Multi-Cam AI Video Analytics
Using Nvidia DeepStream SDK 7.1

Abhiram Anil[1], Agilan Vellalore Saminathadurairaj[1], Dheeraj Swaroop Saligrama Mahesh[1], Divya Ramesh Teli[1], Farzad Mehrdad[1], Prajwal Mangaluru[1], Shilpa Gopakumar[1], Thomas Schumann[2]
[1]Master Student at Hochschule Darmstadt/University of Applied Sciences, Germany
[2]Faculty of Electrical Engineering, Hochschule Darmstadt/University of Applied Sciences, Germany

Abstract—**This paper implements a real-time, multi-camera anomaly detection system using the NVIDIA DeepStream SDK 7.1 on the Jetson Orin Nano platform. The pipeline was extended to support live streams from Intel RealSense cameras via the GStreamer plugin v4l2src, enabling AI-driven video analytics at 1280×720 and 30 FPS. Inference is handled by nvinfer or nvinferserver (plugins from NVIDIA DeepStream SDK), with motion detection using dsdirection. A custom shell script was developed to serve as a wrapper layer, automating multi-camera setup, GStreamer previews, and parallel processing, while also managing device control and synchronization. A dual-view display shows both raw and processed outputs. GPU benchmarking on Jetson Orin Nano with two cameras showed 98.4% usage at high resolution and lower load at reduced resolutions, demonstrating efficiency for low-cost Edge AI.**

I. INTRODUCTION

This paper introduces a real-time, multi-camera anomaly detection system optimized for scalable deployment on the NVIDIA Jetson platform, as shown in Fig. 1, using DeepStream SDK 7.1. The project relies on the Intel® RealSense™ D455 cameras for their excellent depth-sensing capabilities and compatibility, which are essential for achieving the sensor fusion goal. The original DeepStream reference application [1], previously limited to offline MP4 file inputs, has been extended to support live camera streams. This was achieved by integrating v4l2src and adapting configuration files for /dev/video* devices. The enhanced pipeline enables real-time, AI-driven video analytics. The system leverages nvinfer or nvinferserver for deep learning inference and dsdirection for optical flow-based motion analysis, facilitating robust detection of anomalous behaviors.

The Jetson platform serves as the central processing unit, efficiently executing the machine learning models required for robust object tracking and analysis. This setup enables the rapid prototyping and deployment of complex AI vision applications directly at the edge.

Fig. 1: NVIDIA Jetson Orin Nano

II. BENCHMARKING

A. Process

Benchmarking was performed using the jtop plugin [2] over the ResNet-18 anomaly detection model deployed on the NVIDIA Jetson Orin Nano under the DeepStream SDK 7.1 environment. Various input resolutions and frame rates were tested to evaluate the pipeline's performance and resource utilization.

B. Results on Jetson Orin Nano

Table 1 shows a comparison between benchmarks; GPU usage remains consistently high at higher resolutions, reaching up to 99.1% for 1280×800 at 30 fps, while dropping to 49.7% for the lowest resolution of 424×240 at 30 fps. CPU usage follows a similar trend, ranging from around 34% at higher resolutions down to approximately 10–12% at lower resolutions. Memory consumption remains stable between 3.3G and 3.7G across all tests. Power consumption varies modestly, ranging from 1.8W at high resolutions to as low as 1.3W at lower frame rates. These results confirm that the ResNet-18 application-specific model is capable of real-time inference even at reduced resolutions and resource footprints, enabling flexible deployment scenarios on edge hardware like Jetson Orin Nano.

Table 1: Benchmark Results

Parameter	DeepStream SDK 7.1 - ResNet18 (Nvidia Jetson Orin Nano with 2 cameras)					
Resolutions & Frame rates	1280x800 @30fps	1280x720 @30fps	848x480 @30fps	640x350 @30fps	640x360 @15fps	424x240 @30fps
GPU Usage	99.1%	98.4%	75.8%	66.6%	63.4%	49.7%
CPU Usage	34.6%	33.5%	33.2%	10.7%	11.1%	12%
Memory Usage (RAM)	3.4GB	3.5GB	3.5GB	3.6GB	3.6GB	3.7GB
Power Consumption	1.8W	1.8W	1.7W	1.7W	1.3W	1.7W

Swap memory usage is zero in all cases.

III. IMPLEMENTATION

A. Reference Framework and Environment Setup

This model is based on NVIDIA's DeepStream SDK 7.1 Anomaly Detection Reference App [1], which demonstrates real-time anomaly detection in video streams using NVIDIA Optical Flow (nvof) and visualization (nvofvisual) plugins. The system was imaged with latest JetPack and DeepStream SDK versions aligned with DeepStream 7.1 requirements

[3], ensuring compatibility, improved performance, and stability throughout the anomaly detection workflow.

NVIDIA's DeepStream SDK provides a high-performance, modular pipeline for real-time video analytics. Video streams are decoded and batched using the nvstreammux plugin, then passed through AI inference models such as ResNet-18 via nvinfer or nvinferserver, depending on the chosen deep learning framework. Object tracking is performed using the nvtracker plugin, while visual annotations and bounding boxes are rendered through the nvosd plugin for display. This architecture enables scalable deployment of intelligent video analytics at the edge, as shown in Fig. 2.

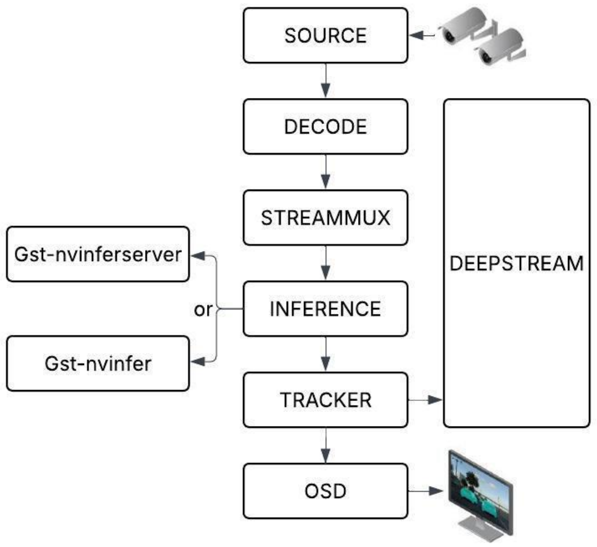

Fig. 2: Multi-Camera AI Analytics Architecture with DeepStream 7.1

B. Model Configuration and Customization

The DeepStream Anomaly Detection Reference App does not support pre-generated engine files. To overcome this, model metadata was extracted from the ONNX file to identify the input tensor (input_1:0). A custom TensorRT engine was manually generated for a ResNet-18 model specifically trained for anomaly detection, with a batch size set to 30 [4]. ResNet-18 was selected for its effective balance of accuracy and computational efficiency in detecting motion-based anomalies. Additionally, compiling the gst-dsdirection plugin succeeded by explicitly specifying CUDA version 12.6 in the Makefile [5], ensuring compatibility with the updated environment.

C. Pipeline Adaptation for Live Streaming

Initially, the pipeline processed offline MP4 files using uridecodebin for automatic demuxing and decoding, but it was limited to non-live scenarios. To support real-time anomaly detection, the pipeline was adapted to accept live streams from Intel RealSense cameras using v4l2src [6]. This required manual configuration of elements such as capsfilter, videoconvert, and nvvideoconvert for proper format adaptation, along with explicit linking of pipeline elements. Resolution (e.g., 1280×720) and framerate (e.g., 30 FPS) were set, and the live-source flag in nvstreammux was enabled to support live processing. A shell script was developed to automate launching DeepStream pipelines on two RealSense cameras simultaneously, managing parallel execution across multiple terminals (see Fig. 3) and starting GStreamer-based previews [7]. Furthermore, a visualization enhancement was introduced to display raw camera streams alongside AI-processed and depth-sensing outputs. This dual-view setup greatly improved debugging, validation, and clarity during live demonstrations.

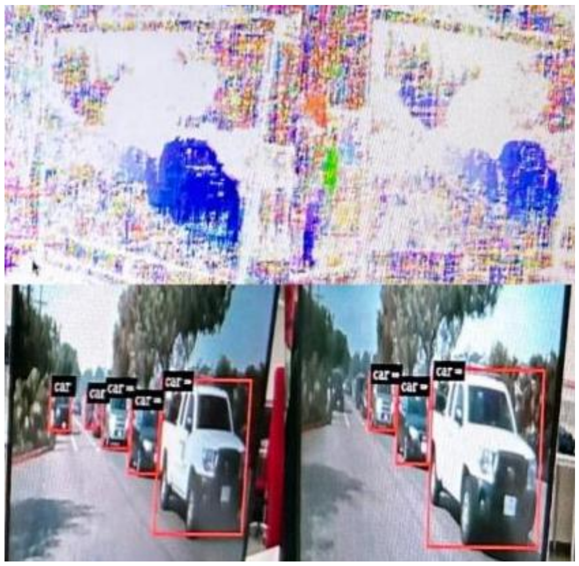

Fig. 3: Real-time Anomaly Detection Outputs

IV. CONCLUSION

This paper presents a real-time, multi-camera anomaly detection pipeline using NVIDIA DeepStream SDK 7.1 and a custom ResNet-18 model. It processes live video from Intel RealSense cameras, automates input handling, and displays both raw and AI-processed outputs. While efficient for edge deployment, the use of ResNet-18 limits the detection of complex anomalies, and depth data from RealSense is underutilized.

Future work includes adopting stronger models like YOLOv7 [8] and integrating depth analysis to improve accuracy and spatial understanding for real-world surveillance.

REFERENCES

[1] NVIDIA AI IOT DeepStream Reference Apps – Anomaly Detection. https://github.com/NVIDIA-AI-IOT/deepstream_reference_apps/blob/master/anomaly/README.md
[2] jtop – jetson-stats: Real-time System Monitoring for NVIDIA Jetson Devices. https://rnext.it/jetson_stats/jtop/jtop.html
[3] NVIDIA Corporation. DeepStream SDK 7.1 Documentation. https://docs.nvidia.com/metropolis/deepstream/dev-guide/index.html
[4] NVIDIA TensorRT Developer Guide. https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html
[5] NVIDIA CUDA Toolkit Documentation. https://docs.nvidia.com/cuda/
[6] Intel RealSense Technology Overview. https://www.intelrealsense.com/
[7] GStreamer Documentation. https://gstreamer.freedesktop.org/documentation/
[8] YOLOv7 Paper: https://arxiv.org/abs/2207.02696