Vehicle Detection and Tracking

- The car images and non-car images are collected to be trained by the classifier.
- 8792 images are present for car images and 8968 images as non car images.
- Data Exploration is done to show the resolution of images and examples of both car and non-car images.

Histograms of Color

•I have tried different color spaces of histogram and "YCrCb" seems to provide the distinctiveness in car and non car images.
•Different histogram color spaces have been plotted below.
•I tried applying different color spaces as features and YCrCb appears to perform better than HSV and HLS.

Spatial Binning

•When handling lot of images, it is always important to do Spatial Binning and the image is reduced to (32,32,3) dimension to fasten the processing.
•Different color spaces like HSV, HLS and LUV are tried to create the feature vector but "YCrCb" shows pure distinctive nature to recognize car and non car images.
•color_hist is the general histogram function.

HOG Features

•Hog features are chosen from scikit library as mentioned in the class.
•Sample examples are shown below.
•When i choose orient to be 9, the trailings or barriers beside the road is classified into car. When i increase the orient value, the classification is better. I went up to 21 but it is not necessary i felt. So stuck with 19 and i get correct classification of cars in the same direction.
•When i dont scale the features to zero, then orient value can be set to 11 and the cars get classified properly. Since it is a requirement to scale the features to zero, I changed the orient values accordingly by testing it.
•Pixels per cell and Cells per block are kept same as in the lesson.
•Y Channel in YCrCb provides distinctive features with respect to car and non-car images.
•In extract_features function, Iam just using hog features.

Classifier

•SVM is used as classifier.
•The Standard Scaler function is used to normalize the data with 0 mean and equal variance.
•I chose SVM because it does not require much effort to tune the model because there are not many hyperparameters in the linear SVM to tune except the cost.
•I tried without normalizing but SVM classifies fine with lesser orientation values of hog features.
•Train Test Split of 80:20 is chosen.
•I felt SVM is the best classifier in this scenario and gave the best accuracy.
•I might try to ensemble the model with others but since I got good accuracy, i stuck with linear SVM.
•The accuracy is 98.4 percent on the test data as shown below in the result.
•Time taken to train and test with some data is presented.

Find Cars

•This is the most important function which combines all the features put together to build the pipeline.
•Spatial and Histogram features are not considered in this work.
•Hog features are considered.
•Feature vectors are constructed by hog features and fed into the trained SVM model to obtain the test prediction.
•ystart is chosen to be 400 in the image as per lesson and yend also.
•Scale is chosen to be 1.5 because 1 is too small box and 2 is very long.
•Sliding Window is also included in this with respect to choosing the starting and ending point of processing and deciding on the x and y positions.
•The blocks are divided and number of blocks per window is calculated. The number of windows chosen is 64.
•The number of cells per step was taken manually to be 2. Other parameters are calculated accordingly.
•For each block the hog feature is classified using the classifier and predicted the result of car and non-car. This approach would also help solve the overlapping window problem.
•Since this method is incorporated and works well, search window is also inside by searching through each block by block for features and it is classified by SVM by using HOG features.

Draw Boxes

•The rectangles returned from find cars function are drawn on a test image.
•To make sure of the positions, heat map is implemented.
•Since there are more rectangles and apply threshold function would give us the absolute count of number of cars detected on the track.
•This is then depicted in draw labelled boxes from the thresholded image and in the example image it is 2 cars found.

Process frame pipeline

•This is again hardcoded seeing the images while testing.
•When the cars are found far from the viewer, then scale selected is 1, then relatively it is high if it is little near which is chosen to be 1.5.
•When the cars are found close, the scale chosen is 3.
•These rectangles are appended and then used for seach in all the images.
•The measurements are according to windows found in the image. Since it is chosen to be 64, I have chosen for scale 1 to have difference of 64, for scale value 2, i have chosen 128 and for scale value 3, i have chosen 192 for ystart and ystop difference values.
•It is then applied on the video as per shown.

Discussions

- One of the difficulties i faced was converting the images of png to jpg while processing.
- I could not implement perfectly the sliding window method and instead chose the manual processing.
- Improvements would be to try with dark backgrounds with different color channels to identify features.
- I would also try to ensemble the classification models and might also use neural networks for classification.

- If the car changes the lane, it will be interesting to watch the detections in case of hypothetical situation.
- When the car is too close to the right, it would fail to detect.

```
[ ]:
```