IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 22 February 2023, accepted 14 March 2023, date of publication 16 March 2023, date of current version 22 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3258360

SURVEY

A Systematic Literature Review on Binary Neural Networks

RATSHIH SAYED[1], **HAYTHAM AZMI**[1], **HEBA SHAWKEY**[1], **A. H. KHALIL**[2], **AND MOHAMED REFKY**[2]

[1]Microelectronics Department, Electronics Research Institute, Cairo 11843, Egypt
[2]Department of Electronics and Communications Engineering, Cairo University, Giza 12613, Egypt

Corresponding author: Ratshih Sayed (ratshih@eri.sci.eg)

ABSTRACT This paper presents an extensive literature review on Binary Neural Network (BNN). BNN utilizes binary weights and activation function parameters to substitute the full-precision values. In digital implementations, BNN replaces the complex calculations of Convolutional Neural Networks (CNNs) with simple bitwise operations. BNN optimizes large computation and memory storage requirements, which leads to less area and power consumption compared to full-precision models. Although there are many advantages of BNN, the binarization process has a significant impact on the performance and accuracy of the generated models. To reflect the state-of-the-art in BNN and explore how to develop and improve BNN-based models, we conduct a systematic literature review on BNN with data extracted from 239 research studies. Our review discusses various BNN architectures and the optimization approaches developed to improve their performance. There are three main research directions in BNN: accuracy optimization, compression optimization, and acceleration optimization. The accuracy optimization approaches include quantization error reduction, special regularization, gradient error minimization, and network structure. The compression optimization approaches combine fractional BNN and pruning. The acceleration optimization approaches comprise computing in-memory, FPGA-based implementations, and ASIC-based implementations. At the end of our review, we present a comprehensive analysis of BNN applications and their evaluation metrics. Also, we shed some light on the most common BNN challenges and the future research trends of BNN.

INDEX TERMS Binary neural network, convolutional neural network, deep learning, optimization approaches, quantization, systematic literature review.

I. INTRODUCTION

IN recent years, Convolutional Neural Network (CNN) achieved massive success in various aspects of image classification [1], [2], object recognition [3], [4], object detection [5], speech emotion recognition [6], [7], and classification of noisy non-stationary signals [8]. The standard CNN models use 32-bit floating-point arithmetic operations that require complex computations exhausting high power and large memory capacity. These problems make CNN inappropriate for limited sources platforms. To handle the CNN drawbacks, compression techniques appeared like parameter quantization [9] and parameter pruning [10], [11]. Quantization is a process to represent the weights of the neural network with low-precision formats, like integers or even binary numbers. Therefore, it is an efficient solution to provide a light implementation of CNN [12]. The maximum quantization level is to use a 1-bit representation, which is called binarization, in which the weights and activations are binary values. In BNN, all layers are binarized except the first and the last layers to keep the model accurate. BNN uses simple binary operations instead of the complex operations used in the full-precision counterpart.

This study aims to conduct a systematic literature review to propose an up-to-date comprehensive view of the BNN. This review keeps track of Kitchenham's guidelines [13]. The contributions of this survey are as follows:

1) Conducting a systematic literature review that presents the state-of-the-art in BNN through the data obtained from 239 research studies.

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Hao Chen.

2) Presenting a comprehensive review of three BNN optimization approaches: accuracy optimization, compression optimization, and acceleration optimization.
3) Exploring the various application domains that utilize BNN implementations and their evaluation metrics.
4) Identifying current challenges in BNN design and the future trends in BNN research.

The paper is organized as follows. Section II reports the methodology of the survey. Section III discusses the previously conducted surveys. Section IV analyzes the findings of the systematic literature review. Finally, section V presents the conclusion of the survey.

II. SURVEY METHODOLOGY

We follow Kitchenham's guidelines [13] to conduct our survey, as depicted in the following sub-sections. We begin by listing the research questions. Next, we explain the selection method, which includes the definition of the search string and the inclusion and exclusion criteria.

A. RESEARCH QUESTIONS

The research questions (RQ) are designed to express the fundamental data for the BNN review. The first three questions are necessary to understand the BNN background. The fourth question explains the BNN optimization approaches. The fifth question reports the existing BNN framework. The sixth question clarifies the applications of BNN and their evaluations. The last question identifies the challenges and the suggested future work of the BNN.

- RQ1: How is BNN defined in the literature, and why is it appropriate for source-limited devices?
- RQ2: Which is the pioneering research in BNN?
- RQ3: How is BNN trained?
- RQ4: What are the different approaches to optimize the performance of BNN?
- RQ5: What are the types of BNN frameworks?
- RQ6: What are the applications that utilize BNN? What are the used datasets in these applications and their evaluation metrics?
- RQ7: What are the challenges and future work of the BNN?

B. SELECTION APPROACH

We conducted our literature review using the following search engines: IEEE Xplore, Google scholar, ArXiv, Science Direct, ACM Digital Library, Springer Link, and Web of Science. The selection approach is divided into the following two parts:

1) DEFINITION OF THE SEARCH STRING

These are the keywords that are utilized to obtain the research results. We used the search strings for each search engine to examine the title, abstract, and keywords of the papers. The search strings are depicted in Table 1.

2) INCLUSION AND EXCLUSION CRITERIA

These criteria specify the related studies to the research questions.

1) Inclusion Criteria
   - We focus on the studies that provide explanations of BNN concepts, improvement approaches, and applications.
   - We comprise the review and survey studies to comprehend the current research trends.
2) Exclusion Criteria
   - Papers that are not published in the English language.
   - Papers that introduce approaches not related to BNN.
   - Repeated studies that have journal and conference versions.

The primary search found 2,392 papers, with 194 from IEEE Xplore, 1,890 from Google scholar, 115 from ArXiv, 101 from ScienceDirect, 82 from ACM Digital Library, 10 from Springer Link. After the repeated papers are removed and the inclusion and exclusion criteria are applied, the final selected papers are illustrated in Table 1. The survey time frame is between 2016 and September 2022. Figure 1 shows the paper selection approach process.

III. RELATED WORK

Few surveys on BNN have been published in the last few years. This section discusses the relevant reviews on BNN and illustrates the differences between our study and the previous surveys.

Khoshavi et al. [14] review the effect of soft errors on BNN, which occur due to compression techniques used that decrease the model size on the memory. Another study explored some BNN applications that are suitable for edge computing [15].

Simons et al. [16] demonstrated the BNN fundamentals and its benefits during the training and inference phases. The survey covered some architectures that improve the BNN performance. Moreover, the paper reviewed hardware implementations on FPGA, a short brief about ASIC implementations, and focused on image classification as an application. Additionally, surveys in [17] and [18] summarized the BNN background, and optimization methods including minimizing the quantization error, improving the network loss function, reducing the gradient error, and network structure. Besides, they mention some of the BNN applications.

Our review differs from the existing reviews from three perspectives: 1) Methodology: we conduct a systematic literature review on BNN that follows Kitchenham's guidelines [13]. While the previous surveys have no clear methodology. 2) Comprehensiveness: the number of studies and scope of the reviewed work that is analyzed in this survey is higher than previous work. 3) Analysis: we provided a comprehensive analysis of optimization approaches that were

TABLE 1. Search strings and their results.

Search Engines	Search Strings	Primary Results	Final Selected Result
IEEE Xplore	"BNN" or "Binary Neural Network" or "BNN training" or "Binary Neural Network" and "optimization techniques" or "Binary Neural Network" and "Hardware implementation"	194	97
Google scholar	Title: ["BNN" or "Binary Neural Network" or "BNN training" or "Binary Neural Network" + "optimization techniques" or "Binary Neural Network" + "Hardware implementation"]	1,890	56
ArXiv	Abstract ["BNN" or "Binary Neural Network" or "BNN training" or "Binary Neural Network" and "optimization techniques" or "Binary Neural Network" and "Hardware implementation"]	115	58
Science Direct	"BNN" or "Binary Neural Network" or "BNN training" or "Binary Neural Network and optimization techniques" or "Binary Neural Network and Hardware implementation"	101	8
ACM Digital Library	[All:"BNN"] or [All:"Binary Neural Network"] or [All:"BNN training"] or [All:"Binary Neural Network" and "optimization techniques"] or [All:"Binary Neural Network" and "Hardware implementation"]	82	12
Springer Link	Topic: ["BNN" or "Binary Neural Network" or "BNN training" or "Binary Neural Network" and "optimization techniques" or "Binary Neural Network" and "Hardware implementation"]	10	8

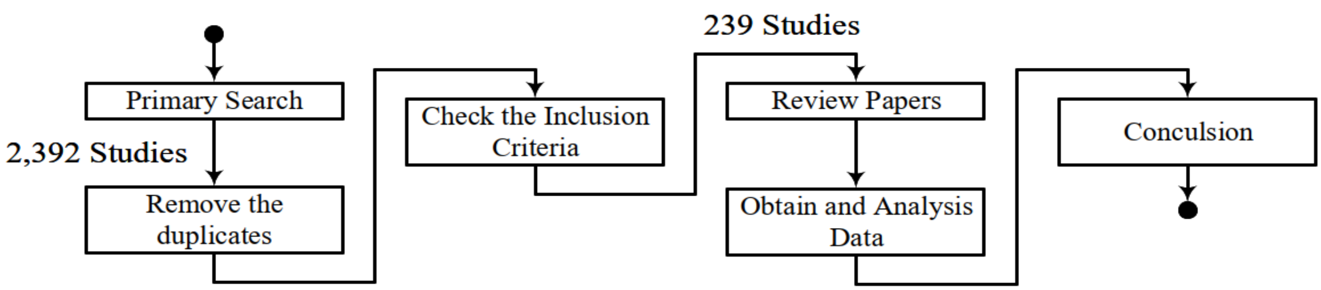

FIGURE 1. The papers selection approach process.

not covered in the previous survey, such as compression and acceleration approaches.

IV. SURVEY FINDINGS

This section summarized the results of the extracted data from the selected studies to answer the research questions.

A. RQ1: HOW IS BNN DEFINED IN THE LITERATURE, AND WHY IS IT APPROPRIATE FOR SOURCE-LIMITED DEVICES?

BNN is a neural network that can use 1-bit for data representation. Therefore, values of -1 (0) and 1 can be used for both weights and activations rather than 32-bit in a full-precision counterpart, which reduces the memory footprint. Another benefit of using the binary values is using binary XNOR operations and pop-count as alternatives to the dense matrix multiplication operations. Consequently, BNN can save much more area and power consumption as it provides a significant performance acceleration. All these features make the BNN suitable for source-limited devices. However, the binarization process causes large information loss. To alleviate this loss, the first and the last layers are not binarized to avoid performance degradation.

B. RQ2: WHICH IS THE PIONEERING RESEARCH OF THE BNN?

The leading work in BNN research began with the BinaryConnect [19], which developed a deep neural network (DNN) using binary weights $\{-1,+1\}$ in forward propagation and utilized the real values for updating the gradients in backward propagation in the training phase. While the BinaryConnect applies binarization for weights, and the activations are still represented by full-precision. Consequently, multiplication and accumulation processes are replaced by fixed-point adders to reduce the area and the power consumption. As an extension of the BinaryConnect, The Binarized Neural Networks (BNN) which use binary weights and activations published in [20] is considered the first BNN. The BNN in [20] achieved a 32 times compression ratio on weights and 7 times faster inference speed, using a custom GPU, compared to BinaryConnect on small datasets like MNIST [21], CIFAR-10 [22], and SVHN [23] datasets. However, experimental results revealed that this training technique was not appropriate for large datasets like ImageNet [24] and caused accuracy degradation.

To enhance the performance on large-scale datasets, XNOR-Net [25] was proposed. XNOR-Net [25] is dif-

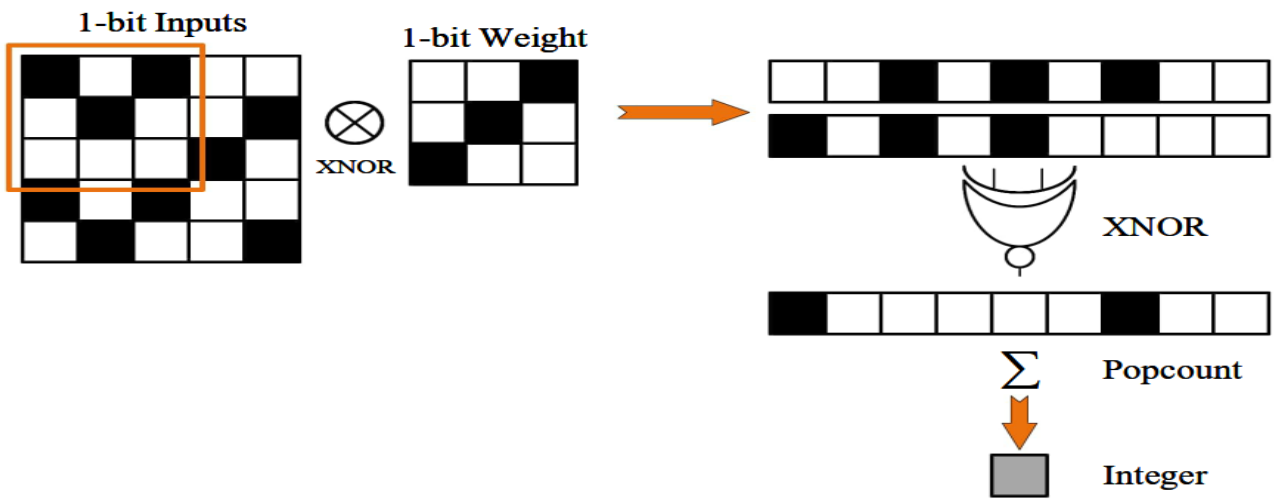

FIGURE 2. Binary convolution in BNN.

ferent from BinaryConnect and BNN [20] in the binarization technique and layer order. The authors utilized binary weights and activations but applied scaling factors to increase the accuracy and decrease the quantization errors. The XNOR-Net [25] achieves 32 times lower memory savings and 58 times faster convolutional operations compared to the full-precision counterpart. This method achieved a better trade-off between compression ratio and accuracy.

C. RQ3: HOW IS BNN TRAINED?

The BNN training process composes of forward propagation and backward propagation. In forward propagation, the input is fed to the input layer and passes through mathematical operations until reaching the output layer. The main mathematical operation in this process is convolution. Also, the forward propagation represents the model inference. In back-propagation, when the output is produced, it is compared with the actual value to determine the error. Then, the parameters' values are updated. The back-propagation is used to fine-tune the network parameters.

BNN utilizes binary weights and activations by employing the sign function to realize binarization. This scheme exchanges the full-precision convolution operation with XNOR and pop-count operations.

During the forward-propagation, the binary weights ($\mathrm{W_{Bin}}$) and binary activations ($\mathrm{A_{Bin}}$) are calculated by using the sign function of their corresponding full-precision ($\mathrm{W_{Real}}$), ($\mathrm{A_{Real}}$), respectively. The sign function is specified by the following equation [20]:

$$sign(y) = \begin{cases} 1, & \text{if } y \geq 0, \\ -1, & \text{otherwise.} \end{cases} \tag{1}$$

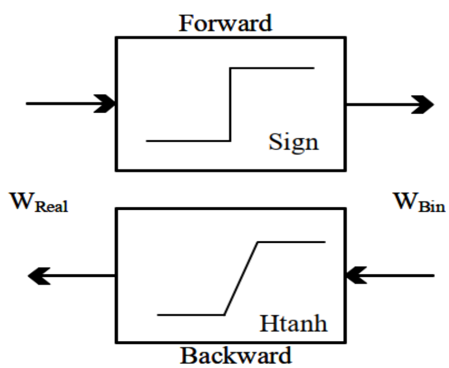

FIGURE 3. Forward and backward propagation functions in BNN.

Subsequently, the binary weights and activations are given by:

$$\begin{aligned} W_{\mathrm{Bin}} &= sign(W_{\mathrm{Real}}). \\ A_{\mathrm{Bin}} &= sign(A_{\mathrm{Real}}). \end{aligned} \tag{2}$$

The full-precision convolution process is performed using the multiplication and accumulation processes that occur in the "neurons". The equivalent process of the convolution in BNN is the XNOR operator followed by pop-count [25] to get the convoluted pixel value, as shown in Figure 2. The convolution without bias can be represented by the following formula:

$$\begin{aligned} &Z = f(popcount(XNOR(W_{Bin}, A_{Bin}))), \\ &Wi, Ai \in \{-1, 1\} \forall i. \end{aligned} \tag{3}$$

where Z is the output of the convolution layer, and $f(.)$ is the activation function represented as a *sign* function in forward propagation and *hard tanh* in backward propagation as shown in Figure 3.

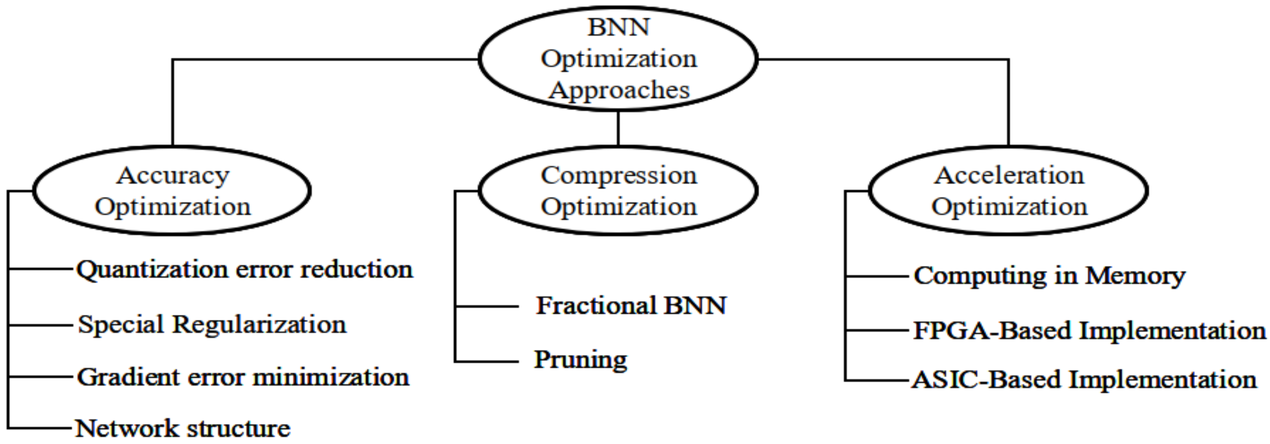

FIGURE 4. BNN optimization approaches.

TABLE 2. The XNOR multiplication operation.

Binary mapped values		Multiplication operation
-1 $\rightarrow$ 0	-1 $\rightarrow$ 0	1 $\rightarrow$ 1
-1 $\rightarrow$ 0	1 $\rightarrow$ 1	-1 $\rightarrow$ 0
1 $\rightarrow$ 1	-1 $\rightarrow$ 0	-1 $\rightarrow$ 0
1 $\rightarrow$ 1	1 $\rightarrow$ 1	1 $\rightarrow$ 1

In back-propagation, the Straight-Through-Estimator (STE) [26] is applied to update the gradient of the cost function at the output, as the sign function is not differentiable. The STE clips the gradients out of the range $\{1, -1\}$. That means the STE applies the *hard tanh* function to update the gradient during back-propagation. Similarly, to binarize the activations the STE is applied in back-propagation to obtain the values inside the interval $[-1,1]$ and 0 otherwise. After having the binary values, the multiplication process between the weights and the activations is reduced to binary operation. The obtained signed binary values are 1 and -1. These values are mapped to 1 and 0. The XNOR operator is applied to the binary mapped values to perform the multiplication process as a dot product, as illustrated in Table 2.

D. RQ4: WHAT ARE THE DIFFERENT APPROACHES TO OPTIMIZE THE PERFORMANCE OF BNN?

While the BNN is faster in inference time and consumes less power and memory footprint regarding the full-precision counterpart, it suffers from information loss due to binary values of weights and activations. Therefore, there are many improvement approaches proposed to enhance the BNN performance. Some of these approaches enhance the accuracy, whereas others are used to compress and accelerate the BNN [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38]. Figure 4 illustrates the classification of the above-mentioned approaches, and Figure 5 shows the percentage of the BNN papers in each optimization approach. These percentages are based on the number of papers on the optimization approaches reviewed in this survey.

1) ACCURACY OPTIMIZATION APPROACHES

The accuracy optimization approaches are used during the training phase to enhance the performance of the BNN. The accuracy optimization approaches are categorized into four categories: quantization error reduction for weights and activations, special regularization that penalizes the network parameters, gradient error minimization during back-propagation, and network structure modification to improve the network performance. Table 3 to Table 4 present a summary of various optimization techniques used in the literature based on ImageNet and CIFAR-10 datasets, respectively.

a: QUANTIZATION ERROR REDUCTION

In BNN, the binary values of both weights and activation allow bit-wise operation, leading to an accuracy drop with respect to the full-precision counterpart. Some research work used quantization for weights like BinaryConnect [19], Ternary Weight Networks (TWN) [39], Fine-Grained Quantization (FGQ) [40], Trained Ternary Quantization (TTQ) [41], Incremental Network Quantization (INQ) [42], and Smart Quantization (SQ) [43].

To decrease the error resulting from the extreme quantization, scaling factors can be applied after the occurrence of the dot product, as in XNOR-Net [25]. XNOR-Net utilizes channel-wise scaling factors for weights and activations. While Zhou et al. [44] proposed DoReFa-Net, which applied the quantization on weights, activations, and gradients. The authors employed a constant scalar to scale all filters instead of channel-wise scaling like XNOR-Net. Similar to XNOR-Net, Hu et al. [45] introduced Binary Weight Networks via Hashing (BWNH). BWNH mapped the binary weights to a hash map multiplied by a scaling factor. Li et al. [46] devised the High-Order Residual Quantization method (HORQ) based on the idea of the XNOR-Net as binary operations, but it performed residual quantization. This quantization method does recursive binary quantization each time the residual error is calculated and then, performs a new round of quantization to generate a group of binary maps corresponding to different quantization scales. Another training algorithm proposed in [47] applies the quantization on both weights and activations by learnable scaling factors. In [48], XNOR-Net++ utilized one learnable scaling factor for both weights and activations. Also, ABC-Net [49] introduced a linear combination of multiple binary weights and multiple binary activations. The ABC-Net approximated all the weights with scaling factors and applied other scaling factors to approximate each channel's weights. Li et al. [50] provided a Fixed-Sign Binary Neural Network (FSB) that learns the scaling factors for weights to be quantized but uses a fixed sign for them.

Besides the scaling factor as an optimization to alleviate the quantization error, some researchers apply quantization functions as in [51], Wide Reduced-Precision Networks (WRPN) quantized both activations and weights by using a quantization function and expanding the number of filters in all layers. Similarly, Half-Wave Gaussian Quantizer (HWGQ) [52] has been used HWGQ as a quantization function in the forward propagation and a clipped ReLU function in the back-propagation. Also, Faraone et al. [53] applied a quantization function for both weights and activations. In addition, the authors improve the quantization by utilizing scaling factors for each group of weights according to their locations in the weight matrix. Another quantization function is provided by Choi et al. [54] provided PArameterized Clipping acTivation (PACT) function to quantize the activations. This method used an optimized learnable clipping scale that determines the activation function's upper limit. Similarly, Zhang et al. [55] proposed a learnable quantization for both weights and activations. Also, Wang et al. [56] proposed another quantization method called Two Steps Quantization (TSQ) performed on two steps. The first step is the code learning step, and the second step is the transformation function learning. The first step used the sparse quantization method to learn sparse and low-bit codes. The second one is a non-linear least square regression problem with low-bit constraints that can be solved iteratively. Yang et al. [57] used a differentiable non-linear function to quantize both weights and activations. This quantization function composed of a group of Sigmoid functions. While Qin et al. [58] provided Information Retention Network (IR-Net), which proposed Libra Parameter Binarization (Libra-PB) in the forward propagation to provide balanced quantization and reduce the information loss of parameters. In [59], The authors introduced bit-level sparsity quantization (BSQ) for mixed-precision quantization that treated every bit of quantized weights as an independent trainable variable. Also, it applied scaling factors to the weights. Gong et al. [60] proposed Differentiable Soft Quantization (DSQ), which is adjusted during the training. DSQ detects the clipping range and

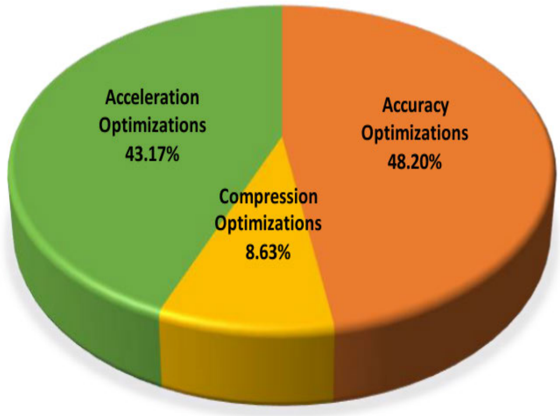

FIGURE 5. The BNN optimization approaches percentage.

quantization approximation by utilizing a series of hyperbolic tangent functions to approximate the staircase function to low-bit quantization. ProxyBNN [61] aimed to minimize the weights quantization errors by providing a proxy matrix. This proxy matrix breaks the pre-binarized weights into a linear combination of the basis and coordinates serve as auxiliary variables. In [35], the authors shifted the activation distribution to be unbalanced to enhance the accuracy of BNN. Zhang et al. [62] applied quantization for both weights and activations by a learnable quantizer to detect the clipping and representation ranges. While Pham et al. [63] proposed a symmetric quantizer named UniQ that allows learning the step size by using the gradient descent procedure. Another work reduces the quantization error; ReCU [64] revived the dead weights and analyzed their effect through the rectified clamp units.

b: SPECIAL REGULARIZATION

Some research works utilize specific regularization or distribution loss to the global loss function to fine-tune the network parameters in the account of the binarization conditions like in [27], the authors proposed a regularization term that drove the weights to be bipolar. The global loss function can be described as follows:

$$L_G = L_{CL} + \lambda L_{DL} \tag{4}$$

where L_G is the gross loss, L_{CL} is the cross-entropy loss, L_{DL} is the distribution loss, and λ is the parameter which adjusts the regularization term. In [54], the authors applied L2-regularization to clip the scale for activations and parameter λ used for weights in the loss function to provide faster convergence. Similarly, Xu et al. [47] used L2-regularization on weight scaling factors. While Hou et al. [65] utilized a proximal Newton algorithm with diagonal Hessian approximation to reduce the loss due to binary weights. Ding et al. [66] used distribution loss to modify the regularization of the activations and allow differentiability. Additionally, in [62], the authors used a KL-based distributional loss to regularize the output.

BSQ [59] provided a bit-level group Lasso regularizer to optimize the layer-wise weight precision and achieve the mixed-precision quantization schemes. In [67], the authors presented a contrastive loss function for better representation capacity of activations. Besides, Shang et al. [68] used the Lipschitz continuity as a regularization term to enhance the robustness of the model. In addition, the authors proposed an approximate Lipschitz constant instead of calculating its exact value.

Some research work improves the accuracy during the training by employing Knowledge Distillation (KD) that depends on knowledge transfer from the stronger model to the compressed one, which mimics the complex model. The knowledge transfer is learning the class distribution output via the loss function. Therefore, the BNN can be under the supervision of a real-value model to improve the learning capability to gain higher accuracy that is closed to the real-valued model like CI-BCNN [69]. CI-BCNN extracts the channel-wise interactions from the prior knowledge to decrease the inconsistency of signs in binary feature maps and keeps the information of input samples during inference. While in [34], the authors utilized standard logit matching loss to transfer learning between the full-precision and the BNN. Also, LNS [37] introduced a specific loss function to learn weights with noisy supervision. The same idea is used in [70]; the authors presented distilled BNN for monaural speech separation. Another BNN model with knowledge distillation is proposed by Qian et al. [71] for speech recognition. While Bulat et al. [72] applied knowledge distillation for image classification and human pose estimation tasks. Yang et al. [73] proposed a BNN based on the knowledge distillation method to transfer channel-wise mean and variance feature statistics from the real-value model to the BNN model. Also, Huang et al. [74] presented binarizing super-resolution network by knowledge distillation.

c: GRADIENT ERROR MINIMIZATION

The training process is divided into forward and backward propagation. BNN utilizes the STE technique to estimate the gradients in back-propagation as an approximation of sign gradients [26] due to the zero value of the derivative of the sign function. As a result, the gradient values are clipped to the range of $[-1,1]$, which causes degradation in the performance. Various research work attempts to alleviate the gradient errors by using other estimators (i.e., rounding functions) in back-propagation. Bi-Real Net [29] used a piece-wise polynomial function to update the activations and used a magnitude-aware gradient to update the weights. Also, Xu et al. [47] replaced the STE in back-propagation with a higher-order estimator method that utilizes a piece-wise polynomial function. In [75], the authors introduced a circulant back-propagation algorithm to update the circulant filters they used. Another work utilized the Error Decay Estimator (EDE) to reduce the information loss of the gradients during the back-propagation [58]. While in [76], the authors developed an Information Enhanced Estimator (IEE) to aid the binary weights update by gradually approximating the sign function.

Besides, in [77], the training used a weight searching algorithm; the low-bit values of arbitrary weights are kept with different probabilities to reduce the gradient error. Kim et al. [78] used smoothed loss function for better estimation of the gradient using Coordinate Discrete Gradient (CDG). SI-BNN [79] proposed trainable parameters for activations and gradients in the back-propagation. In addition, the Fourier Frequency Domain Approximation (FDA) is used to update the gradients in back-propagation in [80]. Lee et al. [81] introduced an Element-Wise Gradient Scaling (EWGS) to update each gradient element by scaling factor. This scaling factor is controlled by using the Hessian information of a network. In [62], the authors presented the Radical Residual Connection (RRC) technique which enabled the information and gradients to stream through every layer freely and used a tanh-based function for back-propagation to minimize the gradient error.

d: NETWORK STRUCTURE

Due to the binary values of weights and activations used in BNN, the feature maps are lower in quality, causing an accuracy drop. Therefore, some researchers are heading to modify the network architecture to increase the accuracy, such as increasing the number of channels, increasing the number of layers, reordering the layers' position, or adding shortcuts.

Some work attempts to modify the network structure, like ABC-Net [49] presented parallel approximate convolutions; each of them is a linear combination of binary convolutions. Also, Bi-Real Net [29] applied shortcuts to connect the full-precision activations before the sign function. Similarly, BiNeal Net [82] modified the ResNet structure by changing the convolution in the basic block and the skip connection to a binary one. In [83], the authors combined multiple BNN by boosting or bagging. While in [84], the authors presented a new custom architecture for BNN called BinaryDenseNet, which based on the BMXNet framework. BinaryDenseNet is used for image classification and object detection tasks. Besides, J. Bethge et al. [85] presented MeliusNet that utilized a Dense-Block and Improvement-Block for increasing the feature capacity and the feature quality, respectively. MoBiNet [86] is a modified binary model of MobileNetV1 architecture that utilizes a skip connection and uses dependency within channels in a depth-wise convolution layer. Another binary network based on MobileNetV1 is ReActNet [87] which proposed ReAct-Sign (RSign) and ReAct-PReLU (RPReLU) as alternatives of the traditional activation function to reshape the activation distribution. While in [88], the authors proposed Activation Self Distribution (ASD) and Weight Self Distribution (WSD) to adjust the sign distribution of activations and weights, respectively, to enhance the accuracy.

To boost feature expression capabilities, Zhang et al. [90] replaced the static RSign and RPReLU in the ReAct-Net [87] with Dynamic Sign (DySign) and Dynamic PReLU (DyPReLU). Also, in [91], the authors proposed Binarized Ghost Module (BGM) as a modification for the ReActNet [87] to improve the feature maps information. IE-Net [76] augments the information of the activations by utilizing several sign functions with several trainable thresholds to produce various binary input features. While RB-Net [92] presented a reshaped point-wise convolution (RPC) and balanced distribution activation (BA) for a more powerful representative ability. Also, AdaBin [93] provides adaptive binary sets for weights and activations for each layer that centers the position and distance of the distribution of the binary values to real-valued distribution. Besides, INSTA-BNN [94] determines the activation threshold value based on the difference between statistical data generated from a batch and each instance to increase the accuracy. In [95], the authors proposed Binary Contextual Dependencies Net (BCDNet), which provides Contextual Dependencies modeling for BNN through binary multi-layer perceptron block as a substitutional to binary convolution blocks. In [67], the authors provided Contrastive Learning for Mutual Information Maximization (CMIM) to learn representative binary activations by determining the amount of information shared between the binary and full-precision activations.

The batch normalization layer affects the training stability of BNN. For example, HWGQ [52] and [56] used the batch normalization layer before the quantization operations to make the output distribution of each layer near Gaussian with zero mean and unit variance. While Chen et al. [96] removed the batch normalization layer and used the adaptive gradient clipping technique, and scaling factor for weights as an alternative. WRPN [51] raised the number of filters in each channel, and CBCN [75] introduced circulant filters and a circulant binary convolution.

Network Architecture Search (NAS) automatically searches the optimal network architecture, utilizing different methods, including evolutionary algorithms. NAS methods are inappropriate for the BNN due to quantization error and unstable gradients. Therefore, a binary-oriented search space and search strategies for binary networks are proposed in [97], [98], and [99]. In [100], The authors used the evolutionary search for group values at convolutional layers to find the appropriate binary structure to binarize the MobileNet. While Bulat et al. [101] utilized NAS that is developed for real-valued networks. In addition, the authors presented expert binary convolution based on condition computing. Other designs used the evolutionary search algorithm to optimize the number of channels in each layer like [102], and [103]. Besides, Vo et al. [107] presented Deepbit searching algorithm to assess the optimum BNN architecture based on the hardware cost estimation regarding the implementation target platforms. While in [104], the authors used the genetic algorithm for searching for the ideal activation functions for BNN.

2) COMPRESSION OPTIMIZATION APPROACHES

BNN appears to maximize the speed and minimize the size of the deep networks that utilize the full-precision representation to be suitable for resource-constrained devices and edge computing. Some research works tend to reduce the BNN size, but this reduction may be a trade-off with the accuracy. The compression approaches of BNN are classified into two categories: fractional BNN and pruning.

a: FRACTIONAL BNN

To minimize the BNN size, some researchers use fractional bit representations for weights or activations. In [108], the authors introduced FleXOR, a flexible encryption scheme for weight quantization. FleXOR allows fractional quantization bits to represent each weight where the quantization differs from one layer to another in the number of bits. FleXOR is implemented as an XOR-gate in inference time. Another fractional quantization, Sub-bit Neural Networks (SNN) [109] proposed a quantization technique that composed of two steps. The first step is random sampling to produce layer-specific subsets of weights. The second step is the refinement step to optimize these subsets of weights. While Y. Zhang et al. [110] developed FracBNN that employed fractional activations and supported a dual-precision activation up to two bits. FracBNN exploited sparse binary convolution and applied binarization to the input layer using thermometer encoding.

b: PRUNING

For more efficient area, compressed BNN models can be obtained by pruning approaches that remove the redundant parameters. However, there is a trade-off between accuracy and pruning; accuracy may decrease when the pruning rates increase. In [111], the authors utilize Bayesian optimization for channel pruning for quantized neural networks. That pruning approach based on the angle preservation feature of high dimensional binary vectors [112] and the euclidean distance. In [113], the authors proposed neuron pruning for the fully connected layer then, retraining the network. While in [105], the authors introduced a learning-based approach for pruning the number of filters/channels in BNN.

Xiao et al. [118] provided the AutoPrune approach that utilized optimizing a group of learnable parameters by using gradient-based search to prune the network as an alternative to direct pruning of the weights. Also, Li et al. [116] pruned the BNN by weight flipping frequency approach for analyzing the sensitivity of the binary weighs to accuracy. In addition, this design support layer-wise pruning to reduce the number of channels in each layer by the same percentage of the insensitive weights. In [117], O3BNN-R proposed a shrunk BNN model using two irregular pruning for

TABLE 3. Comparative analysis of optimization techniques for model training bases on ImageNet dataset.

Reference	Bit-width (W/A)	Network Architecture	Optimization Techniques	Top-1 Accuracy%	Top-5 Accuracy%
[55]	32/32	AlexNet	-	57.1	80.2
[55]	32/32	ResNet-18	-	69.6	89.2
[55]	32/32	ResNet-34	-	73.3	91.3
[55]	32/32	ResNet-50	-	76.0	93.0
[55]	32/32	VGG	-	72.0	90.5
[55]	32/32	GoogLeNet	-	72.9	91.3
[89]	32/32	MobileNet-v2	-	72.0	NA
BNN [20]	1/1	AlexNet	-	27.9	50.4
XNOR-Net [25]	1/1	AlexNet ResNet-18	Scaling factor to reduce the QE	44.2 51.2	69.2 73.2
DoReFa-Net [44]	1/2 1/1	AlexNet AlexNet	Scaling factor to reduce the QE	49.8 43.6	NA NA
[47]	1/1	AlexNet Resnet-18	Scaling factor to reduce the QE, piece-wise polynomial function to reduce the GE.	46.1 54.2	70.9 77.9
XNOR-Net++ [48]	1/1	ResNet-18 AlexNet	Scaling factor to reduce the QE	57.1 46.9	79.9 71.0
ABC-Net [49]	1/1 1/1 5/5	Resnet-18 Resnet-34 Resnet-50	Modify network structure using parallel approximate convolutions, Scaling factor to reduce the QE.	42.7 52.4 70.1	67.6 76.5 89.7
WRPN [51]	1/1 1/1 2/2 4/2 1/1	AlexNet AlexNet 2x-wide AlexNet 2x-wide AlexNet 2x-wide ResNet-34 1x-wide ResNet-34 2x-wide ResNet-34 3x-wide BN-Inception 2x-wide	Quantization function to reduce the QE	44.2 48.3 55.8 57.3 60.54 69.85 72.38 65.02	NA NA NA NA NA NA NA NA
HWGQ [52]	1/2	AlexNet ResNet-18	Quantization function to reduce the QE	52.7 59.6	76.3 82.2
SYQ [53]	1/8 1/2 1/4 2/2 2/8 1/4 2/4	AlexNet VGG ResNet-18 ResNet-34 ResNet-50 AlexNet AlexNet VGG ResNet-18 ResNet-34 ResNet-50 ResNet-50 	Quantization function and scaling factors to reduce the QE	56.6 66.2 62.9 67.0 70.6 55.4 56.2 55.8 58.1 68.7 67.7 70.8 72.3 68.8 70.9	79.4 87.0 84.6 87.6 89.6 78.6 79.4 79.2 80.8 88.5 87.8 89.8 90.9 88.7 90.2
PACT [54]	2/2 2/3 2/4 1/2 1/3 1/4 1/2 2/2 2/4	AlexNet ResNet-18 ResNet-50 	Quantization function to reduce the QE	55.0 55.4 55.4 62.9 65.3 65.0 67.8 72.2 74.5	77.7 77.9 78.0 84.7 85.9 85.9 87.9 90.5 91.9
LQ-Nets [55]	1/2 2/2	ResNet-18 ResNet-34 ResNet-50 AlexNet GoogLeNet-Variant DenseNet-121	Quantization function to reduce the QE	62.6 66.6 68.7 55.7 65.6 69.6	84.3 86.9 88.4 78.8 86.4 89.1
TSQ [56]	2/2	AlexNet VGG-16	Quantization function to reduce the QE	58.0 69.1	80.5 89.2
IR-Net [58]	1/1	ResNet-18 ResNet-34	Libra-PB Quantization function to reduce the QE, EDE to reducing the GE.	58.1 62.9	80.0 84.1

Note: QE means quantization error, GE means gradient error, and NA means not applicable.

TABLE 3. *(Continued.)* **Comparative analysis of optimization techniques for model training bases on ImageNet dataset.**

Reference	Bit-width (W/A)	Network Architecture	Optimization Techniques	Top-1 Accuracy%	Top-5 Accuracy%
Quantization Network [57]	1/1 1/2 1/1 1/2	AlexNet ResNet-18	Quantization function to reduce the gradient mismatch.	47.9 55.4 53.6 63.4	72.5 78.8 75.3 84.9
DSQ [60]	2/2 3/3 2/2 3/3	ResNet-18 ResNet-34	Quantization function to reduce the QE.	65.17 68.66 70.02 72.54	NA NA NA NA
ProxyBNN [61]	1/1	AlexNet	Proxy matrix to reduce the QE.	51.4	75.5
[35]	1/1 1/1	AlexNet (BNN [20]) AlexNet (XNOR-Net [25]) ResNet-18 (XNOR-Net [25]) ResNet-18 (Bi-Real Net [29]) ResNet-34 (Bi-Real Net [29])	Unbalanced activation distribution.	42.1 45.6 54.2 57.2 62.8	66.6 69.6 77.6 80.2 84.5
UniQ [63]	1/1 2/2 1/1 2/2 1/1 2/2	ResNet-18 ResNet-34 MobileNet-V2	Symmetric quantizer.	60.5 67.8 65.8 72.1 23.2 50.5	NA NA NA NA NA NA
ReCU [64]	1/1	ResNet-18 ResNet-34	Revive the dead weights to reduce the QE	66.4 65.1	86.5 85.8
[67]	1/1	ResNet-18 (BiReal [29]+CMIM) ResNet-18 (IR-Net [58]+CMIM) ResNet-18(ReActNet [87]+CMIM) ResNet-34 (IR-Net + CMIM)	Contrastive Learning for Mutual Information Maximization.	60.1 61.2 71.0 64.9	81.3 83.0 86.3 85.8
SLB [77]	1/1 1/2 1/4 1/8	ResNet-18	Weight searching algorithm to reduce the GE.	61.3 64.8 66.0 66.2	83.1 85.5 86.4 86.5
SI-BNN [79]	1/1	AlexNet ResNet-18	Trainable thresholds in the backward function.	50.5 58.9	74.6 81.3
FDA-BNN [80]	1/1	ResNet-18 AlexNet	Fourier Series to reducing the GE.	60.2 46.2	82.3 69.7
[81]	1/1 1/2 1/1 1/2 4/4	ResNet-18 ResNet-34 MobileNet-V2	Element-wise gradient scaling to reduce the GE.	55.3 64.4 61.5 69.6 70.3	NA NA NA NA NA
Bi-Real Net [29]	1/1	ResNet-18(modified) ResNet-34(modified)	Add shortcuts, used piece-wise polynomial function and magnitude aware gradient in back-propagation.	56.4 62.2	79.5 83.9
Real-to-Bin [34]	1/1	ResNet-18	Use standard logit matching loss to transfer learning between the full-precision and BNN	65.4	86.2
LNS [37]	1/1	ResNet-18	Specific loss function to learn weights.	59.4	81.7
BNN-DL [66]	1/1	AlexNet (BNN [20]) AlexNet (XNOR-Net [25]) AlexNet (DoReFa-Net [44]) AlexNet ([27]) AlexNet (WRPN [51])	Use distribution loss to regularize the activation	41.3 47.8 47.8 47.6 53.8	65.8 71.5 71.5 71.9 77.0
[68]	1/1	ResNet-18 ResNet-34	Utilize Lipschitz Continuity function as a regularization term.	59.6 63.5	81.6 84.6
CI-BCNN [69]	1/1	ResNet-18 ResNet-34	Knowledge transfer to extract the channel-wise interactions.	56.73 62.41	80.12 84.35
[72]	1/1	AlexNet ResNet-18	Progressive quantization, network stacking, knowledge distillation.	48.6 53.7	72.8 76.8
[73]	1/1	ResNet-18	KD by transferring the channel-wise mean and variance feature statistics.	59.87	NA
CBCN [75]	1/1	ResNet-18	Circulant back-propagation algorithm	61.4	NA
IE-Net [76]	1/1	ResNet-18 ResNet-34	Enhance the information of the activations and develop IEE	61.4 64.6	83.0 85.2

TABLE 3. *(Continued.)* **Comparative analysis of optimization techniques for model training bases on ImageNet dataset.**

Reference	Bit-width (W/A)	Network Architecture	Optimization Techniques	Top-1 Accuracy%	Top-5 Accuracy%
BinaryDuo [78]	1/1	AlexNet ResNet-18	Quantitatively calculate the gradient mismatch using Coordinate Discrete Gradient (CDG)	52.7 60.4	76.0 82.3
BiNeal Net [82]	1/1	BiNeal Net (1x-wide) BiNeal Net(1.5x-wide) BiNeal Net(1.75x-wide) BiNeal Net(2x-wide)	Modify the ResNet structure and apply a channel multiplier	65.0 69.7 71.2 72.8	NA NA NA NA
BENN [83]	1/1	AlexNet(BENN-SB-6, Bagging) AlexNet(BENN-SB-6, Boosting) ResNet-18(BENN-SB-6, Bagging) ResNet-18(BENN-SB-6, Boosting)	Ensemble multiple BNN	52.0 54.3 57.9 61.0	NA NA NA NA
BinaryDenseNet [84]	1/1	BinaryDenseNet-45	New architecture for BNN	63.7	84.8
MeliusNet [85]	1/1	MeliusNet-C MeliusNet-42 MeliusNet-59	Modify the structure of the network.	64.1 69.2 71.0	NA NA NA
MoBiNet [86]	1/1	MoBiNet-Mid (K = 3) MoBiNet-Mid (K = 4)	Use skip connection and K-dependency.	53.47 54.40	76.46 77.50
ReActNet [87]	1/1	ReActNet-A ReActNet-B ReActNet-C	Add RSign and RPReLU instead of the traditional activation function.	69.4 70.1 71.4	NA NA NA
SD-BNN [88]	1/1	SD-BNN(Bi-Real Net)	Used self distribution for activations and weights.	66.5	86.7
DyBNN [90]	1/1	DyBNN-ResNet18 DyBNN-ReActNet	Used DySign and DyRPReLU.	67.4 71.2	87.4 89.8
[91]	1/1	Customized	Modify the ReActNet	71.4	NA
RB-Net [92]	1/1	RB-Net (ResNet-18) RB-Net (ResNet-18(2x-wider)) RB-Net (ResNet-34)	Presented a reshaped point-wise convolution and balanced distribution activation.	66.8 70.1 70.2	87.1 89.1 89.2
AdaBin [93]	1/1	AlexNet ResNet-18 ResNet-34 ReActNet-A MeliusNet59	proposed adaptive binary sets for weights and activations for each layer.	53.9 66.4 66.4 70.4 71.6	77.6 86.5 86.6 NA NA
INSTA-BNN [94]	1/1	ResNet-18 MobileNetV1	Determines the activation threshold value based on the statistical data from a batch and each instance	68.0 71.7	87.9 90.3
BCDNet [95]	1/1	Used Contextual Dependencies	BCDNet-A BCDNet-B	71.8 72.3	90.3 90.5
[96]	1/1	ReActNet-18(BN-Free) ReActNet-A(BN-Free)	Remove the Batch norm layer, use adaptive gradient clipping and scaling factors for weights.	61.1 68.0	NA NA
BATS [97]	1/1	BATS BATS(2x-wider)	Search algorithm for BNN architecture.	60.4 66.1	83.0 87.0
BNAS [98]	1/1	BNAS-D BNAS-E BNAS-F BNAS-G BNAS-H	Search algorithm for BNN architecture.	57.69 58.76 58.99 59.81 63.51	79.89 80.61 80.85 81.61 83.91
NASB [99]	1/1	NASB (ResNet-18) NASB (ResNet-34) NASB (ResNet-50)	Search algorithm for BNN architecture.	60.5 64.0 65.7	82.2 84.7 85.8
[100]	1/1	Customized	Use evolutionary search to binarize the MobileNet.	60.90	82.60
[101]	1/1	Customized	Search algorithm for BNN architecture and add condition computing for convolution.	71.2	90.1
[102]	1/1	ResNet-18(modified)	Search algorithm to adjust the number of channels in each layer	69.65	89.08
DMS [103]	1/1	ResNet-18(DMS-A) ResNet-18(DMS-B)	Search algorithm to adjust the number of channels in each layer	60.20 67.93	82.94 87.84
[104]	1/1	ResNet-18 NIN-E	Use the genetic algorithm to search for the ideal activation functions	55.514 52.270	78.556 75.822

TABLE 4. Comparative analysis of optimization techniques for model training based on CIFAR-10 dataset.

Reference	Bit-width (W/A)	Network Architecture	Optimization Techniques	Top-1 Accuracy%
[55] [105] [58] [58]	32/32 32/32 32/32 32/32	VGG-small VGG-11 ResNet-18 ResNet-20	- - - -	93.8 83.8 93.0 91.7
BNN [20]	1/1	VGG-small	-	89.9
XNOR-Net [25]	1/1	VGG-Small	Scaling factor to reduce the QE.	89.8
HORQ [46]	1/1	Customized	Scaling factor to reduce the QE.	82
[47]	1/1	VGG-Small	Scaling factor to reduce the QE, piece-wise polynomial function to reduce the GE	92.3
LQ-Nets [55]	1/2 2/2 1/2 2/2	VGG-Small ResNet-20	Quantization function to reduce the QE	93.4 93.5 88.4 90.2
TSQ [56]	2/2	VGG-small	Quantization function to reduce the QE	93.4
IR-Net [58]	1/1	ResNet-18 ResNet-20 VGG-Small	Libra-PB Quantization function to reduce the QE, EDE to reducing the GE.	91.5 86.5 90.4
DSQ [60]	1/1	VGG-Small ResNet-20	Quantization function to reduce the QE	91.72 84.11
ReCU [64]	1/1	ResNet-18 ResNet-20 VGG-small	Revive the dead weights to reduce the QE	92.8 87.4 92.2
LNS [37]	1/1	ResNet-20	Specific loss function to learn weights	85.78
LAB [65]	1/1	VGG-Small	Use proximal Newton algorithm with diagonal Hessian approximation to reduce the information loss	87.72
BNN-DL [66]	1/1	VGG-Small ResNet-18	Use distribution loss to regularize the activation	89.62 90.47
[67]	1/1	ResNet-18 (IR-Net [58]+CMIM) ResNet-20 (IR-Net [58]+CMIM) VGG-small (IR-Net + CMIM)	Contrastive Learning for Mutual Information Maximization.	92.2 87.3 92.0
[68]	1/1	ResNet-18 ResNet-20	Utilize Lipschitz Continuity function as a regularization term.	91.8 86.0
CI-BCNN [69]	1/1	VGG-small ResNet-20	KD to extract the channel-wise interactions	92.47 91.10
[73]	1/1	ResNet-18	KD by transferring the channel-wise mean and variance feature statistics	93.92
AdaBin [93]	1/1	ResNet-18 ResNet-20 VGG-Small	proposed adaptive binary sets for weights and activations for each layer.	93.1 88.2 92.3
IE-Net [76]	1/1	ResNet-18 ResNet-34 VGG-Small	Enhance the information of the activations and develop IEE	92.9 88.5 92.0
SLB [77]	1/1 1/2 1/4 1/1 1/2 1/4	ResNet20 VGG-Small	Weight searching algorithm to reduce the GE	85.5 89.5 90.3 92.0 93.4 93.5
SI-BNN [79]	1/1	VGG-small	Trainable thresholds in the backward function	90.2
[81]	1/1	ResNet-20	Element-wise gradient scaling to reduce the GE	85.6
BinaryDenseNet [84]	1/1	BinaryDenseNet-21	New architecture for BNN	90.3
SD-BNN [88]	1/1	VGG-small ResNet-20 ResNet-18	Used self distribution for activations and weights.	90.8 86.9 92.5
[91]	1/1	Customized	Modify the ReActNet	86.45
RB-Net [92]	1/1	VGG-16 AlexNet	Presented a reshaped point-wise convolution and balanced distribution activation	74.9 56.2

TABLE 4. *(Continued.)* **Comparative analysis of optimization techniques for model training based on CIFAR-10 dataset.**

Reference	Bit-width (W/A)	Network Architecture	Optimization Techniques	Top-1 Accuracy%
INSTA-BNN [94]	1/1	ResNet-20	Determines the activation threshold value based on the statistical data from a batch and each instance	87.32
[96]	1/1	ReActNet-18(BN-Free) ReActNet-A(BN-Free)	Use adaptive gradient clipping and scaling factors for weights	92.08 83.91
[97]	1/1	BATS BATS+AutoAugment	Search algorithm for BNN architecture	95.5 96.1
BNAS [98]	1/1	BNAS-A BNAS-B BNAS-C	Search algorithm for BNN architecture	92.70 93.76 94.43
[102]	1/1	VGG-small(modified)	Search algorithm to adjust the number of chanels in each layer	93.06
DMS [103]	1/1	ResNet-18(DMS-A) ResNet-18(DMS-B) VGG-11(DMS-A) VGG-11(DMS-B)	Search algorithm to adjust the number of channels in each layer	89.32 92.70 84.16 89.10
[104]	1/1	ResNet-18 NIN ResNet-34	Use the genetic algorithm to search for the ideal activation functions	91.40 87.48 92.20
RBNN [106]	1/1	VGG ResNet-18 ReActNet	Train tasks from different areas such as vision and audio.	87.49 86.69 86.81

redundant edges during the inference, one for threshold edge pruning and the other for pooling edge pruning. Besides, Gao et al. [114] exploited the idea of reusing the calculated partial outputs of the duplicated filters in one tile of one neuron to prune the redundant operations in BNN. Wu et al. [115] provided the Slimming Binarized Neural Network (SBNN) that utilizes two compression techniques filter pruning and knowledge distillation.

3) ACCELERATION OPTIMIZATION APPROACHES

Acceleration techniques provide parallelized computations and highly pipelined data flow to improve the latency and throughput performance. The BNN accelerator hardware implementations are categorized into three classes: Computing in memory, FPGA, and ASIC implementations.

a: COMPUTING IN MEMORY

Since data transfer between memory and processors exhaust much energy and accessing time, Some researchers utilize Computing In-Memory (CIM) for more efficient energy and accelerating the inference of BNN and increasing the throughput like in [119], the authors exploited CIM using a 9-transistor Static Random Access Memory (9T-SRAM) to implement binarized VGG-16 on the CIFAR-10 dataset. Agrawal et al. [120] provided Xcel-RAM that allowed in-memory computing. Xcel-RAM divided the SRAM to improve the parallelism of convolutional computing. In addition, the authors proposed two models, the first one utilized charge-sharing between the parasitic capacitance in the standard 10T-SRAM array to perform an approximate pop-count operation, and the second model, modify the SRAM peripheral circuitry to compute accurate pop-count operation. Also, Bankman et al. [121] provided a mixed-signal processor for BNN. The design exploited the switched-capacitor neuron to enhance energy efficiency. Besides, Valavi et al. [122] proposed a high signal to noise ratio for charged-domain mixed-signal CIM with 64 tiles and the charge-domain computation based on metal-oxide-metal capacitors. In [123], the authors implemented an SRAM-CIM unit-macro that supports a binary fully connected neural network using a 6T-SRAM bit-cell. This design used circuit techniques to decrease power consumption, such as a dynamic input-aware reference generation scheme, an algorithm-dependent asymmetric control scheme, a write disturb-free scheme, and a common-mode-insensitive small offset voltage-mode sensing amplifier. Similarly, in [124], the authors proposed XNOR-SRAM that based on CIM. This design support binary weights but ternary activation. In [125], the authors used CIM based on pulse-width modulation. This design used binary weights for AlexNet architecture. While in [126], the authors proposed a CIM-based accelerator that uses two different filter sizes. This design supports near-threshold voltage operation down to 0.4V for more efficient energy. Song et al. [127] merged time-domain (TD) computing with CIM. The authors implement TD computing employing a dual-edge single-input cell topology. Besides, Kushwaha et al. [128] proposed XNOR and accumulation scheme based on CIM-SRAM utilizing a 10-transistor 1-capacitor (10T1C) XNOR bit-cell to achieve a high compute signal to noise ratio and efficient energy design. While in [129], the authors applied 8T2C SRAM cells to consume less energy and avoid the problems of static current. Table 7 shows the results of the CIM based on SRAM schemes.

Another direction is implementing the CIM using the memristor for the BNN inference accelerator because of its low operating voltage and small cell area. In [148], the authors proposed multilevel cell spin-torque transfer

TABLE 5. Comparison between BNN and fractional BNN.

Dataset	Reference	Bit-width (W/A)	Network Architecture	BOP (G)	Top-1 Accuracy%	Top-5 Accuracy%
ImageNet	IR-Net [58]	1/32 1/1	ResNet-18	53.64 [109] 1.677 [109]	66.5 58.1	NA NA
		1/32 1/1	ResNet-34	112.83 [109] 3.526 [109]	70.4 62.9	NA NA
	FleXOR [108]	0.8/32 0.63/32 0.6/32	ResNet-18	NA	63.8 63.3 62.0	84.8 84.5 83.7
	SNN [109]	32/32 32/32	ResNet-18 ResNet-34	- -	69.6 73.3	NA NA
		0.67/1 0.56/1 0.44/1	ResNet-18	0.883 0.501 0.297	56.3 55.1 53.0	NA NA NA
		0.67/32 0.56/32 0.44/32	ResNet-18	28.26 16.03 9.504	64.7 63.4 60.9	NA NA NA
		0.67/1 0.56/1 0.44/1	ResNet-34	1.696 0.965 0.581	61.4 60.2 58.6	NA NA NA
		0.67/32 0.56/32 0.44/32	ResNet-34	54.27 30.88 18.59	68.0 66.9 65.1	NA NA NA
	FracBNN [110]	1/1.4	Customized	7.30	71.8	90.1
CIFAR-10	IR-Net [58]	1/32 1/1	ResNet-18	17.52 [109] 0.547 [109]	92.9 91.5	NA NA
		1/32 1/1	ResNet-20	1.283 [109] 0.040 [109]	90.8 86.5	NA NA
		1/32 1/1	VGG-small	19.30 [109] 0.603 [109]	92.5 91.3	NA NA
	FleXOR [108]	0.8/32 0.6/32	ResNet-32	NA	∼91 ∼90	NA NA
	SNN [109]	32/32 32/32 32/32	ResNet-18 ResNet-20 VGG-small	- - -	93.0 92.5	NA NA NA
		0.67/1 0.56/1 0.44/1	ResNet-18	0.289 0.164 0.097	91.0 90.6 90.1	NA NA NA
		0.67/32 0.56/32 0.44/32	ResNet-18	9.236 5.239 3.106	92.7 92.3 91.9	NA NA NA
		0.67/1 0.56/1 0.44/1	ResNet-20	0.040 0.034 0.025	85.1 84.0 82.5	NA NA NA
		0.67/32 0.56/32 0.44/32	ResNet-20	1.283 1.099 0.822	90.0 88.9 87.6	NA NA NA
		0.67/1 0.56/1 0.44/1	VGG-small	0.194 0.113 0.074	91.0 90.6 90.0	NA NA NA
		0.67/32 0.56/32 0.44/32	VGG-small	6.208 3.616 2.368	92.4 92.1 91.9	NA NA NA
	FracBNN [110]	1/1.4	Customized	0.0715	89.1	NA

Note: BOP means Binary OPerations.

magnetic RAM (STT-MRAM) for the XNOR-Net architecture on the MNIST dataset with low power consumption. Li et al. [149] proposed in-situ learning using a memristor. In [150], the authors provided an analog hybrid CMOS-memristive design for back-propagation learning with a sign control circuit and weight update unit. The design can be adaptive with different neuromorphic architectures. Another Memristor-CMOS design is presented by Van Pham et al. [140] that supports single and double column architectures. Furthermore, they introduced two activation functions: ReLU and sigmoid. Memristor is sensitive to its initial conditions; therefore, Yi et al. [151] studied the effect of the On/Off resistance ratio and the memristor devices' changes on the reading sense and inference accuracy of BNN.

In [141], the authors programmed the memristor by an asymmetrical coarse-programmed for the high resistance state and fine-programmed for the low resistance state during the crossbar training. This scheme based on the

TABLE 6. Comparison between pruning approaches for BNN.

Dataset	Reference	Network Architecture	Accuracy%	Pruning rate %	Pruning of operations %
MNIST	[114]	LeNet-5	98.4	NA	57
	[115]	Customized	98.82	50	NA
CIFAR-10	[105]	NIN	83.11	33.05	NA
		VGG-11	81.97	39.7	NA
		ResNet-18	86.39	39.89	NA
	[111]	VGG-11(BNN [20])	82.31	43.82	NA
	[113]	VGG-11	81.2	39.8	NA
	[116]	NIN	86	NA	20
		AlexNet	85.5	NA	40
	[117]	VGG-like	88.5	NA	42
		VGG-like(with regularization)	85.2	NA	48
	[114]	VGG-like	88.7	NA	51
	[115]	BNN([20])	90.40	50	NA
		ResNet-18	89.31	50	NA
ImageNet	[105]	ResNet-18	50.13	21.592	NA
	[111]	ResNet-18(XNOR-Net [25])	49.48	25.5	NA
	[117]	VGG-16	74.3	NA	27
		VGG-16(with regularization)	71.4	NA	49
		AlexNet	72.7	NA	19
		AlexNet (with regularization)	71.8	NA	43
	[115]	ResNet-18	51.98	25	NA

TABLE 7. Comparison of CIM implementations based on SRAM schemes.

Precision (bit)	Ref.	Dataset	Technology	Supply Voltage V_{max} (V)	On-chip memory (KB)	Performance (GOPS/mm^2)	Energy Efficiency (TOPS/W)	Accuracy%
INT16	[130]	MNIST/ImageNet	65nm	0.8	16	25.2	2.06	99.2/92.7
INT8	[131]	MNIST/CIFAR-10/ImageNet	45nm	0.4	NA	NA	105	98.6/90.2/77.3
INT8	[132]	ImageNet	7nm	NA	NA	NA	6.02	NA
INT8	[133]	CIFAR-10	28nm	0.9	64	NA	16.63	92.02
INT8	[134]	CIFAR-10	28nm	1	64	NA	7.6/7	91.94
INT(1-8)	[135]	CIFAR-10	65nm	1.2	72	600	192	92.4*
1-bit	[121]	CIFAR-10	28nm	0.6	328	913	772	86
	[122]	MNIST/CIFAR-10/SVHN	65nm	0.94	295	1498	866	98.92/83.50/95.10
	[123]	MNIST	65nm	1.2	4	33130	55.8	97.5
	[124]	MNIST/CIFAR-10	65 nm	0.6	16	5461	403	98.84 /88.78
	[125]	ImageNet	28nm	0.6	NA	NA	46.6	NA
	[126]	MNIST/CIFAR-10/SVHN	55nm	0.4	216	913	5526	97.73/ 82.56/ 92.61
	[127]	MNIST	40nm	0.9	8	NA	537	98.0
	[129]	MNIST/CIFAR-10	28nm	0.7	NA	NA	3182	97.37/81.17

Note: INT8 indicates 8-bit integer, INT(1-8) means variable bit precision from 1- to 8-bit.
* This accuracy for 4-bit precision.

incremental pulse steps. In [142], the authors presented an architecture based on a 1-transistor 1-digital memristor (1T1DM). Besides, Hirtzlin et al. [143] used hafnium oxide RRAM in a 2-transistor 2-resistor cell. This design supports bit-errors reduction. In [152], the authors presented a BNN accelerator with a two-column reference memristor structure to map $+1$ and -1 weights on the memristor array and remove the sneak current effect. While Qin et al. [144] used a W/AlOx/Al2O3/Pt memristor with a column architecture. Chen et al. [153] proposed a memristor crossbar design that converts the weights and the feature maps before the convolution process, which guarantees a constant sign of the input voltage of each port in the crossbar. Ahn et al. [145] suggested a technique to increase the parallel activated word-line based on magnetic-RAM. This design supports a retraining method that relies on knowledge distillation to be robust against the memristor device variations. In addition, Huang et al. [146] introduced configurable architecture that utilizes binary inputs, weights, and neurons. This design supports two neuron cases one of them is $\{-1, +1\}$ and the other is 0, 1. While Kingra et al. [147] proposed dual-configuration CIM based on 2T-2R XNOR bitcell with MobileNet architecture for BNN. Also, Parmar et al. [154] presented stochastic sampling CIM based on OxRAM circuit. In [155], the authors utilized a 1-selector 1-resistor architecture instead of a 1-transistor 1-resistor array of the memristor structure to obtain high speed, and high-density integration. While in [156], the authors used a 3D-memristor structure for low power consumption. The summary of the aforementioned works is tabulated in Table 8.

TABLE 8. Comparison of CIM implementations based on Memristor schemes.

Precision (bit)	Ref.	Dataset	SET/RESET Voltage (V)	HRS/LRS ratio	Max. Current (mA)	Accuracy%	Memristor Amount	Power (W)
FL-P32 FL-P16 FX-P32 FX-P16	[136]	ImageNet	2/1	1000	NA	84.4 84.4 81.5 78.6	NA	62.6
FX-P3	[137]	MNIST	2/-2	20	NA	99	NA	NA
INT4*	[138]	MNIST/ CIFAR-10	1.2/1.5	NA	NA	99/85.7	NA	NA
INT8	[139]	MNIST	1.8/4.7	NA	NA	96	NA	0.007
1-bit	[140]	MNIST	2.3/-1.4	100	$\sim$10	96.1	NA	NA
	[141]	MNIST	3/-4	50	10	91.7	NA	NA
	[142]	MNIST	2 to 5/-1 to -2	104	NA	89.34	NA	4.07
	[143]	MNIST/ CIFAR-10/ ImageNet/ ECG	3.3/2.5	NA	20	98.1/86.9 /45 /78.7	NA	NA
	[144]	MNIST	1.4/-1.8	$\geq$1000	$\sim$1	98.3	NA	NA
	[145]	CIFAR-10/ ImageNet	NA	2	NA	89.72/56.82	NA	NA
	[146]	MNIST	1.5/-2	NA	NA	98.2	1,164,800	1.666
	[147]	CIFAR-10	1/-1	NA	$\sim$10	84.9	NA	NA

Note: FL-Px indicates floating point representation with x bit, FX-Px indicates fixed-point representation with x bit.
*This precision is for weights, and the precision of activations is a 3-bit fixed-point.

b: FPGA-BASED IMPLEMENTATIONS

FPGAs have various advantages like re-programmability and parallel data processing. FPGAs are more power-efficient than GPUs. There are several research works performed to accelerate the BNN using the FPGA. Since BNN uses bit-wise operations, that make BNN convenient for FPGA implementation. Nurvitadhi et al. [157] implemented a BNN using binary weights and activations on Aria 10 FPGA and 14 nm ASIC using Verilog code. In addition, they compared them against optimized software on Xeon server CPU, Nvidia Titan X server GPU, and Nvidia TX1 mobile GPU. Liang et al. [158] presented FP-BNN that provides a Resource-Aware Model Analysis (RAMA) method to evaluate the FPGA's resources cost to decide the storage place of the model's parameters using on-chip BRAM for small models and off-chip for the large model. Moreover, they utilized an XNOR model and a pop-count compressor tree for binary multiplication and accumulation processes for convolution and fully connected layers. In order to permit parallel accessing and storing the parameters, [159] used N-RAM blocks, also this design supports a reconfigurable-size convolutional filter. Guo et al. [160] utilized a binary input layer and represented the padding with an odd-even scheme. In [161], the authors used the software implementation of the BNN [20] to build binary MLP for binary classification for three applications imaging, cybersecurity, and high-energy physics. After training the network, the Verilog code of the combinational BNN is generated for FPGA implementation. To enhance the accuracy of the BNN, He et al. [162] introduced Ensemble Binarized DroNet (EBDN) that is a perception-control integrated deep binary DroNet model [163]. The EBDN utilized ensemble learning to decrease the error due to binarization. While Ling et al. [164] enhance the speed by pipelining and sparse local aggregation techniques for local stereo matching accelerator using BNN. Also, Skrimponis et al. [165] enhance the throughput using dynamic partial reconfiguration applied to a BNN remote-accelerator for disaggregated computing. In addition, Cho et al. [166] presented an adaptive parallelism design to enhance the throughput. Besides, in [167], the authors used the K-mean cluster method to reuse the weights for parallel computation and reduce the pop-count operation complexity. In [168], the authors presented channel amplitude and adaptive spatial amplitude models to enhance the BNN computation speed.

On the other hand, there is another FPGA-based implementation using High-Level Synthesis (HLS) where the BNN is designed in C/C++ language and synthesized to generate the HDL code for the neural network layers to the target FPGA [169], [170], [171], [172], [173], [174]. In order to increase the throughput, Zhao et al. [175] used variable-length buffers. While in [176], utilized pipelining and two binary parallel convolution layers instead of one to enhance the throughput. Also, In [177], the authors developed LBCNN for AlexNet architecture and replaced the convolutional layer with two sublayers. The first sublayer had ternary weights, and the second sublayer was 1×1 convolution. Based on the FINN [169] framework, BinaryEye [178] classifies regions of interest within a frame using an integrated streaming camera system. Also, based on the Matrix-Vector Threshold unit (MVTU) proposed in FINN [169], Faraone et al. [53] presented a symmetric quantizer. Besides, Zhang et al. [110] implemented FracBNN that utilized fractional bit representation for activations. Table 9 shows the summary of the FPGA-based implementations.

c: ASIC-BASED IMPLEMENTATION

The Application-Specific Integrated Circuit (ASIC) hardware implementations are energy efficient and have high performance. Therefore, some researchers proposed ASIC-based BNN accelerators. In [203], YodaNN provided an ASIC design implementation for the BinaryConnect with some

TABLE 9. Comparison of FPGA-based Implementations.

Precision (bit)	Dataset	Ref.	Target FPGA	Top-1 Accu.	Top-5 Accu.	Frequency (MHZ)	Throughput		Power (W)	LUT	BRAM	DSP
							FPS	GOP/s				
FL-P32	MNIST	[179]	Xilinx XCZU7EV	96	NA	100	NA	NA	0.67	169,143	304	12
FX-P18	MNIST	[180]	Cyclone10	97.57	NA	150	NA	NA	NA	12588	NA	274
FX-P11	MNIST	[181]	Virtex7	96.8	NA	150	NA	NA	NA	80175	0	83
FL-P32	CIFAR-10	[182]	Xilinx(ZCU102)	64.82	NA	100	NA	28.15	6.89	NA	324	1315
FL-P16	CIFAR-10	[183]	Intel Stratix-10	∼ 85	NA	185	NA	∼ 180	20	239k	2558	1040
INT8	CIFAR-10	[184]	UltraScale+ XCVU9P	NA	NA	200	386.7	-	13.5	480k	NA	4202
FL-P32	ImageNet	[182]	Xilinx(ZCU102)	NA	NA	100	NA	46.99	7.712	NA	787	1508
FX-P16	ImageNet	[185]	Virtex-7 (VC709)	NA	NA	156	391	565.94	30.2	273805	1913	2144
FX-P (8-16)	ImageNet	[186]	Stratix-V (GSD8)	66.58	87.48	120	NA	117.8	19.1	NA	1,439	164
FL-P8	ImageNet	[187]	Xilinx VC709	68.31	NA	200	-	760.83	9.18	231761	913	1027
1-bit	MNIST	[158]	Stratix-V(5SGSD8)	98.24	NA	150	-	5905.40	26.2	182301	2210	20
		[161]	Zynq 7000 Zedboard	96.13	NA	NA	NA	NA	1.47	44670	NA	NA
		[162]	Zynq-7000(XC7Z100)	97.7	NA	200	-	439.1	2.11	43K	286	12
		[114]	Zynq-7000(XC7Z100)	88.7	NA	450	-	6921.97	1.72	NA	NA	NA
		[167]	Ultra96	98.4	NA	300	-	18330	1.795	26780	0	0
				97.7	NA	300	-	7647	0.977	14361	0	0
		[169]	Zynq-7000(Z7045)	98.40	NA	200	1561 k	-	22.6	82988	396	NA
		[170]	Ultra96	97.69	NA	300	-	5,110	11.8	38,205	417	NA
			PYNQ-Z1		NA	100	-	974	2.5	25,358	220	NA
		[171]	Spartan XC7S50	98.25	NA	200	NA	NA	NA	24124	88.8	150
		[178]	XC7K325T	98.40	NA	NA	-	116	13.8	88k	124	NA
	SVHN	[160]	Xilinx ZC702	96.9	NA	NA	NA	NA	3.2	29.6k	103	NA
		[169]	Zynq-7000(Z7045)	94.90	NA	200	21.9 k	-	11.7	46253	186	NA
		[171]	Zynq ZC702	97.00	NA	200	NA	NA	NA	53200	280	165
		[174]	Zynq-7020	97.00	NA	142.85	5310	-	3.5	27,342	94	NA
	CIFAR-10	[110]	Zynq ZU3EG	89.1	NA	250	2806.9	-	4.1	51444	212	126
		[113]	Zedboard (XC7Z020)	81.8	NA	143	408	-	2.2	15680	64	0
		[114]	Zynq-7000(XC7Z100)	88.7	NA	200	-	9685.04	14.89	NA	NA	NA
		[158]	Stratix-V(5SGSD8)	86.31	NA	150	-	9396.41	26.2	219010	2210	20
		[159]	Virtex-7 VXT458t	91.79	NA	200	-	2100	28	232000	832	2352
		[160]	Xilinx ZC702	88.61	NA	NA	NA	NA	3.3	29.6k	103	NA
		[165]	ZynqMP (XCZU9EG)	NA	NA	150	-	667	5.97	29,249	122	4
		[167]	Ultra96	80.2	NA	210	205 k	-	NA	290012	NA	NA
		[169]	Zynq-7000(Z7045)	80.10	NA	200	21.9 k	-	11.7	46253	186	NA
		[171]	Zynq ZC702	86.98	NA	200	NA	NA	NA	53200	280	165
		[173]	Zynq 7Z100	87.15	NA	167	18069	-	5.5	78.2K	603	291
		[174]	Zynq-7020	91.50	NA	142.85	537	-	4.4	37,286	130	NA
		[175]	Zynq 7Z020	87.73	NA	143	NA	NA	4.7	46900	94	3
		[176]	Xilinx PYNQ Z1	86	NA	143	930	-	2.4	23436	135	53
		[188]	Virtex7 (XC7V690T)	NA	NA	450	NA	NA	15.44	NA	372 18K	0
		[189]	Virtex-7 980T	86.06	NA	340.13	NA	NA	NA	NA	NA	NA
	ImageNet	[53]	Xilinx ZU3 FPGA	55.4	NA	300	NA	NA	NA	70.6K	432	360
		[110]	Zynq ZU3EG	71.8	90.1	250	48.1	-	6.1	50656	201	224
		[158]	Stratix-V(5SGSD8)	42.90	66.80	150	-	1963.96	26.2	230918	2210	384
		[166]	ZYNQ (XCZU7EV)	75.5	NA	371	-	177.68	∼ 0.71	4.8K	89	2
		[171]	Virtex VCU108	41.43	NA	200	NA	NA	NA	53760	3041	768
	UNSWNB15 [190]	[161]	Zynq 7000 Zedboard	∼90	NA	NA	NA	NA	1.568	51353	NA	NA
	SUSY [191]	[161]	Zynq 7000 Zedboard	72.18	NA	NA	NA	NA	0.68	19140	NA	NA
	PAMAP2 [192]	[193]	Xilinx Artix-7	99	NA	0.315	0.72	-	0.68	5988	1	0
	Customized	[194]	Xilinx ZCU102	NA	NA	NA	41.1	-	3.1	NA	NA	NA
	Customized	[195]	Virtex-7 (XC7VX485T)	98	NA	200	NA	NA	16.15	27,631	75	NA

modifications for more energy-efficient implementation. YodaNN used the two's complement and multiplexers as an alternative to the multiplications. Moreover, the authors utilized Latch-based Standard Cell Memory (SCM) architecture instead of SRAM to store the images. The design provided three filter sizes 3×3, 5×5, and 7×7, to enhance module flexibility. Also, Wang et al. [204] presented a binary weights CNN that used a compensation scheme for the binary multiplication process. Furthermore, the authors reduced the computation complexity by exploiting early

TABLE 10. Comparison of ASIC-based implementations.

Precision (bit)	Reference	Dataset	Technology	Area (mm^2)	Frequency (MHZ)	Power (mW)	Performance		Accuracy%
							GOP/s	FPS	
FL-P32	[196]	MNIST CIFAR-10	65nm 65nm	∼60 ∼500	NA NA	NA NA	NA NA	NA NA	99.25 89.06
FX-P16	[197]	MNIST	28nm	5.76	1200	63.5	NA	NA	98.36
INT8	[198]	NA	28nm	331	700	40000	92000	-	NA
FX-P(1-16)	[199]	ImageNet	40nm	2.4	204	76	-	47	NA
FX-P(1-16)	[200]	ImageNet	65nm	16	200	297	-	18.3	NA
FX-P16	[201]	ImageNet	65nm	1.77 3.5	400 200	254 260	- -	2.29 2.21	68
FX-P10	[202]	ImageNet	65nm	16	60	52	-	26.3	NA
1-bit	[166]	ImageNet	40nm	0.016	300	NA	74	-	75.5
	[195]	Customized infrared images	28nm	0.26	200	33.03	NA	NA	98
	[203]	MNIST/ CIFAR-10/SVHN	65nm	3.11	400	153	1510	-	NA
	[204]	CIFAR-10	130nm 65nm	44.92 11.23	190 380	768.7 842.6	3501 7002	- -	84.87
	[205]	MIO-TCD [206]	22nm	2.61	NA	NA	-	50M	64.7
	[207]	CIFAR-10	22nm	2.32	492	8.76	108	-	84
	[208]	INRIA+CIFAR-10	32nm	3.38	500	410	8191.8		96.5
	[209]	MNIST Yale [210]	55nm	0.421	0.001 0.0008	51.45	NA NA	NA NA	98.0 93.3
	[211]	CIFAR-10	10nm	0.39	13 622	5.6 607	NA NA	NA NA	86
	[212]	ImageNet	65nm	0.54	476 156	56.2 9.9	746 244	- -	NA

Note: FX-P(1-16) means fixed-point representation with variable bit precision from 1 to 16 bit.

pooling, activations quantization, approximate adders, and compressor tree. Besides, in [205], the authors implemented combinational BNN for low power near sensor processing. The authors utilized two models with 16 × 16 and 32 × 32 binary inputs with variable and fixed weights. Conti et al. [207] provided XNOR Neural Engine (XNE). It is a hardware accelerator IP for BNN. XNE is connected with the microcontroller unit, memory, and I/O subsystems. Another system implementation of the BNN chip supports AMBA stream bus interfaces for Advanced Driver Assistance System (ADAS) applications to detect pedestrians and cars [208]. This design is evaluated by pedestrian detection from INRIA dataset [213] and car detection from the CIFAR-10 dataset. STBNN [209] is a spiking neuron model for BNN inference that utilizes binary inputs and weights and replaces the multiplication process with a 1-bit Signed AND operation. While in [211], the authors used the computing near memory to decrease the cost of the data movement and near-threshold voltage to decrease the sequential elements' overhead. Table 10 shows the summary of the ASIC-based implementations.

4) OTHER TRAINING METHODS

Another training direction for low memory BNN that is suitable for on-edge devices is proposed in [36] that introduced a low memory and low energy training. Laydevant et al. [38] introduced training the BNN utilizing Equilibrium Propagation (EP) that provides the ability of on-chip training. Cai [214] proposed Tiny-Transfer-Learning (TinyTL) that adjusts the weights of the pre-trained models by using the newly gathered data.

Another research attempts to perform on-chip training. Yu et al. [215] utilize the 16 Mb RRAM macro chip to implement the multilayer perceptron algorithm used for the MNIST. The authors used the binary implementation for the classifications and 8 bits for online training to update the parameters. The authors reported accuracy of ∼96.5%. Besides, Koo et al. [216] introduced stochastic Binary Spiking Neural Network (SBSNN) that based on the Spike Timing Dependent Plasticity (STDP) to build energy-efficient on-chip neuromorphic systems. SBSNN utilized 'stochastic bit' to realize the stochastic neurons and binary synapses for training and deterministic ones for inference.

E. RQ5: WHAT ARE THE TYPES OF THE BNN FRAMEWORKS?

BNN uses binary weights and activations; thus, these binary values require special tools and optimization strategies. Although the BNN models are implemented in Python platforms like TensorFlow [217] and Pytorch [218], these platforms do not support storing the data of the model in binary format, so the BNN frameworks appear to solve this problem and also permit users to move smoothly from training to deployment. Some of the existing frameworks are FPGA-based, and others are not.

1) FPGA-BASED FRAMEWORKS

FPGA-based frameworks apply hardware-software co-design that begin with the training phase of the BNN using Python code to obtain the learned parameters of the trained network. Then, design the hardware layers in C/C++ code. The next step is to use Vivado HLS [219] for generating the HDL code of the desired network to be implemented on FPGA. Then, use Vivado design suite [220] to obtain the bitstream file. Examples of these frameworks are discussed in the following lines.

Umuroglu et al. [169] developed the FINN end-to-end framework. This framework supports user selection of the desired throughput. The authors utilize the Theano library for the training stage inspired by BNN [20]. Then, FINN uses C++ code to build the hardware components of the required network. FINN framework proposed optimization on the standard BNN architecture like combining the Batch-Normalization and the activation layers to a threshold. This threshold is compared to the output of the pop-count; if the threshold is greater than the output of the pop-count, then, the output is 1; else, the output is 0. As a result, the Max-Pooling layer becomes Boolean OR-Gate. FINN supports parallelism in matrix multiplication in both convolution and fully connected layers by using a Matrix-Vector Threshold unit (MVTU) controlled by the user's input throughput. The results showed that the FINN framework outperforms its predecessors (which are mentioned in [169]) in the throughput. As an extension of FINN, Blott et al. [170] presented FINN-R that ables to work with BNN besides multi-bit quantized networks. In addition, the design supports pruning to eliminate unimportant operations.

Ghasemzadeh et al. [171] provided another end-to-end framework named ReBNet, which used Keras library [221] for the training phase then, used Vivado HLS to generate the hardware code. The ReBNet framework follows the same outlines of the FINN framework [169] but with some modifications. The first modification, ReBNet utilizes 1-bit weights and multi-level residual binarization for the activation layer. It trains the BNN employing a residual binarization scheme that allows a specific number of levels. The residual binarization scheme sequentially binarizes the residual errors to raise the approximation's precision. As the residual binarization level increase, that leads to higher area and latency for ReBNet. The second modification is multiplying the corresponding scaling factors to the accumulated pop-count. These two modifications, as mentioned earlier, cause building the max-pooling unit as comparators, not OR gates.

Other researchers proposed an end-to-end framework for FPGA implementation called LUTNet [172] that uses software-hardware co-design. LUTNet starts from TensorFlow software to perform the first three steps. The first step is training using high-precision values in forward and backward propagation. The second step is fine-grained pruning to suppress the unimportant weights. The third step is binarization using an approximation of linear combinations of multiple binary values accomplished by residual binarization. LUTNet replaces the XNOR gates in BNN with K-LUT to execute Boolean operations. In addition, it supports network tiling to share data between operations. The inference K-LUTs is written in C language using Vivado HLS to generate the HDL code required for FPGA implementation. The authors reported that their framework utilized less area due to pruning, and as a result, it consumes less power.

2) OTHER FRAMEWORKS

There are some inference frameworks for BNN, some of them are open-source software frameworks like BMXNet [222], BMXNet 2 [223], daBNN [224], Riptide [225], and Larq [226], and the others are not available to the public like BitStream [227]. This section discusses only the open-source software frameworks. The following lines describe these frameworks.

BMXNet [222] is a BNN library based on MXNet [228], it released under Apache license. BMXNet supports binary and quantized weights and inputs. BMXNet supports XNOR-Net and DoReFa-Net. BMXNet used Python for training and validation for the BNN. This library is suitable for CPU and GPU implementations. BMXNet framework is used in research work like [28], [84], and [85]. As an extension of BMXNet, BMXNet 2 [223] proposed three new functions the sign, round functions with STE, and grad-cancel operator. In addition, BMXNet 2 separates the training and inference code mixed in the first version.

daBNN [224] is a fast BNN inference framework for ARM devices, released under the BSD license. daBNN utilized the following methods to speed the inference: upgraded bit-packing scheme, direct binary convolution, and novel memory layout to decrease memory access. daBNN used C++ and ARM assembly in the written codes, it supported Java binding and Android package. daBNN based on standard Open Neural Network Exchange (ONNX) operators to ensure easy deployment on other frameworks. daBNN proposed onnx2bnn, that is a model conversion tool to convert trained BNN models to the daBNN format. daBNN reported six times faster on Bi-Real Net-18 than on the BMXNet. daBNN framework is used in research work like [58] and [229].

Riptide [225] is a fast BNN framework that based on TensorFlow [217] and TVM [230]. Riptide utilized TensorFlow during the training phase and TVM to compile efficient machine code by automatic search and find high-quality hyper-parameters to maximize performance. Riptide achieved 4-12 times speedup compared to the floating-point implementation.

Larq [226] is a framework for BNN and other quantized neural networks. Larq based on Tensorflow-Keras. Larq divided into three parts; the first part is Larq library used for building and training BNN, the second part is Larq Zoo used for testing and maintaining the pre-trained models. The third part is Larq Compute Engine (LCE) used for deployment on mobile and edge devices [231]. LCE reported that it accelerates the binary convolutions by 8.5:18.5 times compared to their full-precision counterparts. Larq is used in research work like [232], [233], [234], [235], [236], and [237].

F. RQ6: WHAT ARE THE APPLICATIONS THAT UTILIZE BNN? WHAT ARE THE USED DATASETS IN THESE APPLICATIONS AND THEIR EVALUATION METRICS?

Many applications can utilize the BNN to benefit from the BNN's advantages like saving memory, area, and power. The following subsections illustrate the applications of BNN.

1) IMAGE CLASSIFICATION

It is an essential application in computer vision and machine learning, so most research works evaluate the BNN on the image classification tasks. Image classification predicts the class of one object from a collection of predefined classes that it has been trained on. The BNN strategies are tested over different deep network structures like VGG [238], AlexNet [239], ResNet-18 [240], ResNet-20, ResNet-34, ResNet-50. These tasks use datasets like MNIST, SVHN, CIFAR-10, and ImageNet. Examples of the image classification tasks are illustrated in the previous tables. The descriptions of the used datasets are in the following lines.

- MNIST dataset [21] denotes the Modified National Institute of Standards and Technology dataset. It is a dataset of 70,000 grayscale images of 28 × 28 pixels for handwritten single digits between 0 and 9. The MNIST database includes 60,000 images for training, and the other 10,000 are for testing.
- SVHN dataset [23] stands for Street View House Numbers dataset. It is a dataset of over 600,000 colored images of 32 × 32 pixels of digit images coming from real-world data of ten classes. The SVHN dataset contains 73,257 digits for training, 26,032 digits for testing, and 53,1131 additional images.
- CIFAR-10 dataset [22] means the Canadian Institute for Advanced Research dataset. It is a dataset of 60,000 colored images of 32 × 32 pixels of 4 different vehicles (airplanes, cars, ships, and trucks), and six different animals (birds, cats, deer, dogs, frogs, and horses).
- ImageNet dataset [24] is a large colored images dataset that has different versions. The commonly used version is the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012); it contains 1.2 million images for training, 50,000 images for validation, and 150,000 images for testing. It has 1000 classes.

The most important evaluation metric of BNN in the image classification task is accuracy which is described in subsection IV-D1. From the results in Table 3 to Table 4, in the first research work in BNN [20], the accuracy approached the full-precision counterpart on the small dataset like CIFAR-10 but the accuracy is sever decreased on large dataset like ImageNet. Therefore, some research works trend to using scale factors or quantization functions to improve the accuracy like WRPN [51], and PACT [54] achieved comparable accuracy to the full-precision counterparts on the ImageNet dataset, where the accuracy of ResNet-18 in [54] is lower than the full-precision counterpart by only ∼7%. While the accuracy of ResNet-34 (3x-wide) in [51] is lower than the full-precision counterpart by only 1%.

Besides, adding specific regularization to the loss function to guide the network parameters in back-propagation to decrease the mismatch due to binarization leads to better accuracy like in [78] and [83], where the accuracy of ResNet-18 (BENN-SB-6, Boosting) in [83] is lower than the full-precision counterpart by only ∼8%. While the accuracy of AlexNet in [78] is lower than the full-precision counterpart by only ∼5%.

In addition, modifying the network structure to improve the accuracy like ReActNet-C [87], INSTA-BNN [93], [94], BCDNet-A [95], and [101]. They have an accuracy very close to the full-precision MobileNet-v2 by a difference of less than 1%. Nevertheless, the drawback of these modifications is increasing the calculation operation that increases the model size. While BiNeal Net (2x-wide) [82] and BCDNet-B [95] surpasse the full-precision MobileNet-v2.

Another evaluation metric is the area which is described in subsection IV-D2. Few research studies tried to minimize the model size of BNN by using fractional weights or fractional activations. From Table 5, both FleXOR [108] and SNN [109] used fractional weights but FleXOR used full-precision activations. SNN gets higher accuracy with lower bit-width in 0.67/32-bit than the FleXOR by 0.9% and realizes a more compact model in 0.44/1-bit with a decrease of around 11%. While FracBNN [110] used fractional activation of 1.4 bit and binary weight. FracBNN achieves comparable accuracy to the full-precision MobileNet-v2 by a very small difference of 0.2%, but it used around twice the number of Binary OPeration (BOP) of that used with ResNet-34 in the IR-Net on ImageNet dataset.

Another approach to decrease the area of the network model is pruning. For example, in Table 6, the results of [117] illustrate that the BNN can be compressed without loss on accuracy on ImageNet by dynamically pruning irregular redundant edges at all layers; this method does not need to retrain the model.

The last evaluation metric is the speed which is described in subsection IV-D3, through CIM, FPGA, and ASIC implementations. Table 7 to 10 provide comparisons between various hardware implementations. After improving the accuracy, it is important that the proposed methods should be hardware friendly to be applicable in real-world applications. Finally, the trade-off between accuracy, area, and speed should be taken into consideration regarding the required application.

2) OBJECT DETECTION

It is an important application in computer vision. Object detection gathers two tasks; the first task is localizing one or more objects in an image and drawing a box around them, then, the turn of the second task comes to classify the objects in the image [252]. There is some BNN research works on object detection, like in [195], the authors presented human

TABLE 11. Summary of the object detection performance of BNN on PASCAL VOC dataset.

Neural Network Approach	Reference	Network Architecture	Binarization Method / Real-Valued	Trained Dataset	mAP%
Customized	[241]	VGG16	Real-Valued	VOC2007	68.9
			BNN [20]	VOC2007	47.3
		Alexnet	Real-Valued	VOC2007	66.0
			BNN [20]	VOC2007	46.4
Faster RCNN [242]	[243]	VGG16	BDNN	VOC2012	62.6
	[244]	ResNet-18	Real-Valued	VOC2007	67.8
			ASDA-FRCNN	VOC2007	54.6
			Bi-Real Net [29]		51.0
		ResNet-18	Real-Valued	VOC2007+2012	73.2
			ASDA-FRCNN	VOC2007+2012	63.4
			Bi-Real Net [29]		60.6
		ResNet-34	Real-Valued	VOC2007+2012	75.6
			ASDA-FRCNN	VOC2007+2012	65.5
			XNOR-Net [25]		54.7
	[245]	ResNet-18	Real-Valued	VOC2007+2012	74.5
			BiDet	VOC2007+2012	50.0
			BiDet(SC)		59.5
			XNOR-Net [25]		48.4
			Bi-Real Net [29]		58.2
	[246]	ResNet-18	Real-Valued	VOC2007+2012	76.4
			LWS-Det	VOC2007+2012	73.2
			Bi-Real-Net [29]		60.9
			BiDet [245]		62.7
			ReActNet [87]		69.6
		ResNet-34	Real-Valued	VOC2007+2012	77.8
			LWS-Det	VOC2007+2012	75.8
			Bi-Real-Net [29]		63.1
			BiDet [245]		65.8
			ReActNet [87]		72.3
		ResNet-50	Real-Valued	VOC2007+2012	79.5
			LWS-Det	VOC2007+2012	76.9
			Bi-Real-Net [29]		65.7
			ReActNet [87]		73.1
	[247]	ResNet-18	Real-Valued	VOC2007	74.5
			DA-BNN	VOC2007	63.5
YOLOv2 [248]	[249]	DarkNet	XNOR-Net [25]	VOC2007	79.6
SSD [250]	[243]	VGG16	BDNN	VOC2007+2012	63.3
			XNOR-Net [25]		60.71
SSD512 [250]	[84]	VGG-16	Real-Valued [250]	VOC2007+2012	76.8
		BinaryDenseNet-37	BinaryDenseNet	VOC2007+2012	66.4
		BinaryDenseNet-45			68.2
SSD300 [250]	[245]	VGG16	Real-Valued	VOC2007+2012	72.4
			BiDet	VOC2007+2012	52.4
			BiDet(SC)		66.0
			XNOR-Net [25]		50.2
			Bi-Real Net [29]		63.8
		MobileNetV1	Real-Valued	VOC2007+2012	68.0
			BiDet	VOC2007+2012	51.2
			XNOR-Net [25]		48.9
	[246]	VGG16	Real-Valued	VOC2007+2012	74.3
			LWS-Det	VOC2007+2012	71.4
			Bi-Real-Net [29]		63.8
			BiDet [245]		66.0
			ReActNet [87]		68.4
Customized [251]	[251]	Customized	Real-Valued	VOC2007+2012	75.4
			BSF-XNOR	VOC2007+2012	65.1

Note: mAP stands for Mean Average Precision, BiDet(SC) means BiDet with extra shortcut for the architectures.

detection on infrared images. Sun et al. [241] suggested a fast object detection algorithm by combining the proposals prediction and object classification processes. While in [249], the authors introduced an FPGA accelerator of BNN, that based on YOLOv2 [248] for object detection. In [243], the authors introduced a greedy layer-wise method as an alternative to binarizing all the weight at the same time. Ojeda et al. [253] proposed filtering stage to the input image stream followed by BNN used for pedestrian detection. In [244], the authors provided Amplitude Suppression and Direction Activation for Faster Region-based CNN (ASDA-FRCNN) that based on suppressing the shared amplitude between the real-valued and binary filters through a new loss function. Besides, Z. Wang et al. [245] suggested BiDet, which is a training method of BNN. This method eliminates the redundant information by the information bottleneck method [254] to concentrate posteriors on informative detection prediction. In [246], the authors introduced LWS-Det, which is a layer-wise searching algorithm that used angular and amplitude loss functions in a student-teacher network. This algorithm reduced the angular and amplitude error learning by utilizing a differential binarization search and the scale factor. Also, Wang et al. [251] introduced Block Scaling Factor XNOR (BSF-XNOR) convolutional

TABLE 12. Summary of the object detection performance of BNN on MS-COCO dataset.

Neural Network Approach	Reference	Network Architecture	Binarization Method / Real-Valued	mAP @.5%	mAP @[.5, .95]%
Faster RCNN	[244]	ResNet-18	Real-Valued [242]	42.7	21.9
			ASDA-FRCNN	37.5	19.4
	[245]	ResNet-18	Real-Valued	44.8	26.0
			BiDet	24.8	12.1
			BiDet(SC)	31.0	15.7
			XNOR-Net [25]	21.6	10.4
			Bi-Real Net [29]	29.0	14.4
	[246]	ResNet-18	Real-Valued	53.8	32.2
			LWS-Det	44.9	26.9
			Bi-Real Net [29]	33.1	17.4
			BiDet [245]	34.6	19.4
			ReActNet [87]	38.5	21.1
		ResNet-34	Real-Valued	57.6	35.8
			LWS-Det	49.2	29.9
			Bi-Real Net [29]	37.1	20.1
			BiDet [245]	41.8	21.7
			ReActNet [87]	43.3	23.4
		ResNet-50	Real-Valued	59.3	37.7
			LWS-Det	52.1	31.7
			Bi-Real Net [29]	40.0	22.9
			ReActNet [87]	47.7	26.1
SSD300	[245]	VGG16	Real-Valued	41.2	23.2
			BiDet	22.5	9.8
			BiDet(SC)	28.3	13.2
			XNOR-Net [25]	19.5	8.1
			Bi-Real Net [29]	26.0	11.2
	[246]	VGG-16	Real-Valued	41.2	23.2
			LWS-Det	32.9	17.1
			Bi-Real Net [29]	26.0	11.2
			BiDet [245]	28.3	13.2
			ReActNet [87]	30.0	15.3
CenterNet [256]	[82]	ResNet-18	Real-Valued	29.5	NA
			BiNeal Net	29.2	NA

Note: mAP@.5 is mAP for Intersection over Union (IoU)=0.5, mAP@[.5, .95] is mAP for IoU $\in$ [0.5 : 0.05 : 0.95].

TABLE 13. Summary of the semantic segmentation performance on BNN.

Reference	Dataset	Neural Network Approach	Network Architecture	Binarization Method / Real-Valued	mIOU
[260]	PASCAL VOC2012	FCN-8s-C5 [264]	ResNet-18	Real-Valued	67.6
				GroupNet-C	61.5
				GroupNet-C + BPAC	66.2
		FCN-8s-C4C5 [264]	ResNet-18	Real-Valued	70.1
				GroupNet-C	63.6
				GroupNet-C + BPAC	69.0
		FCN-8s-C5 [264]	ResNet-34	Real-Valued	75.0
				GroupNet-C	69.3
				GroupNet-C + BPAC	73.9
		FCN-8s-C5 [264]	ResNet-50	Real-Valued	75.5
				GroupNet-C	70.0
				GroupNet-C + BPAC	74.4
[261]	CityScapes [262]+TDG [265]	DeepLabv3	ResNet-18	Real-Valued	97.30
				binary DAD-Net	96.60
	KITTI Road [263]	DeepLabv3	ResNet-18	Real-Valued	94.45
				binary DAD-Net	95.25

Note: C4 and C5 mean extracting features from the final convolutional layer of the 4-th and 5-th stage, respectively. TDG means The training data generator that generate annotations automatically for the drivable area segmentation.

layer and two-level densely connected network structure. While Zhao et al. [247] presented DA-BNN, which used an adaptive amplitude mechanism to improve the feature representation. In addition, in [255], the authors provided an FPGA-based comparison between the CNN, Quantization Neural Network (QNN), and BNN for the object detection task.

Table 11 and Table 12 illustrate the BNN results of the object detection task on the benchmark datasets PASCAL VOC and MS-COCO, respectively. The descriptions of the benchmark datasets are in the following lines.

- PASCAL VOC [257] used to assess the models' performance in different tasks in the aspect of computer vision like object detection and semantic segmentation. It was part of the Visual Object Classes (VOC) challenges. It has 20 object classes. The two commonly used versions are VOC2007 and VOC2012. The VOC2007 dataset contains 5,011 and 4,952 images for training/validation and test data, respectively. While VOC2012 is always used as supplementary data in the training phase. The VOC2012 dataset contains 11,530 images for training/validation data.

TABLE 14. Summary of the image super-resolution performance on BNN.

Ref.	Neural Network Approach	Up-scaling Factor	Binarization Method / Real-Valued	Set5 [271]		Set14 [272]		BSD100 [273]		Urban100 [274]	
				PSRN	SSIM	PSRN	SSIM	PSRN	SSIM	PSRN	SSIM
[267]	VDSR [275]	4	Real-Valued	31.35	0.884	28.01	0.767	27.29	0.725	25.18	0.752
			BNN [20]	29.02	0.827	26.55	0.724	26.29	0.685	23.55	0.685
			DoReFa-Net [44]	29.39	0.837	26.79	0.728	26.45	0.689	23.81	0.696
			ABC-Net [49]	29.59	0.841	29.63	0.730	26.51	0.687	23.96	0.699
			BAM	30.31	0.860	27.46	0.749	26.83	0.706	24.38	0.720
	SRResNet [276]	4	Real-Valued	31.76	0.888	28.25	0.773	27.38	0.727	25.54	0.767
			BNN [20]	29.33	0.826	26.72	0.728	26.45	0.692	23.68	0.683
			DoReFa-Net [44]	30.38	0.862	27.48	0.754	26.87	0.708	24.45	0.720
			ABC-Net [49]	30.78	0.868	27.71	0.756	27.00	0.713	24.54	0.729
			BAM	31.24	0.878	27.97	0.765	27.15	0.719	24.95	0.745
[268]	VDSR [275]	4	Real-Valued	31.35	0.884	28.01	0.767	27.29	0.725	25.18	0.752
			BNN [20]	30.19	0.858	27.30	0.744	26.70	0.700	24.28	0.715
			Bi-Real Net [29]	30.38	0.861	27.41	0.748	26.82	0.705	24.35	0.718
			IR-Net [58]	30.66	0.869	27.62	0.757	26.93	0.713	24.56	0.730
			BTM	30.83	0.873	27.76	0.761	27.03	0.717	24.73	0.736
			IBTM	31.06	0.877	27.85	0.762	27.07	0.718	24.88	0.740
	EDSR [277]	4	Real-Valued	32.46	0.897	28.80	0.787	27.71	0.742	26.64	0.803
			BNN [20]	17.53	0.188	17.51	0.160	17.15	0.151	16.35	0.163
			Bi-Real Net [29]	30.81	0.871	27.71	0.760	27.01	0.716	24.66	0.733
			BTM	31.63	0.886	28.25	0.773	27.34	0.728	25.38	0.762
			IBTM	31.84	0.890	28.33	0.777	27.42	0.732	25.54	0.769
[74]	VDSR [275]	4	Real-Valued	31.35	0.884	28.01	0.767	27.29	0.725	NA	NA
			Customized	30.38	0.864	27.52	0.753	26.88	0.709	NA	NA
	SRResNet [276]	4	Real-Valued	31.76	0.888	28.25	0.773	27.38	0.727	NA	NA
			Customized	31.30	0.880	28.03	0.768	27.20	0.723	NA	NA
[82]	EDSR	4	Real-Valued	32.48	0.894	28.82	0.781	27.72	0.736	26.65	0.805
			BiNeal Net	31.94	0.887	28.47	0.771	27.49	0.726	25.80	0.776
[269]	VDSR	4	Real-Valued	31.61	0.886	28.19	0.772	27.28	0.726	25.32	0.759
			PDBC-F	30.95	0.875	27.81	0.761	27.05	0.717	24.76	0.737
	SRResNet [276]	4	Real-Valued	31.76	0.888	28.25	0.773	27.38	0.727	25.54	0.767
			PDBC-F	31.51	0.883	28.14	0.770	27.27	0.723	25.23	0.756
	EDSR [277]	4	Real-Valued	32.46	0.897	28.80	0.787	27.71	0.742	26.64	0.803
			PDBC-F	31.80	0.889	28.34	0.775	27.39	0.730	25.56	0.769

Note: PSRN means Peak Signal to Noise Ratio and SSIM means structural similarity.

TABLE 15. Comparison of the point cloud classification of BNN on ModelNet40 dataset.

Reference	Neural Network Approach	Binarization Method / Real-Valued	Overall Accuracy%
[279]	PointNet [282]	Real-Valued	88.2
		BNN [20]	7.1
		XNOR-Net [25]	64.9
		BiPointNet	86.4
	PointNet++ [283]	Real-Valued	90.0
		XNOR-Net [25]	63.1
		BiPointNet	87.8
	PointCNN [284]	Real-Valued	90.0
		XNOR-Net [25]	83.0
		BiPointNet	83.8
	DGCNN [285]	Real-Valued	89.2
		XNOR-Net [25]	51.5
		BiPointNet	83.4
[280]	PointNet [282]	Real-Valued	89.2
		XNOR-Net [25]	81.9
		Bi-Real Net [29]	77.5
		POEM	90.2
	PointNet++ [283]	Real-Valued	91.9
		XNOR-Net [25]	83.8
		POEM	91.2
	DGCNN [285]	Real-Valued	89.2
		XNOR-Net [25]	81.5
		POEM	91.1

- MS-COCO [258] dataset stands for Microsoft Common Objects in Context. It is used for various tasks like image recognition, classification, object detection, and segmentation. It has 80 object classes. The MS-COCO release of 2015 has 165,482 images for training, 81,208 images for validation, and 81,434 images for test.

The evaluation metric of the object detection is mean Average Precision (mAP), which measures the sensitivity of the neural network. From the result in Table 11 on PASCAL VOC dataset, the highest mAP is 79% that achieved by [249] it based on YOLOv2 [248] with DarkNet backbone. While LWS-Det [246] based on Faster RCNN with ResNet-34,

TABLE 16. Comparison of the point cloud part segmentation of BNN on ShapeNet Parts dataset.

Reference	Neural Network Approach	Binarization Method / Real-Valued	mIOU
[279]	PointNet [282]	Real-Valued	84.3
		BNN [20]	54.0
		BiPointNet	80.6
[280]	PointNet [282]	Real-Valued	83.7
		XNOR-Net [25]	75.3
		Bi-Real Net [29]	70.0
		POEM	81.1
	PointNet++ [283]	Real-Valued	85.1
		XNOR-Net [25]	77.7
		POEM	82.9
	DGCNN [285]	Real-Valued	85.2
		XNOR-Net [25]	77.4
		POEM	83.1

TABLE 17. Comparison of the point cloud semantic segmentation of BNN on S3DIS dataset.

Reference	Neural Network Approach	Binarization Method / Real-Valued	mIOU	Overall Accuracy%
[279]	PointNet [282]	Real-Valued	54.4	83.5
		BNN [20]	9.5	45.0
		BiPointNet	44.3	76.7
[280]	PointNet [282]	Real-Valued	47.7	78.6
		XNOR-Net [25]	39.1	70.4
		Bi-Real Net [29]	35.5	65.0
		POEM	45.8	77.9
	PointNet++ [283]	Real-Valued	53.2	82.7
		XNOR-Net [25]	43.1	75.9
		POEM	49.8	80.4
	DGCNN [285]	Real-Valued	56.1	84.2
		XNOR-Net [25]	45.6	78.0
		POEM	50.1	81.3

obtain a comparable mAP with the full-precision counterpart by a difference of ∼2%.

From the result in Table 12 on the MS-COCO dataset, the highest mAP is 52.1% that achieved by LWS-Det [246] based on Faster RCNN with ResNet-50, which decreased by ∼7% from its full-precision counterpart.

3) SEMANTIC SEGMENTATION

Semantic segmentation is the process of pixel-level labeling with a set of object categories in the image [259]. Zhuang et al. [260] applied BNN for semantic segmentation task through the GroupNet algorithm. The GroupNet decomposed the network into desired groups and approximated each group utilizing a combination of binary bases. In addition, the authors provided Binary Parallel Atrous Convolution (BPAC) to enhance the performance. While Frickenstein et al. [261] proposed Binarized Driveable Area Detection Network (Binary DAD-Net) that is used for autonomous driving. Table 13 shows the BNN results of the semantic segmentation task on the benchmark datasets PASCAL VOC2012, CityScapes [262], and KITTI Road [263], respectively. The descriptions of the benchmark datasets are in the following lines, and the PASCAL VOC2012 dataset description is aforementioned before.

- CityScapes dataset [262] is a large-scale dataset for pixel-level and instance-level semantic labeling 19 classes. It includes 2,975 training images, 500 validation images, and 1,525 test images, all of these images from German street scenes.
- KITTI Road dataset [263] contains 289 images with manually annotated ground truth labels, 259 images for training, and 30 images for validation.

The evaluation metric of the semantic segmentation is mean Intersection-over-Union (mIoU). From the results of Table 13 on the PASCAL VOC2012 dataset, the (GroupNet-C+BPAC) with ResNet-50 obtains a comparable result to the full-precision counterpart by a difference of 1.1. Also, binary DAD-Net achieves results close to its full-precision counterpart on the CityScapes dataset by a difference of 0.7.

4) IMAGE SUPER-RESOLUTION

The goal of Image Super-Resolution (ISR) is to improve the resolution of images and videos in computer vision [266]. There are a few research works that utilize the BNN to alleviate the heavy computation required by the ISR, such as [267], in which the authors introduced a binarization method based on the Bit-Accumulation Mechanism (BAM) to improve the precision. Also, in [74], the authors suggest a binarization model based on pixel-correlation knowledge distillation and trainable scaling factors. While Jiang et al. [268] provided a Binary Training Mechanism (BTM) that used the feature distribution as an alternative of the Batch-Normalization layer and improved this design precision with a multi-stage knowledge distillation technique. In [269], the authors proposed a precision-driven binary convolution (PDBC) that provides approximate multi-bit representations for activation to replace the traditional binary convolution.

All the above-mentioned work used DIV2K dataset [270] in the training phase, which contains 800 training images, 100 validation images, and 100 testing images. Table 14 shows the evaluation results that are tested on four standard datasets Set5 [271], Set14 [272], BSD100 [273], and Urban100 [274].

The evaluation metrics of the ISR are Peak Signal to Noise Ratio (PSNR) and structural similarity (SSIM). From the results of Table 14, EDSR with BiNeal Net achieves the highest PSNR on Set5, Set 14, BSD100, and Urban100 datasets by differences of 0.54, 0.35, 0.23, and 0.85, from its full-precision counterpart, respectively. Also, EDSR with IBTM achieves the highest SSIM on Set5, Set14, and BSD100 datasets by differences of 0.007, 0.01, and 0.01, from its full-precision counterpart, respectively. While EDSR with BiNeal Net achieves the highest SSIM on Urban100 by the difference of 0.029 from its full-precision counterpart.

5) POINT CLOUD TASKS

There are many point cloud tasks like classification, part segmentation, and semantic segmentation [278]. These tasks are 3D tasks that depend on the computation of a group of point-wise geometric attributes. The 3D tasks are more challenging than the 2D tasks; thus, the binarization process enlarges the information loss in the 3D tasks. Therefore some researchers try to improve this loss by utilizing algorithms that make the BNN fit with the point cloud operations. Qin et al. [279] suggested BiPointNet, which is a binary methodology for point clouds. BiPointNet maximizes the information entropy and restores feature representation efficiently by using Entropy Maximizing Aggregation (EMA) and Layer-wise Scale Recovery (LSR), respectively. While in [280], the authors derived Point-wise Operations based on Expectation-Maximization (POEM). POEM depends on the Expectation-Maximization methodology [281], by which the weights are constrained for a robust bi-modal distribution. In addition, POEM provided trainable scale factors to improve the representation capacity of the binary fully-connected layers.

The point cloud tasks classification, part segmentation, and semantic segmentation utilize benchmark dataset ModelNet40 [286], ShapeNet Parts [287], and S3DIS [288], respectively. The descriptions of these benchmark datasets are in the following lines.

- ModelNet40 [286] dataset is a benchmark for point cloud classification. It includes 12,311 CAD models from 40 object classes.
- ShapeNet [287] dataset includes is a subset of 300M models with 220k categorized into 3,135 classes. ShapeNet Parts is a subset that comprises 31,693 meshes classified into 16 common object classes. Each shape has from two to five portions based on the category with fifty part classes.
- S3DIS [288] dataset denotes Stanford 3D Indoor Scene dataset, that used for semantic segmentation. It comprises 3D scan point clouds for six indoor areas, containing 272 rooms in total, and each point belongs to one of 13 semantic categories.

The evaluation metric of the point cloud classification and semantic segmentation is the overall accuracy, while the evaluation metric of the part segmentation is mIOU. From the results in Table 15 on the ModelNet40 dataset, the highest overall accuracy is achieved by PointNet++ with POEM [280] by a difference of 0.7% from its full-precision counterpart. Table 16 shows that the highest mIOU obtained on the ShapeNet Parts dataset by DGCNN with POEM [280] by a difference of 2.1 from its full-precision counterpart. Table 17 on the S3DIS dataset, illustrates the highest overall accuracy, and mIOU achieved by DGCNN with POEM [280] by a difference of 2.9% and 6 from its full-precision counterpart, respectively.

6) OTHER APPLICATIONS

This part describes other applications that utilize the BNN in the following lines:

- In the facial recognition task, In [289], the authors provided LBP-BNN, in which the BNN with Local Binary Pattern is used for emotion detection application. While in [290], the authors suggested BinaryCoP, which is a BNN accelerator based on FINN framework [169] for facial-mask for wearing positions of the MaskedFace-Net dataset [291].
- In the health care aspect, Hirtzlin et al. [143] employed hybrid Memristor-CMOS to implement BNN for a biomedical signal task like electrocardiography (ECG) signals. In addition, In [62] and [292], the authors applied BNN for medical image segmentation.
- In the natural language processing field, Shridhar et al. [293] can apply the BNN to text classifications.
- In audio tasks, Chen et al. [70] used BNN for monaural speech separation. While in [71], the authors utilized BNN for speech recognition. Besides, Cerutti et . [294] merge analog binary feature extraction with BNN for keyword spotting on microcontroller units. Also in [295], the authors proposed BNN for Keyword Spotting. In addition, Saeed [296] presented an early-exiting approach to accelerate BNN inference for audio tasks.
- In Human Activity Recognition (HAR) field, De Vita et al. [193] proposed FPGA-based implementation for HAR. While in [297], provided BNN for general purpose processors with a RISC-V instruction set. In addition, in [298] the authors suggested Binary-DilatedDenseNet for low-latency and low-memory for HAR.
- In security applications, Xu [299] introduced BNN for person re-identification task. While in [300], the authors provided BNN for multispectral image classification. Besides, in [301], the authors employed BNN with

network architecture search for synthetic aperture radar (SAR) ship classification.
- In fault diagnosis, Tong et al. [302] deployed a fault diagnosis system for an open-circuit fault (OCF) of a Modular Multilevel Converter (MMC).

G. RQ7: WHAT ARE THE CHALLENGES AND FUTURE WORK OF THE BNN?

From 2016 until the submission of this paper, the BNN has developed as discussed in the literature. This section listed the challenges and the future work for the BNN.

- Most of the studies perform offline training that takes a long time. Several training methods are discussed in the literature, based on the chosen network architecture and its application. Therefore, we could not specify a certain method to be the best one. Online training may be the solution for rapid and efficient training. To the best of our knowledge, there is one paper for BNN online training [215]. For real-time applications such as security, the dataset should be quickly updated, which may require online training. Online training is an open research point that requires many research studies.
- Most of the BNN studies are for image classification; there are few studies for the other applications mentioned above in the previous subsection. Therefore, other applications such as the point cloud tasks, image segmentation, and speech applications require more research studies.
- Although the BNN minimizes the memory storage and power consumption regarding the standard DNNs, the data transfers and memory access still exhaust a significant part of the energy during the execution of BNN. Computing in-memory (CIM) is a good solution to reduce the data transfers and memory access for BNN implementation, but it suffers from the circuit non-ideality like the case of memristor that reduces the system accuracy. Therefore, we call for more research to solve this challenge.
- The BNN design cannot be generalized for all tasks; each requires a specific BNN design. Thus, choosing the appropriate network topology for a particular task is still an open question.

V. CONCLUSION

BNNs provide promising solutions for the hardware implementation of machine learning based applications. Compared to CNNs, the implementation of BNNs reduces networks' complexities, memory footprint, and power consumption. However, the main drawback of utilizing BNNs is the information loss due to binarization. Several research studies proposed various techniques to improve the performance of BNNs. We performed a systematic literature review that presents the state-of-the-art in BNN research through data obtained from 239 research studies. We presented a comprehensive review of three BNN optimization approaches: accuracy optimization, compression optimization, and acceleration. We explored various application domains that utilize BNN implementations and their evaluation metrics. Finally, the paper identified current challenges in BNN design and the future trends in BNN research. The discussed optimization approaches include improving the accuracy during the training process, compressing the BNN model, and improving the speed and power consumption by using CIM, FPGA, or ASICs. Our literature review showed that combining different optimization approaches lead to better performance.

REFERENCES

[1] S. S. Basha, S. R. Dubey, V. Pulabaigari, and S. Mukherjee, ''Impact of fully connected layers on performance of convolutional neural networks for image classification,'' *Neurocomputing*, vol. 378, pp. 112–119, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219313803

[2] W. Zhou, H. Wang, and Z. Wan, ''Ore image classification based on improved CNN,'' *Comput. Electr. Eng.*, vol. 99, Apr. 2022, Art. no. 107819.

[3] X. Gao, G. Xing, S. Roy, and H. Liu, ''RAMP-CNN: A novel neural network for enhanced automotive radar object recognition,'' *IEEE Sensors J.*, vol. 21, no. 4, pp. 5119–5132, Feb. 2021.

[4] F. Ashiq, M. Asif, M. B. Ahmad, S. Zafar, K. Masood, T. Mahmood, M. T. Mahmood, and I. H. Lee, ''CNN-based object recognition and tracking system to assist visually impaired people,'' *IEEE Access*, vol. 10, pp. 14819–14834, 2022.

[5] N. Zhang, X. Wei, H. Chen, and W. Liu, ''FPGA implementation for CNN-based optical remote sensing object detection,'' *Electronics*, vol. 10, no. 3, p. 282, Jan. 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/3/282

[6] A. A. Abdelhamid, E.-S.-M. El-Kenawy, B. Alotaibi, G. M. Amer, M. Y. Abdelkader, A. Ibrahim, and M. M. Eid, ''Robust speech emotion recognition using CNN+LSTM based on stochastic fractal search optimization algorithm,'' *IEEE Access*, vol. 10, pp. 49265–49284, 2022.

[7] S. Kwon, ''MLT-DNet: Speech emotion recognition using 1D dilated CNN based on multi-learning trick approach,'' *Expert Syst. Appl.*, vol. 167, Apr. 2021, Art. no. 114177. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420309131

[8] N. Lopac, F. Hrzic, I. P. Vuksanovic, and J. Lerga, ''Detection of non-stationary GW signals in high noise from Cohen's class of time–frequency representations using deep learning,'' *IEEE Access*, vol. 10, pp. 2408–2428, 2022.

[9] Y. Guo, ''A survey on methods and theories of quantized neural networks,'' 2018, *arXiv:1808.04752*.

[10] S. Xu, A. Huang, L. Chen, and B. Zhang, ''Convolutional neural network pruning: A survey,'' in *Proc. 39th Chin. Control Conf. (CCC)*, Jul. 2020, pp. 7458–7463.

[11] Q. Qi, Y. Lu, J. Li, J. Wang, H. Sun, and J. Liao, ''Learning low resource consumption CNN through pruning and quantization,'' *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 886–903, Jan. 2021.

[12] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, ''Pruning and quantization for deep neural network acceleration: A survey,'' *Neurocomputing*, vol. 461, pp. 370–403, Oct. 2021.

[13] B. A. Kitchenham and S. Charters, ''Guidelines for performing systematic literature reviews in software engineering,'' Keele Univ., Keele, U.K., Tech. Rep., EBSE 2007-001, Jul. 2007. [Online]. Available: https://www.elsevier.com/_data/promis_misc/525444systematic reviewsguide.pdf

[14] N. Khoshavi, C. Broyles, and Y. Bi, ''A survey on impact of transient faults on BNN inference accelerators,'' 2020, *arXiv:2004.05915*.

[15] W. Zhao, T. Ma, X. Gong, B. Zhang, and D. Doermann, ''A review of recent advances of binary neural networks for edge computing,'' *IEEE J. Miniaturization Air Space Syst.*, vol. 2, no. 1, pp. 25–35, Mar. 2021, doi: 10.1109/JMASS.2020.3034205.

[16] T. Simons and D.-J. Lee, ''A review of binarized neural networks,'' *Electronics*, vol. 8, no. 6, p. 661, Jun. 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/6/661

[17] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," *Pattern Recognit.*, vol. 105, Sep. 2020, Art. no. 107281, doi: 10.1016/j.patcog.2020.107281.
[18] C. Yuan and S. S. Agaian, "A comprehensive review of binary neural network," 2021, *arXiv:2110.06804*.
[19] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2015, pp. 3123–3131.
[20] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016, *arXiv:1602.02830*.
[21] Y. LeCun, C. Cortes, and C. Burges. (2010). *MNIST Handwritten Digit Database*. ATT Labs. [Online]. Available: http://yann.lecun.com/exdb/mnist
[22] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., Apr. 2009.
[23] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. (Jan. 2011). *Reading Digits in Natural Images With Unsupervised Feature Learning*. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf
[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 248–255.
[25] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. ECCV*, in Lecture Notes in Computer Science, Amsterdam, The Netherlands, Oct. 2016, pp. 525–542.
[26] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.
[27] W. Tang, G. Hua, and L. Wang, "How to train a compact binary neural network with high accuracy?" in *Proc. AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, Feb. 2017, pp. 1–12.
[28] J. Bethge, H. Yang, C. Bartz, and C. Meinel, "Learning to train a binary neural network," 2018, *arXiv:1809.10463*.
[29] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng, "Bi-Real net: Enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm," 2018, *arXiv:1808.00278*.
[30] E. Sari, M. Belbahri, and V. P. Nia, "How does batch normalization help binary training?" 2019, *arXiv:1909.09139*.
[31] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia, "Regularized binary network training," 2018, *arXiv:1812.11800*.
[32] A. Fawaz, P. Klein, S. Piat, S. Severini, and P. Mountney, "Training and meta-training binary neural networks with quantum computing," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1674–1681.
[33] S. Leroux, B. Vankeirsbilck, T. Verbelen, P. Simoens, and B. Dhoedt, "Training binary neural networks with knowledge transfer," *Neurocomputing*, vol. 396, pp. 534–541, Jul. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219304898
[34] B. Martinez, J. Yang, A. Bulat, and G. Tzimiropoulos, "Training binary neural networks with real-to-binary convolutions," 2020, *arXiv:2003.11535*.
[35] H. Kim, J. Park, C. Lee, and J.-J. Kim, "Improving accuracy of binary neural networks using unbalanced activation distribution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7862–7871.
[36] E. Wang, J. J. Davis, D. Moro, P. Zielinski, C. Coelho, S. Chatterjee, P. Y. K. Cheung, and G. A. Constantinides, "Enabling binary neural network training on the edge," in *Proc. 5th Int. Workshop Embedded Mobile Deep Learn. (EMDL)*, Jun. 2021, pp. 37–38.
[37] K. Han, Y. Wang, Y. Xu, C. Xu, E. Wu, and C. Xu, "Training binary neural networks through learning with noisy supervision," in *Proc. 37th Int. Conf. Mach. Learn.*, in Proceedings of Machine Learning Research, vol. 119, A. Singh, Ed., Jul. 2020, pp. 4017–4026. [Online]. Available: https://proceedings.mlr.press/v119/han20d.html
[38] J. Laydevant, M. Ernoult, D. Querlioz, and J. Grollier, "Training dynamical binary neural networks with equilibrium propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 4640–4649.
[39] F. Li, B. Liu, X. Wang, B. Zhang, and J. Yan, "Ternary weight networks," 2016, *arXiv:1605.04711*.
[40] N. Mellempudi, A. Kundu, D. Mudigere, D. Das, B. Kaul, and P. Dubey, "Ternary neural networks with fine-grained quantization," 2017, *arXiv:1705.01462*.
[41] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," 2016, *arXiv:1612.01064*.
[42] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless CNNs with low-precision weights," 2017, *arXiv:1702.03044*.
[43] R. Razani, G. Morin, E. Sari, and V. P. Nia, "Adaptive binary-ternary quantization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 4613–4618.
[44] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.
[45] Q. Hu, P. Wang, and J. Cheng, "From hashing to CNNs: Training binary weight networks via hashing," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Apr. 2018, pp. 1–8.
[46] Z. Li, B. Ni, W. Zhang, X. Yang, and W. Gao, "Performance guaranteed network acceleration via high-order residual quantization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2584–2592.
[47] Z. Xu and R. C. C. Cheung, "Accurate and compact convolutional neural networks with trained binarization," 2019, *arXiv:1909.11366*.
[48] A. Bulat and G. Tzimiropoulos, "XNOR-Net++: Improved binary neural networks," 2019, *arXiv:1909.13863*.
[49] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," 2017, *arXiv:1711.11294*.
[50] Y. Li, Y. Bao, and W. Chen, "Fixed-sign binary neural network: An efficient design of neural network for Internet-of-Things devices," *IEEE Access*, vol. 8, pp. 164858–164863, 2020.
[51] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "WRPN: Wide reduced-precision networks," 2017, *arXiv:1709.01134*.
[52] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave Gaussian quantization," in *Proc. CVPR*, Jun. 2017, pp. 5918–5926.
[53] J. Faraone, N. Fraser, M. Blott, and P. H. Leong, "SYQ: Learning symmetric quantization for efficient deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4300–4309.
[54] J. Choi, Z. Wang, S. Venkataramani, P. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: Parameterized clipping activation for quantized neural networks," 2018, *arXiv:1805.06085*.
[55] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 365–382.
[56] P. Wang, Q. Hu, Y. Zhang, C. Zhang, Y. Liu, and J. Cheng, "Two-step quantization for low-bit neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 4376–4384.
[57] J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, and X.-S. Hua, "Quantization networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Dec. 2019, pp. 7308–7316.
[58] H. Qin, R. Gong, X. Liu, M. Shen, Z. Wei, F. Yu, and J. Song, "Forward and backward information retention for accurate binary neural networks," in *Proc. CVPR*, Jun. 2020, pp. 2250–2259.
[59] H. Yang, L. Duan, Y. Chen, and H. Li, "BSQ: Exploring bit-level sparsity for mixed-precision neural network quantization," 2021, *arXiv:2102.10462*.
[60] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2019, pp. 4852–4861.
[61] X. He, Z. Mo, K. Cheng, W. Xu, Q. Hu, P. Wang, Q. Liu, and J. Cheng, "ProxyBNN: Learning binarized neural networks via proxy matrices," in *Proc. 16th Eur. Conf. Comput. Vis.* Glasgow, U.K.: Springer, Aug. 2020, pp. 223–241.
[62] R. Zhang and A. C. S. Chung, "MedQ: Lossless ultra-low-bit neural network quantization for medical image segmentation," *Med. Image Anal.*, vol. 73, Oct. 2021, Art. no. 102200.
[63] P. Pham, J. A. Abraham, and J. Chung, "Training multi-bit quantized and binarized networks with a learnable symmetric quantizer," *IEEE Access*, vol. 9, pp. 47194–47203, 2021.
[64] Z. Xu, M. Lin, J. Liu, J. Chen, L. Shao, Y. Gao, Y. Tian, and R. Ji, "ReCU: Reviving the dead weights in binary neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2021, pp. 5198–5208.

[65] L. Hou, Q. Yao, and J. T. Kwok, "Loss-aware binarization of deep networks," 2016, *arXiv:1611.01600*.
[66] R. Ding, T.-W. Chin, Z. Liu, and D. Marculescu, "Regularizing activation distribution for training binarized deep networks," 2019, *arXiv:1904.02823*.
[67] Y. Shang, D. Xu, Z. Zong, L. Nie, and Y. Yan, "Network binarization via contrastive learning," 2022, *arXiv:2207.02970*.
[68] Y. Shang, D. Xu, B. Duan, Z. Zong, L. Nie, and Y. Yan, "Lipschitz continuity retained binary neural network," 2022, *arXiv:2207.06540*.
[69] Z. Wang, J. Lu, C. Tao, J. Zhou, and Q. Tian, "Learning channel-wise interactions for binary convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1–20.
[70] X. Chen, G. Liu, J. Shi, J. Xu, and B. Xu, "Distilled binary neural network for monaural speech separation," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.
[71] Y.-M. Qian and X. Xiang, "Binary neural networks for speech recognition," *Frontiers Inf. Technol. Electron. Eng.*, vol. 20, no. 5, pp. 701–715, May 2019.
[72] A. Bulat, G. Tzimiropoulos, J. Kossaifi, and M. Pantic, "Improved training of binary networks for human pose estimation and image recognition," 2019, *arXiv:1904.05868*.
[73] J. Yang, B. Martinez, A. Bulat, and G. Tzimiropoulos, "Knowledge distillation via adaptive instance normalization," 2020, *arXiv:2003.04289*.
[74] Q. Huang, Y. Zhang, H. Hu, Y. Zhu, and Z. Zhao, "Binarizing super-resolution networks by pixel-correlation knowledge distillation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2021, pp. 1814–1818.
[75] C. Liu, W. Ding, X. Xia, B. Zhang, J. Gu, J. Liu, R. Ji, and D. Doermann, "Circulant binary convolutional networks: Enhancing the performance of 1-bit DCNNs with circulant back propagation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 2691–2699.
[76] R. Ding, H. Liu, and X. Zhou, "IE-Net: Information-enhanced binary neural networks for accurate classification," *Electronics*, vol. 11, no. 6, p. 937, Mar. 2022.
[77] Z. Yang, Y. Wang, K. Han, C. Xu, C. Xu, D. Tao, and C. Xu, "Searching for low-bit weights in quantized neural networks," 2020, *arXiv:2009.08695*.
[78] H. Kim, K. Kim, J. Kim, and J.-J. Kim, "BinaryDuo: Reducing gradient mismatch in binary activation network by coupling binary activations," 2020, *arXiv:2002.06517*.
[79] P. Wang, X. He, G. Li, T. Zhao, and J. Cheng, "Sparsity-inducing binarized neural networks," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2020, vol. 34, no. 7, pp. 12192–12199.
[80] Y. Xu, K. Han, C. Xu, Y. Tang, C. Xu, and Y. Wang, "Learning frequency domain approximation for binary neural networks," 2021, *arXiv:2103.00841*.
[81] J. Lee, D. Kim, and B. Ham, "Network quantization with element-wise gradient scaling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6448–6457.
[82] G. Nie, L. Xiao, M. Zhu, D. Chu, Y. Shen, P. Li, K. Yang, L. Du, and B. Chen, "Binary neural networks as a general-propose compute paradigm for on-device computer vision," 2022, *arXiv:2202.03716*.
[83] S. Zhu, X. Dong, and H. Su, "Binary ensemble neural network: More bits per network or more networks per bit?" in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4923–4932.
[84] J. Bethge, H. Yang, M. Bornstein, and C. Meinel, "BinaryDenseNet: Developing an architecture for binary neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1–13.
[85] J. Bethge, C. Bartz, H. Yang, Y. Chen, and C. Meinel, "MeliusNet: Can binary neural networks achieve MobileNet-level accuracy?" 2020, *arXiv:2001.05936*.
[86] H. Phan, D. Huynh, Y. He, M. Savvides, and Z. Shen, "MoBiNet: A mobile binary network for image classification," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 3453–3462.
[87] Z. Liu, Z. Shen, M. Savvides, and K.-T. Cheng, "ReActNet: Towards precise binary neural network with generalized activation functions," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Aug. 2020, pp. 143–159.
[88] P. Xue, Y. Lu, J. Chang, X. Wei, and Z. Wei, "Self-distribution binary neural networks," *Appl. Intell.*, vol. 2022, pp. 1–13, Feb. 2022.
[89] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3453–3462.
[90] J. Zhang, Z. Su, Y. Feng, X. Lu, M. Pietikainen, and L. Liu, "Dynamic binary neural network by learning channel-wise thresholds," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 1885–1889.
[91] R. Sun, W. Zou, and Y. Zhan, "'Ghost' and attention in binary neural network," *IEEE Access*, vol. 10, pp. 60550–60557, 2022.
[92] C. Liu, W. Ding, P. Chen, B. Zhuang, Y. Wang, Y. Zhao, B. Zhang, and Y. Han, "RB-Net: Training highly accurate and efficient binary neural networks with reshaped point-wise convolution and balanced activation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 9, pp. 6414–6424, Sep. 2022.
[93] Z. Tu, X. Chen, P. Ren, and Y. Wang, "AdaBin: Improving binary neural networks with adaptive binary sets," 2022, *arXiv:2208.08084*.
[94] C. Lee, H. Kim, E. Park, and J.-J. Kim, "INSTA-BNN: Binary neural network with instance-aware threshold," 2022, *arXiv:2204.07439*.
[95] X. Xing, Y. Li, W. Li, W. Ding, Y. Jiang, Y. Wang, J. Shao, C. Liu, and X. Liu, "Towards accurate binary neural networks via modeling contextual dependencies," 2022, *arXiv:2209.01404*.
[96] T. Chen, Z. Zhang, X. Ouyang, Z. Liu, Z. Shen, and Z. Wang, "'BNN-BN=?': Training binary neural networks without batch normalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 4619–4629.
[97] A. Bulat, B. Martinez, and G. Tzimiropoulos, "BATS: Binary architecture search," in *Proc. 16th Eur. Conf. Comput. Vis.* Glasgow, U.K.: Springer, Aug. 2020, pp. 309–325.
[98] D. Kim, K. P. Singh, and J. Choi, "Learning architectures for binary networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Aug. 2020, pp. 575–591.
[99] B. Zhu, Z. Al-Ars, and H. P. Hofstee, "NASB: Neural architecture search for binary convolutional neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.
[100] H. Phan, Z. Liu, D. Huynh, M. Savvides, K.-T. Cheng, and Z. Shen, "Binarizing MobileNet via evolution-based searching," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1–10.
[101] A. Bulat, B. Martinez, and G. Tzimiropoulos, "High-capacity expert binary networks," 2020, *arXiv:2010.03558*.
[102] M. Shen, K. Han, C. Xu, and Y. Wang, "Searching for accurate binary neural architectures," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1–4.
[103] Y. Li, R. Gong, F. Yu, X. Dong, and X. Liu, "DMS: Differentiable dimension search for binary neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–12.
[104] Y. Li, T. Geng, S. Stein, A. Li, and H. Yu, "GAAF: Searching activation functions for binary neural networks through genetic algorithm," 2022, *arXiv:2206.03291*.
[105] Y. Xu, X. Dong, Y. Li, and H. Su, "A Main/Subsidiary network framework for simplifying binary neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7154–7162.
[106] H. Qiu, H. Ma, Z. Zhang, Y. Zheng, A. Fu, P. Zhou, Y. Gao, D. Abbott, and S. F. Al-Sarawi, "RBNN: Memory-efficient reconfigurable deep binary neural network with IP protection for Internet of Things," 2021, *arXiv:2105.03822*.
[107] Q. H. Vo, F. Asim, B. Alimkhanuly, S. Lee, and L. Kim, "Hardware platform-aware binarized neural network model optimization," *Appl. Sci.*, vol. 12, no. 3, p. 1296, Jan. 2022.
[108] D. Lee, S. J. Kwon, B. Kim, Y. Jeon, B. Park, and J. Yun, "FleXOR: Trainable fractional quantization," 2020, *arXiv:2009.04126*.
[109] Y. Wang, Y. Yang, F. Sun, and A. Yao, "Sub-bit neural networks: Learning to compress and accelerate binary neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5360–5369.
[110] Y. Zhang, J. Pan, X. Liu, H. Chen, D. Chen, and Z. Zhang, "FracBNN: Accurate and FPGA-efficient binary neural networks with fractional activations," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2021, pp. 171–182.
[111] L. Guerra, B. Zhuang, I. Reid, and T. Drummond, "Automatic pruning for quantized neural networks," 2020, *arXiv:2002.00523*.
[112] A. G. Anderson and C. P. Berg, "The high-dimensional geometry of binary neural networks," 2017, *arXiv:1705.07199*.

[113] T. Fujii, S. Sato, and H. Nakahara, "A threshold neuron pruning for a binarized deep neural network on an FPGA," *IEICE Trans. Inf. Syst.*, vol. 101, no. 2, pp. 376–386, 2018.

[114] J. Gao, Q. Liu, and J. Lai, "An approach of binary neural network energy-efficient implementation," *Electronics*, vol. 10, no. 15, p. 1830, Jul. 2021.

[115] Q. Wu, X. Lu, S. Xue, C. Wang, X. Wu, and J. Fan, "SBNN: Slimming binarized neural network," *Neurocomputing*, vol. 401, pp. 113–122, Aug. 2020.

[116] Y. Li and F. Ren, "BNN pruning: Pruning binary neural network guided by weight flipping frequency," in *Proc. 21st Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2020, pp. 306–311.

[117] T. Geng, A. Li, T. Wang, C. Wu, Y. Li, R. Shi, W. Wu, and M. Herbordt, "O3BNN-R: An out-of-order architecture for high-performance and regularized BNN inference," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 199–213, Jan. 2021.

[118] X. Xiao and Z. Wang, "AutoPrune: Automatic network pruning by regularizing auxiliary parameters," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, Dec. 2019, pp. 1–15.

[119] J. H. Choi, Y.-H. Gong, and S. W. Chung, "A system-level exploration of binary neural network accelerators with monolithic 3D based compute-in-memory SRAM," *Electronics*, vol. 10, no. 5, p. 623, Mar. 2021.

[120] A. Agrawal et al., "Xcel-RAM: Accelerating binary neural networks in high-throughput SRAM compute arrays," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 8, pp. 3064–3076, Aug. 2019.

[121] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8μ J/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, Jan. 2019.

[122] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute," *IEEE J. Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, Jun. 2019.

[123] X. Si, W.-S. Khwa, J.-J. Chen, J.-F. Li, X. Sun, R. Liu, S. Yu, H. Yamauchi, Q. Li, and M.-F. Chang, "A dual-split 6T SRAM-based computing-in-memory unit-macro with fully parallel product-sum operation for binarized DNN edge processors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 11, pp. 4172–4185, Nov. 2019.

[124] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," *IEEE J. Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, Jun. 2020.

[125] J. Yang, Y. Kong, Z. Wang, Y. Liu, B. Wang, S. Yin, and L. Shi, "24.4 sandwich-RAM: An energy-efficient in-memory BWN architecture with pulse-width modulation," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 394–396.

[126] H. Zhang, Y. Shu, W. Jiang, Z. Yin, W. Zhao, and Y. Ha, "A 55 nm, 0.4 V 5526-TOPS/W compute-in-memory binarized CNN accelerator for AIoT applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 5, pp. 1695–1699, May 2021.

[127] J. Song, Y. Wang, M. Guo, X. Ji, K. Cheng, Y. Hu, X. Tang, R. Wang, and R. Huang, "TD-SRAM: Time-domain-based in-memory computing macro for binary neural networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 8, pp. 3377–3387, Aug. 2021.

[128] D. Kushwaha, A. Joshi, C. I. Kumar, N. Gupta, S. Miryala, R. V. Joshi, S. Dasgupta, and A. Bulusu, "An energy-efficient high CSNR XNOR and accumulation scheme for BNN," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 4, pp. 2311–2315, Apr. 2022.

[129] H. Oh, H. Kim, D. Ahn, J. Park, Y. Kim, I. Lee, and J.-J. Kim, "Energy-efficient in-memory binary neural network accelerator design based on 8T2C SRAM cell," *IEEE Solid-State Circuits Lett.*, vol. 5, pp. 70–73, 2022.

[130] H. Kim, T. Yoo, T. T.-H. Kim, and B. Kim, "Colonnade: A reconfigurable SRAM-based digital bit-serial compute-in-memory macro for processing neural networks," *IEEE J. Solid-State Circuits*, vol. 56, no. 7, pp. 2221–2233, Jul. 2021.

[131] S. Nasrin, D. Badawi, A. E. Cetin, W. Gomes, and A. R. Trivedi, "MF-Net: Compute-in-memory SRAM for multibit precision inference using memory-immersed data conversion and multiplication-free operators," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 5, pp. 1966–1978, May 2021.

[132] H. Jiang, X. Peng, S. Huang, and S. Yu, "CIMAT: A transpose SRAM-based compute-in-memory architecture for deep neural network on-chip training," in *Proc. Int. Symp. Memory Syst.*, Sep. 2019, pp. 490–496.

[133] X. Si et al., "A 28 nm 64 Kb 6T SRAM computing-in-memory macro with 8b MAC operation for AI edge chips," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 246–248.

[134] J.-W. Su et al., "A 28 nm 64 Kb inference-training two-way transpose multibit 6T SRAM compute-in-memory macro for AI edge chips," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 240–242.

[135] H. Jia, H. Valavi, Y. Tang, J. Zhang, and N. Verma, "A programmable heterogeneous microprocessor based on bit-scalable in-memory computing," *IEEE J. Solid-State Circuits*, vol. 55, no. 9, pp. 2609–2621, Jan. 2020.

[136] M. Imani, S. Gupta, Y. Kim, and T. Rosing, "FloatPIM: In-memory acceleration of deep neural network training with high precision," in *Proc. ACM/IEEE 46th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2019, pp. 802–815.

[137] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 27–39, Jun. 2016

[138] W. Wan, R. Kubendran, C. Schaefer, S. B. Eryilmaz, W. Zhang, D. Wu, S. Deiss, P. Raina, H. Qian, and B. Gao, "A compute-in-memory chip based on resistive random-access memory," *Nature*, vol. 608, no. 7923, pp. 504–512, 2022.

[139] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, 2020.

[140] K. Van Pham, T. Van Nguyen, S. B. Tran, H. Nam, M. J. Lee, B. J. Choi, S. N. Truong, and K.-S. Min, "Memristor binarized neural networks," *J. Semicond. Technol. Sci.*, vol. 18, no. 5, pp. 568–588, Oct. 2018.

[141] K. Pham, S. Tran, T. Nguyen, and K.-S. Min, "Asymmetrical training scheme of binary-memristor-crossbar-based neural networks for energy-efficient edge-computing nanoscale systems," *Micromachines*, vol. 10, no. 2, p. 141, Feb. 2019.

[142] M. Huang, G. Zhao, X. Wang, W. Zhang, P. Coquet, B. K. Tay, G. Zhong, and J. Li, "Global-gate controlled one-transistor one-digital-memristor structure for low-bit neural network," *IEEE Electron Device Lett.*, vol. 42, no. 1, pp. 106–109, Jan. 2021.

[143] T. Hirtzlin, M. Bocquet, B. Penkovsky, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz, "Digital biologically plausible implementation of binarized neural networks with differential hafnium oxide resistive memory arrays," *Frontiers Neurosci.*, vol. 13, p. 1383, Jan. 2020, doi: 10.3389/fnins.2019.01383.

[144] Y.-F. Qin, R. Kuang, X.-D. Huang, Y. Li, J. Chen, and X.-S. Miao, "Design of high robustness BNN inference accelerator based on binary memristors," *IEEE Trans. Electron Devices*, vol. 67, no. 8, pp. 3435–3441, Aug. 2020.

[145] D. Ahn, H. Oh, H. Kim, Y. Kim, and J.-J. Kim, "Maximizing parallel activation of word-lines in MRAM-based binary neural network accelerators," *IEEE Access*, vol. 9, pp. 141961–141969, 2021.

[146] L. Huang, J. Diao, H. Nie, W. Wang, Z. Li, Q. Li, and H. Liu, "Memristor based binary convolutional neural network architecture with configurable neurons," *Frontiers Neurosci.*, vol. 15, p. 328, Mar. 2021.

[147] S. K. Kingra, V. Parmar, S. Negi, A. Bricalli, G. Piccolboni, A. Regev, J.-F. Nodin, G. Molas, and M. Suri, "Dual-configuration in-memory computing bitcells using SiO_x RRAM for binary neural networks," *Appl. Phys. Lett.*, vol. 120, no. 3, Jan. 2022, Art. no. 034102.

[148] Y. Pan, P. Ouyang, Y. Zhao, W. Kang, S. Yin, Y. Zhang, W. Zhao, and S. Wei, "A multilevel cell STT-MRAM-based computing in-memory accelerator for binary convolutional neural network," *IEEE Trans. Magn.*, vol. 54, no. 11, pp. 1–5, Nov. 2018.

[149] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, and Z. Wang, "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," *Nature Commun.*, vol. 9, no. 1, pp. 1–8, Jun. 2018.

[150] O. Krestinskaya, K. N. Salama, and A. P. James, "Learning in memristive neural network architectures using analog backpropagation circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 2, pp. 719–732, Feb. 2018.

[151] W. Yi, Y. Kim, and J.-J. Kim, "Effect of device variation on mapping binary neural network to memristor crossbar array," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 320–323.

[152] Y. Zhao, Y. Wang, R. Wang, Y. Rong, and X. Jiang, "A highly robust binary neural network inference accelerator based on binary memristors," *Electronics*, vol. 10, no. 21, p. 2600, Oct. 2021.

[153] J. Chen, S. Wen, K. Shi, and Y. Yang, "Highly parallelized memristive binary neural network," *Neural Netw.*, vol. 144, pp. 565–572, Dec. 2021.

[154] V. Parmar, B. Penkovsky, D. Querlioz, and M. Suri, "Hardware-efficient stochastic binary CNN architectures for near-sensor computing," *Frontiers Neurosci.*, vol. 2022, vol. 15, Jan. 2022, doi: 10.3389/fnins.2021.781786.

[155] Z. Zhang, Z. Ge, Y. Wei, X. Cheng, G. Xie, and G. Liu, "1S–1R array: Pure-memristor circuit for binary neural networks," *Microelectron. Eng.*, vol. 254, Feb. 2022, Art. no. 111697. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016793172100201X

[156] J. Yu, W. Zhang, D. Dong, W. Sun, J. Lai, X. Zheng, T. Gong, Y. Li, D. Shang, G. Xing, and X. Xu, "Long-term accuracy enhancement of binary neural networks based on optimized three-dimensional memristor array," *Micromachines*, vol. 13, no. 2, p. 308, Feb. 2022.

[157] E. Nurvitadhi, D. Sheffield, J. Sim, A. Mishra, G. Venkatesh, and D. Marr, "Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC," in *Proc. FPT*, Dec. 2016, pp. 77–84.

[158] S. Liang, S. Yin, L. Liu, W. Luk, and S. Wei, "FP-BNN: Binarized neural network on FPGA," *Neurocomputing*, vol. 275, pp. 1072–1086, Jan. 2018.

[159] Y. Duan, S. Li, R. Zhang, Q. Wang, J. Chen, and G. E. Sobelman, "Energy-efficient architecture for FPGA-based deep convolutional neural networks with binary weights," in *Proc. IEEE 23rd Int. Conf. Digit. Signal Process. (DSP)*, Nov. 2018, pp. 1–5.

[160] P. Guo, H. Ma, R. Chen, P. Li, S. Xie, and D. Wang, "FBNA: A fully binarized neural network accelerator," in *Proc. 28th Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2018, pp. 513–551.

[161] T. Murovič and A. Trost, "Massively parallel combinational binary neural networks for edge processing," *Elektrotehniški Vestnik*, pp. 47–53, 2019.

[162] W. He, D. Yang, H. Peng, S. Liang, and Y. Lin, "An efficient ensemble binarized deep neural network on chip with perception-control integrated," *Sensors*, vol. 21, no. 10, p. 3407, May 2021.

[163] A. Loquercio, A. I. Maqueda, C. R. D. Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018.

[164] Y. Ling, T. He, H. Meng, Y. Zhang, and G. Chen, "Hardware accelerator for an accurate local stereo matching algorithm using binary neural network," *J. Syst. Archit.*, vol. 117, Aug. 2021, Art. no. 102110.

[165] P. Skrimponis, E. Pissadakis, N. Alachiotis, and D. Pnevmatikatos, "Accelerating binarized convolutional neural networks with dynamic partial reconfiguration on disaggregated FPGAs," in *Parallel Computing: Technology Trends*. Amsterdam, The Netherlands: IOS Press, 2020, pp. 691–700.

[166] J. Cho, Y. Jung, S. Lee, and Y. Jung, "Reconfigurable binary neural network accelerator with adaptive parallelism scheme," *Electronics*, vol. 10, no. 3, p. 230, Jan. 2021.

[167] Q. H. Vo, N. L. Le, F. Asim, L.-W. Kim, and C. S. Hong, "A deep learning accelerator based on a streaming architecture for binary neural networks," *IEEE Access*, vol. 10, pp. 21141–21159, 2022.

[168] A. Sasikumar, L. Ravi, K. Kotecha, V. Indragandhi, and V. Subramaniyaswamy, "Reconfigurable and hardware efficient adaptive quantization model-based accelerator for binarized neural network," *Comput. Electr. Eng.*, vol. 102, Sep. 2022, Art. no. 108302.

[169] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2017, pp. 65–74.

[170] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'Brien, Y. Umuroglu, M. Leeser, and K. Vissers, "FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 3, pp. 1–23, 2018.

[171] M. Ghasemzadeh, M. Samragh, and F. Koushanfar, "ReBNet: Residual binarized neural network," in *Proc. IEEE 26th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Los Alamitos, CA, USA, Apr. 2018, pp. 57–64, doi: 10.1109/fccm.2018.00018.

[172] E. Wang, J. J. Davis, P. K. Cheung, and G. A. Constantinides, "LUTNet: Learning FPGA configurations for highly efficient neural network inference," *IEEE Trans. Comput.*, vol. 69, no. 12, pp. 1795–1808, Mar. 2020.

[173] Z. Xu and R. C. C. Cheung, "Binary convolutional neural network acceleration framework for rapid system prototyping," *J. Syst. Archit.*, vol. 109, Oct. 2020, Art. no. 101762.

[174] M. Xiang and T. H. Teo, "Implementation of binarized neural networks in all-programmable system-on-chip platforms," *Electronics*, vol. 11, no. 4, p. 663, Feb. 2022.

[175] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M. Srivastava, R. Gupta, and Z. Zhang, "Accelerating binarized convolutional neural networks with software-programmable FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, New York, NY, USA, Feb. 2017, pp. 15–24.

[176] L. Yang, Z. He, and D. Fan, "A fully onchip binarized convolutional neural network FPGA impelmentation with accurate inference," in *Proc. Int. Symp. Low Power Electron. Design*, Jul. 2018, pp. 1–6.

[177] A. Zhakatayev and J. Lee, "Efficient FPGA implementation of local binary convolutional neural network," in *Proc. 24th Asia South Pacific Design Autom. Conf.*, New York, NY, USA, Jan. 2019, pp. 699–704, doi: 10.1145/3287624.3287719.

[178] P. Jokic, S. Emery, and L. Benini, "BinaryEye: A 20 kfps streaming camera system on FPGA with real-time on-device image recognition using binary neural networks," in *Proc. IEEE 13th Int. Symp. Ind. Embedded Syst. (SIES)*, Jun. 2018, pp. 1–7.

[179] J. Hong, S. Arslan, T. Lee, and H. Kim, "Design of power-efficient training accelerator for convolution neural networks," *Electronics*, vol. 10, no. 7, p. 787, Mar. 2021.

[180] R. Xiao, J. Shi, and C. Zhang, "FPGA implementation of CNN for handwritten digit recognition," in *Proc. IEEE 4th Inf. Technol., Netw., Electron. Autom. Control Conf. (ITNEC)*, Jun. 2020, pp. 1128–1133.

[181] Y. Zhou and J. Jiang, "An FPGA-based accelerator implementation for deep convolutional neural networks," in *Proc. 4th Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT)*, vol. 1, Dec. 2015, pp. 829–832.

[182] Y. Tang, X. Zhang, P. Zhou, and J. Hu, "EF-train: Enable efficient on-device CNN training on FPGA through data reshaping for online adaptation or personalization," *ACM Trans. Design Autom. Electron. Syst.*, vol. 27, no. 5, pp. 1–36, Jun. 2022.

[183] S. K. Venkataramanaiah, H.-S. Suh, S. Yin, E. Nurvitadhi, A. Dasu, Y. Cao, and J.-S. Seo, "FPGA-based low-batch training accelerator for modern CNNs featuring high bandwidth memory," in *Proc. 39th Int. Conf. Comput.-Aided Design*, Nov. 2020, pp. 1–8.

[184] C. Luo, M.-K. Sit, H. Fan, S. Liu, W. Luk, and C. Guo, "Towards efficient deep neural network training by FPGA-based batch-level parallelism," *J. Semiconductors*, vol. 41, no. 2, Feb. 2020, Art. no. 022403.

[185] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, and L. Wang, "A high performance FPGA-based accelerator for large-scale convolutional neural networks," in *Proc. IEEE Int. Conf. Field-Program. Logic Appl. (FPL)*, Aug. 2016, pp. 1–9.

[186] N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. Ma, S. Vrudhula, J.-S. Seo, and Y. Cao, "Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, Feb. 2016, pp. 16–25.

[187] X. Lian, Z. Liu, Z. Song, J. Dai, W. Zhou, and X. Ji, "High-performance FPGA-based CNN accelerator with block-floating-point arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 8, pp. 1874–1885, Aug. 2019.

[188] H. Nakahara, H. Yonekawa, T. Sasao, H. Iwamoto, and M. Motomura, "A memory-based realization of a binarized deep convolutional neural network," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2016, pp. 277–280.

[189] Y. Zhou, S. Redkar, and X. Huang, "Deep learning binary neural network on an FPGA," in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2017, pp. 281–284.

[190] (Nov. 5, 2021). *UNSW-NB15 Dataset*. [Online]. Available: https://research.unsw.edu.au/projects/unsw-nb15-dataset

[191] (Nov. 5, 2021). *SUSY Dataset*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/SUSY

[192] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *Proc. 16th Int. Symp. Wearable Comput. (ISWC)*, Jun. 2012, pp. 108–109.

[193] A. D. Vita, D. Pau, L. D. Benedetto, A. Rubino, F. Petrot, and G. D. Licciardo, "Low power tiny binary neural network with improved accuracy in human recognition systems," in *Proc. 23rd Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2020, pp. 309–315.

[194] S. Ohara, T. Ogata, and H. Awano, "Binary neural network in robotic manipulation: Flexible object manipulation for humanoid robot using partially binarized auto-encoder on FPGA," 2021, *arXiv:2107.00209*.

[195] J. Kung, D. Zhang, G. van der Wal, S. Chai, and S. Mukhopadhyay, "Efficient object detection using embedded binarized neural networks," *J. Signal Process. Syst.*, vol. 90, no. 6, pp. 877–890, Jun. 2018.

[196] R. Ding, Z. Liu, R. Shi, D. Marculescu, and R. D. Blanton, "LightNN: Filling the gap between conventional deep neural networks and binarized networks," in *Proc. Great Lakes Symp. VLSI*, May 2017, pp. 35–40.

[197] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. Brooks, and G.-Y. Wei, "A 28 nm SoC with a 1.2 GHz 568 nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 242–243.

[198] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, and A. Borchers, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 1–12.

[199] B. Moons and M. Verhelst, "An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, Apr. 2017.

[200] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, Jan. 2018.

[201] A. Ardakani, C. Condo, M. Ahmadi, and W. J. Gross, "An architecture to accelerate convolution in deep neural networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 4, pp. 1349–1362, Oct. 2017.

[202] J. Sim, S. Lee, and L.-S. Kim, "An energy-efficient deep convolutional neural network inference processor with enhanced output stationary dataflow in 65-nm CMOS," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 1, pp. 87–100, Jan. 2020.

[203] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "YodaNN: An ultra-low power convolutional neural network accelerator based on binary weights," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 236–241.

[204] Y. Wang, J. Lin, and Z. Wang, "An energy-efficient architecture for binary weight convolutional neural networks," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 2, pp. 280–293, Nov. 2017.

[205] M. Rusci, L. Cavigelli, and L. Benini, "Design automation for binarized neural networks: A quantum leap opportunity?" 2017, *arXiv:1712.01743*.

[206] (Nov. 22, 2021). *MIO-TCD: Miovision Traffic Camera Dataset*. [Online]. Available: https://www.kaggle.com/yash88600/miotcd-dataset-50000-imagesclassification

[207] F. Conti, P. D. Schiavone, and L. Benini, "XNOR neural engine: A hardware accelerator IP for 21.6-fJ/op binary neural network inference," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2940–2951, Nov. 2018.

[208] X. Huang and Y. Zhou, "A 20 TOp/s/W binary neural network accelerator," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[209] G. C. Qiao, S. G. Hu, T. P. Chen, L. M. Rong, N. Ning, Q. Yu, and Y. Liu, "STBNN: Hardware-friendly spatio-temporal binary neural network with high pattern recognition accuracy," *Neurocomputing*, vol. 409, pp. 351–360, Oct. 2020.

[210] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Apr. 1996, pp. 43–58.

[211] P. C. Knag, G. K. Chen, H. E. Sumbul, R. Kumar, S. K. Hsu, A. Agarwal, M. Kar, S. Kim, M. A. Anders, H. Kaul, and R. K. Krishnamurthy, "A 617-TOPS/W all-digital binary neural network accelerator in 10-nm FinFET CMOS," *IEEE J. Solid-State Circuits*, vol. 56, no. 4, pp. 1082–1092, Apr. 2021.

[212] A. A. Bahou, G. Karunaratne, R. Andri, L. Cavigelli, and L. Benini, "XNORBIN: A 95 TOp/s/W hardware accelerator for binary convolutional neural networks," in *Proc. IEEE Symp. Low-Power High-Speed Chips (COOL CHIPS)*, Apr. 2018, pp. 1–3.

[213] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, vol. 1, no. 1, pp. 886–893.

[214] H. Cai, C. Gan, L. Zhu, and S. Han, "TinyTL: Reduce activations, not trainable parameters for efficient on-device learning," 2020, *arXiv:2007.11622*.

[215] S. Yu, Z. Li, P.-Y. Chen, H. Wu, B. Gao, D. Wang, W. Wu, and H. Qian, "Binary neural network with 16 Mb RRAM macro chip for classification and online training," in *IEDM Tech. Dig.*, Dec. 2016, pp. 2–16.

[216] M. Koo, G. Srinivasan, Y. Shim, and K. Roy, "SBSNN: Stochastic-bits enabled binary spiking neural network with on-chip learning for energy efficient neuromorphic computing at the edge," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 8, pp. 2546–2555, Aug. 2020.

[217] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, and M. Isard, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. OSDI*, Nov. 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[218] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, Dec. 2019, pp. 8026–8037.

[219] Xilinx. *Vivado Design Suite User Guide: High-Level Synthesis*. Accessed: Oct. 13, 2021. [Online]. Available: https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0012-vivado-high-level-synthesis-hub.html

[220] Xilinx. *Vivado Design Suite User Guide: Using the Vivado IDE*. Accessed: Oct. 13, 2021. [Online]. Available: https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0002-vivado-design-flows-overview-hub.html

[221] F. Chollet. (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras

[222] H. Yang, M. Fritzsche, C. Bartz, and C. Meinel, "BMXNet," in *Proc. 25th ACM Int. Conf. Multimedia*, New York, NY, USA, Oct. 2017, pp. 1209–1212, doi: 10.1145/3123266.3129393.

[223] J. Bethge, C. Bartz, H. Yang, and C. Meinel, "BMXNet 2: An open source framework for low-bit Networks–reproducing, understanding, designing and showcasing," in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 4469–4472.

[224] J. Zhang, Y. Pan, T. Yao, H. Zhao, and T. Mei, "DaBNN: A super fast inference framework for binary neural networks on ARM devices," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 2272–2275.

[225] J. Fromm, M. Cowan, M. Philipose, L. Ceze, and S. N. Patel, "Riptide: Fast end-to-end binarized neural networks," in *Proc. MLSys*, Mar. 2020, pp. 1–8.

[226] L. Geiger and P. Team, "Larq: An open-source library for training binarized neural networks," *J. Open Source Softw.*, vol. 5, no. 45, p. 1746, Jan. 2020.

[227] T. Zhao, X. He, J. Cheng, and J. Hu, "BitStream: Efficient computing architecture for real-time low-power inference of binary neural networks on CPUs," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 1545–1552.

[228] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015, *arXiv:1512.01274*.

[229] M. Shen, X. Liu, R. Gong, and K. Han, "Balanced binary neural networks with gated residual," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 4197–4201.

[230] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, M. Cowan, H. Shen, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy, "TVM: An automated end-to-end optimizing compiler for deep learning," 2018, *arXiv:1802.04799*.

[231] T. Bannink, A. Bakhtiari, A. Hillier, L. Geiger, T. de Bruin, L. Overweel, J. Neeven, and K. Helwegen, "Larq compute engine: Design, benchmark, and deploy state-of-the-art binarized neural networks," 2020, *arXiv:2011.09398*.

[232] G. Amir, H. Wu, C. Barrett, and G. Katz, "An SMT-based approach for verifying binarized neural networks," in *Tools and Algorithms for the Construction and Analysis of Systems*. Cham, Switzerland: Springer, Mar. 2021, pp. 203–222.

[233] M. Bahri, G. Bahl, and S. Zafeiriou, "Binary graph neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 9492–9501.

[234] Y. Luo and Y. Chen, "FPGA-based acceleration on additive manufacturing defects inspection," *Sensors*, vol. 21, no. 6, p. 2123, Mar. 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2123

[235] Z. Yang, Y. Yang, Y. Xue, F. Y. Shih, J. Ady, and U. Roshan, "Accurate and adversarially robust classification of medical images and ECG time-series with gradient-free trained sign activation neural networks," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Dec. 2020, pp. 2456–2460.

[236] B. Ferrarini, M. J. Milford, K. D. McDonald-Maier, and S. Ehsan, "Binary neural networks for memory-efficient and effective visual place recognition in changing environments," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2617–2631, Aug. 2022.

[237] F. Staudigl, K. J. X. Sturm, M. Bartel, T. Fetz, D. Sisejkovic, J. M. Joseph, L. Bolzani Pöhls, and R. Leupers, "X-fault: Impact of faults on binary neural networks in memristor-crossbar arrays with logic-in-memory computation," 2022, *arXiv:2204.01501*.

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

[239] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25, Dec. 2012, pp. 1097–1105.

[240] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[241] S. Sun, Y. Yin, X. Wang, D. Xu, W. Wu, and Q. Gu, "Fast object detection based on binary deep convolution neural networks," *CAAI Trans. Intell. Technol.*, vol. 3, no. 4, pp. 191–197, Dec. 2018.

[242] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 91–99.

[243] H. Peng and S. Chen, "BDNN: Binary convolution neural networks for fast object detection," *Pattern Recognit. Lett.*, vol. 125, pp. 91–97, Jul. 2019.

[244] S. Xu, Z. Liu, X. Gong, C. Liu, M. Mao, and B. Zhang, "Amplitude suppression and direction activation in networks for 1-bit faster R-CNN," in *Proc. 4th Int. Workshop Embedded Mobile Deep Learn.*, Sep. 2020, pp. 19–24.

[245] Z. Wang, Z. Wu, J. Lu, and J. Zhou, "BiDet: An efficient binarized object detector," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 2049–2058.

[246] S. Xu, J. Zhao, J. Lu, B. Zhang, S. Han, and D. Doermann, "Layer-wise searching for 1-bit detectors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 5682–5691.

[247] J. Zhao, S. Xu, R. Wang, B. Zhang, G. Guo, D. Doermann, and D. Sun, "Data-adaptive binary neural networks for efficient object detection and recognition," *Pattern Recognit. Lett.*, vol. 153, pp. 239–245, Jan. 2022.

[248] H. Nakahara, M. Shimoda, and S. Sato, "A demonstration of FPGA-based you only look once Version2 (YOLOv2)," in *Proc. 28th Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2018, pp. 457–4571.

[249] H. Kim and K. Choi, "The implementation of a power efficient BCNN-based object detection acceleration on a Xilinx FPGA-SoC," in *Proc. Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2019, pp. 240–243.

[250] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Oct. 2016, pp. 21–37.

[251] S. Wang, C. Zhang, D. Su, L. Wang, and H. Jiang, "High-precision binary object detector based on a BSF-XNOR convolutional layer," *IEEE Access*, vol. 9, pp. 106169–106180, 2021.

[252] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128837–128868, 2019.

[253] F. C. Ojeda, A. Bisulco, D. Kepple, V. Isler, and D. D. Lee, "On-device event filtering with binary neural networks for pedestrian detection using neuromorphic vision sensors," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2020, pp. 3084–3088.

[254] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," 2000, *arXiv:physics/0004057*.

[255] V. R. S. Mani, A. Saravanaselvan, and N. Arumugam, "Performance comparison of CNN, QNN and BNN deep neural networks for real-time object detection using Zynq FPGA node," *Microelectron. J.*, vol. 119, Jan. 2022, Art. no. 105319.

[256] K. Chen et al., "MMDetection: Open MMLab detection toolbox and benchmark," 2019, *arXiv:1906.07155*.

[257] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and W. Zisserman, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Sep. 2010.

[258] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Sep. 2014, pp. 740–755.

[259] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3523–3542, Feb. 2021.

[260] B. Zhuang, C. Shen, M. Tan, P. Chen, L. Liu, and I. Reid, "Structured binary neural networks for image recognition," 2019, *arXiv:1909.09934*.

[261] A. Frickenstein, M.-R. Vemparala, J. Mayr, N.-S. Nagaraja, C. Unger, F. Tombari, and W. Stechele, "Binary DAD-Net: Binarized driveable area detection network for autonomous driving," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 2295–2301.

[262] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.

[263] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[264] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.

[265] J. Mayr, C. Unger, and F. Tombari, "Self-supervised learning of the drivable area for autonomous vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 362–369.

[266] Z. Wang, J. Chen, and S. C. H. Hoi, "Deep learning for image super-resolution: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3365–3387, Oct. 2021.

[267] J. Xin, N. Wang, X. Jiang, J. Li, H. Huang, and X. Gao, "Binarized neural network for single image super resolution," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Aug. 2020, pp. 91–107.

[268] X. Jiang, N. Wang, J. Xin, K. Li, X. Yang, and X. Gao, "Training binary neural network without batch normalization for image super-resolution," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, vol. 35, no. 2, pp. 1700–1707.

[269] X. Jiang, N. Wang, J. Xin, K. Li, X. Yang, J. Li, and X. Gao, "Toward pixel-level precision for binary super-resolution with mixed binary representation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 7, 2022, doi: 10.1109/TNNLS.2022.3201528.

[270] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 126–135.

[271] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proc. Brit. Mach. Vis. Conf.* BMVA Press, 2012.

[272] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. 7th Int. Conf. Curves Surf.* Cham, Switzerland: Springer, Jun. 2010, pp. 711–730.

[273] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 2. Jul. 2001, pp. 416–423.

[274] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2015, pp. 5197–5206.

[275] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1646–1654.

[276] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4681–4690.

[277] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2017, pp. 136–144.

[278] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4338–4364, Dec. 2021.

[279] H. Qin, Z. Cai, M. Zhang, Y. Ding, H. Zhao, S. Yi, X. Liu, and H. Su, "BiPointNet: Binary neural network for point clouds," 2020, *arXiv:2010.05501*.

[280] S. Xu, Y. Li, J. Zhao, B. Zhang, and G. Guo, "POEM: 1-bit point-wise operations based on expectation-maximization for efficient point cloud processing," 2021, *arXiv:2111.13386*.

[281] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Process. Mag.*, vol. 13, no. 6, pp. 47–60, Nov. 1996.

[282] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 652–660.

[283] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," 2017, *arXiv:1706.02413*.

[284] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on χ-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 820–830.

[285] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.

[286] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–6.

[287] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.

[288] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3D semantic parsing of large-scale indoor spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1–6.

[289] B. S. Ajay and M. Rao, "Binary neural network based real time emotion detection on an edge computing device to detect passenger anomaly," in *Proc. 34th Int. Conf. VLSI Design 20th Int. Conf. Embedded Syst. (VLSID)*, Feb. 2021, pp. 175–180.

[290] N. Fasfous, M.-R. Vemparala, A. Frickenstein, L. Frickenstein, M. Badawy, and W. Stechele, "BinaryCoP: Binary neural network-based COVID-19 face-mask wear and positioning predictor on edge devices," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, Jun. 2021, pp. 108–115.

[291] A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi, "MaskedFace-Net—A dataset of correctly/incorrectly masked face images in the context of COVID-19," *Smart Health*, vol. 19, Mar. 2021, Art. no. 100144.

[292] K. Brahma, V. Kumar, A. E. Samir, A. P. Chandrakasan, and Y. C. Eldar, "Efficient binary CNN for medical image segmentation," in *Proc. IEEE 18th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2021, pp. 817–821.

[293] K. Shridhar, H. Jain, A. Agarwal, and D. Kleyko, "End to end binarized neural networks for text classification," in *Proc. SustaiNLP, Workshop Simple Efficient Natural Lang. Process.*, 2020, pp. 29–34.

[294] G. Cerutti, L. Cavigelli, R. Andri, M. Magno, E. Farella, and L. Benini, "Sub-mW keyword spotting on an MCU: Analog binary feature extraction and binary neural networks," 2022, *arXiv:2201.03386*.

[295] H. Qin, X. Ma, Y. Ding, X. Li, Y. Zhang, Y. Tian, Z. Ma, J. Luo, and X. Liu, "BiFSMN: Binary neural network for keyword spotting," 2022, *arXiv:2202.06483*.

[296] A. Saeed, "Binary early-exit network for adaptive inference on low-resource devices," 2022, *arXiv:2206.09029*.

[297] F. Daghero, C. Xie, D. J. Pagliari, A. Burrello, M. Castellano, L. Gandolfi, A. Calimera, E. Macii, and M. Poncino, "Ultra-compact binary neural networks for human activity recognition on RISC-V processors," in *Proc. 18th ACM Int. Conf. Comput. Frontiers*, May 2021, pp. 3–11.

[298] F. Luo, S. Khan, Y. Huang, and K. Wu, "Binarized neural network for edge intelligence of sensor-based human activity recognition," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1356–1368, Sep. 2021.

[299] S. Xu, C. Liu, B. Zhang, J. Lü, G. Guo, and D. Doermann, "BiRe-ID: Binary neural network for efficient person re-ID," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 18, no. 1s, pp. 1–22, Feb. 2022, doi: 10.1145/3473340.

[300] W. Jing, X. Zhang, J. Wang, D. Di, G. Chen, and H. Song, "Binary neural network for multispectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, pp. 1–5, 2022.

[301] H. Zhu, S. Guo, W. Sheng, and L. Xiao, "SBNN: A searched binary neural network for SAR ship classification," *Appl. Sci.*, vol. 12, no. 14, p. 6866, Jul. 2022.

[302] L. Tong, Y. Chen, T. Xu, and Y. Kang, "Fault diagnosis for modular multi-level converter (MMC) based on deep learning: An edge implementation using binary neural network," *IEEE J. Emerg. Sel. Topics Power Electron.*, early access, Jul. 29, 2022, doi: 10.1109/JESTPE.2022.3194974.

RATSHIH SAYED received the B.S. degree from the Electronics and Communications Engineering Department, Banha University, Egypt, in 2009, and the master's degree in electronics and communications engineering from Cairo University, Egypt, in 2016, where she is currently pursuing the Ph.D. degree in electronics and communications engineering. She works with the Electronics Research Institute, Cairo, Egypt. Her research interests include digital design and machine learning.

HAYTHAM AZMI is currently a Researcher with the Microelectronics Department, Electronics Research Institute, Cairo, Egypt. His research interests include the design and implementation of digital integrated circuits and the design and development of electronic design automation tools to improve the continuous integration of electronic circuits at the micro- and nanoscale technologies.

HEBA SHAWKEY has been working with the Microelectronics Department, Electronics Research Institute, as a Researcher, since 1994, where she is currently a Professor. She also has many publications in low-power digital circuit designs, biomedical sensors read-out circuits, interconnect modeling, and low-power networks on chip (NoC). Her research interests include wireless communication systems design, analog/mixed-signal/RF circuits, and especially high-speed PLL and frequency synthesizers.

A. H. KHALIL was born in Cairo, Egypt, in 1960. He received the B.Sc. degree in electronics and electrical communications engineering from Cairo University, Giza, Egypt, in 1983, and the M.Sc. and Ph.D. degrees from the Electronics and Electrical Communications Engineering (EECE) Department, Faculty of Engineering, Cairo University, in 1987 and 1992, respectively. He has been a Faculty Member of Cairo University, since 1983, and has been the Vice Director of the Design Laboratory for Electronics and Communication Systems (DLECS), since 2006. He is currently a Professor with the EECE Department, Faculty of Engineering, Cairo University. He teaches several courses on analog and digital electronics and signal processing with Cairo University and other universities and institutes in Egypt. He has been in the Electronics and Communications Industry, with over 32 years of hands-on practical knowledge. His research interests include signal processing, embedded systems, and FPGA-based systems using a broad range of tools and technologies on diverse types of platforms with a solid track record of supervising many mega projects and experience in electronic design and technology integration.

MOHAMED REFKY received the bachelor's and master's degrees from the Electronics and Electrical Communications Engineering (EECE) Department, Cairo University, Giza, Egypt, in 2004 and 2008, respectively, and the Ph.D. degree from the Electrical and Computer Engineering (ECE) Department, University of Waterloo, Canada, in 2012. He is currently an Assistant Professor with the EECE Department, Faculty of Engineering, Cairo University. His research interests include the design of data converters [analog-to-digital converters (ADC) and digital-to-analog converters (DAC)], energy harvesting, and power management.