Object Tracking for Intelligent Vehicles in Palo Alto

CS231N Spring 2022 Final Project: Shanduojiao Jiang, Yan Wang, Fangran Wang

Stanford
Computer Science

Introduction

Recently, the surge in the number of intelligent vehicles has revolutionized people's everyday commute. Some of the most fundamental problems for such vehicles include lane detection, semantic segmentation, and object tracking. In this project, we will investigate object tracking for its importance in ADS (autonomous driving systems), and its intricacy of real-time scene understanding and video data input. With an attempt to better serve the Stanford/Palo Alto community, we are specifically interested in developing our models using real-life driving scene data collected in Palo Alto.

Problem Statement

We divided our project into 3 phases:

1. Object Detection with Existing Dataset:
   We used the Udacity Self Driving Car dataset to train and compare different YOLO model variants.
2. Custom Dataset Collection and Annotation:
   We collected 251 real-life driving scene images taken in Palo Alto using both iPhone and dashcams, and then we annotated the images with Roboflow.
3. Transfer Learning on Custom Dataset:
   we performed transfer learning using YOLOv5 with weights pre-trained on COCO dataset, and reran the model on the Palo Alto driving video mentioned in Phase 1.

The input to our models are images with objects such as cars and pedestrians, and the output of our model would be the predicted bounding boxes and class probabilities of the objects.

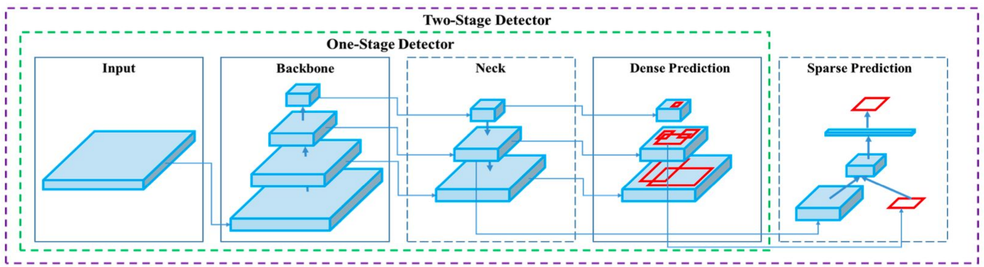

Figure 1. YOLO Model Architecture

The YOLO model architecture contains 3 important components:

1. Backbone: is used as an feature extractor for an given input image
2. Neck: is used to generate feature pyramids that help the model to generalize well on unseen data
3. Head: is used to perform the final detection which includes class probabilities, objectness scores, and bounding boxes

Dataset

Palo Alto Driving Video
One 20-second (603 frames) driving video taken in Palo Alto for testing the object tracking performance of our models.

Udacity Self Driving Dataset
Existing dataset from Udacity with 15,000 images (with size 1280 x 1280) with labels across 11 classes.

Palo Alto Custom Dataset
251 images taken manually using iPhone and dashcam, and annotated with Roboflow.

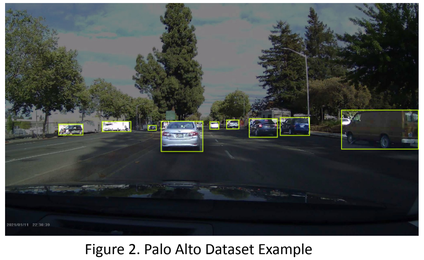

Figure 2. Palo Alto Dataset Example

Methods

YOLO Model Architecture
The main model that we use in this project is YOLOv5. The model structure is shown above.

YOLO Variants
We trained and tested 5 YOLOv5 variants: YOLOv5s, YOLOv5m, YOLOv5l, and 2 of our own custom YOLO models:

	Small	Medium	Large	Custom1	Custom2
Depth	0.33	0.67	1.0	1.0	1.4
Layer	0.50	0.75	1.0	1.0	1.3
Optimizer	SGD	SGD	SGD	Adam	Adam

Figure 3. Difference between YOLO Model Variants

Transfer Learning
To make our model specific to Palo Alto dataset, we performed transfer learning using model initialized with weights pre-trained on COCO dataset.

Model Name	AP@.5	AP@[.5:.95]
YOLOv5s	0.755	0.423
YOLOv5m	0.765	0.446
YOLOv5l	0.776	0.455
custom1	0.775	0.447
custom2	0.740	0.407
Transfer	0.886	0.579

Figure 4. Results of YOLO Model Variants

Results & Analysis

Evaluation Metrics
To measure the object detectors quantitatively, we use Average Precision (AP) for each class and mean Average Precision (mAP). Specifically, AP@.5 means the average precision with the IoU(Intersection over Union) threshold equals 0.5. AP@[.5:.95] means the average AP over different IoU thresholds, from 0.5 to 0.95 with a step of 0.05.

Comparison between YOLO Model Variants
Figure 4 shows the results of YOLO Model Variants. We find that:

1. mAP gets slightly better as the model becomes larger and more complicated.
2. YOLOv5m takes twice the time to train as YOLOv5s, and YOLOv5l takes four times to train as YOLOv5s.
3. The two custom models are slightly worse than the 3 baseline models, and we believe this is because Adam optimizes best when it is fine-tuned.

Performance Boost after Transfer Learning

Figure 5. 5 Predicted Sample Frames

We saw a significant improvement in model performance as we applied transfer learning using our dataset (Fig. 4). We also chose 5 frames from our driving video to compare the object tracking performance (Fig.5). The transfer learning model is able to detect cars with a much higher confidence score.

Conclusion

Although there isn't a significant difference in performance for different YOLO model variants on the existing Udacity Self Driving Dataset, we have seen an evident improvement of model performance on the Palo Alto Driving Video once we performed transfer learning with our custom dataset. In the future, we would like to collect more driving videos in Palo Alto to include different and complicated scenarios. We would also like to detect and track other objects such as pedestrians, bikers, and traffic lights. Finally, we would like to explore SORT and DeepSORT to connect the objects in each frame to form a continuous result instead of stitching the predicted frames together.