Binary Neural Networks: A Survey

Haotong Qin[a], Ruihao Gong[a], Xianglong Liu*[a,b], Xiao Bai[e], Jingkuan Song[c], Nicu Sebe[d]

[a] *State Key Lab of Software Development Environment, Beihang University, Beijing, China.*
[b] *Beijing Advanced Innovation Center for Big Data-Based Precision Medicine, Beihang University, Beijing, China.*
[c] *Center for Future Media and School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.*
[d] *Department of Information Engineering and Computer Science, University of Trento, Trento, Italy.*
[e] *School of Computer Science and Engineering, Beijing Advanced Innovation Center for Big Data and Brain Computing, Jiangxi Research Institute, Beihang University, Beijing, China*

Abstract

The binary neural network, largely saving the storage and computation, serves as a promising technique for deploying deep models on resource-limited devices. However, the binarization inevitably causes severe information loss, and even worse, its discontinuity brings difficulty to the optimization of the deep network. To address these issues, a variety of algorithms have been proposed, and achieved satisfying progress in recent years. In this paper, we present a comprehensive survey of these algorithms, mainly categorized into the native solutions directly conducting binarization, and the optimized ones using techniques like minimizing the quantization error, improving the network loss function, and reducing the gradient error. We also investigate other practical aspects of binary neural networks such as the hardware-friendly design and the training tricks. Then, we give the evaluation and discussions on different tasks, including image classification, object detection and semantic segmentation. Finally, the challenges that may be faced in future research are prospected.

Keywords: binary neural network, deep learning, model compression, network quantization, model acceleration

*Corresponding author

1. Introduction

With the continuous development of deep learning [1], deep neural networks have made significant progress in various fields, such as computer vision, natural language processing and speech recognition. Convolutional neural networks (CNNs) have been proved to be reliable in the fields of image classification [2, 3, 4, 5, 6], object detection [7, 8, 9, 10] and object recognition [11, 12, 2, 13, 14], and thus have been widely used in practice.

Owing to the deep structure with a number of layers and millions of parameters, the deep CNNs enjoy strong learning capacity, and thus usually achieve satisfactory performance. For example, the VGG-16 [12] network contains about 140 million 32-bit floating-point parameters, and can achieve 92.7% top-5 test accuracy for image classification task on ImageNet dataset. The entire network needs to occupy more than 500 megabytes of storage space and perform 1.6×10^{10} floating-point arithmetic operations. This fact makes the deep CNNs heavily rely on the high-performance hardware such as GPU, while in the real-world applications, usually only the devices (*e.g.* , the mobile phones and embedded devices) with limited computational resources are available [15]. For example, embedded devices based on FPGAs usually have only a few thousands of computing units, far from dealing with millions of floating-point operations in the common deep models. There exists a severe contradiction between the complex model and the limited computational resources. Although at present, a large amount of dedicated hardware emerges for deep learning [16, 17, 18, 19, 20], providing efficient vector operations to enable fast convolution in forward inference, the heavy computation and storage still inevitably limit the applications of the deep CNNs in practice. Besides, due to the huge model parameter space, the prediction of the neural networks is usually viewed as a black-box, which brings great challenges to the interpretability of CNNs. Some works like [21, 22, 23] empirically explore the function of each layer in the network. They visualize the feature maps extracted by different filters and view each filter as a visual unit focusing on different visual components.

From the aspect of explainable machine learning, we can summarize that some filters are playing a similar role in the model, especially when the model size is large. So it is reasonable to prune some useless filters or reduce their precision to lower bits. On the one hand, we can enjoy more efficient inference with such compression technique. On the other hand, we can utilize it to further study the interpretability of CNNs, *i.e.* , finding out which layer is important, which layer is useless and can be removed from the black-box, what structure is beneficial for accurate prediction. Many prior studies have proven that there usually exists large redundancy in the deep structure [24, 25, 26, 27]. For example, by simply discarding the redundant weights, one can keep the performance of the ResNet-50 [28], and meanwhile save more than 75% of parameters and 50% computational time. In the literature, approaches for compressing the deep networks can be classified into five categories: parameter pruning [26, 29, 30, 31], parameter quantizing [32, 33, 34, 35, 36, 37, 38, 39, 40, 41], low-rank parameter factorization [42, 43, 44, 45, 46], transferred/compact convolutional filters [47, 48, 49, 50], and knowledge distillation [51, 52, 53, 54, 55, 56]. The parameter pruning and quantizing mainly focus on eliminating the redundancy in the model parameters respectively by removing the redundant/uncritical ones or compressing the parameter space (*e.g.* , from the floating-point weights to the integer ones). Low-rank factorization applies the matrix/tensor decomposition techniques to estimate the informative parameters using the proxy ones of small size. The compact convolutional filter based approaches rely on the carefully-designed structural convolutional filters to reduce the storage and computation complexity. The knowledge distillation methods try to distill a more compact model to reproduce the output of a larger network.

Among the existing network compression techniques, quantization based one serves as a promising and fast solution that yields highly compact models compared to their floating-point counterparts, by representing the network weights with very low precision. Along this direction, the most extreme quantization is binarization, the interest in this survey. Binarization is a 1-bit quantization where data can only have two possible values, namely -1(0) or +1. For network compression, both the weight and activation can be represented by 1-bit without taking too much memory. Besides, with the binarization, the heavy matrix multiplication operations can be replaced with light-weighted bitwise XNOR operations and Bitcount operations. Therefore, compared with other compression methods, binary neural networks enjoy a number of hardware-friendly properties including memory saving, power efficiency and significant acceleration. The pioneering work like BNN [57] and XNOR-Net [58] has proven the effectiveness of the binarization, namely, up to 32× memory saving and 58× speedup on CPUs, which has been achieved by XNOR-Net for a 1-bit convolution layer. Following the paradigm of binary neural network, in the past years a large amount of research has been attracted on this topic from the fields of computer vision and machine learning [1, 2, 12, 28], and has been applied to various popular tasks such as image classification[59, 60, 61, 62, 63], detection [64, 65], and so on. With the binarization technique, the importance of a layer can be easily validated by switching it to full-precision or 1-bit. If the performance greatly decreases after binarizing certain layer, we can conclude that this layer is on the critical path of the network. Furthermore, it is also significant to find out whether the full-precision model and the binarized model work in the same way from the explainable machine learning view.

Besides focusing on the strategies of model binarization, many studies have attempted to reveal the behaviors of model binarization, and further explain the connections between the model robustness and the structure of deep neural networks. This possibly helps to approach the answers to the essential questions: how does the deep network work indeed and what network structure is better? It is very interesting and important to well investigate the studies of binary neural network, which will be very beneficial for understanding the behaviors and structures of the efficient and robust deep learning models. Some of studies in the literature have shown that binary neural networks can filter the input noise, and pointed out that specially designed BNNs are more robust compared with the full-precision neural networks. [66] shows that noise is continuously amplified during the forward propagation of neural networks, and binarization improves robustness by keeping the magnitude of the noise small.

The studies based on BNNs can also help us to analyze how structures in deep neural networks work. Liu *et.al.* creatively proposed Bi-Real Net, which added additional shortcuts (Bi-Real) to reduce the information loss caused by binarization [62]. This structure works like the shortcut in ResNet and it helps to explain why the widely used shortcuts can improve performance of deep neural networks to some extent. On the one hand, by visualizing the activations, it can be seen that more detailed information in the shallow layer can be passed to the deeper layer during forward propagation. On the other hand, gradients can be directly backward propagated through the shortcut to avoid gradient vanish problem. Zhu *et.al.* leveraged ensemble methods to improve the performance of BNNs by building several groups of weak classifiers, and the ensemble methods improve the performance of BNNs although sometimes face over-fitting problem [67]. Based on analysis and experimentation of BNNs, they showed that the number of neurons is more important than the bit-width and it may not be necessary to use real-valued neurons in deep neural networks, which is similar to the principle of biological neural networks. Besides, reducing the bit-width of certain layer to explore its effect on accuracy is one effective approach to study the interpretability of deep neural networks. There are many works to explore the sensitivity of different layers to binarization. It is a common sense that the first layer and the last layer should be kept in higher precision, which means that these layers play a more important role in the prediction of neural networks.

This survey tries to exploit the nature of binary neural networks and categorizes the them into the naive binarization without optimizing the quantization function and the optimized binarization including minimizing quantization error, improving the loss function, and reducing the gradient error. It also discusses the hardware-friendly methods and the useful tricks of training binary neural networks. In addition, we present the common datasets and network structures of evaluation, and compare the performance of current methods on different tasks. The organization of the remaining part is given as the following. Section 2 introduces the preliminaries for binary neural network. Section 3 presents the existing methods falling in different categories and lists the training tricks in practice. Section 4 gives the evaluation protocols and performance analysis. Finally, we conclude and point out the future research trends in Section 5.

2. Preliminary

In full-precision convolutional neural networks, the basic operation can be expressed as

$$\mathbf{z} = \sigma(\mathbf{w} \otimes \mathbf{a}) \tag{1}$$

where $\mathbf{w}$ and $\mathbf{a}$ represent the weight tensor and activation tensor generated by the previous network layer, respectively. $\sigma(\cdot)$ is the non-linear function and $\mathbf{z}$ is the output tensor and $\otimes$ represents the convolution operation. In the forward inference process of neural networks, the convolution operation contains a large number of floating-point operations, including floating-point multiplication and floating-point addition, which correspond to the vast majority of calculations in neural network inference.

2.1. Forward Propagation

The goal of network binarization is to represent the floating-point weights $\mathbf{w}$ and/or activations $\mathbf{a}$ using 1-bit. The popular definition of the binarization function is given as follows:

$$Q_w(\mathbf{w}) = \alpha \mathbf{b}_{\mathbf{w}}, \quad Q_a(\mathbf{a}) = \beta \mathbf{b}_{\mathbf{a}} \tag{2}$$

where $\mathbf{b}_{\mathbf{w}}$ and $\mathbf{b}_{\mathbf{a}}$ are the tensor of binary weights (kernel) and binary activations, with the corresponding scalars α and β. In the literature, the `sign` function is widely used for Q_w and Q_a:

$$\texttt{sign}(x) = \begin{cases} +1, & \text{if } \ x \geq 0 \\ -1, & \text{otherwise} \end{cases} \tag{3}$$

With the binarized weights and activations, the vector multiplication in forward propagation can be reformulated as

$$\mathbf{z} = \sigma(Q_w(\mathbf{w}) \otimes Q_a(\mathbf{a})) = \sigma(\alpha\beta(\mathbf{b_w} \odot \mathbf{b_a})), \tag{4}$$

where $\odot$ denotes the inner product for vectors with bitwise operation XNOR-Bitcount. Figure 1 shows the convolution process in the binary neural networks.

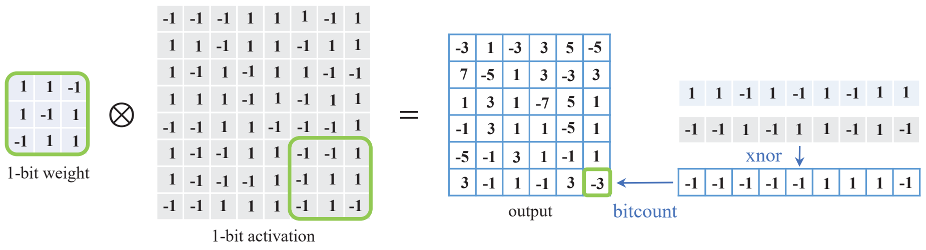

Figure 1: Convolution Process of Binary Neural Networks

2.2. Backward Propagation

Similar to training a full-precision neural network model, when training a binary neural network, it is still straightforward to adopt the powerful backward propagation (BP) algorithm based on the gradient descent to update the parameters. However, usually the binarization function (*e.g.* , `sign`) is not differentiable, and even worse, the derivative value in part of the function vanishes (*e.g.* , 0 almost everywhere for `sign`). Therefore, the common gradient descent based BP algorithm cannot be directly applied to update the binary weights.

Fortunately, the technique called straight-through estimator (STE) has been proposed by Hinton *et.al.* to address the gradient problem occurring when training deep networks binarized by `sign` function [68]. The function of STE is defined as follows

$$\texttt{clip}(x, -1, 1) = \max(-1, \min(1, x)). \tag{5}$$

Through STE, the binary neural network can be directly trained using the same gradient descent method as the ordinary full-precision neural network. However, when the `clip` function is used in backward propagation, if the absolute value of full-precision activations are greater than 1, it cannot be updated in backward propagation. Therefore, in the practical scenarios, the `Identity` function is also chosen to approximate the derivative of the `sign` function.

3. Binary Neural Networks

Compared with the full-precision neural network, the binary neural networks based on 1-bit representation replace the floating-point multiplication and addition operations by the efficient XNOR-Bitcount operations, and thus largely reduce the storage space and the inference time. However, the binarization of weights and activations will cause a severe deviation from the full-precision ones. Also, as aforementioned, the discrete binarization makes the popular gradient descent based BP algorithm usually fail to pursue the satisfactory solution, even with the STE technique. Therefore, the binary neural networks inevitably suffer from the performance degradation. It is still an open research problem that how to optimize the binary neural network.

In recent years, a variety of binary neural networks have been proposed, from the native solutions that directly binarize the weights and inputs using the pre-defined binarization function, to the optimization based ones using different techniques that treat the problem from different perspectives: approximate the full-precision values by minimizing the quantization error, constrain the weights by modifying the network loss function, and learn the discrete parameters by reducing the gradient error. Table 1 summaries the surveyed binarization methods in different categories.

3.1. Naive Binary Neural Networks

The naive binary neural networks directly quantize the weights and activations in the neural network to 1-bit by the fixed binarization function. Then the basic backward propagation strategy equipped with STE is applied to optimize the deep models in the standard training way.

In 2016 Courbariaux *et.al.* proposed BinaryConnect [59] that pioneered the study of binary neural networks. BinaryConnect converts the full-precision

Table 1: Overview of Binary Neural Networks

Type		Method	Key Tech.	Tricks			
				ST	OP	AQ	GA
Naive Binary Neural Networks		BinaryConnect [59]	FP: $\texttt{sign}(x)$ BP: STE	-	A	-	-
		Bitwise Neural Networks [69]		-	-	-	-
		Binarized Neural Networks [57]		-	AM	-	-
Optimization Based Binary Neural Networks	Minimize the Quantization Error	Binary Weight Networks [57]	$J(\mathbf{b},\alpha) = \|\mathbf{x}-\alpha\mathbf{b}\|^2$ $\alpha^*, \mathbf{b}^* = \arg\min_{\alpha,\mathbf{b}} J(\mathbf{b},\alpha)$	-	S	-	-
		XNOR-Net [58]		RB+RP	A	-	-
		DoReFa-Net [60]		-	A	-	-
		High-Order Residual Quantization [70]		-	A	-	-
		ABC-Net [71]		-	S	-	-
		Two-Step Quantization [72]		RB	-	-	-
		Binary Weight Networks via Hashing[73]		-	S	-	-
		PArameterized Clipping acTivation [74]		-	A	-	-
		LQ-Nets [61]		RB	-	-	-
		Wide Reduced-Precision Networks [75]		WD	A	-	-
		XNOR-Net++ [76]		-	A	-	-
		Learning Symmetric Quantization [77]		-	-	✓	-
		BBG [78]		SC	-	-	-
		Real-to-Bin [79]		SC	A	-	✓
	Improve Network Loss Function	Distilled Binary Neural Network [80]	$\mathcal{L}^b_{\text{total}} = \mathcal{L}^b_{\text{original}} + \lambda\mathcal{L}^b_{\text{Customized}}$	-	S	-	-
		Distillation and Quantization [81]		-	S	-	-
		Apprentice [82]		-	-	-	-
		Loss-Aware Binarization [83]		-	A	-	-
		Incremental Network Quantization [84]		-	S	✓	-
		BNN-DL [85]		-	R	-	✓
		CI-BCNN [86]		-	R	-	✓
		Main/Subsidiary Network [87]		RB	-	-	-
	Reduce the Gradient Error	Bi-Real Net [62]	Customized `ApproxFunc` (FP) or `QuantFunc` (BP) or `UpdateFunc` (BP)	SC	S	-	✓
		Circulant Binary Convolutional Networks[88]		SC	S	-	✓
		Half-wave Gaussian Quantization [89]		RB	S	-	✓
		BNN+ [90]		RB	A	-	✓
		Differentiable Soft Quantization [63]		-	A	-	✓
		BCGD [91]		-	-	-	✓
		ProxQuant [92]		-	A	-	✓
		Quantization Networks [93]		-	S	-	✓
		Self-Binarizing Networks [94]		-	A	-	✓
		Improved Training BNN [95]		-	A	-	✓
		IR-Net [96]		-	S	✓	✓

* Tech. = Technology. Tricks: ST = Structure Transformation, OP = Optimizer, AQ = Asymptotic Quantization, GA = Gradient Approximation. Optimizer: S = SGD, A = Adam, AM = AdaMax, R = RMSprop. Structure Transformation: RB = Reorder BN layer, RP = Reorder Pooling layer, WD = Widen, SC = Shortcut. FP = Forward Propagation, BP = Backward Propagation.

weights inside the neural network into 1-bit binary weights. In the forward propagation of training, a stochastic binarization method is adopted to quantize the weights, and the effect of the binary weights during inference is simulated. During the backward propagation, a `clip` function is introduced to cut off the update range of the full-precision weights to prevent the real-valued weights from growing too large without any impact on the binary weights. Though after model binarization the parameters of the neural network model is greatly compressed (even with large quantization error), the binary model can closely reach the state-of-the-art performance on some datasets in the image classification tasks. The stochastic binarization method in BinaryConnect is defined as:

$$w_b = \begin{cases} +1, & \text{with probability } p = \hat{\sigma}(w) \\ -1, & \text{with probability } 1 - p \end{cases} \tag{6}$$

where $\hat{\sigma}$ is the "hard sigmoid" function:

$$\hat{\sigma}(x) = \texttt{clip}(\frac{x+1}{2}, 0, 1) = \max(0, \min(1, \frac{x+1}{2})) \tag{7}$$

Following the paradigm of binarizing the network, Courbariaux *et.al.* further introduced Binarized Neural Network (BNN) [57], presenting the training and acceleration skills in detail. It proved the practicability and acceleration capability of binary neural networks from both theoretical and practical aspects. For the inference acceleration of networks with batch normalization, this method also devised techniques like Shift-based Batch Normalization and XNOR-Bitcount. The experiments on image classification show that BNN takes 32× less storage space and 60% less time. Smaragdis *et.al.* also studied the network binarization and developed Bitwise Neural Network especially suitable for resource-constrained environments [69].

3.2. Optimization Based Binary Neural Networks

The naive binarization methods own the advantages of saving computational resources by quantizing the network in a very simply way. However, without considering the effect of the binarization in the forward and backward process, these methods inevitably suffer the accuracy loss for the wide tasks. Therefore, in order to mitigate the accuracy loss in the binary neural network, in the past years, a great number of optimization-based solutions have been proposed and shown the successful improvement over the native ones.

3.2.1. Minimize the Quantization Error

For the optimization of binary neural networks, a common practice is to reduce the quantization error of weight and activation. This is a straightforward solution similar to the standard quantization mechanism that the quantized parameter should approximate the full-precision parameter as closely as possible, expecting that the performance of the binary neural network model will be close to the full-precision one.

As the early research considering the quantization error, Rastegari *et.al.* proposed Binary Weight Networks (BWN) and XNOR-Net [58]. BWN adopts the setting of binary weights and full-precision activations, while XNOR-Net binarizes both weights and activations. Different from the prior studies, [58] well approximates the floating-point parameters by introducing a scaling factor for the binary parameter. Specifically, the weight quantization process in BWN and XNOR-Net can be formulated as $\mathbf{w} \approx \alpha \mathbf{b}_{\mathbf{w}}$, where α is the floating-point scaling factor for the binarized weight $\mathbf{b}_{\mathbf{w}}$. This means that the weights in BWN are binarized to $\{-\alpha, +\alpha\}$, but still can bring the benefits of fast computation. Then minimizing the quantization error can help to find the optimal scaling factor and binary parameters:

$$\min_{\alpha, \mathbf{b}_{\mathbf{w}}} \|\mathbf{w} - \alpha \mathbf{b}_{\mathbf{w}}\|^2 \tag{8}$$

The solution enjoys much less quantization error than directly using 1-bit (-1/+1), thereby improving the inference accuracy of the network. Figure 2 shows the binarization and the corresponding convolution process in XNOR-Net. Similar idea was also proposed in Binary Weight Networks via Hashing (BWNH) [73], which considers the quantizing process as a hash map with scaling factors. The DoReFa-Net [60] further extends XNOR-Net, so that the network training can be accelerated using quantized gradients. Mishra *et.al.* devised Wide Reduced-Precision Networks (WRPN) [75] that also minimize the quantization error in a similar way to XNOR-Net, but increase the number of filters in each layer. Compared with directly binarizing the network, widening and binarizing together can achieve a good balance between the precision and the acceleration. The work of Faraone *et.al.* groups parameters in training process and gradually quantizes each group with optimized scaling factor to minimize the quantization error [77].

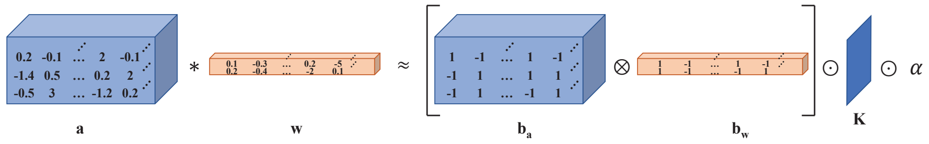

Figure 2: Binarization and Convolution Process of XNOR-Net

To further reduce the quantization error, High-Order Residual Quantization (HORQ) [70] adopts a recursive approximation to the full-precision activation based on the quantized residual, instead of one-step approximation used in XNOR-Net. It generates the final quantized activation by a linear combination of the approximation in each recursive step. In a very similar way, Lin *et.al.* designed ABC-Net [71] that linearly combines multiple binary weight matrices and scaling factors to fit the full-precision weights and activations, which can largely reduce the information loss caused by binarization. Wang *et.al.* pointed out the shortcoming of the previous methods that separately minimizing the quantization error of weights and activations can hardly promise the outputs to be similar to the full-precision ones [72]. To address this problem, a two-step quantization (TSQ) method is designed. During the first step, all weights are full-precision values and all activations are quantized into low-bit format with a learnable quantization function Q_a. During the second step, Q_a is fixed and the low-bit weight vector $\mathbf{b_w}$ and scaling factor α are learned as follow:

$$\min_{\alpha,\mathbf{b_w}} \quad \|\mathbf{z} - Q_a\left(\alpha(\mathbf{a} \odot \mathbf{b_w})\right)\|_2^2 \,, \tag{9}$$

which can be solved efficiently in an iterative manner.

The aforementioned methods usually choose the fixed binarization function (*e.g.* , `sign` function). One can also adopt more flexible binarization function and learn its parameters during minimizing the quantization error. To achieve this goal, Choi *et.al.* proposed PArameterized Clipping Activation (PACT) [74] with a learnable upper bound for the activation function. The optimized upper bound of each layer is able to ensure that the quantization range of each layer is aligned with the original distribution. In practice, PACT performs better on binary networks, and can achieve accuracy close to full-precision network on larger networks. In [61], the Learned Quantization (LQ-Nets) attempts to minimize quantization error by jointly training neural networks and quantizers in the network. Different from the previous work, LQ-Nets learn the quantization thresholds and cutoff values by minimizing the quantization error during the network training, and can support arbitrary bit quantization. In [97], trainable scaling factors for both weights and activations are introduced to increase the value range. And based on XNOR-Net, Bulat *et.al.* fused the activation and weight scaling factors into a single one that is learned discriminatively via backward propagation and proposed XNOR-Net++ [76].

3.2.2. Improve the Network Loss Function

Minimizing the quantization error tries to retain the values of full-precision weights and activations, and thus reduces the information loss in each layer. However, only focusing on the local layers can hardly promise the exact final output passed through a series of layers. Therefore, it is highly required that the network training can globally take the binarization as well as the task-specific objective into account. Recently, an amount of research works at finding the desired network loss function that can guide the learning of the network parameters with restrictions brought by binarization.

Usually the general binarization scheme only focuses on accurate local approximation of the floating-point values and ignores the effect of binary parameters on the global loss. In [83], Hou *et.al.* proposed Loss-Aware Binarization (LAB) that directly minimizes the overall loss associated with binary weights using the quasi-Newton algorithm. The method utilizes information from the second-order moving average that has been calculated by the Adam optimizer to find optimal weights with consideration of the characteristics of binarization. Apart from considering the task-relevant loss from a quantization view, devising additional quantization-aware loss item is proved to be practical. In [85], Ding *et.al.* summarized the problems caused by forward binarization and backward propagation in binary neural networks, including "degeneration", "saturation" and "gradient mismatch". To address these issues, a distribution loss was introduced to explicitly regularize the activation distribution as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{DL} \tag{10}$$

where $\mathcal{L}_{CE}$ is the common cross-entropy loss for training deep neural networks, $\mathcal{L}_{DL}$ is the distribution loss for learning the proper binarization, and λ balances the effect of the two types of losses. With the guide of additional loss, the learned neural network can effectively avoid the aforementioned obstacles and is friendly to binarization. The Incremental Network Quantization (INQ) method [84] proposed by Zhou *et.al.* also proved this point, which adds a regularization term in loss function.

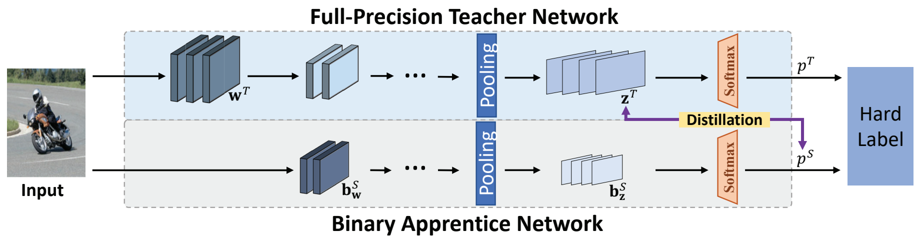
Full-Precision Teacher Network
Input
$\mathbf{w}^T$
Pooling
$\mathbf{z}^T$
Softmax
p^T
Distillation
Hard Label
$\mathbf{b}_\mathbf{w}^S$
Pooling
$\mathbf{b}_\mathbf{z}^S$
Softmax
p^S
Binary Apprentice Network

Figure 3: Schematic of Apprentice Network

The guiding information for training accurate binary neural networks can also derive from the knowledge of a large full-precision model. The Apprentice method [82] trains a low-precision student network using a well-trained, full-precision, large-scale teacher network, using the following loss function:

$$\mathcal{L}\left(x; \mathbf{w}^T, \mathbf{b}_\mathbf{w}^S\right) = \alpha \mathcal{H}\left(y, p^T\right) + \beta \mathcal{H}\left(y, p^S\right) + \gamma \mathcal{H}\left(z^T, p^S\right) \tag{11}$$

where $\mathbf{w}^T$ and $\mathbf{b}_{\mathbf{w}}^S$ are the full-precision weights of the teacher model and binary weights of the student (apprentice) model respectively, y is the label for sample x, $\mathcal{H}(\cdot)$ is the soft and hard label loss function between the teacher and apprentice model, and $\alpha,\ \beta,\ \gamma$ are the weighting factors, p^T and p^S are the predictions of the teacher and student model, respectively. Under the supervision of the teacher network, the binary network can preserve the learning capability and thus obtain the close performance to the teacher network. The process of knowledge distillation is shown in Figure 3. Similar mimic solutions like Distillation and Quantization (DQ) [81], Distilled Binary Neural Network (DBNN) [80] and Main/Subsidiary Network [87] have been studied, and their experiments demonstrate that the loss functions related to the full-precision teacher model help to stabilize the training of binary student model with high accuracy. CI-BCNN proposed in [86] mines the channel-wise interactions, through which prior knowledge is provided to alleviate inconsistency of signs in binary feature maps and preserves the information of input samples during inference. [79] built strong BNNs with a loss function during training, which matches the spatial attention maps computed at the output of the binary and real-valued convolutions.

3.2.3. Reduce the Gradient Error

Training of binary neural networks still relies on the popular BP algorithm. To deal with the gradients for the non-differential binarization function, straight-through estimator (STE) technique is often adopted to estimate the gradients in backward propagation [68]. However, there exists obvious gradient mismatch between the gradient of the binarization function (*e.g.* , `sign`) and STE (*e.g.* , `clip`). Besides, it also suffers the problem that the parameters outside the range of $[-1, +1]$ will not be updated. These problems easily lead to the under-optimized binary networks with severe performance degradation.

Intuitively, an elaborately designed approximate binarization function can help to relieve the gradient mismatch in the backward propagation. Bi-Real [62] presents a customized `ApproxSign` function to replace `sign` for back-propagation gradient calculation as follow:

$$\texttt{ApproxSign}(x) = \begin{cases} -1, & \text{if } \ x < -1 \\ 2x + x^2, & \text{if } \ -1 \leq x < 0 \\ 2x - x^2, & \text{if } \ 0 \leq x < 1 \\ 1, & \text{otherwise} \end{cases} \tag{12}$$

$$\frac{\partial \texttt{ApproxSign}(x)}{\partial x} = \begin{cases} 2 + 2x, & \text{if } \ -1 \leq x < 0 \\ 2 - 2x, & \text{if } \ 0 \leq x < 1 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Compared to the traditional STE, `ApproxSign` has a close shape to that of the original binarization function `sign`, and thus the gradient error can be controlled to some extent. Circulant Binary Convolutional Networks (CBCN) [88] also applied an approximate function to address the gradient mismatch from `sign` function. Binary Neural Networks+ (BNN+) [90] directly proposed an improved approximation to the derivative of the `sign` function, and introduced a regularization function that encourages the learned weights around the binary values.

Besides focusing on the backward propagation, some recent methods attempted to pursue the good quantization functions in forward propagation, which can also reduce the gradient error. In [89], the proposed Half-ware Gaussian Quantization (HWGQ) method gave a low-precision estimation for the more commonly used `ReLU` function in the forward propagation in training process, which surprisingly works well to solve the gradient mismatch problem. Following the same intuition, Gong *et.al.* present a Differential Soft Quantization (DSQ) method [63], replacing the traditional quantization function with a soft quantization function:

$$\varphi(x) = s \tanh\left(k\left(x - m_i\right)\right), \quad \text{if } \ x \in \mathcal{P}_i \tag{14}$$

where k determines the shape of the asymptotic function, s is a scaling factor to make the soft quantization function smooth and m_i is the center of the interval

$\mathcal{P}_i$. DSQ can adjust the cutoff value and the shape of the soft quantization function to gradually approach the standard `sign` function. In fact, the DSQ function rectifies the data distribution in a steerable way, and thus helps to alleviate the gradient mismatch. The overview of DSQ is shown in Figure 4. A similar method [93] also provides a simple and uniform way for weight and activation quantization by formulating it as a differentiable non-linear function. Besides, ProxQuant proposed in [92] formulates quantized network training as a regularized learning problem instead and optimizes it via the prox-gradient method. ProxQuant does backward propagation on the underlying full-precision vector and applies an efficient prox-operator in between stochastic gradient steps. [94] and [95] also explored the smooth transitions for the derivative of the `Sign`, and used `Tanh` function with parameters v and `SoftSign` function to reduce training gradient error. The IR-Net proposed in [96] included a self-adaptive Error Decay Estimator (EDE) to reduce the gradient error in training, which considers different requirements on different stages of training process and balances the update ability of parameters and reduction of gradient error. The IR-Net provided a new perspective for improving BNNs that retaining both forward and backward information is crucial for accurate BNNs, and it is the first to design BNNs considering both forward and backward information retention.

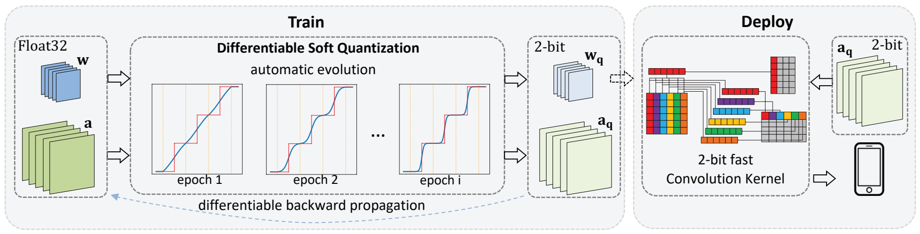

Figure 4: Overview of Differentiable Soft Quantization

Besides modifying binarization function in backward or forward propagation, [91] directly calibrates the gradients by the blended coarse gradient descent (BCGD) algorithm. The weight update of BCGD goes by a weighted average of the full-precision weights and their quantized counterparts:

$$\mathbf{w}^{t+1} = (1-\rho)\mathbf{w}^t + \rho \mathbf{b}_{\mathbf{w}}^t - \eta \nabla f\left(\mathbf{b}_{\mathbf{w}}^t\right) \tag{15}$$

where $\mathbf{w}^t$ denotes the full-precision weights on the t-th step and $\nabla f\left(\mathbf{b}_{\mathbf{w}}^t\right)$ denotes the gradient of $\mathbf{b}_{\mathbf{w}}^t$, thereby yielding sufficient descent in the objective value and thus accelerates the training. [98] further investigated training methods for quantized neural networks from a theoretical viewpoint, and show that training algorithms that exploit high-precision representations have an important greedy search phase that purely quantized training methods lack, which explains the difficulty of training using low-precision arithmetic.

3.3. Efficient Computing Architectures for Binary Neural Networks

The most attractive point of binary neural networks is that they enjoy the advantages of fast computation, low power consumption and low memory footprint, which can faithfully support the general hardware (including FPGA, ASIC, CPU, *etc*) with limited computational resources. FPGAs are the most widely used platforms because they allow for customizing data paths and adjusting the designs. In particular, FPGAs allow optimization around XNOR-Bitcount operations. ASICs have the potential to provide ultimate power and computational efficiency for binary neural networks, because the hardware layout in ASICs can be designed according to network structure. To make the binarization algorithms more practical in the wide scenarios with different hardware environment, researchers also devoted great efforts to developing hardware-friendly binary networks.

XNOR.AI team, who proposed XNOR-Net [58], successfully launched XNOR-Net on the cheap Raspberry Pi device. In order to reduce the amount of computation, they conducted optimization for different targeted hardware. They also tried to combine XNOR-Net with real-time detection algorithms such as YOLO [99], and deployed them in the edge computing scenarios like smart home and autonomous driving. FP-BNN [100] implemented a 64-channel acceleration on the Stratix-V FPGA system and analyzed the performance through the Resource-Aware Model Analysis (RAMA) method. Both [101] and [102] from Xilinx also studied the FPGA-based binary network accelerator using different strategies. [101] depended on variable-length buffers and achieved up to twice the number of operations per second of existing FPGA accelerators. [103] proposed two types of fast and energy-efficient architectures for binary neural network inference. By reusing the results from previous computation, much cycles for data buffer access and computations can be skipped. In order to achieve the most possible memory latency hiding, [102] designed a multi-stream architecture, and applied the Bitcount, Threshhold and OR operations to map the binary network to the FPGA operators. The researchers of the Haas-Platna Software Institute in Germany implemented an accelerated version of BMXNet [104, 105] on GPU for both binary neural networks and linear quantization networks based on MXNet, supporting XNOR-Net and DoReFa-Net. For ARM platform, engineers from JD company developed the binarization inference library daBNN [106] for mobile phone platforms. The library uses ARM assembly instructions, which is 8-24× more efficient than BMXNet.

Table 2: Deployment Performance of Binary Neural Networks

Dataset	**Method**	**Acc. (%)**	**Topology**	**Platform**	**LUTs**	**BRAMs**	**Clk (MHz)**	**FPS**	**Power (W)**
MNIST [107]	FINN-R [108]	97.7	MLP-4	Zynq-8020	25,358	220	100	-	2.5
	FINN-R [108]	97.7	MLP-4	ZynqUltra 3EG	38,250	417	300	-	11.8
	ReBNet[109]	98.3	MLP-4*	Spartan 750	32,600	120	200	-	-
	FINN [102]	98.4	MLP-4	Zynq-7045	82,988	396	200	1,561,000	22.6
	BinaryEye [110]	98.4	MLP-4	Kintex 7325T	40,000	110	100	10,000	12.2
SVHN [111]	FINN [102]	94.9	CNV-6	Zynq-7045	46,253	186	-	21,900	11.7
	FBNA [112]	96.9	CNV-6	Zynq-7020	29,600	103	-	6,451	3.2
	ReBNet [109]	97.0	CNV-6*	Zynq-7020	53,200	280	100	-	-
CIFAR-10 [113]	Zhou *et.al.* [114]	66.6	CNV-2*	Zynq-7045	20,264	-	-	-	-
	Nakahara *et.al.* [115]	-	CNV*	Vertex-7 690T	20,352	372	450	-	15.4
	Fraser *et.al.* [116]	79.1	1/4 cnn*	KintexUltra 115	35,818	144	125	12,000	-
	FINN-R [108]	80.1	CNV-6	ZynqUltra 3EG	41,733	283	300	-	10.7
	FINN-R [108]	80.1	CNV-6	Zynq-7020	25,700	242	100	-	2.3
	FINN [102]	80.1	CNV-6	Zynq-7045	46,253	186	200	21,900	11.7
	FINN [108]	80.1	CNV-6	ADM-PCIE-8K5	365,963	1,659	125	-	41.0
	FINN [117]	80.1	CNV-6	Zynq-7020	42,823	270	166	445	2.5
	Nakahara *et.al.* [117]	81.8	CNV-6	Zynq-7020	14,509	32	143	420	2.3
	Fraser *et.al.* [116]	85.2	1/2 cnn*	KintexUltra 115	93,755	386	125	12,000	-
	Zhou *et.al.* [114]	86.1	CNV-5*	Vertex-7 980T	556,920	-	340	332,158	-
	ReBNet [109]	87.0	CNV-6*	Zynq-7020	53,200	280	200	-	-
	Zhao *et.al.* [101]	87.7	CNV-6	Zynq-7020	46,900	140	143	168	4.7
	Fraser *et.al.* [116]	88.3	BNN*	KintexUltra 115	392,947	1814	125	12,000	-
	FBNA [112]	88.6	CNV-6	Zynq-7020	29,600	103	-	520	3.3
ImageNet [118]	ReBNet [109]	41.0	CNV-5*	VertexUltra 095	1,075,200	3456	200	-	-
	Yonekawa *et.al.* [119]	-	VGG-16	ZynqUltra 9EG	191,784	32,870	150	31.48	22.0

[1] The * represents the network using a customized network structure, and the Acc. in table refers to Top-1 classification accuracy on each dataset.

We provide a comparison of different binary neural network implementations [108, 102, 109, 110, 115, 116, 117, 101, 119, 114] on different FPGA platforms in Table 2. It can be seen that the method proposed by [119] can achieve comparable accuracy to full-precision models, although it is not efficient enough. The implementation of Xilinx's [102] owns the most promising speed with a low power consumption. A series of experiments prove that it can achieve a good balance among accuracy, speed and power consumption. [109] obtains high accuracy on small datasets such as MNIST and CIFAR-10, but a poor result on ImageNet. We have to point out that despite the progress of developing hardware-friendly algorithms, till now there have been quite few binary models that can perform well on large datasets such as ImageNet in terms of both speed and accuracy.

3.4. Applications of Binary Neural Networks

Image classification is a fundamental task in computer vision and machine learning. Therefore, most of existing studies chose to evaluate binary neural networks on the image classification tasks. BNNs can greatly accelerate and compress the neural network models, which is of great attraction to deep learning researchers. Both weights and activations are binary in BNNs and it results in $58\times$ faster convolutional operations and $32\times$ memory savings theoretically. Thus the binary neural networks are also applied to other common tasks such as object detection and semantic segmentation.

In the literature, Kung *et.al.* utilized binary neural networks for both object recognition and image classification tasks [120] on infrared images. In this work, the binary neural networks got comparable performance to full-precision networks on MNIST and IR datasets and achieved at least a $4\times$ acceleration and an energy saving of three orders of magnitude over the GPU. [64] also addressed the fast object detection algorithm by unifying the prediction and object detection process. It obtained $62\times$ acceleration and saved $32\times$ storage space using binary VGG-16 network, where except the last convolution layer, all other layers are binarized. Li *et.al.* generated quantized object detection neural networks based on RetinaNet and faster R-CNN, and show that these detectors achieve very encouraging performance [65]. Leng *et.al.* applied BNNs to different tasks, and evaluated their method on convolutional neural networks for image classification and object detection, and recurrent neural networks for language model [121]. Zhuang *et.al.* proposed a "network decomposition" strategy called Group-Net, which shows strong generalization to different tasks including classification and semantic segmentation, outperforming the previous best binary neural networks in terms of accuracy and major computation savings [122]. In [123], SeerNet considers feature-map sparsity through low-bit quantization, which is applicable to general convolutional neural networks and tasks.

The researchers also studied enhancing the robustness of neural network models through model binarization. Binary models were generally considered to be more robust than full-precision models, because they were considered to filter the noise of input. Lin *et.al.* explored the impact of quantization on the model robustness. They showed that quantization operations on parameters help BNNs to reduce the distance by removing perturbations when magnitude of noise is small. However, for vanilla BNNs, the distance is enlarged when magnitude of noise is large. The inferior robustness comes from the error amplification effect in forward propagation of BNNs, where the quantization operation further enlarges the distance caused by amplified noise. They propose Defensive Quantization (DQ) [66] to defend the adversarial examples for quantized models by suppressing the noise amplification effect and keeping the magnitude of the noise small in each layer. Quantization improves robustness instead of making it worse in DQ models, thus they are even more robust than full-precision networks.

3.5. Tricks for Training Binary Neural Networks

Due to the highly discrete nature of the binarization, training binary neural networks often requires the introduction of special training techniques to make the training process more stable and the convergence accuracy much higher. In this section, we summarize the general and effective binary neural network training techniques that have been widely adopted in the literature, from the aspects including network structure transformation, optimizer and hyper-parameter selection, gradient approximation and asymptotic quantization.

3.5.1. Network Structure Transformation

Binarization converts activations and weights to $\{-1, +1\}$. This is actually equivalent to regularizing the data, making the data distribution changed in an unexpected way after binarization. Adjusting the network structure serves as a promising solution to adapting to the distribution changes.

Simply reordering the layers in the network can improve the performance of the binary neural network. In [124], researchers from the University of Oxford pointed out that almost all binarization studies have repositioned the location of the pooling layer. The pooling layer is always used immediately after the convolutional layer to avoid information loss caused by max pooling after binarization. Experiments have shown that this position reorder has a great improvement in accuracy. In addition to the pooling layer, the location of the batch normalization layer also greatly affects the stability of binary neural network training. [72] and [89] insert a batch normalization layer before all quantization operations to rectify the data. After this transformation, the quantized input obeys a stable distribution (sometimes close to Gaussian), and thus the mean and variance keep within a reasonable range and the training process becomes much smoother.

Based on the similar idea, instead of adding new layers, several recent work attempts to directly modify the network structure. For example, Bi-Real [62] connects the full-precision feature maps across the layer to the subsequent network. This method essentially adjusts the data distribution through structural transformation. Mishra *et.al.* devised Wide Reduced-Precision Networks (WRPN) [75], which increase the number of filters in each layer and thus reform the data distributions. Binary Ensemble Neural Network (BENN) [67] leverages the ensemble method to fit the underlying data distributions. Liu *et.al.* proposed circulant filters (CiFs) and a circulant binary convolution (CBConv) to enhance the capacity of binarized convolutional features, and circulant back propagation (CBP) was also proposed to train the structures [88]. BBG [78] even appended a gated residual to compensate their information loss during the forward process.

3.5.2. Optimizer and Hyper-parameter Selection

Choosing the proper hyper-parameters and specific optimizers when training binary neural networks also improves the performance of BNNs. Most existing binary neural network models chose an adaptive learning rate optimizer, such as Adam. Using Adam can make the training process better and faster, and the smoothing coefficient of the second derivative is especially critical. The analysis by [124] shows that if using a fixed learning rate optimizer that does not consider historical information, such as a stochastic gradient descent (SGD) algorithm, one needs to adopt a large batch size to improve the performance.

The setting of the batch normalization's momentum coefficient is also critical. In [124], by comparing the precision results under different momentum coefficients, it is found that the parameters of the batch normalization need to be set appropriately to adapt to the jitter caused by the binarization operation.

3.5.3. Asymptotic Quantization

Since the quantization has negative impact on training, many methods employed the asymptotic quantization strategy, which gradually increases the degree of quantization, to reduce the losses caused by parameter binarization. Practice shows that this step-by-step quantization method is useful to find the optimal solution. For instance, INQ [84] groups the parameters and gradually increases the number of groups participating in the quantization to achieve group-based step-by-step quantization. [125] introduces the idea of stepping the bit-width, which first quantizes to a higher bit-width and then quantizes to a lower bit-width. This strategy can help to avoid the large perturbations caused by extremely low-bit quantization, compensating the gradient error of quantized parameters during training.

3.5.4. Gradient Approximation

It became a common practice to use a smoother estimator in binary neural network training process. The gradient error usually exists in backward propagation due to the straight-through estimator. Finding an approximate function close to the binarization function serves as the simple and practical solution. This becomes a popular technique widely considered in recent studies [62, 89, 88, 90, 94, 95, 96], where the approximate functions are tailored according to different motivations, to replace the standard `clip` function that causes gradient error. For designing a proper approximate function, an inspiring idea is to align its shape with that of the binarization function [63].

4. Evaluation and Discussions

4.1. Datasets and Network Structures

To evaluate the binary neural network algorithms, the image classification task is widely chosen, and correspondingly two common image datasets: CIFAR-10 [113] and ImageNet [118] are usually used. The CIFAR-10 is a relatively small dataset containing 60,000 images with 10 categories, while ImageNet dataset is currently the most popular image classification dataset. For other tasks like object detection and semantic segmentation, PASCAL VOC [126] and COCO (Common Objects in Context) [127] are also employed for evaluating the performance of the binary neural networks. PASCAL VOC dataset is derived from the PASCAL Visual Object Classes challenge, which is used to evaluate the performance of models for various tasks in the field of computer vision. Many excellent computer vision models (including classification, positioning, detection, segmentation, motion recognition, *etc*) are based on the PASCAL VOC dataset, especially some object detection models. COCO is a dataset provided by the Microsoft team for image recognition and object detection. It collects images by searching 80 object categories and various scene types such as Flickr.

To investigate the generalization capability of the binary neural network algorithm over different network structures, various deep models including VGG [12], AlexNet [11], ResNet-18 [28], ResNet-20, ResNet-34, and ResNet-50, *etc.* will be binarized and tested. These models have outstanding contributions in the progress of deep learning, and make significant breakthrough in ImageNet classification task. Among them, the VGG network contains a large number of parameters and convolution operations, so binarizing VGG can obviously show the inference acceleration of different algorithms. ResNet is currently the most popular deep model in many tasks, with a sufficient number of layers.

4.2. Image Classification Tasks

Most binary neural networks adopt the inference accuracy of image classification as the evaluation metric, as the classical classification models do. Table 3 and 4 respectively illustrate the performance of the typical binary neural network methods on CIFAR-10 and ImageNet, and compare the inference accuracy with different bit-width and network structures.

Comparing the performance of binary neural networks on different datasets, we can first observe that binary neural networks can approach the performance of full-precision neural networks on small datasets (*e.g.* MNIST, CIFAR-10), but still suffer a severe performance drop on large datasets (*e.g.* ImageNet). This is mainly because for the large dataset, the binarized network lacks sufficient capacity to capture the large variations among data. This fact indicates that there still require great efforts for pursuing the delicate binarization and optimization solution to design a satisfactory binary neural network.

From the table 3 and 4, it can be concluded that the neural networks are more sensitive to the binarization of activations. When only quantizing weights to 1-bit and leaving the activations as full-precision, there is a smaller performance degradation. Taking ResNet-18 in ABC-Net [71] on ImageNet dataset as an example, there is only about 7% accuracy loss after applying binarization to weights but there is addition 20% loss after the activations are binarized. Thus eliminating the influence of activation binarization is usually much more important when designing binary network, which becomes the main motivations for studies like [85] and [74]. After adding reasonable regularization to the dis-

Table 3: Image Classification Performance of Binary Neural Networks on CIFAR-10 Dataset

Type		**Method**	**Bit-Width (W/A)**	**Topology**	**Acc. (%)**
Full-Precision Neural Networks		-	32/32	VGG-Small [61]	93.8
			32/32	ResNet-20 [61]	92.1
			32/32	ResNet-32 [92]	92.8
			32/32	ResNet-44 [92]	93.0
			32/32	VGG-11 [87]	83.8
			32/32	NIN [87]	84.2
Naive Binary Neural Networks		BinaryConnect [59]	1/32	VGG-Small	91.7
		BNN [57]	1/1	VGG-Small	89.9
Optimization Based Binary Neural Networks	Minimize the Quantization Error	BWN [58]	1/32	VGG-Small	90.1
		XNOR-Net [58]	1/1	VGG-Small	89.8
			1/1	Customized [70]	77.0
		DoReFa-Net [60]	1/32	ResNet-20	90.0
			1/1	ResNet-20	79.3
		HORQ [70]	2/1	Customized [70]	82.0
		TSQ [72]	3/2	VGG-Small	93.5
		BBG [78]	1/1	ResNet-20	85.3
			1/1	ResNet-20 (2x)	90.7
			1/1	ResNet-20 (4x)	92.5
		LQ-Nets [61]	1/32	ResNet-20	90.1
			1/2	VGG-Small	93.4
	Improve Network Loss Function	LAB [83]	1/32	VGG-Small	89.5
			1/1	VGG-Small	87.7
		Main/Subsidiary Network [87]	1/1	NIN	83.1
			1/1	VGG-11	82.0
			1/1	ResNet-18	86.4
		BCGD [87]	1/4	VGG-11	89.6
			1/4	ResNet-20	90.1
		ProxQuant [92]	1/32	ResNet-20	90.7
			1/32	ResNet-32	91.5
			1/32	ResNet-44	92.2
		BNN-DL [85]	1/1	VGG-Small	90.0
		CI-BCNN [86]	1/1	VGG-Small	92.5
			1/1	ResNet-20	91.1
	Reduce the Gradient Error	DSQ [63]	1/1	VGG-Small	91.7
			1/32	ResNet-20	90.2
			1/1	ResNet-20	84.1
		IR-Net [63]	1/32	ResNet-20	90.2
			1/1	VGG-Small	90.4
			1/1	ResNet-18	91.5
			1/1	ResNet-20	86.5

Table 4: Image Classification Performance of Binary Neural Networks on ImageNet Dataset

Type		Method	Bit-Width (W/A)	Topology	Top-1 (%)	Top-5 (%)
Full-Precision Neural Networks		-	32/32	AlexNet [61]	57.1	80.2
			32/32	ResNet-18 [61]	69.6	89.2
			32/32	ResNet-34 [61]	73.3	91.3
			32/32	ResNet-50 [61]	76.0	93.0
			32/32	VGG-Variant [61]	72.0	90.5
Naive Binary Neural Networks		BinaryConnect [59]	1/32	AlexNet	35.4	61.0
		BNN [57]	1/1	AlexNet	27.9	50.4
Optimization Based Binary Neural Networks	Minimize the Quantization Error	BWN [58]	1/32	AlexNet	56.8	79.4
			1/32	ResNet-18	60.8	83.0
		XNOR-Net [58]	1/1	AlexNet	44.2	69.2
		DoReFa-Net [60]	1/1	AlexNet	43.6	-
			1/1	AlexNet	49.8	-
		ABC-Net [71]	1/32	ResNet-18	62.8	84.4
			2/32	ResNet-18	63.7	85.2
			1/1	ResNet-18	42.7	67.6
			1/1	ResNet-34	52.4	76.5
		TSQ [72]	1/1	AlexNet	58.0	80.5
		BWNH [73]	1/32	AlexNet	58.5	80.9
			1/32	ResNet-18	64.3	85.9
		PACT [74]	1/32	ResNet-18	65.8	86.7
			1/2	ResNet-18	62.9	84.7
			1/2	ResNet-50	67.8	87.9
		LQ-Nets [61]	1/2	ResNet-18	62.6	84.3
			1/2	ResNet-34	66.6	86.9
			1/2	ResNet-50	68.7	88.4
			1/2	AlexNet	55.7	78.8
			1/2	VGG-Variant	67.1	87.6
		SYQ [77]	1/2	AlexNet	55.4	78.6
			1/8	ResNet-18	62.9	84.6
			1/8	ResNet-50	70.6	89.6
		WRPN [75]	1/1 (1×)	ResNet-34	60.5	-
			1/1 (2×)	ResNet-34	69.9	-
			1/1 (3×)	ResNet-34	72.4	-
		XNOR-Net++ [76]	1/1 (1×)	ResNet-18	57.1	79.9
			1/1 (1×)	AlexNet	46.9	71.0
	Improve Network Loss Function	INQ [84]	2/32	ResNet-18	66.0	87.1
		BNN-DL [85]	1/1	AlexNet	41.3	65.8
		XNOR-Net-DL [85]	1/1	AlexNet	47.8	71.5
		DoReFa-Net-DL [85]	1/1	AlexNet	47.8	71.5
		CompactNet-DL [85]	1/2	AlexNet	47.6	71.9
		WRPN-DL [85]	1/1	AlexNet	53.8	77.0
		Main/Subsidiary Network [87]	1/1	ResNet-18	50.1	-

Table 4: (*Cont.*) Image Classification Performance of Binary Neural Networks on ImageNet Dataset

Type		**Method**	**Bit-Width (W/A)**	**Topology**	**Top-1 (%)**	**Top-5 (%)**
Optimization Based Binary Neural Networks		CI-BCNN [86]	1/1	ResNet-18	59.9	84.2
			1/1	ResNet-34	54.9	86.6
	Reduce the Gradient Error	Bi-Real [62]	1/1	ResNet-18	56.4	79.5
			1/1	ResNet-34	62.2	83.9
		HWGQ [89]	1/1	AlexNet	52.7	76.3
		CBCN [88]	1/1	ResNet-18	61.4	82.8
		Quantization Networks [93]	1/32	AlexNet	58.8	81.7
			1/32	ResNet-18	66.5	87.3
			1/32	ResNet-50	72.8	91.3
		BCGD [91]	1/4	ResNet-18	65.5	86.4
			1/4	ResNet-34	68.4	88.3
		DSQ [63]	1/32	ResNet-18	63.7	-
		IR-Net [96]	1/32	ResNet-18	62.9	84.1
			1/32	ResNet-34	70.4	89.5
			1/1	ResNet-18	58.1	80.0
			1/1	ResNet-34	66.5	86.8
		IT-BNN [95]	1/1	ResNet-18	53.7	76.8
			1/1	AlexNet	48.6	72.8

tribution of activations, the harmful effect caused by binarization on activations will be reduced, and subsequently the accuracy is naturally improved.

What's more, the robustness of binary neural networks is highly relevant to their structures. Some specific structure patterns are friendly to binarization, such as skip connections proposed in [62] and wider blocks proposed in [75]. With a shortcut to directly pass full-precision values to the following layers, Bi-Real [62] achieves performance close to full-precision models. With a $3\times$ wider structure, the accuracy loss of ResNet-34 in [75] is lower than 1%. In fact, what they essentially do is to enable the information to pass through the whole network as much as possible. Although the structure modification may increase the amount of calculation, they can still get a significant acceleration benefiting from the XNOR-Bitcount operation.

Different optimization-based methods represent different understandings of BNNs. Among the papers aiming to minimizing the quantization error, many methods that directly reduce the quantization error were proposed to make the binary neural networks approximate full-precision neural networks. These papers believed that the closer the binary parameters are to full-precision parameters, the better the BNNs perform. Another idea is improving the loss function. This type of methods makes the parameter distribution in BNNs friendly to the binarization operation by modifying loss function. Moreover, STE proposed in BinaryConnect is rough, which results in some problems such as gradient mismatch. Thus many recent works use smooth transition such as `Tanh` function to reduce the gradient loss, and it became a common practice to use a smoother estimator.

We believe binary neural networks should not be simply regarded as the approximations of full-precision neural networks, more specific designs for the special characteristics of BNNs are necessary. In fact, some of the recent works essentially worked on this such as XNOR-Net++ [76], CBCN [88], Self-Binarizing Networks [94], BENN [67], *etc.* The results show that specially designed methods considering characteristics of BNNs can achieve better performance. They prove the view that BNNs need different optimization compared with the full-precision models although they share the same network architecture.

It is also worth mentioning that accuracy is not the only criterion of BNNs, the versatility is another key to measure whether a method can be used in practice. Some methods proposed in existing papers are very versatile, such as scale factors proposed in XNOR-Net [58], smooth transition [90], addition shortcuts [62], *etc.* The methods are versatile because of their simple implementation and low coupling. Thus they become common practices to improve the performance of BNNs. Some methods improve the performance of binary neural networks by designing or learning delicate quantizers. Such quantizers usually have stronger ability to preserve the information. However, we have to point out that some of them suffer complicate computation and even multi-stage training pipelines, which is sometimes unfriendly to hardware implementation and reproducibility. This means it is hard to acquire an effective speed up with such quantizers in real-world deployment. Therefore, purely pursuing high accuracy without considering the acceleration implementation makes no sense in practice. The balance between accuracy and speed is also an essential criterion for binarization research that should be always kept in mind.

Table 5: Object Detection Performance of Binary Neural Networks on COCO 2017 Dataset

Topology	Method	Bit-Width (W/A)	mAP (%)
Faster-RCNN [128] ResNet-18	FQN [65]	1/1	28.1
Faster-RCNN [128] ResNet-34	FQN [65]	1/1	31.8
Faster-RCNN [128] ResNet-50	FQN [65]	1/1	33.1
RetinaNet [129] ResNet-18	Quant whitepaper [130]	8/8	22.6
	Integer-only [131]	8/8	19.7
	DoReFa-Net [60]	1/1	3.9
	XNOR-Net [58]	4/4	24.4
	XNOR-Net (Percentile) [65]	4/4	26.7
	FQN [65]	4/4	28.6

Table 6: Object Detection Performance of Binary Neural Networks on PASCAL VOC 2007 Dataset

Topology	Method	Bit-Width (W/A)	mAP (%)
Faster-RCNN [128] VGG	Full-Precision	32/32	68.9
	BWN [64]	1/32	62.5
	BNN [64]	1/1	47.3
Faster-RCNN [128] AlexNet	Full-Precision	32/32	66.0
	BWN [64]	1/32	62.1
	BNN [64]	1/1	46.4
SSD [132] DarkNet	Full-Precision	1/32	62.1
	ELBNN [121]	2/2	62.4
SSD [132] VGG-16	Full-Precision	32/32	62.1
	ELBNN [121]	2/2	46.4

4.3. Other Tasks

It is worth noting that most of the current binary neural networks that focus on image classification tasks cannot be directly generalized to other tasks. For different tasks, it is still highly required to design specific binary neural networks for the desirable performance. In addition to image classification task, there are also a few studies that designed and evaluated the binary neural network models for other tasks, such as object detection and semantic segmentation tasks. For the object detection task, Table 5 and Table 6 respectively list the performance of different binary neural networks on the COCO 2017 and PASCAL VOC 2007 datasets. For the semantic segmentation tasks, Table 7 compares different binary neural networks on the PASCAL VOC 2012 dataset. The experiments are based on different bit-width and network structures.

From Table 5 and 6 we can see that existing binarization algorithms have

Table 7: Semantic Segmentation Performance of Binary Neural Networks on PASCAL VOC 2012 Dataset

Topology	**Method**	**Bit-Width (W/A)**	**mAP (%)**
Faster-RCNN ResNet-18 FCN-32s [122]	Full-precision [122]	32/32	64.9
	LQ-Nets [61]	3/3	62.5
	Group-Net [122]	1/1	60.5
	Group-Net + BPAC [122]	1/1	63.8
	Group-Net**+BPAC [122]	1/1	65.1
Faster-RCNN ResNet-18 FCN-16s	Full-precision [122]	32/32	67.3
	LQ-Nets [61]	3/3	65.1
	Group-Net [122]	1/1	62.7
	Group-Net+BPAC [122]	1/1	66.3
	Group-Net**+BPAC [122]	1/1	67.7
Faster-RCNN ResNet-34 FCN-32s	Full-precision [122]	32/32	72.7
	LQ-Nets [61]	3/3	70.4
	Group-Net [122]	1/1	68.2
	Group-Net+BPAC [122]	1/1	71.2
	Group-Net**+BPAC [122]	1/1	72.8
Faster-RCNN ResNet-50 FCN-32s	Full-precision [122]	32/32	73.1
	LQ-Nets [61]	3/3	70.7
	Group-Net [122]	1/1	67.2
	Group-Net+BPAC [122]	1/1	70.4
	Group-Net**+BPAC [122]	1/1	71.0

achieved encouraging progress for the object detection task, and meanwhile bring the significant acceleration when deployed in real-world systems. But it also should be noted that the binary models still face a great challenge, especially when the activations are quantized to 1-bit. For semantic segmentation task, as shown in Table 7, the very recent method [122] achieved high accuracy using only 1-bit, which is almost the same as the full-precision model. But it is unknown how it works and the actual speed up of that method still needs to be verified.

Among these results, we found that although the binary neural networks perform well on the classification task, there are still unacceptable losses on other tasks. This makes binary neural networks designed for classification tasks hard to be directly applied to other tasks such as object detection and semantic segmentation. In the classification task, the network pays more attention to the global features, while ignoring the loss of local features caused by binarization. However, local features are more important in other tasks. So when designing binary neural networks for other tasks, the local features of the feature map need to be paid more attention.

5. Future Trend and Conclusions

The binary neural networks based on 1-bit representation enjoy the compressed storage and fast inference speed, but meanwhile suffer from the performance degradation. To bridge the gap between the binary and full-precision models, as we summarized in this survey, there are various solutions proposed in recent years, which can be roughly categorized into the naive and the optimized. Our analysis shows that optimizing the binary network using different techniques can promise better performance. These techniques, derived from different motivations, mainly focus on how to preserve the information in the forward propagation and how to optimize the network in the backward propagation. It shows that retaining the various information in forward and backward propagation is one of the key factors in training high-performance BNNs.

Although much progress has been made, existing techniques for binary neural networks still face the performance loss, especially for the large network and datasets. The main reasons might include: (1) it is still unclear what kind of network structure is suitable for binarization, so that the information passing through the network can be preserved, even after binarization. (2) it is a difficult problem to optimize the binary network in a discrete space, even we have the gradient estimator or approximate function for binarization. We believe more practical and theoretical studies will emerge to answer the two questions in the future.

Besides, as the mobile devices are becoming widely used in real world, more research efforts will be devoted to the applications to different tasks and deployment on different hardware. For example, [40] proposed a novel rotation consistent loss considering the open set characteristics of face recognition and achieves competitive performance using 4-bit compared to the full-precision model. Therefore, there will arise the interesting topics such as customizing or transferring binary networks for different tasks, designing hardware-friendly or energy-economic binarization algorithms, *etc.*

In addition to weights and activations, quantizing the backward propagation including gradients to accelerate the whole training process has arisen as a new topic recently. The unified framework proposed in [41] proves the possibility of 8-bit training of neural networks from the accuracy and speed aspect. It is worthy to further explore the feasibility of binarized backward calculation for faster training time.

Last but not the least, the research on explainable machine learning indicates that there are critical paths in the prediction of neural networks and different network structures follow different patterns. So it is also meaningful to design mix-precision strategy according to the importance of layer and devise new architectures that are friendly to the information flow of binary neural networks.

Acknowledgment

This work was supported by National Natural Science Foundation of China (61872021, 61772057 and 61690202), Beijing Nova Program of Science and Technology (Z191100001119050), BSFC No. 4202039, and the support funding from State Key Lab. of Software Development Environment and Jiangxi Research Institute of Beihang University.

References

[1] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: IEEE CVPR, 2015.

[3] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: NeurIPS, 2015.

[4] K. Nogueira, O. A. B. Penatti, J. A. dos Santos, Towards better exploiting convolutional neural networks for remote sensing scene classification, Pattern Recognition 61 (2017) 539–556.

[5] L. Liu, P. W. Fieguth, Y. Guo, X. Wang, M. Pietikäinen, Local binary features for texture classification: Taxonomy and experimental study, Pattern Recognition 62 (2017) 135–160.

[6] C. Deng, X. Liu, C. Li, D. Tao, Active multi-kernel domain adaptation for hyperspectral image classification, Pattern Recognition 77 (2018) 306–315.

[7] R. Girshick, Fast r-cnn, in: IEEE ICCV, 2015.

[8] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, D. Lin, Libra r-cnn: Towards balanced learning for object detection, in: IEEE CVPR, 2019.

[9] S. Ge, J. Li, Q. Ye, Z. Luo, Detecting masked faces in the wild with lle-cnns, in: IEEE CVPR, 2017.

[10] A. T. Lopes, E. de Aguiar, A. F. D. Souza, T. Oliveira-Santos, Facial expression recognition with convolutional neural networks: Coping with few data and the training sample order, Pattern Recognition 61 (2017) 610–628.

[11] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, NeurIPS.

[12] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: ICLR, 2015.

[13] L. Wu, Y. Wang, J. Gao, X. Li, Deep adaptive feature embedding with local sample distributions for person re-identification, Pattern Recognition 73 (2018) 275–288.

[14] J. Li, X. Liu, M. Zhang, D. Wang, Spatio-temporal deformable 3d convnets with attention for action recognition, Pattern Recognition 98.

[15] Y. Cheng, D. Wang, P. Zhou, T. Zhang, A survey of model compression and acceleration for deep neural networks, CoRR abs/1710.09282. arXiv:1710.09282.

[16] Y. Chen, T. Yang, J. S. Emer, V. Sze, Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices, IEEE J. Emerg. Sel. Topics Circuits Syst. 9 (2) (2019) 292–308.

[17] V. Sze, Y. H. Chen, J. Einer, A. Z. Suleiman, Z. Zhang, Hardware for machine learning: Challenges and opportunities, in: CICC, 2017.

[18] Y.-H. Chen, J. Emer, V. Sze, Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks, IEEE Micro PP (99) (2016) 1–1.

[19] Y. H. Chen, T. Krishna, J. Emer, V. Sze, Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks, in: ISSCC, 2016.

[20] Y. Chen, T. Krishna, J. S. Emer, V. Sze, Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks, IEEE Journal of Solid-State Circuits 52 (1) (2017) 127–138.

[21] C. Zhang, A. Liu, X. Liu, Y.-T. Xu, H. Yu, Y. Ma, T. Li, Interpreting and improving adversarial robustness with neuron sensitivity, CoRR abs/1909.06978.

[22] H. Yu, A. Liu, X. Liu, J.-C. Yang, C. Zhang, Towards noise-robust neural networks via progressive adversarial training, CoRR abs/1909.04839. arXiv:1909.04839.

[23] A. Liu, X. Liu, C. Zhang, H. Yu, Q. Liu, J. He, Training robust deep neural networks via adversarial noise propagation, CoRR abs/1909.09034. arXiv:1909.09034.

[24] Y. Izui, A. Pentland, Analysis of neural networks with redundancy, Neural Computation 2 (2) (1990) 226–238.

[25] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, S.-F. Chang, An exploration of parameter redundancy in deep networks with circulant projections, in: IEEE ICCV, 2015.

[26] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, in: NeurIPS, 2015.

[27] S. Srinivas, R. V. Babu, Data-free parameter pruning for deep neural networks, Computer Science (2015) 2830–2838.

[28] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE CVPR, 2016.

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

[30] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, in: IEEE ICCV, 2017.

[31] S. Ge, Z. Luo, S. Zhao, X. Jin, X. Zhang, Compressing deep neural networks for efficient visual inference, in: IEEE ICME, 2017, pp. 667–672.

[32] Y. Gong, L. Liu, M. Yang, L. D. Bourdev, Compressing deep convolutional networks using vector quantization, CoRR abs/1412.6115. arXiv:1412.6115.

[33] J. Wu, C. Leng, Y. Wang, Q. Hu, J. Cheng, Quantized convolutional neural networks for mobile devices, in: IEEE CVPR, 2016.

[34] V. Vanhoucke, A. W. Senior, M. Z. Mao, Improving the speed of neural networks on cpus, 2011.

[35] S. Gupta, A. Agrawal, K. Gopalakrishnan, P. Narayanan, Deep learning with limited numerical precision, ICML (2015) 1737–1746.

[36] S. Ge, Efficient Deep Learning in Network Compression and Acceleration, Digital Systems, Vahid Asadpour, IntechOpen, 2018.

[37] T. Zhao, X. He, J. Cheng, J. Hu, Bitstream: Efficient computing architecture for real-time low-power inference of binary neural networks on cpus, in: ACM MM, 2018, pp. 1545–1552.

[38] Q. Hu, G. Li, P. Wang, Y. Zhang, J. Cheng, Training binary weight networks via semi-binary decomposition, in: ECCV, 2018, pp. 657–673.

[39] S. Chen, W. Wang, S. J. Pan, Metaquant: Learning to quantize by learning to penetrate non-differentiable quantization, in: NeurIPS, 2019, pp. 3918–3928.

[40] Y. Wu, Y. Wu, R. Gong, Y. Lv, K. Chen, D. Liang, X. Hu, X. Liu, J. Yan, Rotation consistent margin loss for efficient low-bit face recognition, CoRR.

[41] F. Zhu, R. Gong, F. Yu, X. Liu, Y. Wang, Z. Li, X. Yang, J. Yan, Towards unified int8 training for convolutional neural network (2019). arXiv:1912.12607.

[42] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, R. Fergus, Exploiting linear structure within convolutional networks for efficient evaluation, in: NeurIPS, 2014.

[43] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, V. S. Lempitsky, Speeding-up convolutional neural networks using fine-tuned cp-decomposition, in: ICLR, 2015.

[44] M. Jaderberg, A. Vedaldi, A. Zisserman, Speeding up convolutional neural networks with low rank expansions, in: BMVC, 2014.

[45] V. Lebedev, V. Lempitsky, Fast convnets using group-wise brain damage, in: IEEE CVPR, 2016.

[46] J. Wen, B. Zhang, Y. Xu, J. Yang, N. Han, Adaptive weighted nonnegative low-rank representation, Pattern Recognition 81 (2018) 326–340.

[47] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, CoRR abs/1704.04861. arXiv:1704.04861.

[48] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: IEEE CVPR, 2018.

[49] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: IEEE CVPR, 2018, pp. 6848–6856.

[50] N. Ma, X. Zhang, H.-T. Zheng, J. Sun, Shufflenet v2: Practical guidelines for efficient cnn architecture design, in: ECCV, 2018, pp. 116–131.

[51] G. E. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, CoRR abs/1503.02531. arXiv:1503.02531.

[52] Z. Xu, Y. Hsu, J. Huang, Training shallow and thin networks for acceleration via knowledge distillation with conditional adversarial networks, in: ICLR, 2018.

[53] Y. Chen, Z. Zhang, N. Wang, Darkrank: Accelerating deep metric learning via cross sample similarities transfer, AAAI (2018) 2852–2859.

[54] J. Yim, D. Joo, J. Bae, J. Kim, A gift from knowledge distillation: Fast optimization, network minimization and transfer learning, in: IEEE CVPR, 2017.

[55] S. Zagoruyko, N. Komodakis, Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, in: ICLR, 2017.

[56] H. Ding, K. Chen, Q. Huo, Compressing CNN-DBLSTM models for OCR with teacher-student learning and tucker decomposition, Pattern Recognition 96.

[57] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized neural networks, in: NeurIPS, 2016.

[58] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, in: ECCV, 2016.

[59] M. Courbariaux, Y. Bengio, J.-P. David, Binaryconnect: Training deep neural networks with binary weights during propagations, in: NeurIPS, 2015.

[60] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, Y. Zou, Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, CoRR abs/1606.06160. arXiv:1606.06160.

[61] D. Zhang, J. Yang, D. Ye, G. Hua, Lq-nets: Learned quantization for highly accurate and compact deep neural networks, in: ECCV, 2018.

[62] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, K.-T. Cheng, Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm, in: ECCV, 2018.

[63] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, J. Yan, Differentiable soft quantization: Bridging full-precision and low-bit neural networks, in: IEEE ICCV, 2019.

[64] S. Sun, Y. Yin, X. Wang, D. Xu, W. Wu, Q. Gu, Fast object detection based on binary deep convolution neural networks, CAAI Transactions on Intelligence Technology 3 (4) (2018) 191–197.

[65] R. Li, Y. Wang, F. Liang, H. Qin, J. Yan, R. Fan, Fully quantized network for object detection, in: IEEE CVPR, 2019.

[66] J. Lin, C. Gan, S. Han, Defensive quantization: When efficiency meets robustness, in: ICLR, 2019.

[67] S. Zhu, X. Dong, H. Su, Binary ensemble neural network: More bits per network or more networks per bit?, in: IEEE CVPR, 2019.

[68] Y. Bengio, N. Léonard, A. C. Courville, Estimating or propagating gradients through stochastic neurons for conditional computation, CoRR abs/1308.3432. arXiv:1308.3432.

[69] M. Kim, P. Smaragdis, Bitwise neural networks, CoRR abs/1601.06071. arXiv:1601.06071.

[70] Z. Li, B. Ni, W. Zhang, X. Yang, W. Gao, Performance guaranteed network acceleration via high-order residual quantization, in: IEEE ICCV, 2017.

[71] X. Lin, C. Zhao, W. Pan, Towards accurate binary convolutional neural network, in: NeurIPS, 2017.

[72] P. Wang, Q. Hu, Y. Zhang, C. Zhang, Y. Liu, J. Cheng, Two-step quantization for low-bit neural networks, in: IEEE CVPR, 2018.

[73] Q. Hu, P. Wang, J. Cheng, From hashing to cnns: Training binary weight networks via hashing, in: AAAI, 2018.

[74] J. Choi, Z. Wang, S. Venkataramani, P. I. Chuang, V. Srinivasan, K. Gopalakrishnan, PACT: parameterized clipping activation for quantized neural networks, CoRR abs/1805.06085. arXiv:1805.06085.

[75] A. Mishra, E. Nurvitadhi, J. J. Cook, D. Marr, WRPN: Wide reduced-precision networks, in: ICLR, 2018.

[76] A. Bulat, G. Tzimiropoulos, Xnor-net++: Improved binary neural networks, CoRR abs/1909.13863. arXiv:1909.13863.

[77] J. Faraone, N. Fraser, M. Blott, P. H. Leong, Syq: Learning symmetric quantization for efficient deep neural networks, in: IEEE CVPR, 2018.

[78] M. Shen, X. Liu, K. Han, R. Gong, Y. Wang, C. Xu, Balanced binary neural networks with gated residual, in: ICASSP, 2020.

[79] B. Martinez, J. Yang, A. Bulat, G. Tzimiropoulos, Training binary neural networks with real-to-binary convolutions, in: ICLR, 2020.

[80] X. Chen, G. Liu, J. Shi, J. Xu, B. Xu, Distilled binary neural network for monaural speech separation, IJCNN.

[81] A. Polino, R. Pascanu, D. Alistarh, Model compression via distillation and quantization, in: ICLR, 2018.

[82] A. Mishra, D. Marr, Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy, in: ICLR, 2018.

[83] L. Hou, Q. Yao, J. T. Kwok, Loss-aware binarization of deep networks, in: ICLR, 2017.

[84] Y. G. L. X. Y. C. Aojun Zhou, Anbang Yao, Incremental network quantization: Towards lossless cnns with low-precision weights, in: ICLR, 2017.

[85] R. Ding, T.-W. Chin, Z. Liu, D. Marculescu, Regularizing activation distribution for training binarized deep networks, in: IEEE CVPR, 2019.

[86] Z. Wang, J. Lu, C. Tao, J. Zhou, Q. Tian, Learning channel-wise interactions for binary convolutional neural networks, in: IEEE CVPR, 2019.

[87] Y. Xu, X. Dong, Y. Li, H. Su, A main/subsidiary network framework for simplifying binary neural networks, in: IEEE CVPR, 2019.

[88] C. Liu, W. Ding, X. Xia, B. Zhang, J. Gu, J. Liu, R. Ji, D. Doermann, Circulant binary convolutional networks: Enhancing the performance of 1-bit dcnns with circulant back propagation, in: IEEE CVPR, 2019.

[89] Z. Cai, X. He, J. Sun, N. Vasconcelos, Deep learning with low precision by half-wave gaussian quantization, in: IEEE CVPR, 2017.

[90] S. Darabi, M. Belbahri, M. Courbariaux, V. P. Nia, BNN+: improved binary network training, CoRR abs/1812.11800. arXiv:1812.11800.

[91] P. Yin, S. Zhang, J. Lyu, S. Osher, Y. Qi, J. Xin, Blended coarse gradient descent for full quantization of deep neural networks, Research in the Mathematical Sciences 6 (1) (2019) 14.

[92] Y. Bai, Y.-X. Wang, E. Liberty, Proxquant: Quantized neural networks via proximal operators, in: ICLR, 2019.

[93] J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, X.-s. Hua, Quantization networks, in: IEEE CVPR, 2019.

[94] F. Lahoud, R. Achanta, P. Márquez-Neila, S. Süsstrunk, Self-binarizing networks, CoRR abs/1902.00730. arXiv:1902.00730.

[95] A. Bulat, G. Tzimiropoulos, J. Kossaifi, M. Pantic, Improved training of binary networks for human pose estimation and image recognition, CoRR abs/1904.05868. arXiv:1904.05868.

[96] H. Qin, R. Gong, X. Liu, M. Shen, Z. Wei, F. Yu, J. Song, Forward and backward information retention for accurate binary neural networks.

[97] Z. Xu, R. C. C. Cheung, Accurate and compact convolutional neural networks with trained binarization, CoRR abs/1909.11366. arXiv:1909.11366.

[98] H. Li, S. De, Z. Xu, C. Studer, H. Samet, T. Goldstein, Training quantized nets: A deeper understanding, in: NeurIPS, 2017.

[99] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, CoRR abs/1804.02767. arXiv:1804.02767.

[100] S. Liang, S. Yin, L. Liu, W. Luk, S. Wei, Fp-bnn: Binarized neural network on fpga, Neurocomputing 275 (2018) 1072 – 1086.

[101] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M. Srivastava, R. Gupta, Z. Zhang, Accelerating binarized convolutional neural networks with software-programmable fpgas, in: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays.

[102] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, K. Vissers, Finn: A framework for fast, scalable binarized neural network inference, in: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays.

[103] C. Fu, S. Zhu, H. Su, C.-E. Lee, J. Zhao, Towards fast and energy-efficient binarized neural network inference on fpga, in: Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays.

[104] H. Yang, M. Fritzsche, C. Bartz, C. Meinel, Bmxnet: An open-source binary neural network implementation based on mxnet, in: Proceedings of the 2017 ACM on Multimedia Conference.

[105] J. Bethge, M. Bornstein, A. Loy, H. Yang, C. Meinel, Training competitive binary neural networks from scratch, CoRR abs/1812.01965. arXiv:1812.01965.

[106] J. Zhang, Y. Pan, T. Yao, H. Zhao, T. Mei, dabnn: A super fast inference framework for binary neural networks on ARM devices, in: ACM MM, 2019.

[107] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[108] M. Blott, T. B. Preusser, N. J. Fraser, G. Gambardella, K. Obrien, Y. Umuroglu, M. Leeser, K. Vissers, Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks, ACM TRETS 11 (3) (2018) 16.

[109] M. Ghasemzadeh, M. Samragh, F. Koushanfar, Rebnet: Residual binarized neural network, in: IEEE FCCM, 2018, pp. 57–64.

[110] P. Jokic, S. Emery, L. Benini, Binaryeye: A 20 kfps streaming camera system on fpga with real-time on-device image recognition using binary neural networks, in: IEEE SIES, 2018, pp. 1–7.

[111] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, Reading digits in natural images with unsupervised feature learning, in: NeurIPS, 2011.

[112] P. Guo, H. Ma, R. Chen, P. Li, S. Xie, D. Wang, Fbna: A fully binarized neural network accelerator, in: IEEE FPL, 2018, pp. 51–513.

[113] A. Krizhevsky, Learning multiple layers of features from tiny images, University of Toronto.

[114] Y. Zhou, S. Redkar, X. Huang, Deep learning binary neural network on an fpga, IEEE MWSCAS (2017) 281–284.

[115] H. Nakahara, H. Yonekawa, T. Sasao, H. Iwamoto, M. Motomura, A memory-based realization of a binarized deep convolutional neural network, in: IEEE FPT, 2016, pp. 277–280.

[116] N. J. Fraser, Y. Umuroglu, G. Gambardella, M. Blott, P. H. W. Leong, M. Jahre, K. A. Vissers, Scaling binarized neural networks on reconfigurable logic, in: PARMA-DITAM@HiPEAC, 2017.

[117] H. Nakahara, T. Fujii, S. Sato, A fully connected layer elimination for a binarizec convolutional neural network on an fpga, in: IEEE FPL, 2017, pp. 1–4.

[118] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, F. F. Li, Imagenet: a large-scale hierarchical image database, in: IEEE CVPR, 2009.

[119] H. Yonekawa, H. Nakahara, On-chip memory based binarized convolutional deep neural network applying batch normalization free technique on an fpga, in: IEEE IPDPSW, 2017, pp. 98–105.

[120] J. Kung, D. Zhang, G. van der Wal, S. Chai, S. Mukhopadhyay, Efficient object detection using embedded binarized neural networks, Journal of Signal Processing Systems.

[121] C. Leng, Z. Dou, H. Li, S. Zhu, R. Jin, Extremely low bit neural network: Squeeze the last bit out with admm, in: AAAI, 2017.

[122] B. Zhuang, C. Shen, M. Tan, L. Liu, I. Reid, Structured binary neural networks for accurate image classification and semantic segmentation, in: IEEE CVPR, 2019.

[123] S. Cao, L. Ma, W. Xiao, C. Zhang, Y. Liu, L. Zhang, L. Nie, Z. Yang, Seernet: Predicting convolutional neural network feature-map sparsity through low-bit quantization, in: IEEE CVPR, 2019.

[124] M. Alizadeh, J. Fernndez-Marqus, N. D. Lane, Y. Gal, A systematic study of binary neural networks' optimisation, in: ICLR, 2019.

[125] B. Zhuang, C. Shen, M. Tan, L. Liu, I. Reid, Towards effective low-bitwidth convolutional neural networks, in: IEEE CVPR, 2018.

[126] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, International Journal of Computer Vision 88 (2) (2010) 303–338.

[127] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, C. L. Zitnick, Microsoft coco: Common objects in context, Lecture Notes in Computer Science (2014) 740755.

[128] S. Ren, K. He, R. Girshick, S. Jian, Faster r-cnn: Towards real-time object detection with region proposal networks, in: NeurIPS, 2015.

[129] T. Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss for dense object detection, IEEE Transactions on Pattern Analysis & Machine Intelligence PP (99) (2017) 2999–3007.

[130] R. Krishnamoorthi, Quantizing deep convolutional networks for efficient inference: A whitepaper, CoRR abs/1806.08342. arXiv:1806.08342.

[131] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, D. Kalenichenko, Quantization and training of neural networks for efficient integer-arithmetic-only inference, in: IEEE CVPR, 2018.

[132] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, A. C. Berg, SSD: single shot multibox detector, in: ECCV, 2016.