CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Real-Time Traffic Density Estimation with YOLOv8: This project uses YOLOv8 for real-time traffic density estimation. It employs fine-tuned vehicle detection models to analyze and count vehicles per frame, aiding urban traffic management and planning.

An Improved YOLO-Based Road Traffic Monitoring System: This system uses a combination of neural networks, image-based tracking, and YOLOv8 to track vehicles. It has been trained with different datasets and tested with real video sequences of road traffic.

Object Detection Using YOLO Framework for Intelligent Traffic Monitoring: This system uses a filtered YOLO for vehicle detection. It is tested for three classes of vehicles such as bus, truck, and car.

Real-Time Traffic Monitoring System Based on Deep Learning and YOLOv8: This system uses the state-of-the-art YOLOv8 algorithm for vehicle detection. These systems aim to improve traffic management by providing real-time, accurate traffic monitoring and analysis. They leverage the power of YOLO object detection and deep learning to offer efficient solutions for traffic control and city planning.

1.2 MOTIVATION

Motivations is the improvement of traffic management. Real-time traffic monitoring can provide traffic control centers with up-to-the-minute information about the traffic flow, allowing them to make informed decisions to reduce congestion and improve road safety. This can lead to a more efficient traffic management system, reducing the time spent in traffic and improving the overall commuting experience.

The current traffic management systems rely on traditional methods such as inductive loop detectors, video cameras, and manual reporting, which can be time-consuming and less accurate. However, with the advent of advanced object detection algorithms like YOLO, real-time traffic monitoring can provide accurate and reliable information about the traffic flow, enabling traffic control centers to make informed decisions quickly.

Real-time traffic monitoring can provide information about the number of vehicles on the road, their speed, and the direction they are moving in. This information can be used to identify congested areas, predict traffic jams, and suggest alternative routes to commuters. Traffic control centers can use this information to optimize traffic signals, manage lane closures, and divert traffic to less congested routes, reducing the time spent in traffic and improving the overall commuting experience.

1.3 PROBLEM DEFINITION

The task of object detection in video streams remains a challenging problem in computer vision, particularly in scenarios with complex backgrounds, occlusions, and variations in object appearance

- Achieving high accuracy in object detection across a wide range of object categories while minimizing false positives and false negatives.
- Ensuring real-time performance suitable for applications such as surveillance, autonomous driving.
- Enhancing the system's robustness to variations in lighting, scale, orientation, and occlusions.
- Minimizing the computational and memory requirements for deployment on embedded systems or edge devices.
- Optimizing traffic flow requires accurate and timely information about the movement of vehicles and pedestrians.
- There is a clear need for improved technologies to enhance the safety of road users.

1.4 OBJECTIVES

The primary objective of the project is to develop a real-time traffic monitoring system using advanced object detection techniques. This system aims to accurately identify and track vehicles and pedestrians in urban traffic scenes, providing valuable insights for traffic management and enhancing road safety.

The specific objectives include:

- Implementing efficient object detection algorithms to detect vehicles and pedestrians in real-time traffic scenarios.
- Utilizing state-of-the-art models like YOLO (You Only Look Once) for rapid and accurate object detection.
- Enabling real-time processing of live video feeds from traffic cameras for immediate detection and tracking.
- Incorporating features for traffic flow analysis, and other relevant metrics to aid traffic management.
- Training the object detection model using diverse datasets to adapt to various traffic conditions.
- Designing a user-friendly interface or visualization tool to present real-time traffic data clearly and comprehensively.

1.5 SCOPE OF THE PROJECT

The project aims to address the challenges faced by existing traffic monitoring systems and contribute to the improvement of urban traffic management. The primary elements within the scope of this project include:

- Object Detection System
- Real-time Processing
- Traffic Analysis Features
- Object Tracking Algorithms
- User-Friendly Interface

CHAPTER 2

LITERATURE SURVEY

SNO	TITLE,YEAR, AUTHORS	ADVANTAGE	METRICS	LIMITATION	TECHNIQUES /METHODS
1	A Survey On Vehicle Detection And Tracking Algorithms In Real Time Video Surveillance, Sri Jamiya S, Esther Rani P, year 2019.	Rani discusses the importance of detecting moving objects in video surveillance systems, highlighting the common steps involved in moving object detection, such as preprocessing, feature extraction, classification, detection, and tracking .	Accuracy: 93.4%	The limitations of the system include the need for testing under extreme weather conditions and occlusion problems the system is vulnerable to noise and has difficulty when focused mainly on a single class, making it difficult to search during classification .	The methodology used for vehicle classification involved processing a dataset of 3074 samples using various machine learning algorithms such as SVM, neural networks, and logical regression. Logical regression showed high performance with a classification rate of 93.4% compared to other machine learning methods.
2	ROAD TRAFFIC CONTROL USING MACHINE LEARNING	Controlling traffic signals in real-time using object detection and image	Controlling traffic signals in real-time is the vehicle density on	Controlling traffic signals in real-time using object	Controlling traffic signals in real-time utilizes object detection and image processing

	Deepthi.V.S, Dhanushri.V.S, Year 2021	processing techniques is its ability to dynamically adjust signal switching timing based on the actual vehicle density on each side of the road. This dynamic adjustment helps in optimizing traffic flow and reducing congestion by ensuring that signal switching is not solely based on predetermined regular intervals but rather on the real-time count of vehicles present. By implementing this approach, the system can effectively manage varying traffic densities and prevent specific lanes	each side of the road, which allows for dynamic adjustment of signal switching timing to optimize traffic flow. By utilizing this metric, the system can effectively manage varying traffic densities and prevent specific lanes from becoming overly congested, ultimately leading to smoother traffic flow .	detection and image processing techniques is the potential for inaccuracies in vehicle detection, which could lead to incorrect adjustments in signal switching timing . If the object detection algorithms used in the system are not robust enough to accurately identify and count vehicles in varying lighting conditions or complex traffic scenarios, it may result in suboptimal traffic flow management and	techniques to monitor and adjust signal switching based on vehicle density. The system employs the Kalman filter and Gaussian Mixture Model for object detection, with a focus on accurately identifying and counting vehicles in varying traffic scenarios . The use of Convolutional Neural Networks (CNN) in machine learning aids in training the model for object detection, while OpenCV is utilized for noise reduction and object identification in the YOLO (You Only Look Once) pretrained model .

		from becoming overly congested, ultimately leading to smoother traffic flow .		potential congestion issues.	
3	Traffic Prediction for Intelligent Transportation System using Machine Learning Gaurav Meena, Deepanjali Sharma, Mehul Mahrishi, Year 2020	The proposed algorithm for identifying traffic congestion in Intelligent Transportation Systems shows promising results in terms of accuracy, The evaluation of different machine learning algorithms for traffic prediction provides insights into their performance, The Random Forest algorithm is identified as the bestperforming algorithm in terms of accuracy for	Metrics used in the study included accuracy, precision, recall, and time taken for different machine learning algorithms. The Random Forest algorithm showed the highest accuracy rate of 91%, followed by Decision Tree with 88% accuracy and SVM with 88% accuracy as well .	Limitation of the study is that the dataset developed does not have many features, which may limit the applicability of deep learning and genetic algorithms	Machine learning algorithms for traffic prediction, including Decision Tree, Support Vector Machine (SVM), and Random Forest. The Decision Tree algorithm was used to predict the value of target variables by performing tests on the training dataset .

		traffic prediction.			
4	Object Detection in Traffic Videos: A Survey, Hang Shi, and Chengjun Liu, Year 2021	A comprehensive review of different algorithms used for object detection in traffic surveillance applications, categorizing the methods into motion-based and appearance-based techniques, potential solutions, recent trends, and future directions in the field of object detection in traffic videos .	Object detection methods in traffic surveillance applications include Precision, Recall, and FMeasure .	Limitation in the field of object detection in traffic surveillance applications is the challenge of occlusions. Locating objects solely based on motion information can lead to severe performance drops in scenarios where objects are occluded by each other	Traditional methods based on handcrafted features like Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), Gabor Features, Convolutional Neural Networks (CNNs): Modern method.
5	Object Detection using You Only Look Once (YOLO) Algorithm in Convolution Neural Network (CNN), Meghana Pulipalupula, Srija Patlola,	One of the advantages of using the You Only Look Once (YOLO) algorithm for object detection is its ability to provide better results for	The You Only Look Once (YOLO) algorithm utilizes the Mean Average Precision (mAP) metric to determine	You Only Look Once (YOLO) algorithm is that it may struggle to detect small objects unless the algorithm is specifically trained on	This technique differs from other methods like RCNN and Fast RCNN, which may require multiple runs to detect objects. YOLO's approach of processing images in a single pass through the

	Mahesh Nayaki, year 2023	object detection compared to other methods like Fast R-CNN and Retina-Net. This algorithm can detect any class of objects, provided that the dataset is trained and tested with images of those classes.	the confidence level of object detection on a scale of 0 to 1.	similar objects . This limitation can impact the overall accuracy and effectiveness of the algorithm in scenarios where the detection of small objects is crucial. the performance of YOLO may vary depending on the quality and quantity of images in the dataset used for training and testing	network contributes to its efficiency in object detection tasks .
6	Real Time Object Detection System with YOLO and CNN Models, Viswanatha V, Chandana R K, Ramachandra A.C	The YOLO algorithm is its ability to train the complete model unified object detection model that is easy to build and train due to its simple loss function. This allows for efficient	object detection algorithms like YOLO is mean Average Precision (mAP). It provides a comprehensive measure of the algorithm's ability to	the YOLO algorithm is that it may struggle with detecting small objects or objects that are closely clustered together in an image. This can lead to	YOLO (You Only Look Once) for efficient and accurate detection of objects in images and videos. YOLO algorithm is known for its ability to provide real-time object detection by processing the complete image in a single pass,

		training and implementation, providing a significant advantage over other object detection models.	accurately detect objects in an image dataset.	lower accuracy in detecting these types of objects compared to other object detection algorithms .	making it suitable for applications requiring speed and accuracy .
7	Enhancing Realtime Object Detection with YOLO Algorithm, Gudala Lavanya Sagar Dhanraj Pande, Year 2023	The YOLO algorithm offers several advantages in object detection, including realtime processing, simplicity, and effective handling of small objects . YOLO can predict all bounding boxes and classes for the entire image in one pass through the network, YOLO has a high prediction capacity with fewer background mistakes, high learning capabilities, and a high-resolution classifier .	YOLOv3 achieved a mAP of 37 on the COCO-2017 validation set with an input resolution of 608x608. This metric indicates the algorithm's ability to accurately detect objects in images and assess its overall performance in object detection tasks.	YOLO algorithm is its reduced accuracy in detecting small objects due to the single-stage architecture and grid-based approach . The algorithm may struggle to accurately localize and classify small objects within images, leading to lower precision and recall rates for these objects.	technique that has been used to enhance underwater image dehazing is the integration of a Convolutional Neural Network (CNN) with a block-greedy algorithm. This method addresses color channel attenuation, optimizes local and global pixel values, and refines image edges using a unique Markov random field.

		These advantages make YOLO a powerful tool for efficient and accurate object detection in computer vision applications.			
8	Literature Review on Traffic Control Systems Used Worldwide 1 Vaishali Mahavar, Prof. Jayesh Juremalani, Year 2018	Optimizing traffic signal control systems. These advantages include real-time adaptation to changing traffic dynamics , reduction of vehicle delays at intersections, balancing traffic flow, and improving operational efficiency of urban street networks . Additionally, adaptive controllers can minimize drawbacks of conventional	The metrics used to evaluate the performance of adaptive traffic signal controllers include realtime traffic conditions, volume, congestion, delay time, user equilibrium traffic, oversaturation, travellers' route choice, and various traffic disruption events . These metrics are essential for formulating effective methods to	The complexity and cost associated with implementing and maintaining these systems, which may pose challenges for some municipalities or transportation agencies . Additionally, the effectiveness of adaptive controllers can be influenced by factors such as the accuracy of	Traffic signal controllers utilize various techniques and methods to optimize traffic signal control systems. Some commonly used techniques include: 1. TRANSYT software 2. Genetic algorithms 3. Generalized proportional allocation controllers 4. Group-based signal control 5. Memetic algorithms 6. Reinforcement learning (RL) algorithms for designing

		traffic controllers by accurately varying green cycle intervals based on heavy traffic loads .	optimize traffic signal control systems in urban areas.	traffic data inputs, the availability of real-time information , and the need for continuous calibration and adjustment to ensure optimal performanc e .	adaptive traffic signal controllers 7. Dynamic routing in a network with adaptive signal control 8. Elimination pairing system 9. Fuzzy approaches 10. Neural networks
9	Survey of The Problem of Object Detection In Real Images , Dilip K. Prasad, year 2012	Edge-based features are highlighted as advantageous for object detection due to their invariance to illumination conditions, variations in object colors, and textures. They also effectively represent object boundaries and efficiently capture data in the large spatial extent of images	Edge-based features as a metric used.	The paper is the challenge of obtaining complete contours for training images when using edge-based features. In real images, incomplete contours are common due to occlusion and noise, which can impact the effectivenes s of using complete contours as features **.**	Edge-based features as a key point for learning and subsequent object detection [3]. These techniques involve extracting the edge map of the image and identifying object features in terms of edges, which are advantageous due to their invariance to illumination conditions, variations in object colors, and textures .

10	Real-Time Object Detection using YOLO, Upulie H.D.I , Lakshini Kuganandamurthy, Year 2021	The YOLO (You Only Look Once) algorithm is its speed of identification, making it applicable for real-time object detection . YOLOv2, the latest version of YOLO, has achieved a mean Average Precision (mAP) rate of 76.8 at 67 Frames per Second (FPS) and 78.6 mAP rate at 76 FPS, outperforming regional-based algorithms such as Faster R-CNN in both speed and accuracy .	The metrics used in evaluating the performance of the YOLO algorithm is the mean Average Precision (mAP) rate. YOLOv2, the latest version of YOLO, has achieved a mean Average Precision (mAP)	The limitations of the YOLO algorithm is its spatial constraints on bounding boxes, as each cell can predict only two boxes and one class, limiting the number of predictable objects nearby to each other in groups. YOLO may struggle with generalizing objects in unusual or new aspect ratios due to being trained only on input data .	The techniques used in object detection algorithms is the utilization of Convolutional Neural Networks (CNNs). CNNs have been instrumental in providing solutions for object detection by extracting features from input data through a weightsharing process, enabling the network to analyze highdimensional data and achieve accurate classification .

CHAPTER 3

REQUIREMENTS

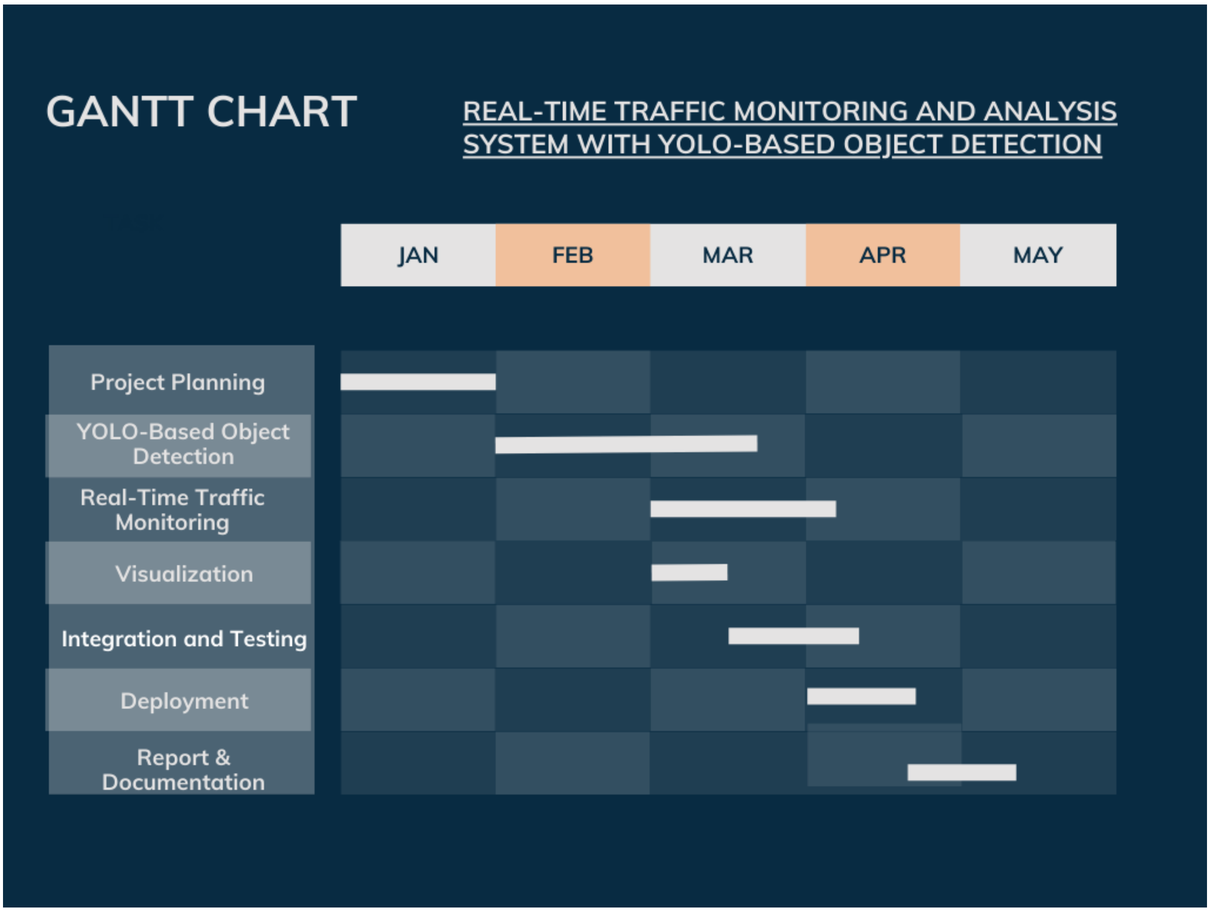
3.1 HARDWARE REQUIREMENTS

- Processor: A multi-core processor with a clock speed of at least 2.5 GHz is recommended. This will ensure that the system can handle the computational demands of machine learning algorithms.
- Memory: At least 16 GB of RAM is recommended for running machine learning algorithms and processing large datasets.
- Storage: A solid-state drive (SSD) with at least 500 GB of storage is recommended for storing datasets and machine learning models.
- Graphics: A dedicated graphics card with at least 4 GB of memory is recommended for accelerating machine learning algorithms.
- Network: A high-speed network connection is recommended for real-time data transfer and communication.

3.2 SOFTWARE REQUIREMENTS

- Operating System: Windows, macOS, or Linux
- Anaconda Distribution: This is a free and open-source distribution of Python and R programming languages for scientific computing, that aims to simplify package management and deployment.
- Python 3.7: This is the version of Python that you will be using for your project.

3.3 GANTT CHART



CHAPTER 4

ANALYSIS & DESIGN

4.1 PROPOSED METHODOLOGY

OBJECT DETECTION: The YOLOv3 object detection model is used to detect objects in each frame. The model is pre-trained on the COCO dataset and is used to detect objects from 80 classes.

OBJECT TRACKING: The Centroid Tracker class is used to track the objects between frames. The class calculates the centroid of each object and compares it with the centroids of the previous frame. If the distance between the centroids is less than a certain threshold, the object is considered to be the same object.

CORRELATION TRACKING: The dlib correlation tracker is used to track the objects between frames. The tracker calculates the correlation between the current frame and the previous frame to estimate the position of the object. MULTI-THREADING: The object detection and tracking algorithms are run in separate threads to improve performance. The object detection is run every 10 frames and the tracking is run in between the detection frames.

YOLO

You only look once (YOLO) is a state-of-the-art, real-time object detection system YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. The object detection task consists in determining the location on the image where certain objects are present, as well as classifying those objects. Previous methods for this, like R- 3 CNN and its variations, used a pipeline to perform this task in multiple steps. This can be slow to run and also hard to optimize, because each individual component must be trained separately. YOLO, does it all with a single neural network.

YOLOv8:

YOLOv8 is the latest version of the YOLO (You Only Look Once) object detection model, developed by Ultralytics. It is a powerful and efficient model designed for object detection, image classification, and instance segmentation tasks. YOLOv8 builds upon the success of previous YOLO models and introduces several improvements, making it a versatile and advanced tool for computer vision tasks.

One of the key features of YOLOv8 is its high accuracy. The model achieves strong accuracy on the COCO dataset, with the YOLOv8m model achieving a 50.2% mean average precision (mAP) when measured on COCO. This high accuracy is achieved through a combination of architectural changes and developer experience enhancements.

YOLOv8 introduces an anchor-free detection system, which improves generalization and reduces the learning speed for custom datasets. This system allows the model to better understand the context of the objects it is detecting, leading to more accurate and reliable detections.

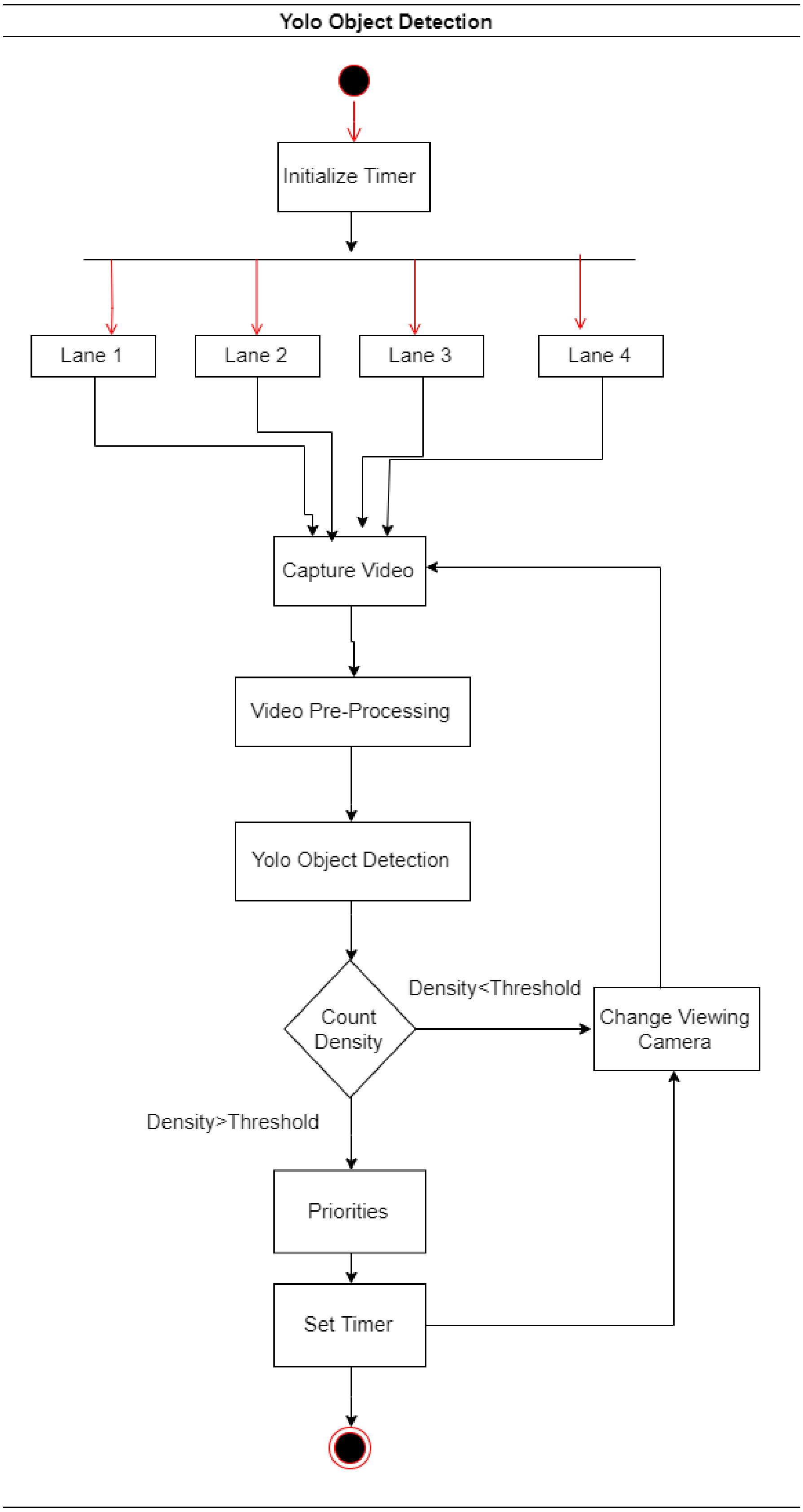
To further enhance the context information, YOLOv8 uses mosaic data augmentation. This technique combines multiple images into a single image, providing the model with a more comprehensive view of the scene. The model uses this augmentation during training, stopping it in the last ten training epochs to improve performance.

To improve performance, YOLOv8 separates the classification and detection heads. This decoupling allows the model to focus on specific aspects of the task, leading to more accurate and efficient detections.

In addition to these improvements, YOLOv8 also uses a modified loss function for better learning. This loss function is designed to improve the model's ability to learn from the data and adapt to new situations.

YOLOv8 is available in five variants based on the number of parameters: nano(n), small(s), medium(m), large(l), and extra large(x). These variants are designed to cater to different use cases, with the smaller variants being more efficient and the larger variants offering higher accuracy.

4.2 SYSTEM ARCHITECTURE



4.3 MODULE DESCRIPTIONS

TENSORNETS:

This module is used to define the YOLOv8 object detection model. The model is pre-trained on the COCO dataset and is used to detect objects in each frame.

CV2:

This module is used for image processing and computer vision tasks. It is used to read and display videos, resize images, and draw bounding boxes and text on the images.

NUMPY:

This module is used for numerical computations and array manipulation. It is used to preprocess the input images and postprocess the output of the YOLOv8 model.

TENSORFLOW:

This module is used for machine learning and deep learning tasks. It is used to define the

THREADING:

This module is used for multi-threading. It is used to run the object detection and tracking algorithms in separate threads.

CHAPTER 5

IMPLEMENTATION & TESTING

5.1 DATA SET

The YOLOv8 model is designed to perform object detection, instance segmentation, and image classification tasks. It comes pre-trained on various datasets, including the COCO detection dataset, COCO segmentation dataset, and the ImageNet dataset. The pre-trained models available for YOLOv8 include object detection checkpoints, instance segmentation checkpoints, and image classification models, all of which are trained on their respective datasets. To use the pre-trained models, you can load them using the Ultralytics YOLO library in Python. For example, to load a pre-trained object detection model.

you can use the following code:

```
python
from ultralytics import YOLO

# Load a COCO-pretrained YOLOv8n model
model = YOLO('yolov8n.pt')
```

Once the model is loaded, you can perform tasks such as training, validation, and inference on the pre-trained model. For instance, to train the model on the COCO dataset,

you can use the train method:

```
# Train the model on the COCO dataset
results = model.train(data='coco.yaml', epochs=100, imgsz=640)
```

To perform inference on an image, you can use the predict method:

```
# Run inference on the 'bus.jpg' image
results = model('path/to/bus.jpg')
```

The pre-trained models are available in various sizes and performance levels, including YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x, each with different trade-offs in terms of model size, speed, and accuracy.

5.2 SAMPLE CODE

app.py

```
import os
from flask import Flask, render_template, request, redirect, url_for
import subprocess

app = Flask(__name__)

# Create a folder for uploads if it doesn't exist
UPLOAD_FOLDER = 'uploads'
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

# Create a folder for annotated videos if it doesn't exist
ANNOTATED_FOLDER = 'annotated_videos'
if not os.path.exists(ANNOTATED_FOLDER):
    os.makedirs(ANNOTATED_FOLDER)

# Dummy database for demonstration purposes
users = {
    'admin': '123',
    'admin': '456'
}
```

```

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']

    if username in users and users[username] == password:
        # Redirect to select page after successful login
        return redirect(url_for('select'))
    else:
        # Redirect back to login page with error message
        return redirect(url_for('index'))

@app.route('/signup', methods=['POST'])
def signup():
    username = request.form['new-username']
    password = request.form['new-password']
```

```python

    # Add new user to the database (in reality, you'd hash the password)
    users[username] = password

    # Redirect to login page after successful signup
    return redirect(url_for('index'))

@app.route('/select')
def select():
    # Render select page after successful login
    return render_template('select.html')

@app.route('/upload', methods=['POST'])
def upload():
    # Get the number of ways and uploaded files from the request
    num_ways = int(request.form['num_ways'])
    files = request.files.getlist('files')

    # Handle file storage here
    file_paths = []
    for file in files:
```

```python
    for file in files:
        filename = file.filename
        file_path = os.path.join(UPLOAD_FOLDER, filename)
        file.save(file_path)
        file_paths.append(file_path)

    # Call the script to process uploaded files
    results = subprocess.check_output(['python', 'counted_vehicles.py'] + file_paths, text=True)

    # Parse the results to extract annotated video paths and vehicle counts
    annotated_videos = []
    for line in results.split('\n'):
        if line.startswith('Annotated video:'):
            annotated_videos.append(line.split(': ')[1])
        elif line.startswith('Total vehicles detected in'):
            annotated_videos.append(line)

    # Redirect to a page showing the uploaded files or perform further processing
    return render_template('uploaded_files.html', annotated_videos=annotated_videos)

if __name__ == '__main__':
    app.run(debug=True)

```

counted_vehicles.py

```python
import sys
import cv2
from ultralytics import YOLO
import os

def vehicle_detection(video_paths):
    # Load the YOLOv8 model
    model = YOLO('yolov8m.pt')

    for video_path in video_paths:
        # Open the video file
        cap = cv2.VideoCapture(video_path)

        # Define the output video file
        filename = os.path.basename(video_path)
        output_path = os.path.join('static', 'annotated_videos', f"{filename[:-4]}_annotated.mp4")
        fps = int(cap.get(cv2.CAP_PROP_FPS))
        fourcc = cv2.VideoWriter_fourcc(*'mp4v')
        out = cv2.VideoWriter(output_path, fourcc, fps, (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)), int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))))

        # Initialize the vehicle count
        vehicle_count = 0

        # Loop through the video frames
        while cap.isOpened():
            # Read a frame from the video
            success, frame = cap.read()

            if success:
                # Run YOLOv8 tracking on the frame
                results = model.track(frame)

                # Increment the vehicle count
                vehicle_count += len(results[0].boxes.data)

                # Visualize the results on the frame
                annotated_frame = results[0].plot()

                # Write the annotated frame to the output video file
                out.write(annotated_frame)

            else:
                # Break the loop if the end of the video is reached
                break

        # Release the video capture object and the output video file
        cap.release()
        out.release()

        # Print the total number of vehicles detected for each video
        print(f"Total vehicles detected  {vehicle_count}")

        # Write the total vehicle count to out.txt
        with open('out.txt', 'a') as file:
            file.write(f"Total vehicles detected {vehicle_count}\n")

    return [output_path for _ in video_paths]

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python counted_vehicles.py <video_path1> [<video_path2> ...]")
        sys.exit(1)

    video_paths = sys.argv[1:]
    annotated_videos = vehicle_detection(video_paths)
    for video in annotated_videos:
        print(f"Annotated video: {video}")
```

Program.py

```python
f = open("out.txt", "r")
no_of_vehicles=[]
no_of_vehicles.append(int(f.readline()))
no_of_vehicles.append(int(f.readline()))
no_of_vehicles.append(int(f.readline()))
no_of_vehicles.append(int(f.readline()))

baseTimer = 120  # baseTimer = int(input("Enter the base timer value"))
timeLimits = [5, 30]  # timeLimits = list(map(int,input("Enter the time limits ").split()))

print("Input no of vehicles : ", *no_of_vehicles)
t = [(i / sum(no_of_vehicles)) * baseTimer if timeLimits[0] < (i / sum(no_of_vehicles)) * baseTimer < timeLimits[1]
else min(timeLimits, key=lambda x: abs(x - (i / sum(no_of_vehicles)) * baseTimer)) for i in no_of_vehicles]

print(t, sum(t))
```

5.3 SAMPLE OUTPUT

Figure 5.3.1:

Login page:

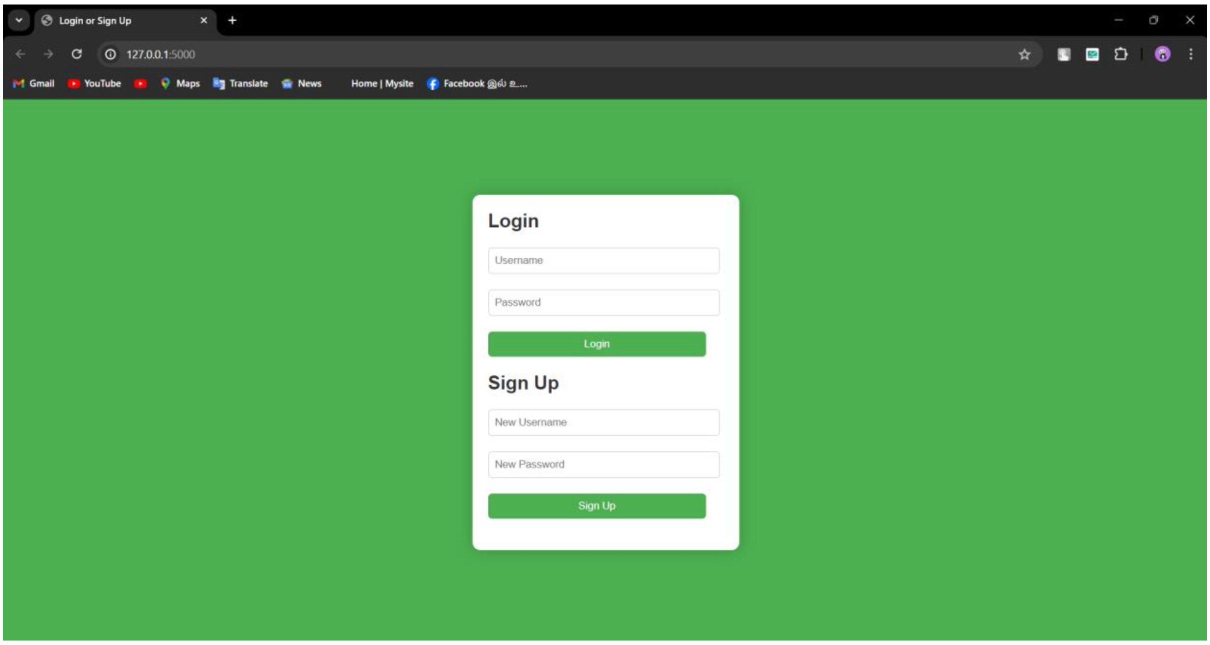


Figure 5.3.2:

Uploading file :

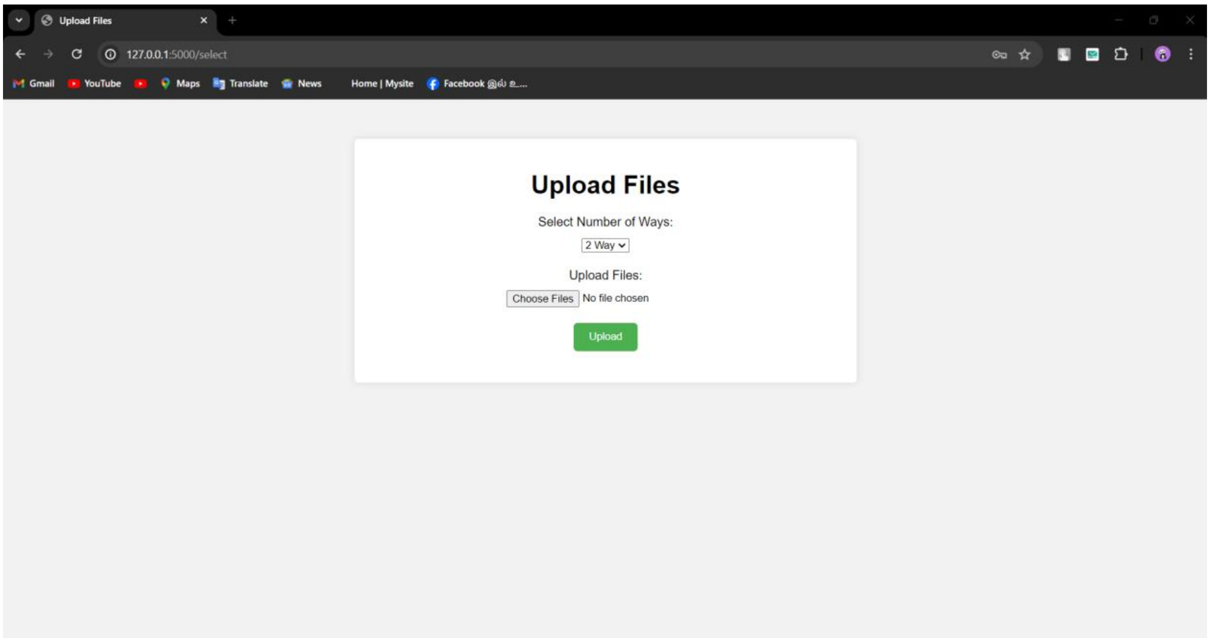


Figure 5.3.3:

Selecting the how many ways:

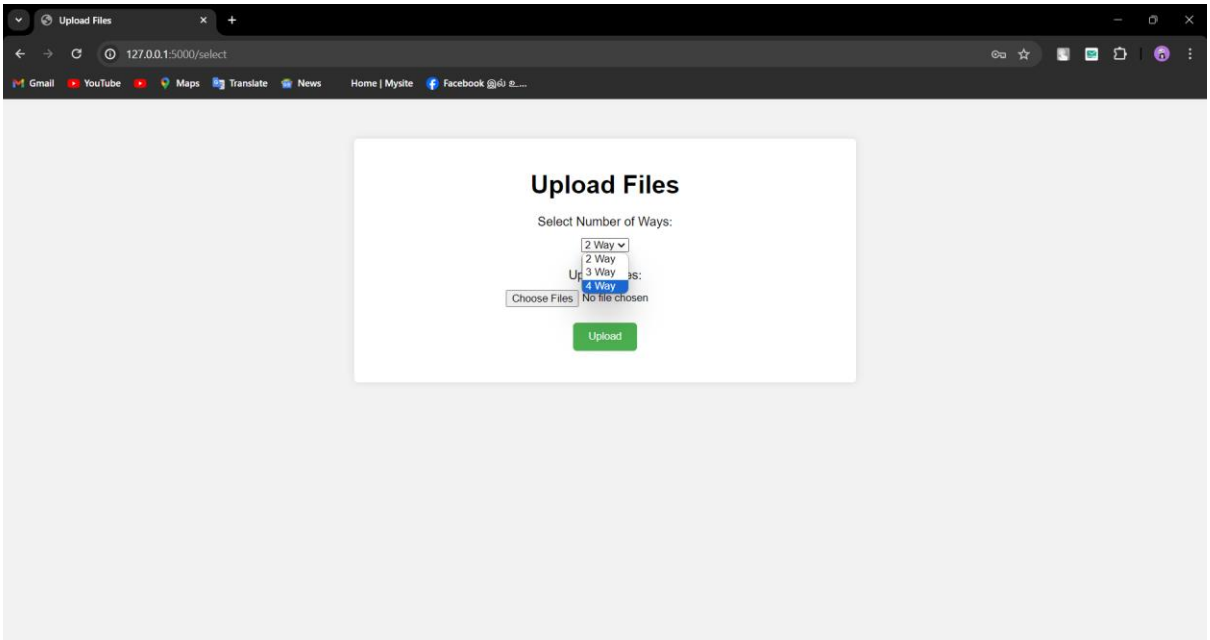


Figure 5.3.4:

Uploading videos:

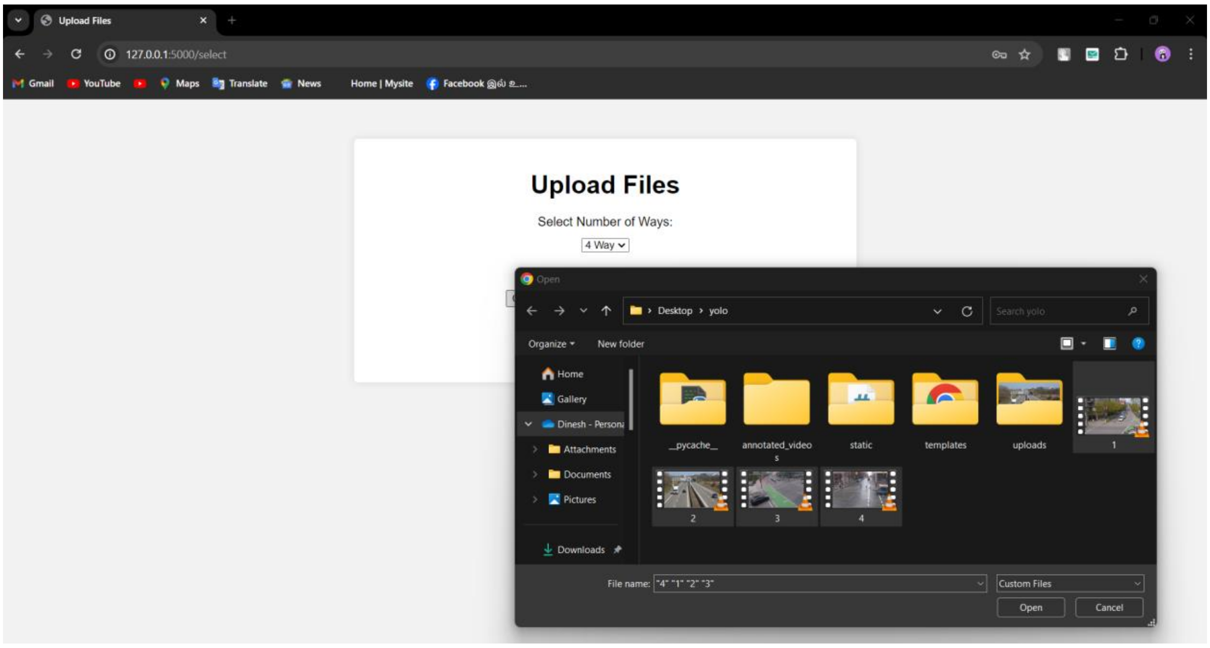
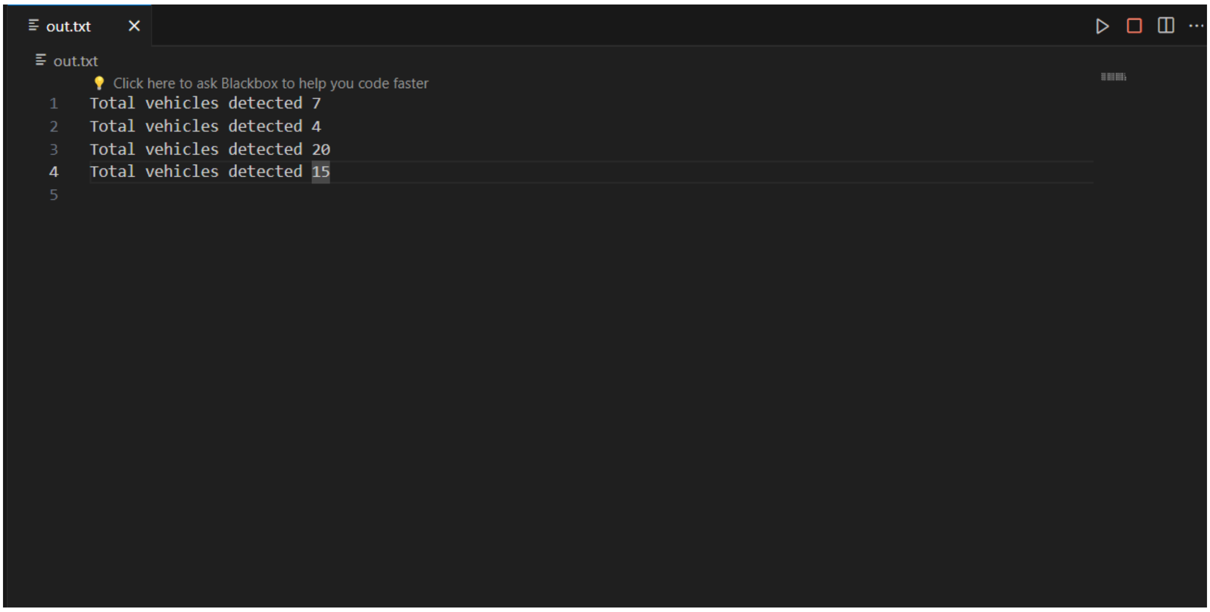


Figure 5.3.5:

Printing how many vehicle is detected from each video:



```
out.txt
Total vehicles detected 7
Total vehicles detected 4
Total vehicles detected 20
Total vehicles detected 15
```

Figure 5.3.6:

Calculating the timing for the signal:

```
[Running] python -u "c:\Users\Dinesh\Desktop\yolo\program.py"
Input no of vehicles :  8 5 14 3
[30, 20.0, 30, 12.0] 92.0
```

Figure 5.3.7:

Detect the vehicles and assign the signal timing :

CHAPTER 6

RESULTS

6.1 RESULT ANALYIS & EVALUATION METRICS

Object detection models are evaluated using various metrics to quantify their performance. The most used metrics include Mean Average Precision (mAP), Precision, Recall, and F1 Score.

Mean Average Precision (mAP) is a popular metric used to evaluate the performance of object detection models. It calculates the average precision across multiple object classes and is useful in multi-class object detection scenarios. The mAP is calculated by averaging the AP values for each class, where AP is the area under the precision-recall curve. The mAP ranges from 0 to 1, with higher values indicating better performance.

Precision and Recall are two other important metrics for object detection. Precision quantifies the proportion of true positives among all positive predictions, assessing the model's capability to avoid false positives.

Recall calculates the proportion of true positives among all actual positives, measuring the model's ability to detect all instances of a class. Both precision and recall are important for understanding the model's performance, as a high precision model may miss some true positives (low recall), while a high recall model may have some false positives (low precision).

F1 Score is the harmonic mean of precision and recall, providing a single value that encapsulates the model's overall performance. It is a useful metric for balancing precision and recall, as it penalizes models that perform well on one metric but poorly on the other.

YOLOv8, the model is evaluated using these metrics to determine its performance in object detection tasks. The mAP is a key metric for evaluating the model's ability to detect objects across multiple classes, while precision and recall are used to assess its performance in specific scenarios. The F1 Score provides an overall measure of the model's performance, balancing precision and recall.

CONCLUSIONS

The development and implementation of a Real-Time Traffic Monitoring and Analysis System utilizing YOLO-based Object Detection mark a significant leap forward in transportation management and urban planning. This innovative system capitalizes on cutting-edge deep learning technology to provide accurate, real-time insights into traffic flow, congestion patterns, and road safety metrics. By leveraging the YOLO (You Only Look Once) algorithm, the system achieves remarkable efficiency in detecting and tracking vehicles, pedestrians, and other objects of interest with minimal computational overhead.

The utilization of YOLO-based Object Detection enables the system to process high volumes of video data streams rapidly, ensuring timely updates on traffic conditions and potential incidents. This rapid processing capability is crucial for facilitating quick decision-making by traffic management authorities and emergency responders, thereby reducing response times and mitigating the impact of traffic-related incidents.

Future work

Future work for the project includes enhancing security measures through robust input validation and authentication methods. Improving error handling and providing clear user feedback are crucial for usability. Exploring asynchronous processing techniques for scalability and optimizing performance will be beneficial. Consideration of potential features like real-time monitoring and advanced analytics could further enhance the project's functionality and utility.

REFERENCES

[1] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. A cm Computing
Surveys (CSUR), 38(4):13, 2006.

[2] Gary Bishop and Greg Welch. An introduction to the Kalman filter. Proc of SIGGRAPH, Course, 8:27599–3175, 2001.

[3] J Cezar Silveira Jacques, Claudio Rosito Jung, and Soraia Raupp Musse. Background subtraction and shadow detection in grayscale video sequences. In Computer Graphics and Image Processing, 2005. SIBGRAPI 2005. 18th Brazilian Symposium on, pages 189–196. IEEE,
2005.

[4] Budi Sugandi, Hyoungseop Kim, Joo Kooi Tan, and Seiji Ishikawa. A block matching technique
for object tracking employing peripheral increment sign correlation image. In Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on, pages 113–117.
IEEE, 2008.

[5] Alan J Lipton, Hironobu Fujiyoshi, and Raju S Patil. Moving target classification and tracking
from real-time video. In Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on, pages 8–14. IEEE, 1998

[6] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., volume 2. IEEE, 1999.

[7] Ya Liu, Haizhou Ai, and Guang-you Xu. Moving object detection and tracking based on background subtraction. In Multispectral Image Processing and Pattern Recognition, pages

62–66. International Society for Optics and Photonics, 2001.

[8] Changick Kim and Jenq-Neng Hwang. Fast and automatic video object segmentation and tracking for content-based applications. Circuits and Systems for Video Technology, IEEE Transactions on, 12(2):122–129, 2002.

[9] Shahbe Mat Desa and Qussay A Salih. Image subtraction for real time moving object extraction. In Computer Graphics, Imaging and Visualization, 2004. CGIV 2004. Proceedings. International Conference on, pages 41–45. IEEE, 2004.

[10] Budi Sugandi, Hyoungseop Kim, Joo Kooi Tan, and Seiji Ishikawa. Tracking of moving objects

by using a low resolution image. In Innovative Computing, Information and Control, 2007. ICICIC'07. Second International Conference on, pages 408–408. IEEE, 2007.

[11] YUTAKA Sato, S Kaneko, and SATORU Igarashi. Robust object detection and segmentation

by peripheral increment sign correlation image. Trans. of the IEICE, 84(12):2585–2594, 2001.

[12] Mahbub Murshed1/2, Md Hasanul Kabir1/2, and Oksam Chae1/2. Moving object tracking-an

edge segment based approach. 2011.

[13] Weiming Hu, Tieniu Tan, Liang Wang, and Steve Maybank. A survey on visual surveillance

of object motion and behaviors. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 34(3):334–352, 2004.

[14] Zhan Chaohui, Duan Xiaohui, Xu Shuoyu, Song Zheng, and Luo Min. An improved moving

object detection algorithm based on frame difference and edge detection. In Image and

Graphics, 2007. ICIG 2007. Fourth International Conference on, pages 519–523. IEEE, 2007.

[15] Ismail Haritaoglu, David Harwood, and Larry S. Davis. W¡ sup¿ 4¡/sup¿: real-time surveillance

of people and their activities. Pattern Analysis and Machine Intelligence, IEEE Transactions

on, 22(8):809–830, 2000.