eIQ Toolkit™ for i.MX RT1170 DeepViewRT™ Getting Started

Revision 6
August 2023

Contents

1 Lab Overview

This document will cover how to use the Model Tool that is part of the eIQ Toolkit to convert Mobilenet v1, a common image classification model, to the DeepViewRT model format (RTM) and run it using the DeepViewRT inference engine available on i.MX RT devices. These same steps could be used for other TFLite models as well.

2 Hardware Requirements

This lab is written for the i.MX RT1170 evaluation board. This lab can also be used with the following evaluation boards that support DeepViewRT in the MCUXpresso SDK by downloading their respective MCUXpresso SDK packages:

- i.MX RT1050
- i.MX RT1060
- i.MX RT1064
- i.MX RT1160
- i.MX RT1170

3 Software Requirements

The following pieces of software are required:

- MCUXpresso IDE
- MCUXpresso SDK for the board being used
  - Make sure to unzip the package as some files inside of it will be used.
- eIQ Toolkit
- The Mobilenet_0.25_128 models that will be converted. This is the same model currently used in the eIQ TFLite and TFLite Micro examples in the SDK:
  - Mobilenet Floating Point
  - Mobilenet Quantized

4 Running Mobilenet v1 on RT1170

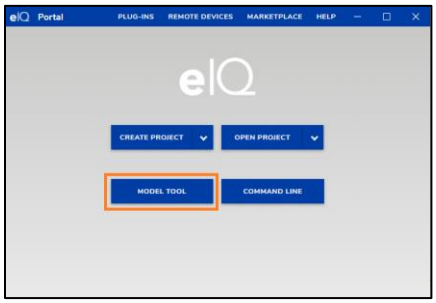
The default DeepViewRT inference example project already contains a "Label Image" type example project that uses a Mobilenet v1 224 model and does inferencing of a picture of a panda. This section will cover how to replace that model with your own custom model and image.

4.1 Converting Mobilenet to RTM Format

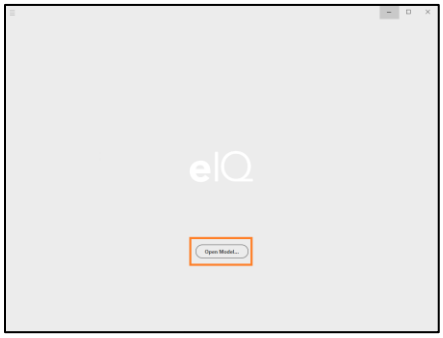
The first step is to convert a model into the DeepViewRT .RTM format. The eIQ Portal is used to do this.

1. Install **eIQ Toolkit**
2. Once installed, start the **eIQ Portal**

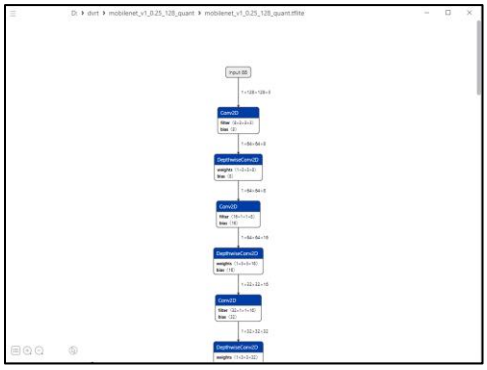
3. On the welcome screen, click on Model Tool



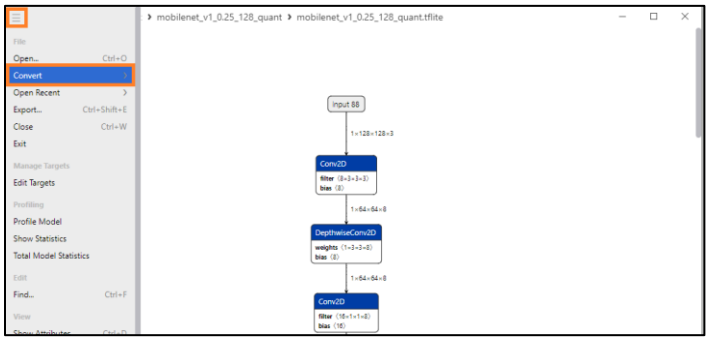
4. In the window that pops up, click on "Open Model.."



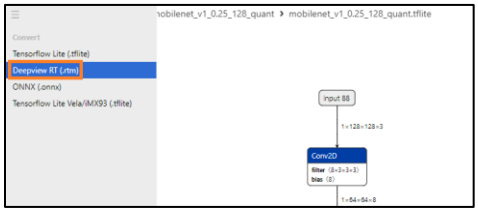
5. Select the Mobilenet model downloaded earlier. The format you need to pick depends on what you want to do:
    - If selecting the floating point version of Mobilenet and intend to run it as a floating point model on the RT1170, select the .tflite version: **mobilenet_v1_0.25_128.tflite**
    - If selecting the floating point version of Mobilenet and intend to quantize it using eIQ Portal, then select the .pb format: **mobilenet_v1_0.25_128_frozen.pb.** Note that you will need to first convert the .pb file to .tflite, and then can quantize the model when converting the .tflite file to the DeepViewRT .rtm format.
    - If selecting the quantized version of Mobilenet, use the .tflite version: **mobilenet_v1_0.25_128_quant.tflite**
6. For this example we'll use the quantized version so select the **mobilenet_v1_0.25_128_quant.tflite** file from the quantized Mobilenet package downloaded earlier.
7. It should look like the following.



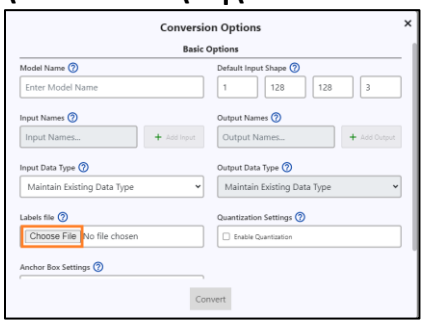
8. Click on the menu icon in the top left and select **Convert**



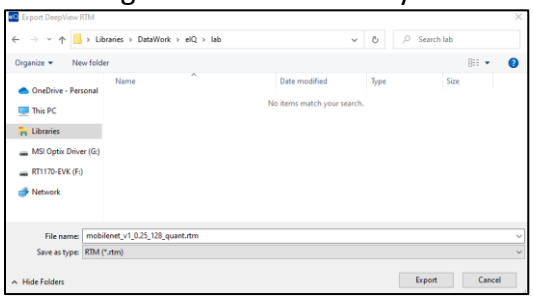
9. Then click on **Deepview RT (.rtm)**



10. In the dialog box that comes up, scroll down to the Labels field and click on Choose File.
11. This dialog box allows the user to include a labels file within the RTM format. Click on **Choose File** and the labels file for Mobilenet can be found in the MCUXpresso SDK at **\middleware\eiq\tensorflow-lite\examples\label_image\labels.txt**

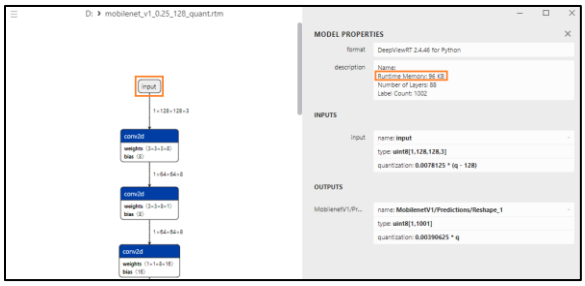


12. The rest of the fields can be left blank. Click on the Convert button at the bottom.
13. It should convert the model and a dialog box will come up asking where to save the .RTM file that was generated. Save it to your hard drive and make note of the location.



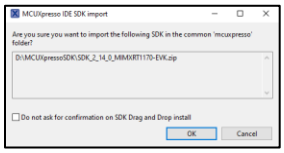
14. Now open that model file you just saved using the Model Tool

15. Click on the first layer, named **input**, and make note of the **Runtime Memory** as that value can be used later in the MCUXpresso SDK code to reduce the memory requirements.

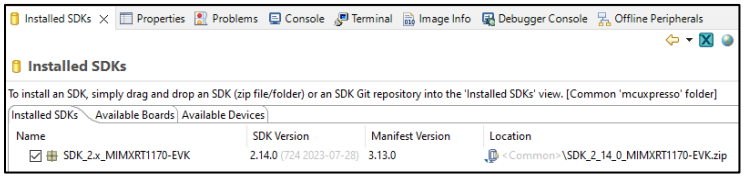


4.2 Import DeepViewRT Inference Example

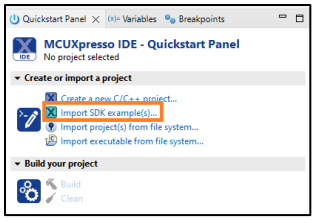
1. Open MCUXpresso IDE and select a workspace location in an empty directory.
2. Drag-and-drop the unzipped SDK folder into the Installed SDKs window, located on a tab at the bottom of the screen named “Installed SDKs”. You will get the following pop-up, so hit OK.



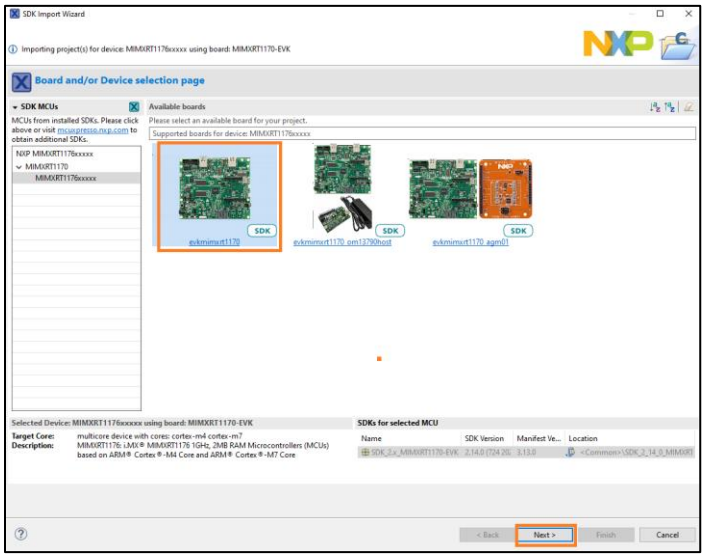
3. Once imported, the Installed SDK panel will look something like this



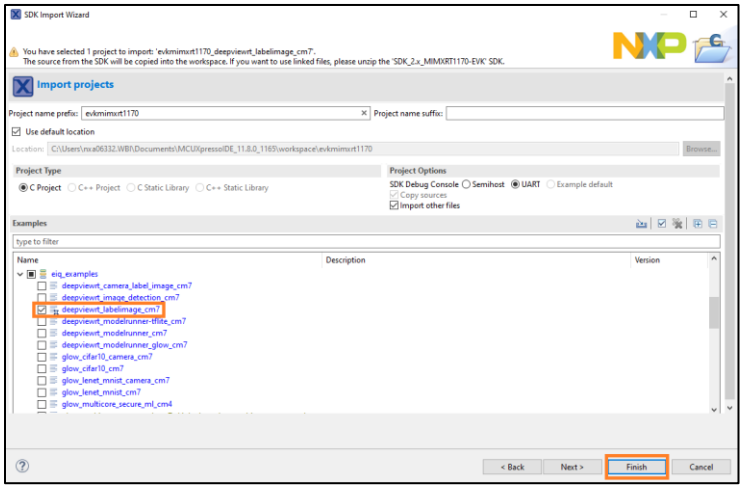
1. In the QuickStart menu, select **Import SDK example(s)…**



2. Select the RT1170-EVK (**evkmimxrt1170**) and click on **Next**



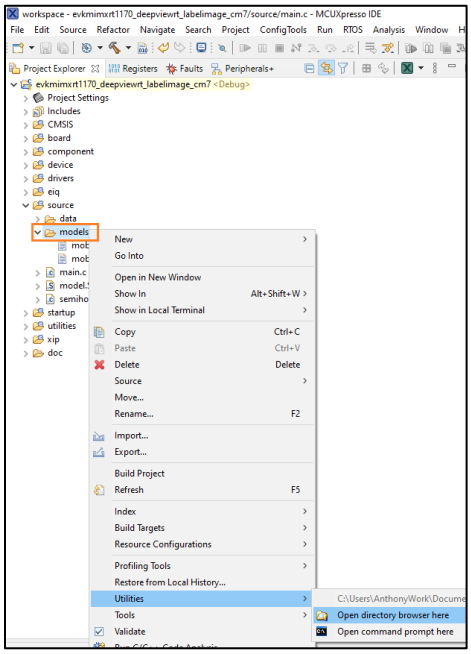
3. Expand the **eIQ** category and select the **deepviewrt_labelimage_cm7** example. Then click on the **Finish** button to import the project into your workspace.



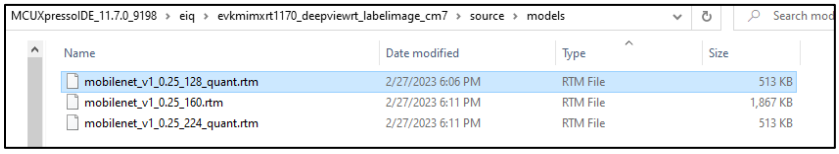
4.3 Modify the DeepViewRT Inference Example

Now we need to modify the default example to bring in the newly converted model from the previous section.

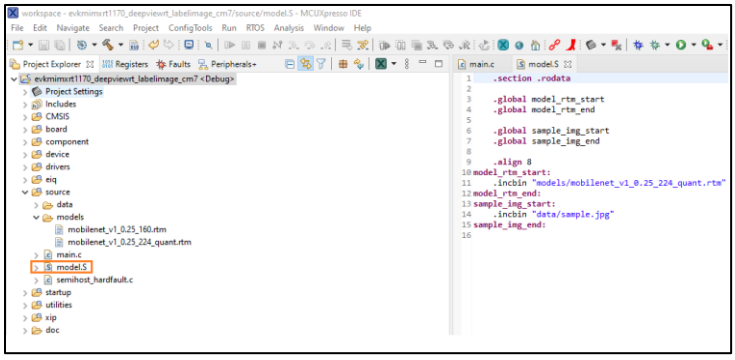
4. Open the Windows Directory folder that the models for this project are located in by expanding the Source->models folder in MCUXpresso IDE, right clicking on the models folder, and select **Utilities->Open directory browser here**



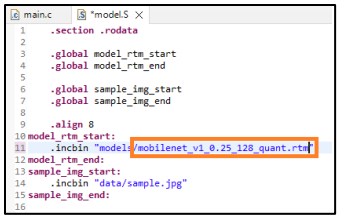
5. Inside the Windows Explorer directory that pops up, copy and paste the previously created .RTM file to this directory. Make sure to note of the name as it'll be used in a later step.



6. Go back to MCUXpresso IDE and double click on the **model.S** file to open it.



7. Change line 11 to specify the new model name:



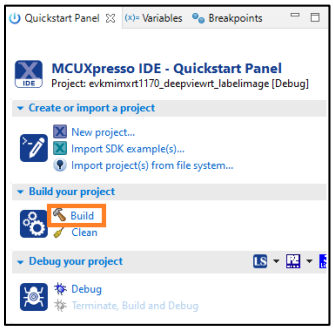
8. You could also choose to change the image to run the inference on by modifying line 14. The DeepViewRT project includes code to convert a JPG or PNG file so no script is necessary to pre-convert it. You could for example use the stopwatch image from the RT1170 SDK at **\middleware\eiq\tensorflow-lite\examples\image\label_image\stopwatch.bmp**, however you would need to use Paint or some other program to save it as a .jpg file first. The size will also be scaled appropriately so the image does not need to fit the model input size, though you may see better accuracy results with proper cropping as the image will may be less distorted.

   If you have the LCD screen for the i.MX RT1170 (or the board being used) then you can use the **deepviewrt_camera_label_image_cm7** SDK example to do inference on the camera data.

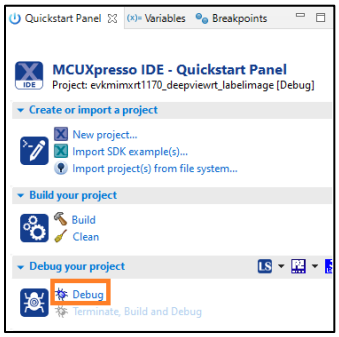
4.4 Run the DeepViewRT Inference Example

Now build and run the modified project

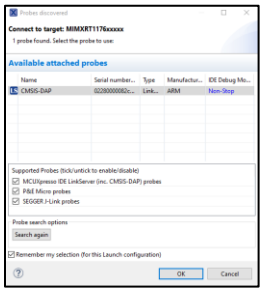
9. Connect a micro-B USB cable to the board on J11
10. Open TeraTerm or other terminal program, and connect to the COM port that the board enumerated as. Use 115200 baud, 1 stop bit, no parity as the terminal settings.
11. In the Quickstart panel, click on **Build**. You should see no errors in the Console tab.



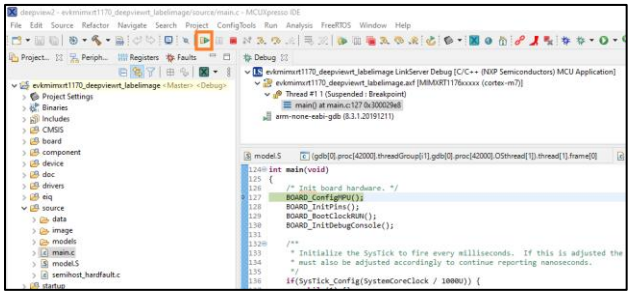
12. In the Quickstart pane select **Debug**



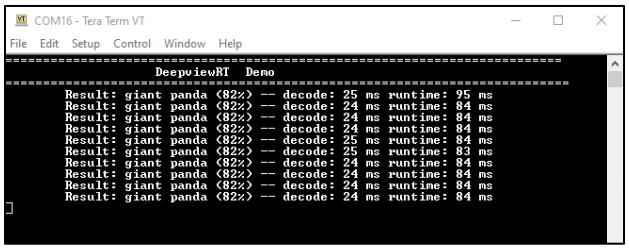
13. It will ask which interface to use. Select the CMSIS-DAP probe and click on OK.



14. The debugger will download the firmware and open up the debug view. Click on the Resume button to start running.



15. In the terminal you should see the inference results and inference time.



5 Optimizations

There are some options that can be modified in the project to optimize for a particular model.

1. By default the DeepViewRT example allocates 10MB of memory for the intermediate buffers the model may need. This can be reduced based on the estimated memory size found after converting the model in the **Runtime Memory** field from the conversion section of this lab. For this particular model, change **#define MEMPOOL_SIZE** on line 71 in **main.c** to be 120000. This is slightly above the number listed in the model analysis, as that is just an estimate.

```
59 /**
60  * The mempool holds the intermediate buffers for evaluating the
61  * model.  This buffer can be multiple megabytes in size and therefore
62  * should be stored in the SDRAM.  You may adjust this size if your
63  * particular model requires more or less memory as reported by the
64  * conversion tool.  If insufficient memory is provided an error will
65  * be reported by nn_context_model_load().
66  */
67 /**
68  *The maximum runtime memroy size,
69  *adjust it according model's runtime size and board's SDRAM size.
70  */
71 #define MEMPOOL_SIZE 120000
72
```

2. The model can be copied to RAM to improve performance by setting **#define LOAD_MODEL_TO_SDRAM** to 1 on line 45 in **main.c**. The performance increase that this change makes will be very dependent on the particular model being used.

```
41 /**
42  * If LOAD_MODEL_TO_SDRAM is set to 1 then the model will be
43  * copied from FLASH into SDRAM before being loaded.
44  */
45 #define LOAD_MODEL_TO_SDRAM 1
46
```

6 Conclusion

This lab demonstrated how to use the eIQ Portal to convert a TFLite model into the DeepViewRT .RTM format, and import that RTM model into an MCUXpresso SDK example project to run the DeepViewRT inference engine.