计算机视觉Programming1实验报告

171250502 软件学院 曹润泽

**代码链接：**<https://github.com/MachineDora/cartoon_detection>

为了降低大小，下文中其他多余的输出图片，以及训练的模型以及数据集都不在该代码中。

数据集在moodle上下载，是一样的。

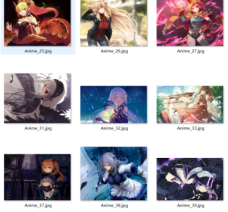
模型权重因为过大，放在百度网盘：<https://pan.baidu.com/s/1HLW8Wg0Tsrv7CIX-qwJiLQ>

提取码：upka

**框架结构：**YOLO3物体检测，框架详情见代码根目录中的model.png，详细细节也会说明。

**参考资料：**<https://github.com/wmylxmj/yolo-v3-iou>

初始代码是以Yolo为框架实现的动漫人脸检测，而这里的动漫人脸基本是日系画风，与本实验的数据集有较大差距：

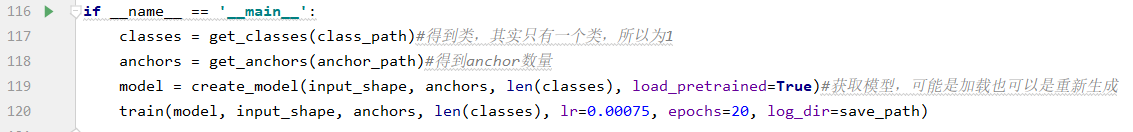
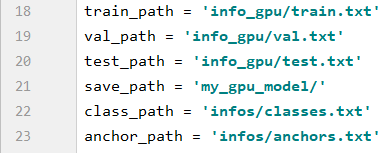
原型数据集： 我们的数据集：

可见原型数据集的画风一致，画面干净，较为容易检测，而我们的数据集画风杂糅，画面元素凌乱，检测困难。但是考虑到实现目标具有大同小异的特点，故采用该代码框架模型训练数据。

**工作详情：**

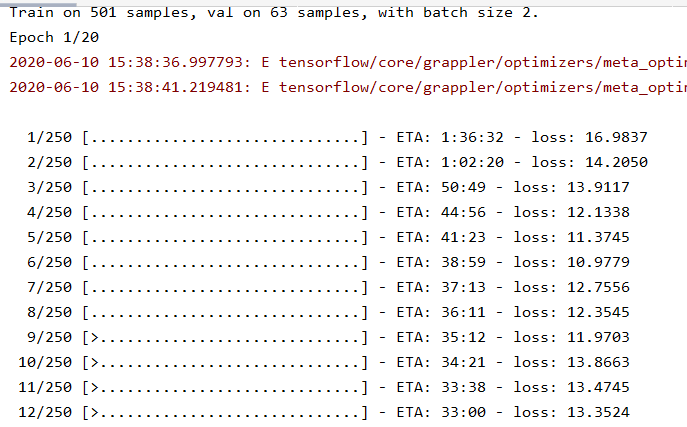
本来，直接使用动漫人脸训练好的model来测试卡通人脸，结果不佳（这是后来才知道的，因为在载入数据集之后训练出来的模型测试出来的结果与直接测试的结果大相径庭，所以才知道效果不好）。所以想使用我们的数据集，这样赋予模型对卡通数据的亲和力，可能效果会很好。

一开始，查看代码模型，只改变了train文件中的各项属性路径和保存路径，没有修改训练方式，即load\_pretrained这项属性没有设置成True，也就是说会加在动漫人脸的权重来训练卡通人脸。



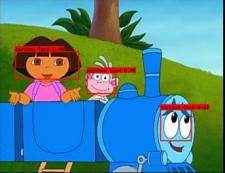
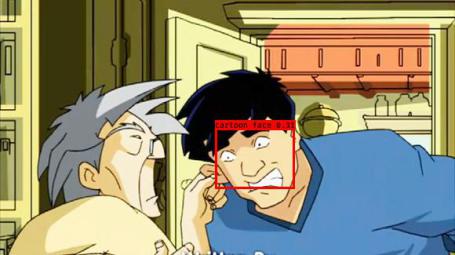
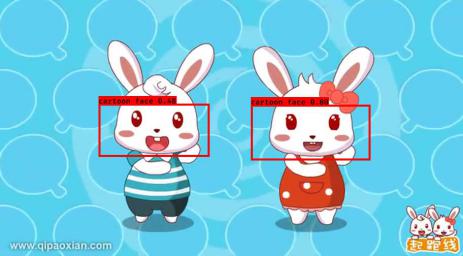
**第一次训练（CPU+625样本+加载动漫人脸识别权重+学习率0.00075）**

本来想一次性训练所有的8000张图片，可是经过实践发现，使用cpu训练的速度实在是太慢了，大概8秒一张图片，要花五六天才能训练完。后来就缩小了数据集，变成625个样本，500训练，62张验证和63张测试。效果如下：



该模型分为20个阶段训练，批处理大小为2，500个训练样本，则需要进行250次迭代，ETA为剩余时间。最后实际上花了大概10个小时这样训练好。

后来测试之后发现效果其实还不错：（红框的数字不是IOU值，是score，一种效果的判定分数）



部分图片确实效果正常，但是很多图片没有识别出，考虑到样本数量太小，无法将更多的特征分布容纳进去。因此尝试使用GPU训练更多的数据集。

**第二次训练（GPU+8000样本+加载动漫人脸识别权重+学习率0.00075）**

第二天，花了一天的时间配GPU环境（简直要累死……各种版本不对应特别难受）。

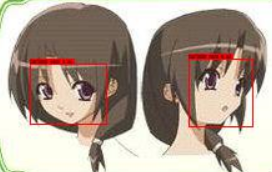
最终配好了GPU环境，开始尝试训练8000张图片。

如下图，与第一次训练同样的各种属性（阶段、batch size等）。将8000张图片相似地划分为6400张为训练集、800为验证集、800为测试集。可见速度比第一次训练快了很多。之前是8s一张图片，现在不到1s为一张。（图片略，和第一次训练相似的结果）

最终花费了将近15个小时训练出来了结果。模型保存到**my\_gpu\_model**中。

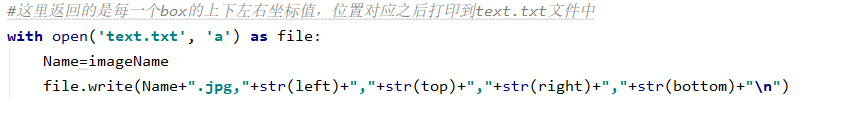
最后测试测试集，发现效果比第一次训练好了一点，之前不能识别出的图片现在很多都能识别出来。但是还存在着较大的问题，有的效果还不如之前的CPU版本，部分图片之前识别不出现在仍然识别不出。详情请见**outImages**文件夹（后来统计了一下，2000张图片有800多张无法识别出）。

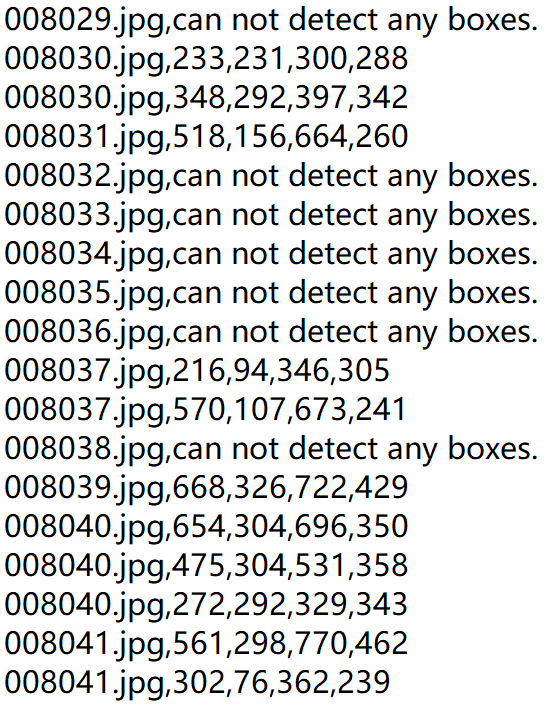
我在想，随着数据集的扩大，虽然训练数据的程度扩大了，但是图片的杂乱程度也增加的，不同画风的图片一旦增多，这种干扰应该影响了之前生成的模型，该模型讲从所有的画风中进行折中，一旦某种画风的存在很低，基本上就测试不出来了。或者是将这种角色脸的特征单一化，这样越普遍的脸越容易识别，越特殊的越难识别。



如上图所示，左边两张图虽然画风不一，但是脸部特征较为明显，而左图就极为夸张，整个数据集基本没有哪张图的脸部形状与其相同，所以识别不出来。

所以针对这种问题，我在想我们的模型只能检测出一种种类，也就是class的数量，我们这个实验中设置成了只有cartoon face这一类，所以容错率就会比较低。可能更好的做法是扩大class的种类，比如日系的划分和美式画风可以归为两类进行测试。这样可能需要引入标签。这次实验没有这么做。还是将class变成了1类。





从最后的text.txt文件（上图）中可以看出，可以识别的图像和不能识别的图像占比接近1:1。可识别的比不能识别的稍微多一点。

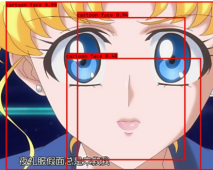
后来想到我不是重新训练的权重，是加载动漫人脸的权重，由此猜想，是不是动漫人脸会影响我这个数据集的检测。

**第三次训练（GPU+8000样本+重新训练权重和模型+学习率0.001）**

很明显的看出，这里和之前最大的差别在于loss的大小，之前的训练loss平均取值为15，这里起始数值就非常非常大，因为是从零开始训练，所以误差比较大。（训练时间图片略）

这样一来，训练的模型就肯定是全部基于我们的数据集了，我原以为效果会变好。

第三次训练再测试的结果保存在outNew中。



从大致结果中可以看到，有些第二次训练不出来的确实测了出来，比如第一张图在第二次训练的时候识别不出，可是第三次可以识别，但是似乎不是很准确。后面两张同上。（不过大多数情况下都是一张脸对应一个box，还算比较正常）

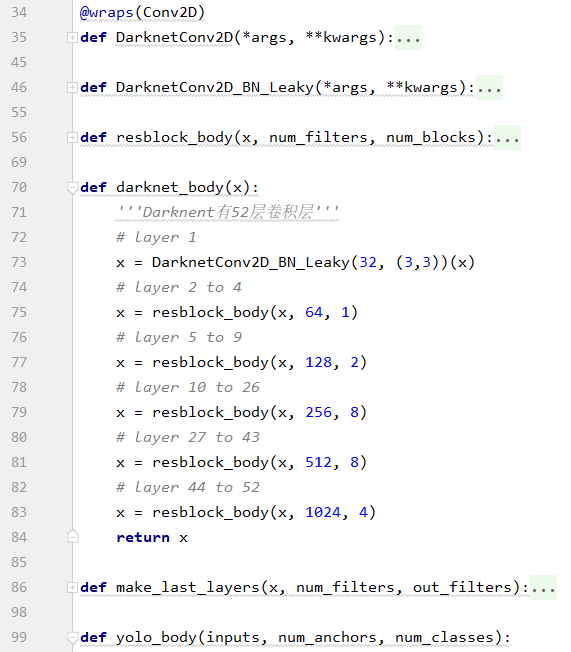
总结得到：自己训练的模型识别的次数确实增加，但是会带来一些其余重叠的box干扰，这是个很严重的问题。平衡来看，还是第二次加载已经训练好的动漫人脸模型较为妥当。

至于为什么出现这种情况，我也不太清楚。以后有机会再研究吧。

综上所示，三次训练下来，效果最好的应该是第二次，即采用动漫人脸权重的模型效果最好。

**模型详情：**

模型结构十分复杂，详情参考项目根目录中的model.png，在这里根据项目中的model.py文件简单说明。



首先**DarknetConv2D**是设计Darknet参数的卷积函数，其中Darknet是YOLO作者写的一个深度学习框架，用于提取特征的主干网络。

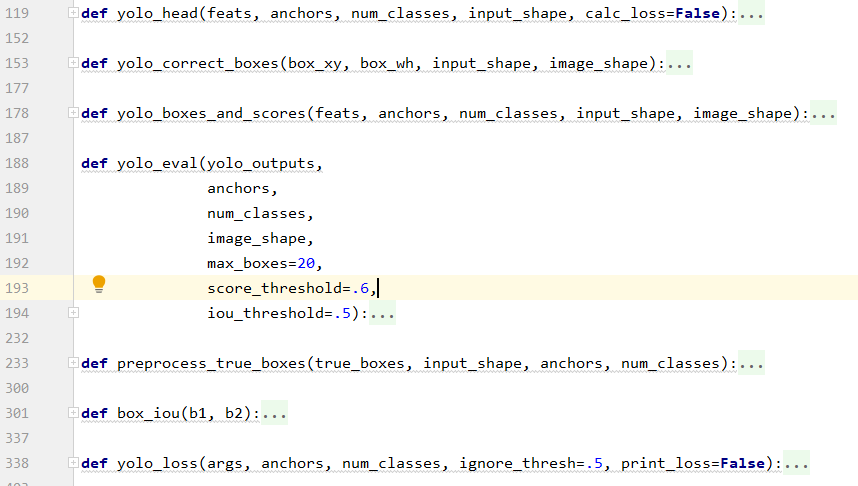
**DarknetConv2D\_BN\_Leaky**是将Darknet进行卷积和归一化以及LeakyReLU激活。

**reblock\_body**表示残差快，用于进行Darknet卷积，用到**DarknetConv2D\_BN\_Leaky**方法。

**darknet\_body**方法用于生成darknet网络，该实验为52成卷积层。该方法与**make\_last\_layers**共同组成yolo模型的主体部分**yolo\_body**方法。

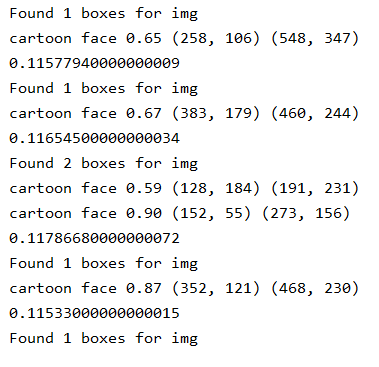
生成模型之后就会被train方法调用。之后会根据具体数据进行训练。

随后的几个方法主要是根据特征网络获取目标box进行数值转换的过程：



我没有很输深入地搞懂这些关系，但是明确一点就是，这几个方法直接决定了IOU的大小，box的准确度，以及box的位置大小。

至于本身模型的评价指标，也做出了点信息：



这是控制台的输出内容，其中第二行第一个小数表明这个box检测的分数如何，越接近1表示越接近卡通人脸，后面就是box的坐标信息。第三行是检测速度。

**心得与思考：**

本次实验让我拥有了一次使用YOLO3模型检测人脸的机会，但是根据我的了解，YOLO是目标检测的优秀框架，这次被用作深度学习检测人脸，我的理解是这些人脸也可以被用作目标。其实任何东西都可以被用作当做目标，这次实践就是很好的证明。

本次实验做了三次训练，十分消耗内存和CPU，也是因为这次实验才有了我装GPU的机会，也让python处理数据的速度更快了。

不过本次还是套用的别人的模型，自己很难写出这么优秀的代码，自己做的工作无非是装载自己的数据集，对数据的输入输出进行处理，打印数据查看过程罢了。不过确实很费时间训练罢了。

最重要的一点是我觉得这个数据集实在是太杂了，这么杂的数据很难做到完全识别检测，有些东西我个人觉得基本上是不可能的。

比如说下面几张图：



这些图我感觉实在是太困难了，不是光线昏暗就是特征非常不明显，和其他大部分图片的划分都迥然不同，还有的图片甚至自己看都很难找出卡通角色脸。可能有这种检测的方法可以检测出结果，但我是无法想象的。就如同我之前说的一下，如果只把他们作为一类感觉是很难的，应该是需要进行分类的。

不管怎么说，实验使我受益匪浅，开拓了我对计算机视觉这门学科的认识，很有意思，还是很有帮助的。