Lecture 05: DNN Quantization

Recap

- Why pruning?
  - Running cost of CNNs and Transformers
- Sparse matrix encoding
- General pruning techniques
- Transformer pruning
- Large model pruning

Topics

- Basic Data Formats
  - Fixed point (INT)
  - Floating point (FP)
  - Block floating point (BFP)
- Quantization methods
  - Taxonomy of Quantization
  - Learnable adaptive quantization scheme
  - Quantization for LLM

NYU SAI LAB

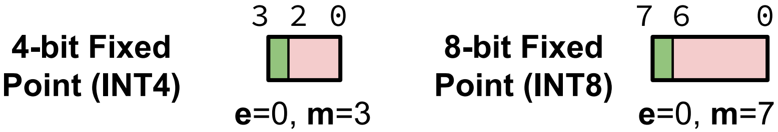
Topics

- Basic Data Formats
  - Fixed point (INT)
  - Floating point (FP)
  - Block floating point (BFP)
- Quantization methods
  - Taxonomy of Quantization
  - Learnable adaptive quantization scheme
  - Quantization for LLM

NYU SAI LAB

Fixed-Point Arithmetic (INT)

Fixed Point Formats

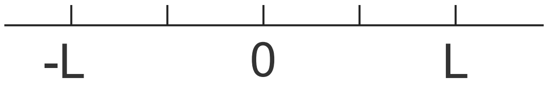


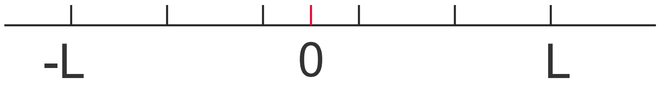
- Hyperparameter associated with the fixed-point format:
  - Clipping range (-L, L): usually symmetrical around 0
  - Bitwidth (b)
- Quantization with Fixed-point format is called **Fixed point quantization** or **INT quantization**.

NYU SAI LAB

Fixed-Point Format (Symmetrical)

- How to convert a number x to INT representation?
  - Set the clipping range: (-L, L), bitwidth: b
  - Compute the scale: $s = 2L/(2^b - 2)$
  - Clip the input x: $x_c = Clip(x, L, -L)$
  - Calculate the INT representation: $x_{int} = round(x_c/s)$
  - Rescale: $x_q = s x_{int}$
- Have a uniform representation power within the clipping range.
- All the computations can be performed using x_{int}





With $s=2L/(2^b-2)$, zero can be represented using quantized number

Example

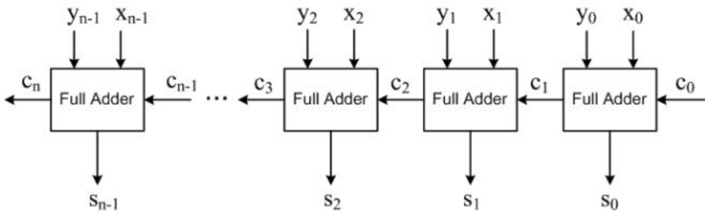
- X = [1.1, 2.4, -0.3, 0.8], bitwidth = 3, L = 2
- How to convert a number x to INT representation?
  - Set the clipping range: (-L, L), bitwidth: b   b=3, L=2
  - Compute the scale: $s = 2L/(2^b - 2)$ s = 4/6 = 2/3
  - Clip the input x: $x_c = Clip(x, L, -L)$ xc = [1.1, 2, -0.3, 0.8]
  - Calculate the INT representation: $x_{int} = round(x_c/s)$ xint = [2, 3, 0, 1]
  - Rescale: $x_q = sx_{int}$ Xq = [1.33, 2.0, 0.0, 0.67]

NYU SAI LAB

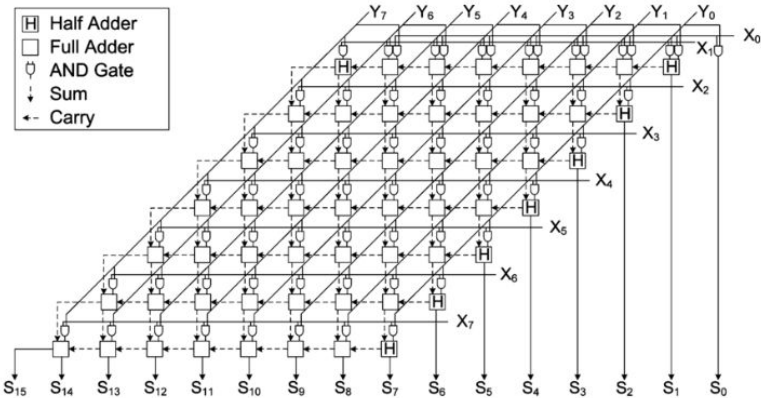
Computation with Fixed-Point Format

- Addition/Subtraction: $x_q \pm y_q = s(x_{int} \pm y_{int})$
- Multiplication: $x_q \times y_q = s^2(x_{int} \times y_{int})$
- Division: $x_q / y_q = x_{int} / y_{int}$

If the scales are the same



Fixed-point adder

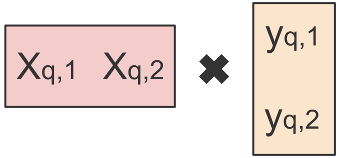


Fixed-point multiplier

NYU SAI LAB

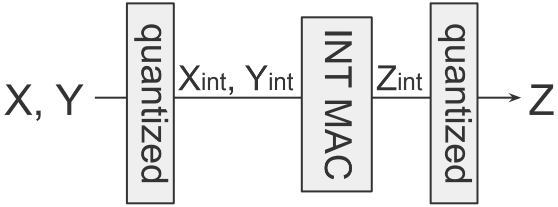
Computation with Fixed-Point Format

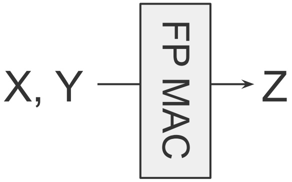
- If we try to compute the dot product between X and Y:



All elements within the tensors are quantized using the same scale

$$x_{q,1} \times y_{q,1} + x_{q,2} \times y_{q,2} = s_x s_y (x_{int,1} \times y_{int,1} + x_{int,2} \times y_{int,2})$$

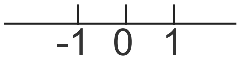




NYU SAI LAB

Computation with Fixed-Point Format

Binary quantization

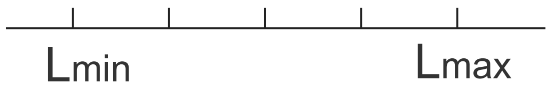


Ternary quantization

- Binary and Ternary neural networks are both multiplication-free DNN.

Fixed Point Format (Unsymmetrical)

- How to convert a number to INT8 representation?
  - Set the clipping range: (L_{min}, L_{max}), bitwidth: b
  - Compute the scale: $s = (L_{max} - L_{min})/(2^b - 1)$
  - Clip the input x: $x_c = Clip(x, L_{min}, L_{max})$
  - Calculate the fixed-point representation: $x_{int} = round((x_c - L_{min})/s)$
  - Rescale: $x_q = s x_{int} + L_{min}$



Example

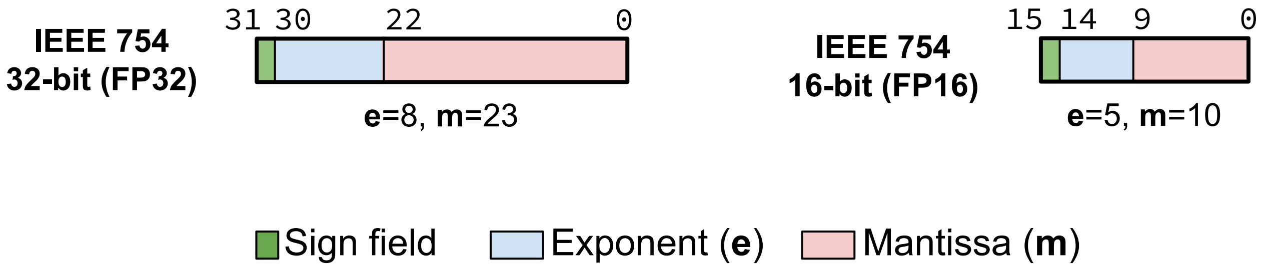
- X = [1.1, 2.4, -0.3, 0.8], bitwidth = 3, L = 2
- How to convert a number to INT8 representation?
  - Set the clipping range: (L_{min}, L_{max}), bitwidth: b   b=3, L_{max}=2, L_{min}=-0.5
  - Compute the scale: $s = (L_{max} - L_{min})/(2^b - 1)$ s = 0.357
  - Clip the input x: $x_c = Clip(x, L_{min}, L_{max})$ X_c = [1.1, 2, -0.3, 0.8]
  - Calculate the fixed-point representation: $x_{int} = round((x_c - L_{min})/s)$ X_{int} = [4,7,1,4]
  - Rescale: $x_q = s x_{int} + L_{min}$ X_q = [0.93, 2.0, -0.14, 0.93]

NYU SAI LAB

Computation with Fixed-Point Format

- Addition/Subtraction:
  $x_q + y_q = s(x_{int} + y_{int}) + 2L_{min}$ $\quad$ $x_q - y_q = s(x_{int} - y_{int})$
- Multiplication (needs additional computation):
  $x_q \times y_q = s_x s_y(x_{int} \times y_{int}) + L_{min,x} y_q s_y + L_{min,y} x_q s_x + L_{min,x} L_{min,y}$
- Division: hard to implement

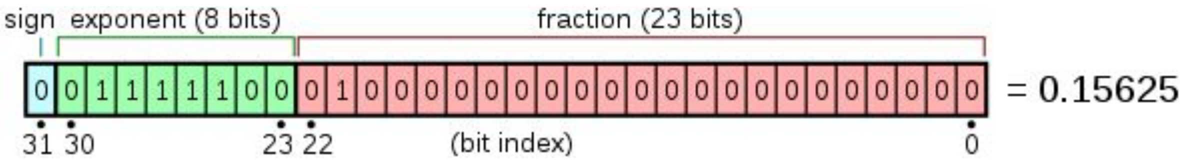
Floating-Point Arithmetic



- The floating-point number has three fields:
  - Sign (s)
  - Exponent (e)
  - Mantissa (m)

NYU SAI LAB

Floating-Point Arithmetic



- Every real number can be converted in the following format:

$$x = (-1)^s \times 2^{e-bias} \times m \quad where\ 1 \leq m < 2$$
$$m = (1.b_0 b_1 b_2 ... b_{22})_2$$

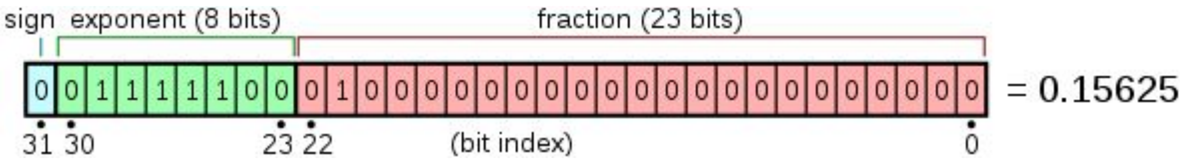
There typically exists a predefined bias: bias = 127 for IEEE 754 FP32.

- For example:
  - $5.5 = (-1)^0 \times 2^{129\text{-}127} \times (1.011)_2$ — s = 0, e = 10000001, m = 0110000…0
  - $-71 = (-1)^1 \times 2^{133\text{-}127} \times (1.000111)_2$ — s = 1, e = 10000101, m = 0001110…0
  - $0.34375 = (-1)^1 \times 2^{125\text{-}127} \times (1.011)_2$ — s = 1, e = 01111101, m = 0110000…0

NYU SAI LAB

Overton, Michael L. "Floating point representation." *Unpublished note* (1996).

Floating-Point Arithmetic



- IEEE-754 standard:

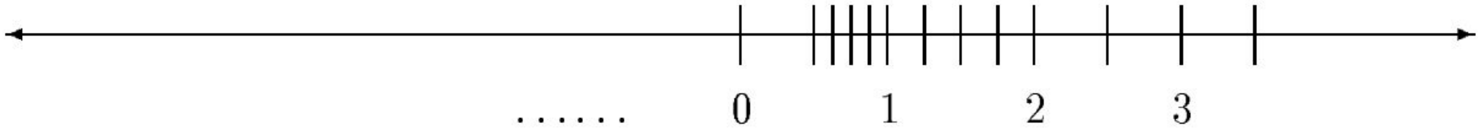
$$x = (-1)^s \times 2^{e-bias} \times m \quad where\ 1 \leq m < 2$$

$$m = (1.b_0 b_1 b_2 ... b_{22})_2$$

- The exponent field is unsigned.
- We need some special representation:
  - A bit stream of all zeros represents 0

NYU SAI LAB

Floating Point Arithmetic



- Have better representation power for values with small magnitudes.
- How to convert a real number x to FP representation?

x = |x| s = sign(x)

$$a = \lfloor log_2 x \rfloor \quad e = a + bias \quad m = \frac{x}{2^a} - 1$$

NYU SAI LAB

Example

x = -13.24, bias=127

x = |x| s = sign(x)

$$a = \lfloor log_2 x \rfloor \quad e = a + bias \quad m = \frac{x}{2^a} - 1$$

a = 3, e = 130, m = 0.655

s = $(0)_2$, e = $(10000010)_2$, m = $(10100111101011100001000)_2$

NYU SAI LAB

Computation with FP Representation

- Addition/Subtraction:
  - Need to align the exponent

    011010 + 001111 = 011010 + 011001 = 011011

    s_1 e_1 m_1 s_2 e_2 m_2 Alignment

- Multiplication/Subtraction:
  - Sum the exponent, multiply the mantissa

    011010 ✖ 001111

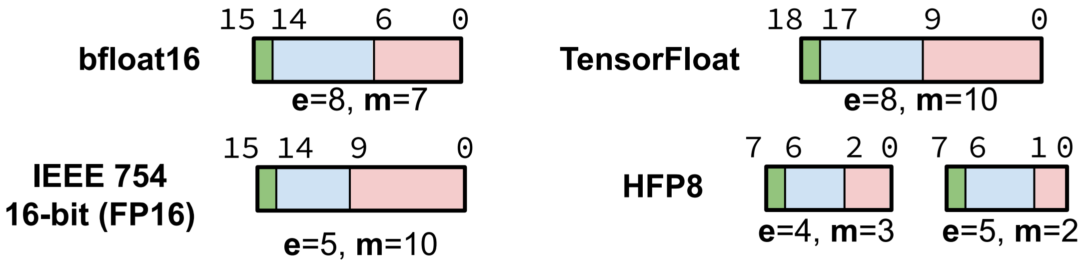
    s_1 e_1 m_1 s_2 e_2 m_2

    $e = e_1 + e_2$

    $m = 1.m_1 \times 1.m_2$

- Addition and subtraction is expensive for FP.

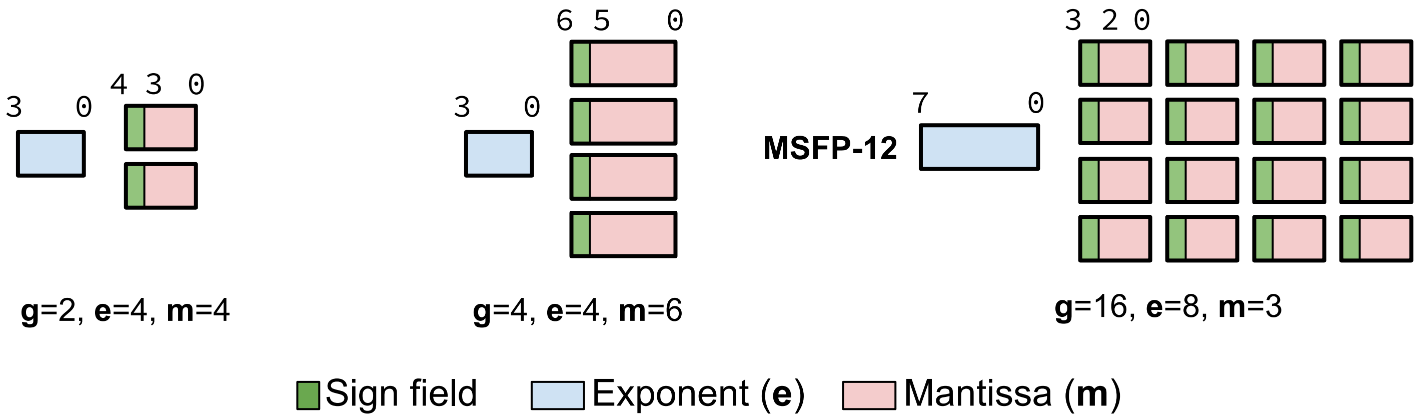
Customized FP Representation



- Numerous customized FP representations have been developed to facilitate DNN execution.

NYU SAI LAB

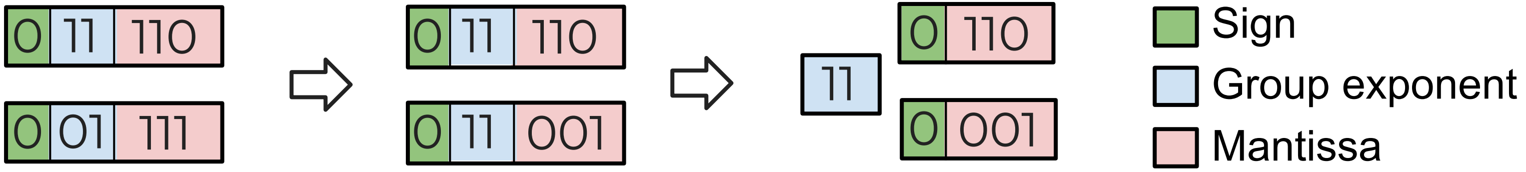
Block Floating Point (BFP)



- BFP formats offer a middle ground between FP and INT formats, by enforcing that a group of values share a common exponent while maintaining individual mantissas.

NYU SAI LAB

Block-Floating Arithmetics (BFP)



- Block floating point (BFP) is a numerical representation method that applies a shared exponent to a block of fixed-point values, balancing precision and dynamic range while reducing computational complexity compared to full floating-point arithmetic.
- There is no “leading 1”.

$$x = (-1)^s \times 2^{e-bias} \times m \quad where\ 1 \leq m < 2$$
$$m = (1.b_0 b_1 b_2 \ldots b_{22})_2$$

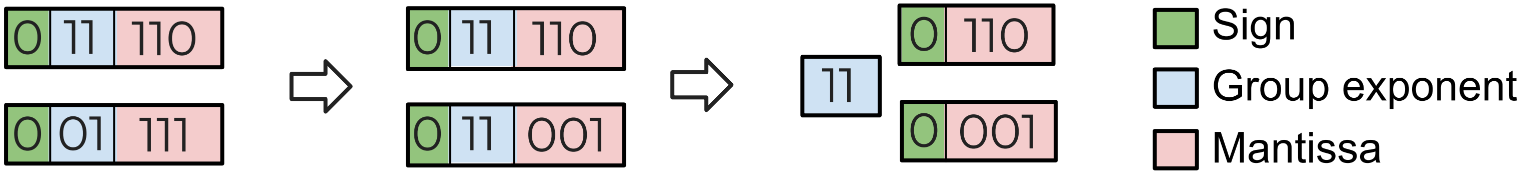
FP

$$x = (-1)^s \times 2^{e-bias} \times m$$
$$m = (b_0.b_1 b_2 b_3 \ldots b_{22})_2$$

BFP

NYU SAI LAB

Block-Floating Arithmetics (BFP)



- Inner-group operations are performed using fixed-point arithmetic.
- Cross-group operations are performed using floating-point arithmetic.
- Each group exponent also includes a bias, which is shared across all the groups.

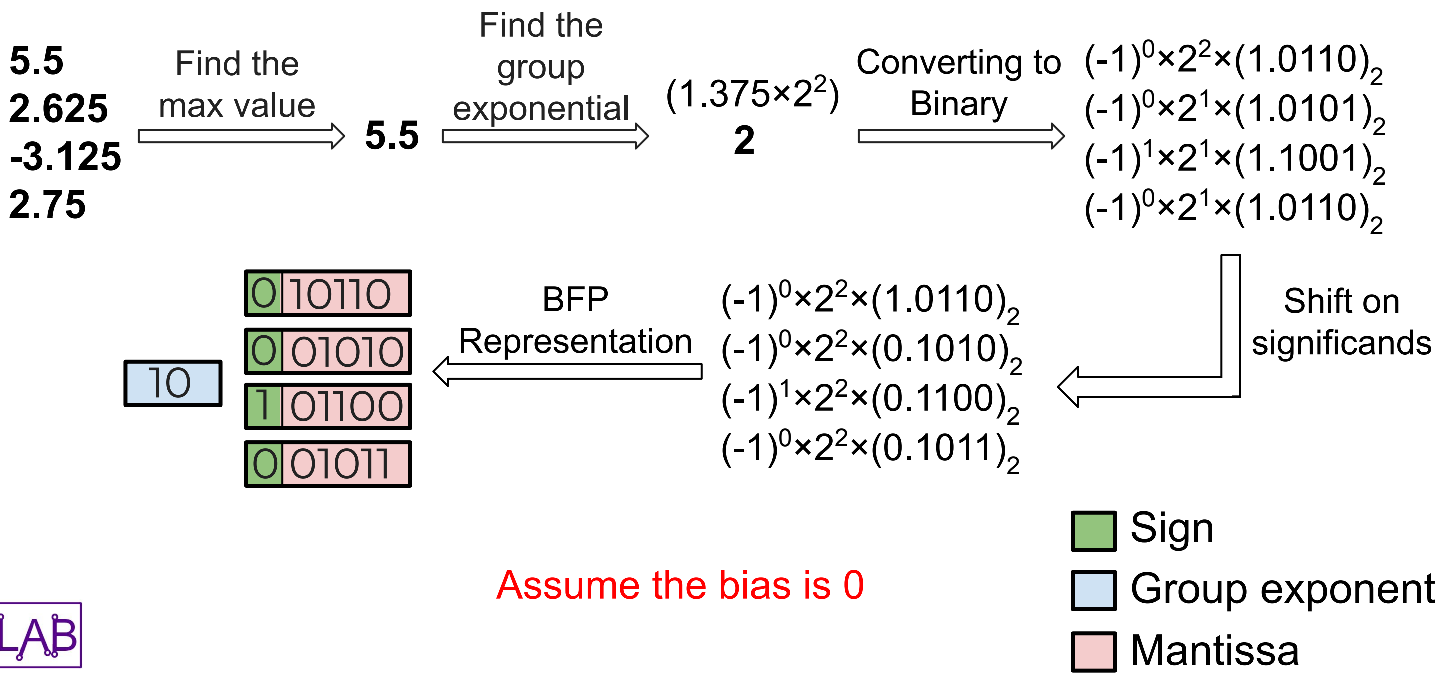
$$x = (-1)^s \times 2^{e-bias} \times m \quad where\ 1 \leq m < 2$$
$$m = (1.b_0 b_1 b_2 \ldots b_{22})_2$$

FP

$$x = (-1)^s \times 2^{e-bias} \times m$$
$$m = (b_0.b_1 b_2 b_3 \ldots b_{22})_2$$

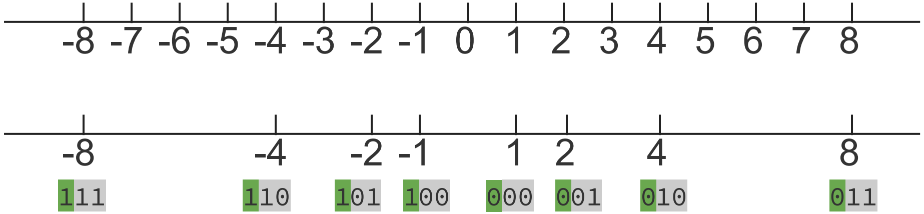
BFP

Example


5.5
2.625
-3.125
2.75
Find the max value
5.5
Find the group exponential
(1.375×2^2)
2
Converting to Binary
$(-1)^0\times2^2\times(1.0110)_2$
$(-1)^0\times2^1\times(1.0101)_2$
$(-1)^1\times2^1\times(1.1001)_2$
$(-1)^0\times2^1\times(1.0110)_2$
Shift on significands
$(-1)^0\times2^2\times(1.0110)_2$
$(-1)^0\times2^2\times(0.1010)_2$
$(-1)^1\times2^2\times(0.1100)_2$
$(-1)^0\times2^2\times(0.1011)_2$
BFP Representation
10
0 10110
0 01010
1 01100
0 01011
Assume the bias is 0
Sign
Group exponent
Mantissa

Logarithm Arithmetics

- A specialized form of integer (INT) quantization
- Utilizes only power-of-two integer values, making hardware multiplication more efficient and cost-effective.



- Each INT number can be represented by its exponent value.
- A total of 8 numbers, 3 bits are needed to encode the bits.

$a = (1100)_2$ $a \times 2 = (11000)_2$ $a \times 8 = (1100000)_2$

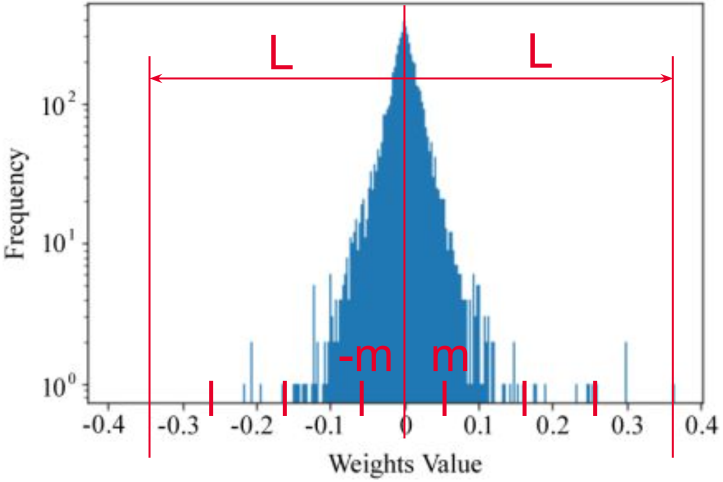
Topics

- Basic Data Formats
  - Fixed point (INT)
  - Floating point (FP)
  - Block floating point (BFP)
- Quantization methods
  - Taxonomy of Quantization
  - Learnable adaptive quantization scheme
  - Quantization for LLM

Taxonomy of Quantization

- Quantization techniques can be classified from different perspectives:
  - Weight quantization, activation quantization
  - Quantization aware training, post training quantization
  - Tensor-based quantization, vector-based quantization, group-based quantization
  - Quantization for inference/training
  - Deterministic quantization, stochastic quantization

Weight Quantization

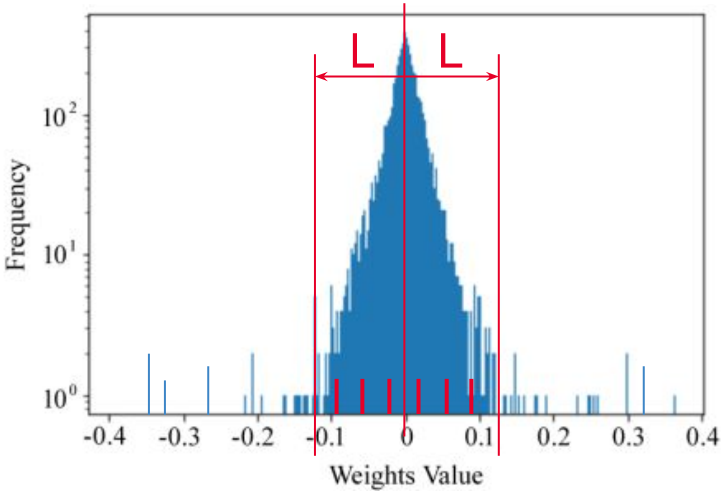


Weight distribution in ResNet

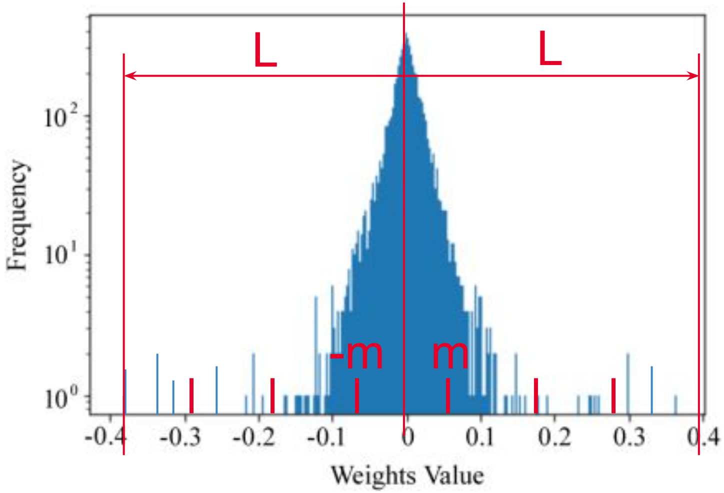
- The weight distribution follows a gaussian-like distribution.
- The outlier will lead to large quantization error.
- A good selection on the clip range L is critical for accuracy performance.

NYU SAI LAB

Weight Quantization



- Large truncation error
- Low quantization error for small values



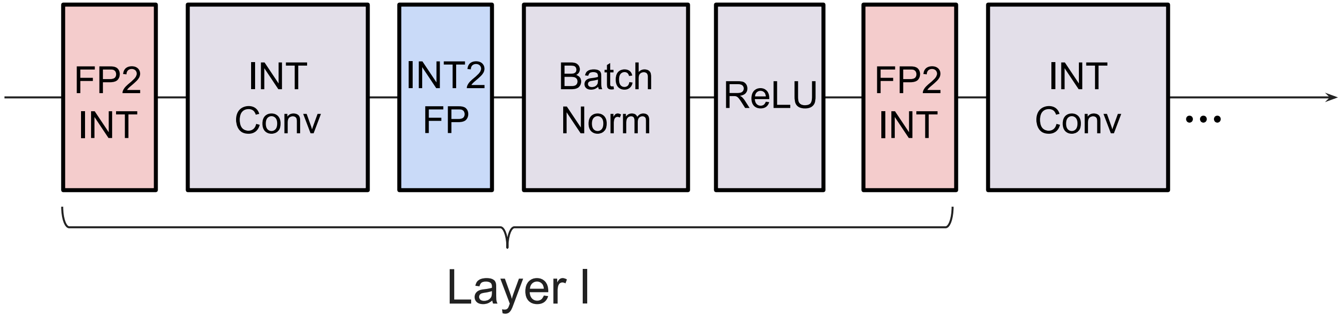
- Small truncation error
- Large quantization error for small values

- L = 0.9×max(|W|), L = 0.95×max(|W|)

NYU SAI LAB

Activation Quantization

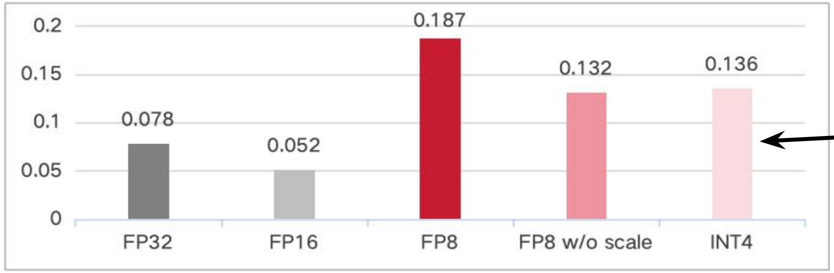
- Quantization on activation needs to be performed dynamically. This will introduce additional compute overhead.
- Also the activation will pass the nonlinear functions, dequantization is required.



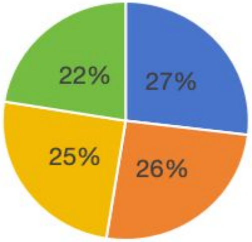
Activation Quantization

$(577 \times 1024) \times (1024 \times 1024)$

Projection Layer:
Input: 577x1024
Weight: 4096x1024



On 4090 GPU



- For low-precision quantization, the quantization process may cause more computation than the computational savings achieved by using low-precision quantization.

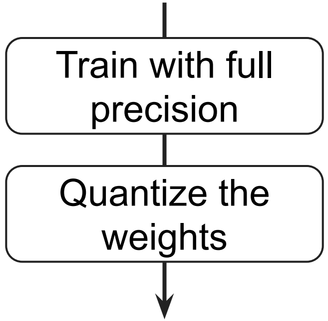
NYU SAI LAB

Taxonomy of Quantization

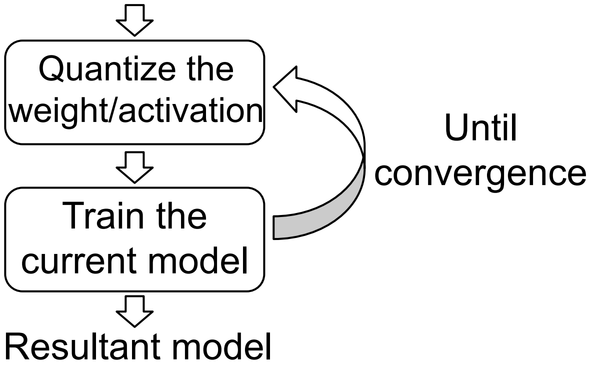
- Quantization techniques can be classified from different perspectives
  - Weight quantization, activation quantization
  - Post training quantization, quantization aware training
  - Tensor-based quantization, vector-based quantization, group-based quantization
  - Quantization for inference/training
  - Deterministic quantization, stochastic quantization

When to Quantize?

Post-training quantization (PTQ)



Quantization-aware Training (QAT)

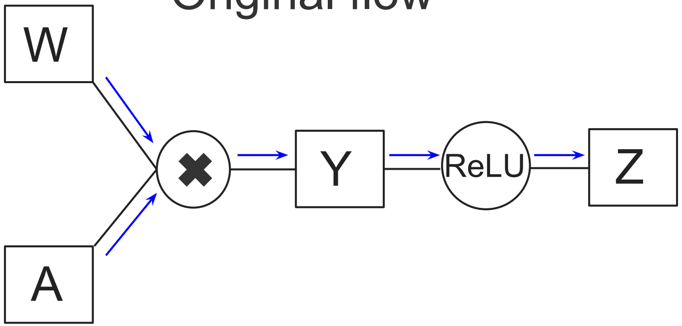


- PTQ has lower computational cost, but accuracy is also lower.
- For the model which is expensive to train (LLM), PTQ is applied to facilitate their implementations.

NYU SAI LAB

Another Way to Look at Quantization

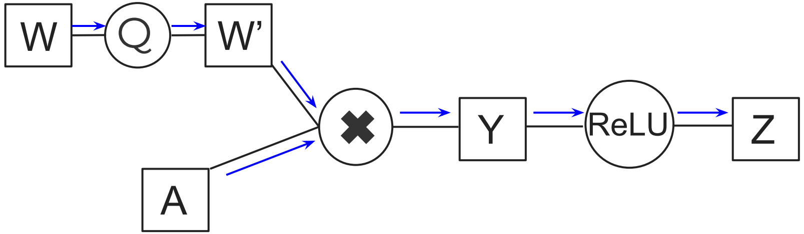
Original flow



Y = WA, Z = ReLU(Y)

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Z}\frac{\partial Z}{\partial Y}\frac{\partial Y}{\partial W}$$

Flow with quantization

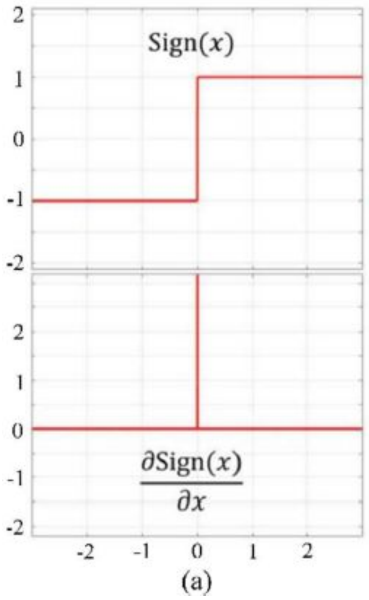


$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Z}\frac{\partial Z}{\partial Y}\frac{\partial Y}{\partial W'}\frac{\partial W'}{\partial W}$$

How to compute $\frac{\partial W'}{\partial W}$?

NYU SAI LAB

Straight Through Estimator (STE)



- Staircase function has a derivative of 0 at most of the values. This will makes the DNN not trainable.
- We instead use STE to estimate the gradient of a non-differentiable quantized function in the backward pass.

$$\frac{\partial W'}{\partial W} = 1$$

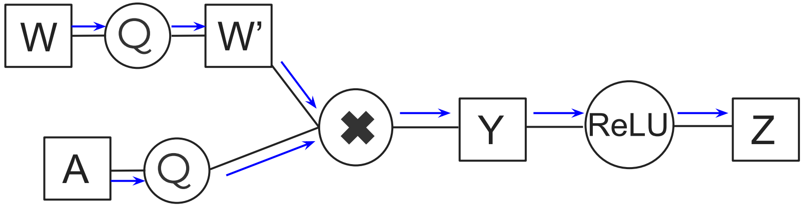
- During the forward pass, apply quantization, for backprop, ignore it.

NYU SAI LAB

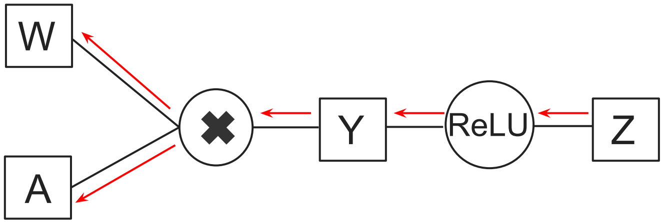
Li, Hao, et al. "Training quantized nets: A deeper understanding." *Advances in Neural Information Processing Systems* 30 (2017).

Straight Through Estimator (STE)

Forward pass



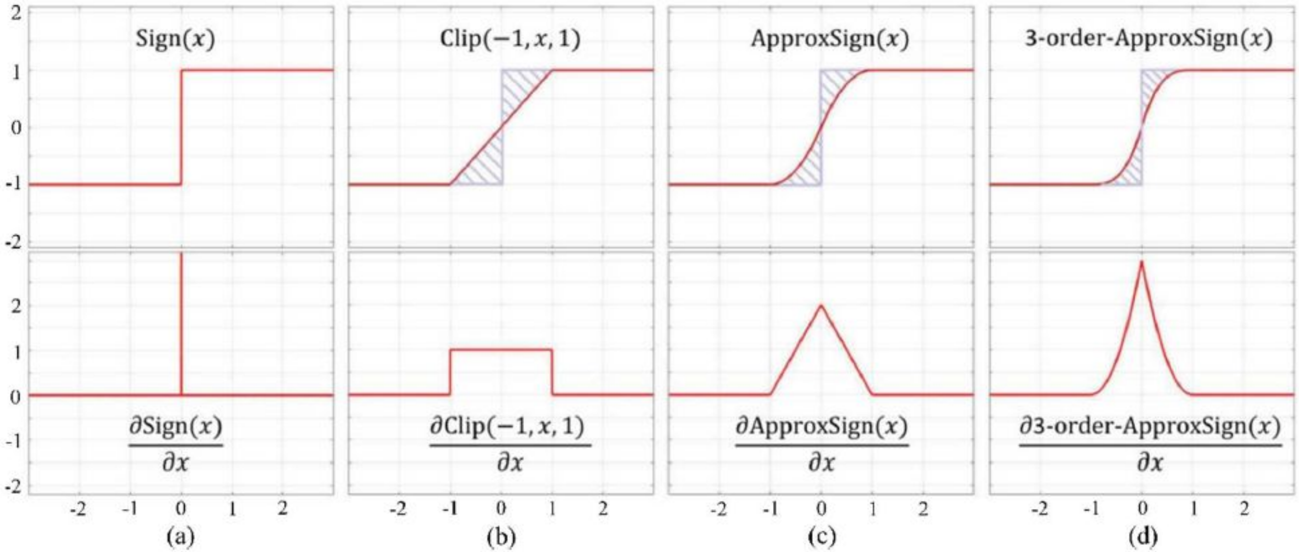
Backward pass



- During the forward pass, apply quantization, for backprop, ignore it.

NYU SAI LAB

Other Ways to Approximate Quantization



Liu, Zechun, et al. "Bi-real net: Binarizing deep network towards real-network performance." International Journal of Computer Vision 128 (2020): 202-219.

Pytorch Implementation of Quantization

```
def forward(self, x):
    y = F.conv2d(self.w, x)
    return y
```

```
def forward(self, x, b, L):
    self.quantized_w = Q(self.w, b, L)
    y = F.conv2d(self.quantized_w, x)
    return y
def Q(w, b, L):
    L = 0.9 * w.abs().max()
    w = torch.clip(w, min=-L, max=L)
    scale = 2L/(2**b-2)
    wq = (w/scale).round() * scale
    return wq
```

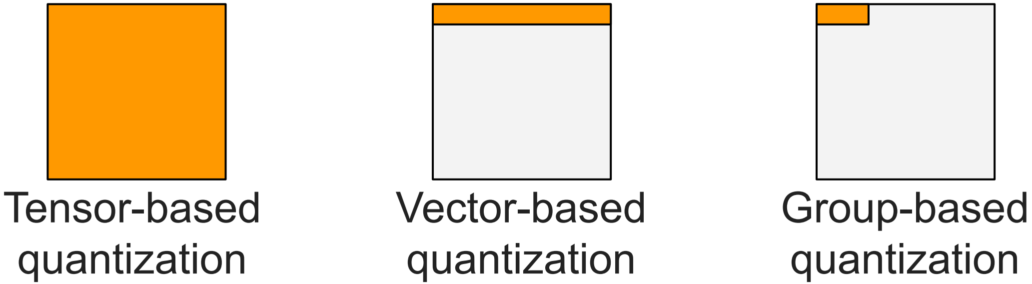
NYU SAI LAB

Taxonomy of Quantization

- Quantization techniques can be classified from different perspectives
  - Weight quantization, activation quantization
  - Post training quantization, quantization aware training
  - Tensor-based quantization, vector-based quantization, group-based quantization
  - Quantization for inference/training
  - Deterministic quantization, stochastic quantization

Granularity of Quantization

- The weight can be quantized with different granularity:
  - Tensor-based quantization
  - Vector-based quantization
  - Group-based quantization
- A higher quantization granularity will lead to a lower quantization error and a higher hardware implementation cost.



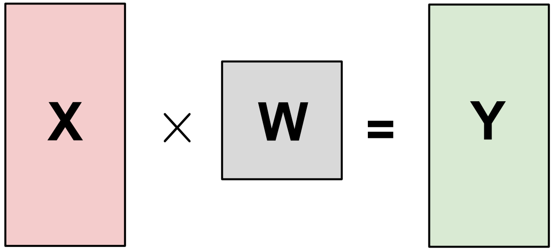
NYU SAI LAB

Taxonomy of Quantization

- Quantization techniques can be classified from different perspectives
  - Weight quantization, activation quantization
  - Post training quantization, quantization aware training
  - Tensor-based quantization, vector-based quantization, group-based quantization
  - Quantization for inference/training
  - Deterministic quantization, stochastic quantization

NYU SAI LAB

Quantization During Training



X: input **W**: weight filters **Y**: output

- The forward propagation is very similar to the inference operation, where the input X is multiplied by weight W, generating the output Y.

NYU SAI LAB

Quantization During Training

Data gradient Computation

$$\nabla Y \times W^T = \nabla X$$

Weight gradient Computation

$$X^T \times \nabla Y = \nabla W$$

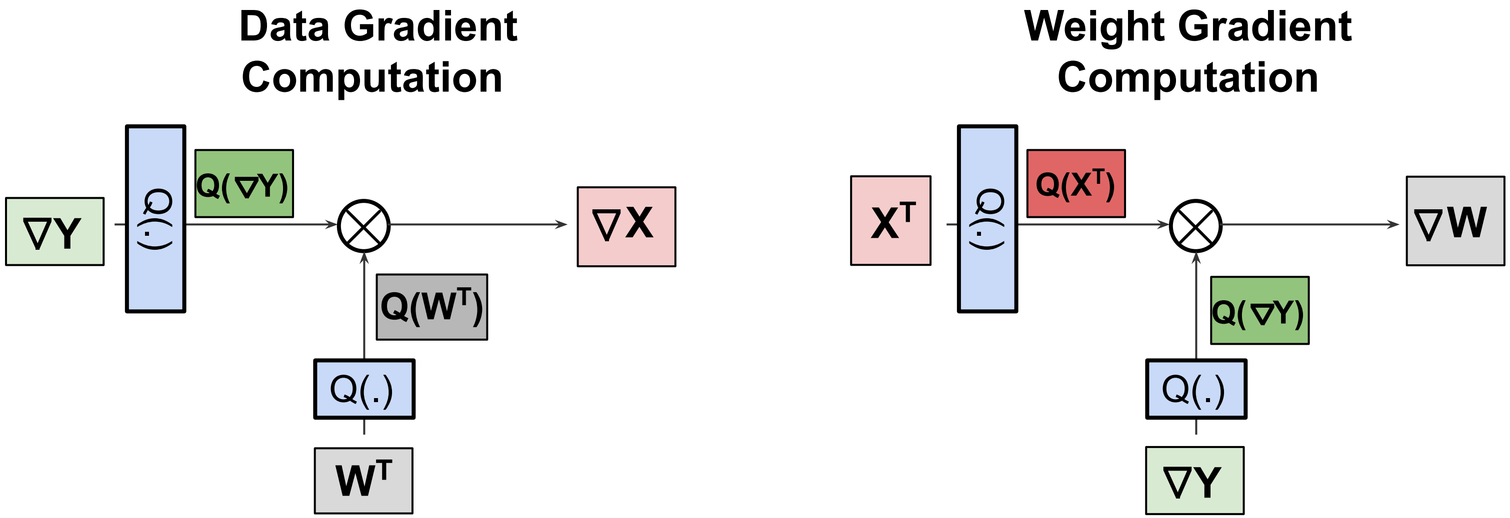
X: input
∇X: input gradient

W: weight filters
∇W: weight gradient

Y: output
∇Y: output gradient

NYU SAI LAB

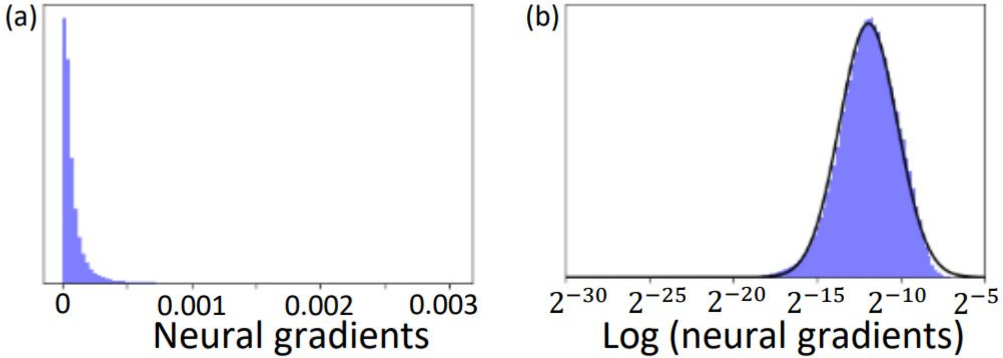
Quantization During Training



- Gradient is much more sensitive to quantization error.

NYU SAI LAB

DNN Gradient Distribution



- DNN gradient is much hard to quantize and very sensitive to quantization error.

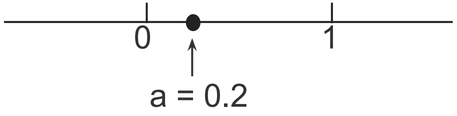
Chmiel, Brian, et al. "Neural gradients are near-lognormal: improved quantized and sparse training." *arXiv preprint arXiv:2006.08173* (2020).

Taxonomy of Quantization

- Quantization techniques can be classified from different perspectives
  - Weight quantization, activation quantization
  - Post training quantization, quantization aware training
  - Tensor-based quantization, vector-based quantization, group-based quantization
  - Quantization for inference/training
  - Deterministic quantization, stochastic quantization

NYU SAI LAB

Deterministic and Stochastic Quantization



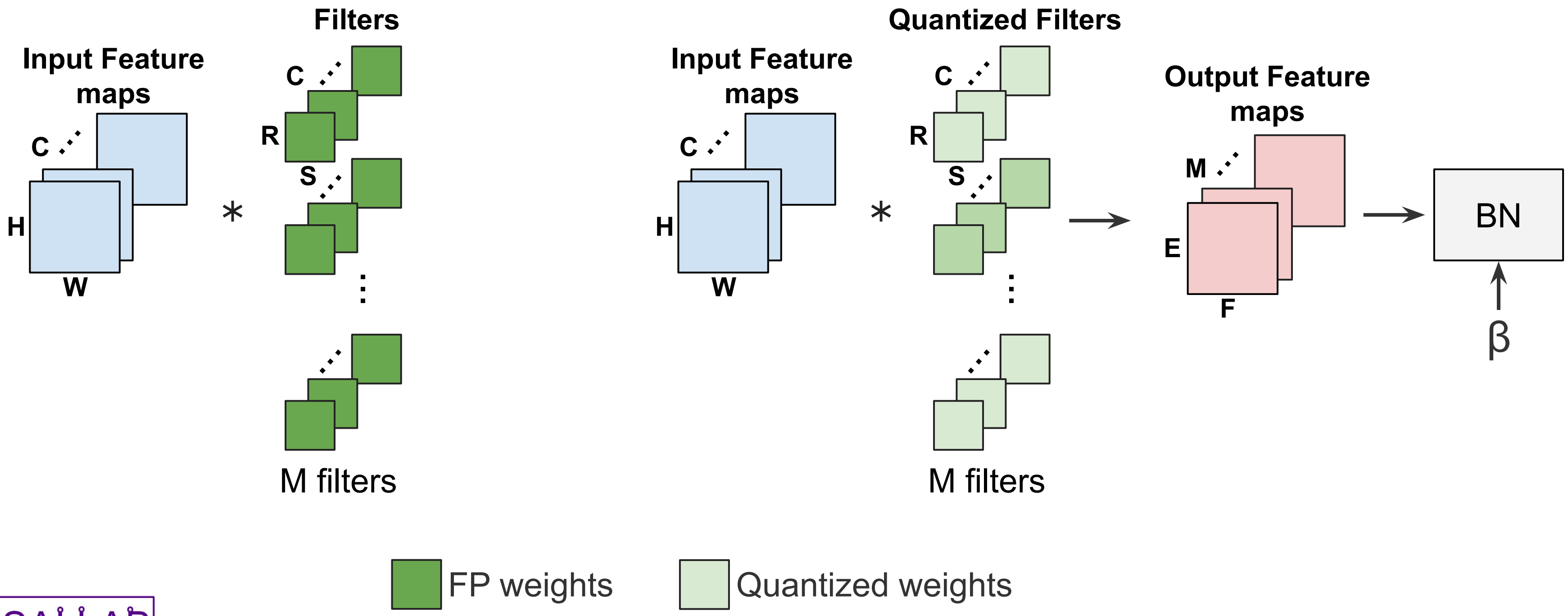
- To quantize a, conventional linear quantization will make q(a) = 0. However, this will cause a bias.
- With stochastic quantization:

$$q(a) = \begin{cases} 1 & \text{for } p = 0.2 \\ 0 & \text{for } p = 0.8 \end{cases}$$

- For quantization during the forward pass of DNN training, the bias will not cause any problem, due to the existence of bias in BN.
- Stochastic quantization is extremely useful when applying quantization to accelerate DNN training.

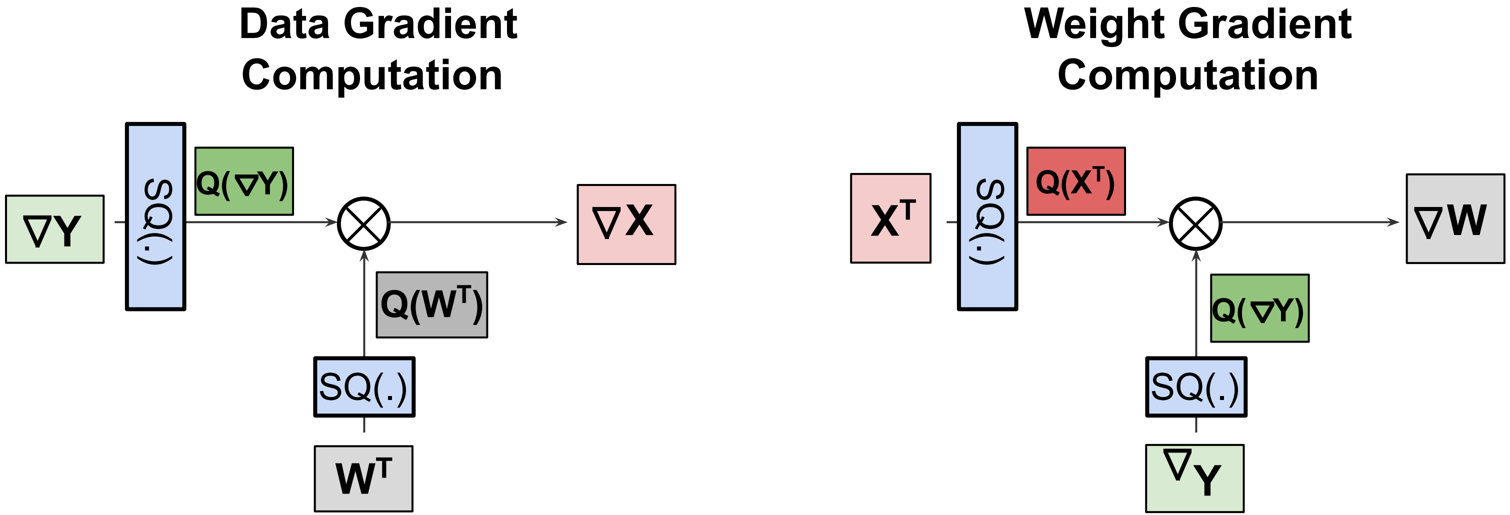
NYU SAI LAB

Deterministic and Stochastic Quantization



NYU SAI LAB

Quantization During Training



NYU SAI LAB

Topics

- Basic Data Formats
  - Fixed point (INT)
  - Floating point (FP)
  - Block floating point (BFP)
- Quantization methods
  - Taxonomy of Quantization
  - Learnable adaptive quantization scheme
  - Quantization for LLM

NYU SAI LAB

Learnable Quantization

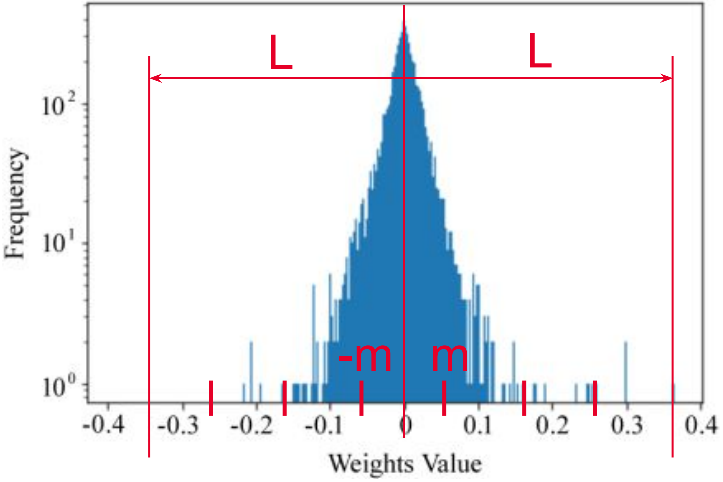
- Multiple methods have been proposed to learn the quantization hyperparameters:
  - PACT
  - QIL
  - Quantization network

Learnable Quantization

- How to convert a number to INT8 representation?
  - Set the clipping range: (-L_{min}, L_{max}), bitwidth: b
  - Compute the scale: $s = (L_{max} - L_{min})/(2^b - 1)$
  - Clip the input x: $x_c = Clip(x, L_{min}, L_{max})$
  - Calculate the fixed-point representation:
    $x_{int} = round((x_c - L_{min})/s)$
  - Rescale: $x_q = sx_{int} + L_{min}$

NYU SAI LAB

Learnable Quantization



Weight distribution in ResNet

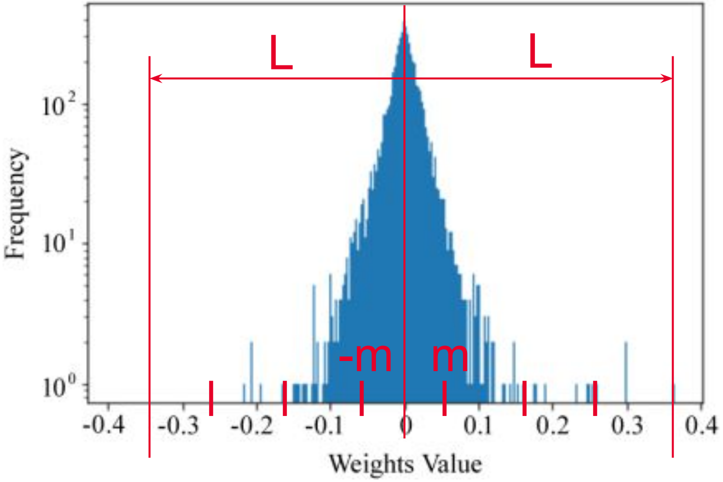
- How to convert a number to INT8 representation?
  - Set the clipping range: (-l, l), bitwidth: b
  - Compute the scale: $s = (2l)/(2^b-1)$
  - Clip the input x: $x_c = \text{Clip}(x, l, -l)$
  - Calculate the fixed-point representation:
    $x_{int} = \text{round}(x_c/s)$
  - Rescale: $x_q = s x_{int}$

l = 0.9×max(|W|), l = 0.95×max(|W|)

Can learn by learnt during training?

NYU SAI LAB

Learnable Quantization



- First we need to apply CLIP function to the input x, where the clip function has a range of (-l, l).
- $$x_c = Clip(x, l) = \begin{cases} l, & \text{if } x \geq l \\ x, & -l \leq x \leq l \\ -l, & x \leq l \end{cases}$$

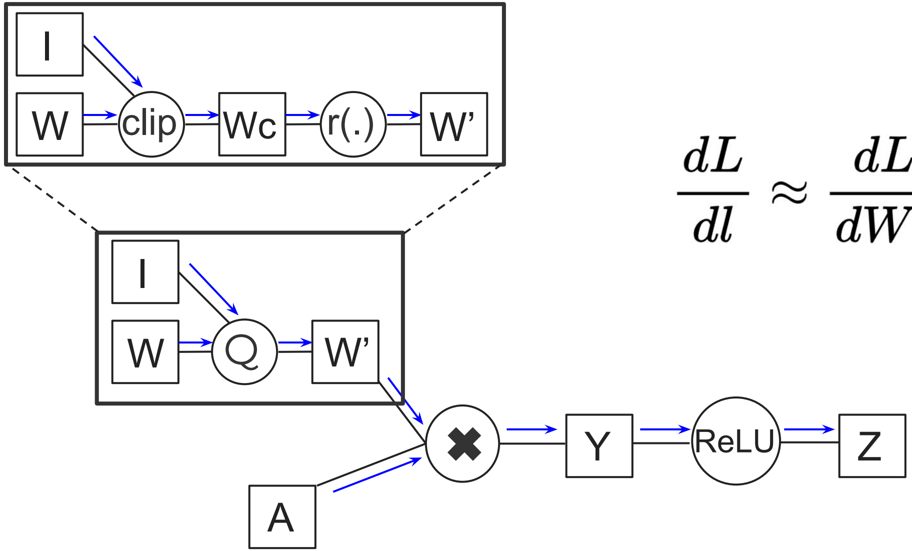
$$x_q = round(\frac{x_c}{s}) \times s$$

- Can we learn l? $\frac{dL}{dl} = \frac{dL}{dx_q}\frac{dx_q}{dx_c}\frac{dx_c}{dl} \approx \frac{dL}{dx_q}\frac{dx_c}{dl}$

NYU SAI LAB

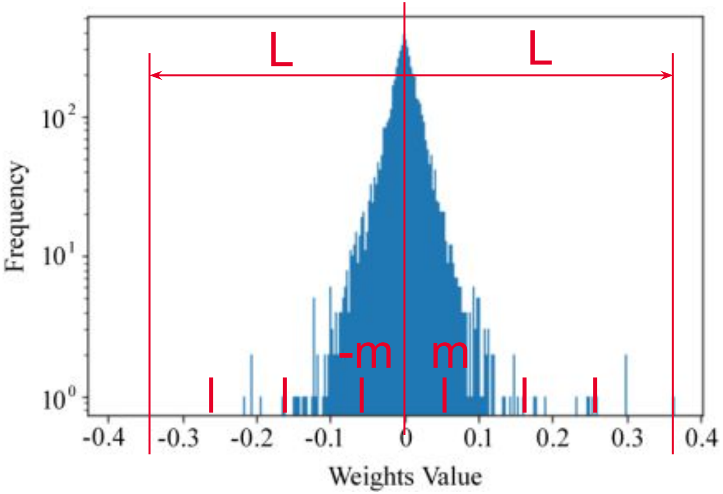
Choi, Jungwook, et al. "Pact: Parameterized clipping activation for quantized neural networks." *arXiv preprint arXiv:1805.06085* (2018).

Learnable Quantization



$$\frac{dL}{dl} \approx \frac{dL}{dW'} \underbrace{\cancel{\frac{dW'}{dW_c}}}_{1} \frac{dW_c}{dl}$$

Learnable Quantization



$$Clip(x,l) = \begin{cases} l, & \text{if } x \geq l \\ x, & -l \leq x \leq l \\ -l, & x \leq l \end{cases}$$

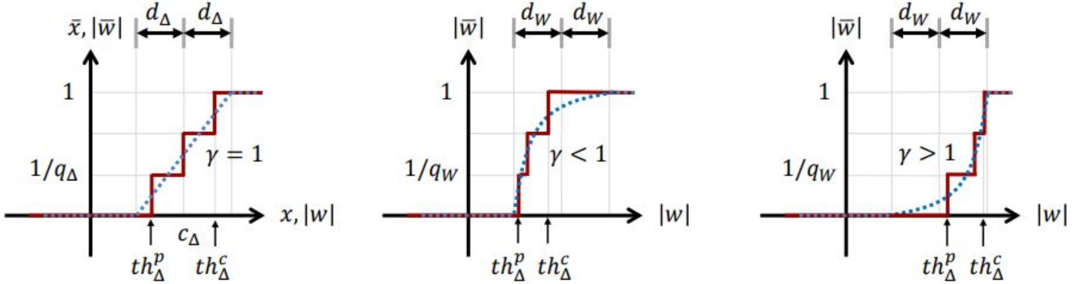
$$\frac{dClip(x,l)}{dx} = \begin{cases} 0, & \text{if } x \geq l \\ 1, & -l \leq x \leq l \\ 0, & x \leq l \end{cases}$$

$$\frac{dClip(x,l)}{dl} = \begin{cases} 1, & \text{if } x \geq l \\ 0, & -l \leq x \leq l \\ -1, & x \leq l \end{cases}$$

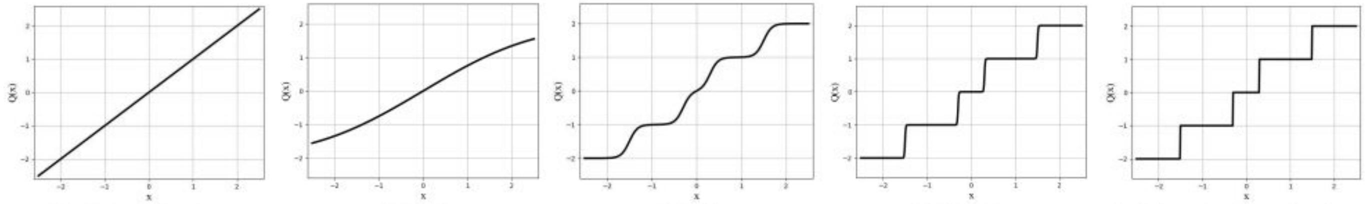
L can be learnable

Choi, Jungwook, et al. "Pact: Parameterized clipping activation for quantized neural networks." *arXiv preprint arXiv:1805.06085* (2018).

Learnable Quantization



QIL



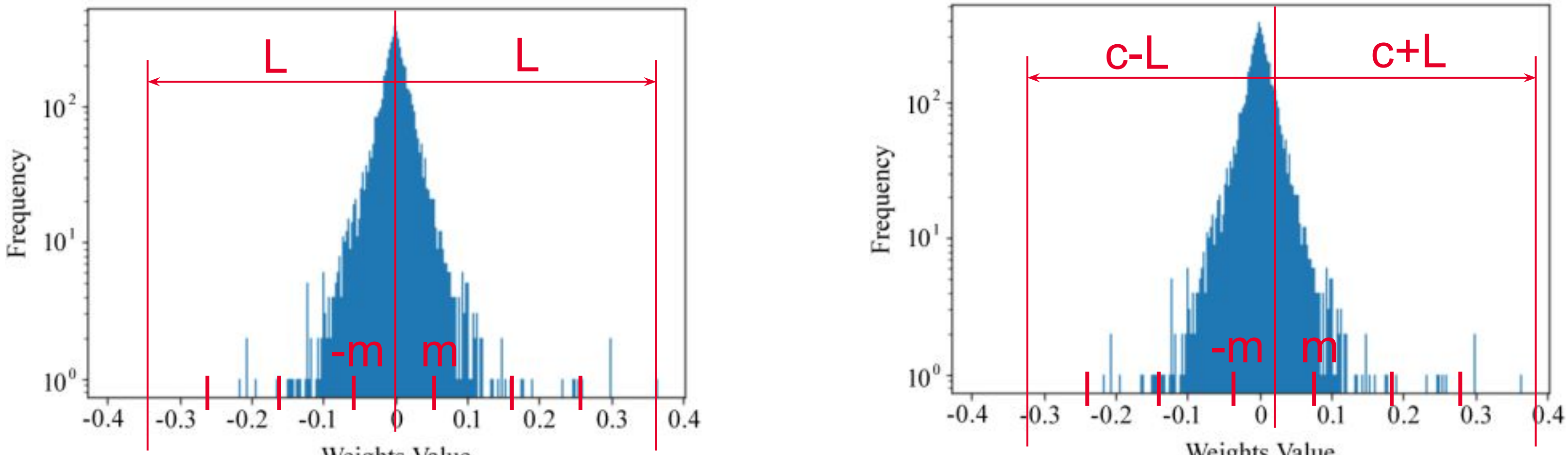
(a) No Quantization (b) T=1 (c) T=11 (d) T=121 (e) Complete Quantization

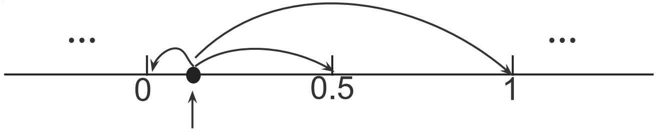
QN

Jung, Sangil, et al. "Learning to quantize deep networks by optimizing quantization intervals with task loss." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Yang, Jiwei, et al. "Quantization networks." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Quantization Interval Learning (QIL)

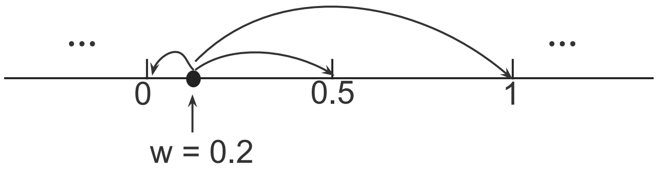




NYU SAI LAB

Jung, Sangil, et al. "Learning to quantize deep networks by optimizing quantization intervals with task loss." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Quantization Interval Learning (QIL)



- To achieve this rounding flexibility, we combine a learnable function with quantization.

$$w_q = Q(w) \longrightarrow w_q = Q(F(w))$$

- F(.) is a function which contains learnable hyperparameters.

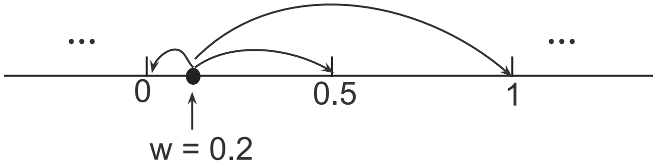
$$\hat{w} = \begin{cases} 0 & |w| < c_W - d_W \\ \text{sign}(w) & |w| > c_W + d_W \\ (\alpha_W |w| + \beta_W)^{\gamma} \cdot \text{sign}(w) & otherwise, \end{cases}$$

NYU SAI LAB

Jung, Sangil, et al. "Learning to quantize deep networks by optimizing quantization intervals with task loss." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

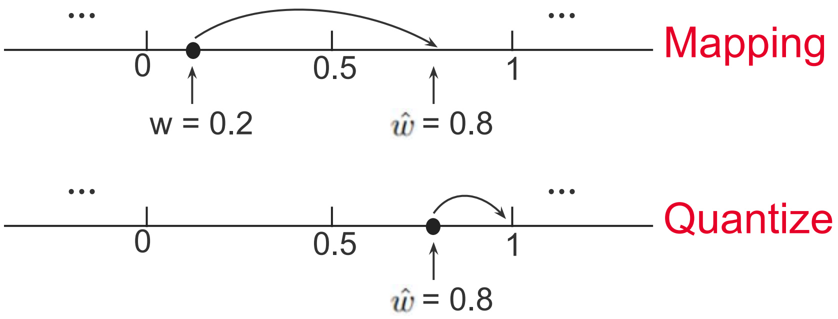
Quantization Interval Learning (QIL)

- QIL offers flexibility to round the FP weights.



$$\hat{w} = \begin{cases} 0 & |w| < c_W - d_W \\ \text{sign}(w) & |w| > c_W + d_W \\ (\alpha_W |w| + \beta_W)^{\gamma} \cdot \text{sign}(w) & \textit{otherwise}, \end{cases}$$

Mapping function contains some learnable parameters



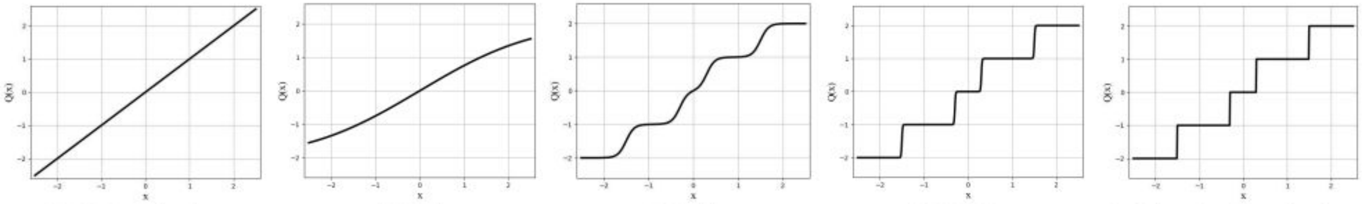
- $w_q = Q(F(w))$ are stored for inference after the training process finished.
- We can not apply this techniques over the activation, due to its large computational overhead.

NYU SAI LAB

Jung, Sangil, et al. "Learning to quantize deep networks by optimizing quantization intervals with task loss." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Quantization Networks

- We propose a novel perspective of interpreting and implementing neural network quantization by formulating low-bit quantization as a differentiable non-linear function.



(a) No Quantization (b) T=1 (c) T=11 (d) T=121 (e) Complete Quantization

$$y = \alpha\left(\sum_{i=1}^{n} s_i \mathcal{A}(\beta x - b_i) - o\right)$$

$$\mathcal{A}(x) = \begin{cases} 1 & x \geq 0, \\ 0 & x < 0. \end{cases}$$

- n + 1 is the number of quantization intervals
- β is the scale factor of inputs
- s_i and b_i are the scales and biases for the unit step functions

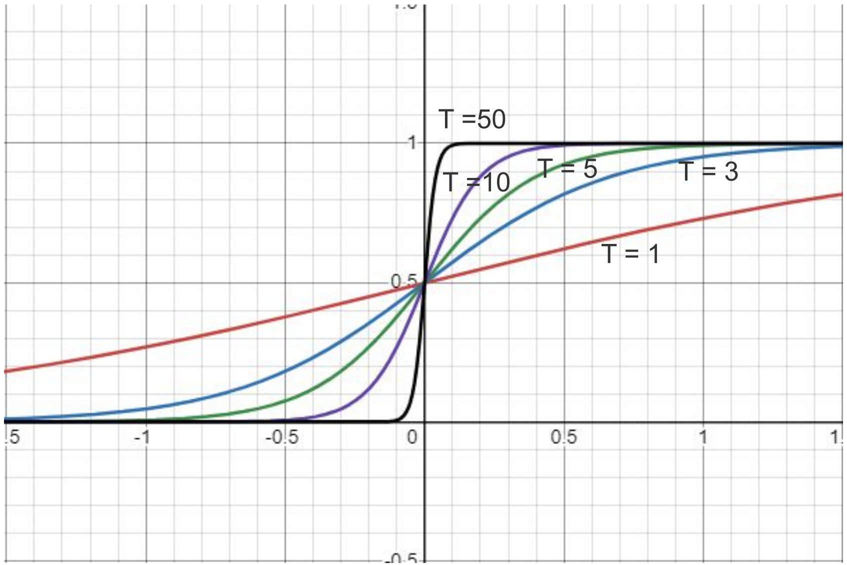
NYU SAI LAB

Yang, Jiwei, et al. "Quantization networks." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Gong, Ruihao, et al. "Differentiable soft quantization: Bridging full-precision and low-bit neural networks." *Proceedings of the IEEE/CVF international conference on computer vision*. 2019.

Quantization Networks

$$\mathcal{A}(x) = \begin{cases} 1 & x \geq 0, \\ 0 & x < 0. \end{cases} \qquad \sigma(Tx) = \frac{1}{1 + exp(-Tx)}$$



- We can replace the staircase function with a sigmoid function.
- We can progressively increases T during the training process.

Topics

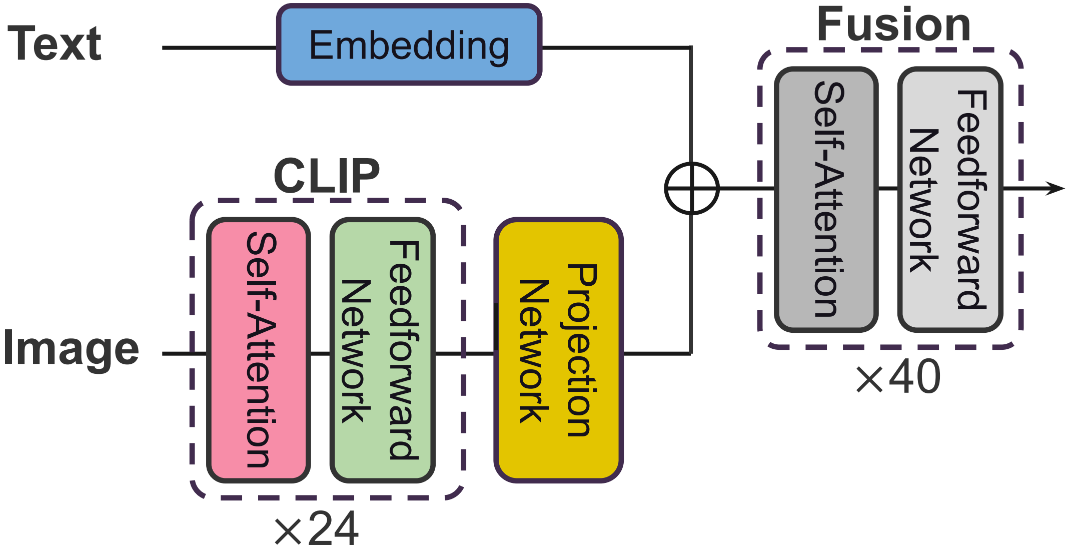
- Basic Data Formats
  - Fixed point (INT)
  - Floating point (FP)
  - Block floating point (BFP)
- Quantization methods
  - Taxonomy of Quantization
  - Learnable adaptive quantization scheme
  - Quantization for LLM
    - Smoothing
    - Quantization

NYU SAI LAB

Post Training Quantization

- Several Methods have been proposed to efficient post-training quantization.
- Given the large size of the modern LLM, it is beneficial to applied the quantization on the model directly without the need of finetuning.

Case Study: CLIP in Llava



⊕ Concat in token dim

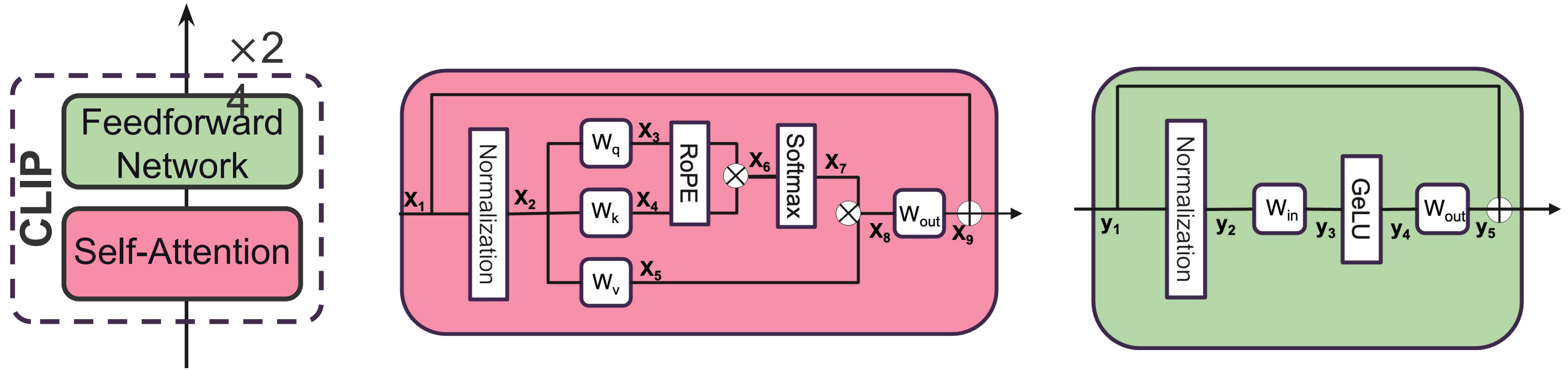
Visual input example, Tricky Question:

User	If there are factual errors in the questions, point it out; if not, proceed answering the question. What's happening in the desert?
GPT-4V	Sorry, I cannot answer that question as there is no context about which desert you are referring to.
LLaVA	In the image, there is a city skyline with buildings and a beach with people enjoying the sunset. The scene is set in a desert, which is unusual for a beach setting.
LLaVA-1.5	There are no deserts in the image. The image features a beach with palm trees, a city skyline, and a large body of water.

NYU SAI LAB

Liu, Haotian, et al. "Visual instruction tuning." *Advances in neural information processing systems* 36 (2024).

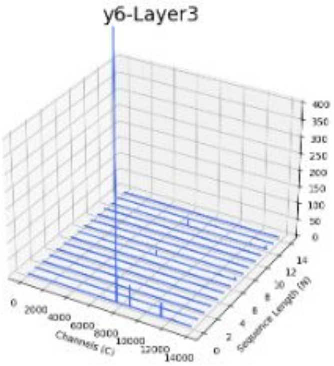
CLIP Architecture

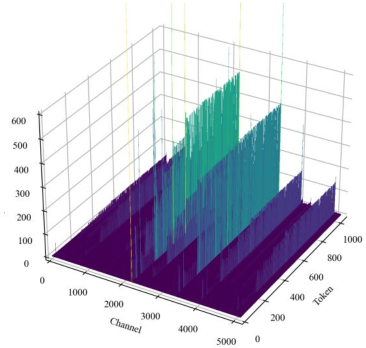


Radford, Alec, et al. "Learning transferable visual models from natural language supervision." *International conference on machine learning*. PMLR, 2021.

Types of Outlier

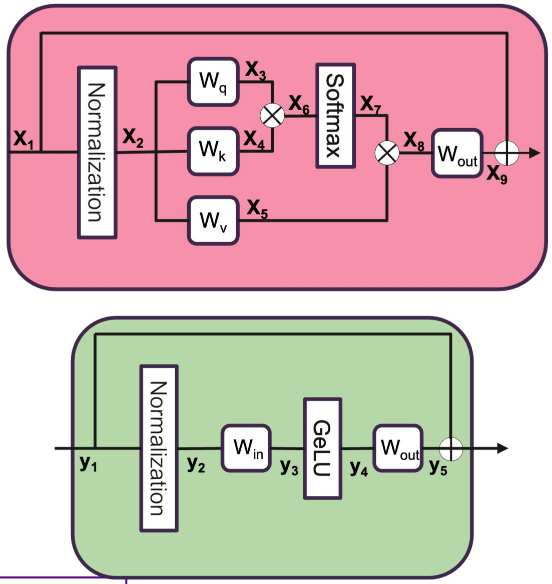
- Massive Activation:
  - For an activation matrix A, an massive activation is an element Aij within it that satisfies:
  - Aij > η✕mean(|A|)
  - Aij > γ
  - η=300, γ=50
- Channelwise Outlier:
  - mean(Ai) > η✕std(A) +mean(|A|)
  - std(Ai) < β
  - η=3, β=0.6

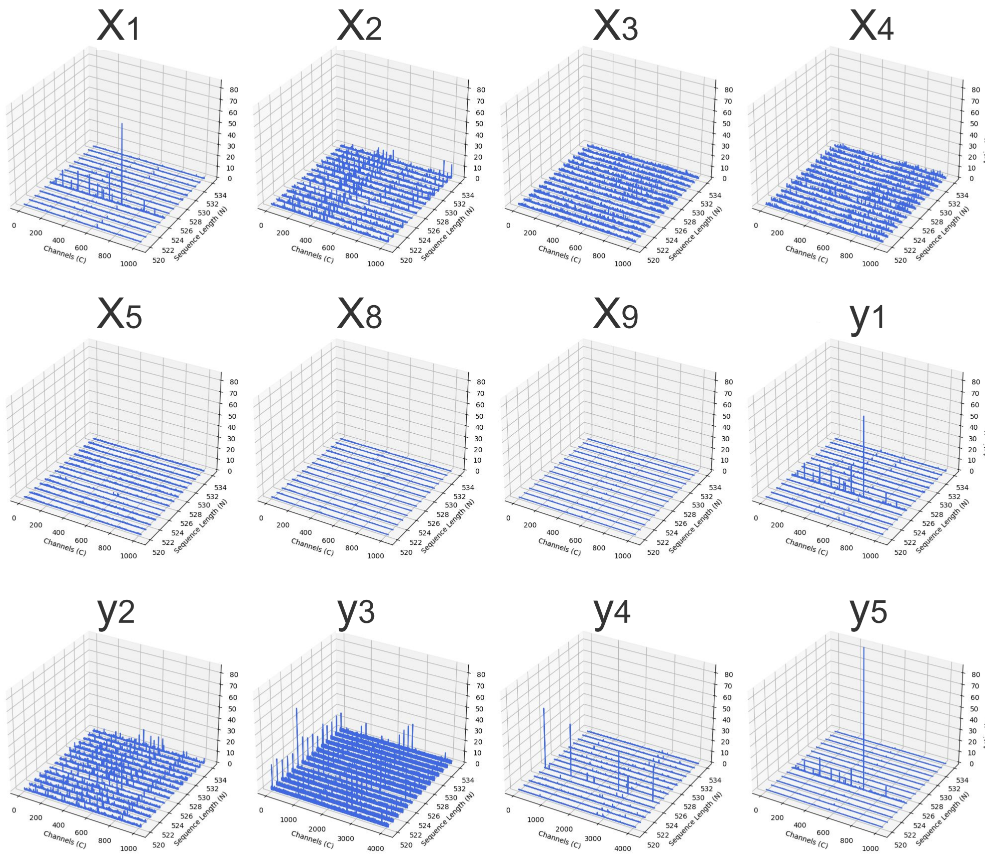




Outlier Study: CLIP Activations

- 3D activation within **layer 12**

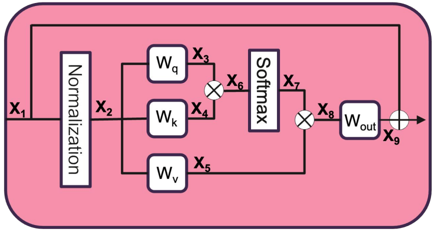




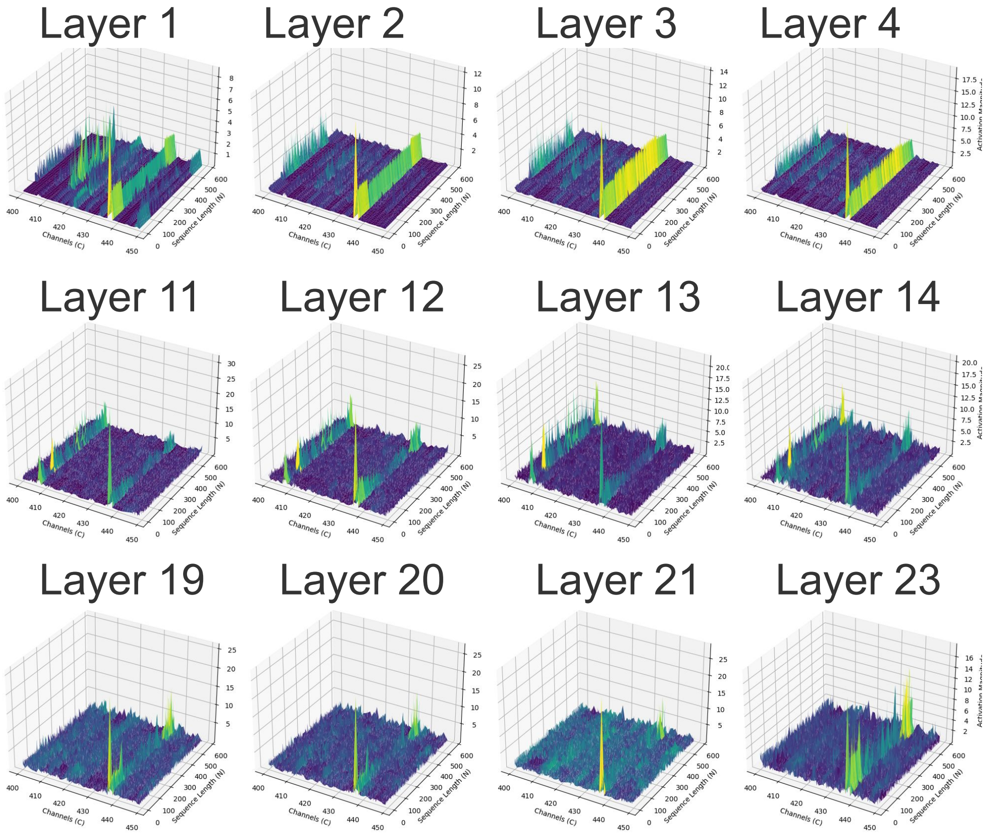
NYU SAI LAB

Outlier Study: CLIP Activations

- 3D plots of X2 across layers.



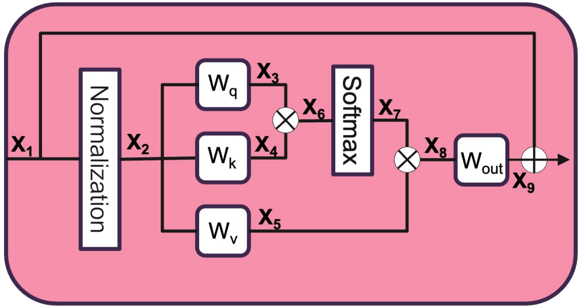
- x2 exhibits channel wise outlier



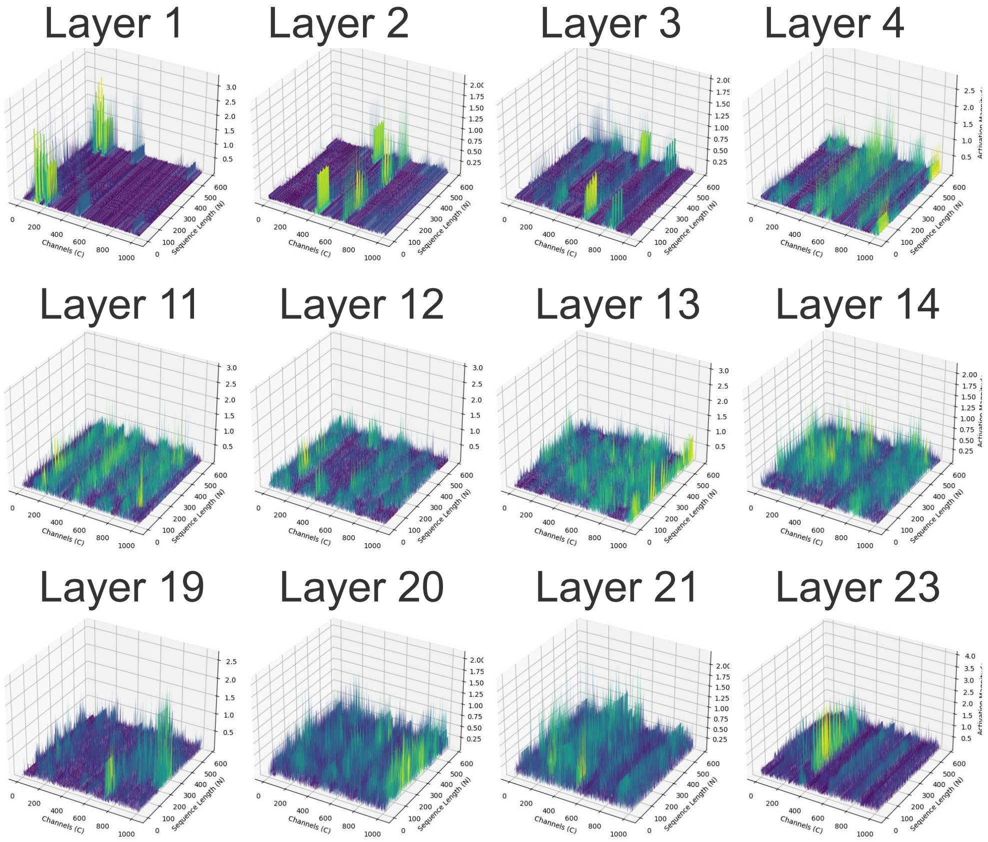
NYU SAI LAB

Outlier Study: CLIP Activations

- 3D plots of x8 across layers.



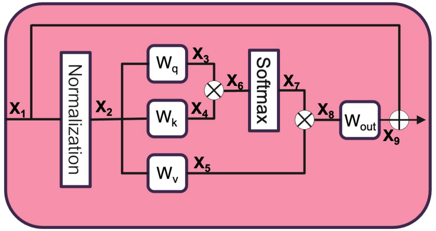
- x8 exhibits channel wise outlier

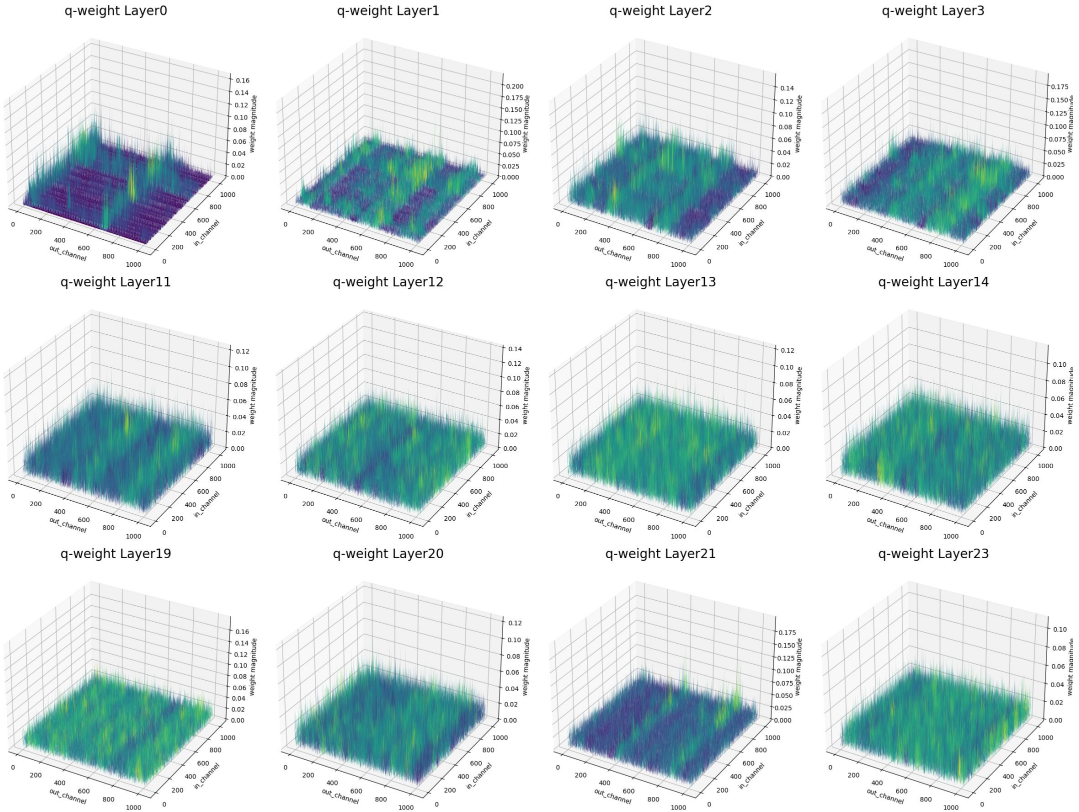


NYU SAI LAB

Outlier Study: CLIP Weights

- W_q across CLIP layers.

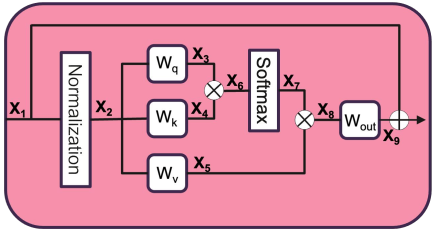


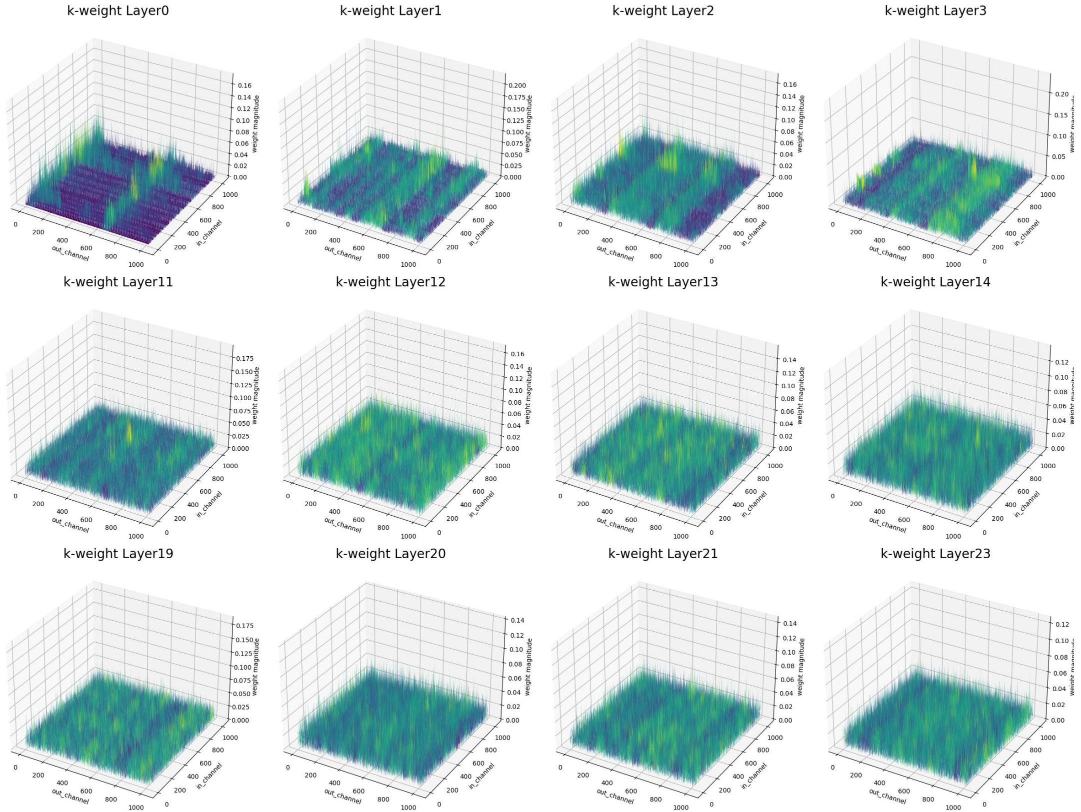


NYU SAI LAB

Outlier Study: CLIP Weights

- W_K across CLIP layers.

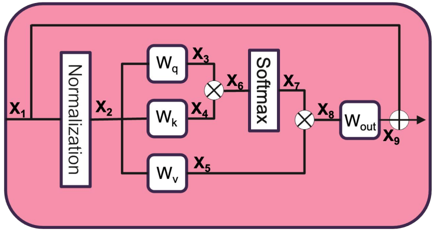


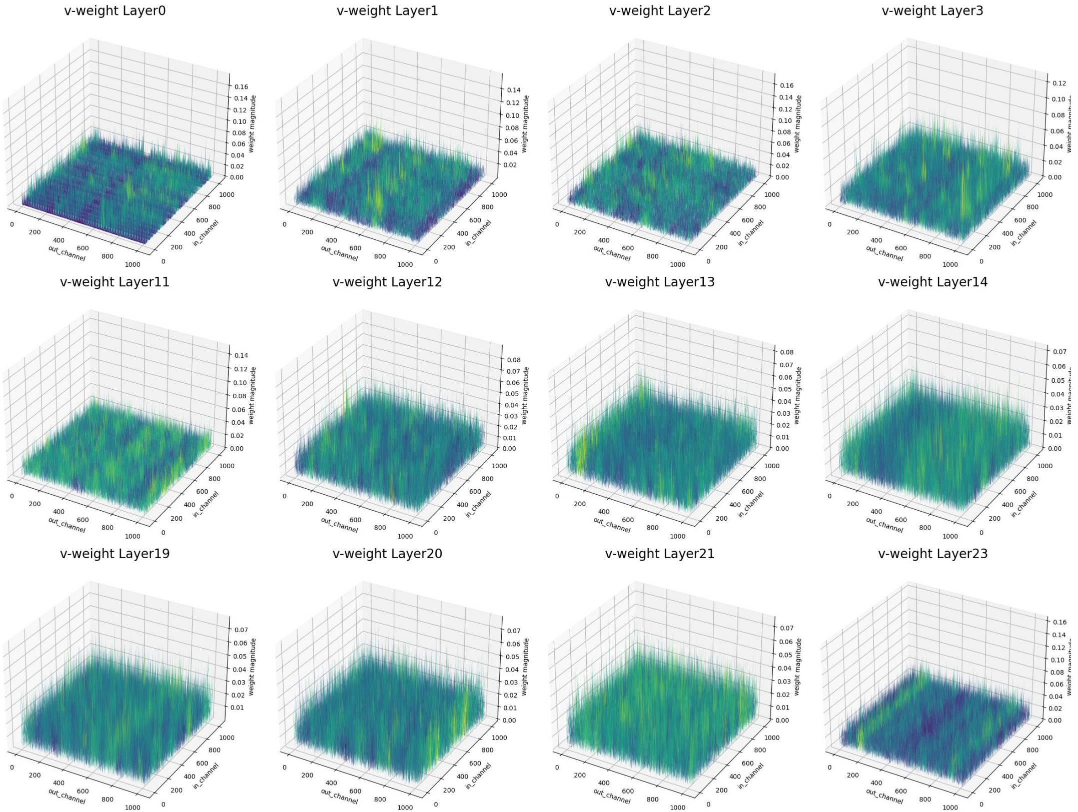


NYU SAI LAB

Outlier Study: CLIP Weights

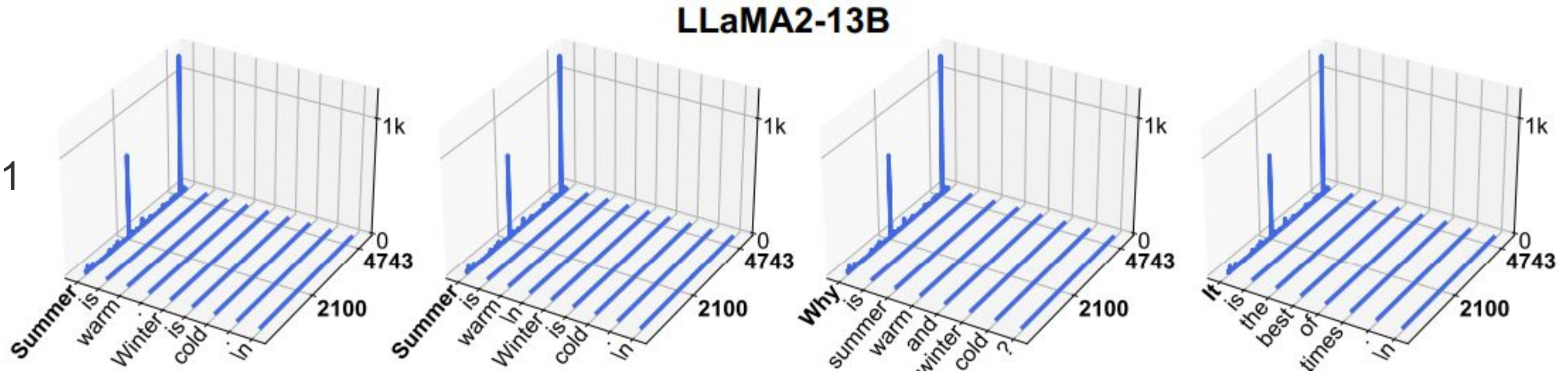
- W_v across CLIP layers.





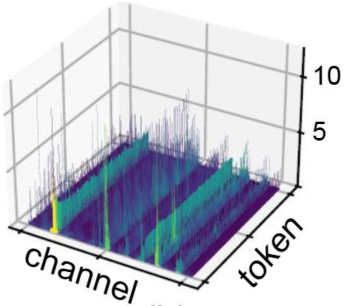
NYU SAI LAB

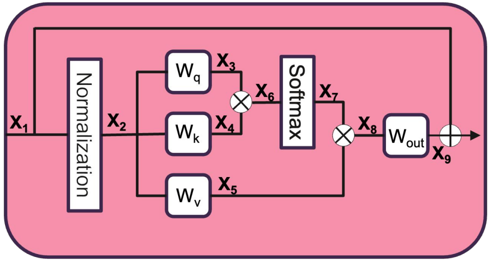
Outlier Study: LLaMA Activations



X1

X2

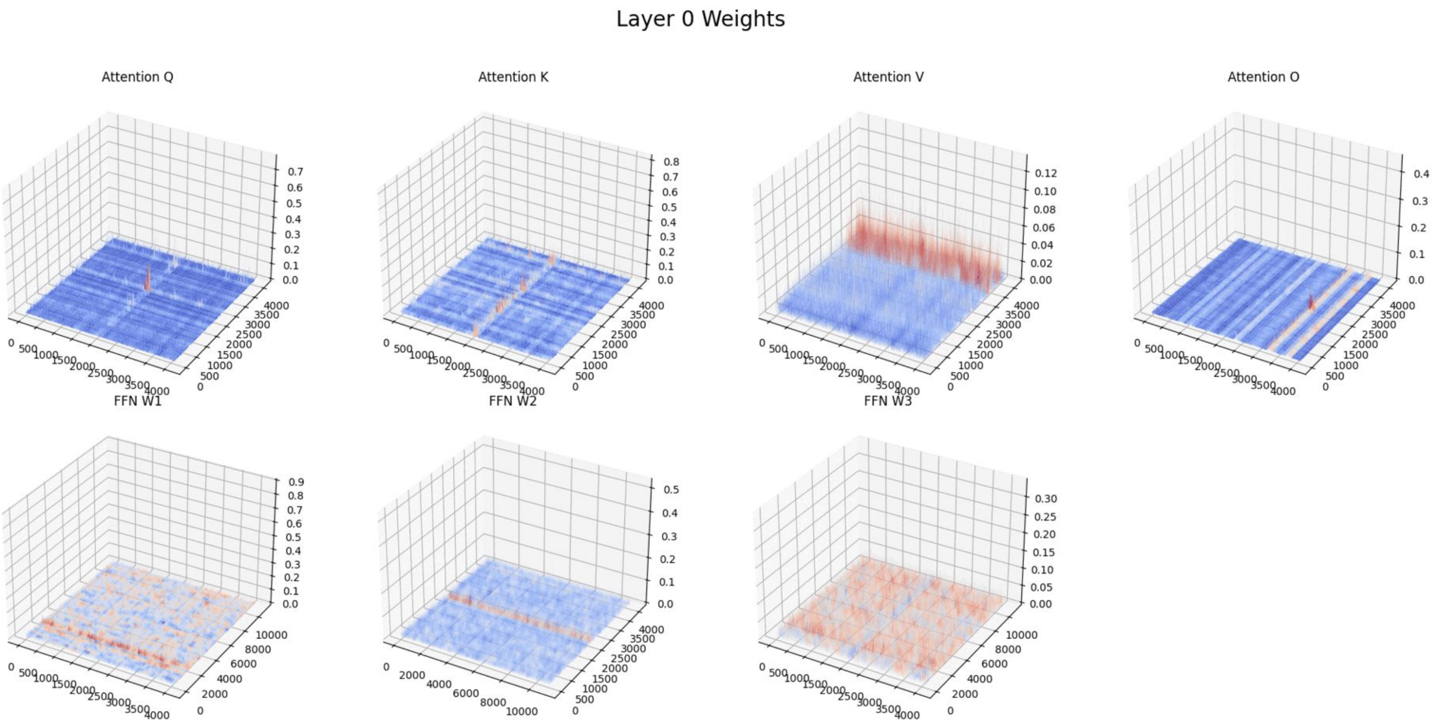




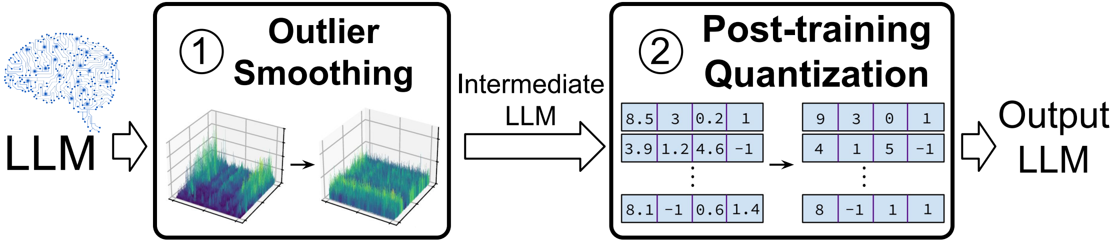
NYU SAI LAB

Sun, Mingjie, et al. "Massive activations in large language models." *arXiv preprint arXiv:2402.17762* (2024).

Study the Reason of LLM Outliers



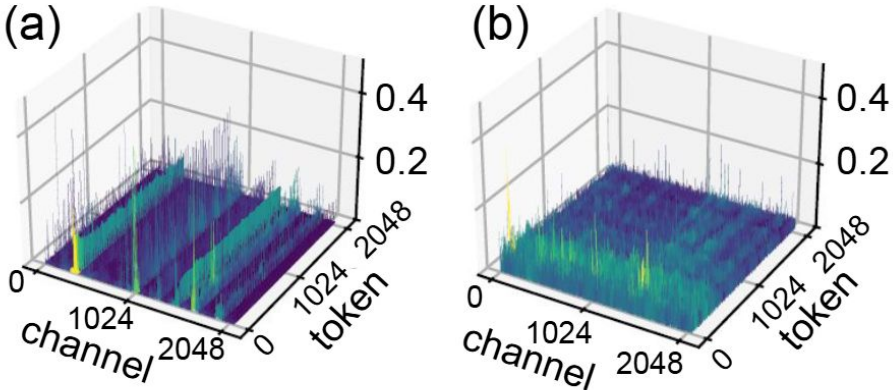
Outlier Smoothing



- When performing post-training quantization on a LLM, it's common to include a step of outlier smoothing prior to the quantization process.

NYU SAI LAB

QuaRot

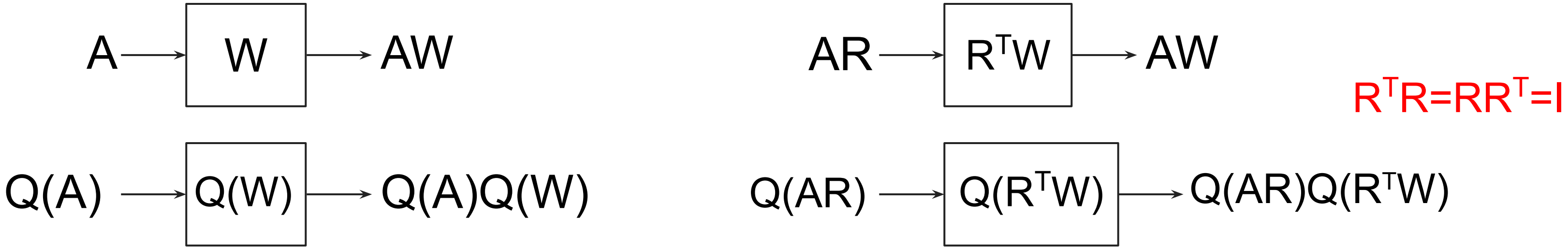


- QuaRot introduces a novel methods to convert the weights and activation of LLM.
- After conversion, most of the outliers within the activation and weights are removed.
- This conversion introduces almost no additional cost during the inference.

Ashkboos, Saleh, et al. "Quarot: Outlier-free 4-bit inference in rotated llms." *arXiv preprint arXiv:2404.00456* (2024).

QuaRot

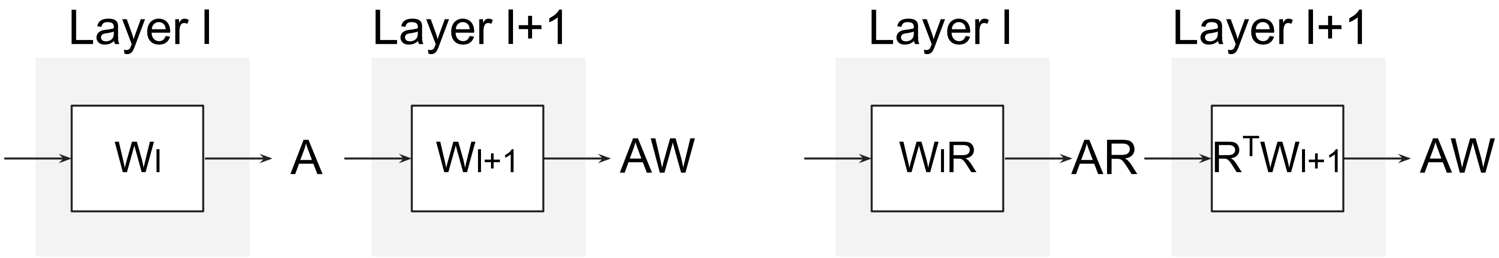
- Assume Y = AW, where A may have outliers, quantizing A and W as Q(A) and Q(W) could result in increased quantization error. Consequently, Q(A)Q(W) may differ significantly from AW.
- With QuaRot, a orthogonal matrix is applied to eliminate the outliers within A.



- R^TW can be computed offline, AR can be generated by modifying the weight matrices of the last layer.

Ashkboos, Saleh, et al. "Quarot: Outlier-free 4-bit inference in rotated llms." *arXiv preprint arXiv:2404.00456* (2024).

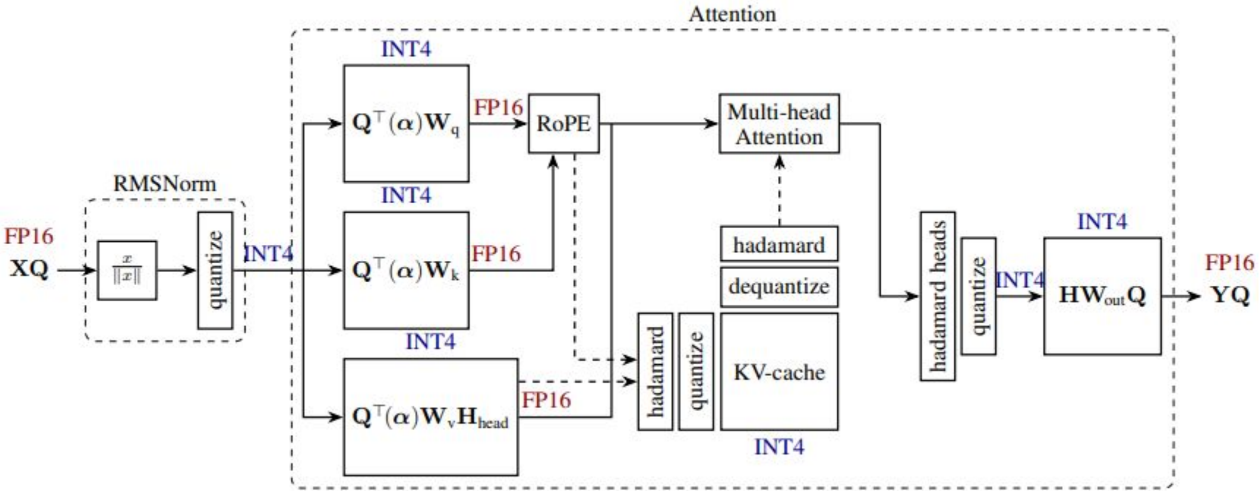
QuaRot



- R^TW can be computed offline, AR can be generated by modifying the weight matrices of the last layer.

 Ashkboos, Saleh, et al. "Quarot: Outlier-free 4-bit inference in rotated llms." *arXiv preprint arXiv:2404.00456* (2024).

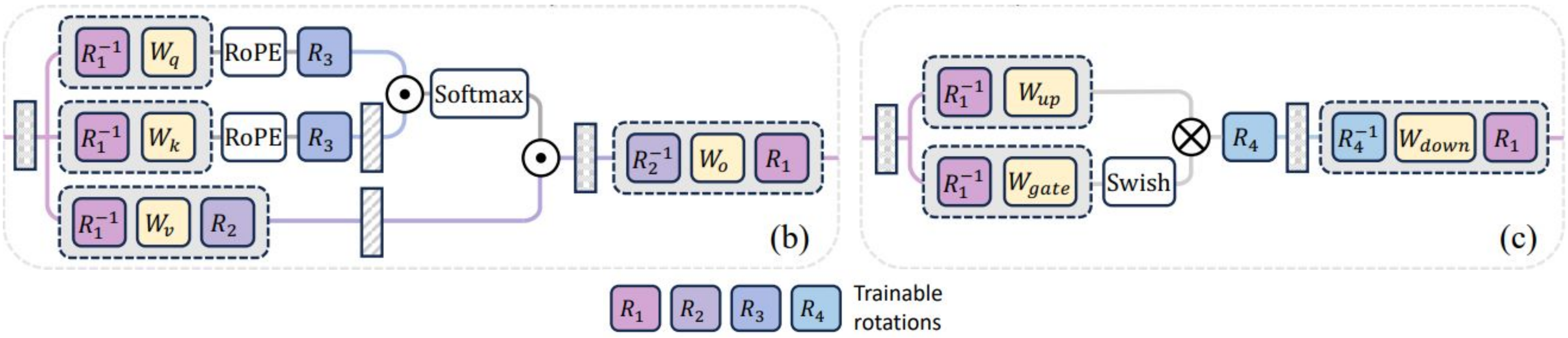
QuaRot



- For some of the layers, the conversion needs to be performed online
- We can use Hadamard matrix, which consists of only 1 and -1 to facilitate the matrix multiplications.

Ashkboos, Saleh, et al. "Quarot: Outlier-free 4-bit inference in rotated llms." *arXiv preprint arXiv:2404.00456* (2024).

SpinQuant

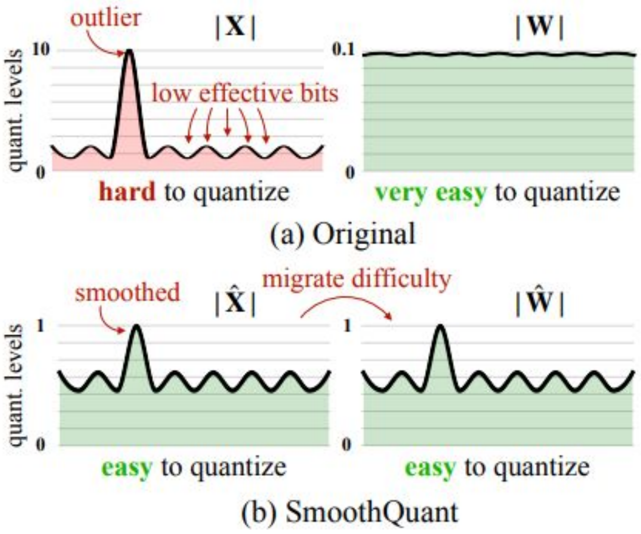


$$\arg\min_{R \in \mathcal{M}} \mathcal{L}_Q(R_1, R_2 \mid W, X)$$

- SpinQuant optimizes (or learns) the rotation matrices to obtain the minimal changes on the training loss.
- We have to ensure the rotational matrix still satisfies the orthogonal property → Cayley Optimization.

Liu, Zechun, et al. "SpinQuant--LLM quantization with learned rotations." *arXiv preprint arXiv:2405.16406* (2024).

SmoothQuant



- The intermediate results within LLM usually have a lot of outliers.
- SmoothQuant smooths the activation outliers by offline migrating the quantization difficulty from activations to weights with a mathematically equivalent transformation.

$$\mathbf{Y} = (\mathbf{X}\text{diag}(\mathbf{s})^{-1}) \cdot (\text{diag}(\mathbf{s})\mathbf{W}) = \hat{\mathbf{X}}\hat{\mathbf{W}}$$

- s depends on the square root of the magnitude of the largest channel

NYU SAI LAB

Xiao, Guangxuan, et al. "Smoothquant: Accurate and efficient post-training quantization for large language models." *International Conference on Machine Learning*. PMLR, 2023.

Activation-Aware Weight Quantization (AWQ)

$$\mathbf{s}^* = \arg\min_{\mathbf{s}} \mathcal{L}(\mathbf{s})$$

$$\mathcal{L}(\mathbf{s}) = \|Q(\mathbf{W} \cdot \text{diag}(\mathbf{s}))(\text{diag}(\mathbf{s})^{-1} \cdot \mathbf{X}) - \mathbf{W}\mathbf{X}\|$$

- AWQ improves the performance of smoothquant by making "s" learnable.

PPL↓		**Llama-2**			**LLaMA**			
		7B	13B	70B	7B	13B	30B	65B
FP16	-	5.47	4.88	3.32	5.68	5.09	4.10	3.53
INT3 g128	RTN	6.66	5.52	3.98	7.01	5.88	4.88	4.24
	GPTQ	6.43	5.48	3.88	8.81	5.66	4.88	4.17
	GPTQ-R	6.42	5.41	3.86	6.53	5.64	4.74	4.21
	AWQ	**6.24**	**5.32**	**3.74**	**6.35**	**5.52**	**4.61**	**3.95**
INT4 g128	RTN	5.73	4.98	3.46	5.96	5.25	4.23	3.67
	GPTQ	5.69	4.98	3.42	6.22	5.23	4.24	3.66
	GPTQ-R	5.63	4.99	3.43	5.83	5.20	4.22	3.66
	AWQ	**5.60**	**4.97**	**3.41**	**5.78**	**5.19**	**4.21**	**3.62**

Lin, Ji, et al. "AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration." *Proceedings of Machine Learning and Systems* 6 (2024): 87-100.

Presentation

- Trained ternary quantization (Athul)
- Incremental network quantization: Towards lossless cnns with low-precision weights (Jay)
- Quantization and training of neural networks for efficient integer-arithmetic-only inference (Chahat)
- Smoothquant: Accurate and efficient post-training quantization for large language models (Naveenraj)