eIQ® Neutron NPU for MCUs Lab

Part 1: Mobilenet

Revision 3

June 2024

NXP Semiconductors
www.nxp.com

Contents

1 Lab Overview....3

2 Software and Hardware Installation....3

2.1 Hardware....3

2.2 NXP Software Installation....3

3 Label Image Example....3

3.1 Convert Models....3

3.2 View Models....7

3.3 Run Models....8

4 Conclusion....18

1 Lab Overview

This document will cover how to convert models using NXP's eIQ Toolkit and will also highlight the performance improvements that can be achieved with the eIQ Neutron NPU on MCX devices.

2 Software and Hardware Installation

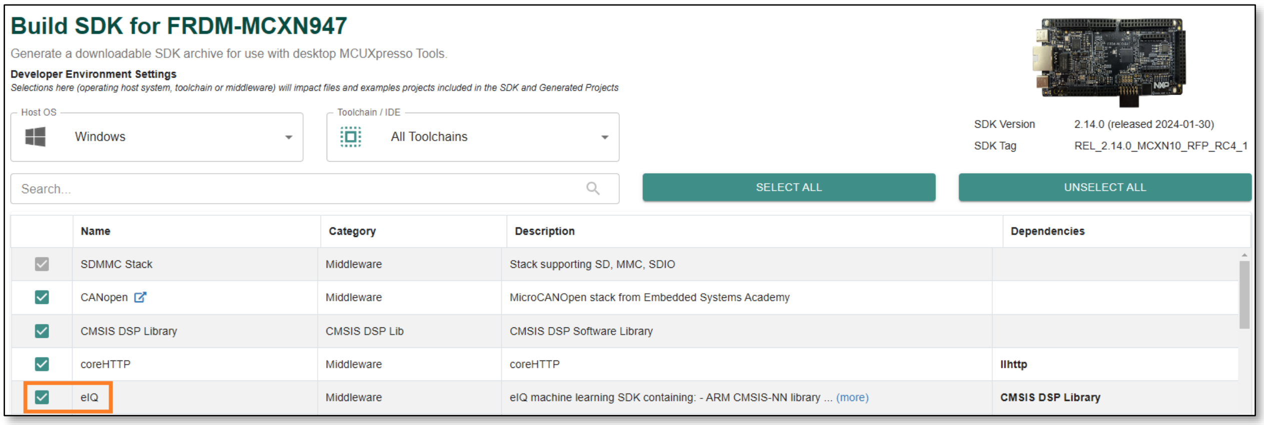
This section will cover the hardware and software needed for this lab.

2.1 Hardware

The FRDM-MCXN947 development board is used in this lab

2.2 NXP Software Installation

1. Install MCUXpresso IDE v11.9.1 or later
2. Install eIQ Toolkit 1.11.4 - During installation it is not required to install the optional components offered by the install wizard. Everything needed for this lab is already installed by default.
3. A quantized Mobilenet model can be found attached to this post or in the i.MX RT1170 MCUXpresso SDK v2.15.0 at **\boards\evkmimxrt1170\eiq_examples\tflm_label_image\doc**
4. Download MCUXpresso SDK 2.14.0 for FRDM-MCXN947
   It includes the eIQ software platform and demos. It must be 2.14.0 to ensure compatibility with eIQ Toolkit v1.11.4
   a) On the SDK builder page, make sure to select the "**eIQ**" middleware and that you're downloading version **2.14.0**



Build SDK for FRDM-MCXN947

Generate a downloadable SDK archive for use with desktop MCUXpresso Tools.

Developer Environment Settings

Selections here (operating host system, toolchain or middleware) will impact files and examples projects included in the SDK and Generated Projects

Host OS: Windows

Toolchain / IDE: All Toolchains

SDK Version: 2.14.0 (released 2024-01-30)

SDK Tag: REL_2.14.0_MCXN10_RFP_RC4_1

	Name	Category	Description	Dependencies
☑	SDMMC Stack	Middleware	Stack supporting SD, MMC, SDIO	
☑	CANopen	Middleware	MicroCANOpen stack from Embedded Systems Academy	
☑	CMSIS DSP Library	CMSIS DSP Lib	CMSIS DSP Software Library	
☑	coreHTTP	Middleware	coreHTTP	llhttp
☑	eIQ	Middleware	eIQ machine learning SDK containing: - ARM CMSIS-NN library ... (more)	CMSIS DSP Library

   b) Then click on the **Download SDK** button and accept the license agreement to download the zip file.

3 Label Image Example

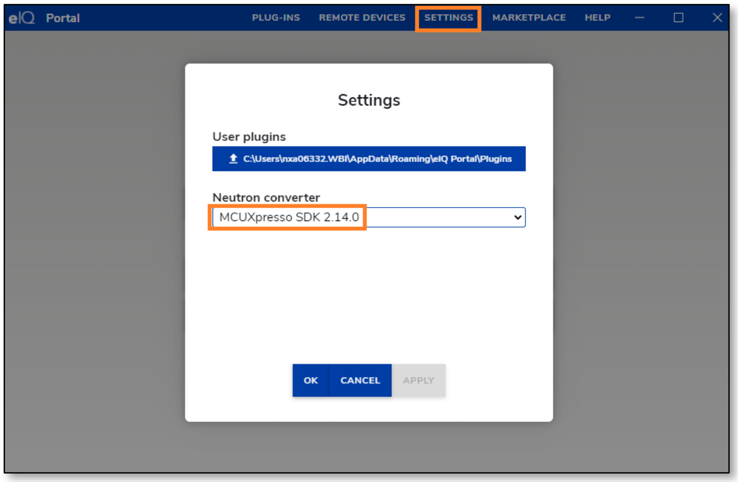
This section will use the eIQ Label Image example found in the MCX N MCUXpresso SDK to showcase how the eIQ Neutron NPU can significantly decrease inference times for quantized models.

3.1 Convert Models

Use eIQ Toolkit to convert a pre-existing Mobilenet model into a Neutron optimized model.

1. Download and unzip the Mobilenet v1 image classification TFLite model attached to this Community post or use this direct link.
2. Open eIQ Portal which is the GUI interface to eIQ Toolkit

3. After it opens, go to the menu bar and click on Settings. In the dialog box that comes up, make sure the Neutron Converter SDK matches the SDK version that you are using. In this case it should be set to **MCUXPresso SDK 2.14.0** by default and require no change. It is very important that the Neutron Converter tool version is targeted to the eIQ Neutron libraries used in the targeted SDK. Click on OK to save the setting and go back to the main screen.



Settings

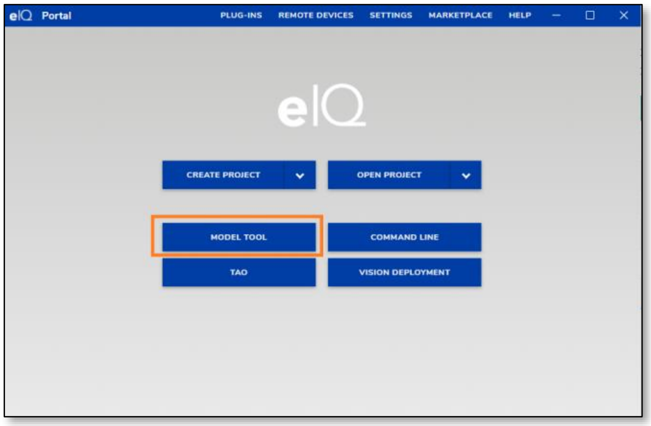
User plugins

C:\Users\nxa06332.WBI\AppData\Roaming\eIQ Portal\Plugins

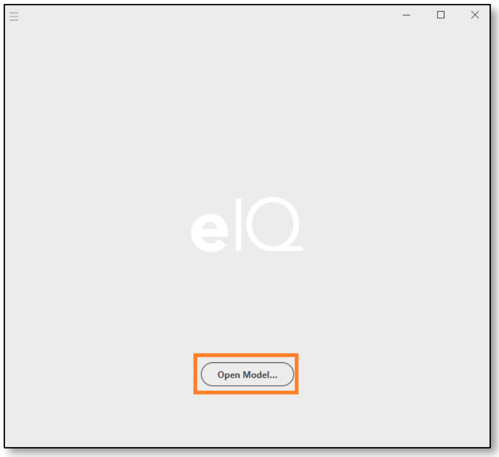
Neutron converter

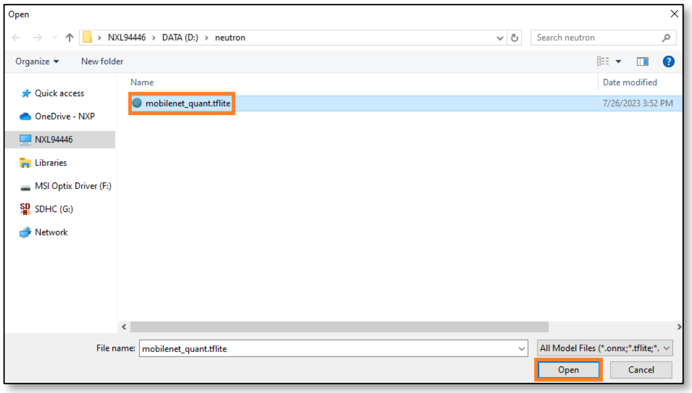
MCUXpresso SDK 2.14.0

4. Click on Model Tool

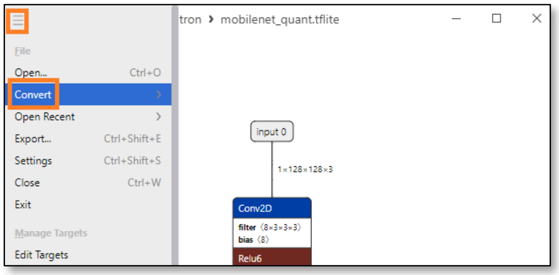


5. Click on Open Model and select the **mobilenet_quant.tflite** model that was unzipped from the file downloaded earlier.

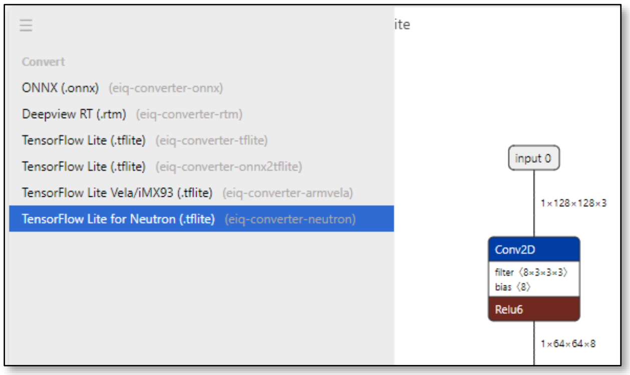




6. After the model is opened you'll see the various layers. Then click on the upper left corner to find the menu and select **Convert**

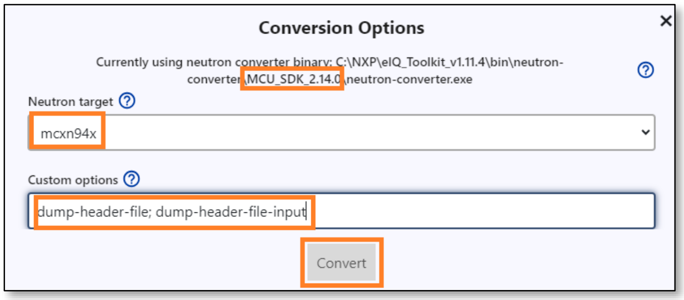


7. It may need to search for plugins. Let it.
8. After the plugins have been found, go back to the menu options and select Convert and then select **TensorFlow Lite for Neutron (.tflite)**


Convert
ONNX (.onnx) (eiq-converter-onnx)
Deepview RT (.rtm) (eiq-converter-rtm)
TensorFlow Lite (.tflite) (eiq-converter-tflite)
TensorFlow Lite (.tflite) (eiq-converter-onnx2tflite)
TensorFlow Lite Vela/iMX93 (.tflite) (eiq-converter-armvela)
TensorFlow Lite for Neutron (.tflite) (eiq-converter-neutron)
ite
input 0
1×128×128×3
Conv2D
filter ⟨8×3×3×3⟩
bias ⟨8⟩
Relu6
1×64×64×8

9. In the dialog box that comes up, select **mcxn94x** as the target and then in the custom options field put: **dump-header-file; dump-header-file-input**

   These two options will generate a C array of both the converted model and the input model, which we'll use to compare the performance of them. In typical situations you would only need the dump-header-file option though. Also make sure the Neutron Converter tool for MCU_SDK_2.14.0 is being used. Click on **Convert** and save the converted TFLite file on your hard drive as **mobilenet_npu.tflite**

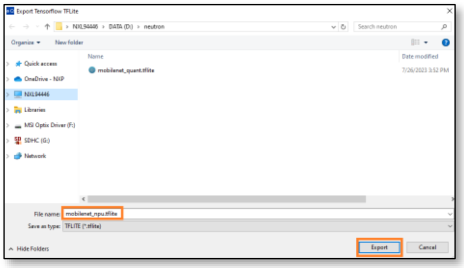


Conversion Options

Currently using neutron converter binary: C:\NXP\eIQ_Toolkit_v1.11.4\bin\neutron-converter\MCU_SDK_2.14.0\neutron-converter.exe

Neutron target: mcxn94x

Custom options: dump-header-file; dump-header-file-input



Export Tensorflow TFLite

NXL94446 > DATA (D:) > neutron

mobilenet_quant.tflite 7/26/2023 3:52 PM

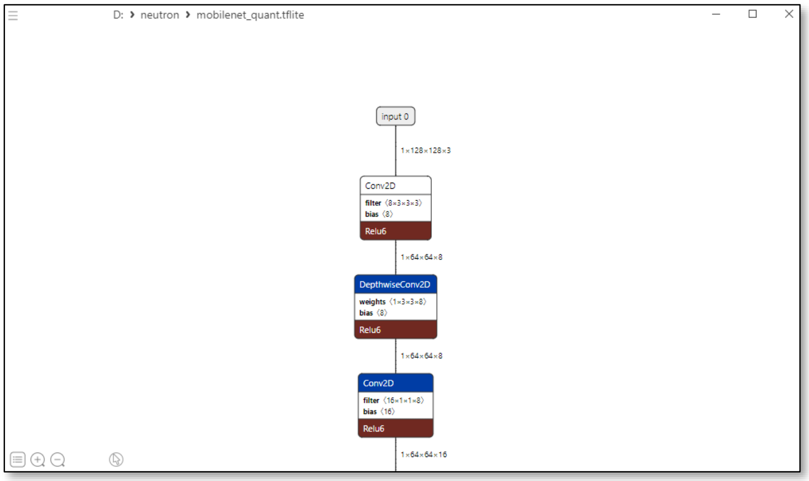
File name: mobilenet_npu.tflite

Save as type: TFLITE (*.tflite)

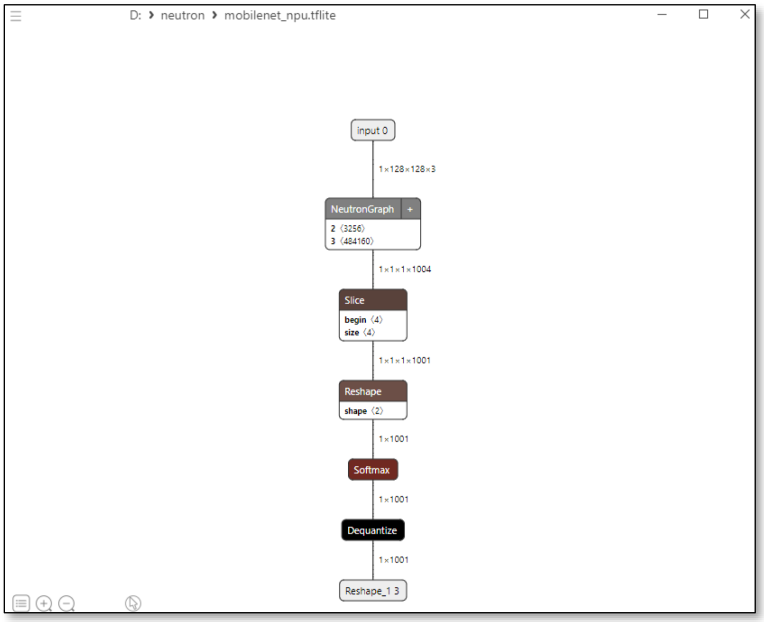
3.2 View Models

After conversion, you'll see the newly converted model optimized for the eIQ Neutron NPU pop up with the Model Tool. Take a moment to look at the original model compared to the new converted model.

1. The original TFLite file: **mobilenet_quant.tflite**



2. The Neutron converted file: **mobilenet_npu.tflite**



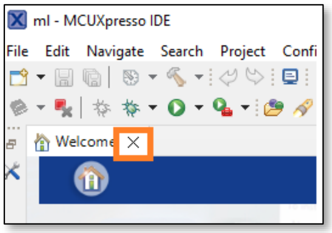
3. You can see how almost all the operators in the original model were replaced with a **NeutronGraph** operator. Those **NeutronGraph** operaters are what will be executed on the eIQ Neutron NPU when this model is ran on the MCX N94x. Any layers not converted to a NeutronGraph operator will instead be ran on the Cortex-M33 core.
4. Take a look at the file size of each of those header files and you can see that, in general, the NPU converted file will take up less flash space. Note that this might be counter-acted by the slightly increased size required for using the eIQ Neutron libraries.
5. During the conversion process the **dump-header-file** argument generated a header file for the NPU optimized model that can be used in the eIQ MCXUpresso SDK projects.

6. The **dump-header-file-input** argument generated a header file for the original non-converted model. This will be used so the inference time of the original model that only runs on the Cortex-M33 core can be compared to the NPU converted model that makes use of the eIQ Neutron NPU.
7. So let's run these models to see the performance improvements.

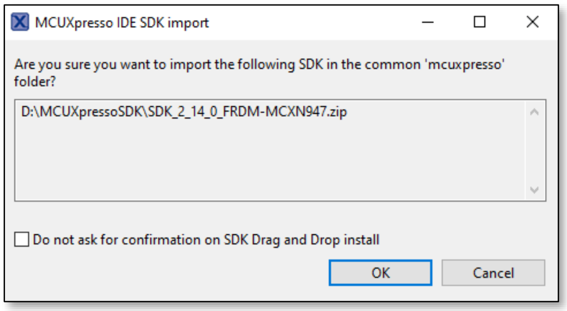
3.3 Run Models

Now let's use the MCUXpresso SDK eIQ Label Images example to run the models and see how long the inference time is.

8. Open MCUXpresso IDE and select a workspace location in an empty directory
9. Close the Welcome Screen tab by clicking on the X in that tab



10. Drag-and-drop the 2.14 FRDM-MCXN947 SDK zip file into the Installed SDKs window, located on a tab at the bottom of the screen named "Installed SDKs". You will get the following pop-up, so hit OK.



MCUXpresso IDE SDK import

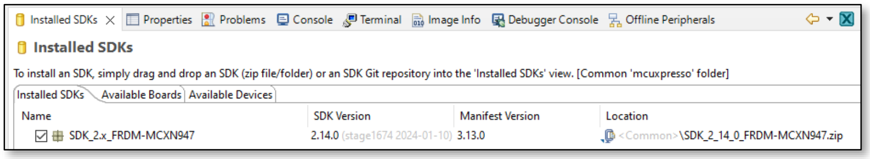
Are you sure you want to import the following SDK in the common 'mcuxpresso' folder?

D:\MCUXpressoSDK\SDK_2_14_0_FRDM-MCXN947.zip

☐ Do not ask for confirmation on SDK Drag and Drop install

OK Cancel

11. Once imported, the Installed SDK panel will look something like this:



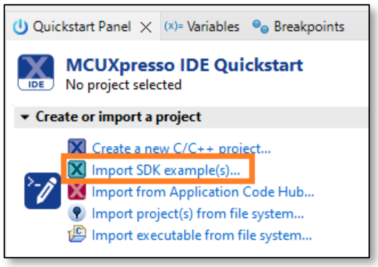
Installed SDKs

To install an SDK, simply drag and drop an SDK (zip file/folder) or an SDK Git repository into the 'Installed SDKs' view. [Common 'mcuxpresso' folder]

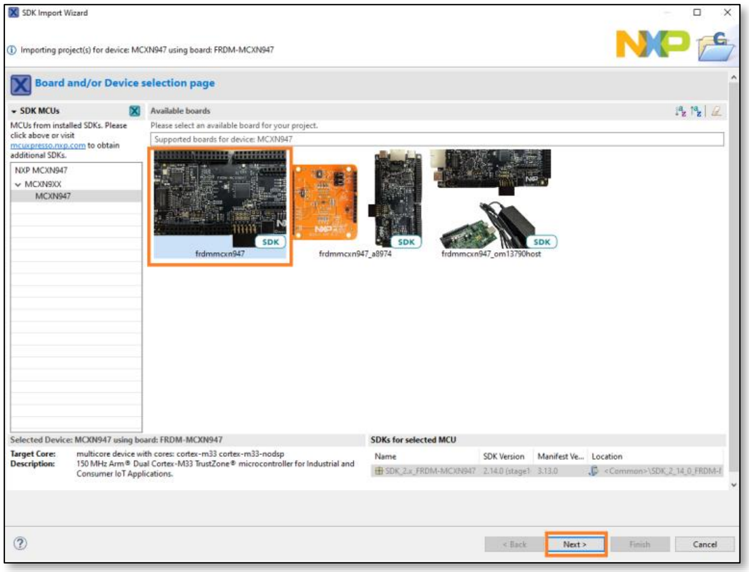
Installed SDKs | Available Boards | Available Devices

Name	SDK Version	Manifest Version	Location
☑ SDK_2.x_FRDM-MCXN947	2.14.0 (stage1674 2024-01-10)	3.13.0	<Common>\SDK_2_14_0_FRDM-MCXN947.zip

12. Next import the desired project. In the Quickstart Panel, select **Import SDK examples(s)…**


Quickstart Panel
Variables
Breakpoints
MCUXpresso IDE Quickstart
No project selected
Create or import a project
Create a new C/C++ project...
Import SDK example(s)...
Import from Application Code Hub...
Import project(s) from file system...
Import executable from file system...

13. Select the **frdmmcxn947** board and click on **Next**



SDK Import Wizard

Importing project(s) for device: MCXN947 using board: FRDM-MCXN947

Board and/or Device selection page

SDK MCUs

MCUs from installed SDKs. Please click above or visit mcuxpresso.nxp.com to obtain additional SDKs.

- NXP MCXN947
  - MCXN9XX
    - MCXN947

Available boards

Please select an available board for your project.

Supported boards for device: MCXN947

Selected Device: MCXN947 using board: FRDM-MCXN947

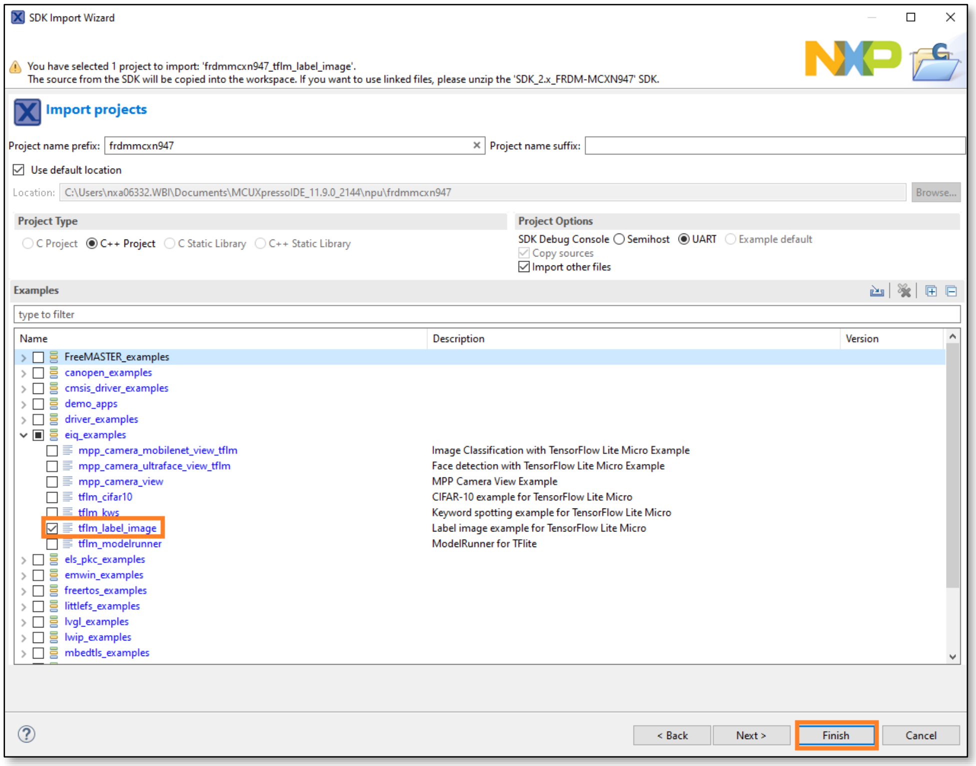
Target Core: multicore device with cores: cortex-m33 cortex-m33-nodsp

Description: 150 MHz Arm® Dual Cortex-M33 TrustZone® microcontroller for Industrial and Consumer IoT Applications.

SDKs for selected MCU

Name	SDK Version	Manifest Ve...	Location
SDK_2.x_FRDM-MCXN947	2.14.0 (stage1	3.13.0	<Common>\SDK_2_14_0_FRDM-I

14. Under the **eiq_examples** category, select the **tflm_label_image** example. Then click on **Finish** to select that project.



SDK Import Wizard

You have selected 1 project to import: 'frdmmcxn947_tflm_label_image'.
The source from the SDK will be copied into the workspace. If you want to use linked files, please unzip the 'SDK_2.x_FRDM-MCXN947' SDK.

Import projects

Project name prefix: frdmmcxn947 Project name suffix:

☑ Use default location

Location: C:\Users\nxa06332.WBI\Documents\MCUXpressoIDE_11.9.0_2144\npu\frdmmcxn947

Project Type

○ C Project ● C++ Project ○ C Static Library ○ C++ Static Library

Project Options

SDK Debug Console ○ Semihost ● UART ○ Example default

☑ Copy sources

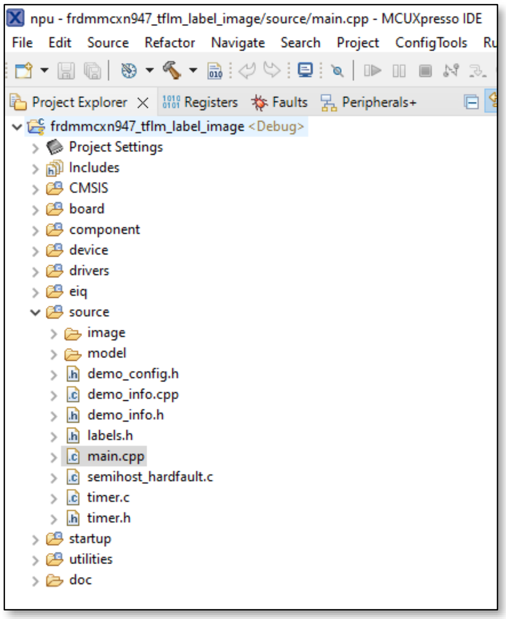
☑ Import other files

Examples

type to filter

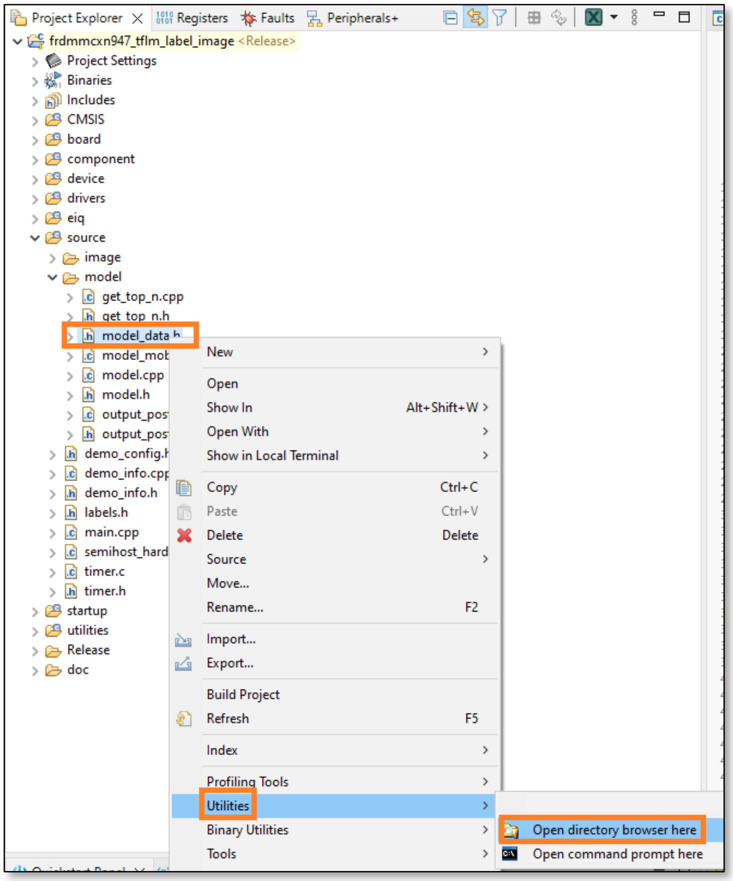
Name	Description	Version
□ FreeMASTER_examples		
□ canopen_examples		
□ cmsis_driver_examples		
□ demo_apps		
□ driver_examples		
■ eiq_examples		
□ mpp_camera_mobilenet_view_tflm	Image Classification with TensorFlow Lite Micro Example	
□ mpp_camera_ultraface_view_tflm	Face detection with TensorFlow Lite Micro Example	
□ mpp_camera_view	MPP Camera View Example	
□ tflm_cifar10	CIFAR-10 example for TensorFlow Lite Micro	
□ tflm_kws	Keyword spotting example for TensorFlow Lite Micro	
☑ tflm_label_image	Label image example for TensorFlow Lite Micro	
□ tflm_modelrunner	ModelRunner for TFlite	
□ els_pkc_examples		
□ emwin_examples		
□ freertos_examples		
□ littlefs_examples		
□ lvgl_examples		
□ lwip_examples		
□ mbedtls_examples		

15. It should look like the following when done:

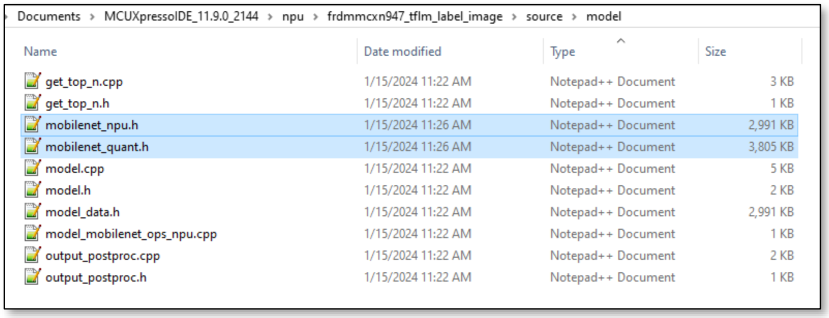

npu - frdmmcxn947_tflm_label_image/source/main.cpp - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Ru
Project Explorer Registers Faults Peripherals+
frdmmcxn947_tflm_label_image <Debug>
Project Settings
Includes
CMSIS
board
component
device
drivers
eiq
source
image
model
demo_config.h
demo_info.cpp
demo_info.h
labels.h
main.cpp
semihost_hardfault.c
timer.c
timer.h
startup
utilities
doc

16. Now we need to import the models that were generated in the last section into this project.

17. Navigate down to the **source->model** folder and right click on **model_data.h** and then select **Utilities->Open directory browser here** to open the location of that file on your hard drive.



18. Now copy and paste the two .h header files that were generated in the previous section into this file location. It should look like the following when complete:

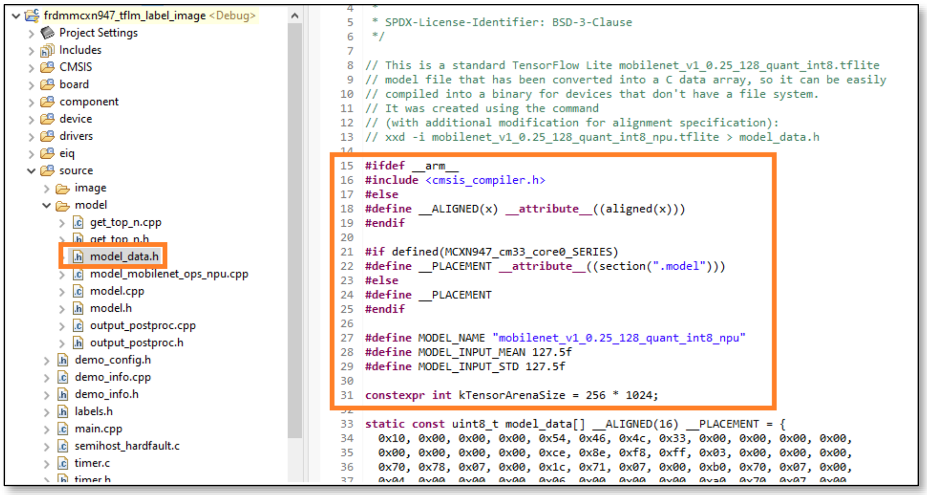


Documents › MCUXpressoIDE_11.9.0_2144 › npu › frdmmcxn947_tflm_label_image › source › model

Name	Date modified	Type	Size
get_top_n.cpp	1/15/2024 11:22 AM	Notepad++ Document	3 KB
get_top_n.h	1/15/2024 11:22 AM	Notepad++ Document	1 KB
mobilenet_npu.h	1/15/2024 11:26 AM	Notepad++ Document	2,991 KB
mobilenet_quant.h	1/15/2024 11:26 AM	Notepad++ Document	3,805 KB
model.cpp	1/15/2024 11:22 AM	Notepad++ Document	5 KB
model.h	1/15/2024 11:22 AM	Notepad++ Document	2 KB
model_data.h	1/15/2024 11:22 AM	Notepad++ Document	2,991 KB
model_mobilenet_ops_npu.cpp	1/15/2024 11:22 AM	Notepad++ Document	1 KB
output_postproc.cpp	1/15/2024 11:22 AM	Notepad++ Document	2 KB
output_postproc.h	1/15/2024 11:22 AM	Notepad++ Document	1 KB

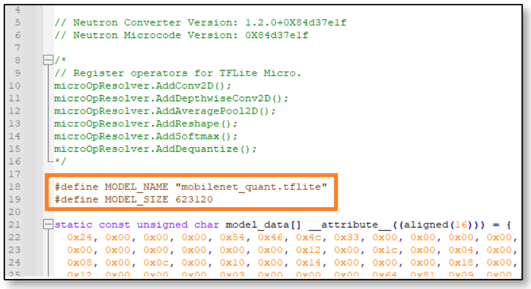
19. Now we need to slightly modify those two header files to add some information for the eIQ MCUXpresso SDK project that describe how much memory this model will require and to describe some of the normalization values that this model uses:

    a) Open **model_data.h,** which contains the default model for this example, and find the following section of code and copy it.

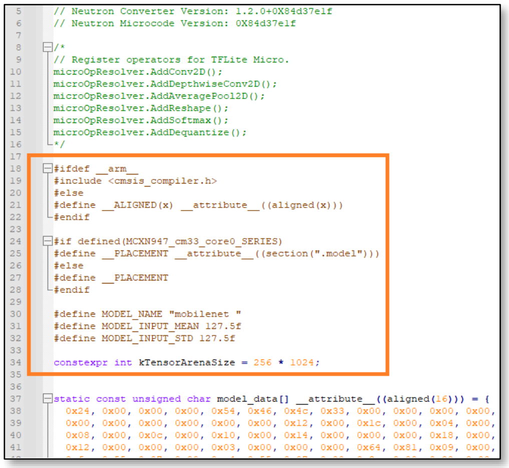


```
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

// This is a standard TensorFlow Lite mobilenet_v1_0.25_128_quant_int8.tflite
// model file that has been converted into a C data array, so it can be easily
// compiled into a binary for devices that don't have a file system.
// It was created using the command
// (with additional modification for alignment specification):
// xxd -i mobilenet_v1_0.25_128_quant_int8_npu.tflite > model_data.h

#ifdef __arm__
#include <cmsis_compiler.h>
#else
#define __ALIGNED(x) __attribute__((aligned(x)))
#endif

#if defined(MCXN947_cm33_core0_SERIES)
#define __PLACEMENT __attribute__((section(".model")))
#else
#define __PLACEMENT
#endif

#define MODEL_NAME "mobilenet_v1_0.25_128_quant_int8_npu"
#define MODEL_INPUT_MEAN 127.5f
#define MODEL_INPUT_STD 127.5f

constexpr int kTensorArenaSize = 256 * 1024;

static const uint8_t model_data[] __ALIGNED(16) __PLACEMENT = {
  0x10, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0xce, 0x8e, 0xf8, 0xff, 0x03, 0x00, 0x00, 0x00,
  0x70, 0x78, 0x07, 0x00, 0x1c, 0x71, 0x07, 0x00, 0xb0, 0x70, 0x07, 0x00,
```

b) Then in both **mobilenet_quant.h** and **mobilenet_ npu.h** copy that code above the array, overwriting the default #defines above the array. Make sure not to erase the commented lines at the top of the file as those comments will be used later. Note that the **MODEL_NAME** can be changed to "mobilenet".

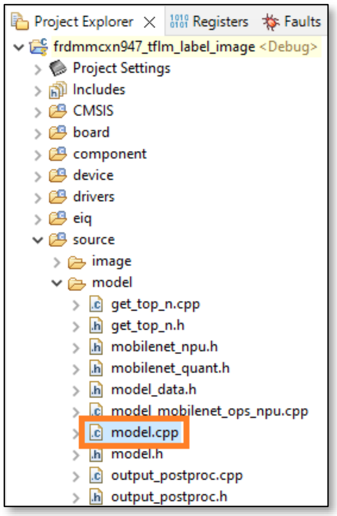


```
// Neutron Converter Version: 1.2.0+0X84d37e1f
// Neutron Microcode Version: 0X84d37e1f

/*
// Register operators for TFLite Micro.
microOpResolver.AddConv2D();
microOpResolver.AddDepthwiseConv2D();
microOpResolver.AddAveragePool2D();
microOpResolver.AddReshape();
microOpResolver.AddSoftmax();
microOpResolver.AddDequantize();
*/

#define MODEL_NAME "mobilenet_quant.tflite"
#define MODEL_SIZE 623120

static const unsigned char model_data[] __attribute__((aligned(16))) = {
  0x24, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x12, 0x00, 0x1c, 0x00, 0x04, 0x00,
  0x08, 0x00, 0x0c, 0x00, 0x10, 0x00, 0x14, 0x00, 0x00, 0x00, 0x18, 0x00,
```

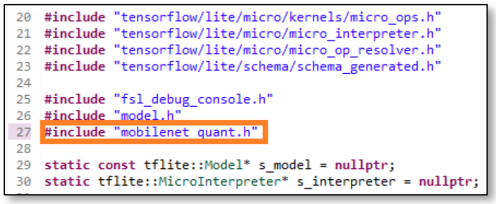


```
// Neutron Converter Version: 1.2.0+0X84d37e1f
// Neutron Microcode Version: 0X84d37e1f

/*
// Register operators for TFLite Micro.
microOpResolver.AddConv2D();
microOpResolver.AddDepthwiseConv2D();
microOpResolver.AddAveragePool2D();
microOpResolver.AddReshape();
microOpResolver.AddSoftmax();
microOpResolver.AddDequantize();
*/

#ifdef __arm__
#include <cmsis_compiler.h>
#else
#define __ALIGNED(x) __attribute__((aligned(x)))
#endif

#if defined(MCXN947_cm33_core0_SERIES)
#define __PLACEMENT __attribute__((section(".model")))
#else
#define __PLACEMENT
#endif

#define MODEL_NAME "mobilenet "
#define MODEL_INPUT_MEAN 127.5f
#define MODEL_INPUT_STD 127.5f

constexpr int kTensorArenaSize = 256 * 1024;


static const unsigned char model_data[] __attribute__((aligned(16))) = {
  0x24, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x12, 0x00, 0x1c, 0x00, 0x04, 0x00,
  0x08, 0x00, 0x0c, 0x00, 0x10, 0x00, 0x14, 0x00, 0x00, 0x00, 0x18, 0x00,
  0x12, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x64, 0x81, 0x09, 0x00,
```

20. After changing both files, now double click on **model.cpp** to open it.

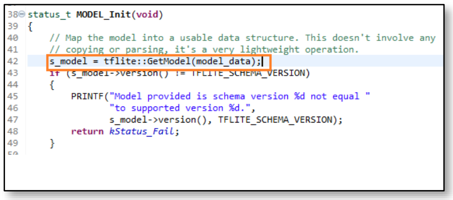


21. Go to line 27 and change it to point to the non-NPU accelerated model in **mobilenet_quant.h**. It should look like the following after changed:



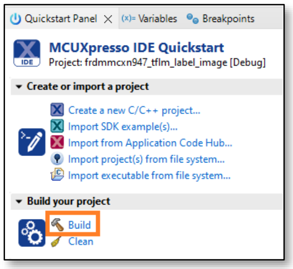
```
20 #include "tensorflow/lite/micro/kernels/micro_ops.h"
21 #include "tensorflow/lite/micro/micro_interpreter.h"
22 #include "tensorflow/lite/micro/micro_op_resolver.h"
23 #include "tensorflow/lite/schema/schema_generated.h"
24
25 #include "fsl_debug_console.h"
26 #include "model.h"
27 #include "mobilenet_quant.h"
28
29 static const tflite::Model* s_model = nullptr;
30 static tflite::MicroInterpreter* s_interpreter = nullptr;
```

22. Next look at line 42 in that same **model.cpp** file to find where the model is loaded by the TFLM inference engine using the C array name **model_data**. Because the model array name in the new header file is the same as the original header file we replaced, **no change is needed here**. This is just for informational purposes only.



```
38 status_t MODEL_Init(void)
39 {
40     // Map the model into a usable data structure. This doesn't involve any
41     // copying or parsing, it's a very lightweight operation.
42     s_model = tflite::GetModel(model_data);
43     if (s_model->version() != TFLITE_SCHEMA_VERSION)
44     {
45         PRINTF("Model provided is schema version %d not equal "
46                "to supported version %d.",
47                s_model->version(), TFLITE_SCHEMA_VERSION);
48         return kStatus_Fail;
49     }
```

23. Compile the project by clicking on the Build button in the Quickstart Panel in the lower left hand corner. Make sure the correct project is listed as well.
Note: Because TFLM is now deployed as a library in MCUXpresso SDK 2.14, the default Debug no-optimization target can be used and still have the same inference time as the Release high-optimization target.

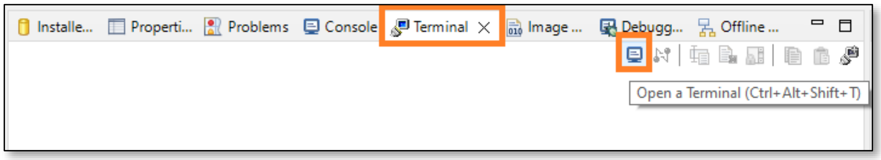


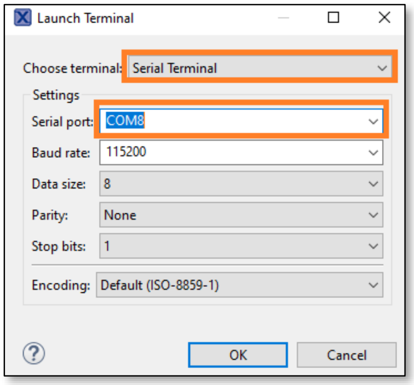
Quickstart Panel
Variables
Breakpoints
MCUXpresso IDE Quickstart

Project: frdmmcxn947_tflm_label_image [Debug]

Create or import a project
Create a new C/C++ project...
Import SDK example(s)...
Import from Application Code Hub...
Import project(s) from file system...
Import executable from file system...
Build your project
Build
Clean

24. Connect a USB C cable from your computer to the USB port on the FRDM-MCXN947 at J17
25. Open TeraTerm or other terminal program, and connect to the virtual COM port that debugger or UART-to-USB converter enumerated as. Use 115200 baud, 1 stop bit, no parity. There is a built-in serial terminal in MCUXpresso IDE that can be used as well:





Choose terminal: Serial Terminal

Settings

Serial port: COM8

Baud rate: 115200

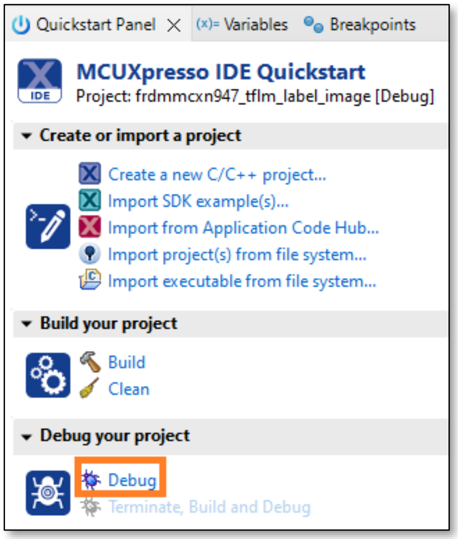
Data size: 8

Parity: None

Stop bits: 1

Encoding: Default (ISO-8859-1)

26. Then in MCUXpresso IDE click on Debug



Quickstart Panel | (x)= Variables | Breakpoints

MCUXpresso IDE Quickstart
Project: frdmmcxn947_tflm_label_image [Debug]

Create or import a project

- Create a new C/C++ project...
- Import SDK example(s)...
- Import from Application Code Hub...
- Import project(s) from file system...
- Import executable from file system...

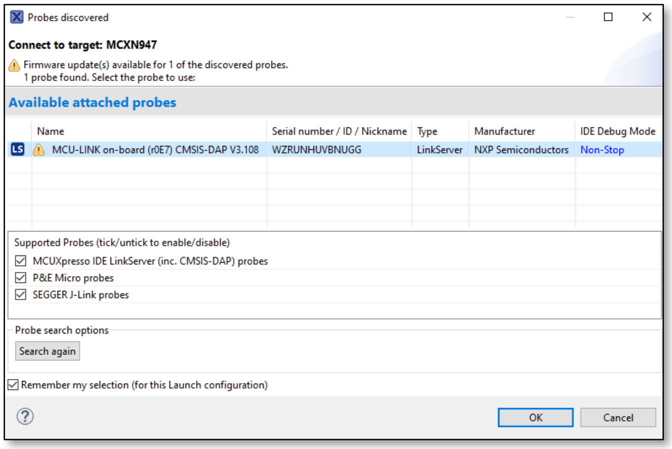
Build your project

- Build
- Clean

Debug your project

- Debug
- Terminate, Build and Debug

27. You'll see the following dialog box come up. It should list your debugger hardware. If there is a notice about a Firmware update available, that can be ignored for now. Click on **OK** to connect to the target.



Probes discovered

Connect to target: MCXN947

⚠ Firmware update(s) available for 1 of the discovered probes.
1 probe found. Select the probe to use:

Available attached probes

	Name	Serial number / ID / Nickname	Type	Manufacturer	IDE Debug Mode
LS	⚠ MCU-LINK on-board (r0E7) CMSIS-DAP V3.108	WZRUNHUVBNUGG	LinkServer	NXP Semiconductors	Non-Stop

Supported Probes (tick/untick to enable/disable)

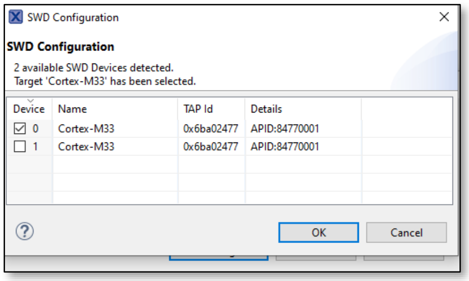
☑ MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes
☑ P&E Micro probes
☑ SEGGER J-Link probes

Probe search options

☑ Remember my selection (for this Launch configuration)

28. You should see MCUXpresso IDE connect and download the program to your board in the Console tab.

29. You may see this dialog box come up the first time you run the program. Keep the default selection to run on the primary core and hit OK



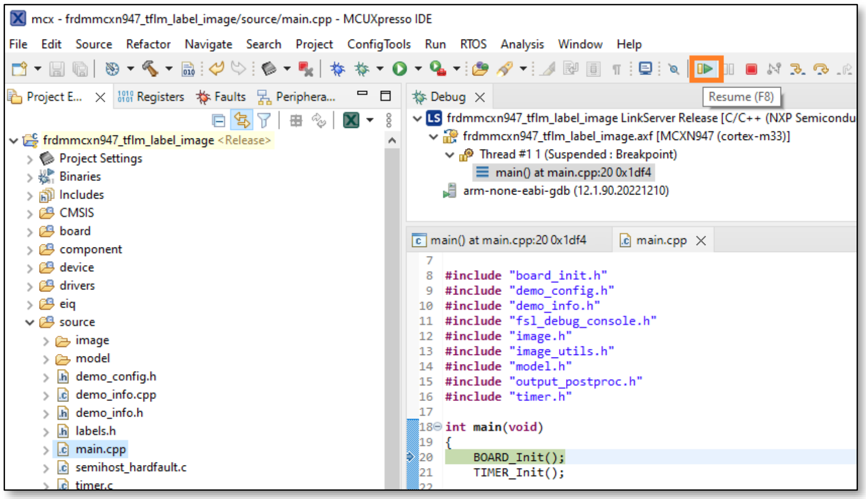
SWD Configuration

SWD Configuration

2 available SWD Devices detected.
Target 'Cortex-M33' has been selected.

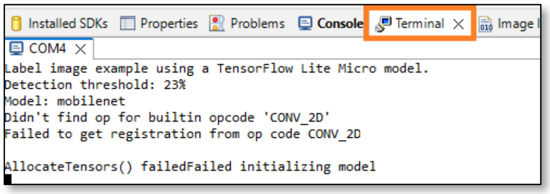
Device	Name	TAP Id	Details
☑ 0	Cortex-M33	0x6ba02477	APID:84770001
☐ 1	Cortex-M33	0x6ba02477	APID:84770001

30. Once complete, it will pause at the start of main(). Hit the Resume icon to run the program and look at the terminal tab.


mcx - frdmmcxn947_tflm_label_image/source/main.cpp - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
Resume (F8)
Project E... Registers Faults Periphera...
frdmmcxn947_tflm_label_image <Release>
Project Settings
Binaries
Includes
CMSIS
board
component
device
drivers
eiq
source
image
model
demo_config.h
demo_info.cpp
demo_info.h
labels.h
main.cpp
semihost_hardfault.c
timer.c
Debug
frdmmcxn947_tflm_label_image LinkServer Release [C/C++ (NXP Semicondu
frdmmcxn947_tflm_label_image.axf [MCXN947 (cortex-m33)]
Thread #1 1 (Suspended : Breakpoint)
main() at main.cpp:20 0x1df4
arm-none-eabi-gdb (12.1.90.20221210)
main() at main.cpp:20 0x1df4
main.cpp

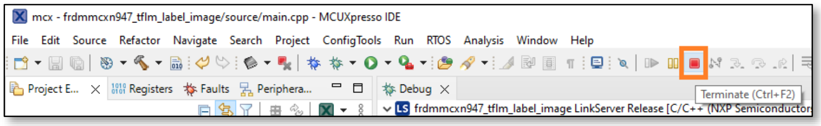
```
#include "board_init.h"
#include "demo_config.h"
#include "demo_info.h"
#include "fsl_debug_console.h"
#include "image.h"
#include "image_utils.h"
#include "model.h"
#include "output_postproc.h"
#include "timer.h"

int main(void)
{
    BOARD_Init();
    TIMER_Init();
```

31. Looks like there’s an error: **Didn't find op for builtin opcode 'CONV_2D'**

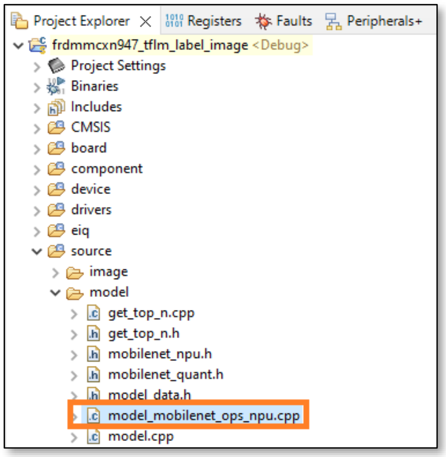


```
Label image example using a TensorFlow Lite Micro model.
Detection threshold: 23%
Model: mobilenet
Didn't find op for builtin opcode 'CONV_2D'
Failed to get registration from op code CONV_2D

AllocateTensors() failedFailed initializing model
```

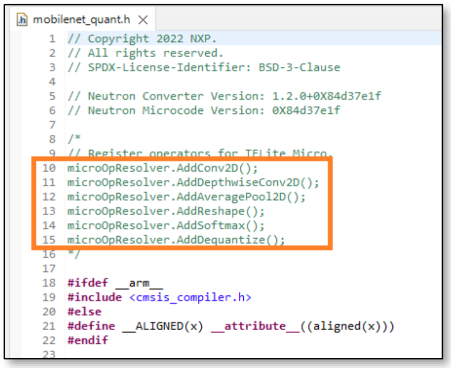
32. Stop the debugger by clicking on the red square


mcx - frdmmcxn947_tflm_label_image/source/main.cpp - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
Project E... Registers Faults Periphera...
Debug
Terminate (Ctrl+F2)
frdmmcxn947_tflm_label_image LinkServer Release [C/C++ (NXP Semiconductor

33. When changing models, the list of operators needs to be updated as well. To fix the error, go back to MCUXpresso IDE and open the **model_mobilenet_ops_npu.cpp** file.

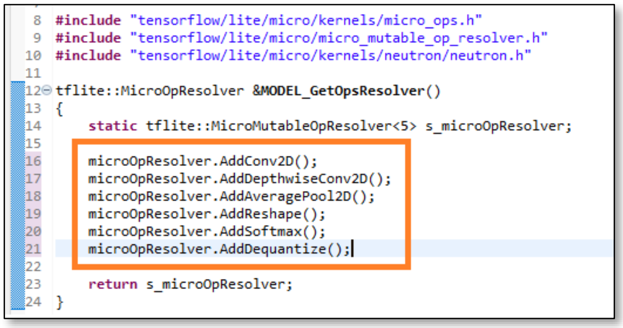


34. Inside the **MODEL_GetOpsResolver** function is a list of operators. If you open the **mobilenet_quant.h** header file you'll also find a list of operators used by the model in the comment block at the top of the file.



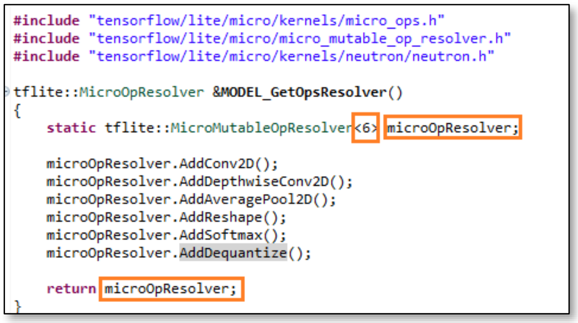
```
// Copyright 2022 NXP.
// All rights reserved.
// SPDX-License-Identifier: BSD-3-Clause

// Neutron Converter Version: 1.2.0+0X84d37e1f
// Neutron Microcode Version: 0X84d37e1f

/*
// Register operators for TFLite Micro.
microOpResolver.AddConv2D();
microOpResolver.AddDepthwiseConv2D();
microOpResolver.AddAveragePool2D();
microOpResolver.AddReshape();
microOpResolver.AddSoftmax();
microOpResolver.AddDequantize();
*/

#ifdef __arm__
#include <cmsis_compiler.h>
#else
#define __ALIGNED(x) __attribute__((aligned(x)))
#endif
```

35. Copy that list into the **MODEL_GetOpsResolver** function replacing the original list.



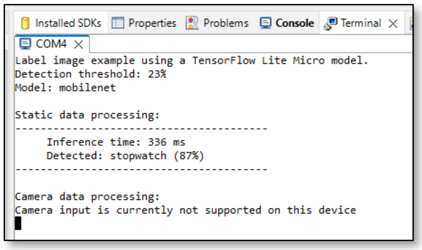
```
#include "tensorflow/lite/micro/kernels/micro_ops.h"
#include "tensorflow/lite/micro/micro_mutable_op_resolver.h"
#include "tensorflow/lite/micro/kernels/neutron/neutron.h"

tflite::MicroOpResolver &MODEL_GetOpsResolver()
{
    static tflite::MicroMutableOpResolver<5> s_microOpResolver;

    microOpResolver.AddConv2D();
    microOpResolver.AddDepthwiseConv2D();
    microOpResolver.AddAveragePool2D();
    microOpResolver.AddReshape();
    microOpResolver.AddSoftmax();
    microOpResolver.AddDequantize();

    return s_microOpResolver;
}
```

36. Also update the number of operators, the name of the operator variable (**s_microOpResolver** to **microOpResolver**), and the return variable name so they all use the **microOpResolver** name.



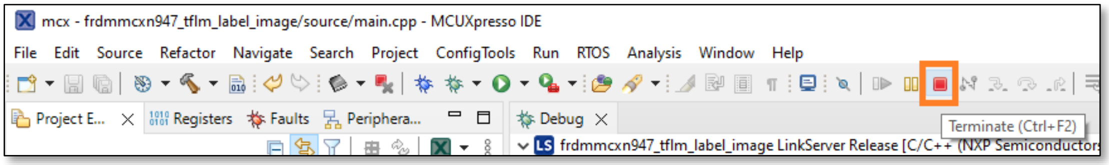
```
#include "tensorflow/lite/micro/kernels/micro_ops.h"
#include "tensorflow/lite/micro/micro_mutable_op_resolver.h"
#include "tensorflow/lite/micro/kernels/neutron/neutron.h"

tflite::MicroOpResolver &MODEL_GetOpsResolver()
{
    static tflite::MicroMutableOpResolver<6> microOpResolver;

    microOpResolver.AddConv2D();
    microOpResolver.AddDepthwiseConv2D();
    microOpResolver.AddAveragePool2D();
    microOpResolver.AddReshape();
    microOpResolver.AddSoftmax();
    microOpResolver.AddDequantize();

    return microOpResolver;
}
```

37. Recompile and reprogram the board using the previous steps.
38. You should now see the following on the serial terminal:

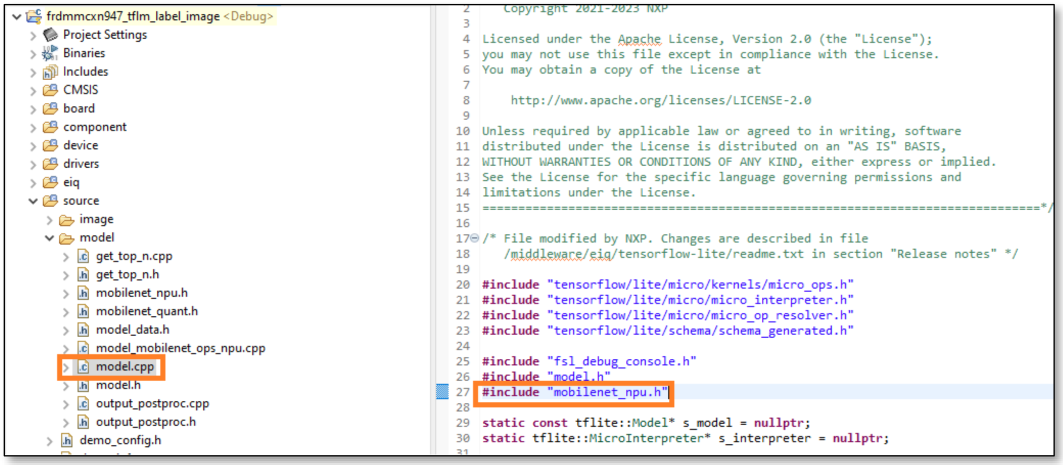


```
Label image example using a TensorFlow Lite Micro model.
Detection threshold: 23%
Model: mobilenet

Static data processing:
----------------------------------------
     Inference time: 336 ms
     Detected: stopwatch (87%)
----------------------------------------

Camera data processing:
Camera input is currently not supported on this device
```

39. Stop the debugger by clicking on the red square

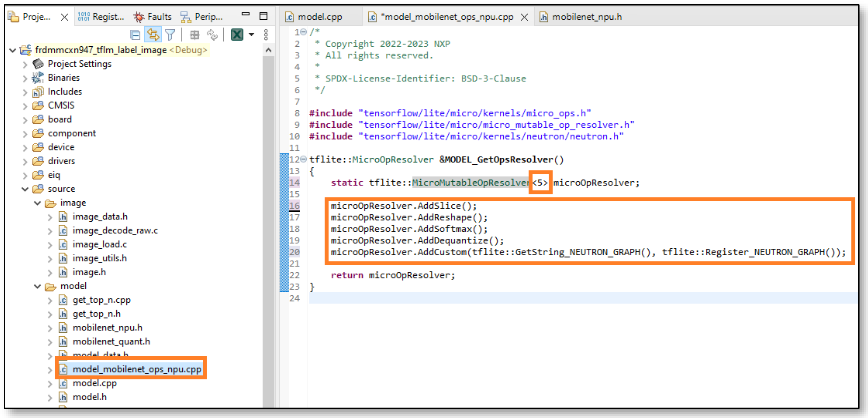

mcx - frdmmcxn947_tflm_label_image/source/main.cpp - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
Project E...
Registers
Faults
Periphera...
Debug
Terminate (Ctrl+F2)
frdmmcxn947_tflm_label_image LinkServer Release [C/C++ (NXP Semiconductors

40. Now run the program again, but this time with the Neutron NPU accelerated version of the model.
41. Re-open **model.cpp** and this time change line 27 to point to the Neutron NPU converted version of the model in the **mobilenet_ npu.h** file:



```
2     Copyright 2021-2023 NXP
3
4  Licensed under the Apache License, Version 2.0 (the "License");
5  you may not use this file except in compliance with the License.
6  You may obtain a copy of the License at
7
8      http://www.apache.org/licenses/LICENSE-2.0
9
10 Unless required by applicable law or agreed to in writing, software
11 distributed under the License is distributed on an "AS IS" BASIS,
12 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 See the License for the specific language governing permissions and
14 limitations under the License.
15 ==============================================================================*/
16
17 /* File modified by NXP. Changes are described in file
18    /middleware/eiq/tensorflow-lite/readme.txt in section "Release notes" */
19
20 #include "tensorflow/lite/micro/kernels/micro_ops.h"
21 #include "tensorflow/lite/micro/micro_interpreter.h"
22 #include "tensorflow/lite/micro/micro_op_resolver.h"
23 #include "tensorflow/lite/schema/schema_generated.h"
24
25 #include "fsl_debug_console.h"
26 #include "model.h"
27 #include "mobilenet_npu.h"
28
29 static const tflite::Model* s_model = nullptr;
30 static tflite::MicroInterpreter* s_interpreter = nullptr;
```

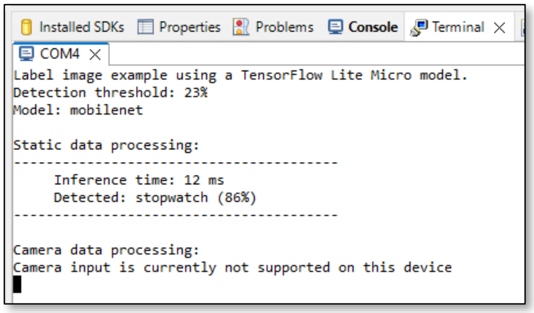
42. Re-open **model_mobilenet_ops_npu.cpp** and update the **MODEL_GetOpsResolver** function with the operators listed in the comment block of the **mobilenet_npu.h** file. Also make sure to update the array size.



```
/*
 * Copyright 2022-2023 NXP
 * All rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include "tensorflow/lite/micro/kernels/micro_ops.h"
#include "tensorflow/lite/micro/micro_mutable_op_resolver.h"
#include "tensorflow/lite/micro/kernels/neutron/neutron.h"

tflite::MicroOpResolver &MODEL_GetOpsResolver()
{
    static tflite::MicroMutableOpResolver<5> microOpResolver;

    microOpResolver.AddSlice();
    microOpResolver.AddReshape();
    microOpResolver.AddSoftmax();
    microOpResolver.AddDequantize();
    microOpResolver.AddCustom(tflite::GetString_NEUTRON_GRAPH(), tflite::Register_NEUTRON_GRAPH());

    return microOpResolver;
}
```

43. Build and program the program as before.

44. This time you should see the following on the terminal. That's over a 28x improvement in inference time, with the same confidence percentage on this static image.



```
Label image example using a TensorFlow Lite Micro model.
Detection threshold: 23%
Model: mobilenet

Static data processing:
----------------------------------------
     Inference time: 12 ms
     Detected: stopwatch (86%)
----------------------------------------

Camera data processing:
Camera input is currently not supported on this device
```

45. The decrease in inference time is very model dependent depending on how well that specific model could be optimized for the NPU.

4 Conclusion

This lab demonstrated how the eIQ Neutron NPU on MCX N devices can significantly decrease inference time on quantized models. These same steps can be used to benchmark other quantized models to see the performance improvements that the eIQ Neutron NPU can have.