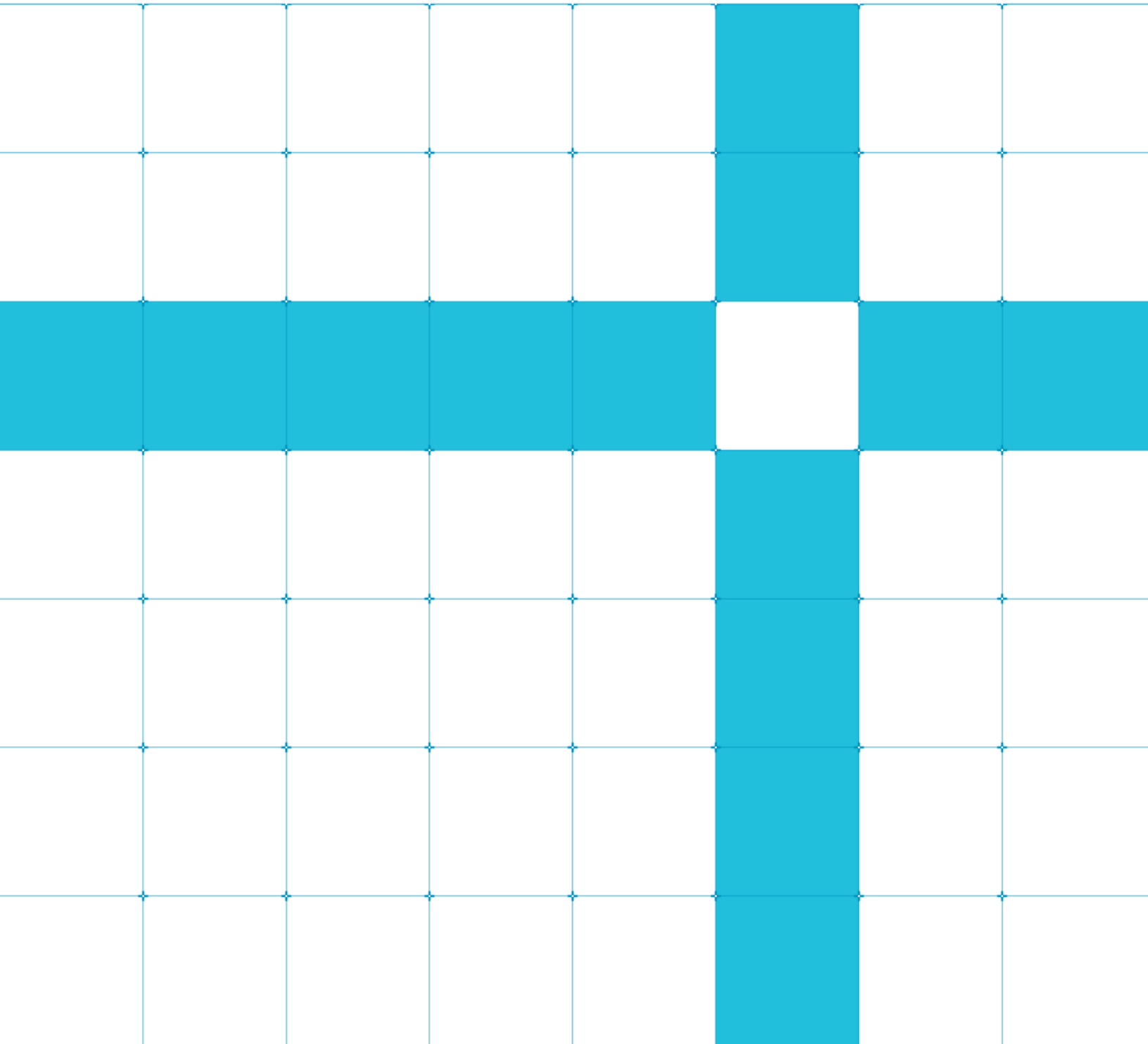




Deploying a Caffe MNIST model using the Arm NN SDK

Version 1.1



Deploying a Caffe MNIST model using the Arm NN SDK

Copyright © 2018-2019 Arm Limited (or its affiliates). All rights reserved.

Release Information

Document History

Version	Date	Confidentiality	Change
1.0	18 April 2018	Non-Confidential	First release.
1.1	04 September 2019	Non-Confidential	Adds additional lines of code to the codeblock in the Deploying an application using the Arm NN SDK section.

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm's trademark usage guidelines at <http://www.arm.com/company/policies/trademarks>.

Copyright © 2018-2019 Arm Limited (or its affiliates). All rights reserved.

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

www.arm.com

Contents

1 Overview	5
2 Before you begin	6
3 Building your Caffe model using the Arm NN SDK	7
3.1. Load and parse the MNIST test set	7
3.2. Import a graph	7
3.3. Optimize and load onto a compute device	8
3.4. Run a graph on a compute device	8
4 Deploying an application using the Arm NN SDK	9

1 Overview

Using an example application, this guide shows you how to run a Caffe model using the open-source [Arm NN SDK](#). You will be able to use the knowledge you gain from this guide to run your own models on Arm Cortex CPUs and Mali GPUs.

2 Before you begin

This guide assumes:

- You are familiar with neural networks and the MNIST dataset. If you are new to either of these concepts, read this [Caffe introduction](#) and this [overview of MNIST](#).
- You have access to the complete [example application](#) with source code, data, and model.

3 Building your Caffe model using the Arm NN SDK

To run a Caffe model using the Arm NN SDK:

1. Load and parse the MNIST test set.
2. Import a graph.
3. Optimize and load onto a compute device.
4. Run a graph on a device.

Using the example code, this guide walks you through each step.

3.1. Load and parse the MNIST test set

To begin building your own Caffe model, load and parse the MNIST test set.

This example code loads and parses the MNIST test set:

```
// Load a test image and its correct label
std::string dataDir = "data/";
int testImageIndex = 0;
std::unique_ptr input = loadMnistImage(dataDir, testImageIndex);
```

3.2. Import a graph

The Arm NN SDK provides parsers for reading model files from Caffe.

The SDK supports Caffe graphs in text and binary ProtoBuf formats. To import the graph:

1. Load the model.
2. Bind the input and output points of its graph.

This example code imports the graph:

```
// Import the Caffe model. Note: use CreateNetworkFromTextFile for text files.
armnnCaffeParser::ICaffeParserPtr parser =
armnnCaffeParser::ICaffeParser::Create();
armnn::INetworkPtr network =
    parser->CreateNetworkFromBinaryFile("model/lenet_iter_9000.caffemodel",
    { }, // input taken from file if empty
    { "prob" }); // output node
```

After this step, the code is common regardless of the framework that you started with.

This example code binds the input and output tensors to the data and selects the loaded network identifier:

```
// Find the binding points for the input and output nodes
armnnCaffeParser::BindingPointInfo inputBindingInfo =
    parser->GetNetworkInputBindingInfo("data");
armnnCaffeParser::BindingPointInfo outputBindingInfo =
    parser->GetNetworkOutputBindingInfo("prob");
```

You can read the result of the inference from the output array and compare it to the MnistImage label from the data file.

3.3. Optimize and load onto a compute device

You must optimize your network and load it onto a compute device. The Arm NN SDK supports optimized execution on multiple CPU and GPU devices. Before you start executing a graph, you must select the appropriate device context and optimize the graph for that device.

You specify a preferential order of compute devices you want workloads to run on when you call the `optimize()` function. The optimizer attempts to schedule the workloads on the first specified device and falls back to the following device if the first is not available or does not support the workload. The following are the compute devices you specify to run workloads:

- The **Arm Mali GPU** which you use `GpuAcc` to specify.
- The **Armv7 or Armv8 CPU** which you use `CpuAcc` to specify.

This example code optimizes and loads your network onto a compute device:

```
//// Create a context and optimize the network for one or more compute devices in order of
// preference
// e.g. GpuAcc, CpuAcc = if available run on Arm Mali GPU, else try to run on ARM v7 or v8 CPU
armnn::IRuntime::CreationOptions options;
armnn::IRuntimePtr context = armnn::IRuntime::Create(options);
armnn::IOptimizedNetworkPtr optNet = armnn::Optimize(*net, {armnn::Compute::GpuAcc,
armnn::Compute::CpuAcc}, runtime->GetDeviceSpec());

// Load the optimized network onto the device
armnn::NetworkId networkIdentifier;
context->LoadNetwork(networkIdentifier, std::move(optNet));
```

3.4. Run a graph on a compute device

Inference on a compute device is performed using the `EnqueueWorkload()` function of the context.

This example code runs a single inference on the test image:

```
// Run a single inference on the test image
std::array<float, 10=""> output;
armnn::Status ret = context->EnqueueWorkload(networkIdentifier,
    MakeInputTensors(inputBindingInfo, &input->image[0]),
    MakeOutputTensors(outputBindingInfo, &output[0]));
```

The `std::distance()` function in the following example code is used to find the index of the largest element in the output. This function is equivalent to NumPy's `argmax()` function.

```
// Convert 1-hot output to an integer label and print
int label = std::distance(output.begin(),
std::max_element(output.begin(), output.end()));
std::cout << "Predicted: " << label << std::endl;
std::cout << " Actual: " << input->label << std::endl;
return 0;
```


4 Deploying an application using the Arm NN SDK

You must link your application with the Arm NN SDK library and the Caffe parsing library. The following example code links your application. You must use your own Arm NN library path to replace `<your_armnn_library_path>` in the code:

```
g++ -O3 -std=c++14 -I$(ARMNN_INC) mnist_caffe.cpp -o mnist_caffe -L$(ARMNN_LIB)-larmnn -  
larmnnCaffeParser  
export LD_LIBRARY_PATH = <your_armnn_library_path>: $LD_LIBRARY_PATH  
./mnist_caffe
```

For convenience, the Arm NN SDK can run with a reference implementation on x86 for development and testing, but the optimized kernels are only available for Arm CPUs and GPUs. This means that you must run your profiling and optimization steps on a real Arm-powered device because x86 performance does not represent real performance.