Create pipeline (detect, decode and classification) using DL Streamer, define system scalability for Intel HW

Category: AI, machine learning, System scalability

Description:
As deployment of Edge and AI are growing, City and transportation are adopting new use-cases where more and more visual cameras are deploying across cites. It is very hard to manual scan all the cameras ’feeds. AI is helping to decode, detect and classify those cameras feeds and providing analytics.

Team Members

Anushka Gupta

B.Tech CSE - SWE
GT

Roll No: RA2311033010120

Email: anushka.connect05@gmail.com

Vansh Rajvanshi

B.Tech CSE -

Roll No: RA2311051010031

Email: vr1932@srmist.edu.in

Faculty Mentor

Dr. G. Saranya
Assistant Professor
Department of Networking and Communications
School of Computing, SRM IST
Email: saranyag3@srmist.edu.in

Create pipeline (detect, decode and classification) using DL Streamer, define system scalability for Intel HW

Abstract

A large-scale study of a decode, detect, and classify pipeline that uses DL Streamer on Intel i3-10100F hardware reports the results of a systematic evaluation that was done to determine the operational feasibility within edge and city-scale camera settings. The performance metrics was observed throughly which included frame rates, latency, and resource use which were measured out under multiple stream conditions while both advanced and basic person detection and attribute classification models were used . This study also looked at the trade-off between throughput and latency which was required for successful deployment in which consistent accuracy and resource efficiency are a must. The advanced models were benchmarked against basic models at many load conditions The operational thresholds were determined which are tailored for city-scale surveillance, institutional monitoring, and traffic management systems which require AI-enabled video analysis on CPU-based infrastructure which does not have a GPU. A very organised approach was placed to test which in turn gives data that is driven by facts related to how the pipeline scales under increased loads and also what can be done to maintain stable operation and at the same time perform in real time. A pipeline configuration was put together that is being used by GLStreamer and OpenVINO which shows that it has the ability to put together reliable detection and classification at practical frame rates into very tight hardware constraints. It can be seen from the evaluation of the GLStreamer integration which was done via Flask that is on track for live deployment through which it would prove pipeline stability, response consistency and that could be integrated within automated monitoring systems which require clean, well thought out and very dependable reports out of the system

Introduction

Automated video analysis is at the core of what is seen in city scale surveillance, traffic monitoring, and event based crowd management. Manually watching multiple camera feeds introduces delay and in between the lines what is seen is a break in the chain of response which in turn affects how decisions are made and our ability to react. Many institutional and urban settings do not have the large scale AI processing which their camera infrastructure requires which in turn leaves a lot of that network under used and are still dependent on human resources for the most part of interpretation and monitoring. It is also seen that in many cases we are not able to use AI for real time video analysis due to hardware constraints which in turn play a role in what we are able to do with these systems and as a result we are left with some issues unresolved and slow responses in very important settings.

In present day it is seen many of the used out of the box detection models trade off between accuracy and speed or they need large scale GPU resources for better performance which in turn raises the bar for deployment which is at the same time complex and expensive. This is a issue for us in that we see the integration of very effective AI into video analysis is put at a standstill in very constrained resources settings which at the same time require that we have that which is consistent in terms of throughput and accuracy without also going out and procuring more GPU infrastructure. We see a gap between what CPU resources we have and what the needs are of these advanced detection and classification models that in turn is a issue for the scale up of AI in monitoring systems ’which in large number of cases is to put camera feeds which could be very useful in real time at best and at worst is to use up band with and storage but not come through with that which is operationally useful. Methodology.

DL Streamer which presents a framework for deployment of top level detection and classification pipelines on CPU based systems, a lot of improvement can be seen in inference time via FP16 precision without sacrifice of detection accuracy. Testing was done on Intel i3-10100f hardware with both advanced and baseline models which we ran through systematic benchmarking at varying stream counts which in turn identified performance tradeoffs and also gave us a picture of achievable frame rates and latency for scalable deployments. A structured pipeline evaluation was conducted which in turn enabled clear planning and practical deployment of AI enabled camera analytics, which in turn was done to ensure getting reliable and actionable outputs in operationally constrained environs also minimising additional hardware requirements.

Proposed Methodology

DL Streamer was put in with OpenVINO to run decode, detect and classify pipelines in structured test scenarios which was done to study system performance at varied stream counts. FP16 precision was used to improve compute efficiency which at the same time. Detection and classification accuracy was preserved for both advanced and baseline models. The baseline models included **face-detection-retail-0004** and **person-attributes-recognition-crossroad-0234** and for advanced the **person-detection-0200** and **human-pose-estimation-0001** were chosen. Observation was done at stream counts of 1, 2, 4, 8, 12, and 16 which was done to systematics capture on frame rates (FPS), median and average latency, and CPU usage to in turn present a picture of system scalability and operational stability as the load increased.

DL Streamer with OpenVINO, testing FP16 precision pipelines across different stream counts.

Baseline models:
face-detection-retail-0004
person-attributes-recognition-crossroad-0234

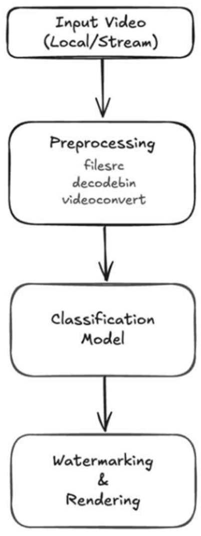
Advanced models:
person-detection-0200
human-pose-estimation-0001

Tests ran under varied n-streams configurations (1, 2, 4, 8, 12, 16) to track FPS, median and average latency, and CPU utilisation while observing stability. Evaluation was done of the Flask + GLStreamer integration to confirm live deployment readiness under real conditions.

Task	Model Name	Purpose
Person Detection	person-detection-0200	Detects people in video frames (SSD-based)
Person Attribute Recognition	person-attributes-recognition-crossroad-0234	Recognizes gender, clothing, etc.
Face Detection	face-detection-retail-0004	Detects faces in input streams
Face Re-identification	face-reidentification-retail-0095	Matches detected faces across frames/cameras
Human Pose Estimation	human-pose-estimation-0001	Detects key points (joints) in human figures

Architecture Diagram

The architecture diagram presents in detail the structure of the DL Streamer pipeline for AI camera uses. Input video streams enter the system, go to Preprocessing for decoding and format preparation. From there the_frames go through Core Operations which have in them the detection and classification models that analyse each frame for persons and attributes.
Next in Post processing addition in overlays and preparing the data for storage or display. At the end which is the Final stage data goes to Output for live monitoring or to Storage for at a later date. This structured process supports stable and scalable performance of real time and archived AI video analysis on CPU based systems.



1.3 *architecture diagram*

Workflow

1. Decode video streams
2. Run SSD person detection
3. Classify key person attributes
4. Watermark overlays
5. Display or store output

Example Testing Command:

```
gst-launch-1.0 filesrc location=input.mp4 ! decodebin ! videoconvert ! \
gvadetect model=models/person-detection-0200.xml device=CPU batch-size=1 nireq=2 ! \
gvaclassify model=models/person-attributes-recognition-crossroad-0234.xml device=CPU
batch-size=1 nireq=2 ! \
gvawatermark ! videoconvert ! fpsdisplaysink sync=false
```

Hardware Used

- **CPU:** Intel Core i3-10100f (4 cores, 8 threads)
- No GPU used, CPU-only pipeline
- Memory: 16GB DDR4

Software Stack

- **OS:** Ubuntu 22.04
- **OpenVINO 2025**
- **DL Streamer**
- **GLStreamer 1.0**
- Flask for web interface testing

Optimization

Performance in a controlled test environment is measured for FPS and latency at different stream counts which was done for baseline and advanced models. The performance analysis data which in turn gave insight into system scalability, throughput consistency and latency behaviour which in turn identifies operational thresholds for scalable deployment.
Graphical depictions of the scaling results which also present how the pipeline keeps to its performance marks as there is an increase the number of streams and at the same time point out when system resources run low.

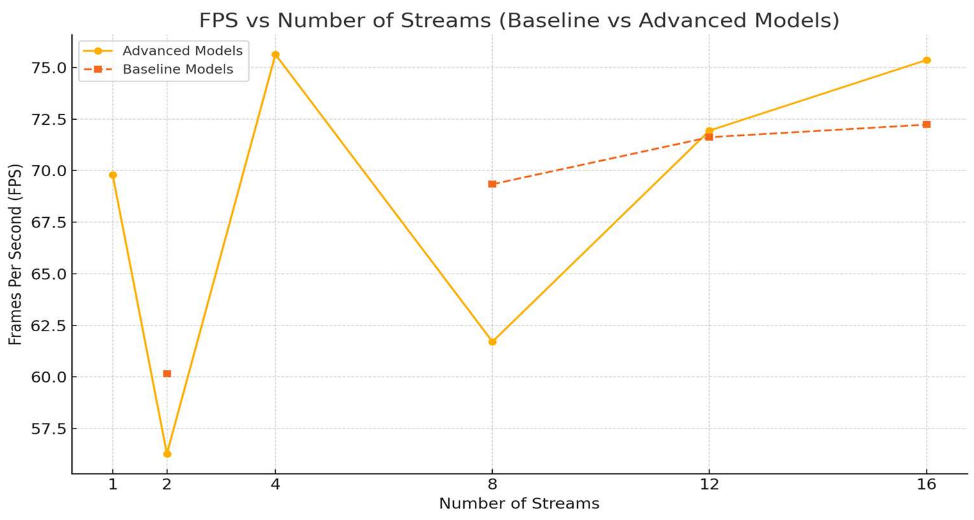


fig 1.1

Graph - FPS vs Streams for Baseline and Advanced Models

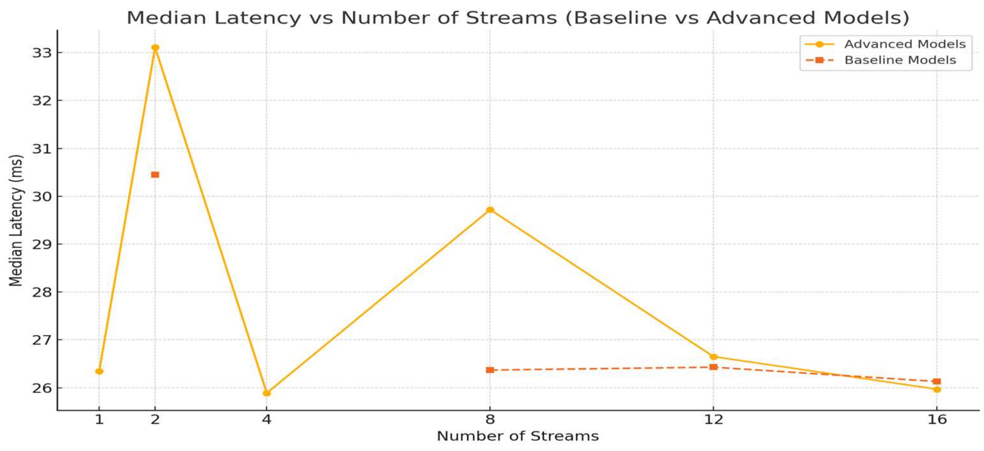


fig1.2

Graph - Median Latency vs Streams for Baseline and Advanced Models

Links

- YouTube demo: https://youtu.be/XXUsDnoPa54
- GitHub pipeline repo: https://github.com/Vansh-Raj/pipeline-dlstreamer.git
- **References**
- Intel DL Streamer documentation: https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/dl-streamer.html
- OpenVINO model zoo: https://www.openvinotoolkit.org/model-zoo/