Neural Network Quantization

2022년도 하계 세미나

Sogang University
Vision & Display Systems Lab, Dept. of Electronic Engineering

Presented By
곽재호

Outline

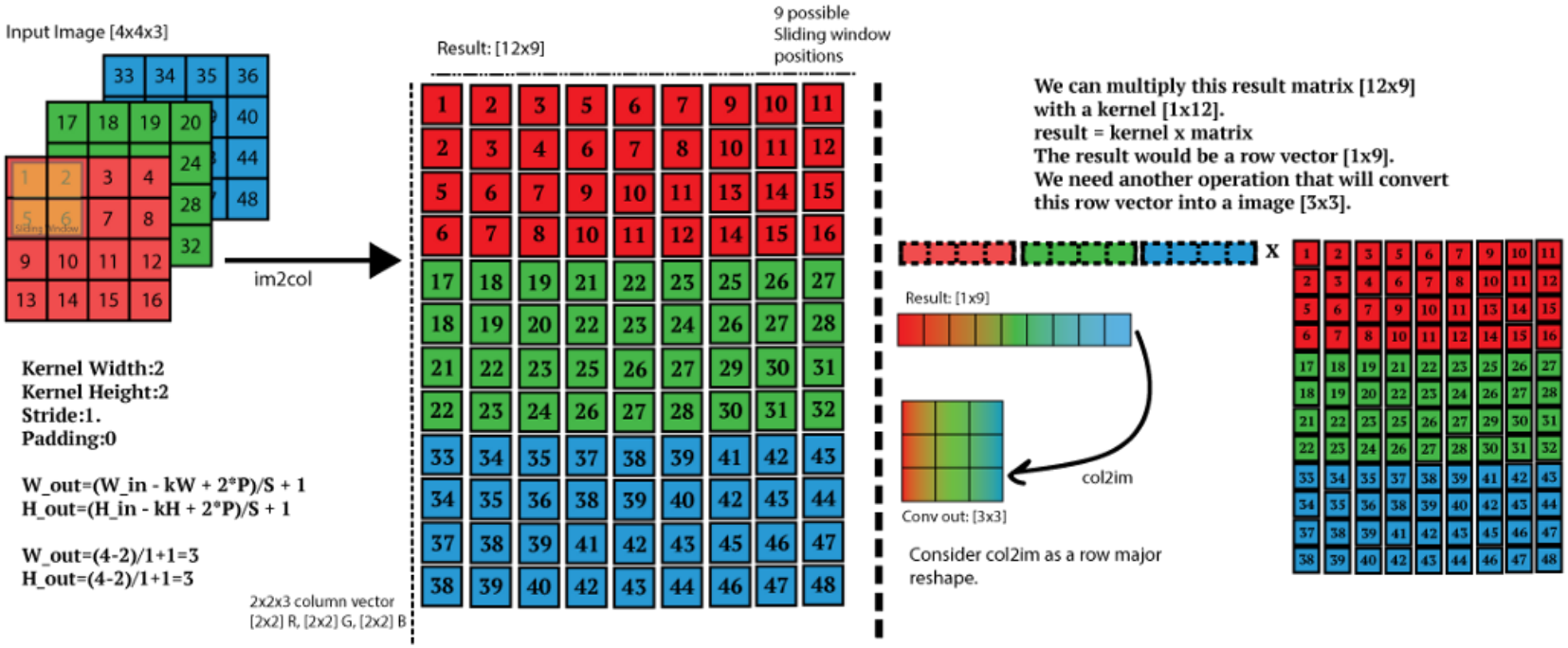
[1] “Integer quantization for deep learning inference: Principles and empirical evaluation." (arXiv 2020)
[2] "Post-training piecewise linear quantization for deep neural networks." (ECCV 2020)
[3] "Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation." (CVPR 2022)

- Backgrounds
  - Accelerating convolutional operations
  - Basic quantization techniques [1]

- Quantization research
  - Post-Training Piecewise Linear Quantization for Deep Neural Networks [2]
  - Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation [3]

Backgrounds

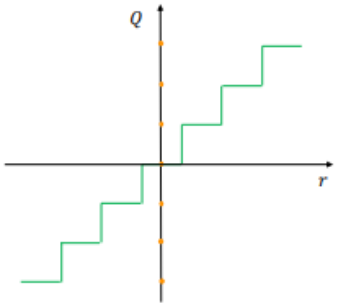
- Accelerating convolutional operations
  - Convolution 연산이 실제 device에서 동작할 땐 matrix 곱 형태로 변환됨
    - NVIDIA의 cuBlas 또는 Intel의 mkl에서 사용하는 방식
  - Matrix 곱의 경우 Winograd 등의 알고리즘을 통해 고속 연산 수행



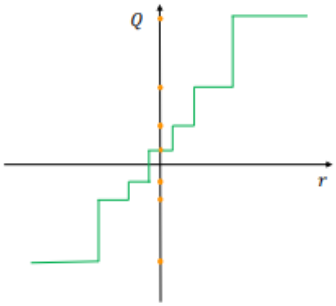
Backgrounds

[1] "Integer quantization for deep learning inference: Principles and empirical evaluation." (arXiv 2020)

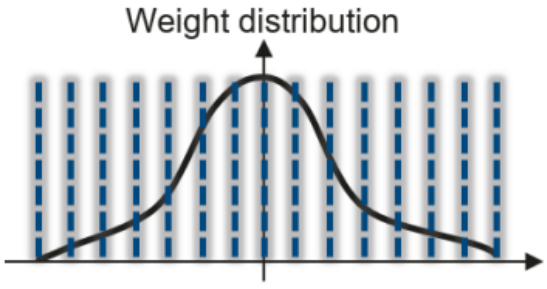
- Basic quantization techniques [1]
  - Neural network의 weight 및 activation을 lower bit-width의 integer 자료형으로 변환
    - Floating point의 matrix 곱 연산에 비해 빠른 연산속도와 낮은 bandwidth를 가짐
    - 하지만 낮은 precision에 의한 information loss가 생기며 이를 quantization error라고 함
    - Information loss를 최소한으로 하는 다양한 quantization 방법들이 연구되고 있음
  - Uniform vs non-uniform quantization
    - Uniform quantization의 경우 round 및 linear operation을 활용하여 quantization 수행
      - Bell 모양의 실제 weight distribution을 제대로 반영하지 못함
    - Non-uniform의 경우 codebook 등을 활용하여 다양한 distribution에 대해 효과적으로 mapping을 수행
      - 하지만 hardware 구현 시 변환 과정에서 overhead가 생겨 추가적인 알고리즘이 필요함



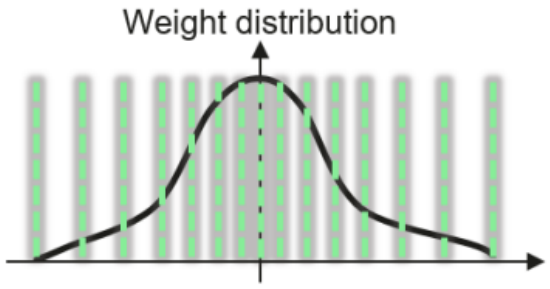
< Uniform quantization >



< Non-uniform quantization >



< Uniform quantization >



< Non-uniform quantization >

Backgrounds

[1] "Integer quantization for deep learning inference: Principles and empirical evaluation." (arXiv 2020)

- Basic quantization techniques [1]
  - Range mapping (Affine / Scale)
    - Quantization시 weight 및 activation의 dynamic range를 결정
    - Affine quantization

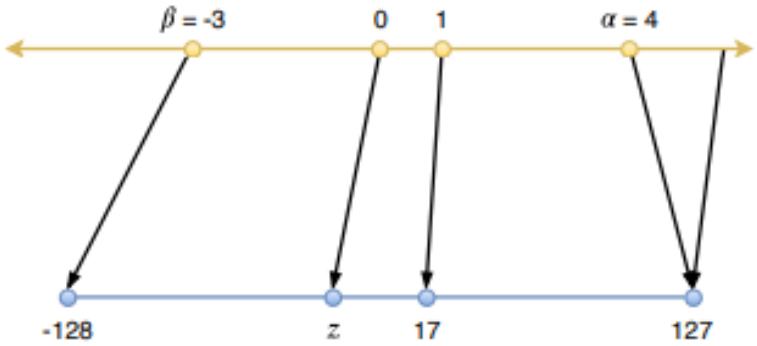
$$x_q = \text{quantize}(x, b, s, z) = \text{clip}(\text{round}(s \cdot x + z), -2^{b-1}, 2^{b-1} - 1) \qquad \hat{x} = \text{dequantize}(x_q, s, z) = \frac{1}{s}(x_q - z)$$

      ⁘ Parameter z ∈ ℝ를 통해 weight의 zero-point를 임의의 위치로 변환할 수 있음

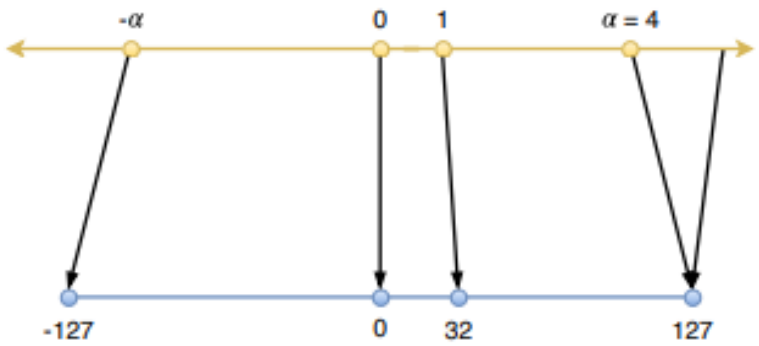
    - Scale quantization

$$x_q = \text{quantize}(x, b, s) = \text{clip}(\text{round}(s \cdot x), -2^{b-1} + 1, 2^{b-1} - 1) \qquad \hat{x} = \text{dequantize}(x_q, s) = \frac{1}{s}x_q$$

      ⁘ Real-value인 scale factor s ∈ ℝ 만을 활용하여 제한적인 quantization을 수행



< Affine quantization >



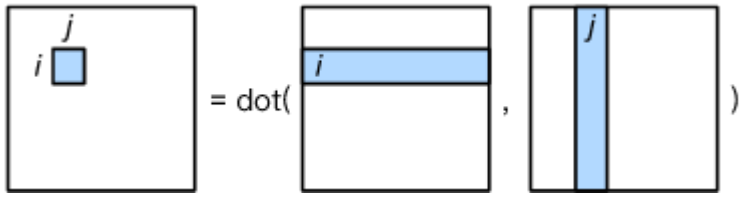
< Scale quantization >

Backgrounds

[1] "Integer quantization for deep learning inference: Principles and empirical evaluation." (arXiv 2020)

- Basic quantization techniques [1]
  - Tensor quantization granularity
    - Tensor에 대한 quantization시 parameter (s, z 등) 의 공유 범위에 따라 구분됨
    - per-element, per-channel, per-tensor 등으로 설정할 수 있음
      - 적용할 layer 또는 tensor에 따라 다른 granularity를 선택하여 사용
        - ✓적절한 granularity를 사용할 시 계산 속도를 크게 향상시킬 수 있음

$Y = XW$



$$y_{ij} = \sum_{k=1}^{p} x_{ik} \cdot w_{kj} \approx \sum_{k=1}^{p} \text{dequantize}(x_{q,ik}, s_{q,ik}) \cdot \text{dequantize}(w_{q,kj}, s_{w,kj})$$

$$= \sum_{k=1}^{p} \frac{1}{s_{x,ik}} x_{q,ik} \cdot \frac{1}{s_{w,kj}} w_{q,kj}$$

If activation (x) has per-row granularity & weight (w) has per-column granularity

$$\frac{1}{s_{x,i} \cdot s_{w,j}} \sum_{k=1}^{p} x_{q,ik} \cdot w_{q,kj}$$

Integer arithmetic

Backgrounds

[1] "Integer quantization for deep learning inference: Principles and empirical evaluation." (arXiv 2020)

- Basic quantization techniques [1]
  - Computational cost of affine quantization
    - Affine quantization을 사용할 시 computational overhead 발생
      - (1)번 식의 경우 scale quantization에서의 integer arithmetic과 동일하며 고속 연산 가능
      - (2)번 식의 경우 모두 사전에 정의된 parameter로 offline에서 미리 계산이 가능함
      - (3)번 식의 경우 quantized input에 의해 (2)번과 같은 offline 연산이 불가능함
        - ✓이로 인해 다음 layer로 넘어가기까지의 computational bottleneck 발생
    - 이러한 이유로 neural network의 weight에 대해선 scale quantization을 사용하는 것을 권장

$$y_{ij} \approx \sum_{k=1}^{p} \frac{1}{s_x}(x_{q,ik} - z_x)\frac{1}{s_{w,j}}(w_{q,kj} - z_{w,j})$$

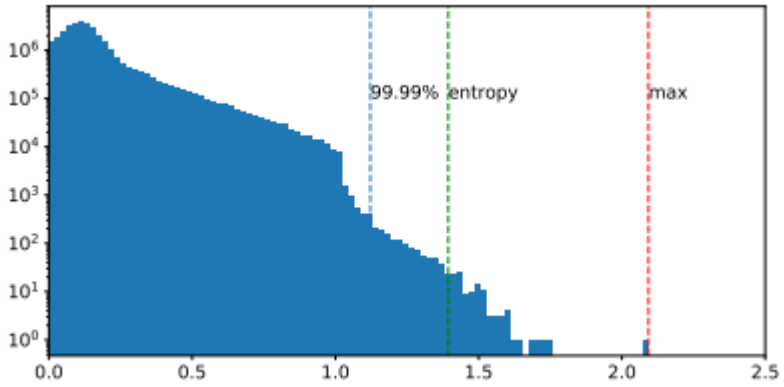
$$= \frac{1}{s_x s_{w,j}}\left(\underbrace{\sum_{k=1}^{p} x_{q,ik}w_{q,kj}}_{(1)} - \underbrace{\sum_{k=1}^{p}(w_{q,kj}z_x + z_x z_{w,j})}_{(2)} - \underbrace{\sum_{k=1}^{p} x_{q,ik}z_{w,j}}_{(3)}\right)$$

Integer arithmetic | Offline computation | Extra computation

Backgrounds

[1] "Integer quantization for deep learning inference: Principles and empirical evaluation." (arXiv 2020)

- Basic quantization techniques [1]
  - Calibrating quantization parameters (Post training quantization, PTQ)
    - 학습이 완료된 neural network에 random한 input을 넣어주어 activation의 statistics를 수집
    - Max calibration
      - Activation에서 가장 큰 값까지 전부 사용하여 quantization하며 outlier에 취약함
    - Percentile calibration
      - Tensor의 histogram에서 일정부분만 사용하는 방식이며 주로 99.99%를 사용
    - Entropy calibration
      - KL divergence를 활용하여 information loss가 제일 적은 지점 까지만 사용



< Calibration 방법 별 clipping 지점 >

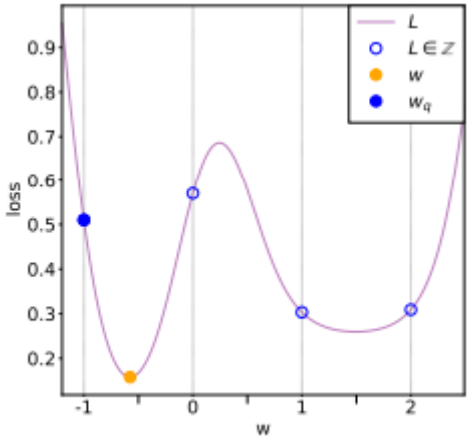
Models	fp32	Max	Entropy	99.9%	99.99%	99.999%
MobileNet v1	71.88	69.51	70.19	**70.39**	70.29	69.97
MobileNet v2	71.88	69.41	70.28	70.68	**71.14**	70.72
ResNet50 v1.5	76.16	75.82	**76.05**	75.68	75.98	75.97
ResNet152 v1.5	78.32	77.93	**78.21**	77.62	78.17	78.17
Inception v3	77.34	72.53	**77.54**	76.21	77.52	77.43
Inception v4	79.71	0.12	79.60	78.16	**79.63**	79.12
ResNeXt50	77.61	77.31	**77.46**	77.04	77.39	77.45
ResNeXt101	79.30	78.74	79.09	78.77	79.15	**79.17**
EfficientNet b0	76.85	22.3	**72.06**	70.87	68.33	51.88
EfficientNet b3	81.61	54.27	76.96	77.80	**80.28**	80.06

< Calibration 방법 별 모델의 성능 >

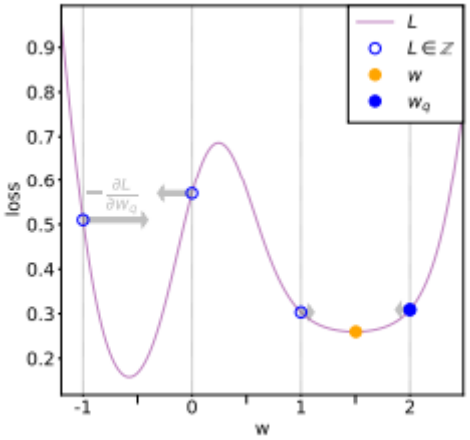
Backgrounds

[1] "Integer quantization for deep learning inference: Principles and empirical evaluation." (arXiv 2020)

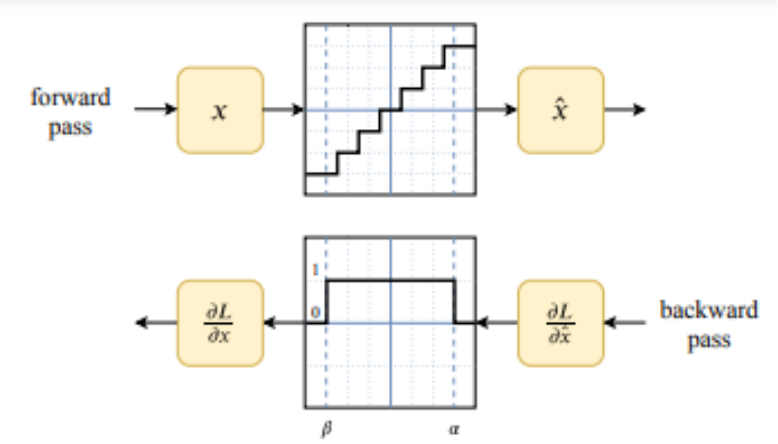
- Basic quantization techniques [1]
  - Quantization aware training (QAT)
    - 학습과정에서 pseudo quantization을 적용해주어 quantization이 되었을 때의 최적의 parameter를 학습
      - 아래 그림에서 주황색 점은 학습이 완료된 FP32에서의 weight
      - Post training quantization만을 적용하였을 때 narrow convex에서 성능이 급격히 감소함
      - 반면 QAT를 사용하였을 때 FP32에서의 성능이 감소하지만 quantization 시 성능이 향상됨
    - Straight-Through Estimator (STE)
      - Neural network 내부에 추가된 quantization operation의 경우 derivative가 불명확함
      - Backward pass에선 quantization operation이 STE operation으로 대체됨



< Post training quantization >



< After quantization aware fine-tuning >

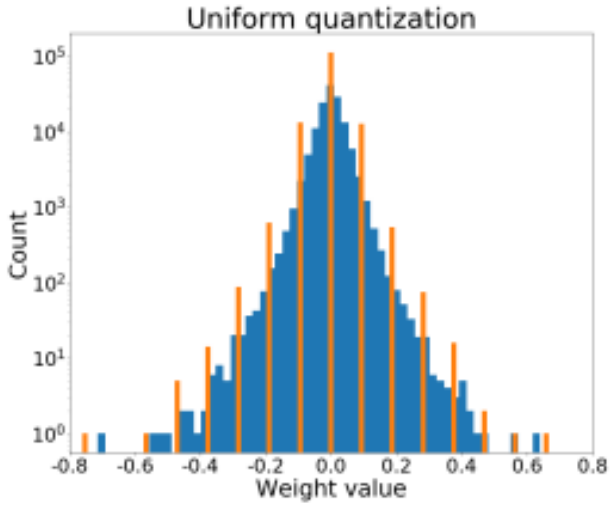


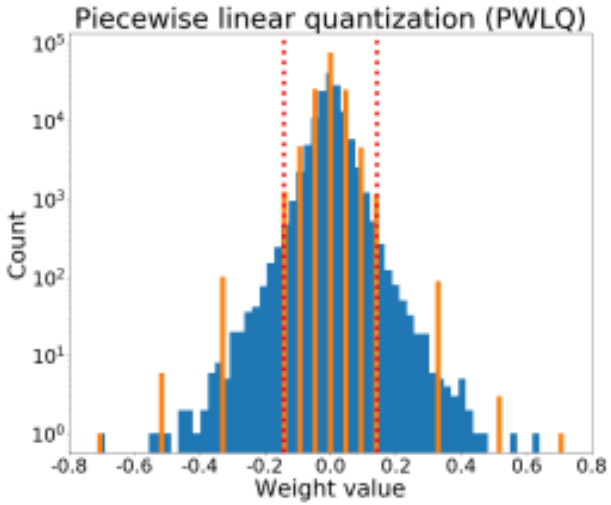
< Straight-Through Estimator >

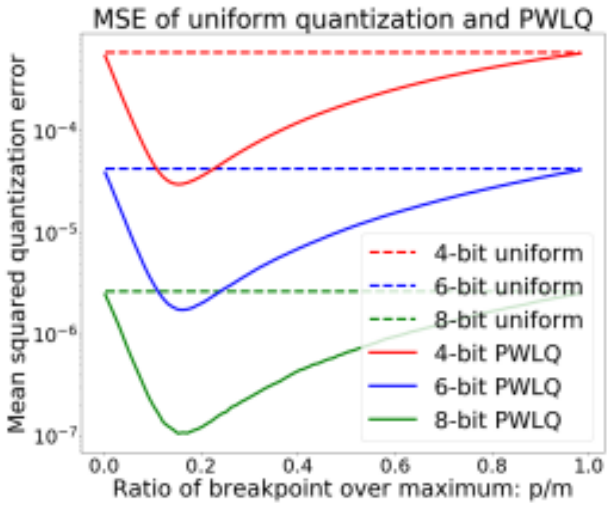
Quantization research

[1] "Post-training piecewise linear quantization for deep neural networks." (ECCV 2020)

- Post-Training Piecewise Linear Quantization for Deep Neural Networks [1]
  - Proposed a **piecewise linear quantization (PWLQ)** scheme for efficient deployment of pre-trained DNNs **without retraining** or access to the full training dataset
  - Presented **a solution to find the optimal breakpoints** and demonstrate that the proposed method achieves a **lower quantization error** than the uniform scheme
  - Provide a comprehensive **evaluation on image classification, semantic segmentation, and object detection benchmarks** and show that the proposed method achieves state-of-the-art results



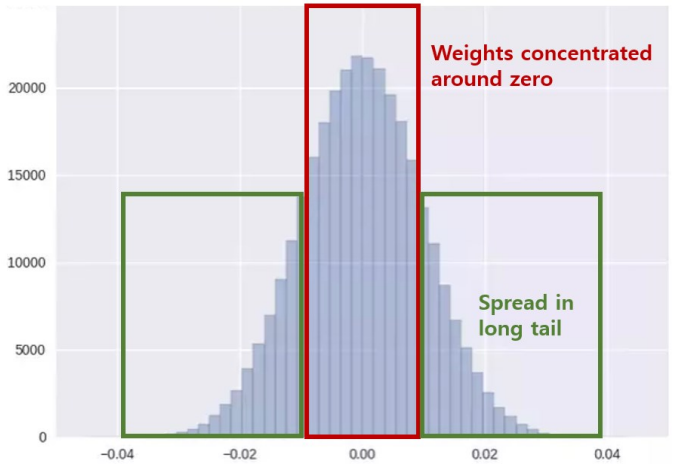




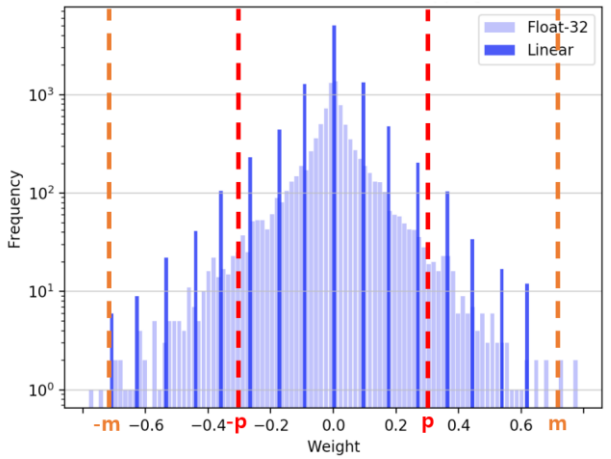
Quantization research

[1] "Post-training piecewise linear quantization for deep neural networks." (ECCV 2020)

- Post-Training Piecewise Linear Quantization for Deep Neural Networks [1]
  - Motivation
    - Hardware 구현에 적합한 uniform quantization method는 lower bit에서 성능이 크게 떨어짐
      - 별도의 transformation이나 look-up table이 필요한 non-linear quantization method는 hardware 구현에 어려움이 있음
    - 이는 uniform quantization이 bell-shaped weight의 분포를 포착하지 못하기 때문
      - 일반적인 neural network의 weight들은 Gaussian이나 Laplacian과 같은 bell-shaped 분포를 가짐
    - Quantization error를 줄이면서 hardware friendly한 linear quantization의 형태를 유지하는 PWLQ 제안
      - Quantization error를 최대한 줄이면 자연스럽게 성능 또한 올라갈 것이기 때문



< Bell-shaped weight distribution >



< p값에 따라 분리된 region >

Quantization research

[1] "Post-training piecewise linear quantization for deep neural networks." (ECCV 2020)

- Post-Training Piecewise Linear Quantization for Deep Neural Networks [1]
  - Deciding Optimal Breakpoint
    - Quantization error (r = full-precision real number)

$$\varepsilon_{pw} = \mathrm{pw}(r; b, m, p) - r$$

    - Expected Quantization Error for Uniform Quantization

$$\mathbb{E}(\varepsilon_{uni}^2; b, r_l, r_u) = \frac{s^2}{12} = C(b)\Delta^2$$

    - Expected Quantization Error for PWLQ

$$\begin{aligned}\mathbb{E}(\varepsilon_{pw}^2; b, m, p) &= C(b-1)\Big\{(m-p)^2\left[F(-p)+1-F(p)\right]+p^2\left[F(p)-F(-p)\right] \\ &= C(b-1)\Big\{(m-p)^2+m(2p-m)\left[2F(p)-1\right]\Big\}\end{aligned}$$

    - Optimal breakpoint p^*

$$p^* = \arg\min_{p\in(0,\frac{m}{2})} \mathbb{E}(\varepsilon_{pw}^2; b, m, p)$$

      ⁂ Optimal point p는 $[0, \frac{p}{2}]$ 사이에 위치하여 해당 optimization 문제는 convex임

Quantization research

[1] "Post-training piecewise linear quantization for deep neural networks." (ECCV 2020)

- Post-Training Piecewise Linear Quantization for Deep Neural Networks [1]
  - Hardware impact of PWLQ
    - Convolution output for uniform quantization

$$\langle \hat{X},\ \hat{W} \rangle = \langle s_x X_q + z_x I,\ s_w W_q \rangle = C_0 \langle X_q,\ W_q \rangle + C_1$$

    - Convolution output for PWLQ

$$P_1 = \langle s_x X_q + z_x I,\ s_{w_1} W_{q_1} \rangle_{R_1} = C_2 \langle X_q,\ W_{q_1} \rangle_{R_1} + C_3$$

$$P_2 = \langle s_x X_q + z_x I,\ s_{w_2} W_{q_2} + pI \rangle_{R_2} = C_4 \langle X_q,\ W_{q_2} \rangle_{R_2} + C_5 \langle X_q,\ I \rangle_{R_2} + C_6$$

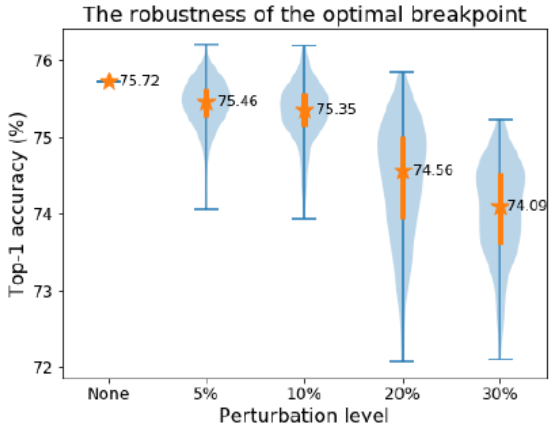
      - P_1, P_2 : separate computational paths

    - PWLQ로 인한 연산 overhead
      - Three accumulators: P_1, P_2 product sum and one more for P_2 activation layer
        - ✓Breakpoint가 증가할수록 activation layer를 위한 accumulator 증가
      - One extra region indicator bit
        - ✓Weight의 영역을 표시하기 위해 필요
        - ✓Computation에는 영향을 주지 않음

Quantization research

[1] "Post-training piecewise linear quantization for deep neural networks." (ECCV 2020)

- Post-Training Piecewise Linear Quantization for Deep Neural Networks [1]
  - Robustness of optimal breakpoint
    - Optimal breakpoint에 perturbation을 더하여 robustness 확인
      - P% 이내에서 전체 layer에 random한 perturbation 적용
    - 10%의 perturbation에서도 성능이 크게 하락하지 않음
      - 제안한 방식으로 구한 optimal point의 유효성 증명



  - Multiple Breakpoints
    - Breakpoint 개수와 hardware overhead 사이의 trade-off
      - Breakpoint가 많아지면 정확도는 올라가지만 그만큼 하드웨어 연산 량이 늘어남
      - Hardware overhead를 고려해 breakpoint는 하나로 유지할 것을 권장

Number of Breakpoints	Hardware Accumulators	Inception-v3 (77.49) 5-bit	4-bit	3-bit	ResNet-50 (76.13) 5-bit	4-bit	3-bit	MobileNet-v2 (71.88) 5-bit	4-bit	3-bit
One	**Three**	77.28	75.72	61.76	75.62	74.28	67.30	69.05	54.34	16.77
Two	Five	77.31	76.73	71.40	75.94	75.24	73.27	70.01	65.74	36.44
Three	Seven	**77.46**	**77.00**	**74.07**	**76.06**	**75.77**	**73.84**	**70.43**	**67.71**	**55.17**

Quantization research

[1] "Post-training piecewise linear quantization for deep neural networks." (ECCV 2020)

- Post-Training Piecewise Linear Quantization for Deep Neural Networks [1]
  - Experiment on ImageNet classification task
    - Inception-v3, ResNet50, MobileNetV2 에 대해 8bit, 4bit quantization에서 가장 높은 성능을 기록

Network	W/A	**PWLQ (Ours)**	QWP [28]	ACIQ [2]	LBQ [7]	SSBD [39]	QRD [31]	UNIQ [3]	DFQ [42]
Inception-v3 (Top1%)	32/32	77.49	78.00	77.20	76.23	77.90	77.97	-	-
	8/8	**+0.04 (77.53)**	0.00 (78.00)	-	-	-0.03 (77.87)	-0.09 (77.88)	-	-
	4/8	**-1.04 (76.45)**	-7.00 (71.00)	-9.00 (68.20)	-1.44 (74.79)	-	-	-	-
	4/4	**-2.58 (74.91)**	-	-10.80 (66.40)	-4.62 (71.61)	-	-	-	-
ResNet-50 (Top1%)	32/32	76.13	75.20	76.10	76.01	75.20	-	76.02	-
	8/8	**-0.03 (76.10)**	-0.10 (75.10)	-	-	-0.25 (74.95)	-	-	-
	4/8	**-0.51 (75.62)**	-21.20 (54.00)	-0.80 (75.30)	-1.03 (74.98)	-	-	-2.56 (73.37)	-
	4/4	**-1.28 (74.85)**	-	-2.30 (73.80)	-3.41 (72.60)	-	-	-	-
MobileNet-v2 (Top1%)	32/32	71.88	71.90	-	-	71.80	71.23	-	71.72
	8/8	**-0.15 (71.73)**	-2.10 (69.80)	-	-	-0.61 (71.19)	-1.68 (69.55)	-	-0.53 (71.19)
	4/8	**-2.68 (69.22)**	-71.80 (0.10)	-	-	-	-	-	-

Quantization research

[1] "Post-training piecewise linear quantization for deep neural networks." (ECCV 2020)

- Post-Training Piecewise Linear Quantization for Deep Neural Networks [1]
  - Experiment on Semantic Segmentation task
    - DeepLabV3+의 MobileNetV2 backbone에 대한 quantization 진행

Network	W/A	32/32	8/8	6/8	4/8
DeepLab-v3+ (mIoU%)	Uniform	70.81	-0.65 (70.16)	-1.54 (69.27)	-20.76 (50.05)
	PWLQ (Ours)	70.81	**-0.12 (70.69)**	**-0.42 (70.39)**	**-3.15 (67.66)**
	DFQ [42]	72.94	-0.61 (72.33)	-	-

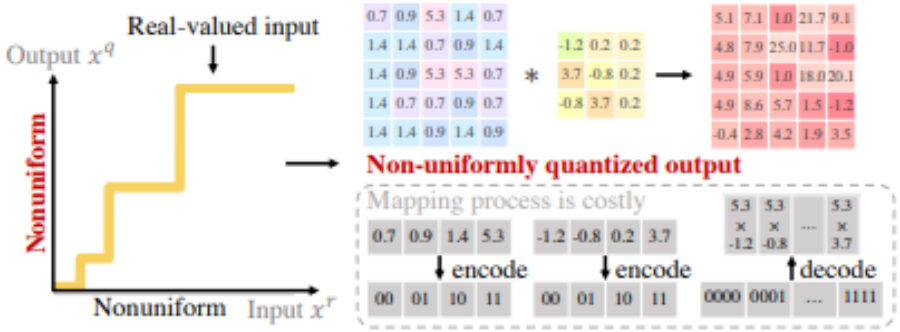
  - Experiment on Object Detection task
    - SSD-Lite의 MobileNetV2 backbone에 대한 quantization 진행

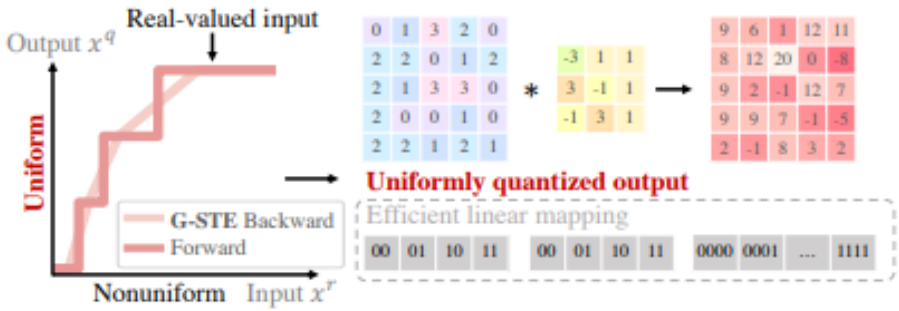
Network	W/A	32/32	8/8	6/8	4/8
SSD-Lite (mAP%)	Uniform	68.70	-0.20 (68.50)	-0.43 (68.37)	-3.91 (64.79)
	PWLQ (Ours)	68.70	**-0.19 (68.51)**	**-0.28 (68.42)**	**-0.38 (68.32)**
	DFQ [42]	68.47	-0.56 (67.91)	-	-

Quantization research

[1] "Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation." (CVPR 2022)

- Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation [1]
  - Proposed **Nonuniform-to-Uniform Quantizer (N2UQ)** for improving the quantization precision via learning input thresholds, while **maintaining hardware-friendliness** in implementation
  - Proposed **Generalized Straight-Through Estimator (GSTE)** to tackle **intractable gradient** computation w.r.t. **input threshold parameters** in N2UQ. G-STE calculates the expectation of the stochastic quantization as the backward approximation to the forward deterministic quantization
  - Proposed **a novel weight regularization** considering the overall weight distribution for further preserving information in weight quantization

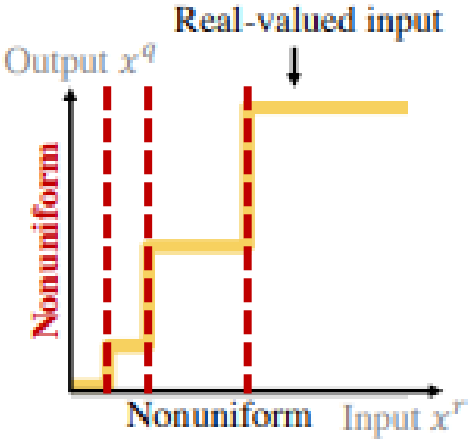


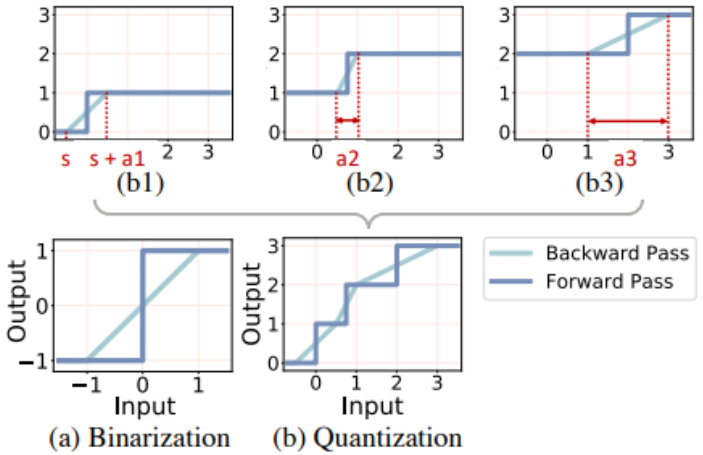


Quantization research

[1] "Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation." (CVPR 2022)

- Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation [1]
  - Motivation
    - Non-uniform quantization은 작은 quantization error을 가지나 hardware implementation이 어려움
      - Uniform quantization에 비해 weight distribution에 대한 좋은 표현력을 가짐
    - Hardware friendly하면서 non-uniform quantization과 비슷한 표현력을 가지는 uniform quantization 구현
      - 다만 기존 uniform quantization과 다른 점은 real-valued input에 대해 동일한 step을 갖지 않음
        - ✓전체 range에 대해 $2^n - 1$개의 learnable threshold를 주입하여 학습 과정에서 이를 학습
      - 이 threshold는 intractable한 변수이며 quantization function은 대부분의 값에서 derivative가 0
    - 이를 해결하기 위해 새로운 G-STE를 제안하였으며 QAT 방식으로 학습





Quantization research

[1] "Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation." (CVPR 2020)

- Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation [1]
  - Forward pass: threshold learning quantization
    - 기존 linear quantization과 동일하게 scale quantization에 기반하여 연산 수행
      - Activation에는 N2UQ를 적용하며 weight에는 일반적인 scale quantization 적용

$$x^q = \begin{cases} 0 & x^r < T_1 \\ 1 & T_1 \leqslant x^r < T_2 \\ \dots & \dots \\ 2^n - 1 & x^r \geqslant T_{2^n-1} \end{cases}$$

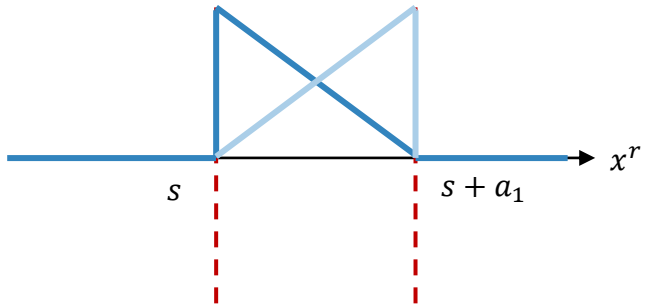
$$a^q \cdot w^q = \sum_{i=0}^{M-1} \sum_{j=0}^{K-1} 2^{i+j} \text{popcnt}[\text{and}(\mathbf{a}_i, \mathbf{w}_j)],$$

$$\mathbf{a}_i, \mathbf{w}_j \in \{0,1\}\ \forall i \in \{0,1,\dots,M-1\}, j \in \{0,1,\dots,K-1\}.$$

  - Backward pass: generalized straight-through estimator (G-STE)
    - **Stochastic binarization function**의 expectation을 활용하여 threshold의 gradient를 추적
      - Real-valued activation x^r 의 위치에 따른 확률에 대한 변수
        - ✓Forward pass에서는 hard thresholding을 통해 사용

$$\widetilde{x}^{q\{0,1\}} = \begin{cases} 0 & \text{with probability } p = \text{clip}\left(\frac{s+a_1-x^r}{a_1}, 0, 1\right) \\ 1 & \text{with probability } p = \text{clip}\left(\frac{x^r-s}{a_1}, 0, 1\right) \end{cases}$$

$$x^{q\{0,1\}} = \begin{cases} 0 & p_{\{\widetilde{x}^q=0\}} > 0.5 \\ 1 & p_{\{\widetilde{x}^q=0\}} \leqslant 0.5 \end{cases} = \begin{cases} 0 & x^r < s + \frac{a_1}{2} \\ 1 & x^r \geqslant s + \frac{a_1}{2} \end{cases}$$



Quantization research

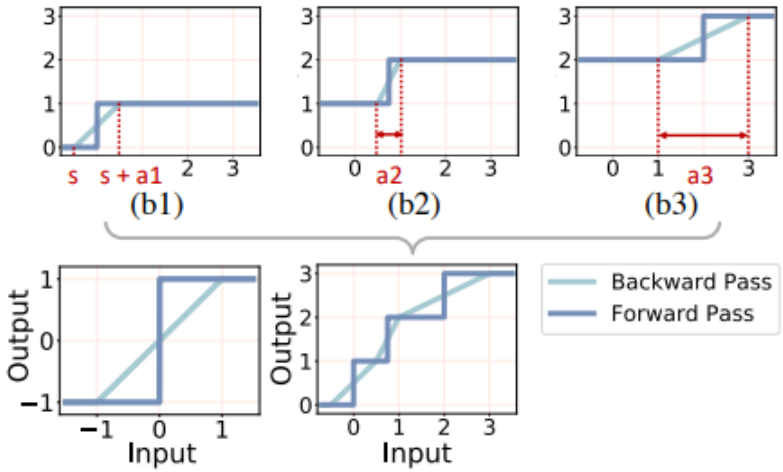
[1] "Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation." (CVPR 2020)

- Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation [1]
  - Backward pass: generalized straight-through estimator (G-STE)
    - Stochastic binarization function의 derivation은 다음과 같음

$$\mathbb{E}[\frac{\partial \widetilde{x}^{q_{\{0,1\}}}}{\partial x^r}] = \frac{\partial}{\partial x^r}\mathbb{E}[\widetilde{x}^{q_{\{0,1\}}}] = \frac{\partial}{\partial x^r}(0 \times p_{\{\widetilde{x}^q=0\}} + 1 \times p_{\{\widetilde{x}^q=1\}}) = \frac{\partial \text{clip}(\frac{x^r-s}{a_1}, 0, 1)}{\partial x^r}$$

    - 구간 별 segment (stochastic binarization function)을 쌓으면 다음과 같음

$$x^q = x^b_{thre_1} + x^b_{thre_2} + \ldots + x^b_{thre_n}$$



$$x^q = \begin{cases} 0 & x^r < d_0 + \frac{a_1}{2} \\ i & d_{i-1} + \frac{a_i}{2} \leqslant x^r < d_i + \frac{a_{i+1}}{2} \\ & i \in \{1, \ldots, 2^n - 2\} \\ 2^n - 1 & x^r \geqslant d_{2^n-2} + \frac{a_{2^n-1}}{2} \end{cases}$$

$$\frac{\partial x^q}{\partial x^r} = \mathbb{E}[\frac{\partial \widetilde{x}^q}{\partial x^r}] = \frac{\partial}{\partial x^r}\mathbb{E}[\widetilde{x}^q]$$
$$= \begin{cases} \frac{\partial}{\partial x^r}(\frac{x^r - d_{i-1}}{a_i} + i - 1) & d_{i-1} \leqslant x^r < d_i \\ & i \in \{1, \ldots, 2^n - 1\} \\ 0 & otherwise \end{cases}$$

< Generalized Straight-Trough Estimator >

Quantization research

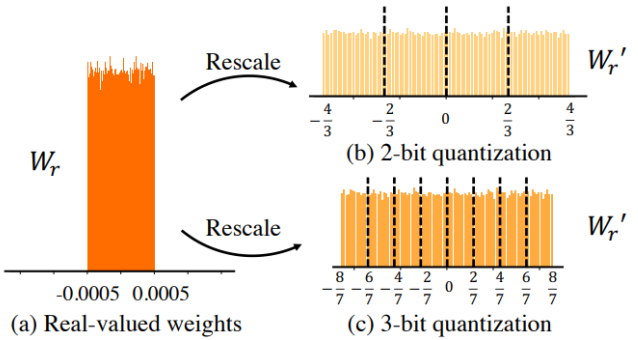
[1] "Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation." (CVPR 2020)

- Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation [1]
  - Backward pass: generalized straight-through estimator (G-STE)
    - Quantization에서 forward 및 backward에서의 연산이 for문을 통해 모든 segment에 대해 이루어짐
      - torch.where 연산을 통해 tensor의 임의의 element x^r 이 위치한 구간을 찾음
      - 이로 인해 일반적인 linear quantization과 동일한 속도를 지니진 않지만 추가적인 matrix multiplication 연산 및 memory access가 필요하지 않음

        ✓Hardware friendly!
  - Entropy preserving weight regularization
    - Quantization 시 weight가 일부 값으로 치우치지 않게 하기 위해 L1 normalization을 적용해줌
      - 기존 scale quantization에선 종종 long-tail의 weight들이 나타나지 않는 경우가 생김
      - 아래와 같은 식을 통해 weight가 [-1, 1] 범위 내로 rescale되며 training 이후 BN에 흡수됨

$$W^{r\prime} = \frac{2^{(n-1)}}{2^n - 1} \boxed{\frac{|W^r|}{||W^r||_{l1}}} W^r$$

1/Absolute mean value



Quantization research

[1] "Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation." (CVPR 2020)

- Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation [1]
  - Comparison with State-of-the-Art Methods
    - ImageNet dataset에 대한 성능 비교
    - Uniform methods보다 높은 성능 기록
      - 파란색 박스
    - Full precision의 network보다도 높은 성능 기록
      - 기존의 network에 redundancy한 부분이 있었으며 제안한 방법이 이러한 점을 잘 걸러내어 학습

Network	Method	Bit-width (W/A)	Accuracy(%) Top-1	Accuracy(%) Top-5	Bit-width (W/A)	Accuracy(%) Top-1	Accuracy(%) Top-5	Bit-width (W/A)	Accuracy(%) Top-1	Accuracy(%) Top-5
ResNet-18 (FP: 71.8)	PACT [8]	2/2	64.4	85.6	3/3	68.1	88.2	4/4	69.2	89.0
	DoReFa-Net [50]	2/2	64.7	84.4	3/3	67.5	87.6	4/4	68.1	88.1
	LQ-Nets [47]	2/2	64.9	85.9	3/3	68.2	87.9	4/4	69.3	88.8
	DSQ [14]	2/2	65.2	–	3/3	68.7	–	4/4	69.6	88.9
	FAQ [33]	–	–	–	–	–	–	4/4	69.8	89.1
	QIL [23]	2/2	65.7	–	3/3	69.2	–	4/4	70.1	–
	DAQ [25]	2/2	66.9	–	3/3	69.6	–	4/4	70.5	–
	DNAS [44]	–	–	–	–	–	–	~4/~4	70.6	–
	APoT [26]	2/2	67.3	87.5	3/3	69.9	89.2	4/4	70.7	89.6
	LSQ [12]	2/2	67.6	87.6	3/3	70.2	89.4	4/4	71.1	90.0
	LCQ [46]	2/2	68.9	–	3/3	70.6	–	4/4	71.5	–
	N2UQ (Ours)	2/2	**69.4**	**88.4**	3/3	**71.9**	**90.5**	4/4	**72.9**	**90.9**
ResNet-34 (FP: 74.9)	LQ-Nets [47]	2/2	69.8	89.1	3/3	71.9	90.2	–	–	–
	DSQ [14]	2/2	70.0	–	3/3	72.5	–	4/4	72.8	–
	FAQ [33]	–	–	–	–	–	–	4/4	73.3	91.3
	QIL [23]	2/2	70.6	–	3/3	73.1	–	4/4	73.7	–
	APoT [26]	2/2	70.9	89.7	3/3	73.4	91.1	4/4	73.8	91.6
	DAQ [25]	2/2	71.0	–	3/3	73.1	–	4/4	73.7	–
	DNAS [44]	–	–	–	–	–	–	~4/~4	74.0	–
	LSQ [12]	2/2	71.6	90.3	3/3	73.4	91.4	4/4	74.1	91.7
	LCQ [46]	2/2	72.7	–	3/3	74.0	–	4/4	74.3	–
	N2UQ (Ours)	2/2	**73.3**	**91.2**	3/3	**75.2**	**92.3**	4/4	**76.0**	**92.8**
ResNet-50 (FP: 77.0)	DoReFa-Net [50]	2/2	67.1	87.3	3/3	69.9	89.2	4/4	71.4	89.8
	LQ-Nets [47]	2/2	71.5	90.3	3/3	74.2	91.6	4/4	75.1	92.4
	FAQ [33]	–	–	–	–	–	–	4/4	76.3	93.0
	PACT [8]	2/2	72.2	90.5	3/3	75.3	92.6	4/4	76.5	93.2
	APoT [26]	2/2	73.4	91.4	3/3	75.8	92.7	4/4	76.6	93.1
	LSQ [12]	2/2	73.7	91.5	3/3	75.8	92.7	4/4	76.7	93.2
	Auxi [52]	2/2	73.8	91.4	3/3	75.4	92.4	–	–	–
	LCQ [46]	2/2	75.1	–	3/3	76.3	–	4/4	76.6	–
	N2UQ (Ours)	2/2	**75.8**	**92.3**	3/3	**77.5**	**93.6**	4/4	**78.0**	**93.9**

Quantization research

[1] "Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation." (CVPR 2020)
[2] "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients" (arXiv 2016)

- Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation [1]
  - Ablation study
    - Baseline [2]와 비교하여 제안한 방법의 효율성을 증명 (Table 3)
    - TLAQ는 적용한 상태로 다른 regularization 방법들과 성능을 비교하여 제안한 방법의 효율성을 증명 (Table 4)

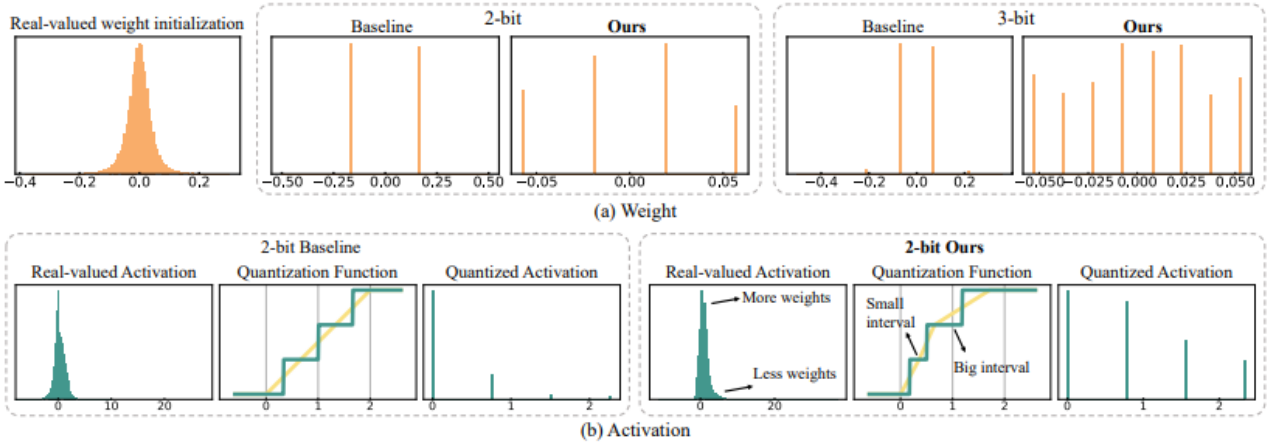
Table 3. Effects of different components of N2UQ on the final performance of a 2-bit quantized ResNet-18 network.

Method	Top-1 Acc
Baseline [50] (our implementation)	65.9
+ Threshold Learning Activation Quantizer with G-STE	68.9
+ Entropy Preserving Weight Regularization	67.8
+ Both (N2UQ)	**69.7**
Corresponding real-valued network	71.8

Table 4. Comparison among different weight regularization schemes for a 2-bit quantized ResNet-18 on ImageNet, based on the proposed threshold learning quantizer with G-STE.

Method	Top-1 Acc
No Regularization	68.9
Weight Norm [38]	67.4
Learnable Scaling Factor	68.6
Entropy Preserving Weight Regularization	**69.7**

  - Visualization



Q&A