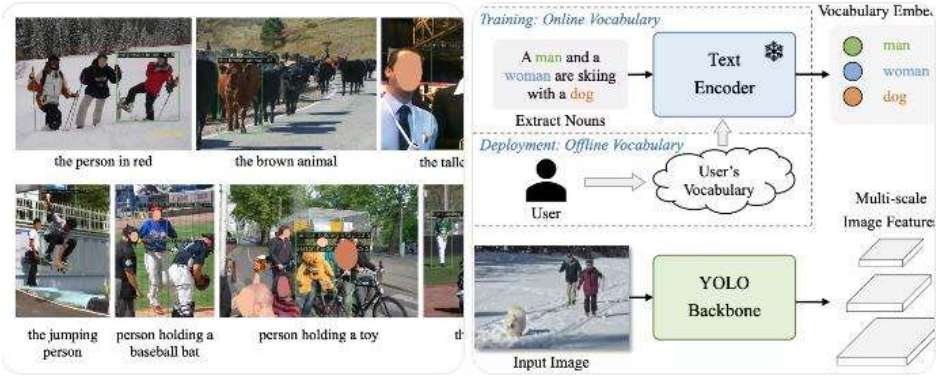
Semantic Event Detection with Optimized Vision–Language Model (VLM)

1. Project Overview

This project presents a real-time **semantic event detection system** built using a Vision–Language Model (VLM), specifically YOLO-World. The system processes video input to detect high-level behavioral events rather than just objects. Unlike traditional object detection pipelines that only identify bounding boxes, this implementation integrates spatial tracking and temporal reasoning to interpret meaningful activities such as human walking, vehicle stopping, and crowd formation.

The architecture combines deep learning–based open-vocabulary object detection with rule-based motion analytics. A Streamlit-powered web interface provides an interactive platform for video upload, processing, and results visualization. Additionally, the model is optimized using ONNX export and INT8 quantization to improve inference speed on CPU hardware, making the system suitable for resource-limited environments.

2. Vision–Language Backbone: YOLO-World



The system uses **YOLO-World**, an open-vocabulary object detector built upon the YOLO framework. Unlike traditional detectors trained on fixed classes, YOLO-World integrates language embeddings to dynamically detect objects described via text prompts.

This capability enables flexible detection of categories such as *person*, *car*, *bus*, and *truck* without retraining. The model outputs bounding boxes, class labels, confidence scores, and tracking IDs. These outputs form the foundation for higher-level semantic reasoning.

3. Event Detection Logic

The system transforms raw detections into semantic events by applying temporal and spatial analysis.

3.1 Person Walking Detection

Walking is detected by measuring centroid displacement across consecutive frames. For each tracked person ID, the system calculates Euclidean distance:

If the displacement exceeds 5 pixels across recent frames, the individual is classified as “Walking.” This method effectively distinguishes stationary individuals from moving ones while maintaining computational simplicity.

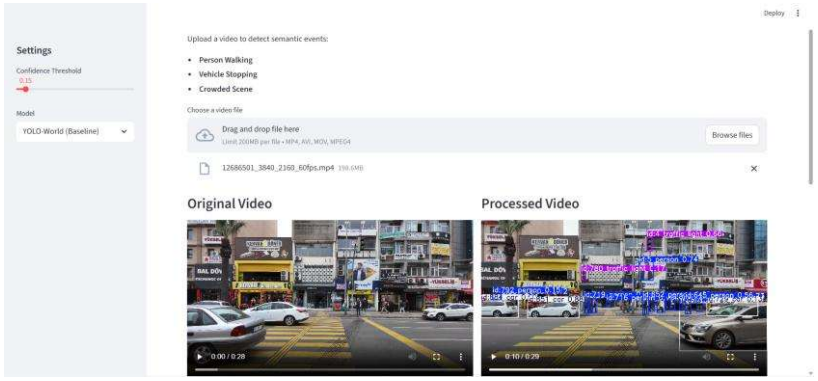
3.2 Vehicle Stopping Detection

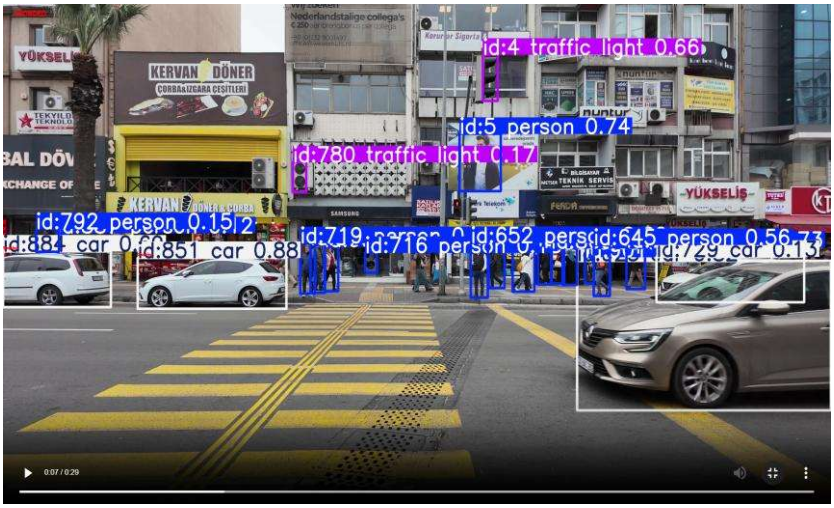
Vehicle stopping is determined through minimal centroid movement over time. If displacement remains below 5 pixels for more than 15 consecutive frames, the vehicle is classified as “Stopped.” This temporal persistence check avoids false positives caused by temporary pauses.

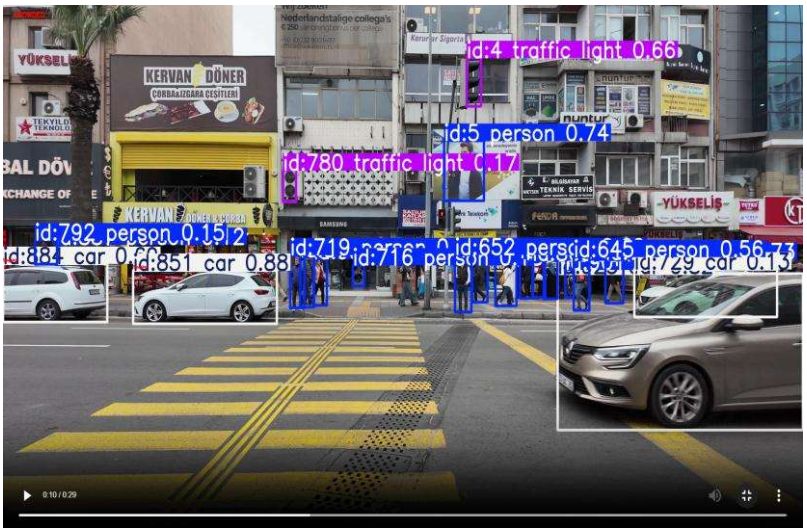
3.3 Crowded Scene Detection

Crowd detection is based on counting unique tracked “person” instances in each frame. If the count exceeds a predefined threshold (default = 5), the frame is labeled as “Crowded Scene.” This approach is scalable and can be adapte:d for public safety or surveillance use cases.

Output







Processing Complete!

Detected Events Log

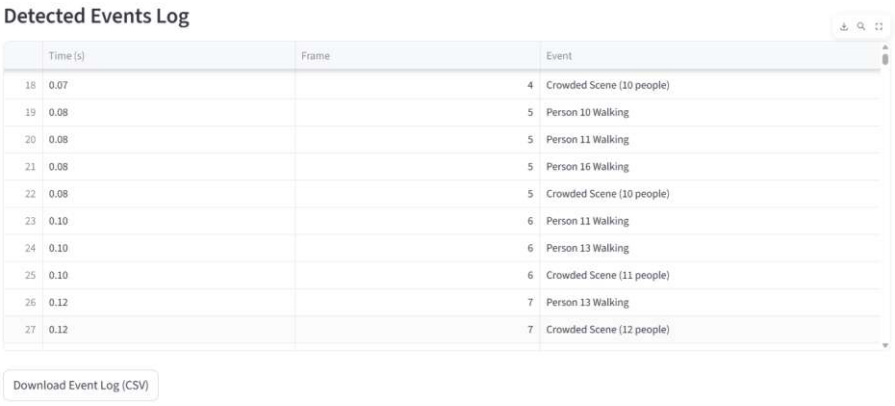
	Time (s)	Frame	Event
0	0.00	0	Crowded Scene (12 people)
1	0.02	1	Person 13 Walking
2	0.02	1	Person 16 Walking
3	0.02	1	Crowded Scene (12 people)
4	0.03	2	Person 10 Walking
5	0.03	2	Person 11 Walking
6	0.03	2	Person 16 Walking
7	0.03	2	Person 18 Walking
8	0.03	2	Person 19 Walking
9	0.03	2	Crowded Scene (12 people)

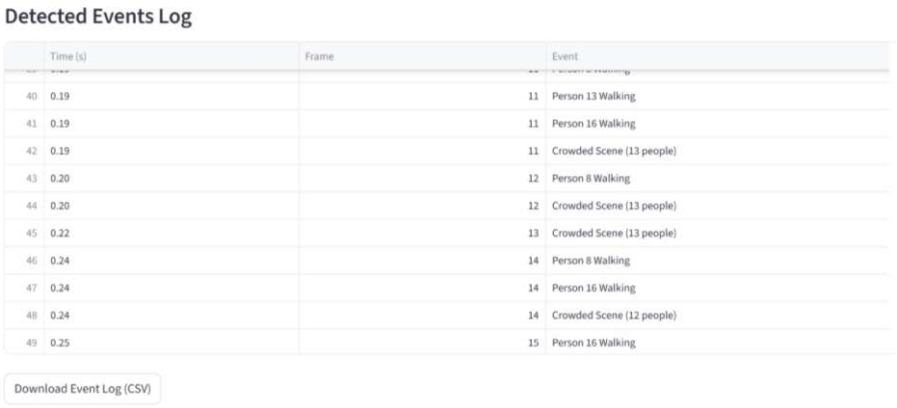
Download Event Log (CSV)

Detected Events Log

	Time (s)	Frame	Event
18	0.07	4	Crowded Scene (10 people)
19	0.08	5	Person 10 Walking
20	0.08	5	Person 11 Walking
21	0.08	5	Person 16 Walking
22	0.08	5	Crowded Scene (10 people)
23	0.10	6	Person 11 Walking
24	0.10	6	Person 13 Walking
25	0.10	6	Crowded Scene (11 people)
26	0.12	7	Person 13 Walking
27	0.12	7	Crowded Scene (12 people)

Download Event Log (CSV)





4. System Architecture

The architecture follows a modular design:

4.1 app.py

This file serves as the Streamlit entry point. It manages:

- Video upload
- Parameter adjustment
- Frame-by-frame inference
- Visualization and logging

4.2 detect_events.py

Contains the EventDetector class. It integrates:

- YOLO detection

- Built-in tracking (persist=True)
- Movement analysis
- Event state transitions

Tracking ensures that each object maintains a consistent ID across frames, enabling temporal reasoning.

4.3 optimize_model.py

Exports the PyTorch model to ONNX format and applies dynamic INT8 quantization using ONNX Runtime. Quantization reduces floating-point precision (FP32 → INT8), decreasing model size and accelerating CPU inference.

4.4 compare_performance.py

Benchmarks inference speed (FPS) for:

- Baseline PyTorch model
- Optimized ONNX INT8 model

This script demonstrates real-world optimization gains.

---

5. Optimization Strategy

The baseline model operates in FP32 precision, which is computationally intensive. To improve deployment efficiency, the system performs:

1. **Model Export to ONNX**
   Converts the PyTorch computation graph into an interoperable format.
2. **Dynamic Quantization (INT8)**
   Reduces numerical precision of weights from 32-bit floating point to 8-bit integers.
3. **ONNX Runtime Execution**
   Uses optimized CPU execution providers for faster inference.

Benefits:

- Reduced model size
- Lower memory consumption
- Faster CPU inference
- Minimal accuracy degradation

This makes the system deployable on edge devices or low-resource systems without GPUs.

---

6. Interactive Dashboard (Streamlit)

The web interface provides:

- Real-time video processing

- Bounding box overlays
- Event timestamp logging
- CSV export functionality
- Adjustable confidence thresholds

The user uploads a video, initiates detection, and observes annotated results with structured event logs.

---

7. Configuration Parameters

Parameter	Default	Function
Walking Threshold	5 pixels	Minimum displacement for walking
Stop Frames	15	Consecutive stationary frames
Crowd Threshold	5 people	Crowd detection trigger
Model	yolov8s-world.pt	Base YOLO-World variant

These parameters are configurable, allowing domain adaptation (e.g., traffic monitoring vs campus surveillance).

---

8. Applications

This system has practical applications in:

- Smart surveillance systems
- Traffic monitoring
- Crowd density analysis
- Public safety analytics
- Edge AI deployments

Because it leverages open-vocabulary detection, it can easily extend to additional semantic events by adjusting textual prompts.

---

9. Conclusion

The Semantic Event Detection system demonstrates how Vision–Language Models can be integrated with rule-based temporal logic to achieve higher-level behavioral understanding in videos. By combining YOLO-World's open-vocabulary detection with centroid-based motion analytics, the system moves beyond object recognition into semantic interpretation.

Furthermore, the ONNX INT8 optimization pipeline ensures practical deployability on CPU-only systems. The modular architecture, real-time dashboard, and benchmarking tools make this implementation both research-ready and production-adaptable.

This project illustrates the transition from pure object detection toward context-aware, event-driven video intelligence systems.