SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY GAUTAM BUDDHA UNIVERSITY, GREATER NOIDA, 201312, U. P., (INDIA)

Candidate’s Declaration

We, hereby, certify that the work embodied in this project report entitled “Vehicle Detection System” in partial fulfillment of the requirements for the award of the degree of Msc Computer Science, submitted to the School of Information and Communication Technology, Gautam Buddha University, Greater Noida, is an authentic record of our own work carried out under the supervision of **Prof. Prakash Kumar Saraswat**, School of ICT. The matter presented in this report has not been submitted in any other University/Institute for the award of any other degree or diploma.

NAME	ROLL NO	SIGNATURE
NISHANT GOEL	235/PMD/001	
ABDUL AJIZ KHAN	235/PMD/002	
RAVINDRA KUMAR	235/PMD/003	

Supervisor’s Certification

This is to certify that the above statement made by the candidates is correct to the best of my knowledge and belief. However, responsibility for any plagiarism-related issue solely stands with the students.

Signature of the Supervisor:
Name with Designation: **Prof. Prakash Kumar Saraswat** (Supervisor)
Date:
Place: Greater Noida

Acknowledgement

I, Nishant Goel (235/PMD/001) and Abdul Aziz Khan (235/PMD/002) and Ravindra Kumar (235/PMD/003) would like to thank those who are involved in the endeavor for their kind cooperation for its successful completion. At the outset, I wish to express my sincere gratitude to all those people who helped me to complete this project in an efficient manner.

I offer my special thanks to my project guide **Prof. Prakash Kumar Saraswat**, Assistant Professor (IT), Gautam buddha University, without whose help and support throughout this project would not have been this successful. Her guidance helped me with all the time of my project and writing this Project Report.

Also, I would like to thanks to my other subject faculties or professors at my college campus i.e. Gautam buddha University, who helped us when I needed their help in my problems that I faced while working on this project. Most of all and more than ever, I would like to thank my family members for their warmness, support, kindness and patience. I am thankful to all my friends who always advised and motivated me throughout the course.

Index

List Of Figure's

CHAPTER 1:

INTRODUCTION

Vehicle detection is a technology that identifies and tracks vehicles in a given area using various sensors and computer vision techniques. This system is widely used in traffic monitoring, surveillance, toll collection, and autonomous driving. The primary goal of vehicle detection is to analyze traffic patterns, enhance road safety, and provide insights for urban planning. The detection process typically involves the use of cameras, sensors, and machine learning models to identify vehicles based on their shape, size, and motion.
Vehicle detection systems can be implemented in various settings, such as **smart traffic management**, **parking automation**, and **law enforcement**. By integrating artificial intelligence and deep learning techniques, these systems can differentiate between different types of vehicles, detect violations, and improve road efficiency. Most modern vehicle detection systems rely on **real-time image processing** and **object recognition algorithms** to analyse traffic movement.

1.1 General Ideas

Our project is called the **Vehicle Detection System**, which has been developed with guidance from our faculty to integrate intelligent detection mechanisms. This system aims to identify vehicles in real time using computer vision and machine learning algorithms. The application can be utilized for traffic monitoring, security surveillance, and autonomous vehicle navigation.

1.1.1 System Objective

In today's digital world, automated vehicle detection plays a crucial role in enhancing transportation efficiency and safety. Manually tracking and analysing vehicle movement is a time-consuming and error-prone task. With an intelligent detection system, large-scale vehicle monitoring can be performed efficiently, reducing human intervention and ensuring accurate data collection. The system automates **traffic management**, **vehicle counting**, and **violation detection**, making transportation networks more efficient.

1.1.2 System Context

This section outlines the environment and interactions of the **Vehicle Detection System** with external entities. The system operates by processing video footage from cameras or sensors installed on roads, highways, or restricted areas. It then identifies and classifies vehicles based on predefined categories such as **cars, buses, trucks, and motorcycles**. The system can be integrated with **law enforcement agencies**, **traffic management authorities**, and **urban planners** to optimize transportation infrastructure. The detection process is automated, ensuring **real-time monitoring**, **data analysis**, and **security enforcement** without manual intervention.
By implementing an advanced vehicle detection system, we aim to **enhance road safety, reduce congestion, and enable smart city development** through AI-driven solutions.

1.2 Objectives:

☐ Develop a Python-based vehicle detection system using computer vision and machine learning.
☐ Provide an interface for real-time vehicle identification and tracking.
☐ Maintain a database of detected vehicles, including type, count, and timestamps.
☐ Analyse traffic patterns and classify vehicles based on size and type.
☐ Ensure smooth interaction between the detection system, traffic management authorities, and security agencies.

1.3 PURPOSE:

The purpose of this application is to detect and track vehicles accurately using computer vision and deep learning techniques. The system records detected vehicles and categorized them based on type, ensuring efficient traffic management and surveillance. This technology is beneficial for monitoring road congestion, enforcing traffic laws, and enhancing transportation safety.

1.4 SCOPE:

The scope of this project is to develop a vehicle detection system capable of identifying and classifying vehicles in real time. The system shall use Python and deep learning models to
detect vehicle movement and analyze traffic patterns. This application idea was initially
suggested by other students and further refined after discussions with faculty members. The project can be expanded for smart city applications, automated toll collection, and security surveillance.

CHAPTER 2:

DESIGN

2.1 SYSTEM DESIGN:

Systems design is simply the design of systems. It implies a systematic and rigorous approach to design—an approach demanded by the scale and complexity of many systems problems. The purpose of System Design is to create a technical solution that satisfies the functional requirements for the system. At this point in the project lifecycle there should be a Functional Specification , containing a complete description of the operational needs of the various organizational entities that will use the new system.

The challenge is to translate all of this information into Technical Specifications that accurately describe the design of the system, and that can be used as input to System Construction.

The Functional Specification produced during System Requirements Analysis is transformed into a physical architecture. System components are distributed across the physical architecture, usable interfaces are designed and prototyped, and Technical Specifications are created for the Application Developers, enabling them to build and test the system.

2.2 EXISTING SYSTEM:

After reviewing the application, We now have to identify all the features that we would implement. we have gained a lot of knowledge from our research; despite being an enthusiastic real time application, we still managed to find games that we had not previously heard of such as Cab Booking System.

All the application we tested were final products and due to the timescale, We will not be able to achieve a project that is completely finished and can be put out to the general public to use without any hiccups. From all the applications we tested, we identified the following core functions which we will implement:

2.3 Attractive user interface:

The interfaces of the games tested are very appealing; each one also has a heads up display showing various in-game parameters, such as the time currently elapsed.

CHAPTER 3:

ANALYSIS

3.1 FEASIBILITY STUDY:

A Feasibility study is undertaken to determine the possibility or portability of either improving the existing system or developing a completely new system. Feasibility is the measure of how beneficial or practical the development of information system will be to an organization. The feasibility study involves following criteria.

- Whether the identified user needs may be satisfied using current software and hardware technologies.
- The study will decide if the proposed system will be cost-effective and if it can be developed given existing budgetary constraints.
- The result should inform the decision of whether to go ahead with a more detailed analysis.

There are three methods of feasibility study.
1. Technical feasibility
2. Economic feasibility
3. Observational feasibility

3.1.1 Technical feasibility:

It is a measure of the practicality of specific technical solutions and the availability of technical resources and expertise. Technical feasibility is computer oriented. The "Profile Manager", "Flappy Bird" and "Nightmare" and is technical feasible because of the following reasons:

- In Game hardware and software requirement are easily available.
- The games have a good GUI interface.
- The games will have user friendly form and screen.

3.1.2 Economic feasibility:

It is a measure of the cost-effectiveness of a project or solution. This is often called a cost-benefit analysis. Economic feasibility deals with the cost and benefits of the information system.

In economic feasibility, the development cost of the system is evaluated weighing it against the ultimate benefit derived from the new system. It is found that the benefit from the new system would be more than the cost and time involved in its development.

The Game is Economical feasible because of the following reasons.

1. Game requires less time to react for the user.
2. The cost of the hardware and software are normal.
3. GUI interface.

The system provides services for decision making. As this does not begin a conversion of the present module into and rather begins creating a new module from scratch, the cost of the module includes cost of the module development; implementation and it not include the maintenance.

3.1.3 Operational Feasibility:

Operational feasibility covers two aspects. One is the technical performance aspect and other is the acceptance. Operational feasibility determines how the proposed system will fit in the current operations and what, if any job restructuring and retraining may be needed to implement the system.

In the system operational feasibility checks, whether the user who is going to use the system can work with the software's with which the system is coded and also the mind of the user going to use the system.

If the user does not understand or is able to work on the system further development is of waste.

3.2 PROBLEM SPECIFICATION:

The definition of our problem lies in manual system and a fully automated system.

3.3 Manual System:

The system is very time-consuming and lazy. This system is more prone to error and sometimes the approach to various problems is unstructured.

3.4 Technical System:

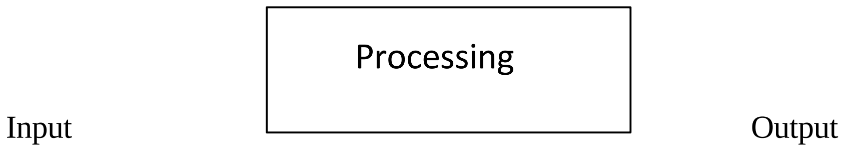
With the advent of the latest technology if we do not update our system then our business results in losses gradually with time. The technical system contains the tools of the latest trend i.e. computers, printers, FAX, Internet etc. The system with the technology are very fast, accurate, user friendly and reliable.

CHAPTER 4:

SYSTEM ARCHITECTURE

4.1 DEFINING A SYSTEM:

Collection of components, which are interconnected, and work together to realize some objective, from a system. There are three components in every system, namely input, processing and output.



4.2 SYSTEM DEVELOPMENT LIFE CYCLE

The Systems development life cycle (SDLC), or Software development process in systems engineering, information systems and software engineering, is a process of creating or altering information systems, and the models and methodologies that people use to develop these systems. In software engineering, the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system in the software development process.

Broadly, the following are the different activities to be considered while defining the system development life cycle for the said project:

- Problem Definition
- System Analysis
- Study of existing system
- Drawback of the existing system
- System Requirement study
- Data flow analysis
- Feasibility study
- System design

- Input Design (Database & Forms)
- Updating
- Report design
- Administration
- Testing
- Implementation
- Maintenance
- SYSTEM

4.3 SYSTEM ANALYSIS:

4.3.1 Functional Requirements

- **User Registration**: Users must be able to register with their details like name, phone number, and location.
- **Cab Booking**: Users can search for available cabs, book rides, and get details about the driver.
- **Fare Calculation**: The system calculates fare based on the distance between the pickup and drop-off locations.
- **Ride History**: Users can view their ride history (completed rides).
- **Admin Panel**: Admins can manage drivers, view all bookings, and update car availability.

4.3.2 Non-functional Requirements

- **Usability**: The system should be easy to use with a simple interface.
- **Performance**: It should handle multiple bookings efficiently without delay.
- **Reliability**: The system must ensure data integrity, such as avoiding duplicate bookings for the same car.

CHAPTER 5:

ER DIAGRAMS

5.1 Data Flow Diagram:

1) $\boxed{\text{External entity}}$

which defines an entity. A producer or consumer of information that resides outside the bounds of the system to be modelled.

2) Arrow,

which shows dataflow. A data object: the arrowhead indicates the direction of data flow.

3) Circle

which represent a process that transforms incoming data into outgoing flow. A transformation of information (a function) that resides within the bounds of the system to be modelled.

4) Open rectangle,

which shows a data store. A repository of data that is to be stored for use by one or more processes may be as simple as a buffer or queue or as sophisticated as a relational database. Process.

LEVEL 0 DFD:

This DFD is explaining the work of User. Firstly, click on Play button to enter the App. After Click on Play button user able to proceed to further.

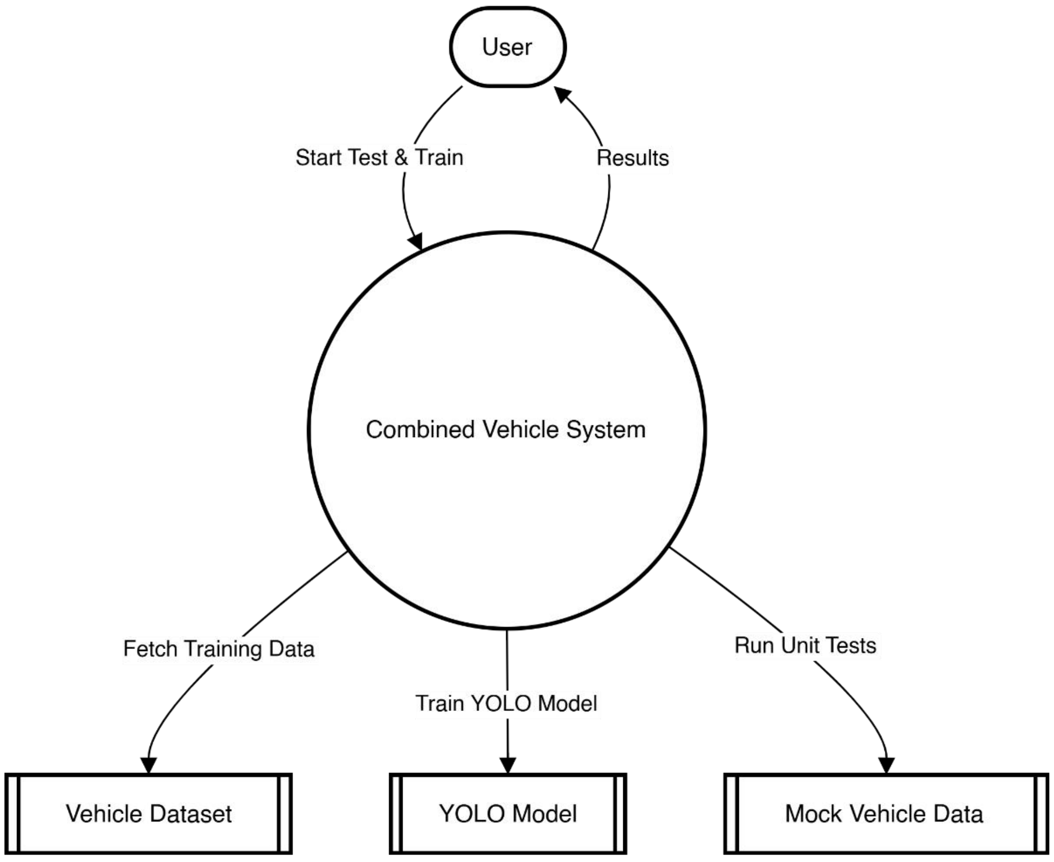


Fig.5.1.1

LEVEL 1 DFD:

This DFD is showing the working of Users. After proceed from Level 0, Now user has to play the application and increase automatically when you see the speed of block is increased.

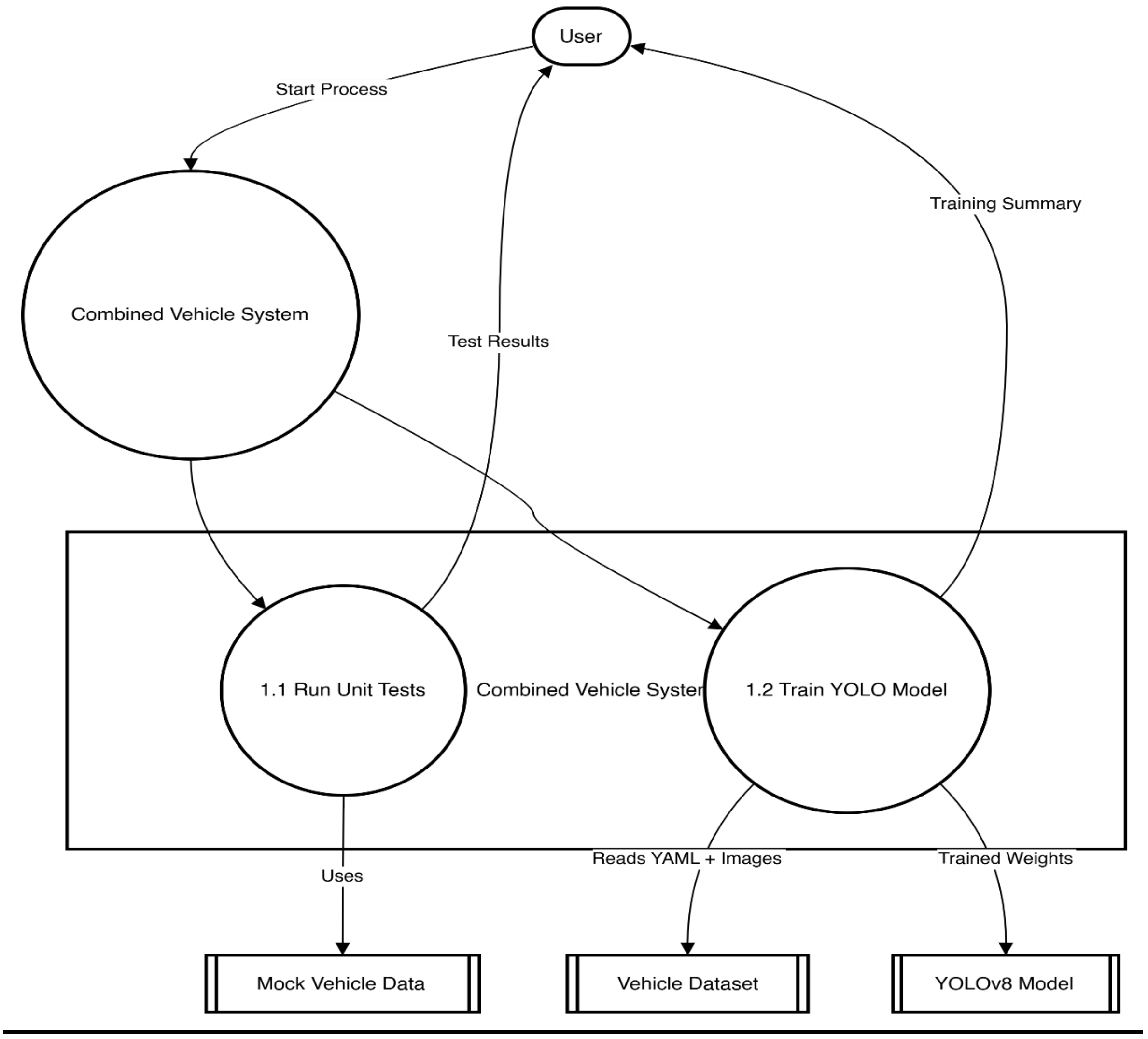


Fig.5.1.2

CHAPTER 6:

TESTING:

6.1 FUNCTIONAL TESTING:

Function testing verifies that each function of the software application operates in conformance with the requirement specification. The testing mainly involves black box testing and it is not concerned with the source code. The functionality of the system by providing appropriate input, verifying the output and comparing the actual result with expected result.

Functional testing can be performed on survival of the fastest to check that the appropriate output is coming from the input provided. Each Activity in Survival of the fastest includes validation in which the next activity is depend upon the user choice.

6.2 STRUCTURAL TESTING:

Structural testing is the testing of the structure of the system or component. Structural testing is often referred to as 'white box' or 'glass box' or 'clear box testing' because in structural testing, testing is done to check what is happening inside the system. In structural testing testers are required to have the knowledge of the internal implementations of the code. Here the testers require knowledge of how the software is implemented, and how it works.

In Survival the fastest structural testing is done to check how the loops in the software are working. Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software. Structural testing checks whether the code written is correct or not. The code in tour and travel is written correctly and is error free. It checks that the menu option is working properly or not. And all the menu Items including the correct link to the correct from.

6.3 LEVEL OF TESTING:

There are different levels during the process of Testing. Levels of testing include the different methodologies that can be used while conducting Software Testing. The following are the main levels of Software Testing:

- Functional Testing.
- Non-Functional Testing.

6.3.1 Functional Testing

Functional testing can be performed on survival of the fastest to check that the appropriate output is coming from the input provided.

i. Unit Testing

Is this testing is done by the developers on the individual units of source code assigned area. The developers use test data that is separate from the test data of the quality assurance team. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

In this we have tested each and every module separately just to know that they are working properly and giving the desired output.

ii. Integration Testing

The testing of combined parts of an application to determine if they function correctly together is Integration testing.

In this phase of testing we are emphasizing on how two or more applications or functions are working with each other . like how are they activities are depend upon each other, which activity is come under which module.

iii. System Testing

This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards.

This type of testing is performed by a specialized testing team. System

testing is so important because of the following reasons:

•System Testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.

•The application is tested thoroughly to verify that it meets the functional and technical specifications.

•The application is tested in an environment which is very close to the production environment where the application will be deployed.

•System Testing enables us to test, verify and validate both the business requirements as well as the Applications Architecture.

iv. Alpha Testing

This test is the first stage of testing and will be performed amongst the teams (developer and QA teams). Unit testing, integration testing and system testing when combined are known as alpha testing. During this phase, the following will be tested in the application:

- Spelling Mistakes
- Broken Links
- Cloudy Directions
- The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

v. **Beta Testing**

This test is performed after Alpha testing has been successfully performed. In beta testing a sample of the intended audience tests the application. Beta testing is also known as pre-release testing. Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program a "real-world" test and partly to provide a preview of the next release. In this phase the audience will be testing the following:

- Users will install, run the application and send their feedback to the project team. • Typographical errors, confusing application flow, and even crashes.

- Getting the feedback, the project team can fix the problems before releasing the software to the actual users.

- The more issues you fix that solve real user problems, the higher the quality of your application will be.

- Having a higher-quality application when you release to the general public will increase customer satisfaction.

6.3.2 <u>Non-Functional Testing:</u>

This section is based upon the testing of the application from its non-functional attributes. Nonfunctional testing of Software involves testing the Software from the requirements which are non functional in nature related but important a well such as performance, security, user interface etc. Some of the important and commonly used non-functional testing types are mentioned as follows:

i. Performance Testing

It is mostly used to identify any bottlenecks or performance issues rather than finding the bugs in software.

There are different causes which contribute in lowering the performance of software:

- Network delay.
- Client side processing.
- Database transaction processing. Load balancing between servers.
- Data rendering.
- Performance testing is considered as one of the important and mandatory testing type in terms of following aspects:
- Speed (i.e. Response Time, data rendering and accessing)
- Capacity
- Stability
- Scalability

CHAPTER 7:

SYSTEM REQUIREMENT:

7.1 Introduction

System requirements are the configurations that a system must have in order for a hardware or software application to run smoothly and efficiently. Failure to meet these requirements can result in installation problems or performance problems. The former may prevent a device or application from getting installed, whereas the latter may cause a product to malfunction or perform below expectation or even to hang or crash. System requirements are also known as minimum system requirements.

7.2 Hardware Requirements

The hardware which are used in the development are as follows hardware specifications:-

S.no.	Hardware	Specifications
1.	Operating System	32/64 bit
2.	RAM	4GB or more
3.	Hard Drive	500GB or more
4.	Processor	9th Generation or more advanced
5.	Motherboard	H31, H81 or more advanced model

Table7.1

7.3 <u>Software Requirements</u>

The software which are used in this system are as under in the table of software specifications:-

S.no.	Softwares	Specifications
1.	Windows	XP/7/8/8.1/10/11
2.	Python	3.9.2 or more latest version
3.	IDLE Shell	Python 3.9.2 or more latest version

Table 7.2

7.4 <u>Library Requirements</u>

There are 5 libraries used in this system are as under in the table below:-

S.no.	**Libraries**
1.	Tensorflow/Keras
2.	PyTorch
3.	Yolov8
4.	OpenCV
5.	Pillow (PIL)
6.	Albumentations

7.5 <u>Python</u>

- Python is a widely used general-purpose, high level programming language.
- It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation.
- It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.
- Python is a programming language that lets you work quickly and integrate systems more efficiently.
- It is used for:
    - web development (server-side),
    - software development,
    - mathematics,
    - system scripting

7.6 <u>About Libraries</u>

The "Python library" contains several different kinds of components. It contains data types that would normally be considered part of the "core" of a language, such as numbers and lists. For these types, the Python language core defines the form of literals and places some constraints on their semantics, but does not fully define the semantics. The library also contains built-in functions and exceptions — objects that can be used by all Python code without the need of an import statement. Some of these libraries are defined below which are used in this system below:-

7.6.1 Tensorflow

:- TensorFlow is an open-source framework for machine learning (ML) and artificial intelligence (AI) that was developed by Google Brain. It was designed to facilitate the development of machine learning models, particularly deep learning models, by providing tools to easily build, train, and deploy them across different platforms.
TensorFlow supports a wide range of applications, from natural language processing (NLP) and computer vision (CV) to time series forecasting and reinforcement learning.

7.6.2 PyTorch

:- PyTorch is a deep learning library built on Python and Torch (a Lua-based framework). It provides GPU acceleration, dynamic computation graphs, and an intuitive interface for deep learning researchers and developers. PyTorch follows a "define-by-run" approach, meaning that its computational graphs are constructed on the fly, allowing for better debugging and model customization.

7.6.3 Yolov8

:- YOLOv8 is a computer vision model architecture developed by Ultralytics, the creators of YOLOv5. You can deploy YOLOv8 models on a wide range of devices, including NVIDIA Jetson, NVIDIA GPUs, and macOS systems with Roboflow Inference, an open source Python package for running vision models.

7.6.4 OpenCv

:- OpenCV, short for Open Source Computer Vision Library, is an open-source computer vision and machine learning software library. Originally developed by Intel, it is now maintained by a community of developers under the OpenCV Foundation.

7.6.5 Pillow(Pil)

:- Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aids in editing, creating and saving images. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and added Python3.x support to it. Pillow was announced as a replacement for PIL for future usage. Pillow supports a large number of image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders.

7.6.6 Albumentations

:- Albumentations is an open-source image augmentation library created in June 2018 by a group of researchers and engineers, including Alexander Buslaev, Vladimir Iglovikov, and Alex Parinov. The library was designed to provide a flexible and efficient framework for data augmentation in computer vision tasks.

7.7 <u>About IDLE Shell(Python)</u>

IDLE is Python's Integrated Development and Learning Environment. IDLE has the following features:

- coded in 100% pure Python, using the tkinter GUI toolkit
- cross-platform: works mostly the same on Windows, Unix, and macOS
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features
- search within any window, replace within editor windows, and search through multiple files (grep)
- debugger with persistent breakpoints, stepping, and viewing of global and local namespaces
- configuration, browsers, and other dialogs

CHAPTER 8
Algorithm

8.1 Algorithm for Vehicle Detection System:-

1. **Setup Phase**
   - **Prepare test vehicle data (mock JSON)**
   - **Configure YOLO training paths**

```
test_data = {
"vehicles": [
{"id": 1, "type": "car", "coords": [x,y]},
{"id": 2, "type": "truck", "coords": [x,y]}
]
}

# Fix YOLO dataset paths
shutil.copy('input/data.yaml', 'working/data.yaml')
```

2. **Test Vehicle Logic**
   - **Verify data loading works**
   - **Test filtering by vehicle type**
   - **Check selection handling**

```
def test_vehicle_filtering():
view_model.set_vehicle_type("car")
assert len(view_model.vehicles) == 3  # Expect 3 cars
```

3. **Train Detection Model**
   - **Load YOLO base model**
   - **Train on vehicle dataset**

```
model = YOLO('yolov8m.pt')
model.train(data='working/data.yaml', epochs=10)
```

4. **Integration**
   - **Connect detection results to view model**

```
def update_vehicles(detections):
for det in detections:
view_model.add_vehicle(det)
```

8.2 Code:-

```
import shutil

from ultralytics import YOLO

import unittest

from unittest.mock import Mock, MagicMock

import json

from corelocation import CLLocationCoordinate2D


class CombinedVehicleSystem:

    def __init__(self):

        # Initialize both testing and training components

        self.setup_test_environment()

        self.setup_training_environment()


    def setup_test_environment(self):

        """Prepare the vehicle view model test environment"""

        # This would be the Swift test environment setup

        # For Python demonstration, we'll create mock equivalents

        self.mock_vehicle_data = {

            "poiList": [

                {

                    "id": 405818,

                    "coordinate": {"latitude": 53.520445, "longitude": 9.768903},

                    "fleetType": "POOLING",

                    "heading": 92.996

                },

                # ... other test vehicles

            ]
```

```
  def setup_training_environment(self):
      """Prepare the YOLO training environment"""

# Copy and modify YAML config
      yaml_input = '/kaggle/input/vehicle-dataset/data.yaml'
      yaml_output = '/kaggle/working/data.yaml'
      shutil.copy(yaml_input, yaml_output)

      # Fix paths in YAML
      with open(yaml_output, 'r') as f:
          yaml_data = f.read()

      yaml_data = yaml_data.replace('/content/vehicle_dataset/train/images',
                                    '/kaggle/input/vehicle-dataset/train/images')
      yaml_data = yaml_data.replace('/content/vehicle_dataset/val/images',
                                   '/kaggle/input/vehicle-dataset/val/images')

      with open(yaml_output, 'w') as f:
          f.write(yaml_data)

      self.training_config = yaml_output

  def run_unit_tests(self):
      """Execute the vehicle view model tests"""
      # Python equivalent of the XCTest cases
      test_suite = unittest.TestLoader().loadTestsFromTestCase(VehicleViewModelTests)
      unittest.TextTestRunner().run(test_suite)

  def train_detection_model(self):
        """Train the YOLO vehicle detection model"""
        model = YOLO('yolov8m.pt')
        results = model.train(
            data=self.training_config,
            epochs=10,
            imgsz=640,
            batch=16,
            workers=2,
            device=0,
            name='vehicle-detection',
            save=True,
            val=True
        )

        return results

class VehicleViewModelTests(unittest.TestCase):
    """Python version of the XCTest cases for demonstration"""

    def setUp(self):
        self.mock_data = {
            "poiList": [
                {
                    "id": 405818,
                    "coordinate": {"latitude": 53.520445, "longitude": 9.768903},
                    "fleetType": "POOLING",
                    "heading": 92.996
                }
            ]
        }
        self.mock_service = MockVehicleService(self.mock_data)
        self.view_model = VehicleViewModel(self.mock_service)

    def test_vehicle_loading(self):
        self.view_model.fetch_vehicles()
        self.mock_service.fetch_success()
        self.assertEqual(len(self.view_model.vehicles), 1)

self.assertEqual(self.view_model.vehicles[0].id, 405818)

class MockVehicleService:
    """Mock service for testing"""
    def __init__(self, data):
        self.data = data
        self.fetch_called = False

    def fetch_vehicles(self):
        self.fetch_called = True

    def fetch_success(self):
        self.view_model.handle_success(self.data)

    def fetch_fail(self):
        self.view_model.handle_failure("Error")

class VehicleViewModel:
    """Simplified view model for demonstration"""
    def __init__(self, service):
        self.service = service
        self.vehicles = []
        service.view_model = self

    def fetch_vehicles(self):
        self.service.fetch_vehicles()

    def handle_success(self, data):
        self.vehicles = data["poiList"]

    def handle_failure(self, error):

  self.error = error

# Main execution
if __name__ == "__main__":
    system = CombinedVehicleSystem()

    print("Running unit tests...")
    system.run_unit_tests()

    print("\nStarting model training...")
    training_results = system.train_detection_model()
    print("Training completed!")

    # Potential integration point:
    # Could connect the trained YOLO model to the view model here
    # for real-time vehicle detection and display
```

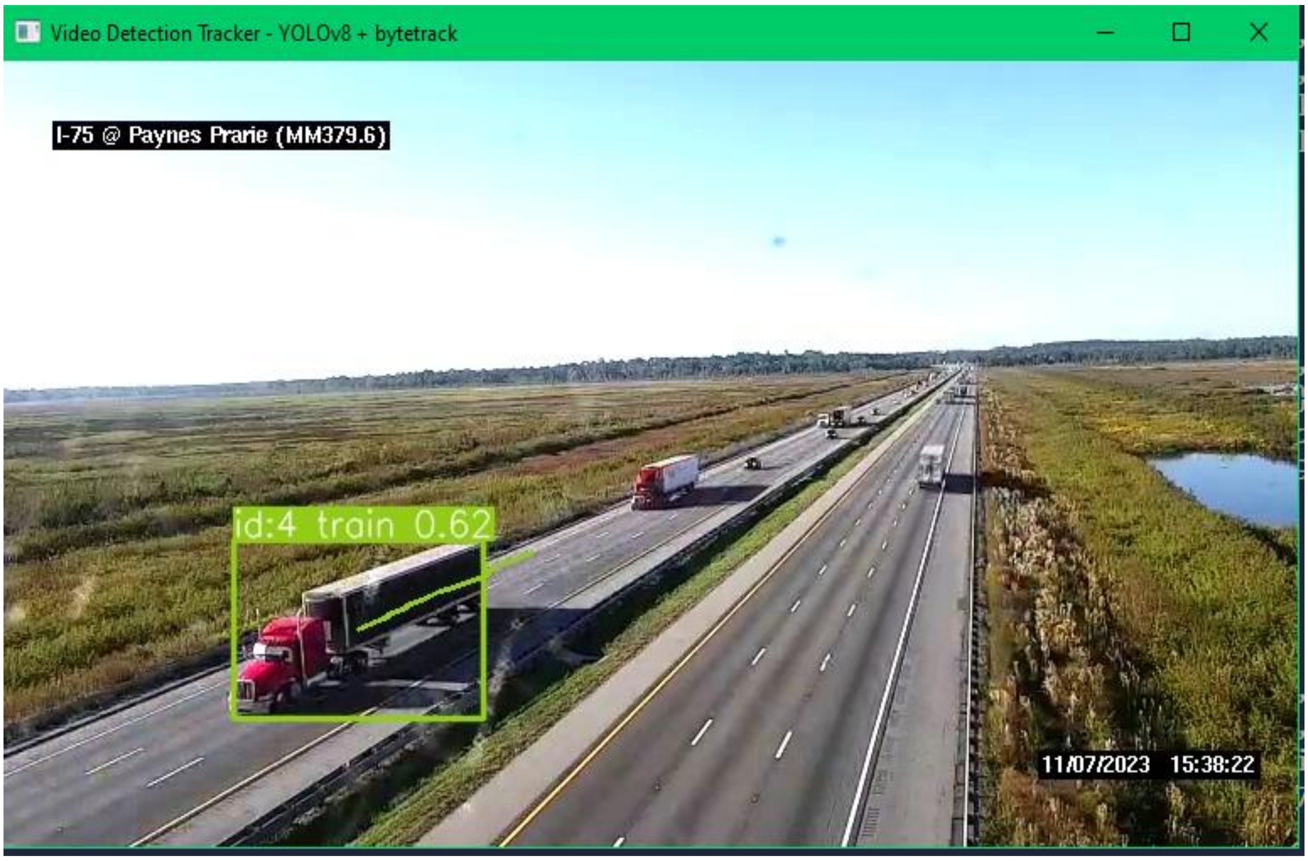
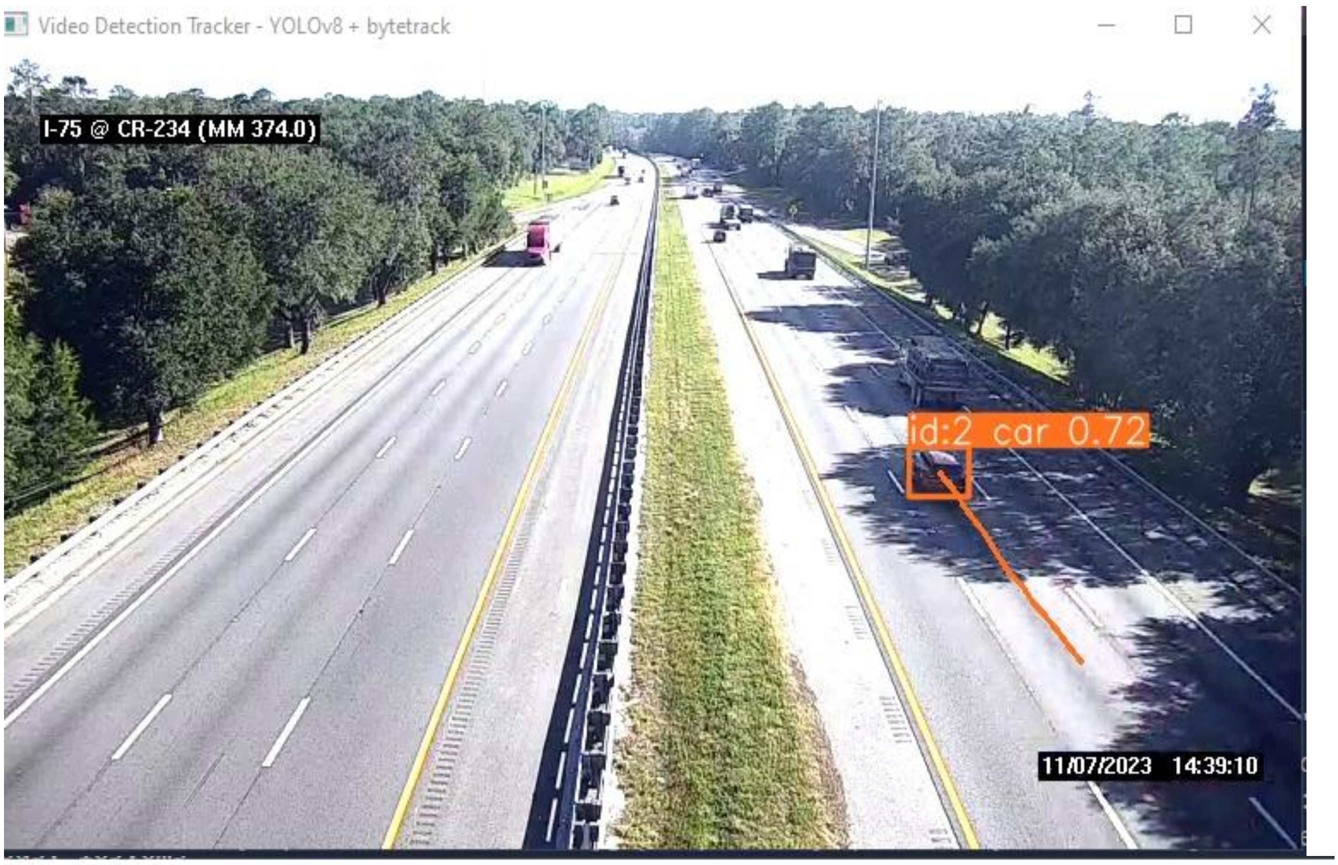

Video Detection Tracker - YOLOv8 + bytetrack
I-75 @ Paynes Prarie (MM379.6)
id:4 train 0.62
11/07/2023 15:38:22

Fig. 8.2.1


Video Detection Tracker - YOLOv8 + bytetrack
I-75 @ CR-234 (MM 374.0)
id:2 car 0.72
11/07/2023 14:39:10

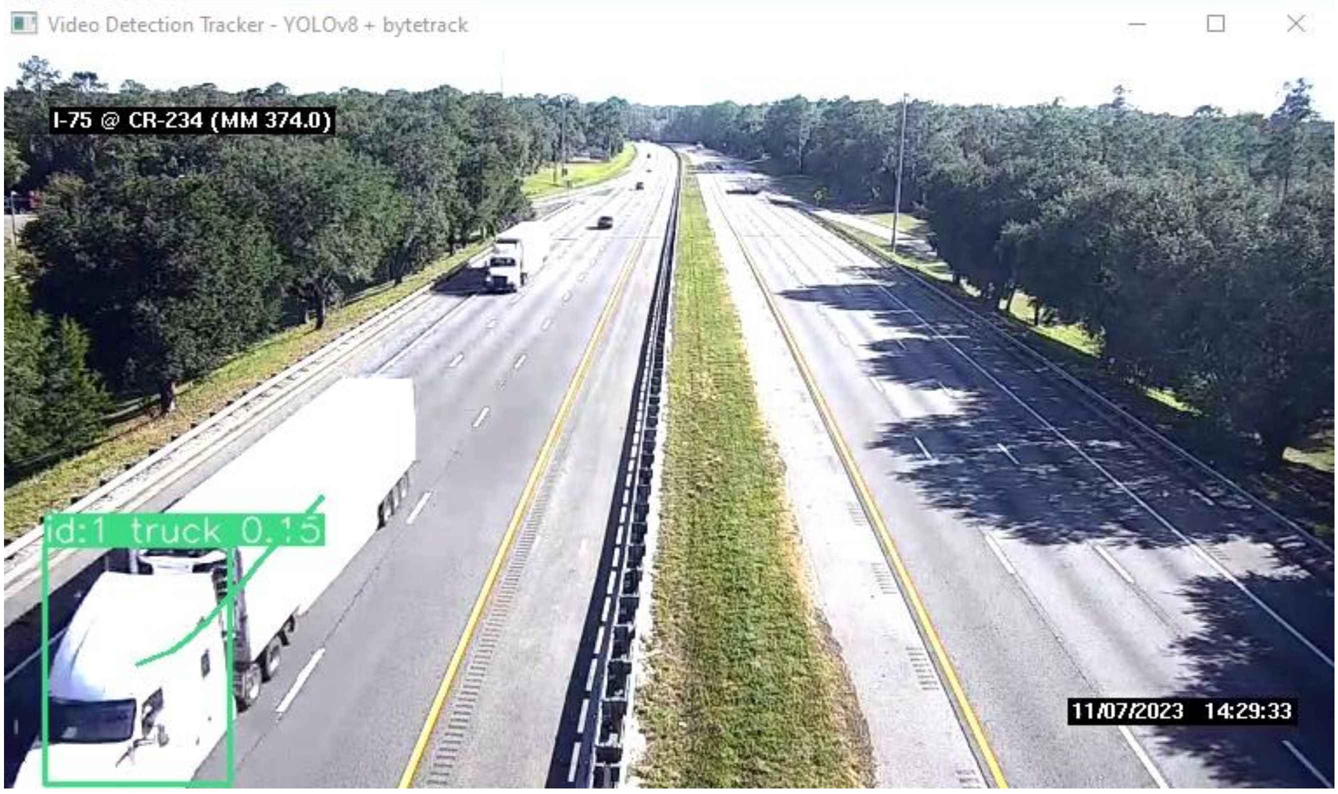
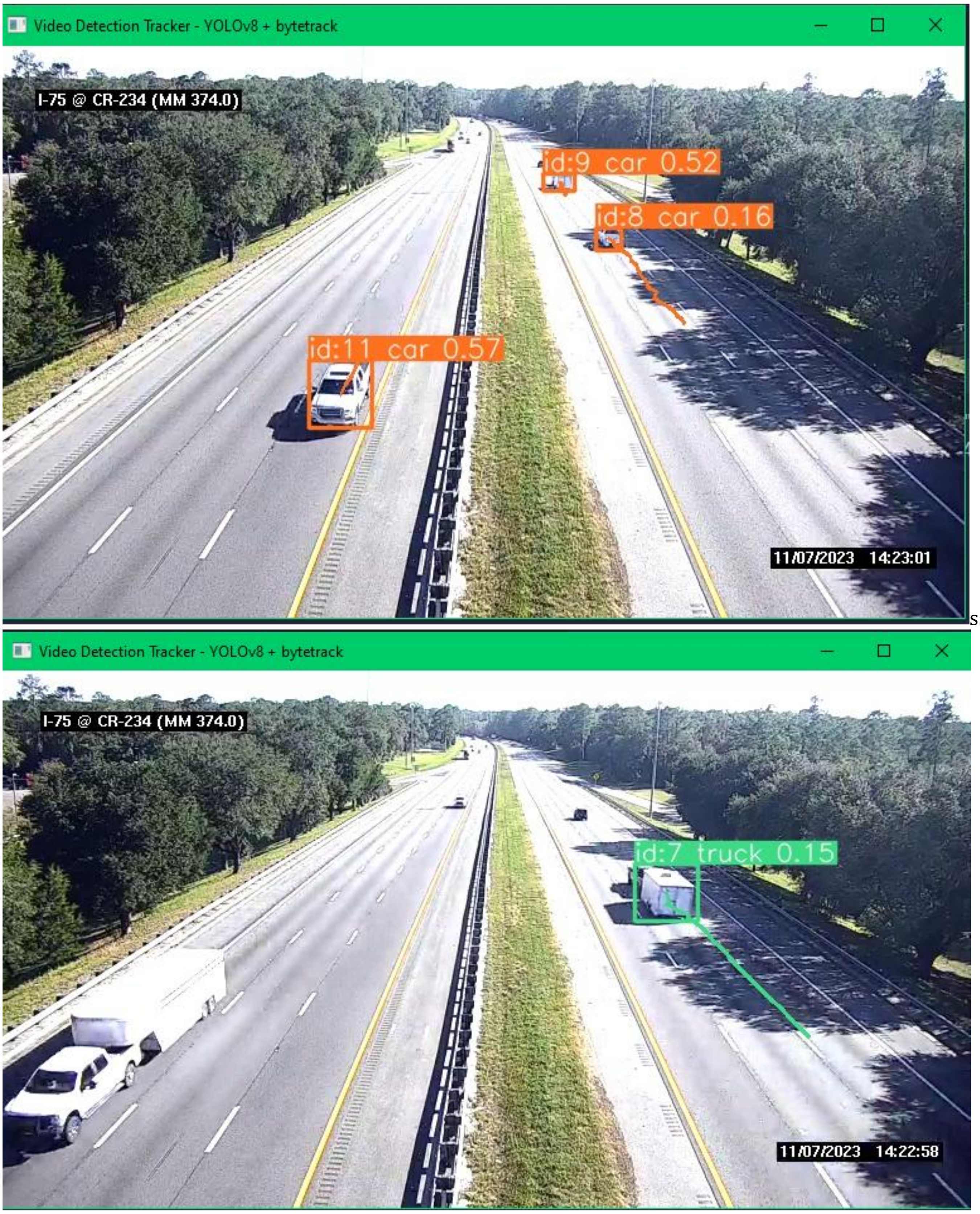
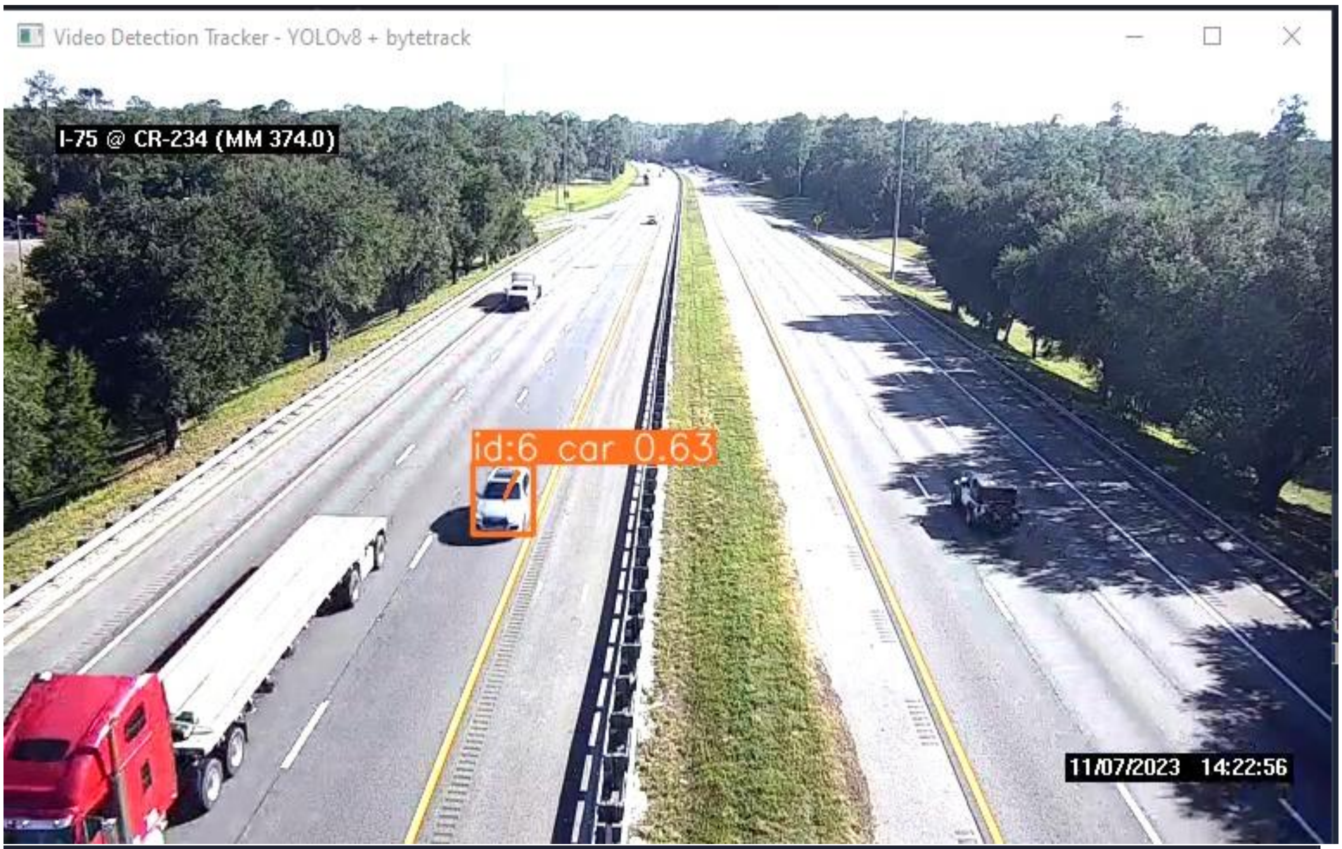

Video Detection Tracker - YOLOv8 + bytetrack
I-75 @ CR-234 (MM 374.0)
id:1 truck 0.15
11/07/2023 14:29:33

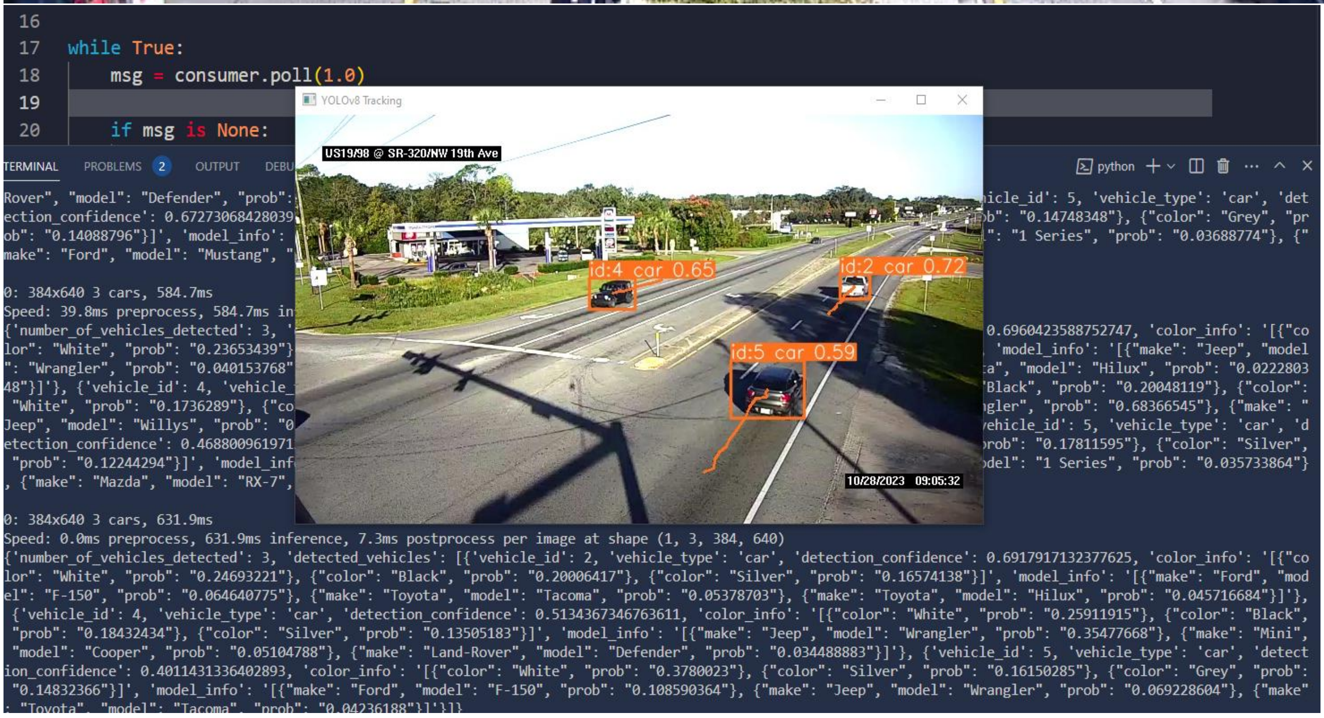
Fig. 8.2.2


Video Detection Tracker - YOLOv8 + bytetrack
I-75 @ CR-234 (MM 374.0)
id:9 car 0.52
id:8 car 0.16
id:11 car 0.57
11/07/2023 14:23:01
Video Detection Tracker - YOLOv8 + bytetrack
I-75 @ CR-234 (MM 374.0)
id:7 truck 0.15
11/07/2023 14:22:58

ss

Fig.8.2.3

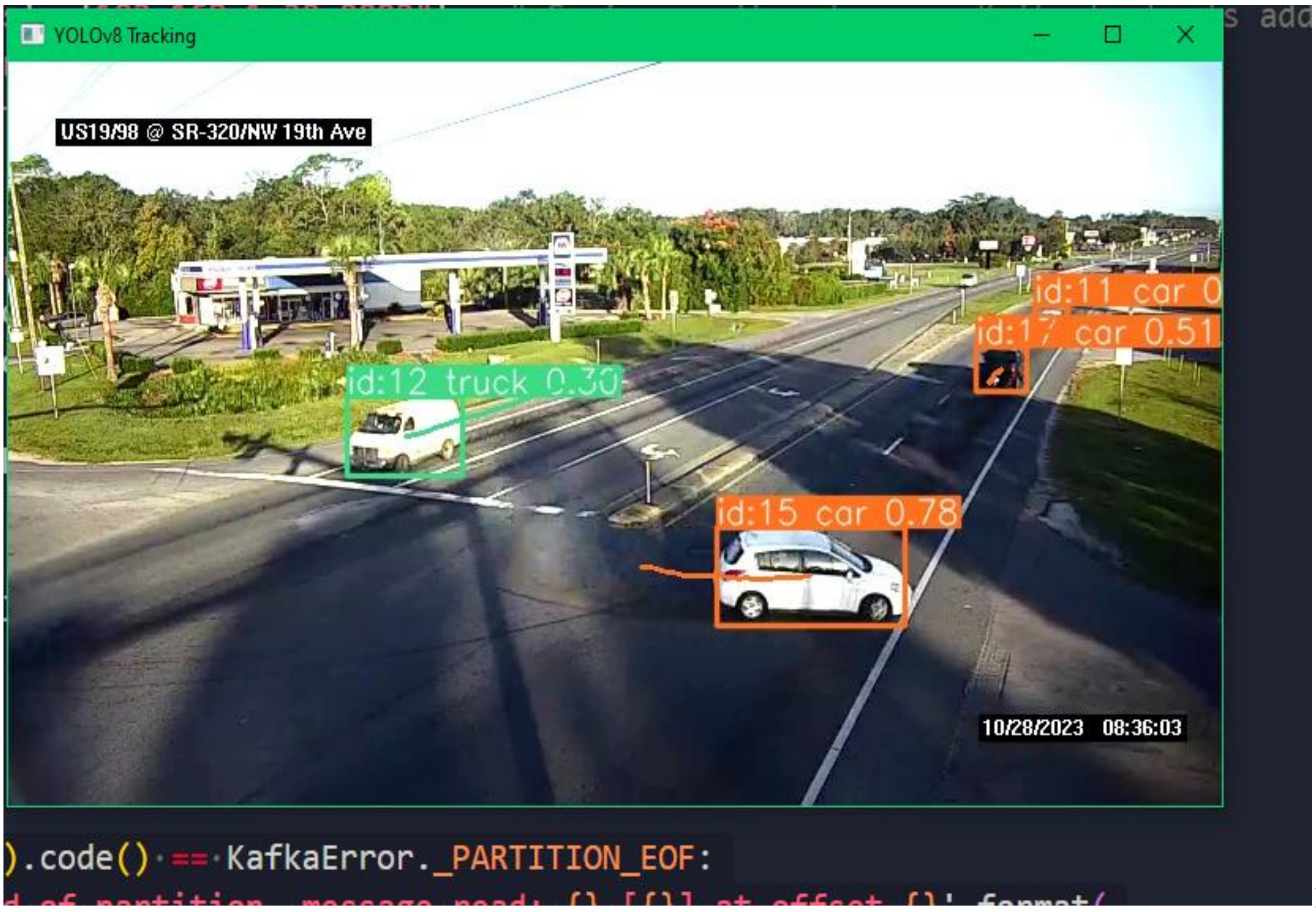




```

while True:
    msg = consumer.poll(1.0)

    if msg is None:
```

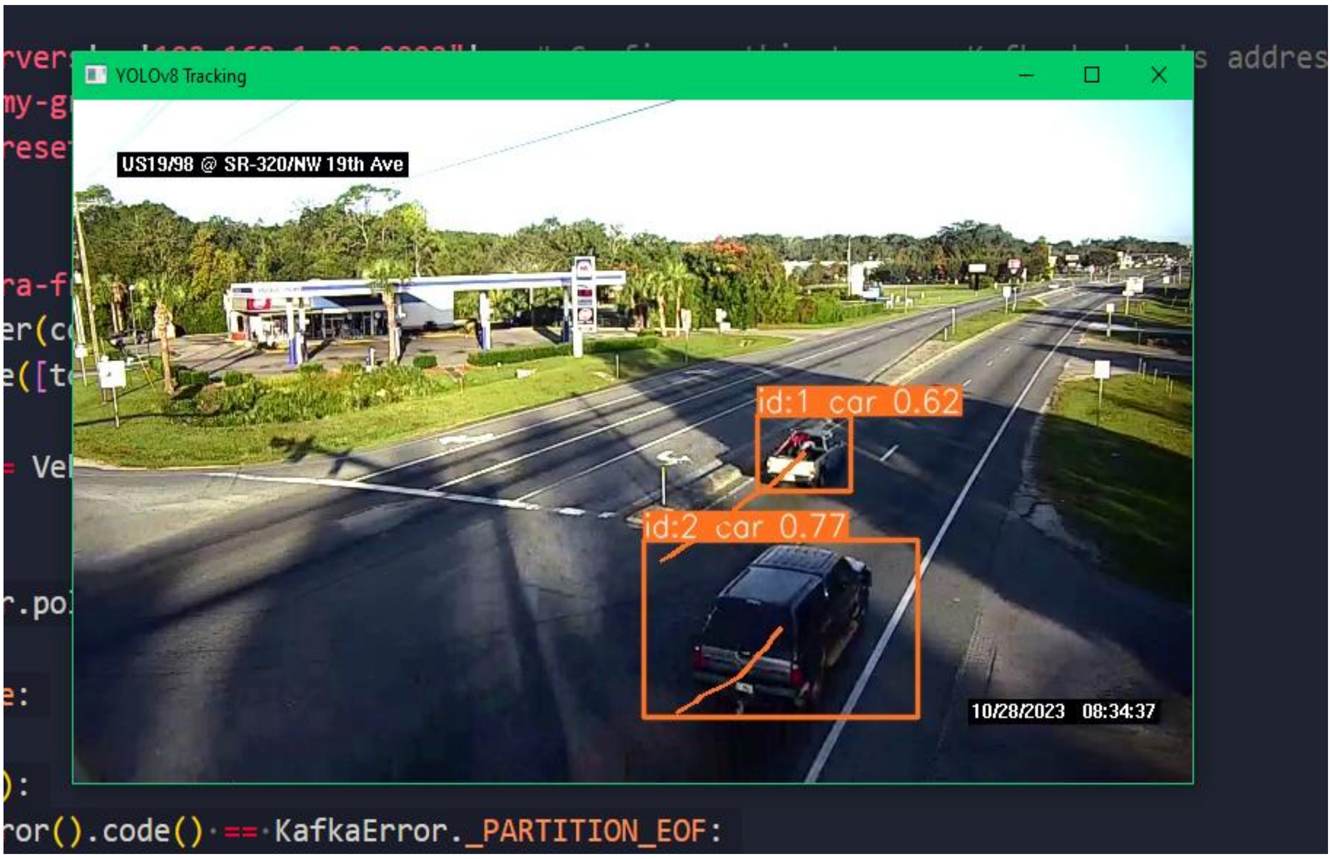
```
Rover", "model": "Defender", "prob":                                                   icle_id': 5, 'vehicle_type': 'car', 'det
ection_confidence': 0.67273068428039                                                  ob": "0.14748348"}, {"color": "Grey", "pr
ob": "0.14088796"}]', 'model_info':                                                   ": "1 Series", "prob": "0.03688774"}, {"
make": "Ford", "model": "Mustang", "

0: 384x640 3 cars, 584.7ms
Speed: 39.8ms preprocess, 584.7ms in
{'number_of_vehicles_detected': 3, '                                                  0.6960423588752747, 'color_info': '[{"co
lor": "White", "prob": "0.23653439"}                                                   'model_info': '[{"make": "Jeep", "model
": "Wrangler", "prob": "0.040153768"                                                  a", "model": "Hilux", "prob": "0.0222803
48"}]'}, {'vehicle_id': 4, 'vehicle_                                                  Black", "prob": "0.20048119"}, {"color":
 "White", "prob": "0.1736289"}, {"co                                                  gler", "prob": "0.68366545"}, {"make": "
Jeep", "model": "Willys", "prob": "0                                                  ehicle_id': 5, 'vehicle_type': 'car', 'd
etection_confidence': 0.468800961971                                                  rob": "0.17811595"}, {"color": "Silver",
 "prob": "0.12244294"}]', 'model_inf                                                  del": "1 Series", "prob": "0.035733864"}
, {"make": "Mazda", "model": "RX-7",

0: 384x640 3 cars, 631.9ms
Speed: 0.0ms preprocess, 631.9ms inference, 7.3ms postprocess per image at shape (1, 3, 384, 640)
{'number_of_vehicles_detected': 3, 'detected_vehicles': [{'vehicle_id': 2, 'vehicle_type': 'car', 'detection_confidence': 0.6917917132377625, 'color_info': '[{"co
lor": "White", "prob": "0.24693221"}, {"color": "Black", "prob": "0.20006417"}, {"color": "Silver", "prob": "0.16574138"}]', 'model_info': '[{"make": "Ford", "mod
el": "F-150", "prob": "0.064640775"}, {"make": "Toyota", "model": "Tacoma", "prob": "0.05378703"}, {"make": "Toyota", "model": "Hilux", "prob": "0.045716684"}]'},
 {'vehicle_id': 4, 'vehicle_type': 'car', 'detection_confidence': 0.5134367346763611, 'color_info': '[{"color": "White", "prob": "0.25911915"}, {"color": "Black",
 "prob": "0.18432434"}, {"color": "Silver", "prob": "0.13505183"}]', 'model_info': '[{"make": "Jeep", "model": "Wrangler", "prob": "0.35477668"}, {"make": "Mini",
 "model": "Cooper", "prob": "0.05104788"}, {"make": "Land-Rover", "model": "Defender", "prob": "0.034488883"}]'}, {'vehicle_id': 5, 'vehicle_type': 'car', 'detect
ion_confidence': 0.4011431336402893, 'color_info': '[{"color": "White", "prob": "0.3780023"}, {"color": "Silver", "prob": "0.16150285"}, {"color": "Grey", "prob":
 "0.14832366"}]', 'model_info': '[{"make": "Ford", "model": "F-150", "prob": "0.108590364"}, {"make": "Jeep", "model": "Wrangler", "prob": "0.069228604"}, {"make"
: "Toyota", "model": "Tacoma", "prob": "0.04236188"}]'}]}
```

Fig.8.2.4



```
).code() == KafkaError._PARTITION_EOF:
```

FF



```
rver
my-g
rese

ra-f
er(c
e([t

= Ve

r.po

e:

):
ror().code() == KafkaError._PARTITION_EOF:
```

Fig.8.2.5