NEW YORK UNIVERSITY

CUSP
CENTER FOR URBAN SCIENCE+PROGRESS

Vehicle Detection for Cities

Machine Learning for Cities

Project Report

Spring 2017

Akshay Penmatcha

Priyanshi Singh

Sunny Kulkarni

1. Introduction:

The task of detecting objects in our physical environment is an interesting one which has a lot of applications in the urban context. New York City's Department of Transportation publishes its traffic camera feeds as part of its Open Data Initiative. The potential applications of building a Vehicle detection model on these cameras could be multifold. It can help us in detecting and tracking congestions is real-time and in turn help in efficient traffic management and planning. Another potential application that is currently being researched at NYU Center for Urban Science and Progress is a Pedestrian tracking and counting using these camera feeds.

Researchers have taken different approaches to solve this for various types of physical objects using the concepts of Image Processing, Machine Learning and Deep Learning. (Dalal & Triggs, 2005) defined a feature descriptor called HOG and used an SVM to detect the presence of a Human in an image. (Viola & Jones, 2004) defined HAAR features motivated to devise a way for recognizing faces. Although it is also used widely for various other object detection tasks. In recent times the use of Convolutional Neural Networks has become widespread when the problem scope includes multiple classes of objects which are complex and when there is a need for reinforcement learning. Training a Neural Network is computationally challenging and to address this (Girshick, Donahue, Darrell, Berkeley, & Malik, 2012) devised a Region based CNN which drastically reduces the computational load because it performs the computations only on a specific region of interest. Later (Girshick, n.d.) and (Ren, He, Girshick, & Sun, 2015) provided enhancements to the R-CNN framework called Fast R-CNN and Faster R-CNN which further improved the computational performance of the CNN framework.

2. Approach:

The task of Vehicle Detection, which is primarily an Object detection problem using Computer Vision techniques could be solved using two approaches. The Shallow Machine Learning techniques and the Deep Learning techniques. The primary difference between the two approaches stems from the way they extract/learn the features in an image. The Shallow Learning methods use Feature Extraction which they input to a classifier to learn the presence of an object. Whereas the portion of Feature Extraction in a Deep Learning method is inbuilt with the first few layers. [6]

During this project we made an attempt to explore the deep learning approach, but due the complexities involved in designing a network and training through back propagation we narrowed down our scope to attempting the novel approaches within the realm of shallow learning techniques. Summarized below are the two different approaches that we have pursued.

I. Object Detection using HOG Features and SVM Classifier. [1]
II. Object Detection using HAAR Wavelets and Cascade Classifier. [5]

3. Datasets Explored:

The primary need for data collection in this project was to find a reliable set of positive training images which contain the object of interest and a set of negative images which contain anything other than the object of interest. Although this process is very similar for both the approaches that we pursued there is a difference in terms of the consistency in images sizes that we use for training. In case of HOG-SVM approach both the positive and negative images need to be in the same dimension. And in case of HAAR-Cascade approach the positive images need to be smaller than the negative images.

The following are the data sources that we explored for the purpose of training the algorithm.

1) ImageNet [7]
2) UIUC Image database for Car Detection [8]

4. Feature Extraction and Classification

A) HOG Features - SVM Classifier

An image is an array of data containing the information about the pixels. To detect the presence of an object with in an image a machine learning classifier needs to understand the localized features of the object rather than just the pixel intensities. Histogram of Oriented Gradients(HOG) is a feature extraction technique which is known to provide excellent performance relative to other feature descriptors [1] available out there.

Histogram of Oriented Gradients (HOG) Features: As (Dalal & Triggs, 2005) defined the idea behind

HOG is "*the local object appearance and shape can be characterized by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions." [1]*

Implementation of HOG: As (Dalal & Triggs, 2005) mentioned that implementation of HOG features is accomplished by calculating the aggregates of a 1-D histograms within a cell (group of pixels) and then to account for any illumination differences it within a normalized within a local block of cells. A visual representation of the HOG features is depicted in Fig-1 and Fig-2.

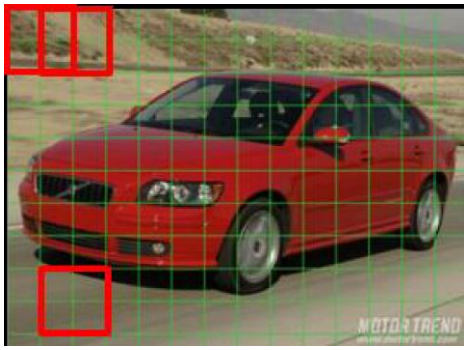


Fig-1: Representation of cells, blocks, stride [7]

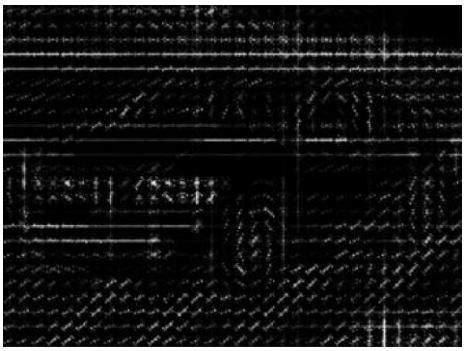


Fig-2: Visual representation of HOG Features

The implementation of HOG can be accomplished in python using scikit-image and OpenCV libraries. The implementation of the same has been demonstrated below.

In scikit-image,

```
fd, hog_image = hog(image, orientations=6, pixels_per_cell=(10, 10),
                    cells_per_block=(2, 2), visualise=True)
```

The primary inputs in this case are the cell size, block size and orientations. The output is the form of a tuple where ***fd*** is a flattened vector and ***hog_image*** is the visual representation of the hog features

In OpenCV,

```
hog = cv2.HOGDescriptor(winSize,blockSize,blockStride,cellSize,nbins,
                        derivAperture,winSigma,histogramNormType,
                        L2HysThreshold,gammaCorrection,nlevels)
```

The primary inputs are the same as scikit-image but having different representations. The output is this case is just a flattened vector.

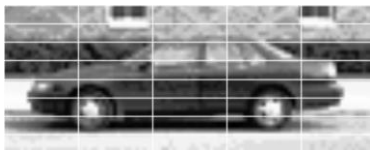
Dimensions of the flattened vector = No: of Blocks * No: of cell per block * No of Bins

Process Flow for the HOG-SVM Object Detection Approach:

This approach is adapted from (Dalal & Triggs, 2005), pyimagesearch[I] and various other blogs on the internet.

Step#1 - Labelled Dataset:

In this approach, we've taken a labelled dataset of 1000 images which containing 500 positive images which have a car and 500 negative images which do not have a car.



Positive: 500 [8]

Negative: 500 [8]

Fig-2: Positive and Negative Training Images

Step#2 - HOG Feature Extraction:

As mentioned previously, we later extracted the HOG feature vectors for these images and subsequently labelled them.

Step#3 - Training the SVM:

Next, we staged our labelled data and split them into a training and testing sets. Then we passed on the data to a Linear Support Vector Machine for training. Later we use our test data to get the confusion matrix. During this process, we found that by varying the parameter 'C' in the SVM input the accuracy of prediction changed. The inflection point for this change was found to be at C=0.1. Below are the results for the confusion matrix for C value 0.1 and 1.

Step#4 - Image Pyramids:

Image Pyramids are used to solve the problem of object scale. When we train our SVM and test on images of the same size, we do not encounter this problem. But when we try to test the Classifier on a larger image where multiple cars of different sizes are present the scale of the object differs throughout the image. In order to tackle this Image Pyramids provide a way to create a series of scaled down images through which we can parse our sliding window.

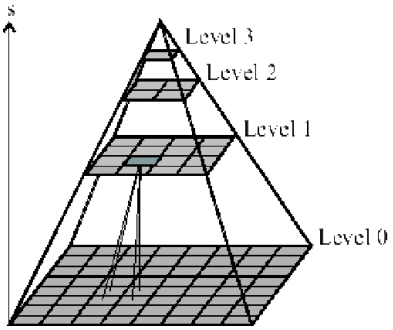


Fig-3: Image Pyramids [12]

Step#5 - Sliding Window:

The length of HOG feature vector that we get as an output is a function of our parameters which remains constant for an image of the same size with the same parameters. But since our real-world test images can be of different sizes we use a sliding window of the same size as our training images to capture the HOG features as it slides across the picture. We then capture all the coordinates of window's where the presence of a vehicle was detected.

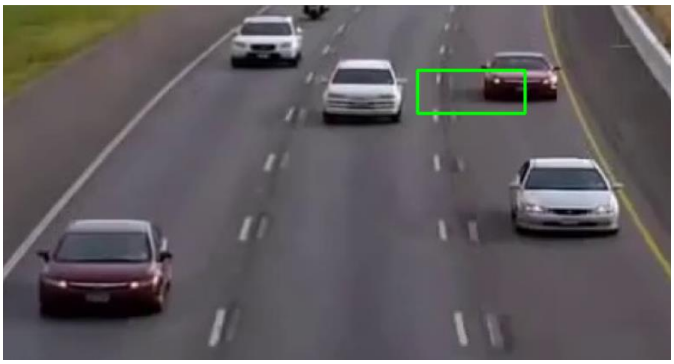


Fig-4: Sliding Window

Step#6 - Non-Maximum Suppression:

As the window slides across the image with a stride there is a possibility that the same object can get detected subsequently across multiple windows. To solve this, we use the Non-Maximum Suppression technique which allows us to filter out the redundant windows and keep only the unique ones detecting the vehicle in our case.

Results from HOG-SVM Approach:

Using a Train-Test split of 0.2 (20%) within our training set, the following are the results that we obtained for different C values in the Linear SVM Classifier.

Confusion Matrix	Predicted: NO	Predicted: YES
Actual: NO	99	1
Actual: YES	1	99

Fig-6: Confusion matix for C=0.1

Confusion Matrix	Predicted: NO	Predicted: YES
Actual: NO	97	2
Actual: YES	3	98

Fig-7: Confusion matix for C=1

Although the above results show a pretty high accuracy. A completely different test image like the one below has yielded very poor results. As we investigated the dataset for possible reasons, we found that the training set is highly skewed with vehicle images from their side view, unlike the image below.

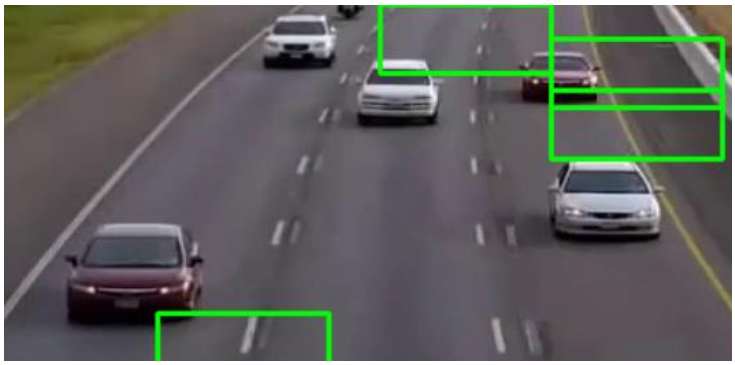


Fig-8: Predicted Cars in an unseen New Test

Hard Negative Mining: To solve the above problem, where our prediction accuracy goes down when we test our classifier on a completely different type image can be solved using the Hard-negative Mining technique. Where we record the sample of a few wrongly classified windows and pass them back to our SVM to retrain the model. Prediction accuracy is known to show a significant increase in the first run of hard-negative mining. Although this work is still in process the above problems were well addressed in the next approach that we have taken using the HAAR features and Cascade Classifier.

To see implementation refer to the below GitHub Repository –

https://github.com/akpen/ml_project

5. Conclusion and Way Forward:

The understanding of HOG and HAAR Image Processing techniques enabled us to understand how the objects are detected behind the libraries such as OpenCV and SKimage. The capabilities of SVM and Adaboost Classifier techniques are aligned with HOG and HAAR techniques thereby providing accurate results.

As seen in HAAR-Cascade Classification implementation, a single positive image is able to detect vehicles both in images and video feed with considerable accuracy.

As we have seen in the above implementation

Way Forward -

This project has given a good understanding of the underlying methodology applied towards object detection. Future scope of the project can be -

a. Count vehicles from traffic cameras to understand road congestion.
b. Classify vehicles into small, medium and large size such as for sedan and trucks.
c. HOG - SVM model can be further improved using hard mining
d. Apply Convolutional Neural Network techniques to improve accuracy and compare results with HOG - SVM and HAAR - Cascading Classifiers.

6. Bibliography/References:

[1] Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, *I*, 886–893. https://doi.org/10.1109/CVPR.2005.177

[2] Girshick, R. (n.d.). Girshick_Fast_R-CNN_ICCV_2015_paper, 1440–1448. https://doi.org/10.1109/iccv.2015.169

[3] Girshick, R., Donahue, J., Darrell, T., Berkeley, U. C., & Malik, J. (2012). Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper, 2–9. https://doi.org/10.1109/CVPR.2014.81

[4] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Nips*, 1–10. https://doi.org/10.1016/j.nima.2015.05.028

[5] Viola, P., & Jones, M. J. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision*, *57*(2), 137–154. https://doi.org/10.1023/B:VISI.0000013087.49260.fb

[6] Yann Le Cun

[7] ImageNet

[8] UIUC Image database for Car Detection

[9] Opencv tutorial for face detection
http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

[10]https://www.youtube.com/watch?v=WfdYYNamHZ8

[11] Creating your own Haar Cascade OpenCV Python Tutorial

Link: https://pythonprogramming.net/haar-cascade-object-detection-python-opencv-tutorial/