1. Use Yolov5 to train traffic signs

In the previous course, we introduced how to use yolov5 to train the model. In this course, we will train our own model through actual operations. Here we choose to train traffic signs. In order to save training time, we train two kinds of signs and multiple kinds of signs. Just modify the corresponding value of the training method.

Notice:

Due to the limited performance of Jetson Nano, the training model cannot be completed. This training process uses a Jetson Orin NX 16G motherboard. You can also use a computer with an independent graphics card. The training process is for reference only. You need to bring your own relevant hardware equipment.

1.1. Run Get_garbageData.py to generate training set images

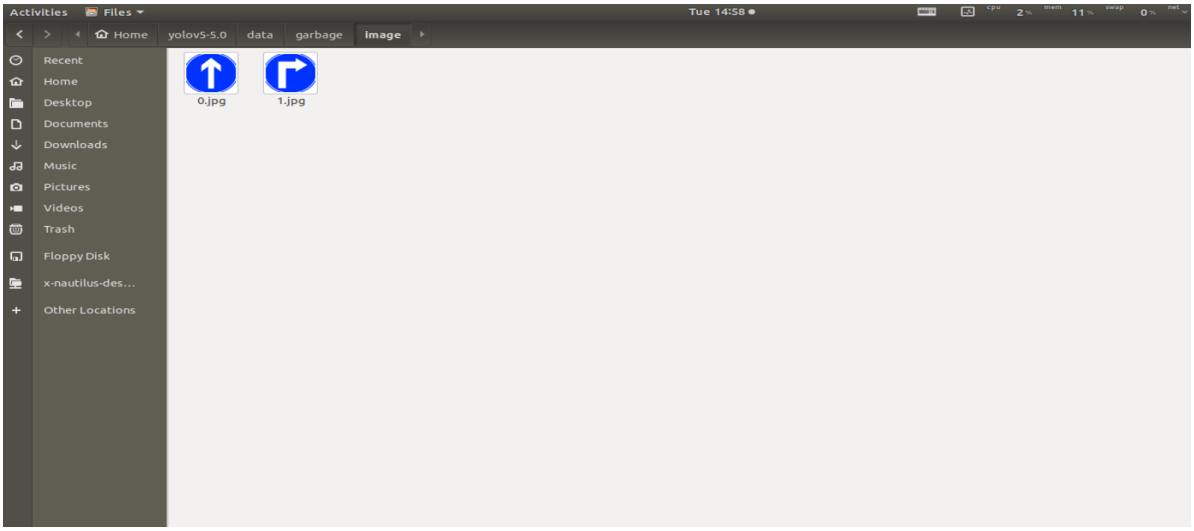
Let's look at the main part of this function first,

```
# Generate images at random locations
# 随机位置生成图片
def transparentOverlay(path):
     # Load image from file
     # 从文件加载图像
     bgImg = cv.imread(path, -1)
     # reset image size
     # 重置图像大小
     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 where the picture is read, which correspond to:

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

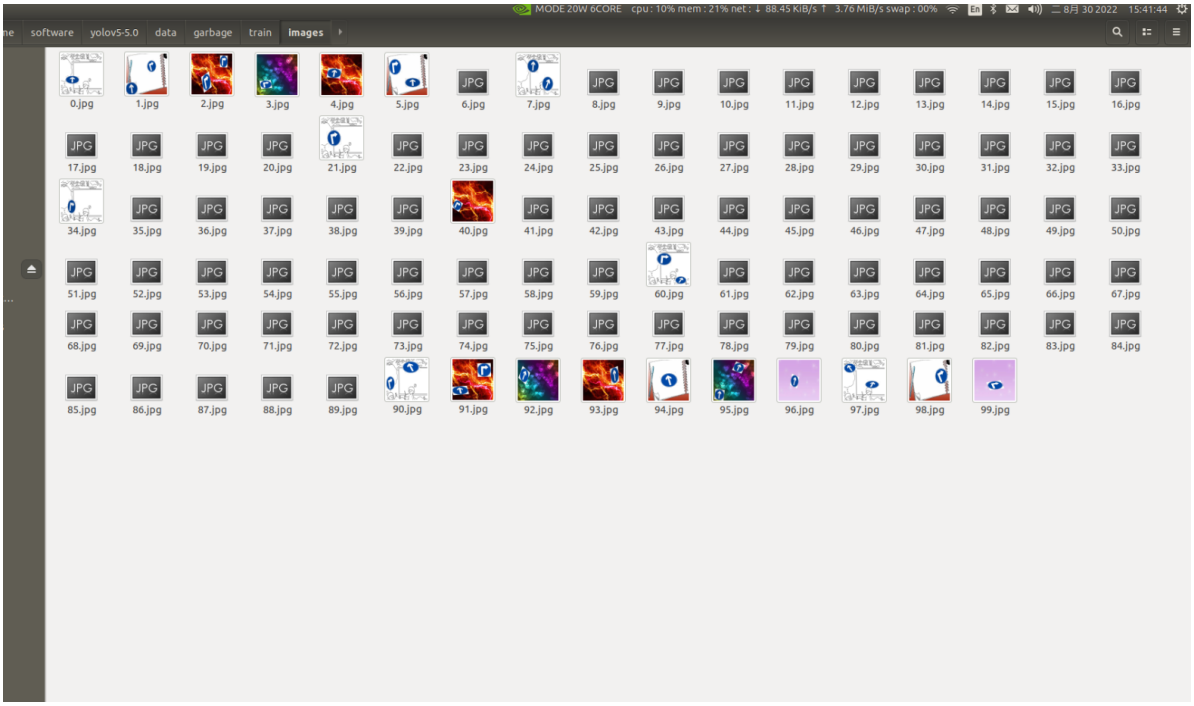
Since we only have two types of traffic signs, we change 16 to 2. The data here is modified based on how many types there are. Before training, we need to put these two types of images in the ~/yolov5-5.0/data/garbage/image directory, as shown in the figure below,



Continue below to see the main content of the function,

```
def generateImage(img_total):
      rootdir = './texture'
      # List all directories and files in a folder
      # 列出文件夹下所有的目录与文件
      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 we have completed training. After training, img_total training images 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, we perform some processing on the picture and then place it on the background picture. Various permutations and combinations form the training picture.

1.2. Modify yaml file

After we get the training images, we can generally train. However, because the training images are relatively large, we need to load them through a file. From the train.py file in the ~/software/yolov5-5.0 directory, we can see that

```
  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 related label information.

We modify the content of this yaml as follows,

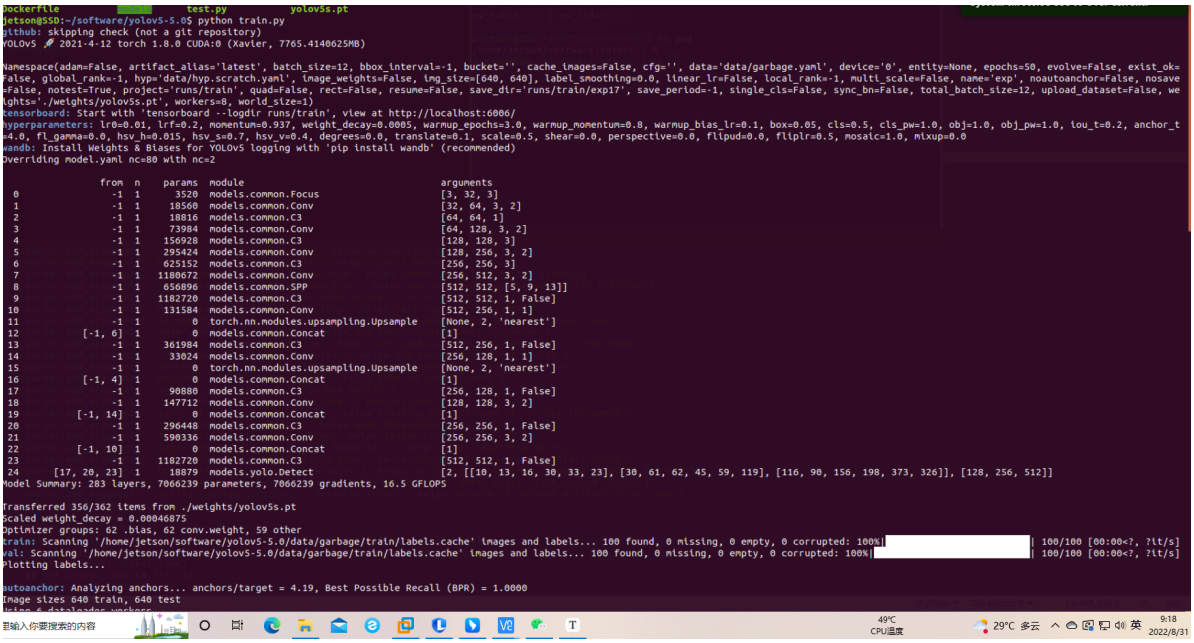
```
# train and val data
train: /home/jetson/software/yolov5-5.0/data/garbage/train/images
val: /home/jetson/software/yolov5-5.0/data/garbage/train/images
# number of classes
NC: 2
# class names
names: ["Go_Straight","Trun_Right"]
```

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

1.3. Run train.py to train the model

```
cd ~/software/yolov5-5.0
pythontrain.py
```

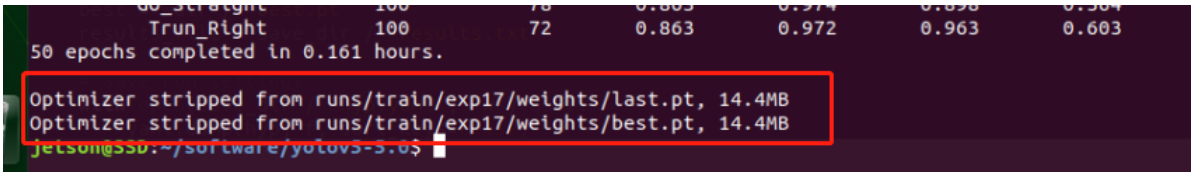
A screenshot of successful operation is shown below:



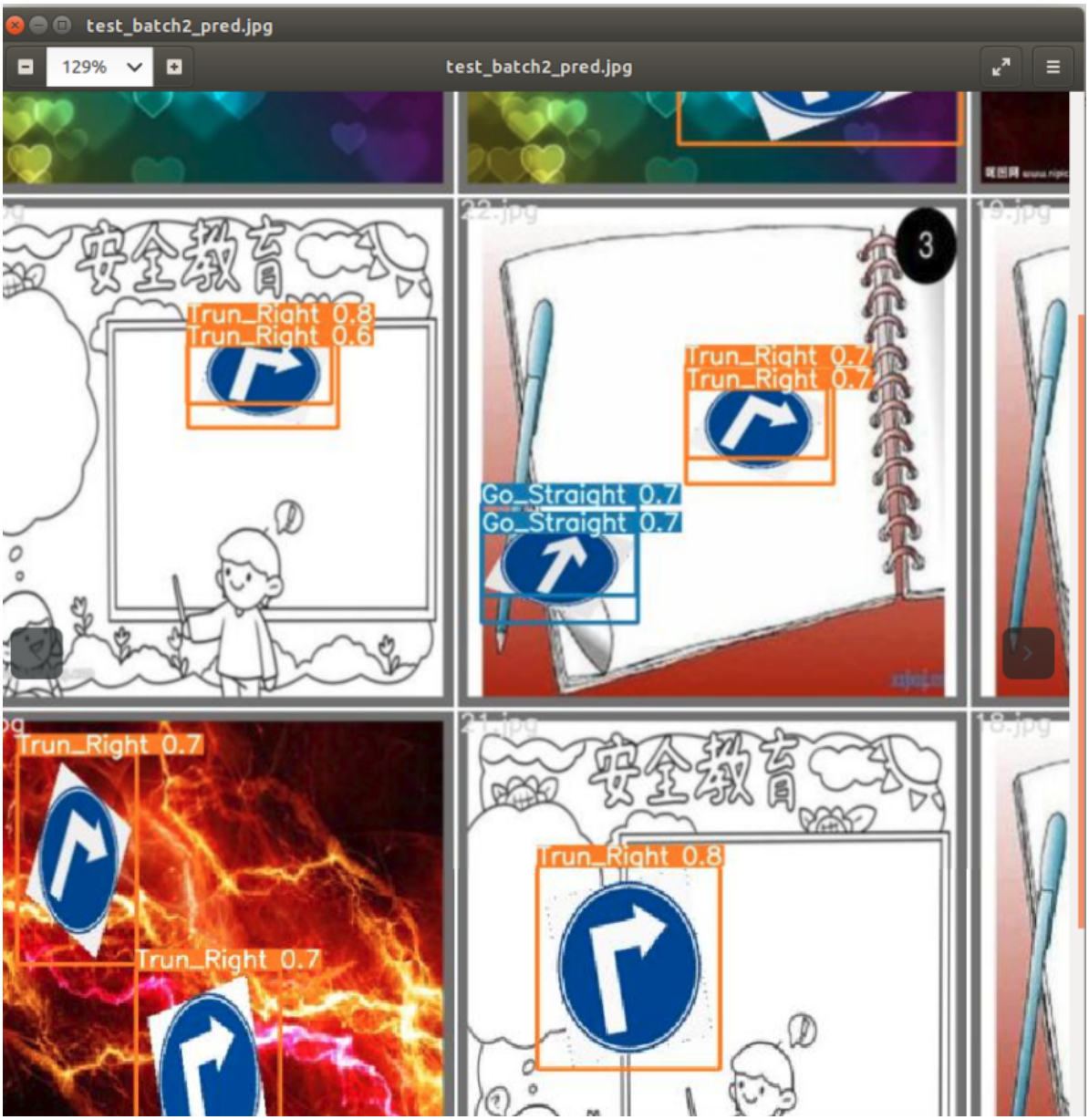
This is trained 50 times, that is, the epoch value is 50. **If a virtual machine is used for training, it needs to be modified.**

```
Put parser.add_argument('--device', default='0', help='cuda device, i.e. 0 or
0,1,2,3 or cpu')
Modify 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 contents inside. The test_batch0_pred.jpg, test_batch1_pred.jpg and test_batch2_pred.jpg inside are the pictures we predicted.

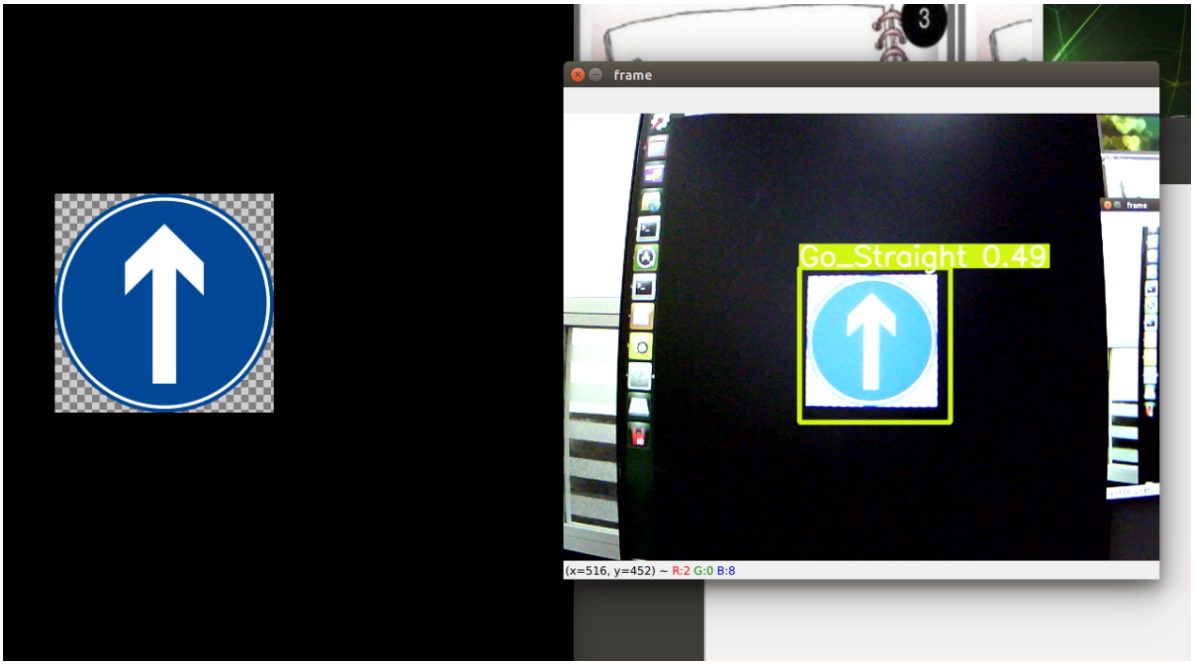


It can be seen that the recognition accuracy is quite high. Then, we put ~/software/yolov5-5.0/train/runs/train/exp17/weights/best.pt in the ~/software/yolov5-5.0 directory, and then modify

The content inside detection_video.py,

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

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


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