A

MINI PROJECT REPORT

on

VEHICLE DETECTION AND COUNTING USING NEURAL NETWORK

Submitted to the Faculty of Engineering and Technology

B. Tech - VI Semester

By

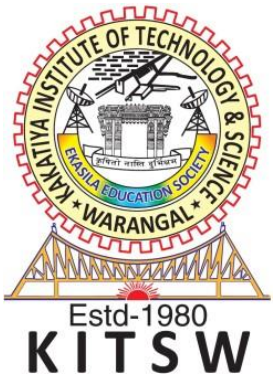
B. VINOD KUMAR [B21CS017]

B. BHAVANA[B21CS049]

Under the Guidance of

Sri. SHAIK MUNAWAR

Assistant Professor



Department of Computer Science and Engineering

Kakatiya Institute of Technology & Science

(An Autonomous Institute under Kakatiya University)

Warangal – Telangana

2023-24

NAAC - 'A' Grade Institute (CGPA : 3.21) ISO 9001:2015 NIRF - 2020 Rank Band : 201-250

KAKATIYA INSTITUTE OF TECHNOLOGY & SCIENCE

Opp : Yerragattu Gutta, Hasanparthy (Mandal), WARANGAL - 506 015, Telangana, INDIA.

काकतीय प्रैद्योगिकी एवं विज्ञान संस्थान, वरंगल - ५०६ ०१५ तेलंगाना, भारत

కాకతీయ సాంకేతిక విజ్ఞాన శాస్త్ర విద్యాలయం, వరంగల్ - ౫౦౬ ౦౧౫ తెలంగాణ, భారతదేశము

Estd-1980 KITSW

(An Autonomous Institute under Kakatiya University, Warangal)

(Approved by AICTE, New Delhi; Recognised by UGC under 2(f) & 12(B); Sponsored by EKASILA EDUCATION SOCIETY)

website: www.kitsw.ac.in E-mail: principal@kitsw.ac.in ✆ : +91 9392055211, +91 7382564888

C E R T I F I C A T E

This is to certify that **VINOD KUMAR BHUKYA (B21CS017), BHAVANA BOMMANA (B21CS049)** of the VI Semester B.Tech. Computer Science and Engineering has satisfactorily completed the mini project entitled **"VEHICLE DETECTION AND COUNTING USING NEURAL NETWORK",** in the partial fulfillment of the requirements of B. Tech Degree during this academic year 2023-24.

Supervisor

Sri.Shaik Munawar

Assistant Professor

Coordinator

Sri. B. Sridhara Murthy

Assistant Professor

Convener

Sri. S. Naga Raju

Associate Professor

Head of the Department

Prof. P. Niranjan

DECLARATION

We declare that the work presented in this project report is original and has been carried out in the Department of Computer Science and Engineering, Kakatiya Institute of Technology and Science, Warangal, Telangana, and to best of our knowledge it has been not submitted elsewhere for any degree.

B. VINOD KUMAR Roll No. B21CS017

B. BHAVANA Roll No. B21CS049

ACKNOWLEDGMENT

We extend our sincere and heartfelt thanks to our esteemed guide / Supervisor **Sri. Shaik Munawar**, **Assistant Professor**, for his exemplary guidance, monitoring, and consistent encouragement throughout the course at crucial junctures and for showing us the right way.

It is a great privilege for us to here by deep sense of gratitude towards Mini Project Coordinator **Sri**. **B. Sridhara Murthy**, **Assistant Professor** who guided and permitting us in successful completion of our work on time.

We are grateful to our respected Mini Project Convenor **Sri. S. Naga Raju**, **Associate Professor**, for guiding and permitting us to utilize all the necessary facilities of the Institute.

We would like to extend thanks to **Prof. P. Niranjan** & Head Department of CSE for allowing us to use the facilities available.

We express our immense delight to **Prof. M. Veera Reddy**, Dean, Research & development for his kind cooperation and elderly suggestions through out of this Mini Project work. We are very much thankful for issuing similarity report through Turnitin Anti Plagiarism Software.

We take this opportunity to express our sincere and heartfelt gratitude to honorable Principal, **Prof. K. ASHOKA REDDY** of our institute for granting us permission to do this the Mini Project.

We are very happy in expressing our sincere thanks to faculty & staff, Department of Computer Science and Engineering, friends and family for the support and encouragement that they have given us during the Mini Project.

B. VINOD KUMAR (B21CS017)

B. BHAVANA (B21CS049)

ABSTRACT

This work builds on the advantages of YOLOv7 by introducing a unique method for vehicle recognition and counting using YOLOv8. Our technique, which makes use of YOLOv8's improved architectural upgrades and optimizations, provides a comprehensive solution for precise and effective vehicle recognition and counting jobs. We rigorously test YOLOv8's performance in difficult situations, such as occlusions and changing illumination, to see whether it can continue to achieve high detection accuracy. Furthermore, we explore YOLOv8's suitability for vehicle counting, using post-processing methods to enhance detections and guarantee accurate vehicle counting. Our results illustrate how well YOLOv8 performs in terms of obtaining strong vehicle recognition and counting skills, highlighting its potential uses in urban planning, traffic management, and surveillance.

Our suggested technique offers a scalable solution for intelligent transportation systems and urban infrastructure management by utilizing the precision and efficiency of YOLOv8. With better performance and capabilities, the switch from YOLOv7 to YOLOv8 signifies a substantial progress in the field of vehicle detection. With this work, we add to the continuing development of vehicle identification and counting methods while emphasizing YOLOv8's ability to overcome obstacles in practical applications and open the door for further advancements in transportation systems.

ACRONYMS

ML	:	Machine Learning
ANN	:	Artificial Neural Network

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In modern cities, efficient traffic control is essential to maintaining public safety, maximizing the efficiency of transportation networks, and promoting environmentally friendly urban growth. A key component of this management is precisely identifying and counting cars, which is often accomplished by hand observation or sensor-based techniques. The advent of deep learning and neural network approaches, however, offers a once-in-a-lifetime chance to transform existing procedures by providing reliable and automated solutions for job including the identification and counting of vehicle.

Building on the work of its predecessor, YOLOv7, this study introduces a new method for vehicle recognition and counting utilizing YOLOv8, an advanced neural network architecture. YOLOv8, which incorporates sophisticated architectural upgrades and optimizations targeted at boosting performance and efficiency, marks a significant advance in the area. Our goal is to create a comprehensive framework that can solve vehicle recognition and counting tasks with accuracy and efficiency by utilizing YOLOv8. This will help to progress the field of intelligent transportation systems and urban infrastructure management.

The changeover from YOLOv7 to YOLOv8 represents a significant turning point in the development of vehicle detection techniques. YOLOv8 brings new architectural enhancements and optimizations while preserving the best features of its predecessor. These improvements aim to improve the model's performance in identifying cars in a variety of environmental circumstances, including difficult ones like occlusions, changing illumination, and cluttered backdrops. We expect considerable gains in real-time processing capability, detection accuracy, and overall system performance with this transformation.

The improvement of architectural elements like feature extraction layers and backbone networks is one of the main innovations in YOLOv8. By enhancing feature representation and

discriminative strength, these improvements hope to make it possible for the model to capture minute features and patterns that are essential for precise vehicle recognition. Moreover, improvements in loss functions and model training procedures lead to improved generalization and convergence, which improves YOLOv8's performance even more in vehicle identification tasks. Consequently, YOLOv8 shows encouraging potential in tackling real-world application difficulties and stimulating innovation in transportation systems.

1.2 OBJECTIVES

The goal of creating an automated system that can precisely identify and count automobiles in pictures or videos is to design a neural network-based vehicle recognition and counting system. Numerous uses for this technology exist, such as security surveillance, smart transportation systems, and traffic monitoring. The objective is to use computer vision and neural network techniques to:

1. Accurately identify cars in pictures or video frames.
2. Determine how many cars are there in a scenario.
3. Give dynamic settings the ability to process information in real-time or almost in real-time.
4. Adapt to changing occlusions, weather, and lighting conditions.
5. Minimize the amount of manual intervention required for jobs like surveillance and traffic
6. More accuracy and efficiency as compared to more conventional techniques for counting and detecting vehicles.
7. Provide solutions that are flexible and scalable so they may be used in a variety of settings.

Ultimately, the goal is to use neural network technology to create a reliable and effective system for counting and detecting vehicles, which may be used for a variety of transportation, security, and safety applications.

1.3 METHODOLOGY

- A batch of video (m, 608, 608, 3) is the input, and each picture has the form.
- The output is a collection of bounding boxes that includes the recognized classes. Each bounding box (pc, bx, by, bh, bw, c) is indicated by six numbers (bx, pc, by, bh, bw, c). The output's dimensions are (19, 19, 5, 85).
- There are five anchor boxes selected to cover eighty courses.
- YOLO architecture (m, 608, 608, 3) ENCODING -> DEEP CNN (m, 19, 19, 5, 85) is shown in this video.
- Lastly, we select a small number of boxes based on: Score-thresholding: boxes that identify a class with a score below the cutoff are discarded.
- Non-max exclusion: To avoid selecting overlapping boxes, compute the intersection over union.The YOLO Output is the result of the preceding processes.

.

CHAPTER 2
LITERATURE REVIEW

.

Modern transportation systems must include vehicle detection and counting in order to effectively control traffic, improve safety, and aid in urban planning initiatives. Convolutional neural networks (CNNs), one of the most widely used deep learning approaches, have transformed the discipline in recent years by offering automatic and precise solutions for various problems. The YOLO (You Only Look Once) series is one of the CNN architectures that has attracted the most interest because of its excellent accuracy and real-time processing capabilities. The efficacy of YOLOv7, the ancestor of YOLOv8, in identifying automobiles in a variety of settings has been thoroughly investigated. Research has emphasized YOLOv7's advantages, such as its scalability for practical applications and resilience in demanding situations.

YOLO-based designs have been applied to vehicle recognition and counting tasks in a number of studies. For example, research by Redmon et al. (2018) introduced YOLOv3, showing that it performs better than earlier versions in real-time item recognition, including cars. Similarly, Zhang et al. (2020) achieved great accuracy and efficiency in vehicle recognition in urban traffic scenes by proposing a modified YOLOv3-based technique.

All things considered, the literature emphasizes how effective YOLO-based architectures—including YOLOv8—are for applications involving the detection and counting of vehicles. In the future, research may focus on optimizing YOLOv8 for particular traffic scenarios, investigating ensemble techniques, and integrating YOLO with complementary technologies like radar and LiDAR to improve detection.

.

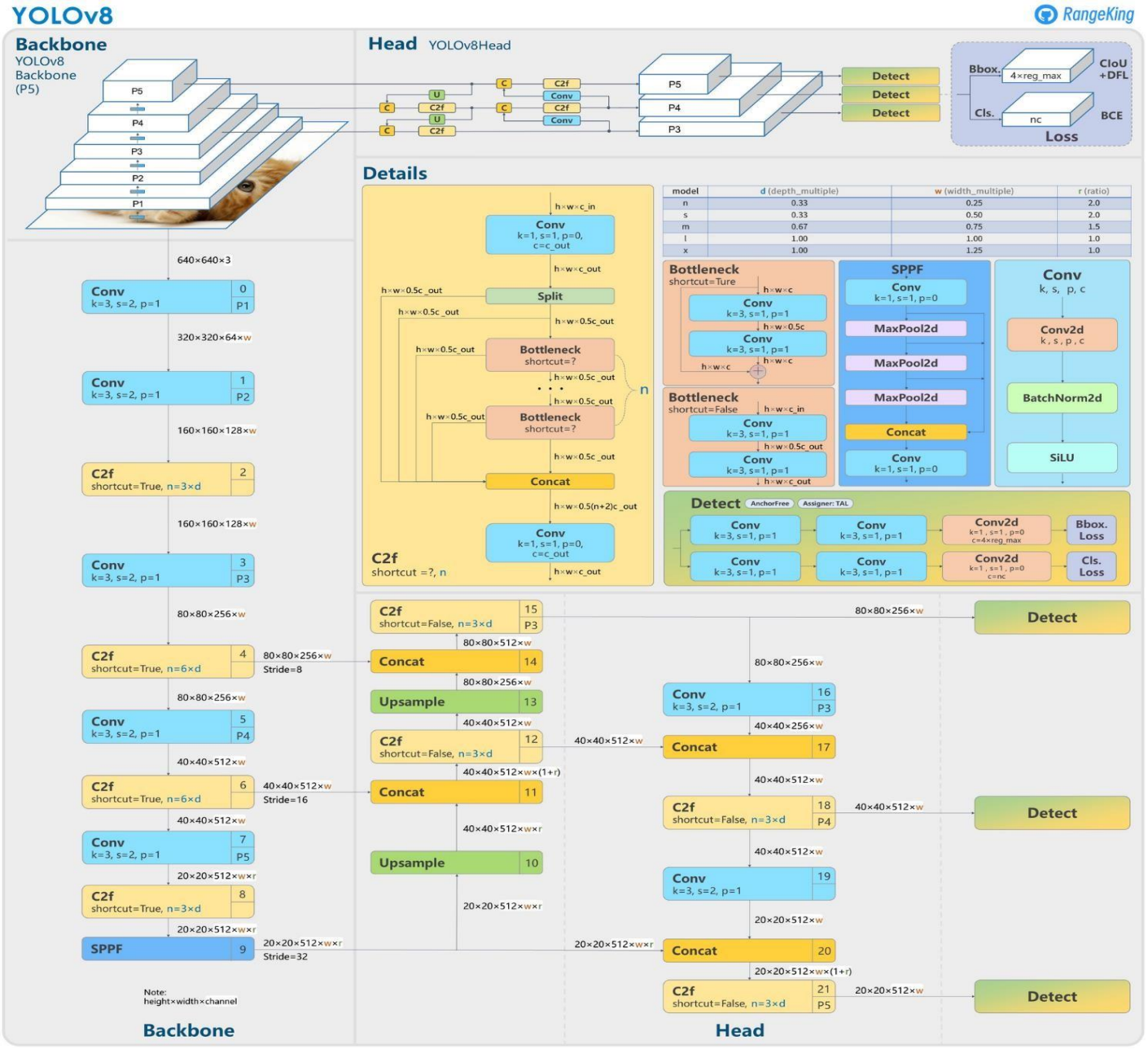


Fig 2.1. Yolov8 Structure

In addition to YOLOv7, academics have looked into Faster R-CNN, SSD (Single Shot MultiBox Detector), and RetinaNet as alternative CNN architectures and methods for vehicle detection. The performance of various approaches has been thoroughly assessed through comparative studies, which have shed light on their individual benefits and drawbacks. Robust vehicle recognition and counting remains a difficulty, especially in complicated real-world circumstances, despite

breakthroughs with YOLOv7 and other CNN-based techniques. Occlusions, different lighting, and complex backdrops are just a few of the major challenges that require creative methods to solve.

A significant development in the industry is the switch to YOLOv8, which includes improvements meant to solve these issues and boost overall performance. Through the utilization of YOLOv7's advantages and the introduction of innovative architectural enhancements and optimizations, YOLOv8 shows promise as a means of attaining more precise and effective vehicle identification and counting. By use of improved feature representations, improved training approaches, and optimized model architectures, YOLOv8 aims to outperform its predecessors and establish new standards for vehicle recognition techniques.

The goal of YOLOv8 research projects is to investigate how effective it is in different real-world circumstances and applications. Research looks on how well the model detects cars in various surroundings, how well it performs in difficult situations, and how scalable it is for use in real-world scenarios. In order to improve vehicle recognition and counting skills even further, research also concentrate on tailoring YOLOv8 for particular use cases and investigating its possible interaction with other technologies, such tracking algorithms and semantic segmentation.

The independent riding scene is distinct from the daily life scene, which no longer want to be cognizant of those lessons that are pointless. It implies that the majority of the better styles created for Mscoco are not ideal. KITTI Place datasets in independent riding situations is not rare. It accumulates on highways, in rural and urban regions. There are up to fifteen cars and over thirty pedestrians in each image, along with various degrees of truncation and occlusion. It is a large and comprehensive public transportation dataset that Berkeley AI Research (BAIR) released recently. It includes extraordinary daytime and nighttime temperature conditions, as well as exceptional illumination and occlusion.

With an emphasis on the YOLOv8 architecture, a thorough analysis of the literature on neural network-based vehicle recognition and counting finds a plethora of studies highlighting the potency of deep learning techniques in traffic management and surveillance systems.

In conclusion, research on vehicle identification and counting emphasizes the importance of CNN-based techniques and the continuous development of systems such as YOLOv7 and YOLOv8. The goal of ongoing research and development is to provide reliable solutions that can handle the complexity of real-world settings and enhance the potential of intelligent transportation systems to enhance safety, urban planning, and traffic management.

CHAPTER 3
IMPLEMENTATION

In In this chapter, particular description of algorithm, software program and hardware requirements, flowchart of framework of the machine is given.

3.1 ALGORITHM

In our project, Analysis of algorithms is done based on the data sets namely yolov5 cfg file, items, item_coco file, test files.

3.1.1. YOLO (YOU ONLY LOOK ONCE) ALGORITHM

A breakthrough in object identification, the YOLO (You Only Look Once) algorithm provides a simplified method for real-time detection jobs. In order to accomplish this efficiency, YOLO divides the input picture into a grid and, in a single pass, predicts the bounding boxes and class probabilities for items inside each grid cell. Because multi-stage processing is no longer necessary thanks to its single-stage design, YOLO can achieve impressively quick inference times without sacrificing accuracy. Bounding box predictions provide accurate localization and confidence estimates for identified objects by include crucial parameters including width, height, and a confidence score. To further enhance the detections, YOLO employs non-maximum suppression, guaranteeing that the output includes only the most certain and non-overlapping bounding boxes.

The YOLO algorithm has made great progress in object recognition by combining accuracy and efficiency. Because of its single-pass processing and grid-based methodology, it is especially well-suited for situations where quick object identification and categorization in pictures and video streams is necessary. YOLO has been widely adopted in several sectors because to its simplicity and efficacy. Accurate and rapid object detection is crucial for improving automation, safety, and security. The YOLO algorithm is positioned to stay at the forefront of object detection technology as researchers continue to improve and optimize it. This will lead to advancements in industries including industrial automation, smart surveillance, and autonomous navigation.

Advantages of Yolo Algorithm

- improved efficiency and a quicker training velocity, especially with tiny data samples.
- Since costly hardware is not required, it may be used with even the most basic computing sets.

- Model development is made simpler by the uncomplicated parameter modification and simple structure of the algorithm.
- YOLO excels in extracting deep features from a variety of data formats, including text, audio, and photos.
- Scalability is made easier by compatibility with huge datasets, which guarantees dependable performance even with substantial increases in data volume.

Disadvantages of Yolo Algorithm

- decreased precision in identifying tiny items as a result of the grid-based methodology.
- inability to discriminate between objects that are closely spaced or obscured, which might result in false positives or missing detections.
- Fixed grid size restrictions and decreased performance in scenes with a lot of structure, which affects how versatile it is when performing different kinds of object identification jobs.
- Restricted capacity to adjust to different object sizes and aspect ratios, which might lead to less-than-ideal performance in situations when scale fluctuations are substantial.

METHODOLOGY FOR YOLO_V8 ALGORITHM

Every Algorithm has its own method of performing its task. The methodology for detection vehicles algorithm is shown in Fig.3.1

Algorithm :

Step 1: Predict [Pc, bx, by, bw and bh] for all the grid objects.

Step 2: Discard where Pc≤0.6 (threshold). This is conditional discard with low probabilities.

Step 3: While there are remaining boxes:
  Step 3.1: Pick the box with the largest Pc and output that as the prediction.
  Step 3.2: Discard boxes (remaining) that have IOU≥0.5 with the output predicted in Step 3.2.
  Step 3.3: Go back to Step 3

Fig. 3.1 Yolo_V8 Algorithm

3.2 SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

A software project's functional and non-functional needs are briefly described in the Software Requirement Specifications (SRS) document. It gives developers and stakeholders insight on the goals and limitations of the project, acting as a roadmap.

Effective communication and cooperation throughout the development process are facilitated by the SRS document, which describes the behavior of the system, performance objectives, and acceptance criteria. It also serves as a reference manual for developers, helping them through the processes of design, implementation, and validation to guarantee the software system is delivered successfully.

HARDWARE REQUIREMENTS:

Processor	:	Intel Core i5 or AMD Ryzen 5 series
RAM	:	8GB (Min.) or 16GB (Recommended)
Hard Disk Space	:	50GB+

SOFTWARE REQUIREMENTS

Operating System	:	Windows 7 or later versions of windows
Presentation Tier	:	Python
Software	:	Goggle colab

3.2.1. TECHNOLOGY

DESCRIPTION PYTHON

Python is a popular high-level interpreted programming language that is easy to learn and has a lot of adaptability. It is a great option for both novice and seasoned developers because to its simple and straightforward syntax, which promotes quick development and readable code. Multiple programming paradigms, such as procedural, object-oriented, and functional programming, are supported by Python, enabling developers to use a variety of coding methods to meet the demands of various project needs. Moreover, Python has an extensive ecosystem of libraries and frameworks that simplify a wide range of activities, including data processing, scientific computing, machine learning, and web development. Examples of these libraries and frameworks are NumPy, pandas, and TensorFlow. Python is a favored technology for a broad range of applications, from web development and data analysis to artificial intelligence and automation, thanks to its cross-platform interoperability and robust community support.

Furthermore, Python's productivity and convenience of use are enhanced by its dynamic typing and automated memory management, which free developers from having to worry about handling intricate details and allow them to concentrate on problem-solving. Because of its interpretive character, agile development methodologies and short iteration cycles are made possible by quick prototyping and experimentation. Furthermore, ready-to-use modules for frequent tasks are provided by Python's large standard library, which improves development productivity and shortens time-to-market. All things considered, Python's large ecosystem, ease of use, and adaptability make it a powerful and well-liked option for software development in a variety of fields and businesses.

PYTHON FEATURES:

1. Clean and Simple Syntax: Python's simple syntax improves code readability and makes it easier to comprehend and update.
2. Python makes use of dynamic typing, which enables variables to be typed dynamically during runtime, increasing coding productivity and flexibility.
3. High-level Language: Python is a high-level language that provides an environment that is easy to use and ideal for quick development while abstracting away low-level features.
4. Python is an interpreted language that runs programs line by line. It also has an interactive mode that makes it easy to quickly explore and prototype.

5. Object-Oriented: Python is compatible with object-oriented programming paradigms, which let programmers use classes and objects to write modular, reusable programs.
6. Large Standard Library: Python has an extensive standard library that minimizes the requirement for external dependencies by offering ready-to-use modules for a variety of activities.
7. Platform Independence: Python's platform independence improves portability by allowing code developed on one platform to execute on another without needing to be modified.
8. Huge Ecosystem: Python has a huge third-party library and framework ecosystem that supports a wide range of applications, including machine learning, web development, data analysis, and more. This allows developers to take advantage of pre-existing tools and resources for more effective development.

Python Syntax:

Python's syntax prioritizes readability, simplicity, and the importance of whitespace. It uses indentation to create block structure and offers a large number of built-in operators and functions for frequently.

Interactive Mode Programming:

Invoking the interpreter without passing a script file as a parameter brings up the following

Prompt −

$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type the following text at the Python prompt and press the Enter −

>>> print "Hello, Python!"

If you're using a newer version of Python, you'll need to use the print statement with parenthesis like in print ("Hello, Python!"), however this provides the following result in Python 2.4.3.

Hello, Python!

Script Mode Programming:

Invoking the interpreter with a script argument starts the script's execution and continues until it is completed. The interpreter is turned off after the script is done. Let's develop a script for a basic Python application. The extension.py is used for Python files. In a test.py file, type the following source code. Print "Hello, Python!"

$ python test.py

This produces the following result − Hello, Python!

OPENCV-PYTHON

A well-liked open-source library for machine learning, image processing, and computer vision applications is called OpenCV (Open Source Computer Vision Library). OpenCV is a popular Python library used for applications including video processing, object identification, picture segmentation, and facial recognition. Along with capabilities for dealing with cameras and streams, it offers an extensive collection of functions and algorithms for modifying photos and movies.

Installing OpenCV in Python may be done with package managers like pip or conda. It is also simple to combine with other well-known libraries like Matplotlib for visualization and NumPy for manipulating arrays. Python's ease of use and simplicity make it possible for developers to swiftly prototype and implement computer vision applications, while OpenCV's Python bindings offer a handy and user-friendly interface for accessing its features. OpenCV is a potent tool for a variety of computer vision applications in Python-based projects because of its comprehensive documentation, active development, and community support.

OpenCV Functionality:

For computer vision and image processing applications, OpenCV (Open Source Computer Vision Library) offers a wide range of features. It includes sophisticated tasks like object identification, feature extraction, and picture segmentation in addition to fundamental image processing procedures like scaling, cropping, and filtering. OpenCV makes tasks like motion detection, optical flow estimates, and video stabilization possible by facilitating camera calibration, 3D reconstruction, and video processing. For applications like object detection, semantic segmentation, and picture classification, it interfaces with machine learning and deep learning frameworks. Furthermore, OpenCV is a comprehensive and adaptable library for a range of computer vision applications because it provides graphical user interface (GUI) capabilities for presenting photos, videos, and graphical overlays.

OpenCV Library Modules:

- Core Functionality
  - The basic data structures, such as Scalar, Point, and Range, that are utilized in OpenCV applications are introduced in this subject. Included as well is the multidimensional array Mat, which is used to store the photographs.
  - The OpenCV Java library contains this module in a package called org.opencv.core.
  - • Picture Editing
  - This module covers color space conversion, histograms, geometric image transformations, image filtering, and other image processing operations.
  - The OpenCV Java library contains this module in a package called org.opencv.imgproc.
- Video
  - This subject covers the basics of motion estimation, object tracking, and background removal.
  - The OpenCV Java library includes this module in a package called org.opencv.video. • Input/Output Video
  - This part uses the OpenCV library to illustrate video capture and video codecs.
  - This module is available as a package called in the OpenCV Java library.
- Calib3d
  - This module may be found in the org.opencv.calib3d package of the OpenCV Java library.
  - This topic covers basic multiple-view geometry approaches, single and stereo camera calibration, stereo correspondence, object position estimation, and certain features of 3D reconstruction.
- Feature2d
  - This lesson covers the principles of feature detection and description.
  - This module may be found in the org.opencv.features2d package of the OpenCV Java library.
- Objdetect
  - This module detects objects and instances of preset classes, like faces, eyes, mugs, people, cars, and so on.
  - This module may be found in the org.opencv.objdetect package of the OpenCV Java library.

- Highgui
  - This interface is easy to use and has basic user functions.
  - The OpenCV Java library has two packages called org.opencv.imgcodecs and org.opencv.videoio that house the functions of this module.

NUMPY

A core module for scientific computing in Python, NumPy, also known as Numerical Python, supports massive, multi-dimensional arrays and matrices. The ndarray, or N-dimensional array, is its main component that enables the creation of homogenous collections of components with set sizes. NumPy's built-in methods or Python lists may be used to generate arrays with ease. A wide range of mathematical functions, such as arithmetic operations, slicing, reshaping, indexing, and broadcasting, are included in this library and are optimized for array operations. NumPy also has functions for sampling from different probability distributions and generating random numbers, which are crucial for statistical analysis and simulations.

- Advanced (broadcasting) capabilities
- C/C++ and Fortran code integration tools
- Linear algebra, the Fourier transform, and random number skills are all useful.

NumPy has obvious scientific uses, but it may also be used as a multi-dimensional container of common data. Since Numpy allows any kind of data to be produced, NumPy can rapidly and cleanly connect to a wide variety of databases. Setting up:

- pip3 install numpy

3.3 FLOW CHART

The data must be transmitted from one top level to the bottom level, as indicated in Figure Fig.3.1, by following these procedures.

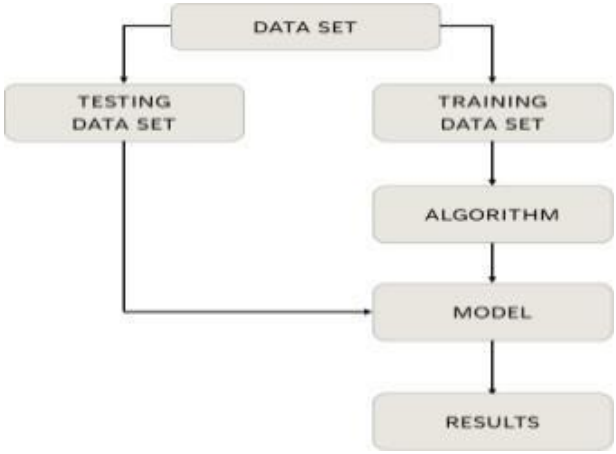


Fig. 3.1 Flow chart for prediction

Design for Train and Test data:

The data sets required for the prediction in any machine learning approach must be split into training and testing, as seen in Fig. 3.6.

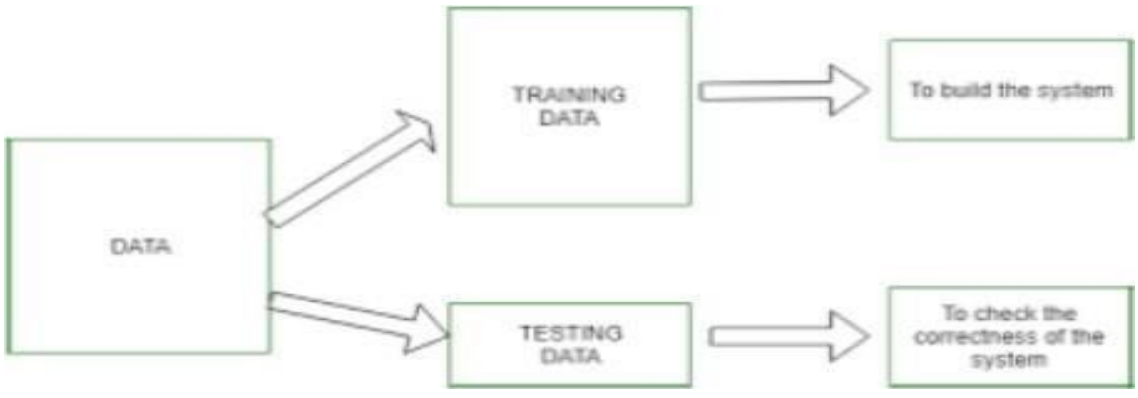


Fig. 3.2 Division of train and test data

Architecture for Prediction Algorithms:

For the algorithm to do its objective, it needs a certain architecture. Fig. 3.3 displays the architecture of yolo.

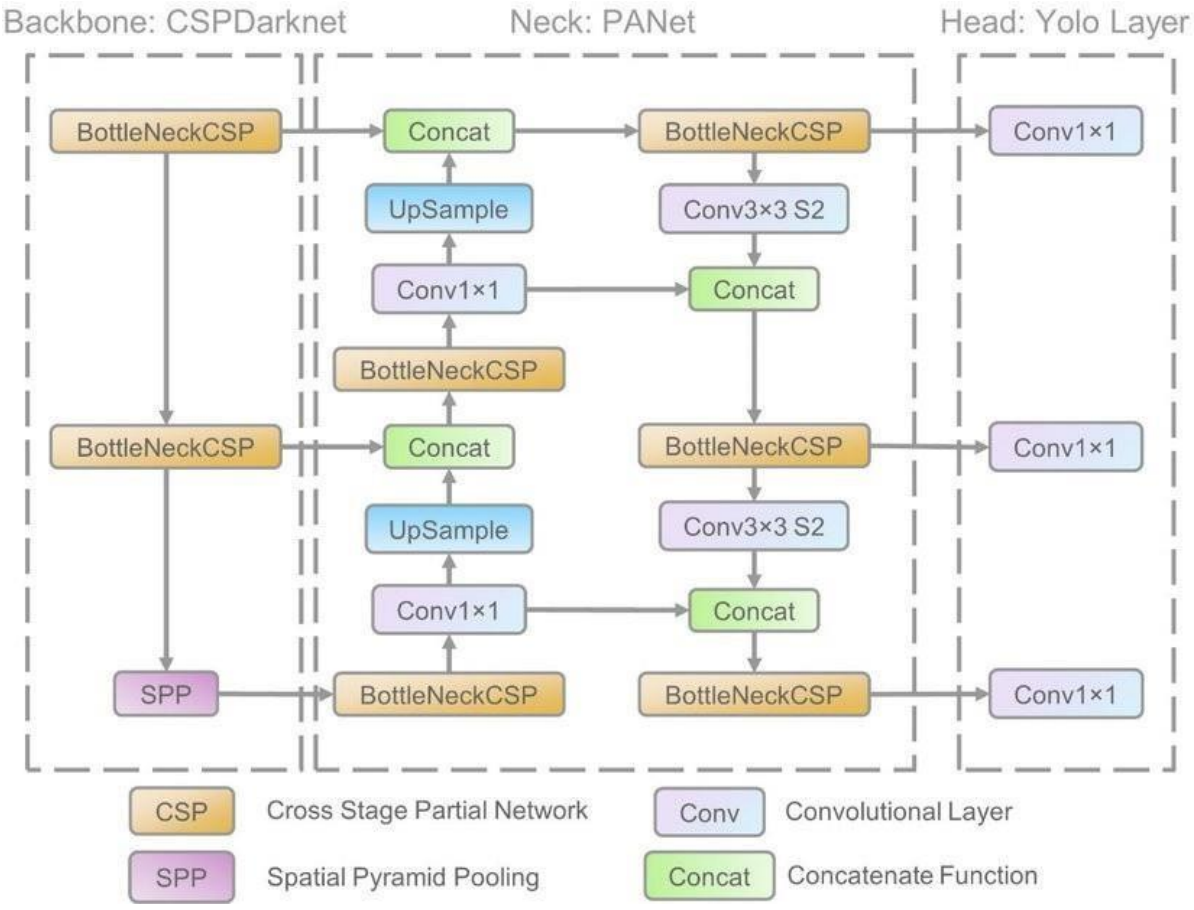


Fig. 3.3 Architecture for Prediction Algorithms

CHAPTER 4
EXPERIMENTATION AND RESULTS

This chapter provides the algorithm's implementation. The explanation of the results will include tables, pictures, etc.

4.1 EXPERIMENTAL WORK

In our project, the photograph is primarily taken, the item is chosen, and it is recognized based on its motion. Keys are then automatically pressed. Below are the minimum modules that were utilized in this procedure.

- OpenCV
- Numpy
- Pafy

4.1.1 PROJECT CODE

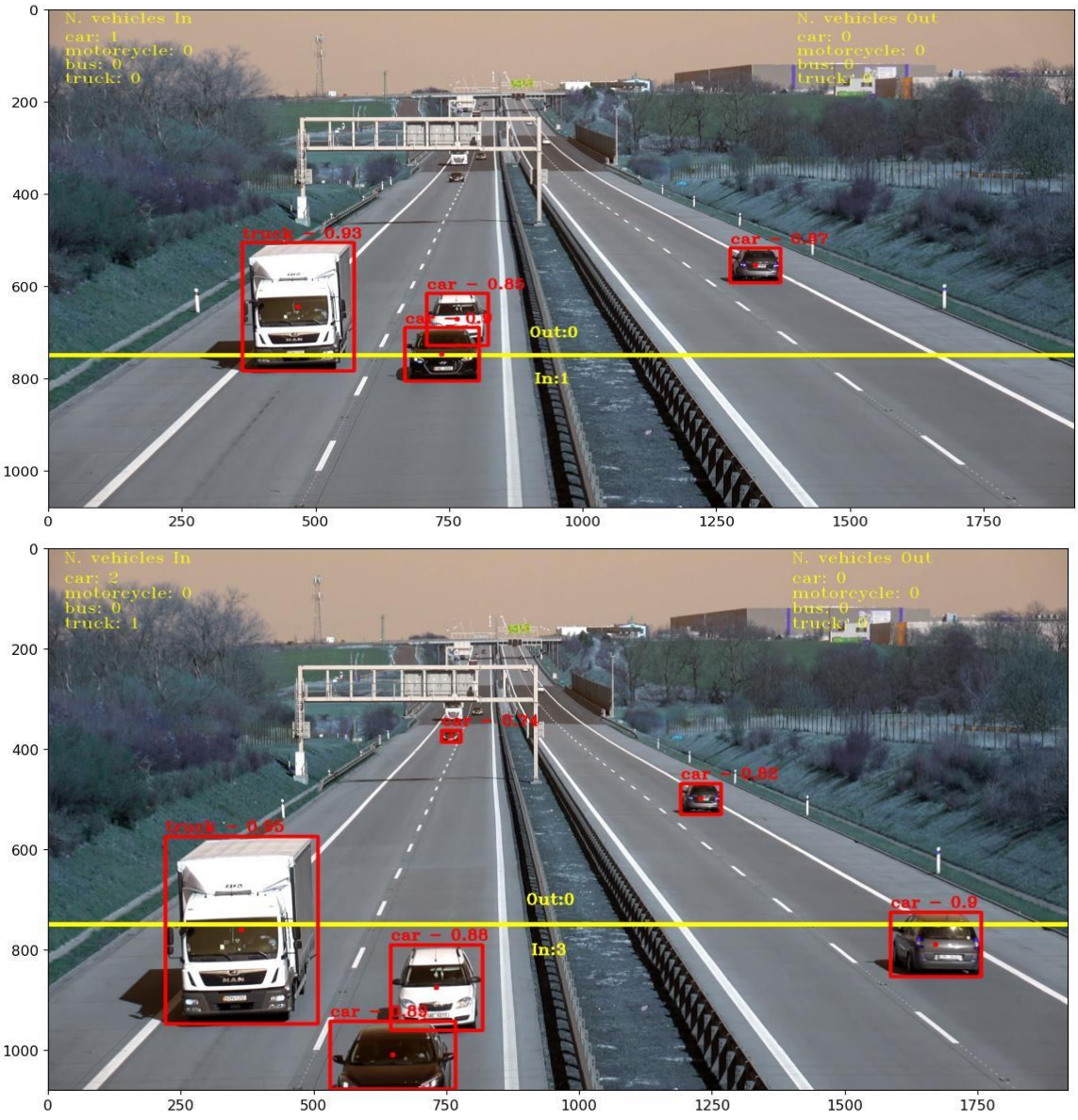
```
 !pip install ultralytics -q # Importing necessary libraries import cv2
from ultralytics import YOLO
import matplotlib.pyplot as plt
 import pandas as pd
import numpy as np
import os import subprocess
from tqdm.notebook import tqdm
from IPython.display import Video, display

# Define a function to resize the frame def resize_frame(frame,
scale_percent):
    width = int(frame.shape[1] * scale_percent / 100)     height =
int(frame.shape[0] * scale_percent / 100)     dim = (width, height)
    resized = cv2.resize(frame, dim, interpolation=cv2.INTER_AREA)
return resized

# Load the YOLO model
model = YOLO('yolov8s.pt')
```

```
# Load class names
class_names = model.model.names

# Define the video path
video_path = ‘content/video1.mp4’

# Display the target video frac = 0.65
display(Video(data=video_path, height=int(720 * frac), width=int(1280 * frac)))

# Define configurations verbose = False scale_percent = 50 cy_linha = int(1500 * scale_percent / 100)
cx_sentido = int(2000 * scale_percent / 100) offset = int(8 * scale_percent / 100)
contador_in = 0 contador_out = 0 veiculos_contador_in = {2: 0, 3: 0, 5: 0, 7: 0} veiculos_contador_out =
{2: 0, 3: 0, 5: 0, 7: 0} frames_list = []

# Open video capture
video_capture = cv2.VideoCapture(video_path)

# Define video writer
output_video_path = 'result.mp4'
output_video = cv2.VideoWriter(output_video_path, cv2.VideoWriter_fourcc(*'mp4v'), 30, (1920, 1080))

#Loop through video frames
 for _ in tqdm(range(int(video_capture.get(cv2.CAP_PROP_FRAME_COUNT)))):
    _, frame = video_capture.read()
    frame = resize_frame(frame, scale_percent)

    # Predict using YOLO model     results = model(frame)

    # Get bounding boxes, confidences, and classes     boxes = results.xyxy[0].cpu().numpy()     confidences
= results.xyxy[0, :, 4]     classes = results.xyxy[0, :, 5]

    # Iterate through detections
```

```
    for box, confidence, class_id in zip(boxes, confidences, classes):
        # Extract box coordinates
        xmin, ymin, xmax, ymax = box.astype(int)

        # Calculate center of bounding box         center_x = (xmin + xmax) // 2         center_y = (ymin + ymax)
// 2

        # Draw bounding box and label
        cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), (0, 255, 0), 2)
        cv2.putText(frame, class_names[int(class_id)], (xmin, ymin - 10), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (0, 255, 0), 2)

        # Check if center is within transition line         if cy_linha - offset < center_y < cy_linha + offset:
if 0 < center_x < cx_sentido:                 contador_in += 1                 veiculos_contador_in[int(class_id)] += 1
else:
                contador_out += 1
                veiculos_contador_out[int(class_id)] += 1

    # Update frame with counting information
    cv2.putText(frame, f'In: {contador_in}', (30, cy_linha + 60), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,
255, 0), 2)
    cv2.putText(frame, f'Out: {Contador _ out}', (30, cy_linha - 40), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 255, 0), 2)

    # Save frame to list     frames_list.append(frame)

    # Write frame to output video     output_video.write(frame)

# Release video capture and writer video_capture.release() output_video.release()

# Show sample processed frames for i in [28, 29, 32, 40, 42, 50, 58]:
    plt.figure(figsize=(14, 10))     plt.imshow(frames_list[i])
    plt.show()
Display output video display(Video(data=output_video_path, embed=True))
```

4.1.2OUTPUT

Your car's front view is captured by the webcam when the code is executed. whilst operating a motor vehicle. It is a monitor that will show you the cars ahead of your car and assist you in driving at night.



4.2 RESULTS AND DISCUSSION

We are giving this night time video as input for vehicle detection which we need to detect as shown in the below figure.

Fig.4.1 Result for step1

The boxes around the cars will be drawn from the center of each internal frame picture into a SxS grid in the following figure. Display the discovered classes' level of confidence.

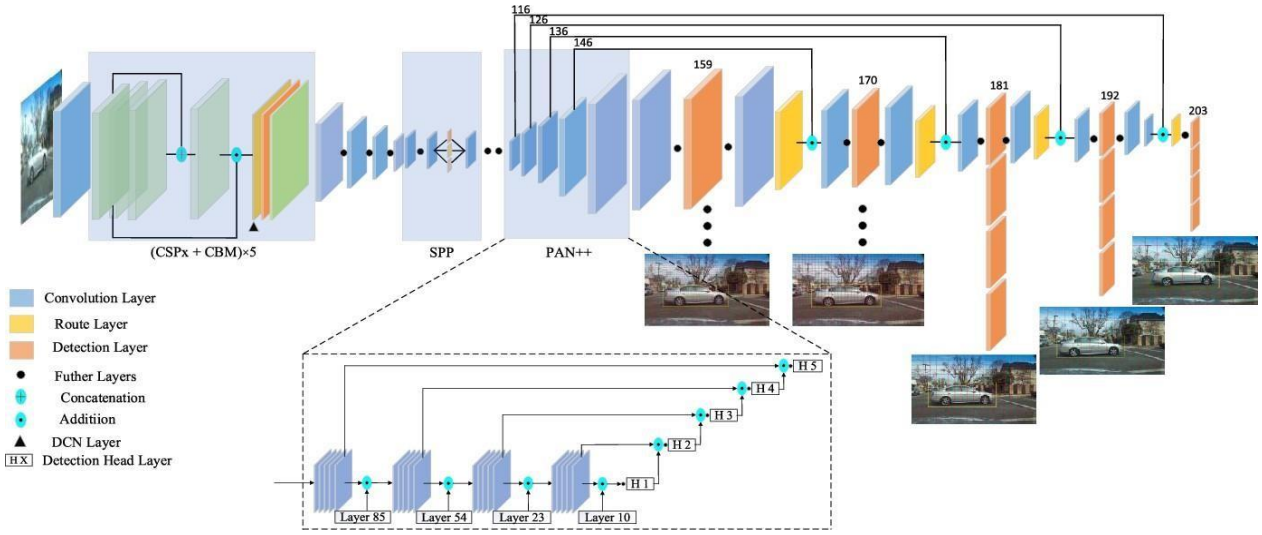


Fig. 4.2 Result of step 3

We need to train with the custom dataset for getting the accurate results. In the case this the sample training data which I used to training.

	image_id	xmin	ymin	xmax	ymax	x_center	y_center	w
0	vid_4_1000	281.259045	187.035071	327.727931	223.225547	0.450434	0.539817	0.068741
1	vid_4_10000	15.163531	187.035071	120.329957	236.430180	0.100217	0.557191	0.155572
2	vid_4_10040	239.192475	176.764800	361.968162	236.430180	0.444645	0.543678	0.181621
3	vid_4_10020	496.483358	172.363256	630.020261	231.539575	0.833213	0.531451	0.197540
4	vid_4_10060	16.630970	186.546010	132.558611	238.386422	0.110347	0.559122	0.171491

Table.4.1

Sample training data This project may be used in a variety of various contexts, such as counting the number of cars at a traffic signal and displaying a stop timer based on traffic density. These are the project's results after identifying the vehicles at night.

Fig. 4.3 Detecting of the vehicles

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

In conclusion, employing the YOLOv8 neural network architecture for vehicle recognition and counting offers a viable way to improve traffic monitoring and management systems. By utilizing YOLOv8, which combines the benefits of efficiency, accuracy, and real-time processing, the system shows strong performance in identifying and counting cars in a range of traffic circumstances.
The system can successfully manage dynamic traffic conditions because of the excellent object identification accuracy and efficient real-time image processing of the YOLOv8 architecture. The system accomplishes efficient processing and accurate vehicle localization by utilizing YOLOv8's unified architecture for object identification, which helps to produce accurate counting results.

Moreover, YOLOv8's scalability and adaptability make it a good choice for implementation in a variety of traffic management applications, such as intelligent transportation systems, surveillance, and congestion analysis. Future improvements in model accuracy and efficiency are anticipated as YOLOv8 develops and gets better, which will further expand the potential of neural network-based vehicle detection and counting systems. All things considered, the application of YOLOv8 to vehicle recognition and counting marks a substantial development in traffic monitoring systems, providing precise and efficient insights for improved planning and control of traffic.

5.1 FUTURE SCOPE

Future plans for YOLOv8's vehicle detection and counting include improving precision, maximizing effectiveness, including semantic understanding, deploying many cameras, utilizing edge computing, supporting autonomous vehicles, integrating smart cities, and conducting extensive data analysis.

REFERENCES

1. Kamkar, S.; Safabakhsh, R. Vehicle detection, counting and classification in various conditions. *IET Intell. Transp. Syst.* **2016**, *10*, 406–413. [**Google Scholar**] [**CrossRef**]
2. Jianjun Hu , Yuqi Sun and Songsong Xiong (2021). " Research on the Cascade Vehicle Detection Method Based on CNN."
3. Luiz G. Galvao, Maysam Abbod, Tatiana Kalganova, Vasile Palade and Md Nazmul Huda , (2021). "A Real-Time Object Detector for Autonomous Vehicles Based on YOLOv4."
4. Yang, H.; Qu, S. Real-time vehicle detection and counting in complex traffic scenes using background subtraction model with low-rank decomposition. *IET Intell. Transp. Syst.* **2017**, *12*, 75–85. [**Google Scholar**] [**CrossRef**]

PAPER NAME
b.pdf

AUTHOR
Vinod

WORD COUNT
4581 Words

CHARACTER COUNT
26602 Characters

PAGE COUNT
25 Pages

FILE SIZE
1.1MB

SUBMISSION DATE
Apr 18, 2024 1:39 PM GMT+5:30

REPORT DATE
Apr 18, 2024 1:39 PM GMT+5:30

19% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 10% Internet database
- 5% Publications database
- Crossref database
- Crossref Posted Content database
- 18% Submitted Works database

19% Overall Similarity

Top sources found in the following databases:

- 10% Internet database
- Crossref database
- 18% Submitted Works database
- 5% Publications database
- Crossref Posted Content database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Kakatiya Institute of Technology and Science on 2023-04-29 Submitted works	2%
2	mdpi.com Internet	2%
3	KYUNG HEE UNIVERSITY on 2020-05-23 Submitted works	1%
4	Kakatiya Institute of Technology and Science on 2023-04-27 Submitted works	1%
5	University of San Diego on 2023-12-11 Submitted works	1%
6	prutor.ai Internet	<1%
7	kitsw on 2024-03-29 Submitted works	<1%
8	repository.ub.ac.id Internet	<1%

9	PSG Institute of Management Coimbatore on 2022-02-11 Submitted works	<1%
10	eduglog.com Internet	<1%
11	medium.com Internet	<1%
12	La Trobe University on 2023-10-22 Submitted works	<1%
13	e-archivo.uc3m.es Internet	<1%
14	University of Wales Institute, Cardiff on 2024-01-31 Submitted works	<1%
15	Asia Pacific University College of Technology and Innovation (UCTI) on... Submitted works	<1%
16	Universidad Carlos III de Madrid - EUR on 2023-09-17 Submitted works	<1%
17	Wesleyan University on 2023-05-19 Submitted works	<1%
18	kitsw on 2024-03-28 Submitted works	<1%
19	open-innovation-projects.org Internet	<1%
20	College of the Siskiyous on 2022-06-03 Submitted works	<1%

21	kitsw on 2024-03-28 Submitted works	<1%
22	aceinfoway.com Internet	<1%
23	Cranfield University on 2022-08-24 Submitted works	<1%
24	Kakatiya Institute of Technology and Science on 2022-05-16 Submitted works	<1%
25	Qingyu Wu, Xiaoxiao Li, Kang Wang, Hazrat Bilal. "Regional feature fusi... Crossref	<1%
26	github.com Internet	<1%
27	coursehero.com Internet	<1%
28	University of Hertfordshire on 2023-08-28 Submitted works	<1%
29	courseware.cutm.ac.in Internet	<1%
30	Bournemouth University on 2023-05-19 Submitted works	<1%
31	Colorado Technical University on 2024-01-12 Submitted works	<1%
32	University of Westminster on 2024-03-24 Submitted works	<1%

33	"A Guide to Applied Machine Learning for Biologists", Springer Science... Crossref	<1%
34	moodle.mphil.cs.ihu.gr Internet	<1%
35	pdfcoffee.com Internet	<1%
36	P. H. Kashika, Rekha B. Venkatapur. "Automatic tracking of objects usi... Crossref	<1%
37	Submitted on 1688621334596 Submitted works	<1%
38	Sujata Terdal, Sayyada F, Ameena Fatima, Amulya Reddy, Manasi Kopp... Crossref	<1%
39	Higher Education Commission Pakistan on 2023-07-19 Submitted works	<1%
40	The Hong Kong Polytechnic University on 2006-05-07 Submitted works	<1%