FEATURE MAP VULNERABILITY EVALUATION IN CNNS

Abdulrahman Mahmoud [1] **Siva Kumar Sastry Hari** [2] **Christopher W. Fletcher** [1] **Sarita V. Adve** [1] **Charbel Sakr** [1] **Naresh Shanbhag** [1] **Pavlo Molchanov** [2] **Michael B. Sullivan** [2] **Timothy Tsai** [2] **Stephen W. Keckler** [2]

ABSTRACT

As Convolutional Neural Networks (CNNs) are increasingly being employed in safety-critical applications, it is important that they behave reliably in the face of hardware errors. Transient hardware errors may percolate undesirable state during execution, resulting in software-manifested errors which can adversely affect high-level decision making. We present ***HarDNN***, a software-directed approach to identify vulnerable computations during a CNN inference and selectively protect them based on their propensity towards corrupting the inference output in the presence of a hardware error. We show that HarDNN can accurately estimate relative vulnerability of a feature map in CNNs using a statistical error injection campaign, and explore heuristics for fast vulnerability assessment. Based on these results, we analyze the tradeoff between error coverage and computational overhead that the system designers can use to employ selective protection. Results show that the improvement in resilience for the added computation is superlinear in many cases with HarDNN. For example, HarDNN improves SqueezeNet's resilience by $10\times$ with just 30% additional computations.

1 INTRODUCTION

CNNs have seen a recent surge in usage across many application domains ranging from High Performance Computing (HPC) to safety-critical systems such as autonomous vehicles and medical devices. We have also seen a rise in the use of efficient platforms that accelerate CNN executions such as GPUs and domain-specific accelerators such as the one deployed in Tesla's Full Self-Driving (FSD) System (9; 5; 7). As CNNs continue to permeate the fabric of everyday life with increasing utilization in safety-critical applications, it is important that they are resilient to transient hardware errors (also known as soft errors).

Studies have shown that hardware errors could have severe unintended consequences unless the system is designed to detect these errors (6; 10). For example, following a series of unintended acceleration events by Toyota vehicles, a taskforce following up on a NASA investigation showed that, "as little as a single bit flip ... could make a car run out of control." To mitigate such scenarios, hardware in safety-critical systems must fulfill high integrity requirements, such as those outlined in the ISO-26262 standard (2).

While processors deployed in safety-critical systems will employ ECC/parity to protect large storage structures (storing weights and intermediate data), the level of protection they offer will likely be not sufficient to meet the stringent requirements set by standards such as ISO-26262. Conventional reliability solutions, such as full duplication through hardware or software, suffer from high overheads in cost, area, power, and/or performance (1; 3; 8), yet are still commonly used in practice to ensure high resilience. For instance, Tesla's FSD system deploys two fully redundant FSD chips along with accompanying redundant control logic, power, and peripheral packaging for reliability.

With the goal of developing a reliability solution that is much lower cost than full duplication, we seek to understand the underlying vulnerability characteristics of CNNs. Instead of simply approaching a CNN as a single, large computational block, we explore its vulnerability at finer granularities (i.e., neurons, feature maps, and layers). We hypothesize that not all sub-components of a CNN contribute equally to the overall network vulnerability, and develop methodologies to quantify vulnerability at a finer granularity. Our results show that errors in some feature maps or layers are more likely to corrupt the output of a CNN. Furthermore, we recognize that feature maps are robust to translation effects in the input, maintaining higher-level information required by the CNN for inference, while a technique that operates at neuron-level will not have this benefit. Based on this advantage and the fact that we can compose the vulnerability estimates at layer or network level using feature map level analysis, we focus on feature map level granularity in this work.

One technique commonly used to quantify an application's vulnerability to transient errors is error injection experiments. An exhaustive study, where one error simulation is performed for a possible hardware error in an applica-

[1]University of Illinois at Urbana-Champaign [2]NVIDIA
This material is based upon work supported in part by the Applications Driving Architectures (ADA) Research Center, a JUMP Center co-sponsored by SRC and DARPA. A portion of this work was performed while Abdulrahman Mahmoud interned at NVIDIA.

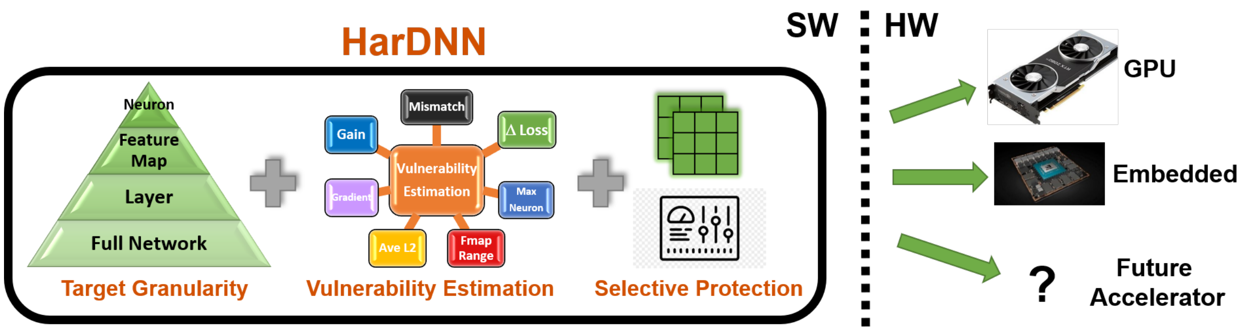

Figure 1. Overview of HarDNN. HarDNN takes as input a pretrained network, ranks feature maps based on vulnerability, then selectively duplicates feature maps before deployment.

tion state to quantify the effect on the output, is often intractable. Instead, resilience analyses typically employ a statistical technique to limit the number of error simulation runs, while preserving the quality of results. We leverage a similar approach to quantify the vulnerability of a CNN's component when subjected to various transient error models by analyzing the likelihood of a Top-1 misclassification for (classification) models.

In an error injection run, the output of a CNN can be corrupted but the classification might still be correct. To capture the severity of output corruption in each injection run, we propose using an alternate metric that uses the average change in cross entropy loss ($\Delta loss$). We find that capturing the fine-grain severity metric (instead of binary classification result) can produce vulnerability estimates that are comparable whether the classification changes, but several times faster (e.g., 10$\times$ for ResNet50). For an additionally faster method, we explore six heuristics that do not perform error injections to estimate vulnerability. We leverage activation values and gradient information during inference and back propagation. These heuristics provide an additional tradeoff between accuracy and speed for vulnerability assessment.

We also study the tradeoff between resilience improvement and increase in computation for added resilience offered by protecting highly vulnerable feature maps. Results show that a fraction of feature maps typically account for a disproportionately large percentage of output corruptions (on average, 30% of feature maps account for 76% of output corruptions for the studied CNNs). Since each feature map is a convolution of the input based on an given filter, a low-cost mitigation technique can be selective filter (or feature map) duplication.

2 Feature Map Vulnerability Evaluation in CNNs

HarDNN is a highly tunable technique which can be used to identify and harden the most vulnerable components of a CNN for hardware-error-resilient inferences. Figure 1 shows a high level overview of the system. Specifically, we make the following contributions in this work (more details are available in the full paper (4)):

- We compare various granularities for protection to avoid full CNN duplication. We identify that *feature maps* provide a "sweet spot" for analysis and protection, due to their robustness to translation effects of inputs, and their composability for high-level (e.g., layer-level) protection.

- We study the sensitivity of error models on the contribution of a feature map towards the total vulnerability, which we call *relative vulnerability*. Results show that the relative vulnerabilities of feature maps do not change much with the studied error models.

- We introduce Δloss as an accurate metric to measure vulnerability. Δloss measures the *magnitude* of the difference between an error-free inference and an erronous inference. It better captures fine-grain perturbations in inference output and converges to the relative vulnerability estimate per feature map with far fewer injections compared to the baseline Top-1 classification-based criterion.

- We evaluate multiple non-injection based heuristics for vulnerability estimation and compare their accuracy and speed to Δloss. The heuristics we explore leverage forward- and/or backward- pass information during a DNN inference, to estimate vulnerability.

- We study the tradeoff between resilience improvement and increase in computation for the resilience offered by protecting highly vulnerable feature maps. Results show that HarDNN improves resilience of SqueezeNet, for example, by 10$\times$ with just 30% additional computations.

References

[1] Wendy Bartlett and Lisa Spainhower. Commercial Fault Tolerance: A Tale of Two Systems. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, pages 87–96, January 2004.

[2] International Organization for Standardization. Road vehicles – Functional safety. Website, 2011. https://www.iso.org/standard/43464.html.

[3] X. Iturbe, B. Venu, E. Ozer, and S. Das. A Triple Core Lock-Step (TCLS) ARM® Cortex®-R5 Processor for Safety-Critical and Ultra-Reliable Applications. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 246–249, June 2016.

[4] Abdulrahman Mahmoud, Siva Kumar Sastry Hari, Christopher W. Fletcher, Sarita V. Adve, Charbel Sakr, Naresh Shanbhag, Pavlo Molchanov, Michael B. Sullivan, Timothy Tsai, and Stephen W. Keckler. HarDNN: Feature Map Vulnerability Evaluation in CNNs. arXiv:2002.09786, February 2020.

[5] NVIDIA. Self-Driving Car Hardware — NVIDIA DRIVE. https://www.nvidia.com/en-us/self-driving-cars/drive-platform/hardware/.

[6] Safety Research and Strategies, Inc. Toyota unintended acceleration and the big bowl of 'spaghetti' code. Website, 2013.

[7] Sean Hollister. Tesla's new self-driving chip is here, and this is your best look yet. 2019.

[8] Alex Shye, Joseph Blomstedt, Tipp Moseley, Vijay Janapa Reddi, and Daniel A. Connors. PLR: A Software Approach to Transient Fault Tolerance for Multicore Architectures. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 6(2):135–148, April 2009.

[9] V. Sze, Y. Chen, T. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, Dec 2017.

[10] Junko Yoshida. Toyota case: Single bit flip that killed. Website, 2013.