**端侧c++版本的mAP计算**

目录

[初步版 1](#_Toc143074799)

[**最终版：** 2](#_Toc143074800)

[**总结：** 3](#_Toc143074801)

[**端侧c++版本的mAP计算程序使用介绍** 4](#_Toc143074802)

[**端侧mAP计算简单用法：** 4](#_Toc143074803)

[**端侧mAP计算整体使用流程：** 5](#_Toc143074804)

[**均值平均精度（mean average precision，mAP）计算逻辑：** 7](#_Toc143074805)

安全帽检测测试集

imgsz=(640, 640), # inference size (height, width)

conf\_thres=0.25, # confidence threshold

iou\_thres=0.45, # NMS IOU threshold

# 初步版

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验序号 | 推理方法 | mAP计算方法 | 输入的前处理 | Class 0 AP  (helmet) | Class 1 AP  (no\_helmet) | mAP |
| 1 | 飞书上记录的 | | 不做额外前处理 | 0.893 | 0.935 | 0.914 |
| 2 | 在桌面GPU采用原版YOLOv5（python） | 在桌面GPU采用原版YOLOv5的val.py（python） | Resize成1080p制作成视频，对应更改labels坐标信息 | 0.907 | 0.931 | 0.919 |
| 3 | 在桌面GPU采用原版YOLOv5（python） | 端侧c++ | Resize成1080p制作成视频，对应更改labels坐标信息 | 0.885388 | 0.848228 | 0.866808 |
| 4 | 端侧deepstream6.2 | 端侧c++ | Resize成1080p制作成视频，对应更改labels坐标信息 | 0.815713 | 0.835647 | 0.82568 |

小结：

1. 对比实验1和实验2。mAP基本一致，说明输入前处理是没问题的：即“Resize成1080p制作成视频，对应更改labels坐标信息”是正确的。

2. 对比实验2和实验3。mAP下降0.04左右，说明端侧c++计算mAP的程序在细节上可能还存在一些问题。

**针对第2点，修改置信度排序逻辑**

**阅读桌面云端yolo计算方法源码后确定其使用的是101-point interp (COCO)，即101点插值计算，相应修改端侧计算方式**

**最终版：**

**安全帽检测测试集**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验序号 | 推理方法 | mAP计算方法 | 输入的前处理 | Class 0 AP  (helmet) | Class 1 AP  (no\_helmet) | mAP |
| 1 | 飞书上记录的 | | 不做额外前处理 | 0.893 | 0.935 | 0.914 |
| 2 | 在桌面GPU采用原版YOLOv5（python） | 在桌面GPU采用原版YOLOv5（python） | Resize成1080p制作成视频，对应更改labels坐标信息 | 0.907 | 0.931 | 0.919 |
| 3 | 在桌面GPU采用原版YOLOv5（python） | 端侧c++ | Resize成1080p制作成视频，对应更改labels坐标信息 | 0.942427 | 0.923167 | 0.932797 |
| 4 | 端侧deepstream6.2 | 端侧c++ | Resize成1080p制作成视频，对应更改labels坐标信息 | 0.829247 | 0.922482 | **0.875864** |

**总结：**

1. **对比实验2和3。**

所有条件（推理方法，推理结果）一致的情况下，端侧计算方法所得结果比云侧高百分之一。

结果在误差容许范围内，mAP计算程序可以正常使用。

**产生误差的可能原因：**

1. 数值计算精度： C++和Python在处理浮点数时可能存在微小的差异，这可能会影响计算结果。
2. 编译优化： C++编译器可能会对代码进行优化，从而提高执行效率和计算精度。
3. 库和依赖： 可能在C++和Python实现中使用了不同的库和依赖项，这可能会影响结果。
4. **对比实验3和4。**

端侧deepstream6.2使用fp16的模型推理结果mAP为0.875864，

飞书上记录的，云侧推理的mAP为0.914，

端侧比云侧的mAP下降约0.04

**端侧c++版本的mAP计算程序使用介绍**

**端侧mAP计算简单用法：**

默认计算yolo云侧推理结果的mAP：

cd /home/nvidia/Downloads/helmet-test/val\_images/

./mapchange5-yolocal-101

如果需要计算其他推理结果（比如端侧，在map7-3.cpp的379行附近修改输入文件路径）或者需要修改mAP计算方式，修改mAP源码后重新编译再执行：

mAP源码也在这个目录下，即map7-3.cpp

编译：

g++ map7-3.cpp -o mapchange5-yolocal-101

执行：

./mapchange5-yolocal-101

**端侧mAP计算整体使用流程：**

1. **制作视频和对应的标签**

端侧推理的输入是1080p的视频，所以需要将图片制成1080p视频，并对应修改groundtruth的label的坐标信息（因为改成1080p受resize和padding的影响）

1.1 resize+padding程序文件使用，会将图片尺寸修改，同时修改原label。输入文件夹路径和输出文件夹路径可以在padimage-label3.py中修改。

cd /home/nvidia/Downloads/helmet-test/val\_images/

python3 padimage-label3.py

（如果需要可视化新的label的效果，可以使用/home/nvidia/Downloads/helmet-test/val\_images/remake-image-label4-blank/visualize.py ，相应修改里面的输入输出路径即可）

* 1. 在输出文件夹里会有images和labels两个文件夹，cd进入到此输出文件夹下,拷贝rerange.py到此文件夹并执行，用以给images和labels中的文件按照自然数序列重新命名（用以生成视频），会生成对应的两个新文件夹imagesnew和labelsnew。

cp /home/nvidia/Downloads/helmet-test/val\_images/rerange.py

python3 rerange.py

* 1. cd 进入imagesnew文件夹，执行命令将图片生成视频。

ffmpeg -r 2 -i %d.jpg out.mp4

（会生成out.mp4视频文件）

（-r 2 表示以每秒两帧的速度制成视频）

1. **端侧推理获取预测结果prediction文件夹**

以下是使用deepstream5.1生成的结果，预测结果不可靠，漏检错检很多。真正能用的是deepstream6.2生成的结果，需要从飞哥的机器上拷贝代码。

但是可以从以下deepstream5.1示例中学习使用方法：

使用root权限修改/opt/seaway/algorithm\_config.json中的"video\_url": <file:///home/nvidia/Downloads/helmet-test/val_images/remake-image-label2-blank/image100-testmulti/out.mp4> 将其修改为新生成的视频文件的路径。

进入/home/nvidia/4lujunda-firstchange-mychange/newseawaystream626/SeawayStream

执行 ./run.sh 会生成预测结果文件pre102文件夹

（在/home/nvidia/4lujunda-firstchange-mychange/newseawaystream626/SeawayStream/ deepstream\_app\_main.cpp中约457行位置可以修改输出文件夹路径）

**3.mAP计算需要读取groundtruth文件夹和预测结果prediction文件夹**

默认计算yolo云侧推理结果的mAP：

cd /home/nvidia/Downloads/helmet-test/val\_images/

./mapchange5-yolocal-101

如果需要计算其他推理结果（比如端侧，在map7-3.cpp的379行附近修改输入文件路径）或者需要修改mAP计算方式，修改mAP源码后重新编译再执行：

mAP源码也在这个目录下，即map7-3.cpp

编译：

g++ map7-3.cpp -o mapchange5-yolocal-101

执行：

./mapchange5-yolocal-101

**均值平均精度（mean average precision，mAP）计算逻辑：**

1.数据准备： 首先，需要有一个包含标注信息的测试数据集。对于目标检测任务，每个图像中的物体位置和类别都应该被标注。

2.模型预测： 使用训练好的目标检测模型对测试集中的图像进行预测，得到物体的检测结果。每个检测结果通常包括物体的类别、置信度分数以及边界框的位置。

3.置信度排序： 对每个类别的检测结果按照置信度从高到低进行排序。这是为了确保在计算精度时，优先考虑置信度较高的检测结果。

4.计算精度和召回率： 对于每个类别，根据排序后的检测结果，计算不同阈值下的精度（Precision）和召回率（Recall）。精度表示在所有被模型预测为正例的样本中，真正例的比例。召回率表示在所有实际正例中，被模型正确预测为正例的比例。

5.计算平均精度： 对于每个类别，根据精度-召回率曲线计算该类别的平均精度（AP）。AP是精度在召回率变化时的平均值，可以通过计算精度-召回率曲线下的面积来获得。然后，所有类别的AP值进行平均，得到mAP。

6.输出结果： 最终输出每个类别的AP值以及整体的mAP值作为模型在测试数据集上的性能指标。