AI & ML

Project- Object Detection & Tracking

Contributors-

1. Mohd. Hermain Irfan

Object detection and tracking

In this project, we developed a Python program utilizing the pre-trained YOLO model for object detection and tracking. The project has two main functionalities:

1. **Object Tracking**: Counting the number of cars passing along a highway.
2. **Object Detection**: Identifying objects in real-time using a camera feed.

This implementation demonstrates the effectiveness of YOLO in tracking and detecting objects efficiently.

1. Highway Object tracking

First file- object_tracking.py

```python
import cv2
import numpy as np
from object_detection import ObjectDetection
import math

# Initialize Object Detection
od = ObjectDetection()

cap = cv2.VideoCapture("los_angeles.mp4")

# Initialize count
count = 0
center_points_prev_frame = []

tracking_objects = {}
track_id = 0

while True:
    ret, frame = cap.read()
    count += 1
    if not ret:
        break

    # Point current frame
    center_points_cur_frame = []

    # Detect objects on frame
    (class_ids, scores, boxes) = od.detect(frame)
    for box in boxes:
        (x, y, w, h) = box
        cx = int((x + x + w) / 2)
        cy = int((y + y + h) / 2)
        center_points_cur_frame.append((cx, cy))
        #print("FRAME N°", count, " ", x, y, w, h)

        # cv2.circle(frame, (cx, cy), 5, (0, 0, 255), -1)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

```

```
        # Only at the beginning we compare previous and current frame
        if count <= 2:
            for pt in center_points_cur_frame:
                for pt2 in center_points_prev_frame:
                    distance = math.hypot(pt2[0] - pt[0], pt2[1] - pt[1])

                    if distance < 20:
                        tracking_objects[track_id] = pt
                        track_id += 1
        else:

            tracking_objects_copy = tracking_objects.copy()
            center_points_cur_frame_copy = center_points_cur_frame.copy()

            for object_id, pt2 in tracking_objects_copy.items():
                object_exists = False
                for pt in center_points_cur_frame_copy:
                    distance = math.hypot(pt2[0] - pt[0], pt2[1] - pt[1])

                    # Update IDs position
                    if distance < 20:
                        tracking_objects[object_id] = pt
                        object_exists = True
                        if pt in center_points_cur_frame:
                            center_points_cur_frame.remove(pt)
                        continue

                # Remove IDs lost
                if not object_exists:
                    tracking_objects.pop(object_id)

            # Add new IDs found
            for pt in center_points_cur_frame:
                tracking_objects[track_id] = pt
                track_id += 1

        for object_id, pt in tracking_objects.items():
            cv2.circle(frame, pt, 5, (0, 0, 255), -1)
            cv2.putText(frame, str(object_id), (pt[0], pt[1] - 7), 0, 1, (0, 0, 255), 2)
```

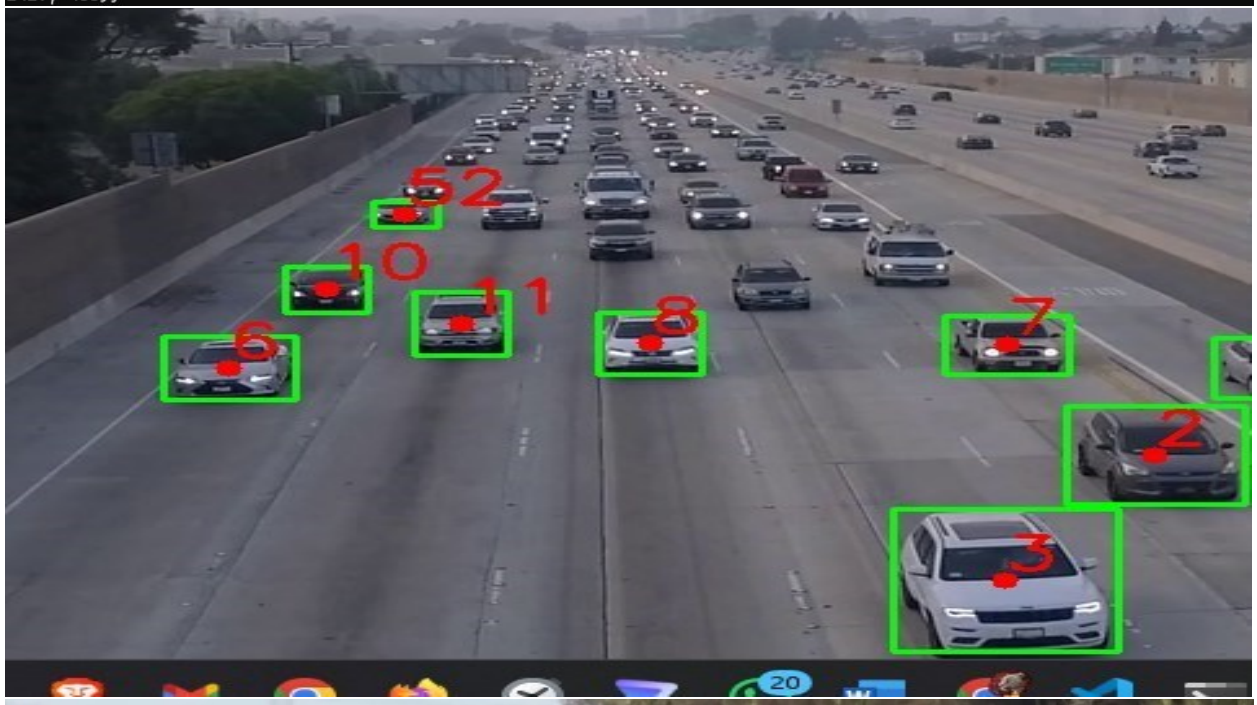
```
            # Add new IDs found
            for pt in center_points_cur_frame:
                tracking_objects[track_id] = pt
                track_id += 1

        for object_id, pt in tracking_objects.items():
            cv2.circle(frame, pt, 5, (0, 0, 255), -1)
            cv2.putText(frame, str(object_id), (pt[0], pt[1] - 7), 0, 1, (0, 0, 255), 2)

        print("Tracking objects")
        print(tracking_objects)


        print("CUR FRAME LEFT PTS")
        print(center_points_cur_frame)


        cv2.imshow("Frame", frame)

        # Make a copy of the points
        center_points_prev_frame = center_points_cur_frame.copy()

        key = cv2.waitKey(1)
        if key == 27:
            break

    cap.release()
    cv2.destroyAllWindows()

```

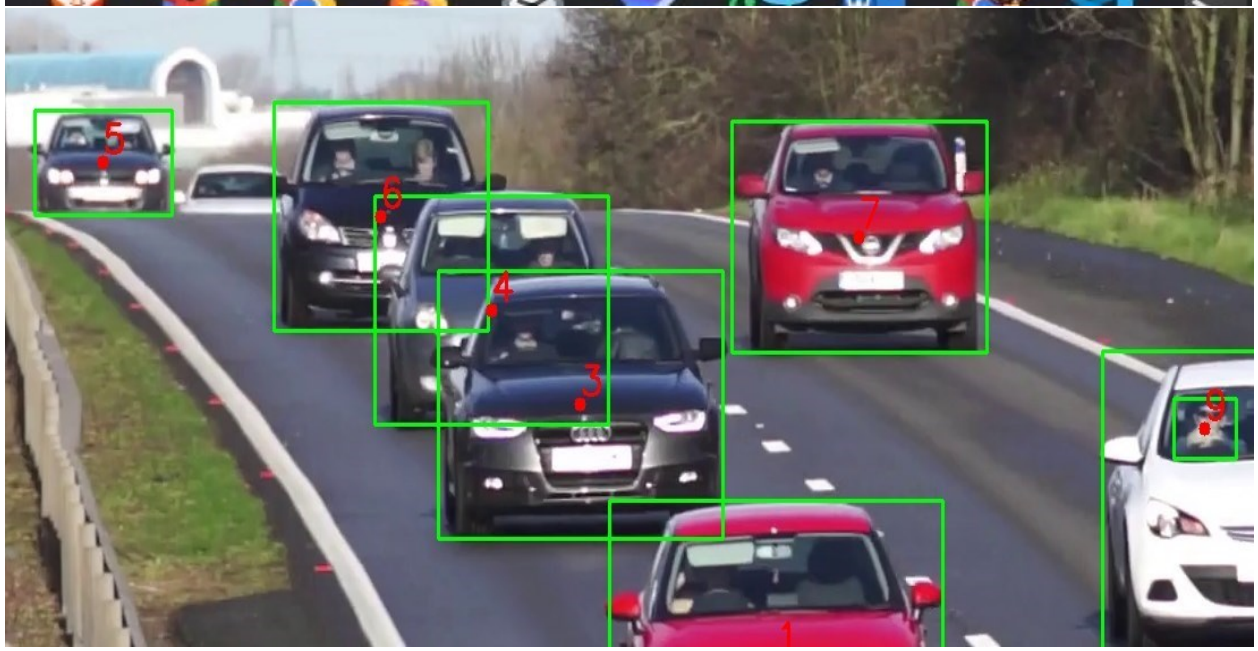
Second-file – object_detection.py

```python
import cv2
import numpy as np


class ObjectDetection:
    def __init__(self, weights_path="dnn_model/yolov4.weights", cfg_path="dnn_model/yolov4.cfg"):
        print("Loading Object Detection")
        print("Running opencv dnn with YOLOv4")
        self.nmsThreshold = 0.4
        self.confThreshold = 0.5
        self.image_size = 608

        # Load Network
        net = cv2.dnn.readNet(weights_path, cfg_path)

        # Enable GPU CUDA
        net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
        net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)
        self.model = cv2.dnn_DetectionModel(net)

        self.classes = []
        self.load_class_names()
        self.colors = np.random.uniform(0, 255, size=(80, 3))

        self.model.setInputParams(size=(self.image_size, self.image_size), scale=1/255)

    def load_class_names(self, classes_path="dnn_model/classes.txt"):

        with open(classes_path, "r") as file_object:
            for class_name in file_object.readlines():
                class_name = class_name.strip()
                self.classes.append(class_name)

        self.colors = np.random.uniform(0, 255, size=(80, 3))
        return self.classes

    def detect(self, frame):
        return self.model.detect(frame, nmsThreshold=self.nmsThreshold, confThreshold=self.confThreshold)

```

Output-

```
PS C:\Users\mohdh\Desktop\AI & ML project\Highway car count> python .\object_tracking.py
Loading Object Detection
Running opencv dnn with YOLOv4
[ WARN:0@1.253] global net_impl.cpp:178 cv::dnn::dnn4_v20240521::Net::Impl::setUpNet DNN module was not built with CUDA
backend; switching to CPU
Tracking objects
{}
CUR FRAME LEFT PTS
[(571, 891), (437, 742), (1750, 635), (761, 648), (1018, 647), (928, 533), (856, 565), (612, 473), (881, 473), (1336, 98
0), (753, 463), (1118, 436), (942, 458), (1877, 605), (642, 435), (688, 450), (1268, 433), (1424, 466)]
Tracking objects
{0: (567, 905), 1: (434, 750), 2: (930, 534), 3: (857, 567), 4: (1019, 650), 5: (761, 655), 6: (612, 474), 7: (881, 473)
, 8: (753, 463), 9: (1867, 590), 10: (641, 436), 11: (687, 451), 12: (943, 459), 13: (1347, 984), 14: (1114, 435), 15: (
1417, 465)}
```





2. Object Detection on WebCam

This project leverages Python and the pre-trained YOLO (You Only Look Once) model for efficient object detection. YOLO is a state-of-the-art deep learning algorithm known for its speed and accuracy in identifying objects in images or video streams. By integrating YOLO, the project can detect and classify multiple objects in real-time, making it suitable for applications such as surveillance, traffic monitoring, and more. The implementation highlights the ease of using pre-trained models for advanced tasks without requiring extensive training or data preprocessing.

first-file- main.py

```python
import cv2
from gui_buttons import Buttons

 (variable) button: Buttons
button = Buttons()
button.add_button("person", 20, 20)
button.add_button("cell phone", 20, 100)
button.add_button("keyboard", 20, 180)
button.add_button("remote", 20, 260)
button.add_button("scissors", 20, 340)

colors = button.colors

# Opencv DNN
net = cv2.dnn.readNet("dnn_model/yolov4-tiny.weights", "dnn_model/yolov4-tiny.cfg")
model = cv2.dnn_DetectionModel(net)
model.setInputParams(size=(320, 320), scale=1/255)

# Load class lists
classes = []
with open("dnn_model/classes.txt", "r") as file_object:
    for class_name in file_object.readlines():
        class_name = class_name.strip()
        classes.append(class_name)

print("Objects list")
print(classes)

# Initialize camera
cap = cv2.VideoCapture(0)  # Ensure the correct index for your camera
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

# Callback for button clicks
def click_button(event, x, y, flags, params):
    global button_person
    if event == cv2.EVENT_LBUTTONDOWN:
        button.button_click(x, y)

# Create window
cv2.namedWindow("Frame")
cv2.setMouseCallback("Frame", click_button)

while True:
    # Get frames
    ret, frame = cap.read()

    if not ret or frame is None:
        print("Failed to capture frame. Exiting...")
        break

    # Get active buttons list
    active_buttons = button.active_buttons_list()

    # Object Detection
    (class_ids, scores, bboxes) = model.detect(frame, confThreshold=0.3, nmsThreshold=0.4)
    for class_id, score, bbox in zip(class_ids, scores, bboxes):
        (x, y, w, h) = bbox
        class_name = classes[class_id]
        color = colors[class_id]

        if class_name in active_buttons:
            cv2.putText(frame, class_name, (x, y - 10), cv2.FONT_HERSHEY_PLAIN, 3, color, 2)
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 3)

    # Display buttons
    button.display_buttons(frame)

    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1)
    if key == 27:  # ESC key to exit
        break

cap.release()
cv2.destroyAllWindows()

```

Second-file-setup.py

setup.py

```
import sys
from cx_Freeze import setup, Executable


setup(name="Simple Object Detection Software",
      version="0.1",
      description="This software detects objects in realtime",
      executables=[Executable("main.py")]
      )


```

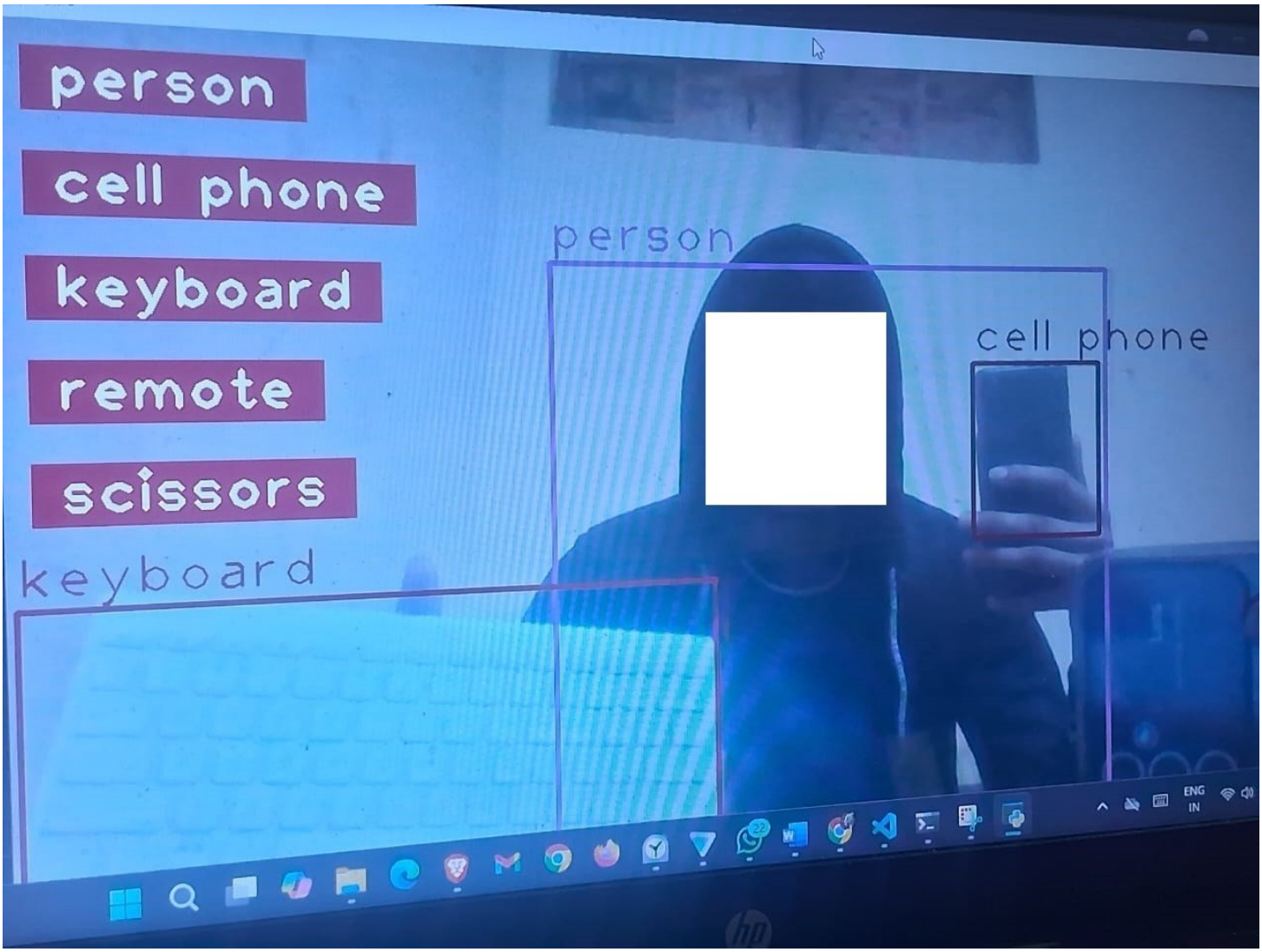
Third file- Gui-buttons.py

```
import cv2
import numpy as np


class Buttons:
    def __init__(self):
        # Font
        self.font = cv2.FONT_HERSHEY_PLAIN
        self.text_scale = 3
        self.text_thick = 3
        self.x_margin = 20
        self.y_margin = 10

        # Buttons
        self.buttons = {}
        self.button_index = 0
        self.buttons_area = []

        np.random.seed(0)
        self.colors = []
        self.generate_random_colors()

    def generate_random_colors(self):
        for i in range(91):
            random_c = np.random.randint(256, size=3)
            self.colors.append((int(random_c[0]), int(random_c[1]), int(random_c[2])))


    def add_button(self, text, x, y):
        # Get text size
        textsize = cv2.getTextSize(text, self.font, self.text_scale, self.text_thick)[0]
        right_x = x + (self.x_margin * 2) + textsize[0]
        bottom_y = y + (self.y_margin * 2) + textsize[1]

        self.buttons[self.button_index] = {"text": text, "position": [x, y, right_x, bottom_y], "acti
        self.button_index += 1

```

```python
    def display_buttons(self, frame):
        for b_index, button_value in self.buttons.items():
            button_text = button_value["text"]
            (x, y, right_x, bottom_y) = button_value["position"]
            active = button_value["active"]

            if active:
                button_color = (0, 0, 200)
                text_color = (255, 255, 255)
                thickness = -1
            else:
                button_color = (0, 0, 200)
                text_color = (0, 0, 200)
                thickness = 3

            # Get  text size
            cv2.rectangle(frame, (x, y), (right_x, bottom_y),
                          button_color, thickness)
            cv2.putText(frame, button_text, (x + self.x_margin, bottom_y - self.y_margin),
                        self.font, self.text_scale, text_color, self.text_thick)
        return frame


    def button_click(self, mouse_x, mouse_y):
        for b_index, button_value in self.buttons.items():
            (x, y, right_x, bottom_y) = button_value["position"]
            active = button_value["active"]
            area = [(x, y), (right_x, y), (right_x, bottom_y), (x, bottom_y)]

            inside = cv2.pointPolygonTest(np.array(area, np.int32), (int(mouse_x), int(mouse_y)), False)
            if inside > 0:
                print("IS Ac", active)
                new_status = False if active is True else True
                self.buttons[b_index]["active"] = new_status

    def active_buttons_list(self):
        active_list = []
        for b_index, button_value in self.buttons.items():
            active = button_value["active"]
            text = button_value["text"]
            if active:
                active_list.append(str(text).lower())

        return active_list
```

Output-

```
PS C:\Users\mohdh\Desktop\AI & ML project\WebCam Obj Detection> python .\main.py
Objects list
['person', 'bicycle', 'car', 'motorbike', 'aeroplane', 'bus', 'train', 'truck', 'boat', 'traffic light', 'fire hydrant',
 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'gir
affe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'bas
eball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife'
, 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', '
chair', 'sofa', 'pottedplant', 'bed', 'diningtable', 'toilet', 'tvmonitor', 'laptop', 'mouse', 'remote', 'keyboard', 'ce
ll phone', 'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'h
air drier', 'toothbrush']
IS Ac False
IS Ac False
IS Ac False
IS Ac False
```


person
cell phone
keyboard
remote
scissors
person
cell phone
keyboard