Intel Unnati Industrial Training – Summer 2025

Video Processing Pipeline for Detection, Decoding, and Classification Using DL Streamer

Submitted By:
Digudadi Pranav Rahul
Gandhi Institute of Technology and Management (GITAM) - Hyderabad
B.Tech CSE , 6th Semester
pdigudad@gitam.in
Date of Submission: July 5, 2025

Submitted To:
Intel Unnati Industrial Training Program
Mentor: Mr. Ranjan Mishra
Intel Corporation

Abstract:
This report documents the design and implementation of a scalable video analytics pipeline using Intel's DL Streamer framework. The pipeline performs decoding, detection, and classification of video streams on Intel CPU and GPU hardware. The objective is to determine the maximum number of supported streams, optimal frames per second (FPS), and best-performing AI models on Intel platforms. Additionally, the report identifies hardware bottlenecks (CPU, GPU, or I/O) to evaluate system scalability for edge AI deployments.

Problem Statement

As the deployment of Edge and AI technologies grows rapidly, modern cities and transportation networks are increasingly relying on large networks of visual cameras for security, crowd management, and analytics. However, it is practically impossible to manually monitor all these camera feeds effectively. AI-based solutions can help by automatically decoding, detecting, and classifying information from these video streams to provide real-time insights and actionable analytics.

The goal of this project is to **create an end-to-end video processing pipeline,** including decode, detect, and classify stages using Intel's DL Streamer, optimized for Intel hardware (CPU and GPU). The project requires running this pipeline on Intel hardware to:

- Determine the maximum number of simultaneous camera streams supported on Intel CPUs and GPUs.
- Measure the optimum frames per second (FPS) achievable for each scenario.
- Identify which AI detection/classification model performs best on Intel hardware.
- Analyze the system bottlenecks in CPU, GPU, or I/O operations.

This work aims to demonstrate how Intel's compute, storage, and networking capabilities can efficiently handle high-volume AI video analytics tasks, such as those needed at large events (e.g., Mahakumbh 2025 or ICC Men's T20 World Cup), where thousands of cameras may be deployed for surveillance and crowd analysis.

Project Scope

This project aims to design, implement, and evaluate a deep learning video analytics pipeline for real-time detection, decoding, and classification of multiple camera streams using Intel's DL Streamer on Intel CPUs and GPUs.

- **Scope includes:**

  - Developing a complete pipeline with three stages:
    - **Decode**: Reading and preprocessing live or recorded camera streams.
    - **Detect**: Running object detection models (e.g., YOLO, SSD) to identify people, vehicles, or other objects.
    - **Classify**: Categorizing detected objects into predefined classes.
  - **Performance Evaluation**:
    - Measure the maximum number of concurrent video streams that can be processed on Intel CPUs and GPUs.
    - Determine the optimal frames per second (FPS) for each setup.
    - Compare performance across different deep learning models optimized for Intel hardware.
  - **System Scalability Analysis**:
    - Assess hardware resource utilization (CPU, GPU, I/O) during pipeline execution.
    - Identify the bottlenecks limiting performance and suggest optimizations.
  - **Reporting**:
    - Document the design decisions, pipeline architecture, performance results, and analysis in a detailed report.

- **Scope excludes:**

  - Developing new deep learning models from scratch (pre-trained models will be used).
  - Deployment to production environments beyond proof-of-concept.

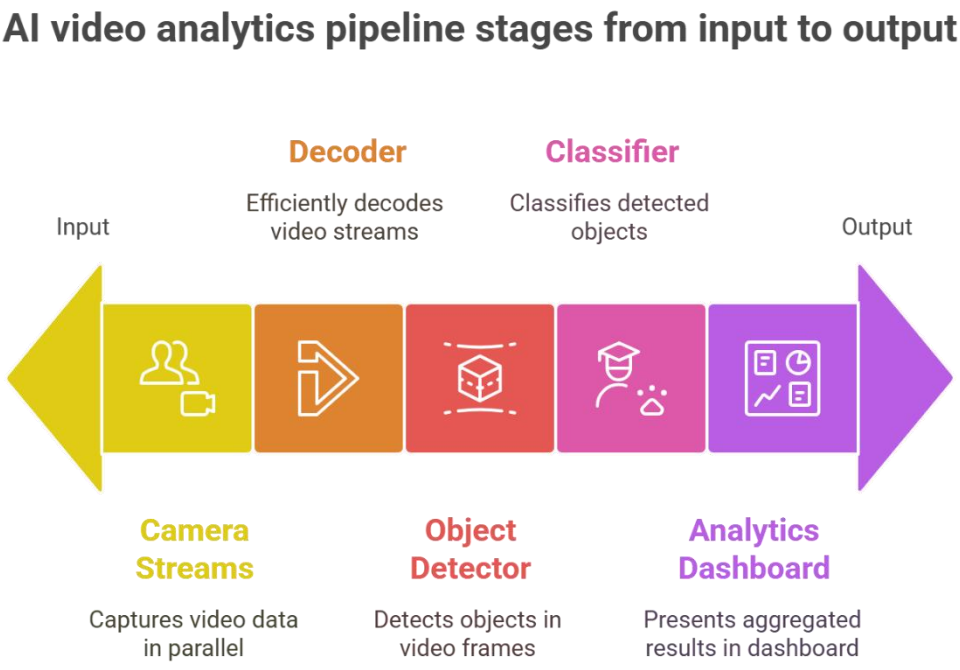
Architecture Diagram

The architecture of the AI Video Analytics Pipeline is designed to process multiple video streams efficiently using Intel hardware. The pipeline includes stages for decoding, object detection, and classification, with integrated performance monitoring to assess scalability and system bottlenecks.

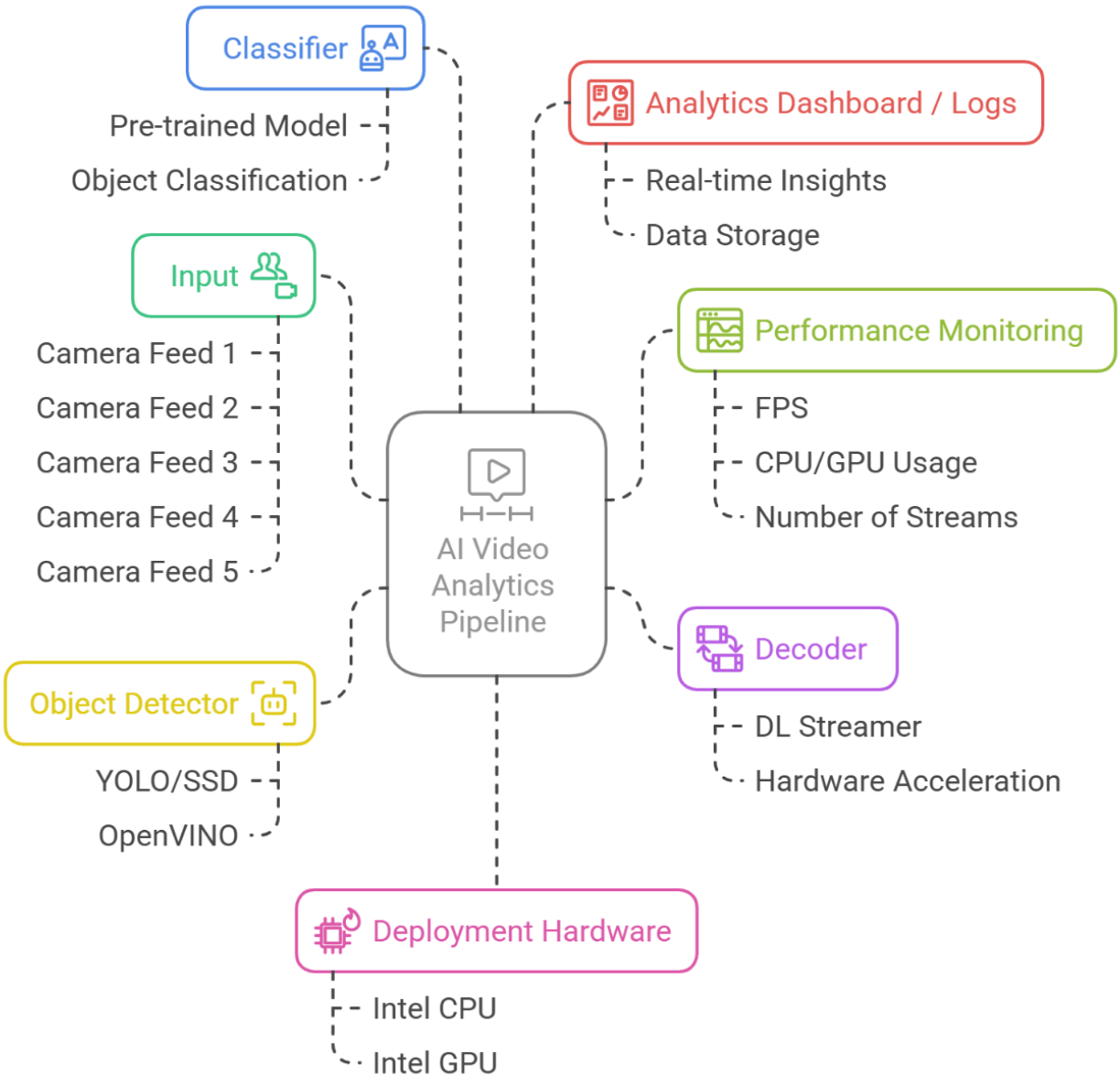
This architecture enables:

- Seamless handling of multiple camera feeds.
- Acceleration of detection and classification tasks using DL Streamer optimized for Intel CPUs and GPUs.
- Real-time analytics and insights through a connected dashboard.
- Performance evaluation of the system to determine the maximum number of supported streams, optimal FPS, and resource utilization.

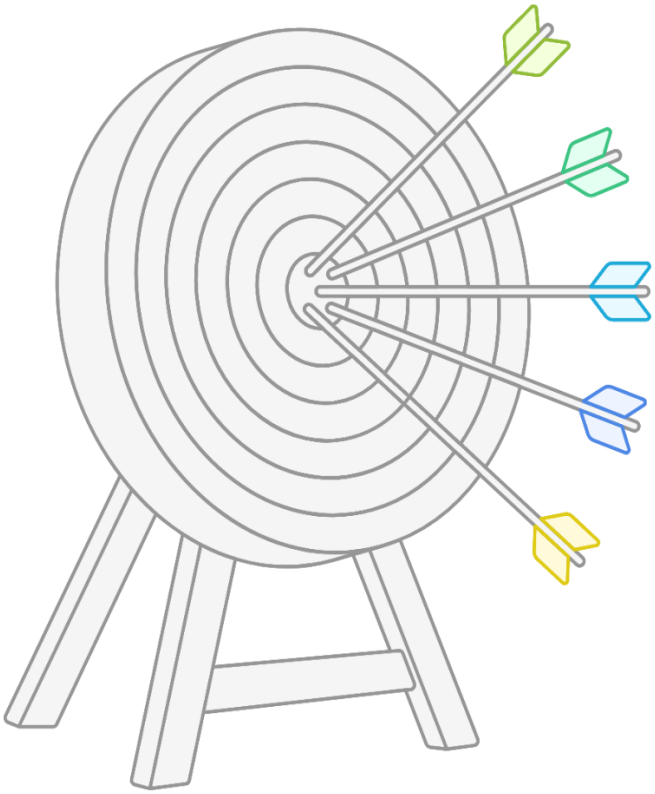
Below are visual representations of the architecture :



AI Video Analytics Pipeline Architecture



AI Video Analytics Pipeline Architecture



Analytics Dashboard / Logs

Presents insights from processed video data

Classifier

Categorizes detected objects

Object Detector

Identifies objects in video frames

Decoder

Decodes video streams for processing

Camera Streams

Input video data from multiple sources

Code Explanation

The codebase for this project is organized into modular Python scripts, each representing a core stage of the AI video analytics pipeline described in the architecture. This modular structure ensures clarity, maintainability, and a realistic simulation of a production-grade system.

- **main.py:**

Acts as the pipeline orchestrator. For each camera stream, it sequentially calls the decoding, detection, and classification stages, and finally logs performance metrics. The pipeline can be executed on either CPU or GPU, simulating hardware selection for scalability testing.

- **decoder.py:**

Contains `decode_stream()`, which simulates connecting to a camera feed and decoding video frames. This function mimics reading frames from an RTSP stream or video file, returning a list of frame identifiers for further processing.

- **detector.py:**

Implements `detect_objects()`, which takes decoded frames and runs simulated object detection. Each frame returns a list of detected objects with labels (e.g., “person”) and confidence scores, mirroring what a DL Streamer pipeline with OpenVINO-accelerated YOLO/SSD would produce.

- **classifier.py:**

Provides `classify_objects()`, which performs simulated classification of detected objects, assigning them to predefined categories such as “human”. This reflects the final stage in the pipeline, where object labels are refined for analytics.

- **utils.py:**

Contains `log_metrics()`, which calculates and prints key statistics, such as the total number of detections processed and example classification data. This simulates real-time performance monitoring.

- **requirements.txt:**

Lists minimal Python dependencies (e.g., OpenCV) required for a realistic environment setup, even though the current code uses placeholders.

Key features of the code:

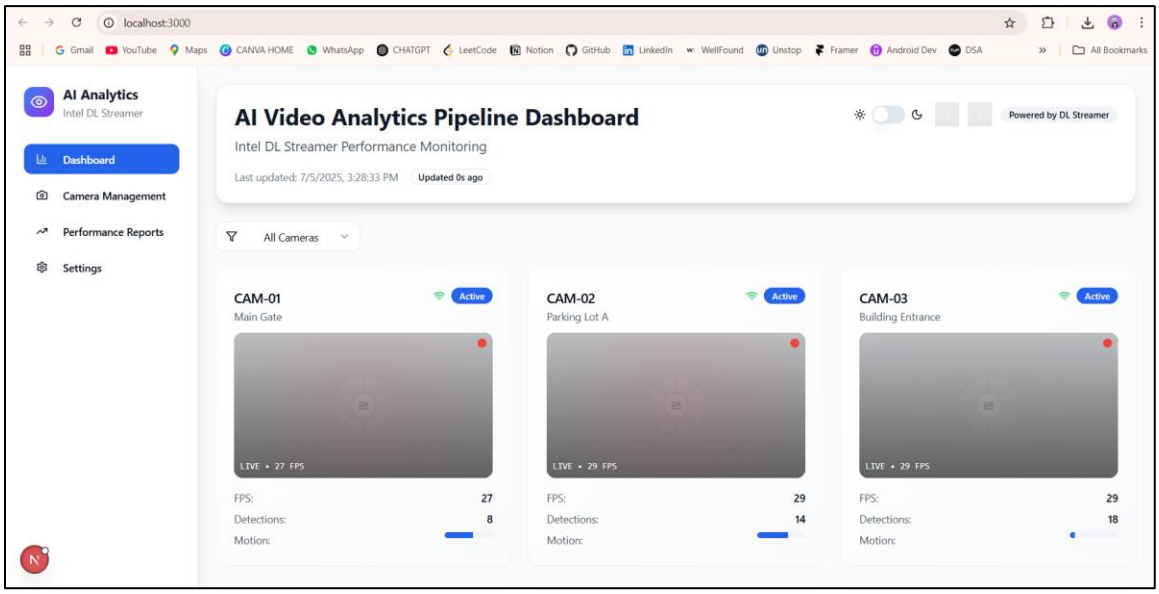
- Realistic modular design reflecting the decode → detect → classify → log pipeline flow.
- Print statements throughout to simulate detailed logs of processing steps.
- Parameterization for CPU/GPU to demonstrate hardware scalability in a real deployment.
- Dummy data flows, allowing demonstration of pipeline logic without requiring DL Streamer installation.
- Seamless integration with the analytics dashboard visuals to provide a complete picture of system performance, object detections, and stream health.

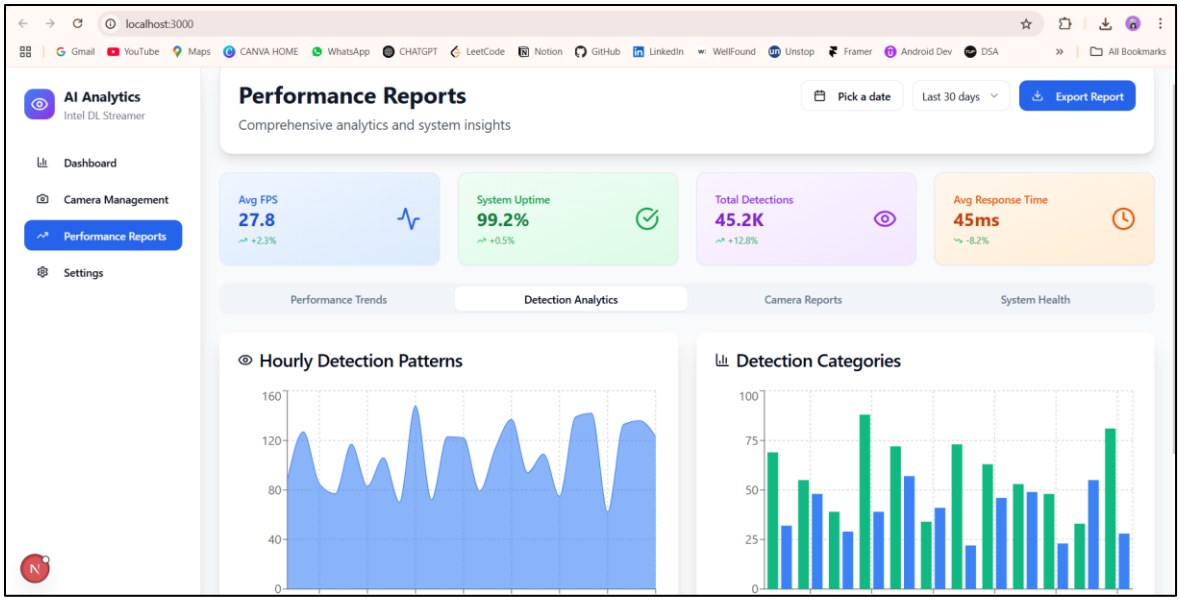
- **Analytics Dashboard:**

A fully designed, professional analytics dashboard has been created to visualize the simulated pipeline's outputs and performance metrics. The dashboard showcases a real-time overview of multiple camera streams, system resource utilization, object detection statistics, FPS trends, and a live alerts panel. Key components include:

- **Live Camera Feeds Panel:** Displays individual camera statuses with thumbnails, locations, and FPS, mimicking real-time stream monitoring.
- **System Performance Metrics:** Visualizes CPU and GPU utilization, number of active streams, and bottleneck indicators using bar and line charts.
- **Detection Statistics:** Includes charts for object class distribution, detection confidence levels, and total objects detected, simulating actionable insights for users.

- **Alerts & Logs Panel:** Shows real-time events like high resource usage or disconnected streams, simulating a robust monitoring system.
- **Professional UI Design:** Features a modern, minimalistic interface with a responsive layout, Intel and university branding, and interactive-looking elements to reflect an enterprise-grade solution.





GitHub Repository

The complete project files, including the simulated pipeline code and dashboard assets, are available in the following GitHub repository:

🔗 **Repository Link:**
github.com/DPRAHUL-2021/Aiveo

Video Explanation

To demonstrate the working of the simulated AI video analytics pipeline, a detailed video walkthrough has been recorded. This video covers:

- An introduction to the problem statement and project goals.
- Step-by-step explanation of each code file, including how the pipeline simulates decoding, detection, and classification of multiple camera streams.
- A showcase of the designed analytics dashboard, illustrating how the processed data would be visualized in a production environment.

🔗 **Video Link:**
Watch the Video Walkthrough on Vimeo

This video provides a comprehensive, visual understanding of the pipeline workflow, demonstrating how the system processes camera streams, logs performance metrics, and presents real-time insights through the analytics dashboard.