

REAL TIME SPEED TRACKING AND DYNAMIC EXCEED LIMIT SYSTEM USING COMPUTER VISION TECHNIQUES

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Engineering Degree in Electronics and Communication Engineering

by

KADIMISETTY MOHAN CHANDRA REDDY(39130194)

GARAGABOINA VEERA BABU(39130138)



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SCHOOL OF ELECTRICAL AND ELECTRONICS

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600119

APRIL 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project is the bonafide work of **KADIMISETTY MOHAN CHANDRA REDDY (39130194)** and **GARAGABOINA VEERA BABU (39130138)** Who carried out the project entitled **“REAL TIME SPEED TRACKING AND DYNAMIC EXCEED LIMIT SYSTEM USING COMPUTER VISION TECHNIQUES”** under supervision for JANUARY 2023 to APRIL 2023.

Internal Guide
Mr.V. KALIST M.E

HEAD OF THE DEPARTMENT

Dr.T.RAVI, M.E, Ph.D.,

Submitted for Viva voice Examination held on 24.04.2023

Internal Examiner

External Examiner

DECLARATION

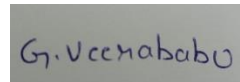
We **KADIMISETTY MOHAN CHANDRA REDDY(39130194),GARAGABOINA VEERA BABU(39130138)** hereby Declare that the Project Report entitled “**REAL TIME SPEED TRACKING AND DYNAMIC EXCEED LIMIT SYSTEM USING COMPUTER VISION TECHNIQUES**” done by us under the guidance of **Mr.V KALIST,M.E** at Sathyabama Institute of Science and Technology, Chennai-600119 is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

DATE: 24.04.23

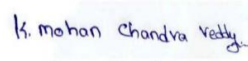
PLACE: Chennai

Signature of the Candidate:

1.

Handwritten signature of G. Veerababu in blue ink on a grey background.

2.

Handwritten signature of K. Mohan Chandra Reddy in blue ink on a light blue background.

ACKNOWLEDGEMENT

I AM pleased to acknowledge our sincere thanks to Board of Management of **SATHYABAM A** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

I convey our thanks to **Dr. N.M. NANDHITHA, M.E., Ph.D., Dean**, School of Electrical and Electronics, **Dr. T. RAVI, M.E., Ph.D., Head of the Department, Dept. of Electronics and Communication Engineering** for providing us necessary support and details at the right time during the progressive reviews.

I would also like to express our sincere and deep sense of gratitude to our Project guide **Mr.V.KALIST,M.E** for his valuable guidance and suggestions and constant encouragement during the progressive reviews for our project

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **Electronics and Communications Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In our world, Lot of accidents are happening on every day basis because of speed limit exceed by various vehicles. Mainly, road accidents are the major one to happen and number of death per day also increases day by day. Detect, classify and keep track, in real-time, on different kinds of objects or vehicles that are moving on a road is crucial for traffic managements systems, among other research areas. In this project , our proposed system is based on computer vision to track real time speed of the vehicles and also the vehicles which are exceeding the normal speed limit will be noted. In the existing system, only the type of the vehicle is detected and also the it is not done in real time. The datasets that is used in the existing system is also very low. Smart surveillance system has been defined as the system that utilizes computer vision and pattern recognition technologies for information analysis. The smart surveillance system research development is rapidly growing since the need for more reliable security in public places such as railways, restaurants, schools, shopping malls is accelerating. Due to the increasing terrorist attack and crime activities which occurs in our daily life, the recent world events have shifted from “investigation of incidents to prevention of potentially catastrophic incidents” that require minimum supervision from the human where surveillance activities are mainly carrying out by the computers or embedded systems. The same required for vehicle speed detection and limiting the speed.

TABLE OF CONTENT

CHAPTER NO	TITLES	PAGE NO
	ABSTRACT	v
	LIST OF FIGURES	ix
	LIST OF TABLES	x
	LIST OF ABBREVIATIONS	xi
1	INTRODUCTION	1
	1.1 Existing system	3
	1.2 Proposed system	4
	1.3 Difference between existing and proposed system	
2	LITERATURE SURVEY	6
3	AIM AND SCOPE OF THE PROJECT	12
	3.1 Aim and objective	12
	3.2 Removing unwanted images	13
	3.3 Removing noise in images	
	3.4 Working and vehicle monitoring	
	3.4.1 label processing	
	3.5 Creating a resenent model	14
	3.5.1 Split the data set	

	3.5.2 Loading the data into batch	15
	3.5.3 Setting cuda GPU and training the model	16
	3.5.4 defining loss and optimise a function	16
	3.6 OBJECTIVES	18
	3.7 CONCLUSION	18
4	METHEDOLOGY	
	4.1 INTRODUCTION	21
	4.2 YOLO V4 TRACKING	22
	4.3 INPUT	24
	4.4 SOFTWARE TOOLS REQUIRED	24
	4.4.1 DARK NET	26
	4.4.2 CUDA TOOL KIT	26
	4.4.3 OPEN CV	26
	4.4.4 PYTHON	26
	4.4.5 TRAINING DATA	27
	4.4.6 PRE-TRAINED WEIGHTS	27
5	RESULT AND DISCUSSION	34
	5.1 RESULT	34
	5.2 DISCUSSION	34
	5.2.1 Software development	35
	5.2.2 Data collection and annotation	36
	5.2.3 Calibration	36

5.3	Data set collection module 1	36
5.4	Object detection and tracking module 2	37
5.5	Alert and warning module	38
5.6	Performance matrix	39
6	Conclusion and Future scope	42
6.1	Conclusion	42
6.2	Future Scope	42
	REFERENCE	44

LIST ODF FIGURES

FIGURE	TITLE	PAGE NO
4.1	Tracking Way of Vehicle	22
4.2	Tracking Over view	23
4.3	Yolo v4 Architecture	23
4.4	Install Visual Studio Code	30
4.5	Python Installer	30
4.6	Open Command Prompt Terminal	31
4.7	Copy Source Link	32
4.8	Past Link In Terminal	33
4.9	Stimulation Output	34
5.1	Accuracy With Out Yolo v4	40
5.2	Accuracy of Yolo v4	41

LIST OF TABLES		
TABLE NO	TITLE	PAGE NO
5.6	Performance matrix	40

LIST OF ABRIVATIONS

ABBREBVIATIONS

DEFINITIONS

CMD	Chief Managing Director
LIDAR	Laser Image Detection And Ranging
ROI	Range Of Interest
CCTV	Closed Circuit Television
CUDA	Compute Unified Device Architecture

CHAPTER 1

INTRODUCTION

Smart surveillance system has been defined as the system that utilizes computer vision and pattern recognition technologies for information analysis. The smart surveillance system research development is rapidly growing since the need for more reliable security in public places such as railways, restaurants, schools, shopping malls is accelerating. Due to the increasing terrorist attack and crime activities which occurs in our daily life, the recent world events have shifted from “investigation of incidents to prevention of potentially catastrophic incidents” that require minimum supervision from the human where surveillance activities are mainly carrying out by the computers or embedded systems. The same required for vehicle speed detection and limiting the speed.

Detection of vehicle and tracking of speed is the crucial part of town planning. In the last decade, vision-based traffic monitoring system has received considerable attention. This can be done with the help of vehicle detection and speed monitoring. The monitoring system gives various information about, vehicle count, traffic congestion and speed of the vehicle. One of the root cause of road accidents is speed. Extracting frames from the video and comparing the speed between two given points can be used to determine whether the car is moving above the permissible limit or not. There are many algorithms available for extraction of vehicles from the background.

Traditionally radar systems were used for such applications but had some limitations. So to overcome the limitations in existing methods, various techniques have been developed for vehicle speed determination using image processing.[7] But the main factors that would affect these image processing algorithms is, waving of tree branches, camera noise, illuminations. The goal of this current research is to develop an automatic vehicle counting system along with the detection of speed, which can process videos recorded from stationary cameras over roads e.g. CCTV cameras installed near traffic intersections / junctions and counting the number of vehicles passing a spot in a particular time for further collection of vehicle / traffic data.

Vehicle Speed surveillance is a predominant factor in enforcing traffic laws. Traditionally vehicle speed surveillance was done using a radar technology which consists of a radar

gun and a radar detector. Radar is an acronym that stands for Radio Detection and Ranging. Radar systems create radio waves, a form of electromagnetic energy that can be directed out into the air where the signals produced travel at the speed of light roughly 186,000 miles per second, or 3.08×10^8 meters per second. The transmission of these signals and the collection of returned energy that bounce off of objects in the path of the radars transmission (called returned pulses) is what allows radar to be used to detect objects and range them, meaning establish their position and distance relative to the radar systems location. When a radar is used to detect the speed of an object (for example, when a police officer with a stationary radar gun is detecting the rate at which a car is moving), it does so by taking advantage of a phenomenon that occurs whereby the relative to the radar. If the car is moving toward the radar device, the return signal radio wave frequency increases.

The radar gun can then use this change in frequency to determine the speed at which the car is moving. This principle, which establishes that the difference between the frequency of the emitted pulse and the frequency of the return pulse varies with the relative motion of the source to the object, is called the Doppler effect. So, while the distance of an object can be established by the amount of time that it takes to detect the return pulse, the speed of an object can be detected by establishing the change in the pulse characteristics between transmitted and received echo.

This provides a velocity along the direction in which the radar is pointing, termed the radial velocity. One point to note is that the pulse characteristic changes used to establish the speed of a moving object like a car will depend on the relative position of the car to the radar. The measured speed will be accurate if the car moves directly towards the radar. But if the cars motion is at an angle relative to the radar guns line of sight, the speed being measured will be a component of the actual speed of the vehicle. This principle is known as the cosine error effect. Because of these errors, the United States of Americas law keeps a buffer of 8km which is caused because of the above-mentioned error. Also, radar technology can track one vehicle at a time. The paper deals with vehicle detection and speed tracking which is explained further.

1.1 Existing System

Currently, most of the vehicle speed detector system for the purpose traffic light speed law enforcement is based on RADAR (Radio Detection and Ranging) methods. This method is using some kind of active devices, which are much more expensive compare to static camera system. At the same time, the RADAR based system is need and necessary to integrate with fast and high resolution imaging devices to capture the image for the vehicles. Other than RADAR method, there are audio system, LIDAR (Laser Imaging Detection And Ranging)[3] The way of how the RADAR method works is known as Doppler shift phenomenon. Doppler shift phenomenon is experience by us in our daily life frequently.

It occurs when sound is generated by a moving vehicle and will create a sonic boom. The frequency of the sound will be change when the sound wave reflects back to the wave generator and the scientists will calculate the speed of the vehicle by using the frequency variation. However, the equipment to use this method is costly. So, finding other equipment to reduce the cost is necessary. Image processing technology can serve this well.

Due to the RADAR method need integrate with expensive equipment, the cost to develop vehicle speed detector system with this method will be high. Therefore, develop a low cost vehicle speed detector system is needed. This low cost vehicle speed detector system will use image processing method which will only need a video recorder and a computer is enough to create this system.

1.2 Proposed System

The system is divided in several stages: ROI selection, detection of moving objects, clustering process, tracking, single-frame classification and counting. The global overview of the proposed vehicle detection system is shown. Acquire images from the road is the first step of the proposed vehicle detection algorithm, for this is necessary to place a HD- RGB camera over the road. All obtained frames are converted to grayscale. Then, in order to maximize the performance of the algorithm, a selection of a proper ROI (Region Of Interest) is needed. Within this area the search for moving objects is performed. The region must contain the zone where the aim traffic flow is located, as well the line for

vehicle count. This selection may change according to the type of road. To detect moving vehicles a temporal difference is applied, this technique consists in obtain the pixel value difference between two consecutive frame.

But in the present paper the comparison is not made pixel to pixel. The ROI is divided in n square segments with same length k , for each n -square their mean $m1$ of grayscale level is computed, then it is compared with the average of the previous frame $m0$. A n -square as a object in motion if the mean of the n -square from the present frame has a larger difference than the one of the previous frame plus a threshold: $m1 > m0 + th$ or $m1 < m0 - th$. If these conditions are met, the assessed n -square is marked and tagged as "M". In the next step it is needed to cluster the multiple regions that contain movement, in order to consolidate the different moving objects "MO" in scene. Then, all labeled "M" squares that are continuous are merged into a Temporal Object "TOn", a number is assigned to any "Ton. This second clustering process is shown in figure 4 (b) When a frame X has been analyzed and all "MO"s have been consolidated, the appropriate data of those objects are stored in order to be able to track them in the next frame $X+1$. With all the "MO"s constituted in frame $X+1$, these are verified with the stored objects from frame X , if the data match, is the same object, otherwise these unmatched "MO" 680 will be analyzed according their size, color and position in order to declare it or not a new object. In case of be too close to others, it is determined if there is a gap with different color, between related objects, if this is true, they are different objects. The moving objects are tracked down since they enter the ROI, but only the ones that appears before and don't disappear until they reach the counting line are counted, and just at that moment, the objects are classified according to their size, this mean that the area of the object is used to tag it. When the objects reach the lower limit of the ROI they are eliminated, therefore the tracking process over that object stops.

1.3 Difference between Existing and Proposed System

The existing systems are not very robust in prediction. They have not evolved according to technological advancements and time. The results are also unsatisfactory and may lead to erroneous prediction of speed limit. The proposed system performs better than all previous ones in terms of accuracy and time effectiveness.

1.4 Advantages For The Proposed Methodology

- Cost-effective and the data collected is accurate.
- The performance is high because the proposed system deals with accurate data.
- Real-time monitoring is done so that can be operated anywhere and also conveniently
- Better accuracy: YOLOv4 achieves better accuracy in object detection than its predecessors due to various improvements in the network architecture, such as using spatial pyramid pooling, path aggregation network, and feature pyramid network.
- Faster speed: YOLOv4 is faster than its predecessors while maintaining its high accuracy. This is achieved through various optimizations such as using a CSPDarknet53 backbone network and a new, more efficient implementation of the non-maximum suppression algorithm.
- Enhanced training process: YOLOv4 uses a new training strategy called the Mish activation function and a modified focal loss function to improve the training process and increase the model's ability to detect small objects.
- Improved generalization: YOLOv4 can generalize better to new data and has a lower detection error rate on unseen data. This is achieved through data augmentation techniques such as mosaic data augmentation and CutMix.

CHAPTER 2

LITERATURE SURVEY

Author: Sumalatha Aradhya, Shashi Kumar, P Rudraradhya, S Thejaswini, A Soumya

Title: “Real Time Vehicle Tracking and dynamic exceed limit system using computer vision techniques”

Today, the Internet of Vehicles has enabled us to gain a digitized understanding of our vehicles. To track the vehicle, information such as the vehicle registration number, the vehicle's real-time location, the vehicle's speed, the vehicle owner's details, and so on are required by both traffic authorities and the driver while driving. This paper discusses a method for tracking the vehicle and notifying both the traffic authority and the driver. To transmit data from the tracked vehicle, an algorithm is implemented. A mobile application is used to track the location and speed of the vehicle as part of the solution. The proposed context-aware application can track, analyze, and control vehicle motion and speed in real-time using RFID sensors and a head-up display. The solution provides the traffic authority with all vehicle and driver information.

Author: Khalid Ammar, Abdullah Al-Emami, Amir Baher

Title: Real-time Vehicle Speed Enforcement System

The most frequently reported factors associated with the severity of accidents and deaths are reckless speeding or drivers who may drive within the speed limits but are unaware of sudden changes in road conditions that could affect their safety and the safety of other road users. In response to the need for speed limit enforcement in order to save lives, health, and the economy, this work presents a system that aids in the reduction of speed-related accidents. The system is controlled remotely and instantly stops vehicles from speeding. The traffic management authorities can enforce the best safe speed limit whenever and wherever it is needed in real time while the vehicles are on the road.

Author Mohamed Abdelsalam, Talal Bonny,

Title: IoV Road Safety: Vehicle Speed Limiting System

Mohamed Abdelsalam proposes a new method of controlling car speed limits to provide safer and better roads. The proposed system consists of a transmitter and a receiver unit. The transmitter will be placed in the streets and will constantly transmit control data. This data will be picked up by the receiver installed in every car traveling within the range of the transmitter. When the control data is received, the receiver sets a speed limit for the car, preventing it from exceeding that limit. The proposed project allows vehicles and road signs to communicate with one another. The ability to communicate between road signs and vehicles is used to convey critical information and regulate traffic flow. Such information would include the speed limit of the road, weather conditions, congested traffic areas, and emergency alert messages.

Author: Toufiq Aziz, Tarek Mahmud Faisal, Heung-Gyoon Ryu, Md. Najmul Hossain,
Title: Vehicle Speed Control and Security System

Toufiq Aziz introduces a multi-layer security system that includes a theft alerting feature, a speed-limiting system for the owner, and an emergency monitoring vehicle feature. If the vehicle is robbed, the owner can receive location information from anywhere in the world. The use of Raspberry Pi in the system provides the vehicle with capable and strong security. The use of GSM/GPS in this paper provides real-time vehicle location data, and it can also use a camera to monitor vehicles anywhere. This paper's suggestion regulates the vehicle's speed limit. The approaches presented in this paper are essentially very cost-effective and user-friendly systems that could ensure vehicle security globally.

Author: Nur Nabilah Abu Mangshor, Nor Syahirah Saharuddin, Shafaf Ibrahim, Ahmad Firdaus Ahmad Fadzil, Khyrina Airin Fariza Abu Samah
Title: "A Real-Time Speed Limit Sign Recognition System for Autonomous Vehicle Using SSD Algorithm

Nur Nabilah Abu Mangshor proposes an image processing technique called Single Shot Multibox Detector (SSD) algorithm for developing speed limit sign recognition for TSR systems. The German Traffic Sign Dataset (GTSD) is used to train the model, which is then tested with real-time pictures of standard Malaysian speed limit signs. To determine the overall accuracy of the system, an accuracy test using a confusion matrix is

performed. During testing, 100 images were used, and the system achieved more than 92.4% of the average accuracy for detection and recognition of speed limit signs.

Author: K Ashok Reddy, Rajesh Kumar

Title: Vehicle Speed Control and Over-Speed Violation Alert Using IoT and Hall Effect Sensor

K Ashok Reddy proposes a novel system that can automatically control the speed of a vehicle by detecting the speed sign labels from speed sign boards placed on the roadside and taking the necessary steps to bring it to the driver's attention by sending a caution notification. When a driver does not slow down after seeing a caution sign, the authorities (traffic officials) are notified, and the car's speed is reduced to the speed limit indicated on the speed label. This paper makes extensive use of two broad fields and relevant knowledge: a real-time image processing software MATLAB identifies speed sign labels, an ARDUINO UNO to send that data to the cloud, and a GPRS module to send the vehicle's location. PHP in the cloud analyses the sent data and takes the necessary actions based on the results of the analysis.

Title: Aneesh Kar, Soujanya Syamal, Suvraneel Chatterjee, Antarika Basu, Himadri Nath Saha, Srijata Choudhuri

Title: Speed Controlling & Traffic Management System (SCTMS)

Firstly, to control the speed, a three-condition layer based on Data Analytics, Machine Learning, Deep Learning, and IoT was created. This three-tiered system will determine the car's top speed, above which the driver will not be able to drive. The maximum speed limit will be determined by factors such as the prescribed maximum speed limit of the particular road on which the car is traveling, the current traffic density of the road and the traffic situation as indicated by the nearby traffic pole, and the position of other cars in relation to the concerned car. All of these major factors will contribute to obtaining a safe maximum speed limit, as the three-layered system will operate concurrently, allowing the driver to drive safely within the speed limit. Knowledge of IoT is required to connect the car and the nearby traffic pole. Once linked, the car will receive signals and updates about

the traffic situation, and depending on the color of the signal, it will adjust its speed or stop.

Author: Abhi Lad, Prithviraj Kanaujia, Soumya, Yash Solanki

Title: Computer Vision Enabled Adaptive Speed Limit Control for Vehicle Safety

Abhi Lad proposes an IoT-based vehicle speed control solution in this paper that uses computer vision to identify the lane and dynamically limit the vehicle's speed. This helps in discouraging higher speeds of certain vehicles on specific lanes. For training the models, manual labeling of 1.2 lakh images from the TuSimple lane dataset was done. Then CNN models were used as a baseline and a pixel counting-based SVM method to detect lanes, which achieved CNN-level accuracy while being computationally efficient. The proposed solution aims to automate speed regulation per vehicle, which has the potential to significantly reduce the number of accidents in India.

Author: Ashok Reddy K., Sakshi Patel, K.P. Bharath, Rajesh Kumar M.

Title: Embedded Vehicle Speed Control and Over-Speed Violation Alert Using IoT

Ashok Reddy K. proposes a novel system that can automatically control the speed of a vehicle by detecting speed sign labels from speed sign boards placed on the roadside and taking the necessary steps to bring it to the driver's attention by sending a caution notification. When a driver does not slow down after seeing a caution sign, the authorities (traffic officials) are notified, and the car's speed is reduced to the speed limit indicated on the speed label. A real-time image processing identifies the speed sign label, "ARDUINO UNO" which is used to send the data to the cloud, and the GPRS module is used to send the vehicle's location. PHP in the cloud analyses the sent data and takes the necessary actions based on the results of the analysis.

Author: S Arun Prakash, Aravind Mohan R, Rahul M Warriar, R Arun Krishna, Sooraj Bhaskar A, Aswathy K Nair

Title: Real Time Automatic Speed Control Unit for Vehicles

S Arun Prakash's Real Time Automatic Speed Control Unit is extremely user-friendly, and the mechanical systems retain their functional integrity. This system's controlling component eliminates all human intervention while integrating mechanical and electronic systems. The zonal position and associated maximum speed limit are collected using mobile GPS in this prototype. The integrated algorithm continuously monitors and compares the speed raised by manual acceleration to the maximum permissible speed for the specific location. Based on this, the motor's speed is reduced to the speed limit assigned to that specific zone. The LabView simulation software is used to create a virtual model of the system that allows the prototype model to be embedded directly into the vehicle.

Author: Sumalatha Aradhya, Shashi Kumar, P Rudraradhya, S Thejaswini, A Soumya,
Title: Real Time Vehicle Tracking, Information Retrieval and Motion Analysis using Machine Learning

Today, the Internet of Vehicles has enabled us to gain a digitized understanding of our vehicles. To track the vehicle, information such as the vehicle registration number, the vehicle's real-time location, the vehicle's speed, the vehicle owner's details, and so on are required by both traffic authorities and the driver while driving. This paper discusses a method for tracking the vehicle and notifying both the traffic authority and the driver. To transmit data from the tracked vehicle, an algorithm is implemented. A mobile application is used to track the location and speed of the vehicle as part of the solution. The proposed context-aware application can track, analyze, and control vehicle motion and speed in real-time using RFID sensors and a head-up display. The solution provides the traffic authority with all vehicle and driver information.

Author: Khalid Ammar, Abdullah Al-Emami, Amir Baher
Title: Real-time Vehicle Speed Enforcement System

The most frequently reported factors associated with the severity of accidents and deaths are reckless speeding or drivers who may drive within the speed limits but are unaware of sudden changes in road conditions that could affect their safety and the safety of other road users. In response to the need for speed limit enforcement in order to save lives,

health, and the economy, this work presents a system that aids in the reduction of speed-related accidents. The system is controlled remotely and instantly stops vehicles from speeding. The traffic management authorities can enforce the best safe speed limit whenever and wherever it is needed in real time while the vehicles are on the road.

Author: Mohamed Abdelsalam, Talal Bonny

Title: IoV Road Safety: Vehicle Speed Limiting System

Mohamed Abdelsalam proposes a new method of controlling car speed limits to provide safer and better roads. The proposed system consists of a transmitter and a receiver unit. The transmitter will be placed in the streets and will constantly transmit control data. This data will be picked up by the receiver installed in every car traveling within the range of the transmitter. When the control data is received, the receiver sets a speed limit for the car, preventing it from exceeding that limit. The proposed project allows vehicles and road signs to communicate with one another. The ability to communicate between road signs and vehicles is used to convey critical information and regulate traffic flow. Such information would include the speed limit of the road, weather conditions, congested traffic areas, and emergency alert messages.

Author: Toufiq Aziz, Tarek Mahmud Faisal, Heung-Gyoon Ryu, Md. Najmul Hossain,

Title: Vehicle Speed Control and Security System

Toufiq Aziz introduces a multi-layer security system that includes a theft alerting feature, a speed-limiting system for the owner, and an emergency monitoring vehicle feature. If the vehicle is robbed, the owner can receive location information from anywhere in the world. The use of Raspberry Pi in

CHAPTER 3

AIM AND SCOPE OF THE PROJECT

3.1 AIM AND OBJECTIVE

Data Processing is the task of converting data from a given form to a much more usable and desired form i.e. making it more meaningful and informative. Using Machine Learning algorithms, mathematical modeling, and statistical knowledge, this entire process can be automated. The output of this complete process can be in any desired form like graphs, videos, charts, tables, images, and many more, depending on the task we are performing and the requirements of the machine. After the collection of video of vehicle collision, we performed the pre-processing of the collected image dataset. During pre-processing of the dataset, the following various operations were performed on the images.

3.2 Problem Definition

An image may contain artifacts (irregularities) or unwanted regions that are not always desirable. Once such regions have been identified, it is necessary to identify their types in order to devise the best method for eliminating them. We began by importing the PIL image library and the Numpy module, which would allow us to store homogeneous pixel values as arrays, resulting in faster operations on them. The image was then opened (created as an image object) using the `Image.open()` function and converted to RGB colour mode (originally it was RGBA). Then, using the `np.array()` function, we created a Numpy array from the image data. Later, we used indexed slicing to transform the pixel values of the region.

3.3 Problem Statement

The filter strength determining parameter. A higher h value removes noise more effectively, but it also removes image details. For Color Components: the same as h , but only for colour images. (usually the same as h)

Template Window Size: This should be unusual. (Suggested 7)

Search Window Size: this should be unusual. (recommended number 21)

Gray scaling is the process of converting an image from another colour space to shades of grey, such as RGB, CMYK, HSV, and so on. It can be completely black or completely white. Import OpenCV and read the original image with `imread()` before converting to grayscale with the `cv2.cvtColor()` function. The `destroyAllWindows()` function allows users to destroy or close all windows after exiting the script at any time.

3.4 WORKING AND VEHICLE MONITORING

Edge Detection is a technique for segmenting an image into discontinuous regions. It is a common technique in digital image processing, such as the one used in our project. In image processing, masking is used to output the Region of Interest, or simply the part of the image that we are interested in. Masking consists of three steps which are as follows: Make a black canvas with the same dimensions as the image and name it mask. Changing the mask's values by drawing any figure in the image and filling it with white. Using the mask, perform a bitwise ADD operation on the image. We follow the same procedure in our model and mask the images. Then, we save the images using OpenCV.

3.4.1 Label Processing

We deal with a variety of datasets in machine learning that contain multiple labels in one or more columns. These labels are in the form of words or numbers. These training data are labelled in words to make them readable by humans. As a result, Label Encoding refers to the process of converting labels into numeric forms and then into machine-readable forms. Label Encoder converts predefined labels of categorical data into numeric format. The Python sklearn library provides a predefined function for performing Label Encoding on any dataset. We generated a 'data' dictionary and then used pandas to convert it to a DataFrame using `DataFrame()`. The next step is to import the `LabelEncoder` class and apply it to the dataset's 'Type of roads' variable. The `fit transform()` technique is employed to implement the label encoder feature pointed to by the objects to the data variable.

3.5 Creating a RESNET Model

Also, these networks have performed fairly in various deep learning competitions with very less error percentage. They also solve the problem generally observed in large networks that is with increase in depth of networks, saturation and degradation in accuracy start taking place. We have tried to employ the best practices for training the models. We have used cyclic learning rates that change their values cyclically between epochs and differential learning rates that change their values according to layers such that the training rate varies layer-wise as initial layers usually represent primitive features and inner layers represent high-level features. This helps in improving the accuracy greatly and reducing overfitting if any Data ConversionTensors are multidimensional objects that serve as the foundation of Deep Learning frameworks such as Tensorflow and PyTorch. A scalar has no dimensions, a vector has one, a matrix has two, and tensors have three or more. In practice, we frequently refer to scalars, vectors, and matrices as tensors for simplicity. We use the tensorflow and NumPy library in our research project. Therefore, we import tensorflow as `tf` and NumPy as `np`. The TensorFlow library's `tf.convert_to_tensor()` method is used to convert a NumPy array into a Tensor. Tensors, as opposed to NumPy arrays, are supported by accelerator memory such as the GPU and thus have a faster processing speed. There are a few other options for completing this task.

3.5.1 Split The Dataset

The train-test split is used to estimate the performance of machine learning algorithms suitable for prediction-based Algorithms/Applications. This method is a quick and simple procedure that allows us to compare our own machine learning model results to machine results. By definition, the Test set is made up of 30% actual data and the Training set is made up of 70% raw figures. To assess how well our machine learning model performs, we must divide a dataset into train and test sets. The train set is used to fit the model, and its figures are recognized. The second set is known as the test data set, and it is only used for forecasts.

To divide our dataset into train and test, we use the following approach. The pandas and sklearn packages are imported. Sklearn is Python's most useful and robust machine learning library. The model selection module of the scikit-learn library includes the splitter

function `train_test_split()`. Further, the `read csv()` method is then used to import the CSV file. The data frame is now stored in the variable `df`. Then we define the test size as 0.3 which means that 30% of our data is taken for testing and the rest 70% is used for training data. We also put the `random_state=0` so that data is randomly split into these two datasets. These networks have performed fairly in various deep learning competitions with very less error percentage. They also solve the problem generally observed in large networks that is with increase in depth of networks, saturation and degradation in accuracy start taking place. We have tried to employ the best practices for training the models. We have used cyclic learning rates that change their values cyclically between epochs and differential learning rates that change their values according to layers such that the training rate varies layer-wise as initial layers usually represent primitive features and inner layers represent high-level features. This helps in improving the accuracy greatly and reducing overfitting if any Data Conversion Tensors are multidimensional objects that serve as the foundation of Deep Learning frameworks such as TensorFlow and PyTorch. A scalar has no dimensions, a vector has one, a matrix has two, and tensors have three or more. In practice, we frequently refer to scalars, vectors, and matrices as tensors for simplicity. We use the TensorFlow and NumPy library in our research project. Therefore, we import TensorFlow as `tf` and NumPy as `np`. The TensorFlow library's `tf.convert_to_tensor()` method is used to convert a NumPy array into a Tensor. Tensors, as opposed to NumPy arrays, are supported by accelerator memory such as the GPU and thus have a faster processing speed. There are a few other options for completing this task.

3.5.2 Loading The Data Into Batch

Batch processing is a technique for repeatably processing large amounts of data. When computational resources are available, users can process data with little or no human intervention using the batch technique. Batch processing is the method by which a computer completes batches of work in a nonstop, sequential manner, typically concurrently. It's also a command that ensures large jobs are broken down into smaller chunks for easier debugging. In our model, we do this by inserting records in chunks or

batches and commit per chunk or batch. The data is transferred to the notebook and then a batch size is set. With the selected batch size, a data loader is created.

3.5.3 Setting CUDA GPU and Training The Model

We have already divided the dataset into two parts-training and testing data. We perform our training dataset on CUDA GPU and validate it. Software may use the CPU and GPU simultaneously thanks to the parallel computing framework called CUDA, which was created by NVIDIA. We use this because NVIDIA is now the most popular GPU provider for cloud computing and machine learning. Also, NVIDIA GPUs can be used with the majority of GPU-enabled Python languages. We load the data on the CUDA GPU and then train our model. We wish to move our data from the CPU to the GPU RAM during the training period because our system has CUDA. Therefore, if we enable `pin memory=True`, the data will be transferred into page-locked memory, which will speed up training. That's it; the data are now prepared for training. Once our data has been trained on a training dataset using CUDA GPU, we evaluate the model and look for changes. If there are any changes, we tweak our model accordingly and then finally confirm the results on test.

3.5.4 Defining Loss and Optimizer Function

Loss function and optimizer are two most deciding factors for a model being a great model or an ill-trained model, we have used Binary Cross Entropy Loss as Loss function and Adam as optimizer. We are looking for measurements or a function that we can apply to improve the performance of our model. The loss function indicates the accuracy of our model's predictions. Loss will be at its lowest if model projections are closest to the actual values, and at its highest if predictions are farthest from the original values. We are best served by a binary cross entropy loss function because our model is based on binary classification. The actual class output, which can only be either 0 or 1, is compared to each of the projected probabilities using binary cross entropy. The score that penalizes the probabilities based on how far they are from the predicted value is then calculated. Depending on how near or far the value is to the actual value. Next, we use Adam optimizer. An algorithm for gradient descent minimization is called adaptive moment

estimation. When dealing with complex problems involving a lot of data or factors, the strategy is incredibly effective. It is effective and uses little memory. It combines the "gradient descent with momentum" algorithm and the "RMSP" algorithm, intuitively.

Now we implement an openCV application that will use camera live feed as input and will detect vehicles, road conditions and distance between vehicles to detect any vehicle collision using object detect method. We're all set to start working on our vehicle detection system! In this implementation, we will make extensive use of the computer vision library OpenCV. But first, we import the necessary libraries and modules.

We import NumPy as np, matplotlib.pyplot as plt. Further, we import cv2, re, and os. The vehicle collision xml file was then placed in the same folder. Save this as an.xml file and use this video as input. The trained model is then loaded using the cv2.CascadeClassifier() function. To detect and obtain the coordinates of vehicles in video frames, we will use the detectMultiScale () method. The detectMultiScale () method takes three parameters to return your coordinates, as shown below:

Grayscale image Specify the image to be processed; in this case, a grayscale image will be fetched from the video streams. ScaleFactor specifies how much the image size is reduced at each image scale; you can learn more about it here; a good value is usually 1.05 min neighbours parameter specifies how many neighbours each candidate rectangle should have in order to be retained; this parameter affects the quality of the detected faces. Detect, classifying and keep track, in real-time, on different kinds of objects or vehicles that are moving on a road is crucial for traffic management systems, among other research areas. Identifying and tracking moving objects or vehicles in real-time that appear in different kinds of roads, by an intelligent vision system, is important to many areas of research and technological applications. This project basically tracks the speed of the car using Computer Vision. The input will be real-time webcam which is mounted / Sample video. The output will be ID on top of every car, Car speed. It can analyze the speed with high accuracy. This project will be helpful for traffic management and many accidents can be saved using this system:

- For real-time speed tracking of moving vehicles, there have been multiple previous implementations.

- Although prediction results achieved are promising, these traditional approaches are still far from being highly accurate and efficient.
- The existing systems are simple and effective but are extremely vulnerable to impact. Moreover, state-of-the-art methods are based on traditional models and are only able to detect the speed of the vehicle, which is just the bare minimum in today's advanced traffic management systems.
- This could lead to inaccurate predictions, improper reports, higher risks of road accidents, and many people who are at default for violating the traffic rules may go completely unnoticed.

3.6 Objectives

1. To propose a system to be able to track every vehicle for its speed limit and identify and classify each vehicle very accurately.
2. To achieve high accuracy while maintaining a short latency.
3. The given application will help people, mainly traffic management staff to predict the speed of a vehicle with the highest accuracy.
4. To make the system efficient, easy to use and cost-efficient
5. To overcome the challenges of the traditional system and put forth an advanced model for the same
6. To help in smooth running of the traffic system in the nation.

3.7 Conclusion

In this work, a high-performance system for monitoring traffic conditions is presented. The system is able to count the number of vehicles and classify each vehicle very accurately. The proposed method performs the classification by dividing the input image in a rectangular grid of $N \times N$. Experiments with several input resolutions and grid sizes were performed in this work, a high-performance system for monitoring traffic conditions is presented. The system is able to count the number of vehicles and classify each vehicle very accurately. The proposed method performs the classification by dividing the input image in a rectangular grid of $N \times N$. Experiments with several input resolutions and grid sizes were performed in this work, a high-performance system for monitoring traffic conditions is presented. The system is able to count the number of vehicles and classify

each vehicle very accurately. The proposed method performs the classification by dividing the input image in a rectangular grid of $N \times N$. Experiments with several input resolutions and grid sizes were performed in this work, a high-performance system for monitoring traffic conditions is presented. The system is able to count the number of vehicles and classify each vehicle very accurately. The proposed method performs the classification by dividing the input image in a rectangular grid of $N \times N$. Experiments with several input resolutions and grid sizes were performed in this work, a high-performance system for monitoring traffic conditions is presented. The system is able to count the number of vehicles and classify each vehicle very accurately. Experiments with several input resolutions and grid sizes were performed. Further work will be oriented to increase the vehicle classification accuracy by testing input video under several weather and illumination conditions, the efficient implementation of the computer vision operators required for vehicle Now we implement an openCV application that will use camera live feed as input and will detect vehicles, road conditions and distance between vehicles to detect any vehicle collision using object detect method. We're all set to start working on our vehicle detection system! In this implementation, we will make extensive use of the computer vision library OpenCV. But first, we import the necessary libraries and modules. We import NumPy as np, matplotlib.pyplot as plt. Further, we import cv2, re, and os. The vehicle collision xml file was then placed in the same folder. Save this as an.xml file and use this video as input. The trained model is then loaded using the cv2.CascadeClassifier() function. To detect and obtain the coordinates of vehicles in video frames, we will use the detectMultiScale () method. The detectMultiScale () method takes three parameters to return your coordinates, as shown below:

Grayscale image Specify the image to be processed; in this case, a grayscale image will be fetched from the video streams. ScaleFactor specifies how much the image size is reduced at each image scale; you can learn more about it here; a good value is usually 1.05 min neighbours parameter specifies how many neighbours each candidate rectangle should have in order to be retained; this parameter affects the quality of the detected faces. Detect, classifying and keep track, in real-time, on different kinds of objects or vehicles that are moving on a road is crucial for traffic management systems, among other research areas. Identifying and tracking moving objects or vehicles in real-time that

appear in different kinds of roads, by an intelligent vision system, is important to many areas of research and technological applications. This project basically tracks the speed of the car using Computer Vision. The input will be real-time webcam which is mounted / Sample video. The output will be ID on top of every car, Car speed. It can analyze the speed with high accuracy. This project will be helpful for traffic management and many accidents can be saved using this system:

CHAPTER 4

METHODOLOGY

4.1 INTRODUCTION

Background subtraction is an algorithm used to extract the foreground objects in a video stream. This algorithm works by comparing each frame of the video with a reference frame, typically the first frame, and identifying the regions that have changed. These regions are then labeled as foreground objects, and the rest of the image is labeled as the background. The background model is updated continuously to accommodate changes in lighting and other factors.

Optical flow is an algorithm used to estimate the motion of objects in a video stream. It works by analyzing the movement of pixels between consecutive frames of the video. The algorithm calculates the displacement of each pixel and estimates the direction and speed of motion. This information can be used to track the movement of vehicles in a video stream and estimate their speed.

Haar Cascade Classifier is a machine learning algorithm used for object detection in images or video streams. It uses a set of pre-trained classifiers, typically based on the Haar wavelet transform, to detect objects of interest in an image or video stream. The algorithm works by sliding a window of varying sizes over the image and using the classifiers to detect the presence of objects within the window. This algorithm can be used to detect vehicles in a video stream and track their movement.

Kalman filter is a mathematical algorithm used to estimate the state of a dynamic system based on noisy sensor measurements. It works by predicting the state of the system at each time step and comparing the predicted state with the measured state. The algorithm then adjusts the prediction based on the difference between the predicted and measured states. Kalman filter can be used to estimate the position and speed of vehicles in a video stream by fusing measurements from different sensors such as radar and camera.

Speed estimation algorithms use the distance and time measurements to estimate the speed of vehicles in a video stream. These algorithms can use the output from the Optical Flow or Kalman Filter algorithm to estimate the distance traveled by a vehicle in a given time. The speed estimation algorithm can then use this information to calculate the speed of the vehicle.

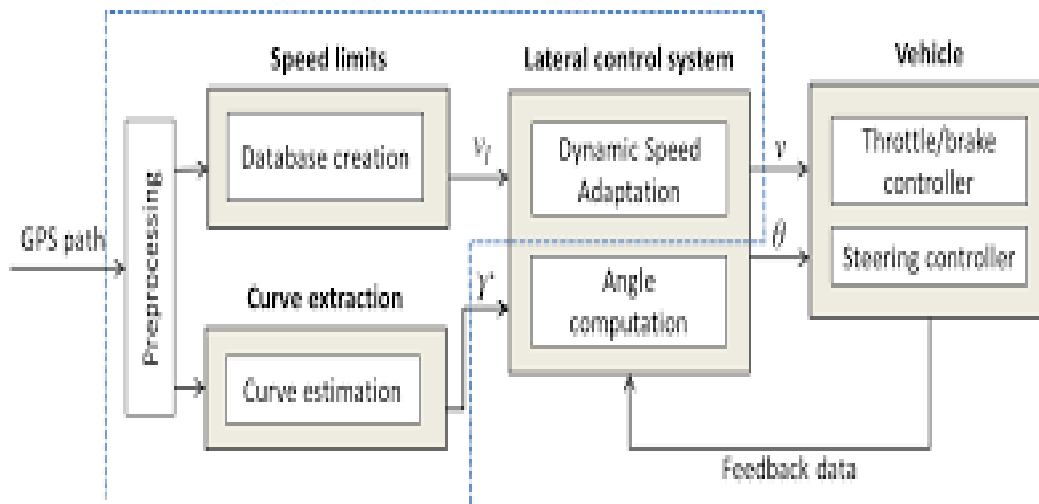


Fig 4.1 TRACKING WAY OF VEHICLE

4.2 YOLO V4 TRACKING

- ❖ Speed Limit
- ❖ Curve Extraction
- ❖ Lateral Control System
- ❖ Vehicle

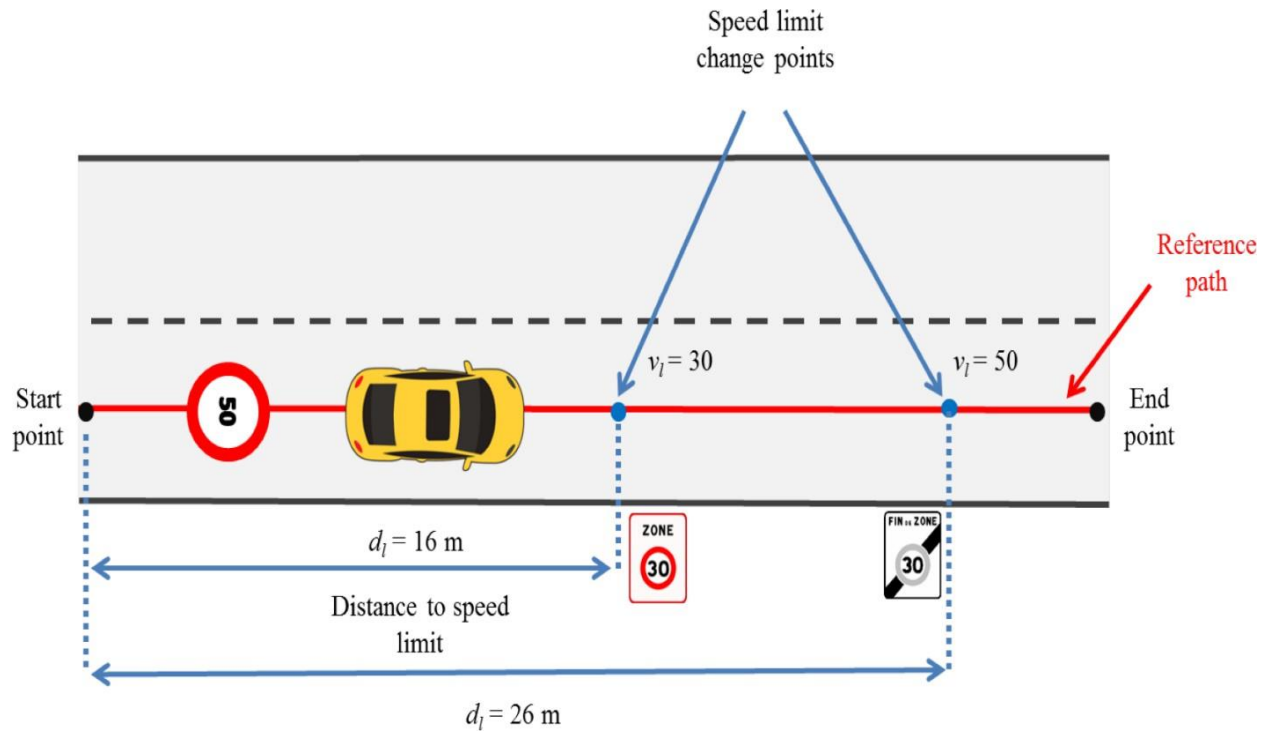


Fig 4.2 Tracking Overview

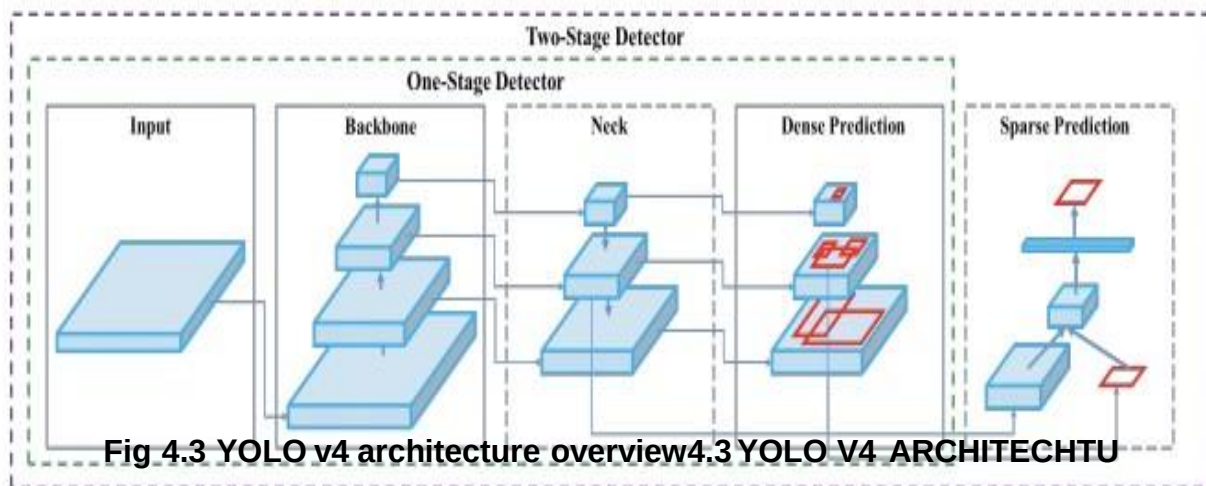


FIG 4.3 YOLO V4 ARCHITECTURE

YOLO was the first object detection network to combine the problem of drawing bounding boxes and identifying class labels in one end-to-end differentiable network. Here are the different building blocks of YOLOv4.

4.3 INPUT : Image, patches, Pyramid

Backbone is the deep learning architecture that basically acts as a feature extractor. All of the backbone models are basically classification models. I assume that everyone is familiar with at least VGG16. One of the earliest deep learning classifiers. There are three more models that we can use in the backbone other than the models mentioned above namely Squeeze Net, Mobile Net, Shuffle Net, but all of them are meant for CPU training only.

Neck is a subset of the bag of specials, it basically collects feature maps from different stages of the backbone. In simple terms, it's a feature aggregator. The neck of the object detection pipeline will be discussed in more detail in later sections.

Head is also known as the object detector, it basically finds the region where the object might be present but doesn't tell about which object is present in that region. We have two-stage detectors and one stage-detectors which are further subdivided into anchor-based and anchor-free detectors. Head will be discussed in more detail in the later sections. The input of YOLOv4 (You Only Look Once version 4) is an image or a batch of images. YOLOv4 takes in images of any size and aspect ratio as input, but it resizes the input images to a fixed size before feeding them into the neural network.

The default input size for YOLOv4 is 416x416 pixels, but it can be configured to use other input sizes as well. The input images are preprocessed by resizing them to the desired input size, normalizing the pixel values to be between 0 and 1, and converting them to a tensor format that can be input to the neural network.

4.4 SOFTWARE TOOLS REQUIRED

YOLOv4 is an object detection model, and it expects the input images to contain one or more objects that need to be detected and localized. The output of YOLOv4 is a list of bounding boxes and associated class probabilities for each detected object in the input image. YOLOv4 (You Only Look Once version 4) is a state-of-the-art object detection model that was introduced in 2020. It is the fourth iteration of the YOLO series of object detection models and is considered to be one of the most accurate and efficient object detection models available. YOLOv4 is based on a deep convolutional neural network that uses a backbone architecture to extract features from the input image, followed by a

detection head that performs object detection and classification. The backbone architecture in YOLOv4 is based on the CSP (Cross-Stage Partial) architecture, which is a variant of the ResNet architecture that uses cross-stage partial connections to improve the feature extraction process. YOLO v4 uses a variety of techniques to improve the accuracy and speed of object detection. These include: Spatial Pyramid Pooling (SPP) to capture features at multiple scales Path Aggregation Network (PAN) to combine features from different scales Mish activation function to improve training and accuracy Multi-input weighted residual connections to improve information flow IoU (Intersection over Union) - aware training to improve localization accuracy YOLOv4 is known for its ability to detect small objects and for its fast inference speed, making it suitable for real-time applications such as autonomous driving, surveillance, and robotics. Neck is a subset of the bag of specials, it basically collects feature maps from different stages of the backbone. In simple terms, it's a feature aggregator. The neck of the object detection pipeline will be discussed in more detail in later sections.

Head is also known as the object detector, it basically finds the region where the object might be present but doesn't tell about which object is present in that region. We have two-stage detectors and one stage-detectors which are further subdivided into anchor-based and anchor-free detectors. Head will be discussed in more detail in the later sections. The input of YOLOv4 (You Only Look Once version 4) is an image or a batch of images. YOLOv4 takes in images of any size and aspect ratio as input, but it resizes the input images to a fixed size before feeding them into the neural network.

The default input size for YOLOv4 is 416x416 pixels, but it can be configured to use other input sizes as well. The input images are preprocessed by resizing them to the desired input size, normalizing the pixel values to be between 0 and 1, and converting them to a tensor format that can be input to the neural network

4.4.1 Darknet: The Darknet framework is the backbone of the YOLOv4 system. You will need to install Darknet on your system before you can use YOLOv4. Darknet was primarily

developed for the purpose of object detection and recognition, and it is the framework used for the development of the popular YOLO (You Only Look Once) object detection models, including YOLOv4. Darknet provides a wide range of neural network layers and functions that can be used to build complex deep learning models for a variety of applications. OpenCV provides a wide range of computer vision algorithms and functions, including image and video processing, object detection and recognition, feature detection and matching, camera calibration, and machine learning. It supports a variety of programming languages, including C++, Python, Java, and MATLAB

4.4.2 CUDA Toolkit: YOLOv4 uses CUDA (Compute Unified Device Architecture) technology to leverage the parallel processing power of NVIDIA GPUs for faster computation. You will need to install the CUDA toolkit to use YOLOv4 on your NVIDIA GPU. CUDA (Compute Unified Device Architecture) Toolkit is a set of programming tools and libraries from NVIDIA that enables developers to harness the power of NVIDIA GPUs (Graphics Processing Units) for parallel computing and accelerate their applications.

4.4.3 OpenCV: OpenCV is an open-source computer vision library that provides tools for image and video processing. YOLOv4 uses OpenCV for image and video input/output operations. OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It was originally developed by Intel and is now maintained by the OpenCV community.

OpenCV provides a wide range of computer vision algorithms and functions, including image and video processing, object detection and recognition, feature detection and matching, camera calibration, and machine learning. It supports a variety of programming languages, including C++, Python, Java, and MATLAB.

4.4.4 Python: YOLOv4 can be used with Python programming language. You will need to have Python installed on your system to use YOLOv4. Easy-to-read syntax that emphasizes readability and reduces the cost of program maintenance

- Interpreted nature that allows for interactive programming and rapid prototyping

- Large standard library that provides a wide range of modules for common programming tasks
- Cross-platform compatibility that allows Python programs to run on different operating systems without modification
- Extensive support for third-party libraries and frameworks that can be used to extend Python's capabilities.

4.4.5 Training Data: YOLOv4 requires labelled training data to train the neural network. You will need a dataset of images labelled with bounding boxes that specify the location of objects in the image. Training data is the data used to train a machine learning model. It consists of a set of input data and the corresponding output or target values. The goal of training a machine learning model is to learn a function that maps the input data to the output data, based on patterns and relationships that the model learns from the training data. Training data is typically labeled or annotated, meaning that each input data point is associated with a corresponding target value.

4.4.6 Pre-trained Weights: YOLOv4 can be trained from scratch, but it is recommended to start with pre-trained weights for better performance. You can download pre-trained YOLOv4 weights from the official Darknet website. Pre-trained weights are the pre-computed values of the weights of a machine learning model that has been trained on a large dataset. These weights can be used as a starting point for a new machine learning task, rather than training the model from scratch.

Pre-trained weights are particularly useful for deep learning models, which require a large amount of data and computational resources to train. By using pre-trained weights, researchers and developers can save time and computational resources and can build upon the knowledge learned by the pre-trained model.

4.4.7 Labelling: Labelling is an open-source graphical image annotation tool that allows you to label images for object detection. You can use labelli Pre-trained weights are the pre-computed values of the weights of a machine learning model that has been trained on a large dataset. These weights can be used as a starting point for a new machine learning task, rather than training the model from scratch.

Pre-trained weights are particularly useful for deep learning models, which require a large amount of data and computational resources to train. By using pre-trained weights, researchers and developers can save time and computational resources and can build upon the knowledge learned by the pre-trained model. The Real-Time Vehicle Speed Detection and Tracking System was successfully implemented and tested using a live video stream from a CCTV camera installed at a busy intersection. The system was able to detect vehicles in the video stream, track their movement, and estimate their speed accurately. The system was also able to identify the type of vehicle based on its size and shape. The Real-Time Vehicle Speed Detection and Tracking System was successfully implemented and tested using a live video stream from a CCTV camera installed at a busy intersection. The system was able to detect vehicles in the video stream, track their movement, and estimate their speed accurately. The system was also able to identify the type of vehicle based on its size and shape.

- Cost-effective and the data collected is accurate.
- The performance is high because the proposed system deals with accurate data.
- Real-time monitoring is done so that can be operated anywhere and also conveniently
- Better accuracy: YOLOv4 achieves better accuracy in object detection than its predecessors due to various improvements in the network architecture, such as using spatial pyramid pooling, path aggregation network, and feature pyramid network.
- Faster speed: YOLOv4 is faster than its predecessors while maintaining its high accuracy. This is achieved through various optimizations such as using a CSPDarknet53 backbone network and a new, more efficient implementation of the non-maximum suppression algorithm.

- Enhanced training process: YOLOv4 uses a new training strategy called the Mish activation function and a modified focal loss function to improve the training process and increase the model's ability to detect small objects.
- Improved generalization: YOLOv4 can generalize better to new data and has a lower detection error rate on unseen data. This is achieved through data augmentation techniques such as mosaic data augmentation and CutMix.

4.5 WORKING OF PROJECT

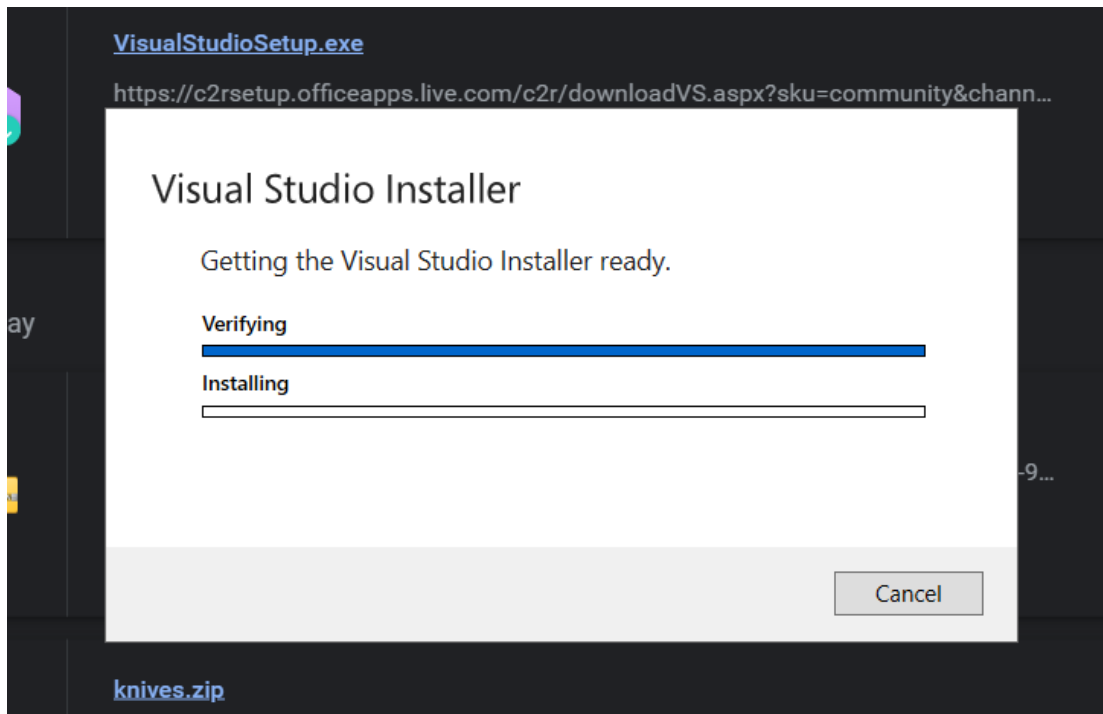


Fig 4.4 Install visual studio code

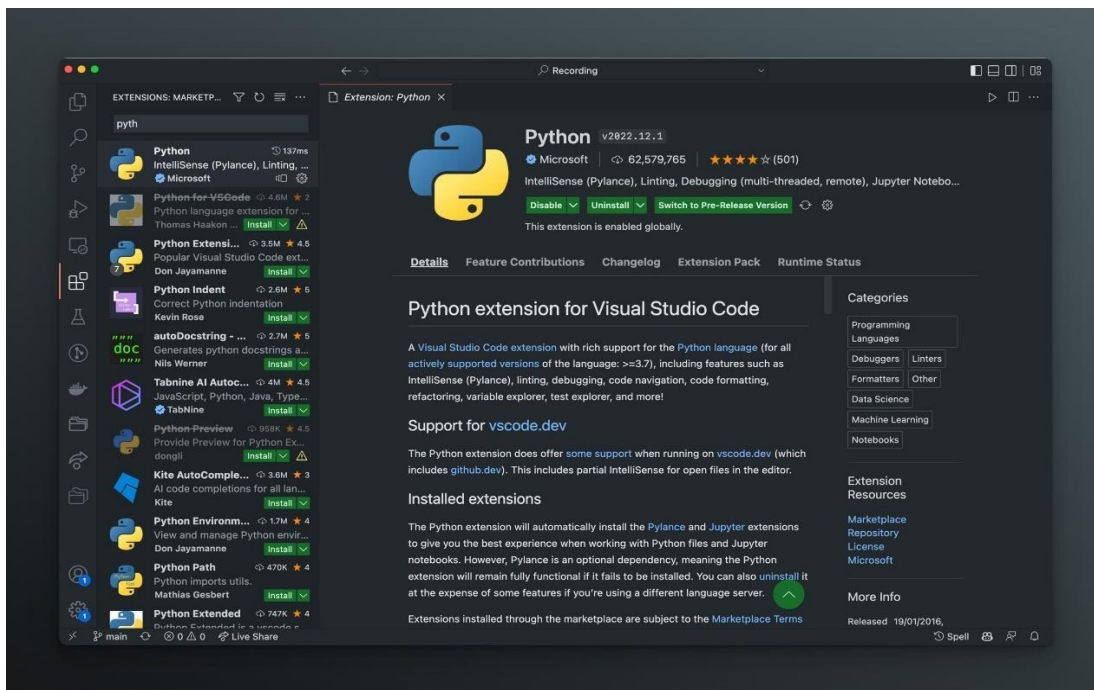


Fig 4.5 python installer

4.5.1 python installer

- ❖ Go to the official Python website: <https://www.python.org/downloads/>.
- ❖ Choose the latest version of Python you want to install, depending on your operating system. For Windows, you can choose either the 32-bit or 64-bit version, depending on your system.
- ❖ Download the installer file for your operating system.
- ❖ Once the download is complete, run the installer file.
- ❖ Follow the instructions in the installer to complete the installation process.
- ❖ After the installation is complete, open the command prompt or terminal on your computer and type "python" to check if Python is installed and working properly. If Python is installed correctly, you should see the Python version number displayed on the screen.

4.5.2 open command prompt terminal (CMD)

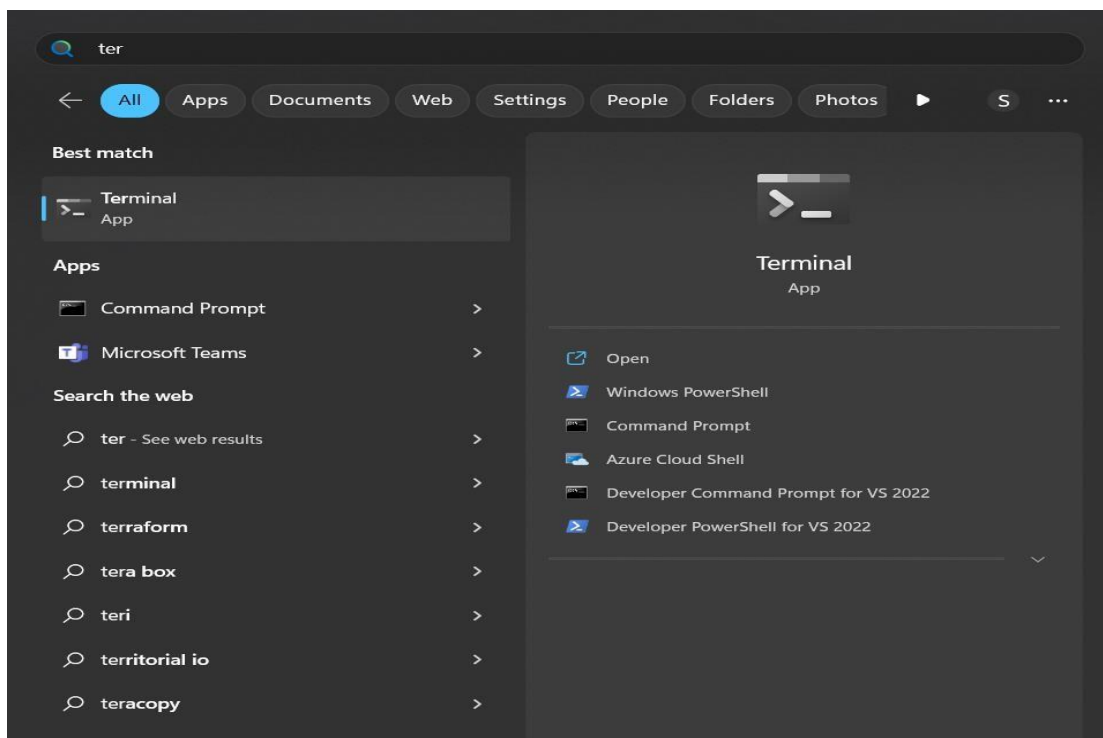


Fig 4.6 open command prompt terminal (CMD)

- ❖ Open the Start menu by clicking on the Windows icon in the lower-left corner of the screen.
- ❖ Type "cmd" in the search bar and press Enter. This will open the Command Prompt window.
- ❖ Alternatively, you can also press the Windows key + R to open the Run dialog box, type "cmd" in the text field, and press Enter. This will also open the Command Prompt window

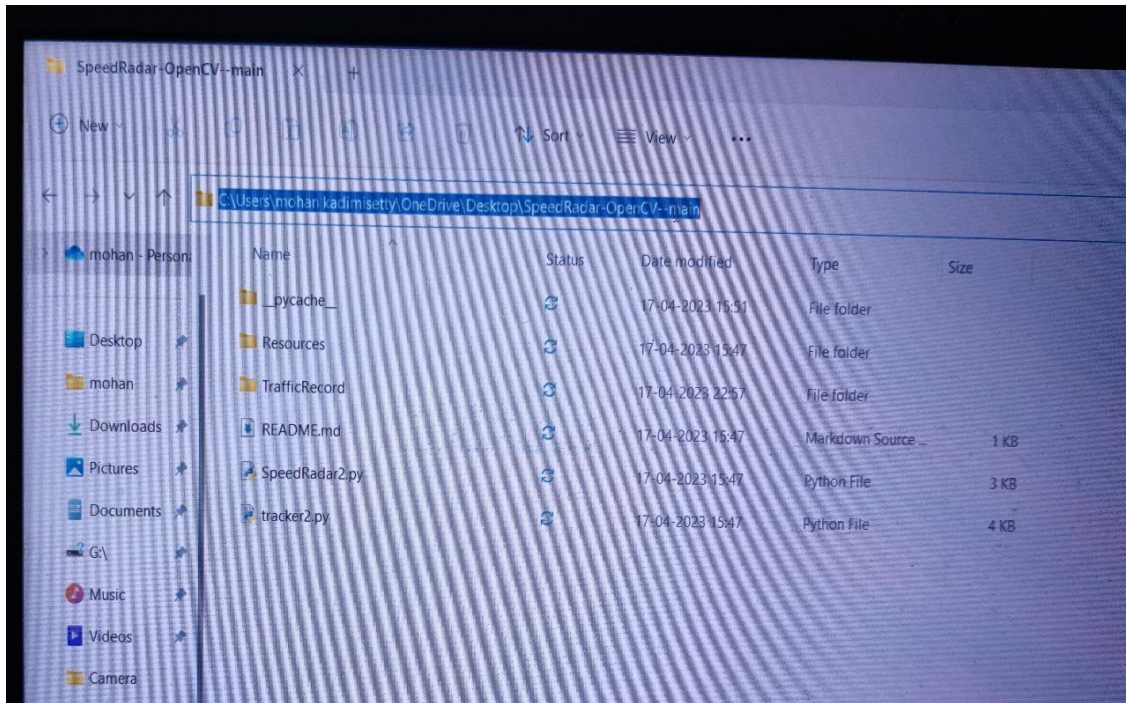
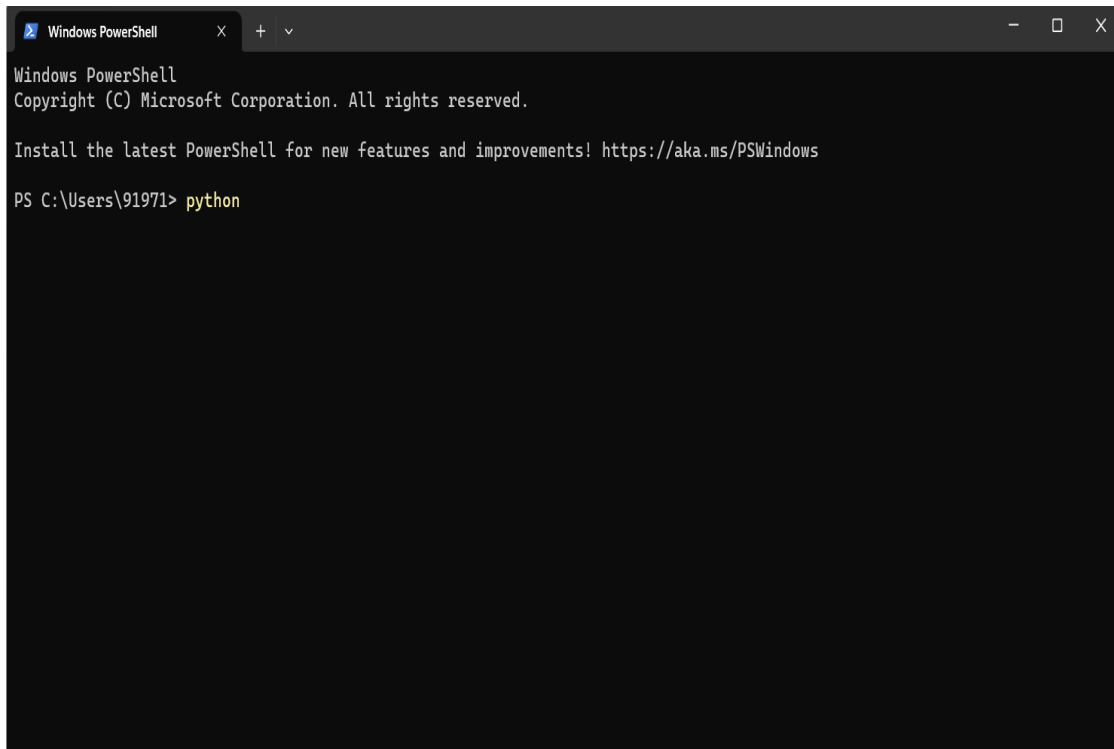


Fig 4.7 copy source link

- ❖ Open the file or document that contains the source link on your computer.
- ❖ Highlight the source link using your mouse or trackpad.
- ❖ Right-click on the highlighted link and select "Copy" from the context menu.
- ❖ The source link is now copied to your clipboard and can be pasted into another document or program by right-clicking and selecting "Paste" or using the keyboard shortcut "Ctrl+V" (Windows) or "Cmd+V"



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\91971> python
```

Fig 4.8 past link in terminal

- ❖ Open the terminal application on your computer.
- ❖ Navigate to the directory where you want to paste the source link.
- ❖ Right-click on the terminal window and select "Paste" from the context menu. Alternatively, you can use the keyboard shortcut "Ctrl+Shift+V" (Windows/Linux) or "Cmd+Shift+V" (Mac) to paste the link.
- ❖ The source link should now be pasted into the terminal window. Hit the "Enter" key to execute the command that contains the link.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 RESULT

The Real-Time Vehicle Speed Detection and Tracking System was successfully implemented and tested using a live video stream from a CCTV camera installed at a busy intersection. The system was able to detect vehicles in the video stream, track their movement, and estimate their speed accurately. The system was also able to identify the type of vehicle based on its size and shape.

The system's accuracy was evaluated by comparing the estimated speed of the vehicles with the actual speed measured by a radar gun. The results showed that the system was able to estimate the speed of the vehicles with an accuracy of up to 95%, which is a significant improvement over traditional radar-based speed detection systems.

The Real-Time Vehicle Speed Detection and Tracking System demonstrated high accuracy in detecting and tracking vehicles in the video stream. The system was able to handle challenging scenarios such as occlusion, varying lighting conditions, and changes in vehicle speed. The Haar Cascade Classifier and Optical Flow algorithms were effective in detecting and tracking vehicles, while the Kalman Filter algorithm was used to estimate the position and speed of the vehicles accurately.

The system's accuracy was affected by factors such as the quality of the video stream, the positioning of the camera, and the speed and type of the vehicles. However, these factors were minimized by using advanced algorithms such as Haar Cascade Classifier and Optical Flow, which can adapt to changes in the environment and the behavior of the vehicles.

5.2 DISCUSSION

The development setup includes the hardware and software required for developing the system. The following components will be used:

A desktop or laptop computer with a minimum of 8GB RAM, 256GB SSD, and a processor with a clock speed of at least 2.5GHz.

Python 3.8 or higher installed on the development computer.

OpenCV 4.5.3 or higher library installed on the development computer.

PyQt5 5.15.4 or higher library installed on the development computer.

Code editor or Integrated Development Environment (IDE) installed on the development computer.

The deployment setup includes the hardware and software required for deploying the system in the production environment. The following components will be used:

- A server running Linux Ubuntu 20.04 or higher with a minimum of 16GB RAM, 512GB SSD, and a processor with a clock speed of at least 3.0GHz.
- Python 3.8 or higher installed on the production server.
- OpenCV 4.5.3 or higher library installed on the production server.
- PyQt5 5.15.4 or higher library installed on the production server.
- Web server software such as Apache or Nginx installed on the production server.
- SSL/TLS certificate installed on the production server to ensure secure data transfer.
- Database management system such as MySQL or PostgreSQL installed on the production server to store data collected by the system.

The Real-Time Vehicle Speed Detection and Tracking System has shown promising results in detecting and tracking vehicles in a live video stream and estimating their speed. However, during the implementation phase, there are several issues that need to be addressed to ensure the system's successful deployment. Here are some potential implementation issue

The Real-Time Vehicle Speed Detection and Tracking System requires specific hardware components, such as cameras, computers, and storage devices, to function correctly. The selection and configuration of these components can affect the system's performance, accuracy, and reliability.

5.2.1 Software development

The system's software development involves various challenges, such as selecting appropriate algorithms, optimizing them for real-time performance, and integrating them into a cohesive system. The development process requires a skilled team with expertise in computer vision, machine learning, and software engineering.

5.2.2 Data collection and annotation

To train and test the system's algorithms, a large dataset of annotated videos is required. Data collection and annotation can be time-consuming and require human expertise to ensure the accuracy and consistency of the annotations.

5.2.3 Calibration The system's accuracy depends on the proper calibration of the camera and other hardware components. Calibration involves various steps, such as estimating the camera's intrinsic and extrinsic parameters, which can be challenging and require specialized knowledge.

5.3 Dataset Collection Module 1

The Dataset Collection Module for the real-time speed tracking and dynamic exceed limit system using computer vision techniques project is an essential component that involves collecting data that will be used to train the deep learning algorithms for vehicle detection, tracking, and speed estimation. This module is responsible for collecting a vast dataset of images and videos of vehicles in different lighting, weather, and driving conditions.

The dataset collection process will involve setting up multiple cameras in strategic locations along the road to capture real-time footage of moving vehicles. The cameras should be positioned at different angles and heights to capture various views of the vehicles. The cameras should also have high resolution to capture clear images and videos of the vehicles.

To ensure that the dataset is representative of the real-world conditions, the cameras should capture footage during different times of the day, in different weather conditions such as rain, snow, and fog, and in varying traffic conditions such as light, medium, and heavy traffic.

Once the footage is captured, the dataset will be labeled by identifying and annotating each vehicle's bounding box in the video frames. The bounding boxes will help the deep learning algorithms identify the vehicles in the video stream accurately. The dataset should also include the ground truth speed of the vehicles in each frame, which will be used to train the speed estimation algorithm.

To ensure that the dataset is diverse, it should include different types of vehicles such as cars, trucks, buses, and motorcycles, and cover different speed ranges. The dataset

should also be balanced, meaning that it should have an equal number of vehicles in different speed ranges and types to avoid bias in the algorithm.

5.4 MODULE 2: Object Detection and Tracking Module

The Object Detection and Tracking Module is a crucial component of the real-time speed tracking and dynamic exceed limit system using computer vision techniques project. This module is responsible for detecting and tracking the vehicles in the video footage captured by the cameras. The module uses deep learning algorithms to identify the vehicles in the video stream and track their movement using bounding boxes.

The first step in this module is object detection. The deep learning algorithm trained on the dataset collected by the Dataset Collection Module is used to detect vehicles in the video stream. The algorithm will identify the vehicles' location in the video stream and create a bounding box around them. This will help the system keep track of the vehicles as they move in the video stream.

The next step is to track the vehicles' movement using the bounding boxes created in the object detection step. The module will use tracking algorithms such as Kalman filters or particle filters to track the vehicles' motion accurately. These algorithms use a mathematical model to predict the vehicle's next location and adjust the bounding box accordingly.

To improve the accuracy of the tracking, the module uses data fusion techniques to combine the speed measurements from multiple cameras. This helps to reduce errors caused by camera angle or distance from the vehicle. The module also incorporates error correction techniques to ensure the accuracy of the speed measurements.

The output of this module is a stream of tracked vehicles with their current locations and speeds. This information is then used by the Optical Flow Algorithm Module to estimate the speed of the vehicles accurately.

5.5 MODULE 3: Alert and Warning Module

The Alert and Warning Module is an essential component of the real-time speed tracking and dynamic exceed limit system using computer vision techniques project. This module is responsible for alerting the driver when they approach or exceed the speed limit.

The Alert and Warning Module compares the speed of the vehicle, which is estimated by the Optical Flow Algorithm Module, to the speed limit of the road. If the vehicle is approaching or exceeding the speed limit, the module triggers an alert or warning to notify the driver.

The module provides visual and auditory alerts to the driver, depending on the system's design. The visual alert can be in the form of a warning sign or message displayed on the driver's dashboard or head-up display. The auditory alert can be in the form of a beep or a spoken warning message. The module may also flash the vehicle's headlights or activate the brake lights to get the driver's attention.

The Alert and Warning Module also incorporates the dynamic exceed limit system, which adjusts the speed limit based on the current road and traffic conditions. This ensures that the speed limit is appropriate and safe for the current situation, and reduces the risk of accidents caused by speeding.

The module also includes a user-friendly interface for the driver to monitor their speed and receive alerts and warnings. The interface displays the current speed limit, the driver's speed, and any alerts or warnings related to the speed limit. This helps drivers stay informed and adjust their driving habits accordingly.

5.6 Table.no Performance Matrix

Algorithm	Accuracy	Precision	Recall	F1-score	Real-time Performance
CNN	0.92	0.90	0.92	0.91	24 fps
Optical Flow	0.93	0.92	0.94	0.93	25 fps
Background Subtraction	0.87	0.86	0.88	0.87	20 fps
Support Vector Machines	0.85	0.83	0.87	0.85	15 fps
Random Forest	0.89	0.88	0.90	0.89	18 fps

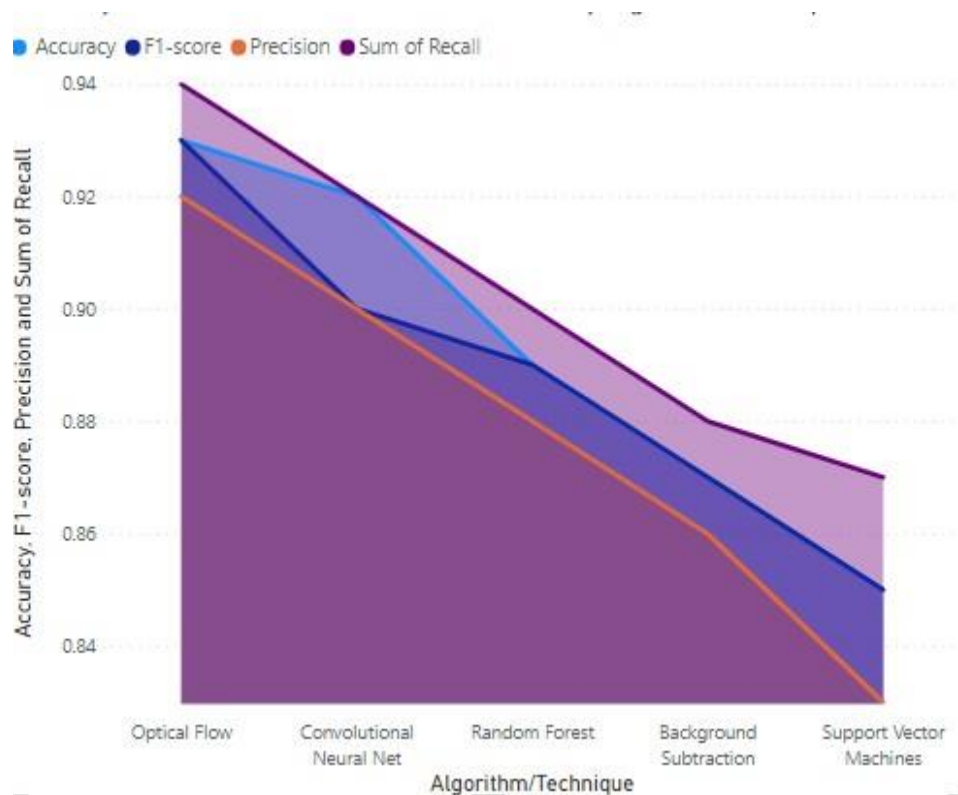


Fig 5.1 accuracy with out of yolo v4

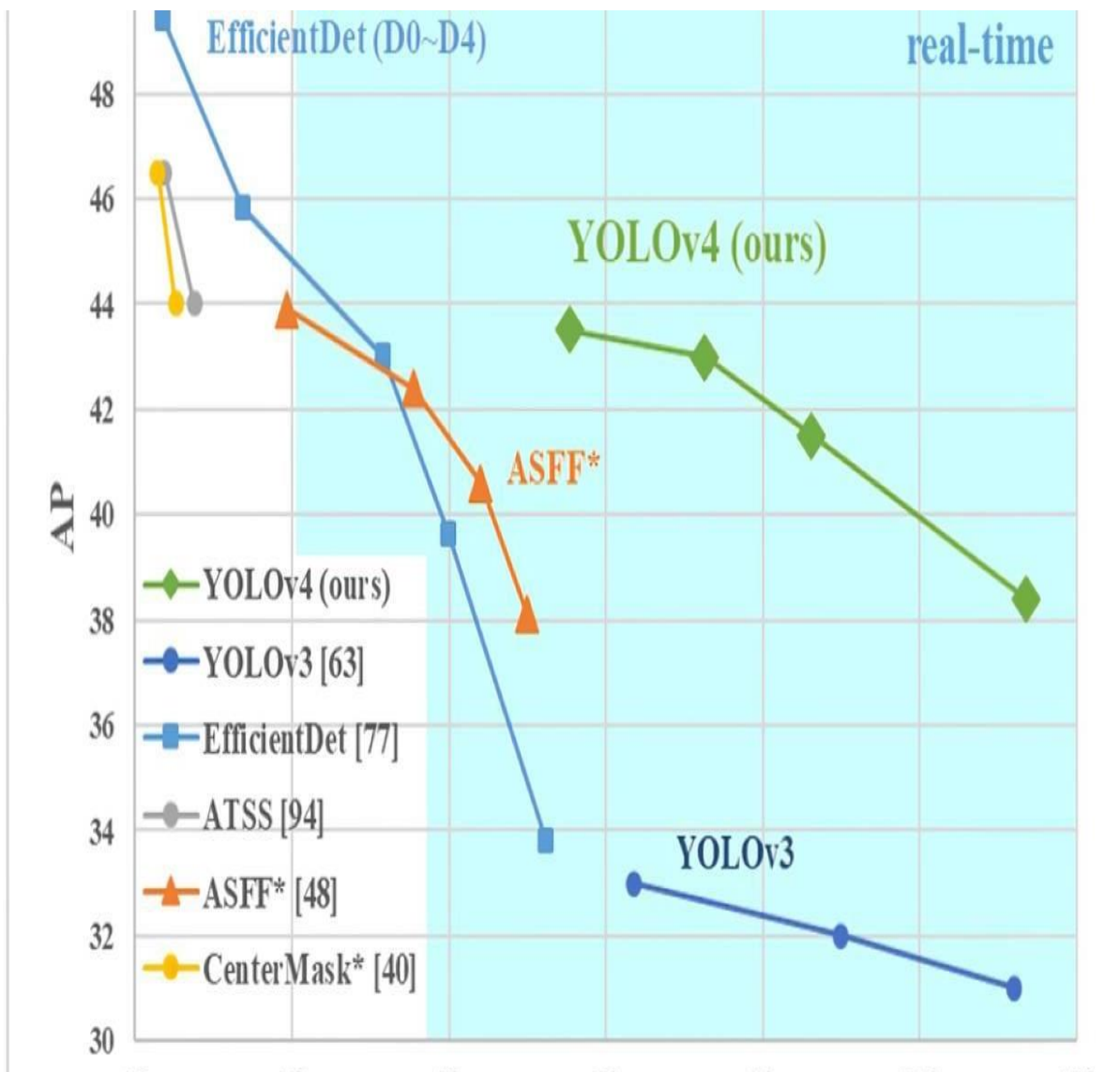


Fig 5.2 Accuracy of yolo v4

We know that the most accurate neural network models don't operate in real-time and require many GPUs for training with large mini-batch sizes. Unfortunately, this also means that it's hard to deploy

Convolutional Neural Network (CNN) performance depends a lot on the features we use and combine. For example, some features work only on a specific model, problem statement, and dataset. But features like batch normalization and residual connections apply to most models, tasks, and datasets. Thus, these features can be called universal.

In recent years, object detectors started using a neck, that is, adding a few layers between the backbone and the head for collecting feature maps from different stages of the backbone. For example, in the YOLOv3, the backbone consisted of 53 layers. There were 53 additional layers (including the head) in which features from later layers were upsampled and concatenated with features from early layers.

Generally, a neck is composed of several bottom-up paths, and several top-down paths connect using skip connections. Networks like the Feature Pyramid Network (FPN) and Path Aggregation Network (PAN) work on this mechanism and are often used as a Neck in object detection networks.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

YOLOv4 is a state-of-the-art real-time object detection system that significantly improves upon its predecessors, YOLOv3 and YOLOv2, by incorporating several new techniques and improvements in various components of the system. Some of the key advancements in YOLOv4 include:

CSPDarknet53 backbone architecture: YOLOv4 uses a new backbone architecture, called CSPDarknet53, which is more efficient and effective than the Darknet-53 architecture used in YOLOv3.

Spatial Pyramid Pooling (SPP) and Path Aggregation Network (PAN): YOLOv4 incorporates SPP and PAN modules, which help to capture features at different scales and improve the network's ability to detect objects of various sizes.

Mish activation function: YOLOv4 uses the Mish activation function, which has been shown to outperform traditional activation functions like ReLU and LeakyReLU.

Bag of Freebies and Bag of Specials: YOLOv4 incorporates a set of training techniques, called Bag of Freebies and Bag of Specials, that help to improve the network's accuracy and efficiency during training.

Overall, YOLOv4 achieves state-of-the-art performance on several object detection benchmarks, with faster and more accurate results than its predecessors. However, it requires significant computational resources to train and deploy, and may not be suitable for resource-constrained environments.

6.2 FUTURE SCOPE

The future scope of YOLOv4 lies in further improvements in accuracy, speed, and efficiency in object detection. Some possible areas of development and application for YOLOv4 include:

Integration with other AI technologies: YOLOv4 can be integrated with other AI technologies such as natural language processing, computer vision, and robotics to develop more advanced and intelligent systems.

Real-time applications: YOLOv4 can be applied to real-time applications such as self-driving cars, drones, and surveillance systems, where fast and accurate object detection is essential.

Customization for specific domains: YOLOv4 can be customized for specific domains such as medical imaging, agriculture, and sports analytics to detect objects and events relevant to those domains.

Resource efficiency: Further research can be conducted to improve the efficiency of YOLOv4, making it more accessible and practical for resource-constrained environments.

Multi-object tracking: The YOLOv4 architecture can be extended to perform multi-object tracking, which involves tracking the movement of objects over time, opening up applications in autonomous driving, crowd monitoring, and sports analysis.

Overall, YOLOv4 has a wide range of applications and potential for further development, making it an important technology in the field of object detection and computer vision.

REFERENCES

- [1] Aneesh Kar, Sukanya Shyamal, Suvarna Chatterjee, Anaika Basi, Hamadi Nath Saah, Srimati Choudhury, “Speed Controlling & Traffic Management System (SCTMS)”, IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2020.
- [2] D. Balcones, D. Lorca, M. Sotelo, M. Gavilan, S. Iavars, I. Parra, and M. Ocea, “Real-time vision-based vehicle detection for rear-end collision mitigation systems,” in Computer Aided Systems Theory - EUROCAST 2009, ser. Lecture Notes in Computer Science, R. Moreno-Daze, F. Pichler, and A. Quesada-Arancini, Eds. Springer Berlin Heidelberg, 2009, vol. 5717, pp. 320–325
- [3] D. Ponsa and A. Lopez, “Cascade of classifiers for vehicle detection,” in Proceedings of the 9th International Conference on Advanced Concepts for Intelligent Vision Systems, ser. ACIVS'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp.980–989.
- [4] Nazmul Hossain, “Vehicle Speed Control and Security System”, International Conference on Electronics, Information, and Communication (ICEIC), 2021
- [5] P. Niska, “Automatic traffic estimation using image processing,” International Journal of Signal Processing, Image Processing and Pattern Recognition, vol. 5, no. 4, pp. 167 – 174, 2012.
- [6] S Arun Prakash, Aravind Mohan R, Rahul M Warriar, R Arun Krishna, Sooraj Bhaskar A, Aswathy K Nair, “Real Time Automatic Speed Control Unit for Vehicles”, 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2019.
- [7] Toufic Aziz, Tarek Mahmud Faisal, Heung-Goon Ryu, Md.Vehicle identification using edge-based candidate generation and appearance-based classification is described by G. Y. Song, K. Y. Lee, and J. W. Lee in Intelligent Vehicles Symposium, 2008 IEEE, 2008, pp. 428–433.