Multi-Class Object Detection

Submitted by: Dhruv Kamboj  
Roll Number: 102303645  
Course: COE

# Introduction

# Object detection is a critical task in computer vision that involves identifying and classifying objects within digital images or video streams. It plays a pivotal role in applications such as surveillance, autonomous vehicles, robotics, medical image analysis, and augmented reality. In the context of this project, the goal is to develop an object detection system that can analyze a video stream, detect multiple classes of objects, and provide real-time annotations (bounding boxes and object counts) on each frame.

# The challenge lies in the accurate detection and classification of objects under varying conditions such as changes in lighting, occlusion, scale, and object orientation. This project demonstrates how powerful object detection algorithms, leveraging deep learning models, can be used to address these challenges efficiently. Through a combination of deep learning and traditional computer vision techniques, this project aims to detect multiple classes of objects in real-time and present this data in an understandable format.

# The system developed in this project uses a general object detection pipeline with a pre-trained deep learning model, focusing on detecting classes such as people, vehicles, animals, and other common categories, with each frame being processed and annotated accordingly.

# Overview of Object Detection in Computer Vision

Object detection is one of the most fundamental tasks in computer vision and has evolved rapidly over the last decade, particularly with the advent of deep learning. There are two primary stages in object detection:

1. **Object Localization**: Identifying where an object is in the image or video. This is typically done by drawing a bounding box around the object.
2. **Object Classification**: Assigning a label to the detected object (e.g., person, bicycle, dog, etc.).

In recent years, deep learning has drastically improved the performance of object detection. Models like **Faster R-CNN**, **YOLO (You Only Look Once)**, and **SSD (Single Shot MultiBox Detector)** are widely used in real-time object detection tasks. Each of these models has strengths and weaknesses:

* **Faster R-CNN**: Offers high accuracy but can be slow for real-time applications due to its region proposal network.
* **YOLO**: A single-pass detector that is incredibly fast and accurate, making it ideal for real-time applications like video surveillance or self-driving cars.
* **SSD**: Similar to YOLO but typically provides a balance between accuracy and speed, particularly in multi-scale detection scenarios.

For this project, we leverage a general object detection pipeline based on deep learning models (specifically the YOLOv5 variant) to detect and classify multiple objects in each frame of the video. This approach allows the system to achieve high detection accuracy and fast processing speeds suitable for real-time applications.

# Tools and Libraries Used

The project leverages a range of tools from the **computer vision** and **deep learning** domains to implement and execute the object detection pipeline. The key libraries and technologies used are:

* **Python**: The programming language for developing the detection system, as it provides easy integration with machine learning and computer vision libraries.
* **OpenCV**: A widely used open-source library for computer vision tasks. OpenCV helps with video frame capture, pre-processing, post-processing, and drawing annotations (bounding boxes, labels).
* **Ultralytics/YOLOv5**: For real-time object detection, YOLOv5 is used to perform the object classification and bounding box generation. However, this can be swapped with other models like Faster R-CNN or SSD based on the requirements.
* **COCO Dataset**: The model used for detection has been pre-trained on the **COCO dataset**, which includes 80 object categories commonly found in everyday environments. This ensures that the model is capable of detecting a wide variety of objects such as people, animals, vehicles, and everyday items.

**Project Objective**

The primary goal of this project is to build a real-time multi-class object detection system that processes a video stream, detects various objects, counts the occurrences of each class, and displays these results with real-time annotations. The system must:

* **Detect objects** in a given video.
* **Classify objects** into predefined categories (e.g., people, vehicles, animals).
* **Count occurrences** of each object class across frames.
* **Annotate video frames** with bounding boxes and class labels.
* **Display results** in real-time while the video is being processed.

The overall goal is to create a robust, efficient, and easy-to-understand framework that can handle the detection of multiple objects from different classes in video data, even in challenging real-time environments.

**Dataset & Model**

For this project, the **COCO dataset** was used as the foundational dataset for the object detection model. This dataset contains images annotated with 80 object categories, ranging from common items like 'person', 'bicycle', 'car', 'dog', 'cat', to more specific items like 'frisbee', 'baseball bat', and 'backpack'. These object categories are relevant to a wide range of real-world applications, ensuring that the model can generalize well to everyday scenarios.

The model used in this project, **YOLOv5** (pre-trained), has been trained on the COCO dataset. However, the implementation can be extended to include other models like Faster R-CNN or SSD, depending on accuracy and speed trade-offs for specific use cases.

* **Video Input**: The video used in this project is named 'people.mp4'. It contains scenes featuring various object types (e.g., people, cars, bicycles). The video serves as a test case for the model’s real-time object detection capability.
* **Model**: The pre-trained model **YOLOv5** (using the 'yolov5su.pt' variant) is used in this project to perform the actual detection. The model provides the necessary functionality to identify and classify the objects in the video frames.

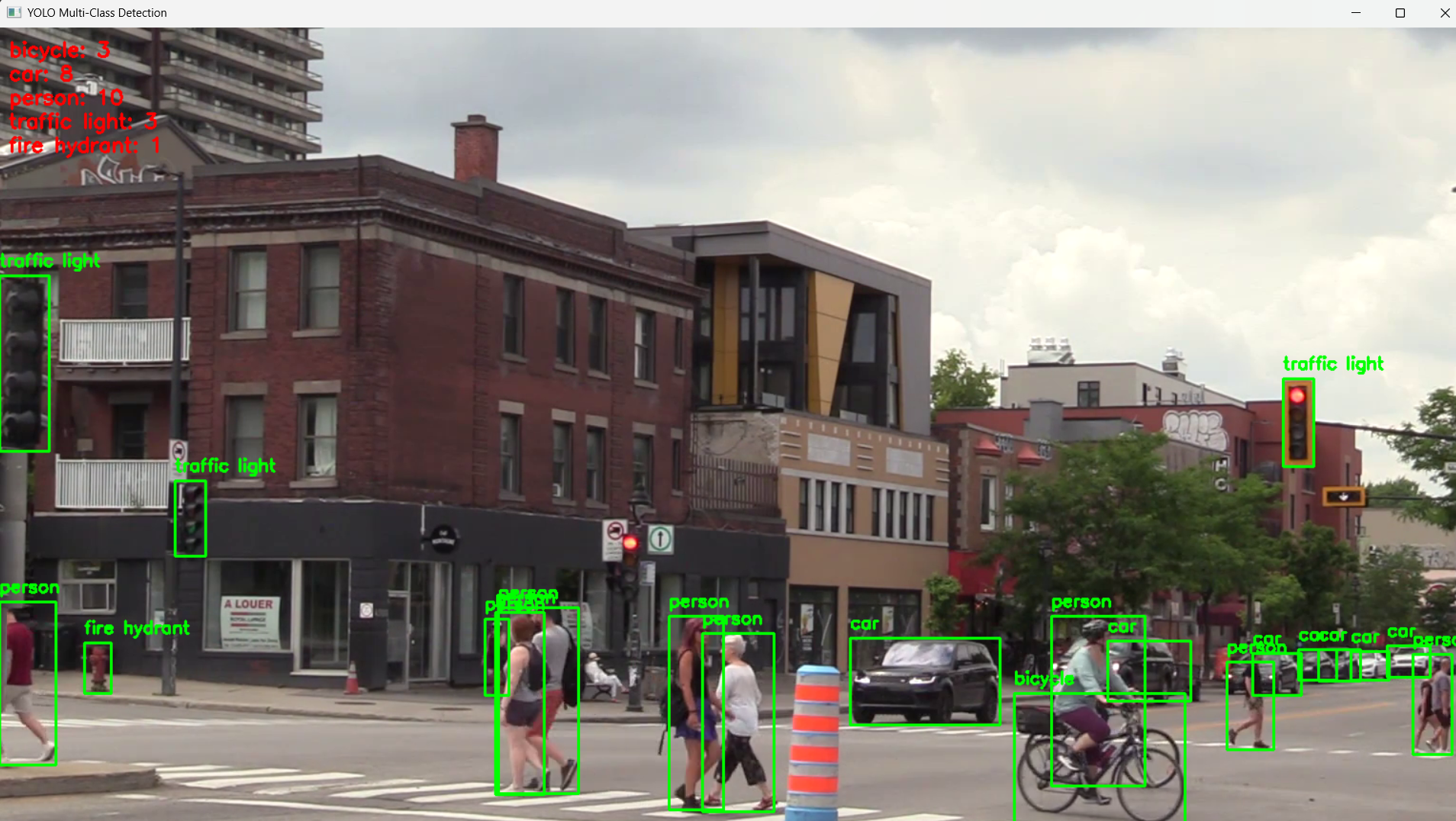
**Methodology**

The project follows a systematic approach for real-time multi-class object detection, which involves the following steps:

1. **Loading the Model**: We load the pre-trained YOLOv5 model.
2. **Reading Video Frames**: The input video is read frame by frame using **OpenCV**, a popular computer vision library. Each frame is processed individually to detect and classify objects.
3. **Object Detection**: The frame is passed through the detection model, which identifies objects and assigns each object a class label. The output is a set of bounding boxes with class names and confidence scores for each detected object.
4. **Drawing Bounding Boxes and Labels**: For each detected object, a bounding box is drawn around it with the corresponding class label (e.g., 'person', 'car'). This is done using OpenCV's drawing functions.
5. **Counting Object Classes**: A dictionary or counter keeps track of the number of occurrences of each object class detected in the current frame.
6. **Real-Time Display**: The annotated frame is displayed in a separate window, showing bounding boxes, labels, and object counts in real time.
7. **Exiting the Program**: The user can press the 'q' key to exit the program at any time.

**Sample Output and Results**

Sample output includes frames with bounding boxes labeled with object names such as 'person: 3', 'bicycle: 1', etc.



The object detection system successfully detected and classified multiple objects in each video frame. The model's ability to correctly identify people, vehicles, and animals was demonstrated, and the results were accurate given the limitations of the pre-trained model.

* **Detection Performance**: The system maintained real-time performance on a typical desktop machine. This demonstrates the potential of using deep learning-based object detection in real-time applications.
* **Accuracy**: The accuracy of the detections was high, especially for objects that were well-defined and not occluded.

**Limitations**

* **Lighting and Resolution**: Poor lighting or low resolution videos can reduce the model’s ability to detect objects accurately.
* **Custom Objects**: The system is limited to detecting only the classes present in the COCO dataset unless the model is retrained on a custom dataset.
* **Processing Speed**: While YOLO is fast, on machines with lower computational power, the inference speed may decrease, particularly for higher-resolution videos.

**Future Scope**

* **Object Tracking**: To improve detection continuity across frames, object tracking can be integrated to link detections between frames. This can reduce flickering and improve tracking accuracy.
* **Edge Deployment**: Future work could involve deploying the object detection system on edge devices like Raspberry Pi or NVIDIA Jetson to enable real-time object detection in field applications like smart surveillance or autonomous drones.
* **Custom Training**: A key future improvement would be to fine-tune or retrain the detection model on a custom dataset for detecting specialized objects (e.g., medical instruments, industrial equipment).

**References**

* YOLOv5 by Ultralytics: <https://github.com/ultralytics/ultralytics>
* COCO Dataset: <https://cocodataset.org>
* OpenCV Documentation: <https://docs.opencv.org>