Multi-stage Traffic Light Detection

Sean Wolfe, Nikilesh Subramaniam Christopher Truong
Department of Computer Science, University of Virginia, Charlottesville, VA 22904
[ns4bb, scw2tt, cvt7bm

Abstract

Our goal was to identify the state (color) of all traffic lights in a given image of the road. The dataset we used consisted of images taken from the driver's point of view. Our method involved using a Faster RCNN to get the bounding boxes of the traffic lights in the image and train a Convolutional Neural Network to detect the color of these traffic lights. The Faster RCNN successfully found traffic lights, even though some of the traffic lights were just 20 pixels big. The overall model was able to correctly classify red and green lights, but struggled with yellow lights. Some problems that were faced in the project was with our training data, as some traffic lights were covered or facing a different way. Overall, our our model had some success, but a Single-Shot Detector can be used in future to use one model instead of two.

1. Introduction

In recent years, development in self driving cars has been accelerated due to advancements in deep learning technologies. As part of the process of creating an autonomous vehicle, the computer needs to gain an understanding of the road around it. A small part of this understanding is that of identifying and knowing the state of traffic lights in view of the vehicle. Previous approaches to this problem have included using sliding windows over images and feeding the output of said windows to a well working image classification network. In recent years, state of the art has advanced towards networks like Faster RCNN, which use region proposal networks to suggest where to look for classification, and SSD, which train feature map priors in addition to the classification base network. This allows SSD to perform both localization and classification in a single forward pass of the network. Here, we'll be using Faster RCNN to find regions which we want to do more classification on, particularly on determining whether traffic lights are red, yellow, or green.

Figure 1: Here we show some sample images from the DriveU Dataset. They are taken from the point of view of the car

2. Related Work

One method [2] used to solve this problem utilized a kNN classifier. This method iterated over training data to develop cluster centroids representing 'Red', 'Yellow', and 'Green'. Then, using given an sample image, the kNN classifier was able to generate a distance image. Then circle detection would be done on the distance image to isolate the light. Finally, the color of the light would be determined. This method worked well when the traffic light was directly facing the camera and a full circle was visible. However, the image was less accurate in cases where the traffic signal

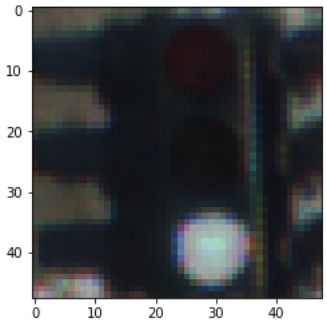

Figure 2: Cropped output of Faster RCNN network. This is fed into our convnet along with the associated light state label from our dataset.

Figure 3: Here we have a sample detection from Faster RCNN, the red boxes are the output from the network, and blue are the ground truth.

was slightly covered by its cover.

Another more advanced method [3] utilizes domain specific modifications to SSD in order to achieve better accuracy on traffic lights. In particular, the base VGG network is replaced by an Inception V3 network, and the box stride was reduced to better deal with the small sizes of the traffic lights relative to the images. This was found to work quite well for the the DriveU dataset the network was trained on.

3. Model

The dataset that was an annotated set of images taken from the perspective of a car. The annotations included bounding boxes for the traffic lights in the image as well as the color of those lights.

We first used the Faster RCNN model to get the bounding boxes of the traffic lights from the image. These bounding boxes were used to crop the images in the dataset to create training data for a network that would determine the color of the traffic lights. First, each image was run through a pretrained pytorch Faster RCNN. This network was trained on the COCO dataset and had an output of labels, bounding boxes, and confidence scores for each image. The output was trimmed to only include bounding boxes with labels for traffic lights and confidence scores above a threshold. Then each outputted bounding box was used to crop the original image and generate an image that isolated a traffic light. The bounding box was compared to the ground truth bounding boxes of the image. The cropped image's label was set to the ground truth bounding boxes label that was closest to it.

Given a set of cropped traffic light images and their color labels, the dataset was split into a training and validation set. This dataset was used to train a ResNet model. A pretrained ResNet18 model was used. The last fully connected layer of the model was replaced to have four outputs, representing red, yellow, green, and other. The ResNet model was retrained to finetune the model to classify traffic lights.

4. Experiments and Results

We used Pytorch to implement our model, and used a pretrained FasterRCNN model with a Resnet-50 backbone. We wrote a custom dataset for the DriveU dataset, where we output the images, bounding boxes and classifcation labels for input into Faster RCNN. For further classification of the traffic lights, we use a Resnet variant to determine the color of the traffic light. From the Faster RCNN output, we crop out the bounded traffic light, and create a dataset from this, similar to how the Cats and Dogs dataset was structured. From there, we load a Resnet-50 model, pretrained on ImageNet, and set to finetune to the new classes, Red, Yellow, Green, and Other.

We ran into a few limitations regarding our implementation. Firstly, the dataset generated from the RCNN output was limited in size - normally much larger datasets would be used, even for finetuning. Secondly, we noticed that RCNN would find traffic lights that weren't facing the camera. In other words, we wouldn't be able to find the state of these traffic lights from the perspective we were given. In order to deal with this, we decided to have the "Other" class.

Referring to Table 1, we achieve relatively good class accuracy for Red and Green lights, hovering around 73% and 76.6% respectively. For yellow, we found that none of the validation yellow lights were classified correctly. After inspecting the data by hand, we found that 1. there were very few instances of Yellow lights in the training set and 2. those that were labeled Yellow were often mislabeled pieces of data.

In terms of training, we ran it for relatively few epochs, mostly to avoid over fitting to the training set, since the generated dataset was small. We find that the both the training loss and validation loss drop quickly and hit a floor after 6 epochs, as seen in Figure 4.

5. Conclusion

Through this project, we learned how to work with bounding boxes in a machine learning model and how to trim a general RCNN model to fit our needs. The RCNN was generally able locate the traffic lights in the larger image, even though the traffic lights were relatively small. Generating the training data from the RCNN was a challenge as matching bounding boxes to labels may not have been perfect. Since the training data may have had errors, those errors could have affected the model. Future recommendations for this project would be to try using a Single-Shot-Detector and compare the accuracy of that model to this model. Using a SSD, we could train it to generate bounding boxes and classifications. This would eliminate the need for two machine learning models Also, another extension of this project would be to identify other traffic light symbols such as turn lights.

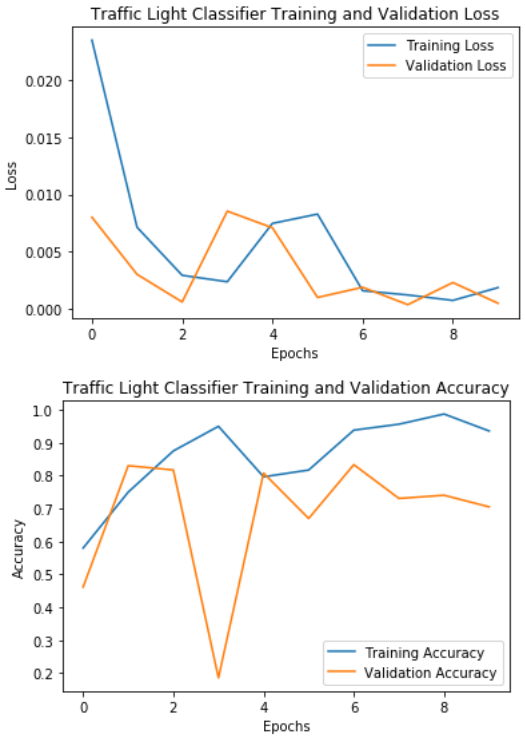

Figure 4: Loss and Accuracy of Modified Resnet Network during Training

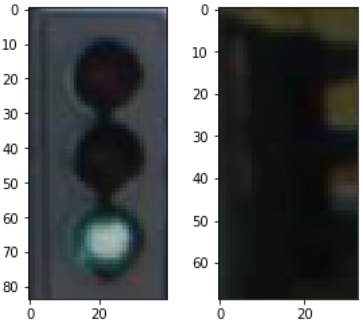

Figure 5: Classification examples. The image on the left was successfully classified as a green light bu the image on the right was unsuccessfully classified as red instead of other

Light Color	Total Images	Correct Predictions
Red	37	27
Yellow	23	0
Green	252	193

Table 1: Validation results for each light color.

References

[1] The DriveU Traffic Light Dataset: Introduction and Comparison with Existing Datasets, A. Fregin and J. Muller and U. Krebel and K. Dietmayer, 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018,

[2] Traffic Light Detection, Swati Bhartiya, Rochester Institute of Technology, 2016

[3] Detecting Traffic Lights by Single Shot Detection J. Muller and K. Dietmayer, 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018