# 云天励飞DESDK Benchmark工具使用手册

版本：v1.0.0

**更新记录**

| 版本 | 修改日期 | 修改说明 |
| --- | --- | --- |
| V1.0.0 | 2020.12.9 | 初始版本 |

**目录**



# 1 概述

benchmark工具用于对模型在芯片上的精度和推理性能进行测试和统计，使用户对DeepEye芯片当前配置下的推理能力有定量的了解，帮助用户根据需要选择合理的配置参数。

本文档主要介绍如何使用benchmark工具对模型在DeepEye芯片上的推理的算法精度和性能进行测试和统计。

# 2 基本概念

## 2.1 算法精度和精度损失

算法精度是指对模型推理的结果增加后处理后的结果与测试集已标注的结果对比得到的准确度。

对于分类模型来说后处理的结果一般是由分类概率从大到小排列的分类标签及置信度，验证时一般使用Top1和Top5，表示实际分类结果在测试集标注结果中命中第1个和前5个的概率。

对于检测模型来说一般是检测框的位置和置信度，验证时一般使用mAP(mean Average Precision)，表示检测框与实际框的交并比IoU(Intersection over Union)的平均值。

将原始模型在框架下的运行结果和芯片上运行结果分别做算法精度，得到Top或mAP精度。两者相减可以得到芯片运行的精度损失，作为评价芯片运行精度的指标。

## 2.2 硬件性能和端到端性能

DeepEye1000芯片为协处理器，硬件性能是指在芯片侧直接运行模型的性能，端到端性能指芯片作为协处理时从主控侧运行模型的性能。从主控侧运行比从芯片侧直接运行要增加核间调度和传输时间。

两种性能一般都从时延和吞吐量两个维度衡量。benchmark结果中硬件时延和吞吐量被称为芯片时延和芯片吞吐，端到端时延和吞吐量被称为主控时延和主控吞吐。

当模型较大，单次推理时间较长，流水调度比较充分，调度和传输时间会被掩盖，吞吐量相差不大；当模型较小时，单次推理时间较短，流水调度不够充分，调度和传输时间掩盖不掉，吞吐量会有一定差别。

## 2.3 时延

时延是指单个推理任务从数据送入推理接口开始到获得推理结果之间所消耗的时间的平均值。一般来说，并行推理的任务越多，任务间冲突越频繁，单个任务的时延越长。因此，单个任务独立运行时时延为最小。

## 2.4 吞吐量

吞吐量是指单位时间内完成推理任务的次数。一般来说，并行推理会更加充分利用硬件资源。在一定的并行数量范围内，并行的任务越多，吞吐量越大。并行任务超过一定的数量，由于资源限制和调度开销，吞吐量会趋于平稳甚至下降。因此，可以根据并行任务数和吞吐量得到趋势图，进而推算出吞吐量最大值。

# 3 测试流程

## 3.1 准备模型和数据

精度测试需要准备标注后的测试集数据和标注数据，性能测试只需要准备测试集数据。

示例使用的测试集数据和标注数据放在/DEngine/data/datasets目录下。

## 3.2 源码编译

DEngine发布版本中包含了benchmark运行所需的主控和芯片侧库，用户可直接使用。同时提供了benchmark实现的源码供用户参考，如果需要重新编译源码，流程如下：

进入/DEngine/desdk/cpp目录，执行：

```
# sh build.sh
```

如果是linux-x64平台，生成的主控侧库放在/DEngine/desdk/platform/host_linux-x64/lib/libbenchmark.so。

参见《云天励飞DEngine集成开发指南》准备好csky芯片侧交叉编译环境，执行：

```
# sh build_device.sh
```

生成的芯片侧库放在/DEngine/desdk/platform/dev_linux-dp1000/lib/libbenchmark.so。

# 3.3 修改配置文件

benchmark执行脚本会自动解析同目录下benchmark.json，按照配置执行相应测试。

配置项意义：

- device_num 设备（芯片）数目，每个芯片执行相同测试，测试结果为总的吞吐统计
- test_suites 测试集数组，每个元素为一个测试集配置
- test_suite_name 测试集名称
- topn 分类模型精度测试，在推理结果前topn个结果中做匹配
- accuracy_datasets_path 精度测试数据集目录
- synset_path 模型分类结果文件路径
- val_path 精度测试标注文件路径
- fps_datasets_path 测试数据集路径
- models_root 待测试芯片模型根目录
- thread_list 线程数列表，如果配置为[1,2]，则表示分别以1线程和2线程做两次测试，分别输出测试结果
- batch_list batchsize列表，遍历列表中的batchsize对模型做多次测试
- nnp_num_list 与batch_list相对应，每种batchsize下配置的nnp个数
- resource_num_list 与batch_list相对应，每种batchsize下配置的调度资源数
- num_criteria 测试执行推理次数，测试程序会遍历读取文件，不足num_criteria的部分会循环使用
- accuracy_test 是否做精度测试
- fps_test 是否做吞吐量测试
- input_quesize 测试入口队列的深度，默认为32。此值如果太小会导致调度受限，硬件资源不能完全利用。
- test_models 待测试模型集合（字典）。

每一个键值对以模型子目录名作为key，do_test表示此模型是否执行测试；shape表示此模型的输入[h,w]；rgb表示模型输入格式；nnp_num_list和resource_num_list也可以在这里覆盖配置，仅对本模型生效。

配置文件示例内容如下：

```
{
  "device_num":1,
  "test_suites":
```

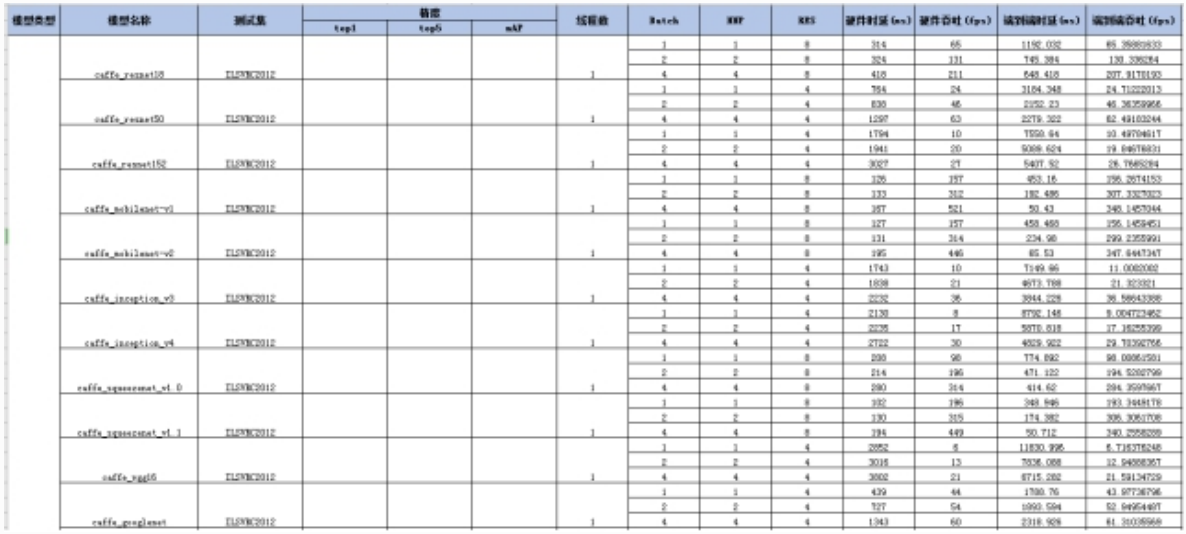
```
  [{
    "test_suite_name":"normal",
    "accuracy_datasets_path":"/DEngine/data/datasets/ILSVRC2012",
    "synset_path":"/DEngine/data/datasets/synset_1000.txt",
    "val_path":"/DEngine/data/datasets/ILSVRC2012_val.txt",
    "fps_datasets_path":"/DEngine/data/datasets/ILSVRC2012",
    "models_root":"/DEngine/model/dp1000/caffe_squeezenet_v1.1",
    "thread_list":[1],
    "batch_list":[1,2,4],
    "nnp_num_list":[1,2,4],
    "resource_num_list":[4,4,4],
    "topn":5,
    "time_criteria": 10.0,
    "num_criteria": 50,
    "accuracy_test": false,
    "fps_test": true,
    "input_quesize": 32,
    "test_models":
    {
      "caffe_squeezenet_v1.1": {"do_test":true, "shape":[227,227], "rgb":true,
"resource_num_list":[8,8,8]}
    }
  }]
}
```

## 3.4 开始测试

进入/DEngine/desdk/tools/benchmark目录，执行命令：

```
# sh /DEngine/run.sh benchmark.py
```

测试完成后，将生成benchmark.xlsx文件，内容如下图所示。



| 模型类型 | 模型名称 | 测试集 | 精度 | | | 场景数 | Batch | NNP | RES | 硬件时延(ms) | 硬件吞吐(fps) | 端到端时延(ms) | 端到端吞吐(fps) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | top1 | top5 | mAP | | | | | | | | |
| | caffe_resnet18 | ILSVRC2012 | | | | 1 | 1 | 1 | 8 | 314 | 65 | 1192.032 | 85.38883633 |
| | | | | | | | 2 | 2 | 8 | 324 | 131 | 745.384 | 130.336284 |
| | | | | | | | 4 | 4 | 8 | 418 | 211 | 648.418 | 207.8178193 |
| | caffe_resnet50 | ILSVRC2012 | | | | 1 | 1 | 1 | 4 | 784 | 24 | 3184.348 | 24.71222113 |
| | | | | | | | 2 | 2 | 4 | 830 | 46 | 2252.23 | 46.36359866 |
| | | | | | | | 4 | 4 | 4 | 1297 | 63 | 2279.302 | 62.48103244 |
| | caffe_resnet152 | ILSVRC2012 | | | | 1 | 1 | 1 | 4 | 1794 | 10 | 7550.64 | 10.49704617 |
| | | | | | | | 2 | 2 | 4 | 1941 | 20 | 5089.624 | 19.8657833 |
| | | | | | | | 4 | 4 | 4 | 3027 | 27 | 5407.92 | 28.7565284 |
| | caffe_mobilenet-v1 | ILSVRC2012 | | | | 1 | 1 | 1 | 8 | 326 | 157 | 653.16 | 156.2574153 |
| | | | | | | | 2 | 2 | 8 | 333 | 312 | 182.486 | 307.3325823 |
| | | | | | | | 4 | 4 | 8 | 387 | 521 | 50.43 | 340.1457044 |
| | caffe_mobilenet-v2 | ILSVRC2012 | | | | 1 | 1 | 1 | 8 | 327 | 157 | 458.468 | 156.1458451 |
| | | | | | | | 2 | 2 | 8 | 331 | 314 | 234.98 | 299.2355991 |
| | | | | | | | 4 | 4 | 8 | 395 | 446 | 65.53 | 347.6447347 |
| | caffe_inception_v3 | ILSVRC2012 | | | | 1 | 1 | 1 | 4 | 1743 | 10 | 7349.96 | 11.0080082 |
| | | | | | | | 2 | 2 | 4 | 1838 | 21 | 4673.788 | 21.3233021 |
| | | | | | | | 4 | 4 | 4 | 2232 | 36 | 3844.228 | 36.58643388 |
| | caffe_inception_v4 | ILSVRC2012 | | | | 1 | 1 | 1 | 4 | 2130 | 8 | 8732.148 | 8.0047123462 |
| | | | | | | | 2 | 2 | 4 | 2235 | 17 | 5870.818 | 17.15255399 |
| | | | | | | | 4 | 4 | 4 | 2722 | 30 | 4929.982 | 29.73396756 |
| | caffe_squeezenet_v1.0 | ILSVRC2012 | | | | 1 | 1 | 1 | 8 | 290 | 98 | 774.892 | 98.00862501 |
| | | | | | | | 2 | 2 | 8 | 314 | 196 | 471.122 | 194.5280799 |
| | | | | | | | 4 | 4 | 8 | 380 | 314 | 414.62 | 284.3597867 |
| | caffe_squeezenet_v1.1 | ILSVRC2012 | | | | 1 | 1 | 1 | 8 | 332 | 196 | 368.946 | 193.3448178 |
| | | | | | | | 2 | 2 | 8 | 330 | 315 | 174.382 | 306.3061708 |
| | | | | | | | 4 | 4 | 8 | 394 | 449 | 50.712 | 340.2558289 |
| | caffe_vgg16 | ILSVRC2012 | | | | 1 | 1 | 1 | 4 | 2052 | 8 | 11830.996 | 8.7353762248 |
| | | | | | | | 2 | 2 | 4 | 3015 | 13 | 7038.088 | 12.94008367 |
| | | | | | | | 4 | 4 | 4 | 3002 | 21 | 6715.202 | 21.59134729 |
| | caffe_googlenet | ILSVRC2012 | | | | 1 | 1 | 1 | 4 | 439 | 44 | 1780.76 | 43.97736796 |
| | | | | | | | 2 | 2 | 4 | 727 | 54 | 1060.594 | 52.84954497 |
| | | | | | | | 4 | 4 | 4 | 1343 | 60 | 2318.928 | 61.31038968 |

# 4 精度测试

精度测试是指使用指定数据集和标注结果对指定模型进行算法精度测试，以评估芯片运行的精度损失。

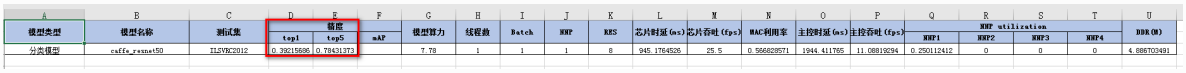
精度测试需要配置的key有accuracy_test（true）、accuracy_datasets_path、val_path。测试时使用的线程、batch等参数不影响结果，都用1即可。

精度测试分为分类模型精度测试和检测模型精度测试。分类模型精度测试目标为测试TOP1、TOP5的分类精度；检测模型的精度测试目标为mAP精度。

## 4.1 分类模型精度测试

分类模型精度测试时，需要准备好数据集文件`synset_1000.txt`、标签文件`ILSVRC2012_val.txt`，然后在benchmark.json文件中的synset_path和val_path字段中分别指定这两个文件，并设置accuracy_test为true。

精度测试结果在benchmark.xlsx文件中体现，如下图所示。



| 模型类型 | 模型名称 | 测试集 | 精度 | | mAP | 模型算力 | 线程数 | Batch | NNP | RES | 芯片时延(ms) | 芯片吞吐(fps) | MAC利用率 | 主控时延(ms) | 主控吞吐(fps) | NNP utilization | | | | DDR(M) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | top1 | top5 | | | | | | | | | | | | NNP1 | NNP2 | NNP3 | NNP4 | |
| 分类模型 | caffe_resnet50 | ILSVRC2012 | 0.73215686 | 0.78431373 | | 7.78 | 1 | 1 | 1 | 8 | 945.1764526 | 25.5 | 0.566828571 | 1944.411765 | 11.08819294 | 0.250112412 | 0 | 0 | 0 | 4.886703491 |

## 4.2 检测模型精度测试

**注：目前检测模型精度测试功能正在完善中，暂不支持。**

# 5 性能测试

性能测试是指使用指定数据集对指定模型进行多种模式和参数下的性能测试，以评估芯片运行的效率。

性能测试主要分为吞吐量测试和时延测试两种。

## 5.1 吞吐量测试

吞吐量测试模式下，测试程序将所有测试数据一次性发给芯片，驱动芯片以该配置下的最大能力执行，得到最大的吞吐量数据。

由于芯片同一时间处理能力有限，未及时处理的测试数据会暂时存放在测试入口队列，等待前一批数据处理完之后进行处理。而在测试入口队列中的等待时间也会被记入时延，导致吞吐量测试模式下时延数据虚高，因此吞吐量测试模式下不用关注时延数据。

测试入口队列深度（input_quesize）减小会降低时延，但太小会导致硬件资源不能充分调度，导致吞吐量下降。

通过配置线程数、batch、nnp、res的不同组合来进行测试，可以获取一个最优配置达到最优的吞吐性能。一般情况，小模型可以适当配置res高一些提高性能，大模型配置多线程以提高吞吐。未达到最高性能应将nnp尽量配满，但多nnp情况下芯片mac利用率可能会有下降。

## 5.2 时延测试

时延测试模式下，测试程序将测试数据按照一定规则发给芯片，得到该规则下数据的最大、最小和平均时延。

数据的发送规则一般为单一发送、平均间隔发送和泊松分布发送。

**注：目前时延测试功能正在完善中，暂不支持。**