HAWIS: Hardware-Aware Automated WIdth Search for Accurate, Energy-Efficient and Robust Binary Neural Network on ReRAM Dot-Product Engine

Qidong Tang[1], Zhezhi He[1,*], Fangxin Liu[1], Zongwu Wang[1], Yiyuan Zhou[1], Yinghuan Zhang[2], Li Jiang[1,2,3,*]
[1] *Shanghai Jiao Tong University, Shanghai, China,* [2] *Shanghai Qi Zhi Institute, Shanghai, China*
[3] *MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China*

Abstract— Binary Neural Networks (BNNs) have attracted tremendous attention in ReRAM-based Process-In-Memory (PIM) systems, since they significantly simplify the hardware-expensive peripheral circuits and memory footprint. Meanwhile, BNNs are proven to have superior bit error tolerance, which inspires us to make use of this capability in PIM systems whose memory bit-cell suffers from severe device defects. Nevertheless, prior works of BNN do not simultaneously meet the criterion that 1) achieving similar accuracy w.r.t its full-precision counterpart; 2) fully binarized without full-precision operation; and 3) rapid BNN construction, which hampers its real-world deployment. This work proposes the first framework called HAWIS, whose generated BNN can satisfy all the above criteria. The proposed framework utilizes the super-net pre-training technique and reinforcement-learning based width search for BNN generation. Our experimental results show that the BNN generated by HAWIS achieves 69.3% top-1 accuracy on ImageNet with ResNet-18. In terms of robustness, our method maximally increases the inference accuracy by 66.9% and 20% compared to 8-bit and baseline 1-bit counterparts under ReRAM non-ideal effects. Our-code is available at: https://github.com/DamonAtSjtu/HAWIS.

I. Introduction

The emerging Resistive Random-Access Memory (ReRAM) is a promising candidate for future Neural Network (NN) accelerators. It supports vector-matrix multiplications in the analog domain, resulting in over 100× energy efficiency improvements than the Von-Neumann system [3]. However, there are three challenges to deploy NN on ReRAM accelerators: 1) the processing technology of ReRAM to provide multi-bit precision is still commodity immature [17]; 2) The peripheral circuits (e.g., digital-to-analog converters) consume a great portion of the on-chip area and energy [15]; 3) The inference accuracy is vulnerable to the various device defects, such as resistance variation and Stuck-At-Fault (SAF) [5].

Binary Neural Network (BNN), which constrains the weights and activations to two levels (i.e., -1 or +1), is a promising solution to overcome these challenges of ReRAM. Prior works have proved that BNNs could effectively lower the hardware overhead in terms of energy, area, etc. [17, 18]. Beyond that, BNN reveals superior bit error tolerance [13], expected to counter the non-ideal effects in ReRAM crossbars.

Network binarization, nevertheless, is also accompanied by the following drawbacks: 1) Even with a series of advanced optimization techniques applied (e.g., minimizing the quantization error [12], reducing the gradient error [9]), BNNs still face a drastic accuracy degradation w.r.t its full-precision (FP) counterpart. A state-of-the-art work [12] reports the binarization causes 11.5% accuracy drop of ResNet-18 on ImageNet,

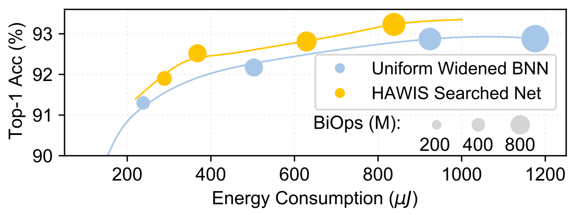


Fig. 1. HAWIS lies on a better Pareto curve (yellow) than uniformly widened BNNs (blue). BiOps denote #(Binary Operations).

which hampers the adoption of BNNs in real-world applications; 2) BNNs introduced in the prior literature [9, 12, 24] still contain a great portion of non-binarized computation; for example, the first and the last layer are not binarized. Thus, they require computing units with FP computation capability. Besides, applying binarization on the remained FP computation operands will further lower the accuracy to an unacceptable level (e.g., ∼24.2% accuracy drop of ResNet-18 on ImageNet based on our experimental results in Table I).

As the countermeasure, [11] proposes to widen the neural network to mitigate the accuracy drop caused by DNN quantization, which is one of the most effective solutions yet. Unfortunately, it uniformly widens the target DNN (i.e., same expansion ratio across the network), leading to model overfitting. One recent work [16] attempts to widen the neural network non-uniformly (i.e., specific expansion ratio for each layer) via an evolution algorithm. However, to reduce the searching cost, a small searching space including only 6 options of expansion ratio is adopted by [16]. Nevertheless, the searching cost is still high (60 GPU days) and will further increase with finer-grain expansion ratios.

To summarize the above-mentioned prior efforts in building high accuracy BNN, there still lacks a work of BNN that can achieve the following objectives simultaneously:

O1. *Similar accuracy w.r.t to the FP counterpart.*
O2. *Fully binarized that can be executed on ReRAM crossbar without additional computing units for FP operations.*
O3. *Rapid and hardware-aware BNN construction (i.e., searching and training).*

In this work, we bring up the framework called **HAWIS** to enable hardware-aware automated width search for accurate, energy-efficient and robust BNN on ReRAM dot-product engine, with all three objectives above satisfied. Our contributions in this work can be enumerated as follows:

- As far as we know, this work is the first BNN satisfying objectives (O1-O3) simultaneously.
- We propose *network re-factorization* and *initialization pre-training* to prepare a well-trained super-net for rapid

and precise performance evaluation. Based on the super-net, we propose the search framework based on reinforcement learning, with fine-grain layer-wise expansion ratio searching space .

- Experimental results show that HAWIS achieves 69.3% top-1 accuracy on ImageNet with ResNet-18. The BNNs generated by HAWIS achieve superior accuracy and energy tradeoff shown in Fig. 1. In terms of robustness, our method maximally increases the inference accuracy by 66.9% and 20% compared to 8-bit and baseline 1-bit counterparts under ReRAM non-ideal effects.

II. Preliminaries

A. Neural Network Binarization

Network binarization is the extreme low bit-width DNN quantization scheme in which weights and activations are compressed into 1 bit. In DoReFa [24], inputs $\mathcal{I}^{\text{fp}}$ are approximated by the binary value $\mathcal{I}^{\text{b}} = \text{round}(\text{clip}(\mathcal{I}^{\text{fp}}, 0, 1))$, while weights $\mathcal{W}^{\text{fp}}$ are represented by binary value $\mathcal{W}^{\text{b}} = \text{sign}(\mathcal{W}^{\text{fp}}) \cdot \mathbb{E}(|\mathcal{W}^{\text{fp}}|)$. We leverage this plain binarization scheme as it brings no extra FP computation on ReRAM-based NN accelerators. More importantly, the activation binarization and the Batch-Normalization (BN) can be fused and then realized by a simple comparison operation [20]. Most previous BNN works [9, 12, 24] still remain non-negligible FP computations, such as the non-binarized input and output layer. These non-binarized operations require separate FP processing units and may consume a large part of the on-chip area and energy, which is what we try to avoid.

B. Efficient Neural Architecture Search in BNN

Neural Architecture Search (NAS) is committed to finding efficient neural architectures automatically. However, adapting the identified networks to resource-limited platforms is still inefficient due to the FP or multi-bit computing. Binarized NAS (BNAS) explores the advantages of binarized operations on memory saving and computational cost reduction to address these issues. On the one hand, some BNAS methods inherit traditional NAS and focus on searching the cell-based network topology. [2, 6] formulate new cell-based search spaces and new search strategies as they observe that traditional NAS methods can not be directly applied for BNN search. On the other hand, some methods adaptively optimize the width of BNNs based on a baseline. [16] searches for the optimal width for each layer through an evolutionary algorithm. However, to reduce the search cost, they adopt an extremely limited searching space with 6 expansion ratios, which causes sub-optimal results. One solution is to make the expansion ratio continuous, but evolution cannot efficiently support this.

C. Slimmable Networks

Slimmable networks [21] are a family of neural networks that can instantly adjust the width on-the-fly. Given the lower bound r_{min} and upper bound r_{max} of width expansion ratio, n sub-models $\hat{\mathcal{A}}_i$ are sampled sequentially to update the slimmable network $\mathcal{A}$ with dataset $(\mathbf{X}, \boldsymbol{y})$:

$$\min \sum_i \Big(\mathcal{L}(f(\hat{\mathcal{A}}_i, \mathbf{X}), \boldsymbol{y})\Big) \quad i \in (1, 2, 3, \cdots, n) \tag{1}$$

Instead of assigning the same expansion ratio within a sub-model as [21], we allow layer-specific expansion ratios to create a non-uniform slimmable super-net. The initially well-trained super-net provides proper initialization for candidate models during the search to avoid training them from scratch.

D. Crossbar Array and Device Defects

ReRAM is a two-terminal device with programmable resistance ranging from low resistance state R_{min} to high resistance state R_{max}. It can be utilized as an analog matrix-vector multiplication engine with high parallelism and efficiency [3, 8]. The weight is represented by the resistance of ReRAM and the input is converted to the voltage. Then the accumulated current represents the result of matrix-vector multiplication. Owing to the weights can be either positive and negative, each weight need to be represented by the subtraction of two crossbars (i.e. $R^{+}_{i,j}$ and $R^{-}_{i,j}$). In our work, the mapping of binary weight $\mathcal{W}^{\text{b}}_{i,j}$ upon crossbar can be formulated as follows:

$$R^{+}_{i,j}, R^{-}_{i,j} = \begin{cases} R_{\text{min}}, R_{\text{max}} & \text{if } \mathcal{W}^{\text{b}}_{i,j} = 1 \\ R_{\text{max}}, R_{\text{min}} & \text{if } \mathcal{W}^{\text{b}}_{i,j} = -1 \end{cases} \tag{2}$$

ReRAM cells mainly suffer two defects: resistance variations and SAFs [5, 10]. Resistance variation is a deviation of the actual resistance R' and the target value R_0. In general, the actual resistance of an ReRAM cell follows a log-normal distribution according to [10]:

$$R' = R_0 \cdot e^{\theta} \quad \theta \sim N(0, \sigma^2) \tag{3}$$

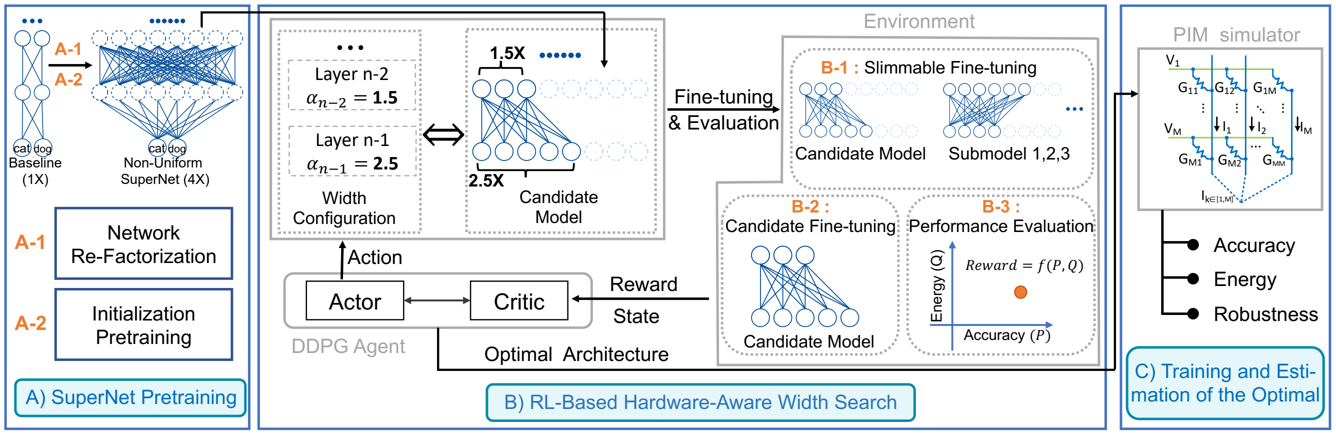
where θ follows a normal distribution with zero mean and a standard deviation of σ. Alternatively, there are two kinds of SAFs: Stuck-At-Zero (SA0) and Stuck-At-One (SA1) [5]. A ReRAM cell with SA0 defect caused by over-forming defects is always in a low resistance state. The SA1 defect is normally caused by open-switch defects, making ReRAM cells stuck in a high resistance state. The ReRAM fabrication shows 1.75% and 9.04% defects rate for SA0 and SA1, respectively [5], which is taken as the region of interest in this work.

III. Approach

We present an overview of the proposed HAWIS framework in Fig. 2. The objective is to obtain a BNN with comparable accuracy and minimized energy w.r.t the FP baseline. The optimization process includes three stages. Stage-A creates and pretrains a super-net. Stage-B identifies the optimal width configuration through RL-based search. Stage-C trains the optimal architecture and estimates its accuracy, energy consumption and robustness.

A. SuperNet Pretraining

Given a FP baseline $\mathcal{A}_{\text{fp}}$ to prepare for high-performance BNN generation, we first *re-factorize* it to obtain a binarized super-net $\mathcal{B}_{\text{b}}$. Then we utilize the slimmable training technique introduced in section II-C to *initially pretrain* $\mathcal{B}_{\text{b}}$.



A-1
A-2
Baseline (1X)
Non-Uniform SuperNet (4X)
A-1 Network Re-Factorization
A-2 Initialization Pretraining
A) SuperNet Pretraining
Layer n-2 α_{n-2} = 1.5
Layer n-1 α_{n-1} = 2.5
Width Configuration
1.5X
2.5X
Candidate Model
Action
Actor
Critic
Reward
State
DDPG Agent
Optimal Architecture
Fine-tuning & Evaluation
Environment
B-1 : Slimmable Fine-tuning
Candidate Model
Submodel 1,2,3
B-2 : Candidate Fine-tuning
B-3 : Performance Evaluation
Reward = f(P, Q)
Energy (Q)
Accuracy (P)
B) RL-Based Hardware-Aware Width Search
PIM simulator
Accuracy
Energy
Robustness
C) Training and Estimation of the Optimal

Fig. 2. An overview of the proposed HAWIS framework. A) *Network re-factorization* and *initialization pretraining* pretrain a binarized super-net. B) We leverage reinforcement learning to determine the width in a layer-by-layer manner. The agent takes the state as input and outputs an action, which makes up the width configuration of the candidate model. *Slimmable fine-tuning* first updates the super-net (B-1). Then the candidate model is initialized according to the super-net and fine-tuned for few epochs (B-2). *Performance evaluation* estimates the accuracy and energy consumption (B-3). The reward is returned to update the agent. C) We train the optimal architecture and estimate the accuracy, energy consumption, and robustness on a crossbar-based PIM simulator.

1) Network Re-factorization: We perform the following three measures to obtain a binarized super-net $\mathcal{B}_{\mathrm{b}}$.

Binarization Function Insertion. We binarize all Conv and FC layers in $\mathcal{A}_{\mathrm{fp}}$. We dismiss the non-linear activation layer (i.e., ReLU) as binarization introduces non-linearity. We remove the max-pooling layer and set the stride of the corresponding Conv layer as two. The BN and activation binarization can be fused and implemented by the comparator. In this way, no extra FP processing units are needed. A fully binarized network is generated, and we name it $\mathcal{A}_{\mathrm{b}}$.

Topology Modification and Two-Side Regularization. Binarizing the input and output layer results in low accuracy even with thermometer coding [22], as indicated by 45.43% accuracy in Table I. We find two observations and propose the following countermeasures to compensate for such accuracy degradation. 1) Valuable information is filtered by the avg-pooling before the FC classifier in BNN, leading to low-dimension and low-discrimination features. Thus we remove the avg-pooling layer, directly flatten the activations and feed them to the classifier. This *topology modification* increases the accuracy to 50.24%, with a negligible increase in the number of parameters. 2) L2 regularization (known as weight decay) limits the value of weights at small amplitude, resulting in more close-to-zero weights. These close-to-zero weights are extremely sensitive in BNN because one update may change the sign and thus generate opposite outputs after binarization. We propose a *two-side regularization* as follows to solve this problem while keeping superior generalization.

$$\Omega(w) = \|(|w| - w_0)\|_2 = \sum_i (|w_i| - w_0)^2 \tag{4}$$

which encourages the weights clustering around w_0 and $-w_0$ instead of 0. The accuracy increases to 51.92%.

Uniform Layer Width Expansion. We uniformly expand the binarized baseline $\mathcal{A}_{\mathrm{b}}$ to create the super-net $\mathcal{B}_{\mathrm{b}}$. We observe that uniformly widened BNNs basically get overfitted when the expansion ratio reaches 5 based on the results in Table II. Thus we set the upper bound of the ratio as 5.

TABLE I
RESULTS OF TOPOLOGY MODIFICATION (TM) AND TWO-SIDE REGULARIZATION(TS).

Model	Method	Acc. (%)	(Δ %)
Binarized ResNet-18 On ImageNet	Base	45.43	0.0
	TM	50.24	+ 4.81
	TM + TS	51.92	+ 6.49

2) Initialization Pretraining: We leverage the slimmable training technique to pretrain the super-net $\mathcal{B}_{\mathrm{b}}$, which encourages the learned weights to own descending importance along the output channel index. Important weights are shared by the sub-networks so that $\mathcal{B}_{\mathrm{b}}$ can provide nice initialization for candidate models in stage-B. Thus, we avoid training candidates from scratch to obtain precise accuracy evaluation, resulting in significant search costs reduction.

B. RL-Based Hardware-Aware Automated Width Search

We leverage Deep Deterministic Policy Gradient (DDPG) based reinforcement learning algorithm to assign layer-wise width expansion ratio (r_l for the l-th layer) optimizing both the accuracy and energy consumption. There are $(L-1)$ steps in an episode for an L-layer model as the output channel number of the last layer is fixed. After getting the super-net $\mathcal{B}_{\mathrm{b}}$ from stage-A, the optimization flow of stage-B in an episode can be described as follows: **1)** The DDPG-agent makes actions to determine the width configuration$\{r_l\}_{l=1}^{L-1}$ in a layer-wise manner, based on the state and reward acquired from the environment; **2)** A candidate model w.r.t the current width configuration is generated from $\mathcal{B}_{\mathrm{b}}$. **3)** Based on the candidate and sampled sub-networks, *slimmable fine-tuning* continues to update $\mathcal{B}_{\mathrm{b}}$ during the search. **4)** *Candidate fine-tuning* further fine-tunes the candidate model; **5)** *Performance evaluation* evaluates the accuracy and energy consumption of the candidate. **6)** The reward is returned for generating the actions in the successive episode.

1) Problem Formulation: The objective of HAWIS is to optimize the width $\{r_l\}_{l=1}^{L-1}$ to achieve a comparable accuracy of the FP counterpart with minimized energy consumption in

PIM system. We denote a sub-net sampled from the super-net as $\mathcal{B}_{\text{b}}(\{r_l\}_{l=1}^{L-1}|\hat{\boldsymbol{\theta}})$ with $\hat{\boldsymbol{\theta}}$ assigned by the corresponding weights in $\mathcal{B}_{\text{b}}$. The reward function is formulated as follows:

$$\mathcal{R} = -\text{Error}\left(\mathcal{B}_{\text{b}}^{*}(\hat{\boldsymbol{\theta}}), \mathbf{X}_{\text{eval}}\right) \cdot \log \frac{Q(\mathcal{B}_{\text{b}}^{*}(\hat{\boldsymbol{\theta}}))/\lambda}{Q(\mathcal{A}_{\text{b}})} \tag{5}$$

where $\mathbf{X}_{\text{eval}}$ is the hold-out evaluation data, $\mathcal{A}_b$ is the binarized baseline and $\mathcal{B}_{\text{b}}^{*}(\hat{\boldsymbol{\theta}})$ is the candidate model. The first term is the distance between the inference accuracy and the given target. The second term is a penalty weighted by λ, where $Q(\cdot)$ represents the energy consumption. We apply a variant form of Bellman's Equation to update the critic-network and sampled policy gradient to update the actor-network, which is not specified here due to space constraints.

2) State Space: In each episode, the DDPG agent sequentially obtains a state vector for each layer defined as follows:

$$\boldsymbol{s}_l = (l, l_{\text{s}}, c_{\text{in}}, c_{\text{out}}, n_{\text{ker}}, n_{\text{str}}, n_{\text{param}}, n_{\text{fmap}}, a_{l-1}, c_{l-1}) \tag{6}$$

where l and l_{s} are the layer index and block index, c_{in} and c_{out} are #input- and #output- channels, n_{ker} and n_{str} denote the kernel size and stride size, n_{param} is #parameter and n_{fmap} is #feature map. All the indicators mentioned above are measured from $\mathcal{A}_{\text{fp}}$. a_{l-1} and c_{l-1} are the action and the expanded channel number of the previous layer, respectively. We normalize each dimension of $\boldsymbol{s}_l$ into $[0, 1]$.

3) Action Space: We use a continuous action space since it keeps the relative order: e.g., 128-channel is wider than 96-channel. The action a_l for l-th layer is firstly converted to the expansion ratio r_l and then discretized into the actual channel number c_l:

$$r_l = a_l(r_{\max} - r_{\min}) + r_{\min} \tag{7}$$

$$c_l = \text{round}(c_{\text{out}} \cdot r_l / d) \cdot d \tag{8}$$

where $r_{\max}$ and $r_{\min}$ denote the upper and lower bound of the expansion ratio. d is the minimal channel width interval. We may adopt varying upper and lower bound for each layer. However, for simplicity, we choose the identical bound across the entire network.

4) Environment: We introduce a three-step process for the rapid and accurate evaluation of the candidate model.

Slimmable Fine-tuning. The candidate $\mathcal{B}_{\text{b}}^{*}(\hat{\boldsymbol{\theta}})$ generated by the actor and another three sub-models $\mathcal{B}_{\text{b}}^{1\sim3}(\hat{\boldsymbol{\theta}})$ are included. $\mathcal{B}_{\text{b}}^{1}(\hat{\boldsymbol{\theta}})$ and $\mathcal{B}_{\text{b}}^{2}(\hat{\boldsymbol{\theta}})$ are randomly sampled from $\mathcal{B}_{\text{b}}$, while the width for different layers in $\mathcal{B}_{\text{b}}^{1}(\hat{\boldsymbol{\theta}})$ is the uniform and in $\mathcal{B}_{\text{b}}^{2}(\hat{\boldsymbol{\theta}})$ is different; $\mathcal{B}_{\text{b}}^{3}(\hat{\boldsymbol{\theta}})$ is morphed from $\mathcal{B}_{\text{b}}^{*}(\hat{\boldsymbol{\theta}})$ via applying randomness on the channel width. Then slimmable training technique updates the super-net $\mathcal{B}_{\text{b}}$ for 1 epoch with the training data $\mathbf{X}_{\text{train}}$. The purpose is to synchronize the shared weights across all channels so that $\mathcal{B}_{\text{b}}$ can provide nice initialization for any sub-model during the search.

Candidate Fine-tuning. We execute customized fine-tuning for the candidate model for few epochs. The purpose is to maximize the accuracy for precise evaluation of the candidate. Note that, the weights updated in this step will be restored after the *performance evaluation* and will not be carried into the next episode.

TABLE II
COMPARISON OF HIGH BIT-WIDTH, UNIFORMLY WIDENED BINARIZED (U-) AND HAWIS NETWORKS. RES-20 DENOTES RESNET-20.

Model	**Res-20 CIFAR10**		**Res-32 CIFAR10**		**Res-18 ImageNet**	
	Energy (μJ)	Acc. (%)	Energy (μJ)	Acc. (%)	Energy (mJ)	Acc. (%)
FP	-	92.1	-	92.8	-	69.6
Quan-8bit	1387	92.2	2349	92.9	66.5	69.8
U-1×	32.7	81.22	50.6	83.91	3.8	51.92
U-2×	120	88.95	195	90.22	8.2	63.38
U-3×	238	91.4	393	92.11	15.0	66.57
U-4×	503	92.17	893	92.49	25.1	68.19
U-5×	924	92.77	1571	93.00	43.5	69.22
U-6×	1176	92.78	1984	93.07	-	-
HAWIS-A	368	92.42	949	92.91	21.3	68.21
HAWIS-B	849	93.13	1045	93.18	29.4	69.29

Performance Evaluation. We estimate the test accuracy on $\mathbf{X}_{\text{eval}}$ and the energy consumption of the candidate model using MNSIM [19]. The reward is calculated according to Eq. (5) and then returned for the update of the agent, which targets high accuracy and low energy consumption.

C. Training and Estimation of the Optimal Model

According to the optimal architecture $\{r_l\}_{l=1}^{L-1}$ generated in stage-B, the optimal binarized model is constructed and trained from scratch. We estimate the final accuracy, energy consumption and robustness under device defects.

IV. EXPERIMENTS

We conduct experiments to prove the effectiveness of HAWIS on multiple backbones on CIFAR-10 [7] and ImageNet [14]. The setting of the ReRAM crossbar system follows [5] except for 1-bit DACs. The upper bound of the width expansion ratio is set as 5. Due to the constraints of GPU memory, we fix the output channel of the first 7×7 Conv layer as 96 and creates another 3×3 Conv layer whose width is adjustable during the search. Stage-A pretrains the super-net for 120 (30) epochs on CIFAR-10 (ImageNet). Stage-B contains 600 episodes, where 20% (10%) training data of CIFAR-10 (ImageNet) are held out for performance evaluation. One epoch and three epochs are included in *slimmable fine-tuning* and *candidate fine-tuning*, respectively. Stage-C uses the full dataset. The learning rate starts from 0.1(0.05), decays by 0.1 in the epochs of {200, 260, 320} ({30, 50, 65}) for CIFAR-10 (ImageNet).

A. Comparison Against High Bit-width and Uniformly Widened Binary Networks

Although Uniformly widening improves the accuracy loss caused by binarization, ResNet-20/32 need to be widened by 5× to reach the accuracy of Quan-8bit models on CIFAR-10. As a comparison in Table III, HAWIS can optimize accuracy and energy consumption simultaneously. We assign a larger λ (defined in Eq. (5)) for HAWIS-A models and a smaller one for HAWIS-B. HAWIS-A models consume much less energy to reach the accuracy of FP baseline. Assigned a smaller penalty on energy, HAWIS-B exceeds the accuracy of U-6× models on ResNet-20/32 with lower energy consumption. On ImageNet, HAWIS also achieves better overall performance,

TABLE III
PERFORMANCE AND COMPLEXITY COMPARISON ON CIFAR-10.

Arch	Precision (W/A)	BiOps ($\times 10^6$)	FLOPs ($\times 10^6$)	Search Cost (GPU-days)	Top-1 (%)
ResNet-20 [4]	8/8	0	41	-	92.2
Bi-Real-18 [9]	1/1	561	11	-	91.2
BARS [23]	1/1	1048	2	-	92.98
BNAS [6]	1/1	670	3	0.42	92.7
BATS [2]	1/1	410	30	0.25	93.7
HAWIS	**1/1**	**1100**	**0**	**1.25**	**93.13**

TABLE IV
PERFORMANCE AND COMPLEXITY COMPARISON ON IMAGENET.

Arch	Precision (W/A)	BiOps ($\times 10^9$)	FLOPs ($\times 10^8$)	Search Cost (GPU-days)	Top-1 (%)
Resnet-18 [4]	8/8	0	18.2	-	69.8
Bi-Real-18 [9]	1/1	1.68	1.38	-	56.4
Bi-Real-34 [9]	1/1	3.53	1.39	-	62.2
MeliusNet-42 [1]	1/1	9.69	1.74	-	69.2
FracBNN [22]	1/1.4	7.30.	0.01	-	71.8
BARS [23]	1/1	2.59	2.54	-	60.3
BNAS [6]	1/1	15.30	4.10	0.42	63.5
BATS [2]	1/1	2.16	1.21	0.25	66.1
Res18-Auto [16]	1/1	19.40	3.55	60	69.7
HAWIS	**1/1**	**37.8**	**0**	**16**	**69.3**

which consumes less energy to reach similar accuracy of uniformly widen BNNs. Although the best accuracy is about 0.5% lower than Quan-8bit model, HAWIS reduces the energy by $\sim 2.3\times$ (29.4 vs. 66.5 mJ).

B. Comparison Against State-of-the-Art Efficient Models

We compare the performance and complexity (i.e., FLOPs and BiOps, which are the number of FP and binary operations, respectively) of our method with other state-of-the-art efficient models. The results are shown in Table III and Table IV. HAWIS on CIFAR-10, with fully binarized layers, outperforms the 8-bit ResNet-20 baseline by about 1% and achieves comparable accuracy compared with Binary NAS methods (i.e. BNAS and BARS). BATS [2] achieves slightly better accuracy but it contains almost the same number of FP operations as the 8-bit baseline.

On ImageNet, although the accuracy of HAWIS is 0.5% lower than the 8-bit ResNet-18 baseline , it outperforms most manually designed BNNs and Binary NAS methods which still own a large part of FLOPs. FracBNN [22] obtains 71.8% Top-1 accuracy, but it possesses a FP output layer and fractional convolutions, where 1-bit or 2-bit weights are used dynamically. Res18-Auto[16] obtains 0.4% higher accuracy than ours, but it contains enormous FP operations (FLOPs=3.55) and requires a longer search time.

C. Robustness Under Device Defects

Fig. 3 compares the robustness of HAWIS, U-1$\times$ BNN and Quan-8bit counterparts. Quan-8bit networks are susceptible to SA0 defects, where 1% and 0.2% SA0 rates lead to a system malfunction on CIFAR-10 and ImageNet, respectively. In contrast, binarized models keep stable under SA0 defects. U-1$\times$ BNN is more robust than Quan-8bit models under SA1 and resistance variation while HAWIS further

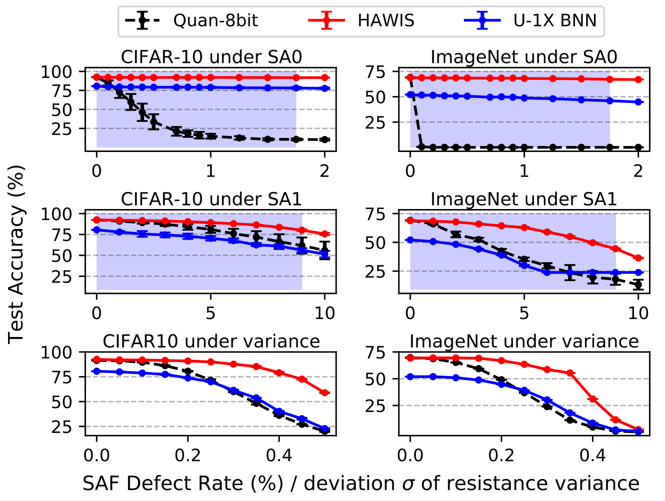


Fig. 3. Test Accuracy under SA0/SA1/resistance variance. (Left) ResNet-20 on CIFAR-10; (Right) ResNet-18 on ImageNet. Error-bar denotes mean±std with multiple trials. Blue shadow shows regions of interest in SAFs.

improves the robustness of the binary baseline. In Quan-8bit models, faults on high-amplitude bits cause more significant calculation deviation, making them suffer more from device defects, especially SA0. Our fully binarized networks have only two levels (1 and -1) for weights. Thus influence of different bits becomes average, resulting in higher robustness. HAWIS models possess finer structures and more parameters than U-1$\times$ BNNs, which further improves their resistance to device faults. The binarized HAWIS models can keep high performance and be deployed in real-world applications with the advancement of ReRAM technology, while 8-bit quantized networks easily come into a malfunction.

D. Ablation Study

Ablation study proves the necessity of *slimmable fine-tuning* and *candidate fine-tuning* to produce the best architecture jointly. Disabling candidate fine-tuning prevents the candidates from convergence to the highest accuracy and leads to low reward (-0.338 and -0.359 in Table V). The under-estimated performance induces the RL agent to choose smaller architectures as they consume less energy. Consequently, the final model chosen by the RL agent is small but low-accuracy. If slimmable fine-tuning is disabled, the impact of a good action cannot be directly passed to the next episode. The weights in the super-net are fixed all the time during the search and may bring great bias to the exploration direction. Correspondingly, the agent converges faster and lacks exploitation. As shown in Fig. 4, the blue and purple curves have fewer rounds and a narrower breadth of exploration than the yellow one, which makes it easier to fall into a local optimal solution.

TABLE V
EFFECTS OF SLIMMABLE FINE-TUNING AND CANDIDATE FINE-TUNING.

	Reward	Energy (μJ)	Acc. (%)
Slimmable + Candidate	-0.019	333	92.15
Slimmable Fine-tuning	-0.338	283	91.71
Candidate Fine-tuning	-0.023	357	91.88
No Fine-tuning	-0.359	207	91.48

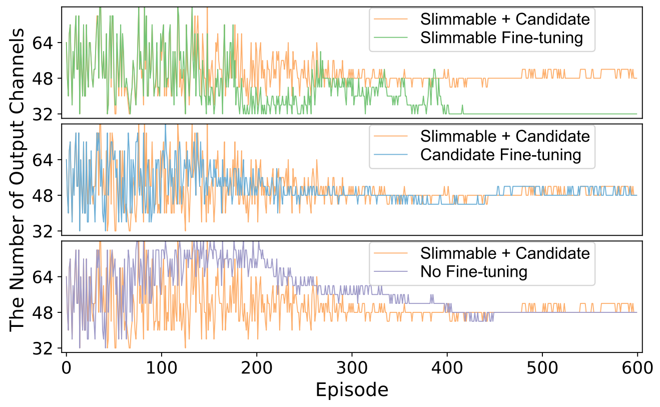


Fig. 4. #(Output Channels) in the 2-th layer of ResNet-20 in each Episode.

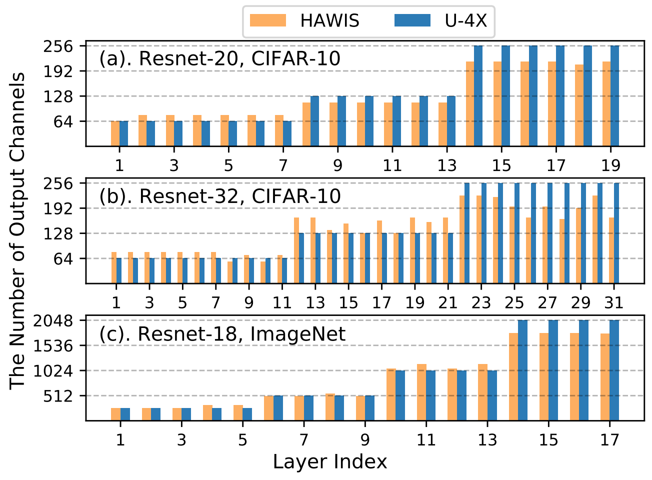


Fig. 5. Comparison of the number of output channels in each layer between 4×-Uniformly widen BNN (blue) and our HAWIS model (yellow).

E. Analysis of the Searched Architecture

To further analyze the optimal architectures generated by HAWIS, Fig. 5 visualizes the number of output channels in each layer. We have three observations. 1) HAWIS architectures commonly possess more channels in the front layers and fewer channels in the tail layers compared with U-4X model. This reduces the information loss at the beginning and transmits more useful features to the following network. 2) HAWIS has a bottleneck-like structure in ResNet-32. Layers with the same width in baseline make up a stage and two successive Conv layers consists of a residual block. The RL agent chooses a narrow width for the former layer within a block at the end of a stage (i.e. the 8/18/28-th layer is narrower than the 9/19/29-th). 3) The selected channel numbers are energy-efficient. The agent assigns 112 channels for the 8-th to 13-th layers in ResNet-20, which happens to completely occupy 4 crossbars (full utilization). These observations prove the significant necessity of carefully determining the optimal backbone-specific and device-specific width configuration to maximize the performance of BNNs.

V. Conclusion

In this work, we introduce a RL-based hardware-aware framework to search for the optimal width configuration in BNNs. We leverage slimmable fine-tuning and candidate fine-tuning for rapid and precise performance evaluation via weight sharing over the huge search space. Combined with network re-factorization, we are the first to obtain high-performance fully binarized networks. Extensive experiments show that HAWIS architectures have high accuracy, low energy consumption and superior robustness against device defects, which shows great potential for future large-scale applications on ReRAM-based PIM accelerators.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grant No. 61834006, 62102257), National Key Research and Development Program of China (2018YFB1403400). Li Jiang and ZheZhi He are the corresponding authors.

References

[1] Joseph Bethge, Christian Bartz, Haojin Yang, Ying Chen, and Christoph Meinel. Meliusnet: Can binary neural networks achieve mobilenet-level accuracy? *arXiv preprint arXiv:2001.05936*, 2020.
[2] Adrian Bulat, Brais Martinez, and Georgios Tzimiropoulos. Bats: Binary architecture search. *ECCV*, 2020.
[3] Ping Chi et al. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. *ISCA*, 2016.
[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
[5] Zhezhi He, Jie Lin, Rickard Ewetz, Jiann-Shiun Yuan, and Deliang Fan. Noise injection adaption: End-to-end reram crossbar non-ideal effect adaption for neural network mapping. In *DAC*, 2019.
[6] Dahyun Kim, Kunal Pratap Singh, and Jonghyun Choi. Learning architectures for binary networks. *ECCV*, 2020.
[7] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
[8] Fangxin Liu, Wenbo Zhao, Zongwu Wang, Tao Yang, and Li Jiang. Im3a: Boosting deep neural network efficiency via in-memory addressing-assisted acceleration. In *GLSVLSI*, 2021.
[9] Zechun Liu et al. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *ECCV*, 2018.
[10] Chang Ma et al. Go unary: A novel synapse coding and mapping scheme for reliable reram-based neuromorphic computing. In *DATE*, 2020.
[11] Asit Mishra, Eriko Nurvitadhi, Jeffrey J Cook, and Debbie Marr. Wrpn: Wide reduced-precision networks. *ICLR*, 2018.
[12] Haotong Qin et al. Forward and backward information retention for accurate binary neural networks. In *CVPR*, 2020.
[13] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Bit-flip attack: Crushing neural network with progressive bit search. In *ICCV*, 2019.
[14] Olga Russakovsky et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.
[15] Ali Shafiee et al. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ISCA*, 2016.
[16] Mingzhu Shen, Kai Han, Chunjing Xu, and Yunhe Wang. Searching for accurate binary neural architectures. In *ICCV Workshops*, 2019.
[17] Linghao Song, You Wu, Xuehai Qian, Hai Li, and Yiran Chen. Rebnn: in-situ acceleration of binarized neural networks in reram using complementary resistive cell. *THPC*, 2019.
[18] Tianqi Tang, Lixue Xia, Boxun Li, Yu Wang, and Huazhong Yang. Binary convolutional neural network on rram. In *ASP-DAC*, 2017.
[19] Lixue Xia et al. Mnsim: Simulation platform for memristor-based neuromorphic computing system. *IEEE T COMPUT AID D*, 2017.
[20] Li Yang, Zhezhi He, and Deliang Fan. A fully onchip binarized convolutional neural network fpga impelmentation with accurate inference. In *ISLPED*, 2018.
[21] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *ICCV*, 2019.
[22] Yichi Zhang et al. Fracbnn: Accurate and fpga-efficient binary neural networks with fractional activations. In *FPGA*, 2021.
[23] Tianchen Zhao et al. Bars: Joint search of cell topology and layout for accurate and efficient binary architectures. *arXiv preprint arXiv:2011.10804*, 2020.
[24] Shuchang Zhou et al. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.