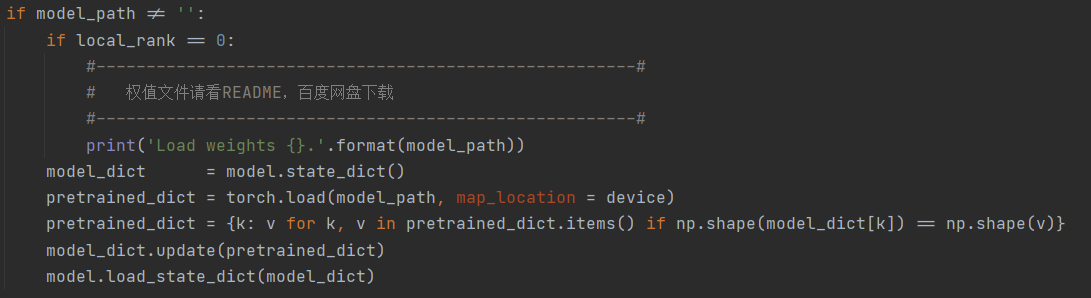
yolo.py这个文件是与训练完之后进行预测用的，与训练过程无关。于是当下载了代码和预训练权重之后，由于预训练权重是在coco数据集上训练出来的，于是yolo.py里面的所要去预测的类别的txt文件只能指向coco\_classes.txt。虽然voc数据集（20个类）里面的类在coco数据集（80个类）里面都有，但是如果修改为voc\_classes.txt就会报错。

载入预训练权重可以预测voc里面没有但是coco里有的类，比如umbrella。（如果只想预测voc里面的类，请在代码里面修改，加上类别判定条件即可。）

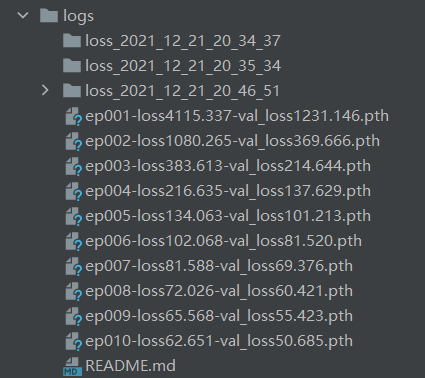
关于使用预训练权重，比如使用coco训练出来的YOLO权重，coco有80各类，而自己的数据集不是80各类，其实只有最后输出的三个特征图在通道数上有区别，其他任何地方都没有区别，所以加载的时候可以通过下面的代码加载匹配的权重：



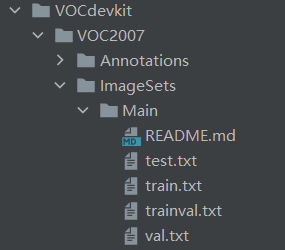
所以，如果采用迁移学习去训练的话，不管是自己的数据集还是voc2007+2012数据集，首先下载代码与预训练权重（注：如果修改了网络结构的话就没办法用预训练权重了），参照以下步骤：

（前提要求：标签必须是VOC格式的，即XML文件。图片必须是jpg格式的。Coco数据集是json格式的标签，可以用程序转成voc格式。其他格式的图片也能转为jpg格式）

1. 首先删除一些东西：
2. logs文件夹下所有权重和日志文件



1. VOCdevkit/VOC2007/ImageSets/Main文件夹下的四个txt文件

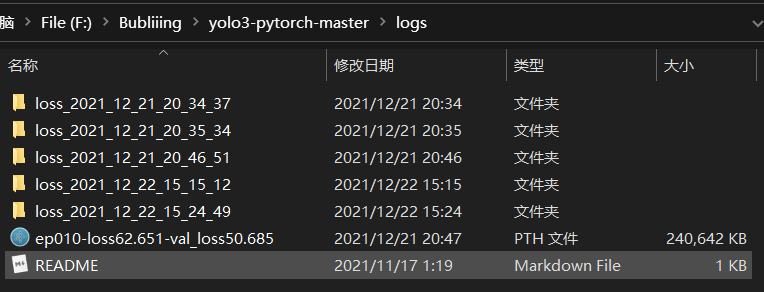


（3）2007\_train.txt和2007\_val.txt文件



（4）map\_out文件夹。因为保存的是上一次测试的结果，**必须删掉**。

1. 用labelimg制作数据集，原图片需要是jpg格式，标签必须是voc格式（XML文件，且格式必须和VOC2007+2012的一样）。可以用python函数将png格式转成jpg格式，函数在F盘的DataSet\_VOC/convert\_img.py
2. 将制作的数据集中的图片（只能是jpg格式）和标签分别放在VOCdevkit\VOC2007文件夹下的JPEGImages文件夹和Anotations文件夹下。
3. 在model\_data文件夹下新建一个txt文件，里面是数据集里面的类别，要竖着写，比如dog、cat、person......。比如命名为my\_classes.txt。
4. 打开voc\_anotations.py文件：
5. 修改annotation\_mode = 0
6. 修改classes\_path = 'model\_data/my\_classes.txt'
7. 运行voc\_anotations.py生成所需txt文件
8. 打开train.py文件:
9. 修改classes\_path = 'model\_data/my\_classes.txt'，这个txt必须和voc\_annotation.py里面的txt一样！。
10. model\_path = 'model\_data/yolo\_weights.pth'是加载别人预训练的权重，一般不修改（要先把这个权重文件下载下来放在对应文件夹下面）
11. input\_shape= [416, 416]，可以根据自己数据集图片的大小修改，比如640，608，320等，但必须是32的倍数
12. 冻结阶段与解冻阶段的参数修改，如epoch、batch\_size、lr等
13. Freeze\_Train为True。利用了迁移学习。
14. num\_workers在Windows下可以设置为4，但训练程序必须在main函数内。
15. 运行train.py进行训练，同时会在logs文件夹下面保存损失函数信息，文件夹里面放的是损失函数（包括训练、测试与验证），以及模型权重。



1. 打开predict.py文件，利用训练好的权重进行预测：
2. 在预测之前，进入YOLO()这个类里面调整一些参数：

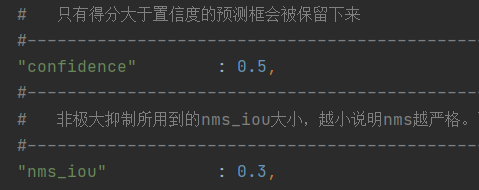
首先是载入自己训练出的权重和指明数据集类别--->

"model\_path" : './logs/ep010-loss62.651-val\_loss50.685.pth',

"classes\_path" : 'model\_data/my\_classes.txt',

注：此处类别的txt必须和训练的train.py文件里的一样。

然后可以调整置信度阈值和nms\_iou阈值--->



1. 返回predict.py，设置model：predict、video、fps等。运行便能得到检测结果

如果没看到结果只看到原图，可能是confidence和nms\_iou设置不当，可以重新调整之后再进行检测。

1. 在训练完之后可以测试mAP：

（1）首先像7(1)一样，在YOLO类里面要修改classes\_path，指向自己的txt文件，

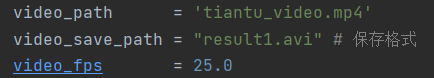
classes\_path = 'model\_data/my\_classes.txt'，以及载入自己训练出来的权重。

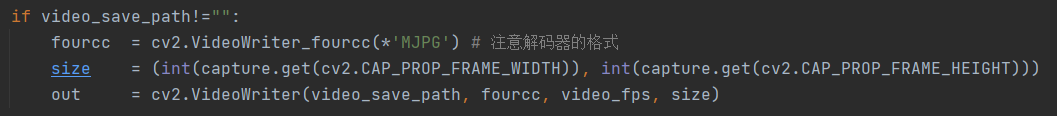
（2）打开get\_map.py文件，可以修改map\_mode为01234，但是若修改为4需要注意下注释。

1. 选择修改：MINCVERLAP、map\_vis等。
2. 运行get\_map.py可以看到结果。
3. 程序运行之后会生成map\_out文件夹并保存结果，比如在result文件夹下面保存了AP、F1、Precision、Recall等。当然如果之前就存在这个文件夹的话，会被清空然后再重新保存结果，所以其实一开始可以将整个map\_out文件夹删掉。

对于训练自己的数据集或者voc数据集，虽然载入了预训练权重，但是是coco的预训练权重（含80个类），所以换到其他数据集，如果选择的训练参数不合适，不如lr，epoch，batch\_size等等，训练出来的模型甚至会检测不到任何物体。

若predict.py中模式选择为video且想保存视频，那么注意保存视频的后缀以及视频解码器的格式选择，如下。





* 如果发现训练效果不佳，比如mAP低，或者精度、召回率低，或者无法检测物体直接返回原图，那么考虑以下几个方面：

（1）目标信息问题，查看2007\_train.txt文件是否有目标信息，没有的话请修改voc\_annotation.py。

（2）数据集问题，小于500的自行考虑增加数据集，同时测试不同的模型，确认数据集是好的。另外不同数据集可能需要不同的anchor，所以可以考虑对数据集进行聚类产生合适的anchor再进行训练

（3）是否解冻训练，如果数据集分布与常规画面差距过大需要进一步解冻训练，调整主干，加强特征提取能力。

（4）网络问题，比如SSD不适合小目标，因为先验框固定了。YOLOv4\_tiny也不适合小目标检测

（5）训练时长问题，有些同学只训练了几代表示没有效果，按默认参数训练完。

（6）确认自己是否按照步骤去做了，如果比如voc\_annotation.py里面的classes是否修改了等。

（8）不同网络的LOSS不同，LOSS只是一个参考指标，用于查看网络是否收敛，而非评价网络好坏，LOSS的值不重要，重要的是是否收敛。

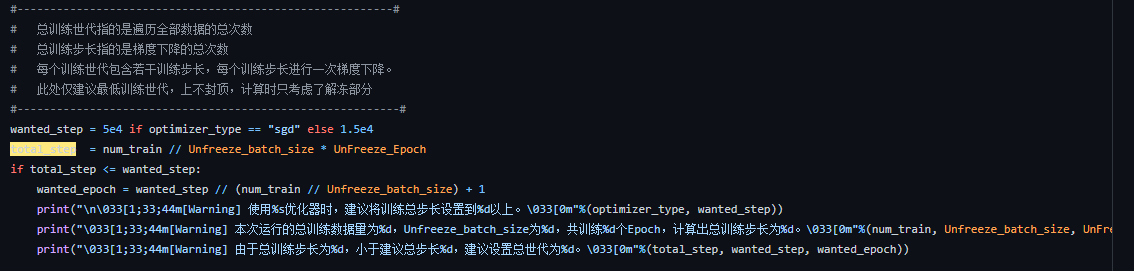
（8）是否修改了网络的主干，如果修改了没有预训练权重，网络不容易收敛，自然效果不好。

（9）数据集分布不均，比如想训练识别人＋车，但是训练集中人的标注远远少于车的标注。

（10）优化器选择的问题，SGD效果比Adam要好，默认用SGD，但是需要训练更多的世代才能收敛。用Adam优化器的话最好不要用权重衰减

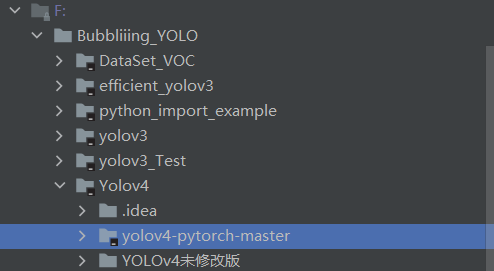
（11）关于训练轮数见YOLOv4里面的train.py代码，见下图

<https://github.com/bubbliiiing/yolov4-pytorch/blob/master/train.py>

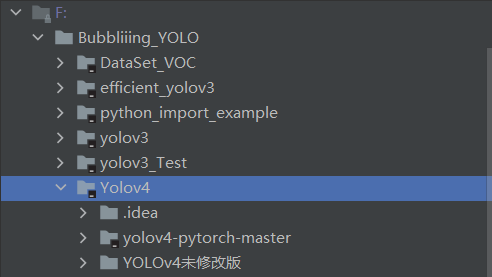


* no module name utils.utils(no module name nets.yolo)等问题，一般是根目录不对

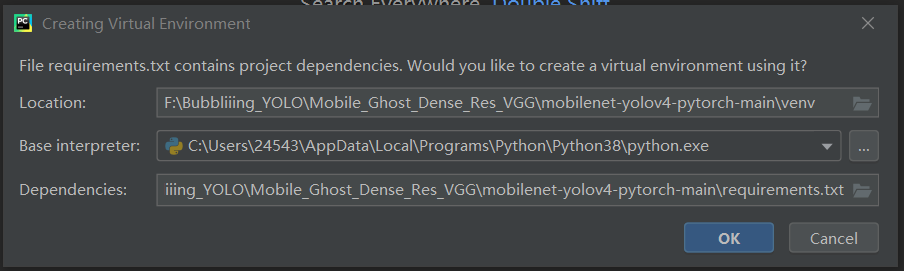
下载下来的代码要像这样打开：



而不是像这样打开：



* 注意下载了github代码之后，打开文件pycharm会问是否为该project创建一个虚拟环境，一定要点**Cancel。**然后在右下角选择python3.7解释器即可。



像这样打开代码就是在base环境中，不需要激活其他虚拟环境，因此如果缺什么包的话，直接在终端里面用pip install packagename即可。

* 关于网络的初始化方式：

1. 如果直接定义了一个网络，那么网络内的权重都是随机的，即使没有用譬如何恺明方法初始化或者加载预训练权重，网络也是有随机权重的；
2. 用某些初始化方法，比如何恺明初始化，normal初始化；
3. 加载整个网络的预训练权重；
4. 只加载主干backbone的预训练权重，此时neck和head部分的权重如果没有初始化，那么都是些随机的权重。

* 关于mAP：

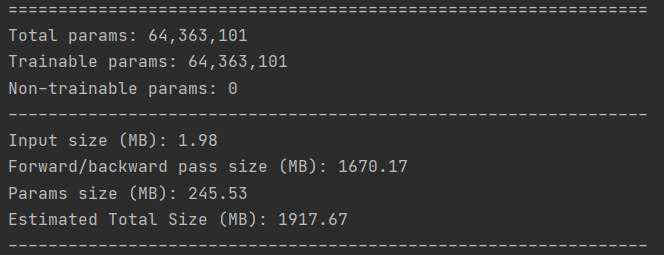
注意目标检测算法的mAP必须是针对具体的数据集而言的，比如同一个YOLO算法，在简单的Voc数据集上mAP在90%以上，而像在visdrone这样非常有挑战性的数据集mAP甚至在40%左右。所以要验证算法在mAP上提升了多少必须先用数据集在原算法上测得mAP，然后用改进算法对同一个数据集测mAP，这样比较才有意义。

* 关于网络的参数量

参数量只和网络有关，计算量还和输入数据有关，比如AlexNet

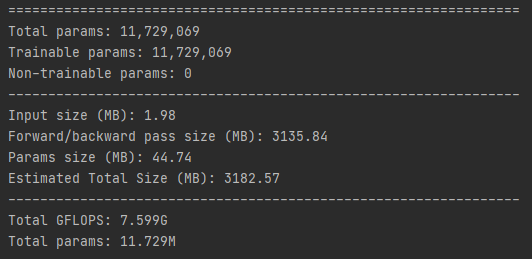


用summary可以看网络参数量：



上面是YOLOv4的，Total params：64363101就是参数量，有64.363M。

而修改了使用了深度可分离卷积的mobilenetv3-yolov4则如下：



可以看到参数量大幅减少，为11.729M

* 关于网络的训练方式：

1. 如果是修改了主干网络，比如在主干网络中加了注意力机制，或者对主干有任何变动，但是又没有新主干的权重，就得从0开始训练，需要的epoch较大，一般不建议；
2. 如果是修改了主干网络且有新主干的权重，那么就从主干开始训练，只加载主干权重，其余权重以一定方式初始化；
3. 如果修改了neck，但是主干没有变，那么从主干开始训练。只加载主干权重，其余权以一定方式初始化；
4. 整个网络及其权重都有，只想训练自己的数据集，那么加载整个网络的权重即可。

可以在定义网络的时候加载主干的权重（如下图），注意加载了之后就初始化了，否则加载了也被覆盖了。或者先初始化再加载主干的权重，顺序不能乱。

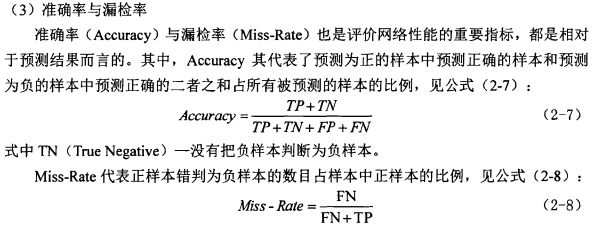


也可以实例化一个网络之后，像这样加载：

Model = YoloBody(.......)

Model.backbone.load\_state\_dict(torch.load(.......))

* 什么是漏检率



* Colab配置

