# Evaluation and mitigation of faults affecting Swin Transformers

G. Gavarini, A. Ruospo, E. Sanchez
Politecnico di Torino, DAUIN, Torino, Italy

***Abstract*—In the last decade, a huge effort has been spent on assessing the reliability of Convolutional Neural networks (CNNs), probably the most popular architecture for image classification tasks. However, modern Deep Neural Networks (DNNs) are rapidly overtaking CNNs, as state-of-the-art results for many tasks are achieved with the Transformers, innovative DNN models. Transformers' architecture introduces the concept of attention as an alternative to the classical convolution operation. The aim of this work is to propose a reliability analysis of the Swin Transformer, one of the most accurate DNN used for Image Classification, that greatly improves the results obtained by traditional CNNs. In particular, this paper shows that, similar to CNNs, Transformers are susceptible to single faults affecting weights and neurons. Furthermore, it is shown how output ranging, a well-known technique to reduce the impact of a fault in CNNs, is not as effective for the Transformer. The alternative solution proposed by this work is to introduce a ranging not only on the output, but also on the input and on the weight of the fully connected layers. Results show that, on average, the number of critical faults (i.e., that modify the network's output) affecting neurons decreases by a factor of $1.91$, while for faults affecting the network's weights this value decreases by a factor of $1 \cdot 10^5$.**

***Index Terms*—Deep Neural Network, Reliability, Fault Injection, Transformer**

## I. Introduction

Nowadays, Deep Neural Networks (DNNs) are one of the most used algorithms to solve many different tasks, ranging from Image classification to Natural Language Processing. Due to their effectiveness, they are used in multiple fields: as generative language models, in robotics and even in finance. Moreover, thanks to the benefit they provide, they are often employed in safety-critical environments. However, while DNNs were initially believed to be inherently fault-tolerant, due to their distributed and parallel nature, multiple studies remark how this is not the case [1], [2].

Until now, the main focus of multiple research works has been to understand how faults affect and propagate in the most commonly used DNN models: Convolutional Neural Networks (CNNs). While some studies have focused on hardware-level fault simulations and remedies [3], others have proposed more high-level solutions [1], [4], [5]. A software-level approach, while typically unintrusive and easy to apply, can be effective to assess the resilience of DNN models to hardware faults. In particular, multiple solutions trade a tolerable loss in accuracy for a more fault resistant architecture. For example, in [4], the matrix multiplication used to compute the results of a convolutional layer are strengthened by means of a checksum implementing Algorithm-Based Fault Tolerance (ABFT). A different solution, presented in [5], [6], introduces a range restriction on the non-linear activation function: when the data is represented as 32 bit floating-point values, this solution proves to be extremely effective and inexpensive.

However, while CNNs are starting to become more and more fault-tolerant, they are falling behind in terms of effectiveness: the trend in DNNs is rapidly shifting towards Transformers [7]. This new kind of architecture has been proved to be more successful than CNNs in solving different tasks, thanks to the newly introduced *attention* mechanism. *Attention* is an innovative concept that allows to efficiently deal with sequences of elements, without the drawbacks of previous architecture, such as the Long-Short Term Memory (LSTM) networks. While Transformers have been initially employed in machine translation, they have been quickly and successfully adopted in other tasks such Image Classification, Instance and Semantic Segmentation [8].

The reliability of this new type of DNN architectures still need to be properly addressed. To the best of authors' knowledge, very few research works target Transformers' reliability. For example, in [9], the authors study the reliability of a selected group of Transformer neural networks, proposing to apply a revised version of some of the techniques used to mitigate faults in CNNs. In particular, they propose to use ABFT to strengthen linear operations and range-restriction for non-linear computations. This work experimentally demonstrates that, similar to what was found in the case of DNNs, some intrinsic mitigation is also found in the case of Transformers. However, this is far from a compressive analysis. Therefore, it is necessary to further investigate the applicability of solutions originally developed for CNNs to Transformers.

This research work proposes a novel analysis of Swin Transformer, a state-of-the-art architecture capable of reaching one the best accuracy on the ImageNet dataset while being trained only on publicly available data. The first objective of this paper is to explore Transformers from a reliability point of view, highlighting their differences and similarities with CNNs. Once this has been addressed, this work proposes a methodology capable of dealing with some of the weakness of Transformers. In particular, taking advantage of both its similarity, and its difference, with respect to a convolutional layer, this paper proposes a technique to safely estimate the output of a neuron of the network in presence of a fault, by ignoring out-of-range inputs and weights (i.e., the network parameters) prior to the computation of the output.

The reminder of the paper is organized as follows: Section II provides a brief background on Transformer Neural Networks and the Swin Transformer, as well as some state-of the art solution to improve the fault reliability of CNNs and Transformers. Next, Section III describes the analysis performed on the network, as well as the techniques used to have a more educate estimate of the neuron outputs. Section V reports on the experimental results. To conclude, Section VI summarizes the contributions of the paper and describes future directions.

## II. Background and Related Works

To address the concerns relative to the fault tolerance of Transformers, this work first presents their architecture, juxtaposing it to the one of CNNs used for Image Classification. Then, a brief overview on some of the state-of-the-art fault analysis and mitigation techniques for both CNNs and Transformers is provided.

### A. Transformers

Transformer Neural Networks were originally developed for sequence modelling and translation tasks. The advantage of this kind of networks is that they are able to efficiently work with sequences and to introduce a relation between multiple elements thanks to the *attention* mechanism. The *self-attention* mechanism (first proposed in [7]) can be thought as a function that correlates a query $Q$ and a key-value $K, V$ pair to an output, that measures how relevant the pair is to the query. As $Q, K, V$ can be expressed as vectors, the output value can be computed via matrix multiplications. Fig. 1 shows this implementation. Additionally, [7] proposes to parallelize the attention mechanism. By computing multiple separate attention outputs in parallel, each representing a *head*, it is possible to obtain a *multi-head self-attention mechanism* that increase the processing performances, as illustrated in Fig. 1.

In order to apply the attention mechanism to Image Classification, it is necessary to formulate this problem as a sequence. In particular, the solution proposed by Swin Transformer [8] is to subdivide the input image into multiple non-overlapping windows (or *patches*) and to encode the pixels present in each window as a vector. The encoded vector is then fed to a multi-head attention mechanism (detailed in Fig. 2) and the output vectors of neighbouring patches are merged together. The ensemble of the attention mechanism and of the patch merging constitutes the basic block of the architecture. The Swin Transformer is composed of multiple consecutive blocks. This subdivision allows a hierarchical partitioning of the image, similarly to what happens in a canonical CNN structure. For this reason, [8] proposes the Swin Transformer also as backbone for networks employed in different tasks, such as Instance and Semantic Segmentation.

### B. Fault Analysis on CNNs

So far, CNNs have been the main target of reliability investigations targeting the reliability of DNNs, especially at the software level. Even if a convolutional layer offers a great deal of parameter over-provisioning and redundancy, it is vulnerable to faults, especially if the parameters are represented as 32-bit floating point [2]. Since a fault in the most significant bit of the exponent produces extremely large values, out of their typical range, many have noted that a good solution is to take advantage of the non-linear ReLU operation, adding an upper limit to the propagated value [5], [10]. This extremely low-cost solution is capable of greatly reducing the number of faults that propagate to the network's output, at the cost of a reduced network accuracy.

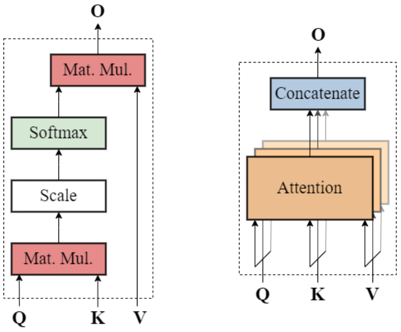


Fig. 1: The attention mechanism (left) and its multi-head extension (right) as presented in [7]

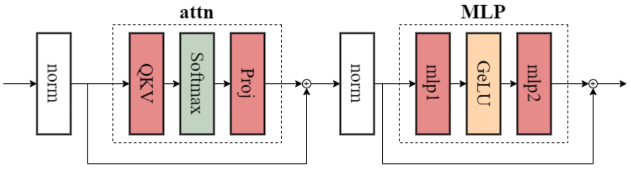


Fig. 2: The Swin Transformer Block

This latter solution is not effective against faults that do not produce significant variations in elements of the network. For this reason, [11] proposes to use Open-Set Recognition, typically used for recognizing out-of-distributions images, to identify those faults that change the network prediction and do not cause out-of-range values. In particular, by computing some metrics on the output of the network, this work is able to identify the majority of faults affecting the network. Also in this case, however, the accuracy of the network is affected.

A different solution, that that does not apply only to faults affecting the most critical bit of the exponent in 32-bit floating-point networks, has been proposed in [4]. In this paper, the authors propose to use well-known Algorithm-Based Fault Tolerance (ABFT) techniques to strengthen the matrix operations used to implement the convolution operations. The core idea of ABFT is to identify and correct errors in matrix multiplications by adding a checksum. However, since this can significantly increase the computational costs, they propose to replace one column and one row of the original matrices with a checksum. Similarly to the constrained ReLU solution, however, the ABFT proposed in [4] has an impact on the network accuracy, albeit in an acceptable manner.

### C. Fault Analysis on Transformers

As Transformer-based architectures are much more recent than CNNs, there have been fewer research studies investigating the fault resilience of these models. In particular, recent works have focused on understanding how to extend fault mitigation mechanisms already present on different DNNs architecture to transformers. The work presented in [9] studies how multiple bit-flips on certain layers' output may affect the performance of the network. Thanks to this, they are able to assign a vulnerability factor to each layer. Furthermore, they propose to increase the reliability of the network by using well-known techniques for CNNs, such as range restriction for non-linearities [5] and algorithm-based fault tolerance (ABFT) [4] for linear computations. As this latter method can be quite expensive, they propose to apply it only to the most vulnerable layers of the network that have a higher vulnerability factor.

|  | Block Analysis |  |  |  |  |  | Transformer Analysis |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| **Fault Model** | **patch_embed** | **layer_0** | **layer_1** | **layer_2** | **layer_3** | **head** | **attn.qkv** | **attn.porj** | **mlp.fc1** | **mlp.fc2** |
| **Stuck-at Parameters** | 6,620 | 6,784 | 6,782 | 6,826 | 6,778 | 6,776 | 6,760 | 6,756 | 6,764 | 6,764 |
| **Bit-flip Neurons** | 6,468 | 6,762 | 6,769 | 6,769 | 6,768 | 5,584 | 6,768 | 6,766 | 6,769 | 6,769 |

TABLE I: Number of faults injected in each layer in order to achieve an error margin $e = 1\%$ and a confidence level of $90\%$.

## III. Proposed Solution

This work proposes an analysis of a state-of-the-art Transformer DNN used for Image Classification: Swin Transformer [8]. Similarly to other DNNs, this particular architecture is mainly defined by a collection of *weights* (i.e., the network *parameters*), grouped in *layers* and in *blocks* of layers. The output of each layer is a multidimensional tensor, where each element is called a *neuron*. The network produces as output an $n$-dimensional *score* vector, where $n$ is the number of classes. The $j^{\text{th}}$ element of this vector represents the probability that the input image belongs to class $j$.

The first objective of this work is to propose an analysis on how different blocks of layers behave in presence of faults, to understand whether there is a link between the depth of the network and its fault resilience. After that, this work focuses on studying the transformer block (Fig. 2) in presence of faults, comparing it to the convolution operation. Having a clear picture on the fault resilience of the Swin Transformer, this work proposes a solution to increase the resistance of the network to faults.

As discussed in Section II-B, an inexpensive solution that greatly decreases the impact of critical faults is to put an upper bound on the ReLU activation function [5]. While this has also been done to improve Transformers reliability [9], the advantages of this technique are not clear, mainly due to the lack of activation functions between Fully Connected (FC) layers. For this reason, this paper proposes to apply a *non-linear filtering operation to all the FC layers of the network*. The core idea is to set out-of-bound inputs and weights to zeros, while clipping out-of-bound output values. The idea of replacing faulty values with zeros has also been explored by [12], when dealing with segmentation CNNs.

However, in an FC layer, imposing a range restriction is not as effective as in a Convolutional layer: the lack of a convolution means that the faulty value of a neuron is not mitigated by its neighbours' values. As such, differently from what previously proposed, values that are out-of-bound (i.e., that are outside the imposed range) should not be simply clipped, rather an effort should be made to reduce the faulty value impact on the network computations. In this optic, the proposed solution details a more refined range restriction technique to obtain a more exact estimate of the neuron output value. In particular, the range restriction technique is applied to the inputs, outputs and weights of Fully Connected layers, as described in the following sections.

### A. Weight Range Restriction

Since in fully connected layers a weight is used only to compute the value of a single output neuron, a faulty out-of-bound weight can be replaced with a 0, preventing it from contributing to the network output. Given an FC layer with $N$ input features and $M$ output features, the value of an output neuron $o_n$ can be computed as:

$$o_m = \sum_{n=o}^{N} i_n \cdot w_{n,m} \tag{1}$$

Assuming that a fault affect weight $w_{\dot{n},m}$ makes it go out of bound, a more accurate estimate of the golden output $o_m$ can be obtained by voiding its contribution, setting $w_{\dot{n},m} to 0$. As such, we can rewrite (1) as:

$$o_m = \sum_{\substack{n=o \\ n \neq \dot{n}}}^{N} i_n \cdot w_{n,m} \tag{2}$$

The reader should note that (1) and (2) are equivalent in absence of faults, since the weights values are known at inference time. This means that this weight range restriction technique does not affect the accuracy of the network in absence of faults.

### B. Input Range Restriction

A similar reasoning can be extended to the input values. However, differently from the weight: (i) a single input neuron contributes to multiple outputs and (ii) modifying an out-of-bound input can change the network output in absence of faults. Therefore, while an out-of-bound input neuron has a bigger impact on the network, changing all the layer's output neurons, it can be treaded the same as a weight out of bound and set to 0.

For this reason, if an input neuron $i_{\overline{n}}$ is out-of-bound, (2) can be extended to:

$$o_m = \sum_{\substack{n=o \\ n \neq \overline{n}, \dot{n}}}^{N} i_n \cdot w_{n,m} \tag{3}$$

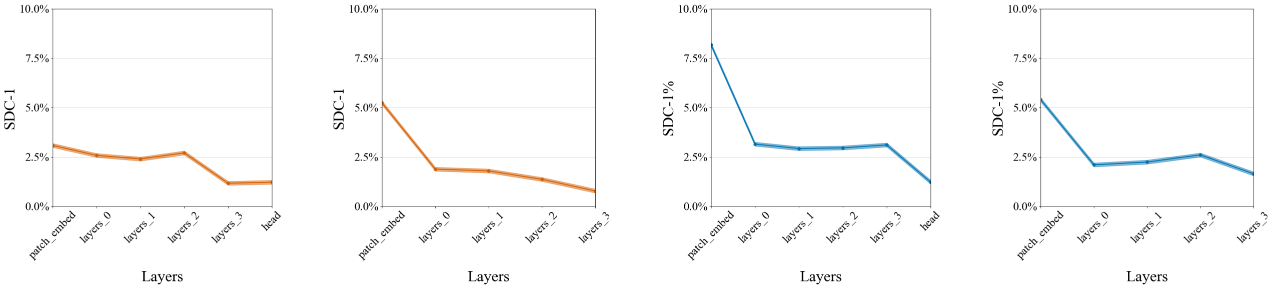
### C. Output Range Restriction

As a last measure, a range restriction is applied to the output of the layer. This is necessary, as the value of the computed with the weight and input restriction might still be over-estimating the actual value of the neuron. For this, we implement a solution that takes advantage of a linearized version of the hyperbolic tangent, the *hard hyperbolic tangent* function $htanh$:

$$htanh(x, x^U, x^L) = \begin{cases} x & \text{if } x > x^L \text{ and } x < x^U \\ x^U & \text{if } x \geq x^U \\ x^L & \text{if } x \leq x^L \end{cases} \tag{4}$$

In particular, in the context of output filtering, this function can be applied in conjunction with (3):

$$o_m = htanh\Big( \sum_{\substack{n=o \\ n \neq \overline{n}, \dot{n}}}^{N} i_n \cdot w_{n,m},\ o^U,\ o^L \Big) \tag{5}$$



(a) SDC-1 s@ Parameters (b) SDC-1 bit-flip neurons (c) SDC-1% s@ Parameters (d) SDC-1% bit-flip neurons

Fig. 3: Block Reliability of the different blocks of Swin Transformer for stuck-at faults in the network parameters and bit-flip in the output neurons. The graphs also report the observed error margin deriving from the statistical observation.

Where $o^U$ and $o^L$ are the upper and lower bound. The reader should note that, compared with traditional range restriction techniques, the proposed methodology provides a better estimate of the output of a neuron. The objective of restricting weights and inputs is not to protect their value, but rather to provide a more accurate estimate of the output of a neuron. This is possible because, *differently from a traditional CNN, the core layer of the Transformer is a linear layer*. This means that, it is possible to isolate the contribution of every single weight and input. Therefore, when a weight-input pair is deemed to be harmful (i.e., out of range), its contribution can be nullified by setting its value to $0$.

### D. Profiling

For the input and output restriction processes, it is important to individuate, at run time, the correct upper and lower bounds for each layer. This can be easily done with a *profiling* phase, where the intermediate layer results are scanned to individuate the minimum and the maximum values of each input and output neuron. To have an unbiased estimate of these values, it is important to have a large dataset to be processed, disjoint from the one used to evaluate the range restriction method effectiveness.

The reader should note that bound values can be either specific for a single element or relative to multiple elements. For example, it is possible to have an output bound that is the same for all the neurons of a layer, while having an input bound that is different for each neuron.

## IV. Case Study

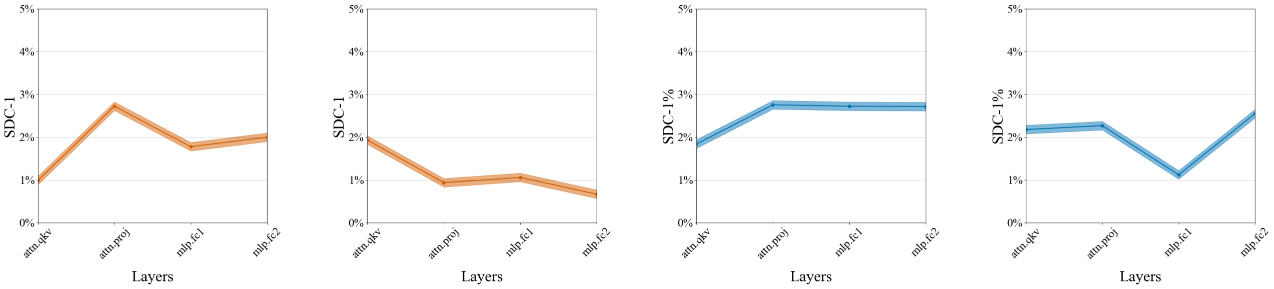
As previously discussed, the aim of this work is to study how the Swin Transformer reacts to faults and how to improve its resilience. There are multiple variants of this network, characterized by a different number of parameters. Since all the variants have all the same identical structure, this work studies the Tiny variant, that has the lower number of parameters. As the target is to study the architecture and the impact of the Transformers in terms of network reliability, studying a larger version is more computationally expensive and does not offer additional insights. Tiny Swin Transformer is a Transformer Neural Network composed of $29 \cdot 10^6$ parameters that reaches an ImageNet top-1 accuracy of $81.3\%$. The network used in this paper is pre-trained as described in [8].

The fault injection process has been carried out using a fault injection tool called SCI-FI [13], a PyTorch-based software fault injector that speeds up the injection process by implementing two techniques: the Fault Dropping and the Delayed Start. The first consists in dropping faults that have no impact after the layer they are injected in, while the second allows the inference to start from the computation of the layer affected by the fault, avoiding the computation of previous layers.

The dataset used for the reliability analysis and for the fault mitigation is the ImageNet validation set, composed of $50,000$ images, $50$ for each of the $1,000$ classes. To avoid adding any bias to evaluation of the fault mitigation efficacy, this dataset is further divided into two equally sized subsets: the first split is used to evaluate the reliability of the network and of the proposed mitigation technique, while the second has been used for the profiling described in Section III-D.

To have a more comprehensive view of the whole process, this paper studies the behaviour of the network under two well-known single fault models: stuck-at faults in the network parameters and bit-flip in the network output neurons. It is true that memories can be protected with error correction codes (ECC) mechanisms, but not all devices come with this extra circuitry, especially low-power and low-cost AI-oriented edge devices. Then, this justifies the adoption of the above-mentioned fault models.

A fault consists in a random bit of the weight or of the output being set to a fixed value (stuck-at) or to its inverse (bit-flip). Due to the dimensions of the targeted neural network, an exhaustive fault injection campaign would have been too expensive to be performed. For this reason, this work studies Swin Transformer by using the *statistical* approach described in [14]. In particular, instead of injecting all the possible $N$ faults, only a subpopulation of $n$ faults is injected in order to achieve a target error rate $e = 1\%$ with a target confidence of $90\%$ on the observed metric. For the reliability analysis, since the aim is to perform a block-wise study, this sampling process is done at a block level. This means that the population $N$ is relative to all the faults injectable in that block, and that for each block a different sample $n$ is chosen. Further details on the sizing of the fault injection campaigns are reported in Table I.



(a) SDC-1 s@ Parameters (b) SDC-1 bit-flip neurons (c) SDC-1% s@ Parameters (d) SDC-1% bit-flip neurons

Fig. 4: Reliability of the different component of Swin Transformer for stuck-at faults in the network parameters and bit-flip in the output neurons. The graphs also report the observed error margin deriving from the statistical observation.

## V. Experimental Results

The analysis of Swin Transformer is split in three parts. First, an analysis on how different blocks behave when affected by faults is presented, to have an overview on the fault resilience of the network. Then, the analysis moves to a deeper level, focusing on the fault resilience of the various components of the Transformer blocks. Finally, this section discusses the effectiveness of the proposed range restriction mechanism in reducing the impact of faults.

### A. Block Reliability

The first objective is to determine how resilient are the different blocks of the network. The main reason for this investigation is to locate the most critical high-level component of the Swin Transformer architecture. Additionally, the experiments discussed in this section aim to show the masking ability of each component. To investigate the criticality, we measure the percentage of faults that change the golden network prediction, often referred to as Silent Data Corruption 1 (*SDC-1*). At the same time, to evaluate the masking ability of each layer we measure the *SDC-1%*, which measures how many faults change the score of more than $1\%$ of the golden value. This means that a layer with an SDC-1% of $0\%$ has a near perfect masking ability, as all the faults change the score associated with the predicted class by less than $1\%$. The reader should note that, while an SDC-1 fault is not necessarily also an SDC-1%, in our experiments $99.33\%$ of all the SDC-1 faults are also SDC-1%.

Tiny Swin Tensor is composed of a patch-embedding layer (`patch_embed`) followed by 4 transformers blocks (`layers_0` trough `layers_3`); the output of the last block is finally processed by an FC layer (`head`). The complete results of the FI campaigns are shown in Fig. 3. It is interesting to observe from Fig. 3a and Fig. 3b that the patch-embedding block is more susceptible than the other blocks to faults in the output neurons. This is probably due to the fact that the patch-embedding block is the only one of the network that contains a convolutional layer. Another interesting observation that can be made is that all the blocks react similarly to faults in the parameters and faults in the neurons, beside the patch-embedding. Yet again, this can be explained by the presence of the convolutional operation. Finally, it is also possible to observe how (Fig. 3c and Fig. 3d) the SDC-1% rate for every block is similar to the SDC-1 rate. For example, in `layer_0`, for the parameters fault model, we can observe that only $2.58\%$ of faults are SDC-1 and $3.15\%$ of faults are SDC-1%. This means that: (i) *more than* $96\%$ *of faults have a negligible impact* and (ii) *the majority of the faults that have a significant impact (i.e., are SDC-1%) also change the network prediction.*

### B. Transformer Reliability

As illustrated in Fig. 2, the Swin Transformer block is composed of 2 main parts, interleaved by normalization layers. In particular, the first part is devoted to the computation of the attention (`attn`) and is formed by two FC layers and a Softmax activation function. The second part is a Multi-Layer Perceptron (`MLP`) composed of two fully connected layers and a GeLU activation layer. For each part (`attn` and `MLP`), it is also present a residual connection that concatenates their outputs with their inputs.

The results of the fault injections in this Transformer are reported in Fig. 4. As observed for the Block Analysis (Section V-A), the number of SDC-1 and SDC-1% are very similar for each component of the Transformer, hinting that most faults are either SDC-1 or they have a negligible effect on the output.

Concerning the faults affecting the parameters (Fig. 4a), it does not emerge a clear pattern of criticality: the first layer (`attn.qkv`) seems to be the least susceptible of all, while the following projection layer is the most susceptible. The reason why `attn.qkv` is the least susceptible to faults on the weight is because extremely high output and, in general, out-of-bound values are masked by the Softmax function, that normalizes values in the [0, 1] range.

From Fig. 4b the reader can observe how deeper layers have a higher resilience to faults. However, from Fig. 4d it appears clear that all layers have a similar percentage of SDC-1% faults. This suggests that deeper layers in the Transformer are better at masking faults. The higher percentage of SDC-1 faults measured for the `attn.qkv` block when it is affected by neurons faults, compared to when it is affected by parameters faults, is due to the fact that a neuron fault affecting the Softmax layer is classified as affecting the `attn.qkv` block.

### C. Fault Mitigation

For the proposed fault mitigation techniques, we first executed a profiling inference run to get: (i) the maximum and minimum value of each weight tensor of the network and, (ii) the maximum

| **Layer** | **Parameters<br>stuck-at** | | | **Neurons<br>bit-flip** | | | **Overhead<br>[%]** |
|---|---|---|---|---|---|---|---|
| | **Unprotected<br>SDC-1<br>[%]** | **Protected<br>SDC-1<br>[%]** | **Improv.** | **Unprotected<br>SDC-1<br>[%]** | **Protected<br>SDC-1<br>[%]** | **Improv.** | |
| patch.embed | 3.08 | $9 \cdot 10^{-4}$ | x$3 \cdot 10^3$ | 5.24 | 2.76 | x1.89 | - |
| layers_0 | 2.58 | $2 \cdot 10^{-5}$ | x$1 \cdot 10^5$ | 1.89 | 0.86 | x2.19 | - |
| layers_1 | 2.40 | $5 \cdot 10^{-6}$ | x$5 \cdot 10^5$ | 1.80 | 0.80 | **x2.25** | - |
| layers_2 | 2.71 | $2 \cdot 10^{-6}$ | x$\mathbf{1 \cdot 10^6}$ | 1.37 | 0.83 | x1.65 | - |
| layers_3 | 1.17 | 0 | - | 0.78 | 0.41 | x1.90 | - |
| head | 1.22 | 0 | - | 1.54 | 1.52 | x1.01 | - |
| **network-wise** | 2.49 | $2 \cdot 10^{-5}$ | x$\mathbf{1 \cdot 10^5}$ | 1.73 | 0.87 | **x1.99** | **68.61** |

TABLE II: Layer-by-layer improvements provided by the presented fault mitigation technique with the added overhead.

and minimum value of each input neuron for each layer and (iii) the maximum and minimum value of each output neurons for each layer. This allows us to correctly apply (5) on the first split of validation dataset, as described in Section IV.

To apply the proposed technique to all the linear layers of the network, (5) has also been applied to the convolutional layer inside the patch-embedding layer. As shown in Table II, different blocks react differently to the proposed mitigation technique. For what concerns stuck-at faults on the network parameters, the efficacy of the method improves the deeper the block is: faults are completely masked starting from the last Transformer block (`layers_3`). In general, the effect of critical faults is diminished by several orders of magnitude: among all the faults injected in the network, only the $2 \cdot 10^{-5}\%$ changes the protected network output, against the $2.49\%$ of faults in the unprotected network. Overall, we observe an improvement of a factor of $1 \cdot 10^5$.

For what concern the output neuron fault model, the effect of the proposed technique is more modest, albeit still significant. Contrarily from what observed for the stuck-at faults in the network parameters, there is not a direct correlation between the benefits provided by the range restriction and the depth of the model. Curiously, we observe how the proposed fault mitigation technique does not increase the resiliency of the last layer. This is, however, to be expected since the output in this layer is the final score of the network, hence a change in this layer necessarily changes the final score. Overall, the number of critical faults is halved, as the improvement factor offered by the proposed technique is of $1.99$.

The main drawback of this technique is the overhead added by the range restriction. While we consider the profiling cost to be negligible, as it simply requires an inference over the first split of the dataset, the total overhead added by the proposed technique, in terms of additional time required for a network inference, is of $68.61\%$. However, the reader should note that the reason for such a high cost can be ascribed to the high-level implementation of the range restriction: the result shown in this section are obtained by implementing (5) at PyTorch level. Better results, in terms of added computational time, can be obtained with a low-level CUDA specific implementation. Furthermore, it is possible to change the impact of the proposed methodology by changing the granularity at which the range restriction is applied. In fact, applying range restriction individually to each neuron is much more expensive than applying a layer-wise range restriction. Therefore, the granularity at which the proposed solution is applied acts as a trade-off between the efficacy of the method and its computational cost.

The complete results of the hardening techniques are reported in Table II. Using the proposed technique, the accuracy of the Swin Transformer on the second split of the validation dataset is reduced from $86.09\%$ to $85.21\%$. We believe that this can be viewed as an acceptable result in terms of accuracy loss, as it is outweighed by a substantial increase in fault tolerance. The reader should note that, however, by multiplying the maximum bound value for a coefficient $\alpha > 1$ and the minimum for a coefficient $\beta < 1$ it is possible to reduce the accuracy loss. Clearly, using these parameters negatively affects the efficacy of the ranging technique.

## VI. Conclusions and Future Directions

The rapid advancement in how DNNs are implemented requires new analysis and new fault mitigation techniques. In particular, as Transformers are rapidly replacing CNNs as the optimal choice for many tasks, an assessment of their reliability is needed. While fault mitigation techniques developed for CNNs can be extended to Transformer, ad-hoc solutions can provide further improvements. For these reasons, this work proposes an analysis of Swin Transformer, a state-of-the-art Transformer-based DNN used for Image Classification. Additionally, this work proposes to apply range restriction techniques as a heuristic to obtain a reliable estimate of neuron outputs. While this technique proves to be effective for different kinds of fault models, its cost is not negligible. Therefore, future works will include an improvement of its computational cost, by means of a specifically designed CUDA kernel. Furthermore, as this approach can also be applied to convolutional layers, a further investigation will be on the application of the proposed mitigation to canonical and known CNN architectures. A final direction worth exploring is the identification of the most critical neurons in Transformer DNNs: since putting these neurons to 0 could be counter-productive, a more specific solution should be devised.

## Acknowledgment

This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

# References

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

[2] A. Bosio, P. Bernardi, A. Ruospo, and E. Sanchez, “A reliability analysis of a deep neural network,” in *2019 IEEE Latin American Test Symposium (LATS)*, 2019, pp. 1–6.

[3] J. E. R. Condia, J.-D. Guerrero-Balaguera, F. F. Dos Santos, M. S. Reorda, and P. Rech, “A multi-level approach to evaluate the impact of gpu permanent faults on cnn’s reliability,” in *2022 IEEE International Test Conference (ITC)*, 2022, pp. 278–287.

[4] F. F. d. Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, “Analyzing and increasing the reliability of convolutional neural networks on gpus,” *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, 2019.

[5] Z. Chen, G. Li, and K. Pattabiraman, “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)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2021, pp. 1–13. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/DSN48987.2021.00018

[6] L.-H. Hoang, M. A. Hanif, and M. Shafique, “Ft-clipact: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation,” in *Proceedings of the 23rd Conference on Design, Automation and Test in Europe*, ser. DATE ’20. San Jose, CA, USA: EDA Consortium, 2020, p. 1241–1246.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[8] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” *CoRR*, vol. abs/2103.14030, 2021. [Online]. Available: https://arxiv.org/abs/2103.14030

[9] X. Xue, C. Liu, Y. Wang, B. Yang, T. Luo, L. Zhang, H. Li, and X. Li, “Reliability analysis of vision transformers,” 2023. [Online]. Available: https://arxiv.org/abs/2302.10468

[10] F. Angione *et al.*, “Test, reliability and functional safety trends for automotive system-on-chip,” in *2022 IEEE European Test Symposium (ETS)*, 2022, pp. 1–10.

[11] G. Gavarini, D. Stucchi, A. Ruospo, G. Boracchi, and E. Sanchez, “Open-set recognition: an inexpensive strategy to increase dnn reliability,” in *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2022, pp. 1–7.

[12] S. BurelT, A. EvansT, and L. Anghel, “Improving dnn fault tolerance in semantic segmentation applications,” in *2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2022, pp. 1–6.

[13] G. Gavarini, A. Ruospo, and E. Sanchez, “Sci-fi: a smart, accurate and unintrusive fault-injector for deep neural networks,” in *2023 European Test Symposium*, 2023, In press.

[14] A. Ruospo, G. Gavarini, C. D. Sio, J. Guerrero, L. Sterpone, M. S. Reorda, E. Sanchez, R. Mariani, J. Aribido, and J. Athavale, “Assessing convolutional neural networks reliability through statistical fault injections,” in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023, [In press].