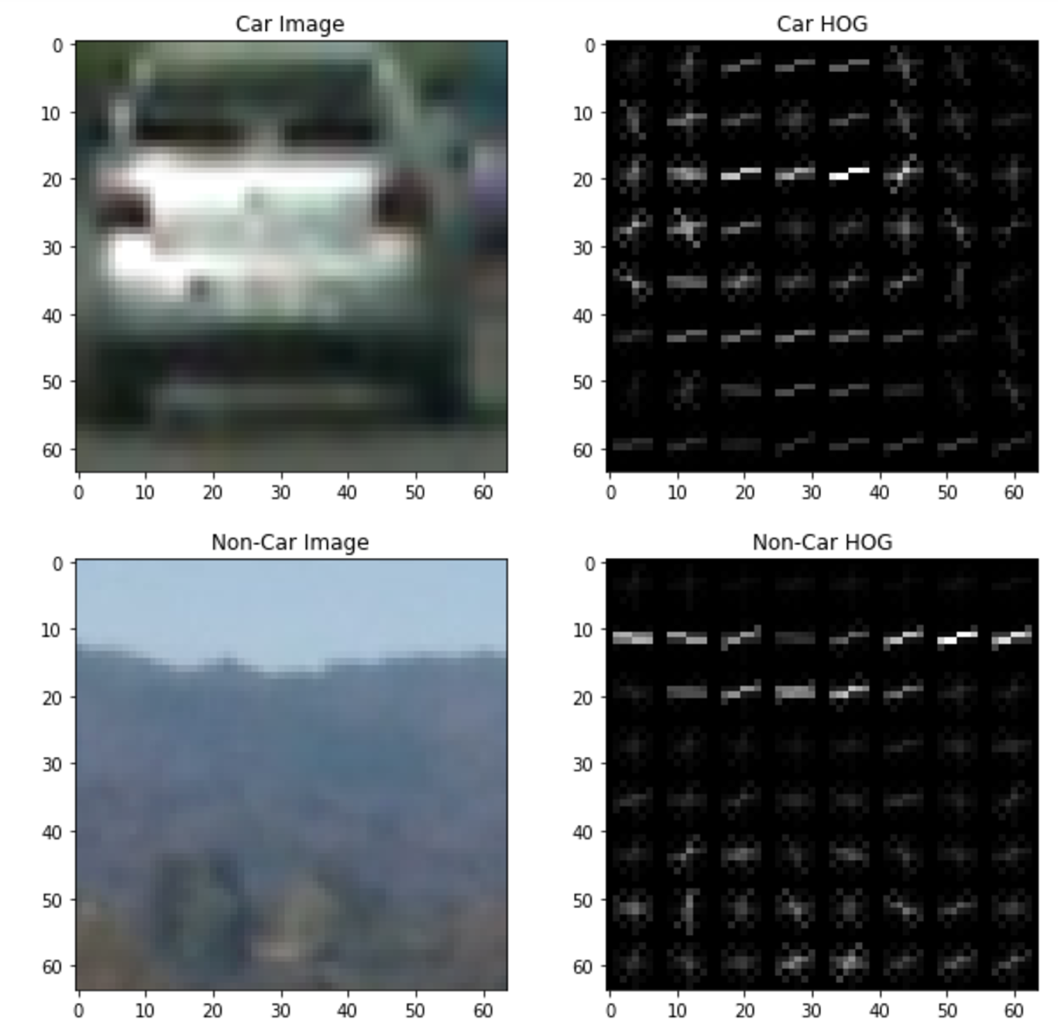
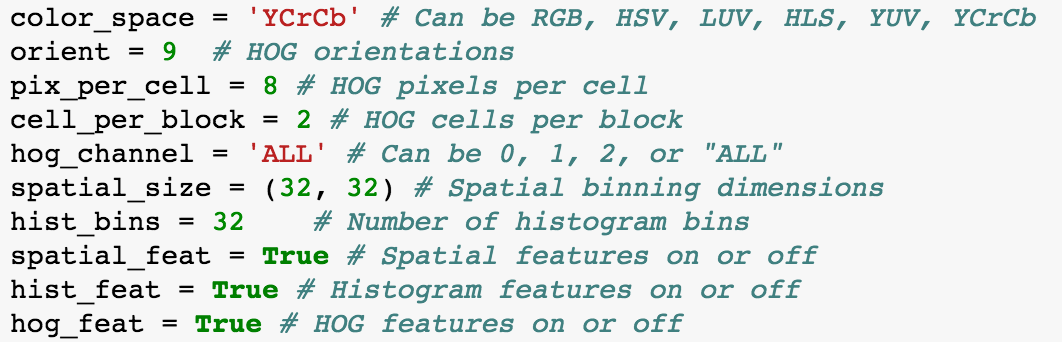
**Explain how (and identify where in your code) you extracted HOG features from the training images. Explain how you settled on your final choice of HOG parameters.**

The get\_hog\_features function is defined to extract HOG features from the training images. I tried various combinations of the parameters and settled with orient=9, pix\_per\_cell=cell\_per\_block=8 as the final choice. Below is an example of the original/HOG images of a car and a non-car.



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

The extract\_features function combines HOG features with spatial (bin\_spatial)and color channel (color\_hist) features. I split 20% dataset as the test set and use the rest 80% as the training set. As mentioned in the previous question, the parameters are tuned constantly to achieve the highest accuracy rate and lowest time spending. The final parameters are shown below and it gives me 98.93% accuracy rate and reasonable time to run the model (135.57 seconds to extract and 31.45 seconds to train the SVC model).

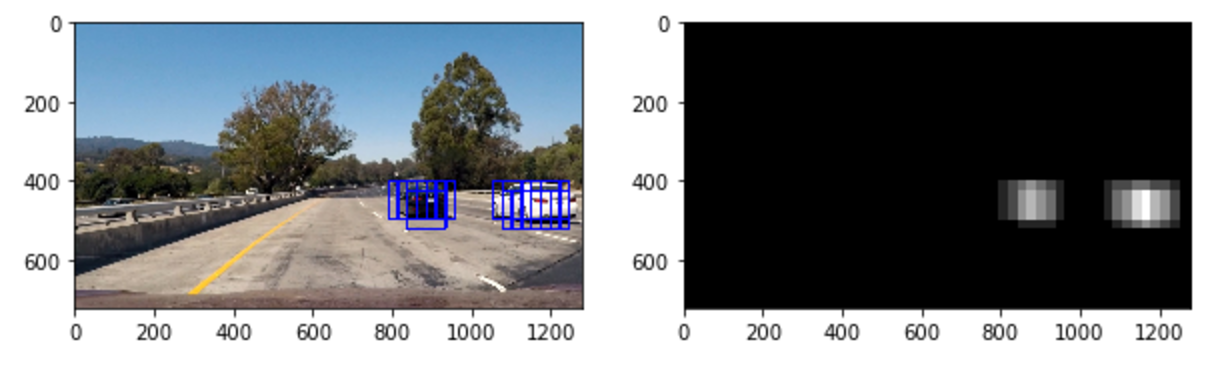


**Describe how (and identify where in your code) you implemented a sliding window search. How did you decide what scales to search and how much to overlap windows?**

I define a find\_car function (adapted from the course material) to perform the sliding window search and feature extraction. The function extracts the feature once within the defined area [400,650] and those features are sub-sampled according to the window size. Scale=1.5 and cells\_per\_step=2 (75% overlap) appear to be the combination that produces the best result to capture vehicles and prevent false positives.

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

As mentioned in the previous question, I optimize the perfomrnce of the classifier by restricting the search area in the [400,650] in the y-axis.



**Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (somewhat wobbly or unstable bounding boxes are ok as long as you are identifying the vehicles most of the time with minimal false positives.)**

Please see the video attached.

**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.**

I implement a threshold method to prevent false positives. 10 sequential frames are stored and a threshold of 5 is used to identify the true positives. I then draw the boundary of the vehicle detection box based on the threshold result. The image\_process function combines all the previous steps and is used to produce the video output.

**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?**

This version improves significantly compared with the previous version by applying the threshold method.