Shopformer Implementation

Objective

Implement an anomaly detection pipeline inspired by the ShopFormer paper using 2D human pose keypoints. Since pre-trained weights or the full ShopFormer architecture were unavailable, a **custom autoencoder-based model** was built, trained on the PoseLift dataset, and used to detect abnormal human behavior via reconstruction error.

Implementation Summary

1. Dataset (PoseLift Dataset):

- Each JSON file contains frame-wise 2D keypoints in the form [x1, y1, c1, x2, y2, c2, ...].
- The dataset was split into train/ (normal behavior) and test/ folders.

2. Custom Pose Autoencoder (instead of Shopformer):

- Input: 17 human joints → 34 values per frame.
- The autoencoder learns to reconstruct normal human poses.
- Trained using MSE loss to minimize reconstruction error.
- Saved weights as pose_model.pth.

3. Anomaly Detection Logic:

- During inference, each frame's pose is passed through the trained model.
- Reconstruction error = MSE(original_pose, reconstructed_pose)
- Frames with error above a threshold are flagged as anomalies (shoplifting behavior).
- Anomaly frames, error plots, and sample visualizations were generated.

Results

Component	Status
Autoencoder Training	Done on PoseLift dataset
Model Weights (.pth)	Generated successfully
Reconstruction Error Plot	Implemented
Anomaly Frame Detection	Implemented
Video Annotation	Works with bounding boxes + "Anomaly" label
ShopFormer Architecture	Not used (no weights/code available)

Challenges

Challenge	How it was addressed
No pre-trained ShopFormer weights available	Built a custom pose autoencoder instead.
No official ShopFormer training pipeline/code	Designed own training workflow
Only pose keypoints available (no raw video model input)	Adapted pipeline to work directly on PoseLift JSON files.
Setting a reliable anomaly threshold	Used statistical approach (mean + std-dev of reconstruction error).

Future Improvements

- Train a temporal model (LSTM/Transformer) to analyze pose sequences, not isolated frames.
- Add evaluation metrics (Precision/Recall/F1) using labeled anomaly test videos.
- Optimize for real-time inference + webcam / CCTV feed deployment.
- Export pipeline as a modular Python package or REST API.