TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A

PROJECT REPORT

ON

SMART TRIAL SYSTEM

SUBMITTED BY:

AYUSH SHAKYA (PUL075BEI010)

DEWASHISH ATAL (PUL075BEI012)

PRASHANNA KARN (PUL075BEI022)

PRASHANNA RAJ PANDIT(PUL075BEI023)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

March, 2023

Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **"Smart Trial System"** submitted by **Prashanna Raj Pandit**, **Ayush Shakya**, **Dewashish Atal** and **Prashanna Karn** in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

.....
Supervisor
Surendra Shrestha
Associate Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

.....
Internal examiner
Person B
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

.....
External examiner
Person C
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

Date of approval:

Copyright

Acknowledgement

We would like to express our sincere gratitude to all those who have contributed to the successful completion of our project, "Smart Trial System". We recognize all the teachers whose contributions have been undeniably helpful and their valuable assistance in the preparation of this project has been of great importance. We would like to thank the Department of Electronics and Computer Engineering, IOE, Pulchowk Campus, and academic authorities for providing necessary guidelines, suggestions and support. We especially recognize our supervisor, Assoc. Prof. Surendra Shrestha, for his priceless and meticulous supervision, advice and vital contribution during the research of this project.

Further on, we also acknowledge the role played by the seniors in guiding us through earlier projects and setting the benchmark which motivates us. Finally, we would like to express our appreciation to all those who have directly or indirectly contributed to our project. We have gained valuable insights and experiences from working on this project, and we hope that our work will be of benefit to others in the field of computer vision. We have not just learned the content of the project, but also learned group work and problem-tackling among the group members while researching for the project. Therefore, we thank all the concerned authorities for their valuable contributions.

Abstract

Driving is a common and important mode of transportation in the daily lives of people. It allows people to move freely and independently from one place to another. To drive legally in Nepal, as in most countries, a person requires a driving license, for which he/she has to undergo a series of tests in the government office. In Nepal, a candidate applying for a license has to drive over a fixed path in front of the authorities. The candidate has to complete the test in accordance with specific rules and if he fails to do so he/she will be disqualified. Manually monitoring the errors of the candidates is done by these authorities. Also, the issue of bribery for obtaining a driver's license is a significant problem in Nepal. To address this issue, we have proposed an AI-based solution named CarSight. CarSight is a Computer Vision model designed to analyze visual data from cameras that verifies whether a driver has passed or failed the driving test. It uses YOLO-v5 to detect and track the motion of the trial vehicles in a set of real-time videos obtained from cameras placed around the model track. The system gives an instant pass or fails result depending on the driver's performance once they finish the trial exam. The accuracy of detecting and evaluating the driver's performance was successfully demonstrated by our experimental results for our system. Our proposed system can help to bring fairness in the process of obtaining a driver's license.

Keywords:*trial center, trial exam, YOLO object detection, image processing, illegal license, CarSight*

Contents

List of Figures

List of Abbreviations

AI	Artificial Intelligence
CIAA	Commission for the Investigation of Abuse of Authority
CNN	Convolutional Neural Network
CSP	Cross Stage Partial
CSV	Comma Separated Values
DFD	Data Flow Diagram
DOTM	Department of Transport Management
GPU	Graphics Processing Unit
GPS	Global Positioning System
GSM	Global System for Mobile communication
HSV	Hue Saturation Value
IoU	Intersection Over Union
IOT	Internet of Things
IR	InfraRed
JSON	JavaScript Object Notation
ML	Machine Learning
mAP	mean Average Precision
MP	Megapixel
NMS	Non-Maximal Suppression
OpenCV	Open Computer Vision
PANet	Path Aggregation Network
R-CNN	Region-based Convolutional Neural Network
R-FCN	Region-based Fully Convolutional Network
RPN	Region Proposal Network
SMS	Short Message Service
SPP	Spatial Pyramid Pooling
SSD	Single Shot Multi-Box Detector
STS	Smart Trial System
VR	Virtual Reality
YOLO	You Only Look Once

1. Introduction

The project is aimed at real-time evaluation of driving license trial examination using computer vision. The system detects and tracks the motion of 4-wheeler vehicles during the trial and evaluates the completion of the examination, according to government regulations.

1.1 Background

A driver's license is an official document permitting a specific individual to operate one or more types of vehicles, such as a motorcycle, car, truck, or bus on a public road. Nepal contains various categories of driving licenses, ranging from A, B, to K for different types of vehicles. The Department of Transport Management under the Ministry of Physical Infrastructure and Transport handles the issuing of driving licenses in Nepal. In the current scenario, a Nepali citizen can get a driving license through the following steps.

1. Online Registration form: A candidate should apply for a driver's license in an online portal of the Department of Transport Management. The website for the application is https://applydl.dotm.gov.np. The candidate has to verify their phone number to access the website. The candidate must create an account and fill the required personal details, address and citizenship details. He may then apply for a driving license by selecting the correct license category, location for office visit and date of visit.

2. Office Visit: On the chosen appointment date, the candidate must visit the transport office. During this visit, the online registered data are verified and new data like signature, thumbprint and photograph are taken. The candidate has to undergo several medical tests including eyesight check and Color Plate test for color-blindness. The payment of the form and submission of required documents is done on the same day.

3. Written Exam: The candidate has to undergo an on-premise written test which contains 20 questions related to driving and regulations in Nepal. The pass marks for the exam are 50%, and the results can be received within 24 hours through SMS service, Facebook page, and website. Candidates who failed the written test can retake it after a week for a maximum of 3 retakes. If a candidate fails in all attempts, they should reapply for the license again after 3 months and should go through all the steps again.

4. Trial Exam: A candidate has to complete the trial examination in the pre-specified trial

center. He/she must strictly follow all the traffic rules while giving the examination. The trial exam includes a total of six tests: 8-test, speed breakers, u-turn, traffic lights, garage parking, and ramp. There is a provision of 70% cutoff score for the examination. If the candidate fails any of the mentioned tests he/she is disqualified. They will get a total of 3 chances to pass the trial and there should be at least a 10 days gap from each trial in case they failed.

5. Payment and License: Once the candidate has passed the trial, final payment should be done after 10 days and they will be able to legally drive with the payment slip until a digital license is issued. Payment slips should be submitted to get a smart license. The status of digital license issuance can be checked through SMS service.

Trial Examination In Nepal for a four-wheeler:

(a) 8-test: The beginning of the trial test starts with the 8-test. The candidate has to drive the vehicle in between the 8-lane. The width of the track is 3m and the total length is 21m. If the vehicle touches the lane or if the engine stops, the candidate is considered fail. The candidate should take a complete and a half round of the '8' shaped track. When the candidate completes the 8-test without failing, he moves forward for the speed breaker test.

(b) Speed Breakers (Humps): Once the candidate passes the 8- test, he moves forward for the speed breaker test. There are five humps each of width 4m and the total length of 16m. If the engine stops or if the vehicle slips back, the candidate is considered fail.

(c) U-turn: The candidate, after passing the speed breaker test moves forward for U-turn test. The width of the track is 4m and the radius of the track is 7.2m. Here the candidate should light the right indicator before entering the U- shaped track and after crossing all the humps. The indicator should be turned immediately after getting out of the U- shaped track. If the indicator is lit before crossing all the humps or after entering the U-shaped track, the candidate is considered fail. If the candidate does not turn on the indicator at all, he is considered fail. If the candidate touches the lane of U-shaped track, he is considered fail. The candidate should turn off the indicator immediately after crossing the U-shaped track and before entering the Traffic light and zebra crossing test.

(d) Traffic lights: This test consists of traffic lights and zebra crossing. The candidate will fail if they fail to follow the traffic signals.

(e) Garage parking: The width of the track is 3m and total length of 20m. The candidate i considered failed if the vehicle moves forward while reversing. The candidate will fail the entire test if the vehicle moves forward more than once and if the vehicle touches the specified line.

(f) Ramp: The ramp is 5m wide and 23m long. The candidate will fail the entire test if they stop the vehicle over the specified line or if they move ahead without stopping the vehicle on the line. When the candidate has to move ahead after stopping the vehicle but if it slips behind more than 6 inches than the specified line, the candidate is considered failed the entire test. The candidate also fails the entire test if the vehicle crosses the line upon stop while descending or if it moves ahead without stopping and if the vehicle stops more than once.[1]

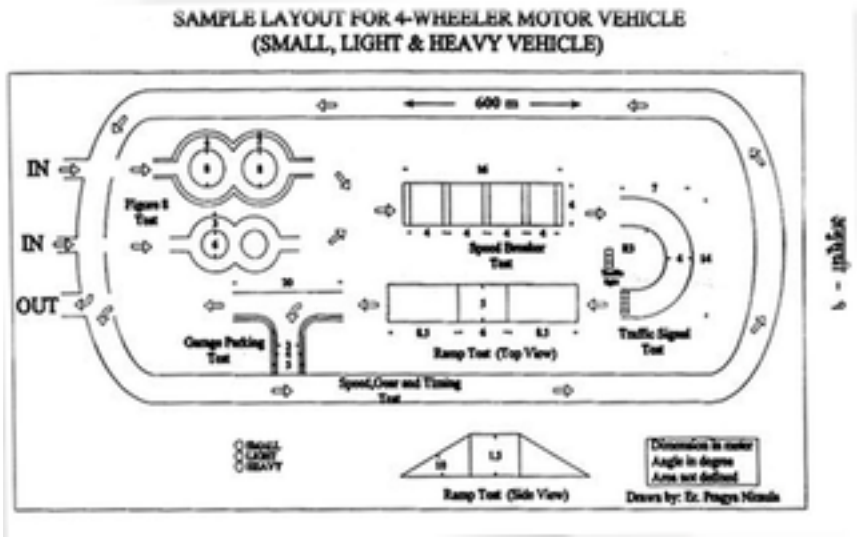


Figure 1.1: Sample Track for 4-wheeler Vehicle

(Source: Department of Transport Management)

1.2 Problem statements

The current trial system is fully manual with the examinee being observed by traffic officers. The major problem associated with the current system is the fake license issue which lends to corruption. Due to the manual system, there are a lot of possibilities for fake license issues. Fake licenses are issued mainly in two ways. one through "proxy" candidates and another by marking "pass" to the failed candidate. Some candidates have other people perform their trial examination; while many failed candidates bribe brokers to get licenses. Some such occurrences can be seen in the past. On March 12, 2023, police arrested four people for a fake driving license racket, where the group was found charging up to 90,000 for providing a fake driving license[2]. Similarly, on March 18, 2022, CIAA filed a case against nine people for altering driving license results [3]. The accused group altered the results where an unsuccessful candidate was mentioned as 'pass' and the ones who successfully got through all the tests were stated as 'failed' ones and the ones who did not appear in the examination were also put in the group of 'pass' candidates. These cases mostly occur with the joint agreement of traffic officials and brokers. As a result, people who hardly know how to ride the vehicles, get a license, resulting in accidents and large amounts of casualties. To solve this problem, we have designed an AI-based system that can automate the trial examination system using Computer Vision technology.

1.3 Objectives

- To use Computer Vision and image processing models to accurately evaluate the results of trial examinations

1.4 Scope

The project will be of great assistance to trial centers in Nepal in automating the trial process. The proposed system can be initially implemented in test centers across Kathmandu Valley, but can later be expanded to all parts of Nepal. This can also be useful for drivers and prospective candidates to check their performance in the driving exams. This system of detecting vehicles can further be used for traffic management across roads to track vehicles and detect the vehicles that are breaching traffic rules.

2. Literature Review

2.1 Related work

There have been many projects conducted worldwide in order to modernize the process of driving license trials and driving as a whole. These systems use a wide range of techniques such as computer vision, image processing, Internet of Things, sensors, etc to detect and track vehicles. This literature review aims to provide an overview of the current state of the technology in smart trial systems.

An Arduino-based system with necessary sensor modules is developed for observing the candidate for getting their license. In this system, the candidates who take up the test were monitored and the result whether the candidate passed or failed was updated to the candidate as well as the authorities wirelessly using IOT and GSM modules. This monitoring of the driving test ground was done autonomously using the Arduino system.[4]

An automated driving test using wireless sensor networks has also been developed in which the applicant is allotted the test vehicle for a test drive with the number of sensors embedded in the vehicle sending data using a wireless sensor network to a remote server to get processed. Result analysis is done by comparing the received data with previous data. This system is implemented using a Bayesian logic classification algorithm and feature extraction algorithm. The proposed system needed to be designed for the wireless sensor network and also the multi-sensor fusion-based detection approach for detecting results.[5]

The technological solution for the trial center was also developed by customizing an 8051 controller-based embedded system and VB-based virtual instrument. This system used the controller module to sense the motion of the test vehicle on the test track and to evaluate the performance of the test driver based on the metric known as zero rpm measurement.[6]

A smart driving test system is also proposed that uses IoT technologies to monitor

and evaluate a driver's behavior during a driving test. The system included multiple sensors, such as cameras, GPS, and accelerometers, to collect data about the driver's driving performance and behavior.[7]

A system that uses a combination of facial recognition and heart rate sensors to detect and prevent impersonation during driving license tests was also proposed. The system was found to be effective in preventing cheating behavior, with a high level of accuracy.[8] Similar research work and projects have been conducted to check the effectiveness of computer vision on driving performance. The proposed system uses computer vision and machine learning algorithms to detect and analyze the behavior of drivers during their driving license tests. The system was found to be effective in detecting unsafe driving behavior, such as speeding and reckless driving.[9] In a study conducted by Zhang et al, the performance of a smart trial system was found to be more effective than human examiners in detecting unsafe driving behaviors. The system included a camera placed in the test vehicle to capture images of the driver and the road ahead, and utilized features such as the driver's head orientation, eye gaze, and hand movement to evaluate the performance based on criteria such as adherence to traffic rules, safe driving behavior, and overall driving skills.[10]

A smart trial for driving license detection system is also implemented in a driving test center using multiple cameras to take videos of the vehicle and road. It used machine learning algorithms such as CNN to identify patterns of cheating behaviors and distinguish them from normal driving behaviors. The authors concluded that "the proposed system could help prevent cheating during driving license tests and improve the fairness and reliability of the test results."[11] A smart driving license test system that uses virtual reality (VR) technology that allows the driver to experience different driving scenarios in a safe and controlled environment. The system also uses machine learning algorithms to evaluate the driver's performance and provide feedback.[12]

2.2 Related theory

2.2.1 YOLO Object Detection

YOLO (You Only Look Once) is a popular object detection algorithm that has been widely used in computer vision applications. It is based on deep neural network architecture and is designed to be fast and accurate. One of the key features of YOLOv5 is its ability to detect objects at different scales, using a range of anchor boxes. This allows the algorithm to accurately detect objects of different sizes and shapes, even in cluttered or complex environments. YOLOv5 also incorporates several other improvements over previous versions of the YOLO algorithm, including a more efficient backbone architecture, a new focus mechanism that allows the network to attend to relevant features more effectively, and a more accurate bounding box regression mechanism. Overall, YOLOv5 is a powerful and flexible object detection algorithm that has the potential to be applied in a wide range of computer vision applications, including the detection and tracking of vehicles, pedestrians, and other objects in real time.
The system divides the image into an S x S grid. Each of these grid cells predicts B bounding boxes and confidence scores for these boxes. The confidence score indicates how sure the model is that the box contains an object and also how accurate it thinks the box that predicts. The confidence score can be calculated using the formula:
C = Pr(object)* IoU
IoU: Intersection over Union between the predicted box and the ground truth.
If no object exists in a cell, its confidence score should be zero.

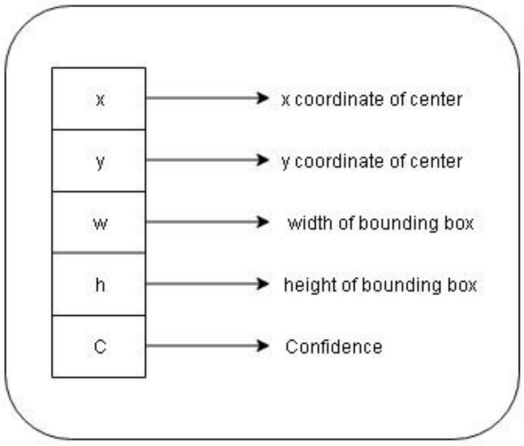


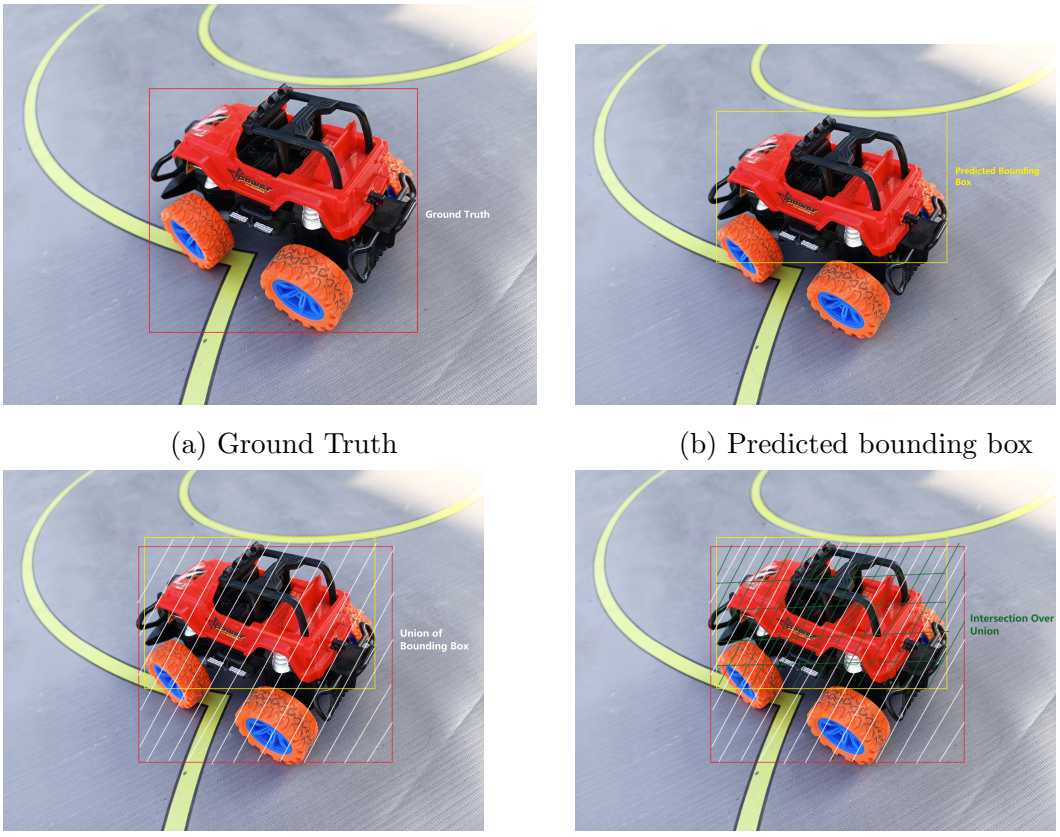
Figure 2.1: Parameters for YOLO Object Detection

Each grid cell also predicts C conditional class probabilities Pr(Class i I Object). It only

predicts one set of class probabilities per grid cell, regardless of the number of boxes B. During testing, these conditional class probabilities are multiplied by individual box confidence predictions which give class-specific confidence scores for each box. These scores show both the probability of that class and how well the box fits the object.
Pr(Class i I Object)*Pr(Object)*IoU = Pr(Class i)*IoU.
The final predictions are encoded as an S x S x (B*5 + C) tensor.

2.2.2 Intersection Over Union(IoU)

IoU is used to evaluate the object detection algorithm. It is the overlap between the ground truth and the predicted bounding box, i.e. it calculates how similar the predicted box is with respect to the ground truth.



(a) Ground Truth

(b) Predicted bounding box

(c) Union of bounding boxes

(d) Intersection of bounding boxes

Figure 2.2: Bounding Box Formation

Usually, the threshold for IoU is kept as greater than 0.5. Although many researchers apply a much more stringent threshold like 0.6 or 0.7. If a bounding box has an IoU less than the specified threshold, that bounding box is not taken into consideration.

2.2.3 Non-Max Suppression

The algorithm may find multiple detections of the same object. Non-max suppression is a technique by which the algorithm detects the object only once. Consider an example where the algorithm detected three bounding boxes for the same object. The boxes with respective probabilities are shown in Figure 2.6

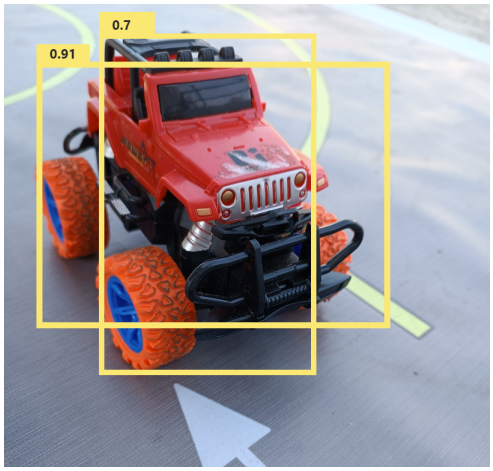


Figure 2.3: Probabilities of Multiple Detected Object

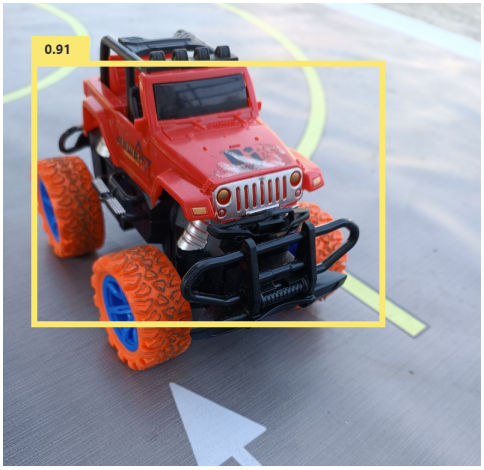


Figure 2.4: Object with Higher Probability Chosen

The probabilities of the boxes are obtained as 0.7 and 0.91. To remove the duplicates, the box with the highest probability is selected and output as a prediction. Then any bounding box with IoU less than 0.8 (or any threshold value) are eliminated to obtain the predicted output. The result will be as shown in Figure 2.4.

2.2.4 Bounding Box Prediction

In YOLO, a single neural network is applied to an image, which divides the image into many regions and predicts bounding boxes and probabilities for each region. These bounding boxes are given their own weighted probabilities. The predictions are made with information of the global context of the image.[4]

Figure 2.5: Grid Division of Input Image

Figure 2.6: Bounding Box Creation in Grid

For multiple objects without any overlap, YOLO works soundly as one grid works for one object. But for objects with overlap, anchor boxes are used. This helps to create a longer grid cell vector where multiple classes are associated with a single grid cell. Each of these anchor boxes are associated with their weighted probabilities. The anchor boxes with low probability, i.e., less likely, are then removed with the help of NMS. A certain NMS threshold is pre-determined and the predicted bounding boxes with detection probability less than the threshold are then gotten rid of. After this, the bounding boxes with the highest detection probabilities are selected and the other boxes with IoU value greater than a pre-specified threshold are removed.

The process continues till only the bounding box with highest detection probability remains for a single object and all non-maximal bounding boxes have been removed for every class.[13]

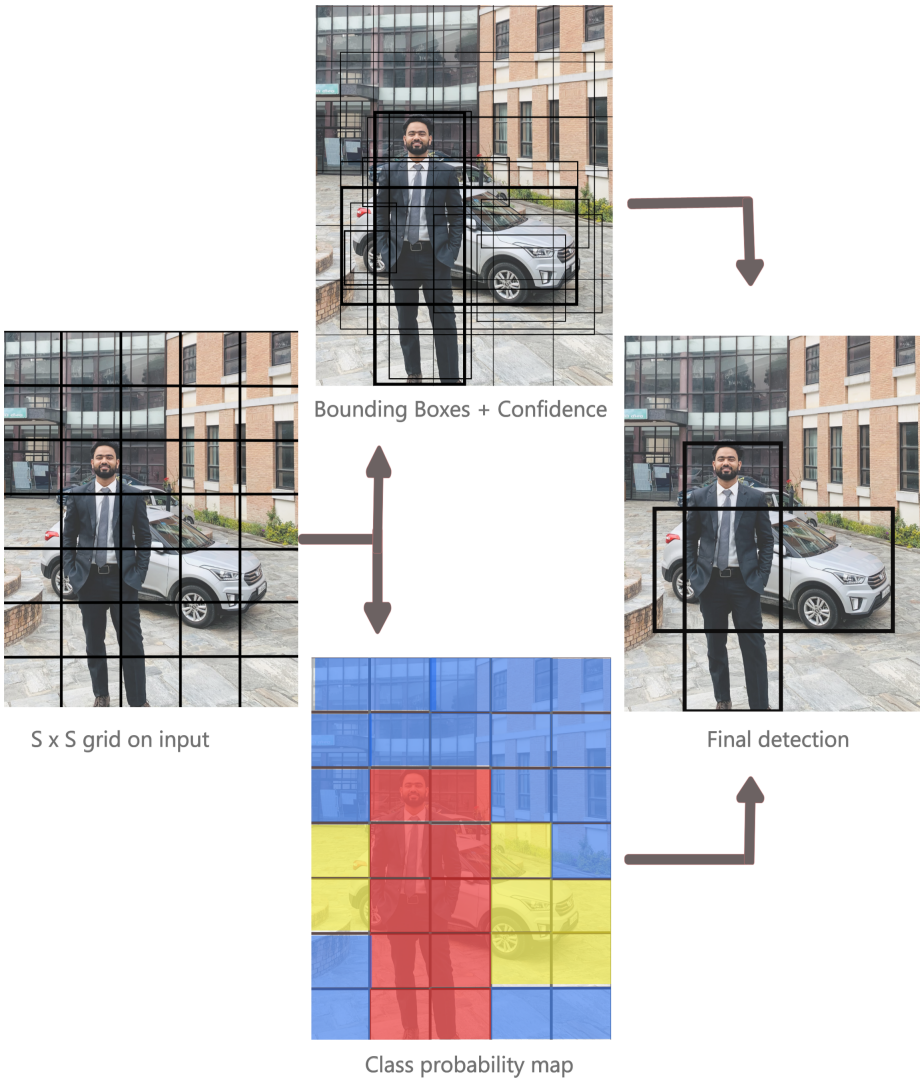


Figure 2.7: Working of YOLO Object Detection

2.2.5 Object Detection Models

Object detection models are computer vision models that are capable of detecting and localizing objects within an image or a video stream. They are usually trained to detect the presence of specific objects in images, videos, or real-time operations. Object detection algorithms have a large variety of applications, ranging from number plate recognition, face detection and recognition, object tracking, self driving cars to robotics. Object detection algorithms can be broadly classified int two types: multi-stage detection models and single stage detection models.

Multi-stage Detection Models

Multi-stage object detection models are a type of deep learning model used to detect and classify objects in images or video that use a two-stage approach to detect objects. The first stage of a multi-stage object detection model is object proposal generation, where potential object locations are generated. The second stage then classifies these proposals and refines their position, producing the final detection result. Some popular multi stage detection models are SPP-Net, Faster R-CNN and Feature Pyramid Networks. Multi-stage object detection models typically consist of two main components: a proposal generation network and a classification network. The proposal generation network is responsible for generating a set of object proposals, which are regions in the image that are likely to contain objects. The proposal generation network can be implemented using various techniques such as region proposal networks (RPNs), fully convolutional networks (FCNs), or other methods. The classification network is responsible for taking the object proposals generated by the proposal generation network and classifying them into different object categories. It is implemented using CNNs, which are trained on large datasets to learn features that are relevant to object detection.

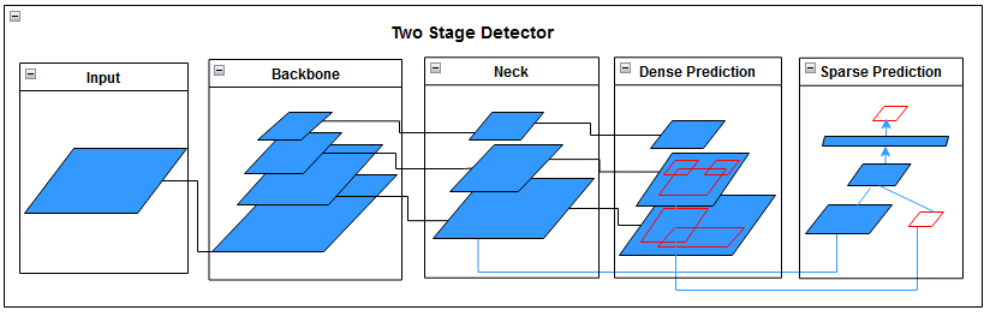


Figure 2.8: Two Stage Detection Model for Object Detection

Single-stage Detection Models

Single-stage object detection models are deep learning models that generate object detection directly in a single step, without an explicit proposal generation step. They are faster and more computationally efficient than multi-stage models, making them suitable for real-time applications. The architecture of an SSD model typically consists of a base network that extracts features from the input image, followed by several detection layers that predict object bounding boxes and class probabilities. Each detection layer predicts object detections at a different resolution, allowing the model to detect objects of different sizes. Single-stage object detection architecture are composed of three components: backbone, neck and a head to make dense predictions. The backbone is a pre-trained network used to extract rich feature representation for images which helps reducing the spatial resolution of the image and increasing its feature (channel) resolution. The model neck is used to extract feature pyramids and helps the model to generalize well to objects on different sizes and scales. The model head is used to perform the final stage operations like applying anchor boxes on feature maps and rendering the final output: classes, objectness scores and bounding boxes. YOLO and Single Shot Multibox Detector are some popular single stage object detection algorithms.

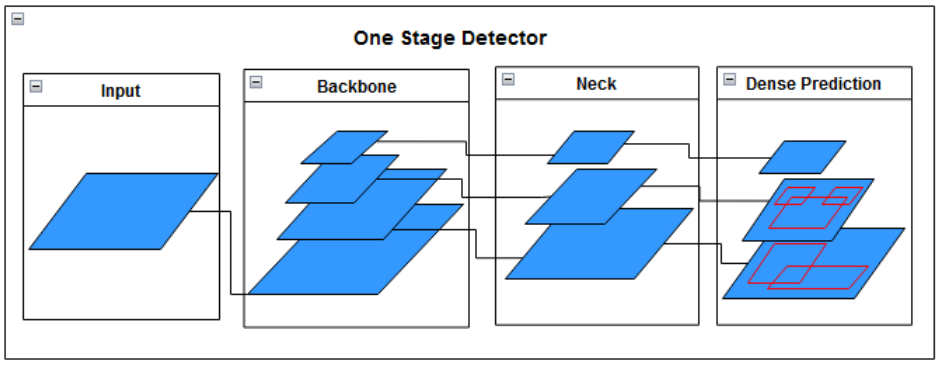


Figure 2.9: Single Stage Detection Model for Object Detection

2.2.6 Network Architecture

YOLO-v5 models are composed of the same 3 components: CSP-Darknet53 as a backbone, SPP and PANet in the model neck and the head used in YOLOv4. The CSP network preserves the advantage of DenseNet's feature reuse characteristics and helps reduce the excessive amount of redundant gradient information by truncating the gradient flow. YOLOv5 employs CSPNet strategy to partition the feature map of the base layer into two parts and then merges them through a cross-stage hierarchy.

In the neck of YOLO-v5, PANet has been modified by applying the CSPNet strategy to it and is used to improve information flow and to help in the proper localization of pixels in the task of mask prediction. SPP block performs an aggregation of the information that receives from the inputs and returns a fixed length output. The head of YOLOv5 is the same as YOLOv3 and YOLOv4. It is composed from three convolution layers that predicts the location of the bounding boxes (x,y,height,width), the scores and the objects classes.

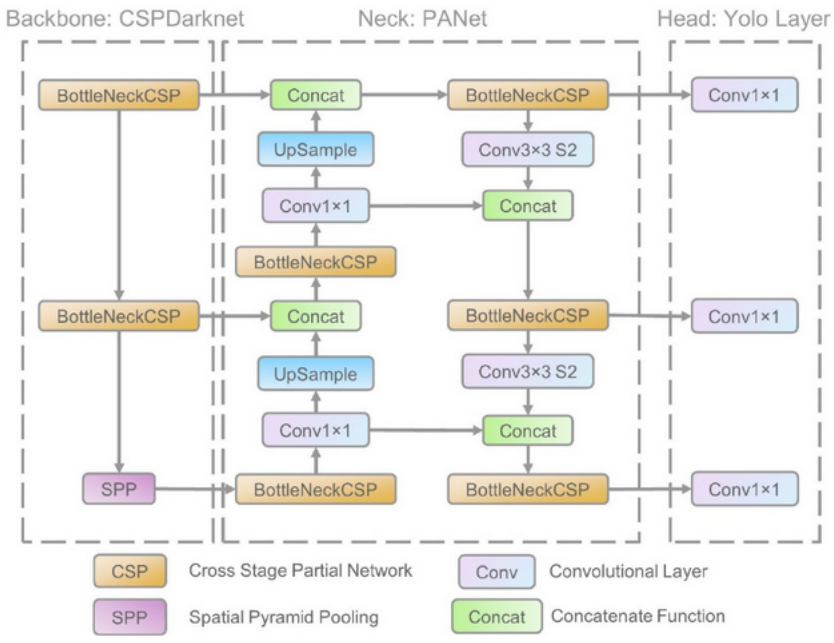


Figure 2.10: Network Architecture of YOLO-v5 Object Detection Algorithm

YOLO Backbone

CSP-Darknet53 is a convolutional neural network architecture used as the backbone network for the YOLO-v5 object detection model. It is formed by the application of CSP network strategy on the Darknet convolutional model. YOLOv5 employs CSPNet strategy to help reduce the number of parameters and reduce computation which leads to increase in the inference speed of the object detection model.

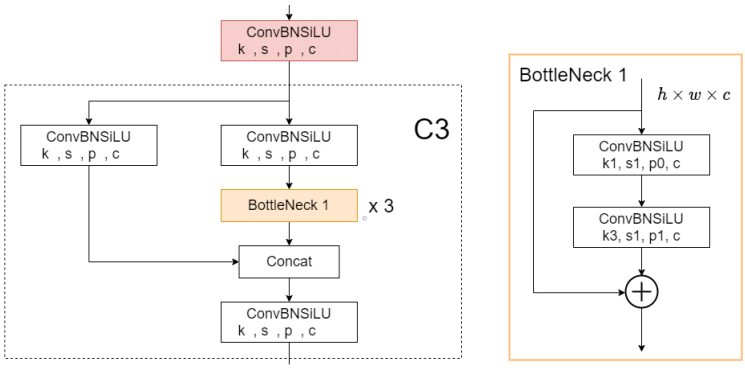


Figure 2.11: BottleneckCSP module architecture

YOLO Neck

The neck of YOLO model is a set of intermiediate layers including the spatial pyramid pooling module, path aggregation network and the upsampling and concatenation layers. The SPP module is used to generate feature maps at multiple scales, allowing the network to detect objects of different sizes. The upsampling and concatenation layers are used to merge the feature maps from different stages of the network, allowing the network to generate more accurate object detections. The PAN module is used to aggregate features from different stages of the network, allowing the network to learn more complex representations of the input image.

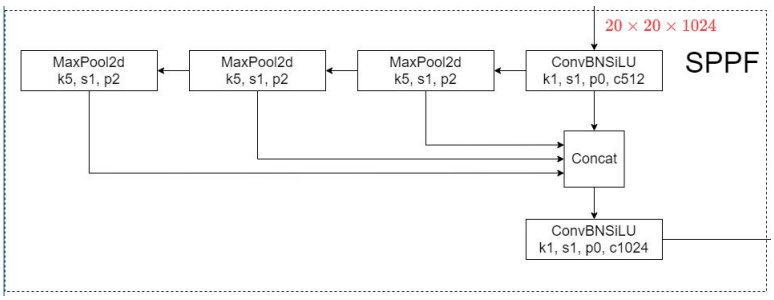


Figure 2.12: Structure of SPPF module

YOLO Network Head

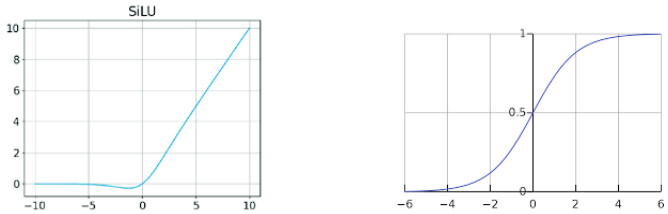
In YOLOv5, the head consists of several convolutional layers that predict the bounding boxes and class probabilities for each object in the input image. It is composed from three convolution layers that predicts the location of the bounding boxes (x,y,height,width), the scores and the objects classes.

$$\begin{aligned} b_x &= (2.\sigma(t_x) - 0.5) + c_x \\ b_y &= (2.\sigma(t_y) - 0.5) + c_y \\ b_w &= p_w.(2.\sigma(t_w))^2 \\ b_h &= p_h.(2.\sigma(t_h))^2 \end{aligned} \tag{2.1}$$

The equations are used to compute the target coordinates for the bounding boxes in YOLO-v5.

Activation Function

YOLO-v5 uses Sigmoid Linear Unit (SiLU) with the convolution operations in the hidden layers and the Sigmoid function with the convolution operations in the output layer as the activation functions.



(a) SiLu Function (b) Sigmoid Function

Figure 2.13: Activation Functions for YOLO-v5

Loss Function

YOLOv5 returns three outputs: the classes of the detected objects, their bounding boxes and the objectness scores. Thus, it uses Binary Cross Entropy(BCE) to compute the classes loss and the objectness loss. and Complete Intersection over Union(CIoU) loss to compute the location loss.

$$Loss = \lambda_1 L_c ls + \lambda_2 L_o bj + \lambda_3 L_l oc \tag{2.2}$$

3. Methodology

3.1 Requirement Analysis

3.1.1 Functional Requirements

The functional requirements of our system are:

(a) The system must contain a camera module to record the video of the trial examination.

(b) The system must be able to detect the vehicle in real time over the course of the trial.

(c) The system must be able to identify the proper instances where the examinee has broken the rules.

(d) The system should give information on the success of the examinee in real time

3.1.2 Non-Functional Requirements

These requirements are not needed by the system for its completion but are essential for the better performance of the Smart Trial Detection System. The points focus on the non-functional requirement of the system.

(a) Reliability: This system should be reliable and robust enough to handle video in various environmental conditions.. We intend to make the system as reliable as possible by performing various optimization and error-handling techniques.

(b) Maintainability: The system will be modularized and divided into smaller sub modules during development. Each subsection of the trial exam will have a separate module. As a result, this system will be easy to maintain because each sub-module will be easy to check and manage.

(c) Accuracy: The result given by the system should be exact to that of a human examiner.

(d) Performance: The smart trial detection system should provide fast and accurate results within a reasonable time frame.

3.1.3 Hardware Requirements

Camera Module: A camera module is an optical sensor that is used to capture images and videos. Camera modules vary in size and resolution. The resolutions range from 0.1 Megapixels (MP) to 13 MP, depending on the types of components used. A camera Module is required to capture the incoming video footage of the trial. It is needed to observe all aspects of the trial center. It is also necessary to detect and verify the identity of the trial applicant by using facial detection.

Dahua technology 2MP IR bullet network camera have been used in this project. It is a smart IR camera with PoE having a wide voltage range of DC 12V ± 30% and high definition image sensor.The camera is shown in Figure 3.1

Figure 3.1: Dahua IR network camera

Remote-Controlled Car:A remote-controlled car is used as the test subject for the data collection and demonstrations. We have modified the remote control car to include both sidelights. The range of RC car ranges from 2 to 3 meters. The car is shown in Figure 3.2.

Figure 3.2: Remote Controlled Car

Server: Servers are necessary to handle the processing of incoming video footage. It is connected to the camera modules and is responsible for the handling of the footage. It handles all the video streams coming from all six cameras. It also helps to handle the processing of data and maintain intermodular communication. The video of trial test is also stored at server at the end.

3.1.4 Software Requirements

Python:
Python is a high-level programming language that contains multiple in-built standard libraries for performing a multitude of tasks. Python supports multiple programming paradigms and is one of the most versatile and popular languages used in the world. One of the most common uses of Python is in the field of AI and ML.

OpenCV:
OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library that supports model execution for Machine Learning (ML). It was developed to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. OpenCV library contains more than 2500 optimized algorithms which can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, etc.

YOLO:
YOLO (short for 'You Only Look Once') is an object detection algorithm that divides images into a grid system. The individual cells in the grids are then responsible for the detection of objects in the object. YOLO looks at the image once, then creates blob objects for the image. The individual objects present in the image are then detected by the network and then separated using a bounding box.

NumPy:
NumPy is a standard library in Python that helps in dealing with numerical calculations while implementing the model. NumPy consists of parallel computing which greatly simplifies the execution of the model while also reducing its size.

Matplotlib:
Matplotlib is a data visualization library in Python that will help us generate the accuracy, loss, classes scores, heat maps and confusion matrix reports of the model.

Google Colab:
Google Colab (or Colaboratory) is a free Jupyter notebook environment provided by Google that runs in the cloud and stores its notebooks in Google Drive. Colaboratory

allows us to train the model for a large number of epochs without any hardware or GPU restrictions. Using a cloud-based service like Google Colab enables us to get more accurate results.

Torch:
In the context of deep learning and computer vision, Torch is a popular open-source machine learning library, which is often used for building deep neural networks. In the context of object detection, a Torch model plays a significant role in the overall object detection pipeline. A Torch model is typically used as the backbone of the object detection network. This model is responsible for extracting features from the input image, which can then be used by subsequent layers of the network to detect objects in the image. The Torch model is pre-trained on a large dataset to learn a set of generic features that can be used for object detection.

3.2 Making CarSight:

CarSight is a custom object detection model which is built using YOLO(You Only Look Once) version 5. YOLOv5s is an open-source object detection framework based on deep learning. Following are the steps to train CarSight.

Data Collection and Labelling:
An important starting task is to collect data for the project. The required pictures of the data were taken using camera modules placed in the trial center. The images need to be taken from different parts of the trial process, including the 8-section, ramps, and U-turn. As there are only two classes to be detected i.e. car and sidelight, approximately 500 images were collected. About 200 images contained only the car and the rest of the images contain both car and sidelights. The image data obtained are then collected and stored in .jpg format. The collected data was annotated using a labeling tool. The dataset of images contains cars, sidelights, and traffic lights along with the annotations that specify the bounding box of each object.

Dataset Splitting:
The dataset is then split into two parts: 80% training data and 20% testing data. Training data is the portion of the actual dataset that is fed into the machine learning model (CarSight) to discover and learn patterns. This helps to train CarSight how to perform. Once the CarSight is built, previously unseen data is fed into the model to check its performance. This dataset is known as testing data and helps to evaluate the progress of the CarSight in terms of accuracy and precision, and helps to optimize it for future use. divide the labeled dataset into two parts:
(a) Training Set: used to train the model
(b) Validation Set: used to evaluate the model during training.

Dataset Formatting:
YOLOv5 requires the dataset to be in a specific format, such as COCO JSON or YOLOv5 txt. so we converted the annotated data into .txt format.

Model Training:
We used the YOLOv5s training script to train CarSight. YOLOv5s is the smallest version of YOLOv5 that contains 8 convolutional layers in the backbone network. To train CarSight, first, we cloned the YOLOv5 repository and installed the requirements

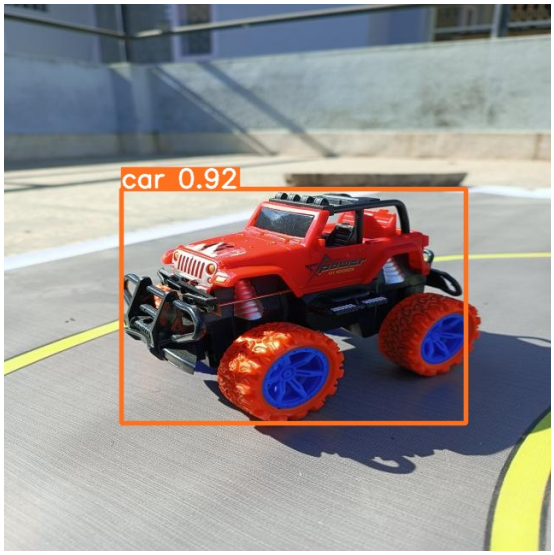
such as the Torch library and "os" module. In the context of deep learning and computer vision, Torch is a popular open-source machine learning library, which is often used for building deep neural networks. In the context of object detection, a Torch model plays a significant role in the overall object detection pipeline.

A Torch model is typically used as the backbone of the object detection network. This model is responsible for extracting features from the input image, which can then be used by subsequent layers of the network to detect objects in the image. The Torch model is pre-trained on a large dataset to learn a set of generic features that can be used for object detection.

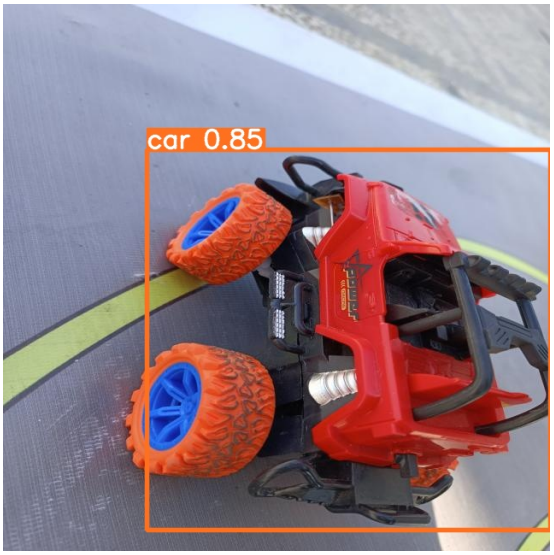
Once the Torch model has extracted features from the input image, these features are typically fed into additional layers that perform object detection. These layers may include region proposal networks (RPNs), which are used to generate proposals for object locations, and classification and regression layers, which are used to classify the proposals and refine their locations. After installing all the requirements we configure the batch size of 16 and epochs of 150 to train CarSight. After training the model, the best.pt file contains the final model for final Detection & Classification. results.txt file contains the summary of Accuracy Losses achieved at each epoch. The output obtained is shown in Figure 5.1.

Model Evaluation:
Using the YOLOv5 evaluation script the performance of CarSight was evaluated on the validation set. This gives an idea of the model's accuracy and any areas for improvement. Some outputs while testing the model are shown in Figure 3.3.

(a) Detection of car with 0.93 confidence score



(b) Detection of car with 0.92 confidence score



(c) Detection of car with 0.85 confidence score

(d) Detection of car and sidelight with 0.93 and 0.78 confidence score respectively

Figure 3.3: Bounding Box Prediction of Model Car

Figure 3.3 contains four images containing the car and sidelight prediction. The confidence scores of the car can be seen as 0.93, 0.92, 0.85, and 0.93 and the confidence score of the sidelight is 0.78. The confidence scores can be interpreted as the level of sureness of the object class detected by the model.

Model Deployment:

The performance of the trained model was deemed to be satisfactory, so a YOLOv5 inference script was used to deploy the model and perform object detection on live video streams. The overall process of training the model is shown in Figure 3.4.

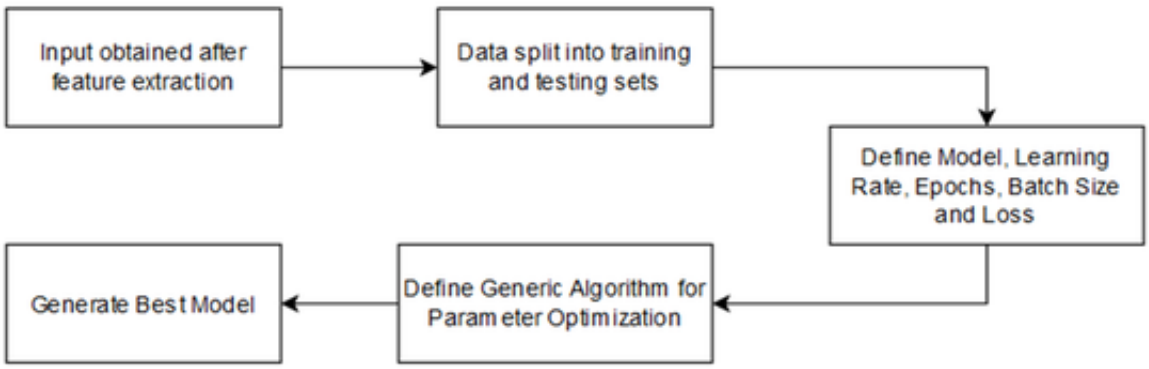


Figure 3.4: Model Training Process

3.3 Load CarSight For real-time object detection

Loading Carsight:
First of all, we need to load the trained YOLOv5 model i.e. CarSight directly from its pre-trained weights. This can be done using the Torch library. By using a loading script, CarSight was successfully loaded in our main program where various decision-making tasks were performed.

Reading input frames:
Now we need to read the input frames from a camera that we have installed at 6 different locations. This was done by using the OpenCV library.

Preprocess the input frames:
Preprocessing of the input image must be done before running the model. The data obtained from the cameras may not all be uniform. So, all the images received have to be resized to a consistent size. Video frames need to be pre-processed so that objects can be easily detected by the model. The opacity of the pictures may vary, due to differing surrounding lighting. Some images appear brighter, while others appear dark. So we preprocess the input frames by resizing, normalizing, and converting them to the required format by applying resize() function, converting them to grayscale and Gaussian blur followed by Binary thresholding.

Run the model:
The model was then run on each preprocessed image frame to get the predictions for bounding boxes, class identification, and confidence scores for each object in the frame.

Post-Process the Predictions:
Post-processing steps such as non-maximum suppression were performed to remove overlapping bounding boxes and keep only the most confident ones. we put a confidence level of 40% and overlapping 30%. After getting the coordinates of the bounding boxes, the bounding box of the object was plotted.

The tricky part then was to find the intersection of the bounding box and track. To do this, the input image was converted to HSV Color space because of the ease of segmenting color in the HSV Color space.

The lower and upper threshold values of the yellow color track were noted as:
lower = np.array([0,179,171])
upper = np.array([255,255,255])

The binary mask was created within the specified range by inRange() function. The inRange() function in Computer Vision is a method for image segmentation. This function separates objects of interest from the background of an image. Here, we separate the different shapes of the track from the image and create a mask of them. The inRange() function takes the input image and threshold values, which defines a range of values that correspond to the different objects of interest (i.e. track). This is how the mask of the track can be created. Now to find the intersection between the track and the bounding box, the mask of the bounding box is needed. For that, a blank image with pixel intensity 0 is created with np.zeros_like() function in Python. np.zeros_like() function is a NumPy function that returns an array of zeros with the same shape and data type as the input array. Now, the bounding box is plotted on that binary image and also fill the bounding box with pixel intensity 1.

The intersection was obtained by using the bitwise_and() function. This is a function in the openCV library that performs a bitwise AND operation between two images (i.e. track_mask and bbox_mask). The function computes the bitwise AND between each pixel in 'track_mask' and the corresponding pixel in 'bbox_mask'. Now the number of non-zero pixels in the resulting image was counted by the 'countNonZero()' function. This function gives the number of white pixels in the binary image. If the number of white pixels is greater than 100 then the alarm rings to declare the fail condition.Since only two object classes have been defined while training our model (i.e. car and sidelight).

In the 8-test, the alarm is generated when the car touches the 8-shaped boundary line. This is done by finding the intersection between the binary mask of the boundary box of car and the binary mask of the track. The system also monitors the side lights in the frames obtained from camera-1. The alarm is also generated when the sidelight is detected in the 8-test and the candidate is declared as fail.

In the U-turn test, the video frames from camera-3 are continuously monitored to detect the line touch and sidelight blink. If the car touches the boundary line or sidelight is not detected then the candidate is considered a fail.

In the traffic-light violation test, the pixel intensity of lights is monitored continuously. If the red pixel is 200 then the red light is considered as ON. During this time if the candidate tries to cross Zebra Crossing then the alarm rings and he/she is declared as fail. Here, an imaginary line is drawn on frames just ahead of zebra-crossing. when the car tries to cross the zebra crossing in red light then the mid-point of the bounding box intersects with the line and hence generates the alarm.

In the rest of the tests like the hump test and parking test, there is only one fail condition if the car touches the boundary line. So, the video frames obtained from camera-2 and camera-6 are monitored to check only this condition only.

One condition that universally results in failure is if the car turns off during any of the track segments. To detect this, the system counts the number of still frames, and if the counter reaches 150, the car is considered to be in a stopped position and the candidate fails. In other words, if the car remains stationary for more than 150 frames, it is deemed a failed condition.

Results visualization:
Finally, the results have been visualized by drawing the bounding boxes and class labels on the input frames and displaying the desired output frames in a window. The result has also been saved through an output video file. The video file is written into memory only if the car is present in the frame.

4. System design

4.1 Trial System Flow design

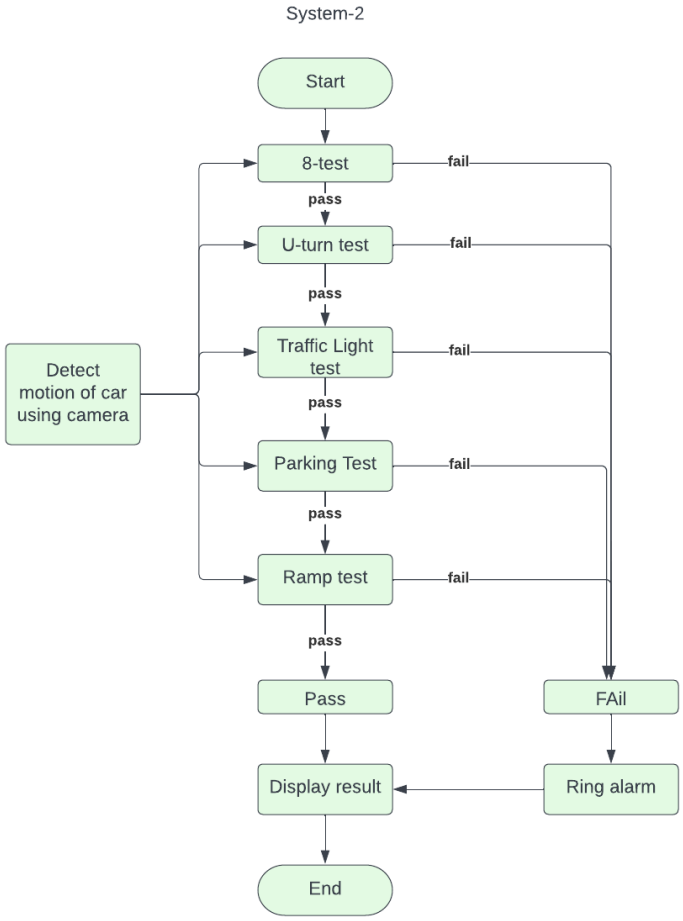


Figure 4.1: Trial System Flow design

Object detection starts as soon as the candidate enters the track with the car. There are six different camera modules that handle the detection of different stages. The detection starts serially from the 8-test and continues till the end of the test, in case of success as shown in Figure 4.1. If the driver makes a mistake during any of these stages, an alarm is generated and upon enough mistakes, the examinee is deemed FAIL. The result is then displayed at the end of the trial examination.

4.2 Model Track Design

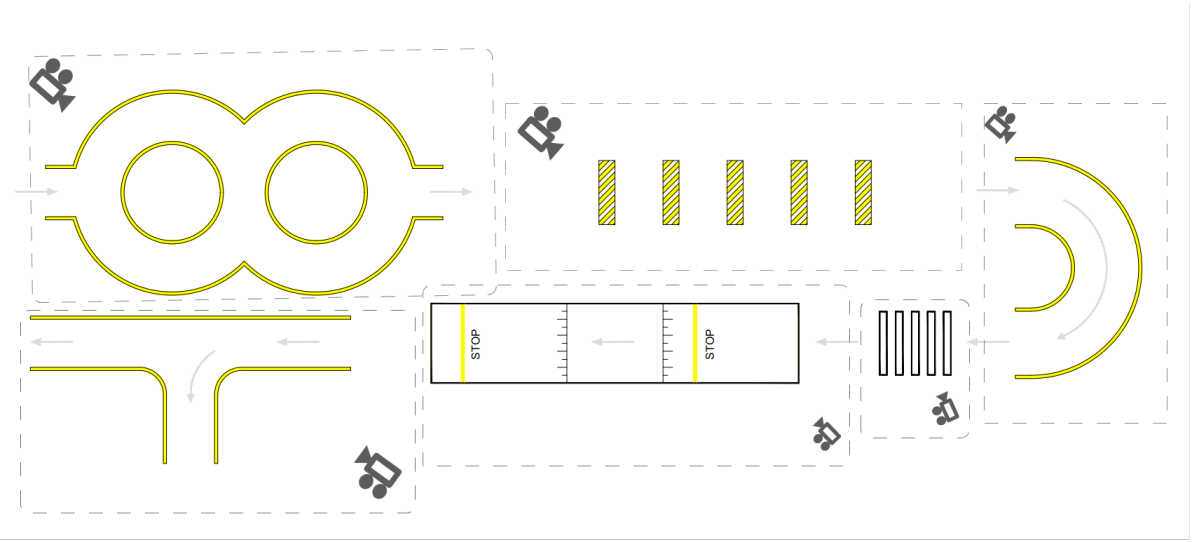


Figure 4.2: Model design of trial track (scale 1:13)

Figure 4.2 is a model design of the trial center designed using AutoCAD. The overall track is divided into six segments including the 8-test, speed breakers, U-turn, traffic lights and zebra crossing, ramp, and garage parking. The individual segments of the track have a camera that records the vehicles in each segment. The 8-test is recorded by camera-1, similarly, the hump test, U-turn test, traffic light violation test, ramp test, and garage parking tests are recorded by camera-2, camera-3, camera-4, camera-5, and camera-6 respectively. These sections are to the scale of the remote-controlled car. All these six sections were then printed on a separate flex board, where the data collection and demonstration took place.

The scaling of the prototype is done with respect to the dimension of remote controlled car. The actual width of the Maruti Alto which is provided as the trial vehicle is 1515mm. The width of the remote-controlled car that was used in the prototyping was 115mm. The defined ratio of the width of the trial car to the width of the track is 1:2. Hence the ratio of the width of the remote-controlled car to the width of the track in the prototype was also 1:2. So the ratio of the prototype track to the actual track was scaled to 1:13 to meet all scalings and match the necessary dimensions.

4.3 Use Case Diagram

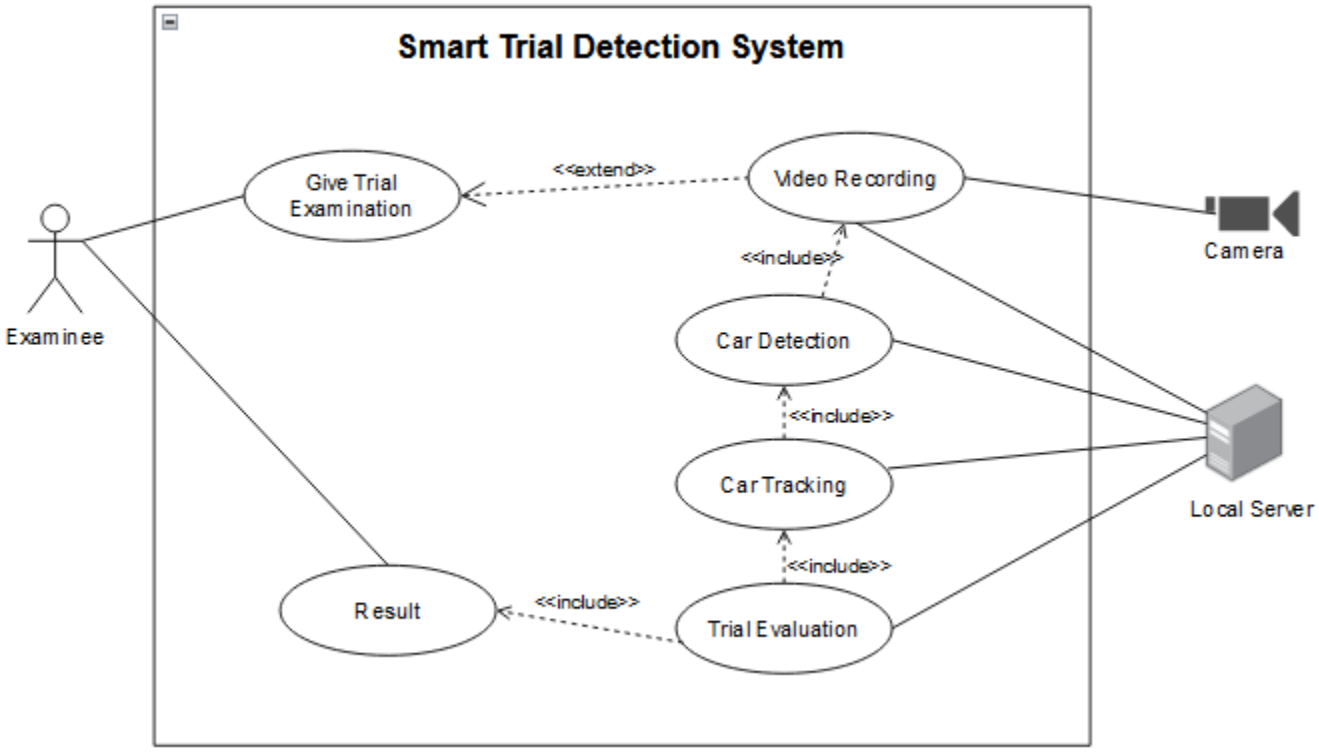


Figure 4.3: Use Case Diagram of Smart Trial System

Figure 4.3 is the use case diagram of the smart trial system that describes the relationships between the entities involved in the system. The examinee gives the trial examination at the trial center, which is recorded live by as many as six camera modules. This is done for various sections of the driving test, including 8-test, u-turn test, zebra crossing and parking test. The recordings are saved to a local server. The recordings are passed through the real-time detection model that simultaneously performs car and sidelight detection and tracking and checks the fail conditions to verify the passing of the candidate's trial. The results of the examinee are stored in the local server for future use.

4.4 Activity Diagram

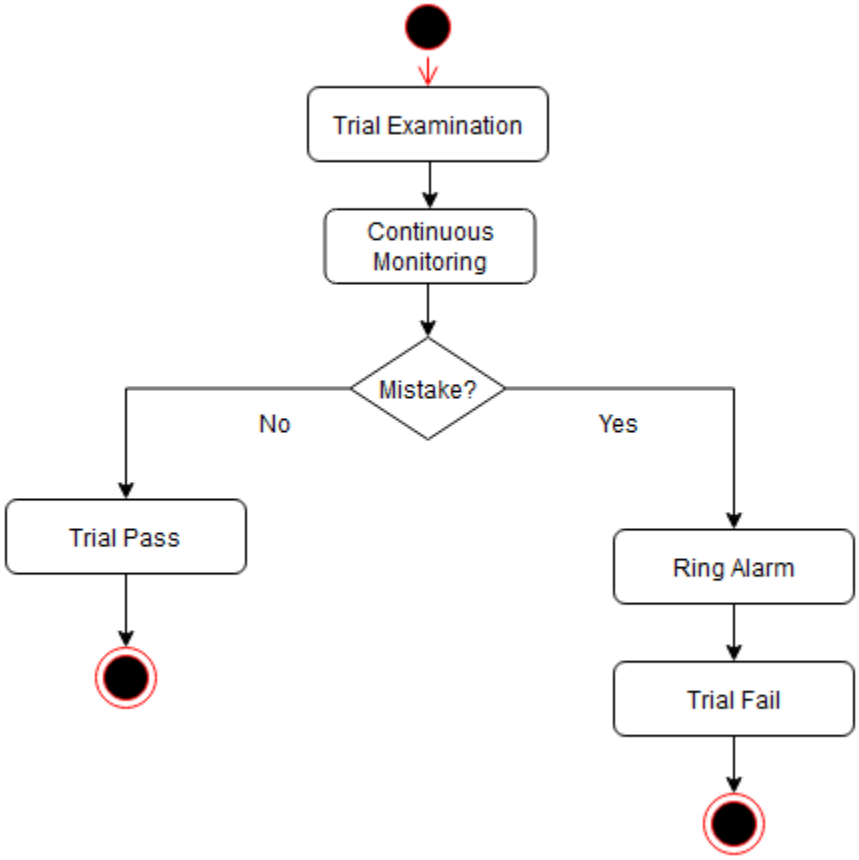


Figure 4.4: Activity Diagram of Smart Trial System

Figure 4.4 is the activity diagram that depicts the flow of actions carried out in the trial detection system. The system consists of a camera module placed in different locations that continuously monitor and evaluate the performance of the driver throughout different phases of the trial examination, including the 8-test, U-turn test, ramp test, and parking test. If no mistake is caused by the examinee, the trial is considered as a pass, whereas an alarm is sounded in the case of a mistake to denote trial failure.

4.5 Data Flow Diagram

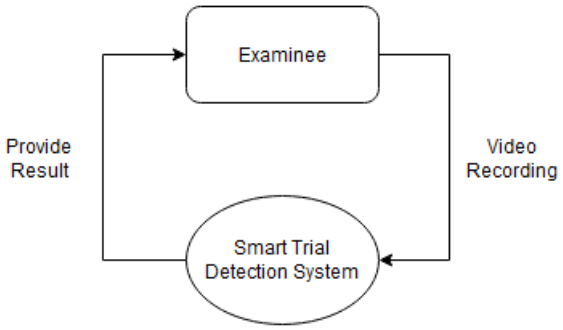


Figure 4.5: 0-level DFD

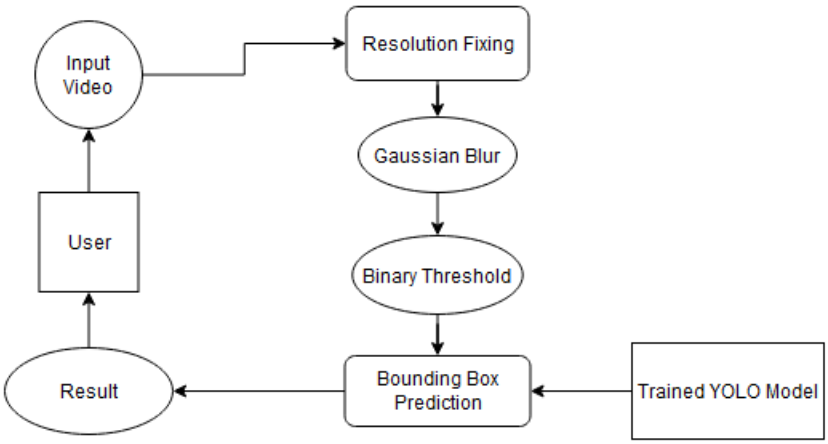


Figure 4.6: 1-level DFD

Figure 4.5 and Figure 4.6 are the data flow diagrams that depict the flow of data in the trial detection system. The 0-level DFD gives an external perspective of the flow of data from the user to the trial detection system. Data from the examinee is collected through the mounted cameras and then transported to the modeling system. 1-level DFD shows the flow of data inside the system through various processes like resolution fixing, Gaussian blur, and bounding box prediction. Resolution fixing is done to maintain uniformity in all the collected images; Gaussian blur helps to soften the edges and Binary Threshold helps to remove noise. The preprocessed images are passed to the trained YOLO model to predict bounding boxes and track the result of the trial examination.

5. Results & Discussion

The overall mAP(mean Average Precision) of the model CarSight was 95%. Figure 5.1 contains the precision, recall, and mAP of two classes. Here, class 0 represents the sidelight, and class 1 & car both represent the car. The precision, Recall, and mean Average Precision of individual classes are also shown in Figure 5.2.

```
Model summary: 157 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95:
                   all         35         51      0.933      0.938       0.95      0.677
                     0         35         24      0.981          1      0.995      0.774
                     1         35         16      0.858      0.812      0.859      0.314
                   car         35         11      0.959          1      0.995      0.944
```

Figure 5.1: Model output

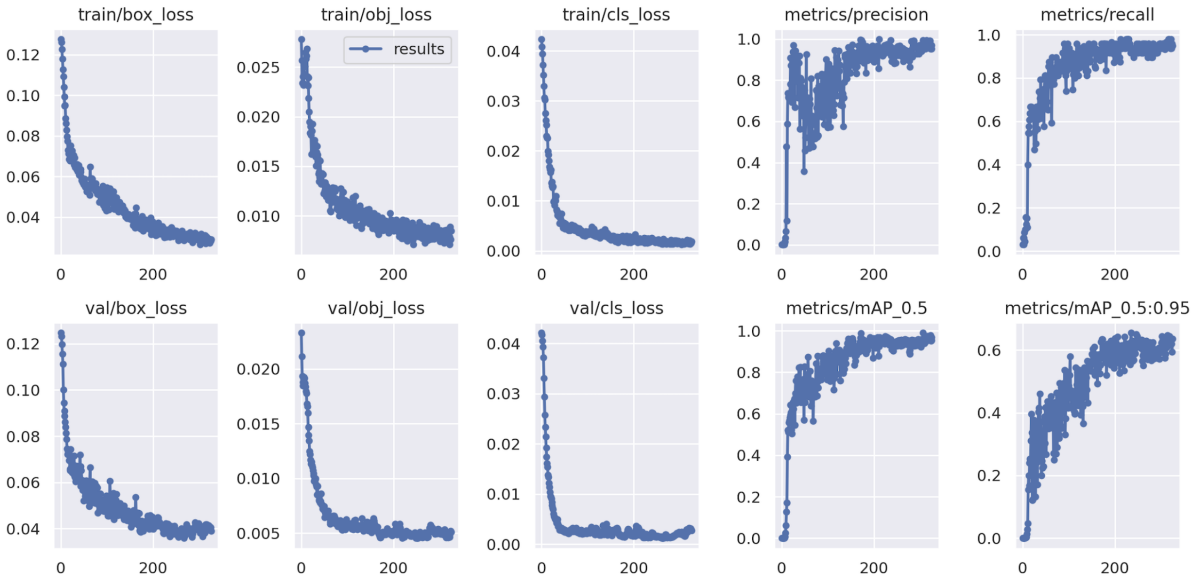


Figure 5.2: Training Results Output

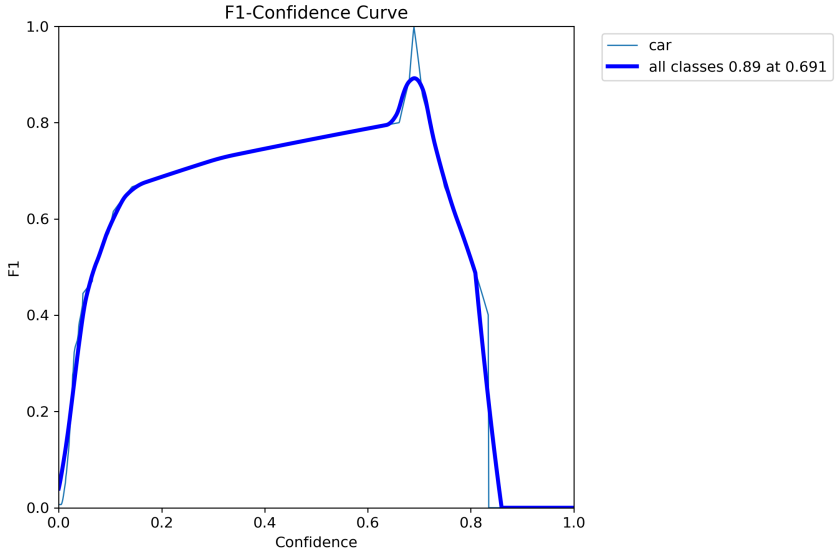


Figure 5.3: F1 Confidence Curve

The F1 confidence curve shown in Figure 5.3 represents the relationship between the confidence level of CarSight predictions and the corresponding F1 score. The F1 score is a metric used to evaluate the performance of binary classification models, which takes into account both precision and recall. Precision measures how accurate the positive predictions are, while recall measures how many of the actual positives the model correctly identified. The F1 score is the harmonic mean of precision and recall, ranging from 0 to 1, with a higher score indicating better performance.

The curve starts at 0 F1 score with the confidence level 0, indicating that the model's predictions are random guesses. The F1 score increases with the confidence value, reaching 0.6 at a certain point, indicating that the model is becoming more accurate in its predictions as the confidence level increases. At a confidence level of 0.691, there is a sharp rise in the F1 score, highlighting a significant improvement in model performance. The F1 score reaches a perfect score of 1 for the class 'car' and 0.89 for others, suggesting that the model is making accurate predictions with high confidence. However, as the confidence level increases further beyond 0.89, the Fl score drops sharply to 0. This indicates that the model's predictions become very confident but incorrect. This could be due to the model overfitting to the training data or making incorrect assumptions about the underlying distribution of the data.

Overall, the F1 confidence curve of CarSight suggests an improvement in the model's performance as the confidence level increases up to a certain point, beyond which

the performance of the model deteriorates. It is important to identify the optimal confidence threshold to balance accuracy and confidence in the model's predictions, especially when the model makes predictions for all classes simultaneously.

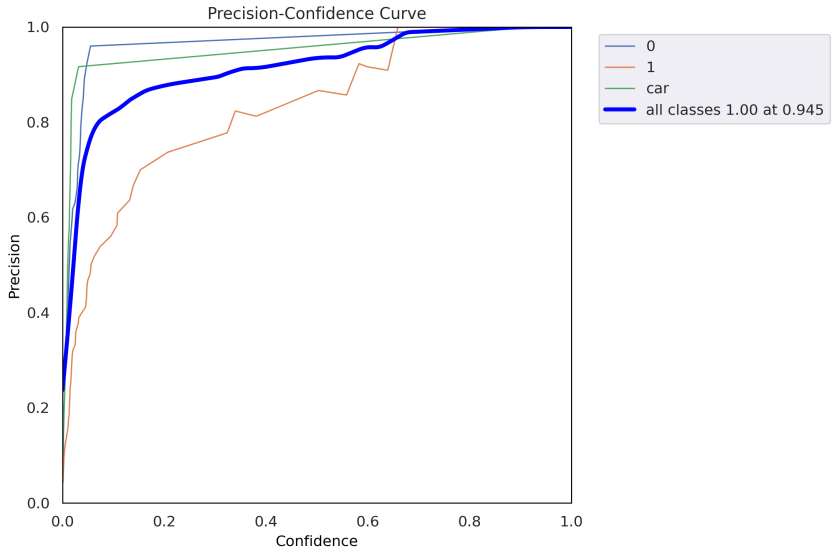


Figure 5.4: Precision Confidence Curve

The precision-confidence curve shown in Figure 5.4 represents the relationship between the confidence level of CarSight predictions and the corresponding precision score.

The curve starts at a precision of 0 at confidence level is 0, indicating that the model's predictions are random guesses. As the confidence level increases, the precision score also increases. At a confidence level of 0.1, the precision of all classes increases to 0.8, suggesting that the model is becoming more accurate in its predictions as the confidence level increases.As the confidence level increases further to 0.7, the precision tends to 1 and converges to 1 as the confidence level increases to 1. This indicates that the model's predictions become very confident and highly accurate. This suggests that the model is performing well and making highly accurate predictions with high confidence. All classes are labeled as 1 at a confidence level of 0.945, suggesting that the model is very confident in its predictions for all classes.

Overall, the precision-confidence curve of CarSight suggests that the model's performance improves as the confidence level increases up to a certain point, beyond which the performance of the model stabilizes. It is important to identify the optimal confidence threshold to balance precision and confidence in the mode's predictions. In this

case, the curve reaches its peak at a confidence level of 1, indicating that the model is making highly accurate predictions with high confidence.

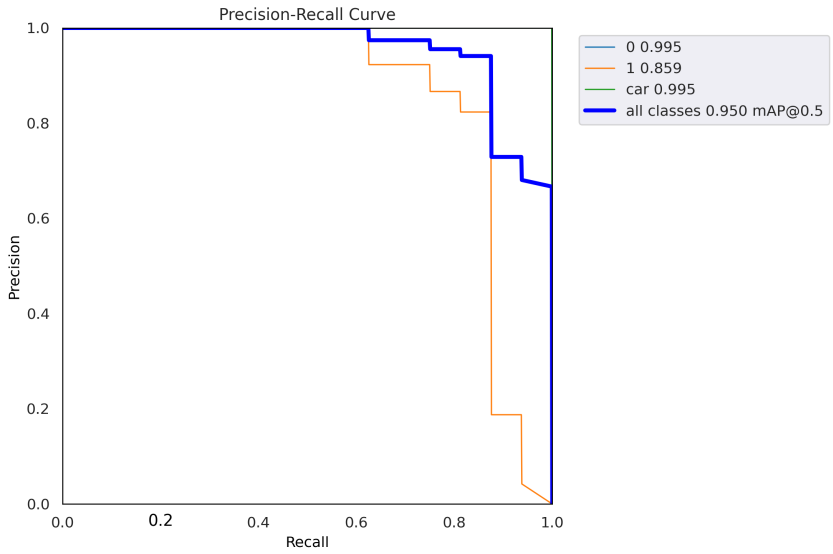


Figure 5.5: Precision-Recall Curve

The precision-recall curve shown in Figure 5.5 represents the relationship between the precision and recall of CarSight predictions.

The curve starts at a recall of 0 and a precision of 1, indicating that the model is correctly identifying all the positive samples and not making any false positive predictions. As the recall increases up to 0.6, the precision remains at 1, suggesting that the model is continuing to accurately identify positive samples without any false positives

However, beyond a recall of 0.6, the precision starts to decrease. This indicates that the model is starting to make false positive predictions, which are negatively affecting the precision. The precision drops to zero when the recall reaches 1, suggesting that the model is incorrectly identifying all positive samples as negatives. The class car is labeled as 0.995, which suggests that the model has a high level of confidence in its predictions for this class. The class sidelight is labeled as 0.859, which indicates that the model is slightly less confident in its predictions for this class. Overall, the model has an average precision of 0.95 mAP@0.5, indicating that the model is generally accurate in its predictions.

In summary, the precision-recall curve of CarSight suggests that the model has a high

level of precision when recall is low, but the precision drops as recall increases, indicating that the model is making more false positive predictions. It is important to balance precision and recall to identify the optimal threshold for the model's predictions, especially when dealing with imbalanced datasets where certain classes may have higher confidence levels than others.

The output obtained after the implementation of CarSight in 8-test, hump, U-turn, traffic light, and parking are discussed here. Figure 5.6 shows the actual implementation of CarSight in the Smart Trial System. It shows how CarSight is implemented for the decision-making process 8-test. The figure contains four windows arranged in a row named result_img, mask,bounding_box_mask, and tracking. The window result_img shows the output of Color masking of the track. The window mask shows the output of the binary mask of track 8. The third window bounding_box_mask shows the binary mask of the bounding box of the car and the fourth window named as tracking shows the actual output of the Smart Trial System showing the original track and bounding box of a car. During the 8-test, the alarm was generated when the car touches the line by finding the intersection between the windows Bounding_box_mask and mask. Also when the class 0 (i.e. sidelight) was detected during the 8-test, the alarm was generated by the system.

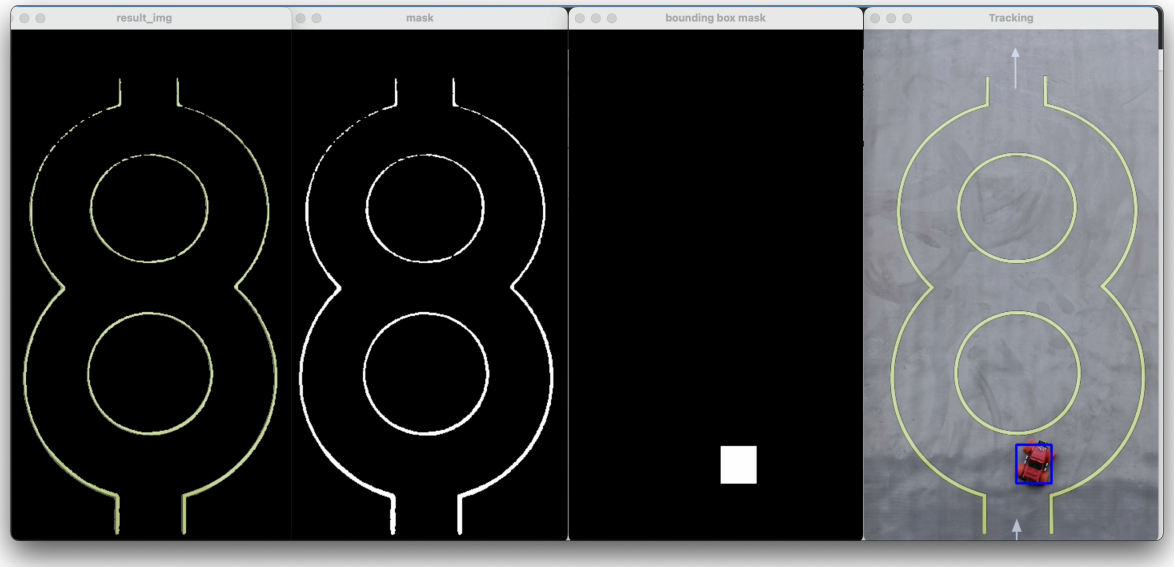


Figure 5.6: Combined image of four windows during 8-test.

Figure 5.6 shows the actual output of the model CarSight. It shows the bounding box

prediction of the car on track during the 8-test. This image frame is the output of Camera-1.

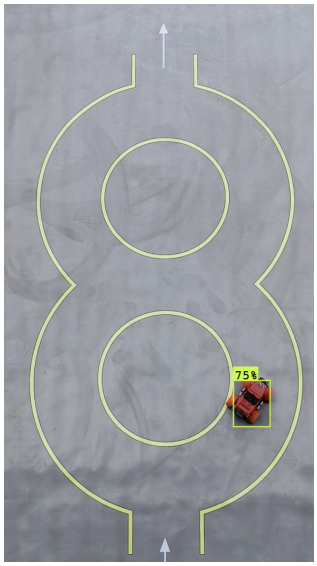


Figure 5.7: Bounding Box prediction of the vehicle in 8-test

Figure 5.7 shows the output of the Speed Breaker test also called the hump test. This image is the output of camera-2 which shows a car on a hump track with its bounding box prediction. This section detects if the car travels in a straight line or not and also the side light. If the car goes out of track or detects any side light blinking then the candidate is declared as fail and the alarm rings.

Figure 5.8: Vehicle Tracking in Speed Hump test

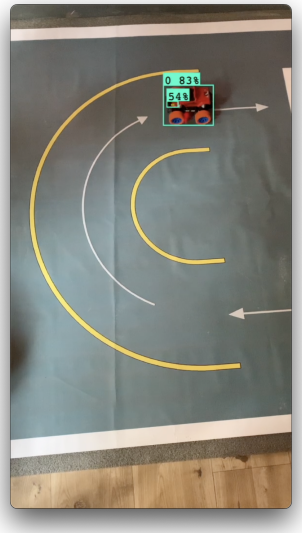


Figure 5.9: Vehicle Tracking in U-Turn test

Figure 5.9 shows the output of the U-turn test which is taken from camera-3. In this section checks for both line crossing and side lights. If the line is crossed or a side light is detected in this video frame then the candidate is declared as fail and the alarm rings.

Figure 5.10: Vehicle Tracking in the Zebra Crossing section

Figure 5.10 shows the output of the traffic light violation test. This frame is taken from camera-4. Here if the car crosses the zebra-crossing in the red light then the candidate is considered as failure and the alarm rings.

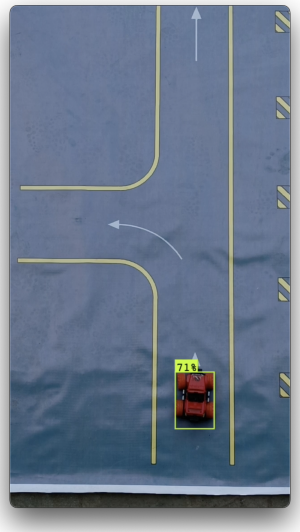


Figure 5.11: Vehicle Tracking in the Garage Parking test

Figure 5.11 shows the output of garage parking, taken from camera-5. The image shows a car on a parking track with bounding box prediction. The candidate is tracked along their path, and an alarm is sounded to denote failure when the vehicle touches the yellow boundary line.

6. Conclusion

This project mainly focuses on making CarSight. CarSight is a custom object detection model built using YOLOv5s. CarSight is a model capable of detecting cars in trial centers and capable of analyzing visual data from cameras. We converted the current driving license system into Smart Trial System by implementing CarSight. While doing the project, we had a lot of challenges in collecting the dataset and training the model. We divided the track into six segments and we printed separate flex to test the tracking model. A custom dataset was created by taking images of a Remote Controlled car on the track. After collecting the data, the model was trained in Google Colaboratory using YOLOv5, until we got satisfactory results. Upon getting the mAP score of 95%, the model was deployed for real-time object detection and decision-making part. In the decision-making part, different Image Processing techniques have been used for masking the track and bounding box. The decision was made by calculating the number of intersecting pixels of both masks. similarly, by checking the object class present in each frame, we made the further decision disqualifying the candidate.

Finally, a prototype of the Smart Trial System was developed, and after the model testing, desirable accuracy was obtained.

7. Limitations and Future Enhancement

Since this is the first version of our prototype, it has some limitations. If the car gets turned off, the FAIL decision has to be made by calculating the number of still frames, i.e. if the previous frame and current frame are the same for 150 counts (i.e. 5 sec if the video is 30fps). But sometimes the system may fail or predict a false positive case if the car is not turned off and is stood in the same position for 5 seconds. The system also fails when the car turns off and on within 4 sec because our system has to count up to 150 still frames to determine the stopping and make the candidate fail. Also in the ramp section, the system fails to detect the cases when the car slides back, due to the limitation of the printed flex. This smart trial system is also hard to implement for two-wheelers, as it requires a lot of camera angles to determine if the rider has touched the ground with his feet. Thus, these are the limitations of our project.

By solving these limitations, this project can be implemented in real-life trial scenarios. The project could be enhanced by implementing a similar detection model for two-wheelers. Different sensors could also be implemented in this project to solve the limitation of the vehicle stopping. This project can easily be scaled as the proposed system is fully economical and a one-time investment.

8. References

[1] Department of Transport Management: https://www.dotm.gov.np/MainData/ ProcedureCodeofConduct (cited date: June 18, 2022)

[2] Poudel N., Kathmandu Post: https://kathmandupost.com/miscellaneous/2019/07/04/police-arrest-four-persons-of-a-fake-driving-licence-racket (cited date: November 15, 2022)

[3] People's Review: https://www.peoplesreview.com.np/2022/03/18/ciaa-files-case-against-9-for-altering-driving-license-results/ (cited date: November 15, 2022)

[4] Prahith N, Shridhar, K., Dey, P., Aishwarya S V, IoT based Automatic Driving License Test, International Journal of Engineering Research Technology (IJERT) Vol. 9 Issue 04, April 2020

[5] Ms. Suvarna A Dodke, Automation of Driving License test using Wireless Sensor network, International Research Journal of Engineering and Technology IRJET Volume: 02 Issue: 08 Nov 2015.

[6] Margale KA, Pawale, P., Patil, A., Waykule, J., Driving License Test Automation Using VB. International Journal of Engineering and Applied Sciences (IJEAS) Volume: 2 Issue: 4 April 2015.

[7] Han, H., Zhang, Y., Guo, Y. (2020). Driving behavior recognition based on improved YOLOv3 and LSTM network. IEEE Access, 8, 195874-195883.

[8] Fang, Y., Chen, Y., Chen, X. (2019). An Identity Authentication and Anti-Cheating System for Driving Test. In International Conference on Computer Science, Engineering and Applications (pp. 453-460). Springer, Singapore.

[9] Zhang, X., Wu, Q., Xie, X., Huang, X., Zhang, L. (2021). A Driving License Test Monitoring System Based on Computer Vision and Machine Learning. IEEE Access, 9, 50600-50609.

[10] Wang, H., Li, Q., Song, Y., Liu, Q., Zheng, Y. (2018). An anti-cheating system for driving license test based on computer vision. In 2018 IEEE International Conference on Mechatronics and Automation (ICMA) (pp. 1323-1328). IEEE.

[11] Mamun, O., Akhtaruzzaman M., and Anowarul Abedin, M. (2018). Convolutional Neural Network-Based Road Detection and Extraction.Journal of Imaging.

[12] Zhang, F., Wu, X., Zha, H., and Yang, M.. (2016). Robust Lane Detection and Tracking in Challenging Scenarios. IEEE International Conference on Robotics and Automation (ICRA).

[13] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016).You Only Look Once: Unified, Real-Time Object Detection published in University of Washington