CPE 313 - Final Project

Object Detection-based Vehicle classification and Speed Estimation

Dela Cruz, Gabrielle E. & Vista, Jens Liam P.

Targeted Sustainable Development Goals (SDG)

Better road monitoring leading to safer driving conditions, reduces injuries from road traffic accidents.

Addition to traffic monitoring strategies, improvement of road safety, and better transport system.

Problem

The number of road fatalities and accidents is increasing because there are no traffic enforcers to monitor and apprehend speeding vehicles.

Solution

Develop a computer-vision system powered by machine learning that measures vehicle speed in areas where no traffic enforcer is present.

Object Detection for Vehicle Detection (Introduction)

Object Segmentation

Object Detection

In this paper, the team has deployed models tailored for object detection in the classification of moving vehicle types. This is preferred to avoid heavy taxing of resources and longer training times compared to object segmentation.

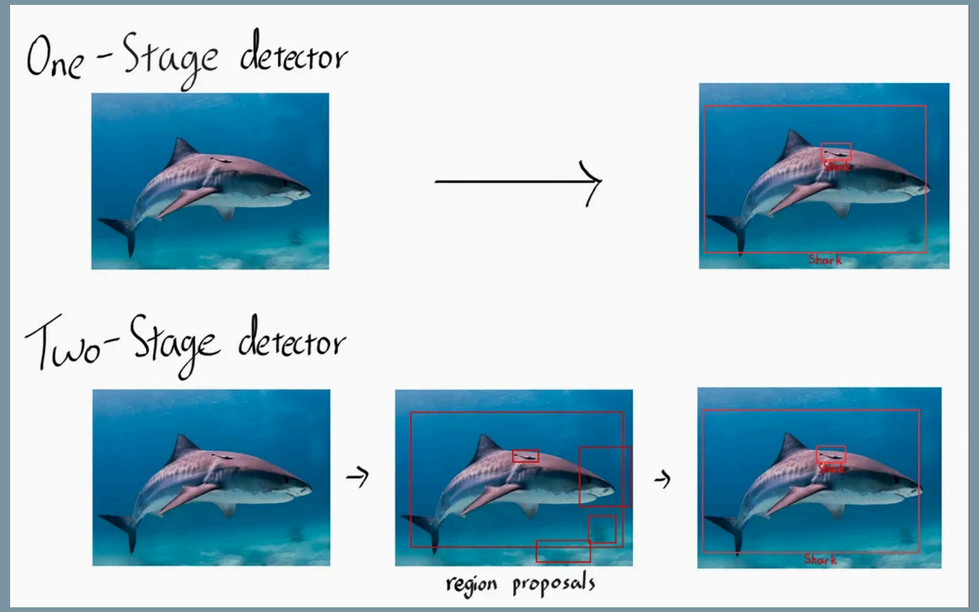
Deployment of One-stage and Two-stage Detectors (Methods)

One-stage Detectors Used:
YOLOv11 and SSDLite (Single Shot MultiBox Detector) with MobileNetv3

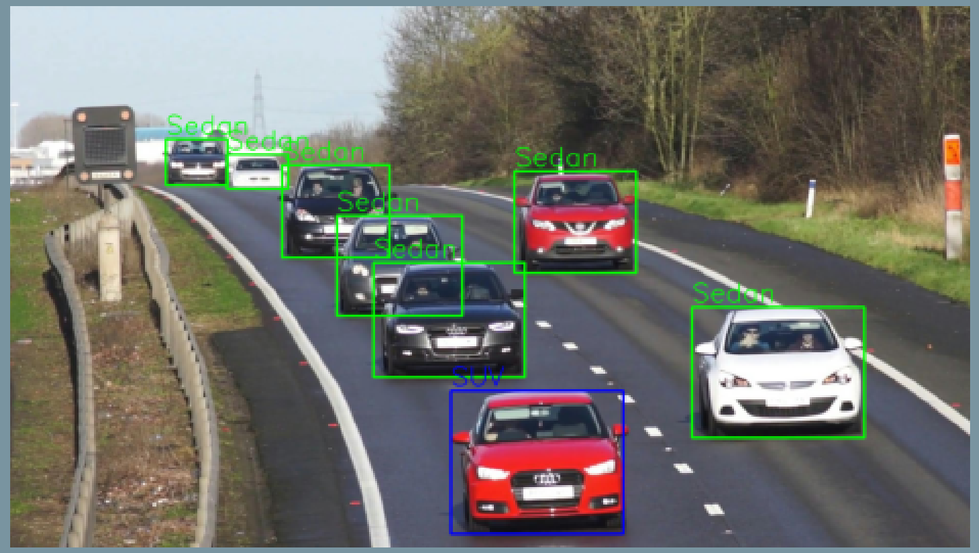
Two-stage Detector Used:
Faster R-CNN with ResNet50:

Notes:

- Use of RoboFlow for automatic labeling (description of classes)
- Installing packages and importing modules by RoboFlow and Ultralytics for YOLO models.
- Dataset consisting of 2.5k+ images of Sedan and SUV.
- Determining model performances through COCO Evaluation.



Two-Stage Detectors generate region proposals for better accuracy but slower due to its two-step processing.



Model Evaluation - YOLOv11 Nano

YOLOv11 Model Evaluation (at 3 epochs, 640x640)

At 100 layers, 2,582,542 parameters, 0 gradients, 6.3 GFLOPs

	Instances	True Positives	Precision	Recall	mAP@0.50	mAP@0.5:0.95
All	420	419	0.925	0.915	0.971	0.908
Sedan	212	212	0.895	0.923	0.969	0.896
SUV	207	207	0.954	0.907	0.974	0.920

- **Chosen model to be deployed in Streamlit.**
- **Trained over 3 epochs with image size at 640x640 pixels.**
- **Easy implementation but demonstrates impressive performance.**
- **Builds on earlier iterations, YOLOv8 and YOLOv5.**
- **According to Bochkovskiy et al. (2020), models of the YOLO suite offer excellent trade-offs between speed and accuracy [1].**

[1] Bochokovskiy, A., Wang, C., & Liao, H.M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv:2004.10934. https://doi.org/10.48550/arXiv.2004.10934

Model Evaluation - Faster R-CNN vs SSDLite

Faster R-CNN ResNet50 vs SSDLite MobileNetV3 Performance Comparison

Metrics	Faster R-CNN (ResNet50)	SSDLite (MobileNetV3)
AP@ [0.50:0.95] (mAP)	0.936	0.883
AP@0.50	0.984	0.981
AP@0.75	0.983	0.948
AP (Large Objects)	0.936	0.883
AR (Large Objects)	0.957	0.909
AR@1 (Recall, maxDets=1)	0.957	0.907
AR@10 (Recall, maxDets=10)	0.957	0.908
AR@100 (Recall, maxDets=100)	0.957	0.909
COCO-style mAP	0.9356	0.8826

- A comparison between One-stage vs Two-stage Detectors.
- Used COCO Evaluation by JSON file instead of YAML (for YOLO).
- Minor differences:

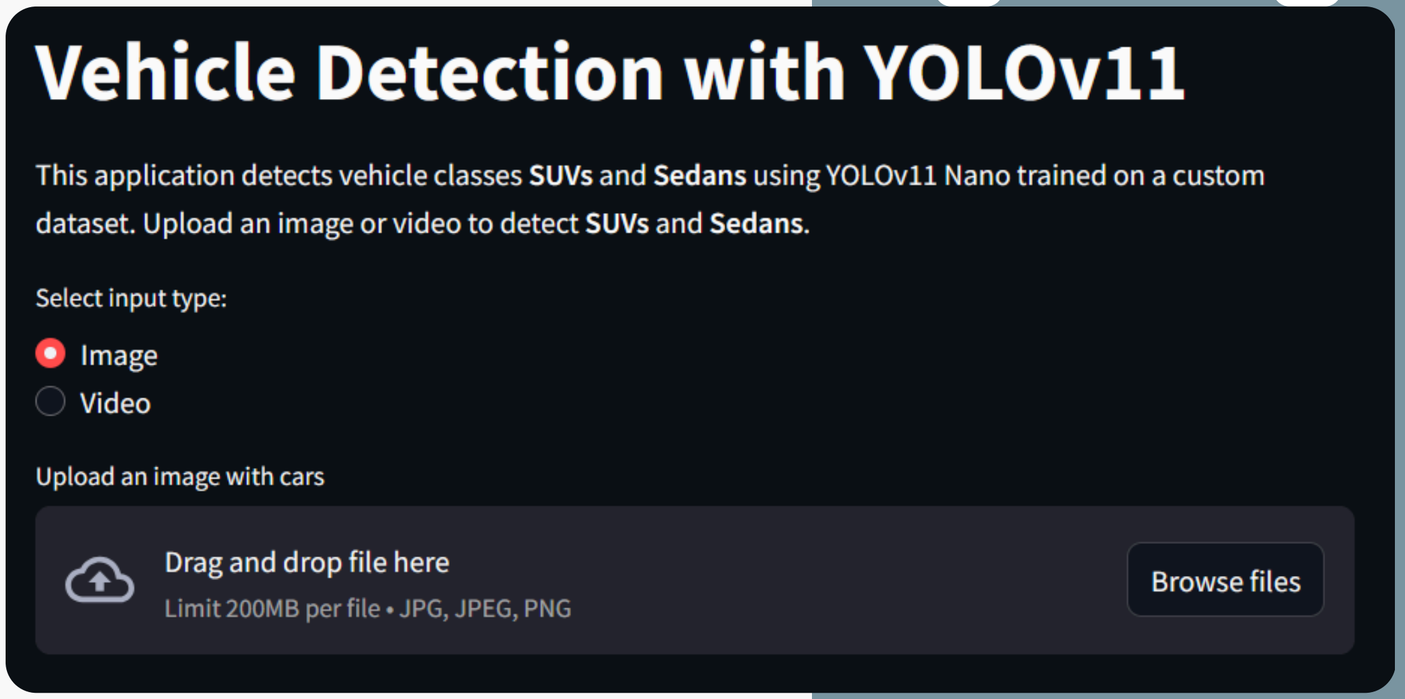
1. Faster R-CNN with ResNet50 performs better but takes more time and more computationally expensive.
2. SSDLite with MobileNetV3 is best recommended for lower-end, smaller systems.

Deployment on Streamlit (Component 1)

- **Uploading of necessary files:**

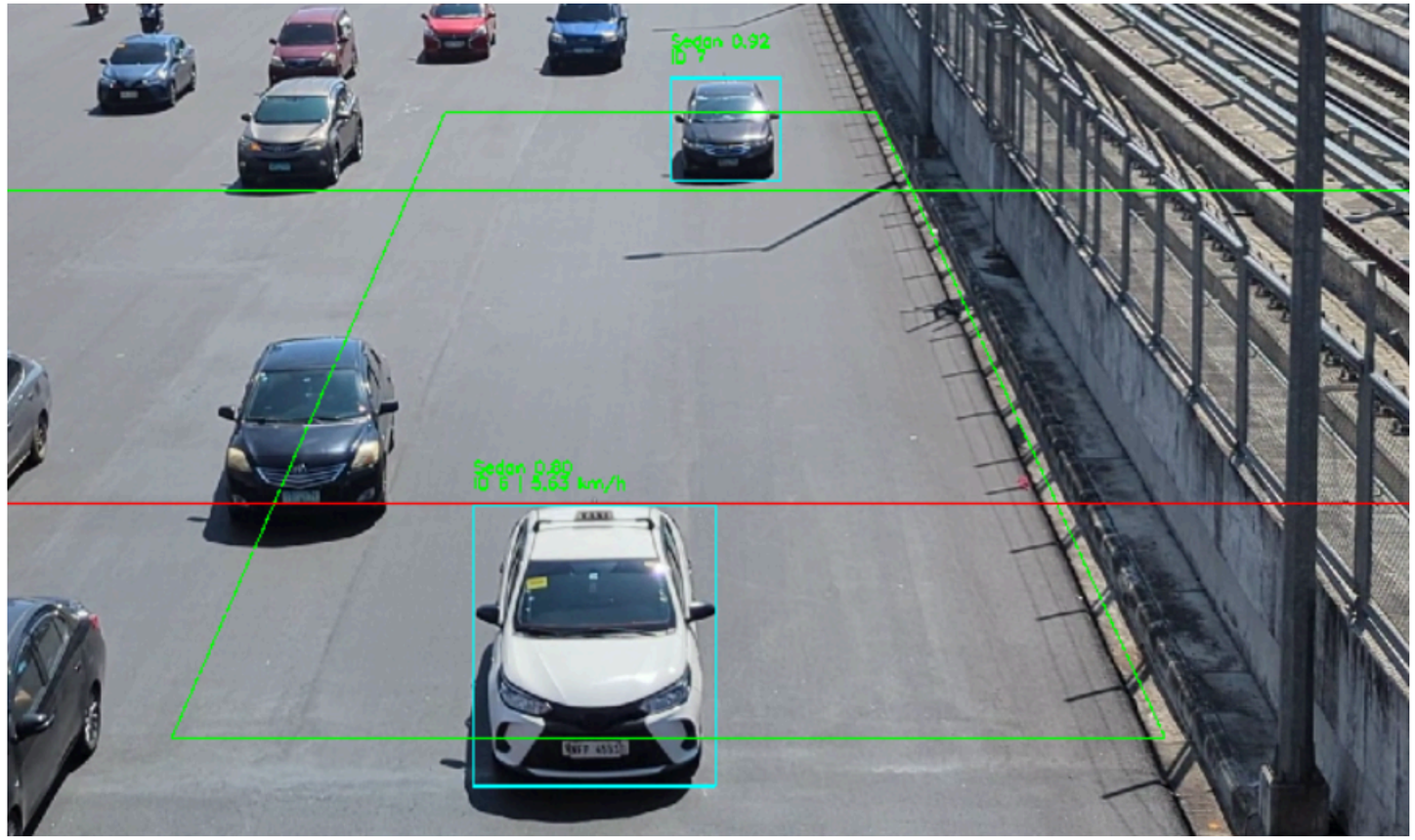
1. Python Source File (app.py)
2. Best-trained YOLOv11 model (best.pt)
3. Instructions for dependencies (requirements.txt)
4. Additional OpenCV requirements (packages.txt)

- **Takes Image and Video Inputs.**
- **Produces Downloadable Outputs.**



Link towards the Streamlit Application:
https://cpe313-final-project-nxtumxvdtrxl8ut8upakje.streamlit.app/

Speed estimation logic



Using the trained classification model

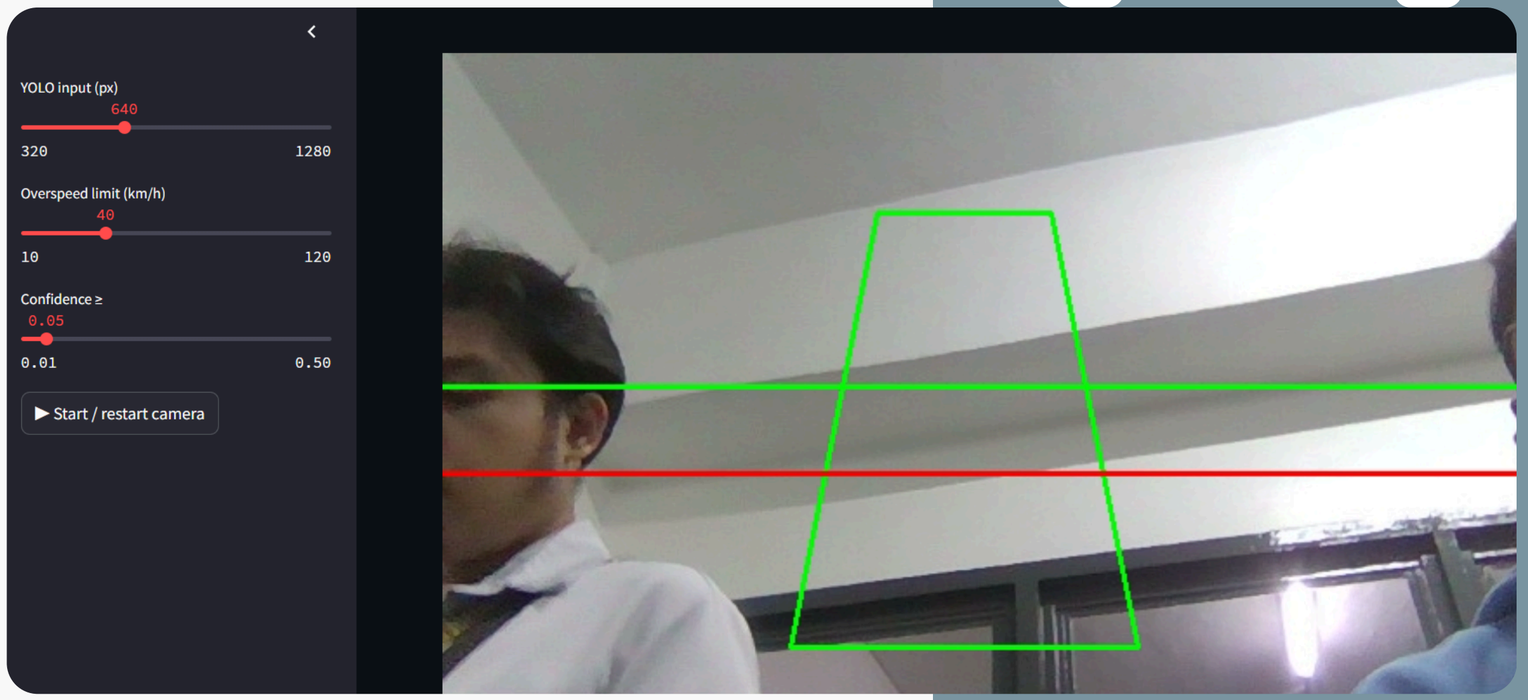
1. Uses two horizontal lines namely point A and point B
2. Uses polygon zone to sort objects to be detected
3. Speed = distance / time

Deployment on Streamlit (Component 2)

- **Uploading of necessary files:**

1. Python Source File (app.py)
2. Best-trained YOLOv11 model (best.pt)
3. Instructions for dependencies (requirements.txt)
4. Additional OpenCV requirements (packages.txt)

- **Process real time inputs and detection**



Link towards the Streamlit Application:
https://cpe313-final-project-nxtumxvdtrxl8ut8upakje.streamlit.app/