1. yolov5 to train traffic signs

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 yourself, and the code provided in this chapter cannot be used directly.

In the previous section thirteen, we introduced how to use yolov5 to train the model. In this lesson, we will train our own model through practical operations. Here we choose to train traffic signs. In order to save training time, we train two kinds of signs, it is enough to modify the corresponding value for the training method of various signs. **This demo runs on a jetson NX board.**

1.1、Run Get_garbageData.py to generate training set pictures

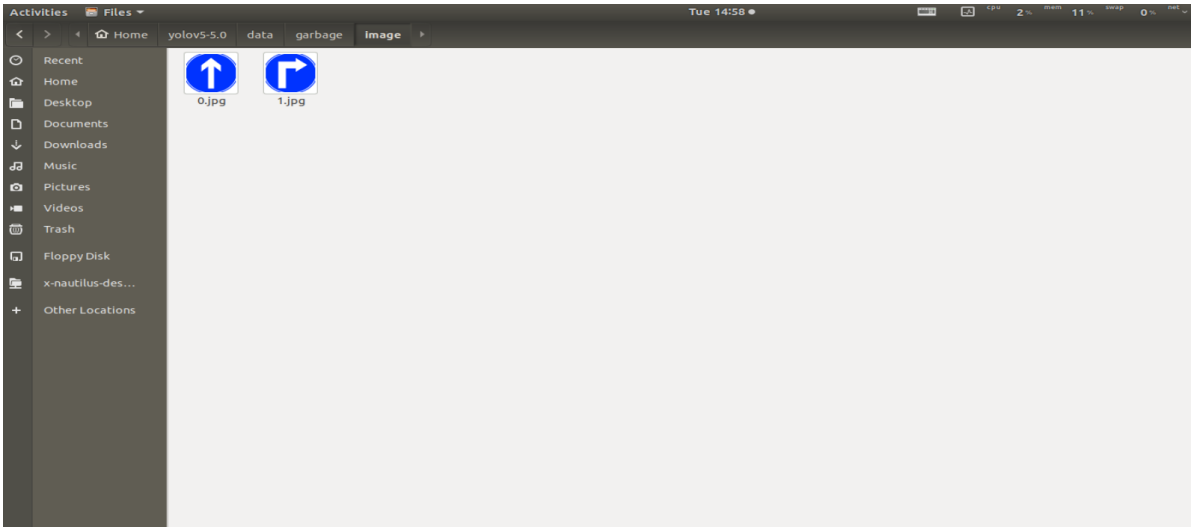
Let's first look at the main part of this function,

```
# Generate images at random locations
# Generate pictures at random positions
def transparentOverlay(path):
    # Load image from file
    # Load image from file
    bgImg = cv.imread(path, -1)
    # reset image size
    # Reset Image Size
    target = cv.resize(bgImg, (416, 416))
    rows, cols, _ = target.shape  # background image
    rectangles = []
    label = ' '
    for i in range(0, 10):
        index = np.random.randint(0, 16)
        readimg = cv.imread('./image/' + str(index) + '.png')
```

There are two key information points here, one is the number of random reads, and the other is the position of the read picture, corresponding to,

```
index = np.random.randint(0, 16)
readimg = cv.imread('./image/' + str(index) + '.png')
```

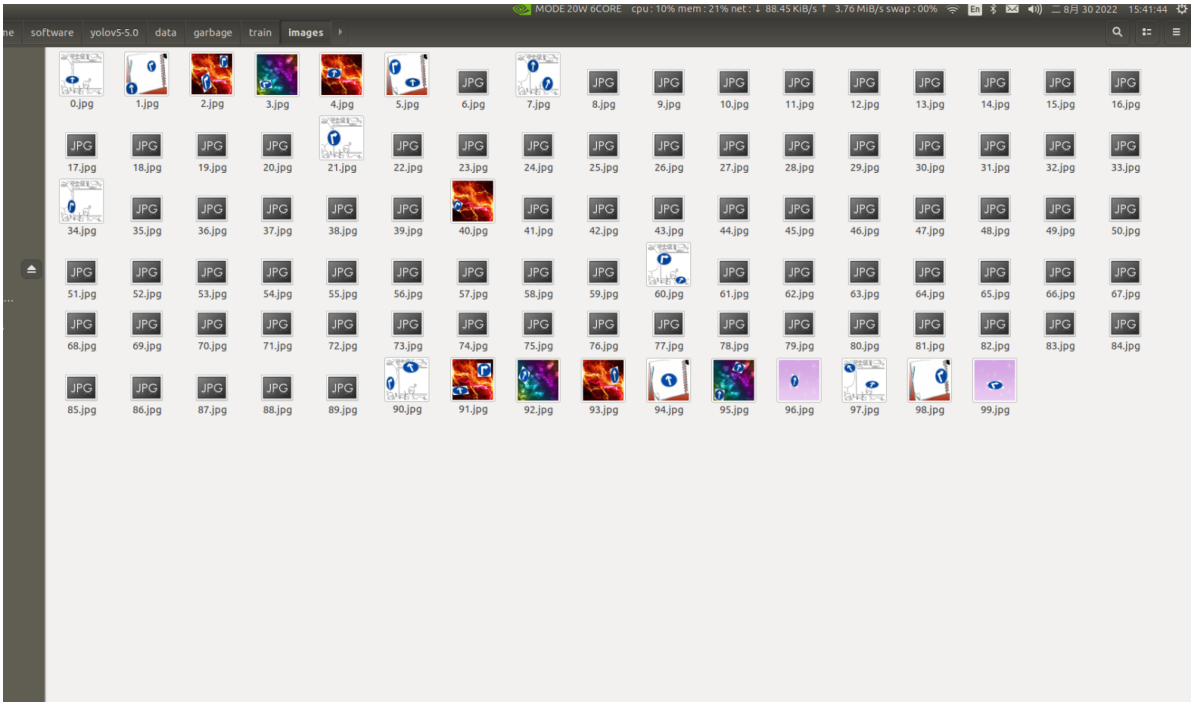
Since we **only have two types of traffic signs, we changed 16 to 2, and the data here is modified according to how many types there are**. Before training, we need to put these two types of pictures in the ~/yolov5-5.0/data/garbage/image directory, as shown in the following figure.



Continue to look down at the main content of the function,

```
def generateImage(img_total):
    rootdir = './texture'
    # List all directories and files in a folder
    # List all directories and files under the folder
    list = os.listdir(rootdir)
    for i in range(0, img_total):
        index = np.random.randint(0, len(list))
        txt_path = os.path.join(rootdir, list[index])
        overlay, label = transparentOverlay(txt_path)
        cv.imwrite("./train/images/" + str(i) + ".jpg", overlay)
        with open("./train/labels/" + str(i) + ".txt", "w") as wf:
            wf.write(label)
            wf.flush()
```

Here is the storage location of the pictures and labels after our training is completed. After the training is over, img_total training pictures and training labels will be generated in this directory. Here we set img_total to 100.



We can take a look at one of the training pictures. In fact, it is to process the picture and put it on the background picture. Various permutations and combinations form the training picture.

1.2. Modify yaml file

After we get the training picture, we can generally train it, but because the training picture is relatively large, we need to load it through a file. It can be seen from the train.py file in the ~/software/yolov5-5.0 directory,

```
 parser.add_argument('--data', type=str, default='data/garbage.yaml',
help='data.yaml path')
```

During training, this yaml file will be loaded, and the content here is the path of the trained image and the information of related tags.

We modify the content of this yaml as follows,

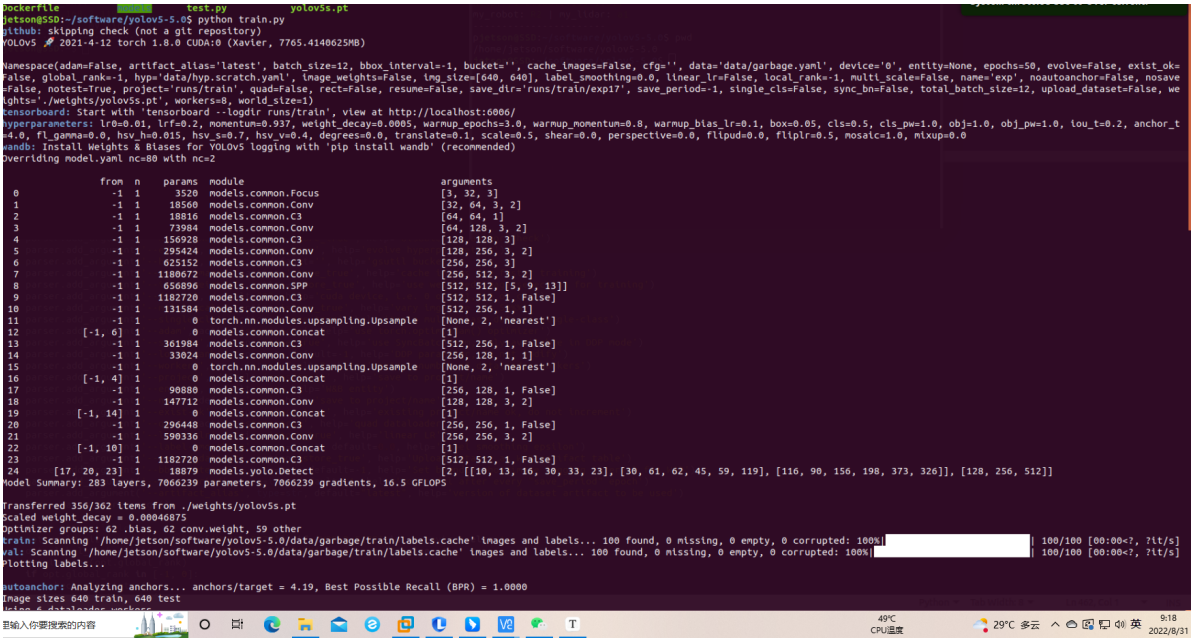
```
# train and val data
train: /home/jetson/software/yolov5-5.0/data/garbage/train/images
val: /home/jetson/software/yolov5-5.0/data/garbage/train/images
# number of classes
nc: 2
# class names
names: ["Go_Straight","Trun_Right"]
```

train and val indicate the location of the training pictures, nc indicates how many types of pictures there are, and name indicates the type names of these pictures, which need to be consistent with the order of ~/yolov5-5.0/data/garbage/image pictures.

1.3.Run train.py training model

```
cd ~/software/yolov5-5.0
python train.py
```

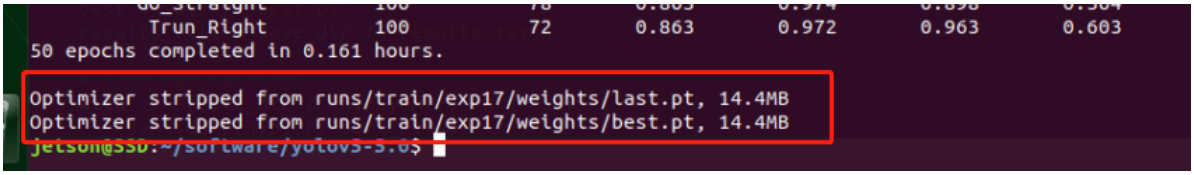
The screenshot of successful operation is shown in the figure below.



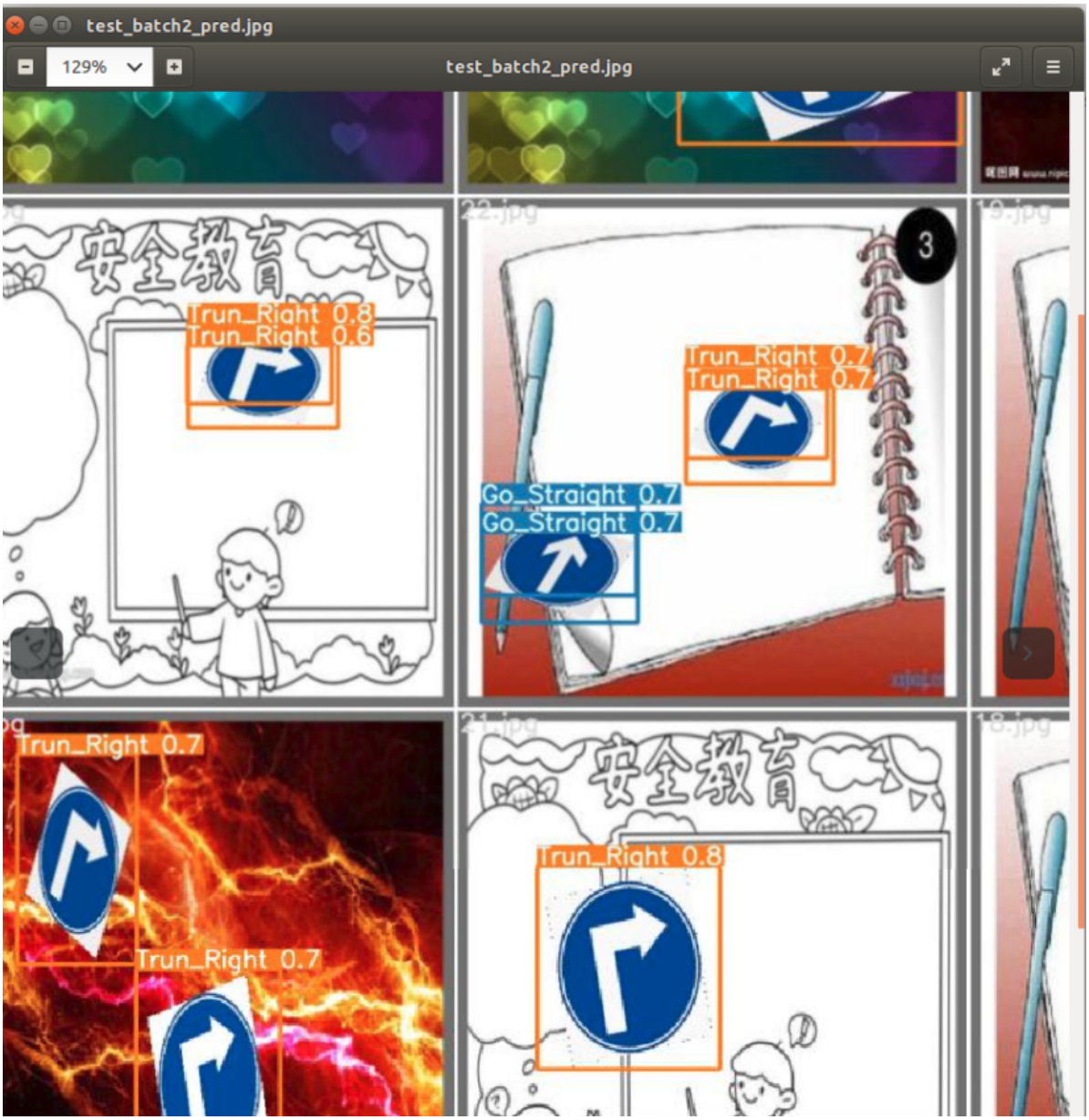
Here it is trained 50 times, which means the epoch value is 50. **If it is a virtual machine to train, it needs to be modified,**

```
Put parser.add_argument('--device', default='0', help='cuda device, i.e. 0 or
0,1,2,3 or cpu')
Change to,parser.add_argument('--device', default='cpu', help='cuda device, i.e.
0 or 0,1,2,3 or cpu')
```

After training, the path to save the model will be printed on the terminal, as shown in the figure below,



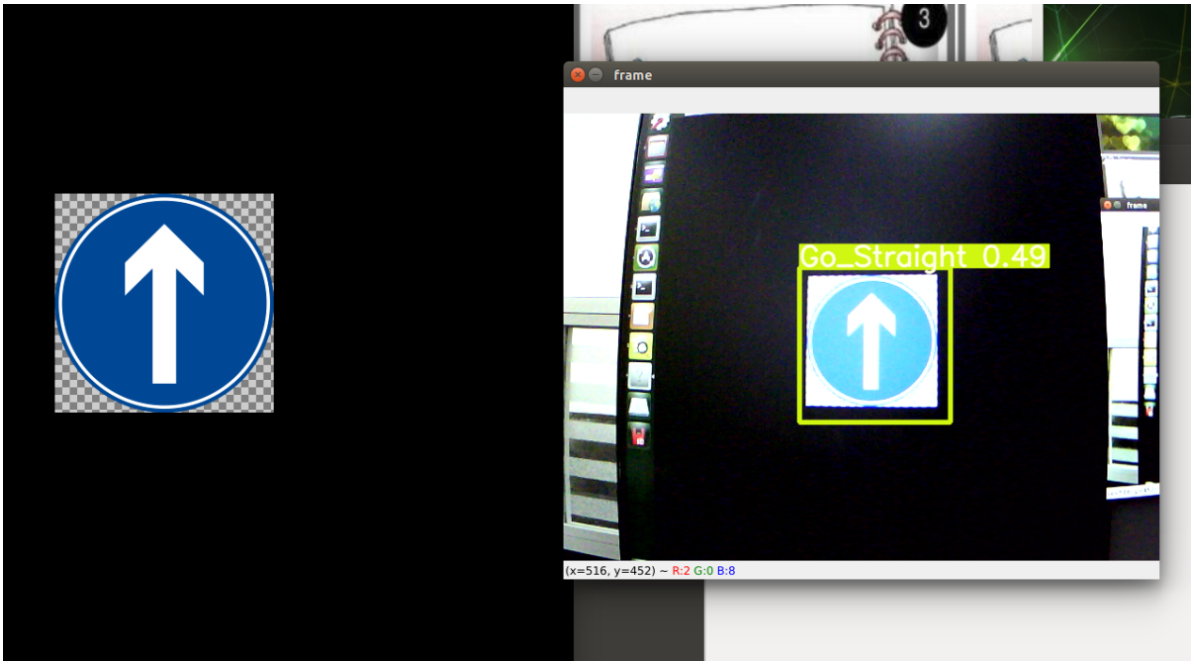
Go to ~/software/yolov5-5.0/train/runs/train/exp17 to view the content inside. The test_batch0_pred.jpg, test_batch1_pred.jpg and test_batch2_pred.jpg inside are the pictures we predicted.



It can be seen that the recognition accuracy is quite high. Then, we put ~/software/yolov5-5.0/train/runs/train/exp17/weights/best.pt in the ~/software/yolov5-5.0 directory, and then modify the content in detection_video.py,

```
Change model_path = 'weights/yolov5s.pt' to,
model_path = './best.pt
```

Run detection_video.py, you can use the model we just trained to recognize the two signs just trained in real time. As shown below,


frame
Go_Straight 0.49
(x=516, y=452) ~ R:2 G:0 B:8