2018 IEEE 36th International Conference on Computer Design

A Reliability Study on CNNs for Critical Embedded Systems

Mohamed A. Neggaz[1], Ihsen Alouani[1], Pablo R. Lorenzo[2], and Smail Niar[1]

[1]Polytechnic University Hauts-De-France, France
[2]The Silesian University of Technology, Poland

***Abstract*— Deep learning systems such as Convolutional Neural Networks (CNNs) have shown remarkable efficiency in dealing with a variety of complex real life problems. To accelerate the execution of these heavy algorithms, a plethora of software implementations and hardware accelerators have been proposed. In a context of shrinking devices dimensions, reliability issues of CNN-hosting systems are under-explored. In this paper, we experimentally evaluate the inherent fault tolerance of CNNs by injecting errors within network modules, namely processing elements and memories. Our experiments demonstrate a non uniform sensitivity between different parts of the system. While CNNs are relatively resilient to errors occurring in processing elements, transient faults hitting memories lead to catastrophic degradation of accuracy.**

I. Introduction and Related Work

Deep learning systems such as Convolution Neural Networks (CNNs) have achieved remarkable results in dealing with a plethora of complex real life problems. From handwritten digit recognition, to environment perception for autonomous cars, deep neural networks (DNNs) have demonstrated their ability to train robust feature extractors that can be successfully exploited by a classifier. Because of the massive amount of data handled by CNNs, their implementation in embedded systems requires high performance Hardware platforms. While the literature is focusing on strategies for increasing the accuracy of CNNs and adapting them to new tasks, the reliability of CNN-dedicated hardware still remains under-explored.

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. In safety-critical systems, incorrect values being unreliably computed represent a serious issue, as these systems must comply with strict safety standards [1].

The impact of faults on layer activations can be compared to approximate computing. Chen et. al [2] have shown this effect by using approximate memories. Other approximating techniques involves fixed-point quantization of a CNN [3] which plays on bit-width of operators (activations and weights). However, faults have a random aspect that cannot be compared to approximate computing techniques where possible changes are defined.

Recently, the resilience to errors issue has been addressed in [4] where authors propose a method for predicting the error resilience of neurons in a deep neural network. However, authors didn't consider multiple events neither bit flips within memory. Fault characterization in CNN-based embedded systems is important. Because of resource constraints, hardening approaches for these systems must incur a small overhead in terms of area and energy.

In this paper, we study the impact of errors on the accuracy of CNN models for critical embedded systems. We present a methodology that simulates two scenarios: in-memory errors that can affect the network's weights stored in SRAMs, and incorrect propagations in combinational circuits that can produce corrupted layer activations. We undertook an extensive experimental study, involving scenarios with different levels of error injection. We showed that:

- Our approach successfully characterizes the distribution of errors in layer-wise activations of the CNN.
- Our approach shows that errors introduced in activations have a small impact, and their effect can be mitigated naturally by the network.
- Our approach shows that errors introduced in memories have a huge impact on the network accuracy.
- Our approach can be used to construct a set of reliability guidelines for the deployment of CNNs in critical and aggressive environments.

The remaining of the paper is organized as follows: Section II discusses the background and state of the art of Soft Errors and CNNs. In section III we introduce our proposed methodology. Section IV displays the experimental setup, and Section V discusses the results obtained. Section VI concludes the paper.

II. Background

A. Soft Errors

As sub-micron technology dimensions sharply decreased to a few nanometer range in commercialized ICs, the sensitivity of electronic circuits increased drastically. Errors may result from a voltage transient event induced by alpha particles from packaging material or neutron particles from cosmic rays [5]. A sufficient amount of accumulated charge in the struck node may invert the state of one or multiple sequential elements, such as latches and static SRAM cells, thereby resulting in single event upsets (SEUs) or multiple bit upsets (MBUs) [6].

2576-6996/18/$31.00 ©2018 IEEE
DOI 10.1109/ICCD.2018.00077

B. Convolutional Neural Networks

CNNs are composed by a sequence of layers, where each layer transforms one volume of activations to another through a differentiable function. In their simplest formulations, CNNs are composed by three different types of layers (i.e *convolutional*, *pooling* and *fully connected* layers) that are stacked together with the goal of expressing a single differentiable score. The objective of these networks is to develop powerful feature extractors, which are able to capture the most salient features of the input data.

Ultimately, the highest-level features are exploited by a classifier to produce a prediction. A drawback of CNNs is that they need to be trained using a high volume of data.

III. Proposed Methodology

We define a method for error injection relying on logical flips at bit level. It is constructed as a custom DNN layer, and can be seamlessly inserted at any stage in a model. When an input tensor—typically Float32—denoted as is I is received, our method selects n random values $I_{i,j}$, flipping one bit in each of them. In our error injection method, a bit is selected from the single-precision floating-point representation of the index $I_{i,j}$ by drawing from a uniform distribution $\sim \mathcal{U}(0, 32)$. Finally, this bit is flipped and the index $I_{i,j}$ is substituted with its altered version $I'_{i,j}$.

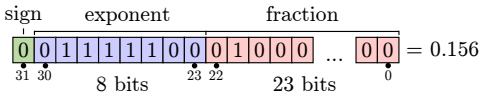


Fig. 1: Description of the IEEE-745 single-precision floating-point format.

Note that the relative impact of an error depends on the section where it occurs, and the significance of this bit with respect to its section.

A. Error injection in layer activations

Let $\mathcal{A}_0$ denote the *golden run* and let $\mathcal{A}_k$ denote the *k-th dirty run*—referring to an activation with k injected errors. It is possible to build a minimal model with only two levels deep to collect a large enough sample to characterize the distribution of the errors in $\mathcal{A}$ with respect to $\mathcal{G}$. The two layers are: a KCWH convolutional layer with K number of filters, C channels, W filter width and H filter height.

B. Full model fault injection

The aim of this experiment is to quantify errors on a fully trained model. We study accuracy after fault injection, which is defined as the ratio between correct and incorrect classifications. We distinguish two targets of soft-errors: memory errors, and compute errors targeting the combinatorial circuits.

In the case of events affecting compute units, the impact is transient. In our model, operations only depend on the given input and the network parameters. In our simulation model, given a probability, events will only affect the results (classification accuracy) of that given run.

If a fault caused by a transient event flips a bit in memory, corrupted data will be stored for future use. To model this behavior, we consider the impact of successive events each of which causes a bit-flip that should be seen by future events.

IV. Experimental Setup

Initially, three different experiments are created as part of an ablation study, in order to understand the effect that error injection has on the performance of full DNN models. The first experiment consists of characterizing the effect of logical errors in the activation of a convolutional layer. With this purpose, a 2-layer model composed by a convolutional and our custom bit flip layer is assembled, and a number of activations are computed to characterize the distribution of the error defined as $|\mathcal{A}_k - \mathcal{A}_0|$. In the second and third experiments, we track the impact of transient errors on the overall performance of the DNN. We consider single and multiple event transients on combinational circuits as well as single and multiple event upsets in memories.

A. Simulation environment

Our method for fault injection was implemented in Python 3.6 with NumPy, and DNN were built with PyTorch 0.4 over CUDA 9.0 and CuDNN 7.1.

Experiments on layer activations were run on Intel i7-6850K (15M Cache, 3.80 GHz) with 32GB RAM and NVIDIA Titan X Ultimate Pascal GPU 12GB GDDR5X. We used a fixed input size of 28×28, and the convolutional layer had 1 filter of size 3×3 with stride 1. Output activation was of size 26×26 and a rectified linear unit (ReLU) was used as activation function. Results were collected from 1 million sample activations with different parameterizations for k.

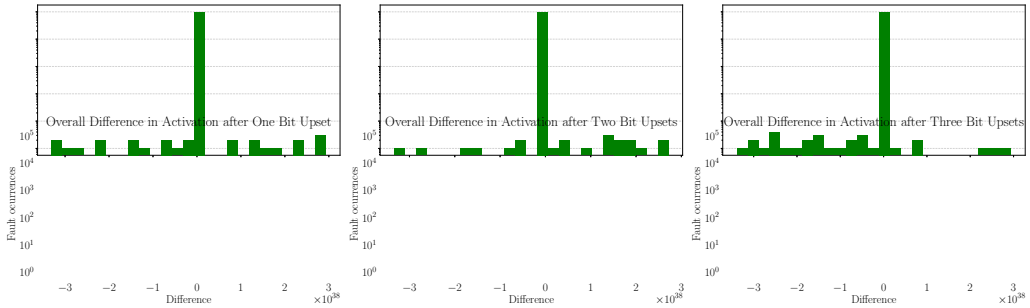
Full-model study was run on an Intel i7-6500U with 8GB of RAM running Python 3.5.2 with PyTorch 0.4. Experiments were performed on LeNet network trained for 16 epochs on the MNIST dataset.

B. Dataset

In full-model fault injection experiments, we focus on characterizing the performance of the network on multi-class classification tasks. For this, we use the well known MINST dataset [7]. MNIST is a dataset of handwritten digits consisting of 70,000 gray-scale images (28×28 pixels) divided into 10 classes, with approximately 7,000 images per class.

C. DNN model

In our full-model experiments, we study the effect of noise on the well known LeNet5 architecture [8]. LeNet5 consists of two parts (from left to right): a feature extractor, and a classifier appended to it. Feature extraction is performed by two unpadded convolutional layers with small filters of size 5×5, with 20 and 50 filters respectively, and followed by ReLU activations. The classifier is composed by 3 fully connected layers, terminated by a softmax activation.



(a) Distribution of $\mathcal{A}_1$. (b) Distribution of $\mathcal{A}_2$. (c) Distribution of $\mathcal{A}_3$.

Fig. 2: Distribution of $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ over 1 million activations in logarithmic scale.

V. Experimental Results

A. Layer-wise Analysis of Multiple Event Transients

In this experiment, we focus on characterizing the distribution of of the error between golden activations and faulty activations, defined as $|\mathcal{A}_k - \mathcal{A}_0|$. For this, we build a 2-layer model that takes 28×28 input tensors and produces a 26×26 output resulting from an unpadded KCHW convolution, where $K = 1$, $C = 1$ and $H = W = 3$, terminated by a bit flip layer. For this experiment, $k = \{1, 2, 3\}$ are selected in order to observe how increasing levels of corruption can affect the produced activation.

Figure 2 displays the distribution of $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ over 1 million activations under single, double and triple bit flip injections. We observe that the activation errors are visibly centered around zero, with 90% of activations being of small size. This is consistent with the statistical expectation of the error, as comes defined by the disposition of the bits with respect to the IEEE-754 format. It is interesting that the error doesn't seem to grow out of control even with 3 bit flips by occurrence. This is clearly due to the fact that alterations in the fraction part of a 32-bit float might carry a low impact in the final error.

B. Impact of soft-errors on model performance

This experiment evaluates the performance degradation caused [illegible] inferen[illegible] of 0.9[illegible] compu[illegible]

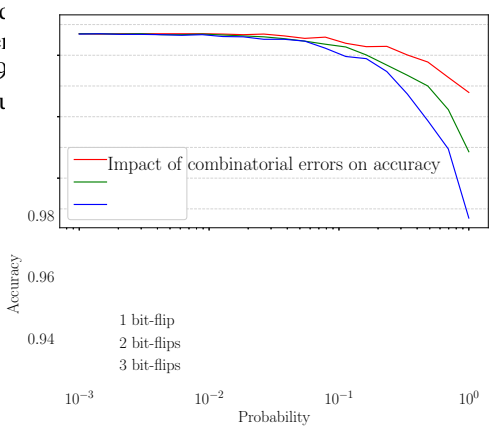


Fig. 3: Comparison between single and multiple event transients and their impact on overall network performance.

1) Single/multiple event transients: We insert the previously defined injection layer at each step and record the obtained accuracy while increasing the fault injection probability. For each probability, we measure the average accuracy on the test set of 10k samples.

Figure 3 illustrates the drop of accuracy of the aforementioned network. As expected, the negative impact of injecting multiple errors is higher than single error injection. The accuracy loss starts at a fault injection probability of 10^{-2} and accelerates exponentially. However, even with multiple fault injections in combinational circuits, the overall accuracy degradation is not dramatic. This is not only due to the alterations in the fraction part of a 32-bit float, but also to the CNN properties. Practically, these capabilities of neural networks make them inherently robust to SETs.

2) Single/multiple event upsets: Although both SETs and SEUs are caused by the same *transient* phenomenon, there is a fundamental difference between them. While glitches caused by SETs are volatile in time, SRAM cells are bistable elements and a bit flip leads the cell to a new stable corrupted state. This is translated by a cumulative aspect of the errors that needs to be considered in simulations.

Figure 4 shows the effects of memory errors that correspond to alterations in weights on the network accuracy. As we can notice, the best case scenario stagnates at almost perfect accuracy. However, the worst case scenario can transform the network from an almost always right to an almost random number generator. Before the 10 error step, the network can instantly drop below acceptable accuracy. This is explained by the nature of weights and their relative importance when determine the network's final decision.

The drop of accuracy continues until reaching of 10%. At this level, the network is either outputting the same number each time or outputting random numbers.

The impact of memory errors is not equally distributed between layers. Some layers are more critical if exposed to the same amount of faults. In the next experiment we evaluate the responsibility of each layer on the overall network performance.

The difference of singular impact can be viewed in Figure 5. The Figure 5a shows that the first layer is the most resilient compared to others. Convolutional layers do not have much impact when compared to fully connected layers (Figure 5c, 5d). This is due to two main reasons:

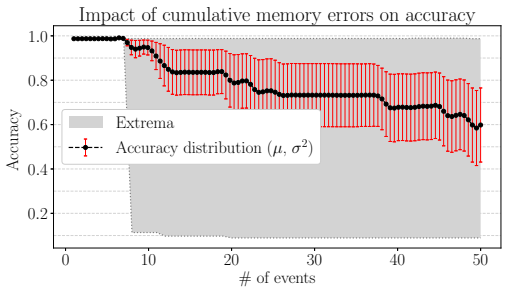


Fig. 4: Evolution of average accuracy (in black) when the number of errors augments. The dispersion and the extreme cases are shown in red and gray respectively.

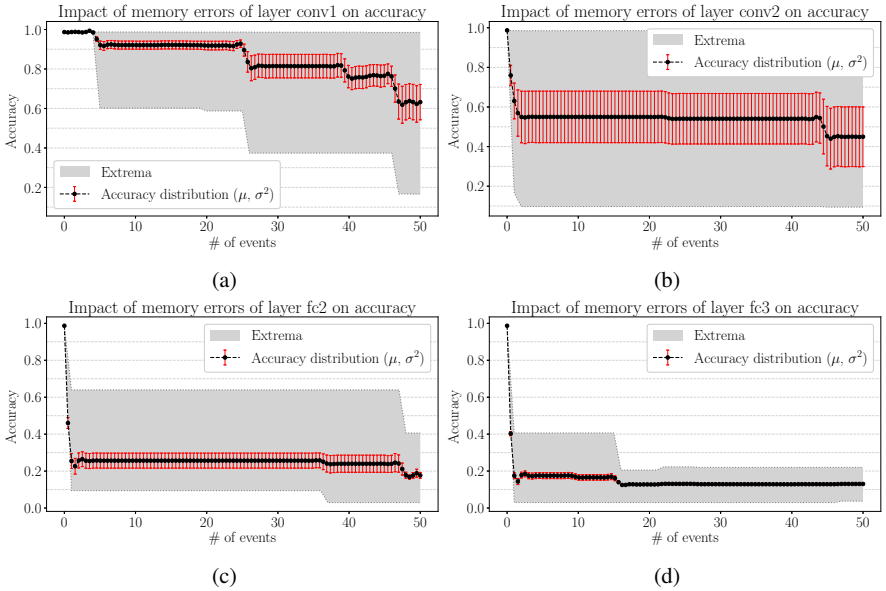


Fig. 5: Layer-wise analysis of memory soft-errors with independent responsibility. Average accuracy is in black circles with the standard deviation in red. The interval of extreme cases (minimum and maximum) is illustrated with the gray region.

- Errors in fully connected layers lead to direct impact on the output. However, errors in convolutional layers can be masked during the classification.
- Convolutional layers are followed by max-pooling layers. It is not the case in fully connected layers.

VI. Conclusions

This paper presents an exhaustive exploration of the transient errors impact on deep neural networks, notably CNNs. Although the regularization abilities of deep neural networks provide them with an added degree of robustness against external disturbance, our empirical findings add nuances to this widely admitted claim. In fact, while errors in processing elements do not have a huge impact on the overall system accuracy, a drastic degradation was noticed with faults occurring in memory.

References

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

[2] Y. Chen, Y. Zhu, F. Qiao, J. Han, Y. Liu, and H. Yang, "Evaluating data resilience in cnns from an approximate memory perspective," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, ser. GLSVLSI '17. New York, NY, USA: ACM, 2017, pp. 89–94. [Online]. Available: http://doi.acm.org/10.1145/3060403.3060435

[3] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," *CoRR*, vol. abs/1511.06393, 2015. [Online]. Available: http://arxiv.org/abs/1511.06393

[4] C. Schorn, A. Guntoro, and G. Ascheid, "Accurate neuron resilience prediction for a flexible reliability management in neural network accelerators," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 979–984.

[5] J. Ziegler and al., "Ibm experiments in soft fails in computer electronics," *IBM Journal of Research and Development*, vol. 40, no. 1, 1996.

[6] I. C. et al., "Impact of technology scaling on sram soft error rates," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 6, pp. 3512–3518, Dec. 2014.

[7] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks."

[8] 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.