Model prediction

1. Best performance mode

Enable Jetson Clocks

Enabling Jetson Clocks will ensure that all CPU and GPU cores run at maximum frequency:

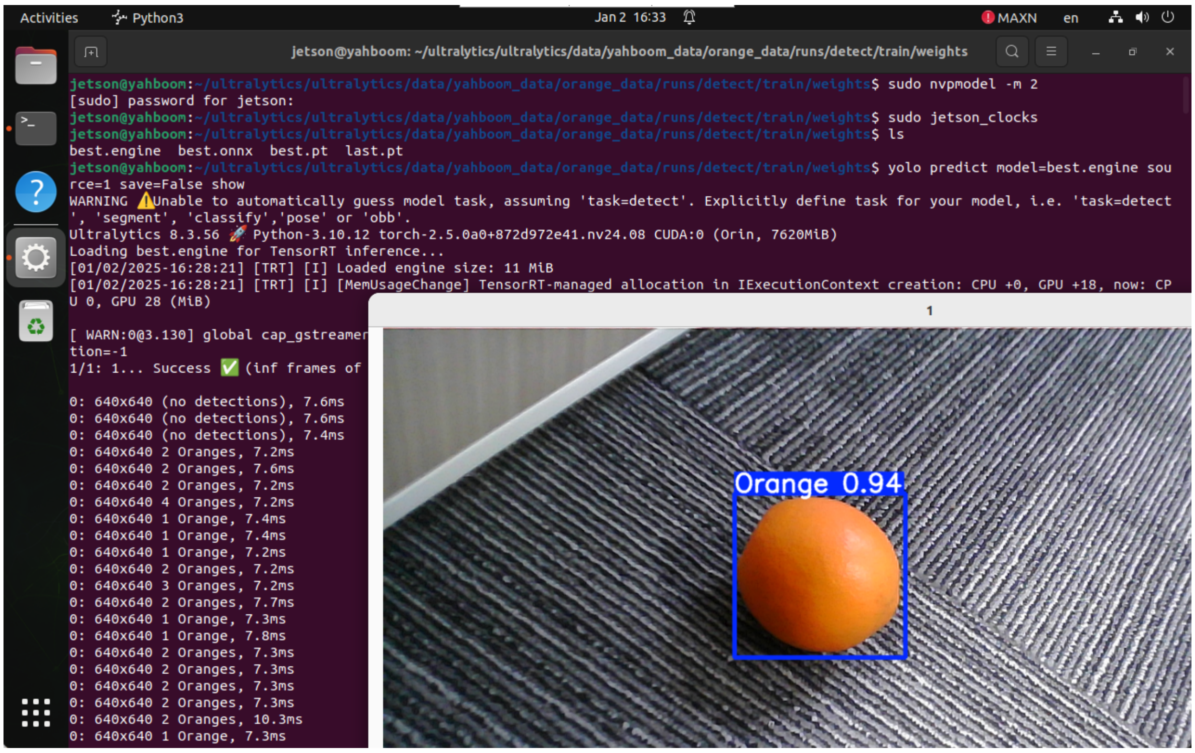
```
sudo jetson_clocks
```

2. Model prediction

2.1. CLI usage

CLI currently only supports calling USB cameras. Nuwa camera users can directly modify the previous python code to call onnx and engine models!

```
yolo predict model=best.engine source=0 save=False show # For multiple cameras, change the number after source.
```



Activities
Python3
Jan 2 16:33
MAXN
en
jetson@yahboom: ~/ultralytics/ultralytics/data/yahboom_data/orange_data/runs/detect/train/weights
1
Orange 0.94

```
jetson@yahboom:~/ultralytics/ultralytics/data/yahboom_data/orange_data/runs/detect/train/weights$ sudo nvpmodel -m 2
[sudo] password for jetson:
jetson@yahboom:~/ultralytics/ultralytics/data/yahboom_data/orange_data/runs/detect/train/weights$ sudo jetson_clocks
jetson@yahboom:~/ultralytics/ultralytics/data/yahboom_data/orange_data/runs/detect/train/weights$ ls
best.engine  best.onnx  best.pt  last.pt
jetson@yahboom:~/ultralytics/ultralytics/data/yahboom_data/orange_data/runs/detect/train/weights$ yolo predict model=best.engine sou
rce=1 save=False show
WARNING ⚠️Unable to automatically guess model task, assuming 'task=detect'. Explicitly define task for your model, i.e. 'task=detect
', 'segment', 'classify','pose' or 'obb'.
Ultralytics 8.3.56 🚀 Python-3.10.12 torch-2.5.0a0+872d972e41.nv24.08 CUDA:0 (Orin, 7620MiB)
Loading best.engine for TensorRT inference...
[01/02/2025-16:28:21] [TRT] [I] Loaded engine size: 11 MiB
[01/02/2025-16:28:21] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in IExecutionContext creation: CPU +0, GPU +18, now: CP
U 0, GPU 28 (MiB)

[ WARN:0@3.130] global cap_gstreamer
tion=-1
1/1: 1... Success ✅ (inf frames of

0: 640x640 (no detections), 7.6ms
0: 640x640 (no detections), 7.6ms
0: 640x640 (no detections), 7.4ms
0: 640x640 2 Oranges, 7.2ms
0: 640x640 2 Oranges, 7.6ms
0: 640x640 2 Oranges, 7.2ms
0: 640x640 4 Oranges, 7.2ms
0: 640x640 1 Orange, 7.4ms
0: 640x640 1 Orange, 7.4ms
0: 640x640 1 Orange, 7.2ms
0: 640x640 2 Oranges, 7.2ms
0: 640x640 3 Oranges, 7.2ms
0: 640x640 2 Oranges, 7.7ms
0: 640x640 1 Orange, 7.3ms
0: 640x640 1 Orange, 7.8ms
0: 640x640 2 Oranges, 7.3ms
0: 640x640 2 Oranges, 7.3ms
0: 640x640 2 Oranges, 7.3ms
0: 640x640 2 Oranges, 7.3ms
0: 640x640 2 Oranges, 10.3ms
0: 640x640 1 Orange, 7.3ms
```

2.2, Python usage

Use best.engine to predict the camera image:

Start the camera.

```
#usb camera
ros2 launch usb_cam camera.launch.py
#nuwa camera
ros2 launch ascamera hp60c.launch.py
```

Enter the code folder:

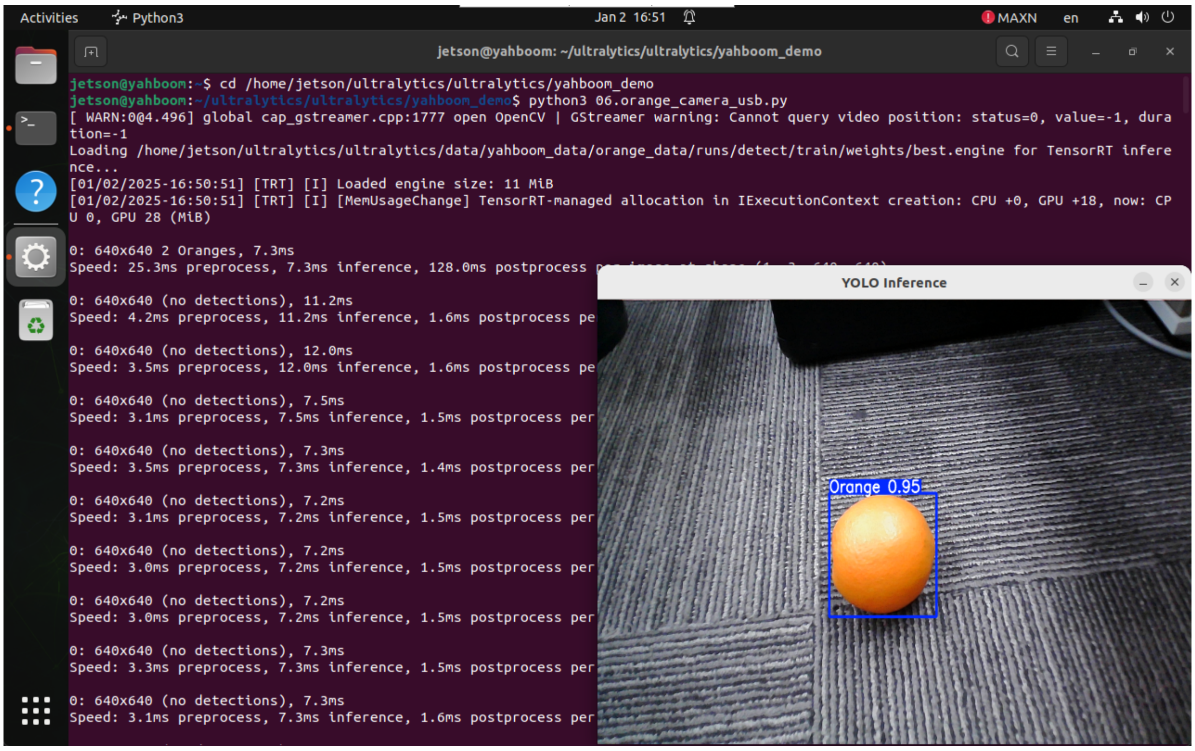
```
cd /home/jetson/ultralytics/ultralytics/yahboom_demo
```

Run the code: Click the preview image, press the q key to terminate the program!

```
python3 06.orange_camera_usb.py
```

Effect preview

Yolo recognizes the output video location: /home/jetson/ultralytics/ultralytics/output/



```
jetson@yahboom:~$ cd /home/jetson/ultralytics/ultralytics/yahboom_demo
jetson@yahboom:~/ultralytics/ultralytics/yahboom_demo$ python3 06.orange_camera_usb.py
[ WARN:0@4.496] global cap_gstreamer.cpp:1777 open OpenCV | GStreamer warning: Cannot query video position: status=0, value=-1, dura
tion=-1
Loading /home/jetson/ultralytics/ultralytics/data/yahboom_data/orange_data/runs/detect/train/weights/best.engine for TensorRT infere
nce...
[01/02/2025-16:50:51] [TRT] [I] Loaded engine size: 11 MiB
[01/02/2025-16:50:51] [TRT] [I] [MemUsageChange] TensorRT-managed allocation in IExecutionContext creation: CPU +0, GPU +18, now: CP
U 0, GPU 28 (MiB)

0: 640x640 2 Oranges, 7.3ms
Speed: 25.3ms preprocess, 7.3ms inference, 128.0ms postprocess

0: 640x640 (no detections), 11.2ms
Speed: 4.2ms preprocess, 11.2ms inference, 1.6ms postprocess pe

0: 640x640 (no detections), 12.0ms
Speed: 3.5ms preprocess, 12.0ms inference, 1.6ms postprocess pe

0: 640x640 (no detections), 7.5ms
Speed: 3.1ms preprocess, 7.5ms inference, 1.5ms postprocess per

0: 640x640 (no detections), 7.3ms
Speed: 3.5ms preprocess, 7.3ms inference, 1.4ms postprocess per

0: 640x640 (no detections), 7.2ms
Speed: 3.1ms preprocess, 7.2ms inference, 1.5ms postprocess per

0: 640x640 (no detections), 7.2ms
Speed: 3.0ms preprocess, 7.2ms inference, 1.5ms postprocess per

0: 640x640 (no detections), 7.2ms
Speed: 3.0ms preprocess, 7.2ms inference, 1.5ms postprocess per

0: 640x640 (no detections), 7.3ms
Speed: 3.3ms preprocess, 7.3ms inference, 1.5ms postprocess per

0: 640x640 (no detections), 7.3ms
Speed: 3.1ms preprocess, 7.3ms inference, 1.6ms postprocess per
```

Sample code:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import Image, CompressedImage
from cv_bridge import CvBridge
import cv2
from ultralytics import YOLO
import os

class Image_detection(Node):
    def __init__(self):
        super().__init__('Image_detection')
        self.model =
YOLO("/home/jetson/ultralytics/ultralytics/data/yahboom_data/orange_data/runs/de
tect/train/weights/best.engine")
        self.camera_type = os.getenv('CAMERA_TYPE', 'usb')
        self.bridge = CvBridge()
        if self.camera_type == 'usb':
            topic_name = '/usb_cam/image_raw'
        else:
            topic_name = '/ascamera_hp60c/camera_publisher/rgb0/image'

        self.subscription = self.create_subscription(Image,topic_name,
self.image_callback,10)

        # Get the video frame size and frame rate
        frame_width = 640
        frame_height = 480
        fps = 15

        output_path =
"/home/jetson/ultralytics/ultralytics/output/06.orange_camera_usb.mp4"
        fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # You can use 'XVID' or 'mp4v'
depending on your platform
        self.out = cv2.VideoWriter(output_path, fourcc, fps, (frame_width,
frame_height))

    def image_callback(self, msg):
        cv_image = self.bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')

        self.proecc(cv_image)


# Loop through the video frames
    def proecc(self,frame):
        # Run YOLO inference on the frame
        results = self.model(frame)

        # Visualize the results on the frame
        annotated_frame = results[0].plot()

        # Write the annotated frame to the output video file
        self.out.write(annotated_frame)


        # Display the annotated frame
        cv2.imshow("YOLO Inference", cv2.resize(annotated_frame, (640, 480)))

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

    def cancel(self):
        cv2.destroyAllWindows()
        self.out.release()


def main(args=None):
    rclpy.init(args=args)
```

```python
    node = Image_detection()
    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        pass
    finally:
        node.cancel()
        node.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```