A Resilience Framework for Synapse Weight Errors and Firing Threshold Perturbations in RRAM Spiking Neural Networks

Anurup Saha, Chandramouli Amarnath and Abhijit Chatterjee
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta GA 30332
Email: {asaha74, chandamarnath}@gatech.edu, abhijit.chatterjee@ece.gatech.edu

2023 IEEE European Test Symposium (ETS) | 979-8-3503-3634-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ETS56758.2023.10174229

***Abstract*—Spiking Neural Networks (SNNs) can be implemented with power-efficient digital as well as analog circuitry. However, in Resistive RAM (RRAM) based SNN accelerators, synapse weights programmed into the crossbar can differ from their ideal values due to defects and programming errors, degrading inference accuracy. In addition, circuit nonidealities within analog spiking neurons that alter the neuron spiking rate (modeled by variations in neuron firing threshold) can degrade SNN inference accuracy when the value of inference time steps (ITSteps) of SNN is set to a critical minimum that maximizes network throughput. We first develop a *recursive linearized check* to detect synapse weight errors with high sensitivity. This triggers a *correction methodology* which sets out-of-range synapse values to zero. For correcting the effects of firing threshold variations, we develop a test methodology that calibrates the extent of such variations. This is then used to proportionally increase inference time steps during inference for chips with higher variation. Experiments on a variety of SNNs prove the viability of the proposed resilience methods.**

***Index Terms*—Spiking Neural Network, Fault Tolerance, Error Correction, Neuromorphic Computing**

I. Introduction

Brain inspired Spiking Neural Networks (SNNs) are an energy efficient alternative to Artificial Neural Networks (ANNs). Although SNNs are highly error resilient, synapse weight errors due to Resistive RRAM (RRAM) device defects and programming errors as well as circuit non-idealities within analog spiking neurons can degrade their accuracy.

Recent research has focused on hardware acceleration and fault tolerance of SNNs [1], [2]. In [3], a framework is proposed to address RRAM reliability soft errors during the training phase of RRAM based neuromorphic systems. In [4], soft error mitigation for digital SNN accelerators is addressed. Fault aware training and mapping (FATM) has been proposed to mitigate stuck at faults and low voltage induced errors in memory [5]. In [6], fault hopping is used to recover accuracy in presence of dead and saturated neuron faults. The work of [7] explores how RRAM crossbar non-idealities impact the accuracy of SNNs. However, for RRAM based mixed-signal SNN accelerators, the problem of synapse weight error detection during inference has remained unaddressed. In this context, we propose *recursive linearized checks* to detect synapse weight errors with high resolution. In the presence of weight errors, if the synapse input to a neuron exceeds a calibrated cutoff, it is set to zero to recover accuracy. We also compensate for neuron circuit non-idealities such as comparator offset and transient noise by re-calibrating the firing threshold of each neuron with an *Effective Firing Threshold (EFT)* and perturbing the *EFT* in proportion to a *Threshold Perturbation Coefficient (TPC)*. We model the TPC of each neuron as a sum of *systematic and random components*.

During SNN inference, the forward pass is repeated a fixed number of time steps *ITSteps*, which determines a tradeoff between SNN accuracy and inference latency. The accuracy of the fault-free SNN stays constant over a wide range of ITSteps. However, in the presence of systematic firing threshold variations (modeled using *Systematic Perturbations Coefficient or SPC*), fault injection experiments show that SNN classification accuracy degrades in *inverse proportion to the magnitude of SPC* when fewer inference time steps (ITSteps) are used to optimize *throughput and inference latency*. A natural question is: Can we calibrate the number of necessary ITSteps for an SNN (minimal for maximizing SNN throughput) using a test procedure for *systematic firing threshold* (SPC) variability? Such a test mechanism is developed in this research and is used to determine ITSteps on a per-SNN basis.

In summary, the key contributions of our work are: **(1)** For RRAM based SNN accelerators, we develop *recursive linearized checks* to detect *synapse weight errors* and use *selective input suppression* (SIS) to recover accuracy. **(2)** We show that systematic component of neuron firing threshold variation degrades SNN accuracy when fewer ITSteps are used. **(3)** We develop a calibration test to estimate the SPC of firing threshold variability and set appropriate number of ITSteps for an SNN implementation (silicon).

II. Preliminaries

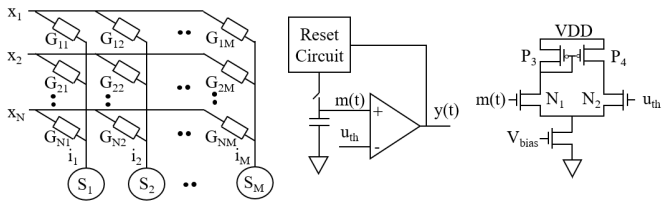
A. Overview and Hardware Implementation of SNN

A Leaky Integrate and Fire (LIF) neuron integrates spikes from input neurons that determine its membrane potential. At time step t, the membrane potential of a LIF neuron is updated as $m(t) = \beta m(t-1) + i(t), \forall t > 1$, where β is the "Leak Factor" ($\beta < 1$) and $i(t)$ is the input to a neuron. The neuron emits a spike if the membrane potential $m(t)$ exceeds its firing threshold u_{th} i.e. the output of the neuron is $y(t) = \mathbb{I}[m(t) > u_{th}]$ where $\mathbb{I}(.)$ denotes the indicator function. Once a neuron emits a spike, its $m(t)$ is reset. Taking this reset into account, the membrane potential can be written as $m(t) = \beta m(t-1) + i(t) - y(t-1)$.

Prior work has proposed RRAM based architectures for SNNs [8]. Fig. 1(a) shows such an SNN architecture, which uses RRAM crossbars to scale spikes by their corresponding

979-8-3503-3634-4/23/$31.00 ©2023 IEEE

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on August 03,2023 at 06:33:51 UTC from IEEE Xplore. Restrictions apply.



(a) Implementation of SNN using Memristive Crossbar (b) Circuit of Spiking Neuron (c) Comparator

Figure 1: Hardware Implementation of SNN

weights. Each weight is mapped to an equivalent conductance. Crossbar inputs are applied along rows (x_n is the input for n-th row) and each column computes the dot product of its row-wise conductances with the crossbar inputs. The input to the m-th neuron S_m is computed as the crossbar output current from the m-th column, $i_m = \sum_{n=1}^{N} G_{nm} x_n$, where G_{nm} is the conductance in n-th row and m-th column. Within a spiking neuron, as shown in Fig. 1(b), a capacitor is used to store membrane potential, $m(t)$. The membrane potential $m(t)$ is compared with firing threshold u_{th} using a comparator. When $m(t)$ exceeds u_{th}, the comparator is high and resets the membrane potential.

B. Fault Model

Synapse Weight Errors: In a crossbar, RRAM devices store synapse weights. Due to manufacturing defects, some devices are permanently stuck at "Low Resistance State" or "High Resistance State". When weights are programmed in a crossbar, the programmed state of a device might differ from its targeted state. These factors lead to the programmed weight being different from the actual weight. We call these errors *Synapse Weight Errors*. In our error injection framework, we inject synapse weight errors based on a *Weight Error Rate* (WER), defined as the probability that the stored value of a synapse weight is different from its actual value. For each of the erroneous weights, we flip a fixed (user-chosen) number of bits at random bit positions in the actual weight.

Threshold Variations: Comparators in spiking neurons do not have expensive input offset cancellation mechanisms [9]. If the input referred offset of the comparator is Ω, then the comparator output is $y(t) = \mathbb{I}[m(t) > (u_{th} + \Omega)] = \mathbb{I}[m(t) > e_{th}]$, where e_{th} is the *Effective Firing Threshold*, defined as $e_{th} = u_{th} + \Omega = u_{th}(1 + \frac{\Omega}{u_{th}})$. For example, inside a comparator as shown in Fig. 1(c), the threshold voltage difference between transistors N_1 and N_2 as well as threshold voltage difference between P_3 and P_4 will appear as an input offset. Deviations in the generation of u_{th} as well as the effects of random noise can be modeled by adding the magnitude of the deviation to the input referred offset. These non-idealities lead to variability in the firing thresholds of each neuron, modeled in simulation by perturbing each neuron's ideal firing threshold. The effective firing threshold is thus calculated as $e_{th} = u_{th}(1 + \frac{\Omega}{u_{th}}) = u_{th}(1 + p)$, where p is the *Threshold Perturbation Coefficient* (TPC). The TPC of the j-th neuron of the l-th layer, $p^{l,j}$, is modeled as the sum of a *Systematic Perturbation Coefficient* (SPC) p_{sys} and a *Random Perturbation Coefficient* (RPC) $p^{l,j}_{rand}$, i.e., $p^{l,j} = p_{sys} + p^{l,j}_{rand}$. We sample $p^{l,j}_{rand}$ from a normal distribution $\mathcal{N}(0, \sigma^2)$.

III. Weight Error Detection and Correction

A. Error Detection

To detect synapse weight errors, we derive a recursively updating check, whose left and right sides are $RCL_1(t)$ and $RCL_2(t)$. In the absence of weight errors we expect $RCL_1(t) - RCL_2(t) = 0$. As a result, if we observe $RCL_1(t) - RCL_2(t) \neq 0$, the presence of synapse weight errors is indicated. A proof of this is detailed below.

Lemma III.1. *We claim that in absence of synapse weight errors, we can compute two vectors v_1 and v_2 such that, the following property holds at time step t:*

$$v_1^T \cdot M^l(t) + \sum_{k=0}^{t-2} v_1^T \cdot Y^l(t-1-k)\beta^k = \sum_{k=0}^{t-1} v_2^T \cdot Y^{l-1}(t-k)\beta^k \tag{1}$$

where $Y^l(t)$ is the output of the l-th layer, $M^l(t)$ is the membrane potential of neurons of l-th layer, W is the weight matrix between layers $l-1$ and l and $I^l(t)$ is the input current to layer l.

Proof. Membrane potential of neurons of layer l,

$$\begin{aligned}
M^l(t) &= \beta M^l(t-1) + I^l(t) - Y^l(t-1) \\
&= \beta[\beta M^l(t-2) + I^l(t-1) - Y^l(t-2)] \\
&\quad + I^l(t) - Y^l(t-1) \\
&= \beta^2 M^l(t-2) + [I^l(t) + \beta I^l(t-1)] \\
&\quad - [Y^l(t-1) + \beta Y^l(t-2)] \\
&\cdots \\
&= [I^l(t) + \beta I^l(t-1) + \cdots + \beta^{t-1} I^l(1)] \\
&\quad - [Y^l(t-1) + \beta Y^l(t-2) + \cdots + \beta^{t-2} Y^l(1)] \\
&= \sum_{k=0}^{t-1} I^l(t-k)\beta^k - \sum_{k=0}^{t-2} Y^l(t-1-k)\beta^k
\end{aligned}$$

Now, we (a) multiply both sides with v_1^T, (b) set $I^l(t-k) = W \cdot Y^{l-1}(t-k)$, $v_2^T = v_1^T \cdot W$ and (c) rearrange the equation to obtain,

$$v_1^T \cdot M^l(t) + \sum_{k=0}^{t-2} v_1^T \cdot Y^l(t-1-k)\beta^k = \sum_{k=0}^{t-1} v_2^T \cdot Y^{l-1}(t-k)\beta^k$$

□

To implement $RCL_1(t)$ and $RCL_2(t)$ (left and right side of Equation 1 respectively), we choose $v_1 = [\frac{1}{n} \ \frac{1}{n} \cdots \frac{1}{n}]^T$ and $v_2 = W^T \cdot v_1$, where n is the number of neurons in the l-th layer. Since $RCL_1(t)$ and $RCL_2(t)$ are updated recursively based on $RCL_1(t-1)$ and $RCL_2(t-1)$, we call the resulting checks *Recursive Linearized Checks*.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on August 03,2023 at 06:33:51 UTC from IEEE Xplore. Restrictions apply.

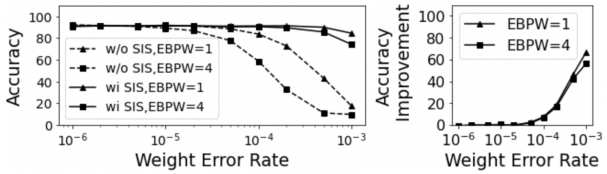
B. Weight Error Recovery

In the presence of severe synapse weight errors which significantly reduce performance, inputs to certain neurons have high values. Our error recovery mechanism aims to reset these erroneous inputs to zero. After training of the SNN is complete, we apply the entire training image set to the SNN. For each layer l, we record the maximum input (I^l_{cutoff}) to a neuron in the layer across all time steps. During inference, if an input to a neuron exceeds this limit at any time step t_0, we set the input to 0 for the remaining time steps, i.e., if $i^{l,j}(t = t_0) > I^l_{cutoff}$, then set $i^{l,j}(t) = 0$ for $t = \{t_0, t_0 + 1, \cdots, T\}$. Since the error recovery mechanism selectively suppresses inputs to neurons, we call this *Selective Input Suppression* (SIS).

IV. Mitigating Threshold Variations

As we reduce the Inference Timesteps (T) to reduce inference latency, variations in the e_{th} values of neurons start to degrade SNN accuracy and accuracy degradation depends only on the systematic component of the variation, quantified by the *Systematic Perturbation Coefficient* (discussed in detail in Section V-C). We define *Cutoff SPC* (p_{cutoff}) such that if the SPC of an SNN chip (p_{sys}) is less than the *Cutoff SPC* (p_{cutoff}), then the accuracy degradation of the chip due to systematic variation is less than δ. If the accuracy of the chip is $Acc(p_{sys})$ and the baseline accuracy is measured as the accuracy of a chip with $p_{sys} = 0$, then the accuracy degradation is $Acc_{baseline} - Acc(p_{sys})$. For a fixed δ, we define p_{cutoff} as: $p_{sys} < p_{cutoff} \iff Acc(p_{sys}) \geq Acc_{baseline} - \delta$. For every SNN chip, a calibration test is used to estimate whether the SPC of an SNN chip (p_{sys}) is within the Cutoff SPC (p_{cutoff}). An SNN chip is deemed a "good device" if $p_{sys} < p_{cutoff}$, otherwise it is deemed a "bad device". Since we cannot measure p_{sys} directly, we measure the *Calibration Spiking Rate* (CSR or μ), of the Device Under Test (DUT) and compare it with the *Cutoff Calibration Spiking Rate* (μ_{cutoff}) to predict whether the device is "good" or "bad". The inference time steps (T) of good devices are set to a small value. For bad devices, T is set to a large value to maintain classification accuracy. The value of p_{cutoff} is calculated from simulation.

The test methodology has three steps: **Step 1**, The Calibration Image Set $\mathcal{C}$ is built by choosing 64 random images from the training image set. We monitor the outputs of all neurons in the output layer. The set of output layer neurons is referred as the *Monitored Set* $\mathcal{M}$. We apply each image $c \in \mathcal{C}$ to the DUT. **Step 2**, From the output response of the Monitored Set, we measure the CSR of the DUT, defined as: $\mu = \frac{\sum_{c\in\mathcal{C}} \sum_{m\in\mathcal{M}} \sum_{t=1}^{T} y_m(t)}{|\mathcal{C}||\mathcal{M}|}$, where $y_m(t)$ is the output of the m-th neuron in $\mathcal{M}$ at timestep t when the c-th image from $\mathcal{C}$ is applied to the DUT and $|.|$ here denotes the set cardinality operator. **Step 3**, The calibration test predicts a DUT as "good"($p_{sys} < p_{cutoff}$) if the measured CSR of the DUT is more than Cutoff CSR, i.e., $\mu > \mu_{cutoff}$, Otherwise it predicts the DUT as "bad".



(a) (b)

Figure 2: (a) Accuracy with and without SIS for VGG5 (b) Accuracy Improvement Due to SIS (EBPW = Number of bits flipped in every erroneous weight)

V. Results

A. Experimental Setup

The proposed approach was validated on two Convolutional Spiking Neural Networks (CSNN). We designed a CSNN using the Lenet-5 [10] architecture on the MNIST dataset [11] and a CSNN using the VGG5 architecure on the FashionMNIST dataset [12]. These networks use rate encoding and are trained with SNNTorch Framework [13] using cross-entropy loss and Surrogate Gradient Descent. Fault injection and error correction simulations were run using PyTorch. During training, we used an initial learning rate of 3×10^{-4} and exponential learning rate decay with a decay factor of 0.9. Throughout our experiments the "Leak Factor" β was kept at 0.95. We evaluate Selective Input Suppression using the metric *Critical Weight Error Rate* CWER, defined as the WER for which accuracy of the faulty networks is 5% lower than the fault-free accuracy. To evaluate the calibration test, we define the *Characterization Accuracy* (CA) Score as: CA Score $= \frac{\text{Number of correctly categorized devices}}{\text{Total number of devices under test}}$

B. Weight Error Correction

We evaluate the performance of SIS on a VGG5 SNN trained on the Fashion MNIST dataset. We use 16 bit fixed point representation for weights, with 1 sign bit, 5 integer bits and 10 fraction bits. In every faulty weight we inject a fixed number of bitflips, denoted by Erroneous Bits Per Weight (EBPW). We evaluate the impact of SIS with EBPW = 1 and 4. The fault-free accuracy of the network is 91.9%. Fig. 2(a) shows the accuracy of the CSNN with and without SIS. For EBPW = 1, SIS improves CWER from 7×10^{-5} to 8×10^{-4} and for EBPW = 4, SIS improves CWER from 2×10^{-5} to 4×10^{-4}. Fig. 2(b) shows that SIS achieves up to 64% more accuracy than a faulty SNN without SIS under high error rates.

C. Threshold Variation Resilience

In absence of threshold variations, the inference latency of VGG5 SNN improves by 3x (by reducing T from 25 to 8) at the expense of compromising accuracy by just 1.5% (90.4% from 91.9%). LeNet-5 shows a similar trend, where reducing T from 25 to 8 leads to just 1.1% loss in accuracy (97.7% to 96.6%).

We evaluate the impact of threshold variation on the accuracy of VGG5 SNN for a range of T. We use p_{sys} to quantify systematic variation. We sample $p_{rand} \sim \mathcal{N}(0, \sigma^2)$ and use 3σ to quantify random variation. First, we inject systematic variation alone. To vary effective firing threshold e_{th} from

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on August 03,2023 at 06:33:51 UTC from IEEE Xplore. Restrictions apply.

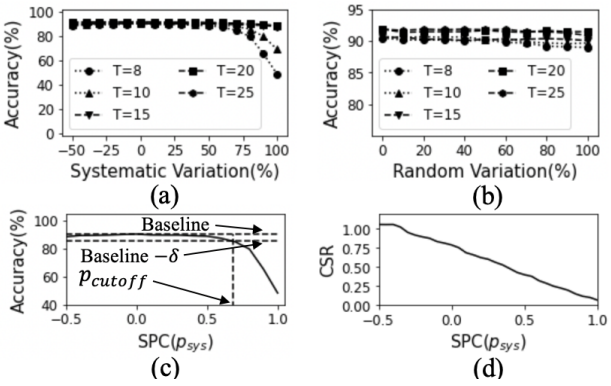


Figure 3: (a) Accuracy vs systematic variation for VGG5 (b) Accuracy vs random variation (c) Measurement of Cutoff SPC from Accuracy vs SPC plot (d) CSR vs SPC

$u_{th}/2$ to $2u_{th}$, we sweep p_{sys} from -0.5 to 1. Fig. 3(a) shows the that for $T \geq 15$, accuracy is hardly impacted by systematic variation. However, in the presence of maximum systematic variation, accuracy degrades by 42% for $T = 8$, and 21% for $T = 10$. Next, we evaluate the impact of random threshold variations alone. We sweep 3σ from 0 to 1 and set $p_{sys} = 0$. Fig. 3(b) shows that accuracy is not sensitive to random variations for all values of T. For $T = 8$, the maximum accuracy degradation is seen to be just 1.5%. We

Table I: Impact of SPC and RPC on Accuracy

p_{sys}	0	0.50	0.50	0.75	0.75
3σ	0	0	0.50	0	0.25
Accuracy	90.4	88.7	88.6	83.0	82.9

also evaluate the impact of systematic and random threshold variations together for $T = 8$. Table I shows that for systematic variation of 75% ($p_{sys} = 0.75$), a random variation of 25% degrades accuracy by only 0.1% (from 83% to 82.9%). Thus, in the presence of both systematic and random variations, the accuracy degradation can be approximated from only the level of the systematic variation. From the previous three experiments, we conclude that *threshold variation leads to significant accuracy loss only for low values of T and the accuracy loss can be accurately estimated using only the SPC.*

For the calibration test designed for VGG5 SNN, we set $T = 8$ and fix $\delta = 5\%$. From Fig. 3(c), we obtain $p_{cutoff} = 0.68$. Fig. 3(d) shows CSR monotonically decreases with p_{sys} when p_{rand} is 0. So, CSR is used to estimate p_{sys}. To compute μ_{cutoff}, we generate 1000 devices with $p_{sys} = p_{cutoff}$ and 3σ is sampled uniformly from $[0, 1]$. We compute Cutoff CSR as the maximum CSR measured from these 1000 devices. We obtain $\mu_{cutoff} = 0.2953$. A device is a "good device" if its $p_{sys} < 0.68$, whereas the calibration test *predicts* a device as "good device" if its $\mu > 0.2953$. Hence a device can fall in one of four categories: (a) True Positive (TP) (b) True Negative (TN) (c) False Positive (FP) and (d) False Negative (FN). To measure CA score of the test, we generate 10000 SNN devices using $p_{sys} \sim uniform(-0.5, 1)$ and $3\sigma \sim uniform(0, 1)$. Table II summarises the performance of the characterization test. It achieves a CA score of 97.78%.

Table II: Results of Calibration Test with 10000 devices

		True Characterization of DUT	
		Good	Bad
Characterization of DUT using Calibration	Good	TN=7639	FN=1
	Bad	FP=221	TP=2139

For an SNN with LeNet-5 architecture, systematic variation reduces inference accuracy to 62.8% and 82.8% for $T = 8$ and 10 respectively (from 97%). For $T \geq 15$, accuracy is not impacted by systematic variation. Accuracy is not degraded due to random threshold variations as well. For Calibration Test with $T = 8$, we obtain $p_{cutoff} = 0.8$ (assuming $\delta = 5\%$) and $\mu_{cutoff} = 0.2375$ from simulation. The predictions of the calibration test can be categorized as: $TP = 1336$, $FP = 308$, $FN = 4$ and $TN = 8352$, giving a CA score of 96.88%.

VI. Conclusion

Resilience strategies for synapse weight errors and circuit non-idealities that alter firing threshold have been discussed. Selective Input Suppression is used to mitigate synapse weight errors. A calibration test is proposed to optimize the latency and accuracy of SNNs in presence of threshold variations.

Acknowledgement

This research was supported in part by School of ECE, Georgia Institute of Technology and by the U.S. National Science Foundation under Grant: 2128149.

References

[1] T. Spyrou *et al.*, "Reliability analysis of a spiking neural network hardware accelerator," *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2022.

[2] A. Carpegna *et al.*, "Spiker: an fpga-optimized hardware accelerator for spiking neural networks," *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2022.

[3] A. M. S. Tosson *et al.*, "A study of the effect of rram reliability soft errors on the performance of rram-based neuromorphic systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 11, 2017.

[4] R. V. W. Putra *et al.*, "Softsnn: low-cost fault tolerance for spiking neural network accelerators under soft errors," *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022.

[5] ——, "Respawn: Energy efficient fault tolerance for spiking neural networks considering unreliable memories," *IEEE Conference on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[6] T. Spyrou *et al.*, "Neuron fault tolerance in spiking neural networks," *Design, Automation Test in Europe Conference Exhibition*, 2021.

[7] A. Bhattacharje *et al.*, "Examining the robustness of spiking neural networks on non-ideal memristive crossbars," *International Symposium on Low Power Electronics and Design*, 2022.

[8] T. Bohnstingl *et al.*, "Accelerating spiking neural networks using memristive crossbar arrays," *IEEE International Conference on Electronics, Circuits and Systems*, 2020.

[9] R. Serrano-Gotarredona *et al.*, "A neuromorphic cortical-layer microchip for spike-based event processing vision systems," *IEEE Transactions on Circuits and Systems*, vol. 53, no. 12, 2006.

[10] Y. Lecun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[12] H. Xiao *et al.*, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv:1708.07747v2*, 2017.

[13] J. K. Eshraghian *et al.*, "Training spiking neural networks using lessons from deep learning," *arXiv preprint arXiv:2109.12894*, 2021.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on August 03,2023 at 06:33:51 UTC from IEEE Xplore. Restrictions apply.