Real-Time Traffic Lights Identification using YOLOv3 Algorithm For Autonomous Vehicles

Rachel Kozel, and Naeemah Robert
School of Electrical and Computer Engineering
University of Arizona, USA

August 12, 2020

1 Abstract

Safety has always been a priority in automobile manufacture. Traffic light detection plays a major role in regards to safety in autonomous vehicles. Previous methods involved utilizing a combination of image processing and training a neural network model. Those methods were not fully successful as they presented limitations such as trouble detecting yellow and arrow traffic lights, the inability to identify a traffic light from a few pixels due to long distances, and the obstruction of traffic lights causing failure of detection. We propose to identify traffic lights and their states in both urban and suburban areas by developing a deep learning model utilizing a YOLOv3 model. Additionally, we will use the large dataset of traffic light images from the 'Bosch Small Traffic Lights Dataset' to train the YOLOv3. Our proposed method will be successful because a better processed dataset indicates traffic lights in busier environments, and a balanced dataset allows the model to be better trained at identifying traffic signals and their status. The success of our proposed method will benefit autonomous vehicles' feasibility by improving safety at intersections. The risks of this method include missing potential traffic signals which could lead to potential accidents. We will use a camera that will record the data that will later be processed. The mid-term check for success is accurate detection of traffic lights in ROS simulations, and the final check for success is the successful test drive of University of Arizona's CAT Vehicle by detecting traffic lights at intersections.

Figure 1: This is the University of Arizona CAT vehicle.

2 Introduction

Object detection has been a field within signal processing that has been investigated for a while. Although there are different image processing methods, machine learning has increasingly been used lately thanks to the developments in that field over recent years. Different neural networks have been constructed and tested out such as Regional Convolutional Neural Networks (RCNN), Fast RCNN, Faster RCNN, Mask RCNN, and You Only Look Once (YOLO), which all yield different results in varying circumstances [1]. Approaches also have been made to combine image processing with deep learning by using image processing to preprocess the data before feeding it into a neural network [1][2][3][4][5][6][7]. This often yields better results that just using one of those formerly mentioned methods alone, especially when variables like dataset size, object size, and varying illumination have been items that have inhibited object detection [1][6]. In the situation that our team deals with of detecting traffic lights, those variables are key factors that are still at play in addition to the numerous states that need to be detected for later decision making. Problems have risen over the years with detecting different states because of the lack of a balanced dataset [6]. This occurs because the high prevalence of red traffic lights (TLs) compared to yellow or arrow TLs and is often a topic addressed in the results of different papers as lowering the accuracy of neural network models [8]. Another situation that is brought up in detection of TLs is constructing a model that is usable for autonomous vehicles in real-time [8][6]. This affected the decision of the model that we ended up using. We found that we had to switch from using the Mask RCNN that prioritizes accuracy but sacrifices real-time capabilities to use the YOLOv3, which does detection in real-time. The goal was to achieve real-time detection and try to train the model with anchors suited for the dataset even though the dataset used did not have the length and potentially balance that will be needed for accuracy.

3 Background

Previous research focuses on using image processing and deep learning. Image processing-based methods often used transformation of the colorspace and adaptive templates to detect the TLs [8]. The colorspace conversion in [8] used a grayscale transformation and detective step and the adaptive template did the identification [8]. Another method implemented by [9] used the mapping of TLs done by GPS to help track the TLs that could potentially be encountered. This allowed them to not miss TLs, which was their goal [9]. In [10] they used a similar method by using the database of already mapped TLs to have a digital map that would help localize the TLs position. The Blob detector and the Circular Hough Transform were used to help find candidate objects, and then the light color was classified with a confidence and probability attached that were determined by with prior information such as the digital maps [10]. The relative position of the TLs from each other and their respective states helped find the probability of an arrow light [10]. In [11] they focused on solely detecting an object and used gray-scaling as their primary color processing method. They did shape detection and edge detection also with the Circular Hough Transform, but it was not always helpful with overlapping objects that they were trying to count in MATLAB [11].

Most deep learning methods used preprocessing methods that had their roots in image processing. Although [3] used a Convolutional Neural Network (CNN), they used it to detect text from images by using corner edge detection and position-sensitive segmentation. Colorspace transformations like HSV [2][4] and

HSI [6] are popular since they separate the varying illumination component from the chromatic components to help find candidate regions. In [2] they also used a Maximally Stable Extremal Region (MSER) to help shrink down the possible areas for TL structures before sending the information to a Histogram of Oriented Gradients (HOG) to extract the features that would be confirmed with a Support Vector Machine (SVM). This was done as preprocessing before sending the results to a CNN to identify the state of the TL in the image [2]. The SVM was also utilized in addition to Gaussian blurring and the HSV transformation in [4] where they used the LISA TL dataset, but in this method they compared the SVM to an artificial neural network, Naive Bayes method, and Random Forest (RF) and found the SVM yielded the best result at the time. Other sources like [7] did object localization and received edge information from Generic Edge Tokens and used a Best First Search algorithm to assist in adjusting it to get scores from the Deep CNN. Another Deep CNN-based model was used in [6] but they also did the HSI color-space to remove the varying illumination problems. What they ended up recommending was they needed more TL cases, though, with worse lighting conditions to better help the dataset [6]. Another suggestion in [5] was that preprocessing methods often improved the results, and that when they were used in combination with one another, it improved the results. The method used in [1] involved cropping, resizing, and subtracting the RGB pixel values from the mean for preprocessing before sending the result to a Single-Shot Detector (SSD) to determine the candidates then assign points according to a reward system to root out false positives.

With all of this in mind and considering the time constraints after the inadequacy of using Mask RCNN, the ultimate decision was made to implement the YOLOv3 algorithm since it was one that could run in real-time, good accuracy, and ease of implementation within the short time frame.

4 Methodology

In our research project, we proposed at first to use the Mask Region-Based Convolutional Neural Network (Mask RCNN). The Mask RCNN is the last variant of RCNN which was introduced in 2018 by Hermenn et al. in [12]. Mask RCNN is used for object detection and object segmentation. In other words, Mask RCNN is able to detect different objects in one image and give the classes, the bounding boxes, and the masks of the detected objects in the images [13] [14]. Additionally, we decided to use the Mask RCNN because it involves a region proposal algorithm to better speed up the time taken to train as well as faster identification. The additional reason for the choice of the region proposal algorithm was because it involved computing the probability an object of a specific class was contained within that region [15]. It continues to compute and shrink down the number of regions that have lower probability until a few remain above a certain threshold, which will constitute the mask [15]. This appeared to be beneficial for the detection of TLs because their higher probability at appearing in specific locations of image frames. Code already exists for this algorithm that had simple modifications for the training of multiple classes, which made it seem ideal for our team's purposes.

Our work was done on a virtual machine in Vmware Workstation Player 15 with Ubuntu 18.04. In order to train our model, we used the "Bosch Small Traffic Lights Dataset" that was previously used in [16]. The dataset came with 5093 images and their corresponding annotation files. The annotation files contain thirteen classes and the coordinates for the bounding boxes. We used previous Mask RCNN models in [13] and made modifications according to the Bosch dataset. In [13], they used a Kangaroo Dataset for which we had to replace with the traffic lights dataset. We loaded the dataset we were working with by adding the corresponding classes to the dataset as it was done in [17] and in `https://stackoverflow.com/questions/59811406/adding-multiple-classes-in-mask-r-cnn`, then defined the number of images that would be used for training and testing, extracted the bounding boxes with the coordinates from the annotation files, loaded the masks, and finally displayed the images with the boxes, masks, and classes.

After we loaded the masks on the images in our dataset, we moved on to the training process. We specified the number of classes and epochs which were 13 and 60 respectively. For the training process, we downloaded the COCO weights because we were training our model based on the Mask RCNN COCO dataset. As the training process began, some of our computers could not keep up with the GPU being used up to ninety percent of its performance. To overcome this problem, we signed into the University of Arizona's ECE server and moved the training process to the University of Arizona's High Performance Computing (HPC) server. HPC is a network of computers which can handle big computer jobs such as the training of a model. With those computers combined, the jobs are processed much faster. In our case, the training process took 3 hours, which otherwise would have taken us at least three days. We then proceeded with the testing of the trained model with the images that were set for testing in the beginning. The model was able to detect the different states of the TLs in the images we previously set for testing (see fig.2). However, we wanted the confidence to be at least ninety percent and up.

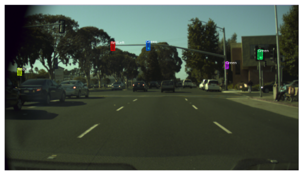

Figure 2: Testing of Trained Model.

After our neural net model was trained, we wanted to use it for real-time TL detection. In order to do so, we used the OpenCV library and matplotlib with our trained model in addition to slight modifications as it was implemented in [18]. Our model was then able to detect TLs in images found on Google (see fig.3). However, with the Mask RCNN model, the real-time detection was very slow and impossible. Therefore, we had to switch our approach to a different machine learning model in order to satisfy our real-time criteria.

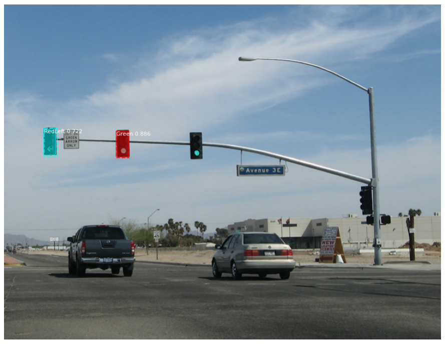

Figure 3: Testing of Trained Model on Google Images.

After realizing that Mask RCNN was not giving the results we were expecting, we switched to YOLOv3. YOLOv3 is the latest variant of YOLO-which stands for You Only Look Once- that was introduced in 2018 in [19]. YOLO is an object detection algorithm known for its detection speed and the ability to detect objects in video feed or live feed on webcam [13]. We were able to find Yolov3 models which have been previously implemented in [13]. We started by downloading the yolov3 weights then ran the code to see the results of its implementation with Keras. As the results met our expectations, we changed some data in the configuration and the voc files downloaded from this Github repository in `https://github.com/experiencor/keras-yolo3.git` to adjust the code to our dataset. We added the thirtheen classes of our dataset and developed the anchors based on our dataset. Once again, we continued with the training process on the UA HPC server for a faster training time. After training the model for 24 hours, we moved on with the testing process. We wanted to test the real time detection of our model and it was able to detect traffic lights with a confidence between forty and fifty percent. We continued the training process because, as previously mentioned, we wanted the confidence between ninety and ninety nine percent.

5 Results

The YOLOv3 model has been trained over 3 times for over 100 epochs. It is able to detect red and green traffic lights (see fig.4). It takes more training time for the YOLOv3 model than the Mask RCNN model to give a confidence level between ninety percent and ninety-nine percent. Additionally, the YOLOv3 model has difficulties detecting yellow lights and arrow lights (see fig.5). It is one of the problem we did not have a chance to resolve as it requires to train the model for a longer time and on a more balanced dataset. Moreover, we tested the model for real-time traffic light detection. The results are promising as it is detecting traffic lights where they are placed on the road, and it detected red and green lights with a close level of accuracy. It also did not detect vehicle tail lights as traffic lights. We noticed once more the struggle the model has for detecting yellow and arrow lights in addition to trying to do this on a lower-quality, regular laptop GPU. The lack of a stronger GPU made real-time detection with an HD 1080p Logitech webcam have some lag.

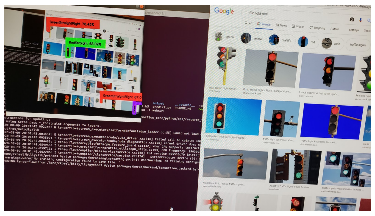
Figure 4: Testing of Trained Yolov3 Model on Google Images.

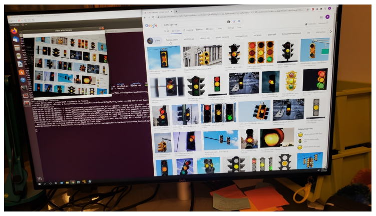
Figure 5: Testing of Trained Yolov3 Model on Google Images.

6 Conclusion

Upon examining the results, our team has learned that the Mask RCNN was not adequate for our purposes based on the criteria it did not function in what we considered real-time. That prompted the switch to the YOLOv3 model. Based on the results discussed, our team has established that it does function close to in real-time despite some lag, which is the main criteria. It does detect TLs with okay accuracy despite the distance of the TL from the camera source. The test on our team's personal webcams due to the COVID-19 crisis shows that it does detect TLs on Google images, and upon taking the webcams and computers to test live, it also detects TLs with similar accuracy. The main challenges our team could not overcome in time consist of the struggle to recognize yellow and arrow lights as well as the lag in our webcam feed. This is a source for further research that we would recommend for future research with a more balanced dataset, more training time, and a stronger GPU. Future work could also consist of testing this model in different weather conditions and illumination to confirm its integrity outside the indoor environment as well as with more diverse datasets in different regions. Our team would like to thank the University of Arizona and our mentor Safwan for giving us the opportunity to be able to work on this project and learn about this topic and its importance to autonomous vehicles.

References

[1] J. Kim et al. “Deep Traffic Light Detection for Self-driving Cars from a Large-scale Dataset”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 280–285.

[2] S. Saini et al. “An efficient vision-based traffic light detection and state recognition for autonomous vehicles”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 606–611.

[3] Ulagamuthalvi, J. B. J. Felicita, and D. Abinaya. “An Efficient Object Detection Model Using Convolution Neural Networks”. In: *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. 2019, pp. 142–147.

[4] J. L. Binangkit and D. H. Widyantoro. “Increasing accuracy of traffic light color detection and recognition using machine learning”. In: *2016 10th International Conference on Telecommunication Systems Services and Applications (TSSA)*. 2016, pp. 1–5.

[5] D Anggraeni et al. “Enhancing CNN with Preprocessing Stage in Automatic Emotion Recognition”. In: *Procedia Computer Science* 116 (2017), pp. 523–529.

[6] Z. Ouyang et al. “Deep CNN-Based Real-Time Traffic Light Detector for Self-Driving Vehicles”. In: *IEEE Transactions on Mobile Computing* 19.2 (2020), pp. 300–313.

[7] E. Etemad and Q. Gao. “Object localization by optimizing convolutional neural network detection score using generic edge features”. In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 675–679.

[8] R. de Charette and F. Nashashibi. “Traffic light recognition using image processing compared to learning processes”. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 333–338.

[9] J. Levinson et al. “Traffic light mapping, localization, and state detection for autonomous vehicles”. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 5784–5791.

[10] Keisuke Yoneda et al. “Robust traffic light and arrow detection using digital map with spatial prior information for automated driving”. In: *Sensors* 20.4 (2020), p. 1181.

[11] R Hussin et al. “Digital image processing techniques for object detection from complex background image”. In: *Procedia Engineering* 41 (2012), pp. 340–344.

[12] Kaiming He et al. “Mask R-CNN”. In: *CoRR* abs/1703.06870 (2017). arXiv: `1703.06870`. URL: `http://arxiv.org/abs/1703.06870`.

[13] Jason Brownlee. “Deep Learning for Computer Vision: Image Classification, Object Detection, and Face Recognition in Python”. In: Machine Learning Mastery, 2019, pp. 368–388, 405–443.

[14] X Zhang. *Simple Understanding of Mask RCNN*. 2018.

[15] Pulkit Sharma. "Computer Vision Tutorial: Implementing Mask R-CNN for Image Segmentation (with Python Code)". In: (2019). URL: `https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/`.

[16] Dennis Hein. "Traffic Light Detection with Convolutional Neural Networks and 2D Camera Data". PhD thesis. fu-berlin, 2020.

[17] SriRam Govardhanam. "Training your own Data set using Mask R-CNN for Detecting Multiple Classes". In: (2020). URL: `https://medium.com/analytics-vidhya/training-your-own-data-set-using-mask-r-cnn-for-detecting-multiple-classes-3960ada85079`.

[18] Ablajan Sulaiman. "Image, Video and Real-Time Webcam Object Detection & Instance Segmentation using Mask R-CNN". In: (2020). URL: `https://medium.com/@toarches/image-video-and-real-time-webcam-object-detection-and-instance-segmentation-with-mask-rcnn-37a4675dcb49`.

[19] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).