**Introduction**

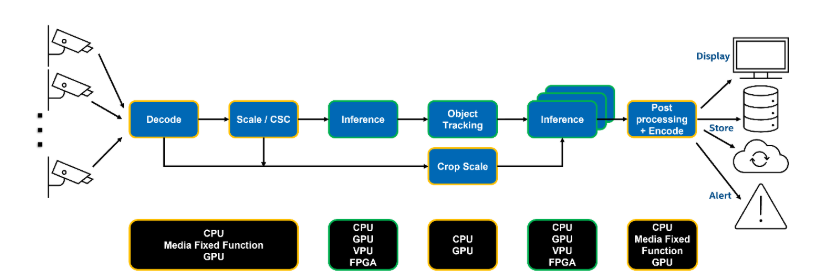
DLStreamer is an open-source framework designed to facilitate the development and deployment of deep learning inference pipelines, particularly for video analytics and computer vision applications. Built a top GStreamer, DLStreamer provides a flexible and efficient way to integrate multiple deep learning models and media processing components into a unified pipeline, enabling real-time processing and inference on video streams.

The pipeline architecture in DLStreamer allows developers to chain together various processing elements, such as video decoders, pre-processing modules, inference engines, post-processing units, and output sinks. By leveraging the modularity of GStreamer, DLStreamer supports a wide range of input sources (including files, cameras, and network streams), diverse hardware accelerations (such as CPUs, GPUs, and VPUs), and interoperability with different deep learning frameworks (like TensorFlow, PyTorch, and OpenVINO).

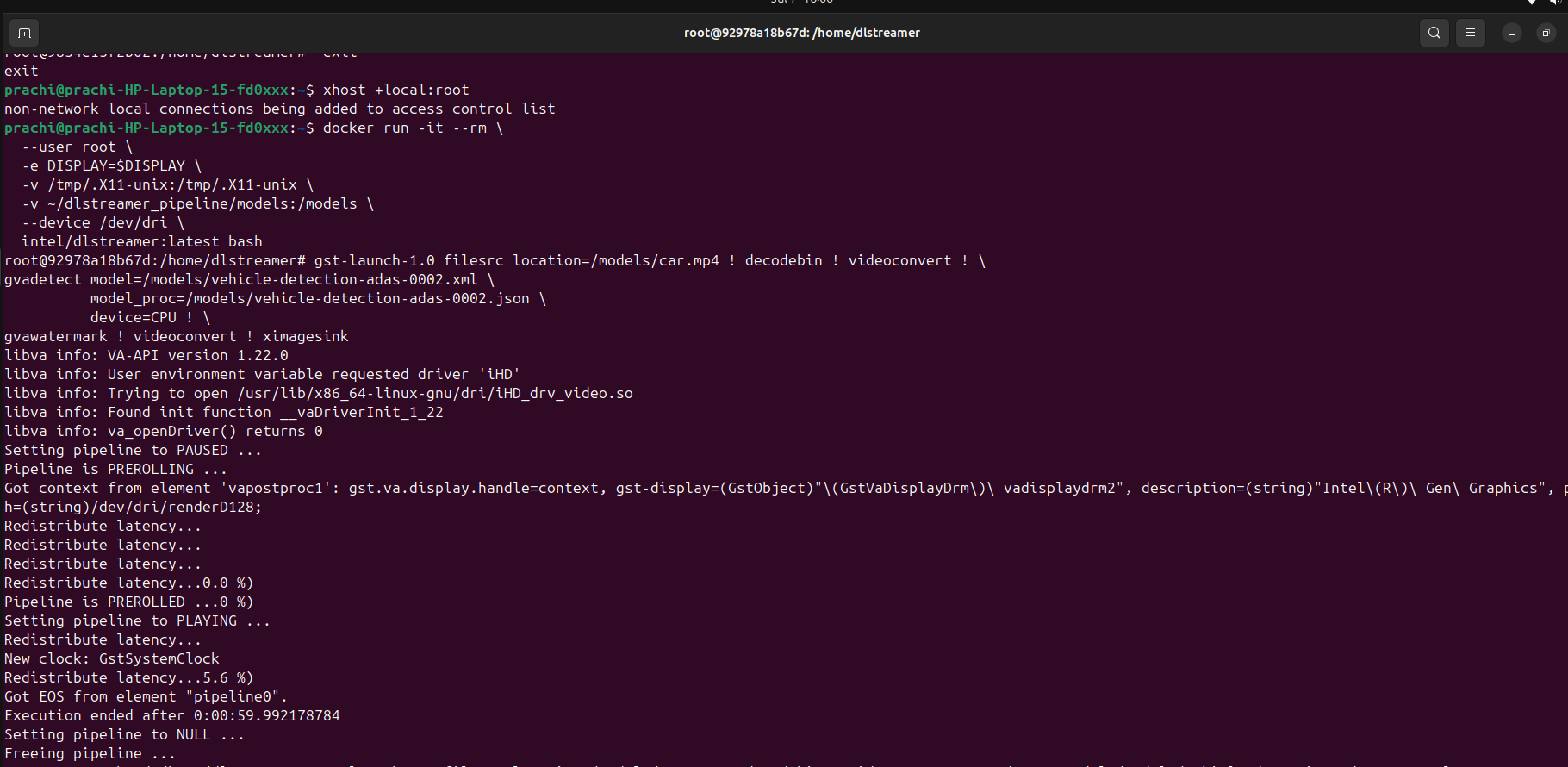
**Abstract**

This report presents the development and implementation of a deep learning inference pipeline utilizing DLStreamer, an open-source framework built on top of GStreamer. The pipeline is designed to process video streams in real time, enabling efficient integration of multiple neural network models for tasks such as object detection, classification, and tracking. By leveraging the modular architecture of DLStreamer, the solution achieves high flexibility, scalability, and performance across diverse hardware platforms.

**Architecture Diagram :-**



**Output:-**







**Link for Output Video:-**

<https://drive.google.com/file/d/1K3bAsvaFFJioGWMQq5oxqbPIav_cqSXz/view?usp=drivesdk>

**Reason For GPU and CPU and Bottleneck :-**

**CPU Usage:**

The CPU typically handles general-purpose processing tasks such as pipeline orchestration, data input/output, pre-processing (e.g., decoding video streams, resizing, color conversion), and post-processing (e.g., drawing bounding boxes, encoding output). Certain lightweight inference models or operations that do not benefit much from parallelization may also run efficiently on the CPU.

**GPU Usage:**

The GPU excels at highly parallelizable tasks, such as deep learning inference for computer vision models. Offloading these compute-intensive operations to the GPU accelerates the overall pipeline, enabling real-time performance, especially with high-resolution or multiple video streams.

**Data Transfer Overhead:**

Transferring data between CPU and GPU memory (via the PCIe bus or integrated memory subsystems) can introduce latency. Excessive transfers or inefficient batching can significantly slow down the pipeline.

**I/O Limitations:**

Reading from or writing to storage, or handling high-bandwidth video streams, can also constrain pipeline performance if I/O throughput is insufficient.

**For Extra information (Github Link):-**

<https://github.com/SskshiKore/Create-a-pipeline-detect-decode-and-classification-using-DL-Streamer-for-intel-hw>