UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUCAS KLEIN DRAGHETTI

Detecting Errors in Convolutional Neural Networks Using Inter Frame Spatio-Temporal Correlation

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Engineering

Advisor: Prof. Dr. Paolo Rech

Porto Alegre
July 2019

“Life is an improvisation.
You have no ideas what’s going to happen next
and you are mostly making things up as you go along.”
— Stephen Colbert

ACKNOWLEDGEMENTS

I wish to thank my family, for supporting and helping me through all my life.

I wish to thank my girlfriend, Luana Freitas, for the patience and support whenever I needed.

I wish to thank my colleagues and all my friends, for helping me when i needed and making my whole graduation more fun.

I wish to thank my advisor, Professor Paolo Rech, for his patience and guidance provided during several years of my graduation.

I wish to thank all the teachers, for the quality lessons and knowledge sharing.

ABSTRACT

Object detection, a critical feature for autonomous vehicles, is performed today using Convolutional Neural Networks (CNNs). Errors in a CNN execution can modify the way the vehicle sense the surrounding environment, potentially causing accidents or unexpected behaviors. The high computational requirements of CNNs combined with the need to perform detection in real-time allow little margin for implementing error detection.

In this project, an extremely efficient error detection solution for radiation induced errors in CNN is presented based on the observation that, in the absence of errors, the differences between the input frames and the detection provided by the CNN should be strictly correlated. In other words, if the image between two subsequent frames does not change significantly, the detection should also be very similar. Similarly, if the detection varies considerably from a frame to the next, then the input image should also have been different. Whenever input images and output detection don't correlate, it is possible to detect an error. After formalizing and evaluating the inter-frame and output correlation thresholds, the detection strategy is implemented and validated, utilizing data from previous radiation experiments. Exploiting the intrinsic efficiency in processing images of devices used to execute CNNs, up to 80% of errors are detected, while adding low overhead.

The same error detection solution is then proposed to detect false positives in fault-free CNN executions. This strategy is also implemented and validated, utilizing ground truth annotations and fault-free CNN executions. For this application, 9% of the false positives can be detected reliably. A deeper analysis shows that more false positives can be detected, if a certain percentage of wrong detections is accepted.

Keywords: Convolutional Neural Networks. Object Detection. Fault Tolerance. Error Detection.

Detectando Erros em Redes Neurais Convolucionais Usando Correlação Espaço-temporal Entre Quadros

RESUMO

Detecção de objetos, um recurso crítico para veículos autônomos, é realizada hoje usando Redes Neurais de Convolução (CNNs). Erros em uma execução de uma CNN podem modificar a maneira como o veículo detecta o ambiente ao redor, potencialmente causando acidentes ou comportamentos inesperados. Os altos requisitos computacionais de CNNs, combinados com a necessidade de realizar a detecção em tempo real, permitem pouca margem para a implementação da detecção de erros.

Neste projeto, uma solução de detecção de erros extremamente eficiente para erros induzidos por radiação em CNN é apresentada com base na observação de que, na ausência de erros, as diferenças entre os quadros de entrada e a detecção fornecida pela CNN devem ser estritamente correlacionadas. Em outras palavras, se a imagem entre dois quadros subseqüentes não mudar significativamente, a detecção também deve ser muito semelhante. Da mesma forma, se a detecção variar consideravelmente de um quadro para outro, a imagem de entrada também deve ter sido diferente. Sempre que as imagens de entrada e a detecção de saída não se correlacionarem, é possível detectar um erro. Depois de formalizar e avaliar os limiares de correlação entre-quadros e de saída, a estratégia de detecção é implementada e validada, utilizando dados de experimentos anteriores de radiação. Explorando a eficiência intrínseca no processamento de imagens de dispositivos usados para executar CNNs, até 80% de erros são detectados, adicionando também pouca sobrecarga. A mesma solução de detecção de erro é então proposta para detectar falsos positivos em execuções de CNN sem falhas. Esta estratégia também é implementada e validada, utilizando anotações de valores de referência e execuções de CNN sem falhas. Para esta aplicação, 9 % dos falsos positivos podem ser detectados de forma confiável. Uma análise mais profunda mostra que mais falsos positivos podem ser detectados, se uma certa porcentagem de detecções incorretas for aceita.

Palavras-chave: Redes Neurais Convolucionais, Detecção de Objetos, Tolerância a Falhas, Detecção de Erros.

LIST OF ABBREVIATIONS AND ACRONYMS

CNN	Convolutional Neural Network
DNN	Deep Neural Network
SIMT	Single Instruction Multiple Thread
SDC	Silent Data Corruption
DUE	Detected Unrecoverrable Error
YOLO	You Only Look Once
R-CNN	Region-based Convolutional Neural Network
COCO	Common Objects in Context
mAP	Mean Average Precision
IoU	Intersection over Union
PASCAL VOC	PASCAL Visual Object Classes
BB	Bounding Box
RAL	Rutherford Appleton Laboratory
LANSCE	Los Alamos Neutron Science Center
ECC	Error Correction Code
CPU	Central Processing Unit
GPU	Graphics Processing Unit
SSD	Sum of Squared Differences

LIST OF FIGURES

Figure 3.1 Percentages of experimentally observed tolerable and critical errors in CNNs....19

Figure 4.1 Error Detection Idea 21

Figure 5.1 Precision and SSD calculated over the Caltech dataset in a fault-free environment. 26
Figure 5.2 Recall and SSD calculated over the Caltech dataset in a fault-free environment. 27
Figure 5.3 Precision and SSD calculated over the experimentally observed errors....29
Figure 5.4 Recall and SSD calculated over the experimentally observed errors. 30
Figure 5.5 Precision and SSD calculated over the KITTI ground truth annotations. 32
Figure 5.6 Recall and SSD calculated over the KITTI ground truth annotations. 32
Figure 5.7 CNN Error Distribution 34
Figure 5.8 Precision and SSD calculated over the KITTI dataset....35
Figure 5.9 Recall and SSD calculated over the KITTI dataset. 36
Figure 5.10 Analysis of the error detections for each region. 36

CONTENTS

1 INTRODUCTION....10
2 OBJECT DETECTION AND CONVOLUTIONAL NEURAL NETWORKS 12
2.1 Object Detection....12
2.2 Metrics....12
2.2.1 Intersection Over Union....12
2.2.2 Precision and Recall....13
2.2.3 Mean Average Precision....13
2.3 Datasets....13
2.3.1 Caltech Pedestrian Dataset....14
2.3.2 KITTI Vision Benchmark Suite....14
2.4 Convolutional Neural Networks....14
2.4.1 R-CNN....15
2.4.2 Fast R-CNN....15
2.4.3 Faster R-CNN....16
2.4.4 YOLO....16
3 TRANSIENT FAULTS IN OBJECT DETECTION....17
3.1 Detected Unrecoverable Error....17
3.2 Silent Data Corruption....17
3.3 SDC Criticality....17
3.4 Experimental Data Samples....18
4 METHODOLOGY....20
4.1 Basic Idea....20
4.2 Implementation Details....21
4.3 Overhead....23
5 RESULTS....25
5.1 Validation for Detection of Errors from Radiation....25
5.2 Detection of Errors from Radiation....28
5.3 Validation for Detection of Errors in Fault-Free Executions....31
5.3.1 CNN Error Distribution....33
5.4 Detection of Errors in Fault-Free Executions....33
6 CONCLUSION....38
REFERENCES....39

1 INTRODUCTION

Autonomous vehicles are about to change the transportation systems, the automotive and military markets, and burst deep space exploration. However, while autonomous cars are expected to reduce of two-three orders of magnitude the number of accidents (SAXENA, 2016; FERNANDES et al., 2016), current self-driving systems are not yet ready to be employed in large-scale. One of the main issues related with the software and hardware systems for autonomous vehicles is to qualify them for the strict ISO26262 reliability requirements (DONGARRA; MEUER; STROHMAIER, 2015). In particular, object detection, a critical task in autonomous vehicles, has been shown to be vulnerable to transient faults (FERNANDES et al., 2016) and to be responsible for the majority of reported accidents in current self-driving cars prototypes (BANERJEE et al., 2018). Also, false positives were reported as the cause of the largest problems in advanced-driver assistance systems, as they may cause the system to apply the brakes needlessly (HUVAL et al., 2015). Intuitively, when the system detects non-existent objects or doesn't detect existent objects, it will behave differently from expected. The vehicle may stop unnecessarily or, worse, it may drive into something or someone, causing dangerous accidents.

Today's object detection is performed through *Convolutional* Neural Networks, a new class of Deep Neural Networks that extract features to detect and classify objects. *Convolution*, combined with a very deep topology, makes CNNs significantly different from traditional neural networks. Previous studies in the late 1990's that showed inherent reliability for Neural Networks may not, then, be applied to current object detection frameworks. Additionally, the devices required to satisfy to the high computational demand of CNNs are built with parallel or heterogeneous architectures. These architectures have been shown to have some inherent vulnerabilities as they tend to propagate a single fault to multiple computing units (RECH et al., 2013; OLIVEIRA et al., 2014), invalidating the single-fault model used in the past.

The high number of layers and the high computational demand of each layer impose the use of parallel and performant devices to execute CNN in real-time. This requirements makes the design of fault-tolerant techniques for CNNs extremely challenging for two main reasons: (1) traditional error detection solutions based on replication or redundancy are inapplicable because of their high computing or cost overhead and (2) parallel architectures have been found to spread a single fault to multiple computing elements (OLIVEIRA et al., 2017), significantly modifying the fault model and, then, the

impact of faults in CNN execution, increasing the need of error detection strategies.

We want to use all the information at hand when detecting the object to propose an efficient and effective error detection solution. CNNs process each frame independently of other frames, but in reality, each frame has a temporal correlation with the preceding one. We found that not only the input frames but also the output from the CNN, containing the detections made, correlate with each other. We use both the input and output information to detect errors in a frame as it is processed. After each frame is processed, we measure the difference between its resulting Bounding Boxes (BB, i.e., potential object) from the CNN and the resulting Bounding Boxes from the previous frame using precision and recall scores. We also measure the difference between the input image and the preceding image using the Sum of Squared Differences method. If the system is functioning correctly, drastic changes in the output detection are justified by changes in the input image. If that is not the case, we detect an error.

The same idea is applied firstly with the goal of detecting radiation induced errrors and secondly with the goal of detecting false posives in fault-free CNN executions.

2 OBJECT DETECTION AND CONVOLUTIONAL NEURAL NETWORKS

2.1 Object Detection

Object Detection is the act of processing images and detecting the bounding box (position and size) of an object and determining its class. There are many applications for it: character detection and recognition, face detection, self driving cars and many more. The classes detected vary depending on the application and environment. For example, self driving cars will need to detect most objects on the street, like other cars, pedestrians, cyclists, horses and more, while face detection applications only need to detect human faces.

Many different algorithms were created to try and detect objects reliably. To measure their effectiveness, many datasets were made, with different classes, formats and technologies. These datasets provide a large amount of images or videos and descriptions of the objects (usually position, size and class) so that the algorithms can be trained and tested. Some datasets, like Common Objects in Context (COCO), are very broad, and present many different classes to detect. Other, like Caltech Pedestrian Dataset, focus only on urban videos, and only provide annotations for the pedestrian class.

2.2 Metrics

To compare the algorithm's detection with the ground truth provided by the dataset, it is necessary to use some specific metrics, the most common one being mean Average Precision (mAP). To calculate the mAP, it is necessary to use the Intersection over Union and Precision and Recall scores.

2.2.1 Intersection Over Union

Intersection over Union (IoU) is the most common way to quantify the similarities between two bounding boxes. Its value is equal to the area of the two bounding boxes' intersection divided by their union. Thus, perfectly matching bounding boxes will have a score equal to 1 (or 100%). In the literature, it is common to consider any IoU value above 0.5 as a succesful detection (FAWCETT, 2006).

2.2.2 Precision and Recall

Succesful detections are classified as True Positives, while incorrect detections are classified as False Positives and undetected objects represent a False Negative. The Precision score will be the fraction of the detections that are correct, or True Positives/(True Positives + False Positives). The Recall score will be the fraction of objects that were correctly detected, or True Positives/(True Positives + False Negatives).

In object detection, the precision represents how good the detected bounding boxes are, i. e., the ratio between the amount of objects correctly detected and the total amount of detections. The recall represents how complete the detection was, i. e., the ratio between the amount of objects correctly detected and the total amount of objects.

2.2.3 Mean Average Precision

Most algorithms can adjust a detection threshold and make a trade-off between Precision and Recall. Lower detection thresholds will provide a higher recall as more objects are detected, but will also provide a lower precision, as false positives are potentially added. It is possible, then, to plot a precision-recall curve, where the Precision is a function of the Recall. The Average Precision will be the total area under the Precision-Recall curve. To get the mAP, the mean from the Average Precision from the complete dataset is calculated.

2.3 Datasets

There is a wide variety of datasets with ground truth annotations for object detection available. Since this project uses temporal correlation, the first key factor is that the dataset used must be continuous (e.g., A video). A consideration was also made in relation to the relevance of possible practical uses of the project's method. Since self-driving cars are the main motivation, datasets that contain images from the perspective of a camera in a moving car were selected.

2.3.1 Caltech Pedestrian Dataset

The Caltech Pedestrian Dataset consists of approximately 10 hours of video, with resolution 640x480 and frame rate 30Hz. It contains multiple segments of video taken from a moving vehicle driving in a urban environment. Ground-truth annotations for pedestrians in approximately 250,000 frames were made, totalling 350,000 bounding boxes and 2,300 unique pedestrians.

2.3.2 KITTI Vision Benchmark Suite

The KITTI Vision Benchmark Suite consists of several separate datasets with different characteristics. These datasets contain data gathered by an autonomous driving platform of the author's property, which is equipped with two high-resolution color and grayscale video cameras, a Velodyne laser scanner and a GPS localization system.

In this project, the only dataset from the KITTI Vision Benchmark Suite used was the object tracking benchmark from 2012, which consists of 50 sequences of 10Hz videos with ground-truth annotations for 8 different classes. It also contains stereo, laser and map information, as well as annotations for 3D bounding boxes. These features are not considered in the scope of this project.

2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of Deep Neural Networks (DNNs) that efficiently performs object classification and detection. CNNs are extensively used in today's self-driving vehicles to detect objects in real-time (REDMON et al., 2015). The main difference from CNNs in relation to other DNNs is the presence of Convolutional layers in their design. Normally, in fully connected layers, each neuron has connections to the whole input matrix. In contrast, convolutional layers' neurons only have connections to a small area of the input matrix each. This is done by using a set of learnable filters as the layers' weights. The filters are then convolved with the input during the forward pass, creating a spatial dependency between the input and the output, which is absent from fully connected layers. It is worth noting that convolution can be effectively mapped to matrix multiplication, making CNNs very efficient when executed

on GPUs.

To evaluate the efficacy of the proposed errors detection strategy, two modern frameworks are considered: You Only Look Once (YOLO) (REDMON et al., 2015) and Faster Region-based Convolutional Neural Network (Faster R-CNN) (REN et al., 2015), both sharing the main characteristic of a generic CNN.

2.4.1 R-CNN

Region-based Convolutional Network (R-CNN) is the predecessor of Fast R-CNN, which is itself the predecessor of Faster R-CNN. R-CNN is an object detection system that consists of three modules. The first module uses the selective search method to generate category-independent region proposals. The second module is a CNN that processes the proposed regions, generating a fixed-lenght feature vector for each of them. The third module a set of support-vector machines that will classify each region based on the features extracted from the CNN.

R-CNN was very innovative and outperformed all popular object detection methods at the time of its creation. On the object detection dataset PASCAL Visual Object Classes (PASCAL VOC) 2012, it achieved a mAP of 53.3%, an improvement of more than 30% relative to the previous best result.

2.4.2 Fast R-CNN

Fast R-CNN advanced the state-of-the-art by using a similar idea to the existing R-CNN. The main difference is that the neural network processes the whole image at once to generate a feature map, instead of multiple regions separately. Then, a region of interest pooling layer converts these feature maps into fixed-size feature maps, which are then mapped to feature vector by fully connected layers. After one more fully connected layer, the softmax probabilities and bounding-box regressions are generated.

Fast R-CNN improved on R-CNN by achieving higher mAP on PASCAL VOC 2012 (65.7%), 9 times faster training and 213 times faster testing.

2.4.3 Faster R-CNN

Faster R-CNN introduces a Region Proposal Network (RPN), an alternative to the region proposal algorithm from Fast R-CNN. An RPN is convolutional network that can be trained to generate region proposals used by Fast R-CNN. RPN and Fast R-CNN can share convolutional features, reducing drastically the processing time of the whole system. Faster R-CNN's architecture is composed of 13 convolutional layers, shared between the RPN and the Fast R-CNN, followed by 3 fully connected layers.

Faster R-CNN achieved a frame rate of 5fps and mAP of 70.4% on PASCAL VOC 2012.

2.4.4 YOLO

You Only Look Once does not use region proposal algorithms at all. Instead, it processes the whole image only once (hence the name of the CNN). The system divides the image in an SxS grid, and each grid cell predicts a certain number of bounding boxes, their class probabilities and confidence. YOLO's architecture is composed of 24 convolutional layers followed by 2 fully connected layers.

YOLO achieved a lower mAP score compared to other state-of-the-art systems,reaching only 63,4% on PASCAL VOC 2007. Still, YOLO is very relevant as it has much higher speed than its competitors, achieving a frame rate of 45fps.

3 TRANSIENT FAULTS IN OBJECT DETECTION

Atmospheric neutrons, ionizing particles, voltage/temperature/process variations, and other interferences may perturb a transistor's state, generating bit-flips in memory or current spikes in logic circuits that, if latched, lead to an error. A transient error leads to: (1) no effect on the program output (i.e., the fault is masked, or the corrupted data is not used), (2) a Detected Unrecoverable Error (DUE), or (3) a Silent Data Corruption (SDC). Radiation-induced events are particularly critical as they have been found to dominate modern devices error rate (BAUMANN, 2005).

3.1 Detected Unrecoverable Error

DUEs are errors that keep the system from running in a normal state. Usually, this kind of error leads to a program crash or system hang. Most times, information will be lost, and the program will have to be re-run.

3.2 Silent Data Corruption

SDCs are errors where the program continues to run normally, but the output of the program is incorrect. In the case of Object Detection CNNs, where the output is a list of bounding boxes with their respective class probabilities, SDCs can cause many kinds of incorrectnesses on the output, such as missing bounding boxes, extra bounding boxes or bounding boxes with incorrect size and position.

3.3 SDC Criticality

SDCs can be considered critical or not depending on the practical consequences of the error caused. A critical error is an error that affects the output in a significant manner. A non-critical error is an error that affect unimportant parts of the output, making it insignificant. Examples of possibly non-critical errors are modifications in Least Significant Bits in the output or changes in unused parts of the output.

Not all the SDCs are critical for object detection. SDCs that modify the probability in such a way that does not impact an object's rank or changes the coordinates of a low-

probability bounding box are not considered critical. Additionally, an SDC that changes the bounding box coordinates but still allows a sufficiently good detection can also be considered as tolerable. According to Fawcett et al. at, detection is useful when the BB covers at least 50% of the object area.

3.4 Experimental Data Samples

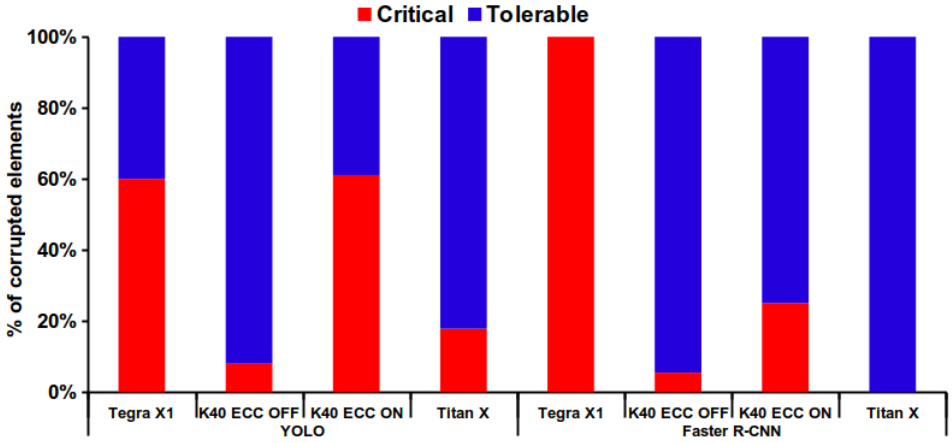
To justify and validate the proposed error detection strategy, a set of errors are used. These errors were observed in recently performed accelerated neutron beam experiments at the ChipIR facility of the Rutherford Appleton Laboratory (RAL) in Didcot, UK, and at the Los Alamos Neutron Science Center (LANSCE) facility. The experimental data has been presented in (Santos et al., 2018) and is made available by authors in a public repository (UFRGSFCAROL, 2017). The experimental nature of data ensures that the effectiveness of the proposed object detection strategy is not biased on a specific or simplified error model. Thanks to the accelerated factor of neutron beams, the available data is representative of more than 110,000 years of natural exposure, ensuring a wide sample of the possible neutron effects on CNN execution.

The available data refer to YOLO and Faster RCNN executed on three NVIDIA GPUs architectures, Tesla K40, Tegra X1, and Titan X. It is also possible to compare data obtained with Error Correction Code (ECC) enabled and disabled for the Tesla K40. The fact of considering different architectures, different transistor layout, and ECC, widens the applicability of this project's strategy and increases the confidence in its validation.

The Tesla K40 supports the $Kepler$ ISA and is fabricated in 28nm standard CMOS technology. The register files shared memory, and caches are protected by ECC, while the read-only data cache is parity protected. We can then compare data obtained with ECC enabled and disabled for the Tesla K40. The Tegra X1 is an embedded version of the $Maxwell$ GPU, and includes an ARM quad-core CPU. It is fabricated in 20nm standard CMOS technology. The third microarchitecture Titan X is designed with the $Pascal$ architecture, in 16nm FinFET technology. The fact of considering different architectures, different transistor layout, and ECC, widen the applicability of our strategy and increases the confidence in its validation.

Figure 3.1 shows the percentage of critical errors (i.e., detection is significantly modified) and tolerable errors (i.e., detection is sufficiently good) observed during the radiation experiments. The comparison of the error rates between devices and algorithms

Figure 3.1: Percentages of experimentally observed tolerable and critical errors in CNNs.



Source: The Author

is presented in (Santos et al., 2018).

For all the tested configurations but the Faster R-CNN executed on the Tegra X1, a significant number of the radiation-induced errors that propagate to the output are tolerable. The percentage of critical SDCs is much lower for Faster R-CNN than for YOLO, independent of the architecture. For YOLO, the portion of critical errors is 8% for an unhardened Tesla K40, 61% for the Tesla K40 with ECC ON, and 18% on the Titan X. For Faster R-CNN, the critical SDCs are 5% for an unhardened Tesla K40 and 25% for the Tesla K40 with ECC ON. On the Titan X, for Faster R-CNN, none of the hundreds of observed SDCs had an impact on detection. Faster R-CNN is less prone to critical errors as it performs detections combining several overlapping regions and it describes the object more reliably than YOLO (details in (Santos et al., 2018)). Tegra X1 has only critical errors for Faster R-CNN as the complexity of the CNN makes it necessary to overload the small Tegra X1, making any error very critical. Finally, a valuable insight that derives from the comparison of Tesla K40 with ECC ON and OFF is that ECC increases the percentage of critical errors (but eventually reduces the overall number of errors). Unfortunately, ECC is not effective in masking critical errors as these errors are typically caused by faults in unprotected resources like internal queues, scheduler, etc. (detailed in (Santos et al., 2018)). It is not safe to rely on ECC to improve the reliability of CNN. Novel and smart error detection strategies are necessary.

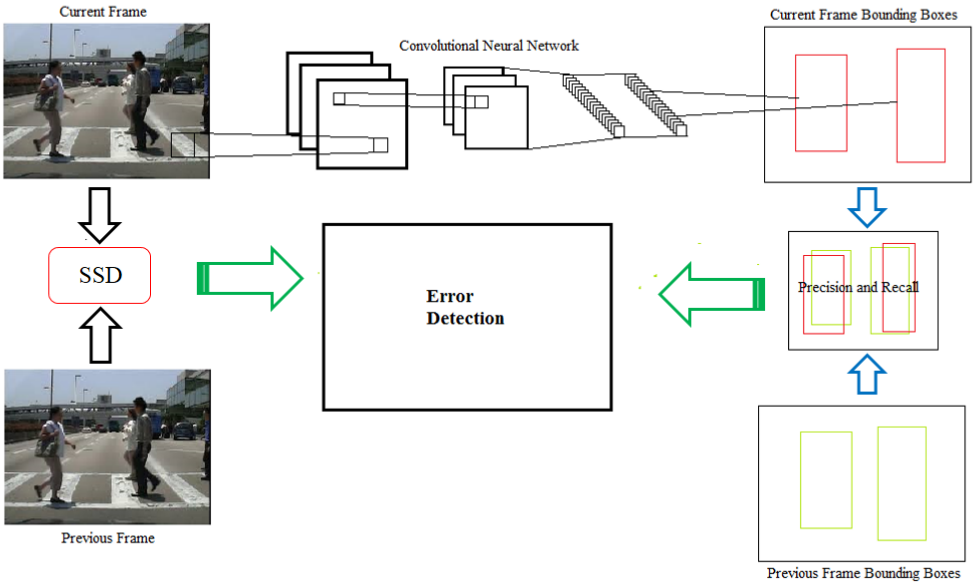
4 METHODOLOGY

4.1 Basic Idea

CNNs analyze each frame as a new set of data, without using any correlation with the previously analyzed frames. It is possible to take advantage of the strong correlation between frames and the correspondent detection to identify possible execution errors and incorrect detections. The idea of this project is based on the fact that CNNs are deterministic systems, that is, they will always have the same output if given the same input. If two consecutive frames from a video are exactly the same, the output will also be exactly the same. A necessary condition for object detection to be different in two images is that the images themselves are different. Of course, the scenario where both images are exactly the same is unrealistic in a practical implementation of object detection. However, it is still possible to claim that if nothing moves on a video, the frames will be very similar and, then, the outputs of the CNN will also be very similar. On the other hand, if all the objects in a scene move a lot, both the detection and the input image will change significantly. The goal is to demonstrate that the correlation between the input variation and the output variation can be fruitfully exploited to detect errors. In other words, if no object in the scene moves but the output of CNN is entirely different from the previous output, there is a very high chance that an error has occurred during detection.

The difference between two images is quantified by the amount of change in the color and the shape of any form present in it. As for Bounding Boxes, the differences in their sizes and positions are considered. In this project, the algorithm used to compare images will be the Sum of Squared Differences (SSD), a standard image comparison algorithm. This algorithm compares two images pixel per pixel. Every color and shape difference between the two images will be sensed by the SSD. While there may be other image processing algorithms more suitable than SSD, SSD was chosen just to have a preliminary validate the proposed idea. The efficacy comparison of different algorithms is left as future work. As for CNN, SSD (and most image processing algorithms) exploits the GPU Single Instruction, Multiple Threads (SIMT) architecture very efficiently. As such, SSD will execute very efficiently in the device that runs the CNN, guaranteeing low computational overhead of this strategy. To compare the detection of a frame with the detection computed for the previous frame, the Precision and Recall is measured between the BBs of the two outputs. With these metrics, it is possible to identify if the detected

Figure 4.1: Error Detection Idea



Source: The Author

objects have the same area, shape, and position as in the last computed frame and if some boxes are absent, moved or even if there are newly detected objects in the processed image.

Figure 4.1 illustrates the basic idea for this method. Using the ideas mentioned above applied to the CNN executions on the Caltech dataset, it was found out that some combinations of variations in the input and the output can never occur. Then, data from previous radiation experiments can be used to see which errors can be identified and which ones cannot. In a second part of the project, the same basic ideas are applied to the KITTI dataset ground truth annotations, finding out the combinations that could never occur in a perfect detection scenario. Then, data from fault-free CNN executions on the KITTI dataset are analyzed, with the main goal of detecting false positives generated by the CNN.

4.2 Implementation Details

YOLO and Faster R-CNN are the two CNNs considered. YOLO is designed to detect objects in real time, while Faster R-CNN is a high precision framework created using Python and cuDNN libraries. They process images given as input and return a

list of BBs and the respective confidence associated with each of them as output. Each frame is examined separately. It is worth noting that, since the frames are treated in a chronological order, whenever a frame is processed, the output from the preceding frame is already available.

Firstly, the difference between the current frame and the previous one is found using SSD. Since a big part of the image usually does not have relevant objects, only the frame sub-regions contained in the union of the BBs detected in the current and previous frame are examined. This is necessary to avoid differences in the landscape like clouds moving, billboards flashing, etc., that do not participate in object detection to bias the comparison. The union of the BBs is considered, as it is not known a priori which of the two detections contains an error (if any). Then, SSD algorithm is applied in these sub-images and they are normalized according to the BBs size.

Secondly, the outputs of the CNN from the current and previous frames are compared. Both the precision and recall are calculated, comparing the BBs detected in the current frame with the BBs in the previously computed frame. In the image processing community, a threshold of IoU >= 0.5 is used to trigger the reduction of precision or recall. In other words, if at least 50% of the object area is detected, the detection is considered successful. Setting a simple threshold to precision and recall function is not enough for the purposes of this project, as detecting smaller movements is needed as well. Several different thresholds were chosen and weighted to refine the results. When analyzing detections that remain the same across two frames, the expectation is that the input images also to remain the same. Similarly, for worse Bounding Box matches, the input images should show more differences.

$$p = \sum_{i \in I} \frac{prec(i)}{|I|}$$

Here, p is the final precision value used for a detection, considering the range of detection thresholds. I is the list of detection thresholds utilized. $prec(i)$ is the precision calculated for a detection, with an detection threshold i. The same calculation is used for the recall.

For the experiments on the Caltech dataset, the detection thresholds are $I = (0.5, 0.5 + k, 0.5 + 2k, \cdots, 1)$, with k being the smallest step possible. Hence, all thresholds from IoU >= 0.5, the standard detection threshold, to IoU >= 1, the threshold where the boxes match perfectly, are considered. A linear function was used to simulate these

thresholds, simplifying the implementation of the calculations. Since the KIITI dataset has a significantly lower frame rate, the detection thresholds considered are $I = (0.3, 0.3+ k, 0.3 + 2k, \cdots, 1)$ for the experiments utilizing it. This adjustment was made to account for the fact that objects tend to move greater distances from frame to frame when there is a lower frame rate (KITTI has a 10Hz frame rate, compared to Caltech's 30Hz). With a lower threshold, it is possible to detect bigger movements, as bounding box matches will be facilitated.

Some Neural Network 'glitches' were also observed in some detections, detecting objects that are not there. Usually, when this happens, the CNN outputs a confidence value that is above the detection thresholds but is lower than the average detection. To account for this, the Bounding Box matches are weighted according to their confidence, making the error detection method more robust.

$$prec(i) = \frac{\sum_{c \in T} c}{\sum_{c \in T} c + \sum_{c \in F} c}$$

Here, T is the list of confidences for all the detections considered true positives in the bounding box comparison with the given detection threshold i. F is the list of confidences considered false positives. For the recall calculations, the only difference is the consideration of false negatives instead os false positives.

Note that the input and output comparisons can be made in parallel, as they do not depend on each other.

4.3 Overhead

In real-time applications, the execution time of an algorithm is crucial to its correct and optimized functioning. Even though researchers suggest there are ways to reduce CNN computational costs, it is still an issue for some application requirements [15]. This method adds very little overhead to a complete system. A simple CPU-only test was made, using YOLO's CPU version and a CPU version of our algorithm. In this test, the overhead measured was less than 5%. The computational bottleneck of the strategy is the image comparison, which can be efficiently computed with a single instruction in SIMT architectures used in GPUs, potentially reducing the overhead even more when transitioning to a GPU version. The whole images are also not compared, but just the regions covered by the BBs. The time complexity of this comparison is linear (i.e., O(n))

relative to the BB sizes. This means that the method also scales well for higher resolution images.

5 RESULTS

5.1 Validation for Detection of Errors from Radiation

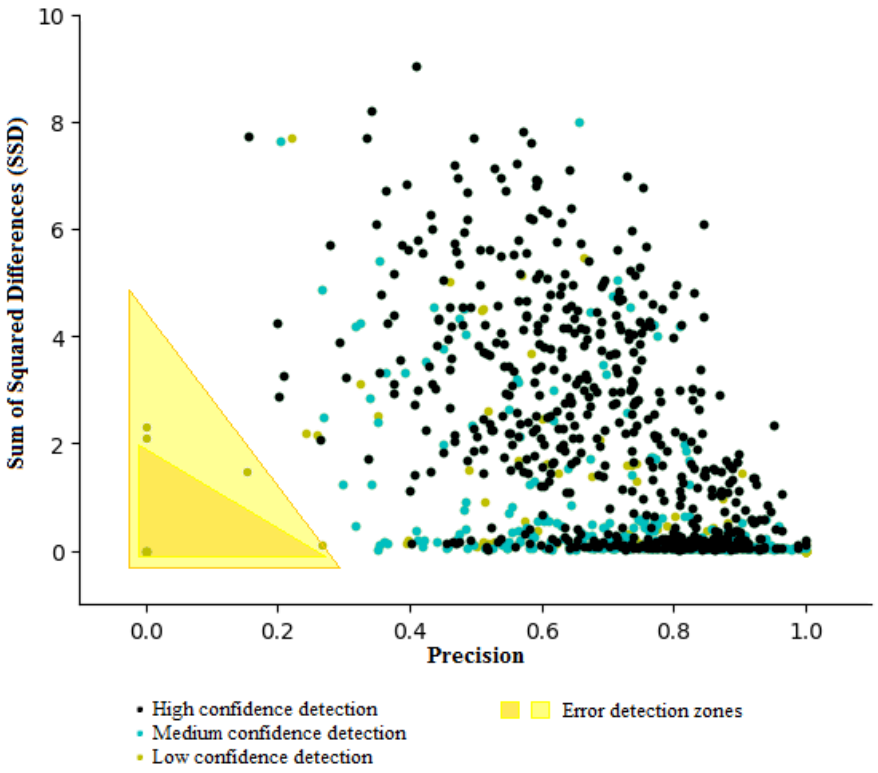
To validate the method, the CNN execution is first considered in a fault-free environment. Each frame and the correspondent object detection are compared with the previous one. The goal is to identify a correlation between the difference in two consecutive frames and the correspondent difference between the detected Bounding Boxes in the two images. If some unlikely or impossible correlations are found in the whole dataset, it will be used to identify possible errors during operation. The dataset used was the Caltech dataset.

To visualize the correlation between the differences of the input frames with the differences in the correspondent detection, the SSD of two consecutive input frames is plotted against the precision and, separately, the recall of the detected Bbs.

Figure 5.1 and Figure 5.2 show the correlation between the SSD of two consecutive frames with the precision and recall resulting from the comparison of the BBs detected in the two frames, respectively. Each point in the figures represents the correlation between the SSD of two consecutive frames and the precision (Figure 5.1) or recall (Figure 5.2) of the correspondent detection. A low SSD means that the two input frames are incredibly similar. A low precision indicates that a significant number of new objects were detected in the current frame compared with the previous frame while a low recall indicates that a high number of objects disappeared in the current frame compared with the previous frame. When precision and recall are close to 1, it means that the number, shape, and position of the detected objects in the current frame are incredibly similar to the results for the previous frame. The color of the points represents the level of the detection/classification confidence, being black for high confidence (higher than 0.45), light blue for medium confidence (between 0.35 and 0.45) and yellow for low confidence (lower than 0.35, but higher than 0.25, the CNN's detection threshold). Higher confidence intervals showed no significantly different behavior and were, for this reason, unified in the graph.

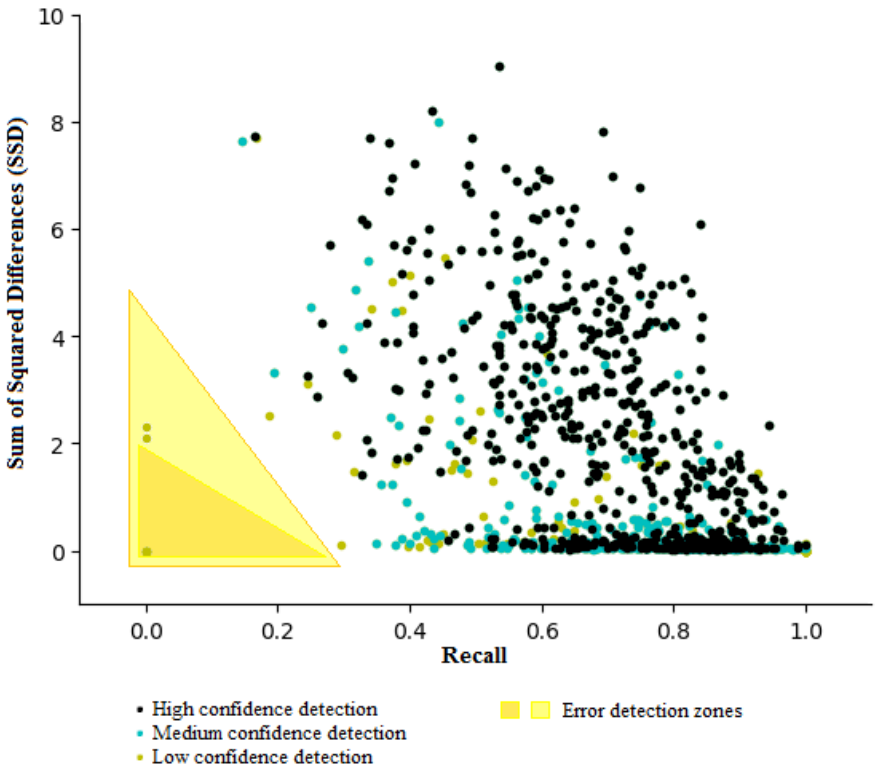
Even though Figure 5.1 and Figure 5.2 seem similar, it is important to notice that they represent two different (while correlated) concepts. Precision and recall were plotted separately to have a deeper analysis, considering errors that lead the CNN to miss objects (recall) or false positives (precision). Additionally, the precision and recall values can be

Figure 5.1: Precision and SSD calculated over the Caltech dataset in a fault-free environment.



Source: The Author

Figure 5.2: Recall and SSD calculated over the Caltech dataset in a fault-free environment.



Source: The Author

completely different in some specific frame pairs. Thus, points in the same position of Figure 5.1 and Figure 5.2 could be the results of a completely different pair of frames.

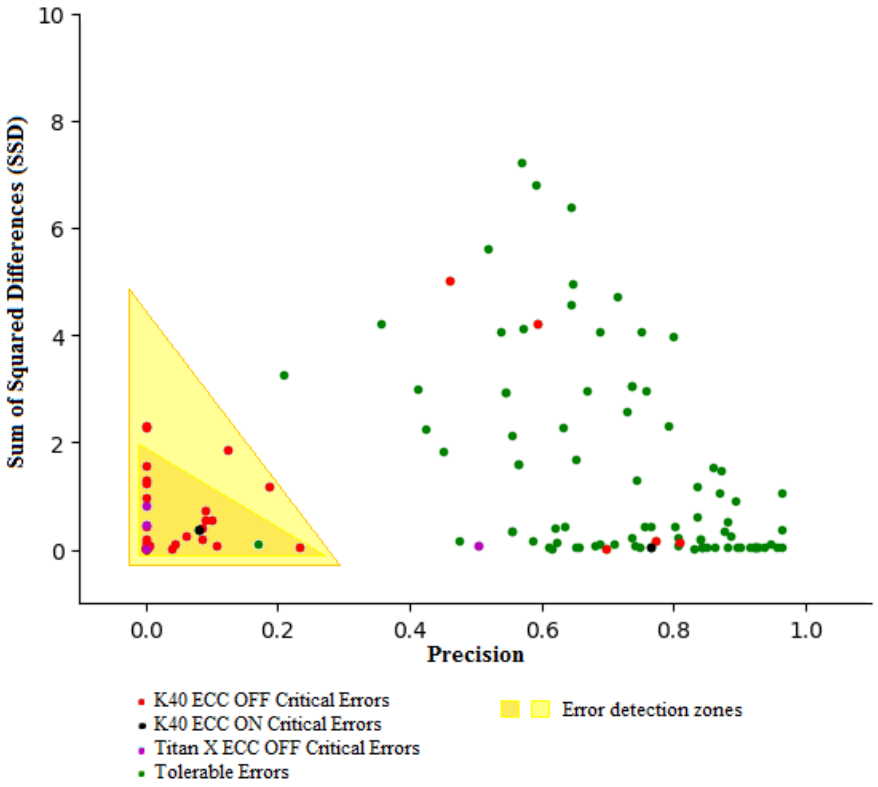
Figure 5.1 and Figure 5.2 show that a significant number of images are very similar to the previous image, resulting in a small SSD value. More than 57% of frames have a SSD lower than 1% with respect to the previous frame. Additionally, it is noticeable that when two consecutive frames show very different BBs (low precision or low recall), the input image comparison tends to also be different, especially for detections with greater confidence. As expected, it is possible to find some regions in both figures in which, in the absence of errors, it is improbable to have SSD - precision or recall correlation. In Figure 5.1 and Figure 5.2, the regions where two input images are very similar, but the output detection is entirely different were highlighted. Those unlikely correlations could be used to detect errors.

Two regions were selected, a bigger (and, thus, more aggressive) and a smaller one. It is worth noting that there are 18 different frame pairs fall into the bigger region (light yellow). This actually won't cause a correct detection to be detected as erroneous. Only 2 of the 18 points in the regions appears in both the precision and the recall graphs. Hence, the strategy will not trigger error detection for the 16 points that fall in the detection region only for precision and recall. The two points that are inside the detection regions for both precision and recall will trigger error detection. Nevertheless, looking at the frames and correspondent detection of the two points we found that they are false positives: the CNN detects an object which does not exist (thus reducing precision) and, in the next frame, the (inexistent) object disappears, (reducing recall). While those two points are not radiation-induced errors, they are false positives, and their detection is likely to improve the reliability of the CNN. Additionally, a less aggressive possibility is shown by the smaller region (orange), which contains no points that would be identified as an error.

5.2 Detection of Errors from Radiation

The analysis performed on the error-free dataset execution suggests that a frame pair that falls in the highlighted regions in both Figure 5.1 and Figure 5.2 can be identified as an anomalous behavior, possibly caused by an error. To test our method, we considered the errors observed during previously performed radiation experiment campaigns. All data is publicly available on a public Github repository [14]. It is worth noting that having experimental data is essential to prove the effectiveness of the proposed strategy without

Figure 5.3: Precision and SSD calculated over the experimentally observed errors.



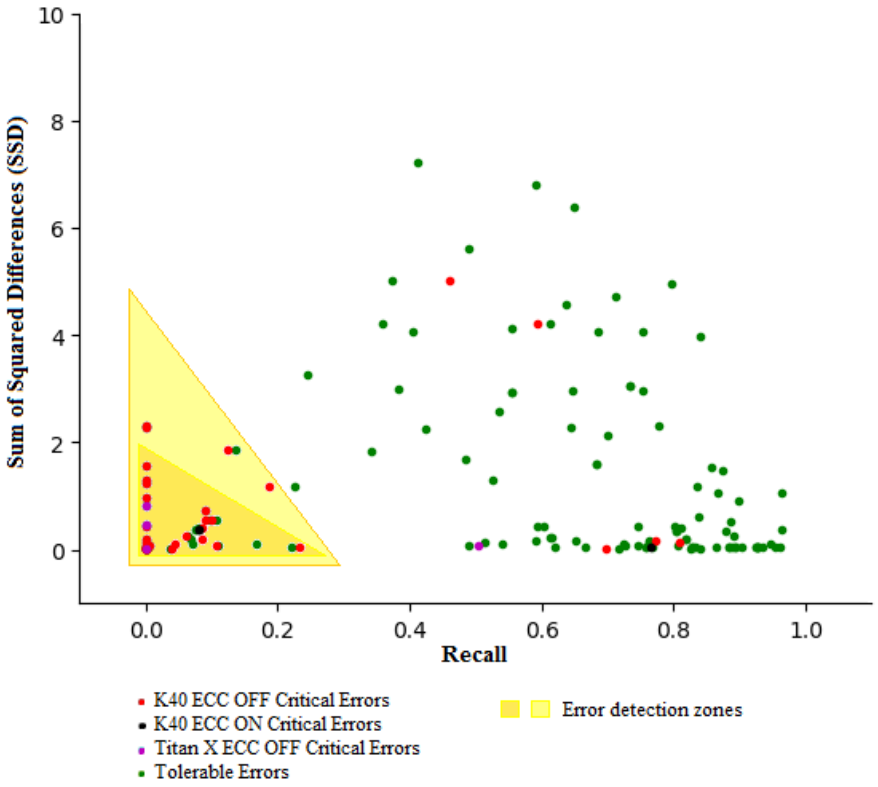
Source: The Author

being biased on the fault model.

The experimental data graphs (Figures 5.3 and 5.4) show the frames-detection correlation as explained in Section 6.1, but instead of comparing two error-free detection, the comparison was made between a frame with erroneous output and its preceding frame. The observed errors were separated based on the tested architecture and the criticality of the error. The colors on these graphs represent the architecture and criticality of the error, with red, black and purple being a critical error for a specific architecture and green being a tolerable error (for all architectures). We recall that a tolerable error is an error that does not impact detection and, thus, will not modify an autonomous vehicle behavior, while a critical error is an error that affects the detection and can change the vehicle behavior, potentially causing accidents.

Figures 5.3 and 5.4 are very similar to Figures 5.1 and 5.2. It is clear to see that most tolerable errors are distributed in the graph in a very similar way to the error-free dataset. These errors cannot be detected by this method, as they do not show different behavior when compared to error-free data. However, they do not impact detection and, thus, their detection is less important. Critical errors, though, tend to create significant differences in the output detections, even when the input images are very similar. As we

Figure 5.4: Recall and SSD calculated over the experimentally observed errors.



Source: The Author

can see, most critical errors are located in the highlighted anomalous regions. All the critical errors that are found inside these regions in one graph are also located inside this region in the other graph, so we can use the intersection of the errors detected in the precision and recall graphs to filter our results and quickly identify them. The reported data shows that a sufficient condition for errors to be critical is that detection precision/recall does not correlate well with the input images SSD.

Using this method, 80% of the critical errors were detected with the more aggressive strategy (light yellow region) or 68% with the less aggressive strategy (orange region). The critical errors that went undetected consisted mostly of detections where some Bounding Boxes are correct or almost correct, resulting in higher overall precision and recall score for the whole image. Even though the tested architectures do have different error rates and critical error rates, the critical errors produced behaved similarly for every architecture. All architectures produced detectable critical errors as well as critical errors that went undetected, showing no significantly different behavior relative to the method.

5.3 Validation for Detection of Errors in Fault-Free Executions

To validate the method on fault-free executions, a similar approach to the previous validation is made. The key difference is that the ground truth annotations from the dataset are used, instead of CNN executions. Each frame and its corresponding ground truth annotations are compared with the previous one, with the goal of identifying a correlation between their differences. Again, if an unlikely or impossible correlation is found in the dataset, it will be used to identify errors later. The dataset used was the KITTI dataset.

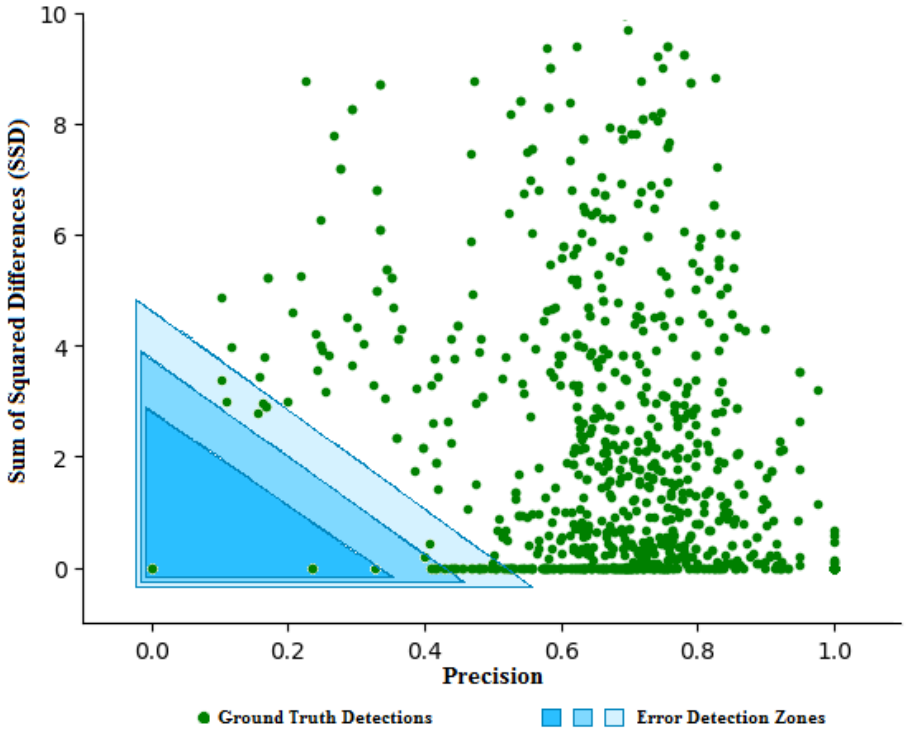
Figure 5.5 and Figure 5.6 show the correlation between the SSD of two consecutive frames with the precision and recall resulting from the comparison of the bounding boxes from the ground truth annotations from both frames. Each point in the figures represents the correlation between the SSD of two consecutive frames and the precision (Figure 5.5) or recall (Figure 5.6) of the correspondent ground truth annotation. A low SSD means that the two input frames are incredibly similar. A low precision indicates that the current frame contains a significant number of new objects compared with the previous frame while a low recall indicates that a high number of objects disappeared in the current frame compared with the previous frame. When precision and recall are close to 1, it means that the number, shape, and position of the objects in the current frame are incredibly similar to the previous frame.

Similar to Figures 5.1 and 5.2, Figures 5.5 and 5.6 represent precision and recall separately to have a deeper analysis. Even considering the ground truth annotations to be a perfect representation of the objects present in each frame, some frame pairs have a low amount of objects, which may lead to larger differences between the precision and recall scores. Thus, points in the same position of Figure 5.5 and Figure 5.6 could be the results of a completely different pair of frames.

It is noticeable that when two consecutive frames show very different BBs (low precision or low recall), the input image comparison tends to also be different. Also, most frame pairs with low precision scores do not have low recall scores. It is possible to find some regions in both figures in which, in the absence of errors, it is improbable to have SSD - precision or recall correlation. Similar to Figure 5.1 and Figure 5.2, regions which correspond to unlikely correlations were highlighted. These regions could be used to detect incorrect CNN detections.

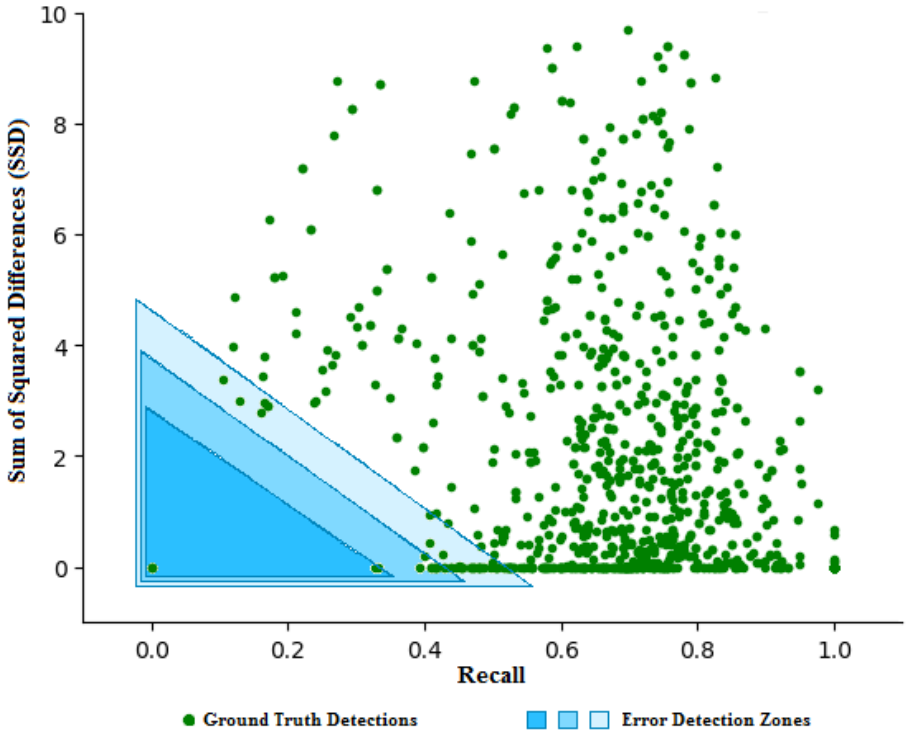
Three regions were selected. Region 1 is the smaller region (dark blue in the graph), a conservative detection threshold. Region 3 is the bigger region (light blue in

Figure 5.5: Precision and SSD calculated over the KITTI ground truth annotations.



Source: The Author

Figure 5.6: Recall and SSD calculated over the KITTI ground truth annotations.



Source: The Author

the graph), a very aggressive detection threshold. Region 2 is the middle-ground (blue in the graph). There are several points that fall into all three regions, but only the ones that fall in both graphs will be cause a error detection. With this strategy, Region 1 does not trigger any error detection in the ground truth, which is the ideal scenario. 0.2% of the frame pairs triggered a wrong detection on Region 2 and 3.5% of the frame pairs triggered a wrong detection on Region 3 (Figure 5.7).

5.3.1 CNN Error Distribution

Since the main goal of the application of this method in fault-free executions is to detect false positives, an analysis of the fault-free CNN detections is made. For this analysis, the precision and recall were calculated for the CNN detections in relation to the dataset, with a detection threshold of IoU >= 0.5. Detections with precision < 1 were labeled as false positives, since the CNN detected an object that is not present in the image. The rest of the detections were categorized according to the recall score. Lower recall scores indicate a high amount of false negatives in relation to the total amount of objects in the image. Higher recall scores indicate a lower amount of false negatives, since most, if not all, objects were detected. It is worth noting that very few image detections had a precision and recall scores of 1.

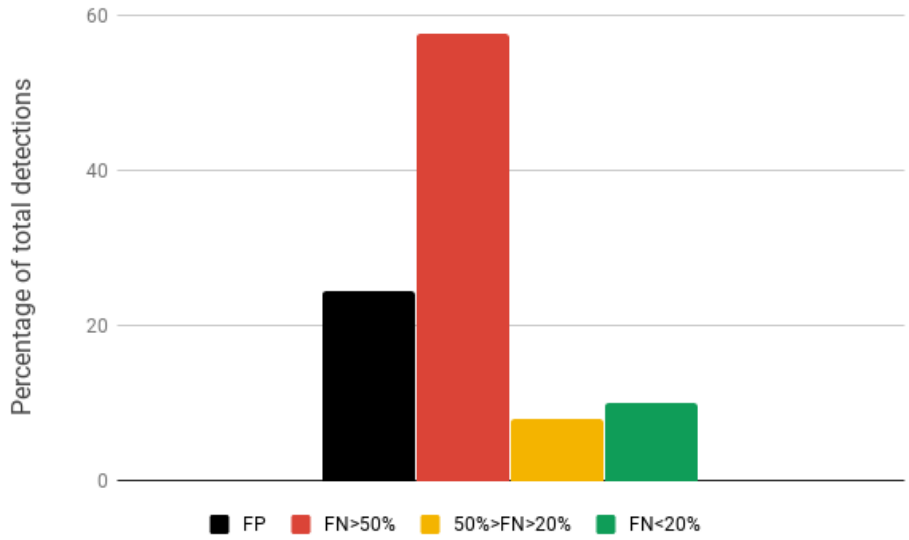
Images which generated no false positives were categorized in three groups according to the proportion of false negatives (FN). Since the error detection method does not make any distinction between error types, it is expected that detections with a high amount of false negatives also trigger the error detection.

Figure 6 shows that 24% of the detections contained false positives, 58% missed more than 50% of the objects (FN>50%), 8% missed between 20% and 50% of the objects (50%>FN>20%) and 10% missed less than 20% of the objects (FN<20%).

5.4 Detection of Errors in Fault-Free Executions

The analysis performed on the ground truth annotations suggests that a frame pair that falls in the highlighted regions in both Figure 5.5 and Figure 5.6 can be identified as an anomalous or, at the very least, uncommon behavior. To test the method, the CNN was run with the same dataset as the ground truth annotations, in a fault-free environment.

Figure 5.7: CNN Error Distribution



Source: The Author

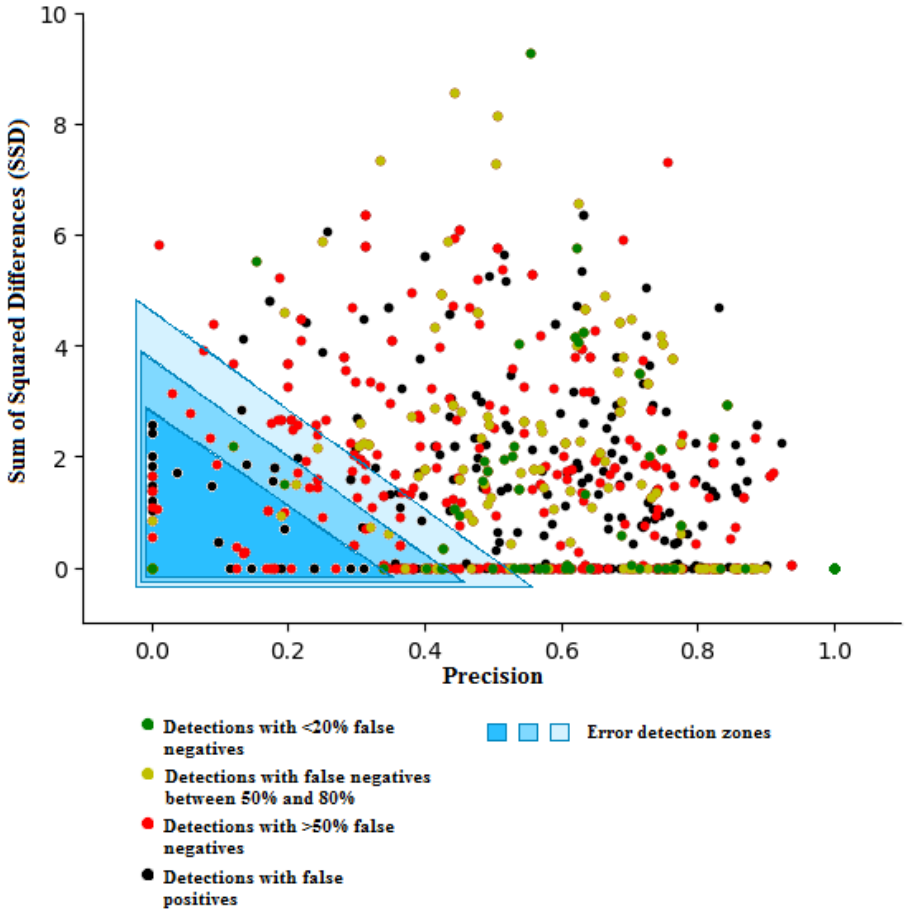
The experimental data graphs (Figures 5.8 and 5.9) show the frames-detection correlation, similar to Figures 5.3 and 5.4, but this time with falut-free CNN detections and utilizing the detection regions obtained in the ground truth validation. The detections were separated based on the previous CNN error distribution analysis. The colors on these graphs represent the CNN detection categories, with black being detections that contained at least one false positive, red being detections with more than 50% of false negatives, yellow being between 50% and 20% and green being less than 20%.

The graphs show that detections that contain a lower amount of false negatives and no false positives (green and yellow points) tend to behave in a very similar form to the ground truth annotations. Very few of them are located inside the error detection regions. Detections that contain false positives or a higher amount of false negatives do appear in a significant amount inside the detection regions.

Figure 5.10 shows an analysis of the error detections for each region. Any point on Figures 5.8 and 5.9 triggers the error detection if, and only if, it falls into the respective region in both precision and recall graphs.

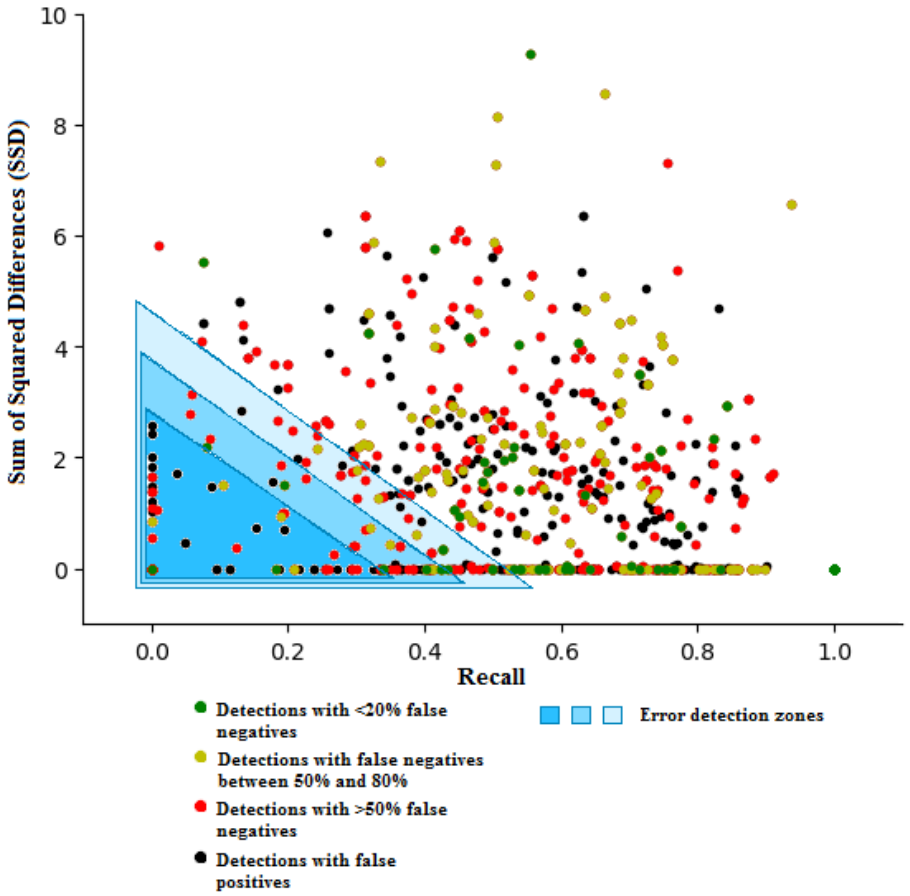
Region 1 detected 9% of the false positives and 2.2% of the frames with higher false negatives, and did not detect any of the frames with lower false negatives. Thus, Region 1 was able to detect a significant amount of false positives, and even though it detected some frames that did not contain false positives, these frames generated a high amount of false negatives and their detection would probably improve the reliability of the CNN.

Figure 5.8: Precision and SSD calculated over the KITTI dataset.



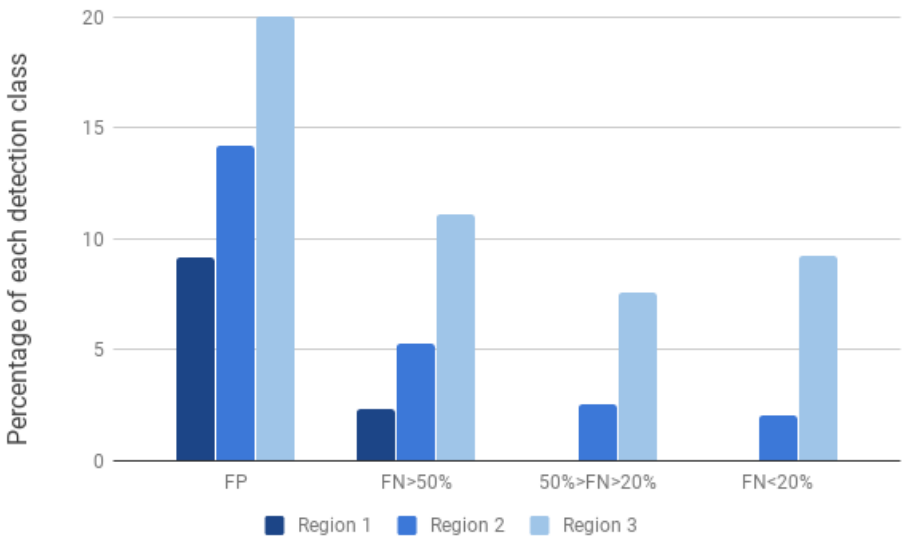
Source: The Author

Figure 5.9: Recall and SSD calculated over the KITTI dataset.



Source: The Author

Figure 5.10: Analysis of the error detections for each region.



Source: The Author

Region 2 detected 14% of the false positives and 5% of the frames with higher false negatives, but it also detected 2% of the frames in each of the other two categories. Even though it did detect more false positives than Region 1, the detection of frames that did not generate false positives or a high amount of false negatives makes this region less precise.

Finally, Region 3 detected 20% of the false positives and 11% of the frames with higher false negatives, but it also detected a much higher amount of the frames of the other two categories (8% and 9%) when compared to Region 2. It still has a higher density of false positives when compared to the whole dataset, but it is considerably less precise than Region 2.

6 CONCLUSION

In this project, an efficient detection method for radiation induced errors in object detection was presented, based on the spatio-temporal correlation from any two consecutive frames and their respective detected objects. The Caltech dataset was analyzed and the method succeeded in identifying unlikely frames-detection correlations in a fault-free environment. Whenever a discrepancy was found between the comparison of subsequent input images and their respective detected objects, an error was detected. The method was applied to radiation experimental data, and shows that most critical errors are detectable, validating the effectiveness of the proposed error detection strategy.

The method was also presented as a false positive detector for object detection CNNs. The KITTI dataset was analyzed and the method succeeded in identifying unlikely frame-bounding box correlations in the ground truth annotations. In this application, the method was not as efficient, but it did succeed in detecting a significant amount of false positives. A deeper analysis showed that it is possible to detect a higher number of false positives if a certain percentage of wrong detections are acceptable.

REFERENCES

BANERJEE, S. S. et al. Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data. In: **2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)**. [S.l.: s.n.], 2018. p. 586–597. ISSN 2158-3927.

BAUMANN, R. Radiation-induced soft errors in advanced semiconductor technologies. **Device and Materials Reliability, IEEE Transactions on**, v. 5, n. 3, p. 305–316, Sept 2005. ISSN 1530-4388.

DONGARRA, J.; MEUER, H.; STROHMAIER, E. **ISO26262 Standard**. 2015. Available from Internet: <https://www.iso.org/obp/ui/\#iso:std:iso:26262:-1:ed-1:v1:en>.

FAWCETT, T. An introduction to roc analysis. **Pattern recognition letters**, Elsevier, v. 27, n. 8, p. 861–874, 2006.

FERNANDES, F. et al. Evaluation of histogram of oriented gradients soft errors criticality for automotive applications. **ACM Trans. Archit. Code Optim.**, ACM, New York, NY, USA, v. 13, n. 4, p. 38:1–38:25, nov. 2016. ISSN 1544-3566. Available from Internet: <http://doi.acm.org/10.1145/2998573>.

HUVAL, B. et al. An empirical evaluation of deep learning on highway driving. **arXiv preprint arXiv:1504.01716**, 2015.

OLIVEIRA, D. et al. Radiation-Induced Error Criticality in Modern HPC Parallel Accelerators. In: ACM. **Proceedings of 21st IEEE Symp. on High Performance Computer Architecture (HPCA)**. [S.l.], 2017.

OLIVEIRA, D. et al. Modern gpus radiation sensitivity evaluation and mitigation through duplication with comparison. **Nuclear Science, IEEE Transactions on**, v. 61, n. 6, p. 3115–3122, Dec 2014. ISSN 0018-9499.

RECH, P. et al. Threads distribution effects on graphics processing units neutron sensitivity. **Nuclear Science, IEEE Transactions on**, v. 60, n. 6, p. 4220–4225, Dec 2013. ISSN 0018-9499.

REDMON, J. et al. You only look once: Unified, real-time object detection. **CoRR**, abs/1506.02640, 2015. Available from Internet: <http://arxiv.org/abs/1506.02640>.

REN, S. et al. Faster R-CNN: Towards real-time object detection with region proposal networks. In: **Advances in Neural Information Processing Systems (NIPS)**. [S.l.: s.n.], 2015.

Santos, F. F. d. et al. Analyzing and increasing the reliability of convolutional neural networks on gpus. **IEEE Transactions on Reliability**, p. 1–15, 2018. ISSN 0018-9529.

SAXENA, N. R. Autonomous Car is the New Driver for Resilient Computing and Design-for-Test. In: **Silicon Errors in Logic - System Effects (SELSE) Keyonte**. [s.n.], 2016. Available from Internet: <http://www.selse.org>.

UFRGSFCAROL. **Log database from beam experiments**. 2017. Github. Available from Internet: <https://github.com/UFRGS-CAROL/transaction_on_reliability_2018>.