

A comprehensive review of Binary Neural Network

Chunyu Yuan¹ · Sos S. Agaian^{1,2}

Abstract

Deep learning (DL) has recently changed the development of intelligent systems and is widely adopted in many real-life applications. Despite their various benefits and potentials, there is a high demand for DL processing in different computationally limited and energy-constrained devices. It is natural to study game-changing technologies such as Binary Neural Networks (BNN) to increase deep learning capabilities. Recently remarkable progress has been made in BNN since they can be implemented and embedded on tiny restricted devices and save a significant amount of storage, computation cost, and energy consumption. However, nearly all BNN acts trade with extra memory, computation cost, and higher performance. This article provides a complete overview of recent developments in BNN. This article focuses exclusively on 1-bit activations and weights 1-bit convolution networks, contrary to previous surveys in which low-bit works are mixed in. It conducted a complete investigation of BNN's development -from their predecessors to the latest BNN algorithms/techniques, presenting a broad design pipeline and discussing each module's variants. Along the way, it examines BNN (a) purpose: their early successes and challenges; (b) BNN optimization: selected representative works that contain essential optimization techniques; (c) deployment: open-source frameworks for BNN modeling and development; (d) terminal: efficient computing architectures and devices for BNN and (e) applications: diverse applications with BNN. Moreover, this paper discusses potential directions and future research opportunities in each section.

Keywords: Binary Neural Network, Convolution Neural Network, Model compression and acceleration, Binarization, Quantization

1. Introduction

1.1. Background

Artificial intelligence (AI) means the simulation of human intelligence in machines. Because of increased volume of big data, continually advanced algorithms, and incessant improvements in hardware, AI is growing to be one of the most popular topics in today's world. In recent years, AI holds tremendous promise to power nearly all aspects of society. In AI community, convolution neural networks (CNN) is one common method to solve vision problems such as image classification(Lu and Weng, 2007; Nath *et al.*, 2014; Li *et al.*, 2018a; Deepa *et al.*, 2011; Wang *et al.*, 2019a), object detection(Zou *et al.*, 2019; Liu *et al.*, 2020a; Borji *et al.*, 2019; Shantaiya, Verma, and Mehta, 2013; Jiao *et al.*, 2019) and object recognition(Sukanya, Gokul, and Paul, 2016; Goyal and Benjamin, 2014; Jafri *et al.*, 2014; Campbell and Flynn, 2001; Zhou *et al.*, 2021).

Although new CNN models were continually presented and advanced, such as ResNeXt(Xie *et al.*, 2017), SE-Net(Hu, Shen, and Sun, 2018) and SK-Net(Li *et al.*, 2019b), the CNN architectures don't change too much compared to that before 2017. And as CNN models become larger so that they require more computational

Chunyu Yuan
cyuan1@gradcenter.cuny.edu

Sos S. Agaian
sos.agaian@csi.cuny.edu

¹ The Graduate Center, City University of New York

² College of Staten Island, City University of New York

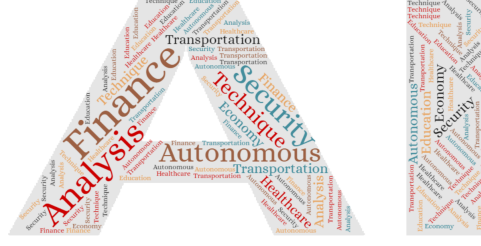


Figure 1. Popular topics using AI

power and storage, they cannot be equipped on resource-constraint platforms such as smart phones and tiny Internet of Things (IoT) devices. Therefore, it is reasonable to generate a new open problem, which is how to develop more compact, lightweight and efficient power networks which can simultaneously maintain acceptable accuracy. So that trained models can be effectively utilized on devices that billions of customers use in their everyday lives.

Model Compression and Acceleration for deep neural networks is one type of solution to solve the problem mentioned above, trying to save the memory and reduce computational costs of CNN while still offering similar capabilities of full-precision CNN models. Based on solutions' properties, this type can be subdivided into five major categories: parameters quantization, parameters pruning, low-rank matrix factorization, transferred/compact convolutional filters and knowledge distillation.

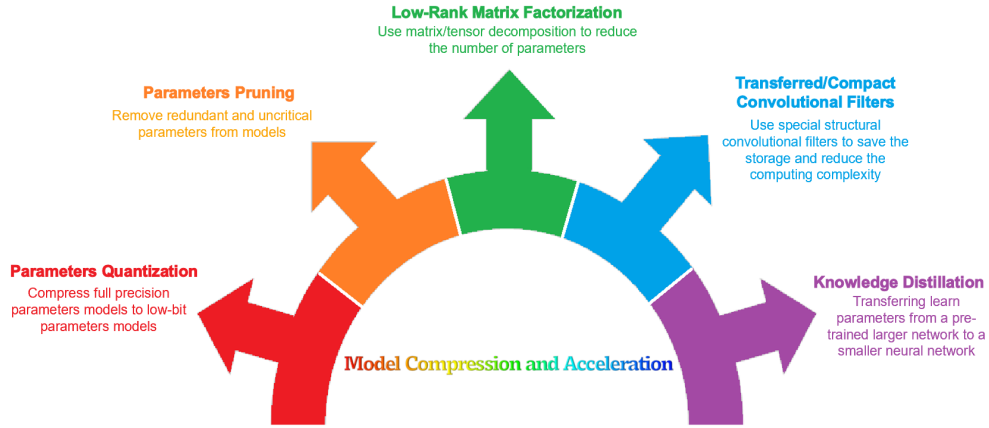


Figure 2. Topics of Model Compression and Acceleration

1.2. Motivation

It is known that current deep neural network models are computationally expensive and use memory resources extensively. For example, (Krizhevsky, Sutskever, and Hinton, 2012) designed AlexNet, which contains 60 million float-point parameters and 650,000 neurons, on the Imagenet dataset. (Simonyan and Zisserman, 2014)'s model VGG-16, when applied to the same dataset, has over 130 million float-point parameters. Large-weight models are difficult to deploy on tiny devices with limited resources. It is natural to study game-changing technologies such as BNN to increase deep learning capabilities. BNN method is an extreme case of parameters quantization methods. Completing activations and weights to 1-bit values can theoretically have 32 times lower memory storage and 58 times faster inference speed than traditional 32-bit CNN. BNN can be used in a variety of problems including classification (Chen *et al.*, 2021; Qin *et al.*, 2020b), pattern recognition (Qiao *et al.*, 2020), computer vision (Frickenstein *et al.*, 2020), natural language processing (NLP) (Xiang, Qian, and Yu, 2017; Qian and Xiang, 2019; Gao *et al.*, 2021), etc. Because of BNN's incredible advantages with fewer size parameters and faster inference speed can be easily applied and embedded on resource-limited devices such as wearable devices and tiny sensors. In recent years, with the increasing trend on lightweight and practical networks, more and more researchers are turning their attention to BNN. In 2021, a workshop spotlights BNN called binary networks for computer vision held by Computer Vision and Pattern Recognition (CVPR). BNN has become one popular and important research topic in the AI community.

1.3. Related Work

There are a few published surveys on BNN that are (Simons and Lee, 2019) and (Qin *et al.*, 2020a). However, by carefully checking those surveys, we find some of the previous representative BNN techniques were not reviewed and discussed. Even worse, some benchmark results in the literature were wrongly collected and cited. For example, the results on the dataset COCO-2017 (Caesar, Uijlings, and Ferrari, 2018) from a 4-bit quantization network called FQN (Li *et al.*, 2019a) were mistaken as the BNN in (Qin *et al.*, 2020a). Efficiency indicators measured on FPGA were incorrectly cited in both of (Simons and Lee, 2019) and (Qin *et al.*, 2020a). Besides, in the most recent year, there is a great number of new BNN methods published. Some of them have already been crucially improved BNN performance and have generated new directions that were not included and discussed in those previous survey works. For those reasons, we present a new extensive review of BNN which covers all the BNN design pipeline topics including algorithms, deployment, and applications. Especially, different from prior surveys that mixed low-bit networks reviews, we only focus on reviewing the pure and truthful BNN where has 1-bit activations and weights in the convolution. We accurately summarize the BNN’s major optimization techniques and subdivided them into five categories to discuss. More importantly, we noticed that each previous work may contain several optimization techniques. So we don’t simply relegate each work to one category. More carefully, we utilize tables to list all the associated works with their contributions in that category. To the best of our ability, we collect all previous BNN works which were published in realizable conferences and journals till the date. We believe this work can serve as educational materials on the topic of BNN, and as reference information for professional researchers in this field.

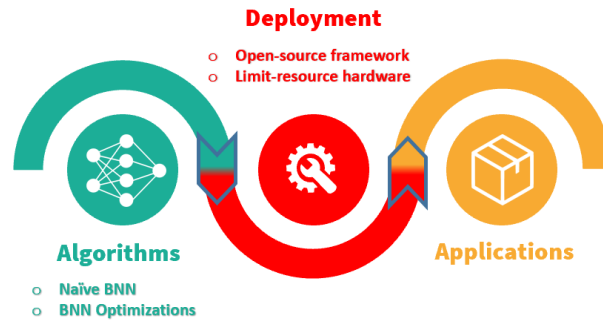


Figure 3. Topics covered in this survey paper

1.4. Organization

The organization of this work is structured as follows. Section 2 introduces the basic principles of BNN and their early successes and challenges. Section 3 mainly reviews the optimization methods for BNN based on selected representative works that contain vital optimization techniques. Section 4 reviews the open-source frameworks for the BNN modeling. Section 5 introduces popular efficiency hardware platforms and their definitions of common terms. Section 6 presents the recent BNN applications, including the associated tables with the performance progress history. Section 7 is our conclusion and summary.

2. Binary Neural Network

2.1. What is BNN?

BNN is a type of neural network that activations(or called features) and weights are 1-bit values in all the hidden layers (except the input and output layers). In a few words, BNN is an extremely compacted case of CNN. Because BNN and CNN have the same structures except for the different precision activations and weights. BNN also specifically refers to BNN techniques that compact 32-bit activations and weights to 1-bit values. The process of compacting 32-bit to 1-bit values is binarization. The purpose of binarization not only can save the expensive model’s storage, but also reduce the matrix computation costs by using XNOR and popcount operations. (Rastegari *et al.*, 2016) reported that BNN can have 32 times lower memory saving and 58 times faster convolution operations than 32-bit CNN. In traditional CNN, the vast majority of computing

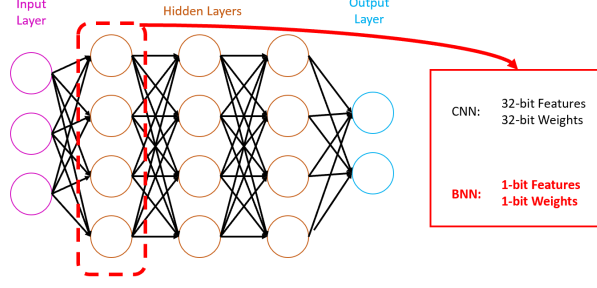


Figure 4. An artificial neural network

cost spent in matrix multiplication inside the convolution operation. The basic convolution operation without bias can be expressed as:

$$Z = I * W \quad (1)$$

where I and W represent activations and weights, respectively, Z is the output of the convolution operation with matrix multiplication. Because in such a multiplication operation, it contains a large of floating-point operations, including floating-point multiplication and floating-point addition, which is the reason for low-speed performance in the neural network inference. To resolve this issue, Courbariaux *et al.* (2016) and Kim and Smaragdis (2016) separately proposed their vanilla BNN architectures.

Table 1. Naive BNN

BNN Name	Key Techniques
Binarized Neural Networks (Courbariaux <i>et al.</i> , 2016)	FP: $\text{sign}(x)$ BP: $\text{clip}(x, -1, 1) = \max(-1, \min(1, x))$
Bitwise Neural Networks (Kim and Smaragdis, 2016)	FP: $\text{sign}(x)$ BP: two steps, similar to $\text{clip}(x, -1, 1) = \max(-1, \min(1, x))$

Note: **FP**: Forward Propagation, **BP**: Backward Propagation .

An artificial neural network consists of two processes: forward propagation and backward propagation. Forward propagation is the process of moving from the input layer (left) to the output layer (right) in the figure. 4, which also refers model inference. Backward propagation is the process of moving from the output layer (right) to the input layer (left) in the figure. 4, which represent the process of fine-tuning the model’s weights. Subsection 2.2 and 2.3 discuss how BNN works in forward propagation and backward propagation.

2.2. Forward Propagation

The neural cell is a basic computation structure in the forward path of a neural network. Different from the 32-bit CNN, neural cell in the BNN’s forward path adds the binarization steps to the activations I and weights W before convolution operation. The binarization steps’ purpose is to represent the floating-point activations and weights using 1-bit. Figure. 5 presents the difference in computation steps inside a neural cell along the forward path between naive BNN and 32-bit CNN.

The sign function is widely used for binarization:

$$\text{Sign}(x) = \begin{cases} +1, & \text{if } x \geq 0, \\ -1, & \text{otherwise.} \end{cases} \quad (2)$$

After binarization, activations I and weights W will be:

$$I \approx \text{sign}(I) = B_I \quad (3)$$

$$W \approx \text{sign}(W) = B_W \quad (4)$$

where B_I and B_W are binary activations and binary weights, respectively.

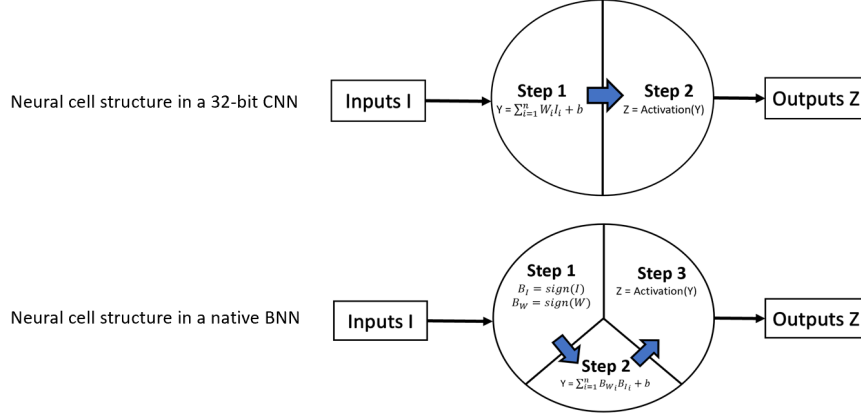


Figure 5. Neural cell structures of native BNN and 32-bit CNN

Table 2. BNN XNOR Operations

Binary Activations	Binary Weights	XNOR Result
-1(0)	-1(0)	+1(1)
-1(0)	+1(1)	-1(0)
+1(1)	-1(0)	-1(0)
+1(1)	+1(1)	+1(1)

Because B_I and B_W values are $\{+1, -1\}$ which has the similar XNOR results compared to $\{0, +1\}$ as table 2 shown. Then, we can use bitwise XNOR and popcount to replace expansive matrix multiplication calculation. Figure. 6 presents the examples of convolution computation processes in native BNN and 32-bit CNN.

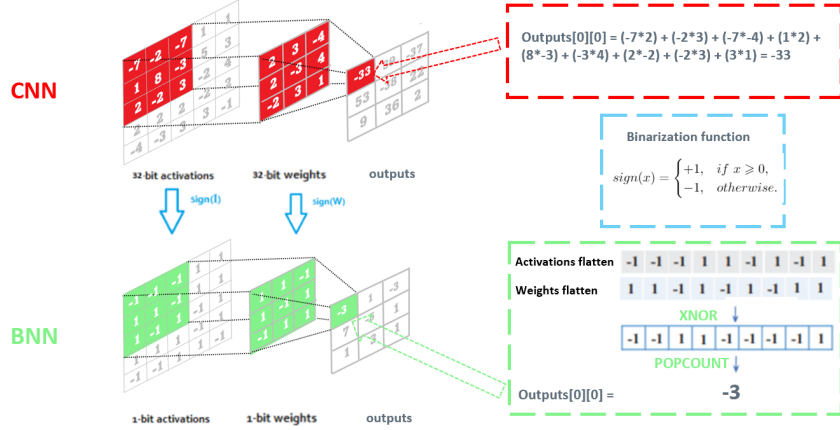


Figure 6. naive BNN's forward propagation compared to CNN's

2.3. Backward Propagation

Because the derivation result of binarization function (sign) is 0. Binary weights cannot be learned with the traditional gradient descent method based on a backward propagation algorithm. To resolve this issue, Binarized-Neural-Networks (Courbariaux *et al.*, 2016) apply the technique called straight-through estimator (STE) (Tieleman and Hinton, 2012; Bengio, Léonard, and Courville, 2013) to learn binary weights in backward propagation. Figure 7 explains the process of learning Binarization weights in Binarized-Neural-Networks. During the BNN training steps, each layer's real weights are kept and updated using STE. After training, binarized weights are saved and the real weights are discarded. Besides, Bitwise-Neural-Networks

(Kim and Smaragdis, 2016) contains two steps to train the BNN model. The first step is to train some compressed network parameters in real-value networks with weight compression. Then, the authors initialize the real-valued parameters for the target bitwise neural network, and adopt a training strategy that is similar to STE.

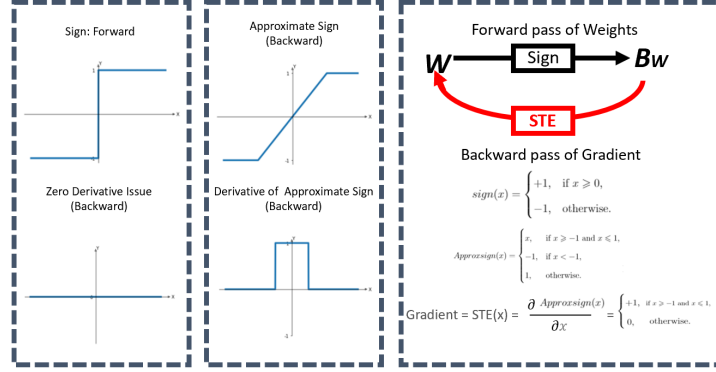


Figure 7. Binarized Neural Networks' Backward propagation

2.4. Summary

Although naive BNN has faster inference speeds and smaller weight sizes, the accuracy performance is much lower than that using full-precision CNN in the early stages. The reason is the severe information loss due to parameter binarization, including binary activations and binary weights. To address the above issue, a variety of novel optimization solutions have been proposed in recent years. In the next section, we regulate these methods into categories.

3. Binary Neural Network Optimization

To report as the latest solutions at the published time, each BNN model contained several optimization and improvement points/methods. We regulate these enhancement methods to 5 categories as the figure 8 shown: (1) quantization error minimization, (2) loss function improvement, (3) gradient approximation, (4) network topology structure and (5) training strategy and tricks.

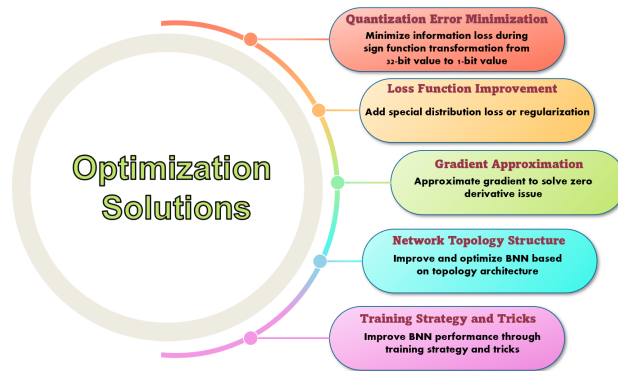


Figure 8. BNN enhancement methods

3.1. Quantization Error Minimization

3.1.1. Scaling Factor

To reduce the information loss during sign function transformation from 32-bit value to 1-bit value, XNOR-Net (Rastegari *et al.*, 2016) adds channel-wise scaling factors α and β for activations and weights. Therefore, equations 3 and 4 can be changed to:

$$I \approx \alpha * \text{sign}(I) = \alpha * B_I \quad (5)$$

$$W \approx \beta * \text{sign}(W) = \beta * B_W \quad (6)$$

where α and β are :

$$\alpha = \frac{1}{n} \|I\|_{L_1} \quad (7)$$

$$\beta = \frac{1}{n} \|W\|_{L_1} \quad (8)$$

Therefore, equation 1 can be changed:

$$Z = I * W \approx (\alpha * B_I) * (\beta * B_W) = (\alpha * \beta) * (B_I \otimes B_W) \quad (9)$$

Hadamard matrices (Agaian, 1986) have the same properties as binarized matrices in which all the values are +1 or -1. Advancing on top of XNOR-Net, HadaNet (Akhauri, 2019) applies the concept of hadamard transforms (Agaian *et al.*, 2011) to binary activations and weights without increasing filter map counts. Also, XNOR-Net++ (Bulat and Tzimiropoulos, 2019) proposes to merge the activation and weight scaling factors into a single one, and explore various ways to construct the shape of scaling factor based on input, output, channel and their combinations. Later, Zhao *et al.* (2021) design DA-BNN which is a data-adaptive method that can generate an adaptive amplitude based on spatial and channel attention.

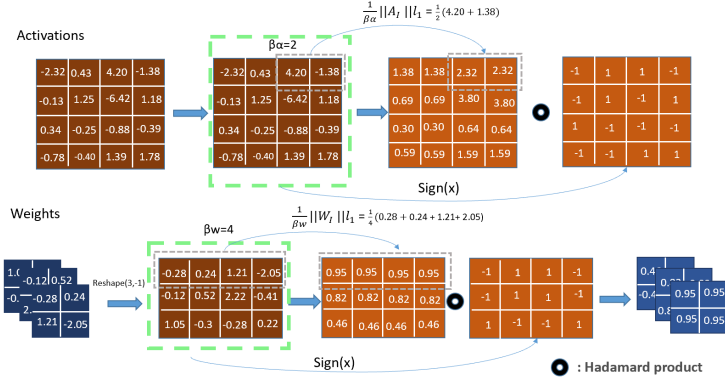


Figure 9. Binarization process in HadaNet

3.1.2. Quantization Function

Different using sign functions to do activation and weight binarization, several prior works present new methods to binary the parameters to [-1, +1]. DoReFa-Net (Zhou *et al.*, 2016), UniQ (Pham, Abraham, and Chung, 2021), Quantization-Networks (Yang *et al.*, 2019) and DSQ (Gong *et al.*, 2019) propose the k-bit method for parameter quantization including binarization. Their 1-bit methods provide a different way to binary parameters compared to that used sign function. SI-BNN (Wang *et al.*, 2020a) proposes and finds that binary activations to [0, +1] and binary weights to [-1, +1] can alleviate information loss. Rectified clamp unit (ReCU) (Xu *et al.*, 2021b) is a weights standardization method to reveal the inherent contradiction between minimizing the quantization error and maximizing the information entropy in BNN.

3.1.3. Activations/Weights distribution and Others

Different to directly optimization the binarization process in the convolution layer, IR-Net (Qin *et al.*, 2020c), BBG-Net (Shen *et al.*, 2020), SLB (Yang *et al.*, 2020), RBNN (Lin *et al.*, 2020) and IA-BNN (Kim *et al.*, 2020b) optimize and reshape and activations and weights distribution before binarization in their models. LAB2 (Hou, Yao, and Kwok, 2016) applies the proximal Newton algorithm to binary weights by directly considering the binarization loss. HORQ (Li *et al.*, 2017) proposes to use recursive binary quantization to lighten information loss. CI-BCNN (Wang *et al.*, 2019c), via learning reinforcement graph model, mines the channel-wise interactions to iterate popcount and reduce inconsistency of signs in binary feature maps and preserves the information of input samples. LNS (Han *et al.*, 2020) proposes to train binarization function to predict binarization weights via supervision noise learning. ProxyBNN (He *et al.*, 2020) constructs the pre-binarization weights matrix using the basis and coordinates submatrix to reduce information loss after binarization.

Table 3. Summary Table for Quantization Error Minimization

BNN Name	Code Available	Key Idea	Idea Category
XNOR-NET(2016)	✓ (O,UO)	Channel-wise scaling factor α and β for activations and weights	▲
LAB2(2016)	✓ (O)	Minize the weight binarization loss through the proximal Newton algorithm with diagonal Hessian approximation	●
DoReFa-Net(2016)	✓ (O,UO)	New quantization method to get binary or low bitwidth weights and activations using low bitwidth parameter gradients	■
HORQ(2017)	✗	Order-Two Residual Quantization to alleviate information loss	●
HadaNet(Akhauri, 2019)	✓ (O)	Apply hadamard product to binarization	▲
XNOR-Net++(2019)	✗	Several ways to construct the scale factors	▲
Quantization Networks (2019)	✓ (O)	Soft quantization function: formulate quantization as a differentiable non-linear mapping function based on sigmoid	■
CI-BCNN(2019c)	✗	Interacted to alleviate xnor and popcount quantization error via channel-wise interaction by a reinforcement graph model	●
DSQ(2019)	✓ (O)	Soft quantization function: formulate quantization as a differentiable non-linear mapping function based on tanh	■
IR-Net(2020c)	✓ (O)	Libra Parameter Binarization: a implicit rectifier that reshapes the data distribution before binarization	●
BBG-Net(2020)	✗	Maximizing entropy with balanced binary weights	●
SI-BNN(2020a)	✗	Binary activations to 0 or +1 and binary weights to -1 or +1 to remains most information	■
LNS(2020)	✗	Binary weights mapping with noisy supervision	●
SLB(2020)	✗	State batch normalization and low-bit search including binary	●
ProxyBNN(2020)	✗	Learning orthogonal matrix basis coefficients to and construct the pre-binarization weights	●
RBNN(2020)	✓ (O)	Rotate the full precision weight vector to its binary vector to reduce the angular bias	●
UniQ(2021)	✓ (O)	Symmetric quantizer with a trainable step size	■
IA-BNN(2020b)	✗	Unbalanced Distribution of binary activations actually improves the accuracy of BNN by shifting the trainable thresholds of binary activations	●
DA-BNN (2021)	✗	Data-adaptive re-scaling	▲
ReCU(2021b)	✓ (O)	Weights standardization	●

Note: O: Official implementation, UO: Un-official implementation, ▲: Scaling Factor, ■: Quantization Function, ●: Activations/Weights distribution and Others

3.2. Loss Function Improvement

To close the accuracy gap from real-valued networks, How-to-Train (Tang, Hua, and Wang, 2017), Binarized-Convolutional (Bulat and Tzimiropoulos, 2017), BNN-RBNT (Darabi *et al.*, 2018), PCNN (Gu *et al.*, 2019b), BNN-DL (Ding *et al.*, 2019), CCNN (Xu and Cheung, 2019), BONN (Gu *et al.*, 2019a) and RBNN (Lin *et al.*, 2020) propose adding distribution loss or special regularization to the overall loss function. Their basic types can be expressed as :

$$\mathbf{L}_T = \mathbf{L}_S + \lambda \mathbf{L}_{DR} \quad (10)$$

where L_T is total loss, L_S is a cross-entropy loss, L_{DR} is the added special distribution loss or regularization and λ is a balancing hyper-parameter. LNS (Han *et al.*, 2020), Real-to-Bin (Martinez *et al.*, 2020) and ReActNet (Liu *et al.*, 2020c) proposes special loss functions for servicing transfer learning strategy.

3.3. Gradient Approximation

As the derivative result of sign function equals to zero, it leads weights fail to get updated in the back-propagation. Straight-through estimator (STE) is one available method to approximate sign gradients.

Table 4. Summary table for BNN that proposed or used techniques for BNN loss function

BNN Name	Code Available	Key Idea	Idea Category
How to Train(2017)	✗	New regularization to replace L2 regularization (Euclidean)	▲
Binarized Convolutional (2017)	✓ (O)	Sigmoid cross-entropy pixel-wise loss function	▲
BNN-RBNT(2018)	✗	Add regularization functions(Manhattan or Euclidean) to the overall loss function	▲
PCNN(2019b)	✓ (O)	Add projection loss to jointly learned with the conventional cross-entropy loss	▲
BNN-DL(2019)	✓ (O)	Add distribution loss to the overall loss function	▲
CCNN(2019)	✗	L2 regularization term acting on the weight scaling factors	▲
BONN(2019a)	✓ (O)	Add Bayesian kernel loss and Bayesian feature loss to the overall loss function	▲
RBNN(2020)	✗	Add kernel approximation and adversarial learning loss to the overall loss function	▲
LNS(2020)	✗	Unbiased auxiliary loss for binary weights mapping	■
Real-to-Bin(2020)	✓ (UO)	Standard logit matching loss for attention transfer between BNN and real-valued networks	■
ReActNet(2020c)	✓ (O)	Distributional Loss to learn similar between BNN and real-valued networks	■

Note: O: official implementation, UO: Un-official implementation, ▲: Non-transferring Learning, ■: Transferring Learning

However, using STE fails to learn weights near the borders of -1 and $+1$, that greatly harms the updating ability of back propagation. GB-Net (Sakr *et al.*, 2018) uses true gradient-based learning to train BNN with parametrized clipping functions (PCF) and replace PCF by scaled binary activation function (SBAF) to obtain final BNN interface. BNN-RBNT (Darabi *et al.*, 2018) proposes a backward approximation based on the sigmoid function. Bi-Real-Net (Liu *et al.*, 2018) proposed a polynomial steps function to approximate forward sign function. CCNN (Xu and Cheung, 2019) introduces the derivation estimator to approximate their binarization function. CBCN (Liu *et al.*, 2019a) designs the gradient approximation based on Gaussian functions. Although the authors of CBCN presented the function’s characteristics and displayed the function graph, we don’t know the detailed implementation for their function and in their open source code, we find they in effect use Bi-Real-Net’s method for gradient approximation. IR-Net (Qin *et al.*, 2020c) and RBNN (Lin *et al.*, 2020) separately design a dynamic gradient estimator which can adjust the gradient approximation during the training process. SI-BNN (Wang *et al.*, 2020a) designs their gradient estimator with two trainable parameters on the top of STE. BinaryDuo (Kim *et al.*, 2020a) Quantitatively analyzed the differentiable approximation function and proposed to use the gradient of smoothed loss function to estimate the gradient. Table 5 is a summary table for BNN that proposes techniques for gradient approximation.

3.4. Network Topology Structure

Network architectures can affect BNN performance. ABC-Net (Lin, Zhao, and Pan, 2017), CBCN (Liu *et al.*, 2019a), Bi-Real-Net (Liu *et al.*, 2018), WPRN (Mishra *et al.*, 2017), Group-Net (Zhuang *et al.*, 2019), BBG-Net (Shen *et al.*, 2020) and Real-to-Bin (Martinez *et al.*, 2020) propose to modified classical network (e.g. ResNet) to improve accuracy performance. BENN (Zhu, Dong, and Su, 2019) proposes to leverage ensemble BNN to improve prediction performance. BinaryDenseNet (Bethge *et al.*, 2019) proposes methods and constructs customized BNN special dense network to improve accuracy performance against similar model size BNN. Search Accurate (Shen *et al.*, 2019) and DMS (Li *et al.*, 2020), via designing search algorithm, adjust the number of channel to close the accuracy gap compared to full-precision network. BATS (Bulat, Martinez, and Tzimiropoulos, 2020a), BNAS (Kim, Singh, and Choi, 2020), NASB (Zhu, Al-Ars, and Hofstee, 2020) and High-Capacity-Expert (Bulat, Martinez, and Tzimiropoulos, 2020b), through designed NAS methods, search architectures for BNN to compare accuracy performance with other BNNs which have similar model sizes and binarized from classic network (e.g. ResNet). Especially, High-Capacity-Expert (Bulat, Martinez, and Tzimiropoulos, 2020b) first applied condition computing within BNN called expert convolution and combined it with grouped convolution. Inspired by MobileNet-v1, MoBiNet-Mid (Phan *et al.*, 2020b) and Binarized MobileNet (Phan *et al.*, 2020a) propose new BNN architecture with high

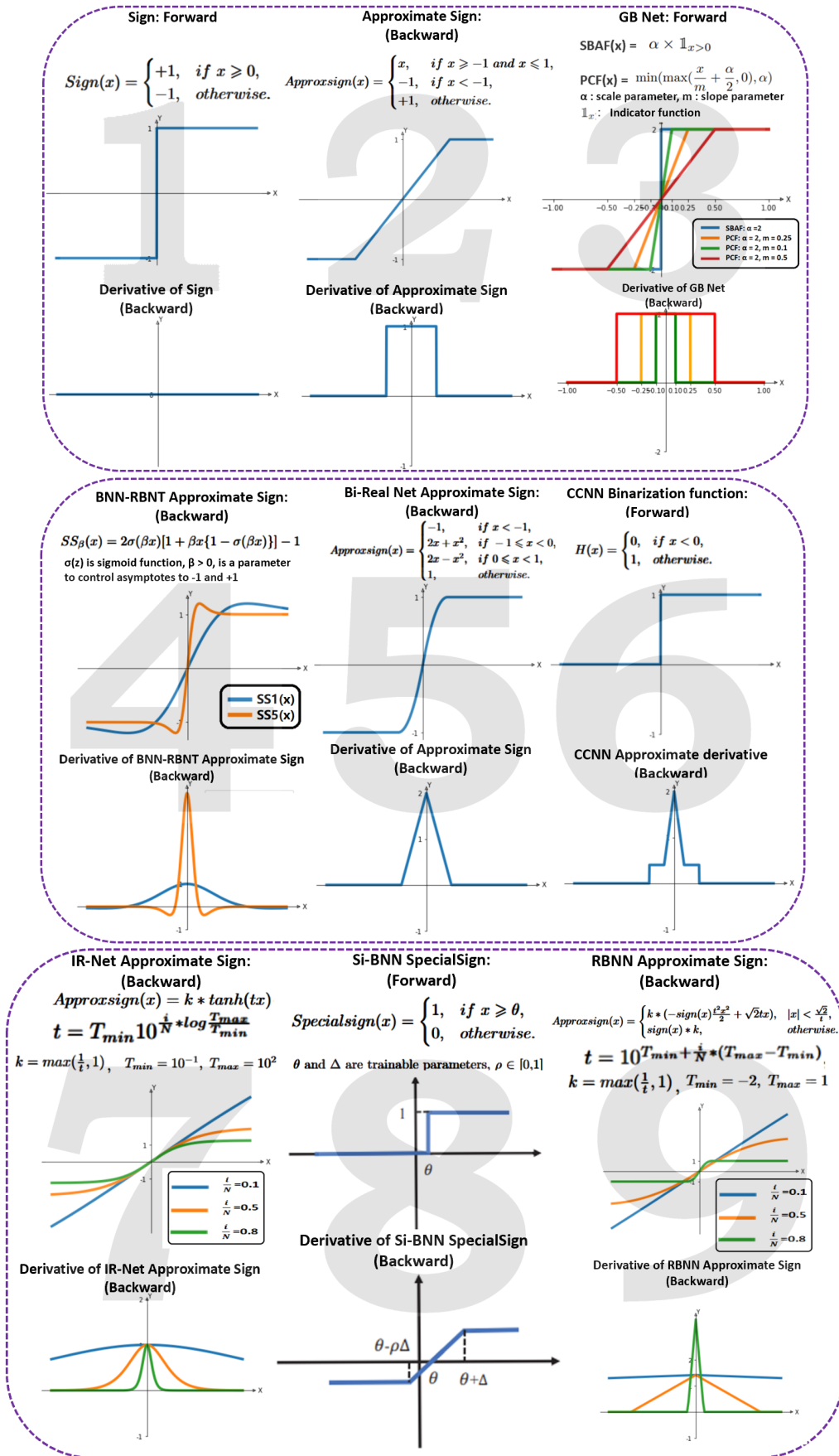


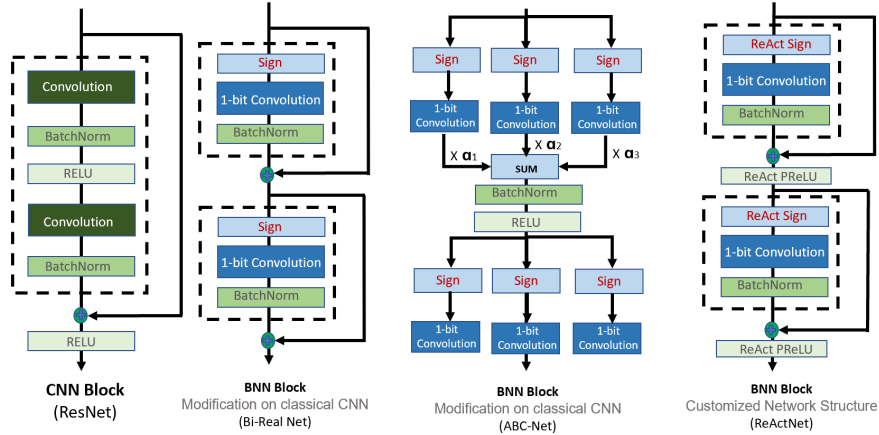
Figure 10. Shapes of sign or approx-sign functions and their derivatives

Table 5. Summary Table for Gradient Approximation

BNN Name	Code Available	Key Idea	Shape Number
GB-Net(2018)	✗	parametrized clipping functions(PCF), scaled binary activation function(SBAF)	3
BNN-RBNT(2018)	✗	SignSwish:a Gradient Approximation based on sigmoid function	4
Bi-Real-Net(2018)	✓ (O, UO)	Tight approximation to the derivative of sign function with respect to activations, magnitude-aware gradient with respect to weights	5
CCNN(2019)	✗	Long-tailed higher-order estimator	6
CBCN(2019a)	✓ (O)	Gaussian function as the approximation of the gradient	-
IR-Net(2020c)	✓ (O)	Error Decay Estimator: a training-aware Gradient Approximation function based on tanh function	7
SI-BNN(2020a)	✗	Trainable thresholds into backward propagation	8
RBNN(2020)	✓ (O)	Training-aware Gradient Approximation function based on sign function	9
BinaryDuo(2020a)	✓ (O)	Quantitatively estimate the gradient mismatch using cosine similarity applying CDG	-

Note: O: Official implementation, UO: Un-official implementation, Number: shape plot in Figure 10, - means none

accuracy performance, fewer ops and lighter model size. MeliusNet (Bethge *et al.*, 2020b) and ReActNet (Liu *et al.*, 2020c) designs new BNN architectures which can beat the accuracy rate of full-precision light-weight MobileNet with fewer OPs computation cost. Inspired by BN-free (Brock *et al.*, 2021), BNN-BN-free (Chen *et al.*, 2021) replaces the batch normalization (BatchNorm) with scaling factor and the ReActNet without BatchNorm still has competitive classification top-1 accuracy on the ImageNet dataset. FracBNN (Zhang *et al.*, 2021b) extends ReActNet’s topology, re-balances the blocks of networks and designs a two 1-bit activation scheme to improve feature learning. FracBNN has a competitive top-1 prediction result on the ImageNet dataset compared to full precision MobileNet-v2. BCNN (Redfern, Zhu, and Newquist, 2021) designs a customized structure for ImageNet classification with lower model size compared to MeliusNet and ReActNet.

**Figure 11.** Representative CNN and BNN block structure

3.5. Training Strategy and Tricks

Different training scheme and tricks can also affect final accuracy of BNN. SQ-BWN (Dong *et al.*, 2017) applied Stochastic Quantization (SQ) algorithm to gradually train and quantize BNN to compensate the quantization error. Bi-Real-Net (Liu *et al.*, 2018) initiated trainable parameters based on pre-train real value network and replaced RELU activation function with Htanh function. How to Train (Tang, Hua, and Wang, 2017) replaced the RELU activation function with PReLU function and explored that learning rate can have an influence on the accuracy of the final trained BNN. Empirical Study (Alizadeh *et al.*, 2018) explored the impact of pooling, optimizer and learning rate initialization for training BNN. Bop (Helweggen *et al.*, 2019)

Table 6. Summary Table for Network Topology Structure

BNN Name	Code Available	Key Idea	Idea Category
ABC-Net(2017)	✓ (UO)	Employ multiple binary activations and linear combination of multiple binary weight to alleviate single channel information loss	★, ►
WPRN(2017)	✗	Increase the number of filters to compensate for information loss	★, ◀
Bi-Real-Net(2018)	✓ (O, UO)	Shortcut for one layer per block	★, ▲
CBCN(2019a)	✓ (O)	Circulant filters(CiFs) and circulant binary convolution(CBConv) to enhance the capacity of binarized convolutional features	★, ◀
Group-Net(2019)	✓ (O)	Group binarization base block to approximate full precision network	★, ►
BENN(2019)	✓ (O)	Use ensemble variant bagging to aggregate multiple BNN results	★, ▼
BinaryDenseNet(2019)	✓ (O)	New BNN efficiency architecture: BinaryDenseNet	★, ▼
Search Accurate(2019)	✗	Evolutionary algorithm to search and adjust the number of channels in each convolutional layer after binarization	●, ■
DMS(2020)	✓ (O)	Differentiable dimension search algorithm to search and adjust the number of channels in each convolutional layer after binarization	●, ■
BATS(2020a)	✗	space specially BNN architectures search	●, ■
BNAS(2020)	✓ (O)	BNN architectures searches based on the cell based on search methods	●, ■
NASB(2020)	✗	Neural Architecture Search algorithm for optimal BNN architecture	●, ■
BBG-Net(2020)	✗	Reconstructing information flow with gated residual	★, ▲
Real-to-Bin(2020)	✓ (UO)	Data-driven channel re-scaling gated residual	★, ▲
MoBiNet-Mid(2020b)	✗	MoBiNet: a lightweight module binarization with the support of skip connection, the three block designs and the K-dependency	●, ◆
MeliusNet(2020b)	✓ (O)	DenseBlock: increases the feature capacity, Improvement Block: increases the feature quality	●, ◆
Binarized MobileNet (2020a)	✗	Evolutionary search to explore group structures when either using depth-wise or fully convolutional layers in MobileNet	●, ◆
High-Capacity-Expert(2020b)	✓ (O)	Condition computing(experts convolution) grouped convolution and NAS strategy	●, ◆
ReActNet(2020c)	✓ (O)	ReAct-Sign and ReAct-PReLU to reshape and shift the activation distributions	●, ◆
FracBNN(2021b)	✓ (O)	Design a dual-precision activation scheme to compute features	●, ◆
BCNN(2021)	✗	Design a network for ImageNet classification with lower model size	●, ◆
BNN-BN-free(2021)	✓ (O)	Replace batch normlization with scaling factors	●, ★, ▼

Note: O: Official implementation, UO: Un-official implementation, ★: Modification on classical CNN model, ●: Customized Network Structure, ▲: Increasing shortcut/residual gate, ◀: Increasing filters, ►: Increasing number of channels, ▼: other, ■: Search algorithms to optimize BNN, ◆: aims to beat real value light-weight network

and UniQ (Pham, Abraham, and Chung, 2021) separately proposed new optimizer for their BNN training. Main/Subsidiary (Xu *et al.*, 2019) proposed filter pruning for BNN. Also, inspired by The Lottery Ticket Hypothesis (Frankle and Carbin, 2018), MPT (Diffenderfer and Kailkhura, 2021) designed a scheme to learn highly accurate BNN simply by pruning and quantizing randomly weighted full precision CNN. CI-BCNN (Wang *et al.*, 2019c) simultaneously trained reinforcement graph model and BNN to alleviate binarization inconsistency, their training scheme is similar to RBNN (Lin *et al.*, 2020) which applied generative adversarial network (GAN) to train BNN. Real-to-Bin (Martinez *et al.*, 2020) designed a two-step training strategy that applied transfer teaching method to train BNN through learning real value pre-train network. Utilizing Real-to-Bin’s training strategy, some BNN works finally trained a high accuracy model such as ReActNet

(Liu *et al.*, 2020c), High-Capacity-Expert (Bulat, Martinez, and Tzimiropoulos, 2020b) and BCNN (Redfern, Zhu, and Newquist, 2021). Extend based on Real-to-Bin’s training strategy, BNN-Adam (Liu *et al.*, 2021) investigates and designs a new training scheme based on Adam optimizer and can successfully improve Real-to-Bin and ReActNet’s trained performance. BinaryDuo (Kim *et al.*, 2020a) proposed a two-stage training scheme to decouple a ternary-activation network into a two-binary-activation BNN network. MD-tanh-s (Ajanthan *et al.*, 2021) applied mirror descent to optimize BNN’ optimizer. Instead of training BNN on conventional hardware such as GPU and TPU, BNN-EP (Laydevant *et al.*, 2021) and BNN-Edge (Wang *et al.*, 2021a) explores to directly train BNN on the chip and Edge. BNN-EP (Laydevant *et al.*, 2021) proposes to use Equilibrium Propagation (EP) to train BNN and finds its possibility of training on-chip BNNs with compact circuitry. BNN-Edge (2021a) designs a low-memory and low-energy training scheme by modifying the forward propagation and back propagation including binarizing weight gradients, changing batch normalization layer and using low-precision floating-point data. BNN-stochastic (2021a) proposes a transfer training and initialization scheme for BNN using the stochastic relaxation approach and improves the accuracy on the small-scale CIFAR-10 dataset. Sub-bit Neural Networks (SNNs) (2021c) proposed a new method to further compress and accelerate BNN in FPGA based on the observation of the binary kernels in BNN. Table 7 is a summary table for BNN that proposed techniques for training strategy and tricks.

Table 7. Summary Table for Training Strategy and Tricks

BNN Name	Code Available	Key Idea	Idea Category
SQ-BWN(2017)	✓ (O)	Stochastic quantization(SQ) algorithm for training	●
Bi-Real-Net(2018)	✓ (O, UO)	Initialization: replace ReLU with clip(-1, x, 1) to pre-train the real-valued CNN model	●, ▲
How to Train(2017)	✗	Low learning rate better, use PReLU, scale layer, multiple activation	●
Empirical Study(2018)	✓ (O)	Identify the essential techniques required for optimisation of BNN	▲
Bop(2019)	✓ (O)	Bop: a Latent-Free Optimizer designed specifically for BNN	▲
Main/Subsidiary(2019)	✓ (O)	BNN filter-level pruning	▲
CI-BCNN(2019c)	✗	Train BNN and reinforcement graph model simultaneously to alleviate binarization inconsistency	●
RBNN(2020)	✗	Use generative adversarial network to train	●
Real-to-Bin(2020)	✓ (UO)	Two-step training strategy: spatial attention transfer computed from a teacher real-valued network to the binary network.	●
BinaryDuo(2020a)	✓ (O)	Decouple ternary activation to two binary activations	●
ReActNet(2020c)	✓ (O)	Adopt two-step training strategy from Real-to-Bin	●
High-Capacity-Expert(2020b)	✓ (O)	Adopt and improve two-step training strategy from Real-to-Bin	●
MD-tanh-s(2021)	✓ (O)	Apply mirror descent to BNN	▲
UniQ(2021)	✓ (O)	Special optimizer and warm-up strategy for binary training with symmetric quantizer	●,▲
BCNN(2021)	✗	Adopt two-step training strategy from Real-to-Bin	●
MPT(2021)	✓ (O)	Multi-Prize Lottery Ticket Hypothesis	▲
BNN-stochastic(2021b)	✗	Initialization and Transfer Learning stochastic BNN	
BNN-Edge(2021a)	✓ (O)	Low-memory and low-energy training	●
BNN-EP(2021)	✓ (O)	Equilibrium Propagation for training BNN	●
BNN-Adam(2021)	✓ (O)	Adam-based optimizers investigation	●,▲
SNNs(2021c)	✓ (O)	Further compress BNN	●,▲

Note: O: Official implementation, UO: Un-official implementation, ●: Train strategy, ▲: tricks/activations/optimizer/learning rate/pruning

3.6. Summary

In this section, we put BNN enhancement methods into five categories: (1) quantization error minimization, (2) loss function improvement, (3) gradient approximation, (4) network topology structure, and (5) training strategy and tricks. With the development of BNN optimization, we notice that just unitizing one

enhancement method is hard to improve BNN’s accuracy performance. We understand some problems are still unsolved.

How does each binarization layer affect the entire BNN performance? Understanding the degree of information loss from each binarization layer can promote the production of layer-wise optimization methods. Besides, OPs are becoming an equally important performance indicator as well as accuracy rate. To design BNN with high accuracy and lower operations per second (OPs) simultaneously becomes critical. How to effectively speed up BNN training time? Although BNN has faster inference speed and lighter weight size in resource-limited devices, training BNN still has to be done on conventional devices such as GPU and takes expansive computing cost and a long time. There are a few published works that can reduce memory and energy usage. But we still cannot find any breakthrough to significantly reduce BNN training time. How to effectively speed up BNN training time is still an open problem.

For a given BNN application, which training strategy and tricks should be used? There are many different published works that report their proposed training strategies and tricks that can improve trained models’ accuracy performance. However, all the benchmark results were tested based on the designed BNN structure and specific datasets such as CIFAR-10 or ImageNet. We are not sure if a similar improvement effect could be achieved in different datasets and BNN variants. It is necessary to do a survey research study to compare the difference among proposed training strategies and tricks.

4. Open Source Frameworks of Binary Neural Network

4.1. Open Source Frameworks of BNN Introductions

BNN has the ability to decrease the memory consumption and computational complexity. However, most published implementations of BNN do not really store their weight parameters in the binary format and cannot use XNOR and popcount to perform binary matrix multiplications in convolutions and fully connected layers. The reason is that deep learning models are directly implemented by python frameworks such as TensorFlow (Abadi *et al.*, 2016) and PyTorch (Paszke *et al.*, 2019), but python cannot store the data in binary form and does bit type data operations like C/C++ language. In the literature, there are several published available open-source BNN inference framework that can make the BNN’ models achieve the actual BNN performance. This section introduces and reviews the published BNN inference frameworks BMXNet (Yang *et al.*, 2017), BMXNet2 (Bethge *et al.*, 2020a), daBNN (Zhang *et al.*, 2019), Riptide (Fromm *et al.*, 2020), FINN (Blott *et al.*, 2018) and Larq (Bannink *et al.*, 2021).

BMXNet is an Apache-licensed open-source BNN library framework. It is written based on MXNet (Chen *et al.*, 2015) which is a high-performance and modular deep learning library. Depends on custom MXNet operators such as QActivation, QConvolution and QFullyConnected, BMXNet is able to support quantization and binarization of input data and weights. BMXNet can store the weights of convolutional and fully connected layers in their binarized format and perform matrix multiplication using bit-wise operations (XNOR and popcount). The BMXNet library, several sample code and a collection of pre-trained binary deep models are available at <https://github.com/hpi-xnor>

daBNN is a BSD-licensed open-source BNN inference framework highly optimized for ARM-based devices. daBNN designs an upgraded bit-packing scheme to pack multiple elements simultaneously, which reports that the speed of naive sequential method by about 4 times. In additions, daBNN proposes a new binary direct convolution to squeeze the cost of extra instructions in binary convolution, and creates a new novel memory layout to reduce memory access. daBNN is implemented in C++ and ARM assembly. Also, daBNN provides Java and Android package. Compared to BMXNet, daBNN is constructed based on standard ONNX (2019) operators (Sign and Convolution) to ensure interoperability. daBNN can convert PyTorch float-points models to BNN models. daBNN reports that their converted BNN model’s performance is 7-23 times faster on a single binary convolution than BMXNet. daBNN’s source code, sample projects and pre-trained models are available on-line: <https://github.com/JDAI-CV/dabnn>

BMXNet2 is an Apache-licensed open-source BNN framework, implemented based on BMXNet framework. Compared to the original BMXNet framework, BMXNet2 reuses more of the original MXNet operators and only adds three new functions in the C++ backend: the sign, round with STE and gradcancel operator. BMXNet2 can easily have minimal changes to C++ code to get better maintainability with future versions of MXNet. Besides, in the original BMXNet, the code for optimized inference was mixed with the training code. In BMXNet2, the two parts’ code is separately implemented, which can further simplify debugging and unit testing. The BMXNet2 source code and demos are available at <https://github.com/hpi-xnor/BMXNet-v2>

Riptide is a private licensed open-source BNN framework. It is built on the top of TensorFlow and TVM (Chen *et al.*, 2018), to service for BNN training and deployment. TVM is an open source deep learning compiler framework for diverse hardware environments including CPUs, GPUs, and deep learning accelerators. TVM can automatically deliver the optimized kernels for deep learning models on a special hardware platform. Depended on TVM, Riptide designs and develops new customs functions to support the optimized kernels generation for BNN. And Riptide proposes a new solution to completely remove floating-point arithmetic in the intermediate ‘glue’ layers(weight scaling, batch normalisation, and binary re-quantization) between pairs of binarized convolutions. Riptide reports their BNN models performance can achieve 4-12 times speed-up compared to a floating-point implementation. The Riptide source code and library are available at <https://github.com/jwfromm/Riptide>

FINN is a BSD-3-Clause License framework developed and maintained by Xilinx Research Labs. It services Xilinx’s series of FPGA boards. The framework can support model’s development, training, format conversion and embed in boards for both low-bit networks and 1-bit BNN. FINN has three components where are (1) brevitas: a PyTorch library for model develop and training; (2) FINN compiler: model format transformation and compiled; and (3) PYNQ: a python package to connect transformed model with Xilinx board. The FINN source code, library and demos are available at <https://github.com/Xilinx/FINN>

Larq is an Apache-licensed open-source BNN framework. It is built on the top of TensorFlow and TensorFlow Lite, and servicing for BNN model deployment, training and conversion. Larq contains two parts: Larq library and Larq Compute Engine (LCE). Larq library includes BNN quantization functions which extends TensorFlow. LCE contains a TensorFlow model graph converter and highly optimized implementations of binary operations and accelerations. Larq reports that it is the art-of-the-state fastest BNN framework over the existing inference frameworks. The Larq source code, library and demos are available at <https://github.com/larq/larq>

Table 8. BNN Frameworks Comparisons and Characterises

Framework	Model Format Base	Model Training	Maintenance
BMXNet	MXNet	MXNet	Until Nov 18, 2019
BMXNet2	MXNet and BMXNet	MXNet	Until Jul 2, 2020
daBNN	ONNX	PyTorch	Until Nov 11, 2019
Riptide	TensorFlow and TVM	TensorFlow	Until May 13, 2020
FINN	ONNX	Brevitas(PyTorch Modification)	Present -
Larq	TensorFlow and TensorFlow Lite	TensorFlow	Present -

4.2. Summary

When native BNN was published and taking full advantage of bits, users had to re-implement the model in a low-level programming language such as C/C++ to embed the BNN model on tiny devices such as FPGA or mobile phones. Such an engineering task is not convenient for un-professional software developers. With the rapid development of BNN open-source platforms, we can easily transform the trained BNN model to the related inference format on tiny devices without worrying about professional engineering works.

However, as table 8 shows, only two BNN frameworks, FINN and Larq, are actively maintained. BMXNet, daBNN, BMXNet2, and Riptide have stopped version updating. Unfortunately, Larq only supports TensorFlow based models, and FINN exclusively services Xilinx’s FPGA boards. There are no other options for developers who prefer to use different libraries such as PyTorch to design and develop BNN systems. How to create and maintain the cross-platform open-source framework that can efficiently import any library-based BNN models, like NCNN (Tencent, 2017), is a new exciting research problem and opportunity.

5. Limit-resource Hardware Architecture

Currently, in many real-world applications such as robotics, wearable devices and even self-driving car, recognition vision tasks need to be carried out in a timely fashion on a computationally limited platform. Instead of applying complex and heavy-weight neural network in expensive hardware such as graphics processing unit (GPU), it is a trend to use the resource constraint hardware to embed with efficiency trained models. In this

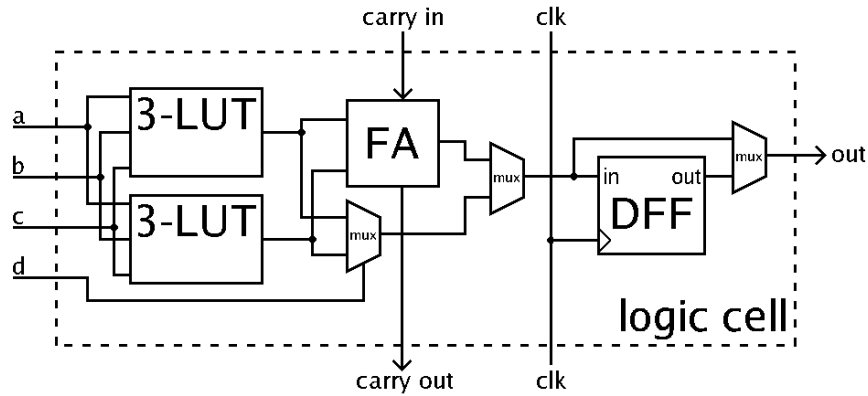


Figure 12. An example of FPGA logic cell (Drawed by Petter Kallstrom, public domain license)

section, limit-resource hardware is briefly introduced and discussed. Among the various kinds of efficiency devices, field programmable gate array (FPGA) is one of the most popular circuit architectures. Because the dominant computations on FPGA are bitwise logic operations and FPGA's memory requirements are greatly reduced, FPGA is well suited for BNN. Figure 12 is an example of an FPGA logic cell. To make this figure and the rest of this section easy to understand, we first define the terms and units used in this section.

FPGA: an abbreviation for field programmable gate array. FPGA is an integrated circuit that enable users to program for different algorithms after manufacturing.

LUT: an abbreviation for Look-Up Table. FPGA uses it to do boolean algebra such as XNOR, AND, OR, etc. The LUT can be programmed by the designer to execute a boolean algebra equation.

FA: an abbreviation for Full Adder. FA is central to most digital circuits that perform addition or subtraction.

DFF: an abbreviation for D flip-flop. DFF is also known as a "data" or "delay" flip-flop, which is used for the storage of state. One DFF keeps a single bit (binary digit) of data; one of its two states represents a "one" and the other represents a "zero".

MUX: an abbreviation for multiplexer, that selects a single input among input set to output.

BRAM: an abbreviation for Block Random Access Memory. BRAM is also known as "Block RAMs" Block RAMs are used for storing large amounts of data in FPGA.

clk: an abbreviation for clock. clk is a signal inside any digital circuit which presents how the performance a flip flop (or a group of flip flops) works. On the FPGA platform, the faster the clock, the faster the designed function will run.

DSP slices: (digital signal processing)DSP slices form the basis of a versatile, coarse grain DSP architecture in Xilinx FPGA, that can enable efficiently add powerful FPGA-based DSP functionality.

FPS: an abbreviation for Frames Per Second. FPS measures the frame rate that evaluates one trained model's inference speed.

ASIC: an abbreviation for Application Specific Integrated Circuit. Different from FPGA, it does not allow users to reprogram or modify after fabrication.

SBC: an abbreviation for Single-Board Computer. SBC is a complete computer built on a single circuit board such as Raspberry Pi. It contains microprocessor(s), memory, input/output(I/O) and other features required of a functional computer.

ARM-processor: stand for one kinds of CPU based on reduced instruction set computing(RISC) architectures. It is widely used in mobile machine and efficiency platform.

SoC: an abbreviation for System-on-a-Chip. SoC comprises of various functional units on a single silicon chip.

SoC FPGA: SoC FPGA devices integrate both processor and FPGA architectures into a single device such as Xilinx family's FPGA development board.



Figure 13. Representative SBC platforms

Figure 13 shows some representative resource-constrained SBC platforms based on FPGA and ARM processors. To testing if one neural network has a good efficiency performance on the resource-limited platform, there are common efficiency indicators including inference memory usage and inference speed such as FPS or latency time. In a FPGA platform, the number of measured DSP, LUT, BRAM and powers can also explain one trained model’s efficiency performance. In recent years, there are some works that report efficiency performance testing for BNN in above platforms. The performance results based on the special dataset are listed in the tables for reference in the next section.

6. Binary Neural Network Applications

6.1. Image Classification

Image classification is the basic benchmark testing task to evaluate BNN’ performance. The following tables show the summary of a variety of BNN’ performance on common benchmark image datasets including CIFAR-10 (Krizhevsky, 2009) and ILSVRC12 ImageNet (Russakovsky *et al.*, 2015). For each BNN model result, we contain the published years, BNN name, topology and accuracy. In particular for ImageNet results, we also contain the BOPs, FLOPs and OPs results(OPs = FLOPs + 1/64*BOPs (Bethge *et al.*, 2019; Liu *et al.*, 2018)) which are equally important to represent the BNN efficiency performance compared to accuracy results. For BNN, FLOPs is for the floating-point operations in the networks, excepting the operation calculated in BOPs which is bitwise. For full precision networks, OPs equals to FLOPs. To the best of our ability, we collect all the published reliable BNN’ performance results at the time of this paper submission.

6.1.1. CIFAR-10 Dataset

CIFAR-10 dataset consists of 60000 color images in 10 classes. Each class has 6000 images and each image is 32x32 pixels. There are 50000 training images and 10000 test images.

Table 9. BNN efficiency comparisons on CIFAR-10 using FPGA

BNN	Device	FPS	Acc(%)	Bits(W/A)	F_{max} (MHz)	Power(W)	DSP	BRAM	LUT
Acce-BNN(2017)	Zynq 7Z020	168.4	87.73 88.8 ¹	1/1	143	4.7	3	94	46900
FC-BNN(2017)	Zynq 7ZC020	420	81.8	1/1	143	2.3	1	32	14509
FO-BNN(2018)	Zynq 7ZC020	930	86	1/1	143	2.4	53	135	23436
FP-BNN (2018)	Stratix-V	7692.3	86.31	1/1	150	26.2	20	2210	219010
ReBNet(2018)	Zynq ZC702	2000	86.98	1/1	200	-	-	-	-
FBNA(2018)	Zynq ZC702	520.8	88.61	1/1	-	3.3	-	103	29600
FracBNN(2021b)	Zynq ZU3EG	2806.9	89.1	1/1.4 ²	250	4.1	126	212	51444

Note: ¹: support materials from its code GitHub page, ²: 1.4bit based on the analysis of quantized activations, convolution still executes 1W/1A operations

Table 10. BNN Accuracy comparisons on CIFAR-10

Full Precision Base CNN		
CNN Name		Acc(%)
VGG-Small(2018)		93.8
VGG-11(2019)		83.8
NIN(2019)		84.2
ResNet-18(2020d)		93.0
ResNet-20(2020d)		91.7
WRN-22(2016)		92.62
WRN-22(4 x Kernel Stage) ¹ (2016)		95.75
BNN Accuracy Performance		
BNN Name	Topology	Acc(%)
BNN(2016)	VGG-Small	87.13
XNOR-Net(2016)	VGG-Small	87.38
	WRN-22	81.90(2019a)
	WRN-22(4 x Kernel Stage) ¹	88.52(2019a)
	ResNet-18	90.21(2021)
LAB2(2016)	VGG-Small	87.72
DoReFa-Net(2016)	ResNet-20	79.3(2016)
HORQ(2017)	Customized	82.0(2016)
GB-Net(2018)	Customized	89.59
Bi-Real-Net(2018)	ResNet-18** ²	89.12(2021)
HadaNet(2019)	Customized($\beta w=4; \beta a=4$)	88.64
	NIN($\beta w=4; \beta a=4$)	87.33
	Customized($\beta w=16; \beta a=2$)	89.02
	NIN($\beta w=16; \beta a=2$)	88.74
PCNN(2019b)	WRN-22	89.17(J=1) ³
	WRN-22	91.27(J=2) ³
	WRN-22	92.79(J=4) ³
	WRN-22(4 x Kernel Stage) ¹	94.31(J=1) ³
	WRN-22(4 x Kernel Stage) ¹	95.39(J=4) ³
BONN(2019a)	WRN-22	87.34
	WRN-22(4 x Kernel Stage) ¹	92.36
CBCN(2019a)	Customized ResNet-18(4W, 4A) ⁴	90.22
RBNN(2020)	WRN-22(4 x Kernel Stage) ¹	93.28
BNN-DL(2019)	VGG-Small	89.90
	ResNet-18	90.47
CCNN(2019)	VGG-Small	92.3
CI-BCNN(2019c)	VGG-Small	92.47
	ResNet-20	91.10
Main/Subsidiary(2019)	NIN	83.11
	VGG-11	81.97
	ResNet-18	86.39
DSQ(2019)	VGG-Small	91.72
	ResNet-20	84.11
Search Accurate(2019)	Customized VGG	92.17
	Customized VGG	93.06
BBG-Net(2020)	ResNet-20	85.34
	ResNet-20(2W, 2A) ¹	90.71
	ResNet-20(4W, 4A) ¹	92.46
SI-BNN(2020a)	VGG-Small	90.2
IR-Net(2020c)	VGG-Small	90.4
	ResNet-18	91.5
	ResNet-20	85.4
	ResNet-20** ²	86.5
SLB(2020)	VGG-Small	92.0
	ResNet-20	85.5

Note: ¹: WRN-22 (Zagoruyko and Komodakis, 2016) original kernel-stage is 16-16-32-64, ²: ResNet**: Variant ResNet((Liu *et al.*, 2018), ³: J is total projection number, ⁴: number of Activation and Weights,

Table 11. Continue Table 11 BNN Accuracy comparisons on CIFAR-10

BNN Name	Topology	Acc(%)
RBNN(2020)	VGG-Small	91.3
	ResNet-18	92.2
	ResNet-20	86.5
	ResNet-20** ²	87.8
DMS(2020)	Customized VGG-11(DMS-A)	84.16
	Customized VGG-11(DMS-B)	89.10
	Customized ResNet-18(DMS-A)	89.32
	Customized ResNet-18(DMS-B)	92.70
BNAS(2020)	Customized ⁵ ResNet-18	92.70
	Customized ⁵ ResNet-34	93.76
BATS(2020a)	Customized	96.1
ReActNet(2020c)	ReActNet-A	82.95(2021)
	(Customized ⁵ MobileNet-v1)	
	Customized ResNet-20	85.8(2021b)
	ReActNet-18	92.31(2021)
FracBNN(2021b)	(Customized ResNet-18)	
	Customized ResNet-20	87.2
MPT(2021)	VGG-Small(75% weights pruned)	88.52
	VGG-Small+BN ⁶ (75% weights pruned)	91.9
BNN-BN-free(2021)	XNOR(Based on ResNet-18)	79.67
	Bi-Real-Net(Based on ResNet-18** ²)	79.59
	ReActNet-18	92.08
	(Customized ResNet-18)	
	ReActNet-A	83.91
ReCU (Xu <i>et al.</i> , 2021b)	(Customized MobileNet-v1)	
	VGG-Small	92.2
	ResNet-18	92.8
	ResNet-20	87.4

Note: ²: ResNet**: Variant ResNet((Liu *et al.*, 2018), ⁵: OPs similar, ⁶: BN: BatchNorm

6.1.2. ImageNet Dataset

ImageNet is a large images dataset that is usually used to test the trained model's performance. There are different versions of ImageNet dataset. The common version used for BNN is ILSVRC2012 ImageNet which was used for the competition dataset of "ImageNet Large Scale Visual Recognition Challenge 2012". The ILSVRC2012 ImageNet consists of three subset parts; training, validation and test dataset. Training dataset has more than 1.2 million color images in about 1000 classes. Validation dataset has 50000 color images and test dataset contains 100000 color images.

Table 12. BNN efficiency comparisons on ImageNet using FPGA

BNN	Device	FPS	Top-1 Acc(%)	FPGA Platform		F_{max} (MHz)	Power(W)	DSP	BRAM	LUT
				Top-5 Acc(%)	Bits (W/A)					
Oc-BNN(2017)	Zynq ZU9EG	31.48	-	-	1/1	150	22	4	1367	-
ReBNet(2018)	Virtex VCU108	170	41.43	-	1/1	200	-	-	-	-
FP-BNN(2018)	Stratix-V	862.1	42.9	66.8	1/1	150	26.2	384	2210	230918
FracBNN(2021b)	Zynq ZU3EG	48.1	71.8	90.1	1/1.4 ²	250	6.1	224	201	50656

Note: ¹: support materials from its code GitHub page, ²: 1.4bit based on the analysis of quantized activations, convolution still executes 1W/1A operations, ³: without bit-shift scales

Table 13. BNN performance comparisons on ImageNet

Full Precision Base CNN						
CNN Name		Top-1 Acc (%)	Top-5 Acc (%)		FLOPs/OPs (x10 ⁸)	
AlexNet(2018)		57.1	80.2		-	
ResNet-18(2018)		69.6	89.2		18.1(2018; 2020b)	
ResNet-34(2018)		73.3	91.3		36.6(2018; 2020b)	
ResNet-50(2018)		76.0	93.0		38.6(2020)	
BN-Inception(2017)		71.64	-		-	
MobileNet-v1 0.5(2017)		63.7	-		1.49(2020b)	
MobileNet-v1 0.75(2017)		68.4	-		3.25(2020b)	
MobileNet-v1 1.0(2017)		70.6	-		5.69(2020b)	
MobileNet-v2(2018)		71.53	-		-	
BNN Accuracy Performance						
BNN Name	Topology	Top-1 Acc (%)	Top-5 Acc (%)	BOPs (x10 ⁹)	FLOPs (x10 ⁸)	OPs (x10 ⁸)
BNN(2016)	AlexNet	27.9	50.42	1.70	1.20	1.47
	ResNet-18	42.2 (2020)	69.2 (2020)	1.70	1.31	1.67
XNOR-Net(2016)	AlexNet	44.2	69.2	-	-	-
	ResNet-18	51.2	73.2	1.70	1.33	1.60
	ResNet-34	56.49 (2020)	79.13 (2020)	-	-	1.78
DoReFa-Net(2016)	AlexNet	40.1	-	-	-	-
	AlexNet	43.6(initialized)	-	-	-	-
ABC-Net(2017)	ResNet-18	42.7	67.6	-	-	1.48
	ResNet-18(3W, 1A) ¹	49.1	73.8	-	-	-
	ResNet-18(3W, 3A) ¹	61.0	83.2	-	-	-
	ResNet-18(3W, 5A) ¹	63.1	84.8	-	-	-
	ResNet-18(5W, 1A) ¹	54.1	78.1	-	-	-
	ResNet-18(5W, 3A) ¹	62.5	84.2	-	-	5.20
	ResNet-18(5W, 5A) ¹	65.0	85.9	-	-	7.85
	ResNet-34	52.4	76.5	-	-	-
	ResNet-34(3W, 3A) ¹	66.7	87.4	-	-	-
	ResNet-34(5W, 5A) ¹	68.4	88.2	-	-	-
	ResNet-50(5W, 5A) ¹	70.1	89.7	-	-	-
WRPN(2017)	AlexNet(2x wide) ²	48.3	-			
	ResNet-34	60.54	-	-	-	-
	ResNet-34(2x wide) ²	69.85	-	-	-	-
	ResNet-34(3x wide) ²	72.38	-	-	-	-
	BN-Inception(2x wide) ²	65.02	-	-	-	-
SQ-BWN(2017)	AlexNet	45.5 (2019)	70.6(2019)	-	-	-
BNN-RBNT(2018)	AlexNet	46.1	75.7	-	-	-
	ResNet-18	53.01	72.98	-	-	-
Bi-Real-Net(2018)	ResNet-18** ⁹	56.4	79.5	1.68	1.39	1.63
	ResNet-18** ⁹ (2020b)	60.6	-			1.14
	ResNet-34** ⁹	62.2	83.9	3.53	1.39	1.93
	ResNet-34** ⁹ (2020b)	63.7	-			1.43
	ResNet-50** ⁹ (2019)	62.6	83.9			
	ResNet-152** ⁹ (2020b)	64.5	-	10.7	4.48	6.15
	MobileNet-v1 1.0(2019)	58.2	-	-	-	-
PCNN(2019b)	ResNet-18** ⁹	57.3	80.0	-	-	1.63
HadaNet(2019)	AlexNet($\beta w=4; \beta a=4$)	46.3	71.2	-	-	-
	AlexNet($\beta w=16; \beta a=2$)	47.3	73.3	-	-	-
	ResNet-18($\beta w=4; \beta a=4$)	53.3	77.3	-	-	-
	ResNet-18($\beta w=16; \beta a=2$)	53.8	77.2	-	-	-
XNOR-Net++(2019)	AlexNet(α, β, γ) ¹	46.9	71.0	-	-	-
	ResNet-18	55.5	78.5	-	-	-
	ResNet-18(α_1) ⁴	56.1	79.0	-	-	-
	ResNet-18(α_2, β_1) ⁵	56.7	79.5	-	-	-
	ResNet-18(α, β, γ) ³	57.1	79.9	1.695	1.333	1.60

Note: ¹: Nums W: number of Weights parallel, Nums A: number of Activations parallel, ²: Nums x wide: number of filters, ⁹: ResNet**: Variant ResNet(Liu *et al.*, 2018)

Table 14. Continue Table 13.

BNN Name	Topology	Top-1 Acc (%)	Top-5 Acc (%)	BOPs (x10 ⁹)	FLOPs (x10 ⁸)	OPs (x10 ⁸)
Bop(2019)	AlexNet(BNN(2016))	41.1	65.4	-	-	-
	AlexNet(XNOR-Net(2016))	45.9	70.0	-	-	-
	ResNet-18** ⁹	56.6	79.4	-	-	1.63
BNN-DL(2019)	AlexNet(BNN(2016))	41.3	65.8	-	-	-
	AlexNet(XNOR-Net(2016))	47.8	71.5	-	-	-
	AlexNet(DoReFa-Net(2016))	47.8	71.5	-	-	-
	AlexNet(Compact-Net(2017) ⁶)	47.6	71.9	-	-	-
	AlexNet(WRPN(2017))	53.8	77.0	-	-	-
Main/Subsidiary (2019)	ResNet-18(78.6% filters)	50.13	-	-	-	-
CCNN(2019)	AlexNet	46.13	70.9	-	-	-
	ResNet-18	54.2	77.9	-	-	-
Quantization Networks (2019)	AlexNet	47.9	72.5	-	-	-
	ResNet-18	53.6	75.3	-	-	1.63
CI-BCNN(2019c)	ResNet-18	56.73	80.12	-	-	1.54
	ResNet-18** ⁹	59.90	84.18	-	-	>1.54
	ResNet-34	62.41	84.35	-	-	1.82
	ResNet-34** ⁹	64.93	86.61	-	-	>1.82
BONN(2019a)	ResNet-18	59.3	81.6	-	-	-
CBCN(2019a)	ResNet-18(4W,4A) ¹	61.4	82.8	-	-	6.56
RBCN(2019b)	ResNet-18	59.5	81.6	-	-	-
Search Accurate (2019)	Customized ResNet-18	68.64	88.46	-	-	4.95
	Customized ResNet-18	69.65	89.08	-	-	6.60
Group-Net(2019)	ResNet-18(4W,4A) ¹	64.2	85.6	-	-	-
	ResNet-18(5W,5A) ¹	64.8	85.7	-	-	-
	ResNet-18(8W,8A) ¹	67.5	88.0	-	-	-
	ResNet-18** ⁹ (4W,4A) ¹	66.3	86.6	-	-	-
	ResNet-18** ⁹ (5W,5A) ¹	67.0(2018)	87.5(2018)	-	-	2.68
	ResNet-34(5W,5A) ¹	68.5	88.0	-	-	-
	ResNet-34(8W,8A) ¹	71.8	90.4	-	-	-
	ResNet-34** ⁹ (5W,5A) ¹	70.5(2018)	89.3((2018)	-	-	4.13
	ResNet-50(5W,5A) ¹	69.5	89.2	-	-	-
	ResNet-50(8W,8A) ¹	72.8	90.5	-	-	-
BinaryDenseNet (2019)	Customized DenseNet28	60.7	82.4	-	-	2.58
	Customized DenseNet28 (2020b)	62.6	-	-	-	2.09
	Customized DenseNet37	62.5	83.9	-	-	2.71
	Customized DenseNet37 (dilated)	63.7	84.7	-	-	-
	Customized DenseNet37 (2020b)	64.2	-	-	-	2.20
BENN(2019)	AlexNet(3W,3A) ¹²	48.8	-	-	-	-
	(bagging(1996))	50.2	-	-	-	-
	AlexNet(3W,3A) ¹²	(boosting(1997))	-	-	-	-
	AlexNet(6W,6A) ¹²	52.0	-	-	-	-
	(bagging(1996))	54.3	-	-	-	-
	AlexNet(6W,6A) ¹²	(boosting(1997))	-	-	-	-
	ResNet-18(3W,3A) ¹²	53.3	-	-	-	-
	(bagging(1996))					

Note: ¹: Nums W: number of Weights parallel, Nums A: number of Activations parallel, ³: α, β, γ statistically learned via channels, heights, weights, ⁴: α_1 a dense scaling, one value for each output pixel, ⁵: α_2 learns the statistics over the output channel dimension, β_1 learns it over the spatial dimensions, ⁶: Compact-Net(Tang, Hua, and Wang, 2017) uses 2 bits for activations while BNN-DL (Ding *et al.*, 2019) only uses 1 bit, ⁹: ResNet**: Variant ResNet(Liu *et al.*, 2018), ¹²: BNN networks bagging,

Table 15. Continue Table 13.

BNN Name	Topology	Top-1 Acc (%)	Top-5 Acc (%)	BOPs (x10 ⁹)	FLOPs (x10 ⁸)	OPs (x10 ⁸)
BENN(2019)	ResNet-18(3W,3A) ¹²	53.6 (boosting(1997))	-	-	-	-
	ResNet-18(6W,6A) ¹²	57.9 (bagging(1996))	-	-	-	-
	ResNet-18(6W,6A) ¹²	61.0 (boosting(1997))	-	-	-	-
IR-Net(2020c)	ResNet-18** ⁹	58.1	80.0	-	-	1.63
	ResNet-34** ⁹	62.9	84.1	-	-	-
BATS(2020a)	Customized	60.4	83.0	1.149	0.805	0.985
	Customized(2x-wider)	66.1	87.0	2.157	1.210	1.547
BNAS(2020)	Customized ¹³ ResNet-18 (XNOR-Net)	57.69	79.89	-	-	1.48
	Customized ¹³ ResNet-18** ⁹	58.76	80.61	-	-	1.63
	Customized ¹³ ResNet-34 (XNOR-Net)	58.99	80.85	-	-	1.78
	Customized ¹³ ResNet-34** ⁹	59.81	81.61	-	-	1.93
	Customized ¹³ ResNet-18 (4W,4A) ¹ (CBCN)	63.51	83.91	-	-	6.56
NASB(2020)	Customized ¹³ ResNet-18	60.5	82.2	-	-	1.71
	Customized ¹³ ResNet-34	64.0	84.7	-	-	2.01
	Customized ¹³ ResNet-50	65.7	85.8	-	-	6.18
Si-BNN(2020a)	AlexNet	50.5	74.6	-	-	-
	ResNet-18	58.9	81.3	-	-	-
	ResNet-18** ⁹	59.7	81.8	-	-	-
	ResNet-34	63.3	84.4	-	-	-
LNS(2020)	AlexNet ⁸	44.4	-	-	-	-
	ResNet-18	59.4	81.7	-	-	1.63
SLB(2020)	ResNet-18(w/o SBN ⁷)	61.0	82.9	-	-	-
	ResNet-18	61.3	83.1	-	-	-
Real-to-Bin(2020)	ResNet-18** ⁹ (baseline)	60.9	83.0	1.68	1.54	1.63
	ResNet-18** ⁹	63.2	84.0	1.68	1.54	1.80
	(BNN-Adam(2021))					
ProxyBNN(2020)	ResNet-18** ⁹	65.4	86.2	1.68	1.56	1.83
	AlexNet	51.4	75.5	-	-	-
	ResNet-18	58.7	81.2	-	-	-
	ResNet-18** ⁹	63.7	84.8	-	-	-
RBNN(2020)	ResNet-34** ⁹	66.3	86.5	-	-	-
	ResNet-18	59.9	81.9	-	-	-
	ResNet-34	63.1	84.4	-	-	-
BinaryDuo(2020a)	AlexNet	52.7	76.0	-	-	1.19
	ResNet-18	60.4	82.3	-	-	1.64
	ResNet-18** ⁹	60.9	82.6	-	-	1.64
MoBiNet-Mid(2020b)	Customized ¹⁰ (K=3)	53.47	76.46	-	-	0.49
	Customized ¹⁰ (K=4)	54.4	77.5	-	-	0.52
MeliusNet(2020b)	MeliusNetA	63.4	84.2	4.85	0.86	1.62
	(Customized ¹¹ ResNet-18** ⁹)					
	MeliusNetB	65.7	85.9	5.72	1.06	1.96
	(Customized ¹¹ ResNet-34** ⁹)					
	MeliusNetC	64.1	85.0	4.35	0.82	1.50
	(Customized ¹¹ MobileNetv1 0.5)					
	MeliusNet42	69.2	88.3	9.69	1.74	3.25
	(Customized ¹¹ MobileNetv1 0.75)					
	MeliusNet59	71.0	89.7	18.3	2.45	5.25
	(Customized ¹¹ MobileNet-v1 1.0)					

Note: ¹: Nums W: number of Weights parallel, Nums A: number of Activations parallel, ⁷: SBN: State Batch Normalization(Yang *et al.*, 2020), ⁸: with only layer-wise scale factor to compare with XNOR-Net, ⁹: ResNet**: Variant ResNet(Liu *et al.*, 2018), ¹⁰: K-layer dependency to improve single-layer dependency in depth-wise convolution, ¹¹: OPs and Size similar, ¹²: BNN networks bagging, ¹³:OPs similar

Table 16. Continue Table 13.

BNN Name	Topology	Top-1 Acc (%)	Top-5 Acc (%)	BOPs (x10 ⁹)	FLOPs (x10 ⁸)	OPs (x10 ⁸)
MeliusNet(2020b)	MeliusNet22 (Customized ¹¹ BDenseNet28(2019))	63.6	84.7	4.62	1.35	2.08
	MeliusNet29 (Customized ¹¹ BDenseNet37(2019))	65.8	86.2	5.47	1.29	2.14
Binarized MobileNet (2020a)	Customized MobileNet-v3	51.06	74.18	-	-	0.33
	Customized MobileNet-v2	59.30	81.00	-	-	0.62
	Customized MobileNet-v1	60.90	82.60	-	-	1.54
DMS(2020)	Customized(DMS-A) ¹⁵	60.20	82.94	-	-	-
	Customized(DMS-B) ¹⁵	67.93	87.84	-	-	-
High-Capacity-Expert (2020b)	Customized(4 experts)	67.5	87.5	1.7	1.1	1.37
	Customized(4 experts) ¹⁸	70.0	89.2	1.7	1.1	1.37
	Customized(4 experts) ¹⁹	71.2	90.1	1.7	1.1	1.37
ReActNet(2020c)	ReActNet-18 (Customized ResNet-18)	65.5(2021)	-	-	-	-
	ReActNet-A (Customized MobileNet-v1)	69.4	88.6(2021)	4.82	0.12	0.87
	ReActNet-A(BNN-Adam(2021)) (Customized MobileNet-v1)	70.5	89.1	4.82	0.12	0.87
	ReActNet-B (Customized MobileNet-v1)	70.1	-	4.69	0.44	1.63
	ReActNet-C (Customized MobileNet-v1)	71.4	-	4.69	1.40	2.14
MD-tanh-s(2021)	ResNet-18** ⁹	60.3	82.3	-	-	-
	ResNet-18** ⁹ (dilated)	62.8	84.3	-	-	-
UniQ(2021)	ResNet-18	60.5	-	-	-	-
	ResNet-34	65.8	-	-	-	-
	MobileNet-v2	23.2	-	-	-	-
IA-BNN(2020b)	AlexNet(BNN(2016))	42.1	66.6	-	-	-
	AlexNet(XNOR-Net(2016))	45.6	69.6	-	-	-
	ResNet-18(XNOR-Net(2016))	54.2	77.6	-	-	-
	ResNet-18** ⁹	57.2	80.2	-	-	-
	ResNet-34** ⁹	62.8	84.5	-	-	-
FracBNN(2021b)	Customized(1W/1.4A) ¹⁵	71.8	90.1	7.30	-	-
Group-Net Extend (2019; 2021)	ResNet-18** ⁹ (4W,4A) ¹	68.2	88.3	-	-	-
	ResNet-34** ⁹ (4W,4A) ¹	72.2	90.5	-	-	-
	ResNet-50** ⁹ (4W,4A) ¹	73.4	91.0	-	-	-
	MobileNet-v1 1.0	70.8	-	-	-	-
BCNN (2021)	Customized(P=1) ¹⁶	69.0	-	2.41	-	1.31
	Customized(P=2) ¹⁶	71.2	-	4.83	-	2.08
MPT(2021)	WRN-34(60% weights pruned)	45.06	-	-	-	-
	WRN-34+BN ¹⁷ (60% weights pruned)	52.07	-	-	-	-
BNN-BN-free(2021)	ReActNet-18	61.1	-	-	-	-
	(Customized ResNet-18)					
	ReActNet-A (Customized MobileNet-v1)	68.0	-	-	-	-
ReCU (2021b)	ResNet-18	61.0	82.6	-	-	-
	ReActNet-18	66.4	86.5	-	-	-
	ResNet-34	65.1	85.8	-	-	-
DA-BNN(2021)	ResNet-18** ⁹	63.1	84.3	-	1.69	-
	ReActNet-18	66.3	86.7	-	1.69	-

Note: ¹: Nums W: number of Weights parallel, Nums A: number of Activations parallel, ⁹: ResNet**: Variant ResNet(Liu *et al.*, 2018), ¹⁰: K-layer dependency to improve single-layer dependency in depth-wise convolution, ¹¹: OPs and Size similar, ¹⁵: 1.4bit based on the analysis of quantized activations, ¹⁶: number of parallel branches, ¹⁷: BN:BatchNorm ¹⁸: training strategy from Real-to-Bin(Martinez *et al.*, 2020), ¹⁹: Improved training strategy from Real-to-Bin(Martinez *et al.*, 2020)

6.2. Point Cloud Classification

Different from 2D image classification tasks, 3D tasks using BNN are much more challenging as binarization will amplify information loss during aggregating point-wise features in pooling layers and lead to huge distortion at the point-wise feature extraction stage. BiPointNet(Qin *et al.*, 2020b) proposed first binarization method for learning on 3D point cloud. Table 17 and table 18 list reported comparison results from (Qin *et al.*, 2020b). The benchmark dataset for testing accuracy used ModelNet40(Wu *et al.*, 2015), which contains 12311 pre-aligned shapes from 40 categories.

Table 17. 3D BNN efficiency comparisons using Arm-based platform

Time Cost			
Method	Device	Bits(W/A)	Time(ms)
PointNet(Qi <i>et al.</i> , 2017b)	Raspberry Pi 3B(ARM Cortex-A53)	32/32	131.8
BiPointNet(Qin <i>et al.</i> , 2020b)	Raspberry Pi 3B(ARM Cortex-A53)	1/1	9
PointNet(Qi <i>et al.</i> , 2017b)	Raspberry Pi 4B(ARM Cortex-A72)	32/32	67.3
BiPointNet(Qin <i>et al.</i> , 2020b)	Raspberry Pi 4B(ARM Cortex-A72)	1/1	5.5
Storage Usage			
Method	Device	Bits(W/A)	Storage(MB)
PointNet(Qi <i>et al.</i> , 2017b)	Raspberry Pi 4B(ARM Cortex-A72)	32/32	3.16
BiPointNet(Qin <i>et al.</i> , 2020b)	Raspberry Pi 4B(ARM Cortex-A72)	1/1	0.17

Note: ¹: support materials from its code GitHub page, ²: 1.4bit based on the analysis of quantized activations, convolution still executes 1W/1A operations,³: without bit-shift scales

Table 18. Accuracy Comparison for Points Cloud Classification

Method/(W/A)	Based Model	Aggregation	OA(%)
Full precision(32/32)	PointNet(Vanilla)(Qi <i>et al.</i> , 2017b)	Max	86.8
	PointNet(Qi <i>et al.</i> , 2017b)	Avg	86.5
		Max	88.2
	PointNet++(Qi <i>et al.</i> , 2017a)	Max	90.0
	PointCNN(Li <i>et al.</i> , 2018b)	Avg	90.0
	DGCNN(Wang <i>et al.</i> , 2019b)	Max	89.2
	PointConv(Wu, Qi, and Fuxin, 2019)	-	90.8
BNN(Courbariaux <i>et al.</i> , 2016)(1/1)	PointNet(Qi <i>et al.</i> , 2017b)	Max	7.1
		EMA-max	16.2
IR-Net(Qin <i>et al.</i> , 2020c)(1/1)	PointNet(Qi <i>et al.</i> , 2017b)	Max	7.3
		EMA-max	63.5
Bi-Real-Net(Liu <i>et al.</i> , 2018)(1/1)	PointNet(Qi <i>et al.</i> , 2017b)	Max	4.0
		EMA-max	77.5
ABC-Net(Lin, Zhao, and Pan, 2017)(1/1)	PointNet(Qi <i>et al.</i> , 2017b)	Max	4.1
		EMA-max	77.8
XNOR-Net++(Bulat and Tzimiropoulos, 2019)(1/1)	PointNet(Qi <i>et al.</i> , 2017b)	Max	4.1
		EMA-max	78.4
XNOR-Net(Rastegari <i>et al.</i> , 2016)(1/1)	PointNet(Vanilla)(Qi <i>et al.</i> , 2017b)	Max	61.0
	PointNet(Qi <i>et al.</i> , 2017b)	Max	64.9
		EMA-max	81.9
	PointNet++(Qi <i>et al.</i> , 2017a)	Max	63.1
	PointCNN(Li <i>et al.</i> , 2018b)	Avg	83.0
	DGCNN(Wang <i>et al.</i> , 2019b)	Max	51.5
	PointConv(Wu, Qi, and Fuxin, 2019)	-	83.1
BiPointNet(Qin <i>et al.</i> , 2020b)(1/1)	PointNet(Vanilla)(Qi <i>et al.</i> , 2017b)	EMA-max	85.6
	PointNet(Qi <i>et al.</i> , 2017b)	Max	4.1
		EMA-max	86.4
	PointNet++(Qi <i>et al.</i> , 2017a)	EMA-max	87.8
	PointCNN(Li <i>et al.</i> , 2018b)	EMA-avg	83.8
	DGCNN(Wang <i>et al.</i> , 2019b)	EMA-max	83.4
	PointConv(Wu, Qi, and Fuxin, 2019)	-	87.9

Note: EMA-avg, EMA-max was proposed in BiPointNet(Qin *et al.*, 2020b), OA: overall accuracy

6.3. Object Detection

Object detection is a more complex and difficult task than 2D image classification. Recently, there are a few published BNN works on objects detection. Sun *et al.* (2018) propose a fast object detection based on BNN (2016). Bethge *et al.* (2019) apply their proposed BinaryDenseNet to object detection, which has a comparable accuracy performance compared to full-precision Yolo. ASDA-FRCNN(Xu *et al.*, 2020) applies the experience from Modulated-Convolutional-Networks(Wang *et al.*, 2018) to represent full-precision kernel with BNN-based amplitude and direction, and designs a new loss function to reconstruct the full-precision kernels. Wang *et al.* (2020b) propose BiDet which employs the information bottleneck (IB) principle to remove redundancy information for taking full advantage of BNN and concentrate posteriors on informative detection prediction via learning the sparse object before. Zhao *et al.* (2021) apply their proposed re-scaling BNN method DA-BNN to Object Detection. Xu *et al.* (2021a) propose LWS-Det which is a training scheme under a student-teacher framework including (1) layer-wise minimizing angular loss by a differentiable binarization search method and (2) layer-wise minimizing amplitude error by learning scale factors. PASCAL-VOC (PASCAL Visual Object Classes) (Everingham *et al.*, 2010) and MS-COCO (Microsoft Common Objects in Context) (Lin *et al.*, 2014) are popular benchmark datasets to evaluate trained models' performance for object detection. Wang *et al.* (2021b) develop the block scaling factor XNOR (BSF-XNOR) convolutional layer and enhanced the accuracy on VisDrone2019 dataset (Zhu *et al.*, 2018) compared to XNOR-Net (Rastegari *et al.*, 2016)

PASCAL-VOC is a collection of datasets for object detection, which was used as the competition dataset in the PASCAL Visual Object Classes Challenge Competition. There are two commonly versions: VOC2007 and VOC2012. VOC2007 (Everingham *et al.*) dataset has 5011 training/validation (trainval) images and 4952 test images from 20 categories. VOC2012 (Everingham *et al.*) dataset has 11540 trainval images including 20 categories. VOC2012 is usually used as an additional data resource for model training.

MS-COCO is a large-scale object detection, segmentation, key-point detection, and captioning dataset which consists of 80 categories of images. The commonly used version of MS-COCO is MS-COCO2014. Table 20 is the collection of benchmark results in BNN and full-precision CNN, in where models were trained on 80000 images from the training set and 35000 images sampled from validation set (MS-COCO trainval35k)ASDA-FRCNN (Xu *et al.*, 2020) applies the experience from Modulated-Convolutional-Networks (Wang *et al.*, 2018) to represent full-precision kernel with BNN-based amplitude and direction, and designs a new loss function to reconstruct the full-precision kernels. Wang *et al.* (2020b) propose BiDet which employs the information bottleneck (IB) principle to remove redundancy information

Table 19 and table 20 are the latest BNN benchmark results on the VOC2007 test dataset. When we summarize this table, we notice (Qin *et al.*, 2020a) wrongly cited FQNTable 19 and table 20 are the latest BNN benchmark results on the VOC2007 test dataset. When we summarize this table, we notice (Qin *et al.*, 2020a) wrongly cited FQN (Li *et al.*, 2019a) as a BNN variant and collected its results on MS-COCO2017(Caesar, Uijlings, and Ferrari, 2018). FQN (Li *et al.*, 2019a) results were actually collected from a 4-bit quantization neural network. Table 21 and table 22 are the late

Table 19. Benchmark results on VOC2007 test dataset

Framework	BackBone	Method	(W/A)	Trained data	mAP(%)
Faster-RCNN(2015)	VGG-16	Full Precision	(32/32)	VOC2007	68.9(2018)
	VGG-16	BNN(2016)	(1/1)	VOC2007	47.3(2018)
	AlexNet	Full Precision	(32/32)	VOC2007	66.0(2018)
	AlexNet	BNN(2016)	(1/1)	VOC2007	46.4(2018)
	ResNet-18	Full Precision	(32/32)	VOC2007	67.8(2020)
	ResNet-18	Full Precision	(32/32)	VOC2007+2012	73.2(2020)
	ResNet-18	Full Precision	(32/32)	VOC2007+2012	74.5(2020b)
	ResNet-18	Full Precision	(32/32)	VOC2007+2012	76.4(2021a)
	ResNet-18	BNN(2016)	(1/1)	VOC2007+2012	35.6(2020b)
	ResNet-18	XNOR-Net(2016)	(1/1)	VOC2007+2012	48.4(2020b)
	ResNet-18	Bi-Real(2018)	(1/1)	VOC2007	51.0(2020)
	ResNet-18	Bi-Real(2018)	(1/1)	VOC2007+2012	58.2(2020b)

Note: mAP: Mean Average, ¹: BiDet(SC) means the proposed method with extra shortcut for the architectures Precision

Table 20. Continue Table 19: Benchmark results on VOC2007 test dataset.

Framework	BackBone	Method	(W/A)	Trained data	mAP(%)
Faster-RCNN(2015)	ResNet-18	Bi-Real(2018)	(1/1)	VOC2007+2012	60.6(2020)
	ResNet-18	Bi-Real(2018)	(1/1)	VOC2007+2012	60.9(2021a)
	ResNet-18	ASDA-FRCNN(2020)	(1/1)	VOC2007	54.6(2020)
	ResNet-18	ASDA-FRCNN(2020)	(1/1)	VOC2007+2012	63.4(2020)
	ResNet-18	BiDet(2020b)	(1/1)	VOC2007+2012	50.0(2020b)
	ResNet-18	BiDet(SC) ¹ (2020b)	(1/1)	VOC2007+2012	59.5(2020b)
	ResNet-18	BiDet(2020b)	(1/1)	VOC2007+2012	62.7(2021a)
	ResNet-18	DA-BNN(2021)	(1/1)	VOC2007+2012	63.5(2021)
	ResNet-18	ReActNet(2020c)	(1/1)	VOC2007+2012	69.6(2021a)
	ResNet-18	LWS-Det(2021a)	(1/1)	VOC2007+2012	73.2(2021a)
	ResNet-34	Full Precision	(32/32)	VOC2007+2012	73.2(2020)
	ResNet-34	Full Precision	(32/32)	VOC2007+2012	77.8(2021a)
	ResNet-34	Bi-Real(2018)	(1/1)	VOC2007+2012	63.1(2021a)
	ResNet-34	ASDA-FRCNN(2020)	(1/1)	VOC2007+2012	65.5(2020)
	ResNet-34	BiDet(2020b)	(1/1)	VOC2007+2012	65.8(2021a)
	ResNet-34	ReActNet(2020c)	(1/1)	VOC2007+2012	72.3(2021a)
	ResNet-34	LWS-Det(2021a)	(1/1)	VOC2007+2012	75.8(2021a)
	ResNet-50	Full Precision	(32/32)	VOC2007+2012	79.5(2021a)
	ResNet-50	Bi-Real(2018)	(1/1)	VOC2007+2012	65.7(2021a)
	ResNet-50	ReActNet(2020c)	(1/1)	VOC2007+2012	73.1(2021a)
	ResNet-50	LWS-Det(2021a)	(1/1)	VOC2007+2012	76.9(2021a)
SSD300(2016)	VGG-16	Full Precision	(32/32)	VOC2007+2012	72.4(2020b)
	VGG-16	Full Precision	(32/32)	VOC2007+2012	74.3(2021a)
	VGG-16	BNN(2016)	(1/1)	VOC2007+2012	42.0(2020b)
	VGG-16	XNOR-Net(2016)	(1/1)	VOC2007+2012	50.2(2020b)
	VGG-16	Bi-Real(2018)	(1/1)	VOC2007+2012	63.8(2020b)
	VGG-16	BiDet(2020b)	(1/1)	VOC2007+2012	52.4(2020b)
	VGG-16	BiDet(SC) ¹ (2020b)	(1/1)	VOC2007+2012	66.0(2020b)
	VGG-16	ReActNet(2020c)	(1/1)	VOC2007+2012	68.4(2021a)
	VGG-16	LWS-Det(2021a)	(1/1)	VOC2007+2012	71.4(2021a)
	MobileNetV1	Full Precision	(32/32)	VOC2007+2012	68.0(2020b)
	MobileNetV1	XNOR-Net(2016)	(1/1)	VOC2007+2012	48.9(2020b)
	MobileNetV1	BiDet	(1/1)	VOC2007+2012	51.2(2020b)
Yolo(2016)	VGG-16	Full Precision	(32/32)	VOC2007+2012	66.4(2019)
SSD512(2016)	VGG-16	Full Precision	(32/32)	VOC2007+2012	76.8(2019)
	BinaryDenseNet37(2019)	BinaryDenseNet	(1/1)	VOC2007+2012	66.4(2019)
	BinaryDenseNet45(2019)	BinaryDenseNet	(1/1)	VOC2007+2012	68.2(2019)

Note: mAP: Mean Average, ¹: BiDet(SC) means the proposed method with extra shortcut for the architectures Precision

6.4. Semantic Segmentation

Group-Net (Zhuang *et al.*, 2019) reports their method can be successfully applied to Semantic Segmentation task. The author proposes Binary Parallel Atrous Convolution (BPAC) to further improve the BNN model performance mIOU which is measured regarding averaged pixel intersection-over-union. Their used dataset for testing semantic segmentation is PASCAL VOC 2012 (Everingham *et al.*, 2010), which contains 20 foreground object classes and one background class. The dataset consists of 1464 (train), 1449 (val) and 1456 (test) images..

6.5. Other Tasks

There are other tasks which take the capacities of BNN. Bulat and Tzimiropoulos (2017) propose a new BNN network for human pose estimation and face alignment. Fasfous *et al.* (2021) present BinaryCoP which is a BNN classifier for correct facial-mask wear and positioning on edge devices. To speed-up large-scale image

Table 21. Benchmark results on MS-COCO minival.

Framework	BackBone	Method	(W/A)	mAP@.5(%)	mAP@[.5,.95](%)
Faster-RCNN(2015)	ResNet-18	Full Precision	(32/32)	42.7(2020)	21.9(2020)
	ResNet-18	Full Precision	(32/32)	44.8(2020b)	26.0(2020b)
	ResNet-18	Full Precision	(32/32)	53.8(2020b)	32.2(2020b)
	ResNet-18	ASDA-FRCNN(2020)	(1/1)	37.5(2020)	19.4(2020)
	ResNet-18	BNN(2016)	(1/1)	14.3(2020b)	5.6(2020b)
	ResNet-18	XNOR-Net(2016)	(1/1)	10.4(2020b)	21.6(2020b)
	ResNet-18	Bi-Real(2018)	(1/1)	29.0(2020b)	14.4(2020b)
	ResNet-18	Bi-Real(2018)	(1/1)	33.1(2021a)	17.4(2021a)
	ResNet-18	BiDet(2020b)	(1/1)	24.8(2020b)	12.1(2020b)
	ResNet-18	BiDet(SC) ¹ (2020b)	(1/1)	31.0(2020b)	15.7(2020b)
	ResNet-18	BiDet(2020b)	(1/1)	24.8(2021a)	12.1(2021a)

Note: mAP: Mean Average (AP), mAP@.5 : mAP for Intersection over Union (IoU) = 0.5, mAP@[.5, .95] : mAP for IoU $\in [0.5 : 0.05 : 0.95]$, ¹: BiDet(SC) means the proposed method with extra shortcut for the architectures Precision

Table 22. Continue Table 21: Benchmark results on MS-COCO minival.

Framework	BackBone	Method	(W/A)	mAP@.5(%)	mAP@[.5,.95](%)
Faster-RCNN(2015)	ResNet-18	ReActNet(2020c)	(1/1)	38.5(2021a)	21.1(2021a)
	ResNet-18	LWS-Det(2021a)	(1/1)	44.9(2021a)	26.9(2021a)
	ResNet-34	Full Precision	(32/32)	57.6(2021a)	35.8(2021a)
	ResNet-34	Bi-Real(2018)	(1/1)	37.1(2021a)	20.1(2021a)
	ResNet-34	BiDet(2020b)	(1/1)	41.8(2021a)	21.7(2021a)
	ResNet-34	ReActNet(2020c)	(1/1)	43.3(2021a)	23.4(2021a)
	ResNet-34	LWS-Det(2021a)	(1/1)	49.2(2021a)	29.9(2021a)
	ResNet-50	Full Precision	(32/32)	59.3(2021a)	37.7(2021a)
	ResNet-50	Bi-Real(2018)	(1/1)	40.0(2021a)	22.9(2021a)
	ResNet-50	ReActNet(2020c)	(1/1)	47.7(2021a)	26.1(2021a)
	ResNet-50	LWS-Det(2021a)	(1/1)	52.1(2021a)	31.7(2021a)
SSD300(2016)	VGG-16	Full Precision	(32/32)	41.2(2020b)	23.2(2020b)
	VGG-16	BNN(2016)	(1/1)	15.9(2020b)	6.2(2020b)
	VGG-16	XNOR-Net(2016)	(1/1)	19.5(2020b)	8.1(2020b)
	VGG-16	Bi-Real(2018)	(1/1)	26.0(2020b)	11.2(2020b)
	VGG-16	BiDet(2020b)	(1/1)	22.5(2020b)	9.8(2020b)
	VGG-16	BiDet(SC) ¹ (2020b)	(1/1)	28.3(2020b)	13.2(2020b)
	ResNet-50	ReActNet(2020c)	(1/1)	30.0(2021a)	15.3(2021a)
	ResNet-50	LWS-Det(2021a)	(1/1)	32.9(2021a)	17.1(2021a)

Note: mAP: Mean Average (AP), mAP@.5 : mAP for Intersection over Union (IoU) = 0.5, mAP@[.5, .95] : mAP for IoU $\in [0.5 : 0.05 : 0.95]$, ¹: BiDet(SC) means the proposed method with extra shortcut for the architectures Precision

retrieval search with low storage cost, Zhang *et al.* (2021a) propose a novelty hashing method called Binary Neural Network Hashing (BNNH) which combine BNN with hashing technique. Li *et al.* (2021) design a 3D BNN to recognize human actions. Their BNN can achieve 89.2% accuracy with 384 frames per second on the dataset KTH (Schuldt, Laptev, and Caputo, 2004). Penkovsky *et al.* (2020) propose methods to apply BNN to biomedical signals tasks such as electrocardiography (ECG) and electroencephalography (EEG) which can enable smart autonomous healthcare devices. In the field of natural language processing (NLP), Jain *et al.* (2020) explore and proposes a BNN method on text classification. (Xiang, Qian, and Yu, 2017), (Qian and Xiang, 2019) and (Gao *et al.*, 2021) are different BNN methods on speech recognition. Bahri, Bahl, and Zafeiriou (2021) explore the field of designing BNN-based graph neural networks (GNNs) by evaluating different strategies for the binarization of GNNs. Frickenstein *et al.* (2020) propose Binary DAD-Net which was the first BNN-based semantic segmentation network on driveable area detection in the field of autonomous driving.

Table 23. PASCAL VOC 2012 testing results

Based Model	BackBone	Method/(W/A)	mIOU
Faster-RCNN(2015), FCN-16s	ResNet-18	Full Precision(32/32)	67.3
		Group-Net(5 x 1/1)(2019)	62.7
		Group-Net + BPAC(5 x 1/1)(2019)	66.3
	ResNet-18**(2018)	Group-Net + BPAC(5 x 1/1)(2019)	67.7
Faster-RCNN(2015), FCN-32s	ResNet-18	Full Precision(32/32)	64.9
		Group-Net(5 x 1/1)(2019)	60.5
		Group-Net + BPAC(5 x 1/1)(2019)	63.8
	ResNet-18**(2018)	Group-Net + BPAC(5 x 1/1)(2019)	65.1
Faster-RCNN(2015), FCN-32s	ResNet-34	Full Precision(32/32)	72.7
		Group-Net(5 x 1/1)(2019)	68.2
		Group-Net + BPAC(5 x 1/1)(2019)	71.2
	ResNet-34**(2018)	Group-Net + BPAC(5 x 1/1)(2019)	72.8
Faster-RCNN(2015) , FCN-32s	ResNet-50	Full Precision(32/32)	73.1
		Group-Net(5 x 1/1)(2019)	67.2
		Group-Net + BPAC(5 x 1/1)(2019)	70.4

6.6. Summary

Although BNN has many successful applications, a few potential opportunities and challenges remain an open issue.

For a given application, what binary neural network architecture should be used? How automatically search architecture or create a 2D or 3D BNN network with higher accuracy and lower OPs. In general, all the layers (except the input and output layers) of a BNN are binarized CNN layers, which are one of the primary sources of losing information. This situation will be more difficult in the deeper layer because the performance drop is accumulated from the previous layers. Also, the information-lose level of different layers inside BNN should not be the same, and we should not be able to treat the information-loss issue equally. Besides, compared to information loss from binarization in 2D tasks, more severe information- loss from binarization can be generated in 3D tasks cases due to more dimensional information availability. Currently, there are few papers for solving 3D tasks.

How to develop transformer-based BNN models for vision tasks? In the past, deep neural networks mostly used costly convolution operations. Moreover, it is the reason and motivation to create the BNN that lowers expansive computing costs in convolution operations. Late last year, (Dosovitskiy et al., 2020) introduced a novel network structure called vision transformer (ViT). The idea for ViT came from the concept of transformer developed in natural language processing (NLP) applications. Instead of using the convolution operations, ViT splits an input image into custom fixed-size patches and feeds the linear projections of these patches along with their image position into a transformer encoder network. More and more transformer-based or a combination of transformer and convolution variants models were published. They reported that their performance could beat CNN-based models with the same weight size. Therefore, how to effectively develop a transformer-based BNN can be a new challenge and a hot research opportunity.

7. Conclusions

As mentioned above, since 2016, BNN techniques have drawn increasing research interest because of their capability to deploy models on resource-limited devices. BNN can significantly reduce storage, network complexity and energy consumption to make neural networks more efficient in embedded settings. However, binarization unavoidably causes a significant performance drop. In this paper, the literature on BNN techniques has been rigorously explored and discussed. For the first time, we solely focus on reviewing mainly 1-bit activations and weights networks that decrease the network memory usage and computational cost.

Furthermore, a comparative classification of these techniques has been performed and discussed under multiple network components: quantization function, activations/weights distribution, loss function improvement, gradient approximation, network topology structure, and training strategy and tricks. Additionally, we present popular efficient platforms for BNN and investigate current BNN applications progress. Also, we discussed and identified the research gap and the best methods available in the literature review. Finally, we

provide several recommendations and research directions for future exploration. We firmly believe that such an intricate field of BNN is just starting to permeate a broad range of artificial intelligence communities and tiny resource-constraint systems and will soon be taught to students and professionals as an essential topic in computer vision and deep learning.

References

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., *et al.*: 2016, Tensorflow: A system for large-scale machine learning. In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 265.
- Agaian, S.: 1986, *Hadamard matrices and their applications*, Springer-Verlag.
- Agaian, S., Sarukhanyan, H., Egiastian, K., Astola, J.: 2011, *Hadamard transforms* 4, SPIE Press, ??? ISBN 0819486477.
- Ajanthan, T., Gupta, K., Torr, P., Hartley, R., Dokania, P.: 2021, Mirror descent view for neural network quantization. In: *International Conference on Artificial Intelligence and Statistics*, 2809. PMLR.
- Akhauri, Y.: 2019, Hadanets: Flexible quantization strategies for neural networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 0.
- Alizadeh, M., Fernández-Marqués, J., Lane, N.D., Gal, Y.: 2018, An empirical study of binary neural networks' optimisation. In: *International Conference on Learning Representations*.
- Bahri, M., Bahl, G., Zafeiriou, S.: 2021, Binary graph neural networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9492.
- Bannink, T., Bakhtiari, A., Hillier, A., Geiger, L., de Bruin, T., Overweel, L., Neeven, J., Helwegen, K.: 2021, *Larq compute engine: Design, benchmark, and deploy state-of-the-art binarized neural networks*.
- Bengio, Y., Léonard, N., Courville, A.: 2013, Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.
- Bethge, J., Yang, H., Bornstein, M., Meinel, C.: 2019, Binarydensenet: Developing an architecture for binary neural networks. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*.
- Bethge, J., Bartz, C., Yang, H., Meinel, C.: 2020a, Bmnnet 2: An open source framework for low-bit networks-reproducing, understanding, designing and showcasing. In: *Proceedings of the 28th ACM International Conference on Multimedia*, 4469.
- Bethge, J., Bartz, C., Yang, H., Chen, Y., Meinel, C.: 2020b, Meliusnet: Can binary neural networks achieve mobilenet-level accuracy? *arXiv preprint arXiv:2001.05936*.
- Blott, M., Preußner, T.B., Fraser, N.J., Gambardella, G., O'brien, K., Umuroglu, Y., Leeser, M., Vissers, K.: 2018, Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 11(3), 1.
- Borji, A., Cheng, M.-M., Hou, Q., Jiang, H., Li, J.: 2019, Salient object detection: A survey. *Computational visual media* 5(2), 117.
- Breiman, L.: 1996, Bagging predictors. *Machine learning* 24(2), 123.
- Brock, A., De, S., Smith, S.L., Simonyan, K.: 2021, High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*.
- Bulat, A., Tzimiropoulos, G.: 2017, Binarized convolutional landmark localizers for human pose estimation and face alignment with limited resources. In: *Proceedings of the IEEE International Conference on Computer Vision*, 3706.
- Bulat, A., Tzimiropoulos, G.: 2019, Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*.
- Bulat, A., Martinez, B., Tzimiropoulos, G.: 2020a, Bats: Binary architecture search. *arXiv preprint arXiv:2003.01711*.
- Bulat, A., Martinez, B., Tzimiropoulos, G.: 2020b, High-capacity expert binary networks. In: *International Conference on Learning Representations*.
- Caesar, H., Uijlings, J., Ferrari, V.: 2018, Coco-stuff: Thing and stuff classes in context. In: *Computer vision and pattern recognition (CVPR), 2018 IEEE conference on*. IEEE.
- Campbell, R.J., Flynn, P.J.: 2001, A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding* 81(2), 166.
- Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., Zhang, Z.: 2015, Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.
- Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., Cowan, M., Wang, L., Hu, Y., Ceze, L., *et al.*: 2018, {TVM}: An automated end-to-end optimizing compiler for deep learning. In: *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 578.
- Chen, T., Zhang, Z., Ouyang, X., Liu, Z., Shen, Z., Wang, Z.: 2021, " bnn-bn=?": Training binary neural networks without batch normalization. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4619.
- Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., Bengio, Y.: 2016, Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*.
- Darabi, S., Belbahri, M., Courbariaux, M., Nia, V.P.: 2018, Regularized binary network training. *arXiv preprint arXiv:1812.11800*.
- Deepa, S., Devi, B.A., *et al.*: 2011, A survey on artificial intelligence approaches for medical image classification. *Indian Journal of Science and Technology* 4(11), 1583.
- Diffenderfer, J., Kaillkhura, B.: 2021, Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning a randomly weighted network. *arXiv preprint arXiv:2103.09377*.
- Ding, R., Chin, T.-W., Liu, Z., Marculescu, D.: 2019, Regularizing activation distribution for training binarized deep networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11408.
- Dong, Y., Ni, R., Li, J., Chen, Y., Zhu, J., Su, H.: 2017, Learning accurate low-bit deep neural networks with stochastic quantization. *arXiv preprint arXiv:1708.01001*.
- Dong, Y., Ni, R., Li, J., Chen, Y., Su, H., Zhu, J.: 2019, Stochastic quantization for learning accurate low-bit deep neural networks. *International Journal of Computer Vision* 127(11), 1629.
- Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*, <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>.

-
- Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*, <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>.
- Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: 2010, The pascal visual object classes (voc) challenge. *International journal of computer vision* **88**(2), 303.
- Fasfous, N., Vemparala, M.-R., Frickenstein, A., Frickenstein, L., Badawy, M., Stechele, W.: 2021, Binarycop: Binary neural network-based covid-19 face-mask wear and positioning predictor on edge devices. In: *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 108. IEEE.
- Frankle, J., Carbin, M.: 2018, The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.
- Freund, Y., Schapire, R.E.: 1997, A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* **55**(1), 119.
- Frickenstein, A., Vemparala, M.-R., Mayr, J., Nagaraja, N.-S., Unger, C., Tombari, F., Stechele, W.: 2020, Binary dad-net: Binarized driveable area detection network for autonomous driving. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2295. IEEE.
- Fromm, J., Cowan, M., Philipose, M., Ceze, L., Patel, S.: 2020, Riptide: Fast end-to-end binarized neural networks. *Proceedings of Machine Learning and Systems* **2**, 379.
- Gao, S., Wang, R., Jiang, L., Zhang, B.: 2021, 1-bit wavenet: Compressing a generative neural network in speech recognition with two binarized methods. In: *2021 IEEE 16th Conference on Industrial Electronics and Applications (ICIEA)*, 2043. DOI.
- Ghasemzadeh, M., Samragh, M., Koushanfar, F.: 2018, Rebnet: Residual binarized neural network. In: *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 57. IEEE.
- Gong, R., Liu, X., Jiang, S., Li, T., Hu, P., Lin, J., Yu, F., Yan, J.: 2019, Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4852.
- Goyal, S., Benjamin, P.: 2014, Object recognition using deep neural networks: A survey. *arXiv preprint arXiv:1412.3684*.
- Gu, J., Zhao, J., Jiang, X., Zhang, B., Liu, J., Guo, G., Ji, R.: 2019a, Bayesian optimized 1-bit cnns. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4909.
- Gu, J., Li, C., Zhang, B., Han, J., Cao, X., Liu, J., Doermann, D.: 2019b, Projection convolutional neural networks for 1-bit cnns via discrete back propagation. In: *Proceedings of the AAAI Conference on Artificial Intelligence* **33**, 8344.
- Guo, P., Ma, H., Chen, R., Li, P., Xie, S., Wang, D.: 2018, Fbna: A fully binarized neural network accelerator. In: *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, 51. DOI.
- Han, K., Wang, Y., Xu, Y., Xu, C., Wu, E., Xu, C.: 2020, Training binary neural networks through learning with noisy supervision. In: *International Conference on Machine Learning*, 4017. PMLR.
- He, X., Mo, Z., Cheng, K., Xu, W., Hu, Q., Wang, P., Liu, Q., Cheng, J.: 2020, Proxybnn: Learning binarized neural networks via proxy matrices. In: *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, 223. Springer.
- Helwegen, K., Widdicombe, J., Geiger, L., Liu, Z., Cheng, K.-T., Nusselder, R.: 2019, Latent weights do not exist: Rethinking binarized neural network optimization. *arXiv preprint arXiv:1906.02107*.
- Hou, L., Yao, Q., Kwok, J.T.: 2016, Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600*.
- Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: 2017, Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
- Hu, J., Shen, L., Sun, G.: 2018, Squeeze-and-excitation networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7132.
- Jafri, R., Ali, S.A., Arabnia, H.R., Fatima, S.: 2014, Computer vision-based object recognition for the visually impaired in an indoors environment: a survey. *The Visual Computer* **30**(11), 1197.
- Jain, H., Agarwal, A., Shridhar, K., Kleyko, D.: 2020, End to end binarized neural networks for text classification. *arXiv preprint arXiv:2010.05223*.
- Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., Qu, R.: 2019, A survey of deep learning-based object detection. *IEEE Access* **7**, 128837.
- Kim, D., Singh, K.P., Choi, J.: 2020, Learning architectures for binary networks. In: *European Conference on Computer Vision*, 575. Springer.
- Kim, H., Kim, K., Kim, J., Kim, J.-J.: 2020a, Binaryduo: Reducing gradient mismatch in binary activation network by coupling binary activations. *arXiv preprint arXiv:2002.06517*.
- Kim, H., Park, J., Lee, C., Kim, J.-J.: 2020b, Improving accuracy of binary neural networks using unbalanced activation distribution. *arXiv preprint arXiv:2012.00938*.
- Kim, M., Smaragdis, P.: 2016, Bitwise neural networks. *arXiv preprint arXiv:1601.06071*.
- Krizhevsky, A.: 2009, Learning multiple layers of features from tiny images.
- Krizhevsky, A., Sutskever, I., Hinton, G.E.: 2012, Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* **25**, 1097.
- Laydevant, J., Ernault, M., Querlioz, D., Grollier, J.: 2021, Training dynamical binary neural networks with equilibrium propagation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4640.
- Li, G., Zhang, M., Zhang, Q., Lin, Z.: 2021, Efficient binary 3d convolutional neural network and hardware accelerator. *Journal of Real-Time Image Processing*, 1.
- Li, R., Wang, Y., Liang, F., Qin, H., Yan, J., Fan, R.: 2019a, Fully quantized network for object detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2810.
- Li, X., Wang, W., Hu, X., Yang, J.: 2019b, Selective kernel networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 510.
- Li, Y., Zhang, H., Xue, X., Jiang, Y., Shen, Q.: 2018a, Deep learning for remote sensing image classification: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **8**(6), e1264.
- Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: 2018b, Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems* **31**, 820.
- Li, Y., Gong, R., Yu, F., Dong, X., Liu, X.: 2020, Dms: Differentiable dimension search for binary neural networks. In: *ICLR 2020 NAS Workshop*.
- Li, Z., Ni, B., Zhang, W., Yang, X., Gao, W.: 2017, Performance guaranteed network acceleration via high-order residual quantization. In: *Proceedings of the IEEE international conference on computer vision*, 2584.

-
- Liang, S., Yin, S., Liu, L., Luk, W., Wei, S.: 2018, Fp-bnn: Binarized neural network on fpga. *Neurocomputing* **275**, 1072.
- Lin, M., Ji, R., Xu, Z., Zhang, B., Wang, Y., Wu, Y., Huang, F., Lin, C.-W.: 2020, Rotated binary neural network. *Advances in Neural Information Processing Systems* **33**.
- Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: 2014, Microsoft coco: Common objects in context. In: *European conference on computer vision*, 740. Springer.
- Lin, X., Zhao, C., Pan, W.: 2017, Towards accurate binary convolutional neural network. *arXiv preprint arXiv:1711.11294*.
- Liu, C., Ding, W., Xia, X., Zhang, B., Gu, J., Liu, J., Ji, R., Doermann, D.: 2019a, Circulant binary convolutional networks: Enhancing the performance of 1-bit dcnn with circulant back propagation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2691.
- Liu, C., Ding, W., Xia, X., Hu, Y., Zhang, B., Liu, J., Zhuang, B., Guo, G.: 2019b, Rbcn: Rectified binary convolutional networks for enhancing the performance of 1-bit dcnn. *arXiv preprint arXiv:1908.07748*.
- Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M.: 2020a, Deep learning for generic object detection: A survey. *International journal of computer vision* **128**(2), 261.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: 2016, Ssd: Single shot multibox detector. In: *European conference on computer vision*, 21. Springer.
- Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., Cheng, K.-T.: 2018, 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)*, 722.
- Liu, Z., Luo, W., Wu, B., Yang, X., Liu, W., Cheng, K.-T.: 2020b, Bi-real net: Binarizing deep network towards real-network performance. *International Journal of Computer Vision* **128**(1), 202.
- Liu, Z., Shen, Z., Savvides, M., Cheng, K.-T.: 2020c, Reactnet: Towards precise binary neural network with generalized activation functions. In: *European Conference on Computer Vision*, 143. Springer.
- Liu, Z., Shen, Z., Li, S., Helwegen, K., Huang, D., Cheng, K.-T.: 2021, How do adam and training strategies help bnns optimization? In: *International Conference on Machine Learning*. PMLR.
- Livochka, A., Shekhovtsov, A.: 2021a, Initialization and transfer learning of stochastic binary networks from real-valued ones. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 4660.
- Livochka, A., Shekhovtsov, A.: 2021b, Initialization and transfer learning of stochastic binary networks from real-valued ones. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4660.
- Long, J., Shelhamer, E., Darrell, T.: 2015, Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431.
- Lu, D., Weng, Q.: 2007, A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing* **28**(5), 823.
- Martinez, B., Yang, J., Bulat, A., Tzimiropoulos, G.: 2020, Training binary neural networks with real-to-binary convolutions. *arXiv preprint arXiv:2003.11535*.
- Mishra, A., Nurvitadhi, E., Cook, J.J., Marr, D.: 2017, Wrpn: Wide reduced-precision networks. *arXiv preprint arXiv:1709.01134*.
- Nakahara, H., Fujii, T., Sato, S.: 2017, A fully connected layer elimination for a binarized convolutional neural network on an fpga. *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, 1.
- Nath, S.S., Mishra, G., Kar, J., Chakraborty, S., Dey, N.: 2014, A survey of image classification methods and techniques. In: *2014 International conference on control, instrumentation, communication and computational technologies (ICCICCT)*, 554. IEEE.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: 2019, Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.
- Penkovsky, B., Bocquet, M., Hirtzlin, T., Klein, J.-O., Nowak, E., Vianello, E., Portal, J.-M., Querlioz, D.: 2020, In-memory resistive ram implementation of binarized neural networks for medical applications. In: *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 690. IEEE.
- Pham, P., Abraham, J.A., Chung, J.: 2021, Training multi-bit quantized and binarized networks with a learnable symmetric quantizer. *IEEE Access* **9**, 47194.
- Phan, H., Liu, Z., Huynh, D., Savvides, M., Cheng, K.-T., Shen, Z.: 2020a, Binarizing mobilenet via evolution-based searching. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13420.
- Phan, H., He, Y., Savvides, M., Shen, Z., et al.: 2020b, Mobinet: A mobile binary network for image classification. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 3453.
- Qi, C.R., Yi, L., Su, H., Guibas, L.J.: 2017a, Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*.
- Qi, C.R., Su, H., Mo, K., Guibas, L.J.: 2017b, Pointnet: Deep learning on point sets for 3d classification and segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652.
- Qian, Y.-m., Xiang, X.: 2019, Binary neural networks for speech recognition. *Frontiers of Information Technology & Electronic Engineering* **20**(5), 701.
- Qiao, G., Hu, S., Chen, T., Rong, L., Ning, N., Yu, Q., Liu, Y.: 2020, Stbnn: Hardware-friendly spatio-temporal binary neural network with high pattern recognition accuracy. *Neurocomputing* **409**, 351.
- Qin, H., Gong, R., Liu, X., Bai, X., Song, J., Sebe, N.: 2020a, Binary neural networks: A survey. *Pattern Recognition* **105**, 107281.
- Qin, H., Cai, Z., Zhang, M., Ding, Y., Zhao, H., Yi, S., Liu, X., Su, H.: 2020b, Bipointnet: Binary neural network for point clouds. *arXiv preprint arXiv:2010.05501*.
- Qin, H., Gong, R., Liu, X., Shen, M., Wei, Z., Yu, F., Song, J.: 2020c, Forward and backward information retention for accurate binary neural networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2250.
- Qin, H., Gong, R., Liu, X., Shen, M., Wei, Z., Yu, F., Song, J.: 2020d, Forward and backward information retention for accurate binary neural networks. In: *IEEE CVPR*.
- Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: 2016, Xnor-net: Imagenet classification using binary convolutional neural networks. In: *European conference on computer vision*, 525. Springer.
- Redfern, A.J., Zhu, L., Newquist, M.K.: 2021, Bcnn: A binary cnn with all matrix ops quantized to 1 bit precision. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4604.
- Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: 2016, You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779.

-
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., *et al.*: 2015, Imagenet large scale visual recognition challenge. *International journal of computer vision* **115**(3), 211.
- Sakr, C., Choi, J., Wang, Z., Gopalakrishnan, K., Shanbhag, N.: 2018, 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)*, 2346. IEEE.
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: 2018, Mobilenetv2: Inverted residuals and linear bottlenecks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510.
- Schuldt, C., Laptev, I., Caputo, B.: 2004, Recognizing human actions: a local svm approach. In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* **3**, 32. IEEE.
- Shantaiya, S., Verma, K., Mehta, K.: 2013, A survey on approaches of object detection. *International Journal of Computer Applications* **65**(18).
- Shen, M., Han, K., Xu, C., Wang, Y.: 2019, Searching for accurate binary neural architectures. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 0.
- Shen, M., Liu, X., Gong, R., Han, K.: 2020, Balanced binary neural networks with gated residual. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4197. IEEE.
- Simons, T., Lee, D.-J.: 2019, A review of binarized neural networks. *Electronics* **8**(6), 661.
- Simonyan, K., Zisserman, A.: 2014, Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Sukanya, C., Gokul, R., Paul, V.: 2016, A survey on object recognition methods. *International Journal of Science, Engineering and Computer Technology* **6**(1), 48.
- Sun, S., Yin, Y., Wang, X., Xu, D., Wu, W., Gu, Q.: 2018, Fast object detection based on binary deep convolution neural networks. *CAAI transactions on intelligence technology* **3**(4), 191.
- Tang, W., Hua, G., Wang, L.: 2017, How to train a compact binary neural network with high accuracy? In: *Proceedings of the AAAI Conference on Artificial Intelligence* **31**.
- Tieleman, T., Hinton, G.: 2012, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* **4**(2), 26.
- Wang, E., Davis, J.J., Moro, D., Zielinski, P., Lim, J.J., Coelho, C., Chatterjee, S., Cheung, P.Y., Constantinides, G.A.: 2021a, Enabling binary neural network training on the edge. In: *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, 37.
- Wang, P., He, X., Li, G., Zhao, T., Cheng, J.: 2020a, Sparsity-inducing binarized neural networks. In: *Proceedings of the AAAI Conference on Artificial Intelligence* **34**, 12192.
- Wang, S., Zhang, C., Su, D., Wang, L., Jiang, H.: 2021b, High-precision binary object detector based on a bsf-xnor convolutional layer. *IEEE Access* **9**, 106169. DOI.
- Wang, W., Yang, Y., Wang, X., Wang, W., Li, J.: 2019a, Development of convolutional neural network and its application in image classification: a survey. *Optical Engineering* **58**(4), 040901.
- Wang, X., Zhang, B., Li, C., Ji, R., Han, J., Cao, X., Liu, J.: 2018, Modulated convolutional networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: 2019b, Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)* **38**(5), 1.
- Wang, Y., Yang, Y., Sun, F., Yao, A.: 2021c, Sub-bit neural networks: Learning to compress and accelerate binary neural networks. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5360.
- Wang, Z., Lu, J., Tao, C., Zhou, J., Tian, Q.: 2019c, Learning channel-wise interactions for binary convolutional neural networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 568.
- Wang, Z., Wu, Z., Lu, J., Zhou, J.: 2020b, Bidet: An efficient binarized object detector. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2049.
- Wu, W., Qi, Z., Fuxin, L.: 2019, Pointconv: Deep convolutional networks on 3d point clouds. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9621.
- Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 2015, 3d shapenets: A deep representation for volumetric shapes. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1912.
- Xiang, X., Qian, Y., Yu, K.: 2017, Binary deep neural networks for speech recognition. In: *INTERSPEECH*, 533.
- Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: 2017, Aggregated residual transformations for deep neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1492.
- Xu, S., Liu, Z., Gong, X., Liu, C., Mao, M., Zhang, B.: 2020, Amplitude suppression and direction activation in networks for 1-bit faster r-cnn. In: *Proceedings of the 4th International Workshop on Embedded and Mobile Deep Learning, EMDL'20*, Association for Computing Machinery, New York, NY, USA, 19–24. ISBN 9781450380737. DOI. <https://doi.org/10.1145/3410338.3412340>.
- Xu, S., Zhao, J., Lu, J., Zhang, B., Han, S., Doermann, D.: 2021a, Layer-wise searching for 1-bit detectors. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5682.
- Xu, Y., Dong, X., Li, Y., Su, H.: 2019, A main/subsidiary network framework for simplifying binary neural networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7154.
- Xu, Z., Cheung, R.C.: 2019, Accurate and compact convolutional neural networks with trained binarization. *arXiv preprint arXiv:1909.11366*.
- Xu, Z., Lin, M., Liu, J., Chen, J., Shao, L., Gao, Y., Tian, Y., Ji, R.: 2021b, Recu: Reviving the dead weights in binary neural networks. *arXiv preprint arXiv:2103.12369*.
- Yang, H., Fritzsche, M., Bartz, C., Meinel, C.: 2017, Bmxnet: An open-source binary neural network implementation based on mxnet. In: *Proceedings of the 25th ACM international conference on Multimedia*, 1209.
- Yang, J., Shen, X., Xing, J., Tian, X., Li, H., Deng, B., Huang, J., Hua, X.-s.: 2019, Quantization networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7308.
- Yang, L., He, Z., Fan, D.: 2018, A fully onchip binarized convolutional neural network fpga impelmentation with accurate inference. In: *Proceedings of the International Symposium on Low Power Electronics and Design, ISLPED '18*, Association for Computing Machinery, New York, NY, USA. ISBN 9781450357043. DOI. <https://doi.org/10.1145/3218603.3218615>.
- Yang, Z., Wang, Y., Han, K., XU, C., Xu, C., Tao, D., Xu, C.: 2020, Searching for low-bit weights in quantized neural networks. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) *Advances in Neural In-*

-
- formation Processing Systems **33**, Curran Associates, Inc., ???, 4091. <https://proceedings.neurips.cc/paper/2020/file/2a084e55c87b1ebcdaad1f62fdbbac8e-Paper.pdf>.
- Yonekawa, H., Nakahara, H.: 2017, On-chip memory based binarized convolutional deep neural network applying batch normalization free technique on an fpga. In: *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 98. DOI.
- Zagoruyko, S., Komodakis, N.: 2016, Wide residual networks. In: *British Machine Vision Conference 2016*. British Machine Vision Association.
- Zhang, D., Yang, J., Ye, D., Hua, G.: 2018, Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In: *Proceedings of the European conference on computer vision (ECCV)*, 365.
- Zhang, J., Pan, Y., Yao, T., Zhao, H., Mei, T.: 2019, dabnn: A super fast inference framework for binary neural networks on arm devices. In: *Proceedings of the 27th ACM international conference on multimedia*, 2272.
- Zhang, W., Wu, D., Zhou, Y., Li, B., Wang, W., Meng, D.: 2021a, Binary neural network hashing for image retrieval. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1318.
- Zhang, Y., Pan, J., Liu, X., Chen, H., Chen, D., Zhang, Z.: 2021b, Fracbnn: Accurate and fpga-efficient binary neural networks with fractional activations. In: *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 171.
- Zhao, J., Xu, S., Wang, R., Zhang, B., Guo, G., Doermann, D., Sun, D.: 2021, Data-adaptive binary neural networks for efficient object detection and recognition. *Pattern Recognition Letters*.
- Zhao, R., Song, W., Zhang, W., Xing, T., Lin, J.-H., Srivastava, M., Gupta, R., Zhang, Z.: 2017, Accelerating Binarized Convolutional Neural Networks with Software-Programmable FPGAs. *Int'l Symp. on Field-Programmable Gate Arrays (FPGA)*.
- Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., Zou, Y.: 2016, Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR* **abs/1606.06160**. <http://arxiv.org/abs/1606.06160>.
- Zhou, T., Fan, D.-P., Cheng, M.-M., Shen, J., Shao, L.: 2021, Rgb-d salient object detection: A survey. *Computational Visual Media*, 1.
- Zhu, B., Al-Ars, Z., Hofstee, H.P.: 2020, Nasb: Neural architecture search for binary convolutional neural networks. In: *2020 International Joint Conference on Neural Networks (IJCNN)*, 1. IEEE.
- Zhu, P., Wen, L., Bian, X., Ling, H., Hu, Q.: 2018, Vision meets drones: A challenge. *arXiv preprint arXiv:1804.07437*.
- Zhu, S., Dong, X., Su, H.: 2019, Binary ensemble neural network: More bits per network or more networks per bit? In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4923.
- Zhuang, B., Shen, C., Tan, M., Liu, L., Reid, I.: 2019, Structured binary neural networks for accurate image classification and semantic segmentation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 413.
- Zhuang, B., Shen, C., Tan, M., Chen, P., Liu, L., Reid, I.: 2021, *Structured binary neural networks for image recognition*.
- Zou, Z., Shi, Z., Guo, Y., Ye, J.: 2019, Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*.
- 019, Open neural network exchange. <https://github.com/onnx/onnx>.