Design Exploration of Fault-Tolerant Deep Neural Networks Using Posit Number Representation System

Morteza Yousefloo and Omid Akbari

***Abstract*— The applications of deep neural networks (DNNs) in different safety-critical systems (such as autonomous vehicles and robotics) are experiencing emerging growth due to their high accuracy and potential for solving complex problems. However, a single failure in the hardware performing these DNN models can lead to irreparable results. Thus, improving the resilience of these models to transient faults (i.e., soft errors) has been of great interest in recent years. However, the traditional hardware redundancy techniques (such as TMR) are not cost-efficient due to their high resource overheads. In this article, we explored the potential of leveraging the posit number representation system for composing the DNN models, to achieve higher fault tolerance compared to the conventional fixed-point and IEEE 754 32-bit floating-point (FLOAT) number representation-based DNNs, without incurring significant overheads of the traditional hardware redundancy techniques. Posit numbers are composed of four fields, including a sign bit, regime value, exponent, and fraction bits, where the regime value does not exist in the FLOAT numbers. Our explorations are performed at the model, layer, and bitwise levels to determine the most appropriate posit format (e.g., the bit width of different posit fields) for composing the fault-tolerant DNN models. We studied the different posit, fixed-point, and FLOAT-based LeNet-5 and ResNet-50 networks trained with the MNIST and CIFAR-10 datasets, respectively. We then proposed a hardware-level method for error detection and correction of posit-based DNN models. Based on the results, the fault tolerance of the posit-based DNNs outperforms the FLOAT-based models, at each of the three investigated levels, where a 32-bit posit-based DNN achieved up to 15% more classification accuracy than the FLOAT-based one, for the LeNet-5 network. Also, in comparison with the fixed-point models, the posit-based networks showed up to 23% higher accuracy. Moreover, the enhanced 8-bit posit-based DNN that employed the proposed error detection and correction method results in, up to 11% and 41% higher classification accuracy than the unprotected posit and conventional FLOAT-based DNNs, respectively.**

***Index Terms*— Deep neural networks (DNNs), fault-tolerant, float, hardware, number representation system, posit, reliability.**

I. Introduction

THE close-to-human-level precision and ability to solve complex problems make the deep neural networks (DNNs) as appealing models to employ in safety-critical applications, such as autonomous vehicles [1], medical diagnosis [2], and airborne collision avoidance systems [3]. These applications are mostly required to be real-time, energy-efficient, and meet a rigid constraint on output accuracy, while their computing-intensive nature and requirement of considerable hardware resources restrict them from being executed on general-purpose processors (GPPs).

In recent years, many efforts have been devoted to designing efficient DNN accelerators to overcome the aforementioned challenges and meet the application demands, such as Google's tensor processing units (TPUs) [4], Nvidia deep learning accelerator (NVDLA) [5], and Eyeriss [6]. However, any single error in these accelerators while performing safety-critical applications may lead to catastrophic situations and irreparable consequences. Errors may occur due to different factors, such as noisy environments, electromagnetic radiations, manufacturing defects, and aging phenomena [7]. Moreover, technology scaling may worsen the susceptibly of these accelerators (e.g., TPU V4.0 was fabricated in 7-nm technology [8]) to the encountered error factors. Higher level integration of transistors in a given area and a lower charge threshold required to change the voltage level of transistors are some reasons for the high sensitivity of integrated circuits (ICs) in the nanoscale era [9].

As an example, a misdiagnosis of the road signs by an autonomous vehicle (e.g., an incorrect diagnosis of the stop sign as the maximum speed of 80 km/h sign, shown in Fig. 1) may result in a disastrous accident and affect human life [10]. Therefore, considering fault tolerance is unavoidable in such safety-critical systems.

DNNs inherently have a level of fault tolerance. However, for a given network that was properly trained, the weights of the network are stored in the weight memory and will be used many times during the inference phase. In the case that a part of the stored weights gets faulty, the network may lead to catastrophic results. This effect will be worsened for the noisier environments. Moreover, with the technology scaling in the nanoscale era, since the required charge to flip the value of memories is lowered, a higher number of transistors inside the chips may be affected by the soft errors [9]. Thus, it is necessary to use fault-tolerance solutions in neural networks deployed in safety-critical applications [7], [11], [12], [13], [14], [15].

Using conventional redundancy techniques, such as duplication or triple modular redundancy (TMR) in these systems can impose significant cost (area) and energy overheads [11]. Thus,

Manuscript received 21 October 2023; revised 7 February 2024, 13 April 2024, and 27 April 2024; accepted 30 April 2024. Date of publication 15 May 2024; date of current version 28 June 2024. *(Corresponding author: Omid Akbari.)*

The authors are with the Department of Electrical and Computer Engineering, Tarbiat Modares University, Tehran 14115-111, Iran (e-mail: morteza_yousefloo@modares.ac.ir; o.akbari@modares.ac.ir).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2024.3396555.

Digital Object Identifier 10.1109/TVLSI.2024.3396555

1063-8210 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

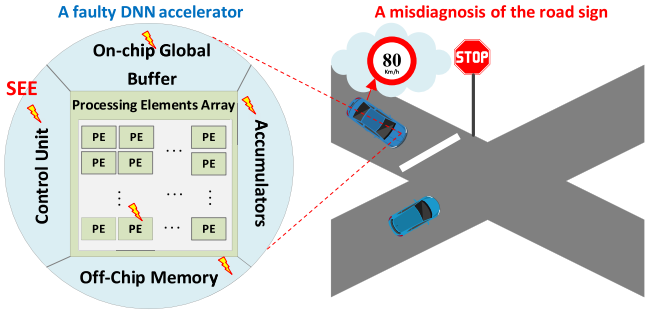

Fig. 1. Effects of single-event errors (SEEs) occurred in the DNN hardware accelerator performing sign detection in an autonomous vehicle.

researchers have suggested different methods to mitigate these overheads, such as fault-aware training [14], fault-aware pruning [15], fault-aware mapping [12], and range restriction [13].

It has been shown that different number representation systems in DNNs' computations may significantly affect performance, energy efficiency, and error resilience [16]. To this end, pioneer companies have proposed various number representation formats. As an example, Google introduced the brain floating-point (BFLOAT 16) format, which is supported in the TPUs [17]. BFLOAT 16 consists of 16 bits, including one sign bit, eight exponent bits, and seven mantissa bits [18]. Also, Flexpoint was proposed by Intel as a 16-bit replacement for IEEE-754 32-bit floating point (FLOAT) [19], which can reduce the required computational resources. NVIDIA [20], Tesla FSD chip [21], IBM, and Greenwaves [22] also have proposed their number representation formats.

Posit number representation system was introduced in 2017 [23]. A posit number is represented by a P(n, es) pair, where n is the total number of bits and es is the number of exponent bits. Compared to the FLOAT, the posit offers a more extensive dynamic range, higher accuracy, simpler hardware, and easier exception management [23]. Although in the initial designs of posit hardware, the area, and energy overhead were more than FLOAT, with the advent of new efficient designs, it has been shown that the posit can offer significantly lower design metrics (i.e., area, power, delay, and energy) compared to the conventional float and fixed-point formats [24]. Moreover, results presented in [25] show significantly lower inference time for almost the same accuracy. In [26], it has been shown that posit-based machine learning systems can be more resilient to errors than other number representation systems, such as FLOAT. However, this research was performed for only an MLP network, and there is not a general fault tolerance exploration of posit-based DNNs, for the different model, layer, and bitwise levels. Moreover, the different posit formats have not been examined in terms of fault tolerance. Thus, a comprehensive investigation is required to extract the fault-tolerant properties of posit-based DNN models.

In this work, we proposed a systematic and methodological framework to explore the fault-tolerant characteristics of the posit-based DNNs. In detail, first, we explore the error resilience of the posit-based DNNs in the three model, layer, and bitwise levels. Toward this, we proposed a fault injection method for the posit-based DNN models to evaluate the

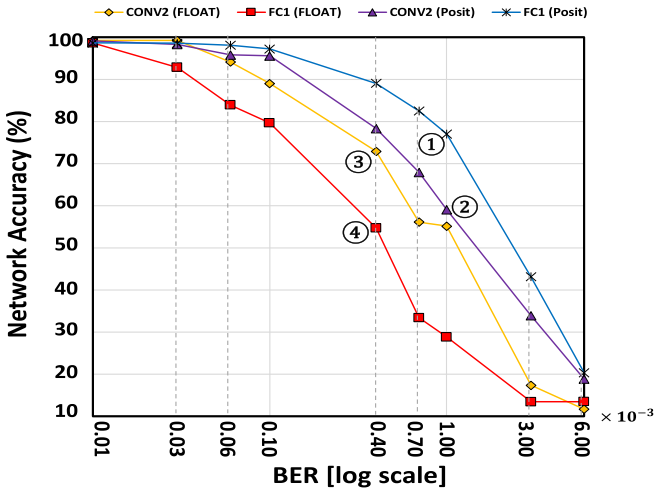

Fig. 2. Accuracy of the different layers of FLOAT and posit-based LeNet-5 networks trained with the MNIST dataset, under various BERs. Key observations. ① and ② CONV2 layer in the posit system is more sensitive to the fault compared to the FC1 layer. ③ and ④ CONV2 layer in the FLOAT system is more error resilient than the FC1 layer. ① –④ Posit-based system is more error resilient compared to the FLOAT-based model.

accuracy of the model for various bit error rates (BERs). Then, we determine the most appropriate pairs of posit [i.e., P(n, es)] for the DNN models in terms of fault tolerance for each of the three investigated levels. Finally, we propose an error detection or correction method to increase the fault tolerance of posit-based DNNs, using the results of the explorations performed in the previous steps.

Our article investigates the error resiliency of DNN models against the soft errors under the various BER, where the fault injection is performed by flipping the bit values of the DNN model's weights (please see Section IV-B). Toward this end, similar to prior state-of-the-art works (e.g., see [7], [26], [27]), the error resiliency of the studied DNN models was examined by the developed software tools (please see Section IV-B). Consequently, by analyzing the behavior of the examined models in Section IV-C, we proposed an error detection and correction method that reduce the effects of soft errors on the DNN model output. The design space of our work is all the possible P(n, es) formats to compose the most error-resilient DNN. Note that, as explained in Section IV-B, we perform this exploration in the three model, layer, and bitwise levels.

A. Motivating Example

Fig. 2 shows the accuracy of the posit and FLOAT-based LeNet-5 models trained by the MNIST dataset under various BERs. The simulation setup for the fault injection method employed in this example will be discussed in Section IV-A. As shown in Fig. 2, almost under the different BERs, fault injection in the fully connected layer 1 (FC1) of the FLOAT-based model drops the network accuracy higher than the fault injection in the convolution layer 2 (CONV2), i.e., the accuracy of the FLOAT-based DNN model is more sensitive to the faults occurred in the FC1 layer.

However, in the posit-based model, FC1 is more error resilient than CONV2. Also, based on the results, the posit-based DNN model provides, on average, 6% and 23%

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

higher error resilient in the CONV2 and FC1, when compared to the FLOAT-based model. This example shows that the fault-tolerant investigation of posit-based DNN models can give useful insight to designers for composing an optimized DNN model in terms of the fault-tolerant, for leveraging them in the safety-critical systems. However, the behavior of the different models/layers of DNNs against fault, as well as the different formats of the posit numbers [i.e., the different pairs of P(n, es)] can be various. As shown in Fig. 2, the fault tolerance behavior of the different layers is different from the float-based model. In detail, the FC1 layer in the posit-based model is more error resilient than the Conv2 layer, whereas this behavior in the same model is reverse for the float-based model, i.e., the Conv2 layer in the float-based model is more error resilient than the Conv2 layer. Moreover, it is seen that in general, the posit-based DNNs can offer higher fault tolerance compared to the float-based ones, where the fault tolerance of both studied layers (i.e., Conv2 and FC1 layers) is significantly higher than the same layers in the float-based model. Based on these two reasons, a comprehensive fault-tolerant design space exploration of the posit-based models is necessary. Addressing this challenge requires a systematic method, which is addressed in this article to compose a fault-tolerant DNN model.

B. Novel Contributions of This Work

Our novel contributions in a nutshell are as follows.

1) Proposing a framework to design fault-tolerant posit-based DNN models.
2) A comprehensive exploration and comparison of the FLOAT, fixed-point, and posit-based DNNs, in terms of fault tolerance (Section IV-C).
3) Exploring various formats of the posit numbers [i.e., the different $P(n, es)$] on the accuracy of the posit-based DNN models to achieve the most error-resilient model (Section IV-C–IV-C1).
4) Investigating the fault tolerance of the posit-based DNN models, at the three model, layer, and bitwise levels (Sections IV-C–IV-E).
5) Case studies of the posit, fixed-point, and FLOAT-based LeNet-5 and ResNet-50 networks trained with the MNIST and CIFAR-10 datasets, respectively (Sections IV-C and IV-D).
6) Investigating the importance of the different fields of a posit number (i.e., the sign bit, regime, exponent, and fraction fields) in fault tolerance of a DNN model (Section IV-E).
7) Proposing an enhanced fault-tolerant scheme by analyzing the behavior of posit-based DNN models against fault, to achieve a higher output accuracy under the same BERs (Section IV-F).

Paper Organization: The rest of this article is organized as follows. Section II reviews prior state-of-the-art works. The required preliminaries are presented in Section III. The proposed framework for analyzing the fault-tolerant posit-based DNNs, fault-injection flow, simulation setup, fault-tolerant analysis of posit and FLOAT-based DNNs, exploring $P(n, es)$ formats to determine the most fault-tolerant format, and model/layer/bitwise fault-tolerant studies of posit-based DNNs are presented in Section IV. In this section, composing enhanced fault-tolerance posit-based DNNs is also presented. Finally, this article is concluded in Section V.

II. Related Works

This section reviews the prior works on fault-tolerant DNNs. Also, some state-of-the-art works on novel number representation systems and their effect on the accuracy of DNN models are studied.

A. Fault-Tolerant DNNs

In some of the recent works (e.g., [7], [27]), the inherent fault tolerance of the DNN models has been examined. In [7], static faults were injected into the DNNs, and then, the sensitivity of various parts, such as the weight memory of the DNN model, was measured. In [27], the convolutional neural networks (CNNs) fault tolerance in a per-layer manner has been investigated, in which the error sensitivity of each layer and its effect on the network's output accuracy were studied. It has also been demonstrated that only hardening the MSB bits of the exponent field in the FLOAT-based model can be sufficient since other bits have less impact on the output accuracy in the presence of faults.

Duplicate modular redundancy (DMR) is one of the well-known techniques for detecting the errors of a system [28]. As an example, Tesla's self-driving car has a computer with two AI brains [21], where the action is taken only if both computers reach the same results. Although higher safety is provided in this system, the performance, cost, and power consumption may be significantly affected. Spatial, temporal, and code redundancy techniques were used in [29] to increase the DNNs' fault tolerance. However, these methods may increase the overall cost, especially when the size of the model grows [10]. In the following, we study some state-of-the-art proposed methods to overcome the aforementioned overheads.

A fault-aware pruning method was proposed in [15] to design fault-tolerant systolic array-based DNN accelerators for the technologies with a high defect rate. In this approach, the training phase is performed in the presence of permanent faults, and it is expected that the network still produces the correct output at the inference phase. This method prunes faulty multiply-and-accumulate (MAC) units and, by retraining, achieves new weights mapped on healthy MACs. However, this method requires to modify the architecture and extract the accelerator fault map that determines the faulty MACs. In [12], a fault-aware mapping method has been proposed, in which the weights with a lower impact on the model's output are mapped to faulty components. However, the fault map of the target hardware platform is a prerequisite of this technique.

Range restriction is another technique to detect the fault in DNN models, in which the range of intermediate computations is determined, and then, if the results are outside of the range, the computation is considered erroneous [12]. This technique was also used in [30], where outputs outside of a given range

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

are labeled faulty and mapped to predefined units. In [31], an algorithm-based method was proposed to increase the fault tolerance of CNNs. As an example, the checksum method is used to robust the computations by detecting and correcting faulty computations in matrix–matrix multiplications.

In [32], a fault injection framework for SNNs and compressed DNNs is proposed, which tries to reduce the time overhead of injecting multiple faults. FIdelity [33] has proposed a framework to check the resilience of hardware accelerators of DNNs, which uses the least amount of information about the hardware to generate accurate software fault models for both datapath and control components without access to RTL. The "fidelity" framework necessitates a minimal degree of high-level design data, which can be derived from architectural descriptions or block diagrams. This information can also be estimated and adjusted for the purpose of sensitivity analysis. harDNNing [34] is a framework that injects faults based on data type and layer. They have used machine learning classification methods to predict the criticality of network parameters and bits, and finally, error correction codes (ECCs) are used to protect critical parameters. In [35], a framework for fault criticality analysis is proposed, which uses digital twins. They have proposed a neural twin for PE to analyze fault criticality. In [36], it is shown that a single fault occurring in a GPU tends to propagate to multiple active threads, significantly reducing the reliability of a CNN. Moreover, relying on error-correcting codes dramatically reduces the number of silent data corruptions (SDCs), while does not reduce the number of critical errors.

In [37], the fault tolerance behavior of CNNs with the two types of data representations, i.e., FLOAT and fixed point, has been assessed. As they have shown, the fixed-point-based LeNet-5 provides a better tradeoff between memory footprint and error resiliency compared to floating-point data. DeepVigor [38] suggests a methodology for conducting resilience analysis on DNNs that allows for a quicker reliability assessment compared to traditional fault injection techniques. Furthermore, DeepVigor introduces and acquires vulnerability ranges for all neurons within DNNs, facilitated by a fault propagation analysis. This process enables precise categorization of faults into critical and noncritical categories.

CLARINET [39] is a comprehensive framework for a GPP that is proposed to facilitate empirical studies in posit arithmetic. The system is composed of two primary constituents—Melodica, a configurable posit arithmetic unit that facilitates quire-based arithmetic, and Clarinet, a RISC-V CPU augmented with special instructions for posit arithmetic and a distinct posit register file (PRF). By allowing floating point and posits to coexist in an application, Clarinet uniquely allows researchers to make tradeoffs between hardware costs, latency, and precision. PERCIVAL, an application-level posit RISC-V core derived from CVA6 was introduced in [40]. This core can execute all posit instructions. Moreover, CLARINET proposed in [39] is a comprehensive framework for a GPP that is proposed to facilitate empirical studies in posit arithmetic.

In [41], a method for performing statistical fault injections (SFIs) on CNNs was proposed. In this method, the probability of a fault becoming a critical failure is measured based on the probability distribution of the golden data representation (the CNN weights).

B. Number Representation Systems

Different studies have shown that the number representation format has a considerable effect on energy consumption, performance, and fault tolerance of the systems. In [42], a residue number system (RNS)-based technique was proposed to decrease the energy consumption of DNNs. In [43], the challenge of high error rate in Multi-Level Cell (MLC) memories has been resolved by using the drop and rearrange (DARA) approach. In this method, the less significant bits are emptied and used as redundant bits for the most significant bits (MSBs). Also, by changing the bits' position, the more valuable bits are placed in safer positions in the memory.

Some state-of-the-art works studied the properties of the posit number representation system, e.g., the posit numbers accuracy and energy efficiency were compared to the other types of number representation systems in [25] and [44]. In [45], employing the posit arithmetic units in the DNN calculations, such as multiplying, accumulating, comparison, and other DNNs frequently used functions (such as sigmoid and hyperbolic functions), was assessed. Results show that using the posit arithmetic units offers a proper tradeoff between accuracy and processing time. In [44], employing posit number representation instead of the FLOAT in autonomous driving applications was studied. Results of this work show that a 12- to 14-bit posit number with zero exponent bits is an appropriate alternative for the FLOAT, without sacrificing the network accuracy. Also, 8- and 10-bit posit numbers provoke a slight accuracy degradation.

Deep PeNSieve is a framework proposed in [46], which executes the training and inference phases of DNNs based on the posit numbering system. Specifically, this work developed a CNN using P(32,2) and P(16,2) posit numbers. Also, it was shown that training a CNN with 16-bit posit numbers results in the same output accuracy as FLOAT, and the 8-bit posit can be used instead of 16-bit float or 8-bit integers during the inference. POSITNN [47] is an open-source framework that can use various bit widths for different steps (e.g., forward propagation, backward propagation, and optimization) of network training and inference phases. In this work, a network was trained using 8–12-bit widths without considerable accuracy degradation. Another general and open-source deep learning framework tool is Qtorch+, which allows the employment of new representation number systems on Pytorch [48].

Several recent works have developed the required hardware for composing the posit number representation-based systems [24], [49], [50]. In [49], an open-source tool (PACoGen) for generating Verilog HDL of posit-based arithmetic units, such as adders, subtractors, multipliers, and dividers, was introduced. ExPAN(N)D proposed in [50] presents a framework for analyzing the efficacy of posit number representation for artificial neural networks (ANNs). However, there is not a comprehensive investigation on the fault tolerance of posit-based DNNs, whereas discussed in Section I-A, the posit number representation system can offer a considerable fault

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

tolerance improvement in the DNNs hardware accelerators deployed in the safety-critical applications.

Unlike the prior works, e.g., [26] that only considered a 32-bit MLP, in this work, we proposed a systematic method for composing fault-tolerant posit-based DNNs. Moreover, we performed different model/layer/bitwise explorations to extract the fault-tolerant characteristics of the posit-based networks. We also compared the fault tolerance features of the posit-based networks with the various formats of the fixed-point and FLOAT-based models. Then, we proposed a methodology to select the proper bit width of the different fields of the posit number from the fault tolerance point of view. Moreover, besides a solid tool flow for fault-tolerant evaluations of the posit-based DNNs, we also proposed an enhanced fault-tolerance scheme to compose posit-based DNNs, achieved by analyzing the behavior of the DNN models against fault.

III. Preliminaries

In this section, first, the structure of conventional DNN accelerators is studied. Also, the types of errors that may occur in DNN hardware accelerators have been detailed. Then, the posit number representation system is presented.

A. Architecture of Conventional DNN Accelerators

Recently, much attention has been devoted to introduce different types of DNN accelerators to achieve higher energy efficiency and performance [6]. Two popular architectures of the DNN accelerators are vector-based [5] and systolic array-based [4] accelerators. One of the most important components in these accelerators is the memory unit keeping the weight values of the DNN. From the fault-tolerant perspective, a fault may occur in the different parts of these accelerators, e.g., in weight memory or the processing elements (PEs). However, the weight memory's significance is due to the reuse of its content in different computations, where a generated error may be propagated through the PEs [7]. Thus, similar to some of the state-of-the-art works (e.g., [27] and [30]), our focus in this work is on the faults that may occur in the weight memory.

Errors in accelerators implementing the DNN models can occur due to various factors, which may affect their normal operation, e.g., timing constraints of an accelerator may be violated by soft errors, aging phenomena, environmental conditions, such as noise, and process variation [7]. In some applications, such as aerospace, high-energy particles induced by the sun or cosmic particles in space may result in faults in electronic devices [7]. Radioactive impurities in device packages and electromagnetic radiations are the other sources of faults in such systems. These challenges become more critical with the technology scaling in the nanoscale era, where the charge threshold required to flip the bits is decreased [9]. Therefore, it is essential to use fault-tolerant methods to robust the DNN accelerators implementing safety-critical applications.

B. Posit Number Representation System

In Fig. 3, the posit number representation schema has been shown, where (1) exhibits the calculation pattern of a posit

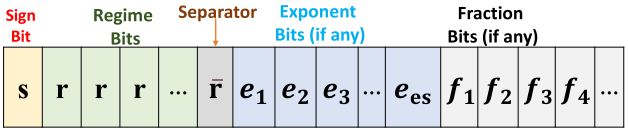

Fig. 3. Posit format for finite and nonzero values.

TABLE I
Counting Consecutive Zeros or Ones in the Regime Field to Calculate k in (1)

Binary	0000	0001	001x	01xx	10xx	110x	1110	1111
Numerical meaning, k	-4	-3	-2	-1	0	1	2	3

number (X). Each posit number is determined by a P(n, es) pair, in which n and es are the posit number's bit width and the maximum bit width of the exponent field, respectively. Also, the MSB bit is the sign bit

$$X = \begin{cases} 0, & 000\ldots0000 \\ \pm\infty & ,100\ldots0000, \\ \text{sign(p)} \times \text{useed}^k \times 2^e \times f & \\ \text{useed} = 2^{2^{es}}, & f = 1.f_1 f_2 f_{3\ldots} f_{f_s}. \end{cases} \tag{1}$$

For a negative number, first, the twos complement of that number is calculated before retrieving k, es, and f in (1). Then, in the decoding step, the value of k is determined based on regime bits. The number of identical continuous bits (a chain of zeros or ones) after the sign bit is m. If these bits are zero, then k is equal to $-m$. Otherwise, k would be $m-1$ (see Table I). Then, the es value is considered as the number of exponent bits (if any), and the rest of the LSB bits are the fraction bits (if any). As an example, 0000110111011101 is a 16-bit posit number with a 3-bit exponent [i.e., P(16,3)]. The sign bit is zero, i.e., the number is positive and the regime bits are 000. After the first one value bit [i.e., the separator bit (r) in (1)], the next three bits are exponent bits (i.e., 101). Subsequently, the rest bits are fraction bits (i.e., 11011101).

IV. Fault Tolerance Analysis of Posit-Based DNN Models

In this section, the simulation setup and tool flow developed for our explorations as well as the fault injection method are presented. Based on these, the fault tolerance analysis of posit, fixed-point, and FLOAT-based DNN models is performed and compared with each other. One of the main targets of this analysis is to achieve the most fault-tolerant $P(n, es)$ pairs, in each of the three aforementioned model, layer, and bitwise levels. The system overview diagram of the proposed posit-based fault-tolerant analysis of DNN models is depicted in Fig. 4, including inputs, DNN model training in posit format, developed fault injection tool flow, model inference in posit format, fault tolerance explorations, and DNN output accuracy evaluation steps.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

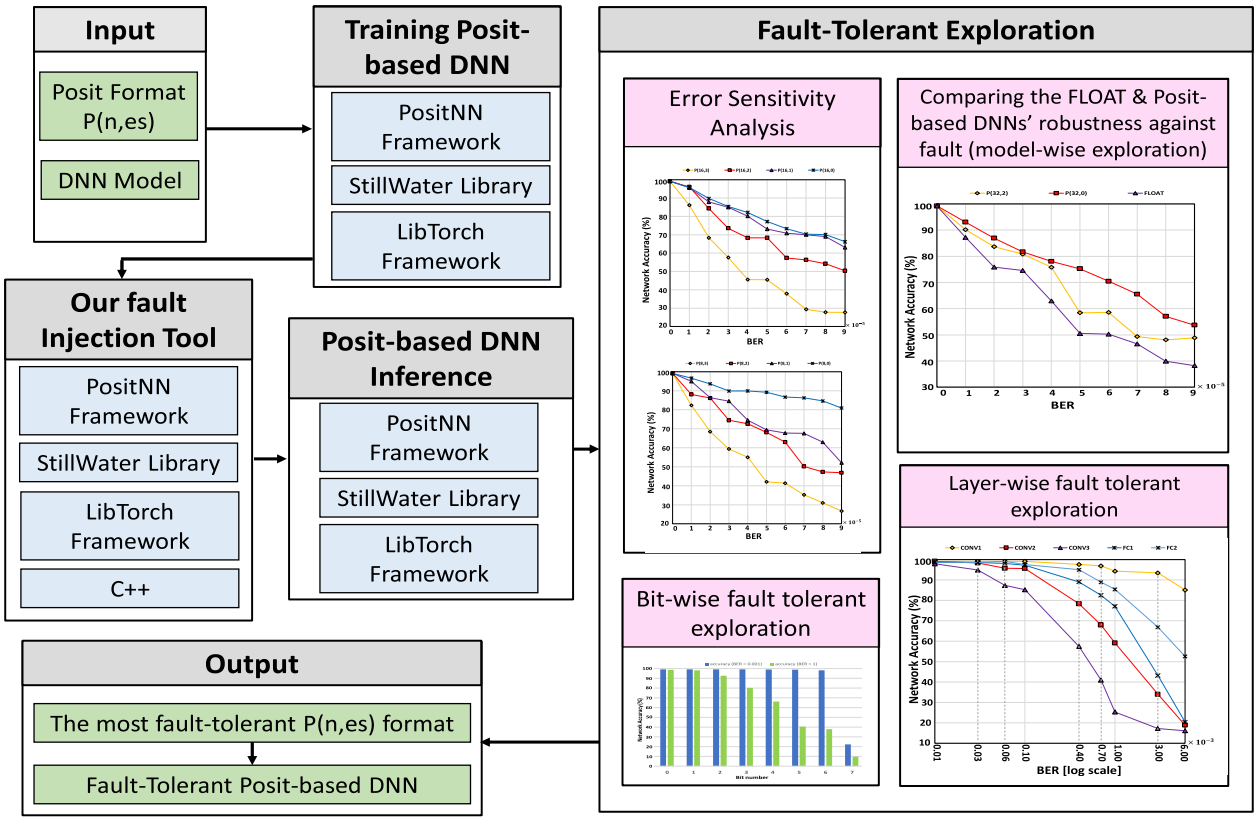

Fig. 4. System overview diagram of the fault tolerance exploration of posit-based DNN models.

A. Simulation Setup

For a comprehensive exploration, the training was performed with various values of n and es of the posit numbers. In our simulations, posit and FLOAT-based LeNet-5 networks were trained using the PositNN framework and LibTorch (PyTorch C++ API). Moreover, FLOAT, fixed-point, and posit-based ResNet-50 networks were trained with CIFAR-10 datasets. Also, we developed the fault injection tool using LibTorch and the StillWater framework. Testing the models was also carried out on an Ubuntu 20.04 operating system running on a system with 12-core Intel Xeon[1] CPU ES-2690 v3 at 2.60 GHz and 20 GB of RAM.

B. Fault Injection Flow

Ares [7] is a fault injection framework for FLOAT-based DNNs. In Ares, two fault injection modes are considered, static and dynamic. Ares performs fault injection in the DNN weights in the static mode, that is, before the DNN inference phase, a fault injection pattern is applied in the weights, and after that, the DNN enters the inference phase with faulty weights. Ares uses bit-level fault injection to change each weight value. It is worth mentioning that this fault pattern is based on a uniform distribution. In our work, fault injection has been done in DNN weights based on Ares. In the following, the fault injection method will be discussed in more detail.

Inspired from [7], in our developed fault injection method, faults are injected into the weight values of a trained DNN model under the various BERs, where the BERs correspond to the number of soft errors that may affect the weight values. Therefore, the BER is the knob that provides scalability of the fault injection method, whereas in our work, the BER is varied in the range of $[1 \times 10^{-7}, 9 \times 10^{-5}]$. As the BER increases, the number of erroneous bits among the model's weight values increases, and therefore, the accuracy of the network begins to decrease. More precisely, each bit of the model's weights values is flipped based on the considered BER, independently. In detail, similar to [9], corresponding to each bit of a given weight value, a uniform random number is generated between 0 and 1. Then, if the generated number is less or equal to the considered error probability (i.e., BER), the corresponding bit is flipped. As an example, for an 8-bit weight of the DNN model under the BER $= 1 \times 10^{-5}$, eight random values are generated between 0 and 1 and assigned to each bit of that weight. Now, those bits whose corresponding random value is less than 1×10^{-5} are flipped. Finally, depending on the considered BER, different numbers of the bits of the model's weight may be erroneous (flipped).

We developed an in-house fault-injection tool to explore the fault tolerance of the different posit-based DNN models, under occurring errors in the weights' value. The developed fault injection tool flow is shown in Fig. 5. In this flow, first, faults are injected on a pretrained DNN model. Note that similar to [7] and [27], the errors are modeled by flipping the bits of a weight value with a uniform random distribution, where the number of the flipped bits is different for the various BERs. Then, the inference phase of the model is performed with these faulty weights. Note that each experiment (fault injection in the weight value) was replicated 50 times under a given BER,

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

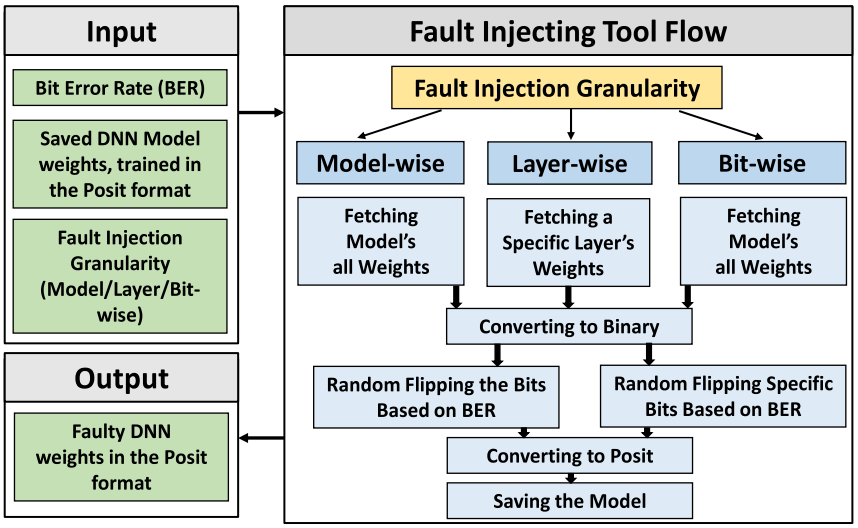

Fig. 5. Proposed fault injection flow (i.e., the detailed flow of the "the developed fault injection tool" block shown in Fig. 4) to evaluate the fault tolerance of posit-based DNNs in the three model, layer, and bit levels.

and the average output accuracy was reported. As discussed in Section I, the fault injection is carried out in the three levels, each serving a specific purpose.

In the first level, faults are injected across all DNN weights to compare the resilience of FLOAT and posit formats against faults. The selected posit formats in our work [i.e., P(32,0), P(32,2), P(16,0), P(16,1), P(16,2), P(16,3), P(14,0), P(12,0), P(12,1), P(10,0), P(10,1), P(8,0), P(8,1), P(8,2), P(8,3), P(6,1), and P(6,2)] are based on the best accuracies obtained in the training and inference of DNNs reported in the state-of-the-art works [29], [46], [47]. Moreover, for a fair comparison, we compared the 32-bit posit with the FLOAT. Then, we determine the most appropriate posit format in terms of fault tolerance, which offers the highest accuracy in the presence of fault.

In the second level, faults are injected into a given layer of the DNN, and then, the accuracy of the network is measured during inference to measure the sensitivity of different posit-based DNN layers against faults. This step is repeated for all layers of the model, where the results are used to compare the sensitivity of the layers in posit and FLOAT systems in the presence of faults. This also provides a useful intuition for employing the fault tolerance methods, where the more sensitive layers are prioritized for applying the redundancy techniques.

Finally, in the third level, the sensitivity of all bits of the weights' value (in the posit format) is measured in the faulty DNN models. This evaluation is performed by flipping a specific bit of the posit number. This step also determines the importance of the different bits of the posit values in terms of the fault tolerance, i.e., injecting faults in the more sensitive bits will have a greater effect in reducing the accuracy of the DNN. Thus, the redundancy techniques can be employed only in the more important bits.

C. Modelwise Fault Tolerance Analysis of Posit-Based DNNs

To explore the fault tolerance of the posit-based DNN models, we selected the LeNet-5. We also examined the ResNet-50 network, which is a much more complex DNN than the LeNet-5 network. This examination performed in posit, fixed-point, FLOAT16, and FLOAT formats. The LeNet-5 architecture is composed of five layers, including CONV1, CONV2, and CONV3 as the convolution layers, and the two fully connected (FC) layers (i.e., FC1 and FC2). This model was trained under the MNIST dataset and using the simulation setup discussed in Section IV-A. Afterward, based on the fault injection flow discussed in Section IV-B, faults were injected into all weights of the trained LeNet-5 composed of 32-bit posit numbers. In this case, posit's exponent was considered 2 [i.e., P(32,2)] based on the Posit 2022 standard [51]. Also, P(32,0) was considered for more examinations. For a fair comparison, the FLOAT-based LeNet-5 was examined under the same conditions, where the FLOAT has widespread usage in today's computer systems [44].

Fig. 6(a) depicts the LeNet-5 accuracy for various BERs, where faults are injected throughout the DNN model with a uniform random error distribution. As shown in this figure, classification accuracy is decreased proportionally to the BERs increase. However, the accuracy of the investigated models drops suddenly from a given point onward [e.g., see pointer ⑦ in Fig. 6(a)], because the error rate has reached to a point, where more crucial locations of the DNN were affected by the errors. In this examination, the FLOAT-based model has a sharp accuracy drop around BER $= 4 \times 10^{-5}$ [see pointer ⑤ in Fig. 6(a)], whereas the accuracy of the P(32,2) and P(32,0)-based models drops suddenly in higher BERs [see pointers ⑥ and ⑦ in Fig. 6(a)]. In detail, where the BER is 7×10^{-5}, P(32,0)-based model has an accuracy of more than 65% [see pointer ⑦ in Fig. 6(a)], while the FLOAT-based model has an accuracy of less than 50% [see pointer ⑤ in Fig. 6(a)]. Also, the results show that the P(32,0)-based model has a higher accuracy under the same BERs than the P(32,2), e.g., at the BER of 4×10^{-5}, P(32,0) has an accuracy of nearly 80%, whereas the accuracy of the P(32,2)-based model is about 75%, and the accuracy of the FLOAT-based model has decreased to 60% at the same BER. In general, the accuracy

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

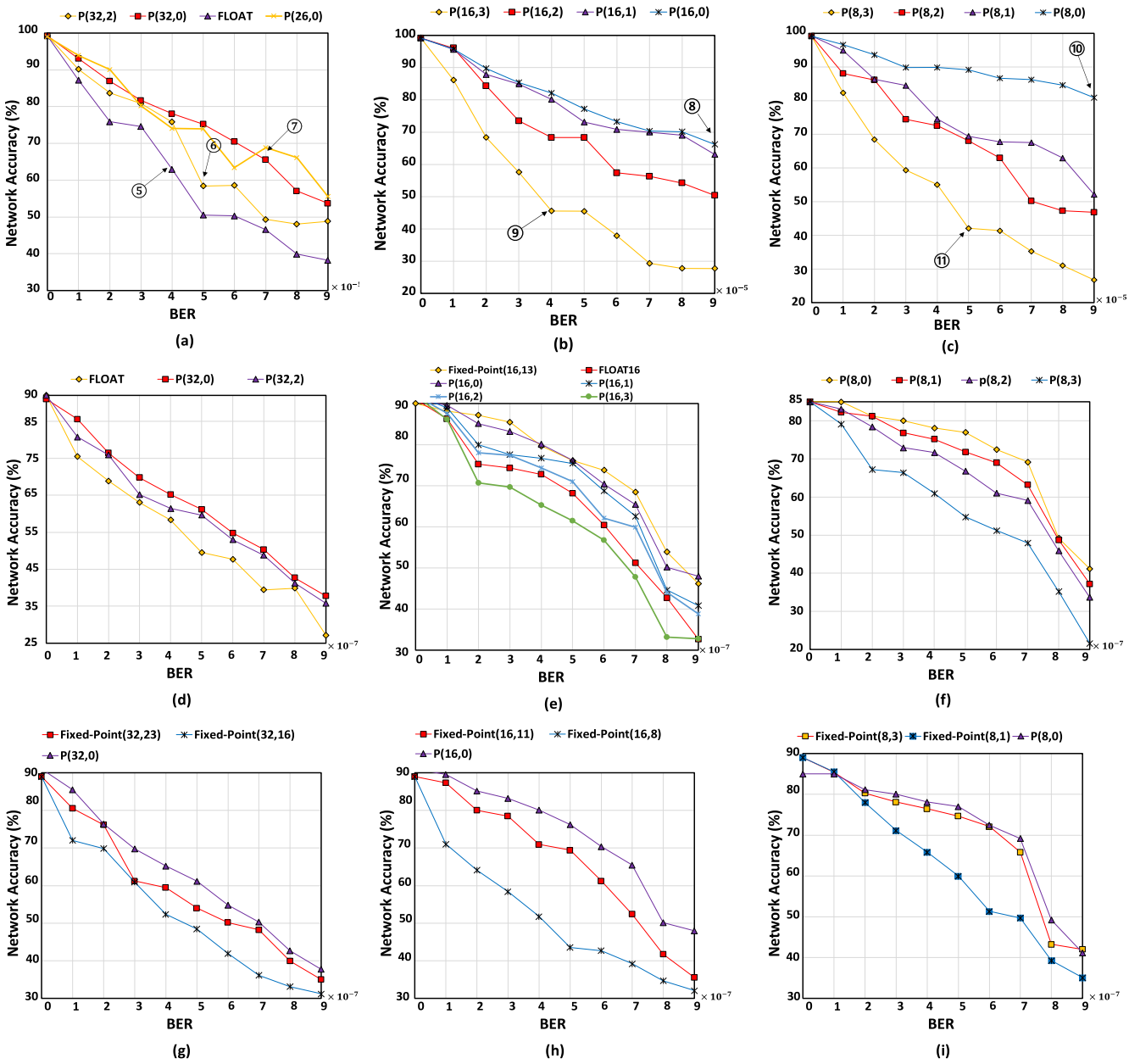

Fig. 6. Classification accuracy of the posit-based LeNet-5 (a)–(c) and ResNet-50 (d)–(i) networks trained with the MNIST and CIFAR-10 datasets, respectively, for the different values of the exponent, and under the various BERs for (a), (d), and (g) 32-bit posit, fixed-point, and FLOAT-based models, (b), (e), and (h) 16-bit fixed-point and posit-based models, (c), (f), and (i) 8-bit fixed-point and posit-based models, and (e) 16-bit posit, fixed-point, and float 16-based models. fixed-point numbers are represented by the fixed point (wl, fl), where wl and fl reference the word length and fractional length, respectively.

of the P(32,0) [P(32,2)] posit model is on average 15% (7%) higher than the FLOAT-based model. Therefore, one may conclude that the P(32,2)-based model is less sensitive to faults than FLOAT, where this verity is more prominent in the P(32,0).

1) Determining the Most Fault-Tolerant Posit Format for the Posit-Based DNN Models: To determine the most fault-tolerant P(n, es) format, various values of n and es are examined. The classification accuracy results of 16 and 8-bit posit-based models under the different BERs are presented in Fig. 6(b) and (c), respectively. Based on the results presented in Fig. 6(c), for the P(8,3), P(8,2), P(8,1), and P(8,0) formats, the investigated model achieved, on average, 49%, 66%, 73%, and 89% classification accuracy, under the different studied BERs. Specifically, for the 8-bit posit-based model, the P(8,0)-based systems demonstrate the highest fault tolerance, where the network does not suffer a sharp drop in the accuracy even at high BERs [see pointer ⑩ in Fig. 6(c)]. In contrast, P(8,3) resulted in the lowest accuracy [see pointer ⑪ in Fig. 6(b)].

For the 16-bit posit-based model, based on the results presented in Fig. 6(b), P(16,3), P(16,2), P(16,1), and P(16,0) formats were achieved to, on average, 47%, 68%, 77%, and 79% classification accuracy, under the various investigated BERs. In detail, the P(16,0)-based model archived the highest level of fault tolerance [see pointer ⑧ in Fig. 6(b)], and P(16,3) has the lowest one [see pointer ⑨ in Fig. 6(b)]. As the results of

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

Fig. 6(b) and (c) reveal, the amount of fault tolerance decreases with growing the number of exponent bits for the explored P(n, es) pairs. Moreover, for the different values of the n and es, the P(16,0) and P(8,0)-based models show the highest classification accuracy under different BERs.

In the following, this behavior is analyzed mathematically for the posit-based DNN models.

2) Mathematical Fault Tolerance Analysis of Posit-Based DNN Models: A positive binary number in the posit number system achieves its maximum value when all $n-1$ bits after the sign bit are set to one. Therefore, there is no bit left for representing e and f, where based on (1), their value will be zero and one, respectively. Additionally, the value of k will be $n-2$. Substituting these values of e and k in (1) yields

$$P(n, es) = (-1)^s \times \left(2^{2^{es}}\right)^{n-2}. \tag{2}$$

In (2), for a constant value of n, increasing es will exponentially increase the overall value of the number, expanding the range of generated values. Similarly, this range expansion for negative numbers can also be proved. This behavior demonstrates how increasing es can be inversely related to fault tolerance of posit-based DNNs, wherein the following, by comparing P(8,0) and P(8,3), this issue becomes even more apparent.

Fig. 7(a) and (c) shows the weights histogram of the P(8,0)- and P(8,3)-based LeNet-5 network trained with the MNIST dataset, respectively. Moreover, the faulty weights histograms of these models were shown in Fig. 7(b) and (d), respectively, under the BER $= 9 \times 10^{-5}$. As shown in these figures, with injecting errors in the P(8,3)-based model, the value of the faulty weights is considerably higher than the P(8,0)-based model [see pointers ⑬ and ⑭ in Fig. 7(b) and ⑮ and ⑯ Fig. 7(d)]. Thus, for the P(8,3)-based model, the accuracy of the network will be severely affected even if errors are occurred at the weights with a lower significance. Hence, the amount of error will have a much greater influence on the output accuracy of the network. In the following, this observation is explained mathematically.

Equations (3) and (4) present P(8,3) and P(8,0) obtained by the posit formula presented in (2)

$$\mathrm{P}(8, 3) = (-1)^s \times 2^{8\times 6} = (-1)^s \times 2^{48} \tag{3}$$

$$\mathrm{P}(8, 0) = (-1)^s \times 2^{1\times 6} = (-1)^s \times 2^{6}. \tag{4}$$

In (3), the generated numbers are within the range from -2^{48} to $+2^{48}$. Thus, any error in the MSBs of the generated number may result in a high amplitude error that can considerably decrease the network accuracy. However, for P(8,0) [i.e., (4)], the generated numbers are within the range from -64 to 64.

In the CNNs, the value of the weights is in the range of $[-1, 1]$ [52], and thus, multiplying data on a faulty weight value that has a significantly higher amplitude, such as the mentioned faulty P(8,3) number, may result in a considerable error magnitude, when compared to a faulty posit number in P(8,0). Thus, in the posit format, by increasing the number of exponent bits, the coefficient of k is increased [see (1)], which results in a high magnitude of error.

The mean square error (mse) of the two erroneous P(8,0)- and P(8,3)-based models shown in Fig. 7(b) and (d), respectively, is calculated by [53]

$$\mathrm{mse} = \frac{1}{n}\sum_{i=1}^{n}(W_{\mathrm{H}i} - W_{\mathrm{F}i})^2 \tag{5}$$

where $W_{\mathrm{H}i}$ and $W_{\mathrm{F}i}$ are the healthy and faulty weight values, respectively. For BER $= 9 \times 10^{-5}$, the mse values of the P(8,0)- and P(8,3)-based models are 0.33 and 4.56×10^6, respectively, which shows that the errors in the weights of the P(8,3)-based model can decrease the network accuracy drastically. Based on the above discussions, one may consider $es = 0$ since it offers a higher fault tolerance. The classification accuracy of the posit-based models, for the various values of n and $es = 0$ [i.e., P(n,0)], is shown in Fig. 8. Based on the results, for the different studied ranges of the BER, P(32,0)-P(16,0)-, P(14,0)-, P(12,0)-, P(10,0)-, and P(8,0)-based models achieved, on average, 73%, 79%, 73%, 74%, 74%, and 88%, respectively, classification accuracy.

Specifically, the P(8,0)-based model provides the most error resiliency even at the highest BER, where its accuracy remained above 80% (see pointer ⑫ in Fig. 8). Finally, to achieve a general insight, the average network accuracy of the different studied posit and FLOAT-based LeNet5 networks under a BER ranging from 1×10^{-5} to 9×10^{-5} is shown in Table II. As the results show, the most fault tolerance is provided by the P(8,0)-based model. The results for the ResNet-50 network trained with CIFAR10 are depicted in Fig. 6(d)–(i). As shown in this figure, the posit number representation system provides better fault tolerance compared to FLOAT. Furthermore, under the same BERs, the posit-based network exhibits 8% more accuracy than the FLOAT-based network. Also, for the 16 bit, the accuracy of posit is very close to fixed point. As shown in Fig. 6(f), the accuracy of the network decreases with increasing the exponent value in P(n, es). Based on the results shown in Fig. 6(g)–(i), for the different studied BERs, P(32,0), P(16,0), and P(8,0) achieve, on average, a higher accuracy of (11%, 4%), (8%, 23%), and (2%,11%), compared to [fixed point(32,16), fixed point(32,23)], [fixed point(16,11), fixed point(16,8)], and [fixed point(8,3), fixed point(8,1)], respectively.

These improvements were achieved, while the posit numbers can also represent a broader range of numbers compared to the fixed point. As shown in Fig. 6(g)–(i), the error tolerance of fixed-point-based models decreases as the fractional bit length decreases, which creates a tradeoff between the range of representable numbers and the level of error tolerance.

D. Layerwise Fault Tolerance Analysis of Posit-Based DNNs

In this section, the aforementioned per-layer fault injection scenario is performed to evaluate the error sensitivity of each DNN's layer when their weight values are faulty and study the impact of each erroneous layer on the output accuracy.

Using the results of this step, more sensitive layers can be prioritized to protect against faults, i.e., one may use the redundancy techniques only for the layers with a higher error sensitivity.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

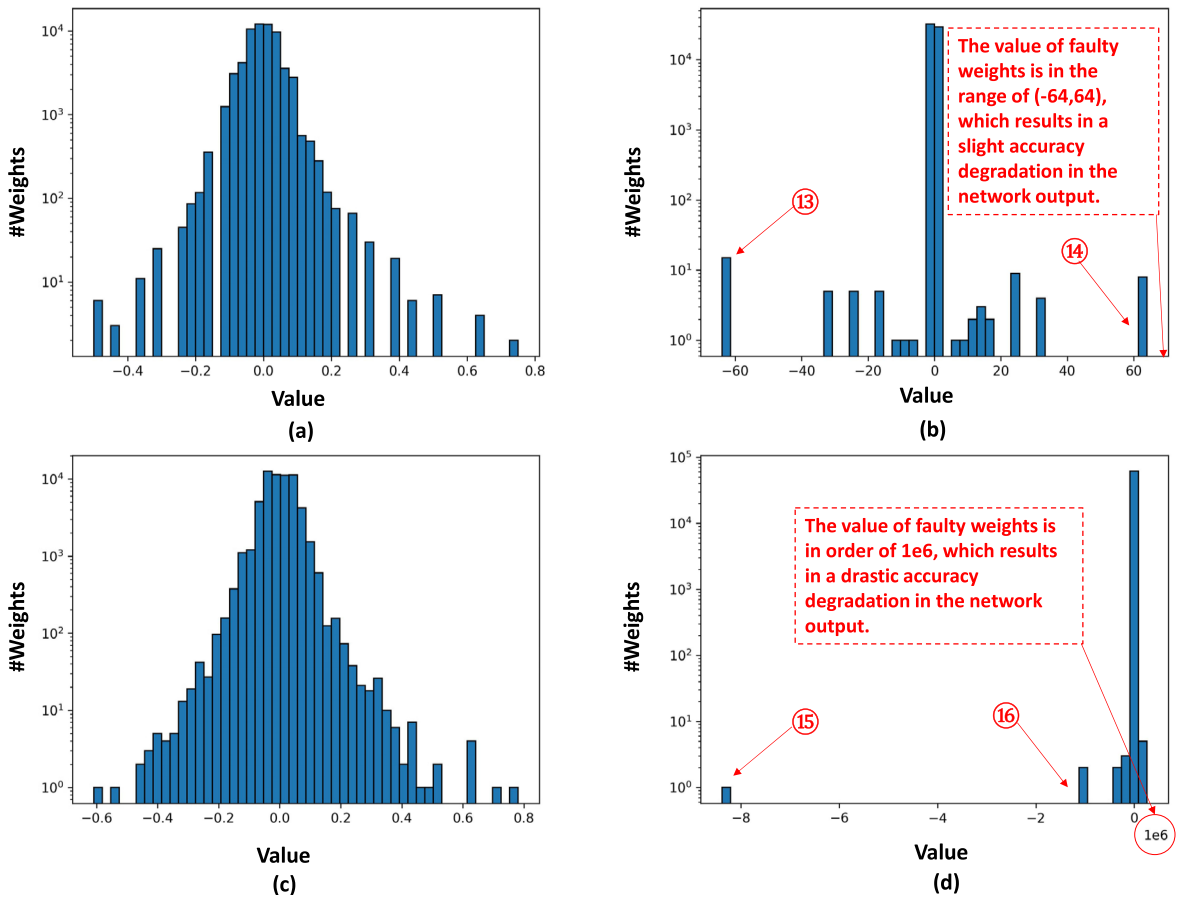

Fig. 7. Histogram of the #Weights of the posit-based LeNet-5 network trained with MNIST dataset, for (a) P(8,0) and (b) P(8,3). Histogram of the faulty weights of (c) P(8,0) and (d) P(8,3)-based models, under the BER of 9×10^{-5}.

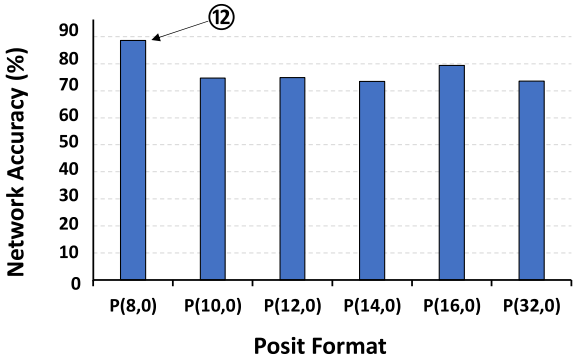

Fig. 8. Classification accuracy of the posit-based models for the various values of n when $es = 0$ [i.e., P(n,0)].

TABLE II

AVERAGE AND VARIANCE OF THE NETWORK ACCURACY FOR THE DIFFERENT STUDIED POSIT AND FLOAT-BASED LENET5 NETWORKS, UNDER THE BER RANGING FROM 1×10^{-5} TO 9×10^{-5}

Number Representation System	Accuracy		Number Representation System	Accuracy	
	average	variance		average	variance
FLOAT	0.5845	0.030	**P(12,1)**	0.7449	0.009
P(32,0)	0.7352	0.017	**P(10,0)**	0.7470	0.014
P(32,2)	0.6597	0.027	**P(10,1)**	0.7327	0.015
P(16,0)	0.7935	0.009	**P(8,0)**	0.8861	0.002
P(16,1)	0.7716	0.011	**P(8,1)**	0.7335	0.017
P(16,2)	0.6763	0.022	**P(8,2)**	0.6628	0.024
P(16,3)	0.4730	0.040	**P(8,3)**	0.4907	0.034
P(14,0)	0.7344	0.019	**P(6,1)**	0.7718	0.012
P(12,0)	0.7478	0.014	**P(6,2)**	0.6950	0.023

1) Fault Tolerance of the Posit-Based LeNet-5: Fig. 9 depicts the classification accuracy of the LeNet-5 network trained under the MNIST dataset when each of its layers is faulty. Based on the results, under the different studied BERs, the model's accuracy drops, on average, 95%, 71%, 58%, 78%, and 86% for the CONV1, CONV2, CONV3, FC1, and FC2 layers, respectively. In detail, among the layers, CONV3 is much more sensitive to error, where the model accuracy drops much faster than the other layers below 60% for the 4×10^{-4} BER (see pointer ㉑ Fig. 9). The second most sensitive layer is the CONV2, wherein the BER of 1×10^{-3}, the accuracy drops below the 60% (see pointer ⑳ in Fig. 9). In contrast, fault injection in the CONV1 layer has the lowest impact on the model's accuracy, where the accuracy of the network remained above 80% at the BER of 6×10^{-3} (see pointer ⑰ in Fig. 9). Afterward, FC1 and FC2 layers have a higher resiliency, respectively, compared to the other layers. When faults are injected into FC1 with the BER of 3×10^{-3}, the

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

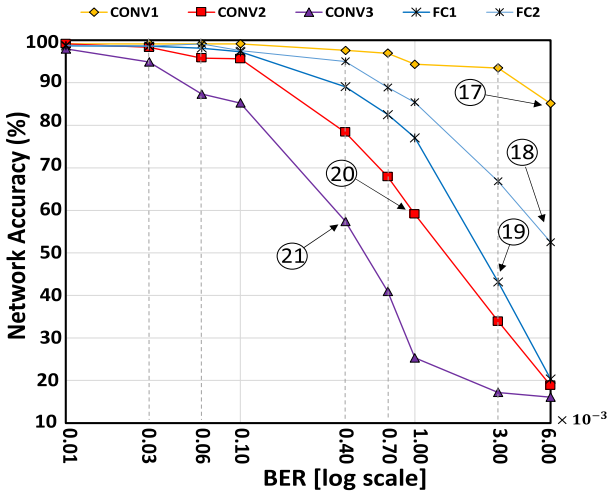

Fig. 9. Classification accuracy of the P(8,0)-based LeNet-5 network trained with MNIST dataset, under the various BERs, where the faults are injected on the layers.

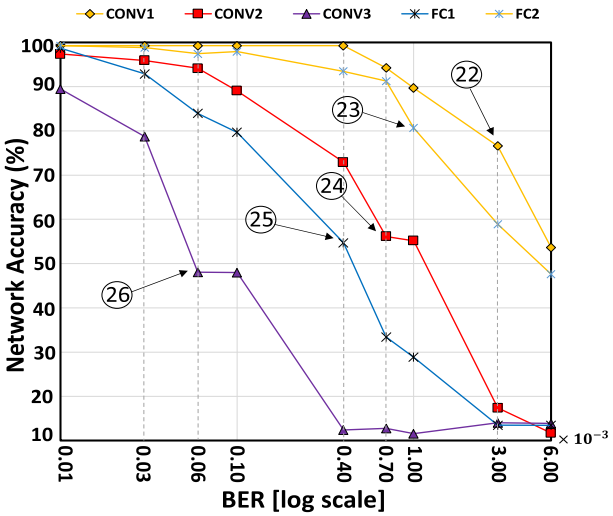

Fig. 10. Classification accuracy of the FLOAT-based LeNet-5 network trained with MNIST dataset, under the various BERs, where the faults are injected on the layers.

accuracy is greatly reduced (see pointer ⑲ in Fig. 9), whereas the FC2 layer is more resistant than FC1, and under the BER of 6×10^{-3}, the accuracy decreases to 52% (see pointer ⑱ in Fig. 9).

2) Fault Tolerance of the FLOAT-Based Lenet-5: Fig. 10 depicts the classification accuracy of the FLOAT-based lenet-5 when the faults are injected in each layer of the model, under the various BERs. Based on the results, under the different studied BERs, the accuracy of the FLOAT-based model drops, on average, 90%, 65%, 36%, 55%, and 85% for the CONV1, CONV2, CONV3, FC1, and FC2 layers, respectively. Specifically, the faulty CONV1 has almost the lowest impact on the output accuracy compared to the other layers, where under the 3×10^{-3} BER, the model's accuracy is 76% (see pointer ㉒ in Fig. 10). Under the fault injection in the FC2 layer with the BER of 1×10^{-3}, the accuracy of the network is about 80% (see pointer ㉓ in Fig. 10). Moreover, injecting faults with a BER of 7×10^{-4} in the CONV2 layer

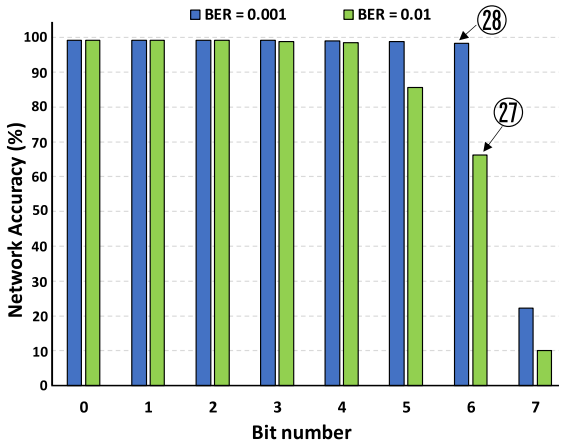

Fig. 11. Bitwise fault tolerance analysis of the P(8,0)-based LeNet network trained with the MNIST dataset, under the two BERs of 0.001 and 0.01.

drops the accuracy of the network below 60% (see pointer ㉔ in Fig. 10). By injecting the faults with a BER of 4×10^{-4} in the FC1 layer, the accuracy of the network decreases to 54%, while at the same BER in the CONV2 layer, the accuracy of the network is more than 72%. Thus, the CONV2 layer has a lower error sensitivity than the FC1 layer (see pointers ㉔ and ㉕ in Fig. 10). Finally, the CONV3 layer has the lowest fault tolerance, whereby injecting the faults with the BER of 6×10^{-5}, the accuracy of the network drops below 50% (see pointer ㉖ in Fig. 10).

The comparison between the results presented in Fig. 9 (i.e., posit-based model) and Fig. 10 (i.e., FLOAT-based model) reveals that the behavior of layers in posit and FLOAT formats is different in terms of fault tolerance. As an example, in the posit format, the FC1 layer has a higher level of fault tolerance than CONV2, whereas this is vice versa in the FLOAT-based model. Therefore, the fault-tolerance investigation of the posit-based DNN models should be performed independently of the results achieved by the FLOAT-based DNN fault-tolerance studies (e.g., see [7] and [10]).

E. Bitwise Fault Tolerance Analysis of Posit-Based DNNs

In this section, to evaluate the bits' error sensitivity of the posit-based DNN models, we perform a bitwise fault tolerance exploration. In detail, we explore the error resiliency of each bit of a posit number individually to determine the effect of each field of the posit number on the network accuracy. Fig. 11 presents the classification accuracy of the P(8,0)-based LeNet-5 network trained using the MNIST dataset, when faults are injected on a dedicated bit. In [27], it is shown that in the FLOAT-based systems, when the errors appeared in the MSBs of the exponent field, the accuracy of the network decreases significantly, whereas the error in the other bits has a lower impact on the accuracy. However, the results presented in Fig. 11 show that in the posit-based network, the sign bit is the most sensitive one to the error. In Fig. 11, it is demonstrated how injecting errors to each bit of the weights in a P(8,0)-based LeNet-5 network (i.e., bits 0–7) impacts the accuracy of the network. Note that when the BER is 0.01, the selected bit [one of those 8 bits in the P(8,0) format] is flipped with a higher

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

probability in all weights of the trained network compared to the BER = 0.001, and thus, the accuracy of the DNN is dropped faster than when the BER is 0.001. As an example, for BER = 0.01, the accuracy of the studied DNN is fallen below 80% from the 6rd bit (see pointer ㉗ in Fig. 11), while for BER = 0.001, the accuracy of the model is still higher than the 95% even by injecting fault in 6th bit (see pointer ㉘ in Fig. 11). In the case of BER = 0.01, the errors in bits 0–4 have a very negligible effect on the network accuracy (less than 1%).

However, by increasing the bit position number, the accuracy is dropped more, where by injecting the errors in bit 7, the accuracy of the network is dropped drastically. For BER $= 1 \times 10^{-3}$, the errors in bits 0–6 resulted in, on average, 1% degradation in the output accuracy, whereas the error in bit 7 reduces the accuracy of the network by 76%. Based on these results, one may leverage conventional redundancy techniques to protect the bits with a higher impact on the network accuracy. As an example, in [54], the selective latch hardening (SLH) method has been used to protect the most important bits of the FLOAT-based DNN models. However, this comes with certain overheads. To address this issue, in Section IV-F, we proposed an enhanced fault-tolerant posit-based DNN model, without requiring the considerable overhead associated with conventional redundancy-based techniques (such as TMR).

F. Enhanced Fault-Tolerant Posit-Based DNN Model

In fault tolerance literature, there are methods known as value-aware parity insertion techniques for error detection and correction in DNNs. Weight nulling [55] employs the least significant bit (LSB) of each weight as a singular parity bit, substituting incorrect weight values with zero when the parity bit identifies an uneven count of bit errors within each weight. Since the LSB of each weight is designated as a parity bit, it may cause modifications to its numerical value, which could compromise the accuracy of the network. Such modifications can diminish the DNN's accuracy. The in-place zero-space ECC method proposed in [56] ensures that weight values are retained within a specific range by employing modified training techniques.

This method may cause computational overhead due to changes in DNN training. In [57], a value-aware parity insertion ECC was proposed for DNNs. This approach enhances the error correction capability from the single to the double error correction one, while simultaneously reducing parity storage overhead and eliminating the necessity for additional training processes. This methodology is specifically designed for 8-bit weights, which may incur significant computational overhead for higher bit widths. Concurrently, this method involves two stages of conversion between two's complement and sign-magnitude representations, which will also contribute to performance overhead. Furthermore, this method relies on the bit pattern of numbers in floating point representation, a concept not applicable to posit. For instance, the number −0.125 is represented by 10001000 in an 8-bit floating point, where the proposed method operates based on the equivalence

TABLE III
SAMPLE NUMBERS IN THE P(8,0) FORMAT, WITH A VALUE BETWEEN −1 AND +1

P(8,0)	Decimal Value
00111010	+0.906250
00100110	+0.593750
00010011	+0.296875
00000000	0
11101101	-0.296875
11011010	-0.593750
11000110	-0.906250

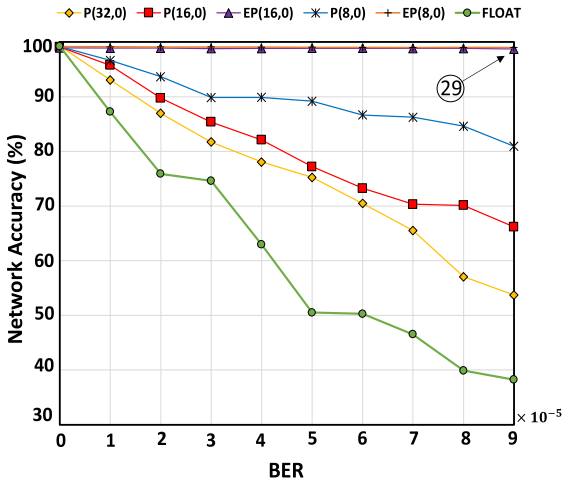

Fig. 12. Classification accuracy of the FLOAT, P(8,0), P(16,0), P(32,0), and the proposed enhanced P(16,0) [i.e., EP(16,0)] and P(8,0) [i.e., EP(8,0)]-based LeNet-5 networks trained with MNIST dataset.

of bits 5 and 6. However, the value of −0.125 is represented as 11011000 in the 8-bit posit representation, indicating that the 5th and 6th bits are not equivalent. Thus, it is unachievable to utilize the method suggested in [57] for 8-bit posit numbers. To overcome this challenge, in the flowing, we propose a method that operates based on the intrinsic features of the posit formats.

In the posit numbering system, the representation of numbers between −1 and +1 is such that the two MSB bits of the numbers are the same [50]. As an example, in a P(8,0) format, bits 7 and 6 are either one or both are zero. Some example numbers of this feature are shown in Table III.

As mentioned in Section IV-C, the weights of CNNs are between −1 and +1 due to the normalization [52], where this feature can be employed to detect errors in the posit-based networks. Thus, if the value of the first two MSB bits of a weight is not the same, it is inferred that an error has occurred in bits 6 or 7, i.e., the error is detected in the two MSBs. Moreover, in Fig. 11, it is shown that these two bits are the most important ones from the fault tolerance perspective. Now, when an error in these bits is detected, to mitigate the impact of the faulty weight in the model calculations, and consequently increase the network accuracy, that weight value is set to zero [30]. Fig. 12 shows the classification accuracy of the LeNet-5 network for P(8,0), P(16,0), P(32,0), FLOAT, and the proposed enhanced P(8,0) [i.e., EP(8,0)] and P(16,0) [i.e., EP(16,0)]-based models. Based on the results,

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

applying the proposed fault-tolerance enhancement method in the posit-based network results in, on average, 41% and 15% higher classification accuracy compared to the FLOAT and unprotected P(8,0)-based models, respectively, under the different studied BERs (see pointer ㉙ in Fig. 12).

V. Conclusion

In this article, we explored employing the posit number representation systems in DNN models from the fault tolerance perspective. By proposing a systematic framework and developing an in-house solid tool flow to analyze the fault tolerance of the posit-based DNN models, a comprehensive fault-tolerant exploration of the posit-based DNNs was performed in the three model, layer, and bitwise manners to achieve the most appropriate format of the posit numbers to compose the DNNs at each of ways. We have shown that the P(8,0) format is an appropriate posit format for the inference phase of DNNS. Also, compared to the FLOAT-based DNN models, the posit numbering representation system showed a superior and different fault-tolerance behaviors. Results show that a P(32,0)-based model can achieve up to 15% higher classification accuracy, compared to the FLOAT-based model, under the same BER. Moreover, the posit-based DNNs showed up to 23% higher accuracy in comparison with the fixed-point models. Based on the simulation results, the sign bit of the posit numbers was found to be the most sensitive bit to error. Finally, an enhanced fault-tolerant posit-based DNN was proposed, where the results showed 41% and 11% higher classification accuracy for the LeNet-5 network trained under the MNIST dataset, compared to the FLOAT and unprotected P(8,0)-based models, respectively.

References

[1] M. Al-Qizwini, I. Barjasteh, H. Al-Qassab, and H. Radha, "Deep learning algorithm for autonomous driving using GoogLeNet," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 89–96.

[2] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: Review, opportunities and challenges," *Briefings Bioinf.*, vol. 19, no. 6, pp. 1236–1246, May 2017.

[3] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer, "Policy compression for aircraft collision avoidance systems," in *Proc. IEEE/AIAA 35th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2016, pp. 1–10.

[4] N. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Architect.*, 2017, pp. 1–12.

[5] F. Sijstermans, "The NVIDIA deep learning accelerator," *in Proc. Hot Chips*, Mar. 2018.

[6] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.

[7] B. Reagen et al., "Ares: A framework for quantifying the resilience of deep neural networks," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6.

[8] N. Jouppi et al., "TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," in *Proc. 50th Annu. Int. Symp. Comput. Archit.*, Jun. 2023, pp. 1147–1160.

[9] J. Vafaei, O. Akbari, M. Shafique, and C. Hochberger, "X-rel: Energy-efficient and low-overhead approximate reliability framework for error-tolerant applications deployed in critical systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 31, no. 7, pp. 1051–1064, Jul. 2023.

[10] M. A. Hanif and M. Shafique, "Dependable deep learning: Towards cost-efficient resilience of deep neural network accelerators against soft errors and permanent faults," in *Proc. IEEE 26th Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, Jul. 2020, pp. 1–4.

[11] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM J. Res. Develop.*, vol. 6, no. 2, pp. 200–209, Apr. 1962.

[12] M. Abdullah Hanif and M. Shafique, "SalvageDNN: Salvaging deep neural network accelerators with permanent faults through saliency-driven fault-aware mapping," *Phil. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 378, no. 2164, Feb. 2020, Art. no. 20190164.

[13] Z. Chen, G. Li, and K. Pattabiraman, "A low-cost fault corrector for deep neural networks through range restriction," in *Proc. 51st Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2021, pp. 1–13.

[14] S. Kim, P. Howe, T. Moreau, A. Alaghi, L. Ceze, and V. S. Sathe, "Energy-efficient neural network acceleration in the presence of bit-level memory errors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 12, pp. 4285–4298, Dec. 2018.

[15] J. J. Zhang, T. Gu, K. Basu, and S. Garg, "Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator," in *Proc. IEEE 36th VLSI Test Symp. (VTS)*, Apr. 2018, pp. 1–6.

[16] G. Alsuhli, V. Sakellariou, H. Saleh, M. Al-Qutayri, B. Mohammad, and T. Stouraitis, "Number systems for deep neural network architectures: A survey," 2023, *arXiv:2307.05035*.

[17] T. Norrie et al., "The design process for Google's training chips: TPUv2 and TPUv3," *IEEE Micro*, vol. 41, no. 2, pp. 56–63, Mar. 2021.

[18] J. White, K. Adámek, J. Roy, S. Dimoudi, S. M. Ransom, and W. Armour, "Bits missing: Finding exotic pulsars using bfloat16 on NVIDIA GPUs," *Astrophysical J. Suppl. Ser.*, vol. 265, no. 1, p. 13, Mar. 2023.

[19] V. Popescu, M. Nassar, X. Wang, E. Tumer, and T. Webb, "Flexpoint: Predictive numerics for deep learning," in *Proc. IEEE 25th Symp. Comput. Arithmetic (ARITH)*, Jun. 2018, pp. 1–4.

[20] I. Hubara, M. Courbariaux, and D. Soudry, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, pp. 1–30, Jan. 2018.

[21] S. Shankland. (2019). *Meet Tesla's Self-Driving Car Computer and Its Two AI Brains*. [Online]. Available: https://www.cnet.com/news/meet-tesla-self-driving-car-computer-and-its-two-ai-brains/

[22] A. C. I. Malossi et al., "The transprecision computing paradigm: Concept, design, and applications," in *Proc. Design, Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2018, pp. 1105–1110.

[23] J. L. Gustafson and I. T. Yonemoto, "Beating floating point at its own game: Posit arithmetic," *Supercomput. Frontiers Innov.*, vol. 4, no. 2, pp. 71–86, 2017.

[24] M. Zolfagharinejad, M. Kamal, A. Afzali-Khusha, and M. Pedram, "Posit process element for using in energy-efficient DNN accelerators," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 6, pp. 844–848, Jun. 2022.

[25] M. Cococcioni, F. Rossi, E. Ruffaldi, S. Saponara, and B. D. D. Dinechin, "Novel arithmetics in deep neural networks signal processing for autonomous driving: Challenges and opportunities," *IEEE Signal Process. Mag.*, vol. 38, no. 1, pp. 97–110, Jan. 2021.

[26] I. Alouani, A. Ben Khalifa, F. Merchant, and R. Leupers, "An investigation on inherent robustness of posit data representation," in *Proc. 34th Int. Conf. VLSI Design 20th Int. Conf. Embedded Syst. (VLSID)*, Feb. 2021, pp. 276–281.

[27] M. A. Neggaz, I. Alouani, S. Niar, and F. Kurdahi, "Are CNNs reliable enough for critical applications? An exploratory study," *IEEE Des. Test*, vol. 37, no. 2, pp. 76–83, Apr. 2020.

[28] R. Vadlamani et al., "Multicore soft error rate stabilization using adaptive dual modular redundancy," in *Proc. DATE*, 2010, pp. 27–32.

[29] L.-C. Chu and B. W. Wah, "Fault tolerant neural networks with hybrid redundancy," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 2, Jun. 1990, pp. 639–649.

[30] L.-H. Hoang, M. A. Hanif, and M. Shafique, "FT-ClipAct: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation," in *Proc. 23rd Conf. Design, Autom. Test Europe*, 2020, pp. 1241–1246.

[31] K. Zhao et al., "FT-CNN: Algorithm-based fault tolerance for convolutional neural networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1677–1689, Jul. 2021.

[32] A. Colucci, A. Steininger, and M. Shafique, "Enpheeph: A fault injection framework for spiking and compressed deep neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 5155–5162.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.

[33] Y. He, P. Balaprakash, and Y. Li, "Fidelity: Efficient resilience analysis framework for deep learning accelerators," in *Proc. Annu. Int. Symp. Microarchitecture (MICRO)*, Oct. 2020, pp. 270–281.
[34] M. Traiola, A. Kritikakou, and O. Sentieys, "HarDNNing: A machine-learning-based framework for fault tolerance assessment and protection of DNNs," in *Proc. IEEE Eur. Test Symp. (ETS)*, May 2023, pp. 1–6.
[35] A. Chaudhuri, C.-Y. Chen, J. Talukdar, S. Madala, A. K. Dubey, and K. Chakrabarty, "Efficient fault-criticality analysis for AI accelerators using a neural twin," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2021, pp. 73–82.
[36] F. F. D. Santos et al., "Analyzing and increasing the reliability of convolutional neural networks on GPUs," *IEEE Trans. Rel.*, vol. 68, no. 2, pp. 663–677, Jun. 2019.
[37] A. Ruospo, E. Sanchez, M. Traiola, I. O'Connor, and A. Bosio, "Investigating data representation for efficient and reliable convolutional neural networks," *Microprocess. Microsyst.*, vol. 86, Oct. 2021, Art. no. 104318.
[38] M. H. Ahmadilivani, M. Taheri, J. Raik, M. Daneshtalab, and M. Jenihhin, "DeepVigor: VulnerabIlity value RanGes and FactORs for DNNs' reliability assessment," in *Proc. IEEE Eur. Test Symp. (ETS)*, May 2023, pp. 1–6.
[39] N. Sharma et al., "CLARINET: A RISC-V based framework for posit arithmetic empiricism," 2020, *arXiv:2006.00364*.
[40] D. Mallasén, R. Murillo, A. A. Del Barrio, G. Botella, L. Piñuel, and M. Prieto, "PERCIVAL: Open-source posit RISC-V core with quire capability," 2021, *arXiv:2111.15286*.
[41] A. Ruospo et al., "Assessing convolutional neural networks reliability through statistical fault injections," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Apr. 2023.
[42] N. Samimi, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Res-DNN: A residue number system-based DNN accelerator unit," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 2, pp. 658–671, Feb. 2020.
[43] M. Jasemi, S. Hessabi, and N. Bagherzadeh, "Enhancing reliability of emerging memory technology for machine learning accelerators," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 4, pp. 2234–2240, Oct. 2021.
[44] M. Cococcioni. *POSIT Arithmetic for Autonomous Driving*. Accessed: Feb. 2022. [Online]. Available: https://vivid-sparks.com/
[45] M. Cococcioni, E. Ruffaldi, and S. Saponara, "Exploiting posit arithmetic for deep neural networks in autonomous driving applications," in *Proc. Int. Conf. Electr. Electron. Technol. Automot.*, Jul. 2018, pp. 1–6.
[46] R. Murillo, A. A. Del Barrio, and G. Botella, "Deep PeNSieve: A deep learning framework based on the posit number system," *Digit. Signal Process.*, vol. 102, Jul. 2020, Art. no. 102762.
[47] G. Raposo, P. Tomás, and N. Roma, "PositNN: Training deep neural networks with mixed low-precision posit," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 7908–7912.
[48] N. M. Ho, H. De Silva, J. L. Gustafson, and W. F. Wong, "Qtorch+: Next generation arithmetic for PyTorch machine learning," in *Proc. Conf. Next Gener. Arithmetic* (Lecture Notes in Computer Science), vol. 13253, 2022, pp. 31–49.
[49] M. K. Jaiswal and H. K.-H. So, "PACoGen: A hardware posit arithmetic core generator," *IEEE Access*, vol. 7, pp. 74586–74601, 2019.
[50] S. Nambi, S. Ullah, S. S. Sahoo, A. Lohana, F. Merchant, and A. Kumar, "ExPAN(N)D: Exploring posits for efficient artificial neural network design in FPGA-based systems," *IEEE Access*, vol. 9, pp. 103691–103708, 2021.
[51] Posit Working Group. (2018). *Posit Standard Documentation*. Posit Standard Documentation. [Online]. Available: https://posithub.org/docs/posit_standard.pdf
[52] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 901–909. [Online]. Available: https://github.com/openai/weightnorm
[53] Z. Carmichael, H. F. Langroudi, C. Khazanov, J. Lillie, J. L. Gustafson, and D. Kudithipudi, "Performance-efficiency trade-off of low-precision numerical formats in deep neural networks," in *Proc. Conf. Next Gener. Arithmetic*, Mar. 2019, pp. 1–9.
[54] G. Li et al., "Understanding error propagation in deep learning neural network (DNN) accelerators and applications," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2017, pp. 1–12.
[55] M. Qin, C. Sun, and D. Vucinic, "Robustness of neural networks against storage media errors," 2017, *arXiv:1709.06173*.
[56] H. Guan et al., "In-place zero-space memory protection for CNN," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–10.
[57] S.-S. Lee and J.-S. Yang, "Value-aware parity insertion ECC for fault-tolerant deep neural network," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2022, pp. 724–729.

Morteza Yousefloo received the M.Sc. degree in computer engineering from Tarbiat Modares University, Tehran, Iran, in 2023.

He is currently a Research Assistance of the Computer Architecture and Dependble Syetems Laboratory (CADS-Lab), Department of Electrical and Computer Engineering, Tarbiat Modares University. His research interests include machine learning (ML), deep neural networks (DNNs), the Internet of Things (IoT), and fault-tolerant systems.

Omid Akbari received the B.Sc. degree from the University of Guilan, Rasht, Iran, in 2011, the M.Sc. degree from Iran University of Science and Technology, Tehran, Iran, in 2013, and the Ph.D. degree from the University of Tehran, Iran, in 2018, all in electrical engineering, electronics–digital systems subdiscipline.

He was a Visiting Researcher at the CARE-Tech Laboratories, Vienna University of Technology (TU Vienna), Vienna, Austria, from April 2017 to October 2017, and a Visiting Research Fellow under the Future Talent Guest Stay program at Technische Universität Darmstadt (TU Darmstadt), Germany, from July 2022 to September 2022. He is currently an Assistant Professor of Electrical and Computer Engineering at Tarbiat Modares University, Tehran, Iran, where he is also the Director of the Computer Architecture and Dependable Systems Laboratory (CADS-Lab). His current research interests include embedded machine learning (ML), reconfigurable computing, energy-efficient computing, distributed learning, and fault-tolerant system design.

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY MADRAS. Downloaded on July 21,2024 at 12:58:15 UTC from IEEE Xplore. Restrictions apply.