CaffeOnACL

User Manual

2017-10-11

OPEN AI LAB

Revision Record

Date	Rev	Change Description	Author
2017-09-20	0.30	Intial	
2017-10-11	0.40	Validation on Arm Compute Library v17.09	

catalog

1 Purpose

This guide help user utilize the code of CaffeOnACL (Caffe+ACL) to improve the performance of their applications based on the Caffe framework.

2 Terminology

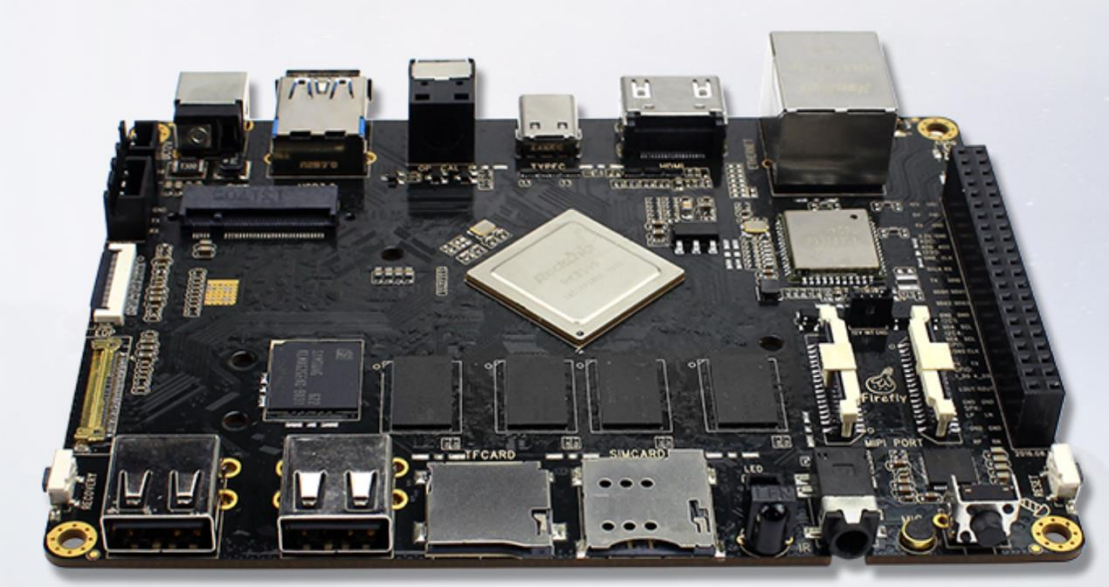
- **ACL:** Arm Computer library
- **Caffe**：A deep learning library or framework. https://github.com/BVLC/caffe and http://caffe.berkeleyvision.org/
- **CaffeOnACL:** optimized Caffe on Arm platform by ACL
- **ACL/GPU**: In the below tables, it is specialized to mean using GPU by Arm Compute Library to test. (Mali: GPU from Arm)
- **ACL/Neon**: In the below tables, it is specialized to mean using Neon by Arm Compute Library to test. (Neon: ARM coprocessor supporting SIMD)
- **OpenBLAS**: An optimized BLAS(Basic Linear Algebra Subprograms) library based on GotoBLAS2 1.13 BSD version
- **Mixed Mode**: Some layers use ACL/Neon and the other layers use OpenBLAS. For instance, “BYPASSACL = 0x14c” means using OpenBLAS layers (Softmax, RELU, FC, CONV) and ACL_NEON layers (LRN, Pooling) in neural network. (details in *User Manual 5.2*)
- **1st :** The first test loop**;** In the test applications ”classification_profiling” and “classification_ profiling_gpu” include all the process
- **2nd~11th :** the 2nd to 11th test loops, unlike the first test loop, aren’t guaranteed to use all the allocation and config processes.
- TPI : The total time for per inference

3 Environment

3.1 Hardware Platform

SoC : Rockchip RK3399

- GPU : Mali T864 (800MHz)
- CPU : Dual-core Cortex-A72 up to 2.0GHz (real frequency is 1.8GHz); Quad-core Cortex-A53 up to 1.5GHz (real frequency is 1.4GHz)



3.2 Software platform

Operating System : Ubuntu 16.04

4 Install Guide

4.1 Directory Structure

Assume the directory structure of the code in Filesystem

ACL：~/ComputeLibrary

Caffe：~/CaffeOnACL

Googletest: ~/googletest

git clone https://github.com/ARM-software/ComputeLibrary.git
git clone https://github.com/OAID/CaffeOnACL.git
git clone https://github.com/google/googletest.git

4.2 Compiled Environment Prepared

Install some dependent packages for preparation, type commands :

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install build-essential
sudo apt-get install  pkg-config automake autoconf
sudo apt-get install libprotobuf-dev libleveldb-dev
sudo apt-get install libsnappy-dev libhdf5-serial-dev protobuf-compiler
```

```
sudo apt-get install libatlas-base-dev
sudo apt-get install --no-install-recommends libboost-all-dev
sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev
sudo apt-get install libopencv-dev
sudo apt-get install libopenblas-dev
sudo apt-get install python-pip python-dev
sudo apt-get install  python-numpy python-scipy
pip install --upgrade pip
sudo apt-get install scons
sudp apt-get install cmake
sudo apt-get install git
sudo apt-get update
sudo apt-get upgrade
```

4.3 Compile ACL

```
cd ~/ComputeLibrary
mkdir build
aarch64-linux-gnu-gcc opencl-1.2-stubs/opencl_stubs.c -Iinclude -shared -o build/libOpenCL.so
scons Werror=1 -j8 debug=0 asserts=1 neon=1 opencl=1 embed_kernels=1 os=linux arch=arm64-v8a
```

4.4 Compile Caffe

```
cd ~/CaffeOnACL
 cp Makefile.config.acl Makefile.config
 make all
 make distribute
```

The default BLAS library is openBLAS, you also can change the BLAS library to atlas in Makefile.config:

```
 BLAS := atlas
```

4.5 Compile Unit test

Build the gtest libraries :

```
cd ~/googletest
cmake CMakeList.txt
make
sudo make install
```

Build Unit test :

```
cd ~/CaffeOnACL/unit_tests
make clean
make
```

4.6 Write the Applications Makefile

In the Makefile, the below content should be included:

```
HOME=
# include the configure file of CaffeOnACL
include $(HOME)/CaffeOnACL/Makefile.config
#caffe's libraries & include files
CAFFE_ROOT=$(HOME)/CaffeOnACL
CAFFE_INCS = -I$(CAFFE_ROOT)/include -I$(CAFFE_ROOT)/distribute/include/
CAFFE_LIBS = -L$(CAFFE_ROOT)/distribute/lib -lcaffe  -lglog -lgflags -lprotobuf -lboost_system -lboost_filesystem
CAFFE_RPATH =$(CAFFE_ROOT)/distribute/lib
```

4.7 How to configure the libraries path to run applications

To configure the libraries of CaffeOnACL:

```
sudo cp ~/ComputeLibrary/build/libarm_compute.so /usr/lib
sudo cp ~/CaffeOnACL/distribute/lib/libcaffe.so  /usr/lib
```

4.8 Run Tests

Run Caffenet :

```
cd  ~/CaffeOnACL/data/ilsvrc12
sudo chmod +x get_ilsvrc_aux.sh
./get_ilsvrc_aux.sh
cd ../..
chmod +x ./scripts/*
./scripts/download_model_binary.py ./models/bvlc_reference_caffenet
./distribute/bin/classification.bin models/bvlc_reference_caffenet/deploy.prototxt models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel data/ilsvrc12/imagenet_mean.binaryproto data/ilsvrc12/synset_words.txt examples/images/cat.jpg
```

output message :

```
---------- Prediction for examples/images/cat.jpg ----------
0.3094 - "n02124075 Egyptian cat"
0.1761 - "n02123159 tiger cat"
0.1221 - "n02123045 tabby, tabby cat"
0.1132 - "n02119022 red fox, Vulpes vulpes"
0.0421 - "n02085620 Chihuahua"
```

Run Unit test :

```
cd ~/CaffeOnACL/unit_tests
./test_caffe_main
```

output message:

[==========] 29 tests from 6 test cases ran. (1236 ms total) [PASSED] 29 tests.

5 Configuration Guide

The configuration guide is for debugging and performance profiling.

5.1 Enable ACL In Compile Time

- ✧ Enable ACL functions by “USE_ACL :=1” in ~/CaffeOnACL/Makefile.config
- ✧ Disable it with “USE_ACL :=0”.

Disabling ACL means Caffe using OpenBLAS not ACL.

The CaffeOnACL enable ACL by default.

5.2 Configure Options In Compile Time

Enable profiling functions by “USE_PROFILING := 1” in ~/CaffeOnACL/Makefile.config, disable it with “USE_PROFILING := 0”

Experimental functions:

- ✧ When USE_PROFILING is true, we will enable “Layer’s performance statistic” which controlled by “-DLAYER_PERF_STAT” in ~/CaffeOnACL/Makefile. You can remove “-DLAYER_PERF_STAT” to disable the feature.
- ✧ Can add “-DUSE_CONV_CACHE” into ~/CaffeOnACL/Makefile to enable the cache of convolution layer

5.3 Enable GPU Path

If you want to use GPU instead of CPU, you need call “Caffe::set_mode(Caffe::GPU)” In your code.

5.4 Configure The Bypass Of ACL Layer In Runtime

First you need set USE_ACL=1 in compiling time, refer to 5.1.

Bypass means using OpenBLAS layers. We can set “BYPASSACL” to bypass the ACL layers which you want, the control bit definitions are listed in the table below:

BYPASS_ACL_ABSVAL	0x00000001
BYPASS_ACL_BNLL	0x00000002
BYPASS_ACL_CONV	0x00000004
BYPASS_ACL_FC	0x00000008
BYPASS_ACL_LRN	0x00000010
BYPASS_ACL_POOLING	0x00000020
BYPASS_ACL_RELU	0x00000040
BYPASS_ACL_SIGMOID	0x00000080
BYPASS_ACL_SOFTMAX	0x00000100
BYPASS_ACL_TANH	0x00000200
BYPASS _ENABLE_ACL_LC	0x00000400
BYPASS _ENABLE_ACL_BN	0x00000800
BYPASS_ENABLE_ACL_CONCAT	0x00001000

For instance, we can use “export BYPASSACL=0x100” to bypass ACL Softmax layer; use “export BYPASSACL=0x124” to bypass ACL Softmax, Pooling and Convolution layers.

5.5 Configure The Log Information In Runtime

First you need set USE_ACL=1 and USE_PROFILING=1 in compiling time, refer to 5.1 and 5.2.

We can set “LOGACL” to log the performance information of the layers which you want, the control bit definitions are listed in the table below:

ENABLE_LOG_APP_TIME	0x00000001
ENABLE_LOG_ALLOCATE	0x00000002
ENABLE_LOG_RUN	0x00000004
ENABLE_LOG_CONFIG	0x00000008
ENABLE_LOG_COPY	0x00000010
ENABLE_LOG_ABSVAL	0x00000020
ENABLE_LOG_BNLL	0x00000040
ENABLE_LOG_CONV	0x00000080
ENABLE_LOG_FC	0x00000100
ENABLE_LOG_LRN	0x00000200
ENABLE_LOG_POOLING	0x00000400
ENABLE_LOG_RELU	0x00000800
ENABLE_LOG_SIGMOID	0x00001000
ENABLE_LOG_SOFTMAX	0x00002000
ENABLE_LOG_TANH	0x00004000
ENABLE_LOG_LC	0x00008000
ENABLE_LOG_BN	0x00010000
ENABLE_LOG_CONCAT	0x00020000

For instance, we can use “export LOGACL=0x100” to output the performance information of FC layer; use “export BYPASSACL=0x380” to output the performance information of LRN, FC and Convolution layers. If we copy the logs into Microsoft excel, we can sum the time with separated terms, the details of the column is:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	apptime	allocate	run	config	copy	ABSVAL	BNLL	CONV	FC	LRN	POOLING	RELU	SIGMOID	SOFTMAX	TANH

5.6 Configure The ACL Direct Convolution In Runtime

In ACL v17.06, ACL support new feature for 1x1 and 3x3 convolution which is named as direct convolution for NEON. It can be enabled by the below command:

```
export DIRECTCONV=1
```

in console, the message is shown as below

```
DIRECTCONV<1>
DIRECTCONV: 1
```

6 Test and Performance Tuning Guide

For some layers ACL has better performance and OpenBLAS has better performance. It's possible to use mixed mode for improving performance.

6.1 Use all ACL Layers

To use all ACL layers by set BYPASSACL to 0

```
export BYPASSACL=0
```

6.2 Log Performance Data

If we compile the CaffeOnACL with "USE_PROFILING := 1", we can decide which information is logged into file by setting LOGACL.

we can log all layers' information by setting LOGACL to 0x7fe1.

```
export LOGACL=0x7fe1
```

if we would like to check if "configure" take lots of time, we can set LOGACL to 0x08.

```
export LOGACL=0x08
```

if we would like to check if "memory copy" take lots of time, we can set LOGACL to 0x10.

```
export LOGACL=0x10
```

And then run your application and get the information of performance.

For instance , we use the AlexNet as the example – command line is :

```
taskset -a
10 ./distribute/bin/classification.bin ./models/bvlc_alexnet/deploy.prototxt ./models/bvlc_alex
net/bvlc_alexnet.caffemodel data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt  examples/images/cat.jpg
```

6.3 Logging Performance Data For The Original Caffe's Layers

Bypassing all ACL layers by set BYPASSACL to 0xffffffff

```
export BYPASSACL=0xffffffff
```

Logging all layers's information by setting LOGACL to 0x7fe1.

```
export LOGACL=0x7fe1
```

In this case: ENABLE_LOG_ALLOCATE,ENABLE_LOG_RUN,ENABLE_LOG_CONFIG and ENABLE_LOG_COPY are invalidate, these flags are all for ACL layers

6.4 Improve The Performance By Mixed Mode

After retrieving the performance statistic data of Caffe's layers and ACL's layers in your application, we can compare their respective performances:

	TPI	CONV	FC	LRN	Pooling	RELU	SOFTMAX
ACL_NEON	3.5360	0.2846	3.198	0.0365	0.0069	0.0086	0.0004
*Caffe_Org	1.027	0.1856	0.3922	0.435	0.0102	0.0029	0.0002

**Original Caffe uses OpenBLAS*

From the table above, we can observe that in the original Caffe's layer, CONV、FC、RELU and Softmax have faster running times than ACL's layers. Therefore, we can set BYPASSACL to 0x14c to BYPASS the 4 ACL layers, and utilize the original caffe's layers in the application. By choosing the layer set with the faster running time for each layer, we can optimize the total running time for this application.

As you can see, we obtain optimal performance in combined mode (ACL: LRN, Pooling Caffe's original Layers：Conv, FC, RELU, Softmax) as in the table below:

	TPI	CONV	FC	LRN	Pooling	RELU	SOFTMAX
*BYPASS	0.564	0.1707	0.3516	0.0321	0.0067	0.0016	0.0002

**Bypass CONV,FC,RELU and Softmax layers*

7 Use Cases

This chapter provides the performance analyzing method for specific models.

7.1 AlexNet Performance Data Logging

```
    echo "AlexNet(Neon)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0
    taskset -a
10 ./distribute/bin/classification_profiling.bin ./models/bvlc_alexnet/deploy.prototxt ./model
s/bvlc_alexnet/bvlc_alexnet.caffemodel data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt  examples/images/cat.jpg > ./log/Alexnet1_0000.log
```

```
    echo "AlexNet(OpenBlas)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0xffff
    taskset -a
10 ./distribute/bin/classification_profiling.bin ./models/bvlc_alexnet/deploy.prototxt ./model
s/bvlc_alexnet/bvlc_alexnet.caffemodel data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt  examples/images/cat.jpg > ./log/Alexnet1_ffff.log
```

```
    echo "AlexNet(Neon+OpenBlas)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0x14c
    taskset -a
10 ./distribute/bin/classification_profiling.bin ./models/bvlc_alexnet/deploy.prototxt ./model
s/bvlc_alexnet/bvlc_alexnet.caffemodel data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt  examples/images/cat.jpg > ./log/Alexnet1_014c.log
```

```
    echo "AlexNet(gpu)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0
    taskset -a 10 ./distribute/bin/
classification_profiling_gpu.bin ./models/bvlc_alexnet/deploy.prototxt ./models/bvlc_alexn
et/bvlc_alexnet.caffemodel data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt  examples/images/cat.jpg > ./log/Alexnet1_gpu.log
```

7.2 GoogleNet Performance Data Logging

```
    echo "GoogleNet(Neon)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0
    taskset -a
10 ./distribute/bin/classification_profiling.bin ./models/bvlc_googlenet/deploy.prototxt ./mo
dels/bvlc_googlenet/bvlc_googlenet.caffemodel data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt  examples/images/cat.jpg > ./log/Googlenet1_0000.log
```

```
    echo "GoogleNet(OpenBlas)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0xffff
    taskset -a
10 ./distribute/bin/classification_profiling.bin ./models/bvlc_googlenet/deploy.prototxt ./mo
dels/bvlc_googlenet/bvlc_googlenet.caffemodel data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt  examples/images/cat.jpg > ./log/Googlenet1_ffff.log
```

```
    echo "GoogleNet(Neon+OpenBlas)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0x14c
    taskset -a
10 ./distribute/bin/classification_profiling.bin ./models/bvlc_googlenet/deploy.prototxt ./mo
dels/bvlc_googlenet/bvlc_googlenet.caffemodel data/ilsvrc12/imagenet_mean.binaryproto
data/ilsvrc12/synset_words.txt  examples/images/cat.jpg > ./log/Googlenet1_014c.log
```

```
    echo "GoogleNet(gpu)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0
    taskset -a
10 ./distribute/bin/classification_profiling_gpu.bin ./models/bvlc_googlenet/deploy.prototxt
./models/bvlc_googlenet/bvlc_googlenet.caffemodel
data/ilsvrc12/imagenet_mean.binaryproto data/ilsvrc12/synset_words.txt
examples/images/cat.jpg > ./log/Googlenet1_gpu.log
```

7.3 SqueezeNet Performance Data Logging

```
    echo "SqueezeNet(Neon)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0
    taskset -a
10 ./distribute/bin/classification_profiling.bin ./models/SqueezeNet/SqueezeNet_v1.1/sque
ezenet.1.1.deploy.prototxt ./models/SqueezeNet/SqueezeNet_v1.1/squeezenet_v1.1.caff
emodel data/ilsvrc12/imagenet_mean.binaryproto data/ilsvrc12/synset_words.txt
examples/images/cat.jpg > ./log/Squeezenet1_0000.log
```

```
    echo "SqueezeNet(OpenBlas)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0xffff
    taskset -a
10 ./distribute/bin/classification_profiling.bin ./models/SqueezeNet/SqueezeNet_v1.1/sque
ezenet.1.1.deploy.prototxt ./models/SqueezeNet/SqueezeNet_v1.1/squeezenet_v1.1.caff
emodel data/ilsvrc12/imagenet_mean.binaryproto data/ilsvrc12/synset_words.txt
examples/images/cat.jpg > ./log/Squeezenet1_ffff.log
```

```
    echo "SqueezeNet(Neon+OpenBlas)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0x14c
    taskset -a
10 ./distribute/bin/classification_profiling.bin ./models/SqueezeNet/SqueezeNet_v1.1/sque
ezenet.1.1.deploy.prototxt ./models/SqueezeNet/SqueezeNet_v1.1/squeezenet_v1.1.caff
emodel data/ilsvrc12/imagenet_mean.binaryproto data/ilsvrc12/synset_words.txt
examples/images/cat.jpg > ./log/Squeezenet1_014c.log
```

```
    echo "SqueezeNet(gpu)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0
    taskset -a
10 ./distribute/bin/classification_profiling_gpu.bin ./models/SqueezeNet/SqueezeNet_v1.1/
squeezenet.1.1.deploy.prototxt ./models/SqueezeNet/SqueezeNet_v1.1/squeezenet_v1.1
.caffemodel data/ilsvrc12/imagenet_mean.binaryproto data/ilsvrc12/synset_words.txt
examples/images/cat.jpg > ./log/Squeezenet1_gpu.log
```

7.4 MobileNet Performance Data Logging

```
    echo "MobileNet(Neon)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0
```

```
    taskset -a
10 ./distribute/bin/classification_profiling.bin ./models/MobileNet/MobileNet_v1.1/MobileNe
t.1.1.deploy.prototxt ./models/MobileNet/MobileNet_v1.1/MobileNet_v1.1.caffemodel
data/ilsvrc12/imagenet_mean.binaryproto data/ilsvrc12/synset_words.txt
examples/images/cat.jpg > ./log/MobileNet1_0000.log
```

```
    echo "MobileNet(OpenBlas)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0xffff
    taskset -a
10 ./distribute/bin/classification_profiling.bin ./models/MobileNet/MobileNet_v1.1/MobileNe
t.1.1.deploy.prototxt ./models/MobileNet/MobileNet_v1.1/MobileNet_v1.1.caffemodel
data/ilsvrc12/imagenet_mean.binaryproto data/ilsvrc12/synset_words.txt
examples/images/cat.jpg > ./log/MobileNet1_ffff.log
```

```
    echo "MobileNet(Neon+OpenBlas)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0x44
    taskset -a
10 ./distribute/bin/classification_profiling.bin ./models/MobileNet/MobileNet_v1.1/MobileNe
t.1.1.deploy.prototxt ./models/MobileNet/MobileNet_v1.1/MobileNet_v1.1.caffemodel
data/ilsvrc12/imagenet_mean.binaryproto data/ilsvrc12/synset_words.txt
examples/images/cat.jpg > ./log/MobileNet1_44.log
```

```
    echo "MobileNet(gpu)"
    export OPENBLAS_NUM_THREADS=1
    export BYPASSACL=0
    taskset -a
10 ./distribute/bin/classification_profiling_gpu.bin ./models/MobileNet/MobileNet_v1.1/Mob
ileNet.1.1.deploy.prototxt ./models/MobileNet/MobileNet_v1.1/MobileNet_v1.1.caffemodel
data/ilsvrc12/imagenet_mean.binaryproto data/ilsvrc12/synset_words.txt
examples/images/cat.jpg > ./log/MobileNet1_gpu.log
```