Post-Training Piecewise Linear Quantization for Deep Neural Networks

Jun Fang[1], Ali Shafiee[1], Hamzah Abdel-Aziz[1], David Thorsley[1], Georgios Georgiadis[2⋆], and Joseph H. Hassoun[1]

[1] Samsung Semiconductor, Inc.
{jun.fang, ali.shafiee, hamzah.a, d.thorsley, j.hassoun}@samsung.com
[2] Microsoft georgios.georgiadis@microsoft.com

Abstract. Quantization plays an important role in the energy-efficient deployment of deep neural networks on resource-limited devices. Post-training quantization is highly desirable since it does not require retraining or access to the full training dataset. The well-established uniform scheme for post-training quantization achieves satisfactory results by converting neural networks from full-precision to 8-bit fixed-point integers. However, it suffers from significant performance degradation when quantizing to lower bit-widths. In this paper, we propose a **p**iece**w**ise **l**inear **q**uantization (PWLQ) scheme[3] to enable accurate approximation for tensor values that have bell-shaped distributions with long tails. Our approach breaks the entire quantization range into non-overlapping regions for each tensor, with each region being assigned an equal number of quantization levels. Optimal breakpoints that divide the entire range are found by minimizing the quantization error. Compared to state-of-the-art post-training quantization methods, experimental results show that our proposed method achieves superior performance on image classification, semantic segmentation, and object detection with minor overhead.

Keywords: deep neural networks, post-training quantization, piecewise linear quantization

1 Introduction

In recent years, deep neural networks (DNNs) have achieved state-of-the-art results in a variety of learning tasks including image classification [55, 23, 24, 54, 19, 53, 29], segmentation [5, 18, 49] and detection [36, 47, 48]. Scaling up DNNs by one or all of the dimensions [55] of network depth [19], width [59] or image resolution [30] attains better accuracy, at a cost of higher computational complexity and increased memory requirements, which makes the deployment of these networks on embedded devices with limited resources impractical.

One feasible way to deploy DNNs on embedded systems is quantization of full-precision (32-bit floating-point, FP32) weights and activations to lower precision (such as 8-bit fixed-point, INT8) integers [25]. By decreasing the bit-width,

⋆ Work performed while at Samsung Semiconductor, Inc.
[3] Code will be made available at https://github.com/jun-fang/PWLQ

the number of discrete values is reduced, while the quantization error, which generally correlates with model performance degradation increases. To minimize the quantization error and maintain the performance of a full-precision model, many recent studies [63, 4, 40, 25, 6, 60, 12, 27] rely on training either from scratch ("quantization-aware" training) or by fine-tuning a pre-trained FP32 model.

However, post-training quantization is highly desirable since it does not require retraining or access to the full training dataset. It saves time-consuming fine-tuning effort, protects data privacy, and allows for easy and fast deployment of DNN applications. Among various post-training quantization schemes proposed in the literature [28, 7, 62], uniform quantization is the most popular approach to quantize weights and activations since it discretizes the domain of values to evenly-spaced low-precision integers which can be efficiently implemented on commodity hardware's integer-arithmetic units.

Recent work [28, 31, 42] shows that post-training quantization based on a uniform scheme with INT8 is sufficient to preserve near original FP32 pre-trained model performance for a wide variety of DNNs. However, ubiquitous usage of DNNs in resource-constrained settings requires even lower bit-width to achieve higher energy efficiency and smaller models. In lower bit-width scenarios, such as 4-bit, post-training uniform quantization causes significant accuracy drop [28, 62]. This is mainly because the distributions of weights and activations of pre-trained DNNs are bell-shaped such as Gaussian or Laplacian [17, 35]. That is, most of the weights are clustered around zero while few of them are spread in a long tail. As a result, when operating at low bit-widths, uniform quantization assigns too few quantization levels to small magnitudes and too many to large ones, which leads to significant accuracy degradation [28, 62].

To mitigate this issue, various quantization schemes [41, 4, 3, 43, 26, 34] are designed to take advantage of the fact that weights and activations of pre-trained DNNs typically have bell-shaped distributions with long tails. Here, we present a new number representation via a piecewise linear approximation to be suited for these phenomena. It breaks the entire quantization range into *non-overlapping regions* where each region is assigned an equal number of quantization levels. Although our method works with an arbitrary number of regions, we suggest limiting them to two to simplify the complexity of the proposed approach and the hardware overhead. The *optimal breakpoints* that divide the entire range can be found by minimizing the quantization error. Compared to uniform quantization, our piecewise linear quantization (PWLQ) provides a richer representation that reduces the quantization error. This indicates its potential to reduce the gap between floating-point and low-bit precision models. It is also more hardware-friendly when compared to other non-linear approaches such as logarithm-based and clustering-based approaches [41, 56, 3], since in our method, the computation can still be carried out without the need of any transforms or look-up tables.

The main contributions of our work are as follows:

- We propose a piecewise linear quantization (PWLQ) scheme for efficient deployment of pre-trained DNNs without retraining or access to the full training dataset. We also investigate its impact on hardware implementation.

- We present a solution to find the optimal breakpoints and demonstrate that our method achieves a lower quantization error than the uniform scheme.
- We provide a comprehensive evaluation on image classification, semantic segmentation, and object detection benchmarks and show that our proposed method achieves state-of-the-art results.

2 Related Work

There is a wide variety of approaches in the literature that facilitate the efficient deployment of DNNs. The first group of techniques relies on designing network architectures that depend on more efficient building blocks. Notable examples include depth/point-wise layers [22, 52] as well as group convolutions [61, 38]. These methods require domain knowledge, training from scratch and full access to the task datasets. The second group of approaches optimizes network architectures in a typical task-agnostic fashion and may or may not require (re)training. Weight pruning [17, 32, 20, 37], activation compression [10, 9, 14], knowledge distillation [21, 45] and quantization [8, 46, 66, 64, 41, 25] fall under this category.

In particular, quantization of activations and weights [15, 16, 57, 35, 6, 60, 62] leads to model compression and acceleration as well as to overall savings in power consumption. Model parameters can be stored in a fewer number of bits while the computation can be executed on integer-arithmetic units rather than on power-hungry floating-point ones [25]. There has been extensive research on quantization with and without (re)training. In the rest of this section, we focus on post-training quantization that directly converts full-precision pre-trained models to their low-precision counterparts.

Recent works [28, 31, 42] have demonstrated that 8-bit quantized models have been able to accomplish negligible accuracy loss for a variety of networks. To improve accuracy, per-channel (or channel-wise) quantization is introduced in [28, 31] to address variations of the range of weight values across channels. Weight equalization/factorization is applied by [39, 42] to rescale the difference of weight ranges between different layers. In addition, bias shifts in the mean and variance of quantized values are observed and counteracting methods are suggested by [2, 13]. A comprehensive evaluation of clipping techniques is presented by [62] along with an outlier channel splitting method to improve quantization performance. Moreover, adaptive processes of assigning different bit-width for each layer are proposed in [35, 65] to optimize the overall bit allocation.

There are also a few attempts to tackle 4-bit post-training quantization by combining multiple techniques. In [2], a combination of analytical clipping, bit allocation, and bias correction is used, while [7] minimizes the mean squared quantization error by representing one tensor with one or multiple 4-bit tensors as well as by optimizing the scaling factors.

Most of the aforementioned works utilize a linear or uniform quantization scheme. However, linear quantization cannot capture the bell-shaped distribution of weights and activations, which results in sub-optimal solutions. To overcome this deficiency, [3] proposes a quantile-based method to improve accuracy but

their method works efficiently only on highly customized hardware; [26] employs two different scale factors on overlapping regions to reduce computation bits over fixed-point implementations. However, its scale factors restricted to powers of two and heuristic options limit the accuracy performance. Instead, we propose a piecewise linear approach that improves over the selection of optimal breakpoints that leads to state-of-the-art quantized model results. Our method can be implemented efficiently with minimal modification to commodity hardware.

3 Quantization Schemes

In this section, we review a uniform quantization scheme and discuss its limitations. We then present PWLQ, our piecewise linear quantization scheme and show that it has a stronger representational power (a smaller quantization error) compared to the uniform scheme.

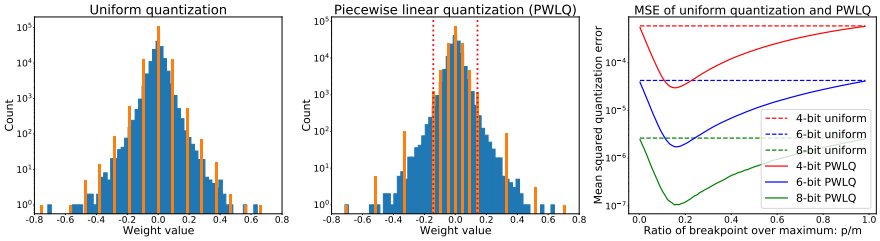

Fig. 1. Quantization of *conv*4 layer weights in a pre-trained Inception-v3. Left: uniform quantization. Middle: piecewise linear quantization (PWLQ) with one breakpoint, dotted line indicates the breakpoint. Right: Mean squared quantization error (MSE) for various bit-widths ($b = 4, 6, 8$). MSE of PWLQ is convex *w.r.t.* the breakpoint p, the b-bit PWLQ can achieve a smaller quantization error than the b-bit uniform scheme

3.1 Uniform Quantization

Uniform quantization (the left of Figure 1) linearly maps full-precision real numbers r into low-precision integer representations. From [25, 7], the approximated version $\hat{r}$ from uniform quantization scheme at b-bit can be defined as:

$$\begin{aligned} &\hat{r} = \text{uni}(r; b, r_l, r_u, z) = s \times r_q + z, \\ &r_q = \left\lceil \frac{\text{clamp}(r; r_l, r_u) - z}{s} \right\rfloor_{\mathbb{Z}_b}, \\ &\text{clamp}(r; r_l, r_u) = \min(\max(r, r_u), r_l), \\ &s = \frac{\Delta}{N-1}, \quad \Delta = r_u - r_l, \quad N = 2^b, \end{aligned} \tag{1}$$

where $[r_l, r_u]$ is the quantization range, s is the scaling factor, z is the offset, N is the number of quantization levels, r_q is the quantized integer computed by a rounding function $\lceil \cdot \rfloor_{\mathbb{Z}_b}$ followed by saturation to the integer domain

$\mathbb{Z}_b$. We set the offset $z = 0$ for symmetric signed distributions combined with $\mathbb{Z}_b = \{-2^{b-1}, ..., 2^{b-1} - 1\}$ and $z = r_l$ for asymmetric unsigned distributions (e.g., ReLU-based activations) with $\mathbb{Z}_b = \{0, ..., 2^b - 1\}$. Since the scheme (1) introduces a quantization error defined as $\varepsilon_{uni} = \hat{r} - r$, the expected quantization error squared is given by:

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

with $C(b) = \frac{1}{12(2^b-1)^2}$ under uniform distributions [58].

From the above definition, uniform quantization divides the range evenly despite the distribution of r. Empirically, the distributions of weights and activations of pre-trained DNNs are similar to bell-shaped Gaussian or Laplacian [17, 35]. Therefore, uniform quantization is not always able to achieve small enough approximation error to maintain model accuracy, especially in low-bit cases.

3.2 Piecewise Linear Quantization (PWLQ)

To improve model accuracy for quantized models, we need to approximate the original model as accurately as possible by minimizing the quantization error. We follow this natural criterion to investigate the quantization performance, even though no direct relationship can easily be established between the quantization error and the final model accuracy [7].

Inspired from [43, 26] that takes advantage of bell-shaped distributions, our approach based on piecewise linear quantization is designed to minimize the quantization error. It breaks the quantization range into two non-overlapping regions: the dense, central region and the sparse, high-magnitude region. An equal number of quantization levels $N = 2^b$ is assigned to these two regions. We chose to use two regions with one breakpoint to maintain simplicity in the inference algorithm (Section 5.1) and the hardware implementation (Section 4). Multiple-region cases are discussed in Section 5.1.

Therefore, we only consider one breakpoint p to divide the bounded quantization range[4] $[-m, m]$ $(m > 0)$ into two symmetric regions: the center region $R_1 = [-p, p]$ and the tail region $R_2 = [-m, -p) \cup (p, m]$. Each region consists of a negative piece and a positive piece. Within each of the four pieces, $(b-1)$-bit $(b \geq 2)$ uniform quantization (1) is applied such that including the sign every value in the quantization range is being represented into b-bit. We define the b-bit piecewise linear quantization (denoted by PWLQ) scheme as:

$$\mathrm{pw}(r; b, m, p) = \begin{cases} \mathrm{sign}(r) \times \mathrm{uni}(|r|; b-1, 0, p, 0), r \in R_1 \\ \mathrm{sign}(r) \times \mathrm{uni}(|r|; b-1, p, m, p), r \in R_2 \end{cases}, \tag{3}$$

where the sign of full-precision real number r is denoted by $\mathrm{sign}(r)$. The associated quantization error is defined as $\varepsilon_{pw} = \mathrm{pw}(r; b, m, p) - r$.

[4] Here we consider symmetric quantization range $[-m, m]$ $(m > 0)$ for simplicity, it is extendable to asymmetric ranges $[m_1, m_2]$ for any real numbers $m_1 < m_2$.

Figure 1 shows the comparison between uniform quantization and PWLQ on the empirical distribution of the *conv*4 layer weights in a pre-trained Inception-v3 model [54]. We emphasize that b-bit PWLQ represents FP32 values into b-bit integers to support b-bit multiply-accumulate operations, even though in total, it has the same number of quantization levels as $(b+1)$-bit uniform quantization. The implications of this are further discussed in Section 4.

3.3 Error Analysis

To study the quantization error for PWLQ, we suppose full-precision real number r has a symmetric probability density function (PDF) $f(r)$ on $[-m, m]$ with a cumulative distribution function (CDF) $F(r)$ satisfying $f(r) = f(-r)$, $F(-m) = 0, F(m) = 1$, and $F(r) = 1 - F(-r)$. Then, we calculate the expected quantization error squared of PWLQ from (2) based on the error of each piece:

$$\begin{aligned}\mathbb{E}(\varepsilon_{pw}^2; b, m, p) &= C(b-1)\Big\{(m-p)^2\big[F(-p) + 1 - F(p)\big] + p^2[F(p) - F(-p)]\Big\} \\ &= C(b-1)\Big\{(m-p)^2 + m(2p-m)\big[2F(p) - 1\big]\Big\}.\end{aligned} \tag{4}$$

The performance of a quantized model with PWLQ critically depends on the value of the breakpoint p. If $p = \frac{m}{2}$, then the PWLQ is essentially equivalent to uniform quantization, because the four pieces have equal quantization ranges and bit-widths. If $p < \frac{m}{2}$, the center region has a smaller range and greater precision than the tail region, as shown in the middle of Figure 1. Conversely, if $p > \frac{m}{2}$, the tail region has greater precision than the center region. To reduce the overall quantization error for bell-shaped distributions found in DNNs, we increase the precision in the center region and decrease it in the tail region. Thus, we limit the breakpoint to the range $0 < p < \frac{m}{2}$. Accordingly, the optimal breakpoint p^* can be estimated by minimizing the expected squared quantization error:

$$p^* = \arg\min_{p \in (0, \frac{m}{2})} \mathbb{E}(\varepsilon_{pw}^2; b, m, p). \tag{5}$$

Since bell-shaped distributions tend to zero as r becomes large, we consider a smooth $f(r)$ is decreasing when r is positive, i.e., $f'(r) < 0$, $\forall r > 0$. Then we prove that the optimization problem (5) is convex with respect to the breakpoint $p \in (0, \frac{m}{2})$. Therefore one unique p^* exists to minimize the quantization error (4), as demonstrated by the following **Lemma 1**.

Lemma 1 *If $f(-r) = f(r)$, $f'(r) < 0$ for all $r > 0$, then $\mathbb{E}(\varepsilon_{pw}^2; b, m, p)$ is a convex function of the breakpoint $p \in (0, \frac{m}{2})$.*

Proof. Taking the first and second derivatives of (4) yields:

$$\begin{aligned}\tfrac{\partial \mathbb{E}(\varepsilon_{pw}^2; b, m, p)}{\partial p} &= 2C(b-1)\Big[p - 2m + 2mF(p) + m(2p-m)f(p)\Big], \\ \tfrac{\partial^2 \mathbb{E}(\varepsilon_{pw}^2; b, m, p)}{\partial p^2} &= 2C(b-1)\Big[1 + 4mf(p) + m(2p-m)f'(p)\Big].\end{aligned} \tag{6}$$

Since $f'(p) < 0$ and $p < \frac{m}{2}$, $m(2p-m)f'(p) > 0$, then $\frac{\partial^2 \mathbb{E}(\varepsilon_{pw}^2; b, m, p)}{\partial p^2} > 0$. Therefore, $\mathbb{E}(\varepsilon_{pw}^2; b, m, p)$ is convex w.r.t. p, and thus a unique p^* exists.

In practice, we can find the optimal breakpoint by solving (5) by assuming an underlying Gaussian or Laplacian distribution using gradient descent [50]. Once the optimal breakpoint p^* is found, both **Lemma 2** and the numerical simulation in the right of Figure 1 show that PWLQ achieves a smaller quantization error than uniform quantization, which indicates its stronger representational power.

Lemma 2 $\mathbb{E}(\varepsilon_{pw}^2; b, m, p^*) < \frac{C(b-1)}{16C(b)}\mathbb{E}(\varepsilon_{uni}^2; b, -m, m)$ *for* $b \geq 2$.

Proof. The b-bit uniform quantization error on $[-m, m]$ is calculated from (2):

$$\mathbb{E}(\varepsilon_{uni}^2; b, -m, m) = C(b)(2m)^2 = 4C(b)m^2. \tag{7}$$

For b-bit PWLQ, we solve the convex problem (5) by letting the first derivative equal to zero in (6), and determine that the optimal breakpoint p^* satisfies:

$$2mF(p^*) = 2m - p^* + m(m - 2p^*)f(p^*). \tag{8}$$

By substituting (8) in (4) and simplifying, we obtain:

$$\mathbb{E}(\varepsilon_{pw}^2; b, m, p^*) = C(b-1)\Big[-(p^*)^2 + mp^* - m(m-2p^*)^2 f(p^*)\Big]. \tag{9}$$

Subtract the above from $\frac{C(b-1)}{16C(b)}$ of (7), we complete the proof:

$$\begin{aligned} &\mathbb{E}(\varepsilon_{pw}^2; b, m, p^*) - \tfrac{C(b-1)}{16C(b)}\mathbb{E}(\varepsilon_{uni}^2; b, -m, m) \\ &= \mathbb{E}(\varepsilon_{pw}^2; b, m, p^*) - C(b-1)(\tfrac{1}{4}m^2) \\ &\leq C(b-1)\Big[-(p^* - \tfrac{m}{2})^2 - m(m-2p^*)^2 f(p^*)\Big] < 0. \end{aligned} \tag{10}$$

Note that $C(b) = \frac{1}{12(2^b-1)^2}$ given from (2), for $b \geq 2$, $\frac{C(b-1)}{16C(b)} \leq \frac{9}{16}$. Therefore, b-bit PWLQ achieves a smaller quantization error, which is at most $\frac{9}{16}$ of b-bit uniform scheme. This improvement in performance requires only an extra bit for storage and no extra multiplication, as we discuss in the next section.

4 Hardware Impact

In this section, we discuss the hardware requirements for efficient deployment of DNNs quantized with PWLQ. In convolutional and fully-connected layers, every output can be computed using an inner product between vector X and vector W, which correspond to the input activation and weight (sub)tensors respectively.

From scheme (1), the approximated versions of uniform quantization are $\hat{X} = s_x X_q + z_x I$ and $\hat{W} = s_w W_q$ (assuming symmetric quantization for weights), where X_q and W_q are quantized integer vectors from X and W, I is an identity vector, s_x, s_w and z_x are associated constant-valued scaling factors and offset, respectively. The output of this uniform quantization is:

$$\langle \hat{X},\, \hat{W} \rangle = \langle s_x X_q + z_x I,\, s_w W_q \rangle = C_0 \langle X_q,\, W_q \rangle + C_1, \tag{11}$$

where $\langle \cdot, \cdot \rangle$ is defined as vector inner product, $C_0 = s_x s_w$ and $C_1 = z_x s_w \langle W_q, I \rangle$ denote floating-point constant terms that can be pre-computed offline.

Equation (11) implies that a uniformly quantized DNN requires two steps: (i) an integer-arithmetic (INT) inner product, and (ii) followed by a floating-point (FP) affine map. The expensive $O(|W|)$ (the size of vector W) FP operations $\langle \hat{X}, \hat{W} \rangle$ are then accelerated via INT operations $\langle X_q, W_q \rangle$, plus $O(1)$ FP re-scaling and adding operands using C_0 and C_1.

As we showed in Section 3.2 when applying PWLQ on weights with one breakpoint, the algorithm breaks the ranges into non-overlapping regions (R_1 and R_2), which requires separate computational paths (P_1 and P_2) as each region has a different scaling factor. We set offsets $z_{w_1} = 0, z_{w_2} = p$ and denote scaling factors by s_{w_1}, s_{w_2} in R_1, R_2, respectively. We also define by $\langle \cdot, \cdot \rangle_{R_i}$ the associated partial vector inner product, and W_{q_i} the associated quantized integer vector of W in region R_i for $i = 1, 2$. Then P_1 is computed using the following equation:

$$P_1 = \langle s_x X_q + z_x I,\ s_{w_1} W_{q_1} \rangle_{R_1} = C_2 \langle X_q,\ W_{q_1} \rangle_{R_1} + C_3. \tag{12}$$

P_2 has additional terms as it has a non-zero offset p:

$$P_2 = \langle s_x X_q + z_x I,\ s_{w_2} W_{q_2} + pI \rangle_{R_2} = C_4 \langle X_q,\ W_{q_2} \rangle_{R_2} + C_5 \langle X_q,\ I \rangle_{R_2} + C_6, \tag{13}$$

where C_2, C_3, C_4, C_5, and C_6 are constant terms, which can be pre-computed similar to C_0 and C_1 in (11).

As indicated by (12) and (13) for PWLQ compared to uniform quantization (11), the extra term $\langle X_q, I \rangle_{R_2}$ is needed due to the non-zero offset p, which sums up the activations corresponding to weights in R_2. Since most of the weights[5] are in R_1, these extra computations in R_2 rarely happen. In addition, FP re-scaling and adding are needed in each region, which also increases the overall FP operation overhead.

In short, an efficient hardware implementation of PWLQ requires: (i) one multiplier for products in both of $\langle X_q, W_{q_1} \rangle_{R_1}$ and $\langle X_q, W_{q_2} \rangle_{R_2}$, (ii) three accumulators: one of each for sum of products in P_1 and P_2, and another one for activations in P_2, and (iii) at most one extra bit for storage[6] per weight value to indicate the region. Note that this extra bit does not increase the multiply-accumulate (MAC) computation and it is only used to determine the appropriate accumulator, which can be done in hardware at negligible cost on the MAC unit.

Based on the above explanation, it is clear that more breakpoints require more accumulators and more storage bits per weight tensor. Also, applying PWLQ on both weights and activations requires accumulators for each combination of activation regions and weight regions, which translates to more hardware overhead. As a result, more than one breakpoint on the weight tensor or applying PWLQ on both weights and activations might not be feasible, from a hardware implementation perspective. We describe more details of the hardware implementation and its impact on energy and latency in the supplementary material.

[5] Around 90% of the weights are locating in the center region R_1 in our experiments.

[6] This extra storage cost can be further compressed by exploiting the non-uniform distribution of values [1, 43].

5 Experiments

We evaluate the robustness of our proposed PWLQ for post-training quantization on popular networks of several computer vision benchmarks: ImageNet classification [51], semantic segmentation and object detection on the Pascal VOC challenge [11]. We perform all experiments in Pytorch 1.2.0 [44]. Unless stated otherwise, we always apply batch normalization folding [25] and then quantize all folded network weights *per-channel*.

5.1 Ablation Study on ImageNet

In this section, we conduct experiments on the ImageNet classification challenge [51] and investigate the effectiveness of our proposed PWLQ method. We evaluate the top-1 accuracy performance on the validation dataset for three popular network architectures: Inception-v3 [54], ResNet-50 [19] and MobileNet-v2 [52]. We use torchvision[7] 0.4.0 and its pre-trained models for our experiments.

Activation Quantization. Throughout this paper, we use a *top-k* median method[8] with $k = 10$ to calibrate the activation range boundaries $[r_l, r_u]$. After sampling from 512 random training images [62], we sort the activations into an array X_{sort} at every layer. We then compute the median of the *top-k* smallest and largest values in X_{sort}, i.e., $r_l = \text{median}(X_{sort}[:k])$ and $r_u = \text{median}(X_{sort}[-k:])$. During inference, unless stated otherwise we apply 8-bit uniform quantization *per-layer* after clipping with these ranges. We report additional experiments applying PWLQ on activations in the supplementary material.

Optimal Breakpoint Selection. In order to apply PWLQ, we require the *optimal breakpoints* to divide the quantization ranges into *non-overlapping regions*. As stated in Section 3.3, we assume weights and activations satisfy Gaussian or Laplacian distributions, then we find the optimal breakpoints by solving (5).

For the case of one optimal breakpoint p^*, we can iteratively find it by gradient descent since the optimization problem (5) is convex; or using a simple and fast approximation of $p^*/m = \ln(0.8614m + 0.6079)$ for normalized Gaussian. Experimental results show that the approximation obtains almost the same accuracy compared to gradient descent, while also being considerably faster. Therefore, unless stated otherwise we use this approximated version of the optimal breakpoint for the rest of this paper. We report results with other assumptions such as Laplacian distributions in the supplementary material.

Other works treat the data distributions differently: BiScaled-DNN [26] proposes a ratio heuristic to divide the data into two overlapping regions; and V-Quant [43] introduces a value-aware method to split them into two non-overlapping regions, e.g, 2% (98%) of large (small) values located in the tail

[7] https://pytorch.org/docs/stable/torchvision

[8] We test the *top-k* median and percentile-based approaches [33] in the supplementary material and use the top-10 median for better robustness of low-bit quantization.

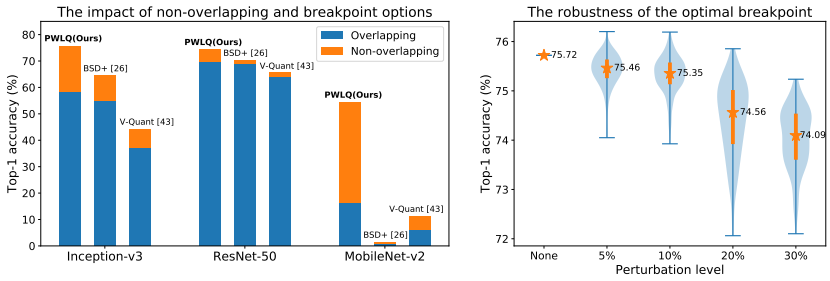

Fig. 2. Left: the impact of non-overlapping and breakpoint options on the top-1 accuracy for 4-bit post-training quantization models. Right: the robustness of the optimal breakpoint found by solving (5) with some perturbation levels from 5% to 30% for 4-bit Inception-v3. The star and the associated number indicate the median accuracy, the bold bar displays the accuracy range between the 25th and 75th percentiles

(center) region, respectively. Our implementation results in Figure 2 (left) show that PWLQ with non-overlapping regions achieves a superior performance on low-bit quantization compared to BiScaled-DNN improved version[9] (denoted by BSD+) and V-Quant, especially with a large margin on 4-bit MobileNet-v2. Non-overlapping approach shortens the quantization ranges (Δ in (2)) for the tail regions by 1.25× to 2×. Therefore, both our choices of *non-overlapping regions* and *optimal breakpoints* have a significant impact on reducing the quantization error and improving the performance of low-bit quantized models.

In Figure 2 (right), we explore the robustness of the optimal breakpoint found by minimizing (5) for 4-bit Inception-v3. We randomly add perturbation levels from 5% to 30% on each optimal breakpoint p^* per-channel per-layer, e.g., the new breakpoint $\widehat{p^*} = 0.95p^*$ or $1.05p^*$ for 5% of perturbation. We run 100 random samples for each perturbation level to generate the results. Overall, model performance decreases as perturbation level increases, which indicates that our selection of the optimal breakpoint is crucial for accurate post-training quantization. Note that when 5% of perturbation is added to the optimal breakpoints, more than half of the experiments produce a lower accuracy, and can be as low as 74.05%, which is a 1.67% drop from the zero-perturbation baseline.

Multiple Breakpoints. In this section, we discuss the trade-off of multiple breakpoints on model accuracy and hardware overhead. Theoretically, as the number of breakpoints on weights increases, the associated hardware cost linearly rises. Meanwhile, the number of non-overlapping regions and the associated total number of quantization levels grows, indicating a stronger representational power. Numerically, the extension of finding the optimal multi-breakpoints is straightforward by calculating the same quantization error (4), and solving the

[9] We improved the original BiScaled-DNN [26] by applying affine-based uniform scheme (1) on each region and per-channel quantization.

same optimization problem (5) with gradient descent in an enlarged search space. Table 1 shows the accuracy performance up to three breakpoints. In general, using more breakpoints consistently improves model accuracy under the growing support of customized hardware. We suggest using one breakpoint to maintain the simplicity of the inference algorithm and its hardware implementation. Thus we only report PWLQ with one breakpoint for the rest of this paper.

Table 1. Top-1 accuracy (%) and requirement of hardware accumulators for PWLQ with multiple breakpoints on weights

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**

PWLQ and Uniform Quantization. In Section 3.3, we analytically and numerically demonstrate that our method, PWLQ, obtains a smaller quantization error than uniform quantization. We compare these two schemes in Table 2. In this table, weights are quantized per-channel with the same computational bit-width $b = 4, 6, 8$; activations are uniformly quantized per-layer into 8-bit. Generally, PWLQ achieves higher accuracy than uniform quantization except for one minor case of 8-bit Inception-v3. When the bit-width is large enough ($b = 8$), the quantization error is small and both uniform quantization and PWLQ provide good accuracy. However, when the bit-width is decreased to 4, PWLQ obtains a notably higher accuracy, i.e., PWLQ attains 75.72% but uniform quantization only attains 44.28% for 4-bit Inception-v3. These results show that PWLQ is a more powerful representation scheme in terms of both quantization error and model accuracy, making it a viable alternative for uniform quantization in low bit-width cases. Moreover, PWLQ applies uniform quantization on each piece, hence it features a simple computational scheme and can benefit from any tricks that improve uniform quantization performance such as bias correction.

Bias Correction. An inherent bias in the mean and variance of the tensor values was observed after the quantization process and the benefits of correcting this bias term have been demonstrated in [2, 13, 42]. This bias can be compensated by folding certain correction terms into the scale and the offset [2]. We adopt this idea into our PWLQ method and show the results in Table 2 (columns with "+BC"). Applying bias correction further improves the performance of low-bit quantized models. It allows 6-bit post-training quantization with piecewise linear scheme for all three networks to achieve near full-precision accuracy within a drop of 0.30%; 4-bit MobileNet-v2, also without retraining, achieves an accuracy

Table 2. Comparison results of top-1 accuracy (%) for uniform and PWLQ schemes on weights. b+BC: b-bit with bias correction for bit-width $b = 4, 6, 8$. Each bold value indicates the best result from different methods for specified bit-width and network

Network	Weight Bit-width	8-bit	8+BC	6-bit	6+BC	4-bit	4+BC
Inception-v3	Uniform	77.53	77.52	76.87	77.24	44.28	62.46
(77.49)	**PWLQ (Ours)**	77.52	**77.53**	77.42	**77.48**	75.72	**76.45**
ResNet-50	Uniform	76.10	**76.14**	75.61	75.92	65.48	72.45
(76.13)	**PWLQ (Ours)**	76.10	76.10	76.03	**76.08**	74.28	**75.62**
MobileNet-v2	Uniform	71.35	71.58	67.76	70.81	11.37	41.80
(71.88)	**PWLQ (Ours)**	71.59	**71.73**	70.82	**71.58**	54.34	**69.22**

of 69.22%. In general, a combination of low-bit PWLQ and bias correction on weights achieves minimal loss of full-precision model performance.

5.2 Comparison to Existing Approaches

In this section, we compare our PWLQ method with other existing approaches, by quoting the reported performance scores from the original literature.

An inclusive evaluation of clipping techniques along with outlier channel splitting (OCS) was presented in [62]. To fairly compare with these methods, we adopt the same setup of applying per-layer quantization on weights and without quantizing the first layer. In Table 3, we show that our PWLQ (no bias correction) outperforms the best results of clipping method combined with OCS. Besides, OCS needs to change the network architecture, in contrast to PWLQ.

Table 3. Comparison results of per-layer PWLQ and best clipping with OCS [62] on top-1 accuracy (%) loss. W/A indicate the bit-width on weights/activations. The accuracy difference values are measured from the full-precision (32/32) result

Network	W/A	32/32	8/8	7/8	6/8	5/8	4/8
Inception-v3	OCS + Best Clip	75.9	-0.6 (75.3)	-1.2 (74.7)	-3.4 (72.5)	-13.0 (62.9)	-71.1 (4.8)
	PWLQ (Ours)	77.5	**+0.1 (77.6)**	**-0.1 (77.4)**	**-0.3 (77.2)**	**-2.0 (75.5)**	**-12.8 (64.7)**
ResNet-50	OCS + Best Clip	76.1	-0.4 (75.7)	-0.5 (75.6)	-0.9 (75.2)	-2.7 (73.4)	-6.8 (69.3)
	PWLQ (Ours)	76.1	**-0.0 (76.1)**	**-0.1 (76.0)**	**-0.2 (75.9)**	**-0.7 (75.5)**	**-2.4 (73.7)**

In Table 4, we provide a comprehensive comparison result of PWLQ to other existing methods. Here we apply per-layer quantization on activations and per-channel PWLQ on weights with bias correction. Except for the 4/4 case where we apply 4-bit PWLQ on activations, we always apply 8-bit uniform quantization on activations for the rest of the 8/8 and 4/8 cases. Under the same bit-width of computational cost among all the methods, our PWLQ combined with bias correction achieves the state-of-the-art results on all cases and it outperforms all other methods with a large margin on 4/8 and 4/4 cases. We emphasize that our PWLQ method is simple and efficient. It achieves the desired accuracy at

the small cost of a few more accumulations per MAC unit and a minor overhead of storage. More importantly, it is orthogonal and applicable to other methods.

Table 4. Comparison of our PWLQ and other methods on top-1 accuracy (%) loss. PWLQ: weights are piecewise linearly quantized per-channel with bias correction, activations are quantized per-layer

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)	-	-	-	-	-	-

5.3 Other Applications

To show the robustness and applicability of our proposed approach, we extend the PWLQ idea to other computer vision tasks including semantic segmentation on DeepLab-v3+ [5] and object detection on SSD [36].

Semantic Segmentation. In this section, we apply PWLQ on DeepLab-v3+ with a backbone of MobileNet-v2. The performance is evaluated using mean intersection over union (mIoU) on the Pascal VOC segmentation challenge [11].

In our experiments, we utilize the implementation of public Pytorch repository[10] to evaluate the performance. After folding batch normalization of the pre-trained model into the weights, we found that several layers of weight ranges become very large (e.g., [-54.4, 64.4]). Considering the fact that quantization range [27], especially in the early layers [7], has a profound impact on the performance of quantized models, we fix the configuration of some early layers in the backbone. More precisely, we apply 8-bit PWLQ on three depth-wise convolution layers with large ranges in all configurations shown in Table 5. Note that the MAC operations of these three layers are negligible in practice since they only contribute 0.2% of the entire network computation, but it is remarkably beneficial to the performance of low-bit quantized models.

As noticed in classification, low-bit uniform quantization causes significant accuracy drop from the full-precision models. In Table 5, applying PWLQ combined with bias correction, the 6-bit model on weights even outperforms 8-bit DFQ [42], which attains 0.42% degradation of the pre-trained model. Moreover, the 4-bit PWLQ significantly improves the mIoU by 17.61% from the 4-bit

[10] https://github.com/jfzhang95/pytorch-deeplab-xception

Table 5. Uniform quantization and PWLQ on DeepLab-v3+. Weights are quantized per-channel with bias correction, activations are uniformly quantized per-layer

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)	-	-

uniform quantized model, indicating the potential of low-bit post-training quantization via piecewise linear approximation for the semantic segmentation task.

Object Detection. We also test our PWLQ for object detection task. The experiments are performed on the public Pytorch implementation[11] of SSD-Lite version [36] with a backbone of MobileNet-v2. The performance is evaluated with mean average precision (mAP) on the Pascal VOC detection challenge [11].

Table 6. Uniform quantization and PWLQ of SSD-Lite version. Weights are quantized per-channel with bias correction, activations are uniformly quantized per-layer

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)	-	-

Table 6 compares the results of the mAP score of quantized models using the uniform and PWLQ schemes. Similar to image classification and semantic segmentation tasks, even with bias correction and per-channel quantization enhancements, 4-bit uniform scheme causes 3.91% performance drop from the full-precision model, while 4-bit PWLQ with these two enhancements is able to remove this notable gap down to 0.38%.

6 Conclusion

In this work, we present a piecewise linear quantization scheme for accurate post-training quantization of deep neural networks. It breaks the bell-shaped distributed values into non-overlapping regions per tensor where each region is assigned an equal number of quantization levels. We further analyze the resulting quantization error as well as the hardware requirements. We show that our approach achieves state-of-the-art low-bit post-training quantization performance on image classification, semantic segmentation, and object detection tasks under the same computational cost. It indicates its potential for efficient and rapid deployment of computer vision applications on resource-limited devices.

[11] https://github.com/qfgaohao/pytorch-ssd

References

1. Bakunas-Milanowski, D., Rego, V., Sang, J., Chansu, Y.: Efficient algorithms for stream compaction on gpus. International Journal of Networking and Computing **7**(2), 208–226 (2017)
2. Banner, R., Nahshan, Y., Hoffer, E., Soudry, D.: Post training 4-bit quantization of convolution networks for rapid-deployment. CoRR, abs/1810.05723 **1**, 2 (2018)
3. Baskin, C., Schwartz, E., Zheltonozhskii, E., Liss, N., Giryes, R., Bronstein, A.M., Mendelson, A.: Uniq: Uniform noise injection for non-uniform quantization of neural networks. arXiv preprint arXiv:1804.10969 (2018)
4. Cai, Z., He, X., Sun, J., Vasconcelos, N.: Deep learning with low precision by half-wave gaussian quantization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5918–5926 (2017)
5. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 801–818 (2018)
6. Choi, J., Wang, Z., Venkataramani, S., Chuang, P.I.J., Srinivasan, V., Gopalakrishnan, K.: Pact: Parameterized clipping activation for quantized neural networks. arXiv preprint arXiv:1805.06085 (2018)
7. Choukroun, Y., Kravchik, E., Kisilev, P.: Low-bit quantization of neural networks for efficient inference. arXiv preprint arXiv:1902.06822 (2019)
8. Courbariaux, M., Bengio, Y., David, J.P.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: Advances in neural information processing systems. pp. 3123–3131 (2015)
9. Dhillon, G.S., Azizzadenesheli, K., Lipton, Z.C., Bernstein, J., Kossaifi, J., Khanna, A., Anandkumar, A.: Stochastic activation pruning for robust adversarial defense. arXiv preprint arXiv:1803.01442 (2018)
10. Dong, X., Huang, J., Yang, Y., Yan, S.: More is less: A more complicated network with less inference complexity. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5840–5848 (2017)
11. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88**(2), 303–338 (2010)
12. Faraone, J., Fraser, N., Blott, M., Leong, P.H.: Syq: Learning symmetric quantization for efficient deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4300–4309 (2018)
13. Finkelstein, A., Almog, U., Grobman, M.: Fighting quantization bias with bias. arXiv preprint arXiv:1906.03193 (2019)
14. Georgiadis, G.: Accelerating convolutional neural networks via activation map compression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7085–7095 (2019)
15. Gong, Y., Liu, L., Yang, M., Bourdev, L.: Compressing deep convolutional networks using vector quantization. arXiv preprint arXiv:1412.6115 (2014)
16. Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P.: Deep learning with limited numerical precision. In: International Conference on Machine Learning. pp. 1737–1746 (2015)
17. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)

18. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
20. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1389–1397 (2017)
21. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
22. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
23. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
24. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
25. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2704–2713 (2018)
26. Jain, S., Venkataramani, S., Srinivasan, V., Choi, J., Gopalakrishnan, K., Chang, L.: Biscaled-dnn: Quantizing long-tailed datastructures with two scale factors for deep neural networks. In: 2019 56th ACM/IEEE Design Automation Conference (DAC). pp. 1–6. IEEE (2019)
27. Jung, S., Son, C., Lee, S., Son, J., Han, J.J., Kwak, Y., Hwang, S.J., Choi, C.: Learning to quantize deep networks by optimizing quantization intervals with task loss. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4350–4359 (2019)
28. Krishnamoorthi, R.: Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342 (2018)
29. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
30. Lai, W.S., Huang, J.B., Ahuja, N., Yang, M.H.: Fast and accurate image super-resolution with deep laplacian pyramid networks. IEEE transactions on pattern analysis and machine intelligence (2018)
31. Lee, J.H., Ha, S., Choi, S., Lee, W.J., Lee, S.: Quantization for rapid deployment of deep neural networks. arXiv preprint arXiv:1810.05488 (2018)
32. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016)
33. Li, R., Wang, Y., Liang, F., Qin, H., Yan, J., Fan, R.: Fully quantized network for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2810–2819 (2019)
34. Li, Y., Dong, X., Wang, W.: Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=BkgXT24tDS
35. Lin, D., Talathi, S., Annapureddy, S.: Fixed point quantization of deep convolutional networks. In: International Conference on Machine Learning. pp. 2849–2858 (2016)

36. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)
37. Luo, J.H., Wu, J., Lin, W.: Thinet: A filter level pruning method for deep neural network compression. In: Proceedings of the IEEE international conference on computer vision. pp. 5058–5066 (2017)
38. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 116–131 (2018)
39. Meller, E., Finkelstein, A., Almog, U., Grobman, M.: Same, same but different-recovering neural network quantization error through weight factorization. arXiv preprint arXiv:1902.01917 (2019)
40. Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al.: Mixed precision training. arXiv preprint arXiv:1710.03740 (2017)
41. Miyashita, D., Lee, E.H., Murmann, B.: Convolutional neural networks using logarithmic data representation. arXiv preprint arXiv:1603.01025 (2016)
42. Nagel, M., van Baalen, M., Blankevoort, T., Welling, M.: Data-free quantization through weight equalization and bias correction. arXiv preprint arXiv:1906.04721 (2019)
43. Park, E., Yoo, S., Vajda, P.: Value-aware quantization for training and inference of neural networks. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 580–595 (2018)
44. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. 31st Conference on Neural Information Processing Systems (2017)
45. Polino, A., Pascanu, R., Alistarh, D.: Model compression via distillation and quantization. arXiv preprint arXiv:1802.05668 (2018)
46. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. In: European Conference on Computer Vision. pp. 525–542. Springer (2016)
47. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7263–7271 (2017)
48. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
49. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
50. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. nature **323**(6088), 533–536 (1986)
51. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
52. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520 (2018)
53. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

54. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2818–2826 (2016)
55. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019)
56. Ullrich, K., Meeds, E., Welling, M.: Soft weight-sharing for neural network compression. arXiv preprint arXiv:1702.04008 (2017)
57. Wu, J., Leng, C., Wang, Y., Hu, Q., Cheng, J.: Quantized convolutional neural networks for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4820–4828 (2016)
58. You, Y.: Audio Coding: Theory and Applications. Springer Science & Business Media (2010)
59. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
60. Zhang, D., Yang, J., Ye, D., Hua, G.: Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 365–382 (2018)
61. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6848–6856 (2018)
62. Zhao, R., Hu, Y., Dotzel, J., De Sa, C., Zhang, Z.: Improving neural network quantization without retraining using outlier channel splitting. In: International Conference on Machine Learning. pp. 7543–7552 (2019)
63. Zhou, A., Yao, A., Guo, Y., Xu, L., Chen, Y.: Incremental network quantization: Towards lossless cnns with low-precision weights. arXiv preprint arXiv:1702.03044 (2017)
64. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., Zou, Y.: Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160 (2016)
65. Zhou, Y., Moosavi-Dezfooli, S.M., Cheung, N.M., Frossard, P.: Adaptive quantization for deep neural network. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
66. Zhu, C., Han, S., Mao, H., Dally, W.J.: Trained ternary quantization. arXiv preprint arXiv:1612.01064 (2016)