Training Binary Weight Networks via Semi-Binary Decomposition

Qinghao Hu[1,2][0000−0001−9458−0760], Gang Li[1,2][0000−0001−7835−4739], Peisong Wang[1,2][0000−0002−6384−0280], Yifan Zhang[1,2][0000−0002−9190−3509], and Jian Cheng[1,2,3]✉[0000−0003−1289−2758]

[1] National Laboratory of Pattern Recognition,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
{qinghao.hu, gang.li, peisong.wang, yfzhang, jcheng}@nlpr.ia.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Center for Excellence in Brain Science and Intelligence Technology, Beijing, China

Abstract. Recently binary weight networks have attracted lots of attentions due to their high computational efficiency and small parameter size. Yet they still suffer from large accuracy drops because of their limited representation capacity. In this paper, we propose a novel semi-binary decomposition method which decomposes a matrix into two binary matrices and a diagonal matrix. Since the matrix product of binary matrices has more numerical values than binary matrix, the proposed semi-binary decomposition has more representation capacity. Besides, we propose an alternating optimization method to solve the semi-binary decomposition problem while keeping binary constraints. Extensive experiments on AlexNet, ResNet-18, and ResNet-50 demonstrate that our method outperforms state-of-the-art methods by a large margin (5 percentage higher in top1 accuracy). We also implement binary weight AlexNet on FPGA platform, which shows that our proposed method can achieve $\sim 9\times$ speed-ups while reducing the consumption of on-chip memory and dedicated multipliers significantly.

Keywords: Deep Neural Networks · Binary Weight Networks · Deep Network Acceleration and Compression.

1 Introduction

Deep convolutional neural networks have become more and more popular since AlexNet [16] made a success in ILSVRC2012. After that, convolutional neural networks have shown significant improvements on a variety of computer vision tasks such as image classification [16], object detection [24], image segmentation [21], and so on. However, the great performance of deep networks comes at the cost of large parameter size and high computational complexity. For applications on mobile phones or embedded devices, it's difficult to deploy deep networks on them due to their limited computation and storage resources.

To alleviate these problems, a lot of methods have been proposed, such as pruning [10, 11, 19], low-rank decomposition [6, 13, 15, 17, 25, 30] and fixed-point

quantization [8, 9, 20, 22, 28]. Binary quantization, a special case of fixed-point quantization, represents the weights of deep networks via only binary values. As there are only binary values in the quantized weights, multiplication operations can be replaced with addition operations. Thus binary quantization can not only achieve high ($32\times$) compression ratio, but also speed up the deep networks. Besides, binary weight networks are more efficient on field-programmable gate array (FPGA), digital signal processor (DSP), and the deep learning accelerator (DLA). On these architectures, binary weight networks usually can achieve higher speed-ups and save more hardware resources. Due to the appealing properties of binary quantization, many binary weight networks have been proposed, such as BC [4], BWN [23], SQ-BWN [7], and so on.

However, state-of-the-art binary weight networks suffer from significant accuracy drop due to their limited representation capacity. Convolutional kernels in BC [4] have only binary patterns, and all the parameters' magnitude equals to 1. This severely lowers down the diversity of convolutional kernels. BWN [23] multiplies each binary convolutional kernel by a different scale factor to approximate the full-precision convolutional kernel, then each convolutional kernel has a different magnitude. But parameters in the same convolutional kernel still share the same magnitude, which limits the representation power of convolutional kernels.

In order to increase the representation capacity of binary weight networks, we propose a novel semi-binary decomposition method which decomposes a matrix into two binary matrices and a diagonal matrix. Besides, we propose an alternating optimization method to learn the decomposition factors with binary constraints. Extensive experiments on ImageNet show that our proposed method outperforms state-of-the-art algorithms. Our main contributions can be summarized as the follows:

- Inspired by that the matrix product of binary matrices has more numerical possibilities than binary matrix, we propose a novel semi-binary decomposition method to train binary weight networks. By using proposed semi-binary decomposition, our binary weight networks have more representation capacity than state-of-the-art methods.
- Since learning the semi-binary decomposition factors is difficult, here we propose an alternating optimization method to solve semi-binary factors while still keeping the binary constraints.
- Extensive experiments are conducted on ImageNet to evaluate our methods. The experiments results on AlexNet, ResNet-18, and ResNet-50 demonstrate that our proposed method outperforms state-of-the-art algorithms by a large margin. In addition, we implement binary weight AlexNet on FPGA platform, and the experiment result shows that our binary weight networks can achieve $\sim 9\times$ speed-ups using less on-chip memory and hardware multipliers.

2 Related Work

In recent years, a lot of methods [3] have been proposed to compress or accelerate deep networks. Most of these methods fall into three categories: pruning-based methods, low-rank decomposition based methods, and quantization-based methods.

2.1 Pruning-based methods

Pruning-based methods compress the deep networks by removing unimportant connections. Early works of pruning [11, 19] use the second derivative of loss functions to determine which connections are unimportant. Recently Han *et al.* [10] propose a three-step method to compress the deep networks. They first prune those unimportant connections, then quantize the remaining weights via K-means, and finally encode the quantized weights using Huffman coding. During the inference phase, a decoder is required to reconstruct the weights, which makes their method inconvenient. Besides, above methods can hardly utilize the Basic Linear Algebra Subprograms (BLAS) since they prune weights in an unstructured way. To cure this problem, Lebedev *et al.* [18] propose the Group-wise Brain Damage. By imposing the group-sparsity regularizer, the weights are pruned in a group-wise fashion. As a result, convolutions can be reduced to multiplications of thinned dense matrices, and they still can use BLAS library to get higher speed-ups.

2.2 Low-rank decomposition based methods

Low-rank decomposition based methods [5, 6, 13] mainly use matrix or tensor decomposition methods to decompose convolutional kernels into several small matrices or tensors. Denton *et al.* propose to use Singule Value Decomposition (SVD) to reduce the computational complexity [6]. Instead of directly approximating the weights, Zhang *et al.* [30] propose to approximate the layer response via a low-rank matrix. Besides, their method also takes the non-linear layers' responses into account. Lebedev *et al.* [17] propose to use CANDECOMP/PARAFAC (CP) decomposition to approximate the convolutional kernels. They only apply their method on a single layer of AlexNet. Similar like CP-decomposition, Tucker decomposition is also used to accelerate the convolutional layers [15]. Differently, Tucker decomposition can be used to compress the whole network while CP decomposition can not. Wang *et al.* [25] propose to use Block Term Decomposition to speed up the convolutional layers. The Block Term Decomposition can be regarded as a compromise between CP-decomposition and Tucker decomposition. Novikov *et al.* propose to use the Tensor-Train format to compress the fully-connected layers of deep networks. Their method can achieve up to 7 $\times$ compression ratio on VGG16 network.

2.3 Quantization based methods

Vector quantization has a long history in data compression. This technique is introduced into network compression by Gong *et al.* . They [8] propose to use vector quantization to compress the fully-connected layers of CNNs. Following this line, Wu *et al.* [28] [2] propose an product quantization based algorithm to simultaneously speed up the computation and reduce the parameter size. Another kind of quantization method is low-bit fixed-point quantization. Gupta *et al.* [9] propose to quantize the weights to fixed-point format via a stochastic rounding scheme instead of deterministic rounding scheme. By using this method, deep networks can be quantized with 16-bit fixed-point numbers with little degradation of accuracy. Wang *et al.* [26] proposed the fixed-point factorized network which decomposes the weights into two fixed-point matrix and one diagonal matrix. As a special case of fixed-point quantization, binary quantization aims to quantize the weights into binary values. Courbariaux *et al.* [4] proposed BinaryConnect to train binary weight networks. Like [9], they used a stochastic binarization scheme instead of deterministic scheme. Since binary values have limited representation capacity, Rastegari *et al.* [23] propose to approximate full-precision convolutioal kernels with binary kernels and a scaling factor. By multiplying a scaling factor, binary kernels have lower quantization loss than directly binary quantization. Dong *et al.* [7] propose a stochastic quantization scheme. In each iteration, they only quantize a portion of parameters to low-bit with a stochastic probability inversely proportional to the quantization error and the remaining parameters stay unchanged with full-precision. Hu *et al.* [12] proposed to train binary weight network from the view of hashing, which learns binary weights using inner-product preserving hashing methods. Wang *et al.* [27] proposed a two-step quantization methods which decomposing the network quantization problem into code learning and transformation function learning step.

3 Our method

In this section, we propose the semi-binary decomposition to increase the representation capacity of binary weight networks. Then an alternating optimization method is proposed to solve the semi-binary decomposition problem. Finally, we analyse the time and space complexity of the proposed binary weight networks in the inference phase.

3.1 Preliminary

Given an L-layer pre-trained CNN model, let $\mathbf{W} \in \mathbb{R}^{T \times S}$ be the full-precision weights of l^{th} layer. To quantize the weights $\mathbf{W}$ into a binary matrix $\mathbf{B}$, a simple binarization method [4] is:

$$\mathbf{B} = sgn(\mathbf{W}) \tag{1}$$

where sgn denotes the sign function, and $sgn\,(x)$=1 for $x > 0$ and -1 otherwise. Simple binarization has limited representation capacity because $\mathbf{B}$ has only *binary patterns*. Thus direct binarization will result in significant quantization loss. Rastegari *et al.* [23] propose to multiply a scale factor α_i for each binary convolutioal kernel $\mathbf{B_i} \in \mathbb{R}^{1\times S}$, and the objective function is :

$$\begin{aligned} \min\ & L(\Lambda, \mathbf{B}) = \|\mathbf{W} - \Lambda\mathbf{B}\|_F^2 \\ s.t.\quad & \mathbf{B} \in \{+1, -1\}^{T\times S} \end{aligned} \tag{2}$$

where $\Lambda \in \mathbb{R}^{T\times T}$ is a diagonal matrix and $\alpha_i = \Lambda_{\mathbf{ii}}$ is the scaling factor for $\mathbf{B_i}$. Different convolutional kernels in [23] have different magnitudes, thus it has better representation power. Yet multiplying a scaling factor for each binary convolutional kernels still suffers from large quantization loss because parameters in the same convolutional kernels has the same magnitude α_i.

3.2 Semi-Binary Decomposition

Since current binary quantization methods have limited representation capacity, here we aims to find better quantization methods to increase the parameter's diversity. In this paper, we propose a novel semi-binary decomposition method which approximates a matrix by the matrix product of two binary matrices and a diagonal matrix, thus the diversity of approximate matrix is higher than binary matrix. Specifically, the proposed semi-binary decomposition can be formulated as :

$$\begin{aligned} \min\ & L(\mathbf{U}, \mathbf{D}, \mathbf{V}) = \|\mathbf{W} - \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}\|_F^2 \\ s.t.\quad & \mathbf{U} \in \{+1, -1\}^{T\times K} \\ & \mathbf{V} \in \{+1, -1\}^{S\times K} \end{aligned} \tag{3}$$

where $\mathbf{D} \in \mathbb{R}^{K\times K}$ is a diagonal matrix, $K \leq min(S, T)$, $\mathbf{U}$ and $\mathbf{V}$ are binary matrix. The proposed semi-binary decomposition is quite suitable for compressing the deep networks because $\mathbf{D}$ has lower computational complexity and $\mathbf{U}$ and $\mathbf{V}$ are still binary matrix. Besides, by using semi-binary decomposition, the representation capacity of binary weight networks is enhanced. Let $\mathbf{W'}$ be the approximate matrix of $\mathbf{W}$ via semi-binary decomposition, then $\mathbf{W'} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}} = \sum_{k=1}^{K} d_k \mathbf{U_k}\mathbf{V_k}^{\mathrm{T}}$ where $d_k = \mathbf{D_{kk}}$, $\mathbf{U_k}$ and $\mathbf{V_k}$ are the k^{th} column of matrix $\mathbf{U}$ and $\mathbf{V}$ respectively. For any parameter $\mathbf{W'_{i,j}}$ in $\mathbf{W'}$, its magnitude has 2^K possibilities while parameter in BC [4] and BWN-like methods [23] [1] [7] has only 2 and T possibilities respectively. Thus the proposed semi-binary decomposition method can improve the representation capacity.

Eq. (3) is hard to solve due to the binary constraints, here we learn the components in a greedy way. Let $\mathbf{W_k}$ be k-term approximation of semi-binary decomposition, then $\mathbf{W_k} = \sum_{i=1}^{k} d_i \mathbf{U_i}\mathbf{V_i}^{\mathrm{T}}$. Let $\mathbf{R_k}$ be the residual matrix after $k-1$ terms of approximation, then $\mathbf{R_k} = \mathbf{W} - \mathbf{W_{k-1}}$ and $\mathbf{R_1} = \mathbf{W}$. In each step,

we learn the k^{th} term via approximating the residual matrix $\mathbf{R_k}$, the objective function is formulated as:

$$\begin{aligned} \min \ & L(\mathbf{U_k}, d_k, \mathbf{V_k}) = \|\mathbf{R_k} - d_k \mathbf{U_k} {\mathbf{V_k}}^{\mathrm{T}}\|_F^2 \\ s.t. \quad & \mathbf{U_k} \in \{+1, -1\}^{T \times 1} \\ & \mathbf{V_k} \in \{+1, -1\}^{S \times 1} \end{aligned} \tag{4}$$

To solve Eq. (4), we propose an alternating optimization method i.e. iteratively update one decomposition factor with other factors fixed.

Update d_k with fixed $\mathbf{U_k}$ and $\mathbf{V_k}$: Given fixed $\mathbf{U_k}$ and $\mathbf{V_k}$, the objective function can be reformulated as:

$$\min \ L(d_k) = -2 d_k {\mathbf{U_k}}^{\mathrm{T}} \mathbf{R_k} \mathbf{V_k} + TS \cdot d_k^2 \tag{5}$$

The optimal solution of above equation is:

$$d_k = \frac{1}{T * S} {\mathbf{U_k}}^{\mathrm{T}} \mathbf{R_k} \mathbf{V_k} \tag{6}$$

Update $\mathbf{U_k}$ with fixed $\mathbf{V_k}$ and d_k: Given fixed $\mathbf{V_k}$, we replace d_k with its optimal solution, then the objective function is transformed as :

$$\begin{aligned} \max \ & L(\mathbf{U_k}) = \frac{({\mathbf{U_k}}^{\mathrm{T}} \mathbf{R_k} \mathbf{V_k})^2}{\|\mathbf{U_k}\|_F^2 \|\mathbf{V_k}\|_F^2} = ({\mathbf{U_k}}^{\mathrm{T}} \mathbf{R_k} \mathbf{V_k})^2 \\ & s.t. \quad \mathbf{U_k} \in \{+1, -1\}^{T \times 1} \end{aligned} \tag{7}$$

The optimal solution for above equation is

$$\mathbf{U_k} = sgn(\mathbf{R_k} \mathbf{V_k}) \tag{8}$$

Update $\mathbf{V_k}$ with fixed $\mathbf{U_k}$ and d_k: similar like updating $\mathbf{U_k}$, the optimal solution for $\mathbf{V_k}$ is:

$$\mathbf{V_k} = sgn({\mathbf{R_k}}^{\mathrm{T}} \mathbf{U_k}) \tag{9}$$

Until now, we have described the optimization algorithm of semi-binary decomposition for one layer. For the whole network quantization, we use the semi-binary decomposition for each layer's weights. This method is denoted as SBD-Direct and the overall training algorithm is summarized in Algorithm 1.

3.3 Featuremap-Oriented Semi-Binary Factors

Directly decomposing $\mathbf{W}$ for all layers of deep networks via semi-binary decomposition has two drawbacks. First, because the weights is multiplied by the input featuremap in the forward propagation, the binary quantization error will be amplified by the input featuremap. Second, directly applying semi-binary decomposition for the whole network can cause large accuracy drop since the quantization error accumulates across multiple layers.

Algorithm 1: Training Binary Weight Networks via SBD-Direct

Input: Pre-trained convolutional neural networks weights $\{\mathbf{W}^l\}_{l=1}^L$ and Max_Iter
Output: Learned binary components $\{\mathbf{U}^l\}_{l=1}^L$,$\{\mathbf{V}^l\}_{l=1}^L$and $\{\mathbf{D}^l\}_{l=1}^L$
for $l = 1; l \leq L$ **do**
  for $k = 1; k \leq K$ **do**
    Update residual matrix R_k
    Initialize $\mathbf{V_k}$ with all-ones matrix
    while *iter* $\leq$ *Max_Iters* **do**
      Update $\mathbf{U_k}$ with Eq.(8)
      Update $\mathbf{V_k}$ with Eq.(9)
    end
    Update d_k with Eq.(6)
  end
end
return $\{\mathbf{U}^l\}_{l=1}^L$, $\{\mathbf{V}^l\}_{l=1}^L$ and $\{\mathbf{D}^l\}_{l=1}^L$;

To cure these problems, here we learn the semi-binary components via minimizing the output featuremap's quantization loss. Let $\mathbf{X}^l \in \mathbb{R}^{S\times N}$ be the l^{th}-layer's input featuremap of full-precision network. Similarly, let l^{th}-layer's input featuremap of quantized network be $\tilde{\mathbf{X}}^l$. Here quantized network means that the first $l-1$ layers have been quantized via semi-binary decomposition, thus $\tilde{\mathbf{X}}^l = \mathbf{U}^{l-1}\mathbf{D}^{l-1}(\mathbf{V}^{l-1})^{\mathrm{T}}\tilde{\mathbf{X}}^{l-1}$. The objective function is formulated as:

$$\begin{aligned} \min\ & L(\mathbf{U}^l, \mathbf{D}^l, \mathbf{V}^l) = \|\mathbf{W}^l\mathbf{X}^l - \mathbf{U}^l\mathbf{D}^l(\mathbf{V}^l)^{\mathrm{T}}\tilde{\mathbf{X}}^l\|_F^2 = \|\mathbf{Y}^l - \sum_{k=1}^{K} d_k^l {\mathbf{U_k}}^l({\mathbf{V_k}}^l)^{\mathrm{T}}\tilde{\mathbf{X}}^l\|_F^2 \\ & s.t. \quad \mathbf{U} \in \{+1,-1\}^{T\times K} \\ & \qquad \mathbf{V} \in \{+1,-1\}^{S\times K} \end{aligned} \tag{10}$$

where $\mathbf{Y}^l = \mathbf{W}^l\mathbf{X}^l$ is the l^{th}- layer's output featuremap. In what follows, we omit the superscript l for convenience. Solving Eq. (10) is difficult due to the binary constraints, here we learn the semi-binary components in a greedy way. Let $\mathbf{Y_k}$ be the k-term approximation of output featuremap, then $\mathbf{Y_k} = \sum_{i=1}^{k} d_i\mathbf{U_i}{\mathbf{V_i}}^{\mathrm{T}}\tilde{\mathbf{X}}$. Let $\mathbf{Z_k}$ be the featuremap's residual matrix after $k-1$ terms of approximation, thus $\mathbf{Z_k} = \mathbf{Y} - \mathbf{Y_{k-1}}$ and $\mathbf{Z_1} = \mathbf{W}$. Then we learn the k^{th} term via approximating the residual matrix $\mathbf{Z_k}$, the objective function is formulated as:

$$\begin{aligned} \min\ & L(\mathbf{U_k}, d_k, \mathbf{V_k}) = \|\mathbf{Z_k} - d_k\mathbf{U_k}{\mathbf{V_k}}^{\mathrm{T}}\tilde{\mathbf{X}}\|_F^2 \\ & s.t. \quad \mathbf{U_k} \in \{+1,-1\}^{T\times 1} \\ & \qquad \mathbf{V_k} \in \{+1,-1\}^{S\times 1} \end{aligned} \tag{11}$$

To solve Eq. (11), we propose an alternating optimization method to update the semi-binary components iteratively.

Update d_k with fixed $\mathbf{U_k}$ and $\mathbf{V_k}$: Given fixed $\mathbf{U_k}$ and $\mathbf{V_k}$, the objective function can be formulated as:

$$\min\ L(d_k) = -2d_k \mathbf{V_k}^{\mathrm{T}} \tilde{\mathbf{X}} \mathbf{Z_k}^{\mathrm{T}} \mathbf{U_k} + d_k^2 \|\mathbf{U_k} \mathbf{V_k}^{\mathrm{T}} \tilde{\mathbf{X}}\|_F^2 \tag{12}$$

The optimal solution of d_k for Eq. (12) is :

$$d_k = \frac{\mathbf{V_k}^{\mathrm{T}} \tilde{\mathbf{X}} \mathbf{Z_k}^{\mathrm{T}} \mathbf{U_k}}{\|\mathbf{U_k} \mathbf{V_k}^{\mathrm{T}} \tilde{\mathbf{X}}\|_F^2} \tag{13}$$

Update $\mathbf{U_k}$ with fixed $\mathbf{V_k}$ and d_k: Given $\mathbf{V_k}$ fixed, we get the following objective by substituting the d_k's optimal solution:

$$max\ L(\mathbf{U_k}) = \frac{(\mathbf{V_k}^{\mathrm{T}} \tilde{\mathbf{X}} \mathbf{Z_k}^{\mathrm{T}} \mathbf{U_k})^2}{\|\mathbf{U_k} \mathbf{V_k}^{\mathrm{T}} \tilde{\mathbf{X}}\|_F^2} = \quad (\mathbf{V_k}^{\mathrm{T}} \tilde{\mathbf{X}} \mathbf{Z_k}^{\mathrm{T}} \mathbf{U_k})^2 \tag{14}$$

Thus the optimal $\mathbf{U_k}$ for above equation is :

$$\mathbf{U_k} = sgn(\mathbf{Z_k} \tilde{\mathbf{X}}^{\mathrm{T}} \mathbf{V_k}) \tag{15}$$

Update $\mathbf{V_k}$ with fixed $\mathbf{U_k}$ and d_k: Given $\mathbf{U_k}$ and d_k fixed, we get the following objective function:

$$\min\ \ L(\mathbf{V_k}) = -2Tr(\mathbf{V_k}^{\mathrm{T}} \mathbf{q}) + \alpha \|\mathbf{V_k}^{\mathrm{T}} \tilde{\mathbf{X}}\|_F^2 \tag{16}$$

where $\mathbf{q} = d_k \tilde{\mathbf{X}} \mathbf{Z_k}^{\mathrm{T}} \mathbf{U_k}$ and $\alpha = d_k^2 \|\mathbf{U_k}\|_F^2$.

Optimizing $\mathbf{V_k}$ for Eq. (16) is still difficult, here we solve $\mathbf{V_k}$ by discrete cyclic coordinate descent method. Specifically, we solve one row of $\mathbf{V_k}$ each time while fixing all other rows. Let v be the j^{th} row of $\mathbf{V_k}$, and $\mathbf{V_k}'$ the column vector of $\mathbf{V_k}$ excluding v. Similarly we denote the j^{th} element of $\mathbf{q}$ as $\mathbf{q_j}$, and let $\mathbf{q}'$ as the $\mathbf{q}$ excluding $\mathbf{q_j}$. Let $\mathbf{x}^{\mathrm{T}}$ be the j^{th} row of matrix $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{X}}'$ be matrix $\tilde{\mathbf{X}}$ excluding $\mathbf{x}^{\mathrm{T}}$. Then problem can be written as:

$$\min\ L(\mathbf{V_k}) = -2vq + 2\alpha \mathbf{V_k}'^{\mathrm{T}} \tilde{\mathbf{X}}' \mathbf{x} v \tag{17}$$

Thus the j^{th} row of $\mathbf{V_k}$ can be updated by:

$$v = sgn(q - \alpha \mathbf{V_k}'^{\mathrm{T}} \tilde{\mathbf{X}}' \mathbf{x}) \tag{18}$$

So far, we have given details of learning semi-binary components by minimizing the featuremap's quantization loss, we denote this method as SBD-FQ and the overall training algorithm of SBD-FQ is summarized in Algorithm 2.

Algorithm 2: Training Binary Weight Networks via SBD-FQ

Input: Pre-trained convolutional neural networks weights $\{\mathbf{W}^l\}_{l=1}^{L}$ and Max_Iter
Output: Learned binary components $\{\mathbf{U}^l\}_{l=1}^{L}$,$\{\mathbf{V}^l\}_{l=1}^{L}$and $\{\mathbf{D}^l\}_{l=1}^{L}$
for $l = 1; l \leq L$ **do**
  Sampling a mini-batch images
  Forward propagation to get $\tilde{\mathbf{X}}^l$ and $\mathbf{X}^l$
  Calculate $\mathbf{Y}$ with $\mathbf{X}^l$ and $\mathbf{W}^l$
  for $i = 1; i \leq N$ **do**
    Update residual matrix $\mathbf{Z_k}$
    Initialize $\mathbf{V_k}$ with all-ones matrix
    while *iter* $\leq$ *Max_Iters* **do**
      Update $\mathbf{U_k}$ with Eq.(15)
      Update d_k with Eq.(13)
      for $j = 1; j \leq S$ **do**
        Update j^{th} element of $\mathbf{V_k}$ with Eq.(18)
      end
    end
  end
end
Fine-tune the binarized CNN model
return $\{\mathbf{U}^l\}_{l=1}^{L}$, $\{\mathbf{V}^l\}_{l=1}^{L}$ and $\{\mathbf{D}^l\}_{l=1}^{L}$;

3.4 Fine-tuning

After direct semi-binary decomposition or minimizing the featuremap's quantization loss, we get the $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{D}$ for each layer. For a convolutional layer with T covolutional kernels of size $c * d * d$. After semi-binary decomposition, we replace the original layer with three layers: a convolutional layer *conv_v*, one scale layer *scale_d*, and a convolutional layer *conv_u*. Layer *conv_v* has K covolutional kernels of size $c * d * d$, layer *conv_u* has T covolutional kernels of size $K * 1 * 1$ and layer *scale_d* has only K parameters.

For the fine-tune stage, we adopt a similar scheme as [4] to maintain the binary values in *conv_v* and *conv_u*. Take *conv_u* layer for example, we adopt a full-precision (32-bit floating) weight matrix $\mathbf{U_f}$ as the proxy of $\mathbf{U}$. $\mathbf{U_f}$ is initialized with $\mathbf{U}$ in the beginning of fine-tuning. In the forward propagation, $\mathbf{U}$ is updated by directly quantizing $\mathbf{U_f}$ to binary value, then $\mathbf{U}$ is used for the forward computation. In the backward propagation, gradients is calculated based on $\mathbf{U}$. The full-precision $\mathbf{U_f}$ is used to accumulate the gradients of weights $\mathbf{U}$.

3.5 Complexity Analysis

In this subsection, we analyse the time and space complexity of our binary weight network in the inference phase. For a convolutional layer with T kernels of size $c * d * d$, let H and W be the height and width of output featuremap

respectively, and let $S = c * d * d$. Let T_m be the time for one multiplication operation, and let T_a be the time for one addition operation. Normally speaking, multiplication operation consumes more time than addition operation, especially for FPGA architecture, thus $T_a \ll T_m$. Since the time and space complexity is highly dependent on K, here we use a hyper-parameter β to control the value of K i.e. let $K = \frac{S*T}{\beta*(S+T)}$. For the experiments in the paper, $\beta = 1$ if not specified.
Time complexity After semi-binary decomposition, the time complexity of layer *conv_v*, *scale_d*, and *conv_u* is $H * W * S * K * T_a$, $H * W * K * T_m$ and $H * W * K * T * T_a$ respectively.
Thus the speed up ratio is:

$$\frac{S * T * (T_m + T_a)}{K(S+T) * T_a + K * T_m} \approx \frac{S * T * (T_m + T_a)}{K(S+T) * T_a} = \frac{\beta(T_m + T_a)}{T_a} \tag{19}$$

Space complexity After semi-binary decomposition, the space complexity of layer *conv_v*, *scale_d*, and *conv_u* is $S * K$, $32K$ and $K * T$ bits respectively.
The compression ratio is :

$$\frac{S * T * 32}{K(S+T) + K * 32} \approx \frac{32 * S * T}{K(S+T)} = 32\beta. \tag{20}$$

For $\beta = 1$, our binary weight networks can achieve $\geq 2\times$ speed-ups and $32\times$ compression ratio. On FPGA platforms, our binary weight networks can achieve higher speed-ups since $T_a \ll T_m$. Table 1 shows that the space and time complexity of our method is less than [23] [1] [7] and nearly equals to [4].

Table 1. Time and Space complexity of state-of-the-art binary weight networks

Method	Time Complexity	Speed-ups	Space Complexity	Compress Ratio
Full-Precision	$S * T * (T_m + T_a)$	1	$32 * S * T$	1
BinaryConnect [4]	$S * T * T_a$	$\frac{T_m + T_a}{T_a}$	$S * T$	32
BWN-like [23] [1] [7]	$S * T * T_a + T * T_m$	$\approx \frac{T_m + T_a}{T_a}$	$S * T + 32T$	≈ 32
Ours($\beta = 1$)	$S * T * T_a + K * T_m$	$\approx \frac{T_m + T_a}{T_a}$	$S * T + 32K$	≈ 32

4 Experiments

In this section, we first give details about training settings, then we compare different methods in terms of quantization loss and classification accuracy. We also implement binary weight AlexNet on FPGA platform, and finally we discuss the effect of different β for semi-binary decomposition.

4.1 Experiment Settings

We implement our method based on the Caffe [14] framework, and experiments are mainly conducted on a GPU Server with 8 Nvidia Titan Xp GPUs.

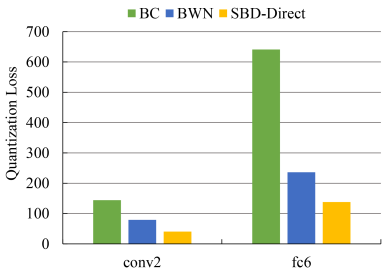

Fig. 1. Binary Quantization Loss via Different Methods

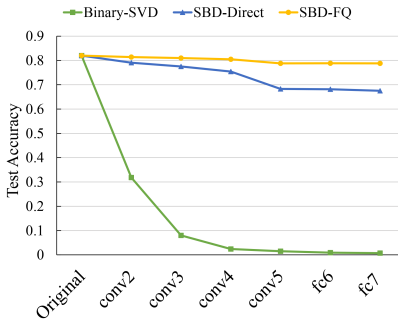

Fig. 2. Top5 accuracy of AlexNet by different quantization methods without fine-tuning

We evaluate our proposed methods on ImageNet2012 with three deep networks i.e. AlexNet, ResNet-18, and ResNet-50. In the proposed alternating optimization method, we set the maximum iterations to 20. For all the experiments in this paper, we train the networks with a SGD solver with momentum=0.9, weight decay=0.0005. As in [1, 23, 31], the first and last layer in the deep networks are still in floating-number format. Following [7, 23], batch normalization layers are used in the AlexNet. We fine-tune AlexNet for 200k iterations with batch-size equals to 256. We set the learning rate to 0.0001 in the beginning, and divide it by 10 after 100k, 150k, and 180k iterations. For ResNet-18, the learning rate starts at 0.0005, and is divided by 10 every 200k iterations. We fine-tune ResNet-18 for 650k iterations with batch size equal to 100. Since fine-tuning ResNet-50 is quite time-consuming, we fine-tune ResNet-50 for only 450K iterations with batchsize=140 by using 7 GPUs. The learning rate is initialized with 0.0001 and divided by 10 every 200k iterations.

4.2 Comparison on Quantization Loss

In this subsection, we compare different binary quantization methods in terms of quantization loss. The quantization loss is defined by Frobenius norm of residual weights between approximate weights and full-precision weights. Here we compare the proposed SBD-Direct with BC [4] and BWN [23]. Figure 1 shows the binary quantization loss of different methods on AlexNet's *conv2* and *fc6* layer. It shows that the proposed method has lower quantization loss than BC [4] and BWN [23], which benefits from the higher representation capacity of semi-binary decomposition.

4.3 Comparison on Learning Methods

In the previous subsection, we have shown that semi-binary decomposition can achieve lower quantization loss than other binary quantization methods, but

which method can learn better semi-binary components has not been discussed. In this subsection, we compare different methods for learning the semi-binary components. Since semi-binary decomposition has a similar form as Singular Vector Decomposition (SVD), a naive method to get the semi-binary components is quantizing the left and right singular vectors to binary values after using SVD for original weight matrix. We denote this method as Binary-SVD.

Figure 2 shows the top5 accuracy of AlexNet after learning semi-binary components via different methods. Here we binarize the weights of AlexNet layer by layer, i.e. *conv4* in the horizontal axis of Figure 2 means that *conv2*, *conv3*, *conv4* are all quantized to binary values. Figure 2 shows that Binary-SVD performs worst among three methods, which means that simply binarizing the singular vectors of SVD can hardly achieve good performance. SBD-Direct still maintains the accuracy after binarizing one or two layers, but it performs worse as more layers are quantized. SBD-FQ aims to minimize the output featuremap's quantization loss, and it performs well even for multiple layers.

4.4 Comparison on Network's Accuracy

To evaluate our proposed method in terms of classification accuracy, we compare our method with BC [4], BWN [23], SQ-BWN [7], and HWGQ-BWN [1]. Table. 2 shows the Top1 and Top5 classification accuracy of AlexNet and ResNet-18 on ImageNet2012 dataset. It's clear that both SBD-Direct and SBD-FQ outperform state-of-the-art methods with a large margin in Top1 and Top5 accuracy. Specifically, our binary ResNet-18 achieves 66.2% top1 accuracy which is 5 percentage higher than state-of-the-art methods.

Table 2. Classification Accuracy of AlexNet and ResNet-18 via different methods

Method	AlexNet		ResNet-18	
	Top1 Acc	Top5 Acc	Top1 Acc	Top5 Acc
Full-Precision	58.5	81.5	69.3	89.2
BinaryConnect [4]	35.4	61.0	-	-
SQ-BWN [7]	51.2	75.1	58.3	81.6
HWGQ-BWN [1]	52.4	75.9	61.3	83.9
BWN [23]	56.8	79.4	60.8	83.0
SBD-Direct (Ours)	**58.0**	**80.3**	**64.9**	**86.4**
SBD-FQ (Ours)	**58.5**	**80.6**	**66.2**	**87.1**

We also evaluate our methods on a more challenging network i.e. ResNet-50. ResNet-50 is deeper than AlexNet and ResNet-18, and it has more 1×1 convolutional kernels. Table 3 reports the Top1 and Top5 accuracy of ResNet-50. After fine-tuning, both SBD-Direct and SBD-FQ outperforms state-of-the-art methods by a large margin (5 percentage in top1 accuracy).
From Table 2 and 3, we can find that SBD-FQ achieves higher accuracy than

SBD-Direct, which shows that minimizing the featuremap's quantization loss is better than direct semi-binary decomposition. But SBD-Direct is faster than SBD-FQ because minimizing the featuremap's quantization loss takes more training time than direct semi-binary decomposition.

Table 3. Classification Accuracy of ResNet-50 via different methods

Method	Classification Accuracy	
	Top1	Top5
Full-Precision	75.2	92.2
BWN [23]	63.9	85.1
SBD-Direct (Ours)	**67.7**	**87.8**
SBD-FQ (Ours)	**68.9**	**88.7**

4.5 Experiments on FPGA

In order to demonstrate the efficiency of our proposed method on hardware acceleration of CNN, we further implement the binary-weight AlexNet on Xilinx Virtex-7 VX485T FPGA platform. The microarchitecture design is based on [29], which is a state-of-the-art CNN accelerator. We quantize the activations of binary-weight AlexNet to 8-bit for the consideration of energy and resource efficiency, and the top1 and top5 accuracy after activation quantization is 58.46% and 80.7% respectively. For fair comparison, we adopt the same platform and working frequency, and restrict the usage of on-chip computing resources (LUTs and FFs) as the same level as in [29].

Table 4 shows the results of our evaluation on the binary-weight AlexNet. It is obvious that our accelerator is 8.78× faster than the floating point counterpart with nearly the same usage of LUTs and FFs. In addition, the consumption of on-chip memory and DSP blocks are drastically reduced due to the weight binarization and low precision representation of activations.

Table 4. Experiment result on FPGA

	Activation	Weight	Resource Utilization				Latency	Speed-ups
			DSP	BRAM	LUT	FF		
Zhang *et al.* [29]	32 bits	32 bits	2240	1024	186251	205704	21.6 ms	1×
Ours	8 bits	1 bit	0	261	211554	303642	2.46 ms	**8.78**×

4.6 The Effect of Different β

Figure 3 shows the top5 accuracy of AlexNet after using the proposed SBD-Direct method with different values of β. With β increasing, we get higher compression ratio but lower accuracy. Besides, we notice that fully-connected layers

is insensitive to the values of β, which means that we can choose larger β for fully-connected layers to achieve higher compression ratio.

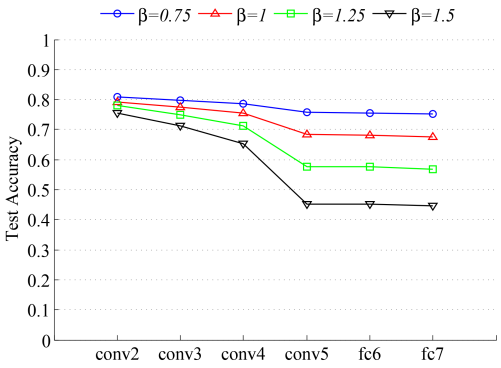

Fig. 3. Top5 accuracy of AlexNet for different β without fine-tuning

5 Conclusion

In this paper, we propose a novel semi-binary quantization method to train the binary weight networks, and we also propose an alternating optimization method to solve the semi-binary decomposition factors under binary constraints. Extensive experiments on ImageNet2012 dataset demonstrate that our methods outperform state-of-the-art methods with a large margin. Experiments on FPGA platform demonstrates that our proposed binary weight networks can achieve nearly 9× speed-ups using less on-chip memory and hardware resources.

6 Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No.61332016, and No.61572500), the Scientific Research Key Program of Beijing Municipal Commission of Education (KZ201610005012), and the Strategic Priority Research Program of Chinese Academy of Science (Grant No.XDBS01000000).

References

1. Cai, Z., He, X., Sun, J., Vasconcelos, N.: Deep learning with low precision by half-wave gaussian quantization. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. pp. 5406–5414. IEEE Computer Society, Honolulu, HI, USA (2017). https://doi.org/10.1109/CVPR.2017.574

2. Cheng, J., Wu, J., Leng, C., Wang, Y., Hu, Q.: Quantized cnn: A unified approach to accelerate and compress convolutional networks. IEEE Transactions on Neural Networks and Learning Systems pp. 1–14 (2017). https://doi.org/10.1109/TNNLS.2017.2774288
3. Cheng, J., Wang, P., Li, G., Hu, Q., Lu, H.: Recent advances in efficient computation of deep convolutional neural networks. Frontiers of IT & EE **19**(1), 64–77 (2018). https://doi.org/10.1631/FITEE.1700789
4. Courbariaux, M., Bengio, Y., David, J.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28. pp. 3123–3131. Montreal, Quebec, Canada (2015)
5. Denil, M., Shakibi, B., Dinh, L., de Freitas, N., et al.: Predicting parameters in deep learning. In: Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 26. pp. 2148–2156. Curran Associates, Inc., Lake Tahoe, Nevada, United States (2013)
6. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27. pp. 1269–1277. Curran Associates, Inc., Montreal, Quebec, Canada (2014)
7. Dong, Y., Ni, R., Li, J., Chen, Y., Zhu, J., Su, H.: Learning accurate low-bit deep neural networks with stochastic quantization. CoRR **abs/1708.01001** (2017), http://arxiv.org/abs/1708.01001
8. Gong, Y., Liu, L., Yang, M., Bourdev, L.: Compressing deep convolutional networks using vector quantization. arXiv preprint arXiv:1412.6115 (2014)
9. Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P.: Deep learning with limited numerical precision. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 37, pp. 1737–1746. PMLR, Lille, France (2015)
10. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural network. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28. pp. 1135–1143. Montreal, Quebec, Canada (2015)
11. Hassibi, B., Stork, D.G.: Second order derivatives for network pruning: Optimal brain surgeon. In: Hanson, S.J., Cowan, J.D., Giles, C.L. (eds.) Advances in Neural Information Processing Systems 5. pp. 164–171. Morgan Kaufmann, Denver, Colorado, USA (1992)
12. Hu, Q., Wang, P., Cheng, J.: From hashing to cnns: Training binary weight networks via hashing. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence. AAAI Press, New Orleans, Louisiana, USA (2018)
13. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. In: Valstar, M.F., French, A.P., Pridmore, T.P. (eds.) British Machine Vision Conference, BMVC 2014. BMVA Press, Nottingham, UK (2014)
14. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the 22Nd ACM International Conference on Multimedia. pp. 675–678. MM '14, ACM, New York, NY, USA (2014). https://doi.org/10.1145/2647868.2654889

15. Kim, Y., Park, E., Yoo, S., Choi, T., Yang, L., Shin, D.: Compression of deep convolutional neural networks for fast and low power mobile applications. CoRR **abs/1511.06530** (2015), http://arxiv.org/abs/1511.06530
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017). https://doi.org/10.1145/3065386
17. Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I.V., Lempitsky, V.S.: Speeding-up convolutional neural networks using fine-tuned cp-decomposition. CoRR **abs/1412.6553** (2014), http://arxiv.org/abs/1412.6553
18. Lebedev, V., Lempitsky, V.S.: Fast convnets using group-wise brain damage. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2554–2564. IEEE Computer Society, Las Vegas, NV, USA (2016). https://doi.org/10.1109/CVPR.2016.280
19. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems 2. pp. 598–605. Morgan Kaufmann, Denver, Colorado, USA (1989)
20. Lin, D., Talathi, S., Annapureddy, S.: Fixed point quantization of deep convolutional networks. In: Balcan, M.F., Weinberger, K.Q. (eds.) Proceedings of The 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 2849–2858. PMLR, New York, New York, USA (2016)
21. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. pp. 3431–3440. IEEE Computer Society (2015). https://doi.org/10.1109/CVPR.2015.7298965
22. Qiu, J., Wang, J., Yao, S., Guo, K., Li, B., Zhou, E., Yu, J., Tang, T., Xu, N., Song, S., Wang, Y., Yang, H.: Going deeper with embedded fpga platform for convolutional neural network. In: Chen, D., Greene, J.W. (eds.) Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. pp. 26–35. FPGA '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2847263.2847265
23. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision – ECCV 2016. pp. 525–542. Springer International Publishing, Cham (2016)
24. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031
25. Wang, P., Cheng, J.: Accelerating convolutional neural networks for mobile applications. In: Proceedings of the 2016 ACM on Multimedia Conference. pp. 541–545. MM '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2964284.2967280
26. Wang, P., Cheng, J.: Fixed-point factorized networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 3966–3974. IEEE Computer Society (2017). https://doi.org/10.1109/CVPR.2017.422
27. Wang, P., Hu, Q., Zhang, Y., Zhang, C., Liu, Y., Cheng, J.: Two-step quantization for low-bit neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society (2018)
28. Wu, J., Leng, C., Wang, Y., Hu, Q., Cheng, J.: Quantized convolutional neural networks for mobile devices. In: IEEE Conference on Computer Vision and Pattern

Recognition (CVPR). pp. 4820–4828. IEEE Computer Society, Las Vegas, NV, USA (2016). https://doi.org/10.1109/CVPR.2016.521
29. Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., Cong, J.: Optimizing fpga-based accelerator design for deep convolutional neural networks. In: Constantinides, G.A., Chen, D. (eds.) Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. pp. 161–170. ACM, Monterey, CA, USA (2015). https://doi.org/10.1145/2684746.2689060
30. Zhang, X., Zou, J., He, K., Sun, J.: Accelerating very deep convolutional networks for classification and detection. IEEE Trans. Pattern Anal. Mach. Intell. **38**(10), 1943–1955 (2016). https://doi.org/10.1109/TPAMI.2015.2502579
31. Zhou, S., Ni, Z., Zhou, X., Wen, H., Wu, Y., Zou, Y.: Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. CoRR **abs/1606.06160** (2016), http://arxiv.org/abs/1606.06160