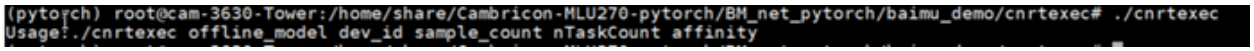
cnrtexec-Readme

附件：cnrtexec.tar.gz

1.概述

1.1 传参说明

Cnrtexec 为离线模型的性能测试程序，利用随机数来测试离线模型的 demo



传参有以下几个：

--offline_model：指定离线模型

--dev_id：声明使用的 mlu-device，只有一块为 0，默认为 0；

--sample_count：测试使用的 sample 数

--nTaskCount：测试启用的线程数

--affinity：内存通道亲和性设置，通常设置为 0 即可，详细说明可参见《寒武纪 CNRT 用户手册-v4.5.1.pdf》8.1 节

使用时可参考传参使用，裁减掉这些传参，使用固定值即可，简化调用。

1.2 前后处理添加

该 demo 需要自行添加模型推理的前后处理，需确认与在线推理的前后处理一致，否则会影响结果；

2.输入数据说明

2.1 输入数据预处理确认

1）该 demo 未包含前处理，需要在推理前添加 C++前处理，确认与在线推理使用的前处理保持一致；

2）如模型量化使用了 firstconv，则推理前不再需要做前处理，与在线一致；

2.2 输入数据确认

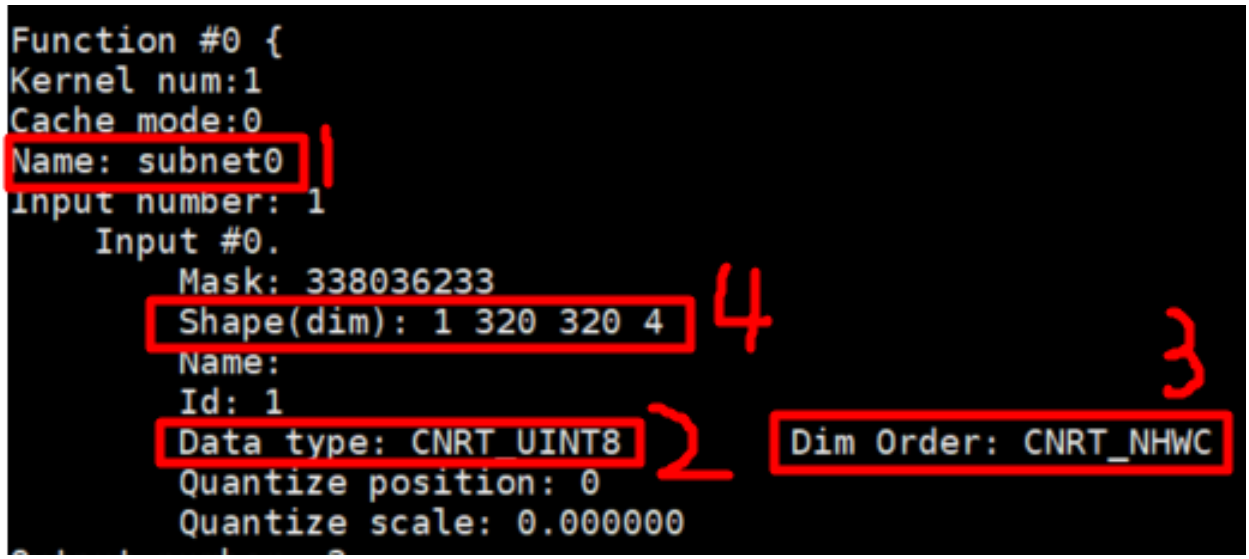


图 2-1

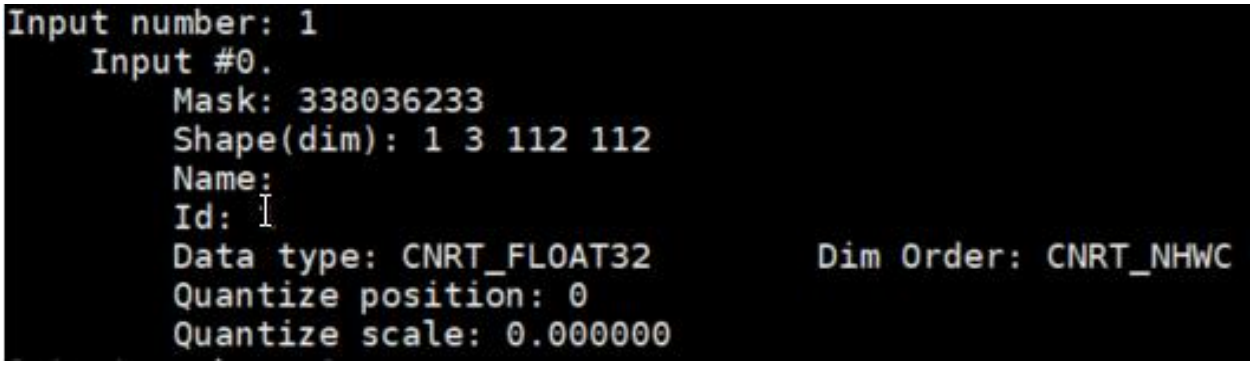


图 2-2

生成离线模型后，可通过查看模型描述文件.cambricon_twins，获取离线模型的输入信息；如图 2-1,2-2 所示：

图 2-1、图 2-2 举例了两个离线模型的输入信息，用以对比说明

其中图 2-1 模型，量化时使用 firstconv，图 2-2 模型，量化时未使用 firstconv，指定 iodtype 为 FLOAT32

1）Name：离线模型名称

通常默认为 subnet0，也有可能为其他，可参考 main.cpp 中 function_name 的调用方法；

2）Data_type：输入数据类型

如图 2-1 所示，该模型量化使用了 firstconv，则输入为 UINT8，

该输入类型在模型量化时，可进行指定，通常使用 FLOAT16，需结合该参数，确认传入正确的数据类型进行推理；图 2-2 中示例的是未使用 firstconv、iodtype 为 FLOAT32 的模型；

3）Dim Order：输入数据的维度顺序

离线模型推理可以接受 NHWC 的输入，也可以接受 NCHW 的输入，

实际输入的数据维度书序需要与模型描述的保持一致，注意做维度转换

维度转换可参考《寒武纪 CNRT 开发者手册-v4.5.1.pdf》-- 4.5.20 cnrtTransOrderAndCast

该函数提供对维度转换的操作以及数据类型的转换

4）Shape(dim)：输入数据各维度的规模

该 shape 顺序与 Dim order 对应，该例中，即 batch_size=1,h=320,w=320,c=4

由于该模型量化使用了 firstconv，因此 C=4（为了内部计算对齐），否则 C=3；

对于 C=4 的模型，推理前需要将输入的 C 扩充为 4

例如，如输入颜色格式为 RGB，则可使用 cv 接口，将颜色格式转为 RGBA 即可

也可手动扩充，在 RGB 后的第 4 维填充 0 即可。

2.3 输入数据拷入

该 demo 使用的是分配的随机数进行推理，在 Cnrtexec.cpp 中，MLUInfer::Detect 传入的 data 并没有被使用；

实际使用时，需将输入数据拷入，进行后续推理

```
CNRT_CHECK(cnrtMemcpy(inputMluPtrS[input_idx], data, inputSizeS[input_idx], CNRT_MEM_TRANS_DIR_HOST2DEV));
```

3.输出数据说明

3.1 输出数据后处理确认

该 demo 未包含后处理，如需要后处理，需在推理后添加 C++后处理，确认与在线推理使用的保持一致；

3.2 输出数据确认

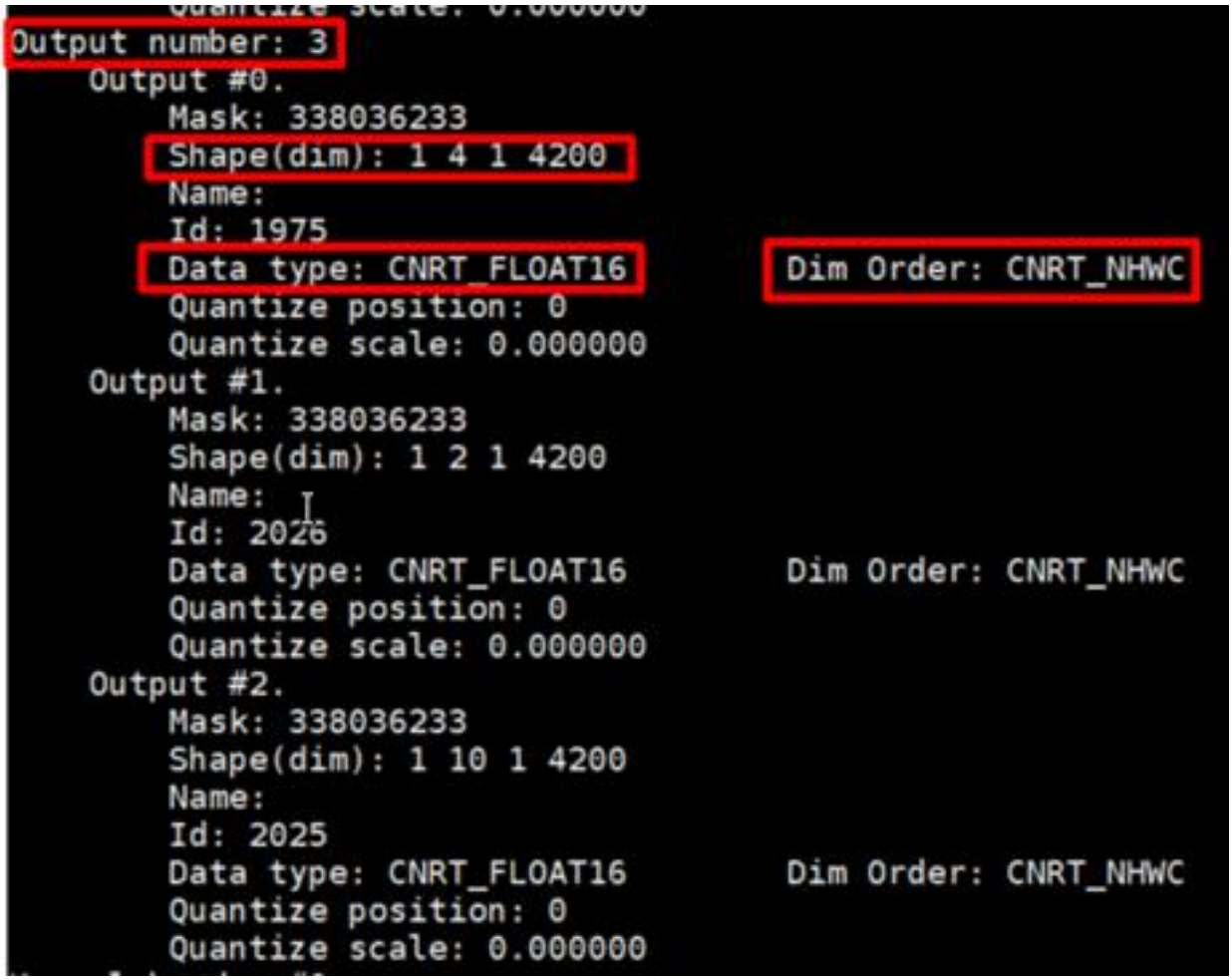


图 3-1

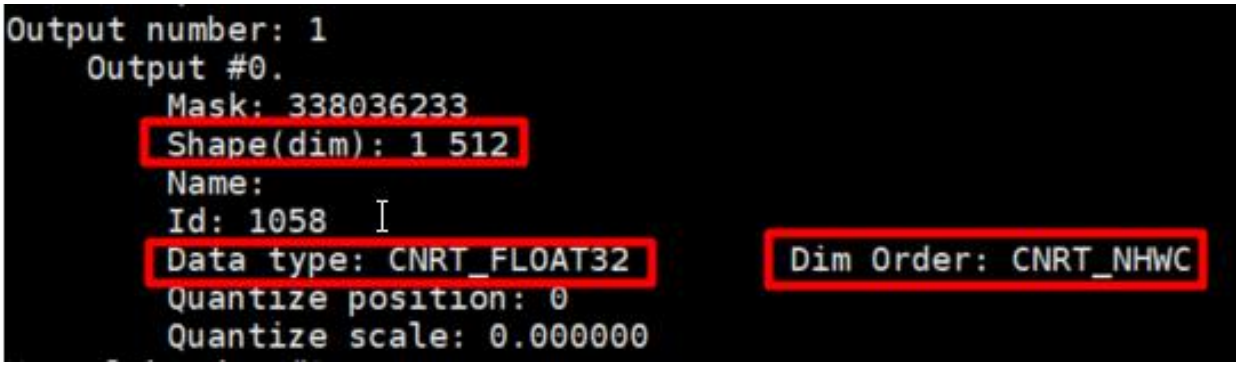


图 3-2

图 3-1 和图 3-2 给出了两个离线模型的输出示例，用以对比说明

图 3-1 中，有 3 个输出，每个输出有四个维度，dimNum=4，维度顺序为 NHWC，量化时指定输出数据类型为 FLOAT16

图 3-2 中，有 1 个输出，输出有 2 个维度，dimNum=2，维度顺序为 NHWC，量化时指定输出数据类型为 FLOAT32

3.3 输出数据转换

对于 FLOAT32 类型的输出结果，可以直接使用。

但对于 FLOAT16 的输出，需要将数据类型转换为 FLOAT32 才可以在 CPU 上使用

因此 cnrtexec.cpp，MLUInfer::Detect 中，有如下代码：

```
#if 1

        float* output_ptr=res;
        for(int output_idx=0;output_idx<outputNum;output_idx++)
        {
                int dim_order[4] = {0, 3, 1, 2};

        CNRT_CHECK(cnrtGetOutputDataShape(&dimValues,&dimNum,output_idx,functi
on));
                if(dimNum==4)
                {

        CNRT_CHECK(cnrtMemcpy(outputCpuPtrS[output_idx],outputMluPtrS[output_i
dx],outputSizeS[output_idx],CNRT_MEM_TRANS_DIR_DEV2HOST));



        CNRT_CHECK(cnrtTransOrderAndCast(reinterpret_cast<void*>(outputCpuPtrS
[output_idx]), outputTypeS[output_idx],

         reinterpret_cast<void*>(outputCpuNchwPtrS[output_idx]), CNRT_FLOAT32,

         nullptr, dimNum, dimValues, dim_order));

        memcpy(output_ptr,outputCpuNchwPtrS[output_idx],output_count[output_id
x]*sizeof(float));
                }
                else
                {
```

```
        CNRT_CHECK(cnrtMemcpy(output_ptr,outputMluPtrS[output_idx],outputSizeS
[output_idx],CNRT_MEM_TRANS_DIR_DEV2HOST));
                }
                output_ptr+=output_count[output_idx];
                 free(dimValues);
        }
#endif
```

上述代码中使用 cnrtGetOutputDataShape，获取到模型输出数据的维度顺序-dimValues，以及维度规模-dimNum；

使用 cnrtTransOrderAndCast 完成对输出数据的类型及维度顺序转换；

传参 dim_order 为维度顺序转换的目的顺序

举例说明：

输入的维度顺序为 NHWC，dim_order 传参[0,3,1,2]，则调用 cnrtTransOrderAndCast 转换后的维度顺序变为 NCHW，即将 3 位置的 C 摆放到 1 位置，1,2 摆放到 2,3 位置

实际使用时，可结合对输出的维度需求，进行设置。

cnrtTransOrderAndCast 的详细使用说明可参考《寒武纪 CNRT 开发者手册-v4.5.1.pdf》-- 4.5.20 cnrtTransOrderAndCast

需要注意的是，这里给出的 demo 是判断 dimNum 为 4 后，才进行的类型转换

实际使用时，需结合模型的实际 dimNum 及数据类型对代码进行修改