Real-time Pedestrian Traffic Light Detection

Roni Ash[+], Dolev Ofri[+], Jonathan Brokman, Idan Friedman and Yair Moshe
Signal and Image Processing Laboratory (SIPL)
Andrew and Erna Viterbi Faculty of Electrical Engineering, Technion – Israel Institute of Technology
Technion City, Haifa 32000, Israel
http://sipl.technion.ac.il/

***Abstract*—Crossing a road is a dangerous activity for pedestrians and therefore pedestrian crossings and intersections often include pedestrian-directed traffic lights. These traffic lights may be accompanied by audio signals to aid the visually impaired. In many cases, when such an audio signal is not available, a visually impaired pedestrian cannot cross the road without help. In this paper, we propose a technique that may help visually impaired people by detecting pedestrian traffic lights and their state (walk/don't walk) from video taken with a mobile phone camera. The proposed technique consists of two main modules - an object detector that uses a deep convolutional network, and a decision module. We investigate two variants for object detection (Faster R-CNN combined with a KCF tracker, or Tiny YOLOv2) and compare them. For better robustness, we exploit the fact that abrupt switching from red to green or vice versa is unique to traffic lights. The proposed technique aims to operate on a mobile phone in a client-server architecture. It proves to be fast and accurate with running time of 6 ms per frame on a desktop computer with GeForce GTX 1080 GPU and detection accuracy of more than 99%.**

Keywords— Pedestrian traffic light detection, Object detection, Faster R-CNN, YOLO object detector, KCF tracker

I. INTRODUCTION

About 217 million people worldwide have moderate to severe vision impairment and about 36 million are blind [1]. Most visually impaired and blind people experience severe difficulties in travelling even short distances in public spaces [2]. They practice travelling on specific routes that they use often, such as the route from home to a supermarket. However, crossing a road while walking in a route may prevent the independent mobility of a person with visual impairment. Crossing a road or an intersection is a dangerous activity for pedestrians and therefore crosswalks and intersections often include pedestrian-directed traffic lights. To make it easier and safer for visually impaired pedestrians to cross roads, walk lights can be accompanied by audio signals to indicate when it is safe to cross. Unfortunately, these accessible pedestrian signals (APS) are not available in many pedestrian crossings worldwide. Thus, making the visually impaired dependent on the assistance of a human guide. There is a great advantage for a user-side automatic guidance without the need for any additional infrastructure installed at each crosswalk.

User-side assistive technology to support independent mobility of visually impaired people has a long history. However, most existing solutions have not gone beyond the prototype stage and others are used by relatively small number of visually impaired people. These solutions usually include additional aiding equipment, making them less available to everyone [2]. As mobile phones with an onboard camera are ubiquitous these days, a solution that does not require any special hardware and is based on a mobile phone in a computer vision approach has great potential. Such an approach exploits the fact that traffic lights have easily noticeable color-coded illuminated signs.

Detection of pedestrian traffic lights using a mobile phone camera faces many challenges [3]:

- Pedestrian traffic lights have different appearances in different countries and even for different manufactures.
- The distance to a pedestrian traffic light may vary (typically 4 to 20 meters) and therefore the scale of a traffic light may vary.
- Traffic lights can be temporarily occluded by vehicles and other pedestrians.
- There may be several traffic lights in the scene but only one is relevant.
- Illumination varies depending on the time of the day and the weather.
- Detection should be robust to different image qualities and resolutions.
- A video stream captured on a mobile phone is usually not stabilized.
- Computation power and memory resources are restricted.

With all these challenges in mind, note that in this application accurate detection is crucial, as false positive detection of green light wrongly indicates that it is safe to cross the road and may be fatal to human life.

Traffic light detection is not only beneficial for pedestrians but is also an important task for driver assistance systems. With the increasing interest in autonomous driving, many recent papers deal with vehicle traffic lights detection. To detect traffic lights, these papers usually suggest to extract from each image meaningful features according to color or shape properties, and then to classify according to these features [4]. Different color spaces are used in the literature to cluster by color. Color properties are often exploited by using a set of sequential rules. Other rules are based on the shape, aspect ratio, texture and size of traffic lights to be detected. More advanced approaches combine a rule-based approach with a classifier such as SVM, HMM or AdaBoost, to detect the structure of a traffic light. Temporal features are usually not explicitly considered due to computational and memory constraints. Using a priori knowledge of traffic light position may also help.

Works concentrating on pedestrian traffic light detection usually follow the same path as works on vehicle traffic light detection, exploiting the unique shape and color of a traffic light in a classical image processing approach. In [3, 5], a prototype system for detecting pedestrian traffic lights was implemented on a Nokia N95 mobile phone. Color parameters are utilized to determine candidate regions, and the parameters of aspect ratio and possible corresponding size are utilized to filter out false positives. To improve robustness, candidate objects are tracked in consecutive video frames. Another

[+] These two authors contributed equally to this work.

978-1-5386-6378-3/18/$31.00 ©2018 IEEE

algorithm for detecting pedestrian traffic lights on the Nokia N95 is presented in [6, 7]. In the first stage, the algorithm uses smartphone sensors to determine the position of the smartphone with respect to the horizon and it analyzes only the upper part of the image. Then, it detects the circular light and the shape of the pedestrian inside the circle. This algorithm also searches for a crosswalk to validate the result. In [8, 9], besides object detection and recognition, a robust setup for image capture is proposed. This setup allows to acquire clearly visible traffic light images regardless of daylight variability due to time and weather. Two recent papers use machine learning to detect pedestrian traffic lights. In [10], a support-vector machine (SVM) classifier is used to classify histogram-of-gradients (HOG) features. In [11], a background filter is applied to identify candidate regions of pedestrian traffic lights and an AdaBoost classifier is used to detect these traffic lights. The algorithm in both [10] and [11] runs on a strong desktop computer to achieve real-time performance.

In spite of the large number of works on vehicle and pedestrian traffic light detection, these tasks are still considered active problems for industries and research groups [4]. In recent years, machine learning techniques using convolutional neural networks (CNNs) have enabled rapid and accurate object detection and recognition for use in various applications. These techniques have a great potential in improving substantially the performance of pedestrian traffic light detection systems. The goal of our work is to exploit state-of-the-art CNN-based object detection to help people with visual impairment safety and independently cross roads with pedestrian traffic lights but without APS. A user will roughly aim his mobile phone at the traffic light and a dedicated future application will acquire video and stream it to a cloud server. The proposed technique, running on the server, will analyze the mobile video stream in real-time and will detect whether a relevant pedestrian traffic light is green ('walk light') or red ('don't walk light'). The application will generate an audio signal accordingly.

In this paper, we present a novel approach for pedestrian traffic light detection using deep neural networks. The approach aims as a smartphone application operating in real-time in a client-server architecture. The paper is organized as follows. In Section II we present the dataset we have built for training and testing. Section III presents two variants of our proposed solution and compares them. The first using Faster R-CNN [12] combined with a KCF tracker [13], and the second using Tiny YOLOv2 [14, 15]. In Section IV we discuss our results in terms of accuracy and running time. Finally, our conclusions are given in Section V.

II. DATASET

A. Image Dataset

To train the detector, we built a dataset of pedestrian traffic light images in their typical environments. A ground truth bounding box was marked manually for each traffic light. Traffic lights have a different appearance in different countries. We focus on the appearance of traffic lights in Israel at daytime. Since we use a supervised machine learning approach, it should be straightforward to extend our system to deal with other pedestrian traffic light appearances by extending the dataset with such images and retraining the classifier.

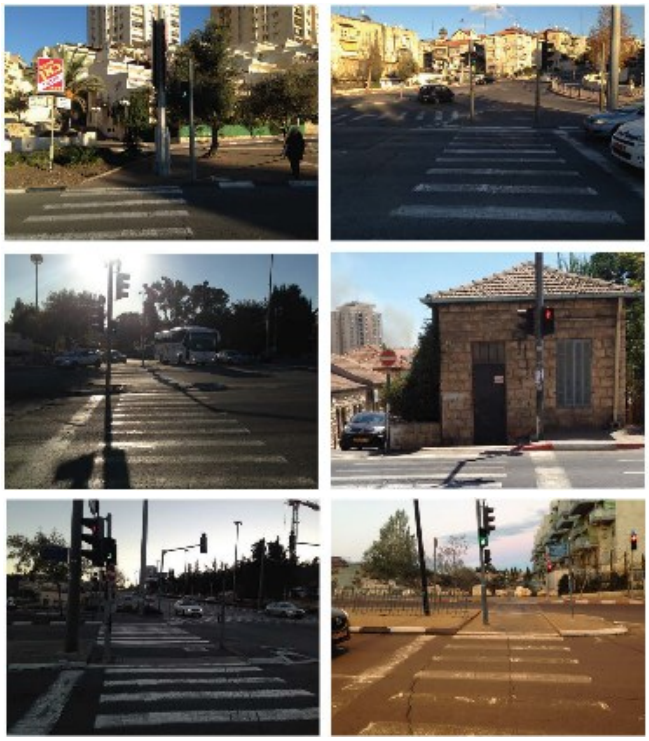

Fig. 1. Example images from our image dataset.

Our dataset contains 950 color images - 450 green 'walk' lights and 500 red 'don't walk' lights. Images were captured according to the following guidelines:

- Taken by a smartphone camera in a standard configuration.
- High resolution (one million pixels and beyond).
- Taken from the position and viewpoint where pedestrians have to wait for a 'walk' signal. From such viewpoint, the angle of the relevant pedestrian traffic light is about frontal.
- The relevant pedestrian traffic light is not occluded.
- Taken during daytime.
- No zoom.

As demonstrated in Fig. 1, following these guidelines simulates images taken by a future application. Still, the images have high diversity – pedestrian traffic lights are at different distances (sizes), illumination can vary, the luminance of a traffic light lamp can vary, and the scene structure, including vegetation, people and vehicles, can be significantly different in different images.

B. Video Dataset

For more robust detection of pedestrian traffic lights, one can exploit the fact that abrupt switching from red to green or vice versa is unique to traffic lights. To assess the accuracy of our technique in detecting this abrupt switching, we built a dataset of 121 short videos (several seconds long). Each video captures a single transition or no transition at all - 50 green to red transitions, 54 red to green transitions, and 17 videos with no transition. An example of such a traffic light switching is shown in Fig. 2.

Fig. 2. Consecutive frames from a video sequence in our video dataset. Note that traffic light switching is abrupt. In this example, in frame 1 to 3 the light is green and in frame 4 it turns to red.

III. PEDESTRAIN TRAFFIC LIGHT DETECETION

We investigate two variants of our algorithm for pedestrian traffic light detection, both of which share the same generic scheme, as depicted in Fig. 3. The video is analyzed frame-by-frame. First, candidates for pedestrian traffic lights are detected by a generic object detector adapted for localizing and recognizing pedestrian traffic lights. Next, a decision is made as to whether it is safe to cross the road. As explained later in this section in detail, the decision can be based only on simple spatial and temporal rules or can be more sophisticated, based also on object tracking. 'Walk' indication is given when a relevant 'walk' traffic light is detected with very high confidence. 'Don't walk' indication is given in one of three possible cases:

- A relevant 'don't walk' traffic light is detected.
- A relevant 'walk' traffic light is detected with a confidence score that is not high enough.
- No relevant traffic light is detected.

The decision module analyzes the live video stream and checks for consistent detections across consecutive video frames. To guarantee safe crossing, if the detection confidence score of a 'walk' traffic light is not high enough, the algorithm outputs an indication not to cross. In this case, the traffic light must first turn red and then, when it turns green again, the algorithm indicates that it is safe to cross. This is the same approach used in many accessible pedestrian signals. Since we assume that the user does not change his position very fast and the rotation angle is small, the decision module can assume only a small translation between two consecutive frames.

A. First variant: Faster R-CNN with a KCF Tracker

The first variant of our algorithm uses the Faster R-CNN object detector [12]. Faster R-CNN consists of two main parts – a Region Proposal Network (RPN) that proposes candidate regions and a Fast Region-based Convolutional Neural Network (Fast R-CNN) [16] that performs object detection in the proposed regions. During training, the following loss function is minimized:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (1)$$

This loss function is used to train the Fast R-CNN part of Faster R-CNN. A similar loss function is used to train the RPN part of Faster R-CNN. It considers the classification loss L_{cls} and the bounding box regression loss L_{reg}. During RPN training, bounding boxes are predicted relative to a fixed set of reference bounding boxes called anchors. For the RPN, p_i is the probability of anchor i being an object. The ground-truth label p_i^* is 1 if the anchor is positive, and is 0 if the anchor is negative. Similarly, for the Fast R-CNN part of Faster R-CNN, p_i and p_i^* are defined as the estimated probability and ground-truth indicator respectively of class i. t_i is a vector representing the 4 parametrized coordinates of the predicted bounding box, and t_i^* is that of the ground-truth box associated with a positive anchor, or region proposal. N_{cls} and N_{reg} are two normalizing parameters and λ is a balancing parameter. We use a Faster R-CNN network consisting of five layers - three convolutional layers and two fully-connected layers. The classifier was pre-trained on the CIFAR-10 dataset [17], consisting of 60,000 32x32 color images in 10 classes, and transfer learning was performed to our image dataset with two classes ('walk' and 'don't walk' pedestrian traffic light).

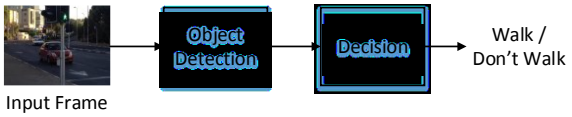

Fig. 3. Generic scheme of proposed pedestrian traffic light detection.

An object tracker based on Kernelized Correlation Filters (KCF) [13] is at the heart of the decision module. KCF is a state-of-the-art object tracker using a Gaussian kernel and histogram-of-oriented-gradients (HOG) features. The tracker is highly efficient due to its use of the 'kernel trick' so all operations are performed in the frequency domain. In each frame, the object detector's output bounding boxes are matched to the tracker's output bounding boxes using the intersection-over-union (IoU) similarity measure. For objects detected by the detector but not by the tracker, the tracker is updated to start tracking those objects. Objects detected by the tracker but not by the detector are considered false alarms and are discarded. The final detection of a pedestrian traffic light is done for a bounding box detected by Faster R-CNN with a high confidence score and with an indication of a light switch (continuous tracking and change of detected object from 'walk' signal to 'don't walk' signal or vice versa). In case of more than one candidate bounding box in a frame, the decision module selects the largest bounding box due to the assumption that the relevant traffic light is the closest to the pedestrian. We found that this strategy gives better results than selecting the bounding box with the highest confidence score.

B. Second variant: YOLOv2

The second variant of our algorithm uses the YOLO (You Only Look Once) object detector [14]. YOLO divides the input image into a grid of SxS cells. Each grid cell predicts B bounding boxes for which it finds the boundaries (x, y, w, h) and objectness confidence score that quantifies how likely it is for a bounding box to contain an object of any class. In addition, each grid cell predicts probabilities $p_i(c)$ of C classes. In total, each grid cell has $5B + C$ channels. A final score is calculated for each bounding box, taking into account its objectness score and class scores. Finally, all bounding boxes with a final score less than a determined threshold are discarded. The YOLO architecture consists of 24 convolutional layers and two fully connected layers. During training, YOLO minimizes a loss function that considers three losses between predictions and ground truth – localization loss, confidence loss and classification loss:

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{obj}[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2] \quad (2)$$
$$+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right]$$
$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2$$
$$+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{noobj}\left(C_i-\hat{C}_i\right)^2$$
$$+\sum_{i=0}^{S^2}\mathbb{I}_{i}^{obj}\sum_{c\in classes}(p_i(c)-\hat{p}_i(c))^2$$

The localization loss, given by the first two terms, is the error between the predicted and ground-truth bounding box. The confidence loss, given by the third and fourth terms, is the objectness of the bounding box. The classification loss is given by the last term. $\mathbb{I}_i^{obj}$ denotes if object appears in cell i and $\mathbb{I}_{ij}^{obj}$ denotes that the jth bounding box predictor in cell i is 'responsible' for that prediction. The loss function only penalizes classification error if an object is present in that grid cell. It also only penalizes bounding box coordinate error if that predictor is 'responsible' for the ground truth box. λ_{coord} and λ_{noobj} are a balancing parameters.

A more recent version of YOLO called YOLOv2 (or YOLO9000) is proposed in [18]. This version incorporates several changes compared to YOLO that are reported to improve results substantially. It contains 19 convolutional layers and 5 max-pooling layers. YOLOv2 has several configurations that allow a tradeoff between speed and detection accuracy. As running in real-time is crucial for our application, we use the Tiny YOLO configuration [15] that is much faster and only slightly less accurate than the standard YOLO model. Tiny YOLOv2 uses only 9 out of the 19 convolutional layers. The classifier was pre-trained on the PASCAL VOC 2007 and PASCAL VOC 2012 datasets [19], consisting together of tens of thousands of objects in 20 classes. Transfer learning was performed to our image dataset with $S = 13$, $B = 5$, and $C = 2$ (one class for 'walk' pedestrian traffic lights and another class for 'don't walk' pedestrian traffic lights).

Similarly to the first variant, in case of more than one candidate bounding box in a frame, the decision module selects the largest bounding box. However, due to the high detection accuracy, no object tracker is required here and the decision module is much simpler than in the first variant. For better robustness to temporary false detections, the output of the decision module is the median of the decision in 5 consecutive frames.

IV. RESULTS

In this section, we compare object detectors in terms of detection accuracy and running time. We also report the detection accuracy of pedestrian traffic lights in video. Table 1 compares the object detectors described in Section III – the first variant (based on Faster R-CNN) and the second variant (based on YOLOv2 or Tiny YOLOv2). The first variant was implemented in MATLAB under Windows. The second variant uses the official YOLO implementation in the Darknet framework under Linux. Results were obtained with cross-validation with 80% training set, 10% validation set and 10% test set. As shown in Table 1, both variants detect pedestrian traffic lights with high accuracy and operate in real-time. The

Fig. 4. Example object detection results of Faster R-CNN (left column) and Tiny YOLOv2 (right column). Candidate objects are marked with a bounding box and a confidence score. Final relevant pedestrian traffic light detection (based only on one frame without temporal infromation) is given in the uper left corner of each image.

second variant has higher precision and recall values and substantially lower running time compared with the first variant. Tiny YOLOv2 is slightly less accurate than YOLOv2 but has a substantially lower running time.

Table 1. Comparison of pedestrian traffic light object detectors. Running times are for a desktop computer with Intel Core i7 CPU and NVIDIA GeForce GTX 1080 GPU.

	'Walk'		'Don't Walk'		Running Time [ms]
	Precision	Recall	Precision	Recall	
Faster R-CNN	98.8	94.2	98.1	97.6	50
YOLOv2	100	98.31	100	97.14	15.4
Tiny YOLOv2	100	94.92	100	100	6

Fig. 4 depicts several pedestrian traffic light detection results for Faster R-CNN and Tiny YOLOv2. Due to its lower precision and recall, Faster R-CNN outputs substantially more false candidate bounding boxes compared with Tiny YOLOv2. Typical false detections are vehicle traffic lights, people wearing green or red shirts and vehicle wheels. A disadvantage of Tiny YOLOv2 is that it struggles with detecting small (distant) objects. This matter is of minor importance as it is reasonable to assume that the relevant traffic light is 'large enough'. Fig. 4g depicts a rare example of misdetection by both detectors. Faster R-CNN detects both traffic lights but still selects the furthest one as the relevant traffic light, while Tiny YOLOv2 does not detect any of the traffic lights due to difficult illumination conditions.

As explained above, we exploit the fact that abrupt switching from red to green or vice versa is unique to traffic lights. We tested for the correct detection of this abrupt switching with all 121 videos in our video dataset. In the case of correct pedestrian traffic light detection, both variants of the proposed algorithm have detected light switching correctly. The second variant, based on Tiny YOLOv2, performs better overall in terms of both accuracy and running time. It operates at 6 ms per frame and achieves 99.99% success rate in detecting pedestrian traffic lights. All false detections resulted in a 'don't walk' indication to make sure that the pedestrian remains safe.

V. Conclusions

In this paper, we proposed a solution for pedestrian traffic light detection using efficient algorithms based on convolutional neural networks. The proposed solution is composed of an object detector and a decision module. Our results show that a solution based on Tiny YOLOv2 object detector achieves better results than a solution based on Faster R-CNN object detector. This solution achieves high enough accuracy and fast enough running time that are suitable for a mobile application that will help visually impaired pedestrians cross a road. In the future, we plan to improve robustness and add supported scenarios by training and testing the proposed solution on a larger dataset. We also plan to develop a complete solution that includes an application that runs on a mobile phone and is based on a client-server architecture.

VI. Acknowledgment

The authors would like to thank the Technion's social hub for partly funding this work. The authors would also like to thank the staff of Signal and Image Processing Lab. (SIPL) at the Technion and especially Prof. David Malah, Nimrod Peleg and Evgeny Slavin, for their continuous assistance and support.

VII. References

[1] "Blindness and Visual Impairment. Fact Sheet," ed: World Health Organization, 2017.

[2] M. Hersh and M. A. Johnson, *Assistive Technology for Visually Impaired and Blind People*: Springer Science & Business Media, 2010.

[3] K. Rothaus, J. Roters, and X. Jiang, "Localization of Pedestrian Lights on Mobile Devices," in *Proceedings: APSIPA ASC: Asia-Pacific Signal and Information Processing Association, Annual Summit and Conference*, 2009, pp. 398-405.

[4] M. Diaz, P. Cerri, G. Pirlo, M. A. Ferrer, and D. Impedovo, "A Survey on Traffic Light Detection," in *International Conference on Image Analysis and Processing*, 2015, pp. 201-208.

[5] J. Roters, X. Jiang, and K. Rothaus, "Recognition of Traffic Lights in Live Video Streams on Mobile Devices," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 21, pp. 1497-1511, 2011.

[6] V. Ivanchenko, J. Coughlan, and H. Shen, "Real-time Walk Light Detection with a Mobile Phone," in *International Conference on Computers for Handicapped Persons*, 2010, pp. 229-234.

[7] J. M. Coughlan and H. Shen, "Crosswatch: A System for Providing Guidance to Visually Impaired Travelers at Traffic Intersection," *Journal of Assistive Technologies,* vol. 7, pp. 131-142, 2013.

[8] S. Mascetti, D. Ahmetovic, A. Gerino, C. Bernareggi, M. Busso, and A. Rizzi, "Robust Traffic Lights Detection on Mobile Devices for Pedestrians with Visual Impairment," *Computer Vision and Image Understanding,* vol. 148, pp. 123-135, 2016.

[9] S. Mascetti, D. Ahmetovic, A. Gerino, C. Bernareggi, M. Busso, and A. Rizzi, "Supporting Pedestrians with Visual Impairment During Road Crossing: A Mobile Application for Traffic Lights Detection," in *International Conference on Computers Helping People with Special Needs*, 2016, pp. 198-201.

[10] R. Cheng, K. Wang, K. Yang, N. Long, J. Bai, and D. Liu, "Real-time Pedestrian Crossing Lights Detection Algorithm for the Visually Impaired," *Multimedia Tools and Applications,* pp. 1-21, 2017.

[11] X.-H. Wu, R. Hu, and Y.-Q. Bao, "Fast Vision-Based Pedestrian Traffic Light Detection," in *IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2018.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91-99.

[13] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed Tracking with Kernelized Correlation Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI),* vol. 37, pp. 583-596, 2015.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779-788.

[15] J. Redmon. (2016). *YOLO: Real-time Object Detection - YOLO v2*. Available: https://pjreddie.com/darknet/yolov2/

[16] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440-1448.

[17] A. Krizhevsky and G. Hinton, "Learning Multiple Layers of Features from Tiny Images," Master's thesis, Department of Computer Science, University of Toronto, 2009.

[18] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517-6525.

[19] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision,* vol. 111, pp. 98-136, 2015.