IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

Indian Institute of Engineering Science and Technology, Shibpur
An Institute of National Importance
भारतीय अभियांत्रिकी विज्ञान एवं प्रौद्योगिकी संस्थान, शिवपुर

Speed Estimation of a moving vehicle from video stream using Machine Learning Techniques

By

Aneek Ghosh(510818039)

Ankan Halder(510818008)

Pranjal Sharma(510818089)

Guided by **Dr. Arindam Biswas**

Department of ***Information Technology*** (8th semester)
IIEST, Shibpur

Motivation & Objective

- **Rash driving** and accidents are increasing.
- To decrease accidents, speed should be **controlled**.
- **Detection of speed** is a necessity.
- Detection of the vehicles using **machine learning techniques**.
- **Tracking** of the vehicle in the video frame.
- **Speed Estimation** of the vehicle.

Content

Project Overview

- We have divided this project into three parts.

    1. Vehicle Detection

    2. Vehicle Tracking

    3. Speed Estimation

- We covered vehicle detection in 7th semester and the rest of the parts have been completed in this semester.

Tools and Technologies

- ➢ We have used the OpenCV library for **preprocessing** the input stream and loading our model into the system.
- ➢ The Machine Learning model we have used in this project is **YOLOv3** which is a faster and lighter model used for object detection.
- ➢ After detection of the vehicles ,we have **tracked** the vehicles in the video stream using openCV library.
- ➢ At last using an approximated mathematical formula we have **estimated** the vehicle speed.

About OpenCV

- ***Open Source Computer Vision***

- Computer vision is a process by which we can **manipulate** and **retrieve** data from images and videos.

- OpenCV is an open-source library for the computer vision, machine learning, and image processing.

About YOLOv3

- YOLO, ***You Only Look Once*** is a real-time object detection algorithm.
- YOLO uses features learned by a ***deep convolutional neural network*** to detect an object.
- YOLO has the advantage of being much **faster** than other networks and still maintains **accuracy**.

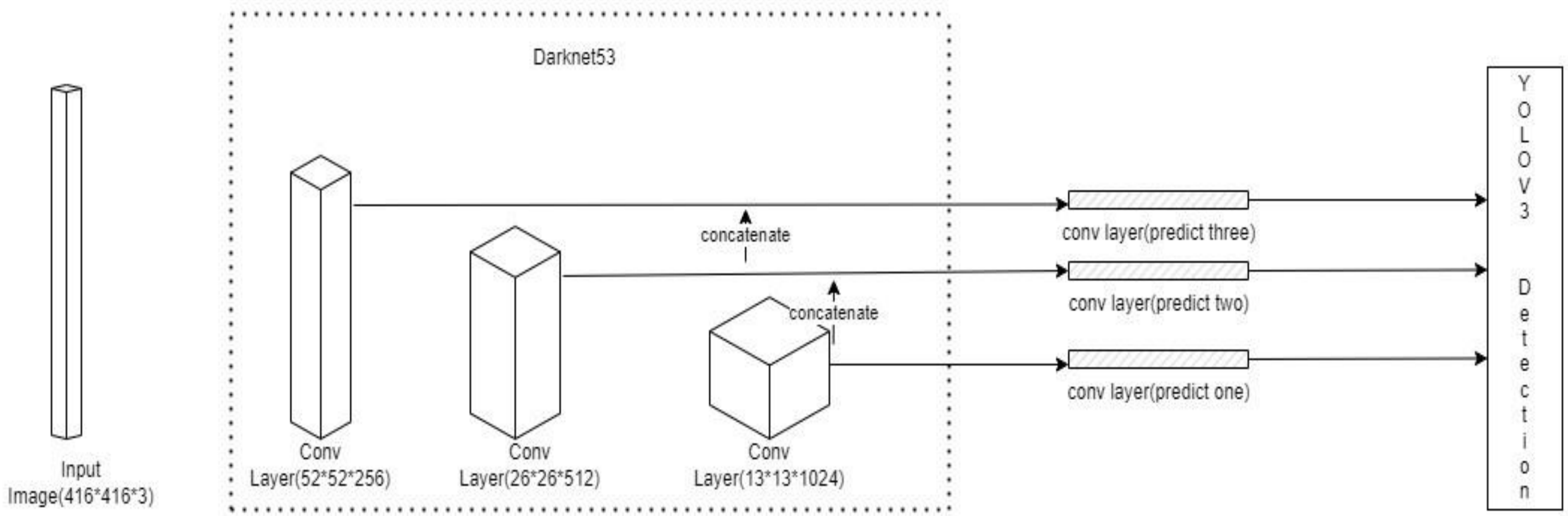
About YOLOv3 (Comparison with other Models)

Model	mAP-50	FPS
SSD321	45.4	16
DSSD321	46.1	12
R-FCN	51.9	12
SSD513	50.4	8
DSSD513	53.3	6
Retinanet-50	50.9	14
YOLOv3-416	55.3	35

YOLOv3 architecture

- ➢ YOLOv3 uses a variant of **Darknet**, a framework to train neural networks, which originally has 53 layers.
- ➢ Another 53 layers are stacked onto it, accumulating to a total of a **106-layer** fully convolutional architecture. The layers include Residual blocks,Upsampling layer, Detection layer.
- ➢ In the convolutional layer, **1 x 1 kernel** is applied to the input.
- ➢ The algorithm makes predictions at **three** different scales.

YOLOv3 Architecture



YOLOv3 algorithm

- The YOLOv3 algorithm first separates an image into a **grid**.
- The output contains the **label, probability** and **bounding boxes** for objects in the image.
- The cell which contains the center of the **ground truth box** of an object is responsible for predicting the object.
- Each bounding box is represented by **6 numbers (pc,bx,by,bh,bw,c)** where c is a vector containing the scores of 80 classes.

YOLOv3 Architecture

box1 =

P_c	b_x	b_y	b_h	b_w	c_1	c_2	c_3	...	...	...	...	c_{78}	c_{79}	c_{80}

$$\text{Scores} = p_c \; * \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{78} \\ c_{79} \\ c_{80} \end{pmatrix} = \begin{pmatrix} p_c c_1 \\ p_c c_2 \\ p_c c_3 \\ \vdots \\ p_c c_{78} \\ p_c c_{79} \\ p_c c_{80} \end{pmatrix} = \begin{pmatrix} 0.12 \\ 0.13 \\ 0.44 \\ \vdots \\ 0.07 \\ 0.01 \\ 0.09 \end{pmatrix}$$

find the max

score = 0.44
box = (b_x, b_y, b_h, b_w)
class = c = 3 ("car")

Box (b_x, b_y, b_h, b_w) has detected c= 3 ("car") with probability score: 0.44

The class with the highest score will be the predicted result.

Vehicle detection

- The video stream is **read** and **processed** by cv2 library.
- We have loaded the YOLOv3 model using **dnn** library of cv2 .
- We store the **confidence score** and **coordinates of bounding boxes** of each of the vehicle that is detected.
- Next, for each of the detection we have **drawn** the bounding boxes and confidence scores on the stream using cv2.

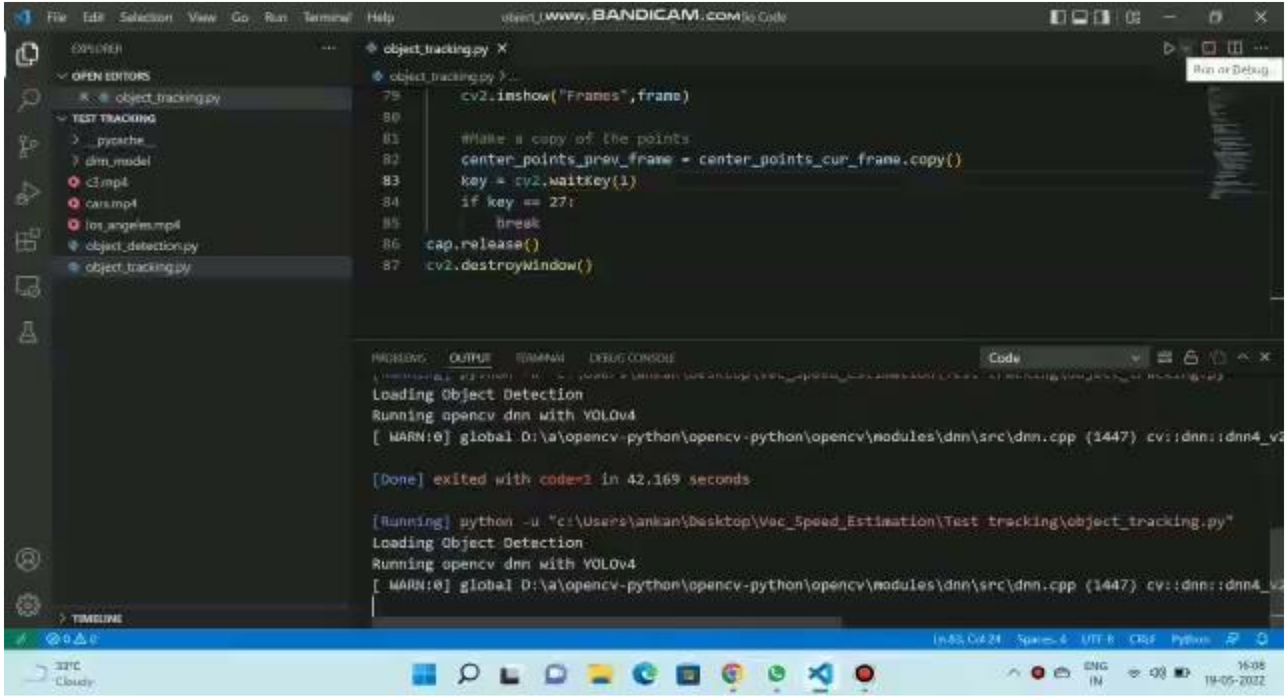
Vehicle tracking

- In the last module we **detected** vehicles in each frame of the video stream.
- We keep track of the **centroid** of each vehicle in an array.
- For each frame change we calculate the **change in the position** of the centroids of the current frame and previous frame.

Vehicle tracking (cont...)

- We **assign or retain a tracking id** to each vehicle if the change in position is less than our assumed threshold value.
- If the change in position is **greater** than our assumed threshold value then the tracking id is **destroyed**.
- In that case a **new tracking id** is assigned to the vehicle if it has not left the video stream.

Vehicle tracking (cont...)



Speed Estimation

- ➢ Till now we have detected and tracked the vehicles in each frame.
- ➢ Now for the speed estimation part, we have used an approximated **mathematical formula**.
- ➢ We assumed the **width** of the vehicles to be **1m** and from the detection output we calculate the **pixels** in horizontal distance.
- ➢ This gives the value of **pixels per meter** (**ppm**).

Speed Estimation(Cont..)

- For each frame change we find the distance covered by the centroid of each vehicle in **pixels (d_pixels)**.
- We divide the d_pixels by ppm to get the **distance in meters** (d_meters).
- Then we multiply d_meters with **fps** of the video which gives us the **distance covered in one second** by each centroid.
- So, we have the speed of each vehicle in meters/second and multiplying it by **3.6** we get the **speed** in **kilometers/hour**.

Estimated speed(Km/hr) = Distance (meters) * FPS * 3.6

Validation of Estimated Speed

We have taken a test case where the vehicle was moving at a speed of around 15 to 20 km/hr and our output is approximately 12 to 14 km/hr which is quite close to the actual speed.

Conclusion

- ➢ This module consists of YOLOv3 as the backbone.
- ➢ It uses features learned by a deep convolutional neural network to detect an object.
- ➢ We have used openCV library for object tracking and speed estimation purpose.
- ➢ At last we have validated the output against a real life test case.

References

- https://pjreddie.com/media/files/papers/YOLOv3.pdf
- https://opencv.org
- https://github.com/opencv
- https://towardsdatascience.com/yolo-v3-explained-ff5b850390f

Thank you!