

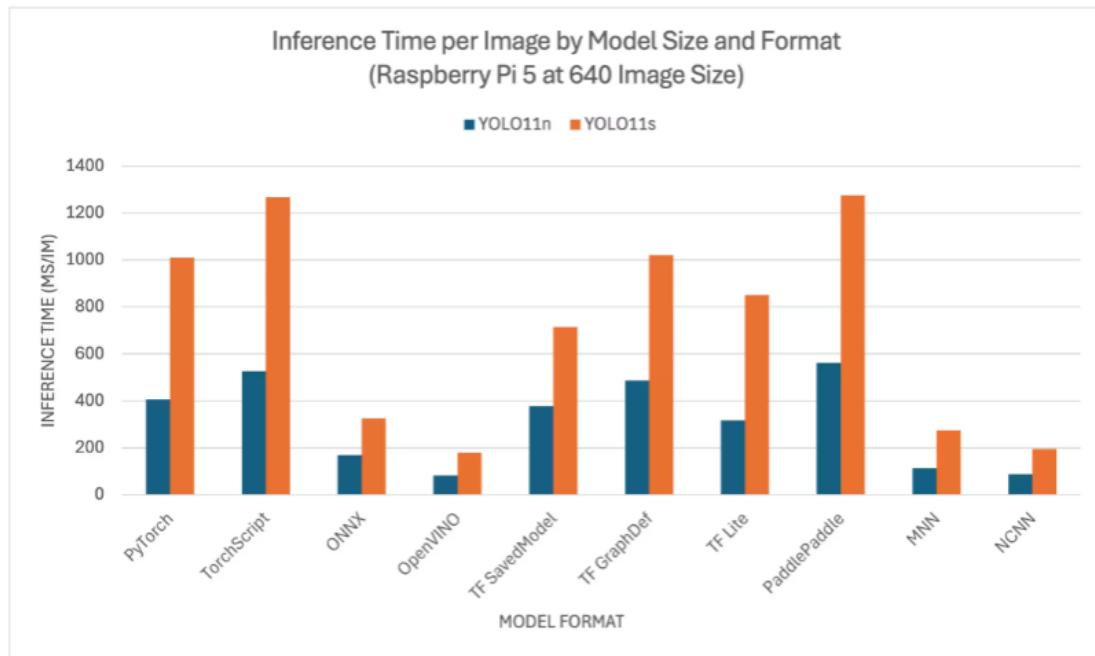
Model Conversion

Note: If using the docker container from the factory image, you don't need to rebuild the environment. The environment is already set up, you just need to enter the docker according to the previous tutorial and run the corresponding function commands to use it.

1. Raspberry Pi 5 YOLO11 (Benchmark)

YOLO11 benchmark data source from Ultralytics team, testing various model formats (data for reference only)

Official benchmark testing was only performed on YOLO11n and YOLO11s models, because other models are too large to run on Raspberry Pis and cannot provide good performance.



2. Model Conversion

The jetson nano mainboard docker cannot perform conversion because PyTorch/ultralytics dependencies are too new and CPU/GPU instruction sets are incompatible. You can first export TensorRT/ONNX on Raspberry Pi or PC, then run on Jetson Nano:

According to the test parameters provided by the Ultralytics team for different model formats, we can find that using TensorRT for inference provides the best performance!

The first time you use YOLO11's export mode, it will automatically install some dependencies, just wait for the automatic completion!

2.1 CLI: pt → onnx, pt → ncnn

Convert PyTorch format models to onnx and ncnn

```
cd ~/ultralytics/ultralytics/
```

Run the following commands in terminal:

```

yolo export model=yolo11n.pt format=onnx
# yolo export model=yolo11n-seg.pt format=onnx
# yolo export model=yolo11n-pose.pt format=onnx
# yolo export model=yolo11n-cls.pt format=onnx
# yolo export model=yolo11n-obb.pt format=onnx

yolo export model=yolo11n.pt format=ncnn
# yolo export model=yolo11n-seg.pt format=ncnn
# yolo export model=yolo11n-pose.pt format=ncnn
# yolo export model=yolo11n-cls.pt format=ncnn
# yolo export model=yolo11n-obb.pt format=ncnn

```

```

root@raspberrypi:~/ultralytics/ultralytics# yolo export model=yolo11n-seg.pt format=onnx
Ultralytics 8.3.154 Python-3.10.12 torch-2.1.2 CPU (Cortex-A76)
YOLO11n-seg summary (fused): 113 layers, 2,868,664 parameters, 0 gradients, 10.4 GFLOPs

PyTorch: starting from 'yolo11n-seg.pt' with input shape (1, 3, 640, 640) BCHW and output shape(s) ((1, 116, 8400), (1, 32, 160, 160)) (5.9 MB)

ONNX: starting export with onnx 1.17.0 opset 17...
WARNING △ ONNX: simplifier failure: cannot import name 'equal_valued' from 'sympy.core.numbers' (/usr/lib/python3/dist-packages/sympy/core/numbers.py)
ONNX: export success ✓ 2.7s, saved as 'yolo11n-seg.onnx' (11.2 MB)

Export complete (4.6s)
Results saved to /root/ultralytics/ultralytics
Predict:      yolo predict task=segment model=yolo11n-seg.onnx imgs=640
Validate:     yolo val task=segment model=yolo11n-seg.onnx imgs=640 data=/ultralytics/ultralytics/cfg/assets/coco.yaml
Visualize:    https://netron.app
Learn more at https://docs.ultralytics.com/modes/export
root@raspberrypi:~/ultralytics/ultralytics# ls

root@raspberrypi:~/ultralytics/ultralytics# ls
__init__.py  data      nn        trackers      yolo11n-cls.pt   yolo11n-seg.pt       yolo11n_ncnn_model
assets       engine    output    utils        yolo11n-obb.pt   yolo11n.onnx
cfg          hub      runs     videos      yolo11n-nose.pt  yolo11n.pt
core         models   solutions yahboom_demo  yolo11n-seg.onnx  yolo11n.torchscript
root@raspberrypi:~/ultralytics/ultralytics#

```

2.2 Python: pt → onnx → ncnn

Convert PyTorch format models to TensorRT: The conversion process will automatically generate ONNX models

```
cd ~/ultralytics/ultralytics/yahboom_demo
```

```
python3 model_pt_onnx_ncnn.py
```

```

File Edit Tabs Help
inputshape = [1,3,640,640]f32
#####
# pass_level0
inline module = torch.nn.modules.linear.Identity
inline module = ultralytics.nn.modules.block.Attention
inline module = ultralytics.nn.modules.block.Bottleneck
inline module = ultralytics.nn.modules.block.C2PSA
inline module = ultralytics.nn.modules.block.C3k
inline module = ultralytics.nn.modules.block.C3k2
inline module = ultralytics.nn.modules.block.DFL
inline module = ultralytics.nn.modules.block.PSABlock
inline module = ultralytics.nn.modules.block.SPPF
inline module = ultralytics.nn.modules.conv.Concat
inline module = ultralytics.nn.modules.conv.Conv
inline module = ultralytics.nn.modules.conv.DWConv
inline module = ultralytics.nn.modules.head.Detect
inline module = torch.nn.modules.linear.Identity
inline module = ultralytics.nn.modules.block.Attention
inline module = ultralytics.nn.modules.block.Bottleneck
inline module = ultralytics.nn.modules.block.C2PSA
inline module = ultralytics.nn.modules.block.C3k
inline module = ultralytics.nn.modules.block.C3k2
inline module = ultralytics.nn.modules.block.DFL
inline module = ultralytics.nn.modules.block.PSABlock
inline module = ultralytics.nn.modules.block.SPPF
inline module = ultralytics.nn.modules.conv.Concat
inline module = ultralytics.nn.modules.conv.Conv
inline module = ultralytics.nn.modules.conv.DWConv
inline module = ultralytics.nn.modules.head.Detect

-----
#####
# pass_level1
# pass_level2
# pass_level3
# pass_level4
# pass_level5
# pass_ncnn
NCNN: export success ✓ 2.7s, saved as '/root/ultralytics/ultralytics/yolo1in_ncnn_model' (10.2 MB)

Export complete (8.8s)
Results saved to /root/ultralytics/ultralytics
Predict:      yolo predict task=detect model=/root/ultralytics/ultralytics/yolo1in_ncnn_model imgsz=640
Validate:     yolo val task=detect model=/root/ultralytics/ultralytics/yolo1in_ncnn_model imgsz=640 data=/usr/src/ultralytics/ultralytics/cfg/datasets/coco.yaml
Visualize:   https://netron.app
root@raspberrypi:~/ultralytics/ultralytics/yahboom_demo# 

```

Example code:

```

from ultralytics import YOLO

# Load a YOLOv1n PyTorch model
model = YOLO("/root/ultralytics/ultralytics/yolo1in.pt")
# model = YOLO("/root/ultralytics/ultralytics/yolo1in-seg.pt")
# model = YOLO("/root/ultralytics/ultralytics/yolo1in-pose.pt")
# model = YOLO("/root/ultralytics/ultralytics/yolo1in-cls.pt")
# model = YOLO("/root/ultralytics/ultralytics/yolo1in-obb.pt")

# Export the model to ONNX format
model.export(format="onnx") # This will create 'yolo1in.onnx' in the same directory

# Export the model to NCNN format
model.export(format="ncnn") # creates 'yolo1in_ncnn_model'

```

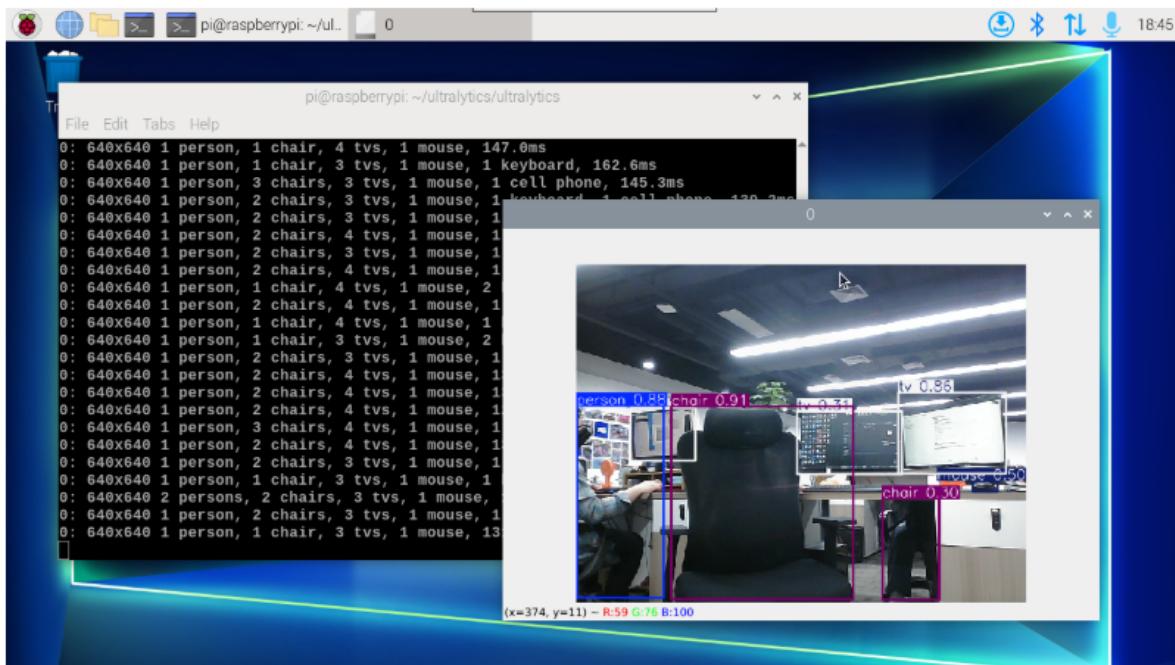
3. Model Prediction

CLI Usage

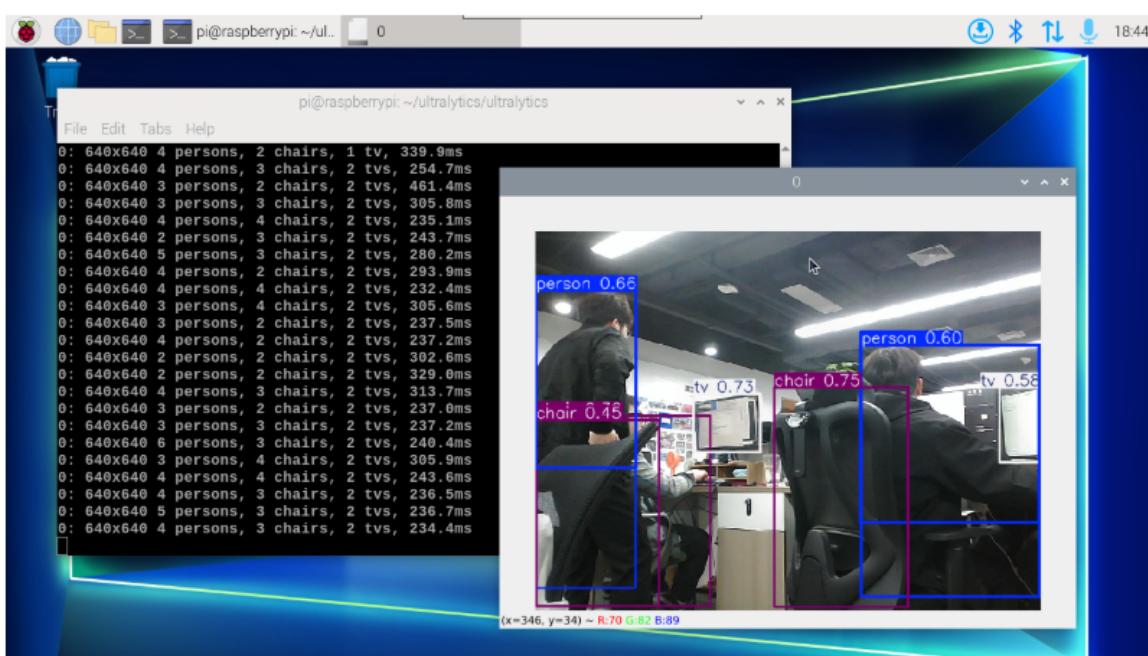
CLI currently only supports calling USB cameras!

```
cd ~/ultralytics/ultralytics/
```

```
yolo predict model=yolol1n.onnx source=0 save=False show
```



```
yolo predict model=yolol1n_ncnn_model source=0 save=False show
```



References

<https://docs.ultralytics.com/guides/nvidia-pi/>

<https://docs.ultralytics.com/integrations/tensorrt/>