Project# 5

Isaac Kargar

Vehicle Detection Project

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.

Rubric Points

Here I will consider the rubric points and describe how I addressed each point in my implementation. The explaination is not seperate for every rubric point.

For selecting best parameters for svm classifier I used gridsearch and the results was rbf kernel with C=10. Performance of this classifier is so good and results are as follow for color based features and HOG based features. But its speed is low and it is not so real time.

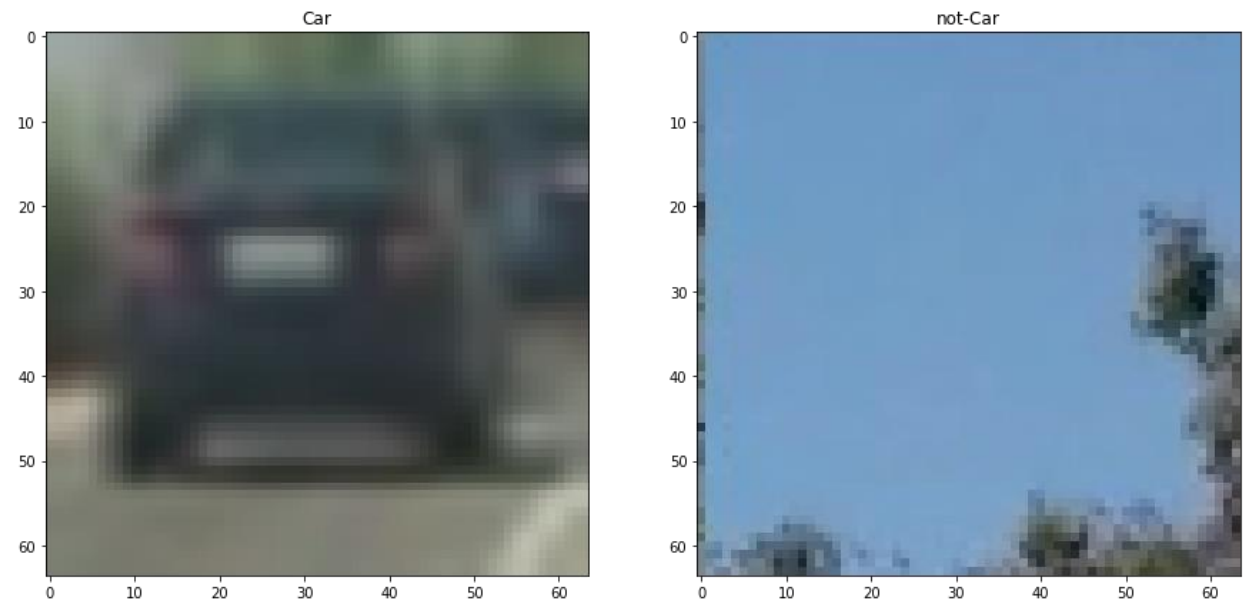
Color based	Using spatial binning of: 32 and 32 histogram bins
	Feature vector length: 3168
	1981.54 Seconds to train SVC...
	Test Accuracy of SVC = 0.9868
	Best parameters: {'C': 10, 'kernel': 'rbf'}
HOG based	82.98 Seconds to extract HOG features...
	Using: 9 orientations 8 pixels per cell and 2 cells per block
	Feature vector length: 1764
	986.4 Seconds to train SVC...
	Test Accuracy of SVC = 0.991
	Best parameters: {'C': 10, 'kernel': 'rbf'}
Combining features	Using spatial binning of: 32 and 32 histogram bins
	Feature vector length: 8460
	3575.03 Seconds to train SVC...
	Test Accuracy of SVC = 0.9972

	Best parameters: {'C': 10, 'kernel': 'rbf'}

Applying this pipeline on project_video.mp4 takes so much time: `1h 28min 25s`
The notebook is in project folder. You can check the results.

After some try and error I decided to use linear svm classifier. In following paragraphs I will explain results of that.

Firstly I loaded all images in cars and notcars list (first cell of notebook). You can see an example as below:



Then I used following pipeline:

- extracted histograms of colors and bin spatial as features and vectorized ((1,n) shape) them tu use in classification.
- Concatenate cars and notcars features in X vector to train classifier.
- Normalize X
- Split X to train and test randomely
- Define linear svm classifier
- Train the classifier
- Predict on test data set and calculate accuracy

Code of this pipeline is in cell #2.

After training classifier using these features we got following results:

```
Using spatial binning of: 32 and 32 histogram bins
Feature vector length: 3168
43.92 Seconds to train SVC...
Test Accuracy of SVC =  0.9158
```

I used svc.get_params command to get parameters of trained classifier:

```
<bound method BaseEstimator.get_params of LinearSVC(C=1.0, class_weight=No
ne, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)>
```

Color space of images is RGB. After this for combining color based and HOG based features we will test different color spaces and see their performance.

The accuracy is more than 90%. That's not bad just for color based features.

Then I used HOG features to train a classifier. The pipeline as same as above. Color space is as same as above. Other parameters are as follow:

colorspace = 'RGB'

orient = 9

pix_per_cell = 8

cell_per_block = 2

hog_channel = 0

Results of this method are as follow:

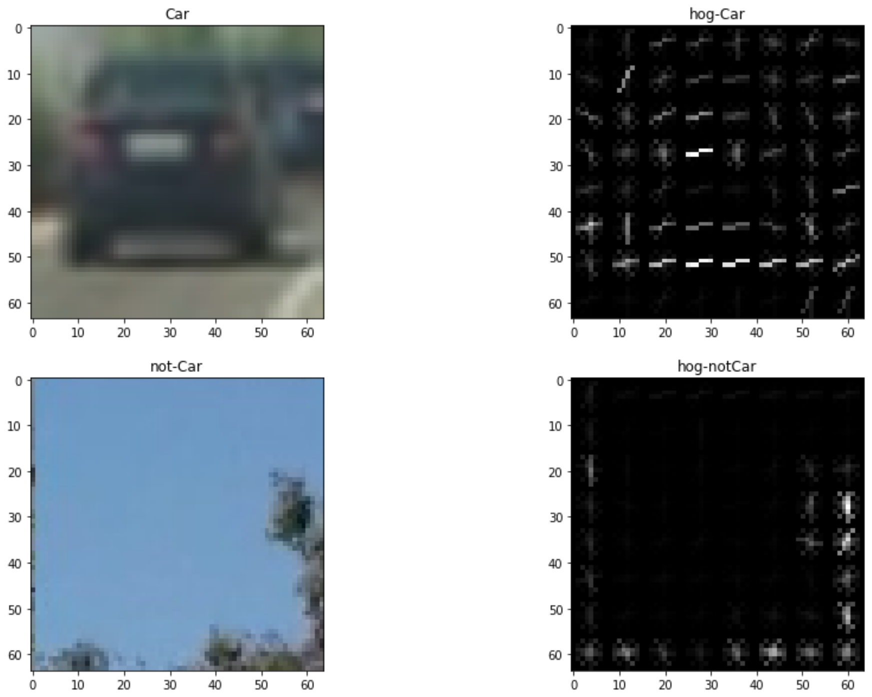
```
104.26 Seconds to extract HOG features...
Using: 9 orientations 8 pixels per cell and 2 cells per block
Feature vector length: 1764
12.06 Seconds to train SVC...
Test Accuracy of SVC =  0.9347
```

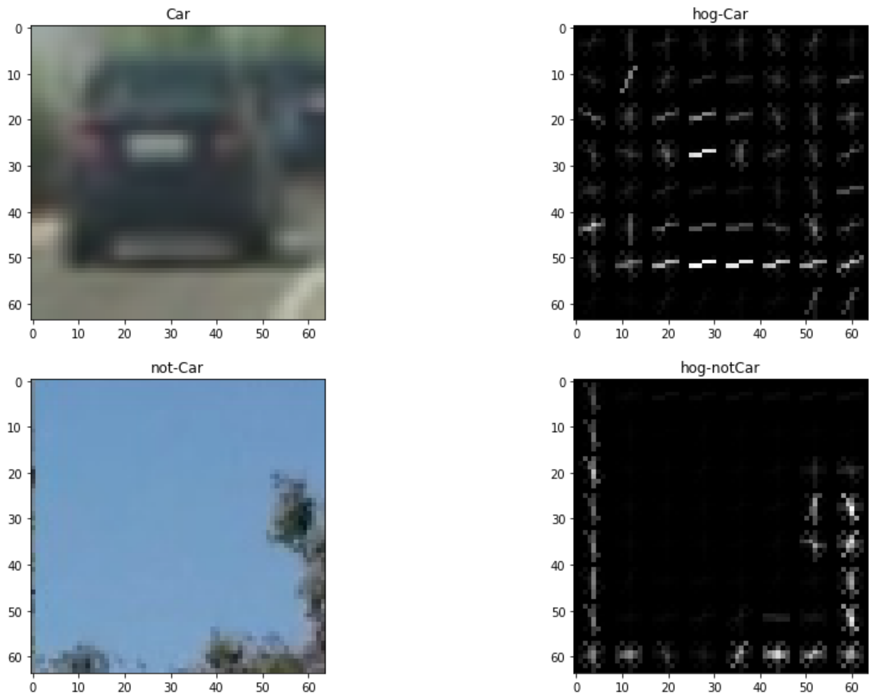
I used svc.get_params command to get parameters of trained classifier:

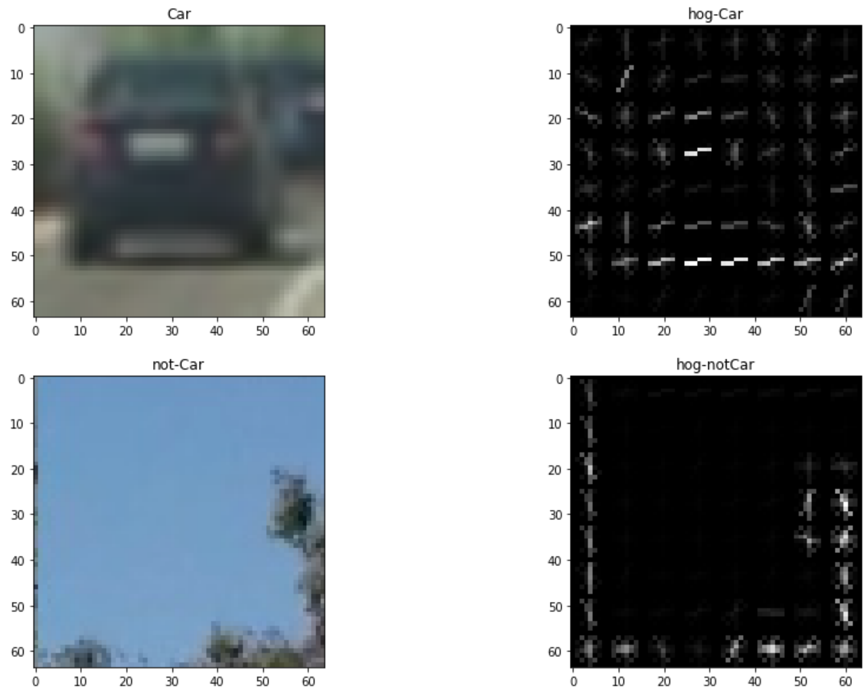
```
<bound method BaseEstimator.get_params of LinearSVC(C=1.0, class_weight=No
ne, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)>
```

The accuracy improved to 93%. Now I combine color based features and HOG based features.

For more explaination, I plot 3 different hog channel of a car and a not-car example. Color space is RGB:


Car
hog-Car
not-Car
hog-notCar


Car
hog-Car
not-Car
hog-notCar



Now I will combine color based and HOG based features as training data.

I tried different set of parameters and selected following parameters:

spatial = 32

histbin = 32

orient = 9

pix_per_cell = 8

cell_per_block = 2

I tested different color spaces and got following results:

RGB	`Using spatial binning of: 32 and 32 histogram bins`
	`Feature vector length: 8460`
	`28.08 Seconds to train SVC...`
	`Test Accuracy of SVC =  0.987`

HSV	`Using spatial binning of: 32 and 32 histogram bins`
	`Feature vector length: 8460`
	`9.18 Seconds to train SVC...`
	`Test Accuracy of SVC =  0.9904`

HLS	`Using spatial binning of: 32 and 32 histogram bins`
	`Feature vector length: 8460`
	`32.28 Seconds to train SVC...`
	`Test Accuracy of SVC =  0.9901`

In the case of RGB color space, the color space conversion would be done from RGB to YcrCb space.

Based on above results, I selected HSV color space.

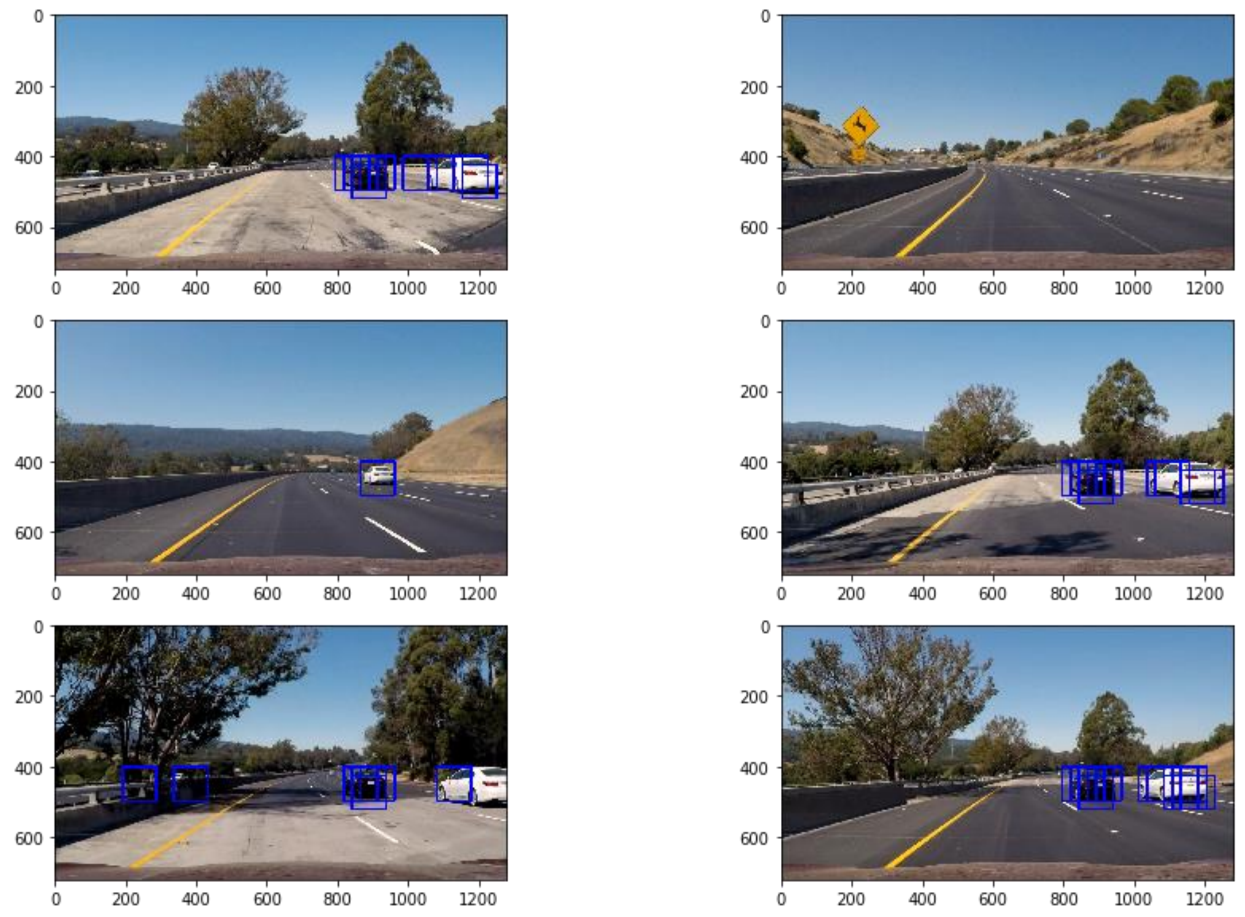
For implementing sliding window I selected region of interest. This is a region that cars would probably be there. Cars would not be in some spaces such as sky. For each video frame I got HOG features of whole frame from 3 channels and then slide a window from a specific y_start to y_stop along all x pixels with step of 2 cell (75% overlap) on region of interest to detect cars. Each cell is 8 pixel. I also implemented multi scale window to search faster. I used 2 scale as below:

Y_start = 400, y_stop = 550, scale = 1.5

Y_start = 500, y_stop = 700, scale = 2.5

I tested HSV color space on test images and results were not good.

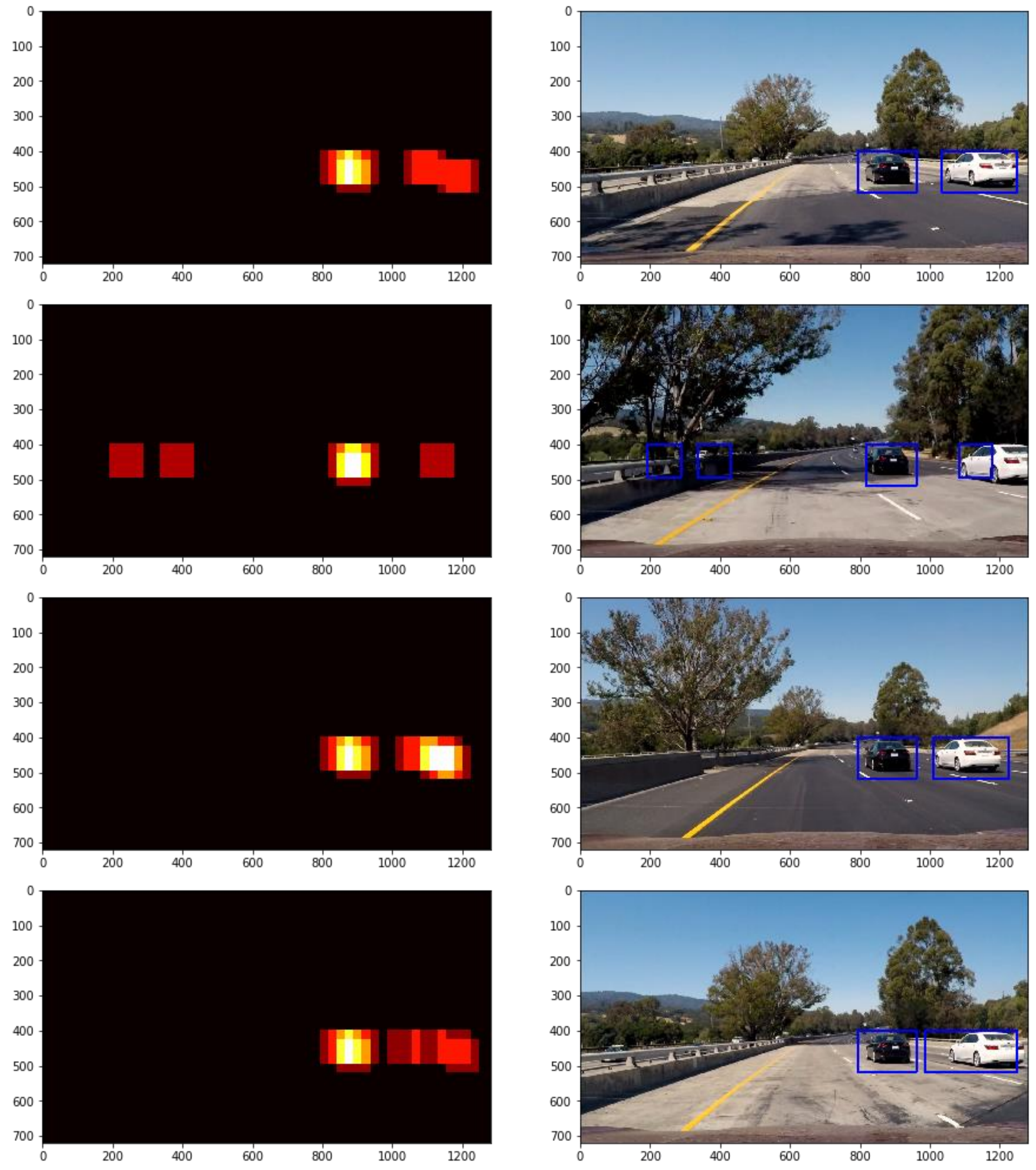
I tested YcrCb color space again:

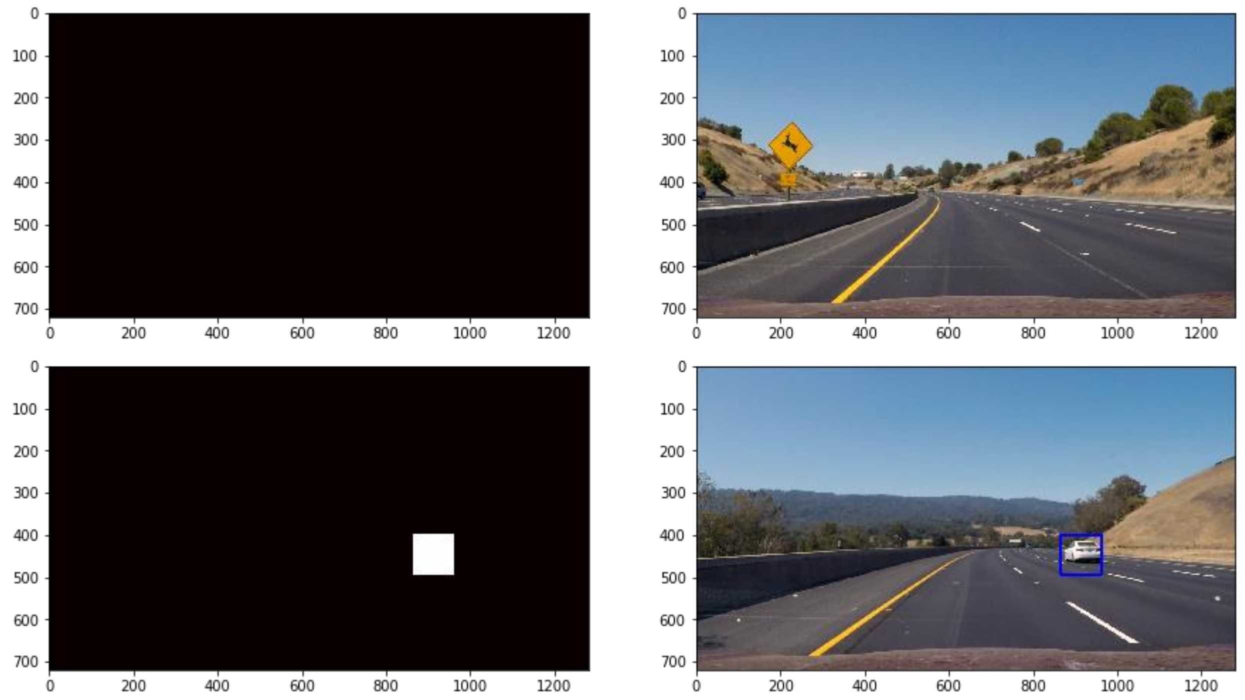


These look better. Hence, I continue with YcrCb color space.

For removing false detections I used heat map technique. I recorded the positions of positive detections in each frame of the video. From the positive detections I created a heatmap and then thresholded that map to identify vehicle positions. I then used scipy.ndimage.measurements.label() to identify individual blobs in the heatmap. I then

assumed each blob corresponded to a vehicle. I constructed bounding boxes to cover the area of each blob detected.
Here's an example result showing the heatmap from a series of frames of video, the result of scipy.ndimage.measurements.label() and the bounding boxes then overlaid on the last frame of video: (left image is heat map result and right image is final bounding box around cars)





Finally I applied the whole pipeline on video frames to detect cars. Here is a link to my video on youtube. The performance on video in good.

I also combined codes of this project with previous project on finding lane lines. Here is a link to my video.

You can also see result of rbf kernel svm here.

Discussion

Performance of this approach is not so bad but I think deep learning methods such as YOLO and SSD can perform faster and better on vehicle detection task. I also saw some efforts on using U-net for segmentation to detect cars. I really like to use deep learning methods to do the task and did this using yolo and keras. Here is a link to my video on this task using YOLO.