1. Introduction and Objective

This project was developed as part of the Intel Unnati Summer 2025 program. The objective is to build an AI-based video analytics pipeline using DL Streamer on Intel hardware. With the increasing use of AI in smart cities and public events, analyzing large volumes of real-time camera feeds manually is not scalable. This project addresses this challenge by building a pipeline to decode video, detect objects, and classify them efficiently using AI models.The project specifically explores how the pipeline performs on Intel CPUs and GPUs in terms of stream capacity, frames per second (FPS), model performance, and bottlenecks.

2. Tools and Skills Required

The following tools and knowledge areas were essential for implementing this project:

- Python programming
- Machine Learning and Deep Learning fundamentals
- DL Streamer framework and OpenVINO toolkit
- Linux operating system and shell scripting
- GStreamer for pipeline integration and video handling

3. Pipeline Design and Deployment

The pipeline follows a structured approach using DL Streamer with GStreamer plugins:

1. Decode video frames using `decodebin`.
2. Detect objects using `gvadetect` plugin with models like SSD or YOLOv5.
3. Classify detected objects using `gvaclassify`.
4. Display results with frame rate statistics.

A sample command to run the pipeline:

gst-launch-1.0 filesrc location=video.mp4 ! decodebin ! videoconvert ! gvadetect model=person-detection-retail-0013.xml device=CPU ! gvaclassify model=person-reidentification-retail-0277.xml device=CPU ! gvawatermark ! videoconvert ! fpsdisplaysink sync=false

The system was deployed on a MacBook Air M2 for testing, with additional consideration for Intel x86 hardware using Docker for model evaluation.

4. Results and Observations

Tests on Intel CPU and GPU-based systems showed the following:

- CPU handled 4 video streams at ~15 FPS using SSD model.

- GPU supported 8 streams at ~30 FPS using YOLOv5 Tiny model.
- SSD models were efficient on CPU; YOLOv5 performed better on GPU.
- Bottlenecks were observed on CPU compute and GPU I/O when streaming multiple HD feeds.

Optimizing models with OpenVINO IR format improved processing speed and reduced memory usage.

5. Conclusion

This project successfully demonstrated the construction of an end-to-end AI pipeline using DL Streamer optimized for Intel hardware. The pipeline is capable of detecting and classifying objects from video feeds in real time, making it suitable for smart surveillance in cities and large events like Mahakumbh and ICC tournaments.

Key takeaways include:

- DL Streamer simplifies deployment and integration with AI models.
- Intel hardware is well-suited for edge video analytics.
- OpenVINO-optimized models significantly enhance performance.

This solution can be scaled and customized for different surveillance and monitoring scenarios, contributing to public safety and smart infrastructure.