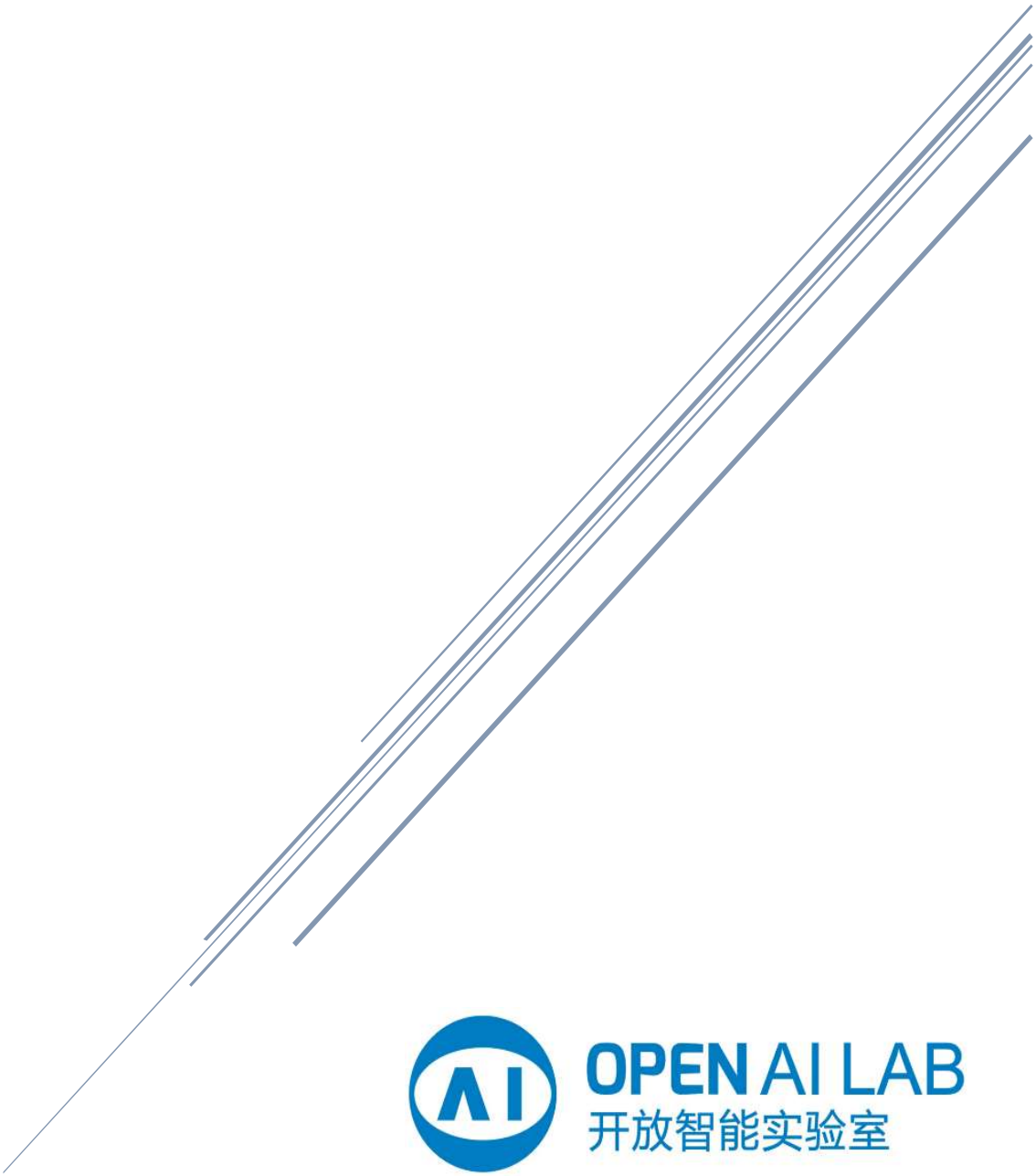
USER MANUAL

OF MXNETONACL



OPEN AI LAB

开放智能实验室

Index

1 Purpose

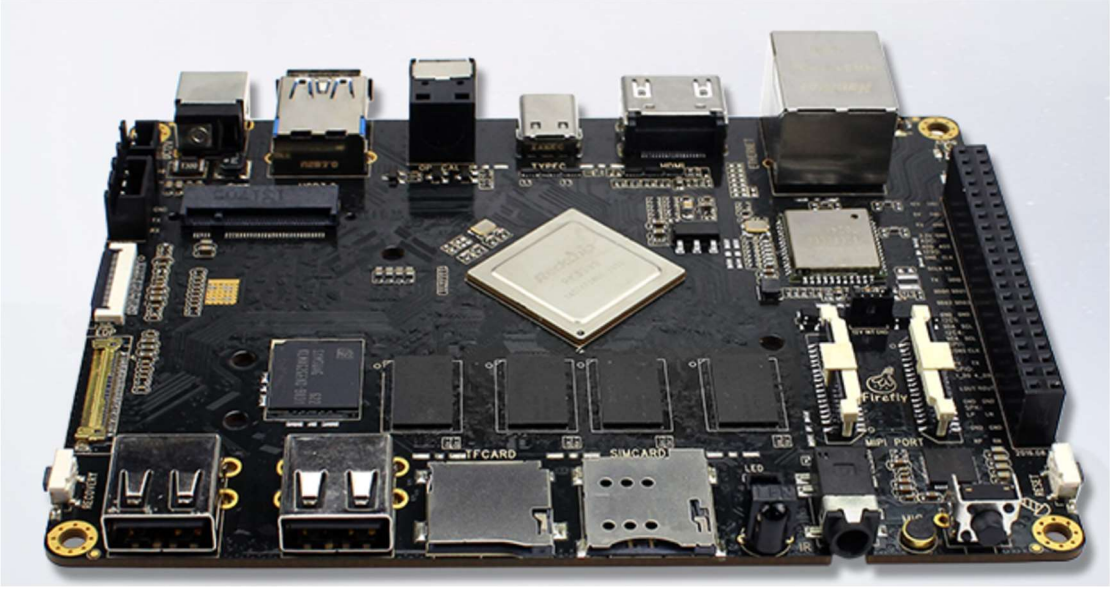
This guide help user utilize the code of MXNetOnACL (MXNet+ACL) to improve the performance of their applications based on the MXNet framework.

2 Terminology

- **ACL:** Arm Computer library
- MXNet：A deep learning library or framework. http://mxnet.io and https://github.com/apache/incubator-mxnet
- **MXNetOnACL:** optimized MXNet 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

Board：http://wiki.t-firefly.com/index.php/Firefly-RK3399
Chip: Rockchip RK3399
System: Ubuntu 16.04
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)



4 Install Guide

4.1 Directory Structure

Assume the directory structure of the code is:

ACL：~/ComputeLibrary

git clone https://github.com/ARM-software/ComputeLibrary.git

MXNet：~/MXNetOnACL

git clone --recursive https://github.com/OAID/MXNetOnACL.git

4.2 Compiled Environment Prepared

```
sudo apt-get update -y
sudo apt-get upgrade -y
sudo apt-get install build-essential git libatlas-base-dev libopencv-dev -y
sudo apt-get install python-pip python-dev -y
sudo apt-get install -y python-numpy python-scipy
sudo pip install --upgrade pip
sudo apt-get install scons –y
sudo  apt-get install git –y
```

4.3 Compile ACL

```
cd ~/ComputeLibrary
  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 MXNet

```
cd ~/MXNetOnACL
cp config.mk.acl config.mk
make
```

4.5 How To Configure The Libraries Path To Run Applications

To configure the libraries of MXNetOnACL:

```
sudo cp ~/ComputeLibrary/build/libarm_compute.so /usr/lib
sudo cp ~/MXNetOnACL/lib/libmxnet.so  /usr/lib
```

4.6 Run MXNet Classification

```
cd ~/MXNetOnACL
example/image-classification/predict-cpp/image-classification-predict
models/caffenet-symbol.json models/caffenet-0000.params mean_224.nd
synset_words.txt cat.jpg
```

The output message:
Best Result: [tabby, tabby cat] id = 281, accuracy = 0.23338266

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 ~/MXNetOnACL/config.mk, disable it

with “USE_ACL :=0”

Disabling ACL means MXNet using OpenBLAS not ACL.

The MXNetOnACL enable ACL by default.

5.2 Configure Options In Compile Time

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

5.3 Enable GPU Path

If you want to use GPU instead of CPU, you need call

MXNet::set_mode(MXNet::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_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_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 MXNetOnACL 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 example/image-classification/predict-cpp/image-classification-predict models/bvlc_alexnet/caffenet-symbol.json models/bvlc_alexnet/caffenet-0000.params mean_224.nd synset_words.txt cat.jpg
```

5.3 Logging Performance Data For The Original MXNet'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 MXNet'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
*MXNet_Org	1.027	0.1856	0.3922	0.435	0.0102	0.0029	0.0002

*Original MXNet uses OpenBLAS

From the table above, we can observe that in the original MXNet'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 MXNet'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 MXNet'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 example/image-classification/predict-cpp/image-classification-
predict_profiling.bin ./models/bvlc_alexnet/deploy.json ./models/bvlc_alexnet/bvl
c_alexnet.params mean_224.nd synset_words.txt
cat.jpg > ./log/Alexnet1_0000.log
```

```
echo "AlexNet(OpenBlas)"
export OPENBLAS_NUM_THREADS=1
export BYPASSACL=0xffff
taskset -a 10 example/image-classification/predict-cpp/image-classification-
predict_profiling.bin ./models/bvlc_alexnet/deploy.json ./models/bvlc_alexnet/bvl
c_alexnet.params mean_224.nd synset_words.txt cat.jpg > ./log/Alexnet1_ffff.log

echo "AlexNet(Neon+OpenBlas)"
export OPENBLAS_NUM_THREADS=1
export BYPASSACL=0x14c
taskset -a 10 example/image-classification/predict-cpp/image-classification-
predict_profiling.bin ./models/bvlc_alexnet/deploy.json ./models/bvlc_alexnet/bvl
c_alexnet.params mean_224.nd synset_words.txt
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.json ./models/bvlc_
alexnet/bvlc_alexnet.params mean_224.nd synset_words.txt
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 example/image-classification/predict-cpp/image-classification-
predict_profiling.bin ./models/bvlc_googlenet/deploy.json ./models/bvlc_googlen
et/bvlc_googlenet.params mean_224.nd synset_words.txt
cat.jpg > ./log/Googlenet1_0000.log

echo "GoogleNet(OpenBlas)"
export OPENBLAS_NUM_THREADS=1
export BYPASSACL=0xffff
taskset -a 10 example/image-classification/predict-cpp/image-classification-
predict_profiling.bin ./models/bvlc_googlenet/deploy.json ./models/bvlc_googlen
et/bvlc_googlenet.params mean_224.nd synset_words.txt
cat.jpg > ./log/Googlenet1_ffff.log

echo "GoogleNet(Neon+OpenBlas)"
export OPENBLAS_NUM_THREADS=1
export BYPASSACL=0x14c
```

```
taskset -a 10 example/image-classification/predict-cpp/image-classification-
predict_profiling.bin ./models/bvlc_googlenet/deploy.json ./models/bvlc_googlen
et/bvlc_googlenet.params mean_224.nd synset_words.txt
cat.jpg > ./log/Googlenet1_014c.log

echo "GoogleNet(gpu)"
export OPENBLAS_NUM_THREADS=1
export BYPASSACL=0
taskset -a 10 example/image-classification/predict-cpp/image-classification-
predict_profiling_gpu.bin ./models/bvlc_googlenet/deploy.json ./models/bvlc_goo
glenet/bvlc_googlenet.params mean_224.nd synset_words.txt
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 example/image-classification/predict-cpp/image-classification-
predict_profiling.bin ./models/SqueezeNet/SqueezeNet_v1.1/squeezenet.1.1.dep
loy.json ./models/SqueezeNet/SqueezeNet_v1.1/squeezenet_v1.1.params
mean_224.nd synset_words.txt cat.jpg > ./log/Squeezenet1_0000.log

echo "SqueezeNet(OpenBlas)"
export OPENBLAS_NUM_THREADS=1
export BYPASSACL=0xffff
taskset -a 10 example/image-classification/predict-cpp/image-classification-
predict_profiling.bin ./models/SqueezeNet/SqueezeNet_v1.1/squeezenet.1.1.dep
loy.json ./models/SqueezeNet/SqueezeNet_v1.1/squeezenet_v1.1.params
mean_224.nd synset_words.txt cat.jpg > ./log/Squeezenet1_ffff.log
echo "SqueezeNet(Neon+OpenBlas)"
export OPENBLAS_NUM_THREADS=1
export BYPASSACL=0x14c
taskset -a 10 example/image-classification/predict-cpp/image-classification-
predict_profiling.bin ./models/SqueezeNet/SqueezeNet_v1.1/squeezenet.1.1.dep
loy.json ./models/SqueezeNet/SqueezeNet_v1.1/squeezenet_v1.1.params
mean_224.nd synset_words.txt cat.jpg > ./log/Squeezenet1_014c.log

echo "SqueezeNet(gpu)"
export OPENBLAS_NUM_THREADS=1
export BYPASSACL=0
taskset -a 10 example/image-classification/predict-cpp/image-classification-
predict_profiling_gpu.bin ./models/SqueezeNet/SqueezeNet_v1.1/squeezenet.1.
```

1.deploy.json ./models/SqueezeNet/SqueezeNet_v1.1/squeezenet_v1.1.params mean_224.nd synset_words.txt 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 example/image-classification/predict-cpp/image-classification-predict_profiling.bin ./models/MobileNet/MobileNet_v1.1/MobileNet.1.1.deploy.json ./models/MobileNet/MobileNet_v1.1/MobileNet_v1.1.params mean_224.nd synset_words.txt cat.jpg > ./log/MobileNet1_0000.log

echo "MobileNet(OpenBlas)"
export OPENBLAS_NUM_THREADS=1
export BYPASSACL=0xffff
taskset -a 10 example/image-classification/predict-cpp/image-classification-predict_profiling.bin ./models/MobileNet/MobileNet_v1.1/MobileNet.1.1.deploy.json ./models/MobileNet/MobileNet_v1.1/MobileNet_v1.1.params mean_224.nd synset_words.txt cat.jpg > ./log/MobileNet1_ffff.log

echo "MobileNet(Neon+OpenBlas)"
export OPENBLAS_NUM_THREADS=1
export BYPASSACL=0x44
taskset -a 10 example/image-classification/predict-cpp/image-classification-predict_profiling.bin ./models/MobileNet/MobileNet_v1.1/MobileNet.1.1.deploy.json ./models/MobileNet/MobileNet_v1.1/MobileNet_v1.1.params mean_224.nd synset_words.txt cat.jpg > ./log/MobileNet1_44.log

echo "MobileNet(gpu)"
export OPENBLAS_NUM_THREADS=1
export BYPASSACL=0
taskset -a 10 example/image-classification/predict-cpp/image-classification-predict_profiling_gpu.bin ./models/MobileNet/MobileNet_v1.1/MobileNet.1.1.deploy.json ./models/MobileNet/MobileNet_v1.1/MobileNet_v1.1.params mean_224.nd synset_words.txt cat.jpg > ./log/MobileNet1_gpu.log
```