DupNet: Towards Very Tiny Quantized CNN with Improved Accuracy for Face Detection

Hongxing Gao, Wei Tao, Dongchao Wen, Junjie Liu
Canon Information Technology (Beijing) Co., LTD
{gaohongxing,taowei,wendongchao, liujunjie}@canon-ib.com.cn

Tse-Wei Chen, Kinya Osa, Masami Kato
Device Technology Development Headquarters, Canon Inc.
twchen@ieee.org

Abstract

*Deploying deep learning based face detectors on edge devices is a challenging task due to the limited computation resources. Even though binarizing the weights of a very tiny network gives impressive compactness on model size (*e.g. *240.9 KB for IFQ-Tinier-YOLO), it is not tiny enough to fit in the embedded devices with strict memory constraints. In this paper, we propose DupNet which consists of two parts. Firstly, we employ weights with duplicated channels for the weight-intensive layers to reduce the model size. Secondly, for the quantization-sensitive layers whose quantization causes notable accuracy drop, we duplicate its input feature maps. It allows us to use more weights channels for convolving more representative outputs. Based on that, we propose a very tiny face detector, DupNet-Tinier-YOLO, which is 6.5× times smaller on model size and 42.0% less complex on computation and meanwhile achieves 2.4% higher detection than IFQ-Tinier-YOLO. Comparing with the full precision Tiny-YOLO, our DupNet-Tinier-YOLO gives 1,694.2× and 389.9× times savings on model size and computation complexity respectively with only 4.0% drop on detection rate (0.880 vs. 0.920). Moreover, our DupNet-Tinier-YOLO is only 36.9 KB, which is the tiniest deep face detector to our best knowledge.*

1. Introduction

Deep neural networks have demonstrated impressive accuracy in many computer vision applications such as image classification, object detection and recognition, semantics segmentation, etc. However, their increasing computation cost leads to the requirement of high-end devices such as GPU for real-time inference. It has been a challenging task to deploy the deep network based face detector on the edge devices due to their limited resources (e.g. memory size and computation power). To deploy the deep models on the edge devices, lots of approaches have been proposed, such as network pruning [8, 14], efficient architecture design (e.g. MobileNet [9, 29]) and quantized networks [23, 1]. Especially, for the embedded devices, quantized networks are particularly attractive because of their impressive compression ratio (*e.g.* 32× times savings on model size) and easy conversion for fixed-point representation. For instance, IFQ-Net [6] designs a tiny fixed-point face detector (240.9 KB) through slimming, quantizing and fixed-point converting the layers of Tiny-YOLO network [31]. Even though the fixed-point converting is lossless, the accuracy drop caused by slimming and quantizing is still notable.

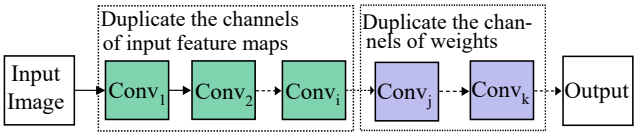

Figure 1. Strategies for DupNet. We duplicate the input feature maps of the quantization-sensitive layers which usually locate in the lower part of a network to improve the accuracy. Besides, to compress the overall model size, it employs duplicated weights for weight-intensive layers that usually locate in the upper part.

In this paper, to compress the model size and meanwhile improve the accuracy of a quantized network, we propose DupNet as shown in Figure 1. Firstly, to compress the model size, DupNet employs weights with duplicated channels for weight-intensive layers which usually are the upper layers. On the other hand, to improve the accuracy, DupNet duplicates the input feature maps and thus uses more weights channels for the quantization-sensitive layers (usually locate in the lower part of the network) whose quantization causes significant accuracy drop. In details, to further compress a quantized network, we force the weights

of weight-intensive layers to have identical channels. As shown in Figure 2, to generate such identical channels, we employ template weights $\mathbf{W}_t$ which has less channels than input feature maps and duplicate it to $\mathbf{W}_{dup}$ for proper convolution. During inference time, $\mathbf{W}_{dup}$ can be restored from the template weights $\mathbf{W}_t$. Consequently, model size savings can be easily achieved by only storing $\mathbf{W}_t$.

Another major issue of the quantized network is that the accuracy is usually downgraded by a large margin. For example, in XNOR-Net [23], binarizing both weights and feature maps of AlexNet [13] leads to 12% accuracy drop on ImageNet dataset [28]. In the case of face detection, we observe that the accuracy drop is mainly caused by the quantization of several specific layers, named as quantization-sensitive layers (see Section 4.2). Usually, they are the lower layers which only have small amount of output feature maps. Thus, quantizing them into extremely low-bits severely harms the representative power of the output features. To address the problem, one may simply employ more feature maps or quantize them into higher bits, both of which would increase the memory usage on feature maps.

In this paper, we propose to further duplicate the input feature maps of the quantization-sensitive layers to improve its accuracy (Figure 2). It is true that simply duplicating the feature maps does not introduce extra information. However, it allows us to use weights with more channels (not identical) for convolving more representative outputs. The advantage of our method is that it does not require extra memory on feature maps which is a critical issue for the embedded devices. Nevertheless, it does increase the memory usage on weights. However, as will be demonstrated in Section 4.2, we experimentally found out that the quantization-sensitive layers are usually the lower layers of a network which only have small amount of weights. Consequently, such memory increase does not affect the network much.

In summary, we propose DupNet which employs duplicated weights for the weight-intensive layers and duplicates the input feature maps for the quantization-sensitive layers of a quantized network. The benefits of our proposal are two-folds: 1) it reduces the model size of a quantized network by duplicated weights for weight-intensive layers; 2) it increases the accuracy through duplicating the input feature maps of its quantization-sensitive layers. Based on the DupNet, we design a very tiny quantized CNN with impressive improvement on accuracy for face detection. The model size of our network is only 36.9 KB which is the tiniest deep learning based face detector to our best knowledge.

2. Related Work

2.1. Face Detection

Two main approaches, namely one-stage and two-stage methods, have been successfully inherited from object detection domain for face detection. Two-stage methods follow a common two steps pipeline: 1) generates a set of region proposals with their local features; 2) pass them to a network for classifying detected objects and regressing their bounding boxes. For example, Faster-RCNN [27] proposes an efficient Region Proposal Network (RPN) to generate region proposals and then use Fast-RCNN network to refine the proposals. To improve the speed of Faster R-CNN, RFCN [4] proposes to share RPN network and Fast-RCNN network. In order to further improve the speed, Li *et al*. [17] proposes Light Head RCNN, which employs light weight head network to reduce the computation complexity. To speedup the R-FCN network for detecting 3000 object classes, Singh *et al*. [30] propose to only employ position-sensitive feature maps for several predefined super-classes.

On the other hand, one-stage approaches usually employ a single network to classify and regress the objects [18, 25, 26]and thus usually can run faster. For example, YOLO [25] predicts 2 bounding boxes in each of the 7×7 grids for VOC object detection [5]. Furthermore, YOLOv2 [26] employs fully convolution network that results in $m \times n$ grids (m, n are the width and height of the output feature) and uses predefined anchors to better predict the bounding boxes of the objects. In [16], Li *et al*. propose a backbone network to improve the accuracy by maintaining high resolution for feature maps and reduce the computation complexity by decreasing the width of upper layers.

In spite of the enormous progresses for reducing the complexity of two-stage methods, such region proposal based frameworks may be expensive for embedded devices because they usually need to store the features from previous layers. Therefore, following [6], we employ the widely used one-stage pipeline YOLOv2 for our face detector.

2.2. Deep Network Compression

To reduce the computation cost of the deep models, many approaches have been studied. One way is to design novel efficient architectures. For example, by replacing a standard convolution layer by the combination of a depth-wise and a point-wise (1×1) convolution layer, MobileNets [9] reduces the weights and computation by $8\times\sim9\times$ times. Similarly, LBCNN [12] employs predefined binary patterns for the depthwise convolution and shares those patterns over multiple layers for further compression.

Another direction for designing a compact model is to compress the network through pruning, quantization, etc. Pruning methods eliminate the less important connections and fine-tune the pruned network to narrow down the accuracy drop. For example, in [32], Wei *et al*. reduce the input and output channels of each layer of VGG by 32 times and design a very small detector whose size is only 132KB. In contrast, quantization approaches aim to quantize the float data of a network into low-bits data. For example, XNOR-

Net [23] and HWGQ-Net [1] achieves 32× times savings on model size via binarizing (1-bit) the network weights. In addition to the quantized weights, further quantizing feature maps into low-bits data can reduce the feature maps memory usage and meanwhile increase the inference speed. For example, XNOR-Net which quantizes both weights and feature maps into 1-bit is theoretically 64× times faster than its full precision counterpart. Furthermore, for embedded devices such as FPGA and ASIC, quantization network is particularly attractive because it leads to higher throughput and lower power consumption through converting the network into a fixed-point one.

One interesting topic is about further exploring the redundancy and compressing the quantized network. For example, in [22], various networks (VGG16, MobileNet) are firstly quantized to 8-bit data and then further pruned by 24%. Similarly, Li and Ren [15] explores the redundancy of a Binarized Neural Network (BNN) and further compresses the model size by 3.9× times through bit-level data pruning.

Different with methods that explore the redundancy through carefully tuned strategies, we propose to simply employ duplicated weights which contain lots of identical channels for the weight-intensive layers. Since the duplicated weights can be easily restored from the template weights which contain all the non-identical channels, it is sufficient to only store the template weights in the memory during inference time.

2.3. Accuracy Improvement for Quantized Network

Even though quantizing the network into low-bits data leads to promising reduction on computation cost, accuracy drop is usually observed. As demonstrated in [10, 19], quantizing the network data into 8-bits only leads to minor accuracy drop on ImageNet classification task [28]. Nevertheless, quantizing the network into lower bits usually results in notable accuracy degradation. For example, XNOR-Net [23] which quantizes both its weights and feature maps into 1-bit and thus observes a 12.6% accuracy drop (56.8% vs. 44.2%). Based on that, HWGQ-Net [1] gains 8.2% accuracy back through using 2-bits on its feature maps (52.4% on ImageNet). Additionally, for object detection tasks, 3%∼5% drop is observed in [21].

To improve the accuracy of quantized networks, lots of efforts have been done on better strategies for training the networks. INQ [34] proposes to incrementally quantize the weights and achieves more accurate quantized networks through iterative fine-tuning. Similarly, in [35], the weights and activations are firstly quantized to 16-bits, then to 4-bits and at the end to 2-bits. PACT [3] optimizes the clipping thresholds for better quantization on feature maps. Besides, knowledge distillation technology additionally uses the knowledge from teacher network to guide the training process of student network [32].

To narrow down the accuracy drop caused by the network quantization, we propose to duplicate the feature maps of its quantization-sensitive layers which allows us to use weights with more channels for convolving more representative features. The advantage of our method is that it gives significant accuracy improvement without increasing the feature maps memory usage.

3. Our Approach: DupNet

To further compress the model size and improve the accuracy of a quantized network for face detection, we propose to employ weights with duplicated channels in the weight-intensive layers and duplicate the input feature maps of its quantization-sensitive layers.

3.1. Duplicated Weights for Model Compression

As discussed in [15], even though the network data is quantized into very low-bits data, redundancy still exists. In this section, we will illustrate our method which employs template weights with less channels and thus less redundancy. During convolution process, we duplicate the template weights to get required channels to convolve with the input feature maps.

We assume the quantized network only employs $a2w1$ convolution which means that the input feature maps and weights are quantized to 2-bits and 1-bit respectively. We represent its weights and feature maps as $\mathbf{W} \in \{-1, +1\}^{c\times 3\times 3}$ and $\mathbf{X} \in \{0, 1, 2, 3\}^{c\times h\times w}$, where c, h, w are the number of channels, the width and height of input feature maps, 3×3 is the kernel size of the convolution[1]. To compress the model size, we define a weights template $\mathbf{W}_t \in \mathbb{R}^{c'\times 3\times 3}$ which has less channels ($c' < c$). However, it can not be used to convolve with $\mathbf{X}$ since they have different number of channels. To solve such problem, we duplicate the channels of template weights into required number and obtain duplicated weights $\mathbf{W}_{dup} \in \mathbb{R}^{c\times 3\times 3}$.

During training process, it is straightforward to compute the gradient of duplicated weights $\frac{\partial L}{\partial \mathbf{W}_{dup}}$ by employing standard convolution, where L represents the loss of the network for given training samples. As shown in Figure 2, to compute the gradient of template weights $\frac{\partial L}{\partial \mathbf{W}_t}$, we average the corresponding channels of the $\frac{\partial L}{\partial \mathbf{W}_{dup}}$. Specifically, we assume that $c = 512$ and $c' = 128$ for 4× times compression, and the 0th, 128th, 256th and 384th channels of $\mathbf{W}_{dup}$ are duplicated from the 0th channel of $\mathbf{W}_t$. In order to compute the 0th channel of the gradient $\frac{\partial L}{\partial \mathbf{W}_t}$, we element-wisely average the 0th, 128th, 256th and 384th channels of $\frac{\partial L}{\partial \mathbf{W}_{dup}}$. Such gradient averaging process is repeated for computing the 1∼127th channels of $\frac{\partial L}{\partial \mathbf{W}_t}$. At the end, the

[1]Even though we use 3×3 for explanation, our method is also able to compress the convolution with other kernel size (e.g. 1×1 and 5×5).

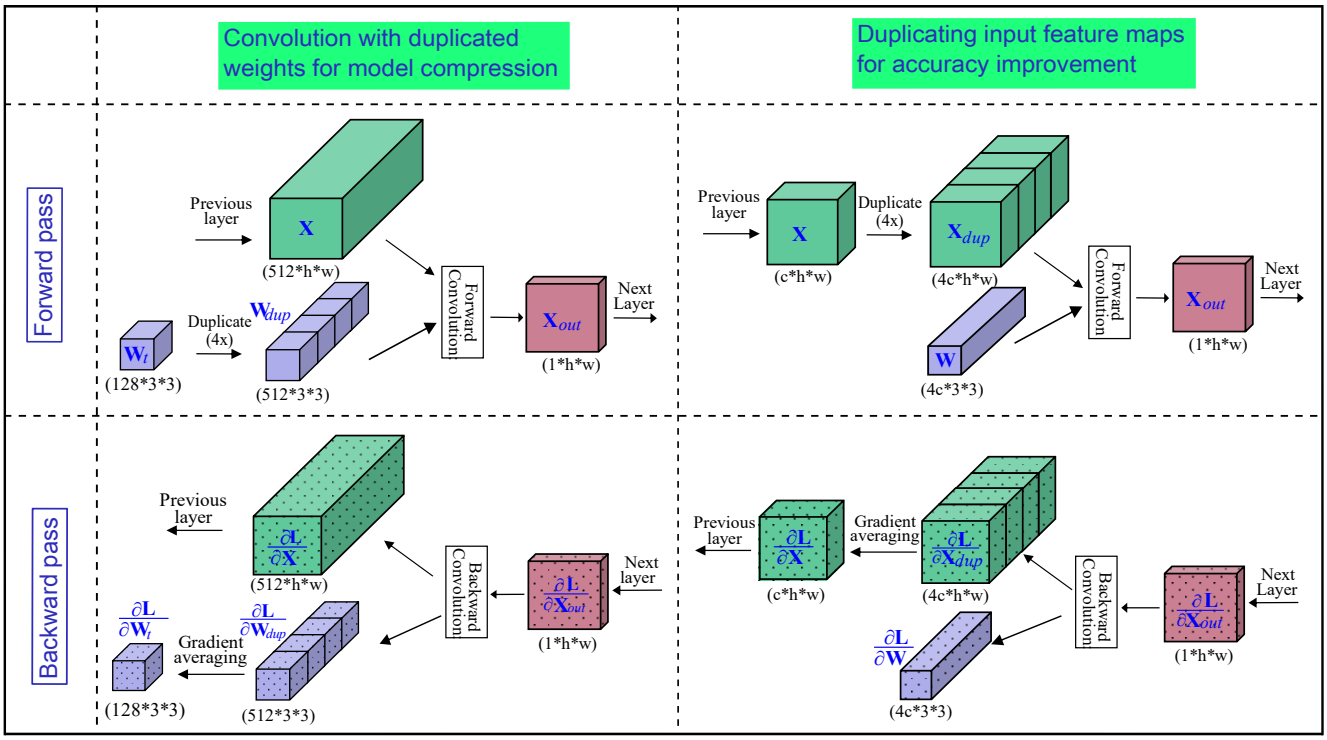

Figure 2. Illustration of the forward and backward pass for both duplicated weights and feature maps duplication.

template weights $\mathbf{W}_t$ can be learned by iteratively updating it with its gradients $\frac{\partial L}{\partial \mathbf{W}_t}$ using SGD optimization.

Since the duplicated weights $\mathbf{W}_{dup}$ can be easily restored from the templates weights $\mathbf{W}_t$, it is only necessary to store $\mathbf{W}_t$. Thus, c/c' times model size reduction can be achieved. Nevertheless, our compression method may harm the accuracy because the duplicated weights contains less non-identical channels. Considering that the model size is usually dominated by the weight-intensive layers, we only apply our compression method on these layers to prevent significant accuracy drop.

Furthermore, given the fact that many channels of the duplicated weights are identical, we can reduce the computation complexity as follows. We split duplicated weights $\mathbf{W}_{dup}$ into $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{W}_4$ while each of them is identical with the template weights $\mathbf{W}_t$. Similarly, the feature maps $\mathbf{X}$ can also be accordingly split into $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4$ which are non-identical. We use $\otimes$ and $Concat(,)$ to represent convolution operation and a function that concatenate its members along channel axis respectively. Then, $\mathbf{W}_{dup} \otimes \mathbf{X} = Concat(\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{W}_4) \otimes Concat(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4) = \mathbf{W}_1 \otimes \mathbf{X}_1 + \mathbf{W}_2 \otimes \mathbf{X}_2 + \mathbf{W}_3 \otimes \mathbf{X}_3 + \mathbf{W}_4 \otimes \mathbf{X}_4 = \mathbf{W}_t \otimes (\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3 + \mathbf{X}_4)$. Consequently, the convolution $\mathbf{W}_{dup} \otimes \mathbf{X}$ can be alternatively computed by $\mathbf{W}_t \otimes \mathbf{X}_{sum}$, where $\mathbf{X}_{sum} = \mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3 + \mathbf{X}_4$. The overall computation complexity of $\mathbf{W}_t \otimes \mathbf{X}_{sum}$ and $\mathbf{X}_{sum}$ is much smaller than $\mathbf{W}_{dup} \otimes \mathbf{X}$.

3.2. Duplicate Feature Maps to Improve Accuracy

The quantized networks quantize their full precision data into low-bits data thus usually leads to notable accuracy drop. In the following, we further improve the degraded accuracy for a very tiny quantized face detector.

As discussed in Section 1, the accuracy degradation of quantized face detector is mainly caused by the weak representation power of the quantized output feature maps of its quantization-sensitive layers. To enhance the representative power of their output features, one straightforward way is to simply make these layers wider (more input feature maps). However, it significantly increases the memory usage on feature maps. Besides, observing that these layers usually locate in the lower part of a network, such memory increase may cause critical issue because their feature maps hold high resolution ($h \times w$). In contrast, the number of channels (c) is usually small and thus their weights ($c \times 3 \times 3$) is small too. Consequently, we propose to duplicate the input feature maps and employ more weights channels for better output features. As shown in Figure 2, the input feature maps are duplicated $4\times$ times and thus the weights size is also increased to $4c \times 3 \times 3$. For the backward pass during training time, to obtain the gradients $\frac{\partial L}{\partial \mathbf{X}}$, we firstly compute the gradients of duplicated feature maps $\frac{\partial L}{\partial \mathbf{X}_{dup}}$, and then average every 4 of its corresponding channels that are identical in $\mathbf{X}_{dup}$. At the end, the gradients $\frac{\partial L}{\partial \mathbf{X}}$ are propagated back to its previous layers.

Comparing with the strategy that simply uses more input feature maps, our method does not require extra memory for feature maps. Thanks to the increased channels of input feature maps, we can employ more channels of weights (c vs. $4c$). Consequently, the input feature maps are convolved with more patterns and thus leads to more representative power on the resulted features. Even though our method increases the weights size, such cost increase has limited influence on the overall model cost because the weights size of these layers are usually very small (see Table 1). Similar with the theory that is explained in Section 3.1, one also can achieve speedup by replacing $\mathbf{W} \otimes \mathbf{X}_{dup}$ with $\mathbf{W}_{sum} \otimes \mathbf{X}$, where $\mathbf{W}_{sum}$ can be obtained by summing the corresponding channels of $\mathbf{W}$.

4. Experimental Results

To design a very tiny CNN for face detection, we borrow the compression ideas from IFQ-Tinier-YOLO [6] which compresses Tiny-YOLO network by 260× times through halving the filter numbers of all convolution layers, replacing one 3×3 layer which contains massive parameters by 1×1 kernels and binarizing the weights in all layers. Moreover, we further halve their filter number and apply the proposed duplicated weights for its weight-intensive layers (Conv6∼Conv8) and achieve 6.7× times further savings on model size. Besides, we will demonstrate that duplicating the input feature maps of its quantization-sensitive layers can significantly improve the accuracy.

We employ WiderFace [33] training images to train our models using Darknet framework [24]. For fair comparison, all the models are trained with the same strategies which are: 1) training the models by 100k iterations with SGD optimization method; 2) the learning rate is initially set to 0.01 and downscaled by a factor of 0.1 at the $30k$-th, $60k$-th, $80k$-th and $90k$-th iteration respectively; 3) all the models are trained from scratch. Furthermore, we use FDDB [11] benchmark which contains 5,171 faces within 2,845 test images to evaluate the accuracy of our face detectors. Inheriting from [6], we use the detection rate when 284 false positive faces are reached (averagely allowing 1 false positive in every 10 images) as the evaluation metric.

4.1. Model Compression

To further compress model size of IFQ-Tinier-YOLO, we analyze the weights size for each of its layer. Meanwhile, to measure the computation complexity, we borrow the term #FLOPs[2] (Floating-point operations) which is generally used for full precision networks [20]. Nevertheless, it is worthy to point out that our network can be lossless converted to fixed-point network and thus does not require any floating-point operation.

Table 1. IFQ-Tinier-YOLO inference costs in terms of weights size and #FLOPs (million) for measuring computation complexity.

	Kernel size (W)	Feature size (X)	#FLOPs (million)	Weights size (KB)
Conv1	$8 \times 3 \times 3$	608×608	10.0	< 0.1
Conv2	$16 \times 3 \times 3$	304×304	3.3	0.1
Conv3	$32 \times 3 \times 3$	152×152	3.3	0.6
Conv4	$64 \times 3 \times 3$	76×76	3.3	2.3
Conv5	$128 \times 3 \times 3$	38×38	3.3	9
Conv6	$256 \times 3 \times 3$	38×38	13.3	36
Conv7	$512 \times 3 \times 3$	38×38	53.2	144
Conv8	$512 \times 1 \times 1$	38×38	11.8	32
Conv9	$30 \times 3 \times 3$	38×38	6.2	8.44
Overall			107.9	240.9

As shown in Table 1, the weight-intensive layers of IFQ-Tinier-YOLO model are Conv6∼Conv8 layers. Consequently, to further compress the model size, we apply two techniques for these layers: halving the filters number and employing duplicated weights. Regarding to the duplicated weights, we casted experiments that employ 2× or 4× or 8× times duplication to figure out the optimal trade-off between high compression ratio and low detection rate.

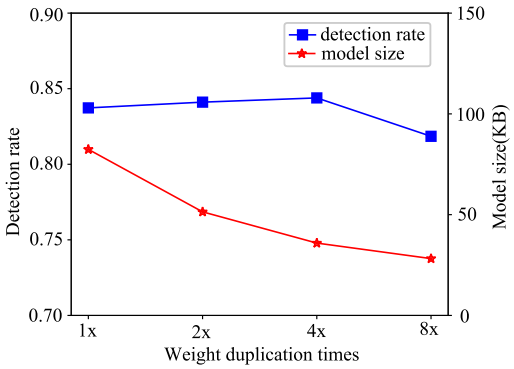

Figure 3. Comparison of the detection rate for the models with various compression ratio on Conv6∼Conv8 layers.

We first halve the filter number of the weight-intensive layers and thus reduce its model size from 240.9 KB to 82.4 KB (marked as "1×" in Figure 3). Meanwhile, it achieves 0.837 on detection rate which is very close to IFQ-Tinier-YOLO (0.84 [6]). Additionally, as shown in Figure 3, employing 2× or 4× times duplicated weights gives further reduction on model size without detection rate drop. More specifically, with the help of halved filter number and 4× times duplicated weights, we reduce the model size of IFQ-Tinier-YOLO from 240.9 KB to 35.9 KB indicating a 6.7× times reduction. Furthermore, when compressing the

[2]As stated in [23], for the 64-bit based computing devices, 64 Multi-Adds of the $a1w1$ convolution are equivalent to 1 FLOP. Similarly, we assume that 32 Multi-Adds of the $a2w1$ convolution and 8 Multi-Adds of the $a8w1$ (Conv1) equal to 1 FLOP respectively.

Conv6∼Conv8 by 8× times, the accuracy only decreases by 2.1% while the compression ratio increases to 8.5× times.

The reason that our compression method does not give notable accuracy drop is that the redundant connections exist in those three layers. However, one may argue that further reducing the number of their filters also can reduce the model size. Consequently, we compare such method (marked as "Filter slimming") with our method in Figure 4. For fair comparison, we reduce the filter numbers of Conv6∼Conv8 to make them have similar model size with our duplicated weights models. For example, to compare with our model with 4× times compression on all Conv6∼Conv8 layers, we instead halve their filter numbers resulting in a 2×, 4× and 2× times compression for these three layers respectively. As demonstrated in Figure 4, for different compression ratios, our weights duplication based method generally outperforms the filter slimming method.

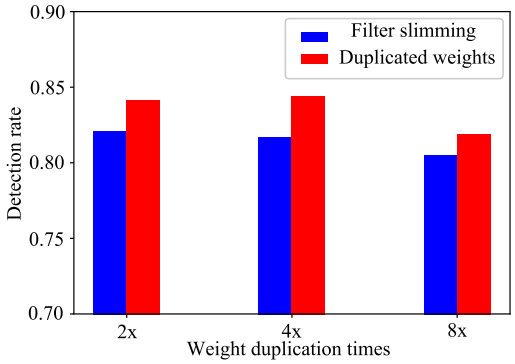

Figure 4. Performance of our duplicated weights based method and the filter slimming method for model compression.

In the above, we have demonstrated that our duplicated weights based compression is very effective for the quantized network whose precision is $a2w1$. To demonstrate the generalization ability of our method, we further test it on the networks that are quantized into different precisions. As shown in Table 2, our method with 4× times weights duplication also gives no accuracy drop for the $a2w2$, $a4w4$ and $a8w8$ networks. When further compressing them by 8× times, slight degradation is observed. One interesting observation is that the higher precision the network is, the less accuracy drop is caused. For example, with 8× times compression, the detection rate drop for $a2w2$ network is 3.8% while it is only 2.0% for $a4w4$ and 0.6% for $a8w8$. We think it is because that the more accurate the network is, the more redundancy usually exists in its connections.

4.2. Accuracy Improvement

The accuracy of quantized networks usually is notably lower than their full precision counterparts. For example, the quantized network based face detector, IFQ-Tinier-YOLO leads to ∼6% drop on detection rate [6]. On the other hand, quantizing different layers leads to widely-varied performance loss [36]. To improve the accuracy, we first locate the quantization-sensitive layers of a quantized face detector through layer-wise quantization strategy.

Table 2. Performance of our compression method on the face detectors with various quantization precision.

Weights duplication	Network precision			
	$a2w1$	$a2w2$	$a4w4$	$a8w8$
1×	0.837	0.866	0.892	0.906
2×	0.841	0.862	0.888	0.907
4×	0.844	0.865	0.890	0.900
8×	0.819	0.828	0.872	0.892

Table 3. Layer-wise quantization to locate the source of accuracy drop.

Quantized Conv. layers				#FLOPs (million)	Model size(KB)	Detection rate
1st	2nd-3rd	4th-8th	9th			
				1,338.9	2,637.3	0.902
		✓		422.2	366.6	0.880
	✓	✓		215.9	344.8	0.858
✓	✓	✓		146.0	344.0	0.845
✓	✓	✓	✓	49.3	82.4	0.837

As shown in Table 3, we firstly quantize Conv4∼Conv8 convolution layers of a full precision counterpart of IFQ-Tinier-YOLO network but with halved filter number in Conv6∼Conv8. In this subsection, to demonstrate the accuracy improvement effect of duplicating the input feature maps, the duplicated weights based compression is not applied. As shown in Table 3, quantizing Conv4∼Conv8 leads to 3.2× and 7.2× times reduction on MFLOPs (million of FLOPs) and model size respectively while the detection rate only drops by 2.2%. Nevertheless, progressively quantizing the Conv3∼Conv2 and then Conv1 causes 2.2%, 1.3% accuracy drop respectively but gives much less reductions on inference cost. Thus, we define the Conv1∼Conv3 as quantization-sensitive layers of the network. We think the reason is that they only contain limited number of feature maps. Consequently, quantizing them severely damages the representative power of their output features. At the end, quantizing Conv9, resulting in a fully quantized Tinier-YOLO model, further gives remarkable savings on computation cost while the accuracy is only decreased by 0.8%.

To improve the accuracy of the fully quantized Tinier-YOLO, we firstly duplicate the input feature maps of Conv2 and Conv3 by 4× and 2× times respectively. As shown in Table 4, it gives 3.0% increase on detection rate while the model size and computation complexity are only increased by 1.2% and 27.0% respectively. Furthermore, additionally duplicating the feature maps of Conv1 by 4× times gives 0.5% increase on detection rate while the model size only increases 0.1KB. However, its computation complexity is

Table 4. Illustration of performance improvement for progressively duplicating the input feature maps of the Conv2-3, Conv1 and Conv9 of fully quantized Tinier-YOLO face detector.

Feature maps duplication?				#FLOPs (million)	Model size(KB)	Detection rate
Conv1	Conv2	Conv3	Conv9			
				49.3	82.4	0.837
	✓	✓		62.6	83.4	0.867
✓	✓	✓		92.6	83.5	0.872
✓	✓	✓	✓	95.7	91.9	**0.890**

increased from 62.6 MFLOPs to 92.6 MFLOPs (47.9% increase). At the end, we further duplicate the feature maps of Conv9 by 2x and achieve 1.8% improvement on detection rate at the price of 3.3% and 10.1% increase on #FLOPs and model size respectively.

Table 5. Comparison between our method and the quantization precision increasing method on improving the detection rate.

Quantization precision			Detection rate
Conv1	Conv2	Conv3	
$a8w1$	$a2w1(4\times)$	$a2w1(2\times)$	**0.867**
$a8w1$	$a2w3$	$a2w2$	0.850
$a8w1(4\times)$	$a2w1(4\times)$	$a2w1(2\times)$	**0.872**
$a8w3$	$a2w3$	$a2w2$	0.860

On the other hand, employing more bits for the weights of quantization-sensitive layers can also improve the accuracy. For fair comparison, in the case of $4\times$ times duplication (*e.g.* Conv2), we use 3-bits on weights ("$a2w3$") to compare it with our method ("$a2w1(4\times)$") which can be computed by $W_{sum} \otimes X$ where $W_{sum} \in \{-4, -2, 0, 2, 4\}$[3] that can be represented using 3-bits. As shown in Table 5, our methods generally gives more than 1.0% improvement on detection rate. Furthermore, our method is more attractive for hardware design in three aspects: 1) it use less information (only 5 values vs. 8 values) which makes the coding-based further compression easier (*e.g.* Huffman coding [7] and RLC [2]); 2) lots of its weights are 0 thus the corresponding computation can be optimized; 3) our model can be computed only using $a2w1$ convolution[4] which can make the hardware design simpler.

4.3. Face Detectors Comparison

As demonstrated in the previous experiments, employing duplicated weights gives remarkable compression without obvious accuracy drop. On the other hand, duplicating the feature maps for the quantization-sensitive layers improves the detection rate by a large margin. In this section, we combine these two technique to design DupNet-Tinier-YOLO which is a very tiny quantized face detector with improved accuracy. In details, we employ $4\times$ times compression for weight-intensive layers (Conv6∼Conv8) and duplicate the input feature maps of Conv2∼Conv3. We initially choose not to duplicate the input feature maps of Conv1 in DupNet-Tinier-YOLO to avoid notable increase on #FLOPs. Regarding to the model size, $4\times$ times weights compression reduces the model size from 82.4KB to 35.9 KB and duplicating the input feature maps increase it to 36.9 KB.

Table 6. Comparison of the face detectors in terms of the computation complexity (#FLOPs), model size and detection rate.

Models	#FLOPs (million)	Model size(KB)	Detection rate
Tiny-YOLO	24,407	62,516	0.920
Tinier-YOLO	3,213	7,707	0.902
IFQ-Tinier-YOLO [6]	107.9	240.9	0.835
DupNet	62.6	36.9	0.859
DupNet+PACT	62.6	36.9	0.880
DupNet-L	95.7	45.4	0.884
DupNet-L+PACT	95.7	45.4	0.906

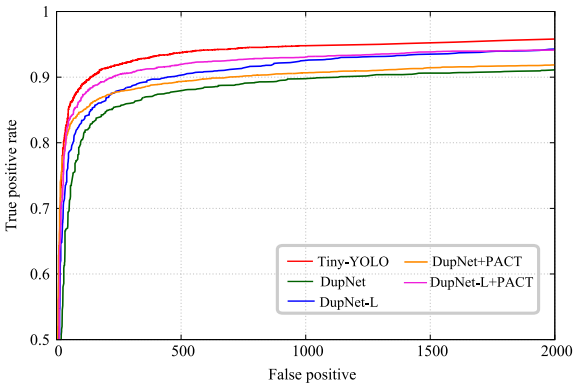

Figure 5. Comparison on the performance of the face detectors in terms of ROC curves of FDDB dataset [11].

As shown in Table 6, comparing with the IFQ-Tinier-YOLO [6], our DupNet-Tinier-YOLO (represented as "DupNet") gives $6.5\times$ times savings on model size and 42.0% less MFLOPs. Meanwhile, it also gives 2.4% improvement on detection rate. To further improve the detection rate with acceptable cost increase, we design DupNet-Tinier-YOLO-L (marked as "DupNet-L") which additionally duplicates the input feature maps fo Conv1 and Conv9 by $4\times$ and $2\times$ respectively. As shown in Table 6, DupNet-Tinier-YOLO-L further gives 2.5% higher detection rate. Nevertheless, the model size and #FLOPs are increased to 45.4 KB and 95.7 MFLOPs respectively, both of which are still smaller than IFQ-Tinier-YOLO.

Furthermore, we employ PACT [3] to train optimal clipping thresholds for feature maps quantization to im-

[3]Each elements of W_{sum} is the summation of four binary elements (either -1 or +1) from four corresponding channels (see Section 3.2).

[4]The $a8w1$ convolution (Conv1) can be computed by the accumulation of four $a2w1$ convolutions.

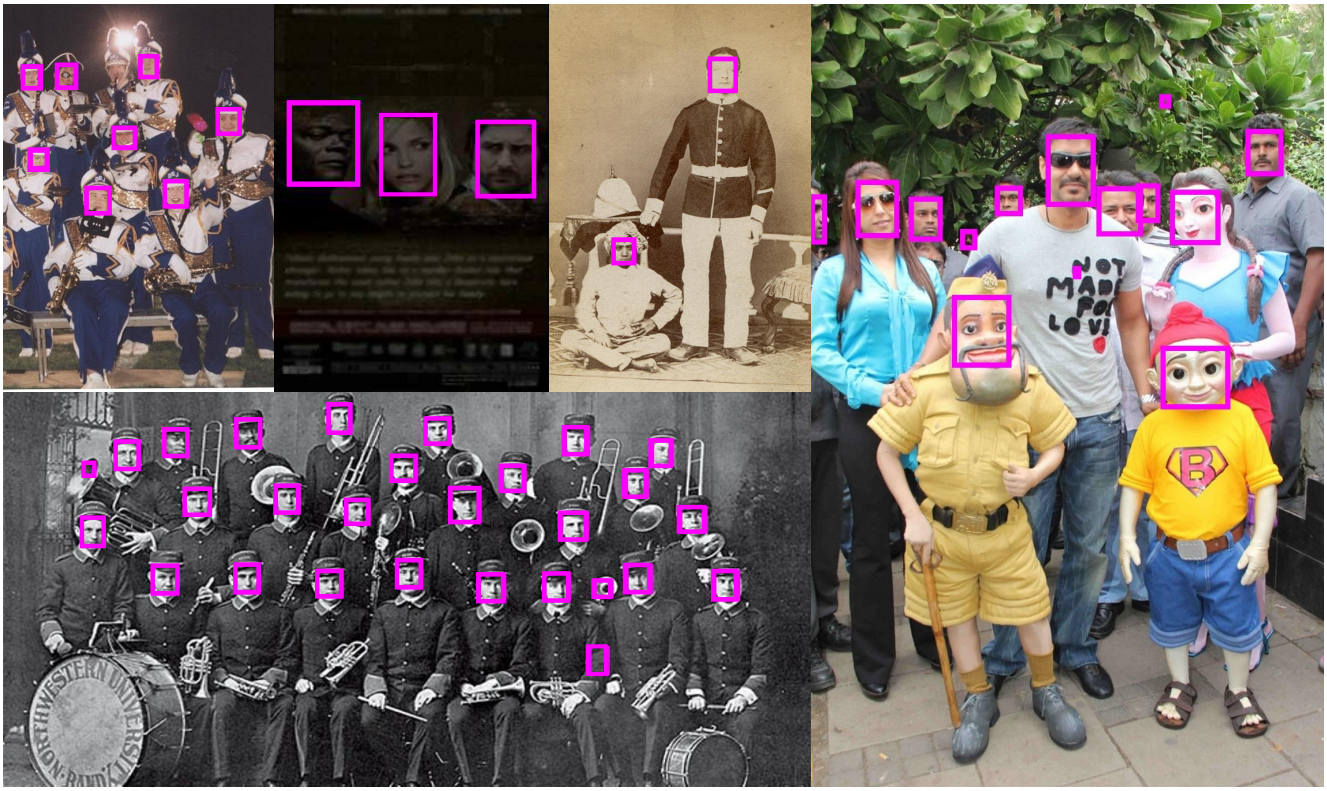

Figure 6. Qualitative results of the proposed DupNet-Tinier-YOLO-L face detector on Wider Face dataset [33].

prove the accuracy. As illustrated in Table 6, PACT algorithm improves the DupNet-Tinier-YOLO and DupNet-Tinier-YOLO-L by 2.1% and 2.2% respectively. Comparing with Tiny-YOLO network, our DupNet-Tinier-YOLO achieves 389.9× and 1694.2× times reduction on #FLOPs and model size respectively while the detection rate is only decreased by 4.0%. On the other hand, the accuracy of DupNet-Tinier-YOLO-L is only decreased by 1.4% while the inference cost reduction is kept impressive. Besides, we compare their accuracy in terms of ROC curves in Figure 5.

To demonstrate the performance of our detector on more challenging faces, we also test DupNet-Tinier-YOLO-L on WiderFace testing dataset [33]. As shown in Figure 6, our model also gives excellent detection quality in various challenging scenarios such as tiny size, low-illumination, severe occlusion and degraded coloring, etc.

In summary, we proposed DupNet-Tinier-YOLO face detector, which is quantized, very tiny and accurate. By employing duplicated weights for the weight-intensive layers, we reduced the model size and #FLOPs of IFQ-Tinier-YOLO by 6.5× times and 42.0% respectively. Meanwhile, we increased its detection rate by 4.5% by using the proposed DupNet and the PACT [3] technique. Moreover, we demonstrated that our DupNet can be flexibly adjusted for different inference cost (*e.g.* DupNet-Tinier-YOLO-L has higher cost and is more accurate).

5. Conclusions

In this paper, we proposed DupNet which employs duplicated weights for the weight-intensive layers of a quantized CNN to compress its model size. Furthermore, we observe that the degraded accuracy of the quantized CNN is mainly caused by quantization-sensitive layers which have poor representative power on their quantized output feature maps. Hence, DupNet also duplicates the input feature maps of these layers and employ more weights channels to improve their output features. Through the experiments on FDDB dataset, we demonstrated that our DupNet-Tinier-YOLO face detector can significantly compress the model size and meanwhile impressively improve the detection rate. Moreover, our DupNet-Tinier-YOLO face detector can be lossless converted into fixed-point network [6] and thus can be easily implemented on embedded devices.

Additionally, our DupNet can be combined with other algorithms that are proposed to improve the performance of compressed networks such as knowledge distillation. Moreover, despite we only test our method on face detection, it is also applicable for other tasks such as object detection or even face recognition or semantic segmentation, etc.

References

[1] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *IEEE Conference on Computer Vision and Pattern Recognition,CVPR*, pages 5406–5414, 2017. 1, 3

[2] Yu-Hsin Chen, Tusha Krishna, Joel S. Emer, and Vivienne Sze. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138, 2017. 7

[3] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: Parameterized clipping activation for quantized neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018. 3, 7, 8

[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object detection via region-based fully convolutional networks. In *Annual Conference on Neural Information Processing Systems, NIPS*, 2016. 2

[5] Mark EveringHam, S.M. Ali Eslami, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The pasacal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 2

[6] Hongxing Gao, Wei Tao, Dongchao Wen, TseWei Chen, Kinya Osa, and Masami Kato. IFQ-Net: Integrated fixed-point quantization networks for embedded vision. In *IEEE Embedded Vision Workshop, CVPRW*, 2018. 1, 2, 5, 6, 7, 8

[7] Song Han, Huizi Mao, and William J. Dally. Deep Compression: compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations, ICLR*, 2016. 7

[8] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Annual Conference on Neural Information Processing Systems, NIPS*, 2015. 1

[9] 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. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017. 1, 2

[10] 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 *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018. 3

[11] Vidit Jain and Erik Learned-Miller. FDDB: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010. 5, 7

[12] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. Local binary convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017. 2

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffery Hinton. Imagenet classification with deep convolutional neural networks. In *Annual Conference on Neural Information Processing Systems, NIPS*, 2012. 2

[14] Hao Li, Asim Kadav, Igor Durdannovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations, ICLR*, 2017. 1

[15] Yixing Li and Fengbo Ren. Building a compact binary neural network through bit-level sensitivity and data. *ArXiv e-prints*, 2018. 3

[16] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. DetNet: a backbone network for object detection. In *IEEE European Conference on Computer Vision, ECCV*, 2018. 2

[17] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Light-Head R-CNN: In defense of two-stage object detector. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018. 2

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *IEEE European Conference on Computer Vision, ECCV*, pages 21–37, 2016. 2

[19] Szymon Migacz. 8-bit inference with TensorRT. http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf, May 2017. 3

[20] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations, ICLR*, 2017. 5

[21] Mi Sun Park, Xiaofan Xu, and Cormac Brick. SQuantizer: Simultaneous learning for both sparse and low-precision neural networks. *ArXiv e-prints*, 2018. 3

[22] Qing Qin, Jie Ren, Jialong Yu, Ling Gao, Hai Wang, Jie Zheng, Jianbin Fang, and Zheng Wang. To compress, or not to compress: Characterizing deep learning model compression for embedded inference. *ArXiv e-prints*, 2018. 3

[23] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: imagenet classification using binary convolutional neural networks. In *IEEE European Conference on Computer Vision, ECCV*, pages 525–542, 2016. 1, 2, 3, 5

[24] Joseph Redmon. Darknet: Open source neural networks in C. http://pjreddie.com/darknet/, 2013–2016. 5

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE International Conference on Computer Vision and Pattern Recognition, CVPR*, pages 779–788, 2016. 2

[26] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *IEEE International Conference on Computer Vision and Pattern Recognition, CVPR*, pages 6517–6525, 2017. 2

[27] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. 2

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2, 3

[29] Mark Sandler, Andrew Howard, Menglong Zhu andAmdrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residual and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018. 1

[30] Bharat Singh, Hengduo Li, Abhishek Sharma, and Larry S. Davis. R-FCN-3000: decoupling detection and classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018. 2

[31] Yap June Wai, Zulkalnain bin Mohd Yussof, Sani Irwan bin Salim, and Lim Kim Chuan. Fixed point implementation of Tiny-YOLO-v2 using OpenCL on FPGA. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 9(10):506–512, 2018. 1

[32] Yi Wei, Xinyu Pan, Hongwei Qin, Wanli Ouyang, and Junjie Yan. Quantization mimic: Towards very tiny cnn for object detection. In *IEEE European Conference on Computer Vision, ECCV*, 2018. 2, 3

[33] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. WIDER FACE: A face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016. 5, 8

[34] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *International Conference on Learning Representations, ICLR*, 2017. 3

[35] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018. 3

[36] Lixue Zhuang, Yi Xu, Bingbing Ni, and Hongteng Xu. Flexible netwrk binarization with layer-wise priority. In *IEEE International Conference on Image Processing, ICIP*, 2018. 6