Resiliency of automotive object detection networks on GPU architectures

Atieh Lotfi, Saurabh Hukerikar, Keshav Balasubramanian, Paul Racunas,
Nirmal Saxena, Richard Bramley, Yanxiang Huang
NVIDIA

***Abstract*—Safety is the most important aspect of an autonomous driving platform. Deep neural networks (DNNs) play an increasingly critical role in localization, perception, and control in these systems. The object detection and classification inference are of particular importance to construct a precise picture of a vehicle's surrounding objects. Graphics Processing Units (GPU) are well-suited to accelerate such DNN-based inference applications since they leverage data and thread-level parallelism in GPU architectures. Understanding the vulnerability of such DNNs to random hardware faults (including transient and permanent faults) in GPU-based systems is essential to meet the safety requirements of auto safety standards such as the ISO 26262, as well as to influence the design of hardware and software-based safety features in current and future generations of GPU architectures and GPU-based automotive platforms. In this paper, we assess the vulnerability of object detection and classification DNNs to permanent and transient faults using fault injection experiments and accelerated neutron beam testing respectively. We also evaluate the effectiveness of chip-level safety mechanisms in GPU architectures, such as ECC and parity, in detecting these random hardware faults. Our studies demonstrate that such object detection networks tend to be vulnerable to random hardware faults, which cause incorrect or mispredicted object detection outcomes. The neutron beam experiments show that existing chip-level protections successfully mitigate all silent data corruption events caused by transient faults. For permanent faults, while ECC and parity are effective in some cases, our results suggest the need for exploring other complementary detection methods, such as periodic online and offline diagnostic testing.**

***Keywords*—Automotive, Functional Safety, Resilience, Graphic processing unit, Object detection network**

I. Introduction

There is much excitement about self-driving cars and trucks powered by artificial intelligence revolutionizing the transportation industry. However, safety is the most important aspect of an autonomous driving platform - automotive applications have stringent demands for functional safety and reliability, such as the ISO 26262 auto safety standard [1], which represents the state-of-the-art in automotive functional safety. A drive system in a fully autonomous car that takes the place of a human driver is expected to be highly resilient. The ISO 26262 provides regulations and guidelines for building functionally safe systems as well as quantifiable safety metrics, such as the probabilistic metric for hardware failure (PMHF), which represents a calculated estimate of the rate of hazard occurrence due to random hardware failures. The Automotive Safety Integrity Level (ASIL) is a key component for ISO 26262 compliance with level D being the highest in terms of hazard levels and stringent safety measures and ASIL A being the least stringent. For ASIL D compliance, the specification requires the system failure rate to be less than 1E-08/hr, which mandates the FIT rate (failures per billion hours of operation) to be less than 10 FIT [2].

Much of the intelligence that allows vehicles to respond to variation and complexity in the driving environment is enabled by learning algorithms based on deep neural networks (DNNs). These networks are used for object detection and localization in driver assistance and autonomous vehicle operation. The output of a detection network is a key input for path planning and navigation control. Graphics processing units (GPU) provide the high-throughput computational performance that enables the use of such networks for dynamic object detection and classification during autonomous driving operation. This capability of detecting obstacles, other vehicle traffic and pedestrians is critical in autonomous vehicles [3]. Therefore, understanding the vulnerability of such object detection neural networks to faults when executing on GPU-based platforms is of great importance to meet the safety guidelines of the ISO 26262.

The ISO 26262 specification requires reliable operation in the presence of random hardware faults. These include permanent faults, which are caused by physical defects (such as burn-out or short circuit between connections) and persist over a period of time, and transient faults caused by exposure to the environmental and cosmic radiation [4] [5]. Any transient or permanent fault in the GPU hardware may impact the object detection network causing incorrect or mispredicted objects in the path of the vehicle, which may result in violating safety goals. Understanding the resiliency properties of the object detection neural networks to both of these fault types influences the design of hardware and software-based safety features in current and future generations of GPU architectures and GPU-driven automotive object detection platforms.

In this paper, we assess the vulnerability of automotive object detection networks to permanent and transient faults using fault injection experiments and accelerated neutron beam testing. Our study demonstrates that these object detection and localization networks tend to demonstrate some vulnerability to random hardware faults which may cause failure of the detection network. We evaluate the effectiveness of safety mechanisms in GPU architectures through beam testing experiments and the extent to which the vulnerability is mitigated by the safety mechanisms. Our studies show that enabling

978-1-7281-4823-6/19/$31.00 ©2019 IEEE

mechanisms such as ECC and parity in GPU architectures eliminate all silent data corruption events related to transient faults seen in an unprotected network. However, for permanent faults that are not detected by ECC/parity, we need periodic testing mechanisms. The main contributions of this paper are:

- We evaluate the impact of permanent faults on object detection networks used in autonomous vehicles through fault injection experiments.
- We perform accelerated neutron beam experiments to evaluate the sensitivity of such object detection networks to transient faults.
- We develop a methodology for determining the criticality of faults on the object detection and classification networks.
- We present analysis of the chip-level protections mechanisms in GPU architectures in terms of their effectiveness in providing adequate coverage to faults.

While the focus of our study is on the vulnerability of object detection networks under the assumption of random hardware faults, this vulnerability also becomes relevant to design or process-related faults (called systematic faults in ISO 26262 nomenclature).

The remainder of this paper is organized as follows. Section II describes the structure of object detection networks and discusses how we evaluate their vulnerability to faults. Section III reviews some background information on the target GPU family architecture. Section IV presents permanent fault injection methodology and results while Section V presents insights from accelerated neutron beam testing about the impact of transient faults. Section V also analyses the adequacy of chip-level protections in existing GPU architectures, such as ECC and parity, for mitigating silent data corruptions that may occur when executing such convolution neural network-based automotive detection and classification. Section VI summarizes the related work. Finally, Section VII concludes the paper.

II. Fault Vulnerability of Object Detection Network on GPUs

A. Object detection network

DNNs are one of the most efficient methods for object detection. Object detection entails image classification and object localization. Image classification determines which objects are in the image. Object localization identifies the location of objects in the image along with their confidence probabilities. The location of an object in a scene is represented using bounding box coordinates. These object detection neural networks usually contain several convolutional layers followed by fully connected layers and other layers such as pooling and normalization layers. After constructing a network topology, the network can be trained by a given set of training input image data. Throughout the training phase, associated weights are learned through a backpropagation mechanism. Once the network training is complete, the network may be used for object detection with test input image data, which is referred to as the inference phase. Autonomous vehicles deploy previously trained object detection networks, and therefore we focus our efforts on understanding the resiliency characteristics of such inference networks. In this work, we use an object detection neural network that predicts a confidence score for each detected bounding box using logistic regression. The inference, which is performed on a GPU, takes a number of images as input, processes the images through multiple layers of the network, and outputs coordinates of bounding boxes of objects and a confidence score based on the probability of the presence of the object in that region of the image. This output requires post-processing that clusters neighbouring bounding boxes with sufficiently high confidence together and outputs more precise outlines for the locations of the detected objects in the image.

B. Fault Vulnerability Evaluation Methodology

Any transient or permanent fault in the GPU hardware may cause a failure in the object detection network, which may result in violating a safety goal. To evaluate the vulnerability of the automotive object detection network, we inject faults while the inference network is running on the GPU and compare the outputs of the faulty network with the outputs produced by a non-faulty *golden* network. The overview of this process is illustrated in Figure 1. Since the output of our object detection network for each input image is a set of bounding boxes and their corresponding confidence values, the process of determining whether a fault has occurred entails the following steps: first, we discard the bounding boxes with confidence less than a specific threshold (*Conf_thr*). Next, we perform a clustering step to discover bounding boxes that have sufficient overlap and to merge them. The clustering is performed by applying the DBscan algorithm [6]. DBscan returns the weighted mean of neighboring rectangle coordinates such that their sum of confidence scores is higher than a specified threshold. To assess whether any two bounding boxes overlap, we use the intersection over union (*IoU*) metric. IoU is the ratio of the intersection area over the union area of two bounding boxes. The closer IoU is to 1, the higher overlap two bounding boxes have. If the IoU is greater than a desired value (*IoU_clustering*), we cluster the bounding boxes into one bounding box, and if the sum of confidence scores among these overlapping neighbors are larger than a desired value (*Min_pts*), then that cluster of bounding boxes is selected

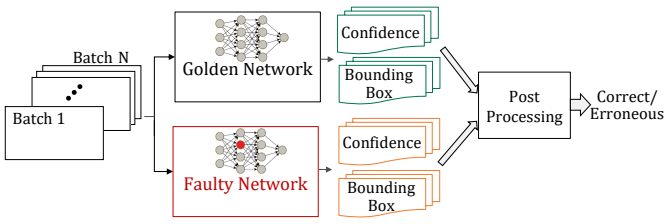


Fig. 1: Evaluating the impact of faults on the inference object detection network accelerated by GPUs

as a detected object in the final set. All the other bounding boxes that are not selected for the clustering are pruned from the final set of bounding boxes presented as the output.

Comparing bounding boxes of the objects detected by the faulty network with those detected by the golden network, we can determine whether the fault caused silent data corruption (*SDC*) in the output of the network (i.e. if it makes the output erroneous and incorrect). It should be noted that not all errors in the output are critical in object detection networks. For example, if the errors modify the probabilities such that they do not impact the rank of a clustered object, or if they only change the coordinates of low-probability bounding boxes, they are not considered as SDCs. To evaluate the fault vulnerability of object detection networks, we use the aforementioned IoU metric to compare outputs of golden and faulty networks. The analysis has three abstracted outcomes. If the IoU of the faulty box and the golden box is greater than a threshold (*IoU_threshold_comparison*), then that specific object is correctly detected; therefore, it is a safe scenario or the error is *masked*. The other outcome is defined as *inclusion*, where the IoU is less than a threshold but the faulty object detects the same object as the golden network and the bounding box detected by the faulty network includes the golden bounding box. The inclusion case can be considered as either safe or SDC: It can be considered safe as the path planning algorithm avoids hitting the object or it can be considered an unsafe event if the object appears closer than it should and results in unanticipated unsafe breaking. Otherwise, if IoU of faulty and golden bounding boxes is less than *IoU_threshold_comparison* and the golden bounding box is outside the faulty bounding box, the outcome is an *SDC* event. This event space is visualized in Figure 2.

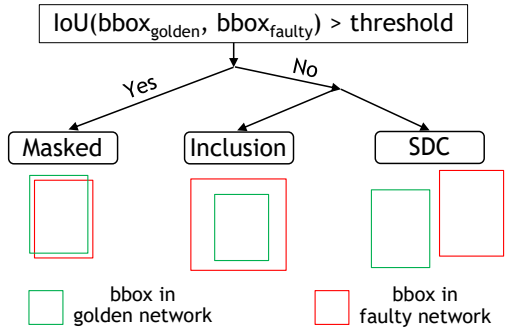
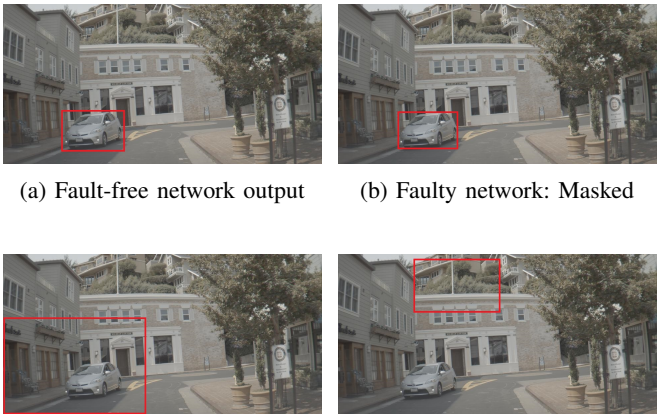


Fig. 2: Comparison outcomes for faulty and golden objection detection networks (bbox stands for bounding box)

Figure 3 shows examples for the effect of fault on the object detection network output. Figure 3a shows the golden network output, where the detected car is marked by a red rectangle. Figures 3b, 3c, and 3d show examples of faulty network when it results in *masked*, *inclusion*, and *SDC* respectively. As can be seen in the figure, the inclusion case can be considered as safe or SDC depending on the decision that the automotive framework makes.

For fault vulnerability analysis throughout this paper, we compare the output of the faulty network against the golden network with DBscan clustering and IoU comparison.



(a) Fault-free network output (b) Faulty network: Masked

(c) Faulty network: Inclusion (d) Faulty network: SDC

Fig. 3: Examples of fault effect on the output of object detection network

III. Target GPU Architecture

This section provides a brief background on our target Volta GPU architecture and its resiliency features. Figure 4 shows a simplified, representative GPU architecture for NVIDIA Volta generation [7]. This architecture has an optimized design suitable for deep learning applications. The GPU architecture comprises multiple GPU Processing Clusters (GPCs) and multiple streaming multiprocessors (SMs) within each GPU. This GPU supports high-bandwidth HBM2 memory technology. This architecture contains various hardware-based mechanisms for the detection and correction of errors. In Volta GPUs, the HBM2 memory subsystem is protected by Single-Error Correcting Double-Error Detecting (SECDED) Error Correction Code (ECC). Similarly, various on-chip SRAM structures may also be protected using parity, which enables detection of bit errors, or using SECDED ECC, which enables the correction of single bit errors and detection of double bit errors. Key SRAM structures in Volta GPU are also protected by SECDED ECC, including the SM register file, L1 cache, and L2 cache.

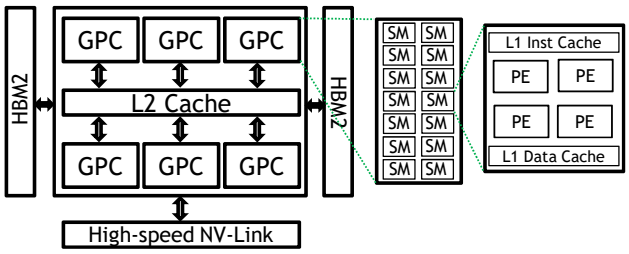


Fig. 4: Volta GPU block diagram

IV. Permanent fault injection experiments

A. Setup

For the permanent fault injection studies, we use a network from NVIDIA DriveWorks [8], which is part of the NVIDIA DRIVE software stack. The network has been trained for detecting and classifying objects in an automotive context to detect obstacles in order to plan the path and control the vehicle. The object detection DNN is one of many networks in DriveWorks that enable situational awareness of different aspects of a vehicle's surroundings. The object detection DNN model is trained to detect and classify five categories of objects: bicycles, cars, trucks, people and traffic signs. The network has 54 layers, which includes convolution, activation and pooling layers. For performing the inference on GPUs, we use the TensorRT framework developed by NVIDIA [9]. TensorRT consists of an API and a runtime system that optimizes inference on GPU architectures. We use TensorRT version 3.0.4 and compile the application using nvcc compiler from the CUDA toolkit version 9.0. Each run of the application is provided with an input of 1000 sample image batch files, which consist of realistic driving scenarios. The data set contains sample images that cover a wide variety of urban, suburban and freeway environments that include objects from the categories that the network has been trained to detect.

B. Fault Injection

We simulate the injection of permanent faults by perturbing the weights of the inference network and the input image. The perturbed values persist for the entire duration of the application run. The injected faults in the weight or image values simulate single-bit faults.

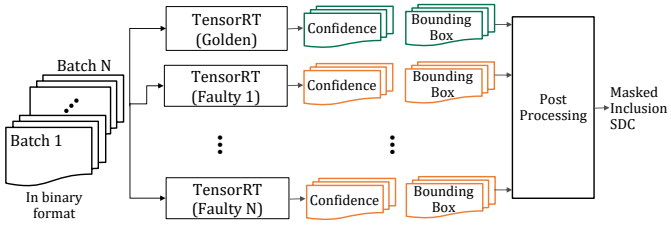


Fig. 5: Permanent fault injection methodology

When the inference is performed by the TensorRT runtime, it modifies the network layer by layer to optimize its performance on a GPU. Since we lack control over this optimization performed by TensorRT, we inject the faults in the input network model and create a faulty version of the TensorRT network for every permanent fault scenario. This methodology of fault injection is illustrated in Figure 5.

For these fault injection experiments, we make a random bit flip in one of the weights in the network, and generate a new network using the new weight parameters. We generate a ground truth inference for every input image using a fault-free network for comparing its outputs with the bounding box inference outputs generated using the faulty networks and process it using the methodology explained in Section II. We have done the fault injection experiments in weights of the network for 322 injected faults.

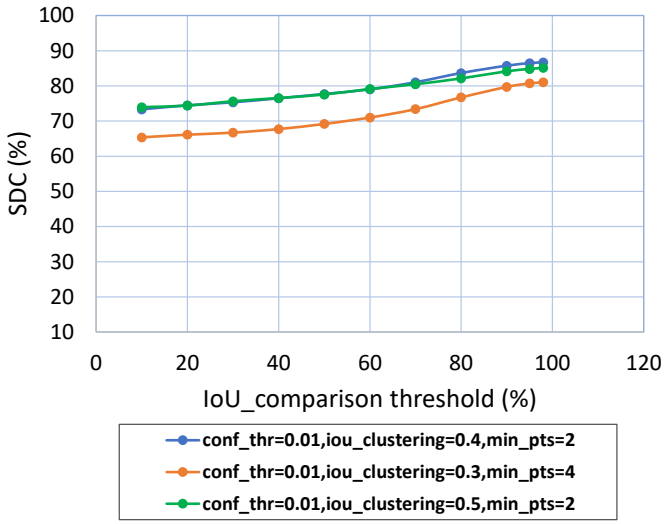
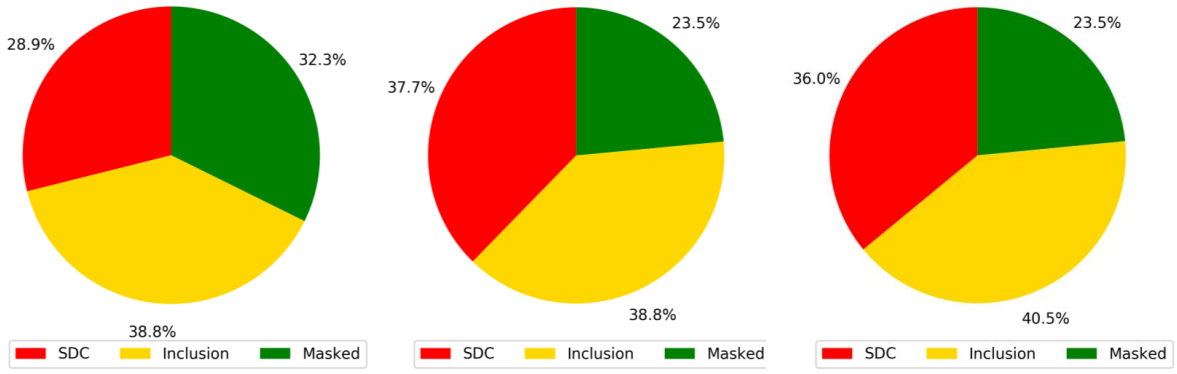


Fig. 6: Permanent fault injection: SDC vulnerability of detection network considering different thresholds for comparison and clustering parameters

C. Results

To find suitable values for the parameters in clustering and comparison algorithm, we changed their values and observed the effect on the output of networks. Changing *conf_thr*, *IoU_clustering*, and *Min_pts* parameters affects the golden network output, therefore, we find values that results in acceptable detected objects in the output of golden network. Our experiments showed that selecting the following values results in acceptable network behaviour: *conf_thr* = 0.01; *IoU_clustering* = {40%, 50%} and *Min_pts* = 2; or *IoU_clustering* = 30% and *Min_pts* = 4. To determine *IoU_threshold_comparison* threshold value, we change its value from 10% to 98% and observe how it affects the SDC rate while injecting random faults in the weights of network. Figure 6 illustrates the outcome of this experiment. Result shows the average over all fault injection experiments and images. In this experiment inclusion outcome is considered as SDC. As can be seen, increasing the value of *IoU_threshold_comparison* results in a more conservative comparison and higher SDC percentage. Depending on the value of other parameters, SDC rate is changed between 65% to 86%. In the rest of this paper, we put the *IoU_threshold_comparison* equal to 40%, as it results in a reasonable outcome for comparison of golden and faulty networks.

Using the aforementioned threshold values, Figure 7 illustrates the summary of safe, inclusion, and SDC events when faults are injected in the network weights. In general the SDC vulnerability range is 28%-37%. If inclusion is considered as unsafe then the SDC vulnerability range increases to 67%-76%. This result shows vulnerability of object detection networks to permanent faults. This indicates that



(a) IoU_clustering = 30%, Min_pts = 4 IoU_threshold_comparison = 40%

(b) IoU_clustering = 40%, Min_pts = 2 IoU_threshold_comparison = 40%

(c) IoU_clustering = 50%, Min_pts = 2 IoU_threshold_comparison = 40%

Fig. 7: Permanent fault injection experiments: SDC vulnerability of object detection network

traditional detection mechanisms should be incorporated to improve the permanent fault detection capability while running these networks.

We also injected single bit random faults in the input batches. The results indicates that if the fault is injected only in the 2 least significant bytes of the floating point input value (lower bits of mantissa), SDC is always 0. Otherwise, if we allow the fault to be injected anywhere in the input, SDC (including inclusion) is below 1.8%. As expected, the object detection network is resilient to the permanent faults in the input image.

V. Accelerated Neutron Beam Experiments

A. Setup

We investigate how transient errors impact the reliability of the object detection and classification application through accelerated neutron beam experiments. The GPUs are exposed to a neutron beam whose energy spectrum is similar to that of atmospheric neutrons at sea level. To gather substantial experimental data, we performed the same radiation experiments across two beam testing campaigns at the Weapons Neutrons Research (WNR) and the ChipIR microelectronics radiation facilities at the Los Alamos Neutron Science Center (LANSCE) and the Rutherford Appleton Laboratory respectively. The spallation neutron sources at WNR and ChipIR produce total neutron flux of up to 10^8 times higher than the flux observed at sea level. Between the two testing sites, the data collected for the object detection and classification application is the equivalent of 2000 years of exposure to terrestrial neutron flux.

We conducted the experiments at both sites using GPUs based on the Volta architecture. The experiment was setup to irradiate the entire GPU chip package for the entire duration of the experiment. Since neutrons easily penetrate the GPU boards, multiple GPU boards are aligned in the flight path of the beam. The experiments with the object detection and classification running were performed over several hours on multiple GPUs. Figure 8 shows the experiment setup at ChipIR. The same inference application (based on the object detection DNN from NVIDIA DriveWorks) and the same input image dataset used for permanent fault injection experiments were used for the beam experiments. The experiments were performed with three ECC configurations: (a) SRAM and DRAM SECDED ECC disabled, (b) DRAM SECDED ECC enabled, SRAM ECC disabled, and (c) both SRAM and DRAM ECC SECDED ECC enabled. We collected data for several hundred runs of the application for each ECC configuration. Based on the flux acceleration, the runtime of the inference application was tuned (by selecting the number of input images processed by each run) such that the chip experienced no more than a single bit flip event during each application run. Each application run processed approximately 100 sample image files. For each application run, we record the detected object bounding boxes and confidence intervals and log the occurrence of any crash events. A radiation-induced transient error can result in the following outcomes:

- **Masked:** Errors that have no effect on the program output;
- **Detectable Uncorrected Error (DUE):** Errors that can be detected by hardware or software but cannot be corrected., e.g. when the program hangs or crashes;
- **Silent Data Corruption (SDC):** Errors which result in generating an incorrect output; or
- **Inclusion:** Errors which result in a mismatch in program output which can be either acceptable or incorrect (defined in Section II-B);

For the recorded bounding box and confidence interval data, we apply the same post-processing steps described in Section II to discard boxes with low confidence values and cluster bounding boxes with significant overlap using DBScan.

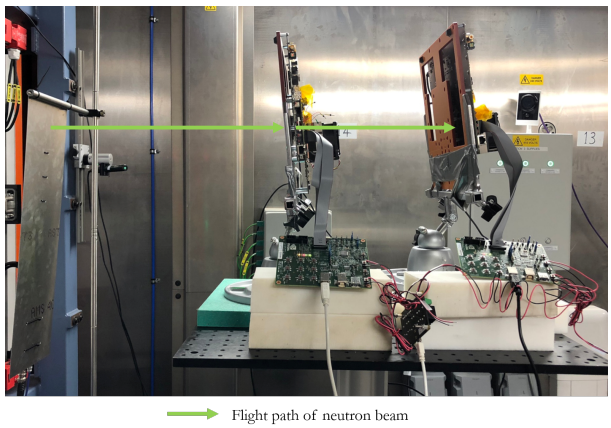


Fig. 8: Neutron beam testing: Experiment setup at ChipIR

B. SDC Analysis

To evaluate whether a transient fault has caused a SDC or inclusion outcome in the final set of predicted bounding boxes, we use the same methodology explained in Section II-B. We employ intersection over union threshold *IoU_threshold_comparison* to compare the predicted bounding boxes generated by the inference network performed under the beam with those produced by a *golden* fault-free run. We find that not all transient faults result in errors that are visible in the outputs of the application, even when SECDED protection for both SRAM and DRAM is disabled. Since the classification uses logistic regression to predict the bounding box coordinates and confidence intervals, there is some masking at the application level. Additionally, when SECDED ECC is enabled, single-bit errors are corrected in hardware and do not become visible to the software.

When both DRAM and SRAM ECC SECDED are enabled, we observed no SDC among the detection and classification outputs produced by the application. However, when SECDED ECC of the SRAM , or both SRAM and DRAM, is disabled, a subset of the application runs show incorrect detection and classification outcomes. For determining whether the outputs of the runs performed in the beam have experienced SDC, we use the same vulnerability analysis, based on using IoU thresholds described in Section II. To validate this SDC sensitivity analysis, we also compare the output bounding box coordinate predictions and confidence intervals of each run from the neutron beam experiment data with the data values from a pre-computed golden reference run with identical input images. For this validation we use a similarity threshold of 90% between the data values to determine whether the beam run contains any SDC.

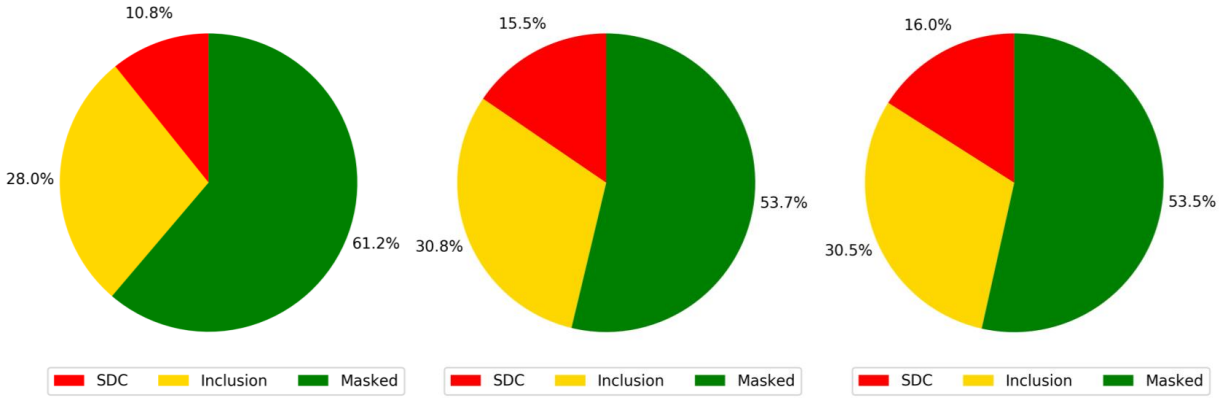
Figure 9 shows the sensitivity of the neural network-based object detection and classification to SDC for the runs performed with both DRAM and SRAM SECDED ECC disabled. On average, the occurrence of SDC leads to 14.1% of incorrect outcomes and 29.76% of inclusion SDC. The analysis reveals that on average 56.1% of the outcomes are correct despite the occurrence of the error during the run. Figure 10 shows the sensitivity of the application to SDC for the runs when SRAM SECDED ECC is disabled, but DRAM ECC is enabled. In comparison to the SDC-affected runs with both SRAM and DRAM ECC disabled, these runs show a much higher percentage (73.6%) of correct bounding box predictions and confidence intervals despite the presence of SDC. With this ECC configuration, 15% of outcomes are incorrect and 11.3% are inclusion SDC. In contrast to permanent faults, where the average incorrect outcomes are 34.2% and inclusion SDC are 39.4% when a run is affected by an error, the transient fault affected have higher percentages of correctly predicted bounding boxes. This is not altogether surprising, since the persistence of the permanent fault over the entire run affects the correctness of the detection and classification of several input image scenes. Based on these observations, it is important that hardware and software resiliency solutions for automotive platforms emphasize coverage for permanent faults.

C. Evaluation of Chip-level Protection Mechanisms: DUE and SDC FIT

Through the neutron beam experiments, we also evaluate the effectiveness of the existing protection mechanisms in the GPU architectures to transient errors. This evaluation enables the assessment of vulnerabilities in the hardware-software stack that can be improved through new architecture-level features or software-driven mechanisms for GPU-based automotive platforms. As detailed in Section III, the GPU contains SECDED or parity-based error correcting codes for detection and correction of bit errors in several SRAM structures as well as ECC protection for the DRAM memory. Additionally, the GPU architecture contains various hardware diagnostics for detecting and reporting hardware errors to the software stack.

The ISO 26262 defines the probabilistic metric for hardware failure (PMHF), which represents a calculated estimate of the rate of hazard occurrence on account of random faults. In the calculation of PMHF, the rate of occurrence λ is often expressed in units of FIT (failures per billion hours). Therefore, we compare the FIT rate for each of the detection DNN-based application for each of the configurations. For the calculation of the FIT, we aggregate the experiment data collected from both neutron beam testing sites, adjusting the FIT by the neutron flux acceleration factor based on the measured real-time flux readings. Figure 11 (a) shows the normalized SDC FIT rate for the object detection and classification application. Enabling the DRAM ECC reduces the silent data corruption rate by at least one order of magnitude. When both the DRAM and SRAM ECC were enabled, we observed no SDC events in any of the experiment runs.

Figure 11 (b) shows the normalized FIT rate for crash events that occur as a result of detected unrecoverable error events during exposure to the neutron beam. Between the configuration in which both the DRAM and SECDED SRAM ECC are disabled and the configuration in which only the SRAM ECC is disabled, there is a 45% reduction in the failure rate of the application. When both the DRAM and

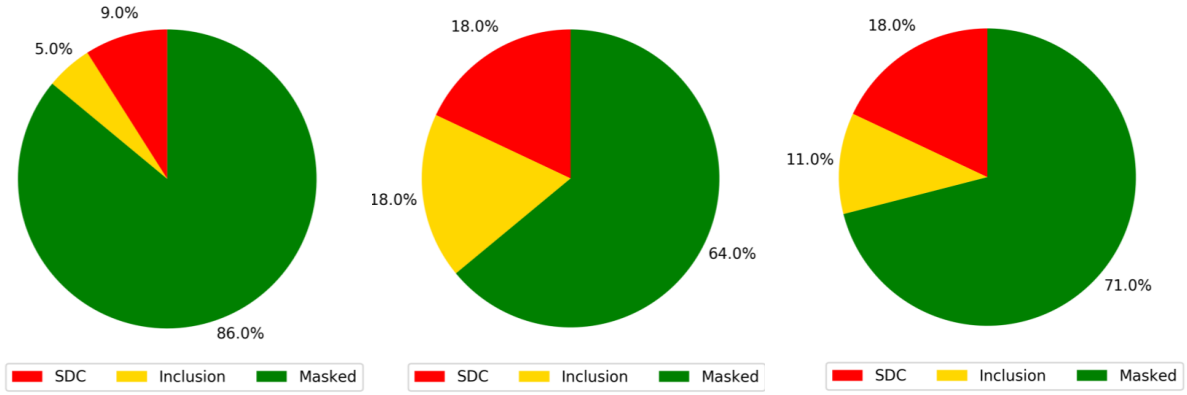


(a) IoU_clustering = 30%, Min_pts = 4 IoU_threshold_comparison = 40%

(b) IoU_clustering = 40%, Min_pts = 2 IoU_threshold_comparison = 40%

(c) IoU_clustering = 50%, Min_pts = 2 IoU_threshold_comparison = 40%

Fig. 9: Neutron beam experiments: SDC vulnerability of object detection network with DRAM and SRAM ECC *OFF*



(a) IoU_clustering = 30%, Min_pts = 4 IoU_threshold_comparison = 40%

(b) IoU_clustering = 40%, Min_pts = 2 IoU_threshold_comparison = 40%

(c) IoU_clustering = 50%, Min_pts = 2 IoU_threshold_comparison = 40%

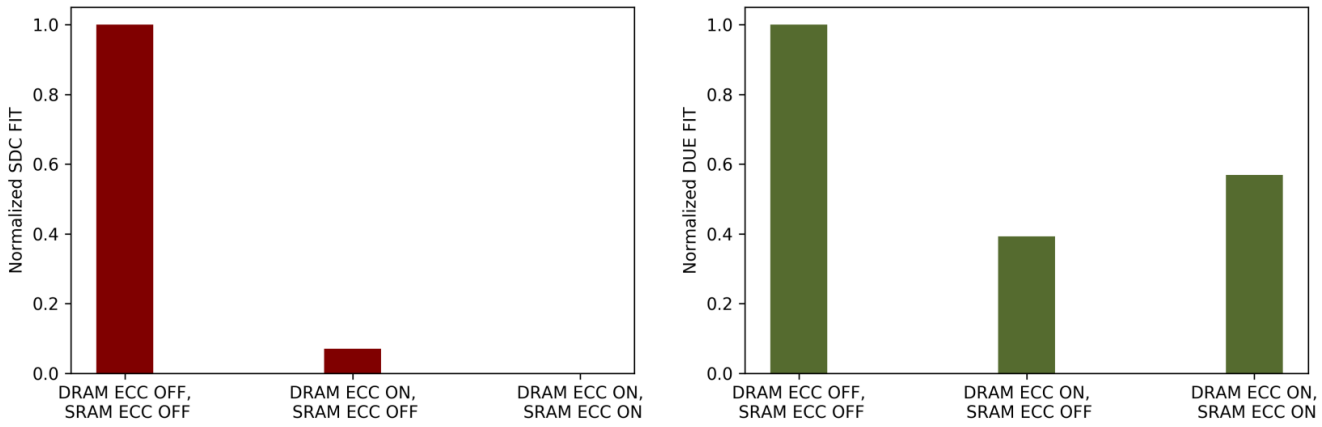
Fig. 10: Neutron beam experiments: SDC vulnerability of object detection network with DRAM ECC *ON* and SRAM ECC *OFF*

SRAM ECC are enabled, there is a modest increase in the DUE rate of the application. Part of this failure rate may be attributed to the transient fault events that strike unprotected regions of the chip, such as the flip-flops and combinational logic. However, the fact that the SDC FIT decreases while the DUE FIT increases when both SRAM and DRAM ECC is enabled suggests that some of the parity violations and multi-bit corruptions that were masked by the logistic regression algorithm (included among the 'Masked' or 'Inclusion' in Figure 10) are being detected by parity and ECC and result in program crash. Therefore, these SDC and DUE FIT results demonstrate that SECDED ECC and parity protections for the SRAM and DRAM in the Volta architecture are sufficient to meet the most stringent ASIL-D requirement imposed by the ISO 26262 for this type of convolution neural network-based object detection and classification application.

VI. Related Work

Some studies have shown that neural networks are inherently robust to noisy inputs and amenable to inexact computations. They leveraged this robustness to achieve performance and energy gains through the design of custom low-precision accelerators [10]. However in an automotive context, a deep neural network has a limited inherent fault tolerance due to the stringent safety requirements. In fact without adequate detection and mitigation mechanisms in the design, the computation using the neural networks is sometimes prone to grossly incorrect inference outcomes.

There have been prior studies that evaluated the resiliency of neural networks. Some of these studies [11] [12] were done using simpler neural network topologies and demonstrated high degree of fault tolerance to transient faults. These studies emulated transient faults using software-based fault injection.



(a) Normalized SDC FIT Rates

(b) Normalized DUE FIT Rates

Fig. 11: FIT reduction through chip-level error protection mechanisms

In contrast to modern deep neural networks, the networks used in these early studies were relatively simple. However, deep neural networks for automotive applications are often complex and require massively parallel architectures (such as GPUs) for training and inference. This complexity has a qualitative impact on resiliency.

There have been more recent efforts that seek to evaluate the resiliency of deep neural networks. Recent research [13] studied the sensitivity of different kernels in a GPU-implemented convolutional neural network. This study demonstrated that faults in kernels that are launched later in time have a higher possibility to cause error and object misprediction. A study by Li et. al. [14] evaluated the impact of soft errors on deep neural networks used for image classification and detection networks when running on specialized accelerators. Through application-level fault injection experiments, the study showed that each layer in the neural network has different sensitivity to transient faults. To study the impact of transient faults, we carried out beam testing experiments, which provide a more realistic profile of neutron-induced transient fault events. In addition, we also evaluated the resiliency of GPU-implemented object detection networks with respect to permanent faults. Permanent faults are also critical in the automotive context as they are another source for random hardware faults.

Other researchers have also performed reliability analysis of previous generations of GPUs through beam testing experiments. DeBardeleben et. al [15] and Oliveira et. al [16] performed neutron beam testing experiments on NVIDIA Fermi and Kepler GPUs running high-performance computing applications. More recent work [17] studied reliability of three generation of NVIDIA GPUs, based on Kepler, Maxwell, and Pascal architectures, through neutron beam testing while executing convolutional neural network applications. The GPU used in our beam testing and permanent fault injection studies is based on the more recent Volta architecture. The NVIDIA Xavier system-on-a-chip (SoC) [18], whose safety architecture is suitable for use in automotive applications, is also based on the Volta architecture and contains similar memory protection mechanisms as the chip used in our experiments. Therefore, while the general trend of the results on the sensitivity of convolutional neural networks to transient faults in [17] match our findings, we are also able to assess the effectiveness and limitations of these chip-level mechanisms for detecting and mitigating faults. Additionally, we perform experiments that evaluate both types of random hardware faults: transient and permanent faults. This paper to the best of our knowledge is also the first one to demonstrate the vulnerability of object detection networks that are based on logistic regression calculation of bounding box coordinates and the corresponding confidence probabilities. In much of the prior work on evaluating resiliency of deep neural networks, the occurrence of errors are determined by a simple comparison of the output values with a golden reference. However, owing to some masking by the algorithm and some tolerance in the output values of the predicted bounding box coordinates, not every value mismatch corresponds to a safety critical error. We have developed a more realistic methodology for determining the criticality of faults on the object detection networks based on the IoU metric.

VII. Conclusion

Functional safety is an essential requirement for automotive computing systems that provide capabilities ranging from driver assistance to full autonomy in path planning and control. These critical tasks are supported by convolutional neural networks that perform object detection and classification. This paper analyzes the resiliency characteristics of this class of networks to random hardware faults on GPU architectures, which includes transient and permanent fault events. We evaluate the resiliency of a detection network to transient

faults using accelerated neutron beam testing experiments. We develop a method to analyze the SDC criticality of faults, which differentiates between faults that mispredict the location of an object in an image and/or make an incorrect classification of object type from those that are masked by the logistic regression. We also use fault injection experiments to analyze the impact of permanent fault events on the object detection and classification. This establishes the higher sensitivity of the network to permanent fault events, in which a larger percentage of fault result in incorrect or mispredicted bounding boxes and confidence intervals on objects in the input images. We calculate the failure rates of the applications with the different GPU protection mechanisms to assess their effectiveness in mitigating SDC. We find that the SECDED ECC and parity protections in GPU architectures reduce the SDC FIT by at least an order of magnitude, and with all protections enabled we find no SDC during the beam experiments. The raw permanent FIT rate (in hundreds) in silicon devices is generally two orders of magnitude lower than the raw transient FIT rate (in tens of thousands) in the device SRAM and flip-flops cells. However, the detection of permanent faults is important in meeting the overall safety goals. SRAM ECC and parity can detect only a limited number of permanent faults that impair the memory cells. Automotive safety standards recommend using off-line structural tests during key-off and key-on events. Periodic structural tests for permanent faults are also recommended to meet the fault-tolerant time interval (FTTI) requirements in the safety standard. FTTI stipulates the required time (in tens of milliseconds) to detect and take a corrective action after the occurrence of a fault. Complementing the ECC and parity protections with periodic structural tests presents a viable path to mitigating risks of potential hazards due to both transient and permanent faults.

REFERENCES

[1] International Organization for Standardization, "Iso 26262 standard, road vehicles - functional safety." [Online]. Available: https://www.iso.org/standard/68383.html

[2] N. Saxena, "Keynote: Autonomous car a new driver for resilient computing and design-for-test," in *Electronics Technology Workshop*, 2016. [Online]. Available: https://nepp.nasa.gov/workshops/etw2016/talks/15WED/20160615-0930-Autonomous_Saxena-Nirmal-Saxena-Rec2016Jun16-nasaNEPP.pdf

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[4] J. Laprie, "Dependable computing and fault tolerance : Concepts and terminology," in *Twenty-Fifth International Symposium on Fault-Tolerant Computing, 1995, ' Highlights from Twenty-Five Years'.*, June 1995.

[5] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, Sep. 2005.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, pp. 226–231. [Online]. Available: http://dl.acm.org/citation.cfm?id=3001460.3001507

[7] "Nvidia tesla v100 gpu architecture," June 2017, White Paper. [Online]. Available: http://www.nvidia.com/object/volta-architecture-whitepaper.html

[8] "NVIDIA DRIVE," https://developer.nvidia.com/drive, 2019.

[9] "NVIDIA TensorRT," https://developer.nvidia.com/tensorrt, 2019.

[10] Z. Du, K. Palem, A. Lingamneni, O. Temam, Y. Chen, and C. Wu, "Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2014, pp. 201–206.

[11] V. Piuri, "Analysis of fault tolerance in artificial neural networks," *J. Parallel Distrib. Comput.*, vol. 61, no. 1, pp. 18–48, Jan. 2001. [Online]. Available: http://dx.doi.org/10.1006/jpdc.2000.1663

[12] C. Alippi, V. Piuri, and M. Sami, "Sensitivity to errors in artificial neural networks: a behavioral approach," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 42, no. 6, pp. 358–361, June 1995.

[13] R. Bramley, "Functional Saftey and the GPU," http://on-demand.gputechconf.com/gtc/2017/presentation/s7372-richard-bramley-functional-safety.pdf, GTC 2017.

[14] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. New York, NY, USA: ACM, 2017, pp. 8:1–8:12. [Online]. Available: http://doi.acm.org/10.1145/3126908.3126964

[15] N. DeBardeleben, S. Blanchard, L. Monroe, P. Romero, D. Grunau, C. Idler, and C. Wright, "Gpu behavior on a large hpc cluster," in *Euro-Par 2013: Parallel Processing Workshops*, D. an Mey, M. Alexander, P. Bientinesi, M. Cannataro, C. Clauss, A. Costan, G. Kecskemeti, C. Morin, L. Ricci, J. Sahuquillo, M. Schulz, V. Scarano, S. L. Scott, and J. Weidendorfer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 680–689.

[16] D. Oliveira, L. Pilla, M. Hanzich, V. Fratin, F. Fernandes, C. Lunardi, J. Cela, P. Navaux, L. Carro, and P. Rech, "Radiation-induced error criticality in modern hpc parallel accelerators," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2017, pp. 577–588.

[17] F. Santos, P. 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*, pp. 1–15, 2018.

[18] "Introducing Xavier, the NVIDIA AI Supercomputer for the Future of Autonomous Transportation," https://blogs.nvidia.com/blog/2016/09/28/xavier, 2019.