Improving Inference Lifetime of Neuromorphic Systems via Intelligent Synapse Mapping

Shihao Song, Twisha Titirsha, and Anup Das
Electrical and Computer Engineering, Drexel University, Philadelphia, PA
{shihao.song,tt624,anup.das}@drexel.edu

***Abstract*—Non-Volatile Memories (NVMs) such as Resistive RAM (RRAM) are used in neuromorphic systems to implement high-density and low-power analog synaptic weights. Unfortunately, an RRAM cell can switch its state after reading its content a certain number of times. Such behavior challenges the integrity and program-once-read-many-times philosophy of implementing machine learning inference on neuromorphic systems, impacting the Quality-of-Service (QoS). Elevated temperatures and frequent usage can significantly shorten the number of times an RRAM cell can be reliably read before it becomes absolutely necessary to reprogram. We propose an architectural solution to extend the read endurance of RRAM-based neuromorphic systems. We make two key contributions. First, we formulate the read endurance of an RRAM cell as a function of the programmed synaptic weight and its activation within a machine learning workload. Second, we propose an intelligent workload mapping strategy incorporating the endurance formulation to place the synapses of a machine learning model onto the RRAM cells of the hardware. The objective is to extend the inference lifetime, defined as the number of times the model can be used to generate output (inference) before the trained weights need to be reprogrammed on the RRAM cells of the system. We evaluate our architectural solution with machine learning workloads on a cycle-accurate simulator of an RRAM-based neuromorphic system. Our results demonstrate a significant increase in inference lifetime with only a minimal performance impact.**

***Index Terms*—Neuromorphic Computing, Non-Volatile Memory (NVM), Endurance, RRAM, Spiking Neural Network (SNN)**

I. Introduction

Neuromorphic systems are integrated circuits that mimic the neuro-biological architecture of the central nervous system [1]. They employ variants of integrate-and-fire (I&F) neurons as computational units and analog weights as synaptic storage. I&F neurons use spikes to encode information, where each spike is a voltage or current pulse, typically of an ms duration [2]. Due to its event (spike)-driven operations, a neuromorphic system consumes less power and therefore, well suited as the hardware for inference of trained machine learning models deployed in power-constrained environments such as Embedded Systems and Internet-of-Things (IoT).

Non-Volatile Memory (NVM) technologies such as Filamentary Oxide-based Resistive RAM (RRAM), Phase-Change Memory (PCM), and Spin-based Magnetic RAM (MRAM) enable low-voltage multilevel operations, making them suitable for implementing analog synaptic weight storage in neuromorphic systems [3]–[5]. Of these emerging new memory technologies, hafnia (HfO_2)-based RRAM has shown a significant promise due to its CMOS compatibility at scaled nodes, allowing the fabrication of high-density synaptic storage for neuromorphic systems. The synaptic weights are programmed on RRAM cells as conductance. An RRAM cell can be programmed to a high-resistance state (HRS) or one of the low-resistance states (LRS). Unfortunately, RRAM cells have limited *read endurance*, i.e., an RRAM cell can switch its state after performing a certain number of reads [6]. To give an example, a single quasi-static read for 5000 ms or 5000 reads with 1-ms read time can lead to an abrupt change from HRS to LRS state in an RRAM cell [6]. To put in the context of neuromorphic computing, an RRAM cell can reliably propagate 5000 1-ms spikes before it becomes absolutely necessary to reprogram the state of the cell.

We now extrapolate this RRAM device behavior to the application level, describing such extrapolation with the running example of VGG, a deep learning model trained on CIFAR-10 dataset and performing inference on an RRAM-based neuromorphic system. Figure 1 shows the histogram of average spikes per image propagating through the synapses of VGG. We collected these statistics by analyzing CIFAR-10 training and test datasets. We see that some synapses propagate more spikes than others when inferring an image. These are called the *critical synapses* and they decide how many images can be reliably inferred using the VGG model before it becomes necessary to reprogram the trained synaptic weights on the RRAM cells of the hardware.

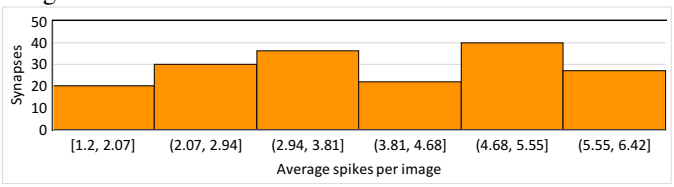


Fig. 1. Spike distribution across the synapses of VGG.

To give an example, assume n to be the maximum number of spikes per image on the critical synapses of VGG ($n = 6.42$ in Figure 1). Then, the RRAM cells need to be reprogrammed once every $\frac{5000}{n} \approx 778$ images to ensure correctness. The time to infer 778 images is called the *inference lifetime*. Formally,

$$\text{Inference Lifetime} = \frac{\text{Read Endurance}}{\text{spikes per image}} \quad (1)$$

If the RRAM devices implementing VGG are not reprogrammed before the inference lifetime expires, then the accuracy of VGG can drop significantly (accuracy in the low 20% is reported in [6]).

Periodic reprogramming of synaptic weights on a neuromorphic system challenges the program-once-read-many-times

philosophy of machine learning inference hardware, which can impose significant system overhead. To give an example, imagine such systems are deployed at the edge nodes of an IoT infrastructure. Frequent updates of these nodes with trained weights will 1) increase communication between the edge and cloud, and 2) reduce the Quality-of-Service (QoS) due to offlining of the edge nodes every time they are reprogrammed.

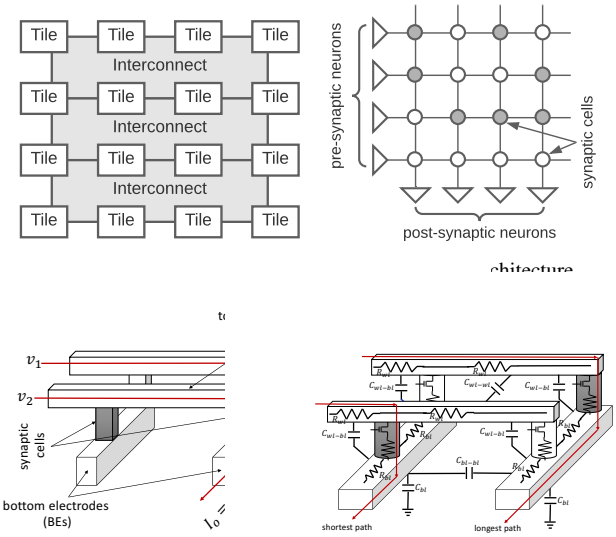
We observe that inside a neuromorphic system, the RRAM cells are organized into crossbars. The parasitic IR drops in a crossbar create a difference in the voltage needed to propagate spike through the RRAM cells in the crossbar [7] Such voltage differences create a variation of read endurance of the RRAM cells, i.e., some RRAM cells are stronger than others, where the *strength* of an RRAM cell is measured terms of its read endurance, which is a function of the voltag Unfortunately, if the critical synapses (those that propaga more spikes) are mapped on weaker RRAM cells (tho that have low read endurance), then the inference lifetin can decrease significantly, lowering the QoS. We propose a *intelligent synapse allocation strategy*, which analyzes spik propagating through each synapse of a machine learning model during inference and uses such information to map the model's synaptic weights to the RRAM cells considering the variation in their read endurance. The objective is to maximize the inference lifetime of the hardware. Our architectural solution is built on the following three key **contributions**.

- First, we investigate the internal architecture of an RRAM-based neuromorphic system and estimate the endurance variation through detailed circuit-level simulations at different process and temperature corners.
- Second, we analyze a trained machine learning model and estimate the spikes propagating through its synapses.
- Finally, we use a Hill-Climbing approach that uses Binary Non-Linear Programming (BNLP) to map the synapses of a machine learning model to the RRAM cells such that the critical synapses are always mapped to stronger RRAM cells, thereby improving the inference lifetime.

We evaluate our architectural approach with different machine learning models on NeuroXplorer [8], a cycle-accurate simulator of RRAM-based neuromorphic system. Results show a significant improvement in inference lifetime with a minimal impact on model performance.

II. Background

A neuromorphic system is implemented as a tiled architecture (see Fig. 2a), where the tiles are interconnected hierarchically using a shared interconnect such as Network-on-Chip (Noc) [9] or Segmented Bus [10]. This is the representative architecture of many recent systems such as TrueNorth [11], Loihi [12], and DYNAPs [13]. In many recent systems, a tile is implemented using a crossbar, which is illustrated in Figure 2b. An MxM crossbar can accommodate M pre-synaptic neurons, mapped along the rows and M post-synaptic neurons, mapped along the columns. There are M^2 synaptic cells, which store the weights. Figure 2c shows a 2x2 crossbar in three-dimension, with the top electrodes forming the rows and bottom electrodes forming the columns. A synaptic cell is placed at each intersection of top and bottom electrodes.



(c) A 3-D view of a crossbar. (d) Parasitics of a crossbar.

Fig. 2. Neuromorphic system architecture with crossbars.

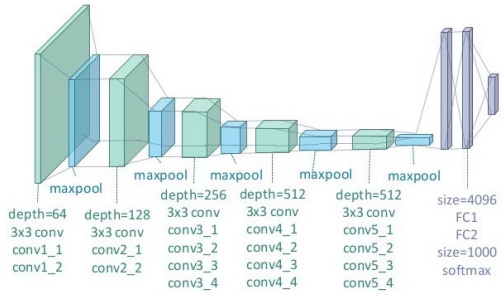
Figure 2d shows the different parasitic components inside a crossbar. Such components cause variable delays on the current paths inside the crossbar. For simplicity, we have only shown the current on the shortest and the longest paths in the crossbar, where the length of a current path is measured in terms of the number of parasitic elements on the path. Therefore, spike propagation delay through synapses on longer paths is higher than on shorter paths. Although optimizing inference lifetime is our primary focus, we also evaluate the impact of our architectural solution on spike propagation delay (see Section V). Parasitic components in a crossbar also lead to voltage variations, which impact read endurance of the synaptic cells. We analyze such impact in Section III.

A. Machine Learning Inference on Neuromorphic Systems

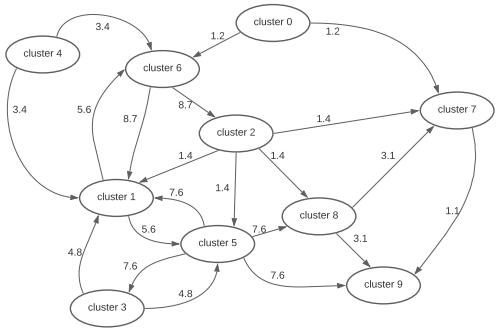
Each crossbar in a neuromorphic system can accommodate only a limited number of neurons and synapses. To map large models, the model is first partitioned into clusters of neurons and synapses, where each cluster can fit onto a crossbar of the hardware [14]–[20]. Figure 3a shows the architecture of VGG for CIFAR-10 classification. Figure 3b shows the first 10 clusters generated using SpiNeMap [14], a state-of-the-art approach to map machine learning inference to neuromorphic systems. The figure illustrates the connections between these clusters, with the number on edge representing the average number of spikes communicated between the source and destination clusters when processing an image during inference.

When partitioning a machine learning model into clusters, SpiNeMap aims to minimize the inter-cluster spikes, which reduces the energy consumption on the interconnect of the hardware. There are also other optimization objectives proposed in literature. Examples include improving crossbar utilization [19], reducing crossbar usage [15], reducing energy consumption [7], [21]–[25], and reducing circuit aging [26]–[30]. None of these approaches address mapping of the

synapses of a cluster to the synaptic cells of a crossbar for the purpose of inference on neuromorphic systems. To understand why such mapping matters, we now introduce the background on filamentary oxide-based RRAM technology, which can be used to design the synaptic cells of a crossbar.



(a) VGG Convolution Neural Network (CNN).



(b) First 10 clusters of VGG (out of 95,452) clusters).

Fig. 3. Trained VGG model and its clusters generated using SpiNeMap [14].

B. Oxide-based Resistive RAM (RRAM) Technology

The resistance switching random access memory (RRAM) technology presents an attractive option for implementing the synaptic cells of a crossbar due to its demonstrated potential for low-power multilevel operation and high integration density [4]. An RRAM cell is composed of an insulating film sandwiched between conducting electrodes forming a metal-insulator-metal (MIM) structure (see Figure 4). Recently, filament-based metal-oxide RRAM implemented with transition-metal-oxides such as HfO_2, ZrO_2, and TiO_2 has received considerable attention due to their low-power and CMOS-compatible scaling.

Synaptic weights are represented as conductance of the insulating layer within each RRAM cell. To program an RRAM cell, elevated voltages are applied at the top and bottom electrodes, which re-arranges the atomic structure of the insulating layer. Figure 4 shows the High-Resistance State (HRS) and the Low-Resistance State (LRS) of an RRAM cell. An RRAM cell can also be programmed into intermediate low-resistance states, allowing its multilevel operations. In this work, we consider each RRAM cell to be programmed to one HRS and three LRS states, implementing two bits per synapse.

III. Variation of Read Endurance

Inside a neuromorphic system, the long bitlines and wordlines of a crossbar are the major sources of parasitic (IR)

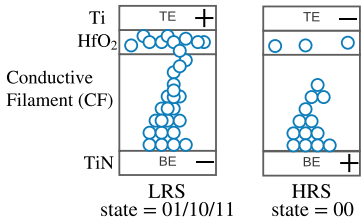


Fig. 4. Operation of an RRAM cell with the HfO_2 layer sandwiched between the metals Ti (top electrode) and TiN (bottom electrode). The left subfigure shows the formation of LRS states with the formation of conducting filament (CF). This represents logic states 01, 10, and 11. The right subfigure shows the depletion of CF on application of a negative voltage on the TE. This represents the HRS state or logic 00.

voltage drops, introducing asymmetry in the voltage applied across the different RRAM cells in the hardware [31]–[33]. To study this behavior, we simulate a 128x128 RRAM-based crossbar circuit using the predictive technology model (PTM) [34] and RRAM-specific parameters [35].

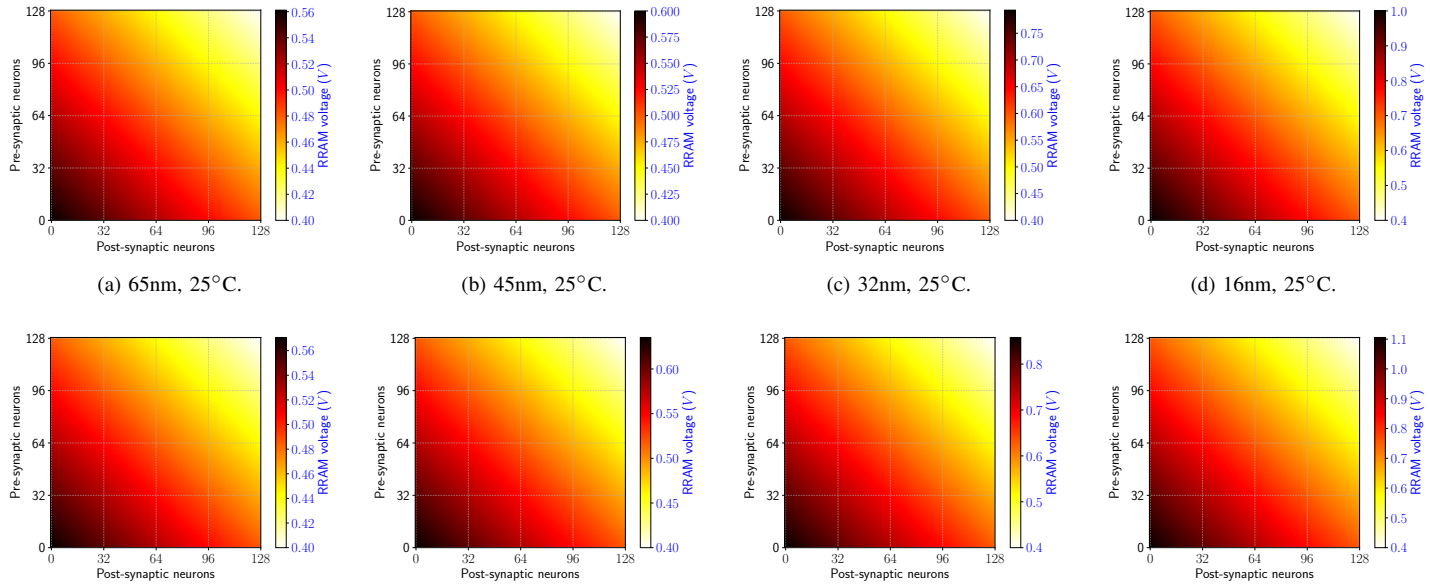
Figure 5 shows the variation of RRAM voltages in a 128x128 crossbar during inference for four technology nodes (65 nm, 45 nm, 32 nm, and 16 nm) and two temperature settings (25°C and 50°C). We make the following three key observations. First, RRAM voltages at the bottom left corner of the crossbar are higher than those at the top right corner. This is because the current paths via the RRAM cells at the bottom left corner are shorter, i.e., they have lower parasitic voltage drops than at the top right corner. Second, with technology scaling, the voltage variation increases. The highest RRAM voltage in the crossbar is 1.1 V at 16 nm and 25°C (Figure 5d) compared to 0.57 V at 65 nm and 25°C (Figure 5a). This difference is because the unit parasitic resistance of the electrodes increases from 1Ω at 65 nm to 3.8Ω at 16 nm [34]. The value of the parasitic resistance is expected to increase with technology scaling, with a value $\approx 25\Omega$ at 5 nm [36]. Third, RRAM voltage increases with temperature (Figures 5e-5h vs. Figures 5a-5d). This is because with increase in temperature, the leakage current through the access transistor of each RRAM cell in the crossbar increases. Therefore, to obtain a certain current margin at the readout unit of the crossbar, the input voltage applied on the top electrodes needs to increase, which increases the RRAM voltages.

A. Voltage-Dependant Read Endurance of RRAM cells

We now provide a formulation of the read endurance of the RRAM cells in a crossbar as a function of the input voltage.[1] In RRAM technology, the transition from HRS state is governed by a sudden decrease of the vertical filament gap on application of stress voltage during spike propagation [6]. The rate of change of the filament gap of the RRAM cell at the $(i,j)^{\text{th}}$ location in the crossbar is

$$\frac{dg_{i,j}}{dt} = -\vartheta_0 \cdot e^{-\frac{E_a}{kT}} sinh\left(\frac{\gamma_{i,j} \cdot a_0}{L} \cdot \frac{qV_{i,j}}{kT}\right), \text{ where } \gamma_{i,j} = \gamma_0 - \beta \cdot \frac{g_{i,j}}{g_0}^3 \quad (2)$$

[1]Limited write endurance of RRAM cells has been studied before in the context of neuromorphic computing [37], [38]. This is the first work that studies the read endurance problem and proposes an intelligent solution.



(a) 65nm, 25°C. (b) 45nm, 25°C. (c) 32nm, 25°C. (d) 16nm, 25°C.

(e) 65nm, 50°C. (f) 45nm, 50°C. (g) 32nm, 50°C. (h) 16nm, 50°C.

Fig. 5. Variation in RRAM voltages within an 128x128 crossbar for inference. Such variations are reported for four technology nodes (65nm, 45nm, 32nm, and 16nm) and two temperature settings (25°C and 50°C).

In the above equation, t defines the state transition time, g_0 is the initial filament gap of the RRAM cell, $V_{i,j}$ is the voltage applied to the cell, $\gamma_{i,j}$ is the local field enhancement factor and is related to the gap $g_{i,j}$, a_0 is the atomic hoping distance, and γ_0 is a fitting constant.

The transition from one of the LRS states is governed by the lateral filament growth [6]. The time for state transition in the $(i,j)^{\text{th}}$ RRAM cell is given by

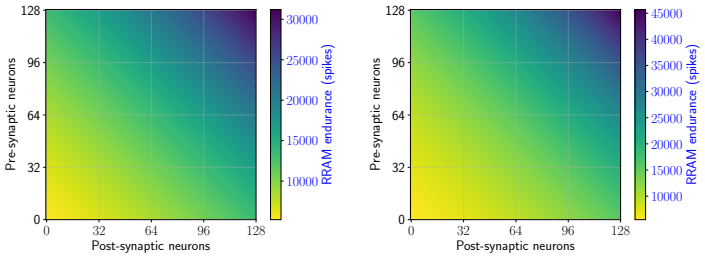
$$t_{i,j}(LRS) = 10^{-14.7 \cdot V_{i,j} + 6.7} \text{sec} \tag{3}$$

Using Equations 2 and 3, the read endurance of the $(i,j)^{\text{th}}$ RRAM cell can be derived as

$$E_{i,j}(HRS/LRS) = \frac{t_{i,j}(HRS/LRS)}{1\ ms\ \text{(spike duration)}} \tag{4}$$

Figure 6 shows the endurance variation of a 128x128 crossbar at 45 nm node and at 25°C with each RRAM cell programmed to 1) HRS state (Figure 6a) and 2) one of the LRS states (Figure 6b). We observe that endurance of an RRAM cell is higher if the cell is programmed to an LRS state compared to when it is programmed to the HRS state.

From the machine learning workload perspective, synapses can either be in the HRS state or in one of the LRS states. Therefore, based on how the synapses of a model are mapped inside a crossbar, the endurance map will assume intermediate forms between Figures 6a and 6b. To simplify the problem formulation, we define *equivalence* in terms of inference lifetime as follows. Consider A and B to be two synaptic weights in a machine learning model with spikes S_A and S_B, respectively. Without loss of generality, consider A to be in HRS state and B in LRS state. Let these weights are programmed on the RRAM cells at the same $(i,j)^{\text{th}}$ location in two different crossbars. Then, the inference lifetime due



(a) Each RRAM cell in HRS state. (b) Each RRAM cell in LRS state.

Fig. 6. Variation in RRAM endurance within an 128x128 crossbar at 45 nm node and at 25°C. Such variations are reported for the RRAM cells programmed in (a) HRS and (b) one of the LRS states.

to A is $\frac{E_{i,j}(HRS)}{S_A}$ and that due to B is $\frac{E_{i,j}(LRS)}{S_B}$. Synaptic weights A and B are considered to be equivalent in terms of the inference lifetime at the $(i,j)^{\text{th}}$ position in a crossbar if

$$\frac{E_{i,j}(HRS)}{S_A} = \frac{E_{i,j}(LRS)}{S_B} \tag{5}$$

We use Equation 5 for mapping and inference lifetime computation purposes. Once the mapping is decided, RRAM cells are programmed to their actual state.

IV. Problem Formulation

The mapping of a machine learning model to hardware is formulated in the following three steps.

1) Formulating inference lifetime of model clusters.
2) Cluster mapping with unlimited hardware resources.
3) Cluster mapping with limited hardware resources.

We now elaborate on these mapping steps.

A. Formulating Inference Lifetime of Model Clusters

We consider the mapping of a cluster $\mathbb{C} = (Pre, Post, Syn)$ of a machine learning model onto a crossbar of the hardware. Here Pre is the set of pre-synaptic neuron, $Post$ is the set of post-synaptic neuron, and Syn is the set of synapses between the pre- and post-synaptic neurons of the cluster. Each neuron $n_i \in Pre$ is characterized by a number $spk(n_i)$, indicating the average number of spikes generated by this neuron per image during inference. Each synapse $s_{i,j} \in Syn$ connecting the pre-synaptic neuron $n_i \in Pre$ and post-synaptic neuron $n_j \in Post$ is associated with a number $wt(s_{i,j})$ representing its synaptic weight. The number of spikes on the synapse $s_{i,j}$ is the same as the number of spikes generated by its pre-synaptic neuron n_i, i.e., $spk(s_{i,j}) = spk(n_i)$.

Consider the mapping of this cluster $\mathbb{C}$ to a MxM crossbar $\mathbb{H} = (In, Out)$ with a set In of input ports to map pre-synaptic neurons and a set Out of output ports to map post-synaptic neurons. Here $|In| = |Out| = M$.

Let $X_{i,k}$ be a binary variable representing the mapping of pre-synaptic neuron $n_i \in Pre$ to the input port $i_k \in In$ and $Y_{j,l}$ be a binary variable representing the mapping of post synaptic neuron $n_j \in Post$ to the output port $o_l \in Out$.

The problem we are aiming to solve is this: find the binary variables $X_{i,j}$ and $Y_{j,l}$ such that the inference lifetime is maximized when mapping the cluster to a crossbar. Therefore, the *objective function* is the inference lifetime. To maximize the objective function, we define the following constraints.

- Each pre-synaptic neuron can be mapped to exactly one input port of the hardware, i.e., $\sum_{k=1}^{M} X_{i,k} = 1 \ \forall i$.
- Each post-synaptic neuron can be mapped to exactly one output port of the hardware, i.e., $\sum_{l=1}^{M} Y_{j,l} = 1 \ \forall j$

To formulate the objective function itself, we consider the synapse $s_{i,j} \in Syn$, which connects the pre-synaptic neuron n_i with the post-synaptic neuron n_j. In terms of the variables $X_{i,k}$ and $Y_{j,l}$, the synapse $s_{i,j}$ is mapped to the RRAM cell at $(k,l)^{\text{th}}$ position in the crossbar. The inference lifetime of the synapse can be computed by first considering the equivalence to HRS state using Equation 5, and then dividing the HRS endurance of the RRAM cell with the number of spikes on the synapse using Equation 1. This is given by

$$f(i,j) = \text{Inference Lifetime}(i,j) = \sum_{k=1}^{M}\sum_{l=1}^{M} X_{i,k} \cdot Y_{j,l} \cdot \frac{E_{k,l}(HRS)}{spk_{eq}(s_{i,j})} \quad (6)$$

The maximization problem is

$$\max_{\substack{1 \le i \le |Pre| \\ 1 \le j \le |Post|}} f(i,j) \quad (7)$$

The use of binary (discrete) variables makes the optimization problem of Equation 7 non-convex, while the product term in Equation 6 makes this non-linear (NL). Therefore, the optimization problem we are aiming to solve is a Non-Convex Binary Non-Linear Programming (BNLP) problem and there is no guarantee of optimality [39]. We use the smoothing method proposed in the Ph.D. dissertation [40] to solve this BNLP problem. The optimized inference lifetime obtained from mapping the cluster $\mathbb{C}$ to the crossbar $\mathbb{H}$ is

$$\text{Inference Lifetime}(\mathbb{C}, \mathbb{H}) = f_{\text{opt}} \quad (8)$$

In this study, we have ignored process variations across the crossbars of a hardware. So, the inference lifetime of a cluster is the same, irrespective of which crossbar this cluster is mapped to in the hardware. This allows us to decouple the crossbar term from Equation 8. We use the variable $\mathcal{L}_i$ to represent the inference lifetime of the cluster $\mathbb{C}_i$.

B. Cluster Mapping with Unlimited Hardware Resources

If the hardware contains unlimited number of crossbars, then each crossbar can map at most one cluster of a machine learning model. Therefore, the inference lifetime for the model when it is mapped to the hardware is the minimum inference lifetime of all the clusters of the hardware, i.e,

$$\text{Inference Lifetime} = \text{minimum}\{\mathcal{L}_i, \ \forall i \in 1, 2, \cdots, N_C\}, \quad (9)$$

where N_C is the number of clusters of the model.

C. Cluster Mapping with Limited Hardware Resources

If the number of crossbars in the hardware is limited, then each crossbar may need to be shared across multiple clusters of a machine learning model. We now formulate the cluster mapping problem as follows. Let the binary variable $Z_{i,j}$ indicate the mapping of cluster $\mathbb{C}_i$ to crossbar $\mathbb{H}_j$, i.e.,

$$Z_{i,j} = \begin{cases} 1 & \text{if cluster } \mathbb{C}_i \text{ is mapped to crossbar } \mathbb{H}_j \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

The problem we are aiming to solve is this: find the binary variables $Z_{i,j}$ such the inference lifetime of the machine learning model on the hardware is improved. We define the following constraint: each cluster must be mapped to only one crossbar, i.e., $\sum_{j=1}^{N_H} Z_{i,j} = 1 \ \forall i$, where N_H is the number of crossbars of the hardware with $N_H \le N_C$.

To explore the cluster-to-crossbar mapping search space for the maximum inference lifetime, we use a *Hill-Climbing*-based local search [41]. Each mapping solution is represented by $\mathbb{Z} \in \mathbb{R}^{N_C \times N_H}$. Figure 7 shows the working of the search algorithm.

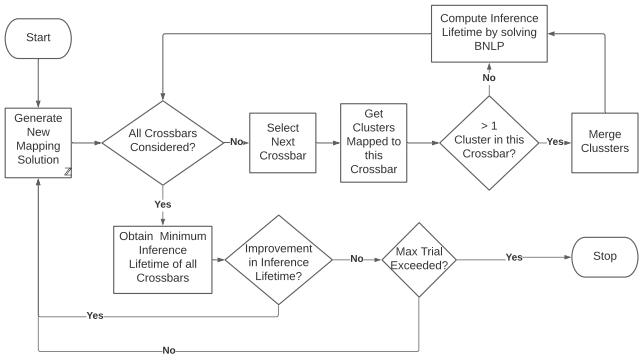


Fig. 7. Flowchart for Hill-Climbing-based search.

For each solution $\mathbb{Z}$ generated by the algorithm, it computes the inference lifetime of each crossbar as follows. If a crossbar has only one cluster, the inference lifetime is computed by solving the proposed BNLP problem formulation (Equation 7). If more than one cluster is mapped to the crossbar, the clusters on this crossbar are merged. Merging a cluster involves combining the clusters to form a larger cluster and

is described mathematically as follows. Merging of clusters $\mathbb{C}_a = (Pre_a, Post_a, Syn_a)$ and $\mathbb{C}_b = (Pre_b, Post_b, Syn_b)$ is

$$\texttt{merge}(\mathbb{C}_a, \mathbb{C}_b) = \mathbb{C}_{a+b} = (Pre_{a+b}, Post_{a+b}, Syn_{a+b}), \tag{11}$$

where $Pre_{a+b} = Pre_a \cup Pre_b$, $Post_{a+b} = Post_a \cup Post_b$, and $Syn_{a+b} = Syn_a \cup Syn_b$. Once the clusters mapped to a crossbar are merged, the inference lifetime of the merged cluster is computed using the proposed BNLP formulation. Then the algorithm computes the overall inference lifetime as the minimum of the inference lifetime of all crossbars of the hardware. If this value is higher than the best solution obtained thus far, the mapping is retained and the algorithm proceeds to find a better mapping solution. Otherwise, the algorithm continues to explore for a few more iterations to see if a better mapping solution can be generated. This general formulation can be applied to the case where $N_H > N_C$, i.e., the number of hardware crossbars is greater than model clusters.

D. Performance Impact

We now formally quantify the performance degradation due to our mapping exploration (Sections IV-A, IV-B, and IV-B) using the previously-introduced notations. Let $\mathbb{Z}_{\text{opt}} \in \{0,1\}^{N_C \times N_H}$ be the optimum mapping (one with the highest inference lifetime obtained using the Hill-Climbing approach of Figure 7) of the clusters of a machine learning model to the crossbars of a neuromorphic hardware.

Within the optimum mapping, let $\mathbb{M}_{\text{opt}_p} = [\mathbb{X}_{\text{opt}_p} \mid \mathbb{Y}_{\text{opt}_p}]\ \forall p \in 1, 2, \cdots, N_C$ be the optimum mapping (one with the highest inference lifetime obtained by solving the BNLP of Equation 7) of pre- and post-synaptic neurons of the p^{th} cluster to a crossbar. Here $\mathbb{X}_{\text{opt}_p} \in \{0,1\}^{|Pre_p| \times |In|}$ and $\mathbb{Y}_{\text{opt}_p} \in \{0,1\}^{|Post_p| \times |Out|}$.

Let $d_{k,l}$ represents the delay in spike propagation through the RRAM cell at the $(k,l)^{\text{th}}$ location in a crossbar. Therefore, the spike propagation delay through the synapse $s_{i,j}$ is

$$\text{synapse\_delay}_{i,j} = \sum_{k=1}^{M} \sum_{l=1}^{M} X_{i,k} \cdot Y_{j,l} \cdot d_{k,l} \tag{12}$$

Therefore, the average spike propagation delay of the cluster when mapped to a crossbar is

$$\text{cluster\_delay} = \frac{\sum_{i=1}^{Pre} \sum_{j=1}^{Post} spk(s_{i,j}) \cdot \text{synapse\_delay}_{i,j}}{\sum_{i=1}^{Pre} \sum_{j=1}^{Post} spk(s_{i,j})} \tag{13}$$

Equation 13 can be extrapolated to compute the spike propagation delay of the entire machine learning model when mapped to the neuromorphic hardware as

$$\text{hardware\_delay} = \frac{\sum_{p=1}^{N_C} \sum_{i=1}^{Pre_p} \sum_{j=1}^{Post_p} spk(s^p_{i,j}) \cdot \text{cluster\_delay}_p}{\sum_{p=1}^{N_C} \sum_{i=1}^{Pre_p} \sum_{j=1}^{Post_p} spk(s^p_{i,j})} \tag{14}$$

V. Results and Discussion

We evaluate the proposed approach using NeuroXplorer [8], a cycle-accurate neuromorphic system simulator that uses tile-based architecture (see Fig. 8). We model the DYNAPs neuromorphic hardware [13] with hierarchical NoC-based interconnect. Each tile has one 128×128 crossbar. Table I reports the relevant hardware parameters.

TABLE I
Major simulation parameters extracted from [13].

Neuron technology	45nm
Synapse technology	HfO_2-based RRAM

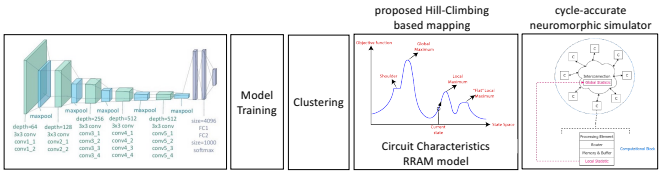


Fig. 8. Evaluation framework based on NeuroXplorer [8].

We evaluate 10 machine learning programs which are representative of three most commonly-used neural network classes: convolutional neural network (CNN), multi-layer perceptron (MLP), and recurrent neural network (RNN). Table II summarizes the topology, the number of neurons and synapses of these applications, and their baseline accuracy on the DYNAPs neuromorphic hardware using the SpiNeMap [14].

TABLE II
Applications used to evaluate the proposed approach.

Class	Applications	Dataset	Synapses	Neurons	Topology	Accuracy
CNN	LeNet	MNIST	282,936	20,602	CNN	85.1%
	AlexNet	ImageNet	38,730,222	230,443	CNN	69.8%
	VGG	CIFAR-10	99,080,704	554,059	CNN	90.7 %
	HeartClass [42], [43]	Physionet	1,049,249	153,730	CNN	63.7%
MLP	MLPDigit	MNIST	79,400	884	FeedForward	91.6%
	EdgeDet	CARLsim	114,057	6,120	FeedForward	100%
	ImgSmooth	CARLsim	9,025	4,096	FeedForward	100%
RNN	HeartEstm [44]	Physionet	66,406	166	Recurrent Reservoir	100%
	VisualPursuit [45]	[45]	163,880	205	Recurrent Reservoir	47.3%
	RNNDigit	MNIST	11,442	567	Recurrent Reservoir	83.6%

A. Inference Lifetime on Unlimited Hardware Resources

Figure 9 reports the inference lifetime for each application for the proposed approach normalized to SpiNeMap. For reference, we have reported the absolute inference lifetime in frames for each application using the proposed approach. For image-based applications (LeNet, AlexNet, VGG, MLPDigit, EdgeDet, ImgSmooth, and RNNDigit), a frame corresponds to an individual image. For other time-series applications (HeartClass, HeartEstm, and VisualPursuit), a frame corresponds to a window of 500ms. We make the following two observations.

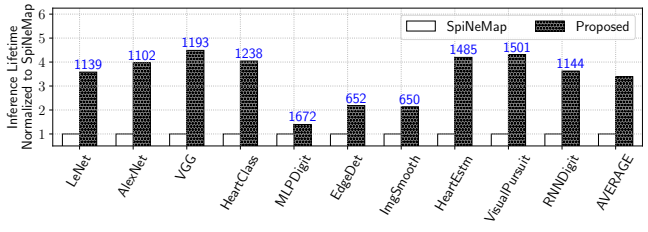


Fig. 9. Inference lifetime normalized to SpiNeMap.

First, the inference lifetime obtained using the proposed approach is on average 3.4x higher than SpiNeMap. This improvement is because the proposed approach uses the novel BNLP formulation to decide how the synaptic weights of a cluster need to be mapped to the RRAM cells of a crossbar to maximize the inference lifetime. To do so, the proposed approach incorporates both RRAM device and machine learning

workload characteristics. The proposed formulation ensures that critical synapses (those that propagate more spikes) are never mapped to the weaker cells (those that have low read endurance). SpiNeMap on the other hand, maps the synaptic weight of a cluster arbitrarily to the RRAM cells of a crossbar.

Second, the inference lifetime improvement of the proposed approach is, in general, lower for smaller applications such as MLPDigit, EdgeDet, and ImgSmooth (average 1.9x), compared to larger applications such as LeNet, AlexNet, and VGG (average 4x). This is because with larger applications (ones with more clusters), the proposed approach has greater scope to improve the inference lifetime by intelligently mapping the synaptic weights in all the clusters.

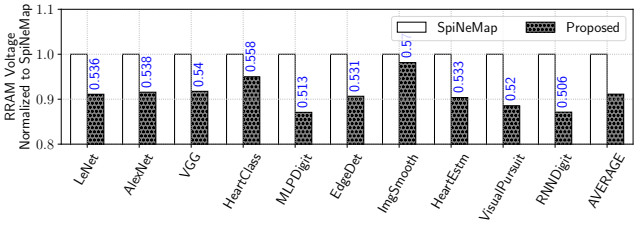


Fig. 10. Average RRAM voltage normalized to SpiNeMap.

To give further insight, Figure 10 plots the average voltage on the RRAM cells within a crossbar when the clusters of each evaluated application are mapped to them. For reference, we have reported the absolute voltage in V for the proposed approach. We observe that the average voltage on the RRAM cells in the proposed approach is 9% lower than SpiNeMap. This is because the proposed approach uses the top right corners of a crossbar (see Figure 2d) to place the synaptic weights. This is where the parasitic voltage drops are higher, resulting in a lower voltage across the RRAM cells.

B. Spike Delay

Unfortunately, the RRAM cells at the top right corner of each crossbar introduce longer spike propagation delay than those at the bottom left corner. To estimate the average increase in spike delay, Figure 11 plots the spike propagation delay through the crossbar for each application, normalized to SpiNeMap. For reference, we have reported the absolute delay in ms using the proposed approach. We observe that the spike propagation delay using the proposed approach is only an average 6% higher than SpiNeMap.

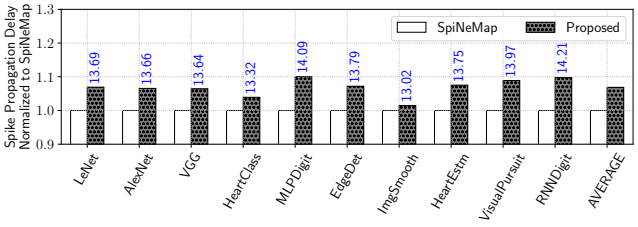


Fig. 11. Crossbar spike propagation delay normalized to SpiNeMap.

C. Inference Lifetime with Limited Hardware Resources

Figure 12 plots the inference lifetime obtained using the proposed approach normalized to SpiNeMap as we increase the hardware size from 256 crossbars to 1024 crossbars. We make the following two observations.

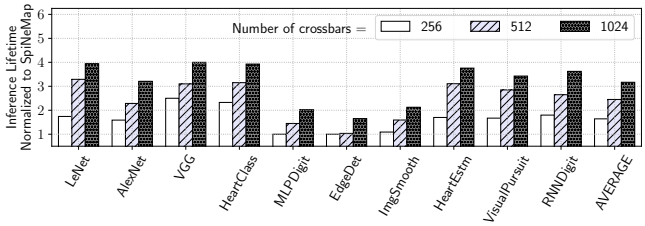


Fig. 12. Inference lifetime for three different hardware configurations.

First, the inference lifetime of the proposed approach increases with an increase in the size of the hardware. With 256, 512, and 1024 crossbars in the hardware, the inference lifetime of the proposed approach is higher than SpiNeMap by an average of 1.64x, 2.45x, and 3.16x. With fewer crossbars in the hardware, the number of clusters mapped to each hardware increases, increasing the crossbar utilization. Therefore, the proposed approach has limited scope to reorganize the synapses onto the RRAM cells, resulting in lower improvement than the case where there are more crossbars in the hardware. Second, the improvement of inference lifetime in smaller applications like MLPDigit, EdgeDet, and ImgSmooth is not significant compared to larger applications like LeNet, AlexNet, and VGG. From these results, we conclude that the proposed approach has a greater opportunity to increase the inference lifetime for crossbars with lower utilization.

D. Exploration Time

Table III reports the exploration time of the proposed Hill-Climbing-based mapping exploration for each of the evaluated applications. Column 2 reports the number of clusters of these applications generated using SpiNeMap [14]. For these clusters, Columns 3, 4, and 5 report the exploration time for three hardware configurations – 256 crossbars, 512 crossbars, and 1024 crossbars, respectively. We make the following two observations. First, the exploration time increases with increase in the number of crossbars due to the increase in the size of the search space. Second, for applications such as ImgSmooth and RNNDigit, there is no significant increase in the exploration time because the number of clusters for these applications is less than the number of crossbars in the hardware. Therefore, the application mapping time is essentially the time in solving the BNLP problem.

TABLE III
EXPLORATION TIME OF THE PROPOSED APPROACH.

Applications	Clusters	256 crossbars	512 crossbars	1024 crossbars
LeNet	2,066	2,189 sec	2,702 sec	3, 183 sec
AlexNet	30,105	40,667 sec	61,434 sec	101,924 sec
VGG	95,452	70,180 sec	144,090 sec	389, 644 sec
HeartClass	7,871	5,111 sec	8,110 sec	12, 145 sec
MLPDigit	520	594 sec	784 sec	1,010 sec
EdgeDet	437	405 sec	612 sec	824 sec
ImgSmooth	41	52 sec	52 sec	52 sec
HeartEstm	325	321 sec	486 sec	611 sec
VisualPursuit	1,001	1,138 sec	1,520 sec	1,921 sec
RNNDigit	90	114 sec	114 sec	114 sec

E. Technology Scaling

Figure 13 plots the inference lifetime of the proposed approach normalized to SpiNeMap for four technology nodes – 65nm, 45nm (default), 32nm, and 16nm. We observe that the

improvement of inference lifetime over SpiNeMap increases as the technology scales down, even though the absolute inference lifetime is lower at scaled nodes. This is because with technology scaling, the endurance variation within each crossbar becomes more significant. Therefore, the proposed approach, which incorporates such variation in the cluster mapping and synapse placement process leads to higher inference lifetime compared to SpiNeMap.

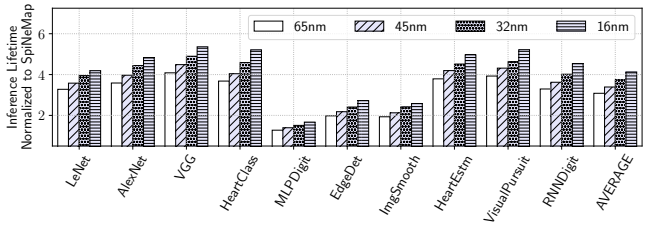


Fig. 13. Impact of technology scaling on inference lifetime.

VI. Conclusions

We present a novel Binary Non-Linear Programming (BNLP) formulation of the inference lifetime of machine learning workloads when mapped on to the RRAM cells of a neuromorphic system. Using such formulation, we show that the parasitic IR drops in the system create a significant difference in read endurance of the RRAM cells. We incorporate the BNLP formulation and endurance variation inside a Hill-Climbing-based mapping exploration to find an optimum mapping of the clusters of an inference model to the crossbars of a hardware, improving its inference lifetime. Our formulation ensures that critical synapses (those that propagate more spikes) are never mapped on to the weaker cells (ones that have lower endurance). We evaluate our approach with 10 machine learning applications on a cycle-accurate simulator of state-of-the-art neuromorphic hardware. Our results demonstrate an average 3.4x improvement in inference lifetime with only 6% increase in spike propagation delay.

Acknowledgment

This work is supported by the National Science Foundation Faculty Early Career Development Award CCF-1942697.

References

[1] C. Mead, "Neuromorphic electronic systems," *Proc. of the IEEE*, 1990.
[2] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, 1997.
[3] G. W. Burr *et al.*, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X*, 2017.
[4] A. Mallik *et al.*, "Design-technology co-optimization for OxRRAM-based synaptic processing unit," in *VLSIT*, 2017.
[5] F. Catthoor *et al.*, "Very large-scale neuromorphic systems for biological signal processing," in *CMOS Circuits for Biological Sensing and Processing*, 2018.
[6] W. Shim *et al.*, "Impact of read disturb on multilevel RRAM based inference engine: Experiments and model prediction," in *IRPS*, 2020.
[7] T. Titirsha *et al.*, "Thermal-aware compilation of spiking neural networks to neuromorphic hardware," in *LCPC*, 2020.
[8] A. Balaji *et al.*, "NeuroXplorer 1.0: An extensible framework for architectural exploration with spiking neural networks," in *ICONS*, 2021.
[9] X. Liu *et al.*, "Neu-NoC: A high-efficient interconnection network for accelerated neuromorphic systems," in *ASP-DAC*, 2018.
[10] A. Balaji *et al.*, "Exploration of segmented bus as scalable global interconnect for neuromorphic computing," in *GLSVLSI*, 2019.
[11] M. V. Debole *et al.*, "TrueNorth: Accelerating from zero to 64 million neurons in 10 years," *Computer*, 2019.
[12] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, 2018.
[13] S. Moradi *et al.*, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *TBCAS*, 2017.
[14] A. Balaji *et al.*, "Mapping spiking neural networks to neuromorphic hardware," *TVLSI*, 2020.
[15] A. Balaji *et al.*, "Enabling resource-aware mapping of spiking neural networks via spatial decomposition," *ESL*, 2020.
[16] S. Song *et al.*, "Compiling spiking neural networks to neuromorphic hardware," in *LCTES*, 2020.
[17] A. Balaji *et al.*, "PyCARL: A PyNN interface for hardware-software co-simulation of spiking neural network," in *IJCNN*, 2020.
[18] A. Balaji *et al.*, "Design methodology for embedded approximate artificial neural networks," in *GLSVLSI*, 2019.
[19] Y. Ji *et al.*, "NEUTRAMS: Neural network transformation and co-design under neuromorphic hardware constraints," in *MICRO*, 2016.
[20] A. Balaji *et al.*, "Compiling spiking neural networks to mitigate neuromorphic hardware constraints"," in *IGSC Workshops*, 2020.
[21] T. Titirsha *et al.*, "On the role of system software in energy management of neuromorphic computing," in *CF*, 2021.
[22] A. Das *et al.*, "Mapping of local and global synapses on spiking neuromorphic hardware," in *DATE*, 2018.
[23] A. Das *et al.*, "Dataflow-based mapping of spiking neural networks on neuromorphic hardware," in *GLSVLSI*, 2018.
[24] A. Balaji *et al.*, "A framework for the analysis of throughput-constraints of SNNs on neuromorphic hardware," in *ISVLSI*, 2019.
[25] A. Balaji *et al.*, "Run-time mapping of spiking neural networks to neuromorphic hardware," *JSPS*, 2020.
[26] S. Song *et al.*, "Improving dependability of neuromorphic computing with non-volatile memory," in *EDCC*, 2020.
[27] S. Kundu *et al.*, "Special Session: Reliability analysis for ML/AI hardware," in *VTS*, 2021.
[28] A. Balaji *et al.*, "A framework to explore workload-specific performance and lifetime trade-offs in neuromorphic computing," *CAL*, 2019.
[29] S. Song *et al.*, "A case for lifetime reliability-aware neuromorphic computing," in *MWSCAS*, 2020.
[30] S. Song *et al.*, "Dynamic reliability management in neuromorphic computing," *JETC*, 2021.
[31] S. Song *et al.*, "Aging-aware request scheduling for non-volatile main memory," in *ASP-DAC*, 2021.
[32] S. Song *et al.*, "Design methodologies for reliable and energy-efficient PCM systems," in *IGSC Workshops*, 2020.
[33] S. Song *et al.*, "Exploiting inter- and intra-memory asymmetries for data mapping in hybrid tiered-memories," in *ISMM*, 2020.
[34] W. Zhao *et al.*, "Predictive technology model for nano-CMOS design exploration," *JETC*, 2007.
[35] P.-Y. Chen *et al.*, "Compact modeling of RRAM devices and its applications in 1T1R and 1S1R array design," *TED*, 2015.
[36] M. E. Fouda *et al.*, "Modeling and analysis of passive switching crossbar arrays," *TCAS I*, 2017.
[37] T. Titirsha *et al.*, "Endurance-aware mapping of spiking neural networks to neuromorphic hardware," *TPDS*, 2021.
[38] T. Titirsha *et al.*, "Reliability-performance trade-offs in neuromorphic computing," in *IGSC Workshops*, 2020.
[39] S. Boyd *et al.*, *Convex optimization*. Cambridge University Press, 2004.
[40] K.-M. Ng, "A continuation approach for solving nonlinear optimization problems with discrete variables," *Doctor Dissertation, Department of Management Science and Engineering of Stanford University*, 2002.
[41] B. Selman *et al.*, "Hill-climbing search," *En. of Cog. Sc.*, 2006.
[42] A. Balaji *et al.*, "Power-accuracy trade-offs for heartbeat classification on neural networks hardware," *JOLPE*, 2018.
[43] A. Das *et al.*, "Heartbeat classification in wearables using multi-layer perceptron and time-frequency joint distribution of ECG," in *CHASE*, 2018.
[44] A. Das *et al.*, "Unsupervised heart-rate estimation in wearables with Liquid states and a probabilistic readout," *Neural Networks*, 2018.
[45] H. J. Kashyap *et al.*, "A recurrent neural network based model of predictive smooth pursuit eye movement in primates," in *IJCNN*, 2018.