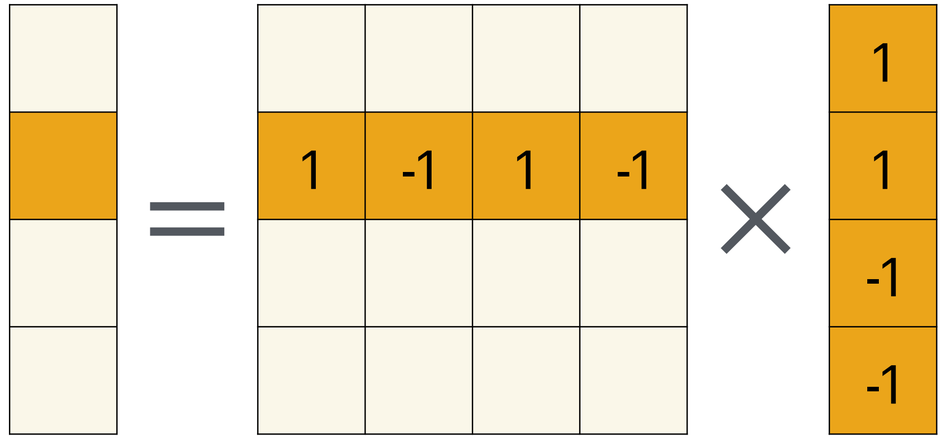
$$= 1 \times 1 + (-1) \times 1 + 1 \times (-1) + (-1) \times 1$$

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



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



Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, September). Xnor-net: Imagenet classification using binary convolutional neural networks. In European conference on computer vision (pp. 525-542). Cham: Springer International Publishing.

If Both Activations and Weights are Binarized

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

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

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

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

Assume everything is -1,

if one xnor result is 1, add 2 to the base:

1: -1 + 2 → -1 << 1

Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, September). Xnor-net: Imagenet classification using binary convolutional neural networks. In European conference on computer vision (pp. 525-542). Cham: Springer International Publishing.

If Both Activations and Weights are Binarized

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

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

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

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

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

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

+2

Base: -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

Assume everything is -1,

if one xnor result is 1, add 2 to the base:

1: -1 + 2 → -1 << 1

Result = base + 2 x #nonzeros

Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, September). Xnor-net: Imagenet classification using binary convolutional neural networks. In European conference on computer vision (pp. 525-542). Cham: Springer International Publishing.

If Both Activations and Weights are Binarized

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

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

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

popcount:
return the number of 1

🤩 **Hardware efficiency** 🤩
popcount
shift
addition

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

Assume everything is -1,

if one xnor result is 1, add 2 to the base:

1: -1 + 2 → -1 << 1

Result = base + 2 x #nonzeros

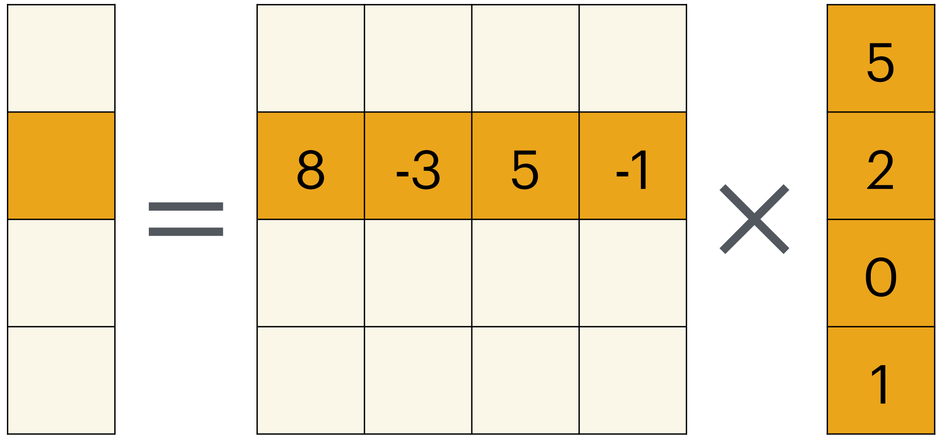
Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, September). Xnor-net: Imagenet classification using binary convolutional neural networks. In European conference on computer vision (pp. 525-542). Cham: Springer International Publishing.

If Weights are Quantized to +1 and -1

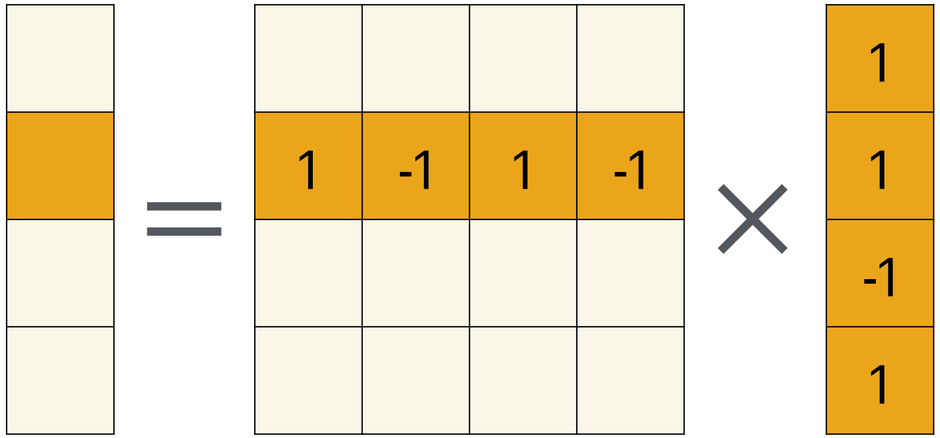
$$y_i = -n + 2 \cdot \sum_j W_{ij} \text{ xnor } x_j$$

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

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



Input	Weight	Operations	Memory	Computation
$\mathbb{R}$	$\mathbb{R}$	$+\times$	$1\times$	$1\times$
$\mathbb{R}$	$\mathbb{B}$	$+-$	~32x less	~2x less
$\mathbb{B}$	$\mathbb{B}$	xnor, popcount	~32x less	~58x less



Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, September). Xnor-net: Imagenet classification using binary convolutional neural networks. In European conference on computer vision (pp. 525-542). Cham: Springer International Publishing.

Accuracy Degradation of Binarization

Neural Network	Quantization	Bid-width		ImageNet Top-1 △Accuracy
		W	A	
AlexNet	BWN	1	32	0.2%
	BNN	1	1	-28.7%
	XNOR-Net	1	1	-12.4%

- 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

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830.
Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, September). Xnor-net: Imagenet classification using binary convolutional neural networks. In European conference on computer vision (pp. 525-542). Cham: Springer International Publishing.

Ternary Weight Networks

Weights are quantized to +1, -1, and 0

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

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

Ternary weights (2-bit)

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

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

X 1.5

$$= \frac{1}{11} \| W_{W^T \neq 0} \|_1$$

ImageNet Top-1 Accuracy on ResNet-18

Full Precision	1 bit (BWN)	2 bit (TWN)
69.6	60.8	65.3

Li, F., Liu, B., Wang, X., Zhang, B., & Yan, J. (2016). Ternary weight networks. arXiv preprint arXiv:1605.04711.

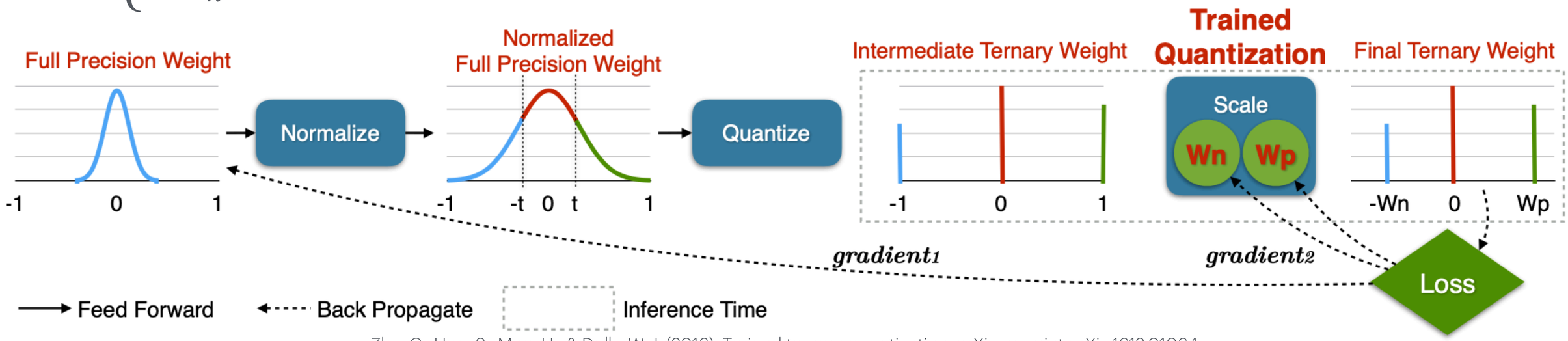
Trained Ternary Quantization

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

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

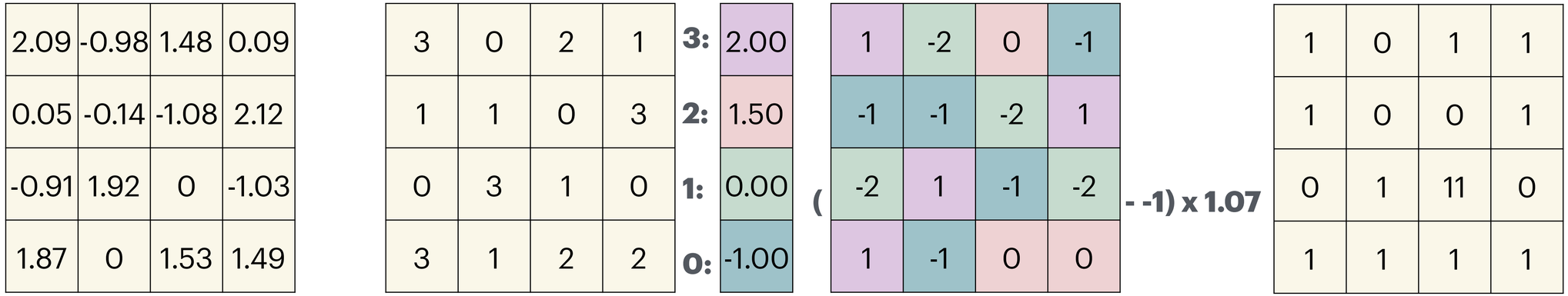
ImageNet Top-1 Accuracy on ResNet-18

Full Precision	1 bit (BWN)	2 bit (TWN)	TTQ
69.6	60.8	65.3	66.6



Zhu, C., Han, S., Mao, H., & Dally, W. J. (2016). Trained ternary quantization. arXiv preprint arXiv:1612.01064.

Neural Network Quantization



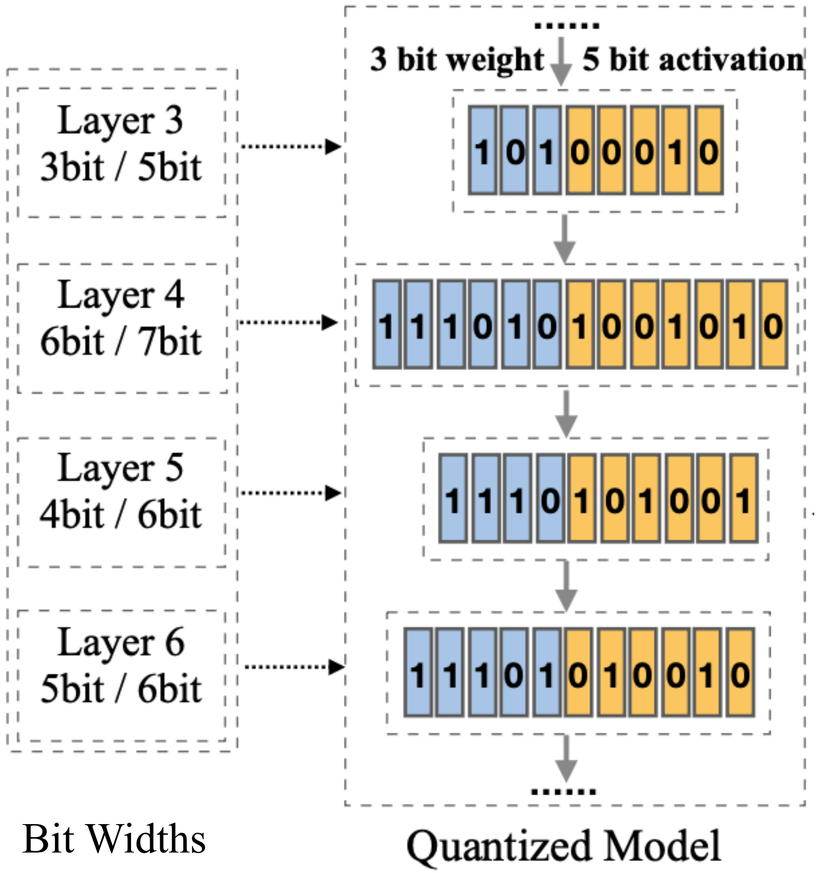
K-Means-based Quantization | **Linear Quantization** | **Binary/Ternary Quantization**

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

Mixed-Precision Quantization

Not all layers require the same number of bits.
Treat different layers differently to minimize the storage and maximize the efficiency.

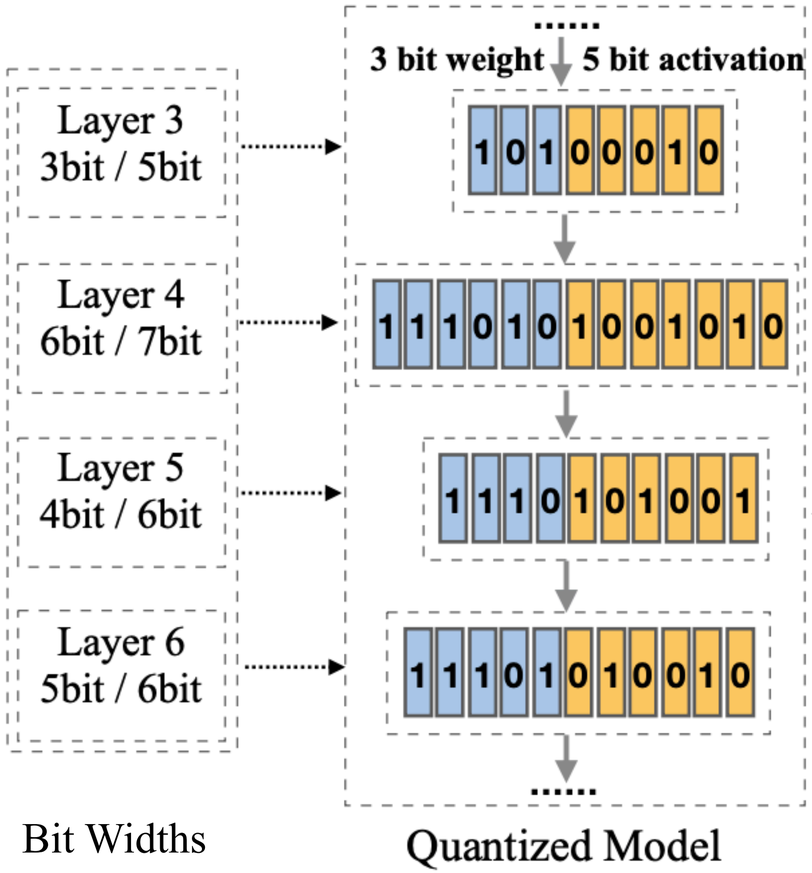
Mixed-Precision Quantization



If the choice for weight and activation are 1-8 bits, repsectively, what is the size of the design space?

Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). Haq: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 8612-8620).

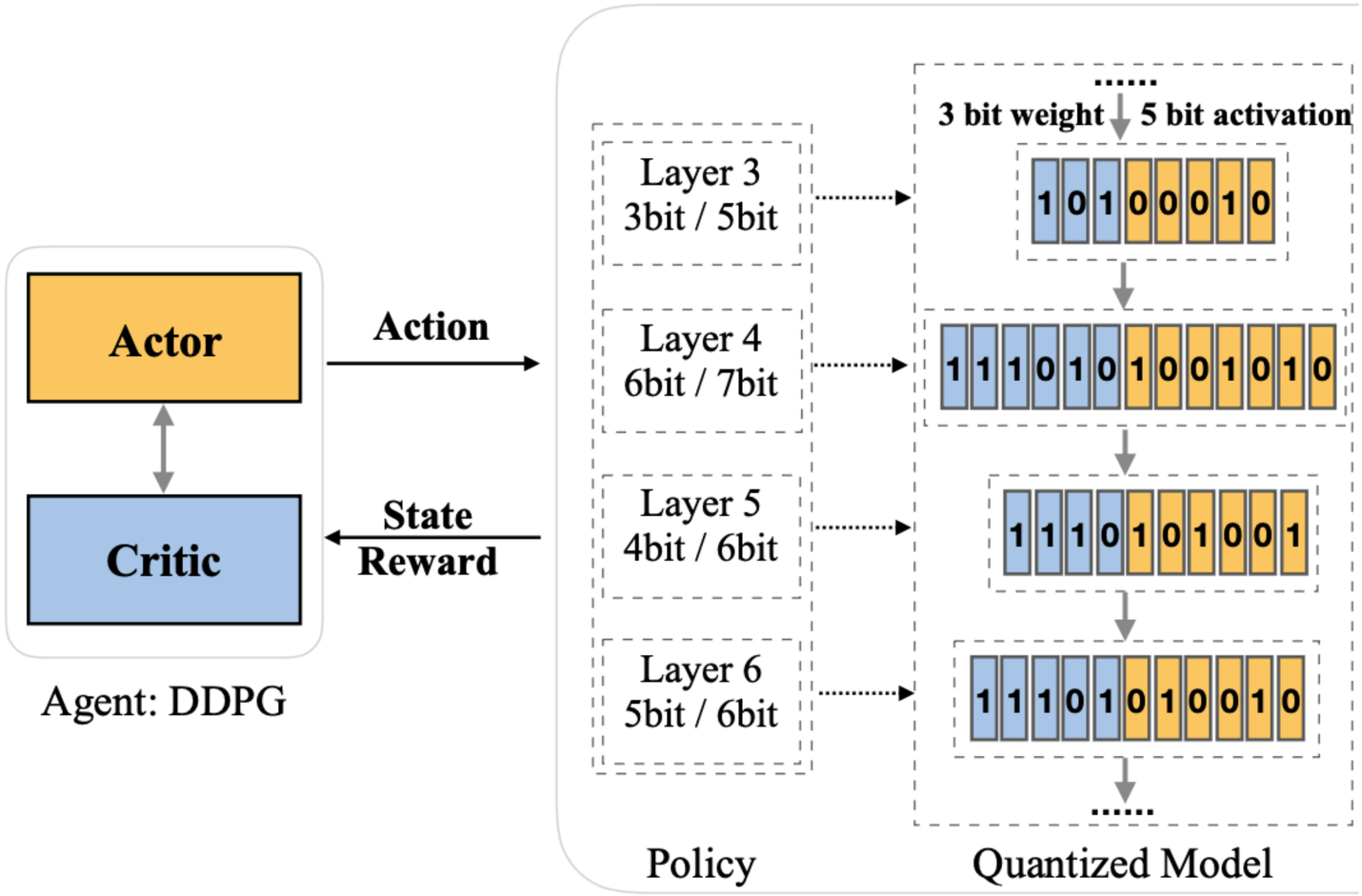
Challenge: Huge Design Space



- Each layers has $8 \times 8 = 64$ choices
- Design space is 64^n, n is the #layers

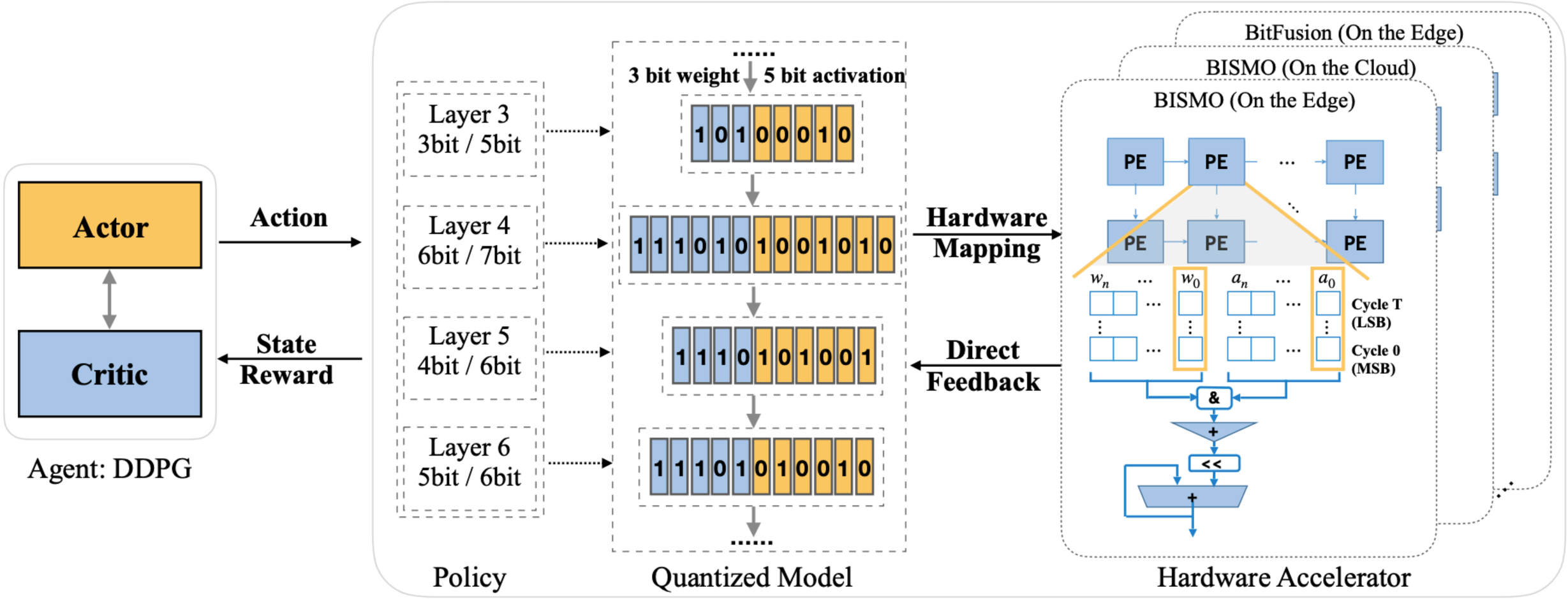
Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). Haq: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 8612-8620).

Solution: Design Automation



Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). Haq: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 8612-8620).

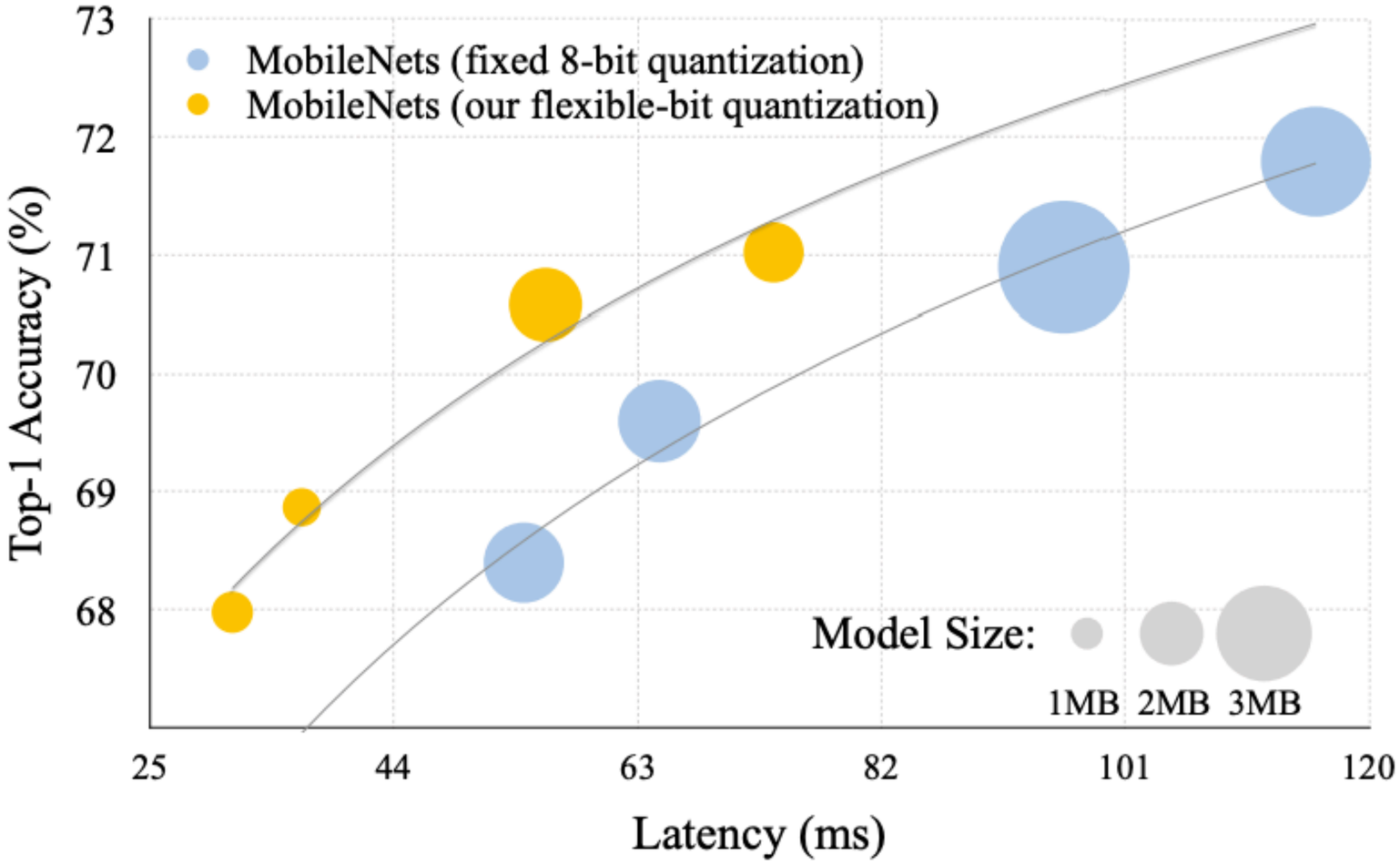
Solution: Design Automation



Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). Haq: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 8612-8620).

HAQ Outperforms Uniform Quantization

Hardware-aware quantization gets better trade-off between latency and accuracy.



Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). Haq: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 8612-8620).

HAQ Supports Multiple Objectives

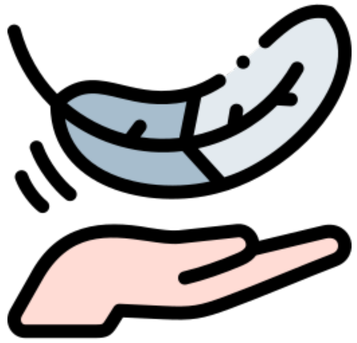
Mixed-precision quantized MobileNetV1

😖 **More engineer efforts.**

	Weights	Acc.-1	Acc.-5	Model Size
Han *et al.* [8]	2 bits	37.62	64.31	1.09 MB
Ours	*flexible*	**57.14**	**81.87**	1.09 MB
Han *et al.* [8]	3 bits	65.93	86.85	1.60 MB
Ours	*flexible*	**67.66**	**88.21**	1.58 MB
Han *et al.* [8]	4 bits	71.14	89.84	2.10 MB
Ours	*flexible*	**71.74**	**90.36**	2.07 MB
Original	32 bits	70.90	89.90	16.14 MB

	Weights	Activations	Acc.-1	Acc.-5	Latency
PACT [2]	4 bits	4 bits	62.44	84.19	7.86 ms
Ours	*flexible*	*flexible*	**67.45**	**87.85**	7.86 ms
PACT [2]	6 bits	4 bits	67.51	87.84	11.10 ms
Ours	*flexible*	*flexible*	**70.40**	**89.69**	11.09 ms
PACT [2]	6 bits	6 bits	70.46	89.59	19.99 ms
Ours	*flexible*	*flexible*	**70.90**	**89.95**	19.98 ms
Original	8 bits	8 bits	70.82	89.85	20.08 ms

	Weights	Activations	Acc.-1	Acc.-5	Energy
PACT [2]	4 bits	4 bits	62.44	84.19	13.47 mJ
Ours	*flexible*	*flexible*	**64.78**	**85.85**	13.69 mJ
PACT [2]	6 bits	4 bits	67.51	87.84	16.57 mJ
Ours	*flexible*	*flexible*	**70.37**	**89.40**	16.30 mJ
PACT [2]	6 bits	6 bits	70.46	89.59	26.80 mJ
Ours	*flexible*	*flexible*	**70.90**	**89.73**	26.67 mJ
Original	8 bits	8 bits	70.82	89.95	31.03 mJ

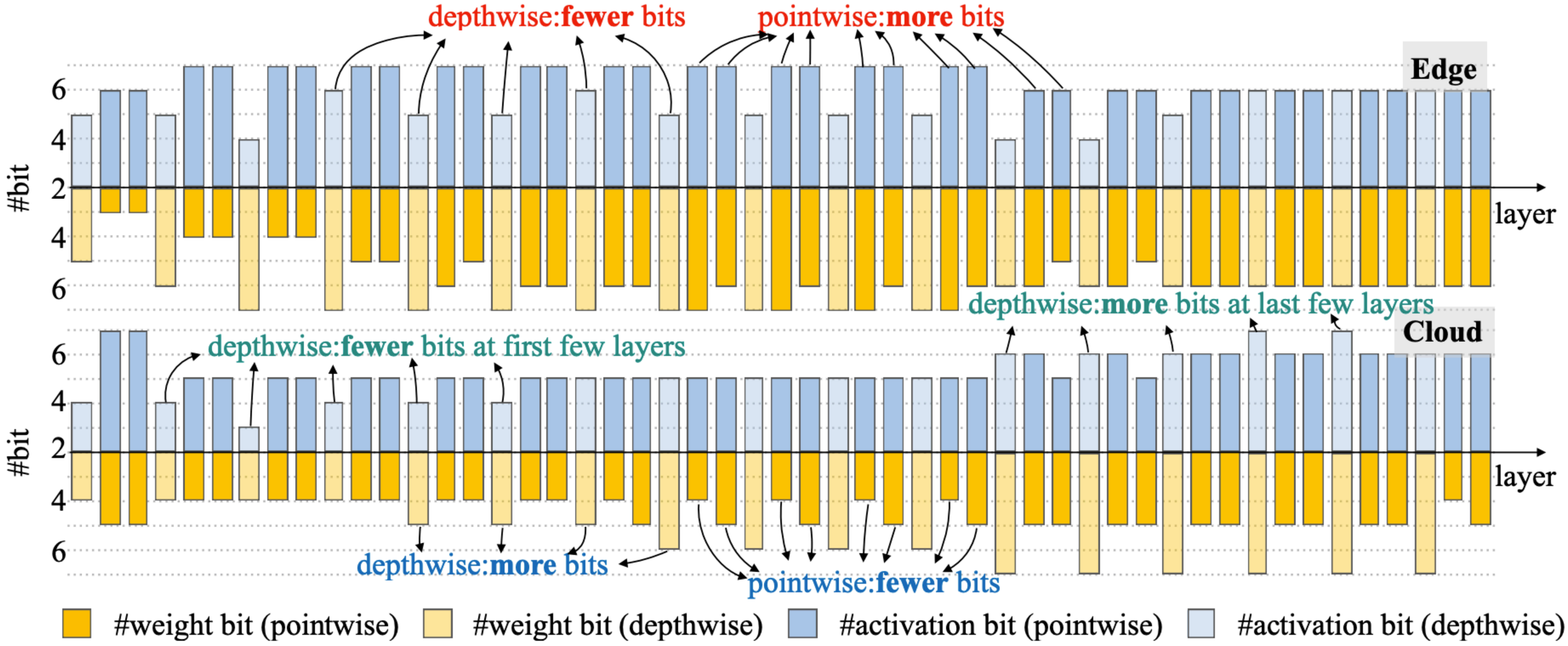


- In practice, industry uses coarse grained precision to balance between the engineering complexity vs. the performance. For example, one precision for all Conv layers, one precision for all FC layers.

Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). Haq: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 8612-8620).

Quantization Policy for Edge and Cloud

Quantization policy under latency constraints for MobileNet-V2



Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). Haq: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 8612-8620).

Lab 3 is out

Mid-term Survey is out

Get your bonus!

Reference

- Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.
- Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2704-2713).
- Nagel, M., Baalen, M. V., Blankevoort, T., & Welling, M. (2019). Data-free quantization through weight equalization and bias correction. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1325-1334).
- Dai, S., Venkatesan, R., Ren, M., Zimmer, B., Dally, W., & Khailany, B. (2021). Vs-quant: Per-vector scaled quantization for accurate low-precision neural network inference. Proceedings of Machine Learning and Systems, 3, 873-884.
- Darvish Rouhani, B., Zhao, R., Elango, V., Shafipour, R., Hall, M., Mesmakhosroshahi, M., ... & Naumov, M. (2023, June). With shared microexponents, a little shifting goes a long way. In Proceedings of the 50th Annual International Symposium on Computer Architecture (pp. 1-13).
- Banner, R., Nahshan, Y., & Soudry, D. (2019). Post training 4-bit quantization of convolutional networks for rapid-deployment. Advances in Neural Information Processing Systems, 32.
- Nagel, M., Amjad, R. A., Van Baalen, M., Louizos, C., & Blankevoort, T. (2020, November). Up or down? adaptive rounding for post-training quantization. In International Conference on Machine Learning (pp. 7197-7206). PMLR.
- Hinton, G., Srivastava, N., & Swersky, K. (2012). Neural networks for machine learning. Coursera, video lectures, 264(1), 2146-2153.
- Bengio, Y., Léonard, N., & Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432.
- Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342.
- BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations
- Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830.
- Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, September). Xnor-net: Imagenet classification using binary convolutional neural networks. In European conference on computer vision (pp. 525-542). Cham: Springer International Publishing.
- Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830.
- Zhu, C., Han, S., Mao, H., & Dally, W. J. (2016). Trained ternary quantization. arXiv preprint arXiv:1612.01064.
- Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). Haq: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 8612-8620).
- Neural Networks for Machine Learning [Hinton et al., Coursera Video Lecture, 2012]
- Quantization [MIT 6.5940]