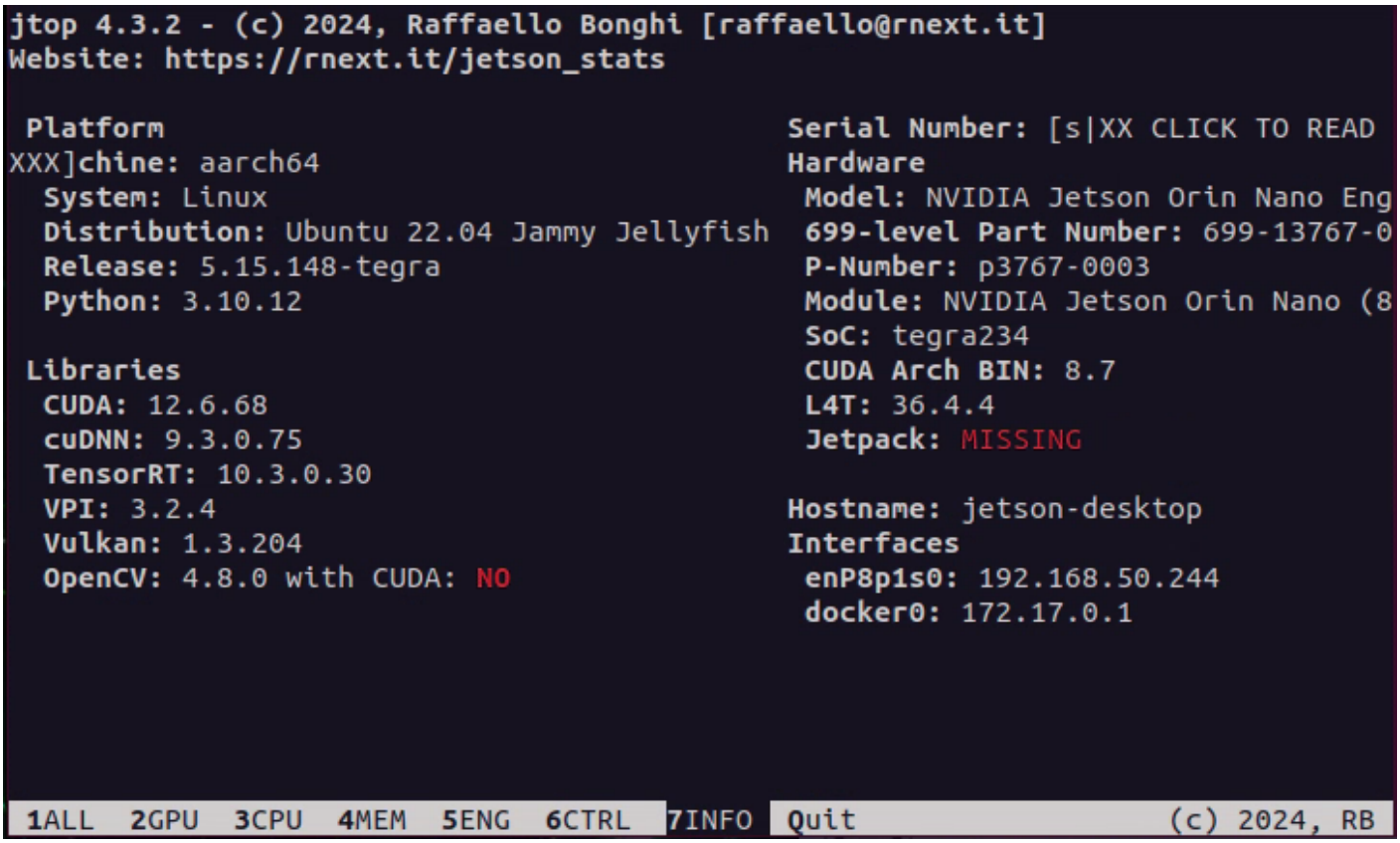
JETSON ORIN ??AI ????

□□□□ Jetson□□□□□□□□

- □□□□ opencv with cuda
- □□□□ pytorch□ torchvision
- □□ Jetson-container□□ Comfy-UI

????opencv with cuda

> JetPack□□□ opencv□□□□ cuda□□□□□□□□□□



1. ??????

```
#!/bin/bash
# opencv_install.sh
# Modified from
https://github.com/AastaNV/JEP/blob/master/script/install_opencv4.10.0_Jetpack6.1.sh

version="4.10.0"
folder="workspace"
remove_old=""

set -e

# Parse command-line arguments
for arg in "$@"; do
```

```
    case $arg in
        --version=*)
            version="${arg#*=}"
            ;;
        --folder=*)
            folder="${arg#*=}"
            ;;
        --remove-old=*)
            remove_old="${arg#*=}"
            ;;
        --help|-h)
            echo "Usage: $0 [--version=4.x.x] [--folder=dir] [--remove-old=yes/no]"
            exit 0
            ;;
        *)
            echo "Unknown parameter: $arg"
            echo "Usage: $0 [--version=4.x.x] [--folder=dir] [--remove-old=yes/no]"
            exit 1
            ;;
    esac
done

# Create installation directory if it doesn't exist
if [ ! -d "$folder" ]; then
    echo "Creating directory: $folder"
    mkdir -p "$folder"
fi
cd "$folder" || exit

# Old OpenCV removal logic
if [ -z "$remove_old" ]; then
    read -rp "Do you want to remove system-installed OpenCV? (yes/no): " remove_old
fi

case "$remove_old" in
    [yY] | [yY][eE][sS])
        echo "** Removing system OpenCV packages"
        sudo apt -y purge *libopencv*
        sudo apt -y autoremove
        ;;
```

```
    *)
        echo "** Skipping system OpenCV removal"
        ;;
esac

echo "-----------------------------------"
echo "** Installing dependencies (1/4)"
echo "-----------------------------------"
sudo apt-get update
sudo apt-get install -y build-essential cmake git libgtk2.0-dev pkg-config \
    libavcodec-dev libavformat-dev libswscale-dev libgstreamer1.0-dev \
    libgstreamer-plugins-base1.0-dev python3-dev python3-numpy libtbb2 \
    libtbb-dev libjpeg-dev libpng-dev libtiff-dev libv4l-dev v4l-utils qv4l2 curl

# Verify essential dependencies installed
for dep in g++ cmake git pkg-config; do
    if ! command -v "$dep" > /dev/null; then
        echo "Error: $dep installation failed"
        exit 1
    fi
done

echo "-----------------------------------"
echo "** Downloading OpenCV ${version} (2/4)"
echo "-----------------------------------"

# Check if source files already exist
download_opencv=false
download_contrib=false

if [ ! -f "opencv-${version}.zip" ]; then
    echo "Downloading opencv-${version}.zip"
    wget -O opencv-${version}.zip https://github.com/opencv/opencv/archive/${version}.zip || {
        echo "Download failed! Check your internet connection or verify the version exists"
        exit 1
    }
    download_opencv=true
else
    echo "opencv-${version}.zip exists, skipping download"
fi
```

```
if [ ! -f "opencv_contrib-${version}.zip" ]; then
    echo "Downloading opencv_contrib-${version}.zip"
    wget -O opencv_contrib-${version}.zip
https://github.com/opencv/opencv_contrib/archive/${version}.zip || {
        echo "Download failed! Check your internet connection or verify the version exists"
        exit 1
    }
    download_contrib=true
else
    echo "opencv_contrib-${version}.zip exists, skipping download"
fi

# Unpack source files
if [ ! -d "opencv-${version}" ] || $download_opencv; then
    if [ -d "opencv-${version}" ]; then
        echo "Removing existing opencv-${version} directory"
        rm -rf "opencv-${version}"
    fi
    echo "Unpacking opencv-${version}.zip"
    unzip -q opencv-${version}.zip || {
        echo "Extraction failed! File may be corrupt"
        exit 1
    }
fi

if [ ! -d "opencv_contrib-${version}" ] || $download_contrib; then
    if [ -d "opencv_contrib-${version}" ]; then
        echo "Removing existing opencv_contrib-${version} directory"
        rm -rf "opencv_contrib-${version}"
    fi
    echo "Unpacking opencv_contrib-${version}.zip"
    unzip -q opencv_contrib-${version}.zip || {
        echo "Extraction failed! File may be corrupt"
        exit 1
    }
fi

# Clean up zip files after successful extraction
if [ $? -eq 0 ]; then
```

```
    rm -f opencv-${version}.zip opencv_contrib-${version}.zip
fi

cd opencv-${version} || exit

echo "-----------------------------------"
echo "** Building OpenCV ${version} (3/4)"
echo "-----------------------------------"
mkdir -p release
cd release

# Auto-detect CUDA architecture
cuda_arch=""
if command -v nvidia-smi &> /dev/null; then
    gpu_name=$(nvidia-smi --query-gpu=name --format=csv,noheader | head -n1)
    if [[ $gpu_name == *"Orin"* ]] || [[ $gpu_name == *"Jetson"* ]]; then
        cuda_arch="8.7"
    elif [[ $gpu_name == *"A100"* ]]; then
        cuda_arch="8.0"
    fi
fi

cmake_cmd="cmake -D WITH_CUDA=ON -D WITH_CUDNN=ON -D OPENCV_GENERATE_PKGCONFIG=ON "
cmake_cmd+="-D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-${version}/modules "
cmake_cmd+="-D WITH_GSTREAMER=ON -D WITH_LIBV4L=ON -D BUILD_opencv_python3=ON "
cmake_cmd+="-D BUILD_TESTS=OFF -D BUILD_PERF_TESTS=OFF -D BUILD_EXAMPLES=OFF "
cmake_cmd+="-D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local "

# Add CUDA architecture if detected
if [ -n "$cuda_arch" ]; then
    echo "Detected NVIDIA GPU: ${gpu_name}, using CUDA_ARCH_BIN=${cuda_arch}"
    cmake_cmd+="-D CUDA_ARCH_BIN=${cuda_arch} -D CUDA_ARCH_PTX=\"\" "
else
    echo "No supported GPU detected, skipping CUDA architecture flags"
fi

# Execute CMake configuration
echo "CMake command: $cmake_cmd .."
$cmake_cmd .. || {
    echo "CMake configuration failed"
```

```
    exit 1
}

# Parallel build (leave one core for system stability)
cpu_cores=$(($(nproc) - 1))
[ $cpu_cores -lt 1 ] && cpu_cores=1
echo "Building with ${cpu_cores} CPU cores"
make -j${cpu_cores} || {
    echo "Compilation failed"
    exit 1
}

echo "-----------------------------------"
echo "** Installing OpenCV ${version} (4/4)"
echo "-----------------------------------"
sudo make install || {
    echo "Installation failed"
    exit 1
}

# Add environment variables to .bashrc (only if not already present)
bashrc=~/.bashrc
env_lines=(
    "export LD_LIBRARY_PATH=/usr/local/lib:\$LD_LIBRARY_PATH"
    "export PYTHONPATH=/usr/local/lib/python3.10/site-packages/:\$PYTHONPATH"
)

for line in "${env_lines[@]}"; do
    if ! grep -Fxq "$line" "$bashrc"; then
        echo "Adding to .bashrc: $line"
        echo "$line" >> "$bashrc"
    else
        echo "Environment variable already exists: $line"
    fi
done

source ~/.bashrc

echo "** OpenCV ${version} installation completed"
echo "Verification commands:"
```

```
echo "  pkg-config --modversion opencv4"
echo "  python3 -c 'import cv2; print(cv2.__version__)'"

echo "** Installation successful!"
```

2. ????

2.1 □□□□ opencv

```
sudo apt-get purge libopencv*
sudo apt autoremove
sudo apt-get update
```

2.2 □□□□□□□

```
sudo apt-get update
sudo apt-get install -y build-essential cmake git libgtk2.0-dev pkg-config libavcodec-dev
libavformat-dev libswscale-dev
sudo apt-get install -y libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev python3.10-dev
python3-numpy
sudo apt-get install -y libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libv4l-dev v4l-
utils qv4l2
sudo apt-get install -y curl
```

2.3 □□ opencv□□ (□ 4.10.0□□□)

```
version=4.10.0
wget -O "opencv-${version}.zip" "https://github.com/opencv/opencv/archive/${version}.zip"
wget -O "opencv_contrib-${version}.zip"
"https://github.com/opencv/opencv_contrib/archive/${version}.zip"
unzip "opencv-${version}.zip"
unzip "opencv_contrib-${version}.zip"
rm "opencv-${version}.zip" "opencv_contrib-${version}.zip"
cd "opencv-${version}/"
```

2.4 □□□□

> □□□□□□□□□□□□□

>

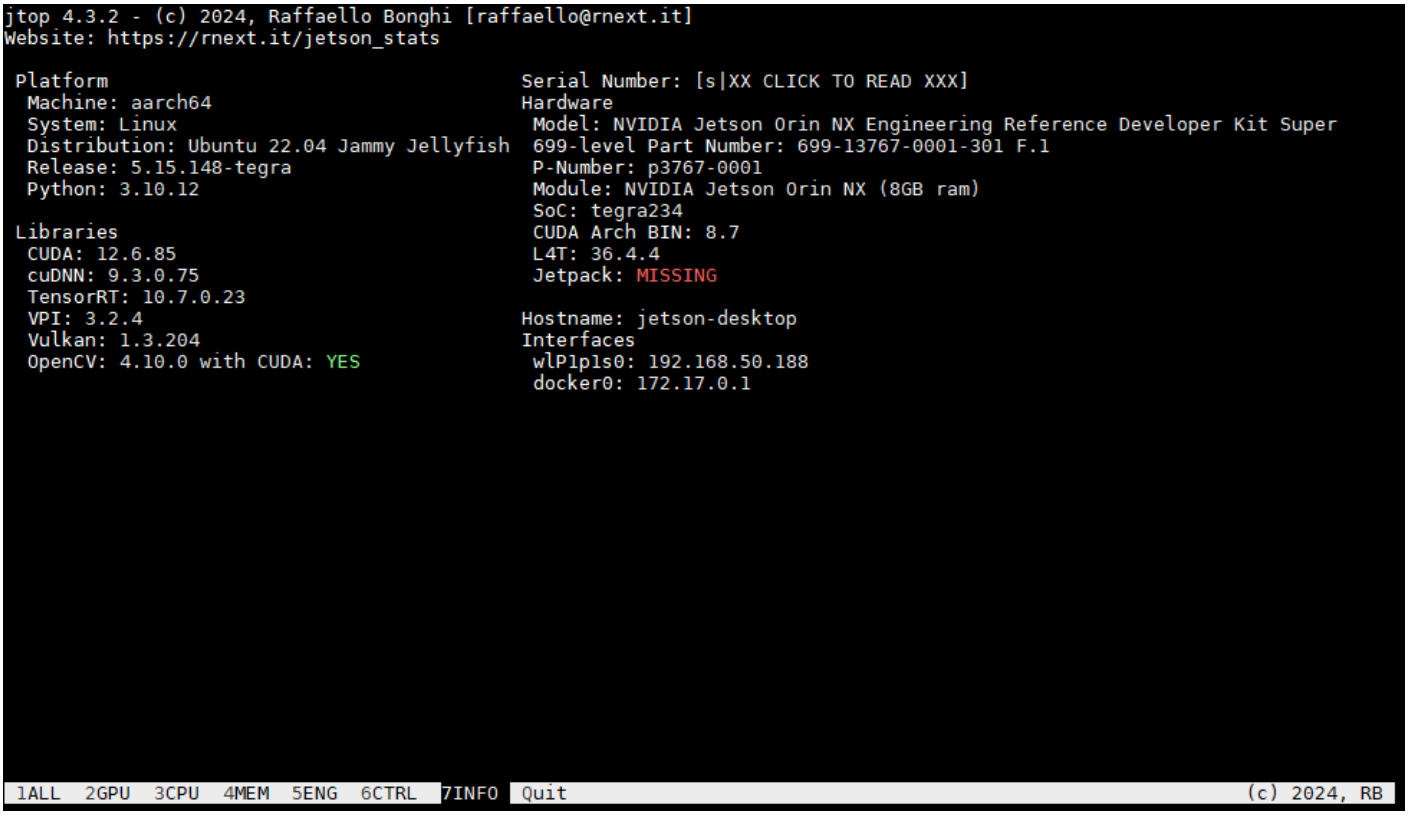
□□□□□□□□□□□□□□□□□□□□□□□□□□

```
mkdir build
cd build/
cmake -D WITH_CUDA=ON -D WITH_CUDNN=ON -D CUDA_ARCH_BIN="8.7" -D CUDA_ARCH_PTX="" -D
OPENCV_GENERATE_PKGCONFIG=ON -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-
${version}/modules -D WITH_GSTREAMER=ON -D WITH_LIBV4L=ON -D BUILD_opencv_python3=ON -D
BUILD_TESTS=OFF -D BUILD_PERF_TESTS=OFF -D BUILD_EXAMPLES=OFF -D CMAKE_BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/usr/local ..
make -j$(nproc)
```

2.5 □□

```
sudo make install
echo 'export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH' >> ~/.bashrc
echo 'export PYTHONPATH=/usr/local/lib/python3.10/site-packages/:$PYTHONPATH' >> ~/.bashrc
source ~/.bashrc
```

??????



```
#--test_cuda.cpp

#include <opencv2/opencv.hpp>
#include <opencv2/core/cuda.hpp>
#include <opencv2/cudaarithm.hpp>
#include <iostream>
#include <chrono>

// CPU □□□□
void cpu_matrix_mult(cv::Mat& a, cv::Mat& b, cv::Mat& result) {
    for (int i = 0; i < 50; i++) {
        result = a * b;
    }
}

// GPU □□□□
void gpu_matrix_mult(cv::cuda::GpuMat& d_a, cv::cuda::GpuMat& d_b, cv::cuda::GpuMat& d_result)
{
    cv::cuda::Stream stream;

    for (int i = 0; i < 50; i++) {
        cv::cuda::gemm(d_a, d_b, 1.0, cv::cuda::GpuMat(), 0, d_result, 0, stream);
        stream.waitForCompletion();
    }
}

int main() {
    try {
        std::cout << "--- OpenCV CUDA Matrix Multiplication Test ---\n";

        // □□□□ 1000x1000 □□□□□
        cv::Mat mat_a(1000, 1000, CV_32FC1);
        cv::Mat mat_b(1000, 1000, CV_32FC1);
        cv::randu(mat_a, 0.0f, 1.0f);
        cv::randu(mat_b, 0.0f, 1.0f);

        cv::Mat cpu_result;

        // CPU □□
        auto start_cpu = std::chrono::high_resolution_clock::now();
```

```
        cpu_matrix_mult(mat_a, mat_b, cpu_result);
        auto end_cpu = std::chrono::high_resolution_clock::now();
        double cpu_time = std::chrono::duration_cast<std::chrono::milliseconds>(end_cpu -
start_cpu).count();

        // GPU □□
        cv::cuda::GpuMat d_mat_a, d_mat_b, d_result;
        d_mat_a.upload(mat_a);
        d_mat_b.upload(mat_b);

        auto start_gpu = std::chrono::high_resolution_clock::now();
        gpu_matrix_mult(d_mat_a, d_mat_b, d_result);
        auto end_gpu = std::chrono::high_resolution_clock::now();
        double gpu_time = std::chrono::duration_cast<std::chrono::milliseconds>(end_gpu -
start_gpu).count();

        // □□□□□□□□□
        cv::Mat gpu_result;
        d_result.download(gpu_result);

        // □□□□(□□□□)
        double diff = cv::norm(cpu_result, gpu_result, cv::NORM_L2);
        std::cout << "Result difference: " << diff << "\n";

        std::cout << "Performance Results:\n"
                  << " - CPU time: " << cpu_time << " ms\n"
                  << " - GPU time: " << gpu_time << " ms\n"
                  << " - Speedup: " << cpu_time / gpu_time << "x\n";

        std::cout << "\n CUDA matrix multiplication test completed\n";
        return 0;

    } catch (const cv::Exception& e) {
        std::cerr << "OpenCV Error (" << e.err << "): " << e.what() << "\n";
        return -1;
    } catch (const std::exception& e) {
        std::cerr << "Standard Error: " << e.what() << "\n";
        return -2;
    }
}
```

□□□□

```
jetson@jetson-desktop:~/work$ g++ test_cuda.cpp -o test_cuda `pkg-config --cflags --libs opencv4`
jetson@jetson-desktop:~/work$ ./test_cuda
--- OpenCV CUDA Performance Test ---
Performance Results:
 - CPU time: 2451 ms
 - GPU time: 918 ms
 - Speedup: 2.66993x

□ CUDA performance test completed
```

????pytorch?torchvision

PyTorch □ Python □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Python □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ API □□□□□□ GPU □□

NVIDIA □ Jetson □□□□□□□□□□□□□□□□□□□□□□□□□□□□

PyTorch Version	NVIDIA Framework Container	NVIDIA Framework Wheel	JetPack Version
2.8.0a0+5228986c39	25.06	-	6.2
2.8.0a0+5228986c39	25.05	-	6.2
2.7.0a0+79aa17489c	25.04	-	6.2
2.7.0a0+7c8ec84dab	25.03	-	6.2
2.7.0a0+6c54963f75	25.02	-	6.2
2.6.0a0+ecf3bae40a	25.01	-	6.1
2.6.0a0+df5bbc09d1	24.12	-	6.1
2.6.0a0+df5bbc0	24.11	-	6.1
2.5.0a0+e000cf0ad9	24.10	-	6.1
2.5.0a0+b465a5843b	24.09	24.09	6.1
2.5.0a0+872d972e41	24.08	-	6.0
2.4.0a0+3bcc3cddb5	24.07	24.07	6.0
2.4.0a0+f70bd71a48	24.06	24.06	6.0
2.4.0a0+07cecf4168	24.05	24.05	6.0
2.3.0a0+6ddf5cf85e	24.04	24.04	6.0 Developer Preview
2.3.0a0+40ec155e58	24.03	24.03	
2.3.0a0+ebedce2	24.02	24.02	
2.2.0a0+81ea7a4	23.12, 24.01	23.12, 24.01	
2.2.0a0+6a974bec	23.11	23.11	

PyTorch Version	NVIDIA Framework Container	NVIDIA Framework Wheel	JetPack Version
2.1.0a		23.06	5.1.x
2.0.0		23.05	
2.0.0a0+fe05266f		23.04	
2.0.0a0+8aa34602		23.03	
1.14.0a0+44dac51c		23.02, 23.01	
1.13.0a0+936e930		22.11	5.0.2
1.13.0a0+d0d6b1f		22.09, 22,10	
1.13.0a0+08820cb	22.07	22.07	
1.13.0a0+340c412	22.06	22.06	5.0.1
1.12.0a0+8a1a93a9	22.05	22.05	5.0
1.12.0a0+bd13bc66		22.04	
1.12.0a0+2c916ef		22.03	
1.11.0a0+bfe5ad28		22.01	4.6.1

□□□□ **JetPack6.2.1 cuda12.6** □□□

1.??torch???

1.1?????torch , torchvison

```
wget https://pypi.jetson-ai-lab.io/jp6/cu126/+f/62a/1beee9f2f1470/torch-2.8.0-cp310-cp310-linux_aarch64.whl
wget https://pypi.jetson-ai-lab.io/jp6/cu126/+f/907/c4c1933789645/torchvision-0.23.0-cp310-cp310-linux_aarch64.whl
pip install torch-2.8.0-cp310-cp310-linux_aarch64.whl torchvision-0.23.0-cp310-cp310-linux_aarch64.whl -i https://pypi.tuna.tsinghua.edu.cn/simple
```

1.2 ????????

□□ python□□□□□□□

```
jetson@jetson-desktop:~$ python
Python 3.10.16 (main, Dec 11 2024, 16:18:56) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> import torch
>>> print(torch.__version__)
2.8.0
>>> print(torch.cuda.is_available())
True
```

2. ??YOLO11

YOLO

是一种实时目标检测算法，它将目标检测视为单阶段回归问题，通过将图像划分为网格并直接预测边界框与类别概率，实现高速且高精度的检测。 YOLO系列因开源易用、部署灵活，广泛应用于自动驾驶、安防监控、工业质检等领域。

2.1 ??miniconda

```
curl -L https://repo.anaconda.com/miniconda/Miniconda3-py310_25.3.1-1-Linux-aarch64.sh | bash
source ~/miniconda3/bin/activate
conda --version
```

2.2 conda??

```
conda config --add channels https://mirrors.ustc.edu.cn/anaconda/pkgs/main/
conda config --add channels https://mirrors.ustc.edu.cn/anaconda/pkgs/free/
conda config --add channels https://mirrors.ustc.edu.cn/anaconda/cloud/conda-forge/
conda config --add channels https://mirrors.ustc.edu.cn/anaconda/cloud/msys2/
conda config --set show_channel_urls yes
```

2.3??conda??

```
conda create -n jetson-ai python=3.10
```

2.4 ??conda??

```
conda activate jetson-ai
```

2.5 ??torch?torchvison

```
wget https://pypi.jetson-ai-lab.io/jp6/cu126/+f/62a/1beee9f2f1470/torch-2.8.0-cp310-cp310-linux_aarch64.whl
wget https://pypi.jetson-ai-lab.io/jp6/cu126/+f/907/c4c1933789645/torchvision-0.23.0-cp310-cp310-linux_aarch64.whl
pip install torch-2.8.0-cp310-cp310-linux_aarch64.whl torchvision-0.23.0-cp310-cp310-linux_aarch64.whl -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2.6 ??ultralytics

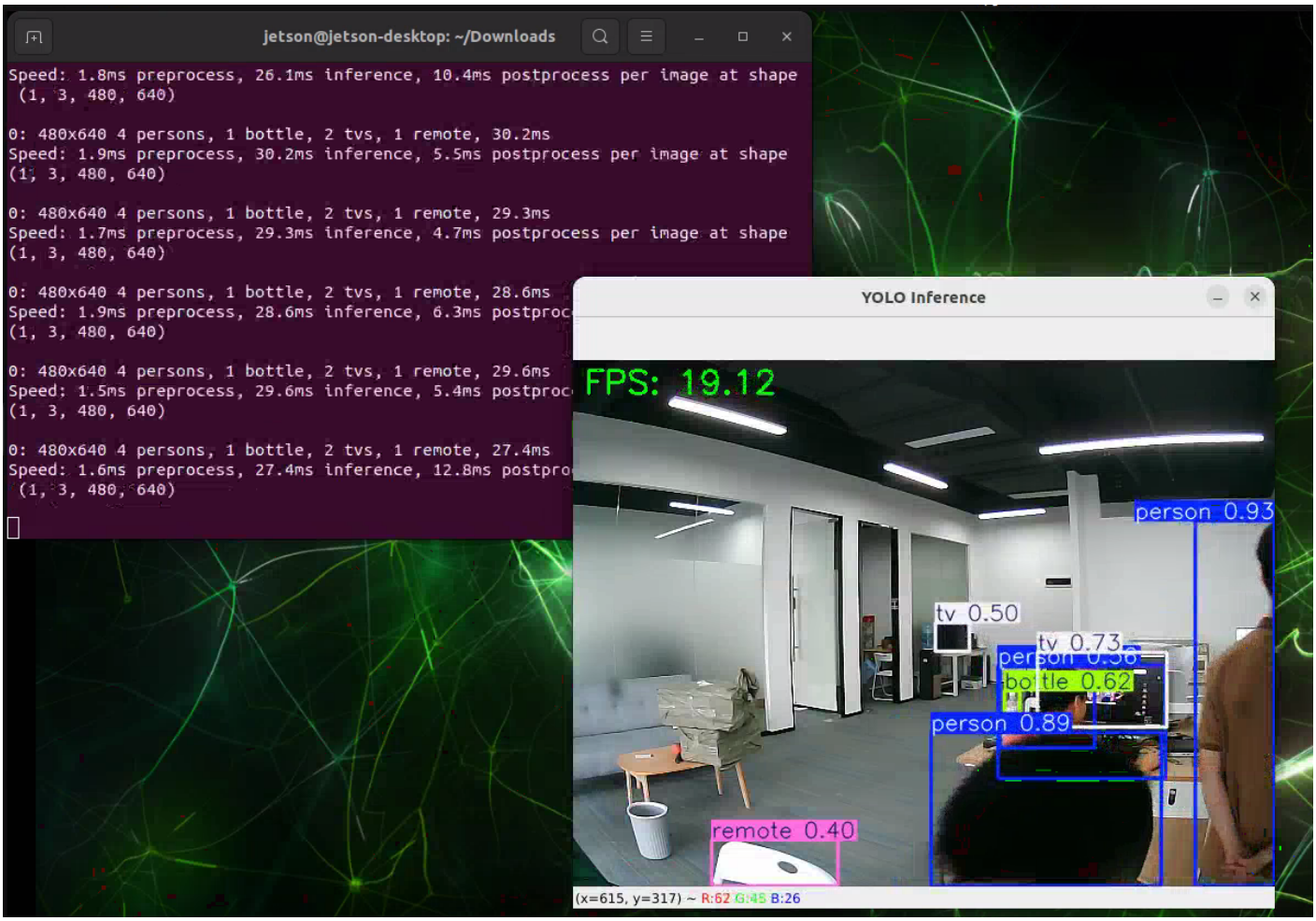
```
pip install ultralytics -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2.7 ???????????

□□□□□□□□□□□□□□□□□□□□□□

```
import cv2
import time
from ultralytics import YOLO
from ultralytics import YOLOWorld

# Load the YOLO model
model = YOLO("yolo11s.pt")

# Open the video file
video_path = 0
cap = cv2.VideoCapture(video_path)

# Loop through the video frames
while cap.isOpened():

    # Read a frame from the video
    success, frame = cap.read()
    start = time.time()
    if success:
        # Run YOLO inference on the frame
        results = model(frame)
        inf_time = time.time() - start
        # Visualize the results on the frame
        annotated_frame = results[0].plot()
        fps = 1.0 / inf_time if inf_time > 0 else 0
        # show FPS
        cv2.putText(annotated_frame, f"FPS: {fps:.2f}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,255,0), 2)
        cv2.imshow("YOLO Inference", annotated_frame)

        # Break the loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
```

```
    else:
        # Break the loop if the end of the video is reached
        break

# Release the video capture object and close the display window
cap.release()
cv2.destroyAllWindows()
```



□□□□□□ Ultralytics YOLO11 -Ultralytics YOLO □□

3. ??????Pytorch/torchvison

□□□□□□□□□□ pytorch□□□□□□□□□□□□□□□ whl □□□□□□□□□□□□□□□□□□□□

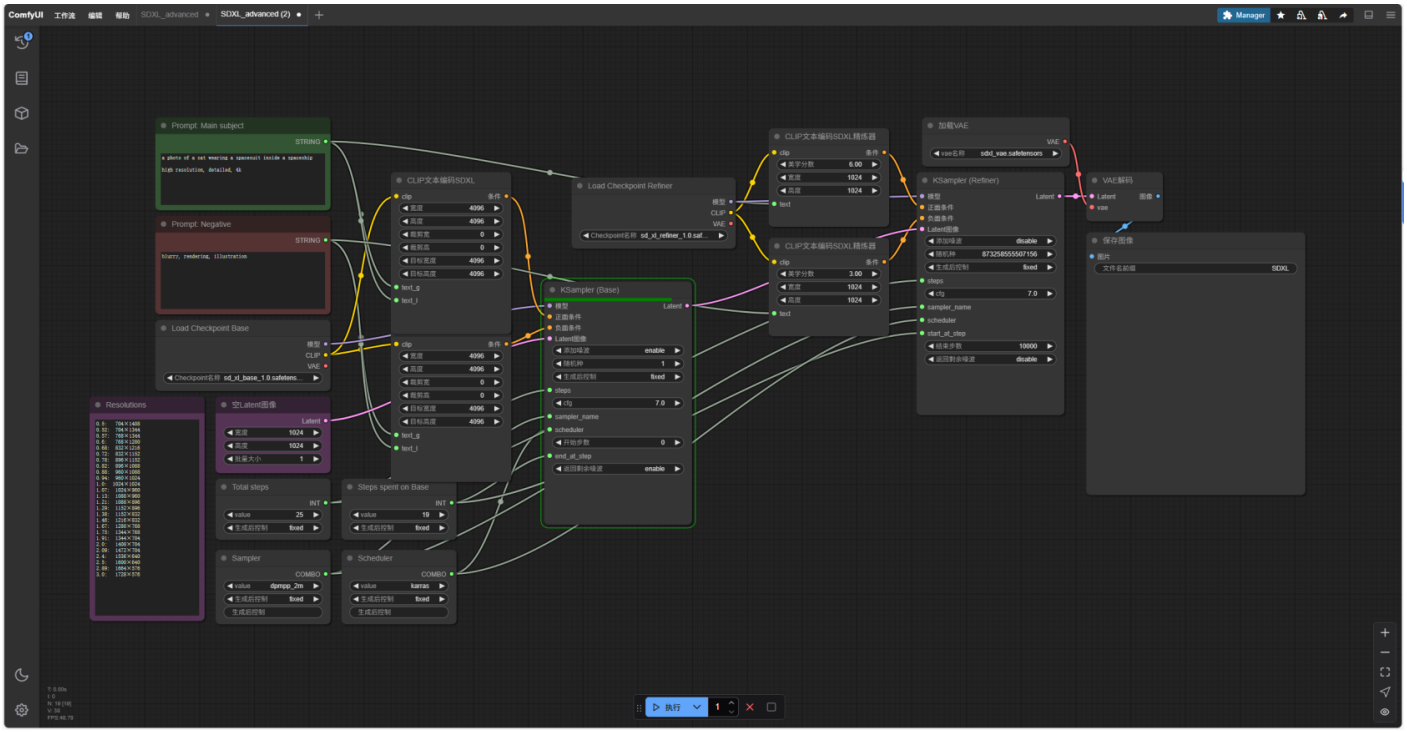
??Jetson-container??Comfy-UI

> [illegible] docker[illegible]

Jetson-container[illegible] NVIDIA[illegible] Jetson[illegible] Docker[illegible] CUDA[illegible] cuDNN[illegible] TensorRT [illegible] AI[illegible]

[illegible] **Jetson Orin NX 16GB[illegible] JetPack6.2.1 [illegible] Comfy-UI** [illegible] [illegible]

Comfy-UI [illegible] Stable Diffusion[illegible] LoRA[illegible] ControlNet[illegible]



1. [illegible] jetson-container[illegible]

```
git clone https://github.com/dusty-nv/jetson-containers.git
cd jetson-containersbash
jetson-containers/install.sh
```

2. [illegible] docker[illegible]

```
jetson-containers run dustynv/comfyui:r36.4.3
```

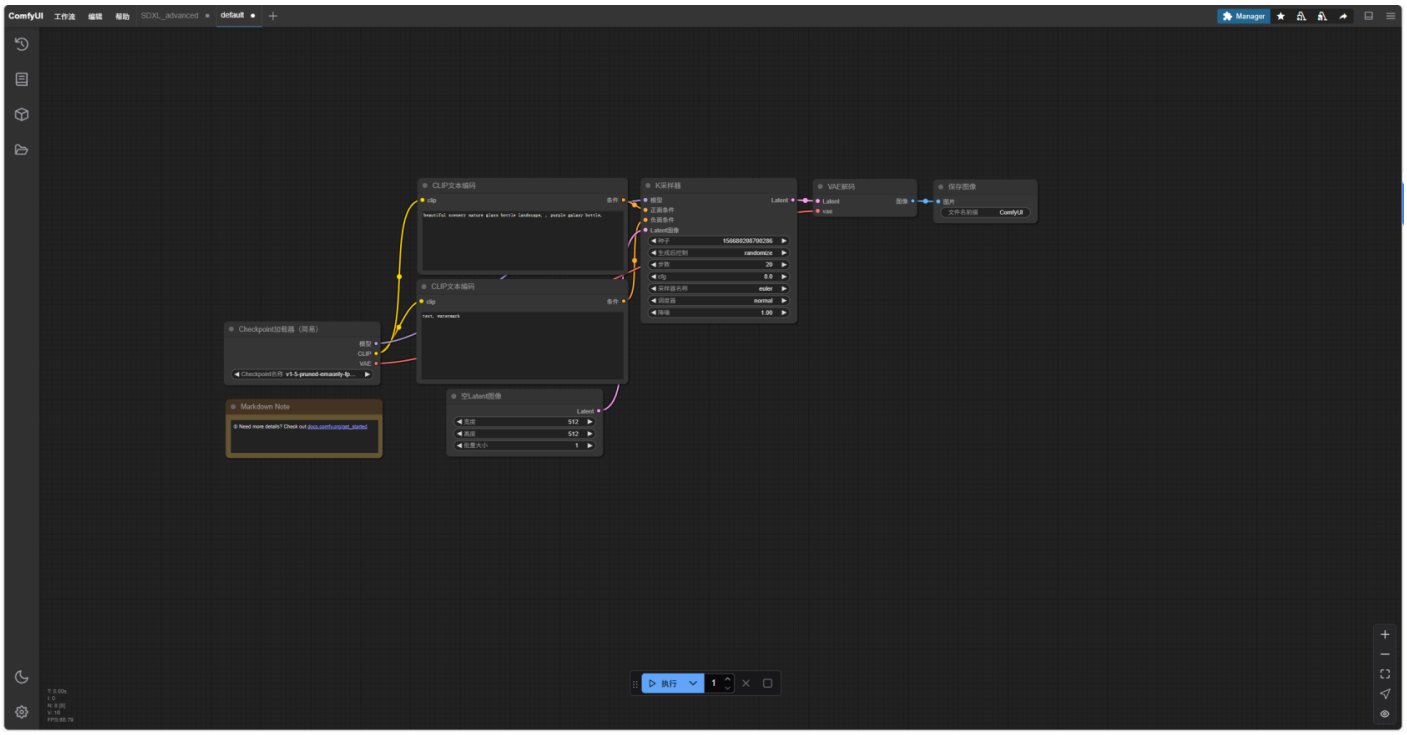
3. [illegible]

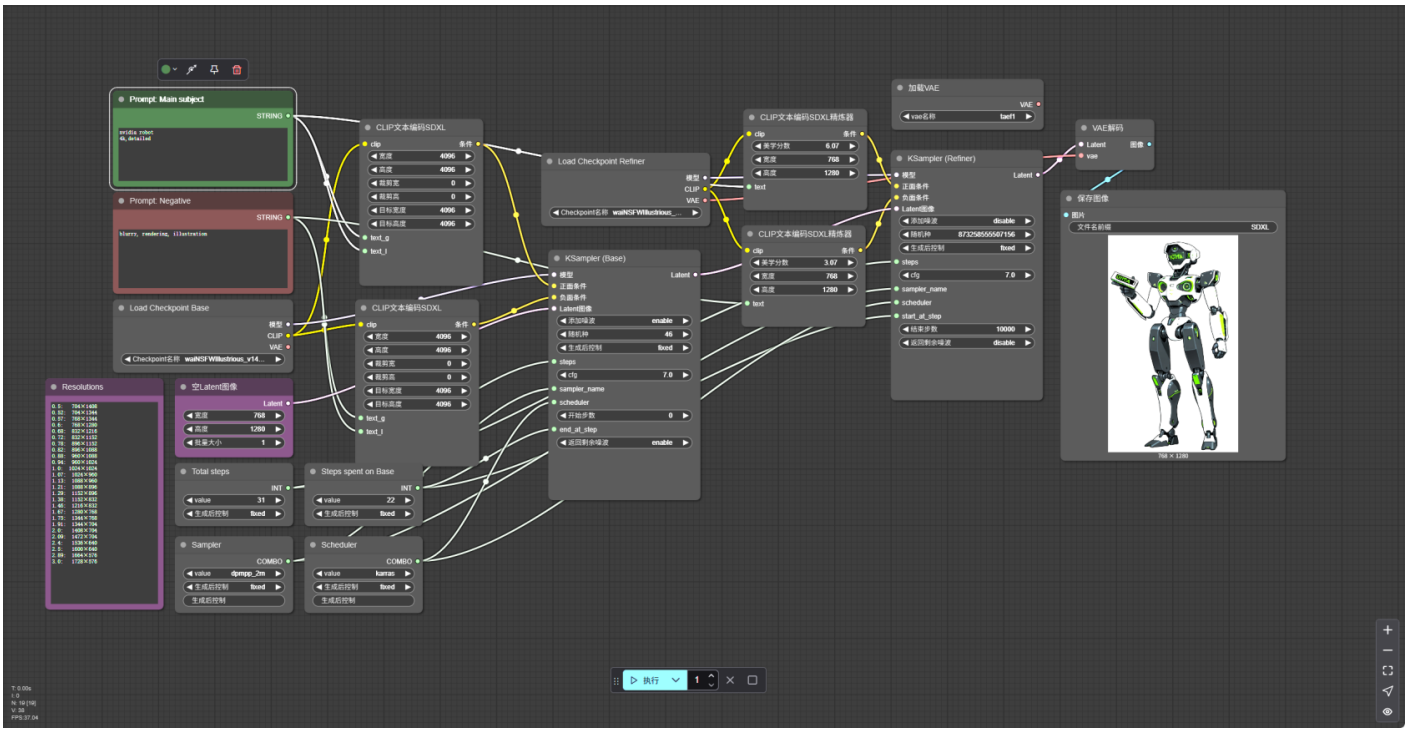
```
  File "<frozen importlib._bootstrap>", line 241, in _call_with_frames_removed
  File "/opt/ComfyUI/custom_nodes/ComfyUI-Crystools/__init__.py", line 18, in <module>
    from .nodes.image import CImagePreviewFromImage, CImageLoadWithMetadata, CImageGetResolution, CImagePrev
iewFromMetadata, \
  File "/opt/ComfyUI/custom_nodes/ComfyUI-Crystools/nodes/image.py", line 5, in <module>
    import piexif
ModuleNotFoundError: No module named 'piexif'

Cannot import /opt/ComfyUI/custom_nodes/ComfyUI-Crystools module for custom nodes: No module named 'piexif'
### Loading: ComfyUI-Manager (V3.30.4)
[ComfyUI-Manager] network_mode: public
### ComfyUI Version: v0.3.26-15-gbc219a64 | Released on '2025-03-11'
ComfyUI found: /opt/ComfyUI
'/opt/ComfyUI' added to sys.path

Import times for custom nodes:
   0.0 seconds: /opt/ComfyUI/custom_nodes/websocket_image_save.py
   0.0 seconds: /opt/ComfyUI/custom_nodes/comfyui-flux-accelerator
   0.1 seconds: /opt/ComfyUI/custom_nodes/ComfyUI-Manager
   0.1 seconds: /opt/ComfyUI/custom_nodes/ComfyUI-to-Python-Extension
   0.7 seconds (IMPORT FAILED): /opt/ComfyUI/custom_nodes/ComfyUI-Crystools

Starting server

To see the GUI go to: http://0.0.0.0:8188
[ComfyUI-Manager] default cache updated: https://raw.githubusercontent.com/ltdrdata/ComfyUI-Manager/main/alt
er-list.json
[ComfyUI-Manager] default cache updated: https://raw.githubusercontent.com/ltdrdata/ComfyUI-Manager/main/git
hub-stats.json
[ComfyUI-Manager] default cache updated: https://raw.githubusercontent.com/ltdrdata/ComfyUI-Manager/main/cus
tom-node-list.json
[ComfyUI-Manager] default cache updated: https://raw.githubusercontent.com/ltdrdata/ComfyUI-Manager/main/mod
el-list.json
[ComfyUI-Manager] default cache updated: https://raw.githubusercontent.com/ltdrdata/ComfyUI-Manager/main/ext
ension-node-map.json
FETCH ComfyRegistry Data: 5/94
FETCH ComfyRegistry Data: 10/94
FETCH ComfyRegistry Data: 15/94
FETCH ComfyRegistry Data: 20/94
FETCH ComfyRegistry Data: 25/94
FETCH ComfyRegistry Data: 30/94
```

4. □□□□□□□ GUI□□

□□ Jetson□□□□□□□□ http://0.0.0.0:8188

□□□□□□□□□□□□□□□ http://<jetson□□ IP>:8188





5. 设定映射路径，将开发板里下载好的模型映射到容器内部目录

```
jetson-containers run ~/models/:/opt/ComfyUI/models/checkpoints dustynv/comfyui:r36.4.3
```

6. 按照需求搭建工作流并调整参数生成图像

关于 ComfyUI的使用教程请参考其官网 ComfyUI | 用 AI生成视频、图像、音频