Collision Detection in Smart-Cities by using Co-Simulation

by

Anukruthi Karre

A project report submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Master of Engineering in Electrical and Computer Engineering

Department of Electrical, Computer, and Software Engineering

University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

November 2022

Project Report REVIEW INFORMATION

Submitted by: **Anukruthi Karre**

Master of Engineering in Electrical and Computer Engineering

Project/Major report title:
Collision Detection in Smart-Cities using Co-Simulation

The project report was approved on November 29, 2022 by the following review committee:

Review Committee:

Research Supervisor	Dr. Akramul Azim
Second Reader	Dr. Haoxiang Lang

The above review committee determined that the project report is acceptable in form and content and that a satisfactory knowledge of the field was covered by the work submitted. A copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

ABSTRACT:

Road traffic has become prominent in everyday living, impacting or disrupting services to people and daily routines. With the rise in automobile manufacturing and the frequency of vehicle crashes, human catastrophes, like fatalities, accidents, impairments, and destruction of property, are surging yearly. Vehicle collision detection has recently gained prominence in decreasing manually operated and autonomous vehicle fatalities. The concept of independent and self-driving cars relies on accurate object recognition, including pedestrians, vehicles, buildings, and other moving objects. Various object-detecting approaches have been proposed to help autonomous vehicles (AVs) achieve consistent, safe driving. Object prediction and detection have noticed numerous algorithmic changes that have improved speed and accuracy. In this study, I used a traffic dataset produced by a CARLA simulator to anticipate collisions using the Yolov7 model. I generated the dataset from a CARLA simulation bench in video sequences, manually annotated the frames, and used the deep learning algorithm Yolov7 to train them. The model predicts the collision a few seconds before it occurs in real-time. I implemented this framework to increase the safety of driving in self-driving vehicles.

Keywords: collision; safety; prediction; detection; simulator

AUTHOR'S DECLARATION

I hereby declare that this project report consists of original work of which I have authored. This is a true copy of the work, including any required final revisions, as accepted by my committee.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this work to other institutions or individuals for the purpose of scholarly research. I further authorize the University of Ontario Institute of Technology (Ontario Tech University) to reproduce this work by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my work may be made electronically available to the public.

Anukruthi Karre YOUR NAME

ACKNOWLEDGEMENTS

This work would not have the spirit or been completed without the invaluable, academic, and educational support and belief in me as a writer and researcher by Dr. Akramul Azim. Dr. Azim was crucial in leading the team both before, during, and after the project.

STATEMENT OF CONTRIBUTION

- Karre, A., Arafat, M.M. and Azim, A. “Design and Development of a ML-based Safety-Critical Fire Detection System.” (Accepted by COMSYS 2022)
- Karre, A. and Azim, A. “Collision Prediction of Smart Cities Using Co-simulation.” (Submitted to SYSCON 2023)

Contents

List Of Figures

CHAPTER 2

CHAPTER 3

CHAPTER 4

List of Tables

CHAPTER 4

LIST OF ABBREVIATIONS AND SYMBOLS

YOLO - You Only Look Once
CVAT – Computer Vision Annotation Tool

Chapter 1. Introduction

1.1 Description and Scope

It is stated that autonomous vehicles (AV) will offer comfortable, affordable, and secure journeys. Without question, self-driving cars will have a significant impact on how we travel. The secure and safe functioning of the AV is one crucial issue that prevents it from being used on the road. Annually, there are more occurrences of death and disability due to daily traffic accidents. Recent data on semi-autonomous driving [1] obtained by the National Highway Traffic Safety Administration recorded 392 crashes in 10 months, and approximately 70 percent of the crashes were caused by self-driving vehicles. The WHO estimates that 1.35 million mortality cases, or 2.2 percent of all fatalities, occur worldwide. Excessive speed, negligent driving, driver exhaustion, wandering animals on the roadways, and inadequate infrastructure are the primary causes of traffic accidents. Most deaths and disabilities in these accidents result from emergency medical assistance's slow reaction. The period immediately after a traumatic injury is known as the "golden hour," during which time the likelihood of preserving a person's life increases, on average, by one-third.

As a result, considerable resources have been devoted recently to ensuring an effective and rapid rescue team. One of the critical issues for intelligent transportation systems is effectively detecting traffic incidents, particularly regarding vehicle collisions. Vision is a cost-effective method for autonomous traffic collision detection since it can supply a wealth of traffic information. It is still a difficult task. However, recent studies reveal that loss of visibility, particularly near junctions, is one of the causes of the highest number of accidents on the road. Driving faults are the primary cause of traffic accidents. It can be easier to spot and avoid collisions if one knows how different types of roads affect a driver's behavior and other factors like weather and road damage.

The ability of autonomous cars (AVs) to improve mobility and security in commuting has received considerable interest recently. Testing and evaluating an AV's vehicle intelligence is a crucial step in creating and implementing self-driving cars since it shows if an AV can function safely and effectively without human assistance.

1.2 Problem Statement

Future transportation will be drastically altered by autonomous vehicles. Self-driving cars still need to go through many tests and analyses before they can reliably and safely navigate the roads. In addition to dealing with erratic conduct from other drivers on the road, they also must deal with unanticipated incidents that happen suddenly and shifting weather patterns. It would be impossible to test every possible situation on our roadways. For autonomous vehicles to establish their dependability in real-world scenarios, they would need to travel countless miles and exist on the roads for centuries. As a result, simulation software for driverless cars has emerged as a crucial resource. All potential circumstances—thousands of them—can be evaluated and realistically replicated in a virtualized environment.

A vast array of various simulation techniques is necessary to mimic the intricate world of autonomous cars. The essential word here is co-simulation. It is vital to test data to study the collisions on different roads. This calls for collecting data or testing the algorithms against the traffic in real-time. Unfortunately, there are downsides to it.

1. There is a limited real-time traffic dataset to test
2. The tests are difficult to perform and expensive [2]
3. It is unsafe and unethical to perform a particular scenario in real-time to study the collision.

The ideal alternative would be to use a simulator for collisions.

1.3 Contributions

- The report introduces the simulation bench from which the dataset was collected and its importance.
- The report evaluates the YOLOv7 algorithm on the dataset to predict and detect collisions.

1.4 Organization

The report is organized as follows. Chapter 2 discusses the background and related work on co-simulation and object identification algorithms to improve traffic safety. Chapter 3 describes the proposed system and the methodology involved in developing it in three stages: Collecting the data from a simulator, annotating, and pre-processing the data, and training a machine learning algorithm on the dataset. Chapter 4 discusses the system specification and implementation process followed by the model predictions. Finally in Chapter 5, I conclude the proposed work and mention the possible future developments.

Chapter 2. Background

2.1 Co-Simulation:

In co-simulation, the subsystems that make up a coupled problem are modeled and simulated in a distributed way. As a result, modeling is carried out at the subsystem level without consideration of the interrelated issue. Additionally, the subsystems are run as black boxes during the connected simulation. The subsystems will communicate with one another throughout the simulation. Co-simulation can be considered the combined simulation of existing well-established tools and semantics when used with the appropriate solvers.

Co-simulation demonstrates its value in validating multi-domain and cyber-physical systems by providing a versatile approach that enables simultaneous evaluation of many domains with various time steps. The work necessary to create a complete system simulator can be broken down into smaller, distinct tasks when we can break aspects of a system into individual components that are then loosely coupled. Additionally, once the system has been broken down into parts, each is solved independently and shares its solutions with the others at designated communication points. Additionally, it enables the distribution of the calculations for improved efficiency.

The many elements linked to one another can also be made up entirely of binary codes, allowing for collaboration throughout the industry and even between rival companies without disclosing open-source applications that might potentially reveal trade secrets.

CARLA simulator: CARLA is an accessible simulator for driverless cars. This was created from the start to serve as a dynamic, modular Interface for solving numerous tasks related to the driverless car issue. One of CARLA's main goals is to serve as a tool that anybody may use and alter to help democratize automated driving innovation and research. The simulation model must be able to satisfy the demands of various use cases that are observed in regular

driving. The OpenDRIVE standard is used to specify streets and urban settings in CARLA, created using Unreal Engine to perform experiments.

2.2 Object Recognition:

The task of object identification and prediction in computer vision involves finding occurrences of particular kinds of visual things, including people, creatures, vehicles, and houses, in images like still pictures or video sequences. Object identification aims to develop analytical techniques that provide the crucial data that machine learning applications require. One of the fundamental challenges in image processing tasks is identifying objects. It is the starting point for many following image processing issues, such as segmentation tasks and motion detection. A few examples of applications for specialized image recognition are human recognition, individuals count, facial identification, textual data recognition, pose identification, or license plate identification.

A deep learning method called YOLO is a real-time object recognition algorithm with several uses in image processing. This method employs only one bounding box to pinpoint elements like object categories, centroid, and dimensions. It outperformed Fast R-CNN, RetinaNet, and Single-Shot MultiBox Detector in terms of prediction accuracy, quickness, and object recognition in a single session to gain a competitive advantage over its competitors. Since its launch in 2016, the YOLO models have continued to grow.

On top of YOLO, the YOLOv2 [3] architecture added various features like Batch Normalization, higher image quality, and anchor boxes.

YOLOv3 [4] built on past versions to boost performance on smaller entities by adding an object class value to bounding box forecasting, increasing connections to the backbone network layers, and making predictions at three different granularity levels.

YOLOv4 [5] included new features such as enhanced grouping of the elements, a "bag of freebies" containing modifications, instant activation, and other improvements.

The YOLOv5 [6], [7] algorithms reduce the experimental costs due to how quickly the algorithm trains while building the model. YOLOv5 could infer single photos, groups of frames, video streams, or webcam connections.

With the hardware in mind, YOLOv6 [8] redesigns the YOLO backbone and neck to create the EfficientRep Backbone and Rep-PAN Neck. The features are separated from the final head by added layers in YOLOv6, which proved that performance is improved. The YOLOv6 GitHub implements some improvements to the training pipeline and architectural modifications. These improvements include training without the anchors, assigning labels dynamically, and SIoU box regression loss.

Yolo works on the following methods.

1) Residual blocks: At this stage, the model separates the incoming image into equal-sized grids, each in charge of identifying an object or a portion of an object inside the grid.

2) Bounding box regression: Each cell contains a bounding box with properties such as weight, height, class, and center that highlight the objects inside. YOLO predicts these using bounding box regression, representing the probability of an object occurring within the bounding box.

3) Intersection over union (IoU): Overlapping bounding boxes are called IoU. Each grid cell is accountable for the anticipated detections and their confidence score. Dividing the overlapped area by the union area determines the IoU. If the anticipated and actual bounding boxes are identical, the IoU equals 1. Here, getting rid of bounding boxes that deviate too much from the actual box is simpler.After partitioning the image into grid cells, each cell anticipates bounding boxes for each object with specific likelihood scores and class probabilities. For tasks with multiple labels, the predictions are made simultaneously by the Yolo algorithm. For the final detection to result in distinctive bounding boxes containing objects, the IoU ensures that the predictions align with reality.

2.3 Related Work

Several approaches were proposed that includes various deep learning algorithms to predict accidents on road [9], [10], [11], [12]. Some of them are listed below:

2.3.1 Detecting video objects with deep learning algorithms

Feng Yang et al [13] used the Yolov7 model for object detection and then integrated it with the Deepsort algorithm for object tracking. Using the identical parameters for both of them, they used the YOLOv7-Deepsort algorithm to MOT16 datasets [14] and compared it to the YOLOv5-Deepsort algorithm. They employed the object detection models YOLOv7, YOLOv5s, YOLOv5m, and YOLOv5l. The YOLOv7 achieved higher accuracy and precision of 40.82 and 82.01 compared to the other models.

This leads them to conclude that the yolov7-deep sort algorithm has a higher tracking accuracy.

2.3.2 Method for improving detection of pedestrians

Devarsh Patel et al. [15] suggested identifying humans from visual images captured in poor lighting conditions with a reasonable level of accuracy utilizing object classification algorithms pix2pixGAN and YOLOv7 on thermal pictures produced through picture transformation. The authors converted the visible photos into thermal images using the pyramid pix2pixGAN algorithm and employed YOLOv7 for effective object detection. They reasoned that object detection on translated infrared images would enhance pedestrian detection tasks without needing specialized, expensive infrared imaging equipment. The performance of object recognition that used this learning algorithm was then evaluated and compared with pre-trained models simply on images captured. They discovered that, even in incredibly low conditions, their method outperformed visible picture models.

2.3.3 AI methods to recognize collisions

A system for vehicle collision detection [16] was suggested by S.V. Gautham and D. Hemavathi utilizing a deep learning algorithm YOLO, where the authors examined live-fed

video streams from the Surveillance cameras. This approach identifies the accident from the input and sends an alert to the appropriate emergency agencies in the area. They used the neural network output to identify vehicular traffic collisions by collecting keyframes and developing their custom variables. To ensure their technique is appropriate for the current accident situations, they created this framework while considering a few extreme variables and factors, such as bright sunlight, poor vision in the dark, weather conditions, and shadow. This method for detecting traffic accidents concerning vehicle movements considers several variables and characteristics.

2.3.4 Method to build and notify a real-time collision and road barriers

Research by Chaeyoung Lee et al. led to the development of a deep learning-based model [17] that can identify unusual driving behavior and a service that can help avoid collisions with other vehicles and heavy traffic. After classifying different car crash kinds using FFmpeg for model production, the authors extracted accident images from traffic accident video data. The neural network-based technique YOLO is used to assess just head-on crash situations. They created this application so that when the automobile accident detection model finds anomalies on the road, it sends a warning notification and images of the accidents or obstructions to the user, as shown the Figure 2 a, Figure 2 b, Figure 2 c, Figure 2 d, and Figure 2 e. They combined their car accident detection model with the road obstacle recognition model. The presented service was validated using simulated trials on Surveillance cameras in various cities. They aimed to enhance the autonomous car sector and improve safety on the road by providing solutions.

Figure 2 a. The two images above show the detection of an accident in (a) and the detection of an obstacle on road in (b) in the northern side of a city [17].

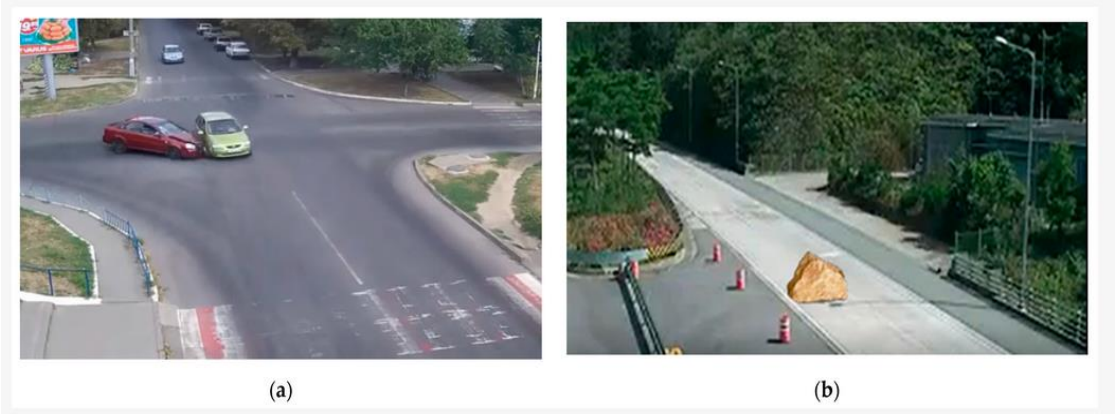

Figure 2 b. The two images above show the accident of a car in (a) and an obstacle on road in (b)in the southern part of the city [17]

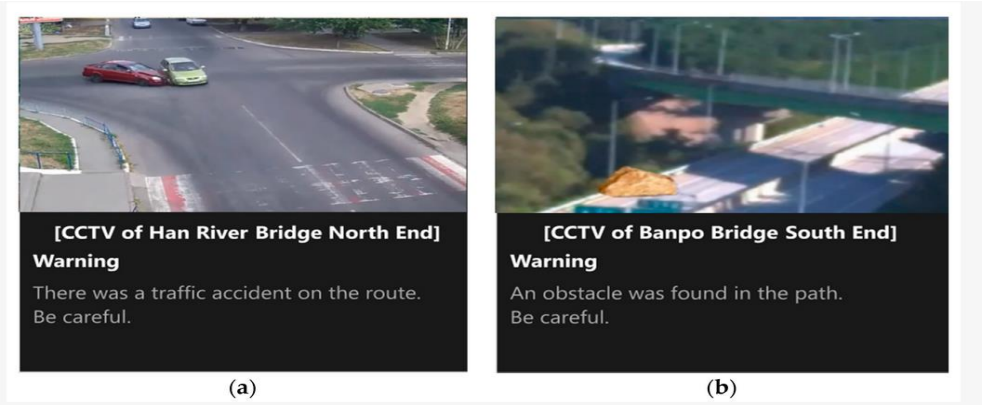

Figure 2 c The two images above show the CCTV notification window in the northern end in (a) and the CCTV notification window in the southern end in (b) [17]

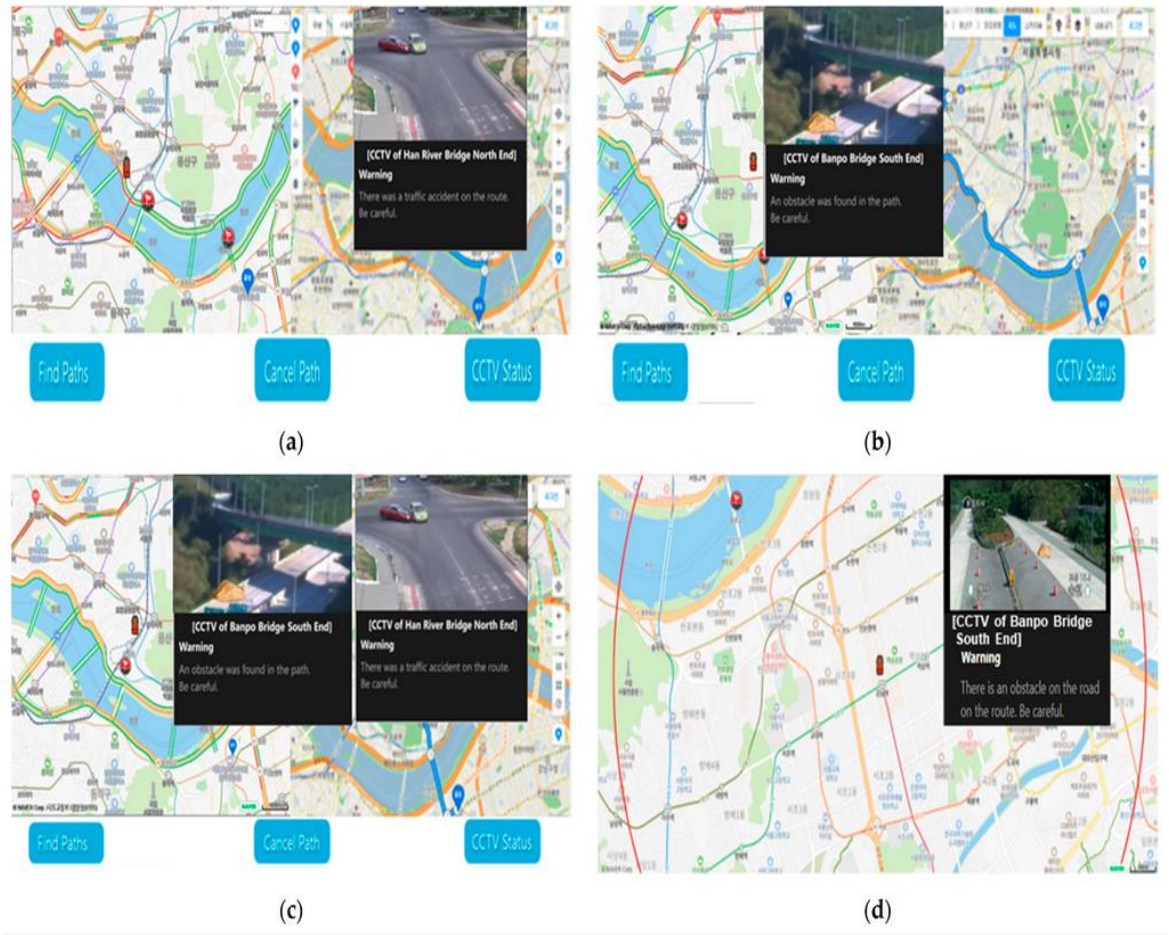

Figure 2 d The four images (a), (b), (c), (d) above show the maps of different routes and the notification received by users in the area the accident happened [17].

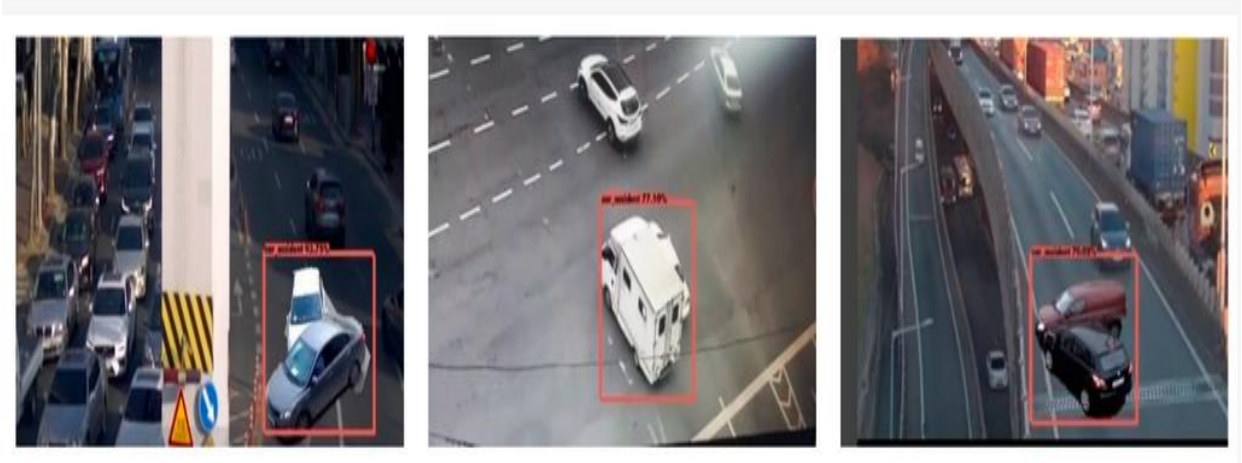

Figure 2 e The above figure shows the accident detected in different roads from the CCTV [17]

Chapter 3. Methodology

3.1 Proposed System

Traffic accidents rose significantly in the past 10 years due to several factors such as bad roads, and careless driving. Several steps are taken to mitigate the issue but it's still not helping. The invention of autonomous vehicles brought in hassle-free and ease of driving. However, there are some safety issues concerning the self-driving vehicles that's stopping them in the market. Hence, we wanted to create a method where we can study the different types of collisions and further reduce it. We did this using a CARLA simulator bench. We first generated the dataset from the simulator and used a deep learning model to predict collisions. A flowchart representing the overall methodology is shown in Figure 3 a

3.2 Dataset Generation from the Simulator Bench:

3.2.1 Simulator Bench – CARLA:

The simulator I used in this report is a CARLA simulator Bench. In simple terms, the simulation bench is a device created to bridge a gap in the autonomous vehicle by offering an extremely stable and streamlined infrastructure to support the creation of autonomous driving systems for vehicles that use widely supported and available open-source software. The system offers a wide range of sensor suites with configurable specifications, including LIDAR, depth sensors, RGB cameras, and many others. By enabling users to co-simulate a digital twin of a real-world car, the system allows for quick, reproducible, accurate, and practically infinite testing and simulation of many of the components of a self-driving system. Due to the lack of restrictions from real-world conditions, users can acquire practical simulation and testing data.

Additionally, the simulation bench offers a hardware-in-the-loop configuration that enables hardware and software components to be evaluated simultaneously in a simulation platform. This allows system users to achieve supervised testing and verification before being deployed

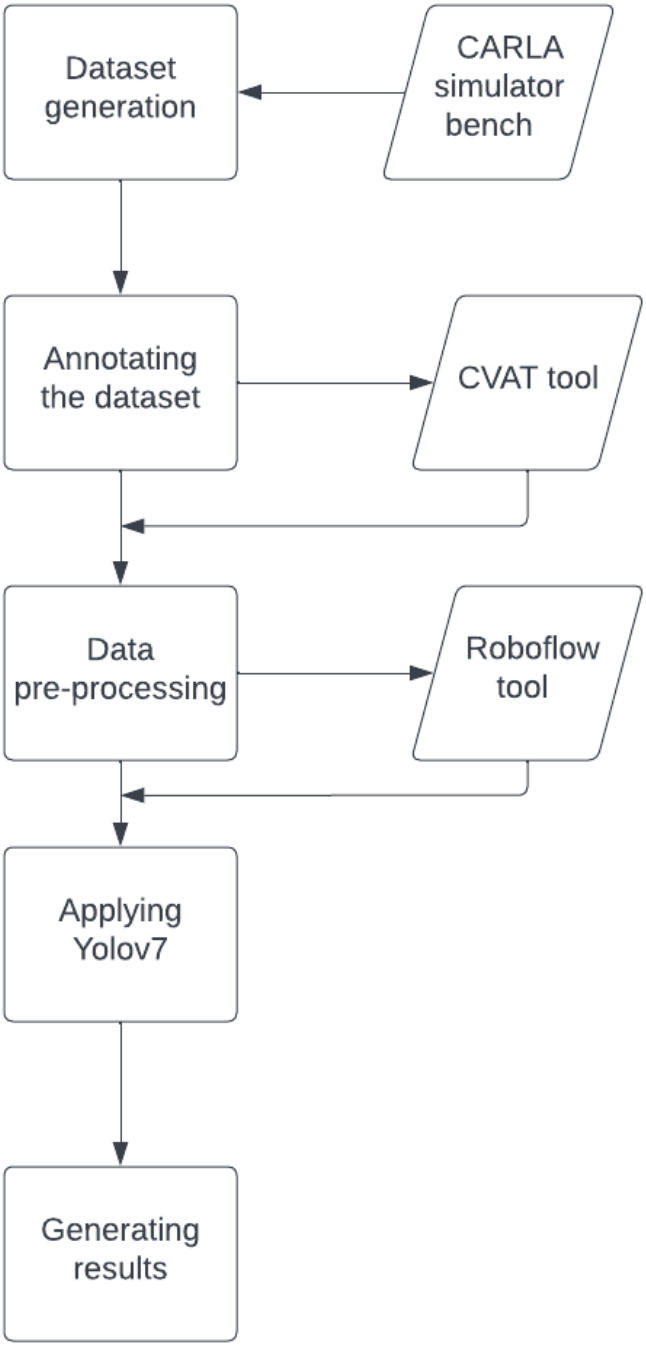

Figure 3 a. The figure shows the overall methodology process

for real-world testing. It enables them to attach different hardware peripherals to the bench and carry out simulation tasks in a closed environment. The bench connects to CARLA Simulator using actual automotive hardware. It creates a real-time driving environment and incorporates controls to test various driverless car modules. It also strives to raise software quality to give customers more realistic and contemporary features (such as Lane Assisting). The most important feature of the simulation bench is its data-capturing and logging capabilities. The capacity to gather and examine the information obtained from simulation and testing activities yields beneficial information that determines which parts of an autonomous vehicle system need adjustments or development.

Users have an accessible platform where they can conduct in-depth testing and obtain analytics on the software and hardware components of their self-driving system due to the controlled testing environment within the simulation bench. Additionally, compared to testing individual units in reality, the simulation-based technique enables users to undertake thorough testing in a shorter time frame. Furthermore, the simulation-based approach would let users run some test scenarios that might only sometimes be feasible in real time because of limitations imposed by the real world. With the need for more available traffic data in the market, this simulator is greatly helpful in generating massive datasets of all possible conditions for further analyses.

Due to the data logging pipeline inside the simulation bench, users can undertake extensive simulation and testing that was previously impossible. Additionally, the bench's usage and HIL design would open considerably greater potential for automated driving research and development.

3.3 Applying Machine Learning Model:

This section discusses applying a regression-based deep neural network for predicting collision. We first discuss the dataset and the preprocessing steps, and further apply a machine-learning model to it.

3.3.1 Data Preparation:

In this report, we used the CARLA simulator to run numerous simulations, and after screen recording the results, we generated the data in an mp4 video file. These video clips show several accident scenarios on the road involving various kinds of automobiles, such as cars, trucks, and motorcycles. We made sure to capture the collisions as realistic as possible for further analysis.

3.3.2 Data Pre-processing:

The obtained data was pre-processed so the Yolo algorithm could train on it. For annotating the videos, we utilized two tools: CVAT and Roboflow. CVAT is a computer vision annotation tool that is used for annotations. The videos' bounding box annotations were all manually created. We first carefully drew bounding boxes on the vehicles, the collision, and the object or vehicle it collides with, on every frame after uploading each video to CVAT. I further labeled them as "vehicle," "accident," "object," and "pedestrian." Figure 3 b shows the overview of the user interface of CVAT. Every frame of the video sequence can be altered and processed using CVAT. Apart from this, CVAT also allows us to adjust the frame rate. For the project, we chose the frame rate to be 40 frames per second. The right side of the interface shows the different labels used in the annotation process. We labeled the videos in such a way that the models learn and give accurate predictions on collision. We annotated Each video sequence a few seconds before the collision happened as shown in Figure 3 c, during the time of the occurrence of the collision as shown in Figure 3 d, and a few frames after the collision, as shown in Figure 3 e. The image's blue, green, and red bounding boxes represent vehicle, object, and accident labels, respectively.

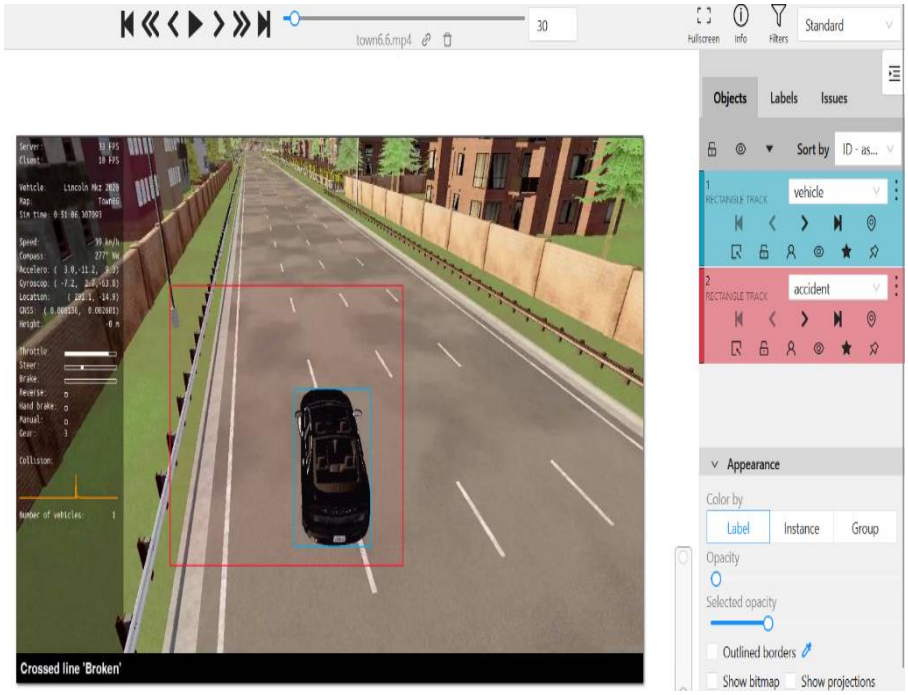

Figure 3 b. The figure shows the graphical user interface of CVAT tool

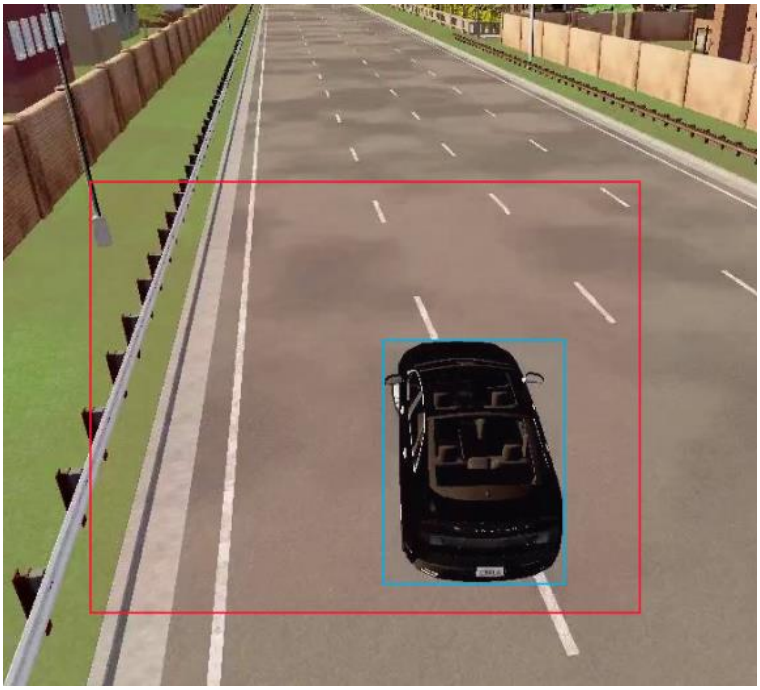

Figure 3 c. The figure shows the bounding box annotations for the frames before the accident

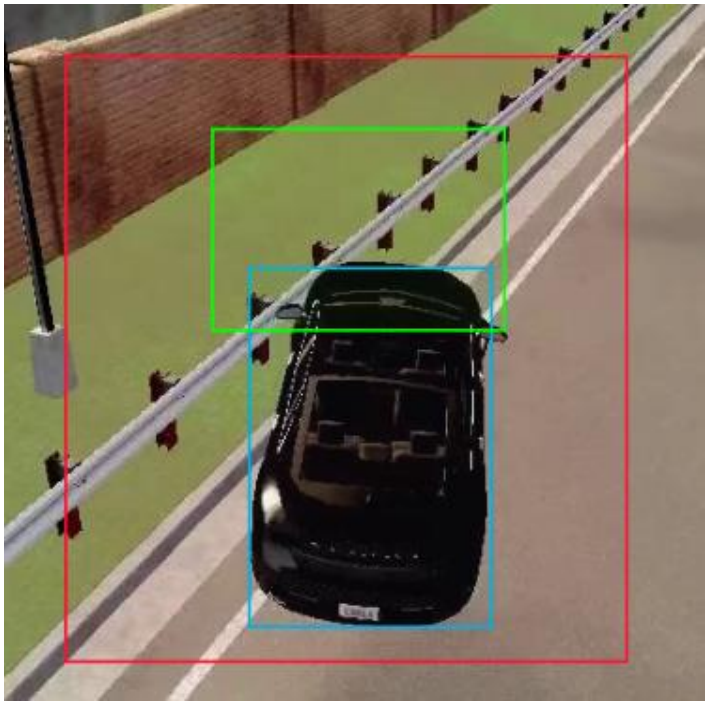

Figure 3 d. The figure shows the bounding box annotations during the occurrence of the collision

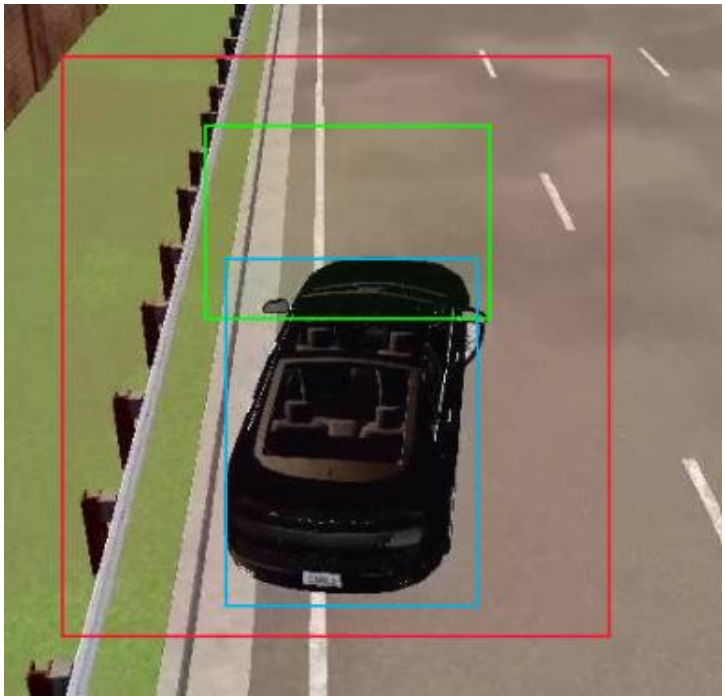

Figure 3 e. The figure shows the bounding box annotation a few seconds after the collision

3.4 Data Modelling: Yolov7

The most recent piece in the YOLO series is YOLOv7 [18]. Based on earlier research, this network significantly increases detection accuracy and speed. The primary considerations in constructing an efficient architecture are the model's size, quantity, and computing density. The VovNet model takes a step further and examines how the input/output channel ratio, the architectural branching number, and each component's operation affects the system's performance.

3.4.1 E-ELAN: Extended Efficient Layer Aggregation Network

The next significant advancement in architecture search is ELAN, and YOLO v7 extends this to E-ELAN. The ELAN article concluded that a deeper network could effectively learn and converge by controlling the shortest and longest gradient path. Large-scale ELAN has attained stability regardless of how many processing units are stacked or how long the gradient approach is. This stable state might be lost if infinitely more computing blocks are piled, and the rate of parameter consumption will drop. E-ELAN employs the techniques of expand, shuffle, and merge cardinality to constantly improve the training efficiency of the model while keeping the initial gradient approach. E-ELAN only changes the computing block's architecture without changing the architecture of the transition layer. This technique makes use of group convolution to expand the range and cardinality of computing units. All the computing modules at a computing level share the same grouping variable and channel multipliers.

Following the predetermined group parameter g, the feature map produced by each computing block is subsequently divided into g groups and concatenated. Currently, the channel count in every set of extracted features will be the same as in the conventional design. By including g groups of feature maps, merge cardinality is carried out.

Along with upholding the original ELAN design architecture, E-ELAN can direct other collections of computational blocks to pick up more varied functionalities. Figure 3 f shows the architecture of E-Elan.

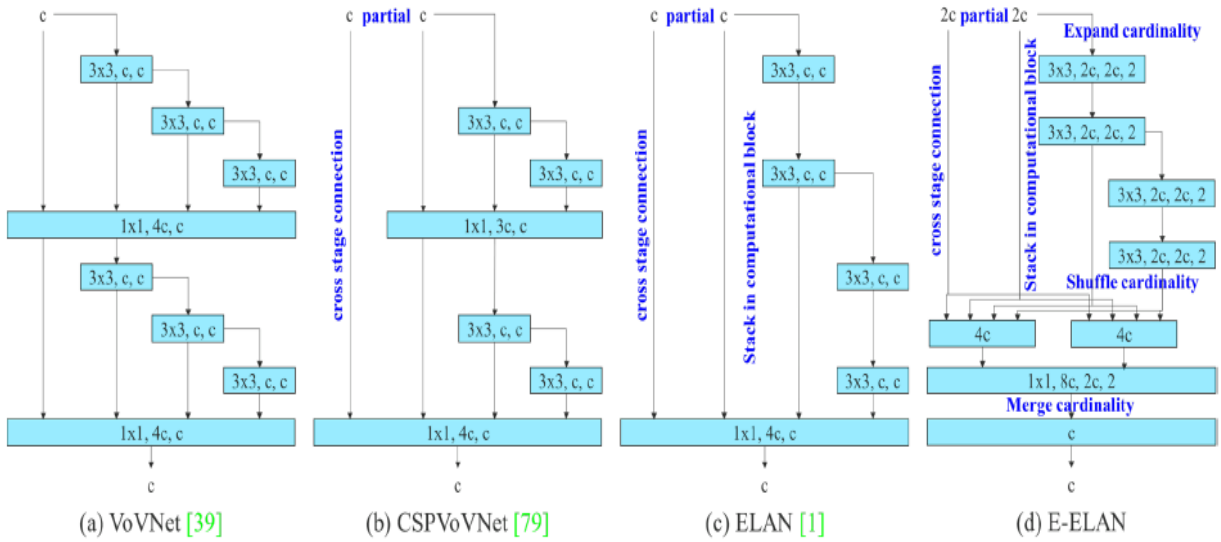

Figure 3 f. The figure shows the extended efficient layer aggregation network architecture of YOLOv7 as stated in [17]

3.4.2 Model scaling for concatenation-based models:

Model scaling is mostly employed to change particular model attributes and create new designs at different scales to account for different observations. The model scales between width, depth, and resolution in EfficeintNet, a well-known Google architectural design. Later, however, researchers attempted to determine the impact of group and vanilla convolution on the number of parameters and computation when executing scale.

The approach taken by EfficientNet is not appropriate for concatenation-based design because, while trying to scale up or down based on depth, the in-degree of a translation layer comes right after a concatenation-based computing block would change. For example, scaling-up depth will result in a shift in the proportion of a transition layer's input channel to the output channel, which could reduce the model's hardware consumption. The suggested method should also figure out how the output channel of a computational block will vary when its depth factor

is adjusted. Figure 3 g displays the result after applying width factor scaling to the transition layers with the same amount of change. The optimal structure and the model's original attributes can both be preserved by the Yolov7 compound scaling method.

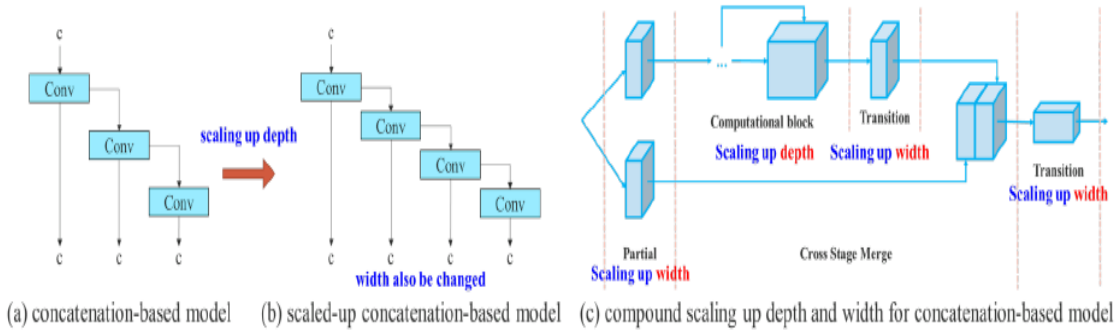

Figure 3 g. The above figure shows model scaling of concatenation-based models present in YOLOv7 as stated in [17]

3.4.3 Trainable bag of freebies:

Researchers from YOLOv7 investigated how different networks should be connected with re-parameterized convolution employing gradient flow transmission paths. Figure 3 h below shows the placement of the convolution blocks. Four of the eight combinations work well.

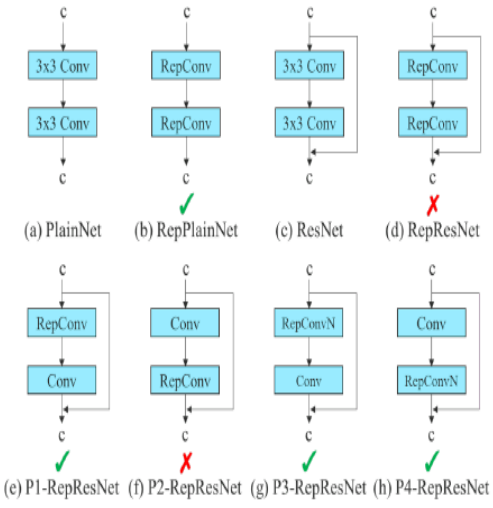

Figure 3 h. The figure shows the re-parameterized model present in YOLOv7 as stated in [17]

3.4.3 Fine for lead loss and coarse for auxiliary:

Deep supervision is a method that is frequently applied when deep networks are being trained. Its main idea is to increase the number of auxiliary heads in the network's intermediate layers while using assistant loss as a reference for shallow network weighting. Deep supervision can considerably enhance the model's performance on many tasks, even for designs like ResNet and DenseNet, which often converge well. The object detector architecture is depicted below in Figure 3 i in both its "without" and "with" deep supervision states. In the YOLOv7 architecture, contributing to training is the responsibility of the auxiliary head, while the primary head oversees producing the outcome.

YOLOv7 utilizes the lead head forecast as a guide to producing coarse-to-fine structured labels employed for the lead head and auxiliary head training processes., using lead head prediction as guidance. The accompanying Figure 3 i displays the two suggested deep supervision label assignment mechanisms.

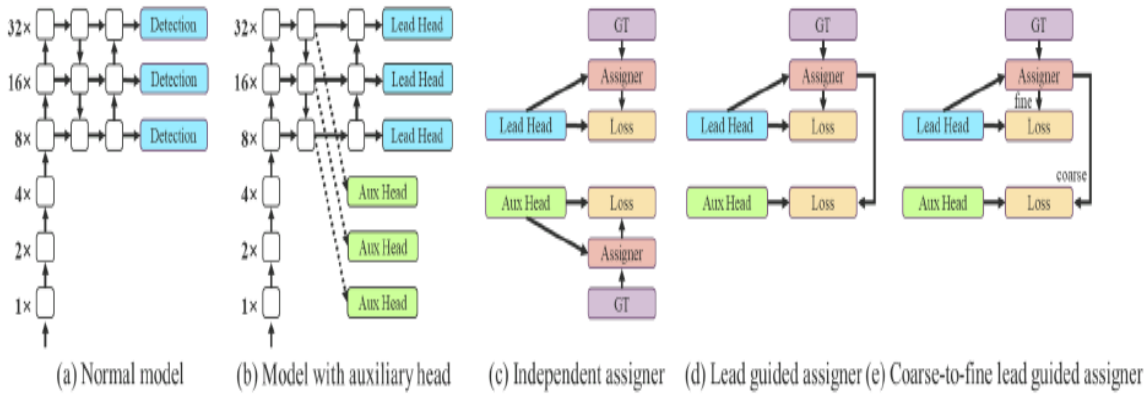

Figure 3 i. The figure shows Coarse for auxiliary and fine for lead head label assigner present in YOLOv7 as stated in [17]

Lead head guided label assigned: By enabling the shallower auxiliary head to directly examine the main head's data, the lead head will be more likely to focus on obtaining the remaining data that is not acquired.

Coarse-to-fine lead head guided label assigned: The optimizable upper bound of the fine label is constantly more significant than the coarse label in this approach, which also enables the dynamic adjustment of the relative relevance of fine and coarse labels during the learning process.

Chapter 4. Implementation and Results:

4.1 Experimental Setup:

The report used two computers for designing the model. The first is Linux OS with GPU NVIDIA Corporation TU116 [GeForce GTX 1660 Ti], used to generate the data, and the second is a Windows OS with GPU NVIDIA GeForce GTX 1050, used to train the model. The model trains on video frames of the simulated data consisting of vehicle collisions.

I then labeled the dataset by building bounding boxes for each frame in every video sequence. This process is called annotation, done using a tool called CVAT. In every video, a bounding box is drawn around the vehicle and the object it collides with. A few seconds before the collision, I drew a bounding box around the car and the object involved in the crash, which is the accident bounding box. All the videos were labeled and annotated carefully.

The report used two computers for designing the model. The first is Linux OS with GPU NVIDIA Corporation TU116 [GeForce GTX 1660 Ti], used to generate the data, and the second is a Windows OS with GPU NVIDIA GeForce GTX 1050, used to train the model. The model trains on video frames of the simulated data consisting. After labeling the dataset, I passed it to the Roboflow tool to generate the yolov7 PyTorch format, which can be given as input to yolov7. The report used 70 percent as training data, 10 percent as validated data, and 20 percent as test data. I then trained the yolov7 model on the dataset for 100 epochs and tested it on the test data. The model took approximately 14 hours to prepare and could predict the collision a few frames before the crash took place. The Figure 4 a shows the prediction of the bounding box of collision before the occurrence of the crash at **40 seconds** in the video output. The blue frame represents the collision, white frame represents the vehicle and the green frame the object vehicle collides into. The Figure 4 b shows the collision that occurred at **43 seconds** in the video file. The algorithm predicted the collision **3 seconds** before the

occurrence of the collision. These 3 seconds might be crucial in alerting the user to stop the vehicle or can be used to develop a safety stop after detecting a collision.

Figure 4 a. The figure shows the bounding box prediction of collision before the occurrence of actual collision

Figure 4 b. The figure shows the bounding box while the collision actually takes place

This data collected can be used to help improve how the autonomous driving agent corrects such errors and may help uncover various other anomalies that may be present in the driving algorithms.

The hyperparameters used in this report are from the original YOLOv7 framework [18].

4.2 Results:

The model can identify vehicles and objects and predict the collision before it takes place. I used the evaluation metrics Precision, Recall, and Mean Average Precision to train the model. **Error! Reference source not found.** shows the model's results.

- Precision: The proportion of accurately identified accurate samples to all positively classified samples is known as precision. Precision makes it possible to see how dependable the machine learning model categorizes the sample data as positive. Figure 4 c shows the precision curve with respect to confidence. The graph shows the precision of all the four classes with varied confidence values.

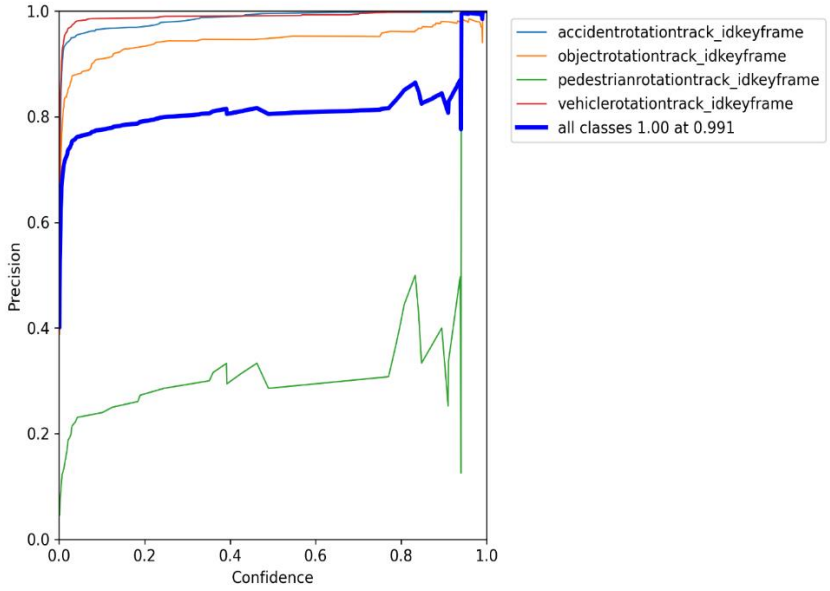

Figure 4 c. The figure depicts the Precision curve of all the classes with respect to the confidence score. Overall, all the classes except for pedestrian class have a precision of over 0.9 as the confidence score rises.

- Recall: The True Positive to All Positive Samples ratio determines recall. Recall evaluates how well a model can identify certain image cases. **Error! Reference source not found.** shows the recall to confidence, and Figure 4 f and the precision-recall curve of all the classes.

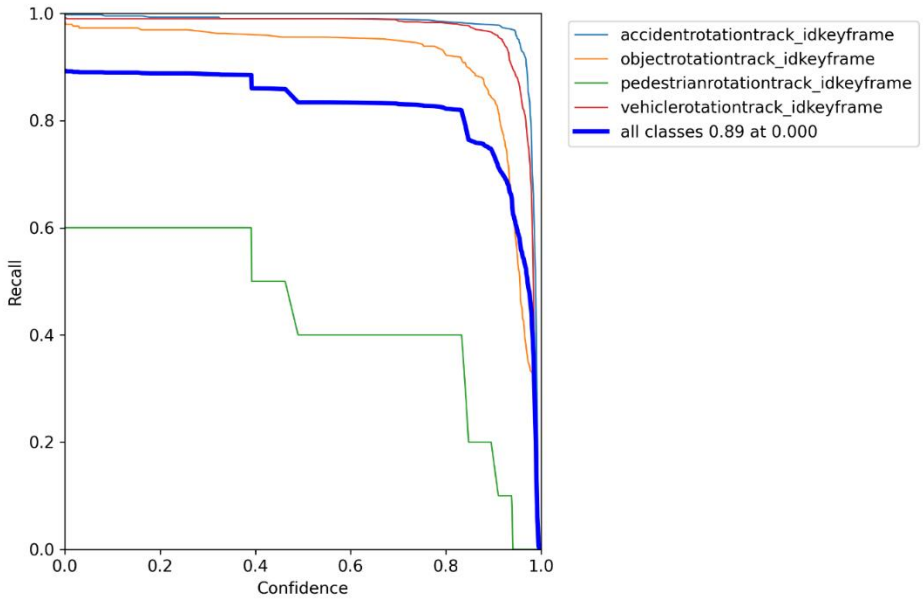

Figure 4 d. The figure shows the recall curve with respect to the confidence score of all the classes.

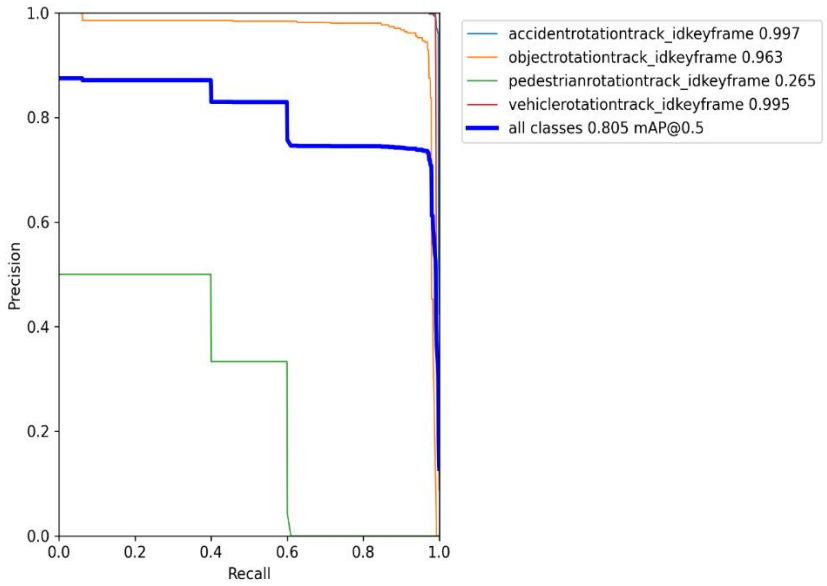

Figure 4 e The figure shows the precision-recall graph of all the classes

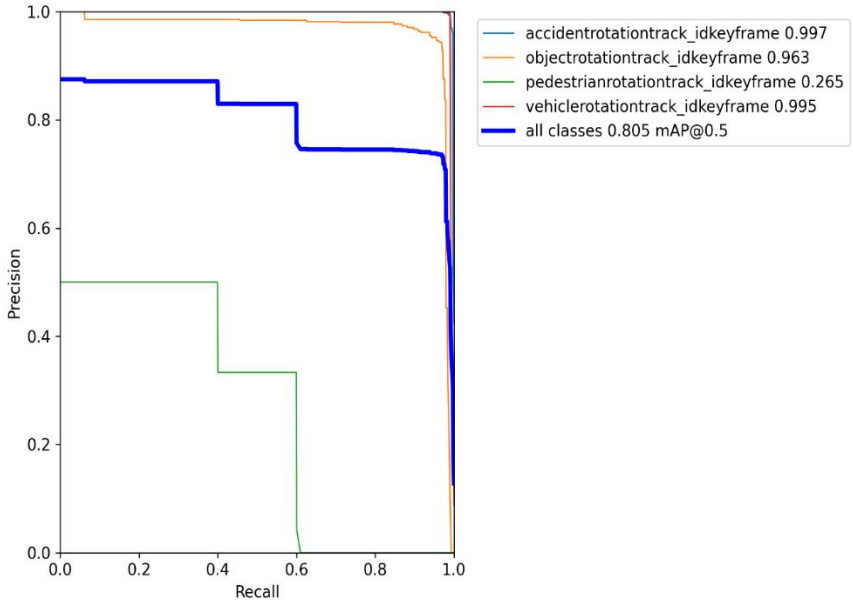

Figure 4 f The figure shows the precision-recall graph of all the classes

- Assessing the identified box to the actual bounding box at an IoU limit of 0.5 yields a value for the mAP@0.5. If there is more than a 0.5 crossover between the anticipated box and the actual bounding box, the data is classified as true positive; otherwise, it is classified as false positive. The better the score, the more accurate the model's detections are.
- "mAP@0.5:0.95" denotes the mean average precision above a range of IoU limits, from 0.5 to 0.95, at 0.05-point intervals.

Evaluation Metric	**Score**
mAP@0.5	0.80
mAP@0.5:0.95	0.71
Precision	0.815
Recall	0.885

Table 1. The table presents the results of the evaluation metrics of the model

- Confusion Matrix: A confusion matrix consists of 4 sections, namely True Positive, False Positive, True Negative, and False Negative. Figure 4f shows the confusion matrix of the model used.

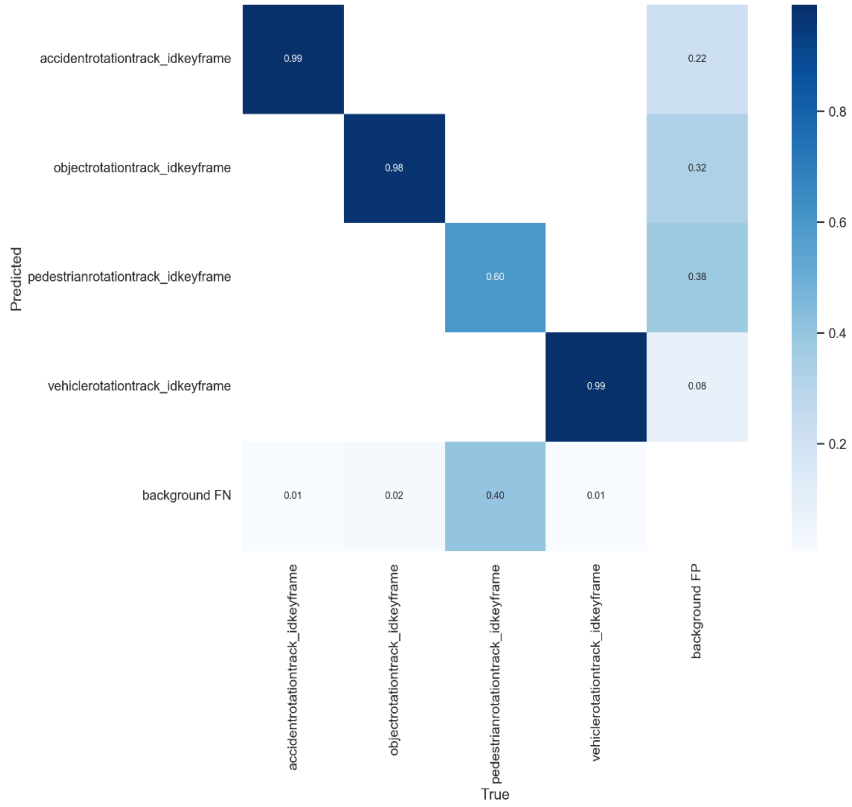

Figure 4 g. The figure shows the confusion matrix of all the classes

- For the 'accident' class, the true positive is 0.99. This value determines the score of the model, where the actual class is 'accident,' and the model predicted 'accident' too.
- The false positive for the 'accident' class is 0.22. This value determines the model's score that predicted a few classes as 'accident' but is not of the 'accident' class. This means that the model falsely predicted the 'accident' class.
- The False negative for the 'accident' class is 0.01. This value determines the model's score that predicted the class as negative (i.e., other class) but is an 'accident' class. The same concepts apply to the rest of the classes.

- F1 Score: The harmonic mean of Precision and Recall is the F1 score. Figure 4g shows F1 curve of all the classes. It considers harmonic mean as its value hugely reduces as precision and recall reduce.

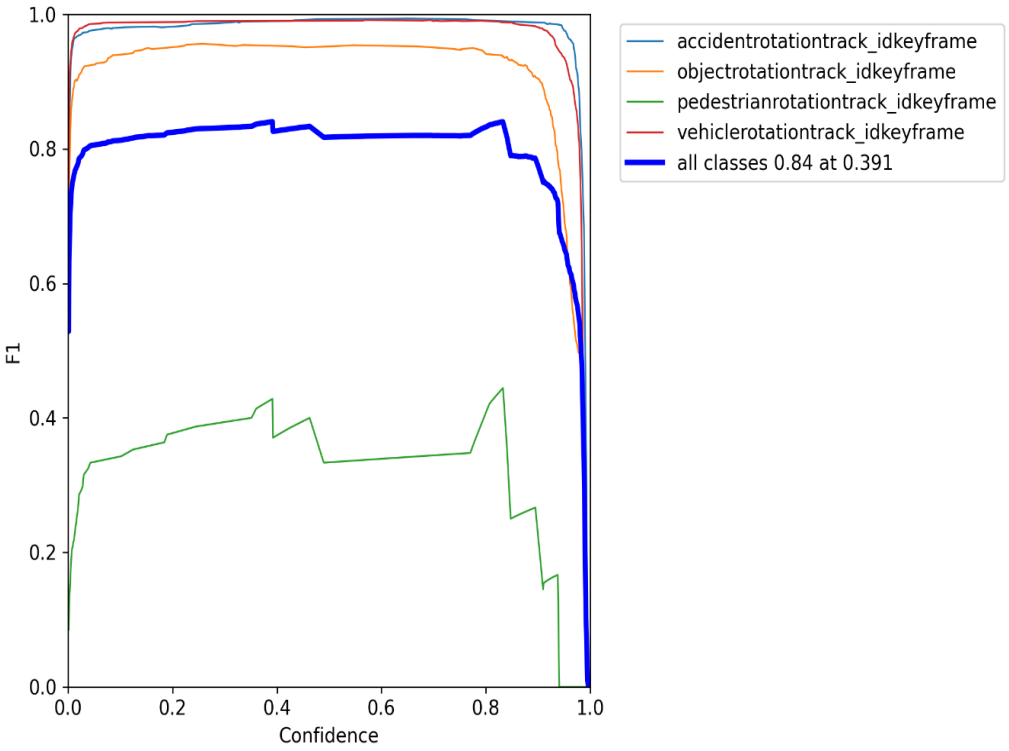

Figure 4 h: F1 curve of all the classes

Chapter 5 Conclusion and Future Work

5.1 Conclusion

Detecting a collision is critical to saving the lives of human beings. In this report, I proposed a collision prediction model by training the YOLOv7 algorithm on a custom dataset. The dataset is from video sequences generated by running simulations on a CARLA vehicle simulator. This simulator was further developed by the members of the Real-Time Embedded lab at Ontario Tech University by connecting real-world autonomous vehicle parts such as the speedometer and pedals for accelerating and applying break. The generated data were analyzed and pre-processed by carefully annotating each class on every frame and labeling it. It was then extracted in Yolo format and passed to the model. The model could successfully predict the collision on test data a few seconds before the crash occurred. The proposed approach could be beneficial in testing autonomous vehicles to avoid accidents and save the life of human beings and other creatures.

5.2 Future work

This report could only produce limited data due to time constraints. In the future, one could expand the dataset by generating more data from the simulator. In that case, there would be an enormous amount of data to analyze the collisions and the behavior of self-driving vehicles. Any specific type of collision could be given more priority, such as pedestrian and two-wheeler crashes, as they would be most affected by any collision. After pre-processing the data, a deep learning model can be trained on the dataset and evaluated using various metrics.

We can test many more variations in the behavior of the autopilot mode of the autonomous car of the simulator by driving the vehicle with different scenarios. The anomalies that arise after testing the car in other conditions can be further analyzed and solved by altering the application's code.

References

[1] J. Fingas, "Engadget," Jun 2022. [Online]. Available: https://www.engadget.com/self-driving-car-technology-crash-data-172606258.html.

[2] D. a. L. P. G. Tola, "A Co-Simulation Based Approach for Developing Safety-Critical Systems," *John Fitzgerald, Tomohiro Oda, and Hugo Daniel Macedo (Editors)*, p. 65, 2021.

[3] J. a. F. A. Redmon, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263--7271.

[4] J. a. F. A. Redmon, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[5] A. a. W. C.-Y. a. L. H.-Y. M. Bochkovskiy, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[6] D. Thuan, "Evolution of Yolo algorithm and Yolov5: The State-of-the-Art object detention algorithm," 2021.

[7] G. a. N. K. a. M. T. a. V. R. Jocher, "YOLOv5 (2020)," *GitHub repository: https://github. com/ultralytics/yolov5*, 2020.

[8] C. a. L. L. a. J. H. a. W. K. a. G. Y. a. L. L. a. K. Z. a. L. Q. a. C. M. a. N. W. a. o. Li, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," *arXiv preprint arXiv:2209.02976*, 2022.

[9] *Video-based algorithms for accident detections*, 2018.

[10] S. a. M.-G. P. a. G.-G. A. a. C.-V. J. A. a. O.-E. S. a. G.-R. J. a. A. A. Oprea, "A review on deep learning techniques for video prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[11] "Deep learning classification applied to traffic accidents prediction," in *XLIII Jornadas de Autom{\'a}tica*, Universidade da Coru{\~n}a. Servizo de Publicaci{\'o}ns, 2022, pp. 964--971.

[12] S. a. S.-T. G. a. B.-B. J. Robles-Serrano, "Automatic Detection of Traffic Accidents from Video Using Deep Learning Techniques," *Computers*, vol. 10, p. 148, 2021.

[13] F. a. Z. X. a. L. B. Yang, "Video object tracking based on YOLOv7 and DeepSORT," *arXiv preprint arXiv:2207.12202*, 2022.

[14] A. a. L.-T. L. a. R. I. a. R. S. a. S. K. Milan, "MOT16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.

[15] D. a. P. S. a. P. M. Patel, "Application of image-to-image translation in improving pedestrian detection," *arXiv preprint arXiv:2209.03625*, 2022.

[16] S. a. H. D. Gautham, "Artificial Intelligence Used in Accident Detection Using YOLO," in *Proceedings of International Conference on Deep Learning, Computing and Intelligence*, Springer, 2022, pp. 457--467.

[17] C. a. K. H. a. O. S. a. D. I. Lee, "A Study on Building a “Real-Time Vehicle Accident and Road Obstacle Notification Model” Using AI CCTV," *Applied Sciences,* vol. 11, p. 8210, 2021.

[18] C.-Y. a. B. A. a. L. H.-Y. M. Wang, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696,* 2022.