**BM1880 Darknet Yolov2/3 model转bmodel 说明**

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Author | Date | Description |
| 1.0 | Bitmain | 2019.06.10 | Initial Draft |
| 1.0.1 | Bitmain | 2019.07.01 | Attach new version calibration/tuning tool |
| 1.0.2 | Bitmain | 2019.07.05 | 添加了对模型精度有影响的操作注意事项 |
| 1.0.3 | Bitmain | 2019.07.12 | 修改了darknet　cfg 文件参数说明，增加fp32/int8 model的测试方法。 |
|  |  |  |  |
|  |  |  |  |

目录

[Revision History 2](#_Toc13834182)

[1 Darknet model（cfg 和weights）如何转bmodel. 4](#_Toc13834183)

[1.1 相关Tool的安装 4](#_Toc13834184)

[1.2 Darknet model（cfg 和weights）如何转bmodel. 5](#_Toc13834185)

[1.2.1 在没有自己Caffe 的prototxt 情况下的模型转换 5](#_Toc13834186)

[1.2.2 在自己生成Caffe 的prototxt 情况下的模型转换 6](#_Toc13834187)

[1.3 模型量化生成bmodel 6](#_Toc13834188)

[1.4 在caffe 跑测试程序测试转完的model 精度 7](#_Toc13834189)

[2 将bmodel 导入到darknet 中利用BM1880完成推理 8](#_Toc13834190)

[2.1 将自己的bmodel 换到下的目录 8](#_Toc13834191)

[2.2 更改darknet 运行的cfg 文件 8](#_Toc13834192)

[2.3 参考Readme 编译并运行 12](#_Toc13834193)

[3 INT8 模型精度的调优 12](#_Toc13834194)

[4 附件 12](#_Toc13834195)

[4.1 Calibration tool 12](#_Toc13834196)

[4.2 Auto tuning tool 12](#_Toc13834197)

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 .

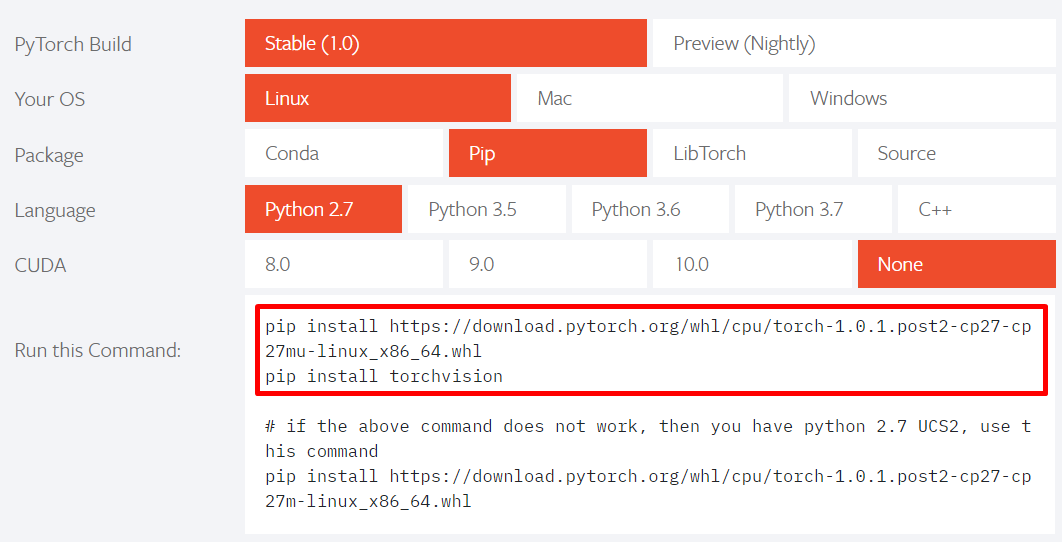
<https://github.com/ChenYingpeng/caffe-yolov3>

<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(calibration\_too2.1.rar 请参考附件章节)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。



**特别注意**：

对于转完的yolov3.prototxt, 里面有单input 的concat layer, 这个layer 要拿掉。否则会影响到INT8的推理。实例如下：



# 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>

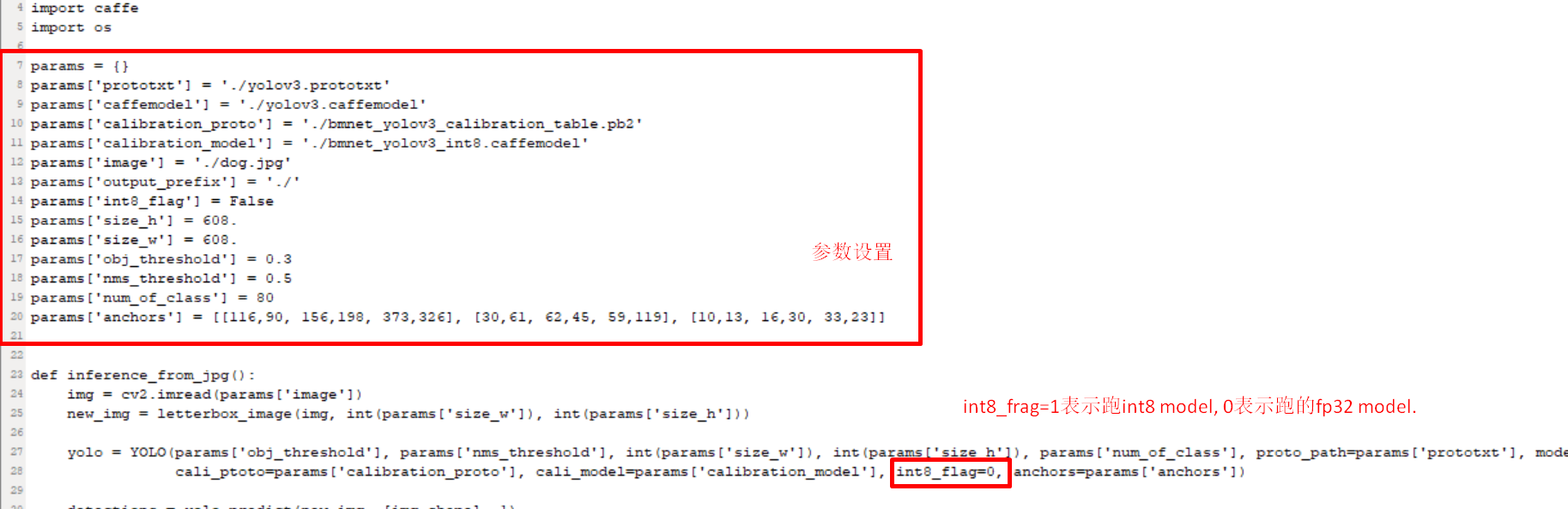
**需要注意的是prototxt input layer 的写法，笔者转的是原生model ，这里要做data 归一化 (0.00392156862 = 1/255)(如下示例) ，客制化model 需要与原model train的时候的data 处理保持一致.**



# 在caffe 跑测试程序测试转完的model 精度

目前在calibration\_tool2.1版本中(参考附件章节), 有包含yolov3的完成python 测试script. Release/samples/yolov3/test\_yolov3.py.

可以在test\_yolov3.py中按自己的model 设置相关的参数后进行测试fp32 caffe model及转换完的int8 model精度。

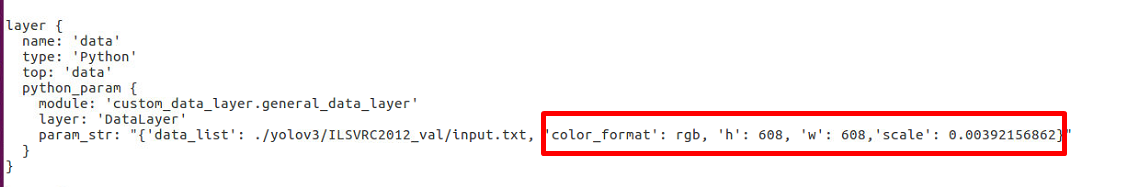


验证：

1. 跑fp32的caffe model的结果与darknet 中的结果进行比较，看是否在darknet2caffe的时候有精度掉。
2. 跑int8的结果与跑fp32的model的结果做比较，看是否在calibration int8的时候有精度掉。

经验分享：

1. 根据目前客户的反馈，上面1中darknet2caffe过程，精度一般都不会掉。
2. 一般转完的int8会有精度损失。这个时候视情况首先检测相当的设置是否对，几个点需要检查
3. 测试程序中prototxt 里input layer type是否改成了input. 只有这样测试程序里才会默认做input data 量化。
4. 转caffe model后，prototxt中是否有手动拿掉单input concat；
5. Calibration的时候，input data 参数是否有设置对，比如下面红框里的设置。



跑完这一步，如果int8的精度也满足了，就可以接下来第2章的内容。否则，需要转到第3章节做精度调优。

# 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:

darknet/cfg/bmnet\_yolov2.cfg. 下面中为yolov2 608实例，红色加粗字体需要根据自

己的模型进行修改。参数如何设置都有相关的注释说明。

[net]

# Testing

batch=1

subdivisions=1

#模型输入的宽高

**width=608**

**height=608**

channels=3

momentum=0.9

decay=0.0005

angle=0

saturation = 1.5

exposure = 1.5

hue=.1

learning\_rate=0.001

burn\_in=1000

max\_batches = 500200

policy=steps

steps=400000,450000

scales=.1,.1

[bmnet]

#模型输出的宽高

**out\_h=19**

**out\_w=19**

# 模型输出的channel数，out\_c=5\*(4+1+classes), classes 是物体检测的种类数量。

**out\_c=425**

#转完的bmodel路径

**bmodel\_file=models/bmnet/yolov2\_1\_3\_608\_608.bmodel**

[region]

#anchors 要与模型训练时保持一致，注意顺序

**anchors = 0.57273, 0.677385, 1.87446, 2.06253, 3.33843, 5.47434, 7.88282, 3.52778, 9.77052, 9.16828**

bias\_match=1

#检测物体种类数

**classes=80**

coords=4

num=5

softmax=1

jitter=.3

rescore=1

object\_scale=5

noobject\_scale=1

class\_scale=1

coord\_scale=1

absolute=1

thresh = .6

random=1

* Yolov3 model:

darknet/cfg/bmnet\_yolov3.cfg. 下面中为yolov3 608实例，红色加粗字体需要根据自

己的模型进行修改。参数如何设置都有相关的注释说明。

[net]

batch=1

subdivisions=1

#模型input 宽高

**width=608**

**height=608**

channels=3

momentum=0.9

decay=0.0005

angle=0

saturation = 1.5

exposure = 1.5

hue=.1

learning\_rate=0.001

burn\_in=1000

max\_batches = 500200

policy=steps

steps=400000,450000

scales=.1,.1

[bmnet]

#out\_w/h模型最终输出的宽高，out\_h=模型输入高/32 , out\_w=模型输入宽/32

**out\_h=19**

**out\_w=19**

#out\_c是bmnet的输出的channel数。计算方法(19\*19\*255+38\*38\*255+76\*76\*255) /19\*19，

#bmnet 的输出 = out\_h\*out\*width\*out\_c

#255=3\*(4+1+80), 4是输出的bbox的x/y/width/height，1是box的置信度，80是检测类#别数，3是共有3个尺度的框。

**out\_c=5355**

#转完的bmodel路径。

**bmodel\_file=models/bmnet/yolov3\_1\_3\_608\_608.bmodel**

[yolo]

#yolo layer会从bmnet 的输出中通过input\_offset去抓对应位置的数据最终算出检测结果。

#第一个yolo offset 设置为0

**input\_offset=0**

#第一个尺度框的大小。

**input\_h=19**

**input\_w=19**

#mask 如果用的anchor是9个就不用变。如果是anchor有变要重新设置.例：如果anchor数量是15，这里就是10,11,12,13,14. (总数分三组的最后一组,0开始)

**mask = 6,7,8**

#anchor box设置需要与模型training 里，保持一置，但要注意设置的顺序。

**anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326**

#Total 类别

**classes=80**

#anchor 数量

**num=9**

jitter=.3

ignore\_thresh = .7

truth\_thresh = 1

random=1

[route]

layers = -2

[yolo]

#第二个yolo layer的offset , 19\*19\*255=92055

**input\_offset=92055**

#第二个尺度框的大小

**input\_h=38**

**input\_w=38**

#mask 如果用的anchor是9个就不用变。如果是anchor有变要重新设置.例：如果anchor数量是15，这里就是5,6,7,8,9(总数分三组的最后二组,0开始)

**mask = 3,4,5**

**anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326**

**classes=80**

**num=9**

jitter=.3

ignore\_thresh = .7

truth\_thresh = 1

random=1

[route]

layers = -4

[yolo]

#第二个yolo layer的offset , 19\*19\*255+38\*38\*255=460275

**input\_offset=460275**

#第三个尺度框的大小

**input\_h=76**

**input\_w=76**

#mask 如果用的anchor是9个就不用变。如果是anchor有变要重新设置.例：如果anchor数量是15，这里就是0,1,2,3,4(总数分三组的最后三组,0开始)

**mask = 0,1,2**

**anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326**

**classes=80**

**num=9**

jitter=.3

ignore\_thresh = .7

truth\_thresh = 1

random=1

# 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。

精度调整的工具能参考附件章节, auto\_tunning.rar.

相关说明：

1. 调整的主文件是main.py .
2. 具体怎么调用请参考main.sh.
3. 参数的具体含义，AutoTuningToolDoc.docx里有详细的说明。
4. 这个Tool会用到caffe , 可以直接使用calibration tool2.1 中的caffe.
5. 调整精度的过程比较慢，建议将GPU caffe的环境搭建起来。

# 附件

# Calibration tool



亦可以在如下的link中获取：

https://github.com/BM1880-BIRD/bm1880-ai-demo-program/blob/master/darknet-yolov2-object-classification-v2/tool.rar

# Auto tuning tool



亦可以在如下的link中获取：

https://github.com/BM1880-BIRD/bm1880-ai-demo-program/blob/master/darknet-yolov2-object-classification-v2/tool.rar