# YOLOV5模型移植研究

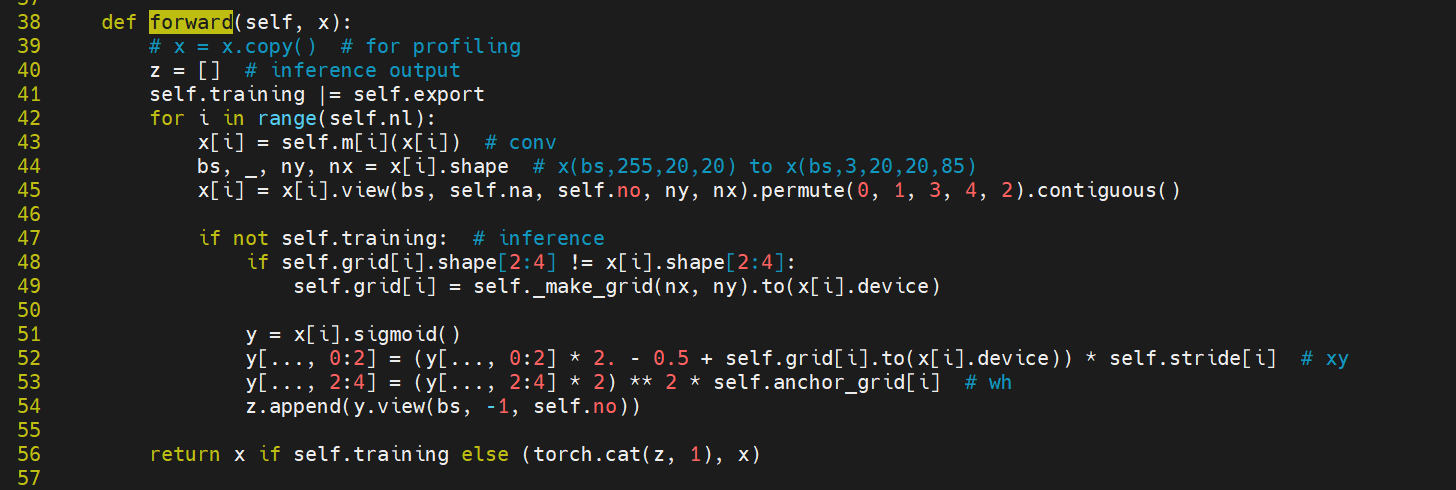
1. 模型导出

YOLOv5导出的输出，不同版本的输出不同，主要取决于model/yolo.py文件中的class Detect的forward函数。

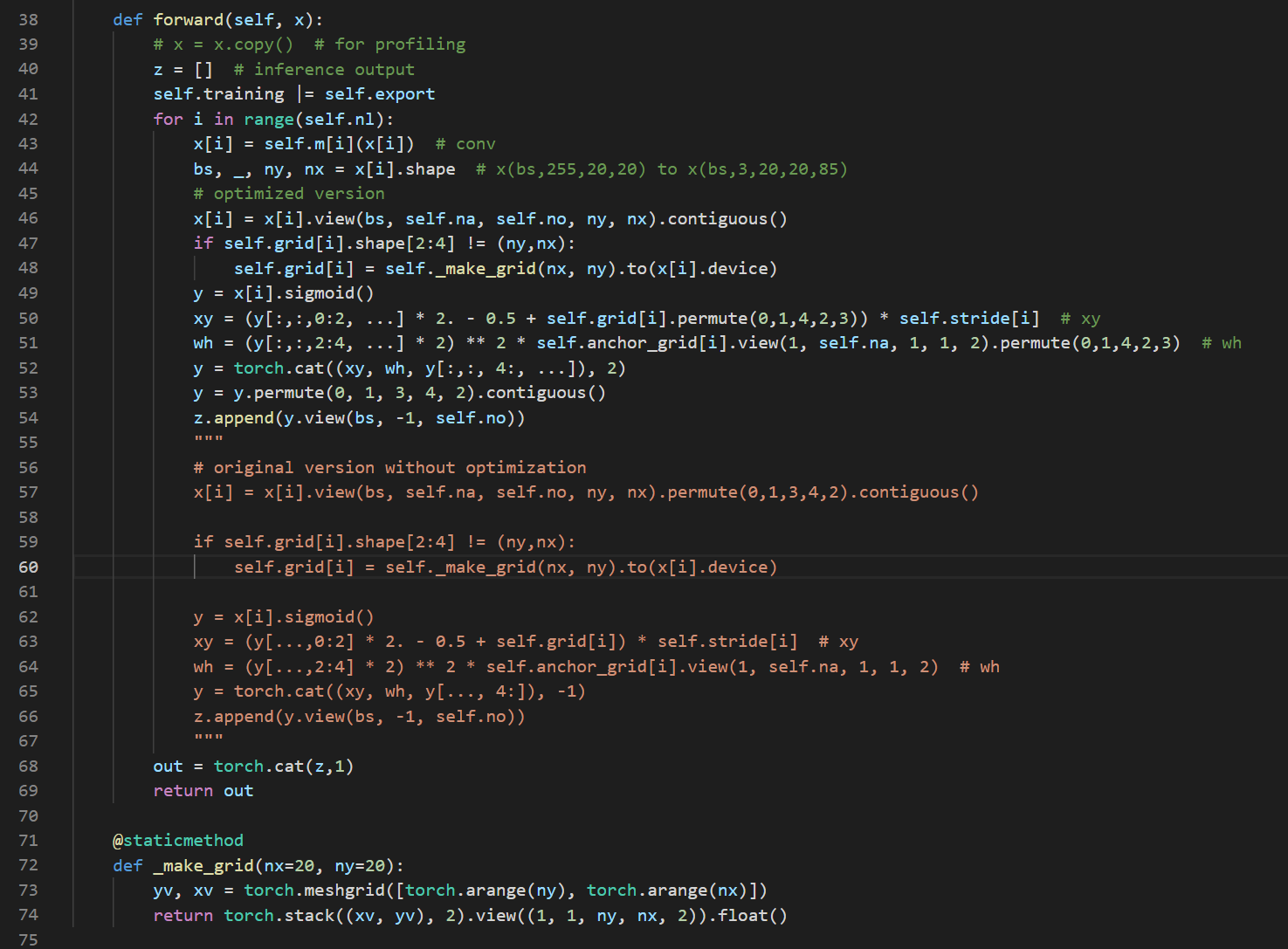
在v3.1版本（git checkout v3.1）和v4.0版本（git checkout v4.0）和v5.0版本（git checkout v5.0）时，由于52行和53行代码是inplace操作（等式左边出现了slice，直接对slice赋值），我们无法支持，**需要将56行改成return x即可**，这时就有了三个输出。

对于640x640的输入，输出的形状是[1,3,80,80,85], [1,3,40,40,85],[1,3,20,20,6]

此时在后处理过程中，**要在CPU上实现51行到54行计算逻辑和NMS处理**



当然也通过改model/yolo.py改成只有一个输出的情况，主要代码如下图所示。可以用附件中的yolo\_for\_v31.py文件替换工程中的model/yolo.py。对于640x640的输入，输出的形状是[1,25200,85]，这样后处理**只要在CPU上实现NMS处理即可**



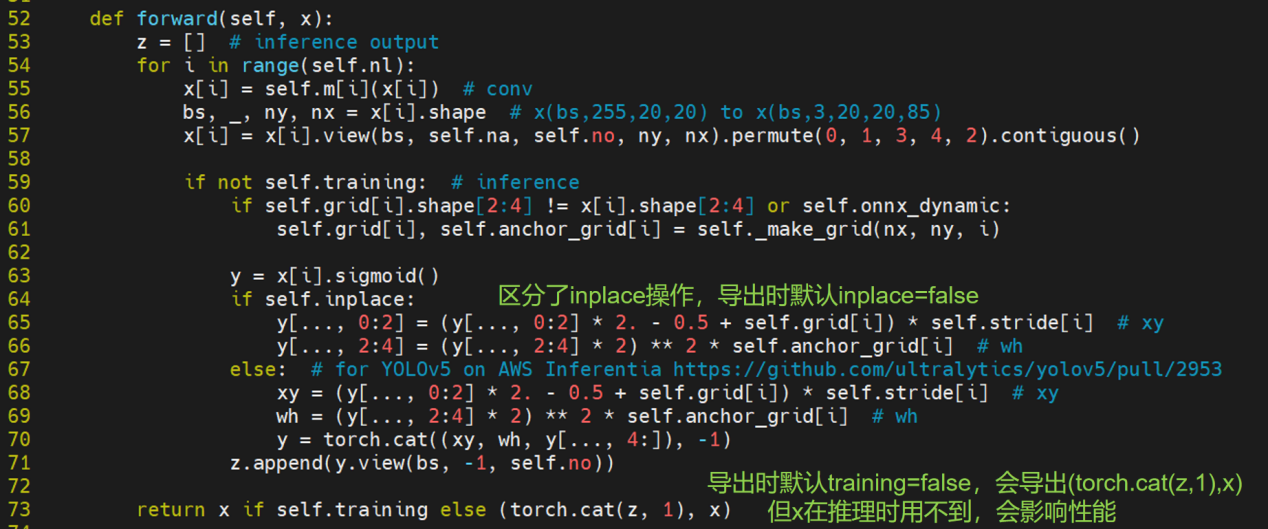
导出命令是

python3 models/export.py --weights yolov5s.pt --img-size 640 --batch-size 1

（注意v3.1好像只能在python3.5下转换成功，python3.7下报了段错误）

我们支持的模型是\*.torchscript.pt文件格式

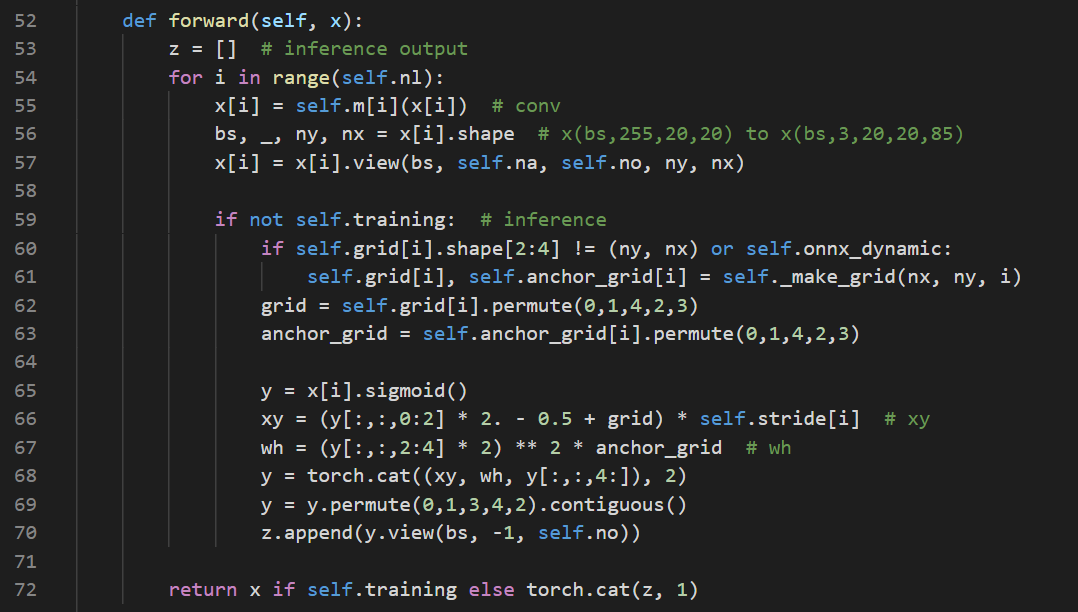
对于v6.0和最新的master分支的yolo.py代码有改动，如下所示，且导出脚本放在了根目录，加入了非inplace写法，导出前不用再改代码，模型会有4个输出，在后处理中，只要对最后一个输出做NMS就可得到最终结果。对于640x640输入来说，四个输出形状为[1,3,80,80,85], [1,3,40,40,85],[1,3,20,20,6],[1,25200,85]。但是其中前三个输出是没有用的，而且影响性能。测试时，四个输出的fp32模型在TPU上运行时间是36ms，而一个输出在TPU上运行时间是24ms。



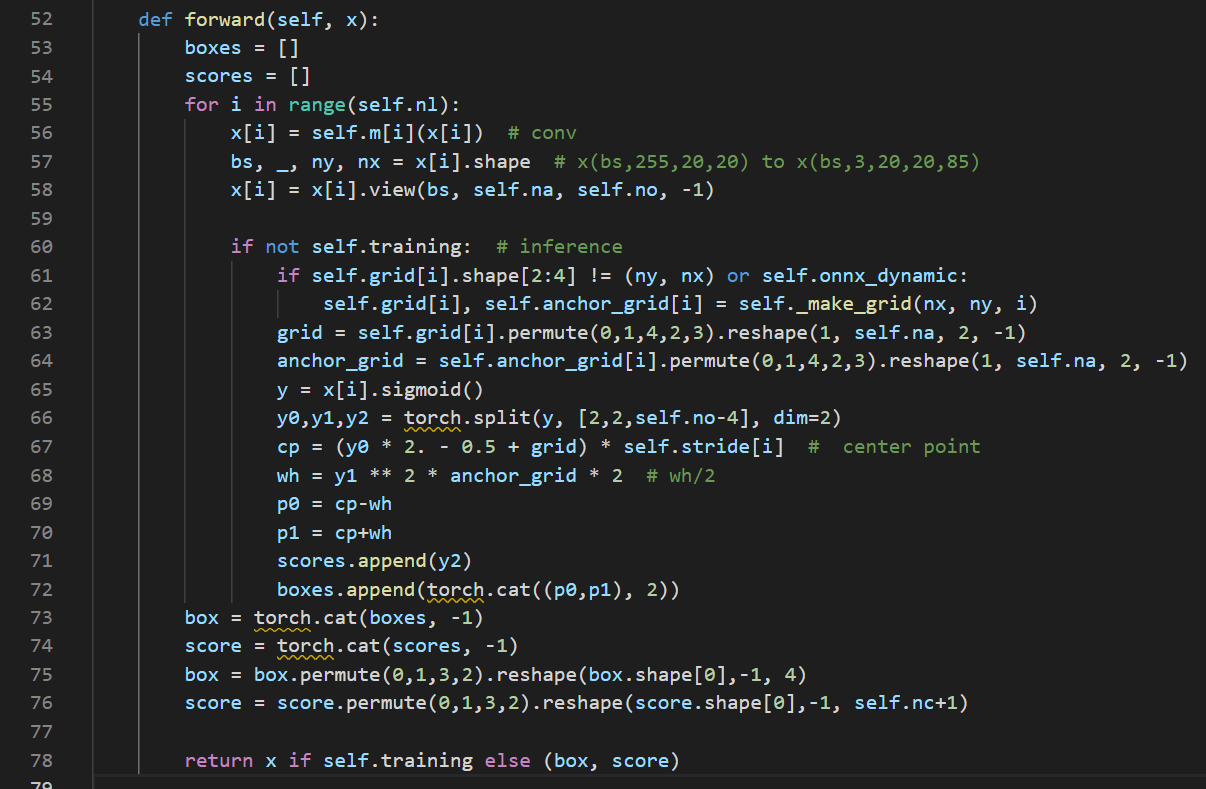
为得到最佳性能，yolo.py函数还是需要改写，最简单的办法是将73行改为:

”return x if self.training else torch.cat(z,1)“, 此时在fp32模型在TPU上运行时间为24ms

后面的输出还可以再优化，由于计算是在最后一维切分的，数据分散，会导致TPU性能有损失，可以考虑改写下，尽量保证后面数据块尽量连续，如下代码所示，输出也是一个的，fp32运行时间是23ms



如果考虑量化，由于最后是把还原的box和score结果concat到一起去，两者量化差异太大（box中x，y取值范围是0-640， w和h相对小些， score的取值范围是0-1），可以考虑把box和score分别作为输出，保证结果的分布一致性，代码改动如下



这时，网络的输出变成了两个，对于640x640的yolov5s的输出为box[1,25200,4]，score[1,25200,81]，为了数据范围的一致性，box的表示由原来的(cx,cy,w,h)改为了(left, top, right, bottom)。fp32的运行性能也有所提升，时间是22.5ms。后处理部分需要少量改动，

官方工程导出命令export放到了根目录下了

python3 export.py --weights yolov5s.pt --img-size 640 --batch-size 1

1. 模型转换命令

转fp32模型(--v和--enable\_profile可选)

python3 -m bmnetp --model yolov5s.torchscript.pt --shapes [[1,3,640,640]] --target BM1684 --v 4 --enable\_profile True

转umodel

python3 -m bmnetp --model yolov5s.torchscript.pt --shapes [[1,3,640,640]] --target BM1684 --v 4 --mode GenUmodel

1. 模型量化

由于模型有不同的导出方式，分别为1个输出(包含anchors decode和score)，2个输出(包含anchors decode，但box和score分离)，3个输出(不包含anchors decode)

1. 1个输出(包含anchors decode和score)

推荐最后一层conv的用fp32进行推理

1. 2个输出情况(包含anchors decode，但box和score分离)

推荐score用int8，box用fp32

1. 3个输出情况(不包含anchors decode)

推荐全部用int8

1. 性能分析

Yolov5中的激活函数对整体网络有很大影响，当激活函数是hardswish或hardtanh（v3.1版本及以前）时，fp32运行时间是12ms，而如果是SiLU（v4.0版本以后），运行时间是24ms左右

如果模型不带后处理(3个输出的情况)，fp32的模型运行时间为20.6ms。但后处理时间有明显增加，至少会增加8ms

1. 性能和精度结果