工作日志

1. VOC格式数据集的准备

VOC格式头盔数据集所有标注bbox信息的数据存储于annotation下的所有xml文件中，yolo训练集要求所有image有一个txt文件与之对应作为label文件，txt文件中的格式为<object-class> <x> <y> <width> <height>，所有含classes的image对应的第一项为1，classes的声明在annotation2txt.py文件的开头。除了label之外，还需要一个指明所有图片对应地址的train.txt和test.txt文件。至此，训练数据准备工作完成之后只要到yolo工程下的cfg\voc.data文件中更改需要的configuration即可。

Keras版本的yolo工程准备的数据格式略有不一样，参考voc\_annotation.py文件进行。Keras版本的yolov3+mobilnet的训练数据与上yolo的数据一致，使用相同的txt文件。

1. 模型训练

Keras版本的yolo工程训练yolo\_tiny,训练文件train.py修改文件头部定义即可。继续训练可以在line31的weights\_path加载储存在log\000\下的.h5模型文件，模型每3次echo存储一次最好的模型。训练策略先训练最后两层输出层，达到一定训练瓶颈后（20个epoch左右）可以解冻前面的网络训练所有参数。

Keras版本的yolo\_MobileNet工程训练，目前只能使用yolo\_body，原作者只对yolo做了mobile的结构改造，想要使用tiny+mobile的结构需要后续自己改造结构。

1. Keras .h5文件转tensorflow.pb文件的要点，keras checkpoint储存h5文件时一定要**同时储存模型weights和结构**（注意参数‘**save\_weights\_only=False**’），否则后续无法加载转换模型。转换过程先定义模型结构，后使用load\_weights加载权重进用户自定义的网络结构（load\_models只能加载keras框架内定义的网络）
2. 云服务器上的模型训练，使用tensorflow1.14.0出现意想不到bug，训练过程中loss下降的很快layers全解冻后甚至loss接近0，但是模型在inference上没有任何效果，在tensorflow改回1.12.0后训练及推断正常。**Tensorflow1.12.0适配keras2.2.4版本无问题**。
3. 数据的augmentation，放弃flip及其他仿射变换类方法，因为为了避免bbox在变换后无法对应上的问题，主要使用blur，noise一类的方法希望模型泛化性能增强
4. MAP工程，注意group truth和label的定义以及xml文件中的信息，xml文件标注的是和有的group truth要求一样的XMin,XMax,YMin,YMax的坐标格式，但是我们的输出是‘**<class\_name> <confidence> <left> <top> <right> <bottom>的格式**（在yolo.py中可改）
5. Map工程的detection result需要运行yolo\_video.py注释的部分，同时将一张一张预测的模式comment，到yolo.py中为image\_detection方法添加一个我放在之后注释中的参数‘list\_file’，同时取消detect\_image方法中关于‘list\_file写txt文件一行注释，运行即可生成批量的detection results files。（过程中注释原作者的画图部分代码‘drawing library’否则在运行上百张图片后产生不明原因bug导致中断）
6. 同4所述，任何importError有关keras.backend.tensorflow\_backend.py没有某些模块均是tensorflow与keras的版本适配问题查看 https://docs.floydhub.com/guides/environments/官方说明安装合适的版本。
7. 云服务器GPU环境配置：

1）tensorflow-gpu1.12.0 2）cuda 9.0 3）cuDNN7 4）Nvidia 驱动384.66 5）gcc编译器 小于4.8版本 6）keras 2.2.4 7）python 3.3 – 3.6

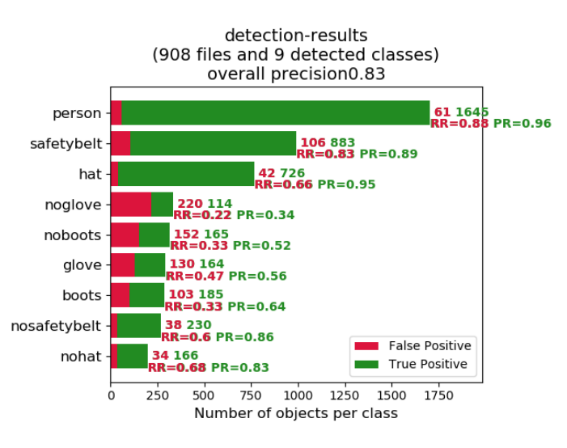
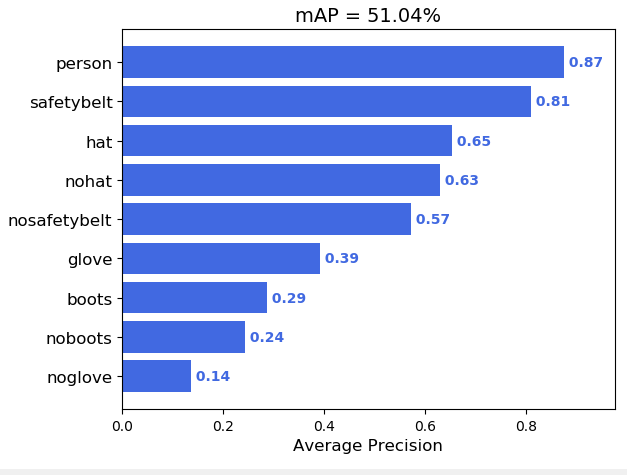
使用pip install可以指定tensorflow-gpu版本进行1.12的安装；conda install cuda==9；

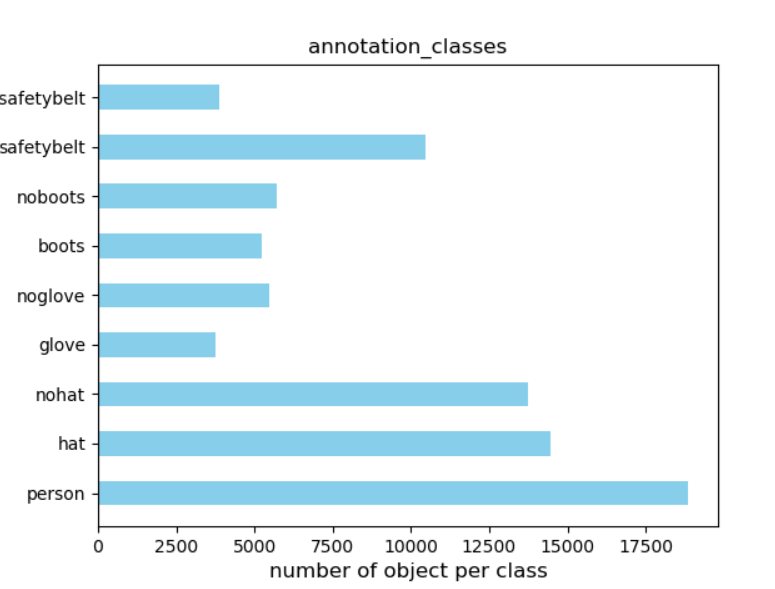
通过NVIDIA官网下载cuDNN的rpm安装文件，选择适配cuda9.0的cuDNN7.6.4下载，linux环境下rpm -ivh filename.rpm安装，重启后生效；使用nvidia-smi查看本机gpu驱动信息，云环境驱动已安装384.66版本；gcc -v检查；pip安装keras合适版本；conda create -n gpuenv python==3.6创建python3.6的gpu虚拟环境

1. mAP工程目前输出的scores\_PR\_curve曲线中的score是自己定义，未找到程序中使用score的地方，需要重新确认PR曲线的定义及相对应的score数值定义。
2. 10月21日，已测试云服务器端的base环境tfenv CPU环境以及gpu GPU环境，以及tensorflow1.12 cpu，gpu版本，1.13.1cpu版本，均无法解决训练loss降得太快，模型无用的bug。相同环境在本机测试无问题。更新于10月22日，已查明原因是train.txt文件不符合格式，只保存了训练数据的路径未保存annotation的结果。使用voc\_annotation.py重新生成txt文件即可。
3. 10月31日，生成voc格式个train.txt和testnoAnnotation.txt的流程是，通过substation数据目录下的DataPreparation.py生成带voc格式annotation的train和test txt文件，如果需要额外的数据修改DataPreparation.py文件内的annotation和JPEGImage路径，生成对应的moreAnnotation目录下的train和test txt文件，将.txt文件中的数据路径和label手动复制到一开始生成的train.txt和test.txt文件中，训练程序会执行shuffle train.txt这里合并之后不需要我们来做shuffle。生成groundtruth（labels文件夹）和detection所使用的的testnoAnnotation.txt需要执行substation数据目录下的valid\_label.py程序，用于清理test.txt中的annotation，生成noAnnotation.txt文件和lebels文件夹。 在keras-yolo3目录下，执行yolo\_video.py --image --multiMode模式读取noAnnotation.txt文件进行批量inference生成detection results.

在云端只需要使用DataPreparation.py准备好train.txt即可，在本机使用DataPreparation.py准备好test.txt并由test.txt和valid\_label.py生成noAnnotation.txt和labels文件夹用于mAP工程检验。

1. 10月31日模型效果评估，yolo-tiny在加入part-B数据补齐hat与nohat之后尽管loss很低为7左右但map极差。可能原因：训练数据过于单一，模型过拟合，特别是加入基于单一数据的新anchors，泛化性差。Yolo-tiny对9个class进行训练，loss26.8 mAP结果显示除boost与glove 4个类之外，其他5个类的recall和precision以及整体mAP都很高，可能原因猜想：数据不平衡boost和glove的标注数据都远少于其他的类，或者yolo-tiny模型对于体积小的物体识别能力有限考虑更换模型。





1. 10月31日，使用原版yolo\_anchors及原单一采集到的数据源训练出来的mobilenet-yolo，loss23，对测试数据表现良好，能够准确识别tiny不能识别出的boost与glove，但是在voc数据源上识别效果较差，猜测原因：数据类型过于单一，对我们的数据过拟合了，需要更多的数据提升泛化能力。
2. 11月1日，使用k-means的anchors以及加入3000hat数据及对应数量的nohat数据增强我们的数据集后，训练的yolo-tiny loss从原先的26.7下降到了20.3，但是测试的mAP结果仍然一般，特别是nohat的recall Rate特别低，跟原先VOC实验基本一致不太意外。对boost和glove识别效果依然不佳。猜想一是训练数据不足，二是tiny对小物体识别效果不佳。