

A Mini Project Report  
On  
**Real Time Vehicle Counting and Classification System using  
Darknet Framework**

*Submitted to CMREC (UGC Autonomous), Affiliated to JNTUH  
In Partial Fulfillment of the requirements for the Award of Degree of*

**BACHELOR OF TECHNOLOGY  
IN  
COMPUTER SCIENCE AND ENGINEERING (AI&ML)**

Submitted By

<b>S. SRIPAL REDDY</b>	<b>(218R1A6651)</b>
<b>MD.AIJAZ AHMED</b>	<b>(218R1A6635)</b>
<b>S. SATHWIK</b>	<b>(218R1A6650)</b>
<b>S. PRAVAS SAGAR</b>	<b>(218R1A6653)</b>

*Under the Esteemed guidance of*

**Mr. G. Venkateswarlu**

**Professor, Department of CSE (AI & ML)**



**Department of Computer Science & Engineering (AI&ML)**

**CMR ENGINEERING COLLEGE**

**UGC AUTONOMOUS**

(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad, Kandlakoya,  
Medchal Road, R.R. Dist. Hyderabad-501 401)

**2024-2025**

# **CMR ENGINEERING COLLEGE**

## **UGC AUTONOMOUS**

*(Accredited by NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)*

*Kandlakoya, Medchal Road, Hyderabad-501 401*

### **Department of Computer Science & Engineering (AI & ML)**



### **CERTIFICATE**

**This is to certify that the project entitled “Real Time Vehicle Counting and Classification System using Darknet Framework”**

is a Bonafide work carried out by

**S. SRIPAL REDDY** (218R1A6651)

**MD.AIJAZ AHMED** (218R1A6635)

**S. SATHWIK** (218R1A6650)

**S. PRAVAS SAGAR** (218R1A6653)

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING (AI&ML)** from **CMR Engineering College**, under our guidance and supervision.

The results presented in this project have been verified and are found to be satisfactory. The results embodied in this project have not been submitted to any other university for the award of any other degree or diploma.

**Internal Guide**  
**Mr. G. Venkateswarlu**  
Professor  
CSE (AI&ML)

**Project Coordinator**  
**Mr. G. Venkateswarlu**  
Assistant Professor  
CSE (AI&ML)

**Head of the Department**  
**Dr. M. Kumara Swamy**  
Professor & HOD  
CSE (AI&ML)

## **DECLARATION**

**This is to certify that the work reported in the present Mini project entitled “Real Time Vehicle Counting and Classification System using Darknet Framework”**

” is a record of bonafide work done by us in the Department of Computer Science and Engineering (AI&ML), CMR Engineering College. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who sharesimilar interests to improve the project in the future.

The results embodied in this Mini project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

<b>S. SRIPAL REDDY</b>	<b>(218R1A6651)</b>
<b>MD.AIJAZ AHMED</b>	<b>(218R1A6635)</b>
<b>S. SATHWIK</b>	<b>(218R1A6650)</b>
<b>S. PRAVAS SAGAR</b>	<b>(218R1A6653)</b>

## **ACKNOWLEDGMENT**

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. M. Kumara Swamy**, HOD, **Department of CSE (AI & ML), CMR Engineering College** for their constant support.

We are extremely thankful to **Mr. G. Venkateswarlu**, Assistant Professor, Internal Guide, Department of CSE (AI & ML), for his constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We thank **Mr. G. Venkateswarlu** Mini Project Coordinator for his constant support in carrying out the project activities and reviews.

We express our thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, We are very much thankful to our parents who guided us for every step.

<b>S. SRIPAL REDDY</b>	<b>(218R1A6651)</b>
<b>MD.AIJAZ AHMED</b>	<b>(218R1A6635)</b>
<b>S. SATHWIK</b>	<b>(218R1A6650)</b>
<b>S. PRAVAS SAGAR</b>	<b>(218R1A6653)</b>

# **Real Time Vehicle Counting and Classification System using Darknet Framework**

## **Abstract**

In today's world, accurately counting and classifying vehicles in real-time has become a critical task for effective traffic management, surveillance, and transportation systems. It plays a crucial role in optimizing road infrastructure, enhancing safety measures, and making informed decisions for traffic planning. With the ever-increasing traffic congestion and road safety concerns, the demand for a robust and automated vehicle counting and classification system has grown significantly. Traditionally, vehicle counting, and classification involved manual deployment of sensors or fixed cameras at specific locations. However, these methods had limitations in handling complex traffic scenarios, especially in real-time, and were less efficient in dealing with varying environmental conditions, occlusions, and different vehicle types. Fortunately, recent advancements in deep learning models have revolutionized object detection, making real-time vehicle counting and classification achievable. One such model is the YOLO (You Only Look Once) algorithm based on the Darknet framework. Leveraging the power of this model, a real-time vehicle counting, and classification system has been developed, utilizing the OpenCV library. The system employs a pretrained YOLO model to detect the number of vehicles present in a given video and classifies the type of each vehicle. By doing so, it eliminates the need for extensive human intervention and ensures automated and accurate counting of vehicles in real-time. Moreover, this system excels in handling varying traffic conditions and different vehicle types, which enhances its accuracy and reliability. The benefits of this proposed system are numerous. It provides valuable data for traffic analysis, enabling better traffic management strategies and improved infrastructure planning. With this system in place, authorities can efficiently address traffic congestion, implement targeted safety measures, and optimize traffic flow. Further, the integration of the YOLO algorithm within the Darknet framework in the proposed system has opened new possibilities for real-time traffic management. By leveraging deep learning, this system offers a reliable and efficient solution to the challenges posed by modern traffic scenarios, helping to create safer and more organized road networks for everyone.

# CHAPTER 1

## INTRODUCTION

### 1.1 Overview

Real-time vehicle counting and classification is a cutting-edge technology and system designed for monitoring and analyzing traffic on roads, highways, and other transportation networks. This technology leverages various sensors, cameras, and computer vision algorithms to capture, analyze, and categorize vehicles as they pass through designated monitoring points. The primary goal of real-time vehicle counting and classification is to provide critical data and insights to transportation authorities, urban planners, and traffic management systems. Here is an overview of this technology in a detailed paragraph:

Real-time vehicle counting and classification systems utilize a network of strategically positioned cameras and sensors equipped with advanced image processing and machine learning capabilities. These devices continuously capture video or images of the traffic flow in real-time, allowing for the automatic detection and tracking of vehicles. Through computer vision algorithms, the system can identify and classify vehicles based on their type, such as cars, trucks, buses, motorcycles, and more. It can also estimate vehicle speed, direction, and lane occupancy.

The applications of real-time vehicle counting and classification are diverse and impactful. Firstly, it provides transportation authorities with critical traffic data, including traffic volume, congestion patterns, and vehicle composition. This data is invaluable for optimizing traffic flow, making informed decisions about road maintenance and expansion, and enhancing road safety measures. Furthermore, these systems are essential for toll collection, parking management, and traffic law enforcement. By automatically categorizing vehicles, authorities can apply appropriate toll rates, manage parking spaces efficiently, and enforce rules and regulations effectively.

Additionally, real-time vehicle counting and classification play a crucial role in intelligent transportation systems (ITS). ITS integrates data from these systems to facilitate dynamic traffic management, improve traffic signal timing, and enhance the overall transportation experience for commuters. In smart cities, this technology is an integral part of efforts to reduce

traffic congestion, minimize environmental impact, and promote sustainable urban development.

So, real-time vehicle counting and classification is a sophisticated technology that enhances traffic management, infrastructure planning, and road safety. It provides real-time insights into traffic conditions, allowing authorities to make data-driven decisions and optimize transportation networks for the benefit of both commuters and the environment. As urbanization continues to grow, the adoption of these systems becomes increasingly important for creating efficient and sustainable transportation systems in modern cities.

## **1.2 Research Motivation**

The research motivation for real-time vehicle counting and classification systems stems from the pressing need to address numerous challenges and opportunities associated with urban transportation and traffic management. Several key factors drive the motivation for conducting research in this field:

- **Traffic Congestion and Urbanization:** The rapid urbanization of cities around the world has led to an exponential increase in vehicular traffic. Traffic congestion is a significant problem in urban areas, leading to increased travel times, fuel consumption, and air pollution. Researchers are motivated to find innovative solutions to alleviate congestion and improve traffic flow.
- **Safety and Accident Prevention:** Traffic accidents remain a global concern, causing injuries and fatalities. Real-time vehicle counting and classification systems can contribute to road safety by monitoring traffic conditions, detecting incidents, and providing early warnings to authorities and drivers. Research in this area aims to develop intelligent systems that can help prevent accidents and save lives.
- **Efficient Resource Allocation:** Efficient resource allocation is crucial for managing transportation infrastructure effectively. By accurately counting and classifying vehicles in real-time, transportation authorities can allocate resources more efficiently. For example, toll collection, parking space management, and road maintenance can be optimized based on the type and volume of vehicles.
- **Environmental Impact:** Vehicular emissions are a significant contributor to air pollution and greenhouse gas emissions. Research in real-time vehicle counting and classification

can contribute to environmental sustainability by providing data to support the development of eco-friendly transportation policies. This includes promoting public transportation, electric vehicles, and alternative fuels.

- **Intelligent Transportation Systems (ITS):** The integration of advanced technologies, including real-time vehicle counting and classification, is a key component of intelligent transportation systems. Researchers are motivated to develop and enhance ITS to create smarter, more responsive transportation networks. These systems can lead to reduced travel times, improved fuel efficiency, and a more seamless commuting experience.
- **Data-Driven Decision-Making:** The availability of real-time traffic data is essential for informed decision-making by transportation authorities and urban planners. Research in this area focuses on improving the accuracy and reliability of data collected from vehicle counting and classification systems. High-quality data is fundamental for optimizing traffic management strategies.
- **Smart City Initiatives:** Many cities worldwide are embracing the concept of smart cities, where data-driven technologies play a central role in improving urban living. Real-time vehicle counting and classification align with smart city goals by providing real-time insights into transportation patterns, enabling responsive traffic management, and enhancing overall quality of life for residents.

### 1.3 Problem Statement

The problem statement for real-time vehicle counting and classification systems revolves around the numerous challenges and shortcomings that exist within urban transportation and traffic management. While these systems offer immense potential, several critical issues and limitations need to be addressed:

- **Inadequate Traffic Data:** One of the fundamental problems in traffic management is the lack of comprehensive and real-time traffic data. Traditional data collection methods, such as manual traffic counts or fixed sensors, are often limited in scope and do not provide continuous, fine-grained information. This data insufficiency hampers the ability of authorities to make informed decisions about traffic management, infrastructure investment, and public transportation planning.



- **Traffic Congestion:** Urban traffic congestion continues to escalate, leading to increased travel times, fuel consumption, and productivity losses. The problem statement revolves around finding effective solutions to mitigate congestion by understanding traffic patterns and bottlenecks. Real-time data on vehicle counts and classifications are essential for devising strategies to alleviate congestion.
- **Road Safety:** Traffic accidents remain a significant concern, with numerous injuries and fatalities occurring annually. The problem lies in the ability to promptly detect and respond to road safety issues. Real-time vehicle counting and classification systems can play a pivotal role in accident prevention by identifying hazardous situations, reckless driving behaviors, and potential collisions.
- **Resource Allocation Inefficiency:** The allocation of resources, such as toll booths, parking spaces, and road maintenance crews, often lacks optimization due to a lack of real-time data. This leads to inefficiencies, resource wastage, and increased costs. Addressing this problem requires accurate and up-to-date information on traffic volume, vehicle types, and usage patterns.
- **Environmental Impact:** Urban transportation contributes significantly to environmental pollution and greenhouse gas emissions. Reducing this impact necessitates a shift toward eco-friendly transportation options and policies. Real-time data on vehicle types and emissions can inform policymakers and city planners about the environmental footprint of different vehicles and drive the adoption of cleaner alternatives.
- **Data Quality and Integration:** The problem of data quality and integration is central to the effective use of real-time traffic data. Inconsistent data sources, data accuracy issues, and interoperability challenges often hinder data-driven decision-making. Researchers and practitioners must work on standardizing data formats, ensuring data accuracy, and integrating data from diverse sources into a unified platform.
- **Traffic Management in Smart Cities:** As cities aspire to become smarter, the challenge lies in integrating real-time vehicle counting and classification systems into broader smart city initiatives. Developing systems that seamlessly integrate with other urban infrastructure and technologies is essential for realizing the full potential of smart cities.

## 1.4 Applications

Real-time vehicle counting and classification systems have a wide range of applications across various domains due to their ability to provide accurate and timely information about traffic patterns and vehicle types. Here are detailed explanations of some of the key applications for this technology:

- **Traffic Management and Optimization:** Real-time vehicle counting and classification systems are foundational for traffic management. They enable traffic authorities to monitor congestion, identify bottlenecks, and implement dynamic traffic management strategies. These systems help optimize traffic signal timing, lane management, and the allocation of traffic resources to improve overall traffic flow and reduce congestion.
- **Road Safety and Accident Prevention:** These systems play a crucial role in enhancing road safety. By continuously monitoring traffic and identifying hazardous situations, such as speeding or erratic driving behavior, they can provide early warnings to authorities and trigger immediate responses to prevent accidents. The data collected can also be used to analyze accident patterns and implement targeted safety measures.
- **Toll Collection:** Real-time vehicle counting and classification are essential for automated toll collection systems. Toll booths equipped with these technologies can accurately determine toll charges based on vehicle types and the distance traveled. This improves the efficiency of toll collection and reduces congestion at toll plazas.
- **Parking Management:** In urban areas, efficient parking management is a significant challenge. These systems help manage parking facilities by monitoring occupancy and providing real-time information to drivers about available parking spaces. This reduces the time spent searching for parking, lowers emissions, and enhances the overall parking experience.
- **Public Transportation Planning:** Understanding traffic patterns and vehicle types is crucial for public transportation planning. These systems provide data that helps authorities optimize public transportation routes, schedules, and capacity to better meet the needs of commuters. This leads to more efficient and sustainable public transit systems.

- **Environmental Impact Assessment:** Real-time vehicle counting and classification enable assessments of the environmental impact of different types of vehicles. By categorizing vehicles by emissions and fuel types, these systems support environmental policies aimed at reducing pollution and promoting cleaner transportation options.
- **Law Enforcement:** Law enforcement agencies utilize this technology to enforce traffic laws and regulations. Automated license plate recognition (ALPR) systems, often integrated with real-time vehicle counting and classification, assist in identifying and apprehending vehicles involved in criminal activities.
- **Intelligent Transportation Systems (ITS):** These systems are a fundamental component of intelligent transportation systems (ITS). ITS integrates data from real-time vehicle counting and classification systems with other technologies such as traffic signals, variable message signs, and traffic cameras to create responsive and adaptive transportation networks.
- **Urban Planning and Development:** Urban planners use traffic data to inform decisions about road expansion, infrastructure development, and city planning. Real-time data helps assess the impact of new developments on traffic flow and ensures that urban planning aligns with future transportation needs.
- **Research and Data Analysis:** Researchers in transportation and urban studies rely on real-time vehicle data for a wide range of studies, including traffic modeling, urban development research, and transportation policy analysis. The availability of high-quality, real-time data enhances the accuracy and reliability of research findings.

## **CHAPTER 2**

### **LITERATURE SURVEY**

Alpatov et al. considered road situation analysis tasks for traffic control and ensuring safety. The following image processing algorithms are proposed: vehicle detection and counting algorithm, road marking detection algorithm. The algorithms are designed to process images obtained from a stationary camera. The developed vehicle detection and counting algorithm was implemented and tested also on an embedded platform of smart cameras.

Song et al. proposed a vision-based vehicle detection and counting system. A new high-definition highway vehicle dataset with a total of 57,290 annotated instances in 11,129 images is published in this study. Compared with the existing public datasets, the proposed dataset contains annotated tiny objects in the image, which provided the complete data foundation for vehicle detection based on deep learning.

Neupane et al. created a training dataset of nearly 30,000 samples from existing cameras with seven classes of vehicles. To tackle P2, this trained and applied transfer learning-based fine-tuning on several state-of-the-art YOLO (You Only Look Once) networks. For P3, this work proposed a multi-vehicle tracking algorithm that obtains the per-lane count, classification, and speed of vehicles in real time.

Lin et al. presented a real-time traffic monitoring system based on a virtual detection zone, Gaussian mixture model (GMM), and YOLO to increase the vehicle counting and classification efficiency. GMM and a virtual detection zone are used for vehicle counting, and YOLO is used to classify vehicles. Moreover, the distance and time traveled by a vehicle are used to estimate the speed of the vehicle. In this study, the Montevideo Audio and Video Dataset (MAVD), the GARM Road-Traffic Monitoring data set (GRAM-RTM), and our collection data sets are used to verify the proposed method.

Chauhan et al. used the state-of-the-art Convolutional Neural Network (CNN) based object detection models and train them for multiple vehicle classes using data from Delhi roads. This work gets upto 75% MAP on an 80-20 train-test split using 5562 video frames from four different locations. As robust network connectivity is scarce in developing regions for continuous video transmissions from the road to cloud servers, this work also evaluated the

latency, energy and hardware cost of embedded implementations of our CNN model-based inferences.

Arinaldi et al. presented a traffic video analysis system based on computer vision techniques. The system is designed to automatically gather important statistics for policy makers and regulators in an automated fashion. These statistics include vehicle counting, vehicle type classification, estimation of vehicle speed from video and lane usage monitoring. The core of such system is the detection and classification of vehicles in traffic videos. This work implemented two models for this purpose, first is a MoG + SVM system and the second is based on Faster RCNN, a recently popular deep learning architecture for detection of objects in images.

Gomaa et al. presented an efficient real-time approach for the detection and counting of moving vehicles based on YOLOv2 and features point motion analysis. The work is based on synchronous vehicle features detection and tracking to achieve accurate counting results. The proposed strategy works in two phases; the first one is vehicle detection and the second is the counting of moving vehicles. For initial object detection, this work has utilized state-of-the-art faster deep learning object detection algorithm YOLOv2 before refining them using K-means clustering and KLT tracker. Then an efficient approach is introduced using temporal information of the detection and tracking feature points between the framesets to assign each vehicle label with their corresponding trajectories and truly counted it.

Oltean et al. proposed an approach for real time vehicle counting by using Tiny YOLO for detection and fast motion estimation for tracking. This application is running in Ubuntu with GPU processing, and the next step is to test it on low-budget devices, as Jetson Nano. Experimental results showed that this approach achieved high accuracy at real time speed (33.5 FPS) on real traffic videos.

Pico et al. proposed the implementation of a low-cost system to identify and classify vehicles using an Embedded ARM based platform (ODROID XU-4) with Ubuntu operating system. The algorithms used are based on the Open-source library (Intel OpenCV) and implemented in Python programming language. The experimentation carried out proved that the efficiency of the algorithm implemented was 95.35%, but it can be improved by increasing the training sample.

Tituana et al. reviewed different previous works developed in this area and identifies the technological methods and tools used in those works; in addition, this work also presented the trends in this area. The most relevant articles were reviewed, and the results were summarized in tables and figures. Trends in the used methods are discussed in each section of the present work.

Khan et al. aimed of this work is that a cost-effective vision-based vehicle counting and classification system that is mainly implemented in OpenCV utilising Python programming and some methods of image processing.

Balid et al. reported on the development and implementation of a novel smart wireless sensor for traffic monitoring. Computationally efficient and reliable algorithms for vehicle detection, speed and length estimation, classification, and time-synchronization were fully developed, integrated, and evaluated. Comprehensive system evaluation and extensive data analysis were performed to tune and validate the system for a reliable and robust operation.

Jahan et al. presented convolutional neural network for classifying four types of common vehicle in our country. Vehicle classification plays a vital role of various application such as surveillance security system, traffic control system. This work addressed these issues and fixed an aim to find a solution to reduce road accident due to traffic related cases. To classify the vehicle, this work used two methods feature extraction and classification. These two methods can straight forwardly be performed by convolutional neural network.

Butt et al. proposed a convolutional neural network-based vehicle classification system to improve robustness of vehicle classification in real-time applications. This work presented a vehicle dataset comprising of 10,000 images categorized into six-common vehicle classes considering adverse illuminous conditions to achieve robustness in real-time vehicle classification systems. Initially, pretrained AlexNet, GoogleNet, Inception-v3, VGG, and ResNet are fine-tuned on self-constructed vehicle dataset to evaluate their performance in terms of accuracy and convergence. Based on better performance, ResNet architecture is further improved by adding a new classification block in the network.

Gonzalez et al. showed a vision-based system to detect, track, count and classify moving vehicles, on any kind of road. The data acquisition system consists of a HD-RGB camera placed on the road, while the information processing is performed by clustering and classification

algorithms. The system obtained an efficiency score over the 95 percent in test cases, as well, the correct classification of 85 percent of the test objects.

## CHAPTER 3

### EXISTING SYSTEM

The system is divided in several stages: ROI selection, detection of moving objects, clustering process, tracking, single-frame classification and counting. In Fig 3.1, the global overview of the proposed vehicle detection system is shown.

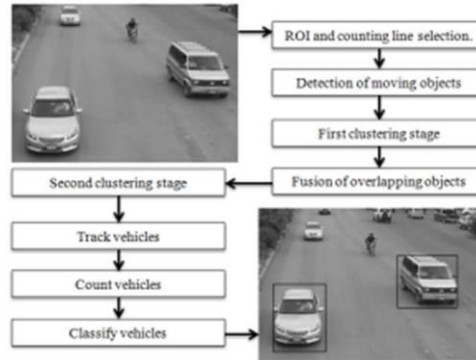


Fig. 3.1: Global overview of the existing vehicle detection system.

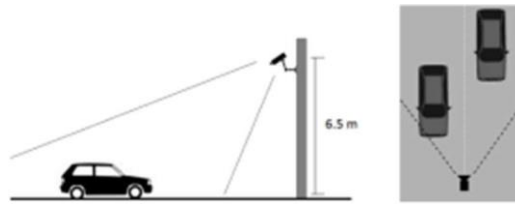


Fig. 3.2: The configuration of the existing hardware is shown.

Acquire images from the road is the first step of the proposed vehicle detection algorithm, for this is necessary to place a HD-RGB camera over the road, like is shown in Fig 3.2. All obtained frames are converted to grayscale.

Then, in order to maximize the performance of the algorithm, a selection of a proper ROI (Region of Interest) is needed. Within this area the search for moving objects is performed. The region must contain the zone where the aim traffic flow is located, as well the line for vehicle count. This selection may change according to the type of road.



To detect moving vehicles a temporal difference is applied, this technique consists in obtain the pixel value difference between two consecutive frames, but in the present paper the comparison is not made pixel to pixel. The ROI is divided in  $n$  square segments with same length  $k$ , for each  $n$ -square their mean  $m1$  of grayscale level is computed, and then it is compared with the average of the previous frame  $m0$ . A  $n$ -square has a object in motion if the mean of the  $n$ -square from the present frame has a larger difference than the one of the previous frame plus a threshold:  $m1 > m0 + th$  or  $m1 < m0 - th$ . If these conditions are met, the assessed  $n$ -square is marked and tagged as “M”, like is shown in figure 3 (a) and (b) respectively.

In the next step it is needed to cluster the multiple regions that contain movement, to consolidate the different moving objects “MO” in scene. Then, all labeled “M” squares that are continuous are merged into a Temporal Object “TOn”, a number is assigned to any “TOn” as it is showed in Fig 3.3 (c).

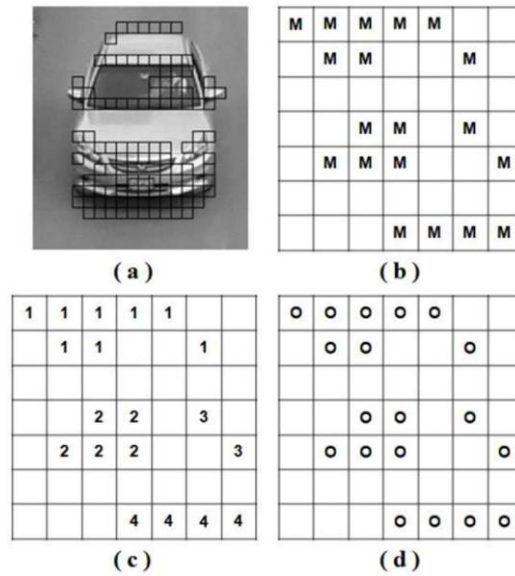


Fig. 3.3: The clustering process of the  $n$ -square is shown. (a) Detected motion inside the ROI. (b) Squares tagged with movement “Sm”. (c) Continuous “Sm” squares are merged into a temporal object “TOn”. (d) Result of the first clustering process.

It is necessary to do another clustering process, in which, a “TOn” object is grouped with all their nearest neighbors to form one, now tagged as Refined Object “RO”, Fig 3.3 (d), this process is repeated with any “TOn”.

All “RO” objects that overlap with another, figure 4 (a), are merged together, now on called as Moving Object “MO”. The remaining “RO” objects are evaluated according to their color and size, to decide whether or not they belong to an “MO” (b) When a frame X has been analyzed and all “MO” s have been consolidated, the appropriate data of those objects are stored to be able to track them in the next frame X+1. With all the “MO” s constituted in frame X+1, these are verified with the stored objects from frame X, if the data match, is the same object, otherwise these unmatched “MO”.

### **Drawbacks of existing system**

- A small change in the data can cause a large change in the structure of the decision tree causing instability.
- For a Decision tree sometimes, calculation can go far more complex compared to other algorithms.
- Decision tree often involves higher time to train the model.
- Decision tree training is relatively expensive as the complexity and time has taken are more.
- The Decision Tree algorithm is inadequate for applying regression and predicting continuous values.

## CHAPTER 4

### PROPOSED SYSTEM

The research work begins with the acquisition of real-time video streams as the primary input source, typically sourced from surveillance cameras or traffic monitoring systems. The first module, Background Subtraction and ROI, plays a crucial role in isolating moving objects, i.e., vehicles, from the stationary background. This is achieved using advanced algorithms to create a Region of Interest (ROI) that narrows down the area for subsequent analysis, reducing computational load and minimizing false positives. The heart of the system lies in the Vehicle Detection and Tracking module, which employs the YOLO (You Only Look Once) deep learning model. YOLO excels at real-time object detection and tracking, enabling the system to identify vehicles within the defined ROI and track their movements across frames, allowing for continuous monitoring. The next step involves the Vehicle Classification module, powered by the Darknet framework, which classifies the detected vehicles into specific categories, such as cars, trucks, or motorcycles. Following this, the Counting and Analytics module quantifies vehicle movements, including counting, speed measurement, and other relevant analytics, providing valuable data for traffic management and research purposes. Finally, the system generates a video output that overlays processed information onto the original video feed, offering a user-friendly visual representation of the vehicle counting and classification results, making it a versatile tool for various applications in traffic monitoring, urban planning, and transportation research. Figure 4.1 shows the proposed system model. The detailed operation illustrated as follows:

**Step 1:** This is the starting point of system. Acquire real-time video streams as input data for the system. These video streams could come from surveillance cameras, traffic cameras, or any source capturing vehicle movements.

**Step 2:** Background subtraction is a crucial step for isolating moving objects (vehicles) from the stationary background. In this module, use algorithms and techniques to detect the background and create a Region of Interest (ROI) where vehicle detection will occur. This step helps reduce noise and focus on the relevant area.

**Step 3:** Use YOLO (You Only Look Once), a deep learning-based object detection model, for vehicle detection. YOLO can efficiently detect and locate objects in real-time video frames. It

identifies the vehicles within the defined ROI and can track their movements across frames, allowing us to follow vehicles as they move through the video.

**Step 4:** After detecting and tracking vehicles, further analyze and classify them using the Darknet framework. Darknet is a neural network framework well-suited for classification tasks. It can classify vehicles into different categories such as cars, trucks, motorcycles, or any other relevant classes.

**Step 5:** In this step, count and analyze the detected and classified vehicles. We can track the number of vehicles passing through specific points or regions of interest, calculate vehicle speed, and gather other relevant analytics data. This information can be useful for traffic management, surveillance, or research purposes.

**Step 6:** Finally, the system provides video output with the processed information overlaid on the original video feed. This output can include counted vehicles, their classifications, and any other relevant data. It allows users to visualize and interpret the results of the vehicle counting and classification system.

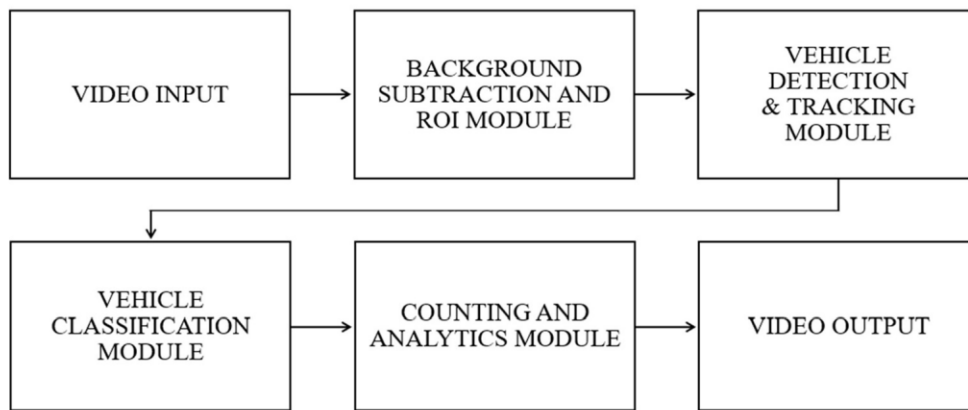


Fig. 4.1: Block diagram of proposed system.

#### 4.1 Background Subtraction

This is the first module in the system whose main purpose is to learn about the background in a sense that how it is different from the foreground. Furthermore, as proposed system works on a video feed, this module extracts the frames from it and learns about the background. In a traffic scene captured with a static camera installed on the roadside, the moving objects can be considered as the foreground and static objects as the background. Image processing algorithms are used to learn about the background using the above-mentioned technique.

## 4.2 Vehicle Detection and Counting

The third and the last module in the proposed system is classification. After applying foreground extraction module, proper contours are acquired, Features of these contours such as centroid. Aspect ratio, area, size and solidity are extracted and are used for the classification of the vehicles. This module consists of three steps, background subtraction, image enhancement and foreground extraction. Background is subtracted so that foreground objects are visible. This is done usually by static pixels of static objects to binary 0. After background subtraction image enhancement techniques such as noise filtering, dilation and erosion are used to get proper contours of the foreground objects. The result obtained from this module is the foreground.

**Region of Interest selection:** In the very first frame of the video, I define a ROI by drawing a close line on the image. The goal is to recognize that ROI in a later frame, but that ROI is not a salient vehicle. It is just a part of a vehicle, and it can deform, rotate, translate and even not be fully in the frame.

**Vehicle Detection:** Active strategy to choose a search window for vehicle detection using an image context was proposed GMM framework to capture the vehicle by sequential actions with top-down attention. It has achieved satisfactory performance on vehicle detection benchmark, by sequentially refining the bounding boxes. Proposed a sequential search strategy to detect visual vehicles in images, where the detection model was trained by proposed a deep RL framework to select a proper action to capture a vehicle in an image.

**Vehicle Counting:** In this module detected vehicles will be counted and these counted results will be updated frequents based on vehicle detection, results will be printed streaming video using OpenCV.

## 4.4 YOLO-V3 Model

Object detection is a phenomenon in computer vision that involves the detection of various objects in digital images or videos. Some of the objects detected include people, cars, chairs, stones, buildings, and animals. This phenomenon seeks to answer two basic questions:

1. What is the object? This question seeks to identify the object in a specific image.
2. Where is it? This question seeks to establish the exact location of the object within the image.

Object detection consists of various approaches such as fast R-CNN, Retina-Net, and Single-Shot MultiBox Detector (SSD). Although these approaches have solved the challenges of data limitation and modeling in object detection, they are not able to detect objects in a single algorithm run. YOLO algorithm has gained popularity because of its superior performance over the aforementioned object detection techniques.

**YOLO Definition:** YOLO is an abbreviation for the term ‘You Only Look Once’. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects. This means that prediction in the entire image is done in a single algorithm run. CNN is used to predict various class probabilities and bounding boxes simultaneously. The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO and YOLOv3.

**Importance of YOLO:** YOLO algorithm is important because of the following reasons:

- Speed: This algorithm improves the speed of detection because it can predict objects in real-time.
- High accuracy: YOLO is a predictive technique that provides accurate results with minimal background errors.
- Learning capabilities: The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

**YOLO algorithm working:** YOLO algorithm works using the following three techniques:

- Residual blocks
- Bounding box regression
- Intersection Over Union (IOU)

**Residual blocks:** First, the image is divided into various grids. Each grid has a dimension of  $S \times S$ . The following Figure 2 shows how an input image is divided into grids. In the Figure 2, there are many grid cells of equal dimension. Every grid cell will detect objects that appear

within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

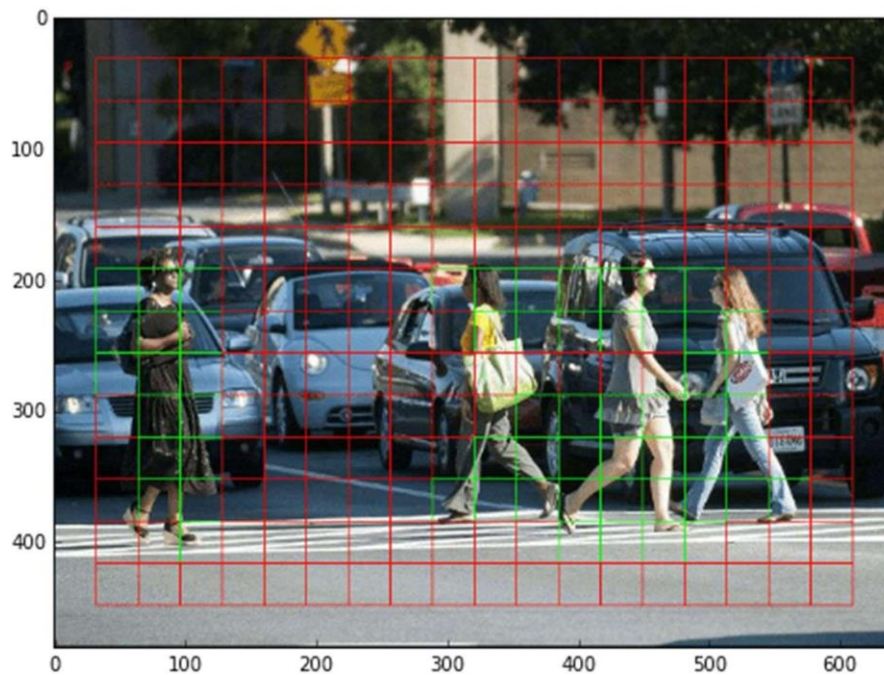


Figure 4.2. Example of residual blocks.

**Bounding box regression:** A bounding box is an outline that highlights an object in an image. Every bounding box in the image consists of the following attributes:

- Width (bw)
- Height (bh)
- Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.
- Bounding box center (bx,by)

The following Figure 3 shows an example of a bounding box. The bounding box has been represented by a yellow outline. YOLO uses a single bounding box regression to predict the height, width, center, and class of objects. In the image above, represents the probability of an object appearing in the bounding box.

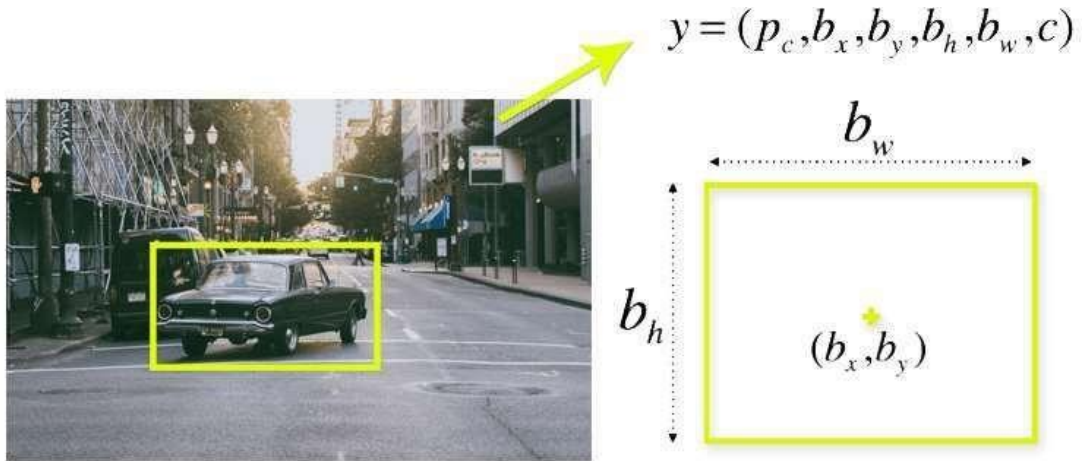


Figure 3. Bounding box regression

**Intersection over union (IOU):** Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly. Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.

**Combination of the three techniques:** The following image shows how the three techniques are applied to produce the final detection results.

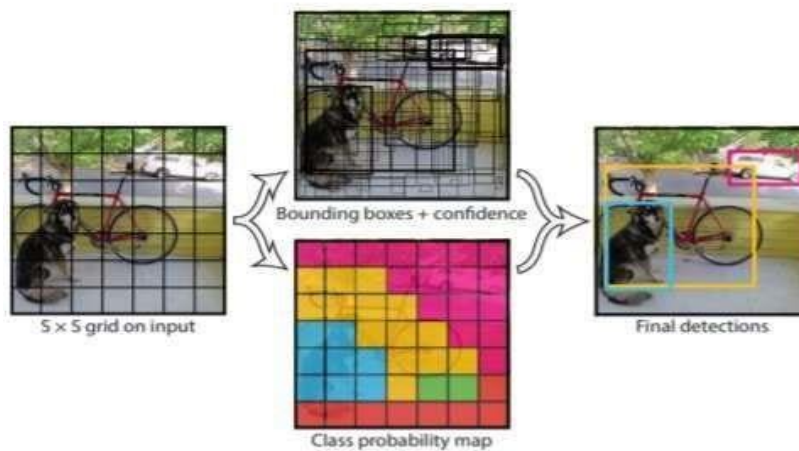
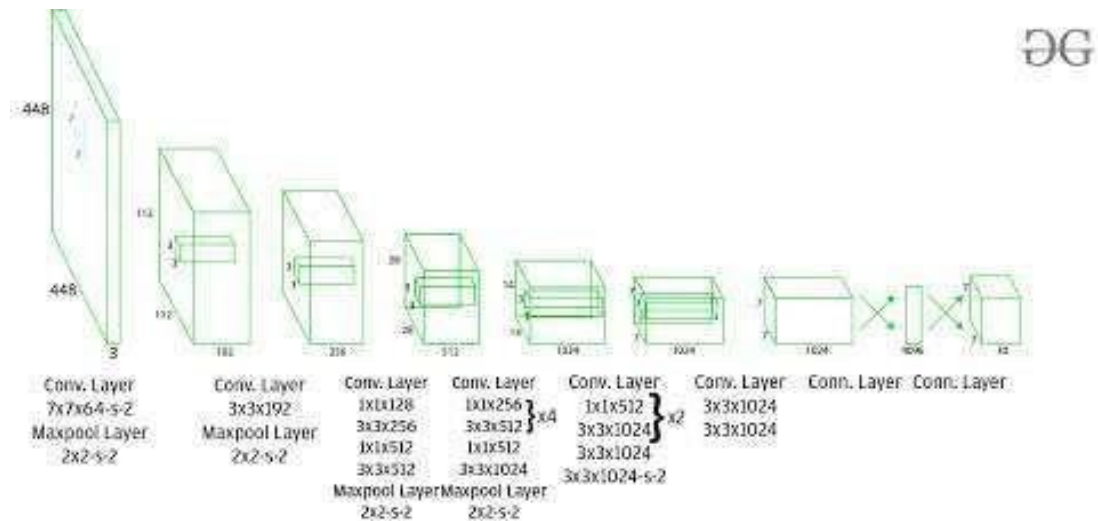


Figure 4. Combination of three modules.

First, the image is divided into grid cells. Each grid cell forecasts B bounding boxes and provides their confidence scores. The cells predict the class probabilities to establish the class of each object. For example, we can notice at least three classes of objects: a car, a dog, and a



bicycle. All the predictions are made simultaneously using a single convolutional neural network. Intersection over union ensures that the predicted bounding boxes are equal to the real boxes of the objects. This phenomenon eliminates unnecessary bounding boxes that do not meet the characteristics of the objects (like height and width). The final detection will consist of unique bounding boxes that fit the objects perfectly. For example, the car is surrounded by the pink bounding box while the bicycle is surrounded by the yellow bounding box. The dog has been highlighted using the blue bounding box.



The first 20 convolution layers of the model are pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to perform detection since previous research showcased that adding convolution and connected layers to a pre-trained network improves performance. YOLO's final fully connected layer predicts both class probabilities and bounding box coordinates.

YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.

One key technique used in the YOLO models is **non-maximum suppression (NMS)**. NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection. In object detection, it is common for multiple bounding boxes to be generated for a single object in an image. These bounding boxes may overlap or be located at different positions, but they all represent the same object. NMS is used to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in the image.

#### 4.5 CNN Layers

According to the facts, training and testing of CNN involves in allowing every source data via a succession of convolution layers by a kernel or filter, rectified linear unit (ReLU), max pooling, fully connected layer and utilize SoftMax layer with classification layer to categorize the objects with probabilistic values ranging from. According to the facts, training and testing of -CNN involves in allowing every source image via a succession of convolution layers by a kernel or filter, rectified linear unit (ReLU), max pooling, fully connected layer and utilize SoftMax layer with classification layer to categorize the objects with probabilistic values ranging from  $[0,1]$ . Convolution layer as depicted in Figure 8 is the primary layer to extract the features from a source image and maintains the relationship between pixels by learning the features of image by employing tiny blocks of source data. It’s a mathematical function which considers two inputs like source image  $\mathbb{I}(\mathbb{I}, \mathbb{I}, \mathbb{I})$  where  $\mathbb{I}$  and  $\mathbb{I}$  denotes the spatial coordinates i.e., number of rows and columns.  $\mathbb{I}$  is denoted as dimension of an image (here  $\mathbb{I} = 3$ , since the source image and a filter or kernel with similar size of input image and can be denoted as  $\mathbb{I}(\mathbb{I}_x, \mathbb{I}_y, \mathbb{I})$ .

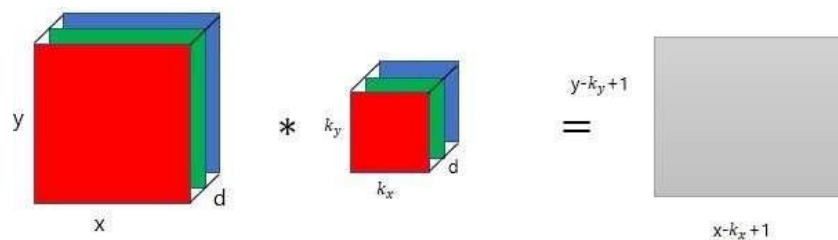


Figure 8: Representation of convolution layer process.

The output obtained from convolution process of input image and filter has a size of  $\mathbb{N} ((\mathbb{N}_x - \mathbb{N}_k + 1), (\mathbb{N}_y - \mathbb{N}_k + 1), 1)$ , which is referred as feature map. An example of convolution procedure is demonstrated in Figure 5.2. Let us assume an input image with a size of  $5 \times 5$  and the filter having the size of  $3 \times 3$ . The feature map of input image is obtained by multiplying the input image values with the filter values as given in Figure 9.

$$\begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array}$$

5x5 image                      3x3 kernel

(a)

$$\begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 3 & 3 & 3 \\ \hline 2 & 2 & 3 \\ \hline 3 & 2 & 3 \\ \hline \end{array}$$

5x5 image                      3x3 kernel                      Feature map

(b)

Figure 9: Example of convolution layer process (a) an image with size  $\mathbb{N} \times \mathbb{N}$  is convolving with  $\mathbb{N} \times \mathbb{N}$  kernel (b) Convolved feature map.

#### 4.5.1 ReLU layer

Networks those utilizes the rectifier operation for the hidden layers are cited as rectified linear unit (ReLU). This ReLU function  $\mathbb{R}(\cdot)$  is a simple computation that returns the value given as input directly if the value of input is greater than zero else returns zero. This can be represented as mathematically using the function  $\mathbb{R}(\cdot)$  over the set of 0 and the input  $\mathbb{N}$  as follows:

$$\mathbb{R}(\mathbb{N}) = \max\{0, \mathbb{N}\}$$

#### 4.5.2 Max pooling layer

This layer mitigates the number of parameters when there are larger size images. This can be called as subsampling or down sampling that mitigates the dimensionality of every feature map by preserving the important information. Max pooling considers the maximum element from the rectified feature map.

### 4.5.3 SoftMax classifier

Generally, as seen in the above picture SoftMax function is added at the end of the output since it is the place where the nodes are meet finally and thus, they can be classified. Here,  $X$  is the input of all the models and the layers between  $X$  and  $Y$  are the hidden layers and the data is passed from  $X$  to all the layers and Received by  $Y$ . Suppose, we have 10 classes, and we predict for which class the given input belongs to. So, for this what we do is allot each class with a particular predicted output. Which means that we have 10 outputs corresponding to 10 different class and predict the class by the highest probability it has.

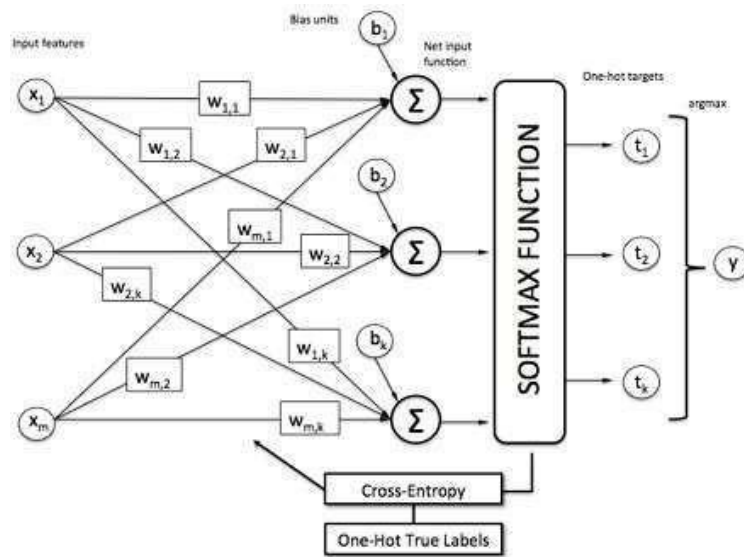


Figure 10: Emotion prediction using SoftMax classifier.

In Figure 11, and we must predict what is the object that is present in the picture. In the normal case, we predict whether the emotion is A. But in this case, we must predict what is the object that is present in the picture. This is the place where softmax comes in handy. As the model is already trained on some data. So, as soon as the picture is given, the model processes the pictures, send it to the hidden layers and then finally send to softmax for classifying the picture. The softmax uses a One-Hot encoding Technique to calculate the cross-entropy loss

and get the max. One-Hot Encoding is the technique that is used to categorize the data. In the previous example, if softmax predicts that the object is class A then the One-Hot Encoding for:

Class A will be [1 0 0]

Class B will be [0 1 0]

Class C will be [0 0 1]

From the diagram, we see that the predictions are occurred. But generally, we don't know the predictions. But the machine must choose the correct predicted object. So, for machine to identify an object correctly, it uses a function called cross-entropy function. So, we choose more similar value by using the below cross-entropy formula.

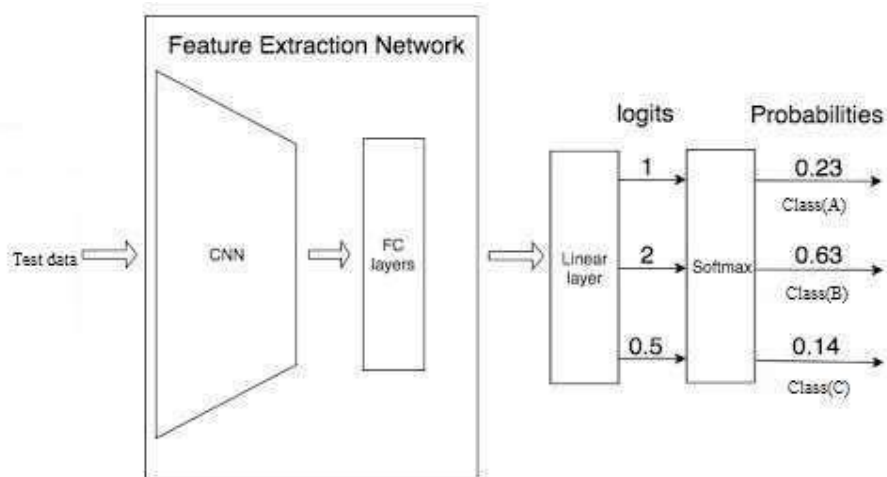


Figure 11: Example of SoftMax classifier.

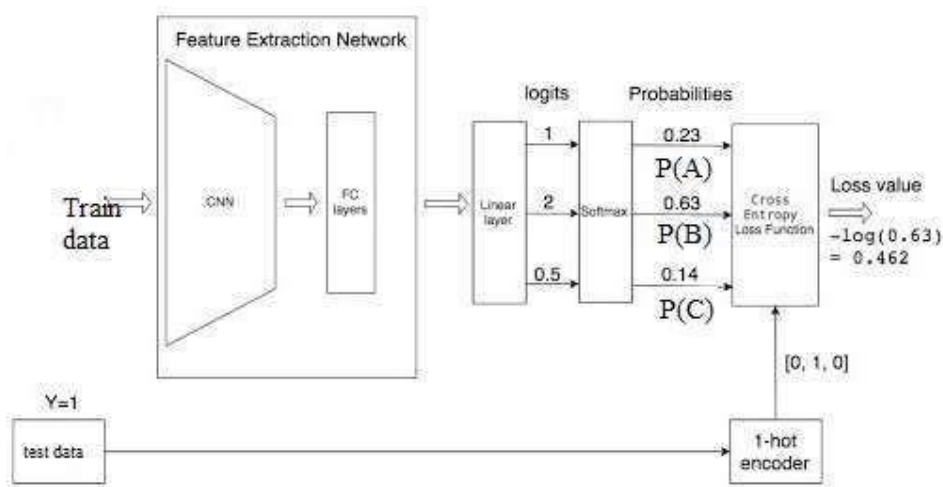


Figure 12: Example of SoftMax classifier with test data.

In the above example we see that 0.462 is the loss of the function for class specific classifier. In the same way, we find loss for remaining classifiers. The lowest the loss function, the better the prediction is. The mathematical representation for loss function can be represented as: -

$$L_{class} = -\log(\text{softmax}(W_{class} * \text{flatten}(\text{output\_map})))$$

## 4.6 Advantages

The procedure outlined above for image testing, preprocessing, person detection, and helmet detection using YOLOv3 offers several distinct advantages in various applications. Firstly, it provides a robust and efficient approach to object detection in images. YOLOv3, known for its real-time capabilities, enables the simultaneous detection of multiple objects within an image. This efficiency is particularly beneficial in scenarios where quick responses are critical, such as security surveillance, industrial safety, or traffic management.

Secondly, the image preprocessing step enhances the quality and consistency of the input data, leading to improved detection accuracy. Resizing the image to a standardized format ensures that YOLOv3 can work consistently across different image sizes. Normalizing pixel values and color channel adjustments help in reducing noise and ensuring that the model generalizes well to various lighting conditions.

Furthermore, the procedure's modularity allows for flexibility in customizing the object detection task. By using pre-trained models, users can easily adapt the system to their specific needs, whether it's person detection, helmet detection, or other object recognition tasks. This adaptability extends to different types of images, making it suitable for diverse applications. Another advantage lies in the post-processing step, where the results from person and helmet detection can be integrated and analyzed together. This integration facilitates the identification of safety violations, such as individuals not wearing helmets in a hazardous environment. Such insights are valuable for safety monitoring and compliance enforcement in industries like construction, manufacturing, or sports.

Additionally, the ability to generate reports and trigger alerts based on the detection results adds a layer of automation and decision support. In security systems, for instance, immediate alerts can be sent to personnel when unauthorized persons are detected, enhancing the overall security posture.

# CHAPTER 5

## UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

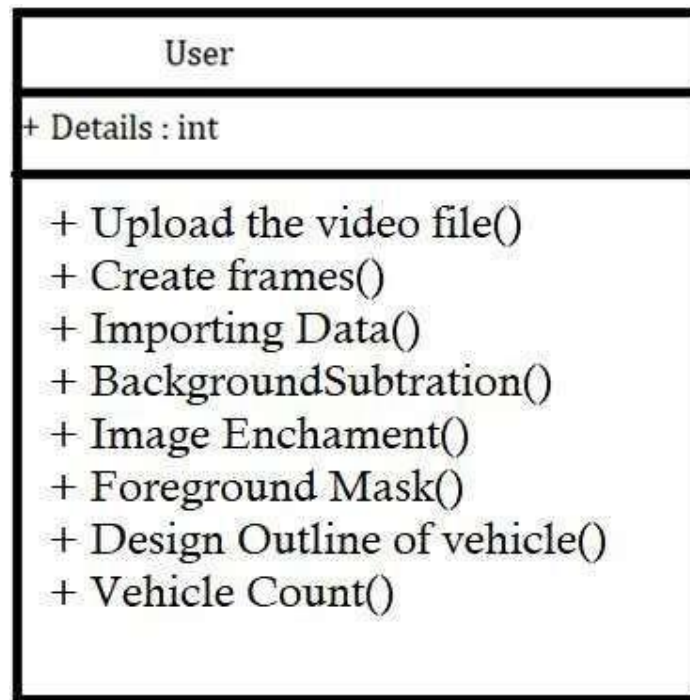
**GOALS:** The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

### **Class diagram**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain

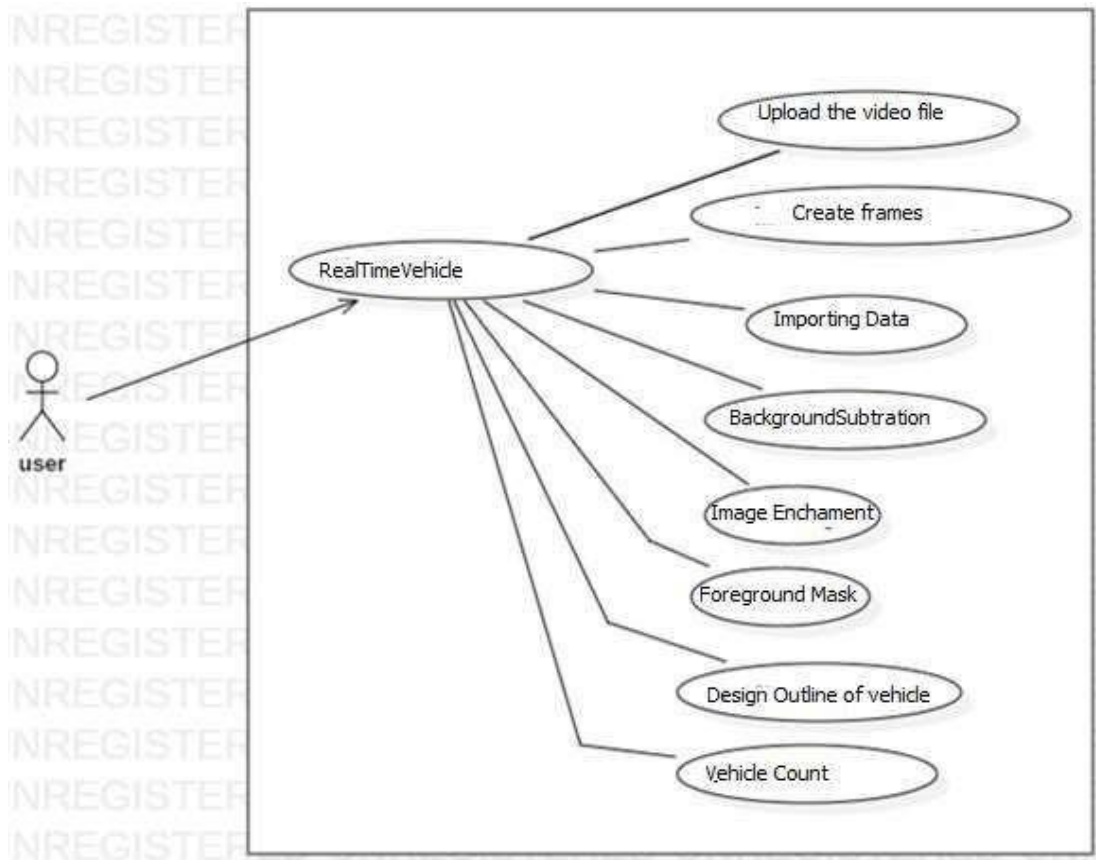
functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.



### Use case Diagram

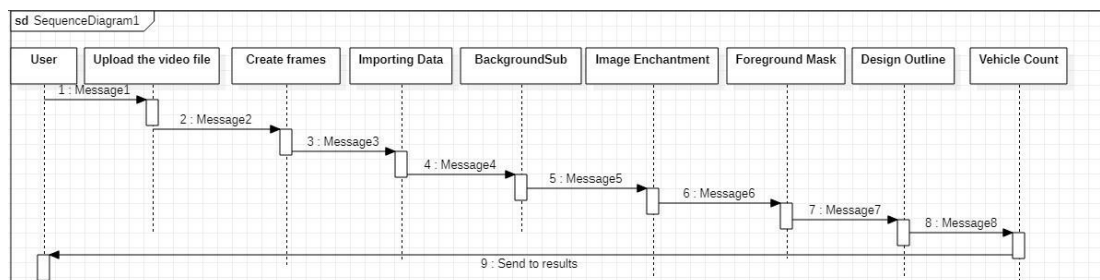
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



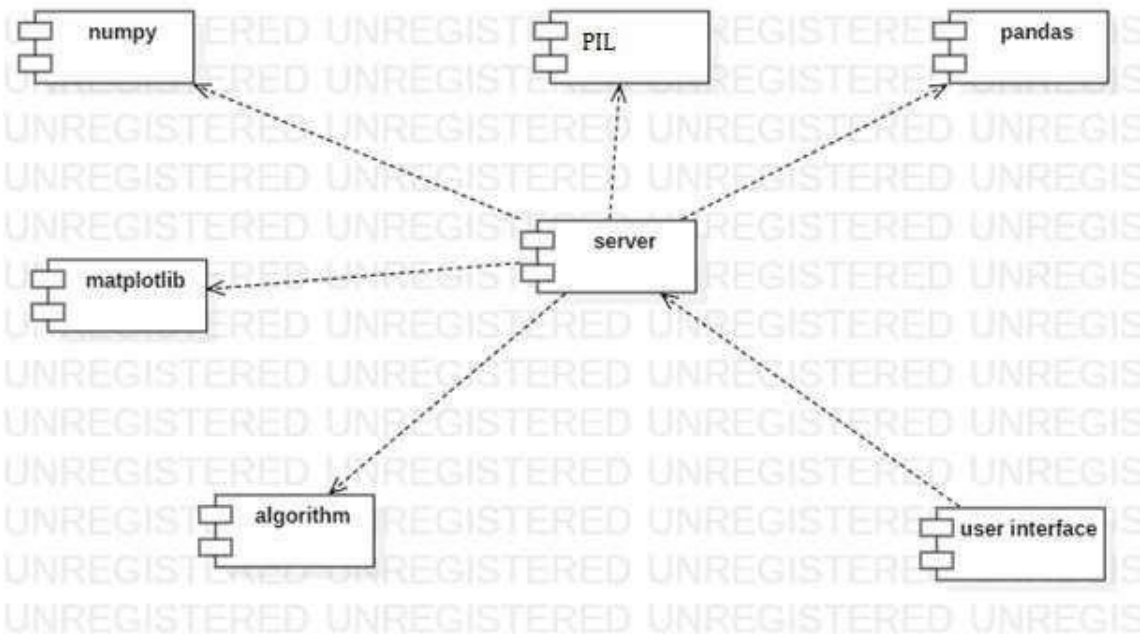


## Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

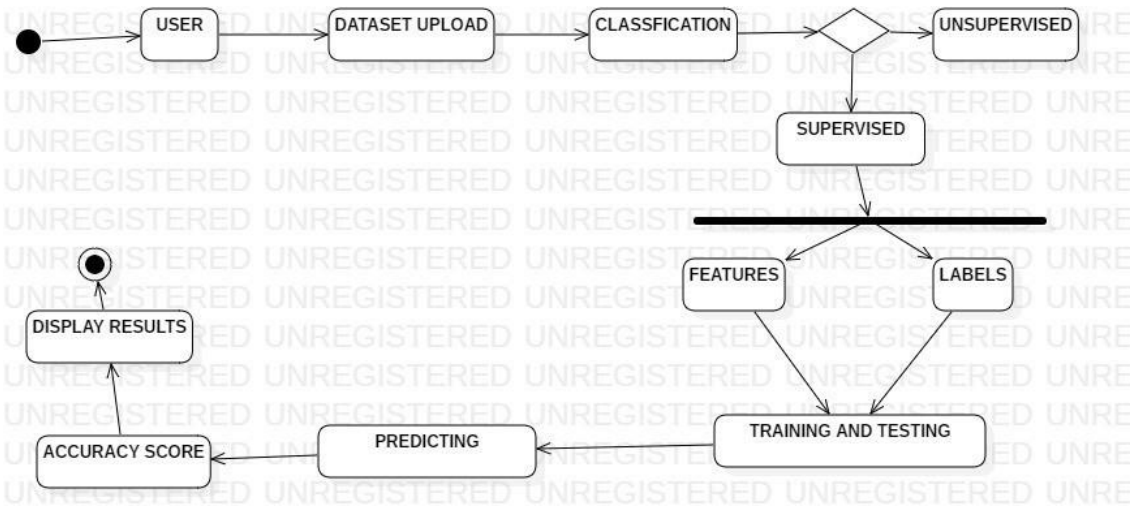


**Component Diagram**



**Activity diagram**

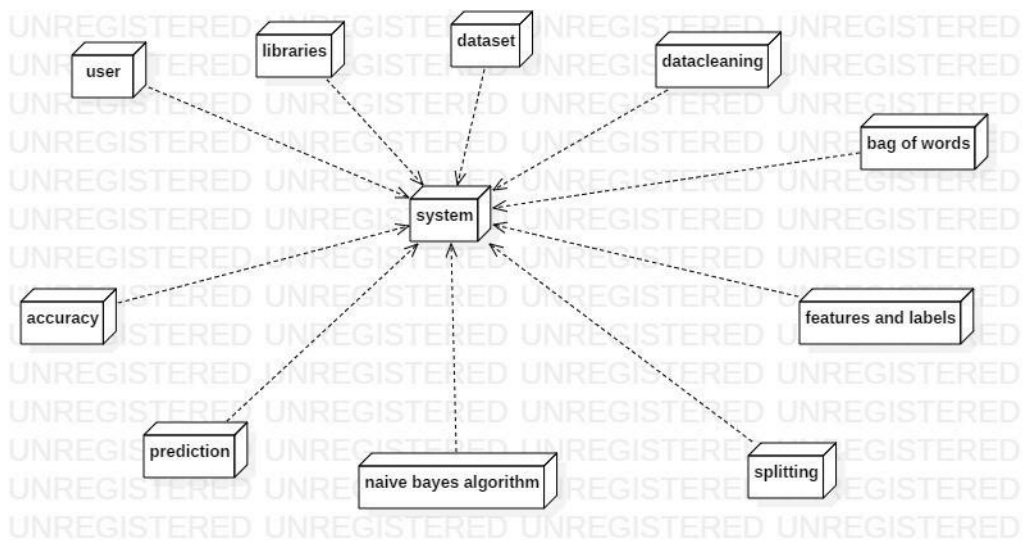
Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



## Deployment diagram

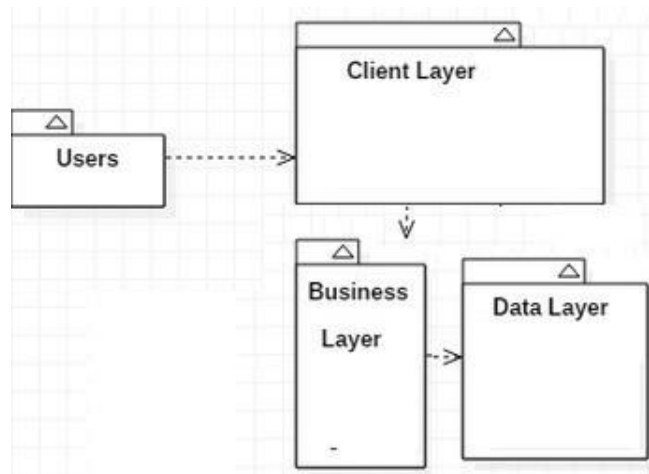
A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes.[1] To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have subnodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



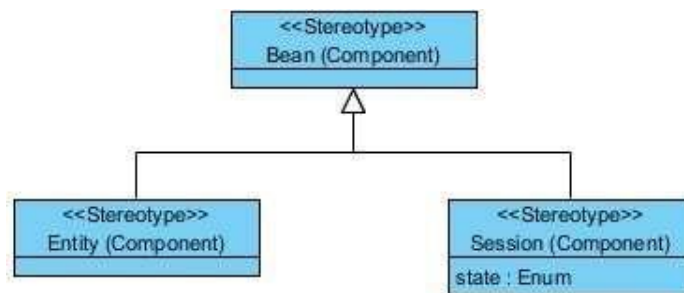
## Package Diagram

Package diagram is UML structure diagram which shows structure of the designed system at the level of packages. The following elements are typically drawn in a diagram: package, packageable element, dependency, element import, Package import, package merge.



## Profile Diagram

A Profile diagram is any diagram created in a «profile» Package. Profiles provide a means of extending the UML. They are based on additional stereotypes and Tagged Values that are applied to UML elements, connectors and their components.



# CHAPTER 6

## MACHINE LEARNING

### **What is Machine Learning**

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

### **Categories of Machine Learning**

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label and is often described as "letting the dataset speak for itself." These models include tasks

such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

### **Need for Machine Learning**

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate, and solve complex problems. On the other side, AI is still in its initial stage and have not surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

### **Challenges in Machines Learning**

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are –

1. Quality of data – Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.
2. Time-Consuming task – Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.
3. Lack of specialist persons – As ML technology is still in its infancy stage, availability of expert resources is a tough job.
4. No clear objective for formulating business problems – Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

5. Issue of overfitting & underfitting – If the model is overfitting or underfitting, it cannot be represented well for the problem.
6. Curse of dimensionality – Another challenge ML model faces is too many features of data points. This can be a real hindrance.
7. Difficulty in deployment – Complexity of the ML model makes it quite difficult to be deployed in real life.

## **Applications of Machines Learning**

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML.

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

## **How to Start Learning Machine Learning?**

Arthur Samuel coined the term “Machine Learning” in 1959 and defined it as a “Field of study that gives computers the capability to learn without being explicitly programmed”.

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is the Best Job of 2019 with a 344% growth and an average base salary of \$146,085 per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So, this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

### **How to start learning ML?**

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

#### **Step 1 – Understand the Prerequisites**

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

##### **(a) Learn Linear Algebra and Multivariate Calculus**

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

##### **(b) Learn Statistics**

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So, it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

##### **(c) Learn Python**



Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So, if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as Fork Python available Free on GeeksforGeeks.

## Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

### **(a) Terminologies of Machine Learning**

- **Model** – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature** – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label)** – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training** – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction** – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

# CHAPTER 7

## SOFTWARE ENVIRONMENT

### What is Python?

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

## **Advantages of Python**

Let's see how Python dominates over other languages.

### **1. Extensive Libraries**

Python downloads with an extensive library and it contains code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

### **2. Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

### **3. Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

### **4. Improved Productivity**

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

### **5. IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

### **6. Simple and Easy**

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

### **7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code.

## 8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

## 9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

## 10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

## 11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

## **Advantages of Python Over Other Languages**

### 1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

## 2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

## 3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

### **Disadvantages of Python**

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

#### 1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

#### 2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

#### 3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

#### 4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

#### 5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

### **History of Python**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venner<sup>1</sup>, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax,

used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

### **Python Development Steps**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it." Some changes in Python 7.3:

- Print is now a function.
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.
- The division of two integers returns a float instead of an integer. "/" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

### **Purpose**

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

## **Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

## **Modules Used in Project**

### **TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.



## **NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

## **Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

## **Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

### **Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

### **Install Python Step-by-Step in Windows and Mac**

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-

level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

## **How to Install Python on Windows and Mac**

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e., operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

## **Download the Correct version into the system**

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>



Now, check for the latest and the correct version for your operating system.








Step 2: Click on the Download Tab.



Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	 Download	<a href="#">Release Notes</a>
Python 3.6.9	July 2, 2019	 Download	<a href="#">Release Notes</a>
Python 3.7.3	March 25, 2019	 Download	<a href="#">Release Notes</a>
Python 3.4.10	March 18, 2019	 Download	<a href="#">Release Notes</a>
Python 3.5.7	March 10, 2018	 Download	<a href="#">Release Notes</a>
Python 2.7.16	March 4, 2019	 Download	<a href="#">Release Notes</a>
Python 3.7.2	Dec. 24, 2018	 Download	<a href="#">Release Notes</a>

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

# CHAPTER 8

## SYSTEM REQUIREMENTS SPECIFICATIONS

### Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

### Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

Operating system	:	Windows, Linux
Processor	:	minimum intel i3
Ram	:	minimum 4 GB
Hard disk	:	minimum 250GB

# CHAPTER 9

## FUNCTIONAL REQUIREMENTS

### Output Design

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

### Output Definition

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

### Input Design

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.

- To ensure that the input is acceptable and understood by the user.

### **Input Stages**

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

### **Input Types**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

### **Input Media**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requiremen

## **User Interface Systems Can Be Broadly Classified As:**

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

### **User Initiated Interfaces**

User initiated interfaces fall into two approximate classes:

- Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

### **Computer-Initiated Interfaces**

The following computer – initiated interfaces were used:

- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

### **Error Message Design**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.



## **Performance Requirements**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the duties.

## CHAPTER 9

### SOURCE CODE

```
import numpy as np

import imutils

import time

from scipy import spatial

import cv2

from input_retrieval import *

list_of_vehicles = ["bicycle", "car", "motorbike", "bus", "truck", "train"]

FRAMES_BEFORE_CURRENT = 10

inputWidth, inputHeight = 416, 416

LABELS, weightsPath, configPath, inputVideoPath, outputVideoPath, \

    preDefinedConfidence, preDefinedThreshold, USE_GPU = \

    parseCommandLineArguments()

np.random.seed(42)

COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),

    dtype="uint8")

def displayVehicleCount(frame, vehicle_count):

    cv2.putText(

        frame, #Image

        'Detected Vehicles and Objects: ' + str(vehicle_count), #Label

        (20, 20), #Position

        cv2.FONT_HERSHEY_SIMPLEX, #Font

        0.8, #Size

        (0, 0xFF, 0), #Color
```

```
2, #Thickness

cv2.FONT_HERSHEY_COMPLEX_SMALL,

)
```

```
def boxAndLineOverlap(x_mid_point, y_mid_point, line_coordinates):

    x1_line, y1_line, x2_line, y2_line = line_coordinates #Unpacking

    if (x_mid_point >= x1_line and x_mid_point <= x2_line+5) and\

        (y_mid_point >= y1_line and y_mid_point <= y2_line+5):

        return True

    return False
```

```
def displayFPS(start_time, num_frames):

    current_time = int(time.time())

    if(current_time > start_time):

        os.system('clear') # Equivalent of CTRL+L on the terminal

        print("FPS:", num_frames)

        num_frames = 0

        start_time = current_time

    return start_time, num_frames
```

```
def drawDetectionBoxes(idxs, boxes, classIDs, confidences, frame):

    # ensure at least one detection exists

    if len(idxs) > 0:

        # loop over the indices we are keeping

        for i in idxs.flatten():
```

```

# extract the bounding box coordinates

(x, y) = (boxes[i][0], boxes[i][1])

(w, h) = (boxes[i][2], boxes[i][3])

# draw a bounding box rectangle and label on the frame

color = [int(c) for c in COLORS[classIDs[i]]]

cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)

text = "{: {:.4f}".format(LABELS[classIDs[i]],

                        confidences[i])

cv2.putText(frame, text, (x, y - 5),

            cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

#Draw a green dot in the middle of the box

cv2.circle(frame, (x + (w//2), y + (h//2)), 2, (0, 0xFF, 0), thickness=2)

```

```

def initializeVideoWriter(video_width, video_height, videoStream):

```

```

    # Getting the fps of the source video

    sourceVideofps = videoStream.get(cv2.CAP_PROP_FPS)

    # initialize our video writer

    fourcc = cv2.VideoWriter_fourcc(*"MJPG")

    return cv2.VideoWriter(outputVideoPath, fourcc, sourceVideofps,

                            (video_width, video_height), True)

```

```

def boxInPreviousFrames(previous_frame_detections, current_box, current_detections):

```

```

    centerX, centerY, width, height = current_box

    dist = np.inf #Initializing the minimum distance

    for i in range(FRAMES_BEFORE_CURRENT):

```

```

        coordinate_list = list(previous_frame_detections[i].keys())

        if len(coordinate_list) == 0: # When there are no detections in the previous
frame
            continue

        temp_dist, index = spatial.KDTree(coordinate_list).query([(centerX,
centerY)])

        if (temp_dist < dist):

            dist = temp_dist

            frame_num = i

            coord = coordinate_list[index[0]]

    if (dist > (max(width, height)/2)):

        return False

    current_detections[(centerX, centerY)] =
previous_frame_detections[frame_num][coord]

    return True

def count_vehicles(idxs, boxes, classIDs, vehicle_count, previous_frame_detections, frame):

    current_detections = { }

    # ensure at least one detection exists

    if len(idxs) > 0:

        # loop over the indices we are keeping

        for i in idxs.flatten():

            # extract the bounding box coordinates

```

```

(x, y) = (boxes[i][0], boxes[i][1])

(w, h) = (boxes[i][2], boxes[i][3])


centerX = x + (w//2)

centerY = y + (h//2)

if (LABELS[classIDs[i]] in list_of_vehicles):

    current_detections[(centerX, centerY)] = vehicle_count

    if (not boxInPreviousFrames(previous_frame_detections,
(centerX, centerY, w, h), current_detections)):

        vehicle_count += 1

    ID = current_detections.get((centerX, centerY))

    if (list(current_detections.values()).count(ID) > 1):

        current_detections[(centerX, centerY)] = vehicle_count

        vehicle_count += 1


cv2.putText(frame, str(ID), (centerX, centerY),\

cv2.FONT_HERSHEY_SIMPLEX, 0.5, [0,0,255], 2)


return vehicle_count, current_detections


print("[INFO] loading YOLO from disk...")

net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)


#Using GPU if flag is passed

if USE_GPU:

```

```

net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)

net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)


ln = net.getLayerNames()

ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]


videoStream = cv2.VideoCapture(inputVideoPath)

video_width = int(videoStream.get(cv2.CAP_PROP_FRAME_WIDTH))

video_height = int(videoStream.get(cv2.CAP_PROP_FRAME_HEIGHT))


# Specifying coordinates for a default line

x1_line = 0

y1_line = video_height//2

x2_line = video_width

y2_line = video_height//2


#Initialization

previous_frame_detections = [{(0,0):0} for i in range(FRAMES_BEFORE_CURRENT)]

num_frames, vehicle_count = 0, 0

writer = initializeVideoWriter(video_width, video_height, videoStream)

start_time = int(time.time())

# loop over frames from the video file stream

while True:

    print("=====NEW FRAME=====")

```

```

num_frames+= 1

print("FRAME:\t", num_frames)

# Initialization for each iteration

boxes, confidences, classIDs = [], [], []

vehicle_crossed_line_flag = False


#Calculating fps each second

start_time, num_frames = displayFPS(start_time, num_frames)

# read the next frame from the file

(grabbed, frame) = videoStream.read()


# if the frame was not grabbed, then we have reached the end of the stream
if not grabbed:

    break


blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (inputWidth, inputHeight),

    swapRB=True, crop=False)

net.setInput(blob)

start = time.time()

layerOutputs = net.forward(ln)

end = time.time()


# loop over each of the layer outputs
for output in layerOutputs:

```



```

# loop over each of the detections

for i, detection in enumerate(output):

    # extract the class ID and confidence (i.e., probability)
    # of the current object detection

    scores = detection[5:]

    classID = np.argmax(scores)

    confidence = scores[classID]

    if confidence > preDefinedConfidence:

        box = detection[0:4] * np.array([video_width, video_height,
video_width, video_height])

        (centerX, centerY, width, height) = box.astype("int")

        # use the center (x, y)-coordinates to derive the top
        # and left corner of the bounding box

        x = int(centerX - (width / 2))

        y = int(centerY - (height / 2))

        boxes.append([x, y, int(width), int(height)])

        confidences.append(float(confidence))

        classIDs.append(classID)

```

```

idxs = cv2.dnn.NMSBoxes(bboxes, confidences, preDefinedConfidence,
                        preDefinedThreshold)

# Draw detection box

drawDetectionBoxes(idx, bboxes, classIDs, confidences, frame)

vehicle_count, current_detections = count_vehicles(idx, bboxes, classIDs,
vehicle_count, previous_frame_detections, frame)

# Display Vehicle Count if a vehicle has passed the line

displayVehicleCount(frame, vehicle_count)

# write the output frame to disk

writer.write(frame)

cv2.imshow('Frame', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

    break

# Updating with the current frame detections

previous_frame_detections.pop(0) #Removing the first frame from the list

# previous_frame_detections.append(spatial.KDTree(current_detections))

previous_frame_detections.append(current_detections)

# release the file pointers

print("[INFO] cleaning up...")

writer.release()

videoStream.release()

import tkinter as tk

from tkinter import filedialog

from tkinter import *

import os

```

```

import subprocess

import numpy

#initialise GUI

top=tk.Tk()

top.geometry('1200x750')

top.title('Object Detection, Vehicle Counting and Classification Sytem')

bg = PhotoImage(file = "a.png")

canvas1 = Canvas( top, width = 800, height = 800)

canvas1.pack(fill = "both", expand = True)

# Display image

canvas1.create_image( 0, 0, image = bg, anchor = "nw")

# top.configure(background= bg)

label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))

sign_image = Label(top)

def classify(file_path):

    # print(file_path)

    Str="Main1.py --input inputVideos/PexelsVideos1721294.mp4 --output
outputVideos/PexelsVideos1721294.avi --yolo yolo-"

    # harActivity = "Main1.py --input inputVideos/bridge.mp4 --output
outputVideos/bridgeOut.avi --yolo yolo-"

    subprocess.call("python "+Str)

def show_classify_button(file_path):

    classify_b=Button(top,text="Get RealTime Reading",command=lambda:
classify(file_path),padx=10,pady=5)

```

```

classify_b.configure(background='#364156', foreground='white',font=('arial',10,'bold'))

classify_b.place(relx=0.79,rely=0.46)

button2_canvas = canvas1.create_window( 800, 300, anchor = "nw", window = classify_b)

def upload_video():

    try:

        file_path=filedialog.askopenfilename()

        label.configure(text="")

        show_classify_button(file_path)

    except:

        pass

upload=Button(top,text="Upload Input Video",command=upload_video,padx=10,pady=5)

upload.configure(background='#364156', foreground='white',font=('arial',10,'bold'))

upload.pack(side=BOTTOM,pady=50)

button1_canvas = canvas1.create_window( 300, 500, anchor = "nw", window = upload)

sign_image.pack(side=BOTTOM,expand=True)

label.pack(side=BOTTOM,expand=True)

heading = Label(top, text="Object Detection, Vehicle Counting and Classification
Sytem",pady=20, font=('arial',20,'bold'))

heading.configure(background='#CDCDCD',foreground='#FF0000')

heading.pack()

button2_canvas = canvas1.create_window( 100, 100, anchor = "nw", window = heading)

top.mainloop()

```

# CHAPTER 10

## RESULTS AND DISCUSSION

### 10.1 Implementation description:

This code appears to be a Python script that combines object detection, vehicle counting, and classification using a pre-trained YOLO (You Only Look Once) model. It also includes a graphical user interface (GUI) using the tkinter library for uploading a video and initiating the object detection process.

Importing Libraries:

- `numpy`: For numerical operations.
- `imutils`: A library for basic image processing tasks.
- `time`: For time-related operations.
- `scipy.spatial`: For spatial distance calculations.
- `cv2`: OpenCV library for computer vision tasks.
- `input_retrieval`: A module that seems to handle command-line argument parsing, but it's not provided here.
- **List of Vehicles:**  
`list_of_vehicles`: A list of strings representing different types of vehicles.
- **Constants:**  
`FRAMES_BEFORE_CURRENT`: An integer constant set to 10.
- **Parsing Command Line Arguments:**  
It is to retrieve various parameters like labels, file paths, confidence thresholds, and GPU usage flag from the command line using a function called `parseCommandLineArguments()`. However, the details of this function are not provided.
- **Setting up Random Colors:**  
Generates random RGB colors for different classes.
- **Displaying Vehicle Count:**  
`displayVehicleCount()`: A function to display the vehicle count on the frame.
- **Box and Line Overlap Check:**  
`boxAndLineOverlap()`: A function to check if a bounding box overlaps with a given line.

- **Displaying FPS:**  
displayFPS(): A function to display frames per second.
- **Drawing Detection Boxes:**  
drawDetectionBoxes(): Draws bounding boxes, labels, and green dots at the center of the detected objects.
- **Initializing Video Writer:**  
initializeVideoWriter(): Initializes a video writer for saving the processed frames.
- **Box in Previous Frames Check:**  
boxInPreviousFrames(): Checks if a box from the current frame has appeared in previous frames.
- **Counting Vehicles:**  
count\_vehicles(): Counts vehicles and updates their IDs.
- **Loading YOLO Model:**  
Reads the YOLO model from the provided configuration and weights files.
- **Using GPU (if specified).**
- **Getting Layer Names:**  
In: Gets the layer names from the YOLO network.
- **Initializing Video Stream:**  
Captures video from the provided input file.
- **Setting Up Line Coordinates:**  
x1\_line, y1\_line, x2\_line, y2\_line: Define coordinates for a default line.
- **Initialization:**  
Initializes various variables and data structures.
- **Main Loop:**  
Iterates over frames from the video stream.
- **Performs object detection, vehicle counting, and classification.**  
Draws boxes, updates vehicle count, and displays frames.
- **GUI using Tkinter:**  
Creates a GUI window for uploading videos.  
Includes a button to trigger real-time object detection and counting.
- **Button Actions:**  
upload\_video(): Opens a file dialog to select a video file.  
classify(file\_path): Calls a function to initiate object detection on the uploaded video.

- Running the GUI:  
Starts the Tkinter main loop to run the GUI.

## 10.2 Results and Description

Figure 1 represents a single frame from a video. In this frame, there are two vehicles detected: a bus and a car. The object detection model has identified a car in the frame, and it has assigned a high confidence score to this classification, indicating a 97% accuracy in identifying it as a car. Similarly, the model has also identified a bus in the frame, but with a slightly lower confidence score, indicating a 61.74% accuracy in identifying it as a bus.

Figure 2 represents another frame from the same video. In this frame, there are two objects detected: a person and a motorbike. The object detection model has identified a person in the frame, and it has assigned a high confidence score to this classification, indicating a 94% accuracy in identifying it as a person. Similarly, the model has also identified a motorbike in the frame, but with a somewhat lower confidence score, indicating a 64% accuracy in identifying it as a motorbike.

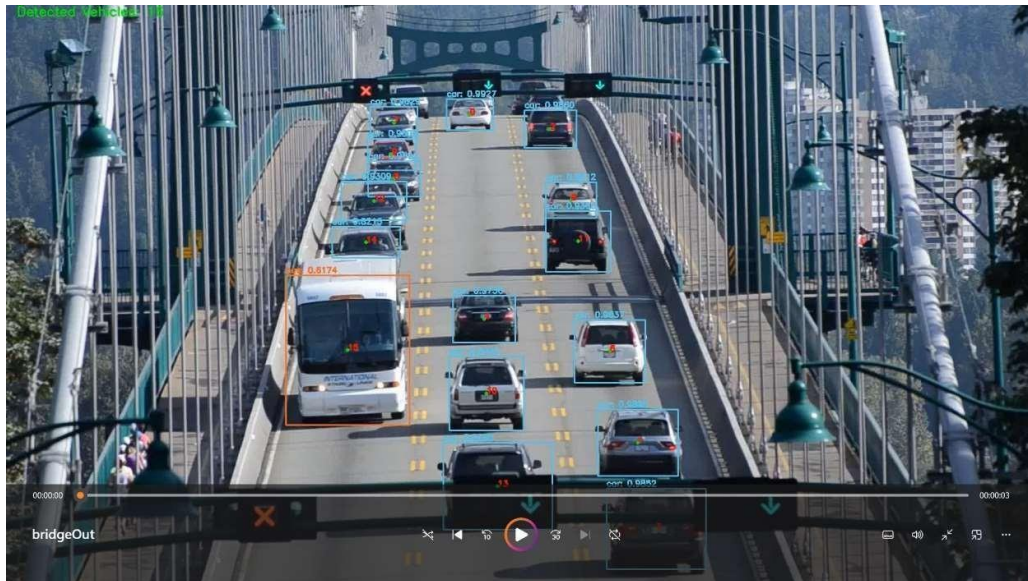


Figure 1: Frame with Bus and car vehicle classification.

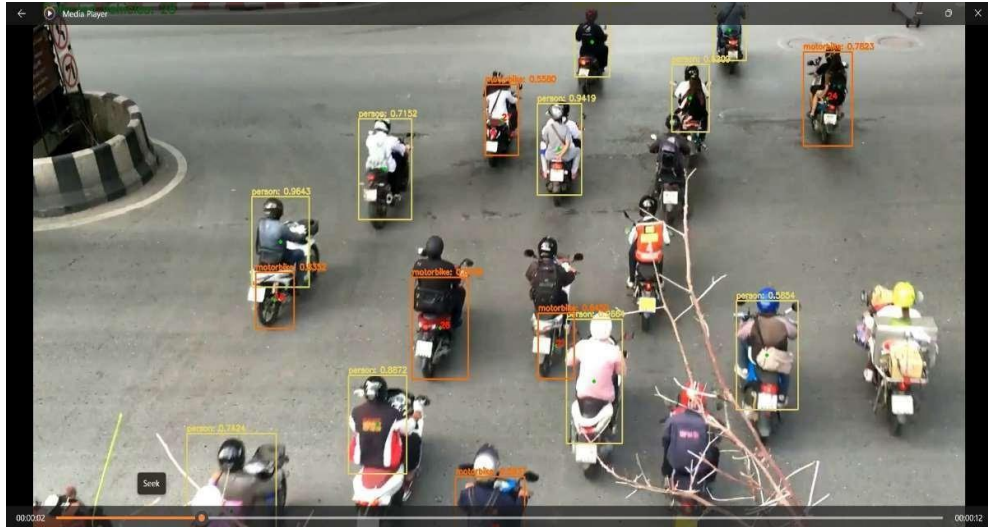


Figure 2: video frame with classification of both person and motor bike.

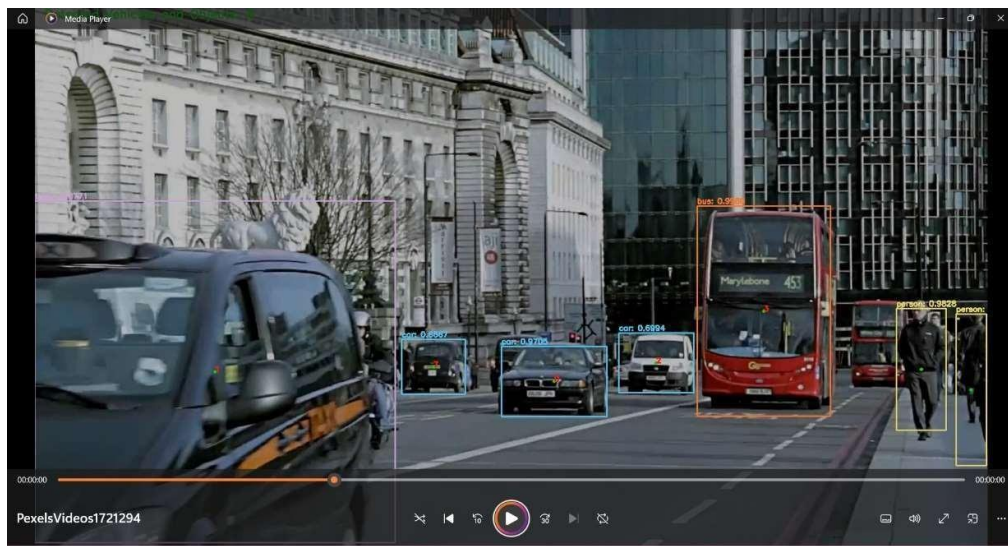


Figure 3: video frame with classification of persons, cars & buses with an accuracy of 94% ,96% & 99%

Figure 3 represents yet another frame from the same video. In this frame, there are multiple objects detected, including persons, cars, and buses. The object detection model has identified persons in the frame with a high confidence score, indicating a 94% accuracy in identifying them as persons. The model has also detected cars in the frame, and it is highly confident in this classification, indicating a 96% accuracy in identifying them as cars. Furthermore, the model has detected buses in the frame with an extremely high confidence score, indicating a 99% accuracy in identifying them as buses.



## **CHAPTER 11**

### **CONCLUSION AND FUTURE SCOPE**

#### **Conclusion**

The proposed solution is implemented on python, using the OpenCV bindings. The traffic camera footages from variety of sources are in implementation. A simple interface is developed for the user to select the region of interest to be analyzed and then image processing techniques are applied to calculate vehicle count and classified the vehicles using machine learning algorithms. From experiments it is apparent that CC method outperforms than BoF and SVM method in all results and gives more close classification results to the ground truth values.

#### **Future scope**

One of the limitations of the system is that it is not efficient at detection of occlusion of the vehicles which affects the accuracy of the counting as well as classification. This problem could be solved by introducing the second level feature classification such as the classification on the bases of color. Another limitation of the current system is that it needs human supervision for defining the region of interest. The user must define an imaginary line where centroid of the contours intersects for the counting of vehicles hence the accuracy is dependent on the judgment of the human supervisor. Furthermore, the camera angle also affects the system hence camera calibration techniques could be used for the detection of the lane for the better view of the road and increasing the efficiency. The system is not capable of detection of vehicles in the night as it needs the foreground objects to be visible for extraction of contour properties as well as features for the classification using SIFT features. The system could also be improved for better accuracy using the more sophisticated image segmentation and artificial intelligence operations.

## REFERENCES

- [1] Alpatov, Boris & Babayan, Pavel & Ershov, Maksim. (2018). Vehicle detection and counting system for real-time traffic surveillance. 1-4. 10.1109/MECO.2018.8406017.
- [2] Song, H., Liang, H., Li, H. et al. Vision-based vehicle detection and counting system using deep learning in highway scenes. *Eur. Transp. Res. Rev.* 11, 51 (2019). <https://doi.org/10.1186/s12544-019-0390-4>.
- [3] Neupane, Bipul et al. "Real-Time Vehicle Classification and Tracking Using a Transfer Learning-Improved Deep Learning Network." *Sensors (Basel, Switzerland)* vol. 22,10 3813. 18 May. 2022, doi:10.3390/s22103813.
- [4] C. J Lin, Shiou-Yun Jeng, Hong-Wei Lioa, "A Real-Time Vehicle Counting, Speed Estimation, and Classification System Based on Virtual Detection Zone and YOLO", *Mathematical Problems in Engineering*, vol. 2021, Article ID 1577614, 10 pages, 2021. <https://doi.org/10.1155/2021/1577614>.
- [5] M. S. Chauhan, A. Singh, M. Khemka, A. Prateek, and Rijurekha Sen. 2019. Embedded CNN based vehicle classification and counting in non-laned road traffic. In *Proceedings of the Tenth International Conference on Information and Communication Technologies and Development (ICTD '19)*. Association for Computing Machinery, New York, NY, USA, Article 5, 1–11. <https://doi.org/10.1145/3287098.3287118>.
- [6] A. Arinaldi, J. A. Pradana, A. A. Gurusinga, "Detection and classification of vehicles for traffic video analytics", *Procedia Computer Science*, Volume 144, 2018, Pages 259-268, ISSN 1877-0509, <https://doi.org/10.1016/j.procs.2018.10.527>.
- [7] Goma, A., Minematsu, T., Abdelwahab, M.M. et al. Faster CNN-based vehicle detection and counting strategy for fixed camera scenes. *Multimed Tools Appl* 81, 25443–25471 (2022). <https://doi.org/10.1007/s11042-022-12370-9>.
- [8] G. Oltean, C. Florea, R. Orghidan and V. Oltean, "Towards Real Time Vehicle Counting using YOLO-Tiny and Fast Motion Estimation," 2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME), 2019, pp. 240-243, doi: 10.1109/SIITME47687.2019.8990708.
- [9] L. C. Pico and D. S. Benítez, "A Low-Cost Real-Time Embedded Vehicle Counting and Classification System for Traffic Management Applications," 2018 IEEE Colombian Conference on Communications and Computing (COLCOM), 2018, pp. 1-6, doi: 10.1109/ColComCon.2018.8466734.

- [10] D. E. V. Tituana, S. G. Yoo and R. O. Andrade, "Vehicle Counting using Computer Vision: A Survey," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), 2022, pp. 1-8, doi: 10.1109/I2CT54291.2022.9824432.
- [11] A. Khan, A., Sabeenian, R.S., Janani, A.S., Akash, P. (2022). Vehicle Classification and Counting from Surveillance Camera Using Computer Vision. In: Suma, V., Baig, Z., K. Shanmugam, S., Lorenz, P. (eds) Inventive Systems and Control. Lecture Notes in Networks and Systems, vol 436. Springer, Singapore. [https://doi.org/10.1007/978-981-19-1012-8\\_31](https://doi.org/10.1007/978-981-19-1012-8_31).
- [12] W. Balid, H. Tafish and H. H. Refai, "Intelligent Vehicle Counting and Classification Sensor for Real-Time Traffic Surveillance," in IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 6, pp. 1784-1794, June 2018, doi: 10.1109/TITS.2017.2741507.
- [13] N. Jahan, S. Islam and M. F. A. Foysal, "Real-Time Vehicle Classification Using CNN," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1-6, doi: 10.1109/ICCCNT49239.2020.9225623.
- [14] M. A. Butt, A. M. Khattak, S. Shafique, B. Hayat, S. Abid, Ki-Il Kim, M. W. Ayub, A. Sajid, A. Adnan, "Convolutional Neural Network Based Vehicle Classification in Adverse Illuminous Conditions for Intelligent Transportation Systems", Complexity, vol. 2021, Article ID 6644861, 11 pages, 2021. <https://doi.org/10.1155/2021/6644861>.
- [15] P. Gonzalez, Raul & Nuño-Maganda, Marco Aurelio. (2014). Computer vision based real-time vehicle tracking and classification system. Midwest Symposium on Circuits and Systems. 679-682. 10.1109/MWSCAS.2014.6908506.