Model training

yolov5 official tutorial: https://github.com/ultralytics/yolov5/blob/master/tutorial.ipynb

yolov5 official source code: https://github.com/ultralytics/yolov5

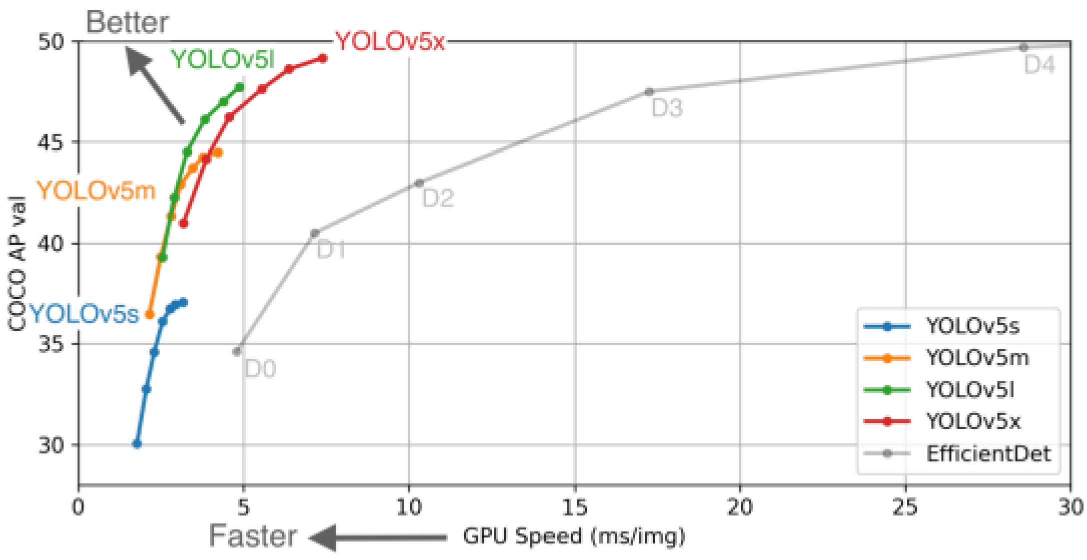
yolov5 weights file: https://github.com/ultralytics/yolov5/releases

This tutorial is only applicable to Jetson Orin NX and Jetson Orin Nano boards.

The Jetson Nano board cannot be used for model training due to its limited performance.

1. yolov5 introduction

Yolov5 algorithm performance test chart



Video test official case

```
cd ~/software/yolov5-5.0
python3 detection_video.py
```

2. Folder structure

```
.
├── data
|   ├── argoverse_hd.yaml
|   ├── coco128.yaml
|   ├── coco.yaml
|   ├── garbage
|   |   ├── Get_garbageData.py           # Automatically generate the
corresponding data set py file (need more)
|   |   ├── image                        # The folder where the original image
(target image to be recognized) is stored
|   |   ├── images                       # Test input picture picture folder
|   |   ├── texture                      # Folder for storing background images
(more needed)
|   |   └── train
|   |       ├── images                   # The image folder where the dataset is
stored
|   |       ├── labels                   # The label folder where the dataset is
stored
|   |       └── labels.cache
|   ├── garbage.yaml
|   ├── hyp.finetune.yaml
|   ├── hyp.scratch.yaml
|   ├── images                           # Test input picture
|   |   ├── bus.jpg
|   |   └── zidane.jpg
|   ├── scripts
|   |   ├── get_argoverse_hd.sh
|   |   ├── get_coco.sh                  # Download the dataset script
|   |   └── get_voc.sh
|   └── voc.yaml
├── detect.py                            # Detection file
├── Dockerfile
├── hubconf.py
├── LICENSE
├── models                               # Model configuration files
|   ├── yolo.py
|   ├── yolov5l.yaml
|   ├── yolov5m.yaml
|   ├── yolov5s.yaml
|   └── yolov5x.yaml
├── README.md
├── requirements.txt                     # Environmental requirements
├── runs
|   ├── detect
|   |   └── exp                          # Test output image path folder
|   |       ├── bus.jpg
|   |       └── zidane.jpg
|   └── train
|       └── exp
|           ├── results.png              # Training result pictures
|           ├── results.txt              # Training log information
|           ├── test_batch0_labels.jpg
|           ├── test_batch0_pred.jpg     # Predicted pictures
|           ├── test_batch1_labels.jpg
|           ├── test_batch1_pred.jpg
|           ├── test_batch2_labels.jpg
|           ├── test_batch2_pred.jpg
|           ├── train_batch0.jpg         # Training pictures
|           ├── train_batch1.jpg
|           ├── train_batch2.jpg
|           └── weights
|               ├── best.pt              # The best training model (usually
selected)
|               └── last.pt              # The last trained model
├── test.py                              # Test files
├── train.py                             # training files
├── tutorial.ipynb                       # Tutorials
├── utils
```

```
└── weights                                 # Folder for storing weight files (pre-
trained models)
    ├── download_weights.sh
    └── yolov5s.pt
```

Note: There are verification images in the runs/exp file.

Check them after training an epoch to verify whether the images are correct. If they are not correct, adjust them immediately. Otherwise, you will have to retrain after all adjustments are made after training.

Label format

```
label  x y w h
```

The values of x, y, w, and h are the positions where the length and height of the image are [0:1]

3. Environmental requirements

The following contents are already configured in the Yahboom image file and do not need to be installed.

```
# pip install -r requirements.txt

# base ----------------------------------------
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python>=4.1.2
Pillow
PyYAML>=5.3.1
scipy>=1.4.1
torch>=1.7.0
torchvision>=0.8.1
tqdm>=4.41.0
imgaug

# logging -------------------------------------
tensorboard>=2.4.1
# wandb

# plotting ------------------------------------
seaborn>=0.11.0
pandas

# export --------------------------------------
# coremltools>=4.1
# onnx>=1.8.1
# scikit-learn==0.19.2  # for coreml quantization

# extras --------------------------------------
thop  # FLOPS computation
pycocotools>=2.0  # COCO mAP
```

Input following command to install

```
sudo pip3 install imgaug
```

4. Use step

- After completing the configuration according to the above process
- Fill more background images in the path [data/garbage/texture] ([more])
- Run the [Get_garbageData.py] file to obtain the data set (line 156 can modify the number of generated data sets [more])
- Run the [train.py] file to start training
- Run [detect.py] for image prediction

5. Custom training dataset

5.1 Collecting the dataset

First, go to Baidu to download or use other methods, and fill in more background images in the path [data/garbage/texture]

Open the [Get_garbageData.py] file

```
sudo vim Get_garbageData.py
```

Modify the total number of generated data sets and fill it in according to your needs.

More data sets are needed, and too few will lead to incorrect training results.

```
img_total=10000
```

Run the [Get_garbageData.py] file to obtain the data set

```
python Get_garbageData.py
```

5.2 Make yaml fiel

Eg garbage.yaml:

```
train: data/garbage/train/images/        # Training set image path
val:   data/garbage/train/images/        # Verification set image path (can also
be separated from the training set)
nc: 16                                   # Categories number
names: ["Zip_top_can", "Apple_core"...]  # Classification name
```

5.3 Modify train.py

```
parser.add_argument('--weights', type=str, default='./weights/yolov5s.pt',
help='initial weights path') # Line 458: Pretrained weights
parser.add_argument('--data', type=str, default='./garbage/garbage.yaml',
help='data.yaml path')        # Line 460: Custom training file
parser.add_argument('--epochs', type=int, default=10)
           # Line 462: Customize training epochs, how many rounds to train
parser.add_argument('--batch-size', type=int, default=12, help='total batch size
for all GPUs')         # The total batch size for all GPUs. If an error occurs,
you can reduce the value
parser.add_argument('--img-size', nargs='+', type=int, default=[640, 640],
help='[train, test] image sizes')    # Image size
parser.add_argument('--device', default='0', help='cuda device, i.e. 0 or
0,1,2,3 or cpu')            # Line 474: Select CPU or GPU
parser.add_argument('--project', default='runs/train', help='save to
project/name')                     # Training result output folder
```

Parameters in other places can be modified according to your needs.

5.4 Modify the model configuration file

Modify the second line of the yaml file of the yolov5 neural network, and use which weight file to modify the corresponding yaml file.

Here we use yolov5s.yaml, so just modify the second line of the models/yolov5s.yaml file

```
nc: 16  # number of classes
```

5.5 Modify detect.py

It is similar to the place where the [train.py] file needs to be modified

```
parser.add_argument('--weights', nargs='+', type=str,
default='weights/yolov5s.pt', help='model.pt path(s)') # Line 151: Pretrained
weights
parser.add_argument('--source', type=str, default='data/images', help='source')
 # file/folder, 0 for webcam # Line 152: Input prediction image
parser.add_argument('--img-size', type=int, default=640, help='inference size
(pixels)')                    # Line 153: Image size
parser.add_argument('--device', default='0', help='cuda device, i.e. 0 or
0,1,2,3 or cpu')            # Line 156: Select CPU or GPU
parser.add_argument('--project', default='runs/detect', help='save results to
project/name')           # Line 165: Prediction result output folder
```

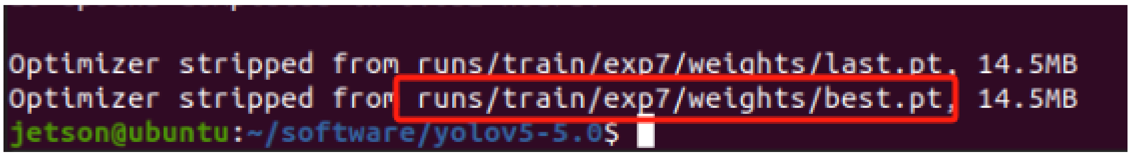
Parameters in other places can be modified according to your needs.

5.6 Training and prediction

Note: Before training the model, please close the docker container and other programs that occupy a large amount of memory to avoid memory.

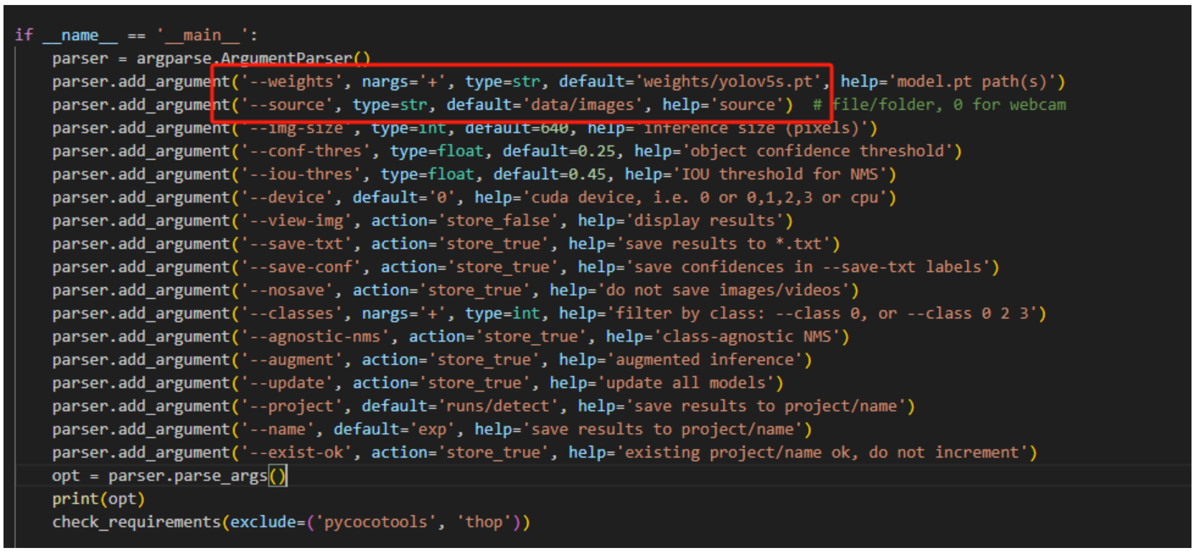
Train the model. After the training is completed, the final model will be generated in the [runs/train] folder.

```
python3 train.py
```



```
Optimizer stripped from runs/train/exp7/weights/last.pt, 14.5MB
Optimizer stripped from runs/train/exp7/weights/best.pt, 14.5MB
jetson@ubuntu:~/software/yolov5-5.0$
```

For image prediction, you need to manually modify the model path and input image, predict the image, and the result is in the [runs/detect] folder.



```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default='weights/yolov5s.pt', help='model.pt path(s)')
    parser.add_argument('--source', type=str, default='data/images', help='source')  # file/folder, 0 for webcam
    parser.add_argument('--img-size', type=int, default=640, help='inference size (pixels)')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='object confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='IOU threshold for NMS')
    parser.add_argument('--device', default='0', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_false', help='display results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in --save-txt labels')
    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --class 0, or --class 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default='runs/detect', help='save results to project/name')
    parser.add_argument('--name', default='exp', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
    opt = parser.parse_args()
    print(opt)
    check_requirements(exclude=('pycocotools', 'thop'))
```

```
python3 detect.py
```

For real-time video monitoring, the model path needs to be modified.

```
python3 detection_video.py
```