Vehicle Detection – term 1 – project 5

Writeup Template

By ~Karanj

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

My project includes the following files:

- Vehicle_Detection2.ipynb: Whole Project file
- project_video_output.mp4: Final Video output file.
- writeup_report_Vehicle_Detections.pdf summarizing the results

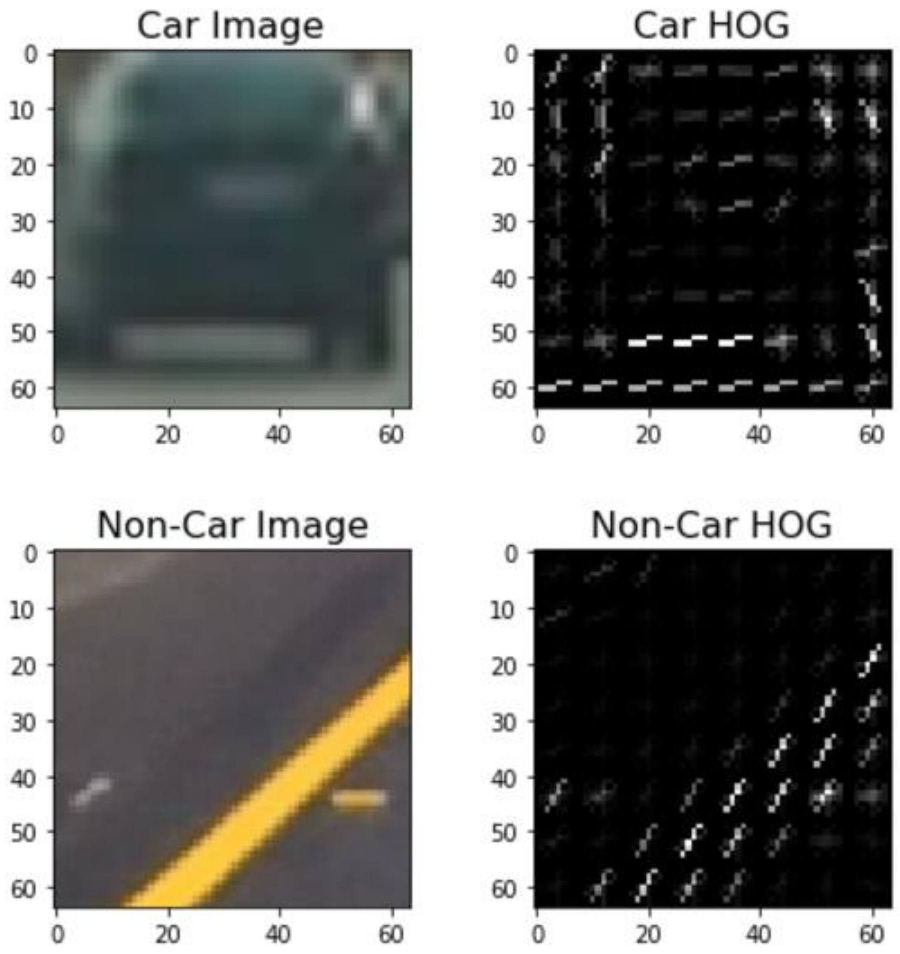
Important Note :

- Cells marked with *** (i.e. Cell12 and Cel 14...) are not called in final pipeline for video rather they are directly implemented in Cell 21

Histogram of Oriented Gradients (HOG)

1. Explain how (and identify where in your code) you extracted HOG features from the training images

- I Loaded all the Vehicles and non-vehicles images from the given sets (Cell 2)
- Cell 3: Visualization
- Cell 4 and 5: Extract HOG features and Visualization (single image from both sets)
  - Function “get_hog_features”



2. Explain how you settled on your final choice of HOG parameters.

- I settled on my final choice of HOG parameters based upon the performance of the SVM classifier produced using them. I considered not only the accuracy with which the classifier made predictions on the test dataset, but also the speed at which the classifier is able to make predictions.
  - color_space = 'YUV' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
  - orient = 11 # HOG orientations
  - pix_per_cell = 8 # HOG pixels per cell
  - cell_per_block = 2 # HOG cells per block
  - hog_channel = 'ALL' # Can be 0, 1, 2, or "ALL"
  - spatial_size = (32, 32) # Spatial binning dimensions
  - hist_bins = 32   # Number of histogram bins

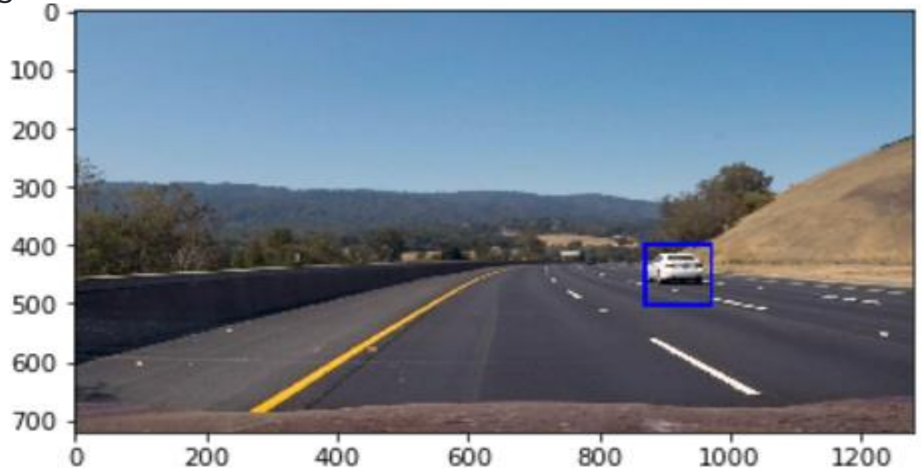
3. Describe how (and identify where in your code) you trained a classifier using your selected HOG features (and color features if you used them).

- Cell 6: Defined Spatial functions to extract Spatial Features and Color Histogram Features
- Cell 7: function "extract_features":
    - Apply color conversion if other than RGB
    - Fetch Spatial Features and color histogram features using functions defined in Cell 6
    - Call get_hog_features()
- Cell 8: Call "extract_feature" from Cell 7
    - Normalize all the features using StandardScaler().fit(X) and transform(X)
    - Split up data into randomized training and test sets
    - Use a linear SVC
    - Summary of Training:
        - Using: 11 orientations 8 pixels per cell and 2 cells per block
        - Feature vector length: 9636
        - 18.97 Seconds to train SVC...
        - Test Accuracy of SVC = 0.9885

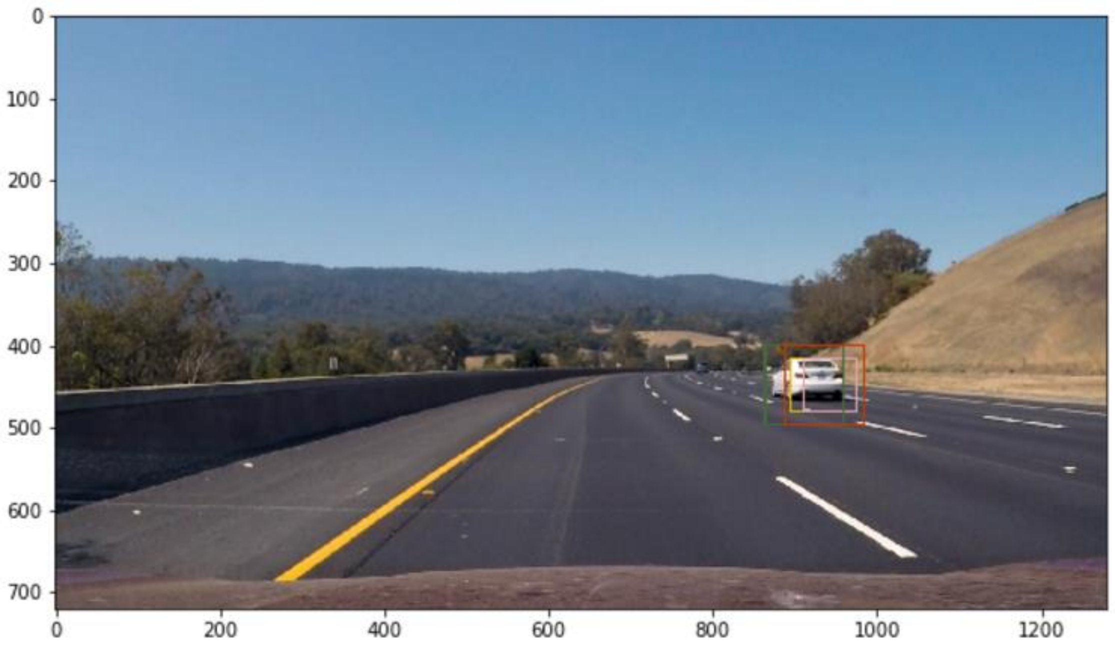
Sliding Window Search

1. Describe how (and identify where in your code) you implemented a sliding window search. How did you decide what scales to search and how much to overlap windows?

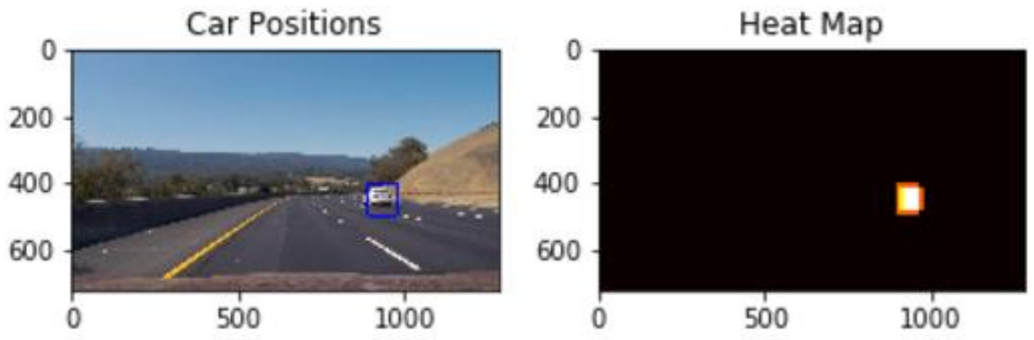
- In the section titled "Method for Using Classifier to Detect Cars in an Image" I adapted the method `find_cars` from the lesson materials. The method combines HOG feature extraction with a sliding window search, but rather than perform feature extraction on each window individually which can be time consuming, the HOG features are extracted for the entire image (or a selected portion of it) and then these full-image features are subsampled according to the size of the window and then fed to the classifier. The method performs the classifier prediction on the HOG features for each window region and returns a list of rectangle objects corresponding to the windows that generated a positive ("car") prediction.
- Cell 9: The image below shows the first attempt at using `find_cars` on one of the test images, using a single window size:



- Cell 10: draw_boxes function which iterates through the coordinates of boxes from find_cars function to draw rectangles.
- Cell 11: I explored several configurations of window sizes and positions, with various overlaps in the X and Y directions. Cell 11 shown with ‘scale = 3.5’.
- Cell 12: The final algorithm calls find_cars for each window scale and the rectangles returned from each method call are aggregated. The image below shows the rectangles returned by find_cars drawn onto one of the test images in the final implementation.

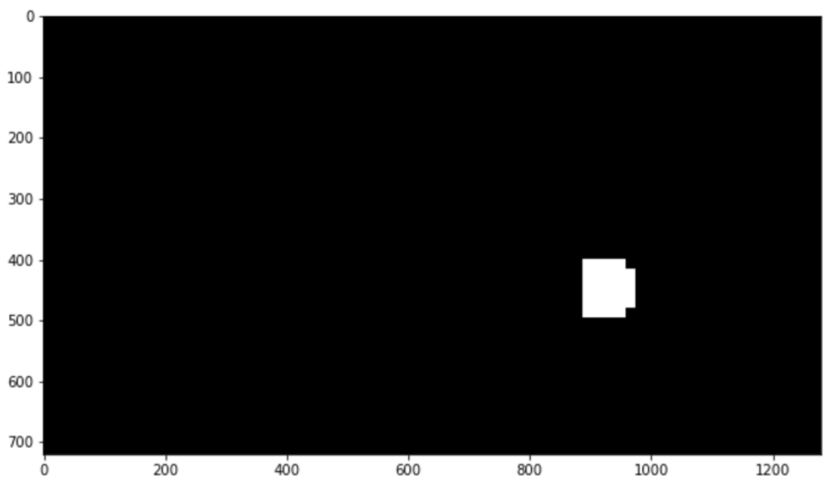


- Cell 13 : Because a true positive is typically accompanied by several positive detections, while false positives are typically accompanied by only one or two detections, a combined heatmap and threshold is used to differentiate the two. The add_heat function increments the pixel value (referred to as "heat") of an all-black image the size of the original image at the location of each detection rectangle. Areas encompassed by more overlapping rectangles are assigned higher levels of heat. The following image is the resulting heatmap with thresholding from the detections in the image above:

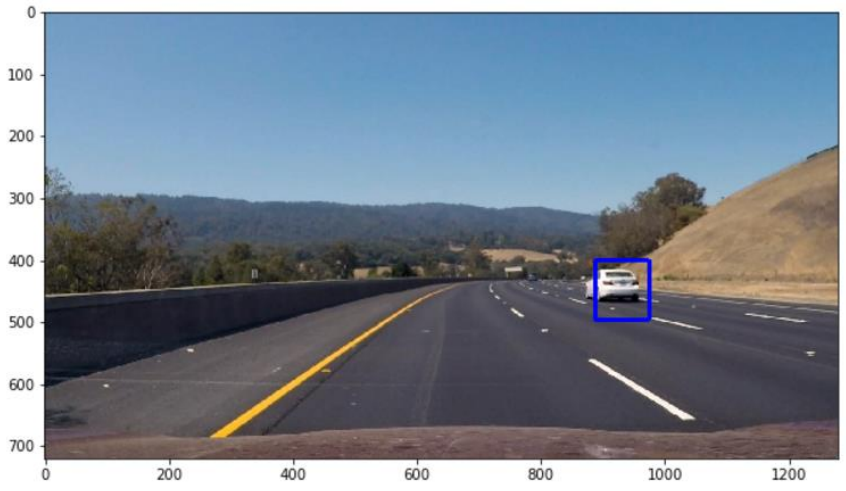


- Cell 14: The scipy.ndimage.measurements.label() function collects spatially contiguous areas of the heatmap and assigns each a label:

```
1 cars found
```

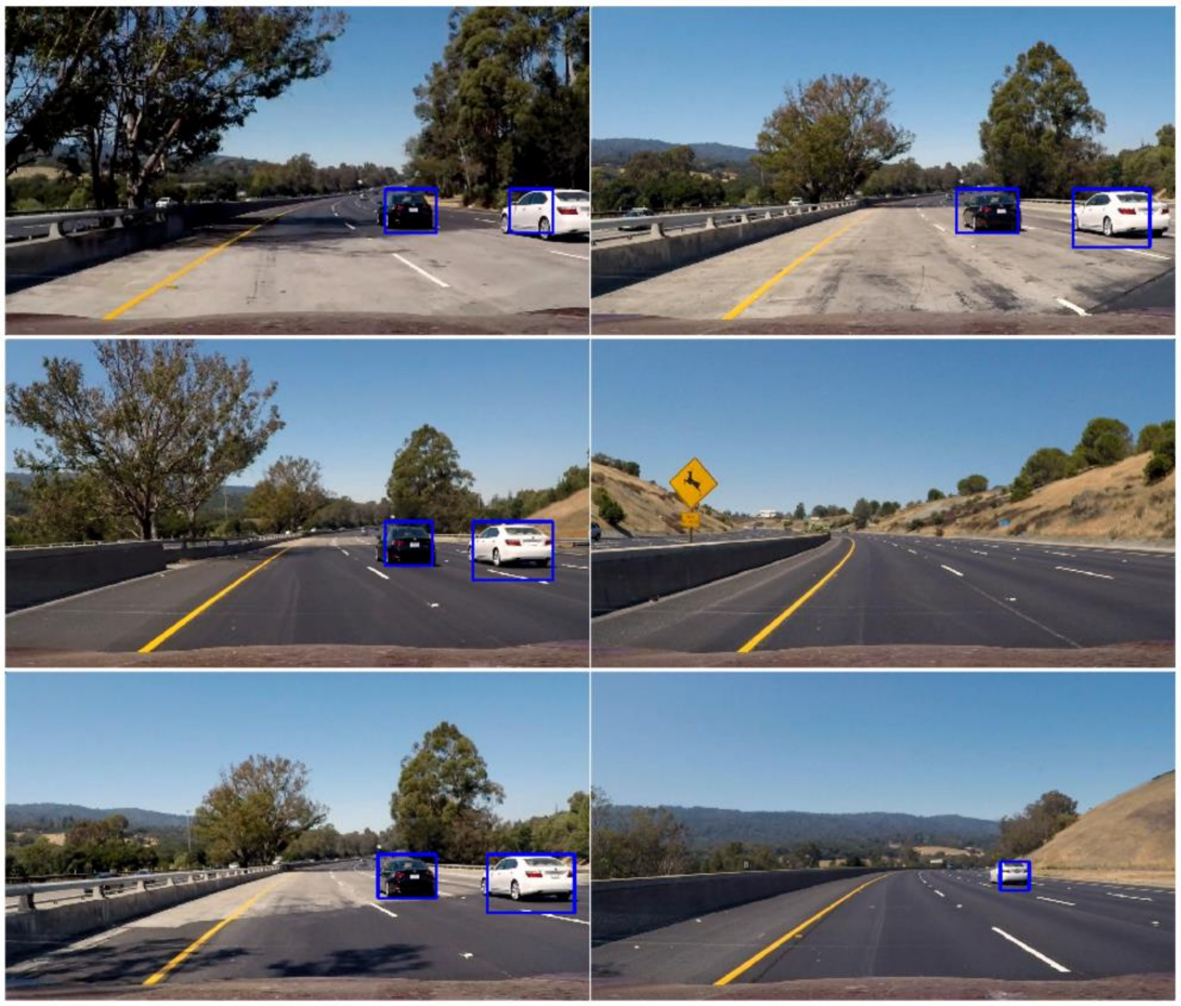


- Cell 15: Defined “draw_labeled_bboxes” and visualization of above image with detection area from heatmap:



2. Show some examples of test images to demonstrate how your pipeline is working. What did you do to optimize the performance of your classifier?

- Cell 16 and 17: using all given test images:



Video Implementation

1. Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video

- Here's a link to my video result

2. Describe how (and identify where in your code) you implemented some kind of filter for false positives and some method for combining overlapping bounding boxes.

- Cell 18 and 19: Cell 18 stores data from video about previous frames. Cell 19 is final pipeline used for videos and it similar to Cell12 and Cell 16 which processes single images or set of images. This also stores the detections (returned by find_cars) from the previous 15 frames of video using the prev_rects parameter from a class called Vehicle_Detect. Rather than performing the heatmap/threshold/label steps for the current frame's detections, the detections for the past 15 frames are combined and added to the heatmap and the threshold for the heatmap is set to 1 + len(det.prev_rects)//2(one more than half the number of rectangle sets contained in the history) - this value was found to perform best empirically and practically. Else, we can also use threshold equal to 7 or 8.

Project Discussion:

- Initially, I tried varying through the parameters combinations:
    - Color schemes tried: YcrCb, HSV, LUV, YUV
        - Was throwing error while using LUV due to use of PNG image. Discussion.
    - Tried various 'orient' and 'pix_per_cell'. Though doesn't seem to make big difference. But yes it may be very important depending upon intensity of variedness of Gradients.
    - Pipeline may fail where environment or/and Vehicles are not similar to training set.
    - Distant cars are still issue as they are not very distinctive from background environments and settings.
    - To improvise accuracy, it would require high accuracy classifier, more training data and combining it with CNN.