eIQ® MCU Vision Pipeline Lab on the FRDM MCXN947

Lab Overview

This document will cover how to run a face detection application using NXP's eIQ MCU Vision Pipeline and to run it on the FRDM MCXN947. Then we will show how we can change the model to run a different application.

The Lab will use VSCode with NXP MCUXpresso extension.

Software and Hardware Installation

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

Hardware

The FRDM MCXN947 is used in this lab.

NXP Software Installation

Install the GCC, Python, and Git dependencies using the MCUXpresso Installer tool

Get familiar with MCU Vision Pipeline

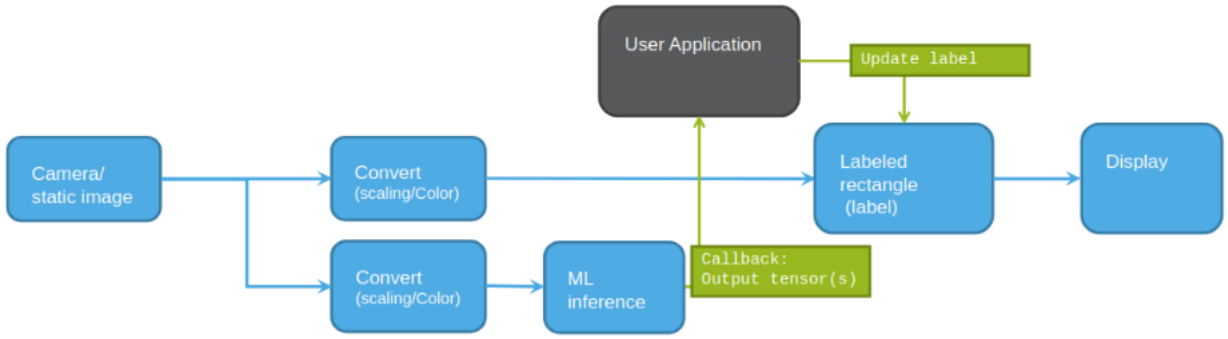
This section will show how to take an existing face detection example based on the MCU Vision pipeline library: MPP (Media Processing Pipeline) and run it on board. Then we will see how to modify this example to run a different application: person detection.

Running Face detection application:

Application creates two pipelines:

- One pipeline that runs the static image preview.
- Another pipeline that runs the ML inference on the static image source.
- Pipeline 1 is split from pipeline 0
- Pipeline 0 executes the processing of each element sequentially and CANNOT be pre-empted by another pipeline.
- Pipeline 1 executes the processing of each element sequentially, but CAN be pre-empted.

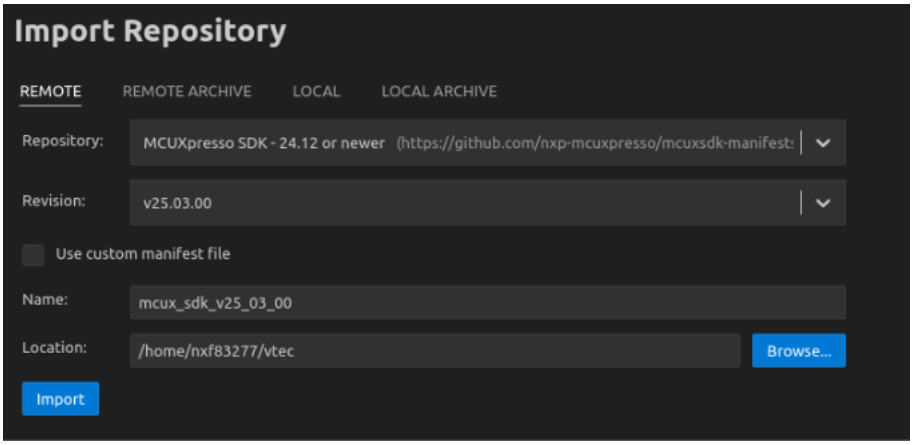
The following graph shows how the different elements are connected:



Now let's use the MCUXpresso SDK eIQ MPP face detection example to run the application

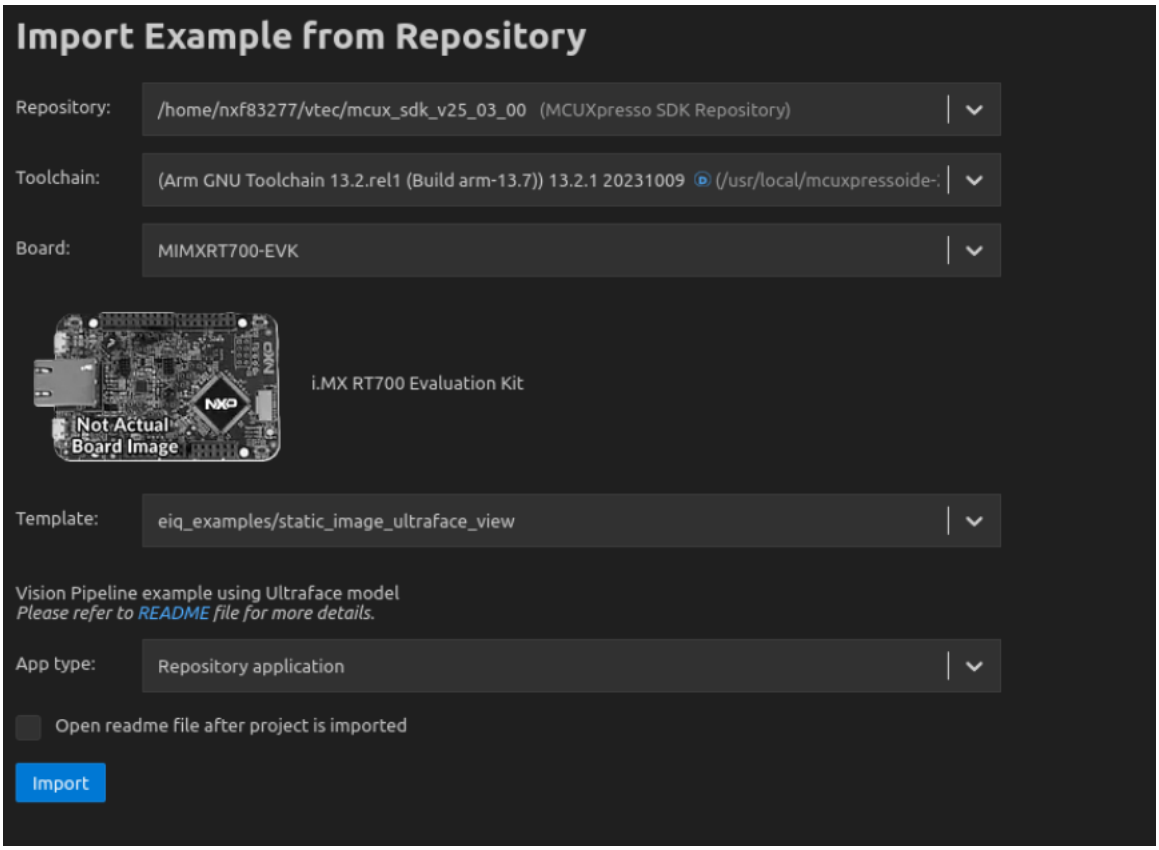
1. First, from the MCUXpresso for VSCode extension use the **Import Repository** tab. To clone the SDK Next git tree use:

https://github.com/nxp-mcuxpresso/mcuxsdk-manifests.git



Then click import, this may take a while to fetch all the repositories.

2. Import MPP face detection example(static_image_ultraface_view) from the repository.



Running person detection application:

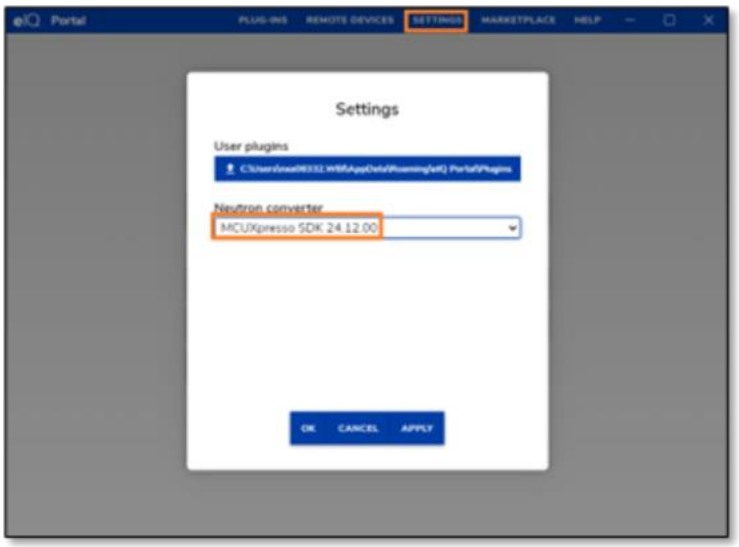
Now let's try to modify the model and the face detection example to create a person detection application using MPP.

Convert Models

Use eIQ Toolkit to convert a pre-existing Fastest-Det person detection model into a Neutron optimized model.

1. Download the Fastest-Det person detection TFLite model from the nxp model zoo following the recipe
2. Open eIQ Portal

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 is set to the correct SDK version (the one you imported earlier) 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.



4. Click on Model Tool

5. Click on Open Model and select the **fastest_det_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)**
9. In the dialog box that comes up, select **frdmmcxn** as the target and then in the custom options field put: **dump-header-file-output; dump-header-file-input; use-sequencer**

   These 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-output option though. It also directs the convertor to use sequencer mode which can result in faster inference times with the trade off of a larger model.. Click on **Convert** and save the converted TFLite file on your hard drive as **fastest_det_npu.tflite**

1. Alternatively the model can be converted from the command line by adding neutron-converter.exe found at **C:\NXP\eIQ_Toolkit_vxxx\bin\neutron-converter\MCU_SDK_xxxx** to your executable path.

   Then use the following command to do the conversion:

   neutron-converter --dump-header-file-output –dump-header-file-input --target frdmmcxn --use-sequencer --input fastest_det_quant.tflite --output fastest_det_npu.tflite

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 : **fastest_det _quant.tflite**

2. The Neutron converted file: **fastest_det _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 run on the board. Any layers that were not converted to a NeutronGraph operator will instead be run on cpu.

4. Take a look at the file size of each of the **.tflite** 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-output** argument generated the **.h** header file for the NPU optimized model that can be used in the eIQ MCUXpresso SDK projects.
6. The **dump-header-file-input** argument generated the **.h** 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 the person detection model.