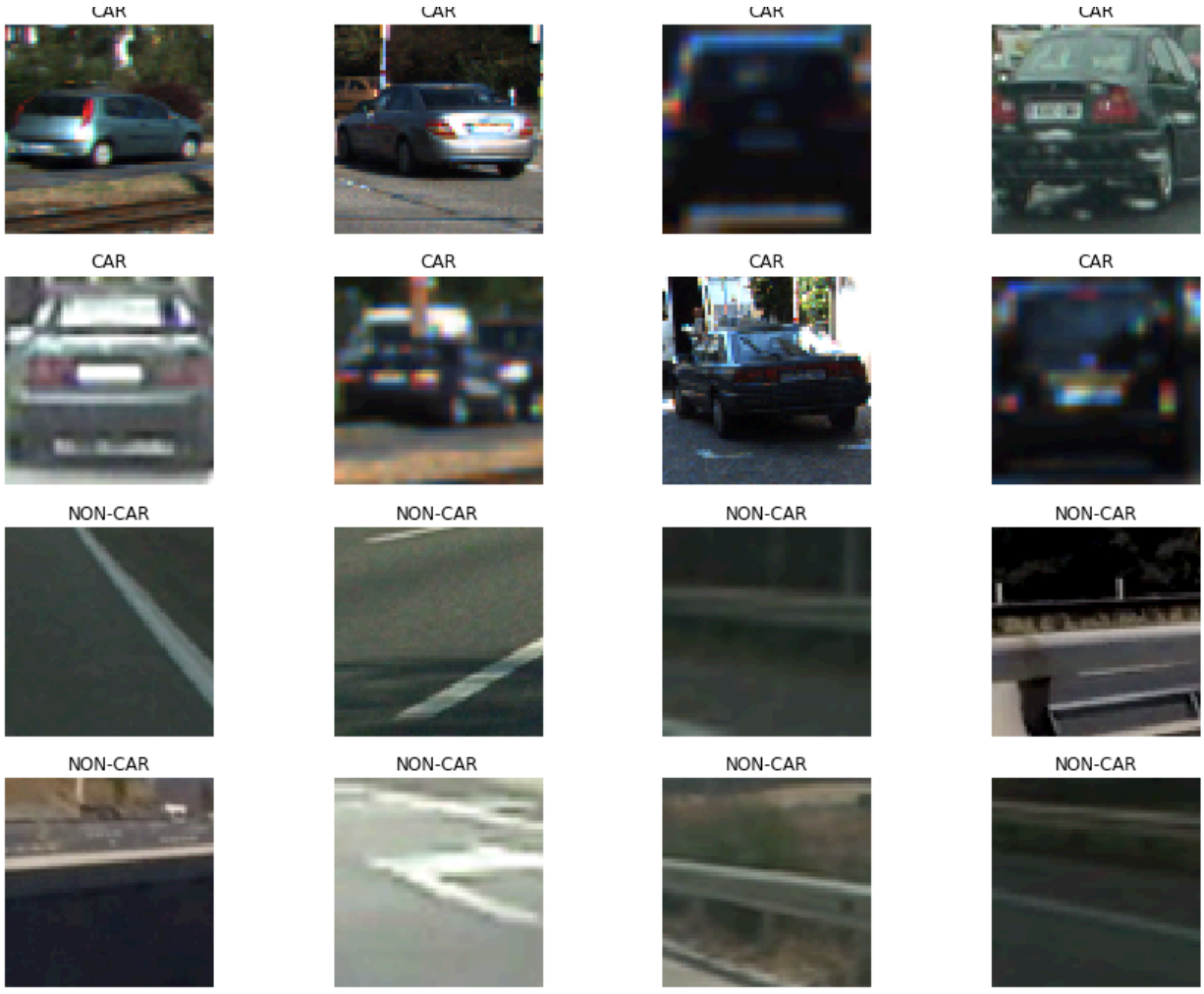
Vehicle Detection Project

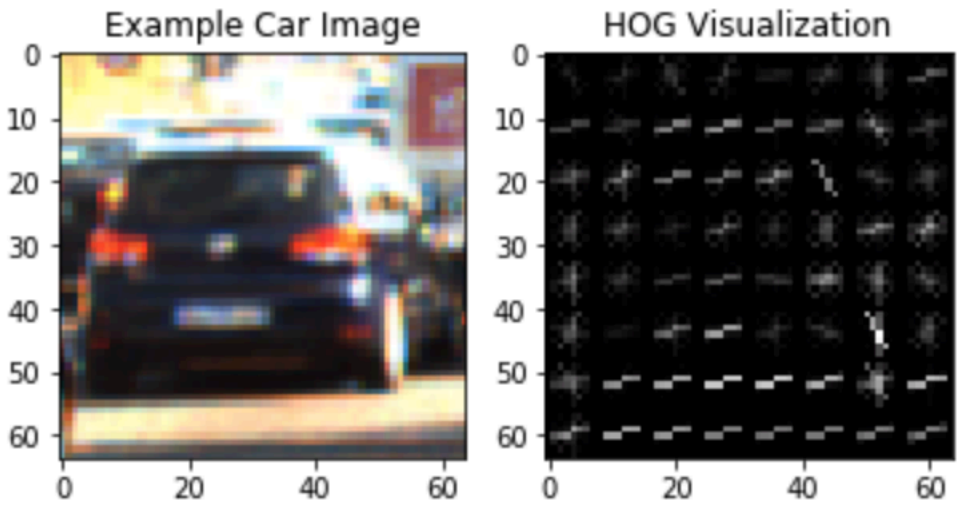
Histogram of Oriented Gradients (HOG)

HOG features from the training images

This step is contained in first 5 cells of project notebook (project.ipynb)
In the first cell required imports are done. In the second cell functions in the lesson which will be used in this project is implemented.
In the third cell car and non-car image file names are loaded. In the fourth cell, some of these images are displayed. Some sample images are in the figure below.



In the fifth cell some hog features are displayed to understand how HOG features are generated. Mostly the code from the lessons are used again. Different parameters are tested to understand working principle. Below is a sample output.



In this output 9 orientations are used. Pixels per cell is 8 and cells per block is 2.

Final choice of HOG parameters

I used above HOG parameters. I have set these parameters after testing whole pipeline. I tried several parameters but these parameters resulted in better performance in training and resulting test with the provided test images.

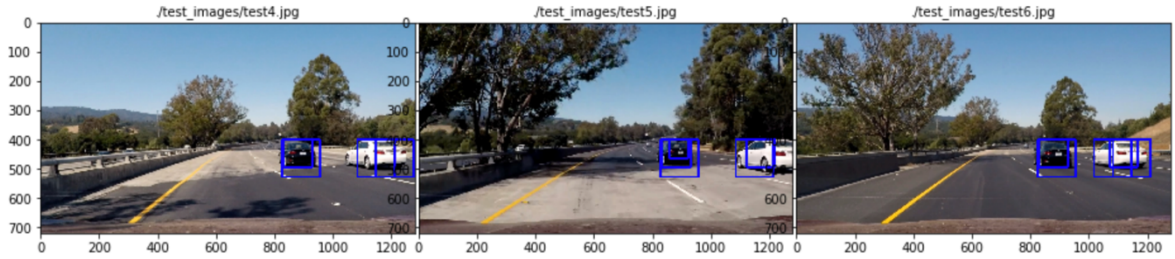
Training the classifier

I used linear SVM for classification. The initial data is split to test and training sets. I tried many color spaces but get the best result in test images with YCrCb. I have reach test accuracy of 0,9901. Following are the parameters used in training.

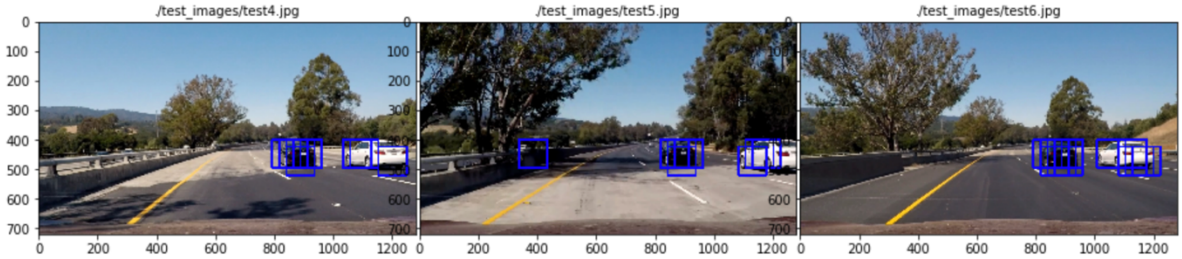
```
color_space = 'YCrCb' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
orient = 9  # HOG orientations
pix_per_cell = 8 # HOG pixels per cell
cell_per_block = 2 # HOG cells per block
hog_channel = "ALL" # Can be 0, 1, 2, or "ALL"
spatial_size = (32, 32) # Spatial binning dimensions
hist_bins = 32    # Number of histogram bins
spatial_feat = True # Spatial features on or off
hist_feat = True # Histogram features on or off
hog_feat = True # HOG features on or off
```

Sliding Window Search

First I tried sliding window search technique from the lesson. I have tried three different search windows with different start stop points. I tried to match them with size of the cars in test images. I used 64x64 , 96x96 and 128x128 windows. I reached the following result.



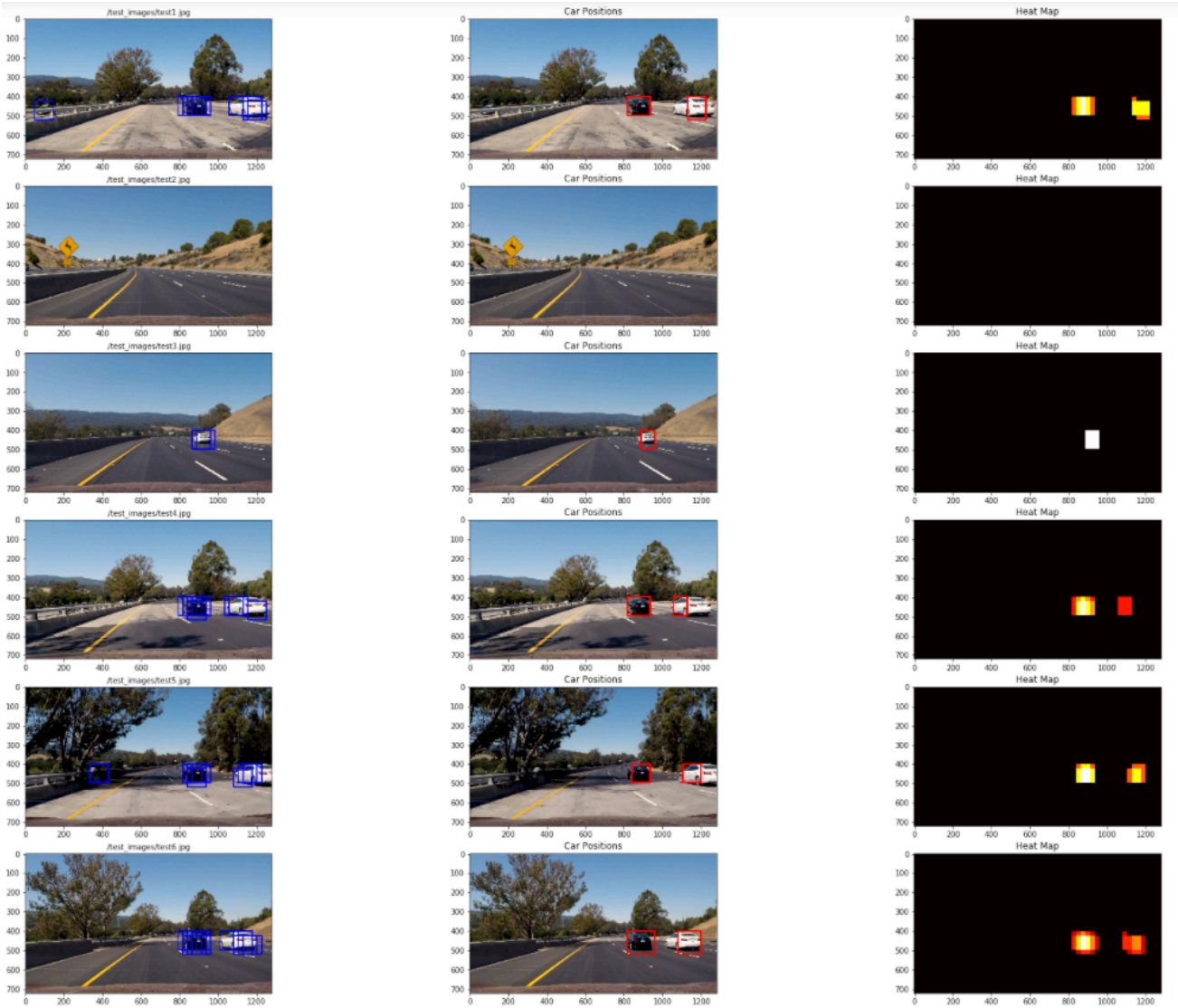
After this I implemented Hug Sub-sampling Window Search as explained in lessons. When I compared the performance with the above method I have seen that in produces better results. Result of this implementation on test images are as follows.



When I compared both methods. In the first method difference with false positives and positives are very little. The second approach generates better results.

Video Implementation

To create bounding boxes for the cars and eliminated false positives heatmap method in the lesson is used. Following is the output of the implementation.



I tried the same methodology on video. But I have seen some false positives and missing detection. I saw that in some consecutive frames even though the scene is not changed number of boxes are small. I decided that using some number of frames boxes in the detection. After some trials I have seen that using last 8 detections produces better results. I have saved the video output to “project_out_video.mp4”.

Discussion

Most of the time methods in the lessons were enough.

- The performance of the systems is seems low it may not be suitable for real time implementation.
- Close by and occluded cars are creating detection problems.
- Most of the parameters and thresholds are empirical values at the moment. They may be effected from lighting conditions.
- A better training set may result in better performance especial for boundary conditions.
- I tried with the extra data results were not good, better and generic algorithms would be required for handling these kinds of conditions. Following example show the result of the extra test