DL Streamer Pipeline Scalability Report

# Page 1: Pipeline Overview

## 1. Introduction:

Smart cities deploy thousands of surveillance cameras.  
Manual video monitoring is impractical.  
DL Streamer enables scalable real-time video analytics by decoding video, detecting people, and classifying their attributes using Intel hardware.

## Pipeline Architecture:

Input Video Streams  
 |  
Decode -> Detection (person-detection-retail-0013) -> Classification (person-attributes-recognition-crossroad-0230)  
 |  
Console Output (Real-time FPS Monitoring)

## Configuration & Execution:

\* Docker Image: intel/dlstreamer:latest  
\* Models:  
 - Detection: person-detection-retail-0013  
 - Classification: person-attributes-recognition-crossroad-0230  
  
\* Command Example:  
docker run -it --rm \  
 -v /root/dlstreamer\_project/video.mp4:/data/video.mp4 \  
 -v /root/dlstreamer\_project/models/detection:/models/detection \  
 -v /root/dlstreamer\_project/models/classification:/models/classification \  
 -v /root/dlstreamer\_project/scripts:/scripts \  
 intel/dlstreamer:latest \  
 bash -c "python3 /scripts/dlstreamer\_multi.py --streams <N>"  
  
\* Replace <N> with the number of parallel streams.

# Page 2: DL Streamer Pipeline Scalability Report

## Summary Table: FPS by Stream Count

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Streams | Total FPS | FPS/Stream | Max Burst FPS | Notes |
| 1 | 1170.14 | 1170.14 | 2438.29 | Max performance |
| 4 | 253.49 | 63.37 | 1327.19 | Slight contention |
| 8 | 532.67 | 66.58 | 2438.29 | Best scalability |

## Performance Phases (8-Stream Example)

|  |  |  |
| --- | --- | --- |
| Phase | Total FPS | Per-Stream FPS / Notes |
| Warm-up | 6.85-489.05 | 0.86-61.13 / Pipeline load time |
| Stable | 350-550 | 40-70 / Real-time inference |
| Bursts | Up to 2438.29 | Up to 304.79 / Async optimization |
| Dips | 120-200 | 15-30 / Contention/memory delays |

## Bottleneck Analysis:

Sustained avg = 532.67 fps  
Peak observed = 2438.29 fps  
Utilization = 21.85%  
Bottleneck = 78.15%

## Stage-wise Bottlenecks

|  |  |  |
| --- | --- | --- |
| Stage | Bottleneck | Explanation |
| Decode | Medium | Based on resolution |
| Detection | High | Main compute workload |
| Classification | Medium | Secondary load |
| I/O | Low | Minor disk delays |

# Page 3: Scalability Comparison

## Scalability Metrics Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metric | 1 Stream | 4 Streams | 8 Streams | Notes |
| Total FPS | 1170.14 | 253.49 | 532.67 |  |
| Per-Stream FPS | 1170.14 | 63.37 | 66.58 |  |
| Max Burst FPS | 2438.29 | 1327.19 | 2438.29 |  |
| Stability | Very High | Moderate | Very Good |  |
| Bottleneck % | ~52.00% | ~80.90% | ~78.15% |  |

## Recommendations:

- Use INT8 quantized models for speed  
- Enable GPU/VPU (OpenVINO) to offload compute  
- Tune batch sizes for latency vs throughput  
- Monitor CPU/memory (htop, perf)  
- Reduce I/O delays via tmpfs or memory disks  
- Add log timestamps to trace system correlation

## Conclusion:

1 Stream: Shows max FPS (1170.14), pipeline limits.  
4 Streams: Slight drop, manageable performance.  
8 Streams: Best balance with 66.58 fps/stream, 532.67 total.  
DL Streamer on CPU alone provides strong parallel inference.  
Detection remains the biggest bottleneck.  
OpenVINO or GPU use recommended for further scaling.