# Multi-Objective Neural Architecture Search for In-Memory Computing

Md Hasibul Amin, Mohammadreza Mohammadi, Ramtin Zand
Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208, USA
e-mail: ma77@email.sc.edu, mohammm@email.sc.edu, ramtin@cse.sc.edu

***Abstract*—In this work, we employ neural architecture search (NAS) to enhance the efficiency of deploying diverse machine learning (ML) tasks on in-memory computing (IMC) architectures. Initially, we design three fundamental components inspired by the convolutional layers found in VGG and ResNet models. Subsequently, we utilize Bayesian optimization to construct a convolutional neural network (CNN) model with adaptable depths, employing these components. Through the Bayesian search algorithm, we explore a vast search space comprising over 640 million network configurations to identify the optimal solution, considering various multi-objective cost functions like accuracy/latency and accuracy/energy. Our evaluation of this NAS approach for IMC architecture deployment spans three distinct image classification datasets, demonstrating the effectiveness of our method in achieving a balanced solution characterized by high accuracy and reduced latency and energy consumption.**

***Index Terms*—In-memory computing, neural architecture search, processing-in-memory, memristive crossbar, optimization.**

## I. Introduction

In-memory computing (IMC) architectures, also referred to as processing-in-memory (PIM) or compute-in-memory (CIM), have emerged as promising alternatives to traditional von Neumann-based machine learning (ML) hardware [1], [2]. These architectures capitalize on characteristics such as massive parallelism, analog computation, and executing computations directly where data is stored, resulting in notable performance enhancements [3]–[5]. The backbone of most IMC architectures lies in memristive crossbar arrays, which leverage resistive memory technologies like resistive random-access memory (RRAM) [6] and magnetoresistive random-access memory (MRAM) [7]. These arrays enable matrix-vector multiplication operations in the analog domain, using fundamental circuit principles such as Ohm's Law and Kirchoff's Current Law [8], [9].

Despite the aforementioned advancements, prior research has shown that deploying ML models that are pre-trained and optimized using digital von Neumann architectures, such as CPU and GPU, on an analog IMC architecture does not consistently yield comparable performance [10]. Several factors contribute to this, including limited numerical precision of memristive devices [11], as well as circuit imperfections such as the interconnect parasitic [12], and device-to-device and cycle-to-cycle variations [13]. The *in-circuit training* [14] is introduced as a mechanism to address the deployment challenges by bringing the IMC circuits within the loop of training and allowing the ML models to learn the circuit imperfections. However, this method may face limited applicability due to the endurance constraints of memristive devices [15]. These devices can lose their storage capability after a certain number of repeated write operations.

Another strategy involves utilizing IMC hardware simulators like Neurosim [16], MNSIM [17], and IMAC-Sim [18] to emulate the characteristics of IMC circuits and the mapping strategies for deploying ML models on an IMC chip. This method, which we termed *pre-silicon optimization*, involves adjusting circuit and device level parameters to achieve specific objectives. One advantage of this approach is its holistic exploration of both IMC circuit and neural architecture parameters. However, pre-silicon optimization methods [19]–[22] are primarily intended for chip manufacturers to make early design decisions before chip fabrication.

Our paper focuses on another mechanism, termed *post-silicon optimization*[1], which integrates hardware deployment into the optimization loop but only adjusts neural architecture parameters related to the ML model while keeping hardware parameters fixed. This approach utilizes hardware measurements such as accuracy, latency, and energy to formulate a multi-objective fitness function (FF). The neural architecture search (NAS) algorithm leverages this fitness function to develop models that can perform effectively despite IMC circuit limitations, without modifying the circuit itself. While our paper does not employ physical hardware implementation of IMC circuits, the proposed methodologies can be readily applied to optimize and deploy various ML models on IMC architectures.

## II. Analog IMC Architecture Background

Figure 4 demonstrates the hierarchical structure of the In-Memory Computing (IMC) architecture [17]. The IMC architecture consists of multiple IMC banks, the global buffer, the global accumulator, and the IMC controller as shown in Fig. 4a. The IMC controller receives the status updates from the CPU and controls the data movement between DRAM and the IMC banks. Global buffer and global accumulator are necessary for the elementwise-sum operation to implement the skip-connections in ML models like ResNet [23]. Inside an IMC bank, arrays of tiles are connected using a network-on-chip structure. Inside a tile, there are multiple processing ele-

[1]The term "post-silicon" does not imply the necessity of physical hardware for optimization. Instead, it emphasizes optimizing parameters that can be adjusted after hardware fabrication.

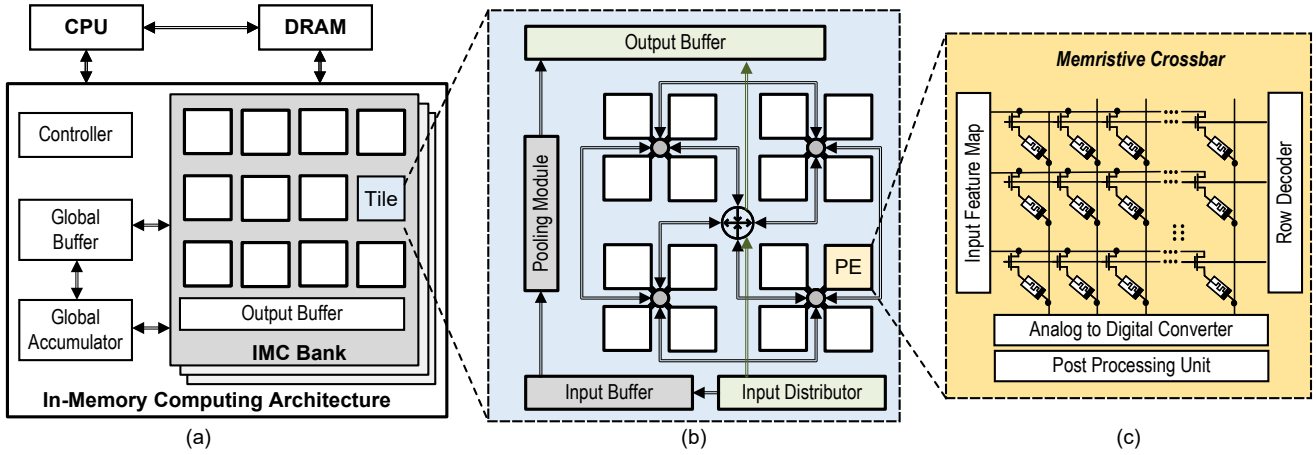



Fig. 1: (a) The analog IMC architecture with multiple IMC banks each of which includes several interconnected IMC tiles [17]. (b) The IMC tile consists of a network of tightly coupled processing elements (PEs). (c) The structure of the IMC processing element that includes memristive crossbars in its core.

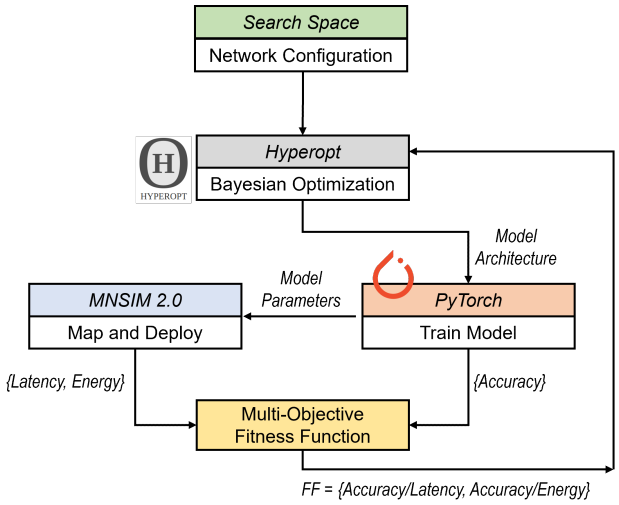



Fig. 2: The proposed NAS methodology for multi-objective optimization of ML workloads deployed on analog IMC architectures.

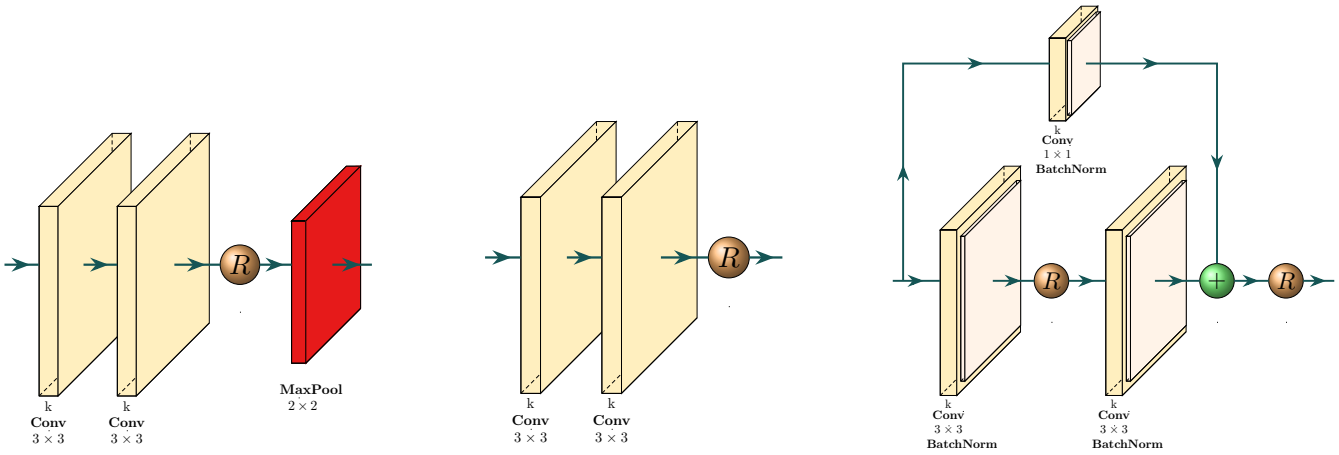
ments (PE) and a pooling module along with input and output buffers as shown in Fig. 4b. The PE blocks contain crossbars of resistive memory devices such as RRAM and the peripheral circuits. Peripheral circuits vary based on analog and digital IMC. Figure 4c shows the internal structure of an analog PE. Analog IMC requires analog-to-digital converter (ADC) and digital-to-analog converter (DAC) blocks as peripheral circuits, while digital IMC requires digital computational units, shift-and-add circuits, and sense amplifiers. Crossbars perform matrix-vector multiplication (MVM) operations in either analog or digital domains. In the analog IMC, which is the architecture utilized in this paper, the weight kernels are expanded into a vector and loaded onto the columns of the crossbars. The input feature map is also expanded into a vector and provided as inputs to the crossbar. The crossbar performs the MVM operation using fundamental circuit principles in parallel and in $O(1)$ time complexity.

## III. The Proposed NAS Methodology

In this paper, we present a novel neural architecture search (NAS) approach for multi-objective optimization of the ML workloads deployed on analog IMC architectures. The high-level description of the proposed methodology is illustrated in Fig. 2. We employ the Hyperopt library [24] to handle the NAS process. Hyperopt [24] is a Python library designed for automated hyperparameter tuning, employing various sequential model-based optimization techniques, also known as Bayesian optimization. The search process is an iterative process that is repeated for a certain number of iterations set by the user, and the best model is selected based on a pre-defined fitness function (FF).

We have integrated Hyperopt with PyTorch and MNSIM 2.0 [17] frameworks to perform the multi-objective optimization. In each iteration, Hyperopt automatically selects a model configuration from the search space and uses PyTorch and MNSIM to obtain the model accuracy as well as the IMC hardware performance metrics such as latency and energy. These metrics are combined to form multi-objective FF such as $^{accuracy}/_{latency}$ or $^{accuracy}/_{energy}$.

Before beginning the optimization, we should first create the search space. Here, we focus on convolutional neural networks (CNNs) as representative ML workloads. We design three distinct building blocks: (1) the VGG block with sub-sampling, (2) the modified VGG block (MVGG) without sub-sampling, and (3) the ResNet block (RES). As shown in Fig. 3a, the VGG block with sub-sampling consists of two consecutive convolutional layers with a kernel size of $3 \times 3$. The number of kernels ($k$) is determined during the optimization phase. Following these layers, a ReLU activation function and a $2 \times 2$ max-pooling layer are applied. In the MVGG block depicted in




(a) VGG block with MaxPooling (b) MVGG block without MaxPooling (c) ResNet block (RES)

Fig. 3: The three building blocks of the CNN architecture.

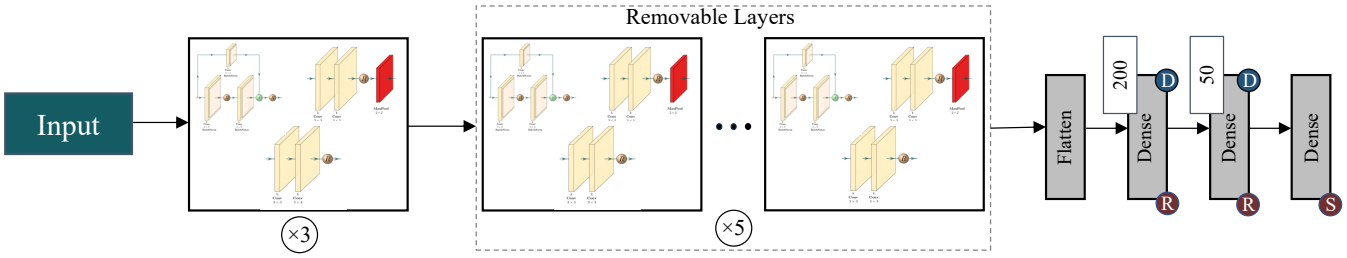



Fig. 4: The network configuration. R ReLU, D Dropout, S Softmax.

Fig. 3b, we remove the sub-sampling block. This decision is made to leverage the inherent properties of the VGG block without downsizing the feature map dimensions within the network. The RES block, depicted in Fig. 3c, consists of two consecutive convolutional layers, each configured with $k$ kernels and a $3 \times 3$ kernel size. Following each convolutional layer, a batch normalization layer is applied, with a ReLU activation function positioned between them. On the residual connection side, there is a convolutional layer with $k$ kernels, each having a size of $1 \times 1$. Similar to VGG blocks, the number of kernels is determined during the optimization phase. The outputs from both the convolutional layers and the residual connection are combined, and then a ReLU function is applied.

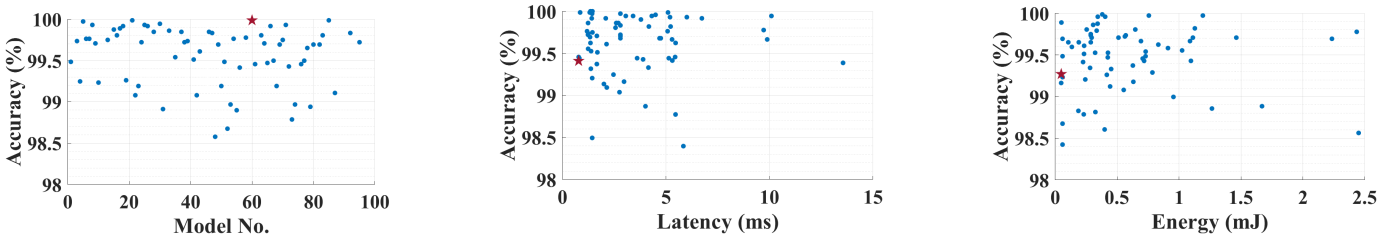
The overall structure of the CNN models created in this paper is illustrated in Fig. 4, incorporating the aforementioned building blocks. We include several adjustable parameters to optimize the networks. As outlined in Table I, the parameters available for tuning are (1) the number of blocks ($Block$) can vary between 3 and 8, allowing for different network depths; (2) the type of each block ($BT$) can be any of the three building blocks, including VGG with sub-sampling ($VGG$), modified VGG without sub-sampling ($MVGG$), and ResNet block ($RES$); (3) the number of kernels ($K$) for each block ranging from 16 to 256. After the *flatten* layer, the number of layers and neurons in the fully connected layers remain fixed. These configurable parameters collectively define a search space comprising over 641 million unique network configurations.

TABLE I: Network configuration settings

| Parameters | Description | Options |
|---|---|---|
| $Block$ | No. of blocks in the network | 3-8 |
| $BT$ | The block type | Res, VGG, MVGG |
| $K$ | No. of kernels in each conv layer | 16, 32, 64, 128, 256 |

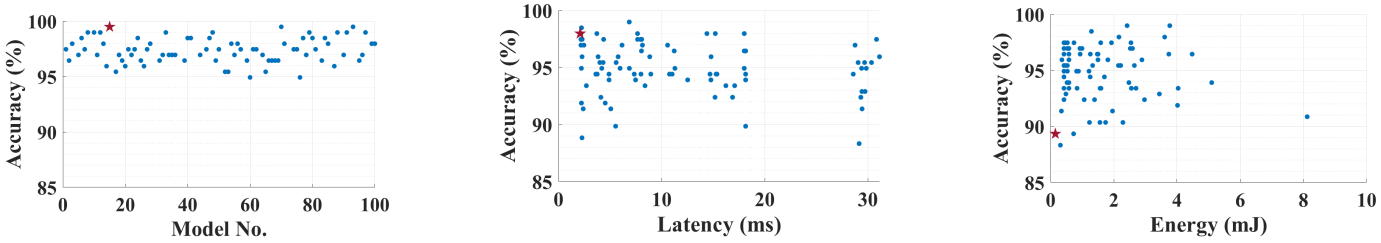
## IV. Results

To validate the effectiveness of our proposed method, we conduct comprehensive experiments utilizing three distinct datasets: (1) the American Sign Language (ASL) Alphabet Dataset [25], featuring static images for hand gesture classification across 24 letters (excluding motion-dependent J and Z); (2) the Extended Cohn-Kanade (CK+) Dataset [26], containing 593 video sequences to evaluate facial expression recognition as subjects transition between neutral and seven distinct emotions; and (3) CIFAR-10 [27], a well-known dataset in the computer vision field for assessing object recognition capabilities with 60,000 color images across 10 classes. To attain the multi-objective optimization, we run our method with three different fitness functions- $accuracy$, $^{accuracy}/_{latency}$, and $^{accuracy}/_{energy}$, which ensures a holistic view of the



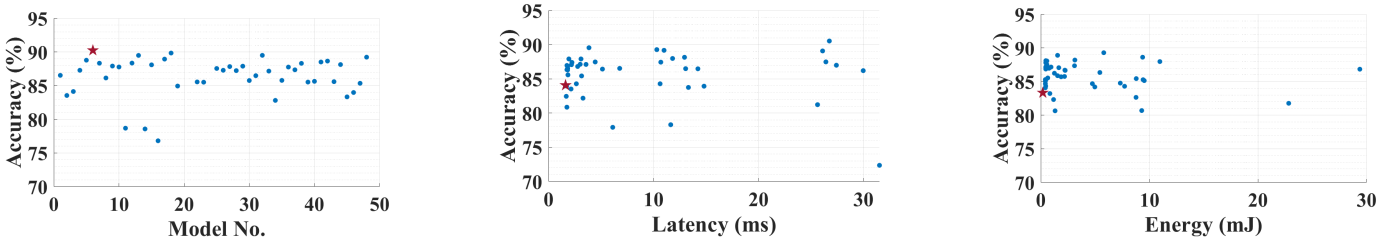

(a) (b) (c)

Fig. 5: Distribution of NAS outputs for ASL dataset (The best model is marked by $\star$) (a) distribution of accuracy when $FF = accuracy$ (b) distribution of accuracy vs latency when $FF = {}^{accuracy}/_{latency}$ (c) distribution of accuracy vs energy when $FF = {}^{accuracy}/_{energy}$.




(a) (b) (c)

Fig. 6: Distribution of NAS outputs for CK+ dataset (The best model is marked by $\star$) (a) distribution of accuracy when $FF = accuracy$ (b) distribution of accuracy vs latency when $FF = {}^{accuracy}/_{latency}$ (c) distribution of accuracy vs energy when $FF = {}^{accuracy}/_{energy}$.




(a) (b) (c)

Fig. 7: Distribution of NAS outputs for CIFAR-10 dataset (The best model is marked by $\star$) (a) distribution of accuracy when $FF = accuracy$ (b) distribution of accuracy vs latency when $FF = {}^{accuracy}/_{latency}$ (c) distribution of accuracy vs energy when $FF = {}^{accuracy}/_{energy}$.

method's performance, encompassing both its classification accuracy and operational efficiency across a variety of visual recognition tasks.

Figure 5, 6 and 7 show the distribution of NAS outputs for ASL, CK+ and CIFAR-10 datasets, respectively. For the ASL and CK+ datasets, we train 100 distinct models for the optimization of each of the fitness functions and identify the model with the best values of the fitness functions. As the outputs do not have much improvements after around 50 model evaluations, we train 50 models for the CIFAR-10 dataset to save simulation time. Table II lists the model descriptions that are identified as the best model in terms of different fitness functions. When the $accuracy$ fitness function is used, the proposed method tends to identify deeper models which leads to higher accuracy due to better generalization in a higher number of layers. The identified models also have a higher number of kernels, which provides a higher number of convolutional filters to allow learning more features. As an instance, for the CIFAR-10 dataset, the best model with the $accuracy$ fitness function picks as many as 6 blocks, which shows $90.24\%$ accuracy, with 15 ms latency and 6.7 mJ energy consumption. Two of the blocks have a kernel size of 256, which was set as the highest number of kernels possible for a block in our search space.

With the ${}^{accuracy}/_{latency}$ FF, the proposed method tends to identify shallower models and less number of kernels which leads to less computation cycle and less latency. The energy consumption also reduces at the same time due to less data movement on the IMC chip and the accuracy drops due to fewer features and less generalization of the features. For the

TABLE II: Best model architecture identified by the proposed NAS method for various datasets and fitness functions. Accuracy, Latency, and Energy metrics are denoted by $Acc$, $Lt$, and $En$, respectively.

| FF | Dataset | Best Model Architecture | | | | | | Accuracy | Latency (ms) | Energy (mJ) |
|---|---|---|---|---|---|---|---|---|---|---|
| | | BT1/K1 | BT2/K2 | BT3/K3 | BT4/K4 | BT5/K5 | BT6/K6 | | | |
| $Acc$ | ASL | MVGG/32 | RES/16 | VGG/128 | VGG/128 | RES/128 | RES/64 | 99.98% | 4.8 | 1.01 |
| | CK+ | RES/16 | VGG/16 | VGG/16 | VGG/64 | MVGG/64 | - | 99.49% | 4.22 | 0.273 |
| | CIFAR | RES/128 | MVGG/32 | VGG/256 | RES/32 | VGG/128 | RES/256 | 90.24% | 15 | 6.7 |
| $Acc/Lt$ | ASL | MVGG/16 | VGG/16 | RES/16 | - | - | - | 99.41% | 0.78 | 0.07 |
| | CK+ | VGG/16 | VGG/64 | VGG/128 | - | - | - | 97.97% | 2.1 | 0.37 |
| | CIFAR | VGG/32 | VGG/32 | RES/32 | - | - | - | 84.07% | 1.62 | 0.158 |
| $Acc/En$ | ASL | VGG/16 | VGG/16 | VGG/64 | - | - | - | 99.27% | 0.72 | 0.048 |
| | CK+ | VGG/16 | VGG/16 | MVGG/64 | MVGG/16 | - | - | 89.34% | 2.04 | 0.15 |
| | CIFAR | VGG/32 | VGG/32 | VGG/64 | VGG/128 | VGG/128 | - | 83.33% | 1.71 | 0.183 |

CIFAR-10 dataset with $^{accuracy}/_{latency}$ fitness function, the identified model uses only three blocks having only 32 kernels each, which shows latency improvement to $1.62$ ms and energy reduction to $0.158$ mJ, but the accuracy drops to $84.07\%$.

The $^{accuracy}/_{energy}$ fitness function tends to identify shallow models and avoids the RES blocks. A possible reason for avoiding the RES blocks might be the usage of the global accumulator to implement the skip-connections in the IMC architecture, which leads to more data transfer and higher energy consumption. With $^{accuracy}/_{energy}$ fitness function on the CIFAR-10 dataset, a 5-block model is picked where none of the blocks are RES blocks. The model shows as low as $0.183$ mJ energy consumption, with an accuracy drop to $83.33\%$ and a latency of $1.71$ ms. It is noticeable that putting the same weights to accuracy and other hardware metrics leads to significant drops in accuracy in some of the datasets such as CIFAR-10 and CK+ with $^{accuracy}/_{energy}$ fitness function. One potential solution is to increase the exponent of the accuracy metric in the fitness function such as $^{accuracy^n}/_{energy}$ where $n > 1$. We also observe that for none of the datasets and fitness functions, models with 7 or 8 blocks are selected although it was a possibility in the search space. It is possible that those blocks would be used for more challenging datasets with more complex features.

## V. Conclusion

In this paper, we explored multi-objective neural architecture search for efficient deployment of CNNs in analog in-memory computing (IMC) architectures. We constructed a search space using various building blocks inspired by the convolution layers in VGG and ResNet models. We utilized multi-objective fitness functions combining network accuracy with hardware performance metrics such as latency and energy. Using Bayesian optimization, we explored the search space over three different image classification datasets to identify the best model based on the suggested fitness functions, which revealed some interesting nuances. The exploration with the $accuracy$ fitness function identified deeper models with a higher number of kernels as optimal solutions. When the hardware evaluation metrics are incorporated into the fitness function, such as $^{accuracy}/_{latency}$ and $^{accuracy}/_{energy}$, shallower models with a lower number of kernels are selected. Moreover, our observations exhibit that the optimization algorithm avoids using RES blocks with skip connection in the network architecture when energy is incorporated into the fitness function. This can be associated with the particular specification of the analog IMC architecture which uses a global accumulator to handle the skip connections.

## Acknowledgment

This work is supported in part by the National Science Foundation (NSF) under grant number 2409697.

## References

[1] D. Kim, C. Yu, S. Xie, Y. Chen, J.-Y. Kim, B. Kim, J. P. Kulkarni, and T. T.-H. Kim, "An overview of processing-in-memory circuits for artificial intelligence and machine learning," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 2, pp. 338–353, 2022.

[2] M. E. Elbtity, B. Reidy, M. H. Amin, and R. Zand, "Heterogeneous integration of in-memory analog computing architectures with tensor processing units," in *Proceedings of the Great Lakes Symposium on VLSI 2023*, ser. GLSVLSI '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 607–612. [Online]. Available: https://doi.org/10.1145/3583781.3590256

[3] A. Ankit, I. E. Hajj, S. R. Chalamalasetti, G. Ndu, M. Foltin, R. S. Williams, P. Faraboschi, W.-m. W. Hwu, J. P. Strachan, K. Roy, and D. S. Milojicic, "Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 715–731. [Online]. Available: https://doi.org/10.1145/3297858.3304049

[4] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16, 2016, p. 27–39.

[5] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," ser. ISCA '16. IEEE Press, 2016, p. 14–26. [Online]. Available: https://doi.org/10.1109/ISCA.2016.12

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

[7] R. Zand, A. Roohi, and R. F. DeMara, "Fundamentals, modeling, and application of magnetic tunnel junctions," in *Nanoscale Devices*. CRC Press, 2018, pp. 337–368.

[8] M. Elbtity, A. Singh, B. Reidy, X. Guo, and R. Zand, "An in-memory analog computing co-processor for energy-efficient cnn inference on mobile devices," in *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2021, pp. 188–193.

[9] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.

[10] M. H. Amin, M. E. Elbtity, and R. Zand, "Xbar-partitioning: A practical way for parasitics and noise tolerance in analog imc circuits," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 4, pp. 867–877, 2022.

[11] M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, and E. Eleftheriou, "Mixed-precision in-memory computing," *Nature Electronics*, vol. 1, no. 4, pp. 246–253, 2018.

[12] M. H. Amin, M. Elbtity, and R. Zand, "Interconnect parasitics and partitioning in fully-analog in-memory computing architectures," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022, pp. 389–393.

[13] S. Kim, H.-D. Kim, and S.-J. Choi, "impact of synaptic device variations on classification accuracy in a binarized neural network," *Scientific reports*, vol. 9, no. 1, pp. 1–7, 2019.

[14] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.

[15] D. B. Strukov, "Endurance-write-speed tradeoffs in nonvolatile memories," *Applied Physics A*, vol. 122, no. 4, p. 302, Mar 2016.

[16] P.-Y. Chen, X. Peng, and S. Yu, "Neurosim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3067–3080, 2018.

[17] Z. Zhu, H. Sun, T. Xie, Y. Zhu, G. Dai, L. Xia, D. Niu, X. Chen, X. S. Hu, Y. Cao, Y. Xie, H. Yang, and Y. Wang, "Mnsim 2.0: A behavior-level modeling tool for processing-in-memory architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 11, pp. 4112–4125, 2023.

[18] M. H. Amin, M. E. Elbtity, and R. Zand, "Imac-sim: A circuit-level simulator for in-memory analog computing architectures," in *Proceedings of the Great Lakes Symposium on VLSI 2023*, ser. GLSVLSI '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 659–664. [Online]. Available: https://doi.org/10.1145/3583781.3590264

[19] H. Sun, Z. Zhu, C. Wang, X. Ning, G. Dai, H. Yang, and Y. Wang, "Gibbon: An efficient co-exploration framework of nn model and processing-in-memory architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 11, pp. 4075–4089, 2023.

[20] W. Jiang, Q. Lou, Z. Yan, L. Yang, J. Hu, X. S. Hu, and Y. Shi, "Device-circuit-architecture co-exploration for computing-in-memory neural accelerators," *IEEE Transactions on Computers*, vol. 70, no. 4, pp. 595–605, 2021.

[21] Z. Yuan, J. Liu, X. Li, L. Yan, H. Chen, B. Wu, Y. Yang, and G. Sun, "Nas4rram: neural network architecture search for inference on rram-based accelerators," *Science China Information Sciences*, vol. 64, no. 6, p. 160407, May 2021. [Online]. Available: https://doi.org/10.1007/s11432-020-3245-7

[22] Z. Yan, D.-C. Juan, X. S. Hu, and Y. Shi, "Uncertainty modeling of emerging device based computing-in-memory neural accelerators with application to neural architecture search," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021, pp. 859–864.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[24] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: a python library for model selection and hyperparameter optimization," *Computational Science & Discovery*, vol. 8, no. 1, p. 014008, 2015.

[25] "Sign language mnist." [Online]. Available: https://www.kaggle.com/datamunge/sign-language-mnist

[26] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression," in *2010 ieee computer society conference on computer vision and pattern recognition-workshops*. IEEE, 2010, pp. 94–101.

[27] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.