Gradient Matters: Designing Binarized Neural Networks via Enhanced Information-Flow

Qi Wang, Senior Member, IEEE, Nianhui Guo, Zhitong Xiong, Zeping Yin, Xuelong Li, Fellow, IEEE

Abstract—Binarized neural networks (BNNs) have drawn significant attention in recent years, owing to great potential in reducing computation and storage consumption. While it is attractive, traditional BNNs usually suffer from slow convergence speed and dramatical accuracy-degradation on large-scale classification datasets. To minimize the gap between BNNs and deep neural networks (DNNs), we propose a new framework of designing BNNs, dubbed Hyper-BinaryNet, from the aspect of enhanced information-flow. Our contributions are threefold: 1) Considering the capacity-limitation in the backward pass, we propose an 1-bit convolution module named HyperConv. By exploiting the capacity of auxiliary neural networks, BNNs gain better performance on large-scale image classification task. 2) Considering the slow convergence speed in BNNs, we rethink the gradient accumulation mechanism and propose a hyper accumulation technique. By accumulating gradients in multiple variables rather than one as before, the gradient paths for each weight increase, which escapes BNNs from the gradient bottleneck problem during training. 3) Considering the ill-posed optimization problem, a novel gradient estimation warmup strategy, dubbed STE-Warmup, is developed. This strategy prevents BNNs from the unstable optimization process by progressively transferring neural networks from 32-bit to 1-bit. We conduct evaluations with variant architectures on three public datasets: CIFAR-10/100 and ImageNet. Compared with state-of-the-art BNNs, Hyper-BinaryNet shows faster convergence speed and outperforms existing BNNs by a large margin.

Index Terms—Neural network accelerating, 1-bit convolution, gradient approximation.

1 Introduction

Deep Neural Networks (DNNs) have achieved remarkable success in various fields such as computer vision [1], [2], speech recognition [3], [4] and reinforcement learning [5], [6]. Various architectures are developed with more parameters and heavier matrix computation. As a result, DNNs have better performance on several tasks. While the over-parameterized DNNs are attractive and powerful, the deployment of DNNs on resource-constrained embedded devices, such as cell phones, drones, self-driving cars, and other edge-devices, is not conductive and prevents DNNs from large-scale commercial applications. Consequently, this raises an interesting question whether current models can be compressed and accelerated? Recent studies show that the capacity of typical architectures such as VGG and ResNet is redundant and various techniques have been explored. The main methods of DNNs compression can be divided into three parts: 1) Compact architecture design where researchers mainly focus on finding new convolution methods with decreased computation. For example, ShuffleNet exploits the group convolution to accelerate DNNs and increase the feature-propagation with a channel-shuffling operation. MobileNet proposes a separable convolution to decouple the feature-extraction and feature-fusion. 2) Network quantization where the numerical precision of DNNs is transferred from 32-bit to fixed-point representation such as 16-bit, 4-bit, or even 1-bit. 3) Neural network pruning where unnecessary weights are pruned away from DNNs based on importance estimation. Among these methods, quantization is a hardware friendly method which accelerates DNNs with modified numerical computation techniques. Specifically, DNNs with fixed-point representation can be accelerated by replacing full precision matrix multiplication with several addition operations and bit-operations. From this perspective, DNNs with quantization methods can maximize the performance of hardware. Therefore, many efforts have been made to train DNNs with limited numerical precision (e.g., 16-bit, 8-bit, 4-bit) [7], [8], [9], [10], with the hope of maintaining similar performance with their 32-bit counterparts.

Among all these quantization methods, 1-bit quantization is the extreme case, where only a single bit is used to replace the full-precision representation of weights or activations in DNNs. Consequently, the fully quantized DNN (i.e., 1-bit) has drawn the attention of researchers [11], [12]. Specifically, when only the weights are binarized, DNNs with 1-bit weights are ∼32x smaller than the equivalent one with 32-bit weights. The convolution operation in DNNs can also be implemented using only addition and subtraction operations, leading to ∼2x computation acceleration [11]. If the intermediate representations in DNNs are simultaneously binarized, all floating-point operations in convolution layer can be replaced with XNOR and bit-counting operations [11], [12], which theoretically accelerates DNNs by ∼58x on CPUs. The pipeline of traditional binarized neural networks can be well summarized as follows [13]: 1) Binarizing the proxy 32-bit weights, activations in DNNs using binarizing function; 2) Going forward-propagation and backward-propagation with the 1-bit weights and activations; 3) Accumulating the 32-bit gradient in real valued variables to ensure that the Stochastic Gradient Descent (SGD) algorithm still works.

- The authors are with the School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi'an 710072, Shaanxi, China. E-mails: crabwq@gmail.com, xiongzhitong@gmail.com, guonianhui199512@gmail.com, li@nwpu.edu.cn.
- X. Li is the corresponding author.

Although the acceleration effect of 1-bit neural network is significant during the inference stage, BNNs still suffer from the serious performance degradation and slow convergence speed problem on large scale datasets like ImageNet. In this paper, we summarize the main reasons behind these problems on three aspects:

Capacity Limitation: As the weights and activations are constrained to be +1 or -1, the capacity of BNNs which means the possible combination of numerical representation, is greatly reduced. Therefore, BNNs with equivalent structure as DNNs can not cover the great diversity in large-scale datasets like ImageNet. Some recent methods have tried to enhance BNNs by increasing the channel numbers of each reduced-precision layer and gained much better performance [14]. This indicates that capacity-limitation is one of the key problems for BNNs.

Gradient Mismatch: The gradient of directly training BNNs is almost zero everywhere due to the non-differential sign function [11], [13]. To train BNNs in a differentiable manner, existing methods address this issue using the Straight Through Estimator [15] scheme. Specifically, at each backward propagation, this scheme manually approximates the gradient of sign function with a related surrogate (normally hyperbolic tangent function) [12], [13], [15]. In this way, a full differentiable pipeline is constructed. However, this unusual "gradient" is actually not the real gradient of the loss function and can only provide a coarse gradient descent direction for weights. Therefore, inaccurate gradients can hurt the training procedure of BNNs and result in more iterations to correct the searching procedure. Such difference between forward and backward passes is called gradient-mismatch.

Gradient Accumulation: As being analyzed above, BNNs go forward and backward using 1-bit weights/activations, but accumulate 32-bit gradients in predefined 32-bit buffers. The reason is that the magnitude of the gradient is too small to be directly used for binary adjustments (± 1) [13] and has to be progressively accumulated in a proxy tensor with equivalent shape. Though accumulating gradients in 32-bit tensors keeps the effectiveness of SGD algorithm, the current accumulation method is certainly not the optimal choice for bit flipping optimization. The accumulation procedure also means that 1-bit representation can only be turned in an inefficient proxy manner, and BNNs can even stop learning if the gradient is too small to change the sign of the proxy tensor.

Considering the aforementioned problems, in this paper, a new framework of training BNNs called Hyper-BinaryNet is proposed to enhance the binarized neural network with improved gradient propagation. Several modules are proposed to address current limitations in BNNs, considering the smoothness of the training procedure. The main contributions can be summarized as follows:

1) This paper is the first attempt to combine the training of BNNs with auxiliary neural networks. By exploiting auxiliary capacity only during the training stage , the capacity-limitation of BNNs can be well relieved.
2) This paper proposes a progressive neural architecture binarizing strategy to alleviate the gradient-mismatch problem in current BNNs. By transferring neural networks from full-precision to single-bit, BNNs are trained with smoother and more accurate gradients.
3) This paper rethinks the importance of gradient-accumulation mechanism in BNNs and an improved Hyper-accumulation is proposed. By increasing gradient paths for each 1-bit weights, current training of BNNs can be well accelerated.
4) To evaluate the effectiveness of Hyper-BinaryNet, experiments on CIFAR-10/100 and ImageNet-12 datasets are conducted. The numerical results show that Hyper-BinaryNet outperforms SOTA methods by a large margin.

The remainder of this paper is organized as follows. Section 2 reviews the related works of neural network binarization. Section 3 introduces the details of the proposed Hyper-BinaryNet. In Section 4, extensive experiments are conducted to validate the proposed method comprehensively. Conclusions and future work are presented in Section 5.

2 Related Work

In this section, we review some early researches related to ours. Firstly, recent studies on neural network binarizing are introduced. Secondly, the related works on numerical discretization are investigated. Finally, the relationship between HyperNetwork [16] and our work is explained.

Network Binarizing: BNNs have drawn the attention of researchers in recent years [11], [12], [13], [17], [18], [19], [20], [21]. For example, [11] shows that filters can be quantized to ± 1 without noticeable dropping of classification accuracy on CIFAR-10. [13] further proves that BNNs still work when both weights and activations are binarized. Further, [12] introduces XNOR-Net, and extends the first benchmark result on the ImageNet-12 classification task, exploiting extra 32-bit scaling factor to approximate typical convolution. More recent studies focus on BNNs with improved representation ability. For instance, [17] introduces ABC-Net, approximating full-precision weights/activations with the linear combination of multiple binary bases. Both [14] and [18] find that wider BNNs suffer less from the accuracy degradation problem, benefiting from the increased capacity. [19] mines the channel-wise interactions by a reinforcement learning model, and imposes channel-wise priors on the intermediate feature maps through the interacted bitcount function. What's more, [20] proposes an improved binary training method (BNN+), by introducing a regularization function that encourages training weights around binary values. [21] shows that by attempting to minimize the discrepancy between the output of the binary and the corresponding real-valued convolution, additional significant accuracy gains can be obtained. Along with early studies, Hyper-BinaryNet is an effective way of improving the capacity of BNNs. However, what needs to note is that though extra full-precision capacity from the auxiliary network is exploited, it is not embedded in BNNs and is removed away during the inference stage, maximizing the advantage of bit-counting operation.

Hard Binarizing: Most BNNs use hard binarization to obtain 1-bit representation during training. Courbariaux et al. [11] combine expectation back-propagation (EBP) [22] with BNNs, and propagate gradients through discrete weights for the first time. Later, they extend the same idea to BinaryNet [13] to binarize activation (± 1) as well, maintaining sufficiently high accuracy on the MNIST, CIFAR10 and SVHN

TABLE 1
A brief introduction of variables used in the paper

G: convolution function	$\mathcal{L}$: loss function	W: weights for G	$\widetilde{\mathrm{W}}$: fused weights of W
F: continuous function	x: general variables	A: input activations for C	$\widetilde{\mathrm{A}}$: fused activations of A
D: Auxiliary Neural Network	Z: embedded vectors for D		
i: channel index of A	λ: scalable scalar for F	σ: exponential decay rate for λ	l: layer index
j: kernel index of W	t: current iteration index	α: learning rate	S: exponential decay step

datasets. They also introduce a binary matrix multiplication GPU kernel to verify its great potential in speeding up computation. [12] further introduces XNOR-Net, where efficient bit-counting operation multiplying full-precision scale factor is used to approximate each typical convolution. Owning to the improved mix precision representation and new architecture, they extend the first benchmark result on ImageNet classification task. More recently [23] introduces a variant of residual module to enhance the representation ability of BNNs, dubbed Bi-Real Net. The idea of involving extra network capacity is also exploited in [24]. An extra trainable bit-projection function is proposed to replace typical weight binarization function, and it is proved to be helpful for BNNs. Furthermore, [25] concludes that BNNs are easier to suffer from intrinsic instability (training time) and non-robustness (train & test time) problems. Therefore, they combine BNNs with ensemble methods, obtaining performance improvement in terms of accuracy, robustness, and stability. However, BNNs using hard binarization (e.g., sign function) from scratch have already been verified to suffer from slow convergence and serious accuracy-degradation on large scale classification dataset ImageNet. The focus of this paper is the efficient training of a network with binary weights and activations. Different from current methods, Hyper-BinaryNet starts from the real-valued network and progressively transfers itself into BNNs, which is a smoother training strategy.

Soft Binarizing: One of the most related method to ours is the soft binarization. To get rid of the gradient approximation problem in DNNs where discretization operations are used, there have already been interests in studying continuous discretization technique. For example, [26] proposes reparameterizable Gumbel-Softmax trick to afford low-variance path derivative gradients for the categorical distribution for the first time. Inspired by this, HashNet [27] explores the relationship between sign function and scalable hyperbolic tangent function and designs a recurrent optimization approach to address the end-to-end hashing problem. Similarly, [28] exploits a similar approach to achieve true-gradient based optimization of BNNs, but the scalable tangent function is replaced with clip function. The method in [28] is only verified on small datasets.(e.g, MNIST, CIFAR10). [29] tries extending the similar idea to ImageNet classification task, but the experimental results 37.84% using XNOR-Net as the backbone is even lower than that reported in XNOR-Net paper 44.2%. Though continuous discretization skills above achieve true-gradient based optimization at the beginning of training, continuous function irresistibly degenerates into discretization function progressively as the continuous function shrinks. As a result, the SGD algorithm is not effective again. Therefore, the soft discretization methods above rely on iterative optimization trick to escape from the bottleneck effect in the backward pass. What needs to be pointed out is that this is not the case in Hyper-BinaryNet. While similar scalable continuous function is used, one of the key ideas on this aspect is the gradient approximation warmup scheme, which means an evolving Straight Through Estimation process during training. This is different from existing soft discretization methods.

Weight Generation: The idea of using a small network to generate weights for a larger network has already been studied in [16]. Then, SMASH [30] exploits the same idea in [16] as a way of accelerating neural architecture search. Specifically, by comparing the relative validation performance of networks with generated weights, a wide range of architectures can be searched in parallel. Further, [31] redesigns the structure of HyperNetwork, and combines it with graph neural network to model the topology of a full architecture. In this paper, we rethink the relationship between the gradient-accumulation mechanism and network convergence. Specifically, we extend HyerNetwork to discrete weight generation and combine it with the training of BNNs for the first time. As a result, each 1-bit weight is generated and tuned via a proxy network rather than single variable, which means improved neural capacity and gradient paths during training. Note that the key insight in this paper is the relationship between gradient-accumulation and training efficiency, rather than a simple weights generation mechanism.

3 Methodology

In this paper, we focus on improving BNNs from three different aspects: capacity-limitation, gradient-accumulation and gradient-approximation. The detailed approach for each aspect and its corresponding motivation will be introduced in this section.

3.1 Standard Binary Neural Network

To realize the compression and acceleration of DNNs, how to construct a neural network with binary weights/activations and effectively train it have become an interesting direction. In [13], a sign function based network binarization method is proposed for the first time:

$$x_b = Sign(x_r) = \begin{cases} +1, & if\ x \geqslant 0, \\ -1, & othersize, \end{cases} \tag{1}$$

where x_b is the binarized variable (e.g., binary weight W_b or activation A_b) and x_r represents the real-valued variable (e.g., W_r or A_r respectively). While binarized representations are used in forward and backward process, [13] accumu-

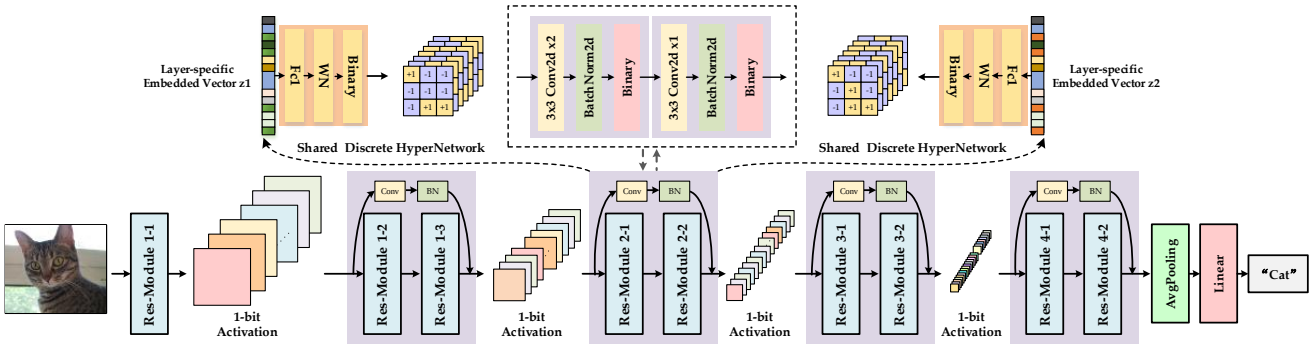

Fig. 1. The pipeline of Hyper-BinaryNet using ResNet-18 as backbone. Firstly, the main binarized neural network is built, except that the kernel size of convolution in skip connection branch is 3 rather than 1. Secondly, convolution layers in ResNet are mostly replaced with HyperConv modules, except the first convolution layer; Thirdly, during each forward propagation, the 1-bit kernels are regenerated by the auxiliary neural networks and used layer by layer. Finally, the real-valued gradients are top-down computed based on 1-bit weights/activations and accumulated in auxiliary networks. After training, all embedded vectors and auxiliary neural networks are removed, leaving a full binarized neural network which can be accelerated based on bit-counting operations.

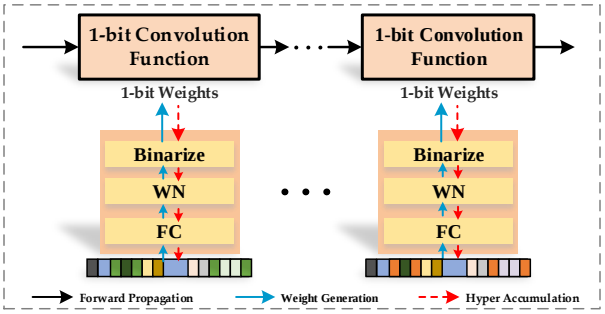

Fig. 2. A brief sketch of the weight-generation and hyper-accumulation mechanism based on auxiliary networks. "W_n" indicates binarized weights. "A_n" indicates input feature maps. Gradient information will be introduced into embedded vectors and proper binary weights will be generated.

lates the gradient in full-precision buffer W_r. The gradient-accumulation process can be clearly formulated as follows:

$$W_r^{t+1} = W_r^t - \alpha_t \frac{\partial \mathcal{L}}{\partial W_r^t}, \quad \frac{\partial \mathcal{L}}{\partial W_r^t} = \frac{\partial \mathcal{L}}{\partial W_b^t} \cdot \frac{\partial W_b^t}{\partial W_r^t}, \tag{2}$$

where $\frac{\partial W_b^t}{\partial W_r^t} = 1_{\|W_r\| \leq 1}$ represents the approximated gradient based on Straight Through Estimator scheme. $\mathcal{L}$ indicates the loss function. t represents current iteration index, and α represents the learning rate. W_r denotes full-precision weights. W_b denotes binary weights. This strategy makes it possible to train BNNs with gradient-based optimization. However, as we have analyzed above, this approach is not efficient and limits BNNs in three aspects: capacity limitation, gradient-mismatch and gradient-accumulation. As a result, BNNs are more difficult than DNNs to converge on large scale datasets like ImageNet [32]. Though several approaches have been explored to relieve one of these problems, they have not considered these problems together and addressed them in one single framework.

3.2 Hyper Binary Neural Network

Considering the capacity-limitation problem in current BNNs, we redesign the proxy optimization mechanism in neural network binarization. The main intuition behind our design is that we intend to alleviate the gradient-clamp problem in training. In traditional BNNs, the gradients of binary weights are clamped to 0 if the absolute value is larger than 1. It makes the network sensitive to initialization and noisy gradients. In this paper, an enhanced 1-bit weights generation mechanism with auxiliary network is introduced. Although 1-bit representation is still used for forward/backward propagation, each 1-bit weight is not optimized based on single full-precision variable anymore. Instead, we combine every 1-bit convolution function with a 32-bit auxiliary network. Specifically, for each convolution layer $G^l(W_b^l, A_b^l)$ in the main BNNs, a kernel-wise shared auxiliary neural network $D^l(z^l)$ is constructed at the beginning of training. A trainable 3-D tensor z^l with the shape of $[C_{in} * C_{out}, Z_n]$ is then defined to be the input of auxiliary network, where C_{in} indicates the number of input channels of convolution layer, C_{out} indicates the number of output channels and Z_n indicates the number of variables used for gradient-accumulation. The overall optimization process can be summarized as follows: 1) Generating 1-bit weights via proxy network; 2) Going forward-propagation and backward-propagation with the 1-bit weights and activations; 3) Accumulating the 32-bit gradient in real-valued proxy neural netwrok D^l to ensure that the Stochastic Gradient Decent (SGD) algorithm still works. By accumulating gradients in multi-variables rather than a single variable, the gradient path for each binary weight is connected to the whole kernel. Specifically, for each forward propagation, the 1-bit convolution layer G^l receives binarized kernels W_b^l generated by $D^l(z^l)$, and use it to perform convolution on input feature maps of the l^{th} layer A_b^l. Then, the outputs are processed in subsequent layers:

$$G^l(W_b^l, A_b^l) = G^l(D^l(z^l), A_b^l), \quad l \in 1, ..., L, \tag{3}$$

where l indicates the index of convolution layer in main BNNs.

To ensure the capacity of auxiliary neural network D^l, a specific multi-layer structure is exploited to generate the discrete kernels. In [16], the auxiliary HyperNetwork is constructed with two trainable full-connected(FC) layers and is shared across all layers. It can generate a slice of 32-bit tensors k_i^j with dimension $[N_{in}, N_{out}, f_{size}, f_{size}]$ at a time. The final

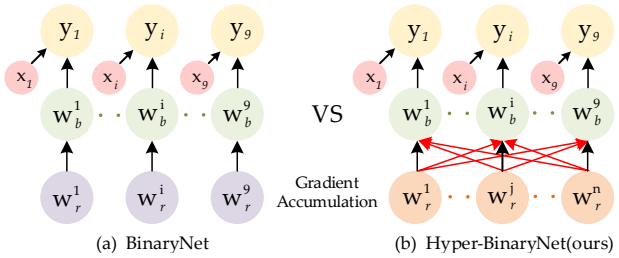

Fig. 3. Comparison of gradient-accumulation process between traditional BNNs and our Hyper-BinaryNet.

kernels k^j within a single layer is the concatenation of every k^i. In this paper, however, the discrete HyperNetwork $G(z)$ is constructed based on three components: fully-connected layer, weight normalization layer [32] and sign function. During the forward propagation, the discrete HyperNetwork $G(z)$ generates a 3-D tensor with dimension $[1, f_{size}, f_{size}]$ at a time, which is easy for parallel acceleration during the training procedure with a kernel size f_{size}.

3.3 Improved Gradient Paths via Hyper Accumulation

Considering the inefficient slow convergence speed in binary neural networks, an improved hyper-accumulation mechanism is proposed in this work. In traditional BNNs [11], [12], [13], each binary weight W_b owns one full-precision buffer W_r, and it is used to accumulate gradients to boost the learning of W_b using stochastic gradient descent algorithm. Though this simple strategy is effective, it still has a limitation: accumulating gradient in single variable is not efficient as the numerical accumulation during training can be slow and even stop because of the coarse gradient. However, in Hyper-BinaryNet, every W_b is generated by a parameterized multi-dimension linear transformation $D(z)$, therefore the main optimization target transfers into real valued auxiliary neural network D and the corresponding embedded vectors z. Compared with traditional method, gradients for each W_b are accumulated in multiple variables $W_r^1, W_r^2 ... W_r^n$. As a result, the gradient paths for each W_b are greatly increased and the tuning of W_b^i (e.g., ± 1) is not limited by single variable W_r^i anymore. For example, if both W_r^i and its gradient are negative, W_b^i will be -1 all the time. Our Hyper Accumulation accumulates every gradient of W_r^i through the Full-Connected(FC) block to update some W_b^i. We hold the point that even if the change of a single W_r^i is too small to update W_b^i, the accumulation of such tiny changes will make a difference. Additionally, owning to the fully-connected structure in auxiliary network, the generated 1-bit kernels can learn structured representations automatically. In this paper, we call this technique hyper-accumulation. For a better understanding of the hyper-accumulation, we further visualize the gradient-accumulation mechanism of our method and that used in existing BNNs methods [11], [12], [13] in Figure 3.

As we can see, the key difference between BinaryNet [13] and Hyper-BinaryNet lies in how are the binary variables $w_b^1, w_b^2 ... w_b^9$ generated. Consequently, BinaryNet [13] here can be viewed as a special case of Hyper-BinaryNet, i.e., Hyper-BinaryNet with gradients accumulated in the single variable w_r^i.

3.4 Progressive Forward Binarization

Considering the ill-posed training problem above, we propose a progressive neural architecture binarizing technique, dubbed STE-Warmup. Although BNNs have great advantages during inference stage, current BNNs are hard to converge and need much more time/samples to reduce the negative effects of the coarse gradients. This raises a direct question that shall we consistently constrain the representation in BNNs to be 1-bit during training stage? The problems in SOTA methods show that it is not necessary and even harmful for the training procedure. Therefore, we do not use the hard binarization technique like sign function at the begining of training as before, considering the ill-posed STE strategy at the beginning of training which is harmful to the convergence of binary neural network. Instead, a scalable, continuous, piecewise differentiable function is used to replace sign function. During each forward propagation, the slope of this function can be adjusted with a single scalar λ. As the scalable function shrinks itself into sign function, the weights and activations passing through it transfer from 32-bit to 1-bit smoothly. During backward propagation stage, though the evolving scalable function $F(x/\lambda)$ is theoretically full differential, it is replaced with $F(x/1)$ to escape BNNs from the bottleneck effect in backward pass. In this way, the network starts with 32-bit representations and ends with 1-bit representations. Meanwhile, the gradients are progressively approximated with a smaller average gap than the standard STE. In this paper, we use Hardtanh function as the initial continuous binarizing function, and the continuous binarizing process in forward propagation can be formulated as follows:

$$F(x, \lambda) = Hardtanh(\frac{x}{\lambda}), \tag{4}$$

$$\lim_{\lambda \to 0} Hardtanh(\frac{x}{\lambda}) = Sign(x). \tag{5}$$

Compared with existing BNNs, the gradient-mismatch problem in progressive binarizing is not obvious or does not exist at the beginning of training. Because we do not constrain the weights and activations to start from binary values [+1, -1]. Further, thanks to the continuous binarization method, we find that Hyper-BinaryNet also has more powerful representation ability during training, and can effectively relieve the capacity-limitation problem in BNNs. Additionally, a specific exponential decay strategy is designed for λ to keep the smoothness of training:

$$\lambda_t = \sigma^{(Max(0,\ t\ -\ M)//S)}, \tag{6}$$

where σ is exponential decay rate of λ, and M represents the minimal number of iterations without binarization. S indicates the exponential decay step. If there are no additional instructions, weights and activations in Hyper-BinaryNet share the same scalable continuous function and each λ is adjusted synchronously in our experiments. To escape Hyper-BinaryNet from the bottleneck effect existing continuous binarization, the gradient of scalable continuous function above is not used in backward propagation. Instead, the continuous function with λ as 1 is used. In this way, the SGD algorithm is still effective as λ in forward propagation is close to 0. For example, when $F(x)$ denotes Hardtanh function, the back-propagation process of Hyper-BinaryNet is formulated as follows:

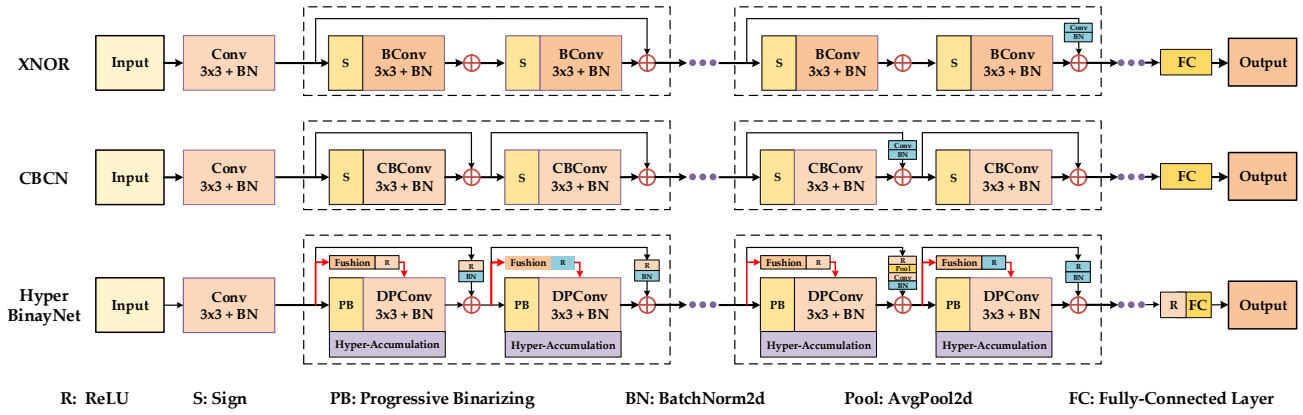

Fig. 4. Network architectures of XNOR-Net, CBCN and Hyper-BinaryNet on ResNet18. Shortcut branch without gradient-mismatch problem are further enhanced in Hyper-BinaryNet.

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial (F(x))} \cdot \frac{\partial (F(x))}{\partial x} = \frac{\partial \mathcal{L}}{\partial (F(x))} \cdot 1_{\|F(x)\| \leq 1}. \quad (7)$$

Consequently, BNNs start from real-valued DNNs without capacity limitation and gradient-mismatch problems, but end with typical BNNs with standard STE. We call this dynamical gradient approximation procedure as STE-Warmup, Algorithm.1.

Algorithm 1 Straight Through Estimation Warmup.

Input: Traning dataset; full-precision W; learning rate α and scaled scalar λ;
Output: Binarized neural network;

```
Initialize W randomly;
for t = 0 → T do
    // Forward Propagation;
    for l = 1 → L do
        // 2D Convolution with continuous binarizing
        A^{l+1} = G^l(F(W^l, λ_t), F(A^l, λ_t)); //with Eq.4
    end for
    //Backward Propagation
    for l = L → 1 do
        calculate gradients δ_{W_t^l}; //with Eq.7
        W_{t+1}^l ← W_t^l − α_t * δ_{W_t^l};
    end for
    Adjust current learning rate α_t and scalable scalar λ_t;
end for
```

3.5 Dual Path Backward Propagation

As introduced above, real-valued DNNs with STE-Warmup gradually transfer themselves into typical BNNs during training. However, the gradient-mismatch problem still exists as λ is close to 0. To further approach the gradient-mismatch problem, we design a new binary convolution module, dubbed DPConv. The key idea is to construct top-down dual path back-propagation mechanism. By propagating gradients through extra path, weights can be adjusted using accurate feedback from loss function. Figure 5 is a basic example of the DPConv module. Except the typical 1-bit convolution in BNNs (blue lines), an extra full-precision shortcut branch (red line) between the real-valued input feature maps and the output feature maps is constructed. In the shortcut branch, we first add all input images through channel (Channel-wise fusion) to get an one channel feature map and then convolute the feature map with a 32-bit kerner. Finally, we concatenate it with results obtained by the binary convolution. Consequently, traditional 1-bit convolution is reformulated as a learning function with respect to 32-bit inputs/weights again, and gradient can be propagated through real-valued branch without gradient-mismatch problem existing in ill-posed binary branch. Though float-point computation is further involved, the extra computation consumption is negligible. Specifically, let W represent the real-valued trainable kernels with shape [O $\times$ M $\times$ 3 $\times$ 3], where O is the number of output and M is the number of input channels. A represents the input activations of convolution with shape [N $\times$ C $\times$ H $\times$ W], and U denotes M$\times$*O represents the layer index in BNNs. We can get a compact representation of W and A as follows:

$$\widetilde{W^l} = \frac{1}{U}\sum_{\mathrm{i}=1}^{U}(W_i^l), \quad \widetilde{A^l} = \frac{1}{C}\sum_{\mathrm{j}=1}^{C}(A_j^l), \quad (8)$$

where i and j represent channel index of W and A respectively. Consequently, each dual path convolution can be formulated as follows:

$$A^{l+1} = G(F(W^l), F(A^l)) \amalg G(\widetilde{W^l}, \widetilde{A^l}). \quad (9)$$

$\amalg$ is the channel-wise concatenation operation. G is traditional convolution function and F represents the continuous function combined with STE-Warmup strategy. Owning to the existence of real-valued shortcut branch, the DPConv module also provides a flexible capacity compensation mechanism in bit-transition process, further smoothing the STE-Warmup process.

3.6 Enhanced Shortcut Propagation

The shortcut branch without gradient-mismatch problem is also not fully exploited in traditional BNNs and we design a new strategy to enhance it. Specifically, in current BNNs, 1-bit convolution layer with binary weights and inputs generates integer outputs. The integer outputs are then transfered into float-point values through BatchNorm layer. However,

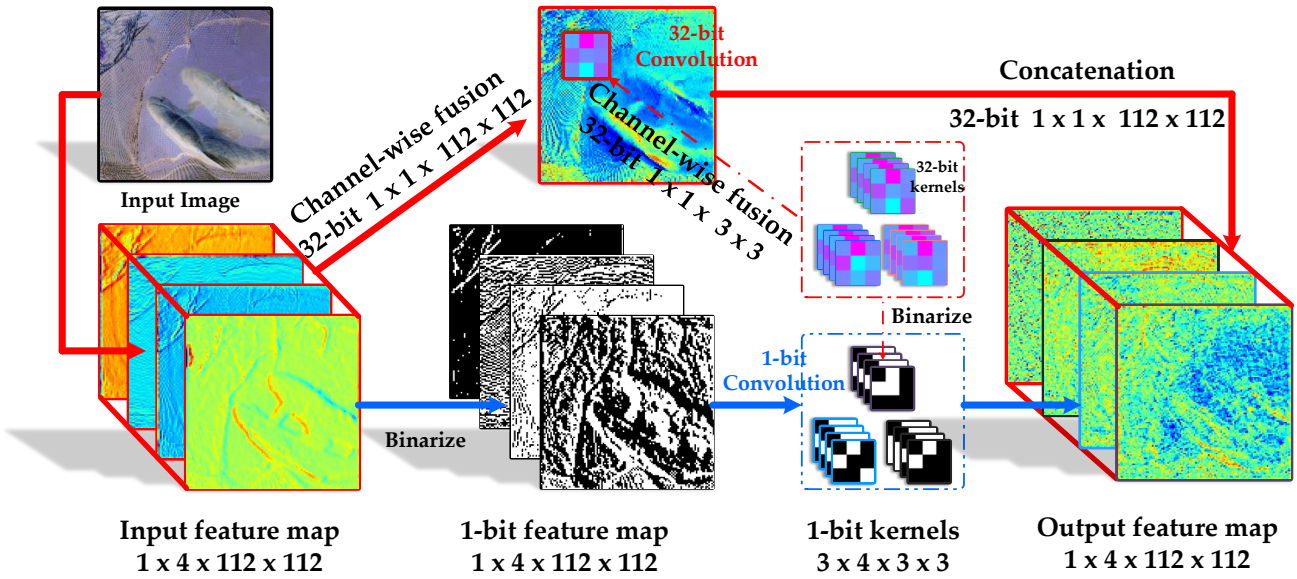

Fig. 5. Example of the dual path convolution module.

the float-point values are binarized to ± 1 again after sign function. Empirically, the most efficient way to propagate information in integers or real activations to subsequent layers is shortcut connection. As we can see in Figure 4, existing SOTA methods have improved BNNs by doubling the shortcut propagation. In this way, information containing in integers or real activations is reused. We call BNNs with such structure as "Bi-Real BNNs", which means informations in BNNs is propagated and processed in two different branches.

However, compared with the binary branch, current methods do not fully exploit the information in real-valued shortcut branches. Based on this observation, we design a new structure of shortcut propagation. As shown in Figure 4. Firstly, we double the shortcut branch to fully exploit the real activations; Secondly, we note that the nonlinear layer is missing in current shortcut strategy, which means reduced information processing capability. Empirically, we find that adding an ReLU layer before the 1×1 convolution in shortcut branch and the last fully-connected layer is beneficial. Thirdly, the 1x1 convolution in the downsampling block ignores $3/4$ of input real-valued feature maps, and we modify it to a 2×2 average pooling layer with a stride of 2 and a 1×1 convolution with a stride of 1.

4 Experiments

In this section ,we first introduce the hyper-parameter settings in all experiments. Then the evaluation experiments on CIFAR-10, CIFAR-100, and ImageNet-12 (ILSVRC2012) datasets are explored. Finally, a detailed ablation study is conducted to study the effectiveness of each module.

4.1 Experiment Setup

Here, the details about training are introduced. In all of our experiments, the Hyper-BinaryNet is constructed by replacing each convolution layer $G(a, w)$ in backbone model with HyperConv module $G(a, D(z))$, and then trained from scratch without leveraging any pre-trained weights. The auxiliary neural network $D(z)$ is initialized using the kaiming uniform initialization [33]. Only standard data argumentations are used in all experiments, including randomly cropping, flipping each image horizontally with probability 0.5 and normalizing. To guarantee fairness in comparison with SOTA methods, no special training skill is applied. We use AMSGrad optimizer with initial learning rate 0.005, and coefficients used for computing running averages of gradient and its square in optimizer are set as $(0.5, 0.99)$. In line with early studies [12], [23], [34], the first and last layers in main BNNs are kept to be full-precision for fair comparison. Additionally, there is no bias term used for 1-bit convolution. We use PReLU when weight-only binarized BNNs are trained. We use the multi-step learning rate scheduling strategy provided by PyTorch and drop the learning rate by 50% every 60 epochs. On ImageNet, however, the initial learning rate is modified to $3e^{-4}$ and it is dropped by 20% at specific epochs 30, 45, 60, 70, 80, 85, 90. The scaled scalar λ is initialized to be 1 and decays along with training iteration using Eq.6. 1x1 convolution layer is widely used in modern DNNs, especially the shortcut branch in ResNet. According to early studies, binarizing it can dramatically damage the information transformation between two stages. Most methods keep the weights and input activations of it to be full-precision [12], [23]. In this paper, we replace 32-bit 1×1 convolution in short-cut branch with weight binarized $3\times3/5\times5$ convolution layer for the maximization of binarization. The Top-1/5 accuracies are used as metrics. The best numerical result is selected for comparison in all experiments.

4.2 Experiments on CIFAR-10/CIFAR-100 Datasets

Datasets: The CIFAR-10 dataset consists of $60,000$ 32x32 colour images in 10 classes, and each class owns $6,000$ images. $50,000$ images are used for training and $10,000$ images are used for testing. The CIFAR-100 dataset is similar, except that it has 100 classes containing 600 images each, which means less samples per class. Specifically, there are 500 training images and 100 testing images for each class.

Backbone: On both datasets, ResNet18, ResNet34, ConvNet [11] and ShuffleNet are used as backbone. Hyper-BinaryResNet18 (HB-ResNet18), Hyper-BinaryResNet-34 (HB-ResNet34), Hyper-BinaryConvNet (HB-ConvNet), and Hyper-BinaryShuffleNet (HB-ShuffleNet) can be constructed by replacing convolution layer in models with HyperConv module. The Top-1 and Top-5 validation accuracy are used as performance metrics, and the training processes on both datasets are visualized in Figure 6. The experiments settings are the same. We use multi-step lr scheduling schemes provided by Pytorch, and drop the learning rate by 50% every 60 epochs.

CIFAR-10: As we can see in Figure 6, on CIFAR-10 dataset, both full binarized ResNet18 and ResNet34 gain 85% Top-1 accuracy within 10 epochs. If only the weights are binarized, a faster convergence speed is observed. But in traditional BNNs, obtaining similar accuracy usually needs several times of training epochs, resulting from the inefficient ill-posed discrete optimization, which is not the case in Hyper-BinaryNet. The main reasons are in two aspects: (1) Owning to hyper-accumulation, gradient paths for each w_b are greatly increased. As a result, the bottleneck effect of gradient in backward pass is well relieved and w_b can learn faster; (2) The STE-Warmup strategy provides better approximation to sign function. Therefore, less time/samples are needed to correct the coarse searching direction.

CIFAR-100: On CIFAR-100, our full binarized networks still show remarkable performance, even close to some higher precision quantization techniques (e.g., 2-bits binarization in [35] with 72.16% Top-1 accuracy). Compared with CIFAR-10, the CIFAR-100 dataset is more challenging. The samples for each class is 0.1x times of that in CIFAR-10. However, the classes need to be classified is 10x times. Whatever the STE strategy is, the approximated gradient only provides n rough searching direction. Consequently, many samples are needed to keep stableness of weights updates. However, it is not available on CIFAR-100 dataset, which explains why binarized models on CIFAR-100 dataset drop more precisions than that on CIFAR-10.

4.3 Experiments on ImageNet-12 Dataset

Dataset and hyper-parameters: The ImageNet-12 [32] is a large image classification dataset. It consists of $1,000$ classes with 1.2 million training images and $50k$ validation images. Being in line with existing methods, the ResNet-18 is used as backbone. In all experiments, it is trained for 100 epochs from scratch with a degradation of learning rate by 10% every 30 epochs.

Furthermore, to demonstrate the effectiveness of Hyper-BinaryNet on large scale ImageNet classification task, we compare Hyper-BinaryNet with the state-of-the-art methods, including BWN [11], BinaryNet [13], XNOR-Net [12], Bi-Real Net [23], ABC-Net [17], PCNN [24] and recent CBCN [38]. Two kinds of binarizing scheme with or without activation binarization are involved in the comparison. ResNet18, ResNet34 and ResNet50 are used as full precision backbone in this part. The numerical results are displayed in Table 3 and the results of SOTA methods are directly quoted from the corresponding references.

ResNet18/34: As we can see in Table 3, when only the weights are binarized, Hyper-BinaryNet achieves 66.9% Top-1 accuracy, outperforming BWN by 6.0%, PCNN by 3.3%. When both the activations and weights are binarized, Hyper-BinaryNet achieves the best Top-1 accuracy 61.7%, outperforming existing SOTA BNNs by a large margin. Specifically, Hyper-BinaryNet outperforms traditional BinaryNet, ABC-Net and XNOR-Net by 10% higher Top-1 accuracy. Compared with recent methods, whose representation ability is enhanced by doubling shortcut propagation or modified architecture, like Bi-Real Net, PCNN and CBCN with k=4, Hyper-BinaryNet still shows notable improvements. Additionally, Hyper-BinaryNet relatively outperforms existing quantization method HWPG, which owns 1-bit weights and 2-bit activations, by 2% higher Top-1 accuracy while requires less memory bandwidth. These obviously show the efficiency of Hyper-BinaryNet. We need to note that, though CBCN achieves similar Top-1 accuracy 61.4%, its computation consumption is k times higher resulting from its specific circulant structure, which is different from Hyper-BinaryNet. We also plot the loss/accuracy curve during training in Figure 8. As we can see, the training procedure is smooth and stable, which means that our STE-Warmup strategy is still helpful on large scale classification dataset. As the depth of ResNet is modified from 18-layers to 34-layers, Hyper-BinaryNet shows similar performance to methods whose weights are 1-bit and activations are 2-bit or higher bit-width.

ResNet50: We also combine Hyper-BinaryNet with ResNet50 to evaluate whether Hyper-BinaryNet can be applied to DNNs with bottleneck structure, which is a popular way of constructing compact architecture in current DNNs. As being pointed out in recent method [39], the 1x1 convolution in bottleneck structure greatly reduces the filters between layers, which means the reduced gradient paths in BNNs. Therefore, constraining it to be binary ± 1 is not suitable for BNNs and we keep it to be full-precision herein. The numerical results are presented in Table 3. When both the weights and input activations are binarized, ResNet50 with Hyper-BinaryNet works very well, achieving 69.6% Top-1 accuracy and 88.3% Top-5 accuracy respectively. When the bottleneck structure is involved, Hyper-BinaryNet still gains attractive accuracies and even outperforms some existing SOTA fixed-point quantization methods HWPG and Lq-Net with 1-bit weights and 2-bit activations. Consequently, Hyper-BinaryNet is a generic and efficient method for binarizing modern neural architectures.

4.4 Abalation Study

As introduced above, we overcome the slow convergence speed and dropping of precision problems in BNNs from the aspects of improving gradient propagation. Several modules are proposed to address the capacity-limitation problem, gradient-mismatch problem, inefficient gradient-accumulation problem in BNNs respectably. To verify how BNNs can benefit from each module, detailed ablation studies are carried out in this subsection. The backbone used for experiments here is ResNet-18, and the selected dataset is CIFAR-10.

Individual Contribution: The full precision ResNet18 with kernel stage as 16-32-64-128 is used as a backbone. We combine it with the proposed modules separately and then collectively to verify the impact of each one, and the numerical results are shown in the Table 4. As we can

TABLE 2
Top-1 accuracy on CIFAR-10/100 datasets and comparison with SOTA BNNs. "W/A" indicates bit-width of weights and activations.

Backbone	Methods	W/A(bit)	Datasets	
			CIFAR-10	CIFAR-100
ConvNet	XNOR-Net [12]	1/1	89.93	-
	BinaryNet [13]	1/1	89.85	-
	BinaryConnect [11]	1/32	91.73	-
	Ours	1/1	90.81	64.17
	Ours	1/32	92.52	68.28
ResNet-18	BinaryNet [13]	1/1	89.85	61.20
	BinaryNet [13]	1/32	91.37	66.20
	Ours	1/1	92.81	69.53
	Ours	1/32	93.23	72.40
ShuffleNet	BinaryNet [13]	1/1	86.49	59.14
	Ours	1/1	88.53	62.03

TABLE 3
Top-1/5 accuracy of Hyper-BinaryNet evaluated on ImageNet-12 dataset and comparison with other BNNs methods.

Comparison	Methods	W/A(bit)	Top-1(%)	Top-5(%)
ResNet-18	Full-Precison	32/32	69.3	89.2
ResNet18	BinaryNet [13]	1/1	42.2	67.1
	XNOR-Net [12]	1/1	51.2	73.2
	ABC-Net [17]	1/1	42.7	61.6
	Bi-Real Net [23]	1/1	56.4	79.5
	PCNN [24]	1/1	57.3	80.0
	LQ-Nets [36].	1/2	62.6	84.3
	HWGQ [34]	1/2	59.6	82.2
	DoReFa-Net [37]	1/2	53.4	-
	BWN [12]	1/32	60.8	83.0
	Ours	1/1	61.6	81.1
	Ours	1/32	66.9	84.1
ResNet34	Full-Precision	32/32	73.3	91.3
	Lq-Net [36]	1/2	66.6	86.9
	HWPG [34]	1/2	64.3	85.7
	ABC-Net [17]	3/3	66.7	88.2
	Ours	1/1	64.5	86.5
ResNet50	Full-Precision	32/32	76.0	92.9
	Lq-Net [36]	1/2	68.7	88.4
	HWPG [34]	1/2	64.6	85.9
	ABC-Net [17]	5/5	70.1	89.1
	Ours	1/1	69.8	88.4

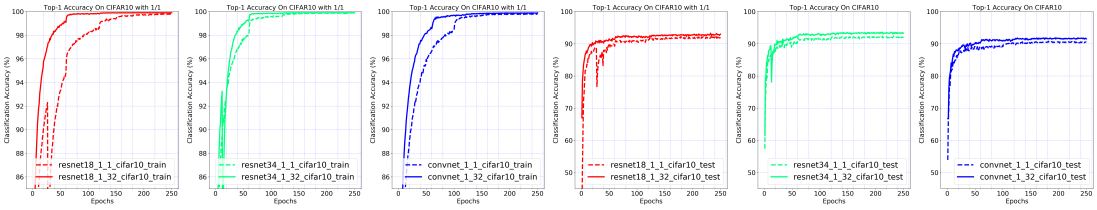

Fig. 6. The Top-1 accuracy evaluated on CIFAR-10 datasets with ResNet18, ResNet34 and ConvNet as base architecture.

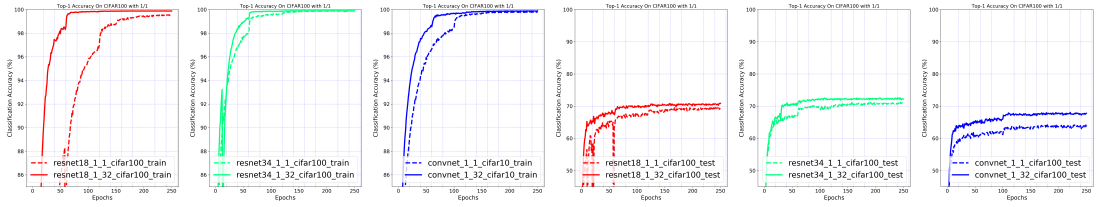

Fig. 7. The Top-1 accuracy evaluated on CIFAR-100 datasets with ResNet18, ResNet34 and ConvNet as base architecture.

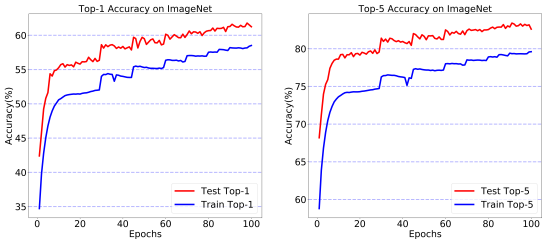

Fig. 8. Training procedure on ImageNet dataset with ResNet18 as backbone. Best viewed in color.

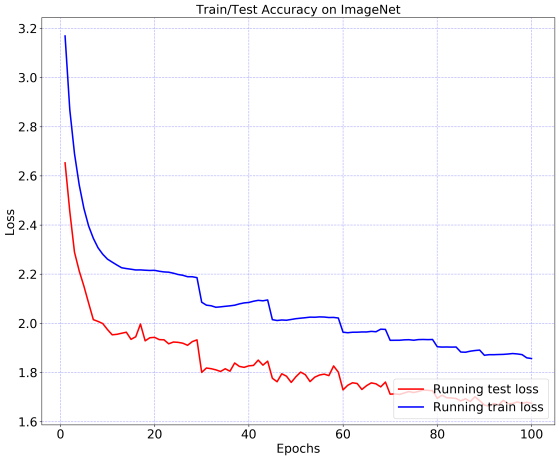

Fig. 9. Training procedure on ImageNet dataset with ResNet18 as backbone. Best viewed in color.

see in Table 4, all proposed modules are beneficial to the convergence of BNNs. Specifically, binary ResNet-18 with hyper-accumulation achieve 82.1% Top-1 accuracy on CIFAR10 dataset and the STE-Warmup strategy also gains 81.%. The numerical improvements show that the inefficient optimization technique in traditional BNNs is harmful to convergence. An improved backward propagation can indeed make BNNs gain better performance. Hence, we further combine BNNs with the other proposed methods. A total 87.1% Top-1 accuracy is gained, which outperforms BinaryNet [13] 76.9% by about 10% higher Top-1 accuracy. The numerical improvements also indicate that traditional BNNs have not fully exploited the full-precision branch existing in BNNs. How to construct automatic trade-off between full-precision operations and binary operations is the next key direction for binary neural networks.

Gradient Approximation: A progressive binarizing strategy is exploited in this paper. To provide a closer understand about the effectiveness of this module, detailed ablation study from two aspects are carried in this part. We maily focus on the impact of this module and the progressiveness speed of it during training stage. The full precision ResNet18 with kernel stage as 16-32-64-128 is used as a backbone. In Table 5, we compare the Top-1 accuracy of continuous binarizing with and without Warmup using different exponential decay rate σ in Eq. 6. In Figure 10, we further plot the training and testing Top-1 accuracy curve. As shown in Table 5, the convergence of BNNs using pure continuous binarizing function stops as the scalable scalar λ shrinks. Such phenomenon is called the

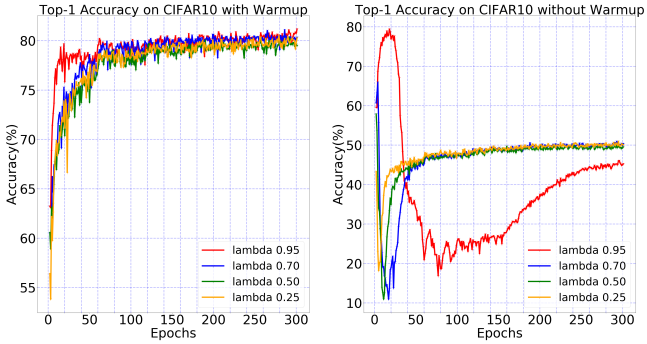

Fig. 10. Training procedure of continuous binarizing with and without warmup technique. Best viewed in color.

bottleneck effect in back propagation, but it is not the case in STE-Warmup. And in both cases, the excessive capacity change can harm the final performance. Table 5 shows that 0.95 is a suitable exponential decay rate for Hyper-BinaryNet on CIFAR10 dataset. In general, our Hyper-BinaryNet owns better Top-1 accuracy and more stable convergence speed.

Auxiliary Capacity: We also investigate whether the capacity of auxiliary neural network can influence the final performance. The width and depth of auxiliary neural network are considered in this part. The width indicates the dimension of input embedded vector for each kernel and is changed from 3 to 36. The depth indicates the number of fully-connected layer in each auxiliary network and is adjusted from 1 to 2. The results are presented in Table 6. In most cases (6-36), Hyper-BinaryNet with different width shares similar accuracy ($\pm$0.23%). But too deep HyperNetwork can obviously hurt final performance. This can be resulted from gradient vanishing problem. In general, the best accuracy is always got when the width of HyperNetwork is the same as vectorized binary kernel, e.g., 9.

Shortcut Binarization: The binarization strategy for shortcut branch in BNNs is to further investigate the design principle of BNNs. Three kinds of convolutions and the corresponding binarizing strategies are collectively explored and results are presented in Table 7. As we can see, keeping shortcut branch to be full-precision can achieve the best Top-1 performance in all related experiments but weight-only binarizing can reach close performance(within 1% accuracy declined). This is not investigated in early research and the weight-only binarized module can be implemented with addition and subtraction operations [12]. Furthermore, the 3×3/5×5 convolutions are investigated in this part. The numerical performance show that larger kernel size is better than normally used 1x1 convolution in all experiments. Therefore, the shortcut strategy in BNNs shall be different from that in DNNs. We believe that combining recent neural architecture searching techniques [40] with BNNs is a better choice.

4.5 Computational Cost

In our work, the Hyper-BinaryNet and STE-Warmup strategy are only used during training phase and has nothing to do with the inference phase. The enhanced shortcut propagation is a lightweight operation, and needs marginal computation. Thus, only the DPConv module involves full-precision convolution. However, its computation load is also marginal.

TABLE 4
Top-1 accuracy on CIFAR10 dataset. "BaseLine" indicates ResNet18 with kernel stage as 16-32-64-128, "H" indicates auxiliary neural network based hyper accumulation, "W" indicates STE-Warmup strategy, "D" indicates DPConv module, "S" indicates enhanced shortcut propagation.

Methods	BaseLine	H	W	D	S	W+D	W+D+S	W+D+S+H
Top-1 Accuracy	76.9	82.1	81.4	83.2	84.6	84.1	86.8	87.1
Top-5 Accuracy	-	99.2	99.0	99.4	99.6	99.2	99.4	99.6

TABLE 5
Top-1 accuracy comparison with different exponential decay rate δ on CIFAR10 dataset.

Optimization Methods	Exponential Decay Rate δ			
	0.95	0.70	0.50	0.25
Soft Binarizing W/O Warmup	46.5	49.8	49.4	50.8
STE-Warmup	81.4	79.9	79.1	78.8

TABLE 6
Top-1 accuracy comparison on CIFAR-10 dataset with different depth and width of auxiliary network

Models	Depth	Variables Per Binary Kernel					
		3	6	9	18	27	36
Binary ResNet-18	1	92.32	92.42	92.81	92.62	92.35	92.21
	2	92.13	92.33	92.51	92.45	92.34	92.05

The DPConv module constructs a full-precision branch which requires very low computational cost: convolution on a one-channel feature map with one kernel. We compare the FLOPs among different binary methods during inference phase, and the results are shown in Table 8. A full-precision Resnet18 with kernel stage 64-128-256-512 is used as backbone.

5 Conclusion and Future Works

In this paper, we investigate a new framework of binary neural networks, called as Hyper-BinaryNet. An auxiliary capacity strategy is investigated to address the capacity limitation problem in BNNs. The traditional proxy optimization technique in BNNs is proved to be a special case of the proposed hyper-accumulation. A progressive binarization technique STE-Warmup scheme and a light-weight DPConv module are studied to alleviate the gradient mismatch problem in BNNs. And the bottleneck effect of gradient in traditional continuous discretization methods and the iterative optimization process does not exist in our STE-Warmup procedure. Owning to the reasonable design principle, the Hyper-BinaryNet algorithm has outperformed existing state-of-the-art methods on CIFAR10/100 and ImageNet classification tasks by a large margin. As a generic BNNs design framework, Hyper-Binary can be easily applied to other modern neural architectures and tasks like object detection and image segmentation, which will be studied in future works.

TABLE 7
Top-1 accuracy comparison on CIFAR10 dataset. Different binarizing strategies on shortcut branch are used.

Binarizing Strategies	kernel size	CIFAR-10	CIFAR-100
Full Binarizatiion	1x1	90.28	64.39
	3x3	91.31	65.87
	5x5	91.03	65.87
Weight Binarizatiion	1x1	90.84	65.73
	3x3	92.81	68.52
	5x5	91.49	68.25
Without Binarizatiion	1x1	92.35	65.90
	3x3	92.86	69.53
	5x5	92.95	68.85

6 Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 61773316 and 61871470.

References

[1] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 4, pp. 834–848, 2017.

[2] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," arXiv preprint arXiv:1809.11096, 2018.

[3] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on. IEEE, 2016, pp. 4960–4964.

[4] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina et al., "State-of-the-art speech recognition with sequence-to-sequence models," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 4774–4778.

[5] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning." in AAAI, vol. 2. Phoenix, AZ, 2016, p. 5.

[6] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in International conference on machine learning, 2016, pp. 1928–1937.

[7] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," arXiv preprint arXiv:1612.01064, 2016.

[8] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," The Journal of Machine Learning Research, vol. 18, no. 1, pp. 6869–6898, 2017.

[9] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaev, G. Venkatesh et al., "Mixed precision training," arXiv preprint arXiv:1710.03740, 2017.

[10] D. Das, N. Mellempudi, D. Mudigere, D. Kalamkar, S. Avancha, K. Banerjee, S. Sridharan, K. Vaidyanathan, B. Kaul, E. Georganas et al., "Mixed precision training of convolutional neural networks using integer operations," arXiv preprint arXiv:1802.00930, 2018.

[11] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in Advances in neural information processing systems, 2015, pp. 3123–3131.

TABLE 8
The number of Floating-point and binary operations in all convolution layers among different binary methods.

Backbone	Methods	The number of operations	
		FLOPs	Binary Operation
ResNet-18	BinaryNet [13]	0	73M
	XNOR-Net [12]	0.1M	73M
	Ours	0.03M	71M

[12] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in European Conference on Computer Vision. Springer, 2016, pp. 525–542.

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

[14] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "Wrpn: wide reduced-precision networks," arXiv preprint arXiv:1709.01134, 2017.

[15] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," arXiv preprint arXiv:1308.3432, 2013.

[16] D. Ha, A. M. Dai, and Q. V. Le, "Hypernetworks," CoRR, vol. abs/1609.09106, 2016. [Online]. Available: http://arxiv.org/abs/1609.09106

[17] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in Advances in Neural Information Processing Systems, 2017, pp. 345–353.

[18] M. D. McDonnell, "Training wide residual networks for deployment using a single bit for each weight," arXiv preprint arXiv:1802.08530, 2018.

[19] Z. Wang, J. Lu, C. Tao, J. Zhou, and Q. Tian, "Learning channel-wise interactions for binary convolutional neural networks," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 568–577.

[20] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia, "BNN+: Improved binary network training," 2019. [Online]. Available: https://openreview.net/forum?id=SJfHg2A5tQ

[21] B. Martinez, J. Yang, A. Bulat, and G. Tzimiropoulos, "Training binary neural networks with real-to-binary convolutions," 2020.

[22] D. Soudry, I. Hubara, and R. Meir, "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights," in Advances in Neural Information Processing Systems, 2014, pp. 963–971.

[23] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng, "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm," in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 722–737.

[24] J. Gu, C. Li, B. Zhang, J. Han, X. Cao, J. Liu, and D. Doermann, "Projection convolutional neural networks for 1-bit cnns via discrete back propagation," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 8344–8351.

[25] S. Zhu, X. Dong, and H. Su, "Binary ensemble neural network: More bits per network or more networks per bit?" in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4923–4932.

[26] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," arXiv preprint arXiv:1611.01144, 2016.

[27] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in Proceedings of the IEEE international conference on computer vision, 2017, pp. 5608–5617.

[28] C. Sakr, J. Choi, Z. Wang, K. Gopalakrishnan, and N. Shanbhag, "True gradient-based training of deep binary activated neural networks via continuous binarization," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 2346–2350.

[29] F. Lahoud, R. Achanta, P. Márquez-Neila, and S. Süsstrunk, "Self-binarizing networks," arXiv preprint arXiv:1902.00730, 2019.

[30] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Smash: one-shot model architecture search through hypernetworks," arXiv preprint arXiv:1708.05344, 2017.

[31] C. Zhang, M. Ren, and R. Urtasun, "Graph hypernetworks for neural architecture search," arXiv preprint arXiv:1810.05749, 2018.

[32] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F. F. Li, "Imagenet: A large-scale hierarchical image database," in IEEE Conference on Computer Vision & Pattern Recognition, 2009.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, 2015, pp. 1026–1034.

[34] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," arXiv preprint arXiv:1702.00953, 2017.

[35] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," arXiv preprint arXiv:1802.05668, 2018.

[36] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 365–382.

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

[38] C. Liu, W. Ding, X. Xia, B. Zhang, J. Gu, J. Liu, R. Ji, and D. Doermann, "Circulant binary convolutional networks: Enhancing the performance of 1-bit dcnns with circulant back propagation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2691–2699.

[39] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, "Structured binary neural networks for accurate image classification and semantic segmentation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 413–422.

[40] B. Zoph and Q. V. le, "Neural architecture search with reinforcement learning," arXiv preprint arXiv:1611.01578, 2016.

Qi Wang (M'15-SM'15) received the B.E. degree in automation and the Ph.D. degree in pattern recognition and intelligent systems from the University of Science and Technology of China, Hefei, China, in 2005 and 2010, respectively. He is currently a Professor with the School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi'an, China. His research interests include computer vision and pattern recognition.

Nianhui Guo received the B.E. degree in Communication Engineering Specialty from the University of Nanchang, Nan'Chang, China, in 2018. He is currently studying for a master's degree in the School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi'an, China. His research interests include computer vision and pattern recognition, compression and acceleration of deep learning models, image semantic analysis.

Zhitong Xiong received the M.E. degree in Northwestern Polytechnical University and is currently working toward the Ph.D. degree with the School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi'an, China. His research interests include computer vision and machine learning.

Zeping Yin received the B.E. degree in Communication Engineering Specialty from the University of Chang'an University, Xi'an, China, in 2020. He is currently studying for a master's degree in the School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi'an, China. His research interests include computer vision and pattern recognition, compression and acceleration of deep learning models, image semantic analysis.

Xuelong Li (M'02-SM'07-F'12) is currently a Full Professor with the School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi'an 710072, China.