eIQ™ Inference with Tensorflow Lite for Microcontrollers on i.MX RT685 -Without Camera

Lab Hand Out - Revision 2

NXP Semiconductors
www.nxp.com

Contents

1 Lab Overview

This lab is specific to the i.MX RT685 device which includes a HiFi4 DSP. This can significantly reduce the inference times when running a model. It does require compiling the TFLM SDK example with Xtensa Explorer IDE and then integrating the resulting binary into the MCUXpresso SDK project.

2 Prerequisites

2.1 RT685 Software Installation

1. Follow all the instructions on the MIMXRT685-EVK Getting Started website. Before continuing with this lab you should have:
    - Installed Xtensa Xplorer IDE
    - Updated LPC-Link2 firmware on the EVK to use the J-Link interface
    - Successfully debugged the **dsp_mu_polling_cm33** SDK example with both MCUXpresso IDE and Xtensa Xplorer IDE.

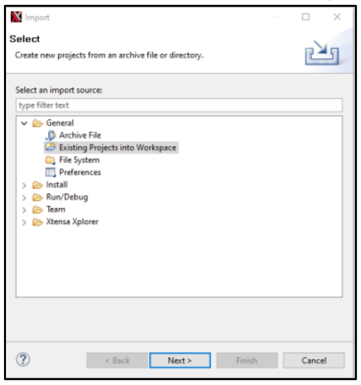
2.2 Create a model

1. Follow all the instructions in the **eIQ TensorFlow Lite for Microcontrollers Lab for RT1170 - Without Camera.pdf** lab until Section 5. You should have:
    - Retrained a model
    - Converted the model into a **flower_model.h** file
    - Converted an example image into a **daisy.h** file
    - Created a **flower_labels.h** file

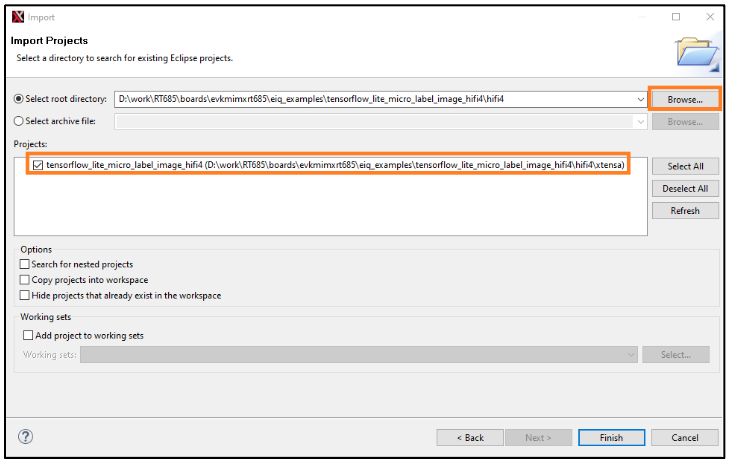
3 Create Updated HiFi4 Binary

Make sure the pre-requisites have been done in the previous section. The next step will be to take the files generated and the Xtensa Explorer project in the SDK and create an updated HiFi4 binary.

1. Unzip the i.MX RT685 SDK if not done already.
    - Make sure it is unzipped into a short filename path (ie C:\nxp\RT685), as an excessively long filename path can cause compile issues
    - Make sure the directory path contains no spaces
2. Open up Xtensa Xplorer.
3. Import the HiFi4 DSP for the TensorFlow for Microcontrollers (TFLM) project by going to File->Import
4. Expand the **General** category to select **Existing Projects into Workspace** and click on **Next**


Select
Create new projects from an archive file or directory.
Select an import source:

5. Click on **Browse** next to **Select root directory** and navigate to where you unzipped the RT685 SDK. Select the **\boards\evkmimxrt685\eiq_examples\tensorflow_lite_micro_label_image_hifi4\hifi4** directory which contains the HiFi4 DSP project for the TFLM Label Image demo. The **tensorflow_lite_micro_label_image_hifi4** project should then already be selected. Leave all the other options unchecked as-is. Click on **Finish** to import this project.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

(●) Select root directory: D:\work\RT685\boards\evkmimxrt685\eiq_examples\tensorflow_lite_micro_label_image_hifi4\hifi4

( ) Select archive file:

Projects:

[✓] tensorflow_lite_micro_label_image_hifi4 (D:\work\RT685\boards\evkmimxrt685\eiq_examples\tensorflow_lite_micro_label_image_hifi4\hifi4\xtensa)

Options

[ ] Search for nested projects

[ ] Copy projects into workspace

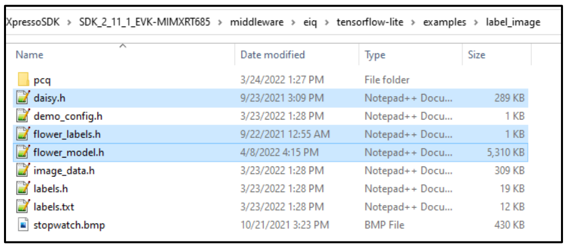
[ ] Hide projects that already exist in the workspace

Working sets

[ ] Add project to working sets

Working sets:

6. Now copy in the files generated from the previous section into **<SDK dir>\middleware\eiq\tensorflow-lite\examples\label_image**. It should look like the following when done:



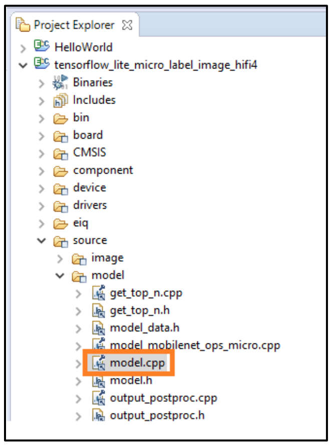
XpressoSDK › SDK_2_11_1_EVK-MIMXRT685 › middleware › eiq › tensorflow-lite › examples › label_image

Name	Date modified	Type	Size
pcq	3/24/2022 1:27 PM	File folder	
daisy.h	9/23/2021 3:09 PM	Notepad++ Docu...	289 KB
demo_config.h	3/23/2022 1:28 PM	Notepad++ Docu...	1 KB
flower_labels.h	9/22/2021 12:55 AM	Notepad++ Docu...	1 KB
flower_model.h	4/8/2022 4:15 PM	Notepad++ Docu...	5,310 KB
image_data.h	3/23/2022 1:28 PM	Notepad++ Docu...	309 KB
labels.h	3/23/2022 1:28 PM	Notepad++ Docu...	19 KB
labels.txt	3/23/2022 1:28 PM	Notepad++ Docu...	12 KB
stopwatch.bmp	10/21/2021 3:23 PM	BMP File	430 KB

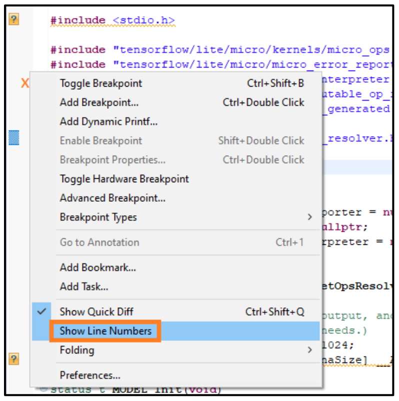
4 Modify Source Code

Now make the same edits as done in the i.MX RT1170 labs, except they'll be made in Xtensa Explorer instead of MCUXpresso IDE. These edits are:

1. Double click on the **model.cpp** file under the "source\model" folder in the Project View to open it.

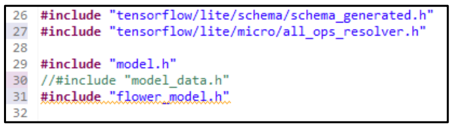


2. Right click on the blank space next to the code and select Show Line Numbers to display the line numbers


Toggle Breakpoint Ctrl+Shift+B
Add Breakpoint... Ctrl+Double Click
Add Dynamic Printf...
Enable Breakpoint Shift+Double Click
Breakpoint Properties... Ctrl+Double Click
Toggle Hardware Breakpoint
Advanced Breakpoint...
Breakpoint Types
Go to Annotation Ctrl+1
Add Bookmark...
Add Task...
Show Quick Diff Ctrl+Shift+Q
Show Line Numbers
Folding
Preferences...

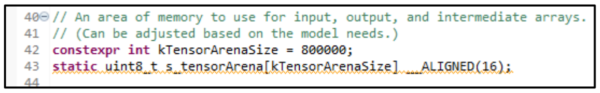
3. On line 27 add the following #include for the ops resolver that supports all the operands used by this retrained model:
#include "tensorflow/lite/micro/all_ops_resolver.h"

4. On line 30, comment out original #include for the original model defined in **model_data.h**. Then add a new #include to bring in the new model with **flower_model.h**. It should look like the following when finished:



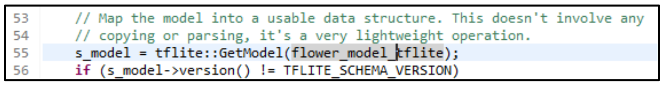
```
26 #include "tensorflow/lite/schema/schema_generated.h"
27 #include "tensorflow/lite/micro/all_ops_resolver.h"
28
29 #include "model.h"
30 //#include "model_data.h"
31 #include "flower_model.h"
32
```

5. On line 42, change the **kTensorArenaSize** variable to **800000**. This flower model is larger than the default example model, so it requires more memory space.



```
40 // An area of memory to use for input, output, and intermediate arrays.
41 // (Can be adjusted based on the model needs.)
42 constexpr int kTensorArenaSize = 800000;
43 static uint8_t s_tensorArena[kTensorArenaSize] __ALIGNED(16);
44
```

6. On line 55, change the model name to the array name in **flower_model.h**:

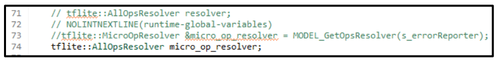


```
53     // Map the model into a usable data structure. This doesn't involve any
54     // copying or parsing, it's a very lightweight operation.
55     s_model = tflite::GetModel(flower_model_tflite);
56     if (s_model->version() != TFLITE_SCHEMA_VERSION)
```

7. To reduce the size of the project, the Label Image example only supports the specific operands required by the default Mobilenet model. Our retrained model uses a few new operands. These specific operands can be determined by analyzing the model with an application called **netron** and then manually add the operands as described in Section 7.1 of the eIQ TensorFlow Lite Library User's Guide. Or alternatively all the TFLite operands can be supported in a project by using the built in **tflite::AllOpsResolver** method. For this lab we'll use the latter method in order to provide the greatest compatibility with other models. On line 74 in model.cpp, comment out the original resolver line. Then add a new line

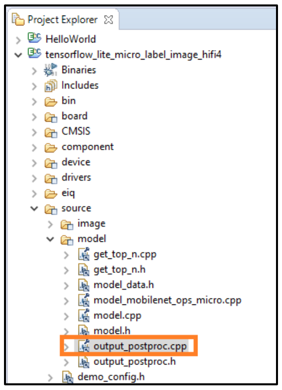
    tflite::AllOpsResolver micro_op_resolver;

It will look like the following when done:



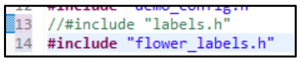
```
71     // tflite::AllOpsResolver resolver;
72     // NOLINTNEXTLINE(runtime-global-variables)
73     //tflite::MicroOpResolver &micro_op_resolver = MODEL_GetOpsResolver(s_errorReporter);
74     tflite::AllOpsResolver micro_op_resolver;
```

8. Next open the **output_postproc.cpp** file.



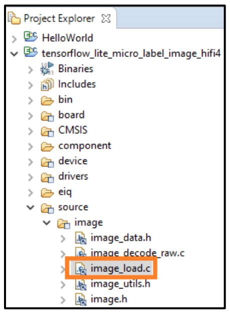
```
Project Explorer
> HelloWorld
v tensorflow_lite_micro_label_image_hifi4
  > Binaries
  > Includes
  > bin
  > board
  > CMSIS
  > component
  > device
  > drivers
  > eiq
  v source
    > image
    v model
      > get_top_n.cpp
      > get_top_n.h
      > model_data.h
      > model_mobilenet_ops_micro.cpp
      > model.cpp
      > model.h
      > output_postproc.cpp
      > output_postproc.h
    > demo_config.h
```

9. On line 13, comment out original #include for the original label file. Then add a new #include to bring in the new labels file. It should look like the following when finished:



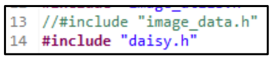
```
13 //#include "labels.h"
14 #include "flower_labels.h"
```

10. Next open the **image_load.c** file under **source\image**



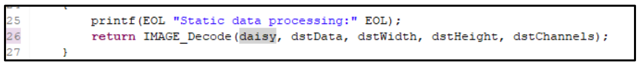
```
Project Explorer
> HelloWorld
v tensorflow_lite_micro_label_image_hifi4
  > Binaries
  > Includes
  > bin
  > board
  > CMSIS
  > component
  > device
  > drivers
  > eiq
  v source
    v image
      > image_data.h
      > image_decode_raw.c
      > image_load.c
      > image_utils.h
      > image.h
```

11. Make the following changes to bring in the daisy image for inferencing:
    a. Comment out the #include on line 13 for image_data.h and add **#include "daisy.h".**



```
13 //#include "image_data.h"
14 #include "daisy.h"
```

    b. Change line 26 for the IMAGE_Decode argument to the name of the daisy array, **daisy**



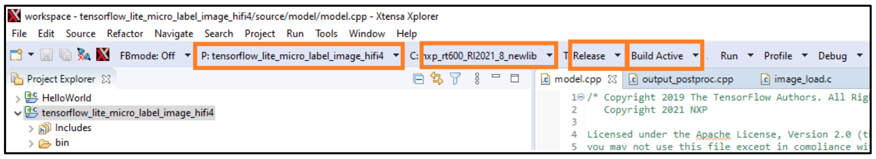
```
25        printf(EOL "Static data processing:" EOL);
26        return IMAGE_Decode(daisy, dstData, dstWidth, dstHeight, dstChannels);
27    }
```

5 Compile and Run

5.1 Compile HiFi4 Code

With all the necessary edits made, it's now time to generate the HiFi4 DSP binary.

7. In the toolbar, select the **tensorflow_lite_micro_label_image_hifi4** project, select the **nxp_rt600_RI2021_8_newlib** library, select the **Release** target, and then click on **Build Active** to build the neural network DSP library. You will see the results in the Console tab at the bottom. If there are any errors, make sure the SDK directory path does not have any spaces.



workspace - tensorflow_lite_micro_label_image_hifi4/source/model/model.cpp - Xtensa Xplorer

File Edit Source Refactor Navigate Search Project Run Tools Window Help

FBmode: Off | P: tensorflow_lite_micro_label_image_hifi4 | C: nxp_rt600_RI2021_8_newlib | T: Release | Build Active | Run | Profile | Debug

Project Explorer: HelloWorld; tensorflow_lite_micro_label_image_hifi4; Includes; bin

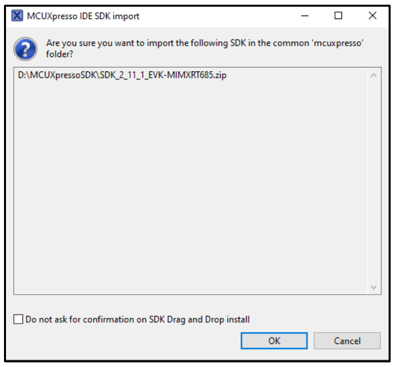
model.cpp | output_postproc.cpp | image_load.c

```
1 /* Copyright 2019 The TensorFlow Authors. All Rig
2    Copyright 2021 NXP
3
4 Licensed under the Apache License, Version 2.0 (t
5 you may not use this file except in compliance wi
```

8. When the compilation is complete, two .bin file will be generated in **<SDK_Path>\boards\evkmimxrt685\eiq_examples\tensorflow_lite_micro_label_image_hifi4\hifi4\binary**
9. These binary files will replace the default DSP library files in the MCUXpresso SDK project so make note of the location because it will be used in the next section.

5.2 Compile MCUXpresso IDE Code

1. Open up MCUXpresso IDE and select a new workspace
2. Install the RT685 SDK into the “Installed SDKs” tab by dragging-and-dropping the **RT685 SDK .zip** downloaded earlier into the Installed SDK window located in the bottom center. This dialog box will come up, and click OK to continue the import:

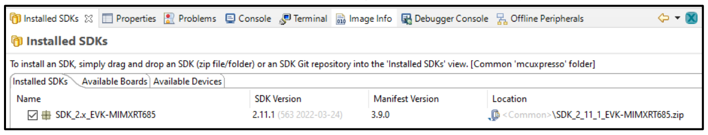

MCUXpresso IDE SDK import

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

D:\MCUXpressoSDK\SDK_2_11_1_EVK-MIMXRT685.zip

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

3. It will look like the following when complete:

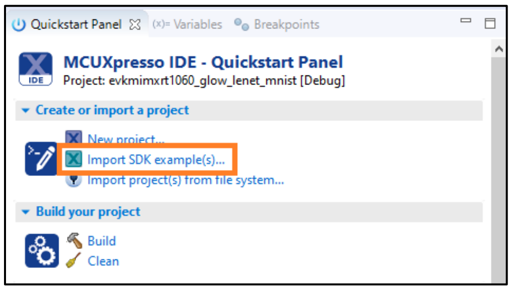


Installed SDKs

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]

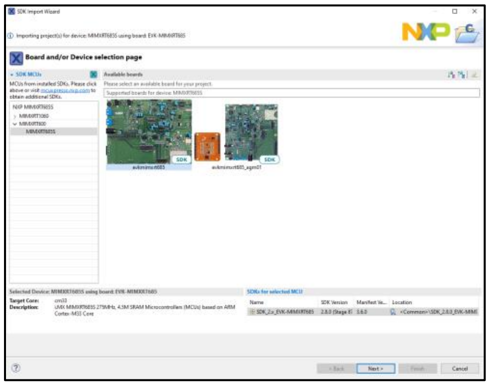
Name	SDK Version	Manifest Version	Location
☑ SDK_2.x_EVK-MIMXRT685	2.11.1 (563 2022-03-24)	3.9.0	<Common>\SDK_2_11_1_EVK-MIMXRT685.zip

4. In the Quickstart Panel in the lower left corner, click on Import SDK examples(s)…



Project: evkmimxrt1060_glow_lenet_mnist [Debug]

5. Select the **evkmimxrt685** board and click on Next



Importing project(s) for device: MIMXRT685S using board: EVK-MIMXRT685

Board and/or Device selection page

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

Please select an available board for your project.

Supported boards for device: MIMXRT685S

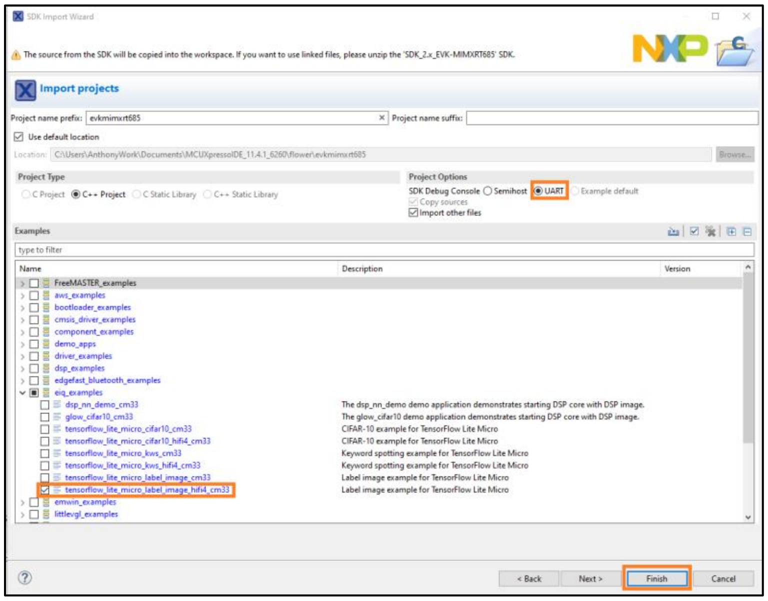
Selected Device: MIMXRT685S using board: EVK-MIMXRT685

Target Core: cm33
Description: i.MX MIMXRT685S 275MHz, 4.5M SRAM Microcontrollers (MCUs) based on ARM Cortex-M33 Core

SDKs for selected MCU

Name	SDK Version	Manifest Ve...	Location
SDK_2.x_EVK-MIMXRT685	2.8.0 (Stage 8)	3.6.0	<Common>\SDK_2.8.0_EVK-MIMI

6. Expand the **eIQ_examples** category and select the **tensorflow_lite_micro_label_image_hifi4_cm33** example. Make sure it's the HiFi4 version. Then make sure **UART** is selected for the SDK Debug Console option. Click on **Finish**.



SDK Import Wizard

The source from the SDK will be copied into the workspace. If you want to use linked files, please unzip the 'SDK_2.x_EVK-MIMXRT685' SDK.

Import projects

Project name prefix: evkmimxrt685

Project name suffix:

[✓] Use default location

Location: C:\Users\AnthonyWork\Documents\MCUXpressoIDE_11.4.1_6260\flower\evkmimxrt685

Project Type: C Project [✓ C++ Project] C Static Library C++ Static Library

Project Options: SDK Debug Console Semihost [✓ UART] Example default

[✓] Copy sources

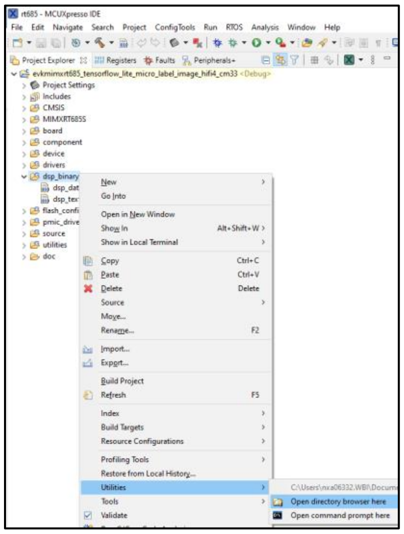
[✓] Import other files

Examples

type to filter

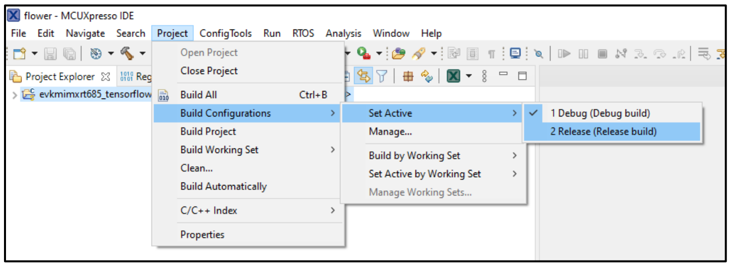
Name	Description	Version
FreeMASTER_examples		
aws_examples		
bootloader_examples		
cmsis_driver_examples		
component_examples		
demo_apps		
driver_examples		
dsp_examples		
edgefast_bluetooth_examples		
eiq_examples		
dsp_nn_demo_cm33	The dsp_nn_demo demo application demonstrates starting DSP core with DSP image.	
glow_cifar10_cm33	The glow_cifar10 demo application demonstrates starting DSP core with DSP image.	
tensorflow_lite_micro_cifar10_cm33	CIFAR-10 example for TensorFlow Lite Micro	
tensorflow_lite_micro_cifar10_hifi4_cm33	CIFAR-10 example for TensorFlow Lite Micro	
tensorflow_lite_micro_kws_cm33	Keyword spotting example for TensorFlow Lite Micro	
tensorflow_lite_micro_kws_hifi4_cm33	Keyword spotting example for TensorFlow Lite Micro	
tensorflow_lite_micro_label_image_cm33	Label image example for TensorFlow Lite Micro	
✓ tensorflow_lite_micro_label_image_hifi4_cm33	Label image example for TensorFlow Lite Micro	
emwin_examples		
littlevgl_examples		

7. This will copy the TLFM demo project to the MCUXpresso IDE workspace.
8. In the Project Explorer tab, right click on the **dsp_binary** folder and go to **Utilities->Open directory browser here** to open up a Windows Explorer at the directory where the default DSP binaries are located.

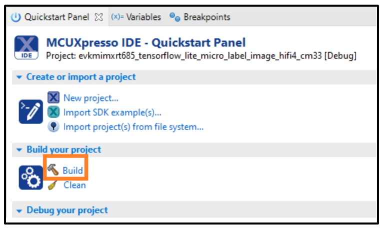


9. In that directory, copy and overwrite the default **dsp_data_release.bin** and **dsp_text_release.bin** files with the new DSP binary files that were generated in the previous section.

12. Change the build configuration to “Release” by clicking on the project and going to the menu bar and going to **Project->Build Configurations->Set Active->Release**. This will enables high compile optimizations.

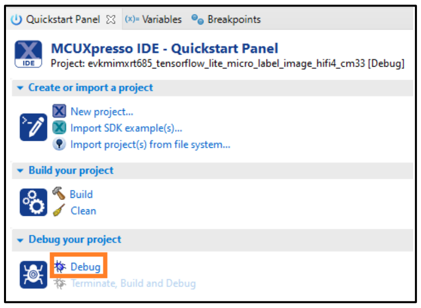


13. Build the project by clicking on “Build” in the Quickstart Panel and make sure there are no errors.

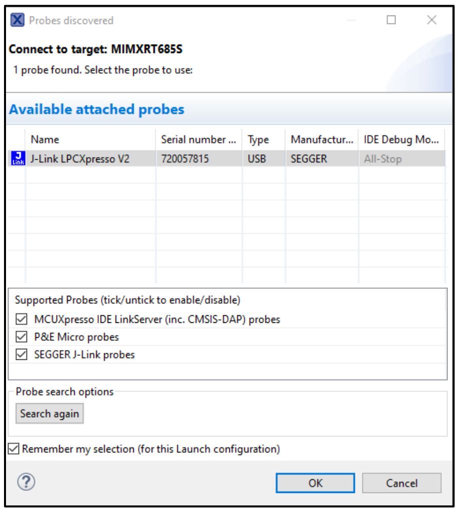

Quickstart Panel
Variables
Breakpoints
MCUXpresso IDE - Quickstart Panel
Project: evkmimxrt685_tensorflow_lite_micro_label_image_hifi4_cm33 [Debug]
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project

5.3 Run Example

14. Plug the micro-B USB cable into the board at J5 on the i.MX RT685 board.
15. Open TeraTerm or other terminal program, and connect to the COM port that the board enumerated as. Use 115200 baud, 1 stop bit, no parity.
16. Debug the project by clicking on “Debug” in the Quickstart Panel.


Quickstart Panel
Variables
Breakpoints
MCUXpresso IDE - Quickstart Panel
Project: evkmimxrt685_tensorflow_lite_micro_label_image_hifi4_cm33 [Debug]
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project
Debug
Terminate, Build and Debug

17. It will ask what interface to use. Select JLink.



Connect to target: MIMXRT685S

1 probe found. Select the probe to use:

Available attached probes

Name	Serial number ...	Type	Manufactur...	IDE Debug Mo...
J-Link LPCXpresso V2	720057815	USB	SEGGER	All-Stop

Supported Probes (tick/untick to enable/disable)

☑ MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes

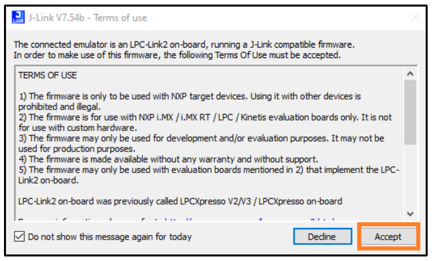
☑ P&E Micro probes

☑ SEGGER J-Link probes

Probe search options

☑ Remember my selection (for this Launch configuration)

18. You may get the following message. Put a checkmark on “Do not show this message again today” and then click on **Accept**.



J-Link V7.54b - Terms of use

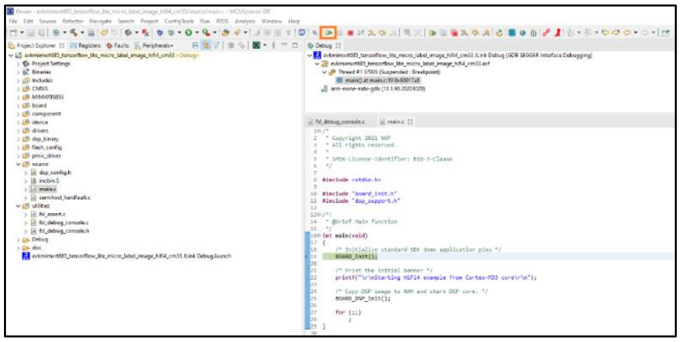
The connected emulator is an LPC-Link2 on-board, running a J-Link compatible firmware.
In order to make use of this firmware, the following Terms Of Use must be accepted.

TERMS OF USE

1) The firmware is only to be used with NXP target devices. Using it with other devices is prohibited and illegal.
2) The firmware is for use with NXP i.MX / i.MX RT / LPC / Kinetis evaluation boards only. It is not for use with custom hardware.
3) The firmware may only be used for development and/or evaluation purposes. It may not be used for production purposes.
4) The firmware is made available without any warranty and without support.
5) The firmware may only be used with evaluation boards mentioned in 2) that implement the LPC-Link2 on-board.

LPC-Link2 on-board was previously called LPCXpresso V2/V3 / LPCXpresso on-board

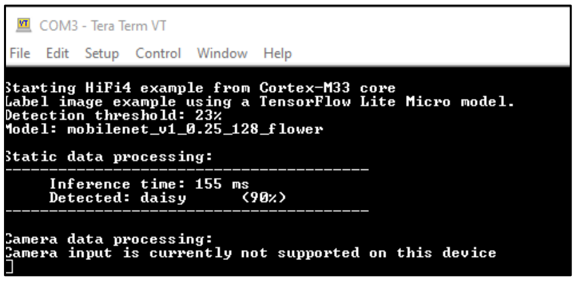
19. The debugger will download the firmware and open up the debug view. It may take some time to download the firmware. Click on the Resume button to start running.



```
evkmimxrt685_tensorflow_lite_micro_label_image_hifi4_cm33 JLink Debug [GDB SEGGER Interface Debugging]
  evkmimxrt685_tensorflow_lite_micro_label_image_hifi4_cm33.axf
    Thread #1 57005 (Suspended : Breakpoint)
      main() at main.c:19 0x80017a8
  arm-none-eabi-gdb (10.1.90.20201028)
```

```
/*
 * Copyright 2021 NXP
 * All rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include <stdio.h>

#include "board_init.h"
#include "dsp_support.h"

/*!
 * @brief Main function
 */
int main(void)
{
    /* Initialize standard SDK demo application pins */
    BOARD_Init();

    /* Print the initial banner */
    printf("\r\nStarting HiFi4 example from Cortex-M33 core\r\n");

    /* Copy DSP image to RAM and start DSP core. */
    BOARD_DSP_Init();

    for (;;)
        ;
}
```

20. Open up a terminal window, and you the result of the inference from the static image:



```
Starting HiFi4 example from Cortex-M33 core
Label image example using a TensorFlow Lite Micro model.
Detection threshold: 23%
Model: mobilenet_v1_0.25_128_flower

Static data processing:
----------------------------------------
     Inference time: 155 ms
     Detected: daisy       (90%)
----------------------------------------

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

6 Conclusion

This lab demonstrated how to use TensorFlow to generate a retrained TensorFlow model in TFLite format that can be imported and ran on an embedded system using the eIQ software platform.

This particular model was used to classify flower images. However, the model can also be trained on other types of images by re-running the script. Just add a new directory name and example images of that classification to the flower_photos directory, and new images can be recognized by this model.

Other types of TensorFlow models can be converted to TFLite format as well by using the tools included with TensorFlow or by using the eIQ Model Tool that is part of the eIQ Toolkit. The eIQ Toolkit can also be used to generate new image classification models in .tflite format that can be inferenced with TFLM. The model used in the Python script can also be changed to improve accuracy at the cost of inference time due to the increased complexity. It’s also important to note that the HiFi4 implementation does not support float32 so the model should be quantized before running on the HiFi4.

By enabling machine learning in embedded systems, there’s a wide world of opportunity for new smarter applications.