Journal of Electronic Testing
https://doi.org/10.1007/s10836-026-06227-4

RESEARCH

Error Resilient Transformers: A Novel Soft Error Vulnerability Guided Approach to Error Checking and Suppression

Kwondo Ma[1] · **Chandramouli Amarnath**[2] · **Jackson Isenberg**[3] · **Abhijit Chatterjee**[3]

Received: 14 September 2025 / Accepted: 30 March 2026

Abstract

Transformer networks have achieved outstanding success in Natural Language Processing (NLP) and Computer Vision due to their advanced capabilities. However, their computational demands and extensive data processing make them highly susceptible to soft errors, necessitating the use of robust error resilience techniques. This study introduces a method to enhance the robustness of Transformer models against such errors by selectively placing error detection and suppression modules in the most vulnerable layers of the Transformer network. Beginning with a vulnerability analysis focused on fully connected layers in encoder and decoder architecture, our approach addresses both neuron output and memory access errors. Error detection leverages algorithmic checksums, while deviations are mitigated through targeted suppression techniques, determining output neurons with out-of-range values and suppressing the same to zero or by clamping. Our experiments demonstrate that, for sequence-to-sequence models the like Marian Transformer, this vulnerability-informed suppression method can recover 98.8% of language translation performance from a near-zero BLEU score to over 52 under a bit error rate of 10^{-4}, while incurring minimal runtime and memory overhead.

Keywords Transformer · Error resilience

1 Introduction

Transformer networks [1] are foundational in Natural Language Processing (NLP), offering a solution to the sequential limitations of Recurrent (RNN) and Long Short-Term Memory (LSTM) networks. They have seen widespread adoption in various NLP tasks, including word/sentence classification [2], text generation [3], summarization, and translation [4]. More recently, Transformer architectures have expanded into computer vision, where they have been applied to tasks such as image recognition [5] and object detection [6]. However, the computational demands of Transformers, especially in large-scale models, make them vulnerable to soft errors that can occur in hardware devices. Figure 1 illustrates the vulnerability of Transformer networks to soft errors in a English-to-French translation task, specifically under two types of soft errors: neuron output errors and memory access errors. Using the Marian Transformer model [7], an English input is accurately translated to French under normal conditions, as shown in the green box. However, when neuron output errors are introduced during inference with a bit error rate (BER) of 10^{-5}, the translation meaning shifts from ‘permissions’ to ‘messages,’ altering the intended content. Furthermore, as depicted in Fig. 1b, even a minimal memory access error with a BER as low as 10^{-8} leads to a severely degraded output, highlighting the model’s susceptibility to soft errors.

Historically, Algorithm-Based Fault Tolerance (ABFT) [8] has been instrumental in creating resilient signal processing algorithms. This concept has been adapted in recent

Responsible Editor: Y. Aibin

✉ Kwondo Ma
magunudo@rebellions.ai

Chandramouli Amarnath
chandamarnath@google.com

Jackson Isenberg
jisenberg3@gatech.edu

Abhijit Chatterjee
abhijit.chatterjee@ece.gatech.edu

1 Rebellions Inc., Seongnam-si, Republic of Korea

2 Google, Mountain View, CA, USA

3 School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA

Published online: 13 April 2026

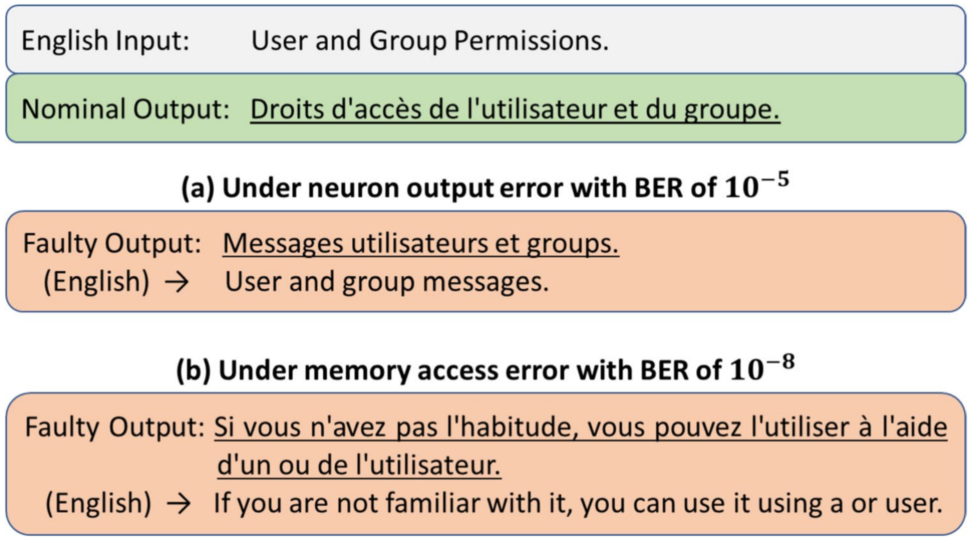


Fig. 1 Example of English-to-French machine translation using an encoder-decoder Transformer under soft errors: (**a**) neuron output error alters the translation meaning, and (**b**) memory access error leads to a nonsensical output

years to address soft error detection in both perceptron-based and convolutional neural networks (CNNs) [9, 10]. To manage errors, one approach involves zeroing erroneous neuron outputs in DNNs [11], although it necessitates modifications to the training process to establish invariant relationships among neurons. A related technique uses statistical analysis to avoid training alterations [12]. Range-restriction methods [13] control neuron output values by using data from error-free circuits; however, they require extensive calculations and are limited to specific layers, which restricts their use in Transformer architectures. Other methods for fault mitigation, such as those applied to systolic array-based accelerators [14] and RRAM-based neural computing systems [15], depend on fault-tolerant training, limiting their adaptability for mitigating soft errors in Transformer architectures.

This study proposes soft error detection and correction strategies for Transformer architectures that (a) do not alter the underlying training algorithms, (b) address both neuron output and memory access errors, and (c) target the most error-prone layers for efficient, rapid recovery. By conducting a comprehensive vulnerability analysis across all fully connected layers in sub-networks of the Transformer network, we selectively deploy error detection and suppression modules in the most error-sensitive layers. Detection is accomplished through algorithmic checksum validation, and suppression involves range-guided droping or clamping neuron output values that fall outside of expected ranges. This submission is a significantly extended version of our earlier work published in the IEEE European Test Symposium (ETS), 2023, titled "Error Resilient Transformers: A Novel Soft Error Vulnerability Guided Approach to Error Checking and Suppression" [19]. *To our knowledge, this work is the first to provide a broad soft error resilience framework for various Transformer architectures, addressing both memory and neuron output errors.* Beyond proposing a detection and correction mechanism, our work evaluates its deployment feasibility across both ASIC-based hardware inference and PyTorch-based software inference platforms. The hardware-centric flow enables architecture-level error mitigation under various error rates and memory bitflips, while the software evaluation verifies resilience and inference correctness in transformer models under random perturbations.

In the following, the state of the art in error resilience of Transformer architectures is discussed. Section 3 presents the architecture and fault model for the Transformer system, followed by a discussion of the proposed error vulnerability analysis, error detection, and suppression techniques in Section 4. Experimental setup and results are presented in Sections 5 and 6. Conclusions are drawn in Section 7.

2 Prior Work

Several recent studies have focused on the resilience of Transformer models, especially Vision Transformers (ViTs), under soft error conditions. These works have primarily concentrated on computer vision tasks, leaving gaps in addressing broader domains such as Natural Language Processing (NLP) and language modeling applications.

Xue et al. [20] investigated soft error vulnerabilities in ViTs using the bit-flip fault model, focusing on Multi-Head Attention and Feed-Forward (FF) layers, and found that linear functions in FF layers are particularly susceptible to soft errors. They proposed a lightweight block-wise algorithm-based fault-tolerance (LB-ABFT) approach for these operations. Similarly, Roquet et al. [21] introduced MaxiMals, a low-overhead fault-tolerance method tailored for ViTs, using neutron beam experiments to identify critical vulnerabilities and applying selective correction via modified identity layers. Liu et al. [22] proposed ALBERTA, which selectively hardens attention mechanisms in auto-encoding Transformer architectures. While these approaches significantly improve resilience in vision-based Transformers, they are largely tailored to ViTs and encoder-based models, and do not directly address generative or auto-regressive Transformer architectures where sequential dependencies amplify error propagation. Finally, Gao et al. [23] analyzed soft error sensitivity in embedding parameters for language models such as BERT and T5. They showed that a small number of critical bit errors can severely degrade performance and proposed selective embedding protection. However, this approach is limited to the embedding layer and does not consider vulnerabilities in attention or feed-forward layers, which dominate computation and error propagation in large Transformer models.

Prior fault-tolerant circuit techniques, such as error-correcting S-box designs for cryptographic applications [24], target small, fixed-function combinational blocks using redundancy and structured codes. Although effective for Boolean circuits, these approaches do not scale to Transformer architectures, which rely on high-dimensional matrix multiplications and attention mechanisms with distinct error propagation characteristics. In contrast, our work introduces vulnerability-guided resilience for fully connected sub-layers in Transformers, employing lightweight checksum-based detection and selective suppression at the architectural level to address reliability challenges in modern AI accelerators and large language models.

Table 1 summarizes a qualitative comparison between the proposed framework and recent state-of-the-art approaches for soft error resilience in Transformer architectures. Recent efforts have begun to address soft error resilience specifically in Transformer architectures. Dai et al. [16] proposed FT-Transformer, which integrates end-to-end fault-tolerant attention (EFTA) using architecture-aware algorithm-based fault tolerance (ABFT) and selective neuron value restriction within fused attention kernels. Similarly, LOST-ViT [17] introduces a low-overhead resilience framework for Vision Transformers using selective bit-level redundancy and vulnerability-aware protection for GEMM and attention operations. In addition, recent work [18] proposes a zero memory overhead approach for protecting Vision Transformer parameters against bit-flip faults by leveraging inherent redundancy without additional storage cost. These approaches primarily focus on kernel-level protection, parameter-level protection, or structured redundancy within specific Transformer variants, as summarized in Table 1. In contrast, our work adopts an architectural-level perspective by performing layer-wise vulnerability analysis across fully connected sub-layers and nonlinear activations and selectively deploying lightweight error detection–suppression modules based on quantified sensitivity. This vulnerability-guided placement avoids retraining, kernel restructuring, or hardware-specific fusion constraints, enabling applicability across encoder-only, decoder-only, and encoder–decoder Transformer architectures under both hardware and software inference settings.

Table 1 Qualitative comparison with recent Transformer soft error resilience methods

Method	Scope	Technique	Retraining	Generality
FT-Transformer [16]	Attention kernels	ABFT + value restriction	Required	Limited (kernel-specific)
LOST-ViT [17]	GEMM + attention	Bit-level redundancy	Not required	Limited (ViT)
Zero-Memory ViT [18]	Model parameters	Parameter protection	Not required	Limited (ViT)
This Work	FC layers & activations	Checksum + suppression (RGD/Clamp)	Not required	Broad (NLP, vision, generative)

The key aspects of the proposed research are:

(1) Soft error vulnerability analysis in Transformer architecture: Memory access errors and neuron output errors in Transformer architectures are modeled using random bitflip injection. It is seen that specific Transformer layers are more vulnerable to soft errors than others and need to be protected aggressively. The most significant bit-targeted error injection method further enhances this analysis by enabling computationally efficient and scalable emulation of critical error patterns with minimal overhead, making it a robust tool for designing error resilience in large-scale Transformer models.

(2) Efficient error detection and suppression: Efficient algorithmic checksums of fully connected layers are used to trigger error suppression on the outputs of the same. Nonlinear activations are checked and corrected on a continual basis. Correction consists of determining neurons with out-of-range values and suppressing the same to zero (this resilience strategy has not been explored in prior research). This is seen to be the most effective strategy for Transformer resilience to both memory access and neuron output errors.

3 Preliminaries and Fault Model

3.1 Transformer Architecture

The Transformer architecture, introduced in [1], features an encoder-decoder framework in which a stack of encoders processes a sequence of input tokens to capture their relationships. The encoder's output is then passed to a stack of decoders, which generates the output sequence one token at a time, using the previously generated tokens as input to predict the next token in the sequence.

3.1.1 Encoder and Decoder

The encoder and decoder networks each consist of a stack of identical layers, incorporating two core sub-layers: *multi-head attention network* and a *feed-forward network*. On the encoder side, the input sequence, tokenized and transformed into embedding vectors, is combined with positional encoding, capturing the relative position of tokens. The two sub-layers produce the output sequence with residual connections and layer normalization applied, as illustrated in the left half of Fig. 2. The decoder network mirrors the encoder's structure, but with an additional multi-head attention mechanism that attends to the encoder outputs. In the decoder, the initial multi-head attention network is *masked* to ensure that predictions at each time depend only on previous tokens, preserving the autoregressive property.

3.1.2 Attention Mechanism

The attention sub-layer in both the encoder and decoder maps input sequences to *queries*, *keys*, and *values*, eliminating the need for recurrent connections. The value vector (V) determines where the model should focus within a sequence, while the query (Q) and key (K) vectors determine the attention weights. These vectors are computed using fully connected (FC) layers as follows:

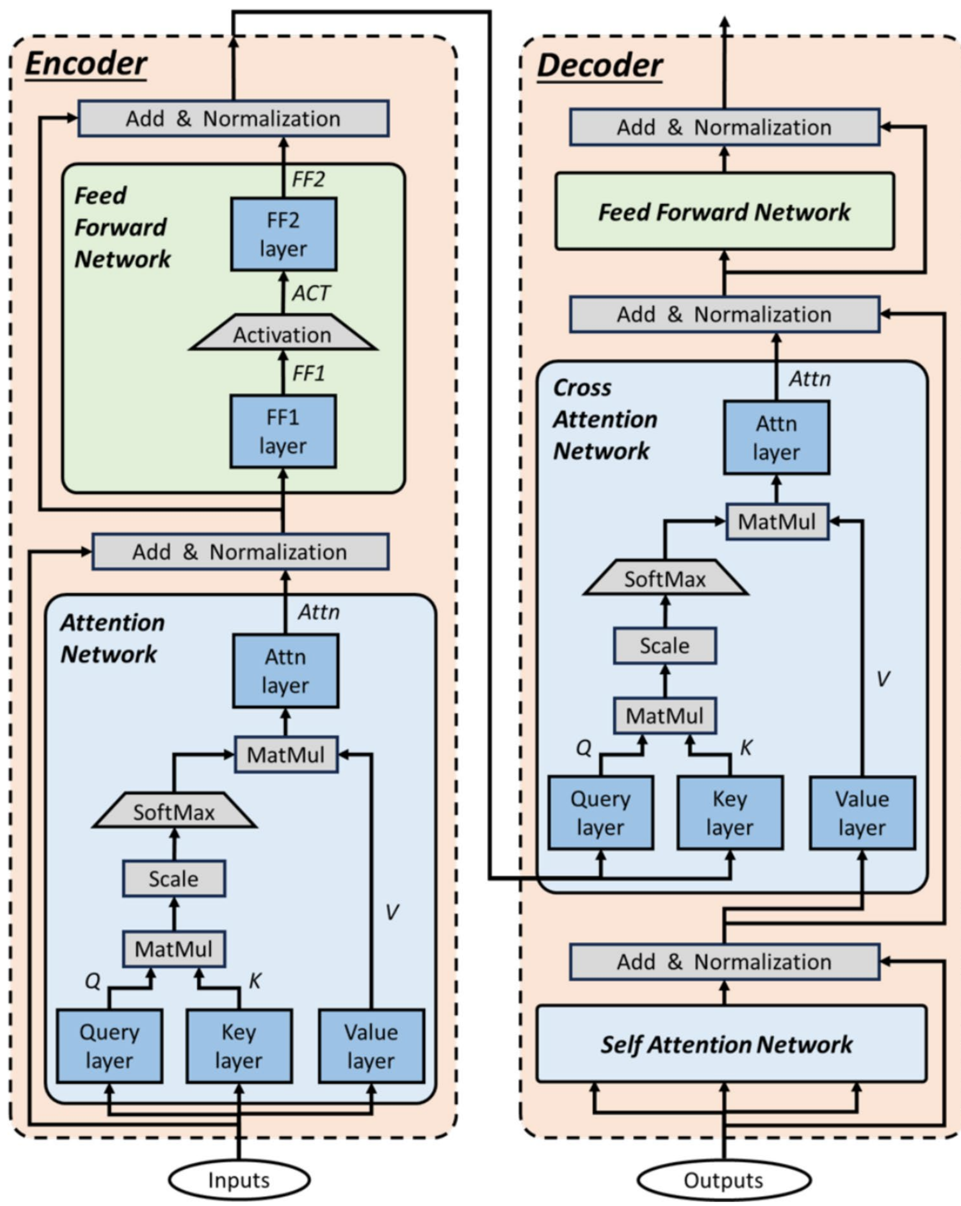


Fig. 2 Architecture of encoder (left) and decoder (right) with attention and feed-forward layers in Transformer model

$$Q/K/V = X \cdot W_{Q/K/V} + B_{Q/K/V} \tag{1}$$

where X is the input, and $W\_$ and $B\_$ are weights and biases specific to the query, key, and value projections. When the same input is used for all three projections, it constitutes *self-attention* (e.g., the first attention block in both encoder and decoder). If different inputs are used for the query and key computations, it becomes *cross-attention* (e.g., the second attention block in the decoder as shown in Fig. 2). The *scaled dot-product (SDP) attention* produces outputs as a weighted sum of values, defined as:

$$Z = SDP(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

where d_k is a dimensionality of the keys, and *softmax*() is the softmax activation function. The original Transformer architecture employs *multi-head* attention by performing multiple parallel attention calculations, then concatenating the results to produce the output sequence. The final attention output is projected via an additional FC layer as $Attn = Z \cdot W_a + B_a$ where W_a and B_a are weight and bias of attention layer.

3.1.3 Feed-Forward Network

The second sub-layer is a feed-forward network, composed of two fully connected layers and an activation function, represented as:

$$FFN(X) = ACT(X \cdot W_{FF1} + B_{FF1}) \cdot W_{FF2} + B_{FF2} \tag{3}$$

where X is the input, and *ACT*() denotes the activation function (e.g., ReLU, GELU). The input and output dimensionality of the feed-forward network is maintained, while the intermediate layer (first FC layer) has usually four times the dimensionality of the input, enabling the model to capture complex transformations efficiently.

3.2 Fault Model

We define a fault model specifically for Transformer architectures to assess their vulnerability to soft errors, also known as hardware transient faults. Transformer models consist of multiple fully connected (FC) layers, activation functions, and layer normalization, all operating within a unique attention mechanism and residual connections. When deployed on hardware such as GPUs, Transformer models are susceptible to soft errors, which can arise from various factors, including external radiation [25], electromagnetic interference [26], and thermal instability [27]. These faults typically manifest as random bit-flips that occur in memory elements and processing engines [28, 29]. Soft errors in Transformer architectures occur in two primary forms during inference: (i) Neuron output errors: Transient errors in the computational units can cause erroneous numerical values of neuron output of FC layers within the Transformer architecture. (ii) Memory access errors: Data access error in main memory or cache, where weight coefficients are stored, can result in random bit-flips in the weights of the corresponding layers.

The bit-flip fault model selects random weights, activations, or dot-product operations in Transformer networks, then flips bits in the binary representation of relevant data exists randomly or in a selective manner. Since Transformer networks rely heavily on floating-point operations, they are particularly vulnerable to soft errors. As opposed to fixed-point representations, floating-point numbers are more vulnerable because errors in the exponent can cause drastic changes in their corresponding numerical values of data [30]. To realistically model these faults, we introduce bit-flips in both the output neuron values and the weight coefficients during the inference operations of the Transformer. This fault model enables us to analyze how soft errors propagate through the Transformer architecture and impact the overall model performance.

4 Proposed Resilience Methodology

This section presents a soft error vulnerability analysis of Transformer network computations. Following this, an efficient error detection and error suppression methods for Transformer architectures are presented. Lastly, we study a selective placement strategy of the proposed error resilience modules for minimizing overhead.

4.1 Soft Error Vulnerability Analysis

Soft error vulnerability analysis is conducted by systematically injecting errors into individual fully connected (FC) layers of Transformer networks. As illustrated in Fig. 3, each layer is evaluated under two soft error types: neuron output errors (blue region) and memory access errors (green region). The analysis begins with the first layer ($L_{\text{current}} = L_1$) and iterates sequentially through all FC layers until $L_c = L_{N+1}$. For neuron output errors, bit flips are injected into the output activations of the current layer L_c independently for each inference, and the erroneous model is evaluated over the entire test dataset. For memory access errors, bit flips are injected once into the weight matrix of L_c prior to inference, after which the model is evaluated on the test set. Model outputs are compared against golden labels to compute relative performance degradation with respect to nominal accuracy. This degradation quantifies the

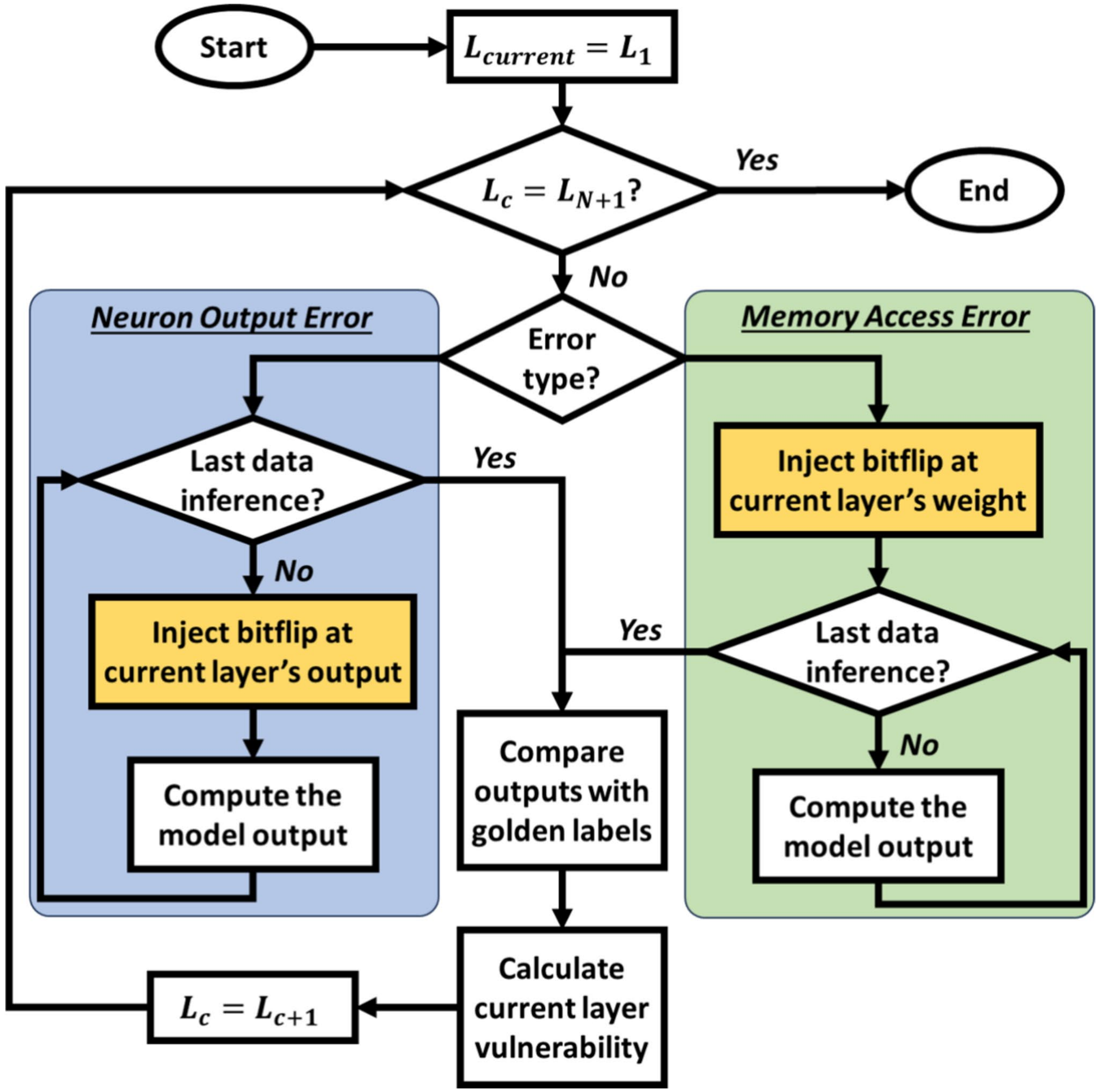
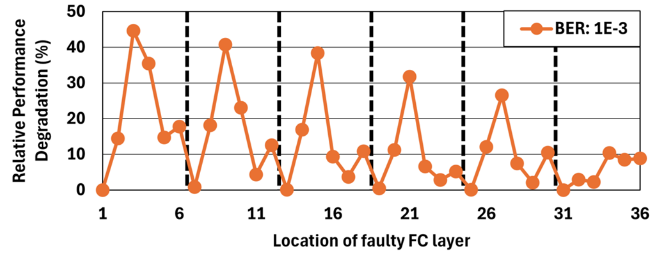


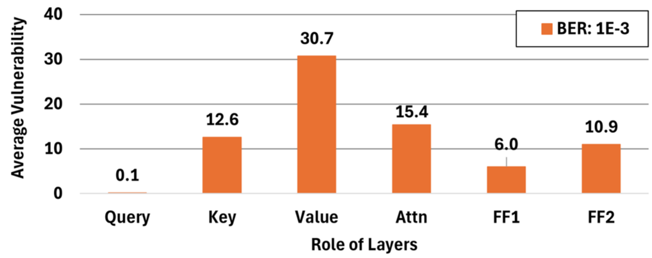
Fig. 3 Flowchart of the soft error vulnerability analysis for Transformer networks

vulnerability of layer L_c. The process then advances to the next layer ($L_c = L_{c+1}$) and continues until all N layers are analyzed.

Figure 4 presents vulnerability results for the text classification task using DistilBERT [31] under neuron output errors with BER = 10^{-3}, where BER denotes the probability of error per output neuron. In Fig. 4a, the x-axis indicates faulty FC layer locations and the y-axis shows relative performance degradation. Vertical segments group the six FC layers within each encoder block. A consistent vulnerability pattern is observed across blocks, with the value layer (third position in each segment) exhibiting the largest degradation. Front-end layers (1–6) show higher average vulnerability (21.15%) than back-end layers (31–36, 5.47%), due to forward error propagation and amplification. Because the vulnerability pattern is uniform across encoder blocks, we derive representative vulnerability values by averaging across blocks. Figure 4b summarizes vulnerability per functional role, showing the value layer as most vulnerable (30.7%) and the query layer as least



(a) Relative performance degradation across fully connected layers



(b) Average vulnerability by functional role of layers

Fig. 4 Soft error vulnerability analysis in the auto-encoding Transformer architecture (DistilBERT)

vulnerable (0.1%). These results guide selective error mitigation placement.

To evaluate vulnerability across error severities, we employ two injection methodologies: (i) *Full-Scale Bit-Level Error Injection (FB-EI)*: Random bit flips are applied across all 32-bit floating-point positions of FC outputs, capturing comprehensive fault effects but with high computational cost. (ii) *Most Significant Bit-Targeted Error Injection (MT-EI)*: Bit flips are restricted to the MSB, which has the greatest impact on floating-point magnitude. MT-EI significantly reduces injection overhead while approximating worst-case behavior. In Fig. 5, we compare the results of the FB-EI with a bit error rate (BER) of 10^{-3} (circle markers) against the results of the MT-EI with the BER of 3×10^{-5} (triangle markers) on same DistilBERT model. The x-axis represents the location of the faulty fully connected layers within the Transformer network, while the y-axis shows the vulnerability of the corresponding erroneous model. The plot demonstrates that despite the significant reduction in the bit error rate (i.e., 33 times lower in MT-EI), the results of MT-EI closely mimic those of FB-EI. The consistency in the results suggests that MT-EI, despite its lower computational cost, effectively captures the impact of soft errors, providing a reliable approximation of the more exhaustive FB-EI approach.

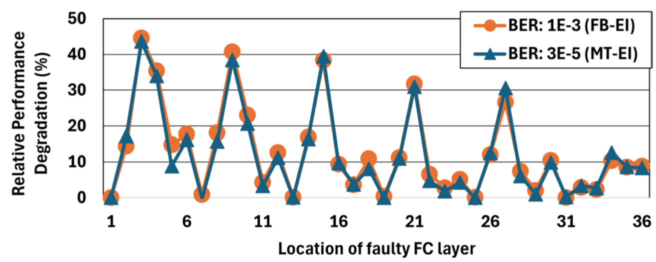


Fig. 5 Comparison of vulnerability analysis in DistilBERT using Full-Scale Bit-Level Error Injection (FB-EI) and MSB-Targeted Error Injection (MT-EI)

4.2 Error Detection

Errors in weights and neuron outputs lead to numerical errors in fully connected (FC) layers, propagating through sub-layers and residual connections, ultimately corrupting the final output of the Transformer networks. To mitigate this error propagation, we place error detection modules after each FC layer within the attention and feed-forward networks, represented by red diamond symbols in Fig. 6a and b. For detecting neuron output errors, we employ *algorithmic checksums*. Given a 2-D input matrix $X \in \mathcal{R}^{l \times d}$, where l is the maximum sequence length and d is the embedding dimension, the output of the n-th FC layer is calculated as:

$$Y_n[i][j] = B_n[j] + \sum_{k=0}^{d-1} X_n[i][k] * W_n[k][j] \tag{4}$$

where B_n, W_n, X_n, and Y_n represent the bias vector, weight matrix, input, and output of the n-th FC layer, respectively.

To verify correctness, we compute a checksum over the output matrix by summing across the embedding dimension: $Y_n^s[i] = \sum_{j=0}^{d-1} Y_n[i][j]$. This produces the output vector Y_n^s, representing row-wise sums for each token's word

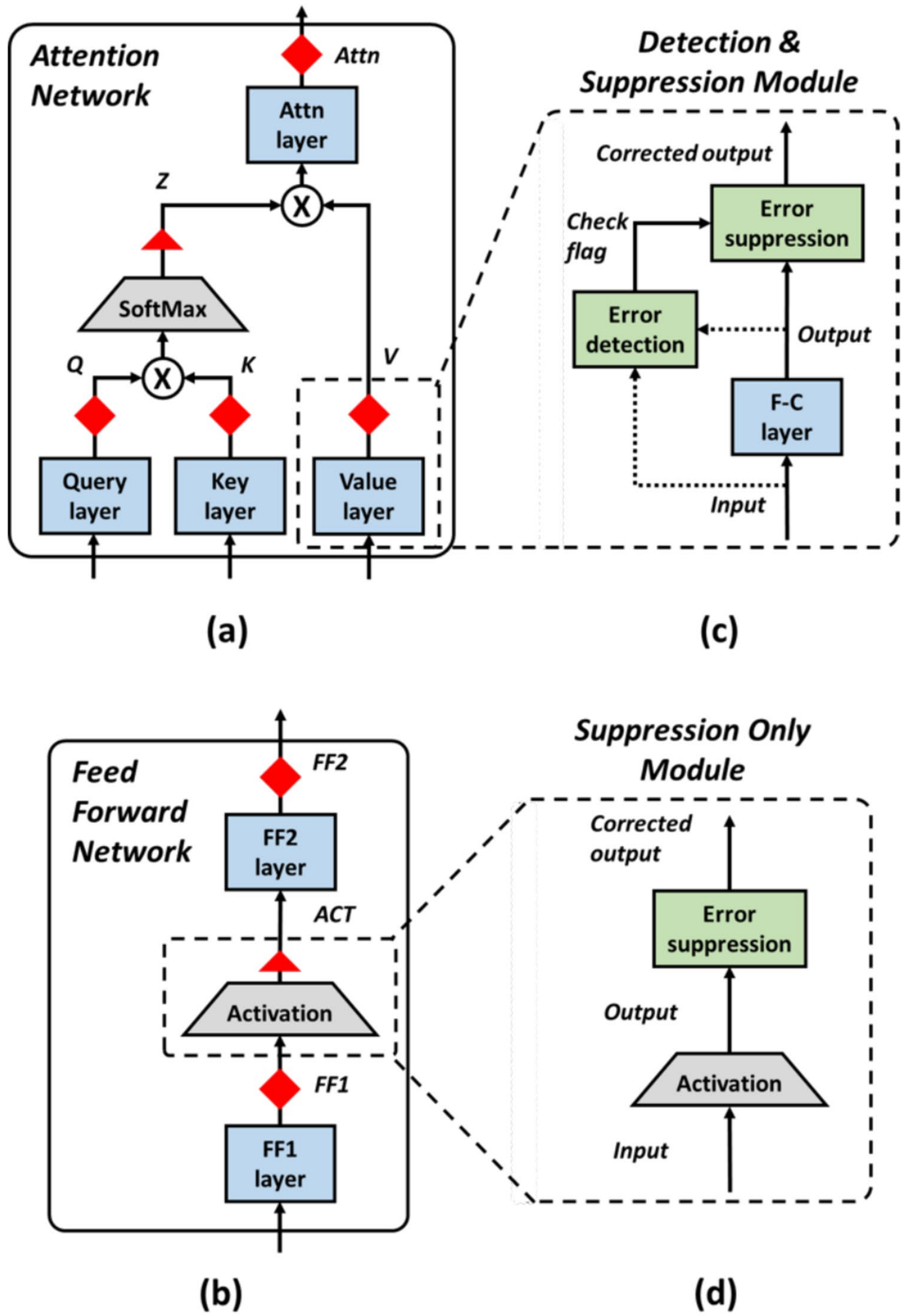


Fig. 6 Placement of error detection and suppression modules in the Transformer architecture: (**a**) Attention network, (**b**) Feed-forward network, (**c**) Architecture of error detection and suppression modules for fully connected layers, and (**d**) Architecture of suppression-only module for activation layers

embeddings. We equally apply a checksum to right side of Eq. 4:

$$
\begin{aligned}
R_n^s[i] &= \sum_{j=0}^{d-1} B_n[j] + \sum_{k=0}^{d-1} X_n[i][k] \sum_{j=0}^{d-1} W_n[k][j] \\
&= B_n^s + \sum_{k=0}^{d-1} X_n[i][k] W_n^s[k]
\end{aligned}
\tag{5}
$$

where B_n^s is the sum of bias elements, and W_n^s is the vector obtained by row-wise summing the weight matrix. In the absence of faults, the norm $||Y_n^s - R_n^s||$ should ideally be zero. During operation, a threshold τ_n is calculated for each FC layer to ensure $||Y_n^s - R_n^s|| < \tau_n$, indicating acceptable bounds for error. The threshold τ_n for each FC layer is calibrated based on statistical analysis of the error-free operation during inference under training dataset, accounting for standard deviations in layer outputs. By setting τ_n based on output statistics, we achieve a balance between false positive and false negative rates. The selection of τ_n impacts the probability $P(||Y_n^s - R_n^s|| < \tau_n)$, where overly stringent thresholds may lead to unnecessary suppression while lenient thresholds could miss actual errors. A probability-based approach allows adaptive thresholding to reduce these occurrences. Figure 6c illustrates the error detection module, where a check flag is computed from each FC layer's inputs and outputs. The summations of bias and weights are stored in memory for each FC layer, facilitating efficient error detection.

4.3 Error Suppression

In Transformer architectures, deviations in neuron output values can have substantial impacts on the attention network due to the scaled dot-product computation and residual connections, which amplify errors across layers. The targeted error suppression technique efficiently prevents runaway error propagation by constraining activations within predefined ranges, thereby stabilizing model performance in attention-sensitive tasks. We propose two strategies to mitigate critical errors that could cause Transformer malfunction by suppressing large deviations in neuron outputs. To determine the suppression thresholds in a principled manner, we perform a one-time nominal inference profiling step prior to fault injection. During this stage, the output distributions of each fully connected (FC) layer are characterized using the evaluation dataset under error-free conditions. The observed statistics are then used to establish optimized operating bounds for each layer. These bounds define the valid numerical range of neuron outputs and serve as reference thresholds for subsequent error suppression. Using these optimized operating bounds, out-of-range outputs are either *clamped* or *dropped*. The clamping method restricts any neuron output that exceeds these bounds, setting it to the derived minimum or maximum operating bounds as follows:

$$Clamp(Y_n) = \begin{cases} y_n^{min} & \text{if } Y_n[i][j] < y_n^{min} \\ y_n^{max} & \text{if } Y_n[i][j] > y_n^{max} \\ Y_n[i][j] & \text{otherwise,} \end{cases} \tag{6}$$

where Y_n represents the output of the n-th layer, and (y_n^{min}, y_n^{max}) denote the layer's operating bounds. Our second strategy, called *Range-Guided Drop (RGD)*, suppresses out-of-range values by setting them to zero:

$$RGD(Y_n) = \begin{cases} Y_n[i][j] & \text{if } Y_n[i][j] \in [y_n^{min}, y_n^{max}] \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

The optimized operating bounds for suppression schemes (y_n^{min}, y_n^{max}) can be derived by minimizing a defined cost function under the inference of training dataset:

$$\mathcal{J} = \sum_{n=1}^{N} ||Y_n - Sup(Y_n)|| \tag{8}$$

where $Sup() \in \{Clamp(), RGD()\}$ and N denotes a number of FC layers in the Transformer. This cost function assesses the impact of suppression on model performance, providing a quantitative basis for determining bounds that minimize performance degradation on nominal operating condition.

The error suppression module is activated based on the check flag from the error detection module, as illustrated in Fig. 6c. For the n-th FC layer, the *Error Detection & Suppression (EDS)* module operates as:

$$EDS(Y_n) = \begin{cases} Y_n & \text{if nominal, } ||Y_n^s - R_n^s|| \le \tau_n \\ Sup(Y_n) & \text{if detected, } ||Y_n^s - R_n^s|| > \tau_n, \end{cases}$$

To further prevent the amplification of minor deviations that may bypass EDS modules during the scaled dot-product attention mechanism or feed-forward network, we also apply suppression to post-activation values. For activation functions, suppression is always enabled, as indicated by red triangle symbols in Fig. 6a and b, and detailed in Fig. 6d. The proposed detection and suppression operations map to simple datapath primitives, including accumulators for checksum computation, comparators for range checking, and multiplexers for conditional value selection. As such, the EDS modules can be integrated as lightweight extensions to existing Transformer datapaths without requiring dedicated encoding/decoding logic or significant pipeline modification.

4.4 Selective Error Detection-Suppression Module Placement

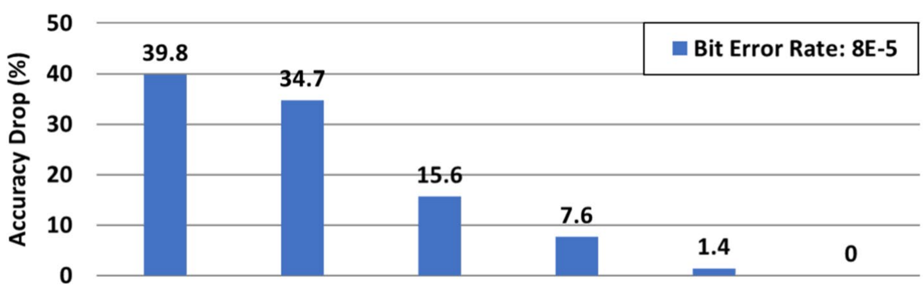
We propose a vulnerability-guided placement strategy for Error Detection-Suppression (EDS) modules that maximizes resilience while minimizing computational and memory overhead. Leveraging the layer-wise sensitivity analysis from Section 4.1, EDS modules are prioritized for layers that provide the largest performance recovery, significantly reducing the search space for optimal placement.

Algorithm 1 Selective EDS module placement.

1: **Input:** Transformer Model $\mathcal{T}$, Bit Error Rate (BER), Layer List $\mathcal{L} = \{L_1, L_2, \ldots, L_n\}$
2: Set Nominal Accuracy $Acc_{nominal} \leftarrow$ Accuracy of $\mathcal{T}$ without errors
3: Set Faulty Accuracy $Acc_{faulty} \leftarrow$ Accuracy of $\mathcal{T}$ under BER
4: Initialize Current Accuracy $Acc \leftarrow Acc_{faulty}$
5: Initialize Remaining Layers $\mathcal{R} \leftarrow \mathcal{L}$
6: Initialize Optimal Module Placement $\mathcal{P} \leftarrow \emptyset$
7: **while** $Acc < Acc_{nominal}$ **do**
8:   **for** each layer $L \in \mathcal{R}$ **do**
9:     Temporarily place EDS module in L
10:     Compute recovery gain $G(L) = Acc(L) - Acc$
11:   **end for**
12:   Select layer $L^* = \arg\max_{L \in \mathcal{R}} G(L)$
13:   Permanently place EDS in L^*, update $\mathcal{P} \leftarrow \mathcal{P} \cup \{L^*\}$
14:   Update Remaining Layers $\mathcal{R} \leftarrow \mathcal{R} \setminus \{L^*\}$
15:   Update Current Accuracy $Acc \leftarrow Acc(L^*)$
16: **end while**
17: **Return** Optimal Module Placement $\mathcal{P}$

Algorithm 1 summarizes the greedy placement framework. The procedure initializes the nominal accuracy $Acc_{nominal}$ (error-free model $\mathcal{T}$), faulty accuracy Acc_{faulty} under a given BER, and the current accuracy Acc which initially set to Acc_{faulty}. The list of all layers $\mathcal{L}$ is also initialized, and an empty set $\mathcal{P}$ is used to track the optimal placement of EDS modules (lines 1-6). In each iteration, EDS modules are temporarily inserted into each remaining layer $L \in \mathcal{R}$, and the recovery gain $G(L) = Acc(L) - Acc$ is computed (lines 8–11). The layer with the highest recovery gain $L^* = \arg\max G(L)$ is then permanently selected (line 12), added to $\mathcal{P}$, removed from $\mathcal{R}$, and Acc is updated (lines 13–15). The process continues until $Acc \geq Acc_{nominal}$, at which point the optimal placement set $\mathcal{P}$ is returned. By iteratively prioritizing the most vulnerable layers for EDS placement, this greedy approach effectively balances the trade-off between resilience and efficiency. This greedy strategy directly aligns EDS deployment with vulnerability patterns, achieving near-nominal recovery with minimal modules and demonstrating scalability across Transformer architectures.

Figure 7 shows a result of selective EDS module placement in an auto-encoding Transformer architecture (DistilBERT). The baseline accuracy of DistilBERT is 92.9%, and the suppression technique used is range-guided drop, which sets out-of-range values to zero. The figure combines a bar chart and table to illustrate accuracy drop percentages and corresponding Transformer performance with different EDS placements under a BER of 8×10^{-5}. The table outlines the incremental recovery achieved by placing EDS modules in various layers, with each column representing model accuracy after EDS module addition in the corresponding layer role. Blue-shaded cells and dashed boxes highlight the optimal module placement order, determined through the greedy approach, where the next best placement is selected based on the largest performance improvement. Column ① in Fig. 7 indicates the performance of the Transformer without any error suppression, showing an accuracy drop of 39.8% in the bar chart (i.e., faulty accuracy without suppression = 53.1%). Column ② in the table demonstrates the accuracy with EDS modules selectively inserted in a single layer (e.g., inserting modules only in the value layer results in an accuracy of 58.2%). Starting with the value layer, inserting EDS modules recovers 5.1% of the accuracy drop, making it the highest-priority layer for the EDS module placement. Following this, adding EDS modules in the key and FF2 layer (columns ③ and ④) further improves accuracy to 77.3% and



	①	②	③	④	⑤	⑥
Query	X	53.1	60.3	78.4	85.9	91.7
Key	X	53.8	77.3	O	O	O
Value	X	58.2	O	O	O	O
Softmax	X	53.2	62.1	77.8	85.4	91.4
Attn	X	53.7	60.5	85	91.5	O
FF1	X	50.8	58.9	84.1	87.9	93.4
Activation	X	50.8	58.9	84.1	87.9	93.4
FF2	X	53.4	61.7	85.3	O	O

Fig. 7 Example of selective EDS module placement in the auto-encoding Transformer architecture (DistilBERT)

85.3%, respectively. These ordering reflects the results of the previous example of vulnerability analysis (see Section 4.1), where the value layer exhibited significant sensitivity to errors and key and FF2 showed moderate vulnerability. Continuing with the prioritized placement, inserting EDS modules into the attention and FF1 layers brings the accuracy to 91.5% and 93.4%, respectively (columns ⑤ and ⑥).

5 Experimental Setup

In this section, we describe the experimental setup and methodology for vulnerability analysis and performance evaluation. All experiments were conducted on a NVIDIA RTX A4000 GPU using 32-bit floating point computation.

Transformer Benchmark Applications Table 2 summarizes the Transformer models and datasets used for evaluation across various tasks. For text classification, we use DistilBERT [31] on the IMDB dataset [33] to measure binary sentiment classification accuracy (positive vs. negative reviews). In image classification, the Vision Transformer (ViT) [5] is evaluated on the CIFAR-10 dataset [34], with classification accuracy as the main metric. Both of these tasks use auto-encoding models (encoder-only Transformers). For word prediction task, we use the auto-regressive GPT-2 model [32], a generative model designed for text generation. Evaluation is conducted on the LAMBADA dataset [35], which tests language models' ability to understand and retain long-range context. The primary metric is token prediction accuracy, measured as the proportion of correctly predicted last words across all test passages. Accuracy on this dataset directly reflects the model's contextual understanding and coherence, as only models that fully comprehend the passage can predict the final word accurately. For neural machine translation (NMT), we use the Marian Transformer [7], a sequence-to-sequence (encoder-decoder) model, to translate English to French on the KDE4 dataset [36]. The BiLingual Evaluation Understudy (BLEU) score is used to assess the quality of the translations against reference texts. Including these diverse models across different Transformer architectures enables a comprehensive evaluation of vulnerability and error resilience across a variety of tasks, from text classification to language translation.

Error Injection Method To evaluate the soft error resilience of Transformer networks, we simulate neuron output errors and memory access errors during the inference. Neuron output errors are introduced by injecting random bitflips into the numerical values of output activations in fully connected (FC) layers. This simulates transient faults in computational units, which can lead to erroneous activations, providing insight into the model's sensitivity to soft errors that affect the output of intermediate layers. For memory access errors, we perturb the pre-trained weight values by injecting random bitflips before the inference begins. These weight errors simulate data corruption and memory access fault in memory elements, such as main memory or cache, where weights are stored. To analyze the model's resilience to weight errors, we generate 100 distinct error configurations by randomly injecting bitflips into the pre-trained weights. For each configuration, the faulty model is evaluated on the test dataset, and the mean and standard deviation of the model performance are calculated across 100 different weight error scenarios.

6 Experimental Results

This section presents detailed vulnerability analyses and evaluates the proposed error resilient Transformer networks under neuron output errors and memory access errors.

6.1 Soft Error Vulnerability Analysis

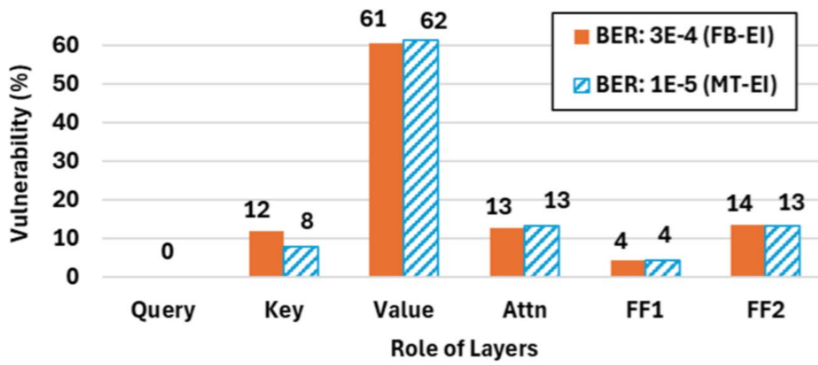
6.1.1 Neuron Output Errors

Figure 8 presents the vulnerability analysis results for neuron output errors in three Transformer architectures:

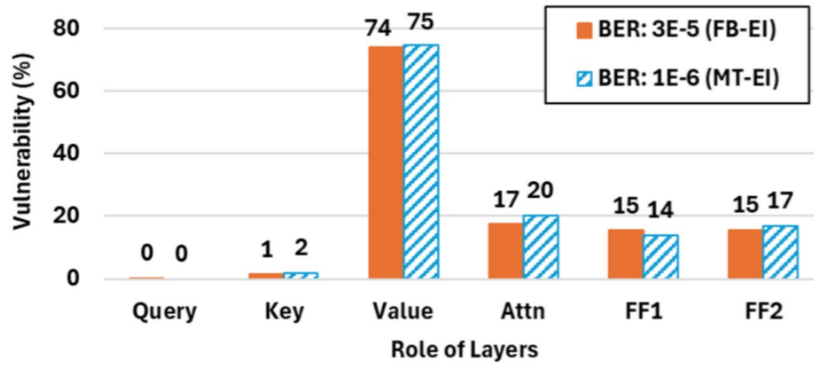
Table 2 Overview of Transformer networks used for evaluation across different architectures, applications, and datasets

Architecture	Auto-encoding Models		Auto-regressive Models	Sequence-to-Sequence Models
	(Encoder-Only Models)		(Decoder-Only Models)	(Encoder-Decoder Models)
Application / Task	Text Classification	Image Classification	Word Prediction	Machine Translation
Model	DistilBERT [31]	Vision Transformer [5]	GPT-2 [32]	Marian [7]
Dataset	Internet Movie Database [33]	CIFAR-10 [34]	LAMBADA [35]	KDE4 (EN-FR)[36]
Data description	Positive and negative texts	General images	Fiction passages	English-French language pair
# of Encoder	6	12	0	6
# of Decoder	0	0	12	6
Evaluation Metric	Classification accuracy	Classification accuracy	Token prediction accuracy	BLEU score

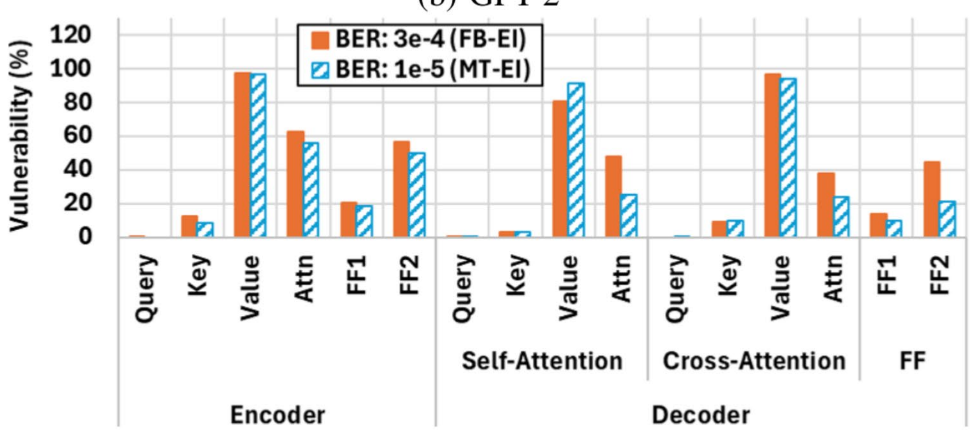
Fig. 8 Vulnerability analysis under neuron output errors, assessed using FB-EI and MT-EI methods



(a) Vision Transformer



(b) GPT-2



(c) Marian

the auto-encoding ViT, the auto-regressive GPT-2, and sequence-to-sequence Marian Transformer. In this analysis, neuron output errors are injected into individual fully connected (FC) layers of encoder or decoder at varying bit error rates, and the performance degradation is averaged based on the functional role of each FC layer. The relative error is defined as the percentage difference between the nominal and perturbed model performance, divided by the nominal performance. Figure 8 shows results obtained from two different error injection methodologies: the Full-Scale Bit-Level Error Injection (FB-EI) method in orange solid columns, and the Most Significant Bit-Targeted Error Injection (MT-EI) method in blue striped columns.

In Fig. 8a, the vulnerability analysis of the Vision Transformer reveals that the value layer is the most susceptible to errors, showing a relative error of 61% under the FB-EI method with a bit error rate (BER) of 3×10^{-4}. This heightened vulnerability is attributed to the value layer's direct influence on the outputs of the scaled dot-product attention mechanism, where errors can cause substantial performance degradation. The key, attention, and FF2 layers exhibit moderate vulnerability, with relative errors between 12% and 14%, while the query and FF1 layers demonstrate minimal vulnerability, with relative errors of 0% and 4%, respectively. This highlights the value layer's critical role in the Vision Transformer's attention mechanism, making it particularly sensitive to soft errors. Conversely, layers like the query and FF1 exhibit higher resilience, likely due to the mitigating effects of subsequent activation functions (i.e., softmax and GLEU), which help reduce the impact of errors.

In Fig. 8b, the vulnerability of the GPT-2 model is depicted, showing that the value layer once again has the highest susceptibility, with a notably high relative error of 74% at a low BER of 3×10^{-5}. This emphasizes the critical role the value layer plays in both the encoder and decoder architectures, as errors here propagate through the attention mechanism, amplifying their impact on the model's output. The attention layer in the decoder also shows moderate vulnerability with a relative error of 17%, followed by the FF1 and FF2 layers with relative errors of 15% each. The query and key layers in the decoder are less impacted, with relative errors of only 0% and 1%, respectively. Unlike in the encoder-only architecture, the key layer in the decoder exhibits significant resilience, while the FF1 layer shows relatively higher vulnerability in the decoder-only model. Overall, the vulnerability analysis across different Transformer architectures (encoder-only and decoder-only) underscores the nonlinear behavior of error propagation within these networks. Understanding these vulnerabilities is crucial for developing targeted resilience strategies, as Transformer architectures demonstrate highly varied error sensitivities based on their layer roles and architectural configurations.

Figure 8c presents the vulnerability analysis for the sequence-to-sequence Transformer model, which comprises both encoder and decoder components. In the encoder block, the value layer again demonstrates the highest vulnerability, with relative errors approaching 100% under a BER of 3×10^{-4}, emphasizing its critical role in the sequence-to-sequence architecture. The attention and FF2 layers exhibit moderate vulnerability, with relative errors nearing 60%. In the decoder, the value layer in both the self-attention and cross-attention mechanisms also emerges as highly vulnerable, with relative errors exceeding 80%. Following the value layer, attention and FF2 layers in decoder block display moderate vulnerability, which is consistent with the results observed in GPT-2 model. In constrast, the query, key and FF1 layers exhibit low vulnerability in both the encoder and decoder blocks, reaffirming their resilience due to the subsequent activation operations that mitigates the impact of injected errors.

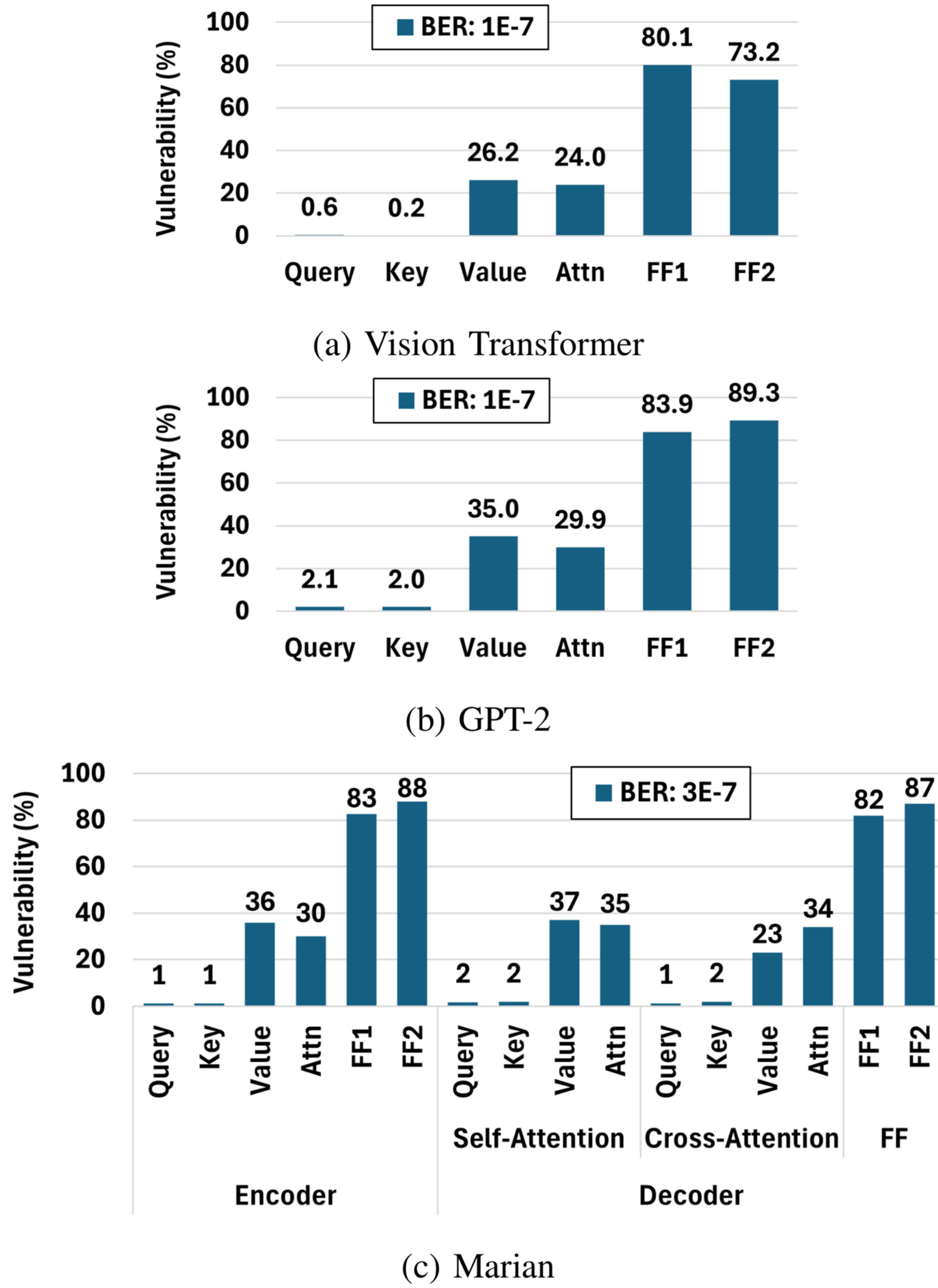
Across the results of soft error vulnerability analysis in Fig. 8, the MT-EI method, despite using a lower error rate (i.e., 30 times lower), yields vulnerability patterns that closely resembling those produced by FB-EI. This further highlights the effectiveness and scalability of MSB-Targeted Error Injection (MT-EI) in approximating full-scale error injection outcomes. Its reduced computational cost makes it especially valuable for assessing error resilience in extremely large and complex models, such as large language models (LLMs), where full-scale error injection would be prohibitively expensive.

6.1.2 Memory Access Errors

Figure 9 illustrates the vulnerability of various Transformer models to memory access errors by injecting random bit-flips into the pre-trained weight matrices under specified bit error rates (BERs). The performance degradation is quantified as the relative error across each layer functional role. In Fig. 9a, the vision Transformer (ViT) exhibits high vulnerability to weight errors in the feed-forward network (FF1 and FF2 layers), with relative errors of 80.1% and 73.2%, under a BER of 10^{-7}. These layers contain significantly larger weight matrices (i.e., 4 times larger matrix size) compared to other layers in the encoder, amplifying the error impact. Errors in the attention and value layers also lead to moderate degradation, with relative errors of 26.2% and 24.0%, respectively. Meanwhile, the query and key layers display minimal sensitivity to weight errors, likely due to the softmax activation function in the attention mechanism that mitigates error propagation in these components. In the auto-regressive GPT-2 model as shown in Fig. 9b, the FF1 and FF2 layers in the decoder are again the most vulnerable, showing relative errors of 83.9% and 89.3%, respectively. The attention and value layers follow, with relative errors of 35.0% and 29.9%, while the query and key layers maintain strong resilience, with relative errors of only around 2%. The higher vulnerability of FF1 and FF2 layers in GPT-2 is consistent with the results observed in ViT, reaffirming the critical role of the feed-forward layers in model performance and stability for both encoder and decoder Transformer architecture.

For the sequence-to-sequence model in Fig. 9c, which comprises both encoder and decoder components, the vulnerability analysis indicates that the feed-forward networks in both the encoder and decoder blocks (FF1 and FF2 layers) are once again the most susceptible to memory access errors. The FF2 layer exhibits the highest relative error, around 88%, with the FF1 layer following closely at approximately 83% under the BER of 3×10^{-7}. In the Marian model's self-attention and cross-attention mechanisms, the value and attention layers also show moderate

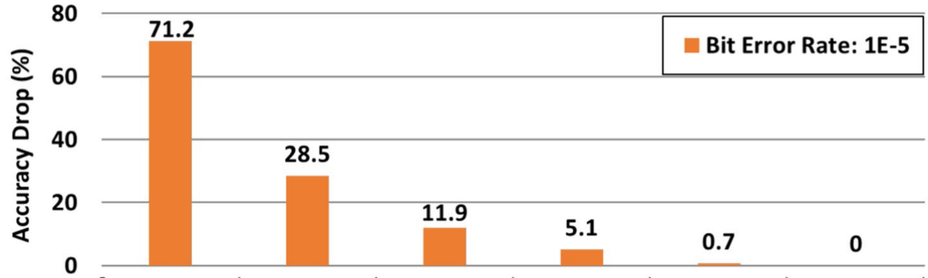
Fig. 9 Vulnerability analysis under memory access errors



vulnerability, while the query and key layers remain robust against weight errors. Overall, across all tested Transformer architectures, feed-forward layers demonstrate a consistent pattern of high vulnerability to memory access errors, while query and key layers exhibit substantial resilience. These findings highlight the importance of error mitigation strategies specifically tailored for feed-forward layers in Transformer architectures. The error-mitigating effect of softmax function, especially on the query and key layers, underscores the non-linear behavior of weight error propagation and points to possible avenues for optimizing error resilience in Transformer architectures through strategic placement of error detection and suppression modules.

6.2 Selective Error Detection-Suppression Module Placement

Following the same principles used for the DistilBERT in Section 4.4, we applied the selective EDS module placement methodology to an auto-encoding model architecture (Vision Transformer), as shown in Fig. 10. The baseline (nominal) accuracy of the Vision Transformer on CIFAR-10



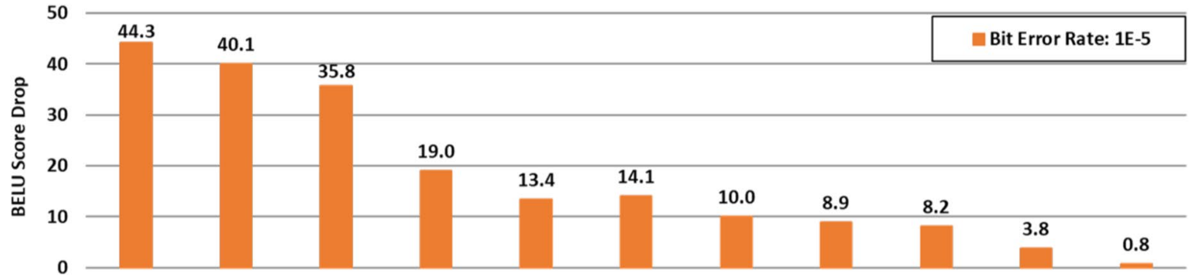
	①	②	③	④	⑤	⑥
Query	X	28.9	70.9	87.3	94.1	98.9
Key	X	42.8	87.4	O	O	O
Value	X	70.8	O	O	O	O
Softmax	X	36.4	78.9	87.6	94.6	99.1
Attn	X	38.4	77.5	91	98.6	O
FF1	X	31.3	74.7	90.1	95.9	99.3
Activation	X	30.9	74.7	90.1	95.9	99.3
FF2	X	37.6	82.1	94.2	O	O

Fig. 10 Selective EDS module placement in the auto-encoding model (ViT) under neuron output errors

is 99.3%, and the suppression technique used is range-guided drop. This result combines a bar chart and a table to illustrate accuracy drop percentages and corresponding Transformer performance with each EDS placements under an error rate of 10^{-5}. The classification accuracy of the ViT without any error suppression shows an accuracy drop of 71.2% in the bar chart (i.e., faulty accuracy without suppression = 28.1%) The selective EDS module placement began with identifying the value layer, which showed the highest vulnerability to neuron output errors. Inserting EDS module in this layer alone recovered a significant portion of the performance, reducing the accuracy drop to 28.5% and raising the overall model accuracy to 70.8%. This mirrors the findings from the vulnerability analysis, where the value layer exhibited the highest sensitivity to soft errors in ViT architecture as shown in Fig. 8a. The next step involved placing EDS modules in the key and FF2 layers, which were identified as moderately vulnerable in vulnerability analysis. These placements further improved accuracy, with the model reaching 94.2% accuracy (5.1% of accuracy drop) after EDS modules were inserted in the FF2 layer (currently, EDS modules are inserted in value, key, and FF2 layers). Continuing with the prioritized placement, inserting EDS modules into the attention and FF1 layers brings down the accuracy drop to 0.7% and 0%, respectively, perfectly restoring the Vision Transformer to its nominal accuracy. The priority sequence derived from this analysis is: Value >> Key > FF2 > Attn > FF1 (=Activation), aligning with the result derived from DistilBERT model.

Figure 11 illustrates the results of selective EDS module placement in the Marian Transformer, which includes both encoder and decoder blocks. The nominal BLEU score for the model on the translation task is 49.0, while the faulty BLEU score under neuron output errors with BER of 10^{-5} drops significantly to 4.8, indicating severe degradation. The figure combines a bar chart showing BLEU score drops and a table detailing the incremental recovery achieved with EDS module placement across different layers in both the encoder and decoder. Column ① in the bar chart represents the performance of the model without any suppression, showing a BLEU score drop of 44.3. The first step in the placement process involves adding EDS modules to the value layers of both the encoder and decoder, which improves the BLEU score to 30.0 (columns ② to ④), making these the most critical initial placements. Inserting EDS modules in the key layer of the decoder (column ⑤) provides additional improvement, raising the BLEU score to 35.6. As an exception, two EDS modules are placed simultaneously in the softmax and FF1 layers of the decoder at column ⑥ due to their minor contributions to performance improvement. The placement sequence aligns with the findings from the previous vulnerability analysis in Fig. 8c, which suggest that adding EDS modules to the key, attention, and FF2 layers of the encoder (columns ⑦, ⑧, and ⑨) further boosts the BLEU score to 40.8, approaching a translation quality close to human level. Finally, placing 12 EDS modules out of a total of 19 potential locations (mainly in FC layers) achieves a BLEU score of 48.2, reducing the BLEU score drop to just 0.8 (columns ⑩ and ⑪).

For this complex Transformer architecture, which includes both encoder and decoder blocks, a clear prioritized placement sequence is not as evident as in the ViT case.



		①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪
Encoder	Query	X	5.3	7.9	14.9	30.4	28.8	30.7	35.5	38.1	39.7	46.7
	Key	X	4.7	7.1	11.9	31.9	29.0	31.9	40.1	O	O	O
	Value	X	5.1	13.2	O	O	O	O	O	O	O	O
	Softmax	X	5.7	6.4	11.4	24.3	24.6	30.3	31.5	38.8	39.3	46.7
	Attn	X	5.2	11.1	15.4	28.7	26.6	39.0	O	O	O	O
	FF1	X	4.3	6.9	13.1	23.9	27.5	28.0	36.6	34.9	40.2	45.3
	FF2	X	3.5	9.9	10.9	28.3	31.2	33.7	38.2	40.8	O	O
Decoder Self-Attention	Query	X	4.4	5.9	13.4	28.2	26.2	35.7	35.7	34.8	42.7	O
	Key	X	5.1	8.5	13.5	27.9	21.8	24.9	34.8	35.2	42.3	40.9
	Value	X	4.1	9.0	30.0	O	O	O	O	O	O	O
	Softmax	X	4.7	8.9	13.2	30.0	35.6	O	O	O	O	O
	Attn	X	5.1	6.0	23.3	23.6	31.9	29.7	34.5	37.7	42.0	44.0
Decoder Cross-Attention	Query	X	4.7	7.1	15.3	28.0	26.3	31	32.6	28.2	40.1	37.3
	Key	X	3.2	8.5	13.0	35.6	O	O	O	O	O	O
	Value	X	8.9	O	O	O	O	O	O	O	O	O
	Softmax	X	4.8	6.4	13.2	28.1	27.3	29.9	36.7	31.1	39.3	43.3
	Attn	X	8.1	7.0	13.6	29.3	28.2	30.1	37.7	37.4	39.7	48.2
Decoder	FF1	X	5.9	7.3	14.2	31.2	34.9	O	O	O	O	O
	FF2	X	6.2	5.9	12.0	32.3	27.7	34.3	38.1	37.8	42.7	O

Fig. 11 Selective EDS module placement in sequence-to-sequence model (Marian Transformer) under neuron output errors

However, it is consistently observed that placing EDS modules in value layers results in higher recovery gains than in any other layers, confirming the critical importance of these layers across both encoder and decoder architecture. This strategy not only confirms the importance of addressing the most vulnerable layers first but also highlights the interplay between encoder and decoder components in recovering translation quality. The placement decisions reflect the relative impact of different layers on performance degradation, showcasing the adaptability of selective EDS module placement for various Transformer architectures.

6.3 Performance Evaluation

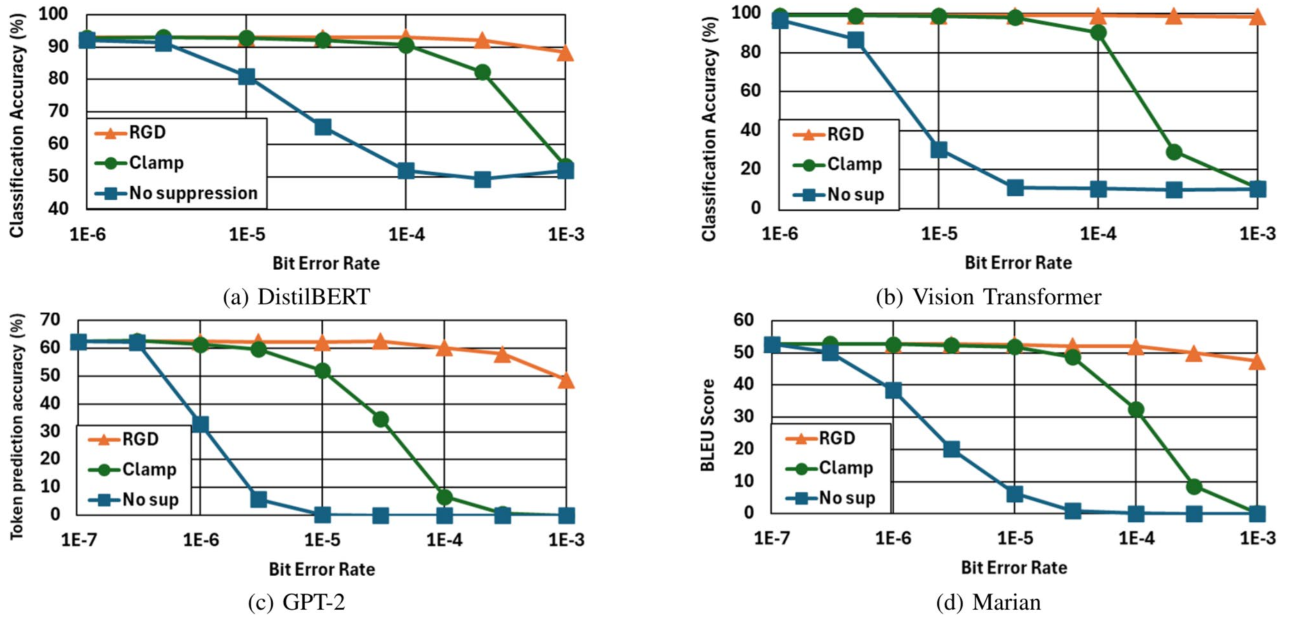
We evaluate the performance of various Transformer networks under neuron output errors and memory access errors, comparing their baseline performance (without error resilience) to two suppression techniques: clamp and range-guided drop (RGD). These error suppression methods are evaluated across varying bit error rates to assess their effectiveness in mitigating performance degradation. To quantify the suppression efficacy, we define 'Recovery' as the percentage of performance degradation that is mitigated by each suppression:

$$\text{Recovery} = \frac{\text{Mitigated Perf} - \text{Faulty Perf}}{\text{Nominal Perf} - \text{Faulty Perf}} \times 100 \tag{9}$$

where 'Perf' refers to the performance metric for each network (e.g., accuracy or BLEU score).

6.3.1 Neuron Output Errors

Figure 12 presents the evaluation results for neuron output errors across various Transformer benchmark applications: DistilBERT, Vision Transformer, GPT-2, and Marian. All models experience a decline in performance as a bit error rate (BER) increases, ranging from 10^{-7} to 10^{-3}. Without any suppression (represented by square markers), each model exhibits natural performance degradation, with drastic drops at higher BER. The classification accuracy of DistilBERT falls to 50% under the BER of 10^{-4}, indicating a



(a) DistilBERT

(b) Vision Transformer

(c) GPT-2

(d) Marian

Fig. 12 Performance evaluation of various benchmark Transformer models under neuron output errors, comparing error suppression techniques: Range-Guided Drop (RGD), clamping, and no suppression baseline across different bit error rates

complete model collapse in binary text classification. Similarly, Vision and Marian Transformers demonstrate sharp declines in performance at the BER of 10^{-5}, with ViT's classification accuracy dropping below 30% and Marian's BLEU score for translation falling below 10, both indicating substantial performance deterioration. GPT-2 shows notable vulnerability beginning at the BER of 10^{-6}, where token prediction accuracy is halved, and it quickly plunges to 0.2% at the BER of 10^{-5}, indicating a severe decline in its ability to generate coherent language.

The two suppression techniques, range-guided drop (triangle markers) and clamping (circle markers), demonstrate varying degrees of effectiveness across these Transformer architectures. As shown in Fig. 12, the range-guided drop (RGD) consistantly outperforms the clamping technique by maintaining model performance and stability at higher bit error rates. In the case of auto-encoding Transformers shown in Fig. 12a and b, average recovery under clamping remains high (92.5%) up to the BER of 10^{-4}; however, RGD achieves an almost perfect recovery of 99.9%. Under severe error conditions with the BER of 10^{-3}, clamping fails to provide meaningful recovery (1.9%), whereas RGD still maintains a robust performance recovery of 94.2%. For GPT-2 in Fig. 12c, RGD sustains token prediction accuracy close to the baseline across a range of error rates up to 10^{-4}, while clamping only achieves 6.7% recovery. Similarly, for Marian Transformer in Fig. 12d, RGD maintains BLEU scores near the nominal level at the BER of 10^{-4}, whereas the clamping scheme quickly deteriorates, reducing the BLEU score to around 30. In constrast, RGD effectively mitigates performance degradation in these auto-regressive and sequence-to-sequence models, achieving 97.6% average recovery at the BER of 10^{-4} and 84% at the BER of 10^{-3}.

In addition, we also evaluated the impact of neuron output errors for a large, contemporary state-of-the-art transformer through Llama 3.1-8B [37] evaluated on the BoolQ dataset [38]. The nominal accuracy of the model is 75.5%. As shown in Fig. 13, we see that RGD degrades gracefully across all fault rates up to 10^{-6} with an average recovery of 75.9%, while clamping only achieves 23.6% average recovery. We also consider a software-level ABFT with selective

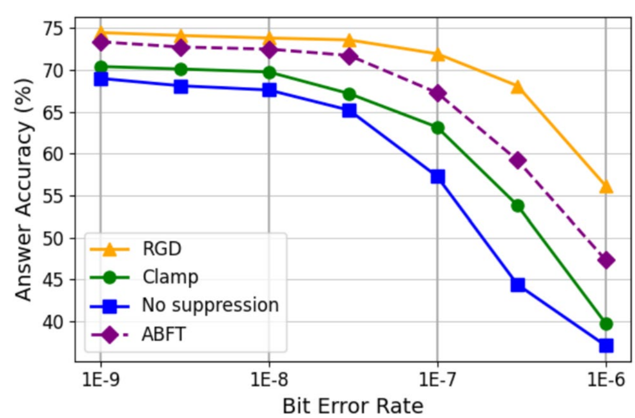


Fig. 13 Performance of Llama 3.1-8B on BoolQ under neuron output errors, comparing fault tolerance techniques: Range-Guided Drop (RGD), output clamping, algorithm-based fault tolerance (ABFT) with selective recomputation, and an uncorrected baseline across varying bit error rates

recomputation as a point of comparison, which achieves 54.8% average recovery, outperforming clamping but falling short of RGD, as the recomputation itself is subject to the same transient bit-flip errors as the original computation, reducing its effectiveness as an increasing function of the bit error rate.

6.3.2 Memory Access Errors

Figure 14 illustrates the impact of weight errors across various Transformer models under different bit error rates (BER), ranging from 10^{-9} to 10^{-6}. As the BER increases, all models exhibit a noticeable decrease in performance without any suppression technique (indicated by square markers). In Fig. 14a, the classification accuracy for DistilBERT begins to decline rapidly at the BER of 10^{-8}, falling below 60% at the BER of 10^{-7} and further to 50% at a BER of 3×10^{-7}. The impacts of memory access errors become evident even at lower BERs in ViT, GPT-2 and Marian Transformers. At the BER of 10^{-7}, ViT shows a drastic drop to 12.1% classification accuracy, indicating a near-complete failure in the 10-class image classification task. Similarly, GPT-2 and Marian present token prediction accuracy of 0.02% and BLEU score of 1.07, respectively, signaling considerable vulnerability in both language generation and translation tasks. These steep declines suggest that memory access errors are significantly more critical and sensitive than neuron output errors. This heightened sensitivity is likely due to the cascading effect of a single erroneous weight, which can propagate hundreds or even thousands of erroneous output values in fully connected layers through matrix multiplication operations.

Under memory access errors, Range-Guided Drop (RGD) consistently achieves superior performance recovery across all Transformer architectures over the entire range of bit error rates. In DistilBERT and ViT, RGD recovers around 93% of classification accuracy at a BER of 10^{-7}, whereas clamping only restores accuracy to approximately 40%. For GPT-2, RGD effectively preserves word prediction accuracy, maintaining it close to nominal performance of 60% even at the BER of 3×10^{-8}, whereas clamping fails to provide sufficient recovery, reducing accuracy to roughly 30%. The auto-regressive GPT-2 model is particularly susceptible to weight errors, with clamping and RGD recovering 13% and 64% of word prediction accuracy at the BER of 10^{-7}, respectively. In Marian, RGD proves especially effective, achieving BLEU score recovery close to 99% up to the BER of 10^{-7}, while clamping mitigates only around 38% of the translation quality. Furthermore, at the highest BER of 10^{-6}, the Marian Transformer with RGD still maintains a BLEU score above 35, which provides relatively decent translation quality comparable to human standards (recovery = 72%). These results underscore RGD's advantage in preserving model performance, especially in memory-intensive sequence-to-sequence Transformer architectures.

From a mathematical perspective, the superiority of Range-Guided Detection (RGD) over clamping can be understood through the attention formulation in Eq. 2. Soft errors in neuron outputs perturb the dot-product, and due to the exponential sensitivity of the Softmax function, even small deviations can significantly distort the result

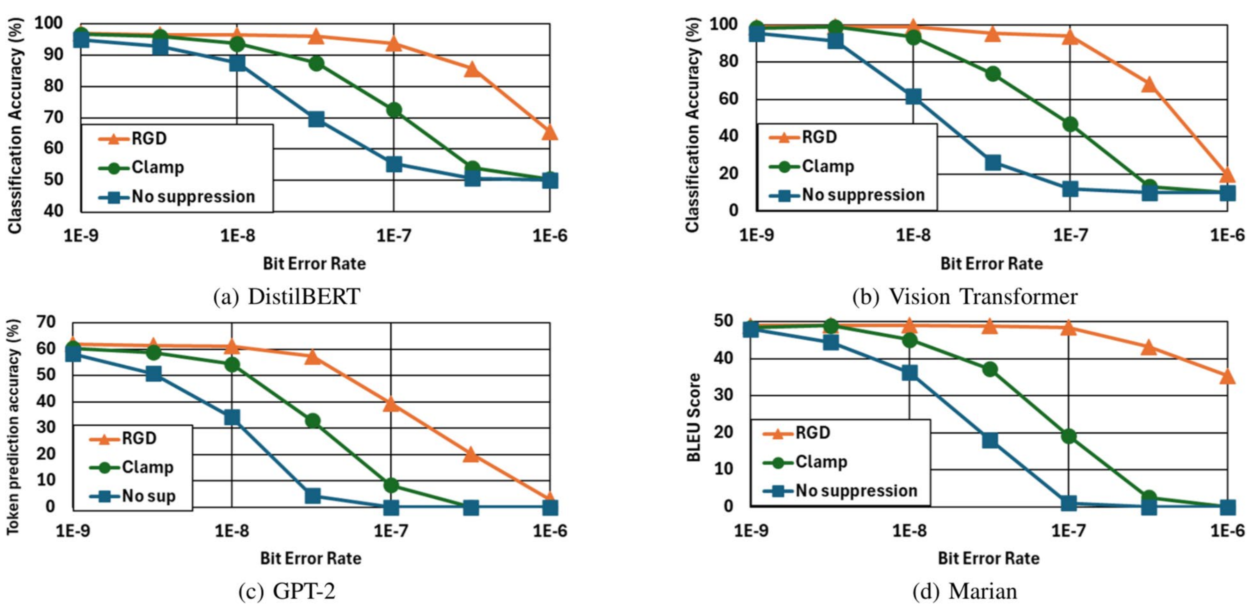


Fig. 14 Performance evaluation of various benchmark Transformer models under memory access errors, comparing error suppression techniques: Range-Guided Drop (RGD), clamping, and no suppression baseline across different bit error rates

of attention. Fixed clamping applies a uniform saturation threshold that may disrupt the relative ordering of logits, leading to attention misallocation. In contrast, RGD constrains activations according to statistically derived nominal ranges, preserving relative logit distribution.

6.4 Overhead Analysis

Table 3 provides a comparative overview of the runtime and memory overhead associated with the three suppression scenarios—No Suppression (Baseline), Clamp, and Range-Guided Drop (RGD)—across various Transformer networks, including DistilBERT, Vision Transformer, GPT-2, and Marian. The table evaluates both runtime (in seconds) and memory usage (in GB), while also showing the percentage overhead introduced by each suppression method relative to the baseline model without suppression. For each model and suppression scheme, these metrics are measured using NVIDIA Nsight systems under the neuron output error injection (with BER ranging from 10^{-4} to 10^{-5}) across 1000 test inferences.

Runtime refers to the total time taken for the execution of a process when working with high-performance computing tasks. In the context of GPU-based operations, runtime encompasses all CUDA API calls, including memory transfers (e.g., cudaMemcpyAsync), kernel launches (e.g., cudaLaunchKernel), and stream synchronizations (e.g., cudaStreamSynchronize). The total runtime is derived by summing the total time spent in each API operation. In terms of runtime, DistilBERT and Vision Transformer show minimal differences across all three suppression methods. For DistilBERT, the runtime remains largely unchanged, with Clamp reducing runtime by 1.73% and RGD causing a negligible overhead of -0.08%. Similarly, in the Vision Transformer, the Clamp method results in a 2.03% increase in runtime, while RGD increases runtime by 2.63%. These results indicate that both methods introduce only minor runtime overheads for these auto-encoding models, suggesting the encoder-only architectures are computationally efficient in preserving performance.

In contrast, the auto-regressive model such as GPT-2 and sequence-to-sequence model like Marian exhibit greater variability in runtime. For GPT-2, Clamp increases runtime by 0.37%, while RGD incurs a more noticeable 4.63% increase in runtime. This increase is likely due to the more complex nature of the language model's generative tasks, particularly in auto-regressive models like GPT-2. On the other hand, for Marian, both Clamp and RGD significantly reduce runtime—by 43.5% and 42.4%, respectively. Unlike in GPT-2, where the overhead increases, Marian benefits from the error suppression module because it relies on intermediate outputs in its decoder, where the output tokens generated from the model are fed back into the decoder for further processing. With our suppression schemes, not only is the model's performance recovered, but the generated output sequences are stabilized, reducing the confusion in translation tasks. As a result, the entire process is accelerated, leading to the substantial reduction in runtime. The suppression technique prevents the propagation of erroneous activations, which could otherwise cause performance degradation and necessitate costly recovery operations, further contributing to runtime savings.

Memory usage provides the overall size of memory operations which is derived as the cumulative memory transfer size during execution, as reported by Nsight Systems. This metric includes data transfers between CPU and GPU, as well as internal GPU memory operations, over the entire duration of the run. It reflects the total memory movement overhead but does not indicate the peak memory allocated on the GPU at any given time. In DistilBERT, the memory overheads introduced by Clamp and RGD are minimal, at 0.33% and 0.34%, respectively, reflecting efficient handling of memory operations during suppression. The Vision Transformer and GPT-2 show no significant memory

Table 3 Runtime and memory overhead of error suppression schemes across various Transformer models under soft errors

Model	Suppression Scheme	Runtime (s)	Runtime Overhead (%)	Memory Usage (GB)	Memory Overhead (%)	Performance	Performance w/o Error Injection
DistilBERT	No Suppression	24.84	-	267.6	-	51.9 (%)	92.9 (%)
	Clamp	24.41	-1.73	268.4	0.33	90.7 (%)	
	Range-Guided Drop	24.82	-0.08	268.5	0.34	93.1 (%)	
Vision Transformer	No Suppression	21.64	-	214.2	-	28.1 (%)	99.3 (%)
	Clamp	22.08	2.03	214.2	0	99.2 (%)	
	Range-Guided Drop	22.21	2.63	214.2	0	99.3 (%)	
GPT-2	No Suppression	16.97	-	90.07	-	0.2 (%)	62.4 (%)
	Clamp	17.03	0.37	90.07	0	52.0 (%)	
	Range-Guided Drop	17.75	4.63	90.07	0	62.1 (%)	
Marian	No Suppression	318.3	-	1727.8	-	6.3 (BLEU)	52.7 (BLEU)
	Clamp	179.7	-43.5	891.3	-48.4	51.9 (BLEU)	
	Range-Guided Drop	183.3	-42.4	863.5	-50.0	52.5 (BLEU)	

overhead for either method, further supporting the lightweight nature of both Clamp and RGD for image classification and word prediction tasks. In the case of Marian Transformer, the memory usage analysis shows similar trends to the runtime analysis. This model displays the most notable results in terms of memory efficiency, with Clamp reducing memory usage by 48.4% and RGD by 50.0%. These substantial reductions reflect the efficiency of both suppression schemes in managing memory during machine translation tasks, where large memory transfers and storage requirements are common due to the size and complexity of the language translation model. By mitigating erroneous activations early (before propagated to the following layers), both methods significantly reduce the total memory overhead.

In terms of performance, both Clamp and RGD offer substantial improvements compared to the No Suppression baseline. In DistilBERT, for example, Clamp restores accuracy to 90.7%, while RGD further improves accuracy to 93.1%, reaching the ideal performance of 92.9%. Similarly, for the Vision Transformer, both methods recover accuracy to nearly the nominal level, with Clamp reaching 99.2% and RGD achieving 99.3%. GPT-2, evaluated using the token prediction accuracy metric, sees significant improvements from both suppression schemes. Clamp improves the accuracy to 52%, while RGD increases it to 62.1%, almost matching the nominal performance of 62.4%. For the Marian model, where BLEU score is the performance metric, Clamp improves BLEU score from 6.3 to 51.9, while RGD increases it slightly further to 52.5, close to the ideal BLEU score of 52.7.

Overall, these results demonstrate that RGD consistently outperforms Clamp in terms of performance recovery, albeit with slightly higher runtime overhead, particularly in auto-regressive models like GPT-2. Notably, both suppression methods introduce negligible to no additional memory overhead; in fact, in language models such as Marian, our suppression approach can even reduce overall memory usage. This analysis underscores the trade-off between performance recovery and computational overhead, with RGD offering a more robust solution for mitigating soft errors, while Clamp provides a more lightweight alternative with minimal overhead.

It is important to note that the observed performance recovery is attributable to the proposed error detection–suppression architecture and vulnerability-guided placement methodology, rather than the underlying simulation tools or assumed hardware technologies. The EDS modules operate at the functional block level of Transformer architectures, targeting fully connected layers and activation outputs using algorithmic checksum validation and range-based suppression. As such, the approach is independent of specific ASIC technologies, fabrication nodes, or backend design tools. All experiments are conducted under identical software and fault-injection conditions, and performance differences arise solely from the selective placement and activation of EDS modules. This confirms that the improvements stem from architectural design choices rather than platform-specific optimizations.

7 Conclusion

This paper presents a comprehensive approach to enhancing error resilience in Transformer networks through targeted error detection and suppression mechanisms. Leveraging algorithmic checksums for precise error detection and applying selective suppression techniques enable substantial recovery from performance degradation caused by both neuron output and memory access errors. Extensive experimental results across diverse Transformer architectures and datasets, spanning tasks from computer vision to language modeling, validate the effectiveness of the proposed error resilience techniques, showing minimal runtime and memory overhead.

Author Contributions K. Ma led the research and development of this work, including the formulation of the problem domain in Transformer architectures, the design of soft error fault models, the development of error detection and suppression techniques, the implementation of the experimental framework, and the execution of all experiments. All authors collaboratively conceptualized the core ideas. K. Ma wrote the main manuscript. C. Amarnath supported the review and refinement of the manuscript, and A. Chatterjee provided supervision and critical feedback throughout the research and writing process. All authors reviewed and approved the final manuscript.

Funding This work was funded by the U.S. National Science Foundation under Grant: 2128149.

Data Availability All data generated or analyzed during this study are available from the corresponding author on reasonable request.

Declarations

Conflicts of interest The authors declare that they have no conflicts of interest.

Competing interests The authors declare no competing interests.

References

1. Vaswani A, Shazeer A, Parmar N, Uszkoreit J, Jones J, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst 30
2. Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805
3. Radford A, Narasimhan K, Salimans T, Sutskever I et al (2018) Improving language understanding by generative pre-training
4. Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V, Zettlemoyer L (2019) Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461
5. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S et al (2020) An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929
6. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S (2020) End-to-end object detection with transformers. In: European conference on computer vision, pp 213–229
7. Junczys-Dowmunt M, Grundkiewicz R, Dwojak T, Hoang H, Heafield K, Neckermann T, Seide F, Germann U, Fikri Aji A, Bogoychev N, Martins AFT, Birch A (2018) Marian: Fast neural machine translation in C++. In: Proceedings of ACL 2018, system demonstrations, pp 116–121
8. Huang K-H, Abraham JA (1984) Algorithm-based fault tolerance for matrix operations. IEEE Trans Comput 100(6):518–528
9. Pandey S, Banerjee S, Chatterjee A (2018) Error resilient neuromorphic networks using checker neurons. In: 2018 IEEE 24th international symposium on on-line testing and robust system design (IOLTS). IEEE, pp 135–138
10. Ozen E, Orailoglu A (2019) Sanity-check: Boosting the reliability of safety-critical deep neural network applications. In: 2019 IEEE 28th asian test symposium (ATS). IEEE, pp 7–75
11. Ozen E, Orailoglu A (2020) Just say zero: containing critical bit-error propagation in deep neural networks with anomalous feature suppression. In: 2020 IEEE/ACM international conference on computer aided design (ICCAD). IEEE, pp 1–9
12. Amarnath C, Mejri M, Ma K, Chatterjee A (2022) Soft error resilient deep learning systems using neuron gradient statistics. In: 2022 IEEE 28th international symposium on on-line testing and robust system design (IOLTS), pp 1–7
13. Chen Z, Li G, Pattabiraman K (2021) A low-cost fault corrector for deep neural networks through range restriction. In: 2021 51st Annual IEEE/IFIP international conference on dependable systems and networks (DSN). IEEE, pp 1–13
14. Zhang JJ, Gu T, Basu K, Garg S (2018) Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator. In: 2018 IEEE 36th VLSI test symposium (VTS). IEEE, pp 1–6
15. Xia L, Liu M, Ning X, Chakrabarty K, Wang Y (2017) Fault-tolerant training with on-line fault detection for rram-based neural computing systems. In: Proceedings of the 54th annual design automation conference 2017, pp 1–6
16. Dai H, Wu S, Huang J, Jian Z, Zhu Y, Hu H, Chen Z (2025) Ft-transformer: Resilient and reliable transformer with end-to-end fault tolerant attention," In: Proceedings of the international conference for high performance computing, networking, storage and analysis, pp 1085–1098
17. Ahsaei S, Raji M (2025) Lost-vit: a low overhead soft error tolerance framework for vision transformers via model compression and selective bit-level redundancy. J Syst Arch 103623
18. Baradaran F, Raji M, Baradaran A, Baradaran A, Akbarifard R (2025) Zero memory overhead approach for protecting vision transformer parameters against bit-flip faults, pp 1–5
19. Ma K, Amarnath C, Chatterjee A (2023) Error resilient transformers: A novel soft error vulnerability guided approach to error checking and suppression. In: 2023 IEEE European test symposium (ETS), pp 1–6
20. Xue X, Liu C, Wang Y, Yang B, Luo T, Zhang L, Li H, Li X (2023) Soft error reliability analysis of vision transformers. IEEE Trans Very Large Scale Integr (VLSI) Syst 31(12):2126–2136
21. Roquet L, dos Santos FF, Rech P, Traiola M, Sentieys O, Kritikakou A (2024) Maximals: A low-cost hardening technique for large vision transformers. In: RADiation effects on components and systems (RADECS)
22. Liu H, Singh V, Filipiuk M, Hari SKS (2024) Alberta: Algorithm-based error resilience in transformer architectures. IEEE Open J Comput Soc
23. Gao Z, Liu S, Reviriego P, Liu S, Lombardi F (2024) Dependability and protection of transformer models against soft errors on text embeddings. IEEE Trans Device Mater Reliab
24. Rashidi B (2024) Fault-tolerant and error-correcting 4-bit s-boxes for cryptography applications with multiple errors detection. J Supercomput 80(2):1464–1490
25. Claeys C, Simoen E (2002) Radiation effects in advanced semiconductor materials and devices. Springer Science & Business Media, vol 57
26. Baumann RC (2005) Radiation-induced soft errors in advanced semiconductor technologies. IEEE Trans Device Mater Reliab 5(3):305–316
27. Mahatme N, Jagannathan S, Loveless T, Massengill L, Bhuva B, Wen S-J, Wong R (2011) Comparison of combinational and sequential error rates for a deep submicron process. IEEE Trans Nucl Sci 58(6):2719–2725
28. Torres-Huitzil C, Girau B (2017) Fault and error tolerance in neural networks: A review. IEEE Access 5:17 322-17 341
29. Li G, Hari SKS, Sullivan M, Tsai T, Pattabiraman K, Emer J, Keckler SW (2017) Understanding error propagation in deep learning neural network (dnn) accelerators and applications. In: Proceedings of the international conference for high performance computing, networking, storage and analysis, pp 1–12
30. Ibrahim Y, Wang H, Liu J, Wei J, Chen L, Rech P, Adam K, Guo G (2020) Soft errors in dnn accelerators: A comprehensive review. Microelectron Reliab 115:113969
31. Sanh V, Debut L, Chaumond J, Wolf T (2019) Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108
32. Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I et al (2019) Language models are unsupervised multitask learners. OpenAI Blog 1(8):9
33. Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies. Portland, Oregon, USA: Association for Computational Linguistics, pp 142–150. Available: http://www.aclweb.org/anthology/P11-1015

34. Krizhevsky A, Hinton G et al (2009) Learning multiple layers of features from tiny images. University of Toronto, Tech. Rep
35. Paperno D, Kruszewski G, Lazaridou A, Pham QN, Bernardi R, Pezzelle S, Baroni M, Boleda G, Fernández R (2016) The lambada dataset: Word prediction requiring a broad discourse context. arXiv preprint arXiv:1606.06031
36. Tiedemann J (2012) Parallel data, tools and interfaces in opus," In: Proceedings of the eight international conference on language resources and evaluation (LREC'12)
37. Patterson D, Gonzalez J, Hölzle U, Le Q, Liang C, Munguia L-M, Rothchild D, So DR, Texier M, Dean J (2022) The carbon footprint of machine learning training will plateau, then shrink. Computer 55(7):18–28
38. Clark C, Lee K, Chang M-W, Kwiatkowski T, Collins M, Toutanova K (2019) Boolq: Exploring the surprising difficulty of natural yes/no questions. In: NAACL

Kwondo Ma received the Ph.D. degree in Electrical and Computer Engineering from the Georgia Institute of Technology, where he worked with Prof. Abhijit Chatterjee. He is currently a Neural Processing Unit (NPU) library software engineer at Rebellions Inc., Seongnam-si, Republic of Korea. His published work covers error resilience in Transformer architectures, fault-tolerant computing for deep neural networks, and post-manufacture tuning of analog in-memory computing systems.

Chandramouli Amarnath is a Ph.D. student in the Smart Resilient Systems Lab at Georgia Tech, working with Prof. Abhijit Chatterjee. He received his B.Tech from the National Institute of Technology Karnataka, Surathkal in 2018 and began his Ph.D. at Georgia Tech in the same year. His published work covers hierarchical failure modeling, recovery in autonomous vehicles, secure operation of DNNs, and error resilience in state estimation.

Jackson Isenberg is a Ph.D. student in the Smart Resilient Systems Lab at Georgia Tech, working with Prof. Abhijit Chatterjee. He received his BS in CS from Georgia Tech in 2024 and his MS in Robotics in 2026. His work focuses on resilience in DNNs, control suites for autonomous vehicles, and secure LLM-robot integration.

Abhijit Chatterjee is a Professor in the School of Electrical and Computer Engineering at Georgia Tech and a Fellow of the IEEE. He received his Ph.D. in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 1990. Dr. Chatterjee has received the NSF Research Initiation Award (1993), NSF CAREER Award (1995), and seven Best Paper Awards. His work on self-healing chips was cited by the Wall Street Journal in 1992. He has authored over 450 papers in refereed journals and meetings, holds 22 patents, and has supervised over 50 Ph.D. dissertations. He co-founded Ardext Technologies Inc., a mixed-signal test solutions company, and served as chief scientist from 2000–2002. His research focuses on error-resilient, self-testing, and adaptive "self-aware" intelligent systems for sensing, communication, machine learning, and control.