IPU 精度问题 Debug SOP
SGS_IPU_SDK

V1.0

REVISION HISTORY

Revision No.	Description	Date
v1.0	• Initial release	05/16/2022

TABLE OF CONTENTS

1. 原始框架模型与 FLOAT.SIM 模型精度问题

1.1. 原始框架模型与 Float.sim 模型结果不一致

1.1.1 Caffe 模型与 Float.sim 模型结果不一致

1）请参考《用户手册》6.1 节对 float.sim dump 出每层数据。

2）请使用 SGS_IPU_SDK/DumpDebug/code/caffe_dump_data.py 脚本 dump caffe 的每层数据。使用该脚本请确保有 caffe 的 python 环境，修改 caffe_dump_data.py 第 3 行 caffe_root 为可用的 caffe python 环境路径。

使用示例：

```
python3 caffe_dump_data.py \
--model_file mobilenet_v1.prototxt \
--weight_file mobilenet_v1.caffemodel \
-i 000775.jpg \
--dump_bin True \
-n mobilenet_v1_preprocess.py
```

注意：模型前处理需要将数据转置到 NCHW 格式。

运行结束后会在当前目录产生 dumpData/caffe_NHWC_outtensor_dump.bin。

3）使用 SGS_IPU_SDK/DumpDebug/auto_dump_debug.sh 脚本运行，查看错误的 Layer。

使用示例：

```
./SGS_IPU_SDK/DumpDebug/auto_dump_debug.sh \
./SGS_IPU_SDK \
/path/to/float_sim_sigma_outtensor_dump.bin \
/path/to/caffe_NHWC_outtensor_dump.bin
```

1.1.2 ONNX 模型与 Float.sim 模型结果不一致

1）请参考《用户手册》6.1 节对 Foat.sim dump 出每层数据。

2）请使用 SGS_IPU_SDK/DumpDebug/code/onnx_dump_data.py 脚本 dump ONNX 的每层数据。使用该脚本时，--model_file 需要传入 ConvertTool.py 转换原始 ONNX 的中间文件，以_refine.onnx 结尾。

使用示例：

```
python3 onnx_dump_data.py \
--model_file mobilenet_v1_refine.onnx \
-i 000775.jpg \
--dump_bin True \
-n mobilenet_v1_preprocess.py
```

注意：模型前处理需要将数据转置到 NCHW 格式。

运行结束后会在当前目录产生 dumpData/onnx_NHWC_outtensor_dump.bin。

3）使用 SGS_IPU_SDK/DumpDebug/auto_dump_debug.sh 脚本运行，查看错误的 Operator。

使用示例：

```
./SGS_IPU_SDK/DumpDebug/auto_dump_debug.sh \
./SGS_IPU_SDK \
/path/to/float_sim_sigma_outtensor_dump.bin \
/path/to/onnx_NHWC_outtensor_dump.bin
```

1.1.3 TensorFlow 模型与 Float.sim 模型结果不一致

1）请参考《用户手册》6.1 节对 Float.sim dump 出每层数据。
2）所有 TensorFlow 模型需要转换至 tflite 格式。转换方法可参考：https://www.tensorflow.org/lite/convert
3）请使用 SGS_IPU_SDK/DumpDebug/code/tflite_dump_data.py 脚本 dump tflite 的每层数据。
使用示例：

```
python3 tflite_dump_data.py \
--model_file mobilenet_v1.tflite \
-i 000775.jpg \
--dump_bin True \
-n mobilenet_v1_preprocess.py
```

运行结束后会在当前目录产生 dumpData/tflite_outtensor_dump.bin。

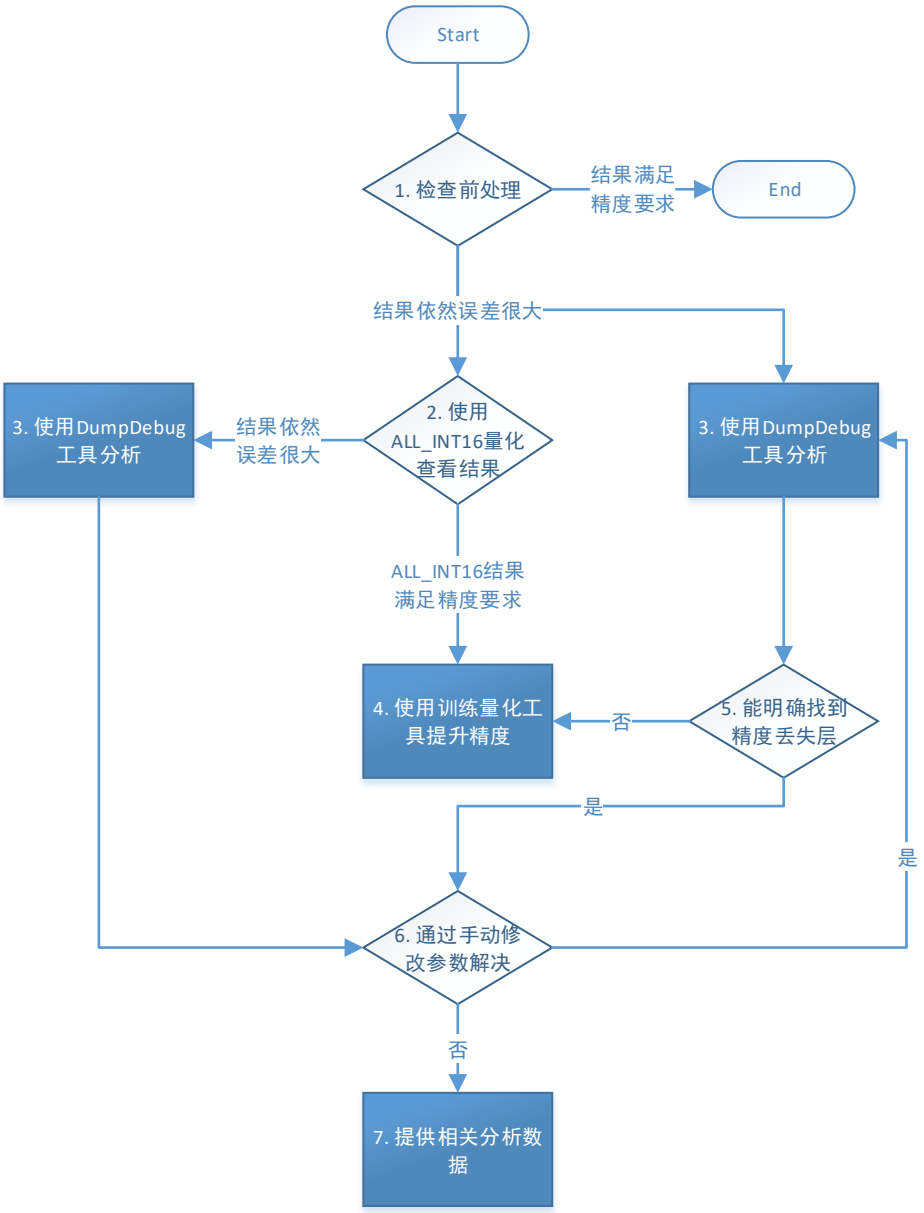
4）使用 SGS_IPU_SDK/DumpDebug/auto_dump_debug.sh 脚本运行，查看错误的 Operator。
使用示例：

```
./SGS_IPU_SDK/DumpDebug/auto_dump_debug.sh \
./SGS_IPU_SDK \
/path/to/float_sim_sigma_outtensor_dump.bin \
/path/to/ tflite_outtensor_dump.bin
```

2. FLOAT.SIM 模型和 FIXED.SIM 模型精度问题

2.1. Float.sim 和 Fixed.sim 误差很大

可以通过如下流程 Debug：



2.1.1 检查前处理

1）图片输入模型

前处理 Python 文件的 image_preprocess 函数定义中需要两个参数：

- 图片路径
- 归一化标记（norm=True）

其中归一化标记用于区分运行是否需要做图片归一化的动作。在运行 Float.sim 模型需要传入归一化后的图片，

image_preprocess 调用时会为 norm 传 True，此时需要将归一化动作放在 norm 为 True 下。运行 Fixed.sim 和 Offline 模型需要传入与模型输入大小的 UINT8 格式大小图片，此时不需要做归一化。因此图片输入模型前要检查是否正确使用了归一化标记，并做了正确的处理。同时，input_config.ini 和前处理 Python 文件中所配置的 mean 和 std 值要保持一致。

2）非图片输入模型
前处理 Python 文件需要实现和模型输入大小一致的 float32 类型的 numpy.ndarray 类型数据，无需关心归一化标记。请检查无论运行 Float.sim、Fixed.sim 还是 Offline 模型，前处理 image_preprocess 函数都返回一致的数据。

2.1.2 使用 ALL_INT16 量化查看结果

在 input_config.ini 中增加如下内容

```
[CONV_CONFIG]
input_format=ALL_INT16;
```

需要从原始模型重新转换生成 Float.sim 才能生效。

2.1.3 使用 DumpDebug 工具分析

参考《IPU 用户手册》6.1 和 6.2 节，对比 Float.sim 和 Fixed.sim 每层结果精度，输出每层 Tensor 的 MSE、COS、RMSE 信息作为参考。
由于 Fixed.sim 模型会有算子融合，部分 Tensor 无法 dump 出数据，可以通过开启 DebugConfig.txt 中 disableDomainFuseOps 选项，重新生成 Fixed.sim 模型后 dump 数据，再用 auto_dump_debug.sh 对比，可以得到更为详细的比对信息。

2.1.4 使用训练量化工具提升精度

有两个工具可以使用：
1）SGS_IPU_SDK/Scripts/examples/sim_optimizer.py
使用该脚本可以参考《IPU 用户手册》6.5 节。该脚本使用较为方便，不需要额外配置，但是最终会生成若干量化参数导入文件，需要依次手动导入后试验 Fixed.sim 精度是否能够满足。由于该工具还是以选择升级 INT16 来提升精度，因此可能会对模型在板端运行速度有影响。

2）SGS_IPU_SDK/Scripts/calibrator/torch_calibrator.py（推荐）
使用该脚本可以参考《IPU 用户手册》3.5 节。该脚本有两大量化等级，推荐使用精度更好的 Q2 量化等级，但是需要额外配置 GPU 环境，否则训练时间可能较长。训练结束后会自动选取最优量化参数生成 Fixed.sim 模型。

2.1.5 能明确找到精度丢失层？

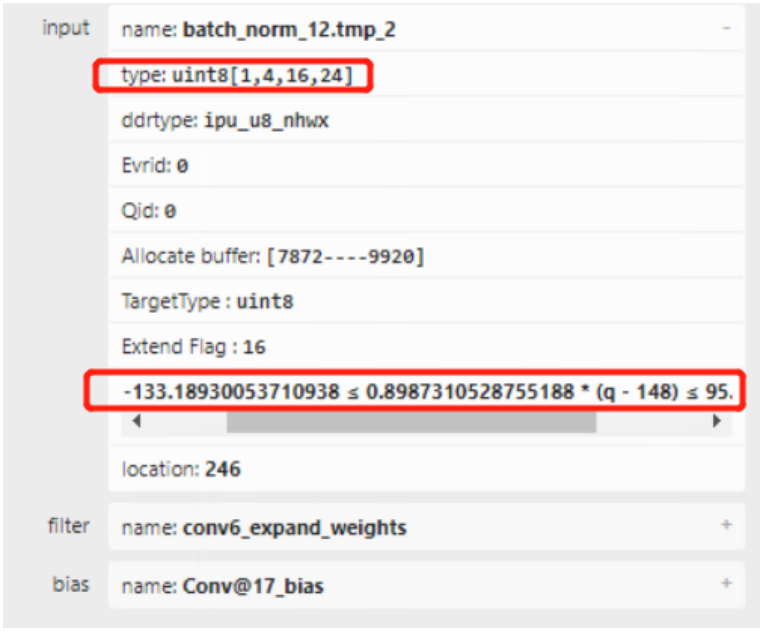
通过使用 DumpDebug 工具得到的 MES、COS、RMSE 信息中，从过往经验判断，当 COS < 0.99 或 RMSE > 0.1，说明该层精度可能已经无法满足。

```
39.batch_norm_6.tmp_2.xx.output0    OP: ADD  Type: float  MSE: 60.729672  COS: 0.997336  RMSE:  0.091513
43.relu_5.tmp_0.xx.output0  OP: RELU  Type: float  MSE: 7.867812  COS: 0.998164  RMSE:  0.077633
47.relu_6.tmp_0.xx.output0  OP: RELU  Type: float  MSE: 16.851153  COS: 0.994130  RMSE: 0.143941
51.elementwise_add_0.xx.output0 OP: ADD  Type: float  MSE: 148.815254  COS: 0.995671  RMSE: 0.106450
54.batch_norm_10.tmp_2.xx.output0    OP: ADD  Type: float  MSE: 204.189643  COS: 0.994502  RMSE: 0.125068
58.hard_swish_1.tmp_0.xx.output0    OP: MUL  Type: float  MSE: 78.561531  COS: 0.997094  RMSE:  0.117254
61.batch_norm_11.tmp_2.xx.output0    OP: ADD  Type: float  MSE: 27.176496  COS: 0.993667  RMSE:  0.134286
65.hard_swish_2.tmp_0.xx.output0    OP: MUL  Type: float  MSE: 17.986079  COS: 0.989773  RMSE:  0.188623
66.pool2d_1.tmp_0.xx.output0    OP: AVERAGE_POOL_2D  Type: float  MSE: 1.724975  COS: 0.998079  RMSE:  0.088585
69.relu_7.tmp_0.xx.output0  OP: RELU  Type: float  MSE: 0.051974  COS: 0.999975  RMSE:  0.007155
75.hard_sigmoid_1.tmp_0.xx.output0  OP: RELU  Type: float  MSE: 0.000007  COS: 0.999935  RMSE:  0.010415
```

如上图所示，在 DumpDebug 工具得到的信息中，如果这三个指标一直震荡，比如 COS 总在 0.99 ~ 0.98 震荡，RMSE 也总在 0.01 ~ 0.1 震荡，可以通过使用训练量化工具后再次对比。
如果能明确发现从某一层或几层开始，MES、COS、RMSE 三个指标都发生了很大改变，说明发生问题的点正是从这层开始的。如果这三个指标没有发生突变，比如 COS 慢慢从 0.99 往下掉，RMSE 慢慢从 0.1 往上升，可以找到第一个上述两个指标超过经验阈值的层作为后续重点查找目标。

2.1.6 通过手动修改参数解决？

手动修改参数的主要目的是通过修改指标不佳的 Tensor 用 INT16 量化，看能否提升到合适的精度。
当找到怀疑的 Tensor 后首先需要使用 SGS_IPU_SDK 中提供的 Netron 工具同时打开 Float.sim 和 Fixed.sim 模型，并定位到出问题的 Operator 位置。



Fixed.sim 模型可以点击对应 Op，在右边弹窗中点开输入输出 Tensor 查看到量化信息。如上图所示，当发现 Tensor 的 min / max 范围超过 20，并且该 Tensor 仍为 UINT8 量化，可以通过修改 input_config.ini 配置卷积升级到 INT16 量化。由于 input_config.ini 中只能配置卷积的输入 Tensor 为 INT16 量化，如果该 Op 不是卷积，可以将该 Op 前后几个卷积都配置成 INT16 后观察精度是否改善。

2.1.7 提供相关分析数据

如果能提供原始模型或者 Float.sim 和 Fixed.sim 模型，能最快复现问题。
如果无法提供模型，需要提供 dump 出的 Float.sim 和 Fixed.sim 的数据，并使用如下脚本 dump 出 Fixed.sim 模型的量化信息：

```
python3 SGS_IPU_SDK/Scripst/examples/save_quant_param.py \
-m mobilenet_v1_fixed.sim
```

会在 mobilenet_v1_fixed.sim 所在目录生成 mobilenet_v1_fixed.sim.json 文件。

需要提供：
1）每层精度对比结果
2）如果能定位出丢失精度的算子，提供丢失精度算子 float 的输入输出数据和 fixed 的输入输出数据
如果不能定位丢失精度的算子，把第一步的对比数据给到 SGS 相关同事帮忙判断具体需要提供的数据

3. 板端精度问题

3.1. 板端模型结果与 PC 仿真 Offline 模型精度检查要点

3.1.1 使用 dump_rawdata 方法验证

PC 上使用 SGS_IPU_SDK/Scripts/calibrator/simulator.py 运行 Offline 模型时，增加--dump_rawdata 选项，会在运行结束后在当前目录生成以图片名 + '.bin'为文件名的二进制数据文件。
例如：

```
python3 SGS_IPU_SDK/Scripts/calibrator/simulator.py \
-i 000775.jpg \
-m mobilenet_v1_fixed.sim_sgsimg.img \
-t Offline \
-n mobilenet_v1_preprocess.py \
--dump_rawdata
```

运行完成后会生成 000775.jpg.bin 的数据文件，该文件基于 mobilenet_v1_preprocess.py 的前处理和转换模型配置的 input_config.ini 中 input_formats 信息，将图片数据转换成 input_formats 格式，并做好了对齐处理，以符合 IPU 硬件特性。
板端使用时将 000775.jpg.bin 内所有数据全部拷贝到模型 Input buffer 内再推演模型，查看结果。
已有示例 demo dla_simulator 完成该功能，使用示例：

```
./prog_dla_simulator \
-i 000775.jpg.bin \
-m mobilenet_v1_fixed.sim_sgsimg.img \
-c Unknown \
--format DUMP_RAWDATA
```

3.1.2 检查 PC 使用的前处理与板端前处理区别

由于 PC 的前处理是 Python 写的，板端前处理是 c++写的，两者可能存在差别，总结常见的差别原因：

1） 有时候为了能提升模型精度，Python 前处理会做 crop 的操作，需检查板端 c++的前处理有同样的操作，并对比两者数据。
2） Python 前处理可能会将数据转换成 FLOAT32 再做图像大小 resize，板端使用 OpenCV 读图片做 resize 时，数据格式若均为 UINT8，会与 PC 端以 FLOAT32 做完 resize 后存在差异。
3） Python 中 numpy 的 round 函数与 c++的 round 函数由于底层实现不一致，结果不完全相同。
4） 如果 input_config.ini 中 input_formats 配置了 YUV_NV12、BGRA、RGBA 时，图像需要根据不同 IPU chip 或配置在宽、高方向上对齐，PC 上对齐的操作可以参考 SGS_IPU_SDK/Scripts/calibrator_custom/utils.py 中 convert_to_input_formats 函数的实现，确保在板端验证时有和上述函数中一样的操作（如果数据从前端 sensor 获取，数据已经按照要求做过对齐）。
5） 非图片输入的模型，如 input_config.ini 中 input_formats 配置为 RAWDATA_S16_NHWC，芯片型号为 1.x.x，Q_x.x 需要在板端将输入数据量化为 INT16 类型，并且数据最内维度向 8 对齐。详见《IPU 用户手册》特殊模型转换章节。

3.1.3 检查模型输出数据类型

假设有一个输出的模型，如下示例代码中 eOutFormat 为该模型输出数据类型：

```
MI_IPU_SubNet_InputOutputDesc_t desc;
MI_IPU_TensorVector_t OutputTensorVector;
/*
MI_IPU_CreateCHN
*/
MI_S32 s32Ret = MI_IPU_GetInOutTensorDesc(u32ChannelID, &desc);
if (s32Ret != MI_SUCCESS) {
    MI_IPU_DestroyCHN(u32ChannelID);
    MI_IPU_DestoryDevice();
    std::cerr << "GetInOutTensorDesc failed! Error Code: " << s32Ret << std::endl;
}
MI_IPU_ELEMENT_FORMAT eOutFormat = desc.astMI_OutputTensorDescs[0].eElmFormat;
```

模型输出数据的读取应以 eOutFormat 作为判断依据，如果 eOutFormat 为 MI_IPU_FORMAT_FP32，则可以使用如下代码读取模型 OutputTensor 的 Buffer：

```
MI_FLOAT* pstOutData =
(MI_FLOAT*)OutputTensorVector.astArrayTensors[0].ptTensorData[0];
```

如果 eOutFormat 为 MI_IPU_FORMAT_INT16，则可以使用如下代码读取模型 OutputTensor 的 Buffer：

```
MI_S16* pstOutData = (MI_S16*)OutputTensorVector.astArrayTensors[0].ptTensorData[0];
```

还需要注意输出数据的对齐要求：
1.x.x 版本所有数据类型的输出都会最内维度向 8 对齐，可能会有 garbage 数据。Garbage 数据请勿对比。
Q_x.x 版本 MI_IPU_FORMAT_FP32 类型数据与模型输出 shape 一致，MI_IPU_FORMAT_INT16 类型数据会最内维度向 8 对齐，可能会有 garbage 数据。Garbage 数据请勿对比。
S6.x.x 及以后版本所有数据类型的输出都与模型输出 shape 一致。