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.

Video Link: https://vimeo.com/213769552

Rubric Points

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

I extracted the HOG features in both the SVM_model.py and v_detection.py scripts. I pulled them after I converted to whatever image size and color space I was using.

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

First I took a look through the data just as Udacity suggested. Following that I tried several different color spaces and found YCrCb to be the best. This color space yielded the best accuracy when fed to the model as well as the best performance in the video. My final parameters are below.

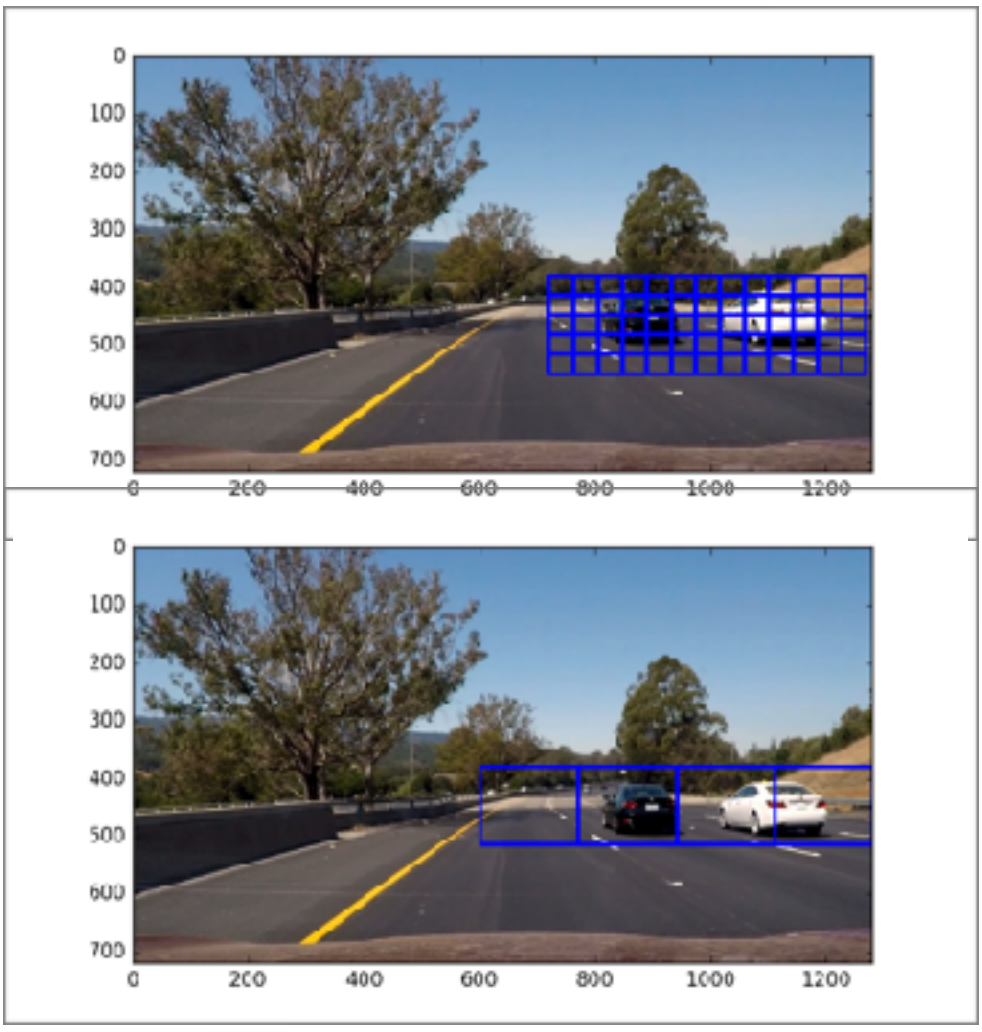
Orient=8
pix_per_cell = 8
cell_per_block = 2

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).

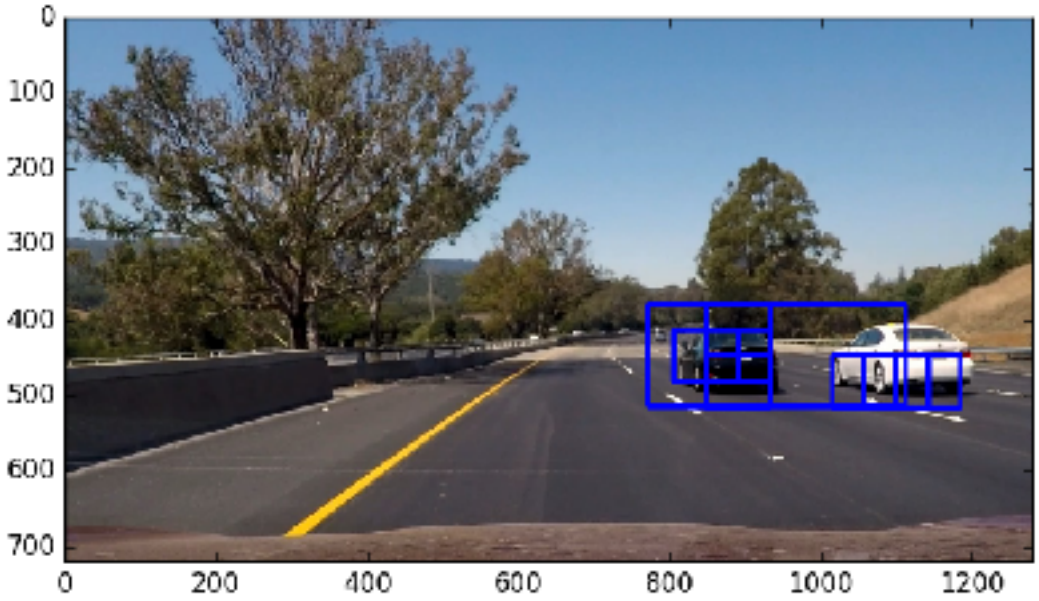
I trained a linear SVM in SVM_model.py. I combined this data with spatial binning and a color histogram. All these features used the same color space (YCrCb).

4. 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?

I implemented the sliding window search in the v_detection.py file. I chose my windows by taking a look at some snapshots of the video and overlaying the boxes I was thinking about using. I tried to position the boxes where I expected a car to be, not just canvas the entire image looking for cars. Below is an image of the final boxes. The box dimensions can be seen in the v_detection file.



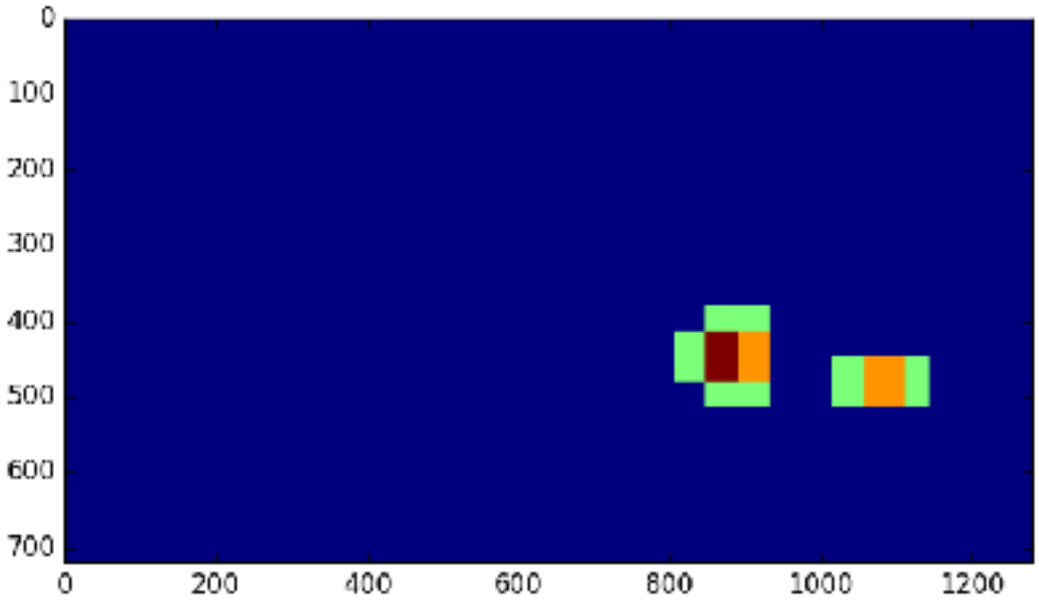
5.Show some examples of test images to demonstrate how your pipeline is working.



What did you do to optimize the performance of your classifier?

Step 1: Hot windows, the algorithm searches for matches inside the set boxes.

Step 2: Heat map and threshold, a heat map is used to eliminated noise. Additionally a threshold is applied to further aid in that effort.



6. Video implementation

My video is supplied in the zip folder. I used a running average of points to filter out false positives from my video pipeline. This means I took a running average of the bounding boxes supplied by the heat map. The running average holds 15 frames.

7. Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail? What could you do to make it more robust?

The largest problems I faced were with building the classifier. I found the data to be tough to work with and I had to write a randomization scheme to avoid the issues caused by video feed supplied data (i.e. overfitting).

My pipeline is likely to fail if the car is not in the left lane. To increase run time I specifically targeted the area I suspected the car to be. I could change the box area but it would slow down the pipeline.