Project Stop Lights:
Solving The Rolling Stop

by John Finberg and Sameer Sinha

Introduction

On campus, there are many streets with stop signs. It is common to see cars creep up to intersections and then proceed to go through them without ever stopping. This practice frustrated my group, and we wanted to create an automated solution to keep track of this type of event.

As a solution, we decided to build an image classification model that, when given a live stream of our intersection, will provide bounding boxes for our crosswalks. We will then have another model frame-by-frame detecting vehicles in the frame. We will keep track of these vehicles and ensure they stop before the identified crosswalks.

Methodology

We broke our research project into two parts. The first is creating a model to recognize crosswalks. This allows us to understand where vehicles are in the intersection. The second is training a model to recognize and track cars through the intersection

Crosswalk Model

For our data set, we choose a synthetically created collection of images from an autonomous driving software Carla. The data set had 20k images distributed among night, day, different weather conditions, and yellow or white crosswalks. Our data came in the form of PNG images and an annotation.JSON file written in COCO annotation style. We adapted a script to convert the annotation.JSON file to an XML file per image. We then resized our images to 640x640 and converted them to JPG files. We then split our data randomly into a train, test, and validation set. We loaded our annotated data set into Tensorflow's Object Detection API. Using a pre-trained Faster RCNN ResNet101 V1 640x640 model, we trained it for 50k epochs with a batch size of 4. We then exported our model and tested its accuracy.

After training our two models, we put them together. We integrated our YoloV5 model into a Strong Sort algorithm. Then we initialized the crosswalk model before calling the YoloV5. This provides us with the bounds of the crosswalk. We wrote a class to track the vehicle's position, given by the YoloV5 model. Then using the crosswalk position and car position, we calculate the distance from the nearest crosswalk. We calculate the distance to the crosswalk as a scale factor of how large the bounding box is. We check if the car is within a radius and if it holds a space for two seconds. Then we mark the vehicle as stopped.

Vehicle Detection Model

We trained our vehicle detection model using a couple different mathods. We first used a YoloV5 framework to attempt to train the model to recognize the bounding boxes. We used a Roboflow dataset with 930 labeled images. This model worked really well on the Roboflow data's testing set, and exported these weights in a PyTorch file to use in our larger model; these weights would be used in the videos to identify images. However, this YoloV5 model only detects cars in an individual frame; we would need to add a strong sort element to ensure that the model recognizes that a car is the same car between frames.

Joint Application

While we were not able to gather direct statistical data on the performance of our joint model application, we ran a series of tests on prerecorded videos, checking that as vehicles stopped or did not stop, the application marked them off appropriately.

Results

Crosswalk Model

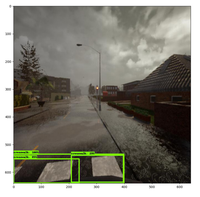
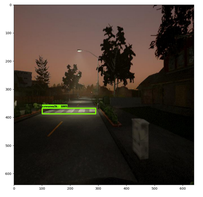
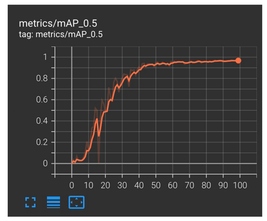
Below is the evaluation results from the testing of our SSD MobileNet V2 FPNLite 640x640 and annotated testing images.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.007
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.039
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.015
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.018
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.084
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.239
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.269
```



Below are the evaluation results of testing and annotated testing images of our RCNN ResNet101 V1 640x640.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.003
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.006
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.004
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.001
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.006
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.034
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.070
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.125
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.067
```

Joint Application

From the visual testing of our application, we had maximum success with the following hyperparameters. A margin of error of 10, stop time of 1 second, and scale factor multiplier of 3.5 times the square root of the bounding box size.

Vehicle Detection Model

Our mean average precision (mAP) ended up at 96.9% on our testing set, and our box loss got as low as 0.0246. Images of detections and mAP graph seen above.

Discussion

Crosswalk Model

Originally, we trained our model on the SSD MobileNet V2 FPNLite 640x640 model. This model scored a higher mAP score, but when we view the resulting annotated images, they are inaccurate. It appears the model has been over-trained on the dataset and does not apply well outside of the train set. The RCNN ResNet101 V1 640x640 model performs accurately on images with one crosswalk towards the middle of the frame. The model does not perform well when crosswalks are at different angles. The model also does not perform well when objects are blocking the crosswalks. Image one shows that as the crosswalk's orientation changes, it has a harder time detecting bounds. Image two shows that when the axis is aligned, the model performs well. However, it sometimes recognizes single stripes as secondary crosswalks. These observations can be traced back to our data set. The data set is images of intersections from the perspective of a car. Thus the crosswalks are either parallel or perpendicular. There are a collection of images in the data set where the car sits on the crosswalk, and only particular stripes are annotated. This leads to the model finding secondary crosswalks within bounding boxes. Adding more annotations and rotating images may increase the accuracy of the mode. While the RCNN ResNet101 V1 640x640 has a lower mAP score, it applies better outside of the train data set.

Joint Application

While we were not able to gather direct statistical data, we were able to see the model making detections of vehicles stopping. We struggled at first to adapt the distance to the crosswalk for vehicles that were farther away, as the same amount of pixels represented more distance. Thus, we developed a system to take the size of the bounding box into account.

Vehicle Detection Model

While the model we trained worked successfully, we opted for a pre-trained model.