# 1. Base code

*CVPR19-RoITransformer倾斜框检测*

<https://github.com/dingjiansw101/AerialDetection>

请按照上面的逐步配置环境、安装依赖，其中务必注意RTX2080必须使用cuda10.0及以上，9.0会报错。

# 2. 主要流程：

## 2.1 目标检测框标注

此过程在自己的个人电脑上完成，不需要使用服务器。

使用标注工具labelme进行标注，每一次画一个框，由于我们画的是倾斜框，应当使用labelme的多边形（polygon）标注模式（其有多个模式，一般的目标检测用的是矩形框模式，多边形（polygon）标注格式，用于倾斜框目标检测和语义分割，我们需要的是后者）

安装教程和使用教程可见：

<https://www.pianshen.com/article/8141190515/>

如果有需要的话，标注前应当首先选择好标注得到groundtruth文件（以下我们对此简称gt, 即groundtruth的缩写）的默认保存位置，文件——>选择默认保存位置；如果不选择，会默认保存为和图片文件夹是同一个文件夹（这样的情况我们我们是不太想要的因为往往我们想要一个纯净的文件夹）。

Q&A：

**Q1: 我们建议如何安排文件夹的结构？**

比如说今天我有1个标注任务是Burberry，那么我会把大文件夹命名为'Burberry’，然后按照如下所示去安排其下属文件目录：

Burberry

- img

- 1.jpg

- 2.jpg

- gt

- 1.json

- 2.json

也就是说我会把图片都复制到img目录下，而标注的gt文件都放在gt目录下（将gt目录作为默认保存目录）。

**Q2：为什么要所有人统一使用一种文件夹安排方式？**

首先，这种安排方式是很多base code中使用的一种工程实践中得到验证非常方便且统一的安排方式，我们认为这种方式值得推广。

其次*，标准化*工作是最大好处是给工程师减轻负担。标注数据的同学在标注阶段的注意和统一会给后续负责算法和训模的小伙伴减轻许多的负担。

## 2.2 数据集转换

刚才说到了检测标注后会每一张图片会对应地得到labelme自己的json格式文件1个，并且分别保存在img, gt中。那么仅有一个标准化的目录结构还是不够的，我们必须要转化成为标准化的通用数据集接口。现行目标检测数据集接口主要有Pascal VOC和coco等几种接口。在我们的RoITransformer的方法（也就是我们用来做倾斜框目标检测的方法）中，使用的是coco接口，所以我们必须要将labelme先转化为coco接口，才可以上传到服务器上使用。

Q&A:

**Q1: 如何转化为coco接口？**

* 如果是单检测框，使用：

labelme2coco.py 或

labelme2coco\_mydataset-3m.py

其区别在于，labelme2coco.py是github上一份转化的代码，我没有直接使用，不过应该没有太多bug；没有直接使用的原因在于我需要对目标检测进行一定程度的数据增强，在labelme2coco\_mydataset-3m.py中手动引入了数据增强

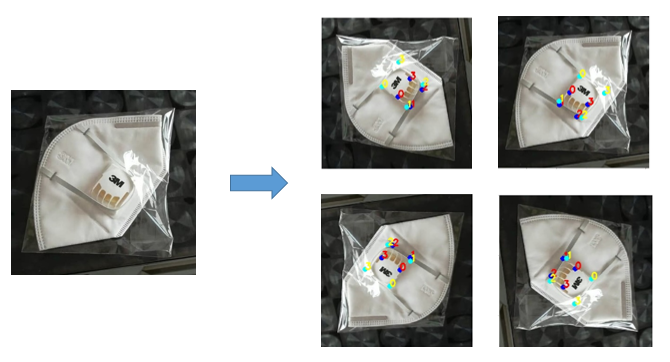
代码的使用非常简单，直接见代码即可，只要填写好src\_gt\_path、src\_image\_path和saved\_coco\_path3个变量即可开始转换。

* 如果是多框，使用labelme2coco\_aug.py，在这之中引入了手动数据增强（aug即为augmentation的简称）

**Q2：代码有几个主要功能？**

以labelme2coco\_mydataset-3m.py为例，代码的功能主要有：

* 对数据集转换coco格式——一般功能
* 使用旋转90度对数据集进行增强——有效克服旋转不变性



控制方式：

362行（不同文件不同的行数）：FlipTransform\_ornot=True/False

* 在已经生成了coco数据集的情况下，利用可视化检查标注是否有误

标注出现问题是有一定可能的，我们需要在这个节点保证没有问题，这样后面出了问题很利于我们debug~

使用vis\_anno\_by\_imgname可以检查标注的正确与否。

vis\_anno\_by\_imgname=partial(vis\_anno,coco\_tr\_imgpath,coco\_tr\_jsonpath,coco\_val\_imgpath,coco\_val\_jsonpath)

vis\_anno\_by\_imgname('5MqtjLaOFYP8yT1qSr00tNaQI3A.cnt.jpg')

**Q3: 数据集完整正常转换完是什么样子的？**

coco/

- annotations/

- instances\_train2017.json

- instances\_val2017.json

- images/

- train2017/

- 1. jpg

- ...

- val2017/

- a. jpg

- ...

我们此时只需要把整个‘coco’文件夹改名为一个有标识性的名字上传服务器即可，比如4月3日创建的Burberry的目标检测数据集我们可以命名为 Burberry\_coco0403

## 2.3 代码config文件配置

完成好2.2我们就可以将生成的coco数据集放到服务器上面了

接下来就真正到算法配置的环节了。

我们以AerialDetection-master为主文件夹

AerialDetection-master/configs/下存放了最关键的config文件

我们一般为单独的一个应用新建一个二级文件夹，在其中放入我们的config文件，比如：

对于gucci的logo检测：

AerialDetection-master/configs/GUCCI\_OD/gucci\_od.py

其中所有的config文件都与gucci\_od.py有着同样的结构，现在这个来讲解。

Q&A:

**Q1: 有哪些地方需要改动？**

**改动点1：**

ctrl+F把所有的num\_classes改为当前需要检测的类别数+1（+1是只背景，余下都是要检测出的前景）比如对于每张图只有一个GUCCI logo的这种情况，num\_classes=2

**改动点2：**

142行-143行附近：

dataset\_type = 'GucciDataset'

data\_root = '/home/xby/dataset/guccilogo\_coco0324/'

在这里的data\_root是什么呢？就是我们在上面提到的coco数据集，所以后面提取图片和标注文件的时候都从data\_root中提取，后面都不需要变了。

这里的dataset\_type是什么呢？这个问题我们在**Q2**中进行解答。

ann\_file=data\_root + 'annotations/instances\_train2017.json',

img\_prefix=data\_root+'images/train2017/'

**改动点3：**

250行要更改work\_dir，这个是让我们好把保存的checkpoint权重保存在这个上面

work\_dir = './work\_dirs/gucci\_od'

**改动点4：**

176行 设置每张卡的batchsize，一般是2，更大也是可以的，占用的显存多一些

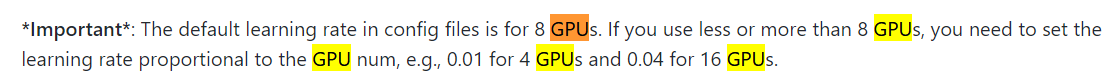
imgs\_per\_gpu=2,

workers\_per\_gpu=2,

**改动点5：**

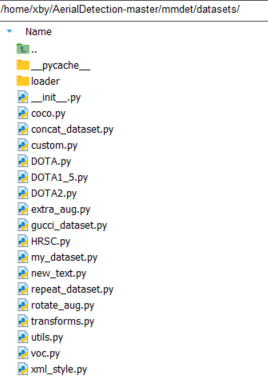
设置learning rate

根据你使用的gpu不同，请根据你自己的情况严格按照这个设置：



**Q2： 上面说到的dataset\_type是什么？**

除了config文件之外，我们还需要在/home/xby/AerialDetection-master/mmdet/datasets/下进行一些小改动



我们需要新建自己的dataset类别初始化文件，比如这里的gucci\_dataset.py就是我建立的，我们以他为例。

整个gucci\_dataset.py的内容如下：

from .coco import CocoDataset

class GucciDataset(CocoDataset):

CLASSES=('gucci')

首先我们需要新建一个类，其超类是CoCODataset，比如对于Gucci，我们就把类名写成GucciDataset了。然后，我们只需要把我们的类别名称按照我们在labelme中标注的类别名那样列在这个元组里就可以了，顺序要按照生成cocodataset那样，当然对于只有1个类别就不需要了。

在新建gucci\_dataset.py之后，我们还要在图片中的AerialDetection-master/mmdet/datasets/文件夹下的\_\_init\_\_.py中加上如下这一行来把我们刚才新建的类import进来。

from .gucci\_dataset import GucciDataset

所以之前的dataset\_type就是这里的类名了，也就是这里的GucciDataset

## 2.4 训练、验证、测试

接下来开始训练过程了，只要我们进行了标注、也配置好所有刚才的config文件，训练过程就不会出问题了。

**Q1: 如何进行训练、验证**

# 从头训练(举例)

python ./tools/train.py configs/GUCCI\_OD/gucci\_od.py --validate --gpus 2

# 中断恢复训练(举例)

python ./tools/train.py configs/GUCCI\_OD/gucci\_od.py --validate --gpus 2 --resume\_from work\_dirs/gucci\_od/epoch\_30.pth

# 

**Q2：如何进行测试？**

**测试指标为precision、rec、F1，具体操作很简单，请参照代码自行更改并运行。**

AerialDetection-master/tools/test\_xby\_for\_text.py

# 3. 可视化及应用

**Q1: 如何可视化？**

使用AerialDetection-master/single\_vis.py，具体操作很简单，请参照代码自行更改并运行，效果如下：



**Q2：如何在具体应用中先检测再裁剪？**

**需要结合crop，请使用**AerialDetection-master/crnn\_data.py**中的crop\_words函数结合，目前，我以单框检测的GUCCI LOGO做为一个例子，写为gucci\_crop.py文件，放在新版的keer\_ object\_detection\_lib中。**