POLITECNICO DI TORINO
Repository ISTITUZIONALE

Self-supervised pretraining and quantization for fault tolerant neural networks: friend or foe?

Original
Self-supervised pretraining and quantization for fault tolerant neural networks: friend or foe? / Milazzo, R., Fosson, S.M., Morra, L., Sterpone, L.. - In: IEEE ACCESS. - ISSN 2169-3536. - 13:(2025), pp. 75546-75562. [10.1109/access.2025.3564834]

Availability:
This version is available at: 11583/2999699 since: 2025-04-30T08:14:18Z

Publisher:
IEEE

Published
DOI:10.1109/access.2025.3564834

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

20 November 2026

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 27 March 2025, accepted 6 April 2025, date of publication 28 April 2025, date of current version 6 May 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3564834

RESEARCH ARTICLE

Self-Supervised Pretraining and Quantization for Fault Tolerant Neural Networks: Friend or Foe?

ROSARIO MILAZZO, SOPHIE M. FOSSON, (Member, IEEE), LIA MORRA, (Senior Member, IEEE), AND LUCA STERPONE, (Senior Member, IEEE)

Department of Control and Computer Engineering, Politecnico di Torino, 10129 Turin, Italy

Corresponding author: Rosario Milazzo (rosario.milazzo@polito.it)

This work was funded by the European Union-NextGenerationEU, Mission 4 Component 2, under Grant ECS00000036-CUP E13B22000020001.

ABSTRACT Deep neural networks (DNNs) are increasingly being applied in critical domains such as healthcare and autonomous driving. However, their predictive capabilities can degrade in the presence of transient hardware faults, which can lead to potentially catastrophic and unpredictable errors. Consequently, various techniques have been proposed to enhance DNN fault tolerance by modifying the network structure or training procedure, thereby reducing the need for costly hardware redundancy. However, there are design or training choices whose impact on fault propagation has been overlooked in the literature. Specifically, self-supervised learning (SSL) as a pre-training technique has been shown to enhance the robustness of learned features, resulting in improved performance on downstream tasks. This study investigates the error tolerance of different DNN and SSL techniques in image classification and segmentation tasks, including those relevant to Earth Observation. Experimental results suggest that SSL pretraining, whether used alone or in combination with error mitigation techniques, generally enhances DNN fault tolerance. We complement these findings with an in-depth analysis of the fault tolerance of quantized networks. In this context, the use of standard SSL techniques leads to a decrease in accuracy. However, this issue can be partially addressed by employing methods that, during the pretraining phase, incorporate the quantization error into the loss function.

INDEX TERMS DNN, error mitigation, fault-tolerance, machine learning, quantization, reliability, resilience, self-supervised learning.

I. INTRODUCTION

In recent years, Deep Neural Networks (DNNs) have become ubiquitous in a variety of fields, including safety-critical applications such as healthcare [1] and self-driving vehicles [2]. Despite their distributed and parallel nature, DNNs are not intrinsically tolerant to radiation-induced transient faults [3], [4], [5]. In particular, in the absence of fault mitigation techniques, the accuracy of Convolutional Neural Networks (CNNs) was shown to drop dramatically even at low fault rates. The probability of errors is further exacerbated by the large size and parameter count of modern CNNs, as well as the complexity of the hardware on which they run. As technology scaling reduces operating voltages and increases integration density, modern hardware becomes more susceptible to such events. Soft errors have been shown to dominate the failure profile of advanced systems, with non-negligible failure-in-time (FIT) rates, defined as the number of failures expected in 10^9 device-hours [6]. These concerns are particularly pronounced in GPU-accelerated systems used for DNN inference and training. Extensive fault injection and beam testing campaigns have demonstrated the vulnerability of modern GPUs to radiation-induced faults under realistic execution conditions. For instance, experiments on NVIDIA Kepler and Fermi architectures have revealed silent data corruption and functional interruption rates in the order of 10^3 FIT for common parallel workloads [7]. Faults affecting register files, caches, and schedulers can propagate

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

© 2025 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

across threads and warp-level parallelism, causing critical errors in algorithmic outputs. These findings substantiate the relevance of fault-tolerant design techniques, particularly when targeting embedded or spaceborne inference systems. Since fault-induced mispredictions can lead to substantial damages and economic losses, it is mandatory to develop techniques to effectively measure fault-induced errors in DNNs [4], as well as increase their fault tolerance [3], [8], [9], [10], [11]. Replication and redundancy, either at the hardware or software level, are effective but expensive solutions [12]. However, recent work has shown that ad hoc modifications of a CNN architecture and/or training procedure may achieve a better trade-off between accuracy and fault tolerance. In particular, several techniques have been proposed to improve CNN's resilience to faults by constraining the activation range of intermediate layers [8], [9], [10], [11].

Experiment-wise, the current literature has so far largely focused on simple benchmarks, such as CIFAR-10 or CIFAR-100, relatively shallow CNNs, and standard training procedures. However, *modern CNNs are large and are typically trained on vast datasets using a variety of complex techniques*. Since DNNs require large amounts of training data, they are seldom trained from scratch in real-life applications, but are rather fine-tuned after pretraining on a large scale dataset such as ImageNet [13]. More recently, self-supervised learning (SSL) techniques are increasingly gaining traction as an alternative to supervised pretraining [14]. Since SSL methods do not require labels, they can scale up to extremely large scale datasets. *CNNs pretrained via SSL were shown to achieve better performance in downstream tasks compared to standard ImageNet supervised pretraining* [14], [15]. At the same time, *networks pre-trained using different techniques can have substantially different weight distributions* [16], so we cannot assume that their fault tolerance is equivalent, even when their accuracy might be similar.

Another critical factor is the computational capacity of edge devices, which in most cases have limited dedicated hardware; as a result, DNNs must be optimized using techniques such as pruning or quantization [17], [18]. However, the effectiveness of quantization depends on the training strategy and may be negatively affected by SSL pretraining [16]. Overlooking these important aspects could result in *lost opportunities* and *hidden threats*, depending on how design choices and operating scenarios amplify or dampen fault propagation in DNNs.

Moving from these premises, our work aims at extensively benchmarking the fault tolerance of CNNs under various conditions in terms of training dataset, architectures, task, choice of SSL pretraining and quantization applied. We evaluated the impact of these factors both on the original network (unmitigated scenario) and in combination with fault mitigation techniques based on activation clipping. Experiments were conducted on several realistic benchmarks, from the well-known ImageNet dataset to several Earth Observation (EO) classification tasks. More specifically, our main contributions are as follows:

- we evaluate the **fault tolerance of different CNNs, both in terms of architecture and depth**, trained in a supervised setting: experimental results show that although activation clipping can in general enhance their fault tolerance without sacrificing performance, adaptive thresholding strategies can result in a performance drop at very low fault rates;
- we examine the positive **effect of SSL pretraining on fault tolerance**, showing that SSL pretraining, especially when combined with activation clipping, improves fault tolerance compared to supervised pretraining;
- we further examine **the effect of quantization, alone and in combination with SSL pretraining**, showing that quantization substantially increases fault tolerance at the expense of accuracy. In quantized CNNs, SSL pretraining generally results in lower accuracy, which is only partially remedied by quantization-aware SSL;
- we compare results on different datasets, highlighting important differences depending on the training set size and the complexity of the classification task.

The present work extends our previous contribution [19] in several ways: first, we include more CNN architectures and SSL techniques; second, we investigate quantization and quantization-aware SSL techniques, which were not considered in the previous work.

The rest of the work is organized as follows. In Section II, we give a brief overview of the main error mitigation techniques and introduce the SSL and quantization techniques. The fault injector, fault mitigation and quantization techniques used in our experiments are analyzed in Section III. The datasets and experimental settings are discussed in Section IV. In Section V, we compare the fault tolerance of various DNNs. Section VI compares the performance of different configurations based on SSL pretraining at multiple fault rates. In Section VII, we analyze the impact of activation clipping on baseline accuracy, highlighting the differences between standard supervised training and SSL-pretrained models. Then, the experiments concerning quantized networks are described in Section VIII. In Section IX, we investigate fault tolerance in segmentation tasks, evaluating the effect of SSL pretraining and activation clipping on model robustness. Finally, in Section X brief conclusions are drawn and future works are introduced.

II. RELATED WORK

A. FAULT TOLERANCE

In recent years, a multitude of studies have explored fault tolerance in DNNs. These studies have yielded essential guidelines for constructing robust models, including the use of compact data types for storing network parameters, fortifying vulnerable bits, or introducing normalization techniques [5].

Hardware redundancy, such as Triple Modular Redundancy [20] and Dual Modular Redundancy, comes with

significant hardware overhead, making it impractical for resource-constrained scenarios. Overhead can be reduced by employing statistical methodologies to identify susceptible hardware components [21], or adopting error detection and correction techniques, such as Error Correction Codes [22]. Similarly, Approximated Triple Modular Redundancy [23], leveraging residual quantization, has been introduced to improve CNN resilience. Additionally, HPR-Mul [24], an energy-efficient redundancy-based multiplier, employs approximate computing to enhance fault resilience.

A complementary approach tackles the problem at the software level, by looking at the design and training of the network itself [8], [9], [11], [25]. The most obvious advantage is reducing or even eliminating overhead at execution time, but the increased fault tolerance may come at the expense of decreased accuracy. Software-level approaches include fault-aware training (FAT) [8], [25], architectural modifications [26], and activation clipping [8], [9], [11]. FAT involves training DNNs to classify accurately in the presence of faults by simulating them during training. It requires retraining of the entire model and access to training data, and was shown to be less effective than architectural modifications [8], [27]. Architectural modifications can involve pruning less impactful nodes, adding extra nodes in the most critical parts of the architecture, or repositioning batch normalization layers.

Finally, recent studies have established the effectiveness of activation clipping in limiting error propagation [8], [9], [10], [11]. Activation clipping can be applied post-training, and does not introduce hardware-level overhead. Different methods, such as Ranger [11] and ClipAct [9], have been developed to optimize clipping thresholds for each layer, striking a balance between accuracy and fault tolerance. These advances collectively represent the current state of the art in fault tolerance for DNNs.

B. FAULT INJECTION

Single Event Effects (SEEs) in circuits encompass a range of phenomena linked to the interaction of energetic particles with the silicon substrate. Energetic particles, such as those emitted by the sun, consist of a multitude of charged and discharged particles, including protons, heavy ions, and neutrons. The different types of SEE can be divided into two main categories: destructive and non-destructive. We focus on Single Event Upsets (SEUs), that are non-destructive and transient faults affecting memories, leading to bit flips that can be rectified by rewriting the data in memory. Low-level fault injectors, such as NVBitFI [28] which inject errors into destination register values after execution, and GPU-Qin [29] working at the assembly instruction level, simulate realistic faults but require significant execution time.

On the other hand, software-level fault injectors are faster to execute, but necessitate a simplified fault model. Numerous tools have been developed to streamline the customization and integration of fault injection into DNNs (TensorFI [30] and PytorchFI [31]). PyTorchFI, specifically, is a runtime perturbation tool designed for DNNs using the PyTorch deep learning platform. While PyTorchFI allows users to perturb weights or activation values of a DNN, it comes with certain limitations. For instance, it only permits modifying a single bit of activation values or altering a weight with a pre-established value, overlooking essential cases such as the insertion of single or multiple bit-flips in the weights.

C. SELF-SUPERVISED LEARNING

SSL is a branch of machine learning that enables a DNN model to learn agnostic feature representations from unlabeled datasets. Such features can then be fine-tuned on a plurality of tasks, such as classification or object detection, achieving competitive and robust performance with small amounts of labelled data [14]. Although many classes of SSL techniques have been proposed, the most effective, and thus popular, ones are *contrastive* methods. They learn representations by comparing positive and negative examples, aiming at pulling similar samples closer in feature space while pushing diverse samples far from each other. In this work, we investigated three different contrastive SSL techniques. **SwAV** (Swapping Assignments between Views) [32] is based on clustering images ensuring consistency between several augmented versions of the same image. **SimSiam** [33] is based on maximizing similarity between representations of the same image or instance across different augmented versions, aimed at enhancing model generalization and robustness. **DINO** [34], on the other hand, is based on a teacher-student framework, in which the teacher and student networks take as input the same image with different augmentations applied. The gradients are propagated only through the student network and the teacher parameters are updated through an exponential moving average of the student parameters. Relying heavily on augmentations, contrastive SSL methods can enforce invariance with respect to common transformations (such as cropping, translation, scale, and intensity manipulation), yielding more robust features. While several works have investigated the positive effects of SSL methods from the point of view of accuracy on downstream tasks [14], quantization [16] and robustness to adversarial examples or label corruption [15], to the best of our knowledge fault tolerance properties have not been investigated.

D. QUANTIZATION

A standard deep neural network typically operates with weights represented in 32-bit floating-point values. However, FP32 computations demand either a floating-point unit or additional hardware resources for dynamic range shift computation, resulting in increased hardware requirements and latency. DNN quantization offers a solution to this dilemma. DNN quantization involves approximating a neural network's parameters and activations to lower bit-widths, for example, 8 bits. Quantization has become the standard for

deploying DNNs on hardware, making it essential to examine it from a fault tolerance perspective. Quantization has several advantages, including reduced memory, lower power consumption, shorter inference times, and compatibility with simpler hardware, while also enhancing the resilience of the DNN model.

Quantization is executed through the following mapping function:

$$Q(x) = \text{round}(r/S - Z) \tag{1}$$

where Z (zero point) and S (scale factor) represent the quantization parameters, and x denotes the FP32 value. The quantization procedure results in a loss of precision or quantization error, indicating the disparity between the original value and the one converted back to FP32.

$$\tilde{x} = (Q(x) - Z)S \tag{2}$$

$$\text{error} = x - \tilde{x} \tag{3}$$

There are several methods for quantizing a model, the simplest being Post-Training Quantization (**PTQ**), which does not require retraining or labeled data, and Quantization-Aware-Training (**QAT**), which requires fine-tuning and access to labeled training data but provides better performance for the same number of bits [17], [18]. QAT mitigates the loss of numerical precision by incorporating a quantization error into the training loss, guiding the network to learn representations that maintain robustness during subsequent quantization.

Pre-trained networks using different techniques, such as SSL methods, can exhibit substantially different weight distributions that sometimes hinder the efficiency of the quantization process, leading to reduced performance [16]. To mitigate this effect, we chose to incorporate an additional SSL technique, referred to as **SSQL** (Synergistic Self-supervised and Quantization Learning) [16], into our analysis. This technique introduces an additional loss component during the SimSiam [33] pretraining phase to simulate quantization errors, thereby improving downstream weight distributions for the quantization process. The current literature have investigated the effects of SEUs on quantized models, generally finding that quantization contributes to improving DNN model resilience. However, the interplay between SSL and quantization and, more in general, between weight distribution and fault tolerance was not extensively investigated.

III. METHODS

This section outlines the primary methods employed for fault injection, fault mitigation, and model quantization, respectively.

A. FAULT INJECTION

Fault injections (FI) was performed at runtime using a customized version of the PytorchFI library [19], [31]. We implemented a custom perturbation model capable of injecting multiple SEUs simultaneously during execution, based on a specified fault rate within the range $[0, 3 \times 10^{-5}]$. The selected fault model is the bit flip, considering both DNN weights and biases as potential fault locations. To simulate the transient effects of faults, we chose to inject random faults into each batch. Unlike the default fault injector in PytorchFI, which set a maximum value that weights could reach in the presence of faults for each layer, we opt for a 32-bit fixed-point representation (1 sign bit, 15 integral bits, and 16 fractional bits). This choice ensured a uniform range of possible values throughout the entire DNN.

Since faults are stochastic in nature, they are by default injected by sampling each potential fault location according to the selected sampling rate, which makes FI campaigns very slow to execute. PytorchFI was modified for greater efficiency by first determining the total number of fault n_{faults} and then randomly assigning each fault to a specific layer. The total number of faults is determined assuming a Binomial distribution with parameters n representing the total number of bits in the entire DNN fault space and p representing the desired fault rate. The faults are then distributed across layers based on the number of bits associated with weights and biases in each layer. Finally, we selected the bit index to be flipped by sampling from a Discrete Uniform Distribution. This modification reduced the complexity from $\mathcal{O}(L \cdot H \cdot W \cdot C \cdot b)$ to $\mathcal{O}(n_faults)$, where L is the number of layers in which we want to inject faults, H and W are the height and width of the convolutional filter, C are the image channels (1 if grayscale or 3 if RGB), and b the number of bits for each weight/bias value (32 in our experiments).

B. FAULT MITIGATION TECHNIQUES

To evaluate the impact of SSL pretraining on fault tolerance we considered the original model without any protection (**Unprotected**) and two versions in which the ReLU activation is replaced throughout the model with a clipped version of the ReLU activation function (**CReLU**), introduced in **ClipAct** [9]. We chose to exclusively utilize fault mitigation techniques based on the clipping of activations. This choice was made to avoid additional modifications to the training phase, aligning it with the pre-trained models that we employed. CReLU maps activation values exceeding a certain threshold T_l to zero, based on the hypothesis that a high activation value is likely induced by a fault, as follows:

$$f(x) = \begin{cases} x & \text{if } 0 \le x \le T_l \\ 0 & \text{if } x > T_l \\ 0 & \text{if } x < 0 \end{cases} \tag{4}$$

The core idea of ClipAct is to determine the optimal value for the threshold T_l in order to maximize fault tolerance at each layer. The corresponding thresholds are effectively treated as learnable parameters, optimized via an iterative algorithm that operates directly on the weights of the trained network. The key metric to be optimized is the accuracy at different fault rates, measured by the Area Under the Curve

(AUC).[1] This approach aims to counteract the significant degradation in accuracy observed when high-intensity faulty activations dominate the results during inference. The method encompasses the following steps:

- Each activation layer is replaced with its corresponding clipped variant. In our specific case, we employ CReLU with adjustable hyper-parameter T_l. Initially, the value of T_l is established by computing the statistical profile of the neural network's activation functions on a subset of the validation dataset, and setting T_l to the maximum value generated by each activation layer. This profile includes the maximum activation value observed for each layer, which serves as the initial threshold. Using this approach, the algorithm ensures that the initial search space encompasses all relevant activation ranges for each layer. The search space, denoted as S, is then defined within the interval $[0, T_l]$.
- The search space S is further subdivided into three equal-sized subintervals, resulting in four possible thresholds: T_1, T_2, T_3 and T_4. These subdivisions aim to identify the regions where the AUC reaches its peak, thereby narrowing the focus on promising ranges of T_l. Each of these threshold values is employed as T_l within the CReLU function, and the model's performance is evaluated on the validation set at each fault rate. This process generates four distinct AUC values, one corresponding to each threshold, effectively capturing the resilience of the network for various clipping levels.
- Subsequently, a new search space S is defined around the most promising threshold from the previous step. For example, if the best AUC is achieved in the prior step using T_3, then S is defined within the interval $[T_2, T_4]$. This iterative narrowing process, inspired by binary search, allows for progressively more precise determination of the optimal threshold.

The last two steps are carried out iteratively for a total of n repetitions. As n increases, we obtain increasingly precise values for the threshold parameters T_l. Since each activation layer may possess a different optimal threshold, the algorithm is executed separately for each layer. The layer-specific optimization ensures that the fault tolerance is maximized globally across the entire network. Faults are injected one layer at a time to determine the most effective threshold. Further details are available in [9].

ClipAct is compared against a simpler approach, referred to as **ActMax** in the following, which sets the threshold T_l as the maximum value calculated for each layers' activations during the initial step of the ClipAct algorithm. The complexity of this method is comparable to that of ReLU6 [8], but: i) the threshold is not fixed, but rather depends on the weight distribution and the layer, and ii) values exceeding the threshold are set to 0, effectively suppressing the feature, whereas in ReLU6 clipped values are set to the threshold itself, as we can see in the following equation:

$$f(x) = \begin{cases} x & \text{if } 0 \leq x \leq 6 \\ 6 & \text{if } x > 6 \\ 0 & \text{if } x < 0 \end{cases} \tag{5}$$

C. QUANTIZATION TECHNIQUES AND OBSERVERS

Regarding quantization techniques, we chose to use both PTQ and QAT [18]. Observers used to estimate the quantization parameters were sourced from the relative PyTorch module.[2] When using PTQ, to compute quantization parameters for activations, we used a default *HistogramObserver*.[3] This observer derives the quantization parameters by generating a histogram of running minimums and maximums. Regarding quantization parameters for weights, we chose a *PerChannelMinMaxObserver*,[3] which calculates the quantization parameters based on the running minimum and maximum values per channel. This observer utilizes the tensor's statistics to determine the per-channel quantization parameters using a symmetric scheme. It keeps track of the running minimum and maximum values of incoming tensors, utilizing this data to compute the quantization parameters. In the experiments where QAT was applied, during the training phase, all weights and biases are stored in FP32, and backpropagation proceeds as usual. However, during the forward pass, quantization is simulated internally using *FakeQuantize*[4] modules. These modules are termed "fake" because they perform quantization and immediate dequantization of the data, introducing quantization noise similar to what might occur during quantized inference. Consequently, the final loss incorporates any anticipated quantization errors. For computing quantization parameters for activations, we employed a *MovingAverageMinMaxObserver*[3], which calculates the quantization parameters based on the moving averages of the minimums and maximums of the incoming tensors. For computing quantization parameters for weights, we utilized a *MovingAveragePerChannelMinMaxObserver*,[3] which uses tensor min/max statistics to calculate the per-channel quantization parameters.

IV. EXPERIMENTAL SETTINGS

A. DATASETS

As previously mentioned, seven datasets were used for the classification task, ImageNet [35], CIFAR-10 [36], and four datasets containing satellite images Brazilian Coffee Dataset (BCS) [37], EuroSAT [38], PatternNet [39] and RSI-CB256 [40]. Additionally, the DFC 2020 dataset [41] was employed for the segmentation task. Fig. 1 showcases examples of

[1] In this instance, the curve is not the ROC curve but rather a curve that reflects fluctuations in accuracy with changes in the fault rate.

[2] https://pytorch.org/docs/stable/quantization-support.html

[3] https://pytorch.org/docs/stable/quantization-support.html#torch-ao-quantization-observer

[4] https://pytorch.org/docs/stable/quantization-support.html#torch-ao-quantization-fake-quantize



FIGURE 1. A sample of images extracted from satellite datasets.

TABLE 1. Main characteristics of the datasets employed in this study and other typical computer vision benchmarks used in previous fault tolerance research (CIFAR-10, CIFAR-100).

Dataset	# images	# classes	Image size
ImageNet [13]	1,431,167	1000	varying
CIFAR-10 [36]	60,000	10	32×32
CIFAR-100 [36]	60,000	100	32×32
BCS [37]	2,876	2	64×64
EuroSAT [38]	27,000	10	64×64
PatterNet [39]	30,400	38	256×256
RSI-CB256 [40]	24,747	35	256×256

images extracted from satellite datasets, and Tab. 1 outlines the key characteristics of the classification datasets.

To ensure the reproducibility and comparability of results obtained in the classification tasks on satellite image datasets, we followed the dataset split outlined in the AiTLAS benchmark [42]. The training, validation, and test sets were allocated as 60%, 20%, and 20% of the total dataset, respectively. These splits, stratified by class, are accessible in the AiTLAS project repositories. In our experiments with satellite datasets, we trained the DNNs on the training set, determined optimal thresholds on the validation set, and subsequently evaluated performance on the test set. For both ImageNet and DFC2020 datasets, we adhered to the original dataset split.

B. CNN MODELS

Whenever feasible, we evaluated fault tolerance using pre-trained models obtained from official repositories. The training process for all models involved either standard supervised learning or pre-training using SSL, followed by fine-tuning for the specific classification task.

TABLE 2. Main characteristics of the CNNs employed in this study.

Architecture	# parameters	# layers	Flops (GFLOPs)	Main features
ResNet-50	∼ 25.6M	50	∼ 4.1	Residual connections
VGG13-BN	∼ 133M	13	∼ 19.7	Multiple 3×3 filters
DenseNet-161	∼ 28.7M	161	∼ 7.8	Dense connectivity
InceptionV3	∼ 23.9M	48	∼ 5.7	Inception modules

The experiments detailed in Section V encompassed a variety of networks, including VGG13-BN, Densenet161, InceptionV3 and ResNet-50, whose main characteristics are described in Tab. 2. In contrast, all experiments in Sections VI and VIII focused exclusively on the ResNet-50 architecture. Pretrained models for CIFAR-10 and ImageNet were sourced from the PyTorch library,[5] encompassing those trained using SSL techniques such as SwAV[6] [32], SimSiam[7] [33] and DINO[8] [34]. For experiments related to the segmentation task, we employed the SwinUNet model, following the code released by the authors for both the version trained from scratch and the one with contrastive SSL pretraining[9] [43].

For satellite classification tasks, models trained with standard supervision, using pre-trained model architectures on the ImageNet, were accessed through the AiTLAS project[10] [42]. Models finetuned from SSL pre-trained backbones were trained by the authors, as detailed in Section IV-C. The fine-tuning process commenced with weights derived from SSL techniques like SwAV[6] [32], SimSiam[7] [33] and DINO[8] [34], which were originally trained on ImageNet [35].

C. HYPERPAMETER SETTINGS

For each fine-tuning experiment detailed in Section VI, we employed hyperparameters identical to those utilized in finetuning the respective version without SSL pretraining [42]. The images were uniformly resized to 256×256, and following the application of data augmentation, a random crop of 224×224 was selected. The training process involved optimizing all networks until convergence using the RAdam optimizer [44], employing a batch size of 128, and setting the learning rate to 1e-3. To enhance training, a learning rate scheduler was implemented, reducing the learning rate by a factor of 0.1 whenever the validation loss reached a plateau, with the patience parameter set to 5. All figures in the following Sections are generated using the mean accuracy value obtained from 5 repetitions of the fault injection process.

[5]https://pytorch.org/vision/0.8/models.html
[6]https://github.com/facebookresearch/swav/tree/main
[7]https://github.com/facebookresearch/simsiam
[8]https://github.com/facebookresearch/dino
[9]https://github.com/HSG-AIML/SSLTransformerRS
[10]https://github.com/biasvariancelabs/aitlas-arena/tree/main

D. FAULT INJECTION HYPER-PARAMETERS

Each trained model underwent evaluation under two conditions: without fault injection (baseline) and at specific fault rates in the following set: $[1 \times 10^{-8}, 3 \times 10^{-8}, 1 \times 10^{-7}, 3 \times 10^{-7}, 1 \times 10^{-6}, 3 \times 10^{-6}, 1 \times 10^{-5}, 3 \times 10^{-5}]$. Fault rates extremely low, falling within the range $[0, 1 \times 10^{-8}]$, were excluded as their impact on performance negligible. Conversely, fault rates exceeding 3×10^{-5} were disregarded to avoid introducing an excessive number of faults that would result in similar DNN performances across all configurations, essentially equivalent to that of a random classifier. Only in experiments with quantized networks on ImageNet, higher fault rates were evaluated, including the following: $[1 \times 10^{-4}, 3 \times 10^{-4}, 1 \times 10^{-3}]$.

In assessing network performance, metrics such as top-1 and top-5 accuracy were employed for ImageNet, while top-1 accuracy was used for both Cifar-10 and satellite datasets. For the segmentation task, performance was evaluated using pixel-level accuracy, which measures the proportion of correctly classified pixels over the total number of pixels in the image.

E. FAULT MITIGATION HYPER-PARAMETERS

Both ClipAct and ActMax depend on a validation set to compute the activation distribution and, therefore, the optimal layer-wise thresholds. For CIFAR-10, BCS, EuroSAT, PatternNet, and RSI-CB256, the entire validation set was used to calculate the thresholds. For ImageNet and DFC2020, the validation split was used in lieu of the test set to report performance, therefore we sampled images from the training set. For DFC2020, the entire training set was used. For ImageNet, due to the large dataset size (over 1.4 million images, with a validation set of 50,000 images), computing thresholds over the entire training set would have been prohibitively expensive. Instead, a random sample of 4,800 images (approximately 0.3% of the training set) was selected from the training set at each iteration of the ClipAct algorithm, ensuring class stratification for balanced representation in the 1,000 categories. By selecting a representative subset, we reduced the overall processing time to approximately two days, striking a balance between computational feasibility and statistical reliability of the estimated thresholds.

In terms of fault rates, a slightly narrower range $[1 \times 10^{-7}, 1 \times 10^{-6}, 3 \times 10^{-6}, 1 \times 10^{-5}, 3 \times 10^{-5}]$ was used; a total of 10 iterations per layer were performed to optimize the threshold values.

F. QUANTIZATION HYPERPARAMETER

For all experiments involving quantized models, we selected 8-bit quantization both for activations and weights. This choice was made to evaluate the impact of quantization, including in combination with SSL pretraining, rather than aiming for the most efficient memory footprint. Regarding experiments on ImageNet, we chose PTQ, utilizing a subset of 4,800 images extrapolated from training set for observer calibration. However, for experiments on EO tasks, we opted for QAT using the entire training set.

V. FAULT TOLERANCE OF DIFFERENT CNN ARCHITECTURES

The accuracy of four popular DNN architectures (ResNet-50, VGG13-BN, Densenet161 and InceptionV3) across different configurations and fault rates is reported in Figs. 2 and 3. Specifically, Fig. 3 illustrates the top-5 accuracy across various error rates on ImageNet. Although the top-1 accuracy is not displayed for brevity, its trend aligns with the top-5 accuracy. As expected given the greater complexity of the task, experiments carried out on ImageNet (Fig. 3) reveal lower fault tolerance compared to those on Cifar-10 (Fig. 2). More interestingly, the experimental results suggest substantial differences between the investigated architectures.

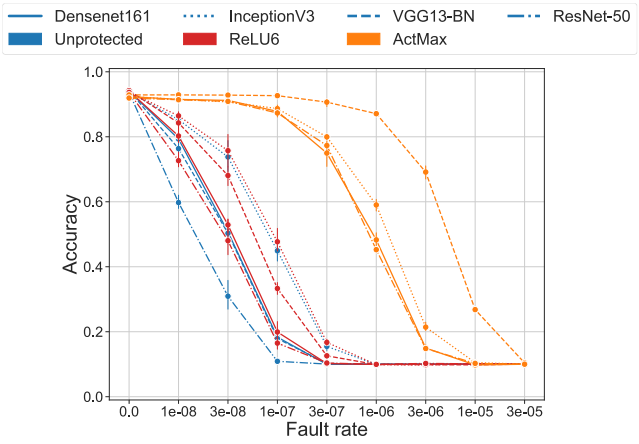


FIGURE 2. Average ResNet-50, VGG13-BN, Densenet161 and InceptionV3 accuracy on the Cifar-10 dataset, using ReLU, ReLU6 and ActMax as activation functions. Figure better viewed online.

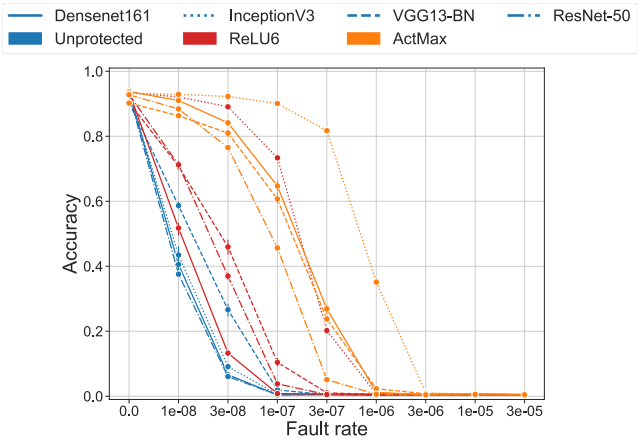


FIGURE 3. Average ResNet-50, VGG13-BN, Densenet161 and InceptionV3 top-5 accuracy on the ImageNet dataset, using ReLU, ReLU6 and ActMax as activation functions. Figure better viewed online.

In general, *DenseNet-161 proved to be particularly vulnerable to faults.* Although dense network connectivity is advantageous for propagating information under ideal conditions, this same characteristic amplifies the impact of faults. Faults in specific layers can rapidly propagate through dense connections, compromising the quality of representations in subsequent layers. This behavior is evident in both datasets, where DenseNet-161 shows a rapid drop in accuracy as the fault rate increases, making it one of the least resilient architectures overall.

In contrast, *InceptionV3 stands out for its greater resilience*, especially on ImageNet. The inception modules, which process information at different scales, enable the network to maintain high performance even in the presence of faults. In the ImageNet dataset, which contains higher-resolution images, the Inception modules support robust representations that mitigate the effects of perturbations in weights and biases. However, in CIFAR-10, the low image resolution appears to limit the potential of the inception modules, resulting in a weaker overall performance.

VGG13-BN has intermediate performance in terms of fault tolerance, but the results are highly dependent on the experimental configuration. Although the lack of skip connections and the limited number of layers reduce the propagation of the fault, the higher number of parameters increases the number of faults according to our fault model. Finally, *ResNet-50 consistently placed as one of the least robust architectures in our experiments*, on both CIFAR-10 and ImageNet. Although residual connections are designed to facilitate the propagation of information, these connections also promote the propagation of errors. This mechanism can amplify the impact of faults, particularly when they occur in critical layers of the network. This result is in line with previous studies that compare ResNet-50 and VGG-13 in the Cifar-10 dataset [9], [10].

The ranking among different architectures is dependent on the specific fault mitigation strategy used. ReLU6 is, in this respect, inherently limited by the use of a fixed threshold, while the distribution of activations varies depending on the dataset, CNN architecture, and the specific layer under scrutiny, as shown in Fig. 5 for the ResNet-50 architecture. In particular, activations are generally higher in the ImageNet dataset than in Cifar-10, surpassing the threshold value of 6. Consequently, even features that are error-free may be affected. ActMax, which uses an adaptive threshold, is more robust to changes in distribution and achieves higher performance across all datasets and architectures. However, it also reveals that certain architectures appear to be more amenable to optimization than others. For instance, on Cifar-10 VGG13-BN achieves surprisingly good performance, but this pattern is not observed on ImageNet.

Furthermore, to assess whether the design of a DNN or the number of parameters is more influential on fault tolerance, we chose to evaluate the performance of DNNs with identical design but varying depths. This investigation aims to determine whether increasing the depth of neural

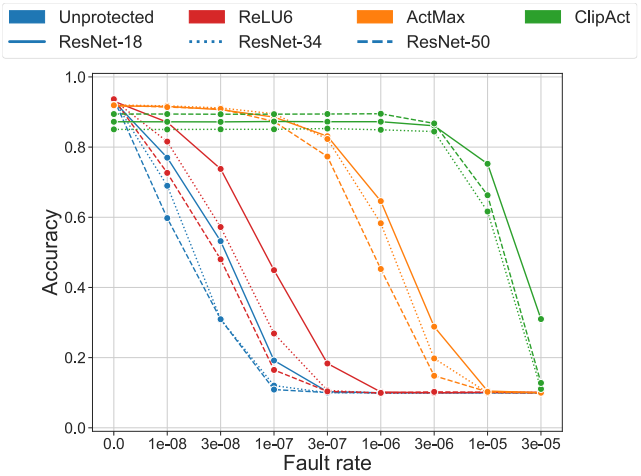


FIGURE 4. Average ResNet-18, Resnet-34, and Resnet-50 accuracy on the Cifar-10 dataset, using ReLU, ReLU6, ActMax and CReLU as activation functions. Figure better viewed online.

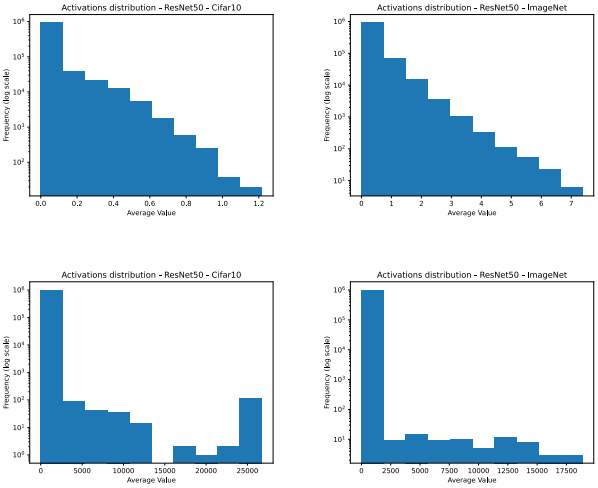


FIGURE 5. Distributions of the activation outputs of a ResNet-50 layer, without and with fault (fault rate equal to 3×10^{-5}) in the top and bottom row, respectively.

networks enhances not only overall performance but also fault tolerance. The results of three different ResNets (ResNet-18, ResNet-34, and ResNet-50) are depicted in Fig. 4. These experiments were carried out on Cifar-10, using a more sophisticated fault mitigation technique (ClipAct). In general, it is observed that the use of a smaller model enhances fault tolerance. However, this advantage diminishes as the fault mitigation technique improves, reaching a point where for example, the use of ClipAct delivers better performance for ResNet-50 in the majority of the failure rates analyzed. Consequently, in subsequent experiments, we opted for this architecture, as it is the most widely used among various ResNets at the state of the art in self-supervised learning. These results underscore the importance of jointly optimizing architectural characteristics and fault mitigation strategies, in fault injection analysis.

VI. IMPACT OF SELF-SUPERVISED PRETRAINING ON FAULT TOLERANCE

We tested the efficacy of SSL pretraining, alone or in combination with fault mitigation techniques, by comparing the accuracy with those of unprotected DNNs. The accuracy distribution of ResNet-50 across different configurations and fault rates is reported in Figs. 6 and 7 for ImageNet and satellite image datasets, respectively.

When using standard supervised training (continuous lines in Figs. 6 and 7), the accuracy of the unprotected DNN drops exponentially even at very low fault rates. On the other hand, when activations are clipped, the performance decay seems to follow a smoother logarithmic profile. Generally speaking, *ClipAct performs better than ActMax for all datasets at fault rates greater than* 3×10^{-7}. ClipAct is especially effective on the Eurosat, Patternet, and RSI-CB256 benchmarks, on which performance is close to the baseline value up to a fault rate of 10^{-6}. This behavior is less pronounced, or even absent, in simpler datasets, like CIFAR-10 and satellite imagery datasets. In contrast, ImageNet features a large number of classes and significant intra-class variability. This complexity intensifies the trade-off between fault tolerance and accuracy, making a uniform approach for each layer to activation clipping less effective. As shown in Fig. 5, the activation distributions differ markedly between more complex datasets like ImageNet and simpler ones like CIFAR-10, highlighting the need for context-specific adjustments.

In all cases, performance degrades to random guessing when the fault rate reaches 3×10^{-5}. It is worth recalling that the number of classes varies widely between datasets, and the chance-level accuracies at which the unprotected network plateaus vary accordingly, as shown in Fig. 7. The results obtained on these datasets are qualitatively similar to those reported in previous experiments on CIFAR-10 and CIFAR-100 [9], [10].

On Imagenet, however, ClipAct significantly degrades performance at baseline (without injecting faults) and at low fault rates; to a lesser extent, this behavior is observed also on BCS. Threshold fine-tuning aims to optimize performance across all fault rates even at the cost of degrading performance at lower fault rates, for which less aggressive thresholds would be preferable, or in the absence of faults. To the best of our knowledge, we are the first to report this behavior, which we attribute to the higher complexity of the datasets compared to previous literatures [9] and [10].

When using SSL pretraining (dashed lines in Figs. 6 and 7), the drop in accuracy for the unprotected DNN is similar across all configurations. However, *when activation clipping is used in conjunction with SSL pretraining*, accuracy improves across all fault rates, and *SSL achieves higher fault tolerance than standard supervised training*. On ImageNet, at a fault rate of 3×10^{-7}, the top-1 accuracy for ActMax is 1.9% without SSL pretraining, 7.7% with SwAV and 35.5% with DINO. Similarly, at a fault rate of 3×10^{-6}, the top-1 accuracy for ClipAct is 7.4%, 22.4%, and 26.5% for supervised, SwAV, and DINO pretraining, respectively. A similar pattern is observed for ImageNet top-5 accuracy.

On the satellite datasets (Fig. 7) *SSL pretraining does not yield significant advantages in the unprotected scenario or when using ClipAct*. On the other hand, with ActMax, SSL pretraining consistently outperforms supervised pretraining, but the performance gain is marginal compared to that offered by ClipAct. For instance, at a fault rate of 3×10^{-7} in Eurosat, the accuracy observed was 59.18%, 89.86%, and 93.11% without SSL, with SwAV, and with DINO, respectively. In PatternNet, under the same conditions, the top-1 accuracies were 46.29% (No SSL), 83.73% (SwAV), and 95.18% (DINO). In contrast, when using ClipAct, at a fault rate of 1×10^{-6} the accuracy on Eurosat and PatternNet was approximately 92% and 98%, respectively, regardless of the pretraining methodology used. It is also worth noticing that these benchmarks are much simpler classification tasks than ImageNet, with baseline accuracy around 99%. It is possible that a simpler decision boundary facilitates the process of fine-tuning the thresholds.

Among the satellite datasets, *the BCS dataset*, which is also the simplest task, *gains the most from SSL pretraining*; at a fault rate of 1×10^{-6}, ClipAct achieves an accuracy of 53.12%, 87.19% and 87.33% without SSL, with SwAV and DINO, respectively. In contrast, experiments using ActMax together with SwAV and DINO at the same error rate yield 63.68% and 76.67% accuracy, respectively. RSI-CB256, on the other hand, is the only dataset where SwAV pretraining performs worse than supervised pretraining.

Numerical results are reported in Tabs. 3 and 4, which further include two additional SSL pretraining strategies (SimSiam and SSQL). All pretraining techniques follow the aforementioned patterns.

To *shed light on the different performance observed with and without SSL fine-tuning*, we analyzed the behavior of ClipAct in a subset of the ImageNet validation set. Specifically, we computed the percentage of activations that exceeded the threshold T_l, and thus were clipped according to Eq. 4. The result was computed for each residual block of the network and for different fault rates, taking into account that ClipAct computes a different threshold for each layer. As depicted in Fig. 8, with standard supervised training, the percentage of activations higher than the threshold is generally much higher, up to and exceeding 10% in some blocks, with the exception of a few blocks, in which the percentages are nearly identical. This behavior suggests that the features learned in these blocks are robust enough to not be influenced by injected faults regardless of whether SSL pretraining is used or not. When SSL pretraining is used instead, activations that exceed the thresholds are generally less than 0.01%. This divergence may stem from the *different weight distributions associated with each pretraining technique* [16]. Representations learned through SSL tend to distribute information differently across the network weights, which significantly mitigates the impact of transient hardware faults, such as bitflips, on network performance. As shown in

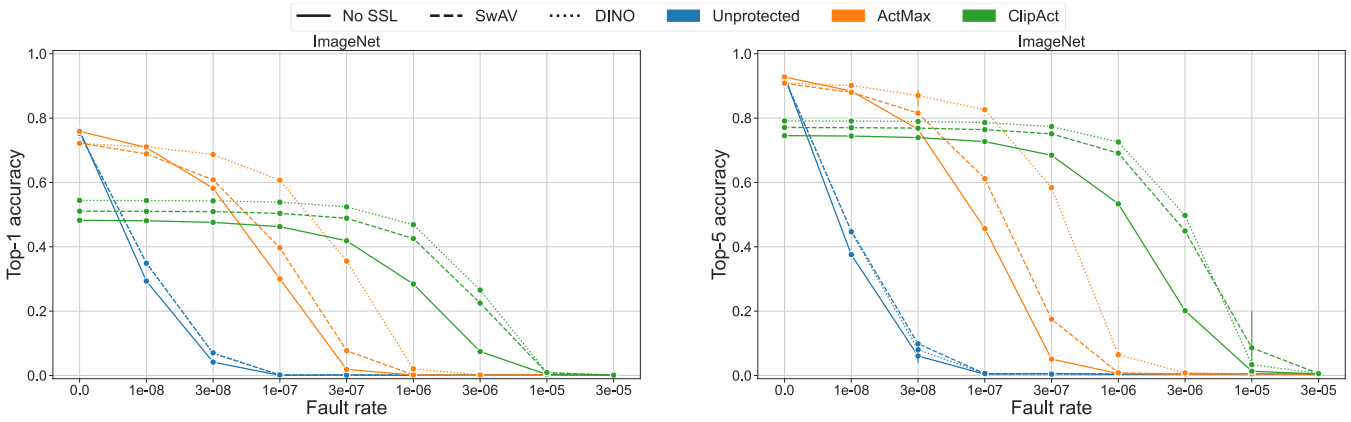


FIGURE 6. Mean ResNet-50 top-1 and top-5 accuracy for unprotected model, ActMax and ClipAct [9], without SSL and with SwAV [32] and DINO [34] pretraining, on ImageNet dataset [35], under different fault rates. Figure better viewed online.

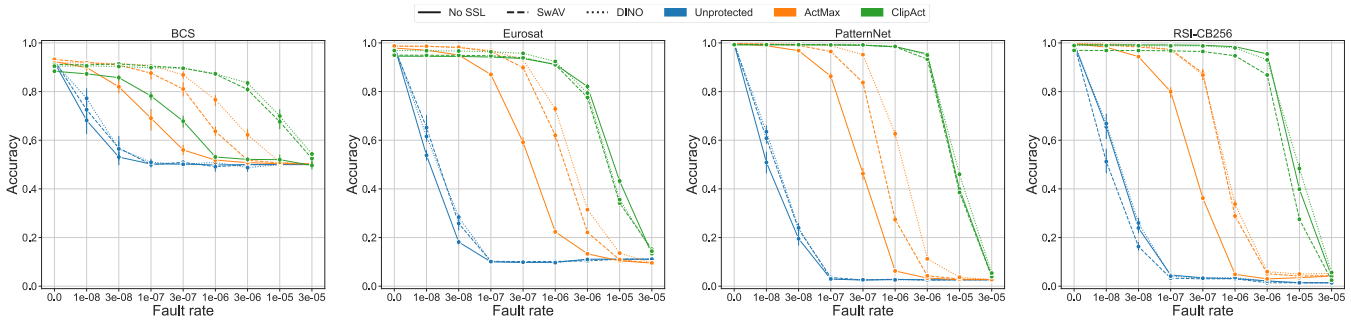


FIGURE 7. Average ResNet-50 accuracy for unprotected model, ActMax and ClipAct [9], without SSL and with SwAV [32] and DINO [34] pretraining, on BCS [37], Eurosat [38], PatternNet [39] and RSI-CB256 [40] dataset, under different fault rates. Figure better viewed online.

TABLE 3. Average ResNet-50 top-1 (top-5) accuracy for quantized and floating point model with ClipAct, without SSL and with SwAV [32], DINO [34], SimSiam [33] and SSQL [16] pretraining, on ImageNet dataset [35], under different fault rates. Best and second-best quantized configurations are indicated in bold and italics characters, respectively.

Pretraining	Top-1 (top-5) accuracy unmitigated model without ClipAct	Quantized	Top-1 (top-5) accuracy with a fault rate equal to 0	3×10^{-7}	3×10^{-6}	3×10^{-5}
No SSL	**76.15 (92.87)**	x	48.23 (74.54)	41.86 (68.45)	7.41 (20.13)	0.09 (0.45)
		✓	**75.42 (92.79)**	**75.45 (92.76)**	**75.47 (92.77)**	**75.20 (92.60)**
SwAV	*75.30 (92.42)*	x	51.07 (77.08)	48.88 (75.10)	22.49 (44.90)	0.15 (0.59)
		✓	65.11 (91.08)	65.08 (91.13)	65.08 (91.07)	64.94 (90.48)
DINO	75.28 (92.57)	x	54.42 (79.11)	52.41 (77.37)	26.54 (49.75)	0.10 (0.60)
		✓	*68.14 (91.60)*	*67.98 (91.55)*	*67.92 (91.54)*	*67.37 (90.80)*
SimSiam	68.29 (88.26)	x	53.08 (78.19)	51.15 (76.85)	31.93 (57.64)	0.09 (0.48)
		✓	65.12 (87.78)	65.13 (87.84)	64.99 (87.80)	64.64 (87.56)
SSQL	67.74 (88.08)	x	62.96 (85.26)	61.47 (84.16)	43.94 (69.84)	0.13 (0.64)
		✓	67.09 (87.67)	67.06 (87.69)	67.05 (87.67)	66.95 (87.59)

the analysis of activation distributions in Fig. 9, the compact, sparse and bounded nature of activations in SSL-pretrained models contributes to their fault resilience. For several layers, the distribution of the activations in SSL-pretrained models is relatively sparse, with a very narrow interquantile range and high mean: activations are zero in the majority of samples, whereas a few samples generate very strong activations. This property, which is more pronounced for SimSiam than DINO or SwAV, reduces the likelihood that errors in weights will cause drastic changes in output. This is because SimSiam-pretrained networks constrain the effect of weight perturbations, limiting how faults propagate through the network. When an error occurs, activations are less likely to amplify its impact, preventing uncontrolled changes in the final output and enhancing the model's overall stability. SSL-pretrained models also benefit more from activation clipping, creating a natural synergy with fault mitigation strategies, and ensuring that fewer activations exceed the thresholds, as shown in Fig. 8. The effect of SSL pretraining is more evident on the intermediate layers of residual block, whereas the initial and output convolutional layers appear to activate more often. Compared to standard supervised training,



FIGURE 8. Percentage of ResNet-50 activations larger than the ClipAct's threshold for different residual blocks, assessed on the ImageNet dataset at fault rates of 0, 1×10^{-7}, 1×10^{-6} and 3×10^{-5}. A logarithmic scale is used for ease of comparison.

SSL-pretrained models appear to concentrate information in fewer layers, and yield broader activation distributions in deeper layers.

A similar but less pronounced trend were observed in the satellite datasets, as exemplified by the analysis conducted on EuroSAT, shown in Fig. 10. It is possible that the beneficial properties of SSL pretraining are partially lost in the fine-tuning process, which repurposes some of the units to learn novel features. We leave to future experiments to evaluate the effect of pretraining on the same data distribution as the target data. These analyses underscore that SSL pretraining enhances fault tolerance by producing more robust features and making it *easier to identify thresholds that discriminate normal from erroneous fault-induced activations*.

VII. IMPACT OF ACTIVATION CLIPPING ON BASELINE ACCURACY

Adaptive activation clipping algorithms such as ClipAct are commonly employed to improve fault tolerance in deep neural networks, but may have a detrimental effect on baseline (fault-free) accuracy. This behavior is especially evident on ImageNet (Fig. 6) but is also observed to a lesser extent on other datasets (Tab. 4). Evaluating the impact of ClipAct in combination with different pretraining techniques requires comparing accuracies across datasets characterized by inherently different complexity and baseline accuracy. Inspired by [45], we consider the log odds, i.e., the accuracy after applying the logit transformation:

$$\text{log odds} = \log\left(\frac{p}{1-p}\right) \tag{6}$$

where p is the Top-1 accuracy of the model. The logit transformation is a commonly used transformation for the analysis of proportion data, as an additive change in log space is equivalent to a multiplicative change in the original space; we consider here the effect of dataset and pretraining strategy to be additive [45].

To compute error bars for the average pretraining accuracy across datasets, we removed the variance due to inherent

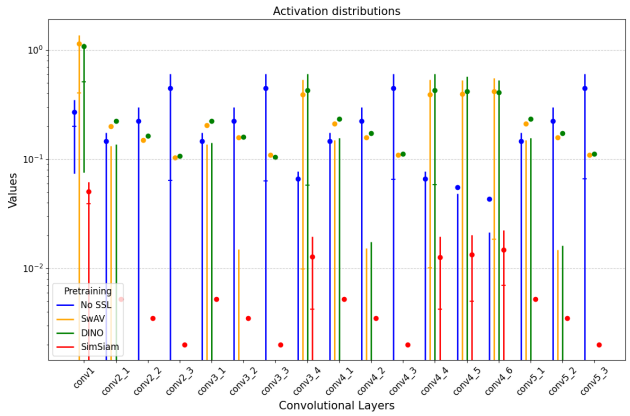


FIGURE 9. ResNet-50 activation distributions across convolutional layers without SSL and with SwAV [32], DINO [34] and SimSiam [33] pretraining, on ImageNet dataset [35]. The markers represent the mean (dots) and median (lines), while the vertical bars indicate the interquartile range.

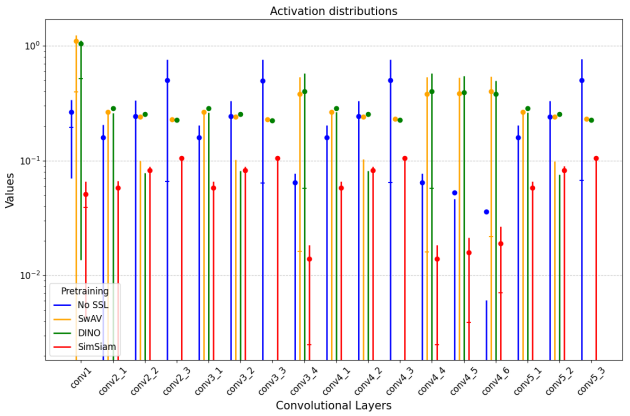


FIGURE 10. ResNet-50 activation distributions across convolutional layers without SSL and with SwAV [32], DINO [34] and SimSiam [33] pretraining, on Eurosat dataset [38]. The markers represent the mean (dots) and median (lines), while the vertical bars indicate the interquartile range.

differences in dataset difficulty. Specifically, for each pretraining method m and dataset d, we first obtained the logit-transformed accuracy values x_{md}. Then, we adjusted each accuracy value by subtracting the mean accuracy of that dataset across all pretraining methods:

$$\text{acc}(m, d) = x_{md} - \frac{1}{|M|} \sum_{n \in M} x_{nd} \tag{7}$$

where M represents the set of pretraining methods.

For each pretraining method, we computed the mean and standard error of these adjusted accuracy values across datasets. To ensure a correct estimation of variability, the standard error was further scaled by a correction factor:

$$\sqrt{\frac{|M|}{|M| - 1}} \tag{8}$$

This adjustment accounts for dataset-specific differences, allowing the error bars to more accurately reflect the variability in model performance across datasets rathen that the inherent dataset difficulty [45].

In Fig. 12, the blue line represents the linear regression of log odds for models trained without ClipAct. This regression captures the general trend between accuracy and pretraining strategies, serving as a reference for unaltered performance. The individual points, instead, correspond to the log odds, computed as the mean across five datasets, obtained without (blue points) and with ClipAct (red points). Each point is plotted at the x-coordinate corresponding to the Top-1 accuracy on ImageNet of the pretrained model (which depends on the SSL pretraining strategy), enabling a direct comparison of accuracy with and without ClipAct.

The impact of ClipAct is not uniform across different SSL pretraining methods. Models trained with standard supervised learning experience the most significant drop in accuracy, with a relative average reduction exceeding 27.95%, suggesting that their feature representations are particularly sensitive to the imposed clipping thresholds. In contrast, SSL-pretrained networks retain their accuracy more effectively, showing a more controlled degradation. Among them, SimSiam and SSQL show the least deviation, with average accuracy losses of only 7.60% and 10.04%, respectively, indicating that their learned feature representations naturally align better with activation clipping constraints. DINO and SwAV also outperform supervised training, with intermediate reductions of approximately 20.94% and 28.31%, respectively, showing that while their features remain relatively stable, they are still affected by the imposed thresholding.

These observations reinforce the idea that SSL-pretrained models learn more structured and resilient feature representations, which not only provide benefits in downstream tasks but also make them more adaptable to architectural constraints like activation clipping. The results suggest that SSL training naturally enforces activation distributions that are already closer to the optimal thresholds imposed by ClipAct, reducing the risk of performance degradation.

VIII. FAULT TOLERANCE OF QUANTIZED NETWORKS

Given the paramount importance of fault tolerance in resource-constrained scenarios like the aerospace and automotive sectors, and acknowledging the substantial impact of quantization on fault resilience acting as a pseudo-activation-clipping mechanism similar to ClipAct and ActMax [46], we investigated fault tolerance concerning quantized networks. This analysis was carried out for all the experiments in the previous section so as to evaluate, beyond the impact of quantization on datasets relating to EO tasks, the impact of a self -supervised pretraining also in this context.

In the case of experiments on ImageNet, we adopted PTQ due to the size of the neural network and the vast dataset. Fig. 11 illustrates the trend of top-1 and top-5 accuracy as fault rates vary. Instead, Tab. 3 presents a comparison of the accuracy between the quantized model and the FP32 model

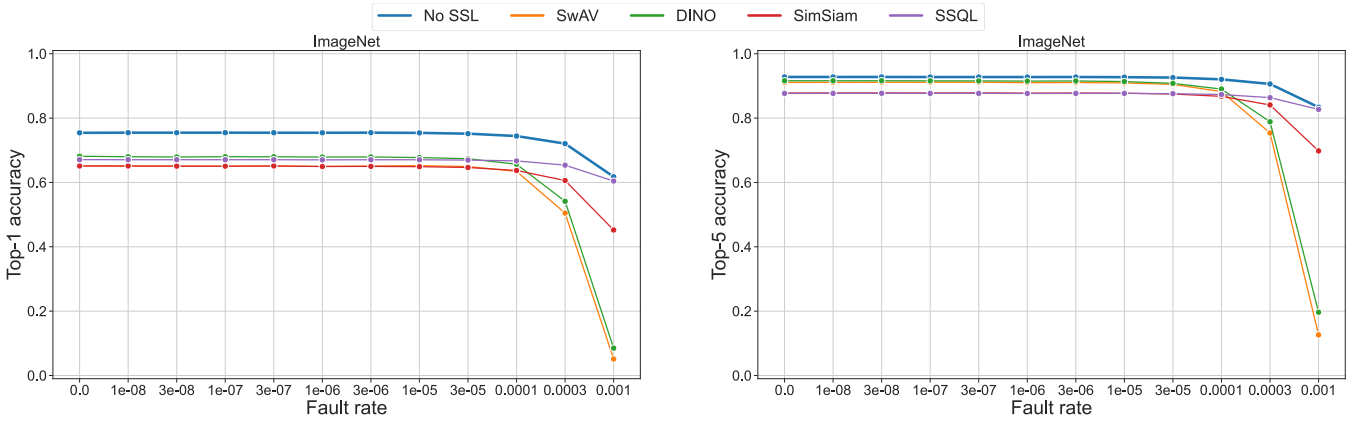


FIGURE 11. Mean ResNet-50 top-1 and top-5 accuracy for quantized model, without SSL and with SwAV [32], DINO [34], SimSiam [33] and SSQL [16] pretraining, on ImageNet dataset [35], under different fault rates. Figure better viewed online.

TABLE 4. Average ResNet-50 accuracy for quantized models and floating point model with ClipAct, without SSL and with SwAV [32], DINO [34], SimSiam [33] and SSQL [16] pretraining, on BCS [37], Eurosat [38], PatternNet [39] and RSI-CB256 [40] dataset, under different fault rates. Best and second-best quantized configurations are indicated in bold and italics characters, respectively for each dataset.

Dataset	Pretraining	Top-1 accuracy unmitigated model without ClipAct	Quantized	Top-1 accuracy with a fault rate equal to 0	3×10^{-7}	3×10^{-6}	3×10^{-5}
BCS	No SSL	92.01	x	88.37	67.85	52.12	49.65
			✓	83.16	82.88	83.09	81.84
	SwAV	*92.71*	x	*91.15*	89.69	80.83	52.50
			✓	49.48	49.31	48.99	48.02
	DINO	92.01	x	90.45	89.58	83.54	54.34
			✓	32.29	32.95	33.26	36.39
	SimSiam	**92.88**	x	**92.01**	**91.77**	87.60	52.43
			✓	90.80	*90.97*	**91.04**	**90.66**
	SSQL	90.97	x	90.80	90.17	85.94	53.44
			✓	90.62	90.14	*90.07*	*88.68*
Eurosat	No SSL	**98.83**	x	94.57	93.58	82.09	13.40
			✓	25.52	25.58	25.61	25.43
	SwAV	98.07	x	94.98	93.91	77.62	15.20
			✓	11.54	11.55	11.55	11.42
	DINO	98.70	x	96.87	95.72	79.42	14.39
			✓	11.94	11.97	11.95	11.7
	SimSiam	98.68	x	**98.24**	**97.87**	**93.41**	19.45
			✓	41.72	41.78	41.50	*41.60*
	SSQL	*98.72*	x	*98.00*	*97.41*	*90.16*	16.01
			✓	54.39	54.33	54.30	**54.45**
PatternNet	No SSL	**99.74**	x	99.21	99.04	95.48	4.09
			✓	86.22	86.10	85.99	*85.41*
	SwAV	*99.72*	x	99.33	99.08	93.34	3.94
			✓	38.98	39.23	38.81	38.02
	DINO	99.64	x	99.29	99.20	95.01	5.39
			✓	54.11	54.30	54.11	52.86
	SimSiam	99.59	x	*99.47*	*99.38*	96.50	6.79
			✓	63.68	63.70	63.59	62.06
	SSQL	99.64	x	**99.54**	**99.48**	*96.98*	4.75
			✓	99.24	99.23	**99.25**	**99.22**
RSI-CB256	No SSL	99.55	x	98.69	98.55	94.79	4.26
			✓	85.05	85.09	85.18	84.04
	SwAV	99.68	x	96.93	96.47	86.82	2.47
			✓	50.75	50.69	50.59	49.52
	DINO	*99.72*	x	98.93	98.77	93.03	5.63
			✓	59.76	60.06	59.66	59.15
	SimSiam	99.68	x	**99.64**	**99.60**	*97.70*	5.86
			✓	90.42	90.46	90.47	*90.00*
	SSQL	**99.78**	x	*99.41*	*99.34*	95.92	4.65
			✓	98.14	98.15	**98.16**	**98.11**

with ClipAct across a subset of fault rates. *Quantization notably resulted in significant performance degradation*, especially in top-1 accuracy when employing self-supervised pretraining: top-1 accuracy compared to the FP32 model is

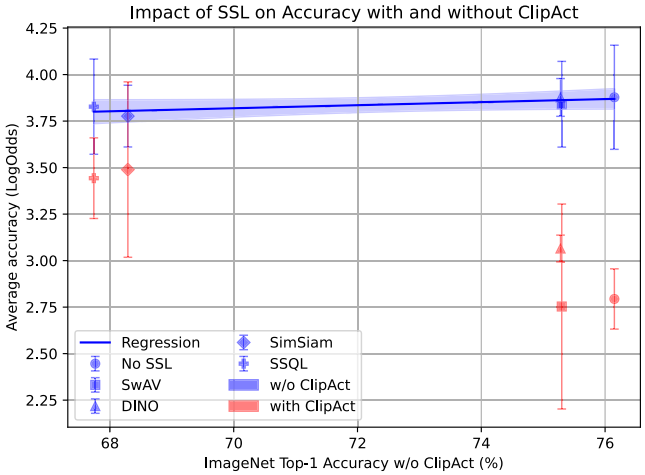


FIGURE 12. Relationship between ResNet-50 Top-1 accuracy on ImageNet dataset [35] and log odds, illustrating the impact of ClipAct on baseline accuracy. The blue line represents the linear regression of log odds for models trained without ClipAct, showing the general trend across different pretraining strategies. The individual points correspond to the log odds without (blue points) and with (red points) ClipAct, computed as the mean across five datasets, plotted at the x-coordinate of their respective accuracy on ImageNet in the unprotected scenario.

degraded of 10.19 and 7.14 points, respectively for SwAV and DINO. This can be attributed to a large, non-uniform weight distribution, which complicates the quantization process and leads to inaccuracies in mapping FP32 values to lower bit-widths [16]. *This problem is mitigated when employing SSQL.* Integrating a loss component related to quantization error during the pretraining phase of SimSiam ensures reduced degradation from quantization. Specifically, the floating-point model trained using SimSiam achieves a top-1 accuracy of 68.3% with a quantization-induced deterioration of 3.18 points. In contrast, with SSQL, the degradation is reduced to only 1.21 points.

Regarding fault tolerance, the quantized network displayed improved performance, with only an accuracy drop at the highest fault rate. Among the SSL techniques, the most effective ones are SimSiam and its optimized version for quantization, which, as depicted in Fig. 11, not only enhance the performance of the quantized network but also improve its fault tolerance. At the highest fault rate, the performance degradation for SSQL was 6.66 points, whereas without any SSL technique, it was 13.66 points. Despite showing poorer performance in fault-free scenarios, which translates to weaker results even when failures occur, *SSQL demonstrates superior fault tolerance*, as evidenced by reduced performance decay, and delivers comparable outcomes at high failure rates. The results obtained with SSQL show slight inconsistencies compared to those presented in the original paper [16]. Yet, it outperforms previous experiments using ClipAct across all failure rates.

For experiments involving datasets with satellite images, characterized by a smaller size compared to ImageNet, we opted for QAT. The general trend aligns with previous experiments, revealing more pronounced deterioration when self-supervised pretraining is employed, with a modest performance decline observed only at the highest fault rate. However, as depicted in Tab. 4, the impact of quantization varies across datasets. In BCS and Eurosat, the datasets with the lowest number of classes (2 and 10, respectively), a notable performance drop occurs upon quantization, resulting in almost random predictions. This effect is particularly evident in Eurosat, where in case without SSL pretraining the accuracy in a fault-free context deteriorates to a value of 25.52%. In cases with SSL pretraining, the results obtained are 11.54% and 11.94% when SwAV and DINO are applied, respectively. Even in scenarios involving satellite datasets, the utilization of the SSQL technique enhances the performance of quantized networks by mitigating the performance degradation caused by the quantization process. As shown in Tab. 4, unlike the experiments conducted on ImageNet, configurations employing *SSQL generally exhibit superior performance* compared to other setups with and without SSL techniques. This analysis underscores not only the high fault tolerance of quantized networks, maintaining accuracy similar to the fault-free scenario with both supervised and self-supervised pretraining, but also highlights that the quantization process is influenced by various factors, including pretraining and dataset characteristics.

IX. FAULT TOLERANCE IN SEGMENTATION TASK

Segmentation models are particularly susceptible to faults, as errors in feature extraction can propagate across spatial dimensions, leading to significant misclassification of pixel-wise predictions. This is due to the hierarchical nature of feature extraction and the dependency on spatial coherence in segmentation networks. Errors occurring in early layers can be amplified as features are upsampled and combined at different scales, affecting wide regions of the segmentation map. In this section, we illustrate experiments on the DFC2020 dataset, using the SwinUNet model, to analyze the impact of fault rates on segmentation accuracy and the role of SSL pretraining as a potential mitigation strategy across diverse tasks. As shown in Tab 5, the baseline accuracy of SwinUNet drops as the fault rate increases. In a fault-free scenario, SSL pretraining yields a modest benefit in terms of performance (44.87% vs. 41.90%). As fault rates increase, accuracy declines progressively, with SSL-pretrained networks maintaining a slight advantage over their non-SSL counterparts across all fault levels. Experiments with activation clipping (Actmax) show a beneficial effect on performance, achieving 39.87% accuracy at a fault rate of 1×10^{-8} and 15.50% at 1×10^{-6}, outperforming the unprotected baseline in all scenarios. When combined with SSL pretraining, ActMax further enhances resilience, achieving 42.96% accuracy at 1×10^{-8} and 19.73% at 1×10^{-6}. This trend is consistently observed across all tested fault rates, demonstrating that SSL pretraining not only improves accuracy in fault-free conditions but also

TABLE 5. Average SwinUNet accuracy for the segmentation task on the DFC 2020 dataset [41], without fault mitigation techniques and with ActMax, with and without SSL pretraining, under different fault rates.

Pretraining	ActMax	Accuracy with a fault rate equal to						
		0	1×10^{-8}	3×10^{-8}	1×10^{-7}	3×10^{-7}	1×10^{-6}	3×10^{-6}
No SSL	x	41.90	38.17	31.78	20.73	14.55	12.59	12.50
SSL	x	44.87	41.17	34.41	22.28	14.58	12.54	12.50
No SSL	✓	41.92	39.87	36.04	27.73	20.98	15.50	12.66
SSL	✓	44.88	42.96	39.05	30.30	21.66	15.26	12.61

mitigates performance degradation in faulty environments. When combined with SSL pretraining, ActMax further enhances resilience, maintaining higher accuracy across all fault rates, with a peak improvement observed at moderate fault levels.

These findings align closely with those obtained in classification tasks, suggesting that the complementary benefits offered by SSL pretraining and activation clipping extend beyond traditional classification models. Our observations are consistent with a recent study that observed similar performance benefits associated to the use of ActMax on a different dataset and fault model [27]. In addition, spatially dense predictions such as semantic segmentation tend to benefit more from SSL pretraining [47].

X. CONCLUSION

This paper provides an overview of how implementation choices in DNNs impact fault tolerance. Different architectures show varying fault tolerance, but common trends emerge. Vanilla networks struggle at low fault rates, requiring fault mitigation techniques. Among the simplest and most effective are those based on activation clipping, which restricts activation propagation. The threshold for clipping is crucial: static values can improve fault tolerance but may reduce performance in fault-free scenarios. Thus, finding the right thresholds is essential for balancing fault tolerance and performance.

While supervised pretraining on labeled datasets like ImageNet is standard for DNN training, SSL pretraining is gaining traction for its scalability and improved performance and robustness in downstream tasks [14]. Experimental results from seven classification benchmarks show that SSL pretraining, combined with post-training activation clipping, matches or exceeds supervised pretraining in baseline accuracy and fault tolerance. DNNs in critical environments benefit from SSL pretraining, with DINO generally outperforming SwAV. In optimal configurations (DINO pretraining with ClipAct), DNNs maintain near-baseline accuracy up to a fault rate of 1×10^{-6}. Additionally, our analysis confirms that the benefits of SSL extend beyond classification tasks, improving fault tolerance in segmentation tasks as well. This reinforces the broader applicability of SSL as a general strategy for enhancing the resilience of deep learning models across different applications. While ClipAct effectively mitigates faults, it can also lead to a drop in baseline accuracy, especially in networks trained with standard supervised learning. In contrast, SSL-pretrained networks exhibit greater adaptability to activation thresholding. This suggests that SSL techniques not only improve robustness against faults but also make models more naturally compatible with activation-based fault mitigation strategies.

Given the importance of fault tolerance in resource-constrained sectors like aerospace and automotive, we evaluated quantization's impact on fault resilience. Quantized networks, acting as pseudo-activation clipping mechanisms, show improved fault tolerance, maintaining near fault-free performance up to a fault rate of 0.0001. However, SSL pretraining combined with quantization leads to performance drops due to weight distribution issues affecting quantization parameter calibration. Addressing this requires integrating a loss accounting for quantization error during pretraining, a technique known as SSQL [16]. This technique excels in downstream tasks with satellite image datasets, surpassing floating point networks using ClipAct for fault tolerance.

In future work, it would be interesting to explore other techniques, such as fault-aware training [8] or trainable activation clipping [10], in combination with SSL pretraining. By combining SSL with fault mitigation methods such as FAT, it would be possible to leverage the robustness of the representations learned during pretraining alongside targeted fault management strategies during training, potentially further enhancing fault tolerance. Additionally, while lower-bit quantization could further impact fault tolerance, it may also lead to a significant drop in accuracy, making its practical applicability challenging. Future research could explore strategies to mitigate this trade-off, such as advanced quantization-aware training techniques or hybrid approaches that balance model robustness and efficiency.

REFERENCES

[1] A. Garg and V. Mago, "Role of machine learning in medical research: A survey," *Comput. Sci. Rev.*, vol. 40, May 2021, Art. no. 100370.
[2] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *J. Field Robot.*, vol. 37, no. 3, pp. 362–386, Apr. 2020.
[3] C. Torres-Huitzil and B. Girau, "Fault and error tolerance in neural networks: A review," *IEEE Access*, vol. 5, pp. 17322–17341, 2017.
[4] C. De Sio, S. Azimi, and L. Sterpone, "FireNN: Neural networks reliability evaluation on hybrid platforms," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 549–563, Apr. 2022.
[5] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (DNN) accelerators and applications," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2017, pp. 1–12.
[6] R. Baumann, "Soft errors in advanced computer systems," *IEEE Design Test Comput.*, vol. 22, no. 3, pp. 258–266, May 2005.

[7] D. A. G. Gonçalves de Oliveira, L. L. Pilla, T. Santini, and P. Rech, "Evaluation and mitigation of radiation-induced soft errors in graphics processing units," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 791–804, Mar. 2016.

[8] N. Cavagnero, F. D. Santos, M. Ciccone, G. Averta, T. Tommasi, and P. Rech, "Transient-Fault-Aware design and training to enhance DNNs reliability with zero-overhead," in *Proc. IEEE 28th Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, Sep. 2022, pp. 1–7.

[9] L.-H. Hoang, M. A. Hanif, and M. Shafique, "FT-ClipAct: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 1241–1246.

[10] B. Ghavami, M. Sadati, Z. Fang, and L. Shannon, "FitAct: Error resilient deep neural networks via fine-grained post-trainable activation functions," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2022, pp. 1239–1244.

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

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

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[14] L. Ericsson, H. Gouk, and T. M. Hospedales, "How well do self-supervised models transfer?" in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Los Alamitos, CA, USA, Jun. 2021, pp. 5410–5419.

[15] S. Yfantidou, D. Spathis, M. Constantinides, A. Vakali, D. Quercia, and F. Kawsar, "Using self-supervised learning can improve model fairness," in *Proc. 30th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2024, pp. 3942–3953.

[16] Y.-H. Cao, P. Sun, Y. Huang, J. Wu, and S. Zhou, "Synergistic self-supervised and quantization learning," in *Proc. Eur. Conf. Comput. Vis.*, Jan. 2022, pp. 587–604.

[17] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, Oct. 2021.

[18] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. V. Baalen, and T. Blankevoort, "A white paper on neural network quantization," 2021, *arXiv:2106.08295*.

[19] R. Milazzo, V. De Marco, C. De Sio, S. Fosson, L. Morra, and L. Sterpone, "On the fault tolerance of self-supervised training in convolutional neural networks," in *Proc. 27th Int. Symp. Design Diag. Electron. Circuits Syst. (DDECS)*, Apr. 2024, pp. 110–115.

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

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

[22] K. Furutani, K. Arimoto, H. Miyamoto, T. Kobayashi, K. Yasuda, and K. Mashiko, "A built-in Hamming code ECC circuit for DRAMs," *IEEE J. Solid-State Circuits*, vol. 24, no. 1, pp. 50–56, Feb. 1989.

[23] Y. Saikawa and Y. Tomioka, "Approximated triple modular redundancy of convolutional neural networks based on residual quantization," in *Proc. IEEE 17th Int. Symp. Embedded Multicore/Many-core Syst.-Chip (MCSoC)*, Dec. 2024, pp. 302–309.

[24] J. Vafaei and O. Akbari, "HPR-mul: An area and energy-efficient high-precision redundancy multiplier by approximate computing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 32, no. 11, pp. 2012–2022, Nov. 2024.

[25] U. Zahid, G. Gambardella, N. J. Fraser, M. Blott, and K. Vissers, "FAT: Training neural networks for reliable inference under hardware faults," in *Proc. IEEE Int. Test Conf. (ITC)*, Nov. 2020, pp. 1–10.

[26] C.-T. Chin, K. Mehrotra, C. K. Mohan, and S. Rankat, "Training techniques to obtain fault-tolerant neural networks," in *Proc. IEEE 24th Int. Symp. Fault- Tolerant Comput.*, Jun. 1994, pp. 360–369.

[27] L. Iurada, N. Cavagnero, F. F. D. Santos, G. Averta, P. Rech, and T. Tommasi, "Transient fault tolerant semantic segmentation for autonomous driving," in *Proc. 3rd Workshop Uncertainty Quantification Comput. Vis.*, Aug. 2024, pp. 1–23.

[28] T. Tsai, S. K. S. Hari, M. Sullivan, O. Villa, and S. W. Keckler, "NVBitFI: Dynamic fault injection for GPUs," in *Proc. 51st Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2021, pp. 284–291.

[29] B. Fang, K. Pattabiraman, M. Ripeanu, and S. Gurumurthi, "GPU-qin: A methodology for evaluating the error resilience of GPGPU applications," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Mar. 2014, pp. 221–230.

[30] G. Li, K. Pattabiraman, and N. DeBardeleben, "TensorFI: A configurable fault injector for TensorFlow applications," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2018, pp. 313–320.

[31] A. Mahmoud, N. Aggarwal, A. Nobbe, J. R. S. Vicarte, S. V. Adve, C. W. Fletcher, I. Frosio, and S. K. S. Hari, "PyTorchFI: A runtime perturbation tool for DNNs," in *Proc. 50th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Jun. 2020, pp. 25–31.

[32] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, Jan. 2020, pp. 1–21.

[33] X. Chen and K. He, "Exploring simple Siamese representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 15745–15753.

[34] M. Caron, H. Touvron, I. Misra, H. Jegou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9630–9640.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[36] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[37] O. A. B. Penatti, K. Nogueira, and J. A. dos Santos, "Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2015, pp. 44–51.

[38] P. Helber, B. Bischke, A. Dengel, and D. Borth, "EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 7, pp. 2217–2226, Jul. 2019.

[39] W. Zhou, S. Newsam, C. Li, and Z. Shao, "PatternNet: A benchmark dataset for performance evaluation of remote sensing image retrieval," *ISPRS J. Photogramm. Remote Sens.*, vol. 145, pp. 197–209, Nov. 2018.

[40] H. Li, X. Dou, C. Tao, Z. Wu, J. Chen, J. Peng, M. Deng, and L. Zhao, "RSI-CB: A large-scale remote sensing image classification benchmark using crowdsourced data," *Sensors*, vol. 20, no. 6, p. 1594, Mar. 2020.

[41] M. Schmitt, L. Hughes, P. Ghamisi, N. Yokoya, and R. Hänsch, 2019, "IEEE GRSS data fusion contest," doi: 10.21227/rha7-m332.

[42] I. Dimitrovski, I. Kitanovski, D. Kocev, and N. Simidjievski, "Current trends in deep learning for Earth observation: An open-source benchmark arena for image classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 197, pp. 18–35, Mar. 2023.

[43] L. Scheibenreif, J. Hanna, M. Mommert, and D. Borth, "Self-supervised vision transformers for land-cover segmentation and classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 1421–1430.

[44] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *Proc. Int. Conf. Learn. Represent.*, Aug. 2019, pp. 1–21.

[45] S. Kornblith, J. Shlens, and Q. V. Le, "Do better ImageNet models transfer better?" in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2656–2666, doi: 10.1109/CVPR.2019.00277.

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

[47] L. Ericsson, H. Gouk, and T. M. Hospedales, "How well do self-supervised models transfer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 5410–5419.

ROSARIO MILAZZO received the bachelor's degree in computer engineering from Politecnico di Torino, in 2020, and the master's degree, in 2022. He is currently pursuing the Ph.D. degree in computer engineering, funded by NODES. In 2022, after his master's degree, he started working with Politecnico di Torino as a Scholarship Holder with the GRAINS Research Group, Department of Control and Computer Engineering (DAUIN), Politecnico di Torino. His main research interests include the optimization and fault tolerance of neural networks in the field of mobility and satellites.

SOPHIE M. FOSSON (Member, IEEE) received the M.Sc. degree in applied mathematics from Politecnico di Torino, Italy, in 2005, and the Ph.D. degree in mathematics for industrial technologies from Scuola Normale Superiore di Pisa, Italy, in 2011. From 2012 to 2016, she was a Postdoctoral Associate with the Department of Electronics and Telecommunications, Politecnico di Torino. She visited the Centre Tecnològic de Telecomunicacions de Catalunya, Spain, in 2013, 2014, and 2016. She was a Researcher with Istituto Superiore Mario Boella, Turin, Italy, in 2017. She is currently an Associate Professor with the Department of Control and Computer Engineering, Politecnico di Torino. Her main research interests include sparse optimization, machine learning, system identification, control, and cyber-physical systems. She is an Associate Editor for IEEE CONTROL SYSTEMS LETTERS.

LIA MORRA (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer engineering from Politecnico di Torino, Italy, in 2002 and 2006, respectively. From 2006 to 2017, she was a Researcher with IM3D, Turin, Italy, where she served as the Chief Scientific Officer, from 2014 to 2017, developing artificial intelligence systems for medical image interpretation. She is currently an Assistant Professor (Tenure Track) with the Department of Control and Computer Engineering, Politecnico di Torino. Her main research interests include deep learning, image interpretation, medical image analysis, and neuro-symbolic artificial intelligence. She is a member of the AAPM Computer-Aided Image Analysis Subcommittee (CADSC). She is an Associate Editor of *IEEE Consumer Electronics Magazine* and *Frontiers in Medical Engineering*.

LUCA STERPONE (Senior Member, IEEE) received the M.S. and Ph.D. degrees in computer engineering from Politecnico di Torino, Italy, in 2003 and 2007, respectively. He is currently a Full Professor with the Department of Control and Computer Engineering, where he is leading the Aerospace, Safety, and Computing Group. He has authored more than 220 articles. He received several awards for his research activities. His current research interests include reconfigurable computing, computer-aided design algorithms, fault tolerance architectures, and radiation effects on components and systems.