Contribution ID: **121** Type: **Oral**

Mitigating Multiple Single-Event Upsets During Deep Neural Network Inference using Fault-Aware Training

Wednesday 2 October 2024 14:40 (20 minutes)

Deep Neural Networks are increasingly deployed at safety-critical operations. In order to enable this technology for harsh environments that contain high levels of radiation, fault analysis and mitigation is required. In this study, we present a model-based fault injection campaign to analyze the impact of multiple Single-Event Upsets (SEUs) in Deep Neural Networks (DNNs). Furthermore, we propose a Fault-aware Training (FAT) methodology to increase the reliability of DNNs without any modification to the hardware. Experimental results show that the FAT methodology improves the fault tolerance up to a factor of 20.

Summary (500 words)

Over the past decade, DNNs have made remarkable advancements in terms of performance. However, their reliability remains a concern for safety-critical operations, especially when deployed in environments that contain high levels of radiation. In these harsh environments, DNNs are susceptible to SEUs, which can lead to bit flips causing numerical errors or even a system crash.

One approach to mitigate SEUs is Radiation Hardening by Design (RHBD). However, as this implies adding redundancy to the hardware, it comes with significant overhead in terms of energy consumption, surface area and latency. Moreover, for challenging environments with a high SEU rate, this approach alone would be highly inefficient. DNNs are inherently redundant, such that an upset may not necessarily lead to a different outcome. Therefore, in this work it is investigated to which extent the necessary overhead in hardware can be reduced by leveraging the DNN's inherent tolerance to faults. In contrast to existing literature, our work focusses on multiple SEUs per inference, targeting the most challenging environments such as particle accelerators and deep-space missions.

This paper makes two key contributions. Firstly, it offers a comprehensive analysis of the fault tolerance of various parts within the DNN. Secondly, it introduces a FAT methodology to increase the fault tolerance of these DNNs.

To achieve these objectives, we designed a model-based and hardware-agnostic fault injection tool, built around the PyTorch framework. A fault injection campaign was conducted on two quantized DNNs including a shallow convolutional neural network (denoted as CCDF) trained on MNIST and a MobileNetV2 model trained on the CIFAR10 dataset. To facilitate our analysis, we separated the quantized DNN-layers into five different modules, including the input, weight and output, which are 8-bit integers and the bias and accumulator, which are 32-bit integers. Notably, we observed a clear correlation between accuracy and the number of injected faults during the model inference. While allowing a reduction of maximum one percent compared to the fault-free accuracy, the CCDF model can tolerate 25 faults, while the MobileNetV2 model can tolerate 50 faults. We also observed that the bias and accumulator are significantly more sensitive than the other parts of the model, which was concluded to be related to the higher dynamic range.

In our second contribution, we proposed a FAT mitigation technique, where we utilize our fault-injection tool to inject bit-faults during the training phase of the DNNs. After repeating the same analysis, we observed that the CCDF model, trained using FAT, can withstand up to five times more faults than the original model, while

allowing an accuracy drop of one percent compared to the fault-free model. The FAT MobileNetV2 model, on the other hand, can withstand up to 20 times more faults, using the same configuration.

These findings indicate that DNNs can be designed to tolerate a significant number of faults.

Authors: Dr DEKKERS, Gert (Magics Technologies); PRINZIE, Jeffrey; JONCKERS, Naïn; Prof. KARSMAKERS, Peter (KU Leuven); Mr VINCK, Toon (KU Leuven)

Presenter: Mr VINCK, Toon (KU Leuven)

Session Classification: Programmable Logic, Design and Verification Tools and Methods

Track Classification: Programmable Logic, Design and Verification Tools and Methods