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 |