A PROJECT REPORT ON

“AUTONOMOUS VEHICLE NAVIGATION SYSTEM USING DEEP LEARNING AND COMPUTER VISION WITH REAL-TIME OBJECT DETECTION”

Submitted in partial fulfilment of requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

CSE (ARTIFICIAL INTELLIGENCE)

Submitted

by

DUDEKULA RAJESH 21AM1A3193

Under the Guidance

Of

Mr. M. Ramachandrudu, M.Tech

Assistant professor

Department of CSE(AI)

DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE)

SVR ENGINEERING COLLEGE

(AUTONOMOUS)

AYYALURUMETTA, NANDYAL, NANDYAL(DIST.)– 518501

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuram)

ECE & CSE Dept are accredited by NBA.

ACADEMIC YEAR: 2024 - 2025

SVR ENGINEERING COLLEGE
(AUTONOMOUS)

AYYALURU METTA, NANDYAL, NANDYAL(DIST.) – 518501

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuram)

ECE & CSE Dept are accredited by NBA.

DEPARTMENT OF CSE (ARTIFICAL INTELLIGENCE)

CERTIFICATE

This is certified that the technical project entitled, "Autonomous Vehicle Navigation System Using Deep Learning and Computer Vision with Real-time Object Detection" is being submitted by **DUDEKULA RAJESH (21AM1A3193),** in partial fulfilment of the requirements for the award of Degree of **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING (ARTIFICIAL INTELLIGENCE) to SVR ENGINEERING COLLEGE** is a record of Bonafide work carried out by them during the academic year 2021- 2025 under our guidance and supervision.

The results presented in this technical project have been verified and found to be satisfactory.

SIGNATURE OF THE GUIDE	HEAD OF THE DEPARTMENT
Mr. M. RAMACHANDRUDU, M.Tech,	**Dr. M.SUBBA REDDY, M.Tech,Ph.D.,**
Assistant Professor,	**Associate Professor,**
Department of CSE-AI,	**Department of CSE-AI,**
SVREC, Nandyal.	**SVREC, Nandyal.**

External Examiner

Student's Declaration

I **DUDEKULA RAJESH (21AM1A3193),** a student of the **CSE (ARTIFICIAL INTELLIGENCE), SVR ENGINEERING COLLEGE(Autonomous)** do hereby declare that I completed the PROJECT from January 2025 to April 2025 under the Guidence of **Mr.M.RAMACHANDRUDU**, Department of **CSE (ARTIFICIAL INTELLIGENCE), SVR ENGINEERING COLLEGE(Autonomous).**

(Signature and Date)

Endorsements

Faculty Guide

Head of the Department

Principal

ACKNOWLEDGEMENT

I earnestly take the responsibility to acknowledge the following distinguished personalities who graciously allowed us to carry out project work successfully.

I express deep gratitude to our guide **Mr. M. RAMACHANDRUDU, M.Tech** Assistant Professor, Department of CSE(AI), **SVR Engineering College**, for the guidance and incessant help and encouragement throughout the course of the project work. Her friendly and informal talks helped us to work under excellent working conditions.

I would like to express our gratitude to project coordinator **Mrs. V. KARUNA, M.Tech,(Ph.D)** Assistant Professor in the Department of CSE(AI), **SVR Engineering College,** for their encouragement throughout the course.

I thankful to the Head of the Department of CSE(AI), **Dr. M. SUBBA REDDY, M.Tech,Ph.D.,** Associate Professor **SVR Engineering College**, for the encouragement and assistance provided to us, which contributed to the successful completion of this project.

I wish to convey my gratitude and express sincere thanks to all **P.R.C** (Project Review Committee) members for their support and Co-operation rendered for successful submission

I thankful to our Principal **Dr. P. MALLIKARJUNA REDDY** who has encouraged and motivated us to complete the project by providing all the necessary facilities to carry out the work in the college.

I thankful to our Honourable Chairman **Sri S.V. RAMI REDDY** & Honourable Managing Director **Sri S. DINESH REDDY** for providing good faculty and for their moral support throughout the course.

I would like to thank all teaching and non-teaching members of the CSE(AI) Department for their generous help in various ways for the completion of this thesis. They have been great sources of inspiration to us and we thank them from the bottom of my heart.

BY

DUDEKULA RAJESH (21AM1A3193)

College Vision

To produce Competent Engineers with "A strong basics of Engineering Knowledge, A Mindset of Lifelong Learning and the necessary Complementary Skills needed to be Successful Professionals".

College Mission

- To Impart Affordable and Quality Education to meet the needs of Society.
- To provide Ambient Teaching - Learning Environment, Adequate Infrastructure and Resources.
- To Empower the students through Complementary Skills such as Decision-making, Interpersonal, Ability to apply knowledge across disciplines, Ability to work effortlessly with others.

Core Values

- Commitment to Continual Improvement
- Inclusiveness
- Enhancement of Student Learning through Innovative Instructional Methods
- Collaboration with Industry and other Institutions for Mutual Benefits
- Pursuit of Excellence in all Activities

SVR ENGINEERING COLLEGE
(AUTONOMOUS)
Approved by AICTE: New Delhi, 2(F) & 12(B) Recognition by UGC Act.1956
Affiliated to JNTUA, Ananthapuramu
Accredited by NAAC & Accredited by NBA, ISO: 9001: 2015 Certified Institution
Ayyalur Metta, NANDYAL Dist. - 518 502, A.P.
ESTD : 2007

Department Vision

- To build an ecosystem to contribute to society by producing leaders in Artificial Intelligence through quality education.

Department Mission

1. Train the students in the state-of-the-art technologies of AI.
2. Sensitize students to solve societal issues through AI techniques by inculcating values and ethics.
3. Enhance employability and entrepreneurial skills in the field of AI through experiential and self-directed learning.

Program Educational Objectives (PEO’s)

PEO1: Graduates will be capable of adapting to new technologies and systems, making contributions, and innovating in the key domains of artificial intelligence.

PEO2: Graduates will be ethically and socially responsible solution providers and entrepreneurs in the field of Computer Science and Engineering with AI Specialization.

PEO3: Graduates will be able to successfully pursue higher education in AI Specialization.

Program specific outcomes (PSO's)

PSO1: Build skills to develop software applications in specialized areas of Artificial Intelligence and Machine Learning.

PSO2: Apply the various AI and ML techniques for industrial applications in the areas of Deep Learning, Cloud Computing, Natural Language Processing, and emerging areas.

Program Outcomes (PO's)

PO Number	Graduate Attributes	PO Statements
1	**Engineering Knowledge**	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex Engineering problems.
2	**Problem Analysis**	Identify, formulate, review research literature, and analyze Complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3	**Design/Development of Solutions**	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4	**Conduct Investigations of Complex Problems**	Ability to review research literature, use research methods to execute project and synthesize the problem to provide valid conclusions.
5	**Modern Tool Usage**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6	**The Engineer and Society**	Apply reasoning informed by the contextual Knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7	**Environment and Sustainability**	Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8	**Ethics**	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9	**Individual and teamwork**	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10	**Communication**	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11	**Project Management and Finance**	Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12	**Life-long Learning**	Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

TITLE	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO 11	PO 12	PSO 1	PSO 2
Autonomous vehicle navigation system using deep learning and computer vision with real-time object detection	3	3	2	3	2	3	1	3	2	1	2	3	1	3

List of Abbreviations

Abbreviation	Full Form
AV	Autonomous Vehicle
YOLO	You Only Look Once
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
R-CNN	Region-based Convolutional Neural Network
Fast R-CNN	Fast Region-based Convolutional Neural Network
Faster R-CNN	Faster Region-based Convolutional Neural Network
SLAM	Simultaneous Localization and Mapping
mAP	mean Average Precision
FPS	Frames Per Second
RGB	Red Green Blue (color model)
ROI	Region of Interest
SSD	Single Shot Detector
TPU	Tensor Processing Unit
GPU	Graphics Processing Unit
OS	Operating System
API	Application Programming Interface
HDF5	Hierarchical Data Format version 5
Pandas	Python Data Analysis Library
NumPy	Numerical Python
Matplotlib	MATLAB Plotting Library
Scikit-learn (sklearn)	Machine Learning Library for Python
Keras	High-level Neural Network API
TensorFlow	Open-source machine learning library

CONTENTS **PGNO**

ABSTRACT

The progress with Autonomous vehicles has been huge, mostly due to the milestones reached in artificial intelligence, more specifically deep learning. Recently, a lot of attention has been given to vision-based Autonomous vehicles, whereby cameras are installed extensively to understand and navigate their environment. Vision-based systems could perhaps get along without expensive sensors like lidar and radar and provide low-cost AVs for mass production.

It presents the progress made so far in vision-based Advanced Vehicle Systems, spanning key areas such as detecting vehicles, identification of pedestrians, recognition of traffic signs and lights, lane and road edge detection, and overall analysis of the traffic scene. Deep learning models provide these systems with the capability to correctly interpret the visual information in a very short time so that real-time decisions related to safe driving could be made.

Yet, some difficulties still have to be overcome, like reliable performance in various weather and lighting conditions and complex road situations. Another important line of research is the integration of vision-based systems with other sensors to achieve higher levels of autonomy. It will then provide an overview of existing technology, pointing out the existing challenges and suggesting future research directions to achieve a fully autonomous, safe, and cost-effective vehicle system.

CHAPTER – 1

1.INTRODUCTION

1. INTRODUCTION:

Autonomous driving is one of the most anticipated technologies of the 21st century and one of the most active research topics at the moment. Autonomous driving attempts navigating roadways without human intervention by sensing and reacting to the vehicles immediate environment. It includes major challenges for Computer Vision and Machine Learning. Object detection is one of the most important requirements for autonomous navigation and consists of localization and classification of objects.

Therefore, accurate object detection algorithms are needed. One challenge for example is the processing of numerous candidate object locations (often called “proposals”). These candidates provide only rough localization that must be refined to achieve precise localization. However, solutions to these problems often compromise speed, accuracy, or simplicity. Recent state-of-the-art deep learning models that address the problem of object detection include Region-Based Convolutional Neural Networks (R-CNN) and their improved versions Fast R-CNN and Faster R-CNN, designed for model performance and first introduced in 2013. A second model for object detection introduced in 2015 is YOLO, designed for speed and real-time use.

1.1 objective of the project

Datasets drive vision progress, yet existing driving datasets are limited in terms of visual content, scene variation, the richness of annotations, and the geographic distribution and supported tasks to study multitask learning for autonomous driving. In 2018 Yu et al. released BDD100K, the largest driving video dataset with 100K videos and 10 tasks to evaluate the progress of image recognition algorithms on autonomous driving.

The dataset possesses geographic, environmental, and weather diversity, which is useful for training models that are less likely to be surprised by new conditions. Provided are bounding box annotations of 13 categories for each of the reference frames of 100K videos and 2D bounding boxes annotated on 100.000 images for ”other vehicle”, ”pedestrian”, ”traffic light”, ”traffic sign”, ”truck”, ”train”, ”other person”, ”bus”, ”car”, ”rider”, ”motorcycle”, ”bicycle”, ”trailer”.

The goal of our project is to detect and classify traffic objects in a video in real-time using two approaches. We trained the two state-of-the-art models YOLO and Faster R-CNN on the Berkeley Deep Drive dataset to compare their performances and achieve a comparable mAP to the current state-of-the-art on BDD100K, which is 45.7 using a hybrid incremental net . We will focus on the context of autonomous driving and compare the models performances on a live video measuring FPS and mAP.

CHAPTER 2

2. LITERATURE REVIEW

The paper reviews tracking of multiple objects for autonomous vehicles navigation by using grouped data from the vehicle's cameras and LiDAR's to achieve full-surround coverage. It identifies how combining visual and LiDAR data reduces blind spots, enhancing object detection and tracking capabilities. A gap is noted in terms of limitations on system robustness under varying environmental conditions and sensor imperfections. Their approach addresses occlusion challenges and blind spots, enhancing situational awareness for safe autonomous navigation. Performance is validated through detailed experiments using metrics such as tracking accuracy, detection precision, and recall, highlighting the system's efficiency in complex real-world scenarios

This paper reviews CNN-based methods for fusing vision and LiDAR data to improve the object classification in autonomous vehicles. It identifies that the fusion of sensor data helps overcome individual sensor weaknesses, resulting in better classification accuracy. This complex scenario ensures robust object recognition using convolutional neural networks (CNNs). Performance of the proposed method is better than single-modality approaches, which justifies the prospect of increasing safety and reliability for the control of autonomous cars. The authors highlight a gap in handling real-time processing demands when integrating CNN-based fusion models

This paper examines guidance methods for autonomous ground vehicles in indoor settings, utilizing cameras, radio signals, and photoelectric sensors. A area to be challenged is noted in integrating multiple sensing techniques seamlessly for smooth navigation in highly dynamic indoor spaces. This describe the improvement in free-space self-driving automobiles navigating within buildings using computer vision, radio, and photoelectric sensing techniques.

This system makes use of all these sensors to precision navigate indoors where GPS is either very weak or not available at all. The integration of visual perceptions with radio and photoelectric sensors will enable system capabilities to detect obstacles, track movements and navigate at real-time basis.

It addresses the challenges of autonomous indoor navigation and provides a basis for developing autonomous vehicles in complex, GPS-denied environments-like those found in warehouses or indoor facilities.

This paper proposed a technique for an autonomously navigated vehicle within a fully map-less area. Here, it had real-time sensor data, such as the LiDAR and cameras, to utilize for localization and path planning in the less structured map-poor environments. The system can create a map, on-the-fly using simultaneous localization and mapping (SLAM), allowing navigation in rural areas where detailed, traditional maps are

not possible.

The experiment shows that the method is very efficient for navigating complex and unstructured terrains to serve all agriculture, forestry, and also rural transportation applications. This work reviews navigation strategies for autonomous vehicles in rural settings lacking pre-existing maps. The gap is in developing reliable navigation systems that work well in unstructured and unpredictable rural terrains.

This review the application of deep learning techniques to improve autonomous driving system safety, key challenges, and future research directions. Deep learning models are applied to tasks like perception, decision-making, and control and their potential to improve object detection, collision avoidance, and overall driving performance. This also poses further challenges in model interpretability, real-time processing and also in robustness in the complex and dynamic environment. In future research directions, they are demanding integration of multi-modal sensor data and also the formation of more reliable and explainable deep learning models for secure and efficient self driving. A gap is noted in achieving reliable safety outcomes in highly dynamic and unpredictable driving environments.

This paper introduces the IDD dataset for studying challenges in autonomous navigation under unconstrained conditions. Real-world driving scenarios and various sensors, such as cameras and LiDAR, have been used to capture the dynamic obstacles and complex road conditions with diverse lighting.

This paper presents how the dataset can be used to experiment on and design robust algorithms for activities like semantic segmentation, identifying objects, and analyzing scene in autonomous vehicles. From the paper, the authors try to develop different advancements in their navigation systems by having a diverse large-scale dataset to help them into developing algorithms that could address all the real- world complexities.

This propose an extensive survey on DNN for vehicle in autonomous driving systems. In this regard, this paper explores the available architectures of deep learning, such as CNN, YOLO, and Faster R-CNN, with regard to the application that could be made toward detecting and classifying objects into the automotive driving scene. Actually, in this paper the advantages of deep learning models are claimed: they can process large-scale datasets and complex features; conversely, real-time processing, robustness to various conditions, and labeled training data are among challenges. In this survey, advanced methods are summarized and the future focus is on improving detection accuracy and system reliability are pointed out, providing a great guide for future study and development

CHAPTER 3

3. METHODOLOGY

3.1 Existing System

Deep learning has gained a tremendous influence on how the world is adapting to Artificial Intelligence since past few years. Some of the popular object detection algorithms are Region-based Convolutional Neural Networks (RCNN), Faster- RCNN, Single Shot Detector (SSD) and You Only Look Once (YOLO).

3.1.1 Disadvantages

1. Limited Generalizability
2. High Computational Requirements

3.2 Proposed System

Real-Time Object Detection using YOLOv4 Objective: The objective of this system is to develop a real- time object detection system using the YOLOv4 algorithm, which can accurately detect objects in images and videos

3.2.1 Advantages

1. Enhanced Safety Features
2. Improved Accuracy and Reliability

3.3 Process model used with justification

To implement this project we have designed following modules

1. Sensor Modules

Camera Module: Captures visual data from the environment, such as traffic lights, pedestrians, and obstacles.

Lidar Module: Uses laser light to create high-resolution 3D maps of the environment. Radar Module: Detects speed and distance of surrounding objects.

GPS Module: Provides location and navigation data.

2. Computer Vision Modules

Object Detection Module: Uses deep learning algorithms like YOLO, SSD, or Faster R-CNN to detect objects in real-time.

Image Processing Module: Enhances and optimizes visualdata for object detection and navigation.

Scene Understanding Module: Interprets the environment and detects potential hazards.

3. Deep Learning Modules

Convolutional Neural Network (CNN) Module: Analyzes visual data and detects patterns. Recurrent Neural Network (RNN) Module: Processes sequential data and predicts future events. Transfer Learning Module: Leverages pre-trained models and fine-tunes them for specific tasks.

4. Navigation and Control Modules

Motion Planning Module: Generates optimal navigation paths based on sensor data and environment conditions.

Control Module: Regulates vehicle speed, steering, and acceleration to ensure safe and efficient navigation. Decision-Making Module: Makes tactical decisions based on sensor data, navigation plans, and environmental conditions.

1. Software and Hardware Modules

Operating System Module: Manages system resources and provides a platform for software development. Hardware Acceleration Module: Utilizes GPUs, TPUs, or other specialized hardware to accelerate deep learning computations.

Communication Module: Enables communication between vehicles, infrastructure, and the cloud.

2. Data Storage and Analytics Modules

Data Storage Module: Stores and manages large amounts of sensor data, navigation plans, and system logs. Data Analytics Module: Analyzes data to improve system performance, detect anomalies, and predict maintenance needs.

These modules can be integrated to create a comprehensive Autonomous Vehicle Navigation System that leverages deep learning, computer vision, and real-time object

3.4 Software requirement Specification

3.4.1 YOLO (You Only Look Once)

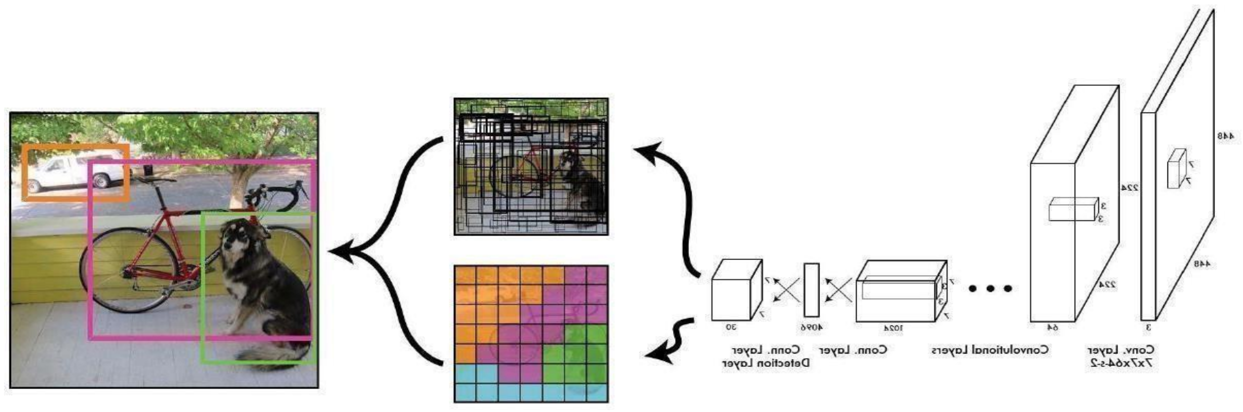


Figure 3.1: Pipeline of YOLO’s algorithm

You Only Look Once (YOLO) is a modern object detection algorithm developed and published in 2015 by Redmon et al. [8]. The name of the algorithm is motivated by the fact that the algorithm only looks once at the image and requires only one forward propagation pass through the neural network to make predictions unlike other state of the art object detection algorithms which work with region proposals and look at the image multiple times. YOLO uses a single end-to-end convolutional neural network which processes RGB images of size 448 x 448 and outputs the bounding box predictions for the given image .

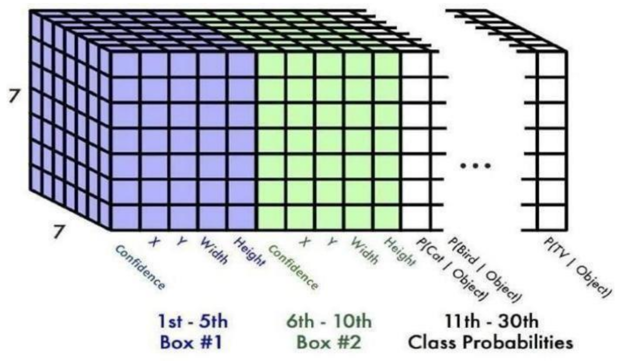
It basically reframes object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities [13]. The algorithm divides the input image into an S x S grid (in the paper S = 7). For each grid cell it predicts B bounding boxes (in the paper B = 2), where each bounding box consists of 4 coordinates and a confidence score for the prediction, and C class probabilities per grid cell taking the highest one as the final class. All of these predictions are encoded as an S x S x (B * 5 + C) tensor which is being outputted by the neural network (3.2).

What the algorithm finally does, is identifying objects in the image and mapping them to the grid cell containing the center of the object. This grid cell will be responsible for predicting the final bounding box of the object and will have the highest confidence score.

In the example (3.2) each cell of the 7 x 7 grid is represented by a vector of size 30 representing a particular area of the image. Each vector contains 2 bounding box predictions (5 values each) and 20 conditional class probabilities P(class|object). The first step upon extracting a valid prediction is to choose the bounding box with the higher confidence score and check

if the confidence score is above a predefined threshold (threshold = 0.25 in the paper) to output it as a valid prediction. This confidence score represents the prior in the conditional probability for the class prediction stating the probability that the given grid cell is the center of an object with a correct bounding box. To extract the class prediction YOLO outputs the conditional probability with the highest score.

YOLO spatially defines each bounding box by four coordinates (X, Y, Width, Height), where (X, Y) represent the center of the bounding box relative to the cell, while (Width, Height) represent the width and the height of the bounding box relative to the whole image. Because of this, a bounding box can be bigger than the cell where it was predicted. The cell is only used as the anchor point for the prediction. One disadvantage of this approach is the fact that every cell is able to predict only one object. If multiple objects are having their center points in the same cell, only one will be predicted



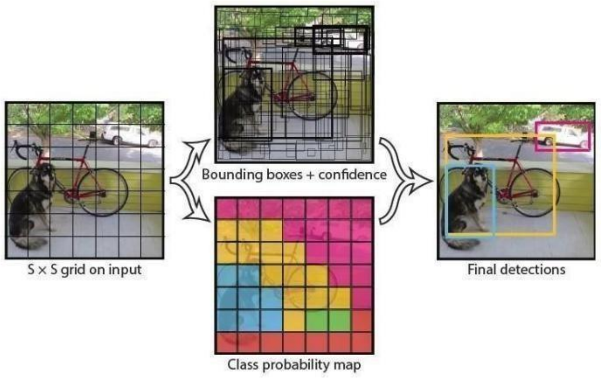


Figure 3.2: Output tensor of YOLO **Figure 3.3:** SxS grid and final detections

3.4.2 Model Architecture

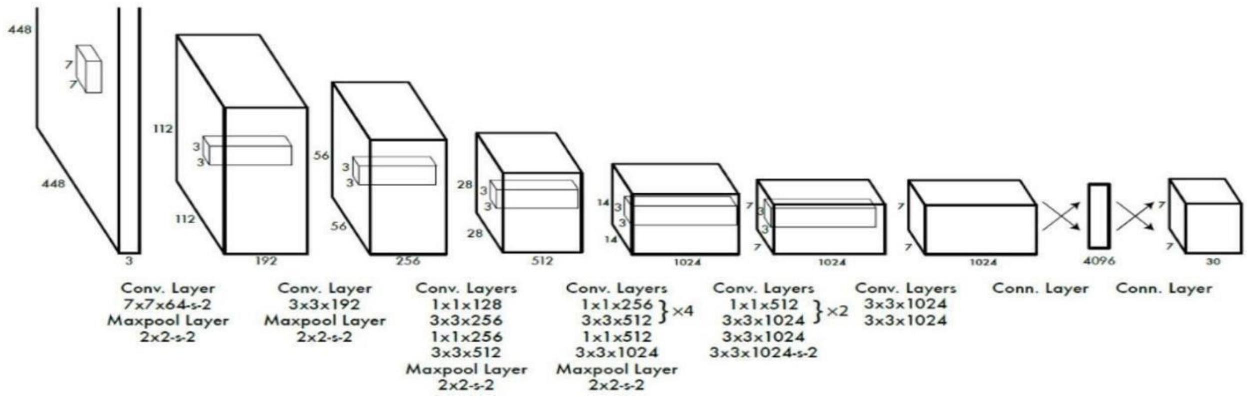


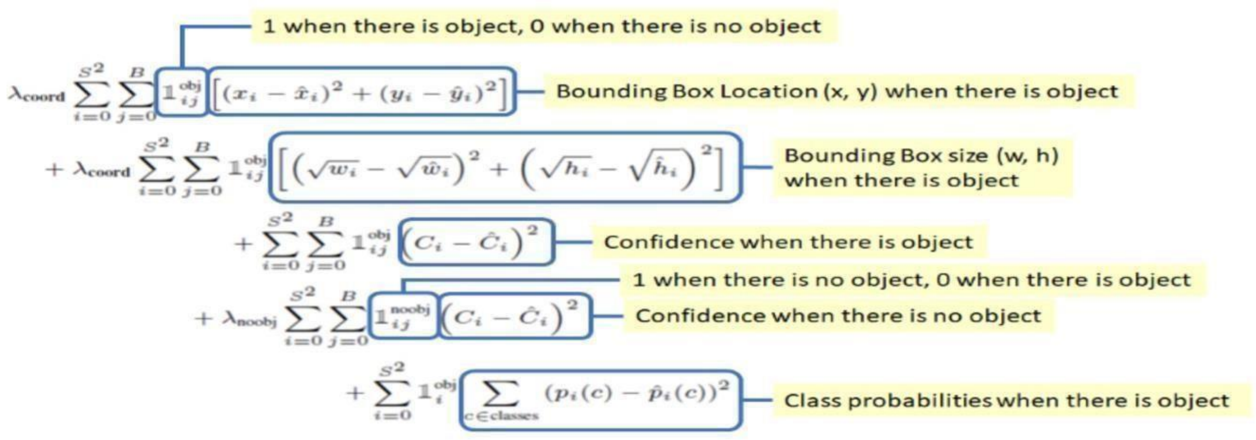
Figure 3.4: YOLO’s modelarchitecture

The model architecture consists of 24 convolutional layers followed by 4 pooling layers and 2 fully connected layers ((3.4). It uses 1 x 1 convolutions to reduce the amount of feature maps which is motivated by the Inception Modules of GoogLeNet [14]. Furthermore it applies the Leaky ReLu activation function after all layers except for the last one and uses dropout between the two fully connected layers in order to tackle overfitting.

3.4.3 Loss Function

The YOLO algorithm uses a custom loss function in order to control the different output domains and their influence on the final loss by using special hyperparameters ((3.5):

1. first term: penalizes bad locations for the center coordinates if the cell contains an object.
2. second term: penalizes bad bounding box width and height values. The square root is present so that errors in small bounding boxes are more penalizing than errors in big bounding boxes.
3. third term: penalizes small confidence scores for cells containing an object.
4. fourth term: penalizes big confidence scores for cells containing no object.
5. fifth term: simple squared classification loss.


1 when there is object, 0 when there is no object
Bounding Box Location (x, y) when there is object
Bounding Box size (w, h) when there is object
Confidence when there is object
1 when there is no object, 0 when there is object
Confidence when there is no object
Class probabilities when there is object

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Figure 3.5: YOLO’s custom loss function for every grid cell

CHAPTER 4

4 SYSTEM DESIGN

4.1 Training YOLO on BDD100K

We implemented and trained YOLO completely from scratch by using only the BDD100K dataset. Since there are numerous objects inside the images of the BDD100K dataset, we decided to increase the split size for the YOLO algorithm from 7 to 14 in order to mitigate the problem of multiple object centres falling into one cell (3.6). By deploying a much finer grid, we increased the amount of output parameters from 1127 to 4508 as well as the amount of parameters inside the last 5 layers. To achieve this, we adjusted the stride of the 23th convolutional layer from 2 to 1 to retain the output size of 14 x 14. Furthermore, we added batch normalization between all layers to increase the train speed and retained the original loss hyperparameters from the paper during training (coord=5 and noobj=0.5). Finally we trained YOLO for 100 epochs with a learning rate of 1e-5 and batch size of 10. Our YOLO algorithm produces 2 bounding box predictions per grid cell on a 14 x 14 grid. The input image has dimension (3, 448, 448) and the algorithm produces a tensor of size (14, 14, 23) as output

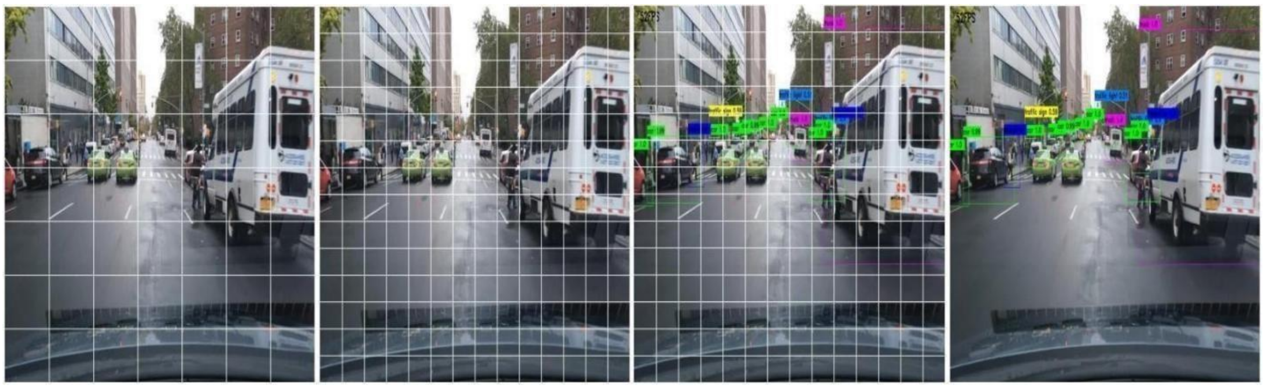


Figure 3.6: YOLO with split size 14 on the BDD100K dataset

4.2 Faster R-CNN

The architecture of faster R-CNN [7] consists of three parts: the network backbone, the region proposal network (RPN) and the older version of this algorithm called fast R-CNN (3.7). The network backbone is in general a classification network like VGG-Net or ResNet pretrained on an image classification dataset. It is used to generate high resolution feature maps and requires an image size of 640 x 640 pixels. In our approach we used ResNet50 as the backbone network pretrained on the ImageNet dataset.

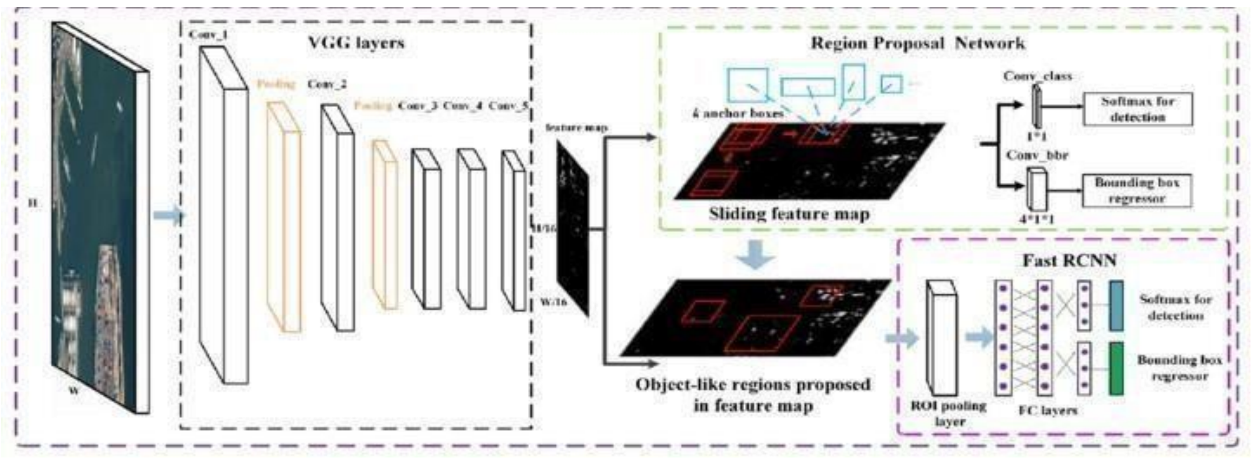


Figure 3.7: Faster R-CNN modelarchitecture

4.3 Region Proposal Network

The region proposal network consists of a single convolutional layer which is then being divided into 2 separate convolutional layers to predict a classification score and bounding boxes for the region proposals. The network uses predefined anchor boxes to generate approximately 2.000 region proposals.

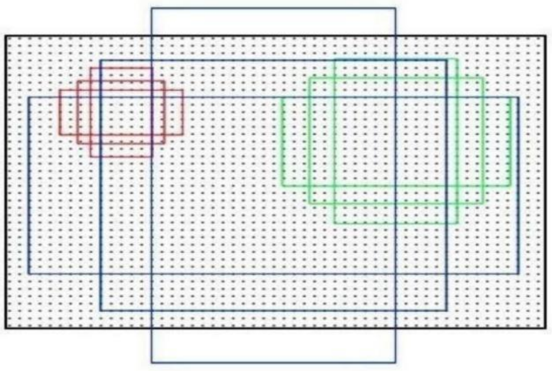


Figure 3.8: Anchor boxes with the sizes {128, 256, 512} with aspect ratios of{0.5, 1, 2}

An anchor is a smaller part of an image. In Our approach we are using anchors with the sizes *{*128, 256,512*}* with the aspect ratios of *{*0.5, 1, 2*}* which leads to 9 different anchor boxes (3.8). Each of these anchors will be slided over the window with a stride of 16 and cuts out the overlapping parts of the image. The classification score decides for all anchors if they include an object or not. The bounding box regressions are for better identifying the objects in the anchors. After the prediction the number of region proposals will be reduced, with non maximum suppression.

4.4 Fast R-CNN

Fast R-CNN, the last part of the architecture gets the high resolution feature maps from the network backbone and the region proposals from the region proposal network as input. To get a fixed size from the different sized region proposals, a method called ROI pooling (3.9) is used, which stands for region of interest pooling. For every region of interest from the input list, it takes a section of the input feature map that corresponds to it and scales it to some predefined size, in our case 7x7. The scaling is done by:

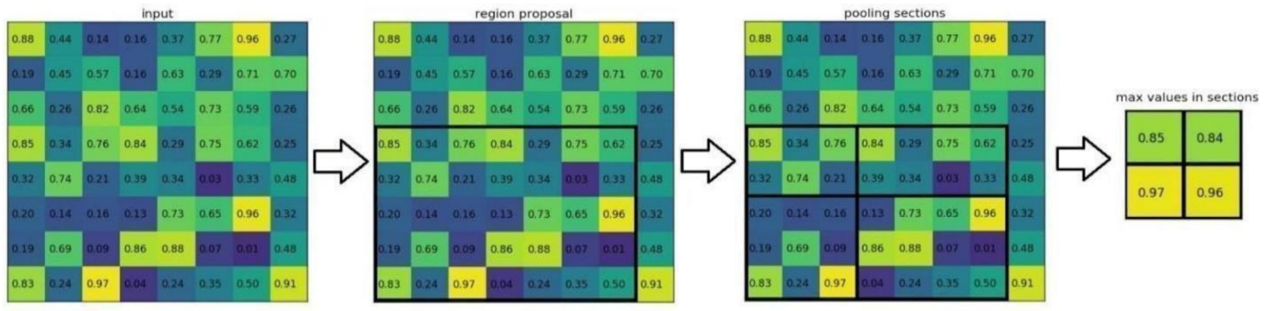


Figure 3.9: Schematic representation of ROI pooling

1. Dividing the region proposal into equal sized sections, where the number of sections is the same as the dimension of the output
2. Finding the largest value in each section
3. Copying these max values to the output buffer

After that there are only 2 fully connected layers followed by two separate output layers which predict the softmax score for every class and the corrected bounding boxes for every region proposal. As loss functions for the regression they use log loss and for the bounding boxes they use smooth L1 Loss and backpropagate the losses together by factors which determine how much the bounding box and the classification loss should affect the entire loss.

$$L_{\text{loc}}(t_u, v) = \text{X}\ \text{smooth}_{L1}(t_{ui} - v_i), \quad i \in \{x, y, w, h,\}$$

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if} |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

4.1 SYSTEM CONFIGURATION:

This project can run on commodity hardware. We ran entire project on an Intel I3 processor with 8 GB Ram, 2 GB Nvidia Graphic Processor, It also has 2 cores which runs at 1.7 GHz,2.1 GHz respectively. First part of the is training phase which takes 10-15 mins of time and the second part is testing part which only takes few seconds to make predictions and calculate accuracy.

4.2 SOFTWARE REQUIREMENTS

1. Python 3.12 in Google Collab is used for data pre-processing, model training and prediction.
2. Operating System: windows 10 and above

4.3 HARDWARE REQUIREMENTS:

1. RAM: 16 GB
2. Storage: 1 TB
3. CPU: 2 GHz or faster
4. Architecture: 64-bit

CHAPTER 5

5 IMPLEMETATION

5.1 Python

5.1.1 Introduction

* One of the most popular languages is Python. Guido van Rossum released this language in 1991. Python is available on the Mac, Windows, and Raspberry Pi operating systems. The syntax of Python is simple and identical to that of English. When compared to Python, it was seen that the other language requires a few extra lines.

*It is an interpreter-based language because code may be run line by line after it has been written. This implies that rapid prototyping is possible across all platforms. Python is a big language with a free, binary-distributed interpreter standard library.

* It is inferior to maintenance that is conducted and is straightforward to learn. It is an object-oriented, interpreted programming language. It supports several different programming paradigms in addition to object-oriented programming, including functional and procedural programming.

* It supports several different programming paradigms in addition to object-oriented programming, including practical and procedural programming. Python is mighty while maintaining a relatively straightforward syntax. Classes, highly dynamic data types, modules, and exceptions are covered. Python can also be utilised by programmes that require programmable interfaces as an external language.

Here are some key features and characteristics of Python:

- Readability: Python emphasizes code readability with its clean and intuitive syntax. It uses indentation and whitespace to structure code blocks, making it easy to understand and maintain.

- Easy to Learn: Python's simplicity and readability make it an excellent choice for beginners. Its straightforward syntax and extensive documentation make it accessible for newcomers to programming.
- Cross-platform Compatibility: Python is available for major operating systems like Windows, macOS, and Linux. This cross-platform compatibility allows developers to write code once and run it on different platforms without modifications.
- Large Standard Library: Python comes with a vast standard library that provides ready-to-use modules and functions for various tasks. It covers areas such as file I/O, networking, regular expressions, databases, and more, saving developers time and effort.
- Extensible and Modular: Python supports modular programming, enabling developers to organize code into reusable modules and packages. Additionally, Python allows integrating modules written in other languages, such as C or C++, providing flexibility and performance optimizations.
- Wide Range of Libraries and Frameworks: Python has a vibrant ecosystem with numerous third-party libraries and frameworks. These libraries, such as NumPy, pandas, TensorFlow, and Django, extend Python's capabilities for specific domains, making it a powerful tool for diverse applications.
- Object-Oriented: Python supports object-oriented programming (OOP) principles, allowing developers to create and work with classes and objects. OOP provides a structured approach to code organization, promoting code reuse and modularity.
- Dynamic Typing: Python is dynamically typed, meaning variable types are determined at runtime. Developers do not need to declare variable types explicitly, which enhances flexibility and simplifies code writing.

5.1.2 Installation

To install Python on your computer, follow these basic steps:

- Step 1: Visit the Python website Go to theofficial Python website at https://www.python.org/.
- Step 2: Select the operating system Choose the appropriate installer for your operating system. Python supports Windows, macOS, and various Linux distributions. Make sure to select the correct version that matches your operating system.
- Step 3: Check which version of Python is installed; if the 3.7.0 version is not there, uninstall it through the control panel and
- Step 4: Install Python 3.7.0 using Cmd.
- Step 5: Install the all libraries that required to run the project
- Step 6: Run

5.1.3 Python Features:

1) **Easy:** Because Python is a more accessible and straightforward language, Python programming is easier to learn.
2) **Interpreted language:** Python is an interpreted language, therefore it can be used to examine the code line by line and provide results.
3) **Open Source:** Python is a free online programming language since it is open- source.
4) **Portable:** Python is portable because the same code may be used on several computer standard
5) **libraries:** Python offers a sizable library that we mayutilize to create applications quickly.
6) **GUI:** It stands for GUI (Graphical User Interface)
7) **Dynamical typed:** Python is a dynamically typed language, therefore the type of the value will be determined at runtime.

5.1.4 library

In Python, libraries (also referred to as modules or packages) are collections of pre-written code that provide additional functionality and tools to extend the capabilities of the Python language. Libraries contain reusable code that developers can leverage to perform specific tasks without having to write everything from scratch.

Python libraries are designed to solve common problems, such as handling data, performing mathematical operations, interacting with databases, working with files, implementing networking protocols, creating graphical user interfaces (GUIs), and much more. They provide ready-to-use functions, classes, and methods that simplify complex operations and save development time.

Libraries in Python offer various advantages:

Code Reusability:

- Efficiency:
- Collaboration
- Domain-Specific Functionality
- To use a Python library, you need to install it first.

There are some libraries following:

- **Pandas:**

Pandas are a Python computer language library for data analysis and manipulation. It offers a specific operation and data format for handling time series and numerical tables. It differs significantly from the release3-clause of the BSD license. It is a well-liked open-source of opinion that is utilized in machine learning and data analysis.

Pandas are a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Pandas are a Python library used for working with data sets

- It has functions for analysing, cleaning, exploring, and manipulating data.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
- Pandas allow us to analyse big data and make conclusions based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science. Pandas are a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas have grown into one of the most popular Python libraries. It has an extremely active community of contributors. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

- **NumPy:**

The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions. It is possible to modify NumPy by utilizing a Python library. Along with line, algebra, and the Fourier transform operations, it also contains several matrices-related functions.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
- NumPywas created in 2005 byTravis Oliphant. It is an open source project and you can use it freely.
- NumPy stands for Numerical Python.
- In Python we have lists that serve the purpose of arrays, but they are slow to process.

- NumPyaims to provide an arrayobject that is up to 50x faster than traditional Python lists.
- The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
- Arrays are very frequently used in data science, where speed and resources are very important.

➢ Matplotlib:

It is a multi-platform, array-based data visualization framework built to interact with the whole SciPy stack. MATLAB is proposed as an open-source alternative. Matplotlib is a Python extension and a cross-platform toolkit for graphical plotting and visualization.

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a flexible and comprehensive set of tools for generating plots, charts, histograms, scatter plots, and more. Matplotlib is widely used in various fields, including data analysis, scientific research, and data visualization.

Here are some key features and functionalities of the Matplotlib library:

- Plotting Functions
- Customization Options
- Multiple Interfaces
- Integration with NumPy and pandas
- Subplots and Figures:
- Saving and Exporting

- **Scikit-learn:**

The most stable and practical machine learning library for Python is scikit-learn. Regression, dimensionality reduction, classification, and clustering are just a few of the helpful tools it provides through the Python interface for statistical modeling and machine learning. It is an essential part of the Python machine learning toolbox used by JP Morgan. It is frequently used in various machine learning applications, including classification and predictive analysis.

Scikit-learn (also referred to as sklearn) is a widely used open-source machine learning library for Python. It provides a comprehensive set of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and pre- processing.

Here are some key features and functionalities ofthe Scikit-learn library:

- Easy-to-Use Interface:
- Broad Range of Algorithms:
- Data Pre-processing and Feature Engineering:
- Model Evaluation and Validation:
- Integration with NumPy and pandas:
- Robust Documentation and Community Support:

➢ **Keras:**

* Google's Keras is a cutting-edge deep learning API for creating neural networks. It is created in Python and is designed to simplify the development of neural networks. Additionally, it enables the use of various neural networks for computation. Deep learning models are developed and tested using the free and open-source Python software known as Keras.

Keras is a high-level deep learning library for Python. It is designed to provide a user-friendly and intuitive interface for building and training deep learning models. Keras acts as a front-end API, allowing developers to define and configure neural networks while leveraging the computational backend engines, such as Tensor Flow or Theano.

Here are some key features and functionalities of the Keras library:

- User-Friendly API
- Multi-backend Support
- Wide Range of Neural Network Architectures
- Pre-trained Models and Transfer Learning:
- Easy Model Training and Evaluation:
- GPU Support:

➢ **h5py:**

* The h5py Python module offers an interface for the binary HDF5 data format. Thanks to p5py, the top can quickly halt the vast amount of numerical data and alter it using the NumPy library. It employs common syntax for Python, NumPy, and dictionary arrays.

h5py is a Python library that provides a simple and efficient interface for working with datasets and files in the Hierarchical Data Format 5 (HDF5) format.

Here are some key features and functionalities of the h5py library:

- HDF5 File Access
- Dataset Handling:
- Group Organization:
- Attributes:
- Compatibility with NumPy
- Performance

➢ **Tensor flow**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow is a popular open-source library for machine learning and deep learning. It provides a comprehensive set of tools, APIs, and computational resources for building and training various types of machine learning models, especially neural networks.

Here are some key features and functionalities of TensorFlow:

- Neural Network Framework:
- Computational Graphs
- Automatic Differentiation
- GPU and TPU Support
- Distributed Computing
- Deployment Capabilities

5.2 Sample Code

```
import cv2
import pandas as pd
import numpy as np
from ultralytics import YOLO
from tracker import*
import time
from math import dist
model=YOLO('yolov8s.pt')

# Defines a callback function RGB which will be triggered when a mouse event occurs.

def RGB(event, x, y, flags, param):
    if event == cv2.EVENT_MOUSEMOVE :
        colorsBGR = [x, y]
        print(colorsBGR)


cv2.namedWindow('RGB')                    # Creates a named window for displaying the
    video frames.
cv2.setMouseCallback('RGB', RGB)          # Associates the callback function RGB with
    the 'RGB' window.

cap=cv2.VideoCapture('stock-footage-traffic-on-the-indian-roads.WEBM')

my_file = open("coco.txt", "r")
data = my_file.read()
class_list = data.split("\n")
#print(class_list)

count=0    # count cars which satisfy both conditions
```

```
tracker=Tracker()

cy1=322    # line 1
cy2=360    # line 2

offset=6

vh_down={}    # dictionary to store id and location
counter=[]    # going down



vh_up={}      # dictionary to store id and location
counter1=[]  # going up

high_speed_cars = []


while True:
    ret,frame = cap.read()           # Reads the next frame from the video capture.
    if not ret:
        break
    count += 1
    if count % 3 != 0:
        continue
    frame=cv2.resize(frame,(1020,500))        # Resizes the frame to a specific width and height
     using OpenCV's resize function.


    results=model.predict(frame)
    #print(results)                        # We have bounding box co-ordinates of rectangle [0,1,2,3]
                                   # We have confidence level [4] and class [5]
```

```
a=results[0].boxes.data
px=pd.DataFrame(a).astype("float")
#print(px)
list=[]

for index,row in px.iterrows():
    #print(row)
    x1=int(row[0])                    # top left
    y1=int(row[1])                    # top right
    x2=int(row[2])                    # bottom left
    y2=int(row[3])                    # bottom right
    d=int(row[5])                    # class (car)
    c=class_list[d]
    if 'car' in c:
        list.append([x1,y1,x2,y2])
bbox_id=tracker.update(list)
for bbox in bbox_id:
    x3,y3,x4,y4,id=bbox
    cx=int(x3+x4)//2        #centre point of car x
    cy=int(y3+y4)//2        #centre point of car y

    cv2.rectangle(frame,(x3,y3),(x4,y4),(0,0,255),2)

    if cy1<(cy+offset) and cy1 > (cy-offset):      #condition for cy1 : car centre touches the
line1  : generate ID and current time
       vh_down[id]=time.time()

    if id in vh_down:                              # if ID in our ductionary count only those

       if cy2<(cy+offset) and cy2 > (cy-offset):   #condition for cy2 : car centre touches the
```

```
line2  : generate current position
        elapsed_time=time.time() - vh_down[id]    # current time(line2) - time at line 1
        if counter.count(id)==0:
           counter.append(id)
           distance = 10                          # dist between L1 and L2 meters
           a_speed_ms = distance / elapsed_time        # calculate speed
           a_speed_kh = a_speed_ms * 3.6              # Km/Hr
           cv2.circle(frame,(cx,cy),4,(0,0,255),-1)

cv2.putText(frame,str(id),(x3,y3),cv2.FONT_HERSHEY_COMPLEX,0.6,(255,255,255),1)
           if a_speed_kh < 40:
cv2.putText(frame,str(int(a_speed_kh))+'Km/h',(x4,y4),cv2.FONT_HERSHEY_COMPLEX
,0.8,(0,255,0),2) # green text for speed < 25
           else:

cv2.putText(frame,str(int(a_speed_kh))+'Km/h',(x4,y4),cv2.FONT_HERSHEY_COMPLEX
,0.8,(0,0,255),2)  # Red Text
               if id not in high_speed_cars:
                   high_speed_cars.append(id)
               if a_speed_kh > 40:
                   with open("over_speeding_cars_ID.txt", "a") as f:       # create file to store IDS
                       f.write(str(id) + "\n")


   #####going UP#####
   if cy2<(cy+offset) and cy2 > (cy-offset):  #condition for cy2 : car centre touches the
line2  : generate ID and current time
      vh_up[id]=time.time()
   if id in vh_up:
      if cy1<(cy+offset) and cy1 > (cy-offset):
        elapsed1_time=time.time()  -  vh_up[id]
        if counter1.count(id)==0:
            counter1.append(id)
            distance1 = 10 # meters
            a_speed_ms1 = distance1 / elapsed1_time
            a_speed_kh1 = a_speed_ms1 * 3.6

            cv2.circle(frame,(cx,cy),4,(0,0,255),-1)

cv2.putText(frame,str(id),(x3,y3),cv2.FONT_HERSHEY_COMPLEX,0.6,(255,255,255),1)
            if a_speed_kh1 < 40:

cv2.putText(frame,str(int(a_speed_kh1))+'Km/h',(x4,y4),cv2.FONT_HERSHEY_COMPLE
X,0.8,(0,255,0),2) # green text
            else:

cv2.putText(frame,str(int(a_speed_kh1))+'Km/h',(x4,y4),cv2.FONT_HERSHEY_COMPLE
X,0.8,(0,0,255),2) # red text
                if id not in high_speed_cars:
                    high_speed_cars.append(id)

                if a_speed_kh1 > 40:
                    with open("over_speeding_cars_ID.txt", "a") as f:
                        f.write(str(id) + "\n")


cv2.line(frame,(100,cy1),(963,cy1),(255,255,255),1)

cv2.putText(frame,('L1'),(100,320),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,255),2)

cv2.line(frame,(10,cy2),(1019,cy2),(255,255,255),1)

cv2.putText(frame,('L2'),(10,360),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,255),2)
```

```
    d=(len(counter))
    u=(len(counter1))
    cv2.putText(frame,('goingdown:-
    ')+str(d),(60,90),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,255),2)

    cv2.putText(frame,('goingup:-
    ')+str(u),(60,130),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,255),2)



    cv2.imshow("RGB", frame)
    if cv2.waitKey(1)&0xFF==27:
        break
cap.release()
cv2.destroyAllWindows()
```

Tracker code:

```
import math


class Tracker:                   # The Tracker class has two main attributes: center_points and id_count.
    def _init_(self):
        # Store the center positions of the objects
        self.center_points = {}
        # Keep the count of the IDs
        # each time a new object id detected, the count will increase by one
        self.id_count = 0


    def update(self, objects_rect):
        # Objects boxes and ids
        objects_bbs_ids = []

        # Get center point of new object
        for rect in objects_rect:
            x, y, w, h = rect
            cx = (x + x + w) // 2
            cy = (y + y + h) // 2

            # Find out if that object was detected already
            same_object_detected = False
            for id, pt in self.center_points.items():       #calculates the euclidean distance
                dist = math.hypot(cx - pt[0], cy - pt[1])  #calculates the length of the hypotenuse

                if dist < 35:
                    self.center_points[id] = (cx, cy) #proximity between the newly detected object and each
     previously tracked object.
#                    print(self.center_points)
                    objects_bbs_ids.append([x, y, w, h, id])
                    same_object_detected = True        # newly detected object corresponds to the same object
```

```
previously tracked.
               break

       # New object is detected we assign the ID to that object
       if same_object_detected is False:
           self.center_points[self.id_count] = (cx, cy)
           objects_bbs_ids.append([x, y, w, h, self.id_count])
           self.id_count += 1

   # Clean the dictionary by center points to remove IDS not used anymore
   new_center_points = {}
   for obj_bb_id in objects_bbs_ids:
       _, _, _, _, object_id = obj_bb_id
       center = self.center_points[object_id]
       new_center_points[object_id] = center

   # Update dictionary with IDs not used removed
   self.center_points  =  new_center_points.copy()
   return objects_bbs_ids
```

CHAPTER 6

6. TESTING

Implementation and Testing:

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

Implementation

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

Module Testing

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

Integration Testing

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

CHAPTER 7

7. EXPERIMENTAL RESULTS WITH SCREENSHOTS

7.1 Training

YOLO was trained for 81 Epochs with a decreasing learning rate of 1e-5 and batchsize 10. Faster R-CNN was trained for 60 epochs with a decreasing learning rate of 1e-4 and batchsize 16. The normalized total loss and the mAP of the training progress is shown in

7.2 Results

The implementation of the models and the processed data are found in this GitHub repository. Furthermore, we provide videos of real-time object detection with our models: Faster R-CNN and YOLO. We measured the FPS and mAP for the evaluation and comparison of YOLO and faster R-CNN on a NVIDIA V100 SXM2 32GB

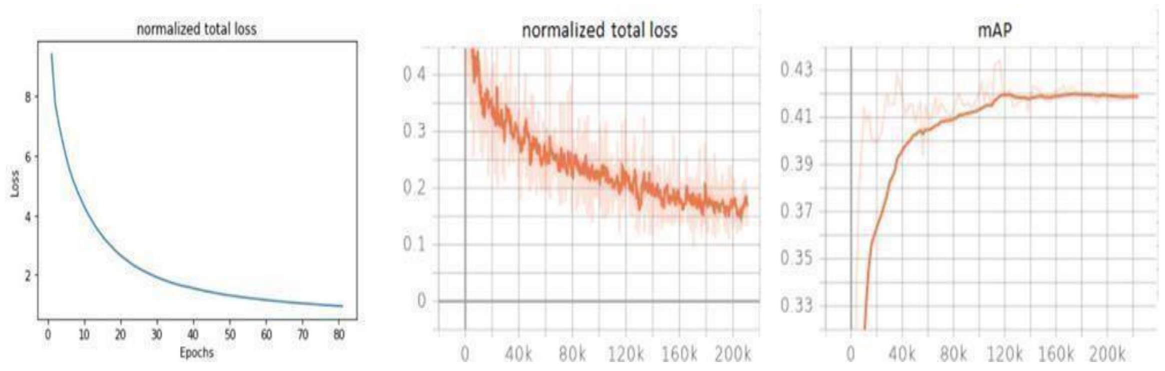


Figure 4.1: Normalized total loss for YOLO (left), normalized total loss for Faster R-CNN (middle) and mean average precision for Faster R-CNN (right)

Table 4.1: Comparison between YOLO, Faster R-CNN and Hybrid incremental net in mAP and FPS on the BDD100K dataset

	mAP	FPS
YOLO	18,6	212
FasterR-CNN	41,8	17,1
Hybridincrementalnet	45,7	N/A

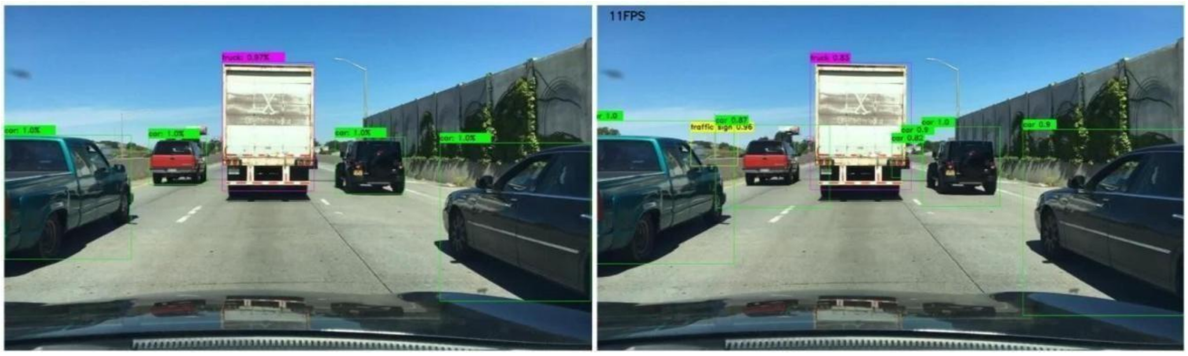
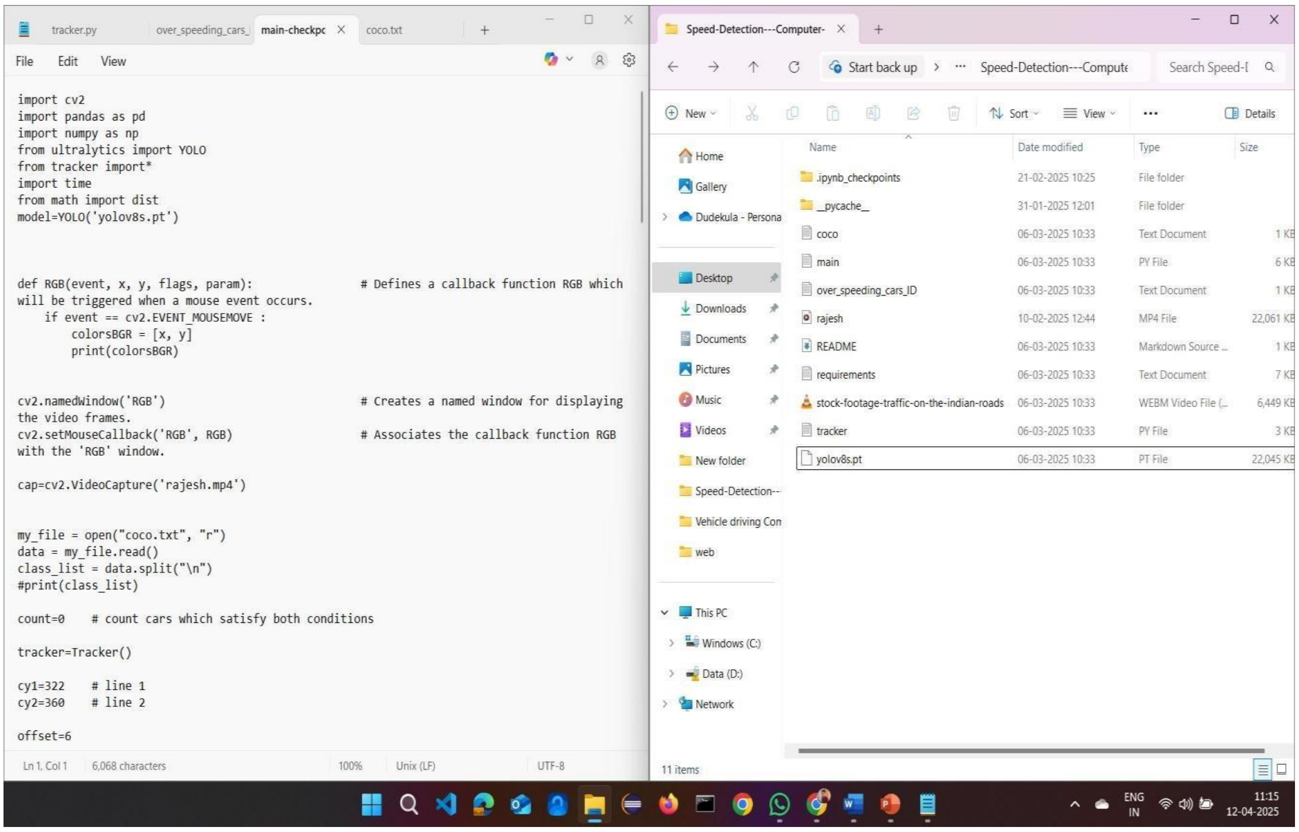


Figure 4.2: Results on BDD100K: Faster R-CNN (left) and YOLO (right).

More results from the YOLO and the Faster R-CNN architecture are shown in the appendix

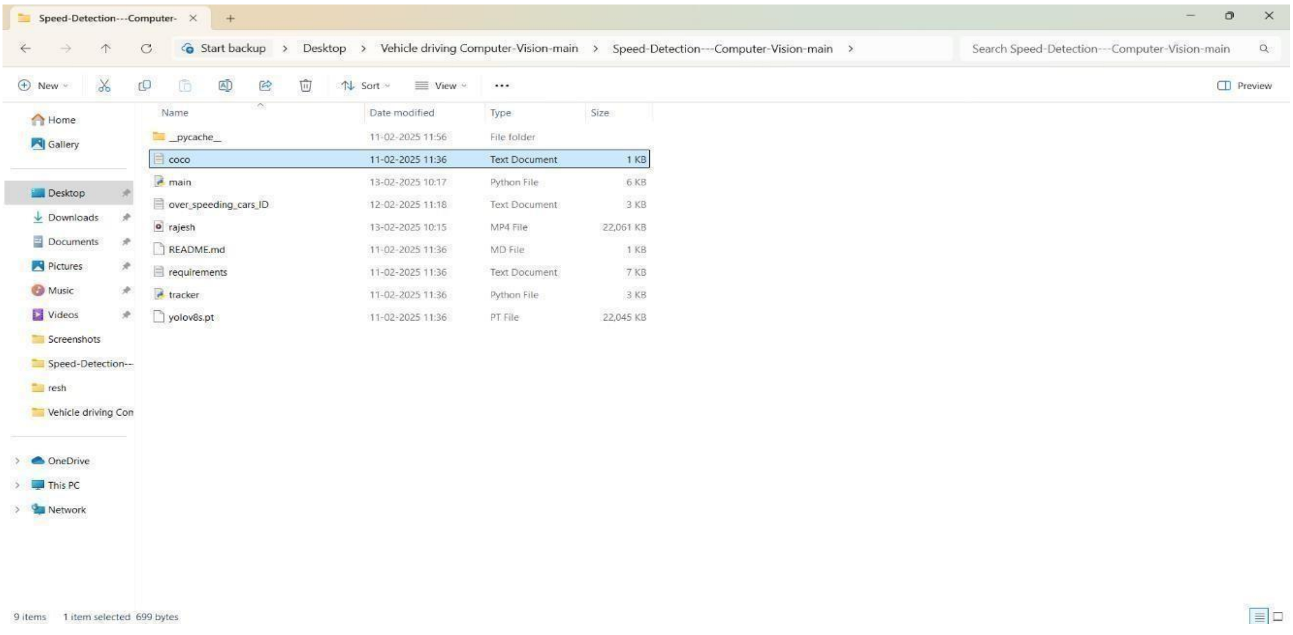
SCREENSHOTS



```
import cv2
import pandas as pd
import numpy as np
from ultralytics import YOLO
from tracker import*
import time
from math import dist
model=YOLO('yolov8s.pt')



def RGB(event, x, y, flags, param):                # Defines a callback function RGB which
will be triggered when a mouse event occurs.
    if event == cv2.EVENT_MOUSEMOVE :
        colorsBGR = [x, y]
        print(colorsBGR)


cv2.namedWindow('RGB')                             # Creates a named window for displaying
the video frames.
cv2.setMouseCallback('RGB', RGB)                   # Associates the callback function RGB
with the 'RGB' window.

cap=cv2.VideoCapture('rajesh.mp4')


my_file = open("coco.txt", "r")
data = my_file.read()
class_list = data.split("\n")
#print(class_list)

count=0    # count cars which satisfy both conditions

tracker=Tracker()

cy1=322    # line 1
cy2=360    # line 2

offset=6
```

Name	Date modified	Type	Size
.ipynb_checkpoints	21-02-2025 10:25	File folder	
__pycache__	31-01-2025 12:01	File folder	
coco	06-03-2025 10:33	Text Document	1 KB
main	06-03-2025 10:33	PY File	6 KB
over_speeding_cars_ID	06-03-2025 10:33	Text Document	1 KB
rajesh	10-02-2025 12:44	MP4 File	22,061 KB
README	06-03-2025 10:33	Markdown Source ...	1 KB
requirements	06-03-2025 10:33	Text Document	7 KB
stock-footage-traffic-on-the-indian-roads	06-03-2025 10:33	WEBM Video File (...	6,449 KB
tracker	06-03-2025 10:33	PY File	3 KB
yolov8s.pt	06-03-2025 10:33	PT File	22,045 KB

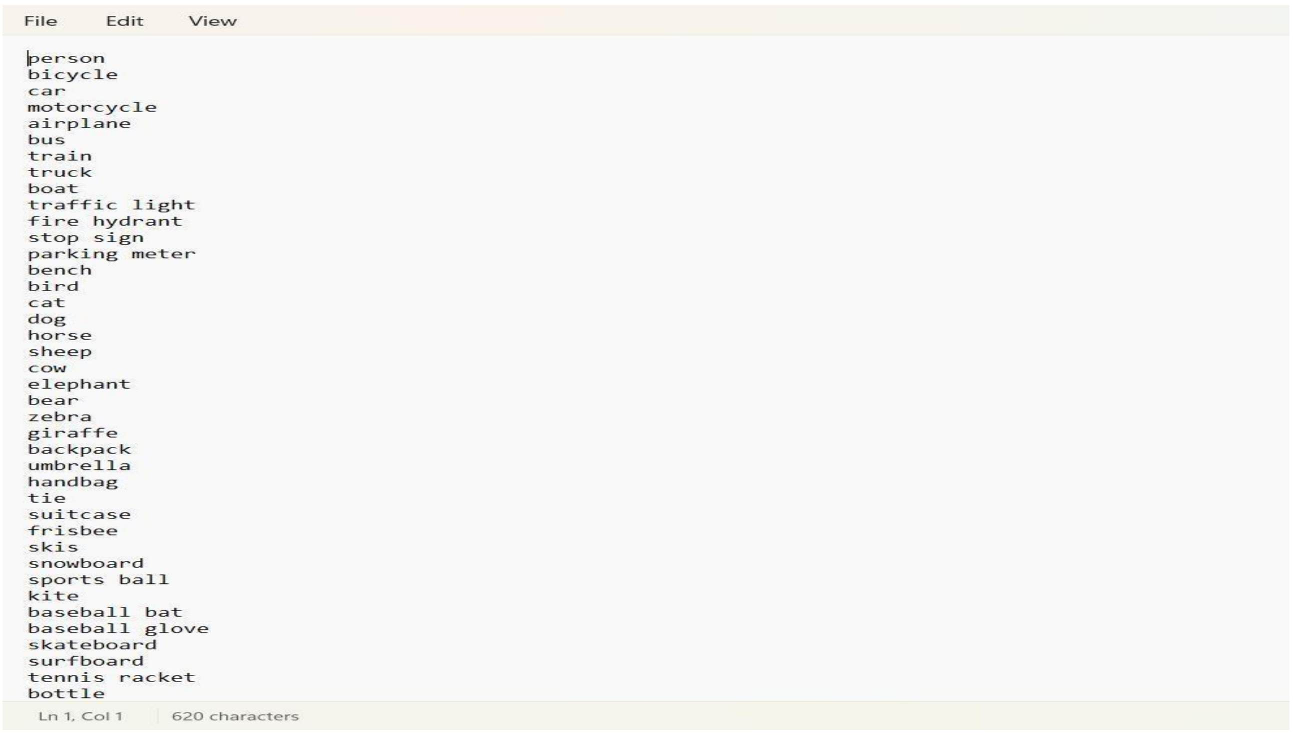
Figure 1: Code and project folder


Speed-Detection---Computer-
Start backup > Desktop > Vehicle driving Computer-Vision-main > Speed-Detection---Computer-Vision-main >
Search Speed-Detection---Computer-Vision-main
New
Sort
View
Preview
Home
Gallery
Desktop
Downloads
Documents
Pictures
Music
Videos
Screenshots
Speed-Detection--
resh
Vehicle driving Con
OneDrive
This PC
Network

Name	Date modified	Type	Size
__pycache__	11-02-2025 11:56	File folder	
coco	11-02-2025 11:36	Text Document	1 KB
main	13-02-2025 10:17	Python File	6 KB
over_speeding_cars_ID	12-02-2025 11:18	Text Document	3 KB
rajesh	13-02-2025 10:15	MP4 File	22,061 KB
README.md	11-02-2025 11:36	MD File	1 KB
requirements	11-02-2025 11:36	Text Document	7 KB
tracker	11-02-2025 11:36	Python File	3 KB
yolov8s.pt	11-02-2025 11:36	PT File	22,045 KB

9 items 1 item selected 699 bytes

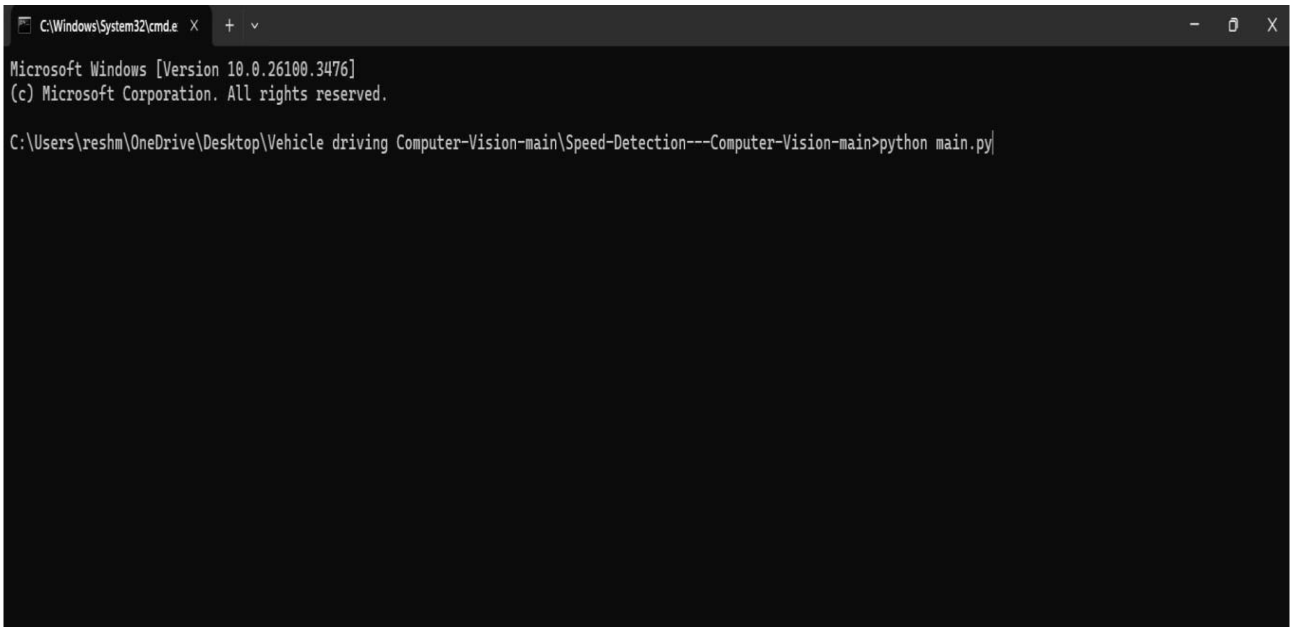
Figure 2: Project folder



```
person
bicycle
car
motorcycle
airplane
bus
train
truck
boat
traffic light
fire hydrant
stop sign
parking meter
bench
bird
cat
dog
horse
sheep
cow
elephant
bear
zebra
giraffe
backpack
umbrella
handbag
tie
suitcase
frisbee
skis
snowboard
sports ball
kite
baseball bat
baseball glove
skateboard
surfboard
tennis racket
bottle
```

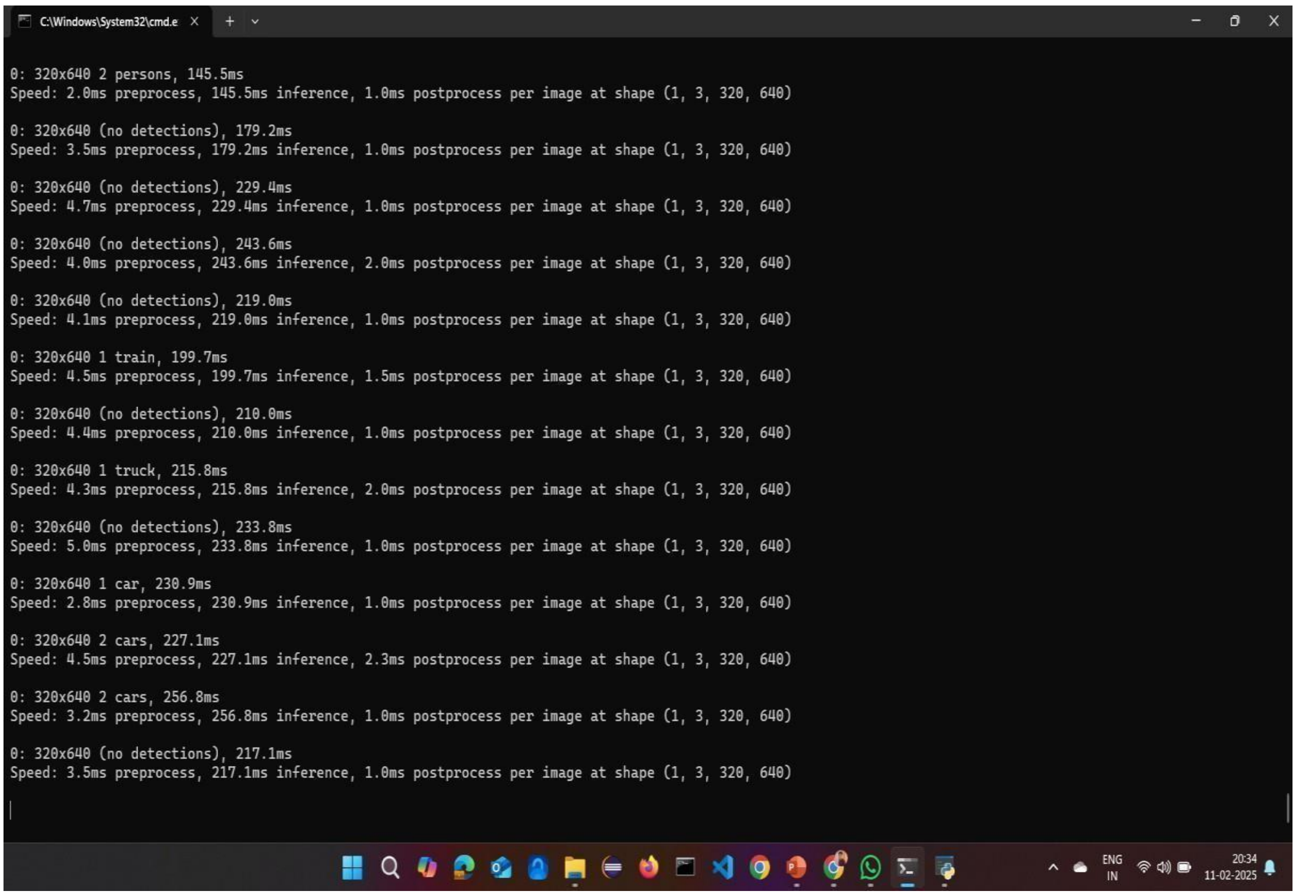
Ln 1, Col 1 620 characters

Figure 3: In below screen we are showing dataset details used in this project



```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\reshm\OneDrive\Desktop\Vehicle driving Computer-Vision-main\Speed-Detection---Computer-Vision-main>python main.py
```

figure 5: Command prompt used to run the code,



```
0: 320x640 2 persons, 145.5ms
Speed: 2.0ms preprocess, 145.5ms inference, 1.0ms postprocess per image at shape (1, 3, 320, 640)

0: 320x640 (no detections), 179.2ms
Speed: 3.5ms preprocess, 179.2ms inference, 1.0ms postprocess per image at shape (1, 3, 320, 640)

0: 320x640 (no detections), 229.4ms
Speed: 4.7ms preprocess, 229.4ms inference, 1.0ms postprocess per image at shape (1, 3, 320, 640)

0: 320x640 (no detections), 243.6ms
Speed: 4.0ms preprocess, 243.6ms inference, 2.0ms postprocess per image at shape (1, 3, 320, 640)

0: 320x640 (no detections), 219.0ms
Speed: 4.1ms preprocess, 219.0ms inference, 1.0ms postprocess per image at shape (1, 3, 320, 640)

0: 320x640 1 train, 199.7ms
Speed: 4.5ms preprocess, 199.7ms inference, 1.5ms postprocess per image at shape (1, 3, 320, 640)

0: 320x640 (no detections), 210.0ms
Speed: 4.4ms preprocess, 210.0ms inference, 1.0ms postprocess per image at shape (1, 3, 320, 640)

0: 320x640 1 truck, 215.8ms
Speed: 4.3ms preprocess, 215.8ms inference, 2.0ms postprocess per image at shape (1, 3, 320, 640)

0: 320x640 (no detections), 233.8ms
Speed: 5.0ms preprocess, 233.8ms inference, 1.0ms postprocess per image at shape (1, 3, 320, 640)

0: 320x640 1 car, 230.9ms
Speed: 2.8ms preprocess, 230.9ms inference, 1.0ms postprocess per image at shape (1, 3, 320, 640)

0: 320x640 2 cars, 227.1ms
Speed: 4.5ms preprocess, 227.1ms inference, 2.3ms postprocess per image at shape (1, 3, 320, 640)

0: 320x640 2 cars, 256.8ms
Speed: 3.2ms preprocess, 256.8ms inference, 1.0ms postprocess per image at shape (1, 3, 320, 640)

0: 320x640 (no detections), 217.1ms
Speed: 3.5ms preprocess, 217.1ms inference, 1.0ms postprocess per image at shape (1, 3, 320, 640)
```

figure 6: Runing the code

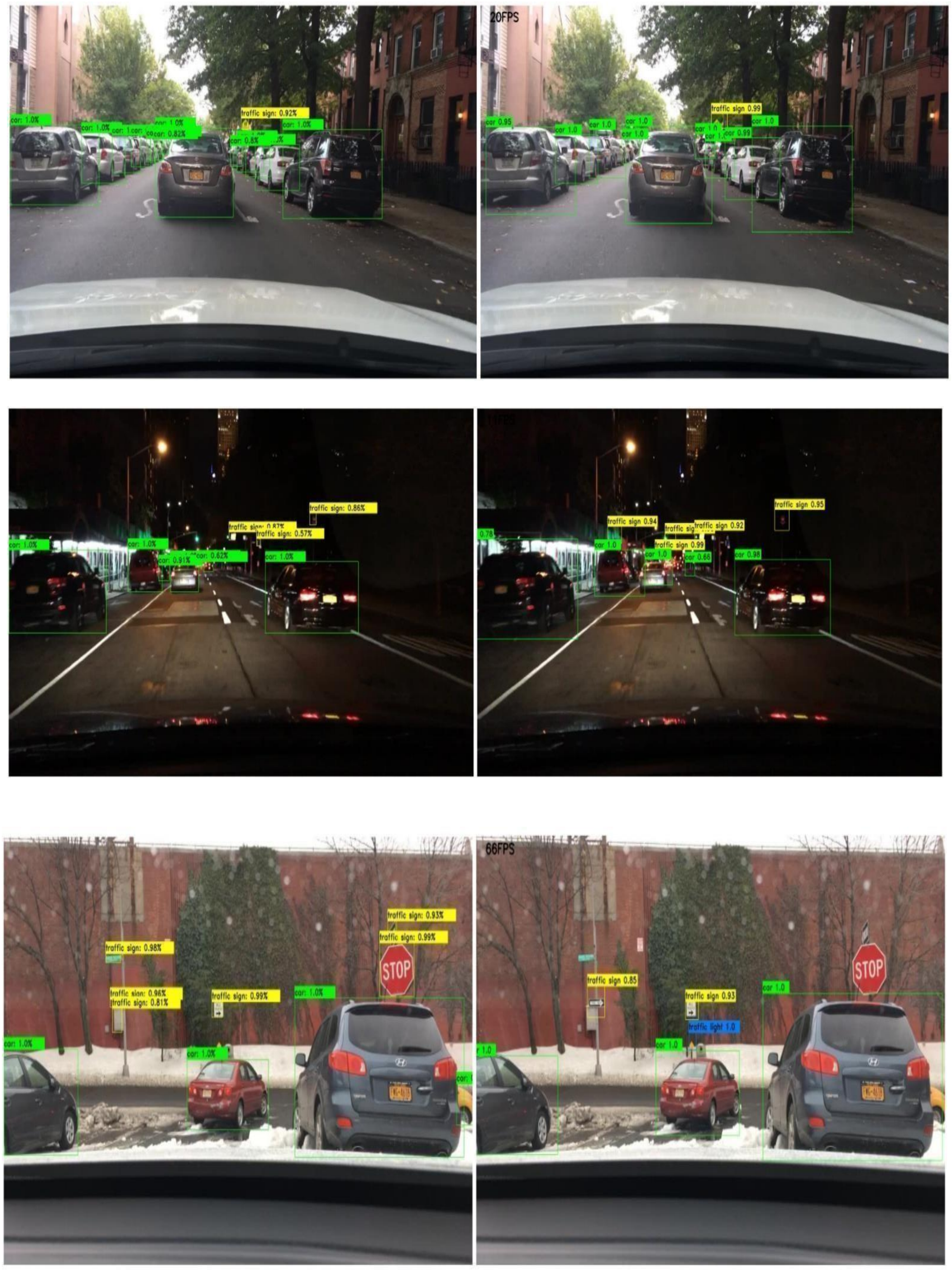
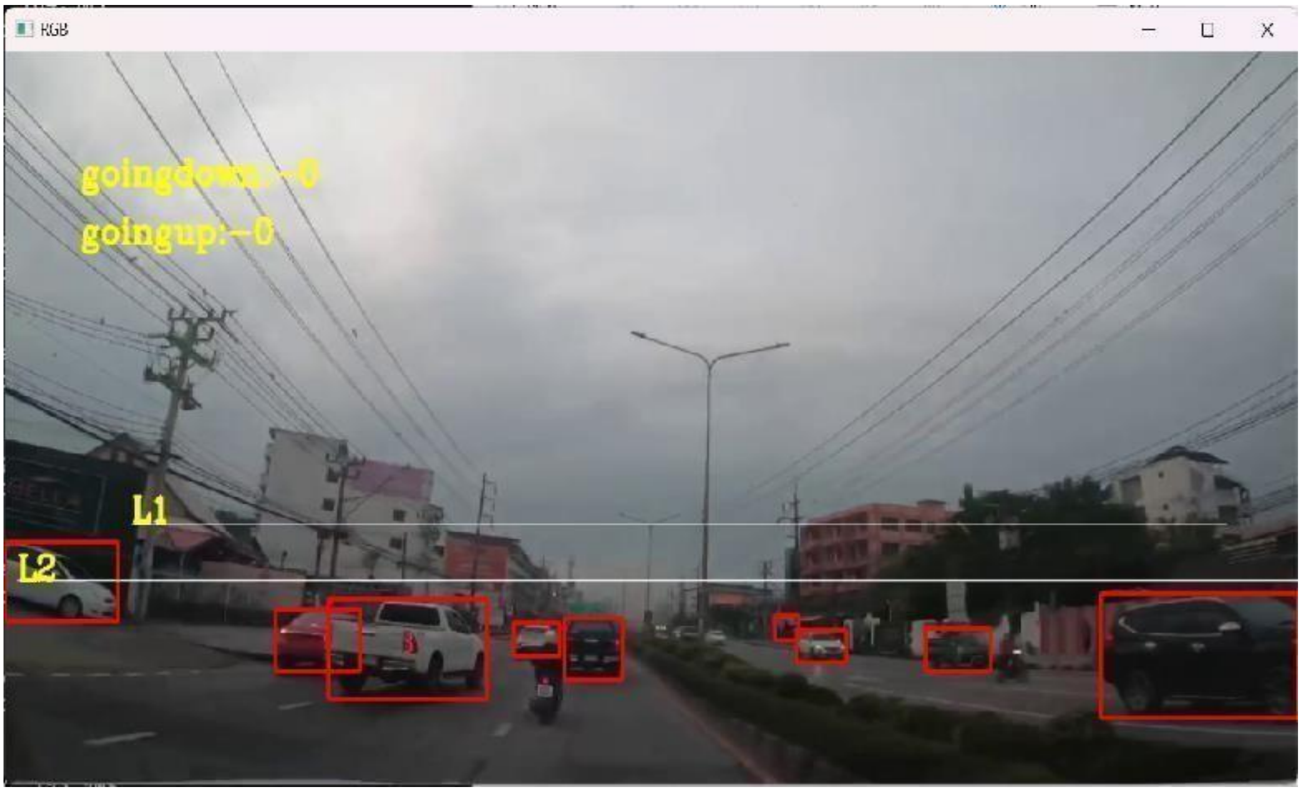
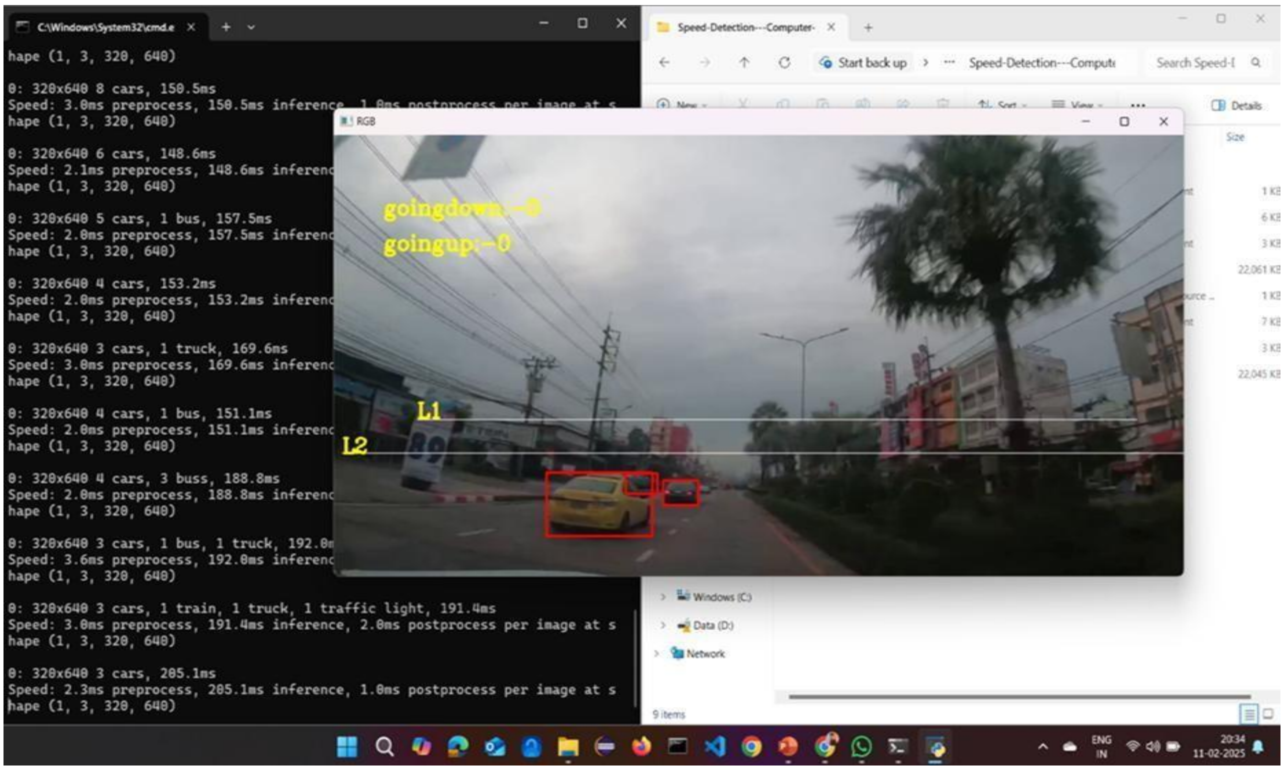


Figure 7: Identifying the objects





```
hape (1, 3, 320, 640)

0: 320x640 8 cars, 150.5ms
Speed: 3.0ms preprocess, 150.5ms inference, 1.0ms postprocess per image at s
hape (1, 3, 320, 640)

0: 320x640 6 cars, 148.6ms
Speed: 2.1ms preprocess, 148.6ms inferenc
hape (1, 3, 320, 640)

0: 320x640 5 cars, 1 bus, 157.5ms
Speed: 2.0ms preprocess, 157.5ms inferenc
hape (1, 3, 320, 640)

0: 320x640 4 cars, 153.2ms
Speed: 2.0ms preprocess, 153.2ms inferenc
hape (1, 3, 320, 640)

0: 320x640 3 cars, 1 truck, 169.6ms
Speed: 3.0ms preprocess, 169.6ms inferenc
hape (1, 3, 320, 640)

0: 320x640 4 cars, 1 bus, 151.1ms
Speed: 2.0ms preprocess, 151.1ms inferenc
hape (1, 3, 320, 640)

0: 320x640 4 cars, 3 buss, 188.8ms
Speed: 2.0ms preprocess, 188.8ms inferenc
hape (1, 3, 320, 640)

0: 320x640 3 cars, 1 bus, 1 truck, 192.0m
Speed: 3.6ms preprocess, 192.0ms inferenc
hape (1, 3, 320, 640)

0: 320x640 3 cars, 1 train, 1 truck, 1 traffic light, 191.4ms
Speed: 3.0ms preprocess, 191.4ms inference, 2.0ms postprocess per image at s
hape (1, 3, 320, 640)

0: 320x640 3 cars, 205.1ms
Speed: 2.3ms preprocess, 205.1ms inference, 1.0ms postprocess per image at s
hape (1, 3, 320, 640)
```

figure 7, 8: The above images are output ofthe code

CHAPTER 8

8. CONCLUSION

The project successfully demonstrated the development and implementation of an Autonomous Vehicle Navigation System using deep learning and computer vision, focusing on real-time object detection. By leveraging powerful models such as YOLO and Faster R-CNN, the system achieved significant performance in detecting and tracking various objects, including vehicles and pedestrians, under diverse conditions. Integration of modules for navigation, decision-making, and object recognition provided a comprehensive solution capable of supporting autonomous driving.

Through extensive training on the BDD100K dataset and evaluation using metrics like mean average precision (mAP) and frames per second (FPS), the system's efficiency and accuracy were validated. While the current solution offers real-time insights and effective monitoring capabilities, some limitations such as accuracy under varying lighting/weather conditions and dependency on hardware quality were observed.

The project lays a strong foundation for future improvements, such as incorporating LiDAR for better depth perception, enhancing robustness in adverse environments, and enabling multi-modal sensor fusion for higher autonomy. With further refinement and real-world testing, this system holds potential for deployment in intelligent traffic surveillance and next-generation self-driving vehicles.

In our project we implemented and trained a one-stage detector YOLO and a two-stage detector Faster R-CNN on the BDD100K dataset in the context of autonomous driving. As expected, the results of the evaluation showed that Faster R-CNN has a higher accuracy but lower FPS. In comparison YOLO has a much higher FPS, but also much lower accuracy because of its simple architecture. Future work includes further experiments with newer models, for example the newer versions of YOLO, since we used the first version of YOLO in this project. Future work in the long term would be reaching performances with high accuracy and high FPS which are suitable for the goal of autonomous driving.

CHAPTER 9

9. REFERENCES

[1] N. Miura, A. Nagasaka, and T. Miyatake, “Extraction of finger-vein patterns using maximum curvature points in image profiles,” Proc. IAPR Conf. Machine Vis. & Appl. pp. 347-350, Tsukuba Science City, May 2005.

[2] X. Wu, R. He, Z. Sun and T. Tan, “A light CNN for deep face representation with noisy labels,” arXiv:1151.02683v2, 2016.

[3] A. Kumar and Y. Zhou, “Human identification using finger images,” IEEE Trans. Image Process., vol. 21, pp. 2228-2244, Apr. 2012

[4] The Hong Kong Polytechnic University Finger Image Database (Version 1.0), http://www.comp.polyu.edu.hk/~csajaykr/fvdatavase.htm, 2012

[5] D. M. Weber and D. Casasent, “Quadratic Gabor filters for object detection,” IEEE Trans. Image Process., vol. 10, pp. 2180230, Feb. 2001

[6] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. arXiv preprint arXiv:1503.03832, 2015.

[7] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014

[8] D. Chen, X. Cao, L. Wang, F. Wen and J. Sun, “Bayesian face revisited: A joint formulation,” Proc. ECCV 2012, 2012.

[9] N. Miura, A. Nagasaka, and T. Miyatake, “Feature extraction of finger vein patterns based on repeated line tracking and its application to personal identification,” Machine Vis. & Appl., pp. 194-203, Jul, 2004

[10] Z. Zhao and A. Kumar, “Accurate periocular recognition under less constrained environment using semantics-assisted convolutional neural network,” IEEE Trans. Info. Forensics & Security, May, 2017

[11] L. Dong, G. Yang, Y. Yin, F. Liu and X. Xi. “Finger vein verification based on a personalized best patches map,” Proc. 2nd IJCB 2014, Tampa, Sep.-Oct. 2014.

[12] L. Chen, J. Wang, S. Yang and H. He, “A finger vein image-based personal identification system with self-adaptive illuminance control,” IEEE Trans. Instrumentation & Measurement, 2017.