**云天励飞TyAssist工具使用说明**

版本：v0.2.0

**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 | v2版，支持4种基础功能量化编译模型、量化精度比较验证、模型demo、模型评估 |

目录

[更新记录
1](#_Toc724058426)

[概述
3](#_Toc17635125)

[开发包介绍
3](#_Toc1200095164)

[示例运行
3](#_Toc559366940)

[1. 模型转换
4](#_Toc170573596)

[2. 模型推理
4](#_Toc1582804920)

[3. 精度分析
5](#_Toc1627734478)

[3.1 量化精度分析
5](#_Toc659237411)

[3.2 芯片精度分析
6](#_Toc1740059053)

[4. 自定义前处理
9](#_Toc1089564627)

[5. benchmark测试 10](#_Toc317143190)

[5.1精度benchmark 10](#_Toc1988467217)

[5.2性能benchmark 11](#_Toc967212176)

[6. 推理结果演示 13](#_Toc1127959495)

**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 |

1. **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。**

1. **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] | 0.999999999999998 |
| 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 |

1. **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 |

1. **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.416x416.rgb.plane.bin, dclModelDirPath=/mnt/DEngine/tyexamples/models/dp2000/caffe\_squeezenet\_v1.1/net\_combine.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。

1. **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\_00000012.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\_00000004.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\_00000002.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\_00000013.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\_00000007.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\_00000016.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\_00000019.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\_00000011.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\_00000017.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\_00000014.JPEG  predict result: data\_out len=1  predict id = 654, prob = [[0.4658203]] |