2. TensorRt acceleration identification mark

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,

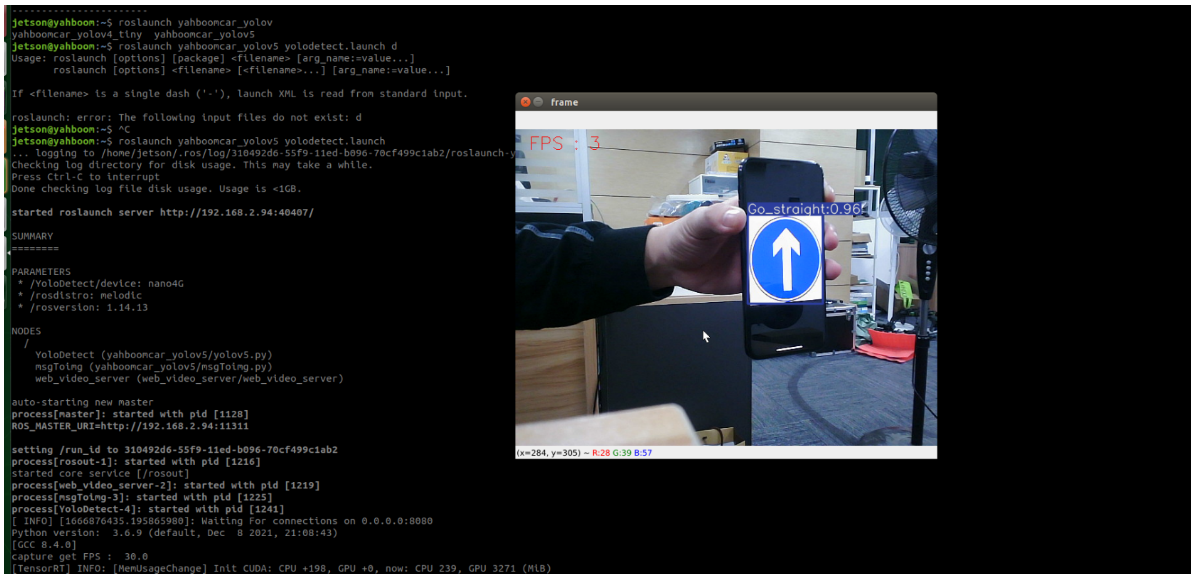
```
file_yaml = param_ + 'traffic.yaml'
```

2. Test run

Take nano as an example, terminal input,

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

As shown below,

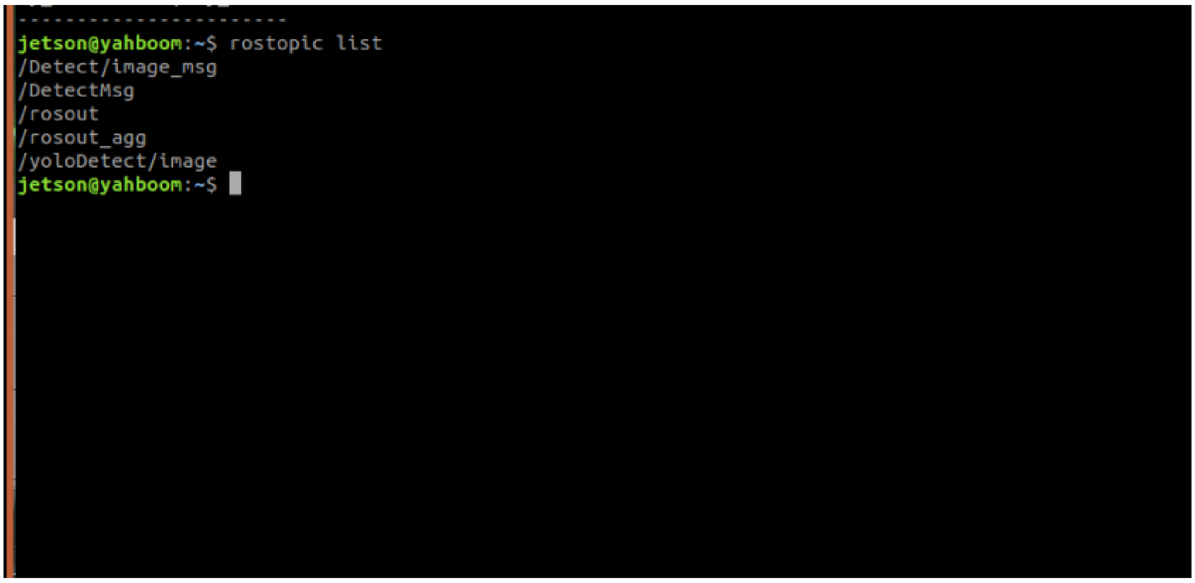


```
------------------------
jetson@yahboom:~$ roslaunch yahboomcar_yolov
yahboomcar_yolov4_tiny  yahboomcar_yolov5
jetson@yahboom:~$ roslaunch yahboomcar_yolov5 yolodetect.launch d
Usage: roslaunch [options] [package] <filename> [arg_name:=value...]
       roslaunch [options] <filename> [<filename>...] [arg_name:=value...]

If <filename> is a single dash ('-'), launch XML is read from standard input.

roslaunch: error: The following input files do not exist: d
jetson@yahboom:~$ ^C
jetson@yahboom:~$ roslaunch yahboomcar_yolov5 yolodetect.launch
... logging to /home/jetson/.ros/log/310492d6-55f9-11ed-b096-70cf499c1ab2/roslaunch-y
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.2.94:40407/

SUMMARY
========

PARAMETERS
 * /YoloDetect/device: nano4G
 * /rosdistro: melodic
 * /rosversion: 1.14.13

NODES
  /
    YoloDetect (yahboomcar_yolov5/yolov5.py)
    msgToimg (yahboomcar_yolov5/msgToimg.py)
    web_video_server (web_video_server/web_video_server)

auto-starting new master
process[master]: started with pid [1128]
ROS_MASTER_URI=http://192.168.2.94:11311

setting /run_id to 310492d6-55f9-11ed-b096-70cf499c1ab2
process[rosout-1]: started with pid [1216]
started core service [/rosout]
process[web_video_server-2]: started with pid [1219]
process[msgToimg-3]: started with pid [1225]
process[YoloDetect-4]: started with pid [1241]
[ INFO] [1666876435.195865980]: Waiting For connections on 0.0.0.0:8080
Python version:  3.6.9 (default, Dec  8 2021, 21:08:43)
[GCC 8.4.0]
capture get FPS :  30.0
[TensorRT] INFO: [MemUsageChange] Init CUDA: CPU +198, GPU +0, now: CPU 239, GPU 3271 (MiB)
```

Use rostopic list to view topics, terminal input,

```
rostopic list
```

As shown below,

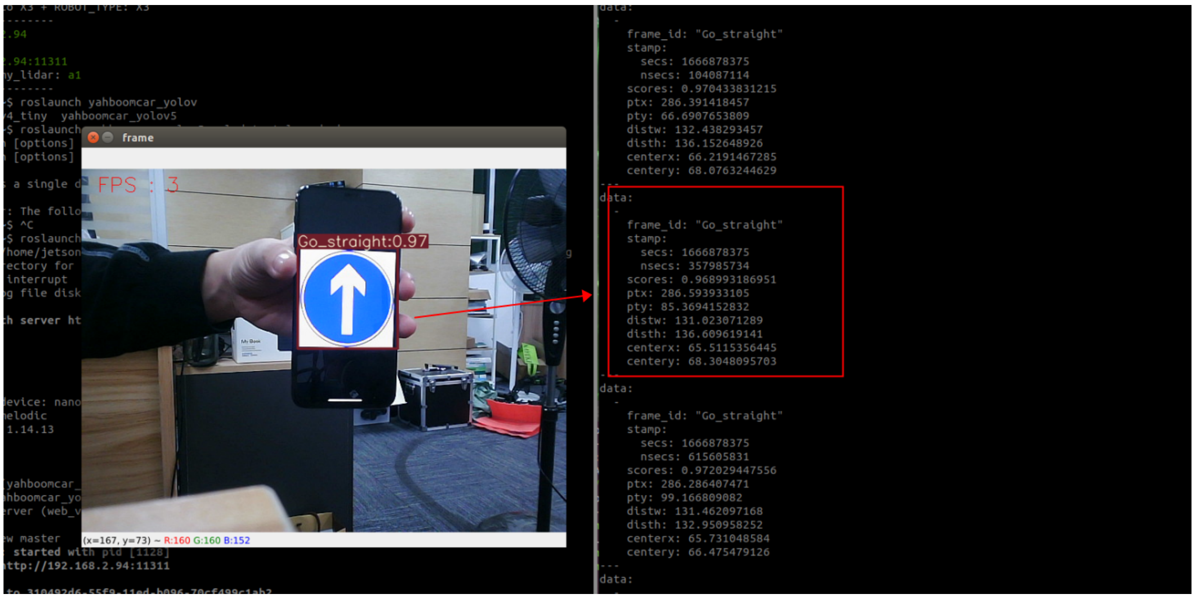


```
------------------------
jetson@yahboom:~$ rostopic list
/Detect/image_msg
/DetectMsg
/rosout
/rosout_agg
/yoloDetect/image
jetson@yahboom:~$ 
```

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.



```
o X3 + ROBOT_TYPE: X3
--------
.94

.94:11311
y_lidar: a1
--------
$ roslaunch yahboomcar_yolov
4_tiny  yahboomcar_yolov5
$ roslaunch
 [options]
 [options]

 a single d

: The follo
$ ^C
$ roslaunch
home/jetson
ectory for
interrupt
g file disk

h server ht




evice: nano
elodic
1.14.13



yahboomcar_
hboomcar_yo
rver (web_v

w master
 started with pid [1128]
ttp://192.168.2.94:11311

to 310492d6-55f9-11ed-b096-70cf499c1ab2
```

```
data:
  -
    frame_id: "Go_straight"
    stamp:
      secs: 1666878375
      nsecs: 104087114
    scores: 0.970433831215
    ptx: 286.391418457
    pty: 66.6907653809
    distw: 132.438293457
    disth: 136.152648926
    centerx: 66.2191467285
    centery: 68.0763244629
---
data:
  -
    frame_id: "Go_straight"
    stamp:
      secs: 1666878375
      nsecs: 357985734
    scores: 0.968993186951
    ptx: 286.593933105
    pty: 85.3694152832
    distw: 131.023071289
    disth: 136.609619141
    centerx: 65.5115356445
    centery: 68.3048095703
---
data:
  -
    frame_id: "Go_straight"
    stamp:
      secs: 1666878375
      nsecs: 615605831
    scores: 0.972029447556
    ptx: 286.286407471
    pty: 99.166809082
    distw: 131.462097168
    disth: 132.950958252
    centerx: 65.731048584
    centery: 66.475479126
---
data:
```