INTEL UNNATI INDUSTRIAL TRAINING 2025

Problem statement 3: Create pipeline (detect, decode and classification) using DL Streamer, define system scalability for Intel HW.

Submitted By –

Reg No.	Branch	Name	Email ID
220953524	CCE	Diya Nair	diya.mitmpl2022@learner.manipal.edu
220911470	IT	Harshavardhan Reddy	vannela.mitmpl2022@learner.manipal.edu
220911676	IT	Piyush Rangdal	piyush1.mitmpl2022@learner.manipal.edu

1. Introduction

This report benchmarks the performance of Intel DL Streamer pipelines executed purely on CPU, without GPU acceleration. Two OpenVINO models were tested:

- person-vehicle-bike-detection-2004
- face-detection-0204

The focus was on evaluating:

- Average FPS (Total)
- Impact of GUI rendering (On vs Off)
- System scalability with 1, 2, and 4 concurrent input streams

2. Test Setup

- Platform: Docker-based Intel DL Streamer on Ubuntu 22.05.5 LTS
- Hardware: Intel CPU (no GPU used)
- Models Tested: person-vehicle-bike-detection-2004, face-detection-0204
- Stream Counts: 1, 2, 4 using GStreamer compositor
- Modes Evaluated: GUI Enabled, GUI Disabled
- GUI Rendering: Enabled via XLaunch for visual output
- Primary Metric: Total Average FPS

Note: The team was unable to test on Intel GPUs due to restricted access to lab infrastructure during the vacation period. As a result, all benchmarking was conducted on CPU-only pipelines. We also explored using cloud platforms such as **Azure** for remote GPU access, but the performance and stability were inadequate for our testing requirements. Future GPU-based evaluations can be conducted once physical lab access is restored.

Input Source

During testing, we used a **sample output video file** to simulate live input streams in the DL Streamer pipeline. This ensured consistent benchmarking across different configurations.

For **real-time video streaming**, tools like **yt-dlp** can be used to fetch YouTube streams and load them into GStreamer-compatible pipelines. This approach enables testing on live video feeds.

The relevant scripts and setup for using yt-dlp with DL Streamer pipelines have been uploaded to our GitHub repository.

3. Performance Results

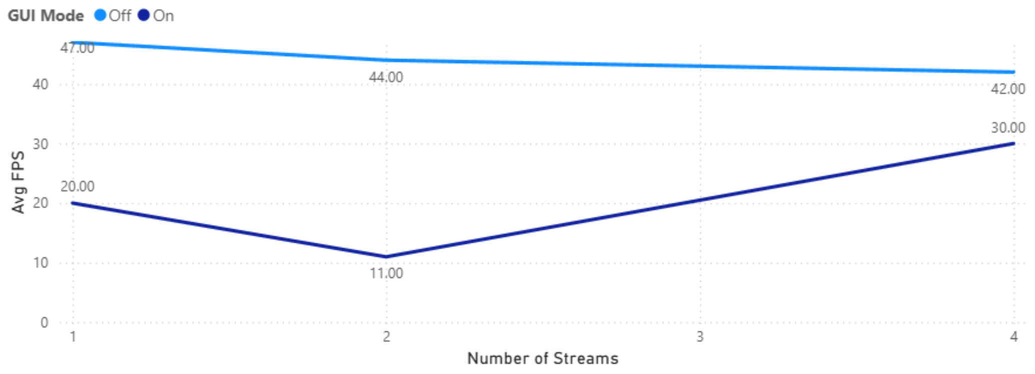

Fig 1. Graph for Number of Streams vs Average FPS in the face-detection-0204 model

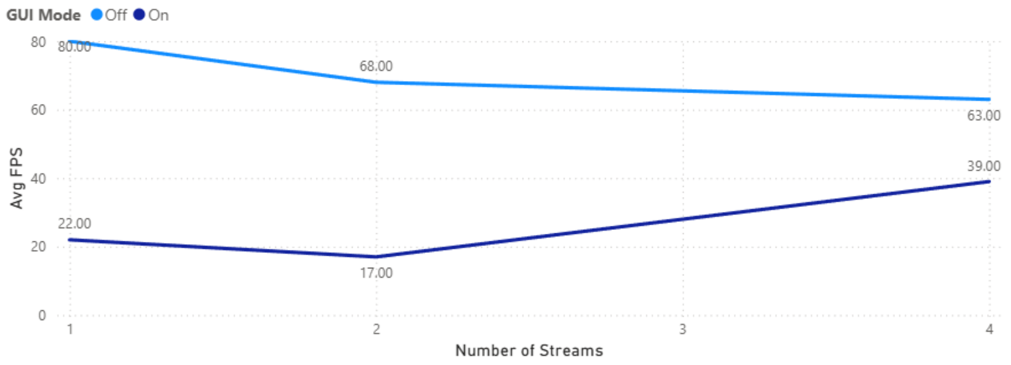

Fig 2. Graph for Number of Streams vs Average FPS in the person-vehicle-bike model

Model	GUI Mode	Streams	Avg FPS (Total)
person-vehicle-bike-detection-2004	On	1	22
		2	17
		4	39
	Off	1	80
		2	68
		4	63

face-detection-0204	On	1	20
		2	11
		4	30
	Off	1	47
		2	44
		4	42

Table 1. FPS Recorded in table format

4. Visual Output

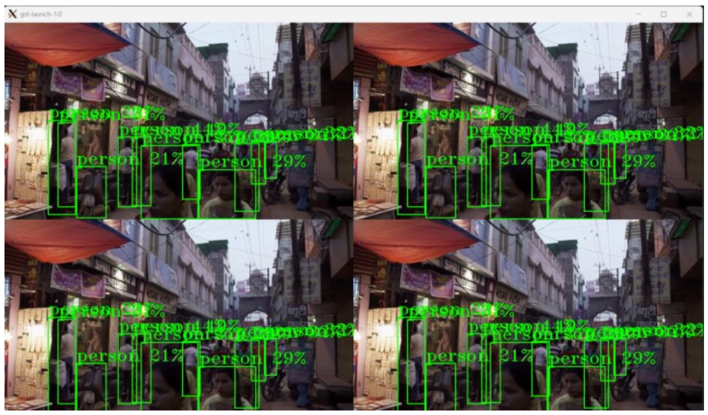

Fig 3. GUI Compositor Output for the 4 streams in person-vehicle-bike model

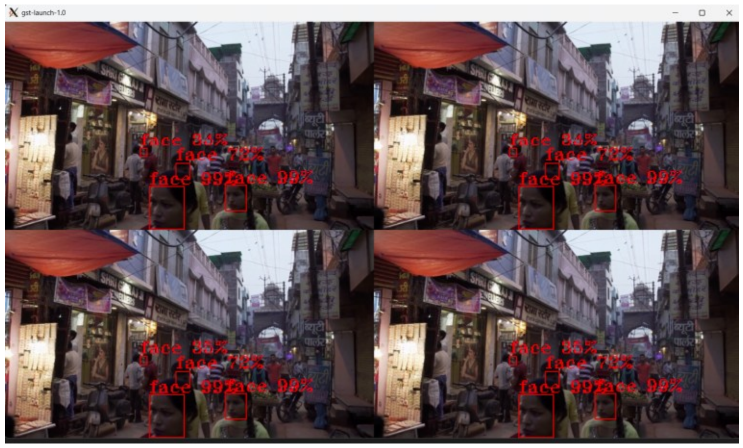

Fig 4. GUI Compositor Output for the 4 streams in face detection model

5. Conclusion

This table helps track how each model scales with more video streams and whether GUI rendering has a noticeable impact on FPS. All tests should be repeated under similar system conditions for consistency.

The **bottleneck here is the graphical rendering load** introduced by the **GUI compositor**, which significantly reduces FPS when enabled. This is evident from the large FPS differences between GUI On and Off modes for the same stream count. The overhead of displaying the visual output impacts the pipeline's throughput, especially as the number of streams increases.

For best performance and scalability in real-world deployments, the **GUI should be disabled** unless visual output is specifically required for debugging or demonstration purposes