Dynamic Reliability Management in Neuromorphic Computing

SHIHAO SONG, JUI HANAMSHET, ADARSHA BALAJI, and ANUP DAS, Drexel University
JEFFREY L. KRICHMAR and NIKIL D. DUTT, University of California, Irvine
NAGARAJAN KANDASAMY, Drexel University
FRANCKY CATTHOOR, Imec, Belgium

Neuromorphic computing systems execute machine learning tasks designed with Spiking Neural Networks (SNNs). These systems are embracing non-volatile memory (NVM) to implement high-density and low-energy synaptic storage. Elevated voltages and currents needed to operate NVMs cause aging of CMOS-based transistors in each neuron and synapse circuit in the hardware, drifting the transistor's parameters from their nominal values. If these circuits are used continuously for too long, the parameter drifts cannot be reversed, resulting in permanent degradation of circuit performance over time, eventually leading to hardware faults. Aggressive device scaling increases power density and temperature, which further accelerates the aging, challenging the reliable operation of neuromorphic systems. Existing reliability-oriented techniques periodically de-stress all neuron and synapse circuits in the hardware at fixed intervals, assuming worst-case operating conditions, without actually tracking their aging at run time. To de-stress these circuits, normal operation must be interrupted, which introduces latency in spike generation and propagation, impacting the inter-spike interval and hence, performance, e.g., accuracy. We observe that in contrast to long-term aging, which permanently damages the hardware, short-term aging in scaled CMOS transistors is mostly due to Bias Temperature Instability (BTI). The latter is heavily workload-dependent and more importantly, partially reversible. We propose a new architectural technique to mitigate the aging-related reliability problems in neuromorphic systems, by designing an intelligent run-time manager (**NCRTM**), which dynamically de-stresses neuron and synapse circuits in response to the short-term aging in their CMOS transistors during the execution of machine learning workloads, with the objective of meeting a reliability target. NCRTM de-stresses these circuits only when it is absolutely necessary to do so, otherwise reducing the performance impact by scheduling de-stress operations off the critical path. We evaluate NCRTM with state-of-the-art machine learning workloads on a neuromorphic hardware. Our results demonstrate that NCRTM significantly improves the reliability of neuromorphic hardware, with marginal impact on performance.

CCS Concepts: • **Hardware → Neural systems**; **Bio-embedded electronics**; **Aging of circuits and systems**; • **Software and its engineering → Runtime environments**.

Additional Key Words and Phrases: Neuromorphic Computing, Machine Learning, Spiking Neural Network (SNN), Bias Temperature Instability (BTI), Lifetime Reliability, Non-Volatile Memory (NVM), Phase-Change Memory (PCM), Run-time Manager (RTM).

Authors' addresses: Shihao Song, shihao.song@drexel.edu; Jui Hanamshet, jh3454@drexel.edu; Adarsha Balaji, ab3586@drexel.edu; Anup Das, anup.das@drexel.edu, Drexel University, 3101 Market Street, Philadelphia, PA, 19104; Jeffrey L. Krichmar, jkrichma@uci.edu; Nikil D. Dutt, dutt@ics.uci.edu, University of California, Irvine, 6210 Donald Bren Hall, Irvine, CA, 92697; Nagarajan Kandasamy, nk78@drexel.edu, Drexel University, 3101 Market Street, Philadelphia, PA, 19104; Francky Catthoor, Francky.Catthoor@imec.be, Imec, Kapeldreef 75, 3001 Leuven, Belgium.

1550-4832/2021/1-ART1 $15.00
https://doi.org/10.1145/3462330

ACM Reference Format:
Shihao Song, Jui Hanamshet, Adarsha Balaji, Anup Das, Jeffrey L. Krichmar, Nikil D. Dutt, Nagarajan Kandasamy, and Francky Catthoor. 2021. Dynamic Reliability Management in Neuromorphic Computing. *ACM J. Emerg. Technol. Comput. Syst.* 1, 1, Article 1 (January 2021), 27 pages. https://doi.org/10.1145/3462330

1 INTRODUCTION

Spiking Neural Networks (SNNs) [60] are machine learning approaches designed with spike-based computations [46] and bio-inspired learning algorithms [15] (See Appendix A for background on SNNs). SNN-based workloads are typically executed on event-driven neuromorphic hardware such as TrueNorth [35], Loihi [34], and DYNAP-SE [64]. These hardware platforms are extremely energy-efficient, thanks to their event-driven activation and their tile-based distributed architecture with in-place neural computations and synaptic storage [72]. We investigate the internal architecture of neurons and synapses in DYNAP-SE (see Figures 3b and 4b), and found that these circuits consist of transistors built using bulk CMOS or FinFet technologies [1, 18, 44].[1] When operated at a high voltage and temperature, the transistor's parameters strongly drift from their nominal values. This is called *aging*. In fact, in scaled technology nodes, this aging happens even under nominal conditions and from the very start of using the devices leading to the so-called soft breakdown. The most important breakdown mechanism is the Bias Temperature Instability (BTI) [50, 51, 99]. Strongly depending on the workload, BTI is highly variable and it is largely reversible under nominal conditions on removal of the stress voltage. So it leads only to parametric time-dependent variability, affecting mainly delay and leakage power. If the neurons and synapses in a neuromorphic hardware are used continuously for long duration at elevated operating conditions, the parameter drifts cannot be reversed [100], leading to permanent functional degradation of the circuit and eventually, hardware faults [53, 68, 91]. The permanent fault rates in integrated circuits can be described by the bathtub curve as shown in Figure 1. Post manufacturing, integrated circuits (IC) are characterized by high failure rates as these circuits are subjected to manufacturing tests, such as stuck-at, at-speed, burn-in, etc., which filters out defective circuits and circuits with short lifetime. The probability of the successful circuits surviving for a longer period of time, increases. The failure rate, therefore, decreases over time. This phase is known as the infant mortality period. This is followed by a period of constant failure rate, often referred as useful life. The last phase is known as the wear-out or the aging phase and is characterized by increasing fault rate. Recent studies on reliability reveal that, if wear-out is not addressed from early device usage stage (e.g., the beginning of useful life period), circuits can age faster than anticipated with the wear-out phase settling earlier in life (shown by the red dashed line in the figure).

To address time-dependent variability or aging, circuit designers often set worst-case and hence highly pessimistic reliability-related extra design margins, which unnecessarily constrain performance. Our objective is to analyze the circuit aging in neuromorphic hardware at real-time and take corrective measures at the architecture-level to reverse the parameter drifts based on the utilization of neuron and synapse circuits within a machine learning workload.

Recently, Non-Volatile Memory (NVM) is used in neuromorphic hardware to implement high-density and low-energy synaptic storage [14]. Several NVMs are explored for this purpose – Oxide-based Resistive RAM (OxRRAM) [62], Phase Change Memory (PCM) [66], Ferro-Electric RAM [65], and Spin-Transfer Torque Magnetic or Spin-Orbit-Torque RAM (STT- and SoT-MRAM) [97].[2] NVMs require either high voltage (OxRRAM, PCM and FeFET) or high current (MRAM) to operate, which accelerates the aging of transistors in neuron and synapse circuits in a neuromorphic hardware [2, 55, 83, 84, 88]. Aggressive device scaling increases power density and temperature,

[1] We believe these architectures are similar for other designs like TrueNorth [35] and Loihi [34].
[2] Beside neuromorphic computing, NVMs are also used as main memory for conventional computing [54, 57, 71, 85, 87, 89, 90].

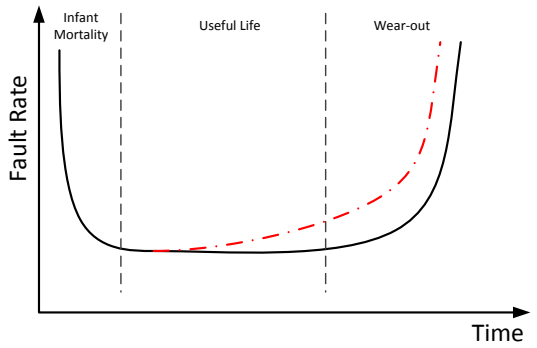


Fig. 1. Bathtub curve for permanent faults.

which makes reliability even worse. Therefore, circuit aging is emerging as one of the primary reliability concerns for neuromorphic hardware designed with NVMs [16].

The reliability problem we are addressing in this work is due to high voltage operations of NVMs. That can also occur in other system contexts,[3] but it is in particular an issue for SNNs due to the following reasons. To address this high voltage NVM problem, periodic de-stress of the peripheral circuit is necessary, which impacts inter-spike interval (ISI) when machine learning models are executed on these circuits. The performance (e.g., accuracy) of SNNs depends on ISI. Therefore, the reliability issues of NVMs lead to performance issues in SNNs.

Prior works on mapping machine learning workloads to neuromorphic hardware have mostly focused on compilation techniques, with the objective of improving machine learning performance on hardware. Examples of such approaches include hardware utilization-based mapping [3, 4, 8, 10, 11, 24, 40, 47, 48, 86], energy-based mapping [9, 32, 94], and endurance-based mapping [92, 93, 95]. The recently-proposed approach RENEU [88] is the only compile-time based technique that maps the neurons and synapses to the hardware to improve the long-term, i.e., the lifetime reliability. Although compile-time based aging mitigation approaches have unique advantages such as low computation overhead, predictability, and performance guarantee, they are often conservative and therefore, may miss significant performance and reliability improvement opportunities. Dynamic approaches are flexible, adaptive, and potentially more effective in a highly dynamic environment, such as ones where the inference data deviates strongly from training examples. We show that both performance and reliability can be improved significantly if neuron and synapse circuits are de-stressed periodically at run-time based on current data.

On the run-time front, very few approaches address the run-time management of neuromorphic computing.[4] In [10], the authors propose a fast approach to remap online learning SNNs on a neuromorphic hardware after every learning epoch to improve model performance. In DTRO [2], the authors propose a hybrid approach to estimate the reliability degradation for machine learning workloads at design-time using training data, and use this information to de-stress all hardware circuits during run-time at fixed intervals, without actually tracking the circuit aging. The effectiveness of this approach is limited to supervised techniques only and the availability of representative training data. To this end, we make the following three key observations.

[3]BTI issues are also a reliability concern for standard DRAM and SRAM memories [67]. However, due to the use of transistors as access devices, the peripheral circuits in DRAM and SRAM can use lower operating voltages ≈ 1.2V. BTI-related reliability issues in DRAM and SRAM are therefore less severe than in NVM contexts [87, 90].

[4]There are works that address run-time management for conventional multi-core systems [81].

- **Observation 1:** *Workload, which includes synaptic weights and their activation on neuromorphic hardware, is specific to the machine learning task being executed and its input.*
- **Observation 2:** *De-stressing all circuits in the hardware periodically, without tracking the actual aging, introduces long latency in spike generation and propagation, which impacts inter-spike interval, leading to information loss in SNNs.*
- **Observation 3:** *Compared to long-term aging under elevated stress conditions, which is permanent and irreversible, short-term aging under nominal conditions is heavily workload-dependent (and hence to some extent controllable), and partially reversible.*

Based on these three observations, we introduce NCRTM, a run-time reliability manager for neuromorphic hardware to de-stress neuron and synapse circuits in the hardware only when needed, by dynamically tracking their short-term aging during the execution of machine learning tasks. NCRTM extends our earlier work DTRO [2] with the following *new* contributions.

- We introduce an intelligent run-time manager **NCRTM**, which improves the long-term reliability of neuromorphic hardware by controlling its short-term aging when executing machine learning tasks.
- We develop a run-time performance monitoring and reliability estimation framework using statistics collected from the neuromorphic hardware.
- We show that NCRTM can be applied to both supervised and unsupervised machine learning approaches and scenarios where the number of training examples are limited.
- We evaluate NCRTM with machine learning workloads designed using Convolution Neural Network (CNN), Multi-layer Perceptron (MLP), and Recurrent Neural Network (RNN) models on a state-of-the-art neuromorphic hardware simulator.

Overall, NCRTM mitigates the aging-related reliability problems in neuromorphic computing by dynamically de-stressing neuron and synapse circuits in response to their short-term aging, with the objective of meeting a reliability target. NCRTM de-stresses these circuits only when it is absolutely necessary to do so, otherwise reducing the performance impact by scheduling all de-stress operations off the critical path by tracking the latency impact of de-stress operations on inter-spike interval (ISI), a key performance measure in SNNs.

2 COMPARISON WITH STATE-OF-THE-ART

Figure 2 illustrates how the proposed approach differs from two reliability-oriented state-of-the-art approaches. Figure 2a illustrates a design-time approach such as RENEU [88], where neurons and synapses are mapped to the hardware to increase the long term, i.e., the lifetime reliability. This approach estimates the aging in neuron and synapse circuits using representative training examples. There are no corrective online measures in place to control the aging, should the aging exceed a critical threshold or the workload behavior changes, for instance, when encountering unseen data at run-time. Figure 2b illustrates a hybrid approach such as DTRO [2], where neurons and synapses are mapped to the hardware using a reliability-oriented mapping technique (e.g., RENEU [88]). Additionally, all neuron and synapse circuits in the hardware are periodically de-stressed to control the aging. The de-stress interval is determined using training examples. The drawbacks of such an approach are the following. First, by not tracking the actual aging in real-time, such approach can introduce significant latency in interrupting normal operation, even when the aging is much below the critical threshold. This is especially critical for SNNs because the performance of machine learning workloads, e.g., their accuracy, depends on the precise times of spikes (see Appendix A). Second, the effectiveness of such a hybrid approach depends heavily on the training data, which may not always be representative. In fact, hybrid approaches present significant limitations for unsupervised applications or applications with limited training data.

(a) Design-time approach (e.g., RENEU [88]). (b) Hybrid approach (e.g., DTRO [2]). (c) Run-time approach (proposed, NCRTM).

Fig. 2. Illustrating the trade-off between neuron aging and SNN performance.

Figure 2c illustrates NCRTM, the proposed run-time approach for reliability management in neuromorphic hardware. NCRTM tracks the aging in neuron and synapse circuits at real-time during the execution of machine learning workloads and de-stresses these circuits only when their aging exceeds a critical threshold. By implementing age tracking and control at run-time, the de-stress decisions of NCRTM are made based on current data. Therefore, NCRTM is relevant for both supervised and unsupervised machine learning approaches.

To the best of our knowledge, NCRTM is the first work for run-time reliability management of neuromorphic hardware. In Section 7, we evaluate NCRTM against these state-of-the-art reliability management approaches using both supervised and unsupervised machine learning workloads.

3 BACKGROUND

In this section, we introduce the background necessary to understand our proposed run-time manager NCRTM. Background on SNNs are provided as Appendix.

3.1 Neuromorphic Hardware

We consider tile-based neuromorphic hardware [6, 17], where the tiles are interconnected using networks-on-chip (NoC) or Segmented Bus [7]. This is similar to several contemporary neuromorphic architectures such as DYNAP-SE [64], Loihi [34], and TrueNorth [35]. Each tile in the hardware consists of a crossbar for synaptic storage, a set of input and output neurons, and a performance monitoring unit, which in its simplest form is a *spike counter* (SC). A crossbar, shown in Figure 3a, is a 3D organization of top electrodes (TEs), which form the rows and bottom electrodes (BEs), which form the columns. A synaptic cell is connected at a crosspoint, i.e., at the intersection of each row and column via an access transistor as shown in Figure 3b. Pre-synaptic neurons are mapped along the TEs and post-synaptic neurons along the BEs. The synaptic weight between a pre- and a post-synaptic neuron is programmed as conductance of the corresponding synaptic cell at the crosspoint. A pre-synaptic neuron's voltage (V_i) applied on the TE is multiplied by the conductance (G_i) to generate current $I_i = V_i \cdot G_i$ (according to Ohm's Law). This current propagates to the post-synaptic neuron to raise its action potential. Current summation occurs on each BE according to Kirchoff's Current Law, when integrating excitation from other pre-synaptic neurons. This implements $\sum_i I_i = \sum_i V_i \cdot G_i$. This is the in-memory multiply accumulate logic implemented inside a crossbar. Figure 3a illustrates the integration of input excitation from two pre-synaptic neurons to one post-synaptic neuron via the synaptic weights w_1 and w_2, respectively. This forms the *data plane* of the neuromorphic hardware. The *control plane* of the hardware consists of control signals, which enable specific access transistors (see Figure 3b) to facilitate current flow in the crossbar. The NVM device of a synaptic cell, shown as a resistive element in Figure 3b, can be implemented for instance with HfO2-based OxRAM or chalcogenide-based PCM as shown in Figure 3c. But our approach is not limited to these specific NVM technologies.

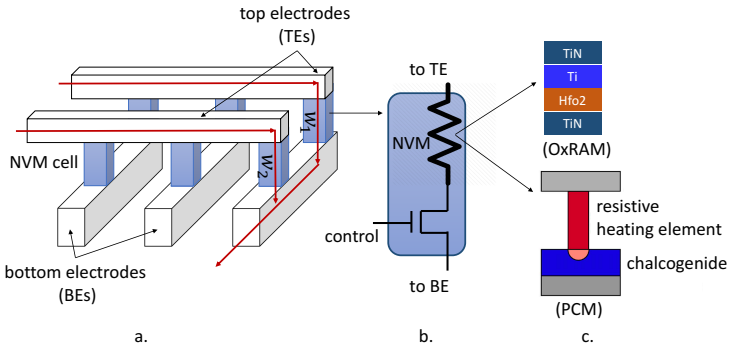
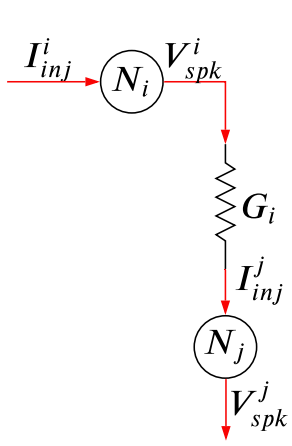
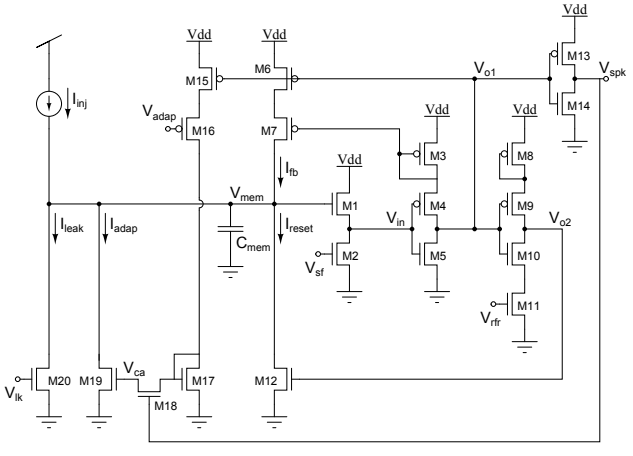


Fig. 3. (a) A crossbar of a neuromorphic hardware, (b) a synaptic cell consisting of a NVM device (a resistive element) and a transistor, and (c) Hfo2-based OxRAM and chalcogenide-based PCM as NVM device.

Figure 4a shows the currents and voltages on the path from the pre-synaptic neuron N_i to the post-synaptic neuron N_j. The input current I^i_{inj} is converted into voltage V^i_{spk} using the neuron N_i. This voltage is multiplied with the conductance G_i (representing synaptic weight) to generate the current I^j_{inj}. This current is converted to voltage V^j_{spk} using neuron N_j. Figure 4b illustrate the internal architecture of a Leaky-Integrate-and-Fire neuron [45]. The current I_{inj} injected into the neuron is proportional to the weighted sum of excitation from all of its pre-synaptic connections. The PMOS and NMOS transistors in the neuron and the reference voltages raise the neuron's membrane voltage. When the voltage crosses a threshold, a spike is generated. The spike voltage (V_{spk}) must be sufficiently high to propagate current through the synaptic cell connected at the output of the neuron.



(a) Current and voltages between the pre-synaptic neuron N_i and the post-synaptic neuron N_j.



(b) Internal architecture of a leaky integrate and fire (LIF) neuron [45], showing different transistors and reference voltages needed to generate the necessary spike voltage at the output.

Fig. 4. a) Connection between neurons and b) a leaky integrate and fire (LIF) neuron [45].

Figure 1: Circuit diagram of the integrate-and-fire neuron.

This design, however, is also undermined by large power consumption, due to the very same problems present in the Axon-Hillock circuit. In [14] van Schaik proposed a circuit with an amplifier at the input, to compare the voltage on the capacitor with a desired spiking threshold voltage. As the input exceeds the spiking threshold, the amplifier drives the inverter, making it switch very rapidly. This cir-

Figure
spike.
tion (1)

for red
put; an
setting
M14,
occurre
M15-M
tion m

Certain NVMs such as PCM, OxRRAM, and FeFET requires high voltages to operate. We consider the case of PCM-based neuromorphic hardware, which requires ≈ 3V [87] to propagate current through it. This high voltage causes aging of the access transistor in each synaptic cell in a crossbar and also of the transistors in each neuron connected along the TEs and the BEs of the crossbar.

3.2 Transistor Aging in Neuromorphic Hardware

High voltage operations of transistors introduce many reliability issues such as Time-Dependent Dielectric Breakdown (TDDB), Bias Temperature Instability (BTI), and Hot-Carrier Injection (HCI). These are the dominant causes of aging in scaled technology nodes (45nm and below) [61]. In older nodes, Electromigration (EM) also plays a key role [23, 25–30, 69, 79].

Transistor aging is accelerated when it is *stressed*, i.e., exposed to high overdrive voltage, where overdrive voltage is defined as the voltage between transistor gate and source (V_{GS}) in excess of the threshold voltage (V_{th}), where V_{th} is the minimum voltage required between gate and source to turn the transistor on. With this understanding, we provide a brief background of these three failure mechanisms.

- *TDDB:* This is a failure mechanism in a CMOS device, when the gate oxide breaks down as a result of long-time application of relatively low electric field (as opposed to immediate breakdown, which is caused by strong electric field) [76]. The lifetime of a CMOS device is measured in terms of its *mean time to failure* (**MTTF**) as

$$\text{MTTF}_{\text{TDDB}} = A.e^{-\gamma\sqrt{V}}, \tag{1}$$

  where A and γ are material-related constants, and V is the overdrive gate voltage of the CMOS device.
- *BTI:* This is a failure mechanism in a CMOS device where positive charges are trapped at the oxide-semiconductor boundary underneath the gate [41]. BTI manifests as 1) decrease in drain current and transconductance, and 2) increase in off current and threshold voltage. The BTI lifetime of the device is

$$\text{MTTF}_{\text{BTI}} = \frac{A}{V^{\gamma}} e^{\frac{E_a}{KT}}, \tag{2}$$

  where A and γ are material-related constants, E_a is the activation energy, K is the Boltzmann constant, T is the temperature, and V is the overdrive gate voltage of the CMOS device.
- *HCI:* This is a failure mechanism in a CMOS device, when a carrier (electron or hole) gains sufficient kinetic energy to overcome the potential barrier of the conducting channel and gets trapped in the gate dielectric, permanently changing its switching characteristic [98].

Unlike the TDDB and BTI failure mechanisms, for which silicon-characterized reliability models are available from foundries, characterized models for HCI failure mechanisms are still in development for scaled nodes. Among these failure mechanisms, BTI is generally accepted as the most important mechanism for sub-10 nm nodes [59, 73, 78]. HCI mostly occurs there under strong voltage/current conditions and TDDB has become less important because technologists have stopped pursuing ultra-high k values in the dielectric.

4 OBSERVATIONS LEADING TO NCRTM

We expand on the three observations made in Section 1.

4.1 Observation 1: Workload-dependent Activation

To illustrate the application and input-dependent neuron activation, Figure 5 plots the spike firing rate of 100 randomly-selected neurons in AlexNet [52], a state-of-the-art CNN used for Imagenet classification. We report results for two randomly-selected training and test images.

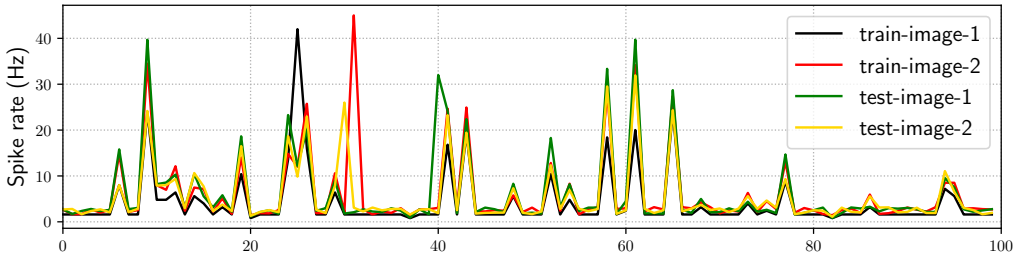


Fig. 5. Spike rate of 100 randomly-selected neurons in AlexNet for 2 training images and 2 test images.

We observe that spike firing rates of neurons depend on the image presented to the AlexNet CNN. Therefore, reliability improvement strategies based on design-time analysis with training examples may not be optimal when they are applied at run-time to process in-field data, a limitation of our prior work [2]. We address this limitation by designing our proposed run-time framework NCRTM, which can adapt its decisions based on current data.

To demonstrate the workload-dependent nature of spike firing rate, Figure 6 plots the minimum, maximum, and average spike rate of all neurons in 10 machine learning workloads (see Section 6) for test examples from their respective dataset. We observe that spike rates of neurons are strongly workload-dependent, and therefore, a workload-specific strategy is needed to optimally control the reliability aspect. This is precisely the objective of NCRTM.

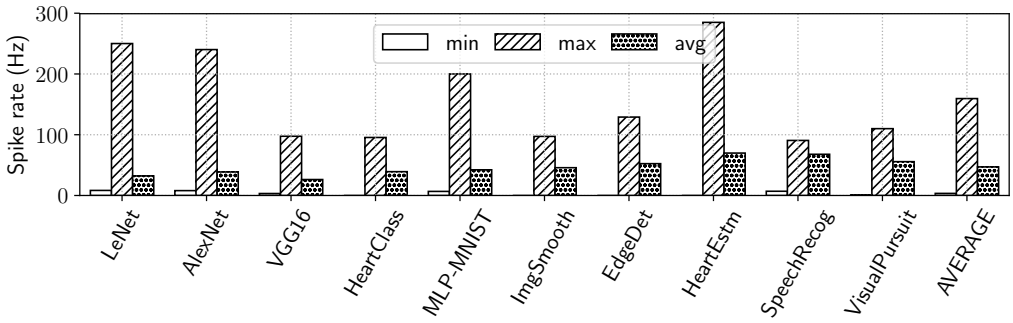
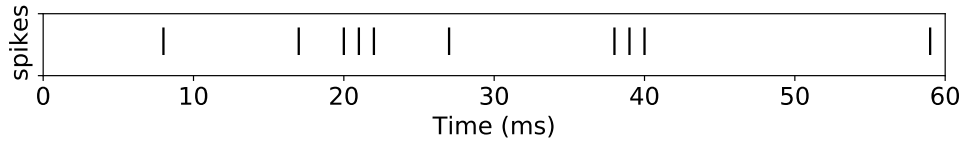


Fig. 6. Minimum, maximum, and average spike rate of all neurons in 10 workloads for test set.

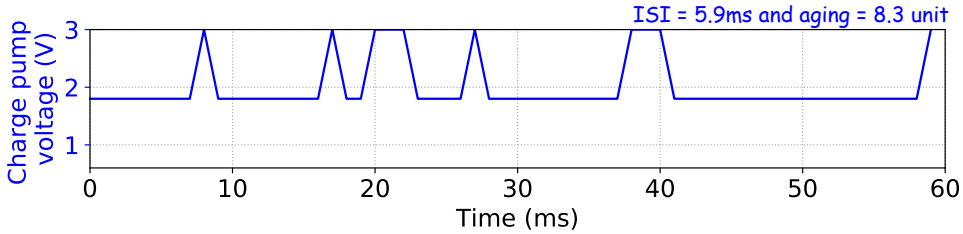
4.2 Observation 2: Performance Trade-off in Reliability Improvement

In SNNs, information is encoded in spike times. Inter-spike interval (ISI) defines the performance metric in SNNs. To demonstrate how ISI is impacted by reliability-oriented decisions, Figure 7(a) shows the spike train generated by a neuron in AlexNet when processing a reference image. Each spike injects current into the crossbar to flow through the NVM cell. Figure 7(b) illustrates the voltage of the on-chip charge pump that supply the reference voltages in the neuron to generate this spike train. The charge pump is operated at $1.8V$ for the entire 60ms interval, boosting its voltage to $3V$ only to generate spikes. Aging of the transistors in the neuron is 8.3 units (see Section 5.3 for aging computation) and the average ISI is 5.9ms (See Section 5.4 for ISI computation). Figure 7(c) illustrates the charge pump's operating voltage when it is discharged to $1.2V$ after generating every spike and boosted again to $1.8V$ before generating the next. This is to de-stress the transistors in the neuron. Once de-stressed, the neuron becomes unavailable to generate spikes, introducing latency

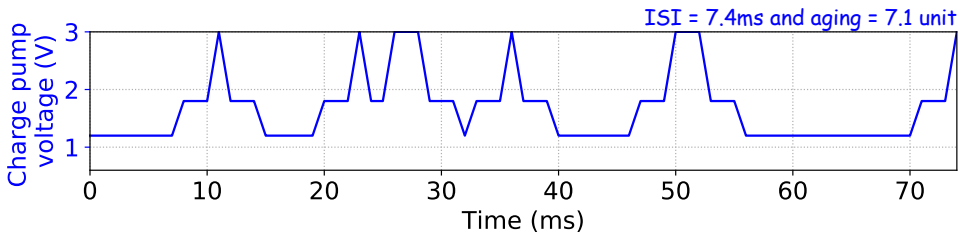
in the spike train. The average ISI increases to 7.4ms, compared to 5.9ms in Figure 7(b). Changes in ISI may lead to accuracy loss. Frequently discharging the charge pump, however, reduces the neuron's aging to 7.1 units, compared to 8.3 units in Figure 7(b). This reduction in aging leads to an improvement of MTTF, i.e., the lifetime of the neuron.



(a) Example spike train from a neuron in AlexNet.



(b) Charge pump voltage to process the spike train.



(c) Charge pump reset to 1.2V after processing every spike.

Fig. 7. Trade-off between neuron aging and SNN performance.

Although it is possible to estimate the worst-case ISI degradation at compile-time using SpiNeMap [9] and other similar approaches, such estimation can deviate significantly from the actual case in a highly dynamic environment, where testing data is different from the training examples (see Figure 5). Therefore, a run-time manager is desirable to dynamically adapt the reliability-oriented decisions to limit ISI degradation.

4.3 Observation 3: Short-term vs. Long-term Aging

In this work, we demonstrate our approach for BTI-related failures. The general principle applies to any other failure mechanism. BTI aging manifests as: 1) A decrease in drain current and transconductance, and 2) An increase in off current and threshold voltage. When operated at a high voltage and temperature, these parameters strongly drift from their nominal values. In fact, in scaled technology nodes, this BTI aging happens even under nominal conditions and from the very start of using the devices leading to the so-called soft breakdown [33, 50, 51, 99].

Recent works such as [41, 42, 50, 51, 70, 77, 99] suggest that BTI is the collective response of two independent defects – the *as-grown hole traps* (AHTs) and *generated defects* (GDs). AHTs and a small proportion of GDs can be recovered by annealing at high temperatures if the BTI stress voltage is removed (*de-stress*). Figure 8 illustrates the stress and recovery of the threshold voltage of a CMOS transistor on application of a high (V_{spk}) and a low voltage (V_{idle}). We observe that both stress and recovery depends on the time of exposure to the corresponding voltage level. This implies that when a neuron is idle, the BTI aging of the neuron recovers from stress.

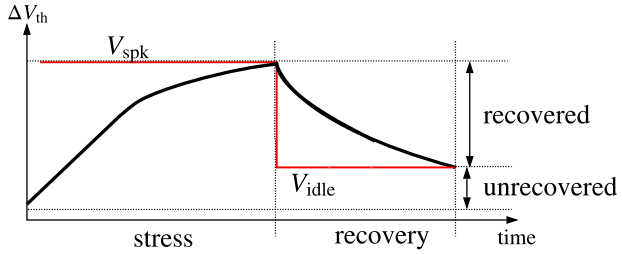


Fig. 8. Demonstration of degradation due to BTI.

Figure 9a shows the shift in threshold voltage of a NMOS transistor in a neuron with a constant firing rate of 50Hz and the neuron circuit de-stressed once every second (see Section 6 for the simulation setup). Figure 9b shows the results using the same setup, but with the neuron circuit de-stressed once every 100ms. As this figure clearly shows, with longer de-stress interval (e.g., once every second), the transistor aging becomes irreversible. Therefore, the shift in threshold voltage of the transistor is higher than the case with shorter de-stress interval (e.g., once every 100ms).

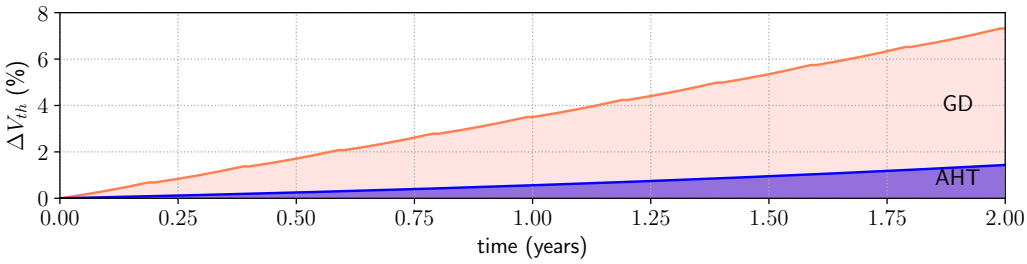
5 RUN-TIME MANAGER FOR NEUROMORPHIC COMPUTING (NCRTM)

5.1 A Motivating Example Showing the Need for Run-time Reliability Management

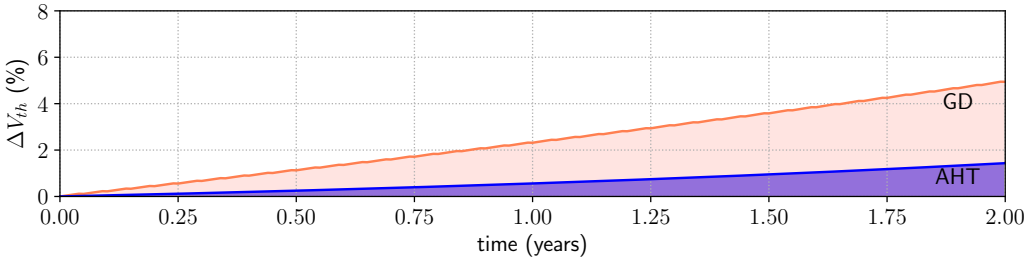
Figure 10 shows an example where four spikes (R1, R2, R3, & R4) are generated from a neuron. These spikes are generated with some idle time between them (based on its input excitation). The figure illustrates the hybrid approach DTRO [2] (see Fig. 2b), where the neuron is de-stressed at run-time after generating 3 spikes. This fixed number is decided statically, considering the neuron's activation in some training example. This is illustrated in the top right corner of the figure, where we observe that the BTI aging ($\mathcal{A}_{BTI}$) exceeds the aging threshold of 10 units after generating three spikes based on the idle periods between spikes in the training example.

Using this static approach, the de-stress operation is initiated upon generating R3, which delays the generation of R4 due to the non-zero latency of the de-stress operation. This causes a change in the ISI, which may lead to performance loss in SNNs (see Appendix B). At the time when the neuron circuit is being de-stressed, the *BTI aging is below the threshold* because a CMOS transistor recovers partially from BTI stress when idle. The length of the idle periods at run-time can be different from those used at design-time when the analysis is performed as shown in this example. Therefore, the static approach will unnecessarily introduce performance penalty in such a situation. Using fixed interval for de-stress (instead of counting the spikes) will also lead to a similar situation because the number of spikes within the de-stress interval still remains unknown at run-time, being dependent on the input excitation.

Figure 10 also shows NCRTM, our dynamic reliability management policy, where the de-stress operation of the neuron circuit is initiated by tracking its aging. NCRTM can generate the spike R4 because the aging of the neuron is lower than the aging threshold at the time of generating the spike. This is because NCRTM models both the stress and recovery of circuit aging at run-time.



(a) BTI aging when de-stressing a neuron circuit once every second.



(b) BTI aging when de-stressing a neuron circuit once every 100ms.

Fig. 9. Shift in threshold voltage of a NMOS transistor in a neuron with a constant firing rate of 50Hz and the neuron circuit de-stressed a) once every second (long-term) and b) once every 100ms (short-term).

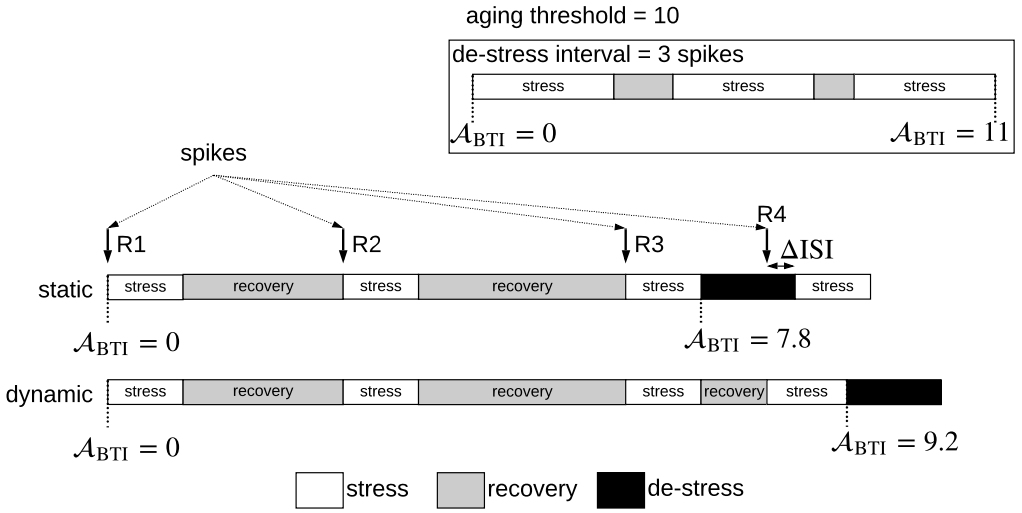


Fig. 10. Comparing static vs. dynamic reliability management policy.

There is no change in ISI. Therefore, NCRTM is better than the static approach both in terms of reliability and performance.

This example demonstrates one scenario with sparse neuron activation. One can also imagine a counter scenario where the neuron is activated too frequently. In this case, the static policy can lead violating the critical threshold because it cannot adapt the de-stress interval at run-time. NCRTM can adjust its de-stress interval at run-time by tracking the aging (both stress and recover). In Section 7.4 we show only a marginal performance impact for workloads with frequent activation. Therefore, NCRTM is better than the static policy, when it comes to managing workload-specific circuit aging.

5.2 High-level Overview

Based on the three observations in Section 4 and the motivating example in Section 5.1, we introduce NCRTM, a run-time reliability manager for neuromorphic hardware. Figure 11 illustrates NCRTM designed for tile-based neuromorphic hardware. We show the architecture of DYNAP-SE with 12 tiles, numbered $1, 2, \cdots, 12$ in the figure. These tiles are connected hierarchically, with groups of 4 tiles connected to a local router R. The local routers can be interconnected via global routers (not shown in the figure) to facilitate spike communication between any two tiles. The figure also shows the on-chip charge pump and the voltage delivery network, which supply the reference voltages for the neuron circuits of each tile.

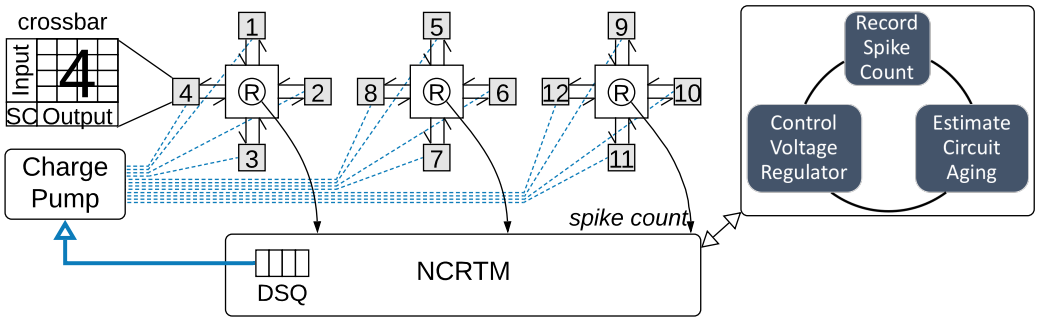


Fig. 11. Overview of NCRTM for tile-based neuromorphic hardware.

NCRTM is implemented as a controller to mitigating the aging of neuron circuits in each tile. To do so, NCRTM estimates maximum aging of the neurons in each tile by recording the number of spikes within a time window (see our aging formulation in Section 5.3). If the aging of a tile exceeds a threshold (th_a),[5] NCRTM schedules the de-stress of the tile by making an entry in the de-stress queue (DSQ). However, the tile may not be de-stressed immediately. NCRTM de-stresses a tile opportunistically by estimating the change in ISI (called ISI distortion) that may result due to offlining the tile (see our ISI formulation in Section 5.4). During de-stresses, a very low voltage is applied to all the neurons in a tile for a time duration $tDSC$ (discharge cycle time). This allows the transistors in the neurons to reverse the threshold voltage drift ΔV_{th}. The recovery time $tDSC$ is modeled using the framework presented in [100].

Fundamental to the aging and ISI computation in NCRTM is a technique to estimate the number of spikes for each neuron. The spike counter (SC) in each tile can facilitate counting the spikes in a time interval. However, not all neuromorphic hardware is equipped with SC. Therefore, we present an alternative software-based technique for implementing spike counting.

[5]The aging threshold th_a is a user-defined parameter used to achieve a given reliability target.

network and 400 billion SOPS per watt for networks with high spike rates and high number of active synapses, whereas todays most energy-efficient supercomputer achieves 4.5 billion floating-point operations per second (FLOPS) per watt [25]. Although the metric units are different, the computational capability can by some means be indicated by the number of operations per second.

2.2.3 Synapse communications

This project is researching the communication mechanisms and architecture of neuromorphic computing. In this section, some existent and conventional protocols and architectures are introduced as the basis of this research.

Address-event representation protocol

Address-event representation (AER) is a communication protocol originally proposed as a method to communicate sparse neural events between neuromorphic chips. Neuron's internal connections among individual neurons or neuron clusters are allocated to the reduced number of channels by time division multiplexing. According to the protocol, each spike is represented by its location and spiking time.

5.2.1 Spike Counting in Software. To understand spike counting in software, we describe the spike communication mechanism in neuromorphic hardware. Spikes from the post-synaptic neurons in a tile are converted into an address-event representation (AER) and broadcasted on the interconnect via the AER encoder. Figure 12 shows an example explaining the principles behind AER. Here, four neurons in a tile spike at time 3, 0, 1 and 2 time units, respectively. The encoder encodes these four spikes in order to be communicated on the interconnect.

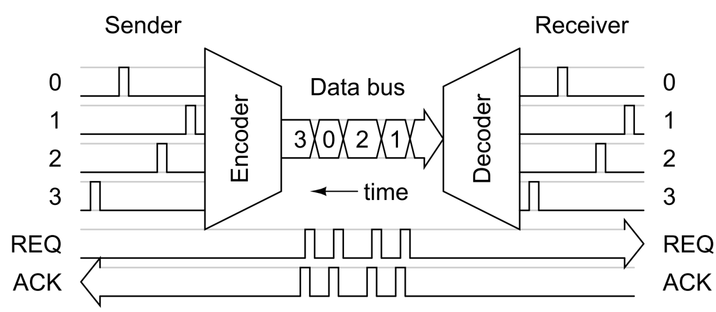


Figure 2.4: A example of AER protocol [2].

Fig. 12. An example AER protocol.

We propose to count the spikes from each neuron by snooping on the interconnect. We implement counters, one for each neuron of the hardware. When a de-stress operation is initiated for a tile, all the counters for the neurons in the tile are reset to start counting the spikes for the next interval. The total storage overhead needed for implementing software-based spike counting for the 12-tile architecture of Figure 11 is $12 * 128 * 16$ bits = 24Kb, with 128 post-synaptic neurons per tile. However, continuous snooping on the bus can introduce performance overhead. In the future, we plan to extend the interconnect routers to facilitate recording the spike packets. This will allow NCRTM to poll these readings periodically. With this necessary background, we now introduce our model for estimating aging (Section 5.3) and ISI (Section 5.4).

5.3 Aging Computation

Equation 2 equates the MTTF of a CMOS transistor for a given overdrive voltage. BTI failures can also be modeled using the Weibull distribution with a scale parameter α and a slope parameter β. Reliability at time t can be written as

$$R(t) = e^{-\left(\frac{t}{\alpha(V)}\right)^{\beta}}, \tag{3}$$

with the corresponding MTTF computed as

$$MTTF = \int_0^{\infty} R(t)dt = \alpha(V)\Gamma\left(1 + \frac{1}{\beta}\right), \tag{4}$$

where Γ is the Gamma function. Using the expressions for MTTF from Equations 2 and 3, and rearranging, we obtain the expression for the scale parameter α as

$$\alpha(V) = \frac{\frac{A}{V^{\gamma}} e^{\frac{E_a}{KT}}}{\Gamma\left(1 + \frac{1}{\beta}\right)}. \tag{5}$$

The aging ($\mathcal{A}$), i.e., the degradation of the CMOS transistor can be expressed as

$$\mathcal{A} = \sum_{i=1}^{n} \frac{\Delta t_i}{\alpha(V_i)}, \text{ such that } R(t_s) = e^{-(\mathcal{A})^{\beta}}, \tag{6}$$

where the scaling factor $\alpha(V_i)$ can be calculated using Eq. 5.

We note that a neuron suffers aging when generating a spike. Each spike is of fixed voltage V_{spk} (see Figure 4b) and a fixed time duration to the order of few ms. Therefore, both V_i and t_i in Equation 6 are constant. This allows us to express the aging formulation as

$$\mathcal{A} = n \cdot \frac{\Delta t}{\alpha(V)}, \tag{7}$$

where n is the number of spikes generated by the neuron, Δt is the fixed spike duration, and V is the fixed spike voltage. Equation 7 allows us to represent the aging in terms of the number of spikes generated by a neuron and the unit aging parameter $\frac{\Delta t}{\alpha(V)}$, which represents the aging per spike. This simplified formulation allows to estimate the aging in each neuron by simply counting the number of spikes it generates. Hence the performance overhead can be kept negligible.

5.4 ISI Computation

To define ISI, we consider a tile consisting of N post-synaptic neurons and a finite interval of time $[0, T]$ for which the tile is active without undergoing a de-stress operation. The post-synaptic neurons generate K spikes in this interval, which are organized based on their generation time and the source neuron as

$$\{t_1^1, t_2^1, \cdots, t_{k_1}^1\}, \{t_1^2, t_2^2, \cdots, t_{k_2}^2\}, \cdots, \{t_1^N, t_2^N, \cdots, t_{k_N}^N\}, \tag{8}$$

where t_i^n is the time of the i^{th} spike generated by the n^{th} neuron and $K = \sum_{i=1}^{N} k_i$. The instantaneous ISI of the spike train from the n^{th} neuron is [43]

$$ISI_{\text{inst}}^n(i) = t_i^n - t_{i-1}^n \tag{9}$$

To estimate the impact of de-stress operation on ISI, we compute two statistics for each neuron – the instantaneous ISI (Equation 9) and the average ISI, which is computed as the average of all ISIs for a neuron. Using $tDSC$ as the time to de-stress a tile, the change in instantaneous and average ISI of the n^{th} neuron are

$$\Delta ISI_{\text{inst}}^n(i) = ISI_{\text{inst}}^n(i) + tDSC \text{ and } \Delta ISI_{\text{avg}}^n = tDSC/k_N \tag{10}$$

6 EVALUATION METHODOLOGY

Figure 13 illustrates our simulation framework. An SNN-based application is simulated using `CARLsim` [20], a GPU accelerated SNN simulator used to train and test SNN models. CARLsim reports spike times for every synapse in the SNN. The spike times are used to perform mapping explorations optimizing some objective, such as performance (PyCARL [8]) and reliability (RENEU [88]). We use `NeuroXplorer` [12], a cycle-accurate simulator of neuromorphic hardware such as DYNAP-SE [64]. The neuron and synapse mapping obtained using the mapping exploration framework is applied to `NeuroXplorer` to perform cycle-accurate simulation of the application on the hardware model, using current data. NCRTM is implemented inside `NeuroXplorer` to estimate the circuit aging and control it by de-stressing the circuit when the aging exceeds a threshold. The change in spike latency due to the de-stress operation can be precisely modeled in `NeuroXplorer`, as shown in [12].

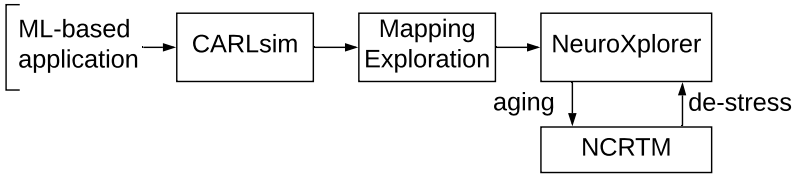


Fig. 13. Our simulation framework.

All simulations are conducted on a system with 8 CPUs, 32GB RAM, and NVIDIA Tesla GPU, running Ubuntu 16.04.

6.1 Evaluated Applications

We evaluated 10 machine learning applications that are representative of three most commonly used neural network classes — convolutional neural network (CNN), multi-layer perceptron (MLP), and recurrent neural network (RNN). These applications are 1) LeNet [56] based handwritten digit recognition with 28×28 images of handwritten digits from the MNIST dataset [37]; 2) AlexNet [52] for Imagenet classification [36]; 3) VGG16 [82], also for Imagenet classification [36]; 4) ECG-based heart-beat classification (HeartClass) [5, 31] using electrocardiogram (ECG) data from the Physionet database [63]; 5) multi-layer perceptron (MLP)-based handwritten digit recognition (MLP-MNIST) [38] using the MNIST database; 6) image smoothing (ImgSmooth) [20] on 64×64 images; 7) edge detection (EdgeDet) [20] on 64×64 images using difference-of-Gaussian; 8) heart-rate estimation (HeartEstm) [22] using ECG data; 9) gender classification using speech data (SpeechRecog) [39]; and 10) RNN-based predictive visual pursuit (VisualPursuit) [49]. The former 7 are supervised applications, while the latter 3 are unsupervised applications. Table 1 summarizes the topology, the number of neurons and synapses of these applications, and their baseline accuracy on DYNAP-SE using PyCARL [8].[6]

Table 1. Applications used to evaluate our approach NCRTM.

Class	Applications	Synapses	Neurons	Topology	Accuracy
CNN	LeNet [56]	159,553	5,576	CNN	94.08%
	AlexNet [52]	1,029,286	650,000	CNN	71.7%
	VGG16 [82]	2,136,560	18,472	CNN	91.62 %
	HeartClass [5]	2,396,521	24,732	CNN	85.12%
MLP	MLP-MNIST [38]	79,400	984	FeedForward (784, 100, 10)	95.5%
	EdgeDet [20]	272,628	1,372	FeedForward (4096, 1024, 1024, 1024)	100%
	ImgSmooth [20]	136,314	980	FeedForward (4096, 1024)	100%
RNN	HeartEstm [22]	636,578	6,952	Recurrent Reservoir	99.2%
	SpeechRecog [39]	39,056	683	Recurrent Reservoir	96.8%
	VisualPursuit [49]	3,25,710	5,717	Recurrent Reservoir	89.0%

6.2 Hardware Models

In our cycle-accurate simulator, we model the architecture of the DYNAP-SE neuromorphic hardware [64] with the following configurations.

- A tiled array, with each tile accommodating 128 input and 128 output neurons. There are 65,536 crosspoints in each tile.
- Spikes are digitized and communicated between cores through a mesh routing network using the Address Event Representation (AER) protocol.

The DYNAP-SE platform uses static random access memory (SRAM) to implement the synaptic cells in each crossbar. However, in our simulator, we use Phase-Change Memory (PCM) as the synaptic element. Table 2 reports the major hardware parameters.

In the future, we will demonstrate NCRTM on a real NVM-based silicon neuromorphic system.

[6]The CNN models LeNet, AlexNet, and VGG16 are converted to spiking domain using our previously proposed converter [5]. For the inference performance of the original model, readers are referred to [74].

Table 2. Major simulation parameters extracted from [64] and extrapolated for PCM technology.

Neuron technology	65nm CMOS
Synapse technology	PCM
Supply voltage	1.0V
Energy per spike	50pJ at 30Hz spike frequency
Energy per routing	147pJ
Switch bandwidth	1.8G. Events/s

6.3 Evaluated State-of-the-art Techniques

We evaluate the following three approaches.

- **PyCARL [8]:** This is a performance-oriented approach to map SNN-based applications to neuromorphic hardware. This approach first generates clusters of neurons and synapses, where each cluster can fit on to the resources of a tile in the hardware. Then it uses an optimization algorithm to place these clusters to the hardware, maximizing performance of the machine learning application on the hardware. CMOS circuits are not de-stressed at run-time.
- **RENEU [88]:** This is a reliability-oriented approach to map SNN-based applications to neuromorphic hardware. This approach also generates clusters of neurons and synapses from an application, but maps the clusters to the hardware minimizing the maximum aging while considering only the training data. CMOS circuits are not de-stressed at run-time.
- **DTRO [2]:** This is a reliability-oriented approach where neuron and synapse circuits of the neuromorphic hardware are de-stressed at run-time at fixed interval. This interval is decided based on analysis performed at design-time using training data.

6.4 Evaluated Metric

We evaluate the following metrics.

- **Aging:** This is the maximum circuit aging in DYNAP-SE for each machine learning workload.
- **ISI:** This is average ISI of each machine learning workload.
- **Application Performance (accuracy):** The performance, e.g., accuracy is defined in terms of misclassification rate for image-based CNN and MLP applications. For RNN applications that use time-series data, performance is measured in terms of error rate [22].
- **Aging per unit ISI distortion:** This is an unified metric reporting the aging per unit ISI distortion for each workload, defined as

$$\text{aging per unit ISI distortion} = \mathcal{A}/\Delta ISI \tag{11}$$

In formulating the optimization objective of Equation 11, NCRTM aims to optimize (i.e., minimize) circuit aging $\mathcal{A}$ for a given constraint on the ISI distortion. In our earlier works [9, 32], we have shown the dependency of application performance, e.g., accuracy on ISI due to inter-spike interval-based information encoding in SNNs. Therefore, any distortion in ISI may lead to a reduction in performance [9, 32]. Correspondingly, the above optimization problem essentially reduces to minimizing the aging $\mathcal{A}$ for a given constraint on SNN accuracy.

6.5 Aging Parameters

To compute aging, the slope parameter of Weibull distribution is set to $\beta = 2$, and the operating temperature is set to $300K$. Other fitting parameters are adjusted to achieve an MTTF of 2 years in the baseline system (PyCARL), corresponding to a threshold voltage shift of 10%. This is what

is typically accepted by technologists as the maximum allowed degradation before timing errors begin to appear.

7 RESULTS AND DISCUSSION

7.1 Summary of Results

Table 3 summarizes the key results.

Table 3. Summary of results.

NCRTM	Aging (Sec. 7.2)	ΔV_{th} (Sec. 7.3)	Δ ISI (Sec. 7.4)	Accuracy (Sec. 7.5)
vs. PyCARL [8]	74%↓	52.0%↓	12%↑	4.52%↓
vs. RENEU [88]	73%↓	50.7%↓	11%↑	4.52%↓
vs. DTRO [2]	60%↓	31.4%↓	2%↑	0.76%↓

7.2 Circuit Aging

Figure 14 plots the aging of the neuron and synapse circuits in DYNAP-SE during the execution of the machine learning applications for each evaluated approach, normalized to PyCARL. We make the following three key observations.

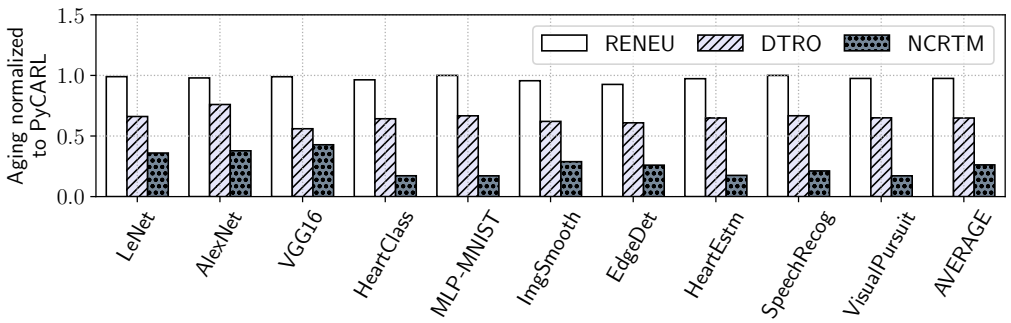


Fig. 14. Aging at 300K normalized to PyCARL.

First, the aging due to RENEU is lower than PyCARL by an average of 2.5%. This improvement is due to the aging-aware neuron and synapse mapping policy of RENEU, which balances the aging of all tiles in the hardware. PyCARL, which balances the utilization of the tiles in the hardware, has higher aging. However, both PyCARL and RENEU are design-time based policies, i.e., they do not make any run-time decisions. Second, DTRO is a hybrid approach, which uses the neuron and synapse mapping of RENEU. Compared to PyCARL and RENEU, DTRO de-stresses all circuits in the hardware periodically at run-time. The de-stress interval is determined at design-time by analyzing the training data. The aging of DTRO is therefore lower than both PyCARL (average 35% lower) and RENEU (average 33.5% lower). Third, NCRTM, which is a run-time approach, has the lowest aging of all. The average aging of NCRTM is 74% lower than PyCARL, 73% lower than RENEU, and 60% lower than DTRO. The improvement of NCRTM is due to the precise tracking of aging at run-time using current data, to achieve a target MTTF.

7.3 Threshold Voltage Shift

Circuit aging manifests as shift in threshold voltage. Figure 15 plots the shift in threshold voltage (ΔV_{th}) in DYNAP-SE after executing each machine learning application continuously till the end of its lifetime of 2 years. We normalize the results so that the threshold voltage shift due to PyCARL is 10%. We make the following key observations.

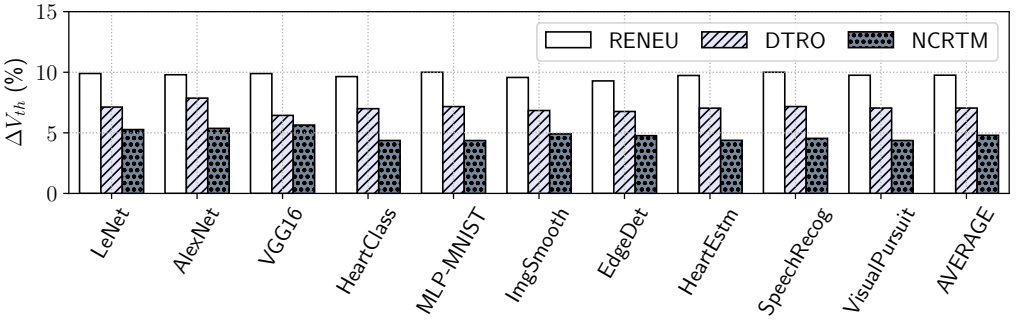


Fig. 15. Threshold voltage shift after 2 years of continuous operation.

We observe that, compared to PyCARL, the average threshold voltage shift when using RENEU is 9.75%, DTRO is 7.0%, and NCRTM is only 4.8%. The threshold voltage shift is the lowest in NCRTM because the aging of NCRTM is lowest of all the approaches, which we reported in Section 7.2. Increase in threshold voltage results in the reduction in drive current, which in turn results in temporal performance degradation of neuron and synapse circuits in the neuromorphic hardware.

7.4 Change in ISI

Figure 16 plots the ISI of the machine learning workloads on DYNAP-SE for each evaluated approach, normalized to PyCARL. We make the following five key observations.

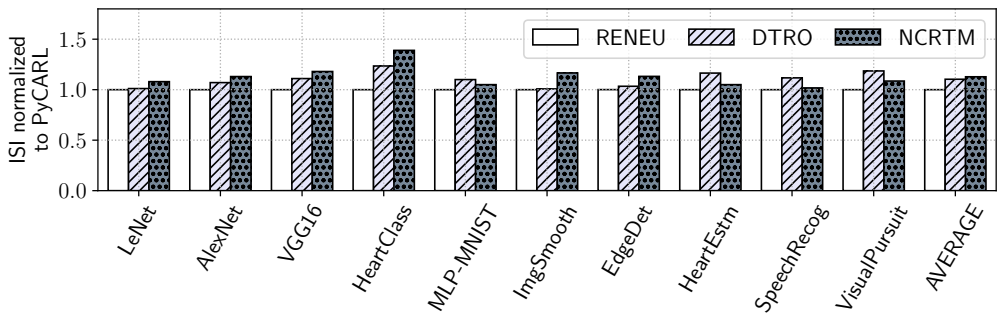


Fig. 16. ISI at 300K normalized to PyCARL.

First, the ISI obtained with RENEU is similar to PyCARL. This is because RENEU generates a mapping of the workload to the hardware, which improves reliability without significantly hurting the performance. Since no run-time decisions are made in both these approaches, their performance at run-time are therefore similar. Second, the ISI obtained with DTRO is higher than RENEU by an average of 10%. This increase is because DTRO make run-time decision of de-stressing the neuron and synapse circuits periodically to control their aging. This leads to increase in latency, which

increases the average ISI (see Equation 10). The advantage in this case is lower aging, which we analyzed in Section 7.2, leading to a lower drift of the threshold voltage (see Section 7.3). Third, the ISI using NCRTM is higher than RENEU by an average of 12%. This increase is due to the run-time de-stresses in NCRTM (similar to DTRO), which introduces latency, impacting the ISI. The ISI using NCRTM is only 2% higher than DTRO. This increase is because NCRTM never allows the aging to reach critical levels and therefore, schedules more de-stresses by precisely tracking it at run-time. However, due to NCRTM's policy to schedule the de-stresses by tracking their latency impact on ISI, NCRTM ensures only marginal change in ISI. ISI change may lead to accuracy impact, which is discussed in Section 7.5. Fourth, the ISI of NCRTM is lower than DTRO for MLP-MNIST. This is because for this application, the circuit aging is generally lower due to the sparsity of spike generation in the workload. So, the BTI stress is recovered in the idle period. DTRO cannot track this recovery and therefore, applies a conservative control, unnecessarily constraining the performance. Finally, for the three unsupervised applications (HeartEstm, SpeechRecog, and VisualPursuit), the ISI using NCRTM is on average 10% lower than DTRO. This is because in the absence of training data for these applications, DTRO applies a conservative policy to de-stress neuron and synapse circuits frequently to prevent their aging from reaching a critical value. NCRTM, on the other hand, tracks the aging at run-time based on the data that these models encounter and de-stress the circuits, only when needed.

*We **conclude** that for machine learning workloads with sparse activation, NCRTM is significantly better than design-time based approaches both in terms of reliability and performance. For dense activation, NCRTM improves reliability significantly compared to these approaches, with marginal impact on performance. Furthermore, NCRTM outperforms any design-time based policy, when the availability of training data is limited.*

7.5 Application Accuracy

Change in ISI manifests as loss in accucay of a machine learning workload. Table 4 reports the accuracy of each machine learning workload on DYNAP-SE using the evaluated approaches. We make the following three key observations. First, the accuracy of RENEU and PyCARL are the same. This is because RENEU maps neurons and synapses of an SNN workload to the hardware resources statically to minimize the aging. It does so, ensuring that the spike communication latency on the interconnect does not induce any change in ISI compared to that obtained using PyCARL. Second, the accuracy of NCRTM is on average 4.52% lower than PyCARL and RENEU, and 0.76% lower than DTRO. This reduction in accuracy is a direct result of the change in ISI, which we analyzed in Section 7.4. Third, although NCRTM results in 4.8% lower accuracy than Baseline for AlexNet (71.7% Baseline accuracy compared to 68.2% accuracy using NCRTM), it reduces circuit aging by 62% compared to Baseline (See Sec. 7.2).

Table 4. Application Accuracy.

Application	Accuracy			Application	Accuracy		
	RENEU/PyCARL	DTRO	NCRTM		RENEU/PyCARL	DTRO	NCRTM
LeNet	94.08%	91.7%	90.6%	AlexNet	71.7%	69.3%	68.2%
VGG16	91.62%	90.2%	90.1%	HeartClass	85.12%	80.7%	80.1%
MLP-MNIST	95.5%	90.1%	90.1%	ImgSmooth	100%	99%	99%
EdgeDet	100%	96%	95%	HeartEstm	99.1%	92.6%	90.0%
SpeechRecog	96.8%	94.0%	94.1%	VisualPursuit	89%	81.5%	81.5%

7.6 Platform Exploration

Figure 17 illustrates the reliability impact of increasing the number of tiles in a neuromorphic hardware. The figure plots the aging results of NCRTM on DYNAP-SE with 16 and 32 tiles, normalized to the aging on DYNAP-SE with the baseline configuration of 12 tiles. We observe that the average aging with 16 and 32 tiles is 18% and 51% lower than the aging with baseline configuration of 12 tiles, respectively. Circuit aging is lower with more number of tiles. This is because with more tiles in the hardware, fewer neurons and synapses are mapped to each tile. Therefore, each tile of the hardware generates fewer spikes, which lowers the aging of its neurons and synapse circuits.

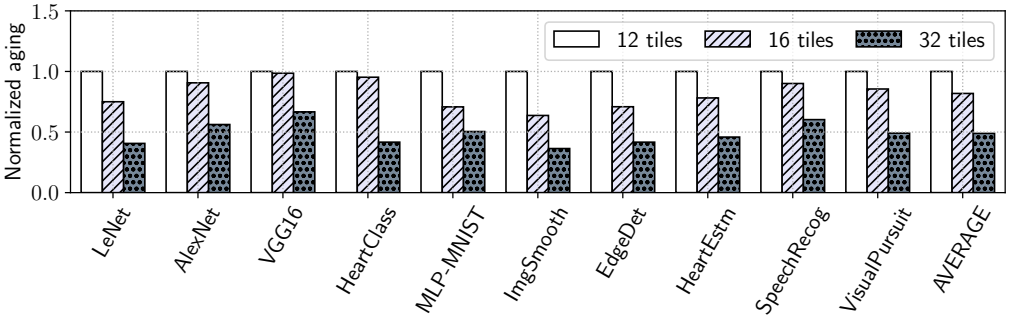


Fig. 17. Aging of DYNAP-SE with 16 and 32 tiles normalized to the aging with 12 tiles.

7.7 Temperature Dependency of Reliability

Figure 18 illustrates the temperature dependency of the aging in a neuromorphic hardware. We report the aging results of NCRTM at two elevated temperatures, 320K and 340K, for each of our machine learning applications. Aging results are normalized to NCRTM at 300K. We observe that aging increases with an increase in temperature. Aging at 320K and 340K is higher than the aging at 300K by an average of 7% and 30%, respectively. This is due to the exponential dependency of circuit aging on temperature (Equation 6). We also observe from this equation that aging also depends on the voltage needed to operate the neurons and synapses in the hardware when generating and propagating spikes. Therefore, VGG16, ImgSmooth, and VisualPursuit, which have more spikes, have higher aging at the elevated temperatures than all other applications. Higher aging leads to higher threshold voltage shift in the transistors.

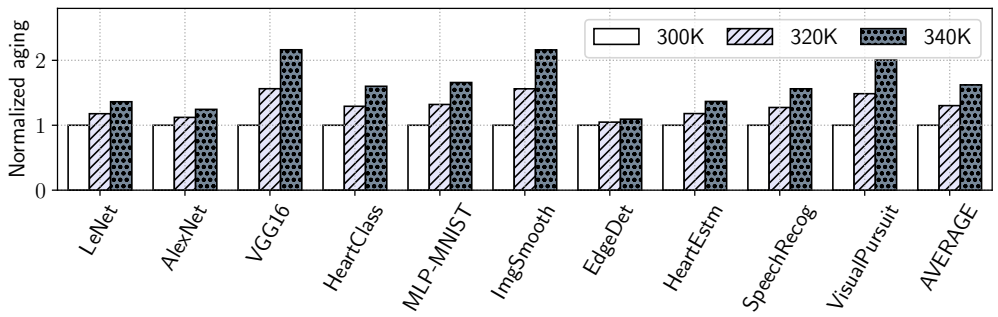


Fig. 18. Aging at 320K and 340K normalized to the aging at 300K.

7.8 Aging Per Unit ISI Distortion

To unify the ISI distortion and aging results in one metric, Figure 19 reports the aging per unit distortion of Equation 11 for the design-time based RENEU and the run-time based NCRTM. We observe that NCRTM has an average 58% lower aging per unit ISI distortion than RENEU. The improvement of NCRTM is what we have analyzed before. This result shows that *for the same amount of ISI distortion (i.e., performance impact), NCRTM will lead to significantly lower circuit aging than RENEU.*

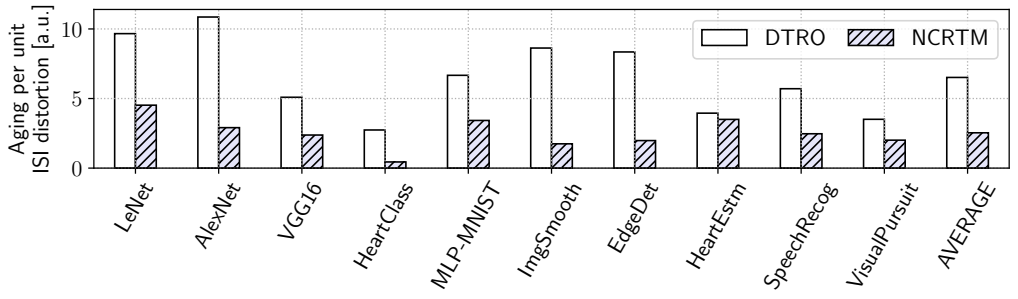


Fig. 19. Aging per unit ISI distortion (lower is better).

8 CONCLUSIONS

This paper introduces NCRTM, a run-time reliability manager for neuromorphic computing. We observe that neurons and synapses in neuromorphic hardware are exposed to high voltages and/or currents because of the operating requirements of the Non-Volatile Memory, which are used for high density and low energy synaptic storage in the hardware. When exposed to these elevated conditions for too often, the CMOS transistors in the neuron and synapse circuit suffer strong aging, leading to hard breakdown. But in strongly scaled sub-10nm technology nodes, even under normal workloads, parametric soft breakdown mechanisms will start drifting the transistor parameters from their nominal values. In contrast to long-term aging, which permanently damages the hardware, short-term aging in scaled CMOS transistors is mostly due to BTI. The latter is heavily workload-dependent and more importantly, partially reversible. Based on these observations, NCRTM dynamically de-stresses neuron and synapse circuits in response to the short-term aging in their CMOS transistors during the execution of machine learning tasks, with the objective of meeting a reliability target. NCRTM de-stresses these circuits only when it is absolutely necessary to do so, otherwise reducing the performance impact by scheduling de-stress operations off the critical path. We evaluate NCRTM with supervised and unsupervised machine learning applications on a neuromorphic hardware. Our results demonstrate that that for machine learning workloads with sparse activation, NCRTM is significantly better than design-time based approaches both in terms of reliability and performance. For dense activation, NCRTM improves reliability significantly compared to these approaches, with only marginal impact on performance. We conclude that NCRTM can be easily extended to incorporate other failure mechanisms. In the future, we plan on implementing NCRTM on NVM-based DYNAP-SE, when such board will be made publicly available.

9 ACKNOWLEDGMENTS

This work is supported by the National Science Foundation Faculty Early Career Development Award CCF-1942697 (CAREER: Facilitating Dependable Neuromorphic Computing: Vision, Architecture, and Impact on Programmability).

REFERENCES

[1] M. R. Azghadi, B. Linares-Barranco, D. Abbott, and P. H. W. Leong, "A hybrid CMOS-memristor neuromorphic synapse," *TBCAS*, 2016.

[2] A. Balaji, S. Song, A. Das, N. Dutt, J. Krichmar, N. Kandasamy, and F. Catthoor, "A framework to explore workload-specific performance and lifetime trade-offs in neuromorphic computing," *CAL*, 2019.

[3] A. Balaji and A. Das, "Compiling spiking neural networks to mitigate neuromorphic hardware constraints"," in *IGSC Workshops*, 2020.

[4] A. Balaji and A. Das, "A framework for the analysis of throughput-constraints of snns on neuromorphic hardware," in *ISVLSI*, 2019.

[5] A. Balaji, F. Corradi, A. Das, S. Pande, S. Schaafsma, and F. Catthoor, "Power-accuracy trade-offs for heartbeat classification on neural networks hardware," *JOLPE*, 2018.

[6] A. Balaji, S. Ullah, A. Das, and A. Kumar, "Design methodology for embedded approximate artificial neural networks," in *GLSVLSI*, 2019.

[7] A. Balaji, Y. Wu, A. Das, F. Catthoor, and S. Schaafsma, "Exploration of segmented bus as scalable global interconnect for neuromorphic computing," in *GLSVLSI*, 2019.

[8] A. Balaji, P. Adiraju, H. J. Kashyap, A. Das, J. L. Krichmar, N. D. Dutt, and F. Catthoor, "PyCARL: A PyNN interface for hardware-software co-simulation of spiking neural network," in *IJCNN*, 2020.

[9] A. Balaji, A. Das, Y. Wu, K. Huynh, F. G. Dell'anna, G. Indiveri, J. L. Krichmar, N. D. Dutt, S. Schaafsma, and F. Catthoor, "Mapping spiking neural networks to neuromorphic hardware," *TVLSI*, 2020.

[10] A. Balaji, T. Marty, A. Das, and F. Catthoor, "Run-time mapping of spiking neural networks to neuromorphic hardware," *JSPS*, 2020.

[11] A. Balaji, S. Song, A. Das, J. Krichmar, N. Dutt, J. Shackleford, N. Kandasamy, and F. Catthoor, "Enabling resource-aware mapping of spiking neural networks via spatial decomposition," *ESL*, 2020.

[12] A. Balaji, S. Song, T. Titirsha, A. Das, J. Krichmar, N. Dutt, J. Shackleford, N. Kandasamy, and F. Catthoor, "Neuroxplorer 1.0: An extensible framework for architectural exploration with spiking neural networks," *arXiv*, 2021.

[13] A. N. Burkitt, "A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input," *Biological Cybernetics*, 2006.

[14] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola, L. L. Sanches, I. Boybat, M. Le Gallo, K. Moon, J. Woo, H. Hwang, and Y. Leblebici, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X*, 2017.

[15] N. Caporale and Y. Dan, "Spike Timing–Dependent Plasticity: a Hebbian Learning rule," *Annual Review Neuroscience*, 2008.

[16] E. A. Cartier, W. Kim, N. Gong, T. Gokmen, M. M. Frank, D. M. Bishop, Y. Kim, S. Kim, T. Ando, E. Y. Wu, P. Adusumilli, J. Rozen, P. M. Solomon, W. Haensch, M. J. Brightsky, A. Sebastian, G. W. Burr, and V. Narayanan, "Reliability challenges with materials for analog computing," in *IRPS*, 2019.

[17] F. Catthoor, S. Mitra, A. Das, and S. Schaafsma, "Very large-scale neuromorphic systems for biological signal processing," in *CMOS Circuits for Biological Sensing and Processing*, 2018.

[18] G. K. Chen, R. Kumar, H. E. Sumbul, P. C. Knag, and R. K. Krishnamurthy, "A 4096-neuron 1M-synapse 3.8-pJ/SOP spiking neural network with on-chip STDP learning and sparse weights in 10-nm FinFET CMOS," *JSSC*, 2018.

[19] T. S. Chou, L. D. Bucci, and J. L. Krichmar, "Learning touch preferences with a tactile robot using dopamine modulated STDP in a model of insular cortex," *Frontiers in Neurorobotics*, 2015.

[20] T. S. Chou, H. J. Kashyap, J. Xing, S. Listopad, E. L. Rounds, M. Beyeler, N. Dutt, and J. L. Krichmar, "CARLsim 4: An open source library for large scale, biologically detailed spiking neural network simulation using heterogeneous clusters," in *IJCNN*, 2018.

[21] Y. Dan and M. M. Poo, "Spike timing-dependent plasticity of neural circuits," *Neuron*, 2004.

[22] A. Das, P. Pradhapan, W. Groenendaal, P. Adiraju, R. Rajan, F. Catthoor, S. Schaafsma, J. Krichmar, N. Dutt, and C. Van Hoof, "Unsupervised heart-rate estimation in wearables with Liquid states and a probabilistic readout," *Neural Networks*, 2018.

[23] A. Das and A. Kumar, "Fault-aware task re-mapping for throughput constrained multimedia applications on noc-based mpsocs," in *RSP*, 2012.

[24] A. Das and A. Kumar, "Dataflow-based mapping of spiking neural networks on neuromorphic hardware," in *GLSVLSI*, 2018.

[25] A. Das, A. Kumar, and B. Veeravalli, "Communication and migration energy aware design space exploration for multicore systems with intermittent faults," in *DATE*, 2013.

[26] A. Das, A. Kumar, and B. Veeravalli, "Communication and migration energy aware task mapping for reliable multiprocessor systems," *FGCS*, 2014.

[27] A. Das, A. Kumar, and B. Veeravalli, "Temperature aware energy-reliability trade-offs for mapping of throughput-constrained applications on multimedia MPSoCs," in *DATE*, 2014.
[28] A. Das, R. A. Shafik, G. V. Merrett, B. M. Al-Hashimi, A. Kumar, and B. Veeravalli, "Reinforcement learning-based inter-and intra-application thermal optimization for lifetime improvement of multicore systems," in *DAC*, 2014.
[29] A. Das, A. Kumar, and B. Veeravalli, "Reliability and energy-aware mapping and scheduling of multimedia applications on multiprocessor systems," *TPDS*, 2015.
[30] A. Das, G. V. Merrett, M. Tribastone, and B. M. Al-Hashimi, "Workload change point detection for runtime thermal management of embedded systems," *TCAD*, 2015.
[31] A. Das, F. Catthoor, and S. Schaafsma, "Heartbeat classification in wearables using multi-layer perceptron and time-frequency joint distribution of ECG," in *CHASE*, 2018.
[32] A. Das, Y. Wu, K. Huynh, F. Dell'Anna, F. Catthoor, and S. Schaafsma, "Mapping of local and global synapses on spiking neuromorphic hardware," in *DATE*, 2018.
[33] A. K. Das, A. Kumar, B. Veeravalli, and F. Catthoor, *Reliable and Energy Efficient Streaming Multiprocessor Systems*, 2018.
[34] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, 2018.
[35] M. V. Debole, B. Taba, A. Amir, F. Akopyan, A. Andreopoulos, W. P. Risk, J. Kusnitz, C. O. Otero, T. K. Nayak, R. Appuswamy, P. J. Carlson, A. S. Cassidy, P. Datta, S. K. Esser, G. J. Garreau, K. L. Holland, S. Lekuch, M. Mastro, J. Mckinstry, C. Di Nolfo, J. Sawada, B. Paulovicks, K. Schleupen, B. G. Shaw, J. L. Klamo, M. D. Flickner, J. V. Arthur, and D. S. Modha, "TrueNorth: Accelerating from zero to 64 million neurons in 10 years," *Computer*, 2019.
[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.
[37] L. Deng, "The mnist database of handwritten digit images for machine learning research," *SPS*, 2012.
[38] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, 2015.
[39] M. Dong, X. Huang, and B. Xu, "Unsupervised speech recognition through spike-timing-dependent plasticity in a convolutional spiking neural network," *PloS One*, 2018.
[40] F. Galluppi, X. Lagorce, E. Stromatias, M. Pfeiffer, L. A. Plana, S. B. Furber, and R. B. Benosman, "A framework for plasticity implementation on the SpiNNaker neural architecture," *Frontiers in Neuroscience*, 2015.
[41] R. Gao, Z. Ji, A. B. Manut, J. F. Zhang, J. Franco, S. W. M. Hatta, W. D. Zhang, B. Kaczer, D. Linten, and G. Groeseneken, "NBTI-Generated defects in nanoscaled devices: Fast characterization methodology and modeling," *TED*, 2017.
[42] T. Grasser, M. Waltl *et al.*, "Implications of gate-sided hydrogen release for post-stress degradation build-up after BTI stress," in *IRPS*, 2017.
[43] S. Grün and S. Rotter, *Analysis of parallel spike trains*, 2010.
[44] D. Hisamoto, W.-C. Lee, J. Kedzierski, H. Takeuchi, K. Asano, C. Kuo, E. Anderson, T.-J. King, J. Bokor, and C. Hu, "FinFET-a self-aligned double-gate MOSFET scalable to 20 nm," *TED*, 2000.
[45] G. Indiveri, "A low-power adaptive integrate-and-fire neuron circuit," in *ISCAS*, 2003.
[46] E. M. Izhikevich, "Polychronization: Computation with spikes," *Neural Computation*, 2006.
[47] Y. Ji, Y. Zhang, S. Li, P. Chi, C. Jiang, P. Qu, Y. Xie, and W. Chen, "NEUTRAMS: Neural network transformation and co-design under neuromorphic hardware constraints," in *MICRO*, 2016.
[48] Y. Ji, Y. Zhang, W. Chen, and Y. Xie, "Bridge the gap between neural networks and neuromorphic hardware with a neural network compiler," in *ASPLOS*, 2018.
[49] H. J. Kashyap *et al.*, "A recurrent neural network based model of predictive smooth pursuit eye movement in primates," in *IJCNN*, 2018.
[50] D. Kraak, I. Agbo, M. Taouil, S. Hamdioui, P. Weckx, S. Cosemans, and F. Catthoor, "Degradation analysis of high performance 14nm FinFET SRAM," in *DATE*, 2018.
[51] D. Kraak, M. Taouil, I. Agbo, S. Hamdioui, P. Weckx, S. Cosemans, and F. Catthoor, "Parametric and Functional Degradation Analysis of Complete 14-nm FinFET SRAM," *TVLSI*, 2019.
[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012.
[53] H. Kükner, M. Khatib, S. Morrison, P. Weckx, P. Raghavan, B. Kaczer, F. Catthoor, L. der Perre, R. Lauwereins, and G. Groeseneken, "Degradation analysis of datapath logic subblocks under NBTI aging in FinFET technology," in *ISQED*, 2014.
[54] E. Kültürsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating STT-RAM as an energy-efficient main memory alternative," in *ISPASS*, 2013.

[55] S. Kundu, K. Basu, M. Sadi, T. Titirsha, S. Song, A. Das, and U. Guin, "Special session: Reliability analysis for ML/AI hardware," in *VTS*, 2021.
[56] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, 2015.
[57] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable DRAM alternative," in *ISCA*, 2009.
[58] R. Legenstein, D. Pecevski, and W. Maass, "A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback," *PLoS Computational Biology*, 2008.
[59] C. Liu, Y. Sun, P. Ren, D. Gao, W. Luo, Z. Chen, and Y. Xia, "New challenges of design for reliability in advanced technology node," in *EDTM*, 2020.
[60] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, 1997.
[61] M. M. Mahmoud and N. Soin, "A comparative study of lifetime reliability of planar MOSFET and FinFET due to BTI for the 16 nm CMOS technology node based on reaction-diffusion model," *Microelectronics Reliability*, 2019.
[62] A. Mallik, D. Garbin, A. Fantini, D. Rodopoulos, R. Degraeve, J. Stuijt, A. K. Das, S. Schaafsma, P. Debacker, G. Donadio, H. Hody, L. Goux, G. S. Kar, A. Furnemont, A. Mocuta, and P. Raghavan, "Design-technology co-optimization for OxRRAM-based synaptic processing unit," in *VLSIT*, 2017.
[63] G. B. Moody, R. G. Mark, and A. L. Goldberger, "Physionet: a web-based resource for the study of physiologic signals," *Engineering in Medicine and Biology Magazine*, 2001.
[64] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *TBCAS*, 2017.
[65] H. Mulaosmanovic, J. Ocker, S. Müller, M. Noack, J. Müller, P. Polakowski, T. Mikolajick, and S. Slesazeck, "Novel ferroelectric FET based synapse for neuromorphic systems," in *VLSIT*, 2017.
[66] S. R. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, "A phase-change memory model for neuromorphic computing," *JAP*, 2018.
[67] S. Navarro, C. Navarro, C. Marquez, N. Salazar, P. Galy, S. Cristoloveanu, and F. Gamiz, "Reliability study of thin-oxide zero-ionization, zero-swing fet 1t-dram memory cell," *EDL*, 2019.
[68] S. W. Pae, H. C. Sagong, C. Liu, M. J. Jin, Y. H. Kim, S. J. Choo, J. J. Kim, H. J. Kim, S. Y. Yoon, H. W. Nam, and Others, "Considering physical mechanisms and geometry dependencies in 14nm FinFET circuit aging and product validations," in *IEDM*, 2015.
[69] D. G. Pierce and P. G. Brusius, "Electromigration: A review," *Microelectronics Reliability*, 1997.
[70] K. Puschkarsky, H. Reisinger, T. Aichinger, W. Gustin, and T. Grasser, "Understanding BTI in SiC MOSFETs and its impact on circuit operation," *TDMR*, 2018.
[71] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *ISCA*, 2009.
[72] B. Rajendran, A. Sebastian, M. Schmuker, N. Srinivasa, and E. Eleftheriou, "Low-Power neuromorphic hardware for signal processing applications: A review of architectural and system-level design approaches," *SPS*, 2019.
[73] S. M. Ramey, C. Prasad, and A. Rahman, "Technology scaling implications for bti reliability," *Microelectronics Reliability*, 2018.
[74] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou *et al.*, "Mlperf inference benchmark," in *ISCA*, 2020.
[75] M. Rossum, "A novel spike distance," *Neural Computation*, 2001.
[76] P. J. Roussel, A. Chasin, S. Demuynck, N. Horiguchi, D. Linten, and A. Mocuta, "New methodology for modelling MOL TDDB coping with variability," in *IRPS*, 2018.
[77] G. Rzepa, J. Franco, B. O'Sullivan, A. Subirats, M. Simicic, G. Hellings, P. Weckx, M. Jech, T. Knobloch, M. Waltl, and Others, "Comphy—A compact-physics framework for unified modeling of BTI," *Microelectronics Reliability*, 2018.
[78] H. C. Sagong, K. Choi, H. Jiang, J. Park, H. Rhee, and S. Pae, "Reliability of advanced finfet technology nodes beyond planar," in *EDTM*, 2020.
[79] R. Santos, S. Venkataraman, A. Das, and A. Kumar, "Criticality-aware scrubbing mechanism for SRAM-based FPGAs," in *FPL*, 2014.
[80] S. Schliebs and N. Kasabov, "Evolving spiking neural network-a survey," *Evolving Systems*, 2013.
[81] R. A. Shafik, A. Das, S. Yang, G. Merrett, and B. M. Al-Hashimi, "Adaptive energy minimization of openmp parallel applications on many-core systems," in *HiPEAC Workshops*, 2015.
[82] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv*, 2014.
[83] S. Song and A. Das, "Design methodologies for reliable and energy-efficient PCM systems," in *IGSC Workshops*, 2020.
[84] S. Song and A. Das, "A case for lifetime reliability-aware neuromorphic computing," in *MWSCAS*, 2020.
[85] S. Song, A. Das, O. Mutlu, and N. Kandasamy, "Enabling and exploiting partition-level parallelism (PALP) in phase change memories," *TECS*, 2019.

[86] S. Song, A. Balaji, A. Das, N. Kandasamy, and J. Shackleford, "Compiling spiking neural networks to neuromorphic hardware," in *LCTES*, 2020.
[87] S. Song, A. Das, and N. Kandasamy, "Exploiting inter- and intra-memory asymmetries for data mapping in hybrid tiered-memories," in *ISMM*, 2020.
[88] S. Song, A. Das, and N. Kandasamy, "Improving dependability of neuromorphic computing with non-volatile memory," in *EDCC*, 2020.
[89] S. Song, A. Das, O. Mutlu, and N. Kandasamy, "Improving phase change memory performance with data content aware access," in *ISMM*, 2020.
[90] S. Song, A. Das, O. Mutlu, and N. Kandasamy, "Aging-aware request scheduling for non-volatile main memory," in *ASP-DAC*, 2021.
[91] S. Taghipour and R. N. Asli, "Aging comparative analysis of high-performance FinFET and CMOS flip-flops," *Microelectronics Reliability*, 2017.
[92] T. Titirsha and A. Das, "Reliability-performance trade-offs in neuromorphic computing," in *IGSC Workshops*, 2020.
[93] T. Titirsha and A. Das, "Thermal-aware compilation of spiking neural networks to neuromorphic hardware," in *LCPC*, 2020.
[94] T. Titirsha, S. Song, A. Balaji, and A. Das, "On the role of system software in energy management of neuromorphic computing," in *CF*, 2021.
[95] T. Titirsha, S. Song, A. Das, J. Krichmar, N. Dutt, N. Kandasamy, and F. Catthoor, "Endurance-aware mapping of spiking neural networks to neuromorphic hardware," *TPDS*, 2021.
[96] R. Van Rullen and S. J. Thorpe, "Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex," *Neural Computation*, 2001.
[97] A. F. Vincent, J. Larroque, N. Locatelli, N. B. Romdhane, O. Bichler, C. Gamrat, W. S. Zhao, J.-O. Klein, S. Galdin-Retailleau, and D. Querlioz, "Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems," *TBCAS*, 2015.
[98] X. Wan, B. Zhu, M. Mohan, K. Wu, D. Choi, and A. Gondal, "HCI Improvement on 14nm FinFET IO Device by Optimization of 3D Junction Profile," in *IRPS*, 2019.
[99] P. Weckx, B. Kaczer, H. Kukner, J. Roussel, P. Raghavan, F. Catthoor, and G. Groeseneken, "Non-Monte-Carlo methodology for high-sigma simulations of circuits under workload-dependent BTI degradation-application to 6T SRAM," in *IRPS*, 2014.
[100] C. Yilmaz, L. Heiß, C. Werner, and D. Schmitt-Landsiedel, "Modeling of NBTI-recovery effects in analog CMOS circuits," in *IRPS*, 2013.

A INTRODUCTION TO SPIKING NEURAL NETWORKS

Spiking Neural Networks (SNNs) [60] are regarded as the third generation of neural networks (see Figure 20a). SNNs consist of spiking neurons, which are implemented using integrate and fire [13] model. In this model, a neuron fires a spike when its membrane voltage exceeds a threshold and subsequently the membrane voltage is reset. The moment of threshold crossing in a neuron defines its *firing time*. Post firing, the neuron goes into a refractory state, where the neuron cannot be excited to generate a second action potential (no matter how intense the input stimulus be) (see Figure 20b). Spiking neurons are interconnected via synapses as shown in Figure 20a.

Information Encoding in SNNs: Information in SNNs can be encoded using different techniques [80], prominent among which are rate coding [96] and temporal coding [75]. Rate coding encodes information as number of spikes within an encoding window without considering the temporal characteristics of the signal. Temporal coding encodes information as inter-spike interval (ISI), capturing the spatio-temopral structure of the input signal.

Machine Learning Approaches using SNNs: SNNs can be used to implement many machine learning approaches. One example is the supervised approach, where an SNN is first *trained* with examples from the field and then used for *inference* with current data. SNNs can also implement unsupervised, semi-supervised, and reinforcement learning-based machine learning approaches.

Learning Algorithms in SNNs: Currently, spike-based learning rules are limited, compared to the wide range of learning rules available for analog or rate-based artificial neural networks (ANNs). Most learning rules are based on unsupervised correlational learning rules, such as spike

timing dependent plasticity (STDP) [21], short-term plasticity (STP), and long-term plasticity (LTP). Other modifications include a localized version of backpropagation suitable for SNNs. However, these variants are supervisory and takes a while to converge. Attempts have been made to add a reinforcer to STDP based on the idea that dopamine in the brain carries a reward prediction error signal. In practice, dopamine modulated STDP (DA-STDP) takes a long time before the network has a strong enough signal to drive behavior [19]. Recently, a reward-modulated STDP (R-STDP) learning is developed to train SNN controllers for obstacle avoiding behavior in mobile robots [58].

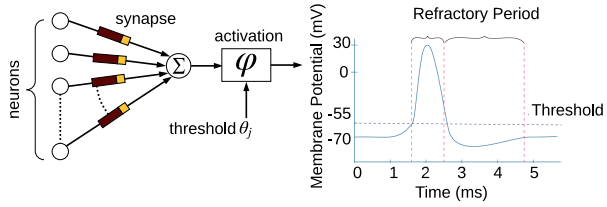
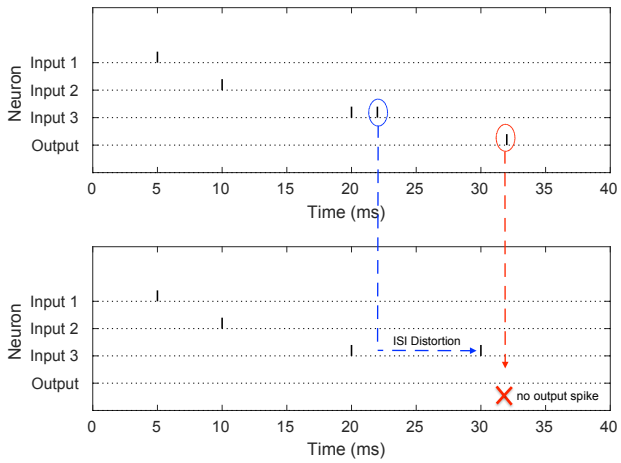


Fig. 20. Illustration of spiking neural networks.

B IMPACT OF ISI DISTORTION ON PERFORMANCE OF SNNS

To illustrate how ISI distortion and spike disorder impact accuracy, we consider a small SNN example where three input neurons are connected to an output neuron. In Figure 21, we illustrate the impact of ISI distortion on the output spike. In the top sub-figure, we observe that a spike is generated at the output neuron at 22ms due to spikes from the input neurons. In the bottom sub-figure, we observe that the second spike from input 3 is delayed, i.e., has ISI distortion. As a result of this distortion, there is no output spike. Missing spikes can impact application accuracy, as spike timings encode information in SNNs.



(a) Impact of ISI distortion on accuracy. Top sub-figure shows a scenario where an output spike is generated based on the spikes received from the three input neurons. Bottom sub-figure shows a scenario where the second spike from neuron 3 is delayed. There are no output spikes generated.

(b) Impact of spike disorder on acc
where spike A is received at the
the out spike at 21ms. Bottom sub-f
order of A & B is reversed. There ar

Fig. 21. Impact of ISI distortion on accuracy. Top sub-figure shows a scenario where an output spike is generated based on the spikes received from the three input neurons. Bottom sub-figure shows a scenario where the second spike from neuron 3 is delayed. As a result of this ISI distortion, there are no output spikes generated. Loss of spikes can lead to accuracy drop.

Fig. 3: Impact of ISI distortion (a) and spike disorder (b) on connected to a single output neuron.

of crossbars in the hardware, latency, ISI, and spike disorder increases. This is because with increase in the number of crossbars, spike traffic on the shared interconnect increases, which increases the congestion, and delays some spikes more than others. When we use a hardware with 36 small crossbars arranged in a 6x6 mesh, we observe a significant increase of latency (average 3.2x), ISI distortion (average 6x), and spike disorder (average 1.5x) compared to the baseline configuration of using 4 large crossbars.

From this analysis, we **conclude** that when more crossbars are used for an application, latency, ISI distortion, and spike disorder increases, which jointly impacts accuracy, as described next.

Accuracy Impact: To illustrate how ISI distortion and spike disorder impact accuracy, we consider a small SNN example where three input neurons are connected to an output neuron. In Figure 3a, we illustrate the impact of ISI distortion on the output spike. In the top sub-figure, we observe that a spike is generated at the output neuron at 22ms due to spikes from the input neurons. In the bottom sub-figure, we observe that the second spike from input 3 is delayed, i.e., has ISI distortion. As a result of this distortion, there is no output spike. Missing spikes can impact application accuracy, as spikes encode information in SNNs.

In Figure 3b, we illustrate the impact of spike disorder on the output spike. In the top sub-figure, we observe that the spike A from input 2 is generated before the spike B from input 3, causing an output spike to be generated at 21ms. In the bottom sub-figure, we observe that the spike order of inputs 2 and 3 is reversed, i.e., the spike B is generated before the spike A. This spike disorder results in no spike being generated at

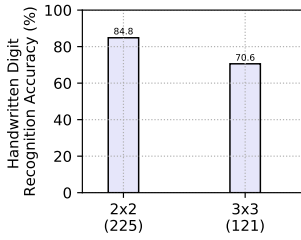


Fig. 4: Handwritten digit r
hardware configuration. This
accuracy, which is obtained

drops from 85% on a 2x2
compared to the accuracy of
ulation. This clearly motivat
of SNNs on the neuromorph

We now describe our r
Spiking Neural Networks on

III. SpiNeMap: Spiking
on Neuromo

A. High-Level overview and

In Figure 5a, we illustrate
is conventionally mapped to
of these conventional approa
neural networks, and custom
NEUTRAMS [25] and Eyeri
approach PACMAN [28] f
large SNNs, (2) customize
Neumann style computing),