Comparison of Support Vector Machines and Deep Learning for Vehicle Detection

Özgür Kaplan and Ediz Şaykol
Beykent University Department of Computer Engineering,
Ayazağa Campus, 34396, Istanbul, Turkey
ozgurkaplan@outlook.com , ediz.saykol@beykent.edu.tr

Abstract

The main goal of this paper is to compare different Vehicle Detection algorithms and to provide an effective comparison technique for developers and researchers. During this study, fine tunings are suggested to improve the implementations of these algorithms. Our focus on Support Vector Machines (SVM) and then Deep Learning based approaches. The SVM based vehicle detection implementation utilizes Histogram Oriented Gradients (HOG). The deep learning approach we consider is the YOLO implementation. Our evaluation employs 400 random frames extracted from a real world driving video. As stated by the experimental results, YOLO is more accurate with %81.9 success than SVM which only scored %57.8.

1. Introduction

According to the World Health Organization (WHO), 1.25 million people die each year from traffic accidents [WHO15]. Prevention of traffic accidents and punishment of drivers who violate the rules are of great importance for humanity. Many municipalities are using automated camera system to detect cars that violates traffic rules. Proper operation of these systems is crucial to avoid further traffic accidents.

It is estimated that 1 in every 10 cars in traffic will be composed of vehicles capable of self-driving in 2030 and 1 in every 3 in 2050 [MKG+16]. It is obvious that the safety of future traffic will be directly linked to the quality of the software these vehicles will be using. For both software, detecting vehicles who violate the rules and self-driving vehicles, it is critical to identify images correctly.

There are several techniques in the literature suited for vehicle detection task. Existing papers suggests using acoustic sensor networks [WCH17], wavelet and interest point based feature extraction [KB13], edge-based constraint filters based vehicle segmentation [Sri02] and time spatial data [MRR10].

Besides, there are also simulation-based approaches to help drivers. For example, [PYL+15] proposed an increased reality system to increase the driver's immediate attention. The system calculates the probability of collision with the vehicles in the same lane as the driver and colorizes the lanes according to the risk ratio. In [RSL+14], it is aimed to help middle-aged and older drivers to make a left turn in running traffic. Drivers have been given hints to return with augmented reality for their left turns in different scenarios such as heavy traffic and flowing traffic.

In this paper, we focus on SVM-based vehicle detection technique, where we can simplify vehicle detection problem to a classification problem by examining individual sections of image and classify whether it is a vehicle image or not a vehicle. On the other side, we consider Deep Learning approaches that automatically learn image features that are required for vehicle detection. There are different deep learning techniques such as Region-based Convolutional Network (R-CNN) [GDD+14], Fast R-CNN [Gir15], Faster R-CNN [RHG+17] and You Only Look Once (YOLO) [RDG+16]. We choose YOLO as it is the faster among these techniques [DU18]. The basic aim is to provide an effective comparison technique for developers and researchers.

The organization of the paper is as follows: In Section 2, we present vehicle detection techniques that we employ in our study. Section 3 provides the comparative study and Section 4 concludes the paper.

2. Vehicle Detection Approaches

Our study contains two major components. First phase is detecting vehicles using SVM and deep learning with YOLO. Second phase is comparing accuracy and performance of these two methods.

As our starting point we utilized implementation of [Fu17] that includes a classical SVM approach using OpenCV and Histogram of Oriented Gradients (HOG) feature extraction as well as a deep neural network with YOLO tensor flow implementation using pre-trained weight object [Cho16]. To simplify the problem we only considered cars as vehicles.

2.1 SVM-based Vehicle Detection using HOG

In this approach, we used supervised learning with pre-categorized images. We used the images provided by GTI vehicle database [GTI18]. There are 3425 images of vehicle rears and 3900 images of road sequences not containing vehicles. All images are 64x64 pixels. Figure 1 shows a vehicle image and a non-vehicle image.

Figure 1: A vehicle image (left) and a non-vehicle image (right) in [GTI18].

First step to prepare the images for SVM classifier is to extract HOG features. The main purpose of HOG is to identify the image as a group of local histograms.

HOG is not scale invariant. In order to use HOG we needed the same size for our training images. All our training set images were already the same size of 64x64 so we were able to use our images directly. Figure 2 shows our HOG implementation details.

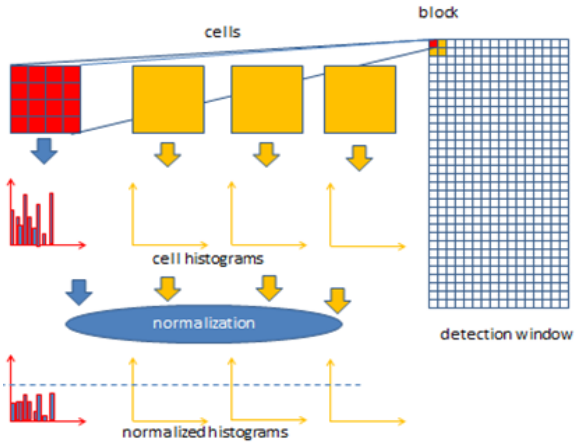

Figure 2: HOG Implementation. (The image source: https://software.intel.com/en-us/ipp-dev-reference-histogram-of-oriented-gradients-hog-descriptor)

The HOG technique counts the occurrences of the gradient orientation of the localized regions of an image. First image is divided into small connected cells, and for each cell gradient directions are calculated. Each cell is splitted into angular bins according to the gradient orientation. The pixel of each cell adds the weight gradient to the corresponding angular bin. Groups of adjacent cells are called blocks. The grouping of cells to blocks is the basis of normalization of histograms. This normalized group of histograms generates the block histogram and set of block histograms represents the descriptor [Int18]. Figure 3 shows a car image and the corresponding HOG transformation.

Figure 3: HOG Transformation of a car image.

To train the model we randomly split the data into two parts with respect to 5-fold. %80 percent is used for training and %20 percent is used for test, as suggested by most of the learning schemes.

We explored different parameters and color spaces for our images to get better results. After some trial and error we managed to get %86.7 success rate.

We wanted to test the classifier on a large scale images. We found raw driving video data on KITTI Vision Benchmark Suite [KIT18]. We modified the code to make it work on 1392 x 512 pixels which is the resolution of raw driving video data.

The 64x64 pixel block of search windows is used to search the entire frame. We used 50% percent overlap for search windows. For each window, our SVM classifier is used to detect if there is a vehicle or not. Figure 4 shows how search windows are used.

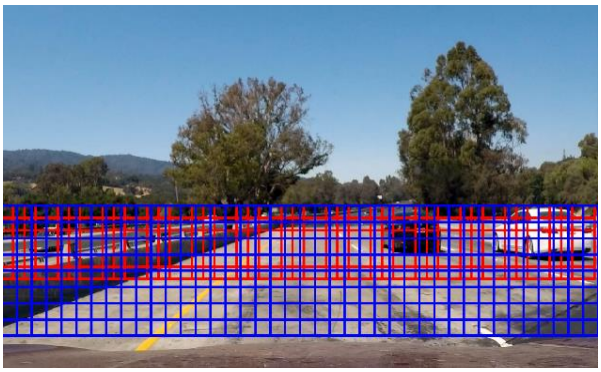

Figure 4: Search Windows (The image source: https://raw.githubusercontent.com/JunshengFu/vehicle-detection/master/examples/search_windows.png)

To get the exact locations of vehicles, heat map generation is used with the windows that may have vehicle. Figure 5 shows a successfully detected vehicle using SVM.

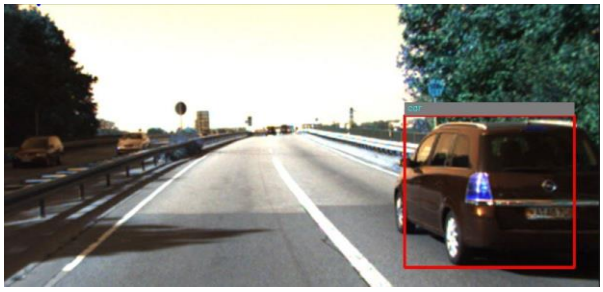

Figure 5: Vehicle Detected with SVM

2.2 Deep Learning based Vehicle Detection

We chose the YOLO technique for this purpose. YOLO uses deep neural network to detect objects. Yolo approaches object detection as a simple single regression problem to find bounding boxes class probabilities. YOLO is trained on full images and it predicts multiple bounding boxes using single neural network. First, YOLO model accepts an image as input then divides it into an SxS grid. Each cell of this grid estimates the B bounding boxes and C class probabilities. The bounding box has 5 components: x, y, w, h and confidence. The (x, y) are center coordinates of the box. The (w, h) are dimensions. The confidence score tells us if there is any shape in the box. If the score is zero then there should be no object in the cell. Each grid cell makes B predictions, so there are total of S x S x B x 5 outputs. The network predicts one class probability per cell that results S x S x C total probabilities. Figure 6 shows YOLO object detection and Figure 7 shows YOLO execution pipeline.

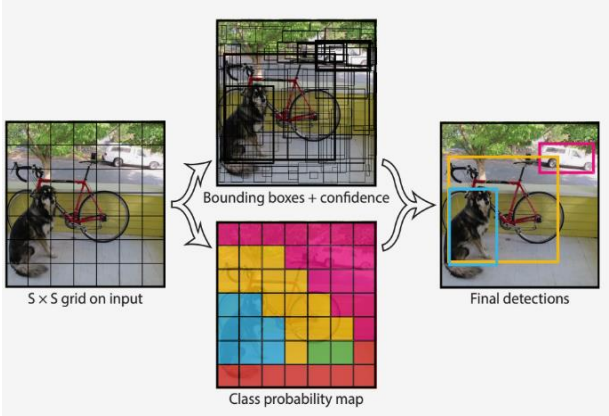

Figure 6: YOLO Object Detection [RDG+16]

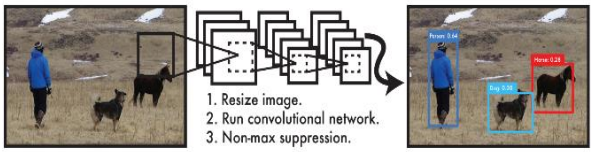

Figure 7: YOLO Pipeline [RDG+16]

We utilized [Fu17] code, which employs tensorflow implementation of YOLO [Cho16]. It uses pre-trained YOLO_small network, which has 20 classes as follows: "aeroplane", "bicycle", "bird", "boat", "bottle", "bus", "car", "cat", "chair", "cow", "dining table", "dog", "horse", "motorbike", "person", "potted plant", "sheep", "sofa", "train","tv monitor". Since vehicle is already known as “car”, we were able to use precomputed settings and apply it directly to our inputs. We used 30% threshold, cells whose car class score is 0.3 or more are selected. Figure 8 shows a successfully detected vehicle using YOLO.

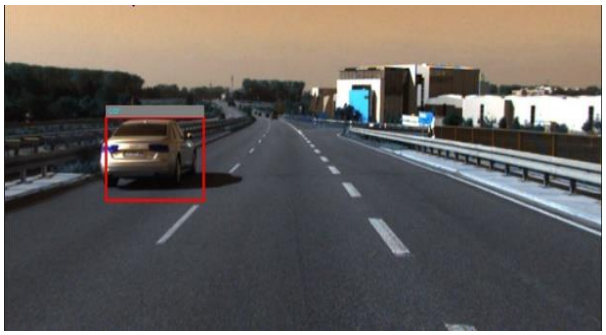

Figure 8: Vehicle Detected with YOLO

3. Comparing SVM and YOLO

We tested both algorithms on the same randomly selected raw driving video data. Random data contains 400 images with total of 266 vehicles in them.

Test results are evaluated in 3 categories;

- Positive : Vehicle detected correctly
- Negative: Vehicle is not detected.
- False Positive: Non-vehicle is detected as vehicle.

Figure 9 shows a sample snapshot of our experimental study employing SVM-based technique. SVM generated a lot of false positives. In some frames, SVM identified road signs, trees and pedestrians as vehicles. These false negatives may be avoided if a pre road line detection is performed. Thus, we can limit SVM search window within actual road.

Figure 9: Sample Snapshot of SVM-based technique.

Figure 10 shows a sample snapshot of our experimental study employing deep learning based YOLO technique. YOLO downscales images to 448x448 which causes distortions. With pre road or line detection, we can send smaller and more useful segments to YOLO to avoid distortions and get better results.

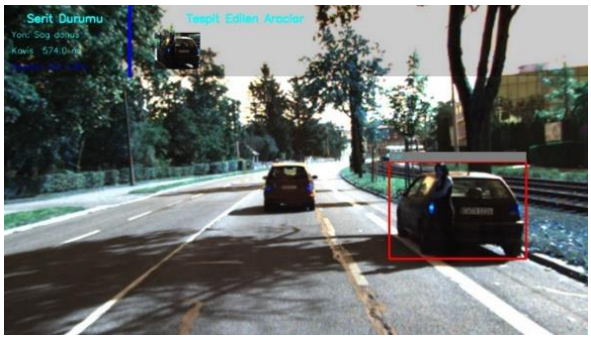

Figure 10: Sample Snapshot of YOLO technique.

Both algorithms failed to detect vehicles, which are further ahead. We may be able to detect these further vehicles with SVM using higher resolution data. However higher resolution data would require more computation and therefore the performance would be worse. As for YOLO, which downscales pictures to 448x448 pixels, it is useless to use higher resolution data.

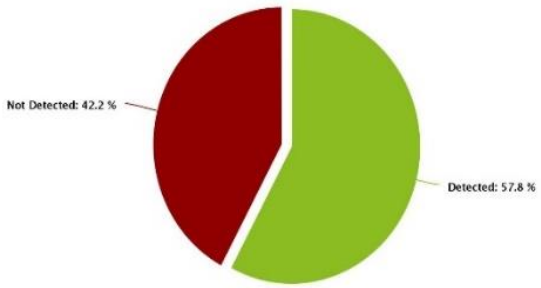

Figure 11: Test results for SVM

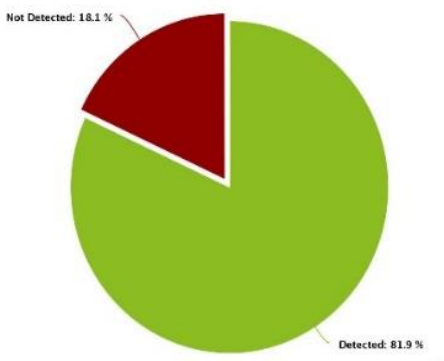

Figure 12: Test results for YOLO

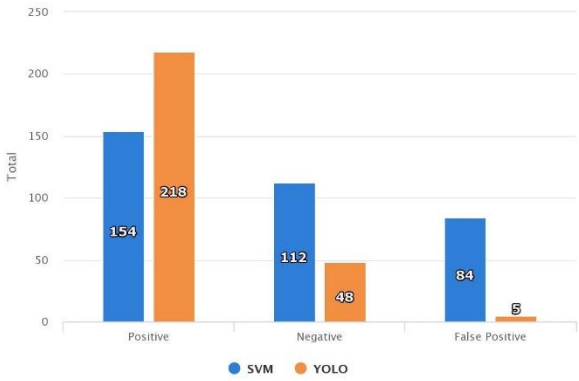

Figure 13: Overall Test Results.

Our test results shows that YOLO's success rate is 81.9%, 218 out of 266, and SVM's is %57.8, 154 out of 266. It also has significantly lower false positives with 5 against SVM's 84. When we apply false negatives YOLO's success rate drops to 80.4% and SVM's drops to 44%.

One reason of high false positives of SVM is that unlike YOLO, who can classify 20 different types, SVM can only identify vehicles. YOLO lowers the class probability if there is another possible object in the frame, which lowers the chance of false positives.

We processed the entire video using Asus Nvdia Geforce 1060 gpu. YOLO performed up to 42 fps while SVM reach just 4 fps. This is because YOLO is lightweight and use single neural network on given frame. On the other hand, SVM needs recursive calculations with multiple sliding window calculations.

4. Conclusion and Future Work

In this paper, we looked into two different vehicle detection algorithm implementations and test them against real world traffic data.

Our results showed that deep neural network with YOLO performed more accurate results. YOLO also performed faster which makes it more suited for real time tasks.

With our approach we are able to find strong and weak sides of both models. Thus, we managed to implement and suggest fine tunings. Our approach has potential to be used to compare different algorithms and to be used to find fine tunings.

References

[WHO15] World Health Organization. (2015). Global status report on road safety 2015. World Health Organization. http://www.who.int/iris/handle/10665/189242

[MKG+16] D.Mohr, H.Kaas, P.Gao, D.Wee, and T.Möller, "Automotive revolution—Perspective towards 2030: How the convergence of disruptive technology-driven trends could transform the auto industry," McKinsey & Company, Washington, DC, USA, Tech. Rep., Jan. 2016

[WCH17]Wang, R., Cao, W., & He, Z. (2017). Vehicle recognition in acoustic sensor networks using multiple kernel sparse representation over learned dictionaries. International Journal of Distributed Sensor Networks. https://doi.org/10.1177/1550147717701435

[KB13] KumarMishra, Pradeep & Banerjee, Biplab. (2013). Multiple Kernel based KNN Classifiers for Vehicle Classification. International Journal of Computer Applications. 71. 1-7. 10.5120/12359-8673.

[Sri02] Srinivasa, Narayan. (2002). Vision-based vehicle detection and tracking method for forward collision warning in automobiles. 10.1109/IVS.2002.1188021.

[MRR10] N. C. Mithun, N. U. Rashid and S. M. M. Rahman, "Detection and Classification of Vehicles From Video Using Multiple Time-Spatial Images," in IEEE Transactions on Intelligent Transportation Systems, vol. 13,

no. 3, pp. 1215-1225, Sept. 2012. doi: 10.1109/TITS.2012.2186128

[PYL+15] B. J. Park, C. Yoon, J. W. Lee and K. H. Kim, "Augmented reality based on driving situation awareness in vehicle," 2015 17th International Conference on Advanced Communication Technology (ICACT), Seoul, 2015, pp. 593-595. doi: 10.1109/ICACT.2015.7224865

[RSL+14] Rusch, Michelle & Schall, Mark & Lee, John & Dawson, Jeffrey & Rizzo, Matthew. (2014). Augmented reality cues to assist older drivers with gap estimation for left-turns. Accident Analysis & Prevention. 71. 210–221. 10.1016/j.aap.2014.05.020.

[GDD+14] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 580-587.doi: 10.1109/CVPR.2014.81

[Gir15] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1440-1448. doi: 10.1109/ICCV.2015.169

[RHG+17] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017. doi: 10.1109/TPAMI.2016.2577031

[RDG+16] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788

[Du18] Juan Du (2018). Understanding of Object Detection Based on CNN Family and YOLO. Journal of Physics: Conference Series, 1004, 012029.

[Fu17] Juhnsheng FU. (2017, March 17). Vehicle Detection for Autonomous Driving. Retrieved from https://github.com/JunshengFu/vehicle-detection

[Cho16] Jinyoung Choi. (2016). Tensorflow Implementation of 'YOLO: Real-Time Object Detection'. Retrieved from https://github.com/gliese581gg/YOLO_tensorflow

[GTI18] Vehicle Image Database (2018, May 12) http://www.gti.ssr.upm.es/data/Vehicle_database.html

[Int18] Histogram of Oriented Gradients (HOG) Descriptor (2018, April 2) https://software.intel.com/en-us/ipp-dev-reference-histogram-of-oriented-gradients-hog-descriptor

[KIT18] KITTI Vision Bench. Suite (2018, May 13) http://www.cvlibs.net/datasets/kitti/raw_data.php