Vehicular Trajectory Classification and Traffic Analysis using YOLOv5 and DeepSORT

Project report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Electronics and Electrical Communication Engineering

by

Sreyan Biswas

(19EC39036)

Under the supervision of

Assistant Professor Adway Mitra

Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology Kharagpur

Autumn Semester, 2022-23

Nov 29, 2022

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: Nov 29, 2022 (Sreyan Biswas)
Place: Kharagpur (19EC39036)

DEPARTMENT OF ELECTRONICS AND ELECTRICAL COMMUNICATION ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Vehicular Trajectory Classification and Traffic Analysis using YOLOv5 and DeepSORT** " submitted by **Sreyan Biswas** (Roll No. 19EC39036) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Electronics and Electrical Communication Engineering is a record of bona fide work carried out by him under my supervision and guidance during Autumn Semester, 2022-23.

Date: Nov 29, 2022

Place: Kharagpur

Assistant Professor Adway Mitra
Department of Electronics and Electrical
Communication Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Sreyan Biswas** Roll No: **19EC39036**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of Electronics and Electrical Communication Engineering**

Thesis title: **Vehicular Trajectory Classification and Traffic Analysis using YOLOv5 and DeepSORT**

Thesis supervisor: **Assistant Professor Adway Mitra**

Month and year of thesis submission: **Nov 29, 2022**

In the realm of highway management, intelligent vehicle detection and counting are becoming more and more crucial. However, because vehicles come in a variety of sizes, it might be difficult to detect them, which has an impact on how accurately counts of vehicles are made. To address this issue, this work looks at a vision-based vehicle detection and counting system. Several highway surveillance videos based on different scenes are used to verify the proposed methods.

The use of deep convolutional networks (CNNs) has achieved amazing success in the field of vehicle object detection. CNNs have a strong ability to learn image features and can perform multiple related tasks, such as classification and bounding box regression. The experimental results show that using deep learning methods can provide higher detection accuracy with multiple objects, especially for the trajectory tracking of the detected vehicles.

Acknowledgements

I place on record, my sincere thanks to Assitant Prof. Adway Mitra, supervisor for this project, for his efficacious advise, perpetual and prolific encouragement and constructive criticism during the course of the work and in the preparation of this project report. I would like to express my sincere gratitude to all the staff of this department for their help rendered in completion of this project-work. Finally I would like to express my gratitude to my team mate Priyanka for her uncompromising dedication towards this project.

Contents

Chapter 1

Introduction

Due to the growth of video surveillance equipment and the increase in processing power over the past decade, vehicle tracking has been an important area of research. Automatic video monitoring, traffic control, and abnormal event detection are all highly necessary. A crucial low-level task required to achieve such intelligence is reliable vehicle detection and tracking.

1.1 Motivation

The major motivation behind the project are:

- Monitoring surveillance is a tedious and complex task, which used to require human checking. Today, machine learning and computer vision system have provided us a way to replace the human monitoring to automatic (AI—artificial intelligence) monitoring. Mostly video data is used for surveillance, for example, vision-based transportation processing and traffic engineering applications. These vision systems, when combined with machine learning algorithm can do wonders like vehicle tracing, flow control, vehicle counting, trajectory detection and prediction, abnormal activity detection, etc.

- The data gathered during vehicle classification and counting can be used in statistical analyzes of the roadway traffic, and the vehicle tracking data obtained can help identify the behavior of the vehicles in the observed part of the roadway
- With the emergence of automatic driving technology that can make the vehicle automatically perceive the surrounding driving environment, vehicle detection and tracking becomes essential.

Chapter 2

Related Work

2.1 Literature Survey

Currently, there are two main categories for vision-based vehicle object detection: conventional machine vision methods and complex deep learning methods. Traditional machine vision methods use the motion of a vehicle to separate it from a fixed background image. In contrast, deep learning uses the concept of "end-to-end learning". These systems involve not only recognizing and classifying every object in an image but localizing each one by drawing the appropriate bounding box around it. In this report, we will focus on detection and tracking using deep learning models.

2.1.1 Particle Filtering

A traffic monitoring method based on image processing and particle filtering. The proposed method detects and classifies automatically moving vehicles in previously defined classes.

- The moving vehicles present in the video sequence are detected using a segmentation algorithm that classifies the image pixels as foreground or background regions, this can be done based on the background and foreground probabilistic models defined for each pixel, and updated in each instant of time t.

- This method is based in the fact that the background regions should have low temporal variability of luminance, and the foreground regions should have high variability of luminance. A binary mask is created with the classification results. Each connected foreground regions is considered as being a moving vehicle.

- The vehicles detected in the previous step are counted and classified according to the size of their area in the image. The vehicle area is the area of the rectangular bounding box that contains the vehicle. Before applying the classification and counting procedure, it is necessary to define a region of interest of the roadway where the vehicles are going to be classified, counted and tracked.

- After the vehicles have been detected and classified, they are tracked by a particle filtering algorithm. Particle Filtering is concerned with the problem of tracking single and multiple objects. It is a hypothesis tracker, that approximates the filtered posterior distribution by a set of weighted particles. It weights particles based on a likelihood score and then propagates these particles according to a motion model. They are sequential Monte Carlo methods based on point mass representations of probability densities, which are applied to any state model.

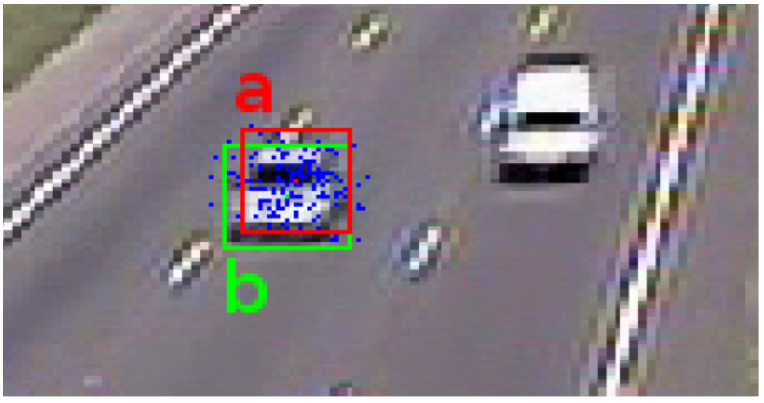

FIGURE 2.1: Example of vehicle tracking using detected vehicle information. The figure shows a detected vehicle (a car) marked with the rectangular box (rectangle "a" in green) and the particle filtering tracking result (rectangle "b" in red)

Chapter 3

Objective and Problem Definition

3.1 Objective

As a part of this thesis, we try achieve the following set of the objective to do the analysis.

- Reviewing the past works done in the domain of Object detection, tracking and traffic analysis.
- Scraping and Building the Extended dataset, to include vehicles that are found on Indian roads.
- Creating Pre-Trained YOLOv5 model for Object Detection.
- Classifying the objects detected into various classes such as cars, trucks, motorcycles etc.
- Training deep learning models to track the direction of the detected objects. Defining of directions as left to right, right to left etc.
- Measuring the precision metrics that will help us in determining the accuracy and precision of the trained model.

3.2 Problem Definition

Vehicle Detection and Vehicle Trajectory analysis

- Past work has been done to develop models that carry out road traffic monitoring method based on image processing
- Deep learning models can speed up the task of object detection and tracking.
- Traffic analysis also includes counting the number of vehicles based on their trajectory i.e the direction of their movement.
- The task is divided into two parts:
  - **Object Detection**: On the road we can find different types of vehicles for example Car, truck, bus. Some other type of vehicles that can be found on Indian roads for example a rickshaw, bullock cart.
  - **Object Tracking**: Track the trajectory of the above detected vehicles.

Chapter 4

Background Material

4.1 Object Detection

You Only Look Once (YOLO) is a state-of-the-art, real-time object detection algorithm introduced in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in their famous research paper "You Only Look Once: Unified, Real-Time Object Detection".

Prior detection systems employ classifiers or localizers to carry out detection. They apply the model to an image at multiple locations and scales. The image's high-scoring areas are referred to as detections. YOLO takes a very different strategy. The image is divided into regions by this network, which also predicts bounding boxes and probabilities for each region. The projected probabilities are used to weight these bounding boxes.YOLO proposes the use of an end-to-end neural network.

Real-time object detection is ingrained in autonomous vehicle systems from the beginning. When compared to straightforward image segmentation methods, YOLO is a better contender due to its real-time feature. This feature enables us to detect and track vehicles for traffic surveillance.

4.1.1 YOLO architecture

In YOLO we have to look only once in the network i.e. only one forward pass is required through the network to make the final predictions. The architecture works as follows:

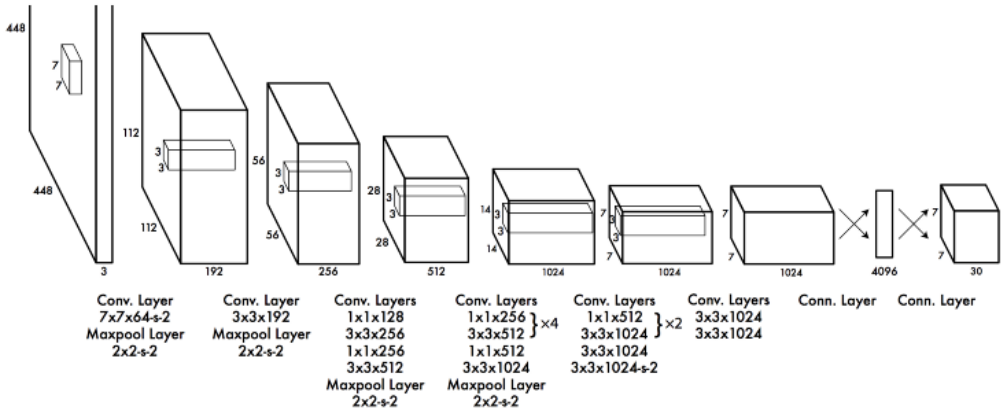

FIGURE 4.1: YOLO Architecture

- Resizes the input image into 448x448 before going through the convolutional network.
- This model has 24 convolution layers, 4 max-pooling layers followed by 2 fully connected layers.
- For reduction of layers, a 1x1 convolution is first applied which is then followed by a 3x3 convolution to generate a cuboidal output.
- The activation function under the hood is ReLU, except for the final layer, which uses a linear activation function
- Some additional techniques, such as batch normalization and dropout, respectively regularize the model and prevent it from overfitting.

4.1.1.1 Residual blocks

This first step starts by dividing the original image (A) into NxN grid cells of equal shape, where N in our case is 4 shown on the image on the right. Each cell in the

grid is responsible for localizing and predicting the class of the object that it covers, along with the probability/confidence value.

4.1.1.2 Bounding box regression

The next step is to determine the bounding boxes which correspond to rectangles highlighting all the objects in the image. We can have as many bounding boxes as there are objects within a given image. YOLO determines the attributes of these bounding boxes using a single regression module in the following format, where Y is the final vector representation for each bounding box.

$$Y = [p_c, b_x, b_y, b_h, b_w, c_1, c_2]$$

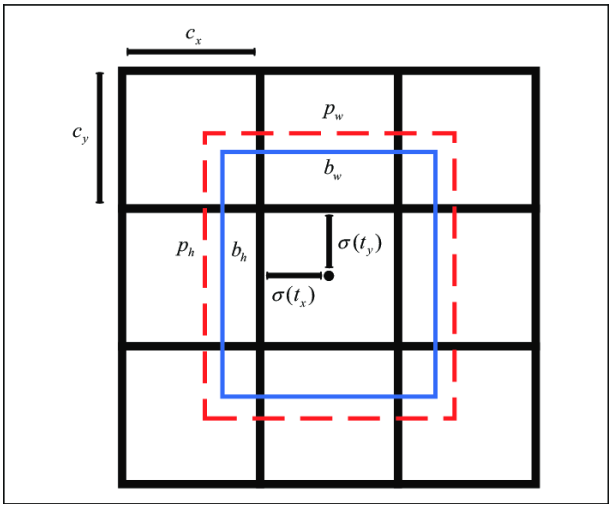

FIGURE 4.2: Bounding box regression

- p_c corresponds to the probability score of the grid containing an object
- b_x, b_y are the x and y coordinates of the center of the bounding box with respect to the enveloping grid cell.
- b_h, b_w correspond to the height and the width of the bounding box with respect to the enveloping grid cell
- c_1 and c_2 correspond to the two classes. We can have as many classes as our use case requires.

4.1.1.3 Intersection Over Unions or IOU

Most of the time, a single object in an image can have multiple grid box candidates for prediction, even though not all of them are relevant. The goal of the IOU (a value between 0 and 1) is to discard such grid boxes to only keep those that are relevant. Here is the logic behind it:

- The user defines its IOU selection threshold, which can be, for instance, 0.5.
- Then YOLO computes the IOU of each grid cell which is the Intersection area divided by the Union Area.
- At last, it ignores the prediction of the grid cells having an IOU $\leq$ threshold and considers those with an IOU $>$ threshold.

FIGURE 4.3: An example of computing Intersection over Unions for various bounding boxes.

4.1.1.4 Non-Max Suppression or NMS

Setting a threshold for the IOU is not always enough because an object can have multiple boxes with IOU beyond the threshold, and leaving all those boxes might include noise. Here is where we use NMS to keep only the boxes with the highest probability score of detection.

4.1.2 Advantages of YOLO

- Speed : YOLO is extremely fast because it does not deal with complex pipelines. It can process images at 45 Frames Per Second (FPS). In addition,

YOLO reaches more than twice the mean Average Precision (mAP) compared to other real-time systems, which makes it a great candidate for real-time processing.

- Detection accuracy : YOLO is far beyond other state-of-the-art models in accuracy with very few background errors.
- Good generalization : YOLO pushed a little further by providing a better generalization for new domains, which makes it great for applications relying on fast and robust object detection
- Open-source : Making YOLO open-source led the community to constantly improve the model. This is one of the reasons why YOLO has made so many improvements in such a limited time.

4.2 Object Tracking

In object detection, we detect an object in a frame, put a bounding box or a mask around it and classify the object. Now, an object tracker on the other hand needs to track a particular object across the entire video.

4.2.1 Simple Online Realtime Tracking (SORT)

A visual multiple-object tracking system based on rudimentary data association and state estimate algorithms is implemented simply with SORT. It is designed for online tracking applications where only past and current frames are available and the method produces object identities on the fly. SORT is made of 4 key components which are as follows:

- Detection: This is the first step in the tracking module. In this step, an object detector detects the objects in the frame that are to be tracked. YOLO is used here.

- Estimation: Using a constant velocity model, we propagate the detections from the current frame to the next frame to estimate the target's position. The detected bounding box is used to update the target state when detection is linked to a target. The Kalman filter framework is used to solve the velocity components optimally.

- Data association: We now have the target bounding box and the detected bounding box. The intersection-over-union (IOU) distance between each detection and all projected bounding boxes from the existing targets is therefore calculated to create a cost matrix. The Hungarian algorithm is used to solve the assignment in the best possible way. The assignment is refused if the IOU of detection and target is below a predetermined threshold known as IOUmin.

- Track Identity Creation and Removal: This unit is in charge of ID creation and deletion. According to the IOUmin, unique identities are produced and discarded. If the overlap of detection and target is less than IOUmin then it signifies the untracked object. If TLost frames are not recognized for a track, it will be terminated. We can specify how many frames should count as TLost. If an object reappears, tracking will automatically start again with a new identity.

4.2.2 DeepSORT

The SORT algorithm has been expanded by DeepSORT which tracks objects by giving a unique ID to each one. By including an appearance descriptor in the SORT algorithm, DeepSORT integrates deep learning to decrease identity switches.

In terms of tracking accuracy and precision, SORT works well. However, SORT fails in cases of occlusion and returns tracks with a significant number of ID switches. The association matrix that was used is responsible for this. A more effective association metric is used by DeepSORT, which integrates motion and appearance descriptors. DeepSORT is a tracking algorithm that follows things not only based on their motion and velocity but also on their appearance.

Let us consider a bounding box represented by the image coordinates of its center(u_{BB}, v_{BB}) and its height and width (h_{BB}, v_{BB}), the detection set D (with N bounding boxes),

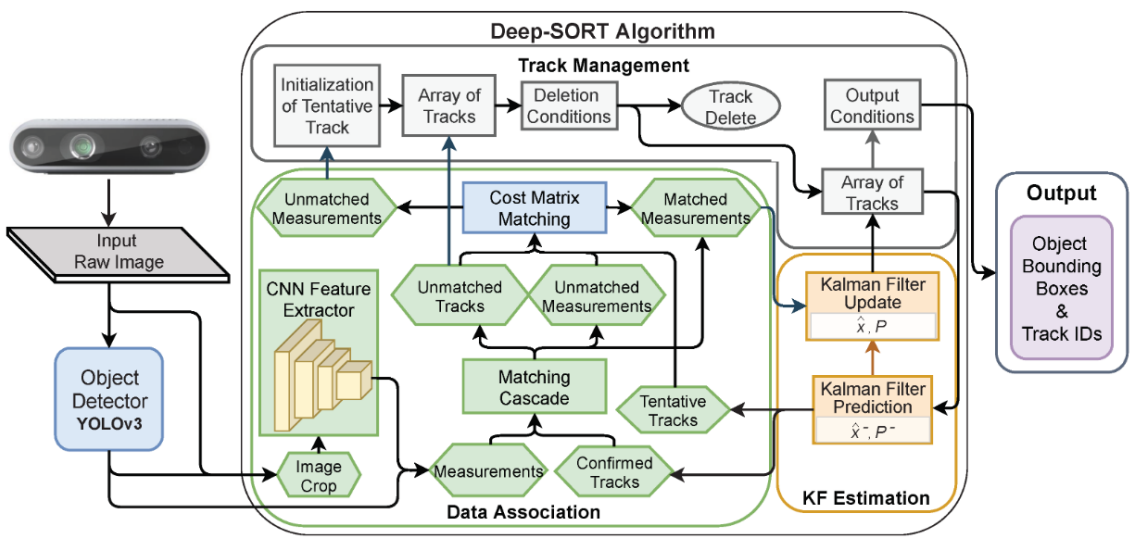

FIGURE 4.4: Overview of the object tracking Deep-SORT algorithm

and the prediction set P (with M bounding boxes). The following cost matrix formulations are proposed:

- Euclidean distance-based cost matrix $D_E(D, P)$:

$$D_E(D,P) = \begin{bmatrix} d(D_1,P_1) & \dots & d(D_1,P_M) \\ d(D_2,P_1) & \dots & d(D_2,P_M) \\ \vdots & \ddots & \vdots \\ d(D_N,P_1) & \dots & d(D_N,P_M) \end{bmatrix}$$

  This represents the distance between bounding box central points normalized into half of the image dimension.

  To formulate the problem as a maximization problem, to be solved using the Hungarian algorithm, the distance is obtained by the difference between the following and the normalized Euclidean distance.

$$IoU(D,P) = \begin{bmatrix} iou(D_1,P_1) & \dots & iou(D_1,P_M) \\ iou(D_2,P_1) & \dots & iou(D_2,P_M) \\ \vdots & \ddots & \vdots \\ iou(D_N,P_1) & \dots & iou(D_N,P_M) \end{bmatrix}$$

The normalized Euclidean distance is given below:

$$d(D_i, P_i) = 1 - \frac{\sqrt{(u_{D_i} - u_{P_i})^2 + (v_{D_i} - v_{P_i})^2}}{\frac{1}{2}\sqrt{h^2 + w^2}}$$

where (h, w) are the height and width of the input image, D_i is a bounding box from the detection set, and P_i is a bounding box from the prediction set.

- Bounding box ratio based cost matrix (R(D,P))—implemented as a ratio between the product of each width and height:

$$R(D, P) = \begin{bmatrix} r(D_1, P_1) & \dots & r(D_1, P_M) \\ r(D_2, P_1) & \dots & r(D_2, P_M) \\ \vdots & \ddots & \vdots \\ r(D_N, P_1) & \dots & r(D_N, P_M) \end{bmatrix}$$

$$r(D_i, P_i) = min(\frac{w_{D_i} h_{D_i}}{w_{P_i} h_{P_i}}, \frac{w_{P_i} h_{P_i}}{w_{D_i} h_{D_i}})$$

This metric outcome with a value closer to 1 compares values close to 0 or much larger than 1 otherwise for boxes with similar shapes. To obtain a result that falls within the [0,1] range, the minimum between the bounding box ratio and its inverse is used.

- SORT's IoU cost matrix combined with the Euclidean distance cost matrix:

$$E_D^{IoU}(D, P) = IoU(D, P) \circ D_E(D, P)$$

where $\circ$ represents the Hadamard product (element-wise product) between two matrices.

- SORT's IoU cost matrix combined with the box ratio based cost matrix:

$$R^{IoU}(D, P) = IoU(D, P) \circ R(D, P)$$

- Euclidean distance cost matrix combined with the box ratio-based cost matrix:

$$R^{D_E}(D, P) = D_E(D, P) \circ R(D, P)$$

- SORT's IoU cost matrix combined with the Euclidean distance cost matrix and the box ratio based cost matrix:

$$M(D,P) = IoU(D,P) \circ D_E(D,P) \circ R(D,P)$$

- Element-wise average of every cost matrix (A(D, P)):

$$A(D_i,P_i) = \frac{IoU(D_i,D_j)+D_E(D_i,D_j)+R(D_i,D_j)}{3}, i \in D, j \in P$$

- Element-wise weighted mean of every cost matrix value:

$$W_M(D_i,P_i) = \lambda_{IoU} \cdot IoU(D_i,P_i) + \lambda_{D_E} \cdot \quad D_E(D_i,P_i) + \lambda_R \cdot R(D_i,P_i), \\ i \in D, j \in P, \lambda_{IoU} + \lambda_{D_E} + \lambda_R = 1$$

To improve tracking performance in multi-class environments, cost matrices can be updated based on the match between predicted and detected object classes.

Chapter 5

Methodology

5.1 Training of Model

The YOLOv5 model is given the fine-tuned weights from the AIC-HCMC-2020 dataset.

Model	Image Size	Precision	Recall	MAP@0.5	MAP@0.5-0.95
YOLOv5m	1024x1024	0.89626	0.91098	0.94711	0.66816

5.2 Preprocessing of Raw Dataset

The input videos were pre-processed by first converting the videos from MTS format to MP4.This was done for two reasons.

- OpenCV does not support reading video files in the MTS format.
- The input video dataset is of 18.6 GB which poses some trouble while processing them by the kernel

The audio component in the videos is entirely unnecessary while detecting and tracking the vehicles. Thus, the audio was removed while converting the videos to MP4

format. Also, the bitrate of the videos was reduced to increase the compression ratio. This was done as a minor reduction in the high-frequency features of the frames in the video will not adversely affect the detection accuracy of the YOLOv5 object detector.

5.3 Configurations for the input dataset

After the pre-processing of the input video, the configurations of the object detector model were tuned. Firstly, The Region of Interest was defined. For finding the coordinates of the boundary points on the image frame, OpenCV functions were used to map the locations of the mouse pointer click to the coordinates within the image. We find 4 points which define a polygon (in our case a rectangle) in anti clockwise direction. This defines the area in which the model will detect the vehicles.

Secondly, after defining the region of interest, the direction vectors were defined using the same method as above. Here two points are needed to define a direction vector. Four direction vectors were defined by aligning the vectors in accordance with the inclination of the roads in the videos. The four directions are comprised of two directions in each vertical and horizontal direction.

Now For direction tracking of the vehicles, the DeepSORT model is used. The direction which is detected by the Tracking algorithm is matched with the predefined direction vectors. Then the best matching direction is marked as the direction of movement of the detected vehicle.

5.4 Data set Explaination

The data set consists of traffic videos captured using a Camcorder. The video consists of different vehicles moving in any particular direction. Our aim is to classify these vehicles and their direction.

The required class labels were defined. Four categories of vehicles used for classification: Two-Wheelers, Four-wheelers, Bus, and Truck. Four direction vectors were defined by aligning the vectors in accordance with the inclination of the roads in the videos.

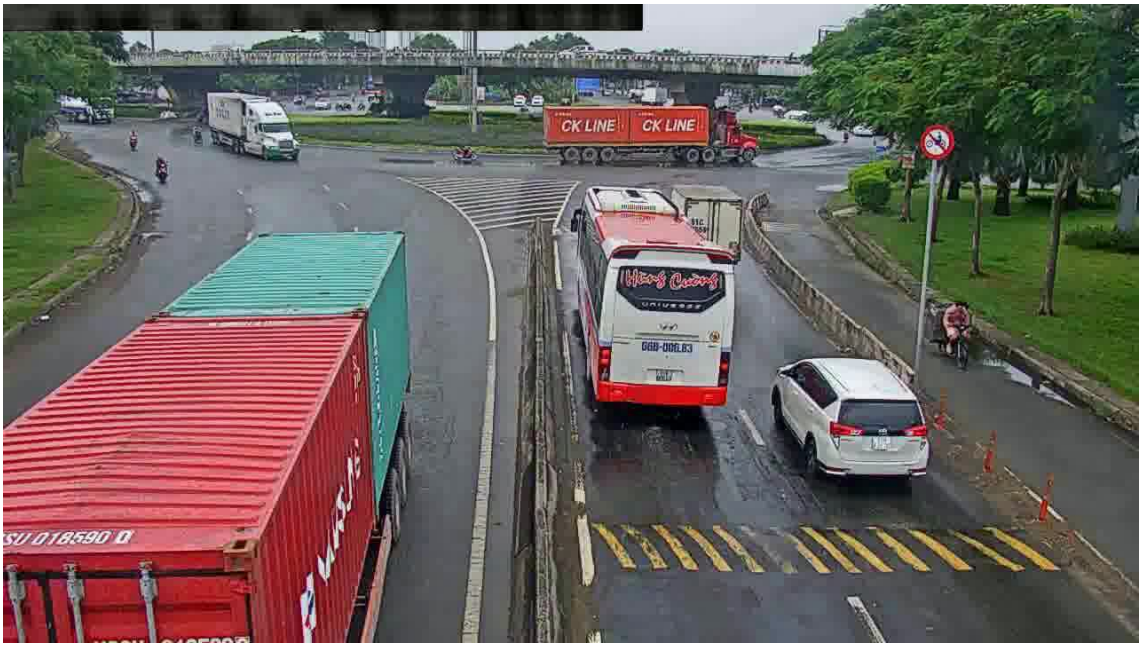

FIGURE 5.1: One of the sample images of AIC-HCMC-2020 dataset

FIGURE 5.2: A sample frame from a video of the given dataset

5.5 Output format

A CSV file is also produced in the following format. This shows the vehicles detected in each frame with their corresponding locations and classes with the following headings:

- **track-id:** the id of the object
- **frame-id:** the current frame
- **box:** the box wraps around the object in the corresponding frame
- **color:** the color which is used to visualize the object
- **direction:** the direction of the object
- **fpoint, lpoint:** first/last coordinate where the object appears
- **fframe, lframe:** first/last frame where the object appears

Chapter 6

Experimental Results

6.1 Deliverables

6.1.1 Procedure

The following steps were followed :

- **Load the model** Activate Google Colab runtime envrionment with GPU. Upload the python code files into the drive.
- Check the state and presence of the GPU using the command **nvidia-smi**
- Install the required libraries by using the command **pip install -r requirements.txt**
- Download the pre-trained weights of the YOLOv5 model from the given link in the code file using the **gdown** library
- **Preparing the inputs:** Upload the Input Video and json file with specifications for the **Region of Interest** and the **direction vectors**. The name of the video file and the json file should be the same.
- Run the code using the command **python run.py** with the arguments: –input_path(give location of the input video file) , –output_path (give location of the output folder) , –weight(give the location of the downloaded weights)

6.1.2 Results

6.1.2.1 Output video with Detected Objects

Class 0: Two-wheelers. Class 1: Four-wheelers. Class 2: Bus. Class 3: Truck

The bounding boxes are drawn around the detected vehicles with a specific ID which is given to each uniquely detected new vehicle.

FIGURE 6.1: Detected vehicles in a frame of Video 00121

FIGURE 6.2: Detected vehicles in a frame of Video 00129

6.1.2.2 Output CSV File with Detections

The CSV file which is produced as output has information about the vehicles detected in each of the frames,their direction,their class, the location of their bounding box The exact format of the table has been described earlier.

track_id	frame_id	box	color	label	direction	fpoint	lpoint	fframe	lframe
1	90	[0, 539, 62, 649]	(204, 50, 153)	0	1	(31.5, 593.0)	(31.0, 594.0)	82	228
1	228	[0, 539, 62, 649]	(204, 50, 153)	0	1	(31.5, 593.0)	(31.0, 594.0)	82	228
2	239	[662, 597, 702, 668]	(238, 104, 123)	0	1	(682.0, 632.5)	(60.5, 854.5)	239	460
2	245	[659, 598, 701, 670]	(238, 104, 123)	0	1	(682.0, 632.5)	(60.5, 854.5)	239	460

6.1.2.3 Analysis of the entire video dataset

The following is the analysis of the number of vehicles detected in each of the input videos.

Direction 1 (Far to Near)

Video:	vid121	vid122	vid123	vid124	vid125	vid126	vid129	vid130	vid131	vid132	Summation label wise
label	count	count	count	count	count	count	count	count	count	count	
0	42	47	41	42	37	22	31	50	82	72	466
1	42	35	33	36	26	12	53	64	48	56	405
2	26	23	18	24	15	9	7	6	12	16	156
3	65	63	49	70	48	29	83	73	75	74	629
										Total vehicles=	**1656**

Direction 2 (Right to Left)

Video:	vid121	vid122	vid123	vid124	vid125	vid126	vid129	vid130	vid131	vid132	Summation label wise
label	count	count	count	count	count	count	count	count	count	count	
0	42	47	41	42	37	22	31	50	82	72	466
1	42	35	33	36	26	12	53	64	48	56	405
2	26	23	18	24	15	9	7	6	12	16	156
3	65	63	49	70	48	29	83	73	75	74	629
										Total vehicles=	**1656**

Direction 3 (Near to Far)

Video:	vid121	vid122	vid123	vid124	vid125	vid126	vid129	vid130	vid131	vid132	Summation label wise
label	count	count	count	count	count	count	count	count	count	count	
0	34	21	16	16	24	10	51	36	63	77	348
1	29	23	13	32	21	8	29	48	44	53	300
2	10	5	6	15	11	6	8	9	17	19	106
3	33	24	35	30	30	16	46	42	77	67	400
										Total vehicles=	**1154**

Direction 4 (Left to Right)

Video:	vid121	vid122	vid123	vid124	vid125	vid126	vid129	vid130	vid131	vid132	Summation label wise
label	count	count	count	count	count	count	count	count	count	count	
0	38	33	11	10	15	4	45	22	31	45	254
1	9	1	5	3	15	2	13	8	11	10	77
2	5	1	1	4	2	1	3	2	4	5	28
3	21	29	30	33	20	7	30	29	33	29	261
										Total vehicles=	**620**

6.1.3 Combined Vehicle Count for each category of vehicle

Label	**Vehicle count considering all directions**
Two-wheel	1383
Four-Wheel	879
Bus	384
Truck	1841
Total Vehicles =	**4487**

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Therefore, we reviewed some of the past work from other people in the domain of multiple object detection and trained a state of the art model, YOLOv5 for object detection and DeepSORT for tracking of the detected object. We saw that SORT performs very well in terms of tracking precision and accuracy. But SORT fails in case of occlusion. DeepSORT solves this problem as it uses a better association metric that combines both motion and appearance descriptors. Cosine distance is a metric that helps the model recover identities in case of long-term occlusion and motion estimation also fails. Using these simple things can make the tracker even more powerful and accurate

7.2 Future Work

The BTP work for this semester revolved around detection of different types of vehicles, counting the number of vehicles going in each direction along with tracking the path of the vehicle. Future work includes more work in the following domains:

- The object size of the vehicle changes greatly with viewing angle, and the detection accuracy of a small object far away from the road is low. In the face

of complex camera scenes, it is essential to effectively solve the above problems and further apply them.

- So far we have focused on the tracking of the vehicles, one step ahead from this would be predicting the path of the vehicles. This could have be a crucial step in autonomous vehicles research.

Chapter 8

References

[1] A. B. d. Oliveira and J. Scharcanski, "Vehicle Counting and Trajectory Detection Based on Particle Filtering," 2010 23rd SIBGRAPI Conference on Graphics, Patterns and Images, 2010, pp. 376-383, doi: 10.1109/SIBGRAPI.2010.57.

[2] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[3] X. Hou, Y. Wang and L. -P. Chau, "Vehicle Tracking Using Deep SORT with Low Confidence Track Filtering," 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2019, pp. 1-6, doi: 10.1109/AVSS.2019.8909903.

[4] Nguyen Tien Hunng (2020). Vehicle Counting AIC HCMC 2020, Version 1. Retrieved July 20, 2022 from:
`https://www.kaggle.com/datasets/hungkhoi/vehicle-counting-aic-hcmc-2020/versions/1`