MOTPE/D: Hardware and Algorithm Co-design for Reconfigurable Neuromorphic Processor

Yuan Li[1], Renzhi Chen[2†], Zhijie Yang[2], Xun Xiao[1], Jingyue Zhao[2],
Zhenhua Zhu[3], Huadong Dai[2], Yuhua Tang[1], Weixia Xu[1], Li Luo[1], Lei Wang[2†]
[1]*National University of Defense Technology, Chang Sha, Hu Nan, China*
[2]*Academy of Military Sciences, Beijing, China,* [3]*Tsinghua University, Beijing, China*
{liyuan22, li_luo, leiwang}@nudt.edu.cn, chenrenzhi1989@gmail.com, zhuzhenhua@mail.tsinghua.edu.cn

2024 IEEE 42nd International Conference on Computer Design (ICCD) | 979-8-3503-8040-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICCD63220.2024.00086

***Abstract*—Recent advances in hardware/algorithm co-design for spiking neural networks have demonstrated its potential for jointly optimizing algorithmic performance while minimizing hardware overhead. However, the gigantic mixed-variable hardware/algorithm co-design space and time-consuming hardware verification still pose an intractable challenge for solutions exploration. To tackle these problems, 1) we propose a generic three-phase hardware/algorithm co-design framework. In this framework, 2) we target a reconfigurable neuromorphic processor, and parameterize the hardware and network architecture in a unified design space. 3) We propose a generic analytical model to estimate the parameter size and power consumption, which can support fast candidate evaluation during the exploration. 4) We extend vanilla TPE (a single-objective optimization algorithm) to MOTPE/D, a generic Multi-objective optimization (MOO) algorithm, by introducing a decomposition strategy.**

***Index Terms*—spiking neural network, hardware/algorithm co-design, neuromorphic processor, multi-objective optimization**

I. Introduction

As a bio-plausible alternative to traditional deep learning methods, Spiking Neural Network (SNN) has exhibited excellent computational efficiency due to the sparse and event-driven information processing mechanism, which makes it well-suited for resource-constrained scenarios like edge-computing devices or embedded systems [1]. However, the sparse binary activations, low-power, and low-latency characteristics of SNN are usually not fully realized, since commercial CPU/GPU-based edge platforms are not optimized for deploying SNNs [2].

There are two main lines of research aiming to improve the performance of hardware deployed SNNs. From the hardware level, enormous neuromorphic processors [3] or highly-specialized SNN accelerators [2] have been proposed to support the execution of large-scale SNNs. Nevertheless, these general or highly customized hardware architectures can hardly maintain satisfying performance when switching to different network architectures. From the algorithm level, a considerable amount of hardware-aware SNN Neural Network Search (NAS) works have been emerging [4]. They aim to optimize the SNN network architecture for a target neuromorphic processor architecture. But the fixed hardware platform configurations always shift the optimal SNN architecture distributions and result in sub-optimal solutions [5]. To that end, it is necessary to co-design the neuromorphic processor and network architecture to improve the performance of SNN models running on neuromorphic hardware platforms.

Manual co-design is unrealistic due to the gigantic SNN network and processor architecture co-design space. A growing body of automatical hardware/algorithm co-design works have been proposed, and they can be mainly classified into three categories: differentiable NAS [6], reinforcement learning (RL) [7], and evolution-based ones [8]. Recent surveys argued that differentiable NAS and evolution-based methods may fall into the local optimal regions [9]. RL-based approaches usually need substantial training data and computational resources to learn effective strategies [7]. To go further, for each candidate model/hardware pair, the time-consuming training of SNN model and laborious hardware performance verification pose another intractable challenge.

In light of the above analysis, we generalize our ideas of SNN algorithm and reconfigurable neuromorphic processor co-design into a three-phase framework. Our main contributions to this work can be summarized as follows:

- We propose a high-level abstract of a reconfigurable neuromorphic processor, which has a tunable set of design parameters. We parameterize the SNN model and neuromorphic processor architecture search in a unified design space.
- Generic parameter size and power consumption estimation models of SNN running on neuromorphic processors are proposed for the first time, which can support fast candidate selection.
- Considering the gigantic co-design space mixed with continuous and discrete variables, we introduce the Tchebycheff decomposition to vanilla Tree-structured Parzen Estimator (TPE) and extend it to MOTPE/D searching strategy - **M**ulti-**O**bjective **T**ree-structured **P**arzen **E**stimator base on **D**ecomposition strategy.

II. Background & Related Works

A. Liquid State Machine

Liquid State Machine (LSM) is typical SNN model including both feed-forward connections and recurrent connections.

†Renzhi Chen and Lei Wang are the corresponding authors.

2576-6996/24/$31.00 ©2024 IEEE
DOI 10.1109/ICCD63220.2024.00086

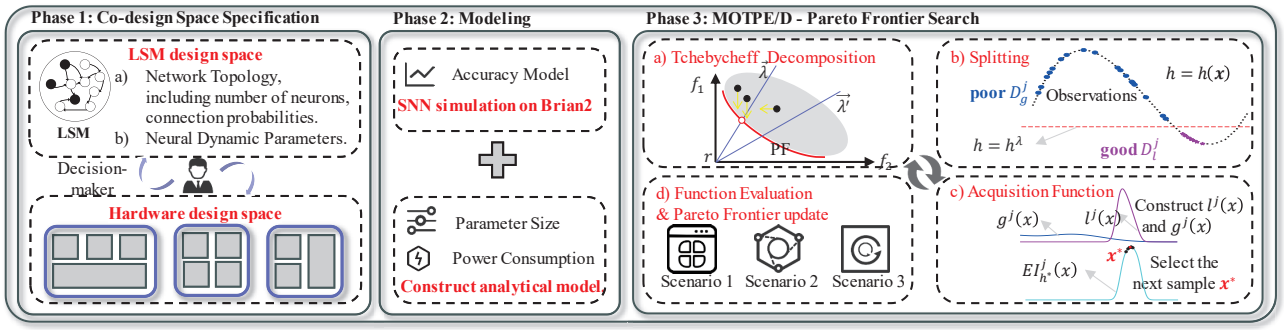


Fig. 1. Overview of the proposed hardware/algorithm co-design framework.

LSM comprises the input, the liquid, and the read-out layer. Leaky integrate and fire (LIF) neuron model is adopted to simulate all neurons' behavior, which can be mathematically expressed as:

$$\tau_m \frac{du}{dt} = -[u - u_{rest}] + RI(t) \tag{1}$$

where τ_m is the membrane constant, and u_{rest} is the resting potential. The input layer encodes and transports input data to the liquid. The liquid layer, a pool of recurrently connected excitatory and inhibitory neurons, transforms input spike trains into state vectors.

B. Hardware/Algorithm Co-design

Shikhar et al. [10] proposed CODEBench, which leveraged Bayesian Optimization using second-order gradients and heteroscedastic surrogate model for network/accelerator co-design. Ruixing et al. [1] proposed STELLAR, which is an algorithm/hardware co-design framework through exploiting rich spatiotemporal dynamics of the SNN for high energy efficiency and low latency. Hongxiang et al. [8] adopted evolutionary algorithms to do hardware and algorithm co-design for reconfigurable convolutional neural network accelerators. Lei et al. [7] adopted a novel reinforcement learning-based Recurrent Neural Network (RNN) controller to simultaneously identify multiple DNN architectures and the associated heterogeneous ASIC accelerator design.

III. Hardware/Algorithm Co-design Framework

A. Problem Formulation

In different contexts, co-designers usually need to make different trade-offs between many conflict optimization objectives. Therefore, the design goal is to provide decision-makers with a set of Pareto optimal solutions. It can be defined as the following Multi-objective Optimization Problem (MOP).

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots f_m(\mathbf{x})) \\ &\text{subject to} \quad \mathbf{x} \in X \end{aligned} \tag{2}$$

where $X = X_1 \times ... \times X_k$ is a k-dimensional design (decision) space, and $f_i : X \to \mathbb{R}(i = 1, ...m)$ is an objective function.

To address the problem given in equation 2, we propose a novel hardware/algorithm co-design framework, as illustrated in Figure 1. We generalized the entire process into three phases: 1) Co-design space specification, 2) Modeling, 3) MOTPE/D - Pareto frontier search.

TABLE I
Design Space Explanations

Parameters	Seach space	Type	Influence
n	$[5^3, 10^3]$	int	chip area, static power
C_{in}	$[0.1, 1.5]$	float	chip area, on-chip storage, static power, dynamic power
C_{ee}	$[0.1, 1.5]$	float	
C_{ei}	$[0.1, 1.5]$	float	
C_{ie}	$[0.1, 1.5]$	float	
C_{ii}	$[0.1, 1.5]$	float	
θ_{e}	$[-55, -47]$	float	dynamic power
θ_{i}	$[-43, -37]$	float	
h1	$[32, 512]$	int	no consideration
h2	$[32, 512]$	int	

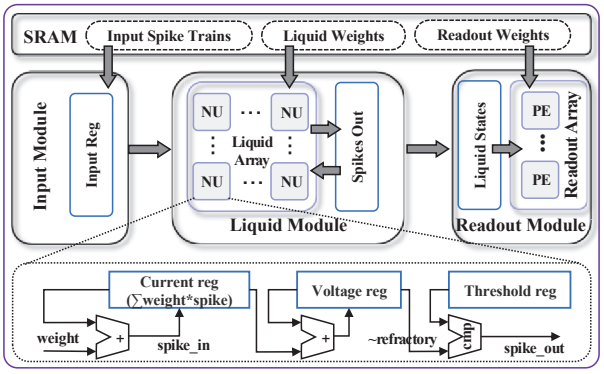


Fig. 2. High-level overview of an abstract neuromorphic processor.

B. Co-Design Space Specification

The performance of LSM highly depends on parameters listed in TABLE I. n is the number of liquid neurons, C_{ei} represents the connection coefficient between the excitatory and inhibitory neurons, and C_{ee}, C_{ie}, C_{ii} follows the same rule. θ_e and θ_i represent the firing threshold of excitatory and inhibitory neurons, respectively. h1 and h2 represent the number of neurons in the first and second hidden layer of the readout MLP.

How can these LSM design parameters directly be related to the neuromorphic processor design? To answer this question, a simplified and generalized single-core neuromorphic architecture is given in Fig 2. There are four major components, namely the global static random access memory (SRAM), the input module, the liquid module, and the readout module. At each time step, the input module fetches the encoded input spike trains from SRAM, and then sends them to the liquid module. Each neuron unit (NU) in the liquid module receives

spikes from the input module and liquid module, accumulates membrane potential, generates spikes, and refreshes its state.

Therefore, the number of liquid neurons has a main effect on the chip area and static power. C_{in}, C_{ee}, C_{ei}, C_{ie}, and C_{ii} regulate the connection probabilities. A higher C value means more connections in the model, thus the connection matrix stored in the SRAM is bigger, and spikes are transmitted more between NUs, which will bring about higher chip area, on-chip storage, static power, and dynamic power. θ_e and θ_i mainly influence the spiking behaviour of each NU.

C. Modeling

We proposed three performance metrics: acc, $params$, and $power$ to respectively evaluate the accuracy, parameter size, and power consumption of the candidate solutions.

1) Acc model: Accuracy indicates the ratio of correctly classified samples to the total number of samples.

2) $Params$ Model: For each synaptic connection, the index of the source neuron, destination neuron, and the synaptic weight between them need to be stored. The $parameter\ size$ can be formulated as Equation 3:

$$parameter\ size = 3 \times (w_{\text{in-e}} + w_{\text{ee}} + w_{\text{ei}} + w_{\text{ie}} + w_{\text{ii}}) \quad (3)$$

Where $w_{\text{in-e}}, w_{\text{ee}}, w_{\text{ei}}, w_{\text{ie}}, w_{\text{ii}}$ represent all the synaptic weights between these two kinds of neurons (in: input neuron, e: excitatory neuron, i: inhibitory neuron).

3) $Power$ model: The total power consumption of this abstract neuromorphic processor architecture can be formulated as Equation 4:

$$f_{\text{power}} = P_{\text{leak}} + P_{\text{idle}} \times F_{\text{clk}} + E_{\text{SO}} \times r_{\text{SO}} \quad (4)$$

where P_{leak} is leakage power without clock activity. $P_{\text{leak}} + P_{\text{idle}} \times F_{\text{clk}}$ is the static power consumption. E_{SO} represents the energy consumed by one synaptic operation (SO). r_{SO} is the average SO processing rate. Hence, the dynamic power consumption is estimated by counting the number of SOs and can be formulated as Equation 5:

$$f_{\text{so}} = \frac{(\sum_{i=1}^{N1} IN_i \times DE_i + \sum_{j=1}^{N2} LS_j \times LD_j)}{T} \times E_{\text{SO}} \quad (5)$$

where $N1$ represents the number of input neurons, $N2$ refers to the the number of liquid neurons, IN_i is the total number of spikes generated by the i_{th} neuron in the input layer, and DE_i is the number of destination neurons connected to the i_{th} input neuron; LS_j is the total number of spikes generated by the j_{th} neuron in the liquid layer, and LD_j is the number of destination neurons connected to the j_{th} neuron. T is the total execution time of the LSM.

D. MOTPE/D - Pareto front Search

Due to the high-dimensional and mixed variable space as listed in Table I , we proposed a multi-objective optimization (MOO) algorithm named MOTPE/D, which is shown in the right part in Figure 1. At each iteration, i.e., the j-th iteration,

a) Tchebycheff decomposition is applied first to convert previous observations $(\boldsymbol{x}, \boldsymbol{y})$ into $(\boldsymbol{x}, h)$ (where, $\boldsymbol{y} = \boldsymbol{F}(\boldsymbol{x})$, $h = \max_{1 \le i \le m} \left\{ \boldsymbol{\lambda}_i^j |\boldsymbol{y}_i - \boldsymbol{r}_i| \right\}$). Weight vector $\boldsymbol{\lambda}^j$ is randomly selected from the even spread weight vectors set S.

b) Previous observations are split into good observations D_l^j and poor ones D_g^j based on the quantile γ. The converted observation sequence for the $j-th$ sub-problem, D^j, is split into good observations D_l^j $(= \{(\boldsymbol{x}, h) \mid \boldsymbol{x}$ with the best-$\lceil \gamma \times \left|D^j\right| \rceil$ values in $D^j\})$ and poor observations $D_g^j (= D^j \setminus D_l^j)$.

c) Probability estimation functions $l^j(\boldsymbol{x})$ (for D_l^j) and $g^j(\boldsymbol{x})$ (for D_g^j) are constructed respectively. Expected Improvement (EI) is adopted in this work as the acquisition function. $\boldsymbol{x}^*$ with the greatest acquisition function value is selected, and $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is added to previous observations D.

d) $\boldsymbol{x}^*$ is evaluated and the Pareto front is updated, then $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is added to previous observations. When the preset iterations run out, the Pareto optimal solutions to the specific scenario are obtained.

IV. Experiments and Analysis

A. Experimental Setup

All experiments are conducted using Python 3.6, the SNN simulator Brian 2.4, and PyTorch 2.0. The optimization process is performed on Intel i9-8950HK CPU processors. Xilinx VU440 evaluation board is used to implement the proposed SNN processor.

To demonstrate the efficiency of the proposed MOO algorithm MOTPE/D, three case studies are introduced: N-MNIST classification, DVS-128 gesture recognition, and binaural sound source localization. MOTPE [14] and ParEGO [15] are adopted as two comparative algorithms in this work.

B. Experimental Results

1) Comparison with the state-of-the-art (SOTA) solutions.: In each case study, we pick out four representative solutions for detailed analysis: the solution with the highest accuracy S_{acc}, the solution with the fewest parameters S_{params}, the solution with the least power consumption S_{power}, and the Knee point. The Knee point is the solution with the smallest trade-off loss between conflict objectives, calculated as [16]. Detailed results are presented in Table II and Figure 3.

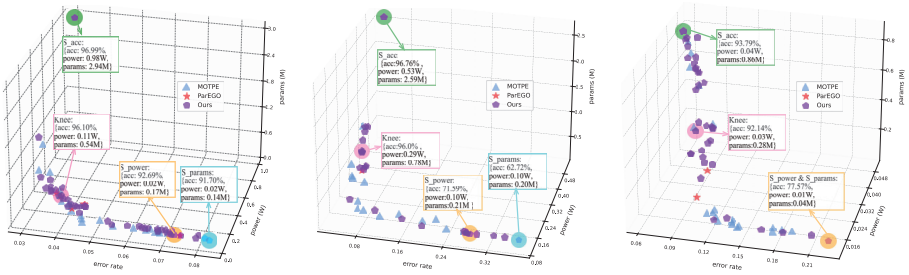
We can observe that obtained Knee points can achieve the best balance between different objective functions in both case studies. For N-MNIST classification, the SOTA solution [11] obtains an accuracy of 98.38%. However, $10,625$ neuron units are required, and the hardware overhead is $31\times$ that of us. Our proposed Knee point can reduce the number of neurons to 341 under the premise that the accuracy loss is less than 2.3%. For DVS-128 classification, it is satisfying that the Knee point found by our method can make a trade-off between acc, $params$, and $power$, and achieve the SOTA accuracy.

2) Ablation Studies: We collected all the Pareto solutions found by the MOPTE/D, MOTPE, and ParEGO within only 500 evaluations on three case studies. To make an intuitive comparison, we put together all the Pareto solutions found by three algorithms. Therefore, only Pareto solutions which can not be dominated by Pareto solutions found by the other two algorithms can be visualized in Figure 3. It can be intuitively observed that, in all three case studies, the Pareto solutions found by MOTPE/D mostly survive and are evenly distributed

TABLE II
MULTI-OBJECTIVE OPTIMIZATION RESULTS WITHIN 500 EVALUATIONS.

Solution	N-MNIST			DVS-128			SSL-7		
	$acc\ (\%)$	$power\ (W)$	$params\ (M)$	$acc\ (\%)$	$power\ (W)$	$params\ (M)$	$acc\ (\%)$	$power\ (W)$	$params\ (M)$
S_{acc}	96.99	0.98	2.94	96.76	0.53	2.59	93.79	0.04	0.86
S_{power}	92.69	0.02	0.17	71.59	0.10	0.21	77.57	0.01	0.04
S_{params}	91.70	0.14	0.02	67.72	0.10	0.20	77.57	0.01	0.04
Knee	96.10	0.11	0.54	96.0	0.29	0.78	92.14	0.03	0.28
SOTA	98.38 [11]	-	-	95.80 [12]	-	1.40	94.29 [13]	-	1.04

S_{acc}: optimal accuracy solution. S_{params}: Minimal-parameters solution. S_{power}: Minimal-power consumption solution.
Knee (**ours**): A solution with the smallest trade-off between all objectives.



(a) N-MNIST (b) DVS-128 (c) SSL-7

Fig. 3. Pareto Optimal solutions visualization of our method and peer algorithms. Extreme points and Knee points in Table II are highlighted and annotated.

along the Pareto front, thus better satisfying decision-makers with varied preferences for different objectives. The experimental results demonstrate that MOTPE/D can quickly find a superior Pareto solution set and effectively address the challenges posed by high-dimensional mixed-variable co-design space for SNN and neuromorphic processor.

V. CONCLUSION

We propose a generic three-phase hardware/algorithm co-design framework for SNN and neuromorphic processor. In this framework, we propose a novel MOO algorithm MOTPE/D aiming at high-dimensional mixed-variable and evaluation-expensive optimization problems. Furthermore, a generic power consumption model for LSM running on neuromorphic processors is formulated to support efficient candidate selection during optimization. Experiments show that, compared with peer MOO algorithms, i.e., MOTPE and ParEGO, MOTPE/D can always find more Pareto solutions, and push the *Pareto front* further within only 500 iterations.

REFERENCES

[1] R. Mao, L. Tang, X. Yuan, Y. Liu, and J. Zhou, "Stellar: Energy-efficient and low-latency snn algorithm and hardware co-design with spatiotemporal computation," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2024, pp. 172–185.

[2] J. Li, G. Shen, D. Zhao, Q. Zhang, and Y. Zeng, "Firefly: A high-throughput hardware accelerator for spiking neural networks with efficient dsp and memory optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 8, pp. 1178–1191, 2023.

[3] M. Davies, N. Srinivasa, T.-H. Lin *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[4] B. Na, J. Mok, S. Park, D. Lee, H. Choe, and S. Yoon, "Autosnn: Towards energy-efficient spiking neural networks," in *International Conference on Machine Learning*. PMLR, 2022, pp. 16 253–16 269.

[5] Y. Zhou, X. Dong, B. Akin *et al.*, "Rethinking co-design of neural architectures and hardware accelerators," *arXiv preprint arXiv:2102.08619*, 2021.

[6] K. Choi, D. Hong, H. Yoon, J. Yu, Y. Kim, and J. Lee, "Dance: Differentiable accelerator/network co-exploration," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 337–342.

[7] L. Yang, Z. Yan, M. Li *et al.*, "Co-exploration of neural architectures and heterogeneous asic accelerator designs targeting multiple tasks," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[8] H. Fan, M. Ferianc, Que *et al.*, "Algorithm and hardware co-design for reconfigurable cnn accelerator," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2022, pp. 250–255.

[9] A. Zela, J. Siems, L. Zimmer *et al.*, "Surrogate nas benchmarks: Going beyond the limited search spaces of tabular nas benchmarks," *arXiv preprint arXiv:2008.09777*, 2020.

[10] S. Tuli, C.-H. Li *et al.*, "Codebench: A neural architecture and hardware accelerator co-design framework," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 3, pp. 1–30, 2023.

[11] E. Iranmehr, S. B. Shouraki *et al.*, "Bio-inspired evolutionary model of spiking neural networks in ionic liquid space," *Frontiers in Neuroscience*, vol. 13, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:207958606

[12] X. Xiao, L. Wang, X. Chen *et al.*, "Dynamic vision sensor based gesture recognition using liquid state machine," in *International Conference on Artificial Neural Networks*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252369390

[13] Y. Li, J. Zhao, X. Xiao, R. Chen, and L. Wang, "Brain-inspired binaural sound source localization method based on liquid state machine," in *International Conference on Neural Information Processing*. Springer, 2023, pp. 198–213.

[14] Y. Ozaki, Y. Tanigaki, Watanabe *et al.*, "Multiobjective tree-structured parzen estimator for computationally expensive optimization problems," in *Proceedings of the 2020 genetic and evolutionary computation conference*, 2020, pp. 533–541.

[15] J. Knowles, "Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE transactions on evolutionary computation*, vol. 10, no. 1, pp. 50–66, 2006.

[16] R. Chen and K. Li, "Knee point identification based on the geometric characteristic," *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 764–769, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:245810610