Abstract

This paper presents a traffic light detection system using spiking neural networks with an intent to provide faster computation and accurately predict the colors even in traffic lights rotated at an angle. The system development is divided into two stages preprocessing and detection stages respectively. In the preprocessing stage, the light(s) is extracted from the real time image and the light(s) dimensions are shrunken which are passed into the detection stage. In the following detection stage, the traffic light color(s) is identified using spiking neural networks to make the final decision of whether to stop or go.

Index Terms—Traffic Light Detection, Spiking Neural Networks, Leaky Fire and Integrate

---

1. Introduction

Autonomous driving is a growing research topic recently. In this, the aim is to make a computer system which should be able to function completely like a human driver. However, to accomplish this, at very basic it is vital that the system recognizes the traffic lights and the color which is active i.e red, green or yellow and act accordingly. Implementing this basic traffic rule in autonomous driving with higher accuracy is very crucial to avoid car accidents on the roads and highways.

There are several techniques that can be used to recognize the traffic light and the prominent tactic is to use Convolutional Neural Networks [1]. Moreover, the approaches have used CNN to train the model on just vertical and horizontal images. However, the issue with that is it does not take in the cases where traffic lights are at some angle and not necessarily horizontal and vertical. This is definitely possible in cases of windy weather conditions. Another issue is the image size used, with larger pixels the network may take long to train. Finally, the same can be achieved with the faster architecture.

The faster architecture we propose is the use of the Spiking neural network (SNN). Spiking neural networks (SNNs) are inspired by the brain in order to imitate its functionality so that the computer can learn and process as efficiently as the brain. The SNN approach uses a binary output (spike/no spike) instead of the continuous output of traditional ANNs. In SNN, a neuron model that fires at the moment of threshold crossing is called a spiking neuron model. In this project, we are using the leaky-integrate fire neuron model.

In this project, we have used the YOLOv3 network to extract the traffic light from the scene or real time images of the road from the car view. In the YOLOv3 network, we used the pretrained weights to extract traffic light, and once we extract the traffic light we detect the traffic light color using the SNN architecture. Once the color of the light is determined, the network can make decisions based on it.

---

2. Background (or Theory)

Spiking Neural Networks are the 3rd generation neural neuron networks that pass the information from one neuron to another in the form of spikes. It is proved that the neurons that convey information by Image Processing with Spiking Neuron Networks are computationally

more powerful than the neurons with sigmoidal activation functions ("10, 10.1.1.3") [2]. Compared to the first and second generation of Neuron networks, Spiking neural networks have higher accuracy. There are two types of encoding - rate based and time-based encoding.Rate encoding has the advantage that an analogy can be drawn between the activation function of a spiking neuron (in terms of the relation of the input rates to the output rate) and that of a formal neuron, leading to the opportunity to map a trained formal network to an identical spiking one [3] . We are using rate-based encoding here where the information is passed from one neuron to another based on the number of spikes generated by the input neuron. There are three common models used in the Spiking Neural network - Izikiweich model, Leaky integrate and Fire, and Hodgkin Huxley model.

We defined “leaky integrate-and-fire model” as a python class in our implementation. We have set all the constants first in its constructor based upon . The only variable here is voltage which changes according to the voltage decay at every given time.

$$C_m \frac{dV}{dt} = I(t) - \frac{V_m(t)}{R_m}$$

In the above equation capacitance (C_m), current (I) and resistance (R_m) are constants which are initialized in the constructor. Moreover, if voltage reaches the threshold value then the model initializes a spike and returns back to the resting state which is set to 0 [4].

To train the model, it is vital to strengthen the connection between those neurons which spiked to result in correct classification. So, if neuron A and B spikes to give the correct output then the connection between A and B should be strengthened. If the output generated is wrong, then the weight should decrease. This connection between neuron A and B is called the synaptic weight. To update the weights, we use Hebb’s rule which states that the neurons that wire together, fire together. The equation is as follows:

$$\frac{d}{dt} w_{ij} = \alpha_2^{corr} v_j^{pre} v_i^{post}$$

So, the change in weight is determined by the learning rate α which basically defines the rate at which the rate should increase when both the neurons spike. Which can be derived by experimenting with the value. V_j is the firing rate of the pre synaptic neuron and V_i is the expected fire rate of the postsynaptic neuron [4].

---

3. Experimental/Modeling Design

We divided the process of developing the model into three parts: preprocessing data, encoding scheme and rate-based learning respectively.

I. Preprocessing Data

We used the simulated traffic light database to train our model. However before training, we took all the images from the dataset and rotated them at 90,180,270 degrees. In this way we ensure that we have images available at various angles and we do not have any way to predict the location of the colors on the light.

Next, before passing the image to detection. We convert the images n x n to 10 x 10 and this matrix will be passed to a learning model to predict the color. Since we have 10 x10 colored images the number of input neurons will be 300 neurons and the output neurons will be 3 each classifying either red, green or blue color.

II. Encoding Scheme

The input image is converted to a RGB matrix and R and G values of the images are extracted. Now, each of these pixel values will be between 0 and 1 (in case of a png image). So, these values are passed in as the current to the lif neurons. Based on this input, the lif neurons will generate some number of spikes (scaled between 0 and 1. 1 been the max current). This way the R and G were converted to the number of spikes. So, rate based encoding is used to train and update the weights of the model.

III. Rate-Based Learning:

We are using the Hebb's learning rule to train the model. Based on the postsynaptic current and the pre-synaptic current the change in weight is calculated based on equation (2). If vj or vi is zero meaning that the neuron remains inactive then there will not be any changes in the weights connecting the two neurons. Here we have taken the correlation constant as 0.7 to ensure that the weights are updated properly with increasing validation accuracy. The algorithm of how the weights are changed is given below. We have different threshold values for input and output neurons. Current to input neurons ranges from 0 to 1 after dividing the RGB values with 255.

```
ALGORITHM
    For every pixel p:
        pre_syn_current = p
        pre_rate = count_spikes(lif_neuron(pre_syn_current)) / max_spikes
        for i in range(3): [0  Red, 1 Green, 2  Yellow]
            post_syn_current = weights[i] * image_pix
            post_rate=count_spikes(lif_neuron(post_syn_current))/max_spikes
            if i == color:
                post_rate =1
                 //increase weight if correctly predicted
                weight[i][pixel] += pre_rate * post_rate  * 0.7
            else:
                // decrease weight if not correctly predicted.
                weight[i][pixel] -= pre_rate * post_rate  * 0.7
```

Hebb's training algorithm - (3a)

IV. Decoding Scheme:

Once the input image is processed, to know the decision of the system, we look at the highest spike generated by the output neuron. We have labelled the output neurons to be

0 → Red

1 → Green

2 → Yellow

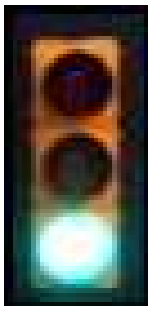
Based on the above labels, if the max spikes are generated by 0 then the model predicts red if 1 then green and so on.

---

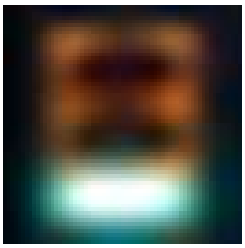
4. Results and Discussion

We have used YOLOv3 to extract the traffic light from the real time image to test the trained model. The YOLOv3 we used in this project already has pre-trained weights so it can detect many objects in the input image. However, we are just using this trained YOLOv3 to extract the traffic light object from the input image (Fig 1a). Once, we get the traffic light image (Fig 1b), we convert that image into 10 x 10 pixels (Fig 1c), and then determine the color of the traffic light to make the decision. We have decided to split the data set: 50% training, 25% validation

(1a) YOLOv3 traffic light detection



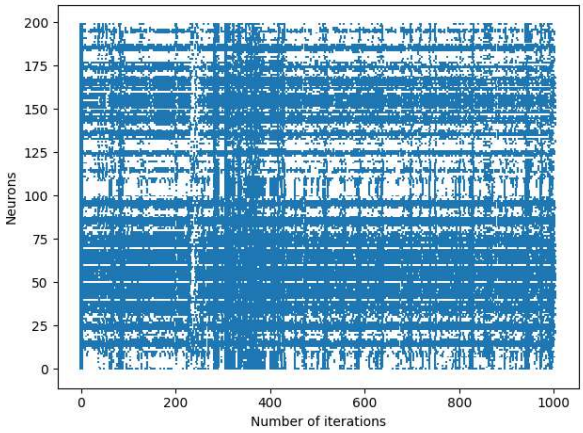
(1b) Traffic light extraction



(1c) Converting it into 10x10 pixel matrix.

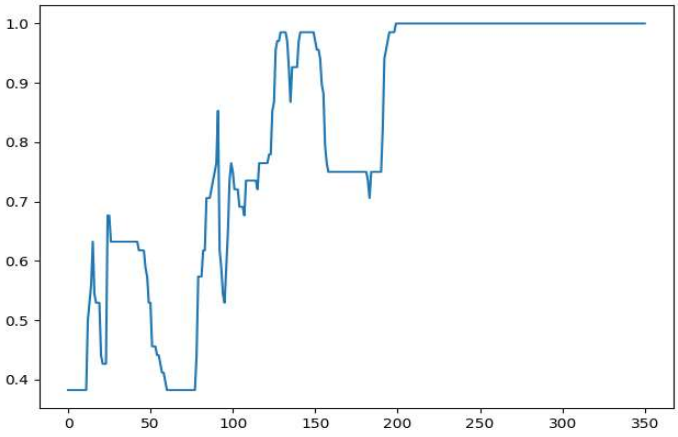
and 25% testing. Our accuracies for each of the sets are given in the following table:

	RED	GREEN	YELLOW
Training Accuracy	94.2	100.0	100.0
Validating Accuracy	92.6	99.48	99.01
Testing Accuracy	88.54	99.21	98.17



(Fig 2) Raster Plot of training Red Traffic Lights

There are a total of 200 input neurons and their spike activity for training Red light is given below in the form of a raster plot (Fig 2). The spiking of the neuron forms a unique pattern which is later used to identify the color of traffic light depending on the location it is found. For example it is unlikely for neurons between 100 and 125 to spike if detecting red color. The model also works for rotated traffic lights the same way without rotation. The final decision of detecting color does not only depend upon location but also depends upon the RGB value of that pixel.



(Fig 3) Accuracy v. Iterations

In figure 3, the X-axis represents the number of iterations and the Y-axis represents accuracy. As we can observe, the accuracy depends on the number of iterations. As the number of iterations increases the accuracy of the model increases. After reaching a certain number of iterations, the accuracy improves and gets close to 99%.

The weight matrix is generated after training the SNN model as shown in the algorithm (3a). After creating the weight matrix the final output is calculated on multiplying the image matrix with weights. This will give 3 outputs following the order in the decoding scheme. The argument with highest value would be the predicted outcome. For an input image of 10x10 (Fig 4a), it will generate values as shown in Fig. 4b, with maximum value in yellow (index 2).

Fig. 4a

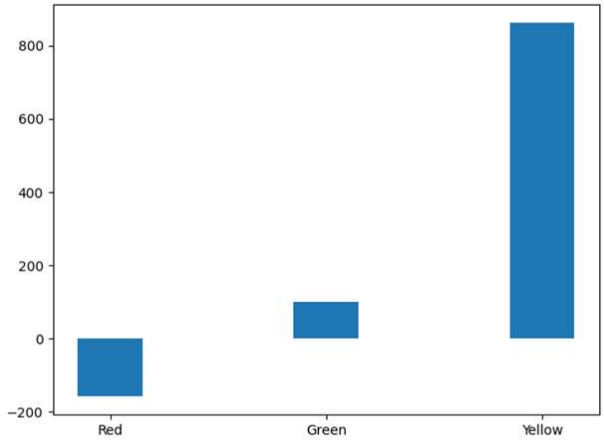


Fig. 4b

5. Conclusions

We proposed a traffic light color detection system which is realized within a single layer spiking neural network. Our system uses pre-trained YOLOv3 to detect the traffic light from the background. Followed by SNN to detect the traffic light color given based on the rate based encoding with high accuracy. Moreover, the proposed solution solves the problem of recognizing the multiple traffic lights given at any angle. However, in the case of the opposite facing traffic our model still detects green. So, in the future research it will be useful to solve that issue and expand the research to make the model discriminate between green straight and green left lights as a car cannot go in both directions at the same time.

Acknowledgments

We are thankful to Professor Konstantinos Michmizos (Rutgers University) and Vladimir Ivanov (Rutgers University) for guiding and assisting us throughout this project. Their experience and skills helped us improve and learn in depth about Spiking Neural Networks.

References

[1] T. Yeh, S. Lin, H. -. Lin, S. Chan, C. Lin and Y. Lin, "Traffic Light Detection using Convolutional Neural Networks and Lidar Data," 2019 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Taipei, Taiwan, 2019, pp. 1-2, doi: 10.1109/ISPACS48206.2019.8986310.

[2] B. Meftah, O. L´ezoray, S. Chaturvedi, A. A. Khurshid, and A. Benyettou, “Image Processing with Spiking Neural Networks” [Online]. Available: https://lezoray.users.greyc.fr/Publis/Meftah_Turing2012.pdf. [Accessed: 21-Dec-2020].

[3]A. Sboev, A. Serenko, R. Rybka, and D. Vlasov, “Solving a classification task by spiking neural network with STDP based on rate and temporal input encoding,” *Wiley Online Library*, 29-Jan-2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mma.6241. [Accessed: 21-Dec-2020].

[4] Gerstner, W., Kistler, W. “Spiking Neuron Models”, Cambridge University Press, 2002.

Appendix

Include all the code that you used – commented and clean.

- https://github.com/arunponnusamy/object-detection-opencv
- https://github.com/hp240920/Term-Project-BIC