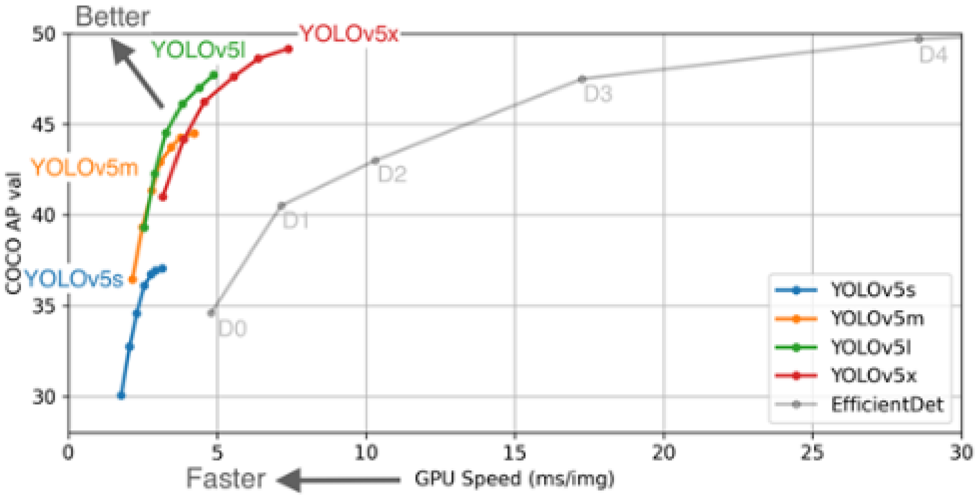
# Model training

## 1.Introduction

Yolov5 author's algorithm performance test chart:



## 2. Environmental requirements

You don't need to reinstall these software to use the virtual machine we provide, just run the program directly.

```
Cython
 matplotlib>=3.2.2
 numpy>=1.18.5
 opencv-python>=4.1.2
 pillow
 PyYAML>=5.3
 scipy>=1.4.1
 tensorboard>=2.2
 torch>=1.6.0
 torchvision>=0.7.0
 tqdm>=4.41.0
 imgaug
```

## 3. Model training

**Using the virtual machine we provide does not require retraining, simply skip it.**
**This tutorial is intended for users of secondary development (you can refer to the official YOLOv5 training tutorial if you need a certain foundation)**

yolov5 Official Tutorial: [https://github.com/ultralytics/yolov5/blob/master/tutorial.ipynb](https://github.com/ultralytics/yolov5/blob/master/tutorial.ipynb)

yolov5 Official source code: [https://github.com/ultralytics/yolov5](https://github.com/ultralytics/yolov5)

yolov5 Weight file: [https://github.com/ultralytics/yolov5/releases](https://github.com/ultralytics/yolov5/releases)

(1) Introduction to folder structure

Data/garage/train/images: storing images of the dataset

Data/garage/train/labels: Store labels for the dataset

```
Label format: label x y w h
x. The values of y, w, and h are the positions where the length and height of
the image are placed at [0:1]
```

inference/images: Test image path

inference/output: Test output path

data/image: Store the original image (target image to be recognized)

data/context: Store background images (more needed)

data/Get_ GarbagData.py: Automatically generate corresponding datasets (requires more)
The weights folder is used to store weight files (pre trained models). If you need the latest version or other model files, please download them from the official website

runs: Training output position

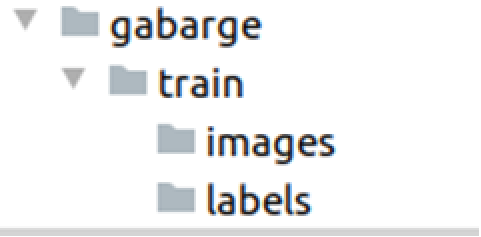
runs/exp/weights: If there are two files, best. pt is the best training model. If there is only one file, there is no choice

There are validation images in the runs/exp file. After training for an epoch, go and check if the images are correct. If they are not correct, adjust them quickly.

Otherwise, if you adjust them after training, you will have to retrain/ Runs/exp1/results. txt: Training log information

(2)Training steps

- Create your own dataset in the following format:



```
garpage/train/images store images of the dataset
garpage/train/labels store labels for datasets
```

Note that the names of the images and label files should correspond, and the format of the label file is category center point x, center point y, border width w, border height h.

The label format is as follows:

```
label  x y w h
14 0.5192307692307692   0.3425480769230769 0.4423076923076923
0.37259615384615385
```

- Create your own garbage.yaml file

```
# train and val data
train: ./garbage/train/images/
val: ./garbage/train/images/
# number of classes
nc: 16
# class names
names: ["Zip_top_can", "Old_school_bag", "Newspaper", "Book", "Toilet_paper",
   "Peach_pit", "Cigarette_butts", "Disposable_chopsticks", "Egg_shell",
   "Apple_core", "Watermelon_rind", "Fish_bone", "Expired_tablets",
   "Expired_cosmetics",  "Used_batteries", "Syringe"]
```

- Modify the models/yolov5s.yaml file

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

Found nc: The number of categories matches the custom yaml file

- Parameter configuration related to model training

```
if __name__ == '__main__':
    check_git_status()
    parser = argparse.ArgumentParser()
    parser.add_argument('--epochs', type=int, default=5)
    parser.add_argument('--batch-size', type=int, default=4)
    parser.add_argument('--cfg', type=str, default='models/yolov5s.yaml',
help='*.cfg path')
    parser.add_argument('--data', type=str, default='data/coco128.yaml',
help='*.data path')
    parser.add_argument('--img-size', nargs='+', type=int, default=[640, 640],
help='train,test sizes')
    parser.add_argument('--rect', action='store_true', help='rectangular
training')
    parser.add_argument('--resume', action='store_true', help='resume training
from last.pt')
    parser.add_argument('--nosave', action='store_true', help='only save final
checkpoint')
    parser.add_argument('--notest', action='store_true', help='only test final
epoch')
    parser.add_argument('--noautoanchor', action='store_true', help='disable
autoanchor check')
    parser.add_argument('--evolve', action='store_true', help='evolve
hyperparameters')
    parser.add_argument('--bucket', type=str, default='', help='gsutil bucket')
    parser.add_argument('--cache-images', action='store_true', help='cache
images for faster training')
    parser.add_argument('--weights', type=str, default='', help='initial weights
path')
    parser.add_argument('--name', default='', help='renames results.txt to
results_name.txt if supplied')
    parser.add_argument('--device', default='0', help='cuda device, i.e. 0 or
0,1,2,3 or cpu')
    parser.add_argument('--adam', action='store_true', help='use adam
optimizer')
```

```
    parser.add_argument('--multi-scale', action='store_true', help='vary img-size +/- 50%')
    parser.add_argument('--single-cls', action='store_true', help='train as single-class dataset')
    opt = parser.parse_args()
```

epochs: Refers to how many times the entire dataset will be iterated during the training process. If the graphics card is not working, you can adjust it down a bit.

batch size: How many images do you need to view at once before updating the weight? For mini batch with gradient descent, if the graphics card doesn't work, you can adjust it down a bit.

cfg: Configuration file for storing model structure

data: A file that stores training and testing data

img size: Input the width and height of the image. If the graphics card doesn't work, you can adjust it down a bit.

rect: Perform rectangular training

resume: Restore the recently saved model and start training

nosave: Only save the final checkpoint

note: Only test the last epoch

evolve: Evolutionary hyperparameters

bucket: gsutil bucket

cache images: cache images to speed up training

weights: weight file path

name: Renaming results. txt to results_ Name.txt

device: cuda device, i.e. 0 or 0,1,2,3 or CPU

adam: Optimizing with Adam

multi scale: Multi scale training, img size+/-50%

single cls: a single class training set

- Open the terminal, execute the command, and start training

```
python train.py --img 416 --batch 16 --epochs 300 --data ./data/garbage.yaml --cfg ./models/yolov5s.yaml --weights yolov5s.pt
```

(3)Custom model testing

```
python detect.py --source 0  # webcam
            file.jpg  # image
            file.mp4  # video
            path/  # directory
            path/*.jpg  # glob
```