International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 07 | July 2021 www.irjet.net p-ISSN: 2395-0072

Automated Traffic Light System using Raspberry Pi and Convolution Neural Network

Siddaraj M G[1], Syeda Umeela[2], Pravalika Gandhi[2] , Saniya kousar[2], Shrivathsa S K[2]

[1]Professor, Dept. of IS&E Engineering, MIT Mysore, Karnataka India
[2]Dept. of IS&E Engineering, MIT Mysore, Karnataka India

Abstract – *Management of traffic is an ever-growing concern and manual management of traffic especially in metropolitan cities is strenuous task. In this paper, we present an automated traffic light system based on image processing and machine learning techniques to automatically identify the number of vehicles in each lane to set the green signal time optimally to avoid large waiting time and also clear congestion at faster rate. The YOLO darknet weights and labels are used with categorized classes to identify and detect number of vehicles in the frame of the captured image. The entire system is implemented using Raspberry pi 3B+ integrated with rotatable webcam. The results show that this system is reliable as it produces 100% accuracy in producing the exact vehicle count and setting of green time optimally to have a wait time no longer than 90 seconds.*

Key Words*: *You Only Look Once (YOLO), Green Time, Convolution Neural Networks (CNN), Raspberry Pi, Wait time, Vehicle Detection.

1.INTRODUCTION

Road traffic is one of the biggest concerns in almost every country in the world. With roads designed to handle a certain amount of traffic, it is almost impossible to maintain traffic to stay at the same rate with a growing population and economy. Some alternate solutions to control the traffic is either through expanding the roads or installing new roadways which can be time-consuming. Most cities employ a density-based traffic signal system [1], which measures the number of vehicles in each lane before deciding the green time**.** However, the density is measured through sensors and does not estimate the exact count of vehicles but gives only a rough estimate. With the advancement in technology concerning image processing and machine learning, it is possible to capture the image of the lane and estimate the exact number of vehicles present in the lane to determine the green time for a particular lane in the intersection. The regulation of traffic lights in the intersection has to be done based on traffic congestion which can be measured through advanced image processing techniques.

In this paper, we present a YOLO Framework-based vehicle count estimation and green time determination for each lane in the intersection. The YOLO or You only look once is an algorithm that uses convolution neural networks for object detection in real-time. They apply a single neural network to the full image and divide the image into regions and predict the bounding boxes which probabilities for each region. These bounding boxes are weighted by the predicted probabilities. YOLO has high accuracy and also its multiple bounding boxes and class probabilities for these boxes. The hardware setup for this project is developed using the raspberry pi and a webcam to capture real-time images. Once the captured image is subjected to YOLO detection, the number of vehicles in the image is known. Based on the number of vehicles the green time is decided and is deployed into the LED drivers acting as traffic signal systems for the particular lane. The green time is decided based on the burst technique. This technique allows the congestion to be under control and the maximum wait time of any lane is not more than 90 seconds.

This paper is organized as follows section 1 describes the existing system and the proposed system, the second section explains the algorithm and the working of the YOLO framework, the third section describes the architecture of the project developed using raspberry pi. The fourth section illustrates the results of the developed system in terms of speed accuracy and other performance metrics.

1.1 Existing System

The Existing system uses PIR based technique for identifying the vehicle density [2]. The Raspberry Pi 2 microcontroller handles the LED lighting of the traffic light according to the times previously set with Python code. Depending on the output of the Pir motion detector sensor which, when detecting the infrared radiation change through its line of sight, sends a HIGH to the microcontroller. On the PCB board all the red-colored LEDs are soldered in parallel. They will flash when the traffic light changes from red to green. The common cathode RGB LED display, soldered in parallel on the PCB board, consists of 4 terminals. The longest terminal corresponds to the common cathode while the other 3 terminals correspond to the emission of red, green light and blue respectively. The microcontroller will send PWM signals to each of these terminals and depending on the work cycle that is applied to the PWM signal of each terminal, the RGB LED of a will light up with the correct color. In [3], the authors implement the image processing-based traffic light control using the Arduino UNO and MATLAB environment. The Arduino is used to send data to the first order now for and then the webcams are selected in sequence these webcams are connected to the first webcam on the system. The colored image acquired from the webcam is converted into binary image, from which the region of interest is selected and used as a mask. The morphological operation is

performed to remove the objects in the image depending on the size and shape of the structuring element. Appropriate MATLAB functions are used to calculate traffic density based on the morphological operations performed in the image. In this system 4 different Arduinos along with 4 different webcams are used which next system more costly and increase the processing time.

1.2 Proposed System

We propose an IOT based system that uses Raspberry Pi 3 and YOLO framework to determine the green time for each lane based on images captured at the lanes. The system is designed considering the parameters like delay, accuracy, performance, cost, and reliability. The system uses a single webcam to capture images of the vehicle in the lane that can be rotated after green time is concluded for a particular lane and has to be estimated for the next lane. The master controller unit which is the raspberry pi is powerful enough to run ML-based algorithms that can be subjected to the input images obtained from the webcam. The entire system works in real-time and can have a very minimal delay. The YOLO Framework is used to determine the number of vehicles in each lane from the obtained images and the green time is scheduled accordingly concerning minimum and maximum allotted green time for each lane. In general considerations, it is assumed that each vehicle takes 3 seconds on average to exit the lane in the intersection. The maximum time for any lane a set for 30 seconds, and a minimum of five seconds of green time is given to each lane irrespective of the number of vehicles. If the number of vehicles is large and the green time exceeds 30 seconds for the number of vehicles present in the lane, cycles may have to wait for the next cycle, no longer than 90 seconds. Python Language is used for the design of the system. The hardware setup consists of a traffic signal prototype designed from red, green, and yellow LEDs controlled from the Raspberry Pi. Fig 1 shows the proposed system architecture.

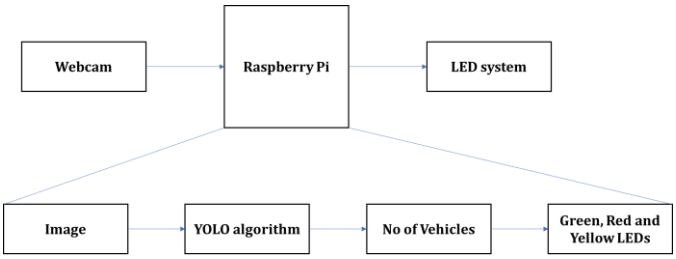

Fig 1: Proposed System

2. YOLO Framework and CNN

It is one of the object detection frameworks chosen for its excessively fast and accurate detection technique. Object detection is one of the classical problems in computer vision that concerns the recognition of position and the type of object present inside the image. Some object detection algorithms do not specify the location on the images or may not work on images containing more than one object. Yolo on the other hand is clever fast and uses Convolution Neural Networks (CNN) for object detection in real time. Its ability to run in real time with high speed and high accuracy, one of the most preferred algorithms for object detection. The single CNN predict the multiple bounding boxes simultaneously along with the class probabilities. Detection is made at three scales precisely given by the term sampling of the dimensions of the input image 32 16 and 8 respectively.

CNN comes under the Deep Learning which is a subdomain of Machine learning. It is applied in most of the image classification process. Image classification involves extraction of features from the image to observe patterns that is available in the data set. Use of CNN eliminates the cost of computation. Convolution is point wise multiplication of two functions that produces the third function. Out of the two functions, one is the image in matrix format and the other is the image filter. The resulting matrix can be termed as a feature map. Fig 2 shows the CNN principle.

The steps involved in CNN are as follows

1. Choose or create a dataset - Image data set can be chosen from available formats like weight are Caffe model files or can be created from custom images collected which contains all possible types of images concerning for a particular application or detection. For example, if CNN is used for face mask detection, the dataset shall contain the images of all kinds of faces with all kinds of masks available in the market.

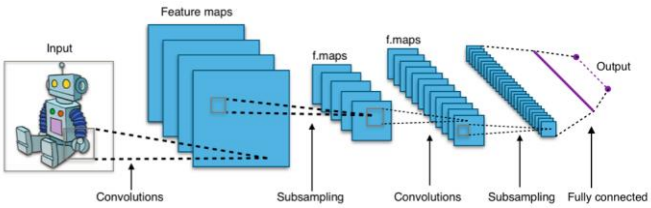

Fig 2: Steps involved in CNN

2. Create Training Data - Creation of training data involves labelling the images that contains the pixel values and the index search with the image in categories list. The images are appended to the array of training by normalizing the categorical data through assignment of corresponding labels. This training data is split into two categories that is test and train data. Conventionally 20 percent of data is used for testing and the other 80% for training.

3. Apply on images – The trained set is compared and contrasted w.r.t the input image for classification. The image is first subsampled into maps and the convolution is applied to these maps to generate the feature maps. The feature map is further subsampled to obtain a fully connected image set contained convolved subsets on which the comparison or classification is carried out.

YOLO object detection is performed either through real time or on custom image or video. Nearly 80 different classes of objects can be determined using the YOLO framework. Out of the available classes, our system uses the pretrained YOLO config and weights to determine few classes namely car, truck, bus, train and motorbike for counting number of vehicles in each lane of the intersection. The detected vehicle count in the image is used to determine the green time as described in Section 3. However, it is to be noted that the algorithm is very powerful and requires a faster computation device with high processing power or can cause delay in the detection process. Also, the webcam should have enough resolution to capture clear images.

3. SYSTEM ARCHITECTURE

The UML diagram for the system architecture is shown in fig 3. The architecture shows the implementation of the system using the Raspberry Pi.

Raspberry Pi 3B+:
The Raspberry pi is a credit card sized computer which runs on Raspbian Operating System. It is a 1.4 GHz 64-bit quadcore processor with 802.11n Wi-Fi, Bluetooth and USB boot capabilities. It has a 300Mbit/s ethernet and SD card for the OS. Python3 is used to program the LEDs based on input from the camera.

OpenCV: Open CV is an open-source computer vision machine learning software library built to provide a common infrastructure of computer vision applications to accelerate the use of perception among the commercial products. It has 2500 optimise algorithms including a comprehensive set of classic and state-of-art CVs. Most of these are used to detect faces, identify objects, classify human actions vehicle detection 3D model extraction stereo camera extractions and many more. It is also used in enhancing upscaling downscaling interpolating and other image processing applications in a simple and effective manner. It has c++, Java and MATLAB interface and supports multi-paradigm programming for higher level project synthesis. In this project cv2 is trained with yolo weights identify the region of interest in an image and extract if an animal is found. Since CV support libraries are strong enough, multiple presence of vehicles can be detected and identified.

NumPy: NumPy is another powerful library for python programming language which supports multidimensional array and matrices with a collection of high-level mathematical functions to perform operations on the arrays. It targets the reference implementation which is a non-optimising bytecode interpreter. It analyses the slowness of the compiler by providing the multidimensional array and functions which could operate efficiently.

RPi GPIO: This package provides a class to control the GPIO on a Raspberry Pi. It allows the python to work as an embedded language for the raspberry Pi. This library is used to control the status voltage input and outputs of all the pins in raspberry pi programmatically. It is also so helpful in integrating the peripherals and external devices connected with the raspberry pi.

Flask: Flask is a micro web framework written in Python. It is classified as a microframework because it

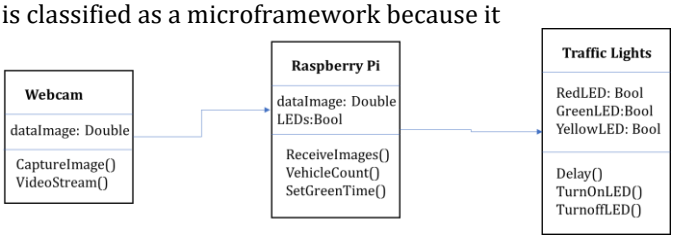

Fig 3: UML diagram for architecture

does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. A Web-Application Framework or Web Framework is the collection of modules and libraries that helps the developer to write applications without writing the low-level codes such as protocols, thread management, etc. Flask is based on WSGI (Web Server Gateway Interface) toolkit and Jinja2 template engine. In this project the flask is used to fetch the data into the URL where the interface is available and also read the user commands from the same URL to perform operations accordingly.

Webcam: A high quality webcam is used to capture the images at each lane. The images must be clear enough to identify the number of vehicles to set the green time. An image is taken as input into the python code to apply ML from the YOLO framework.

LED traffic system: Red, Green and yellow LEDs are used to create a prototype of the traffic system for 4 different lanes.

Green Time Determination: In the first step the image is subjected to CNN based classification to recognise the number of vehicles present in the image. The vehicle count is obtained for each frame of image captured from the webcam based on this count, the green time for each plane is set. The green time is set as three times the number of vehicles obtained from the frame. As illustrated above each vehicle is assumed to take 3 s on an average to exit the line. But the green time is set to minimum of five seconds even if no vehicles are found and the maximum green time is set to 30 seconds even if the number of vehicles is huge. The important thing noted here is that the image is captured from the lane when the light hits yellow. So that the next green signal can be held for the number of seconds determined in the previous step using the time.sleep function in python. The detailed flowchart is shown in fig 4.

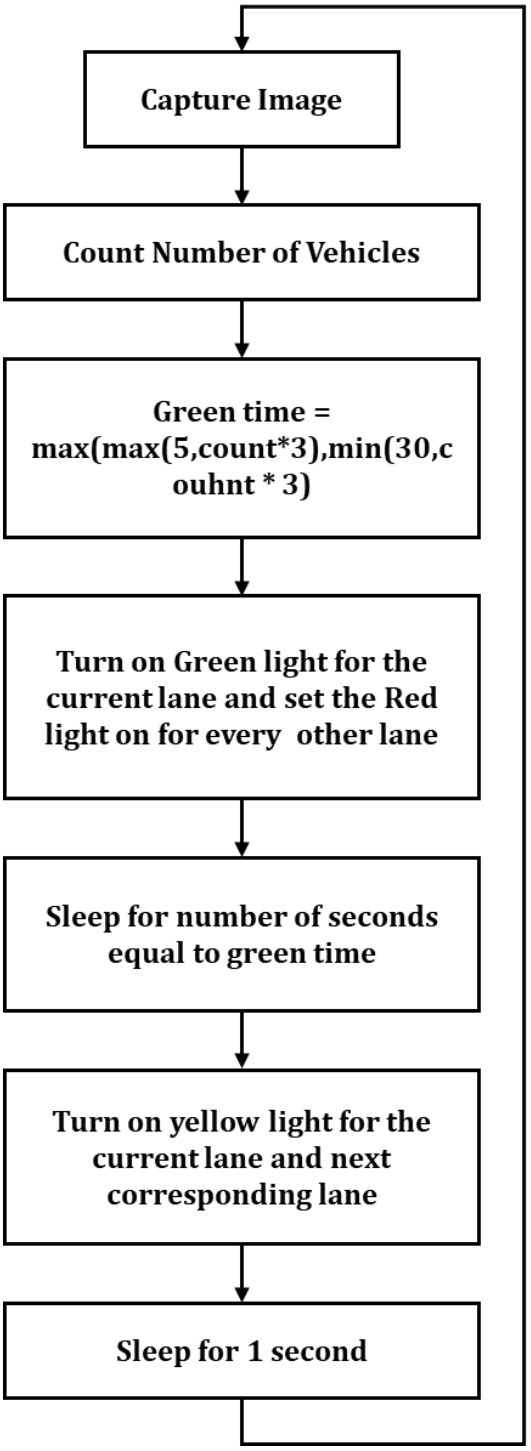

Fig 4: Proposed System Flowchart

As we can see from the flowchart, the green time is held by using the sleep function in python. The flowchart for the Vehicle count detection is illustrated in the fig 5. The detection uses CNN algorithm with YOLO weights.

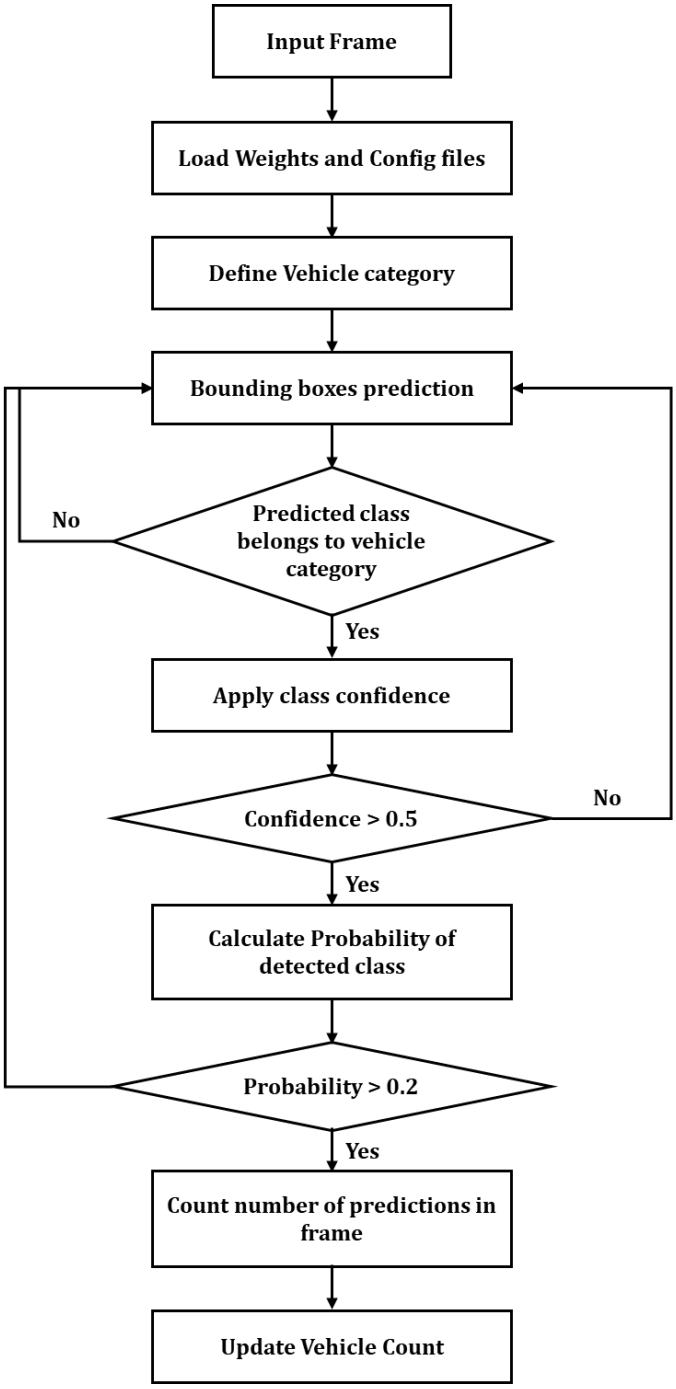

Fig 5: Flowchart for Vehicle count determination

4. IMPLEMENTATION RESULTS AND ANALYSIS

The implementation is carried out in real time for capturing the number of vehicles in the lane and the performance measures are tabulated in table 1. For each input image the detections are shown in fig 6.

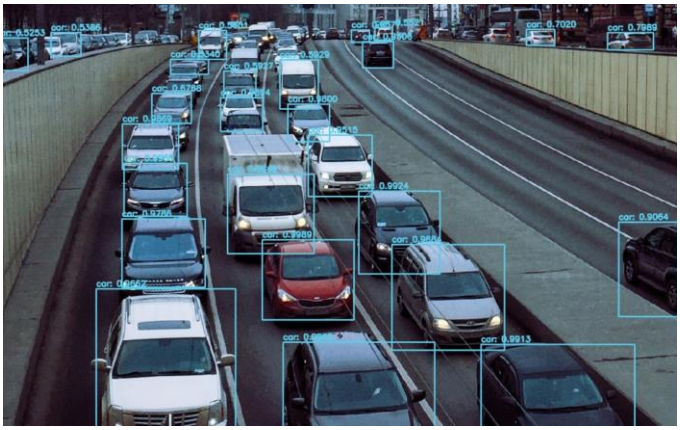

Fig 6: Vehicle Detection Results in CV – trial 1

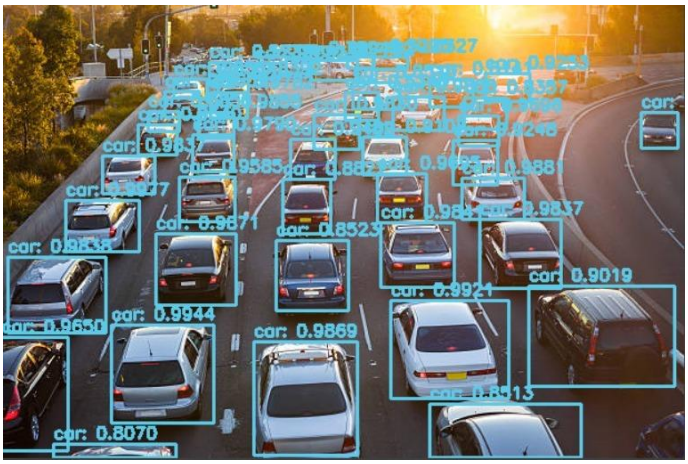

Fig 7: Vehicle Detection Results in CV – trial 2

The further trials and green time are tabulated below.

Lane	No of vehicles	Predicted No of Vehicles	Green Time	Wait time (s)	Accuracy (%)
1	26	26	30	0	100
2	28	28	30	30	100
3	14	14	30	60	100
4	2	2	6	66	100

Table 1: Trial 1 with dense traffic congestion

Lane	No of vehicles	Predicted No of Vehicles	Green Time	Wait time (s)	Accuracy (%)
1	6	6	18	0	100
2	4	4	12	18	100
3	7	7	21	30	100
4	8	8	24	51	100

Table 2: Trial 2 with moderate traffic congestion

5. CONCLUSION

An automatic traffic light system was developed using IOT and Machine learning algorithm to determine the green traffic light duration based on traffic density. The image was captured using webcam for each lane and the image was subjected to YOLO detection algorithm to identify the number of vehicles based on which the green time was set. The detection was performed using the CNN with pretrained YOLO coco weights. The green time was set optimally to avoid longer waiting times. The entire system was developed with Raspberry pi as master controller and LEDs for traffic signal systems. The prototype was run in real time and the results show 100% accuracy in determining the number of vehicles and setting optimized green time.

6. REFERENCES

[1] Gul Shahzad; Heekwon Yang; Arbab Waheed Ahmad; Chankil Lee, "Energy-Efficient Intelligent Street Lighting System Using Traffic-Adaptive Control", IEEE 2016, Volume: 16

[2] Nicole Díaz, Jorge Guerra, Juan Nicola, "Smart Traffic Light Control System", IEEE 2018.

[3] Khushi, "Smart control of Traffic Light System using Image Processing", International Conference on Current Trends in Computer, Electrical, Electronics and Communication (ICCTCEEC-2017)

[4] J. K. and A. Desai, "IoT: Networking Technologies and Research Challenges", International Journal of Computer Applications, vol. 154, no. 7, pp. 1-6, 2016.

[5] " Application of Raspberry Pi and PIR Sensor for Monitoring of Smart Surveillance System", International Journal of Science and Research (IJSR), vol. 5, no. 2, pp. 736-737, 2016.

[6] K. Choi, "Visible Light Communication with Color and Brightness Control of RGB LEDs", ETRI Journal, vol. 35, no. 5, pp. 927-930, 2013.

[7] P. N R, "Smart pi cam based Internet of things for motion detection using Raspberry pi", International Journal of Engineering and Computer Science, 2016.