Project #5: Vehicle Detection

Self-Driving Car Nano Degree | Vikas Sharma

The goals / steps of this project are the following:

- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier
- Optionally, you can also apply a color transform and append binned color features, as well as histograms of color, to your HOG feature vector.
- Note: for those first two steps don't forget to normalize your features and randomize a selection for training and testing.
- Implement a sliding-window technique and use your trained classifier to search for vehicles in images.
- Run your pipeline on a video stream (start with the test_video.mp4 and later implement on full project_video.mp4) and create a heat map of recurring detections frame by frame to reject outliers and follow detected vehicles.
- Estimate a bounding box for vehicles detected.

Data Processing

Firstly data for vehicle and non vehicle images was stored in separate folders,

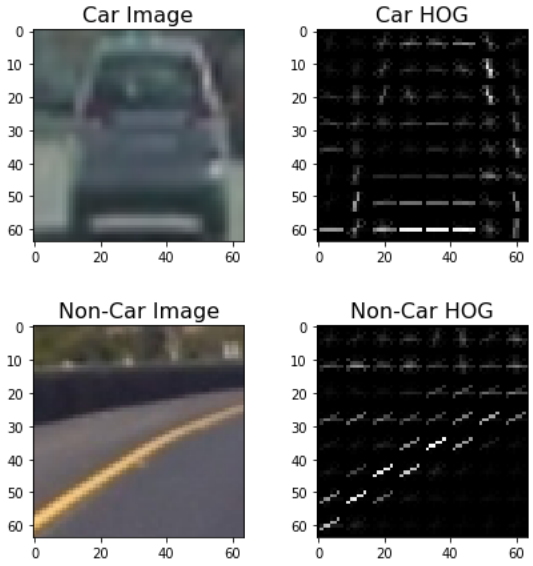
```
vehicle images :  8792
non-vehicle images :  8968
```

e.g.

Histogram of Oriented Gradients (HOG) feature extraction

Used get_hog_features() function from the class to get HOG.

Results:



HOG parameter exploration

I tried following parameter combinations

#Orientations = 8, Pixel per cell = (8,8), cells per block = 2

#Orientations = 9, Pixel per cell = (8,8), cells per block = 2

#Orientations = 8, Pixel per cell = (16,16), cells per block = 2

#Orientations = 9, Pixel per cell = (16,16), cells per block = 2

#Orientations = 8, Pixel per cell = (8,8), cells per block = 1

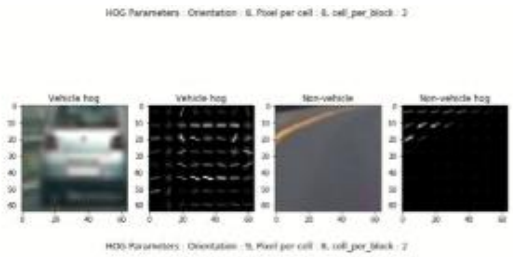
#Orientations = 9, Pixel per cell = (8,8), cells per block = 1

#Orientations = 8, Pixel per cell = (16,16), cells per block = 1

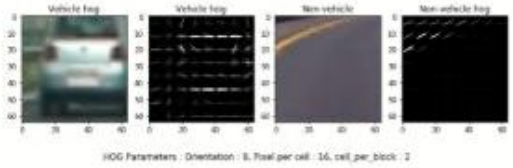
#Orientations = 9, Pixel per cell = (16,16), cells per block = 1

#Orientations = 9, Pixel per cell = (24,24), cells per block = 1

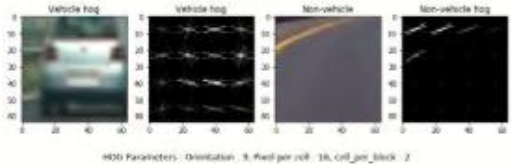
Following 3 combinations were picked for SVM time vs accuracy study:

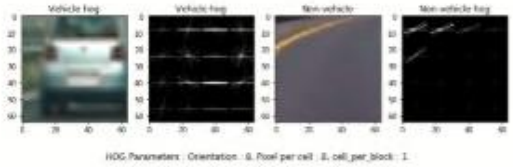


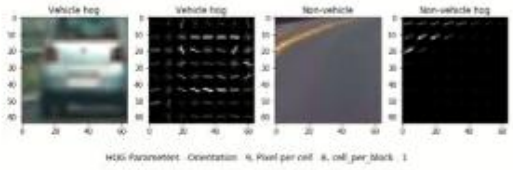
#Orientations = 8, Pixel per cell = (8,8), cells per block = 2



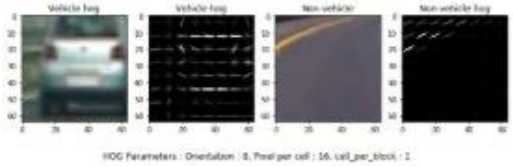
#Orientations = 9, Pixel per cell = (8,8), cells per block = 2







#Orientations = 8, Pixel per cell = (8,8), cells per block = 1



I tested them on ALL channels in HSV, RGB, YCrCb and YUV, to see SVM accuracy vs training time. I found Orientations = 8, Pixel per cell = (8,8), cells per block = 2 as the best parameters, giving an accuracy of 98.8% and training time of 4.09 sec.

Feature extraction using Color Histogram, spatial bin and HOG

I used color_hist, spatial_bin and HOG functions to extract features.

I used following parameters to extract features(ref).

colorspace = 'YCrCb' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb

orient = 8

pix_per_cell = 8

cell_per_block = 2

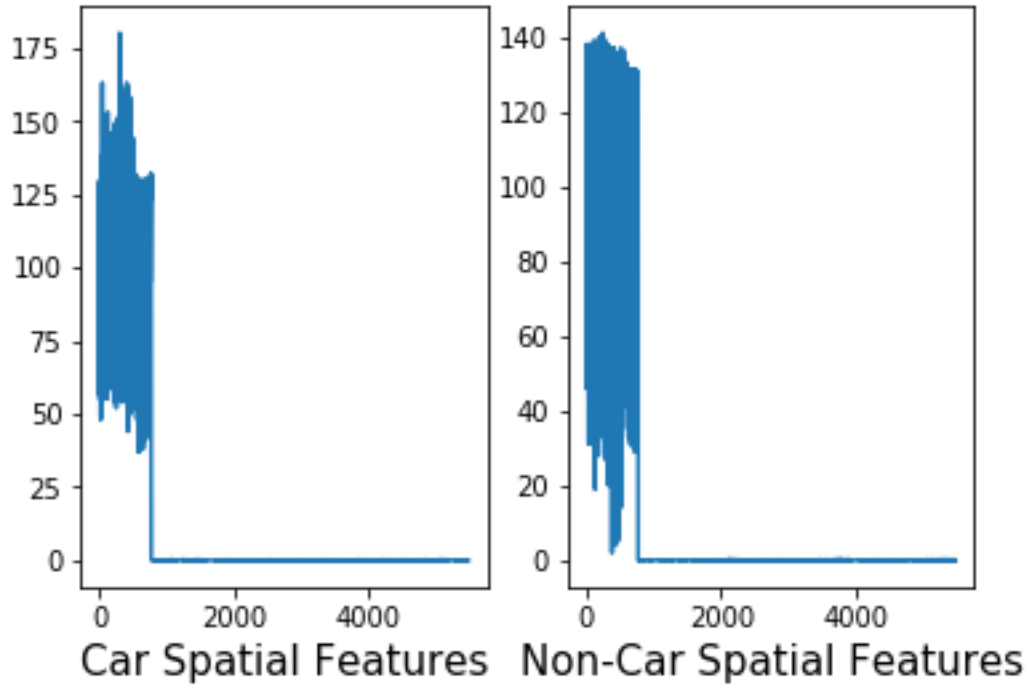
hog_channel = 'ALL' # Can be 0, 1, 2, or "ALL"

spatial_size = (16, 16)

hist_bins = 32,

scale = 1.5

hist_range = (0, 256)

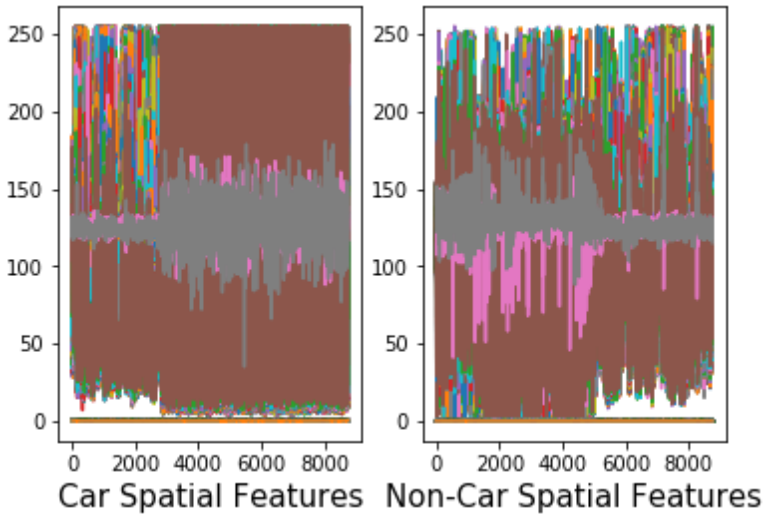


Data Scaling using StandardScaler

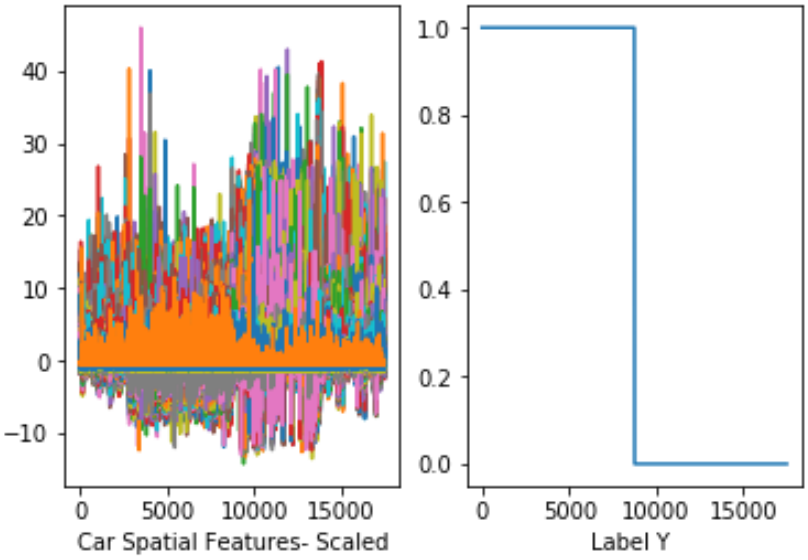
I combined car and non car features in to an array.

I used StandardScaler to center the data around 0. The StandardScaler assumes data is normally distributed . The I generated a label array assuming output of car features to be one and non-car features to be zero.

Non-scaled features



Scaled combined car and not car features and label Y



Train a classifier

I planned to use Linear Support Vector Machine(SVM) for classification.

I split the data and train the classifier. I got an accuracy of 99.26%.

```
Using: 8 orientations 8 pixels per cell and 2 cells per block
Feature vector length: 5472
4.28 Seconds to train SVC...
Test Accuracy of SVC =  0.9926
```

Sliding Window Search

A sliding window is rectangular region of fixed width and height that “slides” across an image.

For each of these windows, we analyse the window region and apply an image cla

ssifier to determine if the window has an object that interests us. I implemented sliding search

using findcars(function) and drew bounding boxes around positive detections/

False detection

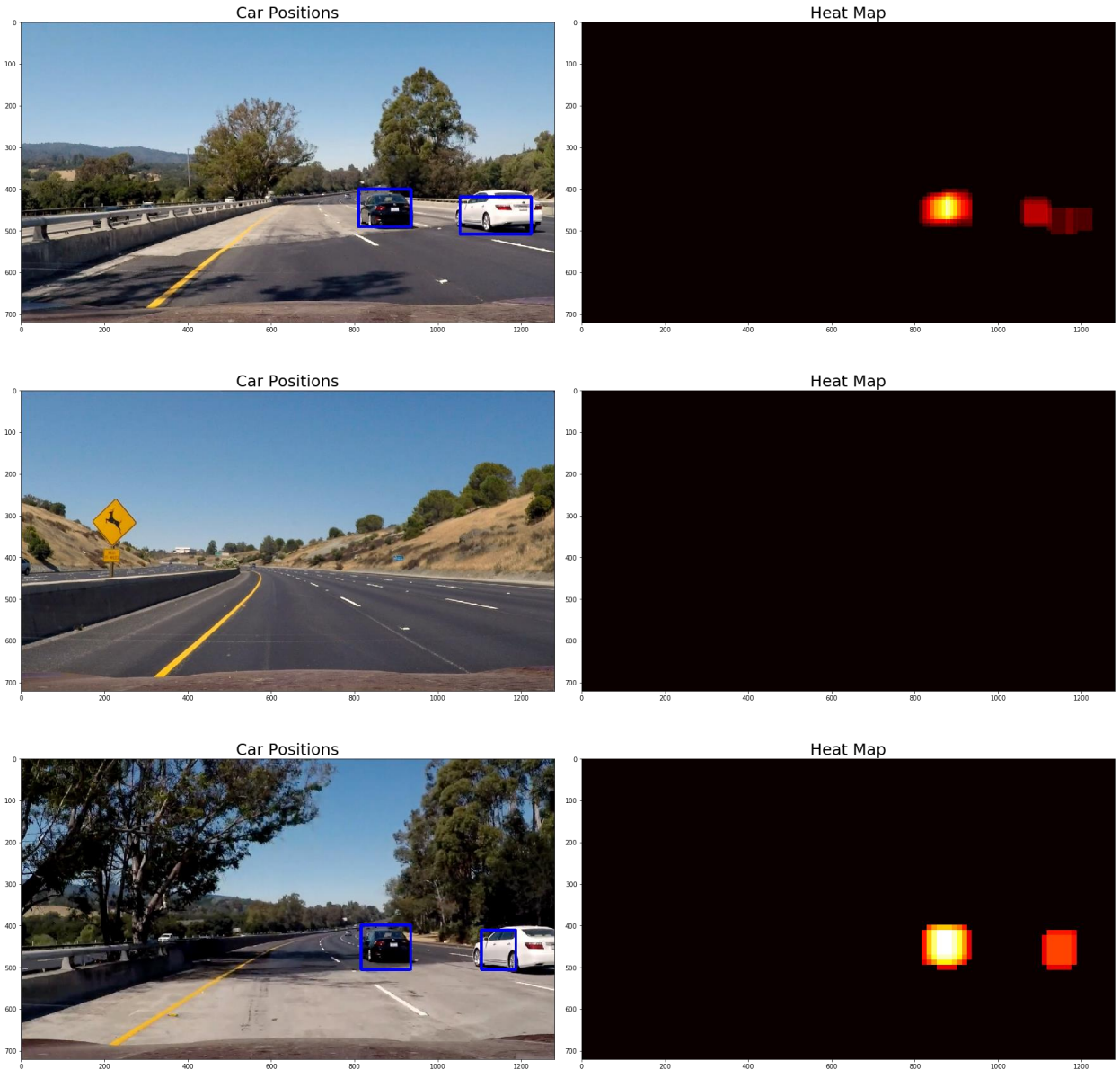
I used heatmaps to combine the bounding boxes and threshold to remove false positives.

I also stored the heatmaps of the N most recent frames in a deque which automatically

deletes the oldest entry when the list exceeds a maximum length. In every frame I took the sum of

all of those heat maps and threshold the combined heat map with 10 frames. This not only

helped filter out false positives, but it also makes the boxes to appear much smoother across frames.



More efficient HOG sub sampling Sliding window search and create pipeline to process video

The hog sub-sampling is more efficient method for doing the sliding window approach. It only extracts hog features once and then can be sub-sampled to get all of its overlaying windows. Each window is defined by a scaling factor where a scale of 1 would result in a window that's 8 x 8 cells then the overlap of each window is in terms of the cell distance. This means that a cells_per_step = 2 would result in a search window overlap of 75%. Its possible to run this same function multiple times for different scale values to generate multiple-scaled search windows. The hog sub-sampling helps to reduce calculation time for finding HOG features.

I used svc.decision_function(X) to pick higher confidence prediction.

Please see repository for processed video results. It does reasonably well.

Discussion

Major challenge I faced was detection accuracy. Even after using heat map and setting threshold, I was unable to get rid of all the false positives. I tried my pipeline on different

videos and it didn't perform nearly as well as I thought it would. Lighting condition and speed of vehicles seem to effect results.