

DeepVigor: Vulnerability Value Ranges and Factors for DNNs' Reliability Assessment

Mohammad Hasan Ahmadilivani¹, Mahdi Taheri¹, Jaan Raik¹, Masoud Daneshtalab^{1,2}, and Maksim Jenihhin¹

¹Tallinn University of Technology, Tallinn, Estonia

²Mälardalen University, Västerås, Sweden

¹{mohammad.ahmadilivani, mahdi.taheri, jaan.raik, maksim.jenihhin}@taltech.ee

²masoud.daneshtalab@mdu.se

Abstract—Deep Neural Networks (DNNs) and their accelerators are being deployed ever more frequently in safety-critical applications leading to increasing reliability concerns. A traditional and accurate method for assessing DNNs' reliability has been resorting to fault injection, which, however, suffers from prohibitive time complexity. While analytical and hybrid fault injection/analytical-based methods have been proposed, they are either inaccurate or specific to particular accelerator architectures.

In this work, we propose a novel accurate, fine-grain, metric-oriented, and accelerator-agnostic method called DeepVigor that provides vulnerability value ranges for DNN neurons' outputs. An outcome of DeepVigor is an analytical model representing vulnerable and non-vulnerable ranges for each neuron that can be exploited to develop different techniques for improving DNNs' reliability. Moreover, DeepVigor provides reliability assessment metrics based on vulnerability factors for bits, neurons, and layers using the vulnerability ranges.

The proposed method is not only faster than fault injection but also provides extensive and accurate information about the reliability of DNNs, independent from the accelerator. The experimental evaluations in the paper indicate that the proposed vulnerability ranges are 99.9% to 100% accurate even when evaluated on previously unseen test data. Also, it is shown that the obtained vulnerability factors represent the criticality of bits, neurons, and layers proficiently. DeepVigor is implemented in the PyTorch framework and validated on complex DNN benchmarks.

I. INTRODUCTION

Deep Neural Networks (DNNs) have recently emerged to be exploited in a wide range of applications. DNN accelerators have also penetrated into safety-critical applications e.g., autonomous vehicles [1], [2]. Therefore, several concerns are raised regarding developing and utilizing DNN accelerators in the realm of safety-critical applications, one of them being the reliability.

Reliability of DNNs concerns their accelerators' ability to perform correctly in the presence of faults [3] originating from either the environment (e.g., soft errors, electromagnetic effects, temperature variations) or inside of the chip (e.g., manufacturing defects, process variations, aging effects) [1], [4]. As shown in Fig. 1, faults may occur in different locations of accelerators either in memory or logic components and they influence the

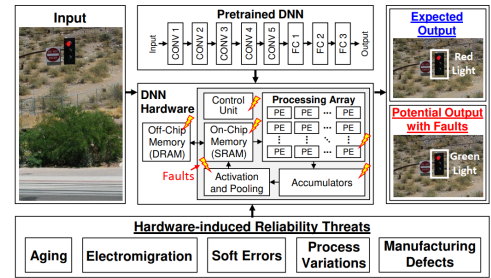


Fig. 1: Hardware reliability threats in DNN accelerators and their impact on the output [1].

parameters (e.g., weights and bias) and intermediate results (layers' activations) of neural networks that can decrease their accuracy drastically [5], [6]. By technology miniaturization, the effect of Single Event Transient (SET) and Single Event Upset (SEU) faults in devices is increasing thereby jeopardizing the reliability of modern digital systems [7].

Recently, several works have been published on the assessment and improvement of the reliability of a variety of DNNs as well as on different levels of system hierarchy [3], [4], [8]. Reliability assessment is the process of modeling the target DNN accelerator and measuring its reliability with respect to the corresponding quantitative evaluation metrics. Reliability assessment is the underlying procedure for improving reliability since it presents how the system could be influenced by threats as well as which locations of the system are more vulnerable to them. Therefore, it is the very first and principal phase of a reliable design process.

Throughout the literature, reliability assessment methods for DNNs are mainly categorized into two major classes: fault injection (FI) and resilience analysis. The majority of the works assess the reliability of DNNs relying on FI, which provides realistic results on the impact of different fault models on the system's execution and is performed directly on the target platform (accelerator's software [9] or RTL model [10], FPGA [11], GPU [12]). FI outputs different evaluations for DNNs' reliability by accuracy loss, vulnerability factors, or fault classification [11], [13], [14]. Moreover, fine-grain evaluations for finding critical bits can be performed by exhaustive FI or an optimized method in [15].

Nevertheless, FI methods are prohibitively time-consuming and carry a high complexity due to the need to inject an enormous amount of faults into a huge number of DNN parameters as

arXiv:2303.06931v1 [cs.LG] 13 Mar 2023

The work is supported in part by the European Union through European Social Fund in the frames of the "Information and Communication Technologies (ICT) programme" ("ITA-IoIT" topic), by the Estonian Research Council grant PUT PRG1467 "CRASHLES" and by Estonian-French PARROT project "EnTrustED".

well as time instances to reach an acceptable confidence level [16], [17]. The more fine-grain evaluation is required the more sophisticated experiments should be performed. In addition, most faults in a FI experiment on DNNs are masked [18] and are thus unnecessarily examined. Furthermore, the outcome of such assessment is application/platform specific which can not be generalized for other platforms [19].

Resilience analysis methods cope with the drawbacks of FI. They analyze the function of DNNs mathematically and have the potential to evaluate their reliability with arbitrary metrics. Therefore, resilience analysis methods can provide a deeper insight into the reliability evaluations of DNNs with lower complexity. Moreover, they can be conducted in different fault-tolerant designs on various platforms [20].

Layer-wise Relevance Propagation (LRP) algorithm is leveraged in [21]–[24] to obtain the contribution of neurons to the output to express their criticality and apply protections to improve the reliability of DNN accelerators. The sensitivity of DNN’s filters is obtained by Taylor expansion with given error rates in [25] for designing an error-resilient and energy-efficient accelerator.

The conducted resilience analyses in these works are not able to provide reliability measurement metrics and detailed vulnerability evaluations. Moreover, they combine the criticality scores of neurons over individual outputs of the DNNs, thus resulting in missing important information about the resilience of DNNs as a whole. Mahmoud et al. [20] proposed different heuristics for vulnerability estimation of feature maps without FI. These estimations which are more coarse grain than the LRP-based methods, lead to hardening the accelerators, however, the accuracy of the vulnerability estimation methods is remarkably lower than that of fault-injection methods.

The aforementioned papers on resilience analysis methods have focused mainly on finding the most critical neurons/weights in a DNN to protect them against faults in a fault-tolerant design. In addition, they do not explain sufficiently how a fault propagates through the network and influence its outputs. Fidelity framework [26] is proposed to take advantage of both FI and analyzing DNN accelerators to provide reliability metrics. However, it requires detailed information of the accelerator architecture/implementation. To the best of our knowledge, there is no accelerator-agnostic resilience analysis method for DNNs that can compete with FI in terms of reliability evaluation to be less time-consuming, and accurate with fine-grain metrics enabling different reliability improvement techniques.

In this research work, we introduce the concept of neurons’ vulnerability ranges expressing whether or not a fault at the output of neurons would misclassify the network. Thus, it enables a comprehensive reliability study with a novel resilience analysis method called DeepVigor where the vulnerability factors of layers, neurons, and bits in a DNN are obtained. The contributions in this work are:

- Proposing DeepVigor, a novel accurate, metric-oriented, and accelerator-agnostic resilience analysis method for DNNs reliability assessment faster than fault injection;
- Introducing and acquiring vulnerability ranges for all

neurons in DNNs, assisted by a fault propagation analysis, providing accurate categorization of critical/non-critical faults;

- Providing fine-grain vulnerability factors as reliability evaluation metrics for layers, neurons, and bits in DNNs, compared with and validated by fault injection.

The remainder of the paper is organized as follows: the resilience analysis method is presented in Section II, and the experimental setup and results are provided in Section III. The applicability of the method is discussed in Section IV, and the work is concluded in Section V.

II. DNN RELIABILITY ASSESSMENT WITH DEEPVIGOR

A. Fault Model

In this work, the fault propagation analysis is performed at the outputs of DNN neurons. However, they will cover a vast majority of internal faults of the neurons occurring inside the MAC units and also a large portion of faults in the weights and neurons’ input activations. It is assumed that only one neuron has an erroneous output per execution due to faults which is a common assumption in the literature [15].

For validation by FI, the single-bit fault model has been applied. While the multiple-bit fault model is more accurate, it requires a prohibitively large number of fault combinations to be considered ($3^n - 1$ combinations, where n is the number of bits). Fortunately, it has been shown that high fault coverage obtained using the single-bit model results in a high fault coverage of multiple-bit faults [27]. Therefore, a vast majority of practical FI and test methods are based on the single-bit fault assumption. Single bitflip faults are injected randomly at neurons’ outputs and once per execution.

B. Fault Propagation Analysis

Fig. 2 depicts an overview of the rationale behind the DeepVigor method. A tiny neural network with few layers and neurons with given inputs, golden (fault-free) activation values (inside of neurons), and weights (on the arrows) is shown. The golden classification output is *class1*. A fault changes the neuron’s output by δ which is the difference between the golden and faulty activation values. This δ that can have either a negative or a positive value will be propagated to the output layer and may change the classification result. The fault propagation will make a difference on each output class as Δ_1 and Δ_2 . Misclassification happens when the value of the output neuron *class2* gets higher than that of neuron *class1*.

Thus, the propagation of the fault can be traced from the neuron to the output and a problem for misclassification can be expressed as shown in Fig. 2. By solving the problem of misclassification condition in the output, the value for δ is obtained as a vulnerability threshold that expresses how much a fault should influence the neuron to misclassify the network. Therefore, a vulnerability value range for the neuron is acquired. In this example, the range $(-\infty, -5.39)$ is a vulnerable range and $[-5.39, +\infty)$ is non-vulnerable range. This idea is generalized for a DNN including multiple output classes and other corresponding functions in this paper.

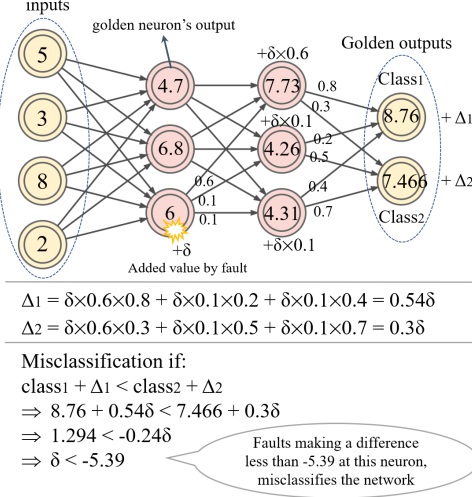


Fig. 2: An example of fault propagation analysis model and finding the vulnerability value ranges for a neuron with a given input.

C. The DeepVigor Method

The steps of the proposed DNNs' resilience analysis method (DeepVigor) and its validation are illustrated in Fig. 3. As shown, an analysis is performed on a set of data (i.e., set1, training set) and outputs the vulnerability value ranges as well as the vulnerability factors. Furthermore, FI is performed on the same and different data (i.e., set2, test set) to validate the outcomes of the analysis. The steps of DeepVigor are as follows:

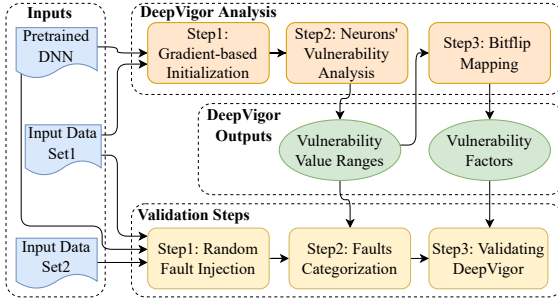


Fig. 3: Steps of the DeepVigor method for DNNs' reliability assessment and its validation.

Step1 - Gradient-based Initialization: In the first step, a neuron is examined whether or not to be processed for the vulnerability analysis. For this purpose, assuming a neural network consisting of L layers with N output classes in $C = \{c_1, c_2, \dots, c_N\}$. Neuron k at layer l is selected to be examined. The neuron's output is corrupted by adding a sample positive or negative value as ϵ_k^l to its output and the feed-forward of the network is executed over a batch of input data. A loss function \mathcal{L} is defined in Equation (1) as:

$$\mathcal{L} = \text{sigmoid}\left(\sum_{j=0}^N (\mathcal{E}_{c_t} - \mathcal{E}_{c_i})\right) \quad (1)$$

where c_t is the golden top class and \mathcal{E}_{c_t} and \mathcal{E}_{c_i} are the erroneous output values corresponding to the respective classes. The loss function computes the summation of differences between the value of the golden top class and the other outputs in the

corrupted network and applies a *sigmoid* function. The *golden top class* is what the fault-free DNN outputs as its classification whether or not it is correctly classified.

\mathcal{L} represents the impact of the neuron's erroneous output on the golden top class of the network. When the gradient of \mathcal{L} w.r.t. the corrupted neuron's output for one input is zero, it means that any error at this neuron's output does not change the output classification. Considering a batch of inputs, if the gradients are zero for a portion of inputs larger than a threshold, the neuron is disregarded for the vulnerability analysis. In case most of the gradients are not zero, a range for searching the vulnerability value is initialized.

Considering ϵ_k^l is a positive value for one input, in case the gradient is positive, there is a minimum value $0 < \delta_k^l < \epsilon_k^l$ for the neuron that if error δ_k^l is added to its output (by a fault at its inputs or the output value itself) the network's golden classification would change. But if the gradient is negative, then δ_k^l should be searched through the values larger than ϵ_k^l . A similar scenario is valid for negative values of ϵ_k^l .

Step2 - Neurons' Vulnerability Analysis: In this step, the vulnerability ranges of neurons under analysis are obtained. Let $R_{NV}(l, k, x) = [r_{lower}, r_{upper}]$ be a *Range of Non-vulnerable Values* for a k -th neuron at layer l with input data x . The bounds of range R for x are calculated as follows:

$$\begin{cases} r_{upper} = \min(\delta_k^l), \delta_k^l > 0, \mathcal{E}_{c_t} < \mathcal{E}_{c_i}, i \neq t \\ r_{lower} = \max(\delta_k^l), \delta_k^l < 0, \mathcal{E}_{c_t} < \mathcal{E}_{c_i}, i \neq t \end{cases} \quad (2)$$

where c_t and c_i are the golden top class and any other output class, respectively, and \mathcal{E}_{c_t} and \mathcal{E}_{c_i} are the erroneous output values corresponding to the respective classes.

Equation (2) finds the maximum negative and minimum positive values induced at the corresponding neuron that do not lead to misclassifying the input data from the golden classification. Further, a *Range of Vulnerable Values* $R_{VV}(l, k, x)$ for a k -th neuron at layer l with input data x is equal to $R_{VV} = (-\infty, r_{lower}) \cup (r_{upper}, \infty)$.

Note, the equation is applied for a single input data. In the case of a data set X containing T input data x_j the R_{NV} and R_{VV} will get refined and will be equal to intersections of their respective ranges over all inputs x_j as follows:

$$\begin{cases} R_{NV}(l, k) = \bigcap_{j=1}^T R_{NV}(l, k, x_j) \\ R_{VV}(l, k) = \bigcap_{j=1}^T R_{VV}(l, k, x_j) \end{cases} \quad (3)$$

The outcome of solving the equations for each neuron and merging the results over all inputs will be the vulnerability value ranges for each class separately, each range specifies the impact of a fault on changing the neuron value whether it influences the network classification result or not. Fig. 4 depicts different cases for vulnerability ranges over all numbers. Three vulnerability ranges are identified as follows:

- **Non-vulnerable range:** If a fault lay an effect on the neuron output in this range, no misclassification happens (hachured-green sections in Fig. 4);

- **Vulnerable range:** If a fault makes a difference at the output of the neuron in this range, the output will be misclassified (cross hatched-red sections in Fig. 4);
- **Semi-vulnerable range:** If a fault causes the neuron value to move as an amount in this range, this fault *may* cause a misclassification (dashed-grey sections in Fig. 4). Cases *d-f* in Fig. 4 happen when the portion of zero gradients in *step1* is less than the *threshold* and more than $1 - \text{threshold}$.

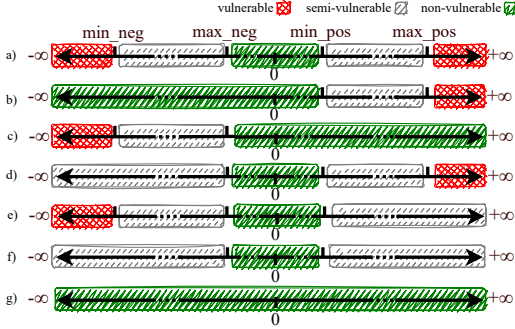


Fig. 4: Different possible cases of vulnerability ranges for each class in a neuron.

Step3 - Bitflip Mapping: In this step, DeepVigor maps the neurons’ bitflipped values over input data on the vulnerability value ranges to indicate fine-grain vulnerability factors as metrics for the DNNs’ reliability. For this purpose, the inputs used in *step2* and obtained vulnerability value ranges are fed to the network and in each bit of each neuron, bitflips are performed. In each bitflip, the difference in the new value of the target neuron is calculated and compared with the corresponding vulnerability range.

Based on the range of what the bitflip maps, the bit is considered vulnerable or non-vulnerable, respectively. By this analysis, the number of vulnerable bits of the neurons is obtained over the inputs. Hence, vulnerability factors of each layer (LVF), neuron (NVF), or bit (BVF) of the DNN can be defined as equations (4), (5), and (6), respectively. Vulnerability factors express the probability of misclassifying the network in case of the occurrence of a bitflip at the target element.

$$LVF = \frac{\#vulnerable\ bits\ in\ layer}{\#inputs \times \#layer's\ neurons \times word\ length} \times 100 \quad (4)$$

$$NVF = \frac{\#vulnerable\ bits\ in\ neuron}{\#inputs \times word\ length} \times 100 \quad (5)$$

$$BVF = \frac{\#vulnerable\ times\ for\ bit}{\#inputs} \times 100 \quad (6)$$

D. Validating DeepVigor By Fault Injection

As illustrated in Fig. 3, DeepVigor results are validated by means of FI over the input data and categorizing faults based on the vulnerability value ranges. The steps of the validation process of DeepVigor are as follows:

Step1 - Random Fault Injection: According to the adopted fault model, when one input is fed to the network, a random single bitflip is injected into a random neuron in a layer. This process is repeated several times for one input depending on the number of neurons and word length of data to reach a 95% confidence level and 1% error margin based on [28]. The required number of faults is obtained by Equation (7) where $N = word\ length \times \#layer's\ neurons$ that represents the total number of bits in the output of a layer.

$$\#layer's\ random\ faults = \frac{N}{1 + (0.01^2 \times \frac{N-1}{1.96^2 \times 0.5^2})} \quad (7)$$

Step2 - Fault Categorization: Once a fault is injected, a difference is produced in the output of the neuron in comparison with the golden model. In this step, the produced difference by a fault at the neuron’s output is compared with the obtained vulnerability ranges, and faults are categorized as:

- **Non-critical fault:** The produced difference is in the non-vulnerable range.
- **Critical fault:** The produced difference is in the vulnerable range.

Step3 - Validating DeepVigor: To validate DeepVigor by FI, injected faults are propagated to the output and the network classification output is examined. The accuracy of the method is defined based on the two metrics as follows:

- **True non-critical faults:** Percentage of faults that are categorized as non-critical and do not change the classification at the output;
- **True critical faults:** Percentage of faults that are categorized as critical and change the classification at the output.

Another metric for validating the outputs of DeepVigor is the correlation between LVF and DNN’s accuracy loss. This correlation shows that the obtained vulnerability factors from DeepVigor represent the criticality of the components properly. Since other vulnerability factors (NVF and BVF) are calculated using the same vulnerability ranges, by validating LVF, they will be also liable metrics for the resilience analysis, consequently.

III. EXPERIMENTAL RESULTS

A. Experimental Setup

All DNNs, steps of DeepVigor, and its validation are implemented in PyTorch and run on NVIDIA 3090 GPU. To explore different DNN structures, six representative DNNs trained on three datasets are examined for the experimental results. We have experimented with two 5-layer MLPs (one with Sigmoid and one with ReLU) trained on MNIST, two LeNet-5 with 3 convolutional (CONV) layers, 2 max-pooling (POOL) layers, and 2 fully-connected (FC) layers trained on MNIST and CIFAR-10, AlexNet with 5 CONV, 3 POOLS, 2 batch normalization (BN) and 3 FCs trained on CIFAR-10, and VGG-16 with 13 CONV, 13 BNs, 5 POOLS and 2 FCs trained on CIFAR-100. The respective networks’ accuracy on the corresponding test sets are 94.64%, 90.55%, 90.4%, 66.15%, 72.73%, and 69.41%.

Data representation in this work is 32-bit floating point IEEE-754 and the *word length* in equations (4)-(7) is 32 bits. For

TABLE I: Accuracy of DeepVigor by fault injection on the same input data as the analysis.

DNN	True non-critical faults	True critical faults
MLP-sigmoid-mnist	99.985%~100%	100%
MLP-relu-mnist	99.991%~100%	100%
LeNet-mnist	99.992%~100%	100%
LeNet-cifar10	99.956%~100%	100%
AlexNet-cifar10	99.973%~100%	99.955%~100%
VGG16-cifar100	99.950%~100%	99.972%~100%

validation, a layer-wise statistical random FI is performed that satisfies a 95% confidence level and 1% error margin.

In the first step of DeepVigor ϵ_k^l is considered $-/+10000$ for range initialization and the whole search range is $[-5 \times 10^5, 5 \times 10^5]$. Finding δ_k^l in all networks by a logarithmic search is performed for negative and positive numbers separately, considering a 0.05 difference from the main value. Also, based on empirical explorations the threshold of neurons' zero-gradients for inputs is considered 98% for all experiments. Corresponding experiments are performed on the whole sets of training (as the input data set1) and test (as the input data set2) data.

B. Results and Validation

We analyze all neurons of the representative DNNs with training sets as the input data set1 by DeepVigor and obtain the vulnerability ranges. In the fault categorization step, faults are categorized into critical and non-critical classes with an accuracy close to 100%. Throughout the results from FI experiments, DeepVigor identified 66.63% to 99.42% of faults as non-critical over different layers of analyzed networks.

For validation, Table I presents the range of obtained accuracy values of the method through all layers of DNNs in terms of true non-critical and critical faults. It is observed that the accuracy of the method for categorizing non-critical faults is 99.950% to 100% and for critical faults ranging from 99.955% to 100% for the same data set.

The minor error seen in the results is due to: 1) Considered error in finding vulnerability values, 2) FI results in "NaN" values in 32-bit floating point IEEE-754 while the computations are being done on a GPU. We have categorized them as critical faults, 3) the effect of few inputs with non-zero gradients in *step1* as described in II-C.

We have also experimented with FI on the test sets (input data set2) to see the validity of the analysis on different sets reported in Table II. As it can be seen, similar high accuracy values to input data set1 are obtained.

TABLE II: Accuracy of DeepVigor by fault injection on a different input data from the analysis.

DNN	True non-critical faults	True critical faults
MLP-sig-mnist	99.985%~99.996%	99.911%~100%
MLP-relu-mnist	99.976%~100%	100%
LeNet-mnist	99.992%~100%	100%
LeNet-cifar10	99.952%~100%	99.970%~100%
AlexNet-cifar10	99.951%~99.997%	99.948%~99.998%
VGG16-cifar100	99.950%~99.983%	99.972%~99.998%

To validate the vulnerability factors, Fig. 5 illustrates the correlation between LVF and accuracy loss for a layer-wise FI on AlexNet. As demonstrated, there is a close relationship between the LVF obtained from DeepVigor and accuracy loss in FI, either

the input sets are similar or different. This correlation is observed similarly in the results for all experimented DNNs. Therefore, LVF represents the vulnerability of layers competently.

DeepVigor also provides *NVF* and *BVF* metrics as vulnerability factors for neurons and bits, respectively. As a representative example, Fig. 6 depicts *NVF* for layer *conv3* of LeNet5-mnist and LeNet5-cifar10 that the more vulnerable neurons can be identified. In this figure, the number of neurons is sorted in each DNN separately, in the ascending order of *NVF*. Also, *BVF* for all neurons in DNNs is obtained and the results show that the most significant bit of exponents is the most vulnerable bit in most cases.

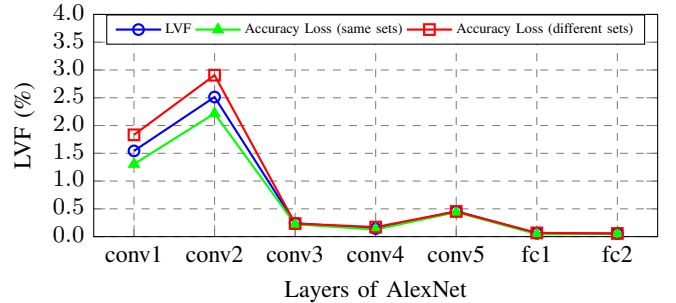


Fig. 5: Correlation between LVF and accuracy loss.

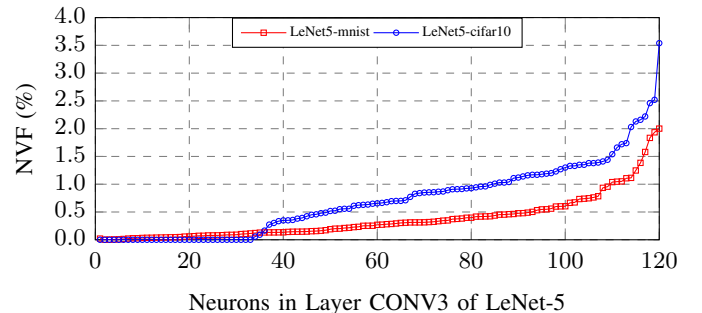


Fig. 6: NVF of neurons in CONV3 for LeNet5-mnist and LeNet5-cifar10.

C. Run-Time Analysis

DeepVigor enables a fine-grain reliability evaluation for DNNs faster than exhaustive FI. In our experiments, *step1* of DeepVigor have removed up to 48% of neurons' vulnerability analysis to be processed in *step2*. Moreover, the range initialization in *step1* has accelerated the search for finding the vulnerability values for 50% to 99% of neurons in *step2* among the DNNs. Based on our experiments, a complete vulnerability range (as in Fig. 4) for one neuron can be obtained by 9.1 times feed-forward execution per neuron on average. While an exhaustive FI experiment runs the feed-forward by the number of bits (32 in our case) per neuron. Therefore, DeepVigor requires 3.5 times fewer feed-forwards translating into a similar amount of speed-up in run-time.

The run-time of DeepVigor depends on:

- Backpropagation execution by the number of neurons *step1* (one for positive and one negative numbers per neuron);
- Feed-forward execution by the number of searches for finding a positive or negative δ_k^l per neuron, in which the

best case is 0 searches (in case of zero gradients), the moderate case is 14 searches (in case of limited range initialization), and the worst case is 22 searches;

- Vulnerability analysis of the neurons in the last layer is performed by simplified mathematics similar to Fig. 2 and requires no iterative feed-forward or searching process through a wide range of numbers;
- Bitflip mapping is merely performing a bitflip at each neuron and a comparison with the obtained vulnerability ranges.

IV. DISCUSSION

DeepVigor method is validated in the previous section, and it is shown how it can evaluate the reliability of DNNs proficiently with shorter run-times than FI. Vulnerability ranges enable a fine-grain and accurate resilience evaluation for neural networks. They are not limited to representing the single bitflip fault model and the outcome of the analysis is valid for an erroneous output for the neurons covering several fault models. This method enables an accelerator-agnostic analysis for DNNs and results can be applied to different accelerators.

The outputs of DeepVigor provide different possibilities for exploiting techniques of reliability improvement, including:

- Selective bits/neurons/layers hardening in accelerators based on the obtained BVF/NVF/LVF metrics;
- Fault-aware mapping for neurons on the processing elements of accelerators as in [21], [23];
- Applying range restriction for neurons' or layers' outputs for preventing faults propagation as in [9], [29], [30].

V. CONCLUSIONS

In this work, a novel resilience analysis method for DNNs reliability assessment named DeepVigor is proposed. The output of this method is the vulnerability value ranges for all neurons through the DNNs which result in vulnerability factors for all layers, neurons, and bits of the DNN, separately. The method is validated extensively by fault injection and its feasibility to categorize non-critical and critical faults on complex DNNs with 99.9% to 100% accuracy is demonstrated. Moreover, vulnerability factors obtained by the proposed analysis provide fine-grain criticality metrics for DNNs' components leading to different reliability improvement techniques. The DeepVigor method is very proficient in the evaluation and explanation of the reliability of DNNs with shorter run-times than fault injection.

REFERENCES

- [1] A. Bosio *et al.*, "Emerging computing devices: Challenges and opportunities for test and reliability," in *2021 IEEE ETS*. IEEE, 2021, pp. 1–10.
- [2] H. Forsberg *et al.*, "Challenges in using neural networks in safety-critical applications," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. IEEE, 2020, pp. 1–7.
- [3] Y. Ibrahim *et al.*, "Soft errors in dnn accelerators: A comprehensive review," *Microelectronics Reliability*, vol. 115, p. 113969, 2020.
- [4] M. Shafique *et al.*, "Robust machine learning systems: Challenges, current trends, perspectives, and the road ahead," *IEEE Design & Test*, vol. 37, no. 2, pp. 30–57, 2020.
- [5] W. Li *et al.*, "Soft error mitigation for deep convolution neural network on fpga accelerators," in *2020 2nd IEEE AICAS*. IEEE, 2020, pp. 1–5.
- [6] M. A. Neggaz *et al.*, "Are cnns reliable enough for critical applications? an exploratory study," *IEEE Design & Test*, vol. 37, no. 2, pp. 76–83, 2019.
- [7] A. Azizimazreah *et al.*, "Tolerating soft errors in deep learning accelerators with reliable on-chip memory designs," in *2018 IEEE International Conference on Networking, Architecture and Storage (NAS)*. IEEE, 2018, pp. 1–10.
- [8] S. Mittal, "A survey on modeling and improving reliability of dnn algorithms and accelerators," *Journal of Systems Architecture*, vol. 104, p. 101689, 2020.
- [9] Z. Chen *et al.*, "A low-cost fault corrector for deep neural networks through range restriction," in *2021 51st IEEE/IFIP DSN*. IEEE, 2021, pp. 1–13.
- [10] D. Xu *et al.*, "A hybrid computing architecture for fault-tolerant deep learning accelerators," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*. IEEE, 2020, pp. 478–485.
- [11] D. Xu *et al.*, "Reliability evaluation and analysis of fpga-based neural network acceleration system," *IEEE TVLSI*, vol. 29, no. 3, pp. 472–484, 2021.
- [12] P. M. Basso *et al.*, "Impact of tensor cores and mixed precision on the reliability of matrix multiplication in gpus," *IEEE Transactions on Nuclear Science*, vol. 67, no. 7, pp. 1560–1565, 2020.
- [13] F. F. dos Santos *et al.*, "Analyzing and increasing the reliability of convolutional neural networks on gpus," *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, 2018.
- [14] N. Khoshavi *et al.*, "Shieldenn: Online accelerated framework for fault-tolerant deep neural network architectures," in *2020 57th ACM/IEEE DAC*. IEEE, 2020, pp. 1–6.
- [15] Z. Chen *et al.*, "Binfi: An efficient fault injector for safety-critical machine learning systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–23.
- [16] M. Taheri, M. H. Ahmadilivani *et al.*, "Deepaxe: A framework for exploration of approximation and reliability trade-offs in dnn accelerators," in *2023 ISQED*. In press, 2023.
- [17] M. Taheri, M. H. Ahmadilivani *et al.*, "Appraiser: Dnn fault resilience analysis employing approximation errors," in *2023 DDECS*. In press, 2023.
- [18] A. Bosio *et al.*, "A reliability analysis of a deep neural network," in *2019 IEEE LATS*. IEEE, 2019, pp. 1–6.
- [19] A. Ruospo *et al.*, "Pros and cons of fault injection approaches for the reliability assessment of deep neural networks," in *2021 IEEE LATS*. IEEE, 2021, pp. 1–5.
- [20] A. Mahmoud *et al.*, "Hardnn: Feature map vulnerability evaluation in cnns," *arXiv preprint arXiv:2002.09786*, 2020.
- [21] C. Schorn *et al.*, "Accurate neuron resilience prediction for a flexible reliability management in neural network accelerators," in *2018 DATE*. IEEE, 2018, pp. 979–984.
- [22] C. Schorn *et al.*, "An efficient bit-flip resilience optimization method for deep neural networks," in *2019 DATE*. IEEE, 2019, pp. 1507–1512.
- [23] A. Ruospo and E. Sanchez, "On the reliability assessment of artificial neural networks running on ai-oriented mpsoes," *Applied Sciences*, vol. 11, no. 14, p. 6455, 2021.
- [24] M. Abdullah Hanif and M. Shafique, "Salvagednn: salvaging deep neural network accelerators with permanent faults through saliency-driven fault-aware mapping," *Philosophical Transactions of the Royal Society A*, vol. 378, no. 2164, 2020.
- [25] W. Choi *et al.*, "Sensitivity based error resilient techniques for energy efficient deep neural network accelerators," in *2019 DAC*, 2019, pp. 1–6.
- [26] Y. He *et al.*, "Fidelity: Efficient resilience analysis framework for deep learning accelerators," in *2020 53rd IEEE/ACM MICRO*. IEEE, 2020, pp. 270–281.
- [27] M. Bushnell and V. Agrawal, *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*. Springer Science & Business Media, 2004, vol. 17.
- [28] R. Leveugle *et al.*, "Statistical fault injection: Quantified error and confidence," in *2009 DATE*. IEEE, 2009, pp. 502–506.
- [29] L.-H. Hoang *et al.*, "Ft-clipact: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation," in *2020 DATE*. IEEE, 2020, pp. 1241–1246.
- [30] B. Ghavami *et al.*, "Fitact: Error resilient deep neural networks via fine-grained post-trainable activation functions," in *2022 DATE*. IEEE, 2022, pp. 1239–1244.