Compute Library

 19.02

# Introduction

The Computer Vision and Machine Learning library is a set of functions optimised for both ARM CPUs and GPUs using SIMD technologies.Several builds of the library are available using various configurations:

* OS: Linux, Android or bare metal.
* Architecture: armv7a (32bit) or arm64-v8a (64bit)
* Technology: NEON / OpenCL / GLES\_COMPUTE / NEON and OpenCL and GLES\_COMPUTE
* Debug / Asserts / Release: Use a build with asserts enabled to debug your application and enable extra validation. Once you are sure your application works as expected you can switch to a release build of the library for maximum performance.

## Pre-built binaries

## File organisation

This archive contains:

* The [**arm\_compute**](https://arm-software.github.io/ComputeLibrary/latest/namespacearm__compute.xhtml) header and source files
* The latest Khronos OpenCL 1.2 C headers from the [Khronos OpenCL registry](https://www.khronos.org/registry/cl/)
* The latest Khronos cl2.hpp from the [Khronos OpenCL registry](https://www.khronos.org/registry/cl/) (API version 2.1 when this document was written)
* The latest Khronos OpenGL ES 3.1 C headers from the [Khronos OpenGL ES registry](https://www.khronos.org/registry/gles/)
* The latest Khronos EGL 1.5 C headers from the [Khronos EGL registry](https://www.khronos.org/registry/gles/)
* The sources for a stub version of libOpenCL.so, libGLESv1\_CM.so, libGLESv2.so and libEGL.so to help you build your application.
* An examples folder containing a few examples to compile and link against the library.
* A [**utils**](https://arm-software.github.io/ComputeLibrary/latest/dir_cbdb8362360e11eafe2fa3bc74cf0ffd.xhtml) folder containing headers with some boiler plate code used by the examples.
* This documentation.

To cross compile a NEON example for Linux 64bit:

aarch64-linux-gnu-g++ examples/neon\_convolution.cpp utils/Utils.cpp -I. -Iinclude -std=c++11 -L. -larm\_compute -larm\_compute\_core -o neon\_convolution

(notice the only difference with the 32 bit command is that we don't need the -mfpu option and the compiler's name is different)

**Note**

If compiling using static libraries, this order must be followed when linking: arm\_compute\_graph\_static, [**arm\_compute**](https://arm-software.github.io/ComputeLibrary/latest/namespacearm__compute.xhtml), arm\_compute\_core

These two commands assume libarm\_compute.so is available in your library path, if not add the path to it using -L

To run the built executable simply run:

LD\_LIBRARY\_PATH=build ./neon\_convolution

or

LD\_LIBRARY\_PATH=build ./cl\_convolution

参考资料：

2.1 使用 /etc/ld.so.conf 配置文件

将库文件所在的路径加入到 /etc/ld.so.conf 尾部，并使之生效：

$ sudo echo '/opt/biosoft/hdf5-1.8.15-patch1/lib/' >> /etc/ld.so.conf

libhdf5.so 在路径 /opt/biosoft/hdf5-1.8.15-patch1/lib/ 下，将该路径加添加到配置文件中

$ sudo ldconfig

运行该命令，重新载入 /ext/ld.so.conf 中的路径，使修改生效。

2.2 修改环境变量

$ export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/opt/biosoft/hdf5-1.8.15-patch1/lib/

修改环境变量 LD\_LIBRARY\_PATH，加入库文件所在路径。使用 export 命令使修改生效。

$ echo 'export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/opt/biosoft/hdf5-1.8.15-patch1/lib/' >> ~/.bashrc

$ source ~/.bashrc

将上述 export 命令加入到配置文件 ~/.bashrc，使之永久生效。

$ export LIBRARY\_PATH=/opt/biosoft/hdf5-1.8.15-patch1/lib/:$LIBRARY\_PATH

若修改变量 LD\_LIBRARY\_PATH 不奏效，则修改变量 LIBRARY\_PATH 。

|  |
| --- |
| $ echo 'export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/home/djiango/NEON/ComputeLibrary-master/lib/' >> ~/.bashrc  $ source ~/.bashrc |

# ACL 整体使用评价

ARM Compute Library是ARM公司刚发布不久的开源工程，旨在为图像/视频/多媒体/计算机视觉等领域的开发者提供arm平台的硬件加速库。这个库中分别用OpenCL与NEON的方式实现了一些上述领域的基本算法，OpenCL主要是arm的Mali GPU加速，NEON是针对arm的A系列CPU。

我最近研究了一下它的源码，主要看了针对CNN的卷积运算需要用到的convolution过程。当然，其他的基本算法也都是同样的流程。工程中是把图像按照列的方式分割成子块，然后分别启动几个线程去处理这些子块。对于convolution来说，NEON方式实现了两种方法，一种是GEMM的方法，把输入图像先im2col，然后interleave操作，把weight进行transposed操作，之后进行矩阵乘法，之所以有interleave与transposed两步是为了矩阵乘法时NEON指令集load数据的连贯性与平顺性，并且不需要重复load，最大限度的发挥了neon指令集的能力。还一种方法是标准的卷积运算。当然其中也是运用了NEON的intrinsic函数调用方式。OpenCL调用GPU加速的方式我还没有细看，不过大体上看来主要流程与NEON的方式类似，也是按照线程数分割图像，然后并行处理子块。其中也是有shape，window，iterator的概念。只是真正的计算中与NEON的指令集不一样。

这个lib发布之后，开发者可以不用关心arm的cpu与gpu怎样通过NEON或OpenCL来实现硬件的加速，直接调用这个库中的接口就可以，对于开发计算机视觉类的应用但是不太了解硬件加速编程的工程师来说十分有利。

---------------------

作者：dark\_knight\_brcwn

来源：CSDN

原文：https://blog.csdn.net/u010957054/article/details/73800217

版权声明：本文为博主原创文章，转载请附上博文链接！

速度测试

ACL测试程序可以参考tvm-mali/acl\_test.cc，在实际测试过程中发现的问题是加载模型比不加载模型的速度慢，在GPU上差异尤其明显。具体原因未知。

#总结

ARM Compute Library整合了ARM自家的CPU和GPU资源，既能够实现常用的图像处理操作，也能继续深度学习推理。考虑到OpenCV亦加入了dnn module，二者存在很大程度上的重合。

再说ARM Compute Library的缺点：

库很大很重，源码>150MB。

用作图像处理的话不支持图像读取转换等操作。

用于深度学习应用时模型加载非常不方便且未提供完整模型转换工具。

库构建使用SCons而不是更为常见且与Android NDK兼容更好的CMake。

底层代码优化度不足，CPU端速度逊于Tencent/ncnn