Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

How to Train a Compact Binary Neural Network with High Accuracy?

Wei Tang,[1*] **Gang Hua,**[4] **Liang Wang**[1,2,3]
[1]Institute of Automation, Chinese Academy of Sciences (CASIA)
[2] Center for Excellence in Brain Science and Intelligence Technology, CAS
[3]University of Chinese Academy of Sciences
[4]Microsoft Research, Beijing, China
{wei.tang, wangliang}@nlpr.ia.ac.cn ganghua@microsoft.com

Abstract

How to train a binary neural network (BinaryNet) with both high compression rate and high accuracy on large scale datasets? We answer this question through a careful analysis of previous work on BinaryNets, in terms of training strategies, regularization, and activation approximation. Our findings first reveal that a low learning rate is highly preferred to avoid frequent sign changes of the weights, which often makes the learning of BinaryNets unstable. Secondly, we propose to use PReLU instead of ReLU in a BinaryNet to conveniently absorb the scale factor for weights to the activation function, which enjoys high computation efficiency for binarized layers while maintains high approximation accuracy. Thirdly, we reveal that instead of imposing L2 regularization, driving all weights to zero which contradicts with the setting of BinaryNets, we introduce a regularization term that encourages the weights to be bipolar. Fourthly, we discover that the failure of binarizing the last layer, which is essential for high compression rate, is due to the improper output range. We propose to use a scale layer to bring it to normal. Last but not least, we propose multiple binarizations to improve the approximation of the activations. The composition of all these enables us to train BinaryNets with both high compression rate and high accuracy, which is strongly supported by our extensive empirical study.

Introduction

Research on binary neural network, which binarizes both the weights and activations of a full precision neural network, has been an active topic (Kim and Smaragdis 2016; Lin et al. 2015; Courbariaux and Bengio 2016; Rastegari et al. 2016; Zhou et al. 2016). The BinaryNet has demonstrated to be quite effective as it can

- compress the network by $32\times$ theoretically, when compared with full precision networks in float point;
- reduce the computation and accelerate the inference speed since the 32-bit floating point multiply accumulations can be replaced by 1-bit *xnor-popcnt* operations.

Recent methods along this line, such as BNN (Courbariaux and Bengio 2016), XNOR-net (Rastegari et al. 2016) and DoReFa-net (Zhou et al. 2016), have achieved promising results. However, just as shown in Table 1, there is still a huge hurdle to put these methods in practice due to the limited compression rate and accuracy on large scale datasets such as ImageNet (Russakovsky et al. 2015).

Table 1: Comparison of different methods on ImageNet dataset. The compression rate (denoted as Comp. rate in the table) considers the overall parameters of the network. Both top-5 and top-1 accuracies are presented in the last column.

Methods	Model size	Comp. rate	Accuracy(%)
AlexNet	232 MB[1]	$1\times$	80.2/56.6
BNN	22.6 MB	$10.3\times$	50.4/27.9
XNOR-net	22.6 MB	$10.3\times$	69.2/44.2
DoReFa-net	22.6 MB	$10.3\times$	– /49.8[2]
Our Method	1.23 MB	$23.6\times$	75.6/51.4

For the issue of compression rate, to make the network easier to train and guarantee reasonable accuracy, all previous methods kept both the first and last layers in full precision. Of course when the trained model is used as a feature extractor, the compression rate is not affected. But once it is utilized for classification in real scenarios, the overall compression rate is severely degraded just as shown in Table 1.

For the issue of accuracy, previous works (Rastegari et al. 2016; Zhou et al. 2016) have attempted to directly apply BNN to large datasets such as ImageNet, but only poor performance is obtained. Since then, much attention has been paid to relaxing the binary constraints, such as introducing scale factors (Rastegari et al. 2016) or using more bits with activations (Zhou et al. 2016). Although obvious accuracy gain is achieved, a careful analysis on the failure of BNN on large datasets has been largely overlooked. We argue that clearly figuring out the reason why BNN failed to obtain high accuracy on large datasets is important because it can guide us to design better training strategies for BinaryNets.

In this paper, we simultaneously address the low compression rate and the low accuracy of previous works on

*This work was done when W. Tang was an intern at MSR supervised by G. Hua.

[1]Model size of AlexNet is 249 MB reported in previous work, our result is calculated based on the configuration in Caffe.

[2]The accuracy with activations of 2 bits is adopted for fair comparison because we approximate the activations two times.

training BinaryNets on large scale datasets through a careful analysis of why these previous works failed to address these two issues, with respect to training strategies, regularization, and activation approximation. Figure 1 shows what improvement can be brought through our work.

To obtain higher compression rate, we first conduct a careful analysis of the dramatic accuracy drop in DoReFa-net (Zhou et al. 2016) when binarizing the last layer. We found that this is caused by the extreme large variation of the output from the last layer after binarization. As a result, we propose to add an adaptive scale parameter after binarizing the last layer, which can effectively boost the compression rate without hurting the accuracy. Then, a newer and smaller baseline model is adopted to further improve the overall compression rate. By combining these two together, we achieve a compression rate of 189$\times$ compared to AlexNet and a model size of only 1.23 MB.

To obtain higher accuracy, we start by analyzing why BNN performs poorly on large dataset such as ImageNet. We found that this is mainly due to improper training strategies, which manifests simple but very effective training strategies for the BinaryNet that improve the accuracy by more than 22% in our experiments. We further propose multiple binarizations in addition to modifying the network architecture to further improve the accuracy of the BinaryNet.

Our contributions in this work are three-fold:

- we revisit and analyze the poor performance of BNN on large datasets and failure cases of binarizing the last layer in binary networks, and propose effective training strategies for the BinaryNet;
- we propose multiple binarization and present step-by-step on how to train a BinaryNet with both high compression rate and high accuracy at the same time;
- our final binarized model has an amazing size of only 1.23 MB, while achieves the current state-of-the-art accuracy on the ImageNet dataset among all the BinaryNets.

Related Work

Inspired by the sweeping win in the image classification challenge (Krizhevsky, Sutskever, and Hinton 2012), deep Convolutional Neural Network (DCNN) has made quantum leap in many areas of computer vision, including object detection (Sermanet et al. 2013; Girshick et al. 2014; Girshick 2015; Ren et al. 2015), semantic segmentation (Long, Shelhamer, and Darrell 2015; Chen et al. 2015), and so on.

However, when it comes to deploying the state-of-the-art CNN-based visual models towards real-world systems, the large memory consumption and computing demand are often unbearable for common computing platforms such as portable or wearable devices. To solve this issue, substantial research efforts are made and various approaches have been proposed recently. These include matrix decomposition (Denil et al. 2013; Denton et al. 2014; Lebedev et al. 2014), weight quantization (Sun and Lin 2016; Gupta et al. 2015; Gong et al. 2014), network pruning (Liu et al. 2015; Han et al. 2015; Han, Mao, and Dally 2016) etc.

Among these methods, a number of methods on training BinaryNets (Kim and Smaragdis 2016; Courbariaux, Bengio, and David 2015; Lin et al. 2015; Courbariaux and Bengio 2016; Rastegari et al. 2016; Zhou et al. 2016) have been proposed which foster the development of BinaryNets improving on both compression rate and accuracy. Here we mainly focus our discussion on methods in which both weights and activations are binary-valued, since this kind of network not only reduces the memory storage, but also is more computational efficient.

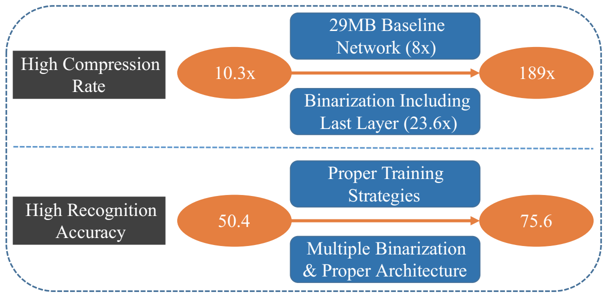
High Compression Rate
10.3x
29MB Baseline Network (8x)
Binarization Including Last Layer (23.6x)
189x
High Recognition Accuracy
50.4
Proper Training Strategies
Multiple Binarization & Proper Architecture
75.6

Figure 1: In terms of both compression rate and accuracy, the gains that can be achieved by our proposed method.

To the best of our knowledge, BNN (Courbariaux and Bengio 2016) is the first work that produces a purely BinaryNet by simultaneously binarizing both the weights and the activations, and achieves comparable accuracy to full precision network on datasets such as CIFAR-10 and CIFAR-100. But previous works (Rastegari et al. 2016; Zhou et al. 2016) showed that poor accuracy is obtained when applying BNN directly to large datasets such as ImageNet. In this paper, we reveal that the poor accuracy is due to improper training strategies. If effective training strategies are adopted, the accuracy can be significantly improved.

XNOR-net (Rastegari et al. 2016) proposes to improve the performance of BNN with a better approximation by introducing scale factors for both weights and activations during binarization. However, the extra full precision convolution is needed for calculating the scale factors of the activations, which makes the computation of convolution layers very complex. In order to mitigate this issue, we propose an alternative that removes the scale factor both in weights and activations, and replace the ReLU activation function with PReLU, which does not increase the parameter size but is simpler and more efficient than XNOR-net.

DoReFa-net (Zhou et al. 2016) aims at boosting the performance by approximating the activations with more bits. Even though they claimed that the last layer could be binarized as well, they suffered severe accuracy drop when doing so. This forces them to instead adopt full precision in the last layer to guarantee good accuracy. However, this made their model compression rate to be largely limited when taking the overall parameter size into consideration. We conduct a detailed analysis on this problem and propose to solve it by adopting a scalar layer after the last binarized layer.

In this paper, we focus on how to train BinaryNets with both high compression rate and high accuracy. We emphasize that proper training strategies are actually as important as how to do better approximation, which was largely over-

Table 2: Results under different learning rates and activation functions. In this table, the last layer is in full precision.

Methods	Learning rate	Activation function	Comp. rate	Accuracy(%)
BNN	0.01	ReLU	6.02×	43.5/20.9
BNN	0.0001	ReLU	6.02×	61.3/36.4
BNN	0.0001	PReLU	5.98×	65.6/41.2

looked by the previous works. Moreover, we solve the problem of binarizing the last layer and propose multiple binarizations. As a result, a 1.23 MB model is achieved with an accuracy outperforming the state-of-the-art method.

Our Approach

In the section, we present in detail how we train a BinaryNet with both high compression rate and accuracy. We first revisit and carefully examine the training strategies for BNN (Courbariaux and Bengio 2016), which was largely overlooked by previous works. We show that BNN can achieve quite reasonable results with proper training strategies even without complex approximation methods. Then we elaborate on how to solve the bottlenecks limiting the compression rate and the accuracy.

Effective training strategies for BinaryNet

As the experiment in (Rastegari et al. 2016) showed, training a BNN in the same way as training a full precision network always achieves poor accuracy. This implies that BinaryNet has its own properties such that directly adapting conventional training method is not desirable. Through our careful examination, we identify that there are three factors that are critical for BinaryNet training, *i.e.*, the learning rate, the scale factor, and the regularizer. These factors seem to be natural, but we argue that special considerations need to be taken for effectively training BinaryNets with high accuracy.

The learning rate. The initial learning rate for full precision DCNN is commonly proposed to be 0.01. Facilitated by batch normalization (Ioffe and Szegedy 2015), the network can be trained with a much larger learning rate, *e.g.*, 0.05.

But the situation is quite different for BNN. As shown in Figure 3(a) and Table 2, when we set the learning rate to 0.01, the accuracy curve is fluctuating and the final top-5 accuracy is only 43.5%, far from the accuracy with full precision. So why this happens? Our examination starts from how binarization is conducted. Recent work on binarization follows a deterministic way proposed by (Courbariaux and Bengio 2016), *i.e.*,

$$w^b = \begin{cases} +1 & w \geq 0, \\ -1 & otherwise. \end{cases} \quad (1)$$

This indicates that only the signs of real-valued weights are important in BinaryNets. So we conjecture that if a high learning rate is adopted, there might be too many weights that frequently change signs, which introduces instability to the network training.

Table 3: Results of adopting full precision or binarized last layer. In this table, The initial learning rate is 0.0001 and the activation function is PReLU.

Methods	Last layer	Scale layer	Comp. rate	Accuracy(%)
BNN	Full	No	5.98×	65.6/41.2
BNN	Binary	No	27.1×	61.0/35.8
BNN	Binary	Yes	27.1×	64.6/39.3

To consolidate our analysis, we have the statistics on how many weights are changing their signs under different learning rates during BNN and full precision network training. The results are shown in Figure 2. It can be observed that, under the same learning rate of 0.01, the sign changes for BNN is nearly 3 orders of magnitude larger than that of a full precision network. Only when the learning rate of BNN is lowered to 0.0001, the two results become close.

Hence we conclude that a lower learning rate is more preferred for training BNN to avoid frequent sign changes. Our experiments show that when the learning rate is lowered to 0.0001, a tremendous accuracy gain is obtained and BNN can even achieve comparable results with XNOR-net. This finding manifests that the initial learning rate for BNN should be far less than that of full precision network if we want to guarantee high accuracy. (Rastegari et al. 2016) trained BNN with an initial learning rate of 0.1, so only a top-5 accuracy of 50.4% was obtained on ImageNet.

The scale factor. To improve the representation, the XNOR-net (Rastegari et al. 2016) introduces real-valued scale factors during binarization. For weights with shape of (N, C, H, W), where N, C, H, and W denote the numbers of the output and input, and the height and width of the kernels respectively, each (C, H, W) shares a scale factor and there are N scale factors in one layer. For activation, it becomes quite complex. Firstly, a matrix $\mathbf{A}$ is calculated by averaging the absolute values of elements in the input feature map $\mathbf{I}$ across the channels, *i.e.*, $\mathbf{A} = \frac{\sum \|\mathbf{I}_{:,:,i}\|}{c}$. Then, the matrix $\mathbf{A}$ is convolved by a kernel $\mathbf{k} \in \mathbb{R}^{w \times h}$ with $\mathbf{k}_{ij} = \frac{1}{w \times h}$ to obtain the scale matrix for the activation $\mathbf{K}$.

This method does improve the accuracy. However, it makes the convolution procedure complex and inefficient due to the way it calculates the scale factors for weights and activations. To mitigate this issue, we propose an elegant alternative, where both weights and activations are directly binarized without any scale factor according to Eq(1), but a PReLU (He et al. 2015) layer is utilized instead of a ReLU layer as the activation function. This means that we omit the scale factors for activations and move the scale factors for weights to the activation function.

This modification brings two benefits. On one hand, this ensures that the convolution layers can all be carried out purely by *xnor-popcnt* operations, thus being computational efficient. On the other hand, it improves the accuracy compared with BNN with no scale factor. Table 2 shows that our method improves the top-5 accuracy by 4.3%.

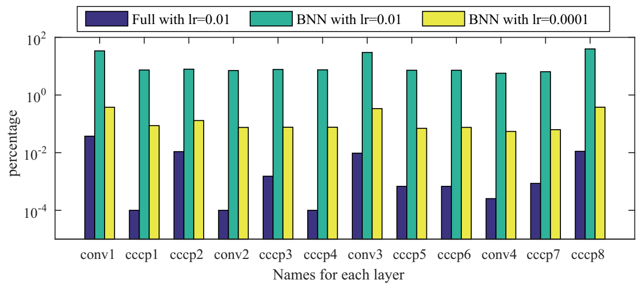

Figure 2: The percentages of sign change in each layer with different learning rates. The y axis is shown in log.

The regularizer. A good regularization term is essential for a deep neural network to obtain robust generalization capacity. In a full precision DCNN, a L_2 regularization term, which tends to decrease the magnitude of the weights and helps prevent overfitting, is widely adopted.

However, the L_2 regularization is contradictory to what a binary network wants to achieve. In a binary network, the real-valued weights are favored to be close to $+1$ or -1, because less binarization error will be introduced when weights are binarized as Eq 1. So if a L_2 regularization term is adopted when training BNNs, which favors to drive the weights to near zero, the loss often jitters heavily or even fails to converge as shown in the red curve of Figure 3(c).

To address this problem. we propose another kind of regularization term which is suitable for a BinaryNet. It takes the following form, *i.e.*,

$$J(\mathbf{W}, \mathbf{b}) = L(\mathbf{W}, \mathbf{b}) + \lambda \sum_{l=1}^{L} \sum_{i=1}^{N_l} \sum_{j=1}^{M_l} (1 - (\mathbf{W}_{l,ij})^2) \quad (2)$$

where L denote the number of layers and N_l and M_l denote the dimensions of the weight matrix in the lth layer. The $L(\mathbf{W}, \mathbf{b})$ is the task-related loss term such as a softmax loss in the classification task. The second term is a regularization term and the parameter λ controls the relative importance of the two terms. It can be seen that, in this formulation, the weights are piloted to 1 or -1 other than to 0 as in a L_2 regularization, therefore it aligns with what a BinaryNet favors. Just as shown in Figure 3(c), the loss curve becomes smooth and stable after adding the proposed regularization term in Eq 2.

Training BinaryNets with high compression

To make the training easier and guarantee reasonable performance, in previous work (including BNN, XNOR-net and DoReFa-net), both the first layer and the last layer are kept in full precision. If the model is used as a feature extractor, the compression rate is not affected. But once it is used for classification, the whole compression rate is severely degraded as shown in Table 1. This degradation is mainly due to the last layer which often occupies large number of parameters.

Why directly binarizing the last layer fails. However, when the last layer is changed from full precision to binary, not only the network becomes very difficult to train, but also the performance drops a lot. This problem has been observed by previous work. To obtain a high compression rate, DoReFa-net has tried to binarize the last layer, but as shown in Table 4, obvious accuracy drop is observed.

Table 4: Comparison between our method and DoReFa-net after binarizing the last layer. For fair comparison, activations of our method are also binarized with 2 bits.

Methods	Last layer	Scale layer	Comp. rate	Accuracy(%)
DoReFa-net	Full	No	$10.3\times$	$-$ /47.1
DoReFa-net	Binary	$-$	$31.4\times$	$-$ /40.3
BNN	Binary	Yes	$27.1\times$	70.4/45.8

Table 5: Results before and after multiple binarizations. A denotes activation.

Methods	Bits of A	Comp. rate	Accuracy(%)
BNN	1	$27.1\times$	64.6/39.3
BNN	2	$27.1\times$	70.4/45.8
BNN (expand)	2	$23.6\times$	75.6/51.4

The significantly decreased accuracy is due to the extremely large variation range of the output after binarizing the last layer. Denote N to be the input channels of the last layer. Once both the inputs and the weights of the last layer are binarized, the range of the output value is $[-N, N]$. Typically, N is set to 1024 in our baseline network and to 4096 in AlexNet. If the output with such large variation range is fed into the softmax function, which is often used for image classification task, it is extremely easy to step into the saturation region. Therefore, the training becomes much more difficult especially when it is done with a large learning rate.

Improving the overall compression rate. To improve the compression rate and reduce the model size of BinaryNet to be applicable to real applications, we propose to add a scale layer before feeding the output of the binarized last layer into the softmax function. The scale layer has only a scalar parameter which is learnable and initialized to be a small value (0.001 in our experiment). As illustrated in Table 3, this improves the compression rate from $6.02\times$ to $27.1\times$ in NIN-net with only 1% top-5 accuracy drop compared with the counterpart network with the last layer in full precision, which demonstrates the effectiveness of the proposed strategy. It should be noted that adding a scale layer after the binarized last layer has also been adopted by (Wu 2016), but this strategy is not the only factor for achieving high compression rate in our binary network, which will be further demonstrated in Section 4.

Training BinaryNets with high accuracy

With strategies proposed above, not only the recognition accuracy of the BinaryNet is tremendously improved from 50.4% reported in XNOR-net to 64.6%, but also the compression rate is significantly improved from $6.02\times$ to $27.1\times$.

But for accuracy, there is still a large margin between the

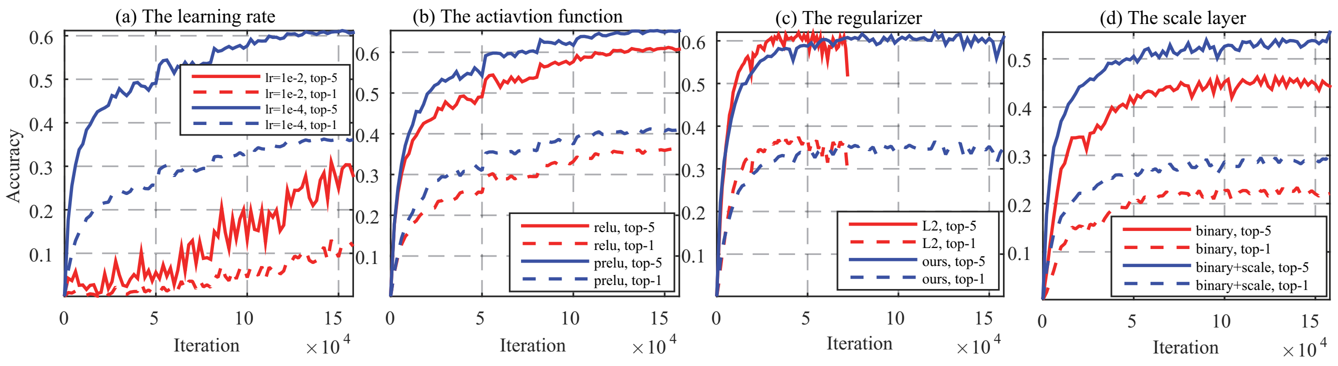

Figure 3: The effectiveness of our proposed strategies on training BNN. (a) The learning rate. (b) The activation function. (c) The regularizer. (d) The scale layer.

accuracy of state-of-the-art BinaryNets and the accuracy of full precision networks. To further improve the accuracy, we first propose multiple binarizations, which can significantly improve the accuracy but without sacrificing the compression rate. Then we propose to properly expand the baseline network with only minimal compression rate drop.

Multiple binarizations. In BNN, both the input activations $\mathbf{A}_{l-1}$ and weights $\mathbf{W}_l$ of the lth layer are directly binarized, *i.e.*, $\mathbf{H}_{l-1} = sign(\mathbf{A}_{l-1})$ and $\mathbf{B}_l = sign(\mathbf{W}_l)$. Here, $\mathbf{H}_{l-1}$ and $\mathbf{B}_l$ are the binarized activations and weights, respectively. To improve the accuracy, but maintain the same compression rate as in BNN, we propose to apply multiple binarizations for activations.

It is carried as follows. For $i = 1$, $\mathbf{H}_{l,1}$ is the sign of $\mathbf{A}_l$ and $\alpha_{l,1}$ is the average absolute value of A_l. For $i > 1$, $H_{l,i}$ and $\alpha_{l,i}$ is calculated in the same way but based on the residue approximation error from step $i-1$, i.e., $\mathbf{E}_{l,i} = \mathbf{A}_l - \sum_{j=1}^{i-1} \alpha_{l,j} * \mathbf{H}_{l,j}$. Thus, $\mathbf{A}_l$ is approximated as follows

$$\mathbf{A}_l \approx \sum_{i=1}^{m} (\alpha_{l,i} \mathbf{H}_{l,i}), \tag{3}$$

where m is the number of times that the activations are binarized. So after multiple binarizations, to take the full connected layer as an example, the output $\mathbf{O}_l$ is calculated as

$$\begin{aligned} \mathbf{O}_l &= \mathbf{W}_l \cdot \mathbf{A}_{l-1} \\ &\approx \textstyle\sum_{i=1}^{m} (\alpha_{l-1,i} \textit{xnor-popcnt}(\mathbf{B}_l, \mathbf{H}_{l-1,i})), \end{aligned} \tag{4}$$

where the $\cdot$ denotes the matrix multiplication and *xnor-popcnt*$(\cdot,\cdot)$ denotes the *xnor-popcnt* operation between two binarized matrices with which BinaryNet can not only get a high compression rate, but also obtain high inference speed compared with a full precision network.

The results before and after multiple binarizations of activations are presented in Table 5, it can be observed that when we adopt $m = 2$, the top-5 accuracy boosts by 5.8%.

Notably, although multiple binarizations can boost the accuracy, it also increases the computation complexity by m times. So we adopt $m = 2$ to strike for the right balance between the inference speed and the accuracy.

DoReFa-net (Zhou et al. 2016) also adopts a similar strategy by using m bits for activations, which is suitable for fixed-point computation such as on FPGA devices, on which numbers in arbitrary bits can be conveniently represented. However, our method is preferable to general computation platforms such as on Intel CPU, GPU, and ARM.

Boosting accuracy with minimal memory overhead. Another alternative to improve the accuracy is to modify the network itself. It is reasonable because the representation capacity of the baseline network is obviously shrank after binarization. However, how to modify the network with minimal memory overhead while boosting the performance remains an issue.

As a guideline, (Han et al. 2015) has shown that parameters in the lower layers are less redundant, and hence less likely be pruned during sparsification. This implies that, to enhance the representation capacity of BinaryNets, parameters in lower layers should be expanded. One big advantage with this design is that lower layers often has less parameters than higher layers in a network, so expanding the lower layers will cause less compression rate drop.

Inspired by this finding, we propose an expanded version of the baseline network, which markedly improves the accuracy while still maintains a high compression rate of $23.6\times$. As shown in Table 5, a 5.4% accuracy gain is achieved with a sacrifice of compression rate from $27.1\times$ to $23.6\times$.

Algorithm 1 demonstrates the whole procedure for training a BinaryNet with all the strategies we proposed.

Experiments

Our method is evaluated in terms of compression rate and accuracy. To measure the compression rate, we take all the parameters in the network including parameters in the first, the last, and the activation function layers into consideration. We measure the accuracy by performing the image classification task on the large-scale ImageNet dataset.

Our proposed training strategies and methods improving the compression rate and the accuracy are universal and can be applied to any kinds of DCNN architecture. To pursue a high compression rate, we adopt NIN-net (Lin, Chen, and Yan 2013), a much smaller network than the networks used in previous works, as the base model in our experiment. We compare our method with three recent works on BinaryNets:

Table 6: Comparison of different methods on the ImageNet dataset. The compression rates inside/outside brackets are the results compared with the baseline/AlexNet model. bef. and aft. denote the model size before and after binarization, respectively.

Methods	Base model	Bits of A	Last layer	Comp. rate	Model size (bef.)	Model size (aft.)	Accuracy(%)
BNN	AlexNet	1	Full	$10.3\times$	232MB	22.6MB	50.4/27.9
XNOR-net	AlexNet	1	Full	$10.3\times$	232MB	22.6MB	69.2/44.2
XNOR-net	ResNet-18	1	Full	$70\times(13.4\times)$	44.6MB	3.34MB	73.2/51.2
DoReFa-net	AlexNet	2	Full	$10.3\times$	232MB	22.6MB	– /49.8
Our method	AlexNet	2	Binary	$31.2\times$	232MB	7.43MB	71.1/46.6
Our method	NIN-net	2	Binary	$189\times(23.6\times)$	29MB	1.23MB	75.6/51.4

BNN (Courbariaux and Bengio 2016), XNOR-net (Rastegari et al. 2016), and DoReFa-net (Zhou et al. 2016).

Implementation details

For the pre-processing of ImageNet dataset, firstly, all the images are resized to 256×256. Then, before sending images to the network, we randomly crop 224×224 image patches with mean subtraction and randomly flipping. No other data augmentation tricks such as contrast adjustment and multi-scale is adopted. We hold out part of training images for hyper-parameter tuning and the final model is evaluated on the validation dataset with only single center crop.

For the hyper-parameters, unless otherwise specified, the initial learning rate is set to 0.0001 and divided by 2 once the training loss stops decreasing. The parameter λ is set to 5×10^{-7} and the batch size is set to 256.

Just as previous works did, batch normalization layer is used before each binary convolution layer and ADAM is used as the solver. We keep the first layer in full precision because it has negligible influence on the compression rate but is essential for high accuracy. We implement our work on Caffe (Jia et al. 2014).

Given the limited space available, how the overall compression rates presented in the tables of this paper are calculated will be illustrated in detail in our supplementary material, only the final results are given here.

Image Classification Results on ILSVRC2012

Table 6 shows the final results of our method compared with previous works with different methods and different models. Our result is based on NIN-net, a 4-layer Network in Network model with a model size of 29MB and sightly better results than AlexNet. But we have modified the network as described in Section 3.3 by expanding the output channels of the first two *cccp* layers from 96 to 128 and the kernel size of the first four *cccp* layers from 1×1 to 3×3.

Our method achieves a compression rate of $189\times$ compared with AlexNet and $23.6\times$ compared with the full precision NIN-net model, which results in a compressed model of only 1.23MB. These results are mainly attributed to the scale layer we propose, which successfully solves the problem of binarizing the last layer and improves the compression rate from $6.02\times$ to $23.6\times$, and the small baseline model we choose which contributes $8\times$ compared with AlexNet.

Algorithm 1 Training a $L-$layers BinaryNet.

```
Require: a minibatch of inputs and targets $(A_0, Y^*)$, previous weights $W$, previous PReLU parameters $P$, previous BatchNorm parameters $\theta$, regularization term coefficient $\lambda$, and previous learning rate $\eta$.
Ensure: updated weights $W^{t+1}$, updated PReLU parameters $P^{t+1}$, updated BatchNorm parameters $\theta^{t+1}$.
  1. Computing the parameters gradients:
  1.1 Forward propagation:
  for $l := 1$ to $L$ do
      $B_l \leftarrow sign(W_l)$
      $A^b_{l-1} \leftarrow A_{l-1}$                # $A^b_l = \sum_{i=1}^m(\alpha_{l,i}H_{l,i})$
      $O_l \leftarrow \sum_{i=1}^m(\alpha_{l-1,i}xnor - popcnt(B_l, H_{l-1,i}))$
      $S_l \leftarrow PReLU(O_l, P_l)$
      $A_l \leftarrow BatchNorm(S_l, \theta_l)$
      if $l = L$ then
          $Y_L \leftarrow Scale(A_L)$              # $scale\ layer$
          $C \leftarrow Loss(Y_L, Y^*)$
      end if
  end for
  1.2 Backward propagation:
  Compute $g_{A_L} = \frac{\partial C}{\partial A_L}$ knowing $A_L$ and $Y^*$
  for $l := L$ to $1$ do
      $(g_{S_l}, g_{\theta_l}) \leftarrow BackBatchNorm(g_{A_l}, S_l, \theta_l)$
      $(g_{O_l}, g_{P_l}) \leftarrow BackPReLU(g_{S_l}, O_l, P_l)$
      $g_{W_l} \leftarrow g_{O_l} A^{b\intercal}_{l-1}$
      $g_{A_{l-1}} \leftarrow (B^\intercal_l g_{O_l}) \circ 1_{|A_{l-1}| \preceq 1}$
  end for
  2. Accumulating the parameters gradients:
  for $l := 1$ to $L$ do
      $\theta^{t+1}_l \leftarrow Update(\theta_l, \eta, g_{\theta_l})$     # $using\ Adam\ solver$
      $P^{t+1}_l \leftarrow Update(P_l, \eta, g_{P_l})$
      $W^{t+1}_l \leftarrow Clip(Update(W_l, \eta, g_{W_l}), -1, 1)$
  end for
```

In terms of accuracy, our method outperforms previous state-of-the-art methods and even is much better than the newest ResNet-18 model by 2.4% with superior compression rate. This credits to the effective strategies we propose for BinaryNet training and the multiple binarization which approximates the full precision activations more accurately.

It should be noted that DoReFa-net also reported a top-1 accuracy of 53.0% on ImageNet with AlexNet, but this is achieved by approximating the activations with 4 bits, training BinaryNet with last layer in full precision, and initializing the network with full precision model. However, not pursuing a single high indicator, our method aims at having a good balance among three practical aspects: the accuracy, the compression rate, and the computational efficiency.

Conclusion and Future Work

In this paper, we address the problem of how to train a compact BinaryNet with high accuracy on large scale dataset. Firstly, we reveal that low learning rate, the scale factor, and better regularization are three critical factors to guarantee high accuracy and computational efficiency for BinaryNets. Moreover, we solve the problem of severe accuracy drop after binarizing the last layer, which is essential for high compression rate. Finally, we propose multiple binarizations to further improve the accuracy of BinaryNets. The final result shows that the proposed method achieves the state-of-the-art compression rate and accuracy.

Our future work on this topic will focus on proposing better optimizing method for BinaryNet training to further improve the accuracy.

Acknowledgments

This work is jointly supported by National Key Research and Development Program of China (2016YFB1001000), National Natural Science Foundation of China (61525306, 61633021, 61420106015) and Foundation of Beijing (4162058).

References

Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L. 2015. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*.

Courbariaux, M., and Bengio, Y. 2016. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*.

Courbariaux, M.; Bengio, Y.; and David, J.-P. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, 3123–3131.

Denil, M.; Shakibi, B.; Dinh, L.; de Freitas, N.; et al. 2013. Predicting parameters in deep learning. In *NIPS*, 2148–2156.

Denton, E. L.; Zaremba, W.; Bruna, J.; LeCun, Y.; and Fergus, R. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 1269–1277.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Girshick, R. 2015. Fast r-cnn. In *ICCV*.

Gong, Y.; Liu, L.; Yang, M.; and Bourdev, L. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; and Narayanan, P. 2015. Deep learning with limited numerical precision. *CoRR, abs/1502.02551* 392.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *NIPS*, 1135–1143.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ICLR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 1026–1034.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Kim, M., and Smaragdis, P. 2016. Bitwise neural networks. *arXiv preprint arXiv:1601.06071*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

Lebedev, V.; Ganin, Y.; Rakhuba, M.; Oseledets, I.; and Lempitsky, V. 2014. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*.

Lin, Z.; Courbariaux, M.; Memisevic, R.; and Bengio, Y. 2015. Neural networks with few multiplications. *arXiv preprint arXiv:1510.03009*.

Lin, M.; Chen, Q.; and Yan, S. 2013. Network in network. *arXiv preprint arXiv:1312.4400*.

Liu, B.; Wang, M.; Foroosh, H.; Tappen, M.; and Pensky, M. 2015. Sparse convolutional neural networks. In *CVPR*, 806–814.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *CVPR*, 3431–3440.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. *arXiv preprint arXiv:1603.05279*.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *IJCV* 115(3):211–252.

Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; and LeCun, Y. 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

Sun, F., and Lin, J. 2016. Memory efficient nonuniform quantization for deep convolutional neural network. *arXiv preprint arXiv:1607.02720*.

Wu, X. 2016. High performance binarized neural networks trained on the imagenet classification task. *arXiv preprint arXiv:1604.03058*.

Zhou, S.; Ni, Z.; Zhou, X.; Wen, H.; Wu, Y.; and Zou, Y. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*.