STUDY AND APPLICATION OF COMPUTER VISION ALGORITHMS/ TECHNIQUES TO DETECT OBJECTS & CONDUCT A NUMERICAL ASSESSMENT OF VEHICULAR TRAFFIC

Amit Sethi, MTech Sem II, JK Lakshmipat University

1. Introduction

1.1 Techniques and capabilities offered by Computer Vision (CV) have led to an epochal transformation in the manner in which machines can be made to interact and perceive the real world just like humans. Moreover, the advent of Neural Networks has further enhanced the ability of machines to aid human intelligence in capturing details of the various activities that take place in the world around us. This project intends to study and apply the algorithms offered by CV to detect objects and integrate this ability in a model, which especially, is able to detect/capture the details of vehicles and obtain a numerical assessment of vehicular traffic.

1.2 There are a large number of real-life applications for the detection, monitoring, and accounting of objects. Moreover, the detection and assessment of vehicular traffic in terms of can be extremely effective in applications such as traffic and access control; parking and theft prevention. Obtaining a precise count of vehicles can be of high importance for ensuring security of access like at the entry gates of gated institutions, residential colonies, hotels and sensitive locations. It may also be utilized or for ensuring that movement of vehicles is accounted in terms of numbers like at toll booths or parking plazas where vehicle count is tallied with tariff collections to prevent fraud.

2. Objective

2.1 The project intends to achieve the following objectives: -

a. To systematically study various CV techniques/algorithms for object detection in images and videos including study of Convolutional Neural Networks (CNNs) and Deep Neural Networks (DNNs).
b. To intuitively arrive at the best technique for detection and numerical assessment of vehicles in real-life scenarios.
c. Use the best assessed technique and create a functional Machine Learning Model for carrying out detection and numerical assessment of vehicles.
d. To create a web app to showcase the functionality achieved.

3. Statement of Problem

3.1 Numerical assessment of objects by the naked eye is tedious and inaccurate especially when this assessment involves Vehicles and Vehicular traffic. Computer Vision algorithms allow object detection by cameras in an efficient and automated manner. The project intends to use computer vision algorithms for numerical assessment of traffic as an alternate to human based manual counting and as a foundation to developing a portable easily deployable system to count varied type of objects and maintain an accountability of these objects. Moreover, object detection

also has a wide array of practical applications in the field of face recognition, surveillance, tracking objects, and crime detection.

4. Project Outline

4.1 The project outline is as follows: -

a. Study Open CV and libraries used for image detection.
b. Study the various techniques involved in image processing, classification, segmentation, object detection and feature description.
c. Study and analyses the image classifiers in Open CV and utilize the same apply so as to create a model for vehicle detection and counting.
d. Study and analyze the CNNs & DNNs for object detection.
e. Implement the best techniques to create a functional vehicle detection and numerical assessment model.
f. Implement the model in a Web Application so as to make it accessible on the Internet.

4. The Building Blocks: Open CV

4.1 Image Transformation. The python package, Open CV provides the foundational basis for Computer Vision. It reads images as either 2-dimensional or 3-dimensional matrices and provides the ability to carry out image Transformation, both affine and non-affine, image translations, edge detection as well as carrying out arithmetic operations on images. While Transformation refers to the process of bringing about geometric distortions such as scaling or rotation, Translation involves moving parts of an image. Most of these methods use a pre-defined matrix as a 'Kernel'. These techniques are broadly classified as Affine, wherein the straight-line geometry or parallelism of the image is maintained and non-affine, where such parallelism, angle and length are not maintained. Open CV also provides functions for thresholding, blurring, binarization and sharpening of images. Open CV also allows for Morphological operations on images which is primarily the process of manipulating images using filters. Filters, again are nothing but kernels which enable operations such as erosion, dilation, opening and closing. These are basic image processing techniques and form a small part in the entire process of object detection.

4.2. Segmentation & Contours. The genesis of object detection lies in segmentation and object detection. Image segmentation is a process by which images are partitioned into different regions. The regions correspond to different objects in the image. The concept relies on the fact that each object has a boundary or a Contour which can be identified. Contours can be sorted and matched hence enabling object matching and detection. This technique can be used for low level object detection, however has disadvantages and computational complexities when applied in real-life situations.

4.3 Blob Detection. Open CV also provides techniques for detecting a similar, connected pixels in a dense environment of pixels. This is called Blob detection. This again can be used for low level detection of repeated objects in similar images. However, all these algorithmic techniques are severely affected by change in camera angles, brightness, contrast and hue.

4.4 Feature Description Theory and Corner Detection. The concept of image 'Features' evolved in order to overcome the disadvantages posed in object detection by image distortions. 'Features' can be described as summary points in an image. They can also be described as 'interesting areas' in an image.

5. **Histogram of Oriented Gradients (HoG)**

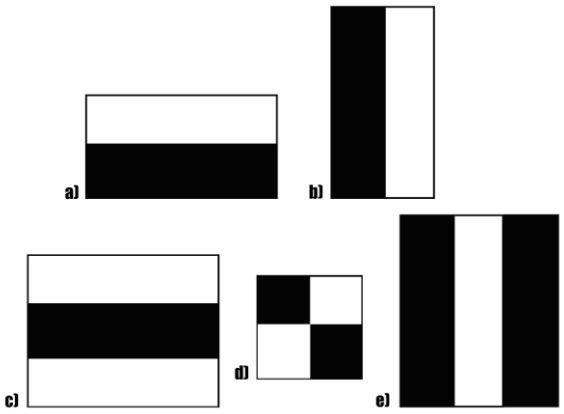
5.1 HoG in its very basic definition is a feature descriptor. A feature descriptor is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information. In HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) are used as features. Gradients (x and y derivatives) of an image are useful because the magnitude of gradients is large around edges and corners (regions of abrupt intensity changes) and it is known that edges and corners pack in lot more information about object shape than flat regions.

5.2 The HoG process involves pre-processing the image as patches which have a fixed dimension and aspect ratio. Calculation of the horizontal and vertical gradients followed by the division of the image into 8×8 cells and preparation of a histogram of gradients for each 8×8 cell.

6. **HAAR Cascade Classifiers**.

6.1 A Haar classifier, or a Haar cascade classifier, is a machine learning object detection program. More specifically it is a supervised classifier mainly used for facial detection, however, it can also be trained to detect other objects. The Haar Feature supported by OpenCV was initially proposed by Paul Viola and Michael Jones in the paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" [1]. This technique harbors on the concept of using **features** (also called Haar kernels) rather than pixels directly. These features on the image makes it easy to find the edges or lines in the image, or pick areas where there is a sudden change in the intensities of the pixels. This allows ease in processing and gaining domain knowledge through correlation of feature sets.

6.2 Fig 1 depicts the Haar features.



6.3 Fig2 is a representation of the use of Haar features for detection if object in an image. The rectangle on the left is a sample representation of an image with pixel values 0.0 to 1.0. The rectangle at the center is a haar kernel which has all the light pixels on the left and all the dark pixels on the right. The haar calculation is done by finding out the difference of the average of the pixel values at the darker region and the average of the pixel values at the lighter region. If the difference is close to 1, then there is an edge detected by the haar feature.

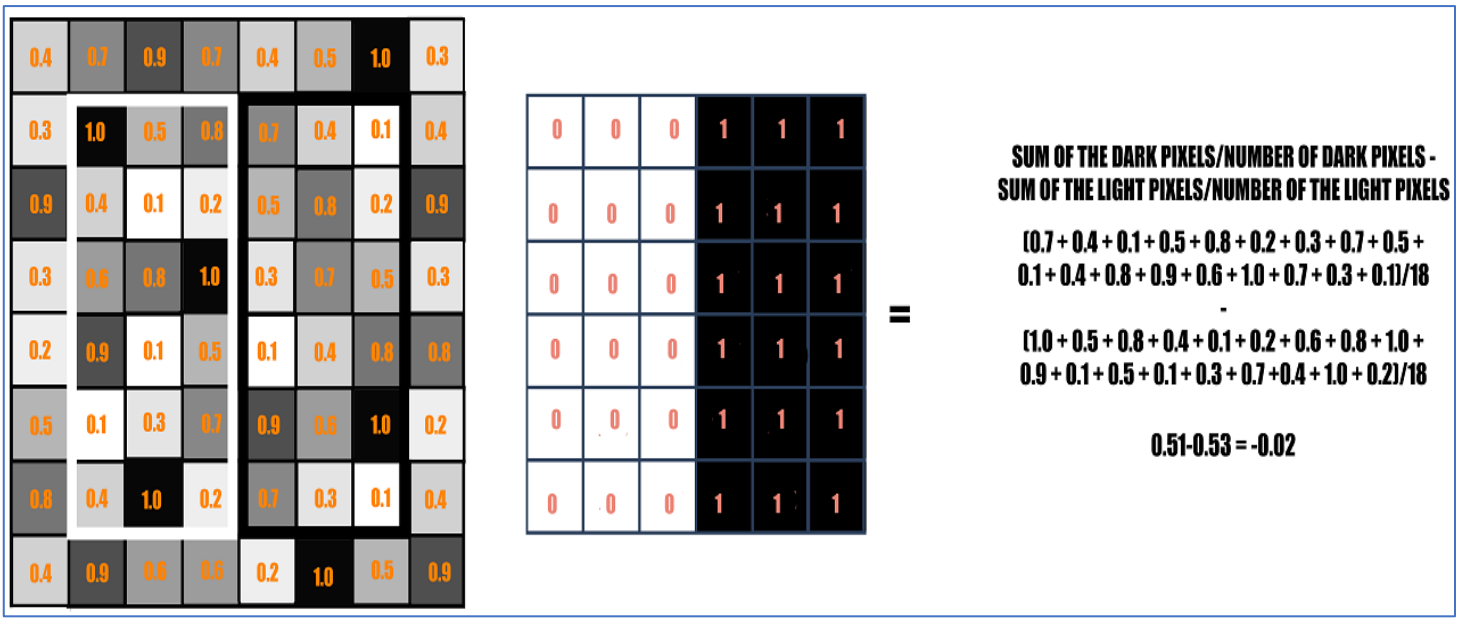


Fig2. Representation of the use of Haar Features

6.4 **Adaboost Training**. In order to determine the best features that represent an object from the hundreds of thousands of Haar features, Adaboost is used. Adaboost essentially chooses the best features and trains the classifiers to use them. It uses a combination of "weak classifiers" to create a "strong classifier" that the algorithm can use to detect objects. Weak learners are created by moving a window over the input image, and computing Haar features for each subsection of the image. This difference is compared to a learned threshold that separates non-objects from objects. Because these are "weak classifiers," a large number of Haar features is needed for accuracy to form a strong classifier.

6.5 **Cascading Classifiers**. The cascade classifier is made up of a series of stages, where each stage is a collection of weak learners. Weak learners are trained using boosting, which allows for a highly accurate classifier from the mean prediction of all weak learners. Based on this prediction, the classifier either decides to indicate an object was found (positive) or move on to the next region (negative). Stages are designed to reject negative samples as fast as possible, because a majority of the windows do not contain anything of interest.

6.6 **Implementation**. Car and Pedestrian Detection using HAAR cascade classifiers has been implemented in python and code is attached as **Annexure I.**

7. Convolutional Neural Networks (CNNs) & Deep Neural Networks (DNNs) for Object Detection

7.1 Object detection has been revolutionized by CNNs and more importantly by DNNs. Modern object detectors, utilizing the power of DNNs are both fast and much more accurate. Two approaches exist towards object detection using DNNs. The first approach is the Classification approach which selects the interesting region of the image then classifies it. The second approach is the Regression approach, which identifies the image in one run of the algorithm. The best examples of Classification Approach are 'Region based- CNNs' while 'Single Shot Multibox Detector' and 'You Look Only One' (YOLO) are the best examples of Regression Approach. For this project I have chosen to adopt the second approach since it offers fast functionality on a generalized network.

7.2 **YOLO** [2]

7.1 In 2016 Redmon, Divvala, Girschick and Farhadi revolutionized object detection with a paper titled: You Only Look Once: Unified, Real-Time Object Detection [3].

7.2 **Core Concept**. The idea behind YOLO is that there are no classification/detection modules that need to sync with each other and no recurring region proposal loops as were designed in R-CNNs based 2-stage detectors. It's basically convolutions with the occasional maxpool layer. Instead of cropping out areas with high probability for an object and feeding them to a network that finds boxes, a single monolithic network takes care of feature extraction, box regression and classification. While Classification models had two output layers — one for the class probability distribution and one for box predictions, YOLO has a single output layer containing everything in different features.

7.3 **YOLO V1 & V2**. Yolo-V1 was the first appearance of the 1-stage detector concept. The architecture employed batch normalization (BN) and leaky ReLU activations, that were relatively new at the time. In version Yolo-V2 the authors, among other changes, removed the fully-connected layer at the end. This enabled the architecture to be truly resolution-independent (i.e. — the network parameters can fit any input resolution). This doesn't necessarily mean that the network would perform well on any resolution. A resolution augmentation routine was employed for that during training. Redmon created multiple flavors of Yolo-V2, including smaller, faster (and less accurate) versions, like Tiny-Yolo-V2 etc. Tiny-Yolo-V2 has an extremely simple architecture since it doesn't have the strange bypass and rearrange operation that like its older sibling. The tiny version is just a nice, long chain of convolutions and maxpools.

7.4 **YOLO V3**. Inspired by ResNet and FPN (Feature-Pyramid Network) architectures, YOLO-V3 feature extractor, called Darknet-53 (it has 52 convolutions) contains skip connections (like ResNet) and 3 prediction heads (like FPN) — each processing the image at a different spatial compression.

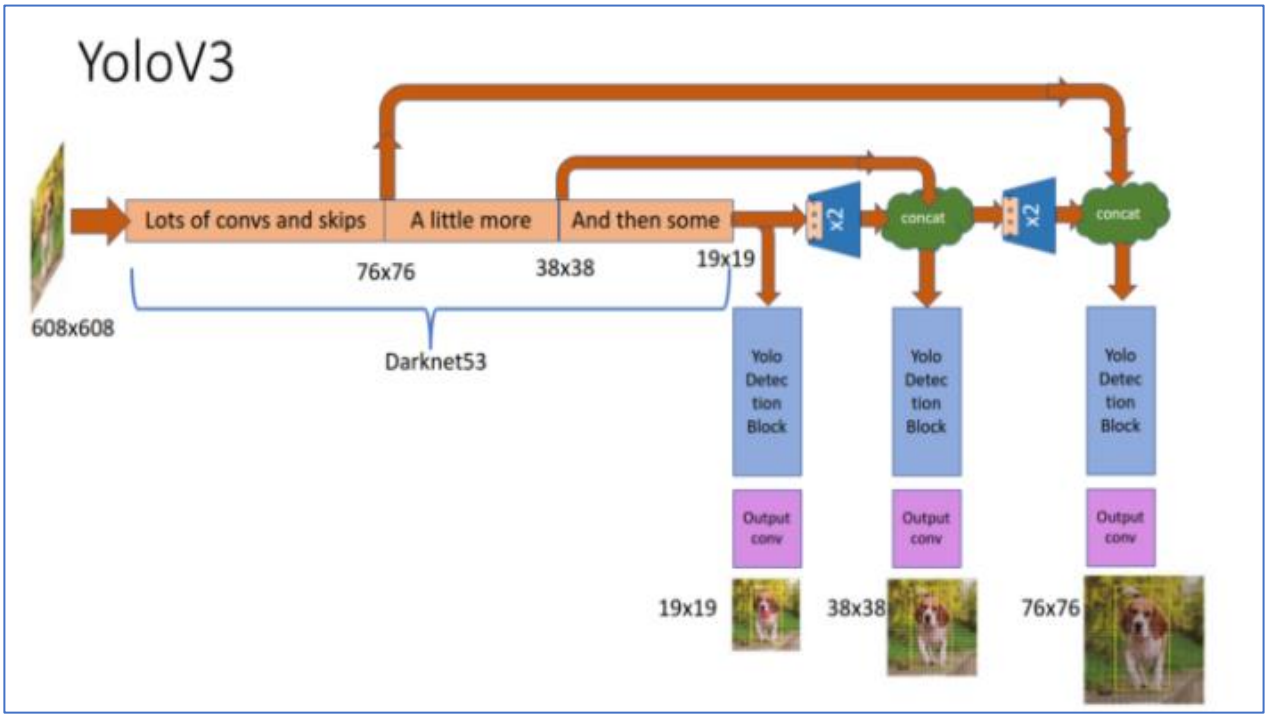


Fig 3. YOLO V3 Architecture

8. **Results and Conclusion**.

8.1 YOLO performs far better than Haar Cascades. Yolo-V3 boasts good performance over a wide range of input resolutions. Tested with input resolution 608x608 on COCO-2017 validation set, Yolo-V3 scored 37 mAP (mean Average Precision).

8.2 The code for implementation is attached as **Annexure II**.

References

[1] M. J. Paul Viola, "Rapid Object Detection using a Boosted Cascade of Simple," *ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001,* 2001.

[2] U. Almog, "YOLO V3 Explained," Medium.com, Oct 2020. [Online].

[3] D. G. F. Redmon, "You Only Look Once: Unified, Real-Time Object Detection.," *arXiv.org,* 2016.