# Are CNNs Reliable Enough for Critical Applications? An Exploratory Study


| | |
|---|---|
| Journal: | *IEEE Design & Test* |
| Manuscript ID | DTSI-2019-04-0046 |
| Manuscript Type: | SI: Robust Resource-Constrained ML |
| Date Submitted by the Author: | 20-Apr-2019 |
| Complete List of Authors: | Neggaz, Mohamed; Polytechnic University Hauts-de-France, Computer Engineering<br>Alouani, Ihsen; Polytechnic University Hauts-de-France, Computer Eng<br>Niar, Smail; Polytechnic University Hauts-de-France, Computer Eng<br>Kurdahi, Fadi; University of California Irvine, |
| Keywords: | B.7.3.c Fault injection < B.7.3 Reliability and Testing < B.7 Integrated Circuits < B Hardware, B.2.3 Reliability, Testing, and Fault-Tolerance < B.2 Arithmetic and Logic Structures < B Hardware, I.2 Artificial Intelligence < I Computing Methodologies |
| | |

# Are CNNs Reliable Enough for Critical Applications? An Exploratory Study

Mohamed A. Neggaz[1], Ihsen Alouani[1], Smail Niar[1], and Fadi Kurdahi[2]

[1]Université Polytechnique Hauts-De-France, France
[2]University of California, Irvine, USA

***Abstract*— While Convolutional Neural Networks (CNNs) are going mainstream and deployed in widespread domains including safety-critical systems, the reliability issues of CNN-hosting systems are still under-explored. In this paper, we experimentally evaluate the inherent fault tolerance of CNNs through fault injection experiments. Our experiments demonstrate that quantization increases reliability. We also show the most vulnerable bits in the IEEE754 format representation. At the end, a study on the network architecture was performed to locate the most vulnerable layers. This exploratory study will make it possible to reinforce critical system reliability with a reduced cost since only the most vulnerable parts will be duplicated.**

## I. Introduction

Deep learning systems such as Convolutional Neural Networks (CNNs) have shown remarkable efficiency in dealing with a variety of complex real life problems. These techniques have been deployed in widespread domains going from mainstream application to safety-critical systems. From handwritten digit recognition, to advanced environment perception for autonomous cars, deep neural networks (DNNs) have demonstrated an effective ability to train robust feature extractors that can be successfully exploited by a classifier.

In a context of performance-driven design requirements, new hardware generations successively shrink transistors dimensions, thereby increasing circuits sensitivity to external events which can negatively affect their reliability. One of the major sources of these errors in modern embedded systems are soft errors such as Single Event Upset (SEU) and Single Event Transient (SET) that are typically caused by high energy particles striking electronic devices. These events can lead to bit flips in sequential parts and memory cells. This situation often leads to system level failures and violations of safety specifications. In safety-critical systems, incorrect values being unreliably computed represent a serious issue, as these systems must comply with strict safety standards [1].

Intentional attacks are an other potential source of faults. The widespread usage of CNNs led to the development of sophisticated attacks. Adversarial attacks are amongst these attacks. Malicious users could intentionally tamper with processed data to fool the network. While these attacks are limited to the input, they can be easily generalized to other parameters of the system such as the CNN weights [2].

Given the trend of high performance, yet sensitive hardware platforms, reliability issues of CNN-hosting systems remain an under-explored topic. In fact, since CNNs can be dedicated to safety critical applications, one cannot rely on their inherent fault tolerance aspect without deep exploration. The reliability of CNNs, especially those dedicated to safety critical applications, should be an early design stage concern, not an afterthought.

In this work, we consider random errors resulting from the environment. These errors are simulated as bit-flips. Redundancy is a common solution to reliability issues. However, it was proven that a very high number of replications is required to achieve complete fault tolerance [3].

We undertook an extensive experimental study, involving scenarios with different levels of error injection. We showed that:

- Our experimental results successfully characterizes the distribution of errors in layer-wise parameters of CNNs.
- Our approach shows that the quantization has, counter-intuitively, a positive impact on CNNs resilience to errors.
- The paper explores the impact of weights' bit significance on the error resilience of CNNs. We show that one single bit, namely the most significant bit of the exponent, needs hardening in floating-point based CNNs. Other bits are insignificant from a reliability impact perspective.
- Our approach can be used to construct a set of reliability guidelines for the deployment of CNNs in critical and aggressive environments.

This paper also presents a fault injection engine operating on CNN weights. The engine studies different reliability issues of a given trained CNN. The source is made publicly available[1].

## II. Related Work

Two types of fault injection were presented in [4]. The authors managed to achieve miss-classification after a series of careful bit-flipping. They report the loss in accuracy for

mohamedayoub.neggaz@uphf.fr
ihsen.alouani@uphf.fr
smail.niar@uphf.fr
kurdahi@uci.edu

[1] https://github.com/cypox/CNN-Fault-Injector

the target class only. In our work, we study the impact on the overall accuracy. Furthermore, the authors assumed the injections are carefully selected whereas in our experimental setup, injections are performed randomly to simulate environment faults.

In [3], a method for estimating fault tolerance in ANNs is proposed. This method exploits redundancy of hidden units to increase the network's fault tolerance. In their results, a very high number of replications (more than 7) is needed to achieve complete fault tolerance. Our study locates the most vulnerable parts to reduce this overhead when redundancy techniques are employed.

The partial fault tolerance (PFT) of ANNs during training was discussed in [5]. The authors considered replication to enhance the PFT of a network. In [6], It was shown that only 17 bit-flips are required to corrupt a network such as Alexnet. Authors carefully selected the target bits to be flipped. In this work, we focus on random error injections on different levels: data representation, position in the representation and position in the architecture.

The inherent fault tolerance of networks has also been studied in [7]. However, the authors focused on relatively small CNNs. Their methodology is based on stuck-at faults. Stuck-ats in feed-forward neural nets was also discussed in [8]. Replication was proposed as a solution to achieve fault tolerance.

The reliability of object detection networks on GPUs have been studied in [9]. Their study was based on fault injection and exploited the error leaking potential between GPU threads. Our study is platform independent and the result could be projected to other embedded systems.

To the best of our knowledge, this is the first study that explores random fault injections in CNNs considering the different quantization parameters, the different representations, the bit position and the layer position.

## III. Experimental Methodology

In this section we present our setup and methodology to evaluate the reliability. We use the same methodology from [10] with different experimental setup. When compared to the previous paper, we evaluate more variables and confirm the obtained results on other networks.

### A. Methodology

Without considering physical damage, soft errors compromises system functionality by causing bit-flips in memory or in computational elements. Since memory errors are more critical and durable, we only focus on bit-flips in memory. In most machine learning accelerator designs, two memories are present: 1) the weights memory ($\mathcal{M}_w$) which stores trained network parameters and 2) intermediate output memory ($\mathcal{M}_i$) which stores the output of hidden layers.

$\mathcal{M}_i$ receives new values for each input. A bit-flip in this memory will only affects the current run, and, only if it occurs before the subsequent layer starts processing. This is similar to errors in computational parts which we will not study. On the other hand, a bit-flip in $\mathcal{M}_w$ will remain active until a new network is deployed. We focus on this kind of errors.

To reproduce this behaviour, we simulate a soft error in $\mathcal{M}_w$ by a number of bit-flips in a random weight once the network is trained. Multiple studies are conducted based on this simulation hypothesis. In each study we evaluate a robustness variable of a CNN based system. The position of the flip is decided by the study and the evaluated variable.

*1) Networks:* CNNs can perform a variety of tasks. Whether it is for images, voice, text or other input types, classification is the most common task performed by CNNs. Other tasks, such as detection, uses a classification subnetwork. In the experimental setup we propose, we only consider classification networks. Consequently, the study can be projected to other variants.

*2) Dataset:* Measuring a CNN's accuracy requires a labeled testset. Since testsets are usually not labeled, we use the validation set of ImageNet as used in the challenge. The set contains $50000$ image with the correspondent class of each image. The set contained $1000$ classes.

*3) Data Representation:* We consider two data representations:

- IEEE-754's 32-bit float: This is the standard representation format for floating point format. It is the dominant representation in CPU and GPU architectures. Many GPUs are optimized to deal with floating point multiplications. For simplicity we refer to this representation as $\mathcal{F}$ in the rest of the paper.
- X-bit fixed point: We used the format from [11]. Trading accuracy for high performance by using low bit-widths is a common practice in CNN acceleration. This representation uses two parameters: bit-width and fractional length. Negative fractional lengths can be used to represent powers of two. This representation is referred to as $\mathcal{Q}$ (for quantized) in the rest of the paper.

*4) Injection Algorithm:* Based on the fault-injection model in [10], we create a fault-injection engine. The engine takes a trained network, a dataset and a test type. The test type dictates the execution flow and the parameters to vary during the test. Multiple test types are developed, more details are provided later this section. Depending on the selected test type, a series of bit-flips are performed in the network's weights. After each test, the engine reports the measured accuracy on the dataset after the injection.

We consider three test types: *full-network*, *index-wise* and *layer-wise* tests.

- *Full*-network injection: the engine generates a list of errors that are identified by their layer and the position in the layer. The engine incrementally injects errors in the network. After each injection, we measure the accuracy on the whole dataset. As a result, we aim to compare the two data representations in terms of inherent resilience. This comparison is useful to decide which data representation is more suitable when faults are present.
- *Indexed* injection: in this test, the generated errors are injected in a fixed bit significance. The engine then

loops over every possible position from the least to the most significant bit[2]. The result of this test type extends on the result of the *full* test. After comparing the two representation, we use this study to explain the difference, if any. Furthermore, this helps localizing the most vulnerable bits to protect.

- *Layer*-wise injection: in this test, errors are generated in the same layer with different positions. This test is repeated for each layer whilst reporting the accuracy after each run. The number of errors injected is proportional to the number of parameters of each layer. This is similar to the real world where the soft-error rate is proportional to the surface of the chip. This study allows us to understand the inherent tolerance of CNNs layers. Finding the most vulnerable layers will assist in creating comprehensive reliability enhancement strategies.

These tests are repeated 60 times. In each run, the engine generates a new set of errors and the injection of the generated errors is performed each run. We then present the mean of the 60 runs as well as the maximum, minimum and the standard deviation of the test.

A single soft error can cause multiple bit-flips. Furthermore, memory errors are cumulative. We fix the number of errors to be injected when varying the index of the bit-flip to 50. For layers, we inject errors proportional to the number of parameters with at least 1 injected error[3]. An extensive study, with variable number of errors, is possible, however, the same tendency reappears.

### B. Experimental Setup

The engine was developed on python. For CNN inference, we used the framework Caffe. The code is made publicly available[4]. As part of our study, we perform injections on quantized (low-precision) CNNs. The weights were obtained using Ristretto.

The experiments were performed on an Nvidia Quadro P5000 GPU with an Intel(R) Xeon(R) W-2123 CPU with 3.60GHz frequency.

*1) CNNs:* We used four network architectures: GoogleNet, Alexnet, VGG16 and SqueezeNet. These networks were selected for their wide usage, diversity, various sizes and high accuracy. Their convolutional layers are widely reproduced as feature extractor in other models. This facilitates generalizing the obtained results to other networks.

For the 32-bit floating point represented weights, we used trained instances from Caffe's Model Zoo[5]. We used Ristretto to quantize and fine-tune the four networks into 8-bit fixed point networks without huge loss in accuracy.

[2]In the case of a trained network represented as 32-bit floating point, the engine loops over the 32 bit positions.

[3]Scales with the size of the target layer.

[4]https://github.com/cypox/CNN-Fault-Injector

[5]Publicly available on: https://github.com/BVLC/caffe/wiki/Model-Zoo

## IV. Experimental Results

The results we collected from the engine are presented in this section. For each test type (*full*, *layer* and *index*) we show the obtained results separately.

### A. Impact of Data Representation and Quantization

The results were obtained on weights represented as 32-bit floating point. We present a comparison between the impact of different data representations on the accuracy of the different networks.

Figure 1 illustrates the result of comparing the two representations. The $\mathcal{Q}$ representation is clearly more resilient than it's counterpart. This tendency is present for the four networks with different rates.

| Network | Alexnet | VGG16 | Googlenet | Squeezenet |
|---|---|---|---|---|
| Weights ($\times 10^6$) | 60.97 | 138.36 | 7 | 1.25 |

TABLE I

The decrease in accuracy in VGG16 and Alexnet is not as fast as the decrease for the same number of errors in Googlenet and Squeezenet. The main reason for this phenomenon is the number of weights as shown in Table I. The same number of errors have less impact if the number of weights is important.

### B. Significance of Bits

To further explore this decrease in accuracy, we investigate the individual impact of the bit position. The injections are performed at the same position on the four networks each run. The only difference being the index of the bit-flip on the binary representation of the weight. We performed this study only on the $\mathcal{F}$ representation. The $\mathcal{Q}$ representation in invulnerable to bit-flips as shown in the previous results in Figure 1.

In Figure 2, the four networks have the same tendency. Unless bits are injected in the exponent's most significant bit, almost no impact on the accuracy is perceived.

### C. Layer Tolerance

The impact of injected faults may depend on its location within the network architecture. This section explores the layers tolerance aspect. Similar to Section IV-B, we isolate the target layer in the fault injection process. This isolation allows to track the individual impact of the chosen layer on the overall accuracy.

Googlenet and Squeezenet have a special architecture. They are built on top of two modules, *inception* for the former and *fire* modules for the latter. These modules regroup a set of convolutional layers working in parallel on the same input. The output of the module is obtained by concatenating the outputs of each execution branch. For clarity, we reduced the individual layers into the corresponding modules. For each module, we take the average accuracy of its individual layers.

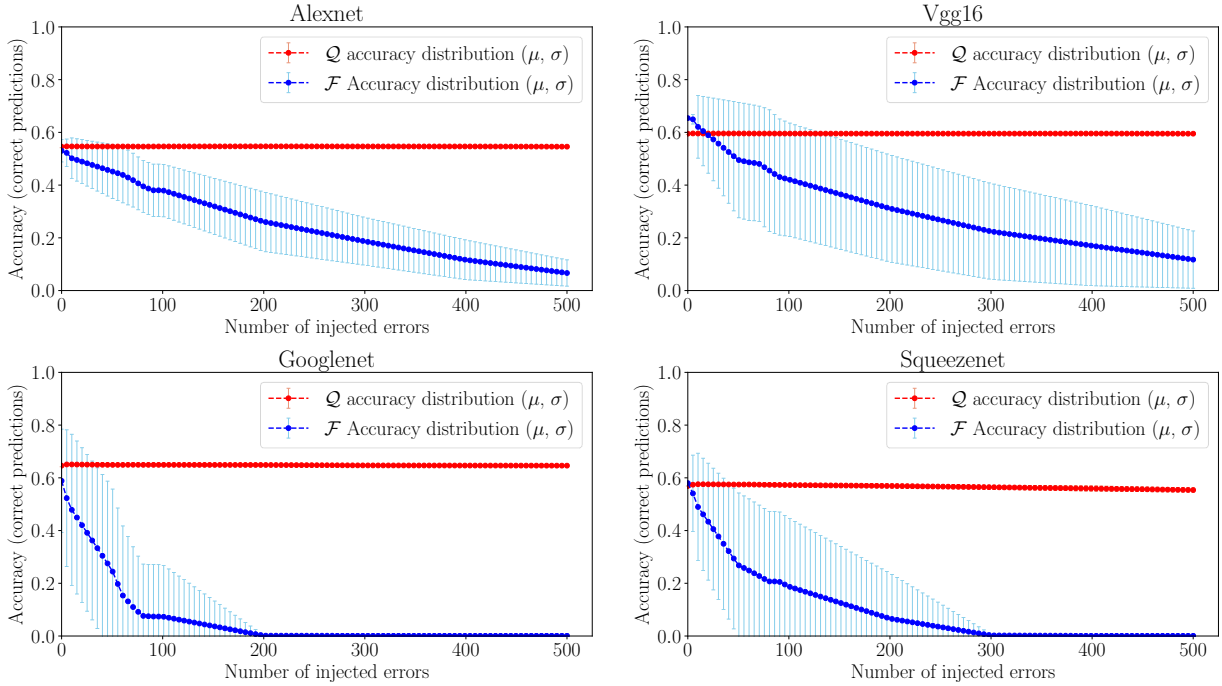


Fig. 1: Comparison between the 8-bit fixed point representation ($\mathcal{Q}$) of weights and the 32-bit IEEE-754 representation ($\mathcal{F}$). The results of different runs are presented as the mean and the standard deviation of the top-1 accuracy.

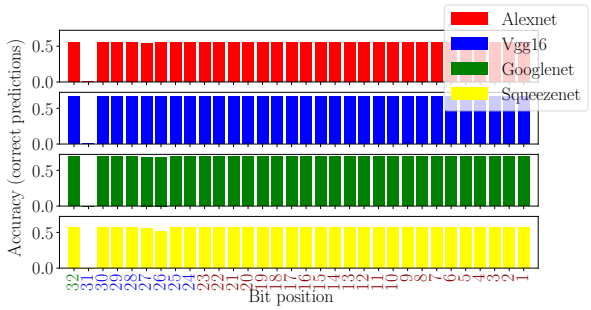


Fig. 2: Position of bit-flips in the value representation and its impact on the accuracy. In the X-axis, red labels represent the mantissa, blue labels represent the exponent and the sign bit is in green. Position 32 is the most significant bit in the representation (sign bit).

Figure 3 presents the results of this study. The four networks tend to lose more accuracy when injections occur in advanced layers. This is correlated to our previous results in [10]. While CNNs have a sequential structure, error propagation is not problematic in CNNs. Errors in early layers have, in general, less impact on the accuracy. This shows the implicit characteristic of CNNs to maintain a sane behaviour when incorrect values are forwarded. This is explained by the implicit redundancy in CNN weights. After training, many weight clusters are repeated. A small number of errors if it occurs in the first layers. Techniques such as pruning can greatly affects this study. Pruning explores weight redundancy to reduce computations. While it achieves high throughput with acceptable accuracy, reliability can be greatly compromised [12]. This trade-off should be considered to evaluate CNN acceleration in aggressive environments.

It is worth to mention that, although the mean value is at a comfortable accuracy, the minimum accuracy reported is almost always $\approx 0\%$[6]. This means that in some runs, the injected errors were able to fully compromise the network. As rare as it could be, anticipating these cases by studying the network should precede any deployment. Also, the fact that a few number of errors can damage a network this far is an other motivation to deeply study the impact of faults.

## V. Discussion

### A. Floating Point and Fixed Point

In contradiction to common belief, the $\mathcal{F}$ representation is more vulnerable to injections even though it has more bits. The individual impact of a bit in a short representation (8-bit fixed point) is greater than its counter part in the $\mathcal{F}$ representation. However, the divergence from the correct value is greater in the later due to the nature of the representation. The exponent is not represented in the fixed point representation. A bit-flip in any position is similar to adding or subtracting a power of 2. Since all weights range from $-1$ to $+1$ [2], the value added or subtracted is minuscule. Hence, it's impact can be logically masked.

[6]The worst case is 0.001 which is equal to randomly guessing the class over the 1000 possibilities.

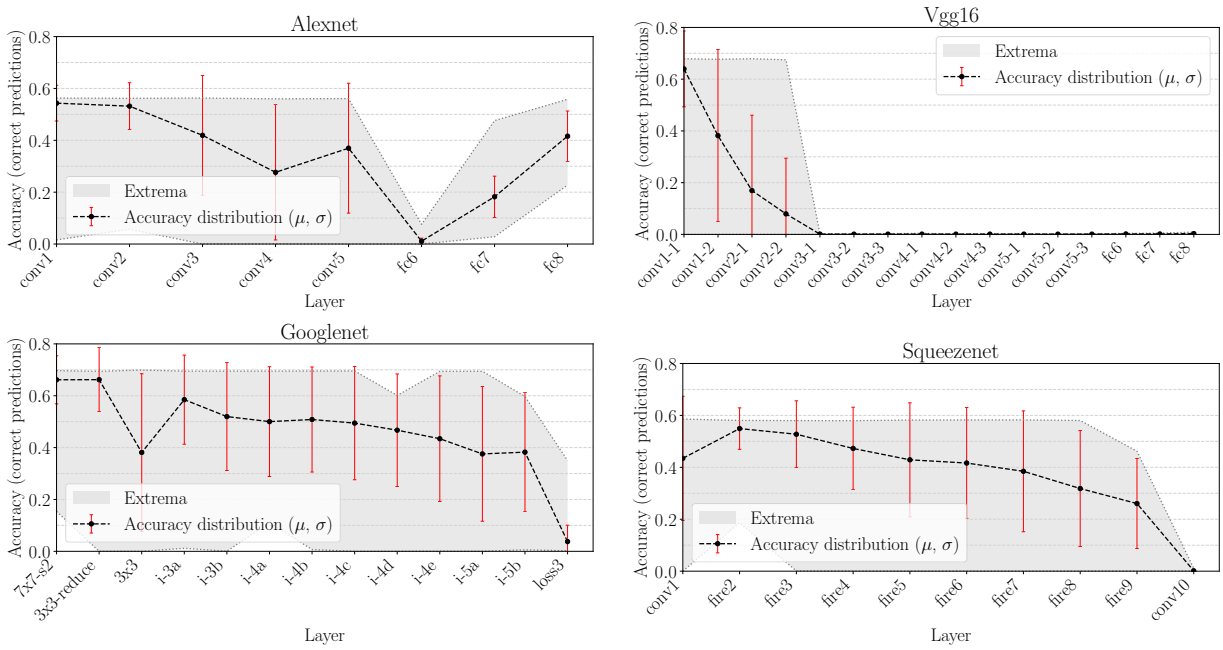


Fig. 3: Impact of faults layer-wise for the four networks. Each series is represented as the mean top-1 accuracy (black dots), the standard deviation (red error-bars) and the minimum/maximum (gray fill).

### B. Bit Position

In the $\mathcal{F}$ representation, not all the bits in the exponent are important. The impact of bits is not linear to the bit position but constant except for the most-significant bit as could be seen in Figure 2. This is partly due to the distribution of CNN weights. Weights range from $-1$ to $+1$. High values in the exponent are always accompanied with a negative exponent sign. Having a bit-flip will decrease the value even more, making it close to 0, which is not a big difference considering the range of weights. The only important change is the most-significant bit of the exponent. If changed from 0 to 1, the new value will be orders of magnitude higher than the others. Combined with the maximum pooling, this leads to catastrophic results in the subsequent layers.

### C. Layer Index

Although the general tendency shows that the last layers are more vulnerable, no conclusions could be drawn. In other words, vulnerable layers of a new CNN can not be located without a simulation. A fault injection engine such as the one we presented in this paper should be used to evaluate the individual layer vulnerability. This is an extension to Netscope [7], a neural network visualizer and analyzer. We introduced the resilience parameter which is computed from the accuracy reported by the fault-injection engine. The modified version allows to extract the most vulnerable layers visually. An example output of the analyzer is available with the engine source code[8].

[7] https://github.com/ethereon/netscope
[8] https://www.github.com/cypox/CNN-Fault-Injector

### D. Suggestions and Guidelines

The first study shows a bit difference between the storage formats. Floating point representation should be used with caution in critical systems. The fixed point representation would result in less memory and computation[9] overhead with higher reliability. System designers should consider this aspect when dealing with aggressive environments.

It was shown that the most significant bit in the exponent is the vulnerable part of the floating point representation. Using this conclusion, the overhead of redundancy techniques could be reduced. Techniques such as TMR will Replicate the whole number three times. This will triple the memory requirements of Googlenet for instance, whose number of weights is $6,996,452$, thereby adding 427 megabytes of required storage. If applied exclusively to the vulnerable bits, only 13 megabytes would be necessary. This result can also be extended to the *layer* test with variable degrees of protection depending on the resilience of each layer as outputted by the injection engine.

## VI. Conclusion

An extensive analysis of fault tolerance in convolutional neural networks was proposed. It was shown that quantization have a positive impact on reliability. The issue with non-quantized networks is the data representation. For the IEEE-754 format, the most significant bit of the exponent is crucial to reliability for CNNs. A layer-wise analysis is then performed. For complex networks, the reliability of the

[9] Fixed point representation is usually coupled with low precision arithmetic. This allows for better efficiency with comparable accuracy.

overall network should be studied based on the architecture. The framework we developed to analyse reliability is made publicly available. This study solves the overhead of reliability techniques by reducing the number of replicated elements.

## REFERENCES


[1] ISO, "Road vehicles – Functional safety," 2011.

[2] Q. Liu, T. Liu, Z. Liu, Y. Wang, Y. Jin, and W. Wen, "Security analysis and enhancement of model compressed deep learning systems under adversarial attacks," in *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*, ser. ASPDAC '18. Piscataway, NJ, USA: IEEE Press, 2018, pp. 721–726. [Online]. Available: http://dl.acm.org/citation.cfm?id=3201607.3201772

[3] D. S. Phatak and I. Koren, "Fault tolerance of feedforward neural nets for classification tasks," in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, vol. 2, June 1992, pp. 386–391 vol.2.

[4] Y. Liu, L. Wei, B. Luo, and Q. Xu, "Fault injection attack on deep neural network," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2017, pp. 131–138.

[5] E. B. Tchernev, R. G. Mulvaney, and D. S. Phatak, "Investigating the fault tolerance of neural networks," *Neural Computation*, vol. 17, no. 7, pp. 1646–1664, July 2005.

[6] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network withprogressive bit search," *arXiv preprint arXiv:1903.12269*, 2019.

[7] P. W. Protzel, D. L. Palumbo, and M. K. Arras, "Performance and fault-tolerance of neural networks for optimization," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 600–614, July 1993.

[8] D. S. Phatak and I. Koren, "Complete and partial fault tolerance of feedforward neural nets," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 446–456, March 1995.

[9] F. dos Santos, P. Foletto 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. PP, pp. 1–15, 11 2018.

[10] M. A. Neggaz, I. Alouani, P. R. Lorenzo, and S. Niar, "A reliability study on cnns for critical embedded systems," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*. IEEE, 2018, pp. 476–479.

[11] M. Courbariaux, Y. Bengio, and J.-P. David, "Training deep neural networks with low precision multiplications," *arXiv preprint arXiv:1412.7024*, 2014.

[12] B. E. Segee and M. J. Carter, "Fault tolerance of pruned multilayer networks," in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, vol. ii, July 1991, pp. 447–452 vol.2.