Object Tracking for Intelligent Vehicles in Palo Alto
CS231N Spring 2022 Final Project: Shanduojiao Jiang, Yan Wang, Fangran Wang

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 model variants on the existing Udacity Self Driving Dataset, and reran the model on the Palo Alto dataset.

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.

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:

<table>
<thead>
<tr>
<th>Model Name</th>
<th>AP@0.5</th>
<th>AP@[.5:.95]</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOLOv5s</td>
<td>0.755</td>
<td>0.423</td>
</tr>
<tr>
<td>YOLOv5m</td>
<td>0.765</td>
<td>0.446</td>
</tr>
<tr>
<td>YOLOv5l</td>
<td>0.776</td>
<td>0.455</td>
</tr>
<tr>
<td>custom1</td>
<td>0.775</td>
<td>0.447</td>
</tr>
<tr>
<td>custom2</td>
<td>0.740</td>
<td>0.407</td>
</tr>
<tr>
<td>Transfer</td>
<td>0.886</td>
<td>0.579</td>
</tr>
</tbody>
</table>

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.

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
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.