Gibbon: Efficient Co-Exploration of NN Model and Processing-In-Memory Architecture

Hanbo Sun*, Chenyu Wang*, Zhenhua Zhu, Xuefei Ning†
Guohao Dai, Huazhong Yang, Yu Wang†
Department of Electronic Engineering, BNRist, Tsinghua University, Beijing, China

***Abstract*—The memristor-based Processing-In-Memory (PIM) architectures have shown great potential to boost the computing energy efficiency of Neural Networks (NNs). Existing work concentrates on hardware architecture design and algorithm-hardware co-optimization, but neglects the non-negligible impact of the correlation between NN models and PIM architectures. To ensure high accuracy and energy efficiency, it is important to co-design the NN model and PIM architecture. However, on the one hand, the co-exploration space of NN model and PIM architecture is extremely tremendous, making searching for the optimal results difficult. On the other hand, during the co-exploration process, PIM simulators pose a heavy computational burden and runtime overhead for evaluation. To address these problems, in this paper, we propose an efficient co-exploration framework for the NN model and PIM architecture, named *Gibbon*. In *Gibbon*, we propose an evolutionary search algorithm with adaptive parameter priority, which focuses on subspace of high priority parameters and alleviates the problem of vast co-design space. Besides, we design a Recurrent Neural Network (RNN) based predictor for accuracy and hardware performances. It substitutes for a large part of the PIM simulator workload and reduces the long simulation time. Experimental results show that the proposed co-exploration framework can find better NN models and PIM architectures than existing studies in only seven GPU hours (8.4∼41.3× speedup). At the same time, *Gibbon* can improve the accuracy of co-design results by 10.7% and reduce the energy-delay-product by 6.48× compared with existing work.**

I. Introduction

Deep Neural Networks (DNNs) have made breakthroughs in many fields [1], [2]. However, the explosive parameters and computations cause high energy consumption and long latency, hindering the deployment and application of DNNs.

The memristor-based Processing-In-Memory (PIM) architectures have shown powerful capabilities in NN computing. PIM architectures can perform *in-situ* Matrix-Vector-Multiplications (MVMs) and reduce the weight data movements, improving the computing energy efficiency [3]–[6].

Existing PIM work mainly focuses on hardware architecture design [3], [4] and algorithm-hardware co-optimization (e.g., pruning and quantization) [5], [6] for given NN models. But these researches neglect the non-negligible impact of the correlation between NN structure parameters (e.g., kernel size) and PIM architecture design parameters (e.g., crossbar size) on accuracy and hardware performance. For different network structures, the corresponding optimal PIM architectures are different and vice versa. For example, as shown in Figure 1, when the network structure is fixed, different PIM architecture

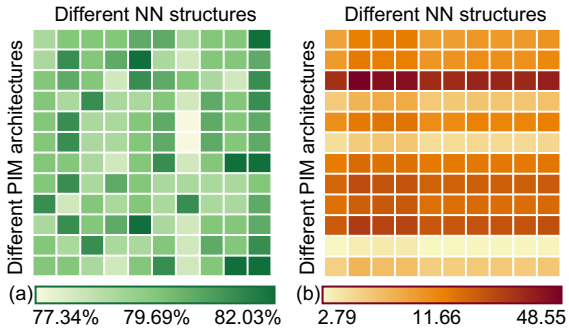

Fig. 1. (a) Accuracy and (b) energy consumption (mJ) of different network structures and PIM architectures on CIFAR-10 dataset [7].

design parameters can cause 2.73% accuracy gap and 91.95% energy consumption gap. When the PIM architecture is fixed, different network structures show 3.45% accuracy gap and 28.31% energy consumption gap. Therefore, NN model and PIM architecture co-design is vital to ensure high accuracy and energy efficiency for PIM-based NN accelerators. However, manually searching for the optimal design of NN model and PIM architecture is unrealistic due to the huge search space. On CIFAR-10 [7], the typical search space size (i.e., the number of possible designs) of NN model[1] and PIM architecture[2] can reach to 10^{58} and 10^{32}, respectively.

Neural Architecture Search (NAS) is an effective way to automatically search well-performing NN models [8]. Researchers have proposed PIM-oriented NAS methods to automatically explore the NN model and PIM architecture co-design space [9], [10]. NACIM [9] and NAS4RRAM [10] incorporate PIM architecture parameters into the search space, and use PIM simulators for performance evaluation. However, these studies suffer from low exploration efficiency and long search time because of the explosive search space expansion and the time-consuming simulation of PIM architectures.

On the one hand, the introduction of PIM architecture parameters into the search space leads to a dramatic expansion of the search space. In the former example, since the structure parameters of NN models and PIM architecture design parameters are independent of each other, the search space increases from 10^{58} to 10^{90} after introducing the PIM architecture-related parameters. As a result, it is more difficult for existing search strategies to explore this largely expanded search space.

*: Both authors contributed equally to this work.
†: Corresponding to foxdoraame@gmail.com, yu-wang@tsinghua.edu.cn

[1]We consider a 30-layer NN model and each layer can be configured with adjustable kernel sizes, group numbers, channel numbers, etc.

[2]The PIM architecture parameters contain crossbar size, DAC/ADC resolution, memristor precision, etc. The per-layer bitwidths of the quantized weights and activations are also considered as architecture-related parameters [5].

On the other hand, the co-exploration of NN model and PIM architecture needs PIM simulators in the loop to evaluate performances, which brings heavy runtime overhead for NAS. Existing PIM simulators (e.g., NeuroSim [11], MNSIM [12]) take about 10 minutes to evaluate the performances of a single NN model. This poses a heavy computational burden to the NAS process, where thousands of search results need to be evaluated. For example, NACIM [9] spends about 60 GPU hours on searching an optimal 8-layer NN on CIFAR-10.

In order to solve these problems, this paper proposes an efficient co-exploration framework for NN model and PIM architecture, which can reduce the search time from hundreds of GPU hours to several GPU hours and generate better search results with higher accuracy and hardware performance. The main contributions of this paper include:

- We propose the evolutionary search algorithm with adaptive parameter priority (ESAPP) to improve the search efficiency in the design space. ESAPP adjusts the search priority of different design parameters according to their convergences during the search. In this way, each search iteration focuses on the sub-search space of high priority parameters, alleviating the problem of huge search space and improving the search efficiency.
- We propose to train a Recurrent Neural Network (RNN) based performance predictor during the search. We use its efficient predictions to substitute for a large portion of costly-simulator-based evaluations. The predictor is designed to predict the accuracy loss induced by the PIM architecture, and this design of "prediction for difference" enables our predictor to be trained with a small amount of actual simulation results. Consequently, the proposed predictor can achieve **150**$\times$ speedup with only **2%** error compared with existing PIM simulators, and can substitute for about 95% of the simulation workload.
- Based on ESAPP and predictor, we propose *Gibbon*, a co-exploration framework for NN model and PIM architecture. Compared with existing PIM-oriented NAS work, *Gibbon* supports more comprehensive algorithm features (e.g., even-sized kernels, group convolution, mixed-precision NN, etc.) and achieves **8.4**$\sim$**41.3**$\times$ search speedup with 10.7% NN accuracy improvement and 6.48$\times$ energy-delay-product (EDP) reduction.
- Based on the results of *Gibbon*, we provide several insights on the correlations of the NN model and PIM architecture, which are helpful to guide PIM-based co-design of model and architecture in the future.

II. Background

A. Processing-in-Memory Architecture

Emerging memristors (e.g., Resistive Random Access Memory, RRAM) provide an alternative solution for realizing PIM architecture. Multiple memristors can construct the crossbar structure. Then the MVMs can be performed in memory by mapping weight matrices onto crossbars and transforming input feature vectors to wordline voltages. Existing work has demonstrated memristor-based PIM can achieve 2$\sim$3 orders of magnitude energy efficiency improvement compared with CMOS-based solutions [3]–[6]. Nevertheless, since PIM performs MVMs in the analog domain, digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) are needed, which brings quantization errors during computations.

B. Neural Architecture Search

NAS aims to leverage machine learning algorithms to automatically design well-performing neural networks with limited computing resources [8], [13]. Generally, a NAS framework consists of three major components: search space, search strategy, and evaluation strategy. The search space defines all the possible NN structures (i.e., candidates) that could be selected. The search strategy decides which NN structure to be sampled during the search. The evaluation strategy evaluates each candidate NN structure and feeds evaluation results back to the search strategy. A typical type of search strategies is evolutionary search [14]–[16]. Evolutionary search strategy conducts three major steps in every search iteration: 1) **Parent selection**: Select parent architectures from the population, which is the set of visited candidates; 2) **Children generation**: Derive child candidates by mutation or crossover; 3) **Population update**: After the evaluation of their performances, update these child candidates into the population.

C. Hardware-Oriented NAS

Recently, NAS researchers have begun to consider hardware performance and co-explore the NN structure and hardware architecture [17]. For PIM-based hardware, NAS4RRAM [10] proposes a PIM-aware NAS framework, searching for deployable networks with the highest accuracy on PIM. Besides searching for NN structures, NACIM [9] also searches for hardware parameters. NACIM uses a time-consuming PIM simulator (i.e., NeuroSim [11]) for the performance evaluation, resulting in a large search cost of about 59 GPU hours. UAE [18] adopts a more sophisticated evaluation strategy, which leads to an even larger search cost (154 GPU hours).

III. Framework Overview

The proposed co-exploration framework for NN model and PIM architecture (*Gibbon*) is shown in Figure 2(a). *Gibbon* consists of three key parts: the joint search space for NN model and PIM architecture co-exploration, the evolutionary search algorithm with adaptive parameter priority (ESAPP), and the RNN-based performance predictor.

The joint search space contains many search candidates, and each candidate specifies the parameters related to both the NN model and the PIM architecture. Each iteration of the search process goes as follows. First, ESAPP samples multiple parents and sends them to the RNN-based predictor. The predictor maps the description of the NN structure and PIM architecture to an embedding vector. Then, an RNN takes the embedding vector as the input and predicts hardware performances and the accuracy difference. The predicted accuracy

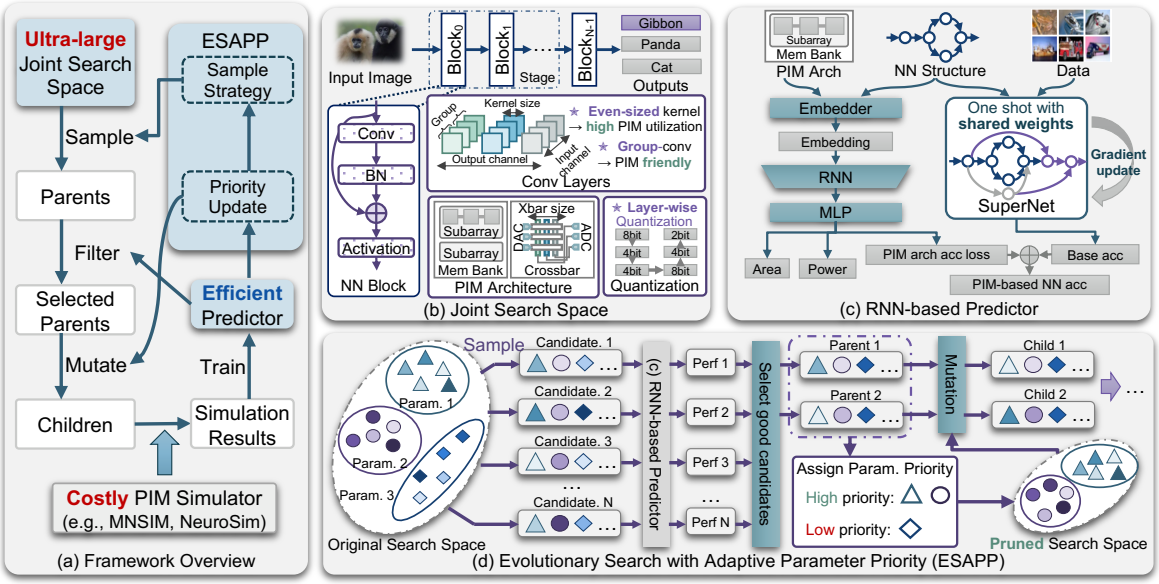

Fig. 2. (a) Framework overview. *Gibbon* contains three key components: (b) joint search space for NN and PIM co-exploration, (c) RNN-based accuracy difference and performance predictor, and (d) evolutionary search algorithm with adaptive parameter priority (ESAPP).

difference depicts the accuracy loss caused by PIM architectures. After getting the prediction results, we use accuracy and hardware performances as the evaluation metric to filter out ∼95% of the sampled parents. Afterward, ESAPP mutates the selected parents to get new candidates (i.e., children), where the mutations are conducted according to the priority of all decision choices. Finally, these candidates are evaluated by the accurate but costly PIM simulator (in this paper, we use MNSIM [12]), and the evaluation results are used to update the RNN-based predictor and the dynamic priorities in ESAPP. *Gibbon* repeats the former steps until the search converges.

IV. Key Techniques in Gibbon

A. Joint Search Space for NN and PIM Co-Exploration

The joint search space (Figure 2(b)) consists of two parts, i.e., the NN model structure part and the PIM-related part.

In the search space of NN structure, each candidate structure consists of three stages separated by down-sampling convolutions. And each stage consists of multiple blocks with various design choices (e.g., output channel number, kernel size, group number of group convolutions, etc.). Also, the block numbers in all stages are adjustable design choices. Compared with existing PIM-oriented NAS methods, we incorporate two PIM-aware features into our search space design. Firstly, we explore the feasibility of even-sized convolution kernels. Commonly used NN models use odd-sized convolution kernels to avoid the feature shifting problem caused by even-sized kernels [19]. However, mapping odd-sized kernels onto the memristor crossbar deteriorates the hardware resource utilization, since the size of memristor crossbar is usually a power of two [20]. To solve this contradiction, *Gibbon* adopts a symmetric padding method [19] to support even-sized convolution kernels without incurring accuracy loss. Secondly, only part of crossbar wordlines can be activated simultaneously due to the current limitation of bitlines, which is consistent with the group-wise calculation behavior of group convolutions. Inspired by this, we introduce group convolutions and take the group number as an adjustable parameter in the search space.

The PIM-related search space is constructed by two parameter sets, i.e., PIM architecture design parameters and quantization parameters. The basic PIM architecture refers to [3], [5], where adjustable design parameters include crossbar size, ADC/DAC resolution, memristor precision, and the number of activated wordlines and bitlines at one time. Because PIM architectures mainly show their superiority for quantized NNs, our PIM-related search space also contains quantization parameters (e.g., quantization bitwidth). In this way, *Gibbon* supports the search for mixed-precision NN, i.e., different layers can have different quantization precision for their weights and activations, which can bring great performance improvements to PIM-based NN systems [5].

B. Evolutionary Search with Adaptive Parameter Priority

The large search space size of PIM-oriented co-exploration poses search efficiency challenges on the application of evolutionary search. For example, the number of candidate designs reaches up to 10^{90} in our joint search space of NN structure parameters and the corresponding PIM-related parameters, resulting in tens to hundreds of GPU hours for search. To tackle this problem, we propose the evolutionary search algorithm with adaptive parameter priority (ESAPP), as shown in Figure 2(d). ESAPP can be regarded as a dynamic search space pruning method to improve the search efficiency, which reduces the average equivalent search space size[3] from 10^{90} to roughly 10^{42} during the search in our experiments.

In the children generation step of each search iteration, ESAPP assigns a priority to each design parameter and determines which parameters to be mutated in this iteration according to the search priority. To be specific, we avoid changing parameters with low priority in this search iteration,

[3]The equivalent search space size is estimated by "omitting" design parameters with low priority from the original search space.

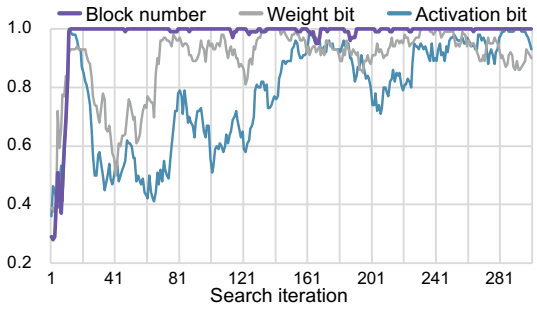

Fig. 3. Convergence of different design parameters. The y-axis indicates the proportion of the optimal candidates among all candidates (the closer to 1.0, the more convergent the parameter is).

realizing equivalent search space pruning. To determine the parameter search priority during the search, we analyze the convergence of different design parameters. Figure 3 reveals that different design parameters show different convergence curves. ESAPP uses the convergence of each parameter to determine its search priority, where lower priority is assigned to converged parameters to reduce redundant search efforts. In this way, ESAPP can explore the large search space in a more organized way, by first evaluating and deciding easy-to-converge parameters, and focusing on difficult-to-converge ones in the later search stage.

Compared with traditional search space pruning methods [21], ESAPP can prevent the search from being stuck into the local optimum, since the parameter priority can be dynamically adjusted throughout the search process. While in traditional search space pruning methods, a parameter cannot become searchable again after being pruned out.

Algorithm 1 shows the details of ESAPP. In each search iteration, ESAPP firstly selects candidates with good performance (e.g., top 100 candidates) in the candidate set as the parents (**line** 2). Then, we use *entropy* to describe the convergence of each parameter, which can be calculated as in Equation 1:

$$Entropy = \sum_{\omega \in \Omega_\omega} -f_\omega \log_2(f_\omega) \tag{1}$$

where Ω_ω is the set of possible values for this design parameter ω, and f_ω represents the occurrence frequency of ω in the selected candidate set. Then, we assign low search priority to parameters with low entropy (**line** 3∼5). After that, we determine the mutation probability according to parameter priorities (**line** 6). The converged parameters with low entropy have low mutation probability, which means these parameters will remain unchanged with a high probability. Finally, the mutation procedure is executed to generate the new candidates for the next search iteration (**line** 7∼12).

C. RNN-based Predictor

The evaluator assesses the NN accuracy and hardware performances of candidate designs. Existing PIM-oriented NAS work [9] uses PIM simulators as the evaluator. However, due to the large-scale PIM architecture and NN model, existing PIM simulators require ∼10 minutes for evaluating a single NN model [11], [12]. Considering that there exist hundreds of search-evaluation iterations in a NAS process, minute-level PIM simulators will bring huge search costs.

Algorithm 1 Pseudo code of ESAPP

Input: Population: $\{Candidate\}_{i-1}$; History entropy table: HET; Evaluation results of candidates: R
Output: New population: $\{Candidate\}_i$
1: Initialize $\{Candidate\}_i$: $\{Candidate\}_i$.init()
2: Select good candidates as parents: $Parents$ = sort($R, \{Candidate\}_{i-1}$)
3: For each design parameter x, calculate the entropy of parents: $Entropy_x$ = calc_e($Parents$)
4: Update HET: $HET = HET$.append($\{Entropy_x\}$)
5: Assign priority to each parameter according to entropy: $Priority_x$ = calc_p(HET)
6: Determine the mutation probability of each parameter: $P_x = Priority_x / \sum_j (Priority_j)$
7: //*Mutate according to* $\{P_x\}$:
8: **for** each $parent$ in $Parents$ **do**
9: $child$ = mutate($parent, \{P_x\}$)
10: $\{Candidate\}_i$.append($parent$)
11: $\{Candidate\}_i$.append($child$)
12: **end for**

1) Predictor Construction: To accelerate the evaluation, we propose an RNN-based NN accuracy and PIM performance predictor. It takes the description of a candidate design as the input and predicts the evaluation results of the simulators, avoiding the time-consuming circuit simulation and complicated PIM computing error analysis. As depicted in Figure 2(c), the RNN-based predictor consists of three key parts: design embedder, feature extractor, and regressor.

The **design embedder** transforms the discrete description of candidate design into one continuous embedding vector. With this transformation, the NN-based predictor can be trained with gradient-based optimization, and can learn the relationship between different design parameters.

The **feature extractor** takes the former embedding vector as inputs and extracts features of the candidate. Each NN model in the search space contains three stages stacked by multiple blocks, and the numbers of blocks are also adjustable. Therefore, we adopt an RNN model as the feature extractor to handle inputs with variable length.

The **regressor** is a three-layer multi-layer perception (MLP), and outputs the final predicted results based on the extracted features (i.e., the average of RNN outputs in all timesteps). During the search, simulation results of PIM simulators are used as the ground-truth values to train the predictor.

2) Ranking Quality of the Predictor: Unlike other CMOS-based NN accelerators, PIM architecture faces more severe computing error caused by non-ideal factors (e.g., device variation) and hardware quantization error (e.g., ADC quantization). Thus it is necessary to take the PIM-related parameters into consideration when predicting the PIM-based NN accuracy. Different from vanilla predictor-based NAS methods [22] that predict the accuracy directly, *Gibbon* proposes to predict the relative accuracy loss brought by PIM architecture of a candidate design, as shown in Figure 2(c). We use the one-

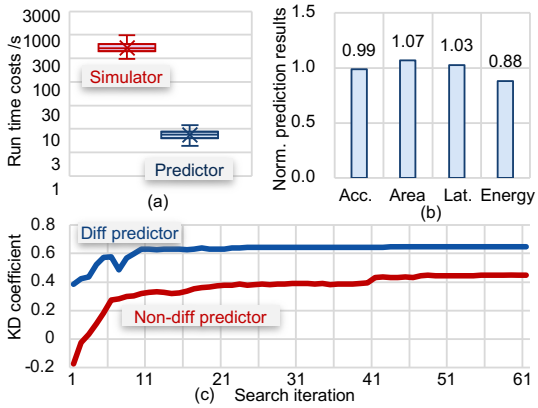

Fig. 4. (a) Time cost comparison of predictor and PIM simulator; (b) Prediction results normalized to PIM simulator (Acc. and Lat. represent PIM-based NN accuracy and computing latency, respectively); (c) KD coefficients of the prediction-for-difference predictor and the vanilla predictor on the same validation set of candidate designs.

shot accuracy [16] as the base accuracy, and the one-shot weights in the supernet are updated jointly in each search iteration. Thanks to the design of "prediction for difference", the predictor only has to model the effects brought by the PIM architecture, which is an easier problem than predicting the absolute accuracy. Therefore, *Gibbon* manages to train a more accurate predictor with a small amount of simulation results as training data.

To demonstrate the effectiveness of our prediction-for-difference design, we compare the Kendall's Tau (KD) ranking correlation of the difference predictor and the vanilla predictor. KD is a commonly adopted criterion for the ranking quality of the predictor or evaluator in NAS studies [22], and its calculation goes as follows:

$$\mathrm{KD} = \sum_{i<j} \mathrm{sgn}(y_i - y_j)\,\mathrm{sgn}(s_i - s_j) / \binom{M}{2} \in [-1, 1] \quad (2)$$

where M denotes the number of candidates, and y_i and s_i represent the ground-truth performance and predicted performance, respectively. $\mathrm{sgn}(\cdot)$ is the sign function. A higher KD indicates that the ranking of predictions is more similar to the ranking of ground-truth performances.

The results in Figure 4(c) show that the prediction-for-difference predictor can give out predictions with better ranking quality using the same amount of training data. As for the prediction error, Figure 4(b) shows that the relative prediction error compared with the simulation results is only 1.2% and 2.6% for NN accuracy and computing latency, respectively.

3) Efficiency of the Predictor: In terms of the evaluation time cost, Figure 4(a) shows that the PIM simulator takes an average of 549s simulation time, while the predictor takes an average of 7.59s for NN accuracy and PIM performance prediction (98.6% reduction). And *Gibbon* uses the predictor to substitute for 95% of the simulation workload.

V. Experiment

A. Experiment Setup

Gibbon is developed based on aw_nas [23], an open source NAS framework. We conduct all experiments on CIFAR-10 [7]. Details of the search space design are summarized in Table I. In this paper, we use the most mature 1-bit and 2-bit memristors, and *Gibbon* also supports the search of other device precision. We adopt MNSIM [12] as the simulator to train the predictor (other PIM simulators can also be used in *Gibbon*). The data of memristor, ADC, and DAC we used refer to the default values provided in MNSIM [12].

TABLE I
NN Model And PIM Related Search Space

NN Model Search Space	
Block Number	2, 4, 6, 8, 10
Output Channel Number	16, 32, 48, 64, 80, 96
Kernel Size	1, 2, 3
Group Number	1, 2, 4, 8, 16
Quantization Search Space	
Weight/Activation Bitwidth	5, 7, 9
Hardware Search Space	
Crossbar Size	32, 64, 128, 256
ADC Resolution	4, 6, 8, 10
DAC Resolution	1, 2
Memristor Precision	1, 2

B. Co-exploration Results Comparison

Table II provides the search results comparison between *Gibbon* and other PIM-oriented NAS work. For a fair comparison, we test the NN model discovered by NACIM [9] with MNSIM to get the hardware performances. UAE [18] and NAS4RRAM [10] only provide NN accuracy without giving hardware performances. We also compare *Gibbon* with the vanilla CARS [16] without ESAPP and the predictor design. We also provide search results of *Gibbon* under three different optimization targets (adjust the weight of each objective in the search reward): EDP optimization, area optimization, and accuracy optimization. The search time is evaluated on a single Nvidia RTX 3090 GPU.

Compared with other PIM-oriented NAS work, *Gibbon* can achieve 0.2∼10.7% accuracy promotion in only seven search hours, realizing 8.4∼41.3× search efficiency improvement. Furthermore, compared with CARS, ESAPP and predictor show 10.3× search speedup with similar search results. In terms of the hardware performance, the EDP optimization *Gibbon* can achieve 10.7% accuracy improvement and 6.48× EDP reduction, and *Gibbon* with area optimization can achieve 2.5% accuracy improvement and 2.51× area reduction.

Figure 5 demonstrates the accuracy and EDP results of *Gibbon* with different optimization targets. The results of *Gibbon* w/o ESAPP and w/ ESAPP are provided under the same search time. The superiority of *Gibbon* w/ ESAPP shows that ESAPP can find better results in the same search time.

TABLE II
Search results (PIM-based NN accuracy, EDP, area, and search time) comparison of different co-exploration approaches.

Method	NN accuracy	EDP ($ms \times mJ$)	Area (mm^2)	Search time (h)
NACIM [9]	73.9%	1.55	17.17	59
UAE [18]	83.0%	–	–	154
NAS4RRAM [10]	84.4%	–	–	289
CARS [16] (acc opt.)	88.0%	11.03	227.73	72
Gibbon (edp opt.)	84.6%	**0.24**	167.16	**7**
Gibbon (area opt.)	76.4%	1.00	**6.84**	**7**
Gibbon (acc opt.)	**88.3%**	14.33	186.32	**7**

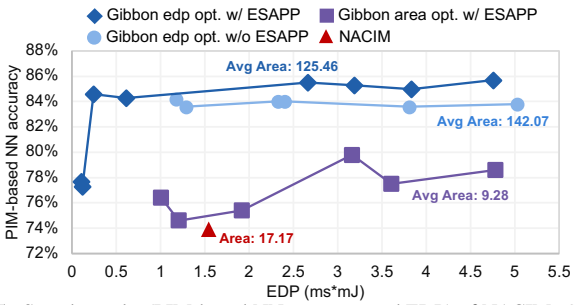

Fig. 5. Search results (PIM-based NN accuracy and EDP) of NACIM, *Gibbon* w/o ESAPP, and *Gibbon* w/ ESAPP. The results of *Gibbon* w/o ESAPP and *Gibbon* w/ ESAPP are given under the same search time.

VI. Insights Provided by Gibbon

By inspecting the NN models and PIM architectures discovered by *Gibbon*, we find some interesting observations as follows, hoping to provide some design suggestions for the co-design of NN model and PIM architecture in the future.

Insight 1: Most convolution layers tend to select even-sized kernels to reduce the area and EDP. When other design parameters are the same, design candidates with 2×2 kernels can decrease ∼85% EDP and ∼10% area compared with those with 3×3 kernels. And the accuracy loss is only ∼1%.

Insight 2: Group convolution can reduce the amount of calculations. When one DAC is multiplexed by four wordlines, most convolution layers prefer to choose two or four as the number of groups, which is less than the number of DAC multiplexes and does not bring additional latency overhead.

Insight 3: For accuracy optimization, the shallower and deeper layers tend to have larger output channel number (e.g., 64), while the middle layers have smaller ones (e.g., 16).

Insight 4: For PIM-based mixed-precision NN on CIFAR-10, considering accuracy and energy consumption, the deeper convolution layers tend to choose high quantization bitwidth of weights while the shallower layers (except the first layer) prefer low weights precisions. For the quantization of activations, the shallower and deeper convolution layers both prefer high bitwidth, while the middle layers tend to choose low bitwidth.

Insight 5: For CIFAR-10, the accuracy of 8-bit ADCs is close to that of 10-bit ADCs. But 6-bit ADCs lead to non-negligible accuracy loss. In terms of crossbar size, the average optimal accuracy can be obtained when the size is 64×64.

Insight 6: The number of output channels of the first few layers has a significant impact on the total latency. Low latency models tend to choose smaller output channel number in shallow layers. Relatively, deep layers prefer large channel number to improve accuracy with little latency overhead.

Insight 7: The energy-optimal PIM design tends to choose large crossbar size (e.g., 256×256), which can reduce the number of kernel splits and average used ADCs/DACs. Compared with crossbars in size of 128×128, PIM architectures with 256×256 crossbars can reduce 75% energy consumption.

VII. Conclusion

In this paper, we propose *Gibbon* to efficiently co-explore NN model and PIM architecture. Compared with existing PIM-oriented NAS work, *Gibbon* leverages ESAPP and RNN-based predictor to improve the search efficiency. Experimental results show that *Gibbon* can achieve 8.4∼41.3× search speedup with 10.7% NN accuracy improvement and 6.48× EDP reduction compared with existing work.

VIII. Acknowledgements

This work was supported by National Key R&D Program of China (No. 2017YFA02077600); National Natural Science Foundation of China (No. 61832007, U19B2019, 61621091); China Postdoctoral Science Foundation (No. 2019M660641); Tsinghua EE Xilinx AI Research Fund; Beijing National Research Center for Information Science and Technology (BNRist); Beijing Innovation Center for Future Chips; Beijing Academy of Artificial Intelligence.

References

[1] A. Krizhevsky, I. Sutskever *et al.*, "Imagenet classification with deep convolutional neural networks," *NeurIPS*, vol. 25, pp. 1097–1105, 2012.
[2] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.
[3] P. Chi, S. Li, C. Xu *et al.*, "Prime: a novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ISCA*, 2016, pp. 27–39.
[4] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *IEEE HPCA*, 2017, pp. 541–552.
[5] Z. Zhu, H. Sun *et al.*, "A configurable multi-precision cnn computing framework based on single bit rram," in *ACM/IEEE DAC*, 2019, pp. 1–6.
[6] T.-H. Yang, H.-Y. Cheng, C.-L. Yang *et al.*, "Sparse reram engine: Joint exploration of activation and weight sparsity in compressed neural networks," in *ISCA*, 2019, pp. 236–249.
[7] Kaggle *et al.*, "Cifar-10 - object recognition in images," website, 2014, https://www.kaggle.com/c/cifar-10.
[8] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.
[9] W. Jiang, Q. Lou, Z. Yan *et al.*, "Device-circuit-architecture co-exploration for computing-in-memory neural accelerators," *IEEE Transactions on Computers*, vol. 70, no. 4, pp. 595–605, 2020.
[10] Z. Yuan, J. Liu, X. Li *et al.*, "Nas4rram: neural network architecture search for inference on rram-based accelerators," *Science China Information Sciences*, vol. 64, no. 6, pp. 1–11, 2021.
[11] X. Peng, S. Huang, H. Jiang *et al.*, "Dnn+neurosim v2. 0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," *IEEE TCAD*, 2020.
[12] Z. Zhu, H. Sun, K. Qiu *et al.*, "Mnsim 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems," in *GLSVLSI*, 2020, pp. 83–88.
[13] P. Ren, Y. Xiao, X. Chang *et al.*, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–34, 2021.
[14] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE TNNLS*, 2021.
[15] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *AAAI*, 2019, pp. 4780–4789.
[16] Z. Yang, Y. Wang, X. Chen *et al.*, "Cars: Continuous evolution for efficient neural architecture search," in *CVPR*, 2020, pp. 1829–1838.
[17] Y. Lin, M. Yang, and S. Han, "Naas: Neural accelerator architecture search," *arXiv preprint arXiv:2105.13258*, 2021.
[18] Z. Yan, D.-C. Juan, X. S. Hu *et al.*, "Uncertainty modeling of emerging device based computing-in-memory neural accelerators with application to neural architecture search," in *IEEE ASP-DAC*, 2021, pp. 859–864.
[19] S. Wu, G. Wang, P. Tang *et al.*, "Convolution with even-sized kernels and symmetric padding," *arXiv preprint arXiv:1903.08385*, 2019.
[20] Z. Zhu, J. Lin *et al.*, "Mixed size crossbar based rram cnn accelerator with overlapped mapping method," in *ICCAD*, 2018, pp. 1–8.
[21] Y. Hu, Y. Liang, Z. Guo *et al.*, "Angle-based search space shrinking for neural architecture search," in *ECCV*, 2020, pp. 119–134.
[22] X. Ning, Y. Zheng, T. Zhao *et al.*, "A generic graph-based neural architecture encoding scheme for predictor-based nas," in *ECCV*, 2020, pp. 189–204.
[23] X. Ning, C. Tang, W. Li *et al.*, "aw_nas: A modularized and extensible nas framework," *arXiv preprint arXiv:2012.10388*, 2020.