Embedded Self Organizing Systems (Vol 8. No 2.) (pp.15-19)

Embedded Selforganizing Systems

Issue Topic: "Towards Autonomous Driving Using Vision Based Intelligent Systems "

Traffic Light and Back-light Recognition using Deep Learning and Image Processing with Raspberry Pi

Julkar Nine
Technische Universität Chemnitz
Department of Computer Engineering
julkar.nine@informatik.tu-chemnitz.de

Rahulganapathi Mathavan
Technische Universität Chemnitz
M.Sc. Embedded Systems
rahulganapathi.mathavan@s2019.tu-chemnitz.de

Abstract—Traffic light detection and back-light recognition are essential research topics in the area of intelligent vehicles because they avoid vehicle collision and provide driver safety. Improved detection and semantic clarity may aid in the prevention of traffic accidents by self-driving cars at crowded junctions, thus improving overall driving safety. Complex traffic situations, on the other hand, make it more difficult for algorithms to identify and recognize objects. The latest state-of-the-art algorithms based on Deep Learning and Computer Vision are successfully addressing the majority of real-time problems for autonomous driving, such as detecting traffic signals, traffic signs, and pedestrians. We propose a combination of deep learning and image processing methods while using the MobileNetSSD (deep neural network architecture) model with transfer learning for real-time detection and identification of traffic lights and back-light. This inference model is obtained from frameworks such as Tensor-Flow and Tensor-Flow Lite which is trained on the COCO data. This study investigates the feasibility of executing object detection on the Raspberry Pi 3B+, a widely used embedded computing board. The algorithm's performance is measured in terms of frames per second (FPS), accuracy, and inference time.

Keywords— MobileNetSSD, Transfer Learning, Computer Vision, Raspberry pi, Object detection.

I. INTRODUCTION

In recent years, intelligent vehicles and self-driving cars have emerged as major research subjects, with the goal of developing ways to improve traffic safety, which is of vital significance in today's world. One such area of research is the detection of traffic signals and the identification of cars' taillights.

Rear-light signal recognition is essential for a driver in understanding the conduct and purpose of the vehicle ahead, to alert drivers of possible hazards, to avoid rear-end collisions and accidents that may result in human casualties or significant property damage. At the present time, more than one million people die every year in road accidents across the globe.

The detection of traffic signals may assist in enhancing driving safety and reducing the number of traffic accidents that occur at junctions. Driving safety may be improved in both self-driving cars and sophisticated driver assistance systems by using an accurate traffic light detecting module. When a self-driving car approaches an intersection, the vehicle's speed is reduced in accordance with safety regulations. As a result, the precision of traffic light detection and identification is essential in many situations.

For traffic light detection, the majority of currently available algorithms use color, shape, and gradient information; for taillight identification, however, the majority of known methods rely on color thresholding and symmetry assumptions. These techniques are ineffective and unreliable in real-world traffic situations that include complicated backdrops, multiple kinds of traffic signals, varied illuminations, motion blur, changing weather conditions, and color shifting. Because of these problems, it is more difficult to analyze traffic lights and taillights.

In this project, we will develop an algorithm that uses image processing and transfer learning to detect and recognize the state of a traffic light and the back-light of a car, analyze the result of the algorithm on the Raspberry Pi 3 B+ for real-time applications, and check and report the algorithm's performance, such as frames per second, accuracy, and inference time.

II. Literature Review

A. Tailight Detection

For a long time, the majority of current techniques, such as [1], [2], [3], [4], [5], were limited to simply detecting taillights in nighttime situations. In recent years, additional methods for detecting car taillights during the day have been suggested. Because car rear lights are red, various color spaces, such as YCrCb, L*a*b*, HSV, or Y'UV, are used to identify red areas in those techniques.

Almagambetov et al. [6] developed a method for detecting vehicle rear lights that could be used both during the day and at night. They begin by detecting areas that are red and white, which are then used to extract potential regions. In order to select pairings of candidate areas, a symmetry test and 3-D histogram comparisons are performed between the candidate regions. The Y-distance between two regions is regarded to be smaller than the height of one area, and their sizes are considered to be comparable. This test involves calculating the Bhattacharyya coefficient for each pair of histograms, and only those pairings with a Bhattacharyya coefficient greater than a certain threshold pass this test. A Kalman filter is used to monitor the movement of each light.

The methods suggested in [7], [8], [9] utilize vehicle detection and then search for lights inside the vehicle areas that have been identified by the systems. Cui et al. [7] {cui_vision-based_2015} utilized the HSV color space to extract red pixels, followed by the OPTICS method to extract the two biggest clusters, in order to select taillight candidates for their study. According to [8], the method presented in [6] makes use of the Y-distance test and then chooses pairings by comparing the sizes and forms of the candidate areas that have been obtained.

According to Chen et al. [9], they developed a technique of detecting rear lights that involves calculating a lamp response function in order to evaluate the intensity of each pixel within a vehicle bounding box that has a red component.

Deep learning methods have also been used to determine whether or not brake lights are illuminated. J. G. Wang et al. [10] describe how they fine-tuned an AlexNet model using a dataset that they had gathered and labelled in order to identify the status of brake lights. The dataset was created by applying a vehicle detector to a collection of pictures and manually labelling the identified cars as either" brake" or" normal," as determined by the authors.

[11] describes another deep learning method for detecting brake lights in real-time. First, the authors used a rapid region-based convolutional neural network (quick RCNN) to identify automobiles, then used a fully convolutional network (FCN) to separate the vehicle rear light areas, and then used an SVM classifier to determine whether or not the vehicle was breaking. Swathy S Pillai developed a system detecting taillights for assessing traffic at night utilizing different morphological procedures like thresholding, filtering, extraction, etc. [15]. According to Zhenwei Shi, adaptive background suppression filters provide a quick and robust technique for traffic light recognition that may be used in a variety of lighting situations

B. Traffic Light Detection

Methods for traffic light identification are often divided into three categories: image processing-based methods, machine learning-based techniques, and map-based techniques [12]. Image processing is used to produce a certain resultant where a single or multiple amount of actions or operations are conducted on the image. Using Color Segmentation and the Circle Hough Transform [13], Dwi H. Widyantoro and Kevin I. Saputra were able to achieve traffic light detection and recognition, whereas Guo Mu was able to achieve the same with RGB to HSV conversion, filtering, histogram of oriented gradients (HOG) features, and support vector machine (SVM) [14].

Despite the fact that the image processing method is straightforward and simple, it passes through crucial stages such as thresholding and filtering. The smallest errors in computations or minor departures from norms throughout these stages may result in confusing results, which is very undesirable in the highly sensitive field of traffic light recognition. To avoid falling into this trap, machine learning-based approaches and algorithms are being tested individually and in combination with many processing techniques in order to prune the incorrect paths. For example, Keyu Lu presented a Convolution Neural Network (CNN) based on the Generalised Haar Filter that may be used for object identification in traffic scenes [17]. Seokwoo Jung developed a CNN-based traffic sign identification algorithm in which the extraction of traffic sign candidates is done in the first stage, and the classification of traffic sign candidates is performed in the second stage using the LeNet-5 CNN architecture [18]. When Gwang-Gook. LEE and Byung Kwan PARK combined a traditional method to image processing with Deep Neural Network (DNN) as a potential classifier [19], they were able to get accurate results for traffic light identification. [20] R. Kulkarni, S. Dhavalikar, and S. Bangar developed a deep learning model called Faster Region-based Convolutional Neural Network (Faster R-CNN) using Inception V2 for tackling traffic light detection, with a focus on Indian cities, and applied it to traffic light detection [21].

III. Methodology

The architecture of the approach (Fig. 1.) is composed of two major components which are Deep Learning and Image processing.

A. MobileNetSSD

A lightweight deep neural network architecture for mobile devices and embedded vision applications. In many real-world applications, such as a self-driving vehicle, the identification tasks must be completed in a timely manner on a device with limited computing capabilities. MobileNet [22] was built in 2017 to meet this need. The depth-wise separable filters that form the foundation of MobileNet are used to construct the core layers. Single Shot Detector (SSD) [23] was also developed by the Google Research team around the same time (2016) to meet the demand for models that can run in real time on embedded devices without a significant trade-off in accuracy. Multiple objects within an image are detected by SSD in a single shot.

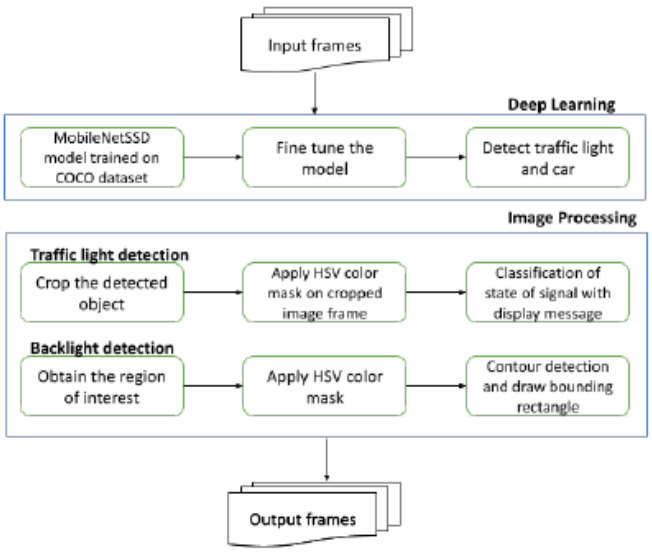

Fig. 1. Flowchart of the proposed approach

(i) MobileNet: makes advantage of depth wise separable convolutions to get its results. When compared to the network with normal convolutions with the same depth in the nets, it substantially lowers the number of parameters. As a consequence of this, we obtain lightweight deep neural networks.

(ii) SSD: seems to remove the proposal step that is common in other methods. Instead, a set of preset bounding boxes is employed, which come in a variety of sizes and aspect ratios. The classifier predicts the presence of an object within those bounding boxes, as well as changes to the bounding box to better match the object's shape. The removal of the region proposal phase and the integration of the complete object identification process into a single network is said to result in a significant speed boost.

B. Hardware

The Raspberry Pi 3 model B+ computer, which is part of a series of tiny single-board computers, was utilized for this project. Among its many features are a 64-bit four-core CPU operating at 1.4GHz, dual-band wireless LAN (2.4GHz and 5GHz), Bluetooth 4.2, a speedier Ethernet connection, and 1GB of LPDDR2 SDRAM. It supports the Micro SD card format for loading the operating system and storing data.

C. COCO Dataset

COCO stands for Common Items in Context, and it is a dataset that is intended to represent a wide variety of objects from 80 various categories that we meet on a regular basis in our daily lives.

The COCO dataset has been labelled, and the data it contains may be used to train supervised computer vision models that are capable of identifying the common items. Though these models are still far from perfect, the COCO dataset serves as a baseline for assessing the periodic improvement of these models that occurs as a result of ongoing computer vision research.

The COCO dataset, which is sponsored by Microsoft, may be utilized for a variety of applications and includes characteristics such as:

(i) A Checkpoint for Transfer Learning: It is made available as a starting point for the training of computer vision models. It is possible to refine the model once it has been trained on the COCO dataset in order for it to learn additional tasks using a custom dataset.

(ii) Object identification, semantic segmentation, key point detection, captioning and other computer vision tasks are some of the tasks that it is used for.

(iii) There are 118,287 train pictures, 40,670 test images, and 5000 validation images in the dataset.

D. Model Optimization

The Raspberry Pi has a limited amount of available memory and processing capacity. Deep learning models may be optimized in a variety of ways so that they can be run within the limitations of their environment. Model optimization aids in the reduction of model size, resulting in reduced memory consumption when the model is installed on the Raspberry Pi and lowers latency, which reduces the amount of computation needed to perform inference using a model. The TensorFlow Lite library includes a number of optimization methods for models that will be executed on edge devices. These include quantization, pruning, and clustering. We choose pre-trained models from TensorFlow Lite that have been optimized via quantization.

IV. Implementation

A. Pre-trained model

There are numerous pre-trained models to choose from. If the job necessitates a high level of precision, we may need a big and complicated model. The usage of a smaller model is recommended for jobs requiring less accuracy since they not only take up less disc space and memory but are also usually quicker and more energy-efficient.

B. Detection of traffic light and car with model tuning

An object detection model is trained to identify the presence and position of several kinds of items. When an image or a video stream is later given to the model, it will return a list of the items it identifies, the position of a bounding box that includes each object, and a score that reflects the confidence that detection was accurate.

(i) Input to the model: The model accepts an image or video stream as input. Let us suppose the anticipated picture is 300x300 pixels, with three channels (red, blue, and green) for each pixel. This should be given to the model as a flattened buffer of 270,000 byte values (300x300x3) (300x300x3). If the model is quantized, each value should be a single byte indicating a value between 0 and 255.

(ii) Output of the model:

Classes: Array of N numbers (output as floating-point values, each representing the index of a class label from the labels file, which corresponds to the object that the model was trained to recognize.

Location: For each identified item, the model will provide an array of four integers indicating a bounding rectangle that surrounds its location.

Confidence score: A score is a number between 0 and 1 that shows confidence that the item was really identified. The closer the number is to 1, the more confident the model is. Depending on the application, a cut-off level is specified below, which you will reject detections. Here we set the cut-off to 0.5 (meaning a 50 percent probability that the detection is valid) then the detections with a confidence score below 0.5 are ignored. With the selection of right cut-off, the false positives (objects that are wrongly identified or areas of the image that are erroneously identified as objects when they are not) are rejected by the model.

The model MobileNet V1 is capable of recognizing items of 90 classes as it is trained using the COCO dataset. Each item is assigned an ID which is specified in the COCO label map file. As we are interested here just in identifying vehicle and traffic light labels, appropriate IDs are given to the model, and the classification head of the model is modified to recognize only these two items.

C. Recognition of the state of traffic light signals

The output of the model after the traffic light has been recognized is with the bounding box rectangle. The dimensions of this bounding box is obtained to crop the region of the detected traffic light. Color thresholding is applied after converting the RGB image frame to HSV color space to identify red, green or yellow signal state of traffic light. A message of 'Stop' is displayed when red or yellow signal and 'Go' is displayed when the signal is green as shown in the Fig 2.

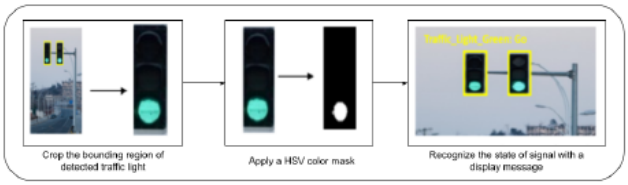

Fig. 2. Recognizing the state of traffic light signals

D. Back-light Recognition

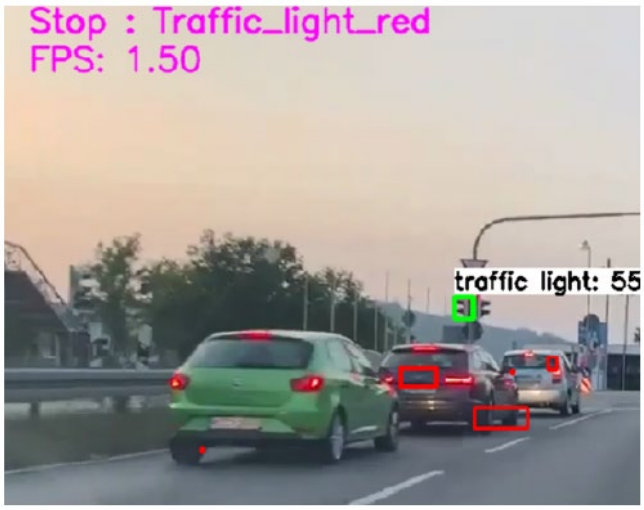

Fig. 3. The bounding boxes of back-light drawn on the video frame are out of place using Approach 1

(i) Approach 1: The bounding box dimensions from the model output is obtained to crop the detected region of the car. Color thresholding is performed on the cropped region after converting the frame to HSV color space to detect the red regions of back light. Contour detection method is applied using OpenCV library to get the boundary region dimensions of the back-light and using the dimensions of the boundary region a bounding box rectangle around the back-light of the car is drawn. With this approach the bounding box drawn on the image frames were out of place rather than being correctly mapped onto the backlight when the video stream was run as shown in Fig. 3. Hence approach 2 as described below was taken into consideration.

Fig. 4. Detection of Back-light with Approach 2

(ii) Approach 2: Area of Interest(ROI) is obtained, which is the bottom half of the frame since most of the vehicles with backlight appear in that region. Color thresholding is done by translating the RGB frame to HSV color space to detect the red color backlight areas. Thereafter contour detection is done to get the border areas of the backlight on which bounding boxes are drawn as shown in Fig. 4.

V. RESULTS

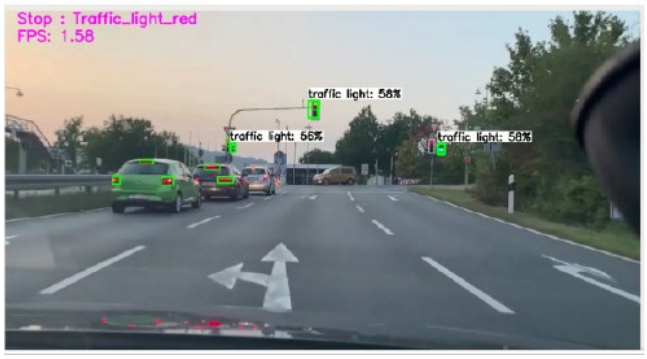

Fig. 5. Output of the proposed algorithm on raspberry pi 3B+ with the quantized model

The TensorFlow model of MobileNetSSD along with the image processing algorithm is evaluated both on Raspberry Pi 3B+ and on a laptop with Intel i7 processor. The comparison is drawn and the performance is evaluated with FPS, inference time and accuracy of the model as shown in Fig. 6. As the deep learning model requires good computing resources which is limited and constrained in Raspberry Pi 3B+ we observe a high inference time and low FPS. Hence a

quantized TensorFlow Lite model is used to provide an optimal performance on Raspberry Pi 3B+ and the output parameters are reported in the Fig. 6. It can be observed from the results that due to the low computing power of Raspberry we obtain low FPS and inference time though we see a betterment of these parameters with the quantized model. The output of the proposed algorithm on a Raspberry Pi 3B+ with the quantized model is shown in Fig. 7.

Parameters / Method	Mobilenet SSD V1 Model on own System (laptop)	Mobilenet SSD V1 Model on Raspberry Pi 3B+	Mobilenet SSD V1 Quantized Model (Optimized Model) on Raspberry Pi 3B+
Model Size	16 MB	16 MB	4 MB
Frames per Second	10-12	0.3-0.5	1.5-2
Accuracy	72 %	72 %	70 %
Inference time (Latency)	80-85 ms	1000-1520 ms	500-520 ms

Fig. 6. Results of model evaluation

The drawback of using color thresholding to detect back-light is if there is any red object apart from back-light within the ROI then a bounding box is drawn around it generating a false positive. As shown in the Fig. 5. an orange bounding rectangle is drawn across the red car instead of red back-light region.

Fig. 7. Drawback of color thresholding for back-light

VI. Conclusion

In this project we proposed a method for recognizing the traffic light and the back-light of the car. The proposed method involves deep learning and image processing. Deep learning used for detecting the car and traffic light and image processing used for recognizing the state of the signal. The algorithm was implemented on the different hardware own laptop (Intel i7 processor,16 RAM) and Raspberry pi 3B+. Further the algorithm was optimized by quantization method and evaluated on the Raspberry Pi 3B+. The result was increased in 3-4 times of FPS, model size was decreased by 4 times and also inference time of model decreased compared to unquantized model but at the cost of 2% loss in accuracy. By performing this experiment, it was observed that there is trade-off between accuracy and inference time. For real-time application of this algorithm FPS and accuracy need to be improved.

VII. Future Scope

The research work can be extended to better the FPS and inference time for the real time recognition of back-light and traffic light on an edge device like Raspberry Pi for which the following points can be considered:

(i) Increased speed and FPS can be achieved with Raspberry Pi 3B+ when combined with the hardware accelerators like Edge TPU and USB Coral.

(ii) Optimization techniques like frame skipping and threading can be implemented on video streams to increase FPS.

(iii) To reduce the false positive rate caused by using color thresholding for back-light recognition we can use transfer learning for the MobileNetSSD model using a custom dataset of back-lights.

(iv) The YOLO model can be tested out as it is said to provide faster inference time with good accuracy.

(v) Real time detection could be possible to achieve with Raspberry Pi 4 along with hardware accelerators.

References

[1] J. F. Krems, M. R. K. Baumann, "Driving and Situation Awareness: A Cognitive Model of Memory-Update Processes," International Conference on Human Centered Design, HCD 2009: Human Centered Design, pp. 986-994.

[2] M. R. Endsley, “Situation Awareness in Future Autonomous Vehicles: Beware of the Unexpected,” Congress of the International Ergonomics Association IEA 2018: Proceedings of the 20th Congress of the International Ergonomics Association (IEA 2018), August 2018, pp 303-309.

[3] Y. Kumar , Y. Jain, “Research Aspects of Expert System”, International Journal of Computing \& Business Research, ISSN (Online): 2229-6166, 2012.

[4] M. R. Endsley, “Toward a Theory of Situation Awareness in Dynamic Systems,” Human Factors The Journal of the Human Factors and Ergonomics Society37(1), March 1995, pp. 32-64.

[5] R. E. T. Jones, E. S. Connors, and M. R. Endsley, “Incorporating the Human Analyst into the Data Fusion Process by Modeling Situation Awareness Using Fuzzy Cognitive Maps,” 12th International Conference on Information Fusion Seattle, WA, USA, July 6-9, 2009, pp. 1265-1271.

[6] A. Niehaus ; R.F. Stengel, “An expert system for automated highway driving,” IEEE Control Systems Magazine (Volume: 11 , Issue: 3 , April 1991), pp. 53-61.

[7] C. L. Forgy , “Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem,” Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213, U.S.A, pp. 17-37.

[8] Julkar Nine, Wolfram Hardt, Shanmugapriyan Manoharan, “Concept of the Comprehension Level of Situation Awareness using an Expert System,” 14th International Forum on Strategic Technology (IFOST), IEEE Computer Society, 978-5-4387-0906-0, October, 2019, pp. 284-288.

[9] Julkar Nine, Shadi Saleh, Owes Khan, Wolfram Hardt, “Traffic Light Sign Recognition for Situation Awareness using Monocular Camera,” Symposium International Symposium on Computer Science, Computer Engineering and Educational Technology(ISCSET), Laubusch, Germany, (2019).