Patrick Oline

Capstone Final Report:
Custom Object Detector

Problem Statement

Use existing models to try and improve detection of specific green street signs. Given a pretrained model, I would like to try and use transfer learning to train the model to detect a class that it does not already know, green street signs. Using this idea of transfer learning, once the process is figured out it can be transferred over to detect any object. Having the process to develop a custom detector can be helpful in any field of computer vision from self-driving cars, to text detection, to sports analytics.

I will first inspect the YOLOv5 model and the basics behind its standard model. I will then collect and annotate a large group of images for the training process. Then, I will use those annotated images and transfer learning with YOLOv5 to create new weights for the model that will help it better detect only green street signs.

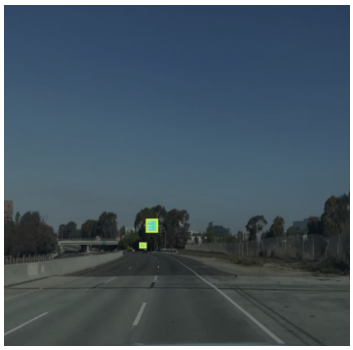
In the end I hope to have a model that can detect only green street signs and nothing else. I will try for the model to identify at least 90% of the green street signs it comes across.

Data Wrangling

I will be using a set of images that I have had to collect. Some of the images were pulled from the internet, but I noticed it was easier to get frames from videos. This led me to writing a few functions which allowed me to select a portion of a video, and capture the frames as image files. In the end I had a mix of 120 different images which had green street signs. The mix included images where the greens signs were very large, very small, and everywhere between.

Exploratory Data Analysis and Pre-Processing

There was not much work in the form of exploratory data analysis for this object recognition project. Once I had the photos I wanted to use for training, testing, and validation, I had to annotate the green signs in each of the images. To annotate, I uploaded my photos to Roboflow, which is a website where you can upload your photo data set, and once the photos are uploaded they can then be annotated. Once annotated, Roboflow splits the images into train, test, and validation set, and allows you to export all the data to use for training.



Model Training

The training of this object detection model proved to be a rather difficult process. Since my goal was to use transfer learning on an already existing model, I was not starting from scratch in terms of parameter development. It would have seemed that this would make the process a much easier one. However, this did not turn out to be the case.

I ran into an assortment of issues during my model's creation. The biggest roadblock was with the python libraries and their dependencies. I created several different virtual environments, but was not successful in getting Tensorflow to successfully run. I researched my problem, and discovered that my Mac OS required a special installation of Tensorflow. I was able to successfully go through the process of installing and running Tensorflow, however when I tried to train my model, tensorflow kept killing my notebook kernel.

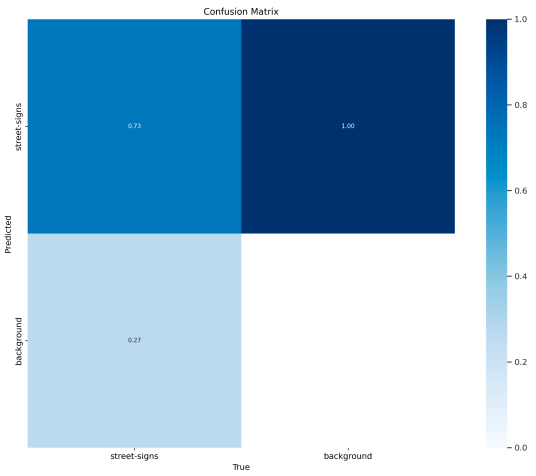
It was at this point I decided to abandon Tensorflow and instead tried to use the Pytorch library for training my model. A little progress was made, but when it came to training the model, I again ran into problems where the Pytorch library tried to incorporate tensorboard and the process again failed.

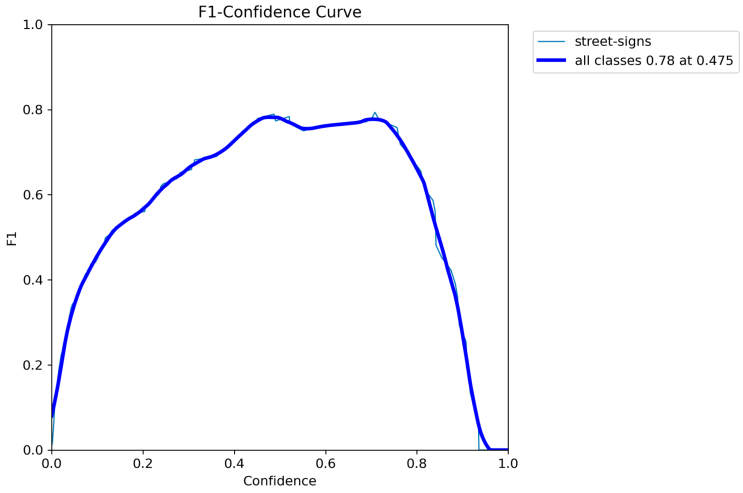
It was at this point that I decided to try and train my model using a cloud platform rather than locally. I found a tutorial on how to train a custom model using Google Colab, and I customize the process for my own needs, using my own data. This process was successful. I was able to finally train and test my model. In the training (84% of pictures), validation (12% of pictures), testing (4% of pictures) process, I used batches of 80 images over 500 epochs.

Model Results

Listed below are a couple of pictures I ran through my image detection model. While the bounding boxes are not perfect, they are successful in detecting green street signs both far and near with high levels of confidence. I was even able to run my models on a video, and the model was able to detect all green street signs with confidence levels as high as 0.97. In the video however, there were a few frames that had incorrect detections, but making the confidence threshold higher eliminated those misdetections.

Having a look at some of the charts can give more insight into the model. The confusion matrix shows that the model correctly predicted street signs with a 73% success rate, and only a 27% rate of predicting background when it was actually street sign. The F1-Confidence curve shows us that at the confidence value of 0.475, the F1 score is at its maximum of 0.78. This F1 score is a fairly good value, while it can be improved, it is good for a first model.





Future Research

There are many things that I would like to continue to work on with this project. The first thing I would like to do is try to improve the proper detection of the green street signs. While the model does a good job of detecting the street signs, the bounding box accuracy could certainly be improved. This would require the time consuming process of finding new data, annotating those new images, and training my current model on those new images. Continuing to train and develop the current model will not only increase the confidence values, but it will also help eliminate the occasional incorrect detections.

Once the model is more accurate, the next step I would like to take is to replace the bounding boxes with actual blank green and white rectangles to simulate the green signs. Eventually, I would like to develop a letter detection algorithm, that would detect the letters in the frame that are on the street sign and recreate the letters on the blank green street sign. In the end the pipeline might look something like: 1) detect sign, 2) detect letters on sign, 3) recreate street sign with green background and white letters, 4) display the recreated sign for users to more easily read.

Finally, the best part about this project is that I can take this process and apply it to any other object I wish to detect. In order to train another object detector, I would just simply have to gather new pictures, annotate them and rerun the training/ testing/ validation script. From this process I will have a new custom object detection model.