Traffic Light Recognition Assistance for Colour Vision Deficiency Using Deep Learning

Jun Yong Lee[1], Hu Ng[1(✉)], Timothy Tzen Vun Yap[1], Vik Tor Goh[2], and Hau Lee Tong[1]

[1] Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Malaysia
nghu@mmu.edu.my

[2] Faculty of Engineering, Multimedia University, Cyberjaya, Malaysia

Abstract. This research intends to train a model to recognize traffic light signals in real-time to allow a person with Colour Vision Deficiency to identify the current signal of traffic lights with a mobile device's camera. First, LISA Traffic Light Dataset is downloaded obtained from Kaggle. Then, two data pre-processing steps are carried out, namely label map generation and TFRecords conversion. A total of six models, SSD MobileNet V2 320 × 320, SSD MobileNet V1 FPN 640 × 640, SSD ResNet50 V1 FPN 1024 × 1024, SSD ResNet101 V1 FPN 1024 × 1024, FasterRCNN ResNet50 V1 FPN1024 × 1024 and FasterRCNN ResNet101 V1 FPN 1024 × 1024 are used from the TensorFlow Model Zoo to perform training and evaluation on the dataset. From the experiment results, the most suitable object detection model is FasterRCNN ResNet101 V1 FPN 1024 × 1024 with the highest recall rate of 52.4% for daytime images and 45.4% for nighttime images.

Keywords: FasterRCNN · Traffic Light · Object Detection · TensorFlow

1 Introduction

Object detection is the capability of computers to analyse and identify an object from an image. Object detection is widely used for face detection, vehicle detection, pedestrian counting, web images, security systems and self-driving cars. Real time object detection can be done in real-time which is the task of performing object detection in real-time with low delays while maintaining a base level of accuracy.

The focus of this paper is traffic light recognition assistance for colour vision deficiency using deep learning on a published dataset (LISA Traffic Light Dataset) [1]. A real-time application is developed to perform traffic light recognition and alert the user of the current state of the traffic light (green or red) via a smart phone.

S.-C. Haw and K. Sonai Muthu (Eds.): CITIC 2022, 10, pp. 289–300, 2022.
https://doi.org/10.2991/978-94-6463-094-7_23

2 Related Work

Wang et al. [2] worked on traffic light recognition from an image and clipping the traffic light as Region of Interest (ROI). A lightweight CNN called TL-NET was then built to classify the traffic light status from the ROI image. They proposed two ways to improve the accuracy of traffic light state recognition. Firstly, obtaining a variety of traffic light samples to improve the performance of TL.NET. The additional approach is included multiple sensors in the intelligent vehicle systems, such as long-focus camera images. They achieved 99% of accuracy rate by YOLOv3 model on BDDV dataset.

Jensen et al. [3, 4] applied YOLO for traffic light detection on 40,764 images from LISA traffic light dataset. The model achieved a recall performance of 69.38% using the input image size of 672 × 672 and a lower recall performance of 59.65% using the input image size of 416 × 416. However, this study focused on traffic light detection and does not detect different traffic light signals such as turn to left or right.

Ennahhal et al. [5] employed multiple models such as Faster R-CNN, R-FCN and SSD approaches for traffic light detections on 43,007 images from LISA traffic light dataset. They found out that Faster R-CNN yielding the best accuracy of 76.37%, using the evaluation metric of mAP@0.5.

Khan and Ansari [6] employed inertial sensor fusion to perform real-time traffic light detection from videos. They built a fusion system by utilizing a smartphone to capture the frames of images and a computer to do the traffic light detection. They achieved accuracy rate of 97% for green lights detection by Heuristic filters model on 26,088 images from self collected dataset.

Evan et al. [7] employed SSD Mobilenet V2 for recognition of pedestrian traffic lights. Unlike traffic lights for vehicles, pedestrian traffic lights usually do not have a yellow light and only have a green and red light. They concluded that SSD MobileNet V2 can detect multiple objects from a single frame in real time scenario and suitable for low specifications computer system. They achieved 86% of accuracy rate on 530 images from self-collected dataset.

Lu et al. [8] presented an attention model-based detection framework to perform the detection of a smaller subset of objects in large high-resolution images. The model was trained with 16,000 images from Tsinghua–Tencent traffic light (TTTL) dataset. They achieved 83% They achieved 86% of accuracy rate on the LISA dataset.

He et al. [9] applied Capsule Neural Networks (Caps Nets) and Convolutional Neural Network (CNN) models to detect traffic lights on 12,630 images from Udacity Carla dataset. Both models were able to reach a validation accuracy of 100%.

3 Methodology

In this study, a well-known published traffic light dataset, LISA Traffic Light Dataset [1] was downloaded for deep learning models training and performance evaluation. The dataset consists of 43,007 images, and 113,888 annotated traffic lights. The sequences are captured by a stereo camera mounted on the roof of a vehicle driving under both night- and daytime with varying light and weather conditions. There are 28943 images of DayClip, and 14064 images of NightClip. In this paper, only the left camera view

Fig. 1. Image from DayClip—Day with bright scenario

from the dataset is being utilised. Figures 1 and 2 show the sample images from the dataset in Day with bright scenario and Night with low-light scenario.

Label Map represents a map of all different labels from the dataset. In this paper, only two situations ("Stop" and "Go") for the traffic light are being identified and applied for detection. Therefore, those unwanted annotations such as "StopLeft", "GoLeft", "Warning" and "WarningLeft" are being removed.

As TensorFlow only allows model to be trained by TFRecord, which is why all the images, annotations and labels used for training are generated into a TFRecord file. The TFRecord format is a simple format used to store a sequence of binary records. Therefore, all the downloaded images are converted into TFRecord for later usage in model training and evaluation testing. Figure 3 shows the flowchart of methodology stages.

3.1 Model Selection

A total of six models, namely SSD MobileNet V2 320 × 320, SSD MobileNet V1 FPN 640 × 640, SSD ResNet50 V1 FPN 1024 × 1024, SSD ResNet101 V1 FPN 1024 × 1024, FasterRCNN ResNet50 V1 FPN1024 × 1024 and FasterRCNN ResNet101 V1 FPN 1024 × 1024 on TensorFlow Model Zoo are selected to be trained on the downloaded dataset, split into DayClip and NightClip respectively. The model which achieved the best result after transfer learning will be selected and be tested for more train-split ratios in the later experiments, with intention to find out the evaluation performance of the related model.

Table 1 shows a summary of performance of DayClip and Table 2 shows a summary performance of NightClip tested on 90:10 train-test ratio. From the Tables 1 and 2, it can be observed that the FasterRCNN ResNet101 V1 FPN 1024 × 1024 model achieved the best result among all the six models.

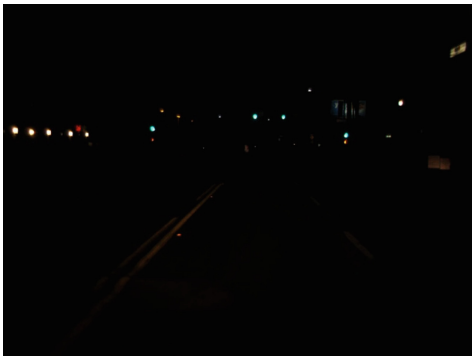

Fig. 2. Image from NightClip—Night with low-light scenario

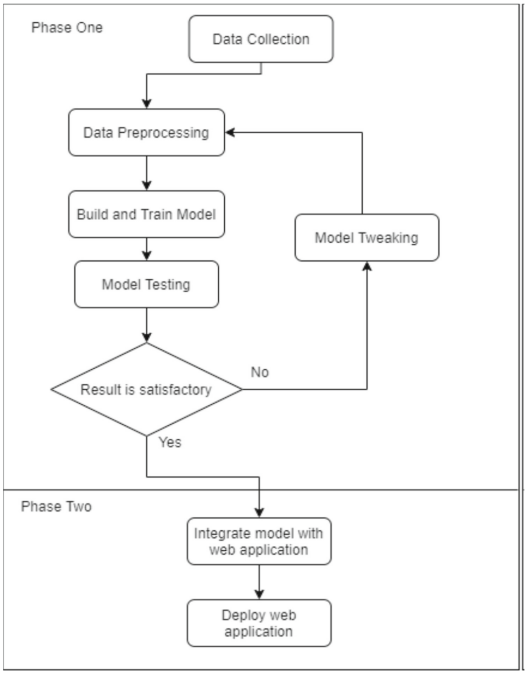

Fig. 3. Flowchart of methodology stages

3.2 Model Evaluation

From the preliminary model training stage, the model FasterRCNN ResNet101 V1 FPN 1024 × 1024 is found to be the model that exhibited the best performance amongst six of the models. Further training and testing are carried on more train-test split ratios, with adding the combination of NightClip and DayClip dataset. Table 3 shows the average precision and recall of each dataset and train-test split ratio respectively.

Table 1. Summary of performance of DayClip during preliminary training tested on 90:10 train-test ratio.

Model	Precision (%)	Recall (%)
SSD MobileNet V2 320 × 320	0	0
SSD MobileNet V1 FPN 640 × 640	5.8	36.1
SSD ResNet50 V1 FPN 1024 × 1024	0	1.7
SSD ResNet101 V1 FPN 1024 × 1024	0.8	3.6
FasterRCNN ResNet50 V1 FPN 1024 × 1024	13.9	46.0
FasterRCNN ResNet101 V1 FPN 1024 × 1024	19.3	52.4

Table 2. Summary of performance of NightClip during preliminary training tested on 90:10 train-test ratio.

Model	Precision (%)	Recall (%)
SSD MobileNet V2 320 × 320	0	0
SSD MobileNet V1 FPN 640 × 640	3.5	16.7
SSD ResNet50 V1 FPN 1024 × 1024	0	0
SSD ResNet101 V1 FPN 1024 × 1024	0	0.9
FasterRCNN ResNet50 V1 FPN 1024 × 1024	1.7	5.4
FasterRCNN ResNet101 V1 FPN 1024 × 1024	10.9	37.8

Figures 4 and 5 show the detection of 'go' and 'stop' traffic light respectively for DayClip scenario. The model can detect all traffic lights correctly, with two different traffic lights to be detected in each of the frame. Figures 6 and 7 show the detection of 'go' and 'stop' traffic light respectively for NightClip scenario.

The model can detect all traffic lights correctly similarly to the NightClip model. However, once major weakness that can be inferenced from the NightClip detection was that the confidence level for detection on the NightClip was significantly lower than the DayClip. On the DayClip, the detections showed a confidence level of more than 80%, compared to the NightClip which showed a confidence level of less than 50%, despite having detected the traffic light correctly.

Figure 8 shows a sample of a false negative inference. Two 'stop' traffic lights can be seen, one on the left and one on the right, but the model was unable to detect the traffic lights. Figure 9 shows a sample of a false positive inference. One 'stop' traffic light can be seen in the image which was detected correctly. However, a 'stop' signal was detect by the model with a confidence of 80% but it was not in the image, making it a false positive detection.

Figure 10 shows a sample of double detection on a frame. It can be observed that there are two traffic lights on the right side with 'go' signal. However, the traffic light on the most right shows two bounding boxes, which 95 and 56% confidence score respectively.

Table 3. Average performance of model FasterRCNN ResNet101 V1 FPN 1024 × 1024 of each dataset and various train-test ratios.

Dataset	Ratio	mAP @ 0.5:0.95 (%)	Recall @ 0.5:0.95 (%)
DayClip	90:10	39.8	52.4
DayClip	80:20	43.7	51.0
DayClip	70:30	39.2	47.9
DayClip	60:40	50.6	50.4
NightClip	90:10	33.7	37.8
NightClip	80:20	37.4	43.8
NightClip	70:30	32.5	38.2
NightClip	60:40	33.2	33.9
DayClip & NightClip	90:10	24.2	42.9
DayClip & NightClip	80:20	35.7	13.0
DayClip & NightClip	70:30	34.2	38.9
DayClip & NightClip	60:40	37.0	36.0

Fig. 4. DayClip detection of ‘go’.

Fig. 5. DayClip detection of ‘stop’.

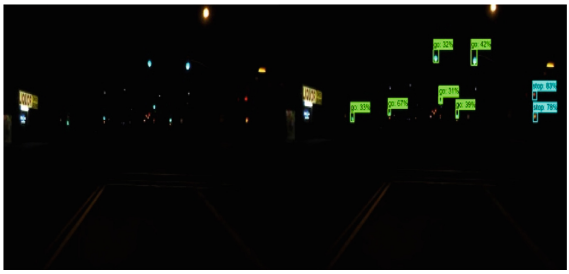

Fig. 6. NightClip detection of ‘go’.

Fig. 7. NightClip detection of ‘stop’.

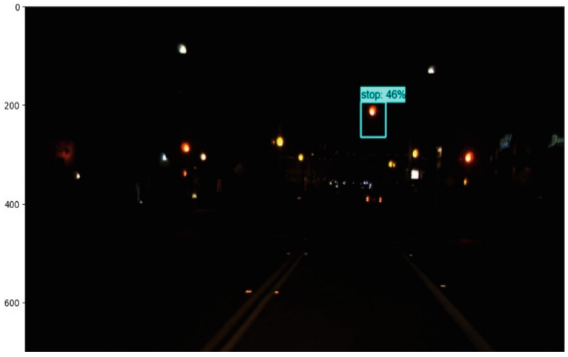

Fig. 8. Sample of false negative inference.

This is an example of double detection as two boxes were drawn where there was only one traffic light in the actual scenario.

3.3 Data Augmentation

The model trained on NightClip had showed significantly weaker performance compared to the model trained on the DayClip dataset. It can be hypothesized that this was due to the lower amount of sample size available on the NightClip. To increase the performance

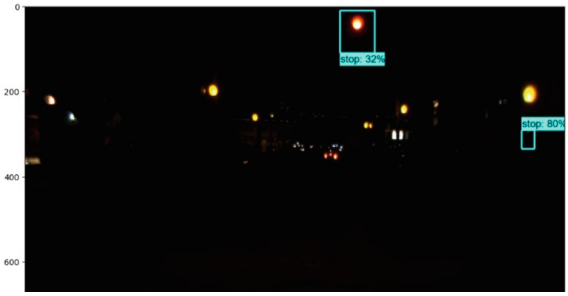

Fig. 9. Sample of false positive inference.

Fig. 10. Sample of double detection.

of NightClip, data augmentation is carried out on the NightClip to expand and diversify the data.

Table 4 shows the results of the model before and after data augmentation on the NightClip dataset. Except for the train-test split ratio of 70:30, the model with data augmentation exhibited lower average precision and recall compared with the model without data augmentation. Due to this reason, the model with data augmentation was rejected and will not be used in the object detection stages of this paper.

4 Real-Time Detection

4.1 Python OpenCV Webcam Stream

The object detection model trained can be frozen and loaded for inference. The loaded model can perform detections on image that was inputted. To improve the user experience when using the detection model, real-time detection was enabled by using the webcam

Table 4. Average performance of model FasterRCNN ResNet101 V1 FPN 1024 × 1024 of each train-test ratio with data augmentation on NightClip dataset.

Ratio	mAP @ 0.5:0.95 (%)	Recall @ 0.5:0.95 (%)
90:10	20.9	23.6
80:20	31.0	36.3
70:30	36.9	45.4
60:40	16.7	26.1

Fig. 11. Detection of traffic light in real-time using a webcam.

with OpenCV Webcam stream. This allows user to perform detection in real-time using the webcam which detect traffic light status without the need of uploading images one by one. Figure 10 shows the screenshot of the detection of traffic light in real-time using a webcam. The program can successfully detect the two 'go' traffic lights in the frame that is shown on a mobile phone with the image opened pointing to the webcam (Fig. 11).

Although the program can detect traffic lights in real-time successfully, the model is found consuming heavy computation resources and may not be worked on all computer machines. The program was run on a machine with a dedicated graphics processing unit (GPU) of NVIDIA GeForce GTX 1050. When the model is running, 90% of the processing power of the GPU was utilized, and 3.5GB out of 4.0GB of the GPU memory is used. Due the high processing power the model, real-time detection may not work on machines with lower specifications of GPU.

4.2 Javascript Web Application Stream

The OpenCV Webcam Stream program can detect the traffic lights in real-time successfully. However, the program developed is lacking on the accessibility as users can only

Fig. 12. Screenshot of the web application for making detections

use it on machines that run with Python installed, which is not available on a normal mobile device.

This research developed a web application to perform real-time traffic lights detection, which can be accessed by all devices if there is internet connection. The web application is designed to perform detection of traffic light in the frame rate of 100ms, drawing the predicted bounding boxes directly in the frame of video.

As the model was run through the browser and not directly through the OS, the model has no assessment to the dedicated GPU. The model was run on the on-board GPU of the machine's motherboard, which is significantly less powerful that the dedicated GPU. Due to this reason, the model is unable to be run successfully, as it had utilized 100% of the GPU and crashed the browser of the machine, rendering the browser unusable.

Figure 12 shows a screenshot of the web application that successfully detected the traffic lights from an image shown on mobile phone to the webcam. Two of the traffic lights were successfully detected. However, it took 15 s to perform the detection on a machine using an on-board GPU of 'Intel(R) HD Graphics 630'. Of the 15 s used to perform the detection, the browser was running with rendering unusable until the detection was performed successfully.

As concluded, the proposed model built in this paper is not suitable for mobile usage, this is due to heavy computer resources and requires a dedicated GPU for real-time detections. This problem can be solved if the smart phone is equipped with more powerful of processor in future.

5 Conclusion

In this paper, a total of six type of models were tested to perform object detection of traffic lights on the LISA dataset. After experimenting, the model 'FasterRCNN ResNet101 V1 FPN 1024 × 1024' was found to be the best performing model out of the six types of models. For the model built for DayClip daytime scenario, mAP@0.5 of 50.6% and average recall of 50.4% was achieved. For the model built for NightClip scenario, mAP@0.5 33.2% and average recall of 33.9% was achieved. Detections on the NightClip

were found to be less accurate than detections on the DayClip due to the lower quality of images from the NightClip, as well as lesser amount of data available.

Although the model 'FasterRCNN ResNet101 V1 FPN 1024 × 1024' was found to be the best performing among the six types of models tested in this paper, this model was however proven to be not suitable for mobile usage, as it took too long to perform inferences on images on machines without a dedicated GPU. Inferencing an image took a minimum of 15 s on a machine without a dedicated GPU.

More models especially light-weight object detection models should be tested and experimented on to allow the use of the model on mobile devices. Accuracy may have to be sacrificed for speed as models with higher accuracy generally requires more time for making detections, thereby having a higher hardware requirement. More datasets should also be acquired, especially for night images, in order to obtain a better result for the trained models. The website built in this paper was a prototype and future works could be done to improve the user interface and user experience design.

Authors' Contributions. Hu Ng, roles: Corresponding author, Conceptualization, Project Administration, Supervision, Validation, Visualization, Writing—Review & Editing.

Jun Yong Lee, roles: Data Curation, Formal Analysis, Investigation, Resources, Writing—Original Draft Preparation.

Timothy Tzen Vun Yap, roles: Conceptualization, Investigation, Project Administration, Supervision, Validation, Writing—Review & Editing.

Vik Tor Goh, roles: Validation, Writing—Review & Editing.

Hau Lee Tong, roles: Supervision, Validation, Writing—Review & Editing.

References

1. https://www.kaggle.com/mbornoe/lisa-traffic-light-dataset. Date: 15 June 2022.
2. Wang, X., Jiang, T., & Xie, Y. (2018, December). A Method of Traffic Light Status Recognition Based on Deep Learning. In Proceedings of the 2018 International Conference on Robotics, Control and Automation Engineering (pp. 166–170).
3. Jensen, M. B., Nasrollahi, K., & Moeslund, T. B. (2017). Evaluating state-of-the-art object detector on challenging traffic light data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 9–15).
4. Jensen, M. B., Philipsen, M. P., Møgelmose, A., Moeslund, T. B., & Trivedi, M. M. (2016). Vision for looking at traffic lights: Issues, survey, and perspectives. IEEE Transactions on Intelligent Transportation Systems, 17(7), 1800–1815.
5. Ennahhal, Z., Berrada, I., & Fardousse, K. (2019). Real time traffic light detection and classification using deep learning. In 2019 International Conference on Wireless Networks and Mobile Communications (WINCOM) (pp. 1–7). IEEE.
6. Khan, N. A., & Ansari, R. (2018, November). Real-time traffic light detection from videos with inertial sensor fusion. In Proceedings of the 1st ACM SIGSPATIAL Workshop on Advances on Resilient and Intelligent Cities (pp. 31–40).
7. Wulandari, M., & Syamsudin, E. (2020, December). Recognition of pedestrian traffic light using tensorflow and SSD MobileNet V2. In IOP Conference Series: Materials Science and Engineering (Vol. 1007, No. 1, p. 012022). IOP Publishing.

8. Lu, Y., Lu, J., Zhang, S., & Hall, P. (2018). Traffic signal detection and classification in street views using an attention model. Computational Visual Media, 4(3), 253–266.
9. He, J. S., Han, M., Zheng, G. J., & Ray, H. (2020, April). Implementing Capsule Neural Networks in Traffic Light Image Recognition. In Proceedings of the 2020 ACM Southeast Conference (pp. 301–302).