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

Kwondo Ma, Chandramouli Amarnath and Abhijit Chatterjee
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta GA 30332
Email: {kma64, chandamarnath}@gatech.edu, abhijit.chatterjee@ece.gatech.edu

***Abstract*—Transformer networks have achieved remarkable success in Natural Language Processing (NLP) and Computer Vision applications. However, the underlying large volumes of Transformer computations demand high reliability and resilience to soft errors in processor hardware. The objective of this research is to develop efficient techniques for design of error resilient Transformer architectures. To enable this, we first perform a soft error vulnerability analysis of every fully connected layers in Transformer computations. Based on this study, error detection and suppression modules are selectively introduced into datapaths to restore Transformer performance under anticipated error rate conditions. Memory access errors and neuron output errors are detected using checksums of linear Transformer computations. Correction consists of determining output neurons with out-of-range values and suppressing the same to zero. For a Transformer with nominal BLEU score of 52.7, such vulnerability guided selective error suppression can recover language translation performance from a BLEU score of 0 to 50.774 with as much as 0.001 probability of activation error, incurring negligible memory and computation overheads.**

***Index Terms*—Transformer, error resilience**

I. Introduction

Transformer networks [1] have achieved remarkable success in Natural Language Processing (NLP), overcoming the limitations of sequential computation in Recurrent (RNN) and long short-term memory (LSTM) networks. There has been wide adoption of Transformer networks in word/sentence classification [2], text generation [3], summarization, and language translation [4]. Moreover, in recent studies, Transformer models have been successfully applied for image recognition [5] and object detection [6] in computer vision. The underlying large volumes of Transformer computations demand high reliability and resilience to soft errors in processor hardware. A memory access error experiment on sequence-to-sequence Transformer networks [7] for a language translation task is illustrated in Fig. 1. An English input sentence is successfully translated to a French output sentence (underlined text in green box) without any errors. However, a Transformer model in which the model weights in memory are subjected to random bitflips with error rate of 10^{-6}, produces significantly incorrect French translations (underlined text in red box).

In the past, Algorithm-Based Fault Tolerance (ABFT) [8] has addressed the design of fault-tolerant signal processing algorithms. Techniques for efficiently detecting soft errors in computations of perceptron and convolutional neural networks drawing on ABFT methods were discussed in [9] and [10], respectively. For error correction, there has been recent work in suppressing erroneous neuron output values in deep neural networks (DNNs) to zero [11]. However, a diagnosis process to determine which neurons are erroneous requires modifications to be made to network training algorithms to generate invariant interrelationships among neurons and accommodate such zero-suppression of erroneous neuron outputs. Zeroing-based error suppression has also been used in more recent work [12] that utilizes statistical analysis of neuron interrelationships to diagnose erroneous neuron outputs, thus avoiding any modification of network training. A recent competing technique works by restricting the ranges of values of the neuron outputs in DNN hidden layers [13] based on experiments performed on error-free circuits with training dataset, however, this requires expensive calculations of neuron interrelationships for on-line error correction. Moreover, the range restriction [13] suffers performance degradation in the face of memory access errors and restricts placement of its range-restriction systems to nonlinear activations, batchnorm and pooling layers, limiting its applicability to transformer networks with multiple layer computations prior to activation operations. Similarly, related work on error mitigation of permanent faults in systolic array based accelerators [14] and RRAM-based neural computing systems [15] requires additional fault-tolerant training and cannot easily be adapted to the problem of soft error and memory access error correction.

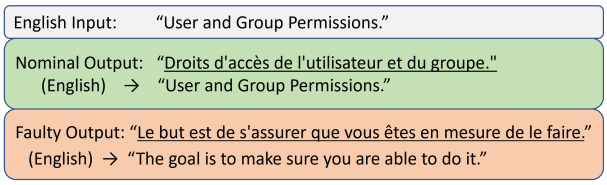


Figure 1: An example of faulty machine translation caused by memory access errors.

The goal of this research is to design error detection and correction mechanisms for Transformer networks (with multi-head attention and feedforward sub-networks) that leverage prior research on error resilience in DNNs, however: (a) require no re-engineering of Transformer network training algorithms, (b) are effective across memory access errors as well as errors in neuron output computations. and (c) are computationally efficient with use of tailored checks and error suppression techniques inserted into the most vulnerable Transformer layers for rapid error recovery.

To achieve the above goals, we perform a soft error vulnerability analysis of all fully connected layers of Transformer computations. Based on this study, error detection and suppression modules are selectively introduced into the datapaths to restore Transformer performance under stipulated error rate

conditions. *Memory access errors* and *neuron output errors* are detected using checksums of linear Transformer computations. Both weight and neuron output errors are corrected by suppressing out-of-range erroneous neuron output values to zero. *To the best of our knowledge, our study is the first to address soft errors in Transformer networks.*

The key aspects of the proposed research are:
(1) Transformer module soft error vulnerability analysis: Memory access errors and neuron output errors in Transformer network 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.
(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.

II. Background and Fault Model

An overview of transformer architecture and assumed fault models are as follows.

A. Transformer Architecture

The transformer architecture [1] consists of an encoder-decoder structure in which a stack of encoders receives a sequence of input tokens and learns the relationships between the sequence of tokens. From the encoder output, a stack of decoders computes tokens of the output sequence at a time, using prior decoder outputs as an additional input for generating the next output sequence token in order.

1) Encoder and Decoder: Encoder and decoder networks are composed of a stack of identical blocks and designed with two types of sub-layers: *multi-head attention* and *feed-forward network*. On the encoder side, a tokenized input sequence is transformed into word embedding vectors and summed with positional encoding with regard to the relative position of tokens in a sequence. Two sub-layers generate the output sequence under corresponding residual connections and layer normalization as shown in the left half of Fig. 2. The decoder block is similar to the encoder in structure but contains additional multi-head attention networks, which compute keys and values of attention over encoder outputs. The first multi-head attention in the decoder is *masked* to ensure that model prediction at time i is only affected by known outputs at time less than i.

2) Attention Mechanism: The attention sub-layer in the encoder and decoder architecture maps inputs to sets of values, queries and keys to encode token sequences instead of using recurrent connections. *Value* determines where attention must focus within a sequence; *Query* and *Key* are used to compute the corresponding attention weights (scores). For computing these quantities, three fully connected (FC) layers are utilized as;

$$Q/K/V = X \cdot W_{Q/K/V} + B_{Q/K/V}, \quad (1)$$

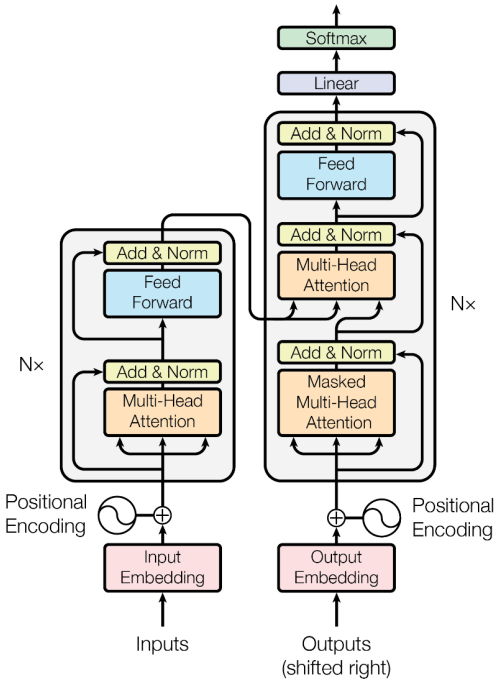


Figure 2: Transformer network architecture [1].

where X is an input, and $W\_$ and $B\_$ are weights and biases of FC layers for query, key and value, respectively. When the inputs of three FC layers are identical, *self-attention* is realized (e.g., first attention block in encoder and decoder network). When two different inputs to the FC layers are used, *cross-attention* is realized (e.g., second attention block in decoder). The *scaled dot-product (SDP) attention* generates its output as a weighted sum of values as;

$$Z = SDP(Q, K, V) = softmax(QK^T/\sqrt{d_k})V, \quad (2)$$

where d_k is a dimension of keys and $softmax()$ indicates the softmax activation function. In the original transformer architecture, *multi-head attention* computes multiple linear projections using multiple parallel attention heads and generates the same output sequence by concatenation. Finally, there is another FC layer for projecting attention output as $Attn = Z \cdot W_O + B_O$.

3) Feed-Forward Network: The second sub-layer contains two fully connected layers and one activation function as;

$$FFN(X) = [ACT(X \cdot W_{FF1} + B_{FF1})] \cdot W_{FF2} + B_{FF2}, \quad (3)$$

where X is an input and $ACT()$ indicates activation function (e.g., RELU, GELU) of feed-forward network. The dimensionality of input and output of feed-forward network has to be preserved, and intermediate layer (i.e., first FC layer) has four times greater dimensionality.

B. Fault Model

An assessment of error resilience of Transformer networks have to our knowledge not been studied. We thus begin by defining a fault model for Transformer networks. As discussed earlier, the Transformer architecture is constructed from multiple fully connected layers, activation functions, and layer normalization using the unique attention mechanism

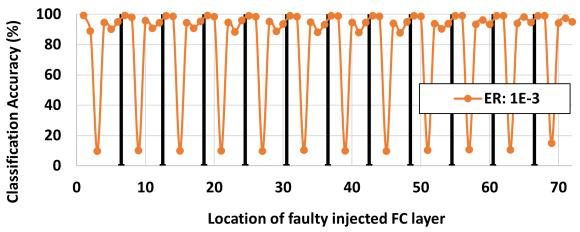


Figure 3: Example vulnerability analysis of auto-encoding Transformer.

and residual connections. While the Transformer network is running on hardware such as a GPU, hardware transient faults (i.e., soft errors) can occur in computational units and processor datapaths. We model these errors as random bitflip injections in the neuron output values of FC layers. Additionally, soft errors in main memory and cache can cause data corruption errors which are manifested as random bitflips in the weight coefficients of corresponding layers. Therefore, we model *neuron output errors* and *memory access errors* during the inference phase of a Transformer using random bitflip injections.

III. Proposed Methodology

This section presents a soft error vulnerability analysis of Transformer network computations. Following this, we propose efficient error detection and efficient error suppression method for Transformer networks.

A. Soft Error Vulnerability Analysis

We perform *error vulnerability analysis by injecting neuron output errors and weight errors into individual fully connected layers* of Transformer networks. In the selected Transformer architecture, the encoder and decoder blocks contain 6 and 10 FC layers respectively, each of which have specific roles. For example, the image classification auto-encoding Transformer [5] studied in this work has 12 encoder blocks and no decoder, totaling 72 fully connected layers. Six different FC layers in each of encoder block are defined based on their roles as $\{Query, Key, Value, Output, FF1, FF2\}$ (see section II-A for details). Fig. 3 presents the results of vulnerability analysis for neuron output errors with error rate of 10^{-3} (the *error rate* denoting the probability of each neuron output values being erroneous), showing the locations of erroneous FC layers versus classification accuracy of the corresponding erroneous Transformer. Black vertical lines in Fig. 3 divide the results into segments of 6 data points, with each segment representing 6 FC layers in each encoder network. The vulnerability of each encoder block is seen to be similar throughout the entire Transformer network. Therefore, *a representative vulnerability is derived for each FC layer role of an encoder (decoder) block by averaging the results of individual vulnerability analysis for all FC layers.*

B. Error Detection Using Algorithmic Checksum

Weight and neuron output errors cause numerical deviations in neuron computations, which are the neuron output values of every fully connected layers in Transformer networks. These deviations eventually corrupt the Transformer output due to a fault propagation into sub-layers and residual connections, causing large value deviations. To minimize error propagation, we place error detection modules after each FC layer in the attention and feed-forward networks, indicated with diamond symbols in Fig. 4a and 4b.

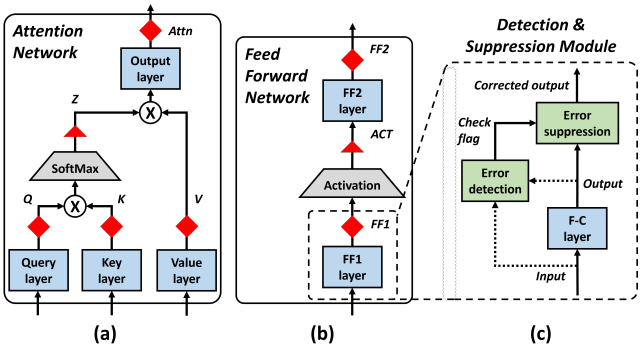


Figure 4: Placement of error detection and suppression modules at (a) Attention network, and (b) Feed-forward network. (c) Architecture of error detection and suppression module.

For detecting errors in neuron output values, we utilize *algorithmic checksums*. The input to the Transformer is a 2-D matrix $X \in \mathcal{R}^{l \times d}$ where l indicates the maximum sequence length and d is the dimension determined by word embedding. The computations of the n-th FC layer are thus:

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

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

A checksum can be computed for the output matrix over the j-th embedding vector axis as $\sum_{j=0}^{d-1} Y_n[i][j] = Y_n^s[i]$. The output sequence vector Y_n^s represents row-wise summations of word embeddings for every tokens in output sequence. Similarly, we apply an identical checksum to the right side of Eq. 4 as:

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

where B_n^s is the sum of bias vector elements and W_n^s is the weight vector derived from row-wise summation of the weight matrix. Ideally, the norm of vector subtraction $||Y_n^s - R_n^s||$ is zero in the absence of faults. In actual operation, we compute a threshold τ_n for the n-th FC layer which bounds this check value: $||Y_n^s - R_n^s|| < \tau_n$. The implementation of this error detection module is demonstrated in Fig. 4(c), where the module computes a check flag using the input and output of its corresponding FC layer. For each FC layer in the Transformer we save the summation of bias and weight matrix in memory, enabling efficient error detection.

C. Error Suppression

We propose two strategies for mitigating critical errors (i.e., those which eventually induce a Transformer malfunction)

by suppressing large numerical deviations of neuron output values. We *clamp* or *drop* erroneous out-of-range neuron output values which are diagnosed based on the predetermined operating bounds of each layer. We derive the operating bounds of neuron output values of each FC layer by a subset of the training dataset. Note that deriving the operating bounds for each layer is a one-time computation. A pair of bounds (upper and lower) per layer is determined for reference comparison from training data. The clamping method restricts neuron outputs of FC layer which are outside of operating bounds by clamping them to the minimum or maximum bounds as:

$$Clamp(Y_n) = \begin{cases} y_n^{min} & \text{if } Y_n[i][j] < y_n^{min} \\ y_n^{max} & \text{if } Y_n[i][j] > y_n^{max} \\ Y_n[i][j] & \text{otherwise.} \end{cases} \quad (5)$$

where Y_n is output, and (y_n^{min}, y_n^{max}) indicate the minimum and maximum bounds of Y_n in n-th layer. Our second strategy suppresses out-of-range values by dropping them to zero. We call this *range guided drop (RGD)* and is different from prior works [11], [13].

$$RGD(Y_n) = \begin{cases} Y_n[i][j] & \text{if } Y_n[i][j] \in [y_n^{min}, y_n^{max}] \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

The error suppression module is triggered based on a check flag from the detection system as shown in Fig. 4c. The placement of the *Error Detection & Suppression (EDS)* module at the n-th FC layer is:

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

where $Sup() \in \{Clamp(), RGD()\}$. Additionally, we suppress post activation values because *minor deviations* (i.e., perturbations which bypass EDS modules) *from two different FC layers can be amplified under multiplication* by the scaled dot-product attention mechanism. For activation mechanisms, error suppression is always turned "on", indicated by red triangle symbols in Fig. 4a and 4b.

D. Selective Error Detection-Suppression Module Placement

In this section we investigate selective *Error Detection-Suppression (EDS)* module placement. Previous sections determined potential placements of EDS modules based on the architecture of Transformer networks. From the previous vulnerability analysis, module placement experiments are conducted on every layer for all encoder/decoder blocks. This reduces the number of possible EDS module placement locations. For instance, 72 individual placement options are decreased to 6 in the previous example of section III-A.

The optimal EDS module placement under these conditions is one which maximizes error resilience of Transformer network with minimum overhead. Initially, we insert EDS modules at each potential position (i.e. for every layer in each role) to search for which placement (role) most effectively recovers Transformer performance. The EDS module achieving the largest error correction is then left in place and the next optimal placement under these conditions is found in a greedy manner. This selective EDS module placement allows prioritization of module insertion points based on the vulnerability of the Transformer to errors in specific layers of the network.

Table I: Transformer networks used for evaluation

Application	Image Classification	Machine Translation
Model	Vision Transformer [5]	Marian [7]
Dataset	CIFAR-10	KDE4 (EN-FR) [16]
Data description	General images	Language pair
# of Encoder	12	6
# of Decoder	0	6
Evaluation Metric	Classification Accuracy	BLEU score

IV. Experimental Setup

In this section, we describe the experimental setup and methodology for vulnerability analysis and evaluation. All experiments were conducted on a Ubuntu 22.04.1 system with an RTX A4000 GPU using 32-bit floating point computation.

Transformer benchmark applications. Table I summarizes details of models and datasets in our evaluation. For image classification, we use the Vision Transformer (ViT) model [5] on CIFAR-10 dataset, measuring classification accuracy. For Neural Machine Translation (NMT), the Marian sequence-to-sequence Transformer [7] is used to translate English to French on the KDE4 dataset [16]. The BiLingual Evaluation Understudy (BLEU) score is used to evaluate the similarity of machine-translated text to reference translations.

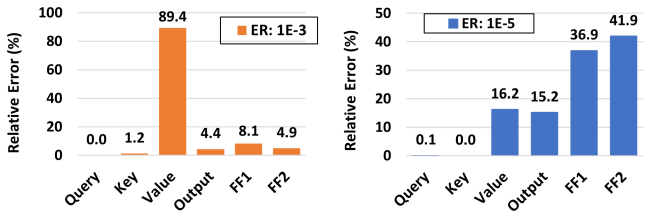
Error injection. We apply PytorchFI [17], a run-time perturbation tool for DNNs to model soft errors in FC layer computations by injecting random bitflips into neuron outputs of FC layers in the Transformer. To evaluate models under errors in neuron outputs, error configurations are randomly generated for each inference across the test dataset. For weight errors, we inject random bitflips on pre-trained weight values before model inference. Once we perturb the model weights under one weight error configuration, we evaluate this faulty model on the entire test dataset. We produce 100 weight error configurations in advance and calculate the mean and standard deviation of Transformer inference accuracy from 100 different inference results.

V. Experimental Results

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

A. Vulnerability Analysis

Fig. 5 shows the results of vulnerability analysis under soft errors in neuron computations and weight error injections on ViT, an auto-encoding Transformer. From Fig. 5a, neuron output errors injected into the *Value* layers completely corrupt the Transformer network (i.e., relative error = 89.4%) under an error rate of 10^{-3}. The *Value* layer has the highest vulnerability because numerical deviations in values are directly accumulated to the output of scaled dot-product attention due to a matrix multiplication (see Eq. 2). Minor degradation is observed when the two FC layers in the feed-forward network and *Output* layer in attention network are perturbed. The *FF1* layer (8.1% relative error) is the most vulnerable among these



(a) Neuron output error injection. (b) Weight error injection.

Figure 5: Vulnerability analysis of auto-encoding Transformer.

three layers due to the larger dimensionality of the layer of output neurons.

Next, vulnerability due to weight errors in ViT is evaluated in Fig. 5b. Weight errors in the feed-forward network cause major network performance degradation because the dimensions of the weight matrix in the *FF1* and *FF2* layers are significantly larger than the other layers of the network. As a result, under identical error rates, the *FF1* and *FF2* layers incur greater error propagation within the network. We categorize the other 4 layers into two groups: The *Value* and *Output* layers have intermediate vulnerability and the *Query* and *Key* layers have almost zero sensitivity to weight errors.

A similar result is derived from a deeper sequence-to-sequence Transformer for language translation under weight error injection. As shown in Fig. 6 for both encoder and decoder, feed-forward networks possess major vulnerabilities, followed by *Value* and *Output* layers at self-attentions and cross-attention layers, which have intermediate sensitivity. The *FF1* layer consistently shows lower relative error than the *FF2* layer (5% lower) despite identical dimensions of corresponding weight matrices. This is because the activation function (e.g., RELU) after *FF1* layer filters out many of the erroneous values (negative values). Moreover, we conclude that the *Query* and *Key* layers have notable inherent resilience due to the softmax function right after their multiplication in the attention network under both neuron output and weight error injections.

B. Selective Error Detection-Suppression Module Placement

Fig. 7 illustrates the result of selective Error Detection-Suppression (EDS) module placement with a correlated column chart and table. In this analysis, we inject a 10^{-4} neuron output errors in FC layers on the ViT model, whose nominal inference accuracy of Transformer is 99.1%. The suppression

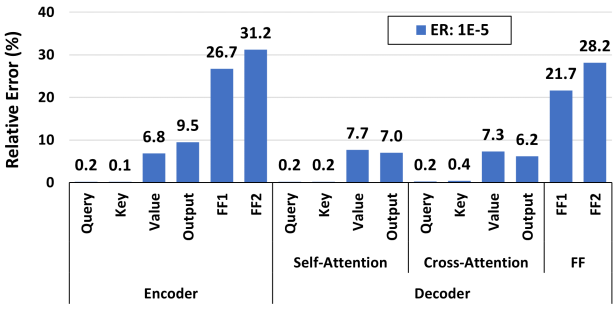


Figure 6: Vulnerability analysis of weight errors of sequence-to-sequence Transformer.

	①	②	③	④	⑤	⑥
Query	X	8.6	75.7	87.6	93.8	98.1
Key	X	8.6	79.6	90.4	95.9	99.1
Value	X	75.9	O	O	O	O
Softmax	X	8.7	76.1	89.6	96.5	98.8
Output	X	30.8	72.1	93.7	O	O
FF1	X	8.7	83.6	89.8	98.2	O
Activation	X	8.7	83.6	89.8	98.2	98.2
FF2	X	8.6	87.8	O	O	O

Figure 7: Selective EDS module placement in auto-encoding Transformer.

strategy here is range guided drop, setting out-of-range values to zero. Column ① in Fig. 7 indicates the Transformer performance without error suppression. The corresponding misclassification rate is shown as 90.4% (i.e., accuracy of faulty Transformer = 8.7%). Column ② displays the classification accuracy of the Transformer with EDS modules inserted in a designated layer (e.g., when we insert modules in all *Value* layers, the Transformer accuracy is 75.9%). Based on results in Column ②, we select the *Value* layer for module insertion, achieving the largest accuracy recovery and reducing the misclassification rate to 23.2%. For each best placement at each stage (marked as dashed box in the table), the corresponding EDS module is inserted and the next optimal placement is found in a greedy manner. Finally, on selecting the *Key* layers in column ⑥, the Transformer accuracy is fully recovered (i.e., misclassification rate = 0%).

Using selective EDS module placement on auto-encoding Transformer, module insertion can be similarly prioritized for optimal recovery. The priority ordering found was *"Value >> FF2 > Output > FF1 (=Activation) > Key"*, which is consistent with the results from the above vulnerability analysis. Suppressing the *Value* layers recovers 67.2% of misclassification rate, indicating it was the most vulnerable layer role in the auto-encoding Transformer. Then, when we correct layers with low vulnerability (i.e., *Output*, *FF1*, and *FF2* layers) which cause more than 4% error in Fig. 5b, we can reduce misclassification rate down to 0.9%.

C. Evaluation

We evaluate the performance of Transformer networks with and without error resilience modules in place. Error detection provides 100% error-caused degradation coverage under neuron output errors and weight errors. For the suppression methodologies, we compare the performance of clamping and range guided dropping. The proposed suppression schemes are low cost (i.e., floating point instruction count (FLOP) overheads from prior studies are projected to be less than 3% [18]). Soft errors are injected into neuron output values in the auto-encoding Transformer and into the weights of layer matrix computations in the sequence-to-sequence Transformer.

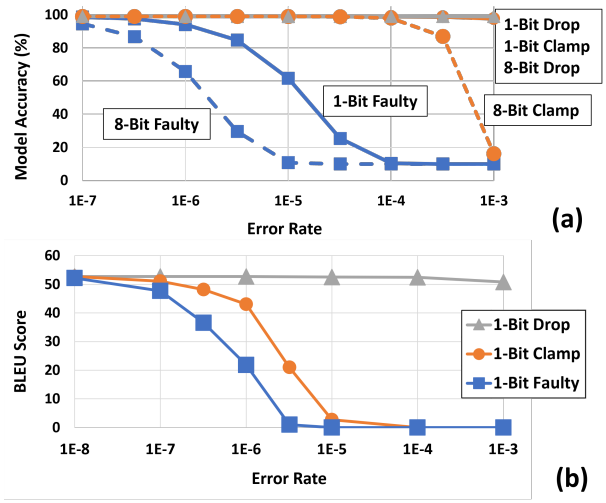


Figure 8: Evaluation of (a) auto-encoding Transformer under neuron output errors, (b) sequence-to-sequence Transformer under weight errors.

1) Neuron Output Errors: We evaluate neuron output errors using single (1-bit) and multiple (8-bits) bitflip injection in Fig. 8a. The accuracy of erroneous Transformer decreases as the error rate increases. Auto-encoding Transformer becomes dysfunctional at an error rate of 10^{-4} for single and 10^{-5} for multiple bitflip injection. For single bitflip injection, error suppression using either clamping or range guided dropping recovers 99% of degraded accuracy at the highest error rate of 10^{-3}. Under word bitflip injection, the error resilient Transformer using clamping suppression starts to degrade when the error rate surpasses 3.2×10^{-4} and sees a collapse in functionality at an error rate of 10^{-3} (i.e., Transformer inference accuracy = 16.19%). By contrast, suppression using range guided drop can successfully recover the performance accuracy of auto-encoding Transformer from 10% in the absence of such suppression to 98.88% at the highest error rate (nominal Transformer accuracy = 98.94%).

2) Weight Errors: We evaluate weight errors using single bitflip injection in the sequence-to-sequence Transformer for language translation (nominal BLEU score 52.71). The performance of the erroneous Transformer rapidly degrades as the error rate increases and completely collapses (i.e., 0.98 BLEU) when the error rate reaches 3.2×10^{-6} as shown by the square marker in Fig. 8b. Using suppression via clamping (graph with circle marker in Fig. 8b), the translation quality slightly recovers but the network malfunctions (i.e., 2.74 BLEU) when the error rate surpasses 10^{-5}. Clamping-based suppression thus shows insufficient performance for weight errors as opposed to neuron output errors. Weight errors degrade the Transformer functionality by a larger amount compared to neuron output errors because the repeated use of weight values can cause multiple accumulated errors. Consequently, under the clamping strategy, multiple neuron outputs are clamped towards minimum and maximum bounds, inducing degraded recovery. On the other hand, an error resilient Transformer using range guided dropping can recover the performance of sequence-to-sequence Transformer with 0 BLEU score to 52.42 and 50.77 under error rates of 10^{-4} and 10^{-3}, respectively. In conclusion, the Transformer architecture exhibits graceful performance degradation with range guided drop (RGD) versus clamp error suppression mechanism with increasing error rates.

VI. Conclusion

This paper presents an error resilience methodology for Transformer networks. We present efficient error detection using algorithmic checksum and selective error suppression, to recover from large deviations caused by neuron output errors and memory access (weight) errors. The proposed scheme is validated on multiple Transformer networks and tasks, recovering more than 96% of degraded performance due to transient soft errors.

Acknowledgment

This research was supported by the U.S. National Science Foundation under Grant: 2128149.

References

[1] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
[2] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[3] A. Radford *et al.*, "Improving language understanding by generative pre-training," 2018.
[4] M. Lewis *et al.*, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.
[5] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
[6] N. Carion *et al.*, "End-to-end object detection with transformers," in *European conference on computer vision*, 2020, pp. 213–229.
[7] M. Junczys-Dowmunt *et al.*, "Marian: Fast neural machine translation in C++," in *Proceedings of ACL 2018, System Demonstrations*, 2018, pp. 116–121.
[8] K.-H. Huang *et al.*, "Algorithm-based fault tolerance for matrix operations," *IEEE transactions on computers*.
[9] S. Pandey *et al.*, "Error resilient neuromorphic networks using checker neurons," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*. IEEE, 2018, pp. 135–138.
[10] E. Ozen *et al.*, "Sanity-check: Boosting the reliability of safety-critical deep neural network applications," in *2019 IEEE 28th Asian Test Symposium (ATS)*. IEEE, 2019, pp. 7–75.
[11] ——, "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, 2020, pp. 1–9.
[12] C. Amarnath *et al.*, "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)*, 2022, pp. 1–7.
[13] Z. Chen *et al.*, "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, 2021, pp. 1–13.
[14] J. J. Zhang *et al.*, "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, 2018, pp. 1–6.
[15] L. Xia *et al.*, "Fault-tolerant training with on-line fault detection for rram-based neural computing systems," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.
[16] J. Tiedemann, "Parallel data, tools and interfaces in opus," in *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, 2012.
[17] A. Mahmoud *et al.*, "Pytorchfi: A runtime perturbation tool for dnns," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2020, pp. 25–31.
[18] C. Amarnath *et al.*, "Tesda: Transform enabled statistical detection of attacks in deep neural networks," *arXiv preprint arXiv:2110.08447*, 2021.