Calibration Tool Guide

文档版本 v1.1

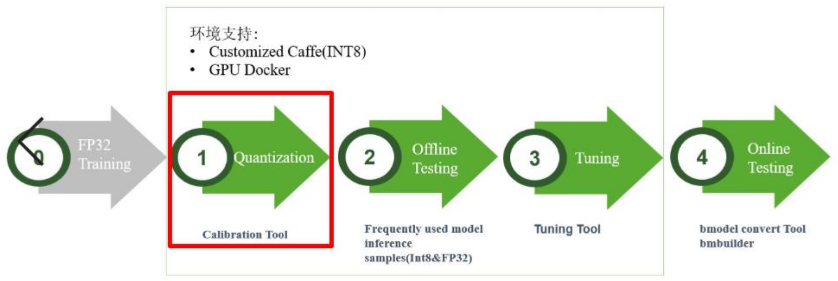
发布时间 2019-09-25

目录

1. 前言

概述

本文档说明量化工具 Calibration Tool 的使用方法和注意事项。Calibration Tool 用于将 Caffe FP32 模型量化到 INT8 模型.



读者对象

本文档主要适用于以下工程师：

- 算法工程师
- 系统整合工程师

修订记录

版本号	作者	审阅人	时间	修改描述
1.1	褚洪君		2019.09.25	• 重新整理各章节的逻辑 . • 增加量化实例部分
1.0	汤道言、褚洪君	朱晓军、王亮	2019.09.19	初版

2. 准备工作

我们**强烈建议**您安装 GPU docker 环境，这样会大幅度提升模型处理的效率。相关的工作(Tuning)，用 CPU 几乎无法完成。

2.1. Host PC 环境

- ubuntu 18.04 or 16.04
- python2.7
- docker installed
- nvidia-docker installed (if you want to setup GPU Docker)

2.2. GPU Docker 环境建立(推荐)

目前我们只提供基于 CUDA9.0 的 prebuilt Caffe 库。所以请确认您主机上的 GPU 显卡驱动是与 CUDA9.0 兼容的。请按如下步骤建立环境:

1. 请把 cafffe_gpu 目录里的库文件覆盖到 caffe 目录。
2. build & run docker image

```
sudo nvidia-docker build -t bmcalibration:2.0_gpu -
f docker/Dockerfile_GPU .
sudo nvidia-docker run -v /workspace:/workspace -it bmcalibration:2.0_gpu
```

这样，GPU Docker 环境即建立完成。请调用如下的 caffe api 来使能 GPU caffe.

```
caffe.set_mode_gpu()
caffe.set_device(device_id)    #device_id means the GPU id on your machine.
```

2.3. CPU Docker 环境建立

请按如下命令建立环境:

```
sudo docker build -t bmcalibration:2.0_cpu -f docker/Dockerfile_CPU .
sudo docker run -v /workspace:/workspace -it bmcalibration:2.0_cpu
```

这样，CPU Docker 环境即建立完成。

3. 如何使用 Calibration Tool

3.1. 细节解析

Calibration 的具体实现包装在 Calibration 类中。具体实现在 calibration.py 里。

```
def run_calibration(args):
    calibration_info = {
        "model_name": args.model_name,
        "in_prototxt": os.path.join(args.model_path,'deploy.prototxt'),
        "in_caffemodel": os.path.join(args.model_path,'custom.caffemodel'),

        "iteration": 1,
        "enable_memory_opt": args.memory_opt,
        "enable_calibration_opt": 1,
        "histogram_bin_num": 2048,
        "math_lib_path": './lib/calibration_math.so',
        "enable_concat_quantize": 0

    }
```

要结合具体模型的需求，设置 calibration_info 内的参数，具体的参数解析如下：

Parameter	Description
model_name	The name of your model
in_prototxt	The input caffe prototxt
in_caffemodel	The input caffemodel

iteration	How many images you want to use for calibration.
enable_calibration_opt	Optimize calibration procedure. Some specific models(ssd is one of them) may encounter errors when use this flag .
enable_memory_opt	To open the memory optimization. It reduces memory usage a lot but takes a little more time. Open this option when you are facing the out of memory problem. If there is ‘LRN’ or ‘Normalize’ layer in your model, you should not use this option.
histogram_bin_num	The histogram bin amount of KLD algorithm. Bigger may be more accurate but takes more times. Note that bigger bin number may need more calibration data. Need to be a multiple of 128. If not pass, the default is 2048.
enable_concat_quantize	Enable Concat layer quantization. It may improve accuracy(please enable it for yolo model).

3.2. 量化数据的处理

量化数据用于计算数据 histogram for KL divergence 以确认最适合的 FP32 转 INT8 过程中各模型 layer 的 threshold. 所选择的图片数据最好与实际场景的场景分布相似，并尽量全部覆盖相关的场景。

目前我们是通过将量化数据给到 deploy.protxt 的 data layer 实现数据输入的。目前支持两种方式：

1. 通常的 caffe data layer 比如 lmdb 形式 .

对于 lmdb 形式，需要先转成 lmdb 格式再写到 prototxt data layer 中。实例如下：

```
layer {
      name: "data"
      type: "Data"
      top: "data"
      include {
      phase: TEST
      }
# The preprocess of your data
      transform_param {
#mirror: false
      #crop_size: 224
      #mean_file: "./imagenet_mean.binaryproto"
      }
data_param {
      # The lmdb file path
      source: "./calibration/ilsvrc12_val_lmdb"
      batch_size: 1
      backend: LMDB
      }
}
```

2. 客制化 python data layer.

实例可参考这个文件: “samples/detection/yolov3/yolo_data_layer.py”.具体的 data layer 写法如下:

```
layer {
    name: 'data'
    type: 'Python'
    top: 'data'
    python_param {
      module: 'yolov3.yolo_data_layer'
      layer: 'YoloDataLayer'
      param_str: "{'image_list': ./input.txt, 'hw': 608}"
    }
}
```

注意:

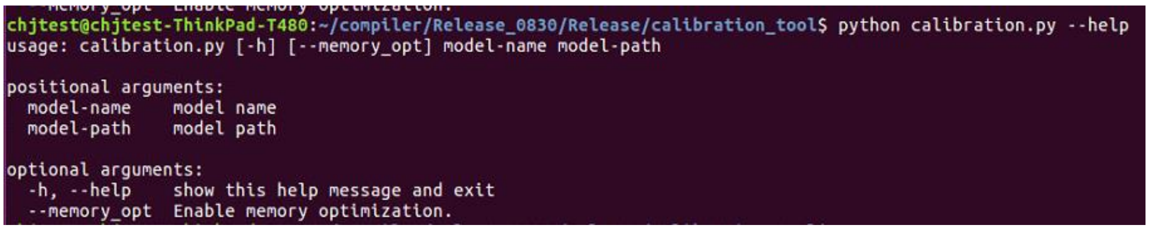
1) 以上两种方式, input layer name 都要写为”data” .

2) Data Input layer 需要与 training model 时做的 preprocess 一致。以我们提供的 general_data_layer.py 为例，支持如下的前处理方式：

Parameter	**Description**
data_list	The file describes where to find the images. Each line is the path to image.
h	input data height
w	input data width
color_format	Specify which color format you want to use. Only support RGB/BGR.
r/g/b_mean	RGB mean value. If > 0, all image will subtract the mean value.
scale	The scale value of the data. It will multiply to the data after minus the r/g/b mean value. The default value is 1
mirror	Specify the data needs to mirror or not. The default value is 0. • 0: no need to mirror • 1: vertical and horizontal • 2: vertical • 3: horizontal
transpose	Specify the data transpose axises. The default is [2,0,1] (equal to [c,h,w] order).
padding	If true, padding is added when resize to the target size to keep the original aspect ratio. The default is false.

3.3. 运行 Calibration Tool

calibration.py 是 calibration tool 的入口。可以参考 help 指引您的操作如下:



```
  --memory_opt  Enable memory optimization.
chjtest@chjtest-ThinkPad-T480:~/compiler/Release_0830/Release/calibration_tool$ python calibration.py --help
usage: calibration.py [-h] [--memory_opt] model-name model-path

positional arguments:
  model-name    model name
  model-path    model path

optional arguments:
  -h, --help    show this help message and exit
  --memory_opt  Enable memory optimization.
```

Parameter	Description
model-name	The model type name of your model.
model-path	model path in which you put your model files (deploy.prototxt and custom.caffemodel) and also generated output files will be in it.
--memory_opt	[optional] To open the memory optimization. It reduces memory usage a lot but takes a little more time. Open this option when you are facing the out of memory problem. If there is ‘LRN’ or ‘Normalize’ layer in your model, you should not use this option.

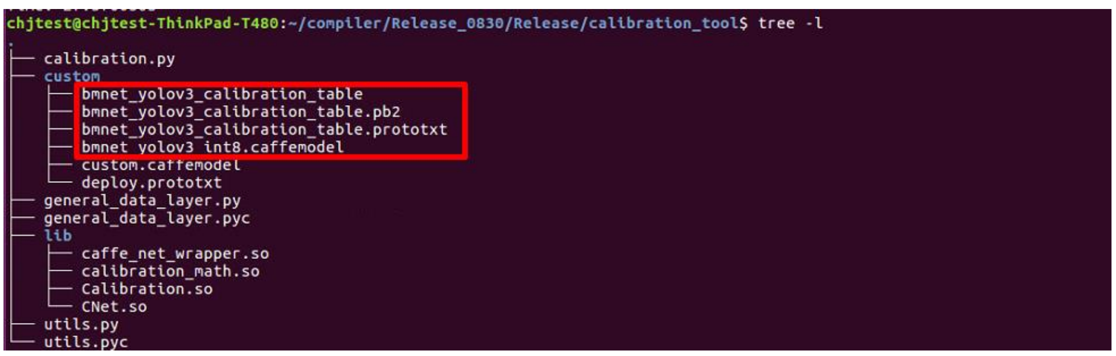
请参考如下步骤完成 calibration.

1. 如 3.2 章节说明的重写 caffe prototxt 中的 input layer 并命名为 ‘deploy.prototxt’
2. 命令 weight 文件为 ‘custom.caffemodel’
3. 把 deploy.prototxt 及 custom.caffemodel 放到你的 model 目录(e.g. custom)

4. 运行 calibration.py 如下 (以 yolov3 model 为例).

   python calibration.py yolov3 custom --memory

   运行的结果如下，红色框里的内容为生成的 INT8 model 文件。



```
chjtest@chjtest-ThinkPad-T480:~/compiler/Release_0830/Release/calibration_tool$ tree -l
.
├── calibration.py
├── custom
│   ├── bmnet_yolov3_calibration_table
│   ├── bmnet_yolov3_calibration_table.pb2
│   ├── bmnet_yolov3_calibration_table.prototxt
│   ├── bmnet_yolov3_int8.caffemodel
│   ├── custom.caffemodel
│   └── deploy.prototxt
├── general_data_layer.py
├── general_data_layer.pyc
├── lib
│   ├── caffe_net_wrapper.so
│   ├── calibration_math.so
│   ├── Calibration.so
│   └── CNet.so
├── utils.py
└── utils.pyc
```

4. 量化实例

本章节以量化 Resnet50 为例，详细说明每一个步骤，以供您参考。

1. 准备 FP32 caffemodel 以及 prototext 及量化数据

目录结构如下:

```
chjtest@chjtest-ThinkPad-T480:~/compiler/Edge-Development-
Toolchain/calibration_tool/Resnet50$ tree -l
.
├── custom.caffemodel
├── deploy.prototxt
├── ILSVRC2012_val
│   ├── ILSVRC2012_val_00000077.JPEG
│   ├── ILSVRC2012_val_00000169.JPEG
│   ├── ILSVRC2012_val_00000179.JPEG
│   ├── ILSVRC2012_val_00000185.JPEG
│   ├── ILSVRC2012_val_00000201.JPEG
│   ├── ILSVRC2012_val_00000297.JPEG
│   ├── ILSVRC2012_val_00000457.JPEG
│   ├── ILSVRC2012_val_00000488.JPEG
│   ├── ILSVRC2012_val_00000558.JPEG
│   ├── ILSVRC2012_val_00000576.JPEG
│   ├── ILSVRC2012_val_00000649.JPEG
│   ├── ILSVRC2012_val_00000745.JPEG
```

ILSVRC2012_val 目录内是提供量化用的图片。可以通过本实例参考 prototxt 内 input data layer 的写法.

注意:

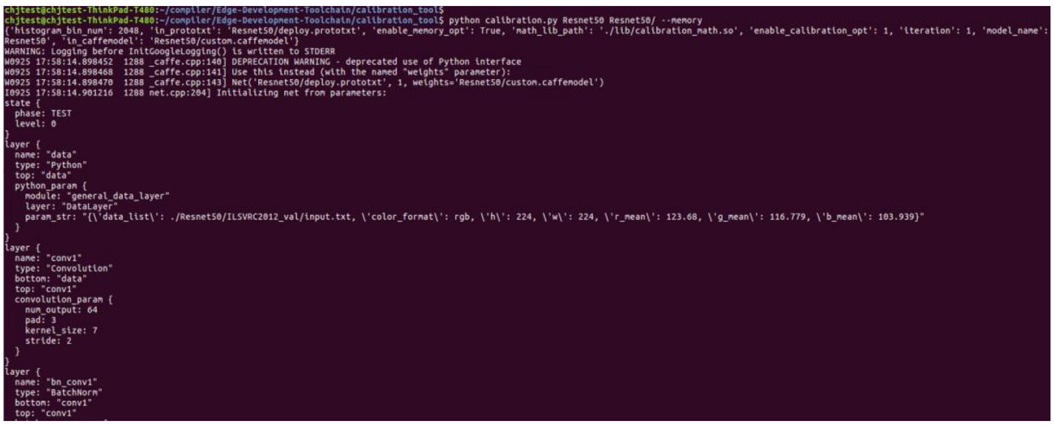
图片目录下面有 input.txt 文件，里面是图片的绝对路径，在您做实验的时候，要改为你本机路径。可以参考如下命令生成 input.txt:

readlink -f * > input.txt

2. 执行量化脚本

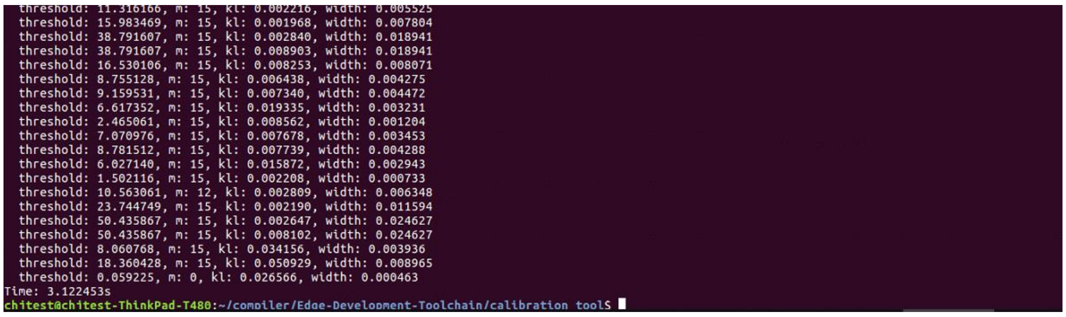
```
chjtest@chjtest-ThinkPad-T480:~/compiler/Edge-Development-
Toolchain/calibration_tool/Resnet50$ python calibration.py Resnet50/ --
memory
```

Log 输出如下:



```
chjtest@chjtest-ThinkPad-T480:~/compiler/Edge-Development-Toolchain/calibration_tool$
chjtest@chjtest-ThinkPad-T480:~/compiler/Edge-Development-Toolchain/calibration_tool$ python calibration.py Resnet50 Resnet50/ --memory
{'histogram_bin_num': 2048, 'in_prototxt': 'Resnet50/deploy.prototxt', 'enable_memory_opt': True, 'math_lib_path': './lib/calibration_math.so', 'enable_calibration_opt': 1, 'iteration': 1, 'model_name':
Resnet50', 'in_caffemodel': 'Resnet50/custom.caffemodel'}
WARNING: Logging before InitGoogleLogging() is written to STDERR
W0925 17:58:14.898452  1288 _caffe.cpp:140] DEPRECATION WARNING - deprecated use of Python interface
W0925 17:58:14.898468  1288 _caffe.cpp:141] Use this instead (with the named "weights" parameter):
W0925 17:58:14.898470  1288 _caffe.cpp:143] Net('Resnet50/deploy.prototxt', 1, weights='Resnet50/custom.caffemodel')
I0925 17:58:14.901216  1288 net.cpp:204] Initializing net from parameters:
state {
  phase: TEST
  level: 0
}
layer {
  name: "data"
  type: "Python"
  top: "data"
  python_param {
    module: "general_data_layer"
    layer: "DataLayer"
    param_str: "{\'data_list\': ./Resnet50/ILSVRC2012_val/input.txt, \'color_format\': rgb, \'h\': 224, \'w\': 224, \'r_mean\': 123.68, \'g_mean\': 116.779, \'b_mean\': 103.939}"
  }
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  convolution_param {
    num_output: 64
    pad: 3
    kernel_size: 7
    stride: 2
  }
}
layer {
  name: "bn_conv1"
  type: "BatchNorm"
  bottom: "conv1"
  top: "conv1"
```

…



```
  threshold: 11.316166, m: 15, kl: 0.002216, width: 0.005525
  threshold: 15.983469, m: 15, kl: 0.001968, width: 0.007804
  threshold: 38.791607, m: 15, kl: 0.002840, width: 0.018941
  threshold: 38.791607, m: 15, kl: 0.008903, width: 0.018941
  threshold: 16.530106, m: 15, kl: 0.008253, width: 0.008071
  threshold: 8.755128, m: 15, kl: 0.006438, width: 0.004275
  threshold: 9.159531, m: 15, kl: 0.007340, width: 0.004472
  threshold: 6.617352, m: 15, kl: 0.019335, width: 0.003231
  threshold: 2.465061, m: 15, kl: 0.008562, width: 0.001204
  threshold: 7.070976, m: 15, kl: 0.007678, width: 0.003453
  threshold: 8.781512, m: 15, kl: 0.007739, width: 0.004288
  threshold: 6.027140, m: 15, kl: 0.015872, width: 0.002943
  threshold: 1.502116, m: 15, kl: 0.002208, width: 0.000733
  threshold: 10.563061, m: 12, kl: 0.002809, width: 0.006348
  threshold: 23.744749, m: 15, kl: 0.002190, width: 0.011594
  threshold: 50.435867, m: 15, kl: 0.002647, width: 0.024627
  threshold: 50.435867, m: 15, kl: 0.008102, width: 0.024627
  threshold: 8.060768, m: 15, kl: 0.034156, width: 0.003936
  threshold: 18.360428, m: 15, kl: 0.050929, width: 0.008965
  threshold: 0.059225, m: 0, kl: 0.026566, width: 0.000463
Time: 3.122453s
chjtest@chjtest-ThinkPad-T480:~/compiler/Edge-Development-Toolchain/calibration_tool$
```

3. 查看结果

以下红色的部分为 tool 产生的文件。后面可以利用 bmnet_Resnet50_calibration_table.pb2 和 bmnet_Resnet50_int8.caffemodel 文件做 offline 精度测试，精度测试 pass 后可以用 bmbuilder 工具生成 bmodel 后部署到板子上.

```
chjtest@chjtest-ThinkPad-T480:~/compiler/Edge-Development-
Toolchain/calibration_tool/Resnet50$ tree -l
.
├── bmnet_Resnet50_calibration_table
├── bmnet_Resnet50_calibration_table.pb2
├── bmnet_Resnet50_calibration_table.prototxt
├── bmnet_Resnet50_int8.caffemodel
├── custom.caffemodel
├── deploy.prototxt
├── ILSVRC2012_val
│   ├── ILSVRC2012_val_00000077.JPEG
│   ├── ILSVRC2012_val_00000169.JPEG
```

5. 实用 Tips

可以参考如下实用的 Tips 以提升您的 INT8 转换精度。

1. 增加量化的图片数量并尽量增加您的图片对实际场景的覆盖范围

   By increasing the *iteration* parameter, this tool will use more data to calculate the quantization threshold. In our experiments, more data often gets more accurate. Note that the maximum iteration should be smaller than the size of LMDB dataset or the size of input text file when you use python data layer.

2. 加大参数 histogram_num 的值

   This means that the tool will search the threshold finer and may find a better one.

3. [经验]有的模型(非全部)在训练的时候打开 L2 regulation 参数会提升转换精度