IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 February 2024, accepted 26 March 2024, date of publication 8 April 2024, date of current version 18 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3386431

RESEARCH ARTICLE

Analysis and Enhancement of Resilience for LSTM Accelerators Using Residue-Based CEDs

NOOSHIN NOSRATI, (Student Member, IEEE), AND ZAINALABEDIN NAVABI, (Life Senior Member, IEEE)
School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran 14174, Iran
Corresponding author: Zainalabedin Navabi (navabi@ut.ac.ir)

ABSTRACT As Long Short-Term Memory (LSTM) accelerators are increasingly being employed in safety-critical applications with high-reliability demands, protecting them against errors becomes imperative. Traditional protection techniques for LSTMs are either costly, conflict with strict area and power constraints for neural network accelerators, or introduce performance overhead that is untenable for real-time and latency-critical accelerators. In this paper, we propose residue-based Concurrent Error Detection (CED) schemes to detect transient faults and alleviate their impact during LSTM computations. CED units are employed in a coarse-grain or fine-grain fashion, depending on the granularity of the components being protected from the whole LSTM computations to individual processing elements. In pursuit of a more cost-effective strategy, we use fine-grain CEDs in a selective manner based on the resilience characteristics of LSTM. The selections are made spatially for LSTM synaptic weights or temporally based on LSTM time steps. The experimental results show that the proposed residue-based CEDs (i) can achieve nearly complete fault coverage even under extremely large bit error rates, (ii) significantly decrease misprediction rates compared to the unprotected LSTM, and (iii) incur low overhead without compromising performance. Our method is compared with modular redundancy techniques such as DMR and TMR (Double and Triple Modular Redundancy).

INDEX TERMS AI accelerators, deep learning, fault tolerance, long short-term memory, reliability.

I. INTRODUCTION

Recurrent Neural Networks (RNNs) are a class of Deep Neural Networks (DNNs) with the capability of remembering past information [1], [2]. Unlike traditional feedforward neural networks, RNNs have feedback connections, allowing them to maintain a memory of previous inputs. The most commonly used RNN is Long Short-Term Memory (LSTM) [1], which can learn the long-term dependencies in input sequences, providing high accuracy. LSTM networks have demonstrated great effectiveness in real-time safety-critical sequence processing tasks, such as robotics [3], healthcare devices [4], and autonomous driving [5].

For efficient performance of LSTMs in the inference phase, dedicated hardware accelerators are employed [2], [6], [7]. However, these accelerators are susceptible to faults stemming from fabrication technology and hardware architectures [8], [9], [10], [11]. From the fabrication technology perspective, in the resilience-wall era [12], faults are caused by factors such as high-energy cosmic radiation, aging, and temperature variations [11]. These factors that are part of the physics of hardware are inevitable in hardware systems. From the hardware architecture point of view, hardware accelerators are usually optimized for performance and power efficiency, which introduce reliability challenges [13].

Our preliminary reliability analysis, as illustrated in Figure 1 and supported by recent studies [8], [14] highlights how vulnerabilities mentioned above from the hardware can propagate and impact the output of LSTM networks. In this figure, the misclassification rate of LSTM, referred to as Silent Data Corruptions (SDCs), is depicted for various bit error rate (BER) values. It can be seen that the SDC rate

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

increases with growing fault rates. Here, random bit-flip faults are injected into the computations of the LSTM networks performing sequence classification on the MNIST [15] and GTSRB [16] datasets.

In a nutshell, faults have the potential to jeopardize the integrity of LSTMs, leading to violations of safety and reliability specifications. Hence, there is a vital necessity to enhance the resilience of these networks in safety-critical applications.

In this paper, we present two Concurrent Error Detection (CED) approaches utilizing residue codes to protect LSTM accelerators against the transient faults that occur during the network computations. The first approach that we refer to as regular protection applies residue-based CEDs at coarse and fine granularity levels to all LSTM computations. The coarse-grain CED scheme monitors the whole LSTM computations in a begin-to-end approach, while the fine-grain CED solution safeguards individual processing elements (PEs).

Beyond the regular protection mechanisms that uniformly safeguard all network computations without any discrimination, we propose an alternative approach named *criticality-aware partial protection* to provide a cost-effective protection mechanism. In this approach, we consider the criticality of area, power, and vulnerability in choosing the arithmetic components to place fine-grain CEDs. Here, we explore both spatial and temporal resilience characteristics of LSTM. Based on this, we employ the fine-grain CEDs in a selective fashion and propose spatial and temporal CED selection schemes tailored to the distinct features of LSTM. These criticality-aware schemes provide a cost-effective protection mechanism that aligns with reliability specifications.

The following summarizes the main contributions of this work.

- While works in the literature mainly focus on reliability issues of feedforward neural networks, we analyze and enhance the resilience of recurrent neural networks, particularly LSTMs.
- We employ residue-based CEDs to make LSTMs reliable against transient faults, while other works in this domain have used AN-codes for feedforward networks.
- This work is unique in applications of CEDs to LSTM computations at coarse-grain and fine-grain levels.
- Instead of indiscriminate application of CEDs to all network PEs at all time steps, we have adopted novel methods of spatial and temporal selection of CEDs based on LSTM resilience analysis. This reduces the protection overheads with minimal impact on SDC coverage.
- We implemented our approach within the TensorFlow deep learning framework and applied it to sequence classification LSTM models to demonstrate reliability enhancements.
- To assess the hardware overheads, the LSTM models were mapped onto a state-of-the-art LSTM accelerator equipped with our CEDs. The protected hardware accelerator was synthesized in 45 nm technology.

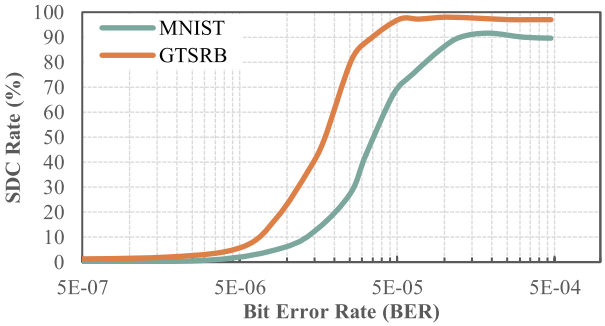


FIGURE 1. LSTM vulnerability assessment. Error bars range from ±0.51% to ±2.66% at the 95% confidence interval.

This work suggests an interaction between designers and developers of an IP-core and those who incorporate the core into a System-on-Chip (SoC) or embedded system and develop an application for it. A possible scenario of use is that the application developer (core user) decides what level of reliability of the IP-core he or she desires, and based on that purchases the IP-core with that specific reliability grading. This reliability grading has the specifications including power and area that the IP-core developer provides. Furthermore, to reduce power usage, the application developer may select the CEDs that should be turned on or remain inactive. This selection can be configured by the core user using JTAG or a similar serial configuration. For various applications, some templates of temporal selections will be provided by the IP-core developer.

The rest of the paper is organized as follows. In Section II, related protection methods for DNNs are reviewed. Section III provides the necessary background on LSTMs. Section IV gives an overview of residue codes and building blocks of our residue-based CEDs. The proposed regular and criticality-aware protection mechanisms are detailed in Sections V and VI, respectively. In Section VII, the simulation results are presented and discussed. Finally, Section VIII concludes the paper.

II. RELATED WORK

In the literature, the majority of research focuses on fault detection and mitigation techniques for feedforward networks, while only a few attempts address fault tolerance issues in recurrent networks. Nevertheless, the existing research on the reliability of DNNs can be categorized into the following primary approaches.

A. HARDWARE MODULAR REDUNDANCY

Modular redundancy in hardware, encompassing both traditional DMR and TMR (Double/Triple Modular Redundancy) techniques [17], ensures robust reliability by operating identical systems concurrently and implementing a voter system to cross-check outputs. However, the substantial energy and hardware overheads they incur are at odds with the primary goal of optimizing DNN accelerators for computational effi-

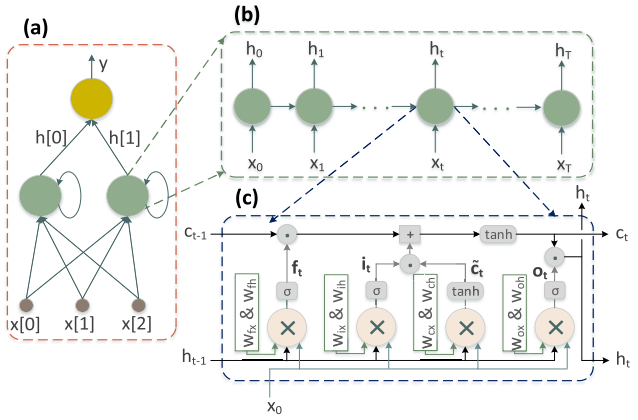


FIGURE 2. (a) RNN with LSTM cells, (b) LSTM cell unfolded over time steps, (c) LSTM cell architecture.

ciency in resource-limited systems. To reduce overheads, works [18], [19], [20] selectively applied modular redundancies to protect only parts of the network. Although these techniques provide a better balance between resilience and resource usage, they often suffer from limited fault coverage. For instance, [18] was capable to cover merely 40% of faults.

B. ALGORITHM-BASED FAULT TOLERANCE

Another line of research leverages algorithmic perspective to make DNNs reliable. These methods, known as Algorithm-Based Fault Tolerance (ABFT) [21], [22], [23], [24], typically compute checksums for input data, which are then stored alongside the original data. Then, both the original and redundant computations are conducted to verify outputs. Zhao et. al. [22] proposed four different ABFT schemes to protect convolutional layers. Hari et. al. [23] suggested an error detection technique with precise checksums for deterministic coverage. In [24], a new ABFT technique was proposed to predict the convolution checksum at the border of the input image. Nevertheless, these techniques usually incur remarkable latency or performance overhead due to redundant computations, which makes them unsuitable for latency-critical DNN inference.

C. SYMPTOM-BASED ERROR DETECTION

The studies outlined in [10], [14], [25], and [26] found deviation of an expected range of values as a symptom of the occurrence of an error. They rectified faulty values through the restriction of bounds. Works presented in [10] and [25] used a global bound reference value for the activation output of all neurons in a network layer to enhance the error resilience of feedforward DNNs. However, global bounds become ineffective with growing BER. Ghavami et. al. [26] addressed this issue by employing a fine-grained neuron-wise activation function in the network. On the other hand, Ahmadilivani et. al. [14] investigated a similar strategy for LSTM networks. They leveraged the range checking for weights instead of the activation outputs. Then, the faulty weights were replaced with zero. Despite the low overhead, these methods are inefficient when applied to quantized models with smaller data types, which are frequently employed to enhance energy efficiency and performance of DNN accelerators.

D. CODE-BASED ERROR DETECTION

Code-based techniques have been used for the protection of data processing [27], [28], [29] and memory [30], [31] in DNNs. Feinberg et. al. [28] proposed data-aware AN-codes to protect memristive accelerators. They leveraged the noise characteristics and data layout of DNN to detect and correct faults through modulus operations and a correction table lookup. Goldstein et. al. [27] employed a similar code-based scheme, i.e., AN-codes, to protect MAC PEs of a systolic array-based accelerator. They necessitated a specific data quantization during network training to reduce protection overhead. It may result in a reduction of accuracy and, in some cases, require retraining. Nevertheless, AN-codes do not draw a line between the coded circuitry and the circuitry performing the original operation. This deficiency leads to a notable loss of practical implementation advantages compared to separable arithmetic codes such as residue-codes that are adopted in this work.

III. BACKGROUND

A. LONG SHORT-TERM MEMORY (LSTM) NETWORKS

Figure 2(a) shows an RNN with LSTM cells/layer [1] that has feedback connections allowing the use of information from previous inputs. An illustration of a single LSTM cell unfolded over time steps is shown in Figure 2(b). As shown, in each time step t, current input (x_t) and information from previous input(s) are taken to compute the current output (h_t).

There are various extensions of LSTM cell depending on the cell structure and learning process. The standard LSTM cell illustrated in Figure 2(c) is used to compute a mapping from an input sequence $x = (x_1, \ldots, x_T)$ to a hidden sequence $h = (h_1, \ldots, h_T)$ by using the following set of equations iteratively from $t = 1$ to T:

$$i_t = \sigma\left(w_{ix}x_t + w_{ih}h_{t-1} + b_i\right), \quad \text{input gate} \tag{1}$$

$$f_t = \sigma\left(w_{fx}x_t + w_{fh}h_{t-1} + b_f\right), \quad \text{forget gate} \tag{2}$$

$$o_t = \sigma\left(w_{ox}x_t + w_{oh}h_{t-1} + b_o\right), \quad \text{output gate} \tag{3}$$

$$\tilde{c}_t = \tanh\left(w_{cx}x_t + w_{ch}h_{t-1} + b_c\right), \quad \text{candidate memory} \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \quad \text{memory cell} \tag{5}$$

$$h_t = o_t \odot tanh\left(c_t\right). \quad \text{hidden state} \tag{6}$$

where w_{jx} and w_{jh} $(j{=}i,f,o,c)$ are input and hidden weight matrices and b_j $(j{=}i,f,o,c)$ represents the biases. c_t is memory data at time step t. $\odot$ denotes element-wise multiplication. σ and $tanh$ are sigmoid and hyperbolic tangent activation functions.

As shown in Equations (1) to (6) and Figure 2(c), an LSTM unit consists of input, forget, and output gates considered

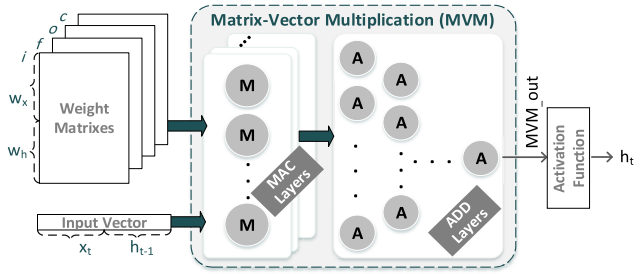


FIGURE 3. MVM Engine in LSTM.

as regulation structures, and a candidate memory. The gates regulate the flow of information into and out of the memory cell, allowing the network to control what information to remember, forget, and output.

1) TRAINING - BACKWARD PASS

Training is the process of turning the weight matrices in order to minimize the total prediction error. The most common method used to train LSTMs is Back-Propagation Through Time (BPTT) [32], which is an extension of backpropagation used in traditional neural networks. BPTT involves unfolding the network in time steps and propagating error signals backward through these time steps. To update the weights, BPTT computes the gradients of total prediction error with respect to each weight ($\partial E_{tot}/\partial W$). E_{tot} is a total error function that is defined as the difference between the predicted output and the actual output in all time steps. It can be calculated using a cost function, such as mean squared error or cross-entropy loss.

It is worth mentioning that the specific training mechanism of LSTM may vary depending on the implementation and the specific task at hand, but the general principles mentioned above form the basis of training an LSTM model.

2) INFERENCE - FORWARD PASS

The inference is the phase in which a trained network is used to predict outputs for unseen input data. Computationally, the gates and candidate memory in Equations (1) to (4) are matrix-vector multiplication (MVM) blocks, followed by a nonlinear elementwise activation function. The MVM operations, as the dominant computation of the LSTM inference, can be constructed by a Multiply/MAC stage followed by an ADD (tree-like) stage as shown in Figure 3. The PEs in this architecture are MAC and ADD units, i.e., *M* and *A* notation in the figure.

As over 90% of the network's computation relies on MAC and ADD operations [2], the correct execution of MAC and ADD PEs directly influences the overall system reliability. Consequently, we build our CED solutions upon the MVM engine depicted in Figure 3.

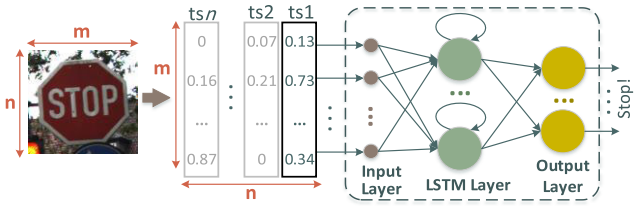


FIGURE 4. Sequence classification using LSTM.

B. SEQUENCE CLASSIFICATION USING LSTM

LSTMs are commonly employed for sequence classification tasks [33], [34]. These applications aim to assign a label or category to an input sequence. The input sequence might take the form of a time series, a series of pixels in an image, a sequence of images within a video frame, or any other type of sequential data.

Figure 4 provides a simple illustration of using LSTM for image sequence classification [34], as adopted in this paper. As shown in the figure, each image is inferred as n rows of m pixels (i.e., columns). At each time step, a single row of m pixels is fed into the LSTM model to compute output sequences. Consequently, the processing of a complete image involves n time steps in LSTM. In this illustration, the input data sequence is directly applied to the LSTM input. However, in more practical situations, the input data undergoes preprocessing through a feature extractor unit or any type of pre-processor to identify important features. Subsequently, the extracted features are fed into the LSTM at each time step.

In the network shown in Figure 4, the LSTM output sequences at the last time step pass through a fully connected layer to calculate the probability distribution across the various classes. The number of neurons in the fully connected layer corresponds to the total number of classes for classification.

IV. ARITHMETIC ERROR CODES

Code-based techniques add information redundancy with the property that certain errors can be detected or sometimes corrected. While non-arithmetic codes, like Hamming and Cyclic codes, are well-suited for memory [30] and communication [35] protection, data processing can efficiently be protected through arithmetic codes like AN-codes and residue codes [27], [28], [36], [37].

Residue codes are a well-known class of arithmetic codes that operate using residue arithmetic to check a set of arithmetic operations. The following subsections discuss how residue codes are adopted for fault detection and the hardware details of a residue generator, serving as a building block utilized in the design of our residue-based CEDs.

A. RESIDUE ARITHMETIC CODES

Residue code error detection involves encoding data through modulo operations prior to executing residue-based calculations. In the encoding process, the data is divided by a

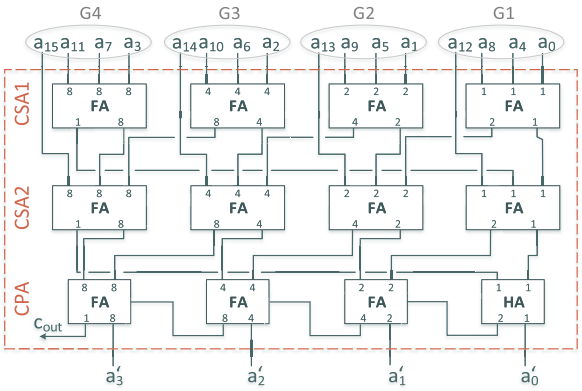


FIGURE 5. 16-bit residue generator modulo 15.

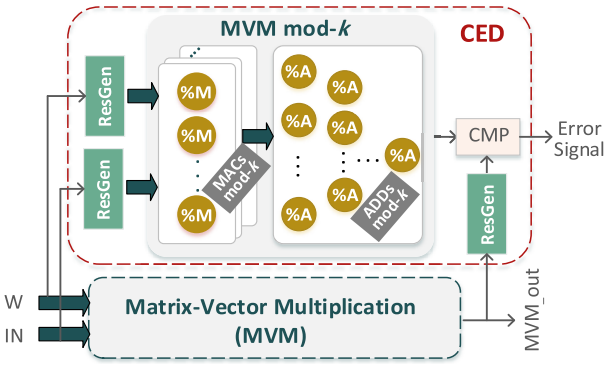


FIGURE 6. Block diagram of the proposed coarse-grain CED. MVM Engine is shown in Figure 3.

specified modulus, and the remainder is retained as the encoded value. After encoding the data, a residue-based execution of the same arithmetic operation as the original is conducted in parallel with the original operations. The results of the residue-based calculations are compared with the original output encoded by the same modulo operation. A mismatch indicates an error in the original arithmetic operation. Formally, an arithmetic operation can be checked by

$$|A \circledast B|_k \stackrel{?}{=} \left| |A|_k \circledast |B|_k \right|_k, \circledast \epsilon \{+, -, \times\} \tag{7}$$

where $|A|_k$ is A modulo (mod) k.

The error detection capability with the residue code depends on the choice of the k value in Equation (7), which is referred to as *check base*. A low-cost implementation of residue code, used in this work, has a check base of Mersenne numbers $k = 2^n - 1$, where n is an integer value greater than 1 [36], [37]. The smallest n value, i.e., k=3, provides the least expensive residue detection hardware. However, this circuit does not offer the best detection capability due to increasing the probability of aliasing. Aliasing occurs when the code-word of a faulty value is the same as the fault-free one.

B. RESIDUE GENERATOR HARDWARE

For fault detection using residue codes modulo $2^n - 1$, an essential building block is the residue generator. The residue generator is responsible for calculating the remainder of the division of an m-bit integer number $A = (a_{m-1} \ldots a_1 a_0)$ by $k = 2^n - 1$, where the remainder is an n-bit integer output.

The work [36] presented a procedure to design fast and low-cost residue generators. It introduced the periodicity of the series of powers of 2 taken modulo k. For the check base $k = 2^n - 1$, the periodicity measure is $P(k) = n$. Based on the periodicity, a carry-save adder/ carry-propagate adder (CSA/CPA) network with end-around carry (EAC) can be built to calculate modulo $2^n - 1$. In this method, the CSA trees are used to reduce partial results with calculations carried on several bits of $P(k)$ groups. Then, the periodicity property is used to add carry from the most significant bit (MSB) back to the next stage of the tree on the least significant bit (LSB).

Figure 5 illustrates the circuit diagram of a 16-bit residue generator modulo 15. For $k = 15$, the periodicity measure is 4, $P(15) = 4$. Therefore, the bits with the same weight are organized into 4 groups. Grouping the bits of the same weight exploits the fact that the result of applying residue modulo 15 to the 16-bit integers 1, 16, 256, and 4096 is consistently 1. Consequently, bits a_0, a_4, a_8, a_{12} are grouped in G_1. Likewise, the application of residue module 15 to 16-bit integers 2, 32, 512, and 8192 consistently yields 2. As a result, bits a_1, a_5, a_9, a_{13} are grouped in G_2, and this pattern continues for G_3 and G_4. After the grouping process, the 16-bit residue generator modulo 15 is constructed by employing two stages of CSA followed by a CPA stage, as shown in the figure. Notably, the CPA stage functions as an 8-bit residue generator modulo 15, effectively serving as the modulo 15 adder.

V. REGULAR PROTECTION MECHANISMS

To improve the reliability of LSTM accelerators, we propose Concurrent Error Detection (CED) solutions rooted in residue code error detection, as explained in the preceding section. In this section, we present two residue-based CED solutions: coarse-grain and fine-grain. These solutions serve as regular protection mechanisms, fully safeguarding LSTM computations.

A. COARSE-GRAIN CED (CG-CED)

In the coarse-grain CED (CG-CED) solution, we construct one residue-based checker on the whole MVM structure of an LSTM. Using Equations (1) to (4), the MVM structure in each time step can be calculated as:

$$MVM_{out\,t} = w_{jx} x_t + w_{jh} h_{t-1} + b_j, j = i, f, o, c. \tag{8}$$

Considering input dimension d_i and hidden dimension d_h, Equation (8) becomes:

$$MVM_{out_t} = \sum\nolimits_{d_i} w_{jx} \times x_t + \sum\nolimits_{d_h} w_{jh} \times h_{t-1} + b_j. \tag{9}$$

We concatenate weights and biases to form matrix W, and concatenate inputs to form vector IN, where

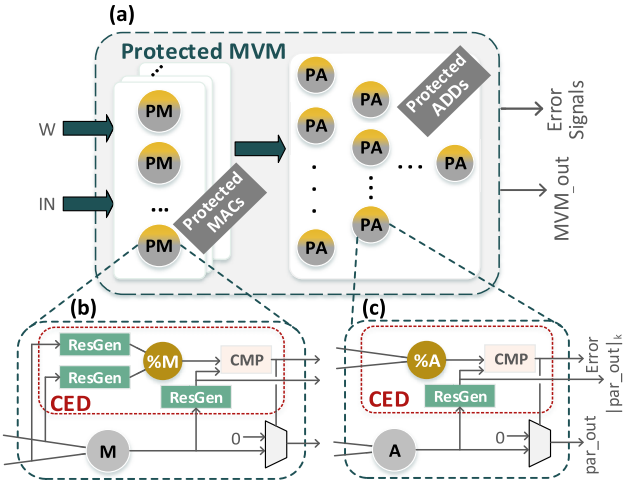


FIGURE 7. (a) Block diagram of the proposed fine-grain CED, (b) Input-side protected PE, (c) Middle protected PE.

$W \in \mathbb{R}^{4d_h \times (d_i+d_h+1)}$ and $IN \in \mathbb{R}^{(d_i+d_h+1)\times 1}$. Therefore, MVM can be simplified as:

$$MVM_{out_t} = \sum_{d_i+d_h+1} W \times IN. \tag{10}$$

Applying Equation (7) to Equation (10), the MVM output in each time step can be checked by the right-hand side of Equation (11),

$$|MVM_{out_t}|_k \stackrel{?}{=} |\sum_{d_i+d_h+1} ||W|_k \times |IN|_k|_k|_k. \tag{11}$$

The CED box in Figure 6 depicts the implementation of Equation (11) and provides an overview of the CG-CED solution. As seen in the figure, the residue code of input operands is produced by the residue generators (e.g., see Figure 5) and then fed into the residue-based MVM. The architecture of the residue-based MVM follows the original circuit with mod-k arithmetic components, i.e., $\%M$ and $\%A$ for M and A PEs. They operate concurrent with the original circuit and perform residue-based matrix multiplication calculations.

A residue generator is placed at the output of the original circuit to calculate $|MVM\_out_t|_k$ in Equation (11). Consequently, the residue code of the original output, $|MVM\_out_t|_k$, is compared with the CED output (the right-hand side of the equality of Equation (11)). A mismatch detected by the residue number comparator indicates an error in the MVM calculation. Note in CG-CED architecture that the residue generators are only placed on the input side and the output side of the MVM structure.

The effectiveness of the residue-based detection scheme is compromised if a fault occurs in the checker, leading to a false alarm in the checker output. To mitigate this issue, one approach is to decrease the likelihood of faults by minimizing the checker's area footprint. In this context, our work utilizes a check base of Mersenne numbers, known for its cost-effective hardware implementation, as discussed in Section IV.

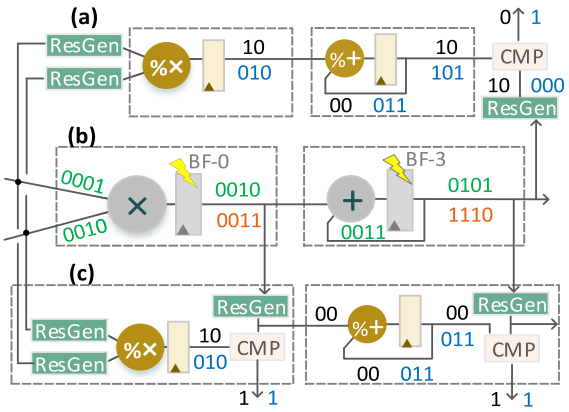


FIGURE 8. An example of aliasing, (a) Protection using CG-CED, (b) Original circuit, (c) Protection using FG-CED.

Another scenario where the CG-CED scheme falters is when faults modify the MVM result as such a fault-free codeword is produced. One solution to this aliasing problem is careful selection of the check base that can reduce the aliasing probability. The CG-CED will be evaluated for several check base values in Section VII. A better solution is to split the coarse-grain CED into multiple fine-grain CEDs to reduce the probability of aliasing. The next section elaborates on this topic.

B. FINE-GRAIN CED (FG-CED)

Instead of placing a coarse-grain CED on the whole MVM structure, from the input side to the output side, a more elaborate scheme is our fine-grain CED solution. This scheme distributes the planned CED to the individual PEs from the input side to the output side among all internal layers. This more-detailed checking not only enhances the error detection rate by minimizing the probability of aliasing but also equips the system with the ability to mitigate the effects of faults.

In applying the distributed CEDs to the MAC PEs, a fine-grain CED checks the partial result at time step t by the right-hand side of Equation (12),

$$|par\_out_t|_k \stackrel{?}{=} |\sum_q ||w|_k \times |in|_k|_k|_k \tag{12}$$

where q is the number of iterations of a processing element to calculate a partial result. $w \in \mathbb{R}^{1\times q}$ and $in \in \mathbb{R}^{q\times 1}$ represent subsets of weights and inputs that the PE operates on.

Figure 7(a) illustrates the overall scheme of FG-CED. As shown, MAC and ADD CEDs are attached to every MAC and ADD PE, respectively. A protected PE on the input side (e.g., an M unit in Figure 7(b)) requires residue generators on its inputs, performs modulo computations (e.g., a $\%M$ unit), and compares the residue value of its original output with the output of the modulo computations. Such a protected PE produces a regular output along with the residue-k of its output as shown in Figure 7(b). Availability of the residue-k outputs eliminates the need for residue generators on the inputs of PEs in the next layers of the MVM structure (Figure 7(c)). Not

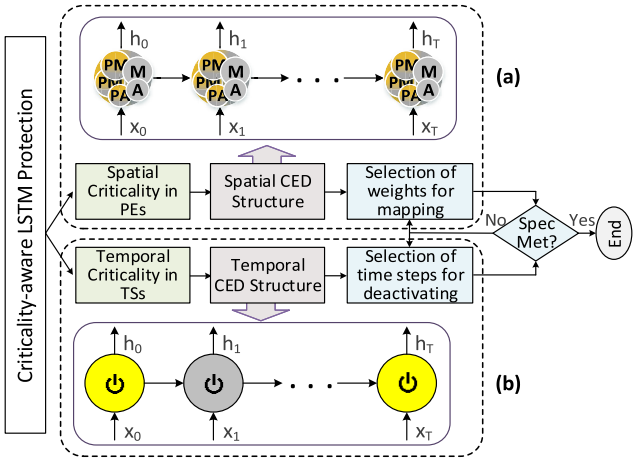
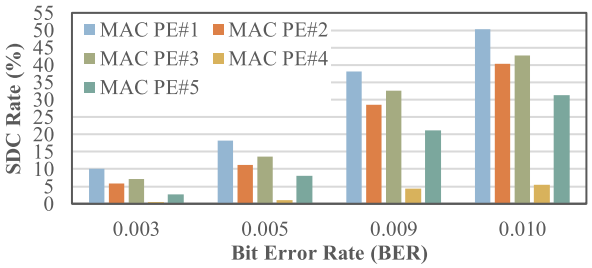
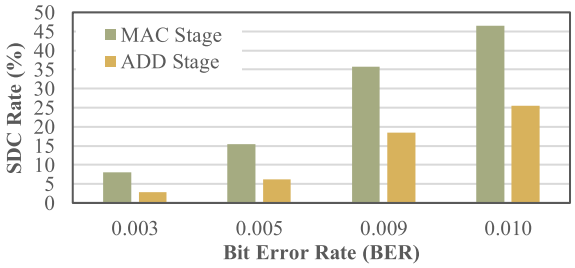


FIGURE 9. An overview of criticality-aware protection approach.



(a) SDC rate of different MAC PEs at various BERs. Error bars range from ±1.03% to ±2.19% at the 95% confidence interval.



(b) SDC rate of MAC and ADD PEs at various BERs. Error bars range from ±1.28% to ±2.54% at the 95% confidence interval.

FIGURE 10. Spatial criticality assessment for LSTM-MNIST.

requiring input residue generators for the PEs in the middle layers significantly reduces the required CED hardware.

C. ALIASING PROBABILITY FOR CG AND FG CEDS WITH DIFFERENT CHECK BASE VALUES

The simple diagram presented in Figure 8 clarifies the distinction between CG-CED and FG-CED in fault detection, particularly in a given scenario where aliasing is a possibility. Figure 8(b) depicts the fault-free (green) and faulty (red) values on the original circuit, comprising MAC and ADD PEs. Let us assume a bit-flip fault occurs at bit position 0 (BF-0) of MAC PE, and a bit-flip occurs at bit position 3 (BF-3) of ADD PE during different clock cycles. Similar scenarios may occur if two or more faults manifest simultaneously at the same clock cycle in one or different PEs. Figures 8(a) and (c) illustrate the coarse-grain and fine-grain detection schemes, respectively. Let us first consider the check base value of 3 (black values) for both CG-CED and FG-CED. With the coarse-grain solution, the effects of faults can be concealed, as the code-word for a faulty value becomes indistinguishable from the fault-free one. Consequently, Residue-3 CG-CED falsely indicates a fault-free situation. In contrast, Residue-3 FG-CED can correctly detect both injected faults. It reduces the probability of aliasing by checking PEs individually.

On the other hand, utilizing larger check base values, such as 7 denoted by blue numbers in the figure, has the potential to decrease the probability of aliasing even in CG-CED. This demonstrates an interesting conclusion that although in CG-CED the number of hardware components may be fewer than those of FG-CED, better reliability necessitates that the components be more complex in CG-CED.

VI. CRITICALITY-AWARE PARTIAL PROTECTION MECHANISMS

The regular protection mechanisms, FG-CED and CG-CED, uniformly safeguard all PEs in all time steps without any discrimination. Alternatively, in a different approach, that we refer to as criticality-aware partial protection, either more sensitive PEs are protected in all time steps, or all PEs are protected in some time steps that are recognized as more susceptible to errors. The former, that we refer to as Spatial CED Selection (SCS), is shown in Figure 9(a), while the latter, referred to as Temporal CED Selection (TCS), is presented in Figure 9(b). As shown in Figure 9, we first investigate the resilience characteristics of LSTM and introduce *spatial* and *temporal* criticalities. Building upon these insights, we present the structure of our CEDs. The CEDs involve selection methods to enable overhead-reliability trade-offs for working at a certain error rate. The details of the SCS and TCS approaches will be discussed in the subsections that follow.

A. SPATIAL CED SELECTION (SCS)

1) SPATIAL CRITICALITY

Spatial CED selection is grounded in the principle that DNN layers, filters, and neurons differ in their error resiliency [10], [13], [38]. The idea can be expanded to LSTM unfolded over time steps. Our preliminary results presented in Figure 10(a) demonstrate the error susceptibility (criticality) variations for PEs of an LSTM accelerator performing individual MAC computations. The study is for five randomly selected PEs. The figure depicts the SDC rate of an LSTM during sequence classification for MNIST at various BERs. Further details on the simulation setup will be discussed in Section VII-A. From this figure, we can observe that errors in PEs involving more critical MAC computations result in a substantial misprediction rate, whereas similar errors in computations with less critical computations lead to a smaller misprediction rate.

Further analysis is conducted on the error resiliency of MAC PEs compared to ADD PEs within the adder-tree, and

the results are shown in Figure 10(b). The findings indicate that the MAC PEs are more sensitive to errors than the ADD PEs. The above studies highlight the need for selection of arithmetic components to be protected, thus the *spatial criticality* associated with MAC and ADD computations in LSTM.

2) SPATIAL CED STRUCTURE

The spatial criticality is the concept of allowing only a subset of the PEs to be protected, instead of placing fine-grain CEDs on all PEs. This partial scheme is shown in the upper part of Figure 9(a). It is effective in reducing the hardware overhead of the protection circuitry, specifically in terms of area footprint and power consumption.

The CED structure used for implementing spatial criticality is in effect a partial FG-CED. Parts that are protected are decided at the design time. The CEDs are like those shown in Figures 7(b) and (c). As shown, the residue generator on the output of a protected PE feeds the residue computations of the protected PE in the next layer. This eliminates the need for residue generators on the inputs of the middle layer PEs. Because of this dependency among protection layers, the placement of CEDs on PEs must be in a contiguous fashion from the input side to the output of the MVM block to achieve appealing hardware overhead savings. If the CEDs are not contiguous, the middle layer CEDs become orphaned due to the absence of residue generator units on their inputs, requiring replacement with more costly input-side CEDs.

3) SELECTION METHOD

To assign the more critical computations to the protected PEs, it is necessary to assess the error sensitivity of the synaptic weights in the LSTM network. We consider this sensitivity as a measure of criticality. For this measure, we adopt the gradient-based approach [39] using the backpropagation algorithm (BPTT) [32], designed initially for updating a network's weights during the training phase. The other resilience prediction approaches like [37] and [39] are orthogonal to our work and can be employed in conjunction with it.

The entire process of the gradient-based sensitivity analysis is described in Algorithm 1. The algorithm involves iterating through each instance in the training dataset. For each training instance, a *forward pass* and a *backward pass* (discussed in Section III-A) are conducted.

In the *forward pass* (lines 5-8), the LSTM processes the input data and updates its internal states, including the cell state and hidden state. The total error is also accumulated at each time step. The *backward pass* is then performed through time. As shown in lines 9-11, we start from the last time step (T) and calculate gradients with respect to the total errors and the network parameters. Unlike the training process, here, we do not update the network parameters, i.e., weights and biases.

After calculating the gradients for all training instances, the average error contribution of each synaptic weight is computed (line 13). Finally, the absolute values of gradient errors are arranged in an ascending order to facilitate the identification of the most critical synapses. The algorithm ends on line 15 with a number of weights (as determined at the design time) selected as critical.

Algorithm 1 Gradient-based Sensitivity Analysis for LSTM.

Input: Training dataset: $\boldsymbol{D}_{train}$, Pre-trained LSTM parameters: $\boldsymbol{W}$, Number of time steps: T, Ratio of protected PEs: R_{CED}
Output: Set of sensitive synaptic weights: $\boldsymbol{S}_{\mathbf{SSW}}$

1 **Begin**
2 initialize ***Grad*** to [0, 0, …, 0]
3 **foreach** *(**x**,**y**) in* $\boldsymbol{D}_{train}$ **do**
4 initialize E_{tot} to 0
// Forward Pass
5 **for** $t = 1$ *to* T **do**
6 $(\boldsymbol{h_t}, \boldsymbol{c_t})$ = lstm_forward($\boldsymbol{x_t}, \boldsymbol{h_{t-1}}, \boldsymbol{c_{t-1}}, \boldsymbol{W}$)
//Equations (1) to (6)
7 E_{tot} += compute_loss($\boldsymbol{y_t}, \boldsymbol{h_t}$)
8 **end for**
// Backward Pass
9 **for** $t = T$ *to* 1 **do**
10 ***Grad*** += lstm_backward($E_{tot}, \boldsymbol{W}$)
11 **end for**
12 **end for**
13 ***Grad*** = ***Grad*** / range($\boldsymbol{D}_{train}$)
14 ***SortedGrad*** = sort_ascending(|***Grad***|)
15 $\boldsymbol{S}_{SSW}$ = ***SortedGrad***[(1-R_{CED})× range($\boldsymbol{W}$): range($\boldsymbol{W}$)]
16 **End**

Determining the optimal number of unprotected PEs in advance is challenging because the critical computation ratio varies based on the specific application being executed on the LSTM hardware and the associated reliability requirements. Consequently, some critical computations, typically associated with protected PEs, may be mapped onto unprotected PEs. A practical solution to address this challenge is to move beyond a binary categorization of PEs as either black (unprotected) or white (protected). Instead, considering a spectrum of protection levels can be accomplished by employing residue-based CEDs with diverse check bases. We propose this as an extension of our work, and it is not the focus of this paper.

B. TEMPORAL CED SELECTION (TCS)

While SCS represents a static method linked to synaptic weights, TCS is a dynamic technique associated with time steps. TCS illustrated in Figure 9(b) will be described here.

1) TEMPORAL CRITICALITY

On one side, the current hidden state, as described in Equations (1) to (6), exhibits a temporal dependency on the preceding hidden state. On the other side, a neuromor-

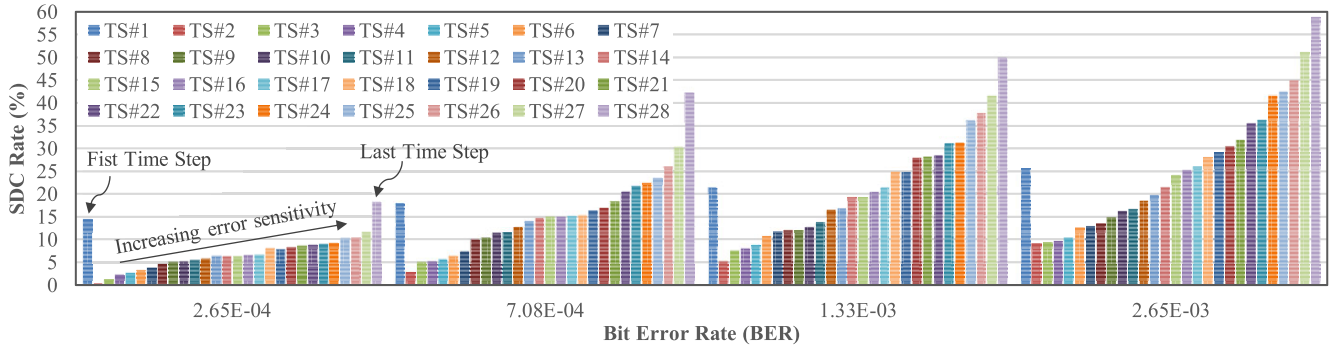


FIGURE 11. Temporal criticality assessment for LSTM-MNIST. Error bars range from ±1.07% to ±2.62% at the 95% confidence interval.

phic principle, *Delta Network*, asserts that hidden sequences exhibit stability in their activity transmission over time [7], [41]. These facts inspire the idea that hidden states at different time steps manifest distinct levels of error resiliency.

Along this line, we investigated the error sensitivity of the hidden sequence across various time steps, as depicted in Figure 11. The figure illustrates the SDC rate of the LSTM for 28 time steps under various BERs (refer to Section VII-A for simulation setup details). For a given BER, similar errors are injected in different time steps.

After a closer examination of the fault simulation results, a noticeable pattern becomes apparent. It is observed that the vulnerability to error increases for higher time steps, except for the first time step. This phenomenon is termed *temporal criticality*, which we define as the likelihood of the LSTM hidden state in a specific time step being susceptible to faults.

The high vulnerability observed in the first time step is attributed to the characteristics of the LSTM architecture detailed in Section III-A. As outlined in Equations (1) to (4), the bulk of the hidden state is constructed in the first time step, performing computations involving the input, forget, and output gates, along with the candidate memory. Then after the first time step, these units collectively regulate the flow of information through the network. Consequently, even a small deviation during the first time step has the potential to result in misprediction. Apart from the first time step, the computations at later time steps make a more significant contribution to the LSTM output. This higher influence in later time steps amplifies their susceptibility to errors.

2) TEMPORAL CED STRUCTURE

The temporal criticality concept allows deactivation of CEDs for time steps in which computations have a less significant effect on LSTM output (resilient time steps). This temporal CED scheme illustrated in the lower part of Figure 9(b) effectively mitigates power consumption associated with the fine-grain CEDs.

The CED deactivation involves minor modifications to the hardware of fine-grain CEDs and the controller. The addition of an inactive mode to the CEDs is accomplished through the insertion of tri-state logic into the inputs of each CED. These tri-state logics are controlled by a rotational shift-register that is responsible for activating/deactivating CEDs at each time step. The shift-register is placed in the controller unit of the LSTM accelerator and must be programmed prior to deploying the hardware accelerator for inference. The programming process will be discussed in the following part.

3) SELECTION METHOD

In an offline process, the time steps are categorized as resilient or sensitive through fault injection-based sensitivity analysis and the application of a heuristic threshold, denoted as Θ.

The sensitivity analysis based on fault injection determines the probability of misprediction for each time step. A hidden sequence in time step t of the LSTM computations is considered resilient if the probability of misprediction is below the threshold value ($misPred\ (h\,(t)) < \Theta$).

Subsequently, a *significance binary sequence* is generated with a length equal to the number of time steps. In this sequence, a '0' signifies that the corresponding time step is resilient, resulting in the deactivation of CEDs, and conversely, a '1' indicates sensitivity. The significance binary sequence is used for programming the controlling shift-register discussed above.

The Θ value plays an important role in achieving favorable reliability while concurrently conserving power consumption. It is noteworthy that the efficiency of TCS can be enhanced by integrating spatial criticality analysis with temporal considerations. This involves making decisions about the activation or deactivation of CEDs based on both resiliency of the protected PEs and the time step at which protection is applied. This makes an interesting and complex problem that can be dealt with in the future work of our research.

VII. EXPERIMENTAL RESULTS AND EVALUATIONS

In this section, we describe our simulation setup and present the simulation results to evaluate the effectiveness of the proposed protection mechanisms.

TABLE 1. Simulation setup.

Dataset	Dataset Description	# Output Classes	LSTM Layer			Test Accuracy
			Time Steps	Hidden Size	# Weights	
MNIST	Hand-written digits	10	28	256	291,840	98.2%
GTSRB	Real-world traffic sign	43	48	256	312,320	96.4%

A. EXPERIMENTAL SETUP

1) DNN MODELS AND DATASETS

For our evaluation, we trained two LSTM-based DNNs from scratch on the MNIST [15] and GTSRB [16] datasets. These network models consist of an LSTM layer followed by a fully connected layer for sequence classification [34], as discussed in Section III-B. Table 1 details the properties of the networks. The baseline accuracy (top-1) values for the LSTM-MNIST and LSTM-GTSRB are 98.2% and 96.4%, respectively.

2) FAULT MODEL

We consider hardware transient faults that occur randomly during the inference phase of the LSTMs. We assume that faults occur in the processing engine of LSTM accelerators rather than the memory, due to the reasons explained in [10] and [27]. Therefore, both weights and input sequences are fault-free. We inject bit-flip faults in storage elements of the datapath of the LSTM processing engine, i.e., MVM PEs. Faults in the controller unit of LSTM accelerators are not considered, as they constitute a negligible portion of the overall hardware. This is in line with the existing literature on reliability [10], [25], [42].

3) RESEARCH QUESTIONS

We answer the following research questions (RQs) in our experiments:

RQ1: How effective are coarse-grain and fine-grain CEDs in the detection of faults?

RQ2: How do coarse-grain and fine-grain CEDs contribute to improving resilience?

RQ3: What are the associated costs of the proposed CEDs?

RQ4: How do SCS and TCS contribute to selectively protecting LSTM?

4) BASELINES

To evaluate our work, we considered three versions of an LSTM model: original (unprotected), protected with residue-based CEDs (as proposed here), and protected with double modular redundancy (DMR). Similar to our CEDs, we have implemented two versions of DMR, coarse-grain and fine-grain, for comparison. The coarse-grain DMR duplicates the entire MVM block with a comparator at the end, whereas the fine-grain DMR performs duplication and comparison for each individual PE.

5) FAULT SIMULATION METHODOLOGY

To analyze the impact of faults in LTSM networks and evaluate the proposed protection methods, we developed our fault injection framework in Python using the TensorFlow framework [43]. In this framework, we create a register-accurate model of the LSTM for inference by associating every line of the inference with its respective hardware component. This methodology, that is in line with work [10], allows us to precisely determine the effects of fault injection on the underlying microarchitectural components. Moreover, we have the register-accurate models for the residue-based CEDs and DMRs to integrate them into the network models.

We use a 16-bit fixed-point data type for model parameters, as it proves to be more energy-efficient in hardware-based model executions. This data representation is in line with [23], [25], [26], and [28].

We generated random fault campaigns based on our fault model and injected them into registers of the LSTM models. Due to the time-intensive nature of fault injection experiments, we randomly choose a subset of 100 images from each of the 7,001 test images of MNIST and the 12,901 test images of GTSRB, following prior work [31]. Each fault injection experiment is repeated 1000 times and the average result is reported.

In all fault injection experiments, the standard error bars are calculated at the 95% confidence intervals. All experiments run on an Intel (R) Core (TM) i7-2620M CPU @ 2.70GHz with 16 GB memory and an NVIDIA GeForce RTX 3050 Ti.

B. SIMULATION RESULTS

This part presents the simulation results to answer the above research questions.

RQ1: Fault detection. Initiating the journey toward a dependable DNN system involves the identification of runtime faults. To demonstrate the ability of the CG-CED and FG-CED checkers to detect injected faults, we measure fault coverage, which indicates the proportion of injected faults that are detected by the checker.

Figure 12 illustrates the fault coverage of the proposed CG-CED compared to coarse-grain DMR at various BERs ranging from 9.48E-06 to 1.90E-04 for two different LSTM models. The residue-based CEDs use the check base values of 3, 7, 15, and 31. While fault coverage exhibits a diminishing trend as BER values increase (as expected), the figure shows that for very harsh environments with high BER values, the residue-3 CEDs become inefficient (fault detection rate falls below 80%). This is due to aliasing presented in the smaller check base as discussed in Section V-C.

With the check base values of 7, 15, and 31, CG-CED exhibits similar fault detection rates for lower BERs. As BER increases, larger check base values demonstrate superior fault

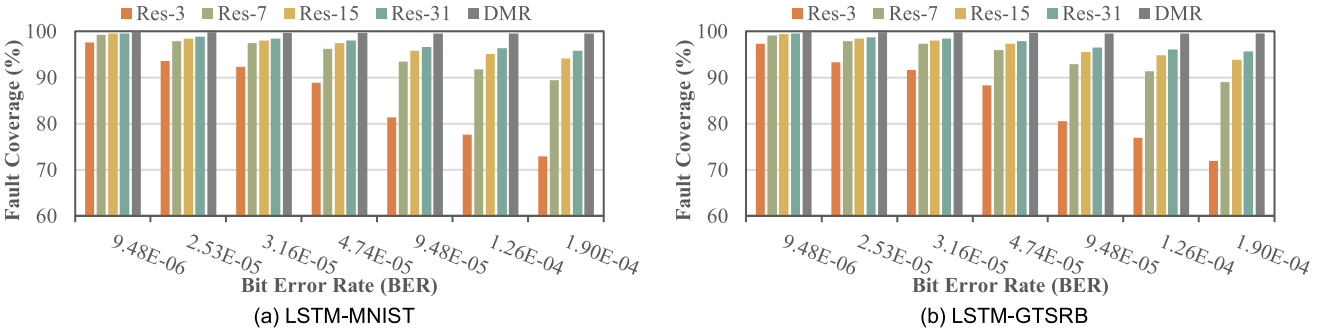


FIGURE 12. Fault coverage of the coarse-grain CED technique at different BERs. For (a) and (b), error bars range from ±0.08% to ±0.21% and from ±0.02% to ±0.18% at the 95% confidence interval, respectively.

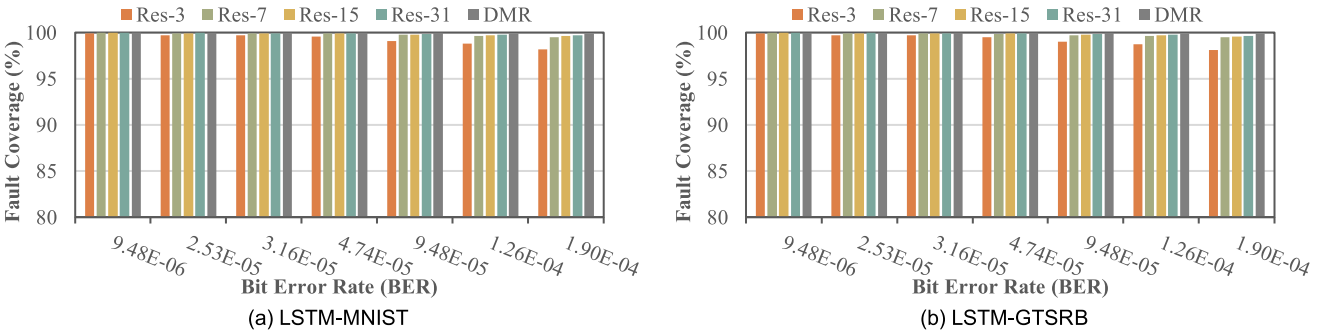


FIGURE 13. Fault coverage of the fine-grain CED technique at different BERs. For (a) and (b), error bars range from ±0.04% to ±0.17% and from ±0.03% to ±0.10% at the 95% confidence interval, respectively.

coverage, which, of course, comes with the price of a larger hardware overhead (discussed in RQ3). As shown, even in high BER values, the residue-7 CEDs are still providing fault coverage at around 90%.

It is notable that the CG-CED solution yields comparable fault coverage rates for both the LSTM-MNIST and LSTM-GTSRB models. This is in contrast with the vulnerability assessment depicted in Figure 1, which shows a greater susceptibility of the LSTM-GTSRB model to random faults.

Figure 13 shows fault coverage for the FG-CED solution compared to fine-grain DMR with various check base values across different BERs for different models. The story is different with fine-grain CEDs that protect each individual PE of the LSTM accelerator.

Figure 13 shows that FG-CED offers significantly higher fault coverage in comparison with CG-CED. We see that even the small check base, i.e., $k=3$, for the harsh error conditions, still provides detection rates above 98%. This is justified because the fine-grain CEDs are applied to units that perform fewer calculations than the coarse-grain CED, and error aliasing is reduced because of that. The difference in fault detection rate between the residue-7 CEDs and CEDs employing larger check base values is negligible, as they all demonstrate nearly complete fault coverage around 100%, similar to DMR.

In a nutshell, CG-CED with a large check base, i.e., k=15 and 31 can provide robust fault coverage (exceeding 95%) for different BERs, while FG-CED for all check base values achieves nearly complete fault coverage similar to DMR.

RQ2: SDC Rate. Upon detection of a fault, the proposed residue-based CEDs enhanced with the Minerva word masking technique [44], mitigate the propagation of fault impact. To show the efficacy of the proposed CEDs in reducing mispredictions and enhancing resilience, we consider the SDC rate as a metric for our evaluation. For DNNs, the SDC rate represents the ratio of mismatch between the classification output of a faulty and the fault-free inference execution [10], [25], [31].

Figure 14 compares the SDC rates of the LSTM models equipped with our FG-CED with those protected by fine-grain DMR and the unprotected ones (original) at various BERs. The results from the LSTM-MNIST model are shown in Figure 14(a). Alongside DMR, FG-CED with zero masking achieves significant SDC reduction for check base values of 3, 7, 15, and 31. It is demonstrated that mispredictions occur when BER goes above 4.74E-05. Prior to this fault rate, mispredictions are effectively reduced to ∼0%. At a high BER of 1.26E-04, the proposed protection mechanism with the check base of 7 (for instance) decreases the SDC rate to 5.8% compared to 90% observed for the unprotected LSTM (15X improvement).

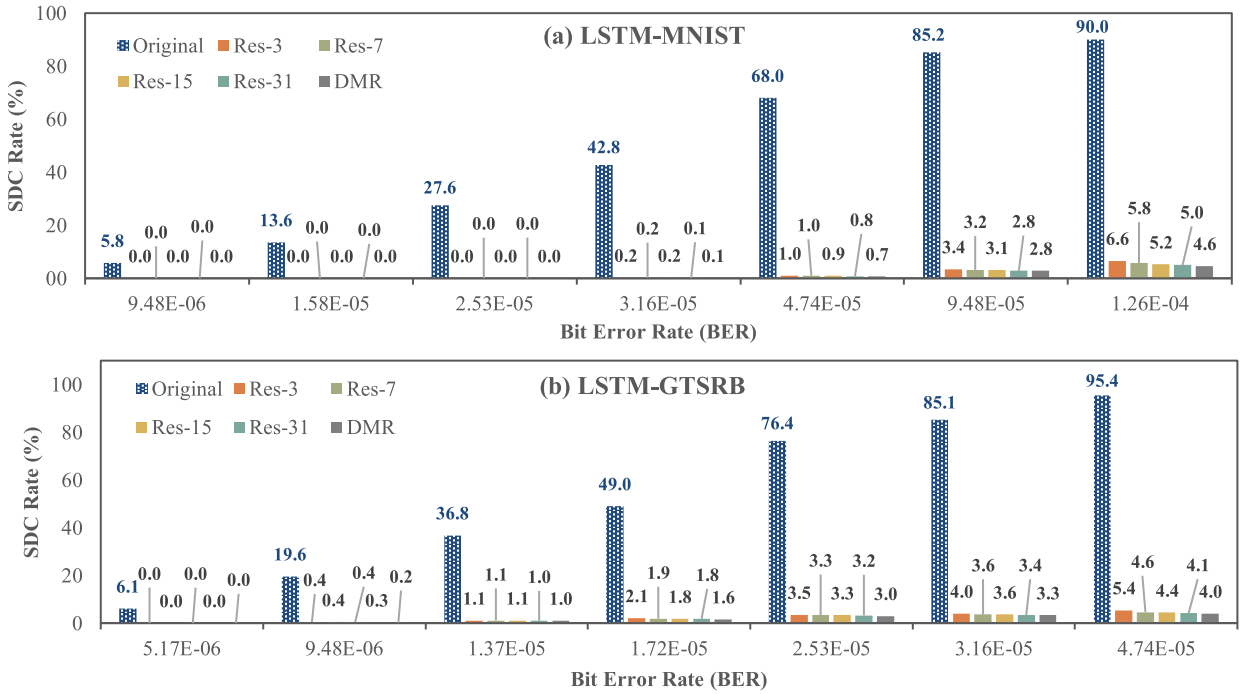


FIGURE 14. SDC rate of the unprotected LSTM and models protected with the fine-grain CED solutions with different check base values across different BERs. For (a) and (b), error bars range from ±0.39% to ±2.71% and from ±0.48% to ±2.03% at the 95% confidence interval, respectively.

A comparable pattern of SDC improvement is apparent in the LSTM-GTSRB network, as depicted in Figure 14(b). The protection generally becomes less effective for the LSTM-GTSRB network, when compared to the LSTM-MNIST. This arises from the fact that the LSTM-GTSRB has 43 output dimensions (depending on the application), thereby elevating the likelihood of falling outside the correct class when applying zero masking to faulty values.

Another takeaway from Figure 14 is that the results indicate no substantial differences between CEDs with smaller and larger check base values. Of course, the larger check base values demonstrate lower SDC rates at high BERs, which stem from the fault detection rate.

The integration of CG-CED with the Minerva masking mechanism [44] fails to eliminate the misprediction caused by transient faults. The reason is that masking will result in nullifying a substantial portion of network calculations. Instead of zero masking, setting the faulty values to their correct/expected values (Oracle) that is employed in [27] can be an effective solution. The assessment of leveraging various masking methods in conjunction with CG-CED is discussed in [45].

To recap, utilizing the FG-CED scheme provides reliability improvement comparable to DMR. FG-CED results in a reduction in SDC rates relative to the unprotected model, ranging from 92% to 100% across various BERs.

RQ3: Protection Cost. The primary cost factors associated with implementing error detection circuitry are design time cost and run time cost, as discussed below.

1. Design time cost: This refers to the effort needed to design the protection mechanism and effectively integrate it into the target system. In terms of hardware, neither coarse-grain nor fine-grain CEDs require a new architectural design. They simply follow the existing architecture of the hardware accelerator. Moreover, our CED solutions are non-intrusive, requiring no changes to the DNN accelerator, and can be easily integrated into the original circuit. This is attributed to the fact that residue codes provide modularity and separability between the original circuitry and the checking circuitry [37].

In terms of software, our approach is agnostic to the DNN architecture, whether during training or outside training unlike the symptom-based methods [25], [26]. Moreover, our method eliminates the need for network retraining, which is an additional burden imposed by [27].

2. Run time cost: The incorporation of error detection mechanisms comes with the price of additional power consumption and silicon area or may influence performance. We examine the run time overheads of our CEDs in relation to area, power, and performance. To conduct these evaluations, we implemented the RTL design of the proposed CEDs in a synthesizable VHDL format and incorporated them into the RTL VHDL description of an LSTM accelerator. For the LSTM accelerator, we considered the n-dimensional DiBA architecture [6] and proceeded to map the LSTM networks onto it. The hardware accelerator and the proposed protection mechanisms are synthesized with the 45 nm technology node using the Nangate open cell library, oper-

TABLE 2. Area and power overheads of the proposed CG-CED and FG-CED schemes in comparison with DMR.

		Area (um^2)	Power (mw)	Area (%)	Power (%)
Original DiBA		49283	9.16	ref	ref
Coarse-grain Checkers	*Res-3*	+10392	+2.21	21.09	24.13
	Res-7	+12248	+3.18	24.85	34.72
	Res-15	+13958	+3.56	28.32	38.86
	Res-31	+15789	+4.05	32.04	44.21
	DMR	+49348	+9.18	100.13	100.17
Fine-grain Checkers	*Res-3*	+20157	+4.42	40.90	48.25
	Res-7	+22221	+5.92	45.09	64.63
	Res-15	+24631	+6.41	49.98	69.98
	Res-31	+26551	+6.74	53.87	73.58
	DMR	+51189	+9.49	103.87	103.56

TABLE 3. Area overhead of the proposed techniques for LSTM accelerators with different architectures and various PE sizes.

Area Overhead (%)							
Configuration †		**1**		**2**		**3**	
HW Acc. Arch.		*1-D*	*2-D*	*1-D*	*2-D*	*1-D*	*2-D*
CG-CED	*Res-3*	21.09	8.35	21.27	5.75	21.53	4.76
	Res-7	24.85	12.59	24.93	9.99	25.15	9.00
	Res-15	28.32	15.59	28.40	12.87	28.62	11.85
	Res-31	32.04	19.61	32.12	16.97	32.34	15.98
FG-CED	*Res-3*	40.90	28.16	40.81	25.29	40.81	23.99
	Res-7	45.09	32.82	44.89	29.95	44.89	28.64
	Res-15	49.98	37.24	49.75	34.23	49.75	32.86
	Res-31	53.87	41.45	53.65	38.51	53.65	37.17

† Configuration 1: # MAC PEs = 24 and # ADD PEs = 5
Configuration 2: # MAC PEs = 120 and # ADD PEs = 11
Configuration 3: # MAC PEs = 512 and # ADD PEs = 19

ating at a frequency of 200 MHz with a supply voltage of 1.1 V.

Table 2 reports the silicon area and power consumption of the DiBA LSTM accelerator, DMR, and the proposed residue-based CED schemes for various check base values. We make two observations from the table. First, the larger check base values result in elevated overheads for both fine-grain and coarse-grain CEDs, as expected. A comparison between the check base values of 3 and 31 provides a clear depiction of this expected trend. As observed, in the case of CG-CED, the area overhead rises from 21.09% to 32.04% and power consumption increases from 24.13% to 44.21%, for 3 and 31 check base values. Further analysis, as shown in Figure 15, indicates that this increase stems from the basic arithmetic circuits rather than the residue generators.

Second, the CG-CED solution entails reduced costs when compared to FG-CED. Taking Residue-7 as an example, the area and power overheads for CG-CED amount to 24.85% and 34.72%, respectively, whereas they reach 45.09% and 64.63% for FG-CED. The reason for that is the comparison process. The coarse-grain approach conducts a single comparison between the outputs of the checker and the original circuity at the end of the MVM computations, whereas the fine-grain CEDs perform the comparison process individually for each PE. As discussed in Section V, each comparison process involves a residue generator for the output of the original circuit along with a modulo comparator.

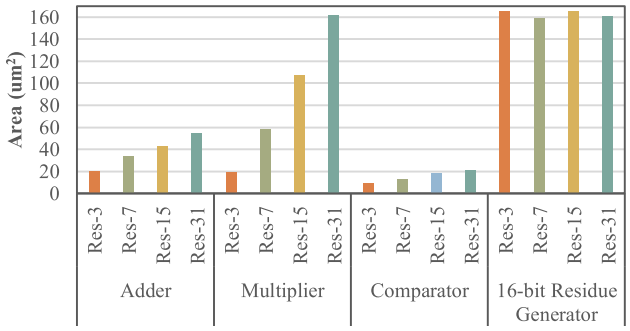


FIGURE 15. Area footprint for residue generator and basic arithmetic circuits mod 3, 7, 15, and 31.

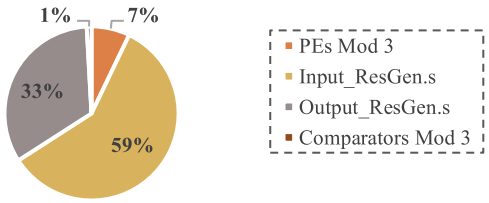


FIGURE 16. Area breakdown of Residue-3 FG-CED with input-side CEDs for all PEs of DiBA.

Figure 16 presents the area breakdown of Residue-3 FG-CED that employs only CEDs with three residue generators as depicted in Figure 7(b), to protect all PEs of the LSTM accelerator. As shown in the figure, the notable area overhead is associated with residue generators on inputs. This highlights the importance of using two distinct types of CEDs for input side and middle PEs (see Figures 7(b) and 7(c)).

As mentioned earlier, our CEDs do not require a new design and simply follow the existing architecture of the hardware accelerator. Table 3 investigates the impact of LSTM accelerator architecture and the number of PEs on the area overhead of the proposed methods. In this investigation, we explore both 1-D architectures, such as those employed by DiBA [6] and LSTM-Sharp [2], and 2-D architectures that use the systolic array style [2]. To facilitate a meaningful comparison, for a 2-D architecture, we examine a typical systolic array with identical PE structures as DiBA.

Two observations can be derived from Table 3. First, utilizing our CEDs in accelerators with 2-D computing arrays results in 13%, 15%, and 17% hardware overhead reduction for configurations 1, 2, and 3, on average, as compared to 1-D architectures. This is because the 2-D architectures use a lower count of costly input-side CEDs. Second, for configurations 1 to 3 as the number of PEs increases, the overhead for 1-D architectures remains relatively constant, whereas it decreases for 2-D architectures.

Regarding the performance analysis, our CEDs do not cause any additional computational cost, in contrast to the ABFT methods [21], [22]. This is attributed to the parallel execution of data checking alongside the original processing. On the other hand, the synthesis results outlined in Table 4 reveal that the inclusion of multiplexers, intended for masking

TABLE 4. Critical path for the DiBA LSTM Accelerator with and without the masking Multiplexers.

	DiBA Acc.	DiBA Acc. with MUXs
Critical Path	3.67 *ns*	3.67 *ns*

faulty values to zero, in the DiBA LSTM accelerator, has no impact on its critical path.

In summary, the area overhead varies between 4.8% and 53.9%, depending on factors such as the granularity of CEDs, check base value, and the accelerator architecture. In the worst-case scenario, our CEDs impose an area and power overhead that is half of the DMR method, with no impact on performance.

RQ4: Efficiency of SCS & TCS. The modularity and separability inherent in our residue-based CEDs enable skipping some operation checks. This provides a trade-off between the accuracy degradation and hardware overhead savings, thus spatial and temporal CED selections. In Figure 17, the SDC rate of LSTM-MNIST is presented for the unprotected model (original), fully protected using the fine-grain CEDs (FG-CED), and selectively protected using SCS (spatial) at various BERs. The full and selective protection methods both use CEDs with a check base of 3. The chart also displays the percentage of hardware overhead saved by SCS (right axis).

In the case of SCS, we assume that 80% of PEs in the LSTM accelerator are safeguarded by Residue-3 CEDs, leaving the remaining 20% of PEs unprotected. This partial protection results in approximately a 10% reduction in protection area overhead as shown in the figure with dotted lines.

Like the full protection, SCS maintains an SDC rate close to 0% for BER values below 2.53E-05. Above this BER, the disparity in the SDC rates between FG-CED and SCS begins to widen. For example, at BER=4.74E-05, the SDC rate is 1% and 3.9% for full and partial protection schemes, respectively. SCS increases the SDC rate by 2.9% compared to full protection, but it still offers a significant SDC reduction of 94.3% relative to the unprotected model. Here, we employed BPTT [32] to characterize the more resilient weights to be mapped onto the unprotected PEs of the LSTM accelerator. Exploring efficient methods as proposed in [37], and [39] can enhance this trade-off.

Figure 18 shows the SDC rate of LSTM-MNIST for the unprotected model, fully protected, and selectively protected with TCS across various heuristic threshold values at a BER of 3.16E-05. The secondary axis shows the power overhead associated with the protection mechanisms. Notably, with larger thresholds (indicating the deactivation of CEDs over an extended duration of time steps), there is a reduction in the TCS power overhead. Simultaneously, the SDC rate experiences an increase, albeit at a slower pace compared to the decrease in power overhead. To exemplify, when the threshold values are set at 8, 11, and 14, power overhead decreases from 48.3% (red dotted line) for full protection to 37.4%, 31.7%, and 26% (green dotted line) for TCS, respectively. This reduction corresponds to an increase in SDC rates from 0.2 for full protection to 1.4%, 5.3%, and 7.7% for TCS at each respective threshold.

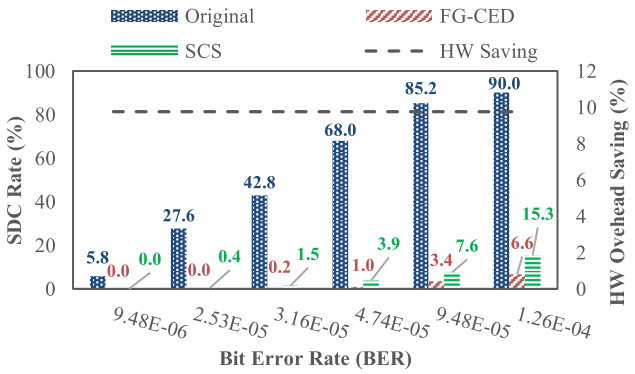


FIGURE 17. SDC rate and hardware saving for SCS with Residue-3 CEDs at various BERs. Error bars range from ±0.51% to ±2.62% at the 95% confidence interval.

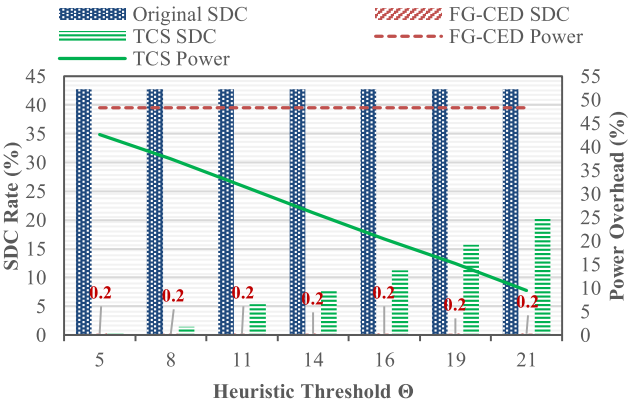


FIGURE 18. SDC rate and power consumption for TCS with Residue-3 CEDs across various thresholds under a BER of 3.16E- 05. Error bars range from ±0.42% to ±2.18% at the 95% confidence interval.

Consequently, SCS and TCS can substantially decrease protection overheads through both static and dynamic decisions, with minimal impact on the SDC rate.

C. COMPARISON WITH PRIOR WORKS

Table 5 displays a comparative analysis between our approach and some existing methods. As can be observed, the selective methods presented in [18] and [19] demonstrate limited effectiveness in reducing SDC. In contrast, the TMR and DMR, despite offering high protection, entail considerable overhead.

The proposed FG-CED achieves SDC coverage ranging from 92% to 100%, marking a significant protection close to DMR. It incurs an overhead ranging from 24% to 54%, which can be acceptable in most safety-critical applications. In addition, SCS and TCS effectively mitigate the protection overhead by employing fine-grain CEDs selectively. SCS can reduce overhead by 10% while providing 84% to 100% SDC coverage for various BERs.

TABLE 5. Comparison with some existing methods.

Technique	*SDC Reduction*	*Overhead*
TMR	100%	200%
Selective TMR† [18]	40%-68%	8%-45%
DMR‡	95%-100%	104%
Selective DMR† [19]	~60%	30%
FG-CED (Ours)	**92%-100%**	**24%-54%**
SCS (Ours)	**84%-100%**	**14%-44%**

†Data are reported from the paper.
‡Implemented in this paper as fine-grain DMR.

D. DISCUSSION

This section discusses additional opportunities and limitations associated with the proposed protection approach.

Opportunities: Although this paper focuses on LSTM networks, the CG-CED, FG-CED, and SCS mechanisms are applicable to feedforward networks such as fully-connected and CNN (Convolutional Neural Networks), as well as other recurrent networks like vanilla RNNs and GRUs (Gated Recurrent Unit). However, the TCS scheme is specifically designed for recurrent neural networks handling time series data. In the future, we plan to employ residue-based CEDs to protect the feedforward networks against bit-flip faults.

Limitations: Our residue-based CEDs are designed for networks with fixed-point data types. Although fixed-point data types are prevalent in the inference phase [6], [7], [23], [25], [26], [28], some systems may use floating-point data types [2]. Such systems cannot be protected with our CEDs, because of complexities in floating-point arithmetic.

VIII. CONCLUSION

This paper presented coarse-grain and fine-grain residue-based CEDs designed to protect the LSTM neural networks against transient faults. The coarse-grain approach is constructed based on the entire MVM structure, whereas fine-grain CEDs are placed on individual PEs. The fine-grain CED scheme exhibited superior fault coverage, nearly reaching 100%, in comparison to the coarse-grain scheme. The complete coverage provided by the fine-grain solution, along with a simple zero-masking mechanism, leads to a reduction in SDC rates from 92% to 100% relative to the unprotected model. This significant protection comes at the cost of an area overhead ranging from 24% to 54%, depending on the LSTM accelerator architecture and problem size. In comparison to the conventional DMR, our fine-grain CEDs offer comparable protection with less than half the overhead.

To further minimize the protection overhead, considering the stringent area and power constraints of DNN accelerators, we introduced two partial protection mechanisms. They employ fine-grain CEDs selectively, guided by the spatial and temporal resilience characteristics of LSTM. Spatial selections are associated with LSTM synaptic weights, while temporal selections are based on LSTM time steps, providing overhead-reliability trade-offs. Experimental results show that partial protection methods provide cost-effective protection when some loss in reliability quality is acceptable.

REFERENCES

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[2] S. Mittal and S. Umesh, "A survey on hardware accelerators and optimization techniques for RNNs," *J. Syst. Archit.*, vol. 112, Jan. 2021, Art. no. 101839.

[3] J. Lee, W. Hong, and P. Hur, "Continuous gait phase estimation using LSTM for robotic transfemoral prosthesis across walking speeds," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 1470–1477, 2021.

[4] S. Kusuma and K. R. Jothi, "ECG signals-based automated diagnosis of congestive heart failure using deep CNN and LSTM architecture," *Biocybern. Biomed. Eng.*, vol. 42, no. 1, pp. 247–257, Jan. 2022.

[5] C. Gómez-Huélamo, M. V. Conde, R. Barea, M. Ocaña, and L. M. Bergasa, "Efficient baselines for motion prediction in autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, early access, 2023.

[6] M. S. Roodsari, M. Ali Saber, and Z. Navabi, "DiBA: N-dimensional bitslice architecture for LSTM implementation," in *Proc. 23rd Int. Symp. Design Diag. Electron. Circuits Syst. (DDECS)*, Apr. 2020, pp. 1–6.

[7] C. Gao, T. Delbruck, and S.-C. Liu, "Spartus: A 9.4 Top/s FPGA-based LSTM accelerator exploiting spatio-temporal sparsity," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 1098–1112, Jun. 2022.

[8] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G.-Y. Wei, "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.

[9] F. Su, C. Liu, and Haralampos-G. Stratigopoulos, "Testability and dependability of AI hardware: Survey, trends, challenges, and perspectives," *IEEE Des. Test. Comput.*, vol. 40, no. 2, pp. 8–58, Apr. 2023.

[10] G. Li, S. Kumar Sastry 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 *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2017, pp. 1–12.

[11] I. Moghaddasi, S. Gorgin, and J.-A. Lee, "Dependable DNN accelerator for safety-critical systems: A review on the aging perspective," *IEEE Access*, vol. 11, pp. 89803–89834, 2023.

[12] S. Mitra, P. Bose, E. Cheng, C.-Y. Cher, H. Cho, R. Joshi, Y. M. Kim, C. R. Lefurgy, Y. Li, K. P. Rodbell, K. Skadron, J. Stathis, and L. Szafaryn, "The resilience wall: Cross-layer solution strategies," in *Proc. Tech. Papers Int. Symp. VLSI Design, Autom. Test*, Apr. 2014, pp. 1–11, doi: 10.1109/VLSI-DAT.2014.6834933.

[13] D. Shin, W. Choi, J. Park, and S. Ghosh, "Sensitivity-based error resilient techniques with heterogeneous multiply–accumulate unit for voltage scalable deep neural network accelerators," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 3, pp. 520–531, Sep. 2019.

[14] M. H. Ahmadilivani, J. Raik, M. Daneshtalab, and A. Kuusik, "Analysis and improvement of resilience for long short-term memory neural networks," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2023, pp. 1–4.

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

[16] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Netw.*, vol. 32, pp. 323–332, Aug. 2012.

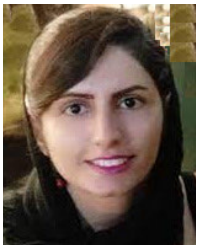
[17] S. Eldridge and A. Joshi, "Exploiting hidden layer modular redundancy for fault-tolerance in neural network accelerators," in *Proc. Boston Area Archit. (BARC) Workshop*, 2015, pp. 1–2.

[18] F. Libano, B. Wilson, J. Anderson, M. J. Wirthlin, C. Cazzaniga, C. Frost, and P. Rech, "Selective hardening for neural networks in FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 66, no. 1, pp. 216–222, Jan. 2019.

[19] A. Mahmoud, S. Kumar Sastry Hari, C. W. Fletcher, S. V. Adve, C. Sakr, N. Shanbhag, P. Molchanov, M. B. Sullivan, T. Tsai, and S. W. Keckler, "HarDNN: Feature map vulnerability evaluation in CNNs," 2020, *arXiv:2002.09786*.

[20] K. Adam, I. I. Mohamed, and Y. Ibrahim, "A selective mitigation technique of soft errors for DNN models used in healthcare applications: DenseNet201 case study," *IEEE Access*, vol. 9, pp. 65803–65823, 2021.

[21] H. Liu, V. Singh, M. Filipiuk, and S. K. S. Hari, "ALBERTA: Algorithm-based error resilience in transformer architectures," 2023, *arXiv:2310.03841*.
[22] K. Zhao, S. Di, S. Li, X. Liang, Y. Zhai, J. Chen, K. Ouyang, F. Cappello, and Z. Chen, "FT-CNN: Algorithm-based fault tolerance for convolutional neural networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1677–1689, Jul. 2021.
[23] S. K. S. Hari, M. B. Sullivan, T. Tsai, and S. W. Keckler, "Making convolutions resilient via algorithm-based error detection techniques," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2546–2558, Jul. 2022.
[24] D. Filippas, N. Margomenos, N. Mitianoudis, C. Nicopoulos, and G. Dimitrakopoulos, "Low-cost online convolution checksum checker," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 2, pp. 201–212, Feb. 2022.
[25] 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.
[26] B. Ghavami, M. Sadati, Z. Fang, and L. Shannon, "FitAct: Error resilient deep neural networks via fine-grained post-trainable activation functions," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2022, pp. 1239–1244.
[27] B. F. Goldstein, V. C. Ferreira, S. Srinivasan, D. Das, A. S. Nery, S. Kundu, and F. M. G. França, "A lightweight error-resiliency mechanism for deep neural networks," in *Proc. 22nd Int. Symp. Quality Electron. Design (ISQED)*, Apr. 2021, pp. 311–316.
[28] B. Feinberg, S. Wang, and E. Ipek, "Making memristive neural network accelerators reliable," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2018, pp. 52–65.
[29] N. Nosrati, S. M. Ghasemi, M. Sadeghipour Roodsari, and Z. Navabi, "Concurrent error detection for LSTM accelerators," in *Proc. IEEE Eur. Test Symp. (ETS)*, May 2022, pp. 1–2.
[30] J. Li, A. S. Rakin, Z. He, D. Fan, and C. Chakrabarti, "RADAR: Run-time adversarial weight attack detection and accuracy recovery," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 790–795.
[31] A. Asgari Khoshouyeh, F. Geissler, S. Qutub, M. Paulitsch, P. J. Nair, and K. Pattabiraman, "Structural coding: A low-cost scheme to protect CNNs from large-granularity memory faults," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2023, pp. 1–17, doi: 10.1145/3581784.3607084.
[32] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.
[33] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," 2015, *arXiv:1506.00019*.
[34] T. M. Breuel, "Benchmarking of LSTM networks," 2015, *arXiv:1508.02774*.
[35] N. Nosrati, K. Basharkhah, R. Sadeghi, and Z. Navabi, "An ESL environment for modeling electrical interconnect faults," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2019, pp. 88–93.
[36] S. J. Piestrak, "Design of residue generators and multioperand modular adders using carry-save adders," *IEEE Trans. Comput.*, vol. 43, no. 1, pp. 68–77, Jan. 1994.
[37] S. J. Piestrak and P. Patronik, "Design of fault-secure transposed FIR filters protected using residue codes," in *Proc. 17th Euromicro Conf. Digit. Syst. Design*, Aug. 2014, pp. 575–582.
[38] C. Schorn, A. Guntoro, and G. Ascheid, "Accurate neuron resilience prediction for a flexible reliability management in neural network accelerators," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 979–984.
[39] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, "ApproxANN: An approximate computing framework for artificial neural network," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2015, pp. 701–706.
[40] C. Sakr, J. Choi, Z. Wang, K. Gopalakrishnan, and N. Shanbhag, "True gradient-based training of deep binary activated neural networks via continuous binarization," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 2346–2350.
[41] D. Neil, J. H. Lee, T. Delbruck, and S.-C. Liu, "Delta networks for optimized recurrent network computation," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2584–2593.
[42] 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.
[43] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement.*, 2016, pp. 265–283.
[44] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 267–278.
[45] N. Nosrati and Z. Navabi, "A low-cost residue-based scheme for error-resiliency of RNN accelerators," in *Proc. 26th Int. Symp. Design Diag. Electron. Circuits Syst. (DDECS)*, May 2023, pp. 83–86.



NOOSHIN NOSRATI (Student Member, IEEE) received the B.S. degree in electrical engineering from the Jundi-Shapur University of Technology at Dezful, Dezful, Iran, in 2014, and the M.S. degree in electronic from Iran University of Science and Technology (IUST), Tehran, Iran, in 2017. She is currently pursuing the Ph.D. degree in digital electronic systems with the University of Tehran. Her current research interests include hardware design and modeling, reliability and fault tolerance in digital systems, and design for test (DFT) and testability of embedded systems.

ZAINALABEDIN NAVABI (Life Senior Member, IEEE) received the B.S. degree from The University of Texas at Austin, Austin, TX, USA, in 1975, and the M.S. and Ph.D. degrees from The University of Arizona, Tucson, AZ, USA, in 1978 and 1981, respectively. He is currently a Professor of electrical and computer engineering with the University of Tehran, Tehran, Iran, and an Adjunct Professor with the Worcester Polytechnic Institute, Worcester, MA, USA. He has authored ten books in various aspects of design and test with hardware description languages (HDLs). He has written numerous articles in HDLs, design automation, and digital system test. His current research interests include high-level design and description methodologies, digital system testing, and design and definition of HDLs.