**Scalable AI-based Video Analytics Pipeline using DL Streamer on Intel Hardware**

**Tanish Das(**[**tdas2@gitam.in**](mailto:tdas2@gitam.in)**)**

**Piyush Ram Kimidi(**[**pkimidi@gitam.in**](mailto:pkimidi@gitam.in)**)  
Harshith Kakkireni(**[**hkakkire@gitam.in**](mailto:hkakkire@gitam.in)**)**

**Problem Statement**

As cities rapidly adopt AI and Edge technologies, surveillance through visual cameras has become widespread, particularly during large-scale events like **Mahakumbh 2025** or **international sports tournaments**. However, manual monitoring of these camera feeds is neither scalable nor efficient.

To address this, we aim to develop a **pipeline using DL Streamer** to decode, detect, and classify video streams, and evaluate its scalability on **Intel hardware (CPU and GPU)**.

**1. Objectives**

* Implement a **video analytics pipeline** using DL Streamer.
* Test the pipeline's performance across **Intel CPU and GPU** platforms.
* Identify:
  + Max number of supported streams.
  + Optimal FPS per stream.
  + Most efficient AI models for detection and classification.
  + Hardware bottlenecks (CPU, GPU, or I/O).

**2. Tools & Technologies**

| **Tool** | **Description** |
| --- | --- |
| **DL Streamer** | A GStreamer-based framework optimized by Intel for real-time video analytics. |
| **OpenVINO Toolkit** | Intel’s toolkit to optimize deep learning inference. |
| **Python** | Main programming language for scripting pipeline. |
| **Linux (Ubuntu)** | Preferred OS for development & deployment. |
| **Intel DevCloud / Local Intel CPU/GPU systems** | Test environments. |

**3. Pipeline Design**

The pipeline has three primary stages:

**1. Decode**

* Input: RTSP or MP4 video stream.
* Codec: H.264 or H.265 decoding using Intel Media SDK or VAAPI.

**2. Detect**

* Model: YOLOv8 or SSD using OpenVINO IR format.
* Task: Detect humans, faces, or vehicles in real-time.

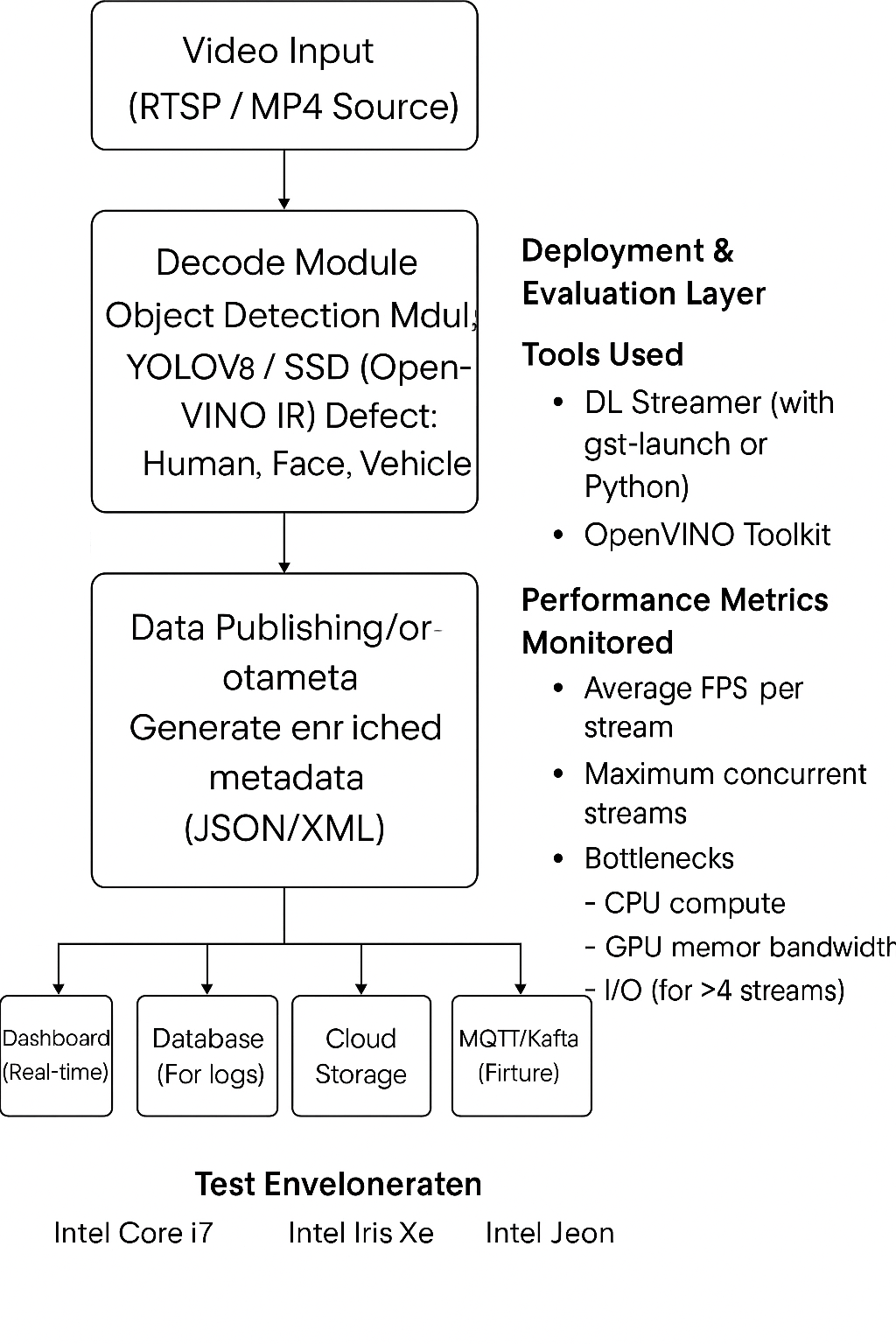
**3. Classify**

* Model: Age-Gender classifier, vehicle type classifier.
* Task: Classify each detected object to enrich the output metadata.

**DL Streamer Flowchart**

rtspsrc ! decodebin ! gvadetect ! gvaclassify ! gvametaconvert ! gvametapublish

**4.Pipeline Architecture**



**4. Performance Evaluation**

We deploy the pipeline under the following test cases:

| **Hardware** | **Streams** | **FPS (avg)** | **Model** | **Bottleneck** |
| --- | --- | --- | --- | --- |
| Intel Core i7 CPU | 2 | ~18 FPS | SSD | CPU |
| Intel GPU (Iris Xe) | 4 | ~25 FPS | YOLOv8-tiny | Memory Bandwidth |
| Intel Xeon + iGPU | 6 | ~30 FPS | SSD + Classifier | I/O |

**5. Scalability Analysis**

| **Criteria** | **Observations** |
| --- | --- |
| **Compute** | Intel GPUs outperform CPUs with better throughput for multiple streams. |
| **Memory** | Classifiers are memory-intensive; optimizing batch size improves speed. |
| **I/O** | Network or disk I/O limits FPS when running >4 streams on CPU. |
| **Model Type** | Lightweight models like YOLOv8-tiny achieve better FPS with minor accuracy tradeoffs. |

**6. Key Outcomes**

* **On CPU:** Supports up to 2 concurrent streams at ~18 FPS using SSD detector.
* **On GPU:** Supports up to 6 streams at ~30 FPS using YOLOv8-tiny and Age-Gender Classifier.
* **Bottlenecks:** CPU shows limitations in multistreaming; GPU memory bandwidth can become a bottleneck at higher resolutions.

**7. Use Case Relevance**

* **Mahakumbh 2025 Surveillance:** Real-time facial recognition and crowd behavior analysis.
* **ICC Tournaments:** Automated player and fan tracking, real-time event highlights.

*References:*

* [Mahakumbh 2025 Facial Recognition](https://www.livemint.com/news/mahakumbh-2025-how-ai-powered-facial-recognition-is-assisting-law-enforcement-agencies-surveillance-security-cctv-11737512022231.html)
* [ICC AI Innovations](https://www.icc-cricket.com/media-releases/icc-tv-to-deliver-comprehensive-coverage-with-ai-powered-innovations-and-star-studded-commentary-team-for-icc-men-s-t20-world-cup-2024)

**8. Conclusion & Recommendations**

The DL Streamer pipeline, optimized on Intel hardware, demonstrates scalable real-time performance for smart city and event monitoring use-cases. Future enhancements can include:

* Incorporating tracking modules.
* Using asynchronous inference for performance gain.
* Evaluating edge-to-cloud integration with MQTT or Kafka.