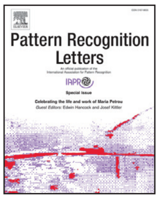
Pattern Recognition Letters 125 (2019) 91–97

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec



BDNN: Binary convolution neural networks for fast object detection[☆]

Hanyu Peng[a,b], Shifeng Chen[a,*]

[a] Multimedia Laboratory, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, 1068 Xueyuan Avenue, Shenzhen University Town, China
[b] University of Chinese Academy of Sciences, China

A R T I C L E I N F O

Article history:
Received 17 February 2018
Available online 8 April 2019

MSC:
41A05
41A10
65D05
65D17

Keywords:
Deep learning
Object detection
Network compression

A B S T R A C T

Object detection methods based on neural networks have made considerable progress. However, methods like Faster RCNN and SSD that adopt large neural networks as the base models. It's still a challenge to deploy such large detection networks in mobile or embedded devices. In this paper, we propose a low bit-width weight optimization approach to train Binary Neural Network for object detection using binary weights in training and testing. We introduce a greedy layer-wise method to train the detection network. This method boosts the performance instead of training the entire network at the same time. Our binary detection neural network (BDNN) can reduce the computational requirements and storage with competitive performance. For example, the binary network based on Faster RCNN with VGG16 can save 95% compression. In our experiments, BDNN achieves comparable performance with mAP 63.3% and outperforms SPPNet by 4.4% on PASCAL VOC 2007.

1. Introduction

Deep neural networks have achieved start-of-the-art performance in a wide range of applications [13,17,29] e.g. speech recognition [1], image classification [13], object detection [28], semantic image segmentation [18], and machine translation [24].

Object detection is a task that requires a high computational cost. Advanced algorithms like Faster RCNN [28] and SSD [21] have broken bottlenecks like region proposal generation. However, the detection approaches are not efficient enough to run on the CPU in real-time. Most networks have a large number of parameters and consume high computation. It's still a challenge to use them in the portable devices. For example, VGG-16 involves 15.3 GFLOPs and 140 M parameters. It's infeasible for the portable devices to deploy such a large network. Therefore, it's important to compress and speed up the network.

Deep model compression has recently gained much attention [5,11,19,26]. These methods have shown promising results. Most of the works are for image classification [5,26]. It's nontrivial to extend the low bit-width weight allocation method to object detection. Detection is a much more difficult task than classification. The outputs of location coordinators for detection are desired to be continuous variables. It is even more difficult for detection network when weights are all constrained to discrete variables.

In this paper, we propose a simple low bit-width weight allocation method to compress the detection networks. We first clip the model weights into the range $[-b, b]$, where b is the maximum absolute value of the weight of a layer and no more than 1. In [2], they found that if the weights are large, it would degrade the performance and the range of weights is not stable. Then, we use m-bit quantized weights to approximate the full precision weight. And we use the m-bit quantized weight during forward-pass, which can transfer the convolution to addition and subtraction operations especially when m is 1. During backward propagation, we use the real-valued weights to update the parameters. Low bit-width weights can not contain the information in smaller gradients. And it would ignore the changes of parameters and degrade the performance of the quantized detection networks. In our experiments, we find that the detection networks are hard to converge if we quantize all the weights at the same time. The performance would also degrade a lot. To make it easier, we use a greedy

[☆] *Conflict of interest:* We wish to draw the attention of the Editor to the following facts which may be considered as potential conflicts of interest and to significant financial contributions to this work. We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us. We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property.

[*] Corresponding author.

E-mail addresses: penghanyu16@mails.ucas.ac.cn (H. Peng), shifeng.chen@siat.ac.cn (S. Chen).

https://doi.org/10.1016/j.patrec.2019.03.026
0167-8655/© 2019 Elsevier B.V. All rights reserved.

layer-wise training method instead of quantizing all the weights at the same time.

1.1. Related work

Network compression methods have gained much attention and achieved great performance. Current methods of network compression can be summarized into three categories:

Knowledge distillation approaches. These ways distill the knowledge in an ensemble of deep teacher networks into much smaller student networks. The idea is first introduced in [3] and is generalized in [14]. Now this method has been widely used in many applications. The work in [4] combines this approach with hash codes to further compress neural networks. The work [23] selects identified neurons leveraging the knowledge of learned face representations in the hidden layers to train a much smaller network.

Tensor decomposition compression [9,25,32]. These methods focus on spatial redundancy by approximating the weights using low-rank tensor decomposition. The method [32] approximates the non-linear responses in multi-layers without gradient descent and determines the trade-off between acceleration and performance. The work [9] obtains the data-independent solution of tensor decomposition when solving the non-convex problem.

Low bit-width weights allocation approaches Low-bit weight quantization can save much more memory with slightly accuracy degradation. A number of recent studies have been conducted on this filed. Typical works include [15,19,26,33]. These algorithms constrain the weights to fixed-point quantization points. In addition, Rastegari et al. [26] uses one-bit activation and weight in the network and achieves considerable performance. Lin et al. [20] proposes to enhance the performance by using more weight parameters. Rastegari et al. [26] proposed to use binary weights and activations for image classification. Wen et al. [30] proposed to use low-bit width weights to accelerate the neural network on segmentation task. Duan et al. [8] introduced binary descriptor with multi-level quantization for image matching. Yan [31] proposed a discriminative compact binary face descriptor for facial kinship verification. Duan et al. [7] proposed a context-aware local binary descriptor for face recognition.

1.2. Contribution

In this paper, we make the following main contributions:

We find that the detection performance of BDNN would degrade a lot if we binarize all the weights of detection networks at the same time. We introduce a greedy layer-wise method to address the issue. The method works much better and boosts the performance.

We extend the quantization method in [15] to train BDNN with low bit-width weights. Our BDNN can save about 95% memory and make it possible to deploy our detection network on portable devices.

Our binary architecture based on SSD with reduced VGG16 achieves competitive performance on PASCAL VOC 2007 with 63.3 mAP, 8.8% less than the 32-bit full precision network. Our binary architecture using Faster RCNN achieves a comparable result with 62.6% mAP on PASCAL VOC 2007, 7.3% lower than the full precision network and outperforms the SPPNet [12] by 4.4%.

2. The proposed methods

In this section, we will introduce the low bit-width weights quantization algorithm, then we present our greedy layer-wise method to train binary detection network.

2.1. Weight quantization

We extend the weight quantization algorithm in [15]. Formally, we consider an $L-th$ layer network, where L is the number of layers in the network. In the $l-th$ $(l=1,2\ldots L)$ layer, $\mathcal{W}^l \in R^{n^l\times k^l\times k^l}$ is the $l-th$ full precision weight, n^l and k^l denote the number of feature map and kernel size. In order to obtain a stable range, we fist clip the weights into $[-b,b]$ where b is the maximum absolute value in the weights of a layer and no more than 1. If the absolute value of weights is larger than 1, the error would accumulate and the network is hard to converge. As a result, it would also increase the quantization error and degrad the performance. Our clipping functions are:

$$\tilde{\mathcal{W}^l} = clip(\mathcal{W}^l, b), \tag{1}$$

$$clip(x,b) = max(-b, min(x,b)), \tag{2}$$

where $\tilde{\mathcal{W}^l}$ denotes the clamping weights. We introduce a data-independent approach to quantize the weights for each 32-bit weight. Assume that we use m-bit to quantize the weights into the range $[-b,b]$. There have 2^m discrete values. The step-size (resolution) β can be decided by the following function:

$$\beta = \frac{2b}{2^m-1}. \tag{3}$$

Note that the step-size (resolution) would decrease at an exponential rate as m increases. The quantized weight set is $\Phi = \{-b, -b+\beta, \ldots, b-\beta, b\}$. We approximate the clipped weights as $\tilde{\mathcal{W}^l} \simeq \alpha * Q$. Here α and Q denote the scaling factor and quantized weight tensor, respectively. if we flatten the weight tensor, $\tilde{\mathcal{W}^l}$ and Q can be reshaped as an equal vector $R^{n^l\times k^l\times k^l}$, then we can formulate the quantization optimization problem as follows:

$$L(\alpha, Q) = \min_{\alpha,Q} \left\| \tilde{\mathcal{W}^l} - \alpha Q \right\|^2. \tag{4}$$

Each element Q_i $(i=1,2\ldots n^l\times k^l\times k^l)$ is picked in Φ with $Q_i = floor(\tilde{\mathcal{W}^l} + \frac{\beta}{2})$, where *floor* indicates the floor function. For each element Q_i, we select the closest value to $\tilde{\mathcal{W}^l}_i$ in Φ. We use L2 norm as the loss function:

$$L(\alpha, Q) = \min_{\alpha,Q} \left\| \tilde{\mathcal{W}^l} - \alpha Q \right\|^2. \tag{5}$$

we can obtain the scaling factor using the following equation:

$$\alpha = \frac{\tilde{\mathcal{W}^l}^T Q}{\|Q\|^2}. \tag{6}$$

2.2. Greedy layer-wise scheme

In [5] all the weights of the network are trained at the same time during the forward step. We have tried to adopt the same method to train our BDNN and we observe that the performance would degrade a lot. When we binarize the whole network, the detection errors would accumulate. the network will converge in a sub-optimal way.

Instead, we introduce a greedy layer-wise method to train the BDNN. To accelerate the training process and stabilize the training procedure, we binarize the weights of the detection network stage by stage. We fine-tune the pre-trained VGG16 model [6] for the detection task. Then we quantize the weights of bottom layers using binary weights and fine-tune the network until convergence. At the next stage, we freeze these bottom layers and continue to quantize the weights in the following layer except for the classification and location layers in convolution network. A greedy layer-wise training process is illustrated in Fig. 1.

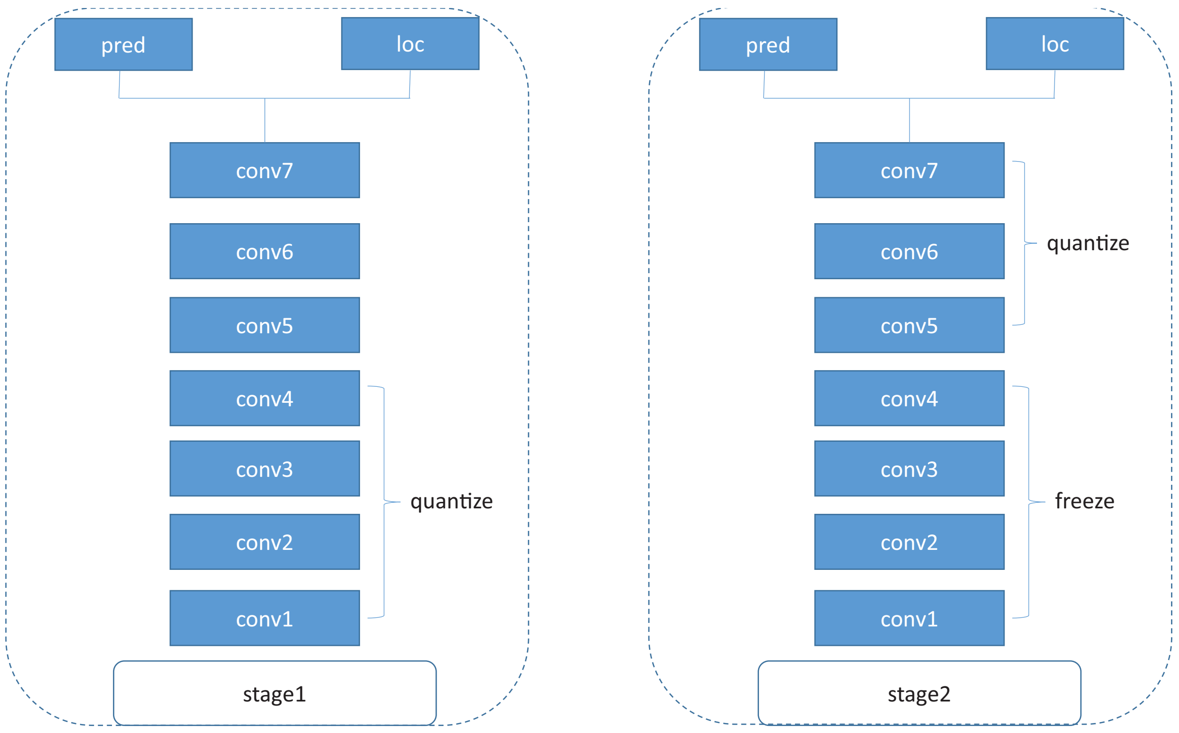


Fig. 1. Example of a 2-stage greedy layer-wise scheme for Fast RCNN with VGG16. We binarize all convolution layers except the first convolution layer, classification layer and location layer.

2.3. Training BDNN

In our experiments, we fine tune the pre-trained VGG16 model on ImageNet for detection. During the forward pass, we clip the weight into the range $[-b, b]$, quantize the weight using the binary values and obtain the scaling factor α. During the backward pass, full precision weights are required to compute the small gradients through ADAM [16] or stochastic gradient descent. We use the greedy layer-wise method to binarize the deep networks instead of binarizing all the weight at the same time. The greedy layer-wise training procedure is described in Algorithm 1.

2.4. Backward propagation via discretization

The *floor* function isn't continuous and most of the gradients are zeros. It's infeasible to use back propagation via the quantized weights of a layer. A straight-forward and fast estimator of the gradients approach is given as [2]

$$g_{\mathcal{W}^l} = g_{\tilde{\mathcal{W}}^l} \cdot 1_{|r|<1}, \tag{7}$$

where $g_{\tilde{\mathcal{W}}^l}$ denotes the derivative $\frac{\partial L(\alpha,Q)}{\partial \tilde{\mathcal{W}}^l}$, and $g_{\mathcal{W}^l}$ denotes the gradient $\frac{\partial L(\alpha,Q)}{\partial \mathcal{W}^l}$. We adopt the same scheme in this paper to estimate the gradients. It's noticed that the gradient can be back-propagated via the full-precision weights. when the weight are out of the range [-1,1], The gradients would be zeros. The results are found as the same as [2] that if r is beyond the range $[-1,1]$, the performance would degrade a lot. The scheme is described in Algorithm 1.

2.5. Loss function

In our experiments, we joint the classification and detection task. The multi-task loss function as following:

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1] \cdot L_{loc}(t^u, v), \tag{8}$$

Algorithm 1 Traning BDNN. We denote s as the quantization stage numbers, N_i $(i = 1, 2 \ldots s)$ as the number of quantized layers in the ith stage, b as the maximum absolute value in the weights of a layer, C as the network function, $L(\alpha, Q)$ as the loss function for computing the optimized scaling factor α and quantized weight tensors, $\mathcal{W}_i^l \in R^{n^l \times k^l \times k^l}$ as the 32-bit weight tensor of the lth layer, $\tilde{\mathcal{W}}_i^l$ as the clipped weights, $\mathcal{I}$ as the inputs, $\mathcal{A}$ as the network outputs, lr as the learning rate, and η as the learning rate decay factor.

```
1: for $i = 1$ to $s$ do
2:     if $i > 1$ then
3:         freeze the weights $\mathcal{W}_i^l$ in the last quantization stage
4:     end if
5:     $\tilde{\mathcal{W}}_i^l = clip(\mathcal{W}_i^l, -b, b)$
6:     for $k = 1$ to $N_i$ do
7:         $\alpha, Q = L(\alpha, Q) = \min_{\alpha,Q} \left\| \tilde{\mathcal{W}}^l - \alpha Q \right\|^2$
8:         $Q = floor(\tilde{\mathcal{W}}_i^l + \frac{\beta}{2})$
9:         $\alpha = \frac{\tilde{\mathcal{W}}^{l^T} Q}{\|Q\|^2}$
10:    end for
11:    $\mathcal{A} = Forward(\mathcal{I}, \alpha, Q, \tilde{\mathcal{W}}_i^l)$
12:    $g_{\tilde{\mathcal{W}}^l} = Backward(C, \mathcal{A}, \mathcal{I}, \mathcal{W}_i^l)$
13:    $g_{\mathcal{W}^l} = g_{\tilde{\mathcal{W}}^l} 1_{|r|<1}$
14:    $\mathcal{W}_{i+1}^l = \mathcal{W}_i^l + lr * g_{\mathcal{W}_i^l}$
15:    $lr = lr * \eta$
16: end for
```

where the first loss L_{cls} is the log loss for the classification task, the second term L_{loc} is for the location task, and $[u \geq 1]$ is an indicator function. For the location task, the regression loss is:

$$L_{loc}(t^u, v) = \sum_{i \in x,y,w,h} smooth_{L_1}(t^u - v_i). \tag{9}$$

We use smooth-L1 loss function to regress the target bounding boxes where $v = (v_x, v_y, v_w, v_h)$ is the tuple of the target bounding box, and $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$ is the tuple of prediction a bounding box for class u,

$$
\begin{aligned}
&t_y = \frac{x - x_a}{w_a}, t_y = \frac{y - y_a}{h_a} \\
&t_w = \log\left(\frac{w}{w_a}\right), t_h = \log\left(\frac{h}{h_a}\right) \\
&t^*_x = \frac{x^* - x_a}{w_a}, t^*_y = (y^* - y_a)/h_a) \\
&t^*_w = \log\left(\frac{w^*}{w_a}\right), t^*_h = \log\left(\frac{h^*}{h_a}\right) \qquad (10)
\end{aligned}
$$

where x, y, w and h are the center coordinates, width, and height of the bounding box. x, x_* and x_a are the corresponding variables for predicted bounding boxes, anchors, and the ground truth (also for y, w, h).

3. Experimental results

3.1. Datasets

We evaluate our algorithm on PASCAL VOC 2007 and 2012 detection datasets. PASCAL VOC 2007 includes about 5000 training and validation images, and 5000 testing images over 21 classes. The PASCAL VOC 2012 comprises 10,000 test images, we present the performance of the split testing results by using Mean Average Precision (mAP). We use mxnet to implement our algorithm.

3.2. Comparison of two binarization methods

We explore two binary methods for BDNN in Section 2.2. In our experiments, we use Fast RCNN with VGG16 as the base model. The full convolution network VGG16 is cast from the original classification network [22]. We transform the inner-product layer fc6 and fc7 in the original classification network into convolution layers. The convolution operations to fc6 implement $\times 7$ filters with stride 1 following conv5_3, and convolution operations to fc7 are converted into 1×1 filters with stride 1. The results are illustrated in Table 1.

We adopt stochastic gradient descent (SGD) to train the Fast RCNN. Each mini-batch comprises two random images. Each image has 64 region proposals. Proposals are selected as foreground objects if the IOU with the ground truth is greater than 0.5.

In PASCAL VOC 2007 dataset, our BDNN has a comparable result with mAP 58.6%, 8.3% lower than the 32-bit network. On PASCAL VOC 2012, our BDNN obtains mAP 58.0%, 7.7% lower than the 32-bit network. The greedy layer-wise method achieves 58.6% mAP on PASCAL VOC 2007, outperforming the whole-quantize method by 18.1%. Results on VOC 2007 and 2012 are illustrated in Table 1 and Table 2 respectively. It's verified that if we binarize weights of the whole network at the same time. The errors would be accumulated layer by layer and making it hard to converge. Classification and location losses are illustrated in Fig. 2. It's found that the classification losses and location losses in greedy layer-wise approach are smaller and converge much faster than the whole-quantized method.

3.3. Faster RCNN

Faster RCNN joints region proposal generation and detection process. It selects proposals and detects objects simultaneously. The region proposal network and detection network share the bottom layers and low-level features. It breaks the bottlenecks that

Table 1
Results of mAP(%) on Pascal VOC 2007. Whole-quantized means we quantize the weights of the detection network at the same time. 07trainval represents PASCAL VOC 2007 training and validation sets. †Fast RCNN results are prepared by the authors of [10].

Method	Train set	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Mbike	Persn	Plant	Sheep	Sofa	Train	Tv	mAP
Fast RCNN†	07train+val	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
Whole-quantize	07train+val	53.9	50.2	26.3	29.7	10.7	56.4	61.2	49.5	18.1	37.5	43.2	33.8	62.2	55.3	43.4	12.4	21.6	42.6	59.9	42.7	40.5
Greedy layer-wise stage1	07train+val	69.8	74.3	52.6	48.3	30.2	76.4	75.2	75.6	36.4	72.4	65.8	71.1	78.8	73.7	65.3	28.5	59.5	67.8	74.7	58.1	62.7
Greedy layer-wise stage2	07train+val	64	70.8	46.2	43.1	24.8	74.2	73.6	69.6	30.8	68.2	62.8	63.8	75.1	73.4	61.1	27.6	55.4	62.4	69.2	55.3	58.6

Table 2
Results of mAP(%) on Pascal VOC 2012. †Fast RCNN results are prepared by the authors of [10]. 12train+val denotes the training and validation sets of PASCAL VOC 2012.

Method	Train set	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Mbike	Persn	Plant	Sheep	Sofa	Train	Tv	mAP
Fast RCNN†	12train+val	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
Greedy layer-wise stage1	12train+val	77.8	72.3	60.0	44.5	31.2	73.4	65.2	84.3	36.9	62.0	51.6	80.9	73.9	76.1	66.0	25.3	62.3	60.8	74.7	57.8	61.9
Greedy layer-wise stage2	07train+val	76.4	69.2	54.1	39.9	26.6	69.4	62.2	81.9	31.7	58.0	47.2	77.5	71.1	71.4	63.3	23.6	56.6	66.6	71.1	52.8	58.0

Table 3
Results of mAP(%) on Pascal VOC 2007. 0712train+val denotes the union of training and validation sets of PASCAL VOC 2007 and 2012. †Faster RCNN and SSD results are presented by the authors of [28] and [21].

Method	Train set	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Mbike	Persn	Plant	Sheep	Sofa	Train	Tv	mAP
32-bit Faster RCNN†	07train+val	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6	69.9
1-bit Faster RCNN	12train+val	66.3	72.3	55.4	48.9	35.7	71.7	77.5	71.5	43.2	66.3	66.3	65.8	78.2	72.2	72.8	34.9	56.4	62.7	72.4	64.1	62.6
32-bit SSD	0712train+val	75.2	79.8	70.5	62.5	41.3	81.1	80.8	86.4	51.5	74.3	72.3	83.5	84.6	80.6	74.5	46.0	71.4	73.8	83.0	69.1	72.1
1-bit SSD	0712train+val	63.2	73.2	57.5	52.1	29.3	72.1	74.8	78.0	42.0	60.0	65.2	73.0	77.5	72.6	68.4	37.2	62.3	68.1	76.0	62.8	63.3

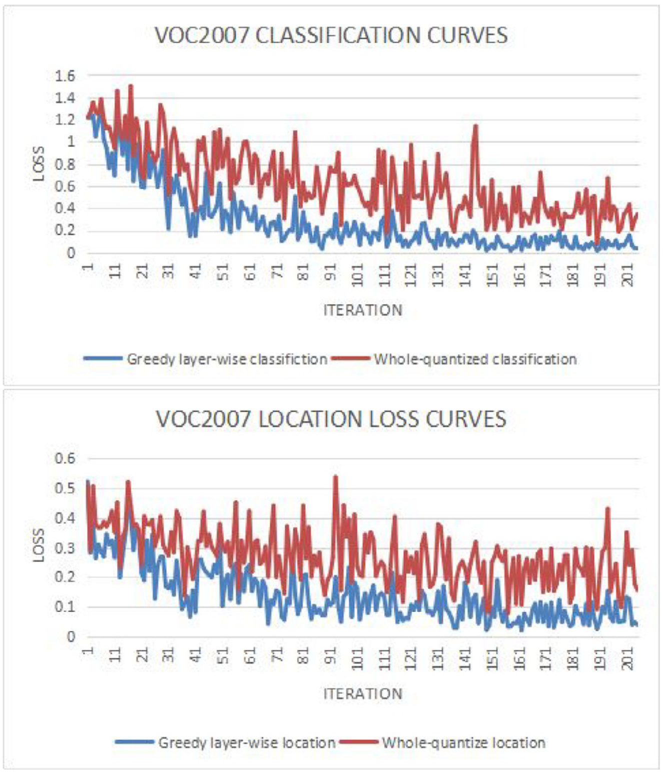


Fig. 2. Classification and detection loss of two binarization methods.

region proposal generation needs lots of computation. In the experiment, we also take the greedy layer-wise method to binarize the weight of the bottom layers. For region proposal, the region proposal network works as the attention mechanism role for the detection task. The region proposal network over shared layers is kept with full-precision. For detection network, considering the outputs of the network are desired to be continuous variables, the location and classification layers aren't binarized. It degrades less than Fast RCNN as region proposal and detection is trained end-to-end with binary weights.

In our implementation, we use a mini-batch size of 256. For λ, we set the default value by 1 to balance the two losses. Other implementation details are as the same as [28]. Results are illustrated in Table 3. Faster RCNN degrades 7.3% mAP compared to the full precision. Furthermore, our method can achieve up to 10.49 $\times$ on CPU compared to the full-precision Faster RCNN.

3.4. SSD

SSD [21] removes object proposals in Faster RCNN. It predicts the bounding box at multi-scales and aspects per feature map. It's a much faster detection network compared to region proposal methods with competitive performance. We adopt the same method to binarize weights in SSD. In our experiment, we binarize weight of all the layers with exception of the first layer, classification and location layer. Results are illustrated in Table 3. Our binary detection network degrades mAP by 8.8% than the full precision network. The experiment shows the effectiveness of our proposed algorithm. Besides, our methods on SSD can achieve 9.86 $\times$ speed-up compared to the full-precision.

3.5. Compression

BDNN can save lots of memories compared to the full-precision detection network while achieving comparable performances. As illustrated in Table 4, our BDNN saves the memory storage significantly. Tiny YOLO [27] is a smaller and faster network than normal YOLO network. Compared with tiny YOLO, our BDNN has better results and smaller size. On PASCAL VOC 2007 test, our binarized SSD outperforms Tiny YOLO by 6.1%.

3.6. Comparison with other methods

We have compared our BDNN with the full-precision network. It's necessary to compare our methods with other binary network to show the effectiveness of our method. Here we carry out another ablation study with two methods: BinaryNet[26] and XNOR-Net[5]. We use mxnet to implement their methods and compress SSD. The results are in Table 5 and demonstrate more clearly that our method outperforms [26] and [5].

3.7. Visual results

We present quality results performed by our BDNN on PASCAL VOC 2007. Predicted bounding boxes are marked with score. Fig. 3 shows the results of SSD and Fig. 4 demonstrates the results of Faster RCNN. Even if the weight of the detection networks are binarized, BDNN can still detect objects of different scales and categories. Visual results indicate the effectiveness of our BDNN and greedy-layer wise training scheme.

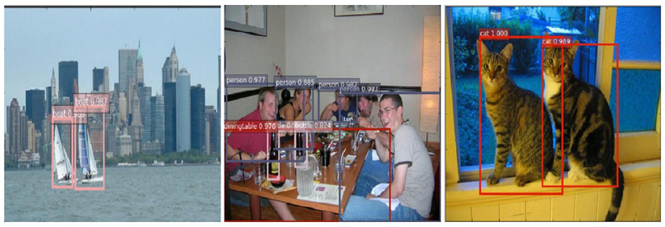


Fig. 3. Detection examples with Faster RCNN.

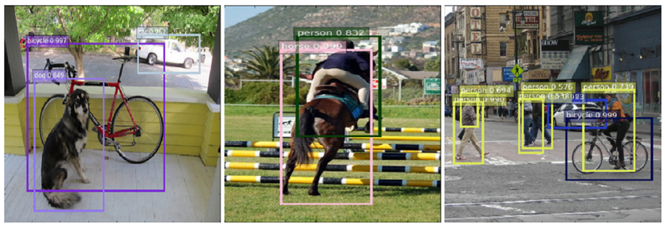


Fig. 4. Detection examples with Faster SSD.

Table 4
Comparison of model size.

Methods	Faster RCNN		SSD		YOLO	
	Mode size(MB)	mAP	Model size(MB)	mAP	Model size(MB)	mAP
32-bit	528	69.9	93.0	72.1	40.7	57.1
1-bit	26.4	62.6	8.36	63.3	-	-

Table 5
Comparison with state-of-art methods.

Methods	BinaryNet		XNOR-Net		Our methods	
	Mode size(MB)	mAP	Model size(MB)	mAP	Model size(MB)	mAP
1-bit	10.47	59.04	9.28	60.71	8.36	63.36

4. Conclusion

In this paper, we introduce a method to binarize weights for fast object detection. BDNN is trained by the greedy layer-wise approach which outperforms the whole-quantized method by a large margin. We evaluate our method on different detection networks. Experiments indicate the effectiveness of our proposed greedy layer-wise method. Deep neural networks can be compressed via our BDNN scheme significantly. Future work can be extended to better weight quantization methods to improve the performance.

Acknowledgments

This work is supported by National Natural Science Foundation of China (U1713203) and Shenzhen Science and Technology Innovation Committee (Project KQJSCX 20180330170238897).

References

[1] D. Bahdanau, J. Chorowski, D. Serdyuk, Y. Bengio, et al., End-to-end attention-based large vocabulary speech recognition, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 4945–4949.
[2] Y. Bengio, N. Léonard, A. Courville, Estimating or propagating gradients through stochastic neurons for conditional computation, arXiv:1308.3432 (2013).
[3] C. Bucilu, R. Caruana, A. Niculescu-Mizil, Model compression, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 535–541.
[4] W. Chen, J.T. Wilson, S. Tyree, K.Q. Weinberger, Y. Chen, Compressing neural networks with the hashing trick, CoRR, arXiv:1504.04788 (2015).
[5] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized neural networks: training deep neural networks with weights and activations constrained to+ 1 or-1, 2016,
[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.
[7] Y. Duan, J. Lu, J. Feng, J. Zhou, Context-aware local binary feature learning for face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 40 (5) (2018) 1139–1153.
[8] Y. Duan, J. Lu, Z. Wang, J. Feng, J. Zhou, Learning deep binary descriptor with multi-quantization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1183–1192.
[9] M. Figurnov, D. Vetrov, P. Kohli, Perforatedcnns: Acceleration through elimination of redundant convolutions, arXiv:1504.08362 (2015).
[10] R. Girshick, Fast r-cnn, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.
[11] S. Han, H. Mao, W.J. Dally, Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding, CoRR, arXiv:1510.00149 2 (2015).
[12] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, in: Proceedings of the European Conference on Computer Vision, Springer, 2014, pp. 346–361.
[13] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, arXiv:1512.03385, (2015).
[14] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv:1503.02531 (2015).
[15] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, Quantized neural networks: Training neural networks with low precision weights and activations, arXiv:1609.07061 (2016).
[16] D. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980 (2014).
[17] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1097–1105.
[18] X. Li, Z. Liu, P. Luo, C.C. Loy, X. Tang, Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade, arXiv:1704.01344 (2017).
[19] D.D. Lin, S.S. Talathi, V.S. Annapureddy, Fixed point quantization of deep convolutional networks, arXiv:1511.06393 (2015).
[20] X. Lin, C. Zhao, W. Pan, Towards accurate binary convolutional neural network, in: Advances in Neural Information Processing Systems, 2017, pp. 345–353.
[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, SSD: Single shot multibox detector, arXiv:1512.02325 (2015).
[22] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.
[23] P. Luo, Z. Zhu, Z. Liu, X. Wang, X. Tang, Face model compression by distilling knowledge from neurons, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016.
[24] M.-T. Luong, H. Pham, C.D. Manning, Effective approaches to attention-based neural machine translation, arXiv:1508.04025 (2015).
[25] A. Novikov, D. Podoprikhin, A. Osokin, D.P. Vetrov, Tensorizing neural networks, in: Advances in Neural Information Processing Systems, 2015, pp. 442–450.
[26] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, arXiv:1603.05279 (2016).
[27] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.
[28] S. Ren, K. He, R. Girshick, J. Sun, Faster R-Cnn: Towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, 2015, pp. 91–99.
[29] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556 (2014).
[30] H. Wen, S. Zhou, Z. Liang, Y. Zhang, D. Feng, X. Zhou, C. Yao, Training bit fully convolutional network for fast semantic segmentation, arXiv:1612.00212 (2016).
[31] H. Yan, Learning discriminative compact binary face descriptor for kinship verification, Pattern Recognit. Lett. 117 (2019) 146–152.
[32] X. Zhang, J. Zou, K. He, J. Sun, Accelerating very deep convolutional networks for classification and detection (2015).
[33] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, Y. Zou, DOREFA-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, arXiv:1606.06160 (2016).