International Journal of Mechatronics, Electrical and Computer Technology (IJMEC)
ISSN: 2305-0543 (Online), ISSN: 2411-6173 (Print)
Vol. 13 (47), Jan. 2023, pp. 5333-5337

Low-Cost Embedded Multi-Object Detection Using Deep Neural Network

Kamal Ahmadiyan
Department of Electrical and Computer Engineering
Isfahan University of technology, Isfahan, Iran
kamal-ahmadiyan@ec.iut.ac.ir
Tel Number: +98-9135581951

Navid Daneshmandpour
Department of Electrical and Electronics Engineering,
Shiraz University of Technology, Shiraz, Iran.
N.Daneshmand@sutech.ac.ir
Tel Number: +98-9913929909

***Abstract*— This paper presents a new method for detecting both vehicles and traffic signs, simultaneously. The proposed method is designed for self-driving cars that need low-cost and low processing power embedded systems like Raspberry Pi. Also, high-performance object detection algorithms of deep learning are hired. This paper uses three models: SSD MobileNet v1, YOLO v3, and YOLO v3-tiny that perform well on embedded computers. The algorithm is implemented on 900 images of the GTSDB dataset for learning traffic signs and 11530 images of the VOC dataset for learning vehicles. The proposed method is tested in a real environment using two Raspberry Pi by the Raspberry Pi camera module v2. The speed of the proposed system is 1.4 frames per second at a high accuracy level.**

Keywords*— Self-driving car, Traffic signs detection, Vehicle detection, YOLO, Embedded

I. INTRODUCTION

Recently, self-driving cars and automated vehicles (AV) are used to reduce driving accidents and improve safety. Interpreting traffic scenes, such as detecting and recognizing vehicle and traffic signs, plays an essential role for automated vehicles.

Detecting objects using classical image processing methods is not efficient enough. However, deep learning models have excellent performance even under challenging conditions, like variable lighting conditions, motion, and weather conditions. The main disadvantage of deep learning is its high processing power which makes it impossible to perform on low-cost embedded computers.

Many researches have been investigated the problem of detecting traffic signs. In [1], traffic sign detection is performed using the YOLO v2 and YOLO v2-Tiny models in GTSDB (that use just 4 of 43 classes). The method succeeded in reaching seven frames per second (fps) and 73% accuracy in raspberry pi 3B. The main disadvantage of [1] was implementing the deep learning models on only a small section of the dataset. Also, the algorithm was not tested in the real environment which makes the results unrealistic.

Most deep learning methods that attempt to detect traffic signs use high-resolution images [2], [3]. However, in practical cases, the traffic images are not very high quality, as they need to be processed on an embedded system with limited processing power.

Reference [4] detects and tracks the vehicles using classical image processing methods such as the Kalman filter. This method has a high processing speed of 12 frames per second. However, various conditions such as lighting conditions and motion have not been considered in the test images.

Other research in deep learning is performed to detect the vehicle using the SSD MobileNet model, resulting in 0.9 frames per second on Raspberry Pi 3B [5].

In this paper, we present a practical system for detecting traffic signs and vehicles simultaneously. The proposed system uses two Raspberry Pi boards implementing deep learning models. The boards are connected with each other via network ports. In addition, a Raspberry Pi camera module v2 is used for testing in real situations.

This paper is structured as follows. In Section II, the proposed system is explained. First, we review the existing datasets in the field of detecting traffic signs and vehicles. Then, object detection models are compared and suitable models for implementing on Raspberry Pi are selected. The procedure of simultaneously detect traffic signs and vehicles in two Raspberry Pi and their results are provided in Section III. Finally, Section IV concludes the paper with a summary and gives some potential directions for future research.

II. THE PROPOSED SYSTEM

In this section, by reviewing the available datasets and models on vehicle and traffic signs detection, we will train one of the datasets on models and see the preliminary results for the final implementation.

Article History:
Received Date: Sep. 16, 2022, Accepted Date: Dec. 15, 2022, Available Online: Jan. 01, 2023

A. DATASETS

1. Vehicles Datasets:

Visual datasets with labels are used to train, evaluate deep learning models and lead to success in computer vision with novel architectures, such as Yolo v3, Yolo v3-Tiny, Yolo v2, and SSD MobileNet v1. Many datasets are available for vehicle detection training. Some of these are seen here:

- COCO 2020: this dataset is containing more than 200000 images and 80 object categories. It is possible to extract vehicle identification data and train the model. the first 14 classes of the COCO dataset related to transportation, including car, bus, etc. This dataset can be used to extract the data related to vehicle detection and use them for training the model.[6]
- Pascal VOC 2012: this dataset has 20 classes, 11530 images containing 27450 ROI annotated objects and 6929 segmentations. Like COCO, we can extract vehicle datasets on it and use them for training.[7]

In this paper, we use both COCO and Pascal VOC datasets for learning the neural network.

2. Traffic sign datasets:

There are several datasets for traffic signs, such as the German Traffic Sign Detection Benchmark (GTSDB) dataset [8], Chinese Traffic Sign Database [9], Tsinghua-Tencent 100K dataset [10]. This paper uses the GTSDB dataset because it has 900 images and 43 classes divided into four categories system (prohibitory, mandatory, danger, others) that suit the properties of various detection approaches with different properties. By considering the importance of the symbols, we selected 18 classes out of 43 into four categories, as follows: prohibitory entail speed limit (30km/h), speed limit (80km/h), and no passing; mandatory made up turn right, turn left, ahead only, keep right, keep left, and roundabout mandatory; danger contain bumpy road, road work, traffic signals, pedestrians, and children crossing; others which have yield, stop, no entry, and general caution.

B. Object detection model

Various models have been trained and tested. SSD MobileNet v1, YOLO v3, Tiny-YOLO v3 had the best overall results in Raspberry Pi. They present below.

1. SSD MobileNet v1

The SSD (Single Shot MultiBox Detector) architecture is a single convolution network that learns to predict bounding box locations and classify these locations into a single pass. The SSD network consists of base architecture (MobileNets in this case) followed by several convolution layers see in fig. 1 SSD operates on feature maps to detect the location of bounding boxes, which does not predict the shape of the box exactly where the box is located [11].

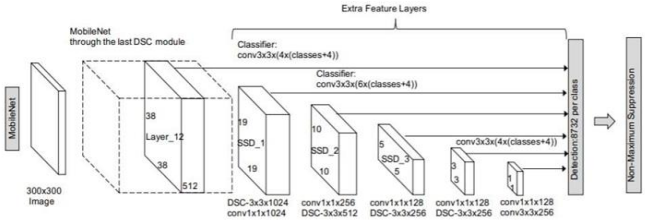

Fig. 1.SSD MobileNet Layered Architecture [11]

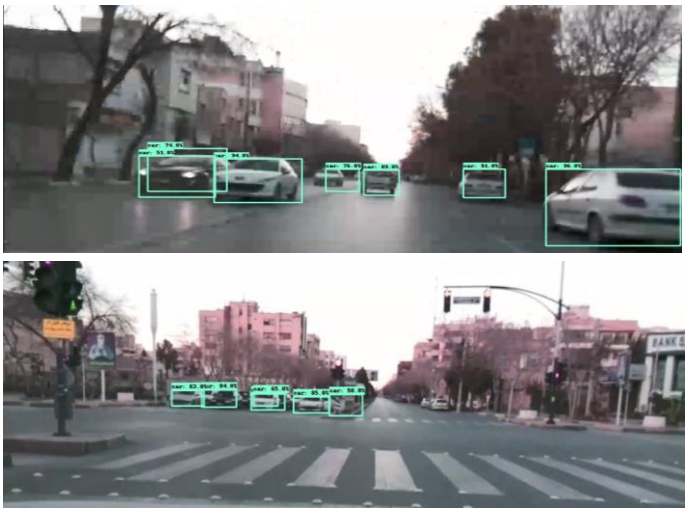

Fig. 2. Tested SSD-MobileNet v1 model on real environment images that detect the vehicle.

We train the lite and quantize model of SSD-MobileNet v1 on the COCO dataset that detects the vehicles faster than other SSD-MobileNet. By testing the model and observing the results, we find that this model achieves a balance between performance accuracy and detection time. Detection time is about 0.6 second per frame or 1.7 fps, which is not too bad compared to YOLO. You can see some results in Fig. 2.

2. YOLO v3

YOLO is one of the faster object detection algorithms; however, it is no longer the most accurate object detection algorithm. It is a very good choice when we need real-time detection without the loss of too much accuracy. This model is simple to construct and can be trained directly on entire images. Unlike classifier-based approaches, YOLO is trained on a loss function that directly corresponds to detection performance, and the entire model is trained jointly [12].

YOLOv2 is the second version of YOLO that aims to increase accuracy while speeding up standard diagnostic tasks such as PASCAL VOC, COCO, and GTSDB. YOLO v2 used 30-layer architecture, originally a 19-layer network that comes from darknet-19, supplemented with 11 more layers for object detection. YOLO v2 often struggled with small object detections since this was attributed to the loss of fine-grained features as the layers downsampled the input. To remedy this, YOLO v2 used identity mapping, concatenating feature maps from a previous layer to capture low-level features [13].

YOLO v3 release is based on the Darknet-53 that better detects small objects. While the accuracy increased dramatically with this version, it traded off against speed, reducing around ten frames per second. In low-resolution

IEEE Office - SEET Complex, Federal University of Technology, Owerri, Nigeria, https://aeuso.org
EISSN: 2305-0543, PISSN: 2411-6173

cameras, such as the Raspberry Pi camera module v2, YOLOv3 acts as a low-cost, fairly accurate detector that makes detection possible in the low time (Fig. 3).

YOLOv3 refined the design further by using tricks, such as multi-scale prediction and bounding box prediction, through logistic regression. The Darknet-53 has 53 layers, but 53 more layers are stacked onto it for the detection task, giving us a 106 layer fully convolutional underlying architecture for YOLO v3 [14].

This section uses the YOLO v3 model to detect traffic signs, which have been training on the GTSDB dataset in 18 out of 43 classes. By testing the model and observing the results, the accuracy of our detection is very high, about 97%, which is very desirable; however, the speed of the detection in Raspberry Pi (CPU only) is approximately 0.5 frames per second, which is about 80 times less than using the computers with GPU. Network structure of YOLO v3-Tiny is shown in Table 1[15].

3. *YOLO v3-Tiny*

The light version of the YOLO model is Tiny-YOLO, a deep neural network for real-time object detection. This model increases speed by reducing layers while reducing weight and accuracy.

By changing the structure of YOLO V3 and reducing the number of its main layers to only seven convolutional layers, we get a lightened model of YOLO v3 called Tiny-YOLO v3. Furthermore, Tiny-YOLO v3 uses the pooling layer instead of YOLO v3's convolutional layer with a step size of 2 to achieve dimensionality reduction, and features are extracted by using a small number of 1x1 and 3x3 convolutional layers [15].

Running speed increases significantly, but detection accuracy decreases, though by sacrificing a little speed, we get more relative accuracy to YOLO v2. In Table 2, we can see that we have compared different YOLO models in different parts, such as speed and weight, which have been done in high-performance GPUs and standard datasets.

According to the procedure of each part of the paper, here we train the YOLO v3-Tiny model with the VOC dataset to detect the Vehicle to have an estimate of the execution time and the accuracy obtained from it in Raspberry Pi for the final stage.

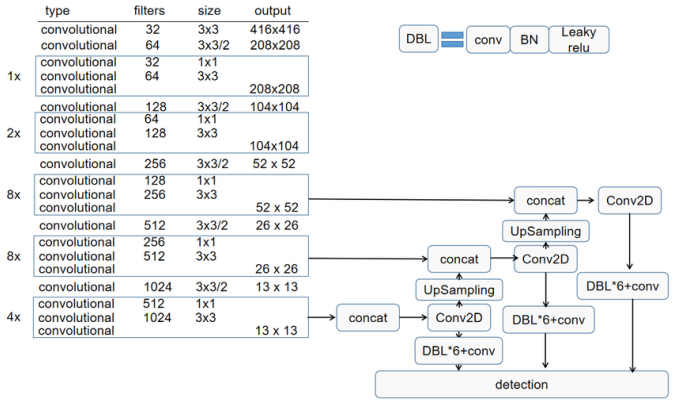

Fig. 3. YOLO v3 network structure [16]

Table 1. network structure of YOLO v3-Tiny

layer	Type	Filters	Size/Stride	Input	Output
0	Convolutional	16	3×3/1	416×416×3	416×416×16
1	Maxpool	-	2×2/2	416×416×16	208×208×16
2	Convolutional	32	3×3/1	208×208×16	208×208×32
3	Maxpool		2×2/2	208×208×32	104×104×32
4	Convolutional	64	3×3/1	104×104×32	104×104×64
5	Maxpool	-	2×2/2	104×104×64	52×52×64
6	Convolutional	128	3×3/1	52×52×64	52×52×128
7	Maxpool	-	2×2/2	52×52×128	26×26×128
8	Convolutional	256	3×3/1	26×26×128	26×26×256
9	Maxpool	-	2×2/2	26×26×256	13×13×256
10	Convolutional	512	3×3/1	13×13×256	13×13×512
11	Maxpool	-	2×2/1	13×13×512	13×13×512
12	Convolutional	1024	3×3/1	13×13×512	13×13×1024
13	Convolutional	256	1×1/1	13×13×1024	13×13×256
14	Convolutional	512	3×3/1	13×13×256	13×13×512
15	Convolutional	255	1×1/1	13×13×512	13×13×255
16	YOLO	-	-	-	-
17	Route 13	-	-	-	-
18	Convolutional	128	1×1/1	13×13×256	13×13×128
19	Up-sampling	-	2×2/1	13×13×128	13×13×128
20	Route 19 8	-	-	-	-
21	Convolutional	256	3×3/1	13×13×384	13×13×256
22	Convolutional	255	1×1/1	13×13×256	13×13×256
23	YOLO	-	-	-	-

Table 2. Comparison of different versions of YOLO and Tiny YOLO. [17]

Model	Number of layers	Flops(B)	FPS	mAP	Dataset
YOLOv1	26	-	45	63.5	VOC
YOLOv1-TIny	9	-	155	52.8	VOC
YOLOv2	32	62.95	40	48.2	COCO
YOLOv2-TIny	16	5.42	244	23.6	COCO
YOLOv3	106	140.70	20	57.8	COCO
YOLOv3-TIny	24	5.57	220	33.2	COCO

Fig. 4 shows that the detection accuracy is high, which is done at an approximate speed of 2 frames per second; this performance was the best performance among the models introduced in this section.

III. IMPLEMENTATION AND OVERALL RESULTS

In the previous sections, we looked at the variant datasets and models that are suited for embedded computer vision/deep learning devices such as the Raspberry Pi, Google Coral, and NVIDIA Jetson Nano. At each part of the last section, we trained one of the traffic signs or vehicle detection to see the result on accuracy and speed on Raspberry Pi for decide to use which for the main purpose of the paper, which is the simultaneous detection of vehicles and traffic signs.

For simultaneous detection, we used two Raspberry Pi 4s model B, which both have 4 Giga RAM, CPU only (ARM Cortex-A72,1.5 GHz, quad-core,64-bit) and Raspberry Pi camera module v2-8 Megapixel,1080p. We connected the Raspberry Pi camera module v2 to one of the Raspberry Pi connected via Ethernet cable and formed a local network. Because of using pair two raspberry pi in the car, we use a

power bank to power them. First Raspberry Pi is responsible for taking photos and detect vehicles, and sent the picture and localization of the bounding box to the second raspberry pi to find traffic signs. Output produces by placing the location of the two bounding boxes on the image, and the detection process is complete.

Before beginning the detection process, we use pre-processing on the image to select ROI (Region of interest) that the detection process takes place there. This pre-processing is actually due to the placement of the camera in front of the car window, with a segmentation of the image could delete the end of the picture that the opposite part of the car and no effect on detection. Doing this process helps increase the speed of the detection process so that we have experienced an average speed increase of about 0.25 frames per second.

According to the results obtained in the previous part in model training and testing, the YOLO v3-Tiny model was the best model in the trade-off between accuracy and speed; in addition to its small size made it useful for traffic signs or vehicles detection in Raspberry Pi. After reviewing the datasets and the effect of their accuracy on detection, we decide to use the VOC dataset for vehicle detection and the GTSDB for traffic signs detection, which is compatible with many traffic signs in different countries and can be used practically.

Since the speed of vehicle detection is about two frames per second, and it is faster than the speed of traffic signs detection, which is 1.6 frames per second, that using the YOLO-Tiny model, so in the first Raspberry Pi, we will detect the vehicle and after the initial processing of the image and determine the position of the bounding boxes, send them to the second Raspberry Pi, where we first find the position of the traffic signs, and then merge it with the localization of the vehicle's bounding boxes and make the output image.

Fig. 5 one of the images due to the threshold level the bottom has an error. Because of these errors, set the vehicle detection threshold level to 0.3 (30%), and the traffic signs detection has a threshold level of 0.5 (50%). This threshold level was achieved based on experiments on about 200 images obtained in different conditions.

The simultaneous detection speed is due to the image transfer between the two Raspberry Pi up to 1.2 frames it drops by the second. However, on average, we have experienced a speed of 1.4 frames per second in 200 images (Fig. 6).

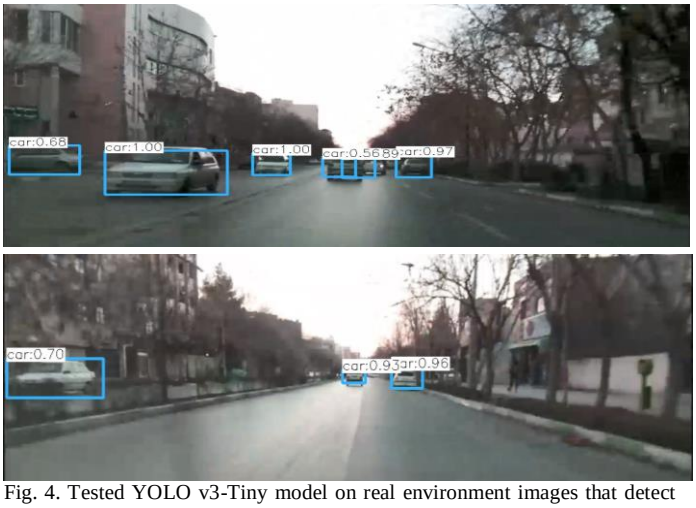

Fig. 4. Tested YOLO v3-Tiny model on real environment images that detect the vehicle

Fig. 5. Detection Error due to the low threshold level of the correct detection.

Fig. 6. Simultaneous vehicle and traffic sign detection

IV. CONCLUSION

In this paper, we implemented a powerful multi-object detection system on Raspberry Pi computers. The system can detect traffic signs and vehicles at the same time. It takes advantage of state-of-the-art deep learning models like YOLO v3-Tiny performing on famous image datasets like GTSDB for learning traffic signs and VOC for learning vehicles. Using two Raspberry Pi connected via network ports, we can simultaneously detect and recognize traffic signs and vehicles. The proposed system succeeded in performing at a detection rate of 1.4 frames per second. The main advantage of the proposed system is implementing two powerful deep learning models on a low-cost and low processing power computer. As future work, we plan to work on deep neural models to build an algorithm with suitable speed and accuracy for the embedded computers with limited processing power and memory.

IEEE Office - SEET Complex, Federal University of Technology, Owerri, Nigeria, https://aeuso.org
EISSN: 2305-0543, PISSN: 2411-6173

REFERENCES

[1] P. S. Zaki, M. M. William, B. K. Soliman, K. G. Alexsan, K. Khalil, and M. El-Moursy, "Traffic Signs Detection and Recognition System using Deep Learning," Mar. 2020, Accessed: Jun. 11, 2021. [Online]. Available: http://arxiv.org/abs/2003.03256.

[2] D. Priyanka, K. Dharani, C. Anirudh, K. Akshay, M. P. Sunil, and S. A. Hariprasad, "Traffic light and sign detection for autonomous land vehicle using Raspberry Pi," in *Proceedings of the International Conference on Inventive Computing and Informatics, ICICI 2017*, May 2018, pp. 160–164, doi: 10.1109/ICICI.2017.8365328.

[3] N. S. R. Et. al., "Implementation paper of Traffic Signal Detection and Recognition using deep learning," *Turkish J. Comput. Math. Educ.*, vol. 12, no. 1S, pp. 212–219, Apr. 2021, doi: 10.17762/turcomat.v12i1s.1760.

[4] M. Anandhalli and V. P. Baligar, "A novel approach in real-time vehicle detection and tracking using Raspberry Pi," *Alexandria Eng. J.*, vol. 57, no. 3, pp. 1597–1607, Sep. 2018, doi: 10.1016/j.aej.2017.06.008.

[5] "Real-Time Vehicle Detection with MobileNet SSD and Xailient." https://www.xailient.com/post/real-time-vehicle-detection-with-mobilenet-ssd-and-xailient (accessed Jun. 11, 2021).

[6] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context."

[7] "The PASCAL Visual Object Classes Challenge 2012 (VOC2012)." http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html (accessed Dec. 25, 2020).

[8] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German traffic sign detection benchmark," 2013, doi: 10.1109/IJCNN.2013.6706807.

[9] "Traffic Sign Detection Database." http://www.nlpr.ia.ac.cn/pal/trafficdata/detection.html (accessed Dec. 26, 2020).

[10] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-Sign Detection and Classification in the Wild," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, vol. 2016-December, pp. 2110–2118, doi: 10.1109/CVPR.2016.232.

[11] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," doi: 10.1007/978-3-319-46448-0.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[13] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6517–6525, 2017, doi: 10.1109/CVPR.2017.690.

[14] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement." [Online]. Available: https://pjreddie.com/yolo/.

[15] W. He, Z. Huang, Z. Wei, C. Li, and B. Guo, "TF-YOLO: An Improved Incremental Network for Real-Time Object Detection," doi: 10.3390/app9163225.

[16] X. Ding and R. Yang, "Vehicle and Parking Space Detection Based on Improved YOLO Network Model," in *Journal of Physics: Conference Series*, Nov. 2019, vol. 1325, no. 1, doi: 10.1088/1742-6596/1325/1/012084.

[17] R. Huang, J. Pedoeem, and C. Chen, "YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers," *Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018*, pp. 2503–2510, 2019, doi: 10.1109/BigData.2018.8621865.