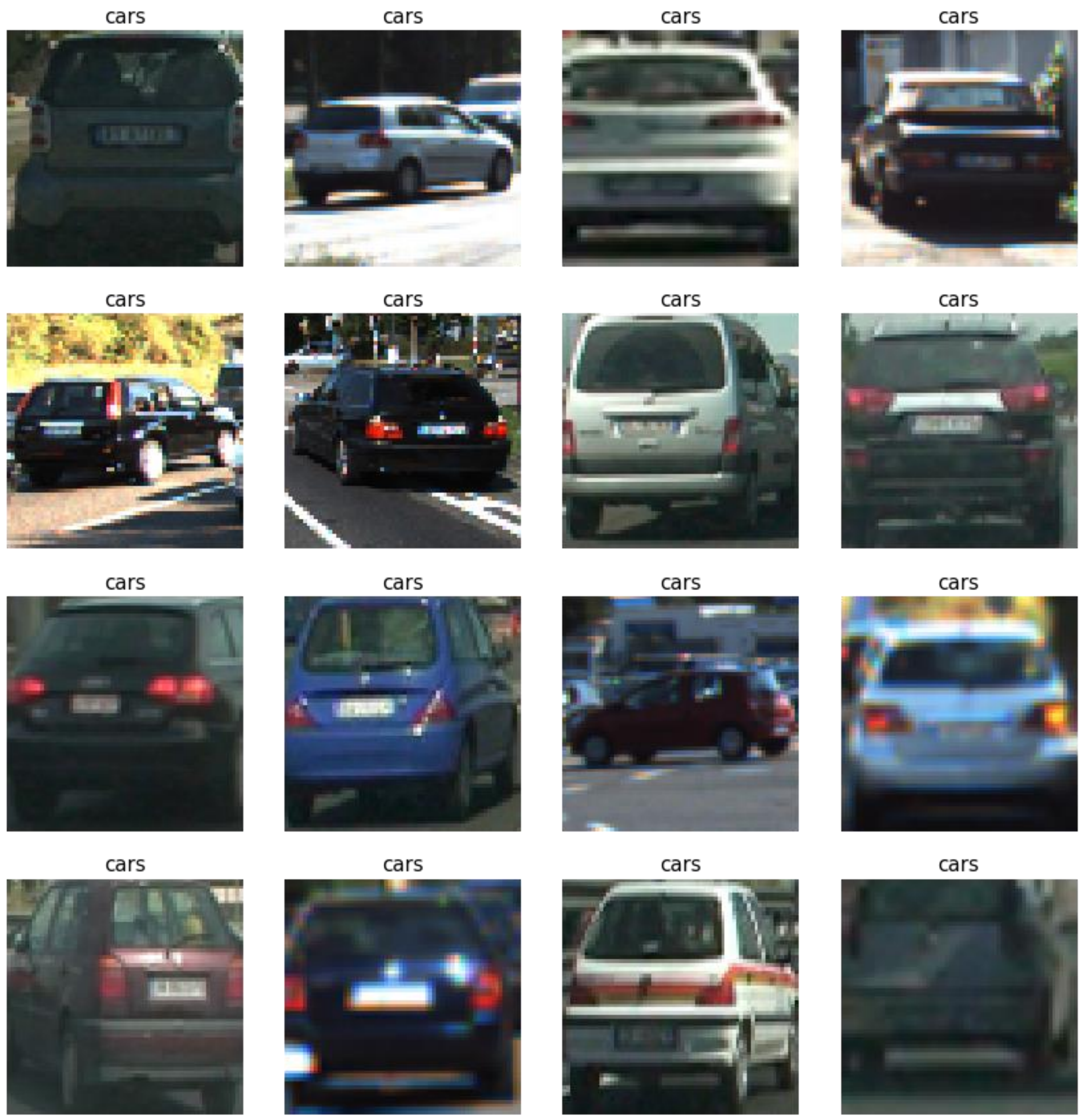
Write Up

In this project I will explain how I tackled with the problem of detecting vehicles in the video.

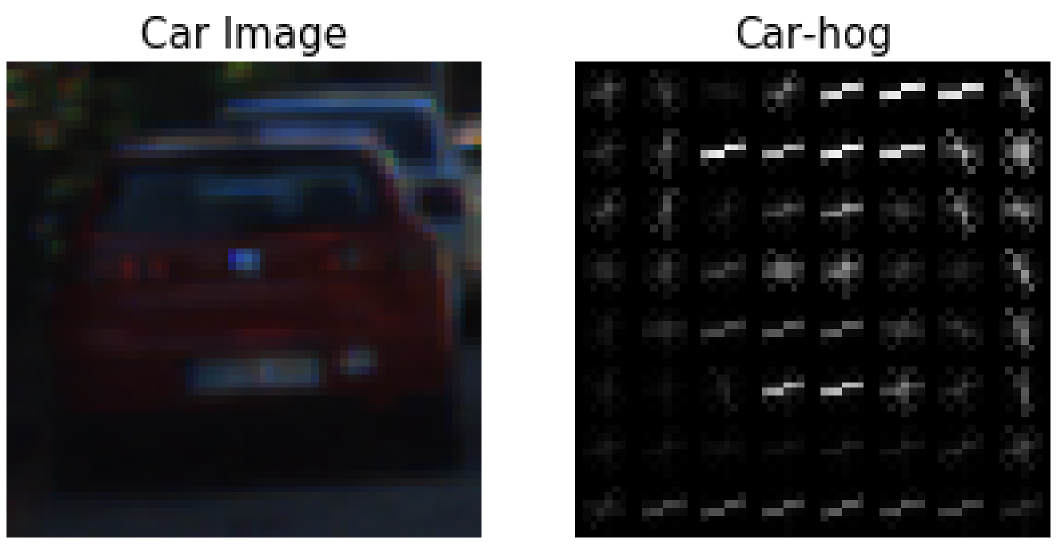
Data

Firstly, I included all of the data that I had in my hand. As it can be seen I had 12868 not car pictures and 12217 car picture. But I chose only images with GTI and KITTI dataset. That contain 8968 not car images and 8792 car images. Those below are some pictures of my car data.



And below you see visualization of not car data

After that I define all of the functions provided by Udacity. Including, bin_spatial() to get 1D flattened array. I also use color_hist() to get histogram of different color masks. I also use Histogram of oriented gradients (HOG) in the function called get_hog_features() to get gradient representation of the image.



Here is the visualization of Hog transform in car photos.

Next I have function extract_feauters(), which combines all of the above mentioned steps. It takes in path of images and then returns flattened combined array of hog, histogram features.

Parameters

In the next section I define parameters for my feature extraction. I have tried many different parameters but comparing by training accuracy of my images and extraction time (YUV seemed better in training accuracy, but it was taking too much time). Finally, I decided to stick on those.

color_space = HLS

orient = 16

pix_per_cell = 16

cell_per_block = 2

hog_channel = 'ALL'

spatial_size = (32, 32)

hist_bins = 32

We are extracting all channels from hog which is more time consuming, but it gave me better performance in training accuracy which we will see in next section.

Training

Here I will explain how I trained my model.

I used normal linear SVM classifier to train my model on differentiating between not car and car images. Here the accuracy score was the most important for my choices.

As you see in the first part I defined

sample_size = len(cars)

cars = cars[:sample_size]

notcars = notcars[:sample_size]

I used this part of code to test my accuracy percentage. I normally made sample_size 1500 and trained my model and feature extraction. My accuracy for 1500 images was less than on 8792 but for parameter tuning it was less time consuming.

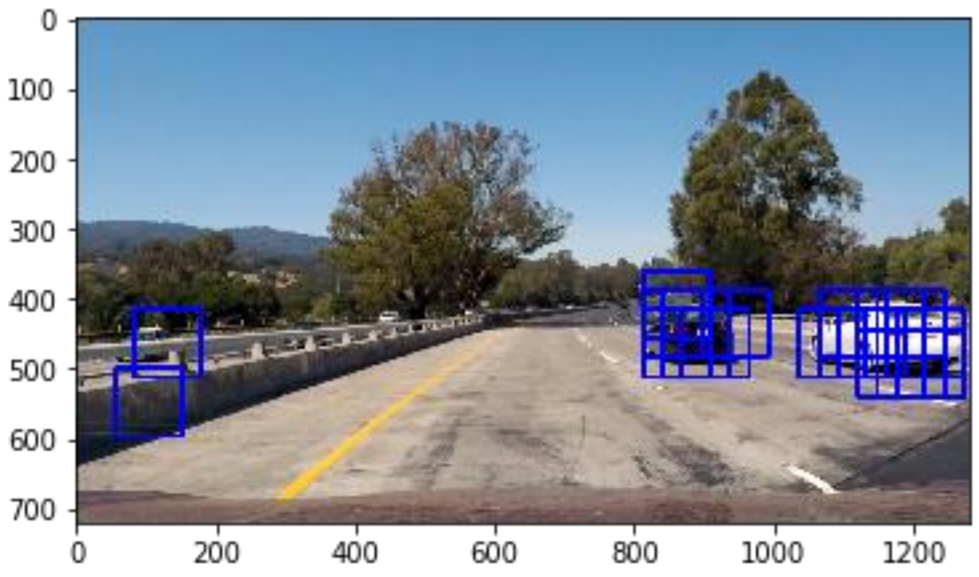
Note*: color space YCrCb with 12 orient gave me better prediction accuracy on the test set, but actually failed too much on project video, so I decided to change it.

As you see, the parameters provided above with 8792 images of not car and car gave me 98.04% accuracy which I think is pretty good. The SVM took 21.41 seconds to train.

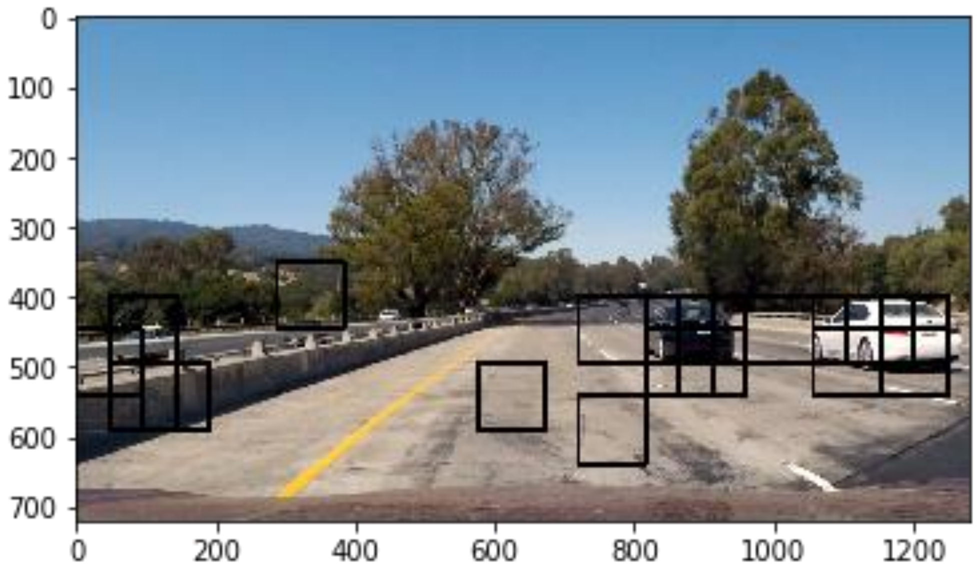
Sliding Windows

In this section I define 2 functions slide_window() and draw_boxes(). They are taken from Udacity quizzes. They help me to get small windows in every images which help me to predict if the scene in this window is car or not. While taking this small windows I use search_windows() to see if this small part of image is predicted car or not and if it is I draw rectangle around it.

But this is too basic, to improve our mode prediction on image I use hog_subsampling provided by Udacity, which is defined in find_cars() function. Which looks pretty successful on finding the cars.



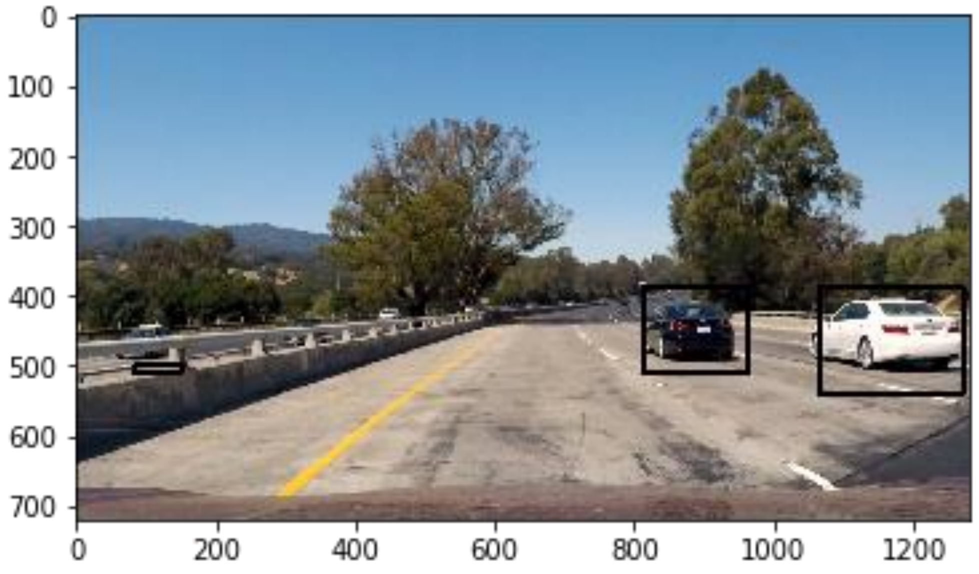
With subsampling



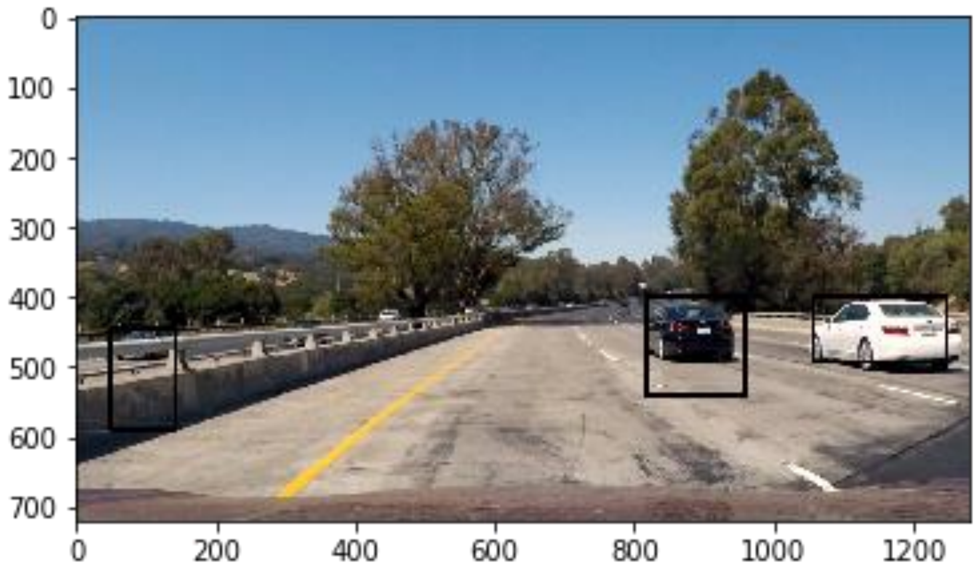
False Positives

False positives can be huge problem. To get rid of them I used heat maps. Which looks at rectangles and if too many are on concentrated on some scene it marks the area as object and draws rectangle around it. I define function applyLabel() where I use heat map and then draw rectangles around my objects.

Here are the results without subsampling



And with subsampling



Final part

I tested my model on different part of images with different hog subsampling scales. function process() is clear example of that. I use different scale for windows on different areas and then apply my find_car function. This helps me to find different size of objects in certain area and to be more accurate.

I test this function on my testing video. And it does not look bad. But improvements can be made. So, I decided to create object which will held previously detected objects like we did in advanced lane finding project. I create class with one function to add to objects only parameter drawn rectangles. Then I use those rectangles to calculate my next values. This can be very helpful at locking target and preventing false positives.

Result

In the end the result is good, but it has several false positives in video. This could be explained because of gradient that change between shadow on traffic.

Further

This was great project where I learned much about hog feature, subsampling, great idea of how to remove false positives and many others. But I don't think SVM is good for detecting objects. Firstly, it can be too slow in real time and secondly it does not have that much of an accuracy. Looking at the deep learning models, YOLO and MASK RCNN seem much more fast and accurate. So, after that I am planning to implement MASK RCNN, which hopefully will be interesting.