TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Khwopa College of Engineering
Libali, Bhaktapur
Department of Computer Engineering

A REPORT ON

Vehicle Speed Estimation System Using YOLO and deepSORT

Submitted in partial fulfillment of the requirements for the degree

BACHELOR OF COMPUTER ENGINEERING

Submitted by

Anish Shrestha	KCE/076/BCT/005
Ankit Kayastha	KCE/076/BCT/006
Asim Bhadra	KCE/076/BCT/010
Pragya Shrestha	KCE/076/BCT/024

Under the Supervision of
Er. Niranjan Bekoju

Khwopa College of Engineering
Libali, Bhaktapur
March 10, 2023

Certificate of Approval

The undersigned certify that the sixth semester project entitled **"Vehicle Speed Estimation System Using YOLO and deepSORT"** submitted by Anish Shrestha, Ankit Kayastha, Ashim Bhadra and Pragya Shrestha to the **Department of Computer Engineering** in partial fulfillment of requirement for the degree of **Bachelor of Engineering in Computer Engineering**. The project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled, bona fide and ready to undertake any commercial and industrial work related to their field of study and hence we recommend the award of Bachelor of Computer Engineering degree.

.....
Er. Niranjan Bekoju
(Project Supervisor)

.....
Er. Dinesh Gothe
Head of Department
Department of Computer Engineering, KhCE

Copyright

Acknowledgement

We would like to express our deepest and sincere gratitude to our Head of Department, Er. Dinesh Gothe, for his insightful advice, motivating suggestions, and constant encouragement throughout our Bachelor's program, including this project.

We are also grateful to the Department of Computer Engineering at Khwopa College of Engineering for providing us with the opportunity to work on this real-world project and for giving us access to all the necessary resources to complete our proposal.

Our special thanks go to our project supervisor, Er. Niranjan Bekoju, for boosting our efforts and morale with his valuable advice and suggestions regarding the project. He also provided us with a practical understanding of the project scenario and supported us in overcoming various challenges. We would also like to thank Er. Anish Baral for his valuable suggestions and support throughout the project.

Finally, we would like to acknowledge everyone who has contributed, directly or indirectly, to the completion of this project.

Anish Shrestha	KCE076BCT005
Ankit Kayastha	KCE076BCT006
Asim Bhadra	KCE076BCT010
Pragya Shrestha	KCE076BCT024

Abstract

The accurate estimation of vehicle speed is critical for maintaining traffic safety and enforcing speed limit laws. Traditionally, the speed of moving vehicles is estimated using RADAR (Radio Detection and Ranging) technology. However, this method can be expensive, and it requires high-end equipment and extensive human involvement, which may not always be feasible in real-world scenarios. To address these limitations, this report proposes an alternative system for vehicle speed estimation using CCTV footage as input. The proposed approach utilizes the concept of deep learning and computer vision to process the input data and provide accurate speed estimates for moving vehicles. Specifically, the system employs the YOLO (You Only Look Once) technique for real-time vehicle detection and the deepSORT algorithm for vehicle tracking. The homography mapping technique is then utilized to estimate the vehicles' speed accurately. By leveraging the power of deep learning and computer vision, the proposed system provides a cost-effective and efficient solution for speed detection. This approach can significantly contribute to ensuring the safety of civilian lives and preventing accidents on the roads. The proposed system is highly promising, and future research can further enhance its accuracy and efficiency. Overall, this report provides valuable insights into the development of a state-of-the-art system for vehicle speed estimation, which can have far-reaching implications in the field of vehicle surveillance.

Keywords:*Computer Vision, Convolutional Neural Network, Deep Learning, deepSORT, Homography, YOLO*

Contents

List of Tables

List of Figures

List of Abbreviation

Abbreviations	**Meaning**
AI	Artificial Intelligence
CCTV	Closed-Circuit Television
CNN	Convolutional Neural Network
DBSCAN	Density-based Spatial Clustering of Applications with Noise
FN	False Negative
FP	False Positive
GPU	Graphics Processing Unit
IoU	Intersection Over Union
KB	Kilo Byte
NMS	Non-Max Suppression
OCR	Optical Character Recognition
OpenCV	Open Source Computer Vision Library
SDLC	Software Development Lifecycle
SORT	Simple, Online and Real Time Tracking
TN	True Negative
TP	True Positive
UI	User Interface
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Background

The daily life of people encounters more problems as the population continuously increases in urban area, and road traffic becomes more congested because of high demand and less level of road capacity and infrastructure. Since the effects of these problems are significant in daily life, it is important to seek efficient solutions to reduce them. [1] The field of traffic survey is very broad and has many purposes such as traffic jam detection, accident detection, vehicle classification, vehicle counting, etc. One of the most important of these purposes is detecting vehicle speed. [2] A speed detection means detecting the speed of the vehicles at some certain range using the IoT. In this, the monitoring of all the vehicles in the road and detecting the speed of the vehicle as well its number plate is done. If any over speed occurs then it can be detected by the system and information about the driver and will be uploaded to traffic police department so that automatic case filing can be done without any complaint. From this method people will automatically become aware of the speed in which they are riding and thus decreasing the speed leading to a safe environment. [3]

Vehicle speed detector systems from video have become a topic of considerable research attention. [4] Vehicle speed monitoring is very important for enforcing speed limit laws. It also tells the traffic conditions of the monitored section of the road of interest. [5] A traffic monitoring system essentially serves as a framework to detect the vehicles that appear on a video image and estimate their position while they remain in the scene. In the case of complex scenes with various vehicle models and high vehicle density, accurately locating and classifying vehicles in traffic flows is difficult. Moreover, limitations occur in vehicle detection due to environmental changes, different vehicle features, and relatively low detection speeds. Therefore, an algorithm must be developed for a real-time traffic monitoring system with the capabilities of real-time computation and accurate vehicle detection. Therefore, the accurate and quick detection of vehicles from traffic images or videos has theoretical and practical significance. [6]

1.2 Motivation

There are several reasons why someone might be motivated to undertake a vehicle speed detection project. Speeding is a major cause of traffic accidents and fatalities, and detecting speeding vehicles can help improve safety on roads and highways. By accurately measuring vehicle speeds, law enforcement agencies can enforce speed limits and discourage drivers from exceeding them.Vehicles that are driven at high speeds consume more fuel and produce more emissions than those that are driven at lower speeds. By detecting and discouraging speeding, it may be possible to reduce the environmental impact of vehicular traffic.

Therefore, our motivation is to develop a cost-effective and efficient solution for speed detection while contributing to improving traffic safety and preventing accidents caused by speeding. By addressing a critical societal issue, leveraging the power of deep learning and computer vision, and innovating for the future, individuals and organizations can potentially make a significant impact on the safety of civilian lives and the economy.

1.3 Problem Definition

Currently, there are many traffic accidents in Nepal, and the main cause is careless driving. According to the Metropolitan Police Department, there were 9,545 traffic accidents nationwide in fiscal 2077/078, with 166 fatalities. 3,543 accidents have occurred due to speeding. Vehicle speeding is a major road safety issue and should be addressed appropriately to minimize accidents.

According to the World Health Organization, 1.35 million people die from road crashes globally. Higher speeding is estimated to be responsible for about half of the total deaths (i.e. 650,000 people). Similarly, 22 million people are estimated to be injured annually because of speeding. Excluding natural disasters, the highest number of death occurs from road accidents in the whole world. Therefore, this issue is not limited to any particular country, but is a global issue.

Radar speed measurement tools are commonly used to measure vehicle speed, but can be inaccurate in certain cases, such as detecting small vehicles with weak echoes. These tools also have difficulty detecting vehicles with frequent or rapid changes in speed. Most of these technologies require constant human involvement. Considering various factors such as timing and effort, it increases the likelihood of speed detection errors which can lead to fatal consequences. Therefore, there is a need for better techniques for traffic surveillance.

With the advancement in the technology, different governing bodies are in the need of some sort of computerized technology to control this problem of over speed driving.

1.4 Objectives

The main objective of this project is:

- To design and implement a system which can estimate the speed of vehicles.

1.5 Scope and Applications

The major scope of the project is transportation industry. Some of the potential applications of the system are as follows:

- Traffic management and enforcement
- Surveillance and security
- Transportation logistics
- Smart city development
- Vehicle Detection and Counting
- Vehicle Speed Estimation
- Research Sector

Chapter 2

Literature Review

2.1 YOLO

2.1.1 YOLOv1

In this paper [7], Redmon et. al. presented a new approach to object detection. Following are the major features of YOLOv1 model:

- One-shot detection: YOLOv1 is a one-shot detection system, meaning it only takes a single forward pass through the network to detect and classify objects in an image.
- Grid-based approach: The image is divided into a grid of cells, and each cell predicts a fixed number of bounding boxes (typically 2) and class probabilities for the objects contained within the bounding boxes.
- Loss function: YOLOv1 uses a sum-squared error loss function to simultaneously predict the class probabilities and bounding boxes of objects in the image.
- Performance: YOLOv1 achieves real-time object detection with high accuracy, outperforming other state-of-the-art object detection systems at the time of its release. However, it struggles with detecting small objects and objects with complex shapes.

2.1.2 YOLOv2

Redmon et. al. presented about the YOLO v1 in [7]. YOLO model made significant number of localization error. So, a new version of YOLO was introduced as YOLO v2. Following are the major features included in the YOLOv2 model:

- Multi-scale feature extraction: YOLOv2 uses a feature pyramid network to extract features at multiple scales from the input image. This allows the network to detect objects of different sizes more accurately.
- Batch normalization: YOLOv2 uses batch normalization to normalize the inputs to each layer, which helps to reduce overfitting and improve generalization.
- Convolution with anchor boxes: YOLOv2 predicts bounding boxes relative to anchor boxes, which are predefined boxes of different aspect ratios and sizes. This makes the predictions more accurate and reduces the number of false positives.

- Dimension clustering: YOLOv2 clusters the dimensions of the ground truth bounding boxes to determine the anchor boxes, rather than using a fixed set of anchor boxes. This leads to more accurate anchor boxes that are better suited to the specific dataset being used.

- Direct location prediction: YOLOv2 directly predicts the coordinates of the bounding box center, rather than predicting the offset of the center from a grid cell. This leads to more accurate predictions and faster convergence during training.

- Darknet-19 architecture: YOLOv2 uses a new, smaller network architecture called Darknet-19 that is faster and more accurate than the original Darknet-53 architecture used in YOLOv1.

2.1.3 YOLOv3

Redmon et. al. [8] presented some update to YOLO. Authors made the model little bigger than last time but more accurate.

- Multiple anchor boxes per grid cell: YOLOv3 predicts multiple bounding boxes per grid cell, each with different aspect ratios. This allows the network to better detect objects of different shapes and sizes.

- Predictions across multiple scales: YOLOv3 makes predictions at three different scales, which helps to detect objects of different sizes more accurately.

- Detection of small objects: YOLOv3 uses a technique called spatial pyramid pooling to detect small objects. This technique involves dividing the input image into grids of different sizes and scales, and computing features separately for each grid.

- Feature fusion: YOLOv3 uses feature fusion to combine features from different scales and layers in the network. This improves the network's ability to detect objects at different scales and orientations.

- Softmax prediction: YOLOv3 uses softmax to predict class probabilities, rather than logistic regression as used in previous versions of YOLO. This leads to more accurate and confident class predictions.

- Better backbone network: YOLOv3 uses a deeper and more powerful backbone network called Darknet-53, which is more accurate than the Darknet-19 network used in YOLOv2.

- Variants for specific applications: YOLOv3 has several variants, including YOLOv3-tiny for faster object detection on resource-constrained devices and YOLOv3-spp for improved accuracy on larger objects.

2.1.4 YOLOv4: Optimal Speed and Accuracy of Object Detection

YOLOv4 is a one-stage object detection model that improves on YOLOv3 with several bags of tricks and modules introduced in the literature. [9] In this section, the authors elaborated the details of YOLOv4. YOLO v4 was developed by three developers Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4 consists of:
Following are the components of YOLOv4:

- Head: YOLOv3
- Neck: SPP, PAN
- Backbone: CSP Darknet53

Following are the major features included in YOLOv4:

- CSPNet backbone: YOLOv4 uses a new backbone network called Cross-Stage Partial Network (CSPNet), which is more efficient and accurate than the Darknet backbone used in previous versions of YOLO.
- Mish activation function: YOLOv4 uses a new activation function called Mish, which is more effective than the traditional ReLU activation function in reducing the vanishing gradient problem and improving the network's accuracy.
- Improved data augmentation: YOLOv4 uses more advanced data augmentation techniques, such as mosaic augmentation and cutmix augmentation, to improve the network's ability to detect objects in complex scenes.
- Weighted-Residual-Connections (WRC): YOLOv4 uses WRC to connect different layers in the network, which helps to reduce vanishing gradients and improve the network's accuracy.
- Bag of Freebies (BoF): YOLOv4 uses BoF, a collection of techniques that are used to improve the training process, such as label smoothing and auto-augmentation.
- Variants for specific applications: YOLOv4 has several variants, including YOLOv4-tiny for faster object detection on resource-constrained devices and YOLOv4x, which is a larger version with improved accuracy but slower speed.

Overall, these improvements make YOLOv4 an accurate and efficient version of the YOLO algorithm to date. YOLOv4 achieved state-of-the-art performance on several object detection benchmarks and is widely used for real-time object detection applications.

2.2 Historical Developments

Balasubramani Ramasamy (2017) [2] proposed several model for vehicle speed detection using different approaches such as edge extraction, object tracking, motion vector technique, absolute, centroid method and background image subtraction.

Gerát, J., Sopiak, D., Oravec, M., & Pavlovicová, J. (2017) [10] proposed a system for vehicle speed detection from camera stream using image processing methods.In the proposed system, optical low method was improved with Kalmanfilter tracking to solve the problem with overlays with static foreground objects and also improve speed detection. Also foreground detection by Gaussian mixture model was combined with DBSCAN clustering to create more precise object representation.

Afifah, Fatima, Sharmin Nasrin, and Abdul Mukit (2018) [11] published a paper for vehicle speed estimation using image processing. In the proposed method, the speed is determined using distance travelled by vehicle over number of frames and frame rate.

Cormoş, A. C., Gheorghiu, R. A., STAN, V. A., & Dănăilă, I. S. (2020) [12] published a paper describing the use of TensorFlow and OpenCV to detect vehicles. Image processing is partially performed via OpenCV with a data set previously trained with TensorFlow. The monitoring area is marked with a polygon. For image classification, models was trained with YOLO network

Rajalakshmi et al.(2020) [13] published a paper on speed detection using IoT. In this model, centroid tracking is used to keep track of a moving object ID and image processing is done by OCR(Optical Character Recognition) technique to ensure the vehicle register number. From the experiments, it was found that the model assumes the camera to be perpendicular to the road.

C.-J. Lin, J. Shiou-Yun, and H.-W. Lioa (2021) [6] proposed a model for real-time vehicle counting, speed estimation and classification system. This study presents a real-time traffic monitoring system based on a virtual detection zone, Gaussian mixture model (GMM), and YOLO to increase the vehicle counting and classification efficiency.

S.N	Title	Summary
1	Research of vehicle speed detection algorithm in video surveillance [14]	The paper studied the vehicle speed detection algorithm based on moving target detection in video. Experimental results showed an accuracy of 90.3% since it did not take account of the actual image-forming principle of camera.
2	Customized Deep Learning Technique for Vehicle Detection Along with Speed Estimation [15]	This project uses YOLO technique for real-time vehicle detection technique, and the centroid approach for vehicle tracking. At optimal conditions, YOLOv5 gave the performance of GFLOPS of 17.0, params of 7.3M, speed of 2.2ms which is great to estimate vehicle speed.
3	A Real-Time Vehicle Counting, Speed Estimation, and Classification System Based on Virtual Detection Zone and YOLO [6]	This study presents a real-time traffic monitoring system based on a virtual detection zone, GMM and YOLO to increase the vehicle counting and classification efficiency.The proposed method with YOLOv4 also achieves the highest classification accuracy of 99.1%, 98.6%, and 98% in daytime, night time, and rainy day, respectively. The average absolute percentage error of vehicle speed estimation with the proposed method is about 7.6%.
4	A Video-Based System for Vehicle Speed Measurement in Urban Roadways [16]	The system uses an optimized motion detector and a noveltext detector to efficiently locate vehicle license plates in image regions containing motion. Vehicle speed is measured by comparing the trajectories of the tracked features to known real world measures. The model achieved a precision of 0.93 and a recall of 0.87.
5	VEHICLE DETECTION USING YOLO V3 FOR COUNTING THE VEHICLES AND TRAFFIC ANALYSIS [17]	This study analyzed two techniques for vehicle detection namely YOLO V3 and CNN. The study concluded that the efficiency of YOLOv3 is 95 and that of CNN is 85. YOLOv3 tends to be additionally reached out for auto vehicle discovery, speed estimation and course of development for every vehicle

Table 2.1: Review Matrix with Research Papers and summary of corresponding papers.

Chapter 3

Requirement Analysis

3.1 Software Requirement

Softwares required for preparing our system includes:

a. Clickup

b. Darknet

c. Excel

d. Flask

e. Github

f. Google Calendar

g. Google Colaboratory

h. OpenCV

i. Overleaf

j. Python

k. Sklearn

l. Slack

m. Star UML

n. Visual Studio Code

3.2 Hardware Requirement

Our prepared system for vehicle speed estimation requires following hardware requirements:

a. Processor: To operate this system, we generally require an Intel-i5 processor. However, the operational feasibility in CPU is quite slow so, it is recommended to use CUDA enabled GPU.

b. Graphics Card: Google Colab provided the privilege to GPU (12GB NVIDIA Tesla K80 GPU) for training our model. In general, the requirement for processing power is CUDA enabled GPU with minimum of NVIDIA Graphics Card (4 GB,GTX 1060 Ti) and 16 GB RAM for smooth experience.

3.3 FUNCTIONAL REQUIREMENT

3.3.1 Vehicle Identification and Tracking

The system should be able to correctly identify vehicles and track it for speed calculations.

3.3.2 Vehicle Speed Calculation

The system should be able to determine the speeds of moving vehicles with higher accuracy.

3.4 NON-FUNCTIONAL REQUIREMENT

These requirements are not needed by the system but are essential for the better performance of the proposed system. The points below focus on the non-functional requirement of the system.

3.4.1 Reliability

The system should be reliable to use and have user friendly interfaces.

3.4.2 Maintainability

The system should be maintainable and it should be able to train on new input data and scalable to millions of data points.

3.4.3 Performance

The system should run smoothly and not consume too much processing power.

3.4.4 Accuracy

Vehicle Speed Detection Engine should be able to accurately estimate the speed of moving vehicle.

3.4.5 Robustness

The system should be able to handle unexpected errors in proper ways without crashing the system.

Chapter 4

Feasibility Study

4.1 Technical Feasibility

Most of the software required are easily available online. All the heavy processing and GPU requirements that was needed was done freely using Google Collaboratory cloud computing service. All the required general processing can be done using available devices. For quality experience, computing devices with minimum of 16 GB RAM, 512 GB SSD storage and intel i5 processor along with NVIDIA Graphics Card (4 GB, GTX 1060 Ti) is required.

4.2 Operational Feasibility

It is attempted to make user friendly interfaces as much as possible for operation. Users can adapt to the system easily even without dense training as it is easy to operate with an interactive UI.

The main parts of our system are:

4.2.1 Vehicle Detection and Tracking

Custom YOLOv4 model was trained using our self prepared dataset for detection, and for tracking part, deepSort algorithm was used which was pre-defined and easy to implement. Integration and synchronization of these modules were quite difficult, but we managed to fix those difficulties.

4.2.2 Vehicle Speed Estimation

The concept of Homography mapping was used for calculating distance which is crucial for speed estimation. It was difficult but completed in short time with teamwork.

4.3 Economic Feasibility

The total expenditure of the project is just computational power. Dataset required was prepared from the footages captured from highway. The computational power required can be fulfilled by using our own laptops alongside Google Colaboratory cloud computing service so, the project is economically feasible.

4.4 Scheduling Feasibility

We managed to meet the deadline of project as mentioned in Gantt Chart at A.3 Schedule. Here is proper management of time as scheduled. We have given almost top priority for the optimum analysis of the algorithm to get higher accuracy and testing of models. Documentation remains as important as any other task so throughout the whole project documenting each and every work was continued. For this in every week we have developed the weekly report and in every meeting, minutes of that particular meeting is taken for the works to be done in next week. Thus, we managed to complete our project within the time frame but lots of things can be further improved in future revised versions.

Chapter 5

Methodology

5.1 Agile Method as Software Development Model

Agile development model is a type of Incremental model in which software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained. It has traditionally used for time critical applications.

The project workflow starts with sprint meeting for initial discussion of sprints. All the backlogs are evaluated and briefly discussed, then they are assigned to each of the team members accordingly. A weekly meeting with the supervisor was conducted every Monday and weekly minutes was submitted to the supervisor. We also conducted stand up meeting in the midweek to discuss the tasks completed and problems occurred during development. Backlogs were shifted in scrum board as required from to do section, doing section and completed section. After end of each sprint, a retrospective meeting was conducted to see the speed of progress and discuss steps towards improvement. Appendix Figure shows out development timeline so far.

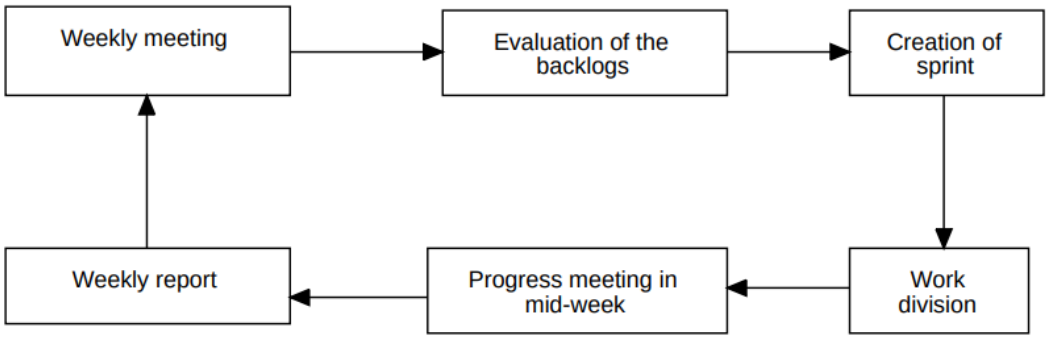


Figure 5.1: SDLC Workflow

5.2 Clickup as Kanban Board

Clickup was used as Kanban Board to organize, distribute, manage and track all the tasks assigned to each members of our team. All the works done in base level as well as the meeting minutes were recorded in Clickup as shown in A.1.

5.3 Overall Phase Followed

The overall phases followed by the project are listed below:

a. Project Initiation

b. Data Exploration

c. Data Processing

d. Model Development

e. Model Evaluation

f. Model Deployment

5.3.1 Project Initiation

In this step, we tried to understand the problem, solve it, and produce an outcome that meets our needs. Before starting our project, we researched about the problem, data, and context. We identified the goal and how it aligns with what's possible using deep learning techniques.

5.3.2 Data Exploration

In this phase, the dataset required for our project were identified. CNN's like YOLO require loads of images to train. These images must have lots of variations in them to be robust enough. Therefore, a popular use case is to use transfer learning where a pre-trained model is used and we can then train a new model by starting from this point. The datasets required for the project has been prepared on our own.
For dataset collection, video footages of highway of around 4 hours and 30 minutes was captured from the overhead bridge of four different locations namely Thimi, Gatthaghar, Sallaghari and Suryabinayak in three different period of time: morning, day and evening with varying exposure settings.
After all the necessary pre-processing, dataset was split on the proportion of 80:20 train test split with respective counts as follows:

Type	**Training**	**Testing**
Vehicle Image	7348	1837

Table 5.1: Dataset Split for Vehicle Detection

5.3.3 Data Processing

For data preprocessing, we applied following techniques:

5.3.3.1 Frame Sampling

After the video footages were captured, we sampled the video frames and obtained images of a single frame per second of the video. Similarly, we also removed the images with abnormalities and images with no vehicles.

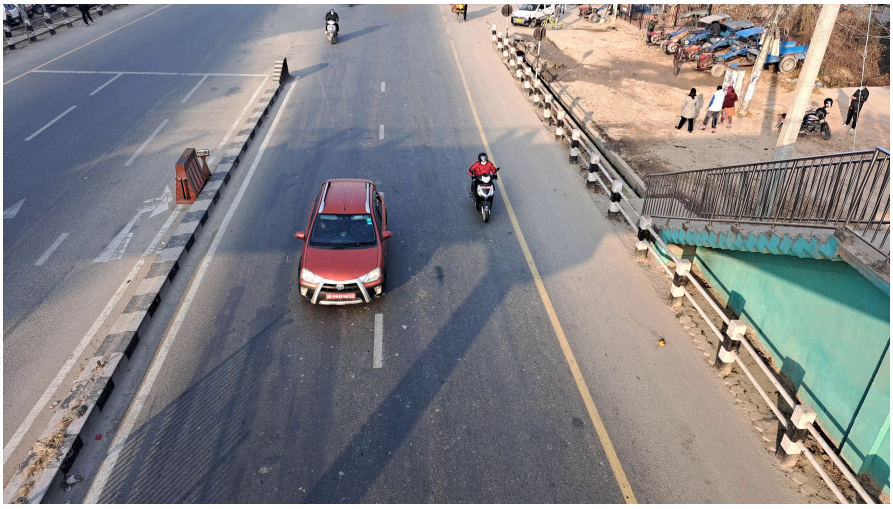


Figure 5.2: Frame after Sampling

5.3.3.2 Region of Interest Selection

After sampling the frames of the videos, region of interests were selected in the image where more vehicles were present and easily distinguishable. Then it is labelled using labelImg tool and assigned a single class named "crop" to each region of interest inside the images. After the selection of region of interest in all images, we cropped the region of interest using the label file and generated cropped images.

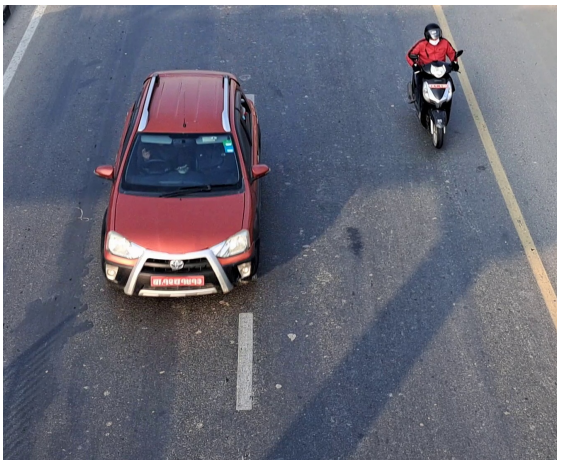


Figure 5.3: Cropped ROI from Frame

5.3.3.3 Dataset Labelling

Once the cropped images were obtained, we labeled each cropped images using labelImg tool. The whole dataset was labeled into two classes: bike and vehicle. The 'bike' class included all the two wheelers and 'vehicle' class included all the four wheelers. The interface for labelling dataset using labelImg is shown below:

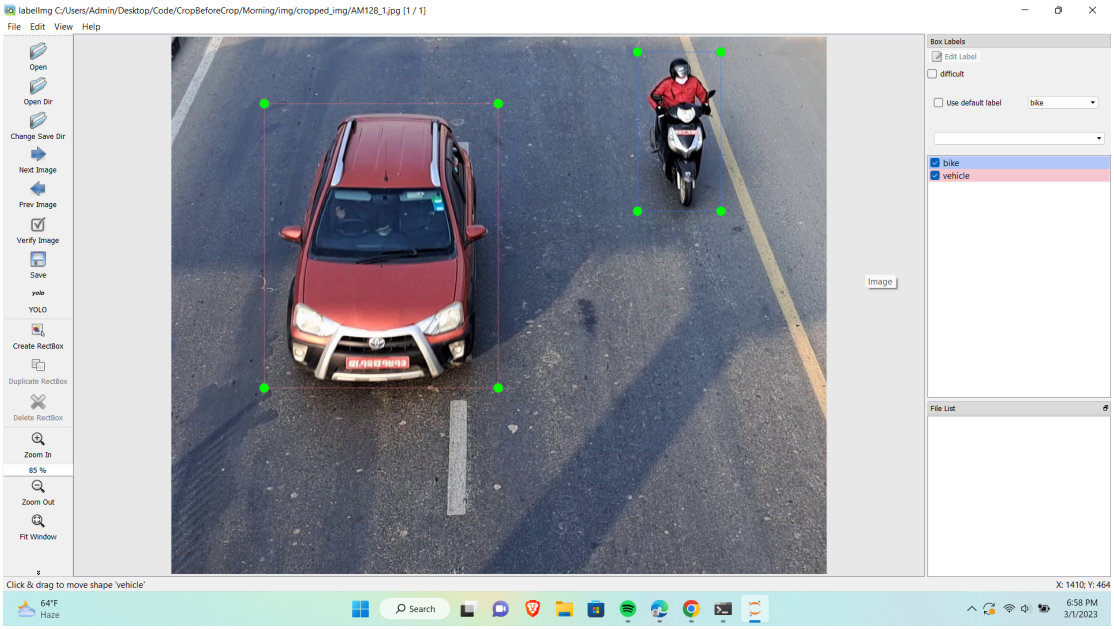


Figure 5.4: Dataset Labelling

5.3.3.4 Dataset Integration and Verification

Since, the preparation of dataset and prepocessing task was divided amongst the team members, the dataset was finally compiled into a single directory and the images and their labels were verified for any anomaly.

5.3.4 Model Training

In this project, YOLOv4 model was trained for vehicle detection using Darknet framework in the Google Colaboratory.

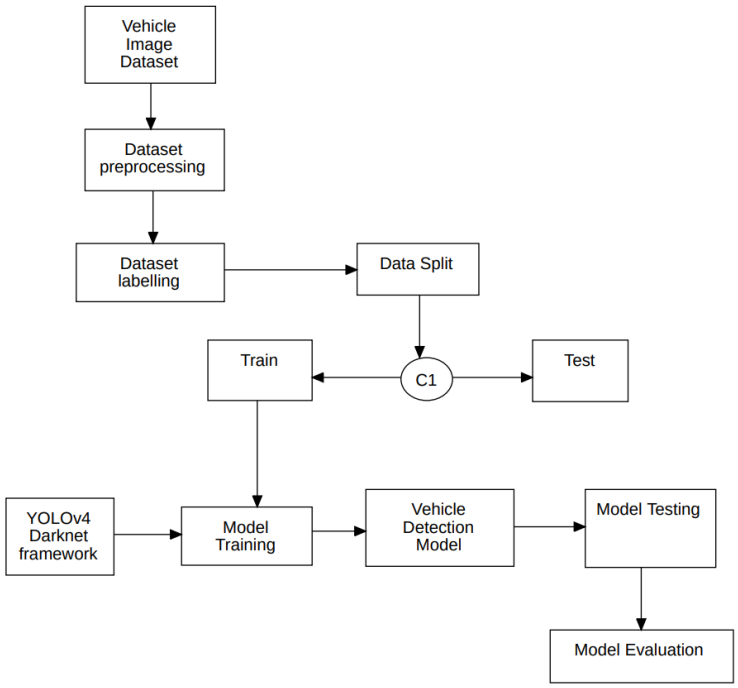


Figure 5.5: Generation of Custom YOLOv4 model for Vehicle Detection

As for hyperparameter tuning, following parameters were taken into consideration:

- Iteration : An iteration refers to a single pass through the entire training dataset during the training process. During an iteration, the YOLOv4 algorithm takes a batch of images from the training dataset and passes them through the network to make predictions for each image. The predictions are compared to the ground truth labels for the images, and the loss function is calculated. The weights of the network are then updated using backpropagation to minimize the loss.
  After the weights are updated, the YOLO algorithm proceeds to the next iteration and repeats the process with a new batch of images from the training dataset.
  We have trained the model upto 6000 iterations.

- Batch: A batch refers to a subset of the training dataset that is processed together during a single iteration of the training process. When training a neural network like YOLOv4, it is common to process the training data in batches rather than processing each training example individually.
  We have set the batch size to 64.

- Subdivisions: Subdivisions refer to a parameter that determines how many times a batch is subdivided during the training process. The batch is divided into smaller subsets, each of which is processed sequentially. The gradients

are then accumulated across the subsets before updating the weights of the network. By using subdivisions, YOLOv4 can process large batches more efficiently and train the network faster with less GPU memory.

- Epoch : An epoch refers to a complete pass through the entire training dataset during the training process. An epoch is completed when the YOLOv4 algorithm has processed every image in the training dataset once. During each epoch, the YOLOv4 algorithm takes the training dataset and splits it into batches of 64. The algorithm then iterates over each batch, passing the images through the network and updating the network weights based on the calculated loss.

- Width and height : Width and height refer to the input size of the images that are processed by the network. The YOLOv4 algorithm requires that all input images have the same size. During the training and testing phases, all input images are resized to the specified width and height before being passed through the network.

- learning rate : The learning rate is a hyperparameter that determines how quickly the weights of the network are updated during training. The weights of the network are updated based on the gradients of the loss function with respect to the weights. The learning rate determines how much the weights are adjusted in the direction of the negative gradient.

In our proposed model, we have adjusted the hyperparameters to obtain optimal speed and accuracy as follows :

SN	**Parameters**	**Value**
1	iteration	6000
2	batch	64
3	subdivisions	32
4	width and height	416
5	learning rate	0.001

Table 5.2: Hyperparameters used

5.3.5 Model Evaluation

In order to evaluate the trained model, we have determined the following parameters:

5.3.5.1 Precision

Precision measures the accuracy of positive predictions, i.e., the fraction of correctly predicted positive examples out of all positive predictions made by the model. It is calculated as the number of true positives divided by the sum of true positives and false positives.

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

5.3.5.2 Recall

Recall is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made.

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

5.3.5.3 mAP(Mean Average Precision)

The mAP compares the ground-truth bounding box to the detected box and returns a score. The higher the score, the more accurate the model is in its detections.

5.3.5.4 F1-score

The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean.

$$F1 - score = \frac{2(Precision * Recall)}{Precision + Recall} \tag{5.3}$$

5.3.5.5 Average IoU

Average IoU is the ratio of intersecting area of detected bounding box and the actual bounding box to the union of area of same two bounding boxes.

$$AvgIoU = \frac{IntersectingArea}{OverlappingArea} \tag{5.4}$$

5.3.6 Model Deployment

In order to deploy the model, we have used PyQt toolkit to create an interactive Graphical User Interface. The system initially takes input of the video file from the user as shown in A.4 and then displays a single frame extracted from the video so that user can mark the points for homography transformation as shown in A.5. Once the user has given inputs of the points, finally the system outputs the video file with estimated speed of the vehicle as shown in A.6 and then saves the obtained data into the directory.

5.4 Work Breakdown

A work breakdown structure is given below:

Task	**Anish**	**Ankit**	**Asim**	**Pragya**
Perform feasibility test	✓	✓	✓	✓
Study about detection model	✓	✓	✓	✓
Select object detection model	✓	✓	✗	✓
Study about detection model architecture	✓	✓	✗	✓
Familiarization with ways of implementing selected model	✓	✓	✗	✓
Study about tracking algorithm	✓	✓	✗	✓
Select object tracking algorithm	✓	✓	✗	✓
Study about tracking algorithm components	✓	✓	✗	✓
Familiarization with ways of implementing selected algorithm	✓	✓	✗	✓
Dataset preperation and preprocessing	✓	✓	✓	✓
Training the detection model	✓	✓	✗	✗
Integrate detection and tracking models	✓	✓	✗	✗
Study about methods for speed calculation	✓	✓	✗	✓
Implement speed estimation module	✗	✓	✗	✗
Deployment	✗	✓	✗	✗
Interpret output of the model	✗	✓	✗	✗
Clarify requirements for implementation	✗	✗	✗	✓
Documentation	✓	✓	✗	✓

Table 5.3: Work Breakdown Structure

Chapter 6

System Design and Architecture

6.1 Use Case Diagram

The Use Case Diagram of the prepared inference system.

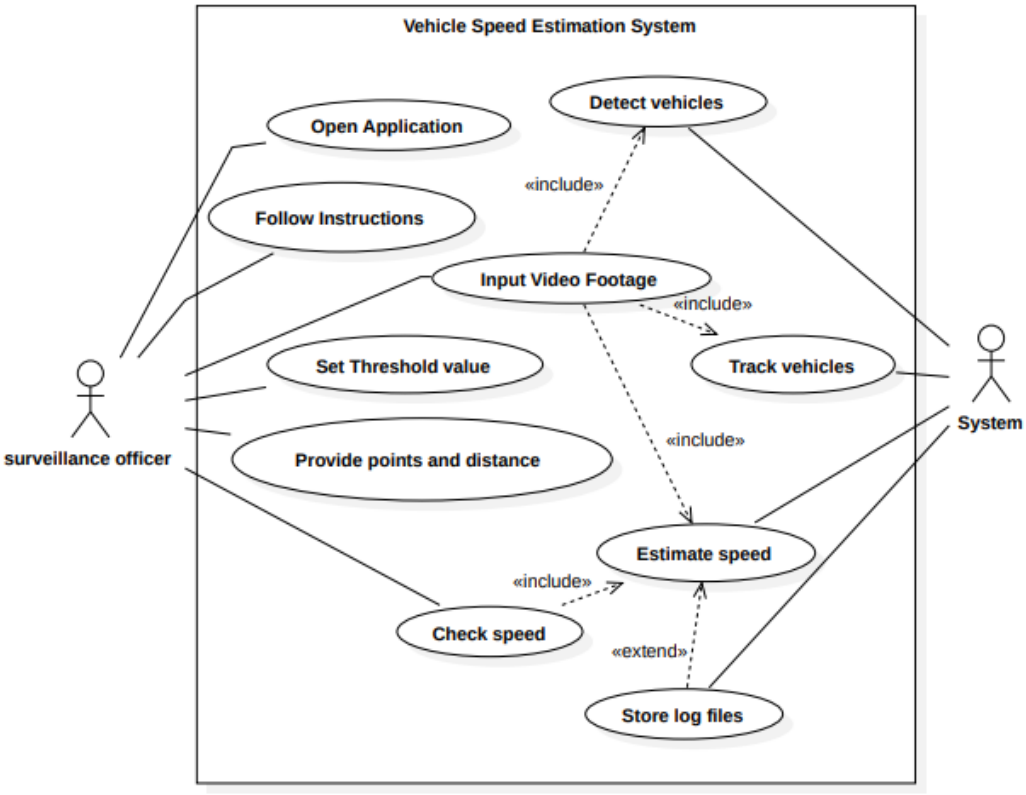


Figure 6.1: Use Case Diagram

6.2 Context Diagram

The Context Diagram shows the top level picture of the system.

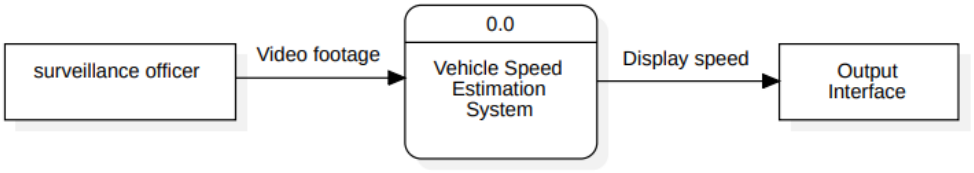


Figure 6.2: Context Diagram of the System

6.3 Sequence Diagram

The Sequence Diagram shows the flow of the process in between the subsystem. And the state when the subsystem are active and when the sub system are passive. The Sequence Diagram of the system is shown below:

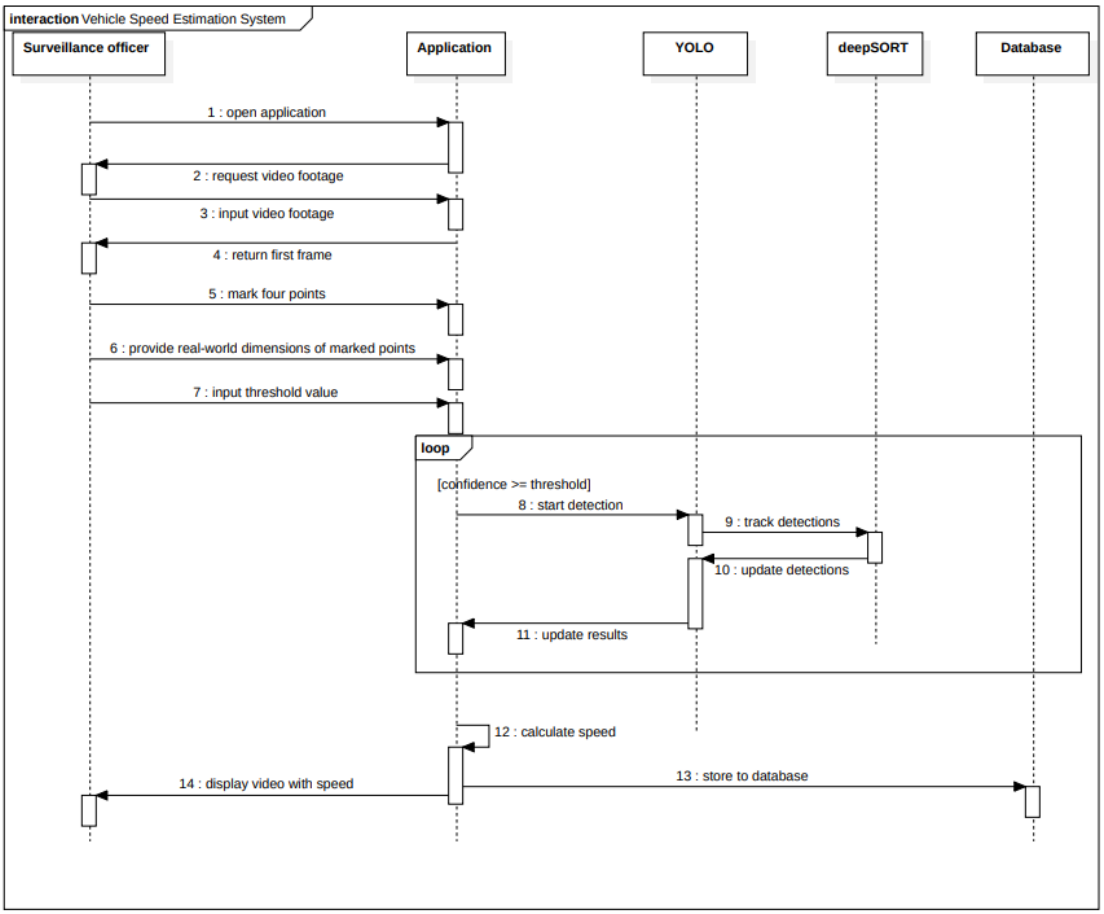


Figure 6.3: Sequence Diagram of the System

6.4 System Block Diagram

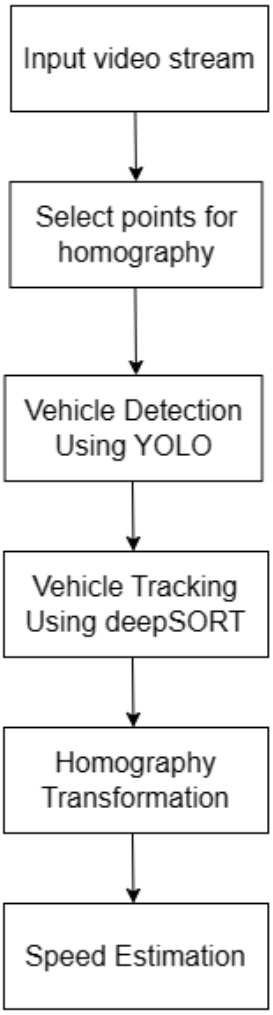


Figure 6.4: Block Diagram of the System

6.5 Workflow of System

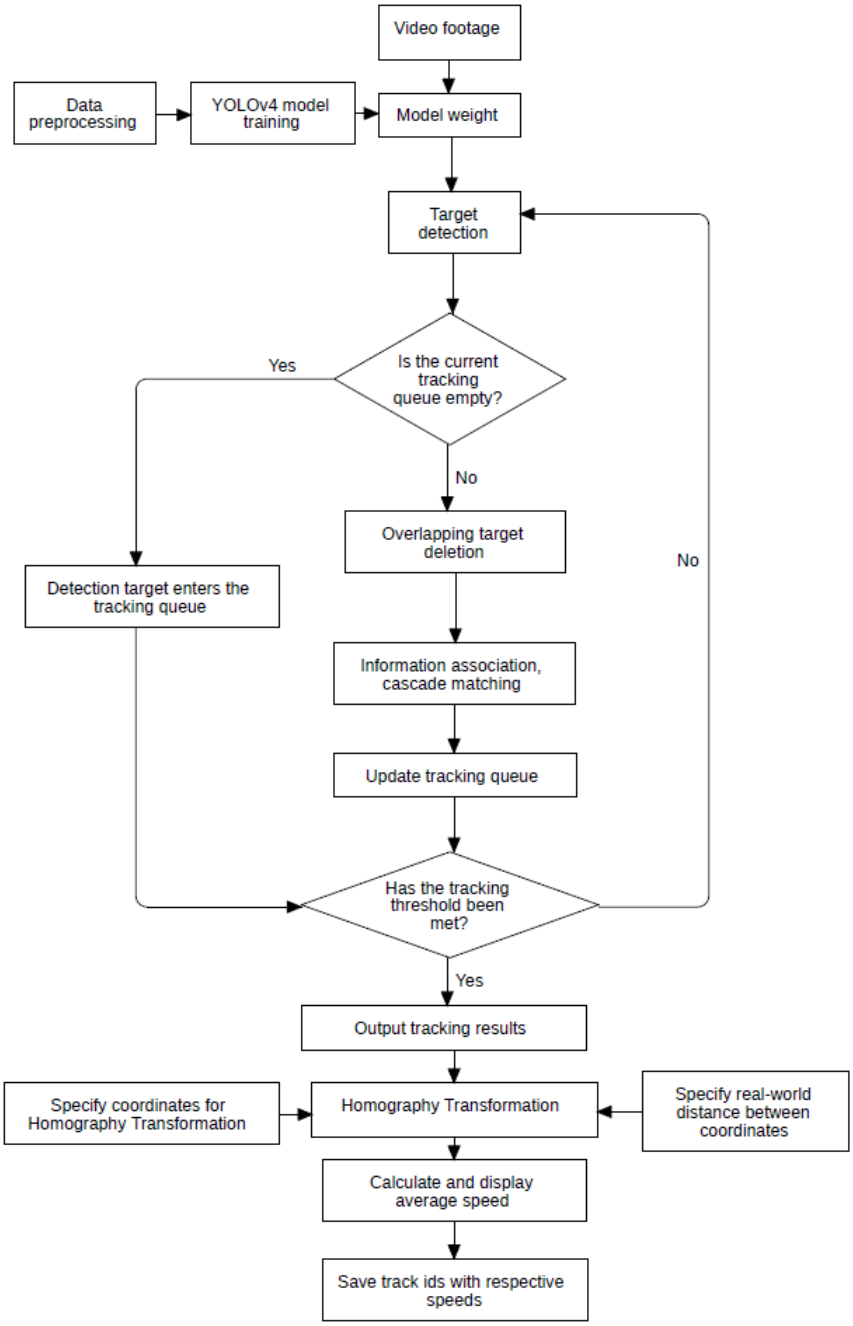


Figure 6.5: Flowchart of the System

6.6 YOLO - You Only Look Once

YOLO stands for "You Only Look Once," and it is a real-time object detection algorithm developed by Joseph Redmon and his team in 2015. YOLO is a neural network-based algorithm that uses a single neural network to perform object detection on an image.
YOLO is known for its fast speed and high accuracy, and is popular choice for object detection in real-time applications such as self-driving cars, surveillance systems.

6.6.1 Residual blocks

First, the image is segmented into a number of square grids. Each grid cell has a dimension of S x S. Every grid cell will predict B boundary boxes and determine confidence score for each box to detect objects that appear within them.

Figure 6.6: Residual Blocks

6.6.2 Bounding box regression

The next step is to determine the bounding boxes which correspond to rectangles highlighting all the objects in the image. Every bounding box in the image consists of the following attributes: probability score of the grid containing an object(pc), Width (bw), Height (bh), Class (c) and Bounding box center (bx,by).

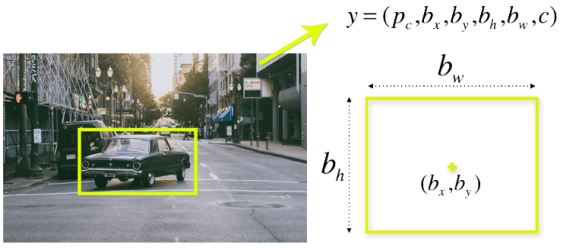


Figure 6.7: Bounding box regression

6.6.3 Intersection Over Union

Intersection over union(IOU) is a measure of overlap between two bounding box. In YOLOv4, IoU is used to measure the accuracy of the predicted bounding boxes for objects in an image.
The IoU is calculated by dividing the area of intersection between the predicted bounding box and the ground truth bounding box by the area of union between the two bounding boxes.

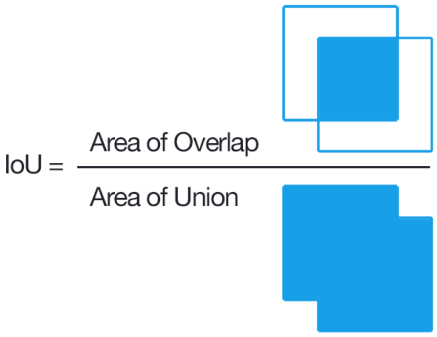


Figure 6.8: Intersection Over Union

In YOLOv4, a threshold value of IoU is set to determine whether the predicted bounding box is considered as a correct detection or not. If the IoU value is greater than the threshold, the detection is considered correct.
Informally, IoU measures how equal two areas are. In terms of size and location

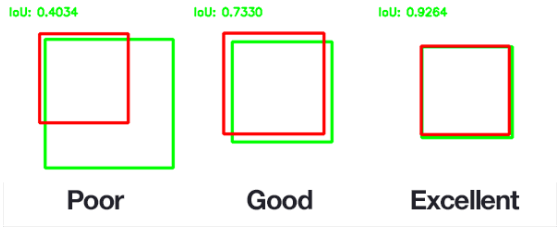


Figure 6.9: Goodness measure of IOU

of the area. If two areas are exactly equal, IOU will be 1. If two areas are far apart, even if their shape is same, they will have IOU 0. And if two areas lie at the same location but their size differs a lot, then also IOU will be a small value. [18]

$$IOU(Box1, Box2) = \frac{Intersection_Size(Box1, Box2)}{Union_Size(Box1, Box2)} \tag{6.1}$$

6.6.4 Non-Maximum Suppression

Non-Maximum Suppression (NMS) is a post-processing algorithm used in object detection tasks to eliminate redundant detections of the same object. It works by selecting the highest-scoring detection (i.e., the detection with the highest confidence score) and suppressing all other detections that have a high overlap (i.e., high IoU) with the selected detection.
The purpose of NMS is to eliminate redundant detections and to select the most accurate detection for each object. This improves the accuracy of the object detection algorithm and reduces false positives.
Input: A list of Proposal boxes B, corresponding confidence scores S and overlap

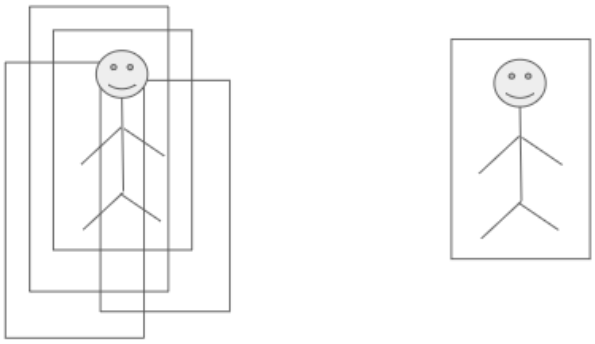


Figure 6.10: Non Maximum Suppression

threshold N.
Output: A list of filtered proposals D.
Algorithm:

A. Sort the detected bounding boxes based on their confidence scores (i.e., probability of containing an object).

B. Select the bounding box with the highest confidence score as the first detection.

C. Calculate the IoU of the selected detection with all other detections.

D. Remove all detections that have a high overlap (i.e., IoU greater than a predefined threshold) with the selected detection.

E. Repeat steps B to D until no more detections are left.

6.6.5 YOLO v1 – CNN Design

YOLO v1 CNN is shown in fig 6.11. [7] It has 24 convolution layer followed by 2 fully connected layers that are responsible for classification of objects and regression of bounding boxes. Here, convolution layers acts as a feature extractor. The final output is 7 x 7 x 30 tensor. Leaky RelU activation function is used for all layers except the final layer. The final layer uses a linear activation function.

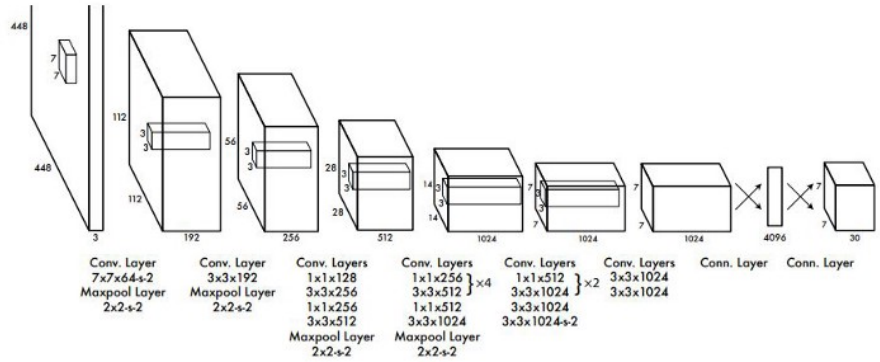


Figure 6.11: YOLO v1 CNN

6.6.6 Network Architecture of YOLOv4

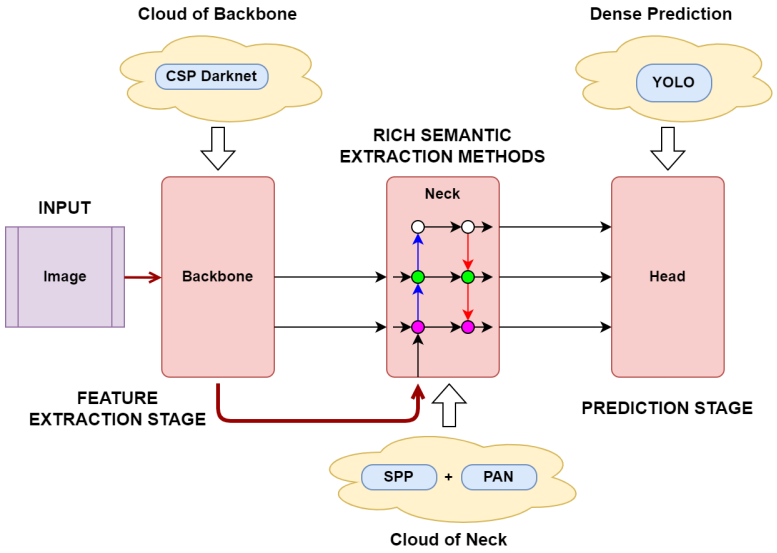


Figure 6.12: Components of YOLOv4

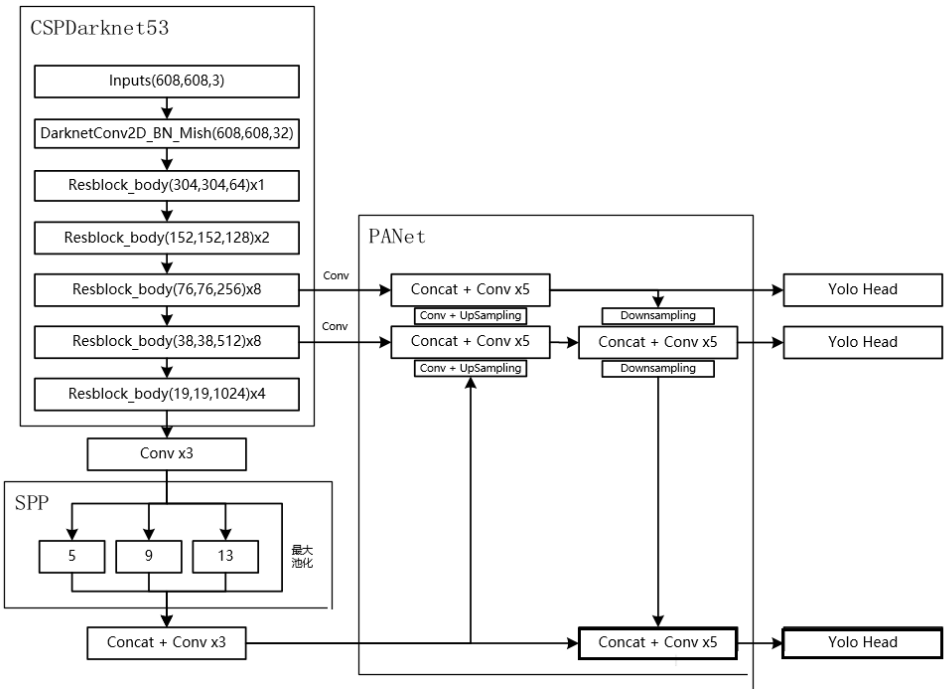


Figure 6.13: Architecture of YOLOv4

The workflow of the YOLOv4 architecture can be broken down into the following steps:

- **Input**: The input to the YOLOv4 algorithm is an image.
- **Preprocessing**: The input image is preprocessed by resizing it to a fixed size and normalizing the pixel values.
- **Backbone Network**: The preprocessed image is fed into the backbone network, which is a modified version of the CSPDarknet53 network. The backbone network extracts features from the input image by passing it through a series of convolutional layers and SPP layers.
- **Neck Network**: The output of the backbone network is then passed through the neck network, which is a set of convolutional layers that further process the feature maps to generate more abstract features.
- **Head Network**: The output of the neck network is then passed to the head network, which is responsible for detecting objects in the image. The head network consists of a set of convolutional layers and a set of detection layers. The detection layers are responsible for predicting the bounding boxes and object classes in the image.
- **Non-Maximum Suppression**: The output of the detection layers can result in multiple bounding boxes for the same object. Non-maximum suppression (NMS) is applied to remove redundant bounding boxes and keep only the most confident one.
- **Postprocessing**: The final output of the YOLOv4 algorithm is a set of bounding boxes and their corresponding object classes and confidence scores. The bounding boxes are converted back to the original image coordinates, and the confidence scores are thresholded to keep only the most confident detections.
- **Output**: The output of the YOLOv4 algorithm is a set of bounding boxes and their corresponding object classes and confidence scores, which indicate the locations of objects in the input image.

6.6.7 Loss Design

YOLO design uses sum squared error as the backbone of loss. [7] Since, multiple grid in the output do not contain any objects and their confidence score is zero. They overpower the gradients from a few cells that contain the objects. To avoid such overpower that leads to training divergence and model instability, YOLOv1 increases the weight ($\lambda_{coord} = 5$) for prediction from bounding box containing object and reduces the weight $\lambda_{noobj} = 0.5$ for predictions from bounding boxes that do not contain any objects.

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2] \tag{6.2}$$

The equation 6.2 shows the first part of YOLOv1 loss. It calculates the error in the prediction of center coordinates of bounding box. The loss function penalizes error in center coordinate of the bounding box, if the predictor is responsible for the ground truth box.

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \tag{6.3}$$

The equation 6.3 shows the second part of YOLOv1 loss. It calculates the error in prediction of bounding box width and height. Here, if the magnitude of error in small bounding box is same as that of large bounding box. It is more wrong for small bounding box that a large bounding box. Hence, a square root of those values is used to calculate the loss. The loss function penalizes error in width and height of the bounding box, if the predictor is responsible for the ground truth box.

$$\sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \tag{6.4}$$

The equation 6.4 shows the third part of YOLOv1 loss. It calculates the error in prediction of object confidence score for bounding boxes that have an object. The loss function only penalize object confidence score, if that predictor is responsible for the ground truth box.

$$\lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \tag{6.5}$$

The equation 6.5 shows the fourth part of the YOLOv1 loss which calculates the error in prediction of object confidence score for bounding boxes that do not have an object.

$$\sum_{i=0}^{s^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \tag{6.6}$$

This equation 6.6 shows fifth part of YOLOv1 loss. It calculates the error in prediction of class probabilities for grid cells that have an object. The loss funciton only penalizes class probabilities error, if an object is present in that grid cell. The loss function is:

$$\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
+\lambda_{coord} &\sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
&+ \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \\
&+\lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \\
&+ \sum_{i=0}^{s^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2
\end{aligned} \tag{6.7}$$

$\mathbb{1}_{i,j}^{obj}$ is 1 when j^{th} bounding box of i^{th} grid has object, otherwise 0.

Where,

x_i : x coordinate of center of bounding box
y_i : y coordinate of center of bounding box
$\hat{x}_i$: x coordinate of center of predicted bounding box
$\hat{y}_i$: y coordinate of center of predicted bounding box
w_i : w coordinate of width of bounding box
h_i : h coordinate of height of bounding box
$\hat{w}_i$: w coordinate of width of predicted bounding box
$\hat{h}_i$: h coordinate of height of predicted bounding box
C_i : object confidence score for bounding box
$\hat{C}_i$: object confidence score for predicted bounding box
$P_i(c)$: class probability
$\hat{P}_i(c)$: predicted class probability
s : output size
B : No. of bounding box per each cell can predict

6.7 DeepSORT

DeepSORT (Deep Learning-based SORT) is a state-of-the-art object tracking algorithm that combines the advantages of deep learning with the traditional SORT (Simple Online and Realtime Tracking) algorithm.

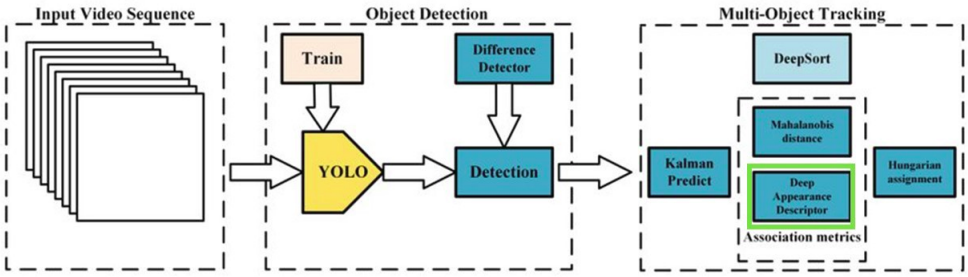


Figure 6.14: Implementation of YOLO with DeepSORT

Following are the major components of deepSORT algorithm:

- **Object detector**: The first component of the DeepSORT algorithm is an object detector, YOLO, which is used to detect objects in each frame of the video. The object detector produces a set of bounding boxes around each object in the frame, along with a confidence score indicating the probability that the object is present.
- **Feature extractor**: The second component of the algorithm is a feature extractor, which is a deep neural network, such as a convolutional neural network (CNN), that is used to extract features from the objects detected in step 1. The features are used to represent the objects in a high-dimensional feature space, where the distance between the features is indicative of the similarity between the objects.
- **Data association**: The third component of the algorithm is a data association module, which is used to associate the objects detected in adjacent frames of the video. The data association module computes the similarity between the features of the objects in adjacent frames and finds the best match based on a distance threshold. This step ensures that the same object is tracked across multiple frames.
- **Kalman filter**: The fourth component of the algorithm is a Kalman filter. The Kalman filter is a mathematical algorithm that uses a series of measurements and predictions to estimate the state of a system. In the context of object tracking, the Kalman filter is used to estimate the position, velocity, and acceleration of the tracked object based on its previous location and motion.

  In DeepSORT, the Kalman filter is used in conjunction with a deep learning-based object detector to track objects in a video stream. The object detector is used to detect the objects in each frame, and the Kalman filter is used to

estimate the state of the detected objects and to predict their future positions.

The Kalman filter works by predicting the future state of the tracked object based on its previous state and the motion model of the object. The predicted state is then updated based on the new measurements from the object detector, using a process called the measurement update. The measurement update combines the predicted state with the new measurements to estimate the current state of the object.

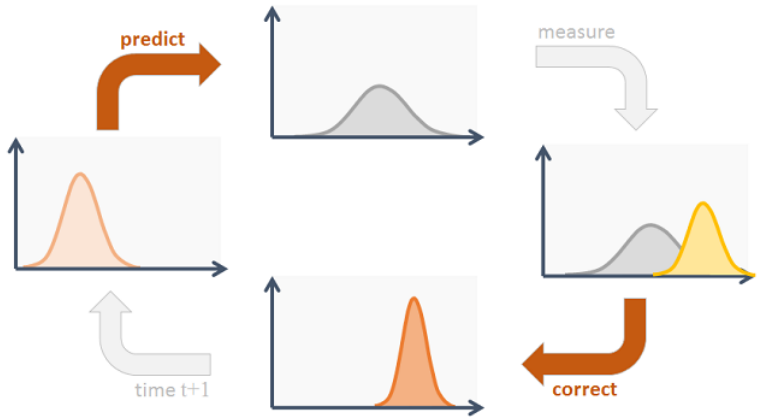


Figure 6.15: Kalman Filter

The use of a Kalman filter in DeepSORT improves the accuracy of the object tracking algorithm by reducing the impact of noisy measurements and tracking errors. By combining deep learning-based object detection with the Kalman filter-based tracking, DeepSORT achieves state-of-the-art results in object tracking benchmarks.

The Kalman filter keeps track of the system and the variance or uncertainty of the estimate. The estimate is updated using a state transition model and measurements.

- **Track management**: The final component of the algorithm is a track management module, which is responsible for creating new tracks for newly detected objects, updating the state of existing tracks, and terminating tracks for objects that are no longer detected. This module maintains a list of all the tracks and their corresponding states and updates the list based on the detections and measurements received from the other modules.

6.8 Homography Transformation

Homography transformation, also known as homography, is a mathematical technique used in computer vision and image processing to map points in one image to corresponding points in another image. It is a type of perspective transformation that is used to correct for the effects of camera perspective and distortion.

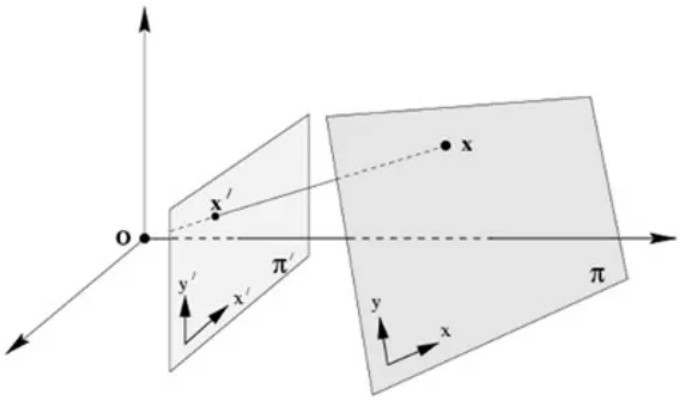


Figure 6.16: Homography Transformation in Coordinate Plane

The homography transformation is typically represented by a 3x3 matrix H, which is used to transform the coordinates of points in one image to the corresponding points in another image. The homography matrix can be computed using a set of corresponding points in the two images, either manually or automatically using computer algorithms.

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Figure 6.17: Homography Transformation Matrix

Once the homography matrix is computed, it can be used to transform the pixels of one image to the coordinate system of the other image. This allows for image alignment and registration. In our project, we have used homography transformation to transform the captured frames into a topview or bird view using the actual real world measurements.

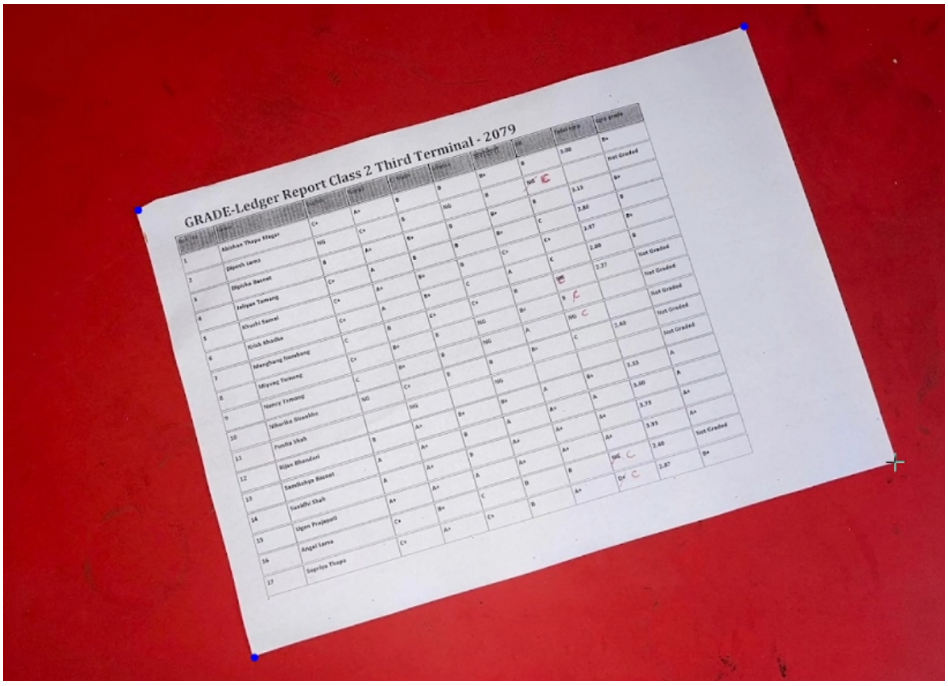


Figure 6.18: Before Homography Transformation

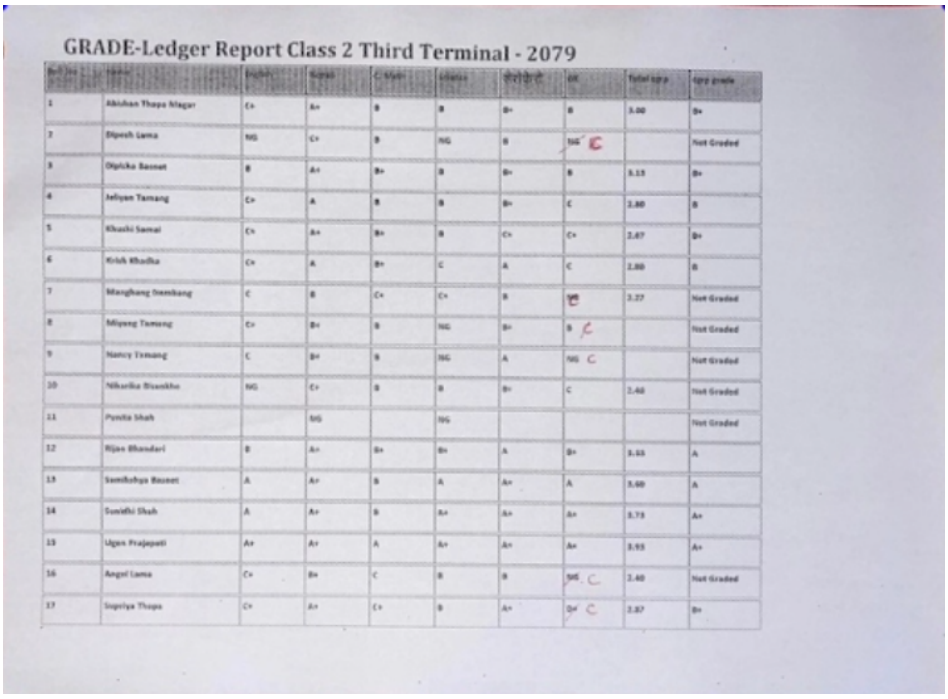


Figure 6.19: After Homography Transformation

In our testcases, we have marked four points on the highway lane and manually collected the measurement between those points. Then we have obtained the homography matrix using those points and we have applied homography transformation as follows to obtain the bird view of the scene for speed estimation.

Figure 6.20: Before Homography Transformation

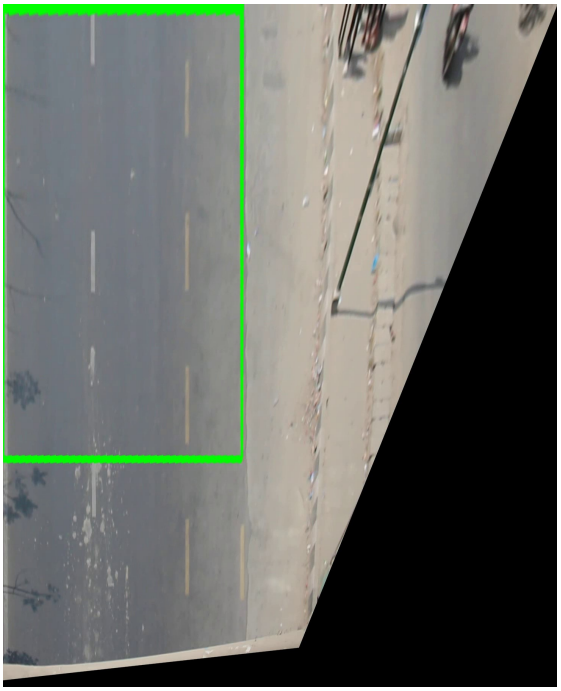


Figure 6.21: After Homography Transformation

6.9 Speed Estimation

Once the bounding box coordinates of each tracked vehicles are obtained in each frames, the model keeps track of the coordinates of the bottom center of the bounding box as a reference point.

Then, the perspective transformation of the reference point is performed using the homography matrix.

Figure 6.22: Tracking of the reference points

After that, the model averages total 5 tracks of the vehicle within the region of interest and calculates average distance between the reference points in each frame.
For speed estimation, following formula is used within the model:

$$speed = Distance\ travelled\ in\ pixels\ per\ frame$$

As the required speed of the vehicle is in km/hr, we need to convert the pixels into km and frames into hr.

As the model takes input of distance between the marked points in centimeters, each pixel in the frames maps to 0.01m.

So, 1 pixel = 0.01 m = 0.01 * 0.001 km
And 1 frame = 1/(fps) seconds = 1/(fps*60*60)

Hence, the desired speed is calculated as :

$$speed = \frac{Distance\ travelled\ in\ pixels\ per\ frame * 0.01 * 0.001}{fps * 60 * 60}\ km/hr$$

Chapter 7

Result and Discussion

7.1 Desktop Outcome

Our actual outcome is quite inconsistent with what we expected. There was unnecessary lagging of processing and the model couldn't properly detect tracked vehicles in the consequent frames which affected the method of speed calculation. In order to produce consistent output, we averaged the speed of the vehicle upon 5 number of tracks from the model.

As for the screenshot of actual outcome main dashboard for speed estimation is shown in figure A.4.

Here, we have only placed overview of the interface with the option to upload video, input homography coordinates and the snap of the desired output consisting of bounding boxeslabeled with object class name and track ID with the estimated speed so that we can verify the detection as well as tracking progress.

7.2 Unit Tests

Several unit test were done for each unit modules after their development and before integration. For unit testing, we have performed two major tests namely evaluation of custom trained YOLOv4 model and the homography testing. As for the tracking model, we have used the pretrained model trained on MARS(Multiple-Attribute Re-IDentification). Finally, the result of the unit tests were used to detect and correct the problems and error founds in modules.

- Detector Evaluation
- Homography Evaluation

7.2.1 Detector Evaluation

Following results were obtained upon evaluation of the YOLOv4 model trained on our custom dataset:
In the test data:
For confidence threshold of 0.25 : TP = 2934, FP = 335 and FN = 78

Metrics	**Calculation**	**Result**
Precision	$\frac{2934}{2934+335}$	0.8975
Recall	$\frac{2934}{2934+78}$	0.9741
mAP	$\frac{0.9872+0.9843}{2}$	0.9858
F1 Score	$2\frac{0.8975*0.9741}{(0.8975+0.9741)}$	0.9342
Avg IoU	–	81.37%

Table 7.1: Evaluation of Detection model

7.2.1.1 Loss Plot

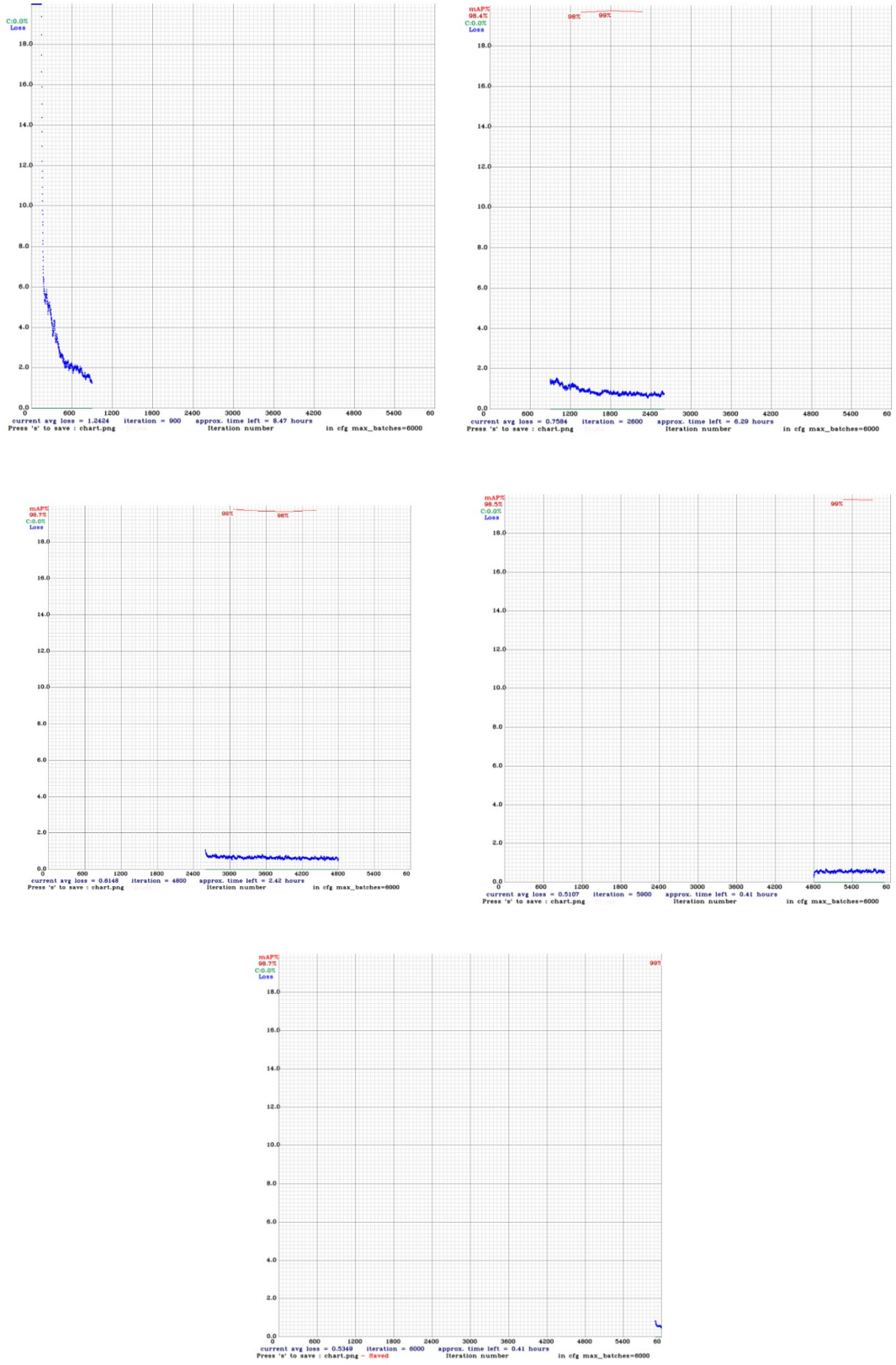


Figure 7.1: Loss Plot for YOLOv4 Detection Model

7.2.1.2 Precision Recall Curve

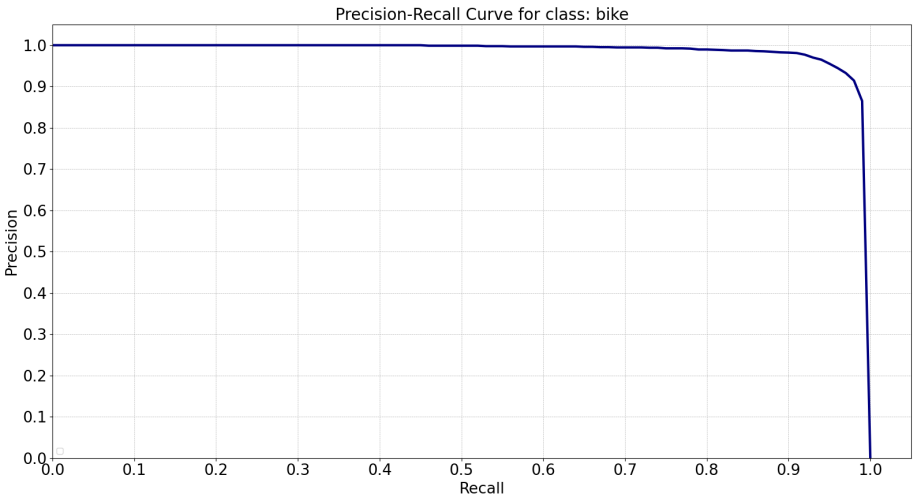


Figure 7.2: Precision Recall Curve for class bike

Average precision = Area under the curve = 0.9872

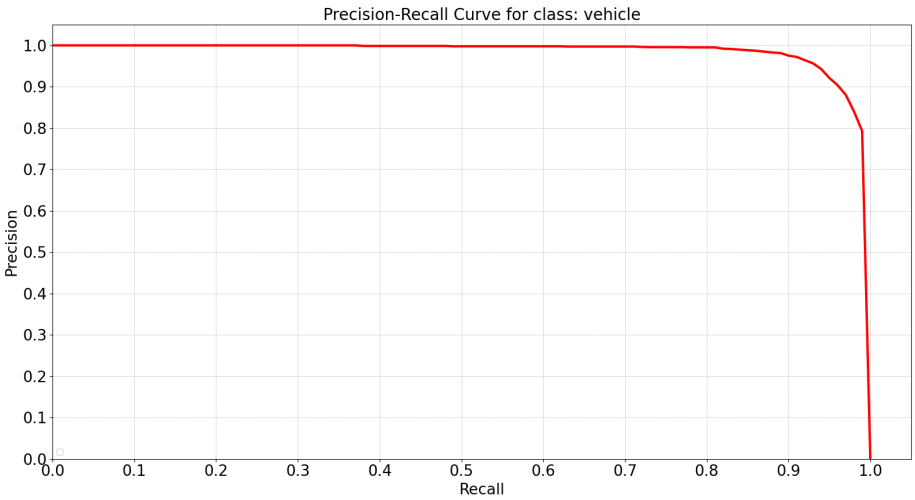


Figure 7.3: Precision Recall Curve for class vehicle

Average precision = Area under the curve = 0.9843

7.2.2 Homography Evaluation

In order for evaluation of the homography transformation, we have taken a reference point on the real world plane. Here, point P1 is marked as the reference point. And the distance between point P1 and the point A as shown in the figure below was measured to be 6.46 m and the distance between point P1 and point B as shown in the figure below was measured to be 7.16 m.

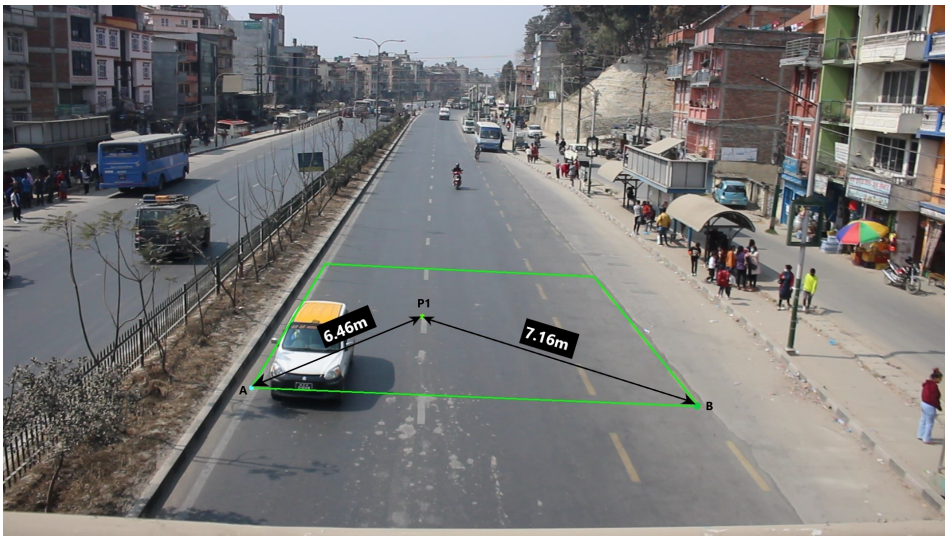


Figure 7.4: Actual Distance Measurement

For the homography transformation, we have taken four reference point that form a rectangle in the real world plane and generated a homography matrix. When the P1 point was transformed about the homography matrix and the distance between point P1 and point A and the distance between point P1 and point B was calculated from the transformed plane, the results were obtained as :

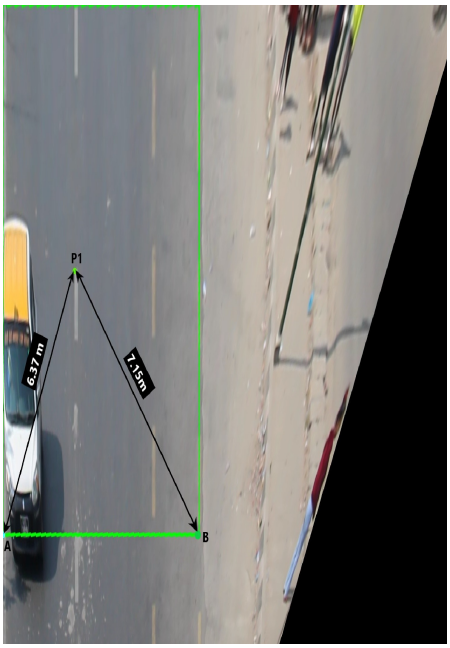


Figure 7.5: Results obtained after homography mapping

7.3 Integration Testing

In order to test the integrated system, we have generated test cases in which the bike under consideration is driven at a constant speed inside the region of interest and the video is captured. Following are the results obtained from the test cases.

S.N.	Grouth Truth Speed	Average Estimated Speed
1	30	28
2	35	33.5
3	39	35.8
4	40	37.2
5	25	22.3

Table 7.2: Evaluation of testcases

7.4 Problems Faced

During the software development lifecycle of our project, we encountered several problems, some of which are listed as follows:-

- **Problems during training**
  While training our model in Google Colab, due to the computation limits of GPU usage in the Colab, we could only train the model to limited number of iterations at a time. So, we had to resume the training process from previous checkpoints. This resulted into excessive cool-down time while training.

  Similarly, the loss graph of the model couldn't be plotted into a single graph as each checkpoints generated new loss graph from the point of resumption. Due to the limitations in computational capability of GPU used in Google Colab, we could only set the subdivisions to 32. This further increased the time required for training.

- **Obtaining ground truth values of speed**
  Speed is a relative concept. So, obtaining the ground truth value of speed is a difficult task. In order to test our system, we have driven the vehicle under consideration at a constant speed.

Chapter 8

Conclusion and Future Enhancements

To conclude, vehicle speed estimation system has been developed succesfully with the model applicable for vehicles operating in Nepal and deployed into a graphical user interface. Although the system predicts the speed value closer to the actual speed, the speed estimated from the system is not accurate and consistent as it gave output of fluctuating speed values when tested with the groundtruth value.

As for Future enhancements for our project, following improvements can be done:

- Optimization of the model is possible by using custom dataset and advanced algorithm for detection and tracking.
- A finer classification can be done. For instance, the vehicle class could be sub-classed into bus,car,truck.
- The model could be refined to detect the vehicle and extract information of the vehicle such as color, number plate information for traffic surveillance purposes.
- Since, the system currently requires users to input the points and their respective distances for homography mapping, this could be replaced with camera auto-calibration methods.
- The accuracy and consistency of the model could be improved.

Bibliography

[1] G. R. Arash, D. Abbas, and R. K. Mohamed, "Vehicle speed detection in video image sequences using cvs method," *International Journal of Physical Sciences*, vol. 5, no. 17, pp. 2555–2563, 2010.

[2] B. Ramasamy, "A review on vehicle speed detection using image processing," vol. 4, pp. 23–28, 11 2017.

[3] G. Aravindh and A. Kowshik, "Speed detection using iot," *International Journal of Computer Applications*, vol. 975, p. 8887.

[4] A. H. Rais and R. Munir, "Vehicle speed estimation using yolo, kalman filter, and frame sampling," in *2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, 2021, pp. 1–6.

[5] J. Wu, Z. Liu, J. Li, C. Gu, M. Si, and F. Tan, "An algorithm for automatic vehicle speed detection using video camera," in *2009 4th International Conference on Computer Science & Education*. IEEE, 2009, pp. 193–196.

[6] C.-J. Lin, J. Shiou-Yun, and H.-W. Lioa, "A real-time vehicle counting, speed estimation, and classification system based on virtual detection zone and yolo," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–10, 11 2021.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[8] A. Farhadi and J. Redmon, "Yolov3: An incremental improvement," *Computer Vision and Pattern Recognition, cite as*, 2018.

[9] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[10] J. Gerát, D. Sopiak, M. Oravec, and J. Pavlovicová, "Vehicle speed detection from camera stream using image processing methods," in *2017 international symposium ELMAR*. IEEE, 2017, pp. 201–204.

[11] F. Afifah, S. Nasrin, and A. Mukit, "Vehicle speed estimation using image processing," *Journal of Advanced Research in Applied Mechanics*, vol. 48, no. 1, pp. 9–16, 2018.

[12] A. C. Cormoş, R. Andrei Gheorghiu, V. A. STAN, and I. Spirea Dănăilă, "Use of tensorflow and opencv to detect vehicles," in *2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2020, pp. 1–4.

[13] S. Jika, "Speed detection using iot," 06 2020.

[14] J.-x. Wang, "Research of vehicle speed detection algorithm in video surveillance," in *2016 International Conference on Audio, Language and Image Processing (ICALIP)*, 2016, pp. 349–352.

[15] S. B. B. D. Benjamin Tamang, Sanskar Poudel and S. Pande, "Customized deep learning technique for vehicle detection along with speed estimation," *EasyChair Preprint*, p. 21, 2022.

[16] D. Luvizon, B. Nassu, and R. Minetto, "A video-based system for vehicle speed measurement in urban roadways," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, pp. 1–12, 09 2016.

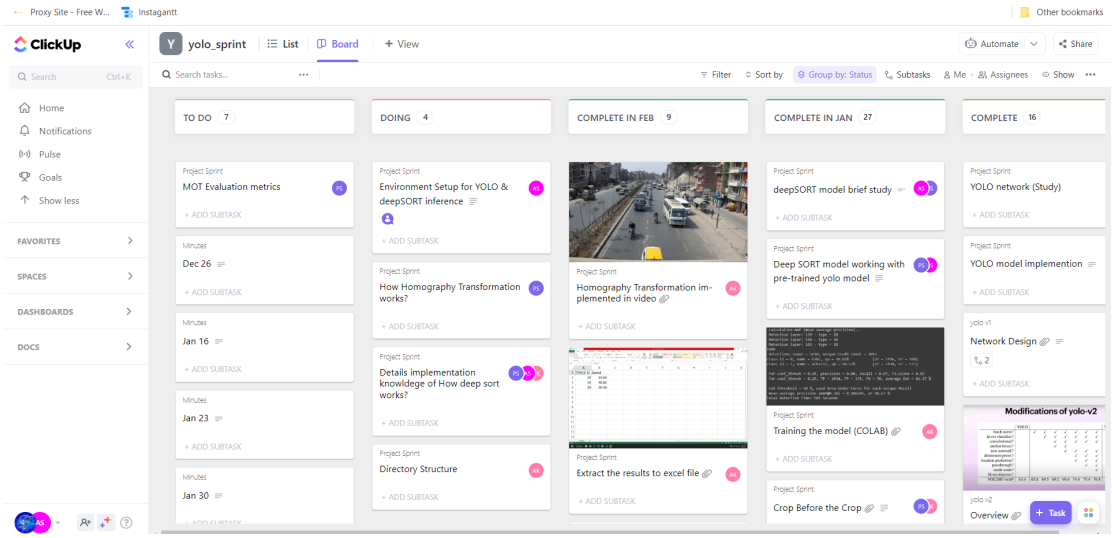
[17] M. G. P. D. G. V. S. G. CH. Ranga Reddy, Angshuman Roy, "Vehicle detection using yolo v3 for counting thevehicles and traffic analysis."

[18] O. Sheremet, "Intersection over union (iou) calculation for evaluating an image segmentation model," Jul 2020. [Online]. Available: https://towardsdatascience.com/intersection-over-union-iou-calculation-for-evaluating-an-image-segmentation

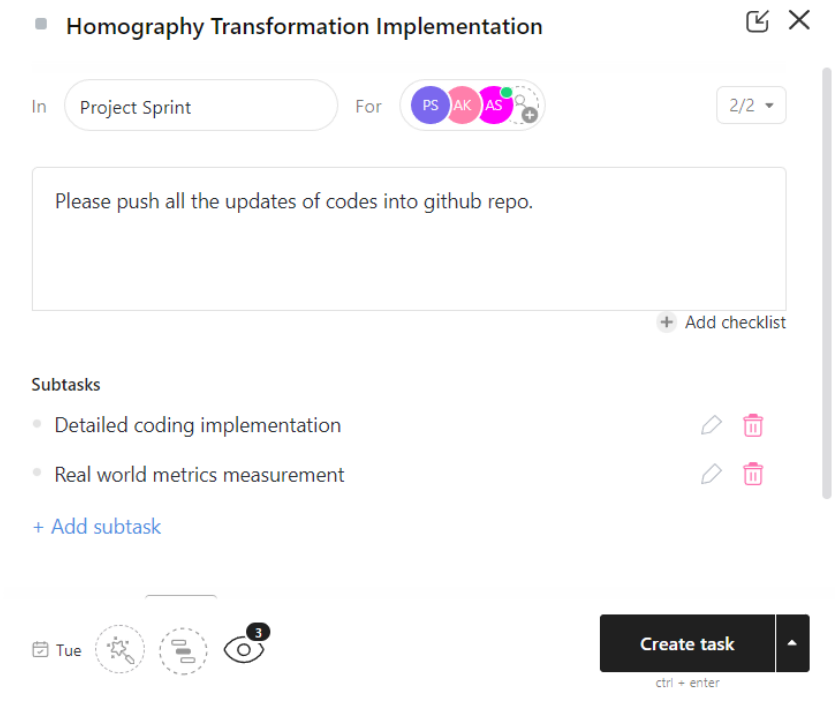
Appendix

A Snapshot

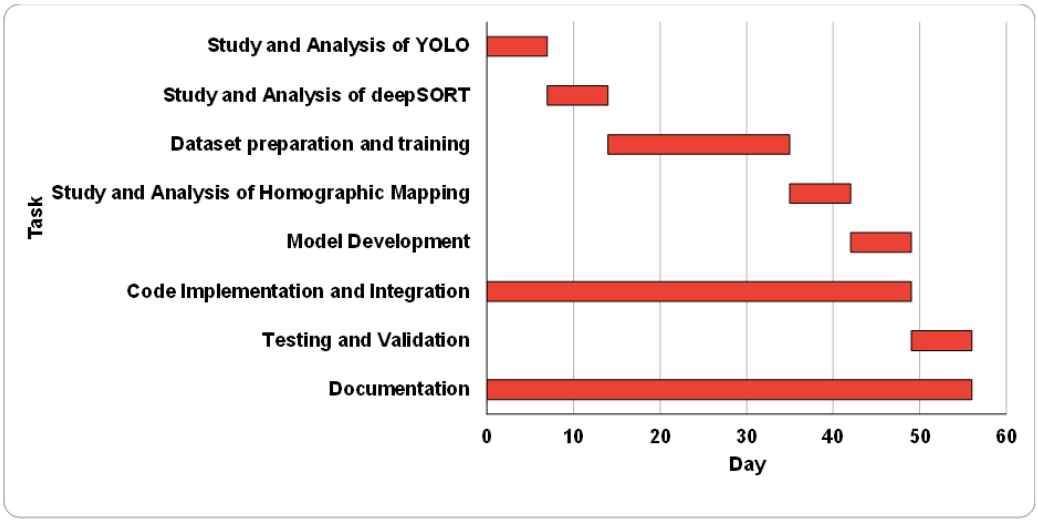
A.1 Clickup



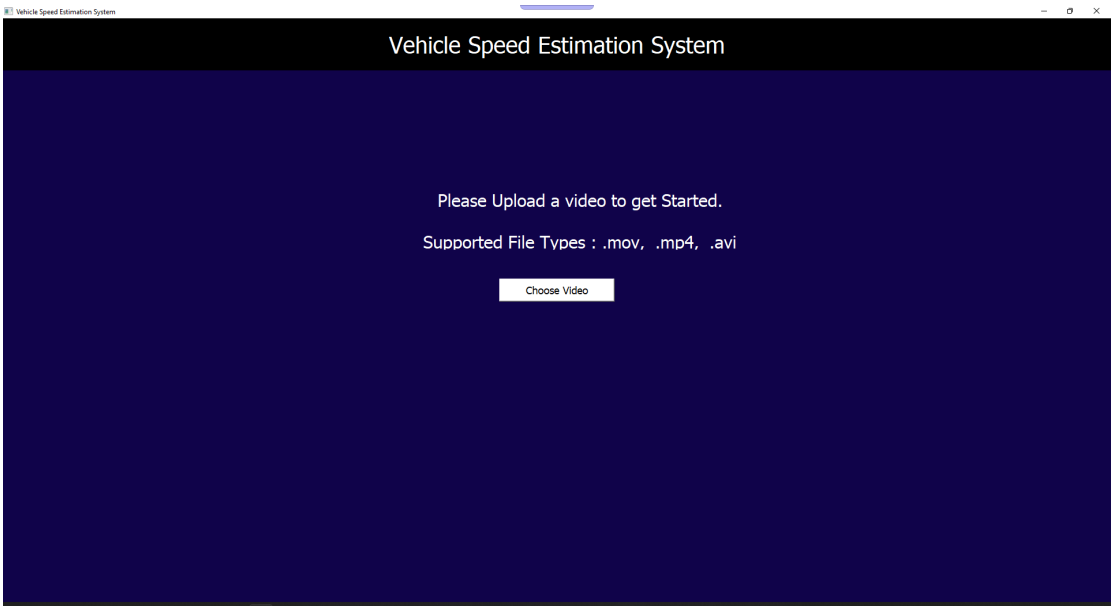
A.2 Create task in clickup



A.3 Gantt Chart of project



A.4 Outcome of Graphical User Interface



A.5 Selection of points for Homography Transformation

A.6 Outcome of the test case