移动端深度学习框架

深度学习框架和加速

Exported on 11/13/2020

Table of Contents

小组讨论会议纪要-20200102

1.目标：合力打造公司级移动端开源框架，性能实现业界领先。

2. 现有框架：CSIG优图rapidnet, ncnn

TEG AI平台部feathercnn（内部版&开源版）

新框架名字: 进一步讨论

3. CodeBase:

优图提出以当前rapidnet为codebase，ncnn多轮重构，跨平台、模型支持良好

PCG、IEG侧OK

AI平台部期望等rapidnet开源后分析merge难度，初步认为OK

4. 各部门主要诉求：

4.1 机型覆盖率

微信：crash率不高于ncnn

PCG：安卓、iOS等各个平台稳定，对标ncnn

IEG研发效能部：覆盖安卓、iOS、iMac、Windows

4.2 性能：

优图rapidnet：当前持续优化中，领先ncnn,mnn

PCG已测试：部分模型squeezenet, 生成模型性能有待提升；

期望提供各个模型、平台benchmark

微信：ncnn性能当前基本需求

AI平台部：期望提供benchmark

5. 分工协作（部分、视具体情况灵活调整）

PCG、优图：主要负责pytorch、caffe模型转换

AI平台部：主要负责TensorFlow模型转换

优图：CV类网络支持和性能优化

AI平台部：RNN模型支持和优化、benchmark开发

IEG研发效能部：协助支持内部模型

微信：适配微信扫一扫相关模型和协助提升覆盖率

6. MileStone:

1. 2月中，优图Rapidnet开源，各方讨论确定code base。
2. 2月中，各部门模型需求、op需求、主要问题
3. 3月中，AI平台部完成集成+评测，协助支持部分op

4. to do

7. 上线计划

优图：各大业务已上线

PCG：手Q、微视、P图部分业务已上线

AI平台部、IEG研发效能部：3月份测试完成后评估

微信：需要验证稳定性和crash率后进一步进行

1 第一次会议纪要

1.1 日期

03 Jan 2020

1.2 参会人

darrenyao(姚达)[1]Unknown User (alohali)[2]anguszhang(张应国)[3]billpeng(彭彪)[4]caronzhang(张银锋)[5]danierdeng(邓民文)[6]fangyunzhou(周方云)[7]Unknown User (gregwang)[8]merrickmei(梅利健)[9]lucasktian(田凯)[10]lincolnlin(林榆耿)[11]junezhan(詹成君)[12]

1.3 目标

合力打造公司级移动端开源框架，性能实现业界领先。

1.4 讨论事项

条目	何人	说明
现有框架	darrenyao(姚达)[13]	CSIG优图rapidnet, ncnn TEG AI平台部feathercnn（内部版&开源版）
新框架名字	darrenyao(姚达)[14]	统一为TNN or TCNN，方便协同
CodeBase	darrenyao(姚达)[15]	• 优图提出以当前rapidnet为codebase，ncnn多轮重构，跨平台、模型支持良好 • PCG、IEG侧OK； • AI平台部期望等rapidnet开源后分析merge难度，初步认为OK

1 https://iwiki.woa.com/display/~darrenyao
2 https://iwiki.woa.com/display/~alohali
3 https://iwiki.woa.com/display/~anguszhang
4 https://iwiki.woa.com/display/~billpeng
5 https://iwiki.woa.com/display/~caronzhang
6 https://iwiki.woa.com/display/~danierdeng
7 https://iwiki.woa.com/display/~fangyunzhou
8 https://iwiki.woa.com/display/~gregwang
9 https://iwiki.woa.com/display/~merrickmei
10 https://iwiki.woa.com/display/~lucasktian
11 https://iwiki.woa.com/display/~lincolnlin
12 https://iwiki.woa.com/display/~junezhan
13 https://iwiki.woa.com/display/~darrenyao
14 https://iwiki.woa.com/display/~darrenyao
15 https://iwiki.woa.com/display/~darrenyao

1.5 MileStone

1. 2020年2月中旬，优图Rapidnet开源，各方讨论确定code base。
2. 2020年2月中旬，各部门模型需求、op需求、主要问题
3. 2020年3月中旬，AI平台部完成集成+评测，协助支持部分op

1.6 上线计划

- [x] **优图**：各大业务已上线
- [x] **PCG**：手Q、微视、P图部分业务已上线
- [ ] **AI平台部、IEG研发效能部**：3月份测试完成后评估
- [ ] **微信**：需要验证稳定性和crash率后进一步进行

1.7 各部门主要诉求

- **机型覆盖率**

- [ ] 微信：crash率不高于ncnn
- [ ] PCG：安卓、iOS等各个平台稳定，对标ncnn
- [ ] IEG研发效能部：覆盖安卓、iOS、iMac、Windows

- **性能：**

- [ ] 优图rapidnet：当前持续优化中，领先ncnn,mnn
- [ ] PCG已测试：部分模型squeezenet, 生成模型性能有待提升；
- [ ] 期望提供各个模型、平台benchmark
- [ ] 微信：ncnn性能当前基本需求
- [ ] AI平台部：期望提供benchmark

- **分工协作（部分、视具体情况灵活调整）**

- [ ] PCG、优图：主要负责pytorch、caffe模型转换
- [ ] AI平台部：主要负责TensorFlow模型转换
- [ ] 优图：CV类网络支持和性能优化
- [ ] AI平台部：RNN模型支持和优化、benchmark开发
- [ ] IEG研发效能部：协助支持内部模型
- [ ] 微信：适配微信扫一扫相关模型和协助提升覆盖率

2 文档

2.1 TNN简介

2.1.1 项目介绍

<TO-DO>

2.1.2 框架特点

2.1.2.1 高性能

- 汇编Neon指令调优, 支持低精度int8量化模型。
- OpenCL指令调优, 针对Adreno和Mali GPU定制优化。
- Metal指令调优, 性能优于原生CoreML。

2.1.2.2 低内存

- 网络内不同层输出内存复用。
- 支持多线程多网络内存复用。

2.1.2.3 通用性

- 支持市面上绝大多数常用的 cnn 网络,目前已经测试通过的 cnn 网络有 28 个.
- 支持 onnx 算子 55 个,caffe 算子26 个.
- iOS支持8.0+系统, Android 支持4.0+系统.

2.1.2.4 易用性

- 支持常用cv mat格式输入及预处理。
- 提供CPU，GPU输入输出支持。

2.2 使用文档

2.2.1 算子支持列表

Operators	cpu	armv7	armv8	opencl	metal
Abs	yes	yes	yes	yes	yes
Add	yes	yes	yes	yes	yes
BatchNorm	yes	yes	yes	yes	yes

Operators	cpu	armv7	armv8	opencl	metal
BiasAdd	yes	yes	yes	yes	yes
Clip	yes	yes	yes	yes	yes
Concat	yes	yes	yes	yes	yes
Convolution	yes	yes	yes	yes	yes
Convolution | depthwise	yes	yes	yes	yes	yes
Convolution | group	yes	yes	yes	yes	yes
Deconvolution	yes	yes	yes	yes	yes
Deconvolution | depthwise	yes	yes	yes	yes	yes
Deconvolution | group	yes	yes	yes	yes	yes
DetectionOutput	yes				
Div	yes	yes	yes	yes	yes
Elu	yes	yes	yes	yes	yes
Exp	yes	yes	yes	yes	yes
Flatten				yes	
HardSigmoid	yes	yes	yes	yes	yes
HardSwish	yes	yes	yes	yes	yes
InnerProduct	yes	yes	yes	yes	yes
InstanceNorm	yes	yes	yes	yes	yes
LeakyRelu	yes	yes	yes	yes	yes
LogSigmoid	yes	yes	yes	yes	yes
LRN	yes				
Max	yes	yes	yes	yes	yes
Min	yes	yes	yes	yes	yes
Mul	yes	yes	yes	yes	yes

Operators	cpu	armv7	armv8	opencl	metal
Neg	yes	yes	yes	yes	
Normalize	yes	yes	yes	yes	yes
Pad	yes	yes	yes	yes	yes
Permute	yes	yes	yes	yes	
Pooling | Max	yes	yes	yes	yes	yes
Pooling | Avg	yes	yes	yes	yes	yes
Pooling | Global	yes	yes	yes	yes	yes
Pow	yes	yes	yes	yes	yes
PRelu	yes	yes	yes	yes	yes
PriorBox	yes			yes	
ReduceMean	yes	yes	yes	yes	yes
Relu	yes	yes	yes	yes	yes
Relu6	yes	yes	yes	yes	yes
Reorg	yes			yes	
Reshape	yes	yes	yes	yes	yes
Scale	yes	yes	yes	yes	yes
ShuffleChannel	yes	yes	yes	yes	yes
Sigmoid	yes	yes	yes	yes	yes
Sign	yes	yes	yes	yes	yes
Softmax	yes	yes	yes	yes	yes
Softplus	yes	yes	yes	yes	yes
Split	-			yes	
Sqrt	yes	yes	yes	yes	yes
StrideSlice	yes	yes	yes	yes	yes

Operators	cpu	armv7	armv8	opencl	metal
Sub	yes	yes	yes	yes	yes
Sum					
Tanh	yes	yes	yes	yes	yes
Upsample	yes	yes	yes	yes	yes
Total |55	**52**	**48**	**48**	**52**	**46**

2.2.2 编译

2.2.2.1 一、iOS库编译

1. 编译环境要求

- Mac系统, Xcode IDE
- cmake（使用2.8及以上版本）

- 2. 编译步骤

1）切换到脚本目录

```
cd /scripts
```

2）执行编译脚本

```
./build_ios.sh
```

编译完成后，在目录`platforms/ios`下产生`xxx.framework`库和`xxx.bundle`资源
3）添加到工程
- 在iOS app工程的根目录中添加`xxx.framework`库和`xxx.bundle`资源;
- 在app Xcode工程的设置中找到`Build Setting -> Linking -> Other Linker Flags`选项;
- 添加`-force_load "$(SRCROOT)/xxx.framework"`;

- 2.2.2.2 二、Android库编译

1. 环境要求

依赖库

- cmake（使用3.6及以上版本）

- NDK配置
  - 下载ndk版本(>=15c) <">https://developer.android.com/ndk/downloads>(see page 11)
  - 配置环境变量 `export ANDROID_NDK=<ndk_path>`

- 2. 编译步骤

  1）切换到脚本目录

  ```
  cd /scripts
  ```

  2）编辑build_android.sh修改配置选项

  ```
  ABIA32="armeabi-v7a with NEON"
  ABIA64="arm64-v8a"
  STL="c++_static"
  SHARED_LIB="ON"                 # ON表示编译动态库，OFF表示编译静态库
  ARM="ON"                        # ON表示编译带有Arm CPU版本的库
  OPENMP="ON"                     # ON表示打开
  OPENCL="ON"                     # ON表示编译带有Arm GPU版本的库
  OPENCL_PROFILING=0              # 1表示打开GPU的profiling功能，可以打印每一层的耗时，
                                  # 仅在性能分析时打开，否则会影响性能。
  SHARING_MEM_WITH_OPENGL=0       # 1表示OpenGL的Texture可以与OpenCL共享
  ```

  3）执行编译脚本

  ```
  ./build_android.sh
  ```

  编译完成后，在当前目录的release目录下生成对应的armeabi-v7a库，arm64-v8a库和include头文件。

2.2.2.3 三、Linux库编译及跨平台交叉编译

1. 环境要求

依赖库

- cmake（使用2.8及以上版本）

- 2. 编译步骤

  1）切换到脚本目录

  ```
  cd /scripts
  ```

  2）编辑build_arm_linux.sh修改配置选项

  ```
  SHARED_LIB="ON"                 # ON表示编译动态库，OFF表示编译静态库
  ARM="ON"                        # ON表示编译带有Arm CPU版本的库
  OPENMP="ON"                     # ON表示打开
  OPENCL="OFF"                    # ON表示编译带有Arm GPU版本的库
  CC=aarch64-linux-gnu-gcc        # 指定C编译器
  CXX=aarch64-linux-gnu-g++       # 指定C++编译器
  ```

  3）执行编译脚本

  ```
  ./build_arm_linux.sh
  ```

2.2.2.4 四、Mac库编译

2.2.2.5 五、Windows库编译

2.2.2.6 编译参数option说明

Option	默认值	说明
TNN_CPU_ENABLE	OFF	代码source/device/cpu编译开关，代码仅用用于调试以及UnitTest基准测试，实现全部为c++代码，不包含特定CPU加速指令。
TNN_X86_ENABLE	OFF	代码source/device/x86编译开关, 当前适配openvino实现，后续会迁入更多加速代码实现。
TNN_ARM_ENABLE	OFF	代码source/device/arm编译开关，代码包含neon加速指令, 且部分实现了int8加速。
TNN_METAL_ENABLE	OFF	代码source/device/metal编译开关，代码包含metal加速指令。
TNN_OPENCL_ENABLE	OFF	代码source/device/opencl编译开关，代码包含opencl加速指令。
TNN_CUDA_ENABLE	OFF	代码source/device/cuda编译开关，代码包含cuda加速指令, 当前仅迁移了小部分实现。
TNN_DSP_ENABLE	OFF	代码source/device/dsp编译开关，当前适配snpe实现。
TNN_ATLAS_ENABLE	OFF	代码source/device/atlas编译开关，当前适配华为atlas加速框架。
TNN_NPU_ENABLE	OFF	代码source/device/npu编译开关，当前适配HiAI加速框架。
TNN_SYMBOL_HIDE	ON	加速库符号隐藏，release发布默认非public接口符号不可见。
TNN_OPENMP_ENABLE	OFF	OpenMP开关，控制是否打开openmp加速。
TNN_BUILD_SHARED	ON	动态库编译开关，关闭则编译静态库。
TNN_TEST_ENABLE	OFF	test代码编译开关
TNN_UNIT_TEST_ENABLE	OFF	unit test编译开关，打开unit test编译开关会自动打开TNN_CPU_ENABLE开关，作为测试基准。
TNN_PROFILER_ENABLE	OFF	性能调试开关，打开后会打印更多性能信息，仅用于调试。
TNN_QUANTIZATION_ENABLE	OFF	量化工具编译开关
TNN_BENCHMARK_ENABLE	OFF	benchmark开关，打开后支持model weights文件为空，可自动生成数据。

更多帮助：Markdown使用指引[16] Markdown问题反馈[17] 助手号DevITHelper[18]

2.2.3 模型转换

2.2.3.1 一、onnx2tnn

onnx2tnn 工具的主要作用是将 onnx 模型转换成 tnn 自定义的格式。目前onnx2tnn支持CNN 网络中大部分的常用 OP。

1. Mac 系统下使用

1. 1. 环境配置

1. 安装最新版protobuf，最低要求3.7.1

```
brew install protobuf
brew link --overwrite protobuf
```

note：版本不一致会导致protobuf库找不到，最好与转化库so所用的版本保持一致，否则都用最新版本，so重新编译。
2. 安装最新版python，最低要求3.6

```
brew install python3
```

3. 安装最新版PyTorch，最低要求1.0.0

```
pip3 install torch torchvision
```

4. 安装最新版onnx，numpy模块

```
pip3 install onnx
pip3 install numpy
```

1. 2. 工具编译

在 mac 系统上提供了编译的脚本，可以直接编译

```
cd /tools/onnx2tnn/onnx-converter
./build.sh
```

1. 3. 工具的使用

```
python3 onnx2tnn.py model.onnx -version=algo_version -optimize=1 -half=0

参数说明：
-version
模型版本号，便于后续算法进行跟踪
```

16 https://iwiki.woa.com/pages/viewpage.action?pageId=43910014
17 https://iwiki.woa.com/pages/viewpage.action?pageId=40633510
18 wxwork://message?username=DevITHelper

```
-optimize
1（默认，开启）：用于对模型进行无损融合优化，，如BN+Scale等f融合进Conv
0 ：如果融合报错可以尝试设为此值

-half
1：转为FP16模型存储，减小模型大小。
0（默认，不开启）：按照FP32模型存储。
Note：实际计算是否用FP16看各个平台特性决定，移动端GPU目前仅支持FP16运算
```

2. Linux(Centos 7.2) 系统下使用

2. 1. 环境搭建

1. protobuf的安装 （version >= 3.4.0）
   直接从官网下载最新版本的 protobuf，然后按照文档安装即可。
2. 安装 python3 以及 python-devel（version >= 3.6. 8）

   ```
   yum install python3 python3-devel
   ```

3. 安装 python 的依赖库

   ```
   pip3 install pytest numpy torch torchvision onnx onnxruntime
   ```

4. cmake （version >= 3.0）
   从的官网下载最新版本的 cmake，然后按照文档安装即可。建议使用最新版本的 cmake。

2. 2. 编译

切换到工具目录

```
cd /tools/onnx2tnn/onnx-converter
```

1. pybind11 的编译

```
cd /tools/onnx2tnn/onnx-converter/pybind11/
mkdir build
cd build
cmake ..
make check -j 4
```

2. onnx_converter 的编译

```
# 新建build目录进行编译
mkdir build
cd build cmake ./../
make -j 4

#复制生成的so
cp ./*.so ./../

#删除build目录
cd ./../
rm -r build
```

2. 3. 工具的使用

```
python3 onnx2tnn.py model.onnx -version=algo_version -optimize=1 -half=0
```

参数说明：
-version
模型版本号，便于后续算法进行跟踪

-optimize
1（默认，开启）：用于对模型进行无损融合优化，，如BN+Scale等f融合进Conv
0 ：如果融合报错可以尝试设为此值

-half
1：转为FP16模型存储，减小模型大小。
0（默认，不开启）：按照FP32模型存储。
Note：实际计算是否用FP16看各个平台特性决定，移动端GPU目前仅支持FP16运算

3. onnx2tnn支持的算子

onnx	tnn
AveragePool	Pooling / Pooling3D
BatchNormalization	BatchNormCxx
Clip	ReLU6
Concat	Concat
Conv	Convolution3D / Convolution
ConvTranspose(ConvTranspose+BatchNormalization)	Deconvolution3D / Deconvolution
DepthToSpace	Reorg
Div	Mul
Gemm	InnerProduct
GlobalAveragePool	Pooling / Pooling3D
GlobalMaxPool	Pooling / Pooling3D
InstanceNormalization	InstBatchNormCxx
LeakyRelu	PReLU
MaxPool	Pooling / Pooling3D
Mul	Mul
Normalize(ReduceL2 + Clip+ Expand+Div)	Normalize
PReLU	PReLU
Pad	Pad / Pad3D
ReduceMean	ReduceMean
Relu	ReLU
Reshape	Reshape
ShuffleChannle(Reshape+Transpose+Reshape)	ShuffleChannle
Slice	StridedSlice
Softmax(Exp + ReduceSum + Div)	SoftmaxCaffe
Softmax(Transpose + Reshape + Softmax + Reshape + Transpose)	SoftmaxCaffe

onnx	tnn
Softplus	Softplus
Split	SplitV
Sub	BatchNormCxx
Tanh	TanH
Tile	Repeat
Transpose	Transpose
Upsample	Upsample

4. 使用限制

1. 目前 rpn 只支持 4 维(nchw)的数据类型。
2. 建议将模型输入的 batch size 设置为 1，不建议将 batch_size,设置为比较大的值。
3. 目前暂不支持 pool 中的 pad 为非对称的情况。（inceptionv1 模型中的 "pool5/7x7_s1_1" 会出现这个情况）

2.2.3.2 二、caffe2onnx

caffe2onnx 工具是将 caffe 格式的模型转换成 onnx 格式的模型，然后再利用 onnx2tnn工具将 onnx 模型转换成 tnn 格式的模型。tnn 为了支持 caffe 模型，将 onnx 当做中间层。
项目 URL：http://git.code.oa.com/deep_learning_framework/TNN-caffe2onnx.git

1. 环境搭建(Mac and Linux)

1. protobuf(version >= 3.4.0)
   从官网现在最新的版本，按照文档进行安装。如果是 Mac 系统直接使用 brew 进行安装。

   ```
   brew install protobuf
   ```

2. python(version >= 3.6)

   ```
   brew install python
   ```

3. onnx(1.6.0)

   ```
   pip3 install onnx numpy
   ```

2. 工具使用

1. 项目下载

   ```
   git clone http://git.code.oa.com/deep_learning_framework/TNN-caffe2onnx.git
   ```

2. 使用

   ```
   python3 convert2onnx.py [-h] [-o ONNX_FILE] proto_file caffe_model_file
   ```

```
positional arguments:
  proto_file        the path for prototxt file, the file name must end with
                    .prototxt
  caffe_model_file  the path for caffe model file, the file name must end with
                    .caffemodel!
optional arguments:
  -h, --help        show this help message and exit
  -o ONNX_FILE      the path for generate onnx file
```

3. 使用限制

1. 目前 caffe2onnx 的工具目前只支持最新版本的 caffe 的格式,所以在使用 caffe2onnx工具之前需要将老版本的 caffe 网络和模型转换为新版. caffe 自带了工具可以把老版本的caffe 网络和模型转换为新版本的格式. 具体的使用方式如下:

   ```
   upgrade_net_proto_text [老prototxt] [新
   upgrade_net_proto_binary [老caffemodel] [新
   ```

   修改后的输入的格式如下所示:

   ```
   layer {
     name: "data"
     type: "input"
     top: "data"
     input_param { shape: { dim: 1 dim: 3 dim: 224 dim: 224 } }
   }
   ```

4. caffe2onnx 支持的算子

Number	caffe layer	onnx operator
1	BatchNorm	BatchNormalization
2	BatchNorm + Scale	BatchNormalization
3	Concat	Concat
4	Convolution	Conv
5	ConvolutionDepthwise	Conv
6	Deconvolution	ConvTranspose
7	DetectionOutput	DetectionOutput(customer defination)
8	Dropout	Dropout
9	Eltwise	Mul/Add/Max
10	Flatten	Reshape
11	InnerProduct	Reshape + Gemm
12	LRN	LRN
13	MaxUnPool	MaxUnPool
14	PReLU	PRelu
15	Permute	Transpose
16	Pooling	MaxPool/AveragePool/GlobalMaxPool/GlobalAveragePool

Number	caffe layer	onnx operator
17	PriorBox	PriorBox(customer defination)
18	ReLU	Relu/LeakyRelu
19	ReLU6	Clip
20	Reshape	Reshape
21	Scale	Mul + Reshape
22	ShuffleChannel	Reshape + Transpose + Reshape
23	Sigmoid	Sigmoid
24	Slice	Slice
25	Softmax	Softmax
26	Upsample	Resize

2.2.3.3 F&A

1. 网络问题

```
//遇到资源无法下载，可切换到GestWiFi，或尝试用代理

export http_proxy=http://web-proxy.oa.com:8080/
export all_proxy=https://web-proxy.oa.com:8080/
export ftp_proxy=ftp://web-proxy.oa.com:8080/
export socks_proxy=socks://web-proxy.oa.com:8080/
export ALL_PROXY=https://web-proxy.oa.com:8080/
export https_proxy=https://web-proxy.oa.com:8080/

//mac下homebrew安装
//https://zhuanlan.zhihu.com/p/59805070
//https://brew.sh/index_zh-cn
//替换国内镜像的安装脚本
```

更多帮助：Markdown使用指引[19] Markdown问题反馈[20] 助手号DevITHelper[21]

2.2.4 模型量化工具

2.2.4.1 一、量化的作用

量化将网络中主要算子（Convolution，Pooling，Binary等）由原先的浮点计算转成低精度的Int8计算，减少模型大小并提升性能。
PS：

19 https://iwiki.woa.com/pages/viewpage.action?pageId=43910014
20 https://iwiki.woa.com/pages/viewpage.action?pageId=40633510
21 wxwork://message?username=DevITHelper

1、关于KL量化方法，可以参考：http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf

2.2.4.2 二、编译

1. 编译脚本

```
cd /platforms/linux/
./build_quanttool.sh -c
```

2. 编译输出

量化模型命令：<path_to_tnn>/platforms/linux/build/quantization_cmd

2.2.4.3 三、量化工具的使用

1. 命令

```
./quantization_cmd [-h] [-p] [-m] [-i] [-b] [-w] [-n] [-s] [-c]
```

2. 参数说明

命令参数	是否必须	带参数	参数说明
-h, --help			输出命令提示。
-p, --proto	√	√	指定rapidproto模型描述文件。
-m, --model	√	√	指定rapidmodel模型参数文件。
-i, --input_path	√	√	指定量化输入文件夹路径。目前支持格式为：• 文本文件（文件后缀为.txt）• 常用图片格式文件（文件后缀为 .jpg .jpeg .png .bmp）会将此目录下面的所有文件作为输入。
-b, --blob_method		√	指定feature map的量化方法：• 0 Min-Max方法（默认）• 2 KL方法
-w, --weight_method		√	指定weights的量化方法：• 0 Min-Max方法（默认）• 1 ADMM方法
-n, --mean		√	预处理，对输入数据各通道进行mean操作，参数格式为：0.0,0.0,0.0
-s, --scale		√	预处理，对输入数据各通道进行scale操作，参数格式为：1.0,1.0,1.0
-c, --merge_channel		√	在量化feature map的时候是否对所有通道一起计算，否则是各通道单独计算。

3. 量化输入

- 输入数据的选取
  输入数据需要包括典型的输入，否则影响输出结果的精度，图片数量在20~50左右。

- 输入预处理
  对输入数据进行预处理，主要通过mean和scale参数进行。公式为：
  input_pre = (input - mean) * scale

4. 命令输出

在执行命令的当前目录下会生成两个文件：

- model_quantized.rapidproto -- 量化后的模型描述文件；
- model_quantized.rapidmodel -- 量化后的模型参数文件；

更多帮助：Markdown使用指引[22] Markdown问题反馈[23] 助手号DevITHelper[24]

2.2.5 示例代码

2.2.5.1 Demo 代码介绍

一、Android Demo 介绍

TODO

二、Arm Linux Demo 介绍

1. Init 函数流程

1.1 在 tnn::ModelConfig 中指明模型文件路径，创建tnn::TNN 实例。
具体代码：

```
tnn::ModelConfig model_config;
model_config.params.push_back(buffer);
model_config.params.push_back(model_file);
CHECK_TNN_STATUS(tnn_.Init(model_config));
```

1.2. 在 tnn::NetworkConfig 中指明设备类型等信息，创建tnn::Instance 实例。
具体代码:

```
tnn::NetworkConfig config;
config.device_type = tnn::DEVICE_ARM;
tnn::Status error;
net_instance_      = tnn_.CreateInst(config, error);
CHECK_TNN_STATUS(error);
```

1.3. 获取输入输出信息。
具体代码

22 https://iwiki.woa.com/pages/viewpage.action?pageId=43910014
23 https://iwiki.woa.com/pages/viewpage.action?pageId=40633510
24 wxwork://message?username=DevITHelper

```
    CHECK_TNN_STATUS(net_instance_->GetAllInputBlobs(input_blobs_));
    CHECK_TNN_STATUS(net_instance_->GetAllOutputBlobs(output_blobs_));
```

2. Forward 函数流程

2.1. 预处理及数据传入。
具体代码:

```
    tnn::BlobConverter input_blob_convert(input_blobs_.begin()->second);
    CHECK_TNN_STATUS(
        input_blob_convert.ConvertFromMat(input_mat, input_convert_param_,
nullptr));
```

2.2. 前向计算。
具体代码:

```
    CHECK_TNN_STATUS( net_instance_->Forward());
```

2.3. 数据传出及后处理。
具体代码:

```
    tnn::BlobConverter output_blob_convert(output_blobs_.begin()->second);
    CHECK_TNN_STATUS(
        output_blob_convert.ConvertToMat(output_mat, output_convert_param_,
nullptr));
```

三、NCNN 模型使用及接口介绍

1. 模型使用

使用ncnn 模型时，需要在 TNN 初始化参数 NetworkConfig 中指明 ModelType 为 MODEL_TYPE_NCNN。
具体代码参考：

```
    ModelConfig model_config;
    model_config.model_type = MODEL_TYPE_NCNN;
    TNN net;
    Status ret = net.Init(model_config);
    auto instance = net.CreateInst(network_config, ret);
```

TNN Instance 在创建时需要声明默认InputShape，通常ncnn.param 的Input 层中会说明。如果其中未指明的话，需要在创建Instance 代码中指明。
具体参考:

```
    InputShapesMap input_shape;
    input_shape["input_name"] = {1, 3, 224, 224};
    auto instance = net.CreateInst(network_config, ret);
```

其他方面使用与正常调用流程相同，可具体参考其他文档。

2. 当前适配完成的NCNN Op

目前已支持以下ncnn Op, Int8 模型适配还在进行中。

Operators
Convolution

Operators

ConvolutionDepthWise
Deconvolution
DeconvolutionDepthWise
BatchNorm
InnerProduct
Pooling
BinaryOp
Concat
AbsVal
ReLU
Softmax
Sigmoid
HardSigmoid
HardSwish
TanH
LRN
ShuffleChannel
Reshape
Spilit
Flatten
Dropout

3. 样例模型

样例模型可从 https://git.code.oa.com/yt-inference/yt-rapidnet-test-models 获取。
如 resnet50, mobilenet_v1, squeezenet 等。

更多帮助：Markdown使用指引[25] Markdown问题反馈[26] 助手号DevITHelper[27]

25 https://iwiki.woa.com/pages/viewpage.action?pageId=43910014
26 https://iwiki.woa.com/pages/viewpage.action?pageId=40633510
27 wxwork://message?username=DevITHelper

2.3 开发文档

2.3.1 模型性能分析工具

2.3.1.1 一、OpenCL平台

1.工具作用

将具体模型的各层GPU耗时打印出来或者执行整个网络测试性能。

2.使用步骤

1. 进入<path_to_rpn>/TNN/platforms/android目录。
2. 新建models文件夹，将待测试模型的rapidproto文件拷贝进models文件夹，并且改名为test.rapidproto。
3. 修改profiling_model_android_ocl.sh脚本。(如果要打印整个模型的性能，则将OPENCL_PROFILING初始化为0，否则会打印模型的每一层的耗时)
4. 执行脚本 ./profiling_model_android_ocl.sh -64 -c -p
5. 执行完毕后，结果保存在dump_data_ocl/test_log.txt文件中。

2.3.1.2 二、iOS平台

1.使用步骤

1. 添加测试模型
   在<path_to_tnn>/model目录下添加测试模型，每个模型一个文件夹，文件夹中包含以proto和model结尾的模型文件。目前工程中已有模型squeezenetv1.0和shufflenetv2
2. 打开benchmark工程
   进入目录<path_to_tnn>/benchmark/benchmark_ios，打开benchmark工程
3. 修改签名，并在真机上运行
   根据自己的苹果开发者账号修改benchmark工程和tnn依赖工程设置Signing & Capabilities中的签名，如因个人账户遇到签名冲突，可尝试修改Bundle Identifier。

更多帮助：Markdown使用指引[28] Markdown问题反馈[29] 助手号DevITHelper[30]

28 https://iwiki.woa.com/pages/viewpage.action?pageId=43910014
29 https://iwiki.woa.com/pages/viewpage.action?pageId=40633510
30 wxwork://message?username=DevITHelper

2.3.2 模型结果校验工具

2.3.2.1 一、工具的作用

校验对应平台（OpenCL，Metal，Cuda，ARM）的模型输出结果是否正确。

2.3.2.2 二、编译

编译model_check工具需要将以下宏设置为ON：

- TNN_CPU_ENABLE
- TNN_MODEL_CHECK_ENABLE
- 对应device的宏，如TNN_OPENCL_ENABLE, TNN_ARM_ENABLE

2.3.2.3 三、校验工具使用

1. 命令

```
./model_check [-h] [-p] [-m] [-d] [-i] [-n] [-s] [-o]
```

2. 参数说明

命令参数	是否必须	带参数	参数说明
-h, --help			输出命令提示。
-p, --proto	√	√	指定rapidproto模型描述文件。
-m, --model	√	√	指定rapidmodel模型参数文件。
-d, --device	√	√	指定模型执行的平台，如OPENCL，ARM，METAL等。
-i, --input_path		√	指定输入文件。目前支持格式为：• 文本文件（文件后缀为.txt）• 常用图片格式文件（文件后缀为 .jpg .jpeg .png .bmp）如果不指定，则会使用 (-1, 1) 随机输入
-n, --mean		√	预处理，对输入数据各通道进行mean操作，参数格式为：0.0,0.0,0.0
-s, --scale		√	预处理，对输入数据各通道进行scale操作，参数格式为：1.0,1.0,1.0
-o, --output			是否保存最终的输出。

2.3.2.4 四、执行脚本

1. Android

1.1 模型准备

将待校验的模型的rapidproto和rapidmodel文件拷贝进<path_to_tnn>/platforms/android/modles，并改名为test.rapidproto和test.rapidmodel

1.2 执行脚本

```
cd /platforms/android/
./model_check_android.sh -c -p
```

2. Linux

2.1. 编译脚本

```
cd /platforms/linux/
./build_model_check.sh -c
```

2.2. 执行命令

```
/platforms/linux/build/model_check -p  -m  -d
```

2.3.2.5 五、工具限制

- 目前只支持fp32的模型校验；
- 目前只针对fp32精度下的结果进行校验；

更多帮助：Markdown使用指引[31] Markdown问题反馈[32] 助手号DevITHelper[33]

31 https://iwiki.woa.com/pages/viewpage.action?pageId=43910014
32 https://iwiki.woa.com/pages/viewpage.action?pageId=40633510
33 wxwork://message?username=DevITHelper