FrostAV: Lane & Sign Detecting Tenth-Scale Vehicle

Lewis Collum

Contents

1 YOLOv3 Real-Time Sign Detection

1.1 Results

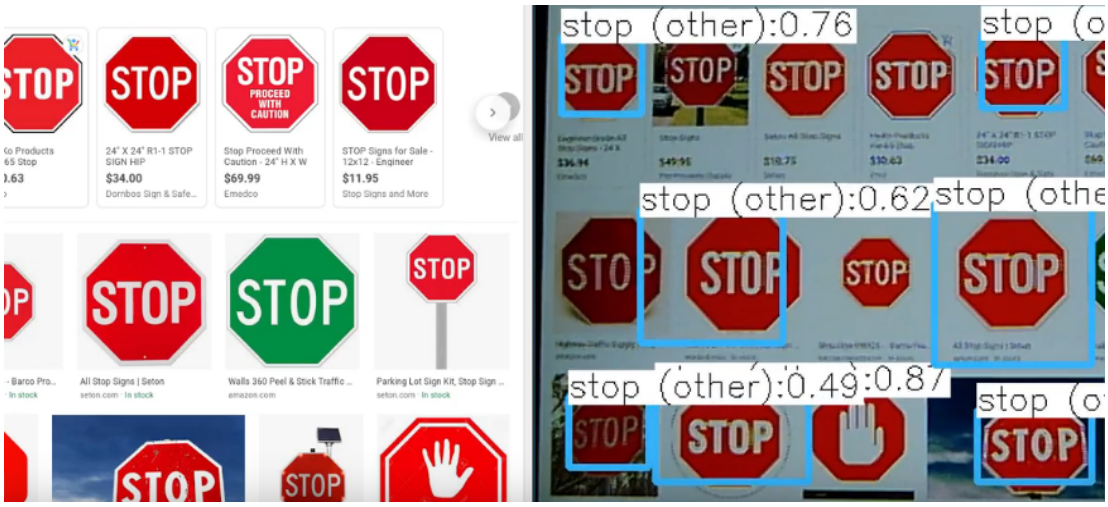

Fig. 1. Stop sign images on a monitor (left), and the processed video feed (right).

1.2 How YOLOv3 was Implemented

1.3 Easy. Detect Signs Then Classify.

No. This is how object detection[1] algorithms started (e.g. R-CNN), but they can be too slow for real-time (and embedded) object detection. Those looking for speed, use algorithms that extract classified objects from the frame in a single pass, as opposed to two passes. YOLOv3 is an algorithm that detects in a single pass.

There are multiple versions of YOLOv3. We are using YOLOv3-tiny-prn, which has the highest frames per second (FPS) compared to other commonly used algorithms (see figure 2). The sacrifice to speed is accuracy, as YOLOv3-tiny-prn borders around 35% average precision. Since we implemented sign detection on a Raspberry Pi as a proof-of-concept, this accuracy is acceptable.

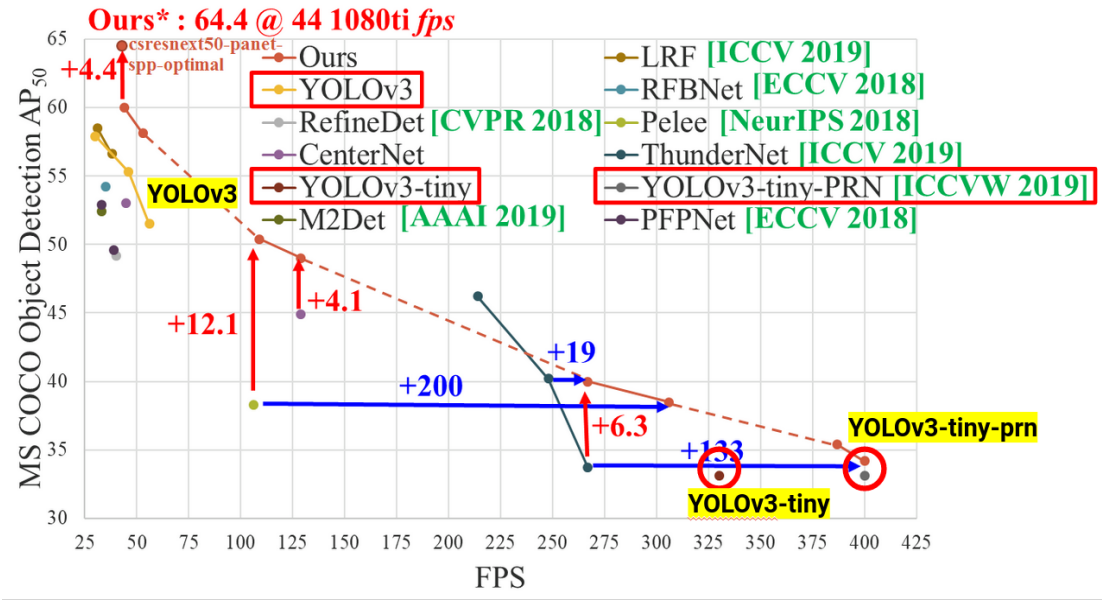

Fig. 2. YOLOv3-tiny-prn has the highest FPS, with an acceptable 35% average precision.

2 Lane Detection

3 Vehicle Monitoring with a Web Server

3.1 Installing Nginx, Flask, and GUnicorn

Since we are using Arch Linux ARM on our Pi, we used Arch Linux's package manager `pacman` for Nginx and Gunicorn. We used `pip` to install Flask.

```
pacman -S nginx
pacman -S gunicorn
pip install flask
```

These commands have to be run as sudo, since we are a user on the Pi.

1. Classification is not detection. Objects first need to be detected before they can be classified. But, for briefness, we say "object detection" when we really mean "object detection and classification."

3.2 Starting the Server the First time

For ease, this repository was cloned on our Raspberry Pi to the user directory (e.g. "/home/alarm"). In the `frostServer/config` directory, there is a setup script which does the following:

1) links the frostServer.service to the system directory;
2) links the frostServer.nginx config file to nginx's enabled-sites directory;
3) enables both the nginx and the frostServer services;
4) starts both services.

For this script to work, it should be run as `sudo`.

Now, on a computer connected to the same WiFi as our Pi, we can type the IP of the Pi into the browser search bar (e.g. "192.168.0.106") and the Frost website appears.

Since we enabled the server service, the server will start at boot without any user intervention.

3.3 Server Overview: Flask, GUnicorn, and Nginx

We use Flask, GUnicorn, and Nginx. Let's work our way down, from the website, to our application code (in python).

Nginx acts as our web server, fulfilling requests from clients for static content from our website (e.g. HTML pages, files and images). Nginx forwards dynamic requests (e.g. all of the requests that we want Python to handle) to GUnicorn, our application server.

GUnicorn ensures that Nginx and our Flask Python application can talk to each other. Ultimately, GUnicorn routes requests (passed through Nginx) to their corresponding function in our Flask application.

Flask is a web microframework in the form of a Python library. It is used in our Python application to provide functions that can receive requests, and return a response (e.g. sensor data).

3.3.1 Example: Real-Time Sensor Reading

We have real-time plots on our front-end that request sensor data. Eventually, we expect to receive this sensor data from a URL of our choice (e.g. "/sensor_reading"). These requests are sent to Nginx, which are then passed to GUnicorn (since they are dynamic requests). GUnicorn sends the requests to a function in our python application that is registered for the formerly mentioned URL. The function is registered with the given URL using Flask. This function has code to get sensor data from the underlying linux system (typically, by reading a sensor file or running a shell command). Finally, the function returns a response which contains the sensor data.

3.3.2 Resources

- https://en.wikipedia.org/wiki/Multitier_architecture
- https://serverfault.com/questions/331256/why-do-i-need-nginx-and-something-like-gunicorn
- https://www.nginx.com/resources/glossary/application-server-vs-web-server/
- https://flask.palletsprojects.com/en/1.1.x/api/#flask.Flask.route