# 头盔检测流程记录

## 1 比赛内容

背景内容：随着建筑施工行业的不断发展和安全意识的提高，确保工人佩戴安全帽成为了一项重要的任务。然而，传统的人工巡检方法效率低下且存在主观性的问题。因此，基于深度学习的安全帽头盔佩戴识别系统的研究和设计具有重要的现实意义和应用价值。通过结合深度学习和计算机视觉技术，该系统能够准确快速地识别建筑工地上工人是否佩戴安全帽，提高安全管理效果，减少事故风险，并为建筑施工行业的安全生产做出贡献。

比赛内容：在有限时间内考验同学们整理数据，搭建模型，训练模型，优化模型的能力。

## 2 比赛文件

数据集：包含图片，且每份图片都有相应的标注性文件，已经区分好包含4000张这样的图片的训练集和1000张图的测试集的头盔模型数据集，其标注性文件annotations的编写格式为voc，其文件格式为xml。这是很多模型框架读取的标注文件格式，此外还有我这次用到的coco格式，他是一个有特定结构的json字典文件。

## 3 比赛过程

### 3.1运行环境准备

#### 3.1.1环境选择

随详见源代码

#### 3.1.2遇到的问题和解决方案

众多众多的依赖性问题，根据我本人研究发现，任何版本都不能用太新的版本。

### 3.2数据准备

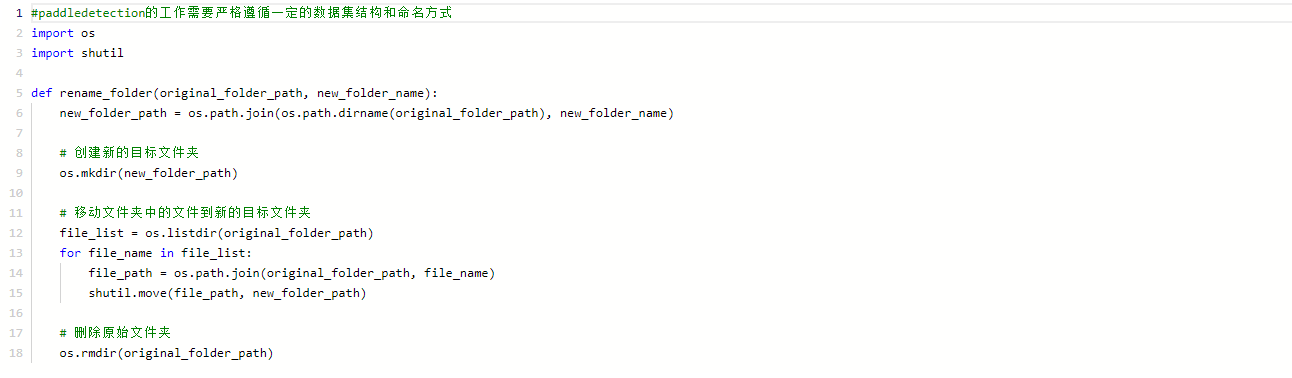
#### 3.2.1划分数据集

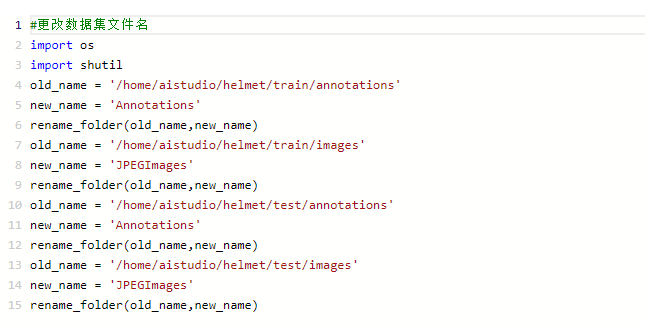
由于数据集的test和train是已经分好的，我们其实并不需要“划分”数据集，之所以还要进行划分的目的是利用PaddleX的划分功能来得到与描述性文件配套的训练列表名和从描述性文件提取的标签列表，因为大部分模型框架的配置文件都明确要求读取：1.图片文件夹的位置 2.其中所有图片的名单 3.所有标注文件的目录 4.所有标注文件中出现的对象类别的“列表”。



此外，由于本数据集要用于模型框架的读取，还需要按照他们喜欢的格式修改目录名称，否则配置文件肯定是无法运行的。

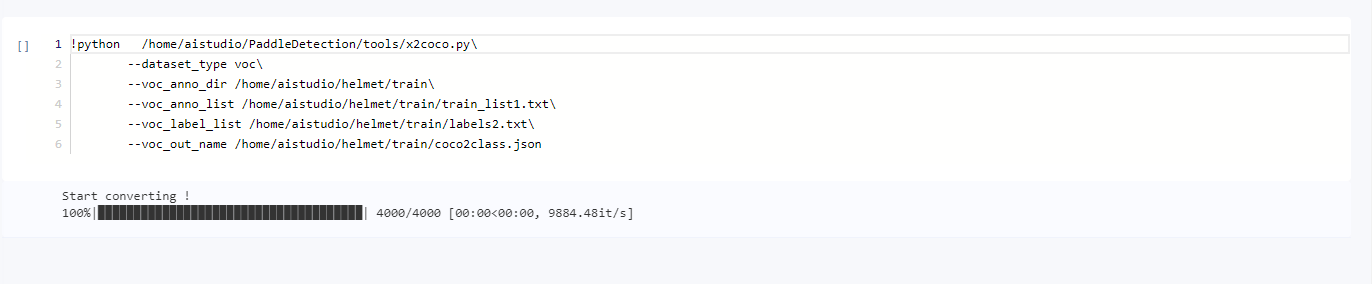
定义了一个有如下功能的函数





#### 3.2.2 生成模型可以读取的coco描述性文件

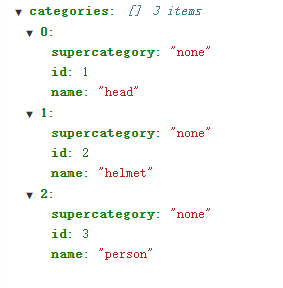
这个源代码里有



#### 3.2.3 遇到的问题及解决方案

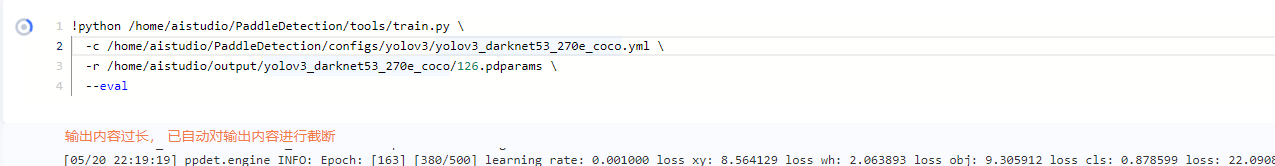
### 3.3 配置模型

#### 3.3.1 配置文件设置



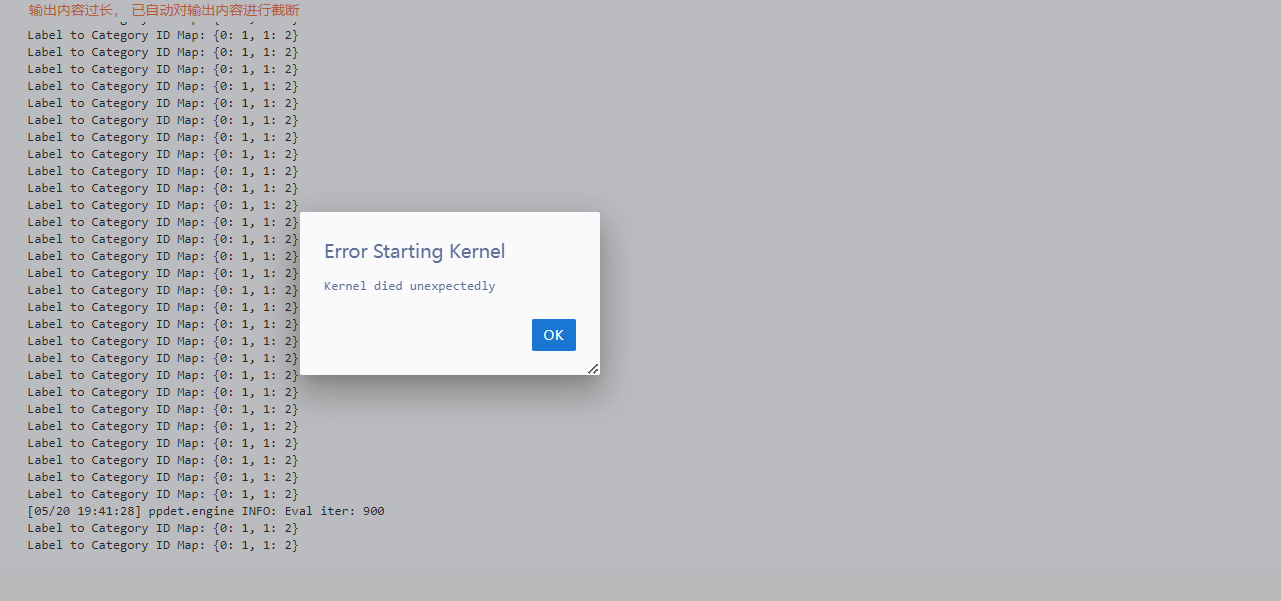
#### 3.3.2 模型选择和配置

### 3.4模型训练和推导结果及评价



直至5/20 22：21还在坚持不懈训练的模型。

#### 3.4.1 训练和评估及遇到的问题（xcj是数据集不行模型过早收敛 我是内存问题）



1. 内核突然阵亡的问题：这种问题也只能重启内核了
2. 如果在训练模型的过程中训练环境突然崩溃，如何避免次次都要从头训练的窘境？

答：使用模型在训练期间生成的权重作为起始参数。

1. 遇到一个稀奇古怪的疑似缓存问题

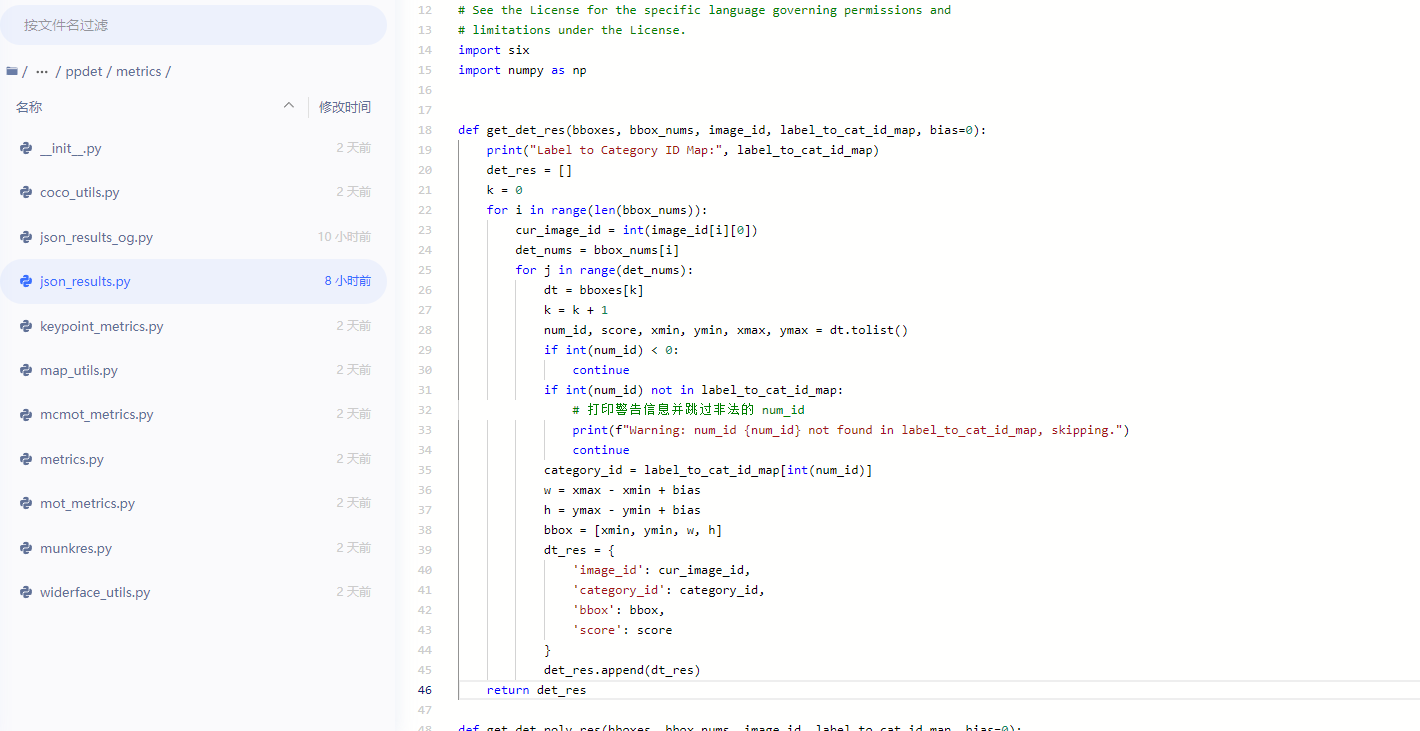


在模型训练的时候没有任何问题，一旦要进行eval的时候居然显示KeyError错误，这是违背逻辑的，因为本项目虽然数据集众多，但是类却很少，假如我使用新数据集生成描述目录coco文件失败，那也顶多有3个类label需要映射，于是我在文件中的get-date-res函数下添加了代码让其被触发的时候打印映射字典：

Label to Category ID Map: {0: 1, 1: 2}

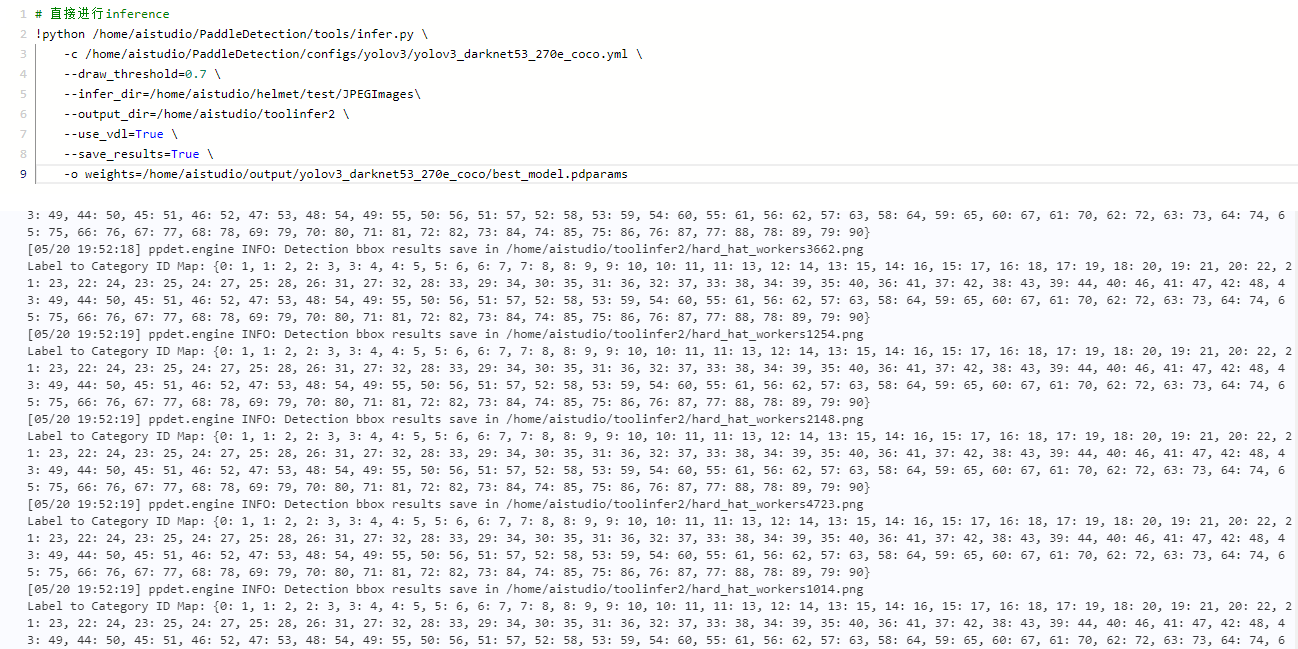
映射字典如此简单，是不可能出错的，因此我怀疑是num\_id的值不对，根据报错显示，这是在转换生成的coco字典的某一个结点的时候出现的问题，因为程序先打印了众多ID MAP，这一位置这个函数被多次正常运行了，直到碰到一个num-id值的异常。

num\_id值怎么会溢出如此之多，肯定不是我的代码问题，因此解决方案是：更改相关函数增加它的鲁棒性，遇到num-ID超出映射字典范围的直接跳过就完事了。



在评估的前几轮，可以通过报错发现确实有很多num-id不正常的实例出现，然而多进行几轮循环，此类异常到5/20/22：08都没有再出现过，因此我的怀疑是旧缓存影响了正常的字典读取和转换。

#### 3.4.2 模型推导（用测试集进行识别）



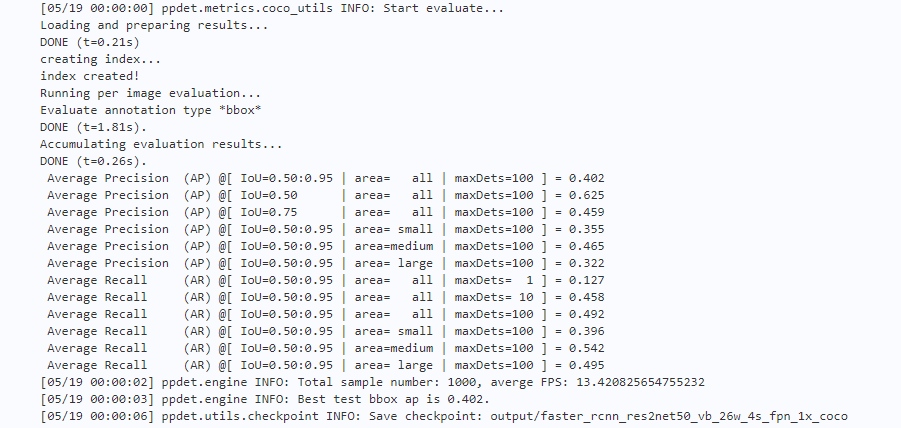
模型推导就是加载我们训练好的参数进模型框架里进行目标识别。以test测试集作为识别对象，最后会输出一个识别结果文件（不是老师要的那个文件）bbox.json，这是一个包含所有经测试图片被标记对象框信息的文件，还有所有经过标记框处理的被infer图集。效果如下所示，不过可能是生成bbox.json的时候catagory的名字丢失，所以标记框似乎用了某种default设置自动填充的类名，实在太离谱了：



#### 3.4.3模型评价

根据配置文件的评估，加载最优参数时对于测试集的平均正确预测率为0.402：

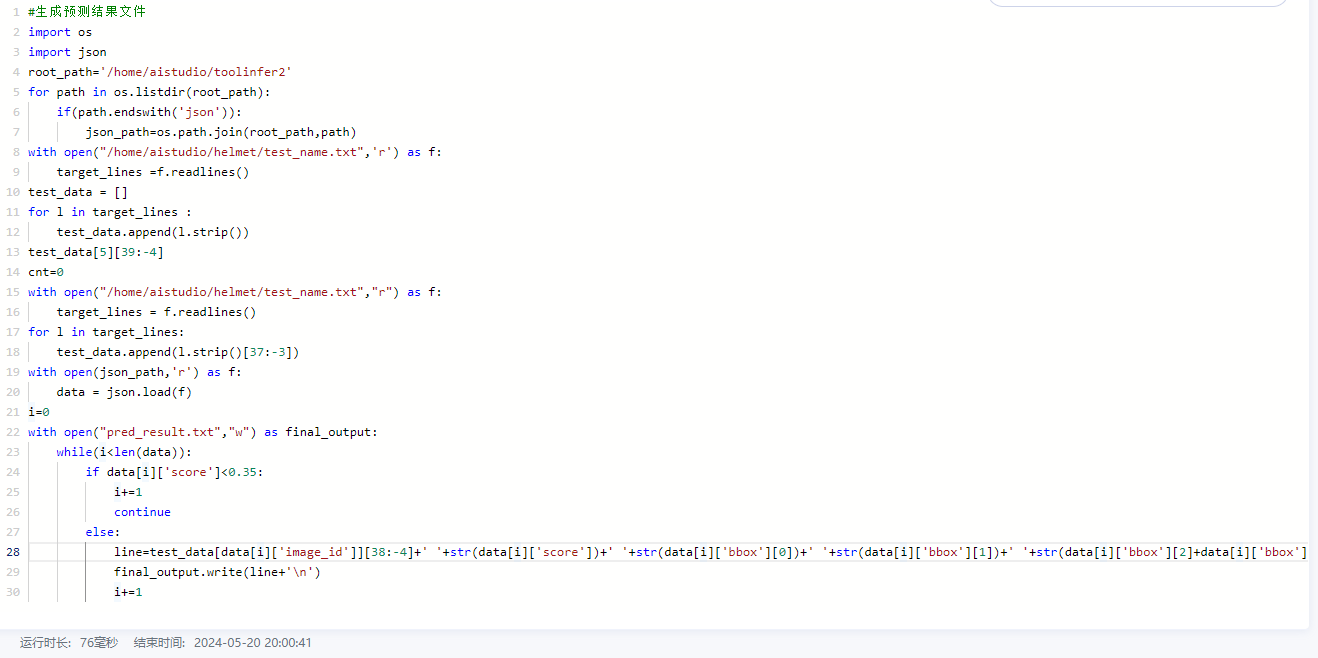
在此模型配置文件自身的评估体系下的平均正确率0.402已经可以准确识别出测试集中绝大部分1类和2类对象，即head和helmet，在比赛评估系统当中进行评估评分为55.7324，考虑到之前有大佬推测过评估系统是依次检测3个类的对象的识别情况，且三个类的权重相等，这意味着每个类的检测占33.333分。



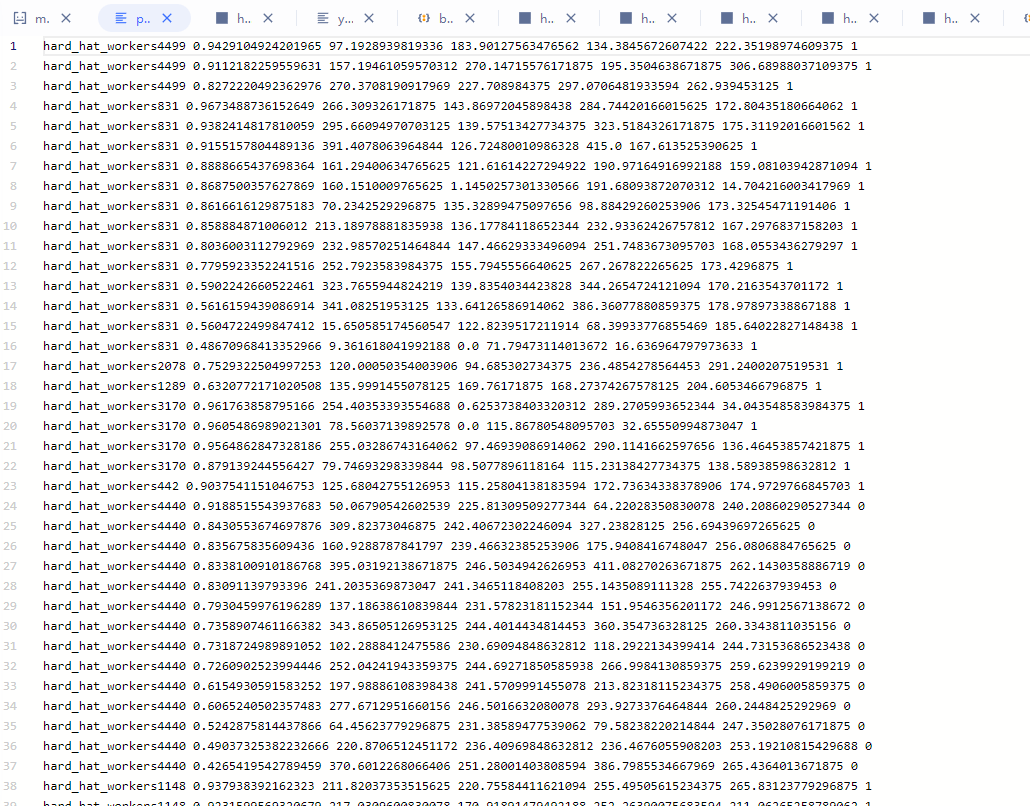
而由于之前提到的数据集的质量问题，在模型训练过程中对于第3类对象person的识别能力几乎无法得到训练，模型真正掌握的只有识别前两类对象的能力，假设另两类对象能够被完全识别，那么最高分数将是66.6667分，与我第一次上传获得的分数相比，可见这个识别准确率已经非常高了。然而模型训练过程中可观察出其收敛速度到后来非常之慢，考虑到res2net构架的强大性能和潜力，这肯定是有问题的。因此模型需要进一步的优化。

### 3.5 生成预测结果文件并提交

生成预测文件的代码（本质就是遍历）：



生成的预测文件如下图所示：

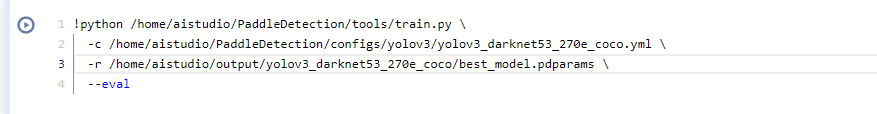


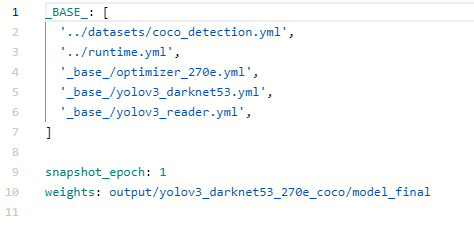
## 4 模型优化

4.1 优化思路

#### 4.1.1更换更优的数据集

#### 4.1.2更改当前训练过程最优参数更新的步长





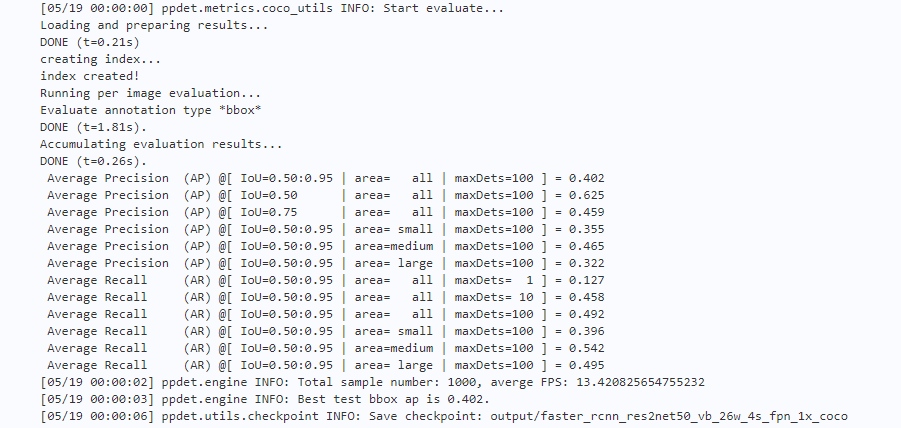
在我选用的框架下，配置文件中snapshort\_epoch，意味着每经过几个epoch进行一次评估和模型权重的更新，这会影响模型训练的效率：此设置默认值为5，根据实验发现，如果在5个epoch（每过一个epoch都会在之前选用的模型权重的基础上进行更新，意味着实际上的模型权重是每进行一轮epoch就更新的，只是不保存）间隔中实际上若有一个epoch的模型权重参数较好，他也不会被直接保存，在这种情况下，只会在一个5个epoch周期后进行评估时发现上次相较上次评估有进步，才会更新权重，但选用的是这个周期最后一个epoch的权重，简而言之，真正能提升模型精度的参数在这个过程中被“漏掉了”。

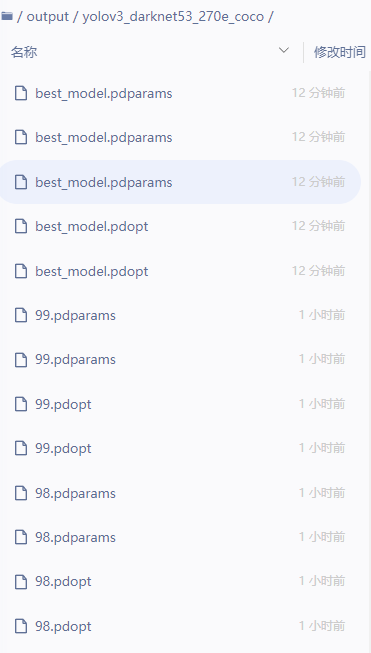
解决方法：每进行一轮epoch就进行评估，更新权重，这样的优点是寻找最佳模型参数的速度较快，缺点也不言而喻，由于每进行一次遍历都要保存权重，项目环境中的空间很容易不足。

在经过一段时间的训练过后，我发现最好的方式是：在模型训练的初期，让架构进行多方向的参数探索，这个时候不能过于快速确定“最佳权重”，因此确定新的权重文件的步数可以选为5个epoch甚至更多，在初期阶段这个值snapshort\_epoch太低会导致ap的进步速度非常慢；当模型的预测ap到达比较正常的水准后，慢慢降低snapshort\_epoch，这意味着参数进化的方向现在是比较确定的，这个时候每一次迭代变化的数据可能产生的突破都不能错过，总而言之，snapshort\_epoch的值在训练过程中应该在训练初期保持较大，训练越往后应该越小，这样一来能保证整个训练段较好的训练效率。

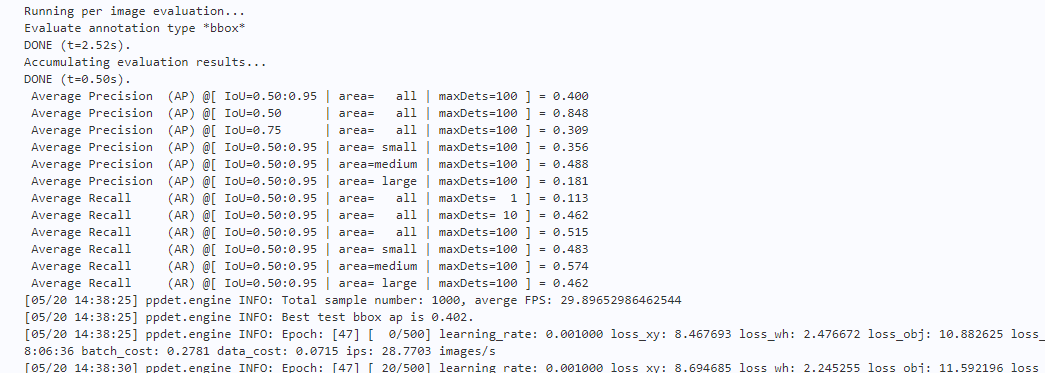
### 4.2 优化效果

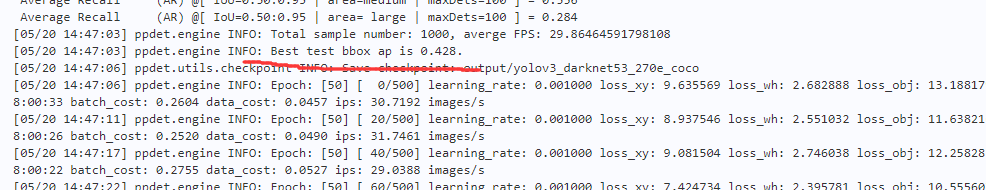
经过长时间的迭代，我终于获得了一个平均正确率比之前模型预测能力明显更高的版本,之前的res2net模型根据配置文件的评估，加载最优参数时对于测试集的平均正确预测率为0.402：

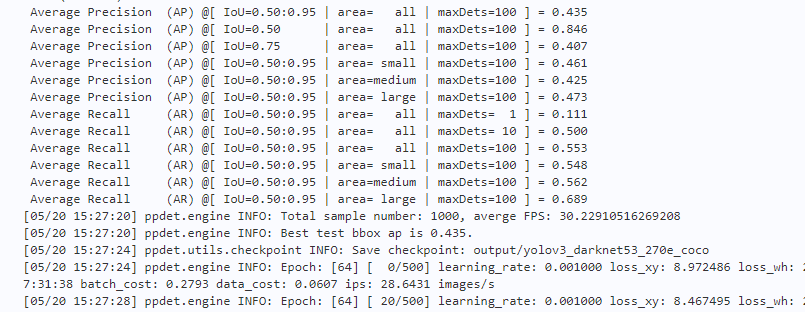


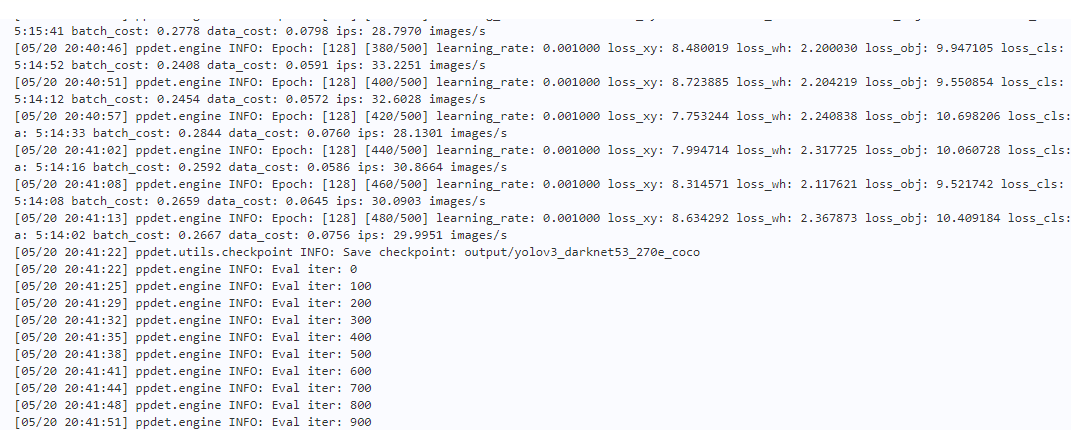


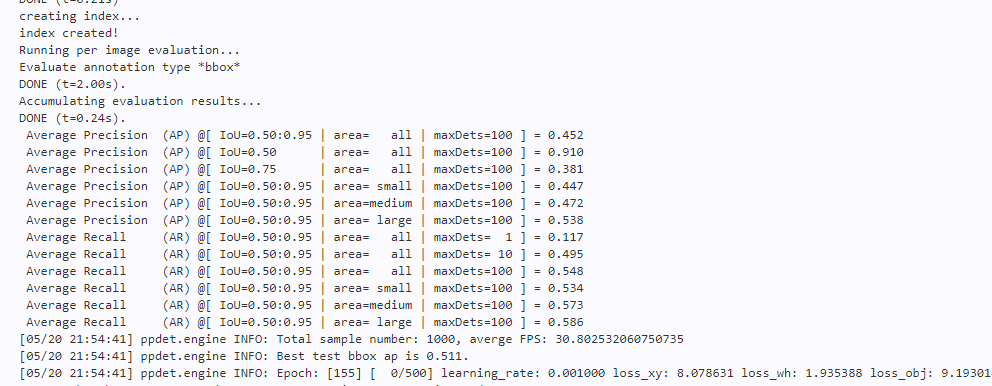
进行4.1步的操作之后：











在128轮的时候到达历史最高值：

Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.116

Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.474

Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.518

Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.454

Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.617

Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.703

[05/20 21:15:36] ppdet.engine INFO: Total sample number: 1000, averge FPS: 31.531612228522366

[05/20 21:15:36] ppdet.engine INFO: Best test bbox ap is 0.511.

到达0.511后又进行了大几十轮的训练，在测试集上的ap没有突破。

## 5 实验结果分析与感想

### 5.1实验结果

1. 构建了有实用意义的头盔识别模型

现可以直接调用训练好的模型进行推导（所谓推导不过就是对模型在训练集测试集以外进行使用的过程），对于图片中helmet和head两类对象的识别率较高而且框选区域大小适中准确，并生成包含所有识别框bbox.json字典文件，方便后续进行数据分析和训练其他模型使用。

1. 熟悉了比较正式的人工智能项目的流程

完成了人工智能课程的学习，非常感谢老师精彩的讲解和帮助我的同学们。

### 5.2实验感想

在构建、训练和使用基于多层感知器的MNIST手写数字识别模型的过程中，我获得了深刻的实验感想。首先，通过实践，我深入了解了机器学习模型的构建流程，包括数据预处理、模型定义、训练和评估等步骤。其次，通过调整模型的架构、超参数和优化算法，我发现这些选择对模型性能的影响巨大，需要进行仔细的调整和优化。此外，训练过程中的可视化和监控对于理解模型的学习情况至关重要，帮助我及时发现问题并进行调整。最后，在使用训练好的模型进行预测时，我对模型的准确性和泛化能力有了更深入的认识，也进一步体会到了机器学习在实际应用中的潜力和局限性。通过这个实验，我不仅加深了对机器学习原理的理解，也提升了自己的编程和问题解决能力。