**OBJECT DETECTION**

**Road Segmentation:**

In this part of the model, the computer has to output a segmentation mask of the road leaving aside all the pedestrians and surrounding media. To do so, we use a technique of deep learning called as Fully Convolutional Network (FCN). FCN introduced the method of segmentation end to end i.e. any input image or video will directly provide the segmented output it is trained for, no pre-processing or post-processing is required.

Upsampling is increasing the size of an given input. E.g., Upsampling a 2x2 matrix with 2x increases the height and width two times the original, thereby forming a 4x4 matrix.

***Architecture of FCN (VGG NET):***

The VGG-based Fully Convolutional Network (VFF Net) architecture typically refers to an FCN variant that uses a VGG network (VGG-16 or VGG-19) as its backbone for feature extraction and then applies upsampling layers to generate dense pixel-wise predictions.

The VFF Net architecture combines the strong feature extraction capabilities of VGG with the spatially dense predictions of FCNs, making it effective for tasks like road segmentation where accurate pixel-level localization is crucial.

**YOLO: (For 2D Object Detection)**

YOLO is a real time object detection system, best known for its speed and accuracy, making it more popular in Computer Vision applications. It has the following features:

* Single-step detector (Entire image is processed only once through the neural network)
* Real-time processing
* Gives High accuracy
* Can detect Multiple objects at the same frame simultaneously.

***Working:***

* YOLO divides the input image into a grid (e.g., 4x4, 6x6, or larger).
* For each grid, YOLO predicts a bounding box and their confidence scores.
* It then predicts the probability of object classes for each bounding box.
* A non-maximum suppression is applied to refine the detections.

The code uses YOLOv8 (most latest version of YOLO system) for object detection and OpenCV for image processing. It uses Ultralytics implementation of YOLOv8 after which we preload a YOLOv8 model. Few parameters have also been set up for distance estimations, speed calculations, confidence thresholds calculations for detection as well as non-maximum suppression threshold.

The code is designed to be flexible, allowing for different input types and provides visual feedback both in real-time display and saved output files.

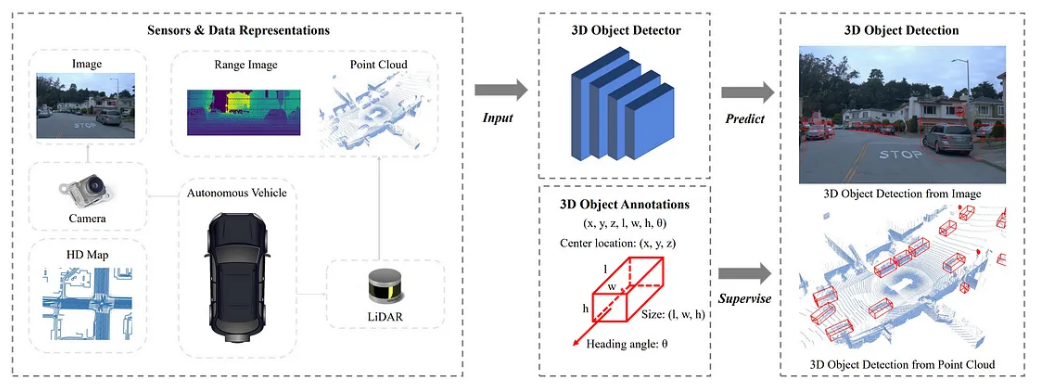
***Anchor Boxes:***

Anchor boxes play a vital role in detecting multiple objects within a single cell, providing a structured approach to identifying different shapes and sizes of objects in an image, contributing to the algorithm’s versatility and adaptability in various scenarios. Its primary purpose is to allow detection of multiple objects within a single grid cell. Without anchor boxes, each grid cell would only be responsible for detecting one object, which can be limiting.

By using multiple anchor boxes with different aspect ratios, the model becomes more adept at accurately identifying and localizing objects of various shapes. This is particularly useful for objects that don't fit neatly into square bounding boxes, like elongated objects or objects at odd angles.

**3D Object Detection**

Based on sensor types, 3D object detection can be divided accordingly: ****LIDAR(Light Detection And Ranging)-based****, ****Camera-based**** or ****Multi-modal 3D Object Detection****.



3D Object Detection for autonomous vehicles

**LIDARs** are the most important piece of hardware in any self-driving car (with the exception of Tesla) and is very necessary for understanding the environment in 3D.

They work very similar to a RADAR or a SONAR, except for the fact that they emit light instead of radio or sound waves. The advantage of a LIDAR over any other sensor is :

1) It has a 360 degree field of view.

2) The output of a LIDAR is a set of points in space, called a point cloud.

A typical point cloud consists of four different values (x, y, z and r) corresponding to the x, y and z co-ordinates of the point of which the light ray was reflected off and r being the reflectance.

Now, with 3D positions of each point in the environment, obtained from the LIDAR, the self-driving car can perceive the environment so much better.

This enables the car to not just see where the object is, but also how far it is from the car and in what direction it is oriented in.

KITTI is one of the best datasets that are available for benchmarking algorithms for 3D object detection. Using the dataset is a bit tricky as the labels in the dataset are not what you think they are.

The dataset needs to be understood really well before converting it into a useful format to use for 3d object detection. The dataset consists mainly of four different kinds of files, namely:

1) camera\_2 image (.png)

2) velodyne point cloud (.bin)

3) camera\_2 label (.txt)

4) calibration (.txt)

The camera\_2 image folder contains the RGB images taken by Cam 2 .

Each time the x-axis of the LIDAR and the y-axis of Cam 0 line up, the camera takes a picture.

The velodyne folder consists of lidar data in the form of bin files.

Each bin file consists of data corresponding to 360 degree rotation of the lidar.

Each point is stored with its (x, y, z) coordinate and an additional reflectance value (r).

Each bin file consists of approximately 120,000 such points (varies file to file).

***SFA3D:***

SFA3D stands for Super Fast and Accurate 3D Object Detection which is based on 3D LIDAR sensor.

It is particularly relevant for autonomous driving applications where real-time processing is crucial.

* It is optimized for real-time processing.
* It maintains a high detection accuracy.
* Converts a Bird’s Eye View (3D model) into 2D Model representation for efficient processing.
* It balances speed and accuracy, thereby making it suitable for embedded systems in vehicles.