**检测作业总结**

**一、tinymind运行地址**

https://www.tinymind.com/jsqiaoliang/w9-object-detection/executions

**二、心路历程**

本周作业的检测任务让我领略了以后工作的日常，即如何在工作中查找bug，以下是我这两周的心路历程。

1. 在按照标准流程配置完linux虚拟机环境后，遇到的第一个问题是protobuf，因我最先下载的是3.6的版本，用编译好的程序来进行作业，发现爆了很多serialized\_options=None的错误，后辗转反侧，最终发现此处应该是option，故而发现编译的版本有问题。后又试了多个版本，由于此时距离我第一次安装已过去了许久，故而安装又出现了几次错误。
2. 在制作数据集时，在pet的基础上按照教程进行了修改并输出tf.record,并最终明白了tf.record文件是一个便于tensorflow程序读取，减少I/O操作的文件格式，其可以通过example将数据源源不断地进行预加载。
3. 开始在本地运行，程序开始后非常慢，且运行了几步后会被killed，我开始以为是虚拟机的问题，觉得任务已经完成，只需进入tinymind即可，谁想实在天真。
4. 刚开始上，tinymind后，发现其我的object\_detection模型用错了，应该用r1.5版本，而错用成了master版。因此了解了github中的代码是分为多个版本的。
5. 在修改完成后，报错找不到配置文件，发现configue放到了data中，因此仔细阅读了run.py的相关路径说明。也学习了gitbub的文件传递、删除等操作。
6. 随后发现tinymind上运行出现了无法追踪的错误，后来才发现问题在protobuf的版本，protobuf是我本次作业最大的坑(虽然期间我试了多个版本，但最终我发现我的protobuf没有删干净，其实文件是在usr/local/sbin下，而我一直对usr/local/bin做了操作)
7. 回过头看，本次作业并不困难，需要的是把过程中每个环境做对，并有对相应环节检查的方法，如即使查看protobuf的版本等等。

**三、学习所得**

而这两周学习了object\_detection的相关知识，收获大致为以下几点：

1. 神经网络能学到的规律是人通过ground true指定的，等同于人告诉神经网络目标物体的框所在位置，通过训练使网络也具备此类能力。
2. 整个检测框架分为特征抽提与目标框位置的回归，损失函数亦由这两部分组成。
3. 目前神经网络faster-rcnn用Region的Anchor机制来生成监测框，因中间卷积层的感受野相当于一个又一个的框，在这些框上任意生成9个不同长宽比的矩形，可以很好得实现检测框的效果，而框的长宽比框取决于ground truth的反馈。
4. 非最大化抑制，每个区域均产生了9个框，整个区域应有很多框，此时保留的原则是对于每个类别，先选与ground true IOU最大的那个，然后删除该类别余下与选择框IOU大于阀值的，同时将此前选择的框提到选择区；然后重复以上过程直到将该类别所有框删除或提到选择区，而最终选择区中的框就是生成的监测框。