5、yolov5+tensorrt acceleration（jetson）

tensorrt source code：https://github.com/wang-xinyu/tensorrtx

Official installation tutorial:：https://docs.nvidia.com/deeplearning/tensorrt/install-guide/index.html

Official tutorial：https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html

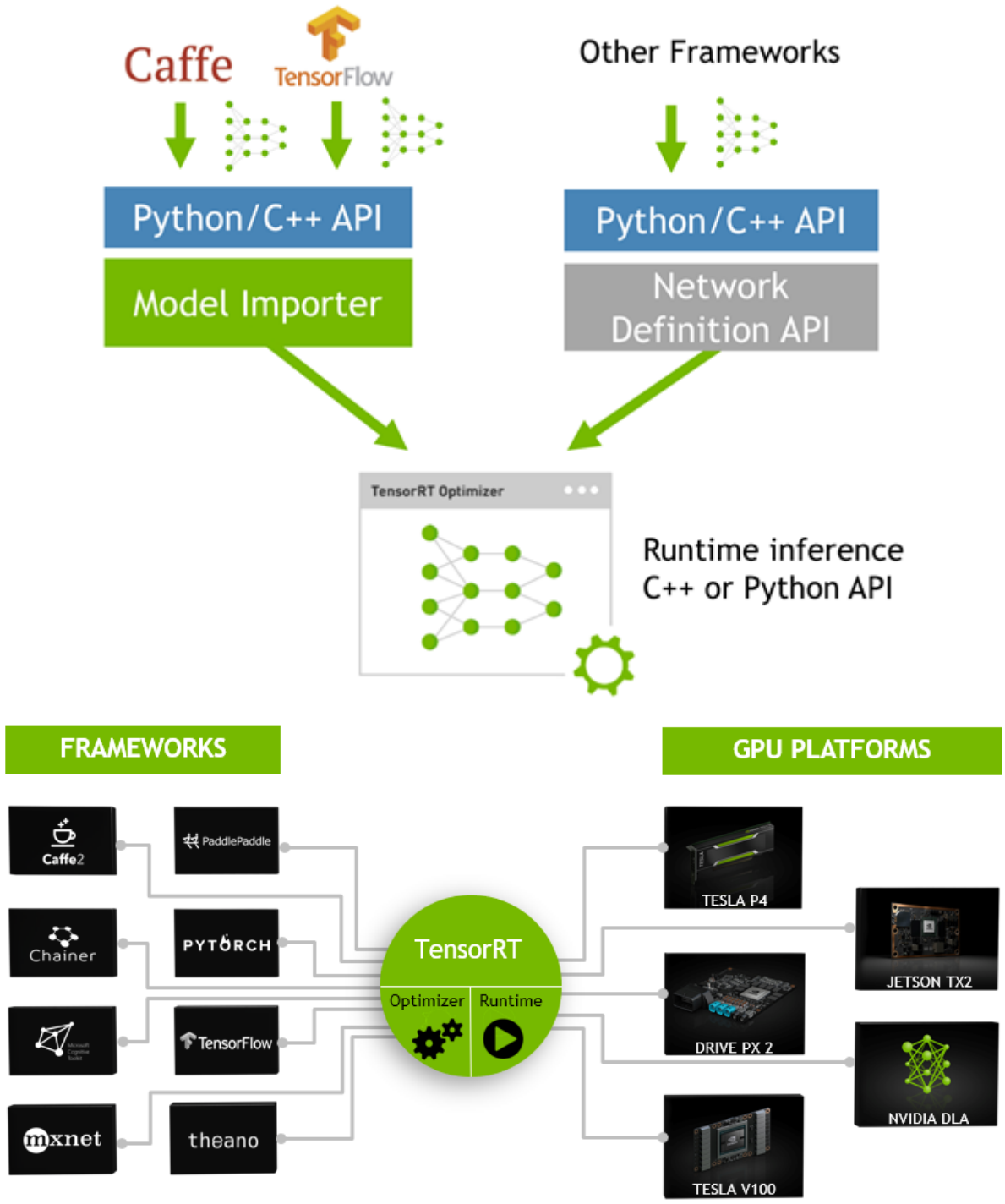
5.1. Introduction

TensorRT is a high-performance deep learning inference (Inference) optimizer that can provide low-latency, high-throughput deployment inference for deep learning applications. TensorRT can be used to accelerate inference in ultra-large-scale data centers, embedded platforms, or autonomous driving platforms. TensorRT can now support almost all deep learning frameworks such as TensorFlow, Caffe, Mxnet, and Pytorch. Combining TensorRT with NVIDIA GPUs can enable fast and efficient deployment and inference in almost all frameworks.

Can TensorRT speed up models?

Yes! According to official documentation, using TensorRT, it can provide 10X or even 100X acceleration in CPU or GPU mode. TensorRT provides 20X acceleration.

tensorRT is only responsible for the inference process of the model and optimizes the trained model. Generally, TensorRT is not used to train the model. tensorRT is just an inference optimizer. After your network is trained, you can throw the training model file directly into tensorRT without relying on the deep learning framework (Caffe, TensorFlow, etc.), as follows:



It can be considered that tensorRT is a deep learning framework with only forward propagation. This framework can parse the network models of Caffe and TensorFlow, and then map them one by one to the corresponding layers in tensorRT, and uniformly convert all models of other frameworks into tensorRT. Then in tensorRT, you can implement optimization strategies for NVIDIA's own GPUs and accelerate deployment.

5.2. Use

```
roslaunch yahboomcar_yolov5 yolodetect.launch device:=OrinNX display:=true
```

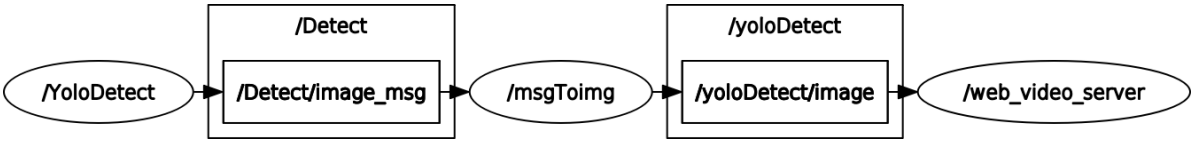
- 【device】Parameters: Robot main control equipment.
- 【display】Parameter： Whether to enable the visual interface.

Support real-time monitoring of web pages, for example:

```
192.168.2.89:8080
```

View node information

```
rqt_graph
```



Print detection information

```
rostopic echo /DetectMsg
```

Print as follows

```
data:
  -
    frame_id: "tv"
    stamp:
      secs: 1646123012
      nsecs: 116803169
    scores: 0.754247903824
    ptx: 106.302101135
    pty: 121.952651978
    distw: 144.548706055
    disth: 90.0989227295
    centerx: 72.2743530273
    centery: 45.0494613647
```

- frame_id：Identification name.
- scores：Identification scores.
- ptx，pty：Coordinates of the upper left corner of the identification box.
- distw，disth：Width and height of the recognition box.
- centerx，centery：Identify center.

5.3. TensorRT deployment process

5.3.1. Generate .wts file

For example：yolov5s.wts

Copy ~/software/tensorrtx-yolov5-v7.0/gen_wts.py and [yolov5-5.0/weights/yolov5s.pt] weight file (or the weight file downloaded from Baidu can also be used, take [yolov5s.pt] as an example)) Copy it to the [~/software/yolov5] folder, and execute gen_wts.py in this directory to generate a .wts file.

```
python3 gen_wts.py -w yolov5s.pt
```

Copy the generated [yolov5s.wts] file to the ~/software/tensorrtx-yolov5-v7.0/yolov5 directory, and compile tensorrtx in this directory

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

Modify kNumClass in config.h to yours. Because the official one uses the coco data set, the default is 80.

Execute makeFile. (Every time it is modified to kNumClass, make is required)

```
make -j4
```

5.3.2. Generate .engine file

Generate .engine file (I use yolov5s, so I use s at the end)

```
sudo ./yolov5_det -s ../yolov5s.wts yolov5s.engine s
```

You can use python files to test, you need to create yaml files.

For example：trash.yaml

```
PLUGIN_LIBRARY: libmyplugins.so             # Plug-in library
engine_file_path: yolov5s.engine            # Engine file path
CONF_THRESH: 0.5                            # CONF threshold
IOU_THRESHOLD: 0.4                          # IOU categories
categories: ["Zip_top_can", "Old_school_bag", "Newspaper", "Book",
"Toilet_paper", "Peach_pit", "Cigarette_butts", "Disposable_chopsticks",
"Egg_shell",
             "Apple_core",
             "Watermelon_rind", "Fish_bone", "Expired_tablets",
"Expired_cosmetics", "Used_batteries", "Syringe"]  # Identify categories
```

5.4. ROS deployment

Copy the generated [yolov5s.engine] and [libmyplugins.so] files to the [yahboomcar_yolov5] function package [param/OrinNX] folder, for example:

```
yahboomcar_yolov5
    ├── CMakeLists.txt
    ├── launch
    │   └── yolodetect.launch
    ├── package.xml
    ├── param
    │   ├── trash.yaml
    │   └── OrinNx
    │       ├── libmyplugins.so
    │       └── yolov5s.engine
    └── scripts
        └── yolov5.py
```

When using it, just follow the steps in [5.2].

Write yaml file

For example：coco.yaml

```
PLUGIN_LIBRARY: libmyplugins.so             # Plug-in library
engine_file_path: yolov5s.engine            # Engine file path
CONF_THRESH: 0.5                            # CONF threshold
IOU_THRESHOLD: 0.4                          # IOU threshold
categories: ["person", "bicycle", ... ...]  # Identify categories
```

If the file name changes, you need to modify the [yolov5.py] file and open it in the file path

```
sudo vim ~/yahboomcar_ws/src/yahboomcar_yolov5/scripts/yolov5.py
```

As shown below

```
class YoloDetect:
    def __init__(self):
        rospy.on_shutdown(self.cancel)
        rospy.init_node("YoloDetect", anonymous=False)
        self.pTime = self.cTime = 0
         device = rospy.get_param("~device", "OrinNX")
        param_ = rospkg.RosPack().get_path("yahboomcar_yolov5") + '/param/'
        file_yaml = param_ + 'trash.yaml'
        PLUGIN_LIBRARY = param_ + device + "/libmyplugins.so"
        engine_file_path = param_ + device + "/yolov5s.engine"
        self.yolov5_wrapper = YoLov5TRT(file_yaml, PLUGIN_LIBRARY,
engine_file_path)
        self.pub_image = rospy.Publisher('Detect/image_msg', Image_Msg,
queue_size=10)
        self.pub_msg = rospy.Publisher('DetectMsg', TargetArray, queue_size=10)
```

The parameters of [file_yaml], [PLUGIN_LIBRARY], and [engine_file_path] must correspond to the names used.