2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS) | 979-8-3503-3267-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/AICAS57966.2023.10168605

SNNOpt: An Application-Specific Design Framework for Spiking Neural Networks

Jingyu He[1], Ziyang Shen[2], Fengshi Tian[1], Jinbo Chen[2],
Jie Yang[2], Mohamad Sawan[2], Tim Cheng[1], Paul Bogdan[3], Chi-Ying Tsui[1]
[1]Hong Kong University of Science and Technology
[2] Westlake University
[3] University of Southern California

***Abstract*—We propose a systematic application-specific hardware design methodology for designing Spiking Neural Network (SNN), SNNOpt, which consists of three novel phases: 1) an Olliver-Ricci-Curvature (ORC)-based architecture-aware network partitioning, 2) a reinforcement learning mapping strategy, and 3) a Bayesian optimization algorithm for NoC design space exploration. Experimental results show that SNNOpt achieves a 47.45% less runtime and 58.64% energy savings over state-of-the-art approaches.**

***Index Terms*—Spiking Neural Network, Network-on-Chip, Reinforcement Learning**

I. Introduction

Real-time cyber-physical systems (CPS) applications based on biosignal interpretation and control (e.g., brain-machine-body interfaces [17]) require algorithms and hardware with low latency and low power consumption features [3]. Due to their unique characteristics, the combination of spiking neural networks (SNN) and neuromorphic systems have become the preferred choice. However, the compatibility between SNNs and neuromorphic systems hinders low latency and low power consumption. Thus, several challenges must be addressed to optimize their collaborative deployment: (**1**) most existing neuromorphic chips possess hardware constraints that compromise the computing power of SNN models. For example, each TrueNorth neurosynaptic core [1] consists of 256 axons, a 256x256 crossbar, connecting neurons between two layers, and 256 neurons, yet limiting the number of post-synaptic neurons that a neuron can connect to inside a core. Hence, multiple cores are required for implementing the whole SNN. (**2**) SNN synaptic connections have many one-to-many connections between neurons. NoCs are designed for one-to-one or one-to-few connections, leading to high average latency and violating SNN timing constraints. Neuromorphic computing requires real-time processing of spikes at the same time step, or they will be discarded and affect system performance. (**3**) Prior works lack a comprehensive design analysis and deployment approach. For instance, PACMAN [6], SpiNeMap [2], SNEAP [11] and [7] address the SNN mapping to single and multiple crossbars). However, SpiNeMap and SNEAP do not involve NoC optimization for latency reduction, while [8] and [15] are hardware-specific.

To overcome all the above-mentioned challenges, we propose a comprehensive analytical application-specific methodology as summarized in Figs. 1&2, i.e., SNNOpt, for optimizing SNN on neuromorphic hardware and making the following novel contributions: (**1**) We are the first to exploit the

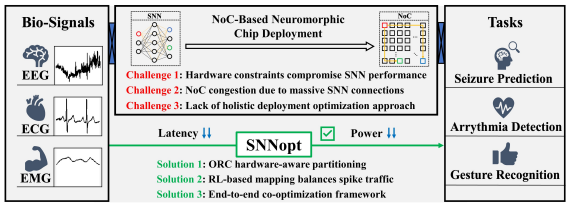

Fig. 1. SNN design challenges and SNNOpt's key features.

differeential geometry concepts, i.e., Olliver-Ricci curvature (ORC), to partition the SNN computational graph into strongly interacting heterogeneous processing clusters of neurons that not only satisfy the hardware constraints of the neuromorphic chips but also reduce the network traffic, hence, minimizing the network latency and energy consumption. (**2**) We propose a reinforcement learning (RL)-based approach with spike traffic re-balancing to map the identified heterogeneous processing clusters of neurons onto crossbars of the neuromorphic chip while minimizing the spike latency and congestion level. (**3**) We provide a Bayesian optimization (BO) NoC design space exploration (DSE) framework for specific SNN applications on neuromorphic chips, which completes the end-to-end co-optimization framework. SNNOpt identifies a set of fine-grained NoC design parameters that satisfy the timing constraints of the SNN and minimize energy consumption. (**4**) We demonstrate the effectiveness and benefits of our general methodology on SNN structures with three widely-used machine learning datasets. This pioneering SNNOpt methodology achieves 32% runtime reduction and 52% energy savings over the state-of-the-art approaches. Overall, SNNOpt provides a comprehensive methodology for optimizing SNN on neuromorphic hardware, by seamlessly integrating ORC-based clustering, RL-based mapping, and BO-based NoC design space exploration. This methodology enables the efficient implementation of SNN on neuromorphic chips, reducing energy consumption while meeting the timing and performance requirements of specific SNN applications.

II. ORC-based SNN-computation Partitioning

SNNs tend to have more neurons and synapses than one crossbar can accommodate. Hence, the SNN has to be partitioned into clusters before deploying onto neuromorphic hardware. A good partition balances the load of each crossbar for parallel execution and minimizes inter-cluster spike communication. In addition, the hardware constraints of the crossbar restrict the size of each neuron cluster. We propose an ORC-based partitioning algorithm that can obtain optimal

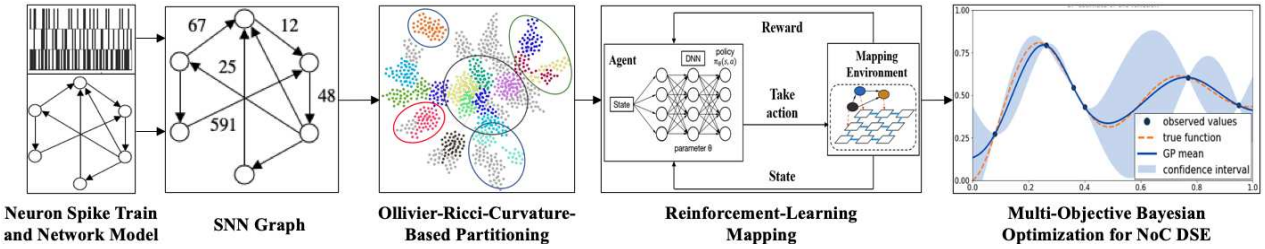

Fig. 2. Overview of SNNopt workflow.

partitioning with better convergence and faster runtime. It has several benefits: Firstly, it is deterministic and will not encounter oscillations which is common to Kernighan-Lin (KL) and other heuristic algorithms. Secondly, it tends to see the finer subdivisions in the network structures based on the local topology as quantified by the edge Ollivier-Ricci curvature (ORC) and hence higher accuracy with small communities. Thirdly, since the algorithm utilizes the differential geometry concept of network curvature, it performs particularly well for weighted graphs with internally densely-connected community structures, which is usually the case for neural networks.

To illustrate the ORC-based partitioning strategy, we define $G(N, S)$ to be a weighted and directed graph that represents an SNN with a set N of neurons and a set S of synapses. A synapse s_{ij} connects neuron i to j and fires w_{ij} spikes. We also define $P(V, E)$ to be a weighted and directed graph that represents the SNN partition graph with a set V of clusters of neurons and a set E of inter-cluster connections. v_i is the set of neurons of cluster i. A inter-cluster connection e_{ij} connects cluster i to j and transmits k_{ij} spikes.

Using these definitions, the problem of SNN partitioning under hardware constraints can be formulated as follows:

Given a $G(N, S)$ describing an SNN computation, **find** the non-overlapping clusters $P(V, E)$ that maximize $\sum_{i=1}^{|V|}(\frac{W_i - A_i}{W} - \frac{(W_i - \overline{W})^2}{W})$, where W_i stands for the number of spikes transmitted inside a cluster v_i, A_i represents the number of spikes transmitted from and to cluster v_i, $\overline{W}$ is the average number of spikes of all clusters, and W is the total spike count. The first term indicates the difference between the inner-cluster spikes and the inter-cluster spikes. To maximize the first term, we minimize the inter-cluster spike communication. The second term captures the difference between the inner-cluster spikes of each cluster and the average spikes of each cluster. The minus sign ensures the maximum degree of load-balancing. Meanwhile, $N_c \geq |V|$ and $\forall v_i \in V, |v_i| \leq c$ enforce the hardware constraints, where N_c is the number of crossbars of the neuromorphic chip, $|v_i|$ is the number of neurons of cluster i, c is the crossbar size.

The details are shown in Algorithm 1. For more information on differential geometry characterization of weighted networks, the reader can refer to [13], [14]. The time complexity of this algorithm is $O(|S|D^2)$, where $|S|$ is the number of edges in $G(N, S)$, and D is the average degree.

Algorithm 1: ORC-based Partitioning Algorithm

Input: $G(N, S)$
Output: $P(V, E)$
$\acute{G}(N, S) \leftarrow G(N, S)$ Calculate Ollivier-Ricci curvature for all edges;
while *there is negative edge curve* **do**
  Remove the most negatively curved edge;
  Re-calculate the Ollivier-Ricci curvature for the affected existing edges;
PreferentialAttachment($\acute{G}(N, S)$, number_of_communities, minimum_community_size);
Label each node in $\acute{G}$ with its community ID;
Regroup $\acute{G}$ into $P(V, E)$;
if $(N_c \geq |V|) \wedge (\forall v_i \in V, |v_i| \leq c)$ **then**
  return $P(V, E)$;
else
  if $N_c < |V|$ **then**
    Merge two of the smallest clusters;
    return $P(V, E)$;
  if $\exists v_i \in V, |v_i| > c$ **then**
    Apply ORC Community Detection to the over-sized cluster;
    return $P(V, E)$;

III. RL-BASED MAPPING

The choice of mapping neuron clusters onto tiled-based NoC heavily affects the latency and energy consumption because mapping different clusters on different tiles incur different routing paths that the spikes have to traverse. We proposed an RL-based mapping approach, which utilizes the deep neural network (DNN) as the decision-making engine. As DNN has a generalization ability, unseen raw states can also be well handled by good actions. Moreover, it is possible to train for objectives that are hard to optimize directly as long as the reward function correlates with the objective.

Objective Function and Problem Statement Prior work [2], [4], [11] minimize the average hop count of the inter-cluster spike since it is related to minimizing latency and energy consumption. The hop count is given by $H = \sum_{i=1}^{|V|}\sum_{j=1}^{|V|} md(i, j)k_{ij}$, where H stands for the total hop count of the mapping, and $md(i, j)$ represents the Manhattan distance between two tiles onto whom the neuron clusters i and j are mapped, respectively. k_{ij} denotes the total number of spikes transmitted between clusters i and j. We define $A(T, R)$ to be a directed graph that represents the architecture of the neuromorphic chip. Each t_i in T represents a crossbar tile and each r_{ij} in R stands for the routing from tile t_i to t_j.

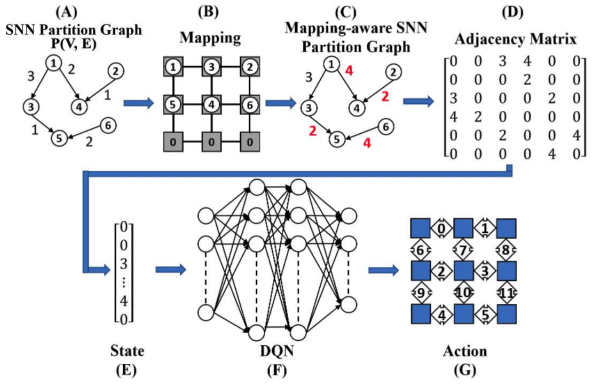

Fig. 3. (A) A partitioned SNN $P(V, E)$. (B) An NoC mapping of a partitioned SNN. (C) The mapping-aware SNN partition graph with edge weight multiplied by manhattan distance. (D) The adjacency matrix of the mapping-aware partition graph. (E) The mapping state is generated by flattening the adjacency matrix. (F) The Deep Q-Network of the reinforcement learning agent. (G) The 12 possible swaps of the mapping action space.

Mapping $M = A(T, R) \leftarrow P(V, E)$ is represented by a binary matrix $(m_{ij}) \in \{0, 1\}^{|T|\times|T|}$, where m_{ij} is 1 if cluster v_i is mapped to crossbar t_j, and 0 otherwise. Each cluster can only be mapped to one crossbar. The RL formulation of neuron cluster mapping consists of three aspects: state representation, action representation and rewards function.

Mapping State Representation We represent the mapping state with a vector of size $|V|^2$, where $|V|$ denotes the number of vertices in a partitioned neuron cluster graph $P(V, E)$. In order to encode the neuron spikes and hop count into the state representation, we first calculate the Manhattan distance between each (v_i, v_j) pair based on the current mapping. Then we multiply the distance with the corresponding weight in the partition graph to get a mapping-aware partition graph and its adjacency matrix. Finally, we flatten the mapping-aware adjacency matrix to get the state representation. Fig. 3(A)-(E) shows the process of mapping an SNN partition graph with 6 clusters onto a 3x3 neuromorphic chip. A tile of value 0 in Fig. 3(B) means the crossbar tile is idle. Since each crossbar can accommodate up to one neuron cluster, there are $(W \times H + 1)!$ possible mapping states in total, where W and H are the width and height of the 2D mesh.

Mapping Action Representation At each time step, the agent will move at least one neuron cluster to a different crossbar to either explore or exploit a better mapping. To keep the action space small and avoid duplicate mapping, we define action as swapping neuron clusters on adjacent crossbars. As shown in Fig. 3(G), for a 3x3 2D mesh neuromorphic chip, the action space is significantly reduced to $2WH - W - H$.

Rewards Based on the objective function in eq. III, the step reward is represented as $R(s, a) = H(s) - H(s')$, where s' is the new state by taking action a on state s. We give a positive reward when the new state generates a lower hop count and penalize the agent when the new state has a higher hop count.

We use Q-network [12] to approximate the action-value function, $Q(s, a; \theta) \approx Q^\star(s, a)$, where θ represents the weights of the neural network. As shown in Fig. 3(F), the network takes a mapping state vector as input and outputs the approximate action value of all possible swaps.

IV. Bayesian Optimization for NoC DSE

In this section, we propose a BO-based algorithm to identify a set of NoC design parameters that strike a balance between latency and power with the fewest possible evaluation runs.

NoC Design Space Exploration Formulation We formulate the NoC design as a multiple-objective optimization problem over a discrete design space $\mathbb{X} \subseteq \mathbb{R}$ to minimize latency and energy consumption. We define two objective functions: $f = (f_1(x) = latency(x), f_2(x) = power(x))$, which are evaluated by Booksim2 [9], a cycle-accurate NoC simulator. The design space consists of 6 NoC design parameters, including routing function, virtual channel (VC) allocation, arbitration type, priority type, number of VCs, and VC buffer size. A design point $x = \langle x_0, x_1, x_2, x_3, x_4, x_5\rangle$, specifying a set of NoC design parameters. The design space has 36288 points in total. Our goal is to identify the Pareto front of f, a set $\Gamma \subseteq \mathbb{X}$ of points, which are not dominated by any other points. To conduct the application-specific search, we modified BookSim2 so that it supports trace-based simulation. In each iteration of the search, the BookSim2 is configured based on the parameters of a design point in the design space and takes the spike trace of the SNN application as input to generate the latency and energy consumption data.

Bayesian Optimization for NoC DSE The objective function $f_1 = latency(x)$ and $f_2 = power(x)$ are modeled as two independent Gaussian Processes (GP), which are characterized by their means $\mu(x)$ and co-variates $k(x, x^\star)$. We first randomly sample several observations for both objectives to construct GP. Next, at each iteration t, the algorithm selects an NoC design point x_t based on $x_i = \text{argmax}_x\, a(x)$, where $a(x)$ is the acquisition function. And for evaluation, it generates a noisy sample as $y_{t,i} \sim \mathcal{N}(f_i(x_t), v_{t,i})$, where $v_{t,i}$ is the variance of noise introduced into the function observations. The evaluations of the acquisition function are cheap compared to the evaluations of the objective functions, and hence the overhead is negligible. We use a probability of improvement (PI) acquisition function to determine how to select an NoC design point x_{t+1} in the next iteration. Later, we append the $(x_t, y_{t,i})$ to the observation set. We repeat these steps until the maximum number of iterations is reached. Finally, we construct the Pareto front from the observation set.

V. Experimental Results

Experimental Setup Three biomedical applications (e.g., `SNN_ECG`, `SNN_EMG`, and `SNN_EEG` [16] and two Convolutional networks (e.g. ConvNet_CIFAR10 and LeNet_CIFAR10) are used for evaluation. We first convert these applications into SNN using SNN-IIR [5]. Then we collect the spike information and feed it into an in-house cycle-accurate TrueNorth simulator, which generates output spikes and crossbar trace. Finally, we modified BookSim2 [9] to support trace-based simulation for NoC DSE. Our evaluation consists of: (1) three individual phase approach comparisons with the state-of-the-art and (2) an ablation

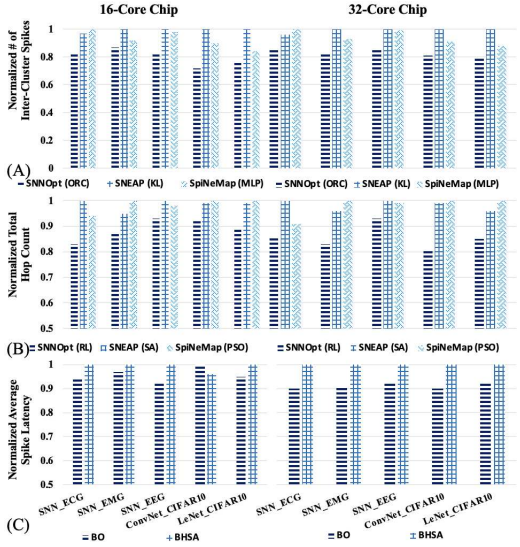

Fig. 4. The (A) partitioning, (B) mapping, and (C) NoC DSE comparison between the SNNOpt and benchmarks on 16 and 32 cores.

study to examine the effects of each component of our toolchain. We choose SpiNeMap [2] with Better-History SA (BHSA) NoC DSE [10] for end-to-end comparison. For the partitioning and mapping comparisons, we use SpiNeMap and SNEAP [11], while for NoC DSE baseline we use BHSA.

Results Fig. 4(A) shows the number of spikes on the shared interconnect normalized to that of SpiNeMap on 16-core, and 32-core configurations. Compared to KL algorithm of SpiNeMap, ORC reduces the spike count on average by 21% and 17% on 16-core and 32-core, respectively. This is because KL is highly sensitive to its initial random solution and requires the partitions to be equal in size, missing the opportunities to find the global optimal sizes when some neurons are more densely connected than others. ORC outperforms SNEAP by 21% and 17% fewer inter-cluster spikes on 16 and 32 partitions, respectively. Our ORC-based algorithm tends to better discover the hierarchical structure of the network and minimizes the spike communication between neuron clusters.

Next, in Fig. 4(B), we compare our RL-based mapping algorithm with two heuristics, i.e., simulated annealing (SA) and particle swarm optimization (PSO). Compared to SA, our mapping algorithm achieved an average of 10.8% and 13.6% less hop count on 16-core and 32-core configurations, respectively. This is because SA searches instead of learning the changing environments while the adaptive characteristics of pre-trained RL agents enable a fast convergence. Moreover, our RL outperforms the PSO of SpiNeMap by as much as 18% when running `ConvNet_CIFAR10` on the 32-core chip. The PSO is prone to converge prematurely and be trapped into a local minimum, especially when the design space is large as shown in the case of a 32-core configuration.

Fig. 4(C) demonstrates the superiority of BO-based NoC DSE over BHSA in terms of both spike latency on interconnect. BO obtains an NoC configuration with a maximum of 7% less average latency. The SA falls out of favor especially when the dimension of 2D-mesh NoC increases. BO is able to work

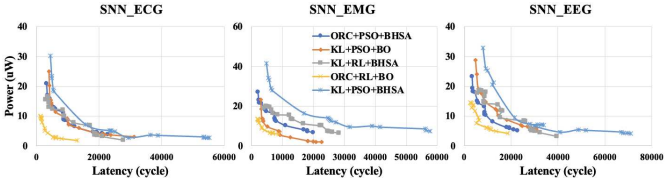

Fig. 5. Pareto fronts of SNN_ECG, SNN_EMG, SNN_EEG.

directly over the black box of the NoC DSE with relatively few iterations and supports multiple objectives evaluation.

We perform ablation studies to evaluate the impact of our three proposed techniques on three applications, and present the results in Fig. 5. The baseline is SpiNeMap with BHSA, i.e. KL+PSO+BHSA. Our methodology, ORC+RL+BO, significantly outperforms the baseline. In the ablation study, ORC+PSO+BHSA replaces the KL partitioning scheme, achieving a 23.78% reduction in packet delay and 14.05% reduction in power consumption. The RL-based mapping approach improves delay and power by an average of 12.24% and 18.78%, respectively. BO alone reduces delay by 9.1% and power by 15.95% on average in KL+PSO+BO. When all three approaches are combined, our framework achieves 47.45% lower runtime and 58.6% lower power consumption compared to the baseline of SpiNeMap with BHSA. While we recognize the importance of detailed accuracy comparison, due to the page limitation, we plan to investigate this aspect in future work. However, preliminary results suggest SNNOpt achieves less degradation of accuracy since the spike latency is dramatically reduced.

VI. Conclusion

This paper introduces a systematic design methodology for SNN with three phases: 1) an ORC-based network partitioning, 2) an RL-based mapping strategy, and 3) a BO-based NoC DSE. Significant runtime reduction and energy savings are achieved when compared with the state-of-the-art.

Acknowledgment

This research was supported by ACCESS – AI Chip Center for Emerging Smart Systems, sponsored by InnoHK funding, Hong Kong SAR. P.B. gratefully acknowledge the support by the National Science Foundation Career award under Grant No. Cyber-Physical Systems / CNS-1453860, the NSF award under Grant CCF-1837131, MCB-1936775, CNS-1932620, the U.S. Army Research Office (ARO), the Defense Advanced Research Projects Agency (DARPA) Young Faculty Award and DARPA Director Award under Grant No. N66001-17-1-4044, an Intel faculty award, the Okawa Foundation award, and a Northrop Grumman grant. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied by the Defense Advanced Research Projects Agency, the Army Research Office, the Department of Defense, or the National Science Foundation.

REFERENCES

[1] F. Akopyan et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

[2] A. Balaji et al. Mapping spiking neural networks to neuromorphic hardware. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(1):76–86, 2019.

[3] P. Bogdan. A cyber-physical systems approach to personalized medicine: challenges and opportunities for noc-based multicore platforms. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 253–258. IEEE, 2015.

[4] A. Das et al. Mapping of local and global synapses on spiking neuromorphic hardware. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1217–1222. IEEE, 2018.

[5] H. Fang et al. Exploiting neuron and synapse filter dynamics in spatial temporal learning of deep spiking neural network. In *29th International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 2799–2806. International Joint Conferences on Artificial Intelligence, 2020.

[6] F. Galluppi et al. A hierachical configuration system for a massively parallel neural hardware platform. In *Proceedings of the 9th Conference on Computing Frontiers*, CF '12, page 183–192, New York, NY, USA, 2012. Association for Computing Machinery.

[7] M. Hu et al. Memristor crossbar-based neuromorphic computing system: A case study. *IEEE Transactions on Neural Networks and Learning Systems*, 25(10):1864–1878, 2014.

[8] Y. Ji et al. Neutrams: Neural network transformation and co-design under neuromorphic hardware constraints. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13. IEEE, 2016.

[9] N. Jiang et al. A detailed and flexible cycle-accurate network-on-chip simulator. In *2013 IEEE international symposium on performance analysis of systems and software (ISPASS)*, pages 86–96. IEEE, 2013.

[10] Z. Kang et al. Application-specific network-on-chip design space exploration framework for neuromorphic processor. In *Proceedings of the 17th ACM International Conference on Computing Frontiers*, pages 71–80, 2020.

[11] S. Li et al. Sneap: A fast and efficient toolchain for mapping large-scale spiking neural network onto noc-based neuromorphic platform. *arXiv preprint arXiv:2004.01639*, 2020.

[12] V. Mnih et al. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[13] J. Sia et al. Ollivier-ricci curvature-based method to community detection in complex networks. *Scientific reports*, 9(1):9800, 2019.

[14] J. Sia, W. Zhang, E. Jonckheere, D. Cook, and P. Bogdan. Inferring functional communities from partially observed biological networks exploiting geometric topology and side information. *Scientific Reports*, 12(1):10883, 2022.

[15] E. Stromatias et al. Scalable energy-efficient, low-latency implementations of trained spiking deep belief networks on spinnaker. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015.

[16] F. Tian, J. Yang, S. Zhao, and M. Sawan. Neurocare: A generic neuromorphic edge computing framework for healthcare applications. *Frontiers in Neuroscience*, 17, 2023.

[17] Y. Xue, S. Rodriguez, and P. Bogdan. A spatio-temporal fractal model for a cps approach to brain-machine-body interfaces. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 642–647. IEEE, 2016.