Open-Set Recognition: an Inexpensive Strategy to Increase DNN Reliability

G. Gavarini*, D. Stucchi†, A. Ruospo*, G. Boracchi†, E. Sanchez*
*Politecnico di Torino, DAUIN, Torino, Italy.
†Politecnico di Milano, Milano, Italy

***Abstract*—Deep Neural Networks (DNNs) are nowadays widely used in low-cost accelerators, characterized by limited computational resources. These models, and in particular DNNs for image classification, are becoming increasingly popular in safety-critical applications, where they are required to be highly reliable. Unfortunately, increasing DNNs reliability without computational overheads, which might not be affordable in low-power devices, is a non-trivial task. Our intuition is to detect network executions affected by faults as outliers with respect to the distribution of normal network's output. To this purpose, we propose to exploit Open-Set Recognition (OSR) techniques to perform Fault Detection in an extremely low-cost manner. In particuar, we analyze the Maximum Logit Score (MLS), which is an established Open-Set Recognition technique, and compare it against other well-known OSR methods, namely OpenMax, energy-based out-of-distribution detection and ODIN. Our experiments, performed on a ResNet-20 classifier trained on CIFAR-10 and SVHN datasets, demonstrate that MLS guarantees satisfactory detection performance while adding a negligible computational overhead. Most remarkably, MLS is extremely convenient to configure and deploy, as it does not require any modification or re-training of the existing network. A discussion of the advantages and limitations of the analysed solutions concludes the paper.**

***Index Terms*—Image Classification, Open-Set Recognition, Deep Neural Network, Reliability, Fault Detection**

I. Introduction

An increasing number of applications exploit Deep Neural Networks (DNNs) due to their effectiveness in tasks such as image classification, image segmentation or speech detection [1], [2]. DNNs are employed in automotive, robotics, finance, and many other sectors, thanks to their unprecedented performances. Not surprisingly, DNNs are often used in safety-critical applications, where there are strict requirements regarding fault tolerance and resilience (e.g., the ISO 26262 for the automotive industry).

DNNs are complex structures made of a huge number of parameters called weights. The number of weights and artificial neurons is often higher than the network needs, and this property, called over-provisioning, apparently makes the DNNs intrinsically fault-tolerant. Initially, neural network were believed to be inherently fault-tolerant. However, many studies are showing that this is not totally true (e.g., [3]–[5]). It is interesting to notice, for example, that neural networks have a highly parallel and distributed architecture where one weight is used for multiple computations. Therefore, a single fault in a single parameter can impact multiple computations. As many recent works have shown [6]–[8], analyzing the fault resilience of a neural network is of vital importance.

978-1-6654-7355-2/22/$31.00 ©2022 IEEE

An example of a safety-critical scenario is autonomous driving. In this domain, the model's resilience to faults and its accuracy are not the only meaningful design criteria. A DNN running in a car has rigorous computational requirements dictated by the low-power hardware accelerators. However, while the computational cost needs to remain low, the throughput of the network should not decrease since it has to examine a continuous input stream. Therefore, an additional design criterion is to minimize the latency between the moment a DNN receives an input and produces an output. There is a substantial interest from the scientific community, in minimizing the latency, the energy, and the power requirements of a network. Examples of this can be seen in hardware-aware Neural Architecture Search techniques [9], [10] that create networks that minimize the costs mentioned above.

The mechanisms used to increase the fault tolerance in a setting such as autonomous driving should not interfere with the other design parameters. For this reason, we propose an extraordinarily inexpensive Fault Detector that uses no more than a comparison of a single output value with a threshold to decide whether a fault has compromised the computation of the DNN. This, in particular, corresponds to the most straightforward solution this paper proposes, the Maximum Logit Score (MLS), which belongs to the family of Open-Set Recognition (OSR) techniques. These methods have been developed in the machine learning community to automatically determine whether an input belongs to one of the classes used to train the network [11]. To the best of our knowledge, we are the first to use these techniques for fault detection, and to illustrate how these inexpensively increase the reliability of a DNN classifier. In particular, we test a few OSR strategies from the most lightweight solutions, including MLS and [12], [13], to more complex ones such as *ODIN* [14], and others [15], [16], which have a considerable computational overhead. An additional advantage of the presented methodologies is that they treat the network as a black box: there is no need to know the DNN architecture, as these methods do not require any form of training or fine-tuning.

Different solutions targeting to improve DNN reliability have been implemented at very different levels and granularity, for example, approaches that mainly protect the static parameters of the DNN extensively exploit the *Error Correcting Codes* (ECC). However, such mechanisms are less effective when multiple faults occur, as described in [17]. Furthermore, ECCs cannot provide any protection to different faults than those happening in the memory. Santos et al. [18] propose an alternative algorithm to identify and correct errors in the network kernel matrix multiplications, dubbed *Algorithm-Based Fault-Tolerance* (ABFT). This technique can recognise up to 60% of critical faults. A high-level solution

is provided by [19], where the authors propose a fault detector called *Reliable Maxpooling*, where the output of all the network's Maxpool layers is compared against a threshold. On the one hand, this solution can recognize many critical faults. On the other, it requires an additional operation for every application of the Maxpool operator. It is worth nothing that this method also requires acting on internal operations of the network, which can be problematic for end-users, that rarely have the means of modifying the network. Instead, the solutions proposed in this paper treat the network as a black-box, requiring only the network's output to recognize faults. Another approach to fault tolerance is process replication [20], which consists of repeating the process to be monitored to detect and react to faults occurred during processing. Clearly, such a solution is often too expensive to be integrated in low-cost systems. In [21], the authors tackle the problem of fault tolerance in image processing applications, proposing the *Fault Impact Estimator*, that substitutes the process replication with an approximate replication followed by a lightweight detector based on a neural network, triggering a re-computation when a fault is detected.

The rest of the paper is organised as follows. Section II provides a broad summary of OSR techniques. Section III formally introduces the fault detection problem. Section IV describes the MLS and the other OSR methods used as Fault Detectors, while Section V outlines our case study. Then, Section VI reports the experimental results. Finally, Section VII draws some final conclusions.

II. Background

In image classification, Open-Set Recognition [11] is the task of recognizing when an input image does not belong to one of the classes represented in the network's training set. More precisely, an input is said to be in-distribution when it belongs to one of the pre-defined classes, otherwise, it is called out-of-distribution. To better understand why OSR techniques are needed, consider the CIFAR-10 dataset, composed of ten classes: aeroplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. A DNN trained on this dataset will assign any input samples to one of these in-distribution classes. When the network is fed with an image depicting a helicopter, it will assign the helicopter to one of the in-distribution classes, mislabelling the image. This is where OSR comes into play. When the network is fed with an out-of-distribution image (e.g., the helicopter), the OSR algorithm will mark it as an outlier, therefore discarding the network prediction.

Distinguishing in-distribution and out-of-distribution samples is a challenging task: many authors presented different solutions to this problem. Some solutions achieve good results simply by analysing the vector score or the logit, i.e., the output of the last layer of the network prior to the application of the Softmax. A simple solution used as a baseline in many works [15], [13], is to study only the maximum element of the logit, the Maximum Logit Score (MLS). Another example of this is OpenMax [12] that compares the logit of an inference with the average logit recorded during training. Other solutions (e.g., [15], [16]) require specific network training or fine-tuning in order to make in-distribution and out-of-distribution easier to distinguish. While an additional training increases the performances of OSR methods, it makes their application heavily context-dependent. Interestingly, there are OSR methods, like energy-based out-of-distribution detection [13], which achieve good results adding a negligible impact on DNN computational cost. Still, by fine-tuning the network to minimize a custom loss, it is possible to improve the performances of the method. Finally, some OSR methods employ more sophisticated techniques, such as Generative Adversarial Networks (GANs) [22].

III. Problem Formulation

We denote as $M_\theta : \mathcal{I} \rightarrow \mathcal{C}$ a DNN classifier, which maps an image $x \in \mathcal{I}$, belonging to the set of images $\mathcal{I}$ of a fixed size, to a label $M_\theta(x)$ belonging to the set of labels $\mathcal{C}$. We want to detect faults affecting the DNN, which can be either Silent Data Corruptions (SDC) [23] of parameters θ, or faults deriving from the hardware running the network. In this paper, we consider faults following the single-fault case, i.e., only one fault can affect the network at a given time. In particular, we are interested in detecting faults $M_\theta \mapsto \widehat{M_\theta}$ that are *critical* for an image x provided as input, namely faults that change the label the classifier would otherwise associate to x:

$$M_\theta(x) \neq \widehat{M_\theta}(x). \tag{1}$$

In the following, we refer to the output of the network not affected by faults as the *golden output*, as opposed to the *faulty output*.

Our primary desiderata is that critical faults are detected by an algorithm adding an inexpensive computational overhead. Moreover, the fault detection algorithm has to be practical to use, avoiding sophisticated configuration, as well as additional network training phases or fine-tuning. To develop our detector of critical faults, we assume that the golden outputs of the test set TS of images are provided.

IV. Proposed Approach

Open-Set Recognition methods have been presented in the machine learning literature to detect when an input x does not correspond to any label represented in the training set TR, thus when x is unknown to the network. Most of OSR methods build upon the assumption that images from unknown classes result in some out-of-distribution quantity during processing, which can be detected by specific statistical or machine learning techniques. Our intuition is that OSR methods can also detect when a critical fault

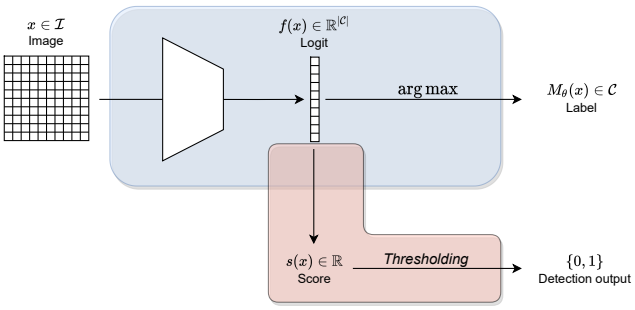


Fig. 1: The proposed Fault Detector (in red) analyzes the logit score $f(x)$ produced by the classification network (in blue) to detect critical faults. The detection consists in the computation of a *score* $s(x)$ and the comparison with a threshold γ, set to guarantee the FPR of the algorithm. In this work, we consider scores $s(\cdot)$ inspired by the OSR literature: the Maximum Logit Score $MLS(\cdot)$ (4), the energy-based score $E_T(\cdot)$ (5), and the OpenMax score $OM(\cdot)$ (6).

occurs, as this might result in some out-of-distribution quantity which an OSR method is expected to detect.

Typically, given an input image x, a classification network computes the score vector $f(x) \in \mathbb{R}^{|\mathcal{C}|}$, called *logit*, comprising the scores associated to each class. The higher the score, the most likely x belongs to the corresponding class. The logit is then fed to the Softmax activation function, that normalizes it into a probability vector, yielding the classifier's result. As a matter of fact, the Softmax normalization rescales extreme values in the logit, thus preventing the detection of out-of-distribution samples. For this reason, most OSR techniques analyze the logit vector $f(x)$ rather than its normalized counterpart.

In this work and for the very first time, we frame Fault Detection as an open set recognition problem, by using logit scores as samples to be classified as in-distribution (ID) or out-of-distribution (OOD). Indeed, we assume that logits from golden outputs follow a similar distribution, defining the in-distribution density. In our modeling assumption, non-critical faulty outputs are assimilated to golden ones, as we are not interested in detecting these. Finally, we expect outputs associated to critical faults to follow a different distribution (see [14], [15]), which gives rise to OOD samples. In this regard, the critical-fault detector becomes an out-of-distribution detector in the logit space.

Our approach to inexpensively detecting critical faults can be formulated as designing a function $D : \mathbb{R}^{|\mathcal{C}|} \rightarrow \{0, 1\}$ that analyzes the logit $f(x)$ to detect whether a critical fault affected the output:

$$D\left(f(x)\right) = \begin{cases} 1 & \text{if } M(x) \neq \widehat{M}(x) \\ 0 & \text{otherwise} \end{cases} . \tag{2}$$

Remarkably, in most of the considered OSR techniques, D simply performs a thresholding on a score $s(x)$ that can be directly computed from the logit $f(x)$, resulting in a negligible overhead with respect to the DNN operations. Such a detector, illustrated in Figure 1, can be expressed as:

$$D\left(f(x)\right) = \begin{cases} 1 & \text{if } s(x) \leq \gamma \\ 0 & \text{otherwise} \end{cases} , \tag{3}$$

where $s(x)$ is a score directly depending on $f(x)$ and on the considered OSR method, and γ is the detection threshold.

Since the golden outputs computed from the test set TS are provided, as stated in Section III, we can use them to categorize critical and non-critical faulty outputs. Hence, we propose to set the threshold γ to control false alarms, and we can compute its value from the set of non-critical faulty outputs yielded by TS (or a portion of it). Precisely, we compute the score function associated to every non-critical output, and we compute the α-quantile of that distribution, where α is the desired Fault Positive Rate (FPR) that we want our detector to achieve. Thanks to the statistical fault injection process [24] that we employ in our experiments and that will be presented in Section V, the threshold computed by setting the empirical FPR over TS guarantees that the desired FPR will be mantained over unseen faults (within a specific confidence level and error margin).

In the following, we employ some of the most interesting OSR techniques as a part of our low cost fault detection scheme. For each OSR technique, the corresponding score function used in (3) is also defined.

A. Maximum Logit Score

Several methods in the OSR literature [14], [15] investigate the relationship between the Maximum Logit Score (MLS), namely the maximum element of the logit, and samples from unknown classes. They have found that inputs that do not belong to one of the classes considered in the training set usually result in a significantly smaller MLS. Furthermore, they propose additional methods to increase the MLS of ID samples and decrease the one of OOD inputs, enhancing the difference between the two. Since our goal is to design a fault detector that does not require specific training and has a low computational cost at inference time, we consider to directly threshold the MLS, hence exploiting this characterization of OOD samples. The score function $s(\cdot)$ associated to the MLS is:

$$MLS(x) = \max_c \left[f(x)\right]_c, \tag{4}$$

where $\left[f(x)\right]_c$ denotes the c-th component of the logit score.

B. Energy-Based

The Energy-Based OSR [13] proposes to compute an *energy score* based on the logits to discriminate between ID and OOD samples, to avoid the compression caused by the Softmax function. According to [13], the energy score tends to be lower for OOD samples than for ID ones, making it possible to linearly separate them with a threshold. We setup a Fault Detector where the score $s(\cdot)$ is defined as the energy function $E_T : \mathcal{I} \rightarrow \mathbb{R}$:

$$E_T\left(x\right) = -T \cdot \log \sum_{c \in \mathcal{C}} e^{[f(x)]_c / T}, \tag{5}$$

where $T \in \mathbb{R}$ is the *temperature* characterizing the shape of the energy function. The temperature parameter is used to smooth the energy function output: using high values for this parameter results in similar values for the energy even for two samples that have very different logits.

Furthermore, [13] shows that the network can be fine-tuned to create a larger energy gap between non-critical and critical outputs. However, this is not viable for the considered application since it requires a deeper knowledge of the network and an additional fine-tuning phase, while this work focuses only in methods that can be used off-the-shelf on pre-trained DNNs. We set up our Fault Detector with a temperature parameter $T = 1$, without any additional training.

C. OpenMax

Following the same principle, [12] proposes to replace the Softmax activation function with the OpenMax function. The idea behind OpenMax is to compute the Mean Activation Vector (MAV) for each of the ID classes. The MAV for a given class is the average logit of all the correctly-classified samples in the training set. The further away a sample's logit is from the MAV of its predicted class, the higher the probability that such input is OOD. At inference time, the distance between the computed logit and the MAV distribution is used to revise the logit and add an *open-set class* (OC) score, yielding the *OpenMax vector*, namely a new score vector $f^{OM}(x) \in \mathbb{R}^{|\mathcal{C}|+1}$. Notice that $f^{OM}(x)$ can be computed directly from the MAV and the logit $f(x)$, without additional training or fine-tuning.

We define the score function $s(\cdot)$ associated to OpenMax as follows:

$$OM(x) = \begin{cases} -\infty & \text{if } \arg\max [f^{OM}(x)]_c = OC \\ \max_c [f^{OM}(x)]_c & \text{otherwise} \end{cases}, \quad (6)$$

where f^{OM} is computed as in [12] and OC represents the open-set class. We set $OM(x)$ to be equal to $-\infty$ when the predicted class is OC, such that the Fault Detector (3) always detects it as a critical fault. While not modifying the network, this solution requires to compute the MAV from a labeled set of images, which makes it slightly less practical than plain MLS.

D. ODIN

The final method examined by this work is ODIN [14], a technique that builds upon the analysis of the MLS by proposing temperature scaling and a pre-processing of the input. In particular, ODIN applies a small perturbation to the input image, a technique often used in adversarial attacks. However, while in an adversarial setting, the aim is to decrease the score of the prediction, here, the objective is to increase it. The authors in [14] show that the noise added to the input has a more significant impact on ID than on OOD samples, making it easier to separate them. The perturbation added by ODIN is the gradient of the network loss function with respect to the input. Clearly, ODIN is much more expensive than the methods presented until now, as it requires the computation of the perturbation δ (via a backward pass) and an additional forward pass, where it computes a new logit $f(x')$, where $x' = x + \delta$. However, this technique can serve as an interesting benchmark for the other methods as it does not require any additional training or fine-tuning of the network.

We setup a Fault Detector based on ODIN, where the score function $s(\cdot)$ is defined as:

$$ODIN\,(x') = \max_c [f(x')]_c, \quad (7)$$

where $x' = x + \delta$ is the perturbed input image computed as in [14]. Notice that this solution does not align with the scheme depicted in Figure 1.

V. Case Study

This work assesses the performance of various OSR methods in detecting faults injected in ResNet-20 [2], a convolutional neural network for image classification. ResNet-20 consists of 19 convolutional layers and one fully connected layer, for a total of 268,346 parameters. In our experiments, we train and test ResNet-20 on two different datasets, CIFAR-10 [25] and SVHN [26]. CIFAR-10 contains 60k 32x32x3 images belonging to 10 different classes. SVHN contains almost 100k 32x32x3 images representing digits in real-world photos. Our instances of ResNet-20 reach a 95.41% accuracy on the CIFAR-10 test set and an accuracy of 96.12% on the SVHN test set.

In this work, we target permanent faults affecting the weights of the network under the single fault assumption. In the literature, this is a common approach to model faults happening in memory and caused by phenomena such as radiation, aging or temperature [23]. We simulate a permanent fault in memory affecting a network weight as a stuck-at fault in one of its bits. Since each weight is represented as a 32-bit floating-point number, it is possible to inject 64 different faults for each weight (for each bit, stuck-at-zero or stuck-at-one). In particular, ResNet-20 contains 268,346 weights, and an exhaustive fault injection campaign requires 17,174,144 single-fault injection campaigns. However, we avoid the cost of an exhaustive search by taking advantage of statistical fault injection [24], and computing the number of injection campaigns necessary to model the behaviour of the network in case of faults. With this method, we avoid injecting all the N possible faults by selecting a subpopulation of n sample faults. In particular, the number of samples n is selected so that the difference between the percentage of critical faults in the population p_N and the percentage of critical faults in the sample p_n is lower than a fixed error margin e (i.e., $p_N \in [p_n - e, p_n + e]$ with a confidence level t. In our experiments, by selecting a confidence level $t = 99\%$ and an error margin $e = 1\%$, we only need 16,609 fault injection campaigns to produce a statistically sound assessment of the network behavior. In fact, using a statistical fault injection guarantees that the metrics used to assess the detection performance presented below are in line with those that would be measured with an exhaustive fault injection campaign.

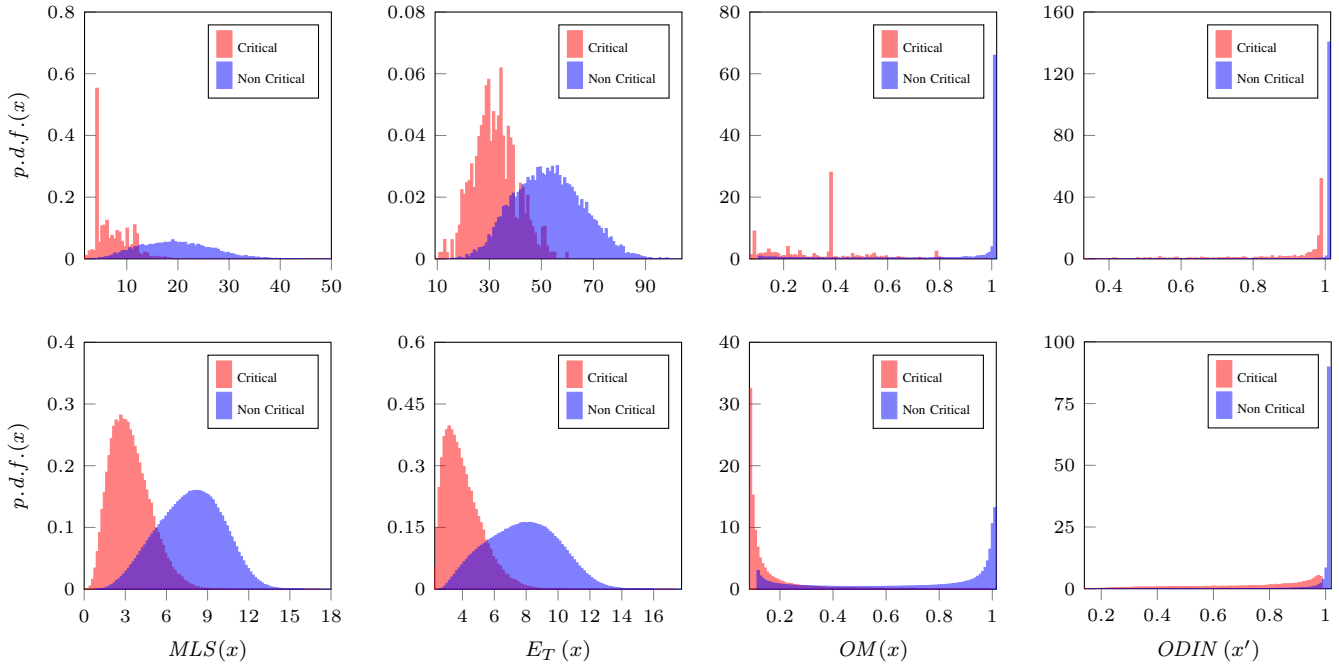
In the following experiments, a fault that changes the prediction of the network (SDC-1 [27]) is considered a critical fault. The performances of the proposed Fault Detector are measured using the same metrics used for evaluating OSR methods:

1) **False Positive Rate (FPR)**: the percentage of non-critical faults flagged as critical;
2) **True Positive Rate (TPR)**: the percentage of critical faults detected;
3) **Area Under The Curve (AUC)**: the area under the Receiver Operating Characteristic (ROC) curve that plots FPR against TPR. The closer the AUC is to 1, the better a detector is.

VI. Experimental Results

This work proposes to study the ability of the Open-Set Recognition approaches presented in Section IV to detect critical faults affecting a DNN through a statistical fault injection campaign, as described in Section V. The first step is to execute the fault-free network over the test set, obtaining the golden output. Then, we repeat the fault injection a statistically significant number of times. For each fault injection, we compute the faulty outputs and categorize the results as *critical* when they change the golden prediction, and *non-critical* in the contrary case. During this process, we filter out all the vector scores and logits that contain at least one *NaN* or infinite value, since they do not constitute silent faults. Finally, we compute the score that each of the considered OSR method associates to the outputs. We use the score of the non-critical outputs to set a detection threshold γ (see Section IV), and we perform the detection as in (3). To assess the performance of the methods, we compute the metrics presented in section V.

The MLS can be easily computed from the faulty output by taking the maximum of the logit (4). Similarly, the energy-based approach only requires to apply the energy function (5) to the logit vector. On the other hand, to evaluate the OpenMax function it is necessary to compute the MAV of each class from the training set, which is required to revise the faulty outputs logit. Then, it is only a matter of taking the maximum of the newly-computed logit vector (6). Finally, the application of ODIN requires to derive the gradient loss associated to each faulty output, which is used to compute fault-and-image specific perturbations of the input. Then, the MLS of the logits computed from the perturbed inputs (7) are



(a) MLS (b) Energy-Based (c) Open-Max (d) ODIN

Fig. 2: Distributions of the critical (red) and non-critical (blue) scores associated to each of the considerd OSR methods, computed over the test set TS by ResNet-20 trained on CIFAR-10 (top) and SVHN (bottom). Our solution (3) consists of a thresholding of the depicted scores, hence the performance increase when these are clearly separable.

used for detection. It is worth nothing that, for all the methods that require a temperature hyper-parameter, we tested various values. However, the best results were achieved by setting temperature $T = 1$ both for Energy-Based method and for ODIN.

The Fault Detectors proposed in this work are setup in a supervised manner. In fact, the detection thresholds are computed from the distribution of the scores yielded by non-critical faults to achieve a fixed FPR. Precisely, the threshold γ is the α-quantile of the non-critical scores, where α is the target FPR that we want to achieve. Setting up the considered methods with the same FPR allows us to perform a fair comparison.

A. Score Distributions

In Figure 2, we report the scores achieved by the considered OSR methods over critical and non-critical outputs computed from CIFAR-10 (above) and SVHN (below). The distribution of the scores achieved by MLS (Figure 2a) over critical and non-critical outputs have clearly different means, which makes them linearly separable, even if their overlap is non-negligible. The energy score (Figure 2b) exhibits a similar trend. However, as proven by the results further in this section, the overlap between the two distributions is more considerable than for the MLS. Conversely, OpenMax presents a different behaviour: the distributions of the scores of critical and non-critical outputs, reported in Figure 2c, result in very distinct peaks. However, it is important to notice the tails of the two distributions present a significant overlap. Finally, ODIN paints a completely different picture (Figure 2d). While the score of critical faults is almost evenly spread, the value of non-critical faults rarely goes under a certain value. As such, ODIN has the potential for a greater separation of critical and non-critical faults, especially for lower thresholds.

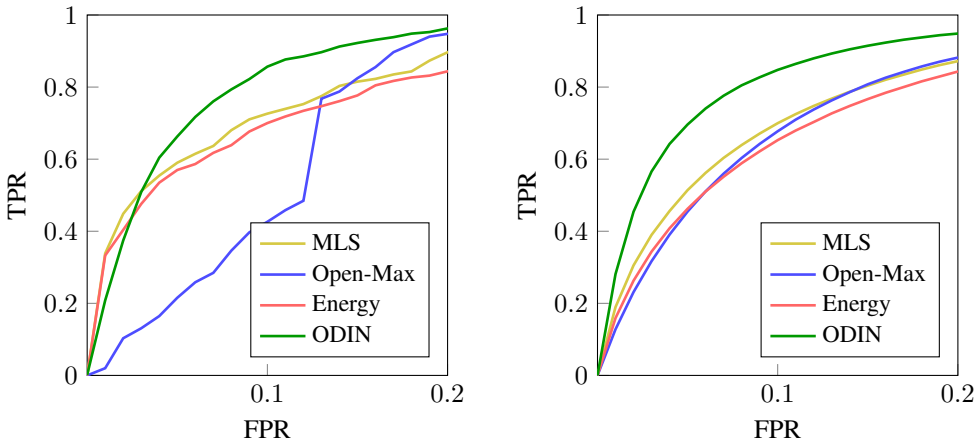
B. Results

Section IV presents the proposed solution, which consists in thresholding the score values computed from faulty outputs. In order to compare different models, we set them up with a threshold that guarantees the same FPR, namely the same rate of non-critical faults detected as critical. Indeed, FPR and TPR usually increase when the threshold decreases, thus we set the same desired FPR to perform a fair comparison.

Table I reports the TPR achieved by the considered methods over both datasets for different values of FPR, and we can appreciate how the performance of different methods change with the increase of the FPR. In particular, we notice that MLS and the energy-based method perform similarly on both datasets at

TABLE I: Comparison between FPR and TPR for the presented methods, with ResNet-20 on CIFAR-10 and SVHN

Dataset	FPR [%]	TPR [%]			
		MLS	Open-Max	Energy-Based	ODIN
CIFAR-10	1	33.85	2.21	33.28	20.82
	10	72.63	38.90	70.01	85.67
	20	85.58	89.66	84.36	96.27
SVHN	1	18.95	12.80	15.91	28.12
	10	69.98	67.74	65.27	84.77
	20	87.19	88.18	83.14	94.84



(a) CIFAR-10 (b) SVHN

Fig. 3: The ROC curves of the proposed Fault Detector set up with different OSR methods to detect faults in ResNet-20 trained on CIFAR-10 (left) and SVHN (right). The plot only shows FPR values between 0 and 0.2, since higher values are deemed to be insignificant for the task of fault detection.

every FPR level, with the former being always slightly better than the latter. Moreover, OpenMax achieves the lowest TPR when the FPR is low, surpassing MLS and energy-based only when the FPR reaches 20%. Surprisingly, despite being far more expensive, the performance of ODIN are always comparable with those of MLS and energy-based, especially for low values of FPR.

Another useful metric to compare different methods is the AUC, which measures the area under the Receiver Operating Characteristic (ROC) curve, namely the line plotting FPR values against the corresponding TPR values. Larger values of AUC correspond to better performance, 1 being the best. Figure 3 reports the ROC curves for the considered methods over CIFAR-10 (a) and SVHN (b). Consistently with the data shown in Table I, the better performing method is ODIN, even if for lower FPR MLS provides comparable results. OpenMax and energy-based lag behind, with the former performing badly on CIFAR-10. However, all the methods reach a 100% TPR when FPR is between 40% and 70% for CIFAR-10, while around 98% for SVHN, as shown in Table II. ODIN is an exception, since it never reaches a 100% TPR for CIFAR-10. The reader should be aware that, while a full coverage of critical faults is achieved only for very high false positive rates, the ROC curves reach 90% TPR rate early on, tending asymptotically to 100%.

Finally, in Table III reports the AUC values for all the proposed methods.

C. Considerations on ODIN

In this work, we compare extremely inexpensive methods with ODIN, which is prohibitively expensive compared to the others. In fact, for each inference, ODIN requires computing the gradient of the network loss function (often referred to as *backpropagation*) and then to compute the classification output associated to the input image perturbed by the gradient. Consequently, the cost of an inference is more than tripled. Proper triplication strategies, such as the one proposed in [28] can provide better results. As such, ODIN is an example of an expensive OSR solution that is used in this work as a benchmark of the capability of other OSR-based solutions, but cannot compare with equivalent-cost fault detection solutions in the literature.

TABLE II: FPR necessary to reach 100% TPR for the different methods on both datasets.

	FPR [%] for 100% TPR	
	CIFAR-10	**SVHN**
MLS	69	98
Energy-Based	56	98
Open-Max	45	99
ODIN	-	97

TABLE III: AUC of the different methods for ResNet-20 on CIFAR-10 and SVHN.

	AUC	
	CIFAR-10	**SVHN**
MLS	0.93	0.91
Energy-Based	0.93	0.90
Open-Max	0.85	0.91
ODIN	0.95	0.95

VII. Conclusions

Reliability and Fault Detection in Deep Neural Networks are crucial problems for many applications. In this work, we successfully used well-known Open-Set Recognition methods adapted to the detection of critical faults affecting classification networks, namely faults leading to a change in the predicted label. Remarkably, most of the considered methods only rely on the logit produced by the network, leading to inexpensive solutions, characterized by a very low computational cost at inference time and no need for network retraining. The results obtained in this paper, other than demonstrating the effectiveness of simple OSR techniques, encourage the application of more sophisticated OSR methods. However, using more elaborated methods brings new challenges and requires a more precise definition of in-distribution and out-of-distribution samples in a fault-affected DNN. Remarkably, we are working on the extension of the proposed approach while targeting different fault models, affecting for example the processing elements running the DNN.

REFERENCES

[1] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[3] V. Piuri, "Analysis of fault tolerance in artificial neural networks," *Journal of Parallel and Distributed Computing*, vol. 61, no. 1, pp. 18 – 48, 2001.

[4] 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.

[5] E. M. El Mhamdi and R. Guerraoui, "When neurons fail," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2017, pp. 1028–1037.

[6] Y. He, P. Balaprakash, and Y. Li, "Fidelity: Efficient resilience analysis framework for deep learning accelerators," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. Athens, Greece: IEEE, 2020, pp. 270–281. [Online]. Available: https://doi.org/10.1109/MICRO50266.2020.00033

[7] A. Lotfi, S. Hukerikar, K. Balasubramanian, P. Racunas, N. Saxena, R. Bramley, and Y. Huang, "Resiliency of automotive object detection networks on gpu architectures," in *2019 IEEE International Test Conference (ITC)*, 2019, pp. 1–9.

[8] A. Ruospo and E. Sanchez, "On the reliability assessment of artificial neural networks running on ai-oriented mpsocs," *Applied Sciences*, vol. 11, no. 14, 2021.

[9] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 734–10 742.

[10] L. Sekanina, "Neural architecture search and hardware accelerator co-search: A survey," *IEEE Access*, vol. 9, pp. 151 337–151 362, 2021.

[11] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1757–1772, 2013.

[12] A. Bendale and T. E. Boult, "Towards open set deep networks," *CoRR*, vol. abs/1511.06233, 2015. [Online]. Available: http://arxiv.org/abs/1511.06233

[13] W. Liu, X. Wang, J. D. Owens, and Y. Li, "Energy-based out-of-distribution detection," *CoRR*, vol. abs/2010.03759, 2020. [Online]. Available: https://arxiv.org/abs/2010.03759

[14] Y. Hsu, Y. Shen, H. Jin, and Z. Kira, "Generalized ODIN: detecting out-of-distribution image without learning from out-of-distribution data," *CoRR*, vol. abs/2002.11297, 2020. [Online]. Available: https://arxiv.org/abs/2002.11297

[15] S. Vaze, K. Han, A. Vedaldi, and A. Zisserman, "Open-set recognition: A good closed-set classifier is all you need," *CoRR*, vol. abs/2110.06207, 2021. [Online]. Available: https://arxiv.org/abs/2110.06207

[16] D. Hendrycks, M. Mazeika, and T. G. Dietterich, "Deep anomaly detection with outlier exposure," *CoRR*, vol. abs/1812.04606, 2018. [Online]. Available: http://arxiv.org/abs/1812.04606

[17] L. Matanaluza *et al.*, "Emulating the effects of radiation-induced soft-errors for the reliability assessment of neural networks," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2021.

[18] F. Fernandes dos Santos, L. Draghetti, L. Weigel, L. Carro, P. Navaux, and P. Rech, "Evaluation and mitigation of soft-errors in neural network-based object detection in three gpu architectures," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2017, pp. 169–176.

[19] F. F. dos Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, "Analyzing and increasing the reliability of convolutional neural networks on gpus," *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, 2018.

[20] P. Pop, V. Izosimov, P. Eles, and Z. Peng, "Design optimization of time- and cost-constrained fault-tolerant embedded systems with checkpointing and replication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 3, pp. 389–402, 2009.

[21] C. Bolchini, G. Boracchi, L. Cassano, A. Miele, and D. Stucchi, "Fault impact estimation for lightweight fault detection in image filtering," *IEEE Transactions on Computers*, 2020.

[22] L. Neal, M. Olson, X. Fern, W.-K. Wong, and F. Li, "Open set learning with counterfactual images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[23] 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 *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[24] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in *2009 Design, Automation Test in Europe Conference Exhibition*, 2009, pp. 502–506.

[25] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[26] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[27] 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 *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[28] A. Ruospo, G. Gavarini, I. Bragaglia, M. Traiola, A. Bosio, and E. Sanchez, "Selective hardening of critical neurons in deep neural networks," in *2022 25th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 2022, pp. 136–141.