1. Yolo v5

1.安装和配置

<https://github.com/ultralytics/yolov5/tree/v5.0>

用的代码是yolo5的版本

Github的下载好之后，pip install -r requirements.txt

解压用pycharm进入到根目录下，配置python的版本要在3.6及3.6以上

2.调参

2.1.测试方法（命令行的测试方法）

通过命令行的方式进行运行（命令行的测试方法）

python detect.py --source 0 # webcam

file.jpg # image 图片

file.mp4 # video 视频

path/ # directory 文件夹

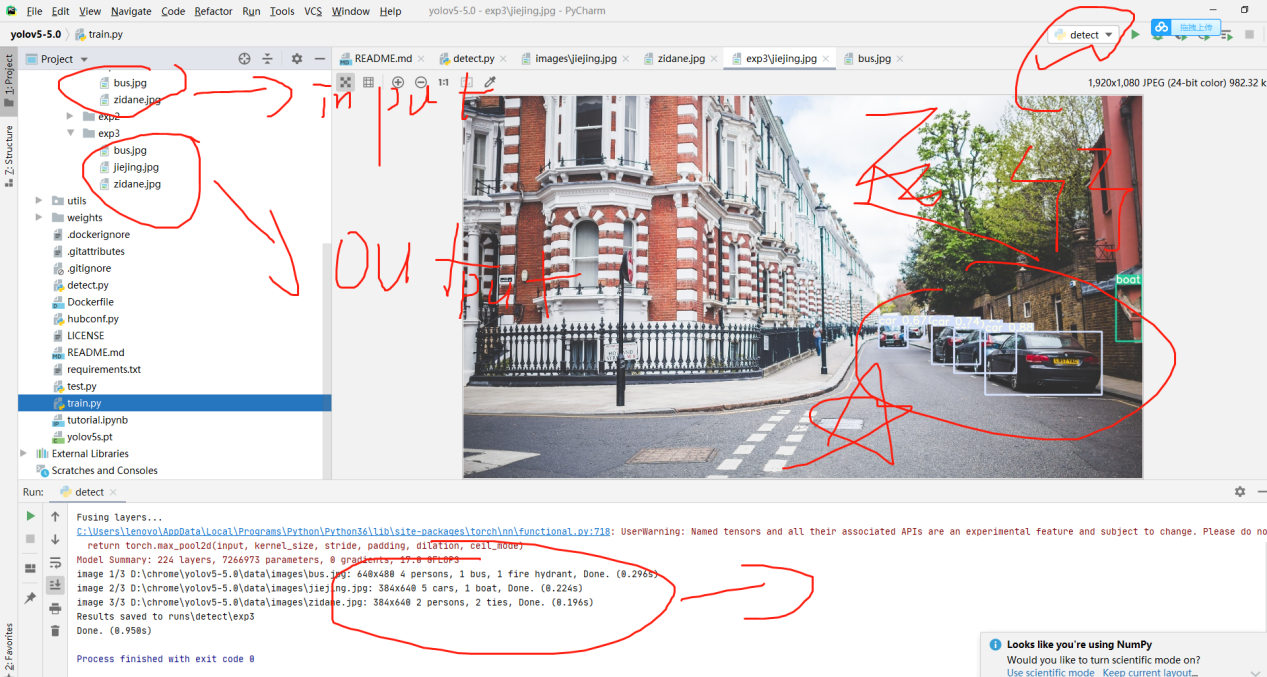
path/\*.jpg # glob 模糊

'https://youtu.be/NUsoVlDFqZg' # YouTube video

'rtsp://example.com/media.mp4' # RTSP, RTMP, HTTP stream 手机摄像头之类的rtsp链接（实时监测）

2.2（pycharm的测试方法）

可以直接运行detect.py



**2.3调参文件detect.py**

import argparse  
import time  
from pathlib import Path  
  
import cv2  
import torch  
import torch.backends.cudnn as cudnn  
from numpy import random  
  
from models.experimental import attempt\_load  
from utils.datasets import LoadStreams, LoadImages  
from utils.general import check\_img\_size, check\_requirements, check\_imshow, non\_max\_suppression, apply\_classifier, \  
 scale\_coords, xyxy2xywh, strip\_optimizer, set\_logging, increment\_path  
from utils.plots import plot\_one\_box  
from utils.torch\_utils import select\_device, load\_classifier, time\_synchronized  
  
  
def detect(save\_img=False):  
 source, weights, view\_img, save\_txt, imgsz = opt.source, opt.weights, opt.view\_img, opt.save\_txt, opt.img\_size  
 save\_img = not opt.nosave and not source.endswith('.txt') # save inference images  
 webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswith(  
 ('rtsp://', 'rtmp://', 'http://', 'https://'))  
  
 # Directories  
 save\_dir = Path(increment\_path(Path(opt.project) / opt.name, exist\_ok=opt.exist\_ok)) # increment run  
 (save\_dir / 'labels' if save\_txt else save\_dir).mkdir(parents=True, exist\_ok=True) # make dir  
  
 # Initialize  
 set\_logging()  
 device = select\_device(opt.device)  
 half = device.type != 'cpu' # half precision only supported on CUDA  
  
 # Load model  
 model = attempt\_load(weights, map\_location=device) # load FP32 model  
 stride = int(model.stride.max()) # model stride  
 imgsz = check\_img\_size(imgsz, s=stride) # check img\_size  
 if half:  
 model.half() # to FP16  
  
 # Second-stage classifier  
 classify = False  
 if classify:  
 modelc = load\_classifier(name='resnet101', n=2) # initialize  
 modelc.load\_state\_dict(torch.load('weights/resnet101.pt', map\_location=device)['model']).to(device).eval()  
  
 # Set Dataloader  
 vid\_path, vid\_writer = None, None  
 if webcam:  
 view\_img = check\_imshow()  
 cudnn.benchmark = True # set True to speed up constant image size inference  
 dataset = LoadStreams(source, img\_size=imgsz, stride=stride)  
 else:  
 dataset = LoadImages(source, img\_size=imgsz, stride=stride)  
  
 # Get names and colors  
 names = model.module.names if hasattr(model, 'module') else model.names  
 colors = [[random.randint(0, 255) for \_ in range(3)] for \_ in names]  
  
 # Run inference  
 if device.type != 'cpu':  
 model(torch.zeros(1, 3, imgsz, imgsz).to(device).type\_as(next(model.parameters()))) # run once  
 t0 = time.time()  
 for path, img, im0s, vid\_cap in dataset:  
 img = torch.from\_numpy(img).to(device)  
 img = img.half() if half else img.float() # uint8 to fp16/32  
 img /= 255.0 # 0 - 255 to 0.0 - 1.0  
 if img.ndimension() == 3:  
 img = img.unsqueeze(0)  
  
 # Inference  
 t1 = time\_synchronized()  
 pred = model(img, augment=opt.augment)[0]  
  
 # Apply NMS  
 pred = non\_max\_suppression(pred, opt.conf\_thres, opt.iou\_thres, classes=opt.classes, agnostic=opt.agnostic\_nms)  
 t2 = time\_synchronized()  
  
 # Apply Classifier  
 if classify:  
 pred = apply\_classifier(pred, modelc, img, im0s)  
  
 # Process detections  
 for i, det in enumerate(pred): # detections per image  
 if webcam: # batch\_size >= 1  
 p, s, im0, frame = path[i], '%g: ' % i, im0s[i].copy(), dataset.count  
 else:  
 p, s, im0, frame = path, '', im0s, getattr(dataset, 'frame', 0)  
  
 p = Path(p) # to Path  
 save\_path = str(save\_dir / p.name) # img.jpg  
 txt\_path = str(save\_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else f'\_{frame}') # img.txt  
 s += '%gx%g ' % img.shape[2:] # print string  
 gn = torch.tensor(im0.shape)[[1, 0, 1, 0]] # normalization gain whwh  
 if len(det):  
 # Rescale boxes from img\_size to im0 size  
 det[:, :4] = scale\_coords(img.shape[2:], det[:, :4], im0.shape).round()  
  
 # Print results  
 for c in det[:, -1].unique():  
 n = (det[:, -1] == c).sum() # detections per class  
 s += f"{n} {names[int(c)]}{'s' \* (n > 1)}, " # add to string  
  
 # Write results  
 for \*xyxy, conf, cls in reversed(det):  
 if save\_txt: # Write to file  
 xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist() # normalized xywh  
 line = (cls, \*xywh, conf) if opt.save\_conf else (cls, \*xywh) # label format  
 with open(txt\_path + '.txt', 'a') as f:  
 f.write(('%g ' \* len(line)).rstrip() % line + '\n')  
  
 if save\_img or view\_img: # Add bbox to image  
 label = f'{names[int(cls)]} {conf:.2f}'  
 plot\_one\_box(xyxy, im0, label=label, color=colors[int(cls)], line\_thickness=3)  
  
 # Print time (inference + NMS)  
 print(f'{s}Done. ({t2 - t1:.3f}s)')  
  
 # Stream results  
 if view\_img:  
 cv2.imshow(str(p), im0)  
 cv2.waitKey(1) # 1 millisecond  
  
 # Save results (image with detections)  
 if save\_img:  
 if dataset.mode == 'image':  
 cv2.imwrite(save\_path, im0)  
 else: # 'video' or 'stream'  
 if vid\_path != save\_path: # new video  
 vid\_path = save\_path  
 if isinstance(vid\_writer, cv2.VideoWriter):  
 vid\_writer.release() # release previous video writer  
 if vid\_cap: # video  
 fps = vid\_cap.get(cv2.CAP\_PROP\_FPS)  
 w = int(vid\_cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))  
 h = int(vid\_cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))  
 else: # stream  
 fps, w, h = 30, im0.shape[1], im0.shape[0]  
 save\_path += '.mp4'  
 vid\_writer = cv2.VideoWriter(save\_path, cv2.VideoWriter\_fourcc(\*'mp4v'), fps, (w, h))  
 vid\_writer.write(im0)  
  
 if save\_txt or save\_img:  
 s = f"\n{len(list(save\_dir.glob('labels/\*.txt')))} labels saved to {save\_dir / 'labels'}" if save\_txt else ''  
 print(f"Results saved to {save\_dir}{s}")  
  
 print(f'Done. ({time.time() - t0:.3f}s)')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':

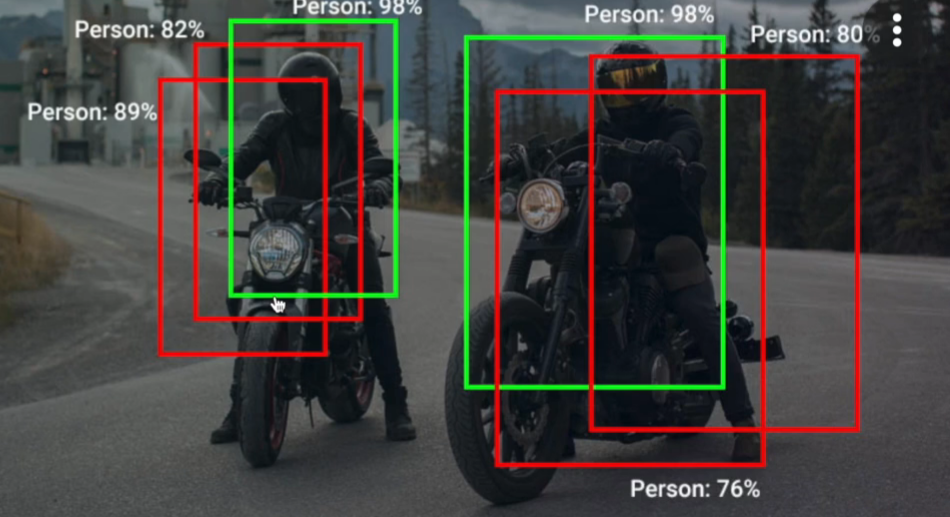
**#可以改的地方**  
 parser = argparse.ArgumentParser()  
 **#比较小的那个模型 yolov5s.pt yolov5m.pt yolov5l.pt yolov5x.pt,可以在github的yolo下面下载，这些模型用于**default=**640会好一点，yolov5s6.pt yolov5m6.pt yolov5l6.pt yolov5x6用于1280**  
 parser.add\_argument('--weights', nargs='+', type=str, default='**yolov5s.pt**', help='model.pt path(s)')

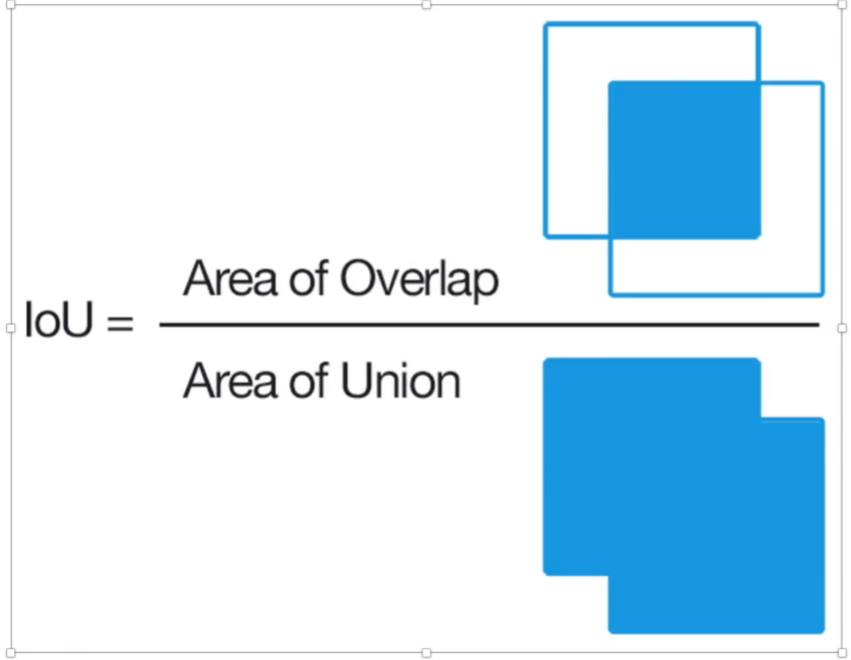
**#输入，可以是图片，可以是视频，可以是rtsp链接**  
 parser.add\_argument('--source', type=str, default='data/images', help='source') # file/folder, 0 for webcam

**#输入输出大小是保持原样的，只会在训练的时候resize**  
 parser.add\_argument('--img-size', type=int, default=**640**, help='inference size (pixels)')  
 **#置信程度，只有大于这个值才会显示出来，如果设置成0的话，那么图片就会有贼多个标签了**

parser.add\_argument('--conf-thres', type=float, default=0.25, help='object confidence threshold')

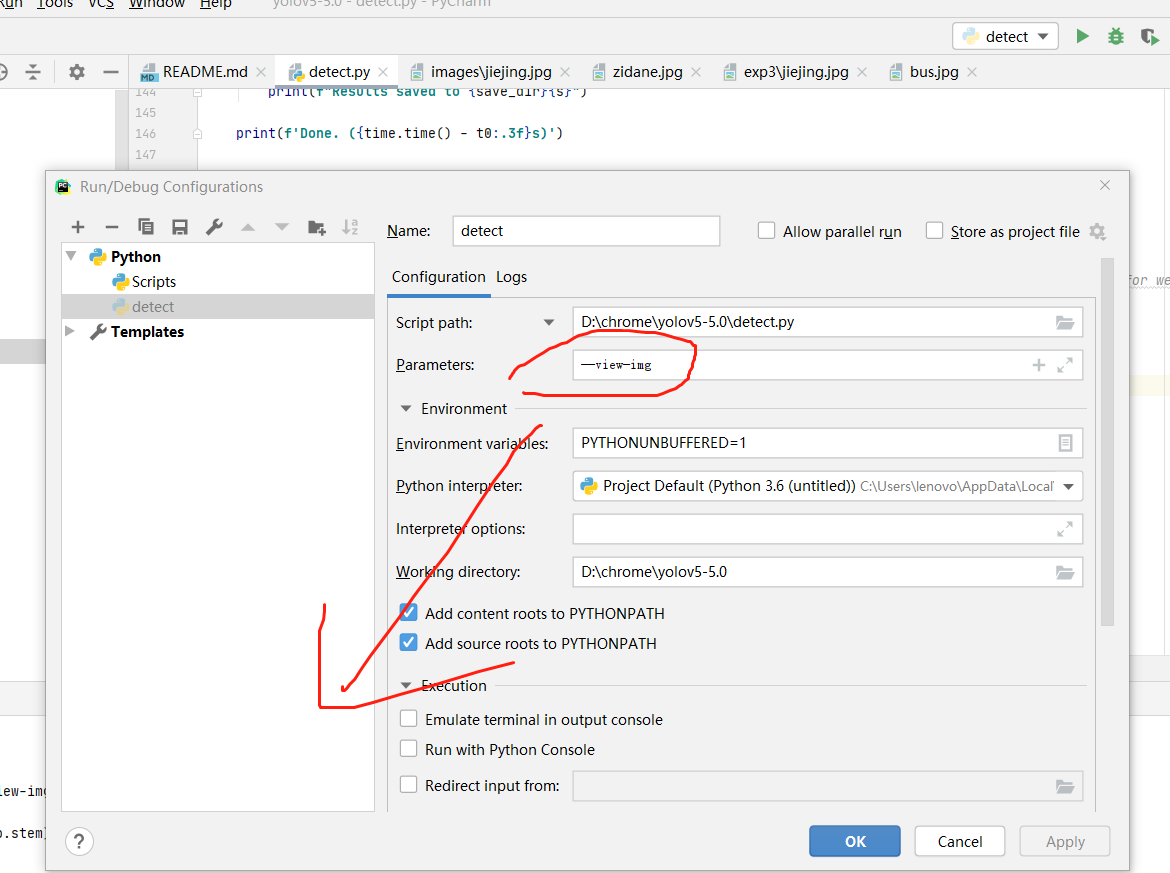
**#NMS：当一个图形识别成了多个相同的标签 如下图，就只会取相似度最大的那一个框**

  
 **#iou：交集 并集 为0完全没有交集随便就合并了，为1完全重合才合并，这种情况有很多的框**

****

parser.add\_argument('--iou-thres', type=float, default=0.45, help='IOU threshold for NMS')  
 parser.add\_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')

**#通过调用可以显示结果调用代码：python detect.py --view-img train的是图片就会训练图片，train的是视频就会显示视频**  
 parser.add\_argument('--view-img', action='store\_true', help='display results')

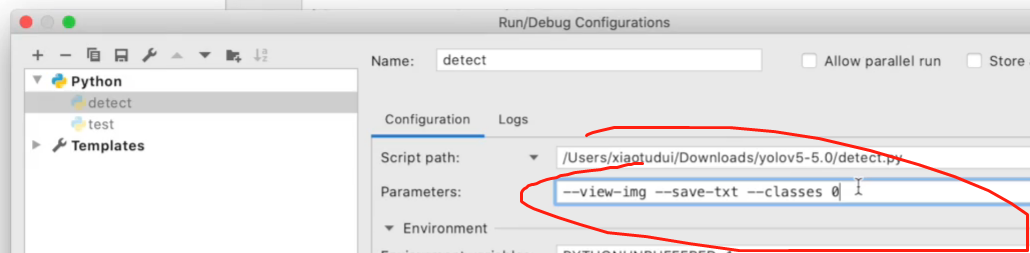
  
 parser.add\_argument('--save-txt', action='store\_true', help='save results to \*.txt')  
 parser.add\_argument('--save-conf', action='store\_true', help='save confidences in --save-txt labels')

**#字面意思：不保存视频**  
 parser.add\_argument('--nosave', action='store\_true', help='do not save images/videos')

**#-classes 0 就只显示0的标签**  
 parser.add\_argument('--classes', nargs='+', type=int, help='filter by class: --class 0, or --class 0 2 3')

**#增强的nms**

parser.add\_argument('--agnostic-nms', action='store\_true', help='class-agnostic NMS')  
 **#增强的目标检测**



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'))  
  
 with torch.no\_grad():  
 if opt.update: # update all models (to fix SourceChangeWarning)  
 for opt.weights in ['yolov5s.pt', 'yolov5m.pt', 'yolov5l.pt', 'yolov5x.pt']:  
 detect()  
 strip\_optimizer(opt.weights)  
 else:  
 detect()

**2.4调参文件Train.py**

if \_\_name\_\_ == '\_\_main\_\_':

**#**  
 parser = argparse.ArgumentParser()

**#这行代码可以把之前训练好的代码加进来，如果是重新训练模型的话，这里的值设成空就好了。可以写yolov5s.pt yolov5m yolov5l.pt yolov5x.pt**  
 parser.add\_argument('--weights', type=str, default='**yolov5s.pt**', help='initial weights path')

**#模型结构的一些设置，yolov5s.yaml yolov5m.yaml yolov5l.yaml yolov5x.yaml** parser.add\_argument('--cfg', type=str, default='**models/yolov5s.yaml**

', help='model.yaml path')

**#数据集 coco coco128 argoverse\_hd voc**  
 parser.add\_argument('--data', type=str, default='data/coco128.yaml', help='data.yaml path')  
 **#模型超参数的一些设置**data/hyp.scratch.yaml 冲头训练用这个

parser.add\_argument('--hyp', type=str, default='data/hyp.scratch.yaml', help='hyperparameters path')  
 parser.add\_argument('**--epochs**', type=int, default=300)  
 parser.add\_argument('**--batch-size**', type=int, default=16, help='total batch size for all GPUs')  
 parser.add\_argument('--img-size', nargs='+', type=int, default=[640, 640], help='[train, test] image sizes')

**#通过长方形的填充加快训练的速度**

parser.add\_argument('--rect', action='store\_true', help='rectangular training')

**#也是载入以前的数据，从中间开始训练**

parser.add\_argument('--resume', nargs='?', const=True, default=False, help='resume most recent training')  
 parser.add\_argument('--nosave', action='store\_true', help='only save final checkpoint')  
 parser.add\_argument('--notest', action='store\_true', help='only test final epoch')  
 **#锚点遍历，这里不用动他。好处是可以不用一个一个像素点的遍历**

parser.add\_argument('--noautoanchor', action='store\_true', help='disable autoanchor check')  
 **#超参数的训练**

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('--image-weights', action='store\_true', help='use weighted image selection for training')  
 parser.add\_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')  
 parser.add\_argument('--multi-scale', action='store\_true', help='vary img-size +/- 50%%')  
 parser.add\_argument('--single-cls', action='store\_true', help='train multi-class data as single-class')

**#超参数的训练，不用这个的话就会i用随机梯度下降**  
 parser.add\_argument('--adam', action='store\_true', help='use torch.optim.Adam() optimizer')  
 parser.add\_argument('--sync-bn', action='store\_true', help='use SyncBatchNorm, only available in DDP mode')  
 parser.add\_argument('--local\_rank', type=int, default=-1, help='DDP parameter, do not modify')  
 parser.add\_argument('--workers', type=int, default=0, help='maximum number of dataloader workers')  
 parser.add\_argument('--project', default='runs/train', help='save to project/name')  
 parser.add\_argument('--entity', default=None, help='W&B entity')  
 parser.add\_argument('--name', default='exp', help='save to project/name')  
 parser.add\_argument('--exist-ok', action='store\_true', help='existing project/name ok, do not increment')

**#可以在更大的一个尺度会更好**

parser.add\_argument('--quad', action='store\_true', help='quad dataloader')

**#学习速率的调整，开启的话，学习速率就会以线性的方式。不开启的话以一种余弦退火的方式来处理**  
 parser.add\_argument('--linear-lr', action='store\_true', help='linear LR')

**#标签平滑。防止过拟合** parser.add\_argument('--label-smoothing', type=float, default=0.0, help='Label smoothing epsilon')  
 parser.add\_argument('--upload\_dataset', action='store\_true', help='Upload dataset as W&B artifact table')  
 parser.add\_argument('--bbox\_interval', type=int, default=-1, help='Set bounding-box image logging interval for W&B')  
 parser.add\_argument('--save\_period', type=int, default=-1, help='Log model after every "save\_period" epoch')  
 parser.add\_argument('--artifact\_alias', type=str, default="latest", help='version of dataset artifact to be used')  
 opt = parser.parse\_args()  
  
 # Set DDP variables  
 opt.world\_size = int(os.environ['WORLD\_SIZE']) if 'WORLD\_SIZE' in os.environ else 1  
 opt.global\_rank = int(os.environ['RANK']) if 'RANK' in os.environ else -1  
 set\_logging(opt.global\_rank)  
 if opt.global\_rank in [-1, 0]:  
 check\_git\_status()  
 check\_requirements()  
  
 # Resume  
 wandb\_run = check\_wandb\_resume(opt)  
 if opt.resume and not wandb\_run: # resume an interrupted run  
 ckpt = opt.resume if isinstance(opt.resume, str) else get\_latest\_run() # specified or most recent path  
 assert os.path.isfile(ckpt), 'ERROR: --resume checkpoint does not exist'  
 apriori = opt.global\_rank, opt.local\_rank  
 with open(Path(ckpt).parent.parent / 'opt.yaml') as f:  
 opt = argparse.Namespace(\*\*yaml.load(f, Loader=yaml.SafeLoader)) # replace  
 opt.cfg, opt.weights, opt.resume, opt.batch\_size, opt.global\_rank, opt.local\_rank = '', ckpt, True, opt.total\_batch\_size, \*apriori # reinstate  
 logger.info('Resuming training from %s' % ckpt)  
 else:  
 # opt.hyp = opt.hyp or ('hyp.finetune.yaml' if opt.weights else 'hyp.scratch.yaml')  
 opt.data, opt.cfg, opt.hyp = check\_file(opt.data), check\_file(opt.cfg), check\_file(opt.hyp) # check files  
 assert len(opt.cfg) or len(opt.weights), 'either --cfg or --weights must be specified'  
 opt.img\_size.extend([opt.img\_size[-1]] \* (2 - len(opt.img\_size))) # extend to 2 sizes (train, test)  
 opt.name = 'evolve' if opt.evolve else opt.name  
 opt.save\_dir = increment\_path(Path(opt.project) / opt.name, exist\_ok=opt.exist\_ok | opt.evolve) # increment run  
  
 # DDP mode  
 opt.total\_batch\_size = opt.batch\_size  
 device = select\_device(opt.device, batch\_size=opt.batch\_size)  
 if opt.local\_rank != -1:  
 assert torch.cuda.device\_count() > opt.local\_rank  
 torch.cuda.set\_device(opt.local\_rank)  
 device = torch.device('cuda', opt.local\_rank)  
 dist.init\_process\_group(backend='nccl', init\_method='env://') # distributed backend  
 assert opt.batch\_size % opt.world\_size == 0, '--batch-size must be multiple of CUDA device count'  
 opt.batch\_size = opt.total\_batch\_size // opt.world\_size  
  
 # Hyperparameters  
 with open(opt.hyp) as f:  
 hyp = yaml.load(f, Loader=yaml.SafeLoader) # load hyps  
  
 # Train  
 logger.info(opt)  
 if not opt.evolve:  
 tb\_writer = None # init loggers  
 if opt.global\_rank in [-1, 0]:  
 prefix = colorstr('tensorboard: ')  
 logger.info(f"{prefix}Start with 'tensorboard --logdir {opt.project}', view at http://localhost:6006/")  
 tb\_writer = SummaryWriter(opt.save\_dir) # Tensorboard  
 train(hyp, opt, device, tb\_writer)  
  
 # Evolve hyperparameters (optional)  
 else:  
 # Hyperparameter evolution metadata (mutation scale 0-1, lower\_limit, upper\_limit)  
 meta = {'lr0': (1, 1e-5, 1e-1), # initial learning rate (SGD=1E-2, Adam=1E-3)  
 'lrf': (1, 0.01, 1.0), # final OneCycleLR learning rate (lr0 \* lrf)  
 'momentum': (0.3, 0.6, 0.98), # SGD momentum/Adam beta1  
 'weight\_decay': (1, 0.0, 0.001), # optimizer weight decay  
 'warmup\_epochs': (1, 0.0, 5.0), # warmup epochs (fractions ok)  
 'warmup\_momentum': (1, 0.0, 0.95), # warmup initial momentum  
 'warmup\_bias\_lr': (1, 0.0, 0.2), # warmup initial bias lr  
 'box': (1, 0.02, 0.2), # box loss gain  
 'cls': (1, 0.2, 4.0), # cls loss gain  
 'cls\_pw': (1, 0.5, 2.0), # cls BCELoss positive\_weight  
 'obj': (1, 0.2, 4.0), # obj loss gain (scale with pixels)  
 'obj\_pw': (1, 0.5, 2.0), # obj BCELoss positive\_weight  
 'iou\_t': (0, 0.1, 0.7), # IoU training threshold  
 'anchor\_t': (1, 2.0, 8.0), # anchor-multiple threshold  
 'anchors': (2, 2.0, 10.0), # anchors per output grid (0 to ignore)  
 'fl\_gamma': (0, 0.0, 2.0), # focal loss gamma (efficientDet default gamma=1.5)  
 'hsv\_h': (1, 0.0, 0.1), # image HSV-Hue augmentation (fraction)  
 'hsv\_s': (1, 0.0, 0.9), # image HSV-Saturation augmentation (fraction)  
 'hsv\_v': (1, 0.0, 0.9), # image HSV-Value augmentation (fraction)  
 'degrees': (1, 0.0, 45.0), # image rotation (+/- deg)  
 'translate': (1, 0.0, 0.9), # image translation (+/- fraction)  
 'scale': (1, 0.0, 0.9), # image scale (+/- gain)  
 'shear': (1, 0.0, 10.0), # image shear (+/- deg)  
 'perspective': (0, 0.0, 0.001), # image perspective (+/- fraction), range 0-0.001  
 'flipud': (1, 0.0, 1.0), # image flip up-down (probability)  
 'fliplr': (0, 0.0, 1.0), # image flip left-right (probability)  
 'mosaic': (1, 0.0, 1.0), # image mixup (probability)  
 'mixup': (1, 0.0, 1.0)} # image mixup (probability)  
  
 assert opt.local\_rank == -1, 'DDP mode not implemented for --evolve'  
 opt.notest, opt.nosave = True, True # only test/save final epoch  
 # ei = [isinstance(x, (int, float)) for x in hyp.values()] # evolvable indices  
 yaml\_file = Path(opt.save\_dir) / 'hyp\_evolved.yaml' # save best result here  
 if opt.bucket:  
 os.system('gsutil cp gs://%s/evolve.txt .' % opt.bucket) # download evolve.txt if exists  
  
 for \_ in range(300): # generations to evolve  
 if Path('evolve.txt').exists(): # if evolve.txt exists: select best hyps and mutate  
 # Select parent(s)  
 parent = 'single' # parent selection method: 'single' or 'weighted'  
 x = np.loadtxt('evolve.txt', ndmin=2)  
 n = min(5, len(x)) # number of previous results to consider  
 x = x[np.argsort(-fitness(x))][:n] # top n mutations  
 w = fitness(x) - fitness(x).min() # weights  
 if parent == 'single' or len(x) == 1:  
 # x = x[random.randint(0, n - 1)] # random selection  
 x = x[random.choices(range(n), weights=w)[0]] # weighted selection  
 elif parent == 'weighted':  
 x = (x \* w.reshape(n, 1)).sum(0) / w.sum() # weighted combination  
  
 # Mutate  
 mp, s = 0.8, 0.2 # mutation probability, sigma  
 npr = np.random  
 npr.seed(int(time.time()))  
 g = np.array([x[0] for x in meta.values()]) # gains 0-1  
 ng = len(meta)  
 v = np.ones(ng)  
 while all(v == 1): # mutate until a change occurs (prevent duplicates)  
 v = (g \* (npr.random(ng) < mp) \* npr.randn(ng) \* npr.random() \* s + 1).clip(0.3, 3.0)  
 for i, k in enumerate(hyp.keys()): # plt.hist(v.ravel(), 300)  
 hyp[k] = float(x[i + 7] \* v[i]) # mutate  
  
 # Constrain to limits  
 for k, v in meta.items():  
 hyp[k] = max(hyp[k], v[1]) # lower limit  
 hyp[k] = min(hyp[k], v[2]) # upper limit  
 hyp[k] = round(hyp[k], 5) # significant digits  
  
 # Train mutation  
 results = train(hyp.copy(), opt, device)  
  
 # Write mutation results  
 print\_mutation(hyp.copy(), results, yaml\_file, opt.bucket)  
  
 # Plot results  
 plot\_evolution(yaml\_file)  
 print(f'Hyperparameter evolution complete. Best results saved as: {yaml\_file}\n'  
 f'Command to train a new model with these hyperparameters: $ python train.py --hyp {yaml\_file}')

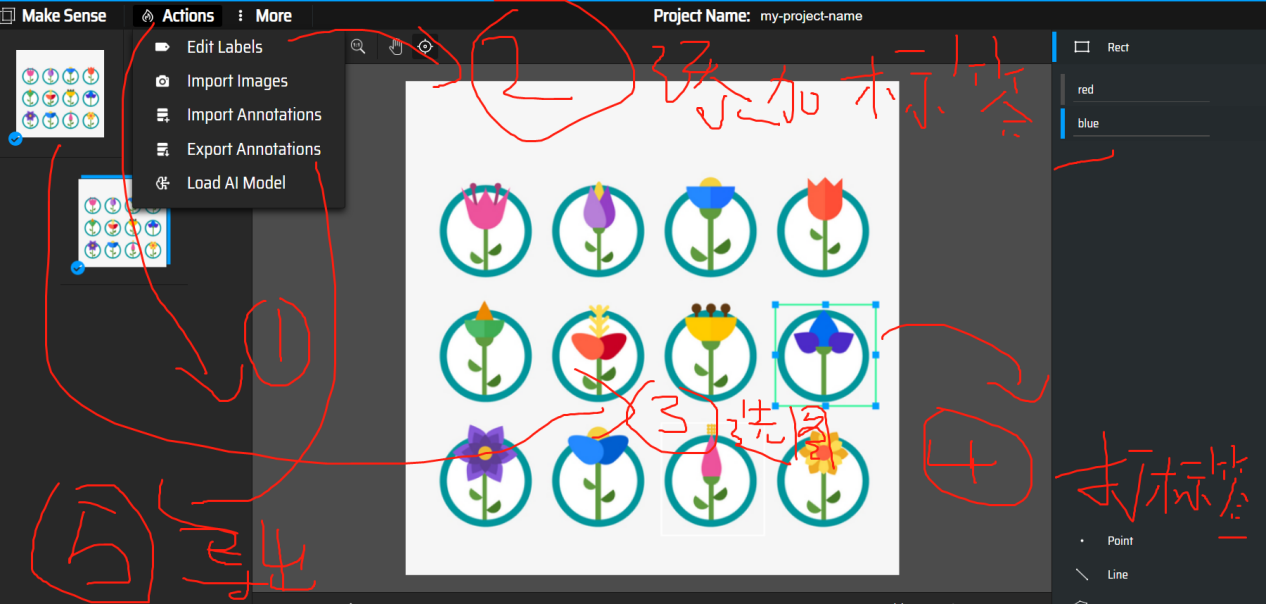
**3.制作自己的数据集与实战**

3.1通过别人的模型detect跑起来

有手就行

3.2 自己训练模型跑起来

打开网站https://www.makesense.ai/

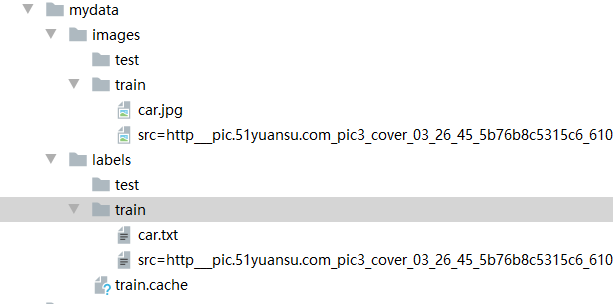


这里可以用load ai 加快检测效率

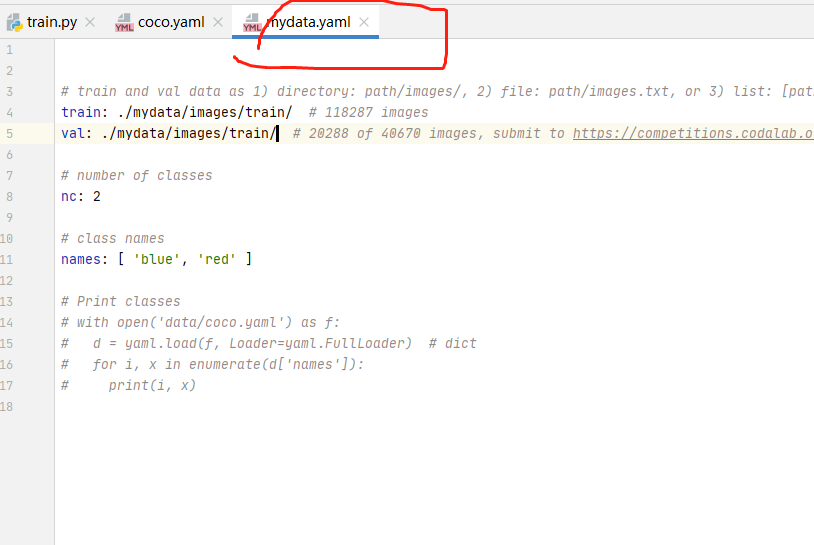
数据结构方面：

导出来的yolo文件是txt格式的，与图片名一一对应

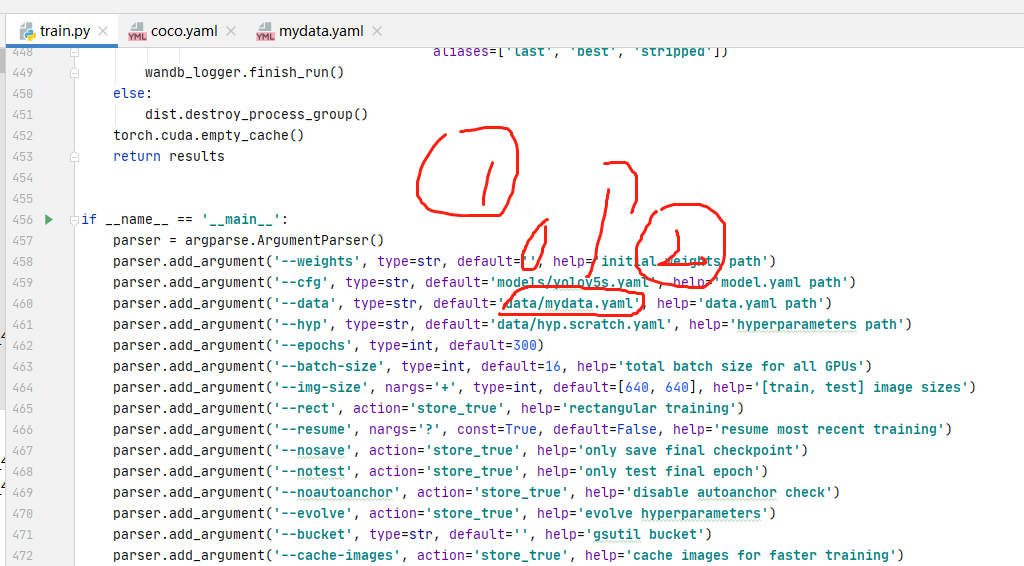
然后在文件的目录下建立如下结构



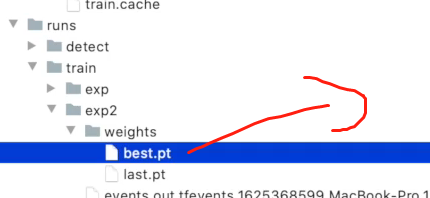
新建一个yaml文件内容差不多如下



Train.py 改变这两个参数还有一个超参数的类别也要设置成一样的  **hyp**



输出的pt文件在中



调用：

改三个地方

改pt文件的位置和训练数据集的位置，马上调用成功

