XNOR-BSNN: In-Memory Computing Model for Deep Binarized Spiking Neural Network

Abstract— **This paper proposed a rate encoding binarized spiking neural network (B-SNN) model suited for in-memory computing (IMC) implementation. While in most of the prior arts, due to the nature of spike represented unipolar format, the B-SNN were implemented using either complex or non-regular logic that is not suited for IMC and/or makes the network inflexible. In this work, we proposed a B-SNN model that permits the direct adoption of a unipolar format spike on the XNOR array, i.e., allows fully exploiting IMC's potential benefit based on the highly regular and simple array structure. Also, instead of indirectly taking the B-SNN model from the trained BNN, we propose to train B-SNN directly using the surrogate gradient method with residual connections. The system simulation shows that our proposed trained network achieves reasonably good accuracy (59.11%) on CIFAR-100 with very low inference latency (only 8 time-steps B-SNN).**

Keywords—***In-memory computing, Binary Spiking Neural Network, surrogate gradient***

I. INTRODUCTION

Multiply-addition-accumulation (MAC) has been the dominant computational workload for deep-neural-network (DNN) processors. This type of computation not only computing but memory intensive. Therefore, the conventional computer architecture with limited memory bandwidth and sequential computing nature is not ideal for AI applications. It is a real challenge to accommodate DNN in on-edge artificial-intelligence (AI) devices like Internet-of-Thing (IoT) or mobile systems, which have strict resource and power consumption constraints.

BNN, first introduced by M. Courbariaux et. al [1], is a type of network whose weights and inputs are represented by the bi-polar value ±1, simplifying the multiplication to be the bitwise function. BNN hence is particularly suited for resource- and area-constrained devices. Nonetheless, the BNN network size, compared to the conventional one, needs to be increased accordingly to compensate for the low-precision in bi-polar presentation. The latter could lead to the surging up of the number of memory access. In-memory computing (IMC), one of the recent revolutionized approaches to solving the current computer architecture's memory bottleneck [2]. This approach proposed to partially shift the processing from the central processing unit to the processing in-memory, which greatly reduces memory access and increases performance and energy efficiency. Nonetheless, the accuracy of BNNs strongly depends on the process variations, which could be quite severe in finer technologies [3].

The spiking neural network has been recently introduced in [4], [5] with resilience, robustness, and fault tolerance capability against the process variation. Furthermore, binarized spiking neural network (B-SNN) is introduced with binary weights and binary spikes [6], [7], enabling the utilization of MAC within in-memory. The authors in these works introduced the conversion of the trained BNN model into B-SNN, where ReLU activation is normalized to be to IF spiking activity. Their proposed S-BNNs achieve comparable accuracy compared to BNN with equivalent network topologies. In [8], the author presents a directly supervised learning algorithm to train B-SNN with the temporal encoding method. Their evaluation obtained 97.0% and 87.3% misclassification accuracy for MNIST and Fashion MNIST, respectively. In [9], the author train the residual stochastic binary convolutional spiking neural network with hybrid spike-timing-dependent plasticity to achieve 66% for the CIFAR-10 dataset. However, these techniques require a large number of time-steps (~100-300) for inference, which leads to high latency and limited energy efficiency on the actual hardware implementation.

The surrogate gradient method is recently introduced to mitigate the inherent non-differentiability of spiking during the backpropagation training process. Studies in [10], [11] demonstrated that training the conventional spiking neural networks using surrogate gradients achieves reasonable accuracy using fewer time-steps. In this work, we proposed a novel method for B-SNN training that achieves an accuracy of 59.11% for CIFAR-100 with only 8 time-steps, approximating 13X lower than previous art (105 time-steps) in [6]. Furthermore, we propose a novel MAC model using only the XNOR array. Compared to the implementation in [12], where the possible product output belongs to the set of (0,+1,-1), therefore fundamental multiplication unit is realized using a non-standard dual output gate, i.e., more complex than XNOR gate. Our proposed model permits the full B-SNN model can be mapped to an existing common in-memory architecture based on the XNOR array.

The remainder of this paper is structured as follows. Section II present the training B-SNN with surrogate gradient. Section III details the B-SNN inference IMC model. The experiment evaluation and results comparison is presented in Section IV. Finally, Section V concludes the paper.

II. Proposed B-SNN Using Surrogate Gradient

In this section, we proposed directly training B-SNN with the surrogate gradient method. In our model, the weights of B-SNN are represented in bipolar format (i.e., ±1) [13]. In the general SNN model using the Integrated-and-Fired (IF) model, the membrane potential $u_i^{t,l}$ at *i*-th neuron time-step t at intermediate layer l-th is defined as follows:

$$u_i^{t,l} = u_i^{t-1,l} + \sum_{j=1}^{M} w_{ij} o_j{}^{t,l} \tag{1}$$

Where M denotes the number of pre-synaptic neurons, $o_j{}^{t,l}$ is the pre-synaptic spike in the j-th neuron, w_{ij} is the weight that links the pre-and the post-neurons. If the membrane potential $u_i^{t,l}$ surpasses the firing threshold θ, the IF model generates a binary spike output $o_i{}^{t,l}$. In this work, we use soft reset, i.e., the membrane potential is subtracted by a threshold if a neuron is fired. In the output layer, the membrane potential $u_i^{T,L}$ in the output layer L at final time-step T, is accumulated without firing, calculates probability distribution after softmax function without information loss [10]. From the accumulated membrane potential, the cross-entropy loss for B-SNN is defined as

$$L = -\sum_{i=1}^{C} y_i log\left(\frac{e^{u_i^{T,L}}}{\sum_{k=1}^{C} e^{u_k^{T,L}}}\right) \tag{2}$$

Here, $Y = (y_1, y_2, \dots, y_C)$ is a label vector, and T is a total number of time-steps. The partially derivative of the loss function with respect to the membrane potential $u_i^{t,l}$ at the layer l is defined as follows

$$\frac{\partial L}{\partial u_i^{t,l}} = \frac{\partial L}{\partial o_i^{t,l}} \frac{\partial o_i^{t,l}}{\partial u_i^{t,l}} + \frac{\partial L}{\partial u_i^{t+1,l}} \frac{\partial u_i^{t+1,l}}{\partial u_i^{t,l}} \tag{3}$$

Due to the non- differentiable spiking activities, $\frac{\partial o_i^{t,l}}{\partial u_i^{t,l}}$ does not exist. To deal with this problem, the authors in [10] introduce an approximate gradient (i.e., surrogate gradient) for SNN training, which is expressed as follows

$$\frac{\partial o_i^{t,l}}{\partial u_i^{t,l}} = \delta \, max\left\{0, 1 - \left|\frac{u_i^{t,l} - \theta}{\theta}\right|\right\} \tag{4}$$

Here, δ is a damping factor for back-propagated gradients, which is empirically set to 0.3 for stable training [10]. Algorithm 1 presented below demonstrates our procedure for training B-SNN with binary weights. Note that our B-SNN does not binarize the first and the last layer as in some previous works (BNN [13], B-SNN [6]). Except for the full-precision weights in the first layer w^1, and the last layer w^L, the dot-product computation $x^{t,l}$ at layer l -th (in Conv block as shown in Fig. 1) between the weights and the pre-synaptic spike is equal to

$$x^{t,l} = \alpha w_{ij}^{b,l} \{o_j^{t,l-1} + f o_j^{t,l-2}\} \tag{5}$$

Where α is a scaling factor and $w_{ij}^{b,l}$ is binary weights that link layer $l-1$ to layer l. In (5) we also apply the identity mapping technique for residual connections in [9] to our model. The convolutional layer l receives residual connections from the pre-synaptic spike in layers $l-1$ and

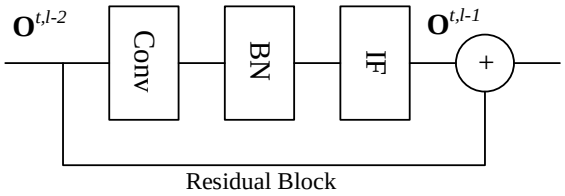

Fig. 1. Residual connections from layer $l-2$, $l-1$ to l [9]

$l-2$. The index f represents the residual connection ($i.e.\ f = 1$ implies the presence of the residual connection, otherwise $f = 0$). The input data is encoded using rate encoding, supported by the *Poisson Generator* function [10]. The batch normalization (BN) is set with two parameters: variance σ, and learn parameters μ and are updated during the training process.

Algorithm 1 The direct B-SNN training algorithm using surrogate gradient

Input: input X, label vector Y

Output: parameters in layer n: $w^1, w^L, \alpha, w_{ij}^{b,l}, \theta$

function $IF\ (u, I)$
  for $t = 1\ to\ T$ **do**
    $u^t = u^{t-1} + I(t)$
  end for
  if $u^t \le \theta$
    $o_i{}^t = 0$
  else
    $o_i{}^t = 1$
    $u^t = u^t - \theta$
  end if
end function

In the Training: % Forward:

for $l = 1$ **do**
  for $t = 1\ to\ T$ **do**
    $o^t = Poisson\ Generator\ (X)$
    $x^{t,1} = w^1 o^t$
    $y^{t,1} = \frac{x^{t,1} - \mu}{\sigma}$
  end for
  $o_i^{t,1} \leftarrow IF\ (u^1, y^{t,1})$
end for
for $l = 2\ to\ L-1$ **do**
  for $t = 1\ to\ T$ **do**
    $x^{t,l} = \alpha w_{ij}^{b,l} \{o_j^{t,l-1} + f o_j^{t,l-2}\}$
    $y^{t,l} = \frac{x^{t,l} - \mu}{\sigma}$
  end for
  $o_i^{t,l} \leftarrow IF\ (u^l, y^{t,l})$
end for
for $l = L$ **do**
  for $t = 1\ to\ T$ **do**
    $x^{t,L} = w^L o^{t,L-1}$
    $y^{t,L} = \frac{x^{t,L} - \mu}{\sigma}$
    $u^{t,L} = u^{t-1,L} + y^{t,L}$
  end for
end for

% Calculate the loss and back-propagation

$L \leftarrow Compute\ Loss(\ Y, u^{T,L})$

$\frac{\partial L}{\partial u_i^{t,l}}, \frac{\partial L}{\partial o_i^{t,l}} \leftarrow Autograd$

III. Proposed Binarized Spiking Neural Network Model Based on The In-memory XNOR Array

This section proposes a MAC model for the accumulated bit-wise product in B-SNN inference using the XNOR cell circuit, which is common in the IMC model as shown in Algorithm 2.

First, we analyze with $f = 0$ on steps (11-12) in Algorithm 1. We have the IF model now is expressed as

$$\begin{aligned} &\textbf{\textit{Integration}}: u_i^{t,l} = u_i^{t-1,l} + \frac{\alpha}{\sigma}\left\{\sum_{j=1}^{M} w_{ij}^{b,l}\, o_j^{t,l-1} - \frac{\mu}{\alpha}\right\} \\ &\textbf{\textit{Firing}}: o_i^{t,l} = \begin{cases} 1, & \text{if } u_i^{t,l} > \theta \\ 0, & \text{otherwise} \end{cases} \\ &\textbf{\textit{Reset}}: u_i^{t,l} = u_i^{t,l} - \theta \end{aligned} \tag{6}$$

To summarize (6), for every time-step, the membrane potential $u_i^{t,l}$ is accumulated with ${}^{\alpha}/_{\sigma}\{\sum_{j=1}^{M} w_{ij}^{b,l}\, o_j^{t,l-1} - {}^{\mu}/_{\alpha}\}$, then compared with threshold θ for firing decision. After firing, the membrane is subtracted by θ in the reset phase.

We can reformulate the integration phase in (6) as

$$\hat{u}_i^{t,l} = \hat{u}_i^{t-1,l} + \sum_{j=1}^{M} w_{ij}^{b,l}\, o_j^{t,l-1} - \frac{\mu}{\alpha} \tag{7}$$

In (7), $\hat{u}_i^{t,l}$ is the membrane potential transformation, $\hat{\theta} = {}^{(\sigma \cdot \theta)}/_{\alpha}$ is a threshold transformation.

To compute the MAC operation $\sum_{j=1}^{M} w_{ij}^{b,l}\, o_j^{t,l-1}$, the prior works [14] proposed to separate the calculation into negative and positive phases of weight. In detail, M weights $w_{ij}^{b,l}$ are divided into M_1 negative weights w_{ij}^{-b} (-1) and M_2 positive weights w_{ij}^{+b} (+1). Then, the MAC is seperated into two sub-MAC operation $\sum_{j=1}^{M_1} w_{ij}^{-b}\, o_j^{t,l-1}$ and $\sum_{j=1}^{M_2} w_{ij}^{+b}\, o_j^{t,l-1}$. The MAC product $s_i^{t,l}$ in B-SNN synapse can be derived as follows.

$$\begin{aligned} s_i^{t,l} &= \sum_{j=1}^{M} w_{ij}^{b,l}\, o_j^{t,l-1} \\ &= \sum_{j=1}^{M_1} w_{ij}^{-b}\, o_j^{t,l-1} + \sum_{j=1}^{M_2} w_{ij}^{+b}\, o_j^{t,l-1} \\ &= -\sum_{j=1}^{M_1} \left|w_{ij}^{-b}\right| \wedge o_j^{t,l-1} + \sum_{j=1}^{M_2} w_{ij}^{+b} \wedge o_j^{t,l-1} \end{aligned} \tag{8}$$

As can be seen from (8), the direct implementation by AND functions with separation into the positive and negative sub-MAC. However, the hardwired separation in the circuit level means that the circuit is not reconfigurable or upgradable when there is a change in the network models (e.g., problem changes or retraining). This rigid design essentially has little meaning in practice. Therefore, take the fact that $w_{ij}^{\pm b} = \pm 1$, we further transform the first and the second MAC product component to be

Algorithm 2 Proposed B-SNN inference model for IMC implementation.

PARAMETERS: $w_{ij}^{b,l}$, $\hat{\theta}$, M, M_1, μ, α
INPUT: $o_j^{t,l-1}$
OUTPUT: $o_i{}^{t,l}$
△ **In-memory MAC computation**
for $t \leftarrow 1$ *to* T **do**
  for $j \leftarrow 1$ *to* M **do**
    $dot_i^{t,l} = \sum_{j=1}^{M} \overline{w_{ij}^{b,l} \oplus o_j^{t,l-1}}$
  end for
end for
△ **Accumulation phase**
for $t \leftarrow 1$ *to* T **do**
  $\hat{u}_i^{t,l} = \hat{u}_i^{t-1,l} + dot_i^{t,l} - \left[M_1 + \frac{\mu}{\alpha}\right]$
end for
△ **Firing & reset phase**
for $t \leftarrow 1$ *to* T **do**
  if $\hat{u}_i^{t,l} > \hat{\theta}$ **then**
    $o_i{}^{t,l} \leftarrow 1$
    $\hat{u}_i^{t,l} \leftarrow \hat{u}_i^{t,l} - \hat{\theta}$
  else
    $o_i{}^{t,l} \leftarrow 0$
    $\hat{u}_i^{t,l} = \hat{u}_i^{t-1,l} + dot_i^{t,l} - \left[M_1 + \frac{\mu}{\alpha}\right]$
  end if
end for

$$\begin{cases} -\sum_{j=1}^{M_1} \left|w_{ij}^{-b}\right| \wedge o_j^{t,l-1} = \sum_{j=1}^{M_1} \left(1 - o_j^{t,l-1}\right) - M_1 \\ \qquad = \sum_{j=1}^{M_1} \overline{(1 + w_{ij}^{-b}) \oplus o_j^{t,l-1}} - M_1 \\ \sum_{j=1}^{M_2} w_{ij}^{+b} \wedge o_j^{t,l-1} = \sum_{j=1}^{M_2} \overline{w_{ij}^{+b} \oplus o_j^{t,l-1}} \end{cases} \tag{9}$$

Finally, if $w_{ij}^{b,l}$ are encoded in unipolar number (-1 into 0, 1 into 1), $s_i^{t,l}$ from (8) can be expressed as

$$\begin{aligned} s_i^{t,l} &= \sum_{j=1}^{M_1} \overline{(1 + w_{ij}^{-b}) \oplus o_j^{t,l-1}} - M_1 \\ &+ \sum_{j=1}^{M_2} \overline{w_{ij}^{+b} \oplus o_j^{t,l-1}} = \sum_{j=1}^{M} \overline{w_{ij}^{b,l} \oplus o_j^{t,l-1}} - M_1 \end{aligned} \tag{10}$$

Express of $s_i^{t,l}$ allows computing MAC ($dot_i^{t,l}$) as mentioned in Algorithm 2 using bit-wise operation, which can be realized entirely using IMC XNOR array as proposed in [3], [15]. Then, the MAC product is biased by $M_1 + \frac{\mu}{\alpha}$ before compared with $\hat{\theta}$ for firing decision.

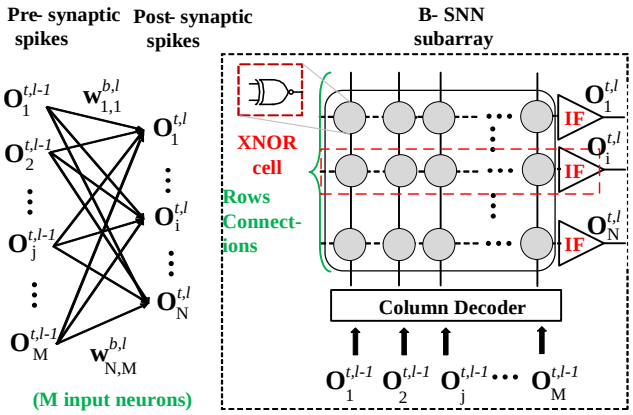

Fig. 2. B-SNN model and mapping onto in-memory architecture using XNOR cells array.

Suppose the residual connection exists ($f = 1$). Exploiting the fact that the weights are shared between spiking in *l-1* ($o_j^{t,l-1}$) and *l-2* layers($o_j^{t,l-2}$), so the MAC operation in step (6) in Algorithm 2 now consists of two MAC components.

$$dot_i^{t,l} = \sum_{j=1}^{M} \overline{w_{ij}^{b,l} \oplus o_j^{t,l-1}} + \sum_{j=1}^{M} \overline{w_{ij}^{b,l} \oplus o_j^{t,l-2}} \qquad (11)$$

For hardware implementation, we can compute the first and second MAC in (11) in the same hardware model using the technique of the shared weight in a time-interleaved manner. Specifically, the time-step t is divided into two time slots τ_1 and τ_2. We can execute the computation of the first MAC in the time slot τ_1, and store their product in the analog buffer (e.g., a simple capacitive buffer). After that, the second MAC is performed in the time slot τ_2 and subsequently added into the first MAC product's pre-stored in the buffer. As a result, the residual connections do not increase the hardware complexity for the proposed IMC model in Algorithm 2.

The final proposed B-SNN subarray model size (N, M) with pre-synaptic spikes $o_1^{t,l-1}, o_2^{t,l-1}, \dots o_M^{t,l-1}$ $(j = 1 \div M)$, and post-synaptic spikes $o_1^{t,l}, o_2^{t,l}, \dots o_N^{t,l}$ $(i = 1 \div N)$ of the output neurons shown in Fig. 2. In the $i-th$ row connection (red line in Fig. 2) is described the MAC between the synaptic weight $w_{ij}^{b,l}$ and the pre-synaptic spikes $o_j^{t,l-1}$ $\sum_{j=1}^{M} \overline{w_{ij}^{b,l} \oplus o_j^{t,l-1}}$, which is presented in step (6) in Algorithm 2. In this topology, the synaptic weight $w_{ij}^{b,l}$ links pre-synaptic spike $o_j^{t,l}$ with the post-synaptic spike $o_i^{t,l}$ (i, j is a row and column index, respectively). In detail, the pre-synaptic spikes of a given layer are applied to the column decoder. The bit-wise products between pre-synaptic spikes with the respective weights stored in a row are summed up and then accumulated in the IF model. If the firing condition in Algorithm 2 is satisfied, the post-synaptic spikes are generated for the next layer processing.

IV. Results and Discussion

We evaluate our method on CIFAR-100 [16] , consisting of 60,000 images with 100 categories that are divided into 50,000 for training and 10,000 for testing. All images are RGB color images whose size is 32x32. The B-SNN architecture on training is VGG11 (with the residual mapping schedule applied for every 3 consecutive layers: 1-2-3, 2-3-4, 3-4-5, 4-5-6, 5-6-7, 7-8-9 as depicted in Fig. 1).

TABLE I. Classification Accuracy Of B-SNN Model On The CIFAR-100 Test Set

Net. Structure	Bit width	Accuracy (%)	Timesteps
8-layer CNN [7]	binary	62.02	300
15-layer CNN [6][a]	binary	62.00	105
Spiking CNN [17]	ternary	55.64	N.A
11-layer CNN [18]	(0, ±1)	55.95	8
11-layer CNN (ours)	**binary**	**59.11**	**8**

[a] There is the baseline accuracy of BNN-SNN conversion without early exit optimization

Our results are shown in Table 1. In comparison with the most representative recent reports on the same dataset: BNN to B-SNN conversion [6] [7], a conventional SNN with ternary bit width in [17], and an SNN with 11-layer CNN based weight quantization (0, ±1) in [18].

In comparison to the prior B-SNN, although the misclassification accuracy in these works is slightly better than ours trained B-SNN (by 2.89% [6] and 2.91% [7]), our spiking time-steps is much lower (8 compared to 105 and 300). The latter leads to not only lower inference latency but also a significant reduction in energy consumption.

In comparison with [17], and [18] our model exhibits better classification accuracy (by 3.47% [17] and 3.16% [18]). From the hardware implementation perspective, the B-SNN essentially is highly energy-and area-efficient and suited for the edge-AI device compared to the ternary SNN and the weight quantization (0, ±1) SNN.

V. Conclusion and Discussion

This paper proposes a novel and effective method for direct training B-SNN, which greatly shrinks the memory space and time-steps for inference. Our model could achieve reasonably good accuracy with the CIFAR-100 dataset while requires much fewer time steps compared to other B-SNN models. Furthermore, we propose the IMC MAC model for B-SNN, which allows performing MAC operation in-memory using only the XNOR array. The proposal IMC MAC model and the proposed B-SNN together greatly simplify the hardware implementation and pave the way for ultra-low-power deep neural network applied in edge-AI applications.

References

[1] M. Courbariaux and et al., "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv*, 2016.

[2] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature nanotechnology*, pp. 529-544, 2020.

[3] T. Pham, Q. Trinh, I. Chang and M. Alioto, "STT-MRAM Architecture with Parallel Accumulator for In-Memory Binary Neural Networks," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021.

[4] X. She, Y. Long and S. Mukhopadhyay, "Improving robustness of reram-based spiking neural network accelerator with stochastic spike-timing-dependent-plasticity," in *International Joint Conference on Neural Networks (IJCNN)*, 2019.

[5] W. Guo, M. Fouda, A. Eltawil and K. Salama, "Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems," *Frontiers in Neuroscience*, 2021.

[6] S. Lu and A. Sengupta, "Exploring the connection between binary and spiking neural networks," *Frontiers in Neuroscience*, p. 535, 2020.

[7] Y. Wang, Y. Xu, R. Yan and H. Tang, "Deep spiking neural networks with binary weights for object recognition," *IEEE Transactions on Cognitive and Developmental Systems*, 2020.

[8] S. Kheradpisheh, M. Mirsadeghi and T. Masquelier, "BS4NN: Binarized Spiking Neural Networks with Temporal Coding and Learning," *arXiv preprint arXiv:2007.04039*, 2020.

[9] G. Srinivasan and K. Roy, "Restocnet: Residual stochastic binary convolutional spiking neural network for memory-efficient neuromorphic computing," *Frontiers in neuroscience*, p. 13, 2019.

[10] Y. Kim and P. Panda, "Revisiting Batch Normalization for Training Low-latency Deep Spiking Neural Networks from Scratch," *arXiv preprint arXiv:2010.01729*, 2020.

[11] Y. Wu and L. Deng, et al, "Direct training for spiking neural networks: Faster, larger, better," in *In Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[12] P. Chuang, P. Tan, C. Wu and J. Lu, "A 90nm 103.14 TOPS/W binary-weight spiking neural network CMOS ASIC for real-time object classification," in *ACM/IEEE Design Automation Conference (DAC)*, 2020.

[13] M. Rastegari, V. Ordonez, J. Redmon and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision* , 2016.

[14] J. Yu and K. Kim, et al., "Accurate and efficient stochastic computing hardware for convolutional neural networks," in *2017 IEEE International Conference on Computer Design (ICCD)*, 2017.

[15] M. Abu Lebdeh, H. Abunahla, B. Mohammad and M. Al-Qutayri, "An Efficient Heterogeneous Memristive xnor for In-Memory Computing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 2427-2437, 2017.

[16] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images.," 2009.

[17] S. Esser and P. Merolla, et al, "Convolutional networks for fast, energy-efficient neuromorphic computing," in *Proceedings of the national academy of sciences*, 2016.

[18] L. Deng and Y. Wu, et al, "Comprehensive snn compression using admm optimization and activity regularization," *arXiv preprint arXiv*, 2019.