ReSpike: A Co-Design Framework for Evaluating SNNs on ReRAM-based Neuromorphic Processors [⋆]

Kazi Asifuzzaman[1] ✉, Aaron R. Young[1], Prasanna Date[1], Shruti R. Kulkarni[1], Narasinga Rao Miniskar[1], Matthew Marinella[2], and Jeffrey S. Vetter[1]

[1] Oak Ridge National Laboratory, 1 Bethel Valley Road, Oak Ridge, TN 37830, USA
{asifuzzamank, youngar, datepa, kulkarnisr, miniskarnr, vetter}@ornl.gov,
[2] Arizona State University, 1151 S Forest Ave, Tempe, AZ 85281, USA
m@asu.edu

Abstract. With Moore's law approaching its end, traditional von Neumann architectures are struggling to keep up with the exceeding performance and memory requirements of artificial intelligence and machine learning algorithms. Unconventional computing approaches such as neuromorphic computing that leverage spiking neural networks (SNNs) to perform computation are gaining traction and seek the paradigm shift necessary to sustain the increasing demands of modern applications. Novel memory technologies, such as resistive RAM (ReRAM), employ a crossbar architecture that possesses the inherent capability of efficiently computing vector-matrix multiplication—a dominant operation in SNNs. The prospect of naturally mapping SNNs to the crossbar structures provides a unique opportunity for achieving a high-performance, power-efficient neuromorphic system. In this work, we present *ReSpike*, which is a new framework, behavioral simulator, and architectural design based on ReRAM crossbar architectures, enabling modeling and co-design to achieve efficient execution of SNNs. We drive this co-design forward by quantifying the impact that ReRAM cell *nonidealities* have on the corresponding accuracy of an SNN application.

Keywords: Neuromorphic computing · Spiking Neural Networks · ReRAM · Co-design framework

1 Introduction

Advanced software functionalities, particularly those utilizing artificial intelligence (AI) and machine learning (ML) algorithms, exhibit exceeding compute

[⋆] Notice: This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a non-exclusive, paid up, irrevocable, world-wide license to publish or reproduce the published form of the manuscript, or allow others to do so, for U.S. Government purposes. The DOE will provide public access to these results in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

and memory performance requirements. As Moore's law is approaching its limits [1], traditional von Neumann architectures struggle to keep up with the unprecedented demands of such applications. To improve efficiency, researchers are exploring novel materials and device-level innovations (e.g., resistive RAM [ReRAM], electrochemical RAM [ECRAM], spin-transfer torque magnetic RAM [STT-MRAM]) in conjunction with alternative computing approaches.

Neuromorphic computing is a promising computing paradigm that performs computations by emulating the human brain, adopting a non-von Neumann approach [2], and colocating processing and memory to significantly reduce the memory transfer overhead. Neuromorphic architectures leverage neuron and synapse primitives for computation, and the "programs" created for these architectures are detailed in neural networks which specify the connections and parameters of the neurons and synapses. A spiking neural network (SNN) most closely portrays a biological neuron-synapse structure, where neurons communicate with each other over discrete, binary spikes [3]. Such organization with event-driven asynchronous operation allows neuromorphic systems to be extremely energy efficient [4] while providing compute capabilities comparable to those of conventional systems. Although mature digital implementations of neuromorphic systems such as Intel Loihi [5] and IBM TrueNorth [6] exist, analog computing approaches anticipate several orders of magnitude improvement in energy efficiency [7], implying that the future of truly energy-efficient, reliable neuromorphic systems lies in fully analog or mixed-signal implementations.

Among the most compelling *beyond-CMOS* candidates, ReRAM is an emerging, nonvolatile memory technology that features analog compute capability, fast writing speed, and high on-off ratio with CMOS compatibility [8]. ReRAM employs a crossbar architecture that is inherently capable of performing vector-matrix multiplication (VMM) operation (in analog domain), which is the most prevalent and critical operation in SNNs [9]. Therefore, enabling efficient VMM operations in analog computation ensures a significant improvement in performance and energy efficiency for neuromorphic systems.

Incorporating novel technologies with an unconventional computing approach brings several challenges regarding the hardware/software ecosystem. In recent years, hardware design has significantly evolved to capture the algorithm and software requirements more closely than ever before. On the other hand, the algorithm and software stack has also evolved to closely incorporate the hardware constraints. This diffusion of ideas and incorporation of constraints from top to bottom (algorithms $\rightarrow$ software $\rightarrow$ hardware) and vice versa are known as *co-design*, which is a crucial step to achieve high-performance hardware that adheres to strict requirements pertaining to size, weight, and power.

In this study, we develop *ReSpike*, an architecture design and simulator with a complete co-design framework for the exploration of neuromorphic architectures aimed at utilizing the analog computing capability of ReRAM crossbars. To this extent, the main contributions of this work are as follows:

– Designing and implementing the ReSpike neuromorphic processor architecture by using ReRAM cell-array structures capable of processing SNNs, leveraging in-memory analog computation.

- Developing a co-design framework that accommodates the ReSpike architecture across the stack from applications/algorithms to devices/materials, enabling seamless training and inference of SNNs.
- Quantifying the impact of certain ReRAM device nonidealities on the deviation of accuracy on the proposed architecture.

The rest of the paper is organized as follows: Section 2 discusses the underlying background and concepts of the study; Section 3 presents the organization and implementation of the ReSpike framework; Section 4 discusses the results obtained from the experiments; Section 5 discusses prior works related to this study; and Section 6 presents the outcome and conclusion of the work.

2 Background

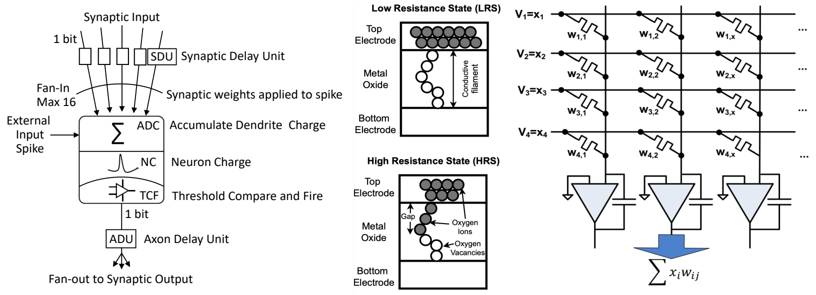
In this section, we discuss the fundamentals of neuromorphic computing, including neurons, synapses, and SNNs; the characteristics, behavior, and purpose of the Smartpixels application along with its training process and platform; and the organization, and operation of ReRAM, providing the essential background information for the techniques and technologies discussed in the study.

2.1 Neuromorphic Computing

Neuromorphic architectures are inspired by the structure of the human brain's neurons and synapses, adopting a non-von Neumann approach and collocating processing and memory, in contrast to the conventional computing systems in which the processor and memory are organized as separate units. Conventional computing systems execute programs based on numerical values that are transformed in binary values for processing, whereas programs for neuromorphic architectures are defined by the structure and organization of neural networks with associated parameters. Neurons and synapses are the basic building blocks of neuromorphic computers, where neurons are connected with each other through synapses and communicate over discrete, binary spikes. Defining when these spikes occur, their magnitude, and shape enables a neuromorphic program's information to be encoded. Collocated processing and memory can significantly improve system throughput, and event-driven execution provide an immense opportunity to be extremely energy efficient [3]. Several implementations of neuromorphic systems have demonstrated their inherent scalability [5, 10].

To carry out the SNN simulation and evaluation, we employ the TENNLab neuromorphic computing framework [11]. This framework provides a software ecosystem across different levels of the compute stack, allowing for training SNNs, developing model abstraction, and co-designing a hardware platform. At the very last layer of the stack lie the architectural, circuit, and device-level constraints, which are provided to the software simulator. The neuron and synapse dynamics are defined also by the underlying hardware dynamics.

To train the SNNs, we utilize Evolutionary Optimization for Neuromorphic Systems (EONS) [12]. EONS is an evolutionary algorithm that optimizes the parameters and structure of an SNN for deployment in neuromorphic systems. It takes into account the constraints of the underlying hardware for optimizing SNNs. It begins with a set of randomly originated SNNs as its initial population.


Synaptic Input
1 bit
SDU
Synaptic Delay Unit
Fan-In Max 16
Synaptic weights applied to spike
External Input Spike
ADC
Accumulate Dendrite Charge
NC
Neuron Charge
TCF
Threshold Compare and Fire
1 bit
ADU
Axon Delay Unit
Fan-out to Synaptic Output
Low Resistance State (LRS)
Top Electrode
Metal Oxide
Conductive filament
Bottom Electrode
High Resistance State (HRS)
Gap
Oxygen Ions
Oxygen Vacancies
$V_1=x_1$
$V_2=x_2$
$V_3=x_3$
$V_4=x_4$
$w_{1,1}$ $w_{1,2}$ $w_{1,x}$
$w_{2,1}$ $w_{2,2}$ $w_{2,x}$
$w_{3,1}$ $w_{3,2}$ $w_{3,x}$
$w_{4,1}$ $w_{4,2}$ $w_{4,x}$
$\sum x_i w_{ij}$

(a) (b)

Fig. 1: (a) Integrate-and-fire neuron model. (b) ReRAM cells in low-resistance / high-resistance states and ReRAM cells organized in crossbar arrays [13], capable of performing VMM operations without additional compute units.

The initial population may also include networks generated from a previous run, generated from a separate training approach, or manually customized by a user. When an initial population of networks is established, each network receives a fitness score, followed by an evaluation of the population. Then, tournament selection is used to select parent networks with the best fitness scores and reproduction operations are conducted, resulting in a child population through crossover, cloning, and random mutations. EONS adopts a special mechanism to retain the best networks in the child population from the previous population. This process to evaluate, select, and reproduce continues repeatedly until the stopping criteria are triggered, such as reaching the desired fitness score or a maximum number of generations. A network of neurons connected via synapses forms the basic structure of SNNs, which is the primary mechanism for deploying AI as well as general-purpose workloads on neuromorphic computers. In this study, we use the integrate-and-fire (IF) neuron model, as shown in Figure 1a. The details of the behavioral SNN model is discussed in Section 3.1 .

2.2 Smartpixels Application

Previous studies have demonstrated an SNN for an application in high energy physics experimentation [14]. It was shown that an SNN trained with EONS can successfully filter out simulated charged clusters in the sensors associated with low transverse momentum (p_T) tracks with a signal efficiency of 91% but with nearly half the number of parameters compared with a similar-performing DNN. It was also shown that the SNN could natively process a temporal signal in the form of spikes encoded from the sensor charge waveforms. As the amount of data in the form of spikes can be very sparse, this also holds potential to reduce the operating energy in the underlying hardware.

For example, a set of cluster samples, having a dimension of 21×13, are converted to streams of spikes, where the number of spikes in each channel is dependent on the rising or falling rate of the corresponding cluster's charge waveform. The 3D spike cluster is further spatially reduced by compressing the spike trains along rows, or columns, or specified dimensions within the 21×13 cluster frame [14]. For filtering out low momentum charge clusters, the SNN is trained to classify the cluster samples into high or low p_T. The SNN has two output neurons, where one neuron spikes the highest to indicate a sample is low p_T, and the other spikes the highest for the high-p_T sample. During training, the samples having p_T less than a threshold (typically 0.2 GeV) are categorized as low-p_T ones and are filtered out by the detector. During inference, the metrics used are signal efficiency, which measures the success of correctly identifying high-p_T samples ($|p_T| > 2$ GeV), and data rejection, which measures the success rate on correctly identifying low-p_T samples ($|p_T| \leq 2$ GeV).

We used the neuromorphic TENNLab framework [11] to create and train SNNs using EONS for the Smartpixels application. The SNN simulation is carried out on a software simulator mimicking the hardware behavior of a field-programmable gate array-based neuromorphic hardware called Caspian [15]. Caspian uses integer precision to represent all the parameters of the network. The Smartpixels SNN was trained with a precision of 9-bits for the weights, 8-bits for the neuron thresholds, and 4-bits for the synaptic delays.

2.3 ReRAM

ReRAM is a nonvolatile memory composed of ReRAM cells that leverage *resistance* properties to store data. A ReRAM cell is typically a two-terminal device with a metal-insulator-metal structure. The insulator is realized with memristors, which can form a low-resistance state (LRS) or a high-resistance state (HRS) by establishing and dissolving a conductive filament through applied voltage. The `SET` process is used to establish the conductive filament by oxygen drifts to the memristive layer. The `RESET` voltage is applied to bring oxygen ions to fill up the vacancies creating a gap in the conductive filament converting it to a HRS [16]. The memristive layer usually consists of HfO_2, NiO, Ta_2O_5, TiO_2, or Al_2O_3, while the terminals are formed with Pt or TiN. Figure 1b presents a simplified structure and operational states of ReRAM cells, as well as how the cells are organized in a crossbar structure [13].

The unique structure and properties of ReRAM crossbars make it naturally capable of performing VMMs, which are a key operation in SNNs. In the crossbar, when an input vector is fed to the word lines as supply voltage, the current accumulated on the bit lines produces the resultants of the VMM operation, according to Kirchhoff's law [13]. As the ReRAM device and crossbar operate in the analog domain, digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) are added to the word-line and bit-line interfaces [4].

Although the analog properties of ReRAM device arrays offer intrinsic efficiency benefits over digital implementations, retaining accuracy has been a consistent problem due to read noise [17], programming errors [13], process variation, parasitic resistance, and other issues. These error and noise models can be further subcategorized in state-independent and state-proportional models. In state-proportional models, the deviation is proportional to the cell's state (i.e., smaller conductance having an smaller error), and state-independent models are, as the name suggests, independent of the cell's current state [18].

3 ReSpike Framework

This section describes both the ReSpike architecture model used to build up the ReRAM devices into neuromorphic computing systems and the co-design effort leveraging the ReSpike simulation framework, which is used to simulate the behavior of these systems.

3.1 Behavioral model of SNNs using IF neurons

As highlighted in Section 2.1 and Figure 1a, our SNN behavioral model uses IF neurons. To model SNNs with IF neurons, the ReSpike architecture leverages the ReRAM crossbar structure to sum weighted synaptic input connections, adding a digital neuron component to it, which converts the accumulated charge from the synapses into a digital value with an ADC. Then, the threshold compare-and-fire block adds this incoming charge to the neuron's previous charge and compares this sum to a threshold value. If the charge exceeds the threshold, then the neuron fires, emitting a spike to the ADU and clearing the stored charge. Otherwise, if the value is less than the threshold, then the neuron does not fire, and the charge is stored for the next cycle. The ADU adds temporal delay to spikes by enabling a programmable delay in the spike propagation. In hardware, this is commonly implemented as a shift register. To allow for all-to-all synaptic connection configurability, all the outputs from the digital neurons are routed back to the input rows of the crossbar. Figure 2a presents a simplified block diagram of this architecture with all connections and components. The architecture is a mixed-signal design with the synapses implemented in the analog domain with the ReRAM devices programmed with the weight values assigned to the synapses.

3.2 Vector-matrix multiplication using crossbar architecture

To process a SNN, the structure essentially performs VMM operations in which the input rows of the crossbar take in the input vector values from the spikes of the previous cycle, and the weight matrix is assigned on the crossbar reflecting the synaptic connections of the network. During continuous cycles, the VMM operations accumulate dendrite charges for each neuron in the digital neuron model and cause the neuron to fire when the charge exceeds the neuron's threshold. While the VMM operation is performed on the crossbar architecture in the analog domain, the Analog-to-Digital Converter (ADC) converts the dendrite charge to a digital value, and the threshold compare-and-fire takes place in the digital domain. These binary spikes are then reconverted to a high (spike) or low (no spike) voltage level by the Digital-to-Analog Converter (DAC) and applied to the rows of the crossbar. External input and output are handled in the digital neuron component by applying the incoming external input directly to the neuron's accumulator and by sending external spike outputs when the neuron fires.

3.3 The co-design approach

The ReSpike architecture must be accommodated through a larger co-design effort to explore spiking neuromorphic systems across the full stack (e.g., applications, algorithms, software, architectures, circuits, devices), as highlighted

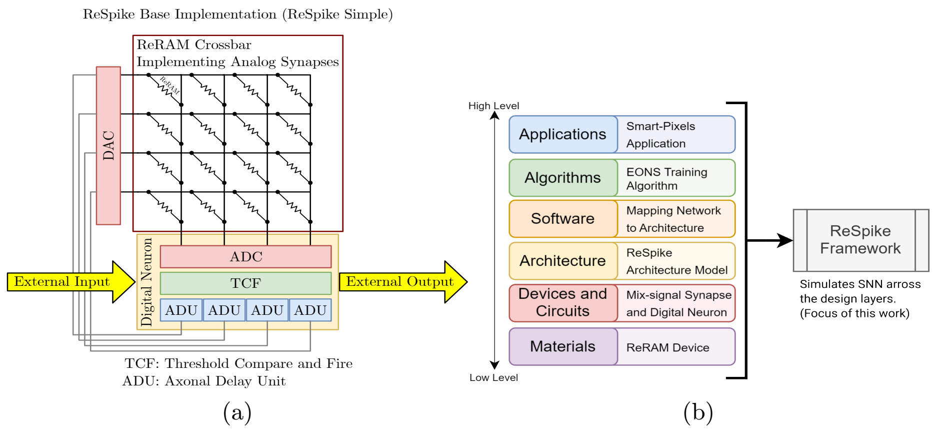
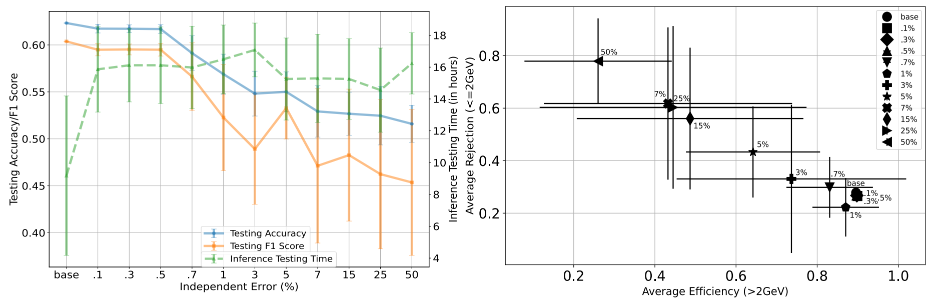


Fig. 2: (a) Block diagram of the ReSpike architecture for a simplified 4×4 crossbar configuration for a 4-neuron SNN, with configurable all-to-all synaptic connections; (b) The co-design effort developing the ReSpike framework—providing support across the full stack from applications, algorithms, software, architecture, circuits and devices, highlighting the selection of models and components used for each layer.

in Figure 2b. This simplified but realistic architecture model allows us to evaluate the performance of the application at the top level in the design stack. The simulator is a behavioral model that can evaluate the accuracy obtained by algorithms developed higher in the stack on lower-level devices.

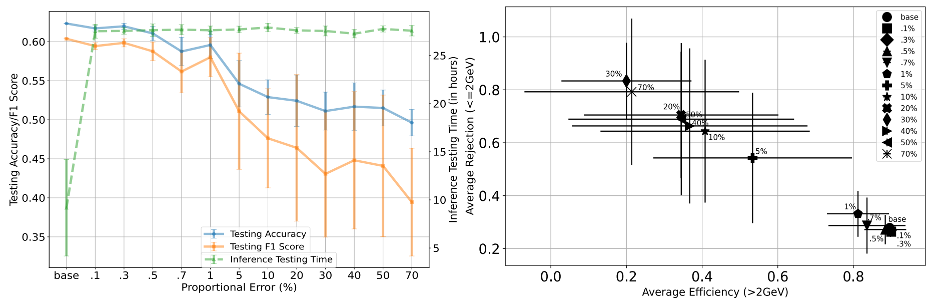
To implement the behavioral simulator of this model, we use the *Analog Core* from CrossSim [19] to model the crossbar component and adjacent peripherals (e.g. ADC, DAC etc.) of the architecture. The digital IF neuron is implemented in a Python class that supports parameter loading, integration, reset, and check for fire functions to simulate its operation. The main ReSpike simple class implements the simple routing and ADU components as a 2D array where neuron fires are inserted at the proper neuron ID and delay value slot. Then, the array shifts in the time dimension, and the last time slot is applied as input to the crossbar. The ReSpike simple class is then wrapped into a processor class that implements a neuromorphic processor fulfilling the processor compatibilities with the TENNLab framework processor interface. This class implements the proper API to load networks, provide input spikes, run the processor, and read back the output spikes. Because the processor is compatible with the TENNLab framework, the target processor in the Smartpixels application can be changed to ReSpike to execute an application using the ReSpike simulator.

The Caspian simulator used when training the Smartpixels application is a configurable digital leaky IF simulator, which is optimized for quick evaluation using a neuromorphic processor event simulator written in C++. To make these Caspian-trained networks compatible with ReSpike, a synaptic to axonal delay conversion to the appropriate threshold and weight range is implemented by creating always-firing-on-input intermediate neurons for every unique synapse



(a) (b)

Fig. 3: Effects of independent error rates on the deviation of (a) accuracy, F1 score, inference testing time; and (b) efficiency vs rejection.



(a) (b)

Fig. 4: Effects of proportional error rates on the deviation of (a) accuracy, F1 score, inference testing time; and (b) efficiency vs rejection.

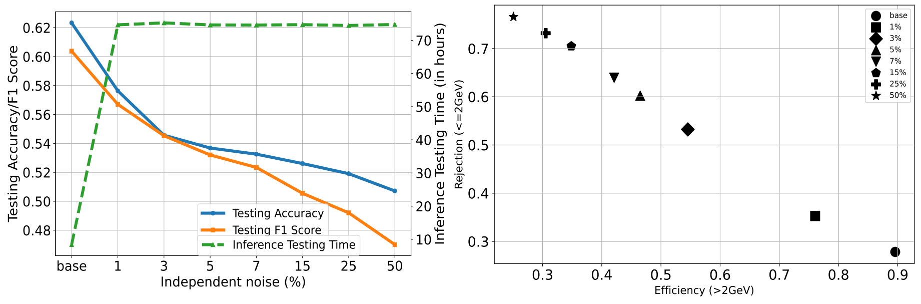
delay value so that the same varying synapse delays can be represented by the different axon delays of the intermediate neurons. To aid in the mapping of neurons to columns in the crossbar, a step is added that compacts the neuron IDs into a contiguous range starting from 0. Finally, a generic parameter value conversion step is used to map the values of the parameters into the correct ranges and into the correct data types. This mapping equation is shown in (1):

$$x' = \texttt{cast}\left(\frac{x - f_{min}}{f_{max} - f_{min}} \times (t_{max} - t_{min}) + t_{min}\right), \qquad (1)$$

where x' is the new parameter value, `cast` is the type cast to the new data type, f_{min} and f_{max} are the previous value range, t_{min} and t_{max} are the new value range, and x in the previous parameter value.

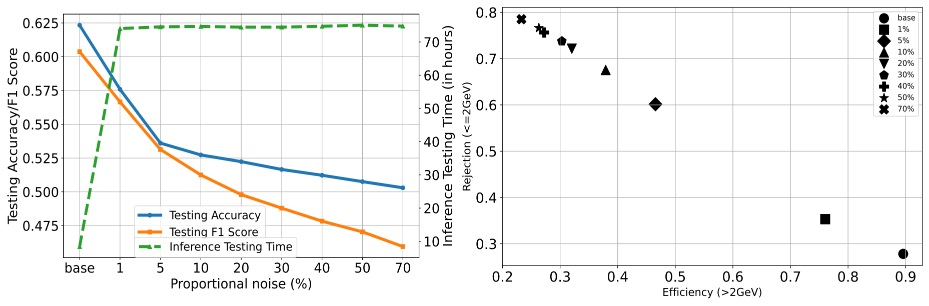
4 Evaluation

In this paper, we evaluate the ReSpike framework using networks and utilities developed for the smart-pixel application. SNNs have a different structure than other artificial neural networks, and custom applications must be developed to leverage emerging hardware architectures as features and implementation details differ without a standard feature set or common set of benchmarks. We compare



(a) (b)

Fig. 5: Effects of independent noise rates on the deviation of (a) accuracy, F1 score, inference testing time; and (b) efficiency vs rejection.



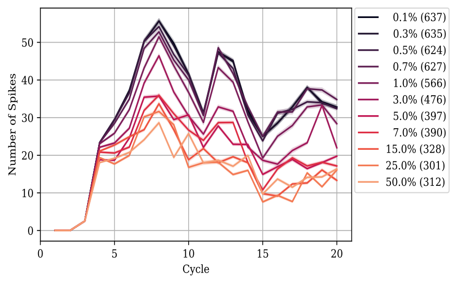
(a) (b)

Fig. 6: Effects of proportional noise rates on the deviation of (a) accuracy, F1 score, inference testing time; and (b) efficiency vs rejection.

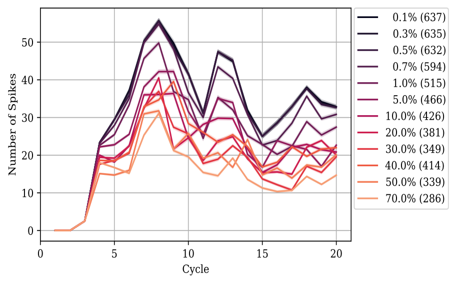
the ReSpike framework with the Caspian simulator and achieve the same inference accuracy for the Smartpixels application with the default ReRAM device configuration that does not incorporate any error or noise variation, and we refer to it as the *base* configuration. The ReSpike framework enables us to use its simulator to quantify the deviation in accuracy and F1 scores While accuracy emphasizes true positives and true negatives, F1 score is used when false negatives and false positives must be observed. for certain percentages of error and noise variability in ReRAM devices. To that end, we perform a sensitivity analysis on several ReRAM device configurations, varying independent error, proportional error, independent noise, and proportional noise (see Section 2.3) and present the corresponding deviation of accuracy and F1 scores. We also report inference testing time and efficiency vs. rejection for each set of experiments.

4.1 Accuracy deviation for independent error

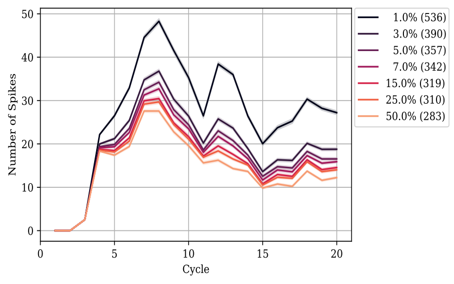
Figure 3(a) presents the accuracy and F1 score deviation for a range of independent error rates. The x-axis provides the independent error rates for the device, the primary y-axis provides the accuracy/F1 score, and the secondary y-axis plots the inference testing time for each case. For each data point, we simulated ten instances of the same configuration and plotted the average value with error bars showing standard deviations. Results show that the accuracy and F1 score



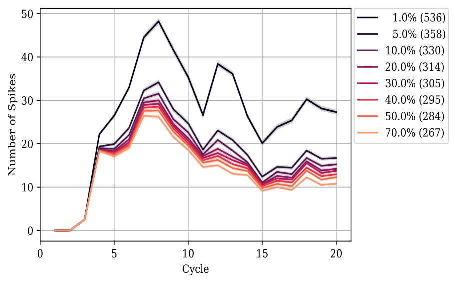
(a)



(b)



(c)



(d)

Fig. 7: Effects of error and noise on spike count. The legend contains the percentage of error or noise and in parentheses the total spike count for each inference averaged across 3,257 samples.

are very sensitive to independent error and gradually slope downward with an increasing rate of independent error having the accuracy dropping down from 0.623 (base) to 0.55 with a 5%, and to 0.515 with a 50% independent error—suggesting higher error rates cause the accuracy to drop, as expected. Configurations with induced error yield a significant increase in inference time as well. Figure 3(b) presents the efficiency vs. rejection plot for the configurations with independent errors. On this chart, an efficiency of 1 (x-axis) with a rejection of 0 (y-axis) indicates that the model always classifies the samples as high p_T. Similarly, an efficiency of 0 and rejection of 1 designate when all samples are being classified as low p_T. The results show that changing the magnitude of independent error affects efficiency vs. rejection points for each configuration, generally moving from lower-right regions to upper-left regions with an increasing error rate. The default decision of the SNN decoder when no neuron fires is to indicate a low-p_T sample, which in turn indicates that the increasing error is leading to a reduction in the overall network's firing output. Hence, as depicted in Figure 3(b), there is an increase in the rejection rate and a drop in the average efficiency.

4.2 Accuracy deviation for proportional error

We also run experiments to analyze the effect of proportional versus independent programming error on accuracy and F1 score deviation. Understanding the sensitivity to the error profile is important because different analog devices tend to have different profiles. In particular, ReRAM, like other resistive memories, tends to have an error profile that is relatively independent of the target state [20]. This has been cited as a disadvantage for CNNs implemented in resistive analog systems when compared to flash, which has a proportional error profile [18]. Our results, as presented in Figure 4(a) indicate that SNNs implemented with resistive devices exhibit very similar trends and range for accuracy and F1 score deviation for proportional errors in comparison to the independent errors. The accuracy drops from 0.623 (base) to 0.546 with a 5% and to 5.15% with a 50% proportional error. Note that this is a very high level of error, which typically may be on the order of 10% [20]. This indicates that for this analog SNN implementation, independent error devices like ReRAM will be equally accurate to proportional error devices like flash. However, inference testing time with induced proportional error is higher and more deterministic. Figure 4(b) also shows similar sensitivity for proportional error on efficiency vs. rejection data in comparison to the independent error variation.

4.3 Accuracy deviation for independent and proportional noise

In addition to programming error, it is important to understand the effect of read noise on accuracy. Physically, read noise represents effects such as random telegraph noise that can manifest as fluctuations in current when held at a constant bias [21]. Figures 5 and 6 present the impact of independent and proportional noise, respectively. It is possible that this noise may or may not be proportional to the target state, hence both are modeled [22]. Introducing higher noise rates reduces the accuracy and F1 scores as expected. Interestingly, as with programming error, there is non a significant difference whether the error is proportional to or independent of state. The efficiency vs. rejection plots show that with increasing noise rates, the model gradually moves from classifying samples as high p_T to low p_T.

4.4 Error and noise's effect on internal spiking behavior

To further investigate the accuracy loss and increasing low-p_T classifications, we ran a separate set of experiments for each configuration to observe the effect of error and noise rates on internal spiking behaviors. Figure 7 presents the number of total spikes fired for the full inference time of 20 cycles for each configuration under observation. Darker lines represent lower error/noise rate and lighter lines represent higher error/noise rate and the legends show total number of spikes for a particular error/noise rate. From the plots we observe that an increasing error and noise rate generally reduces the number of spikes. This loss of spike activity contributes to the observed loss of application performance.

5 Related Work

A few studies propose ReRAM-based solutions for accelerating SNNs. Li et al. [23] investigate the energy bottleneck of ReRAM-based processing in memory and propose ReSiPE, a circuit design supporting the single-spiking multiply-and-accumulate operation. In another study, Ankit et al. [24] propose a reconfigurable energy-efficient architecture with memristive crossbars for deep SNNs and claim to achieve 500× energy efficiency with 300× higher throughput. Jang et al. [25]

explore device-level improvements for ReRAM cells and implement an analog artificial synapse using PCMO-based ReRAM, which is suitable for neuromorphic systems.

Other studies investigate the possibility of using non-ReRAM crossbars for SNNs. Kim et al. [4] propose ADC-less neuromorphic compute-in-memory processor which is fabricated in 28 nm CMOS technology and occupies only a 2.9 mm^2 die area while achieving 92% accuracy for CIFAR-10 with 4-bit input and weights. Bang et al. [26] develop a novel spike prediction technique and a sparse direct feedback alignment technique to reduce the complexity of back propagation delay for on-chip learning of low-energy SNNs. Both of these techniques are implemented in 65 nm process technology, and a 52.1% decrease in energy consumption and a 0.3% accuracy loss are reported.

Long et al. [8] present a ReRAM-based processing-in-memory architecture for accelerating RNNs, characterizes system throughput, area, and power consumption for a 28 nm implementation; and reports a 79$\times$ computing efficiency. Arrassi et al. [27] adopt another approach: an SNN-based computation-in-memory architecture that uses ReRAM devices based on unsupervised spike time–dependent plasticity and supports lightweight online learning and achieves high energy efficiency while maintaining a 95% inference accuracy.

Several studies explore the endurance, reliability, and nonideality aspects of ReRAM cells. Wen et al. [28] focus on endurance degradation aspects of ReRAM cells and propose ReNEW, a novel framework using single-level cells instead of multilevel cells because the write endurance of a single-level ReRAM cell is typically 4–6 magnitudes higher than that of a multilevel ReRAM cell. Dampfhoffer et al. [29] demonstrate that error-aware training can be very effective in mitigating high error rates in neural networks, and static and dynamic errors have different effects on accuracy. The study also reports that SNNs and RNNs are inherently more robust to dynamic errors compared with static errors. Bhattacharjee et al. [30] emphasize the importance of analyzing the effect of intrinsic crossbar nonidealities and show that repetitive crossbar computations through multiple time steps expedite error accumulation, and recommend training SNNs with fewer time steps for better accuracy. Xiao et al. [18] suggest that the solution quality in analog systems can be degraded by noise, process variations, and parasitic resistances. The study conducts and presents an extensive analysis of how nonidealities of analog neural network inference accelerators affect accuracy, examining various parameters such as weight bit slicing, offset subtraction vs. differential cells for handling negative numbers, state-independent and state-proportional errors, and parasitic resistance.

6 Conclusion

This study advances the state-of-the-art with a new architecture for neuromorphic processing leveraging ReRAM crossbar structures and develops a complete framework for all design layers through a rigorous co-design effort. This effort spans over multiple scientific areas from AI, high energy physics to material and device level innovation to develop ReSpike framework that takes advantage of in-memory analog computation at the core, opening up opportunities for highly energy efficient computing. Furthermore, ReSpike provides a platform to explore different device configurations based on emerging materials for various application domains to study their feasibility and accuracy deviation due to nonidealities such as programming errors and read noises. For our experiments, we evaluated SNN of the Smartpixel application on the ReSpike framework and investigated the accuracy deviation for independent/proportional error and noise rates. The results show steep degradation of accuracy for inducing around 5% error/noise and gradual slow downs for higher rates, projecting a downward slope towards a total loss of accuracy. This analysis provides valuable insights on expected accuracy losses with nonideal devices and highlights the needs for error and noise aware training. We also analyze how firing behaviors are changed for various device level configurations. We believe the ReSpike framework will serve as a stepping stone for further investigation of co-design efforts involving novel device, material and computing paradigms enabling design space exploration.

Acknowledgments This research is funded, in part, by the DOE Office of Science Research Program for Microelectronics Codesign (sponsored by ASCR, BES, HEP, NP, and FES) through the Abisko Project with program managers Robinson Pino (ASCR). Hal Finkel (ASCR), and Andrew Schwartz (BES). This research used resources of the Experimental Computing Laboratory (ExCL) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Inyup Kang. The art of scaling: Distributed and connected to sustain the golden age of computation. In *International Solid-State Circuits Conference (ISSCC)*, 2022.
2. Burr, Geoffrey W. et al. Emerging materials in neuromorphic computing: Guest editorial. *APL Materials*, 8(1), 2020.
3. Schuman, Catherine D et al. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, pages 10–19, 2022.
4. Kim, Sangyeob et al. Neuro-cim: Adc-less neuromorphic computing-in-memory processor with operation gating/stopping and digital–analog networks. *IEEE Journal of Solid-State Circuits*, 2023.
5. Davies, Mike and et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 2018.
6. Filipp Akopyan. Design and tool flow of ibm's truenorth: An ultra-low power programmable neurosynaptic chip with 1 million neurons. In *International Symposium on Physical Design*, pages 59–60, 2016.
7. Ige, Afolabi and et al. Analog system high-level synthesis for energy-efficient reconfigurable computing. *Journal of Low Power Electronics and Applications*, 13, 2023.
8. Long, Yun et al. Reram-based processing-in-memory architecture for recurrent neural network acceleration. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2018.
9. Aguirre et al. Hardware implementation of memristor-based artificial neural networks. *Nature Communications*, 15(1974), 2024.

10. Nazeer, Khaleelulla Khan and et al. Language modeling on a spinnaker 2 neuromorphic chip. *arXiv preprint arXiv:2312.09084*, 2023.
11. Plank, James S et al. The tennlab exploratory neuromorphic computing framework. *IEEE Letters of the Computer Society*, 2018.
12. Schuman, Catherine D. et al. Evolutionary optimization for neuromorphic systems. In *Annual Neuro-Inspired Computational Elements Workshop*, 2020.
13. Marinella, Matthew J. et al. Multiscale co-design analysis of energy, latency, area, and accuracy of a reram analog neural training accelerator. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2018.
14. R. Kulkarni, Shruti et al. On-sensor data filtering using neuromorphic computing for high energy physics experiments. In *International Conference on Neuromorphic Systems*, pages 1–8, 2023.
15. Mitchell, J. Parker et al. A small, low cost event-driven architecture for spiking neural networks on fpgas. In *International Conference on Neuromorphic Systems*, 2020.
16. Lee, Matthew Kay Fei et al. A system-level simulator for rram-based neuromorphic computing chips. *ACM Trans. Archit. Code Optim.*, 2019.
17. Schnieders, K. et al. Effect of electron conduction on the read noise characteristics in ReRAM devices. *APL Materials*, 2022.
18. Xiao, T. Patrick et al. On the accuracy of analog neural network inference accelerators. *IEEE Circuits and Systems Magazine*, 2022.
19. T. Patrick Xiao et al. Crosssim: accuracy simulation of analog in-memory computing, 2024.
20. Wan et al. A compute-in-memory chip based on resistive random-access memory. *Nature*, 608:504–512, 2022.
21. F. M. Puglisi. *Noise in Resistive Random Access Memory Devices.* Noise in Nanoscale Semiconductor Devices, 2020.
22. Agarwal, Sapan et al. Resistive memory device requirements for a neural algorithm accelerator. In *International Joint Conference on Neural Networks (IJCNN)*, pages 929–938, 2016.
23. Li, Ziru et al. Resipe: Reram-based single-spiking processing-in-memory engine. In *Design Automation Conference (DAC)*, 2020.
24. Ankit, Aayush et al. Resparc: A reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks. In *Design Automation Conference (DAC)*, pages 1–6, 2017.
25. Jang, Jun-Woo et al. Reram-based synaptic device for neuromorphic computing. In *International Symposium on Circuits and Systems (ISCAS)*, 2014.
26. Bang, Seunghwan et al. An energy-efficient snn processor design based on sparse direct feedback and spike prediction. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021.
27. Arrassi, Asmae El et al. Energy-efficient snn implementation using rram-based computation in-memory (cim). In *International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6, 2022.
28. Wen, Wen et al. Renew: Enhancing lifetime for reram crossbar based neural network accelerators. In *International Conference on Computer Design (ICCD)*, 2019.
29. Dampfhoffer, Manon et al. Improving the robustness of neural networks to noisy multi-level non-volatile memory-based synapses. In *International Joint Conference on Neural Networks (IJCNN)*, 2023.
30. Bhattacharjee, Abhiroop et al. Examining the robustness of spiking neural networks on non-ideal memristive crossbars. In *International Symposium on Low Power Electronics and Design*, 2022.