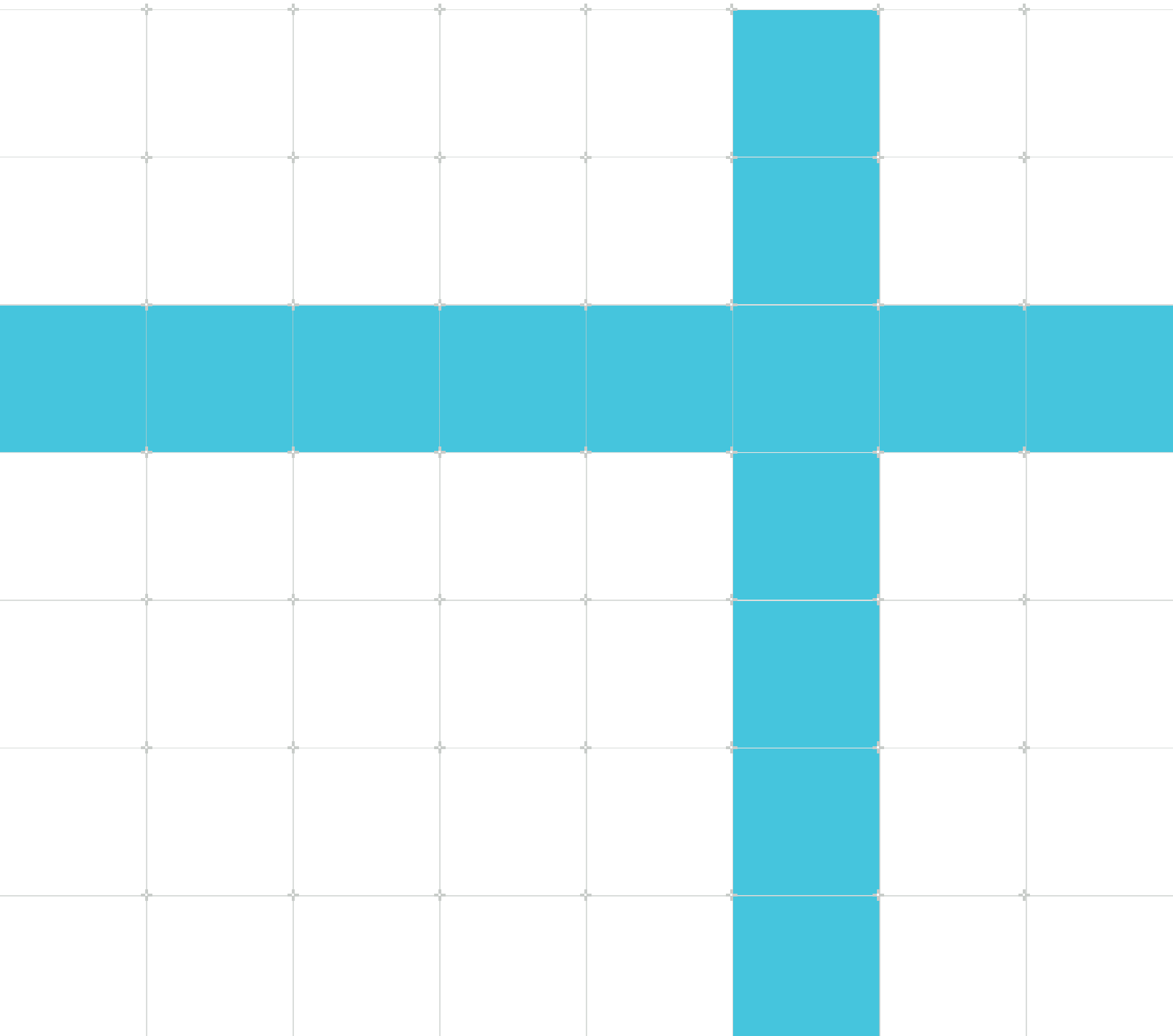
# Cross-compiling Arm NN for the Raspberry Pi

Version 22.08

# Tutorial



**Issue 01**
102649_2208_01_en



# Cross-compiling Arm NN for the Raspberry Pi
## Tutorial



## Release information

### Document history

| Issue | Date | Confidentiality | Change |
|---|---|---|---|
| 0100-01 | 25 October 2018 | Non-Confidential | First release for version 1.01 |
| 0100-02 | 10 June 2019 | Non-Confidential | First release for version 1.02 |
| 0100-03 | 15 July 2019 | Non-Confidential | First release for version 1.03 |
| 0100-04 | 13 August 2019 | Non-Confidential | First release for version 1.04 |
| 0100-05 | 3 September 2019 | Non-Confidential | First release for version 1.05 |
| 0100-06 | 11 November 2019 | Non-Confidential | First release for version 1.06 |
| 0100-07 | 27 November 2019 | Non-Confidential | First release for version 1.07 |
| 0100-08 | 5 December 2019 | Non-Confidential | First release for version 1.08 |
| 0100-09 | 4 May 2020 | Non-Confidential | First release for version 1.09 |
| 0100-10 | 2 June 2020 | Non-Confidential | First release for version 1.10 |
| 0100-11 | 10 August 2020 | Non-Confidential | First release for version 1.11 |
| 0100-12 | 2 September 2020 | Non-Confidential | First release for version 1.12 |
| 0100-13 | 28 January 2021 | Non-Confidential | First release for version 1.13 |
| 0100-14 | 16 March 2021 | Non-Confidential | First release for version 1.14 |
| 0100-15 | 20 May 2021 | Non-Confidential | First release for version 1.15 |
| 2108-01 | 16 August 2021 | Non-Confidential | First release for version 21.08 |
| 2108-02 | 29 September 2021 | Non-Confidential | Second release for version 21.08 |
| 2108-03 | 21 October 2021 | Non-Confidential | Third release for version 21.08 |
| 2202-01 | 16 February 2022 | Non-Confidential | First release for version 22.02 |
| 2202-02 | 12 April 2022 | Non-Confidential | Second release for version 22.02 |

| Issue | Date | Confidentiality | Change |
|---|---|---|---|
| 2205-01 | 27 July 2022 | Non-Confidential | First release for version 22.05 |
| 2208-01 | 30 August 2022 | Non-Confidential | First release for version 22.08 |

## Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

## 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.

## Feedback

Arm® welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

## Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

# Contents

# 1. Introduction

## 1.1 Conventions

The following subsections describe conventions used in Arm documents.

### Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm® Glossary for more information: developer.arm.com/glossary.

### Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

| Convention | Use |
|---|---|
| *italic* | Citations. |
| **bold** | Interface elements, such as menu names.<br><br>Terms in descriptive lists, where appropriate. |
| `monospace` | Text that you can enter at the keyboard, such as commands, file and program names, and source code. |
| `monospace underline` | A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name. |
| `<and>` | Encloses replaceable terms for assembler syntax where they appear in code or code fragments.<br><br>For example:<br><br>`MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>` |
| SMALL CAPITALS | Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**. |
| Caution | Recommendations. Not following these recommendations might lead to system failure or damage. |
| Warning | Requirements for the system. Not following these requirements might result in system failure or damage. |
| Danger | Requirements for the system. Not following these requirements will result in system failure or damage. |
| Note | An important piece of information that needs your attention. |

| Convention | Use |
|---|---|
| Tip | A useful tip that might make it easier, better or faster to perform a task. |
| Remember | A reminder of something important that relates to the information you are reading. |

## 1.2 Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Note

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

Adobe PDF reader products can be downloaded at http://www.adobe.com

## 1.3 Other information

See the Arm website for other relevant information.

- Arm® Developer.
- Arm® Documentation.
- Technical Support.
- Arm® Glossary.

# 2. Overview

The steps to build Arm NN in this guide are being replaced by the Arm NN Build Tool and will be removed soon.


Use the Arm NN Build Tool as a user-friendly way to build Arm NN and its dependencies.

This guide covers what we must do to cross-compile Arm NN using an x86_64 system to target a Raspberry Pi. Cross-compiling Arm NN allows us to work around the limited memory of the Raspberry Pi.

Read the guide to find out how to build the Compute Library, Protobuf, and Arm NN core libraries that you need for compilation. When we finish, we will be able to compile and run programs that use Arm NN on our Raspberry Pis.


The 21.08 release of Arm NN removes Boost library dependency. You are not required to install boost library with 21.08 and newer releases.

Arm estimates that you will take 90-120 minutes to complete this guide.

# 3. Before you begin

The steps to build Arm NN in this guide are being replaced by the Arm NN Build Tool and will be removed soon.


Use the Arm NN Build Tool as a user-friendly way to build Arm NN and its dependencies.

This guide assumes:

- You have a Raspberry Pi board. Arm has tested these instructions on a:
  - Raspberry Pi 3 Model B+ that runs Raspbian 8.
  - Raspberry Pi 4 Model B that runs Raspbian 9.
- You compile on a Linux virtual machine. Arm has tested these instructions on an Ubuntu 20.04 and 20.10 virtual machine that runs on MacOS and Windows 10.

You must install the following software tools on your virtual machine:

Git A version control system software developers use for source code management.

Scons An open-source software construction tool.

GNU C and C++ compilers for the armhf architecture The Raspberry Pi uses the armhf Arm architecture.


For the instructions in this guide to work, the cross-compiler version on the host machine must match the compiler version on your Raspberry Pi.

Curl A tool for transferring data to or from a Linux or Unix-like server.

GNU Autoconf A tool for producing configure scripts for building installing, and packaging software.

GNU libtool A generic library support script.

CMake An open-source and cross-platform family of tools for building, testing, and packaging software.

To install the tools you require, open a command window and enter the following commands:

```
sudo apt-get install git
sudo apt-get install scons
sudo apt-get install gcc-arm-linux-gnueabihf
```

```
sudo apt-get install g++-arm-linux-gnueabihf
sudo apt-get install curl
sudo apt-get install autoconf
sudo apt-get install libtool
sudo apt-get install cmake
sudo apt-get install g++
sudo apt-get install wget
sudo apt-get install unzip
sudo apt-get install xxd
```

# 4. Downloading the repositories and bundles

Create a directory on your virtual machine to build your Arm NN distribution for your Raspberry Pi.

To create a directory and build your distribution:

1. Enter the following commands:

```
mkdir armnn-pi && cd armnn-pi
export BASEDIR=`pwd`
```

2. Download the Compute Library, Protobuf, TensorFlow, ONNX, FlatBuffer, and Arm NN git repositories and source bundles. To download the repositories and bundles, enter the following commands:

```
git clone https://github.com/Arm-software/ComputeLibrary.git
git clone https://github.com/Arm-software/armnn
git clone -b v3.12.0 https://github.com/google/protobuf.git
git clone https://github.com/tensorflow/tensorflow.git
cd tensorflow/
git checkout a4dfb8d1a71385bd6d122e4f27f86dcebb96712d
cd $BASEDIR
git clone https://github.com/onnx/onnx.git
cd onnx
git fetch https://github.com/onnx/onnx.git
 553df22c67bee5f0fe6599cff60f1afc6748c635 && git checkout FETCH_HEAD
cd $BASEDIR

wget -O flatbuffers-1.12.0.tar.gz https://github.com/google/flatbuffers/archive/
v1.12.0.tar.gz
tar xf flatbuffers-1.12.0.tar.gz
```

# 5. Building the Compute Library

You must build the Compute Library on your virtual machine.

To build the library, enter the following command:

```
cd $BASEDIR/ComputeLibrary
scons extra_cxx_flags="-fPIC" Werror=0 debug=0 asserts=0 neon=1 opencl=0 os=linux
 arch=armv7a examples=1
```

Arm estimates that your virtual machine will take approximately 15-20 minutes to execute this command.

# 6. Building the Google Protobuf library

You use two versions of the Google Protobuf library. One version of the library runs on your virtual machine and the other runs on your Raspberry Pi.

## 6.1 Building the Google Protobuf library for your virtual machine

You must build the Google Protobuf library for your virtual machine.

1. Enter the following commands:

```
cd $BASEDIR/protobuf
git submodule update --init --recursive
./autogen.sh
./configure --prefix=$BASEDIR/protobuf-host
make
```

2. To install the Google Protobuf library on your virtual machine, enter the following commands:

```
make install
make clean
```

   Arm estimates that your virtual machine will take approximately 15 minutes to execute these commands.

## 6.2 Building the Google Protobuf library for your Raspberry Pi

You must build the Google Protobuf library for your Raspberry Pi.

1. Enter the following commands:

```
./configure --prefix=$BASEDIR/protobuf-arm --host=arm-linux CC=arm-linux-
gnueabihf-gcc CXX=arm-linux-gnueabihf-g++ --with-protoc=$BASEDIR/protobuf-host/
bin/protoc
make
```

2. To install the Google Protobuf library on your Raspberry pi, enter the following command:

```
make install
```

Arm estimates that your virtual machine will take approximately 15 minutes to execute these commands.

# 7. Building ONNX

To build ONNX, enter the following commands:

```
cd $BASEDIR/onnx
export LD_LIBRARY_PATH=$BASEDIR/protobuf-host/lib:$LD_LIBRARY_PATH
$BASEDIR/protobuf-host/bin/protoc onnx/onnx.proto --proto_path=. --proto_path=
$BASEDIR/protobuf-host/include --cpp_out $BASEDIR/onnx
```

# 8. Building FlatBuffers

To build Flatbuffers, enter the following commands:

```
cd $BASEDIR/flatbuffers-1.12.0
rm -f CMakeCache.txt
rm -rf build
mkdir build
cd build
CXXFLAGS="-fPIC" cmake .. -DFLATBUFFERS_BUILD_FLATC=1 \
-DCMAKE_INSTALL_PREFIX:PATH=$BASEDIR/flatbuffers
make all install
```

# 9. Building FlatBuffers for your Raspberry Pi

To build FlatBuffers for your Raspberry Pi, enter the following commands:

```
cd $BASEDIR/flatbuffers-1.12.0
mkdir build-arm32
cd build-arm32
CXXFLAGS="-fPIC" cmake .. -DCMAKE_C_COMPILER=/usr/bin/arm-linux-gnueabihf-gcc \
-DCMAKE_CXX_COMPILER=/usr/bin/arm-linux-gnueabihf-g++ \
-DFLATBUFFERS_BUILD_FLATC=1 \
-DCMAKE_INSTALL_PREFIX:PATH=$BASEDIR/flatbuffers-arm32 \
-DFLATBUFFERS_BUILD_TESTS=0
make all install
```

# 10. Building TensorFlow Lite

To build TensorFlow Lite, enter the following commands:

```
cd $BASEDIR
mkdir tflite
cd tflite
cp $BASEDIR/tensorflow/tensorflow/lite/schema/schema.fbs .
$BASEDIR/flatbuffers-1.12.0/build/flatc -c --gen-object-api --reflect-types --
reflect-names schema.fbs
```

# 11. Building Arm NN

The steps to build Arm NN in this guide are being replaced by the Arm NN Build Tool and will be removed soon.


Use the Arm NN Build Tool as a user-friendly way to build Arm NN and its dependencies.

To build Arm NN:

1. Enter the following commands:

```
cd $BASEDIR/armnn
mkdir build
cd build
```

2. Place the library files you require in the build directory. To place the library files, enter:

```
cmake .. -DCMAKE_LINKER=/usr/bin/arm-linux-gnueabihf-ld \
-DCMAKE_C_COMPILER=/usr/bin/arm-linux-gnueabihf-gcc \
-DCMAKE_CXX_COMPILER=/usr/bin/arm-linux-gnueabihf-g++ \
-DCMAKE_C_COMPILER_FLAGS=-fPIC \
-DCMAKE_CXX_FLAGS=-mfpu=neon \
-DARMCOMPUTE_ROOT=$BASEDIR/ComputeLibrary \
-DARMCOMPUTE_BUILD_DIR=$BASEDIR/ComputeLibrary/build \
-DBUILD_ONNX_PARSER=1 \
-DONNX_GENERATED_SOURCES=$BASEDIR/onnx \
-DBUILD_TF_LITE_PARSER=1 \
-DTF_LITE_GENERATED_PATH=$BASEDIR/tflite \
-DFLATBUFFERS_ROOT=$BASEDIR/flatbuffers-arm32 \
-DFLATC_DIR=$BASEDIR/flatbuffers-1.12.0/build \
-DPROTOBUF_ROOT=$BASEDIR/protobuf-arm \
-DARMCOMPUTENEON=1 \
-DARMNNREF=1
make
```

   Arm estimates that your virtual machine will take approximately 12 minutes to execute these commands.

   If you want to include standalone sample dynamic backend tests, add the following 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 on Raspberry Pi>
```

   Also, build the standalone sample dynamic backend after building Arm NN using the following commands:

```
#Set the versions based on /armnn/include/armnn/Version.hpp
ARMNN_MAJOR_VERSION=<ARMNN_MAJOR_VERSION>
ARMNN_MINOR_VERSION=<ARMNN_MINOR_VERSION>
ARMNN_PATCH_VERSION=<ARMNN_PATCH_VERSION>
```

```
cd $BASEDIR/armnn/src/dynamic/sample
mkdir build
cd build
cmake -DCMAKE_LINKER=/usr/bin/arm-linux-gnueabihf-ld \
-DCMAKE_C_COMPILER=/usr/bin/arm-linux-gnueabihf-gcc \
-DCMAKE_CXX_COMPILER=/usr/bin/arm-linux-gnueabihf-g++ \
-DCMAKE_CXX_FLAGS=--std=c++14 \
-DCMAKE_C_COMPILER_FLAGS=-fPIC \
-DARMNN_PATH=$BASEDIR/armnn/build/libarmnn.so.$ARMNN_MAJOR_VERSION.
$ARMNN_MINOR_VERS
ION ..
make
```

# 12. Extracting Arm NN to your Raspberry Pi and running a sample program

The following steps cover how to extract Arm NN to your Raspberry Pi and how to run a sample program.

## 12.1 Creating an archive of cross-compiled libraries, binaries, and directories

You must create an archive of cross-compiled libraries, binaries, and directories.

To create an archive of cross-compiled libraries, binaries, and directories, use the following commands:

1. Copy the following libraries, binaries, and directories from your virtual machine. To copy these libraries, binaries, and directories enter the following commands:

```
#Set the versions based on /armnn/include/armnn/Version.hpp for Arm NN and
corresponding Version.hpp in /armnn/include/armnnOnnxParser and
/armnn/include/armnnTfLiteParser for Onnx and TfLite prasers respectively
ARMNN_MAJOR_VERSION=<armnn_major_version>
ARMNN_MINOR_VERSION=<armnn_minor_version>
ARMNN_PATCH_VERSION=<armnn_patch_version>
TFLITE_PARSER_MAJOR_VERSION=<tflite_major_version>
TFLITE_PARSER_MINOR_VERSION=<tflite_minor_version>
TFLITE_PARSER_PATCH_VERSION=<tflite_patch_version>
ONNX_PARSER_MAJOR_VERSION=<onnx_patch_version>
ONNX_PARSER_MINOR_VERSION=<onnx_patch_version>
ONNX_PARSER_PATCH_VERSION=<onnx_patch_version>
cd $BASEDIR
mkdir armnn-dist
mkdir armnn-dist/armnn
mkdir armnn-dist/armnn/lib
cp $BASEDIR/armnn/build/libarmnn.so.$ARMNN_MAJOR_VERSION.$ARMNN_MINOR_VERSION
$BASEDIR/armnn-dist/armnn/lib
ln -s libarmnn.so.$ARMNN_MAJOR_VERSION.$ARMNN_MINOR_VERSION $BASEDIR/armnn-dist/
armnn/lib/libarmnn.so.$ARMNN_MAJOR_VERSION
ln -s libarmnn.so.$ARMNN_MAJOR_VERSION $BASEDIR/armnn-dist/armnn/lib/libarmnn.so
cp
$BASEDIR/armnn/build/libarmnnTfLiteParser.so.$TFLITE_PARSER_MAJOR_VERSION.
$TFLITE_P
ARSER_MINOR_VERSION.$TFLITE_PARSER_PATCH_VERSION $BASEDIR/armnn-dist/armnn/lib
ln -s
libarmnnTfLiteParser.so.$TFLITE_PARSER_MAJOR_VERSION.
$TFLITE_PARSER_MINOR_VERSION.$
TFLITE_PARSER_PATCH_VERSION $BASEDIR/armnn-dist/armnn/lib/
libarmnnTfLiteParser.so.$TFLITE_PARSER_MAJOR_VERSION
ln -s libarmnnTfLiteParser.so.$TFLITE_PARSER_MAJOR_VERSION $BASEDIR/armnn-dist/
armnn/lib/libarmnnTfLiteParser.so
cp
$BASEDIR/armnn/build/libarmnnOnnxParser.so.$ARMNN_MAJOR_VERSION.
$ARMNN_MINOR_VERSIO
N $BASEDIR/armnn-dist/armnn/lib
ln -s
libarmnnOnnxParser.so.$ONNX_PARSER_MAJOR_VERSION.$ONNX_PARSER_MINOR_VERSION.
$ONNX_P
```

```
ARSER_PATCH_VERSION $BASEDIR/armnn-dist/armnn/lib/libarmnnOnnxParser.so.
$ONNX_PARSER_MAJOR_VERSION
ln -s libarmnnOnnxParser.so.$ONNX_PARSER_MAJOR_VERSION $BASEDIR/armnn-dist/armnn/
lib/libarmnnOnnxParser.so
cp $BASEDIR/protobuf-arm/lib/libprotobuf.so.23.0.0 $BASEDIR/armnn-dist/armnn/lib/
libprotobuf.so
cp $BASEDIR/protobuf-arm/lib/libprotobuf.so.23.0.0 $BASEDIR/armnn-dist/armnn/lib/
libprotobuf.so.23
cp -r $BASEDIR/armnn/include $BASEDIR/armnn-dist/armnn/include
```

2. To test that your build of Arm NN works correctly, you need to run Unit Tests. To enable the running of Unit Tests, also copy the `libtimelineDecoder.so`,

   `libtimelineDecoderJson.so`, and `libarmnnBasePipeServer.so` libraries from your virtual machine.

3. Copy the following dynamic backend related files from your virtual machine. To copy these files, enter the following commands:

```
mkdir -p $BASEDIR/armnn-dist/src/backends/backendsCommon/test/
cp -r $BASEDIR/armnn/build/src/backends/backendsCommon/test/testSharedObject
 $BASEDIR/armnn-dist/src/backends/backendsCommon/test/testSharedObject/

cp -r $BASEDIR/armnn/build/src/backends/backendsCommon/test/testDynamicBackend/
 $BASEDIR/armnn-dist/src/backends/backendsCommon/test/testDynamicBackend/
cp -r $BASEDIR/armnn/build/src/backends/backendsCommon/test/backendsTestPath1/
 $BASEDIR/armnn-dist/src/backends/backendsCommon/test/backendsTestPath1/

mkdir -p $BASEDIR/armnn-dist/src/backends/backendsCommon/test/backendsTestPath2
cp $BASEDIR/armnn/build/src/backends/backendsCommon/test/backendsTestPath2/
Arm_CpuAcc_backend.so $BASEDIR/armnn-dist/src/backends/backendsCommon/test/
backendsTestPath2/

ln -s Arm_CpuAcc_backend.so $BASEDIR/armnn-dist/src/backends/backendsCommon/test/
backendsTestPath2/Arm_CpuAcc_backend.so.1
ln -s Arm_CpuAcc_backend.so.1 $BASEDIR/armnn-dist/src/backends/backendsCommon/
test/backendsTestPath2/Arm_CpuAcc_backend.so.1.2
ln -s Arm_CpuAcc_backend.so.1.2 $BASEDIR/armnn-dist/src/backends/backendsCommon/
test/backendsTestPath2/Arm_CpuAcc_backend.so.1.2.3
cp $BASEDIR/armnn/build/src/backends/backendsCommon/test/backendsTestPath2/
Arm_GpuAcc_backend.so $BASEDIR/armnn-dist/src/backends/backendsCommon/test/
backendsTestPath2/
ln -s nothing $BASEDIR/armnn-dist/src/backends/backendsCommon/test/
backendsTestPath2/Arm_no_backend.so

mkdir -p $BASEDIR/armnn-dist/src/backends/backendsCommon/test/backendsTestPath3

cp -r $BASEDIR/armnn/build/src/backends/backendsCommon/test/backendsTestPath5/
 $BASEDIR/armnn-dist/src/backends/backendsCommon/test/backendsTestPath5
cp -r $BASEDIR/armnn/build/src/backends/backendsCommon/test/backendsTestPath6/
 $BASEDIR/armnn-dist/src/backends/backendsCommon/test/backendsTestPath6

mkdir -p $BASEDIR/armnn-dist/src/backends/backendsCommon/test/backendsTestPath7

cp -r $BASEDIR/armnn/build/src/backends/backendsCommon/test/backendsTestPath9/
 $BASEDIR/armnn-dist/src/backends/backendsCommon/test/backendsTestPath9

mkdir -p $BASEDIR/armnn-dist/src/backends/dynamic/reference
cp $BASEDIR/armnn/build/src/backends/dynamic/reference/Arm_CpuRef_backend.so
 $BASEDIR/armnn-dist/src/backends/dynamic/reference/
```

If you enable the standalone sample dynamic backend tests, also copy the dynamic backend file using the following commands:

```
mkdir -p $BASEDIR/armnn-dist/src/dynamic/sample
cp $BASEDIR/armnn/src/dynamic/sample/build/libArm_SampleDynamic_backend.so
$BASEDIR/armnn-dist/src/dynamic/sample/
cp $BASEDIR/armnn/samples/DynamicSample.cpp $BASEDIR/armnn-dist
```

4. Copy the Unit Tests and a sample Arm NN program. To copy this test and program, enter the following commands:

```
cp $BASEDIR/armnn/build/UnitTests $BASEDIR/armnn-dist
cp $BASEDIR/armnn/samples/SimpleSample.cpp $BASEDIR/armnn-dist
```

5. Create the archive for your Raspberry Pi. To create the archive, enter the following command:

```
tar -czf $BASEDIR/armnn-dist.tar.gz armnn-dist
```

## 12.2 Extract the libraries, binaries, and directories to your Raspberry Pi

You must extract the libraries, binaries, and directories to your Raspberry Pi.

To extract the libraries, binaries, and directories enter the following commands:

```
cd /home/pi
tar -xzf /home/pi/armnn-dist.tar.gz
```

## 12.3 Compiling and running a sample Arm NN program on your Raspberry Pi

You must compile and run a sample C++ program that uses Arm NN on your Raspberry Pi.

1. Enter the following commands:

```
export LD_LIBRARY_PATH=/home/pi/armnn-dist/armnn/lib
cd /home/pi/armnn-dist
```

To compile the program, enter the following commands:

```
g++ SimpleSample.cpp -I/home/pi/armnn-dist/armnn/include -L/home/pi/armnn-dist/
armnn/lib -lpthread -larmnn -o SimpleSample
g++ DynamicSample.cpp -I/home/pi/armnn-dist/armnn/include -L/home/pi/armnn-dist/
armnn/lib -lpthread -larmnn -o DynamicSample
```

2. To run the program, enter the following command:

```
./SimpleSample
```

   The console returns the following:

```
Please enter a number:
```

   Enter your number. For example:

```
345
```

   The console returns the following:

```
Your number was 345
```

3. If you enable the standalone sample dynamic backend tests, you can run a sample dynamic backend program as a test.

   To compile the program, enter the following commands:

```
g++ DynamicSample.cpp -I/home/pi/armnn-dist/armnn/include -I/home/pi/armnn-dist/
boost/include -L/home/pi/armnn-dist/armnn/lib -larmnn -o DynamicSample
```

   To run the program, enter the following command:

```
./DynamicSample
```

   If the test is successful, the console returns the following:

```
Addition operator result is {15,11}
```

   If the test fails, the console returns an error message describing the reason for failure.

# 13. Testing your build

To test that your build of Arm NN works correctly, run the Unit Tests. To run the Unit Tests, enter the following:

```
export LD_LIBRARY_PATH=/home/pi/armnn-dist/armnn/lib
cd /home/pi/armnn-dist
./UnitTests
```

If the tests are successful, they output the following to the console:

```
*** 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.


- Arm is aware of an issue resulting in a NeonTimerMeasure unit test error on Raspberry Pi. The implementation by the Raspberry Pi of the timing libraries Arm NN uses to get its timing data causes this error. You can safely ignore this error.
- External profiling for Arm NN on the Raspberry Pi platform is currently not fully supported and results in some External profiling unit tests failing.

# 14. Related information

Arm resources:

- Arm Community - ask development questions and find articles and blogs on specific topics from Arm experts.
- Arm NN Github - raise queries or issues associated with the Arm NN how-to guides.
- Arm NN Product Documentation - find out more about the latest Arm NN features.

# 15. Next Steps

You are now ready to compile and run programs that use Arm NN on your Raspberry Pi.

# Appendix A Revisions

This appendix describes the technical changes between released issues of this book.

**Table A-1: First release for version 1.01**

| Change | Location |
|---|---|
| First release | — |

**Table A-2: First release for version 1.02**

| Change | Location |
|---|---|
| Minor fixes | Building Arm NN |

**Table A-3: First release for version 1.03**

| Change | Location |
|---|---|
| Adds note on known issue | Testing your build |

**Table A-4: First release for version 1.04**

| Change | Location |
|---|---|
| Adds note on using the correct cross-compiler version | Before you begin |

**Table A-5: First release for version 1.05**

| Change | Location |
|---|---|
| Adds new sample code to support the 19.08 release of Arm NN | Extracting Arm NN to your Raspberry Pi and running a sample program |
| Updates formatting in the Building the boost library for your Raspberry pi | — |

**Table A-6: First release for version 1.06**

| Change | Location |
|---|---|
| Adds two extra commands | Downloading the repositories and bundles |

**Table A-7: First release for version 1.07**

| Change | Location |
|---|---|
| Adds extra commands | Downloading the repositories and bundles |

**Table A-8: First release for version 1.08**

| Change | Location |
|---|---|
| Adds extra commands | Downloading the repositories and bundles |

**Table A-9: First release for version 1.09**

| Change | Location |
|---|---|
| Adds instructions on including standalone dynamic backend tests | — |
| Updates the codeblock | Downloading the repositories and bundles |

| Change | Location |
|---|---|
| Updates the codeblocks in step two | Building Arm NN |
| Updates the codeblocks and adds in new step three | Extracting Arm NN to your Raspberry Pi and running a sample program |
| Adds to the note | Testing your build |

**Table A-10: First release for version 1.10**

| Change | Location |
|---|---|
| Adds to the list of repositories and bundles you must download | Downloading the repositories and bundles |
| Adds Generating the TensorFlow Portobuf library, Building FlatBuffers, Building FlatBuffers for Raspberry Pi, and Building TensorFlow Lite sub-sections to the Building the Google Protobuf Library section. | — |
| Updates the commands for placing the library files | Building Arm NN |
| Adds steps for downloading other libraries | — |

**Table A-11: First release for version 1.11**

| Change | Location |
|---|---|
| Removes the AssertZeroExitCode commands | — |

**Table A-12: First release for version 1.12**

| Change | Location |
|---|---|
| Updates to reflect support for Google protobuf library 3.5.2 | — |
| Updates instructions on running Unit Tests | — |

**Table A-13: First release for version 1.13**

| Change | Location |
|---|---|
| Updates the protobuf version | — |
| Updates the TensorFlow checkout | — |
| Updates the ONNX checkout | — |
| Updates the libprotobuf version | — |

**Table A-14: First release for version 1.14**

| Change | Location |
|---|---|
| Updates the tested Ubuntu versions | — |
| Adds some new installation commands for g++ wget unzip xxd | — |
| Adds two cd commands to the instructions for downloading the repositories and bundles. | Downloading the repositories and bundles |
| Changes the CXXFLAGS DCMAKE_INSTALL_PREFIX:PATH value | — |

**Table A-15: First release for version 1.15**

| Change | Location |
|---|---|
| Removes Tensorflow support | — |
| Includes commands to add -ipthread while compiling Arm NN. | — |

**Table A-16: First release for version 21.08**

| Change | Location |
|---|---|
| Adds notes to overview | Overview |
| Removes boost from commands | Downloading the repositories and bundles |
| Removes the Building the Boost library for your Raspberry Pi section. | — |
| Removes commands | Building Arm NN |

**Table A-17: Second release for version 21.08**

| Change | Location |
|---|---|
| Corrects typos | Before you begin |
| Fixes command | Downloading the repositories and bundles |
| Fixes the formatting for these two sections | Building the Google Protobuf library for your virtual machine<br><br>Building the Google Protobuf library for your Raspberry Pi |
| Fixes command | Building the Google Protobuf library for your Raspberry Pi |
| Adds $BASEDIR in front of the cd command | Building FlatBuffers |
| Fixes commands | Building Arm NN |

**Table A-18: Third release for version 21.08**

| Change | Location |
|---|---|
| Adds additional resources | Related information |
| Updates the SHA in the Tensorflow repository | Downloading the repositories and bundles |

**Table A-19: First release for version 22.02**

| Change | Location |
|---|---|
| Updates the list of Raspberry Pi models that these instructions have been tested on. | Before you begin |

**Table A-20: Second release for version 22.02**

| Change | Location |
|---|---|
| Updates the repositories and bundles download command. | Downloading the repositories and bundles |

**Table A-21: First release for version 22.05**

| Change | Location |
|---|---|
| Updates step two of the repositories and bundles download command. | Downloading the repositories and bundles |

**Table A-22: First release for 22.08**

| Change | Location |
|---|---|
| Adds note about the Arm NN Build Tool. | Various |