云天励飞 TyAssist 工具使用说明

版本：v0.3.0

更新记录

版本	修改日期	修改说明
v0.1.0	2022.1.13	初始版本，支持模型转换、推理和芯片逐层精度分析
v0.2.0	2022.6.24	增加精度和性能 benchmark 测试，推理结果演示章节
v0.3.0	2022.8.11	根据 v1.0.0 版本修改，配置文件改为 yaml 文件

目录

概述

TyAssist 是云天励飞天元系列人工智能芯片的开发辅助工具，是 DEngine 软件套件产品之一。基于 tytvm、tyhcp 等其他组件接口之上封装，主要使用 python 开发，利用配置文件实现快速模型转换、推理、精度分析、结果展示、精度和性能测试等芯片评估功能。本文档主要介绍如何使用 TyAssist 工具进行芯片快速评估。

开发包介绍

TyAssist 开发包命名规则为 TyAssist_va.b.c（abc 分别代表 1 个数字），目前适用于 dp2000 芯片（含 nnp3xx 系列 npu），内部目录结构和说明如下：

tyassist/
- ├── docs （使用文档）
- ├── examples （示例模型）
- ├── utils（公用模块）
- ├── datasets（常见数据集）
- ├── base（模型和自定义预处理基类）
- ├── src （工具主体代码）
- ├── version （版本说明）
- └── tyassist.py (工具入口)

示例运行

TyAssist 运行依赖于 TyTVM 和 TyHCP，需要将 TyAssist 和 TyTVM、TyHCP 开发包解压在同级目录下。

1. 模型转换

模型转换功能是将原始框架模型转换为可在芯片上运行的模型。依赖 TyTVM 环境，从 TyTVM 开发包进入 docker，然后切换到/DEngine/tyassist/examples 目录下，执行命令：

```
# cd tytvm
# ./docker_enter.sh
# cd /DEngine/tyassist/examples/caffe_squeezenet_v1_1
# sh build.sh
```

运行成功打印：**SUCCESS**。编译 log 保存在 logs 目录中，若有问题可提交给云天励飞芯片支持人员分析。

打印的最后部分输出了浮点和定点结果的 shape 和 dtype，并计算了指定输入数据每个结果 tensor 的量化相似度（cpu 上浮点与定点仿真结果之间），可供参考。

```
[runoncpu] float(tvm) output tensor [0] shape:(1, 1000, 1, 1) dtype:float32
[runoncpu] fixed(tvm) output tensor [0] shape:(1, 1000, 1, 1) dtype:float32
[runoncpu] fixed(iss) output tensor [0] shape:(1, 1000, 1, 1) dtype:float32
[runoncpu] float(tvm) vs fixed(tvm) output tensor [0] similarity: 0.988882
[runoncpu] float(tvm) vs fixed(iss) output tensor [0] similarity: 0.988880
```

2. 模型推理

模型推理功能是将编译出的模型在芯片或仿真平台运行，查看运行结果。依赖 tyhcp 环境。以 tyhcp docker+仿真平台运行环境为例，从 tyhcp 开发包进入 docker，启动 iss_server（如果是芯片平台不需要启动，而是将/DEngine/tyhcp/net.cfg 中的 ip 地址修改为芯片平台地址），然后切换到 /DEngine/TyAssist/examples/dp2000 目录下，执行命令：

```
# cd tyhcp
# ./docker_enter.sh
# sh iss_server.sh
# cd /DEngine/tyassist/examples/caffe_squeezenet_v1_1
# sh compare.sh
```

运行成功打印：**SUCCESS**。编译 log 保存在 logs 目录中，若有问题可提交给云天励飞芯片支持人员分析。

打印的最后部分输出了执行时间，输出结果的 shape 和 dtype，并计算了指定输入数据每个结果 tensor 的执行相似度（芯片结果与定点、浮点仿真结果之间），可供参考。

```
outputs[0], shape=(1, 1000, 1, 1), dtype=float32
[runonchip] fixed(chip) vs fixed(tvm) output tensor: [0] similarity=0.999993
[runonchip] fixed(chip) vs float(tvm) output tensor: [0] similarity=0.988880
```

注意：TyAssist 示例中使用的模型是前一步模型转换编译出的模型，在./dp2000 目录下，与 tyhcp 示例中使用的模型并不是同一个。

3. 精度分析

精度分析功能是将模型各层结果进行相似度比对，分析出导致精度下降的层。精度下降主要存在于量化和芯片两个环节。

3.1 量化精度分析

量化环节的精度损失主要由于模型数据和参数由浮点转为定点导致。转换时在 ini 配置文件中将 debug_level 设置为 1（默认 1），将开启量化逐层相似度分析。

```
# 量化调试等级，默认为 1
# 0 - 输出整体余弦相似度，
# 1 - 输出整体余弦相似度、输出逐层余弦相似度、逐层保存浮点模型和定点模型的数据、逐层保存浮点定点的对比 PNG 图
# 2 - 输出整体余弦相似度、输出逐层余弦相似度、逐层保存浮点定点的对比 PNG 图
```

```
# 3 - 输出整体余弦相似度、输出逐层余弦相似度
#-1 - 不输出 debug 信息
debug_level: 1,
```

转换完成后，在 log 中搜索 layer cos_similarity 可看到如下打印：

```
>>>>>[layer cos_similarity] the result is:
quantized_layer_name
shape                          cos_similarity                  float_layer_name
---------------
-----                          --------------                  ----------------
fused_nn_conv2d_add_multiply_round_right_shift_clip_0
(1, 3, 227, 227)               0.9999831737362775              fused_nn_conv2d_add_0
fused_nn_conv2d_add_multiply_round_right_shift_clip_nn_relu_clip_13_0
(1, 64, 113, 113)              0.9983502678829336
fused_nn_conv2d_add_nn_relu_13_0
fused_nn_max_pool2d_clip_2_0
(1, 64, 56, 56)                0.9991371423567879              fused_nn_max_pool2d_0
fused_nn_conv2d_add_multiply_round_right_shift_clip_nn_relu_clip_12_0
(1, 16, 56, 56)                0.9993206093256708
fused_nn_conv2d_add_nn_relu_12_0
fused_nn_conv2d_add_multiply_round_right_shift_clip_nn_relu_clip_10_0
(1, 64, 56, 56)                0.987936592977185
......
```

3.2 芯片精度分析

芯片环节的精度损失主要由于芯片和 tvm 仿真平台差异导致，理论上这个损失会非常小。转换时在 yaml 配置文件中将 enable_dump 设置为 1（默认 0），将开启芯片逐层相似度分析。

```
#  是否进行逐层结果分析，默认为 0
enable_dump = 1
```

参照模型转换示例，将 config.yaml 中的 enable_dump 修改为 1 保存，在 tytvm docker 中执行：

```
# sh build.sh
```

转换完成后生成的模型目录下 result 目录下将生成 host_iss_fused_weight.pickle 文件，记录 tvm 仿真的逐层结果。参照模型推理示例，先启动 dump 服务，在 tyhcp docker 中执行：

```
# sh /DEngine/tyhcp/dump_server.sh
# sh compare.sh
```

运行完成后会有打印：

```
['fused_multiply_round_clip_cast', <class 'numpy.int8'>, [1, 3, 227, 227], 1.0]
+----+---------------------------+---------+------------------+--------------------+
| Id |          OpName           |  Dtype  |      Shape       |     CosineDist     |
```

```
+----+--------------------------+--------+------------------+--------------------+
| 0  |    fire2/relu_squeeze1x1     |   int8   |   (1, 16, 56, 56)   | 1.0000000000000016 |
| 1  |     fire2/relu_expand1x1      |   int8   |   (1, 64, 56, 56)   | 0.9999999999999984 |
| 2  |     fire2/relu_expand3x3      |   int8   |   (1, 64, 56, 56)   | 0.9999999999999999 |
| 3  |    fire3/relu_squeeze1x1      |   int8   |   (1, 16, 56, 56)   | 1.0000000000000002 |
| 4  | __TyQuant_22_nn_max_pool2d |    int8   |   (1, 64, 28, 28)   | 1.0000000000000002 |
| 5  | __TyQuant_26_nn_max_pool2d |    int8   |   (1, 64, 28, 28)   | 0.9999999999999993 |
| 6  |    fire4/relu_squeeze1x1      |   int8   |   (1, 32, 28, 28)   | 0.9999999999999998 |
| 7  |     fire4/relu_expand1x1      |   int8   |   (1, 128, 28, 28) | 0.9999999999999984 |
| 8  |     fire4/relu_expand3x3      |   int8   |   (1, 128, 28, 28) | 0.9999999999999996 |
| 9  |    fire5/relu_squeeze1x1      |   int8   |   (1, 32, 28, 28)   | 0.9999999999999998 |
| 10 | __TyQuant_44_nn_max_pool2d |    int8   |   (1, 128, 14, 14) | 1.0000000000000002 |
| 11 | __TyQuant_48_nn_max_pool2d |    int8   |   (1, 128, 14, 14) | 1.0000000000000002 |
| 12 |    fire6/relu_squeeze1x1      |   int8   |   (1, 48, 14, 14)   | 1.0000000000000013 |
| 13 |     fire6/relu_expand1x1      |   int8   |   (1, 192, 14, 14) |        1.0         |
| 14 |     fire6/relu_expand3x3      |   int8   |   (1, 192, 14, 14) |        1.0         |
| 15 |    fire7/relu_squeeze1x1      |   int8   |   (1, 48, 14, 14)   | 1.0000000000000016 |
| 16 |     fire7/relu_expand1x1      |   int8   |   (1, 192, 14, 14) | 1.0000000000000002 |
| 17 |     fire7/relu_expand3x3      |   int8   |   (1, 192, 14, 14) |        1.0         |
| 18 |    fire8/relu_squeeze1x1      |   int8   |   (1, 64, 14, 14)   | 0.9999999999999999 |
| 19 |     fire8/relu_expand1x1      |   int8   |   (1, 256, 14, 14) |        1.0         |
| 20 |     fire8/relu_expand3x3      |   int8   |   (1, 256, 14, 14) | 1.0000000000000002 |
| 21 |    fire9/relu_squeeze1x1      |   int8   |   (1, 64, 14, 14)   |        1.0         |
| 22 |     fire9/relu_expand1x1      |   int8   |   (1, 256, 14, 14) | 0.9999999999999998 |
| 23 |     fire9/relu_expand3x3      |   int8   |   (1, 256, 14, 14) |        1.0         |
| 24 |          relu_conv10          |   int8   | (1, 1000, 14, 14) | 0.999999999999999  |
| 25 |              pool10           |   int8   |   (1, 1000, 1, 1)   | 1.0000000000000002 |
| 26 |      fused_multiply_14        | float16 |   (1, 1000, 1, 1)   | 1.0000000000000002 |
| 27 |      fused_nn_softmax_2       | float16 |   (1, 1000, 1, 1)   |        1.0         |
| 28 |             prob              | float32 |   (1, 1000, 1, 1)   | 0.9999999999999998 |
+----+--------------------------+--------+------------------+--------------------+
```

同时在芯片模型的生成目录下 result/chip_dump_out 目录下生成 similarity.csv 文件，打开可以看到结果如下：

name	dtype	shape	similarity
fused_cast_5	<class 'numpy.float16'>	[1, 3, 227, 227]	1
fused_multiply_round_clip_cast	<class 'numpy.int8'>	[1, 3, 227, 227]	0.999999999999998
fire2/relu_squeeze1x1	<class 'numpy.int8'>	[1, 16, 56, 56]	1
fire2/relu_expand1x1	<class 'numpy.int8'>	[1, 64, 56, 56]	1
fire2/relu_expand3x3	<class 'numpy.int8'>	[1, 64, 56, 56]	1
fire3/relu_squeeze1x1	<class 'numpy.int8'>	[1, 16, 56, 56]	0.999999999999999
__TyQuant_22_nn_max_pool2d	<class 'numpy.int8'>	[1, 64, 28, 28]	1
__TyQuant_26_nn_max_pool2d	<class 'numpy.int8'>	[1, 64, 28, 28]	0.9999999999999

			98
fire4/relu_squeeze1x1	<class 'numpy.int8'>	[1, 32, 28, 28]	1
fire4/relu_expand1x1	<class 'numpy.int8'>	[1, 128, 28, 28]	1
fire4/relu_expand3x3	<class 'numpy.int8'>	[1, 128, 28, 28]	1
fire5/relu_squeeze1x1	<class 'numpy.int8'>	[1, 32, 28, 28]	1
__TyQuant_44_nn_max_pool2d	<class 'numpy.int8'>	[1, 128, 14, 14]	1
__TyQuant_48_nn_max_pool2d	<class 'numpy.int8'>	[1, 128, 14, 14]	1
fire6/relu_squeeze1x1	<class 'numpy.int8'>	[1, 48, 14, 14]	1
fire6/relu_expand1x1	<class 'numpy.int8'>	[1, 192, 14, 14]	1
fire6/relu_expand3x3	<class 'numpy.int8'>	[1, 192, 14, 14]	1
fire7/relu_squeeze1x1	<class 'numpy.int8'>	[1, 48, 14, 14]	1
fire7/relu_expand1x1	<class 'numpy.int8'>	[1, 192, 14, 14]	1
fire7/relu_expand3x3	<class 'numpy.int8'>	[1, 192, 14, 14]	1
fire8/relu_squeeze1x1	<class 'numpy.int8'>	[1, 64, 14, 14]	1
fire8/relu_expand1x1	<class 'numpy.int8'>	[1, 256, 14, 14]	1
fire8/relu_expand3x3	<class 'numpy.int8'>	[1, 256, 14, 14]	1
fire9/relu_squeeze1x1	<class 'numpy.int8'>	[1, 64, 14, 14]	1
fire9/relu_expand1x1	<class 'numpy.int8'>	[1, 256, 14, 14]	0.999999999999999
fire9/relu_expand3x3	<class 'numpy.int8'>	[1, 256, 14, 14]	1
relu_conv10	<class 'numpy.int8'>	[1, 1000, 14, 14]	1
pool10	<class 'numpy.int8'>	[1, 1000, 1, 1]	1
fused_multiply_15	<class 'numpy.float16'>	[1, 1000, 1, 1]	1
fused_nn_softmax_2	<class 'numpy.float16'>	[1, 1000, 1, 1]	1
prob	<class 'numpy.float32'>	[1, 1000, 1, 1]	0.999999999999998

4. 自定义前处理

天元系列芯片有专门的硬件支持单通道和三通道模型数据的 resize 和归一化加速处理，可在 yaml 配置文件中配置 norm 和 resize 参数实现。如果不符合上述特征，就需要自行对数据进行前处理后输入模型，才能进行量化和推理等操作。

这种情况下使用 TyAssist 可以自行继承自定义基类实现，并将函数所在 python 文件名和类名添加至 yaml 配置文件，例如：

```
# 用户自定义量化函数，若不配置此项，则使用默认前处理，多输入或非 1,3 通道数据必须配置
# 自定义预处理模块必须与 yml 配置文件同级目录
custom_preprocess_module: "custom_preprocess",
custom_preprocess_cls: "CustomTensorFlowImgD",
```

参照模型转换示例，在 tytvm docker 中，进入 DEngine/tyassist/examples/tensorflow_imgD 执行:

```
# sh build.sh
```

转换完成后，参照模型推理示例，在 tyhcp docker 中执行：

```
# sh compare.sh
```

可以得到结果：（该示例在仿真平台运行时间较长）

```
[runonchip] predict cost: 10862881.616ms
[runonchip] predict result: data_out len=1
data_out[0], shape=(1, 544, 960, 4), dtype=float32
[runonchip] vs fixed output tensor [0] similarity=0.9991954362470804
```

5. 模型评估

TyAssist 工具提供精度和性能评估功能，方便供用户进行快速模型评估测试。

5.1 精度评估

精度评估是指将模型推理结果加上适当的后处理后与测试数据集的标注结果进行比较，得到针对此测试数据集的精度结果结果，据此可以与原模型精度结果对比，评估芯片部署带来的精度损失。精度结果对于检测模型来说一般以 mAP 表示，对于分类模型来说一般以 top1/top5 表示。用户可根据接口自行定义模型的后处理和数据库评估代码，由 TyAssist 工具调用完成测试。精度 benchmark 在仿真环境和芯片环境均可运行。

以 caffe_squeezenet 为例，在配置文件中配置 benchmark 字段，指定数据集路径、测试数量、数据集模块名和处理类名。其中处理类名定义在模块名的 python 文件中，为精度测试提供数据库处理方法，接口符合 DatasetBase 类的定义，DatasetBase 类定义在 datasets/dataset_base.py 中。同时需要在 common 字段指定 python 处理模块，为测试提供模型后处理。

```
#  模型处理模块名(test/demo 时必须设置)
model_impl_module = model_impl

#  模型处理模块类名(test/demo 时必须设置)
model_impl_cls = SqueezeNetV1_1

...

#  指定 test 的数据集路径(必须设置)
datasets_path = "/DEngine/tymodelzoo/data/ILSVRC2012/ILSVRC2012_img_val",

# test 精度测试样例数，实际测试数为数据集内数据数与本数的较小者，不设置或者配置
0 为数据集内所有数据
test_num = 0

#  数据集模块名(必须设置)
dataset_module = datasets
#  数据集处理类名(必须设置)
dataset_cls = Dataset
```

使用 benchmark.sh 脚本执行配置文件：

```
# sh test.sh
final result:
(0.7, 'precision: top1/top5: 0.700000/0.900000, testnums: 10')
```

5.2 性能评估

性能评估是指将测试数据在各种测试模式下送给模型进行推理，以测试模型在平台上的执行性能。测试模式参考 MLPerf。目前仅支持多线程一般推理测试。

目前仅支持在芯片侧执行，首先在主控侧编译环境中编译 perf_test 工程：

```
# cd /DEngine/tyassist/benchmark/perf_test
# sh build.sh
```

成功后进入芯片侧，配置芯片侧环境后执行

```
# cd /DEngine/tyassist/benchmark/perf_test
# sh run.sh
[INFO]    config: dclConfigPath=/mnt/DEngine/tyhcp/config/sdk.cfg,
dclDataDirPath=/mnt/DEngine/tyexamples/data/bin/COCO_val2014_000000000139.jpg.416x4
16.rgb.plane.bin,
dclModelDirPath=/mnt/DEngine/tyexamples/models/dp2000/caffe_squeezenet_v1.1/net_combi
ne.bin, aippFlag=1, loop=1, thdnum=1

[INFO]    dcl mem init success
[NOTICE] DCL(335) 13669-05:05:26:409 [dcl_log.cc:104]: [dcl::log level]    default level: 1
mID/level: 9/2 7/2 6/2 5/2 4/2 3/2 8/2 0/2 1/2 2/2
libremote_dump_module.so version: DeSDK_v0.3.3-7-gc587fc-dirty, 2022-06-16
[05:05:34]
/data/fandongsheng/git_lib/nn_data_dump/data_dump_client/data_dump_wrapper.cc:304:
Warning: Error connecting 192.168.33.205:9040
[NOTICE] DCL(335) 8025711-05:05:34:420 [dcl_log.cc:118]: [dcl::log level]    default level: 1
mID/level: 9/1 7/2 6/1 5/2 4/2 3/2 8/0 0/2 1/2 2/2
[INFO]    dcl init success
[INFO]    load model
/mnt/DEngine/tyexamples/models/dp2000/caffe_squeezenet_v1.1/net_combine.bin success
[INFO]    load model success,
/mnt/DEngine/tyexamples/models/dp2000/caffe_squeezenet_v1.1/net_combine.bin
[INFO]    create model description success
[INFO]    init model success,
/mnt/DEngine/tyexamples/models/dp2000/caffe_squeezenet_v1.1/net_combine.bin
[INFO]    model output num=1
[INFO]    output[0] dim=4, format=-1, datatype=0
[INFO]    shape[0]=1
[INFO]    shape[1]=1000
[INFO]    shape[2]=1
[INFO]    shape[3]=1
[INFO]    create input success
[INFO]    add output index 0 size 4000
[INFO]    create output success
```

```
[INFO]  perf thread 0, test 0, cost 2.096899ms
[INFO]  thread 0 exit!
[INFO]  perf thread 0 test=1, average cost 2.096899ms
[INFO]  perf total test=1, average cost 2.096899ms
[INFO]  perf test success, all cost 8460.939031ms
[INFO]  unload model success, modelId is 0
[INFO]  destroy model description success
[INFO]  end to finalize dcl
[INFO]  memory de-init done
```

可通过修改 run.sh 中参数修改模型、数据、执行次数、线程数等参数。

6. 推理结果演示

需要在 model 字段指定 python 处理模块，提供模型演示方法。

```
#  模型处理模块名(test/demo 时必须设置)
model_impl_module = model_impl
#  模型处理模块类名(test/demo 时必须设置)
model_impl_cls = SqueezeNetV1_1
```

使用 demo.sh 脚本执行配置文件：

```
# sh demo.sh
==========================================
1/10, data ILSVRC2012_val_00000012.JPEG
/DEngine/tyexamples/data/datasets/ILSVRC2012/ILSVRC2012_img_val/ILSVRC2012_val_00
000012.JPEG
predict result: data_out len=1
predict id = 286, prob = [[0.9970703]]

==========================================
2/10, data ILSVRC2012_val_00000004.JPEG
/DEngine/tyexamples/data/datasets/ILSVRC2012/ILSVRC2012_img_val/ILSVRC2012_val_00
000004.JPEG
predict result: data_out len=1
predict id = 809, prob = [[0.60839844]]

==========================================
3/10, data ILSVRC2012_val_00000002.JPEG
/DEngine/tyexamples/data/datasets/ILSVRC2012/ILSVRC2012_img_val/ILSVRC2012_val_00
000002.JPEG
predict result: data_out len=1
predict id = 795, prob = [[0.9741211]]

==========================================
4/10, data ILSVRC2012_val_00000013.JPEG
/DEngine/tyexamples/data/datasets/ILSVRC2012/ILSVRC2012_img_val/ILSVRC2012_val_00
```

```
000013.JPEG
predict result: data_out len=1
predict id = 370, prob = [[0.9453125]]

==========================================
5/10, data ILSVRC2012_val_00000007.JPEG
/DEngine/tyexamples/data/datasets/ILSVRC2012/ILSVRC2012_img_val/ILSVRC2012_val_00
000007.JPEG
predict result: data_out len=1
predict id = 334, prob = [[0.68652344]]

==========================================
6/10, data ILSVRC2012_val_00000016.JPEG
/DEngine/tyexamples/data/datasets/ILSVRC2012/ILSVRC2012_img_val/ILSVRC2012_val_00
000016.JPEG
predict result: data_out len=1
predict id = 147, prob = [[0.9892578]]

==========================================
7/10, data ILSVRC2012_val_00000019.JPEG
/DEngine/tyexamples/data/datasets/ILSVRC2012/ILSVRC2012_img_val/ILSVRC2012_val_00
000019.JPEG
predict result: data_out len=1
predict id = 478, prob = [[0.82373047]]

==========================================
8/10, data ILSVRC2012_val_00000011.JPEG
/DEngine/tyexamples/data/datasets/ILSVRC2012/ILSVRC2012_img_val/ILSVRC2012_val_00
000011.JPEG
predict result: data_out len=1
predict id = 109, prob = [[0.9892578]]

==========================================
9/10, data ILSVRC2012_val_00000017.JPEG
/DEngine/tyexamples/data/datasets/ILSVRC2012/ILSVRC2012_img_val/ILSVRC2012_val_00
000017.JPEG
predict result: data_out len=1
predict id = 552, prob = [[0.7294922]]

==========================================
10/10, data ILSVRC2012_val_00000014.JPEG
/DEngine/tyexamples/data/datasets/ILSVRC2012/ILSVRC2012_img_val/ILSVRC2012_val_00
000014.JPEG
predict result: data_out len=1
predict id = 654, prob = [[0.4658203]]
```