

## 2. TensorRt acceleration identification mark

---

**Note:** This course is only applicable to the R2L car and Yahboom's autopilot map.

If you use other models or other maps, you need to develop and debug the code by yourself, and the code provided in this chapter cannot be used directly.

### 1. tensorrt deployment process

#### 1.2. Generate .wts file

Copy the best.pt generated in the previous lesson to the ~/software/yolov5-5.0 file, open the terminal,

```
python3 gen_wts.py -w best.pt
```

Copy the generated best.wts file to tensorrtx\_yolov5\_jetson directory, and compile tensorrtx in this directory, terminal input,

```
mkdir build && cd build && cmake ..
```

Change the CLASS\_NUM in yololayer.h to 11, **11 here indicates how many kinds of traffic signs there are**, we trained 11 kinds, so it is 11. The official data set is coco, which is 80 by default. Execute makeFile. (**Every time you modify it to CLASS\_NUM, you must make it once**) Terminal input,

```
make -j4
```

#### 1.3. Generate .engine file

Generate .engine file, terminal input,

```
sudo ./yolov5 -s ../best.wts yolov5s.engine s
```

After running, **libmyplugins.so** and **yolov5s.engine** will be generated in the ~/software/tensorrt\_yolov5\_jetson/build directory, copy these two files to the yahboomcar\_yolov5 function package param/nano4G folder (Take nano4G as an example, it needs to be modified according to the specific master control).

#### 1.4. Write yaml file

In the ~/yahboomcar\_ws/src/yahboomcar\_yolov5/param directory, create a new yaml file named traffic.yaml, and copy the following content into the file,

```

PLUGIN_LIBRARY: libmyplugins.so
engine_file_path: yolov5s.engine
CONF_THRESH: 0.5
IOU_THRESHOLD: 0.4
categories:
["Go_straight", "Turn_right", "whistle", "Sidewalk", "Limiting_velocity", "Shutdown",
"School_decelerate", "Parking_lotB", "Parking_lotA", "Green_light", "Red_light"]

```

## 1.5. Modify the yolov5.py code

Modify yolov5.py in the ~/yahboomcar\_ws/src/yahboomcar\_yolov5/scripts directory, and change the value of file\_yaml to the newly created traffic.yaml, as shown below,

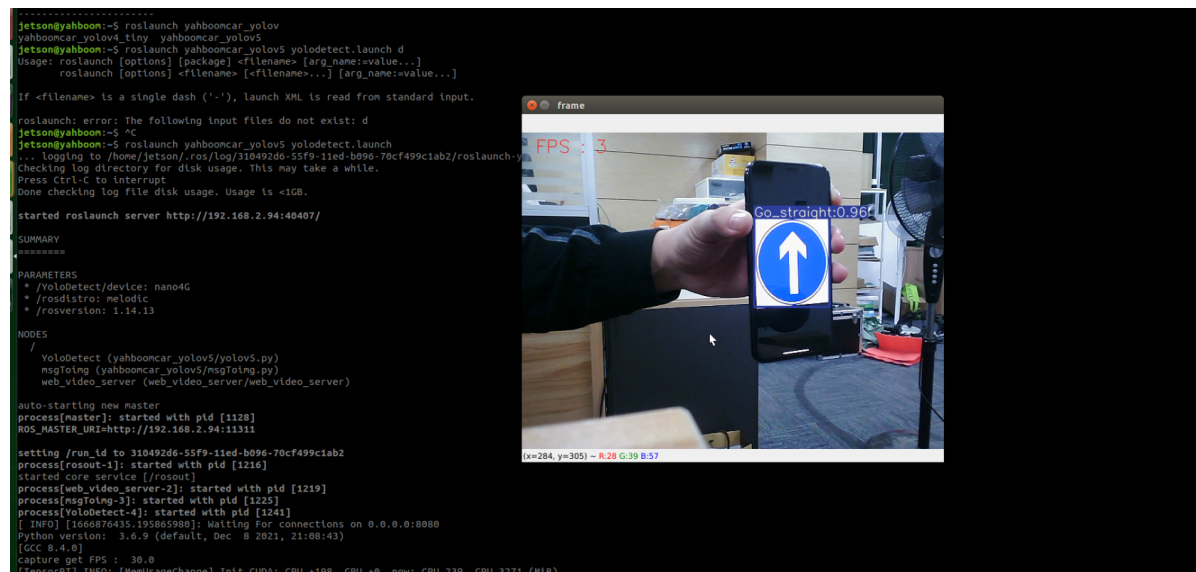
```
file_yaml = param_ + 'traffic.yaml'
```

## 2. Test run

Take nano as an example, terminal input,

```
roslaunch yahboomcar_yolov5 yolodetect.launch device:=nano4G
```

As shown below,



Use rostopic list to view topics, terminal input,

```
rostopic list
```

As shown below,

```

jetson@yahboom:~$ rostopic list
/Detect/image_msg
/DetectMsg
/rosout
/rosout_agg
/yoloDetect/image
jetson@yahboom:~$

```

We can see that there is a topic /DetectMsg, which is the specific information of the recognized traffic sign. Use rostopic echo to view the specific information of the message, terminal input,

```
rostopic echo /DetectMsg
```

As shown in the figure below, the red frame is the specific information of the identified sign, including the frame\_id, which is the category, score, center coordinates, etc. In the following program, as long as we subscribe to this topic, we can get these data.

The image displays a ROS2 environment where a traffic sign is being detected. On the left, a terminal window shows the launch of a custom ROS2 package. In the center, a camera feed shows a hand holding a smartphone with a blue 'Go Straight' traffic sign overlaid. The sign is labeled 'Go\_straight:0.97'. On the right, a terminal window shows the output of the 'rostopic echo /DetectMsg' command, displaying a JSON message with the following data:

```

data:
  frame_id: "Go_straight"
  stamp:
    secs: 1666878375
    nsecs: 104087114
  scores: 0.970433831215
  ptx: 286.391418457
  pty: 66.6907653809
  distw: 132.438293457
  dlsth: 136.152648926
  centerx: 66.2191467285
  centery: 68.0763244629

```