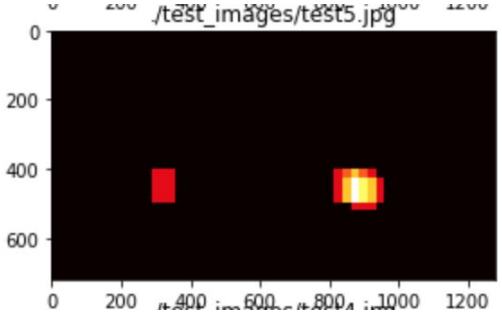
Vehicle Detection

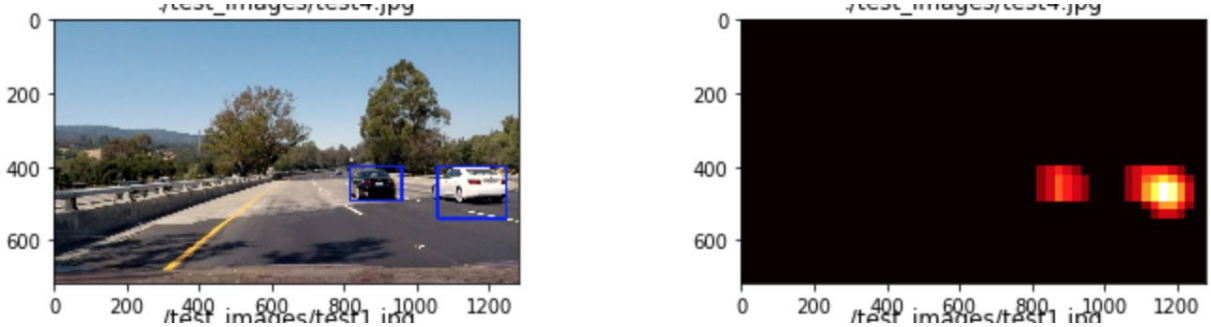
HOG Parameters: I chose the feature parameters by experimenting and training various classifiers to chose one with the best accuracy. I wanted to try to mitigate some of the overtraining aspects that were mentioned in the lessons so I chose smaller image sizes, histogram bins and pixels per cell. As far as color space, I just went through each of the color spaces and chose one with the highest accuracy. The final parameters:

color_space = 'YCrCb'
orient = 9
pix_per_cell = 8
cell_per_block = 2
hog_channel = 'ALL'
spatial_size = (16, 16)
hist_bins = 16
spatial_feat = True
hist_feat = True
hog_feat = True

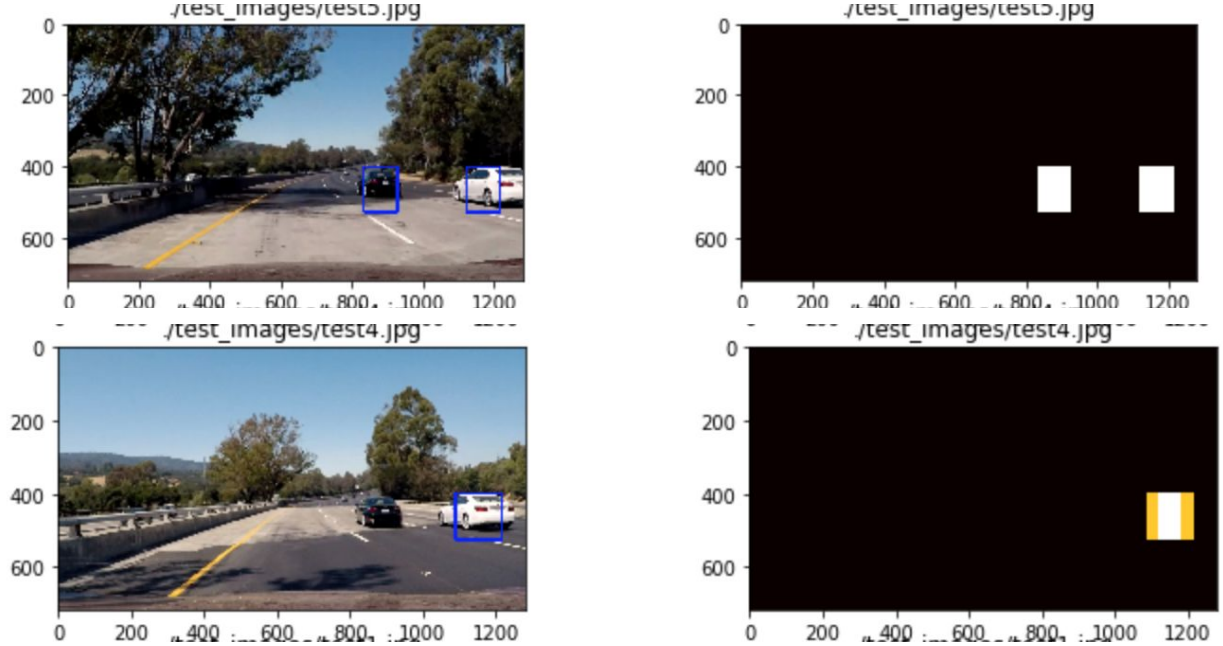
Classifier Training: I took the images provided, the udacity data as well as some screen caps that I took directly from the video manually. After adding the additional data there was a huge imbalance between the two classes so I chose random indexes from the car data to balance the training set. It was a lot of data, total of 17936 images of vehicles and non-vehicles. Then I had to go through and choose a classifier starting with the NaiveBayes mentioned in the lesson, went through LinearSVC, DecisionTrees, LinearSVR and vanilla SVR. Basically I just wanted to chose the classifier with the highest accuracy based on the training data so I ended up using the LinearSCV. (LinearSVR was slightly more accurate but it was insanely slow to use when processing the video files). The final accuracy of the LinearSVC was 99.44%.

Sliding Window: As suggested, I did the HOG transformation on the area of interest and then slide the windows through the resulting image instead of taking individual transformations. Certain scales would result in false positives and missed detections
Window Scale 1.5:

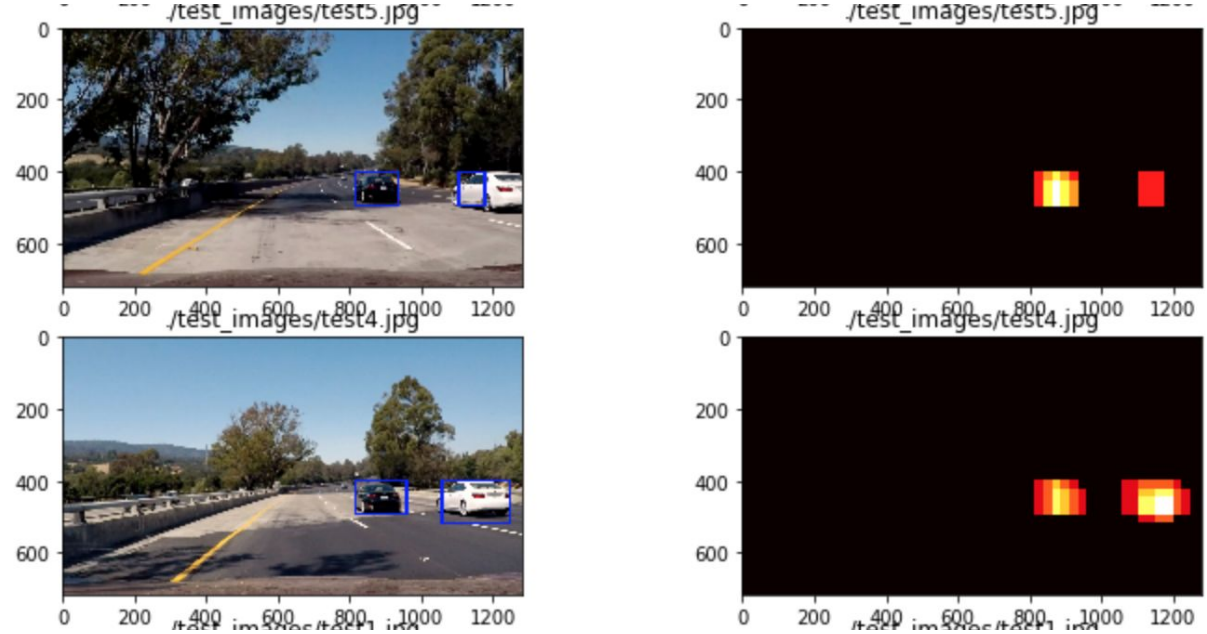




Window Scale 2.0:

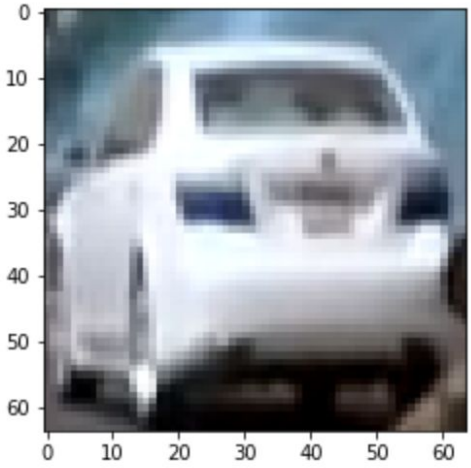


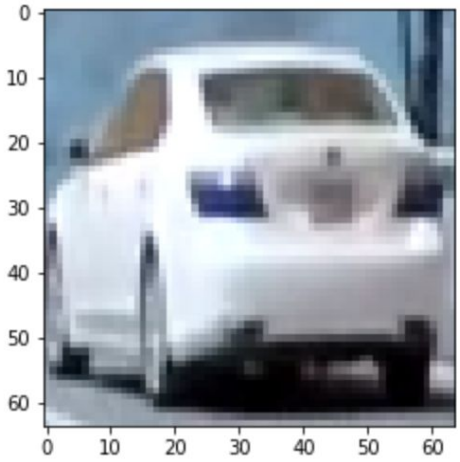
2.0 scale missed one of the cars, 1.5 had a false positive, a combination of 1.5 and 3.0:



Optimizing the Classifier: I tried to improve the training of the classifier by including the udacity data in the training set. Having more images of different cars should make the classifier more extensible to different car sizes, colors and lighting conditions. I also wanted to improve the performance throughout the mid section of the project video by including manual copies of the white car in those situations and I got better results after adding these examples. I had the best results using the SVR but processing the video took over an hour so that was a little unreasonable to work with. I tried balancing the car/notcar data by taking a random sample of the car data to match the size of the not car data and I got better results using this method.

Manual screen caps:





They look pretty much the same and I duplicated them and added them to the training data.

False Positives: To get rid of the false positives and try to keep the bounding boxes more consistent, I kept a running average of the heatmap and then ran a thresholding function over the averaged heatmap. This was very effective in getting rid of false positives but I still think the classifier could be trained better to eliminate more false positives. It lost the white car around 30 seconds for a bit so I think playing with the classifier parameters would result in a more accurate classifier and better results.

Improving the Pipeline: Writing the multiple sliding window approach caused the HOG features to be extracted for each window size and it ran slower as a result. The classifier also had a major role in the video processing speed. When I used the SVR the 50 second video took over and hour to process, with the LinearSVC it took around 5 minutes. What was really interesting to me was that when I trained the classifier with less data, it took longer to process the video but it also gave better results. This kind of confirmed my thought that the classifier is overtraining from the data. I would be very interested in how people got near real-time results, the feature extraction code could definitely be faster but it seems like the classifier is a big bottleneck. Adding the udacity data would hopefully make the classifier more extensible for different cars and situations but I wonder if a machine learning approach and even more training data would get better results.