CVPR 2025

Qualcomm AI research

Power-Efficient Neural Networks Using Low-Precision Data Types and Quantization

Part 2:
Quantization Algorithms Fundamentals

Markus Nagel

Sr. Staff engineer/manager
Qualcomm AI Research
Amsterdam, The Netherlands

markusn@qti.qualcomm.com
https://www.linkedin.com/in/markusnagel/
Google Scholar

Quantization algorithms overview

PTQ

- Only small amount of calibration data needed
- Requires little compute and scales well
- Optimization is often on local loss
- Less user effort and little hyper-parameter tuning

Spectrum

QAT

- Training on end-to-end task loss
- Requires significant amount of training data and compute
- Higher user effort and can require domain knowledge
- Achieves best accuracy-efficiency trade-off

Efficient training

Quantization algorithms overview

Mixed precision

LLM/LVM specific

PTQ

- Quantization range setting & optimization
- Weight assignment optimization using local loss
- Function preserving transformations

QAT

- Gradient estimators for discretization
- Learning the quantization representation

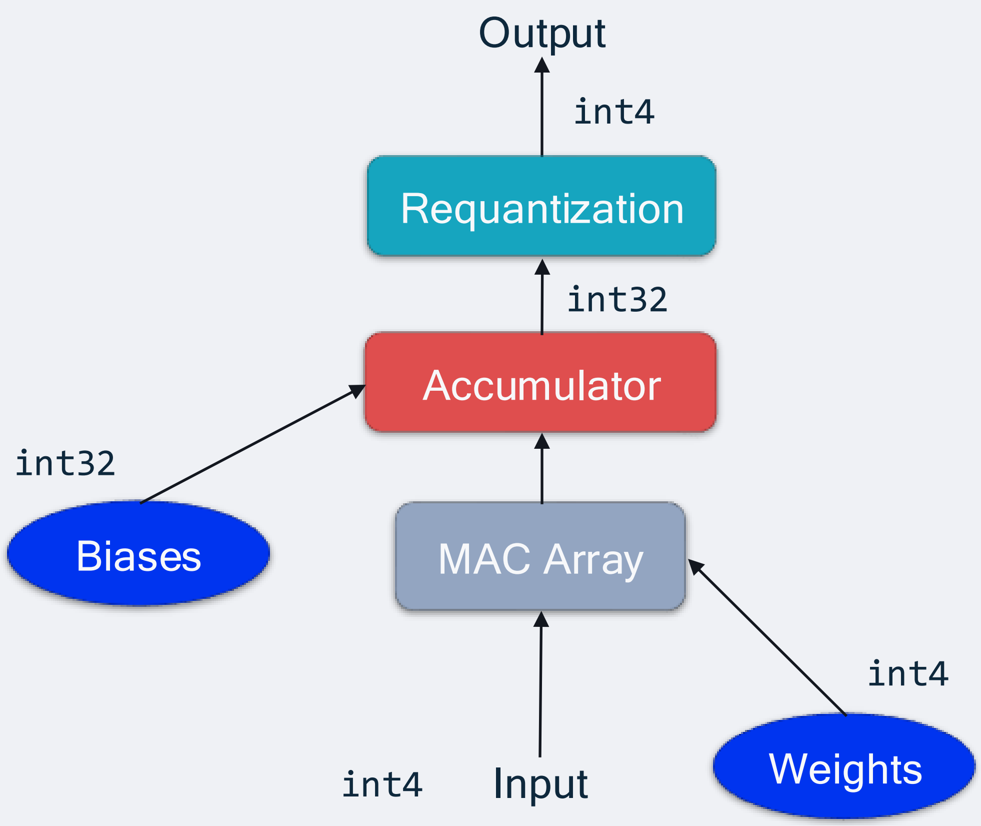
Efficient training

LLM/LVM specific:

- LLM weight compression
- KV compression
- Outliers
- Memory efficient QAT
- QAT scaling laws
- Efficient LLM training

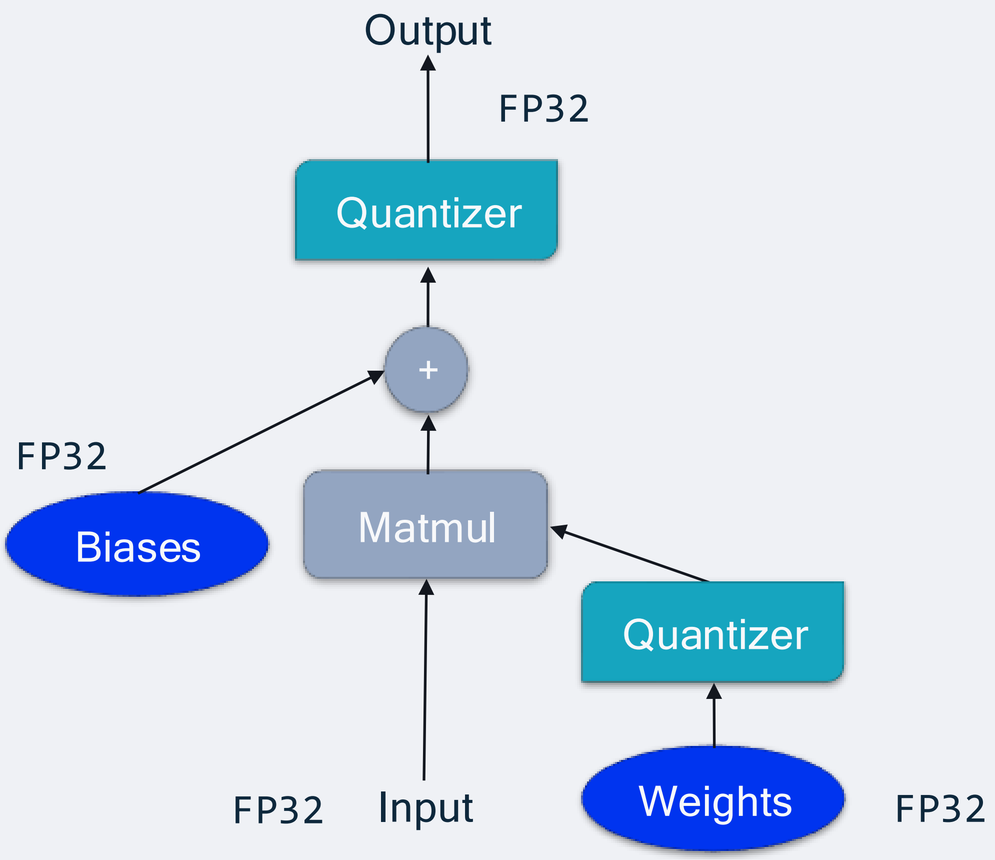
Recap: fixed point quantization

Fixed-point inference



$$Y = WX \approx \frac{s_w s_x}{s_y}(W_{int}X_{int})$$

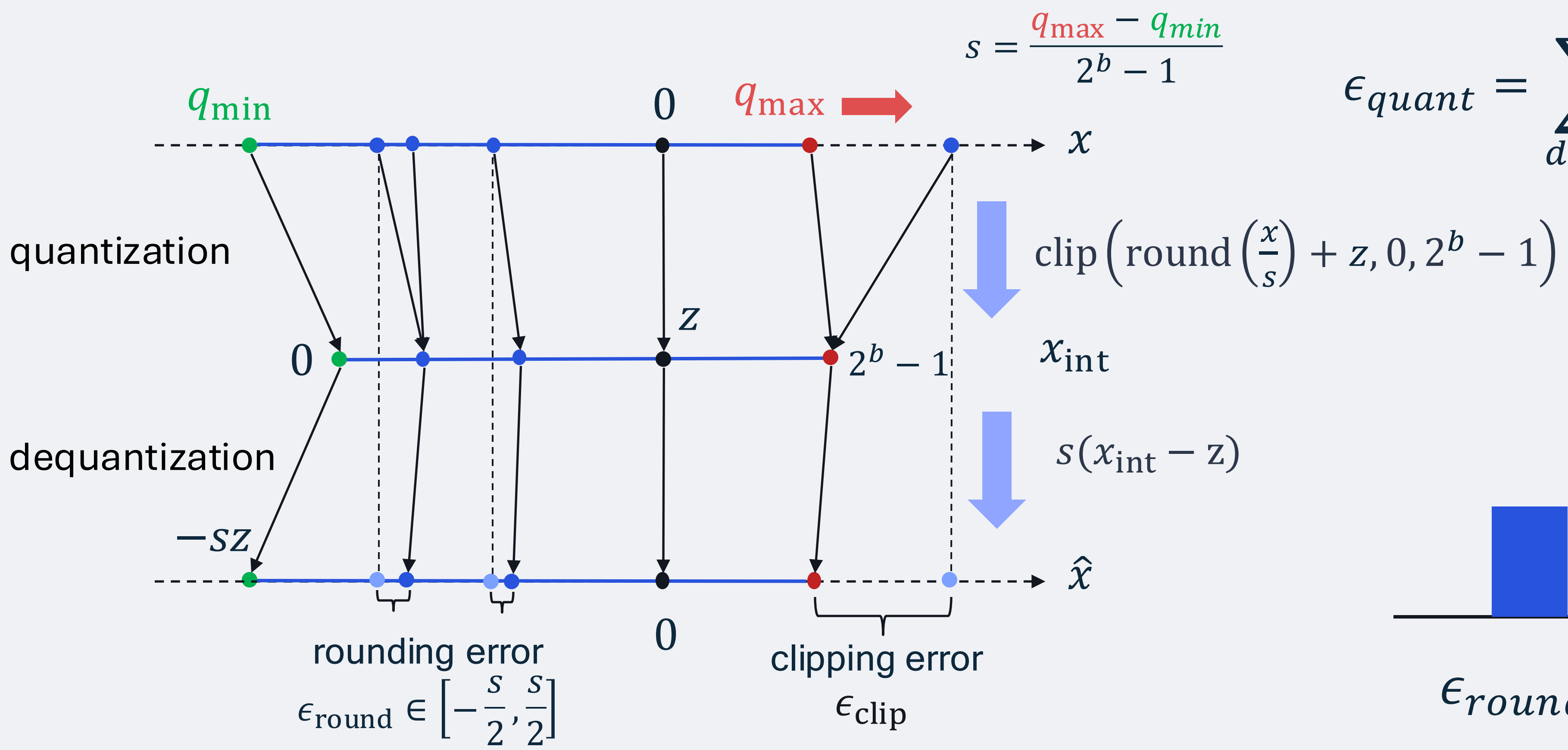
Simulated quantization



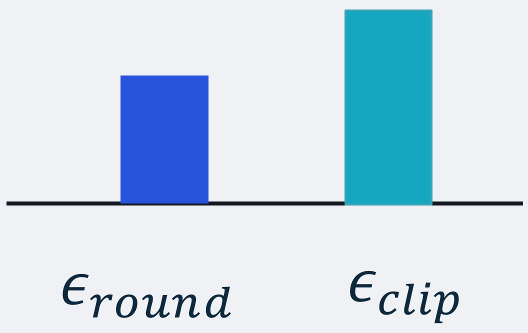
$$Y = WX \approx q(q(W, s_w)q(X, s_x), s_y)$$

Part 1:
Post training quantization

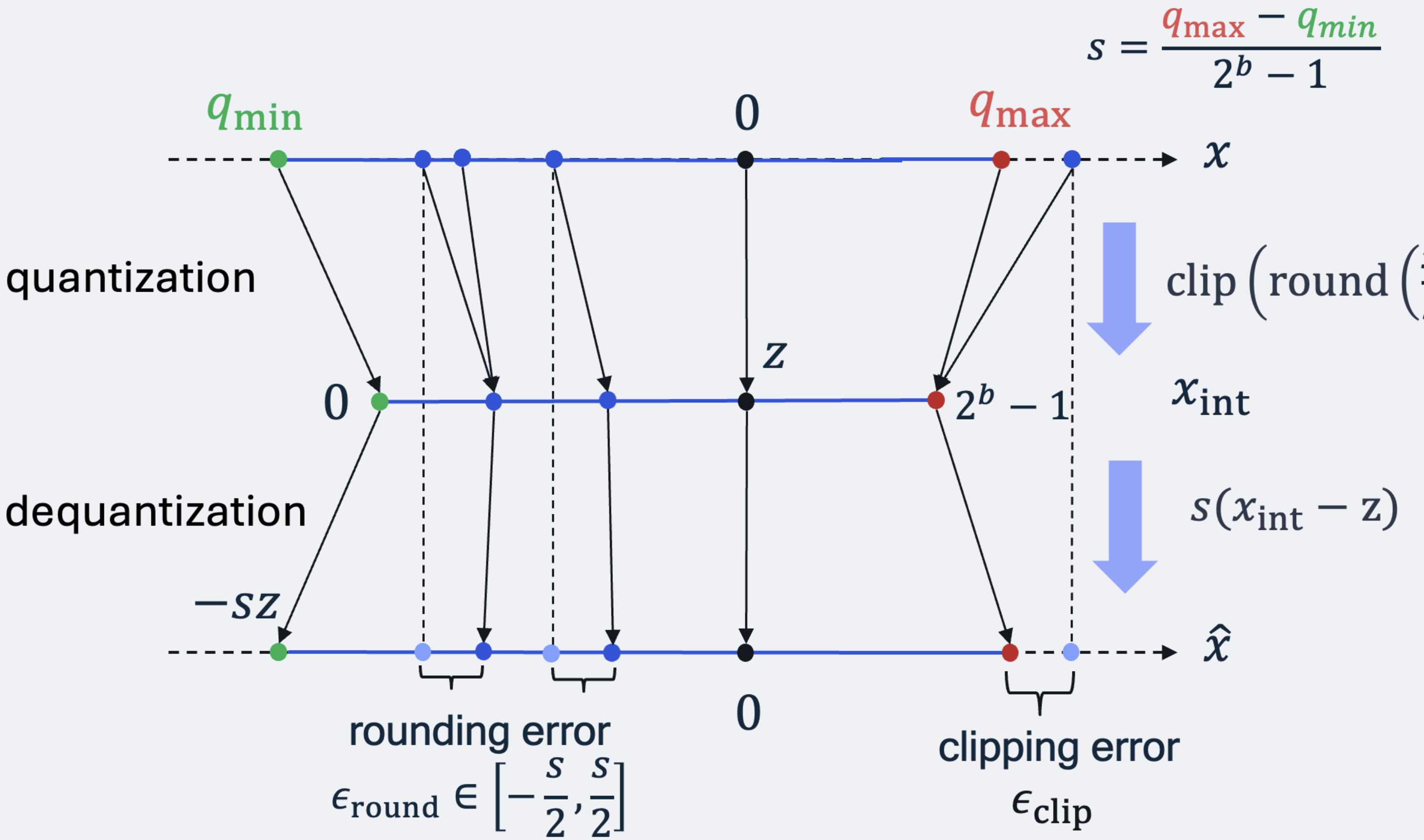
Sources of quantization error



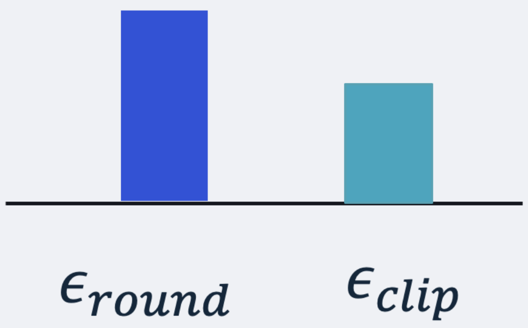
$$\epsilon_{quant} = \sum_{data} \epsilon_{round} + \epsilon_{clip}$$



Sources of quantization error



$$\epsilon_{quant} = \sum_{data} \epsilon_{round} + \epsilon_{clip}$$



Quantization range setting

- Min-max range:

$$q_{\min} = \min \boldsymbol{X}$$
$$q_{\max} = \max \boldsymbol{X}$$

- Optimization-based methods:

$$\operatorname{argmin}_{q_{\min}, q_{\max}} \ell\left(\boldsymbol{X}, \widehat{\boldsymbol{X}}(q_{\min}, q_{\max})\right)$$

Common optimization techniques

- Line/grid search
- Golden section search
- OMSE

Common loss functions

- MSE
- KL divergence
- Cross-entropy
- LP-norm (e.g. p={2.4, 3.0, 3.5})

A White Paper on Neural Network Quantization (Nagel et al., 2021)
Quantizing deep convolutional networks for efficient inference: A whitepaper (Krishnamoorthi, 2018)
Low-bit Quantization of Neural Networks for Efficient Inference (Choukroun et al., ICCVW 2019)

Optimizing weight assignment

Optimizing weight assignment using local loss

- Traditionally, in PTQ we use rounding-to-nearest (RTN)

$$\boldsymbol{W}_{\text{int}} = \text{clip}\left(\textbf{round}\left(\frac{W}{s}\right), n, p\right)$$

- RTN is a natural choice as it is optimal for the local MSE

$$argmin_{W_{int}} ||W - W_{int}s||_F^2$$

- However, is RTN optimal for the task loss?

Rounding-to-nearest is not optimal for task loss

Toy example

- Taylor series expansion of task loss $\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{w})$

$$\mathcal{L}(\mathbf{w} + \Delta\mathbf{w}) - \mathcal{L}(\mathbf{w}) \approx \frac{1}{2}\Delta\mathbf{w}^T \mathbf{H}^{(\mathbf{w})} \Delta\mathbf{w}$$

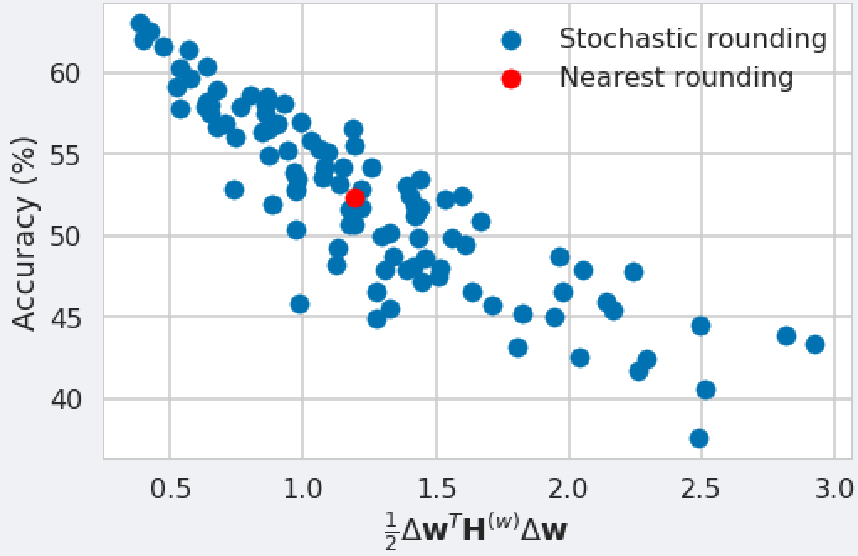
- Assume $\mathbf{H}^{(\mathbf{w})} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$

then $\Delta\mathbf{w}^T \cdot \mathbf{H}^{(\mathbf{w})} \cdot \Delta\mathbf{w} = \Delta\mathbf{w}_1^2 + \Delta\mathbf{w}_2^2 + \Delta\mathbf{w}_1 \Delta\mathbf{w}_2$

- Rounding-to-nearest:
  - Only considers the magnitude of ΔW
  - Ignores all off-diagonal terms of $\boldsymbol{H}^{(\boldsymbol{w})}$

Empirical example

- We draw 100 stochastic rounding samples
  - Out of space of 2^9408 rounding possibilities



- Rounding-to-nearest far from optimal!

Up or Down? Adaptive Rounding for Post-Training Quantization (Nagel, Amjad, et al., ICML 2020)

How can we find systematically the best weight assignment?

Task loss-based weight assignment

$$\underset{\Delta w}{\arg\min} \; E[L(x, y, w + \Delta w) - L(x, y, w)]$$

How can we solve this efficiently?

Taylor series approximation

- Problem formulation:

$$\underset{\Delta w}{\arg\min} \quad E\left[L(x, y, w + \Delta w) - L(x, y, w)\right]$$

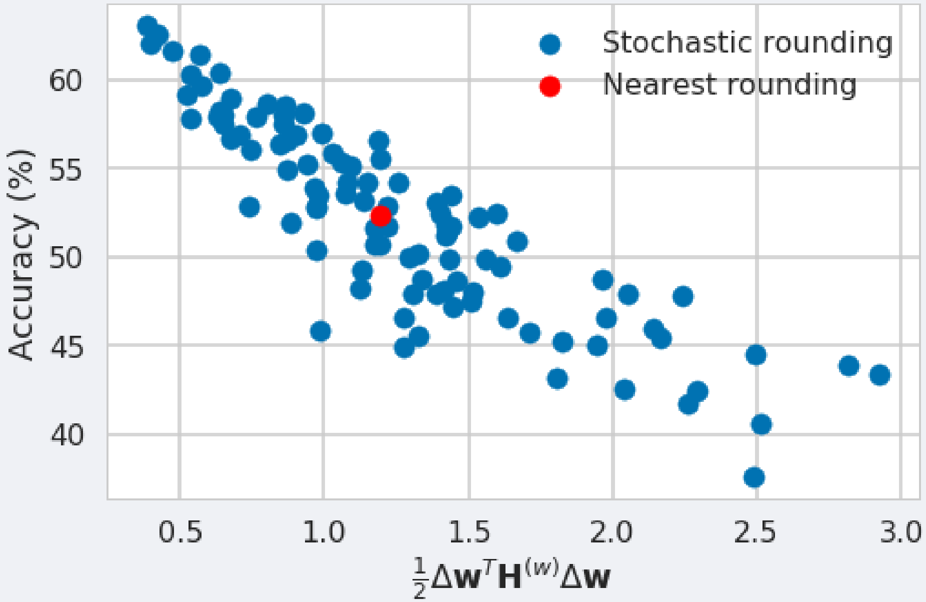
- Taylor series expansion of loss

$$\mathbb{E}\left[\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{w} + \Delta\mathbf{w}) - \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{w})\right]$$

$$= \quad \cancel{\Delta\mathbf{w}^T\mathbf{g}^{(\mathbf{w})}} + \frac{1}{2}\Delta\mathbf{w}^T\mathbf{H}^{(\mathbf{w})}\Delta\mathbf{w} + \cancel{\ldots}$$

- Assumptions:
  - Converged model (ignore gradient term)
  - Small enough $\Delta\mathbf{w}$ (ignore third and higher order terms)
- Final optimization problem

$$\underset{\Delta\mathbf{w}}{\arg\min} \quad \Delta\mathbf{w}^T\mathbf{H}^{(\mathbf{w})}\Delta\mathbf{w}$$



Rounding Method	First Layer (%)
Nearest	52.29
Task loss Hessian	68.62±0.17

4-bit weight quantization of 1st layer of Resnet18.

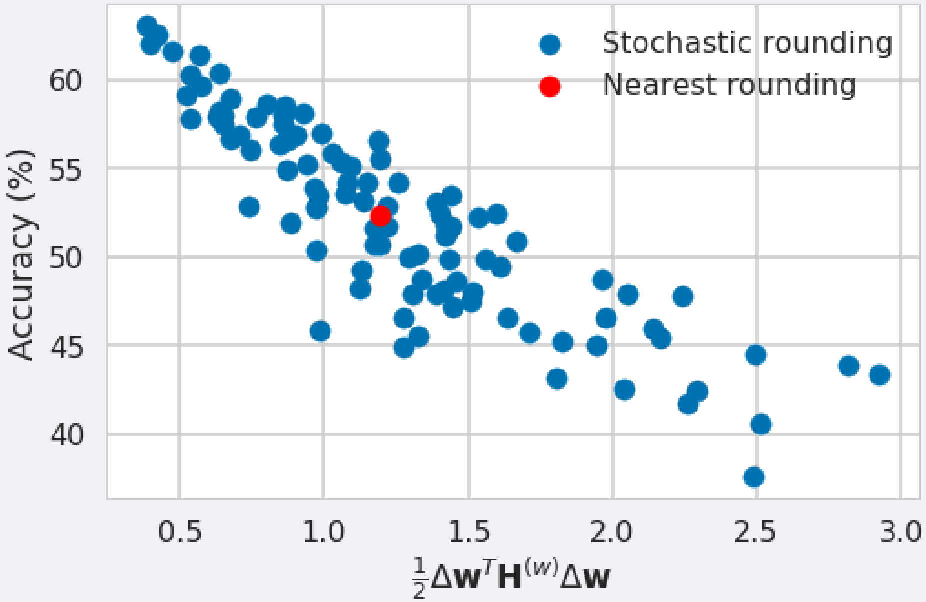
Task loss-based weight assignment

$$\arg\min_{\Delta w} \; \mathbb{E}\left[L\left(x, y, w + \Delta w\right) - L\left(x, y, w\right)\right]$$

Taylor approximation

$$\arg\min_{\Delta \mathbf{w}} \; \Delta\mathbf{w}^T \mathbf{H}^{(\mathbf{w})} \Delta\mathbf{w}$$

Intractable for full network!



Rounding Method	First Layer (%)	All Layers (%)
Baseline (FP32)	69.68	
Nearest	52.29	23.99
Full Hessian	68.62±0.17	N/A

Impact of rounding method for 4-bit weight quantization of Resnet18 in ImageNet.

Layer-wise Hessian approximation

- For a fully-connected layer: $\mathbf{z} = \mathbf{W}\mathbf{x}$

$$\mathbf{H}^{(\mathbf{w})} = \mathbb{E}\left[\mathbf{x}\mathbf{x}^T \otimes \nabla^2_{\mathbf{z}}\mathcal{L}\right]$$

- Assuming diagonal $\nabla^2_{\mathbf{z}}\mathcal{L}$ and constant $\nabla^2_{\mathbf{z}}\mathcal{L}_{i,i}$

$$\begin{aligned}\arg\min_{\Delta\mathbf{w}} \quad \Delta\mathbf{w}^T\mathbf{H}^{(\mathbf{w})}\Delta\mathbf{w} &= \arg\min_{\Delta\mathbf{W}_{k,:}} \quad \Delta\mathbf{W}_{k,:}\,\mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right]\Delta\mathbf{W}_{k,:}{}^T \\ &= \arg\min_{\Delta\mathbf{W}_{k,:}} \quad \mathbb{E}\left[\left(\Delta\mathbf{W}_{k,:}\mathbf{x}\right)^2\right]\end{aligned}$$

- Approximation is applicable to other model efficiency tasks, e.g., pruning
  - Strong relation with layer-wise knowledge distillation

Task loss-based weight assignment

$$\arg\min_{\Delta w} \; E\left[L(x, y, w + \Delta w) - L(x, y, w)\right]$$

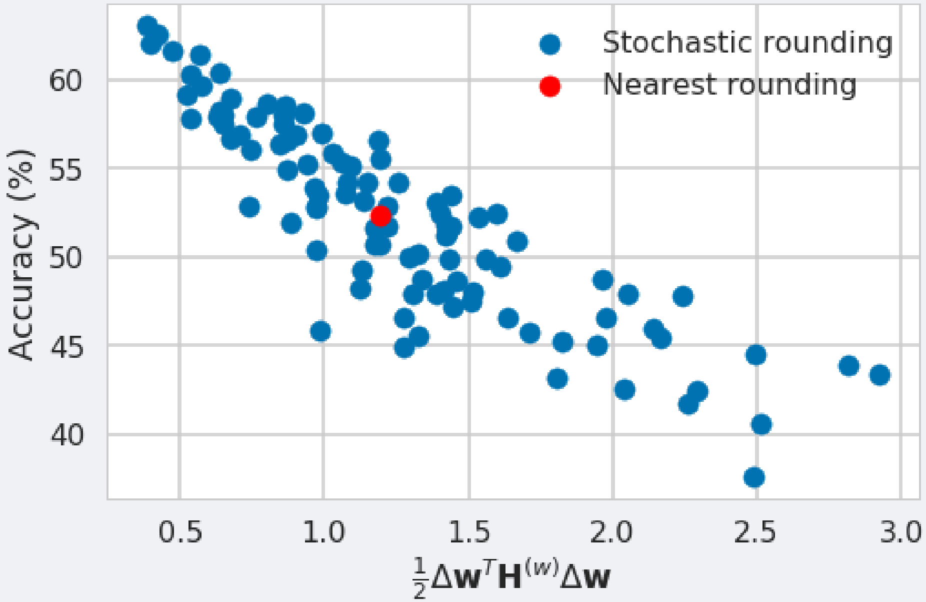
Taylor approximation

$$\arg\min_{\Delta \mathbf{w}} \quad \Delta \mathbf{w}^T \mathbf{H}^{(\mathbf{w})} \Delta \mathbf{w}$$

Hessian approximation

$$\arg\min_{\Delta \mathbf{W}_{k,:}} \quad \mathbb{E}\left[\left(\Delta \mathbf{W}_{k,:}\mathbf{x}\right)^2\right]$$

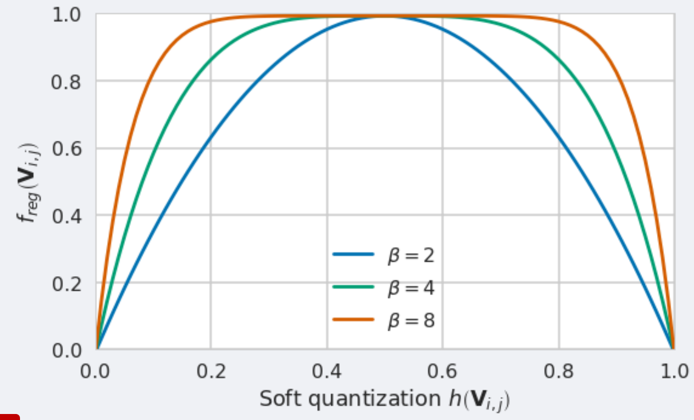
Solving this is NP hard!



Rounding Method	First Layer (%)	All Layers (%)
Baseline (FP32)	69.68	
Nearest	52.29	23.99
Full Hessian	68.62±0.17	N/A
Layer-wise loss	69.39±0.04	65.83±0.14

Impact of rounding method for 4-bit weight quantization of Resnet18 in ImageNet.

AdaRound



- Solve rounding efficiently using continuous relaxation

$$\underset{\mathbf{V}}{\arg\min} \quad \left\| \mathbf{W}\mathbf{x} - \widetilde{\mathbf{W}}\mathbf{x} \right\|_F^2 + \boxed{\lambda f_{reg}(\mathbf{V})}$$

regularizer forces $h(\mathbf{V})$ to be 0 or 1

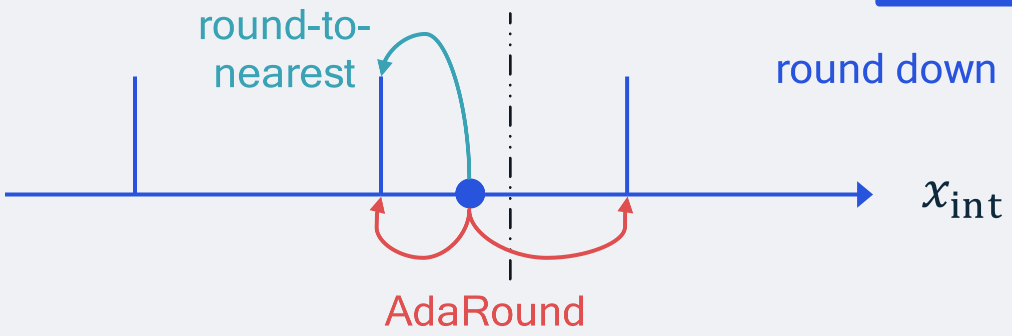
- Where $\widetilde{\boldsymbol{W}}$ are soft-quantized weights:

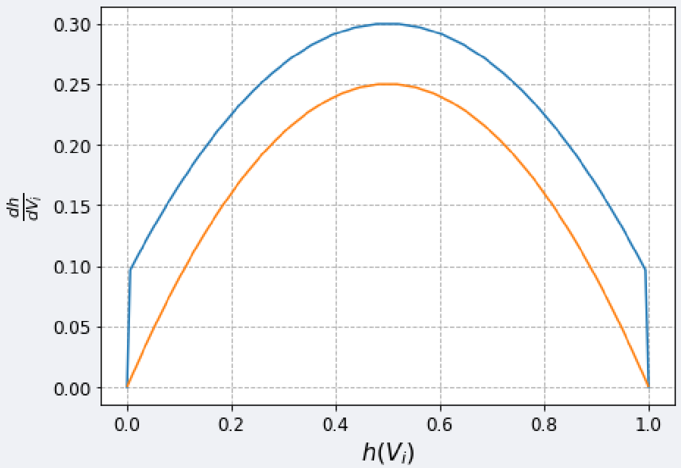
$$h(\mathbf{V}) = \mathrm{clip}\left(\sigma(\mathbf{V})(\zeta - \gamma) + \gamma, 0, 1\right)$$

rectified sigmoid

$$\widetilde{\mathbf{W}} = \mathrm{s} \cdot clip\left(\boxed{\left\lfloor \frac{\mathbf{W}}{\mathrm{s}} \right\rfloor} + \boxed{h(\mathbf{V})}, \mathrm{n}, \mathrm{p}\right)$$

round down + learned value between [0,1]





Task loss-based weight assignment

$$\underset{\Delta w}{\arg\min} \; \mathbb{E}\left[L(x, y, w + \Delta w) - L(x, y, w)\right]$$

Taylor approximation

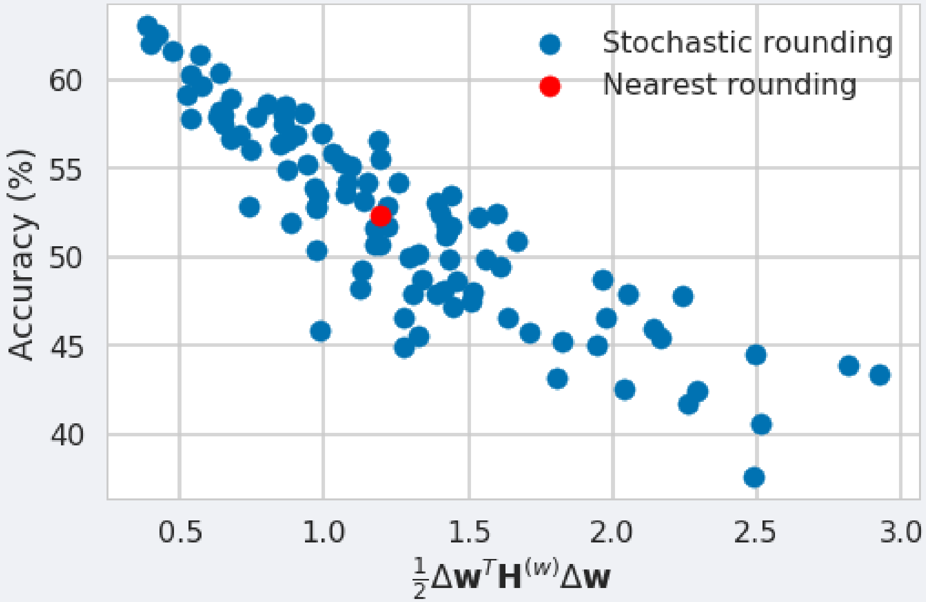
$$\underset{\Delta \mathbf{w}}{\arg\min} \; \Delta \mathbf{w}^T \mathbf{H}^{(\mathbf{w})} \Delta \mathbf{w}$$

Hessian approximation

$$\underset{\Delta \mathbf{W}_{k,:}}{\arg\min} \; \mathbb{E}\left[\left(\Delta \mathbf{W}_{k,:} \mathbf{x}\right)^2\right]$$

Continuous relaxation

$$\underset{V}{\arg\min} \left\| f_a(Wx) - f_a\left(\tilde{W}\hat{x}\right) \right\|_F^2 + \lambda f_{reg}(V)$$



Rounding Method	First Layer (%)	All Layers (%)
Baseline (FP32)	69.68	
Nearest	52.29	23.99
Full Hessian	68.62±0.17	N/A
Local MSE loss	69.39±0.04	65.83±0.14
AdaRound	69.58±0.03	68.71±0.06

Impact of rounding method for 4-bit weight quantization of Resnet18 in ImageNet.

Zooming out a bit

$$\arg\min_{\Delta w} \; E\left[L(x, y, w + \Delta w) - L(x, y, w)\right]$$

Taylor approximation

$$\arg\min_{\Delta \mathbf{w}} \; \Delta\mathbf{w}^T \mathbf{H}^{(\mathbf{w})} \Delta\mathbf{w}$$

Hessian approximation

$$\arg\min_{\Delta \mathbf{W}_{k,:}} \; \mathbb{E}\left[\left(\Delta\mathbf{W}_{k,:}\mathbf{x}\right)^2\right]$$

Continuous relaxation

$$\arg\min_{V} \left\| f_a(Wx) - f_a\left(\tilde{W}\hat{x}\right) \right\|_F^2 + \lambda f_{reg}(V)$$

In most literature this is same, even dates to 1990s (OBD/OBS)

Mostly similar but sometimes different notation/derivation

- $H \approx 2XX^T \rightarrow \text{argmin}_{w_\Delta} w_\Delta X X^T w_\Delta$
- BRECQ uses block-wise Hessian
- Local loss often used elsewhere (e.g. quant params, pruning)

Many different approaches on solving efficiently the weight assignment

Optimal brain damage (LeCun et al., NeurIPS 1990)
Optimal brain surgeon and general network pruning (Hassibi et al., ICNN 1993)
BRECQ: Pushing the limit of Post-Training Quantization by Block Reconstruction (Li et al., ICLR 2021)

BRECQ

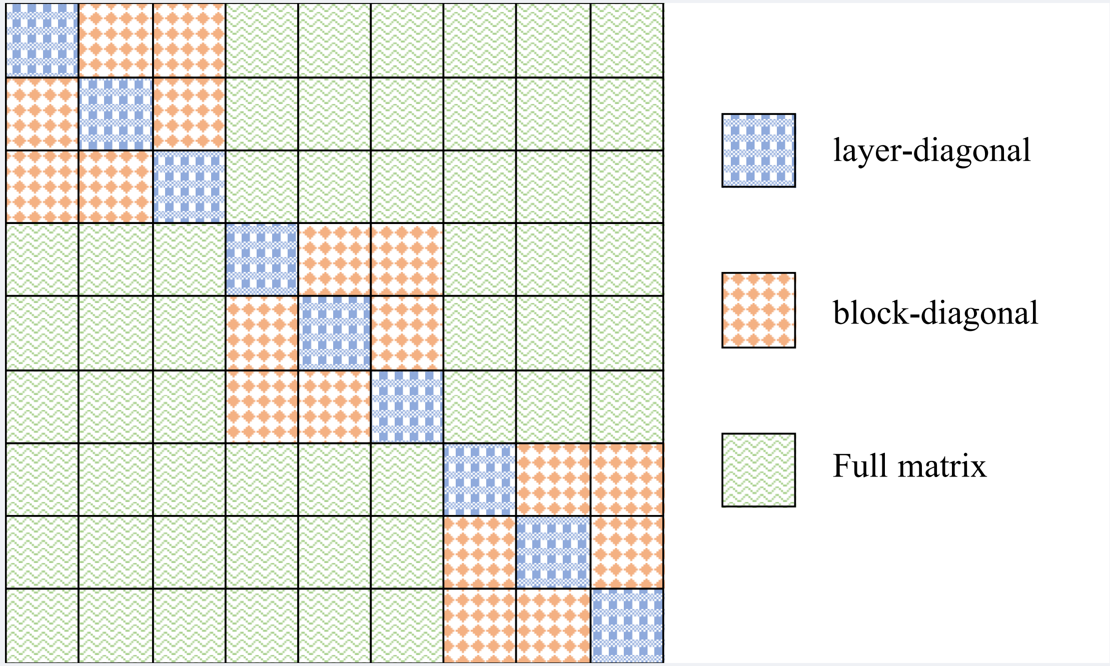


Image courtesy BRECQ paper (fig 1b).

$$\min_{\hat{\mathbf{w}}} \mathbb{E}\left[\Delta\mathbf{z}^{(\ell),\mathsf{T}}\mathbf{H}^{(\mathbf{z}^{(\ell)})}\Delta\mathbf{z}^{(\ell)}\right] = \min_{\hat{\mathbf{w}}} \mathbb{E}\left[\Delta\mathbf{z}^{(\ell),\mathsf{T}}\mathrm{diag}\left((\frac{\partial L}{\partial \mathbf{z}_1^{(\ell)}})^2, \dots, (\frac{\partial L}{\partial \mathbf{z}_a^{(\ell)}})^2\right)\Delta\mathbf{z}^{(\ell)}\right]$$

- Use Fisher Information Matrix (FIM) to approximate cross-layer dependencies.
- Relaxes diagonal $\nabla^2_{\mathbf{z}}\mathcal{L}$ and constant $\nabla^2_{\mathbf{z}}\mathcal{L}_{i,i}$
- Algorithm details:
  - Uses AdaRound continuous relaxation
  - Use LSQ to jointly learn activation ranges
  - Introduced additional mixed precision
- Shows clear improvement, SOTA at the time

Model	Layer	Block	Stage	Net
ResNet-18	65.19	**66.39**	66.01	54.15
MobileNetV2	52.13	**59.67**	54.23	40.76

Table 1 from BRECQ paper.

Learned Step Size Quantization (Esser et al., ICLR 2020)
BRECQ: Pushing the limit of Post-Training Quantization by Block Reconstruction (Li et al., ICLR 2021)

OBQ and GPTQ

Optimal Brain Quantization (OBQ)

- Generalizes Optimal Brain Surgeon (OBS)
- Layer-wise greedy and iterative procedure
  - Quantizes weights row by row with RTN
  - Updates remaining weights to compensate for error

$$w_q = \mathrm{argmin}_{w_q} \frac{(\mathrm{quant}(w_q) - w_q)^2}{[\mathrm{H}_F^{-1}]_{qq}}, \quad \delta_F = -\frac{w_q - \mathrm{quant}(w_q)}{[\mathrm{H}_F^{-1}]_{qq}} \cdot (\mathrm{H}_F^{-1})_{:,q}$$

GPTQ: scaling OBQ to LLMs

- Independent rows → update H_F once per column
- Batched updates for multiple columns
- Cholesky reformulation for stability and efficiency

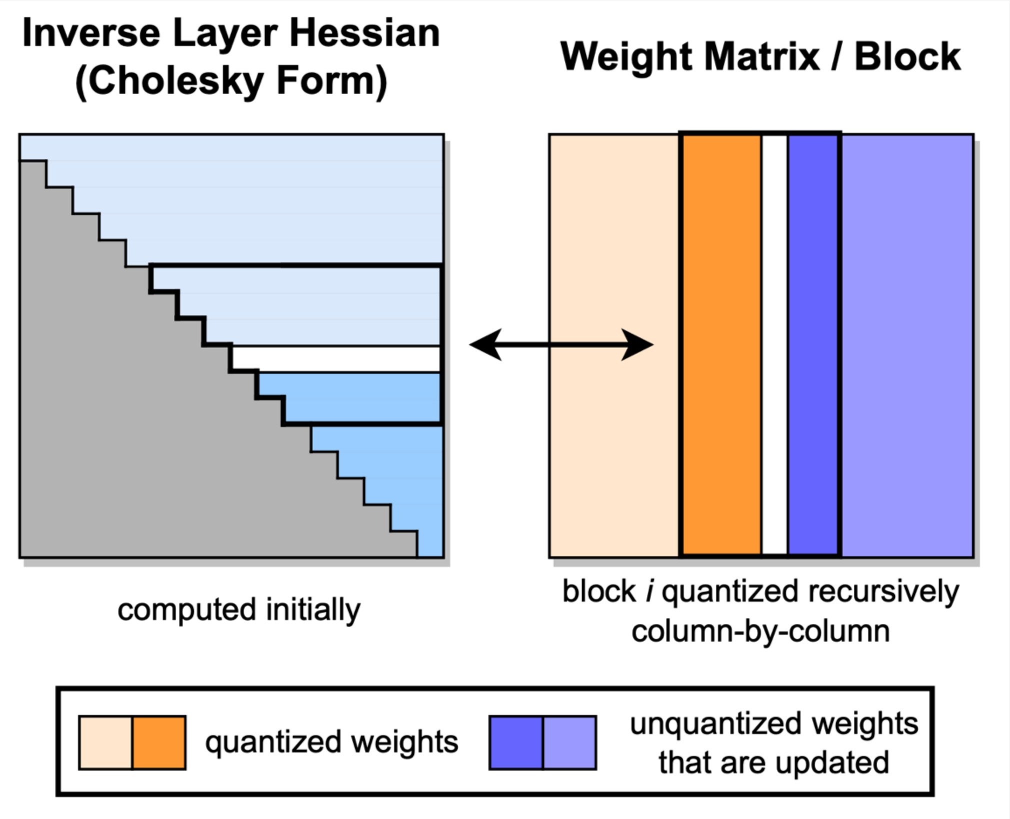


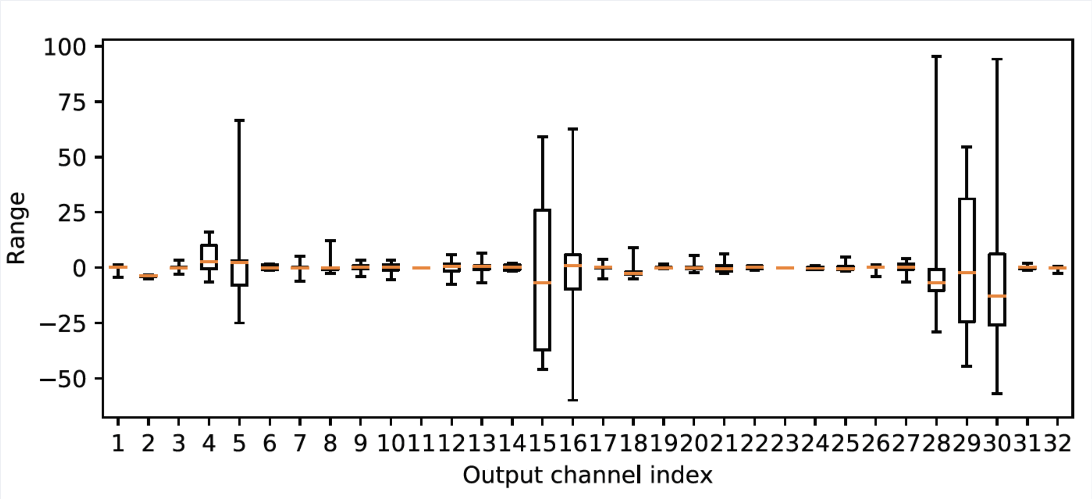
Image courtesy GPTQ paper (fig 2).

Optimal brain surgeon and general network pruning (Hassibi et al., ICNN 1993)
Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning (Frantar et al. NeurIPS 2022)
GPTQ: Accurate Post-Training Quantization for Genrative Pre-trained Transformers (Frantar et al., ICLR 2023)

Function preserving transformations

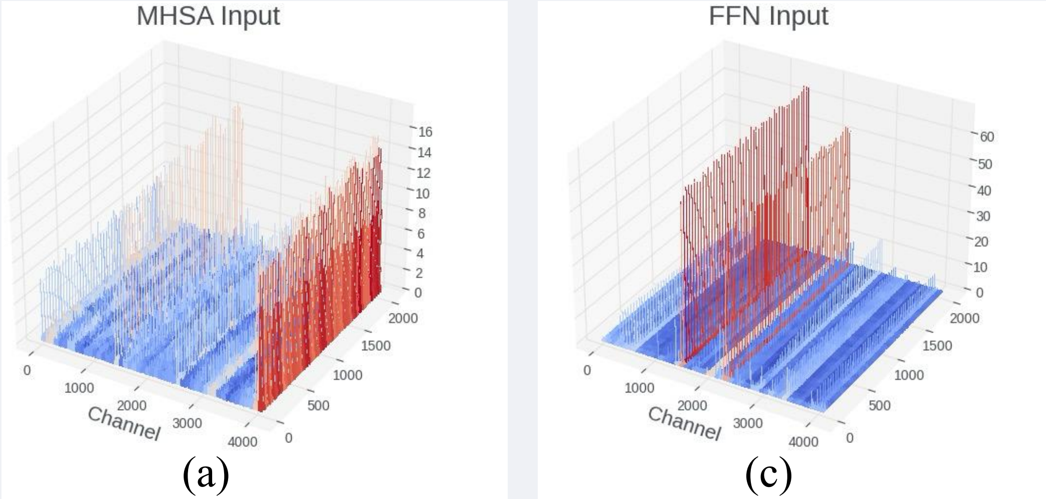
Outliers are common in neural networks

Weight outliers



Weight distribution in MobileNetV2.
Image courtesy DFQ (fig 2).

Activation outliers

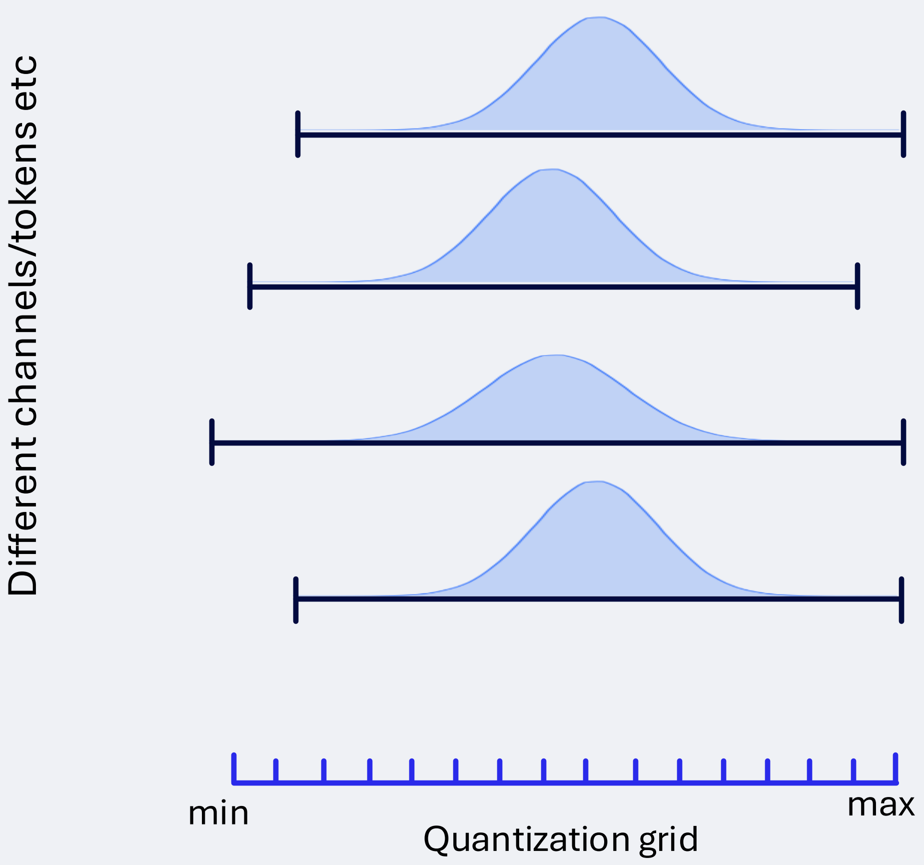


Activation distribution in Llama v2 7B.
Image courtesy SpinQuant (fig 2).

NB: you will hear in the next session more on the structure and origin of outliers in LLMs.

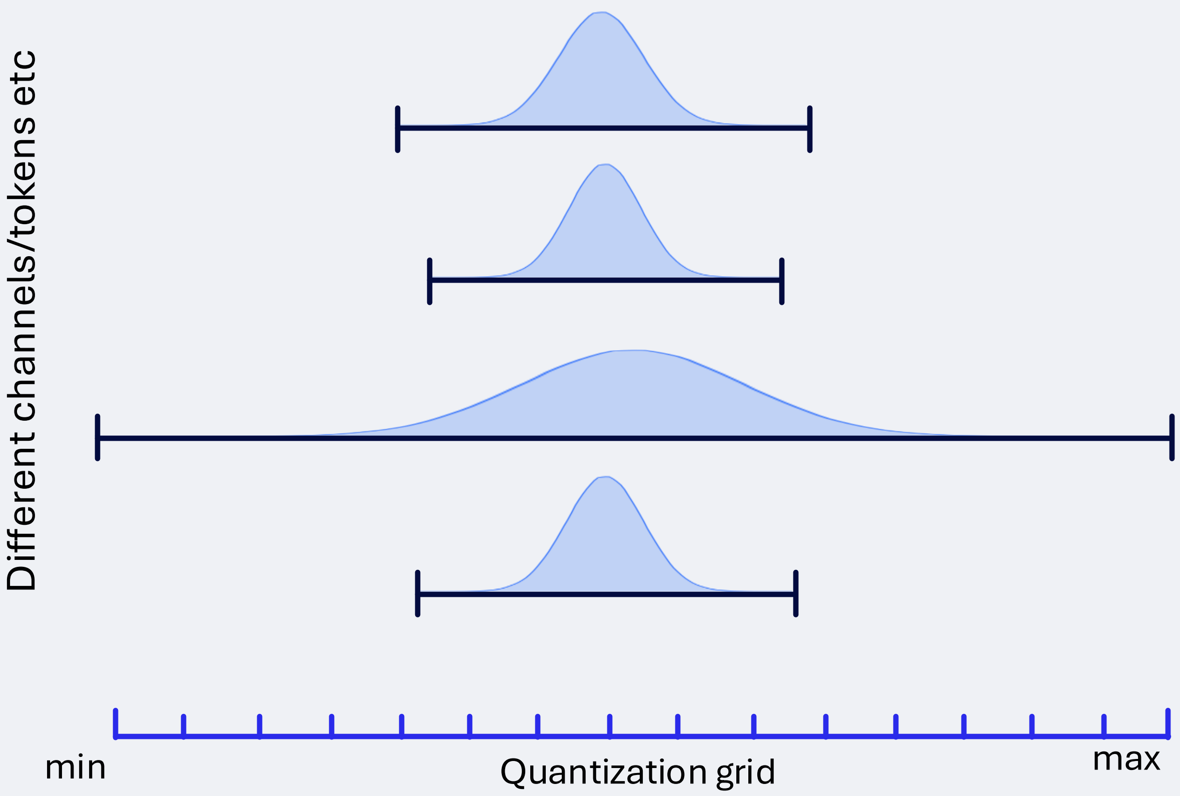
Data Free Quantization Through Weight Equalization and Bias Correction (Nagel et al., ICCV 2019)
SpinQuant: LLM Quantization with Learned Rotations (Liu et al., ICLR2025)

Why are outlier problematic for uniform quantization



Easy to quantize!

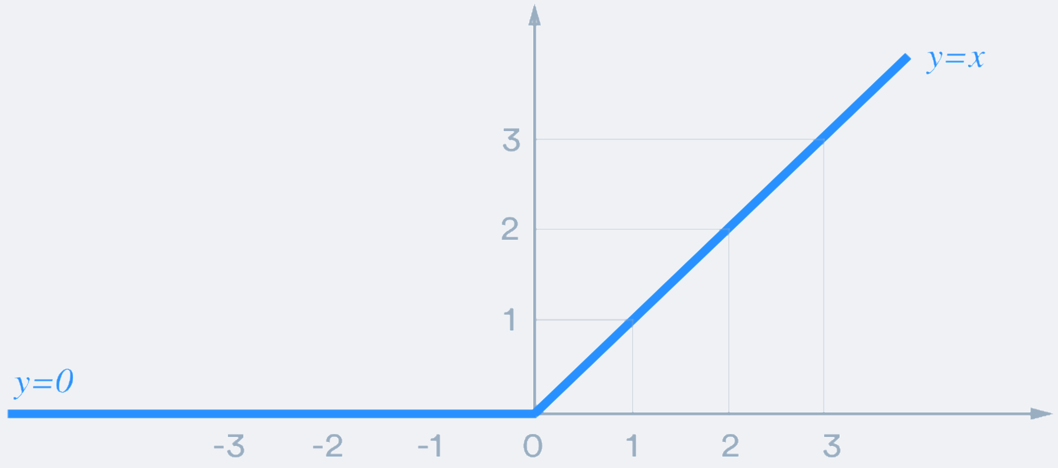
Sufficient precision for all channels/tokens



Low precision on most channels!

High clipping error for outlier channel/tokens!

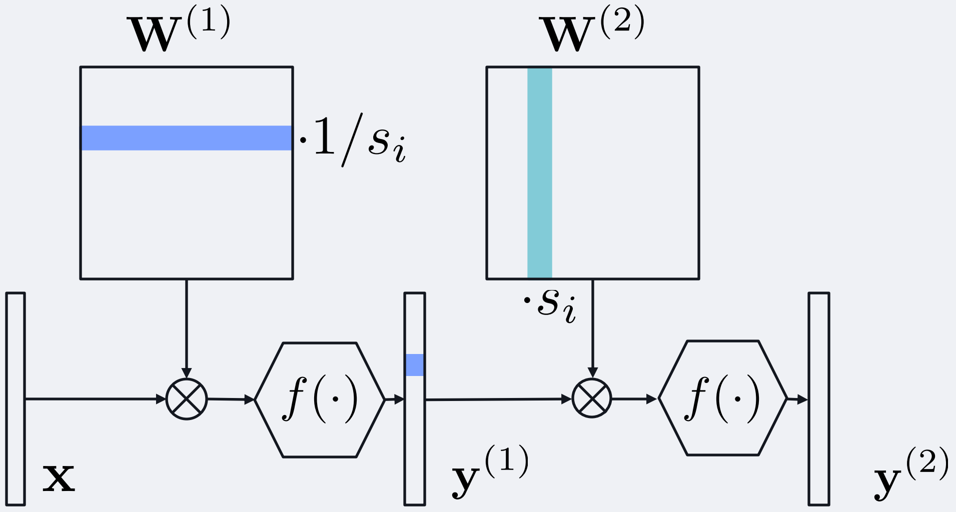
DFQ: Cross-layer equalization



$$\mathrm{ReLU}(x) = \max(0, x)$$

ReLU is scale-equivariant

$$\mathrm{ReLU}(\boldsymbol{s}x) = \boldsymbol{s} \cdot \mathrm{ReLU}(x)$$



We can scale two neighboring layers together to optimize it for quantization

$$y = W_2 \cdot ReLU(W_1 x) = W_2 S \cdot ReLU(S^{-1} W_1 x)$$

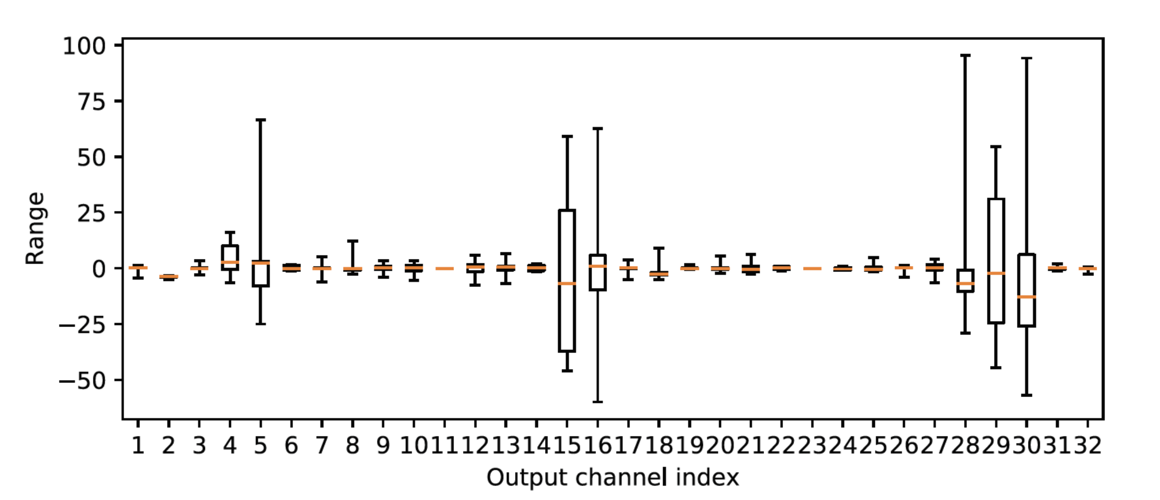
$$\widehat{W}_1 = q(S^{-1} W_1) \qquad \widehat{W}_2 = q(W_2 S)$$

With $S = diag(s)$

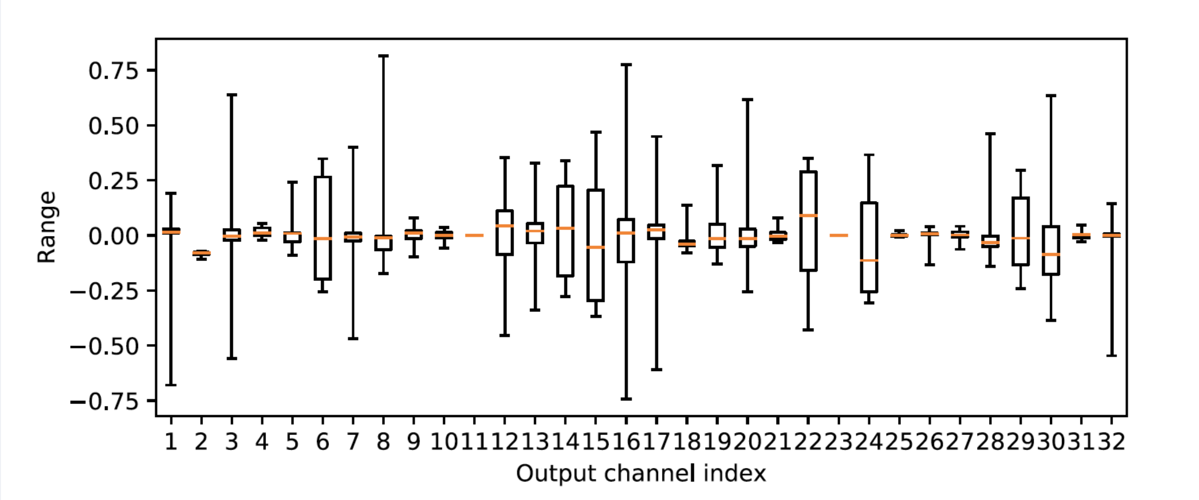
Data Free Quantization Through Weight Equalization and Bias Correction (Nagel et al., ICCV 2019)
Same, Same But Different - Recovering Neural Network Quantization Error Through Weight Factorization (Meller et al., ICML 2019)

Cross-layer equalization significantly improves accuracy

Original weight ranges



After cross-layer equalization



Model	FP32	INT8
Original Model	71.72	0.12
CLE	71.70	69.91
CLE + absorbing bias	71.57	**70.92**

+69.8

ImageNet validation accuracy (%) for MobileNetV2

SmoothQuant

- Activation in LLM have very strong outliers
- Use scaling to migrate difficulty to weights
- Enables INT8 for weights and activations

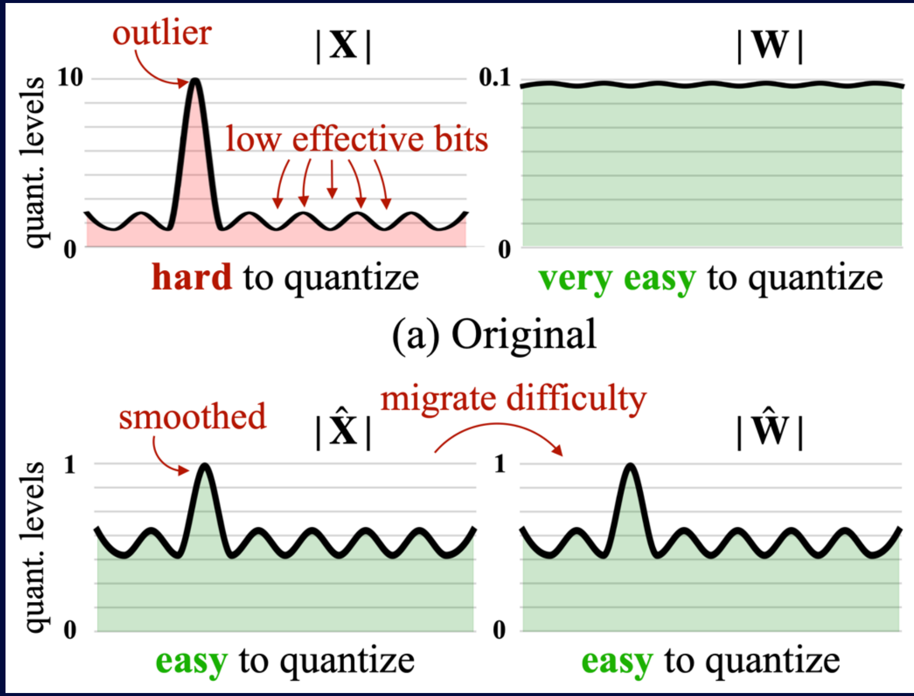


Image courtesy SmoothQuant paper (fig 2).

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models (Xiao et al., ICML 2023)

AWQ

- Salient weights are important to protect
- Find salient weight channels based on activation distribution
- Scale salient weights to minimize quantization error

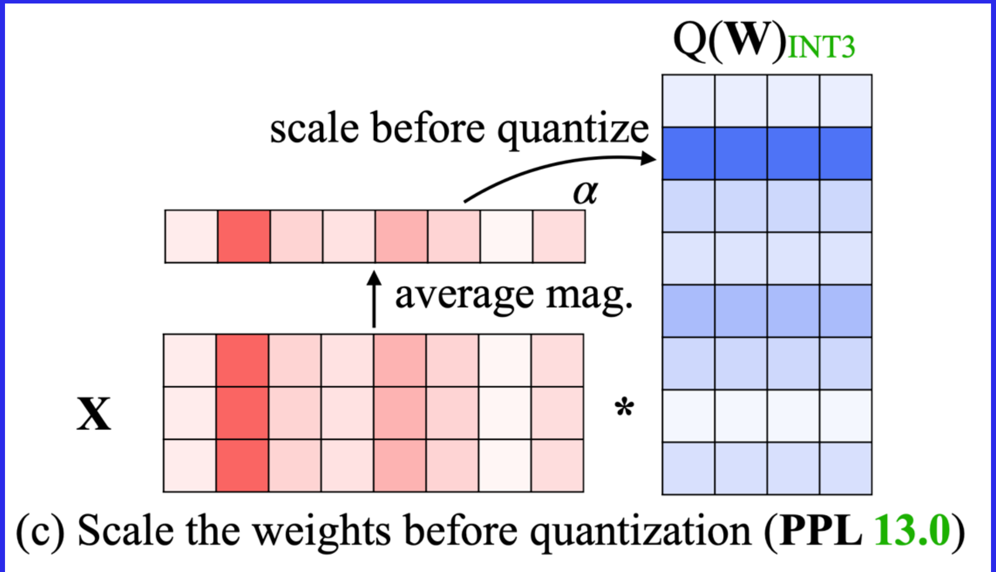


Image courtesy AWQ paper (fig 2).

AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration (Lin et al., MLSys 2024)

OmniQuant

- Learnable equivalent transform (LET)

$$\mathbf{Y} = \mathbf{X}\mathbf{W} + \mathbf{B} = [\underbrace{(\mathbf{X} - \delta) \oslash s}_{\tilde{\mathbf{X}}}] \cdot [\underbrace{s \odot \mathbf{W}}_{\tilde{\mathbf{W}}}] + [\underbrace{\mathbf{B} + \delta\mathbf{W}}_{\tilde{\mathbf{B}}}]$$

$$\mathbf{Y} = Q_a(\tilde{\mathbf{X}})Q_w(\tilde{\mathbf{W}}) + \widetilde{\mathbf{B}},$$

- Learnable weight clipping (LWC)

$$\mathbf{W_q} = \mathrm{clamp}(\lfloor \frac{\mathbf{W}}{h} \rceil + z, 0, 2^N - 1)$$

$$h = \frac{\gamma \max(\mathbf{W}) - \beta \min(\mathbf{W})}{2^N - 1}, z = -\lfloor \frac{\beta \min(\mathbf{W})}{h} \rceil$$

- Trained on block-wise quantization error (cf. AdaRound, BRECQ)

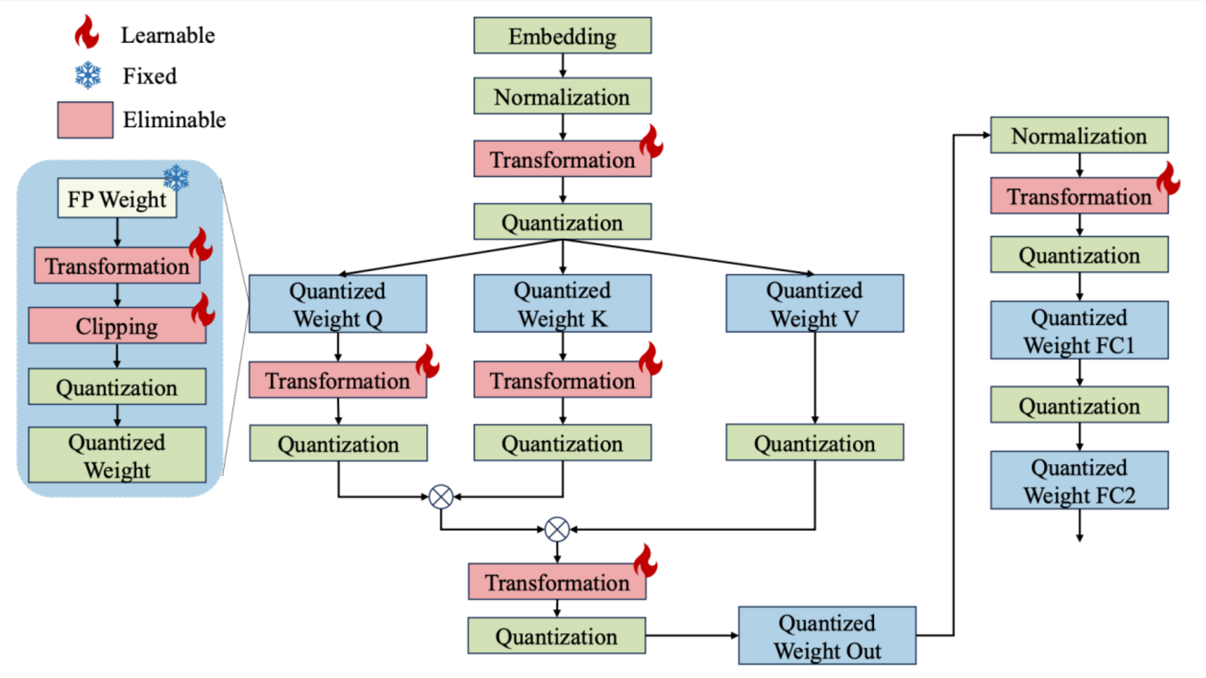


Image courtesy OmniQuant paper (fig 3).

Note, all transformations are mergeable!

OmniQuant: Omnidirectionally Calibrated Quantization for Large Language Models (Shao et al., ICLR 2024)

Function preserving transformations (FPTs)

- Generalize channel-wise scaling to any function preserving transformations (FPTs)

$$Q(X)Q(W) \rightarrow Q(XT)Q(T^{-1}W)$$

- Key properties of function preserving transformations (FPTs):
  1. Function-preservation: Should not change network output. In practice this means the FPT needs to have an inverse operation and operations between needs to commute under the FPT.
  2. Expressivity: Transforms with a continuous parameterization and more degrees of freedom are desirable. They allow optimization with SGD and have more flexibility to reduce quantization error.
  3. Compute overhead: Ideally FPTs are mergeable to adjacent operations. If not, they injure inference time overhead which should be minimal.

- Rotations are commonly used reduce outliers through channel mixing
  - Walsh-Hadamard transformations is commonly used as cheap online transform ($O(d\log_2(d))$)
  - First introduced as incoherence processing in QuIP and QuIP# for extreme weight compression
  - Rotation commutes with RMSNorm (SliceGPT): $\mathrm{RMSNorm}(X)T = \mathrm{RMSNorm}(XT)$

FPTQuant: Function-Preserving Transforms for LLM Quantization (Van Breugel et al., arXiv 2025)
QuIP: 2-Bit Quantization of Large Language Models With Guarantees (Chee et al., NeurIPS 2023)
QuIP#: Even Better LLM Quantization with Hadamard Incoherence and Lattice Codebooks (Tseng et al., ICML 2024)
SliceGPT: Compress Large Language Models by Deleting Rows and Columns (Ashkboos et al., ICLR 2024)

Transformations applied for LLM quantization

QuaRot

- Shows that Hadamard makes activation quantization easy
- Online Hadamard in attention to remove outliers from keys and values
- Enables full INT4 LLM inference (weight, activations, KV cache)

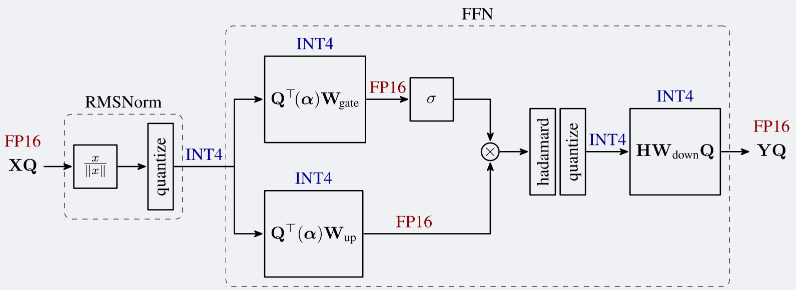


Image courtesy QuaRot paper (fig 3).

SpinQuant

- Learned rotations end-to-end
- Random and Hadamard rotation show substantial variance
- Uses Cayley SGD to optimize rotations on Stiefel manifold

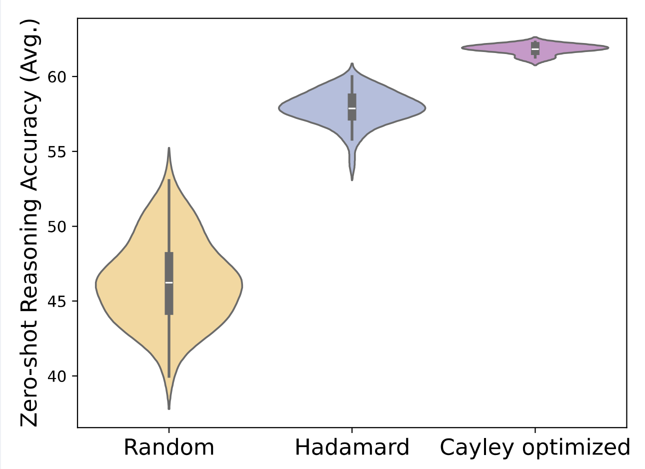
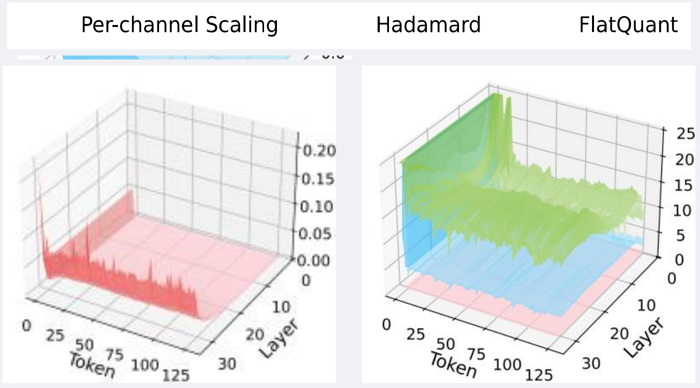


Image courtesy SpinQuant paper (fig 4).

FlatQuant

- Significance of flat distributions
- Fast affine transformation using Kronecker decomposition
- Fused transformation in joint quantization kernel

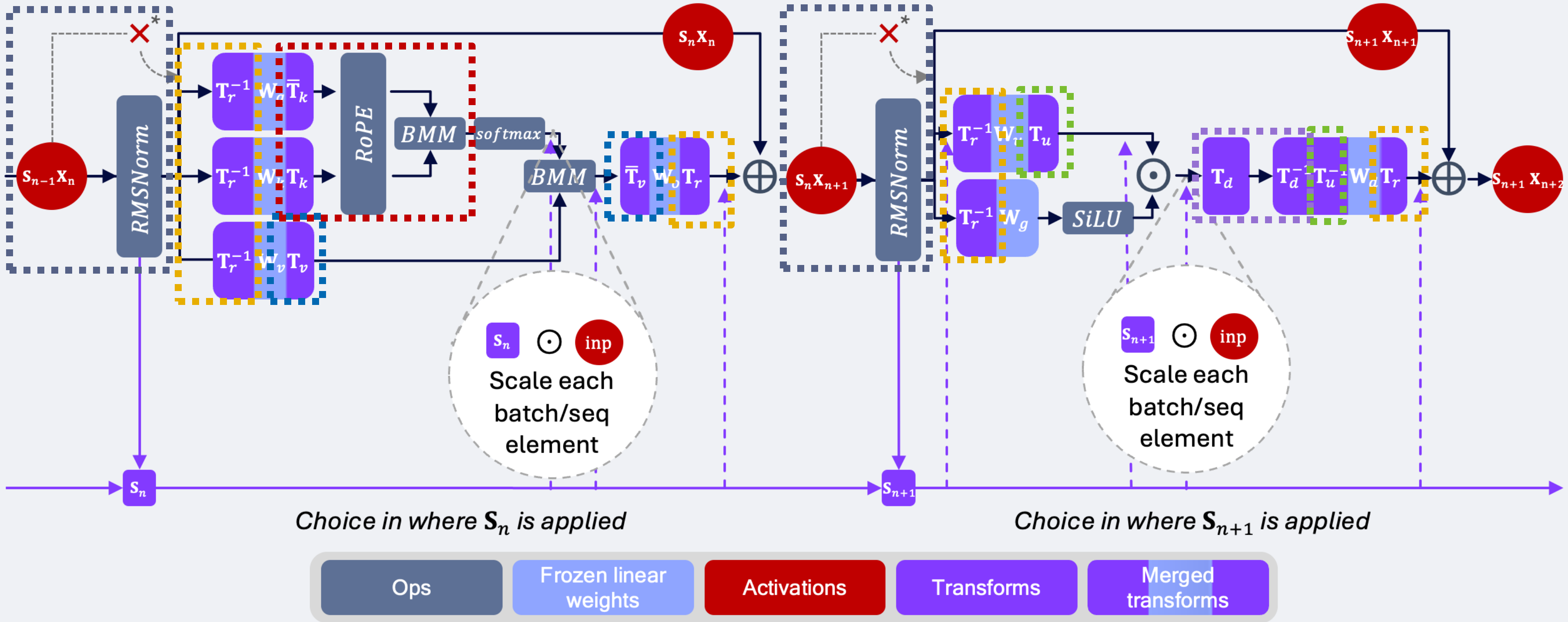


(c) FLATQUANT. (d) Stacked View.

Image courtesy FlatQuant paper (fig 2).

QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs (Ashkboos et al., NeurIPS 2024)
SpinQuant: LLM Quantization with Learned Rotations (Liu et al., ICLR2025)
FlatQuant: Flatness Matters for LLM Quantization (Sun et al., ICML 2025)

FPTQuant: maximizing the amount of mergeable transforms



Introduce novel and fully mergeable transforms:

- PreRoPE transform: scaled 2x2 rotation
- Value transform: linear transform (full matrix per-head)
- Up/down scaling: per-channel scaling
- Pseudo-dynamic residual scaling: normalize residual

Reuse some existing transforms:

- Residual transform: joint over all layers (same as R1 in SpinQuant)
- Down project transform: online Hadamard (same as R4 in SpinQuant)

FPTQuant: Function-Preserving Transforms for LLM Quantization (Van Breugel et al., preprint 2025)

HadaNorm: mean-centered Hadamard transform

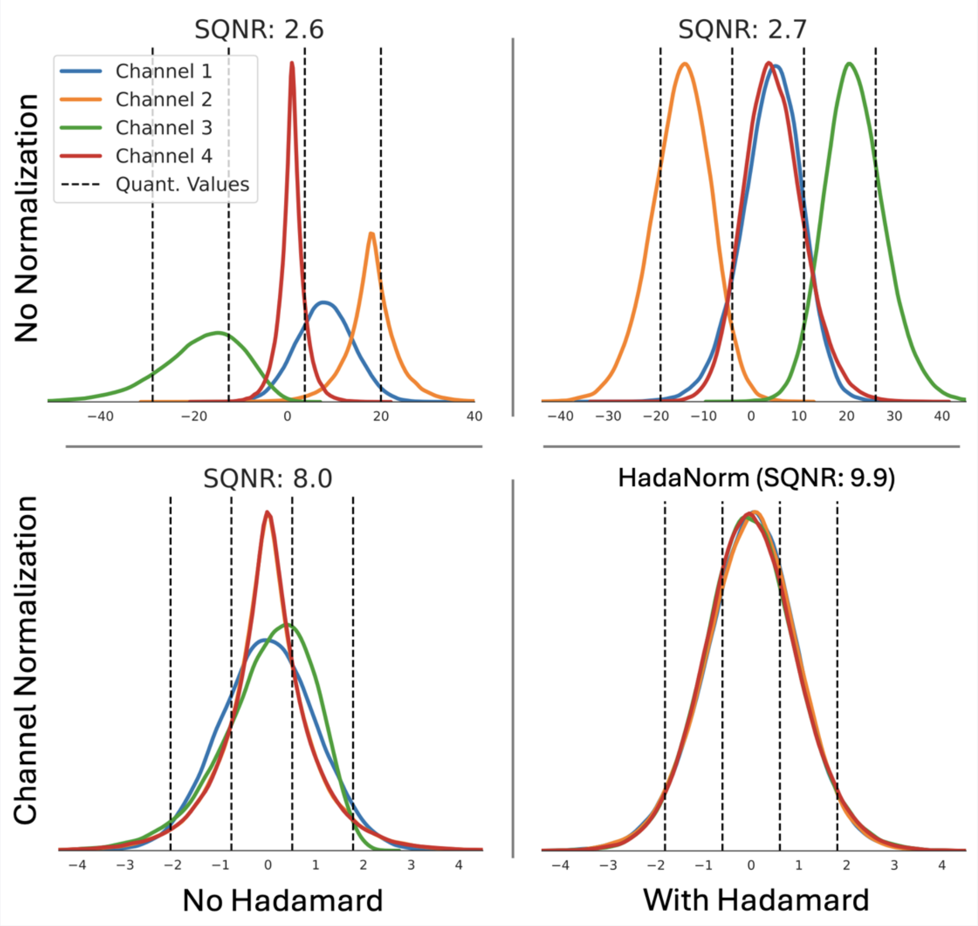


Image courtesy HadaNorm paper (fig 1).

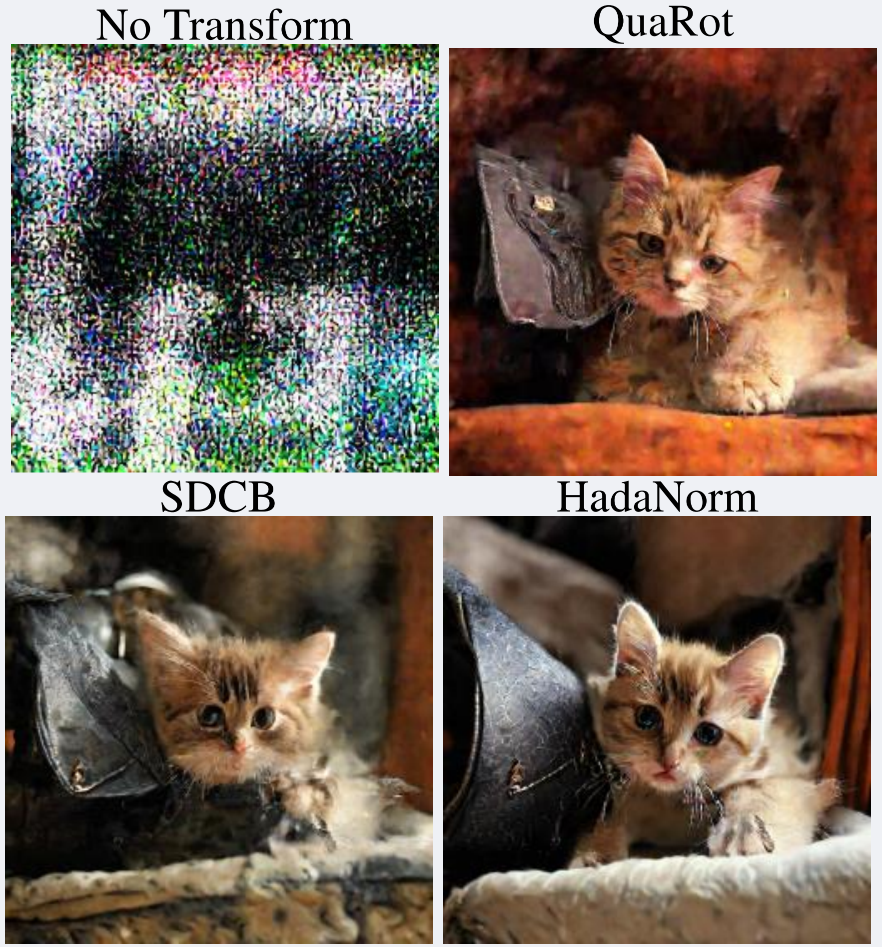


Image courtesy HadaNorm paper (fig 4).

HadaNorm: Diffusion Transformer Quantization through Mean-Centered Transformations (Federici et al., preprint 2025)

Summary PTQ

- Range setting is important to trade-off between rounding and clipping error
- Optimizing weight assignment based on Hessian approximation significantly improves low bit weight quantization
- Outliers are a common issue and function preserving transformations (FTPs) can make distributions more quantization friendly
- NB: there are many other effective algorithms that do not fall in these 3 main buckets
  - E.g. bias correction, channel splitting, high precision outliers and many more

Part 2:
Quantization-aware training

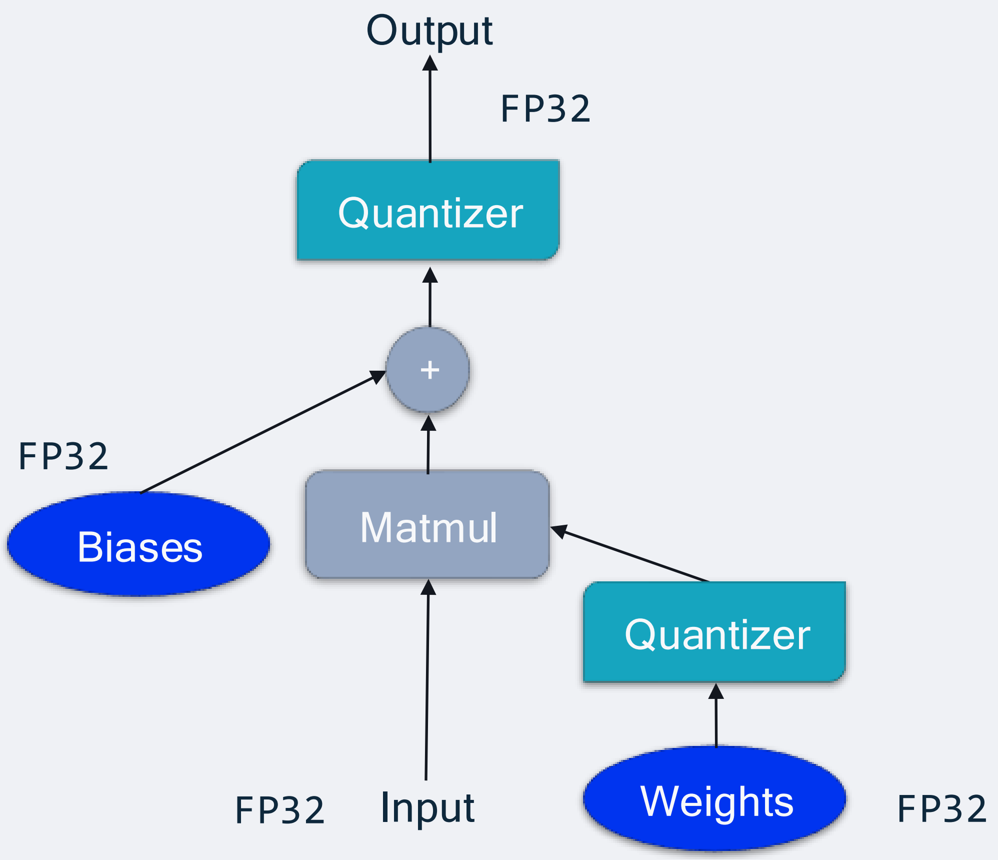
Quantization-Aware Training

- Train with *simulated quantization* to achieve best inference time accuracy
- Quantizers discretize the input data

Key challenges:

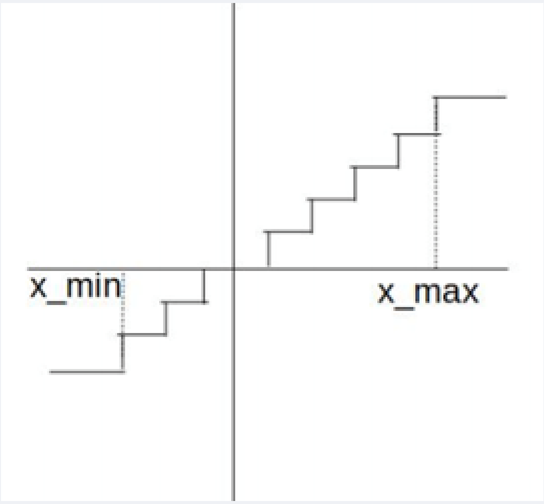
- Discretization is non-differentiable
  - E.g. rounding, codebook assignment etc
- Learning quantization parameters
  - E.g. scale of quantization grid, codebook entries etc

Simulated quantization



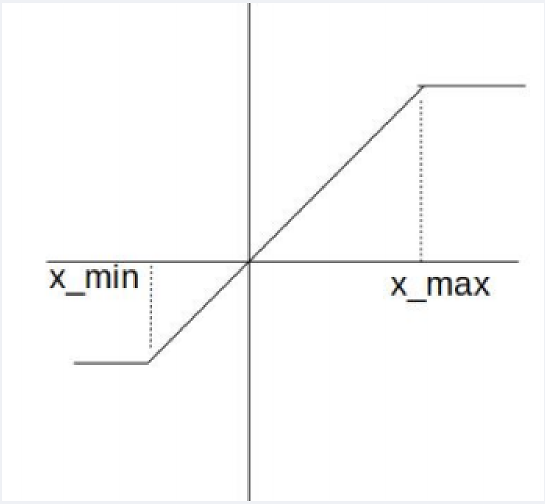
Straight-through estimator (STE)

- Approximated non-differentiable operation with straight-through estimator (STE)
- Most commonly used approach and highly effective



Actual forward pass

$$\frac{\partial \lfloor x \rceil}{\partial x} = 1$$



Simulated forward pass

Known drawbacks of STE:

- STE gradient is biased
- Weights don't converge, they can oscillate

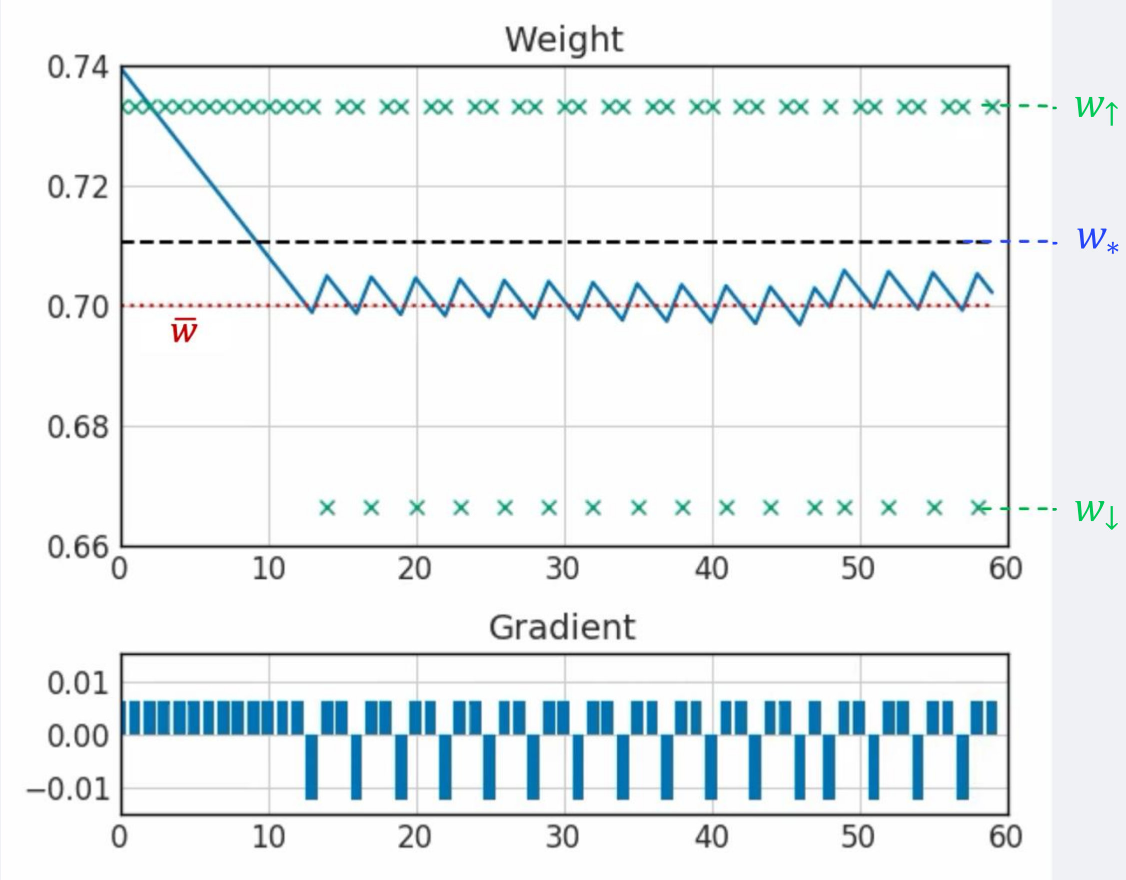
Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation (Bengio et al., 2013)

Oscillating weights in QAT

- Example regression problem:
  - Latent weight: w
  - Quantized weight: $q(w) = s_w \cdot \text{round}(w/s_w)$
  - Objective: $\min_w \mathcal{L}(w) = (w_* - q(w))^2$

- Rounding is approximated by STE[1]:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial q(w)} = \begin{cases} w_* - w_\uparrow, & \text{if } w \geq \bar{w} \\ w_* - w_\downarrow, & \text{if } w < \bar{w} \end{cases}$$



Overcoming Oscillations in Quantization-Aware Training (Nagel et al., ICML 2022)

Alternatives to STE

Stochastic rounding

- Round proportional to distance

$$\text{round}(x) = \begin{cases} \lfloor x \rfloor & \text{w.p. } x - \lceil x \rceil \\ \lceil x \rceil & \text{w.p. } x - \lfloor x \rfloor \end{cases}$$

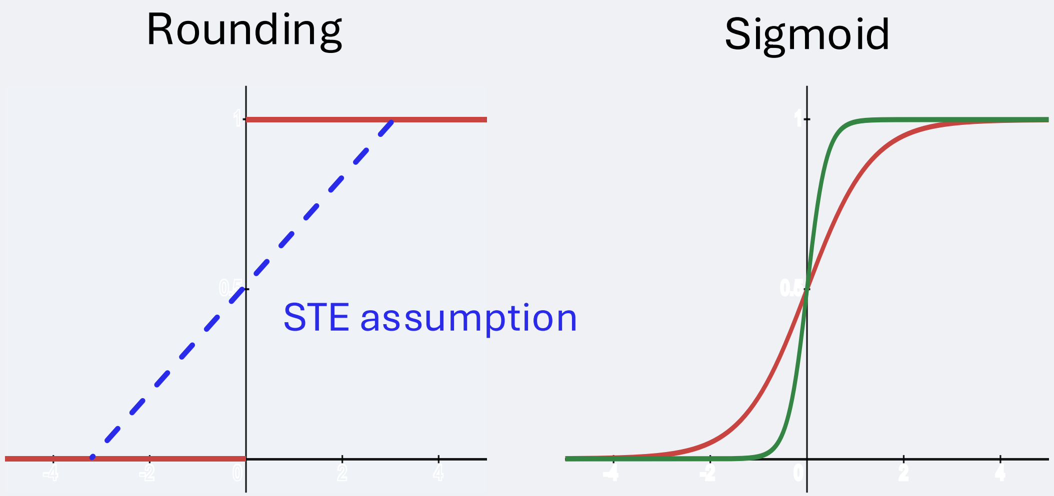
- Unbiased estimator:

$$\mathbb{E}[\text{round}(x)] = x$$

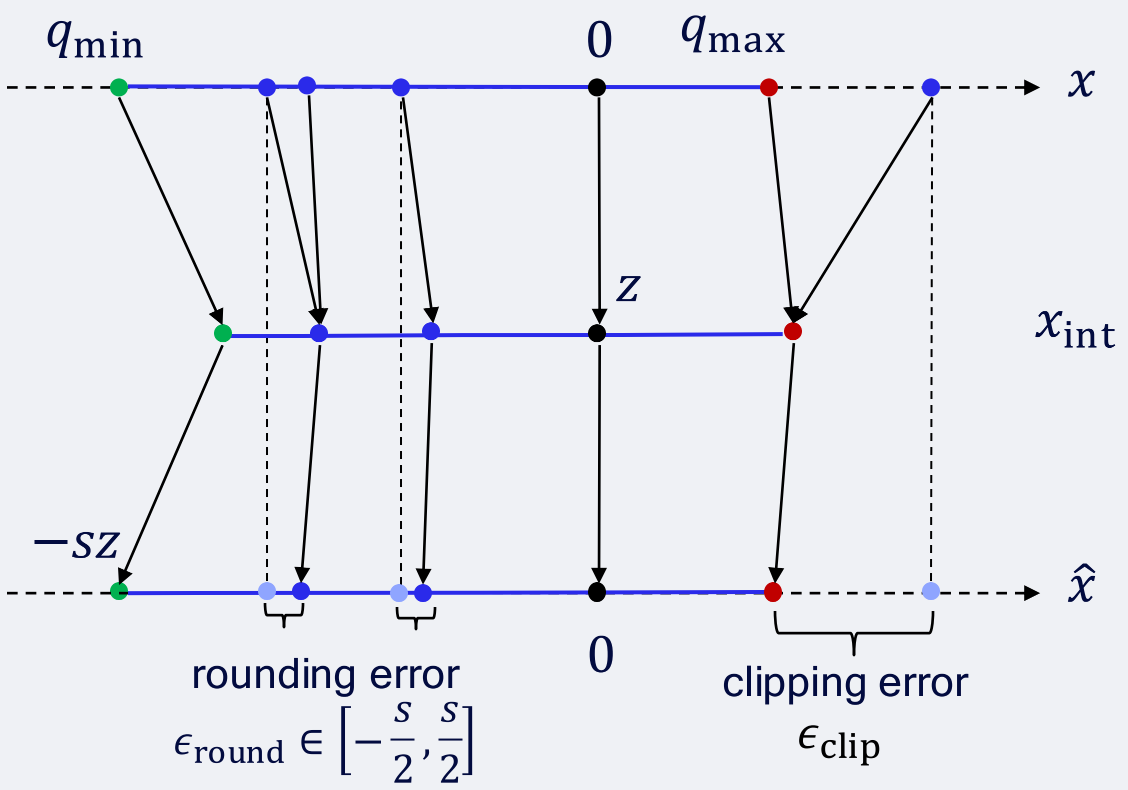
Deep learning with limited numerical precision (Gupta et al., ICML 2015)
Quantization Networks (Yang et al., CVPR 2019)

Approximate rounding function

- Approximate quantization operation with differentiable non-linear mapping
- Stacked sigmoid with temperature



Learning the quantization parameters



Quantization parameters are learnable when STE is applied to rounding only

$$\boldsymbol{X}_{\text{int}} = \text{clamp}\left(\text{round}\left(\frac{\boldsymbol{X}}{s}\right) + z, \min = 0, \max = 2^b - 1\right)$$

$$\widehat{\boldsymbol{X}} = \boldsymbol{s}\,(\boldsymbol{X}_{\text{int}} - \boldsymbol{z})$$

Through task loss gradients, we find the optimal trade-off between ϵ_{clip} & ϵ_{round}

Note, various parameterization might lead to different gradients and learning behaviour

Learned step size quantization (Esser et al., ICLR 2020)
Trained uniform quantization for accurate and efficient neural network inference on fixed-point hardware (Jain et al., MLSys 2020)
Lsq+: Improving low-bit quantization through learnable offsets and better initialization (Bhalgat et al., CVPRW 2020)

Summary QAT

- Discretization is non-differentiable and requires gradient approximation
- Straight-through estimator (STE) works extremely well in practice
- Quantization parameters can be learned end-to-end using auto-grad

FastForward

neural network quantization

pytorch

python first

imperative workflow

extendible

```
import fastforward as ff

[...]

ff.quantize_model(model)
initialize_quantizers(model) # user provided

with ff.estimate_ranges(model, ff.range_estimation.mse):
  for batch in data[:20]:
    model(batch)
```

Example: PTQ using FastForward

- Neural Network quantization library for PyTorch
- Focus on *prototyping* and *experimentation*; embracing PyTorch's imperative workflow
- Find more at on GitHub: github.com/qualcomm-ai-research/fastforward

Code

Docs

Thanks to all my amazing collaborators


Mart van Baalen


Tijmen Blankevoort


Marios Fournarakis


Rana Ali Amjad


Yelysei Bondarenko


Andrey Kuzmin


Christos Louizous


Ties van Rozendaal


Andrii Skliar


Max Welling


Paul Whatmough


Boris van Breugel


Marco Federici


Jorn Peters


Riccardo Del Chiaro


Ivan Koryakovskiy


Nilesh Prasad Pandey


Yash Bhalgat

Markus Nagel
markusn@qti.qualcomm.com
https://www.linkedin.com/in/markusnagel/
Google Scholar

Thank you

Nothing in these materials is an offer to sell any of the components or devices referenced herein.

References in this presentation to "Qualcomm" may mean Qualcomm Incorporated, Qualcomm Technologies, Inc., and/or other subsidiaries or business units within the Qualcomm corporate structure, as applicable. Qualcomm Incorporated includes our licensing business, QTL, and the vast majority of our patent portfolio. Qualcomm Technologies, Inc., a subsidiary of Qualcomm Incorporated, operates, along with its subsidiaries, substantially all of our engineering, research and development functions, and substantially all of our products and services businesses, including our QCT semiconductor business.

Snapdragon and Qualcomm branded products are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patented technologies are licensed by Qualcomm Incorporated.

Follow us on:

For more information, visit us at qualcomm.com & qualcomm.com/blog