## **Introduction**

This document details the complete process of deploying a trained YOLOv8 model on the NVIDIA Jetson Orin Nano. It covers system environment verification, installation of required dependencies including PyTorch and Ultralytics YOLO, setting up the Jupyter Notebook environment with GPU support, and executing inference on images using the GPU-accelerated model.

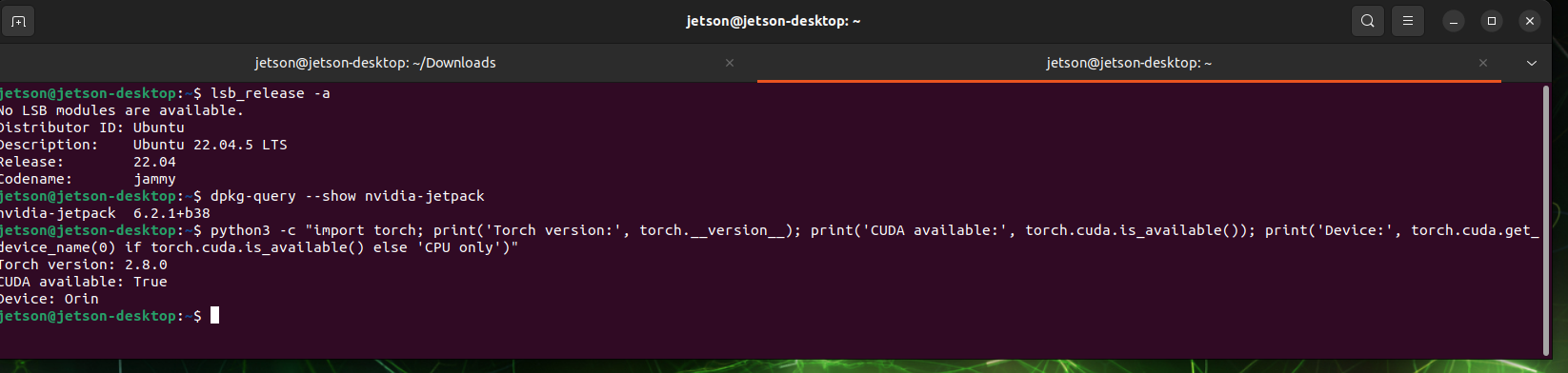
## **2. System and Environment Overview**

* **Jetson Orin Nano Board**
* **Ubuntu Version:** 22.04.5 LTS (Jammy)
* **NVIDIA JetPack Version:** 6.2.1+b38
* **Python Version:** 3.10.13
* **CUDA & cuDNN:** Installed and configured as part of JetPack

## **3. Verification of system and gpu**

* Ubuntu version : lsb\_release -a
* JetPack Version : dpkg-query --show nvidia-jetpack
* CUDA Availability and GPU: python3 -c "import torch; print('Torch version:', torch.\_\_version\_\_); print('CUDA available:', torch.cuda.is\_available()); print('Device:', torch.cuda.get\_device\_name(0) if torch.cuda.is\_available() else 'CPU only')"

Results



**4. Installation of Dependencies**

### **Why Custom Wheel Files Are Required**

Jetson devices come preinstalled with JetPack, which includes a default version of PyTorch. However, this version may not be compiled with full CUDA support, preventing GPU acceleration in deep learning applications.

To ensure GPU acceleration on the **Jetson Orin Nano**, we use **custom .whl (wheel) files** provided by NVIDIA, compiled specifically for Jetson boards with CUDA compatibility.

The required PyTorch wheel files were obtained from the following repository, which provides community-maintained builds specifically targeting JetPack 6 with CUDA 12.2:

* **Repository URL:**<https://pypi.jetson-ai-lab.io/jp6/cu126>

This source provides Python 3.10-compatible builds for ARM64 architecture, which align with the system configuration on Jetson Orin Nano.

#### **Downloaded Files**

The following files were downloaded and installed:

* torch-2.8.0-cp310-cp310-linux\_aarch64.whl
* torchvision-0.23.0-cp310-cp310-linux\_aarch64.whl
* Torchaudio-2.8.0-cp310-cp310-linux\_aarch64.whl

**5 . Environment Configuration and GPU Integration**

After downloading and installing the required wheel files, attention was directed to environment management and ensuring compatibility with GPU-accelerated PyTorch.

It was identified that the Conda base environment on the Jetson Orin Nano was using Python 3.13, located at /home/jetson/miniconda3/bin/python3, whereas the GPU-enabled PyTorch installation was made using the system Python 3.10, located at /usr/bin/python3. This mismatch caused errors when attempting to import PyTorch inside the Conda environment, such as ModuleNotFoundError: No module named 'torch'.

To resolve this, the Conda base environment was deactivated using the command:

***conda deactivate***

To prevent Conda from automatically activating the base environment in new terminal sessions, the following command was executed:

***conda config --set auto\_activate\_base false***

After these steps, terminal sessions defaulted to the system Python environment, where the installed wheel files and CUDA support were correctly accessible.

The correct Python interpreter in use was verified with:

***which python3***

The output confirmed usage of the system Python:

***/usr/bin/python3***

Following this, configuration of Jupyter Notebook was necessary to ensure it would operate using the correct Python environment.

The ipykernel package was installed under the system Python environment to allow Jupyter to recognize it as a kernel:

***python3 -m pip install ipykernel***

A new Jupyter kernel was then registered with:

***python3 -m ipykernel install --user --name=jetson-system --display-name "Python 3 (Jetson System)"***

This created a new selectable kernel in the Jupyter interface named "Python 3 (Jetson System)".

From the notebook interface, the kernel was switched by navigating to:

***Kernel > Change Kernel > Python 3 (Jetson System)***

This ensured that the notebook session would run within the same Python 3.10 system environment where the GPU-compatible versions of PyTorch, TorchVision, and TorchAudio were installed.

To verify GPU access within the notebook, the following python file was executed [yolo\_check](https://drive.google.com/file/d/1WBu9ou4rEQiZ6PBmjJe04BzAw1TtwiJQ/view?usp=drive_link) was executed and the output was CUDA: True which confirmed gpu access, and the detections were stored in separate folder:

