Vehicle Counting and Detection For Parking Area

*A Report Submitted*

*In the partial fulfillment of the requirement*

*For the Degree of*

*Bachelor of Technology*

**NEERAJ TIWARI 201910101110021**

**ABHISHEK PRATAP SINGH 201910101110006**

**MD. ARMAAN 201910101110050**

**ANURAG JAIN 201910101110052**

Project Guide: Mr. Neelesh Mishra

*To the*

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**INSTITUTE OF TECHNOLOGY**

**SHRI RAMSWAROOP MEMORIAL UNIVERSITY**



**JUNE 2023**

# CERTIFICATE

It is certified that the work contained in the project report titled “Vehicle counting and detection for Parking Area”, by “Neeraj Tiwari, Abhishek Pratap Singh, Md. Armaan and Anurag Jain”, has been carried out under my supervision and that this work has not been submitted elsewhere for any other degree.

**Signature of Supervisor**

**Mr. Neelesh Mishra**

**Department of Computer Science and Engineering**

**Shri Ramswaroop Memorial University**

June 2023

**Signature of Dean**

**Dr. Satya Bhushan Verma**

**Department of Computer Science and Engineering**

**Shri Ramswaroop Memorial University**

## DECLARATION

We, Neeraj Tiwari (201910101110021), Abhishek Pratap Singh (201910101110006), Md. Armaan (201910101110050), Anurag Jain (201910101110052), students of Bachelor of Technology, Computer Science & Engineering department at Shri Ramswaroop Memorial University, Lucknow hereby declare that the work presented in this project titled "VEHICLE COUNTING AND DETECTION FOR PARKING AREA" is the outcome of our work, is bona fide, correct to the best of our knowledge and this work has been carried out taking care of engineering ethics, We have completely taken care in acknowledging the contribution of others in this academic work. We further declare that in case of any violation of intellectual property rights or copyrights found at any stage, we as the candidates will be solely responsible for the same.

Date: June 2023

Place:

Neeraj Tiwari (201910101110021)

Abhishek Pratap Singh (201910101110006)

Md. Armaan (201910101110050)

Anurag Jain (201910101110052)

## ABSTRACT

This project titled "Vehicle Counting and Detection for Parking Area" aims to develop a system that can accurately detect and count vehicles in a parking area. The growing number of vehicles on the roads has led to an increased demand for efficient parking management systems. This project addresses the need for an automated solution to monitor and manage parking spaces.

The project utilizes two key technologies: YOLO (You Only Look Once) and OpenCV (Open Source Computer Vision Library). YOLO is a state-of-the-art object detection algorithm known for its speed and accuracy. It can identify and classify objects within an image or video stream. OpenCV, on the other hand, is a powerful computer vision library that provides various tools and functions for image processing and analysis.

The system workflow involves capturing live video footage from surveillance cameras placed in the parking area. The video frames are then processed using YOLO and OpenCV algorithms to detect and classify vehicles. By employing deep learning techniques, the system can accurately identify different types of vehicles, such as cars, motorcycles, and trucks, in real-time.

Furthermore, the system incorporates advanced counting algorithms to track the number of vehicles entering and exiting the parking area. This enables efficient monitoring of parking space availability and helps in managing parking resources effectively. The vehicle count data can be used to generate statistical reports and aid in decision-making processes related to parking management.

The project aims to achieve high accuracy, real-time processing, and robustness in vehicle detection and counting. Extensive testing and evaluation will be conducted to validate the system's performance under various conditions, such as different lighting conditions, varying weather conditions, and varying traffic densities.

Overall, the Vehicle Counting and Detection for Parking Area project leverages the power of YOLO and OpenCV to develop an automated system that can effectively detect, classify, and count vehicles in a parking area. The implementation of this system will contribute to improved parking management, reduced congestion, and enhanced user experience in parking facilities.

## ACKNOWLEDGEMENT

The present work will remain incomplete unless we express our feelings of gratitude towards several people who delightfully co-operated with us in the process of this work. We extend our hearty and sincere gratitude to our project guide, Mr. Neelesh Mishra, Assistant Professor in the Department of Computer Science and Engineering, for his valuable direction, suggestions, and exquisite guidance with ever enthusiastic encouragement ever since the commencement of this project.

This project would not have taken shape, without the guidance provided by our project coordinator, **Dr. Satya Bhushan Verma,** Head of the Department of Computer Science and Engineering, who helped in our project and resolved all the technical as well as other problems related to the project and for always providing us with a helping hand whenever we faced any bottlenecks despite being quite busy with their hectic schedules.

**CHAPTER 1**

**INTRODUCTION**

* 1. **Vehicle counting and detection for parking area:**

In today's urbanized world, parking management has become a significant concern due to the exponential growth in vehicle population. Efficient utilization of parking spaces necessitates accurate vehicle counting and detection systems, which enable effective management and allocation of available parking areas. In this context, our project focuses on developing a robust and intelligent solution for vehicle counting and detection in parking areas.

To achieve this goal, we employ a combination of advanced technologies and algorithms, including YOLO (You Only Look Once), OpenCV (Open Source Computer Vision Library), CNN (Convolutional Neural Network), R-CNN (Region-based Convolutional Neural Network), and the Euclidean distance method.

The YOLO algorithm is a state-of-the-art object detection system that can identify objects in real-time using deep learning techniques. By leveraging its capabilities, we aim to detect and localize vehicles within the parking area. OpenCV, a powerful computer vision library, will provide us with a rich set of tools for image processing, feature extraction, and object recognition, complementing the YOLO algorithm.

To enhance the accuracy of our system, we will utilize CNN and R-CNN models. These neural network architectures have proven to be highly effective in object detection tasks by learning and extracting discriminative features from input images. By training these models on an annotated dataset of vehicle images, we can significantly improve the accuracy and reliability of our vehicle detection system.

Furthermore, we will implement a counting algorithm based on the Euclidean distance method. By calculating the distances between detected vehicle positions in consecutive frames, we can accurately count the number of vehicles entering and exiting the parking area. This counting algorithm will be crucial for monitoring the occupancy of the parking area and providing real-time data to parking management systems.

The integration of YOLO, OpenCV, CNN, R-CNN, and the Euclidean distance algorithm in our project will provide a comprehensive solution for vehicle counting and detection in parking areas. The system will be capable of real-time monitoring, accurate counting, and efficient management of parking spaces, thereby improving overall parking efficiency and customer experience.

In the existing system, the conventional parking management systems often rely on human intervention, which is time-consuming and prone to errors. These systems typically utilize simple sensors or gatekeepers to monitor vehicle entry and exit, lacking the ability to accurately count and detect vehicles. Additionally, they often face challenges in handling crowded parking areas, distinguishing between vehicles, and providing real-time monitoring and analysis.

The system proposed in this project overcomes the limitations of existing parking management systems by employing cutting-edge technologies and algorithms. Here are some key advantages of the proposed system:

* **Advanced Vehicle Detection:** The integration of YOLO (You Only Look Once) and CNN (Convolutional Neural Network) enables accurate and efficient vehicle detection in real-time. YOLO's ability to detect objects within images combined with CNN's capability to learn and identify vehicle patterns ensures reliable detection results.
* **Precise Vehicle Counting:** The project employs the Eucledian Distance Algorithm to accurately count the number of vehicles in a parking area. This algorithm calculates the distances between detected vehicles, ensuring a reliable count even in crowded scenarios and when vehicles are partially occluded.
* **Comprehensive Tracking and Analysis:** By utilizing R-CNN (Region-based Convolutional Neural Network), the system can track and analyze the movement of vehicles within the parking area. This enables better monitoring, identification of parking violations, and efficient space management.
* **Real-time Monitoring and Alerts:** The system provides real-time monitoring of the parking area, allowing parking operators to have instant access to occupancy data, vehicle movement, and any potential anomalies. Automated alerts can be generated in case of overcrowding or unauthorized parking, enabling prompt action.
* **Integration with OpenCV:** OpenCV, a powerful computer vision library, is utilized for image processing, enhancing the accuracy and speed of vehicle detection and tracking. Its extensive set of functions and algorithms further enrich the capabilities of the proposed system.

In this project file, we will present the detailed implementation, methodology, experimental results, and analysis of our vehicle counting and detection system. We will also discuss the challenges faced during the development process and propose potential future enhancements to further improve the system's performance and scalability.

Through this project, we aim to contribute to the field of intelligent transportation systems and promote the adoption of advanced technologies for efficient parking management.

#### Preface:

#### It is mainly divided into two categories: one is the two-stage target detection algorithm based on classification, that is represented by R-CNN, Fast R-CNN and Faster R-CNN, and the other is the one-stage target detection algorithm that using regression algorithm by YOLO. In recent years, breakthroughs have been made in the application of target detection algorithms based on deep learning in Vehicle detection. However, when dealing with complex background and occlusion for With the development of artificial intelligence, the mainstream Vehicle detection method is based on deep learning algorithm objects detection, the accuracy of detecting small objects still needs to be improved.

The specific goals of the project include:

* **Vehicle Detection:** Implementing the YOLO object detection framework in combination with OpenCV, we will accurately identify and localize vehicles within the parking area. This will involve training the YOLO model on a suitable dataset to enable it to detect various types of vehicles.
* **Vehicle Counting:** Developing an advanced counting algorithm using CNN and R-CNN techniques, we will track the movement of vehicles and maintain an accurate count of vehicles entering and exiting the parking area. The counting algorithm will utilize the detections provided by the YOLO model to ensure precise counting results.
* **Vehicle Identification:** By applying the Euclidean distance method, we will enable the system to uniquely identify each vehicle within the parking area. This will involve calculating and comparing the distances between detected vehicles in consecutive frames, allowing us to track and identify individual vehicles throughout their presence in the parking area.
* **Real-time Performance:** Ensuring real-time processing of video streams, we will optimize the system's performance to handle high volumes of vehicles. This will involve leveraging hardware acceleration techniques and optimizing the implementation of the algorithms to achieve efficient processing speeds.
* **User-Friendly Interface:** Creating a user-friendly interface, we will provide an intuitive platform for users to monitor and analyze the vehicle counting and detection results. The interface will display real-time vehicle counts, identify specific vehicles, and generate comprehensive reports for further analysis.

By accomplishing these objectives, the project aims to deliver a reliable and accurate vehicle counting and detection system for parking areas. The implemented solution will provide valuable insights for parking management, traffic flow analysis, and overall optimization of parking resources.

#### Technology Used:

#### Computer vision:

This is advanced technology using computer vision is object detection, and image classification in python. It deals with identifying the object present in images or videos by frame. Object detection is a computer vision technique in which a software system can detect, locate, and trace the object from a given image or video. The special attribute about object detection is that it identifies the class of object (person, table, chair, etc.) and their location-specific coordinates in the given image.

The location is pointed out by drawing a bounding box around the object. The bounding box may or may not accurately locate the position of the object. The ability to locate the object inside an image defines the performance of the algorithm used for detection. A few years ago, the creation of the software and hardware image processing systems was mainly limited to the development of the user interface, which most of the programmers of each firm were engaged in.

The situation has been significantly changed with the advent of the Windows operating system when the majority of the developers switched to solving the problems of image processing itself. However, this has not yet led to the cardinal progress in solving typical tasks of recognizing faces, car numbers, road signs, analyzing remote and medical images, etc.

Humans can detect and identify objects present in an image. The human visual system is fast and accurate and can also perform complex tasks like identifying multiple objects and detect obstacles with little conscious thought. The availability of large sets of data, faster GPUs, and better algorithms, we can now easily train computers to detect and classify multiple objects within an image with high accuracy. We need to understand terms such as object detection, object localization, loss function for object detection and localization, and finally explore an object detection algorithm known as “You only look once” (YOLO).

Object recognition refers to a collection of related tasks for identifying objects in digital photographs. Region-based Convolutional Neural Networks, or R-CNNs, is a family of techniques for addressing object localization and recognition tasks, designed for model performance. You Only Look Once, or YOLO is known as the second family of techniques for object recognition designed for speed and real-time use.

#### Deep Learning:

Deep learning is a machine learning technique. It teaches a computer to filter inputs through layers to learn how to predict and classify information. Observations can be in the form of images, text, or sound. The inspiration for deep learning is the way that the human brain filters information. Its purpose is to mimic how the human brain works to create some real magic. In the human brain, there are about 100 billion neurons.

Each neuron connects to about 100,000 of its neighbors. We‟re kind of recreating that, but in a way and at a level that works for machines. In our brains, a neuron has a body, dendrites, and an axon. The signal from one neuron travels down the axon and transfers to the dendrites of the next neuron.

That connection where the signal passes is called a synapse. Neurons by themselves are kind of useless. But when you have lots of them, they work together to create some serious magic. That’s the idea behind a deep learning algorithm. You get input from observation and you put your input into one layer. That layer creates an output which in turn becomes the input for the next layer, and so on. This happens over and over until your final output signal.

#### NumPy

It's a Python package that helps with multi-dimensional arrays and matrices. To work with these arrays, it also has high-level mathematical algorithms. In all academic domains, it is utilised to analyze pipelines. Because of NumPy [12], the scientific Python ecosystem has grown tremendously. It serves as the interface between libraries and APIs. It's a fantastic tool for mathematical and scientific study. Images are used in our project and will be transformed to NumPy arrays. The labels are later applied to the image dataset.

#### OpenCV

The entire form of OpenCV is an open-source computer vision library; we're utilizing it since we need to install a webcam in the car to identify things in real-time for our project [5]. It's a computer vision library that works in real time [3]. Intel developed this library. This library has the advantage of being open-source and cross-platform. [4] Gaussian blur Method, Canny Edge Detection, and Hough Space are just a handful of the OpenCV algorithms available.

#### PyTorch

PyTorch is an open-source machine learning (ML) framework based on the Torch library and the Python programming language. It is one of the most popular deep learning research platforms. Framework is designed to accelerate the transition from research prototyping to implementation. Tensor computation and functional deep neural networks are two of PyTorch's most notable capabilities [13].

#### Dataset

Data collecting is an important aspect of the research in order to create an effective model. The accuracy of the model is heavily influenced by the data [5]. Microsoft created and maintained the COCO dataset, which we used. Its full form is common objects in context. Although the COCO dataset has 80 different object classes, we are only concerned in the human class [14]. As a result, we used a tool called FiftyOne (which is officially supported by the COCO dataset) to extract the images with the human label and create our own dataset from them.

1. **YOLO (You Only Look Once):**

YOLO is also an object detection algorithm which uses only one convolutional network to predicts the bounding boxes and the class probabilities and thus YOLO differs from other region based algorithms [1].

All the previous object detection algorithms have used regions to localize the object within the image. The network does not look at the complete image. Instead, parts of the image which has high probabilities of containing the object. YOLO or You Only Look Once is an object detection algorithm much is different from the region based algorithms which seen above [5]. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

YOLO is a convolutional neural network that allows to detect and classify objects in the form of bounding boxes. Such bounding box is the minimum sized rectangle, which will contain the whole found object. YOLO works on the principle of Single Shot. This means that the network architecture is arranged in such a way that in one pass of the frame, all objects are detected simultaneously [1].

The neural network has a large set of already trained classes, which can help to test all the capabilities of YOLO, as well as the performance of hardware. Also, YOLO can be trained for any class of images, if you correctly select the source data. To train YOLO, a large dataset is required. Each image should be provided with a text file with the marked regions of the trained class of objects. You also need a file with initial weights and a file with system information, which shows the paths to the images and the path, through which the recovery points will be recorded. Recovery points are the files of the weights at a certain training step. Weighting files are written to permanent memory every 100 iterations, which allows you to interrupt training at any time, and then continue with the last received weight.

The longer the network is trained, the better the detection quality will be. YOLO works by taking an image and split it into an SxS grid, within each of the grid we take m bounding boxes. For each of the bounding box, the network gives an output a class probability and offset values for the bounding box[1]. The bounding boxes have the class probability above a threshold value is selected and used to locate the object within the image. YOLO is orders of magnitude faster(45 frames per second) than any other object detection algorithms [15].

The limitation of YOLO algorithm is that it struggles with the small objects within the image, for example, it might have difficulties in identifying a flock of birds. This is due to the spatial constraints of the algorithm.

## CHAPTER 2

#### REQUIREMENT ANALYSIS AND FEASIBILITY STUDY

#### Requirement Analysis:

In various fields, there is a necessity to detect the target object and also track them effectively while handling occlusions and other included complexities. Many researchers attempted for various approaches in object tracking [5,6]. The nature of the techniques largely depends on the application domain

#### Information Gathering:

For gathering Information we focused on,

* Dataset: Looked for relevant datasets that contain images or videos of parking areas with labeled vehicle annotations. These datasets can be used for training and evaluation purposes.
* Implementation: Explored code examples, tutorials, or research papers that demonstrate the implementation of the technologies and algorithms you're using.
* Performance Evaluation: We Understood the metrics used to evaluate the performance of vehicle detection and counting algorithms, such as precision, recall, F1 score, and mean average precision.

We have also referred some of the research works which made the evolution to proposed work in the field of object tracking which are depicted as follows:

#### Image classification :

It is a prediction of different objects in a given image or video. It is used to classify the objects in the image like cars, trucks, pedestrians on the road [8]. A convoluted neural network is trained to identify the objects that are used in the model. For example, in our model, we use cars, trucks, bikes, and traffic lights [12]. These convolution operations are done to classify the classes. The CNN‟s only objectify one object in a single image; for that, we require a sliding technique to scan everywhere [2,3]. As we get the patch of results from the image, we can use CNN to get the possible objects [1,7]. If we have classes made according to the objects, they are accurate predictions; otherwise, any other thing will be false [15].

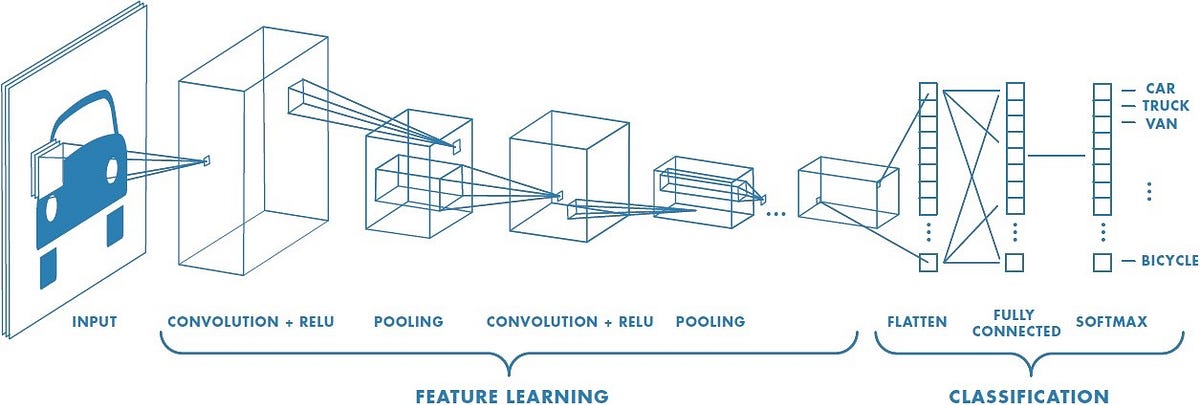
Now the object's size can also be an issue to identify on the image; if the size is too big or small concerning window size, it will be difficult to detect. To get high accuracy, CNN used multiple window sizes to run on the image [15]. It is complex and time taking to get the objects in the given image [12]. The Yolo algorithm is a great help to resolve this problem. As the name says, "you only look once," which means that we have to run CNN only once [7].

In Figure, the screen is split into grids, and we run the CNN. The probability of the class generates the probability of every 8 grid. Yolo is efficient because it predicts tiny grids of the image, and no need to run the CNN on multiple window sizes [2].

#### C:\Users\DELL\Downloads\Screen-Shot-2018-07-10-at-11.04.51-PM.png

#### Figure 2.1: Image split in to grid to obtain class probability[2

Convolution neural network is similar to the natural biological process. Humans and animals can do work, drive and eat their food without putting in too much effort. The computer cannot be trained like that because they have algorithms that are not direct. When there were lots of human research, we analyzed and observed many things that we can apply to the machine [2,9]. CNN is an example that is obtained by the human biological structure of a cortex. The neural network has neurons connected; the weight gets high when the large size of the input. The CNN decreases the parameter with lesser shared weights and down sampling [1,7]



#### Figure 2.2 : Block on ConvNET (Convolution neural network)[3]

In Fig 2.2, it represents the convoluted layers (ConvNET). It is created in such a way that it is having nexus of layers [2]. There is a repetition of convolutional, ReLU, and pooling layers [6]. The system is planned in a way where filters are learning automatically. ReLU layer is used to activate to learn nonlinear combinations that are known ad feature extraction [2,3]. This phenomenon is called activation, where the output of the first layer becomes the input of the second layer.

The use of the pooling layer is to consolidate the feature of the local images by decreasing their inputs. The feature that we obtain by the above process 9 becomes the inputs to the regression function [2,3]. The last layer is the combination of classifier and regression layer to solve classification and object localization problems [2].

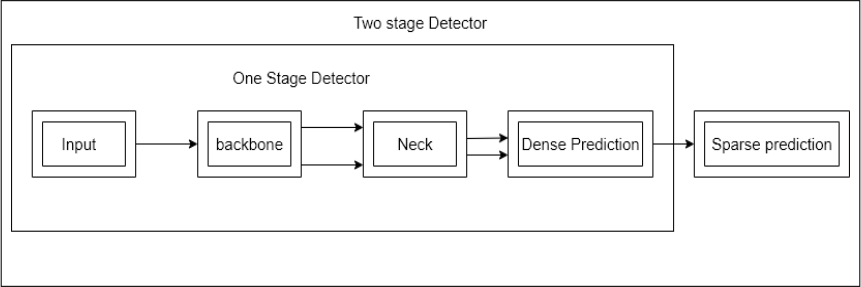
#### Object detection:

The above two processes are for particular objects, but we have multiple objects in any given image in real life. They may have multiple objects place at different positions. In our project, we have to detect cars, traffic lights [2,8], and Vehicles. Several objects have to be taken into consideration for the self-driving car [11]. It has to classify and localize all the objects on the road. So, it creates boxes for each identified object, as shown in the above figure [8]. The boxes around the cars are called the bounding box. In drawing boxes to get better accuracy, we use an instance segmentation algorithm to extract the complete, accurate object from the given picture. The bounding boxes help to get the exact shape of the object [8].

The objects in the detection algorithms depend on the feature. for example, we have birds, animals, and humans. They have their shape and features. The classification is based on the unique features, vertices, and angles [8]. Then the question arises that how the system understands the features of objects required to train the model. The classification depends on the training algorithms. In training, when we feed the features and shape of a given object, it learns every angle and edge of the object, and it then predicts the testing objects showing the class of the objects.

For example, we have images of dogs and cats. We will train the model by giving hundreds of images of cats and dogs labeled in the images. The system will understand each shape and edge of both the animals. When we run the model on the animals, it will identify the cats and dogs in the video. There are many algorithms and techniques to detect objects, but the trending is machine learning and deep learning methods to get the detection operations done right now. Object detection using deep learning is Histogram of oriented gradients, Scale-invariant feature transform, and Viola Janes object detection are few examples. RCNN, Yolo, and SSD are deep learning methods for detections [6,8].Object detection has many uses in real life; many applications require face detection. Now to open a mobile phone, we need face or IRIS recognition. Many face details explain applications about age, origin, gender, and many more things. They are now used as a security password for lots of operations. They are used in self-driving cars to provide critical data to decide when running on the roads [8]. Optical character recognition is used to convert the scanned book pages into an electronic document. For example, they can help automatically fetching data from the ID into the form.

It can be used in robotics, where there is intelligence in mechanical work—for example, collecting garbage and sorting them according to recyclable and non-recyclable. The object detection can be a tracker for the employees in the office and students in school where they can track students' attendance. It can help in managing the traffic plans at the time of high traffic.



#### Figure 2.3: Object Detection [9]

As shown in Fig , the object detector goes through these operations [3]. First, we will give input as Image, Patches, Pyramid, and many more. The backbone can be ResNet-50, Darknet 53, or VGG16 [11]. There are many, and in this model of Yolo, we will be using Darknet53 [6]. The Neck is FPn, Bi FPN, and many more. Now we have separated the head into parts. One is dense prediction, where we use RPN, SSD, or Yolo, and the second one is Sparse prediction, where we use RFCN and Faster RCNN.

#### Yolo Neural Network Framework – Darknet:

Darknet is an open-source neural network framework which is written on CUDA and C and available at GitHub. The features of Darknet is Yolo [1,9] which is state of the art object detection [8]. The best part is it is fast, setup is easy, and supports CPU and GPU. When we say that Yolo is open source, we get lots of work on Yolo on websites. It is because there are few works done by people online [9]. In our model to run Yolo versions 3 and 4, we have used Darknet to train Yolo. In layman's language, we can say that it sets the architecture of the whole network. The creator of Darknet is the same person who created Yolo [9].

## Functional Requirements:

Functional Requirements Include:

* Video Input: The project should support input from a video file or a live video stream from a camera in the parking area.
* Vehicle Detection: The system should employ the YOLO, OpenCV, or R-CNN model to detect vehicles in each frame of the video.
* Vehicle Classification: The system should classify the detected vehicles into different categories (e.g., car, truck, motorcycle) using a CNN model trained for vehicle classification.
* Tracking: The system should track each detected vehicle across multiple frames to maintain continuity and prevent duplicate counts.
* Vehicle Counting: The system should count the number of vehicles entering and exiting the parking area. The counting algorithm should use the Euclidean distance method or a similar technique to determine the direction of movement (entering or exiting)
* Accuracy and Speed: The system should provide accurate and real-time vehicle detection and counting to ensure reliable monitoring of the parking area.
* User Interface: The project file should include a user interface that displays the video stream with the detected vehicles, their classifications, and the current count of vehicles in the parking area.
* Configuration Options: The user interface should allow the user to configure various parameters such as detection confidence threshold, vehicle classification model, tracking algorithm, and counting algorithm.
* Alerts and Notifications: The system should have the capability to generate alerts or notifications when the parking area reaches a certain capacity or when specific events occur (e.g., a vehicle stays parked for an extended period).
* Data Logging and Reporting: The system should log the vehicle counting data, including timestamps, counts, and vehicle classifications, and provide a reporting mechanism to analyze and export this data.
* Integration: The project file should include necessary integration with external components such as cameras, or other systems for seamless operations.
* Documentation: The project file should contain comprehensive documentation explaining the system's setup, cfg, and usage, including any dependencies and installation instructions for the required libraries and models.

Note that these requirements serve as a general guideline and can be customized based on specific project needs and constraints.

* + 1. **Non – Functional Requirements:**

Non-functional requirements may include the following:

* Performance:
  + Accuracy: The system should achieve high accuracy in vehicle detection and counting.
  + Speed: The system should perform real-time or near-real-time detection and counting of vehicles.
  + Scalability: The system should be capable of handling varying traffic loads and scale accordingly.
  + Resource utilization: The system should utilize system resources efficiently to optimize performance.
* Reliability:
  + Availability: The system should be available and operational for the desired duration, minimizing downtime.
  + Robustness: The system should be able to handle different lighting conditions, weather conditions, and vehicle types.
  + Error handling: The system should handle errors gracefully and provide appropriate error messages or recovery mechanisms.
* Security:
  + Data privacy: The system should ensure the privacy and security of any captured or processed data, such as license plate numbers.
  + Access control: The system should have appropriate access controls to prevent unauthorized access or modification.
* Maintainability:
  + Modularity: The system should be modular and well-organized, allowing for easy maintenance and future enhancements.
  + Documentation: The system should have clear and comprehensive documentation, including code comments, user manuals, and technical guides.
  + Code readability: The system's code should be written in a clear and understandable manner, following coding best practices.
* Usability:
  + User interface: The system should have a user-friendly interface that allows users to interact easily and efficiently.
  + Configurability: The system should provide options to configure parameters such as detection sensitivity or counting thresholds.
  + Compatibility: The system should be compatible with different operating systems, browsers, or hardware configurations.
* Integration:
  + API compatibility: The system should integrate smoothly with external APIs or services, such as cloud storage or databases.
  + Interoperability: The system should be able to work seamlessly with other existing systems or components, such as parking management systems.
* Ethical Considerations:
  + Fairness: The system should be designed and trained to avoid biased behavior or discriminatory actions towards specific vehicle types, demographics, or individuals.
  + Transparency: The system should provide insights into its decision-making process and ensure transparency in how it operates.
  + Compliance: The system should adhere to legal and regulatory requirements regarding privacy, data protection, and surveillance.

These non-functional requirements aim to ensure that the vehicle counting and detection system is accurate, reliable, secure, maintainable, usable, integrated with other systems, and considers ethical consideration.

* + - 1. **Hardware Requirements:**

The hardware requirements can vary depending on the scale of the project, the desired frame rate, and the level of accuracy required. However, here are some general hardware recommendations:

* CPU: A modern multi-core processor is recommended to handle the computational load efficiently. A processor with a high clock speed and multiple cores, such as an Intel Core i7 or AMD Ryzen 7, would be suitable. For larger-scale deployments, a server-grade processor or a cluster of machines may be necessary.
* GPU: Since the project involves deep learning models like YOLO, CNN, and R-CNN, a powerful GPU is highly recommended to accelerate the training and inference processes.
* RAM: Sufficient system memory is essential to accommodate the data and intermediate results during the processing pipeline. At least 8 GB of RAM is recommended, but for larger projects or handling multiple video streams simultaneously, 16 GB or more would be better.
* Storage: Adequate storage is necessary to store the project files, dataset, trained models, and any recorded video footage. A solid-state drive (SSD) is preferable over a traditional hard disk drive (HDD) for faster read/write speeds, which can be beneficial for real-time processing.
* Camera: High-resolution cameras capable of capturing clear images or video feeds are required for accurate vehicle detection and counting. The camera specifications will depend on the specific requirements of your project, such as the desired field of view, frame rate, and lighting conditions. Ensure that the camera output is compatible with the chosen framework (e.g., OpenCV).
  + - 1. **Software Requirements:**

Install Python on your computer system:

* Install Image AI and its dependencies like TensorFlow, NumPy, OpenCV, etc.
* Download the Object Detection model file (COCO Dataset).
* Download Yolo weights and configuration files.

#### NumPy

It's a Python package that helps with multi-dimensional arrays and matrices. To work with these arrays, it also has high-level mathematical algorithms. In all academic domains, it is utilised to analyze pipelines. Because of NumPy [12], the scientific Python ecosystem has grown tremendously. It serves as the interface between libraries and APIs. It's a fantastic tool for mathematical and scientific study. Images are used in our project and will be transformed to NumPy arrays. The labels are later applied to the image dataset.

#### OpenCV

The entire form of OpenCV is an open-source computer vision library; we're utilizing it since we need to install a webcam in the car to identify things in real-time for our project [5]. It's a computer vision library that works in real time [3]. Intel developed this library. This library has the advantage of being open-source and cross-platform. [4] Gaussian blur Method, Canny Edge Detection, and Hough Space are just a handful of the OpenCV algorithms available.

#### PyTorch

PyTorch is an open-source machine learning (ML) framework based on the Torch library and the Python programming language. It is one of the most popular deep learning research platforms. The framework is designed to accelerate the transition from research prototyping to implementation. Tensor computation and functional deep neural networks are two of PyTorch's most notable capabilities [13].

#### Dataset

Data collecting is an important aspect of the research in order to create an effective model. The accuracy of the model is heavily influenced by the data [5]. Microsoft created and maintained the COCO dataset, which we used. Its full form is common objects in context. Although the COCO dataset has 80 different object classes, we are only concerned in the human class [14]. As a result, we used a tool called FiftyOne to extract the images with the human label and create our own dataset from them.

1. **YOLO (You Only Look Once):**

YOLO is also an object detection algorithm which uses only one convolutional network to predicts the bounding boxes and the class probabilities and thus YOLO differs from other region based algorithms [1].

All the previous object detection algorithms have used regions to localize the object within the image. The network does not look at the complete image. Instead, parts of the image which has high probabilities of containing the object. YOLO or You Only Look Once is an object detection algorithm much is different from the region based algorithms which seen above [5]. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

YOLO is a convolutional neural network that allows to detect and classify objects in the form of bounding boxes. Such bounding box is the minimum sized rectangle, which will contain the whole found object. YOLO works on the principle of Single Shot. This means that the network architecture is arranged in such a way that in one pass of the frame, all objects are detected simultaneously [1].

The neural network has a large set of already trained classes, which can help to test all the capabilities of YOLO, as well as the performance of hardware. Also, YOLO can be trained for any class of images, if you correctly select the source data. To train YOLO, a large dataset is required. Each image should be provided with a text file with the marked regions of the trained class of objects. You also need a file with initial weights and a file with system information, which shows the paths to the images and the path, through which the recovery points will be recorded. Recovery points are the files of the weights at a certain training step. Weighting files are written to permanent memory every 100 iterations, which allows you to interrupt training at any time, and then continue with the last received weight.

The longer the network is trained, the better the detection quality will be. YOLO works by taking an image and split it into an SxS grid, within each of the grid we take m bounding boxes. For each of the bounding box, the network gives an output a class probability and offset values for the bounding box[1]. The bounding boxes have the class probability above a threshold value is selected and used to locate the object within the image. YOLO is orders of magnitude faster(45 frames per second) than any other object detection algorithms [15].

The limitation of YOLO algorithm is that it struggles with the small objects within the image, for example, it might have difficulties in identifying a flock of birds. This is due to the spatial constraints of the algorithm.

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.
    - 1. **Usability Requirements:**

Here are some key usability requirements we have to consider:

* Ease of Installation and Setup: Ensure that the project file includes clear instructions and documentation on how to install and set up all the necessary dependencies, libraries, and frameworks. Provide step-by-step guidance to make it easy for users to get started with the project.
* User-Friendly Interface: Design a user interface that is intuitive, visually appealing, and easy to navigate. Users should be able to interact with the system without any confusion or complexity.
* Configuration Options: Allow users to configure various parameters and settings of the system easily. This might include options to adjust detection thresholds, define counting zones, specify camera inputs, or select different pre-trained models.
* Real-time Performance: Optimize the implementation to ensure real-time processing of video feeds from cameras. The system should be capable of efficiently handling the vehicle detection and counting tasks within an acceptable time frame.
* Accuracy and Reliability: Strive to achieve accurate vehicle detection and counting results. The system should minimize false positives and negatives, providing reliable data to users. Thoroughly test the algorithms and models to ensure their performance meets the required standards.
* Error Handling and Logging: Implement robust error handling mechanisms to handle unexpected scenarios gracefully. Log any errors, warnings, or critical events encountered during the system's operation to assist in troubleshooting and debugging.
* Compatibility:Ensure compatibility with various operating systems, such as Windows, macOS. Consider cross-platform support to make the project file accessible to a wider range of users.
* Documentation and Support: Provide comprehensive documentation that includes clear instructions on how to use the project file effectively. Offer support channels, such as a dedicated email address, forum, or FAQ section, where users can seek assistance or clarification.
* Scalability: Design the system in a way that allows easy scalability. Users should be able to add more cameras, expand the parking area coverage, or integrate the solution into larger systems without significant modifications or performance degradation.
* Security and Privacy: Address security concerns by ensuring the project file adheres to best practices for data protection, privacy, and access control. Implement mechanisms to safeguard user data and prevent unauthorized access to the system.
  + - 1. **Security Requirements:**

When considering the security requirements several key requirements should betaken into account. Here are some common security requirements for such a project:

* Data Privacy and Protection: Ensure that sensitive data, such as license plate numbers or personal information, is handled securely and protected against unauthorized access or disclosure. Implement encryption and access controls to safeguard data.
* Authentication and Authorization: Implement a secure authentication mechanism to ensure that only authorized individuals can access the system and perform specific actions. This helps prevent unauthorized usage and protects against potential attacks.
* Secure Communication: Employ secure communication protocols (e.g., HTTPS) to transmit data between the system components and any external interfaces. This ensures data integrity and confidentiality during transmission.
* Intrusion Detection and Prevention: Implement mechanisms to detect and prevent unauthorized access attempts or malicious activities within the system. This can involve techniques like intrusion detection systems (IDS), firewalls, and anomaly detection algorithms.
* System Monitoring and Logging: Implement logging mechanisms to record system activities, including user actions, errors, and security-related events. Regularly monitor logs to identify any suspicious or abnormal behavior.
* Secure Software Development Practices: Follow secure coding practices to minimize vulnerabilities and reduce the risk of exploitation. This includes input validation, output sanitization, and secure configuration of the underlying technologies.
* Regular Updates and Patch Management: Keep all software components, frameworks, and libraries up to date with the latest security patches. Regularly check for security advisories and apply necessary updates promptly to address known vulnerabilities.
* Physical Security: Consider physical security measures for the hardware infrastructure that hosts the system, such as servers and storage devices. This includes access controls, surveillance systems, and protection against physical tampering.
* Backup and Disaster Recovery: Implement regular backups of critical data and establish a disaster recovery plan. This ensures that the system can be restored in case of data loss, system failures, or other unforeseen events.
* User Access Management: Define and enforce user roles and permissions to restrict access based on the principle of least privilege. This prevents unauthorized access to sensitive functionalities or data.
  1. **Feasibility Study:**

A feasibility study is conducted to assess the viability and practicality of aproposed project or system. In this case, the feasibility study aims to evaluate the implementation of a vehicle counting and detection system for a parking area using YOLO (You Only Look Once), OpenCV (Open Source Computer Vision Library), CNN (Convolutional Neural Network), R-CNN (Region-based Convolutional Neural Network), and a counting algorithm based on the Euclidean distance method.

* + 1. **Technical Feasibility:**

Technical feasibility assesses whether the proposed system can be developed using the required technologies. In this case, the system utilizes YOLO, OpenCV, CNN, R-CNN, and a counting algorithm based on the Euclidean distance method. Here are the key points to consider:

* Availability of Technology: YOLO, OpenCV, and CNN are well-established technologies widely used in computer vision tasks, including object detection and counting. R-CNN, although older, can still be used effectively. These technologies are readily available and well-documented.
* Integration: The integration of YOLO, OpenCV, CNN, R-CNN, and the counting algorithm is feasible. OpenCV provides a comprehensive set of computer vision functions, and YOLO, CNN, and R-CNN models can be integrated with OpenCV for object detection. The counting algorithm based on the Euclidean distance method can be implemented using mathematical calculations with the detected object positions.
* Computational Requirements: Object detection and counting can be computationally intensive, especially when processing large video streams. The hardware and computational resources, such as CPUs or GPUs, need to be considered to ensure real-time performance, especially if the system is intended for real-time monitoring.
* Accuracy and Reliability: YOLO, CNN, and R-CNN models have demonstrated high accuracy and reliability in object detection tasks. However, the accuracy of counting based on the Euclidean distance method may depend on factors such as occlusion, varying sizes, and environmental conditions.

Overall, the technical feasibility of implementing the proposed system appears feasible. However, thorough testing and optimization would be required to ensure real-time performance and accuracy under different conditions.

* + 1. **Operational Feasibility:**

Operational feasibility evaluates whether the proposed system can be effectively integrated into the existing operational environment. Consider the following aspects:

* System Integration: The vehicle counting and detection system should seamlessly integrate into the parking area infrastructure, including cameras or sensors for capturing video or image data. The system should be compatible with the existing hardware and software infrastructure.
* User Interface: The system should provide a user-friendly interface for monitoring and managing the vehicle counting and detection process. Operators or administrators should be able to easily access and interpret the data generated by the system.
* Scalability: The system should be designed to handle various parking areas of different sizes and configurations. It should be scalable to accommodate increasing numbers of cameras or sensors and provide accurate counting results.
* Maintenance and Support: Adequate technical support and maintenance should be available for the system, including regular updates, bug fixes, and troubleshooting assistance.

Taking these factors into account, the proposed system can be operationally feasible with proper integration, user-friendly interfaces, scalability options, and reliable support.

* + 1. **Economical Feasibility:**

Economical feasibility evaluates whether the proposed system is financially viable and provides a positive return on investment. Key considerations include:

* Cost of Development: The development cost includes the acquisition or development of the necessary s/w, h/w, and associated infrastructure. It also involves the cost of expertise to implement and customize the system.
* Operational Costs: Operational costs include maintenance, support, and ongoing monitoring of the system. It may also include the cost of hardware upgrades or replacements as technology advances.
* Return on Investment (ROI): The benefits of the system, such as improved parking management, increased efficiency, and enhanced security, need to be evaluated against the costs incurred. The potential revenue generation or cost savings resulting from effective parking area management should

## CHAPTER 3

**SYSTEM ANALYSIS AND DESIGN**

In this chapter we will see the implementation of the work that how it uses the algorithms and how to deploy in the environment. In this Deep learning, a subset of machine learning which in turn is a subset of artificial intelligence (AI) has networks capable of learning things from the data that is unstructured or unlabeled.

**3.1. System Analysis:**

First, we installed darknet. Then we split the dataset into two parts, testing and training. We can train data on our model, and after that, we will do the testing. The following things are required for the training and experiment to be conducted, we have separated existing methods used for training and experiment with our proposed method so that it will be easy to understand and differentiate,

#### Existing Methods:

There are several machine learning and deep learning algorithms for object detection. When machine learning approaches are used for detection, it is mandatory to define the features first. Deep learning approaches does not demand to specify the features instead they perform end-to-end detection. Machine learning methods use Support Vector Machine (SVM) and deep learning methods use CNN. R-CNN, fast R-CNN, faster R-CNN [1] are some common algorithms for object detection[2].

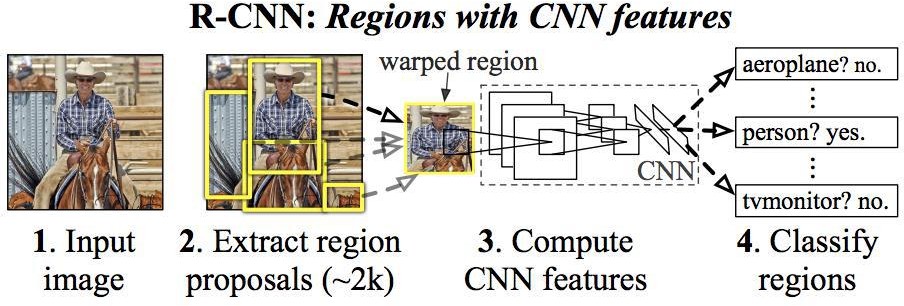
#### R-CNN:

To circumvent the problem of selecting a huge number of regions, Ross Girshick et al. proposed a method where we use the selective search for extract just 2000 regions from the image and he called them region proposals [4]. Therefore, instead of trying to classify the huge number of regions, you can just work with 2000 regions. These 2000 region proposals are generated by using the selective search algorithm which is written below [15]. Selective Search:

* Generate the initial sub-segmentation, we generate many candidate regions
* Use the greedy algorithm to recursively combine similar regions into larger.
* Use generated regions to produce the final candidate region proposals

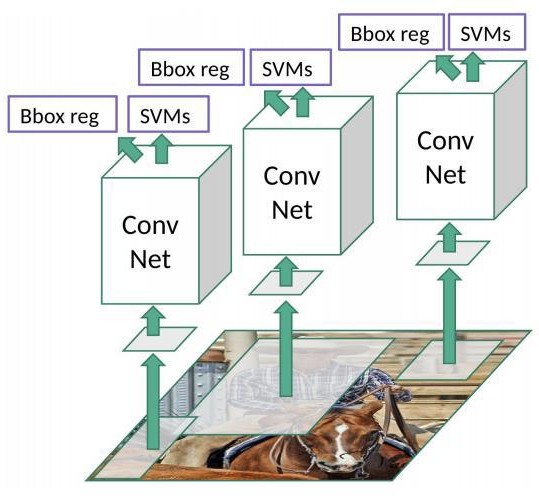
R-CNN uses selective search. By this, bounding boxes are generated. Then, for each bounding box, image classification is done through CNN [4]. Finally, each bounding box are refined using regression [7]. The problems with R-CNN are:

* It takes a huge amount of time to train the network as it requires classification of 2000 region proposals per image. ·
* It cannot be implemented real time due to the time constraints. ·
* Since the selective search algorithm is a fixed algorithm, no learning is happening at that stage.



#### Figure 3.1 R-CNN : Regions with CNN features [15]

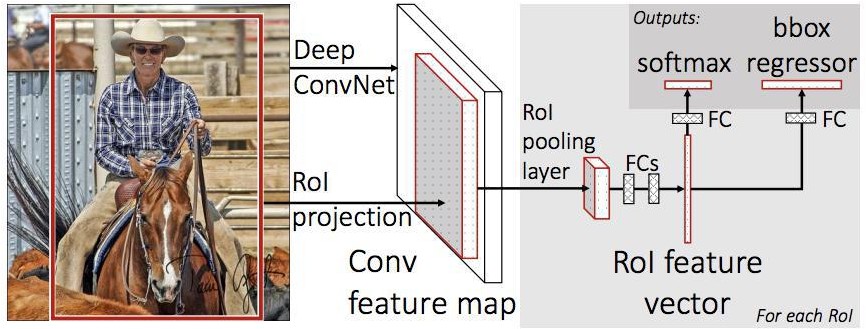
These 2000 candidate regions which are proposals are warped into a square and fed into a convolutional neural network that produces a 4096-dimensional feature vector as output[4]. The CNN plays a role of feature extractor and the output dense layer consists of the features extracted from the image and the extracted features are fed into an SVM for the classify the presence of the object within that candidate region proposal. In addition to predicting the presence of an object within the region proposals, the algorithm also predicts four values which are offset values for increasing the precision of the bounding box. For example, given the region proposal, the algorithm might have predicted the presence of a person but the face of that person within that region proposal could have been cut in half [4]. Therefore, the offset values which is given help in adjusting the bounding box of the region proposal.



**Figure 3.2 R-CNN Working [3]**

#### Fast R-CNN

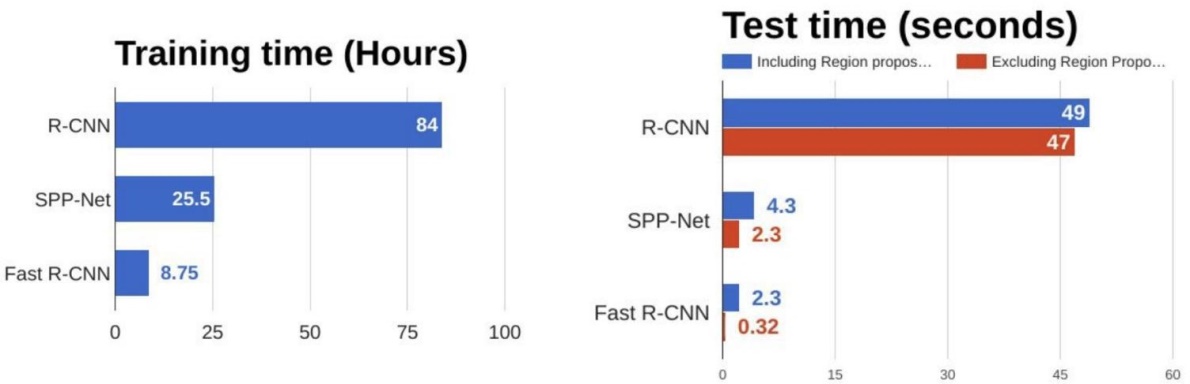
In fast R-CNN, the input image is fed into the CNN. Then, the region of proposals are identified and wrapped into squares[4]. Then, reshaping of regions is done using a RoI pooling layer. Then, a softmax layer is used to predict the class of the proposed region[3]. The problem with this fast R- CNN is: · Performance degradation during testing.



#### Figure 3.3.3 Fast R-CNN [10]

The same author of the previous paper(R-CNN) solved some of the drawbacks of R-CNN to build a faster object detection algorithm and it was called Fast R-CNN. The approach is similar to the R- CNN algorithm. But, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map[3]. From the convolutional feature map, we can identify the region of the proposals and warp them into the squares and by using an RoI pooling layer we reshape them into the fixed size so that it can be fed into a fully connected layer. From the RoI feature vector, we can use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box[4].

The reason “Fast R-CNN” is faster than R-CNN is because you don‟t have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is always done only once per image and a feature map is generated from it.



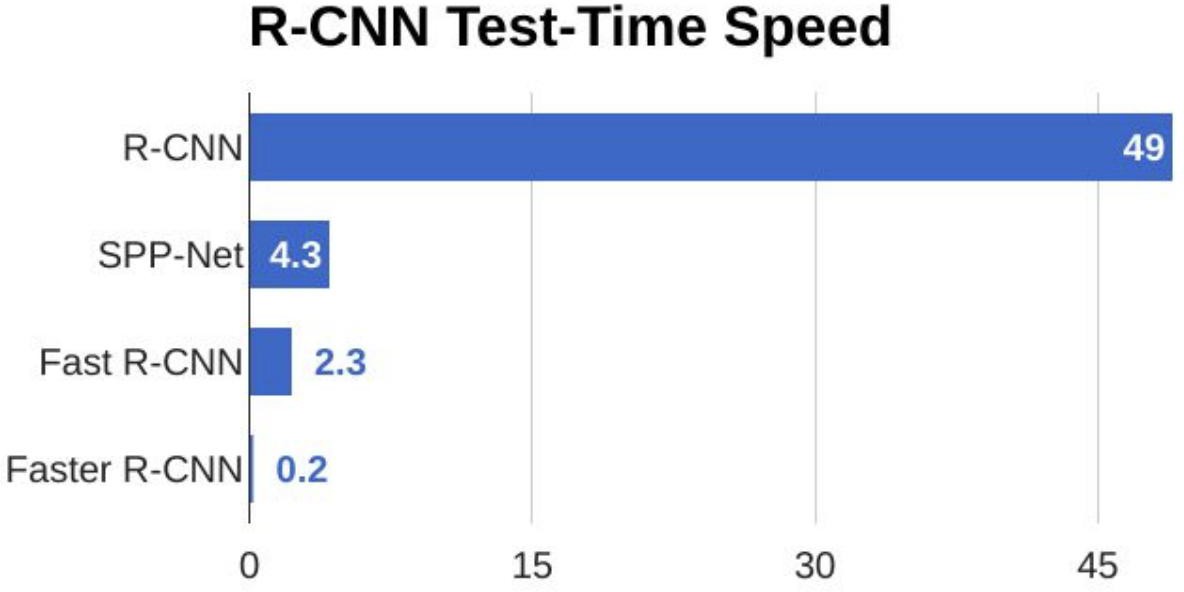
#### Figure 3.4 Comparison of object detection algorithm[10]

From the above graphs, you can infer that Fast R-CNN [6], is significantly faster in training and testing sessions over R-CNN. When you look at the performance of Fast R-CNN during testing time, including region proposals slows down the algorithm significantly when compared to not using region proposals. Therefore, the region which is proposals become bottlenecks in Fast R-CNN algorithm affecting its performance.

#### Faster R-CNN

Both of the above mentioned algorithms uses selective search which is a slow and time-consuming process. Faster R-CNN does not use selective search. It uses a separate network and by that it produces region proposals [5]. All of the previously explained algorithms use region based approach to detect the object in the image without looking at the complete image.

Both of the above algorithms(R-CNN & Fast R-CNN) uses selective search to find out the region proposals. Selective search is the slow and time-consuming process which affect the performance of the network. Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using the selective search algorithm for the feature map to identify the region proposals, a separate network is used to predict the region proposals[8]. The predicted the region which is proposals are then reshaped using an RoI pooling layer which is used to classify the image within the proposed region and predict the offset values for the bounding boxes.



#### Figure 3.5 Comparison of test- time speed of object detection algorithms [10]

From the above graph, you can see that Faster R-CNN is much faster than it’s predecessors [8]. Therefore, it can even be used for real-time object detection.

## Proposed method:

Vehicle detection system model This paper implements Vehicle detection based on the YOLO algorithm. The YOLO algorithm creatively predicts n boxes in each area on the area where the feature map is finally output 77[11]. These boxes have different sizes and positions, covering almost all possible target positions.

1. **Eucledian Distance for tracking:**

The tracker basically uses the Euclidean\_distance concept to keep track of an object. It calculates the difference between two center points of an object in the current frame vs the previous frame, and if the distance is less than the threshold distance then it confirms that the object is the same object of the previous frame.

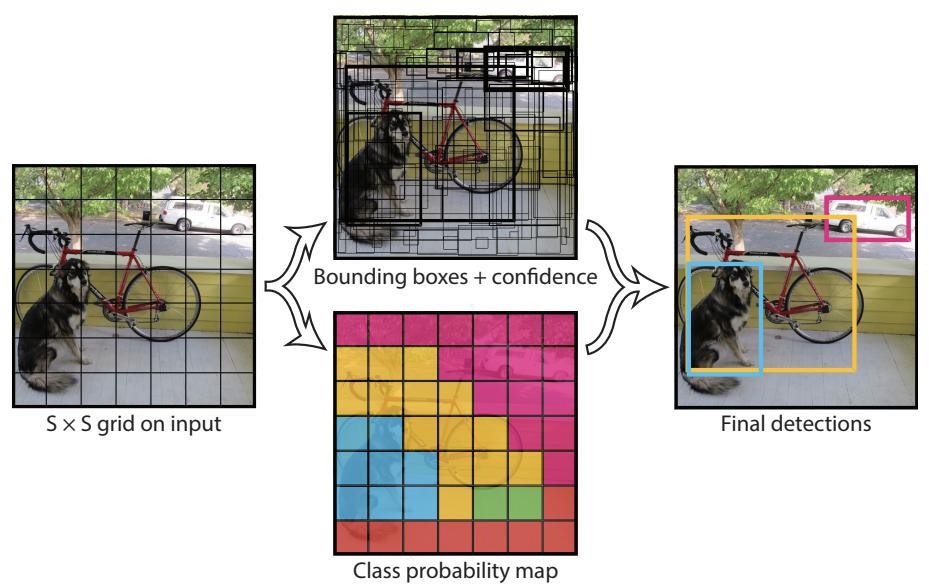
d = √[(x2 – x1)2 + (y2 – y1)2].

#### YOLO — You Only Look Once

YOLO is also an object detection algorithm which uses only one convolutional network to predicts the bounding boxes and the class probabilities and thus YOLO differs from other region based algorithms [1].

All the previous object detection algorithms have used regions to localize the object within the image. The network does not look at the complete image. Instead, parts of the image which has high probabilities of containing the object. YOLO or You Only Look Once is an object detection algorithm much is different from the region based algorithms which seen above [5]. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

YOLO is a convolutional neural network that allows to detect and classify objects in the form of bounding boxes. Such bounding box is the minimum sized rectangle, which will contain the whole found object. YOLO works on the principle of Single Shot. This means that the network architecture is arranged in such a way that in one pass of the frame, all objects are detected simultaneously [1].



#### Figure 3.6 YOLO object detection procedure [15]

The neural network has a large set of already trained classes, which can help to test all the capabilities of YOLO, as well as the performance of hardware. Also, YOLO can be trained for any class of images, if you correctly select the source data. To train YOLO, a large dataset is required. Each image should be provided with a text file with the marked regions of the trained class of objects. You also need a file with initial weights and a file with system information, which shows the

paths to the images and the path, through which the recovery points will be recorded. Recovery points are the files of the weights at a certain training step. Weighting files are written to permanent memory every 100 iterations, which allows you to interrupt training at any time, and then continue with the last received weight.

The longer the network is trained, the better the detection quality will be. YOLO works by taking an image and split it into an SxS grid, within each of the grid we take m bounding boxes. For each of the bounding box, the network gives an output a class probability and offset values for the bounding box[1]. The bounding boxes have the class probability above a threshold value is selected and used to locate the object within the image. YOLO is orders of magnitude faster(45 frames per second) than any other object detection algorithms [15].

The limitation of YOLO algorithm is that it struggles with the small objects within the image, for example, it might have difficulties in identifying a flock of birds. This is due to the spatial constraints of the algorithm.

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.

#### WORKING OF YOLO

YOLO trains and tests on full images and directly optimizes detection performance. YOLO model has several benefits over other traditional methods of object detection like the following. First, YOLO is extremely fast. Since frame detection in YOLO is a regression problem there is no need of complex pipeline. We can simply run our neural network on any new image at test time to make predictions. · Second, YOLO sees the entire image during training and testing unlike other sliding window algorithms which require multiple iterations to process a single image. Third, YOLO learns generalizable object representations [2,3]. When trained on real time images and tested, YOLO outperforms top detection methods like DPM and R-CNN.

YOLO network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and real time speeds while maintaining high average precision[2]. How the YOLO algorithm works. YOLO algorithm works using the following three techniques:

1. Residual blocks
2. Bounding box regression
3. Intersection Over Union (IOU)

These, techniques are explained below, with best examples

#### Residual blocks

First, the image is divided into various grids. Each grid has a dimension of S x S [8]. The following image shows how an input image is divided into grids.



#### Figure 3.7 Residual blocks [15]

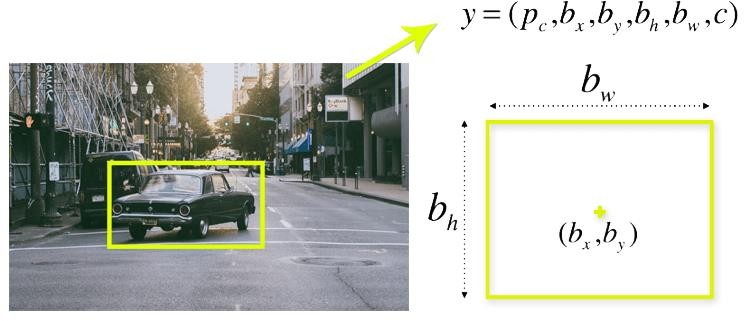
In the image above, there are many grid cells of equal dimension. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

#### 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 image shows an example of a bounding box. The bounding box has been represented by a yellow outline.

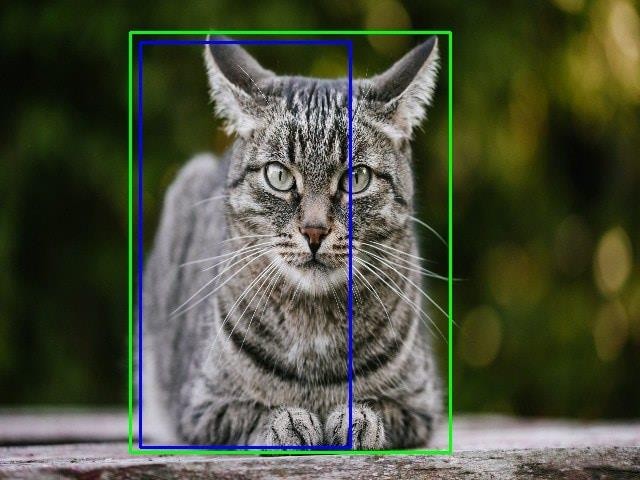


#### Figure 3.8 Single bounding box regression [5]

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 [8,9].

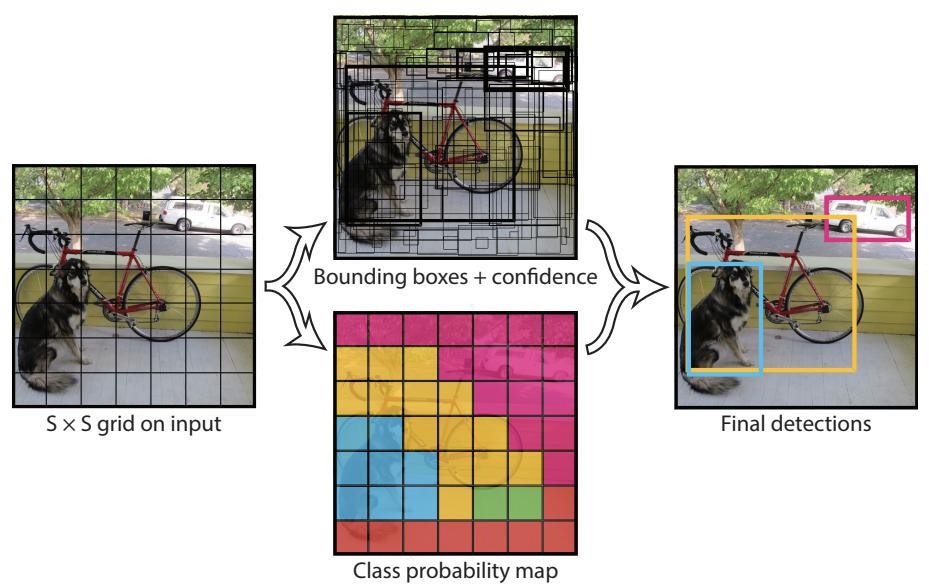
#### Intersection over union (IOU)

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 [6]. The following image provides a simple example of how IOU works.



#### Figure 3.9 IOU works[9]

In the image above, there are two bounding boxes, one in green and the other one in blue. The blue box is the predicted box while the green box is the real box. YOLO ensures that the two bounding boxes are equal. Combination of the three techniques The following image shows how the three techniques are applied to produce the final detection results.



#### Figure 3.10 Combination of three technique [15]

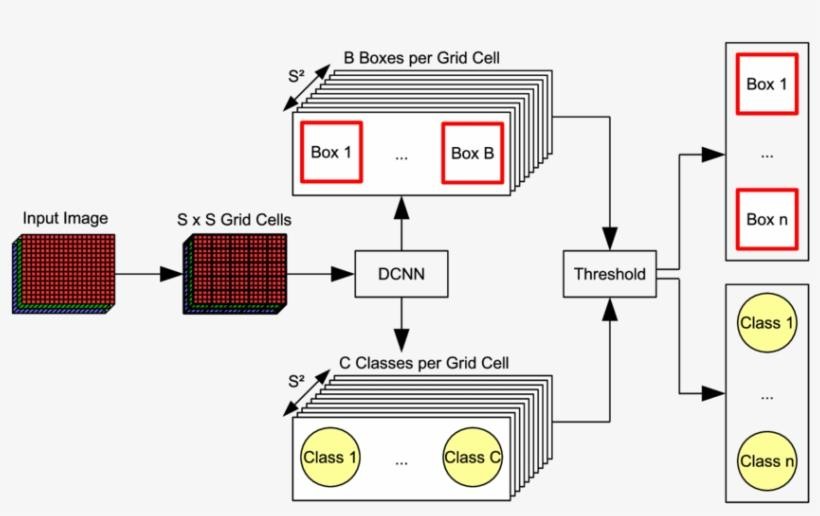
First, the image is divided into grid cells. Each grid cell forecasts B bounding boxes and provides their confidence scores. The cells predict the class probabilities to establish the class of each object. For example, we can notice at least three classes of objects: a car, a dog, and a bicycle. All 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 [15].

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

#### Applications of YOLO

* **Autonomous driving:** YOLO algorithm can be used in autonomous cars to detect objects around cars such as vehicles, people, and parking signals. Object detection in autonomous cars is done to avoid collision since no human driver is controlling the car [2,3].
* **Wildlife:** This algorithm is used to detect various types of animals in forests. This type of detection is used by wildlife rangers and journalists to identify animals in videos (both recorded and real-time) and images. Some of the animals that can be detected include giraffes, elephants, and bears [4,5].
* **Security:** YOLO can also be used in security systems to enforce security in an area. Let’s assume that people have been restricted from passing through a certain area for security reasons. If someone passes through the restricted area, the YOLO algorithm will detect him/her, which will require the security personnel to take further action [8].

#### ARCHITECTURE OF THE PROPOSED MODEL

The Figure shows the Architecture Diagram of the Proposed YOLO Model. Images are given as the input to the system. If Video can also be taken as input as it is nothing but a stream of images. As the name suggests You Only Look Once, the input goes through the network only once and the result of detected object with Bounding Boxes and Labels are obtained [15].

#### Figure 3.11 YOLO Architecture [11]

The images are divided into SXS grid cells before sending to the Convolutional Neural Network (CNN). B Bounding boxes per grid are generated around all the detected objects in the image as the result of the Convolutional Neural Network [8]. On the other hand, the Classes to which the objects belong is also classified by the Convolutional Neural Network, giving C Classes per grid. Then a threshold is set to the Object Detection. In this project we have given a Threshold of 0.3 [11]. Lesser the Threshold value, more number of bounding boxes will appear in the output resulting in the clumsy output.

#### YOLO DATAFLOW :

The fig.10 illustrates the Flow of data in the System. Initially User will be given the options to choose the type of the File to be given to the System as an input. Thus, User can either choose option of File Selection or start the Camera. In the former, User can choose either Image File or a Video File and, in the latter, User can start the Camera module. Once the input is selected Preprocessing is done, where the SXS grids are formed [8].

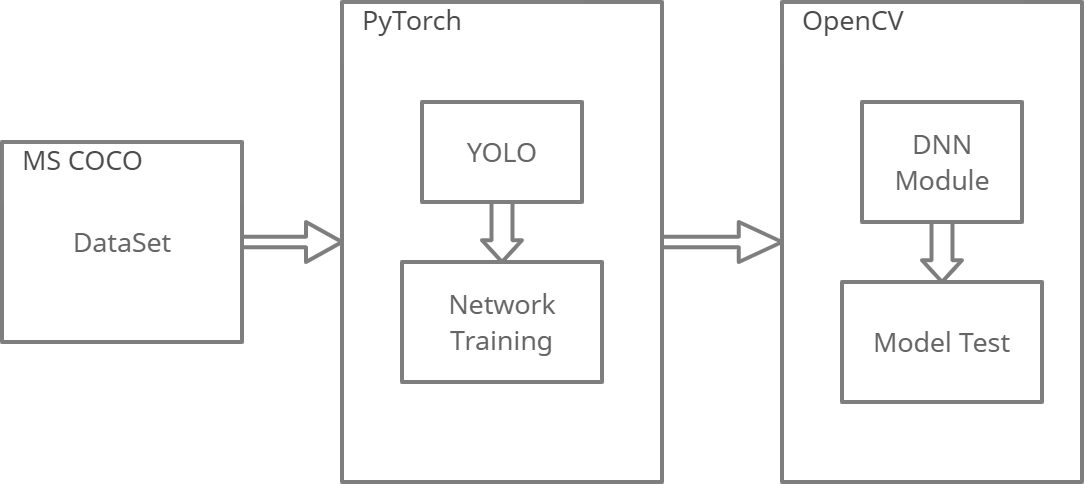
The resultant thus formed with the grids is send to the Bounding Box Prediction process where the Bounding Boxes are drawn around the detected objects. Next the result from the previous process is sent to the Class Prediction where the Class of the object to which it belongs is predicted. Then it is sent to the detection process where a Threshold is set in order to reduce clumsiness in the output with many Bounding Boxes and Labels in the final Output. At the end an image or a stream of images are generated for image and video or camera input respectively with Bounding Boxes and Labels are obtained as the Output[3].

#### Algorithm for Object Detection System:

* The input image is divided into SxS grid
* For each cell it predicts B bounding boxes Each bounding box contains five elements: (x, y, w, h) and a box confidence score
* YOLO detects one object per grid cell only regardless of the number bounding boxes
* It predicts C conditional class probabilities
* If no objects exists then confidence score is zero Else confidence score should be greater or equal to threshold value
* YOLO then draws bounding box around the detected objects and predicts the class to which the object belongs.
  1. **System Design:**

Figure shows the system architecture of object classification, which includes dataset, and training based on classifying the object on their morphologic content and Color, to which module that contains train, test and validation is built followed by processing unit, where the input is divided or framed on processing convenient upon which processing unit detects the object by localizing the position and display it out with classification specification [5].

Figure depicts the object classification system architecture, which includes the dataset, training based on classifying objects based on their morphologic content and color, to which a module containing train, test, and validation is built, followed by a processing unit, where the input is divided or framed for processing convenience, and the processing unit detects the object by localizing the position and displaying it with classification specifications. The neural network is trained on the dataset using PyTorch [13] framework and then the trained model is tested on user input using OpenCV.

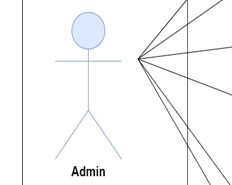
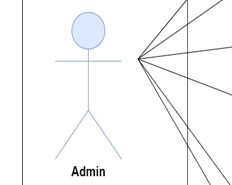


#### Figure 3.12 System Design[3]

Vehicle Detection Based on Traditional Method Traditional methods extract the main features describing Vehicles based on artificially designed feature extractors, then use these features to train classifiers to distinguish Vehicles and other things, and finally achieve the purpose of Vehicle detection. The specific process is shown in Figure [5]:

* + 1. **Use Case Diagram:**

This use case diagram represents functions which can be performed by Guard and Admin Department separately:



Guard Admin

**Figure 3.14 Use case Diagram**

* + 1. **ER Diagram:**

This ER Diagram shows relationship between Guard, Admin and operations/Functionalities they can perform

* + Guard: it has LoginID, User name, Login Password.
  + Admin: it has LoginID, User name, Login Password.
  + Algorithm: This has YOLO and other functions launch option.
  + Database: Here Total count and other information related vehicle movement is stored.

Admin

Login

Guard

Perform operation on

Access

Can Monitor

Output

Database

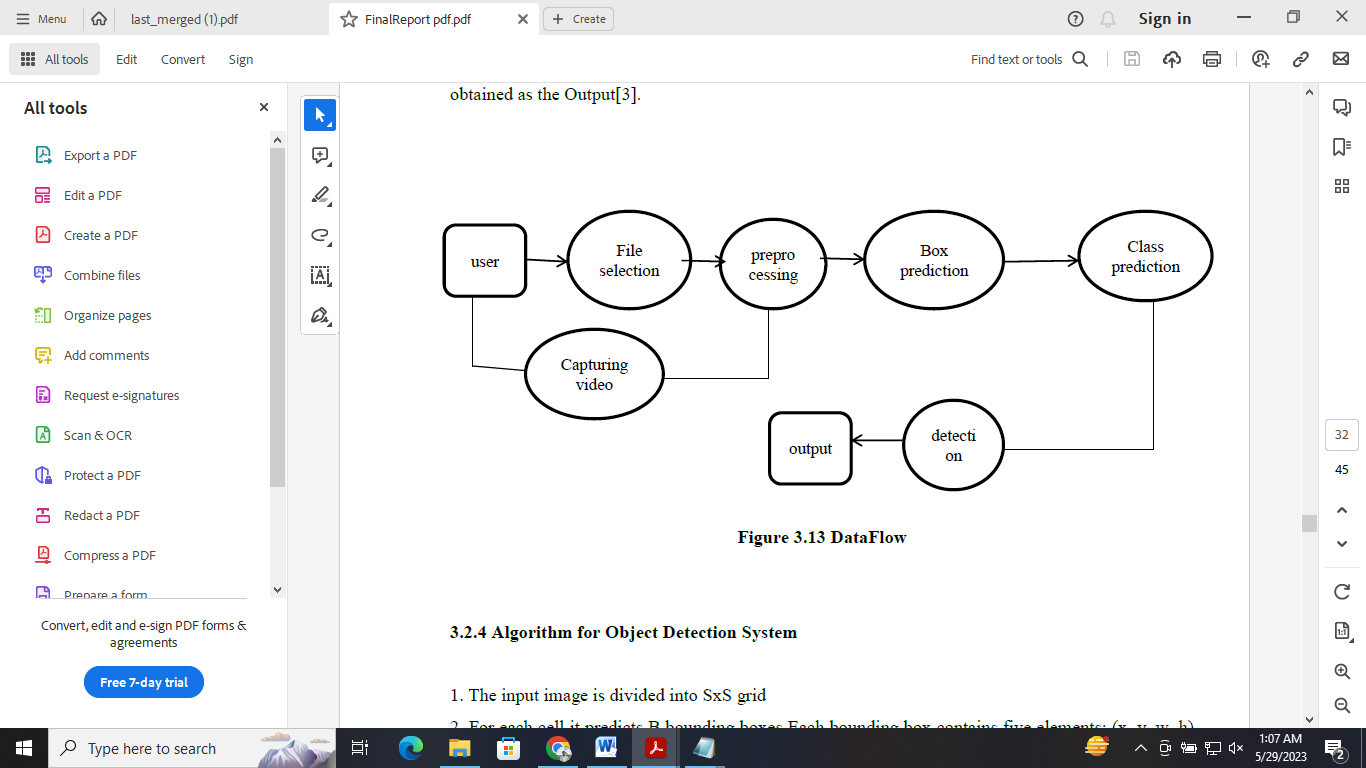
Algorithm

**Figure 3.15 ER Diagram**

* + 1. **Data Flow Diagram:**

Here we have the data flow diagram of whole algorithm, its working and execution,

* User login: Guard or admin will login by entering details.
* File selection: admin can set either real time data or manual file selection.
* Preprocessing: selected data will undergo preprocessing.
* Box prediction: box will be made with YOLO algorithm around selected objects.
* Class prediction: class will be predicted and displayed as label i.e. car, bike, etc.
* Detection: detection, classification and counting will be done.
* Output: final result will be displayed on the screen.

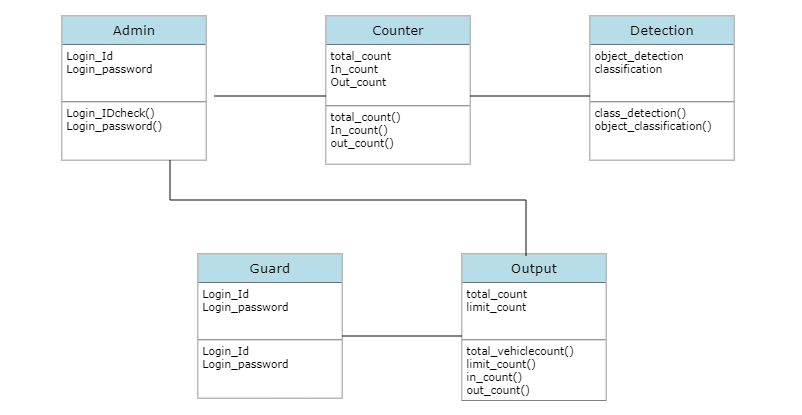
****

**Figure 3.16 Data Flow Diagram**

* + 1. **Class Diagram:**

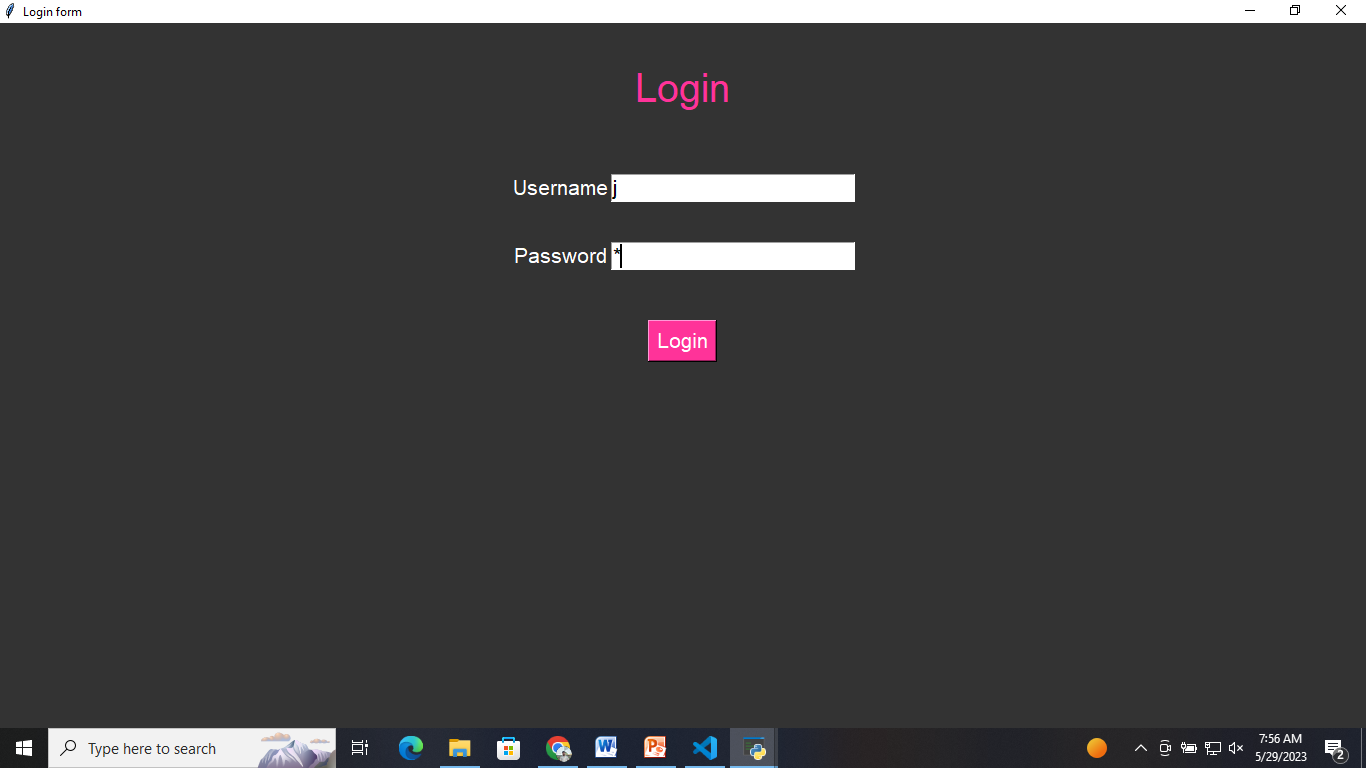
In the below class diagram;

* Admin: Can login and launch functions related to counting and detection and can access output on display and a database.
* Guard: Can login then access and monitor vehicle movement and data at same time.

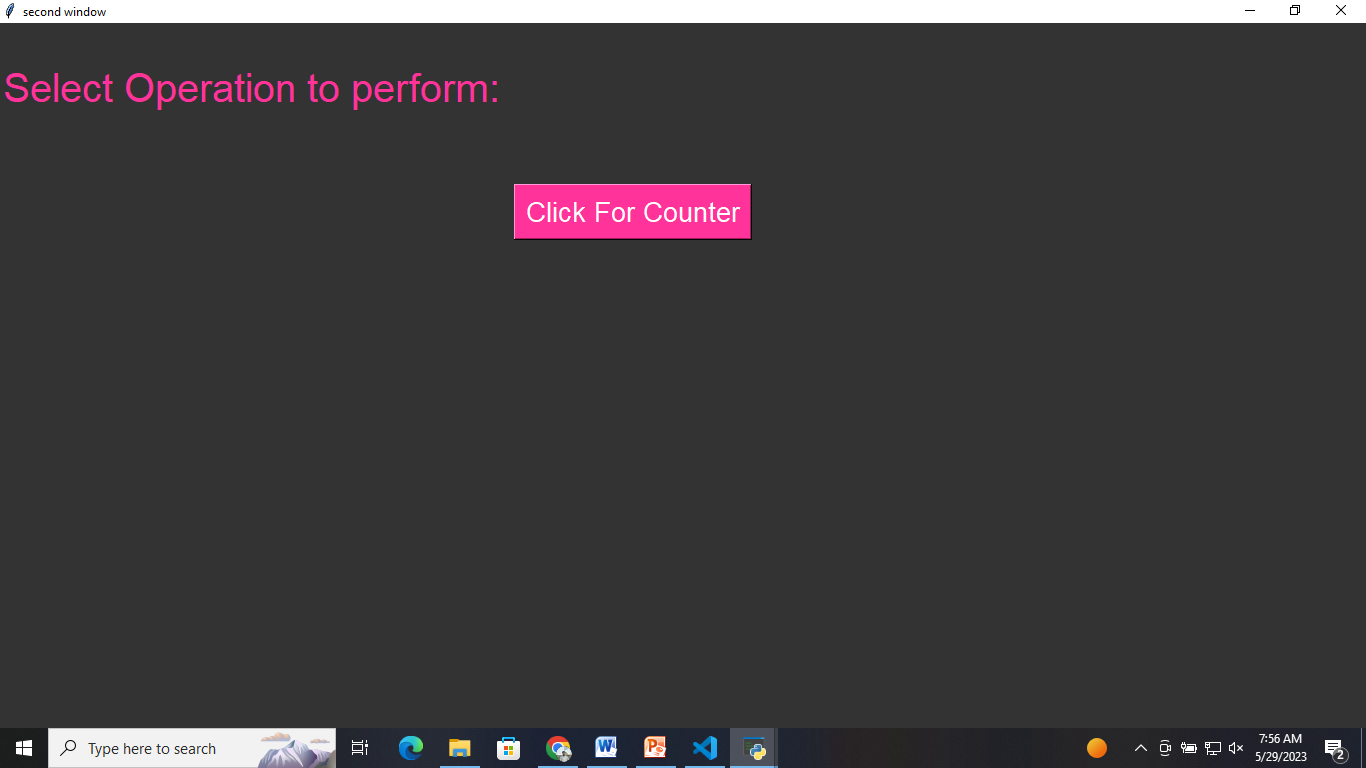
****

**Figure 3.17 Class Diagram**

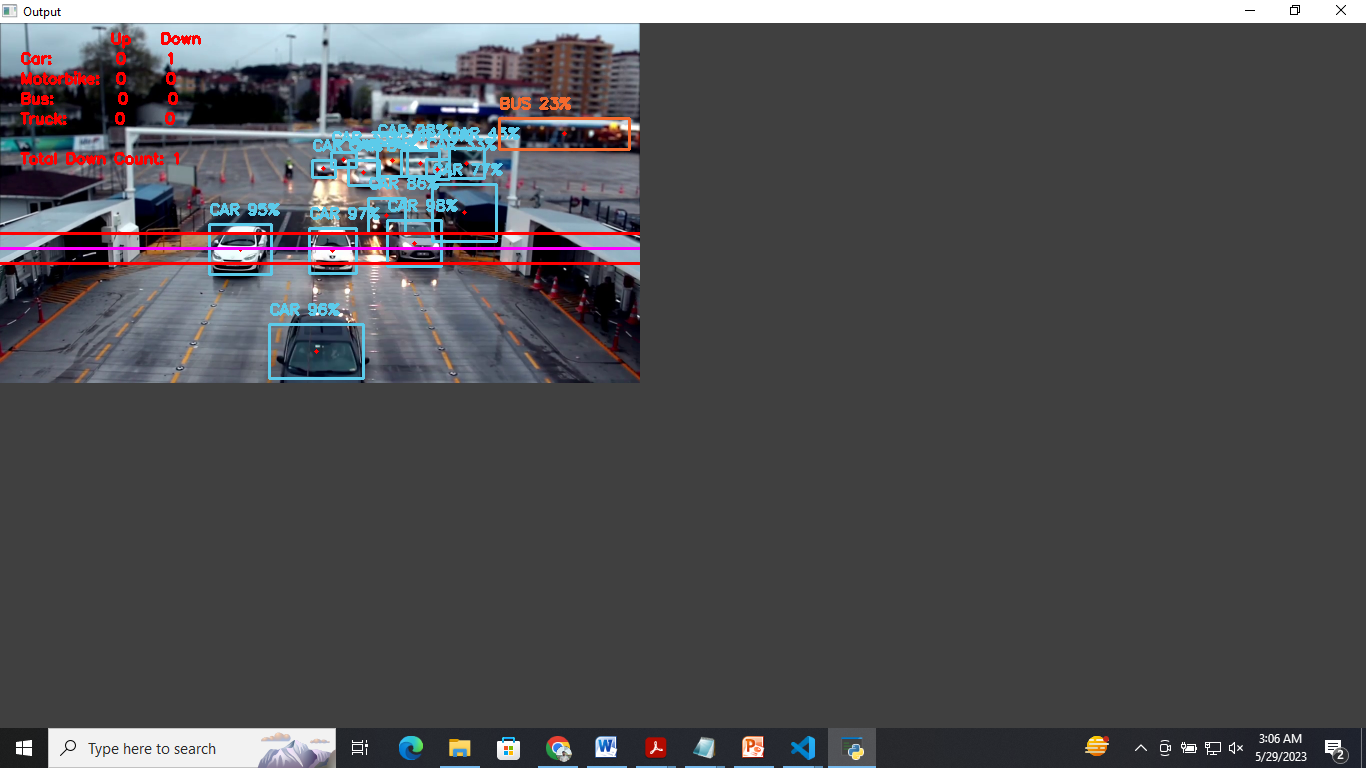
* + 1. **SNAPSHOTS**

****

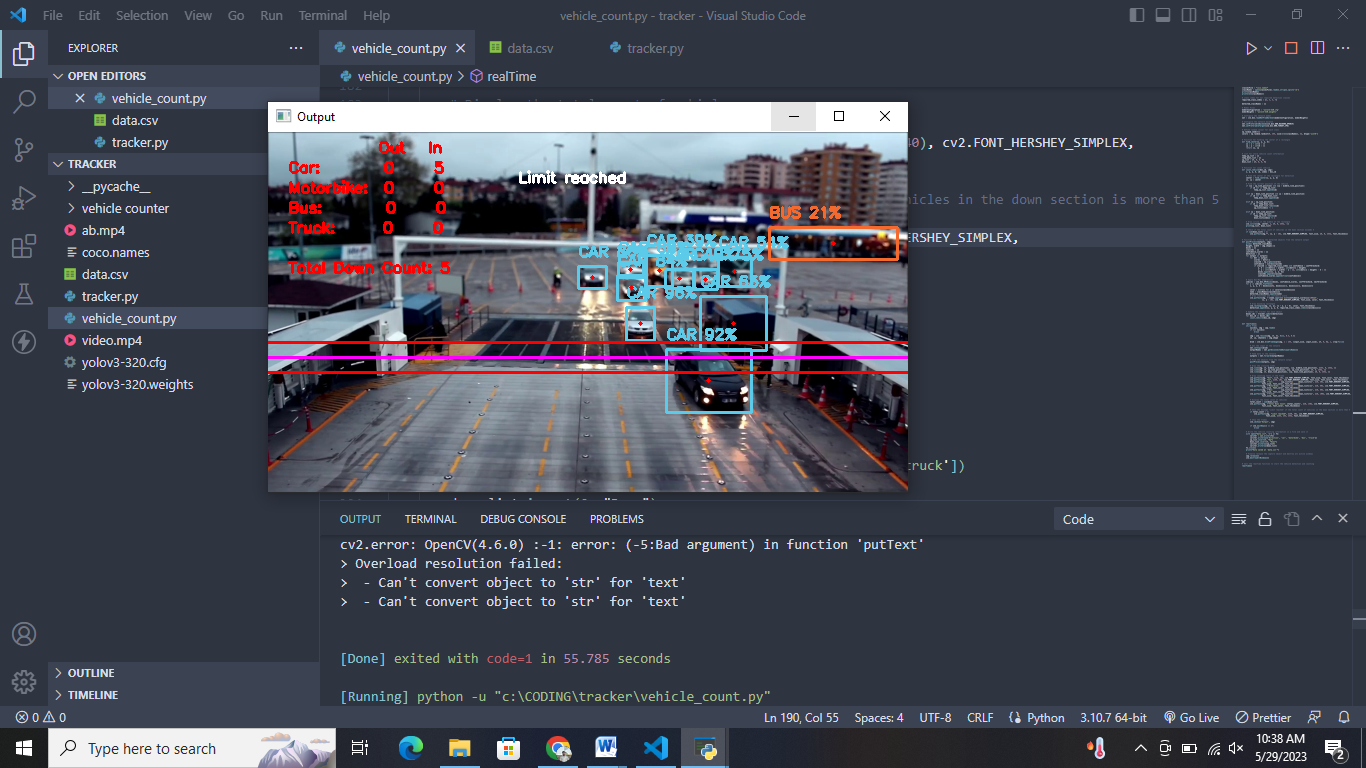
**Figure 3.18 Login page**

****

**Figure 3.19 Program Launch screen**

****

**Figure 3.20 Output Display 1**

****

**Figure 3.21 Output Display 2**

**CHAPTER 4**

# TESTING

* 1. **Technology Used:**

#### NumPy

It's a Python package that helps with multi-dimensional arrays and matrices. To work with these arrays, it also has high-level mathematical algorithms. In all academic domains, it is utilised to analyze pipelines. Because of NumPy [12], the scientific Python ecosystem has grown tremendously. It serves as the interface between libraries and APIs. It's a fantastic tool for mathematical and scientific study. Images are used in our project and will be transformed to NumPy arrays. The labels are later applied to the image dataset.

#### OpenCV

The entire form of OpenCV is an open-source computer vision library; we're utilizing it since we need to install a webcam in the car to identify things in real-time for our project [5]. It's a computer vision library that works in real time [3]. Intel developed this library. This library has the advantage of being open-source and cross-platform. [4] Gaussian blur Method, Canny Edge Detection, and Hough Space are just a handful of the OpenCV algorithms available.

#### PyTorch

PyTorch is an open-source machine learning (ML) framework based on the Torch library and the Python programming language. It is one of the most popular deep learning research platforms. The framework is designed to accelerate the transition from research prototyping to implementation. Tensor computation and functional deep neural networks are two of PyTorch's most notable capabilities [13].

#### Dataset:

Data collecting is an important aspect of the research in order to create an effective model. The accuracy of the model is heavily influenced by the data [5]. Microsoft created and maintained the COCO dataset, which we used. Its full form is common objects in context. Although the COCO dataset has 80 different object classes, we are only concerned in the human class [14]. As a result, we used a tool called FiftyOne (which is officially supported by the COCO dataset) to extract the images with the human label and create our own dataset from them.

#### YOLO (You Only Look Once)

YOLO is also an object detection algorithm which uses only one convolutional network to predicts the bounding boxes and the class probabilities and thus YOLO differs from other region based algorithms [1].

All the previous object detection algorithms have used regions to localize the object within the image. The network does not look at the complete image. Instead, parts of the image which has high probabilities of containing the object. YOLO or You Only Look Once is an object detection algorithm much is different from the region based algorithms which seen above [5]. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

YOLO is a convolutional neural network that allows to detect and classify objects in the form of bounding boxes. Such bounding box is the minimum sized rectangle, which will contain the whole found object. YOLO works on the principle of Single Shot. This means that the network architecture is arranged in such a way that in one pass of the frame, all objects are detected simultaneously [1].

The neural network has a large set of already trained classes, which can help to test all the capabilities of YOLO, as well as the performance of hardware. Also, YOLO can be trained for any class of images, if you correctly select the source data. To train YOLO, a large dataset is required. Each image should be provided with a text file with the marked regions of the trained class of objects. You also need a file with initial weights and a file with system information, which shows the paths to the images and the path, through which the recovery points will be recorded. Recovery points are the files of the weights at a certain training step. Weighting files are written to permanent memory every 100 iterations, which allows you to interrupt training at any time, and then continue with the last received weight.

The longer the network is trained, the better the detection quality will be. YOLO works by taking an image and split it into an SxS grid, within each of the grid we take m bounding boxes. For each of the bounding box, the network gives an output a class probability and offset values for the bounding box[1]. The bounding boxes have the class probability above a threshold value is selected and used to locate the object within the image. YOLO is orders of magnitude faster(45 frames per second) than any other object detection algorithms [15].

The limitation of YOLO algorithm is that it struggles with the small objects within the image, for example, it might have difficulties in identifying a flock of birds. This is due to the spatial constraints of the algorithm.

* 1. **Testing:**

Install the required dependencies first then, we need to make a config file with values that correspond to the classes. Every algorithm interprets the labelling in its own way. We may now begin the training. Now we must comprehend that when the average loss does not alter, we must cease training. If the loss does not diminish after several iterations, we should cease training. These results can be seen using mAP@.

Testing is the final phase in the process. We'll utilise the same images we used for training for testing after we've trained our own custom data set. The weights that we generate throughout training will be used. To test the model, we will utilise OpenCV, which allows us to receive user input and make predictions based on the model that has been trained on our dataset. Testing can be done in three ways. The first step is to supply an image (png, jpeg, etc.) and receive a labelled image with bounding boxes and accuracy percentage as an output. The second option is to insert a video as input, which will then label the Vehicles and produce a labelled video. The third form, which uses a camera to provide real-time input, is the most essential and widely used, all are categorized in following testings:

* + 1. **Unit Testing:**

Unit testing focuses on testing individual components or units of the system to ensure that they function correctly. unit testing would involve testing each of the following components independently:

* YOLO: Test the YOLO model's object detection accuracy by providing sample images or videos and verifying that the model can correctly identify and localize vehicles.
* OpenCV: Test OpenCV functions used for image processing, such as image resizing, cropping, and thresholding. Verify that the outputs of these functions are as expected.
* CNN and R-CNN: If you have implemented custom CNN or R-CNN models for vehicle detection, test them individually to ensure they can accurately detect vehicles.
* Counting Algorithm: Test the counting algorithm using the Euclidean distance method. Provide a set of test data, simulate vehicle movement, and verify that the algorithm can correctly count the vehicles.
  + 1. **Integration Testing:**

Integration testing involves testing how different components of the system work together. In this system we need to test the integration between the various components:

* YOLO + OpenCV: Test how YOLO and OpenCV work together. Provide input images or videos, run the object detection with YOLO, and apply image processing techniques from OpenCV. Verify that the combined output is as expected.
* YOLO + CNN/R-CNN: If you have both YOLO and custom CNN/R-CNN models for vehicle detection, test how they integrate. Run YOLO for initial detection and then pass the detected vehicle regions to the CNN/R-CNN models for further refinement. Verify that the combined approach improves accuracy.
* Counting Algorithm + YOLO: Test the integration between the counting algorithm and YOLO. Run the object detection using YOLO and pass the detected vehicle regions to the counting algorithm. Verify that the algorithm can correctly count the vehicles based on the Euclidean distance method.

**4.2.3 System Testing:**

System testing involves testing the complete system as a whole. It focuses on verifying that the system meets the specified requirements and functions as intended. we can perform the following tests:

* End-to-end testing: Provide input images or videos, run the entire system, and verify that it can accurately detect and count the vehicles.
* Performance testing: Test the system's performance by providing a large number of input images or videos and measuring the processing time and accuracy. Ensure that the system performs within acceptable limits.
* Boundary testing: Test the system with challenging scenarios, such as crowded parking areas, occluded vehicles, or low-light conditions. Verify that the system can handle these situations effectively.
* Robustness testing: Introduce noise or disturbances in the input data, such as adding random objects or modifying the images/videos. Verify that the system can handle these variations without significant degradation in performance.

By conducting unit testing, integration testing, and system testing, we can ensure that our vehicle counting and detection system works correctly, integrates seamlessly, and meets the desired requirements.

**Chapter 5**

**ADVANTAGES AND LIMITATIONS OF THE DEVELOPED SYSTEM**

**5.1. Advantages of developed system:**

* Accurate vehicle detection: The combination of YOLO (You Only Look Once) and CNN (Convolutional Neural Networks) provides high accuracy in detecting vehicles in real-time. YOLO is known for its efficiency and speed in object detection, while CNNs are effective in learning and recognizing complex patterns.
* Real-time processing: YOLO and CNN models, combined with efficient implementation using OpenCV, allow for real-time processing of video streams or camera feeds. This enables the system to continuously detect and count vehicles without significant delays.
* Robustness to varying conditions: YOLO and CNN-based models can handle varying lighting conditions, weather conditions, and vehicle orientations. They are trained on diverse datasets, which helps in generalizing well to different scenarios and environments.
* Object localization: YOLO and R-CNN (Region-based Convolutional Neural Networks) architectures enable accurate object localization, which means the system can precisely identify the position and boundaries of vehicles within the image or video frame. This information can be useful for various purposes, such as parking space allocation or traffic management.
* Easy integration: OpenCV provides a comprehensive set of libraries and functions for image processing, video manipulation, and computer vision tasks. It offers a wide range of tools for pre-processing, post-processing, and visualization, making it easier to integrate the vehicle counting and detection system into existing applications or frameworks.

**5.2. Limitations of developed system:**

* High computational requirements: YOLO and CNN models are computationally intensive, especially when processing high-resolution video streams or large datasets. This can require powerful hardware resources, such as GPUs, to achieve real-time performance.
* Training and fine-tuning complexity: Building and training accurate object detection models like YOLO and R-CNN require a significant amount of labeled training data and expertise in deep learning. The process involves data annotation, model training, hyperparameter tuning, and potentially multiple iterations to achieve satisfactory results.
* Limited performance in complex scenarios: While YOLO and CNN models are effective in many scenarios, they may struggle with certain challenging conditions. For example, occlusions (vehicles partially hidden by other objects), crowded parking areas, or heavily cluttered scenes can pose difficulties for accurate detection and counting.
* Sensitivity to variations in camera perspectives: The Euclidean distance method for counting vehicles relies on calculating the distance between detected objects in consecutive frames. However, the accuracy of this method can be affected by changes in camera perspective, vehicle speed, and occlusions. These factors may introduce errors in the counting algorithm.
* Maintenance and system updates: As technology advances, the system may require regular updates to incorporate new improvements, handle emerging challenges, and adapt to evolving hardware or software environments. This could involve additional development efforts, costs, and system downtime during the update process.

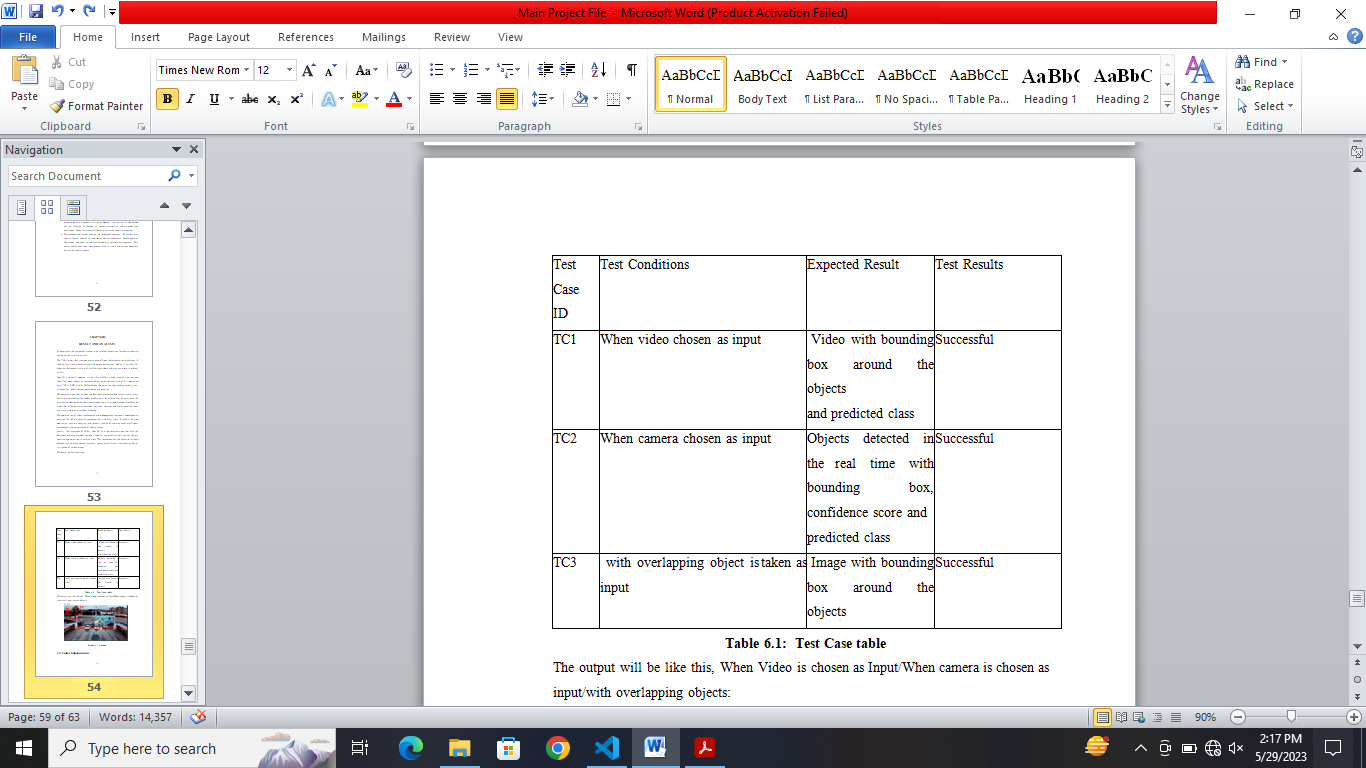
**CHAPTER 6**

### RESULT AND FUTURE ENHANCEMETS

### 6.1. Result:

Deep learning based pedestrian detection has been a research hotspot in recent years. This project starts on generic object detection pipelines which provide base architectures for other related tasks. With the help of this the three other common tasks, namely object detection, face detection and pedestrian detection, can be accomplished. Object detection with deep learning and OpenCV and Efficient, threaded video streams with OpenCV. The camera sensor noise and lightening condition can change the result as it can create problem in recognizing the object.

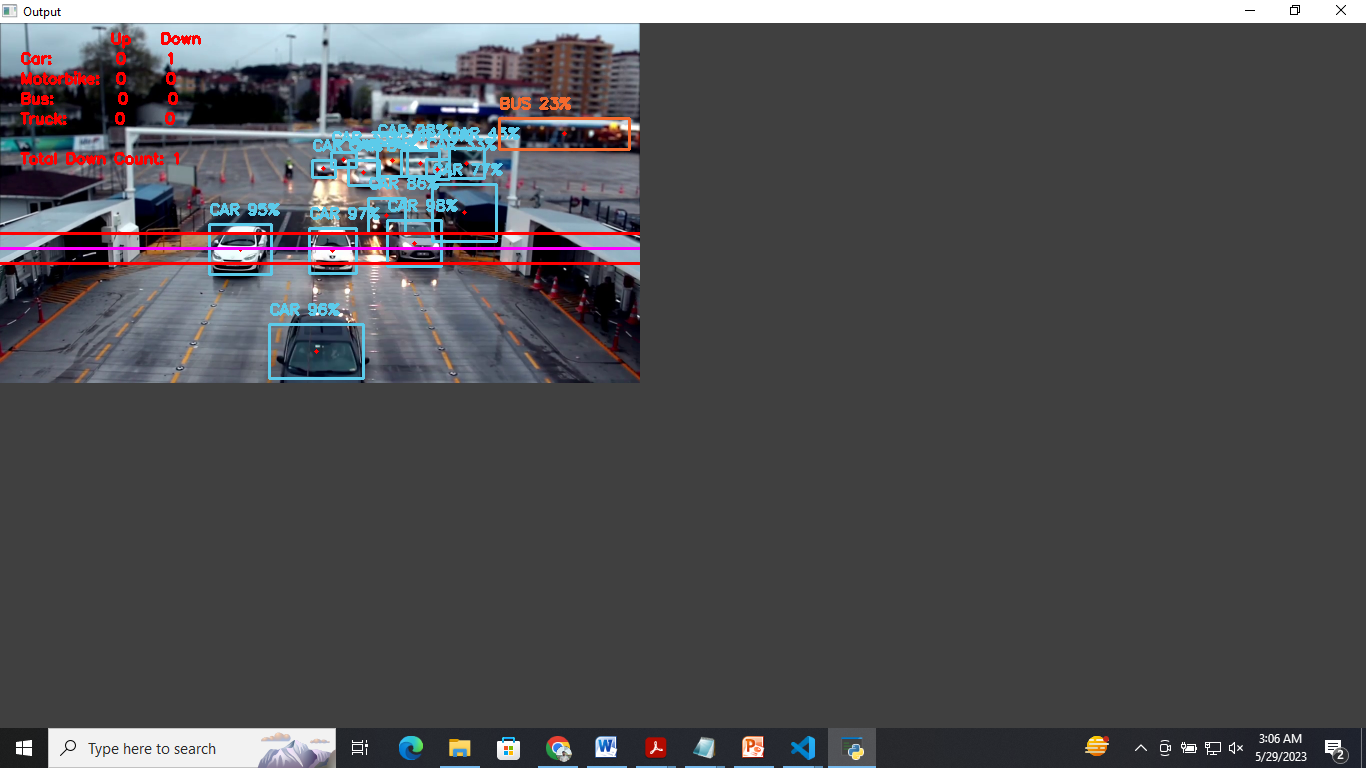
The Result we have obtained:



**Table 6.1: Test Case table**

The output will be like this, When,

* 1.Video is chosen as Input.
* When camera is chosen as input.
* with overlapping objects:

****

#### 

#### Figure 6.1 Output

#### 6.2. Future Enhancements:

There are several future enhancements that can be made:

* Multi-object tracking: Implementing a multi-object tracking algorithm can help in accurately tracking and counting vehicles over time. This can be achieved by associating detections in consecutive frames and maintaining unique IDs for each vehicle.
* Improved detection accuracy: Explore and experiment with newer and more accurate object detection models, such as YOLOv4, EfficientDet, or CenterNet, which can provide better detection performance and handle challenging scenarios.
* Incorporate depth information: Integrate depth sensors or depth estimation algorithms to obtain depth information of vehicles. This can help in distinguishing between different vehicles and detecting their precise positions, especially in crowded parking areas.
* Integration with surveillance systems: Connect the vehicle counting and detection system with surveillance cameras or CCTV networks. This integration can enable real-time monitoring, alerts for suspicious activities, and efficient management of parking areas.
* Automatic license plate recognition (ALPR): Implement ALPR techniques to read and recognize license plates of vehicles entering or leaving the parking area. This can be useful for security purposes, generating entry/exit logs, and identifying unauthorized vehicles.
* Cloud-based processing: Utilize cloud-based processing and storage to handle large-scale parking areas. This can offload the computational load from local devices, enable real-time analysis, and provide scalability for future expansion.
* Integration with parking management systems: Integrate the vehicle counting and detection system with parking management software to automate parking operations, such as availability tracking, automated payments, and efficient space allocation.

Also For Night time visual tracking, night vision mode should be available as an inbuilt feature in the CCTV camera [1,3]. To make the system fully automatic and also to overcome the above limitations, in future, multi view tracking can be implemented using multiple cameras. Multi view tracking has the obvious advantage over single view tracking because of wide coverage range with different viewing angles for the objects to be tracked. In this research work, the object Identification and Visual Tracking has been done through the use of ordinary camera[3]. The concept is well extendable in applications like Intelligent Robots, Automatic Guided Vehicles.

# Chapter 7

# CONCLUSION

In conclusion, YOLO is a real-time object detection algorithm that can accurately identify and locate vehicles in an image or video stream. It is a popular choice for its efficiency and accuracy in object detection tasks.

OpenCV is a powerful library for computer vision tasks, providing various functions and tools for image processing, feature extraction, and object detection. It can be used in conjunction with YOLO to process and analyze images or video frames.

CNN and R-CNN are deep learning architectures that can be trained to recognize and classify objects, including vehicles. By training these networks on a large dataset of vehicle images, they can learn to detect vehicles accurately.

The counting algorithm based on the Euclidean distance method calculates the distance between detected vehicle positions in consecutive frames. By comparing the distances, the algorithm can determine if a vehicle is entering or leaving the parking area, enabling accurate counting.

By combining these technologies, the system can effectively detect and count vehicles in a parking area in real-time. It can be implemented as a standalone application or integrated into existing surveillance systems to monitor parking occupancy and manage parking spaces efficiently.

Overall, the vehicle counting and detection system built with YOLO, OpenCV, CNN, R-CNN, and the Euclidean distance method provides a robust and accurate solution for parking area management, helping optimize parking operations and improve overall efficiency.

# 

# REFERENCE

We have used the following research papers as basis of our research and project work:

[1] Fruit Classification Comparison Based on CNN and YOLO: Riyanshu Raj et al 2021 IOP Conf.

Ser.: Mater. Sci. Eng. 1187 012031

[2] Detection of natural disaster victims using You Only Look Once (YOLO): M Sarosa et al 2021

IOP Conf. Ser.: Mater. Sci. Eng. 1098 032076

[3] Detection and Content Retrieval of Object in an Image using YOLO: B Vinoth Kumar et al 2019

IOP Conf. Ser.: Mater. Sci. Eng. 590 012062

[4] V. Gajjar, A. Gurnani and Y. Khandhediya, "Human Detection and Tracking for Video Surveillance: A Cognitive Science Approach," in 2017 IEEE International Conference on Computer Vision Workshops.

[5] Javed Miya & M. A. Ansari (2020) Medical images performance analysis and observations with SPIHT and wavelet techniques, Journal of Information and Optimization Sciences, 41:1, 273-282,

DOI: 10.1080/02522667.2020.1721616 .

[6] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, realtime object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 779-788).

[7] Ren S Q, He K M, Girshick R, et al.Faster R-CNN: towards real-time object detection with region proposal networks[J].IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(6): 1137-1149.

[8] Javed Miya & M. A. Ansari (2020) Wavelet Techniques for Medical Image Performance Analysis and Observations with EZW and Underwater Image Processing, Wireless Personal Communications, doi.org/10.1007/s11277-020-07238-w Springer Nature: <https://rdcu.be/b3eSe>.

[9] Huang J, Yuan Z, Zhou X. A Learning Framework for Target Detection and Human Face Recognition in Real Time. International journal of technology and human interaction, vol. 15, no. 3, pp. 63-76, 2019. 36

[10] Shahriar Shakir Sumit1, Junzo Watada1, Anurava Roy1 and DRA Rambli1 In object detection deep learning methods, YOLO shows supremum to Mask R-CNN: Shahriar Shakir Sumit et al 2020 J. Phys.: Conf. Ser. 1529 042086.

[11] Javed Miya & M. A. Ansari (2015) Observations and Performance Evaluation of various Techniques on Medical Images, International Journal of Applied Engineering Research, ISSN 0973- 4562 Vol. 10 No.94 pp. 47-51, DOI:10.37622/000000.

[12] NumPy is the fundamental package for scientific computing in Python. Its documentation can be found at <https://numpy.org/doc/stable/>

[13] PyTorch is the framework used for training the neural network, its documentation can be found at <https://pytorch.org/docs/stable/index.html>

[14] Microsoft coco (common objects in context) dataset can be found at

<https://cocodataset.org//#download>

[15] Pedestrian detection based on one-stage YOLO algorithm To cite this article: Xun Zuo et al 2021 J. Phys.: Conf. Ser. 1871 012131.