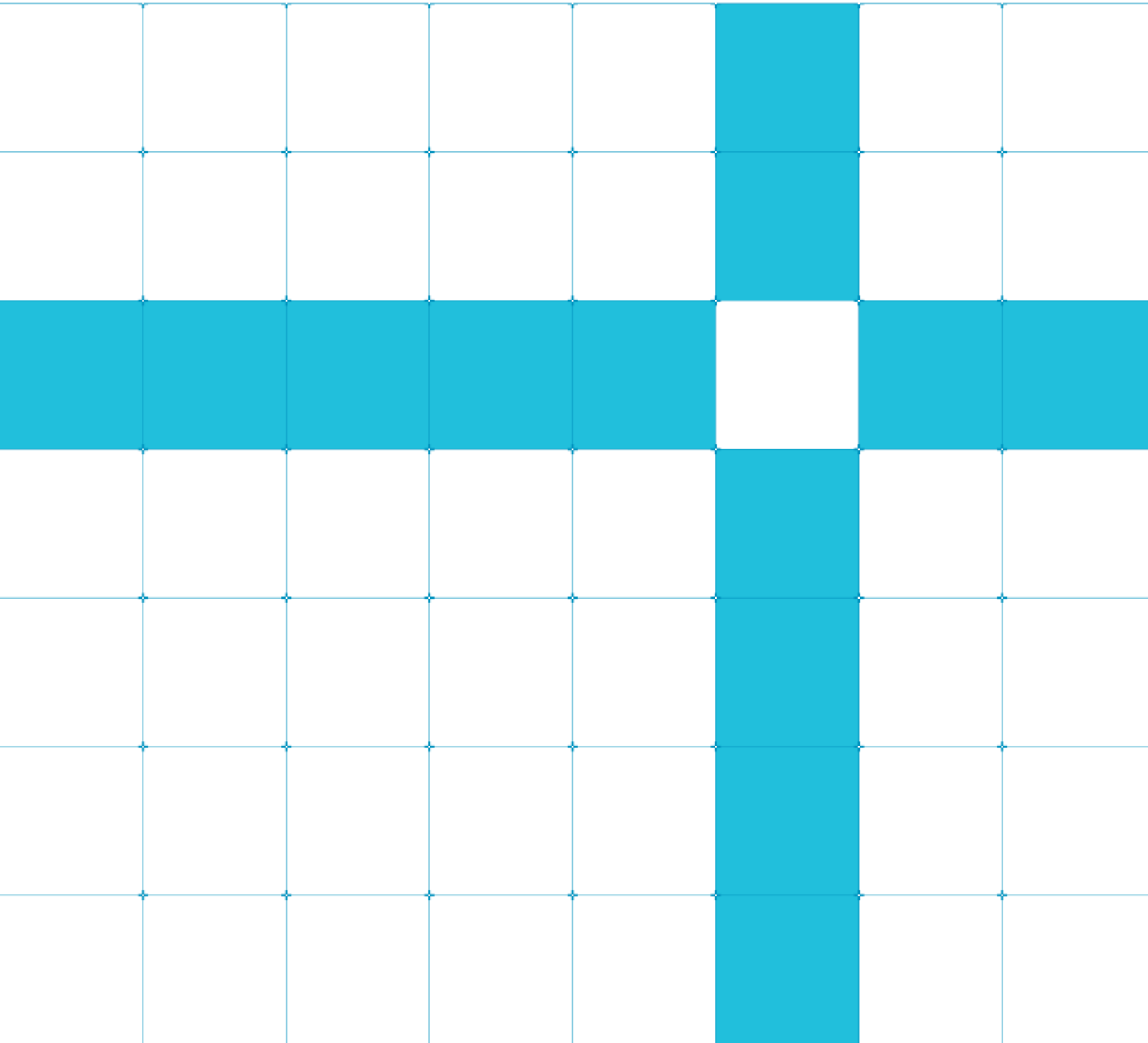
Configure the Arm NN SDK for TensorFlow

Version 1.7



Configure the Arm NN SDK build environment for TensorFlow

Release Information

Document History

Version	Date	Confidentiality	Change
1.0	13 April 2018	Non-Confidential	First release.
1.1	03 August 2018	Non-Confidential	Minor enhancements.
1.2	04 September 2018	Non-Confidential	Minor enhancements.
1.3	30 October 2018	Non-Confidential	Minor enhancements.
1.4	11 March 2019	Non-Confidential	Minor enhancements.
1.5	07 June 2019	Non-Confidential	Minor enhancements to the Before you begin section.
1.6	18 November 2019	Non-Confidential	Adds the command for cloning the Tensorflow source code.
1.7	21 Februrary 2020	Non-Confidential	Adds instructions on including standalone dynamic backend tests. Also, updates the git checkout command in the Setting up the build dependencies section.

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

http://www.arm.com

Contents

1 Overview

Before you run code using Arm NN, you must first set up and configure the build environment and the other required tools on your board or platform. You must then test that your build has completed successfully. This guide walks you through that process.

The configuration process is:

1. Download the Arm NN SDK and **Compute Library** files from GitHub.
2. Set up the build dependencies.
3. Build the environment.
4. Test that the build is successful.

2 Before you begin

Your platform or board must have:

- An **ARMv7-A** or **ARMv8-A CPU**, and optionally an **Arm Mali GPU** using the OpenCL driver.
- At least 4GB of RAM.
- At least 1GB of free storage space.

Before you configure and build your environment, you must install the following tools on your platform or board:

- A Linux distribution.
- Git.
- SCons. Arm has tested SCons 2.4.1 (Ubuntu) and 2.5.1 (Debian). Other versions might work.
- CMake. Arm has tested CMake 3.5.1 (Ubuntu) and 3.7.2 (Debian). Other versions might work.

These instructions assume that you use Ubuntu 16.04 or Debian 9.0, but should work on most Linux distributions. Arm tests the Arm NN SDK on Ubuntu and Debian.

3 Dowloading the Arm NN SDK and Compute Library files from GitHub

To download the Arm NN SDK and the Compute Library, you must clone them from their GitHub repositories using Git.

To clone the Arm NN SDK files, open a terminal or bash screen on your platform or board and enter: `git clone https://github.com/Arm-software/armnn.`

To clone the **Compute Library** files, open a terminal or bash screen on your platform or board and enter: `git clone https://github.com/ARM-software/ComputeLibrary.`

Make a note of where you store these Git repositories, because you will need these locations when you build the environment.

4 Setting up the build dependencies

The Arm NN SDK relies on some software, which you must set up and install.

To set up the build dependencies:

1. Download and build **Boost**. Arm has tested version 1.64. Some other versions might work. For instructions to do this, see the **Boost getting started guide**. When you build Boost, include the following flags:
   `link=static cxxflags=-fPIC --with-filesystem --with-test --with-log --with-program_options --prefix=path/to/installation/prefix`

2. Clone and build protobuf. Arm has tested version 3.5.0. Some other versions might work. To clone version 3.5.0, enter:
   `git clone –b v3.5.0 https://github.com/google/protobuf.git`

   Build protobuf using the C++ installation instructions that you can find on the **Protobuf GitHub**.

   Make a note of the location of the installation, you will need it during the TensorFlow build process.

2. Clone the TensorFlow source code from GitHub with this command:

```
git clone https://github.com/tensorflow/tensorflow.git
cd tensorflow/
git checkout 590d6eef7e91a6a7392c8ffffb7b58f2e0c8bc6b
```

5 Building the environment

To use the Arm NN SDK, you must build the environment on your platform or board.

To build the environment:

1. Compile the Compute Library using SCons. To do this, change your directory to the Compute Library Git repository on your machine, and enter:
   `scons arch=arm64-v8a extra_cxx_flags="-fPIC" benchmark_tests=0 validation_tests=0` for ARMv8-A or
   `scons extra_cxx_flags="-fPIC" benchmark_tests=0 validation_tests=0` for ARMv7-A.

   If you want to enable benchmark tests, set `benchmark_tests` to 1. If you want to enable validation tests, set `validation_tests` to 1.

   You can enable support for **NEON** on the CPU, and support for **OpenCL** on an Arm Mali GPU if you have one.

   If you want to support OpenCL for your Arm Mali GPU, add these arguments to the SCons command:

   `opencl=1 embed_kernels=1`

   If you want to support NEON, add this argument to your SCons command:

   `neon=1`

2. Change the directory to `<the installation root>`.
3. Generate C++ sources and headers using the protobuf compiler. To do this, enter:

```
pushd tensorflow
${ARMNN_DIR}/armnn/scripts/generate_tensorflow_protobuf.sh <directory where the generated
files will be placed> <the protobuf install directory>
popd
```

4. Configure the Arm NN SDK build using CMake. To do this, change your directory to the Arm NN directory and enter:

```
mkdir build
cd build
cmake .. -DARMCOMPUTE_ROOT=<the location of your Compute Library source files directory> -
DARMCOMPUTE_BUILD_DIR=<the location of your Compute Library source files directory>/build -
DBOOST_ROOT=<directory used with the prefix flag> -DTF_GENERATED_SOURCES=<the output
directory from step 3> -DBUILD_TF_PARSER=1 -DPROTOBUF_ROOT=<the location of your protobuf
install directory>
```

   If you are supporting NEON, add this argument to the CMake command:

```
-DARMCOMPUTENEON=1
```

   If you are supporting OpenCL, add this argument to the CMake command:

```
-DARMCOMPUTECL=1
```

   If you want to include Arm NN reference support, add this argument to the CMake command:

```
-DARMNNREF=1
```

   If you want to include standalone sample dynamic backend tests, add the argument to enable the tests and the dynamic backend path to the CMake command:

```
-DSAMPLE_DYNAMIC_BACKEND=1 -DDYNAMIC_BACKEND_PATHS=<the location of the sample dynamic
backend>
```

Also, after building Arm NN, build the standalone sample dynamic backend using the guide in the following path: `$BASEDIR/armnn/src/dynamic/README.md#standalone-dynamic-backend-build.`

5. Enter `make.`

6 Testing the build

To check that your build of the Arm NN SDK is working correctly, you can run the Unit Tests. To do this, change to the Arm NN build directory and enter `./UnitTests`.

If the tests are successful, the output from the tests ends with `*** No errors detected`.

If some of the tests are unsuccessful, go back through the steps and check that all the commands have been entered correctly.

Now that you have built your environment, you are ready to start programming.

7 Other resources

You might find the following resource helpful:

- **Arm NN SDK TensorFlow supported operators**.