4 9

Traffic Signal Detection using Raspberry pi

Pooja Anandathirtha
Department of Electrical and Computer Engineering
Colorado State University
Fort Collins,USA
poojama@colostate.edu

Rohan Kanigere Umesh
Department of Electrical and Computer Engineering
Colorado State University
Fort Collins,USA
rohan.kanigere.umesh@colostate.edu

***Abstract*—With the advancement in autonomous vehicles detection of traffic signals is necessary. Different deep-learning models have been implemented to achieve this. Training the model on the go is not a feasible solution and for real-time applications, inference time is expected to be as low as possible. Running Deep Learning models on edge devices is a challenge as it requires memory and computational resources. The edge device used for this purpose is Raspberry pi. The deep learning model used to detect traffic signals is R-CNN as it helps set a bounding box to detect the signals. With this model implemented on raspberry pi, we were able to train and achieve the detection of traffic signals.**

***Index Terms*—R-CNN, raspberry pi, ONNX, signal detection**

I. Introduction

In modern times, the use of smart systems has increased. Autonomous cars are one among them. Detecting the signals and analyzing them is one of the main features that are necessary in the real world. While disregarding traffic signals or acting against them might lead to accidents, both self-driving cars, and advanced driver assistance systems can benefit from an accurate traffic signal-detecting module to increase road safety. The speed of a self-driving car decreases as it approaches an intersection in accordance with safety regulations. Thus, it is crucial that traffic light detection and recognition be accurate. We can achieve this by using inputs from the sensors for image processing and then feeding them to a machine-learning model to process it. Due to their comparatively low cost, vision-based traffic signal detection techniques are growing in popularity as a solution. But in actual traffic situations, a number of problems still exist, including complicated backgrounds, varied traffic lights, different kinds of illuminations, motion blur, tail lights, and color shifting. These problems make it more difficult to recognize traffic lights. Adding to one of the many solutions, we propose using a deep learning model to achieve it. We hereby use R-CNN (Region-Based Convolutional Neural Network) for training the signal dataset and then implement the results on the raspberry pi to execute the same process in real-time.

Training the dataset using R-CNN (Region-Based Convolutional Neural Network) helps in putting bounding boxes around the desired region. The weights are further saved and converted into.onnx format, which is faster than TensorFlow. These weights are then trained with a simple script and run on ONNXRuntime which is installed on the Raspberry pi.

The results obtained while running the model on a GPU are different from that when it is implemented on an edge device. We obtained a result with an accuracy of approximately 78 while running on the GPU, while the accuracy was approximately 70 on Raspberry pi.

II. Related Work

We conducted a survey and did some research to explore different approaches toward Traffic Lights Detection. Methods for Traffic Light detection are usually classified as image processing-based, machine learning based, or map-based techniques[1]. Zhenwei Shi highlighted the Adaptive Background Suppression filters method for Traffic Light detection under different illumination conditions(image processing)[2]. Though image processing is straight and less complex, it undergoes phases such as thresholding and filtering, and slight deviations from standards in these phases result in ambiguous outcomes which is undesirable in sensitive cases of Traffic Lights detection. To address this limitation, machine learning-based methods and algorithms are used in combination with ample processing techniques to avoid misleading directions. Ida Syafiza Binti Md Isa1 , and Choy Ja Yeong used CNN model classification to detect the signals and implement the same.[3] Here they used CNN algorithm for the traffic sign smart detection system, which focuses on color-based detection, shape-based detection, and sign validation. But, the color of the traffic sign and the lack of light during the night may affect the performance of the CCN.

To attain better results in this learning-based model, one must collect a variety of large training datasets and train the model for a significant amount of time. Considering this issue, we have used R-CNN for efficient image classification and object detection and to achieve better outcomes.

III. Data

A. Bosch Small Traffic Lights Dataset

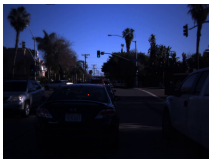
The dataset we have used to test the model is Bosch Small Traffic Lights Dataset. This is a precise dataset for vision-based traffic light recognition. This dataset performs object detection testing approaches including tiny objects in big images. This dataset includes images that cover a variety of road scenarios like busy streets inside cities, Suburban multilane roads traffic, dense stop-and-go traffic, construction works, rainy days, multiple traffic lights, etc Data description

This dataset contains 13427 camera images at a resolution of 1280x720 pixels and contains about 24000 annotated traffic lights. The annotations include bounding boxes of traffic lights as well as the current state (active light) of each traffic light. The camera images are provided as raw 12-bit HDR images taken with a red-clear-clear-blue filter and as reconstructed 8-bit RGB color images. The RGB images are provided for debugging and can also be used for training. However, the RGB conversion process has some drawbacks. Some of the converted images may contain artifacts and the color distribution may seem unusual.

B. Dataset specifications:

1) Training set: : 5093 images
Annotated about every 2 seconds
10756 annotated traffic lights
Median traffic lights width: 8.6 pixels
15 different labels
170 lights are partially occluded
2) : Test set: 8334 consecutive images
Annotated at about 15 fps
13486 annotated traffic lights
Median traffic light width: 8.5 pixels
4 labels (red, yellow, green, off)
2088 lights are partially occluded

For the test set, every frame is annotated and temporal information was used to improve the label accuracy. The test set was recorded independently from the training set but within the same region. The data set was created to prototype traffic light detection approaches, it is not intended to cover all cases and is not to be used for production. Examples:



METHODS

The purpose of our project is to detect traffic signals and traffic lights that require a bounding box. This is easier to implement using R-CNN. R-CNN is a deep convolutional

Fig. 1. Example images.

network that was developed by researchers at UC Berkeley. Here we combine region proposals with CNNs. Hence it is called R-CNN(Regional Convolutional Neural Networks). It is used for detecting different types of objects in images. It extracts features based on convolutional networks. The R-

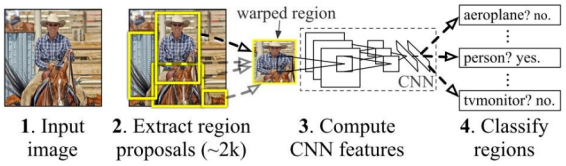


Fig. 2. Object detection system overview.

CNN consists of three main modules for object detection. The first module generates region proposals. The second module extracts a feature vector of length from each region proposal and the third module uses a pre-trained SVM algorithm to classify the region proposal to either the background or one of the object classes. From each region, a 4096-dimension feature vector is extracted and computed by forward propagating a mean subtracted 227 ×227 RGB images through five convolutional layers and two fully connected layers. R-CNN is agnostic to the particular region proposal method, we use selective search to enable a controlled comparison with prior detection work. During test time, the R-CNN method generates around 2000 category-independent region proposals for the input image, extracts a fixed-length feature vector from each proposal using a CNN, and then classifies each region with category-specific linear SVMs. We use a simple technique (affine image warping) to compute a fixed-size CNN input from each region proposal, regardless of the region's shape. The shared CNN parameters and the low dimensional feature vectors computed by CNN make the detection efficient.

Further, the thus trained model is given as an input to Raspberry pi using the ONNX inference engine which is installed on the raspberry pi. To achieve this, we save the weights and then convert them to .onnx format. Conversion is necessary as this has to be run on the ONNX inference engine. ONNX inference engine accelerates machine learning across a wide range of operating systems. The real-time input is taken from PiCam.

Initial consideration of using google's coral dev board was not a cost-effective solution compared to Raspberry pi. Raspberry pi is also a compact board with high processing power and many interfaces.

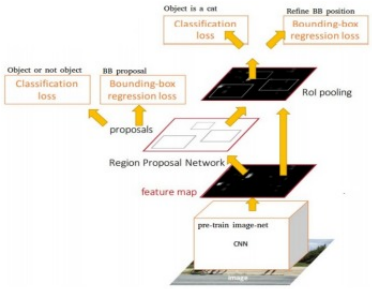


Fig. 3. RCNN.

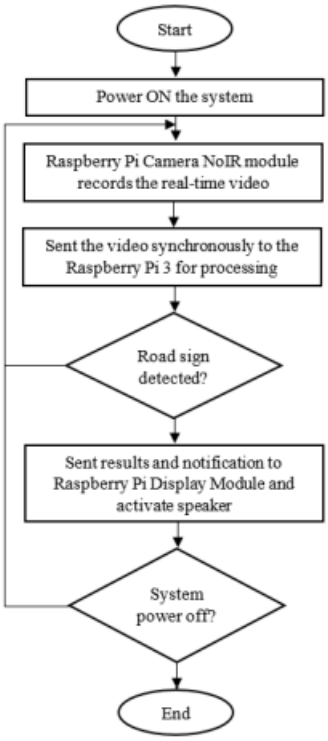


Fig. 4. Flow of the working.

We can understand the flow of the project in Fig 4. This flow chart implies that once the power of the system is ON, the Pi Cam starts capturing real-time video. This video is sent to raspberry pi for processing asynchronously. Based on the training given to the model, it detects the signal and the lights. Once the signal is detected the algorithm analyses it and categorizes it to be red, yellow, or green. If the camera does not detect any traffic signal, it keeps monitoring until it finds one. This goes on and on until the power supply is cut off.

IV. Experiments

Traffic signal detection with raspberry pi implements R-CNN as a deep learning model to detect the traffic signals and identify the signal lights. The experiment implements the Bosh Small Traffic Lights Dataset which contains a mixture of images. These include images with traffic signals, roads, cars, mountains, and more, helping in having diversity. Feature extraction is a major part of signal recognition. The bounding box is applied to the signal only after the edge detection.

The web camera, which is attached to the Raspberry Pi through a USB 2 connector, is used to acquire the video. Data is therefore collected through a serial connection which is asynchronous.

This project also implements ONNX runtime on raspberry pi which is afresh in the community. The installation of ONNX runtime is time-consuming but offers a better output qualitatively. ONNX runtime is installed on the OS for raspberry pi which is Raspbian.

To get the video input we use vnc view to record the obtained results on an electronic device which could be a laptop, mobile, or tablet, any of which has vnc viewer installed on it.

The experiment starts with training the RCNN model. The data fed to this model need not need cleaning as it was previously cleaned. This data produces a good quality result as it has a number of images with diversity, including, images of nature, roads, vehicles, people, signal lights, traffic signs, and much more.

The images are trained to identify the traffic signals and the color of the light on the traffic signal. Bounding boxes are added to the traffic signals and the light. During the test, the accuracy of the output with green light varied from 72 to 76 percent while the red light varied from 74 to 78 percent.

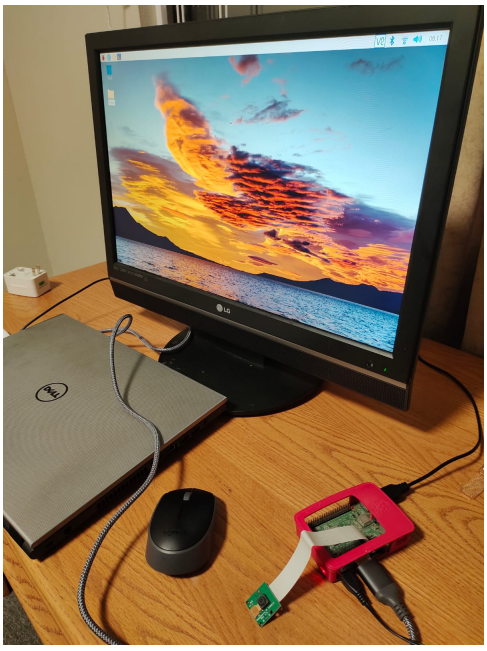


Fig. 5. Experimental setup.

A. *Experimental setup of this project.*

The setup of the project is shown in Fig 5. Raspberry Pi 3 processor is the main controlling unit in the system. A

Raspberry Pi Cam is interfaced with the Raspberry Pi 3 which continuously captures the video frames of the traffic signal during the motion of a vehicle. According to the code in the Raspbian os, the captured frames are processed and displayed on the monitor.

Once the model was trained. The .onnx file which contained the weights was then given as input to the script on the ONNX inference engine.

The script for ONNX runtime inference engine converts the program and the weights after training it to a format compatible with the Rasberry pi to process and hence helps in identifying the signal and comprehending it.

Installation of ONNX is a lengthy procedure and should appear like the following.

Step 24/25 : RUN ls -l /code/onnxruntime/build/Linux

Running in 702f0dabef0c -rwxr-xr-x 1 root root 20660 Sep 1 21:26

/code/onnxruntime/build/Linux/MinSizeRel/libonnxruntime.so -¿ libonnxruntime.so.1.4.0

Step 25/25 : RUN ls -l

/code/onnxruntime/build/Linux

—¿ Running in bebab6345297 -rw-r–r– 1 root root 2793630 Sep 1 21:34 /code/onnxruntime/build/Linux/MinSizeRel/dist/onnxruntime-1.4.0-cp38-cp38-linux$_armv7l.whlRemovingintermediatecontainerbebab6345297 --> ceb5c1feac97$

Successfully built ceb5c1feac97 Successfully tagged onnxruntime-arm32v7:latest

pi@raspberrypi: /onnx-build sudo docker images

REPOSITORY TAG IMAGE ID CREATED SIZE onnxruntime-arm32v7 latest ceb5c1feac97 2 minutes ago 4.71GB

¡none¿ ¡none¿ 5daf4a498aec 16 hours ago 4.18GB

¡none¿ ¡none¿ a8d640118321 4 days ago 4.18GB

¡none¿ ¡none¿ 87ac36b46669 7 days ago 4.18GB

¡none¿ ¡none¿ 3bb41355adb3 8 days ago 4.18GB

balenalib/raspberrypi3-python latest-stretch-build fa379fc77172 2 weeks ago 676MB

hello-world latest 851163c78e4a 8 months ago 4.85kB

pi@raspberrypi: /onnx-build docker create -ti –name onnxruntime$_temponnxruntime - arm32v7bash$

pi@raspberrypi: /onnx-build docker cp

pi@raspberrypi: /onnx-build ls

build$_logDockerfile.arm32v7onnxruntime - 1.4.0 - cp38 - cp38 - linux_armv7l.whl$

pi@raspberrypi: /onnx-build docker rm -fv onnxruntime$_temponnxruntime_temp$

pi@raspberrypi: /onnx-build ls

build$_logDockerfile.arm32v7onnxruntime - 1.4.0 - cp38 - cp38 - linux_armv7l.whl$

pi@raspberrypi: /onnx-build

Common failures can include fetching incompatible dataset, and selecting unusable deep learning models. The problem faced during the experiment was including the syntax of TensorFlow 1 instead of TensorFlow 2. Runtime issues included a cloudy environment for the video and the detection of car break lights. This was overcome by changing the optimizer and retraining the model. Installing and compiling files for ONNX could be a hectic task which was then overcome by referring to videos of installation and Stackoverflow.

The results ranged between 70 to 78 percent while using the input from Pi Cam.

V. Conclusion

This project implements a real-time traffic signal recognition system using the Raspberry Pi 3 processor. The R-CNN machine learning algorithm is used to train the traffic signals and detect the traffic lights through the real-time recording video using a Raspberry Pi camera. The project tested considering the condition of the traffic signals such as blurry, shady, rainy, etc. The performance of the developed system has been evaluated in terms of accuracy, delay, and reliability. The results show that the average accuracy of detecting and identifying traffic light images is above 70Considering the future scope, this project can be optimized and also be substituted with Fast R-CNN as it is faster than R CNN and you don't need to feed the convolutional neural network with 2000 region proposals continuously, instead a feature map is produced from the convolution operation, which is only performed once per image. Also, it can be equipped to respond to commands from law enforcement or highway safety employees.

References

Please number citations consecutively within brackets [**?**]. The sentence punctuation follows the bracket [**?**]. Refer simply to the reference number, as in [**?**]—do not use "Ref. [**?**]" or "reference [**?**]" except at the beginning of a sentence: "Reference [**?**] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [**?**]. Papers that have been accepted for publication should be cited as "in press" [**?**]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [**?**].

References

[1] . John, K. Yoneda, Z. Liu, Senior Member, IEEE, and S. Mita, "Saliency Map Generation by the Convolutional Neural Network for Real-time Traffic Light Detection using Template Matching" in IEEE Transactions on Computational Imaging DOI 10.1109/TCI.2015.2480006.

[2] henwei Shi, Zhengxia Zou, and Changshui Zhang, "Real-Time Traffic Light Detection with Adaptive Background Suppression Filter" in IEEE Transactions on Intelligent Transport Systems, DOI 10.1109/TITS.2015.2481459

[3] Real-time traffic sign detection and recognition using Raspberry Pi" Ida Syafiza Binti Md Isa1 , Choy Ja Yeong , Nur Latif Azyze bin Mohd Shaari Azyze

[4] ai Huu-Phuong Tran, Cuong Cao Pham and Tien Phuoc Nguyen, ”Real-Time Traffic Light Detection Using Color Density”, 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), pp. 1-4, Oct. 2016. Show in Context View Article

[5] . H. Widyantoro Stephen, ”Circle and arrow traffic light recognition”, IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), pp. 34-38, Nov. 2017.

[6] runyawee, W. Leelapatra and T. Satiennam, ”Traffic Light Color Identification for Automatic Traffic Light Violation Detection System”, International Conference on Engineering Applied Sciences and Technology (ICEAST), pp. 1-4, July, 2018.

[7] . Shin, S. Roh and K. Sohn, ”Image-Based Learning to Measure the Stopped Delay in an Approach of a Signalized Intersection,” in IEEE Access, vol. 7, pp. 169888-169898, 2019.

[8] Applied Deep Learning - Part 4: Convolutional Neural Networks” avalable at: https://towardsdatascience.com/applied-deep-learning- part-4-convolutional-neural-networks-584bc134c1e2

[9] ast R-CNN by Ross Girshick, Microsoft Research