

1. yolov5 to train traffic signs

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

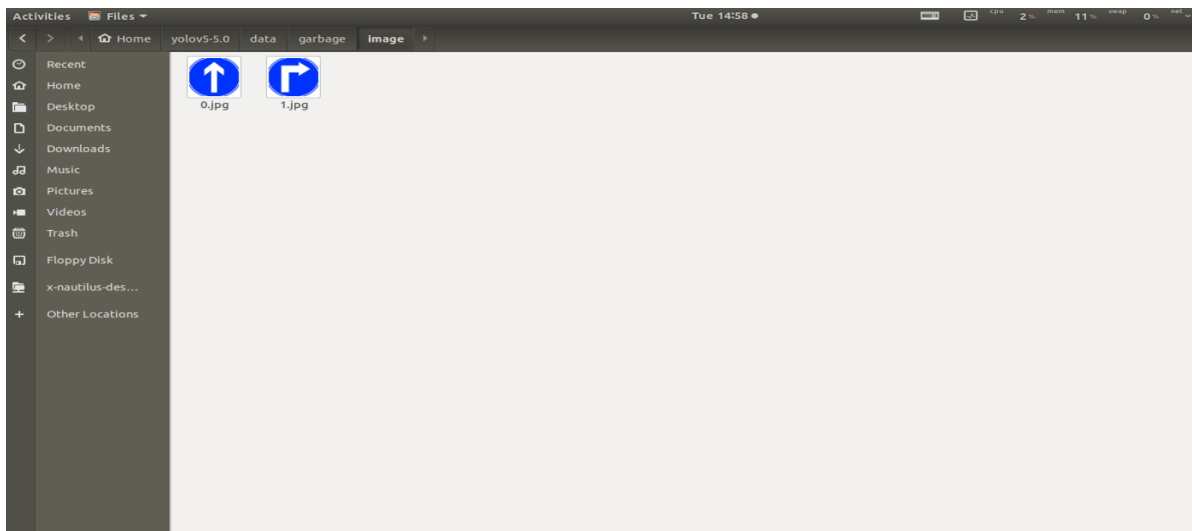
Let's first look at the main part of this function,

```
# 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)
        reading = 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)
reading = cv.imread('./image/' + str(index) + '.png')
```

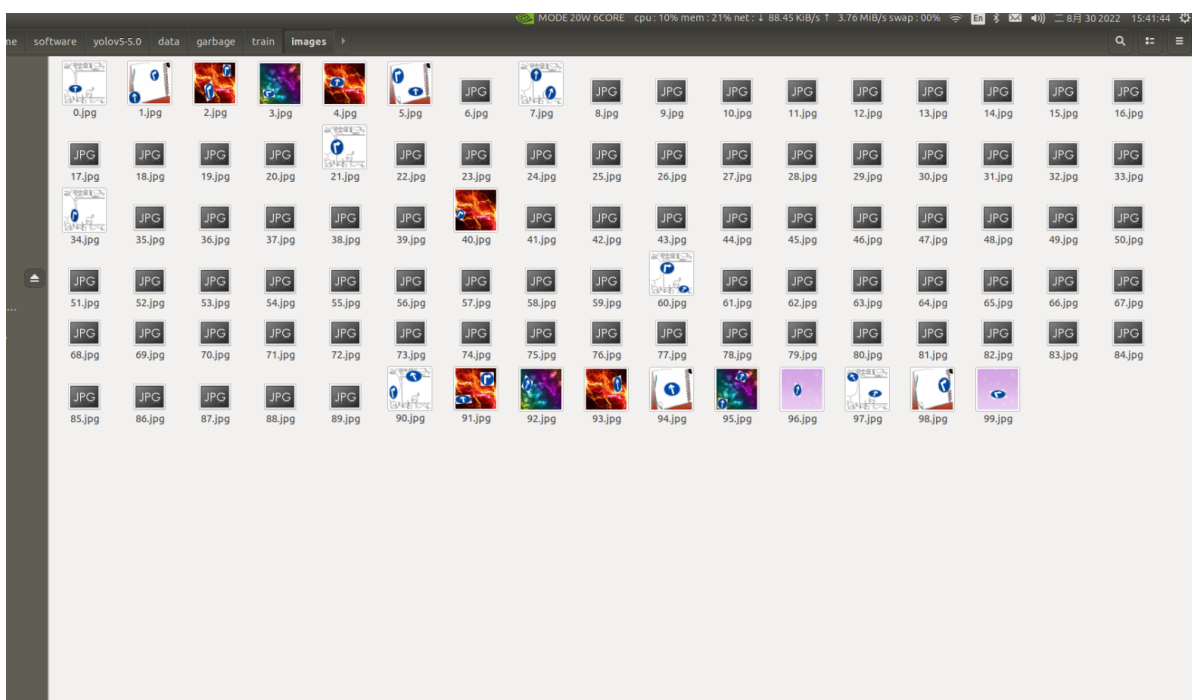
Since we **only have two types of traffic signs, we changed 16 to 2, and the data here is modified according to how many types there are.** Before training, we need to put these two types of pictures in the ~/yolov5-5.0/data/garbage/image directory, as shown in the following figure.



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"]
```

1.3.Run train.py training model

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

```
yoloenv$ python test.py yolov5.pt
Jetson$ sudo -i /software/yolov5-5.0 python train.py
gitHub: skipping check (not a git repository)
YOLOv5 🚀 2021-4-12 torch 1.8.0 CUDA=9 (Xavier), 7765.4146625MB

Namespace(adam=False, artifact_alias='latest', batch_size=12, bbox_interval=-1, bucket='', cache_images=False, cfg='', data='data/garbage.yanl', device='0', entity=None, epochs=50, evolve=False, exist_ok=True, global_rank=-1, hyp='./data/hyp/cratch.yaml', image_weights=False, imgsz=[640, 640], label_smoothing=0, linear_lr=False, local_rank=-1, multi_scale=False, name='exp', noutautoanchor=False, nosave=False, notesize, project='runs/train', quadrate=False, rect=False, resume=False, save_dir='./runs/train/exp/r', save_period=1, single_cls=False, sync_bn=False, total_batch_size=12, upload_dataset=False, weights='./weights/yolov5s.pt', workers=8, world_size=1)

Tensorboard: Start with 'tensorboard --logdir runs/train'. view at http://localhost:6000/
Parameters: lr=0.001, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3, warmup_momentum=0.8, warmup_bias_lr=1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_lw=1.0, iou_t=0.2, anchor_thresh=0.8, fl_gamma=0.0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0
wandb: Install Weights & Biases for YOLOv5 logging with 'pip install wandb' (recommended)

Overriding model.yaml nc=80 with nc=2

   from  n  params module                                arguments
   ---  -  -  ----  -
0     -1  1    3520 models.common.Focus                  [3, 32, 3]
1     -1  1   18560 models.common.Conv                    [32, 64, 3, 2]
2     -1  1   19816 models.common.C3                      [64, 64, 1]
3     -1  1   73984 models.common.Conv                    [64, 128, 3, 2]
4     -1  1  156928 models.common.C3                      [128, 128, 3]
5     -1  1  295424 models.common.Conv                    [128, 256, 3, 2]
6     -1  1  625152 models.common.C3                      [256, 256, 3]
7     -1  1  1180672 models.common.Conv                    [256, 512, 3, 2]
8     -1  1  650896 models.common.SPP                       [512, 512, [5, 9, 13]]
9     -1  1  1182720 models.common.C3                      [512, 512, 1, False]
10    -1  1  131584 models.common.Conv                    [512, 256, 1, 1]
11    -1  1      0 torch.nn.modules.upsampling.Upsample       [None, 2, 'nearest']
12    [-1, 0] 1      0 models.common.Concat                   [1]
13    -1  1  361984 models.common.C3                      [512, 256, 1, False]
14    -1  1  33024 models.common.Conv                    [256, 128, 1, 1]
15    -1  1      0 torch.nn.modules.upsampling.Upsample       [None, 2, 'nearest']
16    [-1, 4] 1      0 models.common.Concat                   [1]
17    -1  1  90880 models.common.C3                      [256, 128, 1, False]
18    -1  1  147712 models.common.Conv                    [128, 128, 3, 2]
19    [-1, 14] 1      0 models.common.Concat                   [1]
20    -1  1  296448 models.common.C3                      [256, 256, 1, False]
21    -1  1  590336 models.common.Conv                    [256, 256, 3, 2]
22    [-1, 10] 1      0 models.common.Concat                   [1]
23    -1  1  1182720 models.common.C3                      [512, 512, 1, False]
24    [17, 20, 23] 1  18872 models.common.YOLODetect          [2, [[10, 13, 16, 30, 33]], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [128, 256, 512]]

Model Summary: 283 layers, 7066239 parameters, 7106239 gradients, 16.5 GFLOPS

Transferred 356/382 Mtores from ./weights/yolov5s.pt
Scaled weight decay = 0.00046875
optimizer groups: 62 .bias, 62 conv.weight, 59 other
Data Scanning: /home/jetson/software/yolov5-5.0/data/garbage/train/labels.cache' Inages and labels... 100 found, 0 missing, 0 empty, 0 corrupted: 100% | 100/100 [00:00<, ?it/s]
Labels Scanning: /home/jetson/software/yolov5-5.0/data/garbage/train/labels.cache' Inages and labels... 100 found, 0 missing, 0 empty, 0 corrupted: 100% | 100/100 [00:00<, ?it/s]
Plotting labels...

AutoAnchor: Analyzing anchors... anchors/target = 4.19, Best Possible Recall (BPR) = 1.0000
Epochs: 44/50 done, 66 test
```

```
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')
```

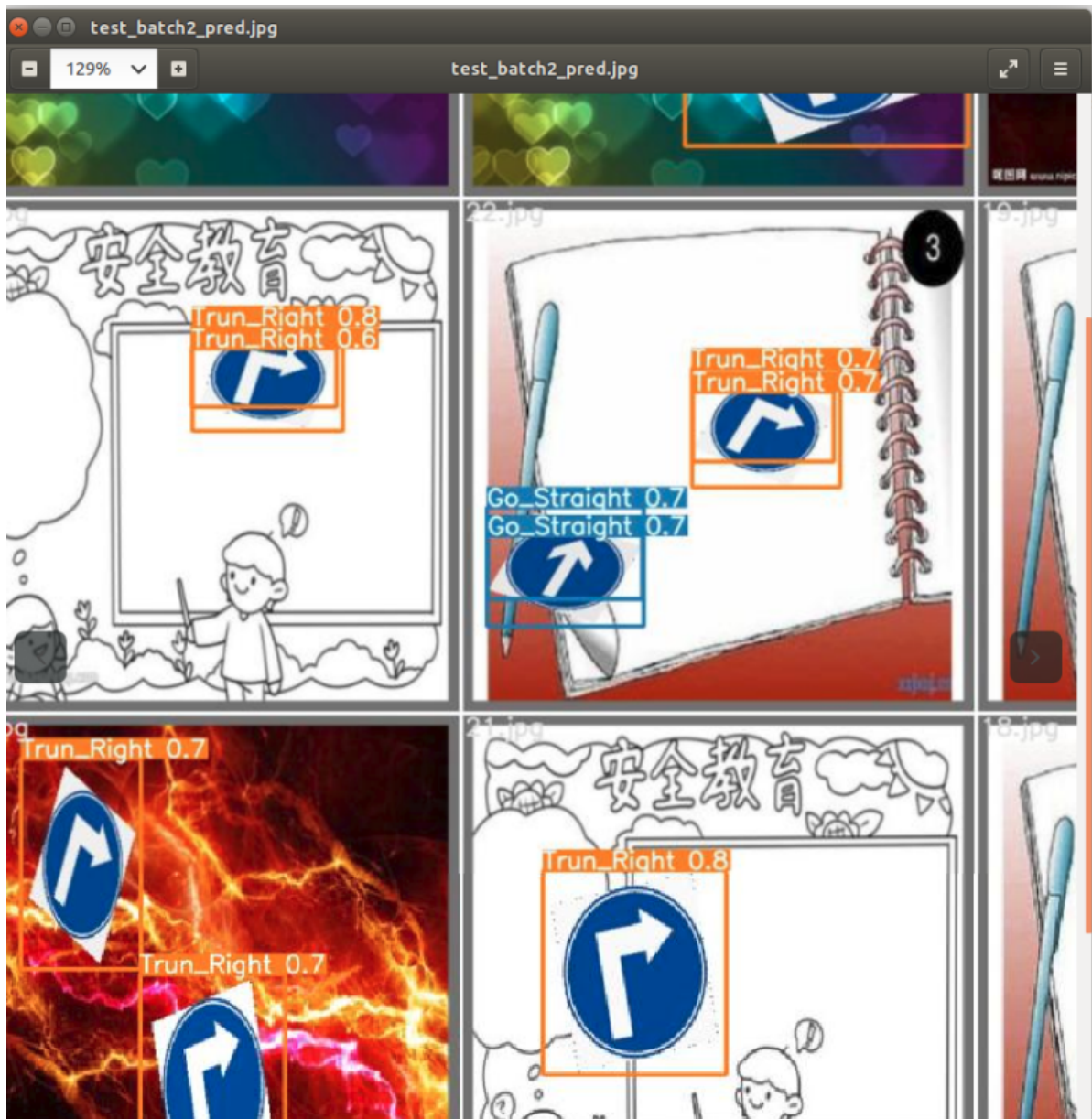
```

60_Straight      100      78      0.863      0.974      0.858      0.964
Trun_Right      100      72      0.863      0.972      0.963      0.603
50 epochs completed in 0.161 hours.

Optimizer stripped from runs/train/exp17/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp17/weights/best.pt, 14.4MB
jetson@SSD:~/software/yolov5-5.0$

```

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,

