Received November 16, 2021, accepted December 7, 2021, date of publication December 16, 2021, date of current version December 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3136193

Computational Failure Analysis of In-Memory RRAM Architecture for Pattern Classification CNN Circuits

NAGARAJ LAKSHMANA PRABHU AND NAGARAJAN RAGHAVAN, (Member, IEEE)
Engineering Product Development (EPD) Pillar, Singapore University of Technology and Design, Singapore 487372

Corresponding author: Nagarajan Raghavan (nagarajan@sutd.edu.sg)

This work was supported by the Agency for Science, Technology and Research (A*STAR) Brain Efficient Nanomechanical Artificial Intelligence Computing (BRENAIC) Programmatic Research under Grant A18A5b0056.

ABSTRACT Power-efficient data processing subsystems performing millions of complex concurrent arithmetic operations per second form part of today's essential solution required to meet the growing demand of edge computing applications, given the volume of data collected by real-time Internet-Of-Things (IoT) sensors. Adding to it, the in-memory computation designed as memory and processing elements on a single wafer has enabled promising performance improvement in terms of computational power savings by avoiding the memory wall created while accessing the memory array. The Resistive RAM (RRAM), with its simple metal-insulator-metal (MIM) structure, proves to be a very appealing candidate for in-memory computation given its ultralow switching power and its Complementary Metal Oxide Semiconductor (CMOS) process fabrication compatibility. However, despite all advantages, the resistive switching (RS) phenomenon in RRAM has an inherent stochastic variability. On the algorithmic side, convolution neural networks (CNN) have gained popularity in image classification applications, and the network's architecture is memory-intense in nature for memorizing the trained weights. Hence, an RRAM-based CNN system will pave way for a power-efficient image classification system on the edge. Accounting however for the inherent variability in RRAM (inter-device and intra-device), the accuracy of CNN's prediction is surely expected to drop. This motivates us to quantify the impact of RRAM variability on the CNN trained weights and classification accuracy (prediction loss). In this study, we have constructed a Look-Up-Table (LUT) based model for encoding wide current compliance (2μA to 250μA) 65nm CMOS 1T1R OxRAM's (TiN/HfO_2/Hf/TiN) resistive variability into CNN's trained weight in a digital regime. The RRAM resistance encoded trained weights are in turn used here to simulate the two extreme CNN architectures, namely, Fully Serial System (FSS) and Fully Parallel System (FPS). The architectures' prediction variability trends are quantified given its current compliance, RRAM resistive variability, CNN's convolution matrix sizes (5×5, 3×3, 1×1, and 1×1 max pool), the total number of layers in the CNN as well as the input image pixel size.

INDEX TERMS Resistive RAM, convolution neural network, look-up-table, in-memory computation, image classification, Internet of Things, complementary metal oxide semiconductor.

I. INTRODUCTION

Video data analytics has transformed today's industries by providing more significant and precise insights into the process automation side to improve productivity, personal safety and building strong customer relations in various services and solutions being provided today [1]. Cloud-based video analytics has proven to be colossally powerful in entrenching features for greater scalability on computational power, data redundancy, quick deployment, and regulatory compliance. However, it fails to perform for applications with low internet bandwidth and mission-critical on-the-fly decision making [2]. Edge computing is a new approach to network architecture, and they are quite powerful, capable of gathering

The associate editor coordinating the review of this manuscript and approving it for publication was Shadi Alawneh.

and processing more data than ever before by which the data processing function is relocated closer to where the data is collected and analyzed in real-time. Every snapshot of data does not get transmitted over to a central server for processing, by which the network latency is drastically reduced with enhanced performance for real-time applications such as convolution neural network (CNN) based image classification applications. Placing the memory and computational logic on a single chip reduces the computational power by avoiding the von-Neumann memory wall created during a memory read and write operation [**3**]. This performance improvement fosters the development of in-memory computation architecture and revitalizes data center's increasing unacceptable levels of power utilization (which require intense expensive cooling solutions right now).

It is essential to have an insight into today's in-memory computation approach to reduce computation power using various emerging memory devices. On a large commercial server, almost $\sim$50% of the total operation power is consumed by the off-chip Dynamic Random-Access Memory (DRAM). A bulk memory transfer using the existing DRAM operation principle is proposed by Onur *et al.* to improve power usage [**4**]. Furthermore, a 10T bit-cell-based Static Random-Access Memory (SRAM) holding 1-bit filter weights exhibiting dot-product mathematical operation with $>$98% accuracy for classifying MNIST hand-written text with better energy efficiency by reducing data transfer has been demonstrated by Biswas *et al.* in Ref. [**5**]. Other than the commercial memories DRAM and SRAM, a variety of emerging memory devices have also been fabricated and studied for in-memory operation, some of which are discussed here.

A 2D-array of processing elements using Spin Transfer Torque RAM (STT-MRAM) based in-memory architecture was proposed by Agarwal *et al.* [**6**] and tested with significant energy savings of 1.75X over the commercial memory for an image classification application. A Boolean NAND operation demonstrated with the other emerging memories, such as Ferroelectric RAM (FeRAM) and Phase Change RAM (PCRAM), by assigning binary codes to different physical device states, has also paved the way for in-memory computation applications [**7**]. Adding to the current in-memory study, quantum-dot cellular automata (QCA) have emerged as a new breed of nanoelectronics with significant performance improvement over the conventional Von Neumann architecture [**8**]. Recent studies presented a fully scalable in-memory Resistive RAM (RRAM) architecture of an edge-aware-anisotropic filtering algorithm aimed at computer vision applications, demonstrating reduced memory operation by 64% to 92% resulting in power saving of up to 75% [**9**]. As CMOS technology approaches its physical limits, NVM-based neuro-inspired computing chips offer a promising route, for which increasing research effort seen among device engineering and an extensive review shown in [**10**].

A. OVERVIEW OF RRAM SYNAPSE-BASED CNN SYSTEM

As a basic preamble to researchers who may not be working in the semiconductor device domain, the RRAM device is a simple metal-insulator-metal (MIM) structure that switches back and forth between conducting and insulating states when subjected to a moderate voltage/electric field at two different polarities and exhibits non-volatile property. Hence this phenomenon where a dielectric instantaneously changes its (two-terminal) resistance between two distinguishable states under the application of a moderate to strong electric field is termed as resistive switching (RS) [**11**]. The features enabling RRAM's popularity amongst the emerging memory technology candidates are its (i) simple device structure that can be integrated into today's CMOS fabrication environment and compatibility with back-end-of-the-line (BEOL) process thermal budgets, (ii) 3-D cross-point architecture with a memory cell area of $4F^2$, (iii) Low-cost non-volatile memory with an operation speed as low as tens of nanoseconds per bit, (iv) Per device multi-bit memory storage feature and (v) significantly high endurance cycle of 10^6 with ultra-low switching energy in the range of a few pJ [**12**].

The RS mechanisms can be broadly classified into electrochemical metallization memory (ECM), in which the conductive switching path is formed by the metal cations in an electrochemical process. The second category is valence change memory (VCM) (also known as oxygen vacancy RAM (OxRAM)), with conductive switching achieved by the oxygen vacancies generated with an active electric field. Typically, ECM devices are fabricated with one active metal electrode (Cu, Ag, Ni) and VCM with one inert metal electrode (Pt, Ru, Au or Ir), when sandwiched in an MIM structure [**13**]. We limit our study here to VCM since it exhibits several orders lower switching characteristics when compared to ECM (metal migration through the dielectric media is always more power intensive than simple bond breaking induced oxygen vacancy generation). The real-world metal-oxide RRAM applications are notably restrained due to the cycle-to-cycle and device-to-device variability inherent in the device switching mechanism. The stochastic nature of the oxygen vacancy generation, migration, and recombination result in the formation of non-uniform conductive filament (CF) with varying size, shape and/or pattern; hence the filamentary formation and rupturing process results in leftover oxygen vacancies inside the tunneling gap region leading to stochastic atomic/ionic motion, making variability a property intrinsic and inherent to the metal-oxide RRAM [**14**]–[**16**]. Even in a controlled device manufacturing process, a single RRAM device exhibits intra device cycle-to-cycle variability while subjected to multiple switching cycles for data storage read and write due to the stochasticity in the filamentation process [**17**].

The Convolution Neural Network (CNN) pervades as a successful edge computation algorithm for on-the-fly image classification applications [**18**]. The CNN was designed as a first-order computation function to achieve image

TABLE 1. Listing of other RRAM based neuromorphic simulation reports focusing on the variability induced performance degradation.

Ref #	First Author & Affiliation	Device Studied / Modeling Approach	ML Architecture Explored	Inference Dataset	Methodology	Strengths and Limitations
[19]	Yan Liao Tsinghua University, China (2020)	Quantum Point Contact Model*	3-Layers of Fully connected Neural Network (NN) (784-200-10)	MNIST	Parallel-vector-matrix-multiplication and weight update process on RRAM crossbars.	Discusses extensively on RRAM I–V & programming nonlinearity, Crossbar asymmetry, and tuning voltage sensitivity, but the neuromorphic simulation is only performed on a small network compared to today's deep networks.
[20]	Tuomin Tao Zhejiang University, China (2021)	Quantum Point Contact Model*	100 × 3 two-layer fully connected NN	MNIST '0','1' & '2'	Cross Entropy loss function and Backpropagation employed to update the weights of synapses.	Two training mechanisms improve device nonlinearity, and a parasitic model demonstrating crossbar interconnects shown with and without the write verification. Comparatively small Crossbar array size used.
[21]	Guillem Boquet, Universitat Auto`noma de Barcelona (UAB) (2021)	Conductance Memristor Model with Normal distribution imperfection*	4 Layers of Quantized Neural Network (728–1000–1000–10)	MNIST & Traffic data	Quantified Neural Network in a Crossbar RRAM array.	Classification not performed on real-time practical neural network.
[22]	Andrew J.Ford, University of Cincinnati, USA (2020)	Gaussian Distributed Model*	Single layered Perceptron (SLP) configured on a Crossbar array	MNIST	SLP trained with SGD Via Backpropagation to study the Gaussian resistance RRAM model's contributing the prediction accuracy loss as a function of device variability.	The study clearly shows most CNN algorithms have a surprisingly high tolerance to variable weight updates and initializations, but the variability is confined to a mathematical model rather than a real-world physical model.
[23]	Parthasarathi Pal, National Cheng Kung University, Taiwan (2021)	Ti/Pt / Al2O3 (4nm) / Ta2O5 (4nm)/ Ti/Pt	2 Layers of Multi Layer Perceptron (MLP) NN	MNIST	Fabricated RRAM device variability is modeled and further simulated in a Neural Network.	The lowest current compliance (CC) explored was 500µA, much lower CC is not explored.
[24]	Markus Fritscher, Friedrich-Alexander-Universita ̈t Erlangen-Nu ̈rnberg (2021)	Verilog-A Analogue Model*	5 Layered Perceptron	MNIST	Various RRAM Oxide thickness variability and impact of its prediction accuracy studies.	Classification accuracy for different current compliance not reported.
[25]	Yimao Cai Peking University, China (2020)	Kinetic Monte Carlo defect switching model accounting for non-idealities*	6 layered – modified CNN	MNIST and CIFAR-10	Random Telegraph Noise (RTN), Early Stage Fluctuation (ERF), and Retention Degradation (RD) modeled RRAM simulated on a NN Crossbar array.	Device-level time dependent fluctuation and temperature variation modeled and NN prediction accuracy studied and confined to a single current compliance.
[26]	Yide Du Shanghai Jiao Tong University (SJTU) China (2020)	TiN/ Ta_2O_5(5nm)/ TiN	Large scale DNN LeNet, AlexNet and VGG16	MNIST, Fashion, CIFAR-10	Lognormal RRAM distribution model is used for DNN simulation. Various RRAM oxide thickness variability and impact of its prediction accuracy studies.	Classification accuracy for different current compliance not reported.
[27]	Hoang-Hiep Le, National Cheng Kung University, Taiwan (2020)	Ni/HfO_2(5nm)/ TiN	Quaternary-weight MLP	MNIST	RRAM resistance variability programmed as an analytical model and simulated in MLP.	Ultra-low-power RRAM device based MLP prediction accuracy trend studied, without considering the additional power consumed by the peripheral circuits ADC/DAC.
[28]	Myonghoon Kwak, Pohang University of Science & Technology (POSTECH) South Korea (2021)	Cu/HfO_2 (3nm)/Ta(3nm)/ Cu_2	4 layers of MLP (784-250-125-10) NN	MNIST	Threshold-triggered training scheme with interfacial and filamentary RRAMs programmed in a complementary fashion.	Classification not performed on real-time NN & Peripheral circuits overheads not considered for mixed-signal crossbar array circuit.
[29]	P.Freitas, Liverpool John Moores University, UK (2020)	TiN/TaN(10nm)/ TaO_x(20nm)/ Ta_2O_5(4nm)/ TiN	3 layers of MLP (784-30-10) NN	MNIST	RRAM conductance distribution function is used to reproduce the simulation variability distribution.	Classification not performed on real-time NN & Peripheral circuits overheads not considered for mixed-signal crossbar array circuit.

TABLE 1. ***(Continued.)* Listing of other RRAM based neuromorphic simulation reports focusing on the variability induced performance degradation.**

Ref #	First Author & Affiliation	Device Studied	ML Architecture Explored	Inference Dataset	Methodology	Strengths and Limitations
[30]	Xiaochen Peng, Georgia Institute of Technology, USA (2019)	TaO_x/HfO_x	VGG-8 and ResNet-18	CIFAR-10	Compute-In-Memory engine (Integrated NeuroSim with Pytorch and Tensorflow).	The study extensively shows the device retention model, ADC quantization effects, and trade-offs between inference accuracy, energy efficiency, throughput, area, and memory utilization. The framework operates the RRAM in the analog regime, and hence further to this study, one can explore operating RRAM in a digital domain for better noise immunity in real-world IoT application.
[31]	J. Doevenspeck, Katholieke Universiteit Leuven / imec, Belgium (2021)	$TiN/Ta(5nm)/TaO_x(3nm)/TaN$	LeNet-5 and ResNet-20	CIFAR-10	Conductance distributed transformation applied to derive the weight distribution with encoded device variability.	RRAM devices operated on an analogue crossbar array, for which no additional circuit overhead such as ADC and DAC is studied. Moreover, the inference accuracy of quantized weights results in degraded performance when compared with the results of actual weight used in a mission-critical such as autonomous vision system.
Our Current Work		$TiN/HfO_2(5nm)/Hf(10nm)/TiN$	Fully Parallel and Fully Serial Deep Inception CNN	ImageNet ILSVRC	Digital Look-Up-Table methodology with device variability encoded false bit trained weights used for inference.	Our study shows the impact of device variability on a practical deep CNN by considering the two extreme architectures and analyzing the prediction trend for a wide current compliance RRAM device by operating the device in a digital domain.

**These studies are proposed with generic RRAM model, which can be applied to any device, but no specific device was shown as a proof of concept.*

classification on complex patterns swiftly with edge filter-based matrix convolution technique followed by a fully connected neural network with an activation function to identify the pre-trained image patterns. The CNN is a biologically inspired model with a memory-centric algorithm for memorizing pre-trained patterns or images like that of a human brain. This memory intense CNN's deep structure perpetuates in-memory computation technique; hence it paves the way for using Resistive RRAM as synapses or memory units creating integrated devices with ultra-low power application capability [32]. As mentioned earlier, the RRAM is subject to stochastic variability, which induces performance degradation on the end application; hence, conducting a study to quantify this performance degradation is critical.

B. VARIABILITY STUDY IN NEUROMORPHIC CIRCUITS

Applying RRAM as a synaptic memory in a CNN has triggered interest among various research groups to investigate the prediction accuracy loss given the device variability and a consolidated review of such studies is provided in Table 1. The table discusses the different RRAM devices, machine learning (ML) simulation methodology used along with their merits and remarks. Filamentary non-ideal RRAM model programmed into an ML simulation architecture in a shallow analog crossbar array to study Neural Network (NN) prediction accuracy variability with MNIST handwritten text is demonstrated in the work of Refs. [19]–[21]. However, today's CNN is packed deeply with many computational intense hidden layers to improve prediction performance; hence this brings interest to study the prediction variability trend in a complex and more practical CNN. Significant efforts are also seen in characterizing new device stacks to improve the RRAM device RS effects. Such studies extensively showcase the device-level time-dependent fluctuation and temperature variations as a function of the crossbar prediction accuracy [22]–[26]. These studies are confined to a single current compliance. Hence, adding to the random noise variability models, a further study with varying and wide current compliance will give more insight into the design considerations of a non-ideal RRAM for low power IoT (Internet-Of-Thing) applications. Substantial research works are also evident in modeling the oxide RRAM conductance data into an MLP (Multi-Layer Perceptron) as a mixed-signal crossbar to perform low power and highly efficient MAC (Multiplier and Accumulator) circuit [27]–[31]. There is a clear computational benefit in terms of lower power consumption for the convolution operation, but the additional peripheral circuits such as analog-to-digital (ADC) and digital-to-analog (DAC) converters add up to the complexity of the design for processing the mixed-signal data. Hence, exploring digital RRAM synapses is useful to design optimized and compact circuits.

With this in mind, we propose here a Look-Up-Table model to extrapolate the electrical resistance variability of a small RRAM device array fabricated in the lab and encode it into fully trained commercial CNN trained weights for quantifying the impact of RRAM variability on the CNN

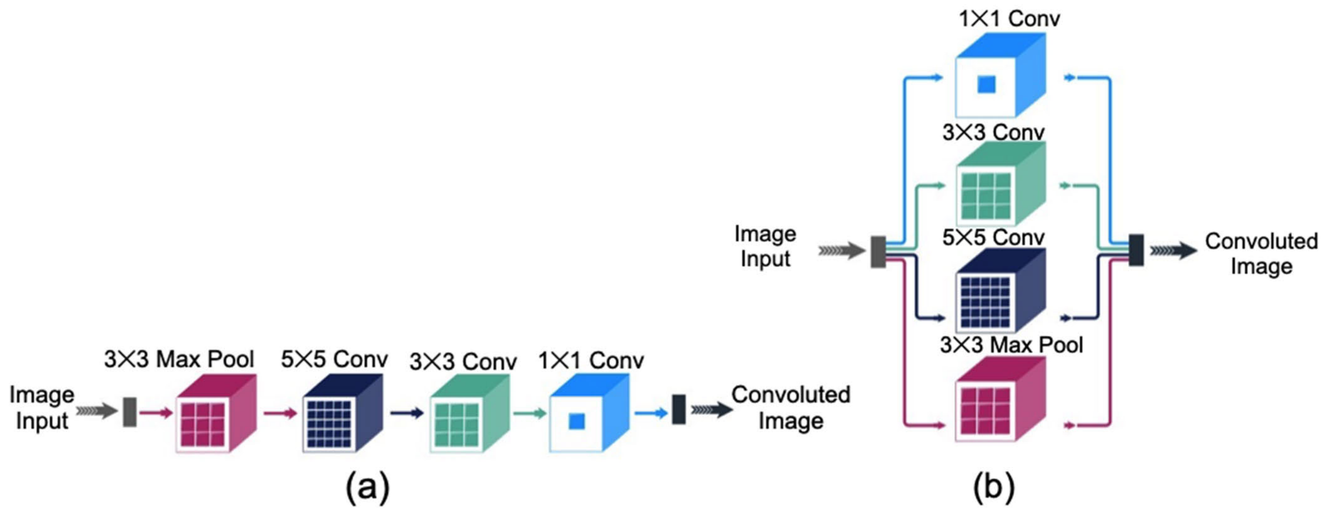


FIGURE 1. **(a) Fully Serial System (FSS) architecture with sequence of varying size convolution operations followed by a *maxpool* function. (b) Fully Parallel System (FPS) architecture with the similar-sized convolution and *maxpool* operators as in FSS, but all connected in parallel.**

prediction accuracy in a digital regime by encoding the logic bits 1's and 0's as HRS and LRS states of the given RRAM. Today's popular CNN algorithms such as AlexNet, GoogleNet, ResNet (etc..) are constructed with cascaded structures of fully parallel and serial blocks of convolution modules with different matrix sizes to enhance prediction accuracy. Hence in this work, we have also considered the two extreme possible architectures, namely the fully serial (FSS) and fully parallel (FPS) systems, as shown in Fig. 1., which are used in the hidden layer of today's CNN architecture. The novelty of this work aims to study the impact of RRAM variability on CNN's prediction accuracy for a wide range of current compliances of RRAM ranging from 2 μA all the way to 250 μA (corresponding to large variations in the shape and size of the conducting filaments) by considering the two extreme convolution architectures (FSS and FPS) of a practical CNN network. It is essential to examine and quantify the hardware (RRAM) variability and its impact on the prediction accuracy for these extreme architectures, which we have recorded in this work. The current work is a marked improvement over recent past studies, which are aimed at only assessing the mean value of the prediction error brought about by RRAM device variability with a very limited range of operating current compliance. Any hardware device used to hold the software-trained weights in a neuromorphic system will have its variability due to the inherent variations in the complex multi-step fabrication process. Hence, resistance variations in the high and low resistance states (HRS and LRS) are an inherent property of any RRAM device used as a synaptic weight. The variability could originate within a device during multiple switching cycles due to the stochastics of the filamentary switching process and also from device-to-device across the wafer due to process metrology issues. This variability must be accounted for in order to quantify the compromise in accuracy of any hardware-based edge application. In the context of RRAM, the compliance used for switching the device will heavily impact the distribution of the two states of the resistance, especially their tails, which will overlap a lot more at lower compliances with a lower memory window. This overlap of the HRS and LRS tail distributions will in turn result in increasing faulty encoding of 0s and 1s in the synaptic weight representation, which will affect the entire network's classification accuracy.

The structure of this paper is as follows. Section II presents the simulation methodology followed for encoding the resistance value of RRAM into the CNN trained weights and compute the prediction error loss between the software and hardware (RRAM) trained weights. Section III discusses the results obtained and the trends observed for a wide current compliance RRAM encoded trained weights while being applied in two extreme convolution architectures in today's CNN. We conclude our work in Section IV after a summary and inference based on all the analysis carried out.

II. SIMULATION METHODOLOGY FOR NEUROMORPHIC APPLICATION

A. ENHANCING PREDICTION ACCURACY WITH EDGE DETECTION BASED INCEPTION FUNCTION

An insight into the generalized CNN architecture unveils the underlying two fundamental mathematical matrix functions, namely the inception and fully connected (FC) neural network. Both have a parallel structure and perform pattern classification by manipulating every image pixel concurrently in the given input image. Among the two functions, the inception persists as intense computation and memory operation, while the FC neural network layer consists of an activation function to compare the similarity between the manipulated image pixel and the trained data to signal to the next neuron with the likelihood of the current image against the trained data [**33**]. The inception function consumes $\sim$80% of the overall resources when compared to the FC

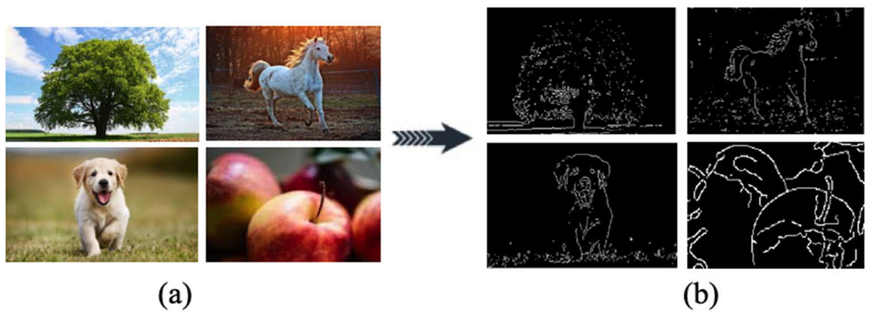


FIGURE 2. The left-side (a) images are applied with an edge filter based on the Sobel algorithm, resulting in an edge enhanced pattern. The images on the left are colorful with all possible RGB pixels. In contrast, (b) the right-side images are reduced to bi-color code with a predominant pattern and can be further used for pattern classification.

neural network in a commercial CNN [**34**]; hence, we limit our simulations and variability examination to the memory intense inception operation alone. The inception is a function of matrix convolution and max-pooling operations; here, both belong to a class of edge enhancement techniques for effective pattern classification application.

Edge enhancement is a type of image processing used to enhance a pattern's edge in an image to improve its apparent sharpness. The edge filter works by increasing the contrast of the edge or boundary between the subject and the background. This effect results in bright and dark highlights on both sides of the edges in the image, making the pattern more prominent from the background. The edges are highlighted by mathematically manipulating every pixel with pre-defined filter data, as illustrated in Fig. 2. A convolution is an advanced edge enhancement technique, wherein a single resultant vector is derived from sum of the products of two given matrices, namely the input image pixel matrix and filter data matrix. Hence, to convolute an image with a pixel size of $m \times n$, we chose a convolution window size of $n \times n$, which is smaller than the given image. The convolution process is repeated on the given image by shifting the convolution window by k pixels, known as the *stride*. The generalized convolution formula for an $m \times n$ matrix is given by Eqn. (1),

$$R_{kl} = \sum\nolimits_{i=1}^{m} \sum\nolimits_{j=1}^{n} I_{(i+s,j+s)} F_{ij} \tag{1}$$

where:

R = Resultant matrix obtained by convoluting input image matrix (I) and trained weight filter matrix (F); Size of R matrix is $m \times n$.

I = Input image matrix, holding the image's RGB pixel values that are to be classified or identified; Size of I matrix is $m \times n$.

F = Filter matrix consists of trained weights obtained by back propagation based stochastic gradient descent algorithm and trained for a considerable amount of labeled data set; Size of F is $n \times n$, which is less than or equal to the size of R matrix.

$n \times n$ = Defines the convolution operation window size in the given 2D image, which is smaller than I's size.

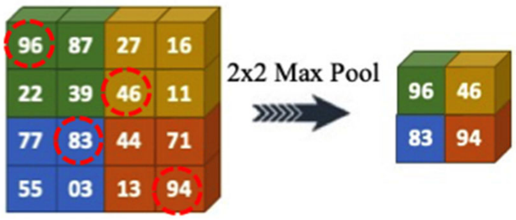


FIGURE 3. An example of a 2 × 2 max pool operation is shown. A 4 × 4 matrix is divided into four 2 × 2 matrices and down sampled to half the size by neglecting all the low-magnitude elements and retaining the highest magnitude element among the 2 × 2 group.

s = Stride is the delta between the location of two consecutive convolution windows.

ij = The row (i) and column (j) number of the given filter matrix and image matrix elements.

kl = The row (k) and column (l) element of R matrix; Size of R matrix represents the convolution operation size.

NOTE : Here, padding is the number of pixels (value equal to zero) added to an image when the kernel or trained weight of a CNN is convoluted to keep the convolution window's size as $n \times n$ when the stride value approaches the end of the given matrix.

The Max pooling is a discrete quantization technique for down sampling the input pixel matrix, and this dramatically reduces the over-fitting at the FC neuron activation layer. Today's inception employs max pooling function in all the layers to improve prediction accuracy and reduce computation power on the following layers significantly by down sampling the image [**35**]. The generalized Max pooling function is shown in Eqn. (2) and a simple 4 × 4 max pool example is shown in Fig. 3, wherein a 4 × 4 matrix gets down sampled to 2×2, where the maximum value of each 2×2 array is copied to the new max pooled matrix as shown.

$$V = Max\,[A_{n\times n}] \tag{2}$$

where:

V = Maximum value from the given input matrix.

A = Input matrix; Size of $n \times n$

A sequence of edge-enhancing convolution functions with varying size filter matrix layered in different groups called as *Inception*, has become the building block of today's

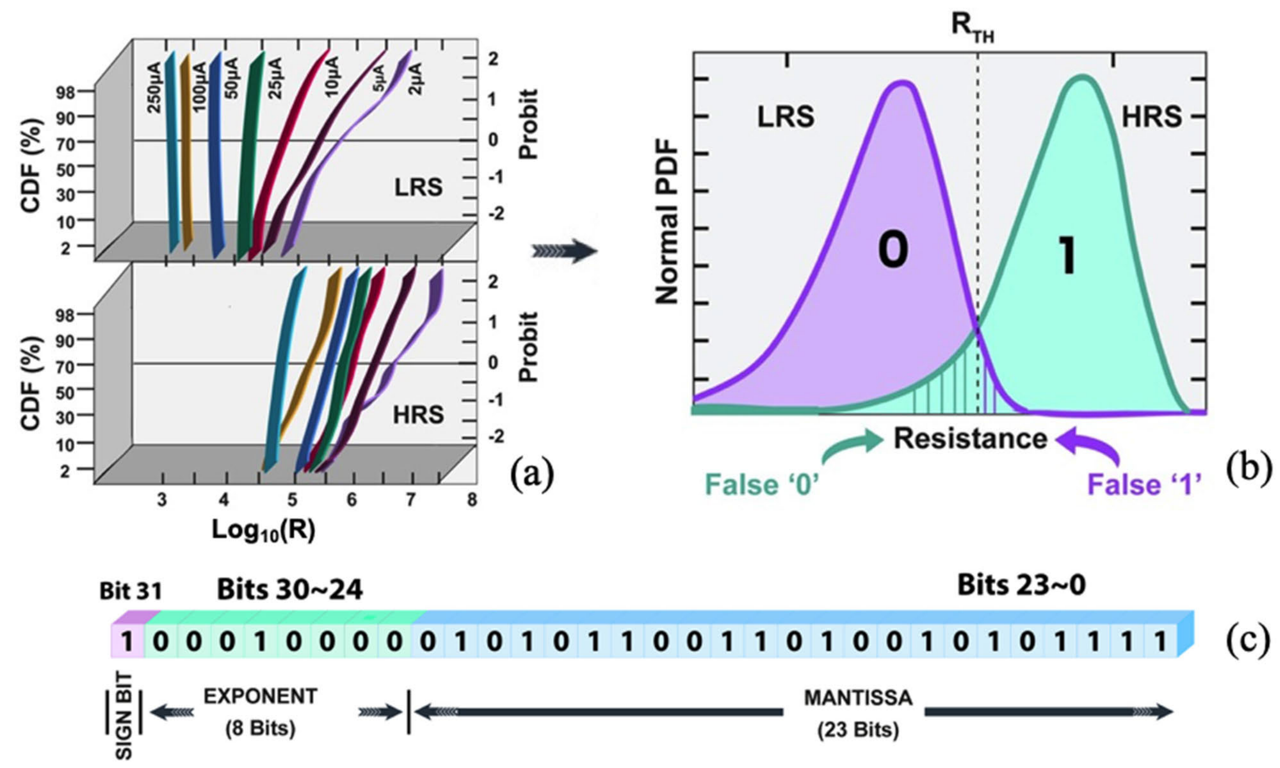


FIGURE 4. (a) Cumulative distribution of cycle-to-cycle resistance variability trend of CMOS 1T1R OxRAM stack comprising TiN/HfO_2/Hf/TiN, extracted from the work of Fantini *et al.* [33] (b) Normal distribution plot shows the false logic-0 and false logic-1 at the intersection of LRS and HRS distribution; this represents the RRAM variability encoded into the trained weight. (c) The trained weights are represented in a 32-bit floating-point format, which consists of the mantissa (23-bits), the exponent (8-bits), and the sign bit (1-bit).

CNN with enhanced prediction accuracy rate. For the given inception network, the smallest convolution function is 1×1 pixel size, and the largest is 5×5-pixel size. Hence, we see that the convolution matrix resolution is maintained low to keep the edge sampling rate as high as possible for improved edge detection. Thus, with higher computational resolution, the prediction performance is increased along with the cost of higher computational power. The matrix convolution of sizes 1×1, 3×3, 5×5, and max pool 3×3 is the most used operation in today's CNN architectures. We considered these four common and prominent computation functions for our simulation study by connecting them in a fully serial and fully parallel sequence, as shown in Fig. 1.

These two architectures are the two extreme operation sequences seen in today's CNN. We conduct our variability simulation study on these extreme architectures by encoding the RRAM's electrical resistance as Look-Up-Table (LUT). This will allow us to quantify the impact of RRAM's variability on these two extreme architectures and analyze the error propagation trend from layer to layer of CNN for the given OxRAM's wide range of current compliance.

B. ENCODING SCHEME OF RRAM RESISTANCE VARIABILITY ON SYNAPTIC DATA

We have extracted the resistance distribution data from one of the most comprehensive RRAM variability data sets published to date, by Fantini *et al.* from IMEC [**36**]. In that study, the authors report OxRAM resistance data distributions for a wide range of compliances $\rightarrow \{2, 5, 10, 25, 50, 100 \text{ and } 250\}$ μA. A 65nm CMOS 1T1R OxRAM stack comprising of TiN/HfO_2/Hf/TiN of size 20×20 nm^2 cell was fabricated in their work. The device was subject to several switching operations from Low Resistive State (LRS) to High Resistive State (HRS). The cycle-to-cycle resistance variability trend was measured and plotted as a lognormal distribution, which we have replotted in Fig. 4a.

The resistive variability trend of LRS and HRS for the wide switching current compliance data set was extracted to construct the CNN trained weight based LUT model so as to further analyze the CNN prediction performance degradation trend. The work in Ref. [**36**] records the device switching for 200 read/write cycles of different current compliance as a cumulative resistance distribution, as shown in Fig. 4a. Here, we further use the linear extrapolation technique to synthesize a significantly large data set at the very low and very high percentiles (tail ends) from the measured 200 cycles of switching data. The X-axis represents the LRS and HRS resistance distribution of different current compliance, and the Y-axis is the device cumulative probability. The LRS data represents logic-0 and HRS represents logic-1; hence, we extrapolate the LRS curve and HRS curve towards the negative X-axis until both the curves of the specific I_{comp} intersect. Thus, we extrapolate a more realistic large data set using the actual 200 read/write cycle data set, which shows a significant overlap between corresponding LRS and HRS. This overlap results in a prediction accuracy error, which is encoded into the LUT CNN, trained weight model. The technique used to encode RRAM resistance into CNN trained weight is explained in the following sections. The formula for the linear extrapolation is shown in Eqn. (3). By using any two specific endpoints on the given resistance distribution curve, namely (X_1, Y_1) and (X_2, Y_2), the new data points are extrapolated by running the formula in a loop. Here the endpoint (X_1, Y_1) is the initial point on the curve, and it is fixed, while the (X_2, Y_2) is the last computed value from every iteration. In general, the approximated overlap between the specific LRS and HRS distributions after extrapolation is relatively low for high I_{comp} and vice versa for low I_{comp}. This is due to the lower number of oxygen vacancy defects and higher relative change in defect count for smaller conducting filaments during the SET and RESET transitions.

$$X_n = X_1 + \frac{Y_n - Y_1}{Y_2 - Y_1}(X_2 - X_1) \tag{3}$$

where:

$X - axis$ = Resistance distribution for the given current compliance.

Y−axis = Cumulative probability value of the resistance for any specific current compliance

X_n, Y_n = Resistance data points to be extrapolated

X_2, Y_2 = Represents last computed resistance data

X_1, Y_1 = Initial Resistance data points

C. GENERATING LUT MODEL FOR VARIABILITY TREND ANALYSIS

The GoogleNet CNN's synaptic weights are trained for 1000 image categories with 1.2 million images and was constructed with an Inception architecture to enhance the prediction accuracy by localized object detection with comparatively fewer hyperparameters of 25million compared against 60million of its predecessor AlexNet [37]. The Inception architecture is formed by stacking smaller CNN's on top of each other to create a deeper network. The basic blocks of such Inception framework are 1 × 1, 3 × 3, 5 × 5 Convolutions, and 3×3 max pooling. The GoogleNet consists of 9 symmetric inception layers, namely 3a, 3b,4a–4e, 5a, and 5b. Here, we extract GoogleNet's weights from the 3a inception layer for our further simulation, and the underlying convolution sizes of this layer are 1 × 1, 3 × 3, 5 × 5, and 3 × 3 max pool.

Each and every trained weight in the CNN is represented in a 32-bit floating-point format, which consists of mantissa (23-bit), exponent (8-bits), and signed bit (1-bit), as illustrated in Fig. 4(c). As proposed in our past work, the normal resistance distribution of RRAM is encoded into the mantissa part of CNN trained weights as logic-0 with LRS data and logic-1 with HRS data (based on a threshold resistance value, R_{TH}, i.e. if $R < R_{TH}$, it is logic "0" and if $R > R_{TH}$, it is logic "1") and both HRS and LRS resistance data are extracted from the extrapolated data set. Our previous work explains the lookup table approach in greater detail in Ref. [**38**]. For every trained weight of the 3a inception layer of GoogleNet, we "encode" 1000 points of logic-0 or logic-1 from the given RRAM resistance distribution plot for any given current compliance. This results in a LUT with 1000 varying mantissa inheriting the RRAM variability for a given single trained weight. Fig. 4(b) shows a false logic-0 and false logic-1 at the intersection of LRS and HRS distribution. This represents the RRAM variability, and these incorrectly encoded "0" and "1" are embedded into the software trained weight mantissa as error data resulting in prediction accuracy drop. Hence, with the proposed LUT technique, the RRAM electrical variability is encoded into the given CNN software trained weight and further used to simulate and quantify the impact of RRAM variability on the prediction accuracy rate if the software trained CNN were to be implemented on the "edge".

The FPS and FSS inception schemes are developed using the Keras framework. Keras is a deep learning software written in Python programming language, running on TensorFlow's machine learning platform. The proposed Keras-based FSS and FPS inception architectures are implemented as two parallel computational pipelines as shown in Fig. 5. The first pipeline works with an original GoogleNet trained weight called software trained weights, and the second pipeline takes the RRAM resistance encoded data, referred to as the hardware trained weights. The difference between the above two inception pipeline's outputs gives the actual prediction error drop. This is due to RRAM resistance variability (and false '0' and '1' as shown in Fig. 4(b)) encoded into the actual trained weight. For the given input image from the ImageNet–ILSVRC (ImageNet Large Scale Visual Recognition Challenge) data set, we simulated 5000 cycles for a single image, and this procedure was repeated for all the I_{comp} data set of the OxRAM. The obtained prediction and power variability trend is discussed in the following sections.

III. RESULTS AND DISCUSSION

The prediction error trends obtained from the computational difference between the two pipelines using software and hardware trained (RRAM) weights are shown in Figs. 6 – 8. The computational difference is a relative error difference between the software and hardware pipeline outputs.

A. IMPACT OF CONVOLUTION SIZE ON PREDICTION ERROR FOR VARYING COMPLIANCE

The mean prediction error for three different convolution operations (1 × 1, 3 × 3, and 5 × 5) based on our simulation framework is shown in Fig. 6. Note that we did not simulate the max pool block of the hidden layers because there is no arithmetic manipulation involved in the max pool function. With a comparative operation module, the max pool operation takes the maximum value in the pixel group and drops the other low-value pixels. Hence, we compare the various convolution operations for the given wide range

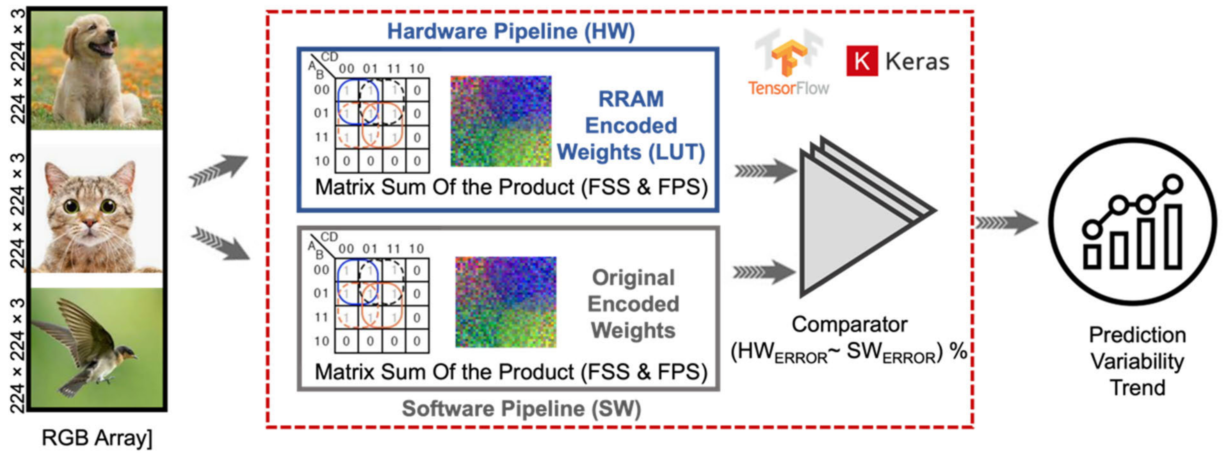


FIGURE 5. Illustration of the simulation framework designed for an input image size of 224 × 224 × 3. The FSS and FPS inception computations are performed in parallel in the software (actual trained weights) and hardware (RRAM variability based binary encoded weights) pipelines. The respective outputs are then passed through an error comparator to compute and plot the relative error trend in prediction accuracy, as shown.

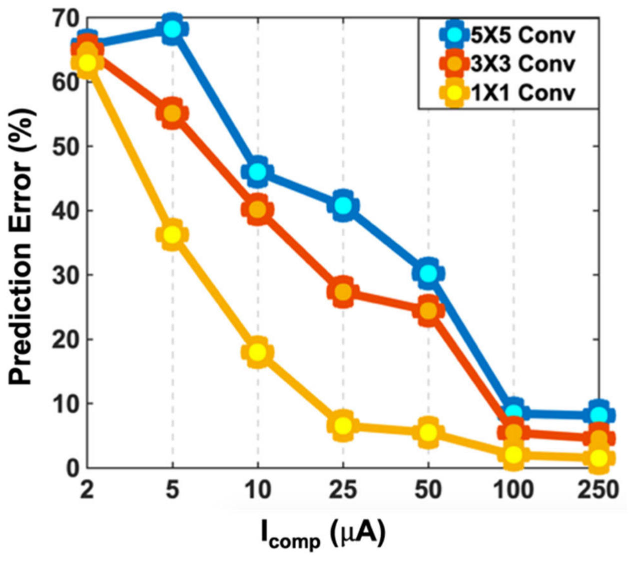


FIGURE 6. Trend of mean error for the three-convolution operations (1 × 1, 3 × 3, and 5 × 5), with prediction error rate on (Y-axis) and the current compliance on (X-axis).

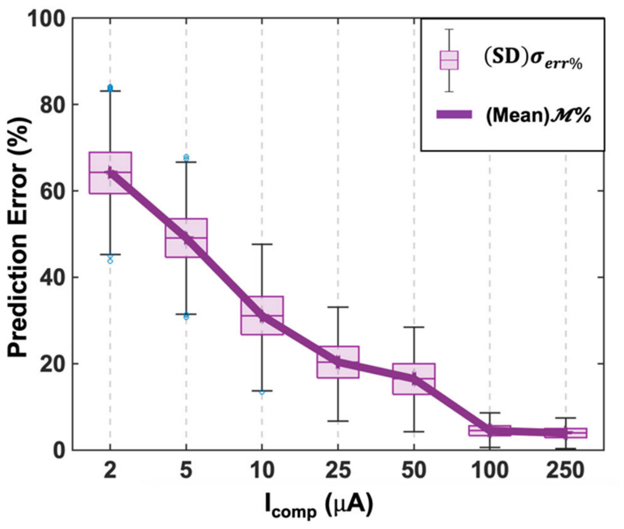


FIGURE 7. Prediction error trend for the Fully Parallel System (FPS) inception architecture for I_{comp} ranging from 2μA to 250μA.

of current compliances. The predictive error trend for the 1 × 1 convolution starts from ~63% prediction error for 2μA I_{comp}, and a steep decline in the error rate is observed as I_{comp} increases to 5μA, 10μA and 25μA, respectively. Subsequently, the slope becomes insensitive to the higher current compliances, as shown in Fig. 6. While we compare the mean trend among the three-convolution operations (1×1, 3 × 3, and 5 × 5), the magnitude of the error value for the 3 × 3 and 5 × 5 convolution are 1.5 times and 2 times higher than the 1 × 1 convolution operation. The size of the 3 × 3 and 5 × 5 matrices is obviously higher than that of the 1 × 1 convolution matrix, and so is the probability of false bits getting encoded in the computation. This explains the higher prediction error for increasing size of convolution operation as more RRAM devices need to be used to construct the synapses of the hidden layer.

The rise in prediction error for lower I_{comp} can again be explained based on Figure 4(a). The memory window between HRS and LRS drastically reduces for lower I_{comp}, which results in higher overlap between the resistance state distributions. Moreover, at low I_{comp}, the conducting filament is very narrow with very few defects in it and hence, for repeated switching, the relative change in defect count within the filament results in a wide variation of the resistance state. In other words, the probability of false-0 and false-1 rise steeply as we move from 50-100 μA (which falls into the ‘‘hard breakdown’’ regime for dielectrics) to 2-5 μA (traditionally referred to as ‘‘soft breakdown’’). Furthermore, subjecting the device to consecutive SET and RESET for

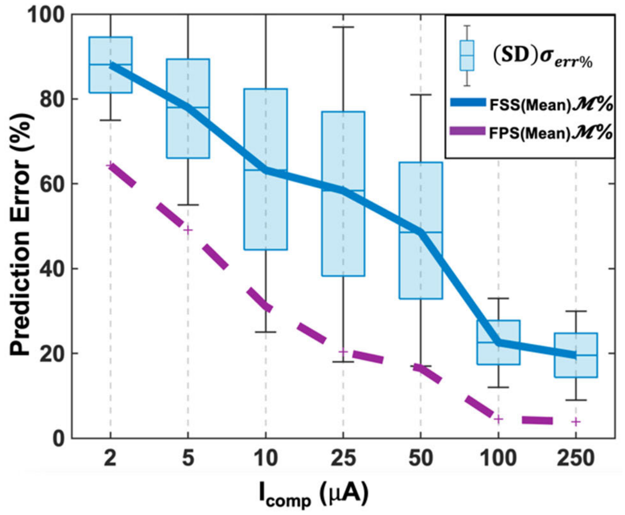


FIGURE 8. Prediction error variance and mean for the Fully Serial System (FSS) inception architecture The mean error trend for Fully Parallel Sequence (FPS) is plotted here as a purple dotted line for comparative analysis.

many thousands of cycles, the defect count and defect density spread are also affected by the gradual reduction in the mobility of the oxygen vacancy defects, resulting in further memory window overlap, more so again at low I_{comp}. These effects get absorbed into the encoded trained weights and further amplified in the convolution layer's matrix multiplication and summation function resulting in the trends as shown in Figures 6-8.

B. COMPARING ERRORS IN FULLY SERIES AND FULLY PARALLEL ARCHITECTURES FOR VARYING COMPLIANCE

The prediction error trend for the two extreme inception architectures (FPS and FSS) originating from RRAM variability encoded trained weights is shown in Figs. 7 and 8. Here, the standard deviation ($\sigma_{err\%}$) of the relative error trend between the hardware and software pipelines is obtained by simulating a single image over 5000 repetitive stimulation cycles are plotted. Every simulation cycle uses a random entry from the LUT with RRAM variability encoded weights to classify the given image. From the simulation results of the FPS architecture, it is clear that both the variance and the mean of the error decreases for higher I_{comp}. (Fig.7). It is also important to note that the error flattens out to a finite non-zero value $\sim 5\%$ for $I_{comp} > 100\mu$A, which suggests that it is unnecessary to operate the device at even higher powers as further reduction in relative error is too low to justify the use of a higher power consuming architecture for the edge application. We can achieve at least $\sim$60% power reduction by choosing 100μA instead of 250μA.

From Fig. 8, it is worth noting that the prediction error mean is much higher, and variance is also comparatively higher for the FSS architecture. The FSS system is constructed with a serial chain of convolution operations where the error trend significantly gets convoluted due to matrix multiplication in every block of the serial chain. This explains why the mean error for the serial architecture is much higher than for the parallel one (shown by dotted purple lines). Surprisingly, the variance for 2μA and 5μA is comparatively smaller than for the higher I_{comp}, contrary to our logical thought flow. It should be noted that this is purely an artifact because of the definition of the error, which cannot be more than 100%. The upper percentile error bars have already hit their ceiling of 100% for $I_{comp} \sim \{2, 5\}$ μA.

C. TRADE-OFF BETWEEN PREDICTION ERROR AND POWER CONSUMPTION

The error variance and the power consumption per memory bit for the two extreme convolution operations, namely 5×5 and 1×1, is studied and plotted in Fig. 9.

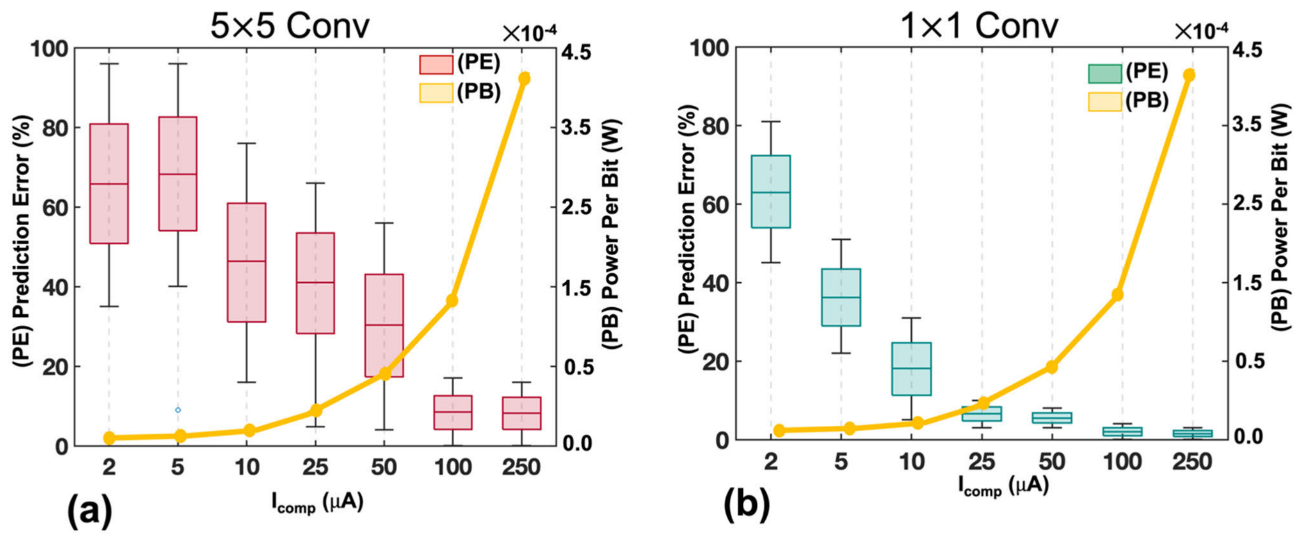
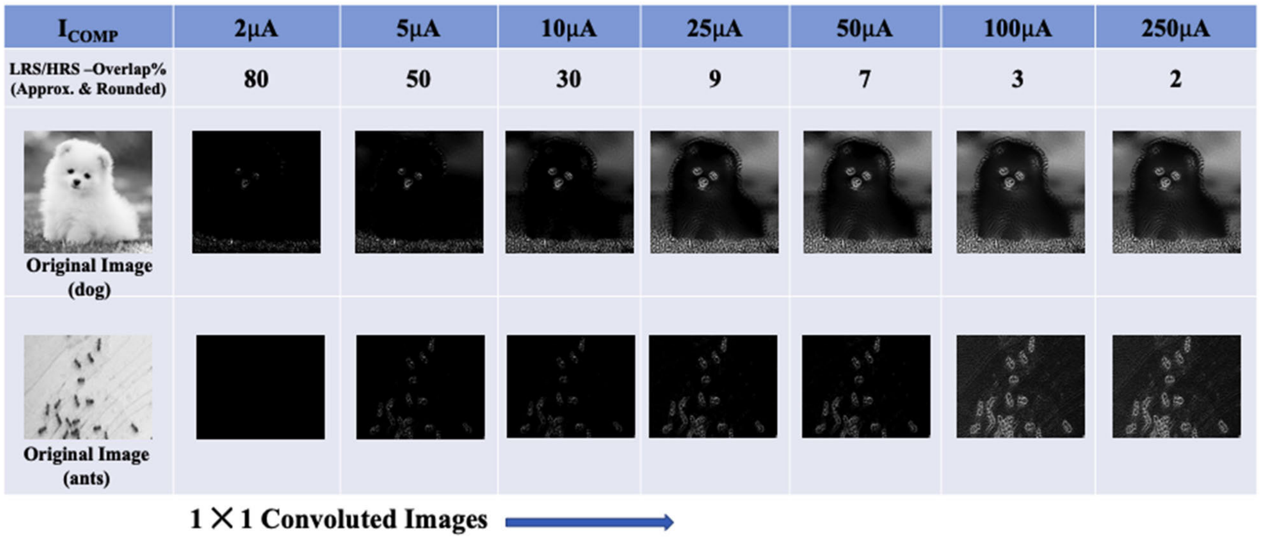


FIGURE 9. Prediction Error (PE) obtained for 5000 simulation cycles for 5 × 5 Conv (a) and 1 × 1 Conv (b) operations plotted on the Y1-axis. The simulations were repeated for a wide range of I_{comp} values, as shown in the X-axis. The Y2-axis represents the Power per Bit (PB) trend for the two extreme convolution architectures.



I_{COMP}	2μA	5μA	10μA	25μA	50μA	100μA	250μA
LRS/HRS –Overlap% (Approx. & Rounded)	80	50	30	9	7	3	2

FIGURE 10. Illustration showing a 1 × 1 Convolution applied to a dog and ant's images. The convolution operation is repeated using encoded trained weights for different I_{comp} of RRAM operation ranging from 2μA to 250μA. it is worth nothing that the edge enhancement is much clearer towards the right for higher current compliance, while for very low I_{comp} $\sim$ 2uA, the edges are hardly discernible.

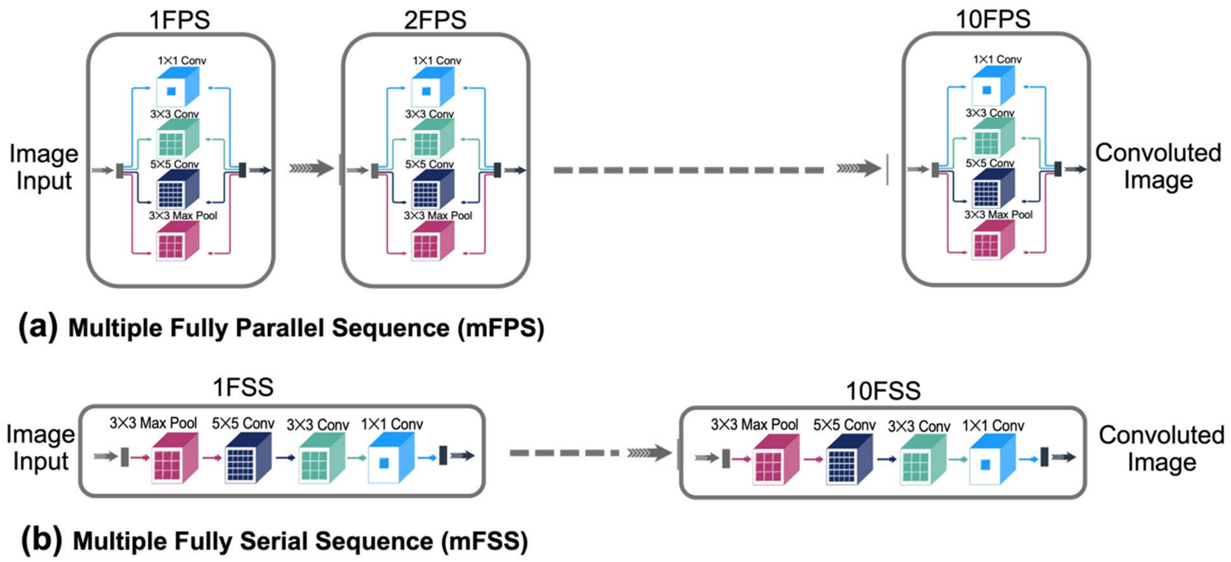


FIGURE 11. Illustration of the two extreme deep CNN architectures explored for prediction error variability quantification – (a) multiple FPS and (b) multiple FSS.

Here, Y_1-axis on the left shows the error distribution for 5000 cycles of all the given I_{comp}, and Y_2-axis on the right shows the corresponding power per bit trend for the synaptic weight with an operating voltage of 1.5V. As we know, for higher I_{comp}, the memory window is wider and with less overlap between LRS and HRS; hence the prediction error spread is less, but with a high computation power budget. For discussion, let us consider the error spread for 5 × 5 and 1 × 1 at 2μA and 250μA; the variables of the function responsible for the error spread are the memory window overlap and the convolution matrix's size. While analyzing the intra curve of the 5 × 5 convolution function for the large I_{comp} of 250μA from Fig. 9(a), the prediction error spread is approximately 4X smaller than for $I_{comp} = 2\mu$A. Furthermore, the power consumption at 250μA is about 15X higher than that for 2μA (see yellow line in Fig. 9(a)). In comparison, the prediction error magnitude for 250μA 1 × 1 convolution is approximately 8X lower than for $I_{comp} = 2\mu$A, as shown in Fig. 9(b).

For a comparative analysis on power saving, let us consider an edge device performing a 1 × 1 convolution operation on a video stream of 224 × 224 pixel and powered by a coin battery of 130mAh. The 1 × 1 convolution matrix requires 23-bits of mantissa and uses 23 RRAM devices to hold the trained weights (1 device per bit, assuming binary digital RRAM). With this scenario, we can compute that the 1 × 1 edge device can be operated for a lifetime of 2826 hours ($\sim$118 days) at $I_{comp} \sim 2\mu$A, whereas the operation would last only about 22 hours (less than a day) for $I_{comp} \sim 250\mu$A. Fig. 10 illustrates the convolution loss using RRAM encoded GoogleNet trained weights for the given wide current compliance resistance distribution scale applied to the 1×1 convolution. For illustrative purposes, we have considered two sets of images. The first image depicts a dog's picture and takes up to 80% of the pixels in the given 224×224 image size. The second image is that of ants, which occupies 30% of the pixels in the standard image size of 224×224 pixels. Here, we can deduce that the ant image is comparatively more convoluted and results in more visible/pattern loss, making it appear more blur than the dog image. Thus, the computational device failure depends on the pattern size in the given standard image pixel of 224×224 and RRAM false 0/1-bit position encoding probability. The RRAM device stack performance also depends on the fabrication conditions which usually spans a wide range of parameters and process conditions. Here, we omit the influence of the fabrication process parameter variables on the device variability for simplicity and stick to the given material stack's resistance distribution data. There is always a trade-off between the CNN network topology, operating power, and end application accuracy. Battery-powered IoT applications in the real-world demand low operation power for a long lifetime; hence compromising prediction accuracy by choosing smaller current compliance will significantly extend battery life. Recent studies show edge AI applications with fault tolerance are the next trend for designing low-powered IoT edge devices in monitoring and sensing in remote applications such as oil platforms, covered drain, remote surveillance systems, etc…[**39**]. An overall prediction error tolerance of 20 to 40% is acceptable in such applications, where sampling and regression trend analysis can determine the error deviation. We can still operate the RRAM in high current compliance with a wide memory window for mission-critical operations alone.

D. ERROR PROPAGATION ACROSS A DEEP CNN ARCHITECTURE

While the previous analysis purely focuses on the error induced by just one single FSS or FPS layer, we know well that the CNN used for most applications easily consists of 4-10 layers. A model to study the error propagation through multiple hidden layers of FSS or FPS will be helpful to understand the end application's overall prediction accuracy drop. Hence, we have considered two different types of multiple hidden layers with 10 internal layers, and each configured with the FPS or FSS extreme architectures, as shown in Fig. 11. The multiple hidden layers with FSS mode are denoted as multiple Fully Serial Sequence (mFSS-CNN), and for the FPS mode, we coin it as multiple Fully Parallel Sequence (mFPS-CNN). Here we use simple statistical formulae to estimate the propagation of layer-to-layer uncertainty. We may assume that the FSS and FPS error variance is fixed for a given current compliance as recorded in Figs. 7 and 8. Chaining the same structure (FSS or FPS)

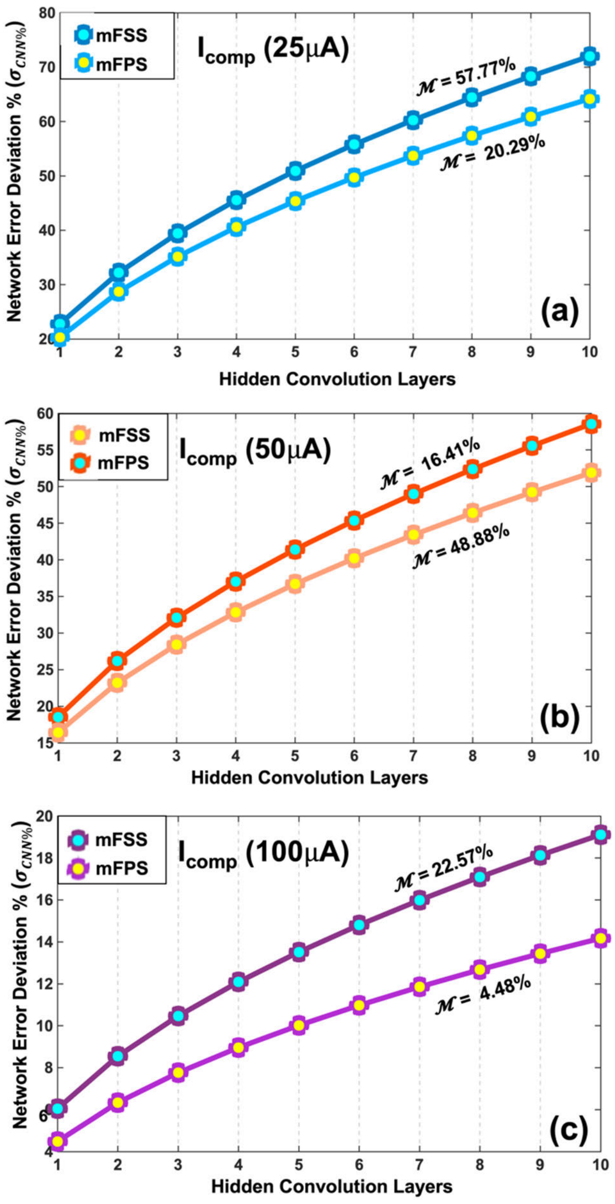


FIGURE 12. Trend in the error propagation across the mFSS and mFPS deep CNN networks, considering 1-10 hidden convolution layers. The mean error (*M*) of mFPS/mFSS for each I_{comp} is shown in the legend. The results for (a) 25μA, (b) 50μA and (c) 100μA compliance levels are shown here.

as multiple hidden layers, the error propagation would follow the model below.

$$\sigma_{\varepsilon} = \sqrt{\frac{\sum(\varepsilon_i - \mu)^2}{N}} \tag{4}$$

where:

$\sigma\varepsilon$ = Standard deviation of error for single I_{comp} simulation
N = Size of the simulation cycle = 5000
ε_i = Error value from each simulation

$\mu =$ The population mean error

$$\sigma^2_{CNN} = \sum\nolimits_{i=1}^{n} \sigma^2_{\varepsilon-Layer-i} \tag{5}$$

The standard deviation in error for the respective FPS and FSS architectures (σ_ε) is computed for all the I_{comp} encoded data set using the Eqn. (4). Here, the mean error obtained from 5000 repetitive stimulation cycles is used in the above equation. The propagation in error across the deep CNN network (σ_{CNN}) can then be estimated by Eqn. (5), where $n = 10$ (total number of layers) is the depth (number of hidden convolution layers) of the network and σ_{CNN} represents the overall network error deviation.

We have used the error variance of $I_{comp} = \{25, 50, 100\}$ μA to compute the layer-to-layer standard deviation error percentage ($\sigma_{err\%}$) using Eqns. (4) and (5) and the results shown in Figs. 12 (a) – (c), respectively. Considering the output of the 10th layer, the difference between the $\sigma_{err\%}$ of 25μA and 100μA is $\sim$4 times higher, and a similar trend follows for their respective hidden layers. Note that the mean of the error in the CNN is likely to be the same as the mean of any single layer of the network. The mean errors are indicated in the legend of the plots in Fig. 12 and denoted by "$\mathscr{M}$". Hence, our approach here for computing the layer-to-layer error variability enhancement can be used as a preliminary setup to quantify the error trend in a multiple hidden layer CNN. For simplicity, we exclude other device fabrication parameters from this discussion.

IV. CONCLUSION OF THE STUDY

In this study, we have highlighted the critical issues involved in the power consumption bottleneck brought about by the memory system in today's cloud servers. The alternative is to move towards in-memory computation for image processing IoT applications. We have compared various modern in-memory technologies and considered RRAM as the candidate of analysis, given its low power footprint advantage, silicon CMOS compatibility as well as ease of fabrication and its robustness. For a wide range of compliance levels ranging from soft (2-5μA) to hard breakdown (100-250μA), we have quantified the trade-off in the power-prediction accuracy for a CNN. The impact of the series-parallel architecture on the prediction error has also been considered and we have extended our analysis to present the worst case (mFSS) and best case (mFPS) scenarios of how error propagates through a deep CNN up to 10 hidden convolution layers. The look-up table-based framework proposed here is device technology agnostic and can be used for error quantification for any edge compute application for any device as long as its operating state variability can be characterized comprehensively, as was done by Fantini *et al.* in Ref. [**33**]. This is the first study that clearly quantifies the impact of a hardware realization of an RRAM-based CNN on a practical large scale open-source network, the popularly used GoogleNet.

Note that our study assumes that the training of the weights in the CNN still happens in the cloud and the edge computing here is purely for the inference side using hardware to replace the software trained weights to minimize or even eliminate latency issues due to server-node communication traffic. A truly edge application would require training (learning) and inference to also happen on the nodes itself and the training process itself will also be heavily affected by the inherent device switching variability. We are in the process of extending our framework to also account for variability and error induced in the training phase of a fully RRAM based network learning process using NAND-based computational logic and 2T-1R architectures to replace the NAND operation. The impact of forward learning and backpropagation on the training robustness of a fully RRAM based CNN will be the subject of our next study, building on our work here.

ACKNOWLEDGMENT

The first author would like to thank the Ministry of Education (MOE), Singapore for providing the research student scholarship (RSS) at SUTD for 2018-2022. The corresponding author would also like to acknowledge the financial and logistical support from the A*STAR Brain Efficient Nanomechanical Artificial Intelligence Computing (BRENAIC) Research Project No. A18A5b0056, which enabled the work to be accomplished.

REFERENCES

[1] A. R. Neto, T. P. Silva, T. Batista, F. C. Delicato, P. F. Pires, and F. Lopes, "Leveraging edge intelligence for video analytics in smart city applications," *Information*, vol. 12, no. 1, p. 14, Dec. 2020.

[2] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–23, Feb. 2019.

[3] A. Agrawal, A. Kosta, S. Kodge, D. E. Kim, and K. Roy, "CASH-RAM: Enabling in-memory computations for edge inference using charge accumulation and sharing in standard 8T-SRAM arrays," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 10, no. 3, pp. 295–305, Sep. 2020.

[4] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "Processing data where it makes sense: Enabling in-memory computation," *Microprocessors Microsyst.*, vol. 67, pp. 28–41, Jun. 2019.

[5] A. Biswas and A. P. Chandrakasan, "CONV-SRAM: An energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, Jan. 2018.

[6] A. Agrawal, A. Ankit, and K. Roy, "SPARE: Spiking neural network acceleration using ROM-embedded RAMs as in-memory-computation primitives," *IEEE Trans. Comput.*, vol. 68, no. 8, pp. 1190–1200, Aug. 2019.

[7] Q.-F. Ou, B.-S. Xiong, L. Yu, J. Wen, L. Wang, and Y. Tong, "In-memory logic operations and neuromorphic computing in non-volatile random access memory," *Materials*, vol. 13, no. 16, p. 3532, Aug. 2020.

[8] M. Goswami, J. Pal, M. R. Choudhury, P. P. Chougule, and B. Sen, "In memory computation using quantum-dot cellular automata," *IET Comput. Digit. Techn.*, vol. 14, no. 6, pp. 336–343, Oct. 2020.

[9] F. Zayer, B. Mohammad, H. Saleh, and G. Gianini, "RRAM crossbar-based in-memory computation of anisotropic filters for image preprocessingloa," *IEEE Access*, vol. 8, pp. 127569–127580, 2020.

[10] W. Zhang, B. Gao, J. Tang, P. Yao, S. Yu, M.-F. Chang, H.-J. Yoo, H. Qian, and H. Wu, "Neuro-inspired computing chips," *Nature Electron.*, vol. 3, no. 7, pp. 371–382, Jul. 2020.

[11] S. Munjal and N. Khare, "Advances in resistive switching based memory devices," *J. Phys. D, Appl. Phys.*, vol. 52, no. 43, Oct. 2019, Art. no. 433002.

[12] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal-oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012.

[13] X. Hong, D. J. Loy, P. A. Dananjaya, F. Tan, C. Ng, and W. Lew, "Oxide-based RRAM materials for neuromorphic computing," *J. Mater. Sci.*, vol. 53, no. 12, pp. 8720–8746, 2018.
[14] P. Pouyan, E. Amat, S. Hamdioui, and A. Rubio, "RRAM variability and its mitigation schemes," in *Proc. 26th Int. Workshop Power Timing Modeling, Optim. Simul. (PATMOS)*, Sep. 2016, pp. 141–146.
[15] E. Pérez, D. Maldonado, C. Acal, J. E. Ruiz-Castro, F. J. Alonso, A. M. Aguilera, F. Jiménez-Molinos, C. Wenger, and J. B. Roldán, "Analysis of the statistics of device-to-device and cycle-to-cycle variability in TiN/Ti/Al: HfO_2/TiN RRAMs," *Microelectron. Eng.*, vol. 214, pp. 104–109, Jun. 2019.
[16] A. Napolean, N. M. Sivamangai, J. Samuel, and V. John, "Overview of current compliance effect on reliability of nano scaled metal oxide resistive random access memory device," in *Proc. 4th Int. Conf. Devices, Circuits Syst. (ICDCS)*, Mar. 2018, pp. 290–296.
[17] S. Wiefels, M. Von Witzleben, M. Huttemann, U. Bottger, R. Waser, and S. Menzel, "Impact of the ohmic electrode on the endurance of oxide-based resistive switching memory," *IEEE Trans. Electron Devices*, vol. 68, no. 3, pp. 1024–1030, Mar. 2021.
[18] V. Borisov, J. Haug, and G. Kasneci, "Cancelout: A layer for feature selection in deep neural networks," in *Proc. Int. Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, Sep. 2019, pp. 72–83.
[19] Y. Liao, B. Gao, F. Xu, P. Yao, J. Chen, W. Zhang, J. Tang, H. Wu, and H. Qian, "A compact model of analog RRAM with device and array nonideal effects for neuromorphic systems," *IEEE Trans. Electron Devices*, vol. 67, no. 4, pp. 1593–1599, Apr. 2020.
[20] T. Tao, H. Ma, Q. Chen, Z.-M. Gu, H. Jin, M. Ahmed, S. Tan, A. Wang, E.-X. Liu, and E.-P. Li, "Circuit modeling for RRAM-based neuromorphic chip crossbar array with and without write-verify scheme," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 5, pp. 1906–1916, May 2021.
[21] G. Boquet, E. Macias, A. Morell, J. Serrano, E. Miranda, and J. L. Vicario, "Offline training for memristor-based neural networks," in *Proc. 28th Eur. Signal Process. Conf. (EUSIPCO)*, Jan. 2021, pp. 1547–1551.
[22] A. J. Ford and R. Jha, "Memristive device variability performance impact on neuromorphic machine learning hardware," in *Proc. 11th Int. Green Sustain. Comput. Workshops (IGSC)*, Oct. 2020, pp. 1–7.
[23] P. Pal, S. Thunder, M. J. Tsai, P. T. Huang, and Y. H. Wang, "Benchmarking the performance of heterogeneous stacked RRAM with CFETSRAM and MRAM for deep neural network application amidst variation and noise," in *Proc. Int. Symp. VLSI Technol., Syst. Appl. (VLSI-TSA)* Apr. 2021, pp. 1–2.
[24] M. Fritscher, J. Knödtel, D. Reiser, M. Mallah, S. Pechmann, D. Fey, and M. Reichenbach, "Simulating large neural networks embedding MLC RRAM as weight storage considering device variations," in *Proc. IEEE 12th Latin America Symp. Circuits Syst. (LASCAS)*, Feb. 2021, pp. 1–4.
[25] Y. Cai, Z. Wang, Z. Yu, Y. Ling, Q. Chen, Y. Yang, S. Bao, L. Wu, L. Bao, R. Wang, and R. Huang, "Technology-array-algorithm co-optimization of RRAM for storage and neuromorphic computing: Device non-idealities and thermal cross-talk," in *IEDM Tech. Dig.*, Dec. 2020, pp. 13.4.1–13.4.4.
[26] Y. Du, L. Jing, H. Fang, H. Chen, Y. Cai, R. Wang, J. Zhang, and Z. Ji, "Exploring the impact of random telegraph noise-induced accuracy loss on resistive RAM-based deep neural network," *IEEE Trans. Electron Devices*, vol. 67, no. 8, pp. 3335–3340, Aug. 2020.
[27] H.-H. Le, W.-C. Hong, J.-W. Du, T.-H. Lin, Y.-X. Hong, I.-H. Chen, W.-J. Lee, N.-Y. Chen, and D. D. Lu, "Ultralow power neuromorphic accelerator for deep learning using Ni/HfO_2/TiN resistive random access memory," in *Proc. 4th IEEE Electron Devices Technol. Manuf. Conf. (EDTM)*, Apr. 2020, pp. 1–4.
[28] M. Kwak, W. Choi, S. Heo, C. Lee, R. Nikam, S. Kim, and H. Hwang, "Excellent pattern recognition accuracy of neural networks using hybrid synapses and complementary training," *IEEE Electron Device Lett.*, vol. 42, no. 4, pp. 609–612, Apr. 2021.
[29] P. Freitas, Z. Chai, W. Zhang, J. F. Zhang, and J. Marsland, "Impact of RTN and variability on RRAM-based neural network," in *Proc. IEEE 15th Int. Conf. Solid-State Integr. Circuit Technol. (ICSICT)* Nov. 2020, pp. 1–4.
[30] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN+NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *IEDM Tech. Dig.*, Dec. 2019, pp. 32.5.1–32.5.4.
[31] J. Doevenspeck, R. Degraeve, A. Fantini, S. Cosemans, A. Mallik, P. Debacker, D. Verkest, R. Lauwereins, and W. Dehaene, "OxRRAM-based analog in-memory computing for deep neural network inference: A conductance variability study," *IEEE Trans. Electron Devices*, vol. 68, no. 5, pp. 2301–2305, May 2021.
[32] M. Pedro, J. Martin-Martinez, R. Rodriguez, M. B. Gonzalez, F. Campabadal, and M. Nafria, "A flexible characterization methodology of RRAM: Application to the modeling of the conductivity changes as synaptic weight updates," *Solid-State Electron.*, vol. 159, pp. 57–62, Sep. 2019.
[33] C. C. Aggarwal, *Neural Networks and Deep Learning*. Cham, Switzerland: Springer, 2018, pp. 3–978.
[34] A. Bhardwaj, W. Di, and J. Wei, *Deep Learning Essentials: Your Hands-on Guide to the Fundamentals of Deep Learning and Neural Network Modeling*. Birmingham, U.K.: Packt Publishing Ltd, 2018.
[35] B. Moons, D. Bankman, and M. Verhelst, *Embedded Deep Learning*. Cham, Switzerland: Springer, 2019.
[36] A. Fantini, L. Goux, R. Degraeve, D. J. Wouters, N. Raghavan, G. Kar, A. Belmonte, Y.-Y. Chen, B. Govoreanu, and M. Jurczak, "Intrinsic switching variability in HfO_2 RRAM," in *Proc. 5th IEEE Int. Memory Workshop*, May 2013, pp. 30–33.
[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[38] N. Prabhu, D. L. Jia Jun, P. Dananjaya, W. Lew, E. Toh, and N. Raghavan, "Exploring the impact of variability in resistance distributions of RRAM on the prediction accuracy of deep learning neural networks," *Electronics*, vol. 9, no. 3, p. 414, Feb. 2020.
[39] W. Lin, A. Adetomi, and T. Arslan, "Low-power ultra-small edge AI accelerators for image recognition with convolution neural networks: Analysis and future directions," *Electronics*, vol. 10, no. 17, p. 2048, Aug. 2021.

NAGARAJ LAKSHMANA PRABHU is currently pursuing the Ph.D. degree. His Ph.D. examining the impact of RRAM device level variability as an in-memory computational element for deep learning neural network (DNN) applications. He also serves as one of the Directors of ALAI Labs, focusing on design and development of vision-based IoT products and solutions (on the cloud and on the edge) for general use daily applications. He has over 18 years of experience in industrial product design and development, specializing in machine vision and cloud computation. He has five scientific publications to his credit relating to construction methodology for look-up table modeled RRAM-based synaptic weight simulation framework and quantification of the DNN prediction error rate for given RRAM device level variability.

NAGARAJAN RAGHAVAN (Member, IEEE) received the Ph.D. degree in microelectronics from the Division of Microelectronics, Nanyang Technological University (NTU), Singapore, in 2012. He is currently an Assistant Professor at the Engineering Product Development (EPD) Pillar, Singapore University of Technology and Design (SUTD). Prior to this, he was a Postdoctoral Fellow at the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, and IMEC, Belgium, in joint association with the Katholieke Universiteit Leuven (KUL). To date, he has authored/coauthored more than 220 international peer-reviewed publications and five invited book chapters. His research interests include prognostics and health management for electromechanical failures, design for reliability, lifecycle management of nanoelectronic devices, physics of failure, optimization of polymer nanocomposites, and uncertainty quantification for additive manufacturing. He was an Invited Member of the IEEE GOLD Committee, from 2012 to 2014. He was a recipient of the IEEE EDS Early Career Award, in 2016, and the IEEE Reliability Society Graduate Scholarship Award, in 2008. He was also a recipient of the IEEE EDS Ph.D. Student Fellowship, in 2011. He served as the General Chair for IEEE IPFA 2021 at Singapore. He has consistently served on the review committee for various IEEE journals and conferences, including IRPS, IIRW, IPFA, and ESREF. He is an Associate Editor of IEEE ACCESS journal.