h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Team Project – SS2025

# Multi-Cam AI Video Analytics on Nvidia Jetson Orin Nano (Object Tracking + Anomaly Detection using 2 cameras)

Guide

Prof. Dr. Thomas Schumann

Team

Abhiram Anil

Agilan Vellalore Saminathadurairaj

Dheeraj Swaroop Saligrama Mahesh

Divya Ramesh Teli

Farzad Mehrdad

Prajwal Mangaluru

Shilpa Gopakumar

## Acknowledgement

We would like to express our heartfelt gratitude to Prof. Dr. Thomas Schumann for his exceptional guidance and for providing us with the necessary facilities to complete this project successfully. His insightful advice and expert knowledge were invaluable at every stage, helping us navigate challenges and refine our approach. The resources and tools he made available greatly facilitated our work, allowing us to explore and implement our ideas with confidence.

Prof. Dr. Schumann’s unwavering support and encouragement have been a constant source of motivation, inspiring us to push the boundaries of our understanding and achieve our best. The opportunity to work under his mentorship has been an enriching experience, and we are sincerely thankful for his dedication and commitment to our academic growth.

# Table of Contents

# 1. Introduction

This document provides a comprehensive overview of the Team Project titled "Multi-Cam AI Video Analytics on Nvidia Jetson Orin Nano: Object Tracking and Anomaly Detection using Dual Cameras." This project is part of the Master of Science in Electrical Engineering and Information Technology program at Hochschule Darmstadt. The project focuses on implementing Sensor Fusion technology by integrating two Intel® RealSense™ Depth Cameras D455 with Nvidia Jetson Orin Nano platform. This system facilitates real-time object tracking and enhances anomaly detection through sensor fusion, enabling advanced Multi-Cam AI Video Analytics applications. The project capitalizes on Nvidia platforms to innovate and deploy on-device machine learning models for high-precision object detection and tracking across multiple scenarios.

# 2. Project Scope

The project's primary goal is to demonstrate the capabilities of sensor fusion and object tracking using the Nvidia Jetson Orin Nano. The project utilizes dual Intel® RealSense™ Depth Cameras D455 to achieve accurate object tracking in real-time. This implementation focuses on the rapid prototyping of on-device Machine Learning models, with the Nvidia Jetson platform serving as the backbone for processing and executing the model efficiently. The Intel® RealSense™ D455 was selected for its compatibility and superior depth-sensing capabilities, making it ideal for this project's sensor fusion objective.

# 3. Hardware Components

## 3.1 NVIDIA Jetson Orin Nano

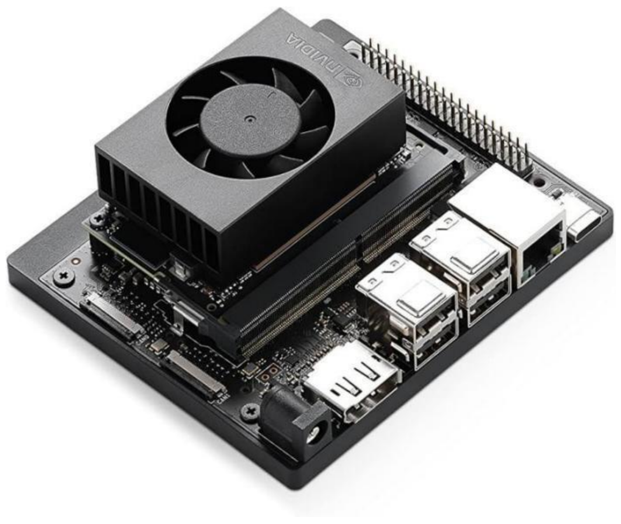


Figure 1: NVIDIA Jetson Orin Nano

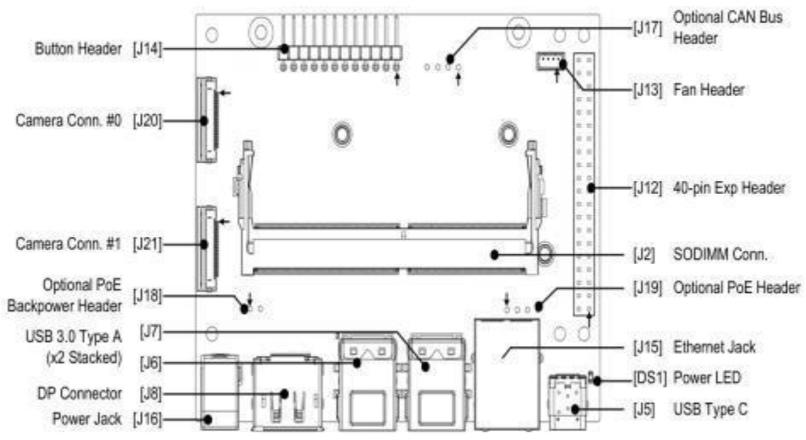
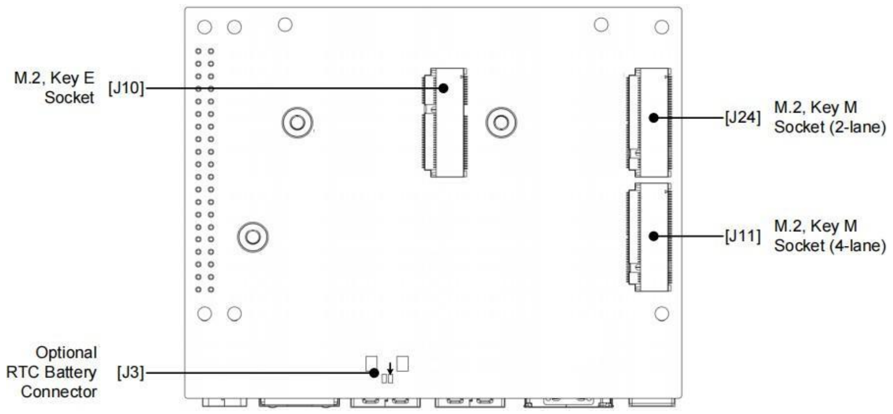


Figure 2: Jetson Orin Nano Carrier Board Placement – Top View



M.2, Key E Socket [J10]
[J24] M.2, Key M Socket (2-lane)
[J11] M.2, Key M Socket (4-lane)
Optional RTC Battery Connector [J3]

Figure 3: Jetson Orin Nano Carrier Board Placement – Bottom View

The developer kit includes a Jetson Orin Nano 8GB module and a reference carrier board that can support all Jetson Orin Nano and Orin NX modules, providing an ideal platform for prototyping next-gen edge AI products. The Jetson Orin Nano 8GB module features an NVIDIA® Ampere GPU (with 1024 CUDA® cores and 32 third-generation Tensor cores) and a 6-core ARM CPU, capable of running multiple concurrent AI application pipelines and delivering high inference performance.

### 3.1.1 Interface details

- [J2] Jetson Orin Nano Conn. (SODIMM, 260-pin)
- [J15] RJ45 Ethernet Socket, 18-pins, RA, Female
- [J5] USB Type C
- USB 3.2 Gen1 Type-A (J6, J7)
- [J16] Power Jack
- [J6] USB Type A Dual Stacked Connector
- [J17] Optional: CAN Bus Header (1x4, 2.54 mm pitch, RA)
- [J7] USB Type A Dual Stacked Connector
- [J18] Optional: PoE Backpower Header (1x2, 2.54 mm pitch)

- [J8] DisplayPort Connector
- [J19] Optional: PoE Header (1x4, 2.54 mm pitch)
- [J12] 40-pin Expansion Header (2x20, 2.54 mm pitch)
- [J20] Camera (#0) Connector (22 pos, 0.5 mm pitch)
- [J13] Fan Header (4-pin, 1.25 mm pitch)
- [J21] Camera (#1) Connector (22 pos, 0.5 mm pitch)
- [J14] Button Header (1x12, 2.54 mm pitch, RA) DS1 Power LED (Green)
- [J3] Optional: RTC Back-up Coin Cell Socket (CR1225)
- [J11] M.2 Key M Slot (75-pin)
- [J10] M.2 Key E Connectivity Slot (75-pin)
- [J24] M.2 Key M Slot (75-pin)

### 3.1.2 Power Guide

**DC Power Jack**

The Jetson Orin Nano carrier board uses a DC power jack (J16) to bring in the power from the included DC power supply. The jack used on the carrier board is a Singatron Enterprise part (part #: 2DC-0005D206F). The mating barrel jack connector dimensions are:

- Barrel length: 9.5 mm
- Barrel diameter: 5.5 mm
- Pin receptacle: Accepts 2.5 mm jack pin
- The center pin is positive (+V)
- Max current supported is 3.5 A

**Optional Power-Over Ethernet and Backpower Headers**

The Jetson Orin Nano carrier board provides an option for an alternate main power input (besides the DC power jack). A 4-pin Power over Ethernet (PoE) header (J19 – 1x4 male, 2.54 mm pitch) brings out the VC power pins of the Ethernet connector. In addition, a 2-pin Backpower header (J18 – 1x2 male, 2.54 mm pitch) provides an alternate path for the input voltage (3 A max). To use this alternate PoE power mechanism, the design will require a power converter to take the high-voltage PoE supply (38 V-60 V) and convert it to the 5 V-20 V input the carrier board requires.

### 3.1.3 Features

- Applications processor (AP): NVIDIA Orin™
- Memory: 8 GB 128-bit wide LPDDR5 DRAM (68 GB/s), Micro SD card socket (UHS-1)
- Swap Memory: Configurable, typically set up to 4 GB or higher depending on the storage available (can be increased as needed).
- GPU:512-core NVIDIA Ampere architecture GPU with 16 Tensor Cores
- Memory Bandwidth: 51.2 GB/s

- Networking: 10/100/1000 BASE-T Ethernet
- Advanced power management (APM)
- Dynamic voltage and frequency scaling
- Multiple clock and power domains

The carrier board included in the developer kit comes with a wide array of connectors, including two MIPI CSI connectors supporting camera modules with up to 4-lanes, allowing higher resolutions and frame rates. The Jetson Orin Nano carrier board is ideal for software development within the Linux environment. Standard connectors are used to access Jetson Orin Nano features and interfaces, enabling a highly flexible and extensible development platform.

The following is a list of features for the carrier board.

- Connection to Jetson Orin Nano: 260-pin SO-DIMM connector
- USB: USB-C Supports Recovery Mode, USB 3.2 (Gen2x1) Hub to 4x Type A (host only)
- Wired network: Gigabit Ethernet (RJ45 connector with LEDs and optional PoE header)
- Display: VESA® DisplayPort™ (DP v1.2 (+MST) and eDP v1.4)
- Camera connectors: 2x 22-position flex connectors, CSI (2.5 Gbps per pair): 1, x2 or x4 and 1, x2, Camera CLK, I2C, and control
- M.2 Key E connector: PCIe (Gen3) x1 Lane, USB 2.0, I2S, UART, I2C Control
- M.2 Key M connector (x2): #1) PCIe (Gen3) x4 lane, control, #2) PCIe (Gen3) x2 lane, control
- Expansion header: 40-pin (2x20) header, I2C (x2), SPI (x2), UART, I2S, audio clock, GPIOs, PWMs
- UI and indicators: Button header: Power, reset, force recovery, debug UART, Auto-Power-On disable, LEDs
- Miscellaneous: Fan connector: 5V, PWM, and tach, Optional RTC back-up connector, Optional CAN header
- Power: DC Jack: 9-20 V (19V supply provided), Optional Ethernet PoE and back power headers, Main 5.0 V Pre-regulator: GS9230 (or A0Z2264) - Provides VDD_IN to module, Main 3.3V supply: GS9230 (or A0Z2264), Main 1.8V supply: GS7116S5, 3.3V AO (always on) supply: GS7116S5, USB VBUS load switches: AP22811AW5-7 (x2), DP 3.3V power switch: APL3552ABI-TRG.
- Developer kit operating temperature range: 0ºC to 35ºC

### 3.1.4 Benefits

- Up to 80X the AI performance of the previous generation NVIDIA® Jetson Nano™
- Supports a wide variety of AI models, including transformers and advanced robotics models.
- Faster Time-to-Market with the NVIDIA AI software stack.

## 3.2 Intel® RealSense™ Depth Camera D455

Figure 4: Intel® RealSense™ Depth Camera D455

### 3.2.1 Features

- Intel® RealSense™ Vision Processor D4
- Up to 1280 x 720 stereo depth resolution
- Up to 1280 x 800 RGB resolution
- Diagonal field of view over 90°
- Dual global shutter sensors for up to 90 FPS depth streaming
- RGB global shutter sensor for up to 90 FPS
- Range 0.4 m to over 6 m (varies with lighting conditions)
- Inertial Measurement Unit (IMU) for 6 Degrees of Freedom (6DoF) data
- D455f camera has IR-pass filter

### 3.2.2 Camera System Block Diagram

The camera system has two main components, Vision Processor D4 and Depth module. The Vision Processor D4 is either on the host processor motherboard or on a discrete board with either a USB2.0/USB 3.1 Gen 1 or a MIPI connection to the host processor. The Depth module incorporates left and right imagers for stereo vision with the optional IR projector and RGB color sensor. The RGB color sensor data is sent to Vision Processor D4 via the color Image Signal Processor (ISP) on the Host Processor motherboard or D4 Board.

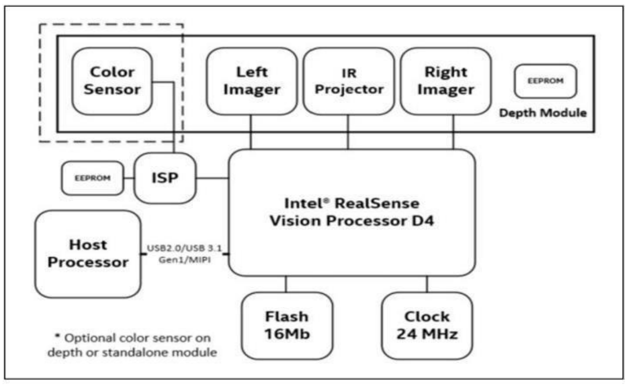


Figure 5: Vision Processor D4 Camera System Block Diagram

| Component | Description |
|---|---|
| Host Processor | Host Processor that receives Depth and other data streams from Vision Processor D4 |
| Vision Processor D4 | Depth Imaging Processor with USB 2.0/USB 3.1 Gen 1 or MIPI interface connection to Host Processor |
| Clock | 24 MHz clock source for Vision Processor D4 |
| Serial Flash Memory | SPI 16 Mb Serial Flash memory for firmware storage |
| Stereo Depth Module | Camera module with left and right imager, color sensor†, IR projector† enclosed in a stiffener |
| Power Delivery | Circuitry on motherboard/Vision Processor D4 Board to deliver and manage power to Vision Processor D4 and Stereo Depth Module. |
| Stereo Depth Connector and Interposer | 50-pin connector on motherboard/Vision Processor D4 Board and Stereo Depth module with interposer for connection |

Table 1: Vision Processor D4 Camera System Components

**Host Processor**
The host processor interface to Vision Processor D4 is either USB 2.0/USB 3.1 Gen 1 or MIPI. To ensure the best quality of service, the Vision Processor D4 must be connected to a dedicated USB 3.1 Gen 1 root port within the host processor system.

**Intel® RealSense™ Vision Processor D4**
The primary function of Vision Processor D4 is to perform depth stereo vision processing. The Vision Processor D4 on the Host Processor motherboard or on the Vision Processor D4 Board communicates to the host processor through USB2.0/USB 3.1 Gen 1 or MIPI and receives sensor data from the stereo depth module. The Vision Processor D4 supports MIPI CSI-2 channels for the connection to the image sensors.

**Vision Processor D4 Features**
• 28 nm CMOS process technology.
• 5 MIPI camera ports with each MIPI lane capable of handling data transfers of up to 750 Mbps.
• USB 2.0/USB 3.1 Gen 1 or MIPI interface to host system.

• Image rectification for camera optics and alignment compensation.
• IR Projector (laser) controls.
• Serial Peripheral Interface for fast data transfer with external SPI flash.
• Integrated I2C ports.
• General purpose Input/Output pins.
• Active power gating.

**Clock**
Vision Processor D4 requires a single 24 MHz clock oscillator. All clocks required by the stereo depth module are generated by Vision Processor D4.

**Serial (SPI) Flash Memory**
Vision Processor D4 requires 16 Mb Serial Flash Memory for its firmware storage. The recommended part number is IS25WP016 (www.issi.com) or equivalent.

**Stereo Depth Module**
The stereo depth module components are described in Table 2. The stereo depth printed circuit board and components are encapsulated in a common metal stiffener.

| Component | Description |
|---|---|
| Left and Right Imagers | 2 image sensors |
| Infrared (IR) Projector | Class 1 laser compliant (optional) |
| Color Sensor | RGB image sensor (optional) |
| Depth Module Connector | 50-pin connector plug |
| Stiffener | Reinforcement housing to keep imagers aligned |
| Label | Manufacture and product identifier information |
| Other Components | Laser driver, EEPROM, voltage regulators, etc. |

Table 2: Stereo depth module components

# 4. Software Requirements:

## 4.1 NVIDIA Driver (NVIDIA JetPack)

NVIDIA JetPack SDK is the most comprehensive solution for building AI applications. JetPack SDK provides a full development environment for hardware-accelerated AI-at-the- edge development. JetPack SDK includes Jetson Linux Driver Package with bootloader, Linux kernel, Ubuntu desktop environment, and a complete set of libraries for acceleration of GPU computing, multimedia, graphics, and computer vision. It also includes samples, documentation, and developer tools for both host computer and developer kit, and supports higher level SDKs such as DeepStream for streaming video analytics, Isaac for robotics and

Riva for conversational AI.

### 4.1.1 Key features of JetPack:

- **TensorRT**

TensorRT is a high performance deep learning inference runtime for image classification, segmentation, and object detection neural networks. TensorRT is built on CUDA, NVIDIA's parallel programming model, and enables you to optimize inference for all deep learning frameworks. It includes a deep learning inference optimizer and runtime that delivers low latency and high throughput for deep learning inference applications. JetPack 6.2 includes TensorRT 10.3.0.

- **DLA**

NVIDIA DLA hardware is a fixed-function accelerator engine targeted for deep learning operations. It's designed to do full hardware acceleration of convolutional neural networks, supporting various layers such as convolution, deconvolution, fully connected, activation, pooling, batch normalization, and others. The DLA software consists of the DLA compiler and the DLA runtime stack. The offline compiler translates the neural network graph into a DLA loadable binary and can be invoked using NVIDIA TensorRT™. The runtime stack consists of the DLA firmware, kernel mode driver, and user mode driver. JetPack 6.2 includes DLA 3.14.

- **cuDNN**

CUDA Deep Neural Network library provides high-performance primitives for deep learning frameworks. It provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers. JetPack 6.2 includes cuDNN 9.3.0.

- **CUDA**

CUDA Toolkit provides a comprehensive development environment for C and C++ developers building GPU-accelerated applications. The toolkit includes a compiler for NVIDIA GPUs, math libraries, and tools for debugging and optimizing the performance of your applications. JetPack 6.2 includes CUDA 12.6.

- **OpenCV**

OpenCV is the leading open source library for computer vision, image processing and machine learning, features GPU acceleration for real-time operation. JetPack 6.2 includes OpenCV 4.8.0

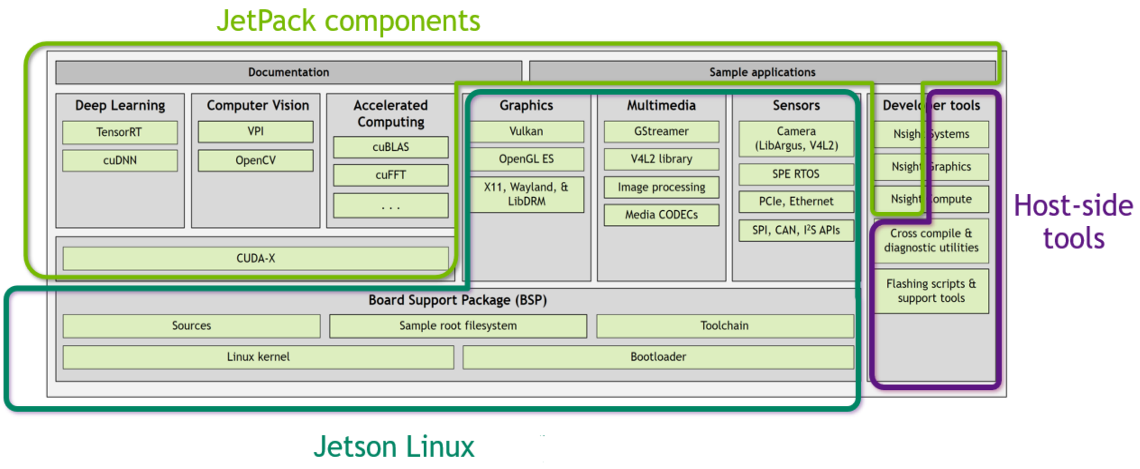


Figure 6: Jetson Software Architecture

### 4.1.2 JetPack 6.2

JetPack 6.2 uses Ubuntu 22.04 LTS as the underlying operating system. Ubuntu 22.04 provides: Long-term support (security updates until 2032), Updated Linux kernel (5.15 LTS), Improved support for modern hardware and security features, Access to recent versions of packages and libraries

**Components:**

- CUDA 12.4
- cuDNN 9.0
- TensorRT 10.x
- DeepStream SDK 7.x
- VPI 3.x
- Kernel 5.15 LTS
- Ubuntu 22.04 LTS (default)
- NVIDIA Container Runtime
- OTA Update Support

**JetPack 5.1.3 OS Image include the following:**

JetPack 5.1.3 includes NVIDIA Jetson Linux 35.3.1 which includes the Linux Kernel 5.10, UEFI based bootloader, Ubuntu 20.04 based root file system, NVIDIA drivers, necessary firmwares, toolchain and more.

### 4.1.3 JetPack 6.2.1

JetPack 6.2.1 is being used in Jetson orin nano.

**Components**

- CUDA 12.6
- TensorRT 10.3.0
- cuDNN 9.3.0
- OpenCV 4.8.0

- DLA 3.14
- VPI 3.2

**JetPack 6.2.1 OS Image include the following:**
Linux 36.4.3 supports all NVIDIA Jetson Orin modules and developer kits and introduces novel features, such as the flexibility to run any upstream Linux Kernel greater than 5.14, and expanded choices of Linux distro options on Jetson.
NVIDIA L4T : This is the core operating system for Jetson devices, derived from Ubuntu.
Bootloader : This software initializes the hardware and loads the operating system.
Linux Kernel : This is the core of the operating system, providing the base functionality for the system.
Firmware : This includes the low-level software that manages hardware.

## 4.2 NVIDIA TensorRT

NVIDIA® TensorRT™ is an ecosystem of APIs high-performance deep learning inference. TensorRT includes an inference runtime and model optimizations that deliver low latency and high throughput for production applications. The TensorRT ecosystem includes TensorRT, TensorRT-LLM, TensorRT Model Optimizer, and TensorRT Cloud. NVIDIA JetPack already includes Tensor RT and related libraries for AI and deep learning. NVIDIA DeepStream SDK integrates with NVIDIA TensorRT to optimize and accelerate deep learning inference tasks. It provides various optimizations to improve the efficiency of deploying trained neural networks. TensorRT optimizes these models by performing operations like layer fusion, precision calibration (e.g., FP32 to FP16 or INT8), kernel auto-tuning, and dynamic tensor memory management. This optimization improves inference speed and reduces memory footprint.

### 4.2.1 Benefits

- Speed Up Inference by 36X
- Optimize Inference Performance
- Accelerate Every Workload
- Deploy, Run, and Scale with Triton

### 4.2.2 Features and Tools of NVIDIA TensorRT

- Large Language Model Inference
- Optimized Inference Engines
- Optimize Neural Networks
- Major Framework Integration

### 4.2.3 Why TensorRT?

- NVIDIA JetPack already includes Tensor RT and related libraries for AI and deep learning.
- NVIDIA DeepStream SDK integrates with NVIDIA TensorRT to optimize and accelerate deep learning inference tasks.
- TensorRT is a high-performance deep learning inference library that provides various optimizations to improve the efficiency of deploying trained neural networks.
- TensorRT optimizes these models by performing operations like layer fusion, precision calibration (e.g., FP32 to FP16 or INT8), kernel auto-tuning, and dynamic tensor memory management. This optimization improves inference speed and reduces memory footprint.

## 4.3 CUDA

CUDA is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU).

**Design goals:**

- Provide a small set of extensions to standard programming languages, like C, that enable a straightforward implementation of parallel algorithms. With CUDA C/C++, programmers can focus on the task of parallelization of the algorithms rather than spending time on their implementation.
- Support heterogeneous computation where applications use both the CPU and GPU. Serial portions of applications are run on the CPU, and parallel portions are offloaded to the GPU. As such, CUDA can be incrementally applied to existing applications. The CPU and GPU are treated as separate devices that have their own memory spaces. This configuration also allows simultaneous computation on the CPU and GPU without contention for memory resources.
- CUDA is essential in enabling high-performance object detection, making it feasible to deploy complex deep learning models in real-world applications.
- CUDA-capable GPUs have hundreds of cores that can collectively run thousands of computing threads. These cores have shared resources including a register file and a shared memory. The on-chip shared memory allows parallel tasks running on these cores to share data without sending it over the system memory bus.

## 4.4 DeepStream

DeepStream is a streaming analytic toolkit to build AI-powered applications. It takes the streaming data as input - from USB/CSI camera, video from file or streams over RTSP, and uses AI and computer vision to generate insights from pixels for better understanding of the environment. DeepStream SDK can be the foundation layer for a number of video analytic solutions like understanding traffic and pedestrians in smart city, health and safety monitoring in hospitals, self-checkout and analytics in retail, detecting component defects at a manufacturing facility and others.

NVIDIA's DeepStream SDK is a complete streaming analytics toolkit based on GStreamer for AI-based multi-sensor processing, video, audio, and image understanding. It's ideal for vision AI developers, software partners, startups, and OEMs building IVA apps and services.

The core SDK consists of several hardware accelerator plugins that use accelerators such as VIC, GPU, DLA, NVDEC and NVENC. By performing all the compute heavy operations in a dedicated accelerator, DeepStream can achieve highest performance for video analytic applications. One of the key capabilities of DeepStream is secure bi-directional communication between edge and cloud. DeepStream ships with several out of the box security protocols such as SASL/Plain authentication using username/password and 2-way TLS authentication.

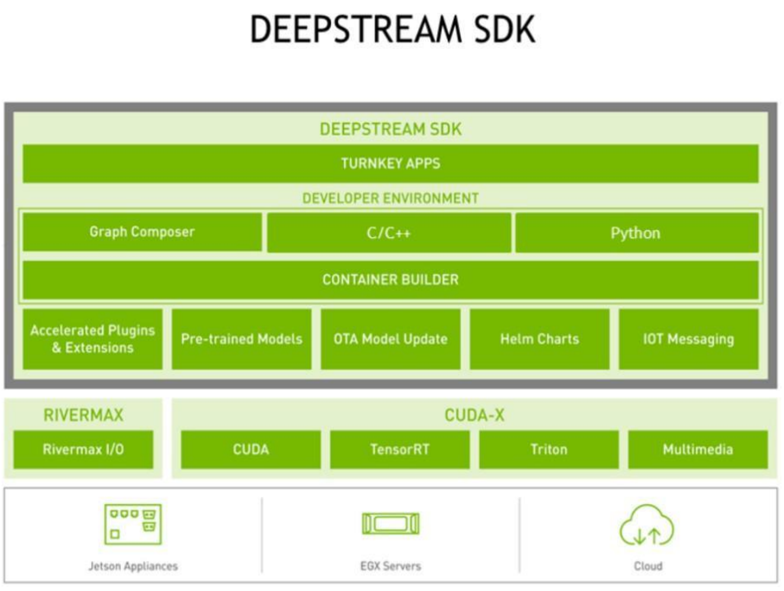


Figure 7: Deepstream SDK

DeepStream supports application development in C/C++ and in Python through the Python bindings. DeepStream builds on top of several NVIDIA libraries from the CUDA-X stack such as CUDA, TensorRT, NVIDIA® Triton™ Inference server and multimedia libraries. TensorRT accelerates the AI inference on the NVIDIA GPU. DeepStream abstracts these libraries in DeepStream plugins, making it easy for developers to build video analytic pipelines without having to learn all the individual libraries.

DeepStream is optimized for NVIDIA GPUs; the application can be deployed on an embedded edge device running Jetson platform or can be deployed on larger edge or datacenter GPUs like T4. DeepStream applications can be deployed in containers using NVIDIA container Runtime.

### 4.4.1 Benefits

- Powerful and Flexible SDK
    - DeepStream SDK is ideal for a wide range of use cases across a broad set of industries.
- Multiple Programming Options
    - Create powerful vision AI applications using C/C++, Python, or Graph Composer's simple and intuitive UI.
- Real-Time Insights
    - Understand rich and multi-modal real-time sensor data at the edge.
- Managed AI Services
    - Deploy AI services in cloud native containers and orchestrate them using Kubernetes.
- Reduced TCO
    - Increase stream density by training, adapting, and optimizing models with TAO toolkit and deploying models with DeepStream.

## 4.5 GStreamer Plugin

One of the most obvious uses of GStreamer is using it to build a media player. GStreamer already includes components for building a media player that can support a very wide variety of formats, including MP3, Ogg/Vorbis, MPEG-1/2, AVI, Quicktime, mod, and more. GStreamer, however, is much more than just another media player. Its main advantages are that the pluggable components can be mixed and matched into arbitrary pipelines so that it's possible to write a full-fledged video or audio editing application.

The framework is based on plugins that will provide the various codec and other functionality. The plugins can be linked and arranged in a pipeline. This pipeline defines the flow of the data. Pipelines can also be edited with a GUI editor and saved as XML so that pipeline libraries can be made with a minimum of effort.

The GStreamer core function is to provide a framework for plugins, data flow and media type handling/negotiation. It also provides an API to write applications using the various plugins.

Specifically, GStreamer provides

- an API for multimedia applications
- a plugin architecture
- a pipeline architecture
- a mechanism for media type handling/negotiation
- a mechanism for synchronization
- over 250 plug-ins providing more than 1000 elements
- a set of tools

GStreamer is packaged into

- gstreamer: the core package
- gst-plugins-base: an essential exemplary set of elements
- gst-plugins-good: a set of good-quality plug-ins under LGPL
- gst-plugins-ugly: a set of good-quality plug-ins that might pose distribution problems
- gst-plugins-bad: a set of plug-ins that need more quality

- gst-libav: a set of plug-ins that wrap libav for decoding and encoding
- a few others packages

# 5. Software set up

## 5.1 Steps for Getting Started

**1. microSD Card**
The Jetson Orin Nano Developer Kit uses a microSD card as both the boot device and the main storage. It’s important to have a card that is fast and large enough for your projects; the minimum recommended size is a 32 GB UHS-I card.

**2. Micro-USB Power Supply**
Power the Jetson Nano developer kit with a good quality power supply that can deliver 5V⎓ 2A at the developer kit’s Micro-USB port. Jetson Orin Nano Developer Kit requires 19V power supply.

**3. For Jetson Orin Nano Developer Kit update the firmware if needed.**
Your Jetson Orin Nano Developer Kit may be ready to run JetPack 6 by having the latest firmware ("Jetson UEFI firmware" on QSPI-NOR flash memory) flashed at the factory. If this isn’t the case, you’ll need to upgrade to the latest firmware.

## 5.2 Write Image to microSD card

Same steps for Jetson Nano Developer Kit SD Card Image or Jetson Orin Nano Developer Kit.

- Download the Jetson Nano Developer Kit SD Card Image or Jetson Orin Nano Developer Kit SD Card image from JetPack SDK Page, and note where it was saved on the computer.
- Format your microSD card using SD Memory Card Formatter from the SD Association.

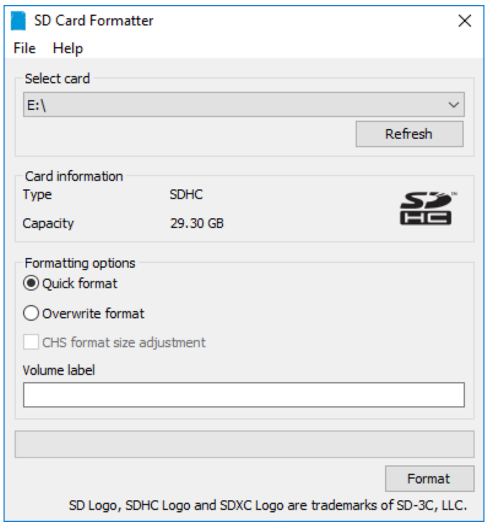

SD Card Formatter
Select card: E:\
Card information
Type: SDHC
Capacity: 29.30 GB
Formatting options
[✓ Quick format]
Overwrite format
CHS format size adjustment
Volume label:
SD Logo, SDHC Logo and SDXC Logo are trademarks of SD-3C, LLC.

Figure 8: SD Card Formatter

1. Download, install, and launch SD Memory Card Formatter for Windows.
2. Select card drive
3. Select “Quick format”
4. Leave “Volume label” blank
5. Click “Format” to start formatting, and “Yes” on the warning dialog

- Use Etcher to write the Jetson Nano Developer Kit SD Card Image to your microSD card.
  1. Download, install, and launch Etcher.

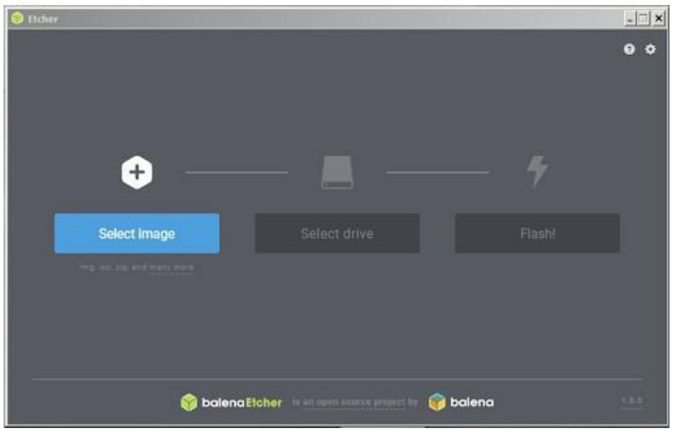
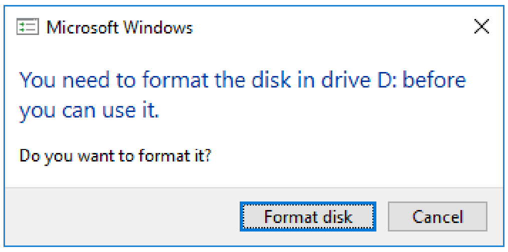


Figure 9: Etcher

  2. Click “Select image” and choose the zipped image file downloaded earlier.
  3. Insert your microSD card if not already inserted.
     Click *Cancel* if Windows prompts you with a dialog like this:



Microsoft Windows

You need to format the disk in drive D: before you can use it.

Do you want to format it?

Format disk | Cancel

Figure 10: Dialogue Box

  4. Click “Select drive” and choose the correct device.

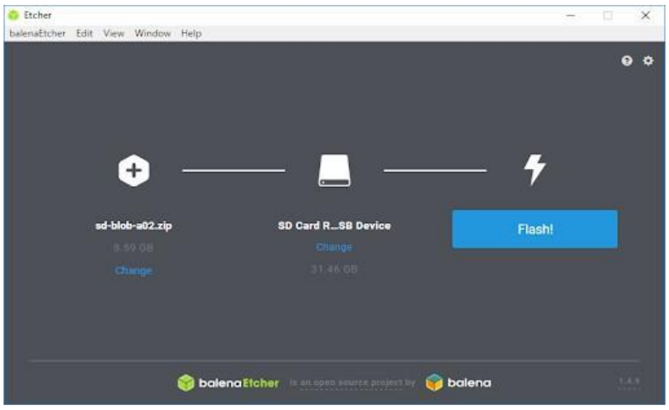
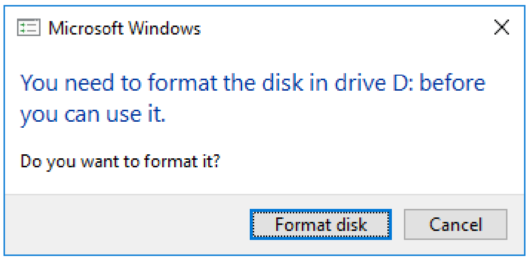


Figure 11: Etcher Flash

5. Click “Flash!” It will take Etcher about 10 minutes to write and validate the image if your microSD card is connected via USB3.
6. After Etcher finishes, Windows may let you know it doesn’t know how to read the SD Card. Just click Cancel and remove the microSD card.


Microsoft Windows

You need to format the disk in drive D: before you can use it.

Do you want to format it?

Figure 12: Dialogue Box

7. After your microSD card is ready, proceed to set up your developer kit.

## 5.3 Setup and First Boot

### 5.3.1 Jetson Orin Nano Developer Kit

There are two ways to interact with the developer kit: 1) with display, keyboard and mouse attached, or 2) in “headless mode” via connection from another computer.
Either way can be conducted for the initial setup.

**1. Initial Setup with microSD-only method Display Attached**

**a) Setup Steps**

1. Insert the microSD card (with system image already written to it) into the slot on the underside of the Jetson Orin Nano module.
2. Power on your computer display and connect it.
3. Connect the USB keyboard and mouse.
4. Connect the provided power supply. The Jetson Orin Nano Developer Kit will power on and boot automatically.

**b) First Boot**

A green LED next to the USB-C connector will light as soon as the developer kit powers on. When you boot the first time, the Jetson Orin Nano Developer Kit will take you through some initial setup, including:

- Review and accept NVIDIA Jetson software EULA
- Select system language, keyboard layout, and time zone
- Connect to Wireless network
- Create username, password, and computer name
- Log in

Note: This setup uses JetPack 5.1.3, which corresponds to L4T 35.6.1.The developer kit was later updated from this version, rather than flashing JetPack 6.2 directly, due to compatibility and image availability at the time of setup.

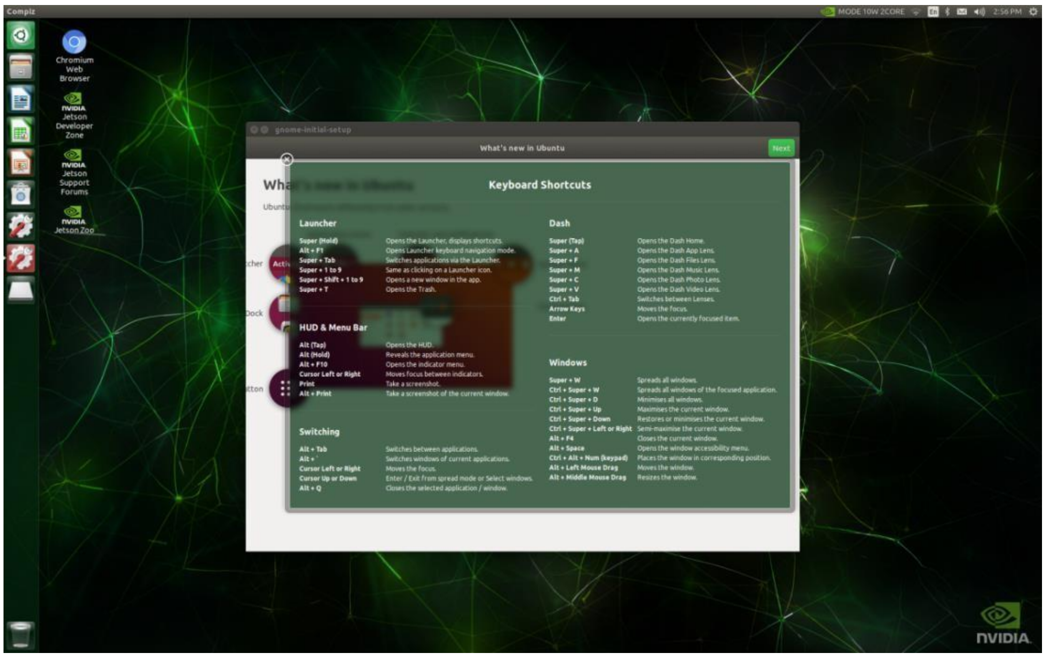
Reasons for updating to JetPack 6.2.1 from existing JetPack 5.1.3 are as follows :
**Lower UEFI Version:** Jetson UEFI Firmware had the version higher than 36.0 hence the above approach of updating the firmware with 5.6.1 is followed to get the JesPack 6.2.1 version working.

**Hardware boot compatibility:** The factory-flashed bootloader/firmware on the developer kit was designed for L4T 35.x (JetPack 5.x). Jumping directly to L4T 36.x (JetPack 6.x) without intermediate updates can cause boot failures or devices not detecting the image.

**Workflow constraints:** Using an SD card image allowed a quick out-of-box setup without a host PC. Updating from 5.1.3 to the desired version afterward was more reliable and faster in the given environment.

### c) After Logging In

You will see this screen.



Keyboard Shortcuts

Launcher

| | |
|---|---|
| Super (Hold) | Opens the Launcher, displays shortcuts. |
| Alt + F1 | Opens Launcher keyboard navigation mode. |
| Super + Tab | Switches applications via the Launcher. |
| Super + 1 to 9 | Same as clicking on a Launcher icon. |
| Super + Shift + 1 to 9 | Opens a new window in the app. |
| Super + T | Opens the Trash. |

HUD & Menu Bar

| | |
|---|---|
| Alt (Tap) | Opens the HUD. |
| Alt (Hold) | Reveals the application menu. |
| Alt + F10 | Opens the indicator menu. |
| Cursor Left or Right | Moves focus between indicators. |
| Print | Take a screenshot. |
| Alt + Print | Take a screenshot of the current window. |

Switching

| | |
|---|---|
| Alt + Tab | Switches between applications. |
| Alt + ` | Switches windows of current applications. |
| Cursor Left or Right | Moves the focus. |
| Cursor Up or Down | Enter / Exit from spread mode or Select windows. |
| Alt + Q | Closes the selected application / window. |

Dash

| | |
|---|---|
| Super (Tap) | Opens the Dash Home. |
| Super + A | Opens the Dash App Lens. |
| Super + F | Opens the Dash Files Lens. |
| Super + M | Opens the Dash Music Lens. |
| Super + C | Opens the Dash Photo Lens. |
| Super + V | Opens the Dash Video Lens. |
| Ctrl + Tab | Switches between Lenses. |
| Arrow Keys | Moves the focus. |
| Enter | Opens the currently focused item. |

Windows

| | |
|---|---|
| Super + W | Spreads all windows. |
| Ctrl + Super + W | Spreads all windows of the focused application. |
| Ctrl + Super + D | Minimises all windows. |
| Ctrl + Super + Up | Maximises the current window. |
| Ctrl + Super + Down | Restores or minimises the current window. |
| Ctrl + Super + Left or Right | Semi-maximise the current window. |
| Alt + F4 | Closes the current window. |
| Alt + Space | Opens the window accessibility menu. |
| Ctrl + Alt + Num (keypad) | Places the window in corresponding position. |
| Alt + Left Mouse Drag | Moves the window. |
| Alt + Middle Mouse Drag | Resizes the window. |

Figure 13: Initial Screen of Jetson Orin Nano

## 2. Initial Setup Headless Mode

To complete setup when no display is attached to the developer kit, you'll need to connect the developer kit to another computer and then communicate with it via a terminal application (e.g., PuTTY) to handle the USB serial communication on that other computer.

Note: Headless initial configuration requires the developer kit to be powered by a DC power supply with barrel jack connector, since the Micro-USB port is required to access the initial configuration prompts.

### a) Setup Steps

1. Unfold the paper stand and place it inside the developer kit box.

2. Insert the microSD card (with system image already written to it) into the slot on the underside of the Jetson Nano module.
3. Set the developer kit on top of the paper stand.
4. Check the Jetson Nano Developer Kit User Guide for location of J48 Power Select Header and J25 Power Jack.
5. Jumper the J48 Power Select Header pins.
6. Connect your other computer to the developer kit's Micro-USB port.
7. Connect a DC power supply to the J25 Power Jack. The developer kit will power on automatically.
8. Allow 1 minute for the developer kit to boot.
9. On your other computer, use the serial terminal application to connect via host serial port to the developer kit.

**b) Locate the correct COM port**

- Assuming you have already connected your Windows PC to the developer kit's Micro-USB port, right click the Windows Start icon and select "Device Manager."
- Open the "Ports (COM & LPT)" to find the COM port number for "USB Serial Device."
- Double click each USB Serial Device entry so you can check its properties. Go to the "Details" tab, and select "Hardware Ids". If you see VID 0955 and PID 7020, that USB Serial Device for your Jetson developer kit. Note the COM port name for later use.

**c) Open the COM port on PuTTY**

- Open the PuTTY application. When "Session" is selected in the left "Category" pane, input the COM port name for "Serial line" and "115200" for "Speed".
- Click "Open" to connect to the console.
- Once connected to the developer kit, hit SPACE if the initial setup screen does not appear automatically.

**d) First Boot**

A green LED next to the Micro-USB connector will light as soon as the developer kit powers on. When you boot the first time, the developer kit will take you through some initial setup, including:

- Review and accept NVIDIA Jetson software EULA
- Select system language, keyboard layout, and time zone
- Create username, password, and computer name
- Select APP partition size—it is recommended to use the max size suggested

**e) After Logging In**

You will see a standard Linux command line prompt in your serial terminal application.

## 5.4 DeepStream Installation

### 5.4.1 Install Dependencies (prerequisite packages)

Enter the following commands to install the prerequisite packages:

```
$ sudo apt install \
libssl3 \
libssl-dev \
libgstreamer1.0-0 \
gstreamer1.0-tools \
gstreamer1.0-plugins-good \
gstreamer1.0-plugins-bad \
gstreamer1.0-plugins-ugly \
gstreamer1.0-libav \
libgstreamer-plugins-base1.0-dev \
libgstrtspserver-1.0-0 \
libjansson4 \
libyaml-cpp-dev
```

### 5.4.2 Install the DeepStream SDK

Using the DeepStream Debian package:
https://catalog.ngc.nvidia.com/orgs/nvidia/resources/deepstream
Download the DeepStream 7.1 Jetson Debian package deepstream-7.1_7.1.0-1_arm64.deb to the Jetson device. Enter the following command:

```
$sudo apt install ./deepstream-7.1_7.1.0-1_arm64.deb
```

## 5.5 CUDA toolkit installation steps

The setup of CUDA development tools on a system running the appropriate version of Windows consists of a few simple steps:

- Verify the system has a CUDA-capable GPU.
- Download the NVIDIA CUDA Toolkit.
- Install the NVIDIA CUDA Toolkit.

### 5.5.1 Verify You Have a CUDA-Capable GPU

You can verify that you have a CUDA-capable GPU through the Display Adapters section in the Windows Device Manager. Here you will find the vendor name and model of your graphics card(s). If you have an NVIDIA card that is listed in http://developer.nvidia.com/cuda-gpus , that GPU is CUDA-capable. The Release Notes for the CUDA Toolkit also contain a list of supported products. The Windows Device Manager can be opened via the following steps:

1. Open a run window from the Start Menu
2. Run: control /name Microsoft.DeviceManager

### 5.5.2 Download the NVIDIA CUDA Toolkit

The NVIDIA CUDA Toolkit is available at http://developer.nvidia.com/cuda-downloads. Choose the platform you are using and one of the following installer formats:

1. Network Installer: A minimal installer which later downloads packages required for installation. Only the packages selected during the selection phase of the installer are downloaded. This installer is useful for users who want to minimize download time.

2. Full Installer: An installer which contains all the components of the CUDA Toolkit and does not require any further download. This installer is useful for systems which lack network access and for enterprise deployment. The CUDA Toolkit installs the CUDA driver and tools needed to create, build and run a CUDA application as well as libraries, header files, CUDA samples source code, and other resources.

### 5.5.3 Install the CUDA Software

Before installing the toolkit, you should read the Release Notes, as they provide details on installation and software functionality.

Graphical Installation Install the CUDA Software by executing the CUDA installer and following the on-screen prompts. Silent Installation The installer can be executed in silent mode by executing the package with the -s flag. Additional parameters can be passed which will install specific subpackages instead of all packages. See the table below for a list of all the subpackage names.

# 6. Project Implementation

## 6.1 Getting Started

- The project contains anomaly detection applications and auxiliary plug-ins to show the capability of Deepstream SDK.
- Clone the app in /opt/nvidia/deepstream/deepstream/sources/apps/sample_apps/

## 6.2 Compilation Steps for dsdirection plugin

```
$ cd plugins/gst-dsdirection/
$ sudo make && sudo make install
```

### 6.2.1 Usage

Anomaly detection: Detect movements that deviate from expected paths (e.g., people walking against the natural traffic direction). Direction-based analytics: Identify predominant flow patterns or unusual motion changes.

Metadata-driven downstream actions: The direction output can be consumed by elements like nvdsanalytics, nvdsosd, or custom sinks to trigger alerts, overlay arrows, or logging.

## 6.2.2 Test direction calculation on video input, on Jetson, run following commands

### one video input

```
$ cd /opt/nvidia/deepstream/deepstream-7.1/
$ gst-launch-1.0 v4l2src device=/dev/video2 ! \
        'video/x-raw,format=UYVY,width=1280,height=720,framerate=30/1' ! \
        nvvidconv ! 'video/x-raw(memory:NVMM),format=NV12,width=1280,height=720' ! \
        queue ! m.sink_0 \
        nvstreammux name=m batch-size=1 width=1280 height=720 live-source=1 ! \
        nvinfer config-file-path=samples/configs/deepstream-app/config_infer_primary.txt ! \
        nvof ! tee name=t ! queue ! nvofvisual ! \
        nvmultistreamtiler width=1280 height=720 rows=1 columns=1 ! nv3dsink sync=0 \
        t. ! queue ! dsdirection ! \
        nvmultistreamtiler width=1280 height=720 rows=1 columns=1 ! nvvideoconvert ! nvdsosd !
nv3dsink sync=0
```

### Dual Camera Video input

```
cd /opt/nvidia/deepstream/deepstream-7.1/
        $ gst-launch-1.0 \v4l2src device=/dev/video2 ! \
         'video/x-raw,format=UYVY,width=1280,height=720,framerate=30/1' ! \
        nvvidconv ! 'video/x-raw(memory:NVMM),format=NV12,width=1280,height=720' ! \
        queue ! m.sink_0 \v4l2src device=/dev/video8 ! \
        'video/x-raw,format=UYVY,width=1280,height=720,framerate=30/1' ! \
        nvvidconv ! 'video/x-raw(memory:NVMM),format=NV12,width=1280,height=720' ! \
        queue ! m.sink_1 \
        nvstreammux name=m batch-size=2 width=1280 height=720 live-source=1 ! \
        nvinfer config-file-path=samples/configs/deepstream-app/config_infer_primary.txt ! \
        nvof ! tee name=t ! queue ! nvofvisual ! \
        nvmultistreamtiler width=2560 height=720 rows=1 columns=2 ! nv3dsink sync=0 \
        t. ! queue ! dsdirection ! \
        nvmultistreamtiler width=2560 height=720 rows=1 columns=2 ! nvvideoconvert ! nvdsosd !
nv3dsink sync=0
```

## 6.3 Actual Application

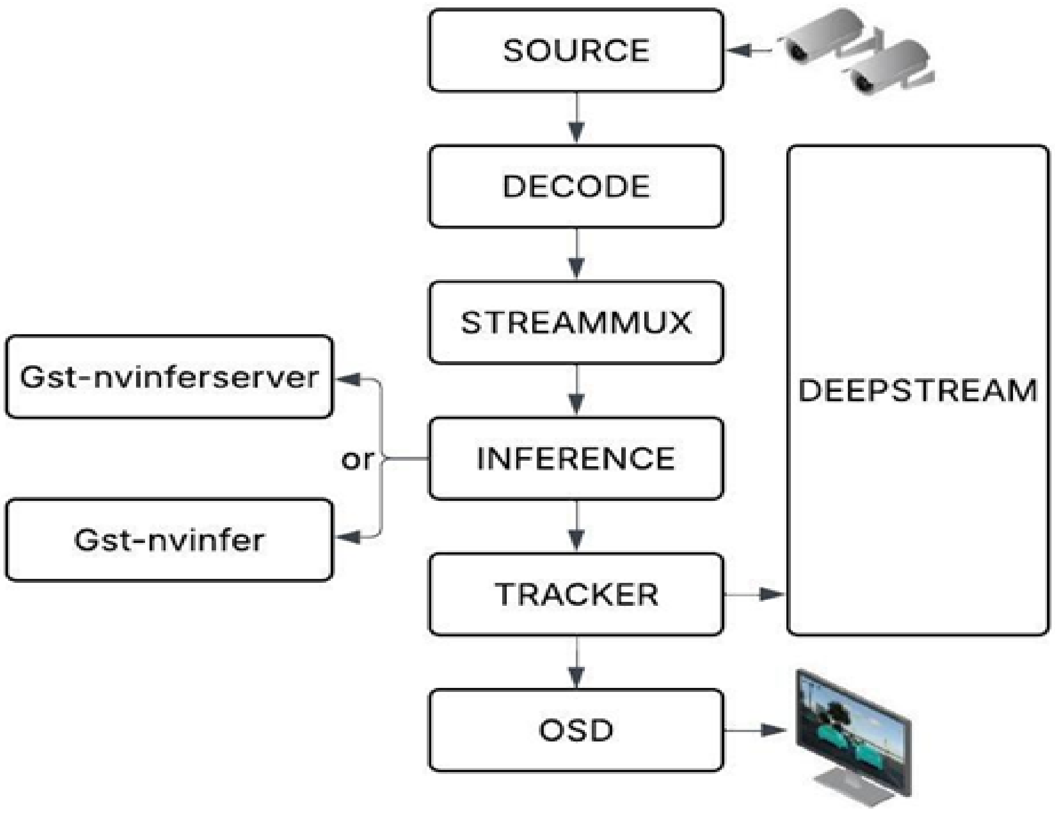


Figure 14: Initial Screen of Jetson Orin Nano

### 6.3.1 Compile the application

```
$ cd sources/apps/sample_apps/deepstream_reference_apps/anomaly/
$ cd apps/deepstream-anomaly-detection/
$ Set CUDA_VER in the MakeFile as per platform.
 For both x86 & Jetson, CUDA_VER=12.6
$ sudo make
```

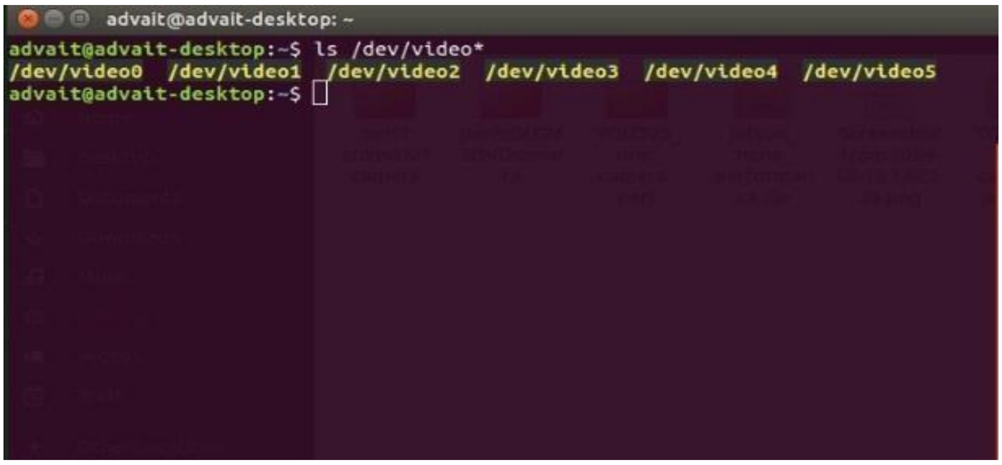
### 6.3.2 Running the commands to start the application

```
$ cd/opt/nvidia/deepstream/deepstream-7.1/sources/apps/sample_apps/
Anomaly_Detection_Camera/deepstream_reference_apps/anomaly/apps/
deepstream-anomaly-detection
$ sudo make
Please modify the camera inputs for the ./deepstream-anomaly-detection-app and gst-streamer,specify
the correct sources
$ sh ./run_deepstream_anomaly_detection_app.sh
```

### 6.3.3 Steps to start and run the application

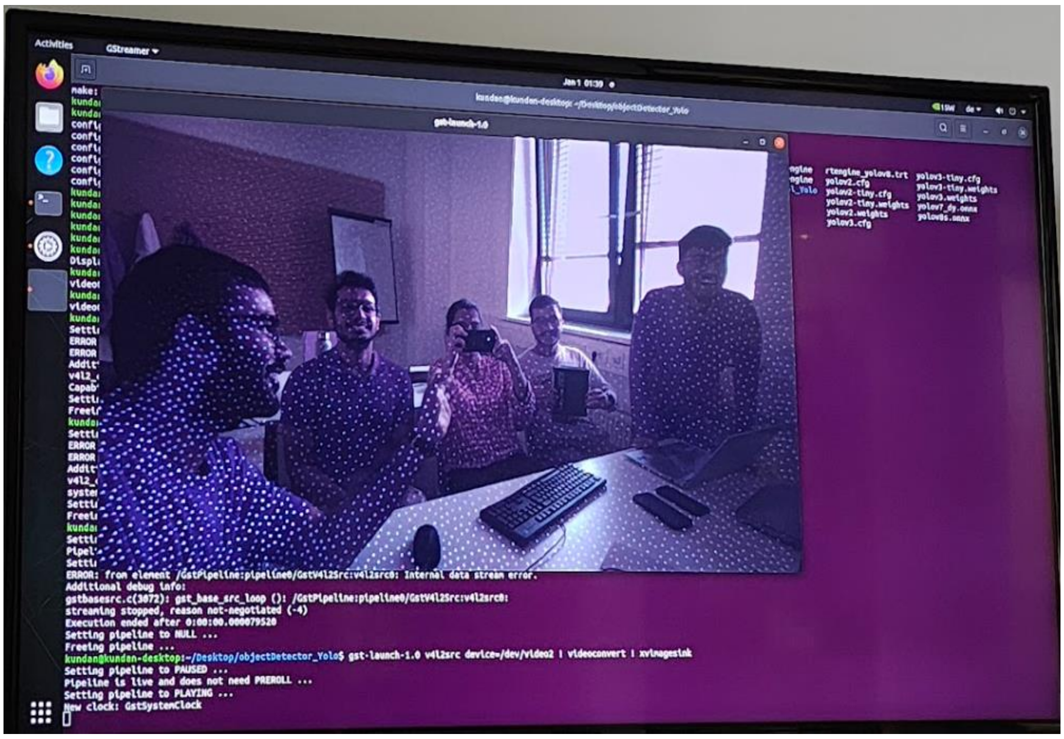
- Connect camera to the Jetson and run command: ls /dev/video*
- Check to which video node your camera is connected by using the command: gst-launch-1.0 v4l2src device=/dev/video2 ! videoconvert ! xvimagesink

(Try this command with every node and check to which node the camera is connected, the output will be as mentioned below)



```
advait@advait-desktop: ~
advait@advait-desktop:~$ ls /dev/video*
/dev/video0  /dev/video1  /dev/video2  /dev/video3  /dev/video4  /dev/video5
advait@advait-desktop:~$ 
```

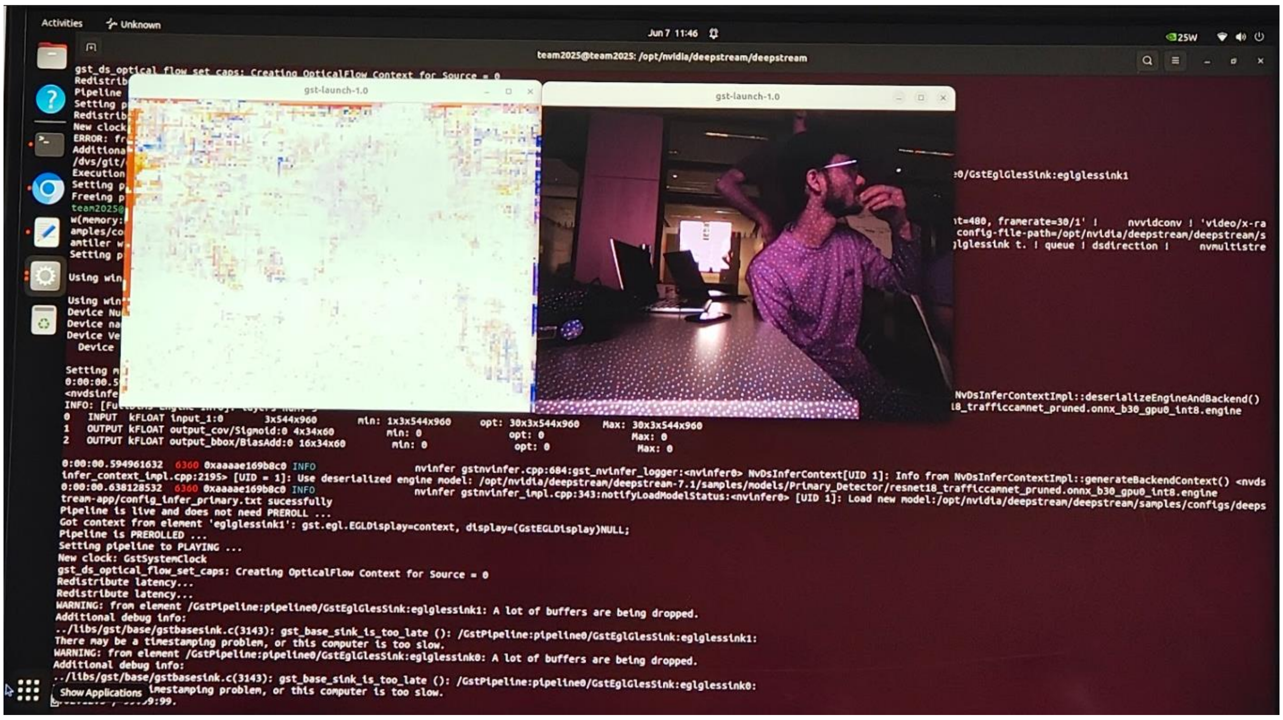
Figure 15: Output to detect the camera devices



```
ERROR: from element /GstPipeline:pipeline0/GstV4l2Src:v4l2src0: Internal data stream error.
Additional debug info:
gstbasesrc.c(3072): gst_base_src_loop (): /GstPipeline:pipeline0/GstV4l2Src:v4l2src0:
streaming stopped, reason not-negotiated (-4)
Execution ended after 0:00:00.000079520
Setting pipeline to NULL ...
Freeing pipeline ...
kundan@kundan-desktop:~/Desktop/objectDetector_Yolo$ gst-launch-1.0 v4l2src device=/dev/video2 ! videoconvert ! xvimagesink
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
```

Figure 16: Output of gst-launch

- Complete the software setup.
- Compile the steps for the dsdirection plugin.



```
gst_ds_optical_flow_set_caps: Creating OpticalFlow Context for Source = 0
Redistrib
Pipeline
Setting p
Redistrib
New clock
ERROR: fr                                                          e0/GstEglGlesSink:eglglessink1
Additiona
/dvs/git/
Execution
Setting p
Freeing p
team2025@                                                          nt=480, framerate=30/1' !     nvvidconv ! 'video/x-ra
w(memory:                                                           config-file-path=/opt/nvidia/deepstream/deepstream/s
amples/co                                                          glglessink t. ! queue ! dsdirection !     nvmultistre
amtiler w
Setting p

Using win

Using win
Device Nu
Device na
Device Ve
  Device

Setting m
0:00:00.5                                                          NvDsInferContextImpl::deserializeEngineAndBackend()
<nvdsinfe                                                          18_trafficcamnet_pruned.onnx_b30_gpu0_int8.engine
INFO: [Fu
0   INPUT  kFLOAT input_1:0         3x544x960       min: 1x3x544x960     opt: 30x3x544x960    Max: 30x3x544x960
1   OUTPUT kFLOAT output_cov/Sigmoid:0 4x34x60          min: 0               opt: 0               Max: 0
2   OUTPUT kFLOAT output_bbox/BiasAdd:0 16x34x60         min: 0               opt: 0               Max: 0

0:00:00.594961632  6360 0xaaaae169b8c0 INFO                 nvinfer gstnvinfer.cpp:684:gst_nvinfer_logger:<nvinfer0> NvDsInferContext[UID 1]: Info from NvDsInferContextImpl::generateBackendContext() <nvds
infer_context_impl.cpp:2195> [UID = 1]: Use deserialized engine model: /opt/nvidia/deepstream/deepstream-7.1/samples/models/Primary_Detector/resnet18_trafficcamnet_pruned.onnx_b30_gpu0_int8.engine
0:00:00.638128532  6360 0xaaaae169b8c0 INFO                 nvinfer gstnvinfer_impl.cpp:343:notifyLoadModelStatus:<nvinfer0> [UID 1]: Load new model:/opt/nvidia/deepstream/deepstream/samples/configs/deeps
tream-app/config_infer_primary.txt sucessfully
Pipeline is live and does not need PREROLL ...
Got context from element 'eglglessink1': gst.egl.EGLDisplay=context, display=(GstEGLDisplay)NULL;
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
gst_ds_optical_flow_set_caps: Creating OpticalFlow Context for Source = 0
Redistribute latency...
Redistribute latency...
WARNING: from element /GstPipeline:pipeline0/GstEglGlesSink:eglglessink1: A lot of buffers are being dropped.
Additional debug info:
../libs/gst/base/gstbasesink.c(3143): gst_base_sink_is_too_late (): /GstPipeline:pipeline0/GstEglGlesSink:eglglessink1:
There may be a timestamping problem, or this computer is too slow.
WARNING: from element /GstPipeline:pipeline0/GstEglGlesSink:eglglessink0: A lot of buffers are being dropped.
Additional debug info:
../libs/gst/base/gstbasesink.c(3143): gst_base_sink_is_too_late (): /GstPipeline:pipeline0/GstEglGlesSink:eglglessink0:
          imestamping problem, or this computer is too slow.
        9:99.
```

Figure 17: Compilation of gst-launch

- Compile the application by setting the CUDA_VER.
- Run the specified commands to start the application by specifying the input source for the gst-streamer and application.

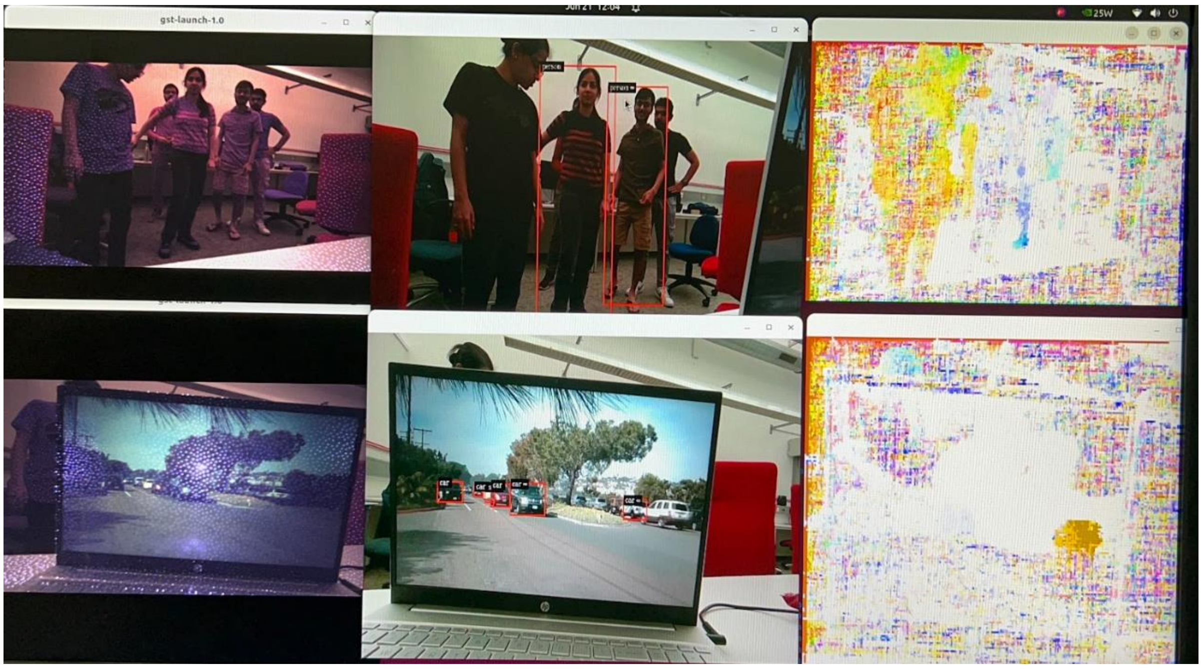


Figure 18: Application output with raw input, object detection, anomaly detection

# 7. Challenges Faced

The initial goal was to gain an understanding of the anomaly detection system by first using an MP4 video file as the input source. This approach provided a controlled environment for experimentation before moving on to using a live Intel RealSense D455 camera input and later displaying raw and processed streams together. During this progression, several challenges were encountered, including software compatibility issues, configuration adjustments, pipeline modifications, and visualization enhancements.

## 7.1 Setup and Compatibility Issues

The initial attempt involved running the Anomaly Detection Reference App using an old SD card previously used for a YOLO Object Detection setup. This configuration failed to execute the project due to incompatibility between installed software versions and the anomaly detection application's requirements. The existing SD card had JetPack 6.0.5 and DeepStream SDK 6.2, whereas the anomaly detection system required DeepStream SDK 7.1 and the latest compatible JetPack version. This necessitated reimaging the SD card with the correct software stack to ensure compatibility and successful execution.

## 7.2 Configuration File Adjustments

The model inference configuration files (dsanomaly_pgie_config.txt or dsanomaly_pgie_nvinferserver_config.txt) required updates to correctly reference the location of the generated TensorRT engine file. Since a ResNet-18 model was used, relative paths in the configuration had to be modified to align with the generated engine file's directory.

## 7.3 Missing TensorRT Engine File

During the GStreamer pipeline execution, the application failed due to the absence of the required TensorRT engine file for the nvinfer plugin. As the GitHub repository did not provide a pre-generated engine file, it had to be manually generated from the provided .onnx model. This process involved extracting model metadata, identifying the correct input tensor (input_1:0), and setting the batch size to 30, as required by the application. The engine generation process took approximately 30 minutes to complete.

## 7.4 Build Errors Due to CUDA Mismatch

While building project components (e.g., the gst-dsdirection plugin), the make command failed due to a mismatch between the default CUDA version in the environment and the one required for the project. This issue was resolved by explicitly specifying CUDA version 12.6 in the Makefile, allowing the build to complete successfully.

## 7.5 Transition from MP4 File Input to Live Camera Input

Moving from an MP4 file source to a live Intel RealSense D455 camera feed introduced new integration challenges. Unlike uridecodebin used for file input, which automatically handles demuxing and decoding, live streaming via v4l2src required explicit inclusion and manual linking of additional elements such as capsfilter, videoconvert, and nvvideoconvert to ensure correct format compatibility. Custom settings for resolution, framerate, and format conversion had to be applied, along with modifications to the GStreamer command to set live-source = 1 for real-time processing.

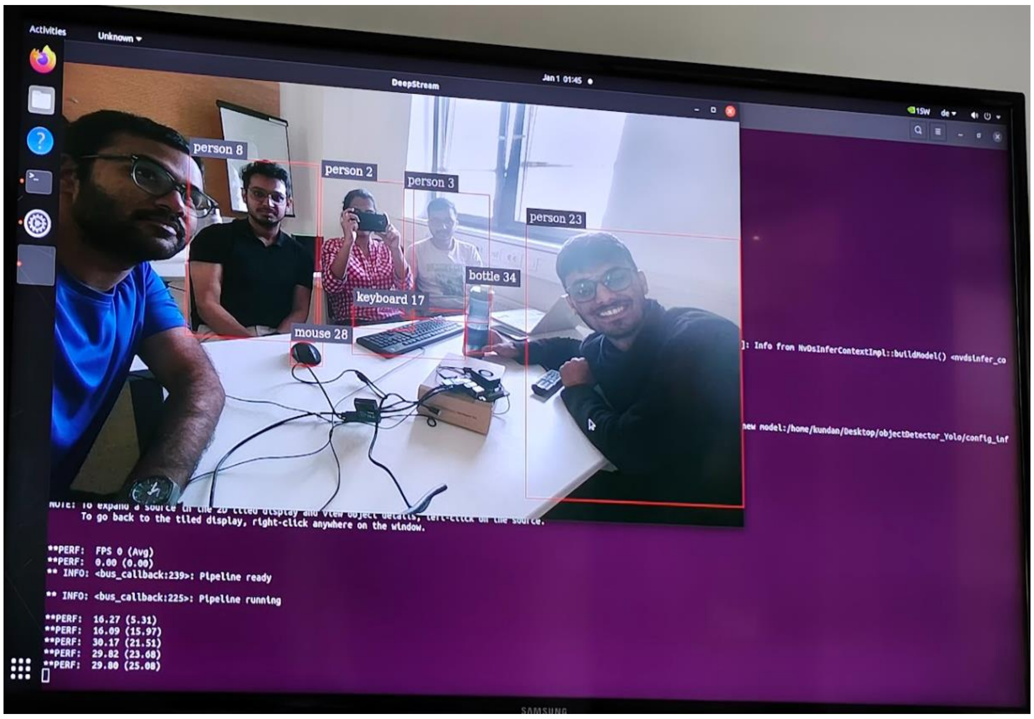
## 7.6 Parallel Display of Raw and Processed Streams

In the subsequent stage, the goal was to display both the processed AI analytics output and the raw camera feed side-by-side for better debugging and validation. Initially, only the processed anomaly detection output was shown. To achieve the dual-display setup, a custom shell script was developed to automate launching multiple processes—running anomaly detection on two RealSense cameras and starting live previews from virtual video devices. This automation reduced manual setup time, minimized human error, and improved productivity. The enhanced view significantly aided in comparing AI interpretation with the actual camera feed, improving both system validation and presentation quality.

Overall, these challenges, though time-consuming, provided a deeper understanding of the anomaly detection pipeline, GStreamer-based streaming, and hardware-software compatibility requirements, laying a strong foundation for subsequent project phases involving live camera input and parallel stream displays.

# 8. Results

## 8.1 Object Detection using DeepStream + 2 RealSense Camera + YoloV7



```
]: Info from NvDsInferContextImpl::buildModel() <nvdsinfer_co

new model:/home/kundan/Desktop/objectDetector_Yolo/config_inf

NOTE: To expand a source in the 2D tiled display and view object details, left-click on the source.
      To go back to the tiled display, right-click anywhere on the window.

**PERF:  FPS 0 (Avg)
**PERF:  0.00 (0.00)
** INFO: <bus_callback:239>: Pipeline ready

** INFO: <bus_callback:225>: Pipeline running

**PERF:  16.27 (5.31)
**PERF:  16.09 (15.97)
**PERF:  30.17 (21.51)
**PERF:  29.82 (23.68)
**PERF:  29.80 (25.08)
```

Figure 19: Object detection using DeepStream

GPU Usage: In the DeepStream scenario, the GPU usage is significantly higher (98.2% in the monitoring window). This indicates that DeepStream is leveraging the GPU for accelerating object detection, which is optimal for such tasks on a Jetson device.

CPU Load: The CPU usage is lower compared to the OpenCV scenario. This reflects the offloading of intensive tasks to the GPU, leaving the CPU available for other operations or reducing overall system load.

Inference Speed: DeepStream is known for high-performance, real-time inference, particularly in video processing and object detection tasks. The higher FPS and real-time performance are evident, given the efficient utilization of the GPU.

**Observations:**

Performance: DeepStream delivers much higher performance due to its efficient GPU utilization, making it suitable for real-time applications. Moreover, object detection with Deepstream is more accurate as compared to opencv.

Resource Utilization: OpenCV with YOLOv7 primarily uses the CPU, making it less efficient on platforms like Jetson, which are designed to maximize GPU usage. DeepStream, on the other hand, is GPU-accelerated and optimized for the Jetson platform.

Use Case Suitability: For applications requiring high throughput and low latency, DeepStream is the preferred choice on Jetson devices. OpenCV might be used for simpler tasks or when GPU acceleration is not a priority.

**Conclusion:**
Object detection on Jetson devices using DeepStream has notable performance, scalability, and ease of use. Below is a highlighting of these:

**1.** Performance Optimization

DeepStream:

- o GPU Acceleration: DeepStream is highly optimized for NVIDIA GPUs, leveraging TensorRT and CUDA to maximize performance. This results in significantly faster inference times, particularly in real-time applications.
- o Efficient Pipelines: Designed for processing multiple video streams in parallel, DeepStream manages complex pipelines efficiently, making it ideal for high-throughput applications like smart cities or large-scale surveillance systems.

**2.** Scalability

DeepStream:

- o Multi-Stream Capability: DeepStream excels in handling multiple video streams simultaneously, making it suitable for applications requiring large-scale deployment, such as monitoring several cameras at once.
- o Edge to Cloud: DeepStream is designed for deployment across various platforms, from edge devices like Jetson Nano/Orin to large cloud-based systems, allowing for scalable AI solutions.

**3.** Ease of Use and Flexibility

- DeepStream:
    - o Purpose-Built SDK: DeepStream is specifically designed for AI-powered video analytics, offering built-in support for object detection, tracking, and other video processing tasks with minimal configuration.
    - o Pre-Integrated Models: DeepStream supports a variety of pre-integrated AI models (like YOLO, SSD, Faster R-CNN), making it easier to implement complex object detection pipelines out of the box.

**4.** Use Cases

- DeepStream:
    - o Real-Time Analytics: Ideal for applications requiring real-time object detection and analytics, such as autonomous vehicles, smart surveillance, and retail analytics.
    - o Large-Scale Deployments: Suitable for environments with multiple video sources and the need for high scalability and performance, such as city-wide

surveillance or industrial monitoring.

**5.** Integration and Ecosystem

- DeepStream:
    - Part of NVIDIA's Ecosystem: DeepStream integrates seamlessly with other NVIDIA tools and libraries, such as TensorRT, CUDA, and JetPack, making it part of a robust ecosystem for AI development on NVIDIA hardware.
    - AI and IoT Integration: DeepStream is designed to integrate with IoT frameworks, cloud services, and AI models, providing a comprehensive solution for edge AI deployments.

## 8.2 Object Detection using Orin Nano + 2 RealSense Camera + YoloV3

Depth sensing was implemented for the first time using the existing DeepStream framework, with the object detection module requiring updates. When an object is positioned farther from the camera, the color gradient in the depth visualization appears lighter.

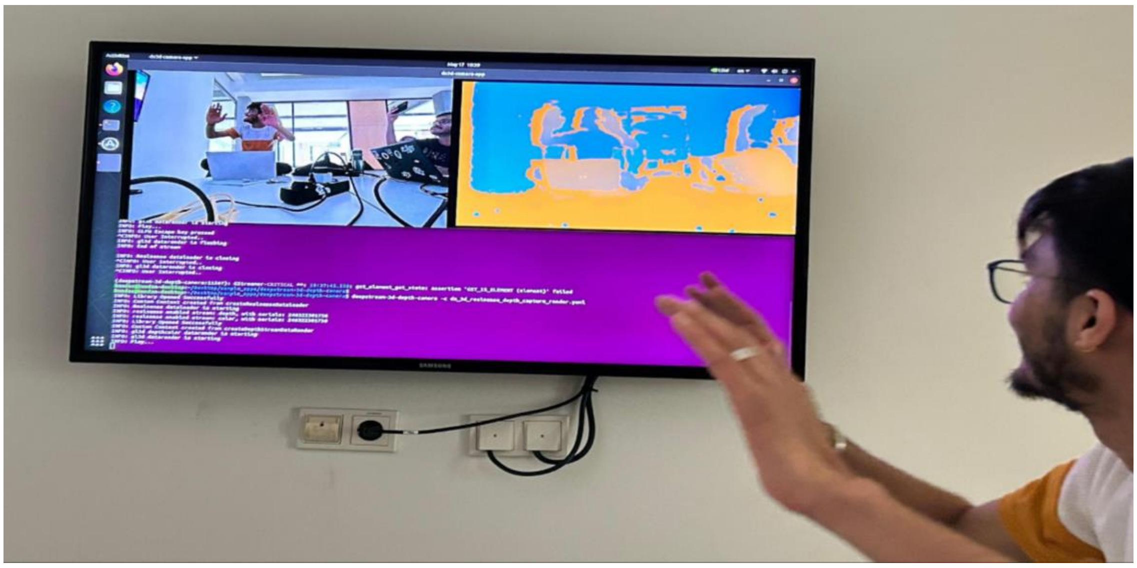


Figure 20: Object detection using Orin Nano

The depth sensing feature was integrated for the first time using the existing DeepStream framework. Updates to the object detection module are required to ensure optimal performance. Observations indicate that objects positioned closer to the camera exhibit a more intense color gradient in the depth visualization.

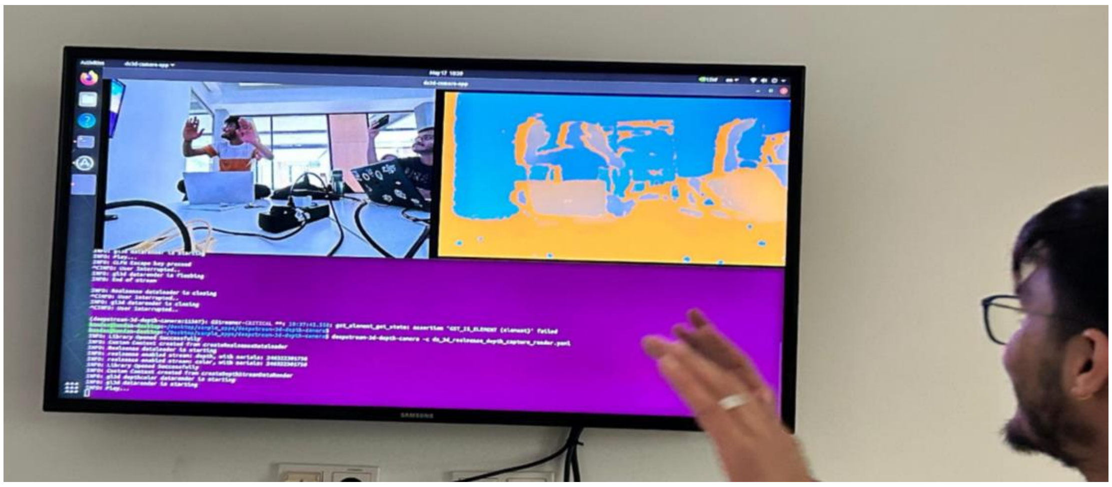


Figure 21: Object detection using Orin Nano

1. OBJECT DETECTION AND ANALYTICS (1 VIDEO INPUT)

This configuration processes a **single video stream** from a stored video file and performs automated detection and analysis in real time.

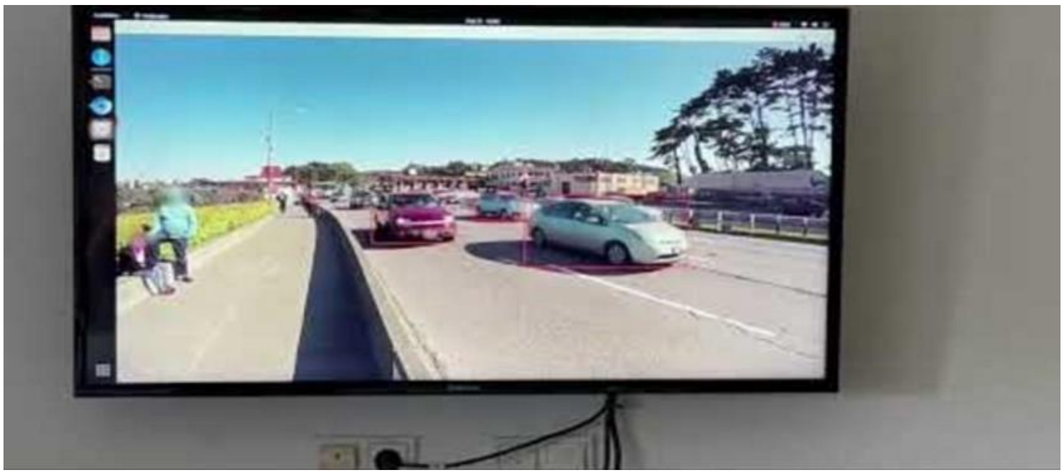


Figure 22: Object detection with existing video input file

2. OBJECT DETECTION AND ANOMALY DETECTION (3 VIDEO INPUTS)

Performance testing with an increased number of input streams revealed that the system experiences significant lag and is prone to crashes under higher loads.

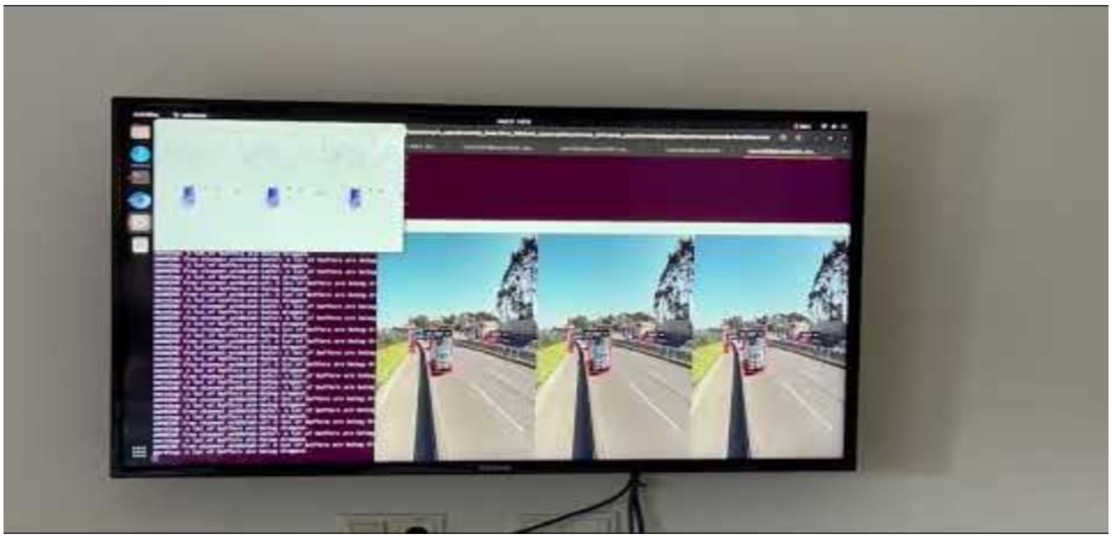


Figure 23: Object detection with multiple video input files

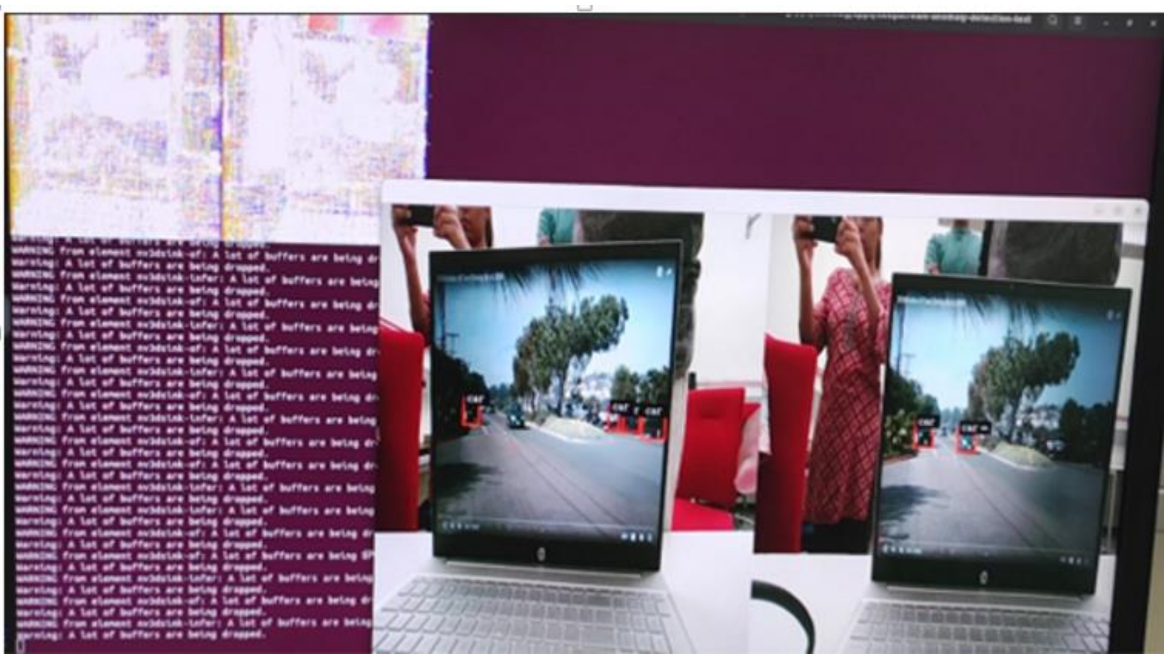
## 8.3 Real-Time Anomaly Detection with Intel RealSense D455 Camera

The anomaly detection pipeline was successfully adapted from an MP4 file source to a real-time feed using the Intel RealSense D455 camera. The GStreamer pipeline was modified to use v4l2src for live video capture, with additional elements (capsfilter, videoconvert, and nvvideoconvert) to ensure format and resolution compatibility (1280×720 @ 30 FPS). Configuration files were updated to accept /dev/video* inputs, and live-source=1 was enabled for continuous streaming.

A shell script was developed to automate multi-camera setup, enabling simultaneous execution of anomaly detection on two RealSense cameras while displaying live previews from virtual devices. The system was enhanced to show both raw camera feeds and processed outputs side-by-side, improving debugging, validation, and demonstration quality.

The final implementation achieved real-time anomaly detection on:

- **a.** Single camera feed
- **b.** Dual camera feeds simultaneously



```
WARNING from element nv3dsink-of: A lot of buffers are being dr
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-infer: A lot of buffers are being
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-of: A lot of buffers are being dr
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-infer: A lot of buffers are being
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-of: A lot of buffers are being dr
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-infer: A lot of buffers are being
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-of: A lot of buffers are being dr
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-infer: A lot of buffers are being
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-of: A lot of buffers are being dr
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-of: A lot of buffers are being dr
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-infer: A lot of buffers are being
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-infer: A lot of buffers are being
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-of: A lot of buffers are being dr
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-of: A lot of buffers are being dr
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-infer: A lot of buffers are being
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-of: A lot of buffers are being dr
Warning: A lot of buffers are being dropped.
WARNING from element nv3dsink-infer: A lot of buffers are being
Warning: A lot of buffers are being dropped.
```

Figure 24: Object detection with 2 camera inputs

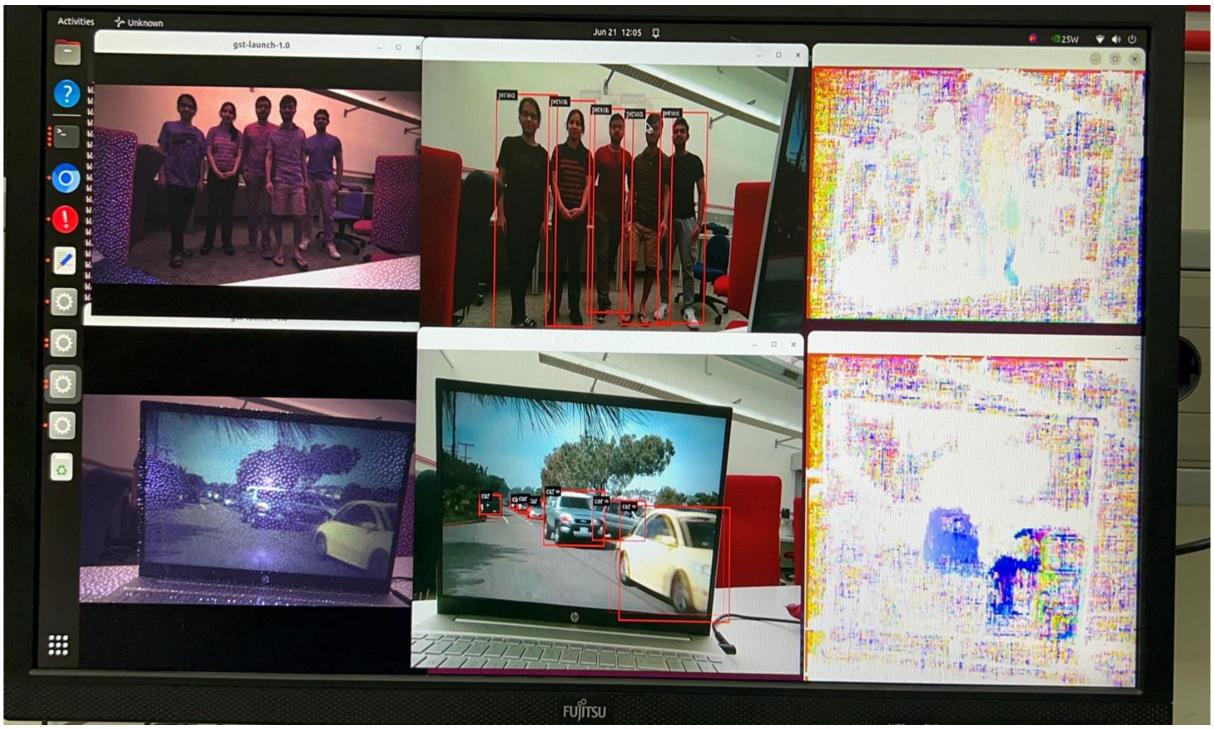
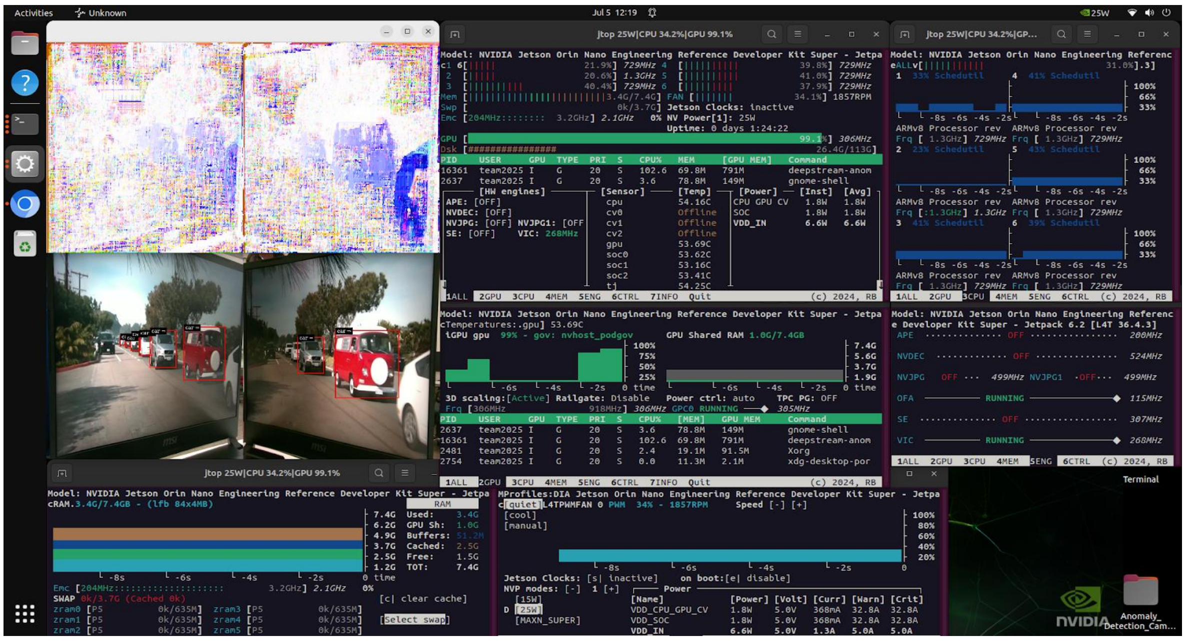


Figure 25: Application (object detection + depth analysis) run on two different cameras

## 9. Benchmarking

Benchmarking was performed using the jtop plugin 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.

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 30fps.
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.



```
jtop 25W|CPU 34.2%|GPU 99.1%
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
c1 6[||||              21.9%] 729MHz 4  [|||||||||        39.8%] 729MHz
  2  [||||             20.6%] 1.3GHz 5  [|||||||||        41.0%] 729MHz
  3  [|||||||||        40.4%] 729MHz 6  [||||||||         37.9%] 729MHz
Mem [||||||||||||||||||||||3.4G/7.4G] FAN [|||||||       34.1%] 1857RPM
Swp [                       0k/3.7G] Jetson Clocks: inactive
Emc [204MHz::::::::  3.2GHz] 2.1GHz   0% NV Power[1]: 25W
                                        Uptime: 0 days 1:24:22
GPU [                                                  99.1%] 306MHz
Dsk [################                                26.4G/113G]
PID    USER      GPU  TYPE  PRI  S   CPU%   MEM     [GPU MEM]  Command
16361  team2025  I    G     20   S   102.6  69.8M   791M       deepstream-anom
2637   team2025  I    G     20   S   3.6    78.8M   149M       gnome-shell
  [HW engines]            [Sensor]   [Temp]   [Power]      [Inst]  [Avg]
 APE: [OFF]                cpu       54.16C   CPU GPU CV   1.8W    1.8W
 NVDEC: [OFF]              cv0       Offline  SOC          1.8W    1.8W
 NVJPG: [OFF] NVJPG1: [OFF cv1       Offline  VDD_IN       6.6W    6.6W
 SE: [OFF]    VIC: 268MHz  cv2       Offline
                           gpu       53.69C
                           soc0      53.62C
                           soc1      53.16C
                           soc2      53.41C
                           tj        54.25C
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL 7INFO Quit                       (c) 2024, RB
```

```
jtop 25W|CPU 34.2%|GP...
Model: NVIDIA Jetson Orin Nano Engineering Referenc
eALLv[||||||||||                          31.0%].3]
 1  33% Schedutil        4  41% Schedutil
                                              100%
                                               66%
                                               33%
 └  └ -8s -6s -4s -2s└  └ -8s -6s -4s -2s
 ARMv8 Processor rev   ARMv8 Processor rev
 Frq [ 1.3GHz] 729MHz Frq [ 1.3GHz] 729MHz
 2  23% Schedutil        5  43% Schedutil
 ARMv8 Processor rev   ARMv8 Processor rev
 Frq [:1.3GHz] 1.3GHz Frq [ 1.3GHz] 729MHz
 3  41% Schedutil        6  39% Schedutil
 ARMv8 Processor rev   ARMv8 Processor rev
 Frq [ 1.3GHz] 729MHz Frq [ 1.3GHz] 729MHz
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL (c) 2024, RB
```

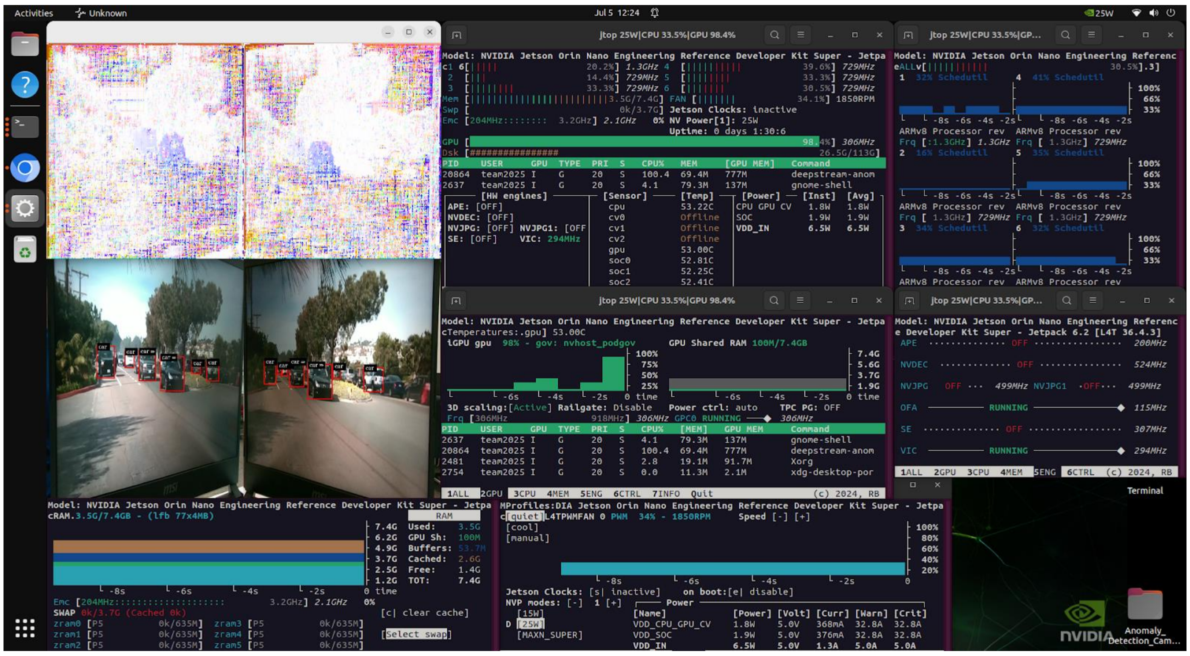
```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
cTemperatures:.gpu] 53.69C
 iGPU gpu  99% - gov: nvhost_podgov      GPU Shared RAM 1.0G/7.4GB
                               100%                              7.4G
                                75%                              5.6G
                                50%                              3.7G
                                25%                              1.9G
 └    └ -6s  └ -4s  └ -2s  0 time  └    └ -6s  └ -4s  └ -2s  0 time
 3D scaling:[Active] Railgate: Disable   Power ctrl: auto    TPC PG: OFF
 Frq [306MHz              918MHz] 306MHz GPC0 RUNNING ──◆ 305MHz
PID    USER      GPU  TYPE  PRI  S   CPU%   [MEM]   GPU MEM   Command
2637   team2025  I    G     20   S   3.6    78.8M   149M      gnome-shell
16361  team2025  I    G     20   S   102.6  69.8M   791M      deepstream-anom
2481   team2025  I    G     20   S   2.4    19.1M   91.5M     Xorg
2754   team2025  I    G     20   S   0.0    11.3M   2.1M      xdg-desktop-por
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL 7INFO Quit                       (c) 2024, RB
```

```
Model: NVIDIA Jetson Orin Nano Engineering Referenc
e Developer Kit Super - Jetpack 6.2 [L4T 36.4.3]
 APE   .............. OFF ...............  200MHz
 NVDEC  ............. OFF ...............  524MHz
 NVJPG   OFF ···  499MHz NVJPG1  ·OFF···  499MHz
 OFA  ────────── RUNNING ─────────────◆  115MHz
 SE  ............... OFF ................  307MHz
 VIC  ────────── RUNNING ─────────────◆  268MHz
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL (c) 2024, RB
```

```
jtop 25W|CPU 34.2%|GPU 99.1%
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
cRAM.3.4G/7.4GB - (lfb 84x4MB)                          RAM
                                          7.4G  Used:      3.4G
                                          6.2G  GPU Sh:    1.0G
                                          4.9G  Buffers:   51.2M
                                          3.7G  Cached:    2.5G
                                          2.5G  Free:      1.5G
                                          1.2G  TOT:       7.4G
      └ -8s    └ -6s    └ -4s    └ -2s    0 time
 Emc [204MHz::::::::::::::::::::   3.2GHz] 2.1GHz   0%
 SWAP 0k/3.7G (Cached 0k)                        [c| clear cache]
 zram0 [P5        0k/635M]  zram3 [P5        0k/635M]
 zram1 [P5        0k/635M]  zram4 [P5        0k/635M]   [Select swap]
 zram2 [P5        0k/635M]  zram5 [P5        0k/635M]
```

```
MProfiles:DIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
c[quiet]L4TPWMFAN 0 PWM  34% - 1857RPM      Speed [-] [+]
 [cool]                                                              100%
 [manual]                                                             80%
                                                                      60%
                                                                      40%
                                                                      20%
              └ -8s        └ -6s        └ -4s        └ -2s            0
 Jetson Clocks: [s| inactive]     on boot:[e| disable]
 NVP modes: [-]  1 [+]  ┌──── Power ─────────────────────────────────────┐
  [15W]                 [Name]            [Power] [Volt] [Curr] [Warn] [Crit]
D [25W]                 VDD_CPU_GPU_CV    1.8W    5.0V   368mA  32.8A  32.8A
  [MAXN_SUPER]          VDD_SOC           1.8W    5.0V   368mA  32.8A  32.8A
                        VDD_IN            6.6W    5.0V   1.3A   5.0A   5.0A
```

Figure 26: Benchmarking at 1280x800 30fps



```
jtop 25W|CPU 33.5%|GPU 98.4%
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
c1 6[||||              20.2%] 1.3GHz 4  [|||||||||        39.6%] 729MHz
  2  [|||              14.4%] 729MHz 5  [||||||||         33.3%] 729MHz
  3  [||||||||         33.3%] 729MHz 6  [|||||||          30.5%] 729MHz
Mem [||||||||||||||||||||||3.5G/7.4G] FAN [|||||||       34.1%] 1850RPM
Swp [                       0k/3.7G] Jetson Clocks: inactive
Emc [204MHz::::::::  3.2GHz] 2.1GHz   0% NV Power[1]: 25W
                                        Uptime: 0 days 1:30:6
GPU [                                                  98.4%] 306MHz
Dsk [################                                26.5G/113G]
PID    USER      GPU  TYPE  PRI  S   CPU%   MEM     [GPU MEM]  Command
20864  team2025  I    G     20   S   100.4  69.4M   777M       deepstream-anom
2637   team2025  I    G     20   S   4.1    79.3M   137M       gnome-shell
  [HW engines]            [Sensor]   [Temp]   [Power]      [Inst]  [Avg]
 APE: [OFF]                cpu       53.22C   CPU GPU CV   1.8W    1.8W
 NVDEC: [OFF]              cv0       Offline  SOC          1.9W    1.9W
 NVJPG: [OFF] NVJPG1: [OFF cv1       Offline  VDD_IN       6.5W    6.5W
 SE: [OFF]    VIC: 294MHz  cv2       Offline
                           gpu       53.00C
                           soc0      52.81C
                           soc1      52.25C
                           soc2      52.41C
```

```
jtop 25W|CPU 33.5%|GP...
Model: NVIDIA Jetson Orin Nano Engineering Referenc
eALLv[||||||||||                          30.5%].3]
 1  32% Schedutil        4  41% Schedutil
                                              100%
                                               66%
                                               33%
 └  └ -8s -6s -4s -2s└  └ -8s -6s -4s -2s
 ARMv8 Processor rev   ARMv8 Processor rev
 Frq [:1.3GHz] 1.3GHz Frq [ 1.3GHz] 729MHz
 2  16% Schedutil        5  35% Schedutil
 ARMv8 Processor rev   ARMv8 Processor rev
 Frq [ 1.3GHz] 729MHz Frq [ 1.3GHz] 729MHz
 3  34% Schedutil        6  32% Schedutil
 ARMv8 Processor rev   ARMv8 Processor rev
```

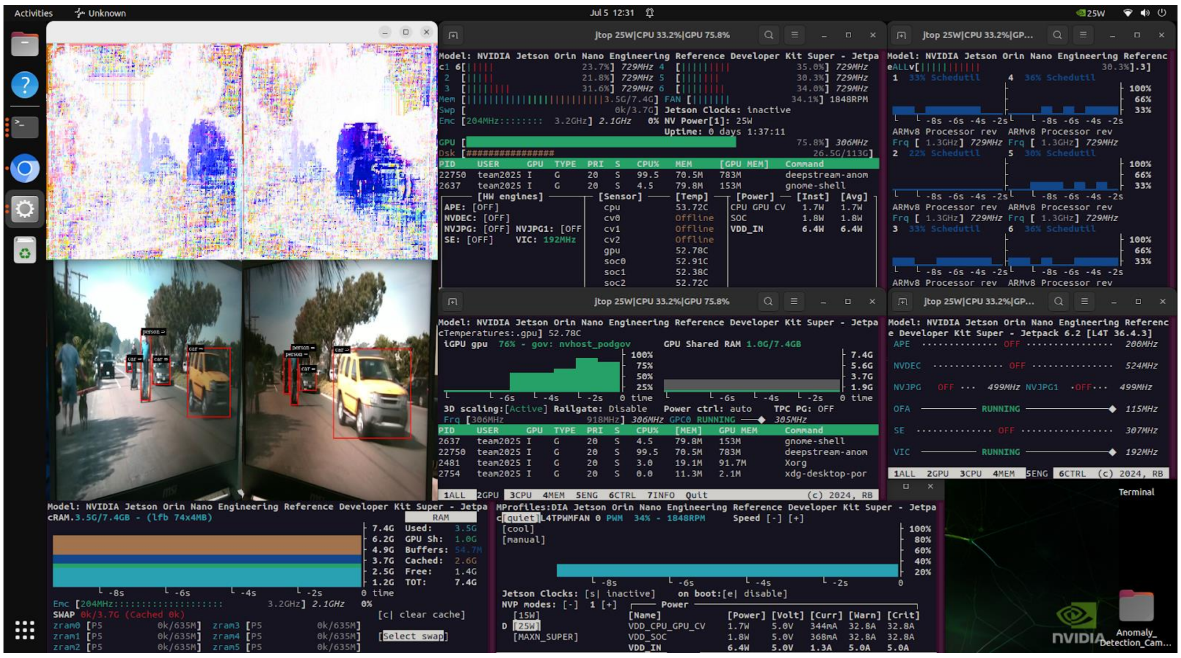
```
jtop 25W|CPU 33.5%|GPU 98.4%
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
cTemperatures:.gpu] 53.00C
 iGPU gpu  98% - gov: nvhost_podgov      GPU Shared RAM 100M/7.4GB
                               100%                              7.4G
                                75%                              5.6G
                                50%                              3.7G
                                25%                              1.9G
 └    └ -6s  └ -4s  └ -2s  0 time  └    └ -6s  └ -4s  └ -2s  0 time
 3D scaling:[Active] Railgate: Disable   Power ctrl: auto    TPC PG: OFF
 Frq [306MHz              918MHz] 306MHz GPC0 RUNNING ──◆ 306MHz
PID    USER      GPU  TYPE  PRI  S   CPU%   [MEM]   GPU MEM   Command
2637   team2025  I    G     20   S   4.1    79.3M   137M      gnome-shell
20864  team2025  I    G     20   S   100.4  69.4M   777M      deepstream-anom
2481   team2025  I    G     20   S   2.8    19.1M   91.7M     Xorg
2754   team2025  I    G     20   S   0.0    11.3M   2.1M      xdg-desktop-por
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL 7INFO Quit                       (c) 2024, RB
```

```
jtop 25W|CPU 33.5%|GP...
Model: NVIDIA Jetson Orin Nano Engineering Referenc
e Developer Kit Super - Jetpack 6.2 [L4T 36.4.3]
 APE   .............. OFF ...............  200MHz
 NVDEC  ............. OFF ...............  524MHz
 NVJPG   OFF ···  499MHz NVJPG1  ·OFF···  499MHz
 OFA  ────────── RUNNING ─────────────◆  115MHz
 SE  ............... OFF ................  307MHz
 VIC  ────────── RUNNING ─────────────◆  294MHz
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL (c) 2024, RB
```

```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
cRAM.3.5G/7.4GB - (lfb 77x4MB)                          RAM
                                          7.4G  Used:      3.5G
                                          6.2G  GPU Sh:    100M
                                          4.9G  Buffers:   53.7M
                                          3.7G  Cached:    2.6G
                                          2.5G  Free:      1.4G
                                          1.2G  TOT:       7.4G
      └ -8s    └ -6s    └ -4s    └ -2s    0 time
 Emc [204MHz::::::::::::::::::::   3.2GHz] 2.1GHz   0%
 SWAP 0k/3.7G (Cached 0k)                        [c| clear cache]
 zram0 [P5        0k/635M]  zram3 [P5        0k/635M]
 zram1 [P5        0k/635M]  zram4 [P5        0k/635M]   [Select swap]
 zram2 [P5        0k/635M]  zram5 [P5        0k/635M]
```

```
MProfiles:DIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
c[quiet]L4TPWMFAN 0 PWM  34% - 1850RPM      Speed [-] [+]
 [cool]                                                              100%
 [manual]                                                             80%
                                                                      60%
                                                                      40%
                                                                      20%
              └ -8s        └ -6s        └ -4s        └ -2s            0
 Jetson Clocks: [s| inactive]     on boot:[e| disable]
 NVP modes: [-]  1 [+]  ┌──── Power ─────────────────────────────────────┐
  [15W]                 [Name]            [Power] [Volt] [Curr] [Warn] [Crit]
D [25W]                 VDD_CPU_GPU_CV    1.8W    5.0V   368mA  32.8A  32.8A
  [MAXN_SUPER]          VDD_SOC           1.9W    5.0V   376mA  32.8A  32.8A
                        VDD_IN            6.5W    5.0V   1.3A   5.0A   5.0A
```

Figure 27: Benchmarking at 1280x720 30fps



```
jtop 25W|CPU 33.2%|GPU 75.8%
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
c1 6[||||                    23.7%] 729MHz 4  [|||||||||            35.0%] 729MHz
  2  [|||||                  21.8%] 729MHz 5  [||||||||             30.3%] 729MHz
  3  [||||||||               31.6%] 729MHz 6  [||||||||             34.0%] 729MHz
Mem [|||||||||||||||||||||||||||3.5G/7.4G] FAN [||||||||          34.1%] 1848RPM
Swp [                          0k/3.7G] Jetson Clocks: inactive
Emc [204MHz::::::::  3.2GHz] 2.1GHz   0% NV Power[1]: 25W
                                       Uptime: 0 days 1:37:11
GPU [                                                       75.8%] 306MHz
Dsk [################                                      26.5G/113G]
PID    USER      GPU  TYPE  PRI  S   CPU%   MEM     [GPU MEM]   Command
22750  team2025  I    G     20   S   99.5   70.5M   783M        deepstream-anom
2637   team2025  I    G     20   S   4.5    79.8M   153M        gnome-shell
 ──── [HW engines] ──────── [Sensor] ──── [Temp] ── [Power] ── [Inst]  [Avg]
 APE: [OFF]                   cpu          53.72C    CPU GPU CV   1.7W    1.7W
 NVDEC: [OFF]                 cv0          Offline   SOC          1.8W    1.8W
 NVJPG: [OFF] NVJPG1: [OFF    cv1          Offline   VDD_IN       6.4W    6.4W
 SE: [OFF]     VIC: 192MHz    cv2          Offline
                              gpu          52.78C
                              soc0         52.91C
                              soc1         52.38C
                              soc2         52.72C
```

```
jtop 25W|CPU 33.2%|GP...
Model: NVIDIA Jetson Orin Nano Engineering Referenc
eALLv[||||||||||||                           30.3%].3]
 1  33% Schedutil        4  36% Schedutil
                                                 100%
                                                  66%
                                                  33%
 └   └ -8s -6s -4s -2s └   └ -8s -6s -4s -2s
 ARMv8 Processor rev   ARMv8 Processor rev
 Frq [ 1.3GHz] 729MHz Frq [ 1.3GHz] 729MHz
 2  22% Schedutil        5  30% Schedutil
                                                 100%
                                                  66%
                                                  33%
 └   └ -8s -6s -4s -2s └   └ -8s -6s -4s -2s
 ARMv8 Processor rev   ARMv8 Processor rev
 Frq [ 1.3GHz] 729MHz Frq [ 1.3GHz] 729MHz
 3  33% Schedutil        6  36% Schedutil
                                                 100%
                                                  66%
                                                  33%
 └   └ -8s -6s -4s -2s └   └ -8s -6s -4s -2s
 ARMv8 Processor rev   ARMv8 Processor rev
```

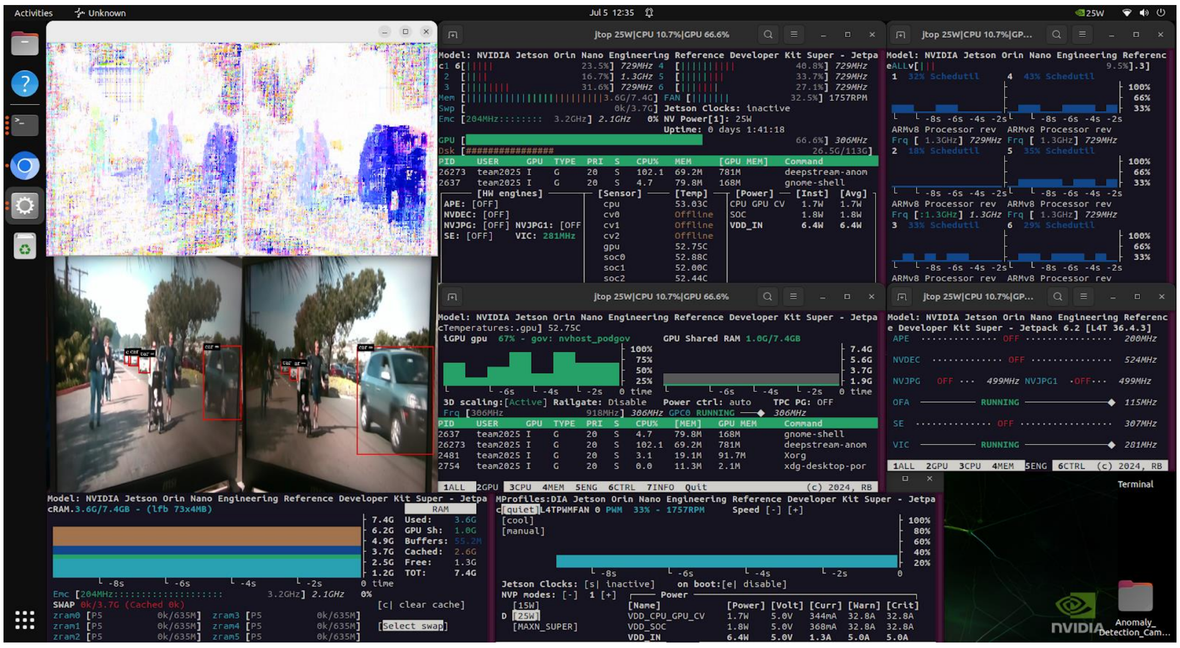
```
jtop 25W|CPU 33.2%|GPU 75.8%
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
cTemperatures:.gpu] 52.78C
 iGPU gpu  76% - gov: nvhost_podgov      GPU Shared RAM 1.0G/7.4GB
                                  100%                                 7.4G
                                   75%                                 5.6G
                                   50%                                 3.7G
                                   25%                                 1.9G
 └     └ -6s  └ -4s  └ -2s   0 time   └     └ -6s  └ -4s  └ -2s   0 time
 3D scaling:[Active] Railgate: Disable    Power ctrl: auto    TPC PG: OFF
 Frq [306MHz               918MHz] 306MHz GPC0 RUNNING ──◆ 305MHz
PID    USER      GPU  TYPE  PRI  S   CPU%   [MEM]   GPU MEM   Command
2637   team2025  I    G     20   S   4.5    79.8M   153M      gnome-shell
22750  team2025  I    G     20   S   99.5   70.5M   783M      deepstream-anom
2481   team2025  I    G     20   S   3.0    19.1M   91.7M     Xorg
2754   team2025  I    G     20   S   0.0    11.3M   2.1M      xdg-desktop-por

 1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL 7INFO Quit                    (c) 2024, RB
```

```
jtop 25W|CPU 33.2%|GP...
Model: NVIDIA Jetson Orin Nano Engineering Referenc
e Developer Kit Super - Jetpack 6.2 [L4T 36.4.3]
 APE   ·············· OFF ················  200MHz

 NVDEC  ············· OFF ···············  524MHz

 NVJPG   OFF ···  499MHz NVJPG1  ·OFF···   499MHz

 OFA  ──────────── RUNNING ─────────────◆  115MHz

 SE  ·············· OFF ················   307MHz

 VIC  ──────────── RUNNING ─────────────◆  192MHz

 1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL (c) 2024, RB
```

```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
cRAM.3.5G/7.4GB - (lfb 74x4MB)                                     RAM
                                                   7.4G  Used:     3.5G
                                                   6.2G  GPU Sh:   1.0G
                                                   4.9G  Buffers:  54.7M
                                                   3.7G  Cached:   2.6G
                                                   2.5G  Free:     1.4G
                                                   1.2G  TOT:      7.4G
        └ -8s       └ -6s       └ -4s      └ -2s   0 time
 Emc [204MHz::::::::::::::::::::      3.2GHz] 2.1GHz   0%
 SWAP 0k/3.7G (Cached 0k)                           [c| clear cache]
 zram0 [P5        0k/635M]  zram3 [P5        0k/635M]
 zram1 [P5        0k/635M]  zram4 [P5        0k/635M]  [Select swap]
 zram2 [P5        0k/635M]  zram5 [P5        0k/635M]
```

```
MProfiles:DIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
c[quiet]L4TPWMFAN 0 PWM  34% - 1848RPM     Speed [-] [+]
 [cool]                                                              100%
 [manual]                                                             80%
                                                                      60%
                                                                      40%
                                                                      20%
           └ -8s       └ -6s        └ -4s       └ -2s                 0
 Jetson Clocks: [s| inactive]    on boot:[e| disable]
 NVP modes: [-]  1 [+]   ┌──── Power ──────────────────────────────────┐
   [15W]                  [Name]            [Power] [Volt] [Curr]  [Warn] [Crit]
 D [25W]                  VDD_CPU_GPU_CV    1.7W    5.0V   344mA   32.8A  32.8A
   [MAXN_SUPER]           VDD_SOC           1.8W    5.0V   368mA   32.8A  32.8A
                          VDD_IN            6.4W    5.0V   1.3A    5.0A   5.0A
```

Figure 28: Benchmarking at 848x480 30fps



```
jtop 25W|CPU 10.7%|GPU 66.6%
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
c1 6[||||                    23.5%] 729MHz 4  [||||||||||           40.8%] 729MHz
  2  [|||                    16.7%] 1.3GHz 5  [||||||||             33.7%] 729MHz
  3  [||||||||               31.6%] 729MHz 6  [|||||||              27.1%] 729MHz
Mem [|||||||||||||||||||||||||||3.6G/7.4G] FAN [||||||||          32.5%] 1757RPM
Swp [                          0k/3.7G] Jetson Clocks: inactive
Emc [204MHz::::::::  3.2GHz] 2.1GHz   0% NV Power[1]: 25W
                                       Uptime: 0 days 1:41:18
GPU [                                                       66.6%] 306MHz
Dsk [################                                      26.5G/113G]
PID    USER      GPU  TYPE  PRI  S   CPU%   MEM     [GPU MEM]   Command
26273  team2025  I    G     20   S   102.1  69.2M   781M        deepstream-anom
2637   team2025  I    G     20   S   4.7    79.8M   168M        gnome-shell
 ──── [HW engines] ──────── [Sensor] ──── [Temp] ── [Power] ── [Inst]  [Avg]
 APE: [OFF]                   cpu          53.03C    CPU GPU CV   1.7W    1.7W
 NVDEC: [OFF]                 cv0          Offline   SOC          1.8W    1.8W
 NVJPG: [OFF] NVJPG1: [OFF    cv1          Offline   VDD_IN       6.4W    6.4W
 SE: [OFF]     VIC: 281MHz    cv2          Offline
                              gpu          52.75C
                              soc0         52.88C
                              soc1         52.00C
                              soc2         52.44C
```

```
jtop 25W|CPU 10.7%|GP...
Model: NVIDIA Jetson Orin Nano Engineering Referenc
eALLv[|||                                     9.5%].3]
 1  32% Schedutil        4  43% Schedutil
                                                 100%
                                                  66%
                                                  33%
 └   └ -8s -6s -4s -2s └   └ -8s -6s -4s -2s
 ARMv8 Processor rev   ARMv8 Processor rev
 Frq [ 1.3GHz] 729MHz Frq [ 1.3GHz] 729MHz
 2  18% Schedutil        5  35% Schedutil
                                                 100%
                                                  66%
                                                  33%
 └   └ -8s -6s -4s -2s └   └ -8s -6s -4s -2s
 ARMv8 Processor rev   ARMv8 Processor rev
 Frq [:1.3GHz] 1.3GHz Frq [ 1.3GHz] 729MHz
 3  33% Schedutil        6  29% Schedutil
                                                 100%
                                                  66%
                                                  33%
 └   └ -8s -6s -4s -2s └   └ -8s -6s -4s -2s
 ARMv8 Processor rev   ARMv8 Processor rev
```

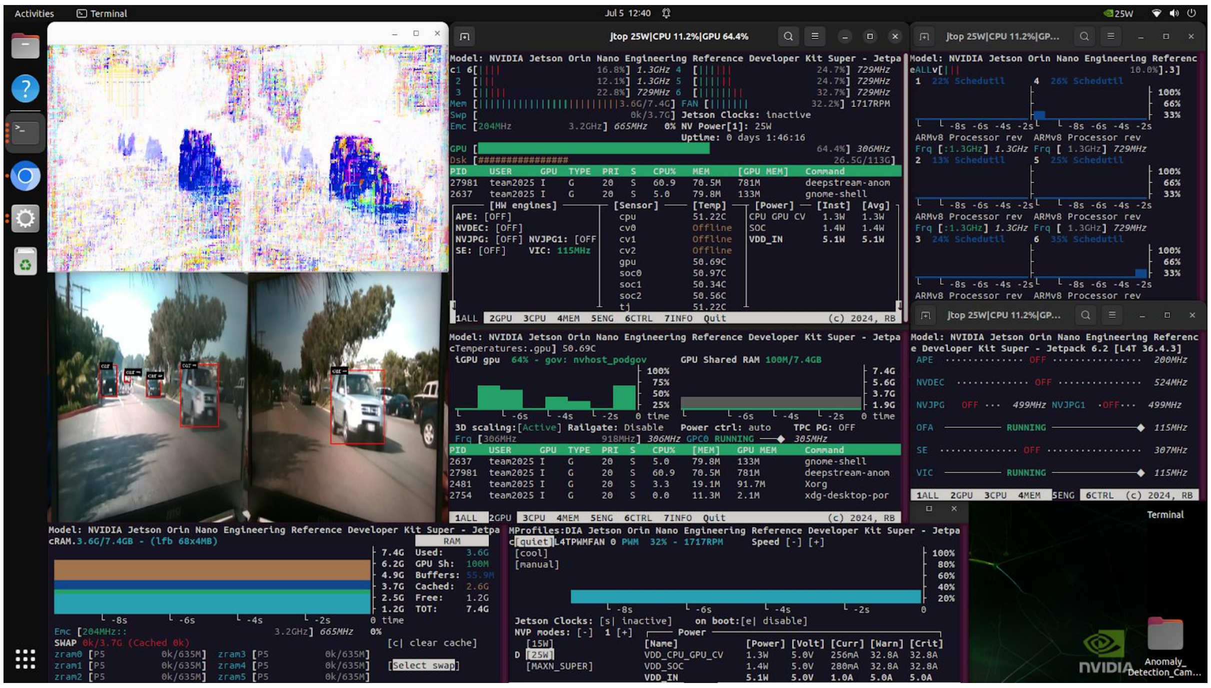
```
jtop 25W|CPU 10.7%|GPU 66.6%
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
cTemperatures:.gpu] 52.75C
 iGPU gpu  67% - gov: nvhost_podgov      GPU Shared RAM 1.0G/7.4GB
                                  100%                                 7.4G
                                   75%                                 5.6G
                                   50%                                 3.7G
                                   25%                                 1.9G
 └     └ -6s  └ -4s  └ -2s   0 time   └     └ -6s  └ -4s  └ -2s   0 time
 3D scaling:[Active] Railgate: Disable    Power ctrl: auto    TPC PG: OFF
 Frq [306MHz               918MHz] 306MHz GPC0 RUNNING ──◆ 306MHz
PID    USER      GPU  TYPE  PRI  S   CPU%   [MEM]   GPU MEM   Command
2637   team2025  I    G     20   S   4.7    79.8M   168M      gnome-shell
26273  team2025  I    G     20   S   102.1  69.2M   781M      deepstream-anom
2481   team2025  I    G     20   S   3.1    19.1M   91.7M     Xorg
2754   team2025  I    G     20   S   0.0    11.3M   2.1M      xdg-desktop-por

 1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL 7INFO Quit                    (c) 2024, RB
```

```
jtop 25W|CPU 10.7%|GP...
Model: NVIDIA Jetson Orin Nano Engineering Referenc
e Developer Kit Super - Jetpack 6.2 [L4T 36.4.3]
 APE   ·············· OFF ················  200MHz

 NVDEC  ············· OFF ···············  524MHz

 NVJPG   OFF ···  499MHz NVJPG1  ·OFF···   499MHz

 OFA  ──────────── RUNNING ─────────────◆  115MHz

 SE  ·············· OFF ················   307MHz

 VIC  ──────────── RUNNING ─────────────◆  281MHz

 1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL (c) 2024, RB
```

```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
cRAM.3.6G/7.4GB - (lfb 73x4MB)                                     RAM
                                                   7.4G  Used:     3.6G
                                                   6.2G  GPU Sh:   1.0G
                                                   4.9G  Buffers:  55.2M
                                                   3.7G  Cached:   2.6G
                                                   2.5G  Free:     1.3G
                                                   1.2G  TOT:      7.4G
        └ -8s       └ -6s       └ -4s      └ -2s   0 time
 Emc [204MHz::::::::::::::::::::      3.2GHz] 2.1GHz   0%
 SWAP 0k/3.7G (Cached 0k)                           [c| clear cache]
 zram0 [P5        0k/635M]  zram3 [P5        0k/635M]
 zram1 [P5        0k/635M]  zram4 [P5        0k/635M]  [Select swap]
 zram2 [P5        0k/635M]  zram5 [P5        0k/635M]
```

```
MProfiles:DIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
c[quiet]L4TPWMFAN 0 PWM  33% - 1757RPM     Speed [-] [+]
 [cool]                                                              100%
 [manual]                                                             80%
                                                                      60%
                                                                      40%
                                                                      20%
           └ -8s       └ -6s        └ -4s       └ -2s                 0
 Jetson Clocks: [s| inactive]    on boot:[e| disable]
 NVP modes: [-]  1 [+]   ┌──── Power ──────────────────────────────────┐
   [15W]                  [Name]            [Power] [Volt] [Curr]  [Warn] [Crit]
 D [25W]                  VDD_CPU_GPU_CV    1.7W    5.0V   344mA   32.8A  32.8A
   [MAXN_SUPER]           VDD_SOC           1.8W    5.0V   368mA   32.8A  32.8A
                          VDD_IN            6.4W    5.0V   1.3A    5.0A   5.0A
```

Figure 29: Benchmarking at 640x350 30fps



```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
c1 6[||||        16.8%] 1.3GHz 4 [|||||       24.7%] 729MHz
 2 [|||          12.1%] 1.3GHz 5 [|||||       24.7%] 729MHz
 3 [||||         22.8%] 729MHz 6 [||||||||    32.7%] 729MHz
Mem [||||||||||||||||||||||3.6G/7.4G] FAN [||||||||    32.2%] 1717RPM
Swp [                     0k/3.7G] Jetson Clocks: inactive
Emc [204MHz        3.2GHz] 665MHz  0% NV Power[1]: 25W
                                   Uptime: 0 days 1:46:16
GPU [                                       64.4%] 306MHz
Dsk [################                          26.5G/113G]
PID   USER     GPU  TYPE  PRI  S  CPU%  MEM    [GPU MEM]  Command
27981 team2025 I    G     20   S  60.9  70.5M  781M       deepstream-anon
2637  team2025 I    G     20   S  5.0   79.8M  133M       gnome-shell
   [HW engines]          [Sensor]  [Temp]   [Power]     [Inst] [Avg]
APE: [OFF]               cpu       51.22C   CPU GPU CV  1.3W   1.3W
NVDEC: [OFF]             cv0       Offline  SOC         1.4W   1.4W
NVJPG: [OFF] NVJPG1: [OFF] cv1     Offline  VDD_IN      5.1W   5.1W
SE: [OFF]    VIC: 115MHz cv2       Offline
                         gpu       50.69C
                         soc0      50.97C
                         soc1      50.34C
                         soc2      50.56C
                         tj        51.22C
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL 7INFO Quit           (c) 2024, RB
```

```
Model: NVIDIA Jetson Orin Nano Engineering Referenc
eALLv[|||                              10.8%].3]
1  22% Schedutil       4  26% Schedutil
ARMv8 Processor rev  ARMv8 Processor rev
Frq [:1.3GHz] 1.3GHz Frq [ 1.3GHz] 729MHz
2  13% Schedutil       5  25% Schedutil
ARMv8 Processor rev  ARMv8 Processor rev
Frq [:1.3GHz] 1.3GHz Frq [ 1.3GHz] 729MHz
3  24% Schedutil       6  35% Schedutil
ARMv8 Processor rev  ARMv8 Processor rev
```

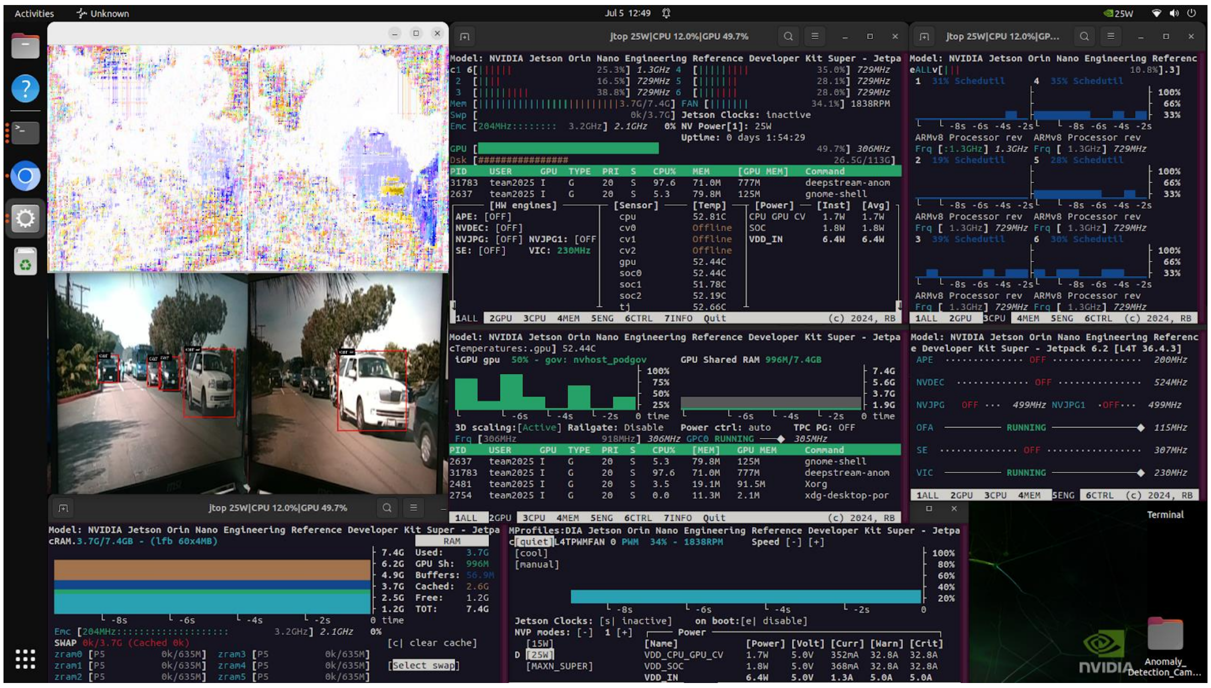
```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
cTemperatures:.gpu] 50.69C
iGPU gpu  64% - gov: nvhost_podgov     GPU Shared RAM 100M/7.4GB
3D scaling:[Active] Railgate: Disable  Power ctrl: auto  TPC PG: OFF
Frq [306MHz          918MHz] 306MHz GPC0 RUNNING ── 305MHz
PID   USER     GPU  TYPE  PRI  S  CPU%  [MEM]  GPU MEM  Command
2637  team2025 I    G     20   S  5.0   79.8M  133M     gnome-shell
27981 team2025 I    G     20   S  60.9  70.5M  781M     deepstream-anon
2481  team2025 I    G     20   S  3.3   19.1M  91.7M    Xorg
2754  team2025 I    G     20   S  0.0   11.3M  2.1M     xdg-desktop-por
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL 7INFO Quit           (c) 2024, RB
```

```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super
e Developer Kit Super - Jetpack 6.2 [L4T 36.4.3]
APE    ............ OFF ..............  200MHz
NVDEC  ............ OFF ..............  524MHz
NVJPG  OFF ...  499MHz NVJPG1 .OFF...   499MHz
OFA    ────────── RUNNING ──────────◆  115MHz
SE     ............ OFF ..............  307MHz
VIC    ────────── RUNNING ──────────◆  115MHz
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL (c) 2024, RB
```

```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
cRAM.3.6G/7.4GB - (lfb 68x4MB)
                   RAM
         7.4G  Used:     3.6G
         6.2G  GPU Sh:   100M
         4.9G  Buffers:  55.9M
         3.7G  Cached:   2.6G
         2.5G  Free:     1.2G
         1.2G  TOT:      7.4G
Emc [204MHz::          3.2GHz] 665MHz  0%
SWAP 0k/3.7G (Cached 0k)              [c| clear cache]
zram0 [P5   0k/635M]  zram3 [P5   0k/635M]
zram1 [P5   0k/635M]  zram4 [P5   0k/635M]  [Select swap]
zram2 [P5   0k/635M]  zram5 [P5   0k/635M]
```

```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
nProfiles:DIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
c[quiet]L4TPWMFAN 0 PWM  32% - 1717RPM      Speed [-] [+]
 [cool]
 [manual]
Jetson Clocks: [s| inactive]    on boot:[e| disable]
NVP modes: [-]  1 [+]     Power
 [15W]         [Name]         [Power] [Volt] [Curr] [Warn] [Crit]
D [25W]        VDD_CPU_GPU_CV 1.3W    5.0V   256mA  32.8A  32.8A
 [MAXN_SUPER]  VDD_SOC        1.4W    5.0V   280mA  32.8A  32.8A
               VDD_IN         5.1W    5.0V   1.0A   5.0A   5.0A
```

Figure 30: Benchmarking at 640x350 15fps



```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
c1 6[||||||      25.3%] 1.3GHz 4 [||||||||||  35.0%] 729MHz
 2 [||||         16.5%] 729MHz 5 [||||||||    28.1%] 729MHz
 3 [||||||||||   38.8%] 729MHz 6 [||||||||    28.0%] 729MHz
Mem [||||||||||||||||||||||3.7G/7.4G] FAN [||||||||    34.1%] 1838RPM
Swp [                     0k/3.7G] Jetson Clocks: inactive
Emc [204MHz::::::::  3.2GHz] 2.1GHz  0% NV Power[1]: 25W
                                   Uptime: 0 days 1:54:29
GPU [                                       49.7%] 306MHz
Dsk [################                          26.5G/113G]
PID   USER     GPU  TYPE  PRI  S  CPU%  MEM    [GPU MEM]  Command
31783 team2025 I    G     20   S  97.6  71.0M  777M       deepstream-anon
2637  team2025 I    G     20   S  5.3   79.8M  125M       gnome-shell
   [HW engines]          [Sensor]  [Temp]   [Power]     [Inst] [Avg]
APE: [OFF]               cpu       52.81C   CPU GPU CV  1.7W   1.7W
NVDEC: [OFF]             cv0       Offline  SOC         1.8W   1.8W
NVJPG: [OFF] NVJPG1: [OFF] cv1     Offline  VDD_IN      6.4W   6.4W
SE: [OFF]    VIC: 230MHz cv2       Offline
                         gpu       52.44C
                         soc0      52.44C
                         soc1      51.78C
                         soc2      52.19C
                         tj        52.66C
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL 7INFO Quit           (c) 2024, RB
```

```
Model: NVIDIA Jetson Orin Nano Engineering Referenc
eALLv[||                               10.8%].3]
1  31% Schedutil       4  35% Schedutil
ARMv8 Processor rev  ARMv8 Processor rev
Frq [:1.3GHz] 1.3GHz Frq [ 1.3GHz] 729MHz
2  19% Schedutil       5  28% Schedutil
ARMv8 Processor rev  ARMv8 Processor rev
Frq [ 1.3GHz] 729MHz Frq [ 1.3GHz] 729MHz
3  39% Schedutil       6  38% Schedutil
ARMv8 Processor rev  ARMv8 Processor rev
Frq [ 1.3GHz] 729MHz Frq [ 1.3GHz] 729MHz
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL (c) 2024, RB
```

```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
cTemperatures:.gpu] 52.44C
iGPU gpu  50% - gov: nvhost_podgov     GPU Shared RAM 996M/7.4GB
3D scaling:[Active] Railgate: Disable  Power ctrl: auto  TPC PG: OFF
Frq [306MHz          918MHz] 306MHz GPC0 RUNNING ── 305MHz
PID   USER     GPU  TYPE  PRI  S  CPU%  [MEM]  GPU MEM  Command
2637  team2025 I    G     20   S  5.3   79.8M  125M     gnome-shell
31783 team2025 I    G     20   S  97.6  71.0M  777M     deepstream-anon
2481  team2025 I    G     20   S  3.5   19.1M  91.5M    Xorg
2754  team2025 I    G     20   S  0.0   11.3M  2.1M     xdg-desktop-por
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL 7INFO Quit           (c) 2024, RB
```

```
Model: NVIDIA Jetson Orin Nano Engineering Referenc
e Developer Kit Super - Jetpack 6.2 [L4T 36.4.3]
APE    ............ OFF ..............  200MHz
NVDEC  ............ OFF ..............  524MHz
NVJPG  OFF ...  499MHz NVJPG1 .OFF...   499MHz
OFA    ────────── RUNNING ──────────◆  115MHz
SE     ............ OFF ..............  307MHz
VIC    ────────── RUNNING ──────────◆  230MHz
1ALL 2GPU 3CPU 4MEM 5ENG 6CTRL (c) 2024, RB
```

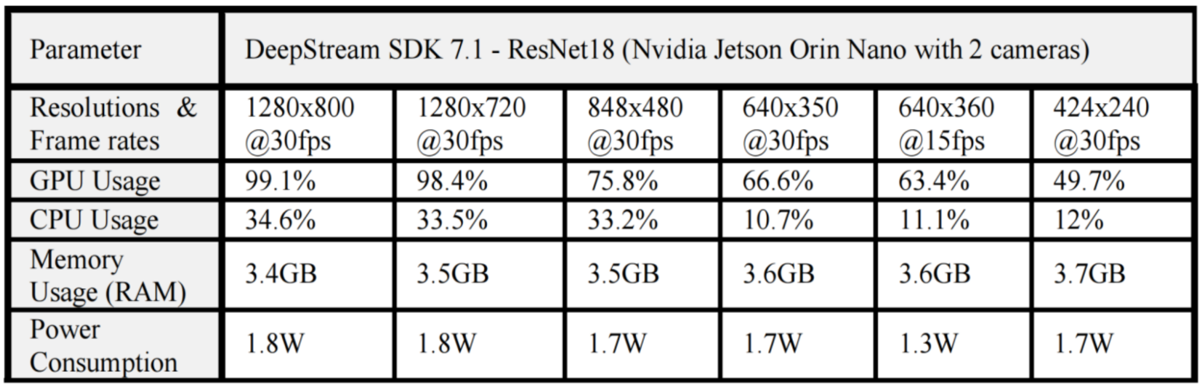
```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
cRAM.3.7G/7.4GB - (lfb 60x4MB)
                   RAM
         7.4G  Used:     3.7G
         6.2G  GPU Sh:   996M
         4.9G  Buffers:  56.9M
         3.7G  Cached:   2.6G
         2.5G  Free:     1.2G
         1.2G  TOT:      7.4G
Emc [204MHz::::::::::::::::::  3.2GHz] 2.1GHz  0%
SWAP 0k/3.7G (Cached 0k)              [c| clear cache]
zram0 [P5   0k/635M]  zram3 [P5   0k/635M]
zram1 [P5   0k/635M]  zram4 [P5   0k/635M]  [Select swap]
zram2 [P5   0k/635M]  zram5 [P5   0k/635M]
```

```
Model: NVIDIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
nProfiles:DIA Jetson Orin Nano Engineering Reference Developer Kit Super - Jetpa
c[quiet]L4TPWMFAN 0 PWM  34% - 1838RPM      Speed [-] [+]
 [cool]
 [manual]
Jetson Clocks: [s| inactive]    on boot:[e| disable]
NVP modes: [-]  1 [+]     Power
 [15W]         [Name]         [Power] [Volt] [Curr] [Warn] [Crit]
D [25W]        VDD_CPU_GPU_CV 1.7W    5.0V   352mA  32.8A  32.8A
 [MAXN_SUPER]  VDD_SOC        1.8W    5.0V   368mA  32.8A  32.8A
               VDD_IN         6.4W    5.0V   1.3A   5.0A   5.0A
```

Figure 31: Benchmarking at 424x240 30fps

# 10.Summary

(The used model of Orin nano has 8 GB RAM)



| 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.
Table 3: Observation Summary

These results confirm that the ResNet-18 application-specific model is capable of real-time inference even at reduced resolutions and resource foot prints, enabling flexible deployment scenarios on edge hardware like Jetson Orin Nano.

DeepStream is the go-to solution for high-performance, scalable, real-time object detection on NVIDIA Jetson devices, particularly in scenarios involving multiple video streams and integration with AI models.

Overall, these results confirm that the ResNet-18 application-specific model is capable of real-time inference even at reduced resolutions and resource foot prints, , enabling flexible deployment scenarios on edge hardware like Jetson Orin Nano which offers a much better balance of performance and scalability, making it the preferred option for most demanding use cases.

# 11.Future Scope

Looking ahead, the "Vision AI on NVIDIA Jetson Nano and NVIDIA Jetson Orin Nano" project presents multiple opportunities for enhancement and diversification:

1. Enhanced Distance Measurement
   By further leveraging Intel® RealSense™ Depth Cameras, the system can achieve more accurate and reliable distance estimation. Improved spatial awareness will be particularly beneficial for applications such as robotics, autonomous navigation, and industrial safety systems.
2. Advanced Object Detection Models
   Upgrading from the current ResNet-18 architecture to more sophisticated and robust models like YOLOv7 can significantly improve detection accuracy, especially for complex or subtle anomalies. This transition will help the system perform better under challenging real-world conditions.
3. Depth-Based Analysis for Anomaly Detection
   Incorporating depth data from RealSense sensors into the detection pipeline can enhance spatial understanding, enabling better classification of anomalies and improving performance in cluttered or dynamic environments.
4. Custom Model Development for Specific Applications
   Creating and deploying task-specific machine learning models—optimized for particular environments, object types, or operational constraints—can make the system highly adaptable. Platforms like EdgeImpulse can streamline this process, ensuring efficient deployment on edge hardware.
5. Expanded Use Cases
   With targeted adaptations, the system can serve diverse domains, such as:
   - Smart Surveillance Systems – Real-time monitoring with improved threat detection and reduced false positives.
   - Autonomous Drones – Combining spatial awareness and anomaly detection for safe, efficient flight operations.
   - Industrial Automation – Monitoring machinery and workspaces to detect operational anomalies and ensure workplace safety.

These advancements will make the system more powerful, versatile, and reliable for edge deployments, unlocking new possibilities in security, automation, and AI-driven spatial intelligence.

# 12.References

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