This CVPR 2021 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Layer-wise Searching for 1-bit Detectors

Sheng Xu[1], Junhe Zhao[1], Jinhu Lü[1,2], Baochang Zhang[1,3]*, Shumin Han[4], David Doermann[5]
[1] School of Automation Science and Electrical Engineering, Beihang University, Beijing, China
[2] State Key Laboratory of Software Development Environment, and Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing, China
[3]Shenzhen Academy of Aerospace Technology, Shenzhen, China
[4]Department of Computer Vision Technology, Baidu Inc., Beijing, China
[5] University at Buffalo, USA
{shengxu, jhzhao, lvjinhu, bczhang}@buaa.edu.cn, hanshumin@baidu.com, doermann@buffalo.edu

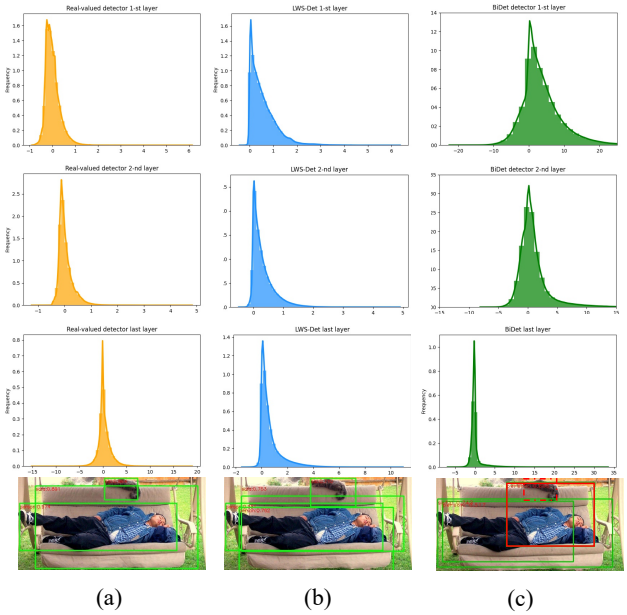
Abstract

1-bit detectors show great promise for resource-constrained embedded devices but often suffer from a significant performance gap compared with their real-valued counterparts. The primary reason lies in the error during binarization. This paper presents a layer-wise searching (LWS) strategy to generate 1-bit detectors that maintain a performance very close to the original real-valued model. The approach introduces angular and amplitude loss functions to increase detector capacity. At 1-bit layers, it exploits a differentiable binarization search (DBS) to minimize the angular error in a student-teacher framework. We also learn the scale factor by minimizing the amplitude loss in the same student-teacher framework. Extensive experiments show that LWS-Det outperforms state-of-the-art 1-bit detectors by a considerable margin on the PASCAL VOC and COCO datasets. For example, the LWS-Det achieves 1-bit Faster-RCNN with ResNet-34 backbone within 2.0% *mAP of its real-valued counterpart on the PASCAL VOC dataset.*

1. Introduction

Object detection is a fundamental task in computer vision [8, 22], and deep convolutional neural networks (CNN) [31–33] dominate recent performance advancements. But CNN models typically have millions of parameters and require billions of floating-point operations (FLOPs) to compute, limiting their deployment on resource-limited platforms.

Substantial efforts have been made to compress and accelerate CNNs for efficient online inference. Methods include compact network design [15, 27], network pruning [14, 19, 44], low-rank decomposition [7], quantization [30, 41], and knowledge distillation [34]. Quantization is particularly suitable for deployment on AI chips because it reduces the bit-width of network parameters and activations for efficient inference. Binarization, an extreme form of quantization, compresses the weights and activations of CNNs into a single bit, which can decrease the storage re-



(a) (b) (c)

Figure 1. Example layer-wise feature map distribution and detection results of (a) a real-valued detector, (b) LWS-Det, and (c) BiDet. We extract the feature maps of the first, second, and final binarized layers and illustrate their distributions based on the frequency-value histogram in rows 1-3. The last row shows the detection result.

*Baochang Zhang is the corresponding author.

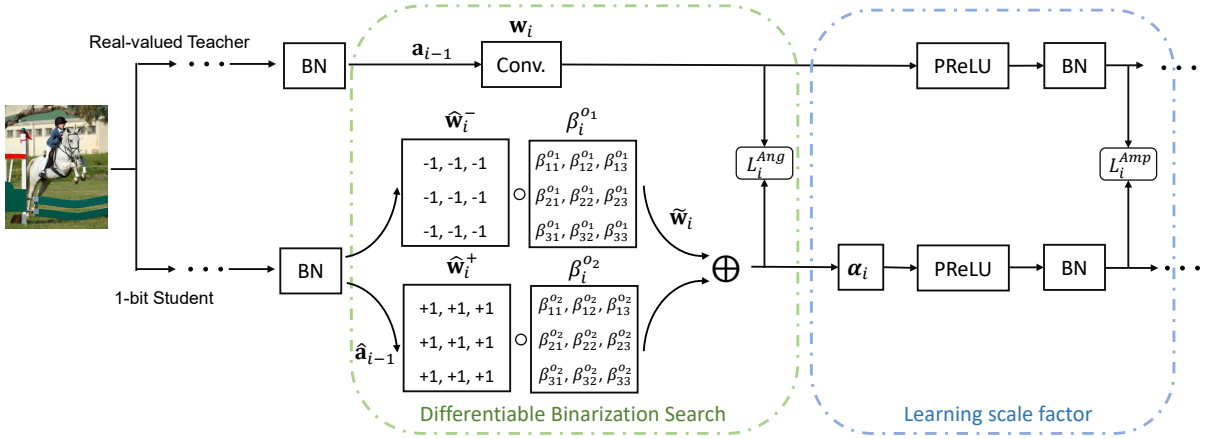


Figure 2. Our LWS-Det. From left to right are the input, the search process, and the learning process. For a given 1-bit convolution layer, LWS-Det first searches for the binary weight (+1 or -1) by minimizing the angular loss supervised by a real-valued teacher detector. LWS-Det learns the real-valued scale factor $\boldsymbol{\alpha}$ to enhance the feature representation ability.

quirements by $32\times$ and computation cost by up to $58\times$ [30]. Binarized detectors have been an important contribution to object detection by speeding up the CNN feature extraction process to allow bounding box detection and object classification in real-time [36, 38, 40]. For example, the 1-bit detectors can theoretically achieve $15\times$ acceleration in an SSD300 framework [24] with its purely logical computation using XNOR operations on binary weights and binary activations. These are highly energy-efficient for embedded devices and possess the potential for being deployed directly on next-generation memristor-based hardware.

Despite these attractive characteristics, the performance of 1-bit detectors typically degrades to the point where they are not broadly deployed on real-world embedded devices. For example, BiDet [38] only achieves 13.2% mAP@$[.5, .95]$ on the COCO `minival` dataset [22], resulting in an accuracy gap of 10.0% below its real-valued counterpart (on SSD300 framework). The reason, we believe, lies in the fact that the layer-wise binarization error significantly affects 1-bit detector learning.

Fig. 1 shows the layer-wise feature map distribution and detection results of a real-valued detector, our LWS-Det, and BiDet [38] from left to right. The first three rows show the distributions of feature maps. The feature map distribution of BiDet has a variance less similar to the one of the real-valued detector, leading to a result with false positives and missed detection in the 4-th row. In comparison, our LWS-Det can reduce the binarization error and provide better detection results.

In this paper, we present layer-wise search method to produce an optimized 1-bit detector (LWS-Det) using the student-teacher framework to narrow the performance gap. As shown in Fig. 2, we minimize the binarization error by decoupling it into an angular error and an amplitude error. We search for the binarized weight supervised by well-designed losses between real-valued convolution and 1-bit convolution under differentiable binarization search (DBS) framework, following the method DARTS [23, 45]. We formulate the binarization problem as the combination of -1 and 1, while a differentiable search can explore the binary space to significantly improve the capacity of 1-bit detectors. To improve representation ability of LWS-Det, we design two losses to supervise the 1-bit convolution layer from angular and amplitude perspective. In this way, we obtain a powerful 1-bit detector (LWS-Det) that can minimize both angular and amplitude errors in the same framework. Our contributions are summarized as:

- We introduce the differentiable search method for 1-bit detectors, which minimize the angular error under the supervision of the real-valued network.
- We learn the scale factor to minimize the amplitude loss based on the student-teacher framework. As a result, the representation ability of our detector is significantly improved.
- We evaluate our LWS-Det on the PASCAL VOC and the large-scale COCO datasets for a comprehensive comparison with state-of-the-art 1-bit detectors. The results show that our methods outperform other state-of-the-art BNN-based detectors by a sizable margin. For example, on COCO, the 1-bit Faster-RCNN with ResNet-50 backbone obtained by LWS-Det achieves 31.7% mAP, which outperforms all current 1-bit detectors.

2. Related Work

1-bit Detectors. BiDet [38] fully utilizes the 1-bit neural network's representational capacity for object detection by

redundancy removal and generalizes the information bottleneck principle to object detection. The amount of information in the high-level feature maps is constrained, and the mutual information between the feature maps and object detection is maximized. ASDA-FRCNN [40] suppresses the shared amplitude between the full-precision and the binary kernels, which significantly improves the performance of the Faster R-CNN detector.

1-bit CNNs. BinaryNet, based on BinaryConnect, was proposed to train CNNs with binary weights [6]. The activations are triggered at run-time while the parameters are computed during training. In [30], XNOR-Net is introduced to improve convolutional operations by binarizing the weight values with an estimation of a binary weight filter. XNOR++ [2] designs an efficient estimator for a layer-wise feature map of 1-bit CNN. Bi-Real-Net [25] designs a magnitude-aware gradient with respect to the weight for updating the binarized weight parameters. [28] rescales the feature maps on the channels according to the input before binarized operations and adds a gating module as the SE-Net [16]. ReActNet [26] replaces the conventional PReLU [12] and the sign function of the BNNs with RPReLU and RSign with a learnable threshold, thus improving the performance of BNNs.

Neural Architecture Search (NAS). Recently, neural architecture search (NAS) [45] has been attracting booming attention, due mainly to its satisfying capacity to model a network. The early efforts of NAS [3, 45] need enormous computation and resources. Later, such method is generalized into a much lighter framework, *i.e.* one-shot architecture search [4, 23, 39, 42]. [23] introduces DARTS, a differentiable framework and thus combines the search and evaluation stages into one. Based on DARTS, researchers have found some of its drawbacks and proposed a few improved approaches over DARTS [3, 39].

Knowledge Distillation. Knowledge distillation is a main branch of model compression methods, aiming to transfer knowledge from a trained teacher network to a smaller and faster student model. Soft-labels generated by the teacher is used to teach the student, which is first proposed by [1]. [9] redefined knowledge distillation as training a shallower network to approach the teacher's output after the softmax layer.

Unlike prior work, our work introduces a differentiable search method to design a new 1-bit detector based on the student-teacher framework. We achieve a powerful 1-bit detector that can bridge the performance gap compared with the real-valued counterparts.

3. LWS-Det

This section describes our LWS-Det in detail. We first illustrate the binarization error of 1-bit CNNs from the angular and amplitude perspective. Secondly, we describe the formulation of LWS-Det and the differentiable binarization search (DBS) method based on the angular loss minimization. We then describe our approach to learn the scale parameters by minimizing the amplitude loss. Finally, we outline the whole training process of LWS-Det.

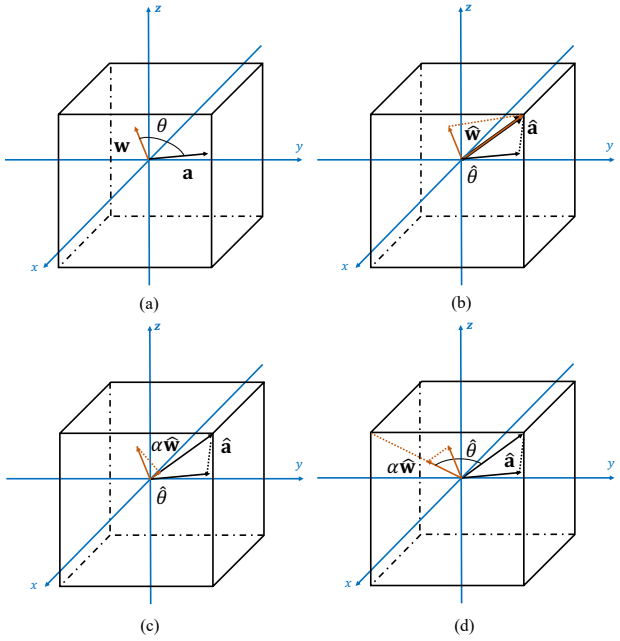


Figure 3. An illustration of binarization error in the 3-dimension space. (a) The intersection angle θ of real-valued weight $\mathbf{w}$ and activation $\mathbf{a}$ is significant. (b) After binarization ($\hat{\mathbf{w}}$, $\hat{\mathbf{a}}$) based on sign function, the intersection angle $\hat{\theta} = 0$. (c) $\hat{\theta} = 0$ based on XNOR-Net binarization. (d) Ideal binarization via angular and amplitude error minimization.

3.1. Preliminaries

Given a conventional CNN model, we denote $\mathbf{w}_i \in \mathbb{R}_{n_i}$ and $\mathbf{a}_i \in \mathbb{R}_{m_i}$ as its weights and feature maps in the i-th layer, where $n_i = C_i \cdot C_{i-1} \cdot K_i \cdot K_i$ and $m_i = C_i \cdot W_i \cdot H_i$. C_i represents the number of output channels of the i-th layer. (W_i, H_i) are the width and height of the feature maps and K_i is the kernel size. We then have

$$\mathbf{a}_i = \mathbf{a}_{i-1} \otimes \mathbf{w}_i, \tag{1}$$

where $\otimes$ is the convolution operation. We omit the batch normalization (BN) and activation layers for simplicity. The 1-bit model aims to quantize $\mathbf{w}_i$ and $\mathbf{a}_i$ into $\widehat{\mathbf{w}}_i \in \{-1, +1\}$ and $\widehat{\mathbf{a}}_i \in \{-1, +1\}$ using the efficient xnor and bit-count operations to replace full-precision operations. Following [5, 6], the forward process of the 1-bit CNN is:

$$\widehat{\mathbf{a}}_i = \mathrm{sign}(\widehat{\mathbf{a}}_{i-1} \odot \widehat{\mathbf{w}}_i), \tag{2}$$

where $\odot$ represents the xnor and bit-count operations and $\mathrm{sign}(\cdot)$ denotes the sign function, which returns 1 if the input is greater than zero, and -1 otherwise. Such binarization

process will bring the binarization error, which can be visible in Fig. 3 (a) and (b). The product of 1-bit convolution (b) cannot simulate the real-valued one (a) in both angular and amplitude.

Substantial efforts have been made to optimize this error. [25, 30, 36] formulate the object as

$$L_i^{\mathbf{w}} = \|\mathbf{w}_i - \boldsymbol{\alpha}_i \circ \widehat{\mathbf{w}}_i\|_2^2, \tag{3}$$

where $\circ$ denotes the channel-wise multiplication and $\boldsymbol{\alpha}_i$ is the vector consisting of channel-wise scale factors. Fig. 3 (c) [25, 30, 36] learns α_i by directing optimizing $L_i^{\mathbf{w}}$ to 0, and thus the explicit solution is

$$\boldsymbol{\alpha}_i^j = \frac{\|\mathbf{w}_i^j\|_1}{C_{i-1} \cdot K_i^j \cdot K_i^j}, \tag{4}$$

where j denotes the j-th channel of i-th layer. Other works [10] dynamically evaluate Equ. 3 rather than explicitly solve or modify $\boldsymbol{\alpha}_i$ to other shapes [2].

Prior works mostly focus on the kernel reconstruction but neglect the angular information, as shown in Fig. 3 (d). One drawback of existing methods lies in its ineffectiveness when binarizing a very small float value as shown in Fig. 3. Differently, we leverage the strong capacity of a differentiable search to fully explore a binary space for an ideal combination of -1 and $+1$ without a ambiguous binarization process involved.

3.2. Formulation of LWS-Det

We regard the 1-bit object detector as a student network, which can be searched and learned based on a teacher network (real-valued detector) layer-wise. Our overall framework is illustrated in Fig. 2. As depicted above, the main learning objective (layer-wise binarization error) is defined as

$$E = \sum_{i=1}^{N} \|\mathbf{a}_{i-1} \otimes \mathbf{w}_i - \widehat{\mathbf{a}}_{i-1} \odot \widehat{\mathbf{w}}_i \circ \boldsymbol{\alpha}_i\|_2^2, \tag{5}$$

where N is the number of binarized layers. We then optimize E layer-wise as

$$\underset{\widehat{\mathbf{w}}_i, \boldsymbol{\alpha}_i}{\operatorname{argmin}} E_i(\widehat{\mathbf{w}}_i, \boldsymbol{\alpha}_i; \mathbf{w}_i, \mathbf{a}_{i-1}, \widehat{\mathbf{a}}_{i-1}), \quad \forall i \in [1,\ N]. \tag{6}$$

In LWS-Det, we learning Equ. 6 by decoupling it into angular loss and amplitude loss, where we optimize the angular loss by differentiable binarization search (DBS) and the amplitude loss by learning the scale factor.

3.3. Differentiable binarization search for the 1-bit weight

We formulate the binarization task as a differentiable search problem. Considering the 1-bit weight is closely related to angular as shown in Fig. 3, we define an angular loss to supervise our search process as

$$\begin{aligned} L_i^{Ang} &= \|\cos\theta_i - \cos\widehat{\theta}_i\|_2^2 \\ &= \|\frac{\mathbf{a}_{i-1} \otimes \mathbf{w}_i}{\|\mathbf{a}_{i-1}\|_2 \|\mathbf{w}_i\|_2} - \frac{\widehat{\mathbf{a}}_{i-1} \odot \widehat{\mathbf{w}}_i}{\|\widehat{\mathbf{a}}_{i-1}\|_2 \|\widehat{\mathbf{w}}_i\|_2}\|_2^2. \end{aligned} \tag{7}$$

For the learning process of the i-th layer, the objective is formulated as

$$\underset{\widehat{\mathbf{w}}_i}{\operatorname{argmin}} L_i^{Ang}(\widehat{\mathbf{w}}_i; \widehat{\mathbf{a}}_i, \mathbf{w}_i, \mathbf{a}_i). \tag{8}$$

We introduce the DARTS framework to solve Equ. 8, named differential binarization search (DBS). We follow [23] to efficiently search for $\widehat{\mathbf{w}}_i$. Specifically, we approximate $\widehat{\mathbf{w}}_i$ by the weighed probability of two matrices whose weights are all set as -1 and +1, respectively. We relax the choice of a particular weight by the probability function defined as

$$p_i^{o_k} = \sum_{o_k \in \mathcal{O}} \frac{\exp(\boldsymbol{\beta}_i^{o_k})}{\sum_{o_k' \in \mathcal{O}} \exp(\boldsymbol{\beta}_i^{o_k'})}, \ \text{ s.t. } \ \mathcal{O} = \{\widehat{\mathbf{w}}_i^-, \widehat{\mathbf{w}}_i^+\}, \tag{9}$$

where $p_i^{o_k}$ is the probability matrix belongs to the operation $o_k \in \mathcal{O}$. The search space $\mathcal{O}$ is defined as the two possible weights: $\{\widehat{\mathbf{w}}_i^-, \widehat{\mathbf{w}}_i^+\}$. For the inference stage, we select the weight owning the max probability as

$$\widetilde{\mathbf{w}}_{i,l} = \arg\max_{o_k} p_{i,l}^{o_k}, \tag{10}$$

where $p_{i,l}^{o_k}$ denotes the probability that the l-th weight of the i-th layer belongs to operation o_k. Hence, the l-th weight of $\widetilde{\mathbf{w}}$, *i.e.* $\widetilde{\mathbf{w}}_{i,l}$, is defined by the operation having the maximum probability. In this way, we modify Equ. 7 by substituting $\widehat{\mathbf{w}}_i$ to $\widetilde{\mathbf{w}}_i$ as

$$L_i^{Ang} = \|\frac{\mathbf{a}_{i-1} \otimes \mathbf{w}_i}{\|\mathbf{a}_{i-1}\|_2 \|\mathbf{w}_i\|_2} - \frac{\widehat{\mathbf{a}}_{i-1} \odot \widetilde{\mathbf{w}}_i}{\|\widehat{\mathbf{a}}_{i-1}\|_2 \|\widetilde{\mathbf{w}}_i\|_2}\|_2^2. \tag{11}$$

By this, we retain the top-1 strongest operations (from distinct weights) for each weight of $\widehat{\mathbf{w}}_i$ in the discrete set {+1, -1}.

3.4. Learning the scale factor

After searching for $\widehat{\mathbf{w}}_i$, we learn the real-valued layers between the i-th and $(i{+}1)$-th 1-bit convolution. We omit the batch normalization (BN) and activation layers for simplicity. We can directly simplify Equ. 5 as

$$L_i^{Amp} = E_i(\boldsymbol{\alpha}_i; \mathbf{w}_i, \widetilde{\mathbf{w}}_i, \mathbf{a}_{i-1}, \widehat{\mathbf{a}}_{i-1}). \tag{12}$$

Following conventional BNNs [10, 11], we employ Equ. 3 to further supervise scale factor $\boldsymbol{\alpha}_i$. According to [37], we

Algorithm 1 Training 1-bit detectors via LWS-Det.

Input: The training dataset, pre-trained teacher model.
Output: 1-bit detector.

Initialize $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i^{o_i} \sim \mathcal{N}(0,1)$ and other real-valued parameters layer-wise;
for $i = 1$ to N **do**
  while Differentiable search **do**
    Compute L_i^{Ang}, L_i^{Amp}, L_i^{W}
  end while
end for
Compute L^{GT}, L^{Lim}
for $i = N$ to 1 **do**
  Update parameters vis back propagation
end for
repeat
until The algorithm converges.

employ fine-grained feature limitation to auxiliarily learning the detection prior. Hence, the supervision of LWS-Det is formulated as

$$L = L^{GT} + \lambda L^{Lim} + \mu \sum_{i=1}^{N} (L_i^{Ang} + L_i^{Amp}) + \gamma \sum_{i=1}^{N} L_i^{\mathbf{w}}, \tag{13}$$

where L^{GT} is the detection loss derived from ground truth label and L^{Lim} is the fine-grained feature limitation defined in [37]. The process of LWS-Det is outlined in Algorithm 1.

4. Experiments

Comprehensive experiments are conducted to evaluate our proposed method on two datasets for object detection: PASCAL VOC [8] and COCO [22]. First, we introduce the implementation details of LWS-Det. Then we validate the effectiveness of differential binarization search (DBS) and convergence of our method through ablation studies. Finally, we compare our method with state-of-the-art 1-bit CNNs on object detection to demonstrate the superiority of LWS-Det.

4.1. Datasets and Implementation Details

PASCAL VOC. The PASCAL VOC dataset contains natural images from 20 different classes. We train our model on the VOC `trainval2007` and VOC `trainval2012` sets, which consist of approximately 16k images, and we evaluate our method on the VOC `test2007` set, which includes 4952 images. Following [8], we use the mean average precision (mAP) as the evaluation criterion.

COCO. The COCO dataset consists of images from 80 categories. We conduct experiments on the COCO `2014` [22] object detection track. Models are trained with the combination of 80k images from the COCO `train2014` and 35k images sampled from COCO `val2014`, *i.e.*, COCO `trainval35k`. On the remaining 5k images from the COCO `minival`, we test our method. Following the standard evaluation metric of COCO, we report the average precision (AP) for IoU$\in [0.5 : 0.05 : 0.95]$ denoted as mAP@[.5, .95]. We also report AP_{50}, AP_{75}, AP_s, AP_m, and AP_l to further analyze our method.

Implementation Details. We train our LWS-Det with the Faster-RCNN[1] [33] and SSD [24] detection framework based on ResNet-18, ResNet-34, ResNet-50 [13] and VGG-16 [35]. We use PyTorch [29] to implement LWS-Det. And we conduct the experiments on 4 NVIDIA Tesla P40 GPUs with 24 GB and 128G RAM. We pre-train the backbone of the teacher model on ImageNet ILSVRC12 [18] for the task of image classification and fine-tune the teacher detector on the dataset of object detection task. Also, we employ DBS method to per-train the backbone on ImageNet ILSVRC12. The batch size is assigned to be 32 and 16 for SSD and Faster-RCNN, respectively. The SGD optimizer is applied.

Following the implementation of 1-bit CNNs in [25], we retain the first layer, shortcut, and last layer (1×1 convolution layer of RPN and a fully-connected layer of the bbox head) in the detection networks as real-valued on Faster-RCNN framework. For the SSD framework, the extra layer is also set as real-valued following BiDet [38]. We modify the architecture of ResNet-18/34 with extra shortcut and PReLU [12] following [38] and [10] respectively. The architecture of VGG-16 is modified with extra shortcut following [38]. Also, we modify the architecture of ResNet-50 following [25]. To enhance the performance, we replace the lateral connection of FPN [21] neck with 3×3 1-bit convolution. All the experiments on Faster-RCNN framework follow this modification.

During LWS-Det, we train the model for 12 epochs with a learning rate set as 0.004 which decay by multiplying 0.1 at the 9-th epoch. We repeat with the VOC for 10 times and the COCO dataset for 5 times for each epoch when training the SSD detector. As for hyper-parameter, we set λ as 1×10^{-2} following [37]. γ is set as 1×10^{-4}. For different frameworks, we conduct experiments to fine-tune μ towards better performance.

4.2. Ablation Study

Effectiveness of DBS. We first compare our DBS method with three other methods to produce binarized weights: Random Search [17], Sign [6], and RSign [26]. As shown in Tab. 1, we evaluate the effectiveness of DBS on two detectors: one-stage SSD and two-stage Faster-RCNN. On the Faster-RCNN detector, the usage of DBS improves the

[1] In this paper, Faster-RCNN denotes the two-stage Faster-RCNN implemented with FPN neck. Different from the one in [38].

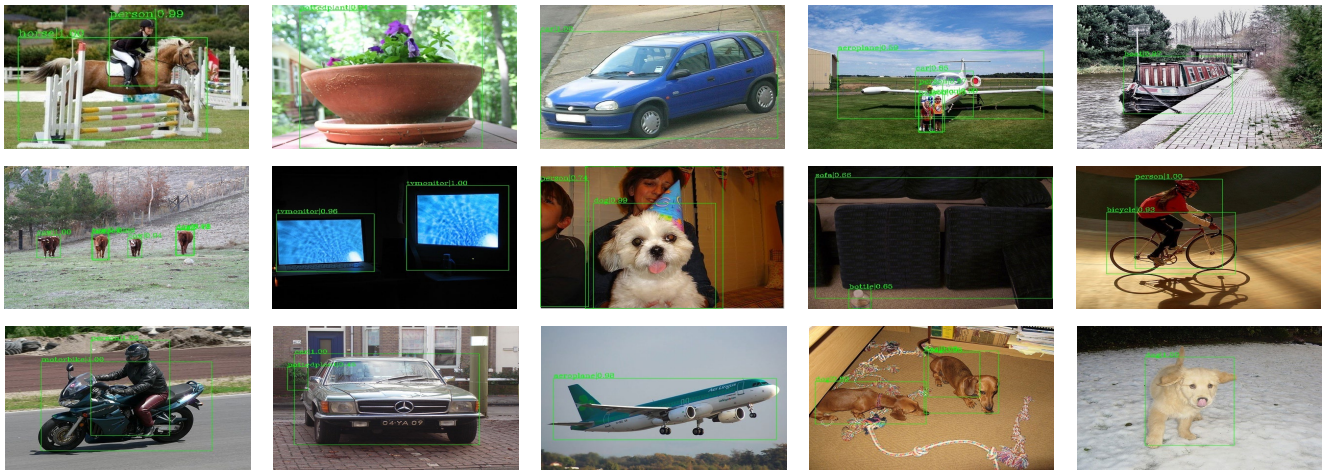


Figure 4. Qualitative results on PASCAL VOC `test2007` (best viewed in color).

Table 1. Ablation study: Comparison of the performance of different binarization method with DBS.

Framework	Backbone	Binarization Method	mAP
Faster-RCNN	ResNet-18	Sign	65.1
		RSign	68.9
		Random Search	64.1
		DBS	73.2
		Real-valued	76.4
SSD	VGG-16	Sign	65.9
		RSign	68.1
		Random Search	60.1
		DBS	71.4
		Real-valued	74.3

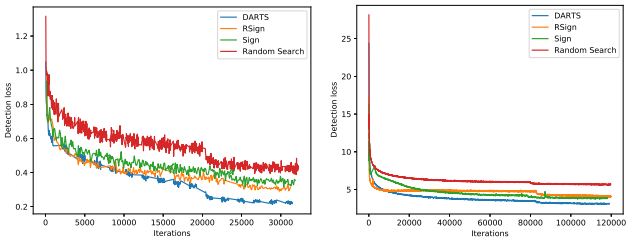


Figure 5. Convergence Faster-RCNN with ResNet-18 backbone (left) and SSD with VGG-16 backbone (right) based on different binarizations training on VOC `trainval2007` and `trainval2012`.

mAP by 8.1%, 4.3% and 9.1% compared to Sign, RSign and Random Search, respectively, under the same student-teacher framework. On the SSD detector, DBS also enhances the mAP by 5.5%, 3.3% and 11.3% compared to other binarization methods, respectively, which is very significant for the object detection task.

Convergence analysis. We evaluate the convergence of detection loss during the training process in comparison with other situations on two detectors: Faster-RCNN with ResNet-18 backbone and SSD with VGG-16 backbone. As plotted in Fig. 5, the training curve of the LWS-Det based on random search oscillates vigorously, which is suspected to be triggered by less optimized angular error resulting from the randomly searched binary weights. Also, our DBS achieves a minimum loss during training in comparison with Sign and RSign. This also confirms our DBS method can binarize the weights with minimum angular error, which explains the best performance in Tab. 1.

4.3. Results on PASCAL VOC

In this section, we compare the proposed LWS-Det with other state-of-the-art 1-bit neural networks, including Bi-Real-Net [25], BiDet [38], and ReActNet [26], on the same framework for the task of object detection on the PASCAL VOC datasets. We also report the detection performance of the multi-bit quantized networks DoReFa-Net [43].

Tab. 2 illustrates the comparison of computational complexity, storage cost, and the mAP across different quantization methods and detection frameworks. Our LWS-Det significantly accelerates the computation and saves the storage on various detectors. We compute the memory usage by comparing our approach with XNOR-Net [30] and the corresponding full-precision network. The memory usage is computed as the summation of 32-bit times the number of full-precision kernels and 1-bit time the number of binary kernels in the networks. The number of float operations (FLOPs) is computed as with Bi-Real-Net [25]. The bit-wise Xnor operation and bit-count operation can be performed in parallel by the current generation of CPUs. The FLOPs are calculated as the number of real-valued floating-point multiplication plus $\frac{1}{64}$ of the number of 1-bit multiplications.

Table 2. Comparison of memory usage, FLOPs, and mAP (%) with state-of-the-art 1-bit CNNs in both one-stage and two-stage detection frameworks on VOC `test2007`. The detector with the real-valued and multi-bit backbone is given for reference. The **bold** denotes the best result.

Framework	Input	Backbone	Quantization Method	W/A (bit)	Memory Usage	GFLOPs	mAP(%)
Faster-RCNN	1000×600	ResNet-18	Real-valued	32/32	112.88MB	96.40	76.4
			DoReFa-Net	4/4	21.59MB	27.15	73.3
			Bi-Real-Net	1/1	16.61MB	18.49	60.9
			BiDet		16.61MB	18.49	62.7
			ReActNet		16.61MB	18.49	69.6
			LWS-Det		16.61MB	18.49	**73.2**
		ResNet-34	Real-valued	32/32	145.12MB	118.80	77.8
			DoReFa-Net	4/4	29.65MB	32.31	75.6
			Bi-Real-Net	1/1	24.68MB	21.49	63.1
			BiDet		24.68MB	21.49	65.8
			ReActNet		24.68MB	21.49	72.3
			LWS-Det		24.68MB	21.49	**75.8**
		ResNet-50	Real-valued	32/32	164.88MB	127.76	79.5
			Bi-Real-Net	1/1	29.61MB	21.95	65.7
			ReActNet		29.61MB	21.95	73.1
			LWS-Det		29.61MB	21.95	**76.9**
SSD	300×300	VGG-16	Real-valued	32/32	105.16MB	31.44	74.3
			DoReFa-Net	4/4	29.58MB	6.67	69.2
			Bi-Real-Net	1/1	21.88MB	2.13	63.8
			BiDet		21.88MB	2.13	66.0
			ReActNet		21.88MB	2.13	68.4
			LWS-Det		21.88MB	2.13	**71.4**

Faster-RCNN. The results for 1-bit Faster-RCNN on VOC `test2007` are summarized from lines 2 to 16 in Tab. 2. Our LWS-Det achieves the comparable performance with real-valued Faster-RCNN with ResNet-18/34/50 backbone ($\{73.2\%, 75.8\%, 76.9\% \} vs. \{76.4\%, 77, 8\%, 79.5\%\}$), which with significantly accelerate the computation and save the storage by $5.21\times/5.52\times/5.82\times$ and $6.79\times/5.88\times/5.57\times$ when compared with the real-valued counterparts.

Compared with 1-bit methods, we observe significant performance improvements with our LWS-Det over other state-of-the-arts. With the ResNet-18 backbone, our LWS-Det outperforms Bi-Real-Net, BiDet, and ReActNet by 12.3%, 10.5%, and 3.6% mAP with the same memory usage and FLOPs. Likewise, with the ResNet-34 backbone, our LWS-Det outperforms the Bi-Real-Net, BiDet, and ReActNet by 12.7%, 10.0% and 3.5% mAP. Furthermore, LWS-Det surpasses Bi-Real-Net and ReActNet by 11.2% and 3.8% with ResNet-50 backbone. All improvements are very significant for the object detection task.

Moreover, our LWS-Det even outperforms the 4-bit DoReFa-Net method by 0.2% mAP with lower FLOPs and memory usage with ResNet-34 backbone.

SSD. The bottom lines in Tab. 2 illustrate that our LWS-Det can accelerate the computation and save storage by $14.76\times$ and $4.81\times$ on the SSD300 framework with a VGG-16 backbone compared with real-valued counterparts. Relatively speaking, the performance gap is rather small (71.4% $vs.$ 74.3%).

Compared to 1-bit Bi-Real-Net, BiDet, and ReActNet, our LWS-Det can achieve 7.6%, 5.4%, and 3.0% higher mAP with the same FLOPs and memory usage. Likewise, our LWS-Det outperforms 4-bit DoReFa-Net by 2.2% with obviously lower FLOPs and memory usage.

In short, we achieved a new state-of-the-art performance compared to other 1-bit CNNs on various detection frameworks with various backbones on PASCAL VOC. We also achieve a much closer performance to full-precision models, as demonstrated in extensive experiments, clearly validating the superiority of our LWS-Det.

4.4. Results on COCO

The COCO dataset is much more challenging for object detection than PASCAL VOC due to its diversity and scale. We compare the proposed LWS-Det with state-of-the-art 1-bit neural networks, including Bi-Real-Net [25], BiDet [38], and ReActNet [26] on COCO. For reference, we report the detection performance of the 4-bit quantized FQN [20].

Tab. 3 shows the mAP, AP with different IoU thresholds, and AP of objects with different scales. Limited by the page width, we do not show the memory usage and FLOPs in Tab. 3. We conduct experiments on Faster-RCNN and SSD detectors and convey the results in two parts.

Faster-RCNN. Compared with the state-of-the-art 1-bit

Table 3. Comparison of mAP@[.5, .95](%), AP with different IoU threshold and AP for objects in various sizes with state-of-the-art binarized object detectors in Faster-RCNN and SSD detection framework on COCO `minival`, where the performance of real-valued and 4-bit detectors is reported for reference. The **bold** denotes the optimal result.

Framework	Input	Backbone	Quantization Method	W/A (bit)	mAP@[.5, .95]	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Faster-RCNN	1333×800	ResNet-18	Real-valued	32/32	32.2	53.8	34.0	18.0	34.7	41.9
			FQN	4/4	28.1	48.4	29.3	14.5	30.4	38.1
			Bi-Real-Net	1/1	17.4	33.1	17.1	7.1	18.7	27.9
			BiDet		19.4	34.6	18.2	7.8	20.4	29.4
			ReActNet		21.1	38.5	20.5	9.7	23.5	32.1
			LWS-Det		**26.9**	**44.9**	**27.7**	**12.9**	**28.7**	**38.3**
		ResNet-34	Real-valued	32/32	35.8	57.6	38.4	21.1	39.0	46.1
			FQN	4/4	31.8	52.9	33.9	17.6	34.4	42.2
			Bi-Real-Net	1/1	20.1	37.1	19.3	7.7	21.4	29.2
			BiDet		21.7	41.8	23.1	10.1	24.9	34.5
			ReActNet		23.4	43.3	24.4	10.7	25.9	35.5
			LWS-Det		**29.9**	**49.2**	**30.1**	**15.1**	**32.1**	**40.9**
		ResNet-50	Real-valued	32/32	37.7	59.3	40.9	22.0	41.5	48.9
			FQN	4/4	33.1	54.0	35.5	18.2	36.2	43.6
			Bi-Real-Net	1/1	22.9	40.0	21.5	10.5	25.6	36.9
			ReActNet		26.1	47.7	35.6	14.1	28.9	38.9
			LWS-Det		**31.7**	**52.1**	**37.5**	**17.7**	**34.9**	**43.1**
SSD	300×300	VGG-16	Real-valued	32/32	23.2	41.2	23.4	5.3	23.2	39.6
			DoReFa-Net	4/4	19.5	35.0	19.6	5.1	20.5	32.8
			Bi-Real-Net	1/1	11.2	26.0	8.3	3.1	12.0	18.3
			BiDet		13.2	28.3	10.5	5.1	14.3	20.5
			ReActNet		15.3	30.0	13.2	5.4	16.3	25.0
			LWS-Det		**17.1**	**32.9**	**16.1**	**5.5**	**17.4**	**26.7**

methods, our LWS-Det outperforms other methods by significant margins. With the ResNet-18 backbone, our LWS-Det improves the mAP@[.5, .95] by 9.5%, 7.5%, and 5.8% compared with state-of-the-art Bi-Real-Net, BiDet, and ReActNet, respectively. Also, our LWS-Det improves the mAP@[.5, .95] by 9.8%, 8.2%, and 6.5% compared with Bi-Real-Net, BiDet, and ReActNet with the ResNet-34 backbone. Furthermore, our LWS-Det outperforms Bi-Real-Net and ReActNet by 9.8% and 5.6% with ResNet-50 backbone. Similarly, on other APs with different IoU thresholds, our LWS-Det outperforms others obviously.

Compared to 4-bit FQN, our 1-bit LWS-Det achieves 1.2%, 1.9% and 1.4% mAP lower with three different backbones. However, the FLOPs and memory usage of LWS-Det are obviously much lower. Also LWS-Det can achieve 31.7% mAP@[.5, .95] on 1-bit Faster-RCNN with ResNet-50 backbone, which is the new state-of-the-art results.

SSD. Our LWS-Det achieves 17.1% mAP@[.5, .95] on the SSD300 framework with the VGG-16 backbone, which outperform Bi-Real-Net, BiDet, and ReActNet by 5.9%, 3.9% and 1.8% mAP, respectively.

To conclude, compared with the baseline methods of network quantization, our method achieves the best performance in terms of the AP with different IoU thresholds, and the AP for objects in different sizes on COCO, demonstrating LWS-Det's superiority and universality in different application settings.

5. Conclusion

This paper has proposed a novel 1-bit convolutional neural network training method to layer-wise minimize the angular and amplitude error under a student-teacher framework. The presented LWS-Det introduces differentiable binarization search. Specifically, LWS-Det searches the best binary weight with minimum angular loss. LWS-Det also learns the scale factor to minimize the amplitude error. As a result, the performance gap between 1-bit and real-valued detectors can be significantly reduced. Extensive experiments validate the superiority of LWS-Det in object detection compared with state-of-the-art 1-bit detectors. Future work will focus on implementing our LWS-Det on ARM CPUs for our future work.

6. Acknowledgement

The work was supported by the Natural Science Foundation of China (62076016). Baochang Zhang is in part supported by Shenzhen Science and Technology Program (No.KQTD2016112515134654). This study was supported by Grant NO.2019JZZY011101 from the Key Research and Development Program of Shandong Province to Dianmin Sun.

References

[1] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Proc. of NIPS Workshop*, 2014. 3

[2] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. In *Proc. of BMVC*, 2019. 3, 4

[3] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *Proc. of AAAI*, 2018. 3

[4] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv*, 2018. 3

[5] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Proc. of NIPS*, 2015. 3

[6] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv*, 2016. 3, 5

[7] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. In *Proc. of NIPS*, 2013. 1

[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 2010. 1, 5

[9] Oriol Vinyals Geoffrey Hinton and Jeff Dean. Distilling the knowledge in a neural network. In *Proc. of NIPS*, 2014. 3

[10] Jiaxin Gu, Ce Li, Baochang Zhang, Jungong Han, Xianbin Cao, Jianzhuang Liu, and David Doermann. Convolutional neural networks for 1-bit cnns via discrete back propagation. In *Proc. of AAAI*, 2019. 4, 5

[11] Jiaxin Gu, Junhe Zhao, Xiaolong Jiang, Baochang Zhang, Jianzhuang Liu, Guodong Guo, and Rongrong Ji. Bayesian optimized 1-bit cnns. In *Proc. of ICCV*, 2019. 4

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. of ICCV*, 2015. 3, 5

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, 2016. 5

[14] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proc. of IJCAI*, 2018. 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. In *Proc. of CVPR*, 2017. 1

[16] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. of CVPR*, 2018. 3

[17] Martin Jaggi Claudiu Musat Kaicheng Yu, Christian Sciuto and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *Proc. of ICLR*, 2020. 5

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. of NIPS*, 2012. 5

[19] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *Proc. of ICLR*, 2016. 1

[20] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *Proc. of CVPR*, 2019. 7

[21] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. of CVPR*, 2017. 5

[22] 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. of ECCV*, 2014. 1, 2, 5

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *Proc. of ICLR*, 2019. 2, 3, 4

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proc. of ECCV*, 2016. 2, 5

[25] Zechun Liu, Wenhan Luo, Baoyuan Wu, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Binarizing deep network towards real-network performance. *International Journal of Computer Vision*, 128(1):202–219, 2020. 3, 4, 5, 6, 7

[26] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *Proc. of ECCV*, 2020. 3, 5, 6, 7

[27] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proc. of ECCV*, 2018. 1

[28] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *Proc. of ICLR*, 2020. 3

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Proc. of NIPS*, 2017. 5

[30] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proc. of ECCV*, 2016. 1, 2, 3, 4, 6

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. of CVPR*, 2016. 1

[32] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proc. of CVPR*, 2017. 1

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. 1, 5

[34] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *Proc. of ICLR*, 2015. 1

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. of ICLR*, 2015. 5

[36] Diwen Wan, Fumin Shen, Li Liu, Fan Zhu, Jie Qin, Ling Shao, and Heng Tao Shen. Tbn: Convolutional neural network with ternary inputs and binary weights. In *Proc. of ECCV*, 2018. 2, 4

[37] Tao Wang, Li Yuan, Xiaopeng Zhang, and Jiashi Feng. Distilling object detectors with fine-grained feature imitation. In *Proc. of CVPR*, 2019. 4, 5

[38] Ziwei Wang, Ziyi Wu, Jiwen Lu, and Jie Zhou. Bidet: An efficient binarized object detector. In *Proc. of CVPR*, 2020. 2, 5, 6, 7

[39] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv*, 2018. 3

[40] Sheng Xu, Zhendong Liu, Xuan Gong, Chunlei Liu, Mingyuan Mao, and Baochang Zhang. Amplitude suppression and direction activation in networks for 1-bit faster r-cnn. In *Proc. of EMDL*, 2020. 2, 3

[41] Zhaohui Yang, Yunhe Wang, Kai Han, Chunjing Xu, Chao Xu, Dacheng Tao, and Chang Xu. Searching for low-bit weights in quantized neural networks. In *Proc. of NIPS*, 2020. 1

[42] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial distribution learning for effective neural architecture search. In *Proc. of ICCV*, October 2019. 3

[43] 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*, 2016. 6

[44] Li'an Zhuo, Baochang Zhang, Linlin Yang, Hanlin Chen, Qixiang Ye, David Doermann, Rongrong Ji, and Guodong Guo. Cogradient descent for bilinear optimization. In *Proc. of CVPR*, 2020. 1

[45] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *Proc. of ICLR*, 2017. 2, 3