"Computer Vision based customer footfall and behaviour analysis" project

Behavior Analysis" project using **TensorFlow** and **YOLOv8**:

### ****1. Problem Definition****

* **Objective:** Track customer footfall and behavior in a retail store to provide actionable insights on store layout and staff allocation.
* **Deliverables:**
  + Customer count (footfall).
  + Heatmaps showing areas of high and low engagement.
  + Dwell time analysis for different store zones.

### ****2. Data Collection****

* **Input Data:**
  + CCTV footage from the store (e.g., 1080p resolution, 30 FPS).
* **Preprocessing:**
  + Extract video frames using tools like **OpenCV** or **FFmpeg**.
  + Example code for extracting frames every second:

python

Copy code

import cv2

video = cv2.VideoCapture('store\_video.mp4')

success, frame = video.read()

count = 0

while success:

if count % 30 == 0: # Assuming 30 FPS, extract one frame per second

cv2.imwrite(f"frames/frame\_{count}.jpg", frame)

success, frame = video.read()

count += 1

video.release()

* Anonymize sensitive data (e.g., blur faces using OpenCV).

### ****3. Model Selection****

* **Model Used:** YOLOv8 (Ultralytics' latest version of YOLO).
  + YOLOv8 can be used for object detection, people tracking, and segmentation.
* **Setup:**
  + Install YOLOv8 with TensorFlow backend:

bash

Copy code

pip install ultralytics tensorflow

### ****4. Model Training****

* **Dataset Preparation:**
  + Annotate the extracted frames using tools like **LabelImg** or **Roboflow**.
  + Label bounding boxes for "person" objects and export in YOLO format.
* **Transfer Learning with YOLOv8:**
  + Fine-tune the pre-trained YOLOv8 model on the store dataset:

python

Copy code

from ultralytics import YOLO

model = YOLO('yolov8n.pt') # Load pre-trained YOLOv8 model

model.train(data='data.yaml', epochs=50, imgsz=640) # Train on custom dataset

* + The data.yaml file specifies:
    - Paths to the dataset.
    - Classes (e.g., person).
* Validate the trained model:

python

Copy code

metrics = model.val()

print(metrics)

### ****5. Tracking Customer Behavior****

* **Tracking with Deep SORT:**
  + Combine YOLOv8 detections with Deep SORT for tracking customer movements across frames.
  + Use the bounding box and confidence outputs from YOLOv8 to initialize trackers.
  + Example YOLOv8 + Deep SORT integration:

python

Copy code

from deep\_sort\_realtime.deepsort\_tracker import DeepSort

tracker = DeepSort(max\_age=30) # Initialize tracker

results = model.predict('frame.jpg') # YOLOv8 prediction

for det in results[0].boxes:

bbox = det.xyxy[0].numpy() # Bounding box

confidence = det.conf[0].item() # Confidence score

tracker.update\_tracks([[\*bbox, confidence, 'person']])

* **Zone Analysis:**
  + Divide the store into predefined zones (e.g., entry, aisles, checkout) using coordinates.
  + Count how long each tracker ID stays in a zone.

### ****6. Data Analysis****

* **Footfall Count:**
  + Aggregate the number of unique tracker IDs detected over time.
* **Dwell Time:**
  + Calculate the time each tracker ID spends in a specific zone.
* **Heatmap Generation:**
  + Visualize customer movements using Matplotlib:

python

Copy code

import seaborn as sns

import matplotlib.pyplot as plt

heatmap\_data = [[0 for \_ in range(10)] for \_ in range(10)] # Example grid

for position in customer\_positions:

x, y = position # Map coordinates to grid

heatmap\_data[y][x] += 1

sns.heatmap(heatmap\_data, cmap='YlOrRd')

plt.show()

### ****7. Visualization and Insights****

* **Interactive Dashboard:**
  + Use **Streamlit** to create a real-time dashboard for:
    - Displaying footfall numbers.
    - Generating live heatmaps.
    - Reporting dwell times for each zone.
  + Example code for a Streamlit app:

python

Copy code

import streamlit as st

st.title("Customer Footfall and Behavior Analysis")

st.metric("Total Footfall", total\_footfall)

st.image("heatmap.png", caption="Customer Heatmap")

### ****8. Deployment****

* **Deployment Environment:**
  + Use a local **edge device** like NVIDIA Jetson Nano or a cloud service (AWS, GCP) for real-time inference.
* **Inference Pipeline:**
  + Stream video input to YOLOv8 model for real-time detection.
  + Process output with Deep SORT for tracking and zone analysis.
  + Save results to a database for later visualization.

### ****9. Monitoring and Feedback****

* Periodically evaluate model performance with new video data.
* Fine-tune the YOLOv8 model as needed to adapt to changes in store layout, lighting, or other factors.

### Example Workflow Diagram:

1. **Input:** CCTV footage.
2. **Detection (YOLOv8):** Identify people in frames.
3. **Tracking (Deep SORT):** Track individuals across frames.
4. **Behavior Analysis:** Analyze dwell times, heatmaps.
5. **Output:** Dashboards, reports, real-time alerts.