Multi-Objective Optimization of ReRAM Crossbars for Robust DNN Inferencing under Stochastic Noise

Xiaoxuan Yang*, Syrine Belakaria†, Biresh Kumar Joardar*, Huanrui Yang*,
Janardhan Rao Doppa†, Partha Pratim Pande†, Krishnendu Chakrabarty*, Hai (Helen) Li*
* Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA,
† School of Electrical Engineering & Computer Science, Washington State University, Pullman, WA, USA
* {xy92, bireshkumar.joardar, huanrui.yang, krish, hai.li}@duke.edu, † {syrine.belakaria, jana.doppa, pande}@wsu.edu

***Abstract*—Resistive random-access memory (ReRAM) is a promising technology for designing hardware accelerators for deep neural network (DNN) inferencing. However, stochastic noise in ReRAM crossbars can degrade the DNN inferencing accuracy. We propose the design and optimization of a high-performance, area-and energy-efficient ReRAM-based hardware accelerator to achieve robust DNN inferencing in the presence of stochastic noise. We make two key technical contributions. First, we propose a stochastic-noise-aware training method, referred to as ReSNA, to improve the accuracy of DNN inferencing on ReRAM crossbars with stochastic noise. Second, we propose an information-theoretic algorithm, referred to as CF-MESMO, to identify the Pareto set of solutions to trade-off multiple objectives, including inferencing accuracy, area overhead, execution time, and energy consumption. The main challenge in this context is that executing the ReSNA method to evaluate each candidate ReRAM design is prohibitive. To address this challenge, we utilize the continuous-fidelity evaluation of ReRAM designs associated with prohibitive high computation cost by varying the number of training epochs to trade-off accuracy and cost. CF-MESMO iteratively selects the candidate ReRAM design and fidelity pair that maximizes the information gained per unit computation cost about the optimal Pareto front. Our experiments on benchmark DNNs show that the proposed algorithms efficiently uncover high-quality Pareto fronts. On average, ReSNA achieves 2.57% inferencing accuracy improvement for ResNet20 on the CIFAR-10 dataset with respect to the baseline configuration. Moreover, CF-MESMO algorithm achieves 90.91% reduction in computation cost compared to the popular multi-objective optimization algorithm NSGA-II to reach the best solution from NSGA-II.**

***Index Terms*—ReRAM crossbar, stochastic noise, DNN inferencing, efficient hardware, multi-objective optimization.**

I. Introduction

Resistive random access memory (ReRAM) has emerged as a promising nonvolatile memory technology due to its multi-level cell, small cell size, and low access time and energy consumption. Prior work has shown that the crossbar structure of ReRAM arrays can efficiently execute matrix-vector multiplication [1], [2], the predominant computational kernel associated with deep neural networks (DNNs). ReRAM-based accelerators for fast and efficient DNN training and inferencing have been extensively studied [3]–[8].

However, a key challenge in executing DNN inferencing [9]–[11] on ReRAM-based architecture arises due to nonidealities of ReRAM devices, which can degrade the accuracy of inferencing. Since DNN inferencing involves a sequence of forward computations over DNN layers, errors due to device nonidealities can propagate and accumulate, resulting in incorrect predictions. The nonidealities of ReRAM crossbars can be classified into two broad categories. The first category includes device defects (e.g., stuck-at-high or stuck-at-low resistance [12]) and device reliability issues (e.g., retention failure [13] and resistance drift [14]) that are mostly deterministic in nature and have been addressed in prior work [15]–[20]. The second category includes stochastic noise in ReRAM devices that includes thermal noise [21], shot noise [22], random telegraph noise (RTN) [23], and programming noise [24]. These nonidealities have not been studied for DNN inferencing in prior work.

This work was supported in part by the US National Science Foundation (NSF) under grants CNS-1955196, CNS-1955353, IIS-1845922, OAC-1910213, and by the USA Army Research Office grant W911NF-17-1-0485. Biresh Kumar Joardar was also supported in part by NSF Grant #2030859 to the Computing Research Association for the CIFellows Project.

This paper studies the impact of stochastic noise on DNN inferencing and shows that there is a significant degradation in inferencing accuracy due to the high amplitude of noise and reduced noise margin of high-resolution ReRAM cells. Prior algorithmic solutions [25], [26] mitigate the accuracy degradation due to programming variations, but they are not effective in the presence of stochastic noise [27]. To overcome this challenge, we propose a ReRAM-based Stochastic-Noise-Aware DNN training method (ReSNA) that considers both hardware design configurations and stochastic noise.

For DNN inferencing on ReRAM using ReSNA, key efficiency metrics include hardware area, execution time (latency), and energy consumption. Therefore, we need to solve a complex multi-objective optimization (MOO) problem to achieve robust DNN inferencing on ReRAM crossbars. The input space consists of different ReRAM crossbar configurations, e.g., ReRAM cell resolution, crossbar size, temperature, and operational frequency. The output space consists of the accuracy of DNN inferencing and hardware efficiency metrics, e.g., hardware area, execution time, and energy consumption. The main challenge in solving this optimization problem is that the input space over ReRAM configurations contains a large number (up to 10^7) of available data points, and evaluation of each candidate ReRAM configuration involves executing the ReSNA method, which is computationally prohibitive (e.g., it takes nearly 30 GPU days to run the training on the crossbar simulator [27] for 100 configurations). Our goal is to efficiently uncover the Pareto optimal set of solutions representing the best possible trade-offs among multiple objectives.

To solve this challenging MOO problem, we propose an information-theoretic algorithm referred to as Continuous-Fidelity Max-value Entropy Search for Multi-Objective Optimization (CF-MESMO). We formulate the continuous-fidelity evaluation by varying the number of training epochs for ReSNA to establish an appropriate trade-off between computation cost and accuracy. In each MOO iteration, the candidate ReRAM design and fidelity (number of iterations of ReSNA training) pair is selected based on the maximization of the information gained per unit computation cost about the optimal Pareto front. We perform comprehensive experiments on benchmark DNNs and datasets to evaluate the proposed algorithms. Our results show that ReSNA can significantly increase DNN inferencing accuracy in the presence of stochastic noise on ReRAM crossbars, and CF-MESMO can achieve faster convergence and efficiently uncover high-quality Pareto fronts when compared to prior methods, including NSGA-II [28] and a state-of-the-art single-fidelity multi-objective optimization method called MESMO [29].

The main contributions of this paper are as follows.

- Study of the impact of stochastic noise on DNN inferencing executed on ReRAM crossbars.

- A hardware-aware training method, referred to as ReSNA, to overcome stochastic noise and improve DNN inferencing accuracy.
- An efficient multi-objective optimization algorithm, referred to as CF-MESMO, to approximate optimal Pareto fronts in terms of inferencing accuracy and hardware efficiency.
- Experimental results on a diverse set of benchmark DNNs and datasets to demonstrate the effectiveness of ReSNA and CF-MESMO and their superiority over state-of-the-art methods.

The remainder of this paper is organized as follows. Section II discusses related prior work. Section III explains the problem setup, and Section IV highlights the impact of stochastic noise. Section V presents the ReSNA approach, and Section VI presents the CF-MESMO algorithm. Section VII presents the experimental results. Section VIII concludes this paper.

II. Related Prior Work

We review related prior work on two key aspects of this paper—mitigating device stochastic noise for DNN inferencing and multi-objective optimization for hardware design.

There is limited prior work on mitigating the DNN inferencing accuracy loss due to stochastic noise. Yan et al. [30] proposed a closed-loop circuit that utilizes the inferencing results to stabilize the DNN weights, but the effectiveness of this method was demonstrated only on small DNNs. Long et al. [25] injected Gaussian noise during training to mimic programming noise, and Joshi et al. [26] incorporated device programming variation extracted from experiments during training. However, these methods only considered programming noise while neglecting the other types of noise. Importantly, all these prior methods overlooked the impact of hardware configurations, such as the crossbar size and the resolution of the digital-to-analog converter (DAC) and the analog-to-digital converter (ADC).

He et al. [27] investigated the integration of stochastic noise during the training process, but their method failed to reach the desired DNN inferencing accuracy. Consequently, they suggested lowering the operational frequency such that the noise amplitude is low. In contrast to prior work, we consider all the four types of stochastic noise and propose a ReRAM hardware-aware training method to increase the inferencing accuracy even under high operational frequencies.

Considering both inferencing accuracy and hardware efficiency, we have a complex MOO problem for ReRAM-based hardware design. Candidate MOO algorithms for ReRAM design optimization can be classified into two broad categories. The first category of MOO algorithms has objective functions that are *cheap* to evaluate. AMOSA [31] and NSGA-II [28] are two popular evolutionary algorithms that belong to this category. NSGA-II evaluates the objective functions for various combinations of input variables and organizes the candidate inputs into a hierarchy of subgroups based on the ordering of Pareto dominance. This method takes advantage of the similarity between members of each subgroup and the Pareto dominance and moves towards the promising area of the input space. Unfortunately, NSGA-II requires the evaluation of a large number of candidate inputs and is not suitable for our problem setting, where objectives are expensive.

Second, for *expensive* objective functions, Bayesian optimization (BO) [32] is an effective framework. The key idea is to build a cheap statistical model from past function evaluations and use it to intelligently explore the input space for finding (near-)optimal solutions. Much of the prior work on BO is for single-objective optimization. There is limited work on multi-objective BO [33]–[35], and MESMO [29] is the state-of-the-art algorithm. In contrast to prior work, we exploit the continuous approximations of the objective functions (hardware-aware DNN training on ReRAM designs with varying the number of epochs) to minimize the computation cost to uncover high-quality Pareto front for hardware design.

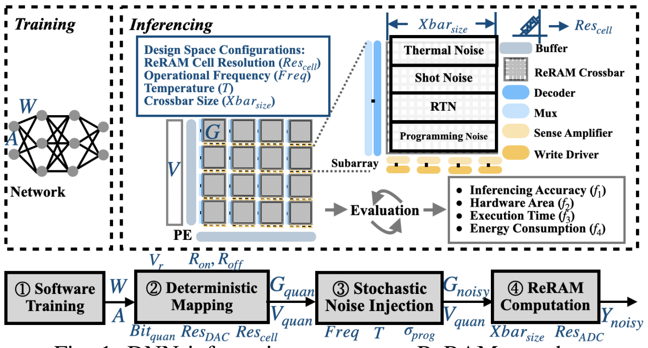

Fig. 1: DNN inferencing process on ReRAM crossbars.

TABLE I: Parameters for inferencing on ReRAM crossbars.

Notation	Meaning
W	DNN weight matrix
A	Activations
Bit_{quan}	Bit number for quantization
R_{on}, R_{off}	Low resistance and high resistance
Res_{cell}	ReRAM cell resolution
Res_{DAC}, Res_{ADC}	DAC and ADC resolution
V_r	ReRAM read voltage
T	Temperature
$Freq$	Operational frequency
σ_{prog}	Programming noise standard deviation
$Xbar_{size}$	Crossbar size

III. Background and Problem Setup

In this section, we first explain how a trained DNN model is deployed on the ReRAM crossbars to perform inferencing. Subsequently, we describe the MOO problem to perform robust DNN inferencing using ReSNA.

A. DNN Inferencing on ReRAM Crossbars

Fig. 1 illustrates the overall flow of deploying a trained DNN model on ReRAM crossbars for inferencing. There are four main steps, as explained below. Table I summarizes the notation associated with the relevant parameters.

① *Software training.* For a given DNN architecture and training dataset, we first perform the training in software. The quantization-aware training technique [36]–[38] can be used to quantize the activations and weights.

② *Deterministic mapping.* The objective of this step is to map the weight matrix of DNN W and the set of activations A to the conductance of the ReRAM cells G_{quan} and crossbar input voltages V_{quan}, according to the resolutions of ReRAM devices and DACs, respectively. When the ReRAM cell resolution is less than the number of bits used in quantization, i.e., $Res_{cell} < Bit_{quan}$, $\lceil Bit_{quan}/Res_{cell} \rceil$ cells are used to represent one weight. A kernel in a convolutional (Conv) layer needs to be first unrolled and mapped to a matrix. As kernels are reused many times during convolution, the kernel in a Conv layer is typically duplicated and deployed on multiple crossbars. Therefore, multiple inputs can be processed simultaneously, increasing parallelism and improving throughput [5]. For a fully-connected (FC) layer, each weight is associated with only one input neuron. Hence, duplication is not necessary.

③ *Stochastic noise injection.* This step mimics the influence of stochastic noise on the conductance values. The noise is modeled using probability distributions [22], [27]. Here G_{noisy} denotes the cell conductance in the presence of thermal noise, shot noise, RTN, and programming noise together. Section IV-A provides more details.

④ *ReRAM-based computation.* This process accumulates the results obtained from ReRAM crossbars and employs ADCs to generate outputs. Here Y_{noisy} denotes the final output of DNN inferencing.

Note that the last three steps together emulate the deployment of DNN inferencing on ReRAM-based hardware. The deterministic mapping needs to be carried out only once. Typically, we need to perform the third and fourth steps multiple times (e.g., ten times) to mimic multiple independent ReRAM deployments on the same device. Based on the multiple runs, we obtain an estimate of DNN inferencing accuracy.

B. MOO problem for ReRAM-based Designs

Our goal is to find ReRAM-based designs with suitable DNN weights to optimize multiple objectives, including inferencing accuracy, hardware area, execution time, and energy consumption.

ReRAM design space. The ReRAM design configuration influences the output objectives. For example, the parameters Bit_{quan}, Res_{DAC}, Res_{ADC} listed in Table I influence the data precision and overall inferencing accuracy. For area overhead, $\lceil Bit_{quan}/Res_{cell} \rceil$ is proportional to the number of cells used in the ReRAM-based design to represent the weights, and $Xbar_{size}$ determines the subarray unit size. For execution time, note that $Freq$ is inversely proportional to the clock cycle. The read voltage V_r and the ReRAM cell resistance range $[R_{on}, R_{off}]$ affect the ReRAM crossbar energy consumption.

MOO formulation. We formulate the MOO problem for robust inferencing on hardware-efficient ReRAM crossbars with stochastic noise as follows. Our input space consists of two parts: the ReRAM design space and the DNN weights. Let $\mathcal{X} \subseteq \mathcal{R}^d$ be the ReRAM design configuration space, which includes the design variables as explained above and also shown in Fig. 1. Each design variable can take values from a bounded candidate set. We need a candidate pair consisting of ReRAM design configuration($\mathbf{x} \in \mathcal{X}$) and DNN weights to be able to evaluate all the output objectives. Without loss of generality, we consider maximizing four objective functions: DNN inferencing accuracy, hardware area, execution time, and energy consumption denoted by $f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}), f_4(\mathbf{x})$, respectively. For each candidate ReRAM design, we execute ReSNA to obtain the DNN weights that give rise to maximum accuracy. Subsequently, we evaluate the objective functions $f_1(\mathbf{x})$, $f_2(\mathbf{x})$, $f_3(\mathbf{x})$, $f_4(\mathbf{x})$.

A design configuration $\mathbf{x}$ is determined to *Pareto dominate* another design $\mathbf{x}'$ if $f_i(\mathbf{x}) \geq f_i(\mathbf{x}')\ \forall i$ and there exists some $j \in \{1, 2, 3, 4\}$ such that $f_j(\mathbf{x}) > f_j(\mathbf{x}')$. An optimal solution of a MOO problem is a set of designs $\mathcal{X}^*$ such that no design $\mathbf{x}' \in \mathcal{X} \setminus \mathcal{X}^*$ pareto-dominates a design $\mathbf{x} \in \mathcal{X}^*$. The solution set $\mathcal{X}^*$ is called the *Pareto set*, and the corresponding set of objective function values is called the *Pareto front*. Our goal is to achieve a high-quality Pareto front for hardware design while minimizing the total computation cost of function evaluations.

IV. Understanding the Impact of Stochastic Noise

In this section, we first discuss the modeling of ReRAM stochastic noise. Next, we demonstrate the impact of stochastic noise on DNN inferencing for specific ReRAM design configurations. Finally, we show that the naïve approach of adding random Gaussian noise cannot improve the robustness of DNN inferencing in the presence of stochastic noise.

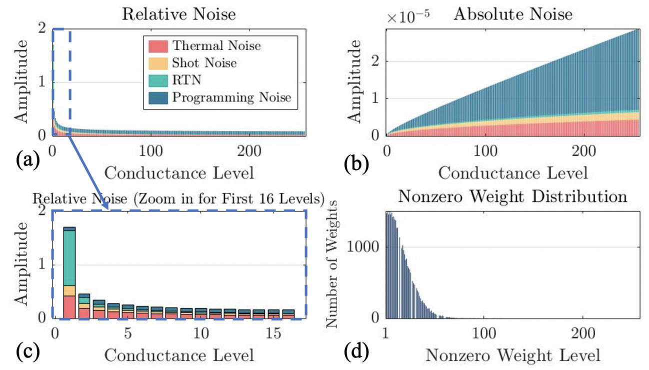

Fig. 2: The distributions of stochastic noise for 8-bit ReRAM cells under $Freq = 500\,\text{MHz}$ and $T = 350\,\text{K}$: (a) in a relative scale, (b) with absolute values, (c) zoomed-in version for relative noise, (d) an example of nonzero weight distribution.

A. Modeling of ReRAM Stochastic Noise

Thermal noise is generated due to the thermal agitation of the charged carriers inside the conductor [39]. *Shot noise* is an electronic noise that originates from the discrete electrons in the current flow. The thermal and shot noise directly affect the current through a device. We convert the change in current to the equivalent conductance change and model these two noise sources using Gaussian distributions [22]: $\Delta G_{thermal} = \mathcal{N}(0, \sqrt{4G \cdot Freq \cdot K_B \cdot T/V})$ and $\Delta G_{shot} = \mathcal{N}(0, \sqrt{2G \cdot Freq \cdot q \cdot V}/V)$, where G and V denote the conductance and terminal voltage respectively. As shown in Table I, $Freq$ denotes the operational frequency, and T denotes the temperature. K_B denotes the Boltzmann constant, and q denotes the electron charge.

Random telegraph noise (RTN) is generated in semiconductors and ultra-thin oxide films. It can be modeled as a Poisson process [23], with the parameters for RTN amplitude (ΔG_{rtn}) reported in [27].

Programming noise is introduced by the programming variation when values are written to a ReRAM device. The programming noise can be estimated using a Gaussian distribution with $\Delta G_{prog} = \mathcal{N}(0, \sigma_{prog} G)$ with standard deviation $\sigma_{prog} = 0.0658$, according to the experimental study reported in [40].

B. Impact of ReRAM Stochastic Noise

Fig. 2 (a)-(b) show the relative and absolute distributions of four kinds of stochastic noise for 8-bit ReRAM cells with $Freq = 500\,\text{MHz}$ and $T = 350\,\text{K}$. Besides, Fig. 2(c) presents the relative noise for the first 16 levels among the 256 conductance levels. Relative noise ($\Delta G/G$) measures the noise amplitude divided by the absolute conductance. Thermal and shot noise have similar patterns: the relative noise is the largest at the smallest conductance level; it then decreases steadily as G increases. The relative RTN presents a sharp peak at the first conductance level, while other conductance levels have much smaller relative RTN. The programming noise increases with the conductance level in the absolute value.

We observe from Fig. 2(c) that the overall amplitude is much higher than each individual case, especially when G is small. Moreover, it is well known that the trained weights of DNNs are concentrated at small values [41]. Fig. 2(d) shows an example distribution of nonzero weights, which is extracted from the 19th layer of ResNet20 with 8-bit quantization. A large number of the weights in this layer are zero; these are not included in the figure. Note that at small values of G, the first three types of stochastic noise dominate. Specifically, the thermal and shot noise are sensitive to the operational frequency. Hence, it is challenging to incorporate high-frequency noise into training. Lowering the operational frequency and reducing noise amplitude

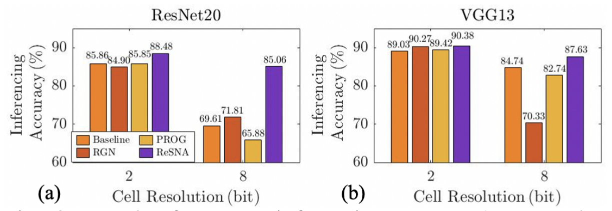

Fig. 3: Results for DNN inferencing on ReRAM crossbars with stochastic noise under different training methods: Baseline, RGN, PROG, and ReSNA training. $Freq = 500\,\text{MHz}$, $T = 350\,\text{K}$, 128×128 crossbars. (a) ResNet20, (b) VGG13.

could be an option, as suggested in previous work [27]. However, such an approach will seriously constrain the use and potential of ReRAM-based hardware due to the exclusion of high operational frequency and short execution time for DNN inferencing.

Utilizing high-resolution cells is another challenge. For example, increasing the cell resolution from 2 to 8 bits can reduce the number of crossbar arrays by 75% (assuming all other settings are the same), leading to a smaller area and lower latency. However, the noise margin drops by $64\times$ as the cell noise margin is inversely proportional to the number of conductance levels (i.e., $2^{Res_{cell}}$).

C. Performance of Existing Training Approaches

We next evaluate the DNN inferencing accuracy of several previously proposed ReRAM hardware-aware training methods with ResNet20 and VGG13 and show the results in Fig. 3. Weights and activations are quantized to 8 bits. We consider two different ReRAM cell resolutions, 2 bits and 8 bits. Models are tested by including the stochastic noise in ReRAM-based hardware. The baseline configuration considers training with no noise. RGN denotes a naïve noise-aware approach, which injects random Gaussian noise into the system during training. The noise standard deviation is related to the maximum absolute weight value, more specifically, $\Delta G_{RGN} = \mathcal{N}(0, 0.01\cdot\max(G))$. PROG considers only the programming noise ΔG_{prog} with $\sigma_{prog} = 0.0658$ [40] for training.

We make the following observations from the results shown in Fig. 3. **1)** The baseline system suffers from stochastic noise, resulting in poor inferencing accuracy. **2)** Previous training methods, such as RGN and PROG, cannot guarantee the mitigation of DNN inferencing accuracy degradation due to stochastic noise. This result is mainly due to the mismatch between Gaussian noise and the actual device stochastic noise, as shown in Fig. 2(a). **3)** Increasing the cell resolution from 2 to 8 bits exacerbates the degradation in DNN inferencing accuracy due to the reduced noise margin of the ReRAM devices.

V. ReSNA: Hardware-Aware Training Approach

In this section, we describe the proposed ReRAM-based stochastic-noise-aware (ReSNA) training method that incorporates stochastic noise to improve DNN inferencing accuracy.

We start with a pre-trained model to initialize noise-aware training. Batch normalization layers are included after each Conv layer [42], [43]. The computation during the training process considers the hardware configurations. In each iteration, a new set of emulated device noise is applied in the stochastic noise injection step. Thus, the loss calculated at the end of the forward pass reflects the influence of stochastic noise. During the backpropagation, gradients of trainable DNN weights are calculated with respect to the loss. We keep a copy of the noise-free weight values and perform the gradient updates on this copy.

The quality of training degrades due to the distortion of the loss function induced by the variation of weight parameters [44]. When the variation is large enough, the gradient update during the

TABLE II: DNN inferencing accuracy comparison under various training configurations with ReSNA for the following setting: ResNet20, $Freq = 500\,\text{MHz}$, $T = 350\,\text{K}$, 8-bit cell resolution, 128×128 crossbars.

Training Configuration	Inferencing Accuracy	
	w/o Voting	w/ Voting
$\text{Conv}_{\text{ideal}} + \text{FC}_{\text{ideal}}$	69.61%	73.69%
$\text{Conv}_{\text{500MHz,350K}} + \text{FC}_{\text{ideal}}$	79.95%	82.61%
$\text{Conv}_{\text{500MHz,350K}} + \text{FC}_{\text{100MHz,300K}}$	84.05%	85.06%
$\text{Conv}_{\text{500MHz,350K}} + \text{FC}_{\text{100MHz,350K}}$	81.65%	83.79%
$\text{Conv}_{\text{500MHz,350K}} + \text{FC}_{\text{500MHz,350K}}$	10%	10%

backpropagation step can be derived from the expected convergence path. The error due to stochastic noise can propagate and accumulate through the forward path, and similarly, the error in the gradient can propagate and accumulate through the backward path. Thus, the convergence of the loss function can be affected by the accumulation of the gradient error.

Moreover, our experiments show that Conv layers are less sensitive to device stochastic noise than FC layers. Let δ_c denote the ReRAM cell's stochastic noise. We assume that one cell is used to represent one weight for simplicity. During the ReRAM-based FC layer computation involving one weight for a total of n times, the accumulation of the cell's stochastic noise can be approximated as $n^2\delta_c$ (assuming a Gaussian distribution). As mentioned above, Conv kernels are typically duplicated to improve the throughput of the ReRAM-based hardware. These copies have identical weights, but the noise can be approximated to have independent statistical distributions. As the computation involving one weight for n times will be distributed to multiple copies, the accumulation of these cells' stochastic noise can be reduced to $n^2\delta_c/k$. The value of k is related to the number of duplicate copies as well as the correlations between the stochastic noise of these devices. Thus, the device stochastic noise affects FC layers more significantly than Conv layers due to the duplication of Conv kernels. To overcome this challenge, we propose two techniques to improve the stability of DNN inferencing accuracy.
Applying smaller noise to FC layers. Since computing the loss function is critical for the backpropagation step and FC layers are more sensitive to the stochastic noise, we propose to lower the noise level on FC layers for improving the stability of inferencing. Table II compares the performance of ReSNA under different noise configurations. We consider the ResNet20 model with an 8-bit weight and activation quantization, along with a crossbar size of 128×128 in this experiment. The temperature is set to 350 K, and the operational frequency is set to 500 MHz.

Table II shows that without including any noise in training ($\text{Conv}_{\text{ideal}} + \text{FC}_{\text{ideal}}$), the DNN inferencing accuracy on this ReRAM design in the presence of stochastic noise is only 69.61%. Including a high-amplitude noise to the entire network ($\text{Conv}_{\text{500MHz,350K}} + \text{FC}_{\text{500MHz,350K}}$) makes the training unstable. Applying the device stochastic noise to Conv layers while appropriately reducing the noise level on FC layers (e.g., $\text{Conv}_{\text{500MHz,350K}} + \text{FC}_{\text{100MHz,300K}}$) helps in maintaining the stability of training and improving the DNN inferencing accuracy.
Majority vote in the classification layer. Alternatively, FC layers can be duplicated and deployed on different crossbar arrays. The stochastic noise of a single weight parameter across these copies is not independent but is less correlated than the variations due to accessing the same device multiple times. We can feed the same input to these duplication layers and take the majority vote (Voting) to determine the predicted output to compensate for the FC layers' impact on the DNN inferencing accuracy. To minimize the area overhead, we apply Voting to only the classification layer (i.e., the last FC layer) with a small number of copies (e.g., 3). The results

in Table II demonstrate that this technique can further improve DNN inferencing accuracy, even under the combination of high-amplitude noise and high-resolution cells.

In summary, ReSNA incorporates stochastic noise and enhances stability, leading to better inferencing accuracy than the baseline and previous work, as shown in Fig. 3.

VI. CF-MESMO: Efficient MOO Algorithm

Evaluating DNN inferencing accuracy and hardware efficiency requires execution of the ReSNA training for each ReRAM design configuration; this step is, however, computationally expensive (e.g., taking over seven hours to execute 100 training epochs for one ReRAM design configuration for ResNet20 with CIFAR10 data). To address this challenge, we propose an efficient information-theoretic MOO algorithm referred to as Continuous-Fidelity Max-value Entropy Search for Multi-objective Optimization (CF-MESMO). Two key innovations here are: first, we formulate continuous-fidelity evaluation of objective functions by varying the number of training epochs of ReSNA. Second, we propose a principled approach to intelligently select the ReRAM configurations and fidelity of ReSNA for evaluation guided by learned statistical models.

A. MOO Formulation with Continuous-Fidelity Evaluations

For each candidate ReRAM design $x \in \mathcal{X}$, we need to execute the ReSNA method to obtain the DNN weights. Subsequently, we evaluate the objective functions $f_1(\mathbf{x})$ (inferencing accuracy), $f_2(\mathbf{x})$ (hardware area), $f_3(\mathbf{x})$ (execution time), $f_4(\mathbf{x})$ (energy consumption). The cost of evaluation of each ReRAM design configuration can be reduced by making an approximation of the objective function(s). We propose to vary the number of training epochs in ReSNA to trade-off computation cost and accuracy of objective function evaluations (i.e., continuous-fidelity evaluation): small training epochs correspond to lower-fidelity evaluation and vice versa. Therefore, we formulate this problem as a continuous-fidelity MOO problem where we have access to an alternative function $g_j(\mathbf{x}, z_j)$ for all $j \in \{1, 2, 3, 4\}$. Function $g_j(\mathbf{x}, z_j)$ can make cheaper approximations of $f_j(\mathbf{x})$ by varying the fidelity variable $z_j \in \mathcal{Z}$. Without loss of generality, let $\mathcal{Z} = [0, 1]$ be the fidelity space. Fidelities for each function vary in the amount of computational resources consumed and the accuracy of evaluation, where $z_j = 0$ and $z_j^* = 1$ refer to the lowest and highest fidelity, respectively. At the highest fidelity z_j^*, $g_j(\mathbf{x}, z_j^*) = f_j(\mathbf{x})$. Let $\mathcal{C}_j(\mathbf{x}, z_j)$ be the cost of evaluating $g_j(\mathbf{x}, z_j)$, i.e., runtime to perform training using ReSNA for the selected number of training epochs. Evaluation of each ReRAM design configuration $\mathbf{x} \in \mathcal{X}$ with fidelity vector $\mathbf{z} = [z_1, z_2, z_3, z_4]$ generates the evaluation vector $\mathbf{y} \equiv [y_1, y_2, y_3, y_4]$, where $y_j = g_j(\mathbf{x}, z_j)$, and the normalized cost of evaluation is $\mathcal{C}(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{4} \left(\mathcal{C}_j(\mathbf{x}, z_j)/\mathcal{C}_j(\mathbf{x}, z_j^*)\right)$. Our goal is to approximate the Pareto set $\mathcal{X}^*$ by minimizing the overall cost of evaluating candidate ReRAM designs.

B. Overview of CF-MESMO

CF-MESMO learns a surrogate model using data obtained from past ReRAM design evaluations and then intelligently selects the next candidate ReRAM design and the fidelity of ReSNA pair for evaluation by trading-off exploration with exploitation to quickly direct the search towards Pareto-optimal solutions. We perform the following steps in each iteration of CF-MESMO as shown in Algorithm 1: **1)** Select the ReRAM design and fidelity of ReSNA for evaluation that maximizes the information gain per unit cost about the optimal Pareto front based on the current surrogate model. **2)** Execute the hardware-aware training approach ReSNA to evaluate objective functions with the selected ReRAM design and fidelity pair. **3)** Employ the new training example in the form of ReRAM design configurations (i.e., input) and four objective function evaluations (i.e., output) to update the surrogate model. After convergence is achieved (i.e., Pareto front solution doesn't change in several consecutive iterations), we compute the Pareto front from the aggregate set of objective evaluations and obtain the ReRAM design configurations and DNN weights corresponding to the Pareto front as the resulting solution.

Algorithm 1 CF-MESMO Algorithm

Input: ReRAM design space $\mathcal{X}$; DNN π; four objective functions f_j and their continuous approximations g_j using ReSNA training; total cost budget $\mathcal{C}_{total}$.

```
1: Initialize GP models $\mathcal{GP}_1, \cdots, \mathcal{GP}_4$ via ReRAM design evaluations $D$
2: While $\mathcal{C}_t \leq \mathcal{C}_{total}$ and not converged do
3:    for each sample $s \in 1, \cdots, S$:
4:       Sample highest-fidelity functions $\tilde{f}_j \sim \mathcal{GP}_j(., z_j^*)$
5:       $\mathcal{F}_s^* \leftarrow$ Solve cheap MOO over $(\tilde{f}_1, \cdots, \tilde{f}_K)$
6:    Select ReRAM design and fidelity pair:
      $(\mathbf{x}_t, \mathbf{z}_t) \leftarrow \arg max_{\mathbf{x} \in \mathcal{X}, \mathbf{z} \in \mathcal{Z}} \ \alpha_t(\mathbf{x}, \mathbf{z}, \mathcal{F}^*)$ Equation (9)
7:    Perform ReSNA training of DNN $\pi$ with ReRAM design and fidelity pair $(\mathbf{x}_t, \mathbf{z}_t)$

8:    Evaluate objectives $f_1, f_2, f_3, f_4$ for trained DNN on ReRAM design $\mathbf{x}_t$
9:    Update the total cost: $\mathcal{C}_t \leftarrow \mathcal{C}_t + \mathcal{C}(\mathbf{x}_t, \mathbf{z}_t)$
10:   Aggregate training data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}_t, \mathbf{y}_t, \mathbf{z}_t)\}$
11:   Update surrogate statistical models $\mathcal{GP}_1, \cdots, \mathcal{GP}_4$
12:   $t \leftarrow t + 1$
13: end
14: return Pareto set and Pareto front of objective functions $f_1(x), \cdots, f_4(x)$
```

Surrogate models for continuous-fidelity. Surrogate models guide the selection of candidate ReRAM designs to quickly uncover high-quality Pareto fronts. Our training data $\mathcal{D}$ for surrogate models after t iterations consists of t training examples of input-output pairs. We employ Gaussian processes (GPs) [45] as our choice of the statistical model due to their superior uncertainty quantification ability. We learn four surrogate statistical models $\mathcal{GP}_1, \cdots, \mathcal{GP}_4$ from $\mathcal{D}$, where each model $\mathcal{GP}_j$ corresponds to the jth function g_j. Continuous-fidelity GPs (CF-GPs) are capable of modeling functions with continuous fidelities within a single model. Hence, we employ CF-GPs to build surrogate statistical models for each function [46]. A CF-GP is a random process defined over the input space and the fidelity space, characterized by a mean function $\mu : \mathcal{X} \times \mathcal{Z} \rightarrow \mathbb{R}$ and a covariance or kernel function $\kappa : (\mathcal{X} \times \mathcal{Z})^2 \rightarrow \mathbb{R}$. We denote the posterior mean and standard deviation of g_j by $\mu_{g_j}(\mathbf{x}, z_j)$ and $\sigma_{g_j}(\mathbf{x}, z_j)$. We denote the posterior mean and standard deviation of the highest fidelity functions $f_j(\mathbf{x}) = g_j(\mathbf{x}, z_j^*)$ by $\mu_{f_j}(\mathbf{x}) = \mu_{g_j}(\mathbf{x}, z_j^*)$ and $\sigma_{f_j}(\mathbf{x}) = \sigma_{g_j}(\mathbf{x}, z_j^*)$, respectively.

C. Selecting ReRAM Design to Evaluate via Information Gain

The effectiveness of CF-MESMO critically depends on the reasoning mechanism to select the candidate ReRAM design and fidelity of ReSNA pair for evaluation in each iteration. Therefore, we propose an information-theoretic approach to perform this selection. *The key idea is to find the ReRAM design and fidelity pair* $\{\mathbf{x}_t, \mathbf{z}_t\}$ *that maximizes the information gain (I) per unit cost about the Pareto front of the highest fidelities* (denoted by $\mathcal{F}^*$), where $\{\mathbf{x}_t, \mathbf{z}_t\}$ represents a candidate ReRAM design configuration $\mathbf{x}_t$ evaluated at fidelities $\mathbf{z}_t$ at iteration t. CF-MESMO performs the joint search over the input space $\mathcal{X}$ and the fidelity space $\mathcal{Z}$:

$$(\mathbf{x}_t, \mathbf{z}_t) \leftarrow \arg\max_{\mathbf{x} \in \mathcal{X}, \mathbf{z} \in \mathcal{Z}} \ \alpha_t(\mathbf{x}, \mathbf{z}), \tag{1}$$

$$\text{where} \quad \alpha_t(\mathbf{x}, \mathbf{z}) = I(\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}, \mathcal{F}^* | \mathcal{D}) / \mathcal{C}(\mathbf{x}, \mathbf{z}). \tag{2}$$

In this joint search, the computation cost $\mathcal{C}(\mathbf{x}, \mathbf{z})$ is considered in Equation (2). The information gain in Equation (2) is the expected reduction in entropy $H(.)$ of the posterior distribution $P(\mathcal{F}^* | \mathcal{D})$ due to the evaluation of the ReRAM design $\mathbf{x}$ at fidelity vector $\mathbf{z}$.

According to the symmetric property, the information gain can be rewritten as follows:

$$I(\{\mathbf{x},\mathbf{y},\mathbf{z}\},\mathcal{F}^*|\mathcal{D}) = H(\mathbf{y}|\mathcal{D},\mathbf{x},\mathbf{z}) - \mathbb{E}_{\mathcal{F}^*}[H(\mathbf{y}|\mathcal{D},\mathbf{x},\mathbf{z},\mathcal{F}^*)]. \tag{3}$$

The first term in Equation (3) is the entropy of a four-dimensional Gaussian distribution that can be computed as follows:

$$H(\mathbf{y}|\mathcal{D},\mathbf{x},\mathbf{z}) = \sum_{j=1}^{4} \ln(\sqrt{2\pi e}\ \sigma_{g_j}(\mathbf{x}, z_j)). \tag{4}$$

The second term in Equation (3) is an expectation over $\mathcal{F}^*$ and can be approximated using Monte-Carlo sampling:

$$\mathbb{E}_{\mathcal{F}^*}[H(\mathbf{y}|\mathcal{D},\mathbf{x},\mathbf{z},\mathcal{F}^*)] \simeq \frac{1}{S}\sum_{s=1}^{S}[H(\mathbf{y}|\mathcal{D},\mathbf{x},\mathbf{z},\mathcal{F}_s^*)], \tag{5}$$

where S denotes the number of samples, and $\mathcal{F}_s^*$ denotes a sample Pareto front achieved over the highest fidelity functions sampled from the surrogate models. To solve Equation (5), we provide solutions to construct Pareto front samples $\mathcal{F}_s^*$ and to compute the entropy of a given Pareto front sample $\mathcal{F}_s^*$.

Computation of Pareto front samples: We sample the highest fidelity functions $\tilde{f}_1, \cdots, \tilde{f}_4$ from the posterior CF-GP models. Then, we solve a cheap MOO problem over the sampled functions with the NSGA-II algorithm [28] and compute the sample Pareto front $\mathcal{F}_s^*$.

Entropy computation for a given Pareto front sample: Let $\mathcal{F}_s^* = \{\mathbf{v}^1, \cdots, \mathbf{v}^l\}$ be the sample Pareto front, where l denotes the size of the Pareto front and each element $\mathbf{v}^i = \{v_1^i, \cdots, v_4^i\}$ is evaluated at the sampled highest-fidelity function. The following inequality holds for each component y_j of $\mathbf{y}$ in the entropy term $H(\mathbf{y}|\mathcal{D},\mathbf{x},\mathbf{z},\mathcal{F}_s^*)$:

$$y_j \le f_s^{j*} \quad \forall j \in \{1, \cdots, 4\}, \tag{6}$$

where $f_s^{j*} = \max\{v_j^1, \cdots v_j^l\}$. Essentially, this inequality means that the j^{th} component of $\mathbf{y}$ is upper-bounded by the maximum of j^{th} components of sample Pareto front $\mathcal{F}_s^*$.

The proof of Equation (6) falls in two cases[1]: **a)** If y_j is evaluated at the highest fidelity (i.e, $z_j = z_j^*$ and $y_j = f_j$), we prove by contradiction. Suppose there exists some component f_j of $\mathbf{f}$ such that $f_j > f_s^{j*}$. However, by definition, since no point dominates $\mathbf{f}$ in the jth dimension, $\mathbf{f}$ is a non-dominated point. This results in $\mathbf{f} \in \mathcal{F}_s^*$, which is a contradiction. Thus, Equation (6) holds. **b)** If y_j is evaluated at one of the lower fidelities (i.e, $z_j \neq z_j^*$), we refer to the assumption that the value of an objective evaluated at lower fidelity is smaller than that evaluated at higher fidelity, i.e., $y_j \le f_j \le f_s^{j*}$. This assumption is true in our problem setting, where the DNN inferencing accuracy improves with more training epochs of ReSNA.

Following Equation (6) and the independence of CF-GP models, we further decompose the entropy of a set of independent variables according to the entropy measure property [47]:

$$H(\mathbf{y}|\mathcal{D},\mathbf{x},\mathbf{z},\mathcal{F}_s^*) \simeq \sum_{j=1}^{4} H(y_j|\mathcal{D},\mathbf{x},z_j,f_s^{j*}). \tag{7}$$

Equation (7) requires the entropy computation of $p(y_j|\mathcal{D},\mathbf{x},z_j,f_s^{j*})$. This conditional distribution can be expressed as $H(y_j|\mathcal{D},\mathbf{x},z_j,y_j \le f_s^{j*})$. As Equation (6) states that $y_j \le f_s^{j*}$ holds under all fidelities, the entropy of $p(y_j|\mathcal{D},\mathbf{x},z_j,f_s^{j*})$ can be approximated by the entropy of a truncated Gaussian distribution as:

$$\begin{aligned} H(y_j|\mathcal{D},\mathbf{x},z_j,y_j \le f_s^{j*}) =& \ln(\sqrt{2\pi e}\ \sigma_{g_j}) + \ln\Phi(\gamma_s^{(g_j)}) \\ &- \frac{\gamma_s^{(g_j)}\phi(\gamma_s^{(g_j)})}{2\Phi(\gamma_s^{(g_j)})}, \end{aligned} \tag{8}$$

[1]For ease of notation, we drop the dependency on $\mathbf{x}$ and $\mathbf{z}$. We use f_j to denote $f_j(\mathbf{x}) = g_j(\mathbf{x}, z_j^*)$ the evaluation of the highest fidelity z_j^* and y_j to denote $g_j(\mathbf{x}, z_j)$ the evaluation of g_j at a lower fidelity $z_j \neq z_j^*$.

where $\gamma_s^{(g_j)} = \frac{f_s^{j*} - \mu_{g_j}}{\sigma_{g_j}}$. Functions ϕ and Φ are the probability density and cumulative distribution function of the standard normal distribution, respectively. From Equations (4), (5), and (8), we get the expression as shown below:

$$\alpha_t(\mathbf{x},\mathbf{z},\mathcal{F}^*) = \frac{1}{\mathcal{C}(\mathbf{x},\mathbf{z})S}\sum_{j=1}^{4}\sum_{s=1}^{S}\frac{\gamma_s^{(g_j)}\phi(\gamma_s^{(g_j)})}{2\Phi(\gamma_s^{(g_j)})} - \ln(\Phi(\gamma_s^{(g_j)})). \tag{9}$$

Therefore, in Algorithm 1, we select the next ReRAM design and the fidelity of ReSNA pair that maximizes the information gain per unit cost about the optimal Pareto front based on Equation (9).

VII. Experiments and Results

In this section, we first explain the details of the experimental setup. Next, we evaluate the effectiveness of ReSNA in improving the inferencing accuracy. Finally, we show that CF-MESMO can achieve high-quality Pareto fronts for DNN inferencing on ReRAM crossbars and analyze the Pareto sets for different DNN models.

A. Experimental Setup

We evaluate ReSNA with five different DNNs—ResNet20, ResNet32, ResNet44 [48], VGG11, and VGG13 [49] on the CIFAR-10 dataset [50]. The CIFAR-10 dataset contains $50,000$ training images and $10,000$ testing images, which belong to 10 classes. Furthermore, to validate the scalability of our method, we also evaluate the performance of ResNet18 [48] using the CIFAR-100 dataset [50]. The number of training and testing images in CIFAR-100 is the same as in CIFAR-10, but these images belong to 100 classes. The image size is $28 \times 28 \times 3$, and the training and testing batch size is 64. Table III(a) summarizes deep neural network configurations, including the numbers of channels in Conv layers, the inferencing accuracy with unquantized weights and activations, and the inferencing accuracy for 8-bit weights and activations. Note that testing on diverse DNNs is more important to test the effectiveness of our approach. Hence, due to space constraints, we provide results on limited datasets noting that our methodology and findings are general.

TABLE III: Experiments setup details.

(a) Network configurations.

Network	# of channels in Conv layers	Unquantized Accuracy	Quantized Accuracy
ResNet20	16, [16,16]×3, [32, 32]×3, [64, 64]×3	91.65%	89.61%
ResNet32	16, [16,16]×5, [32, 32]×5, [64, 64]×5	92.81%	90.06%
ResNet44	16, [16,16]×7, [32, 32]×7, [64, 64]×7	93.24%	91.54%
VGG11	64, 128, 256, 256, 512, 512, 512, 512	92.18%	88.07%
VGG13	64, 64, 128, 128, 256, 256, 512, 512, 512, 512	93.64%	91.14%
ResNet18 for CIFAR-100	64, [64, 64]×2, [128,128]×2, [256,256]×2, [512,512]×2	74.57%	71.92%

(b) ReRAM parameters in ReSNA.

Parameter	Value
Bit_{quan}	8 bit
R_{on}, R_{off}	3.03 kΩ, 3.03 MΩ
Res_{DAC}, Res_{ADC}	8 bits, 8 bits
V_r	1.65V
σ_{prog}	0.0658

(c) Design space configurations.

Parameter	Candidate values
Res_{cell}	8/4/3/2/1-bit
$Freq$	10 MHz-1000 MHz
T	300 K-400 K
$Xbar_{size}$	32×32, 64×64, 128×128

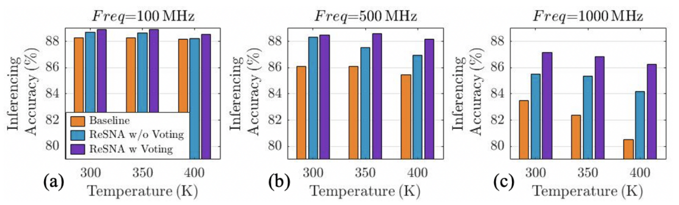

Fig. 4: ReRAM inferencing accuracy under various temperature and frequency settings. 8-bit cell resolution, 64×64 crossbars. ResNet20 on the CIFAR-10 dataset.

We implement the ReSNA method on ReRAM crossbars with stochastic noise using the PytorX simulator [27]. ReSNA uses stochastic gradient descent with a learning rate of 0.001 and a momentum of 0.9. The maximum number of training epochs is 100. For ResNet18, we decay the learning rate by 0.2 for every 20 epochs. Each inferencing test consists of 10 independent runs with stochastic noise, and we report the average inferencing accuracy. All the training and inferencing are conducted on NVIDIA Titan RTX GPU with a memory of 24 GB and a memory bandwidth of 672 GB/s.

Table III(b) summarizes the ReRAM device parameters. The cell resolution, operational frequency, temperature, and crossbar size are inputs to the MOO framework. Table III(c) shows the allowable input configurations. We use NeruoSim [51] along with the 32 nm technology node parameters to evaluate the hardware area, execution time, and energy consumption. The ReRAM crossbar and peripheral configurations are adopted from [52]. For ReSNA with Voting, we duplicate the kernels of the classification layer. For the MOO problem, we run CF-MESMO for a maximum of 100 iterations with 10 available fidelity selections. The baselines for evaluating the efficiency of CF-MESMO are the random search and NSGA-II. We utilize the NSGA-II implementation from *Platypus* python library.

B. ReSNA Results

Inferencing accuracy. We show representative results for ReSNA inferencing accuracy with multiple temperature and frequency settings with respect to the baseline. Recall that the baseline configuration considers training with no noise and performs inferencing in the presence of stochastic noise. Fig. 4(a)-(c) show the inferencing accuracy for ResNet20 with 8-bit cell resolution and 64×64 crossbars. On average, ReSNA without Voting outperforms the baseline by 1.62% over all the test conditions. ReSNA with Voting increases the overall inferencing accuracy by 2.57%. Considering the extreme design configuration, i.e., at 1000 MHz and 400 K, ReSNA with Voting outperforms the baseline by 5.47%. Note that this extreme case assumes that in the future, ReRAM-based hardware will run at this high frequency. We also validate the performance of the ReSNA method with the ResNet18 on the CIFAR-100. With a frequency of 500 MHz and a temperature of 350 K, ReSNA achieves 70.80% inferencing accuracy compared with the baseline inferencing accuracy of 69.37%. With a frequency of 1000 MHz and a temperature of 350 K, ReSNA achieves 68.23% inferencing accuracy compared with the baseline inferencing accuracy of 64.45%.

In summary, ReSNA can achieve considerable inferencing accuracy improvement under stochastic noise with different ReRAM design configurations, DNNs, and datasets.

Design trade-offs considering different objectives. We further explore the design trade-offs considering the inferencing accuracy, hardware area, execution time, and energy consumption. As we have explored the effects of frequency and temperature on inferencing accuracy in the previous analysis, we focus on cell resolution and crossbar size in this discussion. Fig. 5 shows the impact of cell resolution and the crossbar size on ResNet20 at 500 MHz and 350 K.

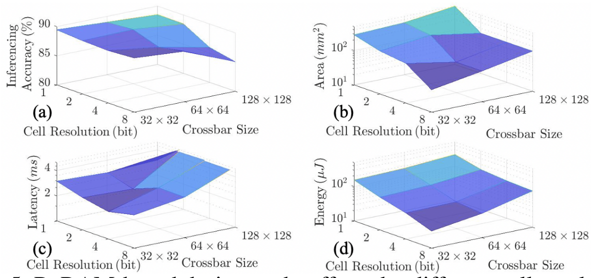

Fig. 5: ReRAM-based design trade-offs under different cell resolution and crossbar size settings. ResNet20, $Freq = 500$ MHz, $T = 350$ K. (a) inferencing accuracy, (b) hardware area, (c) latency, (d) energy consumption.

Fig. 5(a) shows that inferencing accuracy using the ReSNA method. When other settings remain the same, the inferencing accuracy steadily increases as the cell resolution reduces due to the improved noise margin. As discussed in Section III, the ReRAM crossbar array outputs are accumulated across the columns, and hence the crossbar size affects the noise accumulation. Therefore, a large crossbar with high cell resolution is not an optimal design choice with this setting from the inferencing accuracy perspective.

From the area perspective, the 32×32 crossbar with 8-bit cell resolution is the best, while 64×64 crossbar has the least area when the cell resolution is 4-bit. The area evaluation reveals a high correlation between the cell resolution and crossbar size. Fig. 5(c) shows that the minimum latency is achieved by the 64×64 crossbar with the cell resolution of 2-bit, though this particular configuration incurs a relatively larger area overhead than the configuration with 4-bit cell resolution. Fig. 5(d) shows that the energy consumption with large cell resolution and small crossbar size is relatively modest.

C. Results on Using CF-MESMO to Optimize ReRAM Crossbars

Fig. 5 also indicates that different objectives have different optimal design configurations, and a global optimal design configuration is not achievable. Note that the operational frequency and temperature considered above are discrete data points selected for initial performance evaluation. However, the temperature can take any value from 300 K to 400 K. Assuming the temperature resolution to be 0.1 K, we can get 1,000 data points. Similarly, by considering other inputs, we can estimate the number of all design configurations to be 1.485×10^7. While any MOO framework can search over this enormous space to achieve the Pareto front to establish the suitable design trade-offs, the computation cost associated with this search is prohibitively high (e.g., it takes nearly 30 GPU days to run the ReSNA training on the PytorX simulator [27] for 100 configurations). In contrast, the proposed CF-MESMO framework does not traverse through all the data points but can achieve a high-quality Pareto front with significantly reduced computation cost. We use the hypervolume, which measures the volume between the Pareto front and a reference point [53], to indicate the quality of the Pareto front.

CF-MESMO vs. NSGA-II and random search. Fig. 6(a) illustrates the hypervolume result of CF-MESMO compared with NSGA-II and random search under the same computation cost. The unit cost is defined as the runtime for executing 10 training epochs of the ReSNA method (e.g., 42.5 minutes based on our setting for ResNet20). We observe that **1)** CF-MESMO can achieve a higher-quality Pareto optimal set for the same total computation cost for ReRAM design evaluation; **2)** CF-MESMO can produce higher quality Pareto front using significantly lower cost compared to NSGA-II and random search; **3)** CF-MESMO achieves 90.91% and 91.21% reduction in computation cost to reach the same quality Pareto front as NSGA-II and random search, respectively.

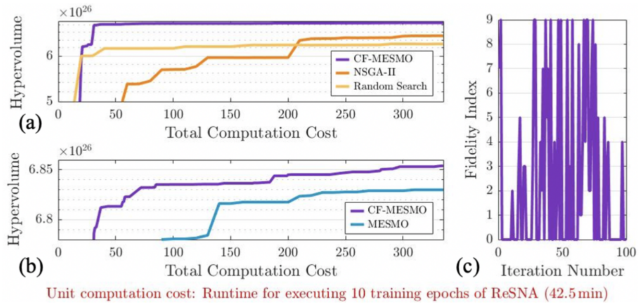

Fig. 6: CF-MESMO framework results for ResNet20: (a) CF-MESMO hypervolume result compared with NSGA-II and random search, (b) CF-MESMO and MESMO hypervolume results, (c) fidelity index over the CF-MESMO iterations.

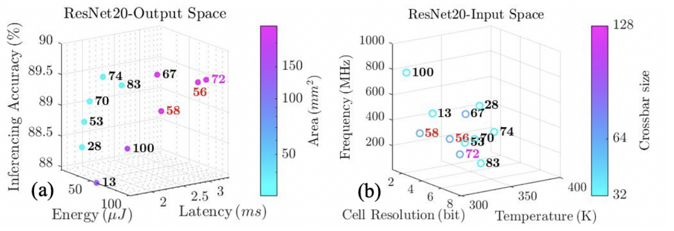

Fig. 7: CF-MESMO framework Pareto front and Pareto set for ResNet20 in a 3-dimensional plot with colorbar, the iteration number is labeled next to each data point: (a) output space, (b) input space.

Continuous fidelity vs. single maximum fidelity. We utilize a continuous fidelity setting in CF-MESMO. As a comparison, we run our optimization method with the single maximum fidelity setting (100 training epochs of ReSNA to evaluate each ReRAM design), denoted as MESMO. Fig. 6(b) compares the hypervolume of these two methods considering the computational cost. The continuous-fidelity setting in CF-MESMO can guarantee higher quality Pareto front with lower computation cost when compared to the single maximum fidelity algorithm MESMO. Specifically, CF-MESMO lowers the computation cost by 78.18% to reach the same quality Pareto front as MESMO. Note that both CF-MESMO and MESMO can outperform NSGA-II and random search, as we observe from Fig. 6(a)-(b). These results validate the effectiveness of the proposed CF-MESMO algorithm for ReRAM based MOO optimization.

Fig. 6(c) shows the fidelity index (small index means ReRAM design evaluation using ReSNA with a small number of training iterations) selection over iterations in CF-MESMO. The optimization starts with a large fidelity index, e.g., 7 and 9 at the 1^{st} and 2^{nd} iterations in the initialization. During the 3^{rd} to 25^{th} iterations, a small fidelity index is preferred to get a fast approximation of the Pareto front by identifying the promising areas of the ReRAM design space. After that, a larger fidelity index is selected to approach the optimal Pareto front by evaluating candidates from this set of promising ReRAM designs. By using this continuous-fidelity setting, we can exclude the non-promising configurations in an early stage. As shown in Equation (2), high fidelity is only utilized when the predicted information gain per unit cost is large.

Optimization results for different DNNs. We use the proposed CF-MESMO framework to achieve robust DNN inferencing with an efficient ReRAM-based hardware platform. For the ResNet20 network, CF-MESMO obtains 11 Pareto optimal designs over 100 iterations. Fig. 7(a) represents these designs in the output space, including the inferencing accuracy, latency, energy consumption, and area overhead. Each data point is labeled with the corresponding iteration number. Although all the 11 design instances appear at the Pareto front, some of them can be excluded due to efficiency

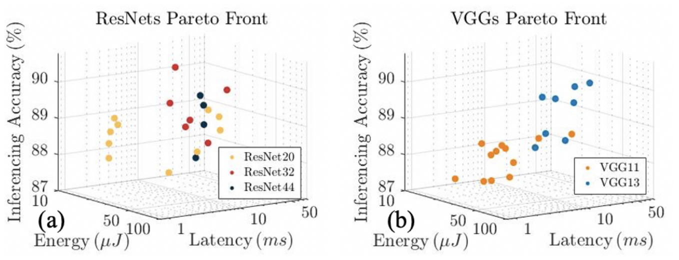

Fig. 8: CF-MESMO framework Pareto front result for ResNets and VGGs in a 3-dimensional plot: (a) ResNets, (b) VGGs.

constraints. For instance, designs '56' and '58' consume 113.3% and 105.2% more energy compared to the average of the other designs, while design '72' requires 57.4% more execution time compared to the average of the other designs. We mark the high-energy design points with red and the high-latency design point with magenta in Fig. 7. According to the input space shown in Fig. 7(b), a relatively low temperature (300K-350K) and small and medium crossbar sizes (32×32 and 64×64) are recommended for ResNet20.

Fig. 8 shows the Pareto fronts for ResNets and VGGs. Note that the area evaluation dimension is not included in the plot for ease of illustration, and latency is normalized under the same area constraint. It should be noted that ResNet32 and VGG13 achieve the best inferencing accuracy for each network class. Comparing Fig. 8(a) with Fig. 8(b), we see that the Pareto fronts for ResNet and VGG have different distributions, while there is a large overlap among the various clusters within the same network class. e.g., the three clusters in Fig. 8(a). These results imply that the network structure is not the only factor to determine hardware efficiency. In designing ReRAM-based accelerators, we should first set the expected inferencing accuracy and hardware efficiency target and then choose the network using the Pareto front.

Based on the Pareto set results from all the evaluations, we make the following observations: **1)** Designs with high temperature can appear in the Pareto front, but the number of those cases is low. **2)** For small DNN models, the channel number is relatively small. Large-sized crossbar results in a low utilization rate and thus is not an optimal choice. As the model size increases, a large-sized crossbar becomes a preferred choice. **3)** As ReSNA improves the inferencing accuracy, high ReRAM cell resolution and high frequency become the preferred candidates for robust DNN inferencing.

VIII. Conclusions

We have presented a ReRAM-based accelerator design and optimization framework to achieve robust DNN inferencing in the presence of stochastic noise. The efficiency of this framework depends on uncovering the Pareto-optimal ReRAM designs to establish a suitable trade-off considering inferencing accuracy, area overhead, execution time, and energy consumption. We have solved this challenging multi-objective optimization (MOO) problem by introducing a Continuous-Fidelity Max-value Entropy Search-based MOO framework, called CF-MESMO. CF-MESMO is aided by a hardware-aware training method to handle stochastic noise, called ReSNA. The CF-MESMO framework provides a high-quality Pareto front for robust DNN inferencing on hardware-efficient ReRAM crossbars with stochastic noise. On average, ReSNA achieves 2.57% inferencing accuracy improvement for ResNet20 on the CIFAR-10 dataset with respect to the baseline configuration. Moreover, the CF-MESMO framework achieves 90.91% reduction in computation cost compared with the popular MOO framework NSGA-II to reach the same quality Pareto front as NSGA-II.

References

[1] M. Hu *et al.*, “Hardware realization of BSB recall function using memristor crossbar arrays,” in *Design Automation Conference*, 2012.

[2] Y. Chen *et al.*, “A survey of accelerator architectures for deep neural networks,” *Engineering*, 2020.

[3] P. Chi *et al.*, “PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory,” in *International Symposium on Computer Architecture*, 2016.

[4] A. Shafiee *et al.*, “ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” in *International Symposium on Computer Architecture*, 2016.

[5] L. Song *et al.*, “PipeLayer: A pipelined ReRAM-based accelerator for deep learning,” in *IEEE International Symposium on High-Performance Computer Architecture*, 2017.

[6] Y. Long *et al.*, “ReRAM-based processing-in-memory architecture for recurrent neural network acceleration,” *IEEE Transactions on Very Large Scale Integration System*, 2018.

[7] Z. Fan *et al.*, “RED: A ReRAM-based deconvolution accelerator,” in *Design, Automation, and Test in Europe*, 2019.

[8] X. Yang *et al.*, “ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration,” in *International Conference on Computer-Aided Design*, 2020.

[9] N. K. Jayakodi *et al.*, “Trading-off accuracy and energy of deep inference on embedded systems: A co-design approach,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[10] N. K. Jayakodi *et al.*, “Design and optimization of energy-accuracy tradeoff networks for mobile platforms via pretrained deep models,” *ACM Transactions on Embedded Computing Systems*, 2020.

[11] T. Belkhouja *et al.*, “Analyzing deep learning for time-series data through adversarial lens in mobile and IoT applications,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[12] C.-Y. Chen *et al.*, “RRAM defect modeling and failure analysis based on march test and a novel squeeze-search scheme,” *IEEE Transactions on Computers*, 2014.

[13] S. Schechter *et al.*, “Use ECP, not ECC, for hard failures in resistive memories,” in *International Symposium on Computer Architecture*, 2010.

[14] Y. Y. Chen *et al.*, “Postcycling LRS retention analysis in HfO2/Hf RRAM IT1R device,” *IEEE Electron Device Letters*, 2013.

[15] C. Xu *et al.*, “Impact of cell failure on reliable cross-point resistive memory design,” *ACM Transactions on Design Automation of Electronic Systems*, 2015.

[16] C. Liu *et al.*, “Rescuing memristor-based neuromorphic design with high defects,” in *Design Automation Conference*, 2017.

[17] L. Xia *et al.*, “Stuck-at fault tolerance in RRAM computing systems,” *IEEE Journal of Emerging and Selected Topics in Power Electronics*, 2017.

[18] L. Chen *et al.*, “Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar,” in *Design, Automation, and Test in Europe*, 2017.

[19] B. Li *et al.*, “Build reliable and efficient neuromorphic design with memristor technology,” in *Asia and South Pacific Design Automation Conference*, 2019.

[20] I. Chakraborty *et al.*, “GENIEx: A generalized approach to emulating non-ideality in memristive xbars using neural networks,” in *Design Automation Conference*, 2020.

[21] D. Soudry *et al.*, “Memristor-based multilayer neural networks with online gradient descent training,” *IEEE Transactions on Neural Networks and Learning Systems*, 2015.

[22] B. Feinberg *et al.*, “Making memristive neural network accelerators reliable,” in *IEEE International Symposium on High-Performance Computer Architecture*, 2018.

[23] D. Ielmini *et al.*, “Resistance-dependent amplitude of random telegraph-signal noise in resistive switching memories,” *Applied Physics Letters*, 2010.

[24] S. R. Lee *et al.*, “Multi-level switching of triple-layered TaOx RRAM with excellent reliability for storage class memory,” in *Symposium on VLSI Technology*, 2012.

[25] Y. Long *et al.*, “Design of reliable DNN accelerator with un-reliable ReRAM,” in *Design, Automation, and Test in Europe*, 2019.

[26] V. Joshi *et al.*, “Accurate deep neural network inference using computational phase-change memory,” *Nature Communications*, 2020.

[27] Z. He *et al.*, “Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping,” in *Design Automation Conference*, 2019.

[28] K. Deb *et al.*, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, 2002.

[29] S. Belakaria *et al.*, “Max-value entropy search for multi-objective Bayesian optimization,” in *Conference on Neural Information Processing Systems*, 2019.

[30] B. Yan *et al.*, “A closed-loop design to enhance weight stability of memristor based neural network chips,” in *International Conference on Computer-Aided Design*, 2017.

[31] S. Bandyopadhyay *et al.*, “A simulated annealing-based multiobjective optimization algorithm: AMOSA,” *IEEE Transactions on Evolutionary Computation*, 2008.

[32] B. Shahriari *et al.*, “Taking the human out of the loop: A review of Bayesian optimization,” *Proceedings of the IEEE*, 2016.

[33] S. Belakaria *et al.*, “Uncertainty-aware search framework for multi-objective Bayesian optimization,” in *AAAI Conference on Artificial Intelligence*, 2020.

[34] Z. Zhou *et al.*, “Design of multi-output switched-capacitor voltage regulator via machine learning,” in *Design, Automation and Test in Europe*, 2020.

[35] S. Belakaria *et al.*, “Machine learning enabled fast multi-objective optimization for electrified aviation power system design,” in *IEEE Energy Conversion Congress and Exposition*, 2020.

[36] S. Han *et al.*, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *arXiv preprint arXiv:1510.00149*, 2015.

[37] S. Zhou *et al.*, “DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients,” *arXiv preprint arXiv:1606.06160*, 2016.

[38] H. Yang *et al.*, “BSQ: Exploring bit-level sparsity for mixed-precision neural network quantization,” *International Conference on Learning Representations*, 2021.

[39] J. B. Johnson, “Thermal agitation of electricity in conductors,” *Physical Review*, 1928.

[40] Y.-H. Lin *et al.*, “Performance impacts of analog ReRAM non-ideality on neuromorphic computing,” *IEEE Transactions on Electron Devices*, 2019.

[41] S. Seo and J. Kim, “Efficient weights quantization of convolutional neural networks using kernel density estimation based non-uniform quantizer,” *Applied Science*, 2019.

[42] B. K. Joardar *et al.*, “AccuReD: High accuracy training of CNNs on ReRAM/GPU heterogeneous 3D architecture,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[43] B. K. Joardar *et al.*, “High-throughput training of deep CNNs on ReRAM-based heterogeneous architectures via optimized normalization layers,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[44] I. Chakraborty *et al.*, “Technology aware training in memristive neuromorphic systems for nonideal synaptic crossbars,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2018.

[45] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT Press, 2006.

[46] K. Kandasamy *et al.*, “Multi-fidelity Bayesian optimisation with continuous approximations,” in *International Conference on Machine Learning*, 2017.

[47] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley and Sons, 2012.

[48] K. He *et al.*, “Deep residual learning for image recognition,” in *International Conference on Computer Vision*, 2016.

[49] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[50] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” 2009.

[51] X. Peng *et al.*, “DNN+ NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies,” in *IEEE International Electron Devices Meeting*, 2019.

[52] X. Peng *et al.*, “Optimizing weight mapping and data flow for convolutional neural networks on processing-in-memory architectures,” *IEEE Transactions on Circuits and Systems I*, 2019.

[53] E. Zitzler and L. Thiele, “Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach,” *IEEE Transactions on Evolutionary Computation*, 1999.