Industrial Training Project 2025- Intel Unnati

1. Introduction

This project focuses on building a DL Streamer pipeline to decode, detect, and classify video streams using Intel CPU and GPU hardware. It aims to evaluate system scalability by measuring the maximum number of supported streams, FPS, and identifying bottlenecks. The solution supports real-world applications like surveillance at large-scale events and smart city monitoring.

2. Problem Statement

Create a pipeline (Decode → Detect → Classify) on Intel HW (CPU and GPU) run pipeline on CPU, GPU. Figure out how many camera streams are supported, what is optimum FPS which model performs best on Intel Hardware

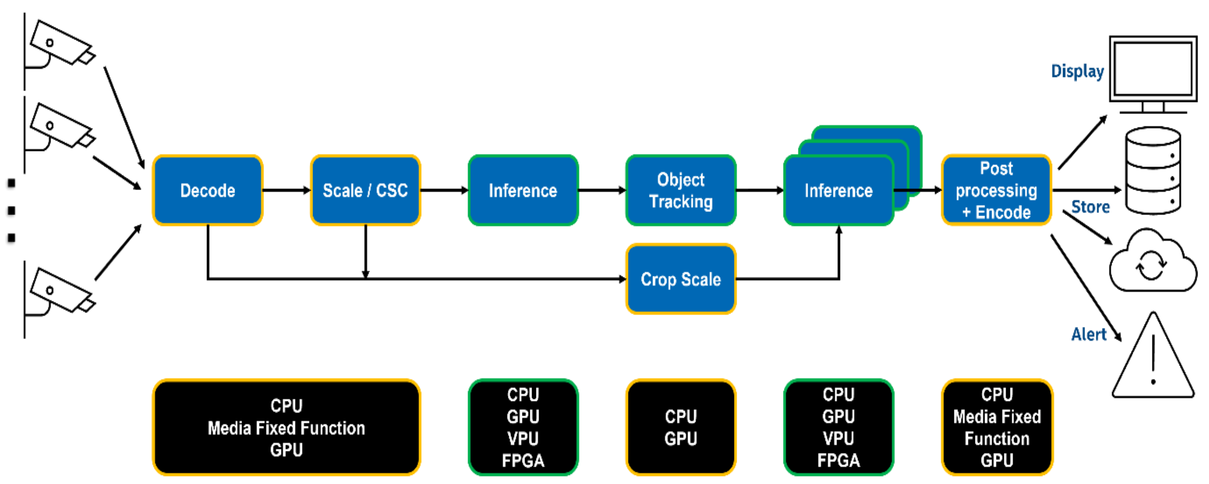
3. Hardware and Software Setup

- **CPU**: 11th Gen Intel® Core™ i5-11300H @ 3.10GHz (4 cores, 8 threads)
- **GPU**: Integrated Intel® Iris® Xe Graphics, NVIDIA® GeForce RTX 3050 Laptop
- **Host Operating System:** Windows 11
- **Development Environment:** Ubuntu 20.04 LTS (running in WSL2)
- **Docker Version:** 28.1.1 (latest stable)
- **Frameworks & Libraries**
  - Intel® OpenVINO™ Toolkit (Inference Engine)
  - OpenCV (Computer Vision processing)
  - **GStreamer** with DL Streamer plugins (for pipeline orchestration)

-**AI Models Used:**
- **Detection**: person-detection-retail-0013 (FP32)
- **Classification**: person-attributes-recognition-crossroad-0230 (FP32)

-**Input Sources:** Simulated or live camera video streams (MP4)

4. Pipeline Architecture



5. **DL Streamer GStreamer Pipeline**

```
$ docker run -it --rm   --device /dev/dri
-v /home/akshaya:/scripts
-v /home/akshaya/data:/data
-v /home/akshaya/model_proc:/home/akshaya/model_proc
-v /home/akshaya/intel/models:/models
 -e XDG_RUNTIME_DIR=/tmp/runtime-root
intel/dlstreamer:2022.1.0-ubuntu20-dpcpp
python3 /scripts/stream_benchmark3.py
```

6. **Model Evaluated**

Type	**Model Name**	**Description**
Detection	person-detection-retail-0013	Detects people in images or video streams
Classification	person-attributes-recognition-crossroad-0230	Detect various attributes of a person including gender, age, clothing and other personal characterstics

7. **Test Scenarios**
   - **Input video**: input1.mp4 (sample input video – vehicle movement and persons movements)
   - **Devices**:
     - **CPU** – Docker on WSL2 (Windows 11)
   - **Methods**: Single-stream & Multi-stream analysis
   - **Measurement**: FPS via **fpsdisplaysink**, CPU usage via **top**

8. **Experimental Results Table (CPU)**

   - The table below presents the average performance of the DL Streamer pipeline executed on CPU across varying numbers of parallel video streams. As the number of streams increases, total throughput shows modest scaling, while per-stream FPS decreases due to CPU resource contention.

Model Combination	**Average FPS**	**Maximum Streams**	**Notes(Bottlenecks)**
face-detection-0004 +age-gender-recognition-0013	38.32	~1-8	Lightweight, fast. Minor decode bottleneck under load.

- **Model Combination: face-detection-retail-0004 + age-gender-recognition-retail-0013**
  This combination delivered the best overall performance with higher FPS and stream support on CPU.

No of Streams	Average FPS (CPU)	Average Per Stream Fps	Total CPU Usage %	Notes (Bottlenecks)
1	49.28	49.28	15-25%	No bottlenecks observed. CPU comfortably handles the load
2	39.81	19.91	25-30%	Good total throughput. Light CPU contention starts
4	37.33	9.33	55-70%	Significant FPS drop per stream. Bottleneck at CPU threads and memory bandwidth
8	38.57	4.82	78-100%	CPU fully saturated. The per stream FPS is so low

9. Notes (Bottlenecks) Explanation

1 Stream: No major bottlenecks. Stable high FPS (~45–65). Minor decode variance at startup.

2 Streams: Slight CPU contention. Classification begins to limit FPS. Good balance (~18–22 FPS per stream).

4 Streams: CPU nearing saturation. Decode + inference cause per-stream FPS drop (~8–11 FPS). Uneven load across streams.

8 Streams: Heavy bottleneck. FPS drops sharply (some streams near 0). Severe imbalance. System overloaded.

10. Conclusion

The DL Streamer pipeline was executed on Ubuntu 20.04 using Docker 28.1.1 and the intel/dlstreamer:2022.1.0-ubuntu20-dpcpp image, with performance tested across 1, 2, 4, and 8 video streams. The results showed that the system performed efficiently for 1 and 2 streams, delivering high and consistent FPS with minimal latency. At 4 streams, CPU utilization increased notably, causing moderate FPS drops and some imbalance across streams. With 8 streams, the CPU became saturated, resulting in severe performance degradation and uneven distribution of processing resources. These observations suggest that the tested CPU configuration can reliably handle up to 4 streams for real-time processing. Beyond this point, performance declines significantly, and GPU acceleration or workload distribution is recommended for scalability.

Team Members: **Sai Sarath Nainala [IT / MLRITM] [237Y1A12G1@mlritm.ac.in]**
Siddi Vinayak Manikyala [IT /MLRITM] [237Y1A12G5@mlritm.ac.in]
Akshaya Nagunoori [IT /MLRITM] [237Y1A12C3@mlritm.ac.in]