On the Fault Tolerance of Self-Supervised Training in Convolutional Neural Networks

On the Fault Tolerance of Self-Supervised Training in Convolutional Neural Networks

Rosario Milazzo, Vincenzo De Marco, Corrado De Sio, Sophie Fosson, Lia Morra, and Luca Sterpone
Department of Control and Computer Engineering, Polytechnic of Torino, Italy
{rosario.milazzo, corrado.desio, sophie.fosson, lia.morra, luca.sterpone}@polito.it, vincenzo.demarco@studenti.polito.it

***Abstract*—Deep neural networks (DNNs) are increasingly used in critical applications from healthcare to autonomous driving. However, their predictions were shown to degrade in the presence of transient hardware faults, leading to potentially catastrophic and unpredictable errors. Consequently, several techniques have been proposed to increase the fault tolerance of DNNs by modifying network structures and/or training procedures, thereby reducing the need for costly hardware redundancy. There are, however, design or training choices whose impact on fault propagation has been overlooked in the literature. In particular, self-supervised learning (SSL), as a pre-training technique, was shown to improve the robustness of the learned features, resulting in better performance in downstream tasks. This study investigates the fault tolerance of several SSL techniques on image classification benchmarks, including several related to Earth Observation. Experimental results suggests that SSL pretraining, alone or in combination with fault mitigation techniques, generally improves DNNs' fault tolerance, although the performance gap vary among datasets and SSL techniques.**

***Index Terms*—Reliability, Resilience, Fault-Tolerance, Error Mitigation, DNN, Machine Learning, 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]–[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. 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], [6]–[9]. Replication and redundancy, either at the hardware or software level, are effective but expensive solutions. However, recent works have shown that changes in the CNN architectures 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 [6]–[9].

Understandably, the current literature has so far focused on simple benchmarks, such as CIFAR-10 or CIFAR-100, relatively shallow CNNs, and standard training procedures. However, *modern CNNs are large and trained on vast datasets using a variety of complex techniques*. 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.

An important design choice is how the network is pre-trained. Since DNNs require large amounts of training data, they are seldom trained from scratch in real-life applications, but are rather fine-tuned after pre-training on a large scale dataset such as ImageNet [10]. More recently, self-supervised learning (SSL) techniques are increasingly graining traction as an alternative to supervised pretraining [11]. Since SSL methods do not require labels, they can scale up to extremely large scale datasets. *CNNs pre-trained via SSL were shown to achieve better performance in downstream tasks compared to standard ImageNet supervised pre-training* [11], [12]. At the same time, since *networks pre-trained via different techniques can have substantially different weight distributions* [13], we cannot assume that their fault tolerance is equivalent even though their accuracy might be similar.

Moving from this hypothesis, **we evaluated the impact of SSL pre-training on fault tolerance** both in the unmitigated scenario and in combination with fault mitigation techniques based on activation clipping. Experiments on complex benchmarks, from ImageNet to several Earth Observation (EO) classification task, show that *activation clipping is more effective when applied to SSL pretraining*. However, this property does not transfer equally well across all datasets and techniques.

The rest of the work is organized as follows. In Section II we first give a brief overview of the main error mitigation techniques and introduce the SSL technique. The fault injector and fault mitigation techniques used in our experiments are analyzed in Section III. The datasets and experimental settings are discussed in Section IV. Section V analyzes the results of our experiments by comparing the performance of the different configurations treated at multiple fault rates. Finally, in Section VI 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

979-8-3503-5934-3/24/$31.00 ©2024 IEEE

data types for storing network parameters, fortifying vulnerable bits, or introducing normalization techniques [5].

Hardware redundancy, such as Triple Modular Redundancy [14] 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 [15], or adopting error detection and correction techniques, such as Error Correction Codes [16].

A complementary approach tackles the problem at the software level, by looking at the design and training of the network itself [6], [7], [9], [17]. 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) [6], [17], architectural modifications [18], and activation clipping [6], [7], [9]. 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, but was shown to be less effective than architectural modifications [6]. 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 [6]–[9]. Activation clipping can be applied post-training, and does not introduce hardware-level overhead, making it ideal for a variety of scenarios. Different methods, such as Ranger [9] and ClipAct [7], 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. 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 amount of labelled data [11]. 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, aims at pulling similar samples closer in feature space while pushing diverse samples far from each other.

In this work, we used two different contrastive SSL techniques. **SwAV** (Swapping Assignments between Views) [19] is based on clustering images ensuring consistency between several augmented versions of the same image simultaneously. **DINO** [20], 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 [11], quantization [13] and robustness to adversarial examples or label corruption [12], to the best of our knowledge relationship with fault tolerance properties have not been investigated.

III. Methods

A. 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. Additionally, Earth is constantly bombarded by a stream of charged particles from cosmic rays. There are different types of SEE and they can be divided into two main categories: destructive and non-destructive. In our experiments, we assessed fault tolerance in the presence of Single Event Upsets (SEUs), which are non-destructive and transient faults affecting memories, leading to bit flips that can be rectified by rewriting the data in memory.

Fault injections (FI) was performed at runtime using an ad-hoc software tool. Specifically, we employed a customized version of the PytorchFI [21], a runtime perturbation tool designed for DNNs within the PyTorch framework. 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, and we considered both DNN weights and biases as potential fault locations. To simulate the transient effects of faults, we chose to inject different 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 faults n_{faults} and then randomly assigning each fault to a specific layer. The total number of faults is determined by 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 $\mathcal{O}(L \cdot H \cdot W \cdot C \cdot b)$, where L is the number of layers in which we want to inject faults, H and W are respectively height and width of the convolutional filter, C are the channels of the image (1 if

grayscale or 3 if RGB), and b the number of bits for each weight/bias value (32 in our case) to $\mathcal{O}(n_{faults})$.

B. Fault mitigation technique

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 [7]. We chose to exclusively utilize fault mitigation techniques based on the clipping of activations. This decision was made to avoid further modifications to the training phase and align with the current state of the art in SSL techniques, which relies on neural networks without any additional modifications. 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:

$$f(x) = \begin{cases} x & \text{if } 0 \leq x \leq T_l \\ 0 & \text{if } x > T_l \\ 0 & \text{if } x < 0 \end{cases} \quad (1)$$

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 a trained network. The key metric to be optimized is the accuracy at different fault rates, measured by the Area Under the Curve (AUC). Here, the curve reflects accuracy fluctuations as the fault rate changes. 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. 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. 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 generates four distinct AUC values, one for each threshold.
- 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].

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. Faults are injected one layer at a time to determine the most effective set of threshold parameters. 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 [6], but i) the threshold is not fixed, but 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.

TABLE I: 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	Size (training set)
ImageNet [10]	1431167	1000	varying
CIFAR-10	60000	10	32×32
CIFAR-100	60000	100	32×32
BCS [23]	2876	2	64×64
EuroSAT [24]	27000	10	64×64
PatterNet [25]	30400	38	256×256
RSI-CB256 [26]	24747	35	256×256

IV. Experimental settings

A. Datasets

As previously mentioned, five datasets were used, ImageNet [22] and four datasets containing satellite images:

- Brazilian Coffee Dataset (**BCS**) [23] has multispectral high-resolution scenes of coffee crops and non-coffee areas (binary classification task) taken by the SPOT (Satellite Pour l'Observation de la Terre) sensor in 2005.
- **EuroSAT** [24] is a benchmark for land use and land cover classification. It was acquired by Sentinel-2 and contains 27,000 labeled and geo-referenced images.
- **PatternNet** [25] is a large-scale remote sensing dataset covering a sample of US cities, collected from Google Earth imagery or via the Google Map API.
- **RSI-CB256** [26] is a large scale remote sensing image classification benchmark obtained via crowdsourcing.

To ensure the reproducibility and comparability of the results, we adopted the same dataset split as proposed in the AiTLAS benchmark [27], with training, validation, and test sets comprising 60%, 20%, and 20% of the total dataset, respectively. The splits are stratified by class and are available in the AiTLAS project repositories. In the experiments conducted on satellite datasets, we trained the DNN using the training set, calculated the optimal thresholds on the validation set, and subsequently evaluated performance on the test set. For the ImageNet dataset, we used the original dataset split. Since there is no public test set, to avoid biases we determined the ClipAct thresholds based on a subset of the training set consisting of 4800 images and assessed performance using the validation set.

B. CNN models

Whenever feasible, we evaluated fault tolerance on pre-trained models available from official repositories. All models were either trained directly using standard supervised learning,

or pre-trained using SSL and then fine-tuned for the target classification task. All experiments were conducted on a ResNet50 architecture and all the trainings were carried out without the injection of any faults. All ImageNet pretrained models were downloaded from the Pytorch library[1], including those trained with SSL techniques SwAV[2] [19] and DINO[3] [20]. For satellite image datasets, models trained with standard supervision were available by the AiTLAS project[4] [27]. Instead, models with SSL pretraining were fine-tuned by the authors (see Section IV-C for hyper-parameters) starting from the weights available by SwAV[2] [19] and DINO[3] [20], and obtained from the respective SSL tasks on ImageNet [22].

C. Hyperpameter settings

For each fine-tuning experiment, we used the same hyper-parameters that were used to train the corresponding version without SSL [27]. All images were resized to 256×256, and after applying data augmentation, a random crop of 224×224 was taken. We trained all networks to convergence using the RAdam optimizer [28] with a batch size of 128 and learning rate of 1e-3. A learning rate scheduler reduced the learning rate by a factor of 0.1 whenever the validation loss reached a plateau, with patience set to 5.

Fault injection hyper-parameter: Each trained models was evaluated both without fault injection (baseline) and at the following fault rates: [1×10^{-8}, 3×10^{-8}, 1×10^{-7}, 3×10^{-7}, 1×10^{-6}, 3×10^{-6}, 1×10^{-5}, 3×10^{-5}]. Extremely low fault rates in the range [0, 1×10^{-8}] were omitted as performance is almost identical to the baseline. On the other hand, fault rates higher than 3×10^{-5} were not taken into account. In this range, too many faults would have been introduced, leading to similar DNN performances across all configurations, essentially equivalent to that of a random classifier. To determine the ClipAct thresholds, we used a slightly narrower range [1×10^{-7}, 1×10^{-6}, 3×10^{-6}, 1×10^{-5}, 3×10^{-5}], and conducted a total of 10 iterations per layer.

To evaluate network performance, we used top-1 and top-5 accuracy for ImageNet, and top-1 accuracy for satellite datasets.

D. Hardware

All experiments were performed on an Intel® Core™ i9-9820X CPU @ 3.30GHz × 20, 32 GB of RAM and a NVIDIA RTX 2080Ti GPU with 12GB VRAM.

V. Results

We tested the efficacy of SSL pre-training, alone or in combination with fault mitigation techniques, by comparing the accuracy with those of unprotected DNNs. Accuracy distribution of ResNet-50 across different configurations and fault rates are reported in Figs. 1 and 2 for ImageNet and satellite image datasets, respectively. Average values are computed over 5 repetitions of the fault injection process.

[1]https://pytorch.org/vision/0.8/models.html
[2]https://github.com/facebookresearch/swav/tree/main
[3]https://github.com/facebookresearch/dino
[4]https://github.com/biasvariancelabs/aitlas-arena/tree/main

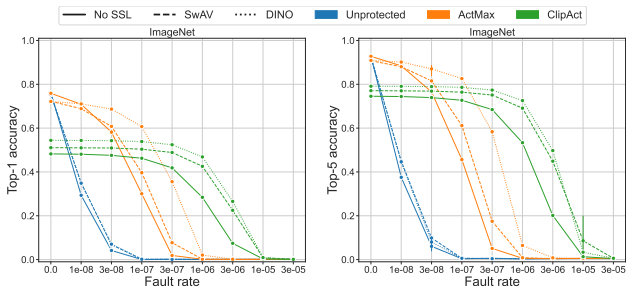

Fig. 1: Mean ResNet-50 top-1 and top-5 accuracy for unprotected model, ActMax and ClipAct [7], without SSL and with SwAV [19] and DINO [20] pretraining, on ImageNet dataset [22], under different fault rates.

When using standard supervised training (continuous lines in Figs. 1 and 2), 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}. In all cases, performance degrades to random guessing when the fault rate reaches 3×10^{-5}. It is worth reminding that the number of classes varies widely across datasets, and the chance-level accuracies at which the unprotected network plateaus vary accordingly, as shown in Fig. 2.

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. Since threshold fine-tuning aims to optimize performance across all fault rates, it converges to lower thresholds, even at the cost of degrading performance at lower fault rates, for which less aggressive thresholds would be preferrable, 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 dataset.

When using SSL pre-training (dashed lines in Figs. 1 and 2), the drop in accuracy for the unprotected DNN is similar across all configurations. However, *when activation clipping is used in conjunction with SSL pre-training*, 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. 2) *SSL pre-training does not yield significant advantages in the unprotected scenario or when using ClipAct*. On the other hand, with ActMax, SSL pre-training consistently outperforms supervised pre-training, but the performance gain is marginal compared to that offered by

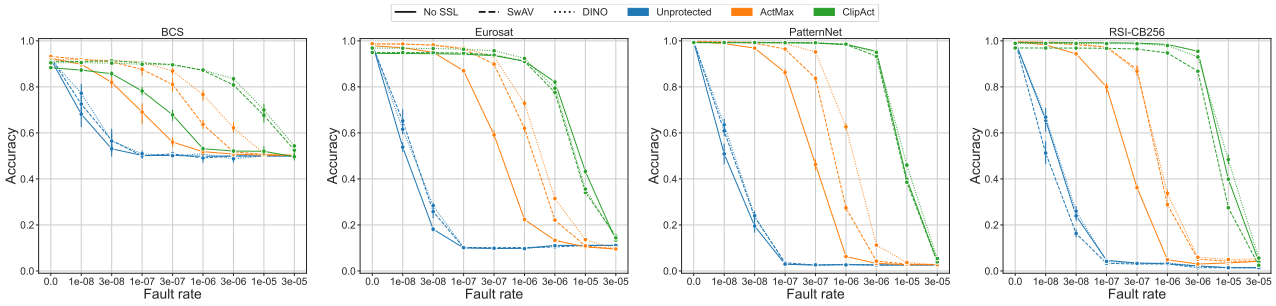

Fig. 2: Average ResNet-50 accuracy for unprotected model, ActMax and ClipAct [7], without SSL and with SwAV [19] and DINO [20] pretraining, on BCS [23], Eurosat [24], PatternNet [25] and RSI-CB256 [26] dataset, under different fault rates.

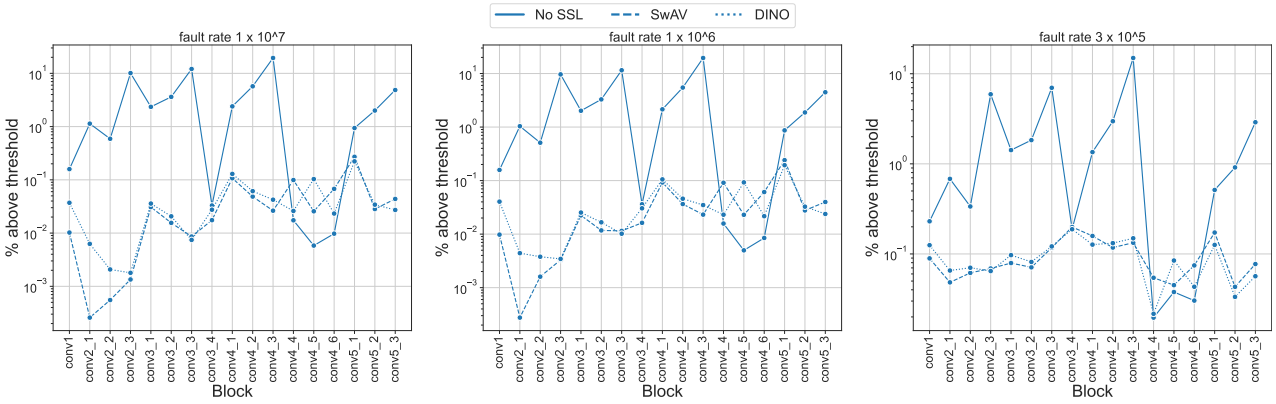

Fig. 3: Percentage of activations larger than the ClipAct's threshold for different residual blocks, assessed on the ImageNet dataset at fault rates of 1×10^{-7}, 1×10^{-6} and 3×10^{-5}. A logarithmic scale is used for ease of comparison.

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 pre-training 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 pre-training*; 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 pre-training performs worse than supervised pretraining.

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. 1. 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. 3, with standard supervised training, the percentage of activations higher than the threshold is generally much higher higher, up to and exceeding 10% in some blocks, with the exception of a few blocks, in which the percentages are nearly identical. This behaviour suggests that the features learnt in these blocks are robust enough to not be influenced by injected faults regardless of whether SSL pretrainig is used or not, moreover the enhancement of fault tolerance in these blocks is observed with an increase in depth. 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 pre-training technique* [13]. This analysis underscores 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*. This advantage is sometimes lost when fine-tuning to the target dataset, even though the activation distribution remains different, as evidenced by the behavior of ActMax. It is possible that, in the process of fine-tuning the network to a target dataset that is very different from the source distribution, the original features are partially destroyed as they are no longer useful.

VI. Discussion and conclusion

While pretraining on labelled datasets such as ImageNet is the de facto standard for training DNNs in a variety of domains [27], SSL pre-training is becoming increasingly attractive as it can scale to arbitrary large datasets, and was shown to increase performance and robustness on downstream tasks [11] In this paper we have investigated the impact of SSL pre-training on the accuracy of a DNN in the presence of transient hardware faults. Experimental findings on five classification benchmarks indicate that SSL pretraining, when used in conjunction with post-training activation clipping, outperforms or matches supervised pretraining in terms of baseline accuracy and fault tolerance. DNNs deployed in critical environments would thus benefit from SSL pretraining. Differences are observed among datasets, clipping techniques, and SSL pretraining algorithms, with DINO generally outperforming SwAV. In the optimal configuration (DINO pre-training in combination with ClipAct), all DNNs maintain an accuracy qualitatively close to the baseline up to a fault rate of 1×10^{-6}.

Activation clipping techniques are attractive thanks to their minimal overhead at inference time, and relatively low configuration cost. In the future, we plan to investigate other potentially effective techniques, such as fault-aware training [6] or trainable activation clipping [8], in combination with SSL pretraining. A further step in this direction would be to modify the SSL pretraining algorithm itself to increase fault resilience. Finally, considering the critical importance of fault tolerance in resource-constrained scenarios, such as the aerospace and automotive sectors, and recognizing the significant influence of quantization on fault resilience by acting as pseudo-activation-clipping mechanism similar to ClipAct and ActMax [29], we aim to expand our investigation to assess the influence of SSL pretraining on the fault tolerance of quantized networks.

Acknowledgement

This paper is part of the project NODES which has received funding from the MUR – M4C2 1.5 of PNRR funded by the European Union - NextGenerationEU (Grant agreement no. ECS00000036)

References

[1] A. Garg and V. Mago, "Role of machine learning in medical research: A survey," *Computer Science Review*, vol. 40, p. 100370, 2021.

[2] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, 11 2019.

[3] 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.

[4] C. De Sio, S. Azimi, and L. Sterpone, "FireNN: Neural networks reliability evaluation on hybrid platforms," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 549–563, 2022.

[5] G. Li et al., "Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications," in *SC17: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[6] N. Cavagnero, F. dos Santos, M. Ciccone, G. Averta, T. Tommasi, and P. Rech, "Transient-Fault-Aware Design and Training to Enhance DNNs Reliability with Zero-Overhead," 09 2022, pp. 1–7.

[7] 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. EDA Consortium, 2020, p. 1241–1246.

[8] B. Ghavami, M. Sadati, Z. Fang, and L. Shannon, "FitAct: Error Resilient Deep Neural Networks via Fine-Grained Post-Trainable Activation Functions," in *2022 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2022, pp. 1239–1244.

[9] Z. Chen, G. Li, and K. Pattabiraman, "A Low-cost Fault Corrector for Deep Neural Networks through Range Restriction," 06 2021, pp. 1–13.

[10] O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[11] L. Ericsson, H. Gouk, and T. M. Hospedales, "How Well Do Self-Supervised Models Transfer?" in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2021, pp. 5410–5419.

[12] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," *Advances in neural information processing systems*, vol. 32, 2019.

[13] Y.-H. Cao, P. Sun, Y. Huang, J. Wu, and S. Zhou, "Synergistic self-supervised and quantization learning," in *European Conference on Computer Vision*. Springer, 2022, pp. 587–604.

[14] R. E. Lyons and W. Vanderkulk, "The Use of Triple-Modular Redundancy to Improve Computer Reliability," *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, 1962.

[15] A. Mahmoud et al., "HarDNN: Feature Map Vulnerability Evaluation in CNNs," 2020.

[16] K. Furutani et al., "A built-in Hamming code ECC circuit for DRAMs," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 1, pp. 50–56, 1989.

[17] U. Zahid, G. Gambardella, N. Fraser, M. Blott, and K. Vissers, "FAT: Training Neural Networks for Reliable Inference Under Hardware Faults," 11 2020.

[18] C.-T. Chin, K. Mehrotra, C. Mohan, and S. Rankat, "Training techniques to obtain fault-tolerant neural networks," in *Proceedings of IEEE 24th International Symposium on Fault- Tolerant Computing*, 1994, pp. 360–369.

[19] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments," 2020.

[20] M. Caron et al., "Emerging Properties in Self-Supervised Vision Transformers," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[21] A. Mahmoud et al., "PyTorchFI: A Runtime Perturbation Tool for DNNs," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2020, pp. 25–31.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[23] O. A. Penatti, K. Nogueira, and J. A. Dos Santos, "Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?" in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2015, pp. 44–51.

[24] 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 Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

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

[26] 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."

[27] 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 Journal of Photogrammetry and Remote Sensing*, vol. 197, pp. 18–35, 2023.

[28] 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," 08 2019.

[29] M. A. Neggaz, I. Alouani, S. Niar, and F. Kurdahi, "Are CNNs Reliable Enough for Critical Applications? An Exploratory Study," *IEEE Design Test*, vol. 37, no. 2, pp. 76–83, 2020.