Projection Convolutional Neural Networks for 1-bit CNNs via Discrete Back Propagation

Jiaxin Gu,[1] Ce Li,[2] Baochang Zhang,[1*] Jungong Han,[3] Xianbin Cao,[1] Jianzhuang Liu,[4] David Doermann[5]
[1]Beihang University, [2]China University of Mining and Technology, Beijing
[3]Lancaster University, [4]Huawei Noah's Ark Lab, [5]University at Buffalo
{jxgu1016, bczhang}@buaa.edu.cn, celi@cumtb.edu.cn

Abstract

The advancement of deep convolutional neural networks (DCNNs) has driven significant improvement in the accuracy of recognition systems for many computer vision tasks. However, their practical applications are often restricted in resource-constrained environments. In this paper, we introduce projection convolutional neural networks (PCNNs) with a discrete back propagation via projection (DBPP) to improve the performance of binarized neural networks (BNNs). The contributions of our paper include: 1) for the first time, the projection function is exploited to efficiently solve the discrete back propagation problem, which leads to a new highly compressed CNNs (termed PCNNs); 2) by exploiting multiple projections, we learn a set of diverse quantized kernels that compress the full-precision kernels in a more efficient way than those proposed previously; 3) PCNNs achieve the best classification performance compared to other state-of-the-art BNNs on the ImageNet and CIFAR datasets.

Introduction

Deep convolutional neural networks (DCNNs) have shown significant ability to learn powerful feature representations directly from image pixels. However, their success has come with requirements for large amounts of memory and computational power, and the practical applications of most DCNNs are limited on smaller embedded platforms and in mobile applications. In light of this, substantial research efforts are being invested in saving bandwidth and computational power by pruning and compressing redundant parameters generated by convolution kernels (Boureau, Ponce, and LeCun 2010; Zhou et al. 2017).

One typically promising method is to compress the representations via approximating floating point weights of the convolution kernels by binary values (Rastegari et al. 2016; Courbariaux et al. 2016; Courbariaux, Bengio, and David 2015). Recently, Local Binary Convolution (LBC) layers have been introduced in (Xu, Boddeti, and Savvides 2016), to approximate the non-linearly activated responses of standard convolutional layers. In (Courbariaux, Bengio, and David 2015), a BinaryConnect scheme using real-valued weights as a key reference is exploited for the binarization process. In (Courbariaux et al. 2016; Rastegari et al. 2016), XNOR-Net is presented where both the kernel weights and inputs attached to the convolution are approximated with binary values, thus allowing an efficient implementation of the convolutional operations. In (Zhou et al. 2016; Lin, Zhao, and Pan 2017), DoReFa-Net exploits bit convolution kernels with low bitwidth parameter gradients to accelerate both training and inference. While ABC-Net (Lin, Zhao, and Pan 2017) adopts multiple binary weights and activations to approximate full-precision weights such that the prediction accuracy degradation can be alleviated. More recently, a simple fixed scaling method incorporated in a 1-bit convolutional layer is employed to binarize CNNs, obtaining closet-to-baseline results with minimal changes (McDonnell 2018). Modulated convolutional networks (MCNs) are presented in (Wang et al. 2018) to merely binarize the kernels, which achieves better results than the baselines.

While reducing storage requirements greatly, these BNNs generally have significant accuracy degradation, compared to those using the full-precision kernels. This is primarily due to the following two reasons. (1) The binarization of CNNs could be essentially solved based on the discrete optimization, but it has long been neglected in previous work. Discrete optimization methods can often provide strong guarantees about the quality of the solutions they find and lead to much better performance in practice (Felzenszwalb and Zabih 2007; Kim et al. 2017; Laude et al. 2018). (2) The loss caused by the binarization of CNNs has not been well studied.

In this paper, we propose a new discrete back propagation via projection (DBPP) algorithm to efficiently build our projection convolutional neural networks (PCNNs) and obtain highly-accurate yet robust BNNs. In the theoretical framework, for the first time, we achieve a projection loss by taking advantage of our DBPP algorithm, i.e., capacity of discrete optimization, on model compression. The advantages of the projection loss also lie in that, on the one hand, it can be jointly learned with the conventional cross-entropy loss in the same pipeline as back propagation; on the other hand, it can enrich diversity and thus improve the modeling capacity of PCNNs. As shown in Fig.1, we develop a generic projection convolution layer which can be easily used in existing convolutional networks, where both quantized kernels and the projection are jointly optimized in an end-to-end manner.

[*]Baochang Zhang is the corresponding author.

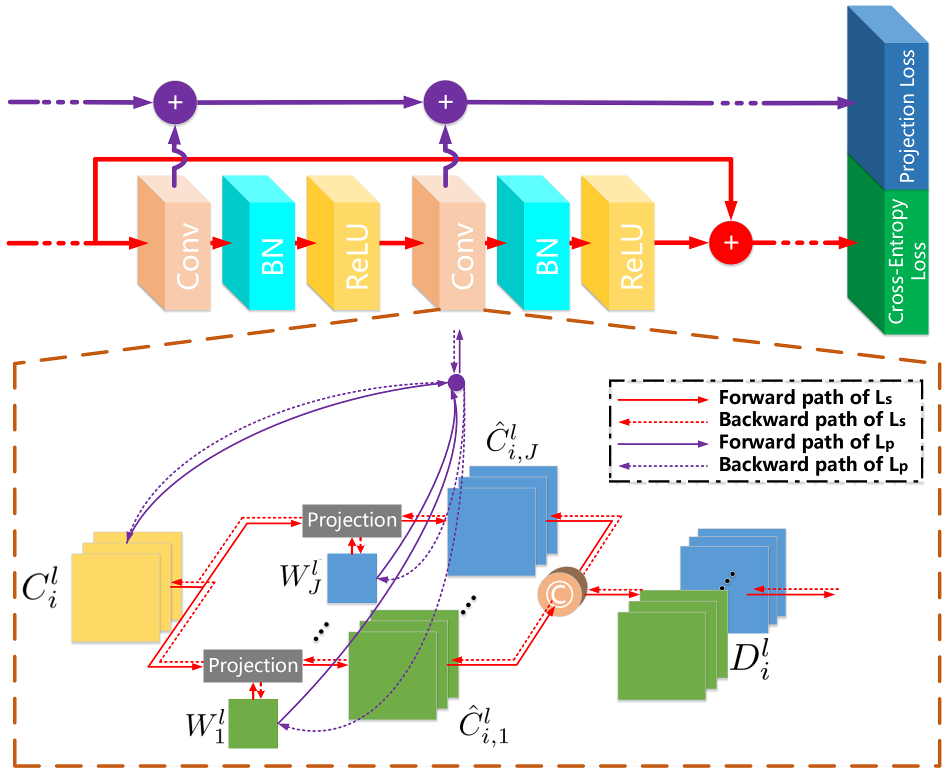

Figure 1: In PCNNs, a new discrete back propagation via projection is proposed to build binarized neural networks in an end-to-end manner. Full-precision convolutional kernels C^l_i are quantized to be $\hat{C}^l_{i,j}$ via the projection. Due to multiple projections, the diversity is enriched. The resulting kernel tensor D^l_i is used the same as in conventional ones. Both the projection loss L_p and the traditional loss L_s are used to train PCNNs. We illustrate our network structure *Basic Block Unit* based on Resnet, and more specific details are shown in the dotted box (projection convolution layer). © indicates the concatenation operation on the channels. Note that the projection matrixes W^l_j and full-precision kernels C^l_i are not used in the inference.

Due to the projection matrices only used for optimization but not for reference, resulting in a compact and portable learning architecture, the PCNNs model can be highly compressed and also efficient which outperforms all other state-of-the-art BNNs. The contributions of this paper include:

(1) A new discrete back propagation via projection (DBPP) algorithm is proposed to build BNNs in an end-to-end manner. By exploiting multiple projections, we learn a set of diverse quantized kernels that thus compress the full-precision models in a better way.

(2) A projection loss is theoretically achieved in DBPP, and we develop a generic projection convolutional layer to efficiently binarize existing convolutional networks, such as VGGs and Resnets.

(3) PCNNs achieve the best classification performance compared to other state-of-the-art BNNs on the ImageNet and CIFAR datasets.

Related Work

Existing CNNs compression works generally follow three pathways, which are quantized neural networks (QNNs) (Li et al. 2017; Zhou et al. 2018; Wu et al. 2018; Han, Mao, and J. Dally 2016), sparse connections (Denton et al. 2014; Tai et al. 2015) and designing new CNN architectures (G. Howard et al. 2017; Zhang et al. 2017; N. Iandola et al. 2017). Recent research efforts on quantized neural networks (QNNs) have considerably reduced the memory requirement and computation complexity in DCNNs by using codebook-based network pruning (Li et al. 2017), Huffman encoding (Han, Mao, and J. Dally 2016), Hash functions (Chen et al. 2015), low bitwidth weights and activations (Hubara et al. 2016), and further generalized low bitwidth gradients (Zhou et al. 2016) and errors (Wu et al. 2018). BinaryNet based on BinaryConnect is proposed to train DCNNs with binary weights, where the activations are triggered at run-time while the parameters are computed at training time (Courbariaux et al. 2016). The XNOR-Net is introduced to approximate the convolution operation using primarily binary operations, which reconstructs unbinarized kernels using binary kernels with a single scaling factor (Rastegari et al. 2016).

Our target is similar to those binarization methods that all attempt to reduce memory consumption and replace most arithmetic operations with binary operations. However, the way we quantize the kernels is different in two respects. First, we use an efficient discrete optimization based on pro-

Table 1: A brief description of main notation used in the paper.

C_i^l: full-precision kernel	W_j^l: projection matrix	D_i^l: set of $\hat{C}_{i,j}^l$	Ω: discrete set
$\hat{C}_{i,j}^l$: quantized kernel	$\widetilde{W}_j^l$: duplicated W	λ: trade-off scaler for L_P	a_i: discrete value in Ω
i: kernel index	j: projection index	l: layer index	$[k]$: iteration index
I: number of kernels	J: total projection number	L: number of layers	h: plane index

jection, in which the continuous values tend to converge to a set of nearest discrete values within our framework. Second, multiple projections are introduced to bring diversity into BNNs and further improve the performance.

Discrete Back Propagation via Projection

Discrete optimization is one of hottest topics in mathematics and is widely used to solve computer vision problems (Kim et al. 2017; Laude et al. 2018). In this paper, we propose a new discrete back propagation algorithm, where a projection function is exploited to binarize or quantize the input variables in a unified framework. Due to a flexible projection scheme in use, we actually obtain diverse binarized models of higher performance than previous ones. In Table 1, we describe the main notation used in the following sections.

Projection

In our work, we define the quantization of the input variable as a projection onto a set,

$$\Omega := \{a_1, a_2, ..., a_U\}, \tag{1}$$

where each element $a_i, i = 1, 2, ..., U$ satisfies the constraint $a_1 < a_2 < ... < a_U$, and is the discrete value of the input variable. Then we define the projection of $x \in \mathrm{R}$ onto Ω as:

$$P_\Omega(x) = \arg\min_{a_i} \|x - a_i\|, i \in \{1, ..., U\}, \tag{2}$$

which indicates that the projection aims to find the nearest discrete value for each continuous value x.

Optimization

For any $f(x)$ whose gradient exists, we minimize it based on the discrete optimization method. Conventionally, the discrete optimization problem is solved by searching for an optimal set of discrete values with respect to the minimization of a loss function. We propose that in the kth iteration, based on the projection in Eq. 2, $x^{[k]}$ is quantized to $\hat{x}^{[k]}$ as:

$$\hat{x}^{[k]} = P_\Omega(x^{[k]}), \tag{3}$$

which is used to define our optimization problem as:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & \hat{x}_j = P_\Omega^j(\omega_j \circ x), \end{array} \tag{4}$$

where ω_j is a projection matrix[1], and $\circ$ denotes the Hadamard product. The new minimization problem in (4) is hard to solve using back propagation (e.g., deep learning paradigm) due to the new constraint $\hat{x}_j^{[k]} = P_\Omega^j(\omega_j \circ x^{[k]})$. To solve the problem within the back propagation framework, we define our update rule as:

$$x \leftarrow x^{[k]} - \eta\delta_{\hat{x}}^{[k]}, \tag{5}$$

where the superscript $[k+1]$ is dropped from x, $\delta_{\hat{x}}$ is the gradient of $f(x)$ with respect to $x = \hat{x}$, and η is the learning rate. The quantization process $\hat{x}^{[k]} \leftarrow x^{[k]}$, i.e., $P_\Omega^j(\omega_j \circ x^{[k]})$, is equivilent to finding the projection of $\omega_j \circ (x + \eta\delta_{\hat{x}}^{[k]})$ onto Ω as:

$$\hat{x}^{[k]} = \arg\min_{\hat{x}} \{\|\hat{x} - \omega_j \circ (x + \eta\delta_{\hat{x}}^{[k]})\|^2, \hat{x} \in \Omega\}. \tag{6}$$

Obviously, $\hat{x}^{[k]}$ is the solution to the problem in (6). So, by incorporating (6) into $f(x)$, we obtain a new formulation for (4) based on the Lagrangian method as:

$$\min f(x) + \lambda \sum_j^J \|\hat{x}^{[k]} - \omega_j \circ (x + \eta\delta_{\hat{x}}^{[k]})\|^2, \tag{7}$$

where J is the total number of projection matrices. The new added part (right) shown in (7) is a quadratic function, and is referred to as the **projection loss**.

Projection Convolutional Neural Networks

Projection convolutional neural networks (PCNNs), shown in Fig. 1, work by taking advantage of DBPP for model quantization. To this end, we reformulate our projection loss shown in (7) into the deep learning paradigm as:

$$L_p = \frac{\lambda}{2} \sum_{l,i}^{L,I} \sum_j^J ||\hat{C}_{i,j}^{l,[k]} - \widetilde{W}_j^{l,[k]} \circ (C_i^{l,[k]} + \eta\delta_{\hat{C}_{i,j}^{l,[k]}})||^2, \tag{8}$$

where $C_i^{l,[k]}$, $l \in \{1, ..., L\}, i \in \{1, ..., I\}$ denotes the ith kernel tensor of the lth convolutional layer in the kth iteration. $\hat{C}_{i,j}^{l,[k]}$ is the quantized kernel of $C_i^{l,[k]}$ via projection $P_\Omega^{l,j}, j \in \{1, ..., J\}$ as:

$$\hat{C}_{i,j}^{l,[k]} = P_\Omega^{l,j}(\widetilde{W}_j^{l,[k]} \circ C_i^{l,[k]}), \tag{9}$$

where $\widetilde{W}_j^{l,[k]}$ is a tensor, calculated by duplicating a learned projection matrix $W_j^{l,[k]}$ along the channels, which thus fits the dimension of $C_i^{l,[k]}$. $\delta_{\hat{C}_{i,j}^{l,[k]}}$ is the gradient at $\hat{C}_{i,j}^{l,[k]}$ calculated based on L_S, i.e., $\delta_{\hat{C}_{i,j}^{l,[k]}} = \frac{\partial L_S}{\partial \hat{C}_{i,j}^{l,[k]}}$. The iteration index $[k]$ is omitted in the following for simplicity.

[1]Since the kernel parameters x are represented as a matrix, ω_j is also represented as a matrix.

In PCNNs, both the cross-entropy loss and projection loss are used to build the total loss L as:

$$L = L_S + L_P. \tag{10}$$

The proposed projection loss regularizes the continuous value converging onto Ω, at the same time minimizing the cross-entropy loss, which is illustrated in Fig. 2 and Fig. 3.

Forward Propagation based on Projection Convolution Layer

For each full-precision kernel C_i^l, the corresponding quantized kernels $\hat{C}_{i,j}^l$ are concatenated to construct the kernel D_i^l which actually participates in convolution operation.

$$D_i^l = \hat{C}_{i,1}^l \oplus \hat{C}_{i,2}^l \oplus \cdots \oplus \hat{C}_{i,J}^l, \tag{11}$$

where $\oplus$ denotes the concatenation operation on tensors.

In PCNNs, the projection convolution is implemented based on D^l and F^l to calculate the feature map F^{l+1} of the next layer:

$$F^{l+1} = Conv2D(F^l, D^l), \tag{12}$$

where $Conv2D$ is the traditional 2D convolution. Although our convolutional kernels are 3D-shaped tensor, we design the following strategy to fit the traditional 2D convolution:

$$F_{h,j}^{l+1} = \sum_{i,h} F_h^l \otimes D_{i,j}^l, \tag{13}$$

$$F_h^{l+1} = F_{h,1}^l \oplus \cdots \oplus F_{h,J}^l, \tag{14}$$

where $\otimes$ denotes the convolutional operation. $F_{h,j}^{l+1}$ is the jth channel of the hth feature map in the $(l+1)$th convolutional layer and F_h^l denotes the hth feature map in the lth convolutional layer.

It should be emphasized that we can utilize multiple projections to enrich the diversity of convolutional kernels D^l, though the single projection already achieves much better performance. This is due to the DBPP in use, which is clearly different from (Lin, Zhao, and Pan 2017) in that it is based on a single quantization scheme. Within our convolutional scheme, there is no dimension disagreement on feature maps and kernels in two successive layers. Thus, we can replace the traditional convolutional layers with ours to change widely-used networks, such as VGGs and Resnets. At the inference time, we only store the set of quantized kernels D_i^l instead of full-precision ones, that is, the projection matrices W_j^l are not used for the inference, which achieves the reduction of storage.

Backward Propagation

According to Eq. 10, what should be learned and updated are the full-precision kernels C_i^l and projection matrix W^l ($\widetilde{W}^l$) using the updated equations described below.

Updating C_i^l: We define δ_{C_i} as the gradient of the full-precision kernel C_i, and have:

$$\delta_{C_i^l} = \frac{\partial L}{\partial C_i^l} = \frac{\partial L_S}{\partial C_i^l} + \frac{\partial L_P}{\partial C_i^l}, \tag{15}$$

$$C_i^l \leftarrow C_i^l - \eta_1 \delta_{C_i^l}, \tag{16}$$

where η_1 is the learning rate for the convolutional kernels.

More specifically, for each item in Eq. 15, we have:

$$\begin{aligned}\frac{\partial L_S}{\partial C_i^l} &= \sum_j^J \frac{\partial L_S}{\partial \hat{C}_{i,j}^l} \frac{\partial P_\Omega^{l,j}(\widetilde{W}_j^l \circ C_i^l)}{\partial(\widetilde{W}_j^l \circ C_i^l)} \frac{\partial(\widetilde{W}_j^l \circ C_i^l)}{\partial C_i^l} \\ &= \sum_j^J \frac{\partial L_S}{\partial \hat{C}_{i,j}^l} \circ \mathbb{1}_{-1 \le \widetilde{W}_j^l \circ C_i^l \le 1} \circ \widetilde{W}_j^l,\end{aligned} \tag{17}$$

$$\frac{\partial L_P}{\partial C_i^l} = \lambda \sum_j^J \left[\widetilde{W}_j^l \circ \left(C_i^l + \eta \delta_{\hat{C}_{i,j}^l}\right) - \hat{C}_{i,j}^l\right] \circ \widetilde{W}_j^l, \tag{18}$$

where $\mathbb{1}$ is the indicator function (Rastegari et al. 2016) widely used to estimate the gradient of non-differentiable function.

Updating W_j^l: Likewise, the gradient of the projection parameter $\delta_{W_j^l}$ consists of the following two parts:

$$\delta_{W_j^l} = \frac{\partial L}{\partial W_j^l} = \frac{\partial L_S}{\partial W_j^l} + \frac{\partial L_P}{\partial W_j^l}, \tag{19}$$

$$W_j^l \leftarrow W_j^l - \eta_2 \delta_{W_j^l}, \tag{20}$$

where η_2 is the learning rate for W_j^l. We further have:

$$\begin{aligned}\frac{\partial L_S}{\partial W_j^l} &= \sum_h^J \left(\frac{\partial L_S}{\partial \widetilde{W}_j^l}\right)_h \\ &= \sum_h^J \left(\sum_i^I \frac{\partial L_S}{\partial \hat{C}_{i,j}^l} \frac{\partial P_\Omega^{l,j}(\widetilde{W}_j^l \circ C_i^l)}{\partial(\widetilde{W}_j^l \circ C_i^l)} \frac{\partial(\widetilde{W}_j^l \circ C_i^l)}{\partial \widetilde{W}_j^l}\right)_h \\ &= \sum_h^J \left(\sum_i^I \frac{\partial L_S}{\partial \hat{C}_{i,j}^l} \circ \mathbb{1}_{-1 \le \widetilde{W}_j^l \circ C_i^l \le 1} \circ C_i^l\right)_h,\end{aligned} \tag{21}$$

$$\frac{\partial L_P}{\partial W_j^l} = \lambda \sum_h^J \left(\sum_i^I \left[\widetilde{W}_j^l \circ \left(C_i^l + \eta \delta_{\hat{C}_{i,j}^l}\right) - \hat{C}_{i,j}^l\right] \circ \left(C_i^l + \eta \delta_{\hat{C}_{i,j}^l}\right)\right)_h, \tag{22}$$

where h indicates the hth plane of the tensor along the channels. It shows that the proposed algorithm can be trainable in an end-to-end manner, and we summarize the training procedure in Alg. 1. In implementation, we use the mean of W in the forward process, but keep the original W in the backward propagation.

Note that in PCNNs for BNNs, we set U=2 and a_2=$-a_1$. Two binarization processes are used in PCNNs. The first one is the kernel binarization, which is done based on the projection onto Ω, whose elements are calculated based on the

Algorithm 1 Discrete Back Propagation via Projection

```
Require:
    The training dataset; the full-precision kernels $C$; the projection matrix $W$; the learning rates $\eta_1$ and $\eta_2$.
Ensure:
    The PCNNs based on the updated $C$ and $W$.
 1: Initialize $C$ and $W$ randomly;
 2: repeat
 3:     // Forward propagation
 4:     for $l = 1$ to $L$ do
 5:         $\hat{C}^l_{i,j} \leftarrow Project(C^l_i)$; // using Eq. 9
 6:         $D^l_i \leftarrow Concatenate(\hat{C}_{i,j})$; // using Eq. 11
 7:         Perform activation binarization; //using the sign function
 8:         Traditional 2D convolution; // using Eq. 12, 13 and 14
 9:     end for
10:     // Backward propagation
11:     Compute $\delta_{\hat{C}^l_{i,j}} = \frac{\partial L_S}{\partial \hat{C}^l_{i,j}}$;
12:     for $l = L$ to 1 do
13:         // Calculate the gradients
14:         calculate $\delta_{C^l_i}$; // using Eq. 15, 17 and 18
15:         calculate $\delta_{W^l_j}$; // using Eq. 19, 21 and 22
16:         // Update the parameters
17:         $C^l_i \leftarrow C^l_i - \eta_1 \delta_{C^l_i}$; // Eq. 16
18:         $W^l_j \leftarrow W^l_j - \eta_2 \delta_{W^l_j}$; //Eq. 20
19:     end for
20:     Adjust the learning rates $\eta_1$ and $\eta_2$.
21: until the network converges
```

mean of absolute values of all the full-precision kernels per layer (Rastegari et al. 2016) as:

$$\frac{1}{I}\sum_{i}^{I}\left(\|C^l_i\|_1\right). \tag{23}$$

where I is the total number of kernels.

Experiments

PCNNs are evaluated on the object classification task with CIFAR10/100 (Krizhevsky, Nair, and Hinton 2014) and ILSVRC12 ImageNet datasets (Deng et al. 2009). Our DBPP algorithm can be applied to any DCNNs. For fair comparison with other state-of-the-art BNNs, we use Wide-Resnet (WRN) (Zagoruyko and Komodakis 2016), Resnet18 (He et al. 2016) and VGG16 (Simonyan and Zisserman 2014) as our full-precision backbone networks, to build our PCNNs by replacing their full-precision convolution layers with our projection convolution.

Datasets and Implementation details

Datasets: CIFAR 10/100 (Krizhevsky, Nair, and Hinton 2014) are natural image classification datasets containing a training set of 50K and a testing set of 10K 32 × 32 color images across the 10/100 classes. For CIFAR10/100 and parameter study, we employ WRNs to evaluate our PCNNs and report the accuracies. Unlike CIFAR 10/100, ILSVRC12 ImageNet object classification dataset (Deng et al. 2009) is more challenging due to its large scale and greater diversity. There are 1000 classes and 1.2 million training images and 50k validation images in it. For comparison of our method to the state-of-the-art on the ImageNet dataset, we adopt Resnet18 and VGG16 to validate the superiority and effectiveness of PCNNs.

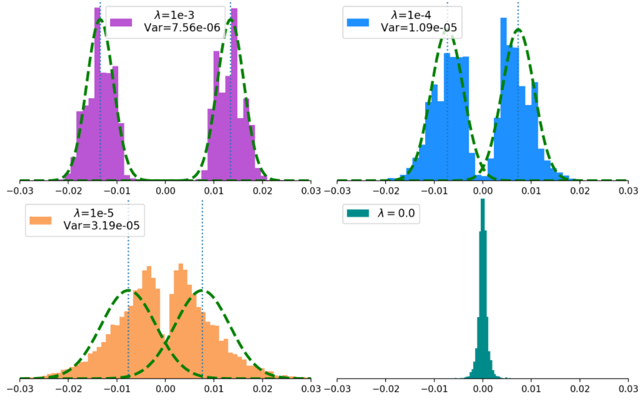

Figure 2: We visualize the kernel weights distribution of the first convolution layer of PCNN-22. When decreasing λ, which balances the projection loss and cross-entropy loss, the variance becomes larger. Particularly, when λ=0 (no projection loss), only one cluster is obtained, wherein the kernel weights distribute around 0, which could result in instability during binarization. Conversely, two Gaussians (with the projection loss, $\lambda > 0$) are more powerful than the single one (without the projection loss), which thus results in better BNNs as also validated in Table 2.

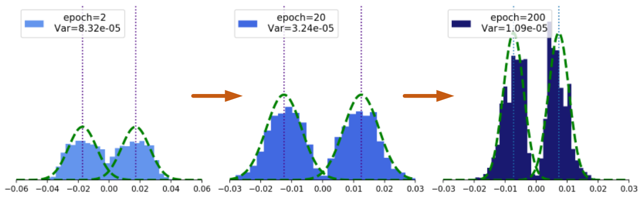

Figure 3: With λ fixed to $1e-4$, the variance of the kernel weights becomes smaller from the 2th epoch to the 200th epoch, which confirms that the projection loss does not affect the convergence.

WRN: WRN is a network structure similar to Resnet with a depth factor k to control the feature map depth dimension expansion through 3 stages, within which the dimensions remain unchanged. For simplicity we fix the depth factor to 1. Each WRN has a parameter i which indicates the channel dimension of the first stage and we vary it between 16 and 64 leading to two network structures, 16-16-32-64 and 64-64-128-256. Other setting including training details is the same as (Zagoruyko and Komodakis 2016) except that we only add dropout layers with a ratio 0.3 to the 64-64-128-256 structure in case of overfitting. WRN-22 indicates a network with 22 convolutional layers and similarly for WRN-40.

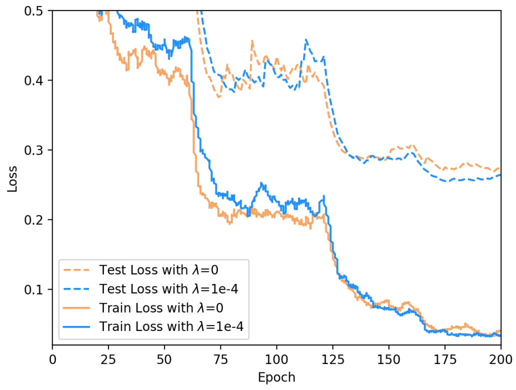

Figure 4: Training and testing curves of PCNN-22 when λ=0 and $1e-4$, which shows that the projection affects little on the convergence.

Resnet18 and VGG16: For these two networks, we simply replace the convolutional layers with the projection convolution layer and keep other components unchanged. To train the two networks, we choose SGD as the optimization algorithm with a momentum of 0.9 and a weight decay $1e-4$. The initial learning rate for W_j^l is 0.01 and for C_i^l and other parameters the initial learning rates are 0.1, with a degradation of 10% for every 20 epochs before it reaches the maximum epoch of 60.

Ablation study

Parameter: As mentioned above, the proposed projection loss has the ability to control the process of quantization, similar to clustering. We compute the distributions of the full-precision kernels and visualize the results in Figs. 2 and 3. The hyper-parameter λ is designed to balance the projection loss and the cross-entropy loss. We vary it from $1e-3$ to $1e-5$ and finally set it to 0 in Fig. 2, where the variance becomes larger as decreasing λ. When λ=0, only one cluster is obtained, where the kernel weights distribute tightly around the threshold=0. This could result in instability during binarization, because little noise may cause a positive weight to be negative and vice versa.

We also show the evolution of the distribution about how the projection loss works in the training process in Fig. 3. A natural question is: do we always need a large λ? As a discrete optimization problem, the answer is no and the experiment in Table 2 can verify it, i.e., both the projection loss and cross-entropy loss should be considered at the same time with a good balance. For example, when λ is set to $1e-4$, the accuracy is higher than those with other values. Thus, we fix λ to $1e-4$ in the following experiments.

Learning convergence: For PCNN-22 in Table 2, the PCNNs model is trained for 200 epochs and then used to conduct inference. In Fig. 4, we plot the training and testing loss with λ=0 and λ=1e-4, respectively. It clearly shows that PCNNs with λ=1e-4 (blue curves) converge faster than PCNNs with λ=0 (yellow curves) when the epoch number > 150.

Diversity visualization: In Fig. 5, we visualize four channels of the binary kernels D_i^l in the first row, feature maps

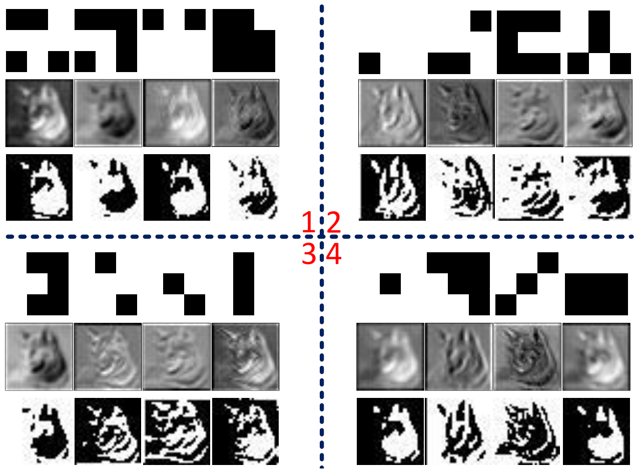

Figure 5: Illustration of binary kernels D_i^l (first row), feature maps produced by D_i^l (second row), and corresponding feature maps after binarization (third row) when J=4. This confirms the diversity in PCNNs.

Table 2: With different λ, the accuracy of PCNN-22 and PCNN-40 based on WRN-22 and WRN-40, respectively, on CIFAR10 dataset.

Model	λ			
	$1e-3$	$1e-4$	$1e-5$	0
PCNN-22	91.92	92.79	92.24	91.52
PCNN-40	92.85	93.78	93.65	92.84

produced by D_i^l in the second row, and corresponding feature maps after binarization in the third row when J=4, which illustrates the diversity of kernels and feature maps in PCNNs. Thus, the multiple projection functions can capture the diverse information and result in a high performance based on the compressed models

Results on CIFAR-10/100 datasets

We first compare our PCNNs with the original WRNs with initial channel dimension i=16 and 64 in Table 3. We compare the performance of different models with similar parameter amount. Thanks to the multiple projections in use (J=4), our results on both datasets are comparable with the full-precision networks. Then, we compare our results with other state-of-the-arts such as BinaryConnect (Courbariaux, Bengio, and David 2015), BNN (Courbariaux et al. 2016), LBCNN (Xu, Boddeti, and Savvides 2016), BWN, XNOR-Net (Rastegari et al. 2016) and the model in (McDonnell 2018). It is observed that at least a 1.5% accuracy improvement is gained with our PCNNs, and in most cases large margins are achieved, which indicates that DBPP is really effective for the task of model compression. As shown in Table 3, the increase of J can also boost the performance and exhibits a better modeling ability than others by the flexible projection scheme with more feature diversity gain. However, for small models like PCNN (i=16), the increase of J

Table 3: Test accuracy on CIFAR10/100 dataset. PCNNs are based on WRN-22. The numbers of parameters refer to the models on CIFAR10.

Model	#Para.	Dataset	
		CIFAR-10	CIFAR-100
WRN (i=16)	0.27M	92.62	68.83
WRN (i=64)	4.29M	95.75	77.34
BinaryConnect	14.02M	91.73	-
BNN	14.02M	89.85	-
LBCNN	14.02M	92.99	-
BWN	14.02M	90.12	-
XNOR-Net	14.02M	89.83	-
(McDonnell 2018)	4.30M	93.87	76.13
PCNN (i=16, J=1)	0.27M	89.17	62.66
PCNN (i=16, J=2)	0.54M	91.27	66.86
PCNN (i=16, J=4)	1.07M	92.79	70.09
PCNN (i=64, J=1)	4.29M	94.31	76.93
PCNN (i=64, J=4)	17.16M	95.39	78.13

Table 4: Test accuracy on ImageNet. 'W' and 'A' refer to the weight and activation bitwidth respectively. The first two PCNNs are based on Resnet18, while the last one is based on VGG16. † and ‡ indicate J=1 and J=2 respectively.

Model	W	A	Top-1	Top-5
Resnet18	32	32	69.3	89.2
BWN	1	32	60.8	83.0
DoReFa-Net	1	4	59.2	81.5
XNOR-Net	1	1	51.2	73.2
ABC-Net	1	1	42.7	67.6
BNN	1	1	42.2	67.1
Bi-Real Net	1	1	56.4	79.5
PCNN	1	32	$63.5^{\dagger}$, $66.1^{\ddagger}$	$85.1^{\dagger}$, $86.7^{\ddagger}$
PCNN	1	1	**$57.3^{\dagger}$**	**$80.0^{\dagger}$**
VGG16	32	32	73.0	91.2
PCNN	1	32	$69.0^{\dagger}$	$89.1^{\dagger}$

seems more significant for avoiding accuracy degradation, but for large models like PCNN (i=64), it results in relatively small improvement. So we suggest to use small J=1 to avoid additional computation in large models, like the PCNNs based on Resnet18 and VGG16, which can still outperform the others.

Results on ILSVRC12 ImageNet classification dataset

For the ImageNet dataset, we employ two data augmentation techniques sequentially: 1) randomly cropping patches of 224×224 from the original image, and 2) horizontally flipping the extracted patches in the training. While in the testing, the Top-1 and Top-5 accuracies on the validation set with single center crop are measured. We modify the architecture of Resnet18 following (Liu et al. 2018) with additional PReLU (He et al. 2015) and the final results of our PCNNs are finetuned based on the pretrained models with only kernel weights binarized, halving the learning rate in the training.

Table 5: Memory usage and efficiency of convolution comparison with XNOR-Net, full-precision Resnet18, and PCNN (J=1). PCNN is based on Resnet18.

Model	Memory usage	Memory saving	Speedup
PCNN	33.7 Mbit	11.10 $\times$	58 $\times$
XNOR-Net	33.7 Mbit	11.10 $\times$	58 $\times$
Resnet18	374.1 Mbit	-	-

In Table 4, we compare our PCNNs with several other state-of-the-art models. The first part of the comparison is based on Resnet18 with 69.3% Top-1 accuracy on the full-precision model. Although BWN (Rastegari et al. 2016) and DoReFa-Net (Zhou et al. 2016) achieve Top-1 accuracy with degradation of less than 10%, it should be noted that they apply full-precision and 4-bit activation respectively. With both of the weights and activations binarized, the BNN model in (Courbariaux et al. 2016), ABC-Net (Lin, Zhao, and Pan 2017) and XNOR-Net (Rastegari et al. 2016) fail to maintain the accuracy and are inferior to our PCNN. For example, compared with the result of XNOR-Net, PCNN increases the Top-1 accuracy by 6.1%. For a fair comparison, we set the activation of our PCNNs to full-precision and vary J from 1 to 2, and these two results consistently outperform BWN. We also compare our method with the full-precision VGG16 model, and the accuracy drop is tolerable if only weights are binarized (see the last two rows). Note that our DBPP algorithm still works very well on the large dataset, particularly when J=1, which further validates the significance of our method.

In short, we achieved a new state-of-the-art performance compared to other BNNs, and much closer performance to full-precision models in the extensive experiments, which clearly validate the superiority of DBPP for the BNNs calculation.

Memory Usage and Efficiency Analysis

Memory use is analyzed by comparing our approach with the state-of-the-art XNOR-Net (Rastegari et al. 2016) and the corresponding full-precision network. The memory usage is computed as the summation of 32 bits times the number of full-precision kernels and 1 bit times the number of the binary kernels in the networks. As shown in Table 5, our proposed PCNNs, along with XNOR-Net, reduces the memory usage by 11.10 times compared with the full-precision Resnet18. Note that when J is set to 1, the parameter amount of our model is almost the same as XNOR-Net. The reason is that the projection parameters W_j^l are only used when training for enriching the diversity in PCNNs, whereas they are not used when inference. For efficiency analysis, if all of the operands of the convolutions are binary, then the convolutions can be estimated by XNOR and bitcounting operations (Courbariaux et al. 2016), which gains $58\times$ speedup in CPUs (Rastegari et al. 2016). For $J > 1$, the memory usage and computation cost for convolution are linear to J.

Conclusion and future work

We have proposed an efficient discrete back propagation via projection (DBPP) algorithm to obtain our projection convolutional neural networks (PCNNs), which can significantly reduce the storage requirement for computationally limited devices. PCNNs have shown to obtain much better performance than other state-of-the-art BNNs on ImageNet and CIFAR datasets. As a general convolutional layer, the PCNNs model can also be used in other deep models and different tasks, which will be explored in our future work.

Acknowledgements

The work was supported by the Natural Science Foundation of China under Contract 61601466, 61672079 and 61473086, and Shenzhen Peacock Plan KQTD2016112515134654. This work is supported by the Open Projects Program of National Laboratory of Pattern Recognition. Baochang Zhang is also with Shenzhen Academy of Aerospace Technology, Shenzhen, China. Jiaxin Gu and Ce Li have the same contributions to the paper.

References

Boureau, Y.-L.; Ponce, J.; and LeCun, Y. 2010. A theoretical analysis of feature pooling in visual recognition. In *ICML*, 111–118.

Chen, W.; Wilson, J.; Tyree, S.; Weinberger, K.; and Chen, Y. 2015. Compressing neural networks with the hashing trick. In *ICML*, 2285–2294.

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

Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255.

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.

Felzenszwalb, P., and Zabih, R. 2007. Discrete optimization algorithms in computer vision. *Tutorial at CVPR*.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, Tobias Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *CVPR*.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Quantized neural networks: Training neural networks with low precision weights and activations. *JMLR* 18(187):1–30.

Kim, S.; Min, D.; Lin, S.; and Sohn, K. 2017. Dctm: Discrete-continuous transformation matching for semantic flow. In *ICCV*, 4529–4538.

Krizhevsky, A.; Nair, V.; and Hinton, G. 2014. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*.

Laude, E.; Lange, J.-H.; Schüpfer, J.; Domokos, C.; Laura, L.-T.; Schmidt, F. R.; Andres, B.; and Cremers, D. 2018. Discrete-continuous admm for transductive inference in higher-order mrfs. In *CVPR*, 4539–4548.

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Peter Graf, H. 2017. Pruning filters for efficient convnets. In *ICLR*.

Lin, X.; Zhao, C.; and Pan, W. 2017. Towards accurate binary convolutional neural network. In *NIPS*, 345–353.

Liu, Z.; Wu, B.; Luo, W.; Yang, X.; Liu, W.; and Cheng, K.-T. 2018. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *ECCV*, 747–763.

McDonnell, M. D. 2018. Training wide residual networks for deployment using a single bit for each weight. In *ICLR*.

N. Iandola, F.; Han, S.; W. Moskewicz, M.; Ashraf, K.; J. Dally, W.; and Keutzer, K. 2017. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. In *ICLR*.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 525–542.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Tai, C.; Xiao, T.; Zhang, Y.; Wang, X.; and E, W. 2015. Convolutional neural networks with low-rank regularization. In *Computer Science*.

Wang, X.; Zhang, B.; Li, C.; Ji, R.; Han, J.; Cao, X.; and Liu, J. 2018. Modulated convolutional networks. In *CVPR*, 840–848.

Wu, S.; Li, G.; Chen, F.; and Shi, L. 2018. Training and inference with integers in deep neural networks. In *ICLR*.

Xu, J. F.; Boddeti, V. N.; and Savvides, M. 2016. Local binary convolutional neural networks. In *CVPR*, 19–28.

Zagoruyko, S., and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2017. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*.

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

Zhou, Y.; Ye, Q.; Qiu, Q.; and Jiao, J. 2017. Oriented response networks. In *CVPR*, 4961–4970.

Zhou, A.; Yao, A.; Wang, K.; and Chen, Y. 2018. Explicit loss-error-aware quantization for low-bit deep neural networks. In *CVPR*, 9426–9435.