MAULANA AZAD
NATIONAL INSTITUTE OF TECHNOLOGY
BHOPAL INDIA, 462003

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Implementing Transfer Learning in YOLO for Intelligent Traffic Light Control System

Major Project Report
Semester VIII

Submitted by:

Mansi Gautam	191112008
Divya Sharma	191112039
Vismay Chourasia	191112071
Ashish Agarwala	191112073

Under the Guidance of
Dr Dhirendra Pratap Singh
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, hereby declare that the following report which is being presented in the Major Project Documentation Entitled "**Implementing Transfer Learning in YOLO for Intelligent Traffic Light Control System**" is authentic documentation of our own original work and the best of our knowledge. The following project and its report, in part or whole, have not been presented or submitted by us for any purpose in any other institute or organization. Any contribution made to the research by others, with whom we have worked at Maulana Azad National Institute of Technology, Bhopal or elsewhere, is explicitly acknowledged in the report.

Mansi Gautam	191112008
Divya Sharma	191112039
Vismay Chourasia	191112071
Ashish Agarwala	191112073

MAULANA AZAD
NATIONAL INSTITUTE OF TECHNOLOGY
BHOPAL INDIA, 462003

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project report carried out on "**Implementing Transfer Learning in YOLO for Intelligent Traffic Light Control System**" by the 4th year (8th Semester) students:

Mansi Gautam	191112008
Divya Sharma	191112039
Vismay Chourasia	191112071
Ashish Agarwala	191112073

Have successfully completed their project in partial fulfilment of their Degree in Bachelor of Technology in Computer Science and Engineering.

Dr Dhirendra Pratap Singh
(Major Project Mentor)

ACKNOWLEDGEMENT

With due respect, we express our deep sense of gratitude to our respected guide and coordinator Dr Dhirendra Pratap Singh, for his valuable help and guidance. We are thankful for the encouragement that he has given us in completing this project successfully.

It is imperative for us to mention the fact that the report of the major project could not have been accomplished without the periodic suggestions and advice of our project guide.

We are also grateful to our respected director Dr N. S. Raghuwanshi and HOD Dr Deepak Singh Tomar for permitting us to utilize all the necessary facilities of the college.

We are also thankful to all the other faculty, staff members and laboratory attendants of our department for their kind cooperation and help. Last but certainly not least, we would like to express our deep appreciation towards our family members and batch mates for providing much-needed support and encouragement.

ABSTRACT

Traffic congestion nowadays is a serious problem as the population is increasing and so is the number of vehicles in cities. Also, traffic jams increase fuel consumption and air pollution and add additional delay and stress to drivers. Also, due to its ever-increasing nature, road traffic density should be calculated in real-time for better traffic management and signal control. Traffic controllers are one of the key factors influencing traffic flow. CCTV cameras can be used to record traffic conditions and machine learning algorithms can add intelligence to the system. In our study, we benefit from advances in computer vision techniques for object detection thanks to Deep Learning, which is precisely based on YOLO. This is a new real-time object detection model. YOLO is an acronym for the words "You Only Look Once". This is an algorithm that detects (in real-time) and classifies different types of objects in the given image. YOLO uses single bounding box regression to predict object height, width, centre and class. Analysing and classifying the entry of vehicles at the arrival line and then counting the vehicles which are present in the region of interest (ROI). All this information is used to calculate the spatial occupancy of vehicles and it provides information by which traffic lights are switched based on vehicle density to ease the congestion. This allows people to move faster and reduce pollution.

TABLE OF CONTENTS

Contents

Implementing Transfer Learning in YOLO for Intelligent Traffic Light Control System 1
DECLARATION 2
CERTIFICATE 3
ACKNOWLEDGEMENT 4
ABSTRACT 5
TABLE OF CONTENTS 6
LIST OF FIGURES 8
LIST OF TABLES 9
1. INTRODUCTION 10
1.1 Topic Introduction 10
1.2 Problem Definition 11
1.3 Research Gaps 12
1.4 Research Objectives 13
1.5 Organization of Thesis 15
2. REVIEW OF LITERATURE 19
2.1 Modern Traffic Control Techniques: 19
2.2 Related Work: 21
2.3 Proposed Model: 22
2.4 YOLO: 23
2.5 Euclidean Distance: 25
2.6 Transfer Learning: 26
3. PROPOSED WORK DESCRIPTION 28
3.1 Flow Diagram: 28
3.2 Workflow: 29
4. REQUIREMENT DESCRIPTION 51
Hardware Requirement Specification: 51
Tools and Technologies: 52
Module and their brief descriptions: 54

5. RESULT ANALYSIS 56
6. CONCLUSIONS AND FUTURE DIRECTIONS 57
Conclusion: 57
Challenges Faced: 57
Future Prospects: 58
REFERENCES 59

LIST OF FIGURES

Figure 1: Clustering done in DPMM. 21
Figure 2: Vehicles are detected using YOLO and are classified based on their type. .. 23
Figure 3: Diagrammatic representation of bounding box formation 24
Figure 4: Calculation of Euclidean Distance 25
Figure 5: Flowchart representation of the process 28
Figure 6: Label Representation in annotation tool 31
Figure 7: Annotated Vehicles using bounding box. 31
Figure 8: Different Class Names of the annotated vehicles 32
Figure 9: The .txt file formed after annotation 32
Figure 10: Format of annotated bounding box 33
Figure 11: Folder for training data 34
Figure 12: Python Script for creating train.txt and test.txt files 35
Figure 13: Python command to create the file in colab 36
Figure 14: Created train.txt file. 36
Figure 15: Created test.txt file. 37
Figure 16: classes.names file containing names in form of a list 37
Figure 17: Labelled_data.data file 38
Figure 18: Python Script for creating classes.names file and labelled_data.data file .. 39
Figure 19: Python script for creation of the files 39
Figure 20: Changes made in the make file 40
Figure 21: Compiling the darknet repository using make command 40
Figure 22: Command for trining darknet model. 41
Figure 23: Region of interest initially 43
Figure 24: Car detected with 63% confidence. 44
Figure 25: Bus detected with 97% confidence. 44
Figure 26: Vehicles entering tracking region at arrival line. 45
Figure 27: Vehicles leaving the tracking region at departure line. 45
Figure 28: Area occupied by the vehicles in ROI. 46
Figure 29: ROI with vehicles 46
Figure 30: Frame considered for calculating ROI. 49

LIST OF TABLES

Table 1: Different parameter values considered for Detection model. 29

Table 2: Class Name and Class IDs 30

Table 3: Standard dimensions of vehicles and their calculated area............................. 47

Table 4: Database considered for training and testing. 49

1. INTRODUCTION

1.1 Topic Introduction

Due to increased vehicle traffic in urban areas, many urban road networks are experiencing reduced capacity and associated quality of service problems. In metropolitan areas, congestion is one of the major problems despite a well-planned road system and adequate infrastructure.

Traffic lights at intersections are regulated by local authorities. Peak hour statistics are used to determine timing, which is maintained even during off-peak hours. Every time in a day, this one standard green time will be used regardless of the number of vehicles on the road or the width of the road. Even though there are only a few vehicles, this approach will result in long green traffic lights, resulting in long waiting times at intersections. With several junction nodes, the optimization of the congested area remains a significant challenge.

Due to the involvement of various parameters, the traffic problem is extremely complicated. First, the amount of traffic varies depending on the time of day, with morning and afternoon typically being the busiest times, and second on the days of the week. In the current scenario, most traffic control systems in use are of the fixed cycle type, dictating a constant red/yellow/green phase for each signal cycle. In addition to these controllers, traffic police officers are also deployed to regulate the flow of traffic during periods of heavy traffic. These static systems cannot adapt to the dynamic situations that traffic police officers may face in real-time situations. Deploying traffic police officers at every intersection is impractical for reasons of manpower shortage and cost. There is an urgent need to build intelligent systems that can automate and control traffic. The increasing demand for road capacity also calls for traffic control solutions found in intelligent transportation systems.

Traditional traffic control techniques:

- In a manual traffic light control system, the operator may be located at the intersection itself. The operator typically follows a predetermined schedule or set of rules to determine when to change the signal phases, based on factors such as the time of day, the volume of traffic, and the presence of pedestrians or other road users. Manual traffic light control systems are typically used in smaller towns and cities, or in intersections with lower traffic volumes.

- A static traffic light control system is a type of traffic control system that uses fixed traffic signals, rather than variable or dynamic signals, to regulate the flow of traffic at an intersection. In a static traffic light control system, the traffic signals remain in a fixed position and do not change in response to changing traffic conditions or other factors.
- Using electronic sensors is another advanced method of placing some loop detectors or proximity sensors on the road. The electronic sensor gives data about the traffic present in the region of interest. Traffic signals are controlled based on sensor data.

These traditional methods face several drawbacks. A video camera system can be used to estimate traffic density and classify vehicles. Use it to control traffic light timers to optimize traffic flow and reduce traffic congestion at intersections. Our proposed system aims to design a traffic light controller based on the space occupancy of vehicles in the region of interest and use it to dynamically control traffic lights. Calculate real-time traffic density by using live images from intersection CCTV cameras to detect the number of vehicles at traffic lights and adjust green light times accordingly. Vehicles are categorized into cars, bikes and buses/trucks for accurate green time estimates. Use YOLO to detect the number of vehicles and adjust traffic light timers according to vehicle density in the appropriate direction. This optimizes green light times, organizes traffic much faster than static systems, reduces unneeded traffic, delays, and wait times, as well as fuel use and pollution.

1.2 Problem Definition

Given a network of roads and intersections in an urban area, the objective of the system is to optimize traffic flow and reduce congestion by using deep learning techniques to detect and track vehicles and pedestrians in real-time. The system must also analyse the spatial occupancy of the road network to predict traffic volume and flow.

The system should be able to adapt to changing traffic conditions and adjust traffic signals in real-time based on the predicted traffic flow. It should also provide real-time information to drivers and pedestrians, such as estimated travel time, congestion levels, and alternative routes.

The problem definition involves developing a deep learning model that can accurately detect and track vehicles and pedestrians in real-time and predict traffic volume and flow based on spatial occupancy. The model should be trained on a large dataset of annotated images and videos, using transfer learning techniques to improve its performance on new data. The system should also be designed to integrate with existing traffic management infrastructure and be scalable to handle larger urban areas.

The goal is to develop an intelligent traffic control system that can optimize traffic flow, reduce travel time, and improve safety on the roads, while also reducing emissions and fuel consumption, contributing to a more sustainable transportation system.

1.3 Research Gaps

Some potential research gaps for an intelligent traffic control system using YOLO and transfer learning based on spatial occupancy could include:

a. Robustness: While YOLO and transfer learning have shown promising results in object detection and recognition tasks, they may not always be robust to varying environmental conditions such as poor lighting, bad weather, and occlusions. Future research could focus on developing more robust models that can handle these challenging conditions.

b. Real-time performance: Real-time performance is essential for traffic control systems as they need to respond quickly to changing traffic conditions. However, complex deep learning models like YOLO can be computationally expensive, which can limit their real-time performance. Future research could focus on developing more efficient models or hardware optimizations to improve real-time performance.

c. Generalization: While transfer learning can help models to generalize well to new data, there is a risk that the model may become too specialized to the specific dataset used during training. Future research could focus on developing more generalized models that can perform well on a wider range of scenarios.

d. Integration with other data sources: An intelligent traffic control system using YOLO and transfer learning based on spatial occupancy may also benefit from integrating data from other sources, such as traffic cameras, sensors, and social media. Future research could focus on developing methods to integrate and fuse these data sources to improve the accuracy and robustness of the system.

e. Ethical and societal implications: An intelligent traffic control system has the potential to impact the lives of many people, and it is essential to consider the ethical and societal implications of such a system. Future research could focus on addressing these issues, such as privacy concerns, bias in the model, and fairness in the system's decision-making process.

In summary, there are several research gaps that need to be addressed to improve the effectiveness and efficiency of an intelligent traffic control system using YOLO and transfer learning based on spatial occupancy. These include robustness, real-time performance, generalization, integration of data from other sources, and ethical and societal implications. Addressing these research gaps can lead to the development of a more effective and efficient traffic control system that can optimize traffic flow, reduce travel time, and improve safety on the roads, while also reducing emissions and fuel consumption.

1.4 **Research Objectives**

The research objective of an intelligent traffic control system using YOLO (You Only Look Once) and transfer learning based on spatial occupancy is to develop a system that can optimize traffic flow and reduce congestion in urban areas. The system uses deep learning techniques to detect and track vehicles and pedestrians in real-time, analyse the spatial occupancy of the road network, and predict traffic volume and flow. The ultimate goal is to improve the efficiency and safety of the transportation system, reduce travel time, and minimize the environmental impact of vehicles.

The system is designed to be scalable, adaptable, and able to handle varying traffic conditions. It should be able to adjust traffic signals in real-time based on predicted traffic flow, adapt to changes in traffic patterns, and provide real-time information to drivers and pedestrians. The system should also be designed to integrate with existing traffic management infrastructure and be compatible with emerging technologies such as autonomous vehicles and smart cities.

The research objectives can be further broken down into several sub-objectives.

1. The first sub-objective is to develop an accurate and efficient object detection and tracking system using YOLO and transfer learning. This involves developing deep learning models that can accurately detect and track vehicles

and pedestrians in real-time, while also being efficient enough to run on low-power devices.

2. The second sub-objective is to analyse the spatial occupancy of the road network. This involves using data from traffic cameras, sensors, and other sources to analyse the traffic flow and congestion levels on the roads. The system should be able to predict traffic volume and flow based on this information and adjust traffic signals accordingly.
3. The third sub-objective is to optimize traffic flow and reduce congestion. This involves developing algorithms and models that can adapt to changing traffic conditions, adjust traffic signals in real-time, and provide real-time information to drivers and pedestrians. The system should be able to optimize traffic flow by reducing the number of stops and starts, minimizing the time vehicles spend idling, and maximizing the efficiency of the road network.
4. The fourth sub-objective is to improve the environmental impact of vehicles. This involves reducing emissions and fuel consumption by optimizing traffic flow and reducing congestion. The system should be able to reduce the number of vehicles on the road, minimize the amount of time they spend idling, and encourage the use of alternative modes of transportation such as public transit or bicycles.
5. The fifth sub-objective is to ensure the system is scalable and adaptable. This involves designing the system to handle varying traffic conditions and to integrate with existing traffic management infrastructure. The system should be compatible with emerging technologies such as autonomous vehicles and smart cities and should be able to adapt to changes in traffic patterns over time.

In summary, the research objective of an intelligent traffic control system using YOLO and transfer learning based on spatial occupancy is to develop a system that can optimize traffic flow and reduce congestion in urban areas. The system uses deep learning techniques to detect and track vehicles and pedestrians in real-time, analyse the spatial occupancy of the road network, and predict traffic volume and flow. The system should be scalable, adaptable, and able to handle varying traffic conditions, and should be designed to improve the efficiency and safety of the transportation system, reduce travel time, and minimize the environmental impact of vehicles.

1.5 Organization of Thesis

Chapter 1: Introduction

The introduction of a thesis on an intelligent traffic control system using YOLO and transfer learning based on spatial occupancy provides an overview of the research topic and sets the context for the study. It typically includes background information on the problem of traffic congestion in urban areas, the challenges of managing traffic flow, and the potential benefits of using artificial intelligence (AI) and machine learning (ML) techniques to optimize traffic control systems.

The introduction also outlines the objectives of the research, which typically include developing an intelligent traffic control system using YOLO and transfer learning based on spatial occupancy, analysing the performance of the system under different traffic conditions, and identifying potential areas for improvement or further research.

The organization of the thesis typically follows a structured approach, with each chapter addressing a specific aspect of the research topic. The first chapter typically provides an introduction to the research topic, outlines the research objectives, and discusses the significance of the study.

Chapter 2: Review of Literature

It provides an overview of the existing research in the field. The purpose of the literature review is to provide a comprehensive understanding of the current state-of-the-art in traffic control systems, AI and machine learning techniques, and related topics, and to identify any gaps or areas for further research.

The organization of the literature review typically follows a structured approach, with each subsection addressing a specific topic related to the research area. The subsections may include:

Introduction to traffic management: This subsection provides an overview of the challenges of managing traffic flow in urban areas, and discusses the various approaches used to control traffic, such as signal control, intersection management, and route guidance.

Review of traffic control systems: This subsection provides a comprehensive review of the existing traffic control systems, such as fixed-time signal control, adaptive signal control, and intelligent transportation systems (ITS). The section may also discuss the advantages and limitations of these systems.

Introduction to machine learning techniques: This subsection provides an introduction to machine learning techniques and their potential applications in traffic management systems.

Review of object detection and tracking algorithms: This subsection provides a review of the existing object detection and tracking algorithms, such as YOLO and their applications in traffic control systems.

Review of transfer learning techniques: This subsection provides a review of the existing transfer learning techniques, such as fine-tuning, domain adaptation, and multi-task learning, and their applications in traffic management systems.

Review of spatial occupancy models: This subsection provides a review of the existing spatial occupancy models, such as the Gaussian mixture model (GMM) and their applications in traffic flow prediction and control.

The literature review section of the thesis typically concludes with a summary of the main findings and a discussion of the gaps or limitations in the existing research. The literature review provides a critical evaluation of the current state-of-the-art in traffic control systems, Machine learning techniques, and related topics, and sets the foundation for the proposed research.

Chapter 3: Proposed Work Description

It provides a detailed description of the research that will be conducted to address the research gaps and limitations identified in the literature review. This section typically includes a detailed methodology, including data collection and pre-processing, model development, and model evaluation.

The proposed work section may be organized into subsections, such as:

Data collection and pre-processing: This subsection describes the data collection process, including the sources of data and the methods used to collect and pre-process the data. The subsection may also describe any data augmentation techniques used to increase the size and diversity of the dataset.

Model development: This subsection describes the process of developing the intelligent traffic control system using YOLO and transfer learning based on spatial occupancy. It includes a description of the deep learning architecture used, the transfer learning techniques employed, and any optimization techniques used to improve model performance.

Model evaluation: This subsection describes the evaluation metrics used to assess the performance of the proposed intelligent traffic control system. It may include a description of the dataset used for evaluation, the performance metrics used to evaluate the model, and any statistical analysis conducted to assess the significance of the results.

The proposed work section of the thesis provides a detailed description of the research methodology and the steps taken to address the research gaps and limitations identified in the literature review. It outlines the research approach and the technical details of the proposed system and provides a roadmap for future researchers in this area.

Chapter 4: Result Analysis and Description

It presents the findings of the research conducted in the proposed work section. This section typically includes a description of the experimental setup, the evaluation metrics used to assess the performance of the proposed system, and the results obtained.

The section may be organized into subsections, such as:

Hardware requirement specification: This subsection describes the hardware and software used in the experiments, including the dataset used for testing and any preprocessing techniques applied to the data.

Evaluation metrics: This subsection describes the evaluation metrics used to assess the performance of the proposed system. These may include metrics such as accuracy, precision, recall.

Results: This subsection presents the results obtained from the experiments, including any comparisons with existing state-of-the-art methods. It may include tables and graphs to visually represent the results and highlight any trends or patterns observed.

The result analysis and description section is critical to the thesis as it provides a detailed description and evaluation of the proposed system's performance. It highlights the contributions of the research, the limitations of the study, and potential future work. The section provides valuable insights for researchers and practitioners interested in developing intelligent traffic control systems using YOLO and transfer learning based on spatial occupancy.

Chapter 5: Summary and Future Prospects

This provides a comprehensive overview of the research conducted in the thesis. This section typically includes a summary of the research objectives, methodology, and findings, as well as potential future research directions.

The section may be organized into subsections, such as:

Summary of research: This subsection provides a brief summary of the main research objectives, methodology, and findings. It highlights the contributions of the research and provides an overview of the proposed system's performance.

Contributions and Challenges faced: This subsection discuss the contributions of the research and its limitations. It may highlight the strengths and weaknesses of the proposed system and identify areas for future research.

Future research directions: This subsection outlines potential future research directions based on the limitations and gaps identified in the research. It may suggest areas for further exploration, such as the integration of other types of sensors, the use of alternative deep learning architectures, or the development of more efficient optimization algorithms.

Conclusion: This subsection provides a concluding remark summarizing the main findings of the thesis and the potential impact of the proposed system.

The summary and future prospects section of the thesis provides a broad overview of the research conducted and its potential implications for the field of intelligent traffic control systems. It emphasizes the need for continued research in this area and provides a roadmap for future researchers to build upon the work presented in the thesis. The section underscores the significance of the research and its potential impact on the field of intelligent transportation systems.

2. REVIEW OF LITERATURE

2.1 Modern Traffic Control Techniques:

2.1.1 **Ala'a Abu Zaid et al. [1]** proposed an Intelligent traffic light system using ANN and fuzzy controller. The system uses images captured by cameras installed at traffic sites. Next, perform segmentation using a sliding window technique to count cars regardless of size. The segmented image is fed into an ANN, and the output is used in a fuzzy controller to time the red and green lights using the output's sharpness.

2.1.2 **Renjith Soman et al. [2]** proposed the use of Support Vector Machine algorithms along with image processing techniques. An algorithm is used after a small-frame image is taken. OpenCV is used for image processing, and before SVM is used, the image is converted to grayscale.

2.1.3 **Nazmus S. Nafi et al. [3]** proposed the use of VANET's V21 communication mechanism to detect the arrival of individual vehicles from different lanes and adaptively change traffic signal phases at intersections in relation to vehicle density. An optimized adaptive signalling system, vehicle mobility model, and communication network model work together within the same control platform in an OPNET-based co-simulation model. However, this technique is very costly and impractical.

2.1.4 **Cao et al. [4], Hausknecht et al. [5], Zhao et al. [6]** discussed signal management at junctions. However, these methods handle traffic signal management using the information available from multiple sources via communication mechanisms. To the best of our knowledge, no work has been done that applies queuing theory to methods based on computer vision to predict signal duration.

2.1.5 **Mihir M. Gandhi et al. [7]** proposed a YOLO-based system to build real-time detection and vehicle tracking before exiting an intersection. When the traffic light turns yellow, the controller receives the number of vehicles per lane and their wait times from his YOLO model. Based on the maximum and average waiting time for each lane and the length of congestion, the next phase duration is calculated to minimize the waiting time.

2.1.6 **Thakur, Dr. Narina et al. [10]** proposed the detection capabilities of different algorithms, namely Faster-RCNN, SSD, and YOLO, on the MOT20 dataset. Additionally, a custom dataset captured by an Unmanned Aerial Vehicle (UAV) within our organization's premises was used to test the algorithms. The experiments were conducted to evaluate the performance of the models and to provide an analysis of their effectiveness.

2.1.7 **Lihe Hu et al. [11]** A transfer learning-based approach for identifying dense objects on the road has been proposed in this study. The proposed method utilizes the pre-trained YOLOv3 model, obtained from the Darknet-53 network structure, and introduces transfer training as the output layer for a special dataset consisting of 2000 images containing vehicles. To optimize the weights of the transfer training model, a random function is adapted for initialization. This transfer training model is designed separately from the pre-trained YOLOv3 model. Additionally, the fully connected layer is replaced by an object detection classifier, which further improves the detection accuracy.

2.1.8 **Xiaoyuan Liang et al. [12]** proposed a deep reinforcement learning model has been proposed to address high dimensional Markov decision processes in traffic management. The model is designed to intelligently analyze the current traffic situation and make decisions based on the volume of vehicles, with the goal of minimizing wait times at traffic intersections. To achieve this, the model is trained using a reward function that incentives minimal waiting times. Over time, the model becomes more advanced and capable of optimal traffic management.

2.2 Related Work:

Santhosh Kelathodi Kumaran et al. [8] proposed the Queuing theory guided intelligent traffic scheduling using the Dirichlet process mixture model (DPMM), It has been noted that the length of the green signal depends on the number of vehicles queuing up for a particular direction of vehicle movement. It has been proposed that spatial occupancy plays a significant role in determining the time to allocate to free up the space of a lane. In the spatial Occupancy model, the count of the number of vehicles in the region of interest (ROI) is calculated, and dynamically traffic signal is controlled in the junction using the count of vehicles in ROI. They made use of temporal clusters in determining the space occupancy of each vehicle and calculated the time to empty these clusters from the lane. For e.g. - considering a car consists of one cluster, a bus can be taken to consist of two clusters as a car's size is almost half that of a bus. But we believe that each vehicle has a different size even if the vehicles with the same type can have different sizes and assigning the clusters assuming the fixed size of the vehicle can give wrong results.
They used the gaussian regression model to assign the clusters which gives good results for clustering. And then calculated the time to free up the space by each cluster considering all clusters have the same speed and take the same time to free up the space.

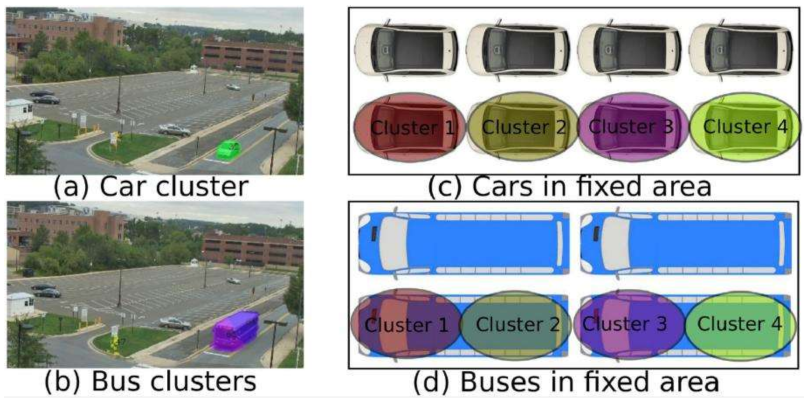


Figure 1: Clustering done in DPMM.

(Image reference: Queuing Theory Guided…)

Limitations of the DPMM model:

1. The DPMM model considers vehicle size to be fixed but size can be varied depending on different types of models of vehicles of a single category and the area they cover. for example, Buses take up more space at intersections than cars and take longer to cross intersections than cars.

2. As they have mentioned earlier, they have used a gaussian regression model to calculate the probability of a pixel to be assigned in which cluster it is best fitted. However, there also exist other models which can give better results than this model.
3. The camera should be aligned in a proper way so that it should cover all the vehicles. And if we consider that a small vehicle can hide from the camera if it is behind a large vehicle having more height. So, the proper alignment of the camera is needed along with a counter of vehicles by categories.
4. The DPMM model considers the area the vehicle covers on the ground, and the actual speed of a vehicle also depends on the volume and weight of a vehicle in ideal condition.

2.3 **Proposed Model:**

Our model maintains the number of vehicles in the region of interest and their classification. The purpose of this study is to adjust the traffic lights at intersections in accordance with the volume of traffic. Our proposed system aims to design a computer vision-based traffic light controller that can adapt to current traffic conditions. Calculate real-time traffic density by using live images from intersection CCTV cameras to detect the number of vehicles at traffic lights and adjust green light times accordingly. Vehicles are categorized as cars, bikes, buses/trucks, or rickshaws to provide accurate green time estimates. YOLO is a fast, accurate object detector, in TensorFlow. It contains a publicly available dataset containing several million natural images. Traditional object detectors first generate proposals. We have used YOLO to detect the number of vehicles and set traffic light timers according to vehicle density in the appropriate direction. This optimizes green light times, organizes traffic much faster than static systems, reduces idling, traffic, and waiting times, as well as the use of fuel and pollution. YOLO is an algorithm for real-time object detection that makes use of neural networks. This algorithm is well-liked for its accuracy and speed. YOLO is an acronym for the words "You Only Look Once". This algorithm identifies and recognises various objects in an image in real time. The class probabilities of the recognised images are provided by YOLO's object recognition as a regression problem. A bounding box is a contour that emphasizes an object in an image. YOLO uses single bounding box regression to predict object height, width, centre, and class.

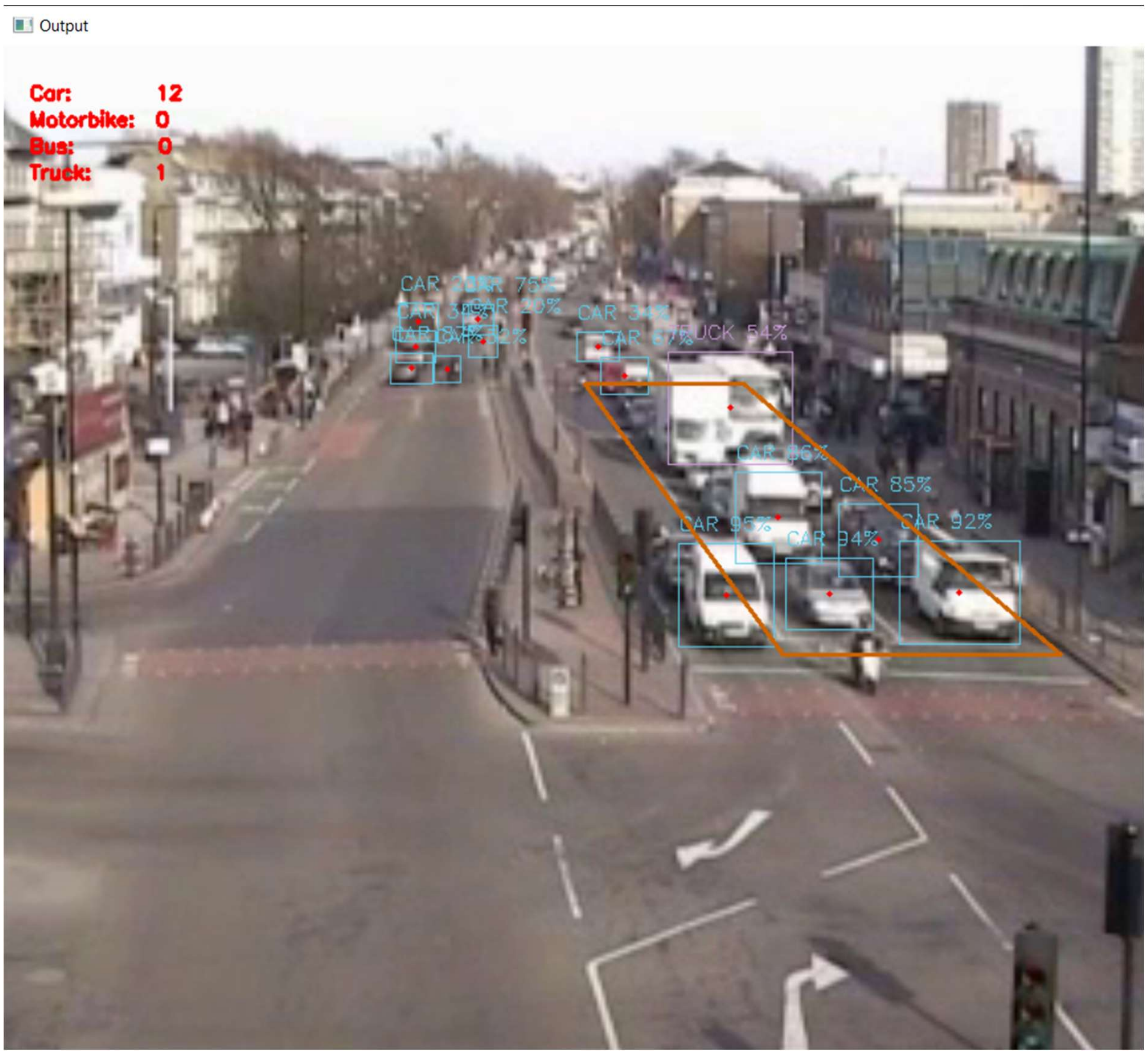


Figure 2: Vehicles are detected using YOLO and are classified based on their type.

2.4 YOLO:

This algorithm is called You Only Look Once (YOLO) because it uses bounding boxes to identify objects and their locations in a single look at the image.

Features of YOLO:

1. Speed: YOLO is very fast because it doesn't handle complicated pipelines. It can process images at 45 frames per second (FPS), making it ideal for real-time processing.
2. High detection accuracy: YOLO is way more accurate than other object-detection algorithms with very few background errors.
3. Open source: Making YOLO open source led the community to constantly improve the model. This is one of the reasons why YOLO has made so many improvements in such a limited time.

YOLO Object detection working:
In this object detection process, an image/frame is divided into S × S grid cells, and each grid cell predicts a B bounding box with its position and dimensions, object probabilities in the underlying grid, and conditional class probabilities. The basic concept behind object detection by any grid cell is that the centre of the object must be within that grid cell.
The next step is to determine the bounding boxes which correspond to rectangles highlighting all the objects in the image. YOLO determines the attributes of these bounding boxes using a single regression module in the following format, where Y is the final vector representation for each bounding box.

Y = [pc, bx, by, bh, bw, (list of classes)]

- pc corresponds to the probability score of the grid containing an object.
- bx, by are the x and y coordinates of the centre of the bounding box.
- bh, bw correspond to the height and the width of the bounding box.

A single object in an image can have multiple grid box candidates for prediction, even if they are not all related. The purpose of the IOU (a value between 0 and 1) is to discard such grid boxes to keep only the relevant ones.

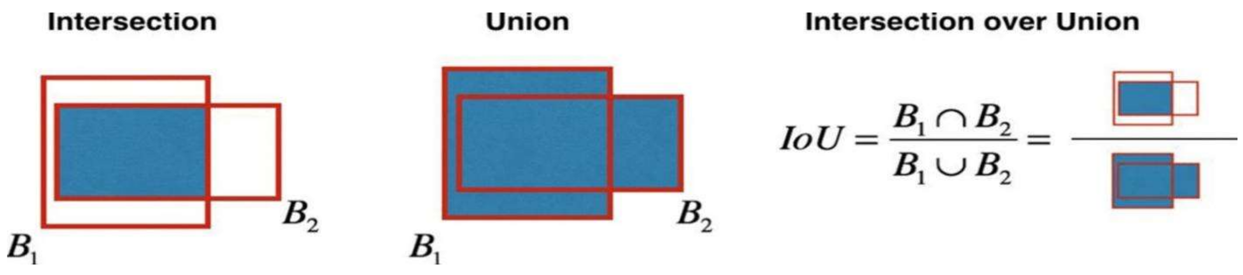


Figure 3: Diagrammatic representation of bounding box formation

It ignores the prediction of the grid cells having an IOU ≤ threshold and considers those with an IOU > threshold.

2.5 **Euclidean Distance:**

The "normal" straight-line distance between any two points in Euclidean space is known as the Euclidean distance or Euclidean metric.
Using the Euclidean distance of each object and the distance between their centre points, calculate the separation between the current frame and the reference frame.

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}.$$

Euclidean Distance Equation

Figure 4: Calculation of Euclidean Distance

If the distance calculated is less than the threshold, they are the same object. Then the old id is replaced with the new one signifying that the new object is the same old object and if distance is greater than the threshold than new id remains as it is, and this also means that the object encountered is a new object and new entry must create for it for tracking. For our model proposed tracking gave better results on threshold as 15, thresholds smaller than this resulted in multiple counting and if greater thresholds were taken then the less counting were taken so best result came out with this proposed threshold only.

So basically in our model there is a list which stores the id and details of the vehicles detected now in the next frame when a vehicle detected with its parameter now its centre Euclidean distance is checked with others and if all greater than threshold it means new object are its id as well parameter entered into the list other the closest point's id and parameter is replaced with updated id and parameters as both as same object.

2.6 Transfer Learning:

Transfer learning is a machine learning technique that allows a model to reuse knowledge learned from one task to solve a related task. In transfer learning, the model is first trained on a source task that has a large amount of labelled data, and then fine-tuned on a target task that has a relatively small amount of labelled data.
The key idea behind transfer learning is that the knowledge gained from solving one problem can be useful for solving a different but related problem. By transferring knowledge, the model can learn to generalize better and make more accurate predictions on the target task, even when the amount of labelled data is limited.
Transfer learning has been successfully applied in various fields such as computer vision, natural language processing, and speech recognition. For example, a model trained on a large dataset of images can be fine-tuned on a smaller dataset of images to perform a specific task such as object detection or classification. Similarly, a language model trained on a large corpus of text can be fine-tuned on a smaller dataset to perform a specific NLP task such as sentiment analysis or named entity recognition.
Overall, transfer learning has become an essential tool in machine learning, enabling the development of more accurate and robust models with less labelled data and reduced training time.

Use of transfer learning in image detection:
Transfer learning is widely used in image detection tasks such as object detection, image classification, and semantic segmentation. The basic idea behind transfer learning in image detection is to leverage a pre-trained model on a large dataset, such as ImageNet, and then fine-tune the model on a smaller dataset specific to the target task.
In image classification, transfer learning can be used to classify images into different categories such as dogs, cats, birds, and so on. A pre-trained model, such as VGG16 or ResNet, can be used as a feature extractor to extract relevant features from the input image. These features are then passed through a classification layer to obtain the final predictions.
In object detection, transfer learning can be used to detect and locate objects in an image. A pre-trained model, such as Faster R-CNN or YOLO, can be used as a feature extractor to extract features from the input image. These features are then used to detect objects and their locations within the image.

In semantic segmentation, transfer learning can be used to label each pixel of an image with its corresponding object class. A pre-trained model, such as U-Net or SegNet, can be used as a feature extractor to extract features from the input image. These features are then passed through a decoder to obtain a segmentation map.
Overall, transfer learning has proven to be a powerful technique in image detection, enabling the development of more accurate and efficient models with less labelled data and reduced training time.

3. PROPOSED WORK DESCRIPTION

3.1 Flow Diagram:

Given below is the complete flow of the process we have followed for computing the accuracy.

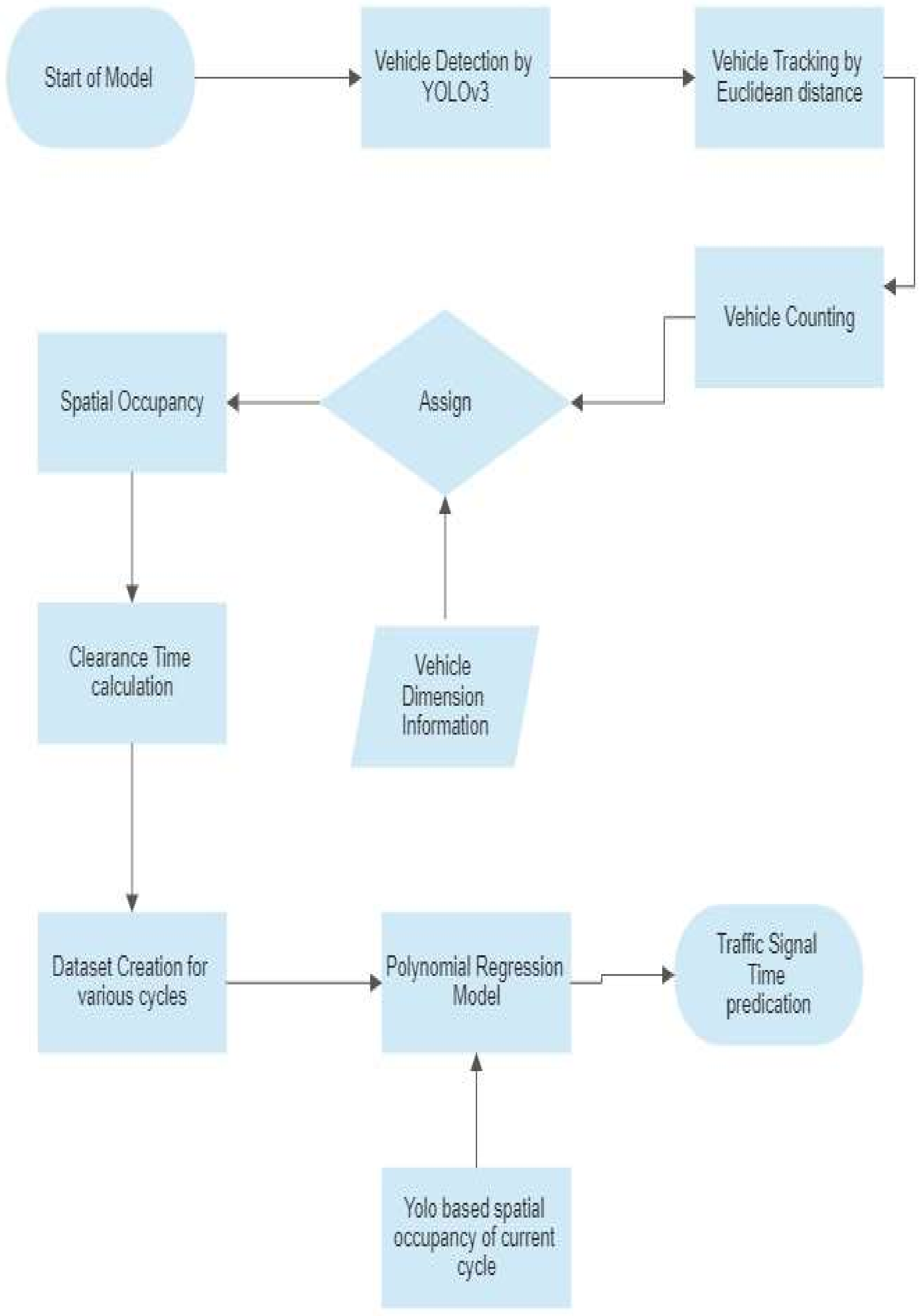

Start of Model
Vehicle Detection by YOLOv3
Vehicle Tracking by Euclidean distance
Vehicle Counting
Assign
Spatial Occupancy
Vehicle Dimension Information
Clearance Time calculation
Dataset Creation for various cycles
Polynomial Regression Model
Traffic Signal Time predication
Yolo based spatial occupancy of current cycle

Figure 5: Flowchart representation of the process

3.2 **Workflow:**

Training and Testing Model:

The train/test method is a way to gauge how accurate your model is. Because you divide the data set into two sets—a training set and a testing set—this method is known as train/test. 80% for training, and 20% for testing. Using the training set, you train the model. Utilizing the testing set, you test the model. To train a model is to build a model. To test a model is to determine its accuracy. Model taken into account for training and testing.

YOLO is a clever convolutional neural network (CNN) for performing object detection in real time. The algorithm uses one neural network to process the entire image before segmenting it into regions and forecasting bounding boxes and probabilities for each one. The predicted probabilities are used to weight these bounding boxes. Because it can run in real time and achieve high accuracy, YOLO is well-liked. The algorithm “only looks once” at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes. With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. The backbone CNN used in YOLO can be further simplified to improve processing time. We have considered YOLOv3 for testing and training our model as it has the advantages of higher detection speed and more accuracy, also it meets the real-time requirements for vehicle detection.

The parameters considered for training and testing our model are:

Table 1: Different parameter values considered for Detection model.

Parameter	Value
confThreshold	0.2
nmsThreshold	0.2
Threshold for tracking in Euclidean	15

A. Implementation of transfer learning:

1) Filtering the sample images -

The screenshots at different time stamps are taken from the video. Now each screenshot has been cropped to the size of 1080x1080 as YOLO constraints that height of the image must be equal to width of the image. And the resulting images are taken for the annotation process.

2) Deciding the labels -

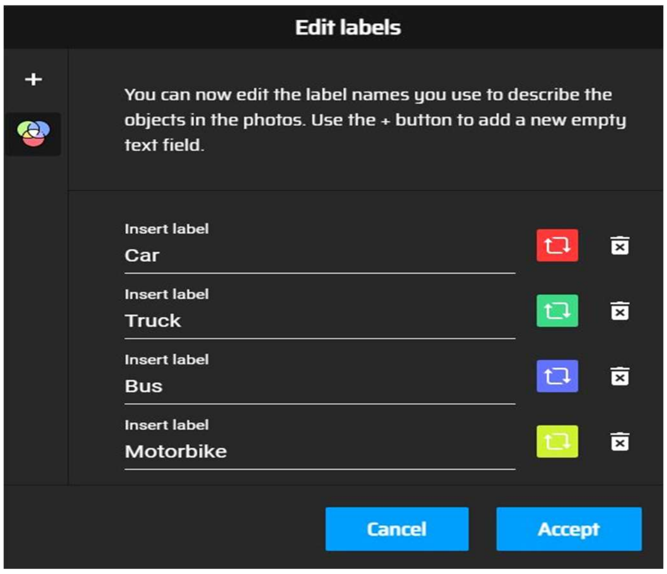
We have used the online platform makesense.ai to annotate the images for object detection. To do so first we need to decide the labels and for our model we have used four labels namely Car, Truck, Bus and Motorbike.

These labels correspond to the different classes in which we intend to classify our detected objects. In case of our model the following Class Names and Class Numbers/IDs are considered:

Table 2: Class Name and Class IDs

Class Name	Class Number/ID
Car	0
Truck	1
Bus	2
Motorbike	3

Below is the label representation in the annotation tool makesense.ai :



Edit labels

You can now edit the label names you use to describe the objects in the photos. Use the + button to add a new empty text field.

Insert label
Car

Insert label
Truck

Insert label
Bus

Insert label
Motorbike

Cancel Accept

Figure 6: Label Representation in annotation tool

3) Annotate the images -

With the help of the tool first draw a bounding box covering the entire vehicle and similarly do this for all the vehicles present in the region of interest. Now select the label from the dropdown for e.g. Car, Bus, Truck or Motorbike which is to be assigned to each individual bounding box.

Below figure represents the annotated objects:

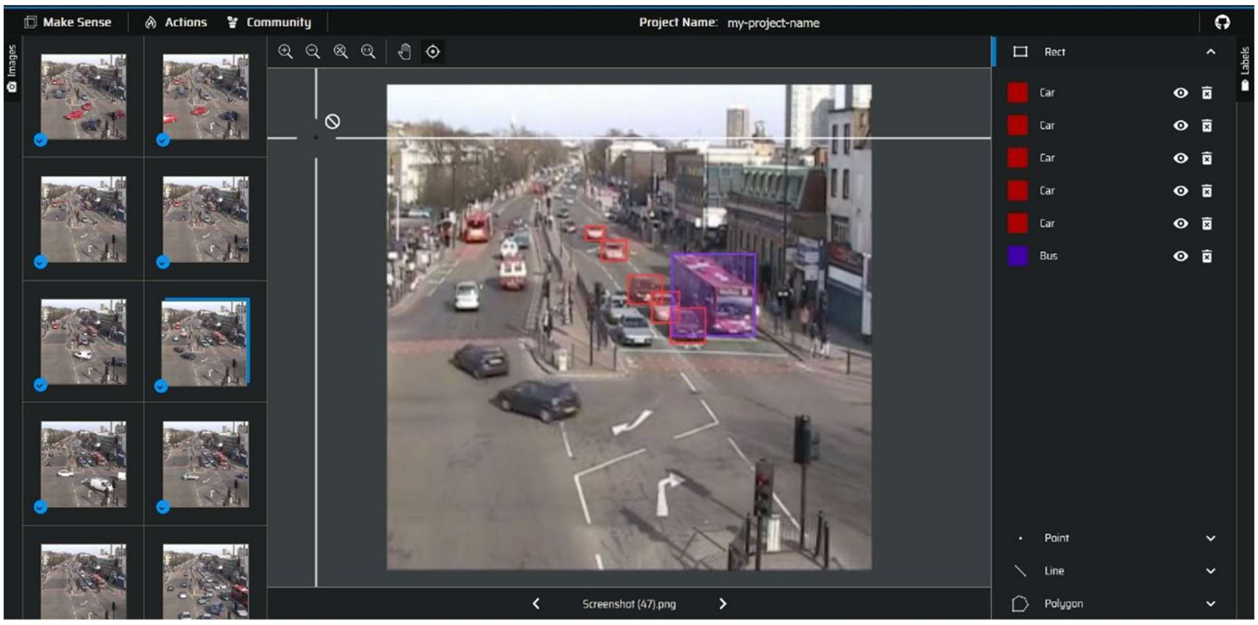


Figure 7: Annotated Vehicles using bounding box.

Below figure shows the dropdown where we can select different Class Names for the objects:

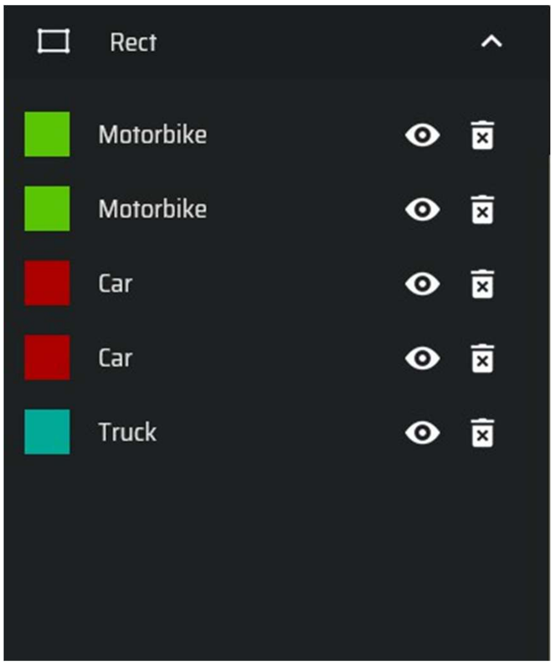
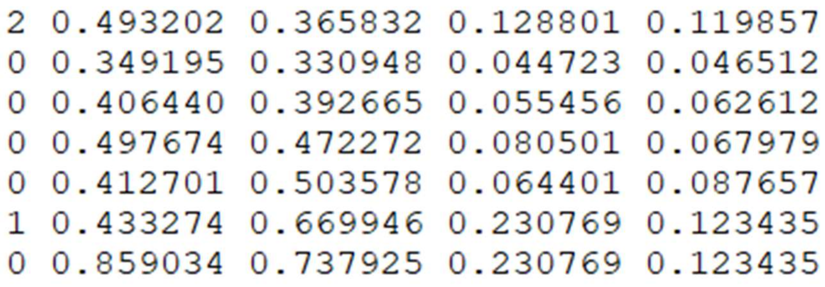


Figure 8: Different Class Names of the annotated vehicles

4) Converting the annotations to YOLO format -

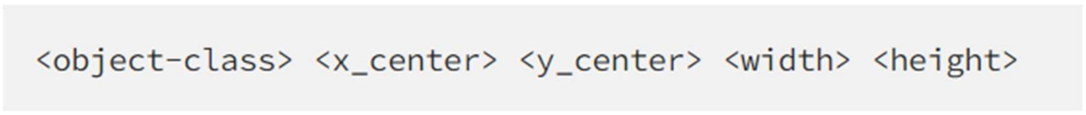
Once all the images have been annotated we can download them in the YOLO format which comprises of a .txt file for each .png file i.e. Screenshot(47).png will have a corresponding Screenshot(47).txt file. The format of the .txt file looks like this :



```
2 0.493202 0.365832 0.128801 0.119857
0 0.349195 0.330948 0.044723 0.046512
0 0.406440 0.392665 0.055456 0.062612
0 0.497674 0.472272 0.080501 0.067979
0 0.412701 0.503578 0.064401 0.087657
1 0.433274 0.669946 0.230769 0.123435
0 0.859034 0.737925 0.230769 0.123435
```

Figure 9: The .txt file formed after annotation.

Here each bounding box annotated has an entry which has the following format:



```
<object-class> <x_center> <y_center> <width> <height>
```

Figure 10: Format of annotated bounding box

where :

- **object-class** refers to the Class Number/ID assigned to the bounding box during the annotation process.
- **x_centre** refers to the x coordinate of the centre of the bounding box.
- **y_centre** refers to the y coordinate of the centre of the bounding box.
- **width** refers to the relative width of the bounding box with respect to the entire image's width.
- **height** refers to the relative height of the bounding box with respect to the entire image's height.

Here object-class is an integer value whereas x_centre, y_centre, width and height are floating point values.

5) Creating the database -

Now with the help of annotated images we create a database including the image file and its corresponding text files. In our database we have taken a total of 42 unique images. We have created a folder named training_data where the complete database is stored.

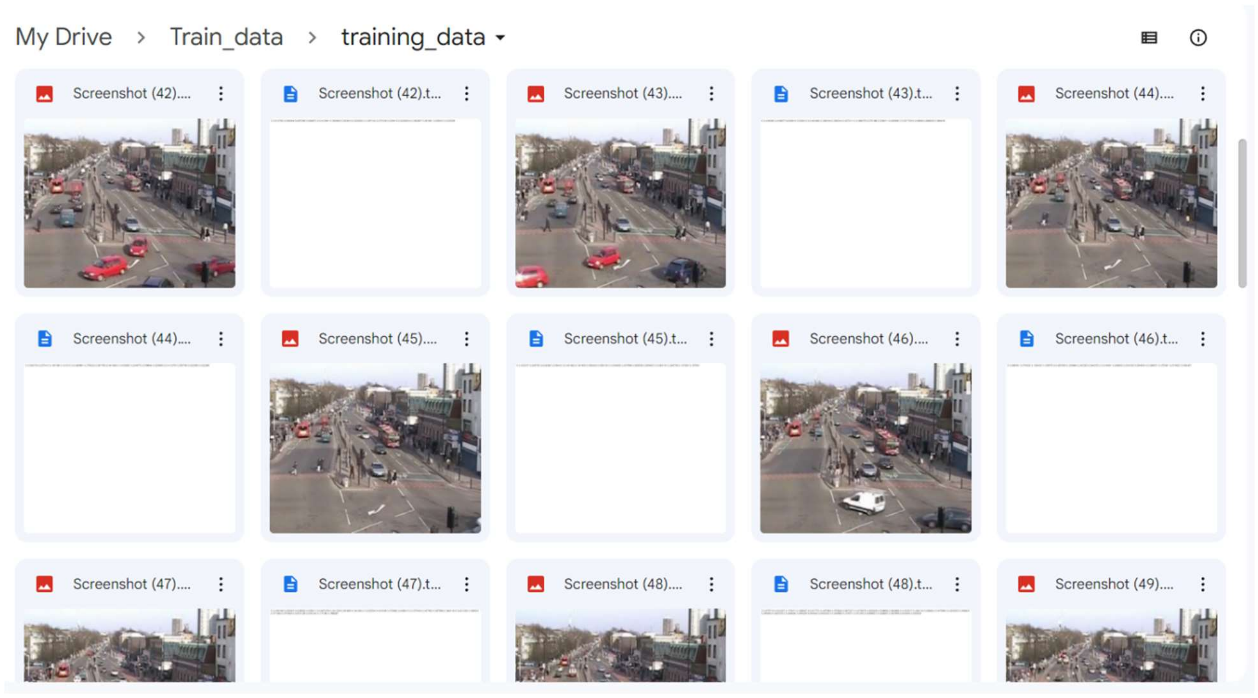


Figure 11: Folder for training data

6) Creating the test and train files -

After the formation of the database we now need to split it into two parts i.e. the training part and the testing part. In order to achieve this we make use of a python script which creates two text files:

train.txt - It includes the paths of the images that will be considered for training the model.

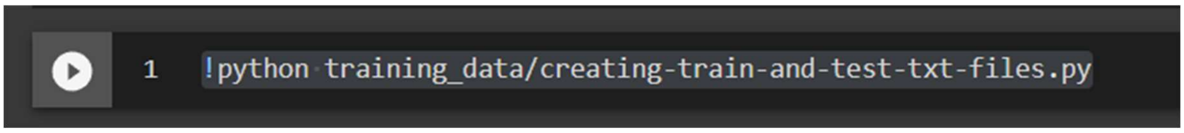
test.txt - It includes the paths of the images that will be considered for testing the model.

Given below is the python script used: \

```
"""
File: creating-train-and-test-txt-files.py
"""

# Importing needed library
import os

"""
Setting up full path to directory with labelled images
"""
full_path_to_images = 'training_data'

"""
Getting list of full paths to labelled images
"""

# Changing the current directory
# to one with images
os.chdir(full_path_to_images)

# Defining list to write paths in
p = []

# Using os.walk for going through all directories
# and files in them from the current directory
# Fullstop in os.walk('.') means the current directory
for current_dir, dirs, files in os.walk('.'):
    # Going through all files
    for f in files:
        # Checking if filename ends with '.jpeg'
        if f.endswith('.png'):
                        path_to_save_into_txt_files =
full_path_to_images + '/' + f

            # Appending the line into the list
            # We use here '\n' to move to the next line
            # when writing lines into txt files
            p.append(path_to_save_into_txt_files + '\n')


# Slicing first 15% of elements from the list
# to write into the test.txt file
p_test = p[:int(len(p) * 0.15)]

# Deleting from initial list first 15% of elements
p = p[int(len(p) * 0.15):]

"""
Creating train.txt and test.txt files
"""

# Creating file train.txt and writing 85% of lines in it
with open('train.txt', 'w') as train_txt:
    # Going through all elements of the list
    for e in p:
        # Writing current path at the end of the file
        train_txt.write(e)



# Creating file test.txt and writing 15% of lines in it
with open('test.txt', 'w') as test_txt:
    # Going through all elements of the list
    for e in p_test:
        # Writing current path at the end of the file
        test txt.write(e)
```

Figure 12: Python Script for creating train.txt and test.txt files

This script named as creating-train-and-test-txt-files.py can be executed using the following command:

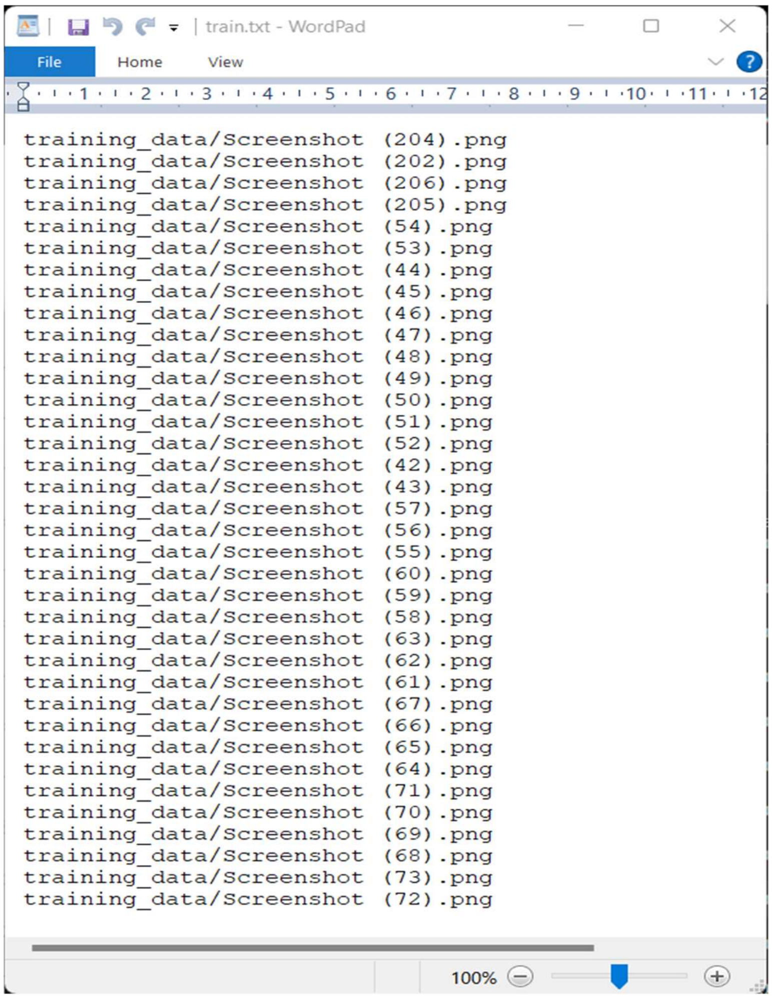


```
!python training_data/creating-train-and-test-txt-files.py
```

Figure 13: Python command to create the file in colab

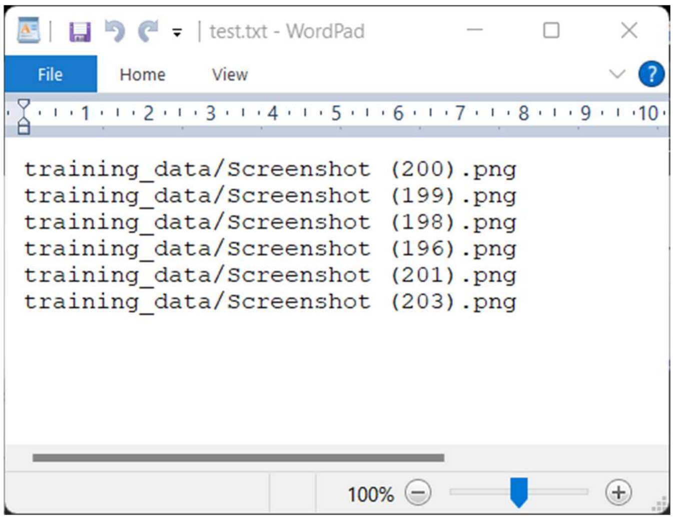
After execution of this script the following files are created

train.txt test.txt



```
training_data/Screenshot (204).png
training_data/Screenshot (202).png
training_data/Screenshot (206).png
training_data/Screenshot (205).png
training_data/Screenshot (54).png
training_data/Screenshot (53).png
training_data/Screenshot (44).png
training_data/Screenshot (45).png
training_data/Screenshot (46).png
training_data/Screenshot (47).png
training_data/Screenshot (48).png
training_data/Screenshot (49).png
training_data/Screenshot (50).png
training_data/Screenshot (51).png
training_data/Screenshot (52).png
training_data/Screenshot (42).png
training_data/Screenshot (43).png
training_data/Screenshot (57).png
training_data/Screenshot (56).png
training_data/Screenshot (55).png
training_data/Screenshot (60).png
training_data/Screenshot (59).png
training_data/Screenshot (58).png
training_data/Screenshot (63).png
training_data/Screenshot (62).png
training_data/Screenshot (61).png
training_data/Screenshot (67).png
training_data/Screenshot (66).png
training_data/Screenshot (65).png
training_data/Screenshot (64).png
training_data/Screenshot (71).png
training_data/Screenshot (70).png
training_data/Screenshot (69).png
training_data/Screenshot (68).png
training_data/Screenshot (73).png
training_data/Screenshot (72).png
```

Figure 14: Created train.txt file.

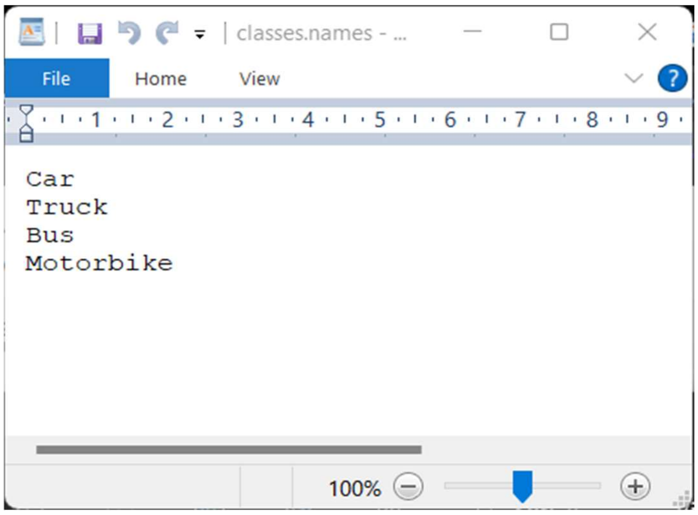


```
training_data/Screenshot (200).png
training_data/Screenshot (199).png
training_data/Screenshot (198).png
training_data/Screenshot (196).png
training_data/Screenshot (201).png
training_data/Screenshot (203).png
```

Figure 15: Created test.txt file.

7) Creating the data and names files -
   In order to configure the Yolo model we need to create two more files namely

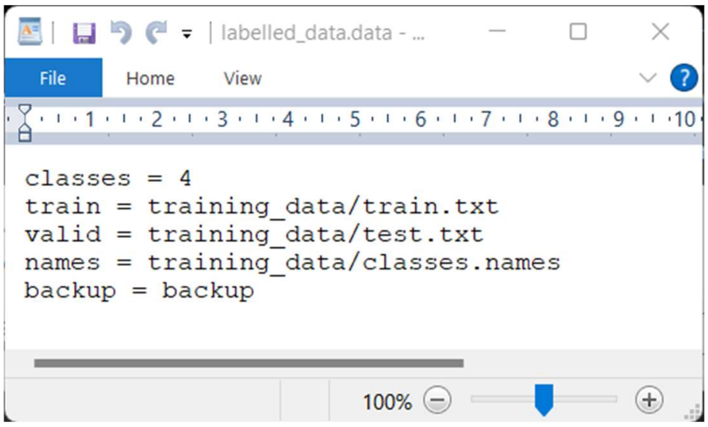
   1. classes.names file which contains the class names in the form of a list, one above the other.



```
Car
Truck
Bus
Motorbike
```

Figure 16: classes.names file containing names in form of a list

2. labelled_data.data file which contains the following information about

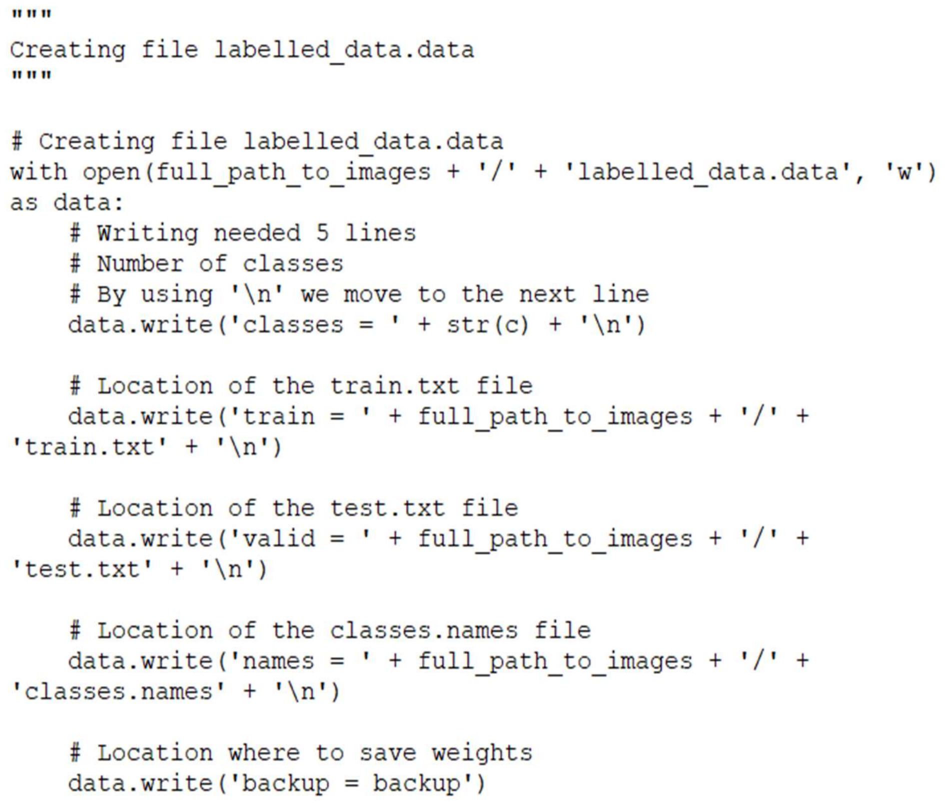


```
classes = 4
train = training_data/train.txt
valid = training_data/test.txt
names = training_data/classes.names
backup = backup
```

Figure 17: Labelled_data.data file

It specifies the number of classes for which we are custom training the model, the paths of train.txt and test.txt files and the path of the folder where the weights shall be stored after training the model.
Given below is the python script used:

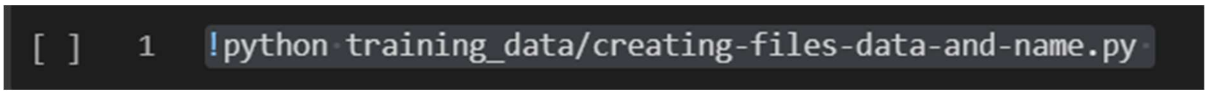
```
"""
File: creating-files-data-and-name.py
"""

# Creating files labelled_data.data and classes.names
# for training in Darknet framework

"""
Setting up full path to directory with labelled images
"""
full_path_to_images = 'training_data'

"""
Creating file classes.names
"""

# Defining counter for classes
c = 0

# Creating file classes.names from existing one classes.txt
with open(full_path_to_images + '/' + 'classes.names', 'w') as
names, \
     open(full_path_to_images + '/' + 'classes.txt', 'r') as txt:

    # Going through all lines in txt file and writing them into
names file
    for line in txt:
        names.write(line)  # Copying all info from file txt to
names

        # Increasing counter
        c += 1
```



```
"""
Creating file labelled_data.data
"""

# Creating file labelled_data.data
with open(full_path_to_images + '/' + 'labelled_data.data', 'w')
as data:
    # Writing needed 5 lines
    # Number of classes
    # By using '\n' we move to the next line
    data.write('classes = ' + str(c) + '\n')

    # Location of the train.txt file
    data.write('train = ' + full_path_to_images + '/' +
'train.txt' + '\n')

    # Location of the test.txt file
    data.write('valid = ' + full_path_to_images + '/' +
'test.txt' + '\n')

    # Location of the classes.names file
    data.write('names = ' + full_path_to_images + '/' +
'classes.names' + '\n')

    # Location where to save weights
    data.write('backup = backup')
```

Figure 18: Python Script for creating classes.names file and labelled_data.data file

The script used for the creation of these files is creating-files-data-and-name.py which can be executed using the following command.



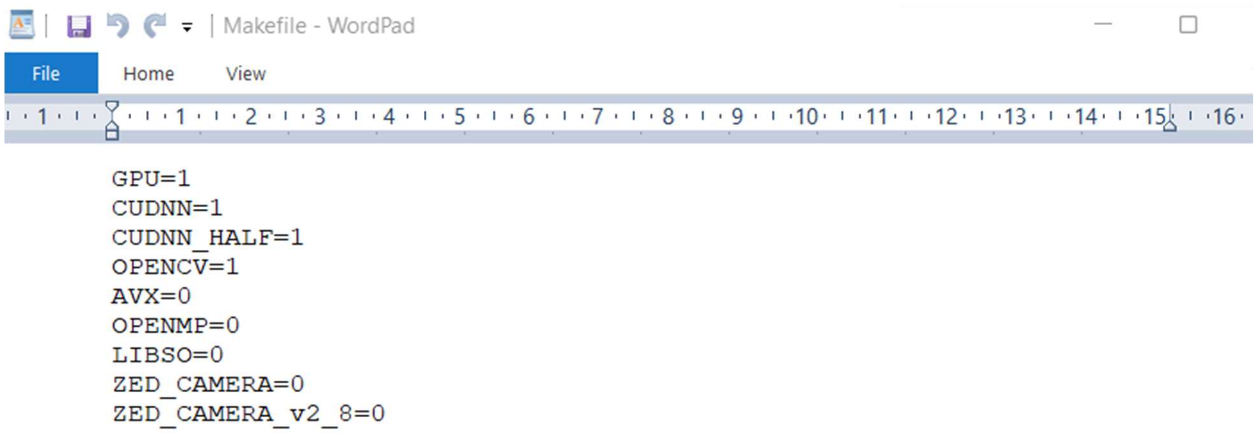
```
!python training_data/creating-files-data-and-name.py
```

Figure 19: Python script for creation of the files

8) Compiling the darknet repository -

To start the training process using our prepared dataset, we need to compile the darknet repository by executing the "make" command. If we want to include specific options like GPU, CUDNN, and OPENCV, we can modify the Makefile accordingly. Once the compilation process is complete, we will have an executable file for darknet.

Hence the changes made in the Makefile are as follows:

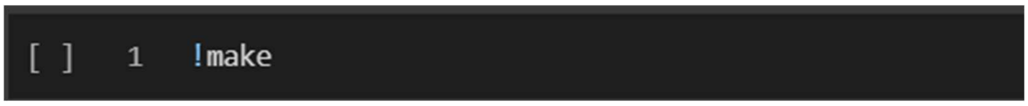


```
GPU=1
CUDNN=1
CUDNN_HALF=1
OPENCV=1
AVX=0
OPENMP=0
LIBSO=0
ZED_CAMERA=0
ZED_CAMERA_v2_8=0
```

Figure 20: Changes made in the make file

Here we have set the value of GPU and CUDNN to 1 inorder to speedup on GPU whereas CUDNN_HALF is set to 1 to further speedup up to three times. The value of OPENCV is set to 1 as it allows us to utilize the capabilities of OpenCV, which is a freely available computer vision library. With OpenCV, we gain access to a vast array of tools and functions that are beneficial for processing images and videos, making it a valuable asset for training neural networks for various tasks like object detection, recognition, and segmentation. By integrating OpenCV into the Darknet compilation process, we can leverage its features for augmenting and preprocessing the training data and also for visualizing the neural network's output during training and testing.

Now with the help of make command we can compile the darknet repository after making the desired changes in the Makefile.



```
!make
```

Figure 21: Compiling the darknet repository using make command

9) Creating the custom configuration for the Yolov3 model -
The existing configuration of the Yolov3 model comprises parameters that are required for training various numbers of classes but in our case we only need to train a limited number of classes i.e. 4 hence we need to redefine those parameters accordingly. We will first create another .cfg file named yolov3_custom.cfg and paste the entire code of the yolov3.cfg file.
Now we will change the following parameters:
i. batch = 16
ii. subdivisions = 8

The subdivision and batch size are determined on the basis of the GPU being used for the parallel processing. They together determine the number of images that shall be handled parallelly while training the model which is batch divided by subdivision.

iii. max_batches = 2000 x number of classes being trained
= 2000 x 4 = 8000

Max_batches gives the number of maximum iterations that will take place during the training of the model. Here, iterations are the number of batches needed to complete one epoch and epoch is the cycle of forward and backward pass on the complete dataset by the neural network.

With the help of above three parameters we determine the number of epochs that will take place during the entire training process.
As the total number of images in the dataset are 42 and the batch size is 16 then for a single epoch the number of iterations will be
42/16 = 2.62
Taking the ceiling of 2.62 we get 3 iterations in a single epoch.
Now to determine the total number of epochs we divide the max_batches or the total number of iterations with 3
8000/3 = 2667 epochs

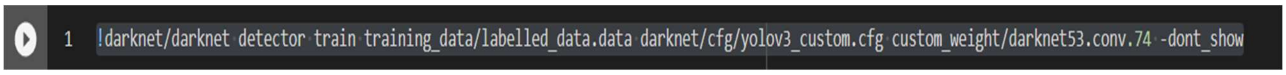
iv. steps = 6400, 7200
This is the 80% and 90% of the max_batches.

v. classes = 4
For each Yolo layer we update the number of classes.

vi. filter = 27
For each convolutional layer above the Yolo layer we update the number of filters with given formula
filter = (number of classes + 5) * 3
= (4+5) * 3 = 27

10) Training the custom model -

The following command is used for training the darknet model :



```
!darknet/darknet detector train training_data/labelled_data.data darknet/cfg/yolov3_custom.cfg custom_weight/darknet53.conv.74 -dont_show
```

Figure 22: Command for trining darknet model.

Here we give the path to our data file, the darknet file and the Yolo custom configuration file.
./darknet detector is a command used to execute the Darknet framework's detector application. . The detector command is followed by other arguments that specify details about the dataset, configuration file, and pre-trained weights for the model being trained or used for detection. The -dont_show flag is an optional parameter that can be used during YOLO model training with Darknet. Its purpose is to suppress the display of detection results during training. This flag is particularly helpful when working with large datasets or running training on a remote server, as displaying results in real-time can use up system resources and slow down the process. Additionally, if you're training without a GUI, displaying results may not possible.
After the training process is completed, we can download the final weights file from the backup folder and use it in the vehicle tracking model for better detections of the vehicle.

B. Starting timer of a cycle:

After the green signal of the previous cycle turns to red signal and the vehicles start to come and occupy the region of interest. We label the vehicles which we will be considering for our calculations as we are only considering the ones which actually have to wait for the red signal to turn into green. The vehicles which did not wait for the green signal are not considered because as it is they do not occupy any area in the region of interest for more than an instant. When the red signal finally turns green, we start the timer.

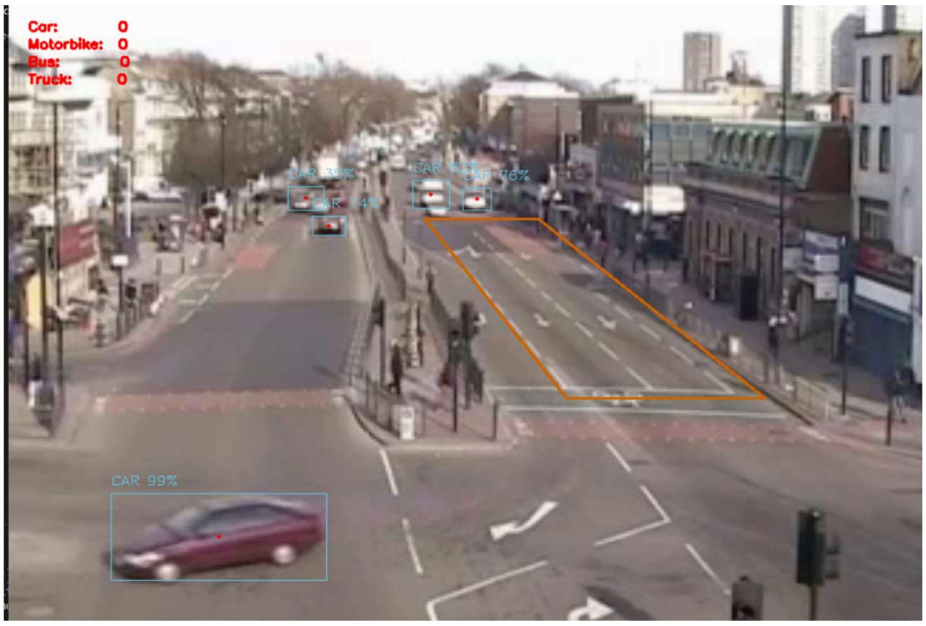


Figure 23: Region of interest initially

C. Detection and Classification:

The weights files which had 80 pre-trained classes have been used. Out of these only 4 classes were cars, trucks, buses, and bikes then imported into code and used for vehicle detection with the help of the OpenCV library. The minimal level of assurance necessary for successful detection is set at a threshold. After the model is loaded and an image is fed to the model that is continuous frames of images from video input, The minimal level of assurance necessary for successful detection is set at a threshold. It provides the output in JSON format, that is, as key-value pairs with labels serving as keys and their confidence and coordinates serving as values. Again, OpenCV can be used to draw the bounding boxes on the images from the labels and coordinates received. And their confidence values are also displayed around the boxes.

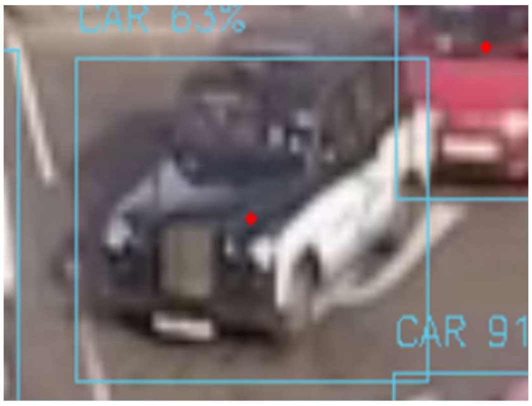


Figure 24: Car detected with 63% confidence.

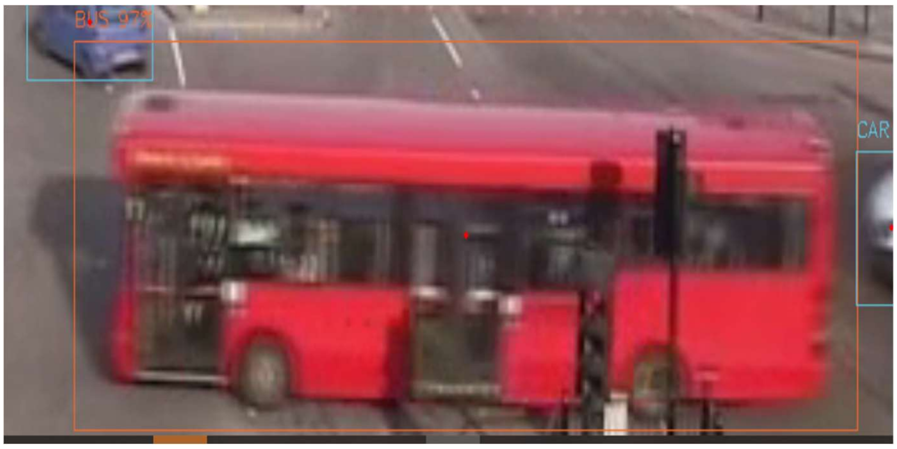


Figure 25: Bus detected with 97% confidence.

D. Tracking vehicles using Euclidean Distance:

Now centres of the bounding boxes are calculated, and ID is generated corresponding to each box generated containing all the values and if the centre is inside the region of interest near the arrival line and Euclidean algorithm is used to find out whether the vehicle entered the region for the first time or was already in the region of interest. If it turns out the vehicle is not present in already tracked vehicles, then the counter corresponding to that particular is increased. Otherwise, the old ID is replaced with the new ID to get updated parameters like the centre of the new position of the already tracked and counted vehicle.

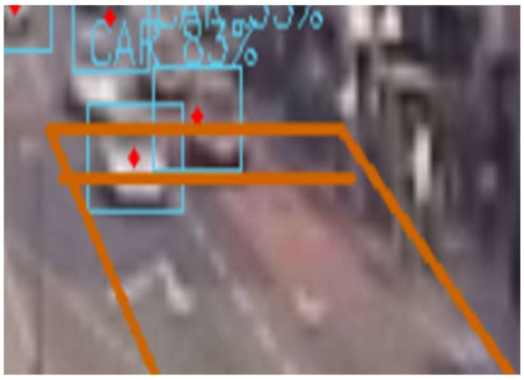


Figure 26: Vehicles entering tracking region at arrival line.

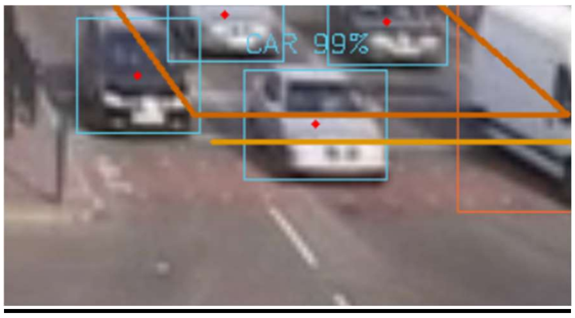


Figure 27: Vehicles leaving the tracking region at departure line.

E. Calculating Area of Occupancy:

So, the above detection, classification and tracking process is continued till the signal in the traffic light is still red and the fraction of area of the region occupied is also maintained which is calculated by multiplying the count of each of the class of vehicles with their pre-dined area occupancy. So as a result, we get the total percentage of area occupied in the region of interest at the end of the red-light timer which would be further used for training, testing and prediction process.

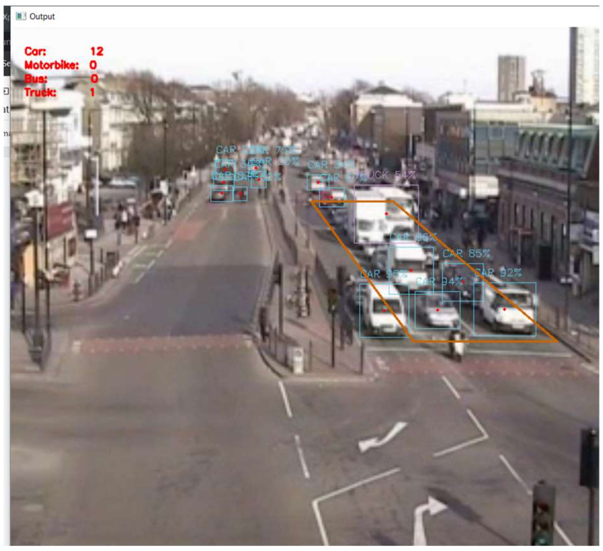


Figure 28: Area occupied by the vehicles in ROI.

F. Calculating ROI clearance time:

After the green signal appears the labelled vehicles start crossing the departure line. We track the total time taken by all the labelled vehicles to cross the departure line. When the region of interest is left with no labelled vehicles, the timer value obtained becomes the total time required to empty the region of interest. This timer value is further considered for calculations in the prediction model.

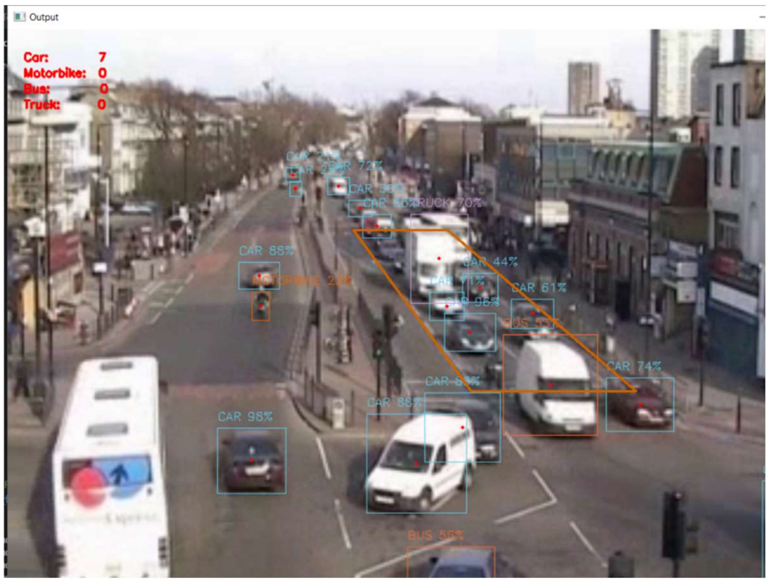


Figure 29: ROI with vehicles

G. Repetition of cycles for Database creation:

This cycle of red signal and green signal is then repeated multiple times to obtain how much time is required by different sets of vehicles to empty the region of interest. The vehicles considered for the creation of the dataset are divided into four different classes namely - “Car”, “Bike”, “Truck” and “Bus”. The total count of each type of vehicle is obtained. Thereafter we multiply the area of each class of vehicle with their respective count which is then added to obtain the total area occupied by the given number of vehicles. For the sake of simplicity, we have considered a standard value of the length and width of each class of vehicle as depicted in the table given below:

Table 3: Standard dimensions of vehicles and their calculated area

Vehicle Class	Car	Bike	Truck	Bus
Length (in meters)	4.9	1.5	12	15
Width (in meters)	1.9	0.6	2.5	2.5
Area (in meter squared)	9.31	0.9	30	37.5

Calculation of the total region of interest:

The complete area of a region cannot be calculated by the means of observation based on the video being considered for the collection of data hence we devised a method where we observed different instants of the video and the one with the highest amount of occupancy was then used for calculating the entire region of the area in the following way:

1. Calculate the total no of vehicles in the region of interest.
2. Calculate the area occupied by them with the help of the table given above.

3. As this is the maximum number of vehicles the region of interest can hold but we also need to consider the unoccupied region which can be roughly assumed to be 20% of the occupied.
4. Hence adding the area calculated in Step 2 and Step 3 we get the complete area which will be considered for finding the percentage of occupancy.

Given below is the frame in which the highest number of vehicles were detected, and we have considered this frame for the calculation of the Region of Interest. The region of Interest is highlighted in Yellow.

Count of total cars in the ROI = 18

Area of one car = 4.9 m x 1.9 m (values taken from Table 1)

Area of 18 cars = 18 x 4.9 x 1.9

= 167.58 m^2

Area unoccupied = 20% of Area occupied

= 20% of 167.58

= 33.516 m^2

Total area of ROI = 167.58 + 33.516

= 201.096 m^2

The total area obtained above is considered for calculating the Percent area occupancy.

The Percent of Area Occupancy is given by:

***ao* = Area occupied by the total number of vehicles in the region of interest/ Total area of the region of interest x 100**

The dataset is created with two columns namely the time and the ***ao*** (Percent of Area Occupancy).

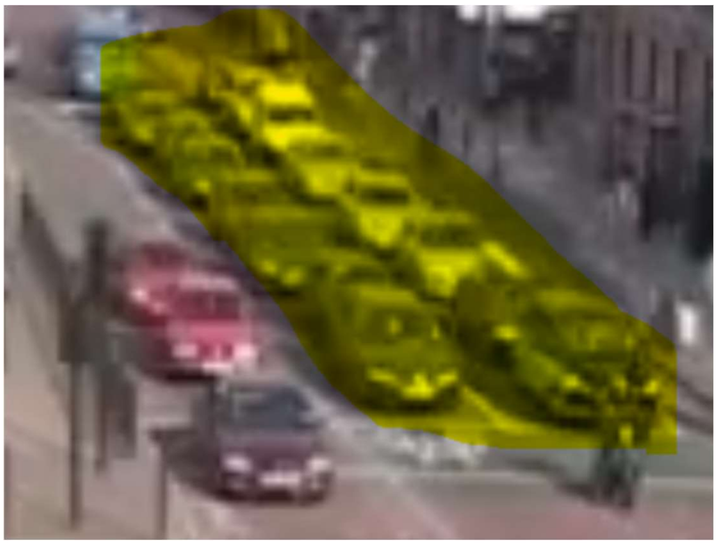
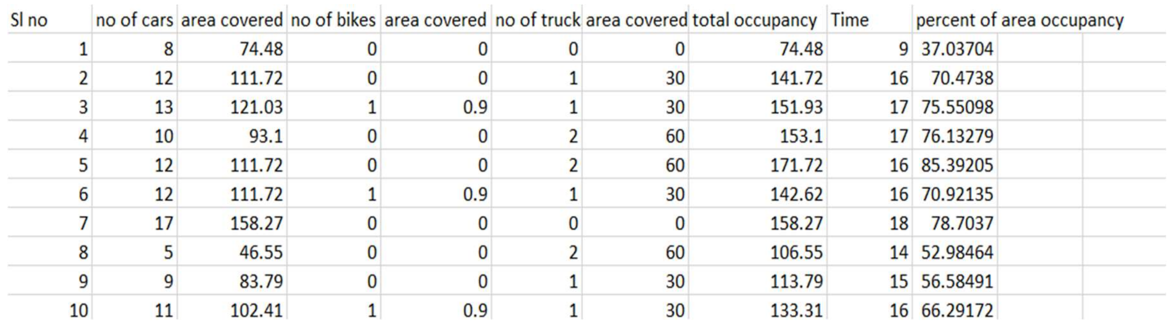


Figure 30: Frame considered for calculating ROI.

The complete dataset looks like this:

Table 4: Database considered for training and testing.



Sl no	no of cars	area covered	no of bikes	area covered	no of truck	area covered	total occupancy	Time	percent of area occupancy
1	8	74.48	0	0	0	0	74.48	9	37.03704
2	12	111.72	0	0	1	30	141.72	16	70.4738
3	13	121.03	1	0.9	1	30	151.93	17	75.55098
4	10	93.1	0	0	2	60	153.1	17	76.13279
5	12	111.72	0	0	2	60	171.72	16	85.39205
6	12	111.72	1	0.9	1	30	142.62	16	70.92135
7	17	158.27	0	0	0	0	158.27	18	78.7037
8	5	46.55	0	0	2	60	106.55	14	52.98464
9	9	83.79	0	0	1	30	113.79	15	56.58491
10	11	102.41	1	0.9	1	30	133.31	16	66.29172

Considering the values given in Table 3, below is the demonstration of Percent of area occupancy that has been calculated.

The area covered by each type of vehicle is given in Table 2 above.

Given -

No. of Cars = 8

Area covered by of 8 Cars = 8 x 9.31 = 74.48 m^2

No. of Bikes = 0

Area covered by 0 Bikes = 0 x 0.9 = 0 m^2

No. of Trucks = 0

Area covered by Trucks = 0 x 30 = 0 m^2

Total area covered by all the vehicles = 74.48 + 0 + 0 = 74.48 m^2

Percent area occupancy ($\boldsymbol{ao}$) = Total area covered by all the vehicles / Total area of ROI x 100

$\boldsymbol{ao}$ = 74.48 / 201.096

$\boldsymbol{ao}$ = 34.03704 %

4. REQUIREMENT DESCRIPTION

Hardware Requirement Specification:

A. High-resolution camera:

The complete model is based on the input requirement of a clear and distinctive video of the traffic signal junction. Hence it is important to have a high-resolution camera which enables good quality capture of the video which in turn helps in better detection of the vehicles and classification based on that detection also becomes accurate. Therefore, a high-resolution camera leads to error-free results.

B. Camera should be aligned at the correct angle:

Also, the camera must be aligned at a correct angle which allows the complete capture of the entire region of interest. Once you start the detection of vehicles all the vehicles in the region of interest must be captured properly hence the camera must be adjusted accordingly.

C. Traffic lighting system:

Traffic lighting systems must be installed at the junctions where we need to implement the Intelligent Traffic Light Control System Using YOLO Based On Spatial Occupancy as only then we will be able to analyse various time intervals according to the traffic light signals namely Green, Red and Yellow. Upon determination of various time intervals, we then can feed them to the model along with their area of occupancy in order to predict traffic signal timings later on.

D. GPU for Yolo detection/classification:

In order to run the YOLO model for smooth and real-time detection and classification we need to make use of the GPU as it provides faster and smoother processing capability. GPU technology has progressed beyond processing high-performance graphics to support use cases requiring quick data processing and highly parallelized calculations. As a result, GPUs offer the parallel processing required to support the intricate multi-step machine learning processes.

E. Arduino

In order to detect the traffic signals in real-time for feeding in the prediction model and for the purpose of calculating the time duration we need to make use of the Arduino. It is used for converting physical observation into digital data which can be further used for analysing and determining various variables.

Tools and Technologies:

- **PYTHON**

a) matplotlib: A Python-based library for data visualizations and graphical plotting, based on NumPy, that works across multiple platforms. As such, It offers a workable, open-source, free substitute for MATLAB. Using the Matplotlib API (Application Programming Interface), developers can also incorporate charts into GUI applications.

b) pandas: The "relational" or "labelled" datasets can be easily and intuitively manipulated with the help of this Python package, which offers a very quick, practical, and impressive data structure. It is meant to serve as a very basic, high-level building block for Python data analysis used in practical, real-world applications. Also, the more general objective of being the most strong, adaptable, and impressive open-source data manipulation and analysis tool available in any language.

c) numpy: NumPy is a Python library used for manipulating arrays. It also has functions for working in the domain of linear algebra, Fourier transforms, and matrices. Python has lists that serve the purpose of arrays but are slower. NumPy aims to provide array objects that are up to 50x faster than traditional Python lists. NumPy's array objects are called ndarrays and provide several support functions that make working with ndarrays very easy. Arrays are very commonly used in data science where speed and resources are very important.

d) csv: The CSV module implements classes for reading and writing table data in CSV format very easily and efficiently. Programmers and users can easily and easily say "write this data to Excel's preferred format" or "read data from this file generated by Excel" without knowing the exact details of the CSV format used by Excel. can be said effectively. Programmers can also write her CSV formats that other applications understand or define their own special her CSV formats.

e) cv2: OpenCV is a great tool for performing image processing and computer vision tasks. This is an open-source library that can be used to perform tasks such as face recognition, object tracking, and landmark detection. It supports multiple languages including Python, Java C++. The library is loaded with hundreds of useful functions and algorithms, all of which are available for free. Some of these functions are very common and are used in almost every computer vision task.

f) math: This module provides access to the math functions defined by the C standard and supports a wide variety of functions. The most commonly used are math.ceil(x), math.comb(n, k), math.factorial(n, k), math.floor(x), etc.

g) sklearn: scikit-learn (Sklearn) is the most useful, impressive and robust machine learning library in Python modules. Through a consistent Python interface, it provides users with a selection of efficient machine learning, statistical, and modeling tools, including many functions such as classification, regression, clustering, dimensionality reduction, and more.

- **MACHINE LEARNING**

Machine learning is a branch, or subset, of artificial intelligence. Artificial intelligence can be broadly defined as the ability of machines to accurately mimic intelligent human behavior. Artificial intelligence systems are used to perform complex tasks in a manner similar to human problem solving, making the results more realistic and accurate. The capabilities of machine learning systems can be descriptive. In other words, the system uses the data to explain what happened. Predictive. In other words, the system uses data to predict what will happen. That is, a system or model uses data to suggest actions to take for assigned tasks.

- **COMPUTER VISION**

It is the field of artificial intelligence(AI) which deals with how computers can be made to understand and interpret the visual world. Computer vision involves the development of algorithms and systems that can process, analyze, and understand images and videos in order to extract useful information and insights. Computer vision techniques are used in a variety of applications also including object recognition, image classification, face detection, and image segmentation. These techniques are often used in conjunction with machine learning algorithms to build systems that can learn to recognize and classify objects, faces, and other features in images and videos.

Computer vision systems can be used for tasks such as identifying objects in an image or video, counting the number of objects in a scene, or detecting changes in an image over time. Computer vision algorithms can also be used to extract text from images, detect and track moving objects, and recognize patterns in images.

Overall, computer vision plays an important role in enabling computers to understand and interact with the world around them in a very intelligent and natural way.

- **TRANSFER LEARNING**

  Transfer learning works by taking a pre-trained model, which has been trained on a large dataset, and then reusing the weights of the model's layers for a different but related task. The approach saves computational resources and time since the model does not need to be trained from scratch. Instead, only the final layer or a few layers of the model are fine-tuned to adapt to the new task. This process allows the model to learn quickly and generalize well to new data.

  One of the biggest advantages of transfer learning is that it enables us to work with less data, which is a common problem in many machine learning applications. Pre-trained models have already learned general features and patterns from a large dataset, which can be transferred to a new task. This feature makes transfer learning very useful in applications where the new dataset is small, and we do not have enough data to train a new model from scratch.

<u>Module and their brief descriptions:</u>

- **YOLO**

  YOLO (You Only Look Once) is a popular object detection algorithm that was developed by Joseph Redmon and Ali Farhadi in 2015. It is a type of convolutional neural network (CNN) that is designed to detect objects in images and videos quickly and accurately.

  The YOLO algorithm works by dividing the input image into a grid of cells, and each cell is responsible for predicting whether there is an object within its area and, if so, which class of object it is. YOLO makes predictions for each cell independently, so it can process the entire image in a single pass and provide real-time object detection.

  The main advantage of YOLO is the speed because it can process images and videos at a much faster rate than other object detection algorithms. It is also relatively simple to implement, making it a popular choice for object detection tasks in a variety of applications.

- **POLYNOMIAL REGRESSION**

  The relationship between the independent variable, let's say x, and the dependent variable, let's say y, is modelled as an nth-degree polynomial in polynomial regression. When relationships between variables are not linear, polynomial regression can be used.

In polynomial regression, the model function is defined as

y = b0 + b1x + b2x^2 + ... + bn*x^n

where b0, b1, ..., bn are the coefficients of the polynomial and n is the degree of the polynomial. The coefficients are determined through the process of fitting the model to the data.

Polynomial regression can be used to model complex relationships between variables and can be more accurate than linear regression for certain types of data. However, it may be more prone to overfitting, particularly when the degree of the polynomial is high.

- **EUCLIDEAN TRACKING**

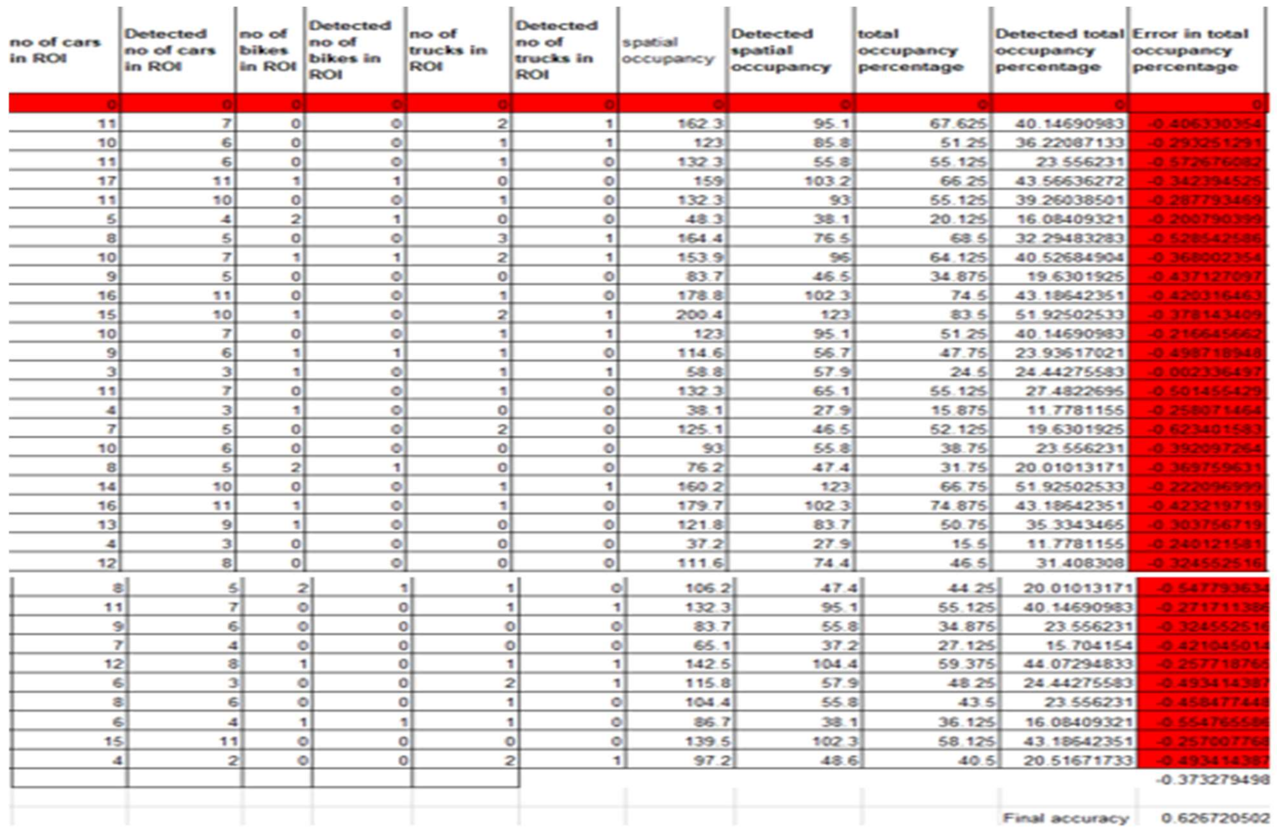
Euclidean tracking is a method of object tracking in which the position and orientation of an object are determined by computing the Euclidean distance between the object and a reference frame. Euclidean tracking is generally used in robotic systems and computer vision applications to track the motion of objects in real-time.

In Euclidean tracking, both the position and orientation of the object are represented by a set of coordinates in the reference frame. The object is detected in successive frames of a video or image sequence, and again the position and orientation of the object are computed based on the Euclidean distance between the object and the reference frame.

Euclidean tracking can be used to track the motion of objects in 3D space, and it is often used in combination with other techniques, such as Kalman filtering or other, to improve the accuracy of the tracking. It is a relatively simple and efficient method of object tracking, but it can be sensitive to noise and may not perform well in cases where the object is partially occluded or has a complex shape.

5. RESULT ANALYSIS

After implementing transfer learning and obtaining the custom Yolo weights, we used them for detection of vehicles in the video dataset. The count of each type of vehicle for all the cycles after using the custom model was tabulated along with the manually observed counts. Below is the table depicting the same:



no of cars in ROI	Detected no of cars in ROI	no of bikes in ROI	Detected no of bikes in ROI	no of trucks in ROI	Detected no of trucks in ROI	spatial occupancy	Detected spatial occupancy	total occupancy percentage	Detected total occupancy percentage	Error in total occupancy percentage
0	0	0	0	0	0	0	0	0	0	0
11	7	0	0	2	1	162.3	95.1	67.625	40.14690983	-0.406330354
10	6	0	0	1	1	123	85.8	51.25	36.22087133	-0.293251291
11	6	0	0	1	0	132.3	55.8	55.125	23.556231	-0.572676082
17	11	1	1	0	0	159	103.2	66.25	43.56636272	-0.342394525
11	10	0	0	1	0	132.3	93	55.125	39.26038501	-0.287793469
5	4	2	1	0	0	48.3	38.1	20.125	16.08409321	-0.200790399
8	5	0	0	3	1	164.4	76.5	68.5	32.29483283	-0.528542586
10	7	1	1	2	1	153.9	96	64.125	40.52684904	-0.368002354
9	5	0	0	0	0	83.7	46.5	34.875	19.6301925	-0.437127097
16	11	0	0	1	0	178.8	102.3	74.5	43.18642351	-0.420316463
15	10	1	0	2	1	200.4	123	83.5	51.92502533	-0.378143409
10	7	0	0	1	1	123	95.1	51.25	40.14690983	-0.216645662
9	6	1	1	1	0	114.6	56.7	47.75	23.93617021	-0.498718948
3	3	1	0	1	1	58.8	57.9	24.5	24.44275583	-0.002336497
11	7	0	0	1	0	132.3	65.1	55.125	27.4822696	-0.501455429
4	3	1	0	0	0	38.1	27.9	15.875	11.7781155	-0.258071464
7	5	0	0	2	0	125.1	46.5	52.125	19.6301925	-0.623401583
10	6	0	0	0	0	93	55.8	38.75	23.556231	-0.392097264
8	5	2	1	0	0	76.2	47.4	31.75	20.01013171	-0.369759631
14	10	0	0	1	1	160.2	123	66.75	51.92502533	-0.222096999
16	11	1	0	1	0	179.7	102.3	74.875	43.18642351	-0.423219719
13	9	1	0	0	0	121.8	83.7	50.75	35.3343465	-0.303756719
4	3	0	0	0	0	37.2	27.9	15.5	11.7781155	-0.240121581
12	8	0	0	0	0	111.6	74.4	46.5	31.408308	-0.324552516
8	5	2	1	1	0	106.2	47.4	44.25	20.01013171	-0.547793634
11	7	0	0	1	1	132.3	95.1	55.125	40.14690983	-0.271711386
9	6	0	0	0	0	83.7	55.8	34.875	23.556231	-0.324552516
7	4	0	0	0	0	65.1	37.2	27.125	15.704154	-0.421045014
12	8	1	0	1	1	142.5	104.4	59.375	44.07294833	-0.257718765
6	3	0	0	2	1	115.8	57.9	48.25	24.44275583	-0.493414387
8	6	0	0	1	0	104.4	55.8	43.5	23.556231	-0.458477448
6	4	1	1	1	0	86.7	38.1	36.125	16.08409321	-0.554765586
15	11	0	0	0	0	139.5	102.3	58.125	43.18642351	-0.257007768
4	2	0	0	2	1	97.2	48.6	40.5	20.51671733	-0.493414387
										-0.373279498
									Final accuracy	0.626720502

Here the actual number of vehicles in the ROI are then used for computing the spatial occupancy which again is used for calculating the total occupancy percentage.Similarly the detected number of vehicles in the ROI with the use of custom made model is used for calculating the detected total occupancy percentage.

The difference in the actual total occupancy percentage and detected total occupancy percentage is then tabulated as the error in total occupancy percentage. The average of each of these errors is then used for calculating the overall error in the detected model.

In this case the Average Error comes out to be 37.33% which gives an accuracy of 62.67%.

Hence the overall detection accuracy of the model is 62.67%.

6. CONCLUSIONS AND FUTURE DIRECTIONS

Conclusion:

In the project **Implementing Transfer Learning in YOLO for Intelligent Traffic Light Control System** we implemented transfer learning to train the Yolov3 model with the dataset that was created using the video which is being used for the purpose of tracking. Transfer learning can be a great tool for training a custom model with our custom dataset. It diversifies the implementation of Yolov3 and makes it inclusive of a greater number of classes and even allows us to train the same classes with different dataset unlike the one used by the traditional Yolov3 model. We have also used this feature in our advantage to train our model to become more robust.

The pure intention of our work was to train the model for more accurate results and the accuracy which was obtained was about 63%.

Challenges Faced:

1. **Limited computing resources**
   The complete model was trained on the free resources available under the Google Collab hence only a limited amount could be utilized for training the model and hence we could not train the model for longer durations.
2. **Dataset Creation**
   The video input considered for the creation of the dataset was a low-resolution video hence during the creation of the dataset each and every cycle was manually observed. Though the input was low resolution still we tried using the best possible model for the most accurate results.
3. **Clustering**
   The proposed model is not based on the clustering methodologies as they were not inclusive of all the possible scenarios. They also did not give desired results.
4. **Calculation of Region of Interest**
   The video considered for taking observations gave an angled view of the Region of interest of the road hence for precise calculation it was necessary for taking in account the most correct area of ROI. In order to tackle this problem, we devised a method for calculating ROI as a relative measurement with respect to the vehicles present in the ROI.
5. **Camera Threshold Range**
   The proposed model is independent of the height and angle of the camera till it gives a clear view of the ROI. Hence, we can keep a threshold range for the height of the camera that the height should lie in the given range.

Future Prospects:

More classifications can be made to determine whether the car is small size, medium size or large because each type of car would have different area occupancy.

1. The data between subsequent traffic junctions can be shared and considered for much better traffic flow. Thus, the traffic signals can be synchronized, and dynamic clearance time can be used for clearing congestion in a much more efficient manner from all consecutive traffic junctions.
2. Also, a lane detection model can be implemented and through which the region of interest can be determined according to the requirement and thus increase the accuracy of the model.
3. Adaptation for emergency vehicles such as ambulances and fire brigade: Emergency and priority vehicles such as ambulances need to get through traffic lights faster with minimal congestion and waiting time. In addition to recognizing vehicles, the model can be trained to recognize emergency vehicles such as ambulances and fire trucks and adjust the timer accordingly. This gives the vehicles above priority and crosses the signal as soon as possible.

REFERENCES

1. A. A. Zaid, Y. Suhweil and M. A. Yaman, "Smart controlling for traffic light time," 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), 2017, pp. 1-5, doi: 10.1109/AEECT.2017.8257768.

2. Renjith Soman "Traffic Light Control and Violation Detection Using Image Processing".IOSR Journal of Engineering (IOSRJEN), vol. 08, no. 4, 2018, pp. 23-27.

3. N. S. Nafi and J. Y. Khan, "A VANET based Intelligent Road Traffic Signalling System," Australasian Telecommunication Networks and Applications Conference (ATNAC) 2012, 2012, pp. 1-6, doi: 10.1109/ATNAC.2012.6398066.

4. Cao, Z., Jiang, S., Zhang, J., and Guo, H. (2017). A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion. IEEE Transactions on Intelligent Transportation Systems, 18(7):1958–1973.

5. Hausknecht, M., Au, T. C., and Stone, P. (2011). Autonomous intersection management: Multi-intersection optimization. In ICIRS, pages 4581–4586.

6. Zhao, J., Li, W., Wang, J., and Ban, X. (2016). Dynamic traffic signal timing optimization strategy incorporating various vehicle fuel consumption characteristics. IEEE Transactions on Vehicular Technology, 65(6):3874–3887.

7. M. M. Gandhi, D. S. Solanki, R. S. Daptardar and N. S. Baloorkar, "Smart Control of Traffic Light Using Artificial Intelligence," 2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 2020, pp. 1-6, doi: 10.1109/ICRAIE51050.2020.9358334.

8. Santhosh Kelathodi Kumaran, Debi Prosad Dogra, Partha Pratim Roy, Queuing theory guided intelligent traffic scheduling through video analysis using Dirichlet process mixture model, Expert Systems with Applications.

9. Yu, F., Xiu, X., Li, Y. A Survey on Deep Transfer Learning and Beyond. Mathematics 2022, 10, 3619. https://doi.org/10.3390/math10193619.

10. Thakur, Dr. Narina & Nagrath, Preeti & Jain, Rachna & Saini, Dharmender & D, Jude. (2021). Object Detection in Deep Surveillance. 10.21203/rs.3.rs-901583/v1.

11. Lihe Hu, Yi Zhang, Yang Wang et al., "Salient Semantic Segmentation Based on RGB-D Camera for Robot Semantic Mapping", Applied Sciences 13(6), pg. 3576, (2023); doi:10.3390/app13063576.

12. X. Liang, X. Du, G. Wang and Z. Han, "A Deep Reinforcement Learning Network for Traffic Light Cycle Control," in IEEE Transactions on Vehicular Technology, vol. 68, no. 2, pp. 1243-1253, Feb. 2019, doi: 10.1109/TVT.2018.2890726.