Yolo5+Tensorrt acceleration+DeepStream

1.Precautions before use

If you are using the YAHBOOM version of the image, there is no need to build the DeepStream environment. If you have built your own image, you need to build the environment for DeepStream. You can refer to the DeepStream building tutorial we provide, or you can also build your own Baidu

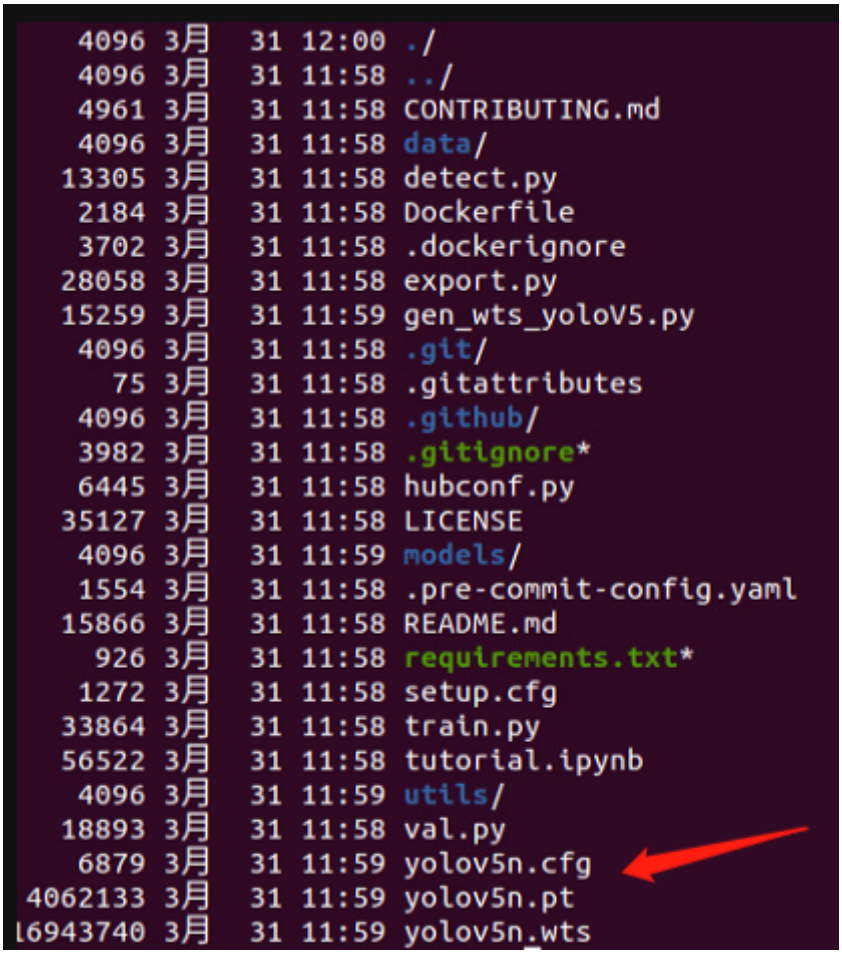
2.instructions

2.1 Model Transformation

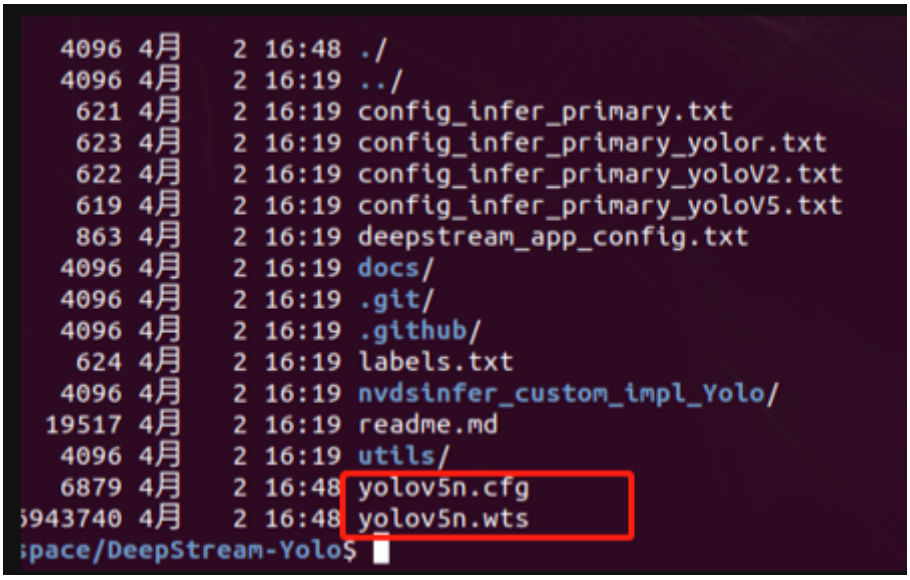
```
git clone https://github.com/marcoslucianops/DeepStream-Yolo.git
cd DeepStream-Yolo/utils

cp gen_wts_yoloV5.py ../../yolov5
cd ../../yolov5

python3 gen_wts_yoloV5.py -w  ./yolov5s.pt
```

2.2 Deployment Model

1. After successfully running the previous step, two files will appear in the directory of yolov5 , yolov5n.cfg and yolov5n.wts



2. Place yolov5n. cfg and yolov5n. wts in the DeepStream Yolo folder of Jetson orin nx



2.3 Modify the deepstream configuration file (this step can be omitted for YAHBOOM version images)

1. Modify Deepstream_ app_ Config.txt file
   The modified content is as follows:
   Comment on 70 lines, add a line after:
   config-file=config_infer_primary_yoloV5.txt

As shown in the figure:

```
65  [primary-gie]
66  enable=1
67  gpu-id=0
68  gie-unique-id=1
69  nvbuf-memory-type=0
70  #config-file=config_infer_primary.txt
71  config-file=config_infer_primary_yoloV5.txt
72
```

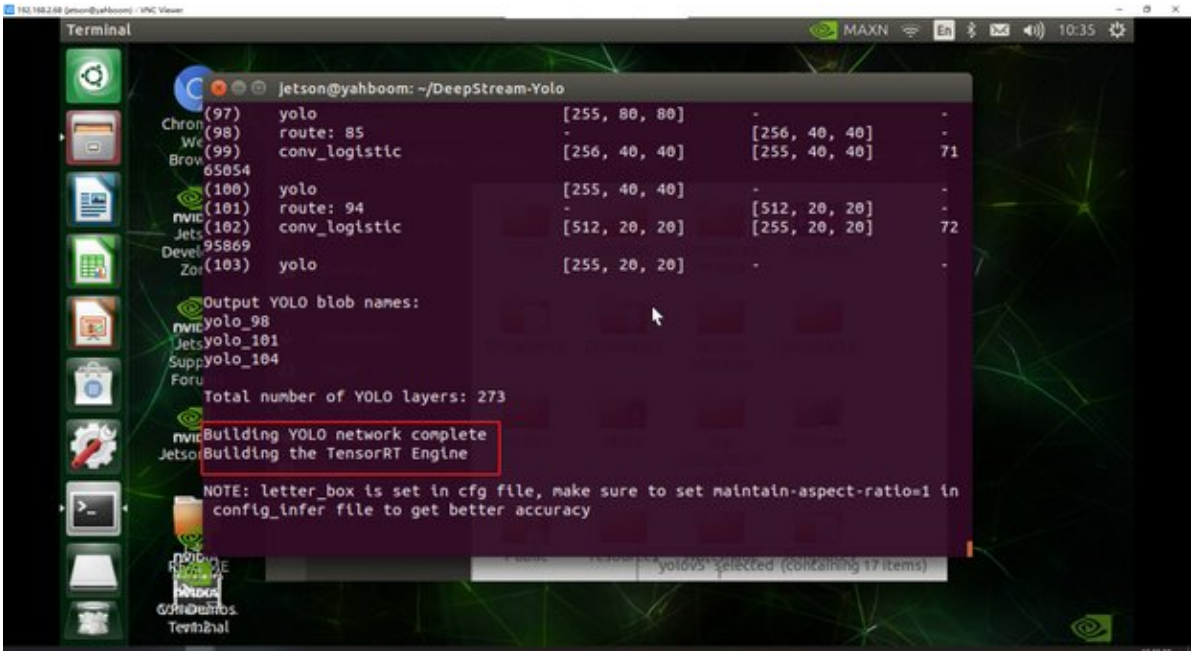
2. Modify the second configuration file config_infer_primary_yoloV5.txt

```
[property]
# omit ...
**model-engine-file=model_b2_gpu0_fp16.engine** #  fp32->fp16
batch-size=2 # batch-size Change to 2, the speed will be faster
# omit...
**network-mode=2 #**  2:Force the use of fp16 inference
# omit ...
```

Note: FPS is related to parameters such as input image size, batch size, interval, etc., and needs to be optimized according to practical applications. Here, we directly change the batch size of the input to 2, which will significantly improve the inference speed of the model

3.Compile Run

```
cd nvdsinfer_custom_impl_Yolo/
CUDA_VER=11.4 make -j4 #Modify the numerical part of 11.4 based on your CUDA version
cd ..
deepstream-app -c deepstream_app_config.txt
```



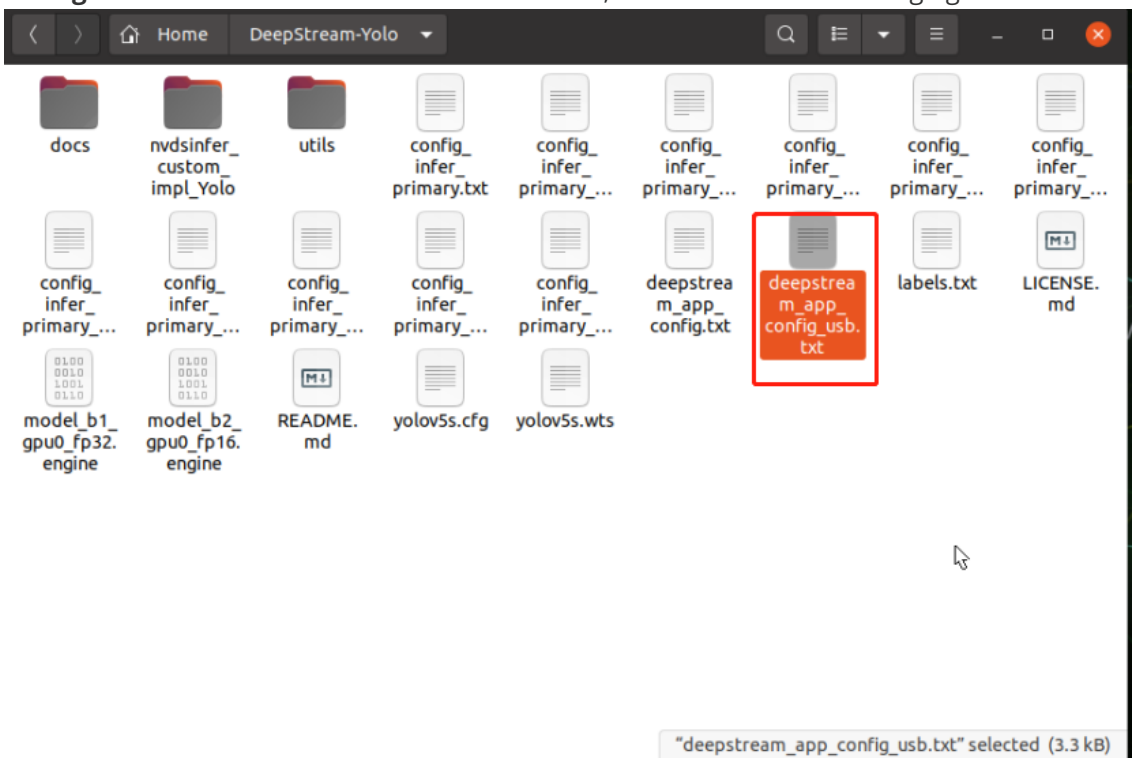
After waiting for a while, you can see that the CSI camera screen has opened

node

1. If you are using your own built image, you need to deepstream_app_config.txt Make modifications
   Modify as shown in the figure:

```
#[source0]
#enable=1
#type=3
#uri=file:///opt/nvidia/deepstream/deepstream/samples/streams/sample_1080p_h264.mp4
#num-sources=1
#gpu-id=0
#cudadec-memtype=0

[source1]
enable=1
#Type - 1=CameraV4L2 2=URI 3=MultiURI 4=RTSP 5=CSI
type=5
camera-csi-sensor-id=0
camera-width=1280
camera-height=720
camera-fps-n=30
camera-fps-d=1
```

2. If you are using a USB camera, you need to include the **deepstream in the attachment of the document_ app_ config_ Upload the file USB. txt** to Jetson and **deepstream_ app_ Config.txt** This file is located in the same location, as shown in the following figure:



then run

```
cd ~/DeepStream-Yolo
deepstream-app -c deepstream_app_config_usb.txt
```

Just a moment, we can achieve the detection.