Object Detection Using YOLOv5 and VisDrone2019 Dataset

Karl Simon

May 6, 2023

Contents

1 Introduction

As part of a current research project, I am trying to fly a quadcopter autonomously where the drone's position is determined by extracting coordinates from images taken by a mounted monocular camera. In order to accomplish this, the computer on the quadcopter needs to be able to accurately locate targets in the real world. Once we know the target, and we have calibrated the camera, we can calculate the position of the quadcopter and feed these coordinates to the flight control algorithm.

As this is the first time I have dealt with an object detection task, I was interested in exploring existing machine learning techniques that address this challenge of taking image data (specifically that from a single camera mounted on a quadcopter) and predicting where in the image a desired target appears.

1.1 Background

Object detection is not a new task, and many algorithms have already been proposed and developed. Typically, these algorithms make use of convolutional neural networks, or CNNs, and popular examples include Single Shot Detection (SSD), Faster Region based Convolutional Neural Networks (Faster R-CNN), and You Only Look Once (YOLO)[3]. Of these different approaches, YOLO seemed to be the best for my application because unlike the sliding window and region proposal-based techniques which the other alrogithms use, YOLO makes predictions of bounding boxes and class probabilities all at once [2]. This simplified pipeline allows YOLO to be much faster than other models, which for real-time image processing is very important. Additionally, the kernel weights in the convolutional layers in the YOLO model are pretrained, so the time needed to train the model on a custom dataset is significantly reduced. Finally, since 2015 when it was introduced, there have been various versions of YOLO. For this project, I decided to use YOLOv5 because it is well documented and has been proven to be very robust, even if it is slightly slower than it's most up-to-date vertion (YOLOv8).

1.2 YOLOv5 Model Overview

The YOLOv5 model is implemented as a convolutional neural network, and consists of 24 convolutional layers followed by 2 fully connected layers. For a given input image, the algorithm divides the image into a grid of cells, and each cell predicts B bounding boxes and confidence scores, which is calculated as $Pr(Object) * IOU_{pred}^{truth}$[2]. Each cell also predicts conditional class probabilities, which is the probability of an object in a cell being a certain class i. It then uses the formula $Pr(Class_i|Object) * Pr(Object) * IOU_{pred}^{truth} = Pr(Class_i) * IOU_{pred}^{truth}$ to determine the class-specific confidence scores for each box in a cell [2].

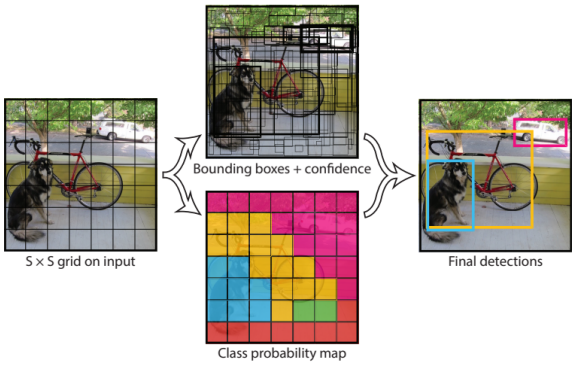

Figure 1: The YOLO Model

The network then uses the convolutional layers to extract features, and the connected layers to predict the probabilities and coordinates. The loss function that the model optimizes is the following:

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)
\end{aligned}
$$

Figure 2: YOLOv5 Loss Function

where $\mathbb{1}_{ij}^{obj}$ denotes that the jth bounding box is the predictor for the ith cell [2].

2 Project Design and Dataset

As a first step in my project, I needed to decide which dataset to use. Specifically, I looked for datasets which contained images taken by drones in interesting environments, and whose objects were already annotated with bounding box coordinates (to save time). Finding a complete dataset was quite challenging, as some were either too large (upwards of 100GB), or didn't have complete annotations. Ultimately, I came across a dataset assembled by a group at Tianjin University in China called VisDrone2019, which included videos and still frames, along with annotations. The benefit of this dataset is that it combines images of common traffic scenes from different angles above the ground, different times of day, and different scales, which means the model could generalize better to test images used from other datasets. To train the the model on this dataset, I used a Google Colab Tesla T4 GPU, and let it run for 10 epochs. Altogether, preprocessing and training took about 4 hours. Once complete, I used the trained weights of the model to detect test images from the dataset, as well as images from the web and other datasets.

3 Data

Figure 3: Upper Left: ground view, only one truck detected. Upper Right: slight motion blur not affecting classifications. Bottom Left: incorrect label. Bottom Right: obstructed objects correctly detected.

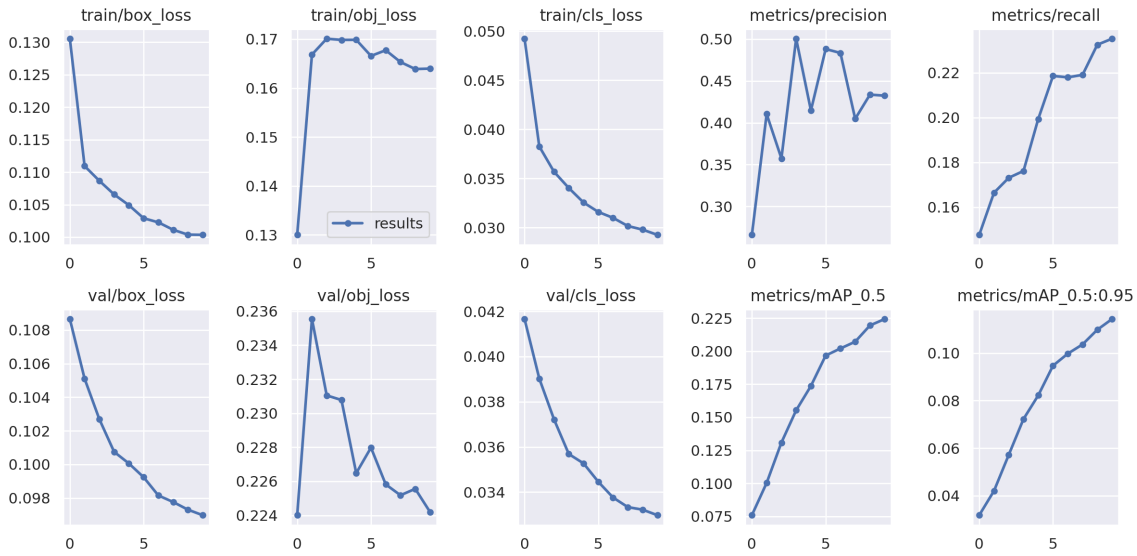

Figure 4: Results. Each increment on the x-axis corresponds to one training epoch.

4 Evaluation

While we can clearly observe from the test results whether an object was correctly detected or not by inspecting the bounding box on the object, we can instead use a value called mAP, or Mean Average Precision. In object detection models, there are two important metrics that determine how accurate a model is on a given dataset; precision and recall; precision refers to the ratio of true positives to all positives, while recall refers to true positives out of all correct predictions. In other words, precision measures the proportion of positive classifications that were actually correct, while recall measures the proportion of true positives out of all actual positives [4]. The details for the predictions for each class can be viewed in Figure 5.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

In Equation (1) above, AP refers to the average precision of each class i, summed over all N classes. Specifically in object detection tasks, the determination of true positive or false positive is measured by the IoU threshold, or Intersection over Threshold. IoU corresponds to how much the ground truth and predicted bounding boxed overlap, and the threshold is the percentage of overlap that is used to determine the classification of a prediction [1].

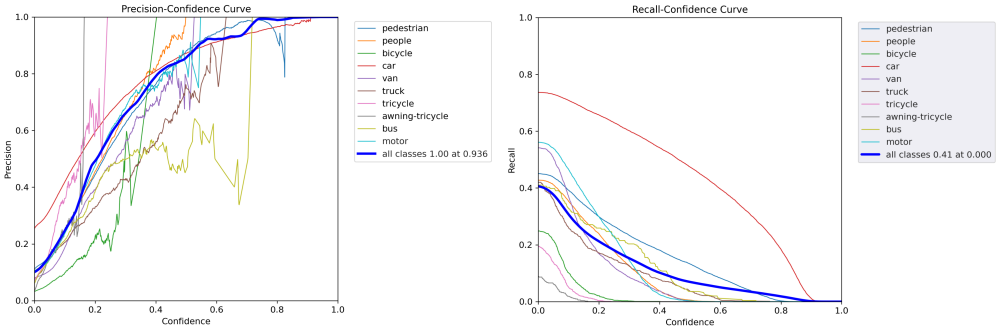

Figure 5: Precision and Recall Plots

Overall, with the model having to tune over 7 million parameters, the model performed quite well. The mAP metrics increased, which corresponds to more correct positive classifications, and losses decreased, which means IoU error decreased. It should be noted that ideally, the model would be

trained for many more epochs to get the most optimal performance. For instance, in viewing the MAP with IoU threshold of 0.5 (bottom right graph in Figure 4), the curve didn't yet flatten out, so the model was not overfitting yet and results could have been improved.

As a test, I wanted to see how the model trained on the VisDrone2019 dataset would respond to images from other datasets. In Figure 3, the bottom left image is from the Stanford Drone Dataset, which features pedestrians and cyclists. Because the image was from overhead looking directly down, street markings were occasionally misclassified as busses and cars. And although the VisDrone2019 dataset included images such as this from this angle, and bicycles were included as potential classes, there must not enough of these kinds of images in the dataset. Thus, this showed how the generalizability of a model really depends on the training data it is given.

5 Conclusions and Future Work

The purpose of this project was to get a first look with how object detection models are constructed and used, specifically using the YOLOv5 model. Although much of the actual implementation of the model was treated as a black box, it gave good insight into how models are evaluated, how to prepare and feed data into the model, and how they can be improved.

An interesting area of future work could be to perform the same task using another model(s), such as R-CNN, and compare it to the YOLOv5 performance. Then the model with the best performance and speed can be adapted for a real-time application. Next, we can use these results along with OpenCV to be able to control a robot based on a camera's image. By calculating the position of the camera relative to the frame of the object being detected, we can determine where the robot is in space, and thus be able to control it.

What this project has shown me is that object detection algorithms are already quite advanced, so an interesting area of study could be how to use this information for new tasks such as object tracking or surveillance.

References

[1] Kiran Krishnan Bonymol Baby Dr. K S Angel Viji Abhinu C G, Aswin P. Multiple object tracking using deep learning with yolo v5. *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH TECHNOLOGY (IJERT)*, 09 – Issue 13, 2021.

[2] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[3] Divekar A.V. Anilkumar C. et al. Srivastava, S. Comparative analysis of deep learning image detection algorithms. *J Big Data 8, 66*, 09 – Issue 13, 2021.

[4] Fadhlan Zaman, Syahrul Abdullah, Noorfadzli Razak, Juliana Johari, Idnin Pasya, and Khairil Kassim. Visual-based motorcycle detection using you only look once (yolo) deep network. *IOP Conference Series: Materials Science and Engineering*, 1051:012004, 02 2021.