Multi-Objective Bayesian Optimization for Efficient HDnn-PIM Software-Hardware Co-Design with Metric Constraints

***Abstract*—The real-world software-hardware co-design for AI accelerators must satisfy design constraints for multiple metrics, including accuracy, power, performance, area (PPA). Such co-design relies on design space exploration (DSE) that explores the whole design space to find eligible design points that meet the requirements. However, the DSE is inefficient due to the significant time and computational resources required to evaluate undesirable design points. In this work, we propose the first multi-objective software-hardware co-optimization framework with metric constraints for AI accelerators. We first define the multi-objective software-hardware co-optimization problem with metric constraints for AI accelerator. We then develop a constraint-guided Bayesian Optimization framework to enhance the efficiency of the design process. Our method prioritizes regions within the design space that satisfy the desired criteria, effectively guiding the optimization process towards eligible data points. Specifically, we apply the optimization framework to optimize the full-stack HDnn-PIM design, which is a difficult multi-objective optimization task, with substantial variability in accuracy and hardware costs. Experimental results show that considering constraints in the optimization process constantly improves the quality of eligible trade-offs and also increases the eligible rate by up to 3.3x.**

***Index Terms*—Software-Hardware Co-Design, Bayesian Optimization (BO), Hyperdimensional Computing, Processing-in-Memory (PIM)**

I. Introduction

Edge computing commonly uses machine learning techniques, such as neural networks and hyperdimensional (HD) computing [14], to process incoming data. Efficiency and accuracy are critical for such applications, which has led to development of accelerators. Such accelerators are typically hand-crafted due to the lack of fast and accurate techniques for design exploration, resulting in sub-optimal implementations.

To optimize and balance the design metrics, data space exploration (DSE) is required to find the optimal design points. Previous works [4], [13], [19], [22], [24], [26] have proposed several multi-objective software-hardware co-optimization methods to consider the overall design parameters and balance between different metrics in a full-stack manner. These multi-objective co-optimization methods are efficient DSE methods that find design points with the optimal trade-offs between accuracy and PPA metrics. However, they failed to consider metric constraints, which are essential in real-world hardware accelerator design. Without constraint consideration, the optimization methods will treat all design trade-offs equally, resulting into ineligible trade-offs that fail to meet metric constraints. Such DSE methods are still far from ideal because they spend significant time and resources evaluating ineligible design points, especially for complex software-hardware co-design problems.

In this paper, we develop the constraint-guided multi-objective software-hardware Bayesian co-optimization method that can quickly search the design space of accelerators and provide more optimal design choices that meet the metric constraints. In particular, our method focuses on regions of the design space aligned with specified constraints. By integrating a constraint function informed by prior knowledge of the metrics, our approach improves DSE efficiency, enabling faster and more accurate identification of optimal configurations satisfying constraints. Even in the face of non-ideal noisy hardware, such as when using emerging non-volatile memories, we demonstrate the capabilities of our method on HDnn-PIM accelerator [8], which combines HD computing with convolutional neural networks (CNNs), and accelerates using error-prone resistive RAM (ReRAM) processing-in-memory (PIM).

Our main contributions are:

- To the best of our knowledge, we are first to tackle the multi-objective software-hardware co-optimization problem with metric constraints. The problem definition better complies with real-world problems that have design metric requirements.
- We develop a novel constraint-guided multi-objective Bayesian optimization framework for software-hardware co-optimization. The framework prioritizes exploring data points that are likely to meet output constraints. This approach improves optimization efficiency by focusing on increasing the hypervolume of the feasible design space.
- We apply the framework on HDnn-PIM design on various datasets. Experimental results show that considering constraints in the optimization process constantly improves the hypervolume of eligible points and also increases the eligible rate by up to 1.7x.

II. Related Work and Background

A. Multi-Objective Accelerator Design

There have been studies on multi-objective accelerator design for various targets in the literature. CoSA [13] is a constrained-optimization-based approach for scheduling tasks on DNN accelerators. Yang et al. [26] provided a software-hardware co-design methodology for mapping on PIM-based architecture. UNICO [22] employed multi-objective Bayesian optimization to optimize hardware design and software mapping. Wohrle et al. [24] also used Bayesian optimization to find parameters for

hardware architecture and physical design. Das et al. [4] utilized a multi-objective genetic algorithm to co-optimize chiplet-based hardware accelerator and mapping. None of the above works consider design metric requirements and overlooked the possibility to improve optimization efficiency when considering metric constraints. Shi et al. [23], on the other hand, proposed a Bayesian optimization method that takes metric constraints into consideration, but in a single-objective manner that only optimizes the energy-delay product (EDP).

B. Bayesian Optimization with Output Constraints

Bayesian Optimization with constraints extends traditional BO by incorporating user-defined constraints into the selection of optimal solutions [9]. This line of method efficiently guides the search towards areas that satisfy both the user's constraints and objectives. This enhancement significantly improves the efficiency and relevance of the optimization, particularly in complex, multi-objective problems for identifying optimal design and operating conditions, especially in scenarios where traditional experimental approaches are impractical due to high costs and risks. State-of-the-art methods, such as MORBO [7], focus on balancing competing objectives using metrics like Expected Hypervolume Improvement (EHVI). q-noisy Expected Hypervolume Improvement (qNEHVI) [6] enhances optimization efficiency by extending NEHVI principles to parallel evaluations. By incorporating auxiliary outcome constraints, qNEHVI integrates output requirements directly into the optimization process, rather than treating constraints as an afterthought.

However, the above work incorporates output constraints by manually designing the indicator function, which cannot directly extend to the case where the constraint is complex, non-linear, or high-dimensional. In such cases, manually defining an indicator function may fail to capture the interdependencies of the constraints, leading to suboptimal performance. Recently, there has been progress on multi-objective Bayesian optimization that have addressed constraints in the output space, but still leaving gaps in efficiency and adaptability. Predictive entropy search for multi-objective Bayesian optimization with constraints (PESMOC) [10] introduces a strategy that iteratively optimizes objectives and constraints, maximizing the reduction of uncertainty in the Pareto set. PESMOC decomposes the acquisition function into objective-specific and constraint-specific components, which allows for more flexibility and potentially lower evaluation costs. While methods like PESMOC may provide a flexible and uncertainty-reducing approach, its computational complexity and reliance on decomposed acquisition functions may hinder scalability and efficiency in high-dimensional or dynamic scenarios, making it less suitable for our comparison.

C. Hyperdimensional Computing

HD computing is a lightweight machine learning paradigm demonstrating its versatility across a wide range of domains, including natural language processing [12], robotics [20], biomedical applications [3], edge computing [15], etc. Hyper-dimensional (HD) only requires extremely lightweight highly parallel operations for both training and inference stages by operating with inputs encoded in a high-dimensional space [14].

HD encoding. This is the first stage in the HD computing paradigm and projects input data representations in a distributed, holistic, and high-dimensional space. Random projection is one of the most used and effective encoders, using matrix-vector operations to map input data into a d-dimensional hypervector (HV). Given an input sample x laying in a relatively low-dimensional space f, the corresponding d-dimensional HV $\vec{\mathcal{H}}$ is computed as: $\vec{\mathcal{H}} = \phi\big(\mathbf{P} \cdot \vec{x}\big)$, where ϕ is an optional binarization function (e.g., sign function).

HD Classification. An HD model consists of a set of class HVs. The HD model classifies input samples using class HVs, each representing a specific class. Classification requires calculating similarity scores between the encoded input HV and class HVs, either as cosine similarity or Hamming distance, in the case of binary data. The predicted class is the one whose corresponding HV has the highest similarity score with the input. Mathematically, the predicted class is computed as: $y = argmax_i\big(\sigma(\vec{\mathcal{H}}, \vec{\mathcal{C}}_i)\big)$, where σ represents the similarity function, $\vec{\mathcal{C}}$ is the set of class HVs, $\vec{\mathcal{H}}$ represents the encoded input vector, and y is the predicted label.

D. HDnn-PIM

HD enables efficient and parallelized computations. To further improve accuracy, recent research [8] has integrated CNNs with HD computing. This hybrid approach, termed HDnn, leverages CNNs for initial feature extraction, followed by HD-based classification. This combination significantly improves accuracy while reducing memory and computational costs compared to traditional CNNs. To further enhance performance, the HDnn-PIM architecture employs PIM techniques such as ReRAM in the HD classifier. By performing computations directly within memory, HDnn-PIM reduces data movement bottlenecks, offering a faster and more energy-efficient solution than conventional GPU-based methods. However, ReRAM introduces noise [18] to HD inference, exacerbating the classification accuracy and thus leading to more variability in the evaluation results.

III. Overview

A. HDnn-PIM Software-Hardware Co-Optimization Framework

In this work, we propose a novel and generalized optimization framework for the software-hardware co-optimization problem with metric constraints. Specifically, we focus on full-stack HDnn-PIM design, which is particularly challenging due to the metric variability of in the design process. Note that our approach can not only address full-stack HDnn-PIM design but can also extend to optimize other hardware designs, such as DNN accelerators and other HD accelerators. Our optimization framework has two main features: 1) The framework is multi-objective that aims to find the optimal trade-offs between four metrics: accuracy, power, performance, and area. 2) The framework efficiently finds eligible points that satisfy the metric

constraints through constraint guidance. We build the framework using a constraint-guided multi-objective Bayesian optimization (MOBO) framework. This MOBO framework is an iterative optimization framework that consists of a constraint-guided (MOBO) algorithm and a full-stack HDnn-PIM design evaluator. The algorithm takes in data from previous evaluation results, trains the surrogate model, and predicts the next parameter set that is likely to be an optimal trade-off that satisfies the constraints. Furthermore, with constraint guidance, the algorithm efficiently explores regions of design space that are more likely to contain eligible design points. We detail our optimization algorithm in Section IV.

B. Full-Stack AI Accelerator Design

Full-stack AI accelerator design is a software-hardware co-design problem. Designers first decide the AI model hyperparameters and train the model to obtain high accuracy on certain datasets. After fixing the model parameters, designers then tailor the hardware accelerator to the model in order to optimize PPA. This approach simplifies the process by decoupling the software and hardware design. However, different model hyperparameters not only directly affect the accuracy obtained from the training process, but also influence the energy and delay in inference. Although it is straightforward that larger model sizes contribute to higher energy and delay, the contribution to accuracy is complicated. Also, different hardware design choices not only lead to trade-offs among PPA metrics, but also introduce noise in inference that affects the accuracy, especially for non-ideal emerging memory technologies. The inter-correlation of the software and hardware parameters to the design metrics is non-trivial without sophisticated optimization methods. Furthermore, with metric requirements, unguided optimization methods are even more likely to waste time and computing resources on evaluating ineligible data points that fail to meet the requirements, especially for designs with complicated design trade-offs such as HDnn-PIM.

C. Full-Stack HDnn-PIM Design Problem Formulation

The HDnn algorithm in this work consists of conv2d layers, one RP encoder, and an HD inference, as illustrated in Figure 1. The tunable parameters for the conv2d layers are kernel sizes k, strides str, and numbers of output channels ch. Note that the number of input channels of the next conv2d layer has to be the same as the number of output channels of the previous conv2d layer. Then the encoder encodes the output of the second conv2d layer into a d-dimensional hypervector before HD inference. The choice of the algorithm parameters affects both accuracy and PPA and implicates complicated trade-offs in between the metrics.

Figure 2 demonstrates the HDnn-PIM accelerator. There are two separate parts of the HDnn-PIM accelerator, the ASIC component and the ReRAM component. The ASIC component implements the accelerators of the conv2d layers and the encoder by respective processing elements (PE). Each PE consists of an input buffer, a 16-kB SRAM, and a 2-D MAC array. These MAC arrays have design parameters X and Y, representing the x and y dimensionality. Increasing X and Y trades larger

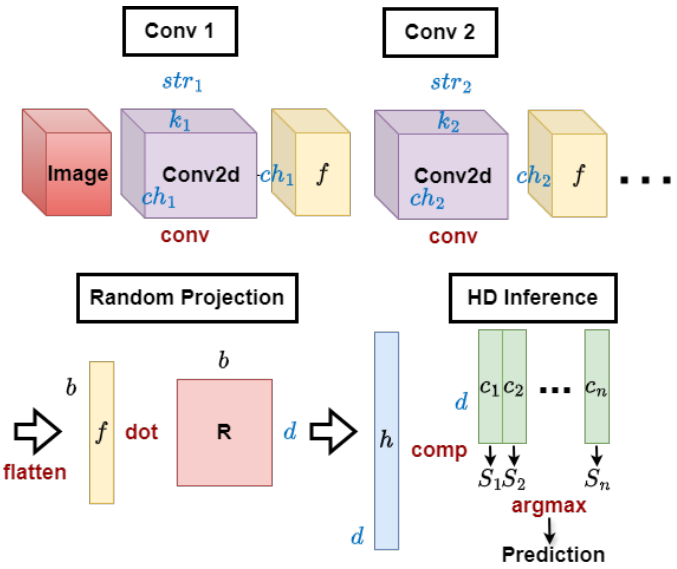

Fig. 1. The HDnn image classification algorithm. Tunable design parameters are in blue. Table I gives a detailed description of the parameters.

area for shorter delay. The ReRAM component implements HD inference by storing class hypervectors in the memory cells and performing inference through parallel ReRAM operation. The choice of ReRAM array size allows for a trade-off between energy consumption, delay and area, with larger arrays reducing energy usage and area but increasing delay.

Before loading the parameters to the HDnn-PIM accelerator, we train the conv2d layers and the class hypervectors. We first connect the two conv2d layers with specified parameters to two inner linear layers with size being d_{in}. Then we train the conv2d and linear layers to minimize the binary cross entropy. After training the conv2d layers, we fix the parameters of the conv2d layers and connect them to the HDnn algorithm as the feature extractor. We then input training data into the feature extractor and RP to train the class hypervectors. The evaluation of design parameter set involves training the conv2d layers and class hypervectors from scratch, causing significant discrepancy between accuracy and PPA results. In this work, we fix the CNN to be 2 conv2d layers following by a (2,2) max-pooling.

IV. Optimization Method

In this section, we introduce our optimization methodology that extends traditional multi-objective optimization paradigms by seamlessly integrating output constraints within the Bayesian Optimization (BO) framework [6]. Our methodology ensures that most generated solutions satisfy predefined feasibility criteria by explicitly incorporating constraints into the optimization process. This is particularly pertinent in complex design spaces, where navigating intricate trade-offs between competing objectives requires careful consideration. This constrained BO approach addresses the limitation of conventional methods. For example, the feasibility first approach focuses solely on finding feasible solutions, causing missed opportunities for better but initially infeasible options. The penalty method transforms the problem by adding penalties for

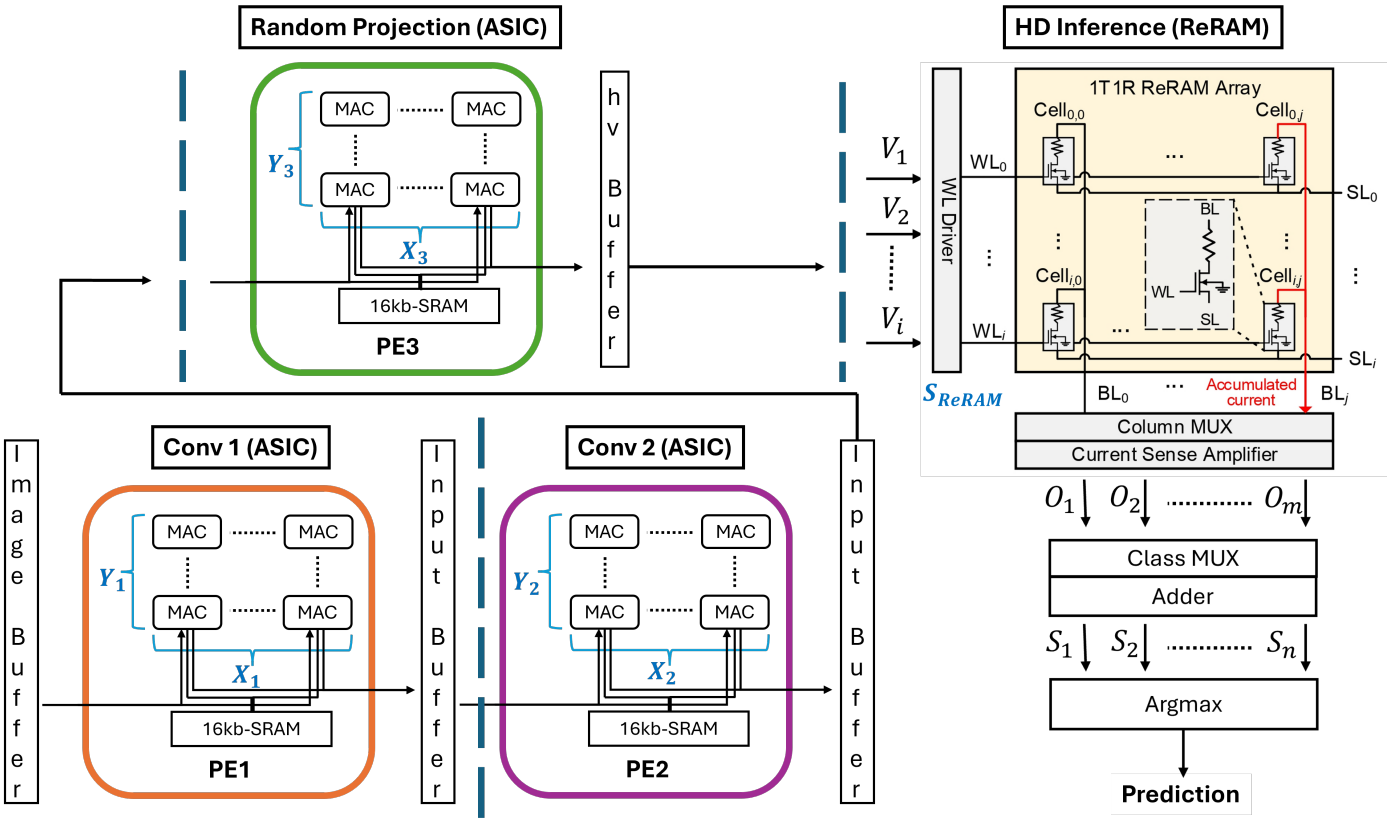

Fig. 2. The HDnn-PIM accelerator architecture for image classification. Tunable design parameters are in blue. Dash lines separate different stages in the pipeline. Table I describes the parameters.

constraint violations, but poorly tuned penalties can mislead the optimization process. Other methods, such dynamic thresholding or repair operators, can create ambiguity or push solutions away from optimal areas.

A. Multi-Objective Software-Hardware Co-Optimization with Metric Constraints

In a multi-objective optimization problem, the goal is to maximize multiple conflicting objectives. In our case, the objective is to maximize accuracy while minimizing power, performance, and area (PPA) metrics. This poses a challenge due to the inherent trade-offs between these objectives. Our optimization framework is designed to systematically address these trade-offs, facilitating the identification of configurations that optimize accuracy while concurrently minimizing PPA. Formally, the problem can be expressed as:

$$\underset{\mathbf{x}\in\mathbb{X}}{\operatorname{argmax}}\, f_{Acc}\left(\mathbf{x}\right), -f_{Energy}\left(\mathbf{x}\right), -f_{Delay}\left(\mathbf{x}\right), -f_{Area}\left(\mathbf{x}\right) \tag{1}$$

where each function $f : \mathbb{X} \subset \mathbb{R}^n \rightarrow \mathbb{Y} \subset \mathbb{R}$ represents the distinct objectives. $\mathbb{X}$ corresponds to the design space, and the collection of points in $\mathbb{R}^4$ corresponds to the objective space. In the context of AI accelerator design, the objectives may include accuracy, power, performance, area, with additional constraints for the objectives to meet certain thresholds.

A solution $\mathbf{x}^\star \in \mathbb{X}$ is Pareto-optimal if no other solution $\mathbf{x} \in \mathbb{X}$ improves at least one objective without causing a deterioration. The Pareto set is formally defined as:

$$\mathbb{X}^\star = \{\mathbf{x}^\star \in \mathbb{X} | \nexists \mathbf{x} \in \mathbb{X} : \mathbf{f}(\mathbf{x}) \succ \mathbf{f}(\mathbf{x}^\star)\} \tag{2}$$

where $\mathbf{y} \succ \mathbf{y}'$ indicates that $\mathbf{y} \neq \mathbf{y}'$ and $y_i \geq y_i'$ for all i.

The goal of multi-objective optimization is to simultaneously optimize multiple conflicting objectives while adhering to specific constraints. Formally, the problem can be expressed as:

$$\begin{aligned} &\underset{\mathbf{x}\in\mathbb{X}}{\operatorname{argmax}}\, f_{Acc}\left(\mathbf{x}\right), -f_{Energy}\left(\mathbf{x}\right), -f_{Delay}\left(\mathbf{x}\right), -f_{Area}\left(\mathbf{x}\right) \\ &\text{subject to } c_{Acc}(\mathbf{x}), c_{Energy}(\mathbf{x}), c_{Delay}(\mathbf{x}), c_{Area}(\mathbf{x}) \geq 0 \end{aligned} \tag{3}$$

where $c_i(\mathbf{x})$ represents each of the 4 constraint functions. These constraints ensure that the solutions are feasible in terms of accuracy and PPA metrics.

B. Incorporating Output Constraints in Bayesian Optimization with Acquistion Function

In this study, we employ Monte-Carlo (MC) based acquisition functions [5] to effectively incorporate output constraints into the optimization process. Monte Carlo-based acquisition typically supports a parallel version, designed to elect a batch of candidate points that are likely to improve the Pareto front, while accounting for noise in the objective functions.

These lines of method can be naturally extended to handle auxiliary constraints specified in the objective, directing the optimization toward configurations that meet performance and resource efficiency criteria within the defined limits. The most natural metric is defined for a single candidate point:

$$M(f(x), c(x)) = M(f(x))\mathbb{1}[c(x) \leq 0] \tag{4}$$

where $M(f, c)$ represents the metric used in the acquisition function, such as the expected hypervolume improvement (EHVI) [5] or noisy expected hypervolume improvement (NEHVI) [6]. These method uses hypervolume indicator as the main metric. The hypervolume indicator (HV) of a finite approximate Pareto frontier $\mathcal{P}$ is usually defined as the M-dimensional Lebesgue measure λ_M of the space dominated by $\mathcal{P}$ and bounded from below by a reference point r, which is assumed to be known and assigned by DM.

$$\text{HV}(\mathcal{P}|r) = \lambda_M\big(\bigcup_{v\in\mathcal{P}}[r, v]\big) \tag{5}$$

, where $[r, v]$ denotes the hyper-rectangle bounded by vertices r and v. The hypervolume improvement (HVI) of a set of points $\mathcal{P}'$ with respect to an existing approximate Pareto frontier $\mathcal{P}$ is then:

$$\text{HVI}(\mathcal{P}'|\mathcal{P}, r) = \text{HV}(\mathcal{P}\cup\mathcal{P}'|r) - \text{HV}(\mathcal{P}|r). \tag{6}$$

In practice, hypervolume computation is achieved by partitioning the region into disjoint axis-aligned hyperrectangles, calculating the volume of each hyperrectangle independently, and summing these volumes to obtain the overall measure.

For example, in the case of using EHVI [5]

$$M_{\text{EHVI}}(f(x)) = \alpha_{\text{EHVI}}(\mathcal{X}|\mathcal{P}) \approx \hat{\alpha}_{\text{EHVI}}(\mathcal{X}|\mathcal{P}) \tag{7}$$

$$= \frac{1}{N}\sum_{t=1}^{N}\text{HVI}(\tilde{f}_t(\mathcal{X})|\mathcal{P}), \tag{8}$$

$$\tag{9}$$

where $\tilde{f}_t \sim p(f|\mathcal{D})$ for $t = 1, ..., N$ and $\mathcal{X} = \{x_i\}_{i=1}^n$. On the other hand, when true Pareto frontier cannot be calculated due to observation noise, the *noisy expected hypervolume improvement* (NEHVI) [6] performs best compared to other acquisition functions, which is defined as

$$M_{\text{NEHVI}}(f(x)) = \int \alpha_{\text{EHVI}}(x|\mathcal{P}_n)p(f|\mathcal{D}_n)df \tag{10}$$

$$\approx \frac{1}{N}\sum_{t=1}^{N}\text{HVI}(\tilde{f}_t(x)|\mathcal{P}_t). \tag{11}$$

Here, P_n denotes the Pareto frontier over $f(X_n)$, and the second part in (10) denotes the posterior probability given previous data $\mathcal{D}_n = \{x_i, y_i, (\Sigma_i)\}_{i=1}^n$, where Σ_i is the noise covariance. The integral in is analytically intractable, so it is standard to use Monte Carlo Integration in (11), where $\tilde{f}_t \sim p(f|D_n)$ for $t = 1, ...N$ are samples from the posterior distribution. In practice, implementation typically adopts box decomposition to enhance efficiency [7].

TABLE I
ALL SOFTWARE AND HARDWARE DESIGN PARAMETERS AND THEIR RANGES.

Parameter	Description	Range
d	HD dimensionality	{1024, 2048 4096, 8192}
k_1, k_2	conv2d kernel size	{3, 5}
str_1, str_2	conv2d stride	{1, 2}
ch_1, ch_2	conv2d channel	{4, 6, 16}, {8, 16, 32}
d_{in}	Inner layer size in CNN training	{1024, 2048 4096}
X_1, X_2, X_3	#MACs on x-axis of PE	{8, 16}
Y_1, Y_2, Y_3	#MACs on y-axis of PE	{8, 16}
S_{ReRAM}	ReRAM size	{128, 256}

Knowing the theoretical and practical form of the adopted acquisition functions, we can view the constrained method as assigning weights on the metric based on the probability of each point's eligibility. While this mechanism may seem simple, it works well because Monte Carlo integration allows sampling from the joint distribution of the objective and constraint functions. By prioritizing configurations that satisfy the constraints while maximizing performance, we enhance the optimization's ability to navigate complex design spaces. This feasibility-weighting on the sample-level approach ensures that the process remains focused on feasible solutions, facilitating the identification of Pareto-optimal configurations that align with prior design requirements.

V. EVALUATION

This section compares the results of different optimization methods to optimize the metrics for the full-stack HDnn-PIM design. We compare the optimization results among 1) random search, 2) EHVI [5], 3) NEHVI [6], 4) EHVI [5] with constraints [5], 5) NEHVI [6] with constraints [6], and show the importance of considering metric constraints in the optimization process. We utilize hypervolume [27] of eligible points as the indicator to assess the quality of eligible trade-offs. Also, we evaluate the eligible rates to assess the efficiency of different optimization methods.

A. Experimental setup

We implement the optimization algorithm in BoTorch [2]. For each optimization, we run 10 random search steps before running the optimization algorithm with 30 trials. We keep the PPA metric values within [0, 1] by dividing the values by energy = $3000\mu J$, delay = $3000\mu s$, and area = $3000mm^2$. We implement the evaluator in Python and run all experiments on different datasets, including MNIST [17], Fashion-MNIST [25], and CIFAR-10 [16] as the workload. The metric constraints are 0.2 for the PPA metrics and 0.9 accuracy for MNIST, 0.8 accuracy for Fashion-MNIST, and 0.3 accuracy for CIFAR-10. We train the conv2d layers with 10 iterations, the HD model with a training step and 10 retraining steps with learning rate being 1.0 for each evaluation. We use PytorX [11] as the noise simulator for ReRAM. We then average 5 independent random evaluations as the accuracy evaluation results. Timeloop [21] is our PPA evaluator for the ASIC component and CiMLoop [1] for the ReRAM component in the HDnn-PIM architecture, both

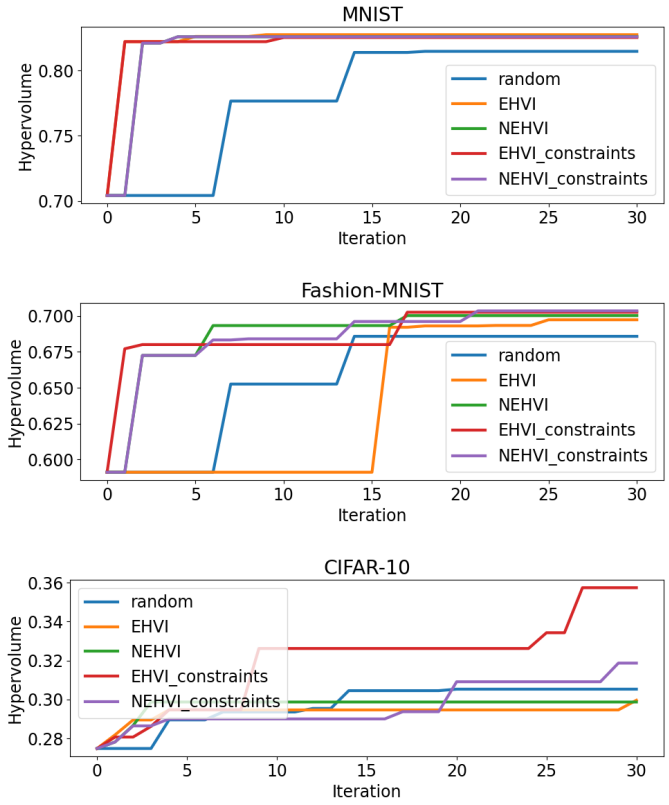

Fig. 3. Eligible hypervolume comparison between different optimization methods versus optimization iteration on different datasets, including MNIST [17], Fashion-MNIST [25] and CIFAR-10 [16].

in 32nm technology. Table I concludes the parameters and their ranges as X in our optimization in the experiments.

We run all experiments with one 11th Gen Intel(R) Core(TM) i7-11700K CPU with 3.60GHz, 62GB memory, and an NVIDIA GeForce RTX 4090 GPU with 24GB memory.

B. Hypervolume Comparison

In our case for hypervolume calculation, we set the reference point as accuracy = 0, energy = 1, delay = 1, and area = 1. We compare the hypervolume of eligible points using different optimization methods as shown in Figure 3. Higher hypervolume value indicates better eligible Pareto set quality, which means better accuracy and PPA values or more trade-off options that meet the metric requirements. Overall, Bayesian methods with constraint consideration perform better than without constraint consideration. In particular, the benefit margin of constraint-considered methods increase as the difficulty of the datasets increases (CIFAR-10 is the most difficult) while in MNIST, EHVI, NEHVI, and EHVI_constraints have the same hypervolume curve. This shows that by integrating constraint consideration in the optimization process, our method provide even better results in more complex tasks. As the number of objectives increase in future research, the optimization process benefits more from constraint guidance, as higher-dimensional and complex problems have a greater risk of generating infeasible solutions without such output considerations.

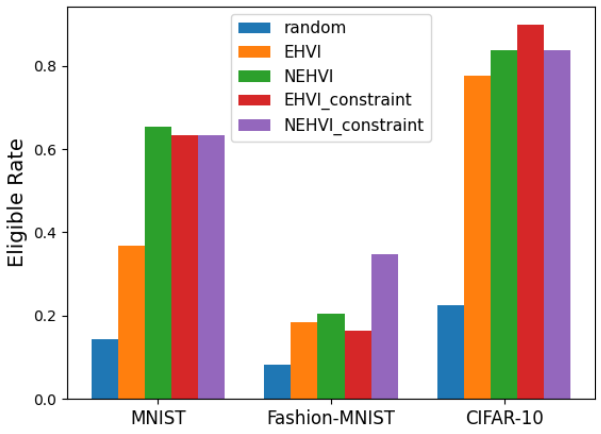

Fig. 4. Eligible rate of different optimization methods on different datasets, including MNIST [17], Fashion-MNIST [25] and CIFAR-10 [16].

C. Eligible Rate Comparison

Also, we take a look at the eligible rate as illustrated in Figure 4. Constraint-considered methods in general provide better eligible rate than their counterparts without constraint consideration. For Fashion-MNIST, NEHVI with constraint consideration outperforms NEHVI without NEHVI without constraint consideration by 1.7x eligible rate. For CIFAR-10 dataset, we also apply stricter constraints with accuracy constraint being 0.35 and get eligible rate for EHVI with and without constraint consideration being 0.53 and 0.16, achieving 3.3x improvement. As discussed in the previous section, the advantage of our constraint-based method becomes more evident when dealing with more difficult tasks. This shows that by natively incorporating the constraint guidance in Bayesian optimization, the optimization process can find not only eligible but high-quality design points.

VI. Conclusion

In this work, we present a novel multi-objective Bayesian optimization framework with metric constraints tailored for the software-hardware co-design of AI accelerators. Our method specifically addresses the inefficiencies in DSE by focusing on feasible design points that meet accuracy and PPA constraints. By guiding the optimization process through constraint-based Bayesian optimization, we demonstrate that the proposed method significantly improves both the quality and the eligible rate of design trade-offs, particularly for complex AI hardware architectures like HDnn-PIM. Experimental results on multiple datasets showed that incorporating constraints leads to a 3.3x increase in the eligible rate compared to conventional methods. The demonstrated improvements highlight the importance of considering metric constraints in real-world AI accelerator design and the potential of our approach to enhance the efficiency and feasibility of DSE in future AI accelerators.

REFERENCES

[1] Tanner Andrulis, Joel S Emer, and Vivienne Sze. Cimloop: A flexible, accurate, and fast compute-in-memory modeling tool. *arXiv preprint arXiv:2405.07259*, 2024.

[2] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[3] Simone Benatti, Fabio Montagna, Victor Kartsch, Abbas Rahimi, Davide Rossi, and Luca Benini. Online learning and classification of emg-based gestures on a parallel ultra-low power platform using hyperdimensional computing. *IEEE transactions on biomedical circuits and systems*, 13(3):516–528, 2019.

[4] Abhijit Das, Enrico Russo, and Maurizio Palesi. Multi-objective hardware-mapping co-optimisation for multi-dnn workloads on chiplet-based accelerators. *IEEE Transactions on Computers*, 2024.

[5] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. *Advances in Neural Information Processing Systems*, 33:9851–9864, 2020.

[6] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *Advances in Neural Information Processing Systems*, 34:2187–2200, 2021.

[7] Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Multi-objective bayesian optimization over high-dimensional search spaces. In *Uncertainty in Artificial Intelligence*, pages 507–517. PMLR, 2022.

[8] Arpan Dutta, Saransh Gupta, Behnam Khaleghi, Rishikanth Chandrasekaran, Weihong Xu, and Tajana Rosing. Hdnn-pim: Efficient in memory design of hyperdimensional computing with feature extraction. GLSVLSI'22.

[9] Peter I. Frazier. A tutorial on bayesian optimization, 2018.

[10] Eduardo C. Garrido-Merchán and Daniel Hernández-Lobato. Predictive entropy search for multi-objective bayesian optimization with constraints. *Neurocomputing*, 361:50–68, 2019.

[11] 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. DAC'19.

[12] Alejandro Hernández-Cano, Yang Ni, Zhuowen Zou, Ali Zakeri, and Mohsen Imani. Hyperdimensional computing with holographic and adaptive encoder. *Frontiers in Artificial Intelligence*, 7:1371988, 2024.

[13] Qijing Huang, Minwoo Kang, Grace Dinh, Thomas Norell, Aravind Kalaiah, James Demmel, John Wawrzynek, and Yakun Sophia Shao. Cosa: Scheduling by constrained optimization for spatial accelerators. ISCA'21.

[14] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive computation*, 1:139–159, 2009.

[15] Behnam Khaleghi, Hanyang Xu, Justin Morris, and Tajana Šimunić Rosing. tiny-hd: Ultra-efficient hyperdimensional computing engine for iot applications. DATE'21.

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[18] Yu-Hsuan Lin, Chao-Hung Wang, Ming-Hsiu Lee, Dai-Ying Lee, Yu-Yu Lin, Feng-Min Lee, Hsiang-Lan Lung, Keh-Chung Wang, Tseung-Yuen Tseng, and Chih-Yuan Lu. Performance impacts of analog reram non-ideality on neuromorphic computing. *IEEE Transactions on Electron Devices*, 66(3):1289–1295, 2019.

[19] Mohammad Loni, Sima Sinaei, Ali Zoljodi, Masoud Daneshtalab, and Mikael Sjödin. Deepmaker: A multi-objective optimization framework for deep neural networks in embedded systems. *Microprocessors and Microsystems*, 73:102989, 2020.

[20] Anton Mitrokhin, P Sutor, Cornelia Fermüller, and Yiannis Aloimonos. Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception. *Science Robotics*, 4(30):eaaw6736, 2019.

[21] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W Keckler, and Joel Emer. Timeloop: A systematic approach to dnn accelerator evaluation. ISPASS'19.

[22] Bahador Rashidi, Chao Gao, Shan Lu, Zhisheng Wang, Chunhua Zhou, Di Niu, and Fengyu Sun. Unico: Unified hardware software co-optimization for robust neural network acceleration. MICRO'23.

[23] Zhan Shi, Chirag Sakhuja, Milad Hashemi, Kevin Swersky, and Calvin Lin. Learned hardware/software co-design of neural accelerators. *arXiv preprint arXiv:2010.02075*, 2020.

[24] Hendrik Wöhrle, Felix Schneider, Fabian Schlenke, Denis Lebold, Mariela De Lucas Alvarez, Frank Kirchner, and Michael Karagounis. Multi-objective surrogate-model-based neural architecture and physical design co-optimization of energy efficient neural network hardware accelerators. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 70(1):40–53, 2022.

[25] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[26] Xiaoxuan Yang, Shiyu Li, Qilin Zheng, and Yiran Chen. Improving the robustness and efficiency of pim-based architecture by sw/hw co-design. ASP-DAC'23.

[27] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.