Vehicle Detection: Capstone Project

Requirement

Identify **all vehicles** in an image — including **buses**, **cars**, and **trucks** — and **draw bounding boxes** around them.

Problem Overview

Detecting vehicles on the road — and more critically, identifying their type — is a challenging task. It's akin to the complexity of distinguishing between a **cat** and a **dog** in an image. Once you can do that, you're halfway to solving the vehicle detection problem.

There are many **pre-trained models** like **YOLO (You Only Look Once)** and **SSD (Single Shot Detector)** that can detect and classify vehicles efficiently, often in **as few as 10 lines of code**.

However, our capstone project has a different requirement:

> ***Build a vehicle detection model from scratch.***

This is significantly more complex. Differentiating between a **car and a traffic signal** isn't trivial.

The Real Challenge

Let's break it down:

Problem 1:

You can detect if an image contains a **cat or dog**. But...

> *Can you detect **how many cats** there are?*

That's a different level of difficulty. It requires **object localization** — not just classification.

Problem 2:

Assuming you can identify a cat or dog, the next step is to **draw a bounding box** around each instance.

This involves:

- **Edge detection**
- **Object localization**
- Drawing **accurate bounding boxes**

Solution

While building your own model, you can still **train it using pre-trained architectures** like YOLO or SSD by preparing your own dataset.

Steps:

1. **Prepare your dataset** (images with vehicles).
2. **Create annotations** — for **each image**, you need an **annotation file**. This is a **1:1 mapping**.

Annotation Details:

Annotations are metadata that describe:

- All vehicles in an image
- Their **coordinates** (x1, y1, x2, y2)
- **Bounding box size** (height, width)
- **Positions** (top-left, center, etc.)

Annotation Formats:

- `.txt` or `.csv` for YOLO
- `.xml` for SSD

How to Create Annotations

1. **Manual labeling** (most accurate)
2. **Labeling tools** like `LabelImg`
    - Upload one image at a time
    - Draw bounding boxes manually
3. **Automated scripts**
4. Use **YOLO pre-trained models** to auto-label your dataset

> ***Note:*** *No tool can perfectly auto-annotate. Manual labeling still gives the best results.*

Final Steps

Once you have **1:1 annotations**, you are **70% done**.

Then, use **Keras** and **TensorFlow** to:

- Build your own object detection model
- Train it on your annotated dataset