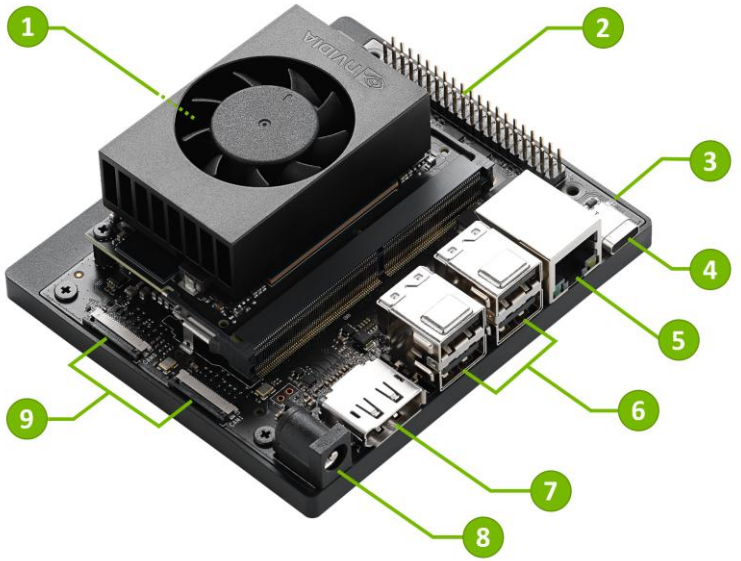

AI Frameworks (PyTorch, TensorFlow) – Run pre-trained models efficiently.

A Practical Guide to Affordable AI at the Edge

Supported Frameworks

- Jetson Orin Nano fully supports leading AI frameworks, allowing you to **train on powerful systems and deploy on edge** with minimal conversion steps:

Framework	Support Level	Optimization	Notes
PyTorch	Full	TensorRT via torch2trt/ONNX	Research & Prototyping
TensorFlow	Full	TensorRT via TF-TRT/ONNX	Production Models
ONNX	Full	TensorRT	Cross-framework Standard
Keras	Via TensorFlow	TensorRT	Lightweight Models
OpenCV AI	With CUDA/GPU	Uses GPU acceleration	Great for vision-based tasks

Deployment Workflow

- ➔ Train on PC/Cloud (TF/PyTorch)
- ➔ Export Model (SavedModel/ONNX)
- ➔ Optimize with TensorRT
- ➔ Deploy on Jetson Orin Nano

Run Pre-trained Models Efficiently

- **Deployment Workflow**
- You **train** on a powerful system (desktop/server) and **deploy optimized inference** on Jetson Orin Nano.

```
graph LR
A[Train on PC/Cloud (TF/PyTorch)] --> B[Export
Model (SavedModel/ONNX)]
B --> C[Optimize with TensorRT]
C --> D[Deploy on Jetson Orin Nano]
```

Optimized Inference Performance

- You can accelerate inference significantly using NVIDIA's **TensorRT**:
- **Convert PyTorch models**:

```
import torch
from torch2trt import torch2trt
model_trt = torch2trt(model, [input_data])
```

- **Convert TensorFlow models**:

```
from tensorflow.python.compiler.tensorrt import trt_convert as trt
converter = trt.TrtGraphConverterV2(...)
converter.convert()
```

Pretrained Model Support (Examples)

Task	Model	Performance (Orin Nano 8GB)
Image Classification	ResNet-50 (ONNX)	~100 FPS
Object Detection	YOLOv5s (PyTorch → ONNX)	~70 FPS
Face Recognition	MobileFaceNet (Keras)	~120 FPS
Segmentation	DeepLabV3+ (TF)	~30–40 FPS
Pose Estimation	OpenPose Lite (ONNX)	~25–35 FPS

Performance Benchmarks

Model	AI Perf (INT8)	ResNet-50 FPS
Orin Nano 4GB	20 TOPS	~50 FPS
Orin Nano 8GB	40 TOPS	~100 FPS

Jetson AI Tools for Frameworks

- **JetPack SDK** (Includes CUDA, cuDNN, TensorRT, DeepStream, etc.)
- **NGC Catalog** – Access to NVIDIA-optimized containers and models
- **Jetson Inference** – Prebuilt PyTorch-based examples:
  - imagenet.py, detectnet.py, etc.
- **TensorRT APIs** – C++/Python API for deep model acceleration

Sample Use-Case: Object Detection with YOLOv5

1. Train on desktop or download from Ultralytics

2. Convert to ONNX:

```
python export.py --weights yolov5s.pt --include onnx
```

3. Optimize using TensorRT on Jetson

4. Run at **70+ FPS** on Orin Nano 8GB

Summary

- Train on desktop/cloud, export, and optimize
- Jetson Orin Nano offers up to 40 TOPS performance
- Supports PyTorch, TensorFlow, ONNX with TensorRT
- Ideal for edge AI in robotics, vision, IoT