**Darknet Yolov2/3 model转bmodel 并在**

**BM1880平台内作推理**

目录

[1 Darknet model（cfg 和weights）如何转bmodel. 3](#_Toc9849494)

[1.1 相关Tool的安装 3](#_Toc9849495)

[1.2 Darknet model（cfg 和weights）如何转bmodel. 4](#_Toc9849496)

[1.2.1 在没有自己Caffe 的prototxt 情况下的模型转换 4](#_Toc9849497)

[1.2.2 在自己生成Caffe 的prototxt 情况下的模型转换 5](#_Toc9849498)

[1.3 模型量化生成bmodel 5](#_Toc9849499)

[2 将bmodel 导入到darknet 中利用BM1880完成推理 6](#_Toc9849500)

[2.1 将自己的bmodel 换到下的目录 6](#_Toc9849501)

[2.2 更改darknet 运行的cfg 文件 6](#_Toc9849502)

[2.3 参考Readme 编译并运行 7](#_Toc9849503)

[3 INT8 模型精度的调优 8](#_Toc9849504)

目前我们提供了Yolov2/3 bmodel 在darknet 框架下作推理的demo 实例（<https://github.com/BM1880-BIRD/bm1880-ai-demo-program/tree/master/darknet-yolov2-object-classification-v2>），下面说明一下如何将Darknet 训练出来的yolov2/3 model转为bmodel以方便开发者做基于yolo 网络的客制化。

# Darknet model（cfg 和weights）如何转bmodel.

目前大部分的yolo网络的训练过程都是在Darknet 框架内完成的，但目前BM1880只能基于Caffe model 做INT8模型(bmodel)的转换. 所以有了Darknet model 第一件事情就是Darknet 转Caffe model. 下面步骤说明如何完成转换。

# 相关Tool的安装

笔者转换主要用到了下面两个git hub 上的开源Tool .

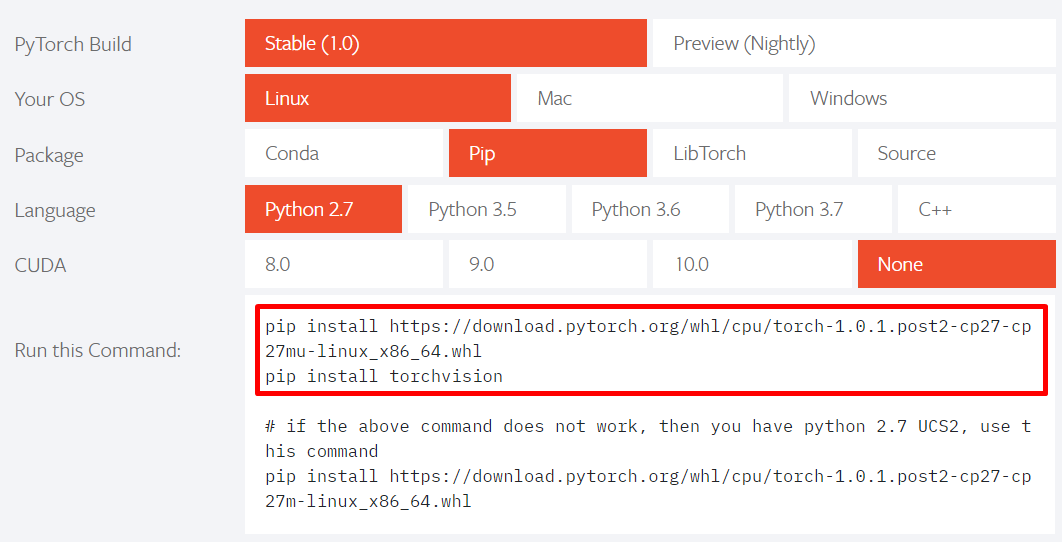
<https://github.com/ChenYingpeng/caffe-yolov3>

<https://github.com/marvis/pytorch-caffe-darknet-convert>

主要用到了前者中caffe-yolov3/yolov3\_darknet2caffe.py 做yolov3 到 caffe model的转换；用后者内的darknet2caffe.py做yolov2 到 caffe model的转换.

这两个py script 的成功运行需要先安装Pytorch 和Caffe . Pytorch笔者用的是pip 安装方式：请参考：<https://pytorch.org/>　按需求选择安装即可。

笔者的安装选择：



Caffe 的安装这里不详细描述（Caffe 需加入对upsample以及reorg两个layer 支持，网上可以自行查找）。对于Caffe 的安装不熟悉的用户，可以直接用我们提供的Calibration Tool(如下)内的编译好的Caffe.



由于darknet2caffe.py目前只支持tiny yolo model 转Caffe，少了对reorg和多参数route layer的转换支持。这里笔都加了如下的patch 以支持这两个layer.



对于yolov3 model, BM1880的calibration tool和最终的run time lib 会吃的upsample layer 的参数是size (而不是Scale）.所以需要加上如下的patch.



# Darknet model（cfg 和weights）如何转bmodel.

# 在没有自己Caffe 的prototxt 情况下的模型转换

**这种转换方式的方便之处是，不用再另外写Caffe 的prototxt ，只需要有darknet 下的cfg 和weights 文件即可。**

**注：请使用python2 做为python 版本运行所有以下的命令。**

首先将Caffe 的目录设置在PYTHONPATH环境变量或者设置在转换脚本里调用

sys.path.insert(). 以便成功import caffe.

笔者将caffe-yolov3/yolov3\_darknet2caffe.py 拷贝到pytorch-caffe-darknet-convert目录下以减化环境变量的设置。

yolov3和yolov2的model 转Caffe 分别执行如下命令，最终会生成Caffe 的prototxt和caffemodel.

* Yolov3:

python yolov3\_darknet2caffe.py yolov3.cfg yolov3.weights　yolov3.prototxt yolov3.caffemodel

* Yolov2:

python darknet2caffe.py yolov2.cfg yolov2.weights yolov2.prototxt yolov2.caffemodel

最终生成的caffemodel ，只需要保留到卷积layer做为output, 如下所示。



# 在自己生成Caffe 的prototxt 情况下的模型转换

如果有自己实现的Caffe model (prototxt), 可以用如下的python script 完成caffe model的生成。将yolov2.prototxt 和yolov2.weights放到相同的目录。

Yolov2:

python convert\_weights\_to\_caffemodel\_yolo2.py



Yolov3:

python convert\_weights\_to\_caffemodel\_yolo3.py



# 模型量化生成bmodel

用calibration tool 作量化并转换生成bmodel , 请参考如下的resnet实例。

<https://sophon-edge.gitbook.io/project/toolkit/bmnet-compiler>

**需要注意的是prototxt input layer 的写法，笔者转的是原生model ，这里要做data 归一化 (0.00392156862 = 1/255)(如下示例) ，客制化model 需要与原model train的时候的data 处理保持一致.**



# 将bmodel 导入到darknet 中利用BM1880完成推理

目前有提供Yolo model 在Darknet 下面的推理demo.

<https://github.com/BM1880-BIRD/bm1880-ai-demo-program/tree/master/darknet-yolov2-object-classification-v2>

# 将自己的bmodel 换到下的目录

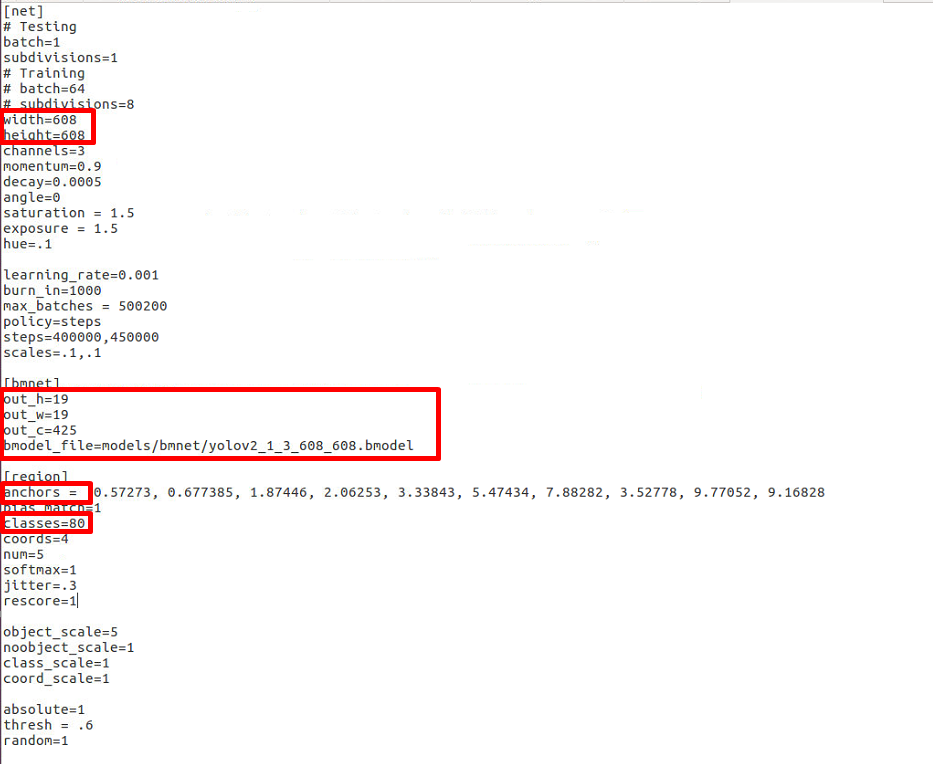
[darknet-yolov2-object-classification-v2/darknet/models/bmnet](https://github.com/BM1880-BIRD/bm1880-ai-demo-program/tree/master/darknet-yolov2-object-classification-v2/darknet/models/bmnet)

# 更改darknet 运行的cfg 文件

* Yolov2 model:

需要改动的是下面红框里的内容，需要根据自己的客制化模型参数做对应的修改。

darknet/cfg/bmnet\_yolov2.cfg



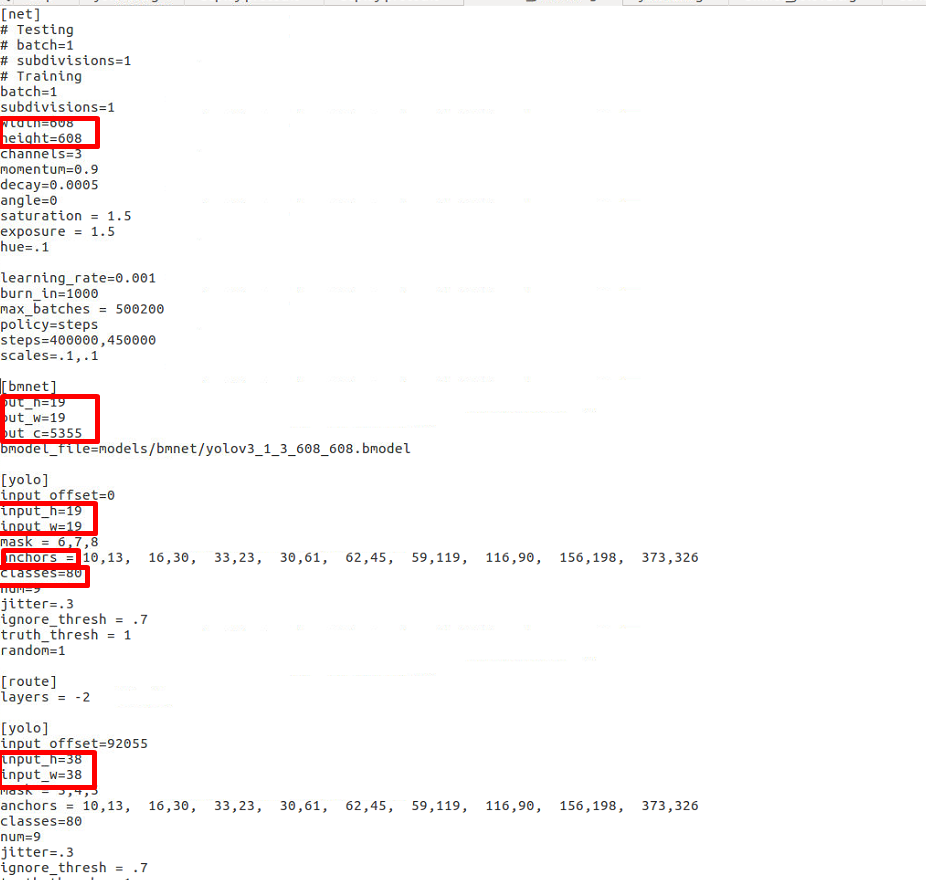
其中, out\_c = out\_c=5\*(4+1+classes), classes 是物体检测的种类数量。

Anchors 值要与darknet training 时一致

* Yolov3 model:

darknet/cfg/bmnet\_yolov3.cfg

下面红框内的部分需要注意一下修改：



在原生608模型中，out\_c = 19\*19\*255+38\*38\*255+76\*76\*255 /19\*19，其中255=３\*(4+1+80)，80是类别数量。Bmnet model的输出 = 19(out\_h)X19(out\_w)X5355(out\_c)

红框内的值都要依据自己的模型重新填写。我们把model output (三个conv 输出layer)放到一起，通过input\_offset去找对应的layer 输出。所以第一个input\_offset是０，第二个是：19\*19\*255, 第三个是19\*19\*255+38\*38\*255.

# 参考Readme 编译并运行

目前我们同时支持USB mode 和SOC mode 两种方式。

# INT8 模型精度的调优

以目前的已发生的情况来看，在按如上的步骤转换成INT8 yolo model都会遇到不同情况的精度损失。精度的调优对于最终产品落地也是不可缺少的一个环节。

目前有完成对于Yolov3的调优demo code. 请参考[Calibration Tool](#Calibration_Tool) 内的tuning\_tool　目录。

相关说明：

目前Demo code 里的调优input dataset 是用了原始model的coco dataset . 相关的模型评价也直接采用了coco data set 的interface计算mAP. 针对这部分，不同的用户在模型Training 的时候会有不同的dataset 和模型评价原则。

所以，具体调客制化model 精度的建议如下：(可任选其一)

* 将自己的data set 转成coco 的形式，直接套用demo的flow ，以mAP做为模型评估指标做Tuning . 具体修改如下的两个路径。



* 吃客制化模型的data set 和自己的模型评价指标，将这部分code 整合到demo里做调优。

具体可修改test\_accuracy.py内的如下两个function.

validate\_on\_coco

coco\_result