MoRS: An Approximate Fault Modelling Framework for Reduced-Voltage SRAMs

İsmail Emir Yüksel[‡], Behzad Salami[†], Oğuz Ergin[‡], Osman S. Ünsal[†], Adrián Cristal Kestelman[†§]
[‡] TOBB University of Economics and Technology (TOBB ETÜ) [†] Barcelona Supercomputing Center (BSC) [§] Universitat Politècnica de Catalunya (UPC)

***Abstract*—On-chip memory (usually based on Static RAMs-SRAMs) are crucial components for various computing devices including heterogeneous devices, *e.g,* GPUs, FPGAs, ASICs to achieve high performance. Modern workloads such as Deep Neural Networks (DNNs) running on these heterogeneous fabrics are highly dependent on the on-chip memory architecture for efficient acceleration. Hence, improving the energy-efficiency of such memories directly leads to an efficient system. One of the common methods to save energy is undervolting *i.e.,* supply voltage underscaling below the nominal level. Such systems can be safely undervolted without incurring faults down to a certain voltage limit. This safe range is also called voltage guardband. However, reducing voltage below the guardband level without decreasing frequency causes timing-based faults.**

In this paper, we propose MoRS, a framework that generates the first approximate undervolting fault model using real faults extracted from experimental undervolting studies on SRAMs to build the model. We inject the faults generated by MoRS into the on-chip memory of the DNN accelerator to evaluate the resilience of the system under the test. MoRS has the advantage of simplicity without any need for high-time overhead experiments while being accurate enough in comparison to a fully randomly-generated fault injection approach. We evaluate our experiment in popular DNN workloads by mapping weights to SRAMs and measure the accuracy difference between the output of the MoRS and the real data. Our results show that the maximum difference between real fault data and the output fault model of MoRS is 6.21%, whereas the maximum difference between real data and random fault injection model is 23.2%. In terms of average proximity to the real data, the output of MoRS outperforms the random fault injection approach by 3.21x.

***Index Terms*—Modeling, Fault-injection, Neural Networks, Undervolting, SRAM.**

I. Introduction

SRAMS are traditionally the building block of different components in different computing systems such as branch predictor (in CPUs), register files (in GPUs), on-chip buffer memory (in hardware accelerators like FPGAs and ASICs), thanks to the low-latency access time of such memories. However, the power consumption of SRAMs significantly contributes to the total system power. For instance, prior works on GPUs [1] show that register file in GPUs consumes 15-20% of the total power. Another work on modern out-of-order cores [2] estimates that the SRAM-dominated front-end of the microprocessor consumes up to 33% of the total power of the CPU. Other work on DNN accelerators on FPGAs [3] shows that on-chip SRAMs consume 27% of the total power.

Since the total power consumption of any underlying hardware is directly related to its supply voltage, voltage underscaling is an effective solution to save power [4], [5]. This technique is widely used in various devices such as CPUs [6]–[10], GPUs [11] FPGAs [3], [12]–[15], and ASICs [16]–[18] as well as DRAMs [19]–[21], HBMs [22], SRAMs [23], [24], and Flash Disks [25]–[28]. However, while performing voltage underscaling, reliability issues can arise due to the increased circuit delay at reduced voltage levels. In most commercial devices, there is a timing-fault free guardband between nominal voltage and minimum safe voltage, *i.e.,* V_{min}. Below this voltage guardband, faults occur as a consequence of circuit delays.

Although further aggressive voltage underscaling below V_{min} can achieve more reduction in power consumption, it compromises system reliability through undervolting faults. This particular region between V_{min} and the lowest operational voltage level, *i.e.,* V_{crash}, still operational but with faults is called the critical area [3]. Several prior fault mitigation techniques were proposed [3], [16], [17], [29], [30]. However, these techniques need either high-effort engineering [3], [30] or totally random fault injection campaigns [16], [17]. These approaches are either impractical or not accurate. Our solution offers the advantages of both random fault injection and empirical experiments.

In this study, we propose MoRS, a framework that generates the first approximate voltage underscaled model for SRAMs. MoRS consists of three steps: 1) Experiment, 2) Behavior Extraction, and 3) Model Generation. In the Experiment step, MoRS uses publicly available undervolted fault map data [31] of SRAMs based on our prior work [3]. In the Behavior Extraction step, we extract the characteristic fault behavior features of undervolted SRAM blocks. We establish and confirm that undervolting based faults do not occur randomly. These faults are correlated with each other in space. We examine the faults of row-based and column-based approaches and see the distance between consecutively faulty bitcells, the number of each bit-fault in both rows and columns, and the total number of faulty rows and columns per SRAM block are not uniformly distributed. These fine-grained features show characteristic behaviors. In this step, we extract characteristic features and categorize them into two profiles: coarse-grained and fine-grained profiling. The reason behind these categorizing is that random fault injection studies use only coarse-grained features, the number of bit faults, and the number of faulty SRAM blocks. The output of MoRS that we term Mixed Model uses both fine-grained and coarse-grained features and applies probabilistic custom modeling algorithm to achieve an approximate model as the last step.

In recent years, the power consumption of SRAMs on DNNs is increasing drastically [3], [16], [17]. We evaluate MoRS for DNN accelerators where on-chip SRAMs play an important role. MoRS generates fault models that are not limited to a certain domain and can be potentially used in many domains such as branch prediction units, register files, caches, and any other memories based on SRAM, unlike the other prior studies. To evaluate MoRS, we generate a baseline model by applying a uniform random distribution function to coarse-grained features. This baseline, which we term as Random Model, is a standard fault injection scheme and used in prior studies [16], [17], [32], [33]. Besides, we process empirical data to see the difference in accuracy between each artificial model (Random Model and Mixed Model) and real data.

In our evaluation methodology, we map weights to SRAM blocks. When we map weights to SRAM blocks, there can be different mapping options such as MSB mapping, LSB mapping, the first half of bits MSB the other half of bits LSB mapping, and the first half of bits LSB the other half of bits MSB mapping. Another method to save energy on DNNs is quantization. Our quantization method is reducing the precision of weights from 32-bit to 16-bit, 8-bit, 4-bit, and 1-bit, respectively. While performing undervolting, some unwanted bits can be flipped, and then the corresponding value becomes infinity or NaN. To avoid these values masking techniques are used [34]. We mask infinity or NaN value to 1 or 0. We examine the behavior of different weight mappings, quantizations, and masking options. We evaluate our experiments on trained LeNeT-5 [35] and cuda-convnet [36] DNNs in the classification stage. We examine the classification accuracy for each voltage level. Our experiments show that generated artificial model has similar behavior with the real data on different DNN benchmarks with an accuracy of 96.4%. We also see that Random Model is not close enough with real data with the difference in accuracy up to 23%. We find that on average our Mixed Model has 3.6% difference with real data and on average 3x up to 7x closer than the baseline random model.

Our contributions are as follows:

- We propose a framework, MoRS that generates an artificial model that can realistically emulate real undervolting fault data with a difference in the accuracy of 3.6% on average. To the best of our knowledge, this study provides the first reasonably accurate model for SRAM blocks under low-voltage conditions.
- We evaluate our models and real data on state-of-the-art DNNs to see how different weight mappings, quantization, and value maskings affect the accuracy of inference. We find the similar observation with prior works [37], [38] that if we continue to reduce the precision of weights, DNNs become more resilient to undervolting. At the lowest reduced voltage level 8-bit LeNeT accuracy is 14% while 4-bit LeNeT accuracy is 60%. Even in this unlikely situation, our Mixed Model shows similar behavior to real data.

The remainder of this paper is structured as follows. In Section II, we introduce the most important concepts used in this paper. In Section III, we propose our approximate fault modeling framework, MoRS. Section IV describes the methodology that we perform our experiments. Section V detail the experimental results. Related works are introduced in Section VI. Finally, Section VII concludes this paper.

II. Background

A. Undervolting SRAM based on-chip Memories

CMOS is the dominant circuit technology for current computing devices. The power consumption of CMOS-based SRAMs is the sum of two parts: the first is dynamic power, also called active power, and the other one is leakage power, also called static power. Earlier studies [39]–[41] show that the power consumption of SRAMs is dominated by dynamic power.

Dynamic power *i.e.,* P_{dyn}, is dissipated when there is switching activity at some nodes in transistors. Leakage power *i.e.,* P_{leak}, is dissipated by the leakage currents flowing even when the device is not active. Mathematically, these power equations are given by,

$$P_{dyn} \propto f \times V_{dd}^2 \tag{1}$$

$$P_{leak} = k_{design} \times n \times I_{leak} \times V_{dd} \tag{2}$$

Here V_{dd} shows the supply voltage, f shows the operating frequency in Equation 1. Further, n denotes the number of transistors, k_{design} is the design-dependent parameter, and I_{leak} shows the leakage current a technology-dependent parameter in Equation 2. From Equation 1, dynamic power consumption can be reduced by adjusting the supply voltage and operating frequency. Likewise, from Equation 2, leakage power consumption can be reduced by underscaling the supply voltage and reducing the total number of transistors. Total power consumption can be reduced by underscaling the supply voltage.

Voltage underscaling is a widely used technique for energy-efficient applications. From Equation 1, dynamic power consumption is reduced quadratically by underscaling the supply voltage. This technique can achieve nominal voltage level *i.e.,* V_{nom} performance due to the operating frequency not being changed. However, further aggressive undervolting below the minimum safe voltage *i.e.,* V_{min} may cause reliability issues as the result of timing faults. Between V_{nom} and V_{min} is called voltage guardband. This guardband is to provide an assurance of correct functionality even in the worst environmental case. Prior works achieve by applying undervolting that power consumption reduces; 39% on FPGA on-chip memories [3], 20% in GPUs [42], and 16% in DRAMs [19] without any timing-related errors. Also, recent studies show that undervolting internal components of FPGAs [12] lead to around 3x power-efficiency and underscaling supply voltage of HBMs [22] achieve a total of 2.3x power savings. In Minerva [16], lowering SRAM voltages achieves a total of 2.7x power savings.

B. Fault Injection

The fault injection mechanism is a way to examine the behavior of systems under different circumstances. Prior works

on fault injection study for reduced-voltage SRAMs are applied on branch prediction units [32], [43], caches [44], [45], and FPGA on-chip memories [33]. There are several approaches for the fault injection with having a trade-off on the engineering effort and accuracy. i) The first one is applying random faults to random locations without any information used from empirical experiments. ii) Another one is directly using the empirical data as a fault map. iii) The last one is the approximate modeling that is based on real data from empirical experiments and close enough to empirical data. Table I gives a summary of comparison of these three fault injection techniques in terms of effort and how close they are to real data. The accuracy percentage is from the results of this study.

TABLE I: Comparison of fault injection techniques in terms of engineering effort and accuracy of representing real data

Fault Injection Method	Engineering Effort	Accuracy (Min)
Random Fault Injection	Low	77%
MoRS	Low	94%
Empirical Data	High	100%

C. Deep Neural Networks

Deep Neural Networks (DNNs) are widely used as an effective solution for object recognition, classification and segmentation, and many other areas. MNIST [46] and CIFAR-10 [47] datasets are widely used by the ML community to showcase the latest technology advancements in DNNs. DNNs perform in two phases: training and prediction (inference). The first process is training that weights of DNNs are fitted to data. The training phase is mainly performed in high-performance computing platforms like CPUs or GPUs. The second phase is the inference phase, DNN models, trained in the first phase, are used to extract unknown output from input data. Usually, the training phase performs once whereas the prediction phase performs repeatedly. The inference phase consists of mainly three elements: input, weights, and output.

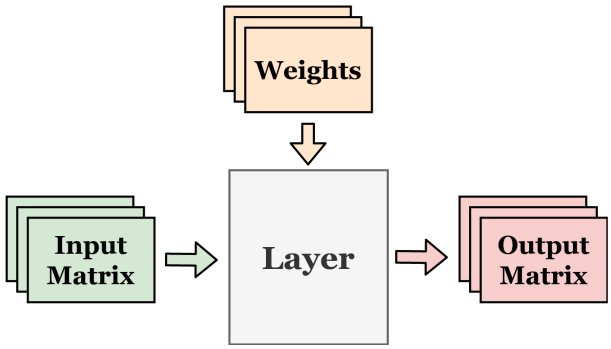


Fig. 1: A high level abstraction of one layer of DNNs.

Figure 1 shows a high-level abstraction of one inference layer. Each inference layer of the network consists of weights, an input matrix, and an output matrix. Weights of these layers are stored in on-chip memories. Due to their large size and computational intensity, DNNs consume power; while they are fault-tolerant to some extend. Some studies improve the power-efficiency and reduce the total power consumption of DNNs by applying voltage underscaling [3], [12], architectural techniques [48]–[55], and hardware-level techniques [56]–[58].

Quantization is an architectural optimization technique that reduces the precision of data types. This technique can improve performance and energy-efficiency. Recent studies [16], [59]–[62] show that quantization to some extent does not significantly affect the accuracy of DNN models. In our study, we reduce weights precisions to 16-bit half-precision floating point, 8-bit($Q_{4.4}$), 4-bit($Q_{2.2}$) in fixed-point format, and 1-bit binary values.

III. MoRS Framework

We propose MoRS, a framework to generate approximate artificial reduced-voltage SRAM fault models. This mechanism is the first framework based on real fault maps. MoRS stands between fully hardware and fully software fault injection techniques. This framework generates an approximate model that is close enough to real data compared to the fully software fault injection mechanism. Also, MoRS does not require high-effort engineering in comparison to a fully hardware approach that uses real data from empirical experiments.

As shown in Figure 2 MoRS consists of three steps: ❶ Experiment, ❷ Behavior Extraction, and ❸ Model Generation. We first explain the first step existing experiment that provides real fault maps from real SRAM blocks in Section 3.1. In Section 3.2 we extract the behavior of fault maps as fine-grained and coarse-grained profiles by using the output of the first step. In fine-grained profiling extract the row and column behaviors of SRAM blocks in terms of physical distance, the number of each bit-faults in rows and columns, and the number of faulty rows and columns per block. Coarse-grained profiling is a shallow approach generally used in prior fault injection studies. In coarse-grained profiling, we extract only the total number of bit-faults and faulty SRAM blocks. In Section 3.3 we generate an artificial model with outputs of the second steps.

At the endpoint, we provide Mixed Model which is an approximate model to real data. It is generated by applying probabilistic modeling on both fine-grained and coarse-grained fault profiles. Figure 2 provides an overview of the three steps of the MoRS.

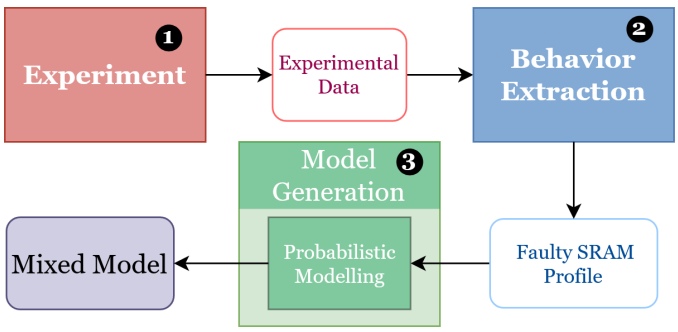


Fig. 2: Overview of the MoRS

Board Name	VC707	ZC702	KC705
Technology Node	28nm	28nm	28nm
Nominal Voltage	1V	1V	1V
Operating Temperature	$50°C$	$50°C$	$50°C$
Minimum Voltage Level	0.54V	0.53V	0.54V
Number of SRAM Block	2060	280	890

TABLE II: The summary of deployed FPGA boards

A. Experiment

Voltage underscaling, *i.e.,* undervolting is a widely used technique to save energy. We study undervolting for modern SRAM blocks. Each SRAM block is a matrix of bitcells formed of rows and columns. Also, each SRAM block is 16 Kbits with 1024 rows and 16 columns. We perform this empirical study on SRAM blocks available in the off-the-shelf Xilinx FPGAs, i.e., VC707, ZC702, and two identical samples of KC705 (referred as KC705-A and KC705-B). Table II shows the detail of deployed FPGAs. This experiment consists of two parts. The first part is the monitor SRAM blocks to see undervolting faults. The second part is adjusting the supply voltage of SRAM blocks by using a power management unit of FPGAs. FPGAs have a power management bus (PMBUS) a voltage regulator that monitors and adjusts voltage rails on FPGAs. This bus uses PMBUS standard and has an I2C protocol to execute commands. The supply voltage of SRAM blocks of FPGAs is named V_{CCBRAM}. By reducing V_{CCBRAM} through PMBUS, only the power consumption of SRAM blocks reduces. Since this situation does not affect any other logic parts (DSPs, LUTs, etc..) the effect of undervolting on SRAM blocks is seen clearly. The setup of this methodology is shown in Figure 3. The method

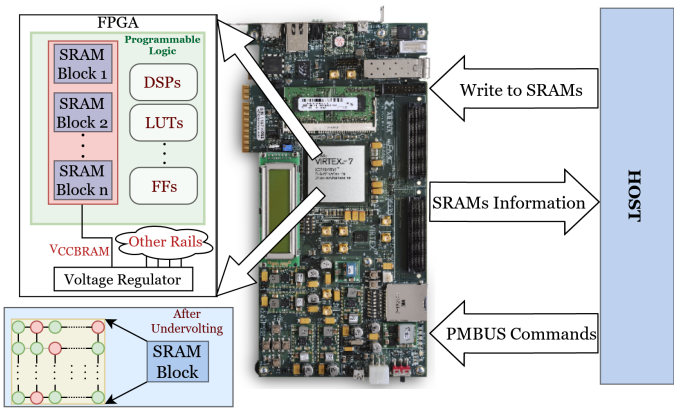


Fig. 3: The methodology of empirical experiment

of this experiment follows an algorithm that as a first step writes data to SRAMs after in the second step analyzes the faults in terms of rate and location, in the third step reduces the supply voltage by $10mV$, finally repeat these steps until FPGA crashes.

When undervolting is applied below the voltage guardband *i.e.,* V_{min}, the fault rate exponentially increases. Voltage can be reduced until the voltage that FPGA stops operating *i.e.,* V_{crash}. Between V_{min} and V_{crash} faults occur due to timing failures while the power consumption significantly reduces. SRAM-based real fault maps significantly vary even for different FPGAs of the same platform as the result of the process variation. Besides, the study shows that the pattern of faults is mostly permanent. Also, we observe that the fault rate and location for different run are mostly the same. Most importantly, undervolting faults are not uniformly distributed over different SRAMs.

We use the publicly available data [31] of the prior work [3] as the output of the first stage (Experimental Data) to generate artificial fault maps. We use 2000 SRAM blocks for extracting behavior and 950 SRAM blocks for testing our framework, 2060 SRAM blocks from VC707 and 890 SRAM blocks from KC705. We make the methodology of MoRS reliable by using different data set to create and also to test our artificial model.

At V_{crash} the fault rate, up to 0.06% and 0.005% per 1 Mbits for VC707 and KC705-B, respectively. It should be noted that faults appear in between V_{min} = 0.6V, V_{crash} = 0.54V for VC707 and V_{min} = 0.59V, V_{crash} = 0.53V for KC705-B. As the prior work [3] mentioned VC707 has the most bit-faults among the other three boards. At the lowest voltage level, VC707 has 10.2% faulty SRAMs with 23706 bit-faults. We refer [3] to more detailed information about this prior work.

B. Behavior Extraction

As we mentioned in Section 3.1 voltage underscaling faults have patterns and these fault patterns *i.e.,* fault maps, are mostly the same. There are features that affect the probability of bit-faults. In this step, we profile the behavior of undervolting-related bit-faults to extract such important features. We perform the profiling in two steps: coarse-grained profiling and fine-grained profiling. A summary of all features, profiling types, and which model they are used in can be found in Table III.

Coarse-grained profiling consists of two features. The first one is the percentage of bit-faults in all SRAMs bitcell *i.e.,* P_F. The second one is the percentage of faulty SRAMs in all SRAM blocks *i.e.,* P_S. For VC707, at V_{crash}, total bit-faults are 0.07% and the percentage of faulty SRAMs is 10.2%.

Fine-grained profiling comprises two parts: row-based, column-based. Both row-based and column-based have three features. The first one is the percentage of faulty rows *i.e.,* $P_{SR_{0..1024}}$ and faulty columns $P_{SC_{0..16}}$ in faulty SRAM blocks. The second one is the percentage of each bit-faults in rows *i.e.,* $P_{FR_{0..16}}$, or columns *i.e.,* $P_{FC_{0..1024}}$. The last one is the percentage of physical distance between consecutively faulty bitcells *i.e.,* bitcell-distances, in the same row *i.e.,* $P_{FDR_{1..15}}$, or column *i.e.,* $P_{FDC_{1..1023}}$. We discover that these two features for row and column are not randomly or uniformly distributed.

At V_{crash},

- The percentage of each bit-faults
  - In terms of row concentrates at 2-bit faults (P_{FR_2}) and no faults (P_{FR_0}).
  - In terms of column it concentrates no faults to 10-bit faults ($P_{FC_{0..10}}$).

TABLE III: Features and which model they are used in

Profiling Type	Features		Models	
	Name	Percentage	Mixed-Model	Random-Model
Coarse-grained	Total number of bit-faults	P_F	✓	✓
	Total number of faulty SRAM blocks	P_S	✓	✓
Fine-grained	Total number of faulty rows per SRAM block	$P_{SR_{0..1024}}$	✓	✗
	Total number of faulty columns per SRAM block	$P_{SC_{0..16}}$	✓	✗
	Number of each bit-fault in rows	$P_{FR_{0..15}}$	✓	✗
	Number of each bit-fault in columns	$P_{FC_{0..1023}}$	✓	✗
	Distance between consecutively faulty bitcells in rows	$P_{FDR_{1..15}}$	✓	✗
	Distance between consecutively faulty bitcells in columns	$P_{FDC_{1..1023}}$	✓	✗

- The percentage of bitcell-distances between consecutively faulty bitcells
  - In terms of row concentrates at 8-bit distance (P_{FDR_8}).
  - For rows that there is no such bitcell-distance is more than 8-bitcells ($P_{FDR_{9..15}}$).
  - For the columns, it is concentrated in even numbers in decreasing order$P_{FDC_{0..2..1024}}$, 2-bitcells distance (P_{FDC_2}) has the highest percentage while 1022-bitcells distance ($P_{FDC_{1022}}$) has the lowest in even numbers.
  - Also, for columns, the bitcell-distances in odd numbers ($P_{FDC_{1..3..1023}}$) are stuck between 0 and 4.

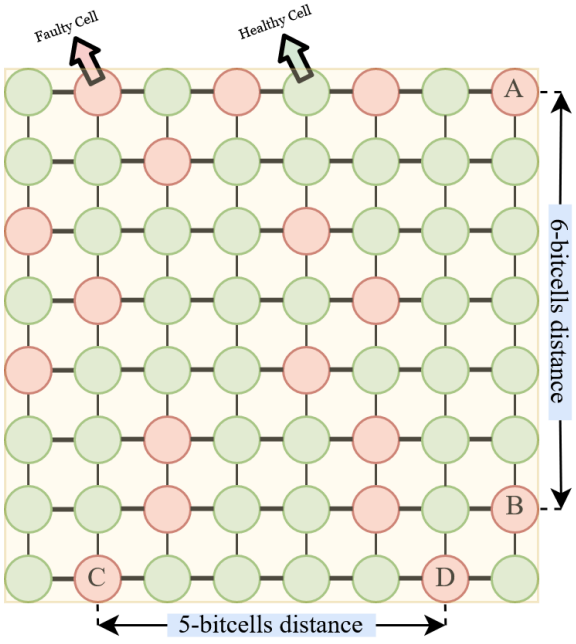


Fig. 4: Physical features on a faulty SRAM block as an example

As an illustration, we show the fault behavior of an $8 * 8$ SRAM block in Figure 4. In this example, the column-based distance between cell A and cell B is 6-bitcells. For cell C and cell D, the row-based distance is 5-bitcells. When we examine this example in Figure 4 in terms of bit-fault for both row-based and column-based, a column that contains cell C has 2-bit faults. Also, cell A's row has 4-bit faults. One of the coarse-grained profile features is the percentage of total bit-faults (P_F). It can be calculated by dividing the total number of faulty cells by all cells. Hence, for Figure 4's SRAM block P_F is 28.125%.

As we extract the behavior of this example in Figure 4, we performed this process for experimental data. When the profiling step is done, we start generating models by using these coarse-grained and fine-grained profiles with probabilistic modeling and uniform random distribution.

Algorithm 1 Generate Mixed Model

Require: $n \leftarrow \#ofSRAMblocks$
  $bitfaults \leftarrow n \times 1024 \times 16 \times P_F$
  $faultyS \leftarrow n \times P_S$
  while $faultyS > 0$ *and* $bitfaults > 0$ **do**
    $block \leftarrow random$
    $faultyBlock \leftarrow SRAMblocks[block]$
    $faultyrows \leftarrow selectfrom(F_{SR_{0..16}})$
    while $faultyrows > 0$ **do**
      $row \leftarrow random$
      $column \leftarrow random$
      $bitfaultsinrow \leftarrow selectfrom(F_{FR_{0..16}})$
      while $bitfaultsinrow > 0$ **do**
        $distance \leftarrow selectfrom(F_{FDR_{1..15}})$
        $column \leftarrow column + distance$
        $faultyBlock[row][column] \leftarrow fault$
        $bitfaultsinrow \leftarrow bitfaultsinrow - 1$
      end while
      $faultyrows \leftarrow faultyrows - 1$
    end while
    $ArtificialCF \leftarrow ColumnFeatures(faultyBlock)$
    $RealCF \leftarrow ColumnFeatures(RealData)$
    if $Similarity(ArtificialCF, RealCF) > 80\%$ **then**
      $SRAMblocks[block] \leftarrow faultyBlock$
      $faultyS \leftarrow faultyS - 1$
      $bitfaults \leftarrow bitfaults - \#offaults(faultyBlock)$
    end if
  end while

C. Model Generation

Probabilistic modeling [34] and uniform random distribution [17], [34], [43], [63] are widely used in many modeling studies to generate fault maps and to inject faults. For an approximate

model *i.e.,* Mixed Model, we use both fine-grained and coarse-grained features with custom probabilistic modeling function. In addition to these, we need the number of SRAM blocks that will generate.

To generate Mixed Model we follow the method shown in Algorithm 1. The input of Algorithm 1 is the number of SRAM blocks and the output is faulty SRAM data, also called fault map. First, in Algorithm 1, faulty SRAM blocks are determined. After that, by using P_F value, the number of faulty cells are calculated. First, we randomly select faulty SRAMs in all SRAM blocks. Then, we inject faults corresponding cells in faulty SRAMs. This injection algorithm uses fine-grained features. This process is performed in two stages: row-based fault injection and column-based control mechanism. The row-based fault injection uses row-based features. We determine how many rows will be faulty by using the probability of the number of faulty rows per SRAM block *i.e.,* $F_{SR_{0..1024}}$ derivating from $P_{SR_{0..1024}}$. Then according to the probability of the number of each bit-faults in rows *i.e.,* $F_{FR_{0..16}}$ derivating from $P_{FR_{0..16}}$ we inject bit-faults to the corresponding row. To inject more than one fault in a row we use the probability of physical distance between consecutively faulty bitcells in rows $F_{FDR_{1..15}}$ derivating from $P_{FDR_{1..15}}$. In the second step of this algorithm, first, we extract column-based features of each artificial fault map.

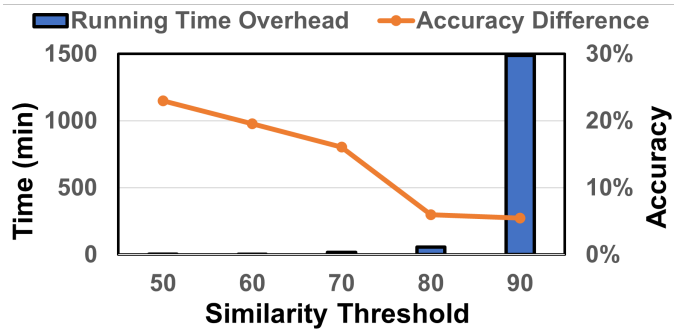


Fig. 5: The running time overhead and accuracy difference between real data and the generated artificial model for different similarity threshold levels

After extraction, we compare these features with fine-grained column-based features extracted from the second step. If an artificial faulty SRAM has a lower than 80% of similarity with experimental real data, we perform these steps again until the similarity is 80% or higher. We select the 80% similarity threshold as a good trade-off between the run-time (to generate the fault models) and accuracy (of the generated fault models). As shown in Figure 5, if we increase the threshold level, the run time of MoRS increases drastically. However, increasing the threshold is not achieve significant accuracy compared to the optimal threshold. Below this threshold, the approximate model converges to the Random Model that do not have an acceptable accuracy.

IV. Experimental Methodology

MoRS is a general framework that generates approximate fault maps for undervolted SRAM blocks. In this study, we test MoRS on state-of-the-art Deep Neural Networks. Our experiments are based on injecting faults into weights of trained DNNs. To evaluate how precise MoRS is we use Caffe [64]. Also, we perform different quantizations (precisions), bit-mappings, and value masking to diversify our experiments. The summary of these options is in Table IV.

TABLE IV: Different options for evaluation

Options	Name
Precision	32-bit single-precision floating point
	16-bit half-precision floating point
	8-bit fixed point (Q4.4)
	4-bit fixed point (Q2.2)
	1-bit (Binary)
Bit-Mapping	MSB
	LSB
	First half MSB and other half LSB
	First half LSB and other half MSB
Value Masking	Infinity or NaN to 1
	Infinity or NaN to 0

The experiment is performed for each voltage level between V_{min} and V_{crash}, precision, mapping, and masking option. To compare artificial models with real data we also process this methodology for real data. Artificial Models are Random Model and Mixed Model. Random Model is a naive random baseline used in prior works to inject faults. Mixed Model is an approximate model, the output of the MoRS. Real Data is the experimental data [31] extracted from the prior empirical study [3]. To evaluate real data, we select the required amount of SRAM blocks randomly from real data. Instead of Artificial Models, we process real data to evaluate in Figure 6.

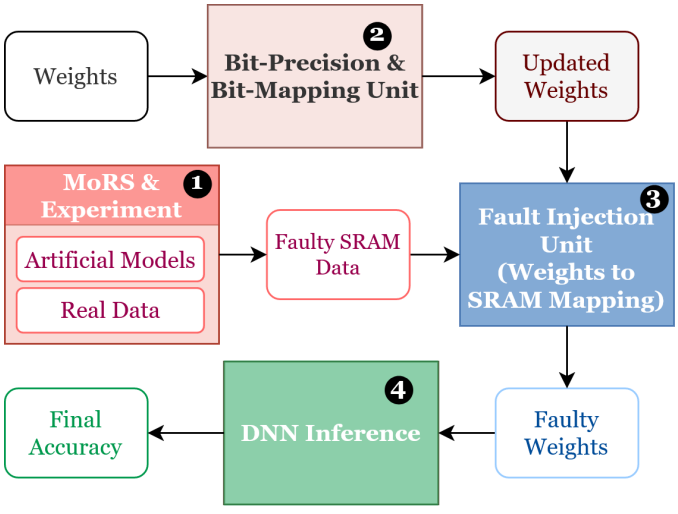


Fig. 6: Overall Methodology

The methodology consists of four parts as shown in Figure 6: ❶ MoRS & Experiment, ❷ Bit-Precision & Bit-Mapping Unit, 3 ❸ Fault Injection Unit (Weights to SRAM Mapping) and ❹ DNN Inference. In the first step, we acquire artificial models from MoRS that are explained in Section 3. In the second step, the Fault Injection Unit changes healthy weights

into updated weights by performing mapping and precision options. In the third step, with the outputs of the first and second steps, we generate faulty weights. In the last step, we obtain the final accuracy percentage.

1. MoRS & Experiment. In this step, we choose which SRAM data is sent to Fault Injection Unit. To evaluate MoRS, we generate a baseline model, *i.e.,* Random Model. Random Model is generated by uniform random distribution with coarse-grained features. We process every step in MoRS described in Section II. However, we apply a directly random distribution function to only coarse-grained features instead of applying custom probabilistic modeling and algorithm to coarse-grained and fine-grained features.

In Random Model, first, with a given number of SRAM blocks we determine which and how many blocks are faulty by using P_S value. Then, using P_F value we calculate how many cells are going to be faulty. When it is calculated, we randomly inject faults in cells of randomly selected faulty SRAM blocks. Because of the uniform random distribution, every cell in faulty blocks has the same probability. The differences between Mixed Model and Random Model are summarized in Table III.

In Figure 6, we call Mixed Model and Random Model Artificial Models. To understand how accurate our approximate model we evaluate empirical data also called Real Data. Since evaluated networks utilize 850 SRAM blocks at maximum, we randomly choose the required amount of SRAM blocks.

2. Bit-Precision & Bit-Mapping Unit. Quantization and undervolting are both effective techniques to improve the energy efficiency of DNNs. However, they may lead to accuracy loss with aggressive exploitation. MoRS enables us to explore their correlation to find an optimal operating point.

In this step, we change weights according to precision and mapping options. Caffe's weights are 32-bit single-precision floating points. Since each row of SRAM block has 16-bit, we store those weights in two rows when precision is not reduced.

We use four fixed point precisions: 16-bit half-precision floating point, 8-bit (Q.4.4), 4-bit (Q2.2), and binary. To enable fixed-precision options we use a prior study [65] an adapted version of the original Caffe with limited numerical precision of weights. For 16-bit half-precision floating point, we use NVCaffe [66], NVIDIA-maintained Caffe that supports 16-bit half-precision floating point train and inference. When precision is reduced to $X - bit$, we store $16/X$ weights in one row consecutively. Therefore, by reducing precision, the usage of SRAM blocks and power consumption decrease with a cost of accuracy loss.

In addition to precision options, we change the mapping of weights to SRAM blocks. There are four mapping options: MSB, LSB, the first half of bit MSB, and another half of bits LSB, the first half of bits LSB, and another half of bit MSB. MSB means the most significant bit of weights maps to the first cell of a row whereas LSB means the least significant bit of weights maps to the first cell of a row.

3. Fault Injection Unit (Weights to SRAM Mapping). After precision and mapping options we obtain updated weights. In this step, we use artificial models to inject faults in updated weights. Each cell of artificial models contains either faulty or healthy information. If a bit of weight is mapped in the faulty cell, we flip its value. Else, the value is not changed. When this bit-flip operation performs, sometimes the value of weights could be infinity or NaN. To prevent this situation we mask these values to either one or zero. The masking operation is only performed for 32-bit single-precision and 16-bit half-precision floating point. Because fixed point does not have any mantissa or exponent parts to converge NaN or infinite value. In our study, the largest fixed point representation is $Q_{4.4}$ and its maximum value is 15.

TABLE V: Details of evaluated neural networks

NN Model	LeNeT-5 [35]	cuda-convnet [36]
Dataset Name	MNIST [46]	CIFAR-10 [47]
# of Weights	430500	89440
# of SRAM Blocks Utilized	850	180
Inference Accuracy (%)	99.05%	79.59%

4. DNN Inference. After injecting faults to the weights, we use Caffe Framework to measure the accuracy of neural networks. Our most accurate baseline is Real Data. We train our model based on part of real data and we test it using another of that data. To diversify we have four different bit-mapping, three different precision, and two different masking options for each voltage level.

V. Experimental Results

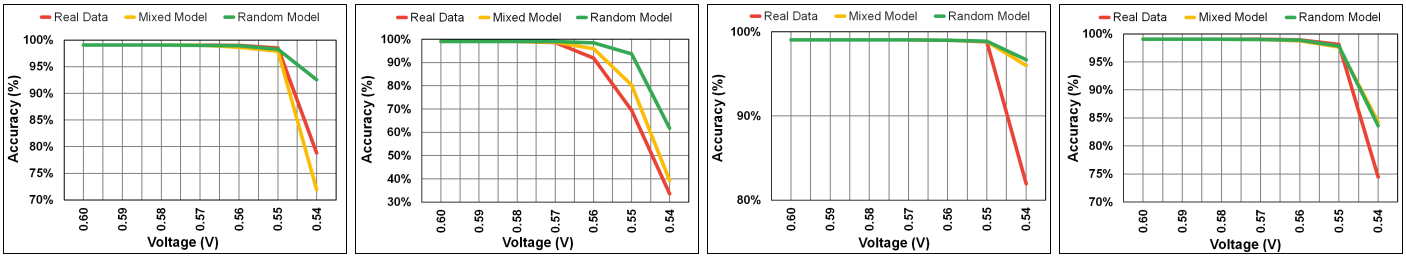
As we mentioned in the previous section we use Caffe [64], a deep learning framework, and test the output of MoRS and random fault injection model on two different neural network models: LeNeT-5 [35] with MNIST dataset [46] and cuda-convnet [36] with CIFAR-10 dataset [47]. We perform different bit-mappings and value maskings for each neural network architecture. In addition to these tests, we also perform reduced precision tests on LeNeT-5. Details of each evaluated benchmark are summarized in Table V.

A. Overall Resilience

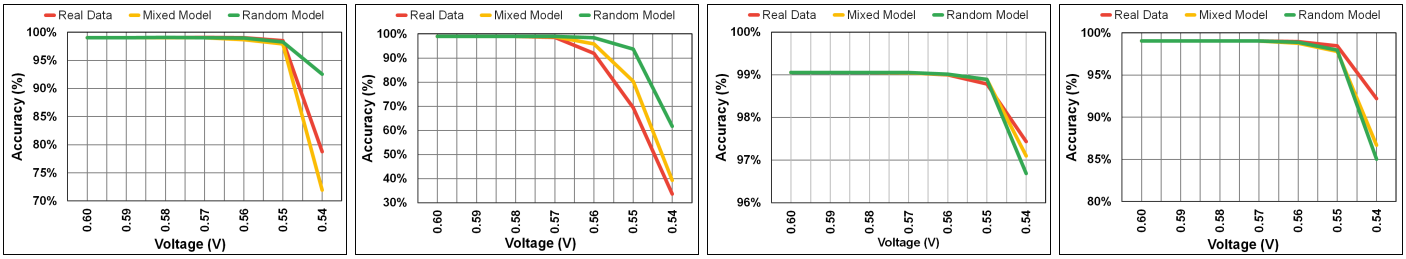
Figure 7 shows the accuracy of LeNeT-5 [35] on the MNIST dataset [46] with different bit-mapping and value masking options. We observe that in all options Mixed Model is more precise than Random Model. Also, if the application becomes less resilient, Mixed Model is closer to the real data and the gap between the baseline and Mixed Model is increasing in terms of how close they are to the real data.

We observe that that if we mask infinity and NaN value to 0, the LeNeT-5 network is more resilient than masking to 1. We think that the cause of this situation is in the MNIST dataset hand-written digits are represented by one and the rest of the background is represented by zero. Therefore, ones more impact than zeros when it comes to classification.

We see that MSB and $MSB \mid LSB$ mapping cause more faults and are less resilient than LSB and $LSB \mid MSB$ mapping. We find that the cause of this is undervolting-based faults generally occur in first cells. Since MSB means the most significant bit, when bit-flips happen it affects the corresponding value more than others. Figure 7b and Figure



(a) $Infinity - NaN\ value \leftarrow 0$ $Mapping \leftarrow MSB$ (b) $Infinity - NaN\ value \leftarrow 1$ $Mapping \leftarrow MSB$ (c) $Infinity - NaN\ value \leftarrow 0$ $Mapping \leftarrow LSB$ (d) $Infinity - NaN\ value \leftarrow 1$ $Mapping \leftarrow LSB$



(e) $Infinity - NaN\ value \leftarrow 0$ $Mapping \leftarrow MSB \mid LSB$ (f) $Infinity - NaN\ value \leftarrow 1$ $Mapping \leftarrow MSB \mid LSB$ (g) $Infinity - NaN\ value \leftarrow 0$ $Mapping \leftarrow LSB \mid MSB$ (h) $Infinity - NaN\ value \leftarrow 1$ $Mapping \leftarrow LSB \mid MSB$

Fig. 7: Voltage and resilience behavior of artificial models and experimental (real) data on LeNeT-5 [35] network for each bit-mapping and value masking option when precision is not reduced (32-bit single-precision floating point.)

7f show more characteristic behavior and are less resilient than others since both have MSB mappings and masking to 1 option.

In Figure,7 we perform 300 iterations for each option (value masking, bit mapping). Then we average these options for LeNeT-5 and cuda-convnet [36] network architectures. In addition to these, we perform these experiments at different precision levels for LeNeT-5 and for each precision level, we average all options.

Figure 8 shows the accuracy of cuda-convnet on the CIFAR-10 dataset [47] without reducing precision, *i.e,* the precision of weights is 32-bit floating points. As we evaluate the LeNeT-5 [35] network in Figure 9a, we average every bit-mapping and value masking option for cuda-convnet [36] network.

We find that the cause of LeNeT-5 works better is cuda-convnet on CIFAR-10 is rather smaller in terms of the number of weights and SRAM block utilization. However, even in this network, Mixed Model is 1.47x more accurate than Random Model in terms of accuracy difference with Real Data on average. While the supply voltage of SRAM blocks is reducing, we observe more characteristic and distinguishing behavior from real data. At V_{crash} level, the accuracy difference between Mixed Model and Real Data is only 6% whereas between Random Model and Real Data is 10%.

B. Quantization

Figure 9 shows the accuracy of LeNeT-5 on the MNIST dataset at different precision levels. We see that in each precision level our the Mixed Model has more similar behavior than Random Model has to the real data. Figure 9a is the average of all options demonstrated in Figure 7.

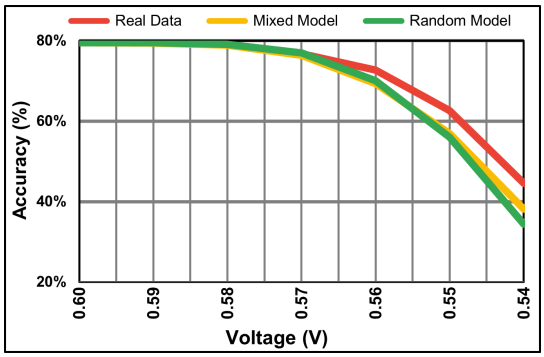
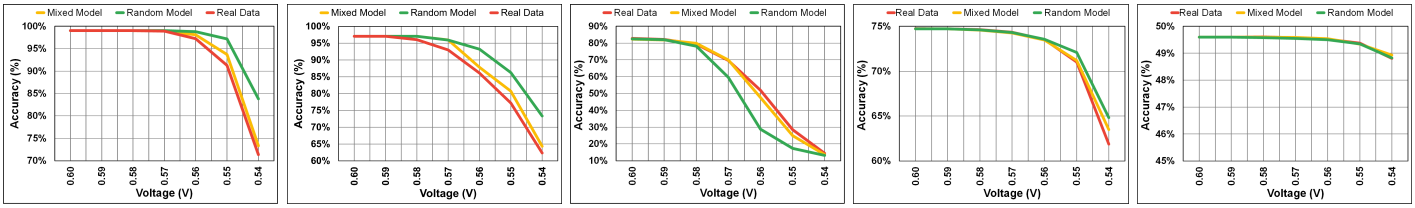


Fig. 8: Average of all options on voltage and resilience behaviour of artificial models and experimental (real) data on cuda-convnet network without reducing precision.

Figure 9a shows the resilience of both artificial models and real data under nominal precision level (32-bit single-precision floating point) in terms of accuracy. Mixed Model is 3.74x closer than Random Model to the real data on average. At V_{crash}, the accuracy difference between Mixed Model and Real Data is 2.46%, on the other hand, the difference between Random Model and Real Data is 12.47%.

Figure 9b shows the accuracy of LeNeT for weights with the precision reduced to 16-bit half-precision floating point. The behavior of all models are similar to the 32-bit architecture, since 16-bit precision is mostly cover all values in 32-bit [67], [68] without significant affect on accuracy. The highest accuracy difference between real data and Mixed Model is 4.47% at 0.55V whereas the highest accuracy difference between real data and Random Model is 9.05% at 0.57V. The average accuracy difference of all voltage level between real data and



(a) 32-bit single-precision floating point (b) 16-bit half-precision floating point (c) 8-bit fixed point (Q4.4) (d) 4-bit fixed point (Q2.2) (e) 1-bit (Binary)

Fig. 9: Average of all options on voltage and resilience behaviour of artificial models and experimental (real) data on LeNeT-5 network with different precisions.

Mixed Model is 2.25% whereas between real data and Random Model is 4.84%, which is 2.15x worse than Mixed Model in terms of similarity of the behavior of real data.

Figure 9c shows the accuracy of LeNeT with 8-bit precision for weights. We see that compared to 32-bit, the network are less resilient to undervolting faults. On average, Mixed Model is 7x better than Random Model in terms of approximation to real data. At $560mV$, the accuracy gap between Mixed Model and Real Data is 4% while the difference between Random Model and Real Data is 23%.

Figure 9d shows the accuracy of both artificial models and real data when the precision of weights reduces to 4-bit. We see that reducing the precision of weights 8-bit to 4-bit network become more resilient and more fault-tolerant to errors based on voltage underscaling. Although, at V_{nom}, 8-bit LeNeT is more accurate than 4-bit LeNeT, at V_{crash}, 8-bit accuracy is 14% whereas 4-bit accuracy is 62%. Even in this unexpected situation, artificial models of the MoRS framework have similar behavior to real data. On average, Mixed Model is 2x closer to real data than Random Model. At $550mV$, the difference in accuracy between Mixed Model and real data is around 1.5%, while the difference in accuracy between Random Model and real data is 3%.

Figure 9e shows the accuracy of LeNeT when precision reduces to 1-bit. In 1-bit tests, we map weights to three different value sets. First one is {-1,1}, second is {-1,0} and the last one is {0,1}. However, we have not observed much difference between value sets. As we see in 4-bit, 1-bit LeNeT network becomes more resilient to faults. The difference between V_{nom} accuracy and V_{crash} accuracy is 0.79%. Both artificial models have the same behavior and do not have significant difference in accuracy. Mixed Model has 0.03% difference in accuracy, whereas Random Model is 0.02%. Because of these negligible statistics, we do not add 1-bit LeNeT to Figure 10.

To point out the resilience of reduced precision networks, we examine the accuracy drop between V_{nom} and V_{crash}. The drop is 68.4%, 12.9% and 0.79% for 8-bit, 4-bit and 1-bit LeNet, respectively.

Table VI shows the SRAM utilization and accuracy in various precision levels of weights. Inference accuracy represents the accuracy at nominal voltage level (V_{nom}). 32-bit FP denotes the 32-bit single-precision floating point, 16-bit Half FP stands for 16-bit half-precision floating points.

LeNeT-5 Precision	SRAM Utilization	Accuracy
32-bit FP	850	99.05%
16-bit Half FP	425	97.03%
8-bit (Q4.4)	213	82.79%
4-bit (Q2.2)	107	74.75%
1-bit (Binary)	27	49.59%

TABLE VI: LeNeT-5 SRAM Block Utilization and Inference Accuracy(at V_{nom}) under different precisions

C. Comparison of Artificial Models

Figure 10 shows the accuracy gap between Real Data and both two artificial models. We see that on the average of all benchmarks, Mixed Model is 3.21x closer than Random Model on average. For most of the benchmarks, the maximum difference in accuracy between Mixed Model and Real Model is under 5%. However, for the Random Model, the maximum difference in accuracy is 23.2%.

We conclude that the proposed model not only has the same behavior with real data against undervolting effects but also can imitate real data with a tolerable difference in terms of accuracy. Most importantly, if the system is not resilient to faults, randomized fault injection does not show the behavior of real data. To be precise in how undervolting affects systems, it has to profile the real data in fine-grained. Coarse-grained features are insufficient to model real SRAM behavior when undervolting is performed.

VI. Related Work

To the best of our knowledge, this study provides the first approximate fault modeling framework and injection in voltage underscaled SRAMs. In this section, we discuss related work on fault injection and modeling on undervolted systems and the resilience of DNNs.

Resilience of DNNs. DNNs are inherently reliable to faults. However, in harsh environments, process variations, voltage underscaling can cause significant accuracy loss. Reagen et al. [16] propose Minerva a fault mitigation mechanism to mitigate low-voltage SRAM faults effects in DNN accelerators. Salami et al. [3] study undervolting SRAM-based FPGA on-chip memories and present intelligently-constrained BRAM placement to mitigate undervolting faults in the NN classification layer. Torres-Huitzil et al. [29] present a comprehensive review

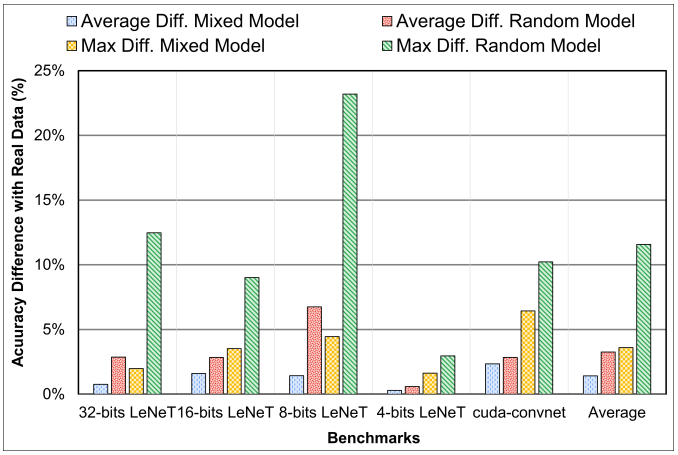


Fig. 10: The accuracy difference between Real Data and each artificial model for all benchmarks.

on fault and error tolerance in neural networks and mention mitigation and fault injection techniques. They also mention that more realist/novel fault models need to be developed to understand the effects of faults on neural networks deeply as an open challenge. Deng et al. [30] present a retraining-based mitigation technique for neural networks to become more resilient.

Fault injection. Fault injection is a widely used technique in resilience studies. Also, fault injection is used as bit-flip-based adversarial weight attacks also called bit-flip attacks(BFA) [69]–[71] and fault-injection attacks [72], [73]. Many studies focus on the reliability and resilience of systems on soft-errors, noise [63], [74]–[76], and voltage underscaling by injecting faults. Koppula et al. [34] propose a framework, EDEN, that proposes combining training on profiled DRAM faults in terms of timing violations and voltage underscaling with mitigation strategies and mappings. EDEN provides four different error models since uniform random distribution does not cover whole DRAMs. Chatzidimitrou et al. [32], [43] inject faults randomly to branch prediction units to examine the effects of voltage underscaling. Chandramoorthy et al. [17] study the undervolting faults in different layers of networks by injecting faults to SRAMs randomly and do not take account of the patterns or spatial distribution of bit errors. Stutz et al. [77] propose random bit error training assumed voltage underscaled SRAMs faults distribute randomly. Salami et al. [33] study the resilience of RTL NN Accelerators and fault characterization and mitigation. To characterize and mitigate they assume that each bitcell of SRAMs has the same probability. Yang et al. [23] study energy-efficient CNNs by performing voltage scaling on SRAMs. To study the effect of bit errors they hypothesize that errors in SRAM are roughly uniformly distributed. The prior work [78] on Near-Threshold Voltage FinFET SRAMs presents a fault model for SRAMs based on uniform random distribution. Givaki et al. [79] study the resilience of DNNs under reduced voltage SRAM-based FPGA on-chip memories by using directly the experimental data to examine the training phase of DNNs.

All of these undervolted on-chip fault injection studies perform injection randomly and do not take into account fine-grained profiling such as spatial distances between cells, row-based and column-based approaches. Randomly injecting faults approach can cause misleading to understand how the system works under the low-voltage domain. As we mentioned in Section 4, if the system does not have much resilience to voltage underscaling, the Mixed Model of MoRS is 7x closer than the randomly injected model to the real data.

VII. Conclusion

In this paper, we propose MoRS, a framework that generates the first approximate fault injection model *i.e.,* artificial fault maps. The advantage of the proposed framework is to inject errors into various systems including heterogeneous computing devices. We evaluated the accuracy of the proposed framework for state-of-the-art DNN applications. To evaluate our proposed model, we measure the difference in accuracy between artificial error models and real data. We show that compared to random model-based error injections, the proposed model can provide 3.21x on average closer than to the real data.

Acknowledgement

This work is partially funded by Open Transprecision Computing (OPRECOM) project, Summer of Code 2020.

References

[1] H. Jeon, G. S. Ravi, N. S. Kim, and M. Annavaram, "Gpu register file virtualization," in *Proceedings of the 48th International Symposium on Microarchitecture*, ser. MICRO-48. New York, NY, USA: Association for Computing Machinery, 2015, p. 420–432. [Online]. Available: https://doi.org/10.1145/2830772.2830784

[2] J. Haj-Yihia, A. Yasin, Y. B. Asher, and A. Mendelson, "Fine-grain power breakdown of modern out-of-order cores and its implications on skylake-based systems," *ACM Trans. Archit. Code Optim.*, vol. 13, no. 4, Dec. 2016. [Online]. Available: https://doi.org/10.1145/3018112

[3] B. Salami, O. S. Unsal, and A. C. Kestelman, "Comprehensive evaluation of supply voltage underscaling in fpga on-chip memories," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2018, pp. 724–736.

[4] G. Papadimitriou, A. Chatzidimitriou, D. Gizopoulos, V. J. Reddi, J. Leng, B. Salami, O. S. Unsal, and A. C. Kestelman, "Exceeding conservative limits: A consolidated analysis on modern hardware margins," *IEEE Transactions on Device and Materials Reliability*, vol. 20, no. 2, pp. 341–350, 2020.

[5] D. Gizopoulos, G. Papadimitriou, A. Chatzidimitriou, V. J. Reddi, B. Salami, O. S. Unsal, A. C. Kestelman, and J. Leng, "Modern hardware margins: Cpus, gpus, fpgas recent system-level studies," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2019, pp. 129–134.

[6] A. Bacha and R. Teodorescu, "Using ecc feedback to guide voltage speculation in low-voltage processors," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014, pp. 306–318.

[7] G. Papadimitriou, A. Chatzidimitriou, and D. Gizopoulos, "Adaptive voltage/frequency scaling and core allocation for balanced energy and performance on multicore cpus," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 133–146.

[8] K. Parasyris, P. Koutsovasilis, V. Vassiliadis, C. D. Antonopoulos, N. Bellas, and S. Lalis, "A framework for evaluating software on reduced margins hardware," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2018, pp. 330–337.

[9] G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, D. Gizopoulos, P. Lawthers, and S. Das, "Harnessing voltage margins for energy efficiency in multicore cpus," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017, pp. 503–516.

[10] A. Bacha and R. Teodorescu, "Dynamic reduction of voltage margins by leveraging on-chip ecc in itanium ii processors," *SIGARCH Comput. Archit. News*, vol. 41, no. 3, p. 297–307, Jun. 2013. [Online]. Available: https://doi.org/10.1145/2508148.2485948

[11] A. Zou, J. Leng, X. He, Y. Zu, C. D. Gill, V. J. Reddi, and X. Zhang, "Voltage-stacked gpus: A control theory driven cross-layer solution for practical voltage stacking in gpus," in *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-51. IEEE Press, 2018, p. 390–402. [Online]. Available: https://doi.org/10.1109/MICRO.2018.00039

[12] B. Salami, E. B. Onural, I. E. Yuksel, F. Koc, O. Ergin, A. C. Kestelman, O. S. Unsal, H. Sarbazi-Azad, and O. Mutlu, "An experimental study of reduced-voltage operation in modern fpgas for neural network acceleration," 2020.

[13] B. Salami, O. S. Unsal, and A. C. Kestelman, "Evaluating built-in ecc of fpga on-chip memories for the mitigation of undervolting faults," in *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. IEEE, 2019, pp. 242–246.

[14] B. Salami, O. Unsal, and A. Cristal, "Fault characterization through fpga undervolting," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 85–853.

[15] B. Salami, "Aggressive undervolting of fpgas: power & reliability tradeoffs," 2018.

[16] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 267–278.

[17] N. Chandramoorthy, K. Swaminathan, M. Cochet, A. Paidimarri, S. Eldridge, R. V. Joshi, M. M. Ziegler, A. Buyuktosunoglu, and P. Bose, "Resilient low voltage accelerators for high energy efficiency," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 147–158.

[18] J. Zhang, K. Rangineni, Z. Ghodsi, and S. Garg, "Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[19] K. K. Chang, A. G. Yağlıkçı, S. Ghose, A. Agrawal, N. Chatterjee, A. Kashyap, D. Lee, M. O'Connor, H. Hassan, and O. Mutlu, "Understanding reduced-voltage operation in modern dram devices: Experimental characterization, analysis, and mechanisms," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 1, Jun. 2017. [Online]. Available: https://doi.org/10.1145/3084447

[20] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling," in *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ser. ICAC '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 31–40. [Online]. Available: https://doi.org/10.1145/1998582.1998590

[21] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini, "Memscale: Active low-power modes for main memory," in *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XVI. New York, NY, USA: Association for Computing Machinery, 2011, p. 225–238. [Online]. Available: https://doi.org/10.1145/1950365.1950392

[22] S. S. N. Larimi, B. Salami, O. S. Unsal, A. C. Kestelman, H. Sarbazi-Azad, and O. Mutlu, "Understanding power consumption and reliability of high-bandwidth memory with voltage underscaling," 2020.

[23] L. Yang and B. Murmann, "Sram voltage scaling for energy-efficient convolutional neural networks," in *2017 18th International Symposium on Quality Electronic Design (ISQED)*, 2017, pp. 7–12.

[24] L. Yang and B. Murmann, "Approximate sram for energy-efficient, privacy-preserving convolutional neural networks," *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 689–694, 2017.

[25] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Crista, O. S. Unsal, and K. Mai, "Error analysis and retention-aware error management for nand flash memory." *Intel Technology Journal*, vol. 17, no. 1, 2013.

[26] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error characterization, mitigation, and recovery in flash-memory-based solid-state drives," *Proceedings of the IEEE*, vol. 105, no. 9, pp. 1666–1704, 2017.

[27] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Error patterns in mlc nand flash memory: Measurement, characterization, and analysis," in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2012, pp. 521–526.

[28] Y. Cai, Y. Luo, S. Ghose, and O. Mutlu, "Read disturb errors in mlc nand flash memory: Characterization, mitigation, and recovery," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2015, pp. 438–449.

[29] C. Torres-Huitzil and B. Girau, "Fault and error tolerance in neural networks: A review," *IEEE Access*, vol. 5, pp. 17 322–17 341, 2017.

[30] J. Deng, Y. Fang, Z. Du, Y. Wang, H. Li, O. Temam, P. Ienne, D. Novo, X. Li, Y. Chen, and C. Wu, "Retraining-based timing error mitigation for hardware neural networks," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2015, pp. 593–596.

[31] B. Salami., "FPGA BRAMs Undervolting Study," https://github.com/behzadsalami/FPGA-BRAMs-Undervoltig-Study, 2018.

[32] A. Chatzidimitriou, G. Papadimitriou, D. Gizopoulos, S. Ganapathy, and J. Kalamatianos, "Analysis and characterization of ultra low power branch predictors," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*, 2018, pp. 144–147.

[33] B. Salami, O. S. Unsal, and A. C. Kestelman, "On the resilience of rtl nn accelerators: Fault characterization and mitigation," in *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2018, pp. 322–329.

[34] S. Koppula, L. Orosa, A. G. Yağlıkçı, R. Azizi, T. Shahroodi, K. Kanellopoulos, and O. Mutlu, "Eden: Enabling energy-efficient, high-performance deep neural network inference using approximate dram," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '52. New York, NY, USA: Association for Computing Machinery, 2019, p. 166–181. [Online]. Available: https://doi.org/10.1145/3352460.3358280

[35] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[36] A. Krizhevsky., "cuda-convnet," https://github.com/akrizhevsky/cuda-convnet2, 2014.

[37] M. Sabbagh, C. Gongye, Y. Fei, and Y. Wang, "Evaluating fault resiliency of compressed deep neural networks," in *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*, 2019, pp. 1–7.

[38] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3126908.3126964

[39] P. Upadhyay, S. Ghosh, R. Kar, D. Mandal, and S. P. Ghoshal, "Low static and dynamic power mtcmos based 12t sram cell for high speed memory system," in *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2014, pp. 212–217.

[40] T. Azam, B. Cheng, and D. R. S. Cumming, "Variability resilient low-power 7t-sram design for nano-scaled technologies," in *2010 11th International Symposium on Quality Electronic Design (ISQED)*, 2010, pp. 9–14.

[41] G. Chen, D. Sylvester, D. Blaauw, and T. Mudge, "Yield-driven near-threshold sram design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 11, pp. 1590–1598, 2010.

[42] J. Leng, A. Buyuktosunoglu, R. Bertran, P. Bose, and V. J. Reddi, "Safe limits on voltage reduction efficiency in gpus: A direct measurement approach," in *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2015, pp. 294–307.

[43] A. Chatzidimitriou, G. Panadimitriou, D. Gizopoulos, S. Ganapathy, and J. Kalamatianos, "Assessing the effects of low voltage in branch prediction units," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2019, pp. 127–136.

[44] L. Rivière, Z. Najm, P. Rauzy, J. Danger, J. Bringer, and L. Sauvage, "High precision fault injections on the instruction cache of armv7-m architectures," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 62–67.

[45] M. Kaliorakis, S. Tselonis, A. Chatzidimitriou, N. Foutris, and D. Gizopoulos, "Differential fault injection on microarchitectural simulators," in *2015 IEEE International Symposium on Workload Characterization*, 2015, pp. 172–182.

[46] Y. Lecun and C. Cortes, "The MNIST database of handwritten digits." http://yann.lecun.com/exdb/mnist/, 1999.

[47] A. Krizhevsky, V. Nair, and G. Hinton, "he CIFAR-10 dataset," https://www.cs.toronto.edu/~kriz/cifar.html.

[48] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," 2017.

[49] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," 2016.

[50] Z. Zhu, H. Sun, Y. Lin, G. Dai, L. Xia, S. Han, Y. Wang, and H. Yang, "A configurable multi-precision cnn computing framework based on single bit rram," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3316781.3317739

[51] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2017.

[52] R. Yazdani, M. Riera, J. Arnau, and A. González, "The dark side of dnn pruning," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 790–801.

[53] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," 2015.

[54] Y. Shen, M. Ferdman, and P. Milder, "Escher: A cnn accelerator with flexible buffering to minimize off-chip transfer," in *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2017, pp. 93–100.

[55] C. Deng, S. Liao, Y. Xie, K. K. Parhi, X. Qian, and B. Yuan, "Permdnn: Efficient compressed dnn architecture with permuted diagonal matrices," 2020.

[56] Y. Shen, M. Ferdman, and P. Milder, "Maximizing cnn accelerator efficiency through resource partitioning," 2018.

[57] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 161–170. [Online]. Available: https://doi.org/10.1145/2684746.2689060

[58] M. Riera, J. Arnau, and A. Gonzalez, "Computation reuse in dnns by exploiting input similarity," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 57–68.

[59] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[60] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *CoRR*, vol. abs/1712.05877, 2017. [Online]. Available: http://arxiv.org/abs/1712.05877

[61] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," *CoRR*, vol. abs/1512.06473, 2015. [Online]. Available: http://arxiv.org/abs/1512.06473

[62] K. Ueyoshi, K. Ando, K. Hirose, S. Takamaeda-Yamazaki, J. Kadomoto, T. Miyata, M. Hamada, T. Kuroda, and M. Motomura, "Quest: A 7.49tops multi-purpose log-quantized dnn inference engine stacked on 96mb 3d sram using inductive-coupling technology in 40nm cmos," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 216–218.

[63] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G. Wei, "Ares: A framework for quantifying the resilience of deep neural networks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[64] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 675–678. [Online]. Available: https://doi.org/10.1145/2647868.2654889

[65] M. Milde, D. Neil, A. Aimar, T. Delbruck, and G. Indiveri, "Adaption: Toolbox and benchmark for training convolutional neural networks with reduced numerical precision weights and activation," 11 2017.

[66] NVIDIA., "NVIDIA-caffe extension," https://github.com/NVIDIA/caffe, 2017.

[67] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh *et al.*, "Mixed precision training," *arXiv preprint arXiv:1710.03740*, 2017.

[68] S. Markidis, S. W. Der Chien, E. Laure, I. B. Peng, and J. S. Vetter, "Nvidia tensor core programmability, performance & precision," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2018, pp. 522–531.

[69] A. S. Rakin, Z. He, J. Li, F. Yao, C. Chakrabarti, and D. Fan, "T-bfa: Targeted bit-flip adversarial weight attack," 2021.

[70] Z. He, A. S. Rakin, J. Li, C. Chakrabarti, and D. Fan, "Defending and harnessing the bit-flip based adversarial weight attack," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 14 083–14 091.

[71] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," 2019.

[72] Y. Liu, L. Wei, B. Luo, and Q. Xu, "Fault injection attack on deep neural network," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 131–138.

[73] M. S. Kelly, K. Mayes, and J. F. Walker, "Characterising a cpu fault attack model via run-time data analysis," in *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2017, pp. 79–84.

[74] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3126908.3126964

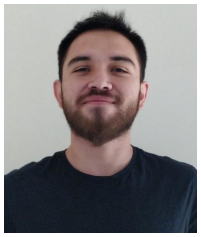
[75] M. A. Neggaz, I. Alouani, P. R. Lorenzo, and S. Niar, "A reliability study on cnns for critical embedded systems," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*, 2018, pp. 476–479.

[76] A. H. Salavati and A. Karbasi, "Multi-level error-resilient neural networks," in *2012 IEEE International Symposium on Information Theory Proceedings*, 2012, pp. 1064–1068.

[77] D. Stutz, N. Chandramoorthy, M. Hein, and B. Schiele, "Bit error robustness for energy-efficient dnn accelerators," 2020.

[78] S. Ganapathy, J. Kalamatianos, K. Kasprak, and S. Raasch, "On characterizing near-threshold sram failures in finfet technology," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017, pp. 1–6.

[79] K. Givaki, B. Salami, R. Hojabr, S. R. Tayaranian, A. Khonsari, D. Rahmati, S. Gorgin, A. Cristal, and O. S. Unsal, "On the resilience of deep learning for reduced-voltage fpgas," in *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. IEEE, 2020, pp. 110–117.



İsmail Emir Yüksel is an MSc student and researcher in the Computer Engineering Department of TOBB University of Economics and Technology (TOBB ETÜ). He received his BSc in Electrical Electronics Engineering from TOBB University of Economics and Technology in 2019. His research interests are energy-efficient heterogeneous computing and low-power & fault-resilient hardware accelerators.

Behzad Salami is a post-doctoral researcher in the Computer Science (CS) department of Barcelona Supercomputing Center (BSC) and an affiliated research member of SAFARI Research Group at ETH Zurich. He received his Ph.D. with honors in Computer Architecture from Universitat Politécnica de Catalunya (UPC) in 2018. Also, he obtained MSc and BSc degrees in Computer Engineering from Amirkabir University of Technology (AUT) and Iran University of Science and Technology (IUST), respectively. He has received mutiple awards and grants for his research. His research interests are heterogeneous systems, low-power & fault-resilient hardware accelerators, and near-data processing systems. Contact him at: behzadsalami@gmail.com

Oğuz Ergin is a professor in the department of computer engineering in TOBB University of Economics and Technology. He received his BS in electrical and electronics engineering from Middle East Technical University, MS, and Ph.D. in computer science from the State University of New York at Binghamton. He was a senior research scientist in Intel Barcelona Research Center prior to joining TOBB ETÜ. He is currently leading a research group in TOBB ETÜ working on energy-efficient, reliable, and high-performance computer architectures.

Osman Sabri Ünsal is co-manager of the Parallel Paradigms for Computer Architecture research group at Barcelona Supercomputing Center (BSC). He got his B.S., M.S., and Ph.D. in Computer Engineering from Istanbul Technical University, Brown University, and the University of Massachusetts, Amherst respectively. His current research interests are in computer architecture, fault-tolerance, energy-efficiency, and heterogeneous computing. He is currently leading LEGaTO EU H2020 research project on heterogeneous energy-efficiency computing.

Adrián Cristal Kestelman received the Licenciatura degree in Computer Science from the Faculty of Exact and Natural Sciences, Universidad de Buenos Aires, Buenos Aires, Argentina, in 1995, and the Ph.D. degree in Computer Science from the Universitat Politécnica de Catalunya (UPC), Barcelona, Spain. Since 2006, he is a co-manager of the Computer Architecture for Parallel Paradigms Research Group at Barcelona Supercomputing Center (BSC). His current research interests include the areas of microarchitecture, multicore, and heterogeneous architectures, and programming models for multicore architectures. Currently, he is leading the architecture development of the vector processor unit in the European Processor Initiative.