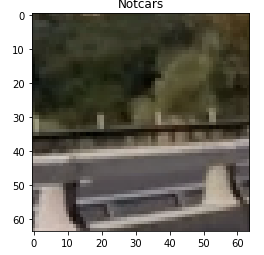
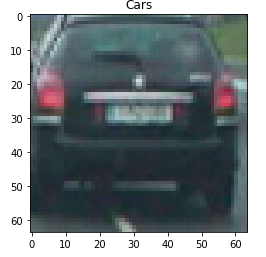
**Histogram of Oriented Gradients (HOG)**

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

First, we have created two list: one for car images and other for notcar images. There are 8792 images for cars and 8968 for notcars. We ploted one image for each:



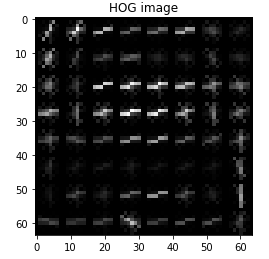
To extract HOG features we used the function provided by Udacity, parameters for this extraction are:

*orient = 9 # HOG orientations*

*pix\_per\_cell = 8 # HOG pixels per cell*

*cell\_per\_block = 2 # HOG cells per block*

The return of this function (‘*get\_hog\_features’*) are *features* and *hog\_image*. The image provided by function is like this:



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

I tried various combinations of parameters:

**FIRST**

*orient = 9 # HOG orientations*

*pix\_per\_cell = 9 # HOG pixels per cell*

*cell\_per\_block = 1 # HOG cells per block*

**SECONTLY***orient = 8 # HOG orientations*

*pix\_per\_cell = 12 # HOG pixels per cell*

*cell\_per\_block = 2 # HOG cells per block*

**LAST***orient = 9 # HOG orientations*

*pix\_per\_cell = 8 # HOG pixels per cell*

*cell\_per\_block = 2 # HOG cells per block*

I used this last configuration since computationally is slower than others but for me have a good results.

**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 with HOG, Spatial and Histogram features of the list of cars and notcars. We extract features of each image in this code lines 220-269. Then we train the model in code lines 312-338.

**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?**

I used functions provided by Udacity to implement sliding window search, code lines 347-440. I decided to implement a *search\_window\_size* (provided by: [*https://github.com/Dalaska/CarND-P15-Vehicle-Detection-and-Tracking/blob/master/Vehicle\_Detection.ipynb*)](https://github.com/Dalaska/CarND-P15-Vehicle-Detection-and-Tracking/blob/master/Vehicle_Detection.ipynb)) to set the scales to search.

#### 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?

We are using HOG features, Spatial features and Histogram features to make work the pipeline. The result is very good and I’m very happy with this. Here, we can see 6 test images with the heat map applied and boxes drew.

#### 

### Video Implementation

#### 1. 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.)

The video is attached.

#### 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.

To detect false positive we used heat map (see the last figure) and threshold function, Line codes 458-473. We detect cars in each frame of the video and save the heat array to the next frame, finally we have cars identified. The final result is this:  
  


### Discussion

#### 1. 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?

#### I firmly believe that this code works really well but we can improve it with another SVM classifier but perhaps this make the code slower. Also we can further work in to set parameters to get a better result.

#### The classifiers still get many false positives, although the accuracy is really good. Maybe, the fences can be confusing to car images. If we set another threshold on heatmap we can reduce this false positives but we also can miss some information.