Drone Obstacle Detection Using YOLOv5

Vihaan Bhaduri - vihaan.bhaduri@gmail.com - 408-807-6878
Saratoga High School - 10th Grade

Overview:

Object detection is a technique used in computer vision to find objects in a picture or video. My project was built with the goal of performing live object detection on a drone. In order to achieve this, I had to collect images, label them, and train a functional model. I collected my dataset using the DJI Mavic Mini Drone and labeled the images on Roboflow [1], a computer vision data labeling website. I used the polygon tool to perform instance segmentation. In order to train my model, I resorted to transfer learning on an expert model YOLOv5 with pre-trained weights. As stated in PyTorch's article on the YOLOv5 [2] model, "YOLOv5 is designed to be fast, accurate, and easy to use, making it an excellent choice for a wide range of object detection, instance segmentation, and image classification tasks."

Approach:

Dataset:

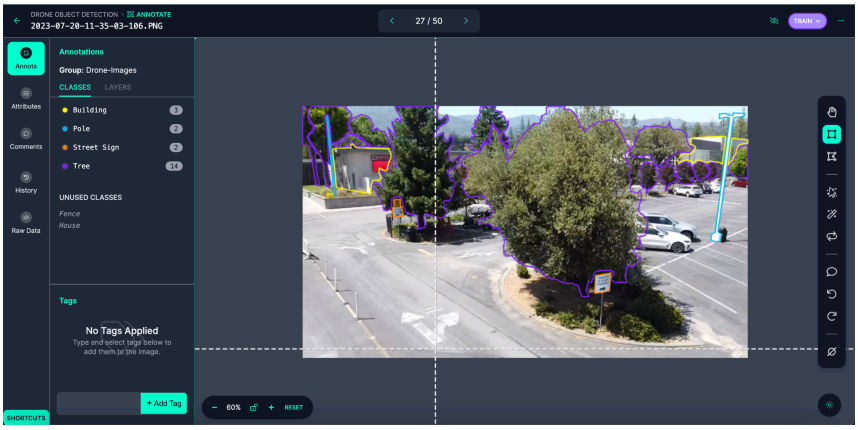
Collecting Images:

One of the biggest obstacles I dealt with during the process of building this model was the lack of focused datasets that contained the images I was looking for. After searching through Kaggle and other public dataset websites, I was left without any usable images. Out of the few datasets [3, 4] that provided drone images, the pictures I found were typically fully focused on a top-down perspective (bird's eye/nadir view) rather than an oblique or horizontal field of view, which would help train the model to assist with lower altitude maneuverability that is essential in the process of safe navigation in crowded regions. The datasets also contained a greater extent of classes than I intended to work with. To deal with this surplus, I resorted to creating my own dataset focused on 6 different classes: buildings, houses, fences, poles, street signs, and trees. Based on these goals, I picked 3 locations to take pictures: Brandywine Drive, Saratoga High School, and downtown Los Gatos. I flew my drone at a varying range of altitudes from near-ground-level images to aerial views, around 5-30 meters off the ground. I

also altered the angle of the gimbal from looking straight down (90°) to horizontal views at around (10°).

Labeling Images:

As aforementioned, I labeled my images on Roboflow. I entered 228 .PNG images onto the website after downloading them from my drone. Throughout the process of labeling, I had to keep the goal of my project in mind: real-time obstacle detection. There was no use of labeling every minute object since they did not directly affect the flight path of the drone. Hence, I did not label many trees/buildings/poles/etc. in the background for specific images, where I did not find it necessary to do so. However, my model was still able to detect and label the necessary objects (Read: Model Analysis Using Images: Model Analysis Using Images).



(Example of Instance Segmentation)

Training:

Without Data Augmentation:

Early on in my experimentation, I quickly understood that I lacked a sufficient number of images in my dataset to create a functional and respectable model. Data augmentation is a technique used to increase the size of a training set by artificially altering aspects of an image in order to create additional copies. I ran around 10 experiments, which I later deleted, using an unaugmented dataset. As shown in my first experiment [5], I was only able to achieve the following accuracies:

Buildings	Fences	Houses	Poles	Street Signs	Trees
40-%	33%	12%	23%	0%	48%

This model was clearly unsatisfactory because it was unable to detect enough obstacles.

With Data Augmentation:

Augmentation Categories:

Data augmentation has been key to the success I have had throughout this project. Labeling was a lengthy process that required manual labor, and a quick way to easily upscale the size of my dataset was through augmentation. I used Roboflow's threefold (3x) augmentation, which provided a variety of different techniques for me to choose from. I ended up settling on 4 types of image-level augmentation: noise, exposure, brightness, and blur. I also played around with augmentation that was solely focused on the bounding boxes in my images. During these tests, I removed normal exposure and brightness augmentations and added bounding box level exposure and brightness augmentations. I observed minor differences with this change, but image-level augmentations created slightly better models than bounding box-level augmentations.

Parameters/Variations of Training:

I constantly tweaked the parameters upon the four categories (Noise, Exposure, Brightness, Blur), settling upon the following values as ideal:

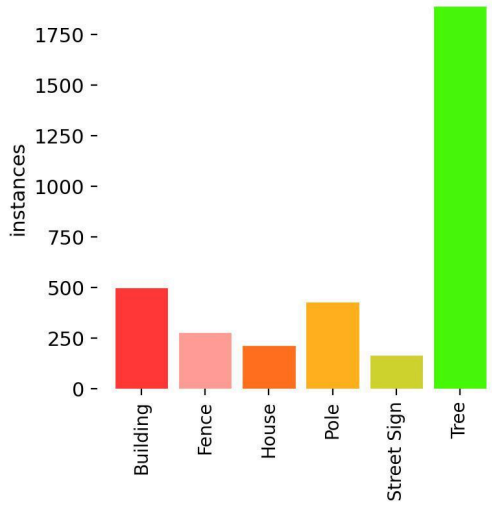
Noise	Brightness	Exposure	Blur
Up to 5%	-25% → 25%	-17% → 17%	Up to 1.5 pixels

I also found that experiment 6 provided the best weights to use for training without overfitting the model. Through these augmented datasets, I was able to build models that improved the results.

Additionally, I experimented with removing classes in my datasets. Due to the disparity in the amount of labels I had for each class, street signs and poles did not receive a sufficient amount of attention during the model's training. This resulted in poor results and missed labels, as can be seen by the consistently disappointing results for the two classes. My street sign

class was only able to manage 27% accuracy, often unable to record any correct labels (0%). My pole class was able to achieve a 45% accuracy at best, normally placing between the mid 20%'s to mid 30%'s. I hypothesized that by removing these classes, my model would be able to focus more on the other four classes (buildings, fences, houses, trees).

As shown in the models that were trained upon these datasets (See rows with N/A in [5]), removing classes surprisingly did not provide much of a difference, and my hypothesis was incorrect. I resorted to training upon all 6 classes. Currently, the latest label disparity is heavily weighted towards the tree class (See image below).



(labels.jpg File - Experiment 80)

Analysis/Results:

Best Model:

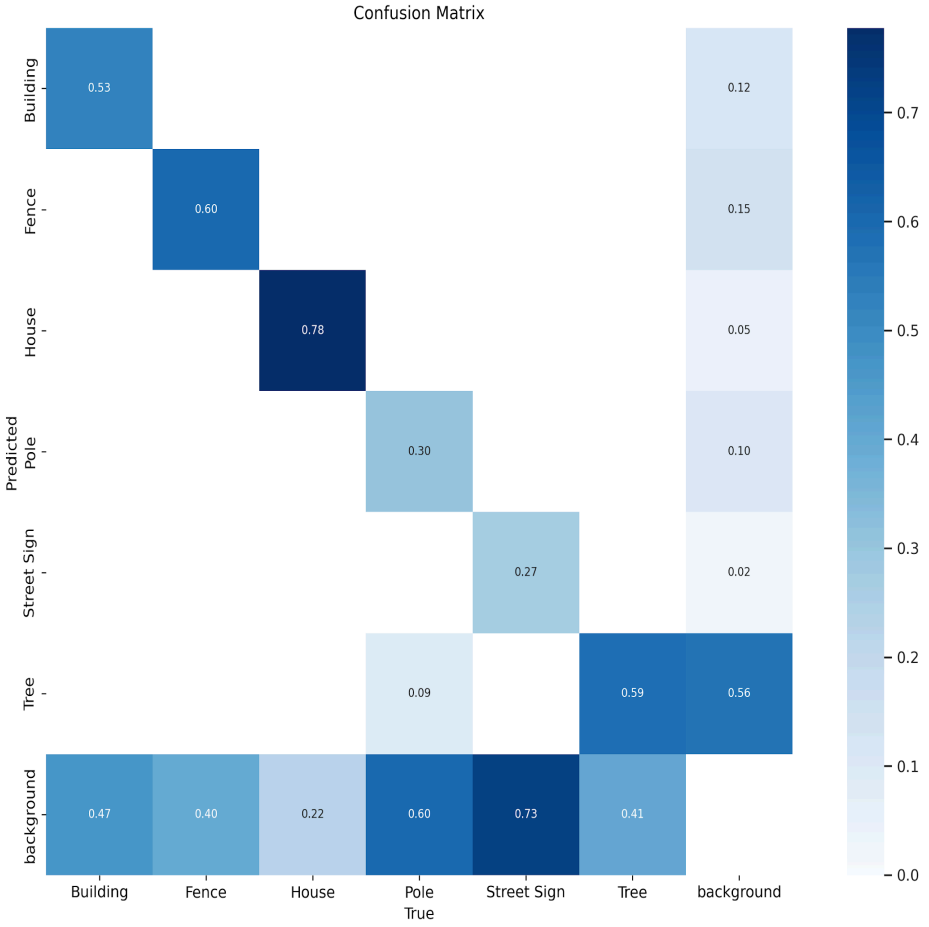
Although it was difficult to achieve a model that was capable of performing well on all classes, there were some that did better than others.

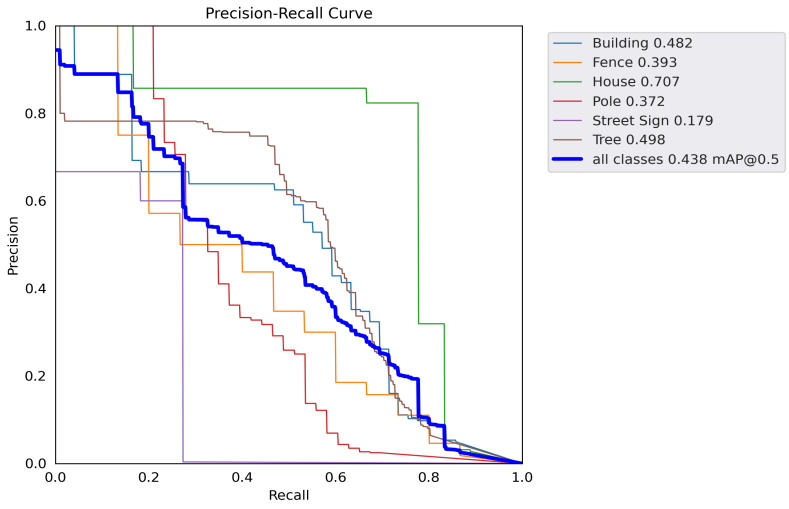
Top 2 Models (With All 6 Classes) - Training Accuracy:

1.) Experiment 74:

Buildings	Fences	Houses	Poles	Street Signs	Trees

53%	60%	78%	30%	27%	59%

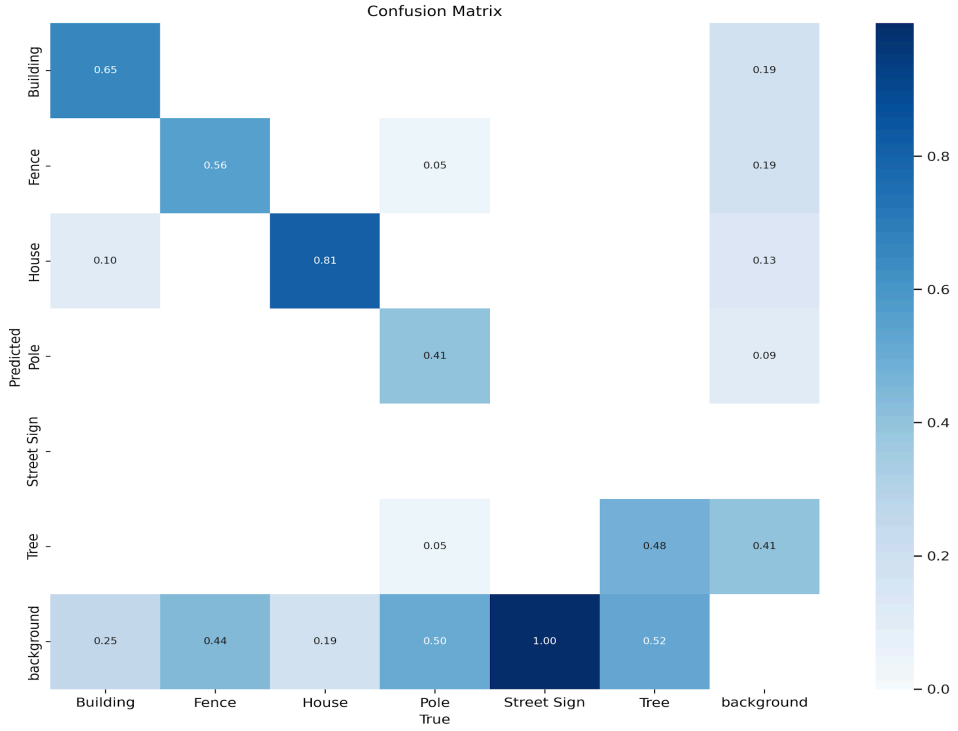


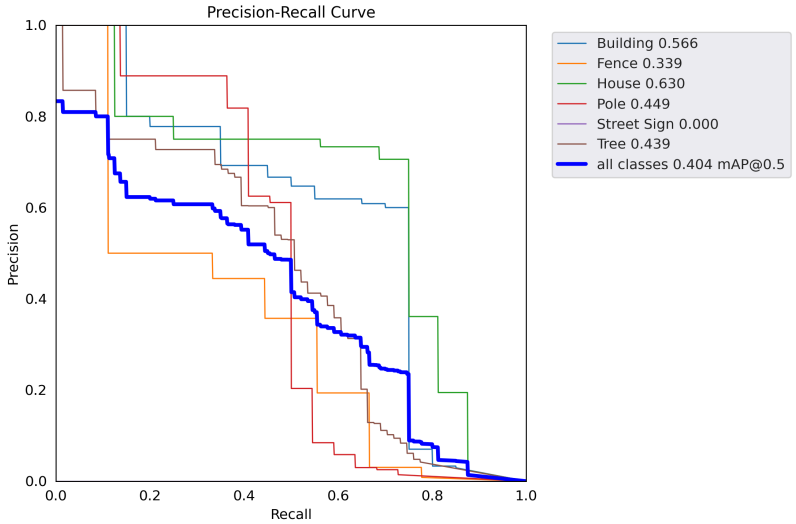


(Confusion Matrix/P-R Curve for Experiment 74)

2.) Experiment 2:

Buildings	Fences	Houses	Poles	Street Signs	Trees
65%	56%	81%	41%	0%	48%





(Confusion Matrix for Experiment 2)

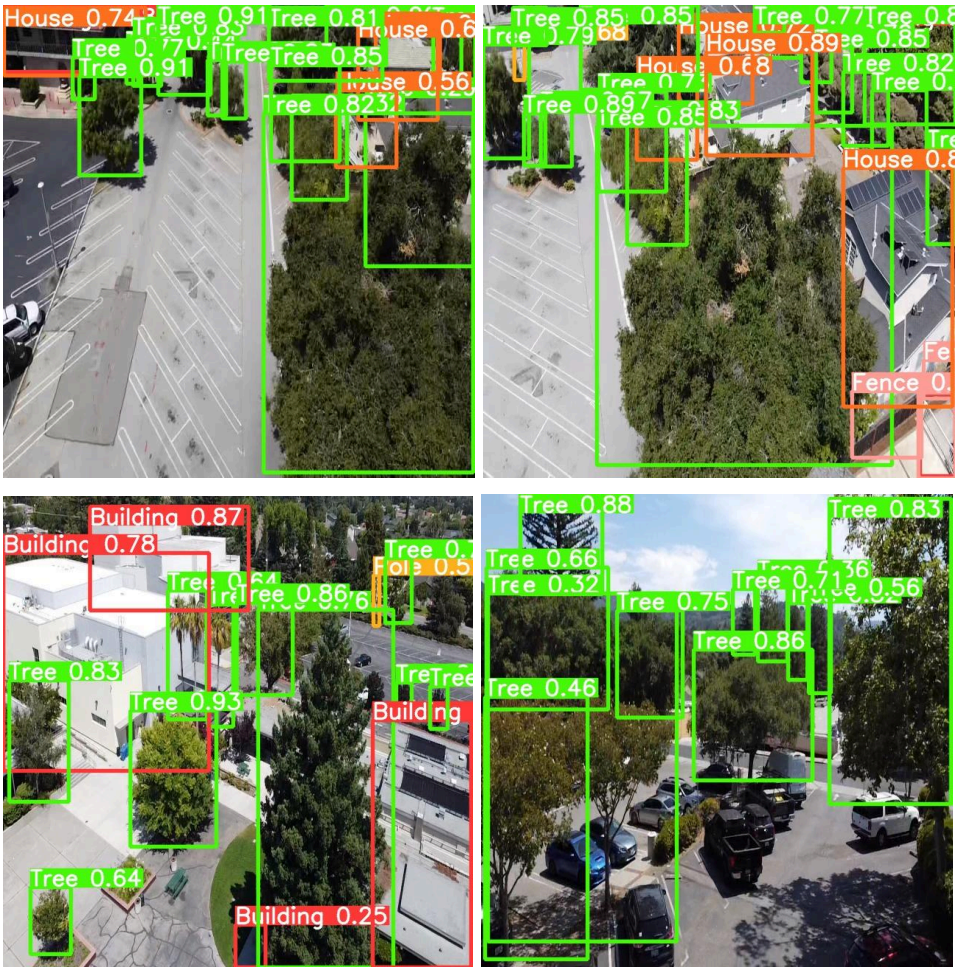
Model Analysis Using Images:

All Classes:

Images Used in Analysis:

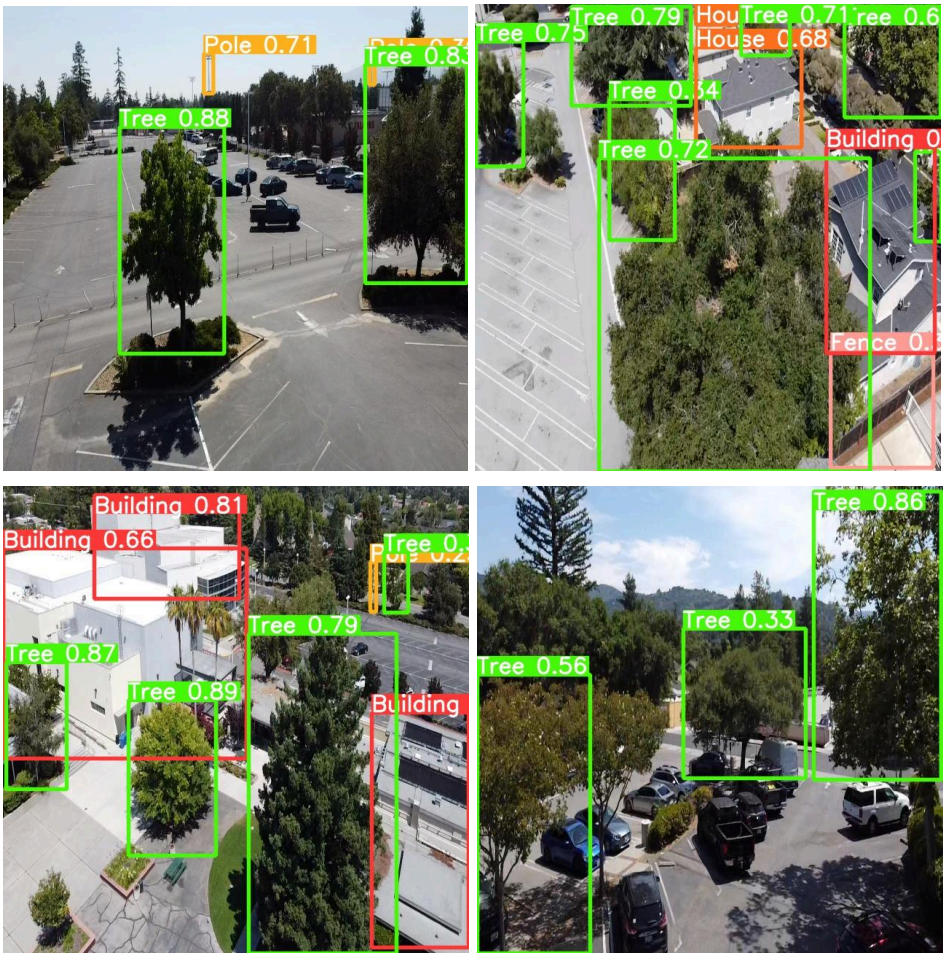
Top left image: 2023-07-23-14-46-46-500.PNG
Top right image: 2023-07-23-14-46-52-975.PNG
Bottom left image: 2023-07-20-11-31-04-928.PNG
Bottom right image: 2023-07-23-14-43-28-338.PNG

Experiment 74 Detect.py:



As demonstrated by these images, the model performed adequately on images in the test set. Although the confusion matrix shows mediocre accuracy percentages, it is able to accurately label objects with high confidence and accurate bounding boxes.

Experiment 2 Detect.py:



Albeit the high training accuracies, this model performed poorly on the test set. It is not able to pick up on the majority of objects. It also has lower confidences than model 74. This shows evident overtraining, clearly separating the model in experiment 74 as the best all class model.

Effects of Using Augmentation in My Dataset:

Augmentation was a catalyst for the rapid improvement in the accuracy of my object classes.

It was able to increase the precision of my classification, as the model in experiment 74 was rarely misclassified. Additionally, this model was able to detect objects with high confidence, achieving the baseline goal of my project.

Conclusion/Next Steps:

Throughout this project, I was able to successfully build an obstacle detection model on drone images. I achieved my goal of detecting obstacles in the foreground, and I did so with high confidence and precise classification thanks to the power of data augmentation. In the near future, I will look to increase the size of my dataset and balance the label count in order to improve my model and address the skew in my dataset. I will focus on taking class-specified images. Additionally, I intend to open-source my current dataset along with my best model on Kaggle and Github to help others explore the relatively unexplored domain of drone AI.

References:

[1]: https://app.roboflow.com/drone-obstacle-detection/drone-object-detection-yhpn6/deploy/23
[2]: https://pytorch.org/hub/ultralytics_yolov5/
[3]: https://www.kaggle.com/datasets/bulentsiyah/semantic-drone-dataset?rvi=1
[4]: https://www.kaggle.com/datasets/nunenuh/semantic-drone
[5]: Drone Object Detection Model Experiment Results