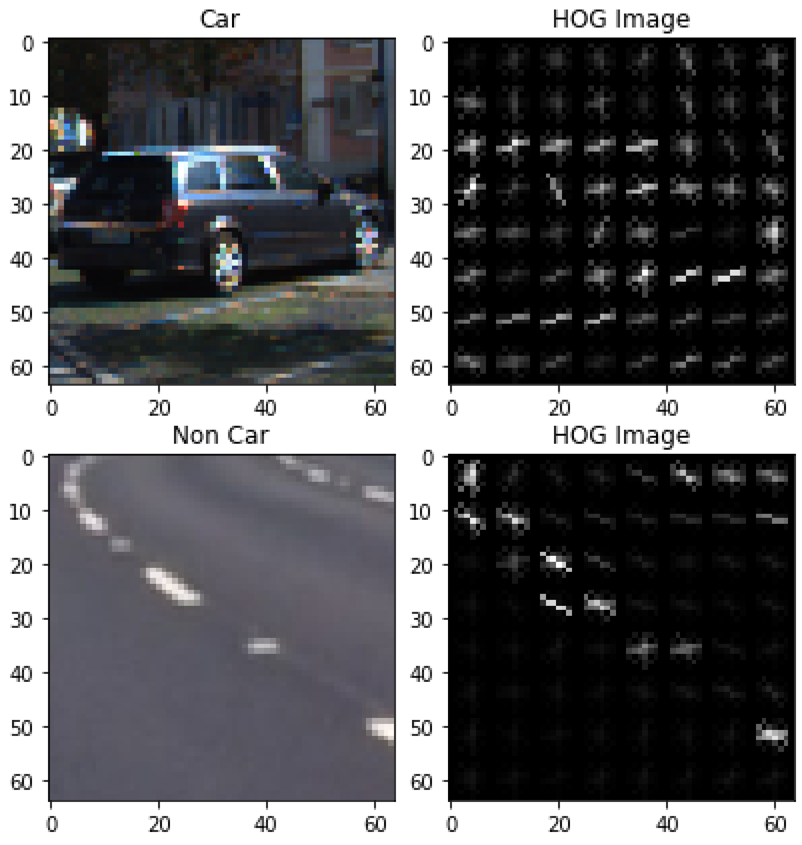
Project-5
Vehicle Detection and Tracking

Histogram Oriented Gradients(HOG)

1. The functions which i used to extract features like hog features, spatial binning and color histogram are as follows:

- get_hog_features()
- bin_spatial()
- color_hist()

The Hog features are defined in Line[6] of the notebook. Spatial binning is defined in In[7] and the color histogram in the In[8] of the notebook.
Here we should convert the car images and non car images into hog features. In order to convert the image into hog feature we should **import hog from skimage.feature**. Then we should pass certain parameters along with the images to hog(). The parameters include images, orientation, pixels_per_cell, cells_per_block etc. These are the import parameters.

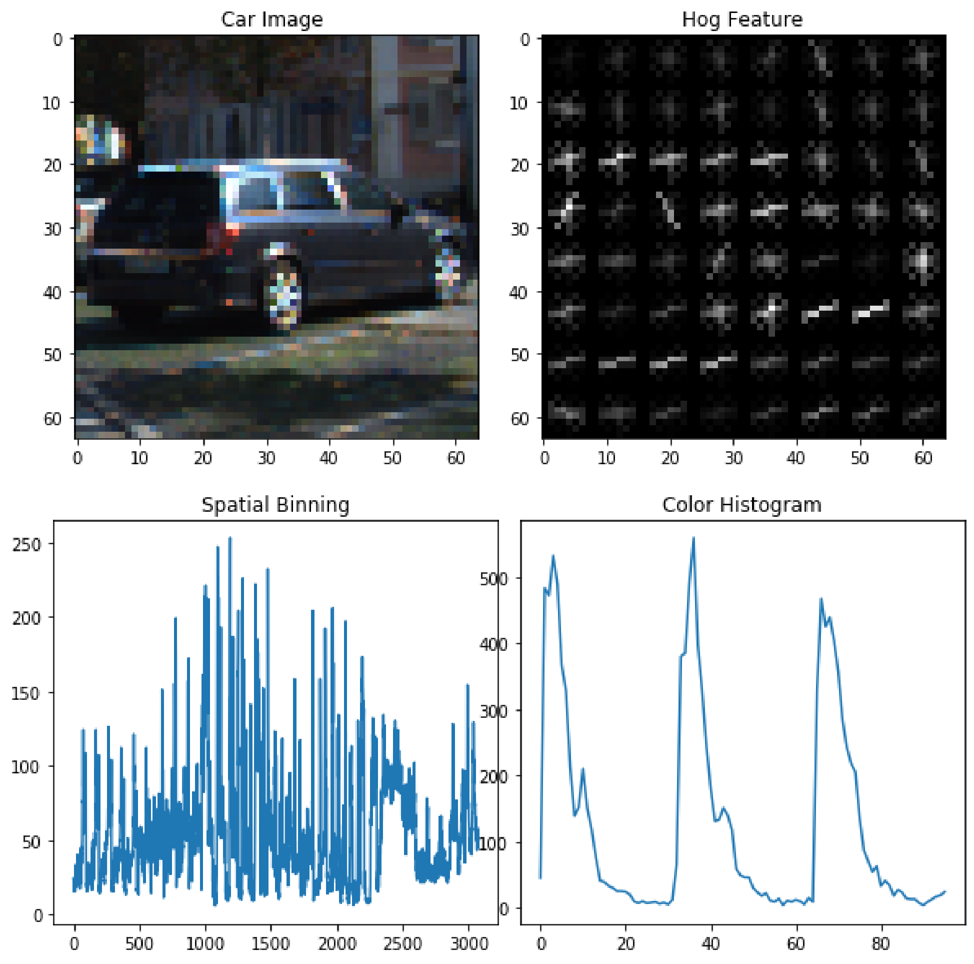


2. In order to train the classifier we should collect all the features(includes hog feature, spatial binning and color histogram) of the image and append it to an array. Here we first collected the

hog feature, secondly the spatial binning and finally the color histogram of the images. After collecting the feature, the collected array should be scaled for splitting and training.
X_scaler = StandardScaler().fit(X)
scaled_X = X_scaler.transform(X)
The code above will scale the images. For splitting the data for testing we use:
X_train, X_test, y_train, y_test = train_test_split(scaled_X, y, test_size=0.2, random_state=22)
This will split the data for training and testing. **X_train, X_test** contains the images and **y_train** and **y_test** will contain the labels of the particular image.
The dataset along with the parameter gave a test accuracy of 0.9899 which means 98.9%. The parameter with values are:

- Orientations: 8
- Pixels per cell 8
- Cells per block 2

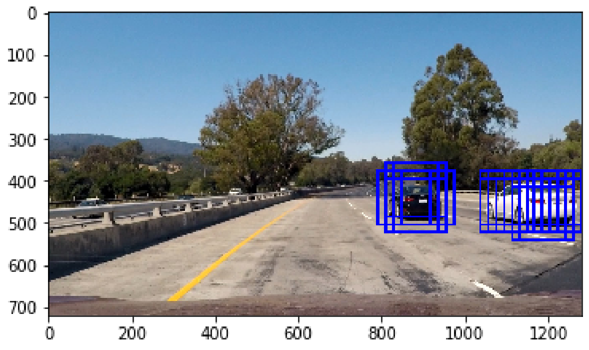
The feature vector which is the trained classifier have samples of 2432. Code lies in In[12] of the notebook file.

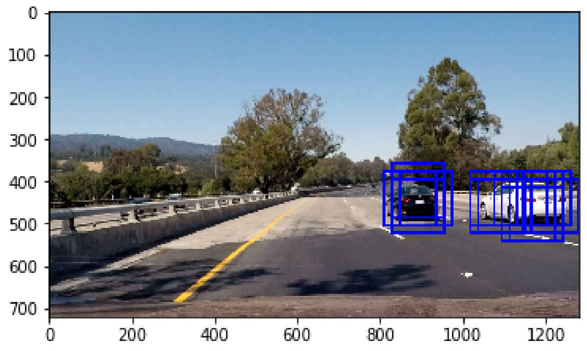


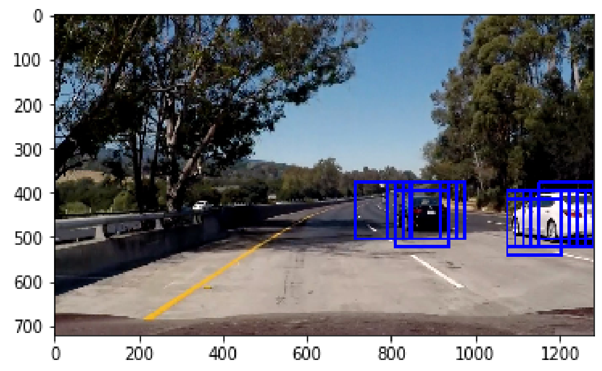
Note: In Color Histogram there are three peaks which represent the R,G,B value of the image respectively.

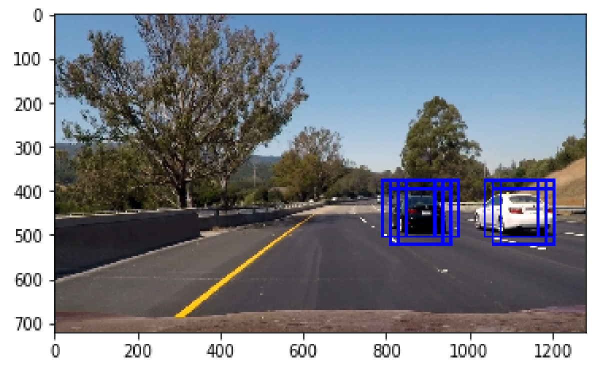
<u>Sliding Window Search</u>

1. Sliding window is used to slide across each blocks in the image and return the list of windows. The search window accepts the window list along with other parameter. The search window also collect the features of the image like Hog, Spatial and Histogram features and append to a list. Then this list will be checked with feature vector(classifier) to identify whether a car is present in the image or not. If there is a car present in the image, the coordinates of the window list will be passed to other functions to draw the particular coordinates. For instances:







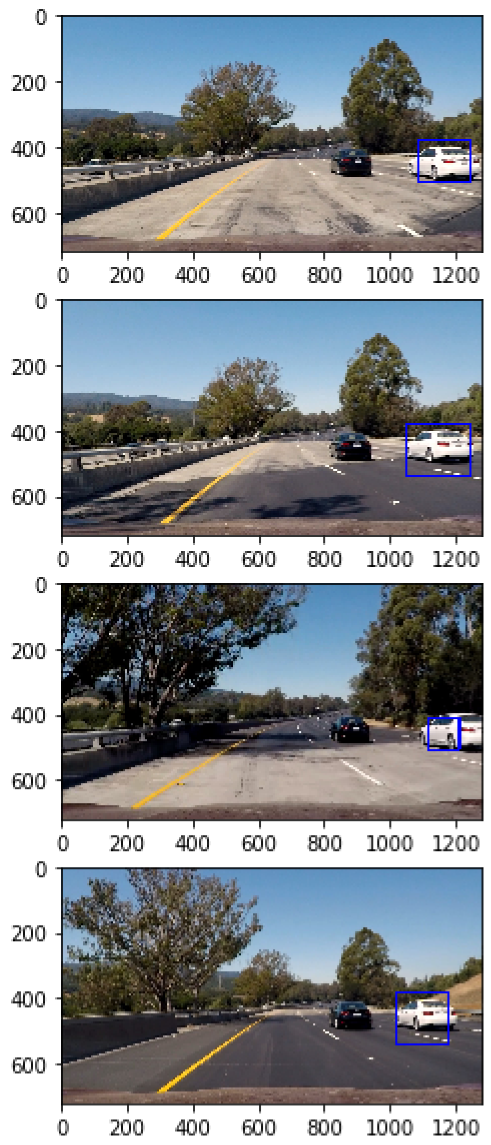


The function used for slide window is slide_window() and search window is search_window(). Slide window in this project overlaps about 50% and the window size is 64*64. Code for Slide window lies in In[14] and search window lies in In[17] of jupyter notebook.

Discussion

It is very hard to maintain the size of box drawn around a car. The pipeline is able to detect the cars but the size of box sometimes be smaller than the car. The pipeline will fail when a car appears on the top half of the image. Because to avoid the algorithm to get overpopulated we only choose the bottom half for searching.

2.



In the four examples car inside 400 to 600 along the y-axis and 600 to 1280 of x-axis are detected are marked inside the box. But the black car in four examples are not detected because this car is not inside the mentioned coordinates.
The performance of the classifier can be optimized by using the following parameter:

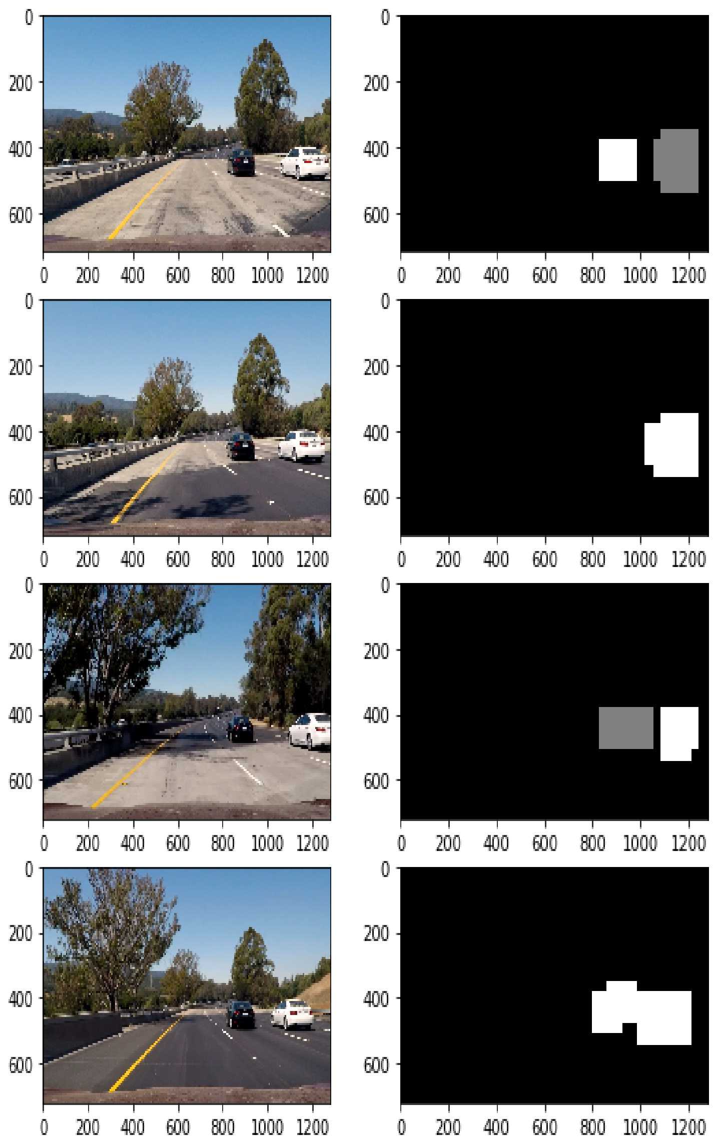
- Orientations
- Pixels per cell
- Cells per block

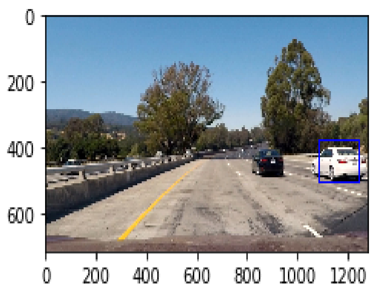
For optimizing the project i have chosen an orientation of 8, pixels_per_cell of 8 and cells per block of 2. By using this parameter, i got an accuracy of 98% which is very good to test and predict items.

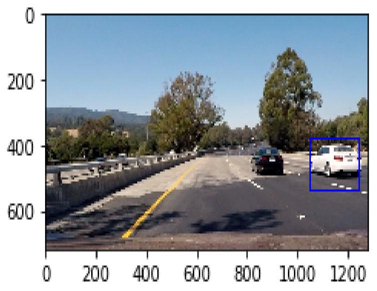
Video Implementation

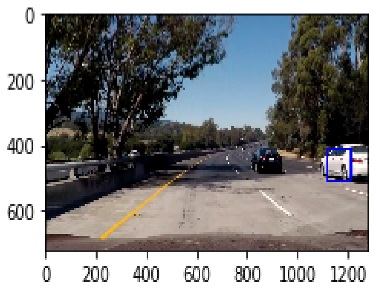
1. The project output video is attached with the submission file(zip file).

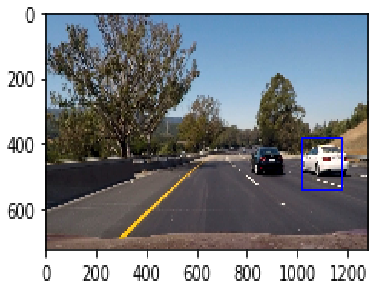
2. When there are several subsequent frames on an image there will be lot of boxes around a car. In order to avoid this we take the heat-map of that particular area and draw a box around the heat-map area. For example:


0
200
400
600
0 200 400 600 800 1000 1200


0
200
400
600
0
200
400
600
800
1000
1200


0
200
400
600
0
200
400
600
800
1000
1200


0
200
400
600
0
200
400
600
800
1000
1200


0
200
400
600
0
200
400
600
800
1000
1200