Bi-LORD: Bit-Wise Low-Cost Real Numbers Dependability Assessment in AI Applications

Bi-LORD: Bit-wise LOw-cost Real numbers Dependability assessment in AI applications

Julia Azziz
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
jazziz@fing.edu.uy

Annachiara Ruospo
Dipartimento di Automatica e Informatica
Politecnico di Torino
Torino, Italia
annachiara.ruospo@polito.it

Ernesto Sánchez
Dipartimento di Automatica e Informatica
Politecnico di Torino
Torino, Italia
ernesto.sanchez@polito.it

Julio Pérez Acle
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
julio@fing.edu.uy

***Abstract*—As deep neural networks (DNNs) are increasingly deployed in critical applications, ensuring their reliability under hardware faults becomes a key concern. The size of the models and the number of weights in a typical network continually increases. As a consequence, there is a strong pressure to reduce the overhead of adding error correction capabilities to the stored data. This approach involves identifying which bit positions are most critical for the dependability of the DNN. This task, particularly in some novel arithmetic formats like Posit, may require extensive fault injection experiments with high computational costs to assess accuracy degradation.**

In this paper, the impact of permanent faults on network weights is analyzed. A low-cost analysis based on the faulty/golden ratio of the affected parameter is presented as a tool to predict fault criticality according to the faulty bit position. To validate this methodology, a set of fault injection campaigns was conducted on a typical DNN application for image classification implemented resorting to two very common 32-bit data formats: Posit and Floating Point. The observed accuracy degradation produced by faults injected in different bit positions suggests that the proposed low-cost analysis is a powerful tool to predict bit criticality.

***Index Terms*—fault tolerance, deep neural network (DNN), posit number format, fault injection**

I. Introduction

Deep neural networks (DNNs) have become central to numerous applications, ranging from image recognition to autonomous systems. As these models are increasingly deployed in real-time, safety-critical environments such as autonomous vehicles, medical devices and aerospace systems, ensuring their robustness and fault tolerance becomes of paramount importance. One of the main challenges in these environments is hardware-induced faults, such as bit flips, which can occur due to cosmic rays, radiation, or aging components. Bit flips in memory or processing units can corrupt the model weights or activations, potentially causing significant degradation in the network's performance or leading to critical failures.

To mitigate the effects of hardware faults, various strategies have been explored, including fault-tolerant architectures, error correction codes and redundancy in computation. Traditionally, DNNs rely on floating-point arithmetic, which provides a balance between precision and dynamic range. As models grow in depth and the number of weights in a typical network increases, however, the sheer volume of data that needs to be stored to deploy a DNN has become the main challenge and disadvantage of floating point representations.

In order to keep both the storage needed for weights and the area required for the arithmetic units within manageable limits, non-standard, reduced-word length floating point representations [1]–[4] have become of common use in the industry. Furthermore, some novel arithmetic formats have been proposed as alternative solutions.

One such format is Posit [5], [6] which employs variable-length coding for part of the exponent. This approach allows for more bits to be allocated to the mantissa when the coded value is close to one, resulting in improved resolution compared to traditional floating-point formats of the same word length. On the other side, this increased flexibility in the length of the fields length makes more difficult to predict the effect that a fault may produce. In floating-point formats, fixed field lengths make the impact of a fault more predictable. In contrast, Posit's variable-length encoding complicates this predictability because faults in a given bit position could affect different fields.

This paper analyzes the impact of permanent bit-flip faults on the parameters of a typical neural network for image classification, examining both the error introduced in the affected parameter values and the resulting modifications in the network outputs across the different bit positions.

II. Background and Related work

In the posit format with precision n, each number consists of four fields: *sign*, *regime*, *exponent* and *fraction*. The posit value p is computed from the corresponding field values s, r, e and f as follows:

$$p = (1+f)2^{(4r+e)} \quad \text{if } s = 0 \tag{1}$$

$$p = (-2+f)2^{-(4r+e+1)} \quad \text{if } s = 1 \tag{2}$$

The *regime* field is a variable-length code represented by a sequence of k identical bits, followed by a complementary bit. The regime value, r, is set to $-k$ when the sequence bits are 0, and $k-1$ when the bits are 1.

978-1-6654-7763-5/25/$31.00 ©2025 IEEE

The *exponent* field is 2 bits long and its value is an unsigned integer between 0 and 3, while the *fraction* value is an unsigned value between 0 and 1. The regime field works as a coarse-grained exponent whose effect is modulated by the sign field: large positive *r* values yield large positive numbers if $s = 0$, or small negative numbers if $s = 1$.

As the length of the *regime* field grows, it reduces the length of the *fraction* field first, and then the *exponent* field. In extreme cases, both the *fraction* and *exponent* fields may be fully truncated.

The posit standard for precision n also defines a *quire* format, which is a fixed-point, 2's complement representation with $16n$ precision. This format allows for the exact accumulation of products of n-precision posit values.

Several recent studies have examined the effects of faults on data encoded in Posit format and compared these effects to other numeric representations. In [7], single bit-flip faults were injected into sample subsets of various scientific datasets. Faults were applied to both the original dataset values in 32-bit floating-point representation and the same values converted to Posit format. The resulting relative and absolute errors were then compared.

Another study [8] investigates the impact of single-event upsets (SEU) by injecting faults at each bit position of the 2^{32} possible values in a 32-bit number and calculating the relative error. For each bit position, the mean relative error is computed across all 2^{32} possible data values for both IEEE-754 and Posit representations. The study also considers double-bit upsets by randomly selecting a second bit for injection. Additionally, results from experiments that inject faults at the inputs of two machine learning systems indicate a superior performance of the Posit representation.

In [9] the authors explore the relationship between the criticality of a bit position and the average change in value caused by a bit flip, both for Posit and Floating point formats. This average change is computed by adopting the Average Bit Flip Distance (ABFD), a metric computing the average distance that a fault in a specific bit position produces between the golden and the faulty values. In [9], the effectiveness of the metric was assessed for different data types (32-bit floating point, 32-bit posit, 16-bit posit and 8-bit integer) and demonstrated a good correlation between the faults' impact and the CNN output degradation. It is worth noting that this outcome was already experimentally demonstrated in [10] for 32-bit floating point numbers, where the ABFD metric was used to configure a data-aware statistical fault injection analysis.

Several works by Sonza et al. [11]–[13] present fault injection campaigns in GPUs and multiplier models operating in both floating-point and Posit formats. In most cases, a greater number of faults resulted in noticeable error values in Posit cores; however, the average error observed in floating-point cores was higher.

In this paper, we investigate the effectiveness of a new metric to analyze the impact of permanent bit-flips in neural network parameters. This analysis is based on the use of the ratio between the faulty and the golden values rather than the relative error used in prior studies, and employs a more detailed box-plot analysis instead of relying on mean values alone. To evaluate this concept, we conduct a fault injection campaign on a typical neural network that aims to expose a correlation between the accuracy degradation resulting from faults in the different bit positions and the results produced by the proposed metric.

These results suggest that the proposed method can be used to obtain an early estimation of the criticality of different bit positions in the stored network parameters, with very low requirements in terms of computational costs.

III. Methodology

With the goal of analyzing the resilience of deep neural networks in the presence of faults, fault injection experiments targeting biases and synaptic weights were conducted. In all the experiments, permanent single bit-flip faults were injected.

Section III-A discusses several analyses that can be done considering only the values of the set of parameters and the effect of the injected fault in the affected parameter. These static fault injection experiments provide useful insight about which are the most critical faults. Moreover, they have very low computation costs, as they do not require the execution of the network inference to obtain the outputs.

Next, in section III-B a fault injection campaign analysing the fault effect in the outputs is presented.

A. Static analysis: parameters set characterization and fault injection effect on parameter value

Before introducing faults, we first characterize the set of parameters of the network under analysis. This characterization includes an examination of the distribution of parameter values. Additionally, given that Posit fields have variable lengths, we present histogram graphs illustrating both the distribution of field sizes and the utilization of each bit position, categorizing them as sign, regime, exponent, or fraction. Subsequently, we inject permanent single bit-flip faults into the parameter values and analyze the impact of these faults on said values.

Previous research has typically focused on mean absolute or relative error. However, since the weights serve as multiplicative factors, we propose the use of the faulty-to-golden value ratio as a more effective indicator than the relative error. As will be shown, when a fault results in a faulty value that is smaller than the golden one, the relative error tends to cluster around 1, which may not accurately reflect the extent of the fault.

A more detailed analysis is also presented using box plots instead of mean error values, as commonly used in previous studies. The box plots provide clearer insights into extreme error values, which are the most likely to propagate through the network and alter its predictions.

The expected impact depends on the bit position affected by the fault, and more precisely on which field was affected. Faults affecting the fraction part have the lesser relative impact.

As shown in section IV, the most dramatic effects are produced when affecting the sign, regime and exponent fields. The characterization of the set of parameters presented above allows us to identify which of the bit positions are effectively carrying the more critical regime and exponent information.

B. Fault effect on network output

To verify whether the bits identified as most critical in the static analysis are indeed more likely to affect the network's output, a fault injection campaign was conducted. As in the previous analysis, single-bit flip faults are injected in the parameter values, but in this case network inference is computed, and output values are compared with the golden ones.

An exhaustive fault injection campaign would require running an experiment to inject a fault in each bit position (32 or 16, depending on the format) of each parameter of the set of parameters and computing the network output for each of the input images of the test set. This cannot be performed in a reasonable time with the computing resources available to the authors. To reduce computation time, faults will be injected only in a random subset of the parameter set, and only a random subset of the test set will be used to evaluate the accuracy of the network for each injected fault.

The resulting accuracy degradation is presented across each bit position, combined with the faulty/golden ratio obtained in the static analysis to assess the relationship between both metrics.

IV. Experiments and results

A. Experimental setup and training results

For the image classification task, a LeNet-5 [14] architecture was tested over the greyscale image dataset MNIST [15]. To conduct our experiments, we based our framework on the DeepPensieve library [16], which uses SoftPosit [17] for the Posit arithmetic backend.

The network was trained using Float32, Posit32 and Posit16 formats. Training was conducted for 30 epochs, using the Adam optimizer with a learning rate of 0.001 and a batch size of 128, similar to the approach taken by DeepPensieve [16]. This model has a total of 61,706 parameters, which include all trainable weights and biases across its five layers.

Table I presents the test accuracy for each trained model, using both top-1 and top-5 accuracy metrics. Results show that accuracy remains largely consistent across all three formats, with only marginal variations observed.

TABLE I
Test accuracy for each trained model.

Training type	Top-1 accuracy (%)	Top-5 accuracy (%)
Float32	90.2	99.9
Posit32	90.1	99.9
Posit16	90.4	99.8

B. Parameter set characterization

As shown in Figure 1, parameter values follow a normal distribution around zero, with all absolute values lower than one.

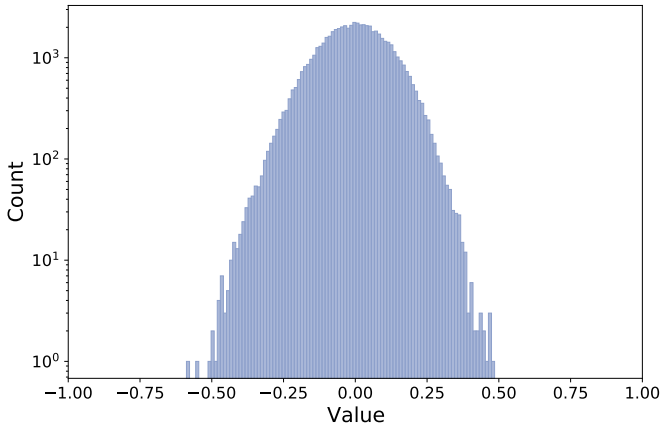

Fig. 1. Histogram of parameters values for a network trained using Posit32.

Figure 2 shows how the variable length of the regime field in Posit format affects the bits usage. In Posit32, for most of the weights, the regime field uses two or three bits, the exponent field uses bits 28 and 27 or 27 and 26, and the fraction starts in bit 26 or 25.

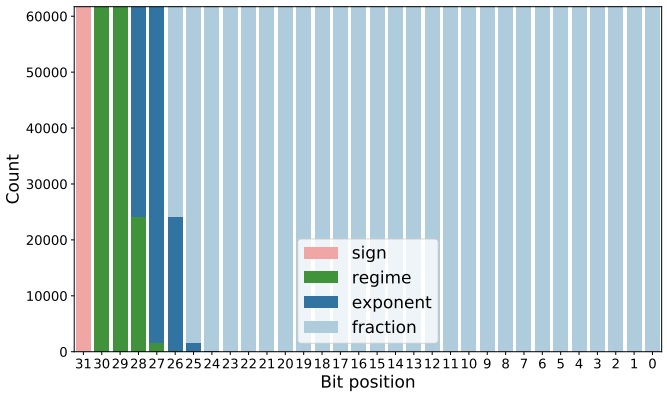

Fig. 2. Bit usage in Posit32 format for all LeNet5 parameters.

C. Fault effect on weight magnitude

A single bit-flip fault was injected on each bit of each of the network parameters, and its effect on the parameter value was analyzed.

Figure 3 shows box-plot for both the relative error and the proposed faulty/golden ratio across the analyzed formats. For the float32 format, the most significant bit of the exponent produces the most pronounced impact. Since all weight magnitudes are less than one, the exponent is always negative and this bit is always 0, according to the offset binary representation. When bit 30 flips from 0 to 1, the parameter value is multiplied by 2^{2^7}, approximately 10^{38}. This dramatic

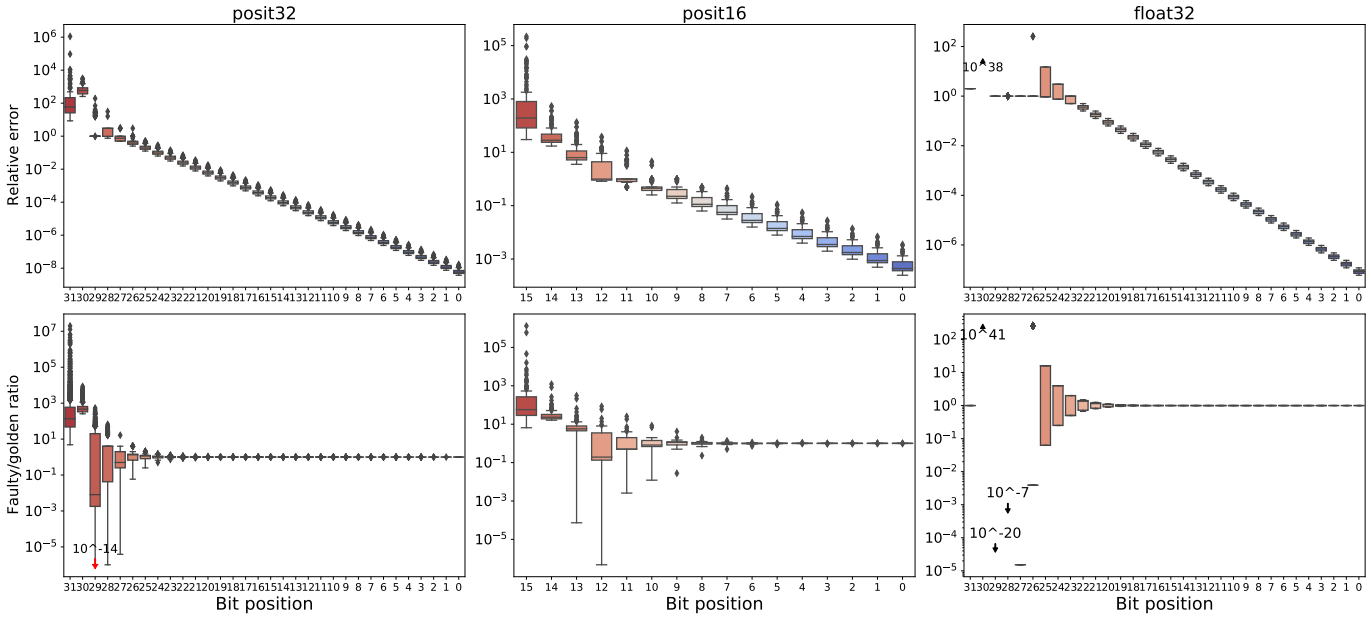

Fig. 3. Box-plots showing relative error (up) and faulty/golden ratio (down) values for posit32, posit16 and float32 formats.

effect is clearly visible in both plots. Also, as all parameters are lower than one, all exponent values differ at least in two bits from the all-ones value reserved for not a number (NaN) or infinity values. So, no single fault will result in these special values.

Figure 3 also highlights that, for float32, faults injected into bits 29–26 lead to a significant reduction in parameter values, with minimal or no dispersion. This limited dispersion is due to the parameter value distribution, where bits 29–26 are consistently 1, except for occasional cases in bit 26. For instance, when bit 29 is flipped, the weight value is always divided by a factor of 2^{2^6}, approximately 10^{20}. However, in these cases, the relative error remains close to one, rendering this effect almost entirely undetectable as can be seen in Figure 3. From bits 25 onward, which represent the least significant bits of the exponent field and the entire fraction field, both the relative error and faulty/golden ratio decrease rapidly.

The plots for Posit formats reveal significantly greater dispersion in the effects of faults compared to the floating point format. The most impactful bit positions are those corresponding to the sign and regime fields. While faults in the exponent field cause smaller yet important modifications, fault impact diminishes rapidly as we move to the least significant bits of the fraction field.

The increased significance of the sign bit in Posit formats, compared to the floating point format, is expected. A flip in the sign bit not only changes the sign of the value but also complements the sign of the exponent of two in the formula that determines the Posit value. Similar to floating point, faults in bits 29 and 28 of the Posit format can lead to substantial perturbations, resulting in faulty/golden ratios below one. However, these effects remain undetected in the relative error box plots. According to these observations, and considering that most parameters are weights used as multiplicative factors, any fault affecting bits 30–26 in the float32 format or bits 31–29 in the posit32 format will cause dramatic changes to the associated product. In some cases, such as bit 30 in float32, the product becomes significantly larger, while in others, such as bit 29 in float32, the product becomes much smaller than expected. To understand the broader implications of these extremes, we conducted a series of experiments to assess their impact on the network's output and whether both extremes lead to similar effects.

D. Fault effect on network output

The impact of single bit-flip faults on the network's output was evaluated using the top-1 accuracy metric. In each experiment, a single fault was injected into a randomly selected parameter, and the faulty network was evaluated over a test set to assess the overall accuracy degradation. Due to the high computational cost of running inference on the entire test set, particularly for the Posit format, a subset of 500 images was used, drawn evenly from all ten classes.

These experiments were conducted over the 32-bit floating point and Posit formats, injecting a random subset of faults on all bit positions. For the floating point network, we injected 600 faults per bit position, totaling 19,200 experiments. For the Posit format, 500 faults were injected for bits 31-26 and 200 for all lower bits, totaling 8,200 experiments.

Figure 4 combines the faulty/golden ratio box-plots with the mean faulty/golden accuracy values for each bit position. The results indicate a clear trend: large faulty/golden ratios correspond to significant accuracy degradation, which was expected. This is particularly evident for bit 30 in the floating

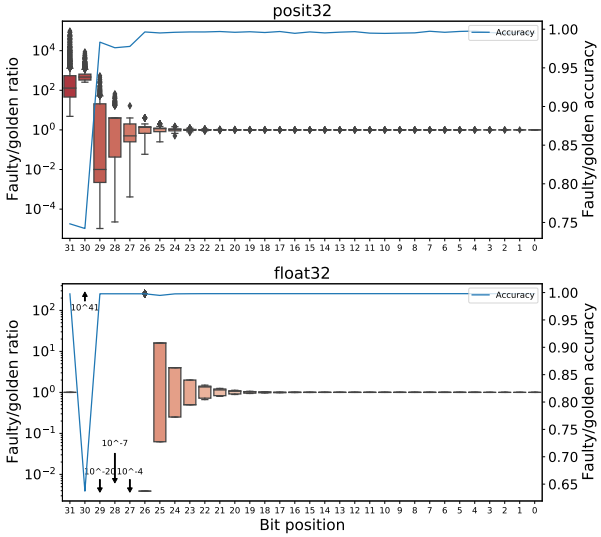

Fig. 4. Faulty/golden ratio box-plot and mean faulty/golden accuracy across all bit positions for 32-bit formats.

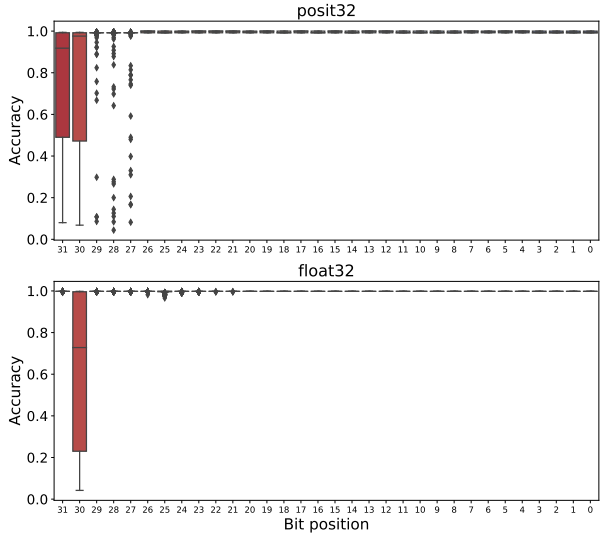

Fig. 5. Accuracy box-plot across all bit positions for 32-bit formats.

point format and for bits 31 and 30 in the Posit format, where the impact on accuracy is especially pronounced.

On the other hand, faults in bits 29-26 of float32 format that result in very small values for the affected parameter do not appear to cause significant accuracy degradation. In these cases, even when the affected parameter approaches zero, virtually suppressing its contribution to the sum of the products, the network output remains largely unaffected in most cases.

To provide further insight, Figure 5 presents box-plots showing the distribution of accuracy values in the faulty networks. The float32 graph reveals that at least half of the faults affecting bit 30 result in significant accuracy degradation. Except for a small number of outliers that slightly alter the accuracy, faults affecting other bits have little to no impact. In the posit32 format, bits 31 and 30 exhibit a similar behavior to bit 31 in float32. Faults in bits 29-27 produce no degradation, with the exception of a few outliers. The number of outliers in the Posit format is larger than in floating point, and their impact on accuracy is also more pronounced. In both formats, faults in the least significant bits (26-0 in posit32, 20-0 in float32) do not lead to any significant reduction in accuracy.

To further illustrate which are the most critical cases in Posit format, Figure 6 presents a scatter plot representing each of the injected faults in the accuracy-ratio plane. The plot shows that a significant amount of the faults with ratio values greater than one produce important accuracy degradation, corresponding mainly to faults injected in bits 30 and 31. Unlike the float32 format case, in posit32 accuracy degradation also shows for several faults that reduce the parameter value (ratio lower than one). All these cases correspond to network biases, not weights.

An additional fault injection campaign was performed choosing the 1,000 faults that produce the most extreme

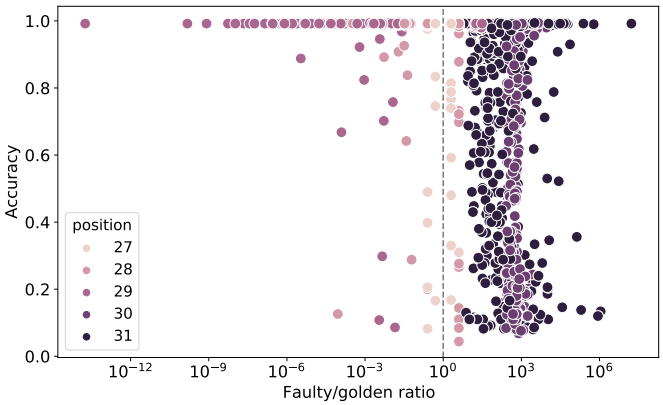

Fig. 6. Scatter plot showing accuracy and faulty/golden ratio for all injected accuracies in Posit format.

faulty/golden ratios, taking into consideration both the largest and smallest values. The accuracy of each faulty network was obtained by running inference over the same test subset as in previous experiments, and the results are presented in Figure 7. All the extreme values greater than one correspond to faults in bit 31, and most of them produce accuracy degradation. The extreme low ratio values correspond to bits 27 to 29 and do not affect the system accuracy.

V. Conclusions

In the present work, a low-cost analysis methodology is proposed to perform a preliminary dependability assessment of a deep neural network system under bit-flip faults affecting its parameters. For a given trained network and arithmetic representation, the method allows to identify the bit positions that are most critical and would justify the use of hardening techniques.

The proposed method is based on analyzing box-plots of the ratio between golden and faulty values of the affected

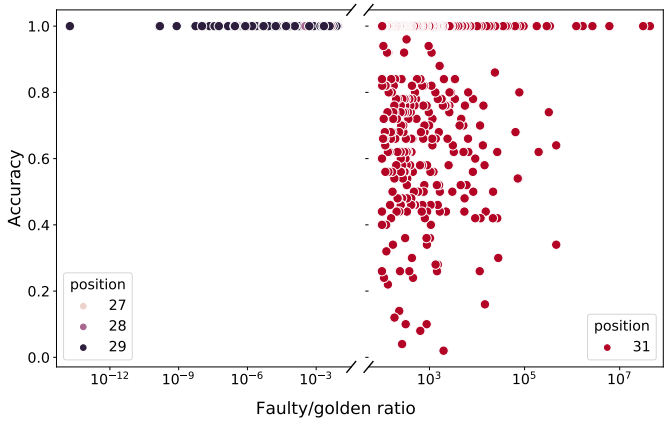

Fig. 7. Scatter plot showing accuracy for the worst 1000 faulty/golden ratio values in Posit format.

parameters, and has extremely low computational costs as the inference computation is not required.

An example LeNet-5 architecture trained for an image classification application using the MNIST dataset was used for a preliminary validation of the method. For each injected fault, the accuracy degradation was computed and analyzed together with the faulty/golden ratio. The preliminary results obtained from the example system suggest some interesting conclusions.

First, the faulty/golden ratio provides a very good indication of the bits in the arithmetic representation that may severely impair the DNN accuracy. Second, as already confirmed in the literature, faults producing a faulty value larger than the golden one (faulty/golden ratio greater than one) could be associated with important accuracy degradation. Interestingly, in some cases, the DNN may also mitigate the fault effect through the full network inference.

Finally, when the faulty value is smaller than the golden one almost no accuracy degradation was observed, with the only exception of a few faults affecting biases. If confirmed, this behavior would imply that faults that tend to decrease the value of the affected parameter (for example stuck at zero or flips to zero in floating-point format) are much less harmful. It also suggests that in this case biases are more critical than weights.

The plans for future work include extending the analysis to more models and datasets with the aim of validating the preliminary conclusions obtained for floating-point and Posit formats, and also to apply the method to other formats of common use in the industry.

REFERENCES

[1] L. Bertaccini, G. Paulin, M. Cavalcante, T. Fischer, S. Mach, and L. Benini, "MiniFloats on RISC-V Cores: ISA Extensions with Mixed-Precision Short Dot Products," IEEE Transactions on Emerging Topics in Computing, 2024.

[2] A. Agrawal, S. M. Mueller, B. M. Fleischer, X. Sun, N. Wang, J. Choi, and K. Gopalakrishnan, "DLFloat: A 16-b Floating Point Format Designed for Deep Learning Training and Inference," in 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH), vol. 2019-June. IEEE, 6 2019, pp. 92–95. [Online]. Available: https://ieeexplore.ieee.org/document/8877411/

[3] Intel, "BFLOAT16-Hardware Numerics Definition," 2018. [Online]. Available: https://www.intel.com/content/dam/develop/external/us/en/documents/bf16-hardware-numerics-definition-white-paper.pdf

[4] X. Sun, J. Choi, C.-Y. Chen, N. Wang, S. Venkataramani, V. V. Srinivasan, X. Cui, W. Zhang, and K. Gopalakrishnan, "Hybrid 8-bit Floating Point (HFP8) Training and Inference for Deep Neural Networks," in Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/65fc9fb4897a89789352e211ca2d398f-Paper.pdf

[5] J. Gustafson and I. Yonemoto, "Beating floating point at its own game: Posit arithmetic," Supercomput. Front. Innov.: Int. J., vol. 4, pp. 71–86, 6 2017. [Online]. Available: https://doi.org/10.14529/jsfi170206

[6] J. Gustafson, G. Bohlender, S. Yee Chung, V. Dimitrov, G. Jones, S. H. Leong (Cerlane), P. Lindstrom, T. Omtzigt, H. Rehr, A. Shewmaker, and I. Yonemoto, "Standard for Posit™ Arithmetic (2022)," Posit Working Group, Tech. Rep., Mar. 2022.

[7] B. Schlueter, J. Calhoun, and A. Poulos, "Evaluating the Resiliency of Posits for Scientific Computing," ACM International Conference Proceeding Series, pp. 477–487, Nov. 2023, publisher: Association for Computing Machinery ISBN: 9798400707858. [Online]. Available: https://dl.acm.org/doi/10.1145/3624062.3624116

[8] I. Alouani, A. Ben Khalifa, F. Merchant, and R. Leupers, "An Investigation on Inherent Robustness of Posit Data Representation," Proceedings of the IEEE International Conference on VLSI Design, vol. 2021-February, pp. 276–281, Feb. 2021, arXiv: 2101.01416 Publisher: IEEE Computer Society ISBN: 9780738112701.

[9] G. Gavarini, A. Ruospo, and E. Sanchez, "On the resilience of representative and novel data formats in CNNs," in IEEE International Symposium on Defect and Fault Tolerance in VLSI, 2023. [Online]. Available: https://gitlab.com/cerlane/SoftPosit-Python

[10] A. Ruospo, G. Gavarini, C. D. Sio, J. Guerrero, L. Sterpone, M. S. Reorda, E. Sanchez, R. Mariani, J. Aribido, and J. Athavale, "Assessing convolutional neural networks reliability through statistical fault injections," in Proceedings -Design, Automation and Test in Europe, DATE, vol. 2023-April, 2023.

[11] R. L. Sierra, J.-D. Guerrero-Balaguera, J. E. R. Condia, and M. S. Reorda, "Analyzing the impact of different real number formats on the structural reliability of tcus in gpus," in VLSI-SOC 2023: 31ST IFIP/IEEE CONFERENCE ON VERY LARGE SCALE INTEGRATION. IEEE Computer Society, 2023, - it is not said if some kind of quire format is used to avoid truncation in intermediate results. [Online]. Available: https://github.com/TheColombianTeam/PyOpenTCU.git

[12] J. E. R. Condia, J. D. Guerrero-Balaguera, R. L. Sierra, and M. S. Reorda, "Analyzing the structural and operational impact of faults in floating-point and posit arithmetic cores for cnn operations," Proceedings of the European Test Workshop, 2024.

[13] R. L. Sierra, J.-D. Guerrero-Balaguera, J. E. R. Condia, and M. S. Reorda, "Exploring hardware fault impacts on different real number representations of the structural resilience of tcus in gpus," Electronics 2024, Vol. 13, Page 578, vol. 13, p. 578, 1 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/3/578/htmhttps://www.mdpi.com/2079-9292/13/3/578

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[15] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141–142, 2012.

[16] R. Murillo, A. A. Del Barrio, and G. Botella, "Deep PeNSieve: A deep learning framework based on the posit number system," Digital Signal Processing, vol. 102, p. 102762, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S105120042030107X

[17] C. Leong, "Softposit-python," https://gitlab.com/cerlane/SoftPosit-Python.