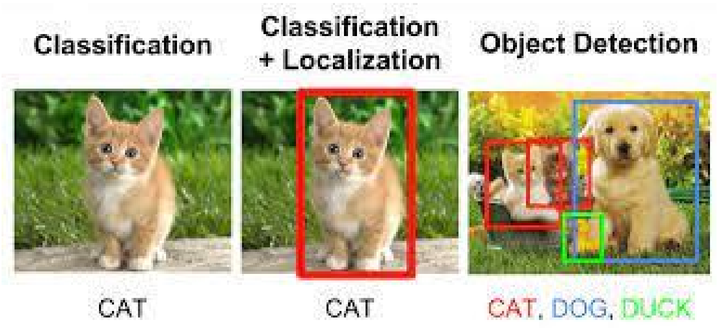
VR MINI PROJECT REPORT

Q3d: Object Detection(Faster RCNN + YOLO V2) & Object Tracking(SORT + Deep SORT)

Ans:

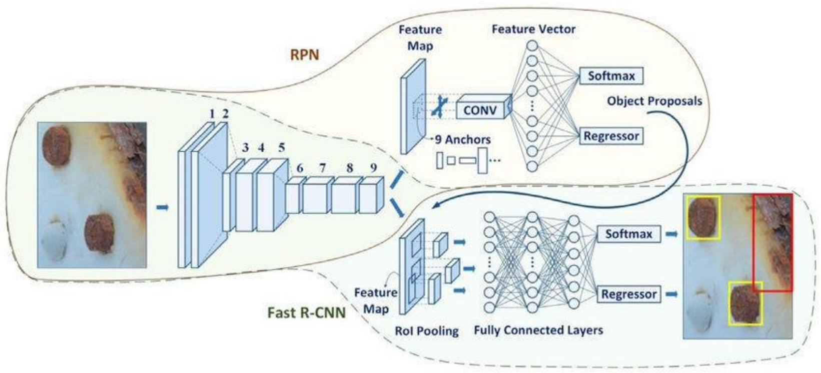
<u>Object Detection</u>: Object detection is a computer vision technique that involves detecting and locating objects within an image or video. The goal of object detection is to identify and localize objects within an image or video and provide information about their class and position. This can be used for a variety of tasks, including surveillance, autonomous driving, image and video analysis, and robotics.



Here in this project for object detection we used two specific methods,namely **Faster RCNN(Region Proposal Based) & YOLO(Dense Sampling Based).**

<u>Faster RCNN</u>:Faster R-CNN is a popular object detection algorithm that was introduced by Ross Girshick in 2015. It is an extension of the earlier R-CNN and Fast R-CNN algorithms and is considered to be one of the state-of-the-art object detection models.

Faster R-CNN is a two-stage detection algorithm, based on a region proposal network (RPN) that shares convolutional layers with the detection network. The RPN generates region proposals by sliding a small network over the convolutional feature map. The network predicts whether each anchor box should be classified as an object or background, and also generates refined bounding box coordinates for each proposal.



Architecture Of Faster RCNN

After generating region proposals, Faster R-CNN passes them through a detection network, which performs object classification and bounding box regression on the proposed regions. The detection network uses features generated by the shared convolutional layers and combines them with the proposals to classify and localize objects.

Compared to its predecessors,(R-CNN & Fast RCNN) Faster R-CNN is faster and more accurate, thanks to the use of a single, unified network for region proposal and object detection. This architecture also allows for end-to-end training, which further improves performance. As a result, Faster R-CNN has become a popular choice for a wide range of applications, including autonomous vehicles, robotics, and surveillance systems.

For being a two stage detection framework, faster RCNN is more robust in performance but slower.

YOLO: YOLO (You Only Look Once) is a one-stage state-of-the-art real-time object detection system. The first version of YOLO was released in 2015 and quickly gained popularity due to its high speed and accuracy.YOLOv2 was released in 2016 and improved upon the original model by incorporating batch normalization, anchor boxes, and dimension clusters. YOLOv3 was released in 2018 and further improved the model's performance by using a more efficient backbone network, adding a feature pyramid, and making use of focal loss.In 2020, YOLOv4 was released which introduced a number of innovations such as the use of Mosaic data augmentation, a new anchor-free detection head, and a new loss function.

In 2021, Ultralytics released YOLOv5, which further improved the model's performance and added new features such as support for panoptic segmentation and object tracking.

YOLO uses a single neural network that simultaneously predicts the bounding boxes and class probabilities for each object in an image by integrating region proposals and detection by acting on a dense sampling of possible locations. It divides the input image into a grid of cells and predicts object bounding boxes and class probabilities for each cell.This means that YOLO can detect multiple objects in a single image in a single forward pass through the neural network, making it very fast.

For being a one-stage detection framework it is simple and fast but performance wise not as good as Region Proposal-based methods like Faster R-CNN.

Object Tracking :

Object tracking is a computer vision technique that involves identifying and following the movement of objects in a video sequence or a series of images. The goal of object tracking is to locate and track the position, orientation, and motion of one or multiple objects over time, even as they move across the camera's field of view or change appearance due to lighting, occlusion, or other factors.Object tracking is a critical component in many applications, including surveillance, robotics, autonomous vehicles, human-computer interaction, and sports analysis, among others.The prerequisite of object tracking is object detection. Tracking can be accomplished using a variety of methods. Here in this project we basically used two methods **SORT and DeepSORT.**

SORT : SORT (Simple Online Realtime Tracking) is a popular algorithm for multi-object tracking in videos. SORT is a simple and efficient algorithm that is capable of handling large numbers of objects in real-time. The SORT algorithm first detects objects in each frame of the video sequence using an object detector. Then, it assigns a unique identity to each object based on its location and appearance. Then, it uses a Kalman filter to predict the location of each object in the next frame, and then associates the predicted locations with the detected objects to track them over time if the overlap (IOU) between predictions (based on last frame) and actual detections crosses a certain thresold.Finally, a matrix holds the IOUs and a Hungarian algorithm associates tracks and detections.When a new detection has an IOU below a specific threshold it is not assigned to an existing track but classified as a new object. Tracks are removed when no detection is assigned for a certain number of frames. That helps to avoid unbound growth.

DeepSORT: Deep Learning based SORT is an extension of SORT that incorporates deep learning techniques to improve the tracking performance.The authors of DeepSORT argue that SORT creates too many "identity switches" when the sight of objects is blocked. Therefore, they propose to enrich the motion model (Kalman filter) with a deep learning component that incorporates the visual features of an object.DeepSORT uses a deep neural network to extract feature embeddings of each object in the video sequence, which are then used to associate objects across frames. The feature embeddings are learned using a Siamese network that is trained to distinguish between objects based on their appearance. DeepSORT also uses a gating mechanism to filter out noisy detections and improve the accuracy of the tracking. The combination of SORT and deep learning techniques in DeepSORT has shown to improve the tracking performance significantly in complex scenarios, such as occlusions, crowded scenes, and long-term tracking.

The main differences between SORT and DeepSORT are:

- **Feature extraction**: SORT uses hand-crafted features, such as color histograms, HOG features, and deep features, to represent objects. In contrast, DeepSORT uses a deep neural network to extract feature embeddings that capture more fine-grained details of the object's appearance.
- **Data association**: SORT uses a simple distance-based matching approach to associate object detections across frames, whereas DeepSORT uses a learned metric to associate objects based on their feature embeddings.
- **Robustness**: DeepSORT is generally more robust than SORT because it can handle occlusions, re-appearances, and appearance changes more effectively.
- **Accuracy**: DeepSORT is generally more accurate than SORT because it can capture more subtle differences in object appearance and can better handle cluttered scenes.
- **Computational complexity**: DeepSORT is more computationally expensive than SORT because it involves training a deep neural network, which requires significant computational resources.

Implementation Details:

In our project we used pretrained(on 80 class COCO dataset) Faster R-CNN from pytorch , which uses RESNET50 as a backbone for object detection. We combined it with SORT algorithm (official implementation in '**https://github.com/abewley/sort**') and DeepSORT (from **deep-sort-realtime** library). We applied them on 3 different videos taken by us in Electronic City.
We also used YOLO V2 for object detection. We downloaded the configuration file, pretrained weights(on coco dataset) and classes file (which is a .txt file containing 80 classes same as coco classes) from '**https://pjreddie.com/darknet/yolo/**' website and used OpenCV's DNN module to configure the YOLO V2 architecture ran inferences on our videos by extracting each frame at a time and prepared them as a blob using:

blob = cv2.dnn.blobFromImage(frame, scale, (416,416), (0,0,0), True, crop=False)
Where scale = 0.00392 or 1/255

and applying the algorithms on them. After detecting Objects we used SORT and DeepSORT to track the objects in our videos.While drawing bounding boxes on detected objects we used a thresold value of 0.8 to ignore weak detections.Even though we ignored weak detections, there will be lot of duplicate detections with overlapping bounding boxes. We used **Non-max suppression** to remove boxes with high overlapping using **cv2.dnn.NMSBoxes()** function **with nms thresold = 0.4** .

As per the requirement of the project we implemented **car counter** for our videos. At first we manually created **ground truths** (i.e. the no. of cars present in our videos) for our videos. Then we ran inference using above detection and tracking combinations . Using each combination we only detected and tracked cars in our videos and created a list which contains the all tracking ids of the various cars detected in our videos. Then we converted that list into a set in python and took the length of that set which basically tells us the no. of unique car ids or cars present in the video.
We also tried object detection using YOLOV3 and applied tracking DeepSORT on top of it.

❖ We attached all the ipython notebooks, which are self explanatory.
❖ We also attached input and output video files.

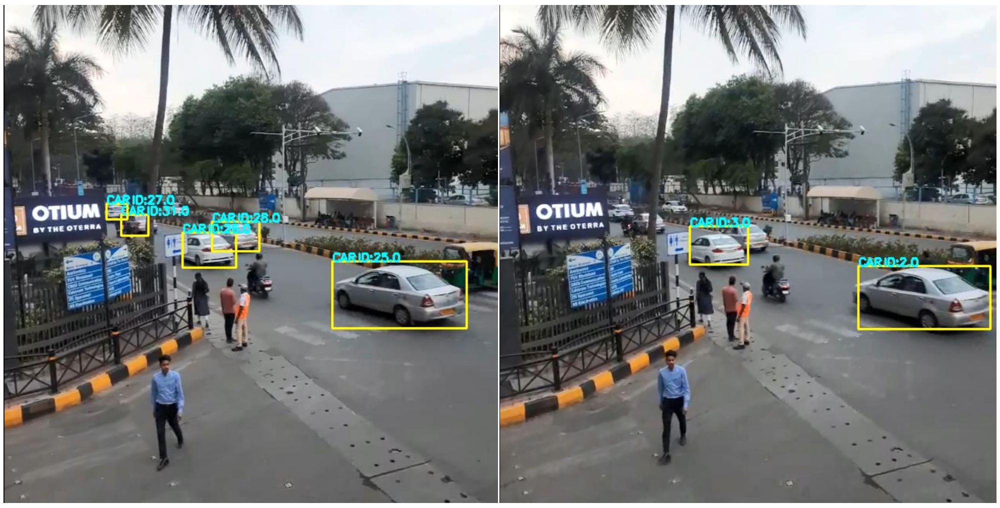
Results:

Input	length	FPS	Groud Truth (#car)	Faster R-CNN & SORT		Faster R-CNN & DeepSORT				YOLO V2 & SORT		YOLO V2 & DeepSORT			
				Inferen Time	Predicted #cars	InferenceTime n_iter		Predicted #cars n_iter		Inference Time	Predicted #cars	Inference Time n_iter		Predicted #cars n_iter	
						5	10	5	10			5	10	5	10
Video1	18s	30	16	24mins	52	44 m	26 m	59	32	4m23s	31	12m 25s	6m	36	24
Video2	15s	30	14	15m34s	41	22 m	15 m	47	36	1m21s	14	8m	4m4 5s	15	14
Video3	16s	30	5	22m58s	13	25 m	18 m	13	7	1m 37s	24	8m1 4s	5m	20	7

Observations:

1. We can see that Faster R-CNNs generally taking way longer time for inference than YOLO V2 based approaches. This is because YOLOv2 is a single-shot detector, meaning that it performs both object localization and classification in a single forward pass through the network. In contrast, Faster R-CNNs use a two-stage approach where region proposals are first generated before object classification and localization is performed. This two-stage process can be computationally expensive and result in slower processing times. So in most of the real time applications we prefer YOLO more than faster R-CNN.

2. On the other hand,Faster R-CNN is more accurate than YOLO, especially in scenarios where there are small objects or multiple objects in close proximity. In contrast, YOLO may struggle to accurately detect small objects or objects that are close together.



Car Detection With Faster R-CNN Car Detection With YOLO V2

Here we can see that Faster R-CNN can accurately detect small objects and objects that are close together, as the RPN can generate proposals at different scales and aspect ratios to capture a wide range of object sizes and shapes. Where as single stage object detection algorithm, YOLO struggles to detect small objects or objects that are close together, as it relies on the grid cells to capture the objects. Moreover, YOLO also has a fixed grid size, which can limit its ability to detect objects of different sizes.That’s why Faster R-CNN is more preferable when accuracy is the concern.

3. We can also see that deepsort based tracking approaches generally taking longer time than sort based ones for with same detection algorithms as DeepSORT involves training a deep neural network, which requires significant computational resources and time.

4. For video1 , we can see that faster R-CNN based approaches predict high values for no of cars like 52 and 59 where YOLO V2 based approaches predict lower values like 31 and 36 where ground truth is much smaller 16.As faster R-CNN's two-stage approach can lead to higher accuracy, but it can also be more prone to false positives, which could be one reason why the predicted number of cars is higher.On the other hand, the YOLO V2 based approaches are detecting fewer false positives but also missing some of the true positives which results into fewer car detections than faster RCNN.

5. Faster R-CNN + SORT predicts less cars (52) than faster R-CNN + DeepSort(59) method for video1. DeepSORT is an extension of SORT that adds deep appearance features to the tracking algorithm. It uses a deep neural network to extract appearance features of objects and uses these features to match detections across frames. This allows DeepSORT to handle more complex scenarios such as occlusions and track objects more accurately.Therefore, it's possible that the Faster R-CNN + DeepSORT method is able to detect and track more objects accurately than the Faster R-CNN + SORT method, leading to a higher predicted number of cars. **However, different hyperparameters and different implementations of these algorithms can affect their performance on this specific video.**Like in our DeepSort tracker we used max_age = 5 i.e. maximum no .of frames a object could disappear before reappearing on the frame with same id , making it a higher value could result into detection of less no of cars which discards the possibility of false positives but can also miss out true positives and n_init =1
i.e. minimum no. of frames a object has to appear before getting assigned a track_id. Making it lower can result into more unique car ids. When we made n_iter =10 we got no. of cars = 24 which is closer to ground truth for video 1 using YOLOV2 + DeepSort.

6. When our detectors and tracker faced occlusion multiple times (like in our first video) it resulted into assigning more unique ids to the same object appearing in different frames with a gap of frames, which resulted into huge no. of predicted cars in case of our video1 & video2. But in our video 3 there wasn't a lot of occlusion so the predicted no. of cars is lot closer to the ground truth.

Conclusion: We clearly see that tuning hyperparameters like max_age, n_iter of DeepSORT Tracker give us better result in terms of inference time and no. of predicted cars. Another importtant thing we noticed that when there is less occlusion in our video we get better result in terms of car counting.

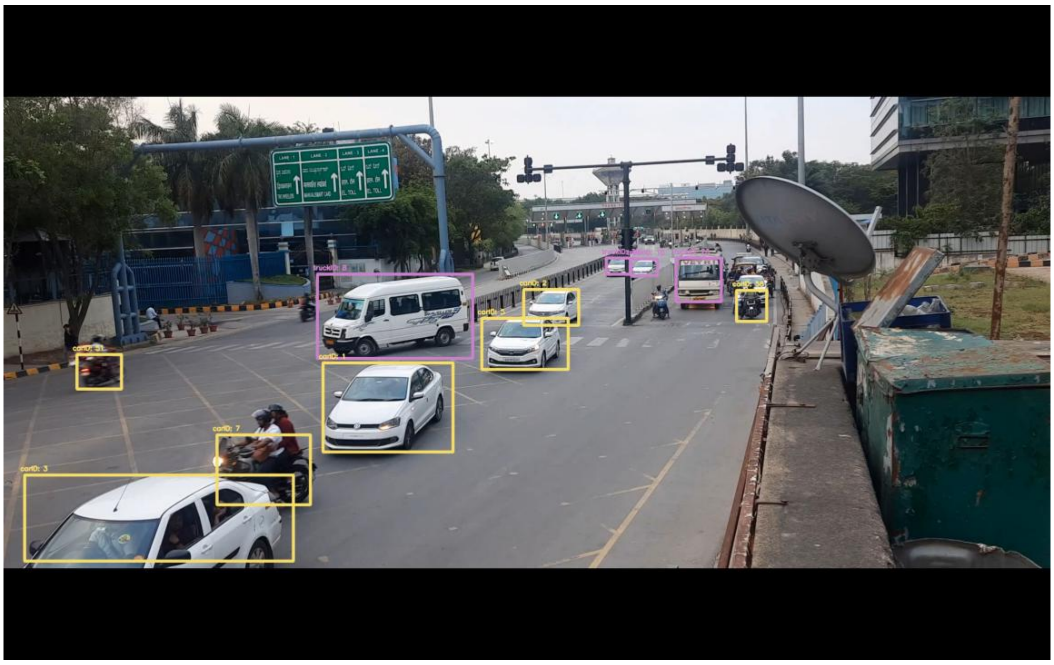
Today YOLO is the state of the art object detection algorithm and DeepSORT is the state of the art object tracking algorithm. Ultralytics YOLOv8 is the latest version of the YOLO (You Only Look Once) object detection and image segmentation model developed by Ultralytics. It provides backward compatibility to all the previous versions of YOLO. YOLOv8 include a new backbone network, a new anchor-free detection head, and a new loss function. YOLOv8 is also highly efficient and can be run on a variety of hardware platforms, from CPUs to GPUs.

So today the best possible combination of detection and tracking algorithm is as per our knowledge YOLO V8 + DeepSORT. We will explore that in future.

Link for i/p and o/p video files: Videos

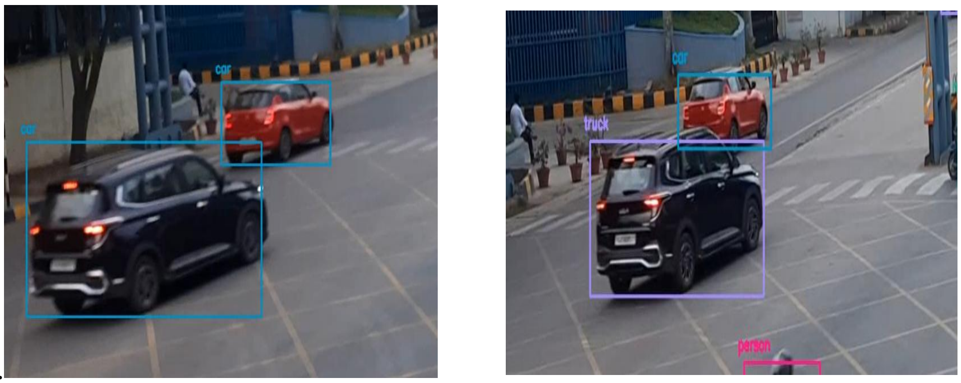
Some more observations on YOLO V3 & SORT

YOLOv3 on Traffic video:



Observations:

We observe that YOLO is fast as compared to other algorithms but the results suggest that it's accuracy hampers in the case of occlusions and similar feature of 2 different objects. When there are 2 or more objects very nearby, we see that the object labels jump from one bounding box to other as it fails to distinguish between the 2 in 2 different frames. Below 2 images show how the label of truck and car changes from frame to frame for the same object



.

Fig: Same object is classified as car and truck in 2 different frames

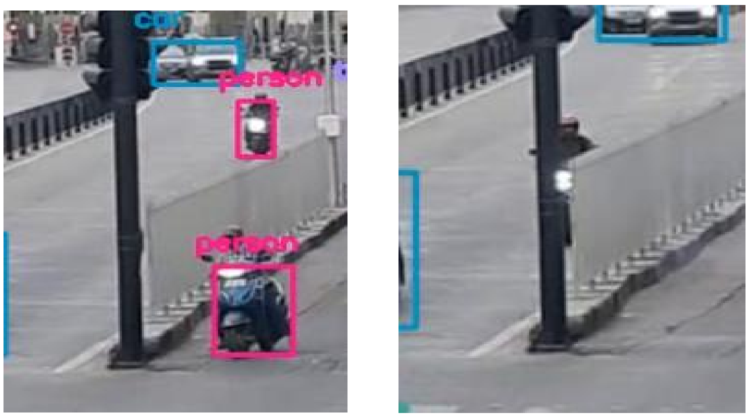


Fig: Person detected in the first scene is not detected due to occlusion in the second scene.

This flaw is more visible when we apply SORT on the detection of YOLO. We can see how the class labels jumps from one box to the other due to vicinity of similar feature object nearby.

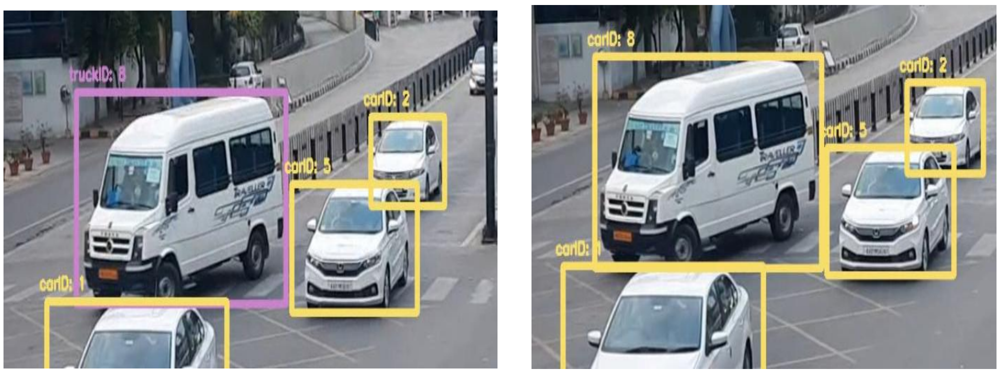
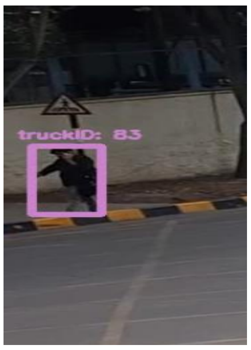


Fig: truck in first frame classified as car in the second.

We also observe that misclassification is a big issue with SORT. Though a ingenious algorithm, it lacks in accuracy due to various reasons. It uses Kalman filter which assumes the movement of the objects to be linear. The SORT implementation we used(**abewley/sort** from github) didn't give similar output length as the input of the detected bounding boxes but the number of classes remain the same which makes is prone to misclassification. In the below image, a person is misclassified as a truck.


truckID: 83