AQD: Towards Accurate Quantized Object Detection

Peng Chen[2*] Jing Liu[1*] Bohan Zhuang[1†] Mingkui Tan[3] Chunhua Shen[1,2]
[1]Monash University [2]The University of Adelaide [3]South China University of Technology

Abstract

Network quantization allows inference to be conducted using low-precision arithmetic for improved inference efficiency of deep neural networks on edge devices. However, designing aggressively low-bit (e.g., 2-bit) quantization schemes on complex tasks, such as object detection, still remains challenging in terms of severe performance degradation and unverifiable efficiency on common hardware. In this paper, we propose an Accurate Quantized object Detection solution, termed AQD, to fully get rid of floating-point computation. To this end, we target using fixed-point operations in all kinds of layers, including the convolutional layers, normalization layers, and skip connections, allowing the inference to be executed using integer-only arithmetic. To demonstrate the improved latency-vs-accuracy trade-off, we apply the proposed methods on RetinaNet and FCOS. In particular, experimental results on MS-COCO dataset show that our AQD achieves comparable or even better performance compared with the full-precision counterpart under extremely low-bit schemes, which is of great practical value. Source code and models are available at: https://github.com/aim-uofa/model-quantization

1. Introduction

Deep neural networks (DNNs) have achieved great success in many computer vision tasks, such as image classification [11, 21, 12], semantic segmentation [31, 10, 5], object detection [37, 27, 26], etc. However, DNNs are always equipped with a large number of parameters and consume heavy computational resources, which hinders their applications, especially on resource-constrained devices such as smartphones and drones. To reduce the memory footprint and computational burden, several network compression methods have been proposed, such as channel pruning [13, 32, 53], efficient architecture design [15, 39, 35] and network quantization [49, 52, 52].

In particular, network quantization aims to project floating-point values onto a spaced grid, where the original floating-point values can be approximated by a set of discrete values. In this way, the compute-intensive floating-point operations can be replaced by power-efficient fixed-point or bitwise operations, which greatly reduces the computational cost of the networks.

Table 1 – Energy and Area cost for different precision operations (on 45nm CMOS technology) [9, 14, 25].

Operation	Energy (pJ)	Area (μm^2)
16-bit Floating-point Add	0.4	1360
16-bit Floating-point Mult	1.1	1640
32-bit Floating-point Add	0.9	4184
32-bit Floating-point Mult	3.7	7700
8-bit Fixed-point Add	0.03	36
8-bit Fixed-point Mult	0.2	282
32-bit Fixed-point Add	0.1	137
32-bit Fixed-point Mult	3.1	3495

Recently, many quantization methods [52, 20, 8] have been proposed and achieved promising results on some tasks such as image classification. However, using aggressively low-bit quantized networks for more complex tasks such as object detection still remains a challenge. Developing quantized object detectors is a challenging task since a detector not only performs object classification, but also needs to predict other rich information, such as the locations of bounding boxes for regression. Some existing quantized object detection methods [18, 50] reduce the precision of detectors to 4 or 8 bits and achieve promising performance. However, when it comes to aggressively low bitwidth (*e.g.*, 2-bit) quantization, directly quantizing the detector incurs a significant performance drop compared to their full-precision counterpart. Moreover, some layers (*e.g.*, batch normalization, and skip connections) in the network still require floating-point arithmetic units for inference. This means both integer and floating-point arithmetic units are needed for inference. As shown in Table 1, compared with fixed-point operations, floating-point operations consume much higher energy and area cost. Besides, data exchange between different types of arithmetic units may further hamper the energy efficiency of the network.

*First two authors contributed equally.
†Corresponding author. E-mail: bohan.zhuang@monash.edu

In this paper, we propose an Accurate Quantized object Detection (AQD) method to fully get rid of floating-point computation while maintaining performance. To this end, we propose to replace floating-point operations with fixed-point operations in all kinds of layers, including the convolutional layers, normalization layers and skip connections. In this way, only integer arithmetic is required during inference, which significantly reduces the computational overheads. To reduce the performance drop from quantization while ensuring pure fixed-point operations, we further propose a new variant of batch normalization (BN) called multi-level BN. Our proposed method is based on the observation there is a large divergence of batch statistics across different feature pyramid levels, where batch statistics are computed using aggressively quantized activations. Therefore, using shared BN statistics in conventional detection frameworks [41, 27] will result in highly poor estimates of statistical quantities. To capture accurate batch statistics, multi-level BN privatizes batch normalization layers for each pyramid level of the head.

Our main contributions are summarized as follows:

- We propose an Accurate Quantized object Detection (AQD) method to fully get rid of floating-point computation in each layer of the network, including convolutional layers, normalization layers and skip connections. As a result, only integer arithmetic is required during inference, which greatly improves the on-device efficiency to carry out inference.
- We highlight that the degraded performance of the quantized detectors is largely due to the inaccurate batch statistics in the network. We therefore propose multi-level batch normalization to capture accurate batch statistics of different levels of feature pyramid.
- We evaluate the proposed methods on the COCO detection benchmark with multiple precisions. Experimental results show that our low bit AQD can achieve comparable or even better performance with its full-precision counterpart.

2. Related work

Network quantization. Network quantization aims to reduce the model size and computational cost by representing the network weights and/or activations with low precision. Existing methods can be divided into two categories, namely, binary quantization [16, 36, 3, 30, 29] and fixed-point quantization [49, 52, 46, 20, 8]. Binary quantization converts the full-precision weights and activations to $\{+1, -1\}$, where the convolution operations are replaced with efficient bitwise operations and can achieve up to $32\times$ memory saving and $58\times$ speedup on CPUs [36, 51]. To reduce the performance gap between the quantized model and the full-precision counterpart, fixed-point quantization methods [49, 4, 52, 6, 46] represent weights and activations with higher bitwidths, showing impressive performance on the image classification task. Besides, some logarithmic quantizers [48, 33, 24] leverage bit-shift operations to accelerate the computation. However, they impose constraints on the quantization algorithm. For example, the quantized activations need to be fixed-point values and the quantized weights are required to be powers-of-2 values, which might result in lower quantization performance.

Apart from the algorithm design, the development of underlying implementation and acceleration libraries [7, 47, 19] are critical to enable highly-efficient execution on resource-constrained platforms. In particular, low-precision training methods [1, 40, 42] quantize weights, activations and gradients to carefully designed data format for improved efficiency while preserving accuracy. To improve the inference efficiency, several methods [47, 19] propose to design efficient bitwise operations on dedicated hardware devices, such as ARM, FPGA and ASIC.

Quantization on Object Detection. Many researchers have studied quantization on object detection to speed up on-device inference and save storage. Wei *et al*. [43] utilize knowledge distillation and quantization to train very tiny CNNs for object detection. Zhuang *et al*. [50] point out the difficulty of propagating gradient and propose to train low-precision network with a full-precision auxiliary module. These works achieve promising quantization performance on object detection. However, they do not quantize all the layers (*e.g*., input and output layers, BN or skip connections), which limits the efficient deployment on resource-constrained platforms. Jacob *et al*. [18] propose a quantization scheme using integer-only arithmetic and perform object detection on COCO dataset with 8-bit precision. Furthermore, when carrying out more aggressive quantization, Li *et al*. [23] observe training instability during the quantized fine-tuning and propose three solutions. However, these works impose extra constraints on both the network structure and quantization algorithm design, which limits them to obtain better performance (refer to Sec. 3.3). In contrast, our quantization scheme is milder and more flexible in aspects of network structure and quantization algorithms which contributes to significant performance improvement of proposed AQD over several state-of-the-art quantized object detectors.

3. Proposed method

3.1. Preliminary

3.1.1 Integer-aware Quantized Representation

Full-precision (32-bit) floating-point is a general data type that is used to represent data in deep learning models. With network quantization, the continuous full-precision weights

and activations are discretized to a limited number of quantized values. Specifically, given a specific bitwidth b, the total number of quantized values is 2^b. To enable efficient integer arithmetic operations on the quantized values, it requires the quantization scheme to be a mapping of real values to integers. Formally, for b-bit quantization of any full-precision value x in a tensor $\mathbf{X}$, its quantized version $\overline{x}$ can be formulated as:

$$\overline{x} = \eta \cdot \alpha, \tag{1}$$

where α is a full-precision floating-point scale factor shared for the whole tensor $\mathbf{X}$, and $\eta \in \mathbb{N}$ implies the corresponding mapping value in the integer domain.

3.1.2 Quantization function

In this work, we propose to quantize both weights and activations with learnable quantization intervals motivated by LSQ [8]. Without loss of generality, given a convolutional layer or fully-connected layer in a network, the weight $\mathbf{W}$ is convolved with the input activation $\mathbf{X}$, where $\mathbf{W}$ and $\mathbf{X}$ are real-valued tensors. We use $x, w \in \mathbb{R}$ to denote the element of $\mathbf{X}$ and $\mathbf{W}$ respectively. Let ν_x and ν_w be the trainable quantization interval parameters that indicate the range of activations and weights to be quantized, which are shared for all elements in $\mathbf{X}$ and $\mathbf{W}$, respectively.

For a given x, we first constrain it to the range $[0, \nu_x]$, with values out of the range clipped into boundaries. We then linearly map values in the interval $[0, \nu_x]$ to the integer domain of $\{0, 1, \ldots, 2^b - 1\}$, where b is the quantization bitwidth. At last, we restore the magnitude of the original x by multiplying the corresponding scale factor. Formally, the quantization process can be formulated as follows:

$$\begin{aligned} \eta_x &= \lfloor \mathrm{clip}(\frac{x}{\nu_x}, 0, 1) \cdot (2^b - 1) \rceil, \\ \overline{x} &= \eta_x \cdot \frac{\nu_x}{2^b - 1}, \end{aligned} \tag{2}$$

where $\lfloor \cdot \rceil$ returns the nearest integer of a given value, $\mathrm{clip}\left(x, x_{\mathrm{low}}, x_{\mathrm{up}}\right) = \min(\max(x, x_{\mathrm{low}}), x_{\mathrm{up}})$, η_x is the corresponding integer-domain mapping value of x, and $\frac{\nu_x}{2^b - 1}$ is the magnitude-restore scale factor.

For the given w from weights, the quantization interval is defined as $[-\nu_w, \nu_w]$. We compute the quantized weight $\overline{w}$ similar to $\overline{x}$ except that an additional transformation is applied , which can be formulated as:

$$\begin{aligned} \eta_w &= \lfloor (\mathrm{clip}(\frac{w}{\nu_w}, -1, 1) + 1)/2 \cdot (2^b - 1) \rceil, \\ \overline{w} &= (\eta_w \cdot \frac{1}{2^b - 1} \cdot 2 - 1) \cdot \nu_w, \end{aligned} \tag{3}$$

where η_w is the corresponding integer-domain mapping value of w.

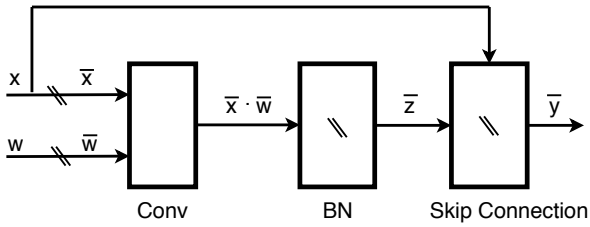


Figure 1 – Illustration of a typical block in the detection network. Double slash lines indicate the positions where quantization applies. Note that in a fully-quantized network, x can be represented in the format of Eq. (1), which is the output $\overline{y}$ of the preceding block.

During network training, the discretization operation by rounding function $\lfloor \cdot \rceil$ is non-differentiable. To avoid gradient vanishing issue, we employ the Straight-Through Estimator (STE) for back-propagation [2].

3.2. Floating-point Free Quantization

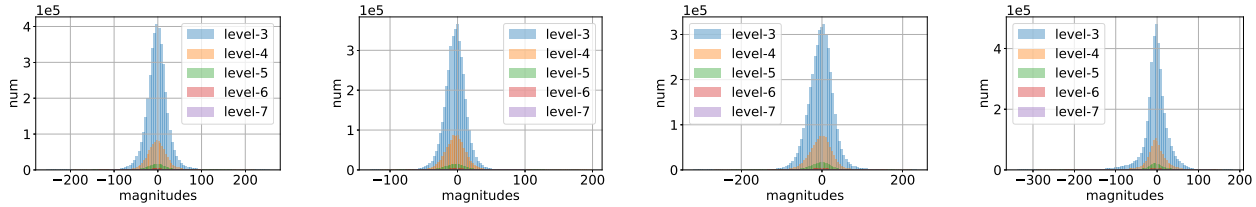
In this paper, we present an Accurate Quantized object Detection (AQD) method that fully gets rid of floating-point computation while still achieving promising performance. In a conventional residual block as shown in Figure 1, we propose to substitute floating-point operations for fixed-point operations in all kinds of layers, including the convolutional layer, batch normalization layer and the skip connection. In this way, only integer arithmetic is required during network inference, which greatly reduces the computational overhead and memory footprint of the network. In the following subsections, we will introduce the details of the proposed method regarding fixed-point operations for each of these layers.

3.2.1 Convolutional and Fully-connected Layers

As elaborated in Sec. 3.1.1, we define a unified quantization representation that allows for integer-only arithmetic, and all tensors in the quantized network are required to follow the rule in Eq. (1). We can observe that the quantized activation $\overline{x}$ and weight $\overline{w}$ are compatible with this representation format. Furthermore, the output activation of a quantized convolutional or fully-connected layer can fit the format as well:

$$\begin{aligned} \overline{x} \cdot \overline{w} &= (\eta_x \cdot \frac{\nu_x}{2^b - 1}) \cdot ((\eta_w \cdot \frac{1}{2^b - 1} \cdot 2 - 1) \cdot \nu_w), \\ &= (\eta_x \cdot (2 \cdot \eta_w - 2^b + 1)) \cdot \frac{\nu_x \cdot \nu_w}{(2^b - 1)^2}, \\ &= \eta_{\mathrm{conv}} \cdot \alpha_{\mathrm{conv}}. \end{aligned} \tag{4}$$

Benefit from the data representation in Eq. (1), the scale factors ν_x and ν_w can be handled independently from η_x



(a) 3-th layer of classification head (b) 3-th layer of regression head (c) 4-th layer of classification head (d) 4-th layer of regression head

Figure 2 – Distribution of input activations at the batch normalization layer in the detection heads of a 2-bit ResNet-18 FCOS detector. Level-x denotes that the predictions are made on the x-th pyramid level. Different levels of features show different batch of statistics.

and η_w. Being shared for the whole activations and fixed in inference, the floating-point scale factor α_{conv} can be passed to the proceeding layer directly. In this case, only fixed-point operations exist in the quantized convolutional or fully-connected layer, which can greatly reduce the computational cost.

3.2.2 Normalization Layers

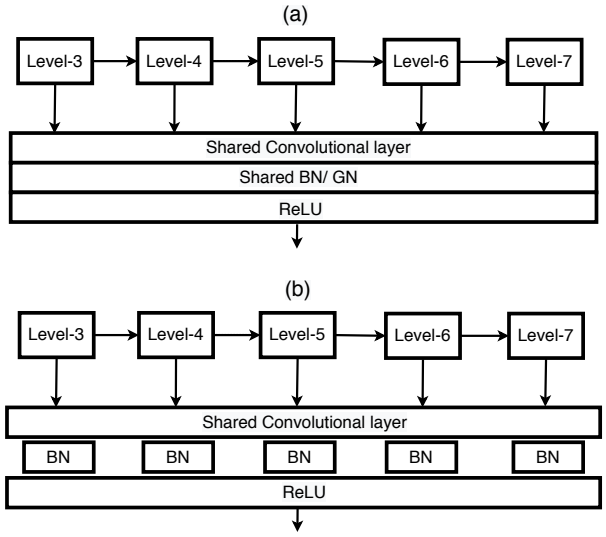


Figure 3 – Illustration of the proposed quantized detection heads design. (a) Conventional detection frameworks with shared group normalization or batch normalization [41, 27]; (b) The proposed multi-level BN that privatizes batch normalization layers for different heads.

Both batch normalization (BN) [17] and group normalization (GN) [44] are widely used in classical object detection networks [27, 41, 22]. We investigate the differences of these two normalization operations and analyze their impact on network quantization and inference.

In particular, BN is employed mainly in the backbone module for feature extraction while GN is preferred in the detection head, which is generally shared to handle extracted feature of different levels in the feature pyramid network (FPN) [26]. Both BN and GN normalize the input activations using mean and variance with an optional affine transformation. However, these two normalization strategies differ in two aspects. On the one hand, a BN layer calculates the statistics of the whole mini-batch while those of GN only reply on the individual input tensor in the mini-batch. Thus, BN is more sensitive to the training batch size, and might suffer from inaccurate statistics when the batch size is small. In contrast, GN behaves more stable to the training batch size. On the other hand, BN keeps track of exponential moving average mean μ and variance σ, and updates them using the minibatch statistics at each forward step during training, while μ and σ are **fixed** during inference. In contrast, GN **re-computes** the statistics information in each forward step during both training and inference procedures. As a result, GN unavoidably incurs floating-point computation during inference. Moreover, compared with BN, which can be potentially fused into the corresponding convolutional layer with no extra computational cost, GN imposes more execution burden during inference.

Therefore, to allow for integer-only arithmetic, we suggest to use batch normalization to replace group normalization in quantized detection networks. However, this replacement might cause training instability since batch statistics are computed using aggressively quantized activations, especially a small training batch size is commonly used. Fortunately, when equipped with the synchronized version of batch normalization (Sync-BN), the inaccurate statistics issue of batch normalization can be partially addressed [34]. However, another issue may occur when FPN [26] is employed in the detectors. We can observe a large divergence of batch statistics between different feature pyramid levels, as shown in Figure 2. Therefore, using shared BN statistics across prediction heads may lead to highly degraded estimates of statistical quantities, which will cause a significant performance drop. To solve this issue, we propose multi-level batch normalization (multi-level BN) to attain better quantization performance. Specifically, multi-level BN pri-

vatizes batch normalization layers for the shared convolutional layers in different detection heads, as illustrated in Figure 3. The proposed multi-level BN can capture individual batch statistics of the corresponding feature level. Furthermore, only negligible parameters (less than 1.1% of the model size) are introduced. Besides, it should be noted that only one batch normalization is activated at a time though multiple batch normalizations are allocated (since extracted features from different levels are computed sequentially). Thus, the proposed multi-level BN will not increase the computational cost compared with the traditional shared batch normalization (or group normalization).

To enable integer-only computation, we now design the quantization strategy for the batch normalization layer to remove the floating-point operations. We assume the previous layer is a quantized convolutional or fully-connected layer, whose output is $\overline{x} \cdot \overline{w}$ as explained in Eq. (4). Batch normalization layer then takes it as input and obtains the normalized output z. Before quantizing the batch normalization layer, the normalized value z can be computed by

$$\begin{aligned} z &= \frac{\eta_{\text{conv}} \cdot \alpha_{\text{conv}} - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma + \beta, \\ &= (\eta_{\text{conv}} + \frac{\beta \cdot \frac{\sqrt{\sigma^2+\epsilon}}{\gamma} - \mu}{\alpha_{\text{conv}}}) \cdot \frac{\alpha_{\text{conv}} \cdot \gamma}{\sqrt{\sigma^2 + \epsilon}}, \\ &= (\eta_{\text{conv}} + s) \cdot \alpha_z, \end{aligned} \tag{5}$$

where γ, β are the affine transformation parameters, ϵ is a constant to stabilize the computation. Then we are able to obtain the quantized batch normalization layer by quantizing s:

$$\overline{z} = (\eta_{\text{conv}} + \lfloor s \rceil) \cdot \alpha_z = \eta_z \cdot \alpha_z. \tag{6}$$

Obviously, for quantized batch normalization, the main computation is the add operation between η_{conv} and $\lfloor s \rceil$ and the floating-point factor α_z can be passed to next layer.

It is worth noting that, compared with the output of the quantized convolutional or fully-connected layer, the quantized output $\overline{z}$ of the batch normalization layer also meets the integer-aware representation in Eq. (1), but with the channel-wise scale factor α_z.

3.2.3 Skip Connections

Skip connections are commonly applied in the detectors with the ResNet [11] backbone and the FPN module. In this subsection, we elaborate how to enable integer-only computation of skip connections (*i.e.*, add operation between two tensors). Diving into the structure of the object detection network, we can learn that skip connections are employed mainly in three occasions: 1) Fusion of the identity mapping x and batch normalization output $\overline{z}$ in a residual block as illustrated in Figure 1. 2) Fusion of the batch normalization output of the downsampling branch and the main branch in a residual block. 3) Fusion of the up-scaled feature map and the current level feature map in FPN. For either of these occasions, given two input quantized values x_1 and x_2, the output y of skip connection and its quantized value $\overline{y}$ are computed as follows:

$$\begin{aligned} y &= x_1 + x_2 = \eta_1 \cdot \alpha_1 + \eta_2 \cdot \alpha_2, \\ \overline{y} &= \begin{cases} (\eta_1 + \eta_2 \cdot F(\alpha_2, \alpha_1)) \cdot \alpha_1 & \text{if } \alpha_2 \geq \alpha_1 \\ (\eta_1 \cdot F(\alpha_1, \alpha_2) + \eta_2) \cdot \alpha_2 & \text{otherwise} \end{cases}, \end{aligned} \tag{7}$$

where function $F(\cdot)$ is defined as

$$F(m, n) = \frac{c}{2^d}, \text{where } c, d = \underset{c \in \mathbb{N}, d \in \mathbb{N}_+}{\arg\min} |\frac{m}{n} - \frac{c}{2^d}|. \tag{8}$$

Function $F(\cdot)$ actually is used for searching an approximation of a given fraction $\frac{m}{n}$ (where $m \geq n$). In particular, the numerator c is constrained to be an integer and the denominator 2^d is constrained to be powers-of-two to allow for bit-shift (d is a positive integer). Function $F(\cdot)$ finds the best combination of c and d to approximate $\frac{m}{n}$. There is limited choice for d, for example $d \in \{0, 1, 2, \ldots, 31\}$ if η_1 and η_2 are represented in 32-bit integer. Therefore, function $F(\cdot)$ can be quickly solved, and Eq. (7) can be implemented with fixed-point operations only (the scale factors α_1 and α_2 do not participate in the computation and are passed to the next layer directly). It is worth mentioning that Eq. (7) is conducted in a channel-wise manner since the scale factor is applied on each channel in the batch normalization layer.

3.2.4 Other Layers

Layers such as max-pooling and nearest interpolation do not introduce floating-point computation. Besides, they do not change the scale factors of the layer input. Thus, there is no special modifications for these layers.

3.3. Discussions

Related to our work, FQN [23] also targets on floating-point-free arithmetic operations during inference. Compared with FQN, our method imposes milder constraints for both the network structure and quantization algorithm design, to get rid of floating-point computation.

On the one hand, instead of quantizing the batch normalization in standalone during training, FQN employs batch normalization folding (BN folding), which fuses the batch normalization layer into the preceding convolutional layer to simulate the quantization effect. However, as indicated in FQN, training instability is observed for the batch normalization parameters and batch statistics after BN folding is leveraged since considerable quantization noise is introduced especially for the extremely low bitwidth. To address this issue, FQN proposes *freezed batch normalization* in

which batch statistics are fixed during quantized finetuning. Instead, we propose to employ the standard batch normalization with statistics and parameters updated, but with the multi-level design.

On the other hand, FQN imposes constraints on the quantization algorithm. In particular, they require the scale factors of the two quantized input activations in the skip connection to satisfy

$$\frac{\alpha_1}{\alpha_2} = 2^d, \tag{9}$$

where d is an integer (different with the scope in Eq. (8), d here can be either positive or negative). This constraint is equivalent to Eq. (8) by fixing c to be constant 1. To meet such a condition, dedicated design of the quantization algorithm for the preceding convolutional layers is required due to the scale factors. In contrast, our method does not make such an assumption. Actually, the proposed method allows to quantize different layers independently. With the proposed simple yet mild design, our method shows superior performance over FQN as demonstrated in Sec. 4.

4. Experiments

We evaluate our proposed method on the COCO detection benchmarks dataset [28]. COCO detection benchmark is a large-scale benchmark dataset for object detection, which is widely used to evaluate the performance of detectors. Following [26, 51], we use the COCO *trainval35k* split (115K images) for training and *minival* split (5K images) for validation.

Comparison methods. To investigate the effectiveness of the proposed method, we compare with several state-of-the-art quantized object detection methods, including Auxi [50], Group-Net [51] and FQN [23]. We also define the following methods for study: **AQD***: Following [51, 50] we quantize all the convolutional layers, except the input layer in the backbone and the output layers in the detection heads, which acts as a strong baseline in our work. **AQD**: We quantize all network layers, including the input and output layers, batch normalization and skip connection layers, which is our complete method. Input and output layers are quantized to 8-bit in all settings.

Implementation details. We implement the proposed method based on Facebook Detectron2 [45]. We apply the proposed AQD on two classical one-stage object detectors, namely, RetinaNet [27] and FCOS [41]. We use ResNet [11] for the backbone module. To stabilize the optimization, we add batch normalization layers and ReLU non-linearities after the convolutional layers by default. For data pre-processing, we follow the strategy in [23, 50] to resize images with a shorter edge to 800 pixels in the training and validation set. Besides, images are augmented by random horizontal flipping during training. We do not perform any augmentations during evaluation. Total 90K iterations are trained with a mini-batch size of 16. We use SGD optimizer with a momentum of 0.9 for optimization. The learning rate is initialized to 0.01, and divided by 10 at iterations 60K and 80K, respectively. We set the weight decay to 0.0001. More detailed settings on the other hyper-parameters can be found in [27, 41]. Training hyper-parameters are the same for the quantized network and the full-precision counterpart, except for the initialization strategy. Specifically, to train the full-precision models, backbones are pre-trained on the ImageNet [38] classification task. Parameters from other parts are randomly initialized. Whereas, for training the quantized detector, the whole network is initialized by the full-precision counterpart.

4.1. Comparison with State-of-the-art Methods

We compare the proposed method with several state-of-the-arts and report the results in Tables 2 and 4. Quantization performance of 4/3/2-bit networks are listed in the tables. Full-precision performance is also provided for comparison. In particular, Group-Net [51] employs 4 binary bases, which corresponds to 2-bit fixed-point quantization.

Based on the results from Tables 2 and 4, we have the following observations. Firstly, our AQD consistently outperforms the compared baselines on different detection frameworks and backbones. For example, our 4-bit RetinaNet detector with ResNet-18 backbone outperforms FQN [23] and Auxi [50] by 5.5% and 2.2% on AP, respectively. Besides, our 2-bit FCOS detector obtains 2.9% and 2.7% AP improvement over the Group-Net with ResNet-18 and ResNet-50 backbones, separately. Moreover, our 4-bit quantized detectors can even outperform the corresponding full-precision models in some cases. Specifically, on a 4-bit RetinaNet detector, our AQD surpasses the full-precision model by 1.8% and 0.8% on AP with ResNet-18 and ResNet-34 backbones, respectively. Furthermore, when performing 3-bit quantization, our AQD achieves near lossless performance compared with the full-precision counterpart. For example, on a 3-bit RetinaNet detector with ResNet-50 backbone, our AQD only leads to 0.9% performance degradation on AP. Lastly, when conducting aggressive 2-bit quantization, our AQD still achieves promising performance. For example, our fully-quantized 2-bit RetinaNet detector with ResNet-50 backbone only incurs 3.0% AP loss compared with its full-precision baseline, but with considerable computation saving. These results justify the superior performance of our proposed AQD.

4.2. Ablation Study

Effect of Multi-level Batch Normalization. To study the effect of multi-level BN, we quantize the FCOS detector with multi-level BN, shared GN (or BN) and report the results in Table 3. Here, the detector with shared GN (or

Table 2 – Performance comparisons on the COCO validation set based on RetinaNet.

Backbone	Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-18	Full-precision	32.3	50.9	34.2	18.9	35.6	42.5
	FQN [23] (4-bit)	28.6	46.9	29.9	14.9	31.2	38.7
	Auxi [50] (4-bit)	31.9	50.4	33.7	16.5	34.6	42.3
	AQD* (4-bit)	34.1	53.4	36.4	19.8	36.4	44.7
	AQD (4-bit)	34.1	53.1	36.3	19.4	36.4	45.0
	AQD* (3-bit)	33.5	52.5	35.6	17.8	35.9	44.9
	AQD (3-bit)	33.4	52.8	35.7	17.9	36.4	43.9
	AQD* (2-bit)	31.0	49.5	32.7	17.0	33.1	41.4
	AQD (2-bit)	30.8	50.0	32.3	16.5	33.1	41.5
ResNet-34	Full-precision	36.3	56.2	39.1	22.4	39.8	46.9
	FQN [23] (4-bit)	31.3	50.4	33.3	16.1	34.4	41.6
	Auxi [50] (4-bit)	34.7	53.7	36.9	19.3	38.0	45.9
	AQD* (4-bit)	37.1	56.8	40.0	21.8	40.3	48.1
	AQD (4-bit)	37.1	56.8	39.8	21.9	40.0	48.0
	AQD* (3-bit)	36.5	56.3	38.9	21.2	39.4	48.2
	AQD (3-bit)	36.5	56.3	38.8	21.4	39.5	47.7
	AQD* (2-bit)	34.3	53.8	36.4	19.6	37.0	45.3
	AQD (2-bit)	33.8	54.0	36.1	19.4	36.8	44.8
ResNet-50	Full-precision	37.8	58.0	40.8	23.8	41.6	48.9
	FQN [23] (4-bit)	32.5	51.5	34.7	17.3	35.6	42.6
	Auxi [50] (4-bit)	36.1	55.8	38.9	21.2	39.9	46.3
	AQD* (4-bit)	38.1	58.5	41.3	23.9	41.8	48.7
	AQD (4-bit)	38.1	58.1	40.7	22.5	41.6	49.8
	AQD* (3-bit)	37.2	57.4	39.5	23.0	40.8	47.8
	AQD (3-bit)	36.9	57.1	39.5	22.0	40.6	47.9
	AQD* (2-bit)	35.0	55.0	37.2	20.6	38.4	45.5
	AQD (2-bit)	34.8	55.4	36.9	20.3	37.9	45.6

Table 3 – Effect of the multi-level batch normalization. We evaluate performance of both full-precision (FP) and 2-bit quantized models based on FCOS on the COCO validation set.

Backbone	Normalization	Precision	Shared	Fixed-point-only	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-18	BatchNorm	FP	✓		29.5	46.6	31.7	19.0	32.8	35.8
	GroupNorm		✓		34.0	51.7	36.3	19.7	36.6	44.0
	Multi-level BatchNorm				33.9	51.2	36.4	19.3	36.2	44.0
	BatchNorm	2-bit	✓	✓	26.4	43.6	28.2	14.3	28.7	34.6
	GroupNorm		✓		29.4	47.2	31.7	15.4	31.6	38.6
	Multi-level BatchNorm			✓	31.8	49.3	34.2	17.3	33.5	42.3
ResNet-50	BatchNorm	FP	✓		35.9	53.9	39.0	21.9	39.2	45.8
	GroupNorm		✓		38.7	57.6	41.4	22.8	42.3	50.2
	Multi-level BatchNorm				38.9	57.4	42.1	23.6	42.0	50.3
	BatchNorm	2-bit	✓	✓	30.3	49.6	32.6	17.3	32.9	39.0
	GroupNorm		✓		33.4	52.2	35.8	18.4	37.2	42.5
	Multi-level BatchNorm			✓	35.4	54.1	38.2	19.5	38.0	46.2

BN) indicates that normalization layers in the detection heads are shared across different pyramid levels, which is the default setting in current prevalent detectors. In practical, synchronized version of batch normalization (Sync-BN) is leveraged for all batch normalization layers. From the results, we have several observations. Firstly, the detector with the multi-level Sync-BN outperforms the one using the shared Sync-BN consistently by a large margin for both full-precision and quantized models with different backbones. Secondly, the detector with multi-level BN obtains comparable performance with the one with GN on full-precision models, and 2% or more AP improvement on the 2-bit quantization, which justifies the multi-level design can effectively solve the training instability during quantized fine-tuning. Thirdly, compared with the group normalization, our multi-level BN performs much better on quantized models while enabling computation to be carried out using integer-only arithmetic, which is more hardware friendly.

Effect of Quantization on Different Components. We further study the effect of quantizing different components in object detection models. The results are shown in Table 5. We observe that quantizing the backbone or the feature pyramid only leads to a small performance drop. Nevertheless, quantizing the detection heads and the fea-

Table 4 – Performance comparisons on the COCO validation set based on FCOS.

Backbone	Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-18	Full-precision	33.9	51.2	36.4	19.3	36.2	44.0
	Group-Net [51] (4 bases)	28.9	45.3	31.2	15.4	30.5	38.1
	AQD* (4-bit)	34.9	52.1	37.3	19.9	36.5	45.6
	AQD (4-bit)	34.1	51.5	36.5	18.3	36.2	45.1
	AQD* (3-bit)	34.3	51.4	36.7	19.4	36.0	45.1
	AQD (3-bit)	33.6	51.1	36.1	18.5	35.0	44.8
	AQD* (2-bit)	32.2	49.0	34.1	17.6	33.7	42.7
	AQD (2-bit)	31.8	49.3	34.2	17.3	33.5	42.3
ResNet-34	Full-precision	38.0	55.9	41.0	23.0	40.3	49.4
	Group-Net [51] (4 bases)	31.5	47.6	33.8	16.9	32.3	40.1
	AQD* (4-bit)	38.3	56.2	41.3	21.8	40.5	49.8
	AQD (4-bit)	37.6	55.5	40.6	20.8	40.0	49.3
	AQD* (3-bit)	37.8	55.8	40.7	22.2	40.4	49.8
	AQD (3-bit)	37.2	55.2	40.2	20.6	39.5	48.8
	AQD* (2-bit)	35.7	53.3	38.3	20.4	37.8	47.2
	AQD (2-bit)	35.0	53.4	37.5	18.9	37.3	47.1
ResNet-50	Full-precision	38.9	57.4	42.1	23.6	42.0	50.3
	Group-Net [51] (4 bases)	32.7	49.0	35.5	17.8	33.6	41.4
	AQD* (4-bit)	38.8	57.1	41.5	23.2	41.5	50.1
	AQD (4-bit)	38.0	56.5	40.7	21.9	41.0	49.2
	AQD* (3-bit)	38.5	57.1	41.9	23.3	41.6	49.7
	AQD (3-bit)	37.5	56.2	40.2	21.6	40.6	48.6
	AQD* (2-bit)	36.0	53.8	38.8	20.0	38.6	47.0
	AQD (2-bit)	35.4	54.1	38.2	19.5	38.0	46.2

Table 5 – Effect of quantization on different components. We quantize the FCOS detector to 2-bit and evaluate the performance on the COCO validation set based on AQD*.

Backbone	Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-18	Full-precision [41]	33.9	51.2	36.4	19.3	36.2	44.0
	Backbone	33.8	50.6	36.1	18.6	35.4	45.4
	Backbone + Feature Pyramid	33.2	49.9	35.4	18.9	34.6	44.0
	Backbone + Feature Pyramid + Heads	32.2	49.0	34.1	17.6	33.7	42.7

ture pyramid will cause significant performance degradation (*i.e.*, 1.7% in AP). These results show that the detection head modules other than the backbone are sensitive to quantization, which provides a direction to improve the performance of the quantized network.

AQD vs. AQD*. We further study the influence of fully-quantizing a model compared to the one with only convolutional layers quantized. From the results in Tables 2 and 4, we observe that AQD has a limited performance drop compared with AQD*. For example, the degradation is less than 0.3% on RetinaNet with ResNet-18 and ResNet-50 backbones of different quantization bitwidths. Besides, the performance gap on FCOS is relatively larger than that on RetinaNet. It can be attributed that FCOS is a fully convolutional pixel prediction framework relying heavily on the pixel-level feature quality, which might be more sensitive to the extreme low precision quantization.

5. Conclusion

In this paper, we have proposed an accurate quantized object detection framework with fully integer-arithmetic operations. Specifically, we have proposed efficient integer-only operations for BN layers and skip connections. Moreover, we have proposed multi-level BN to accurately calculate batch statistics for each pyramid level. To evaluate the performance of the proposed methods, we have applied our AQD on two classical one-stage detectors. Experimental results have justified that our quantized 3-bit detector achieves comparable performance compared with the full-precision counterpart. More importantly, our 4-bit detector can even outperform the full-precision counterpart in some cases, which is of great practical value.

Acknowledgements

MT was in part supported by Key-Area Research and Development Program of Guangdong Province 2018B010107001, and Program for Guangdong Introducing Innovative and Enterpreneurial Teams 2017ZT07X183, and Fundamental Research Funds for the Central Universities D2191240. CS and his employer received no financial support for the research, authorship, and/or publication of this article.

References

[1] Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 5145–5153, 2018. 2

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3

[3] Adrian Bulat and Yorgos Tzimiropoulos. Hierarchical binary cnns for landmark localization with limited resources. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018. 2

[4] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 5918–5926, 2017. 2

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2017. 1

[6] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018. 2

[7] Meghan Cowan, Thierry Moreau, Tianqi Chen, and Luis Ceze. Automating generation of low precision deep learning operators. *arXiv preprint arXiv:1810.11066*, 2018. 2

[8] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *Proc. Int. Conf. Learn. Repren.*, 2020. 1, 2, 3

[9] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. EIE: Efficient inference engine on compressed deep neural network. In *ACM/IEEE Annual Int. Symp. Computer Architecture*, pages 243–254, 2016. 1

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 2961–2969, 2017. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 770–778, 2016. 1, 5, 6

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proc. Eur. Conf. Comp. Vis.*, pages 630–645, 2016. 1

[13] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 1389–1397, 2017. 1

[14] M. Horowitz. 1.1 computing's energy problem (and what we can do about it). In *Proc. IEEE Int. Solid-State Circuits Conf. Digest of Tech. Papers (ISSCC)*, pages 10–14, 2014. 1

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1

[16] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 4107–4115, 2016. 2

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. Int. Conf. Mach. Learn.*, pages 448–456, 2015. 4

[18] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2704–2713, 2018. 1, 2

[19] Xiaotang Jiang, Huan Wang, Yiliu Chen, Ziqi Wu, Lichuan Wang, Bin Zou, Yafeng Yang, Zongyang Cui, Yu Cai, Tianhang Yu, Chengfei Lv, and Zhihua Wu. Mnn: A universal and efficient inference engine. In *Proc. The Conf. Machine Learning & Systems*, 2020. 2

[20] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2019. 1, 2

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 1097–1105, 2012. 1

[22] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proc. Eur. Conf. Comp. Vis.*, pages 734–750, 2018. 4

[23] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2019. 2, 5, 6, 7

[24] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *Proc. Int. Conf. Learn. Repren.*, 2020. 2

[25] Xiaocong Lian, Zhenyu Liu, Zhourui Song, Jiwu Dai, Wei Zhou, and Xiangyang Ji. High-performance fpga-based cnn accelerator with block-floating-point arithmetic. *IEEE T. Very Large Scale Integration (VLSI) Systems*, pages 1–12, 05 2019. 1

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2117–2125, 2017. 1, 4, 6

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 2980–2988, 2017. 1, 2, 4, 6

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proc. Eur. Conf. Comp. Vis.*, pages 740–755. Springer, 2014. 6

[29] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *Proc. Eur. Conf. Comp. Vis.*, 2020. 2

[30] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proc. Eur. Conf. Comp. Vis.*, pages 722–737, 2018. 2

[31] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3431–3440, 2015. 1

[32] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 5058–5066, 2017. 1

[33] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016. 2

[34] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 6181–6189, 2018. 4

[35] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *Proc. Int. Conf. Mach. Learn.*, pages 4092–4101, 2018. 1

[36] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proc. Eur. Conf. Comp. Vis.*, pages 525–542, 2016. 2

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 91–99, 2015. 1

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comp. Vis.*, 115(3):211–252, 2015. 6

[39] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4510–4520, 2018. 1

[40] Xiao Sun, Jungwook Choi, Chia-Yu Chen, Naigang Wang, Swagath Venkataramani, Vijayalakshmi Viji Srinivasan, Xiaodong Cui, Wei Zhang, and Kailash Gopalakrishnan. Hybrid 8-bit floating point (hfp8) training and inference for deep neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 4900–4909, 2019. 2

[41] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2019. 2, 4, 6, 8

[42] Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 7675–7684, 2018. 2

[43] Yi Wei, Xinyu Pan, Hongwei Qin, Wanli Ouyang, and Junjie Yan. Quantization mimic: Towards very tiny cnn for object detection. In *Proc. Eur. Conf. Comp. Vis.*, September 2018. 2

[44] Yuxin Wu and Kaiming He. Group normalization. In *Proc. Eur. Conf. Comp. Vis.*, pages 3–19, 2018. 4

[45] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 6

[46] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proc. Eur. Conf. Comp. Vis.*, 2018. 2

[47] Jianhao Zhang, Yingwei Pan, Ting Yao, He Zhao, and Tao Mei. dabnn: A super fast inference framework for binary neural networks on arm devices, 2019. 2

[48] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *Proc. Int. Conf. Learn. Repren.*, 2017. 2

[49] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 1, 2

[50] Bohan Zhuang, Lingqiao Liu, Mingkui Tan, Chunhua Shen, and Ian Reid. Training quantized neural networks with a full-precision auxiliary module. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2020. 1, 2, 6, 7

[51] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Peng Chen, Lingqiao Liu, and Ian Reid. Structured binary neural networks for image recognition. *arXiv preprint arXiv:1909.09934*, 2019. 2, 6, 8

[52] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 7920–7928, 2018. 1, 2

[53] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 881–892. 2018. 1