Machine Learning Engineer Nanodegree
Udacity

Dibakar Sigdel

May 29, 2017

Capstone Project Report

Contents

1 Definition

1.1 Project Overview

Object detection[1] is a rapidly growing field in computer vision for finding and identifying objects in an image or video sequence. Humans recognize a multitude of objects in an images with little effort, despite the fact that the image of the objects may vary somewhat in different viewpoints, in many different sizes and scales or even when they are translated or rotated. Objects can even be recognized when they are partially obstructed from view. This task is still a challenge for computer vision systems. Many approaches to the task have been implemented over multiple decades.

Properly identifying objects through object detection technique is a very important problem in computer vision. For example: human detection [2] in surveillance cameras, face and facial expression detection[3] in speech videos etc. Our main goal for this project is identifying cars in a video which is a fundamental step in achieving vehicle automation and ultimately fully driverless cars.

This project is inspired by the Vehicle Detection and Tracking Project that is a part of Udacitys Self Driving Car Nanodegree. The link for the dataset is provided by Udacity. This project will build a classifier using two sets of data called vehicle and non vehicle. The data set includes 8792 cars and 8968 non-cars. Each image is of size 64 by 64 with three color channels. Once a model is ready, it will be used to test its performance on test video. Following are the data sources:

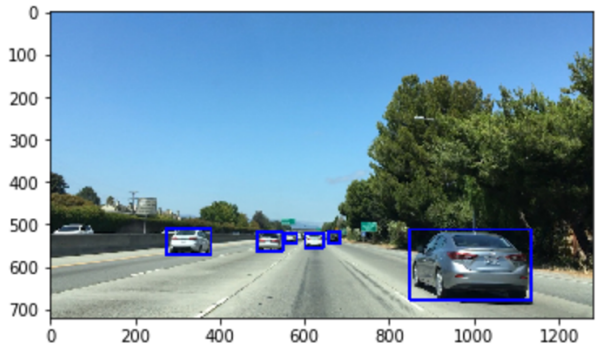
1. Vehicle Images: vehicle-data-link
2. Non-vehicle Images: non-vehicle-data-link
3. Test Video: test-video-link

The problem of detecting vehicle in the test video will be solved by using techniques of Computer Vision for feature extraction and and Machine Learning for search of best classifier model. Feature extraction steps will include use

of the concept of Histograms of Oriented Gradients HOG, and color features like color histogeam and spatial distribution of pixels. A set of features from each images is extracted and then used to train a classifier. In this project we will use a linear support vector machine (SVM) and Neural Network. Once trained with the labeled data the obtained model act as our classifier, predicting if new images contain cars or not.

1.2 Problem Statement

The statement of the problem is **Vehicle detection and tracking in a video using classifier trained over vehicle and non vehicle image data set**. In order to achieve this, multiple computer vision and machine learning methods are used in conjunction. The solution to the problem is that, it should be able to select proper pixels in the image frames in a video as vehicle and draw a rectangular boundary as shown below.



More technically, the accuracy of the classifier will be calculated with respect to train, test data as well as the performance of classifier on videos. The results thus obtained should be replicable using same data set.

In the implementation of this project, labeled images(cars and non-cars) will be processed and represented by some of their feature vectors. The images are separated by the presence of a car in the image or lack of car. By having labeled data, we can differentiate the features of an image that has a car and one that does not. These features can then be used to train a classifier such

that when given a new image it can predict weather or not there is an car in the given image.

Technically, the solution to this problem will be an algorithmic pipeline that will result in a video (which is just a sequence of images) with bounding boxes around the cars.

1.3 Metrics

There are two metrics that we will use to quantify different components of this project. First we will measure the accuracy of the classifier used on the training and testing data. Secondly we will measure the performance of our algorithm as a whole by calculating the accuracy of the vehicle detection in the final video result. Since we are using a binary classifier and we want to account for false-positives we will use the following formula to calculate accuracy:

$$\text{Accuracy} = \frac{(\text{true positives} + \text{true negatives})}{\text{size of dataset}} \tag{1}$$

Here, true positives are cars identified correctly. True negatives are image segments without a car. The size of the dataset is the total number of image segments classified. It has been mentioned that accuracy as a metric is suffered when the dataset is very imbalanced. This is not the case for the dataset in this project. Our dataset is divided into two classes (car and non-car) of equal size. For the implementation of this project we only extract features from exactly 8000 images from each class, making our data completely balanced.

2 Analysis

2.1 Data Exploration

The dataset used in this project is composed of thousands of labeled 8792 cars and 8968 non-cars images. All on these images are 64 x 64 pixels which provides a consistent number of features extracted from each image. Here we present few samples of the images from the dataset.

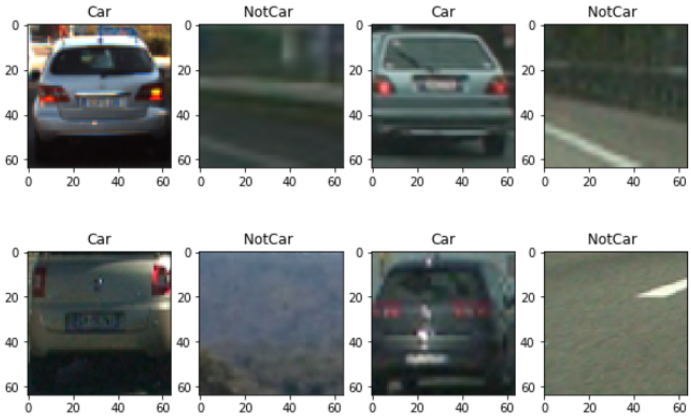


Figure 1: Car and Not-car sample images

2.2 Exploratory visualization

2.2.1 HOG feature

The first set of features extracted from our image dataset is obtained by using a Histogram of Oriented Gradients (HOG). A HOG feature extraction is done by calculating the magnitude and direction of each pixel using scikit-image python library. These are calculated from the set of pixels surrounding a given pixel. Looking at these values we can calculate a gradient. This gradient is useful on getting information about what is represented in an image. Then we segment the original image into cells and create a histogram for

the gradient values of each pixel. These values are then put into orientation bins on a histogram. The bin containing the largest value is our dominating orientation, but all of the bins are taken into consideration when calculating the orientation of the image segment. The histogram now gives us a sense of what is in image cell, and once do the same calculations for each cell of the original image, we get a feature set representing our original image.

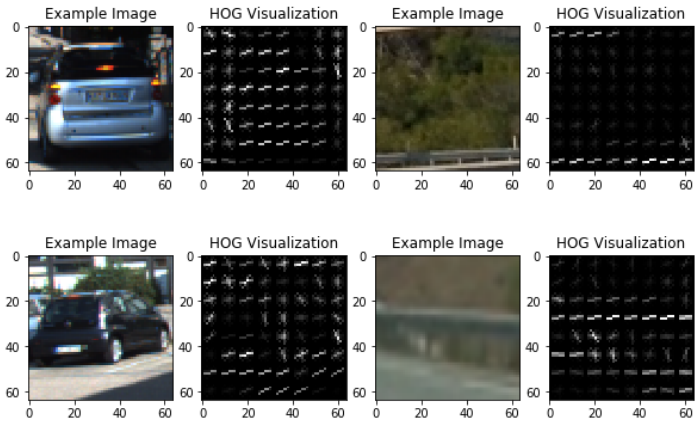


Figure 2: Visualization of HOG feature

In the above image, the input images is on the left, and the HOG visualization is on the right. The image on the right has some distinct darker colored lines which indicate a larger gradients. The image on the bottom right lacks any strong gradients. Different objects generate different histograms, and these features can be used to classify them.

In our case, each image is 64 by 64 pixels. With parameters ($orient = 9$, $pix\_per\_cell = 8$, $cell\_per\_block = 2$, $hog\_channel = "ALL"$), the dimension of the extracted hog feature becomes: $7 \times 7 \times 2 \times 2 \times 9 \times 3 = \mathbf{5292}$

2.2.2 Color and spatial binning features

The other set of features used to classify our images is a histogram of color and spatial distribution of pixels. Histogram of color finds a feature vector of size 32 for each color channels by calculating the number of pixels falling in the

certain color range. The final feature vector using parameters ($hist\_bins = 32$, $hist\_bins\_range = (0, 256)$) to construct histogram of color is of size : $3 \times 32 = \mathbf{96}$. For example the following images show a car and its corresponding histogram of color for each color channels.

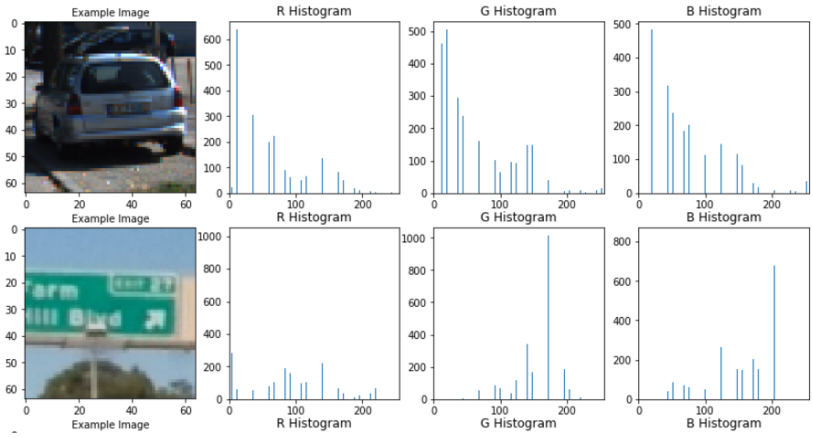


Figure 3: Visualization of color-histogram feature in RBG channels

Another useful feature is extracted from spatial distribution pixels, for this we reduce the size of image to 16 by 16 and extract its spatial pixel vector. The size of vector extracted is: $3 \times 16 \times 16 = \mathbf{768}$. Following are the sample of this feature.

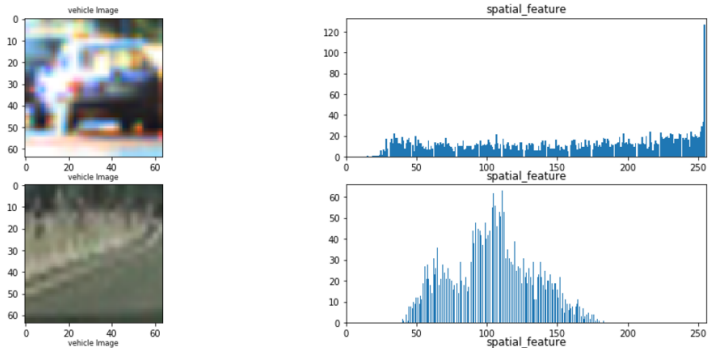


Figure 4: Visualization of spatial-distribution of pixels

Size of final feature vector: We stack all above feature to construct a single feature vector. The size of this final vector is given by

Total size = size of hog feature(5292)
+ size of color histogram(96)
+ size of spatial binning feature(768)

$\therefore$ Total size = 6156

2.3 Algorithm and techniques

This project uses many techniques from computer vision and machine learning to achieve its goals. Two methods (HOG feature extraction and Histogram of Color extractions) have been described in the previous section. These two sets of features are combined and used to train a linear support vector machine(SVM) and Neural Network classifier which is described in detail in coming sections.

2.3.1 SVM classifier - The linear model

A support vector machine plots the feature set in plane or hyperplane, and finds a decision boundary between the labeled features. A support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

Equation of Hyperplane: The equation of a straight line is $y = mx + c$, where m is the slope and c is the y-intercept of the line. Similarly, the generalized equation of a hyperplane is

$$\mathbf{w}^T\mathbf{x} = 0. \tag{2}$$

Where $\mathbf{w} = \big(-c, -m, 1\big)^T$ and $\mathbf{x} = \big(1, x, y\big)^T$ for 2 dimensions and are the vectors and $\mathbf{w}^T\mathbf{x}$ represents the dot product of the two vectors. The vector $\mathbf{w}$ is often called as the weight vector.

The training data in our classification problem is of the a pair of x_i, an n-dimensional feature vector and y_i, the label of x_i. Where $y_i = 1$ implies that the sample with the feature vector x_i belongs to class 1 and if $y_i = -1$, the sample belongs to class -1.For each such i, feature vector has 6156 elements for our specific problem. In a classification problem, we thus try to find out a function,

$$y = f(x) : \mathbb{R}^n \longrightarrow \{-1, 1\}. \tag{3}$$

$f(x)$ learns from the training data set and then applies its knowledge to classify the unseen data. There are an infinite number of functions, $f(x)$ that can exist, so we have to restrict the class of functions that we are dealing with. In the case of SVM, this class of functions is that of the hyperplane represented as $\mathbf{w}^T\mathbf{x} = 0$ or $\mathbf{w}.\mathbf{x} + b = 0; \mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ **Constraints**: In the case of higher dimension, $\mathbf{w}$ and data represents the vector which is normal to the hyperplane. Let $\mathcal{H}_0$ be a hyperplane separating the dataset and other two hyperplanes $\mathcal{H}_1$ and $\mathcal{H}_2$ such that they also separate the data and have the following equations: $\mathbf{w}.\mathbf{x} + b = \delta$ and $\mathbf{w}.\mathbf{x} + b =$ -δ.

This makes $\mathcal{H}_o$ equidistant from $\mathcal{H}_1$ as well as $\mathcal{H}_2$. Setting $\delta = 1$ it becomes

$$\begin{aligned} D(x) &= \mathbf{w}.\mathbf{x} + b = 1 \\ D(x) &= \mathbf{w}.\mathbf{x} + b = \text{-}1 \end{aligned} \tag{4}$$

To make sure that there is no point between them, we select only those hyperplanes which satisfy the constraints:

- $\mathbf{w}.\mathbf{x} + b \leq$ -1 for x_i having the class $y = -1$ or
- $\mathbf{w}.\mathbf{x} + b \geq 1$ for x_i having the class $y = 1$.

Combining both the above equations, we get

$$y_i \left(\mathbf{w}.\mathbf{x}_i + b\right) \geq 1 \text{ for all } 1 \leq i \leq n \tag{5}$$

Maximize the margin: A margin is a separation of line to the closest class points. A good margin is one where this separation is larger for both the classes. The margin, which is $m = \frac{2}{||\mathbf{w}||}$, includes a variable w. Hence

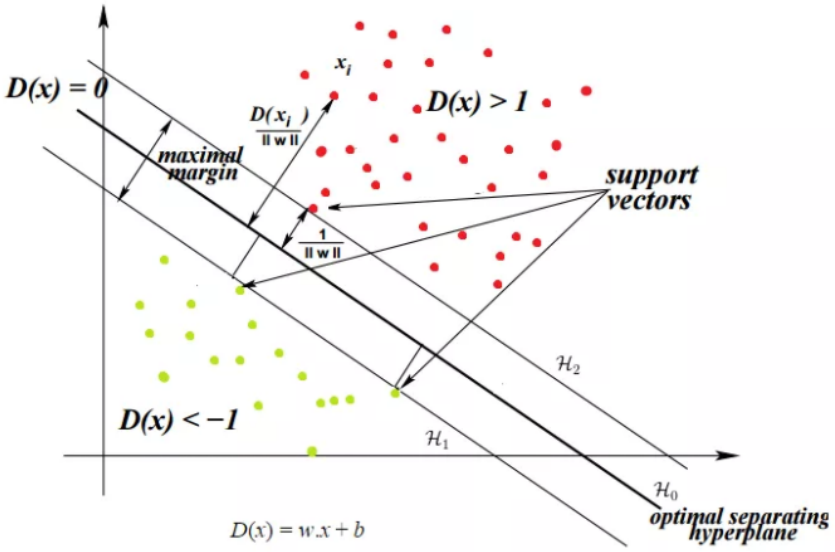


Figure 5: SVM hyperplanes: source

to maximize the margin, we will have to minimize $||\mathbf{w}||$. This leads to the following optimization problem:

$$\textit{Minimize in } (\boldsymbol{w}, b)\{\frac{||\boldsymbol{w}||^2}{2} \textit{ subject to } y_i\,(\boldsymbol{w}.\boldsymbol{x}_i + b) \geq 1 \textit{ for any } i = 1, \ldots, n\}$$

The above is the case when our data is linearly separable. There are many cases where the data can not be perfectly classified through linear separation. In such cases, Support Vector Machine looks for the hyperplane that maximizes the margin and minimizes the misclassifications. For this, we introduce the slack variable, ζ_i which allows some objects to fall off the margin but it penalizes them. In this scenario, the algorithm tries to maintain the slack variable to zero while maximizing the margin. However, it minimizes the sum of distances of the misclassification from the margin hyperplanes and not the number of misclassifications.

Constraints now changes to $y_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1 - \zeta_i$ for all $1 \leq i \leq n, \zeta_i \geq 0$ and the optimization problem changes to

Minimize in $(\vec{w}, b)\{\frac{||\boldsymbol{w}||^2}{2} + C\sum_i \zeta_i$ *subject to* $y_i\,(\boldsymbol{w}.\boldsymbol{x} + b) \geq 1 - \zeta_i$ *for any* $i = 1, \ldots, n\}$

Regularization: Here, the parameter C is the regularization parameter that controls the trade-off between the slack variable penalty (misclassifications) and width of the margin. Small C makes the constraints easy to ignore which leads to a large margin. Large C allows the constraints hard to be ignored which leads to a small margin. For $C = \infty$, all the constraints are enforced.

Concept of Kernel: In Case the feature sets to discriminate are not linearly separable in the feature space, original finite-dimensional feature space is mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $k(x, y)$ selected to suit the problem. Example of some kernel functions are:

1. *Polynomial* : $K(x_i, x) = (x.x)^d$
2. *Radial Basis Function* : $K(x_i, x) = exp\,(-\lambda||x_i - x||^2)\,; \lambda > 0$
3. *Gaussian Radial Basis Function* : $K(x_i, x) = exp\left(\frac{||x_i - x||^2}{2\sigma^2}\right)$

Gamma: The gamma parameter defines how far the influence of a single training example reaches, with low values meaning far and high values meaning close. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Whereas high gamma means the points close to plausible line are considered in calculation.

Scikit-learn provides us with various SVM implementations. For our project we used Linear Support Vector Classification, ***sklearn.svm.LinearSVC***. We chose LinearSVC over other implementations because it is a simpler model that scales better when using large datasets. Other SVM models, ***sklearn.svm.SVC*** with different kernels(*linear, poly, rbf, sigmoid, precomputed*) and tuning parameters(C and gamma) has been practised for

the search of best model. The result obtained has been discussed in Result section.

2.3.2 Neural Network

Next method used for building classifier model in the project is Neural Network. Consider a sample Neural Network as shown below. Let w^l_{jk} denotes the weight for the connection from the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer. Let b^l_j is the bias of the j^{th} neuron in the l^{th} layer. And let a^l_j as the activation of the j^{th} neuron in the l^{th} layer. With these notations, the activation a^l_j of the j^{th} neuron in the l^{th} layer is related to the activations in the $(l-1)^{th}$ layer by the equation

$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{(l-1)}_k + b^l_j\right), \tag{6}$$

where the sum is over all neurons k in the $(l-1)^{th}$ layer.

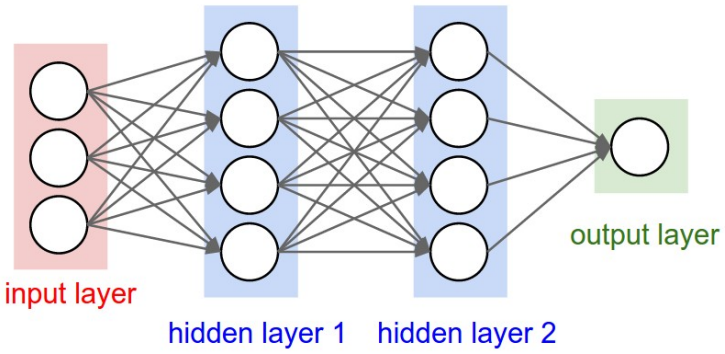


Figure 6: Sample network source

Cost function: The quadratic cost function is given by

$$C = \frac{1}{2n}\sum_x ||y(x) - a^L(x)||^2 \tag{7}$$

where n is the total number of training examples. Other cost functions can also be used. We are using categorical cross entropy as the loss function which best suits for categorical variable (car and not-car) we are dealing in this project.

Network Architecture: For construction of Neural Network Keras library has been used with tensorflow as backend. The model compile parameters are $loss = `categorical\_crossentropy'$, $optimizer = `rmsprop'$, $metrics = [`accuracy']$. Following are the network architecture specifications:

- Input layer: input vector size = 6156
- Hidden layer-1: layer size = 128, activation = 'relu', dropout = 0.15
- Hidden layer-2: layer size = 128, activation= 'relu', dropout = 0.15
- Hidden layer-3: layer size = 64, activation = 'relu',dropout = 0.15
- Hidden layer-4: layer size = 32, activation = 'softmax', dropout = 0.15
- Output layer: layer size = 1

The result obtained from this model are discussed in Result section.

2.3.3 Sliding Windows

Another technique used in this project is called sliding window search. This is useful in retrieving image segments from larger images. In our case we use this technique to identify cars in a dash-cam image. For example, we designate a window size and move the window throughout the image. This helps us retrieve the image encapsulated by the window which we can then feed to our classifier. The windows can have different sizes and overlap by a predetermined amount.

Figure 7 illustrates a the result of plotting the sliding window onto an image. The window size is actually four times larger than the visible boxes because

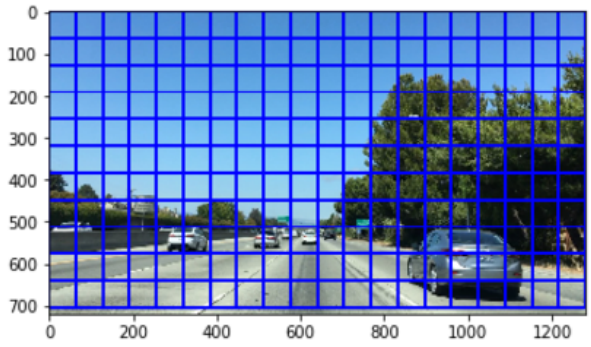


Figure 7: An example of sliding window

of the overlap between windows. Some overlap is useful to better identify and locate an object in an image. Not all part of the image are relevant to detect the vehicle in the videos, for this we have selected lower half of the image and designed three types of sliding windows with different starting, end points and scale as $ystart\_ystop\_scale = [(360, 560, 1.5), (400, 600, 1.8), (440, 700, 2.5)]$ which is shown in figure below.

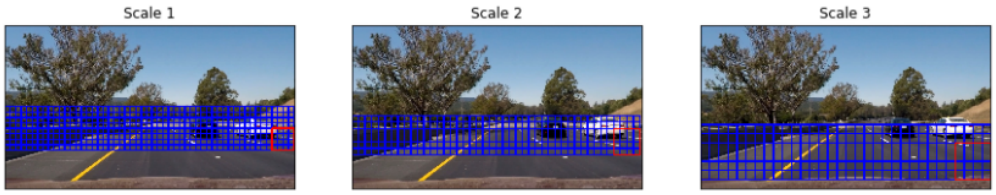


Figure 8: Different shape of slidding windows

2.4 Benchmark

This project will use report on Vehicle Detection in an Image and A Two-Stage Approach to People and Vehicle Detection With HOG-Based SVM as our bench mark model. These models use the concept of Histograms of Oriented Gradients HOG, for feature extraction purpose and some other geometrical aspects like edge detection etc. as well as *SVM* for classifier. Our aim in this project is also to extract color and HOG feature before we train the proper classifier.

3 Methodology

3.1 Data Processing

There is some preprocessing needed for our algorithm to work properly, specially regarding the images. We must first resize them to be the same size as the images used to train the classifier (in this case 64 x 64 pixels). We must also ensure that the images only have 3 channels and if the have a fourth or alpha channel it must be removed before feeding the image to the feature extraction function. For implementation of Neural Network data augmentation (doubling data by taking 180^o rotation) step has been used to increase the data size. The detail of data preprocessing can be found at Data Proessing notebook

3.2 Implementation

The whole methodology is divided into following steps:

1. **Feature Extraction** : This step provides a detail of feature extraction. Proper image size(32,32 3) has been selected. Various color channels like: 'BGR','HSV','LUV','HLS','YUV','YCrCb' has been be

studied for better feature extraction. HOG features and color features are the main part of the features. The detail can be found at Feature-extraction notebook

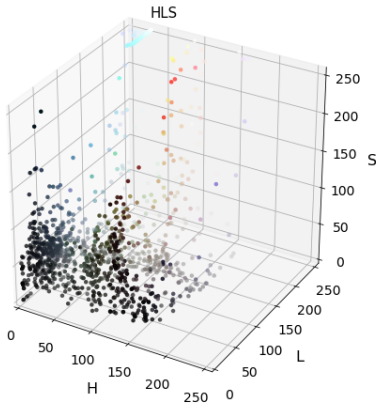


Figure 9: A sample od distribution of pixels on HLS color space

2. **Train a Classifier**: This step provides provide detail of training process, selection of parameters and theory of the model. Feature extracted from step 1 are normalized and split into train test part and feed into the training process. We have trained two models of classifiers: Linear SVM and Neural network classifier which are available at Parameter-tuning-svm notebook and Parameter-tuning-nn notebook

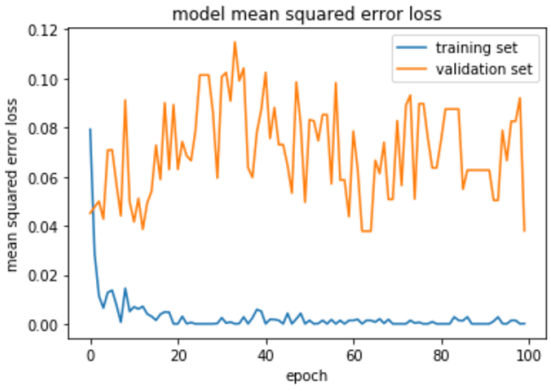


Figure 10: Training and validation accuracy of Neural Network

3. **Technique Sliding Window Search** : This step discusses about the utilities used for search of vehicle in the different parts of an images. An image is segmented into a smaller parts and some parts are given priority for searching vehicle. This results in a set of windows from which suspected part of the image is extracted and tested for presence of vehicle. The relevant functions are available at Sliding window search function

4. **Search and Classify an image** : This step uses all steps mentioned above, pretrained classifier model uses tiny section of the large image one-by-one through sliding window technique. If the classifier predict that tiny section as a car, a rectangle will be drawn around that section as shown below. The relevant functions for vehicle detection pipeline are available Vehicle detection pipeline functions

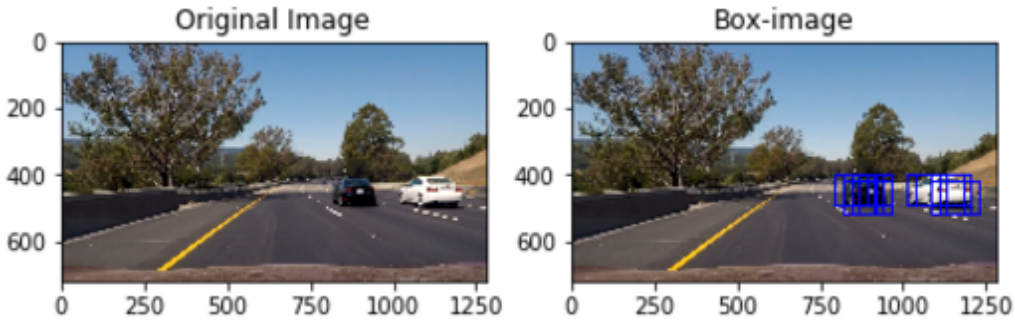


Figure 11: Vehicle found by sliding window search

5. **Search and Classify a Video**: This is the final part of the project which prepares the video along with vehicle inside a rectangular box. Since video is a time series of many images, we use the step 5 in all images in video and reprocess them to form final output video. To be more precise in the result, we used heatmap technique in vehicle detection to catch false positive cases as well. The relevant functions and result video are available at video-output-svm and video-output-nn respectively for svm and Neural Network model.

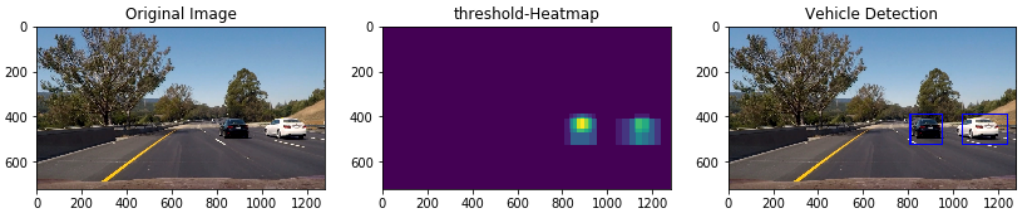


Figure 12: Vehicle detection pipeline with heatmap threshold

3.3 Refinements

Different models was tried with different version of feature sets. The most explicit refinement in this project is the addition of the Color features like color histogram and spatial binning to HOG features and the retraining of the classifier. The accuracy of the classifier on the testing data are the following:

- Test Accuracy of Linear SVC = 0.9904
- Test Accuracy of SVM with 'rbf' kernel = 0.8300
- Test Accuracy of Neural Network = 0.9918

4 Results

4.1 Model Evaluation and Validation

The final evaluation of the pipeline implemented in this project is done by viewing results by running dash-cam video through the algorithm. The resulting video is a copy of the original video but with bounding boxes drawn over the locations where cars have been identified by the classifier. In order to quantify the result we will use the following definition for accuracy.

$$\text{Accuracy} = \frac{(\text{true positives} + \text{true negatives})}{\text{size of dataset}} \tag{8}$$

In the above definition we will use the total number of window images to be classified to be the total population. True Positives are the window images classified as having cars (identified by a blue bounding box) that actually have cars. True Negatives are the window images identified as not having cars. We calculate the true negatives by taking the total number of windows minus the number of true positives and false positives. This process had to be done by hand, reviewing each frame of the video.

In most of the video frame true negative + true positive is equal to number of windows used in the image i.e. there is no false positive. The total accuracy of the classifier on the source video is 99.99%. The accuracy is higher than that of the classifier on the training data, which is quite remarkable. This might be because in the video, only features is a single car for most of its duration, and it is easily identifiable by our classifier.

The video result which I used for accuracy calculation is at test-sample-video. It has 40 frames and all information has been recorded in a file accuracy-measurement-file. This accuracy is not a global accuracy for other output videos included in the repository. I did not calculate this for other videos because of time consuming manual observation. Other output videos shows the trouble on classifying at lane barriers.

4.2 Justification

Using our definition of accuracy is a good metric for our classifier since it accounts for false positives which seem to be the biggest problem in our implementation. However the accuracy seems to be quite high because the classifier does a decent job of identifying non-car images as such, which is overwhelmingly the case for our source video.

5 Conclusion

5.1 Free-form Visualization

A single frame visualization of detected vehicle example:

Figure 13: Single frame example

In this single frame, we can see that the algorithm correctly identifies the two cars on the right lanes.

5.2 Reflection

Overal, the project is relatively successful. An accuracy of 99.9 % is quite high for our pipeline. This is not always true for long run. It would be quite unsafe to believe this accuracy just by evaluating few second video. This project does illustrate how machine learning techniques using proper data are able to identify objects with reasonable accuracy.

While most of the techniques used in this project are straightforward and have much documentation online, using them together was an interesting challenge. One challenge I got was on tuning parameters for SVM model with different kernels and 'C' parameters. The training process was really slow. I tried with subset of data to get the feeling of how slow it is. Rather I selected Linear SVC and Neural network which appeared outstanding finally.

After training the classifier, the next step was to extract image segments from a larger image, which is done with the sliding window technique. Identifying the proper window location was really tricky.

5.3 Improvements

While the results are reasonably successful, there is much room for improvement. While the dataset used to train the classifier is reasonably small, an even larger one could yield some improvement in accuracy.

Some images specific to lane barriers, which is where our algorithm seems to have the most trouble, should be handled carefully. Using more windows of different sizes could also be useful for identifying vehicles at the edge of the image.

Except this, some further parameter tuning could be good point to think again. The most important points to be considered in future research on this project are:

- Technique of deep learning to investigate hidden qualities and abstraction of mathematical property of image pixel distribution which are not clear just by open sight.
- Use of sensor suite as well as computer vision and performance of some sensor fusion to better identify objects are also equally important.

References

1. Object Detection Wikipedia
2. Human detection from images and videos: A survey
3. Machine Learning Technique for Face Analysis
4. Vehicle Detection in an Image
5. A Two-Stage Approach to People and VehicleDetection With HOG-Based SVM
6. Histograms of Oriented Gradients for Human Detection
7. character-recognition link
8. SVM scikit learn-link
9. SVM-review
10. Parameter tuning