A Report On

OBJECT DETECTION FOR AUTONOMOUS VEHICLES

Under the Guidance (Mentor) of

MS. AISHWARYA SAXENA

Final Project Report

By

PAVNI SHRI GUPTA

PIE INFOCOMM PRIVATE LIMITED

PIE INFOCOMM

RESEARCH, TRAINING AND DEVELOPMENT

PIE INFOCOMM PRIVATE LIMITED

LUCKNOW, UTTAR PRADESH

TABLE OF CONTENT

Topics	Page No.
Introduction	**3**
Objective	**3**
Background	**4**
Hardware and Software Requirements	**4**
Coding	**4**
Output Screenshot	**13**
Future Scope	**15**
Conclusion	**15**
References and Bibliography	**16**

OBJECT DETECTION FOR AUTONOMOUS VEHICLES

Introduction

Autonomous vehicles are transforming the transportation sector by utilizing artificial intelligence and computer vision to operate independently of human control. A significant challenge in the realm of autonomous driving is the need for real-time object detection to identify obstacles, pedestrians, other vehicles, and traffic signals. This initiative employs an object detection system based on YOLOv5 (You Only Look Once), a deep learning model recognized for its rapid processing and accuracy. The system features a Tkinter-based graphical user interface (GUI), enabling users to upload images and videos while visualizing detected objects through bounding boxes and labels.

This initiative illustrates how object detection can improve road safety and facilitate vehicle automation by accurately identifying objects in real-time. It acts as a crucial component for advanced driver-assistance systems (ADAS) and fully autonomous vehicles.

Objective

The main objective of this project is to develop a robust real-time object detection system that can accurately identify and classify objects relevant to autonomous driving. The system focuses on:

- Identifying various types of vehicles, including cars, buses, trucks, and motorcycles.
- Recognizing pedestrians to improve safety protocols.
- Detecting traffic signals and road signs to facilitate autonomous navigation.
- Offering an interactive graphical interface that allows users to upload files and examine results.

This initiative seeks to enhance the effectiveness and precision of object detection, thereby advancing the progress of autonomous vehicle technologies.

Background

Object detection plays a vital role in the fields of computer vision and deep learning, allowing machines to understand visual information. In the realm of autonomous vehicles, the ability to detect and classify road elements in real-time is essential for ensuring safe navigation.

YOLOv5 (You Only Look Once) represents a cutting-edge object detection model that prioritizes both speed and precision. In contrast to conventional region-based techniques, YOLO processes the entire image in a single pass through a neural network, which enhances its efficiency for real-time applications.

For this project, the Tkinter library is utilized to develop a user-friendly graphical user interface (GUI). The integration of YOLOv5 with Tkinter offers a straightforward interface for processing images and videos, enabling users to effectively analyze real-world driving situations.

Hardware and Software Requirements

Hardware Requirements:

- Processor: Intel Core i5/i7 or equivalent (for smooth performance)
- RAM: Minimum 8GB (16GB recommended for large datasets)
- GPU: NVIDIA GPU (Recommended for faster deep learning inference)
- Storage: At least 10GB of free disk space

Software Requirements:

- Operating System: Windows 10/11, Linux, or macOS
- Programming Language: Python 3.x
- Libraries & Dependencies:
    - Torch & Torchvision: Required for deep learning model inference
    - OpenCV: Used for image and video processing
    - Pillow: Handles image processing in Tkinter
    - Tkinter: Provides the graphical user interface
    - YOLOv5 (Ultralytics): Pre-trained model for object detection
- Development Environment: VS Code, PyCharm, or Jupyter Notebook

Coding

The project consists of four key Python scripts:

- main.py:
    - Acts as the gateway for the project.
    - Initiates and operates the Tkinter graphical user interface.

```
from gui import ObjectDetectionApp
import tkinter as tk


if __name__ == "__main__":
    root = tk.Tk()
    app = ObjectDetectionApp(root)
    root.mainloop()
```

- gui.py:
    - Oversees the Graphical User Interface (GUI).
    - Enables users to upload images and videos, initiate object detection, and present the outcomes.
    - Transfers the chosen file to detector.py for analysis.

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
from detector import ObjectDetector
class ObjectDetectionApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Object Detection for Autonomous Vehicles")
        self.root.geometry("800x600")
        self.image_label = tk.Label(self.root)
        self.image_label.pack()
        # Buttons
        self.load_button = tk.Button(self.root, text="Load Image/Video", command=self.load_file)
        self.load_button.pack(pady=10)
        self.detect_button = tk.Button(self.root, text="Run Object Detection", command=self.run_detection)
        self.detect_button.pack(pady=10)
        self.file_path = None
        # Initialize Object Detector
        self.detector = ObjectDetector()
    def load_file(self):
        file_types = [("Image/Video Files", "*.jpg *.jpeg *.png *.mp4 *.avi")]
        self.file_path = filedialog.askopenfilename(title="Select Image or Video",
                                                    filetypes=file_types)
        if self.file_path:
            if self.file_path.lower().endswith((".jpg", ".jpeg", ".png")):
                self.display_image(self.file_path)
            elif self.file_path.lower().endswith((".mp4", ".avi")):
                messagebox.showinfo("Info", "Video loaded. Object detection will run on it.")
    def display_image(self, path):
        image = Image.open(path)
        image = image.resize((600, 400))
        photo = ImageTk.PhotoImage(image)

        self.image_label.config(image=photo)
        self.image_label.image = photo

    def run_detection(self):
        if not self.file_path:
            messagebox.showerror("Error", "Please load an image or video first.")
            return
        if self.file_path.lower().endswith((".jpg", ".jpeg", ".png")):
            detected_image = self.detector.detect_image(self.file_path)
            self.display_detected_image(detected_image)
        elif self.file_path.lower().endswith((".mp4", ".avi")):
            messagebox.showinfo("Info", "Video detection is under development.")
    def display_detected_image(self, image):
        img = Image.fromarray(image)
        img = img.resize((600, 400))
        photo = ImageTk.PhotoImage(img)
        self.image_label.config(image=photo)
        self.image_label.image = photo
if __name__ == "__main__":
    root = tk.Tk()
    app = ObjectDetectionApp(root)
    root.mainloop()
```

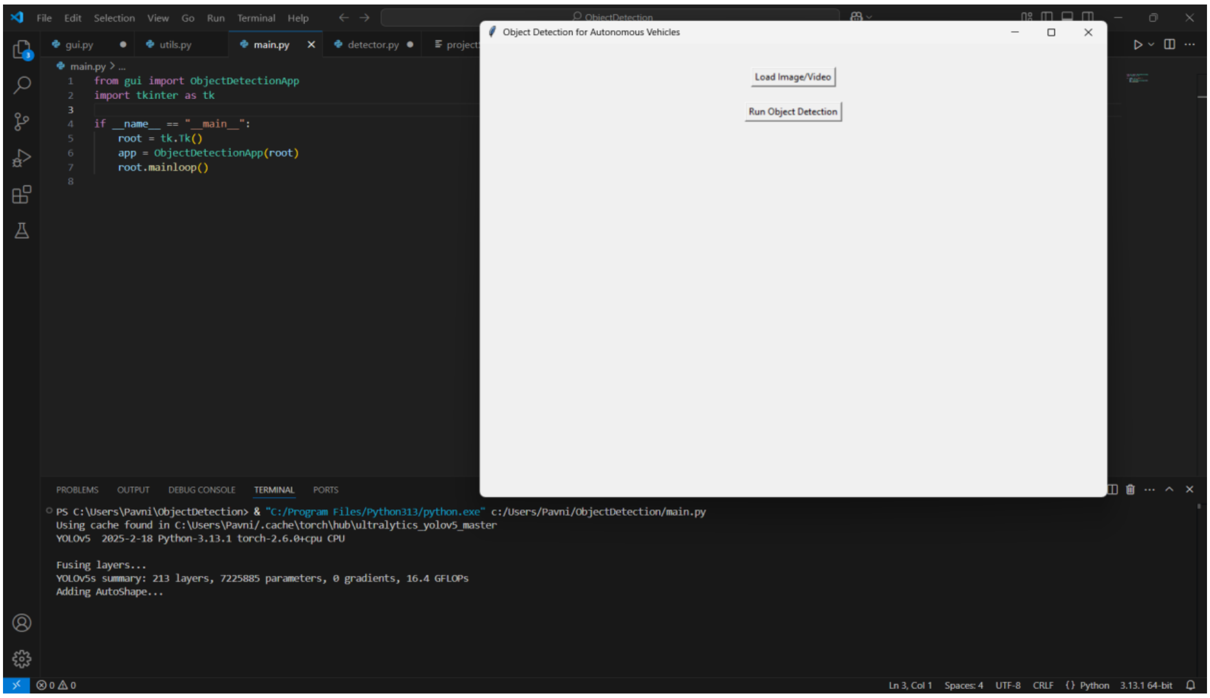
- detector.py:
    - Implements object detection using YOLOv5.
    - Analyzes images and videos, applying bounding boxes along with corresponding labels.
    - Sends the identified objects to gui.py for visualization.

```
import cv2
import torch
import numpy as np
class ObjectDetector:
    def __init__(self):
        self.model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)
    def detect_image(self, image_path):
        results = self.model(image_path)
        img = cv2.imread(image_path)
        detections = results.pandas().xyxy[0]

        for _, row in detections.iterrows():
            x1, y1, x2, y2 = int(row['xmin']), int(row['ymin']), int(row['xmax']), int(row['ymax'])
            label = f"{row['name']} {row['confidence']:.2f}"
            # Draw bounding box
            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 0, 255), 2)
            # Get text size and set position
            font_scale = 0.5
            thickness = 1
            text_size = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, font_scale, thickness)[0]
            text_x, text_y = x1, y1 - 10  # Position text above bounding box
            # Background rectangle for text
            overlay = img.copy()
            cv2.rectangle(overlay, (text_x, text_y - text_size[1] - 4), (text_x + text_size[0] + 6, text_y + 4), (0, 0, 0), -1)
            cv2.addWeighted(overlay, 0.6, img, 0.4, 0, img)
            # Put text label
            cv2.putText(img, label, (text_x + 3, text_y), cv2.FONT_HERSHEY_SIMPLEX, font_scale, (255, 255, 255), thickness)
        return img
```

- utils.py:
    - This section includes utility functions for managing files.
    - It facilitates the selection, validation, and storage of identified images.

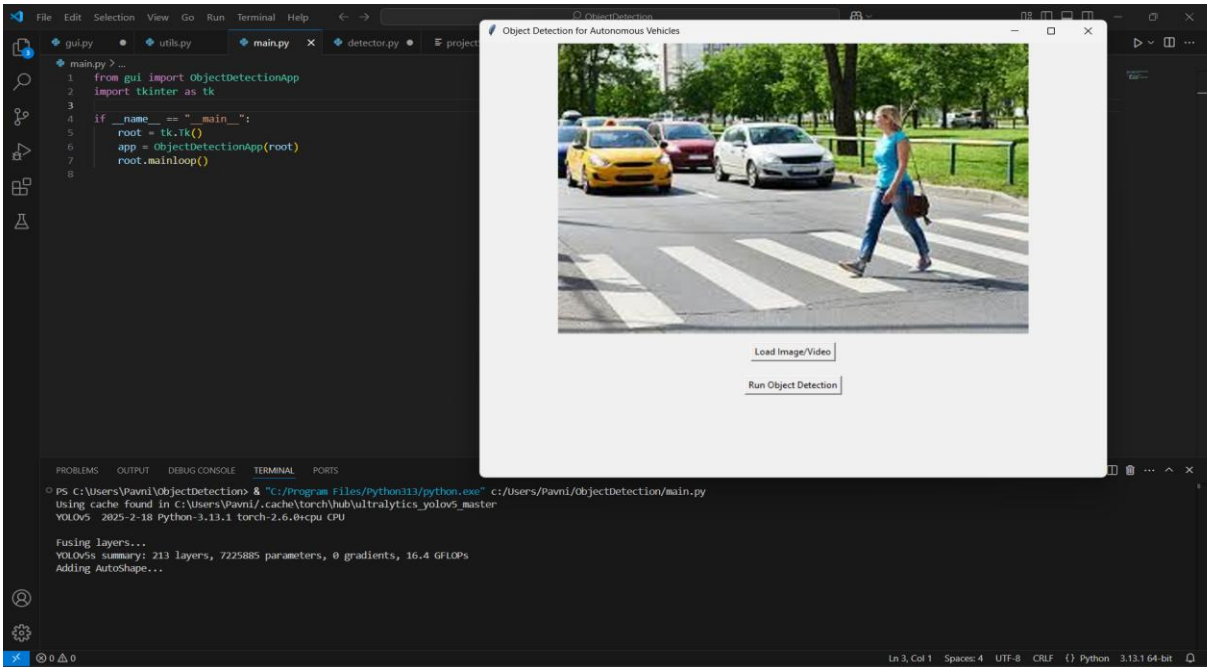
```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
from detector import ObjectDetector
from utils import is_image_file, is_video_file, save_detected_image


class ObjectDetectionApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Object Detection for Autonomous Vehicles")
        self.root.geometry("800x600")
        self.image_label = tk.Label(self.root)

        self.image_label.pack()

        # Buttons

        self.load_button = tk.Button(self.root, text="Load Image/Video", command=self.load_file)

        self.load_button.pack(pady=10)

        self.detect_button = tk.Button(self.root, text="Run Object Detection", command=self.run_detection)

        self.detect_button.pack(pady=10)

        self.save_button = tk.Button(self.root, text="Save Detected Image", command=self.save_image, state=tk.DISABLED)

        self.save_button.pack(pady=10)

        self.file_path = None

        self.detected_image = None

        # Initialize Object Detector

        self.detector = ObjectDetector()

    def load_file(self):

        file_types = [("Image/Video Files", "*.jpg *.jpeg *.png *.mp4 *.avi")]

        self.file_path = filedialog.askopenfilename(title="Select Image or Video", filetypes=file_types)


        if self.file_path:

            if is_image_file(self.file_path):

                self.display_image(self.file_path)
            elif is_video_file(self.file_path):
                messagebox.showinfo("Info", "Video loaded. Object detection will run on it.")
            else:
                messagebox.showerror("Error", "Unsupported file format.")


    def display_image(self, path):
        image = Image.open(path)
        image = image.resize((600, 400))
        photo = ImageTk.PhotoImage(image)
        self.image_label.config(image=photo)
        self.image_label.image = photo
    def run_detection(self):
        if not self.file_path:
            messagebox.showerror("Error", "Please load an image or video first.")
            return
        if is_image_file(self.file_path):
            self.detected_image = self.detector.detect_image(self.file_path)
            self.display_detected_image(self.detected_image)
            self.save_button.config(state=tk.NORMAL)
        elif is_video_file(self.file_path):
            messagebox.showinfo("Info", "Video detection is under development.")
        else:
            messagebox.showerror("Error", "Unsupported file format.")
    def display_detected_image(self, image):
        img = Image.fromarray(image)
        img = img.resize((600, 400))
        photo = ImageTk.PhotoImage(img)
        self.image_label.config(image=photo)
        self.image_label.image = photo
    def save_image(self):
        if self.detected_image is not None:
            img = Image.fromarray(self.detected_image)
            saved_path = save_detected_image(img)
            if saved_path:
                messagebox.showinfo("Success", f"Image saved at {saved_path}")
            else:
                messagebox.showwarning("Cancelled", "Image save cancelled.")
        else:
            messagebox.showerror("Error", "No detected image to save.")
if __name__ == "__main__":
    root = tk.Tk()
    app = ObjectDetectionApp(root)
    root.mainloop()
```

Output Screenshot


File Edit Selection View Go Run Terminal Help
gui.py
utils.py
main.py
detector.py
Object Detection for Autonomous Vehicles
Load Image/Video
Run Object Detection
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Ln 3, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit

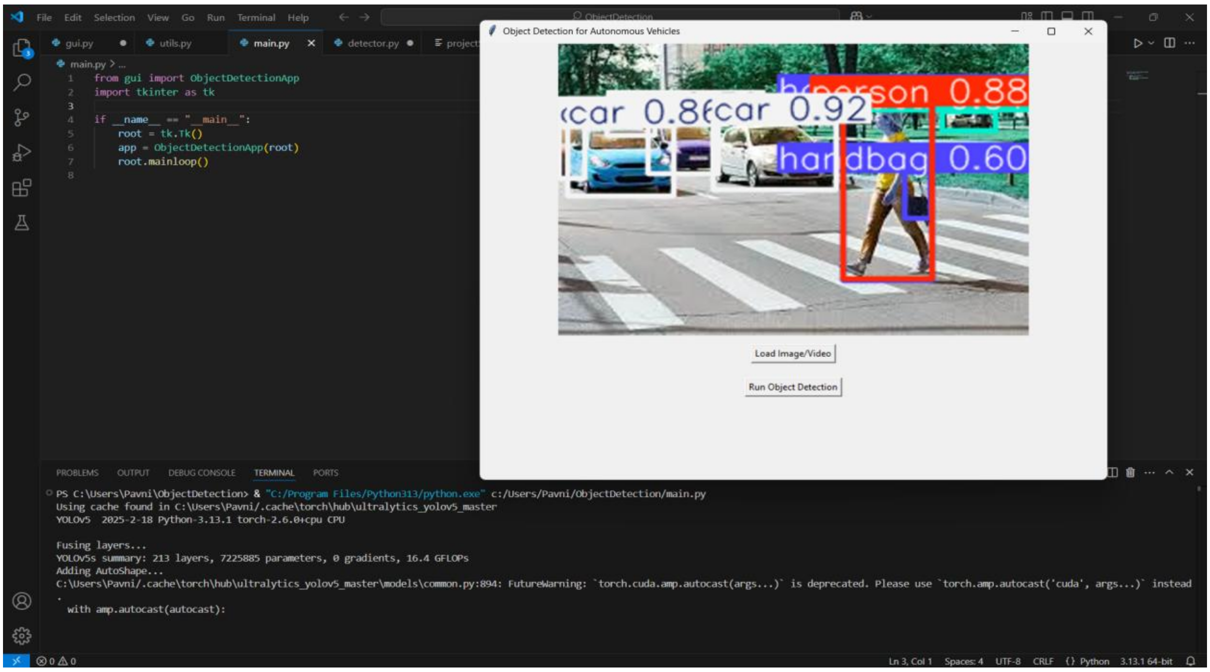
```python
from gui import ObjectDetectionApp
import tkinter as tk

if __name__ == "__main__":
    root = tk.Tk()
    app = ObjectDetectionApp(root)
    root.mainloop()
```

```
PS C:\Users\Pavni\ObjectDetection> & "C:/Program Files/Python313/python.exe" c:/Users/Pavni/ObjectDetection/main.py
Using cache found in C:\Users\Pavni/.cache\torch\hub\ultralytics_yolov5_master
YOLOv5  2025-2-18 Python-3.13.1 torch-2.6.0+cpu CPU

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs
Adding AutoShape...
```


File Edit Selection View Go Run Terminal Help
gui.py
utils.py
main.py
detector.py
Object Detection for Autonomous Vehicles
Load Image/Video
Run Object Detection
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Ln 3, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit

```python
from gui import ObjectDetectionApp
import tkinter as tk

if __name__ == "__main__":
    root = tk.Tk()
    app = ObjectDetectionApp(root)
    root.mainloop()
```

```
PS C:\Users\Pavni\ObjectDetection> & "C:/Program Files/Python313/python.exe" c:/Users/Pavni/ObjectDetection/main.py
Using cache found in C:\Users\Pavni/.cache\torch\hub\ultralytics_yolov5_master
YOLOv5  2025-2-18 Python-3.13.1 torch-2.6.0+cpu CPU

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs
Adding AutoShape...
```

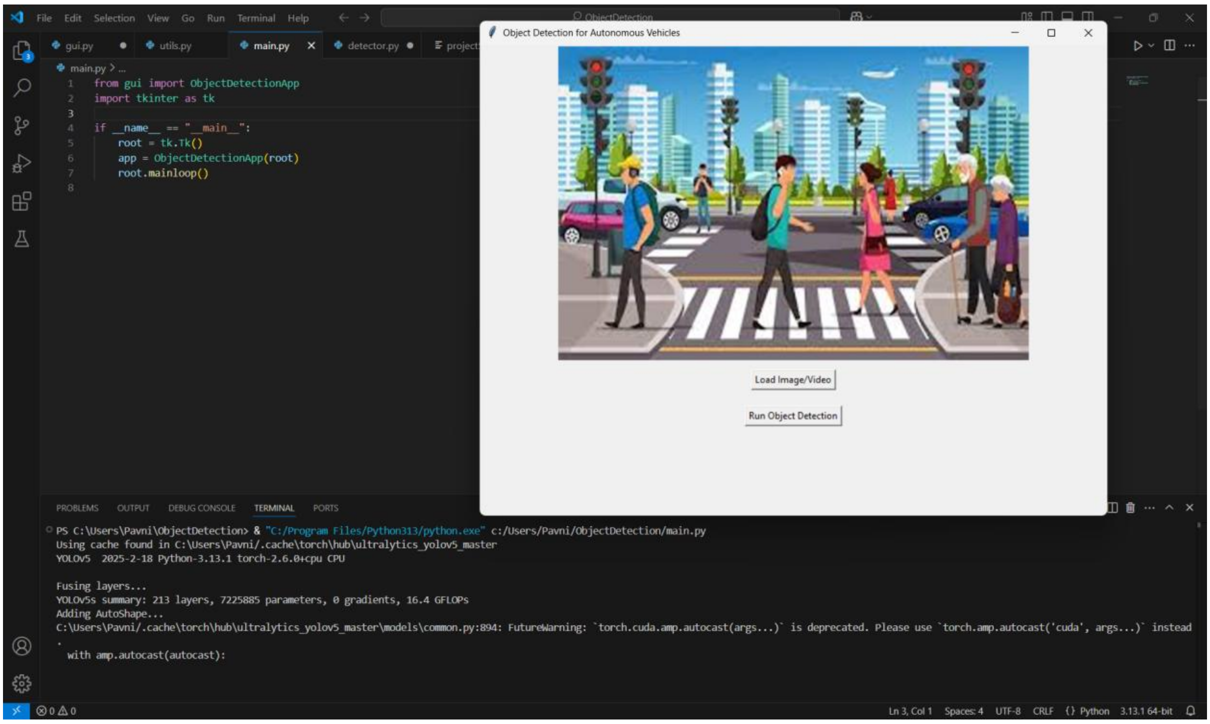


File Edit Selection View Go Run Terminal Help
gui.py
utils.py
main.py
detector.py
Object Detection for Autonomous Vehicles
car 0.86
car 0.92
person 0.88
handbag 0.60
Load Image/Video
Run Object Detection
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Ln 3, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit

main.py > ...

```python
from gui import ObjectDetectionApp
import tkinter as tk

if __name__ == "__main__":
    root = tk.Tk()
    app = ObjectDetectionApp(root)
    root.mainloop()
```

```
PS C:\Users\Pavni\ObjectDetection> & "C:/Program Files/Python313/python.exe" c:/Users/Pavni/ObjectDetection/main.py
Using cache found in C:\Users\Pavni/.cache\torch\hub\ultralytics_yolov5_master
YOLOv5  2025-2-18 Python-3.13.1 torch-2.6.0+cpu CPU

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs
Adding AutoShape...
C:\Users\Pavni/.cache\torch\hub\ultralytics_yolov5_master\models\common.py:894: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead
.
  with amp.autocast(autocast):
```

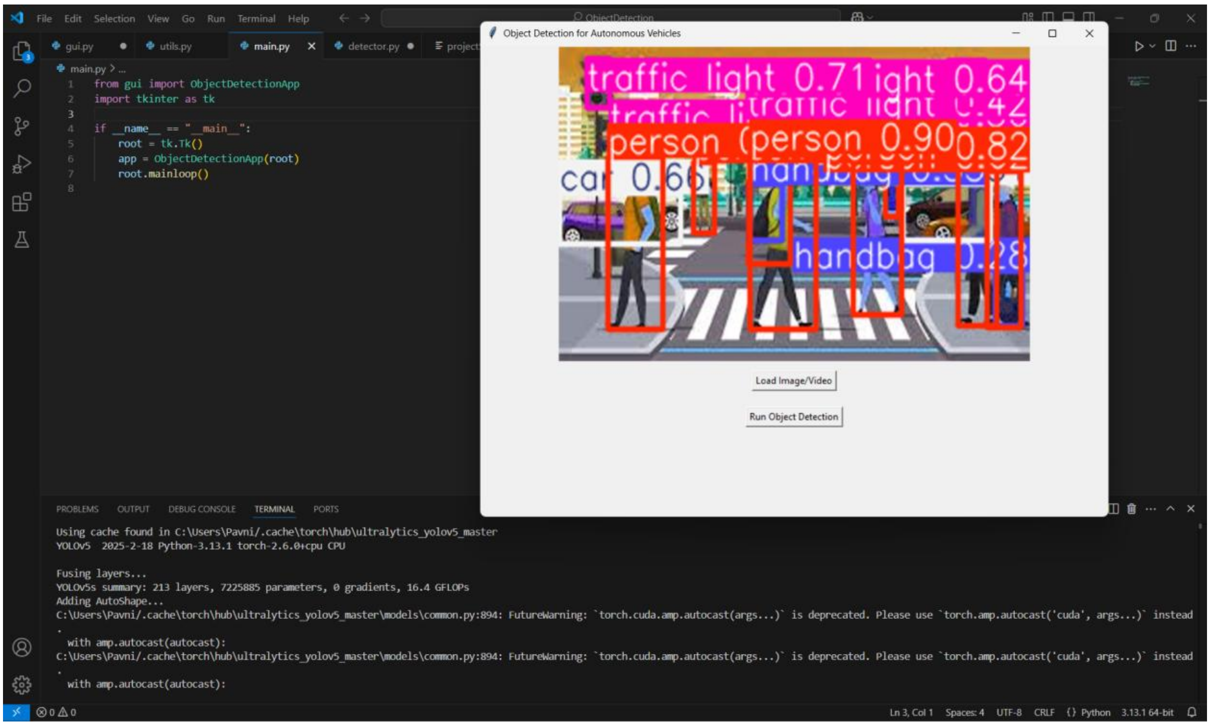


File Edit Selection View Go Run Terminal Help
gui.py
utils.py
main.py
detector.py
Object Detection for Autonomous Vehicles
Load Image/Video
Run Object Detection
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Ln 3, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit

main.py > ...

```python
from gui import ObjectDetectionApp
import tkinter as tk

if __name__ == "__main__":
    root = tk.Tk()
    app = ObjectDetectionApp(root)
    root.mainloop()
```

```
PS C:\Users\Pavni\ObjectDetection> & "C:/Program Files/Python313/python.exe" c:/Users/Pavni/ObjectDetection/main.py
Using cache found in C:\Users\Pavni/.cache\torch\hub\ultralytics_yolov5_master
YOLOv5  2025-2-18 Python-3.13.1 torch-2.6.0+cpu CPU

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs
Adding AutoShape...
C:\Users\Pavni/.cache\torch\hub\ultralytics_yolov5_master\models\common.py:894: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead
.
  with amp.autocast(autocast):
```



main.py > ...

```python
from gui import ObjectDetectionApp
import tkinter as tk

if __name__ == "__main__":
    root = tk.Tk()
    app = ObjectDetectionApp(root)
    root.mainloop()
```

```
Using cache found in C:\Users\Pavni/.cache\torch\hub\ultralytics_yolov5_master
YOLOv5  2025-2-18 Python-3.13.1 torch-2.6.0+cpu CPU

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs
Adding AutoShape...
C:\Users\Pavni/.cache\torch\hub\ultralytics_yolov5_master\models\common.py:894: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead
.
  with amp.autocast(autocast):
C:\Users\Pavni/.cache\torch\hub\ultralytics_yolov5_master\models\common.py:894: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead
.
  with amp.autocast(autocast):
```

Future Scope

This project can be further enhanced with additional functionalities, including:

- Real-time video analysis: Executing object detection on live video streams captured by cameras.
- Multi-object surveillance: Augmenting detection capabilities with tracking algorithms to monitor objects over time.
- Integration with autonomous vehicle simulations: Evaluating the model within simulated settings.
- Edge computing implementation: Executing object detection on Raspberry Pi or embedded systems for practical applications.
- Enhanced precision: Training a tailored YOLO model using a dataset focused on road environments.

Conclusion

This initiative effectively employs YOLOv5 and Tkinter to achieve real-time object detection for autonomous vehicles. The system proficiently identifies and categorizes objects, including vehicles, pedestrians, and traffic signs, showcasing its applicability in self-driving technology. Prospective improvements may concentrate on optimizing real-time performance and integrating with sophisticated driving systems to enhance safety and intelligence in autonomous navigation.

References and Bibliography

- Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement.
- Ultralytics YOLOv5 Documentation: https://github.com/ultralytics/yolov5
- OpenCV Library: https://opencv.org/
- Python Official Documentation: https://docs.python.org/3/