**Detecting Vehicles Using Only Computer Vision**

Currently there are various state-of-the-art algorithms available for object detection and tracking, but jumping straight into this is naïve at the early stage of learning process.

So, I have tried using traditional computer vision technique to achieve the task of detecting vehicles.

The Dataset I used is available on the internet for public use. The Vehicle Image Database of the GTI databases consists of 3425 vehicle images and 3900 non-vehicle images.

In a general Convolutional Neural Network, we normalize the input image, convolve filters/kernels around the input image to detect edges, then Relu activation for non-linearity, followed by Maxpooling to keep only relevant pixels. This is a general convolutional block and we have multiple such convolutional blocks to detect features and generate feature maps. We even add Batch Normalization and Dropout as regularization technique. The final stage of this is to design an Artificial Neural Network and perform classification.

However, in traditional computer vision technique we need to handcraft the features to be extracted.

Following are the steps I followed to achieve the task.

1. Generating Dataset using vehicle and non-vehicle images.
2. Histogram of Oriented Gradients (HOG) is used to extract features for images in vehicle and non-vehicles images. In simple terms HOG algorithm looks for change in pixel intensity and orientation for a given cell(part) of an image. In my case I observed 4096 features extracted from an image by HOG.
3. Dataset is generated after extracting features from HOG and it is divided into training and testing dataset using train test split function of sklearn. My train-test ration is 80%-20% along with shuffling.
4. I used Support Vector Machine Classifier to classify whether the given image belongs to class 0 (non-vehicle) or class 1 (vehicle). The accuracy observed is 94.67%.
5. The sliding window section contains three main functions viz; **draw boxes:** draw boxes around detected objects, **slide window:** using this function I have defined different region of interests to detect objects of different scale, also no need to slide window on entire image as there are no cars in the sky so we can just focus on the road, **detect vehicles:** from the sliding window resize the image as per the height and width of the SVC classifier we trained in Step 4 (In my case it is 64x64). Extract features from the ROI and do the predictions using the trained SVC classifier.
6. The draw boxes function of Step 5 draws multiple overlapping boxes around an image detected by the algorithm. To suppress all the overlapping boxes and to get only one bounding box we make use of heatmap. In Deep Learning the same task is achieved using Non-max suppression.
7. I have defined a pipeline which follows the above-mentioned steps but also keep track of 25 frames in order to average them while drawing boxes. This allows the boxes to be smooth in the final output.

Conclusion:

Using Keras, TensorFlow, Pytorch it is simple to build architecture but at the same time it is also important to learn how is feature extraction is done (I have played with different features of HOG before finalizing the parameters), slightly hypertuned the SVC classifier. I can produce better results if I get more data. There are certain False Positive but initially there were a lot, I fixed it using more number of frames for averaging and also changed the threshold for heatmap generation.