



Application Software Pack: Multiple Person Detection with High-Efficient Neural Network on i.MX RT1060 and RT1170

Lab Hand Out - Revision 1

1 Summary

This lab covers how to deploy, customize and profile an ML model of object detection in NXP's microcontroller. The application gives a lightweight neural network (NN) model for multi-person detection, developed with an open-source NN structure shufflenetv2. The lab gives a unified Microcontroller based Vision Intelligence Algorithms (uVITA) System.

Contents

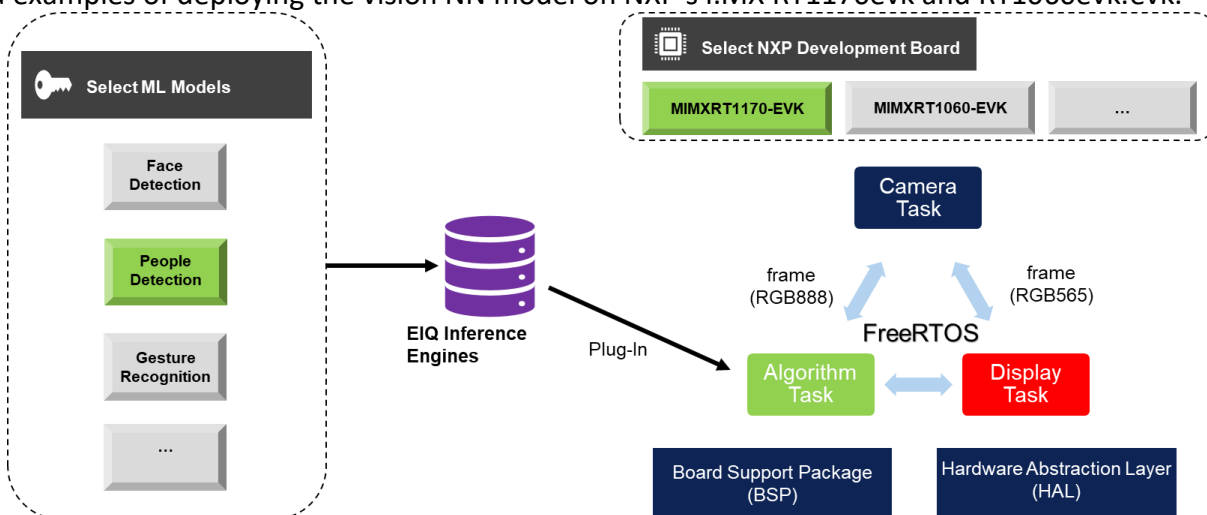
1 Summary.....	1
2 Lab Overview	3
2.1 Application Description	3
2.2 Equipment	3
2.3 Resources.....	3
3 Hardware and Software Installation	3
3.1 Hardware Requirements.....	4
3.2 Download Enablement Software.....	4
3.3 Python Related Package Installation	4
4 Import the App Software Pack Into MCUXpresso IDE.....	4
4.1 Option #1: Get the App Software Pack with MCUXpresso IDE	4
4.2 Option #2: Use the Command Line.....	5
4.3 Importing the Project into Your Workspace	7
5 Running the Application	8
5.1 Building, Flashing, and Running for i.MX RT1170 EVK	8
5.2 Building, Flashing, and Running for i.MX RT1060 EVK	11
6 Profiling the Application.....	12
6.1 Accuracy Verification	12
6.2 Footprint Profiling.....	13
6.3 Latency Profiling.....	14
7 Enabling the ML Person Detector with eIQ	14
7.1 Model profiling	15
7.2 Generate Glow bundle	15
7.3 Modify the model bundle.....	16
8 Conclusion.....	16

2 Lab Overview

The application in this lab shows the multiple person detection algorithms inferenced by EIQ tool on NXP's microcontrollers. This lab covers how to deploy, customize and profile a ML model of object detection in NXP's microcontrollers.

2.1 Application Description

The application first gives a lightweight neural network (NN) model for multi-person detection, developed with an open-source NN structure shufflenetv2. Then, it shows how to use EIQ Glow to quantize the ML model and convert it to executable codes. Besides, we also show how to analyze the performance of the quantized model in terms of accuracy, memory consumption, and inference time. Finally, the lab gives a unified **Microcontroller based Vision Intelligence Algorithms (uVITA)** System to build examples of deploying the vision NN model on NXP's i.MX RT1170evk and RT1060evk.evk.



2.2 Equipment

The following equipment is needed for this lab:

- [MIMXRT1170-EVK](#) or [MIMXRT1060-EVK](#)
- Micro-B USB cable
- J-Link debug probes

2.3 Resources

The following are helpful resources for this lab:

- Application Software Pack Repository
- [MCUXpresso IDE V11.6.0+](#)
- [Glow Installer for Windows](#)
- Application Note AN13924

3 Hardware and Software Installation

This section will cover the hardware configuration and steps needed to install the ML-person-detector Application Software Pack software.

3.1 Hardware Requirements

- i.MX RT1170 EVK board
 - ✧ The RT1170 EVK board, with camera module ov5640 and LCD panel [RK055HDMIPI4MA0](#), is needed to demonstrate the person detector in this application. Note that the camera module ov5640 is attached when the RT1170 EVK board is ordered as described in its [Hardware User's Guide](#).
- i.MX RT1060 EVK board
 - ✧ The camera module mt9m114 and LCD panel [RK043FN66HS-CTG](#), are needed if users want to demonstrate the person detector on [MIMXRT1060-EVK](#), which contains the camera module when ordered.

3.2 Download Enablement Software

1. Install the [MCUXpresso IDE](#)
2. Install [Git](#) and [west](#), and make sure your PATH environment variable contains the directory where pip installed west.exe.

3.3 Python Related Package Installation

Open as Administrator a Windows command prompt or [Git](#) terminal to verify the python command.

```
python -V
```

Note that the python 3 is required with packages of opencv and onnxruntime previously installed if users want to verify the proposed person detector on computer. To ensure software compatibility, users can follow the below version numbers

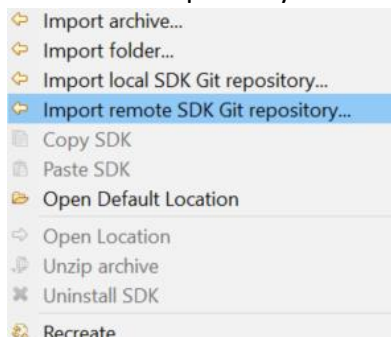
```
python -m pip install opencv-python==4.6.0.66
python -m pip install onnxruntime==1.12.1
python -m pip install numpy==1.21.6
```

4 Import the App Software Pack Into MCUXpresso IDE

There are two methods to get the application software pack. This section will cover both options, but only one needs to be done. It is recommended to use the first option as it is more straight forward.

4.1 Option #1: Get the App Software Pack with MCUXpresso IDE

1. Open MCUXpresso IDE and select a workspace location in an empty directory.
2. Right click in the blank area of the Installed SDKs panel at the bottom and select Import remote SDK Git repository...



3. In the dialog box that comes up:

- In the **Location** field, click on the **Browse** button and select or create an empty directory for the application software pack to be downloaded to. Make note of this location as it'll be used throughout this lab.
 - In the Repository field put: <https://github.com/nxp-mcuxpresso/appswpacks-ml-person-detector.git>
 - In the Revision field put: **mcux_release_github**
- Then hit **OK** to download the application software pack.

☒ Import Remote SDK Git

Import remote SDK Git repository

Location

Local folder where the files will be saved. Folder should be empty.

Git

Remote Git information

Repository

Revision

- Once imported the Installed SDKs tab will look like this:

Installed SDKs × Properties Problems Console Terminal Image Info Debugger Console Offline Perip

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. [Com

Installed SDKs Available Boards Available Devices

Name	SDK Version	Manifest Version
<input checked="" type="checkbox"/> # appswpacks-ml-person-detector (Git)	2.12.0 (34e07791715)	3.10.0
<input checked="" type="checkbox"/> # SDK_2.x_MIMXRT1060-EVK_APPSWPACKS_ML_PERSON_DETECTOR	2.12.0 (611 2022-07-1)	3.10.0
<input checked="" type="checkbox"/> # SDK_2.x_MIMXRT1170-EVK_APPSWPACKS_ML_PERSON_DETECTOR	2.12.0 (611 2022-07-1)	3.10.0

4.2 Option #2: Use the Command Line

- Open up a Windows command line and execute the following lines:

```

...
west init -m https://github.com/nxp-mcuxpresso/appswpacks-ml-person-detector.git --mr
mcux_release_github appswpacks-ml-person-detector
cd appswpacks-ml-person-detector
west update
...
  
```

```
nxf55124@NXL66715 MINGW64 ~/Desktop
$ west init -m https://github.com/nxp-mcuxpresso/appswpacks-ml-person-detector.git
--mr mcux_release_github appswpacks-ml-person-detector
=== Initializing in C:\Users\nxf55124\Desktop\appswpacks-ml-person-detector
--- Cloning manifest repository from https://github.com/nxp-mcuxpresso/appswpacks-ml-person-detector.git, rev. mcux_release_github
Cloning into 'C:\Users\nxf55124\Desktop\appswpacks-ml-person-detector\.west\manifest-tmp'...
remote: Enumerating objects: 383, done.
remote: Counting objects: 100% (383/383), done.
remote: Compressing objects: 100% (330/330), done.
remote: Total 383 (delta 48), reused 383 (delta 48), pack-reused 0
Receiving objects: 100% (383/383), 18.42 MiB | 215.00 KiB/s, done.
Resolving deltas: 100% (48/48), done.
--- setting manifest.path to examples
=== Initialized. Now run "west update" inside C:\Users\nxf55124\Desktop\appswpacks-ml-person-detector.

nxf55124@NXL66715 MINGW64 ~/Desktop
$ cd appswpacks-ml-person-detector/

nxf55124@NXL66715 MINGW64 ~/Desktop/appswpacks-ml-person-detector
$ west update
=== updating mcux-sdk (core):
--- mcux-sdk: initializing
Initialized empty Git repository in C:/Users/nxf55124/Desktop/appswpacks-ml-person-detector/core/.git/
--- mcux-sdk: fetching, need revision MCUX_2.11.0
```

2. Open MCUXpresso IDE and select a workspace location in an empty directory.
3. Drag-and-drop the **appswpacks-ml-person-detector** directory that was created in the previous step into the Installed SDKs window, located on a tab at the bottom of the screen named "Installed SDKs". You will get the following pop-ups, so hit **OK**.

☒ Import SDK Git

Import SDK Git

Location

Select location of the repository and the folder where the manifests are located

Repository location:

Manifest(s) folder:

4. Once imported, the installed SDKs panel will look something like this

Installed SDKs

Properties

Problems

Console

Terminal

Image Info

Debugger Console

Offline Per

Installed SDKs

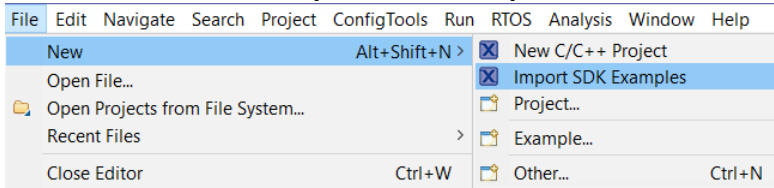
To install an SDK, simply drag and drop an SDK (zip file/folder) or an SDK Git repository into the 'Installed SDKs' view. [Co

Installed SDKs		Available Boards	Available Devices
Name	SDK Version	Manifest Version	
<div> <input checked="" type="checkbox"/> <div>appswpacks-ml-person-detector (Git)</div> </div>	2.12.0 (34e07791715)	3.10.0	
<div> <input checked="" type="checkbox"/> <div>SDK_2.x_MIMXRT1060-EVK_APPSWPACKS_ML_PERSON_DETECTOR</div> </div>	2.12.0 (611 2022-07-	3.10.0	
<div> <input checked="" type="checkbox"/> <div>SDK_2.x_MIMXRT1170-EVK_APPSWPACKS_ML_PERSON_DETECTOR</div> </div>	2.12.0 (611 2022-07-	3.10.0	

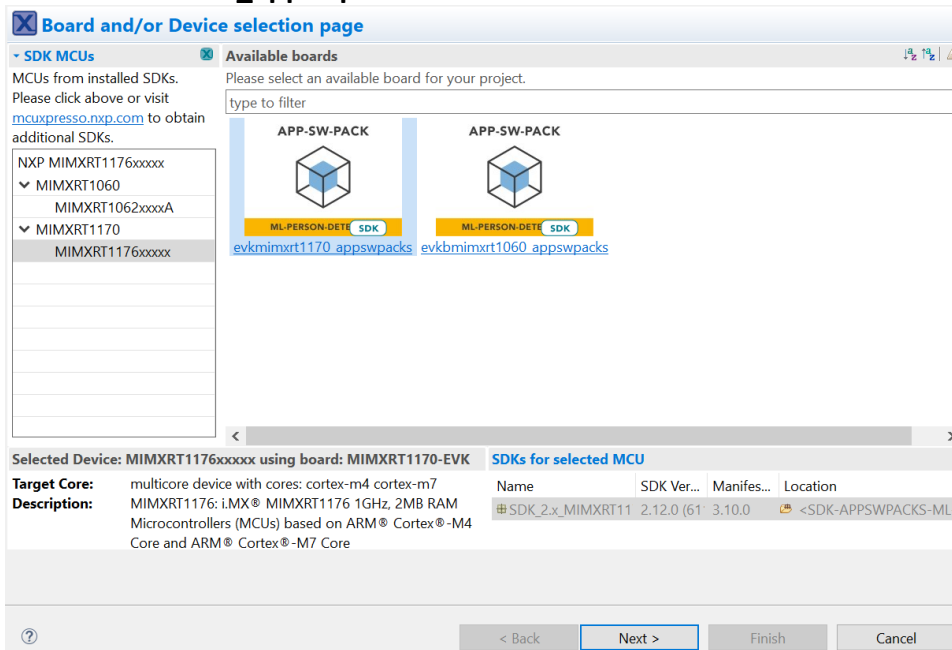
4.3 Importing the Project into Your Workspace

Once you finish importing the Software Pack into the Installed SDKs, it is time to import the project into your workspace.

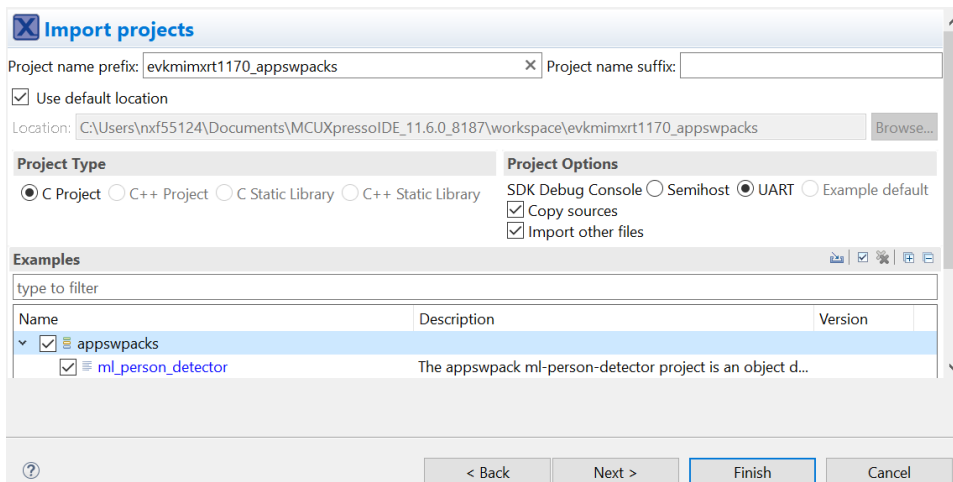
1. Go to **File -> New -> Import SDK Examples**.



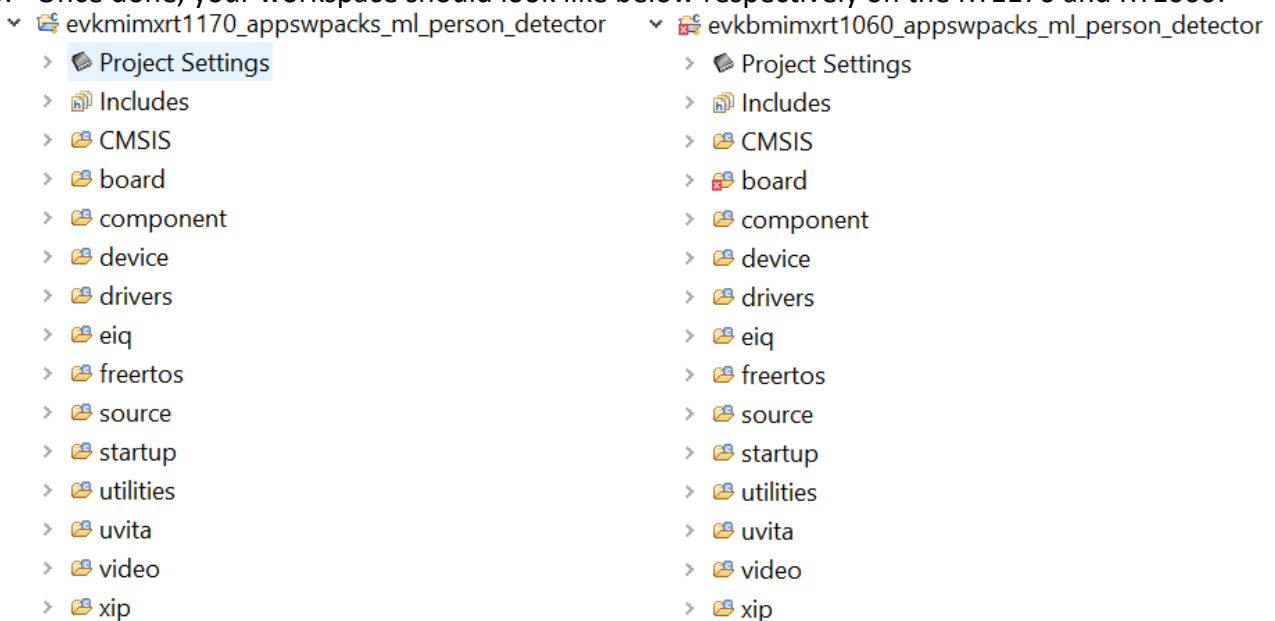
2. Once the SDK Import Wizard pops up, select **evkmimxrt1170_appswpacks** or **evkbmimxrt1060_appswpacks** and click **Next**.



3. Check the **appswpacks** checkbox.
4. Check or uncheck the **Copy sources** checkbox and click **Finish**.
 - a. If checked, MCUXpresso will make copies of all project files and place them in your selected workspace directory.
 - b. If unchecked, MCUXpresso will link the project files to the files in the repository that was cloned in the previous sections.



5. Once done, your workspace should look like below respectively on the RT1170 and RT1060.

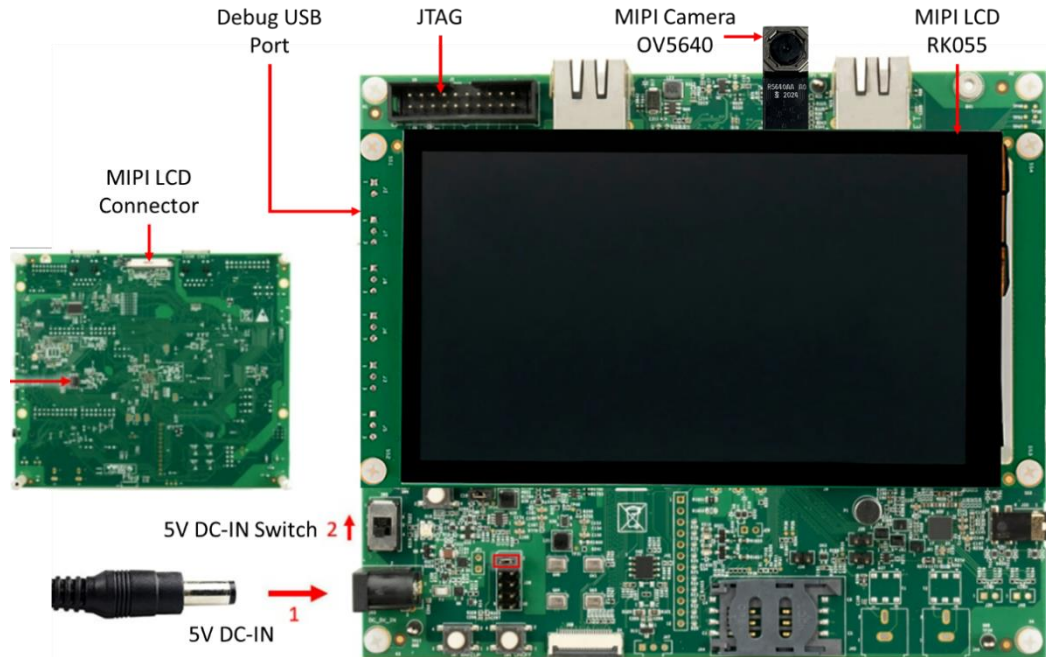


5 Running the Application

This section describes how to configure your EVK for the application and then walks you through the steps to build, flash, and run the application.

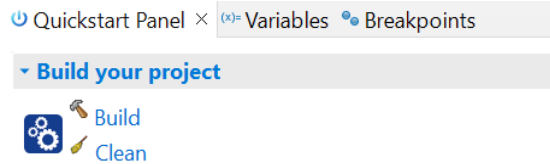
5.1 Building, Flashing, and Running for i.MX RT1170 EVK

Users can follow the user guide of [Getting Started with the i.MX RT1170 Evaluation Kit](#) to get familiar with the i.MX RT1170 EVK board. Note that the camera module ov5640 should be installed on the corresponding connector. Meanwhile, the LCD panel [RK055HDMIPI4MA0](#) should be installed on the MIPI LCD Connector of the EVK board.



Once you finish configuring the RT1170 EVK, you can build, flash, and run the application.

1. Build the project **evkmimxrt1170_appswpacks_ml_person_detector** by clicking **Build** in the Quickstart Panel.

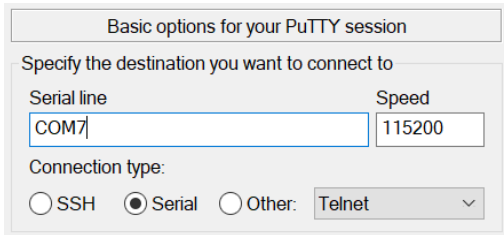


2. Once the build finishes, the output console should look like below.

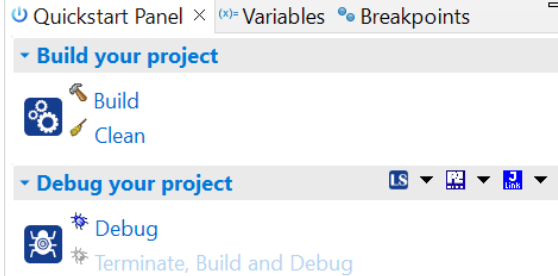
```
Building target: evkmimxrt1170_appswpacks_ml_person_detector.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -Xlinker -Map="evkmimxrt1170_appswpacks_ml_person_detector.map" -Xlinker
Memory region      Used Size  Region Size  %age Used
BOARD_FLASH:        397112 B      16 MB        2.37%
BOARD_SDRAM:        1599304 B     48 MB        3.18%
NCACHE_REGION:      12600 KB      16 MB       76.90%
SRAM_DTC_cm7:         0 GB       256 KB        0.00%
SRAM_ITC_cm7:         0 GB       256 KB        0.00%
SRAM_OC1:            0 GB       768 KB        0.00%
SRAM_OC_ECC1:         0 GB        64 KB        0.00%
SRAM_OC_ECC2:         0 GB        64 KB        0.00%
Finished building target: evkmimxrt1170_appswpacks_ml_person_detector.axf

Performing post-build steps
arm-none-eabi-size "evkmimxrt1170_appswpacks_ml_person_detector.axf"; # arm-none-eabi-objcopy -v -O l
text    data    bss    dec    hex filename
397112   0 14501512  14898624  e355c0 evkmimxrt1170_appswpacks_ml_person_detector.axf
```

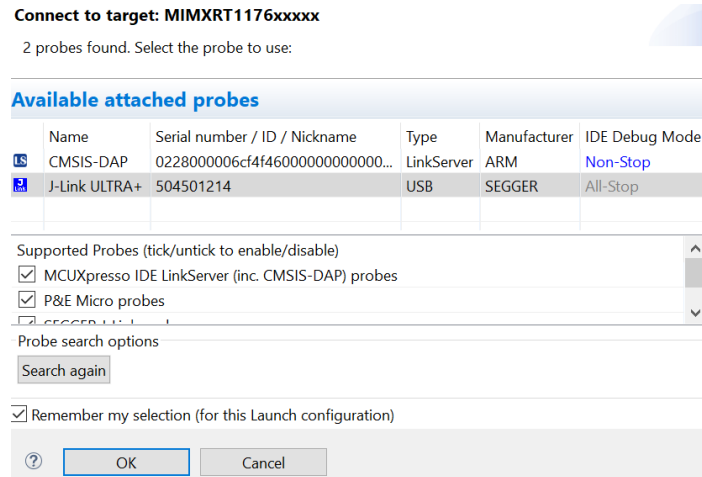
3. Connect the micro-B USB cable to **debug USB port J11** on the RT1170EVK.
4. Open Putty or a different terminal emulator and connect to the COM port belonging to the EVK. Note that the Serial Port connection speed should be **115200**.



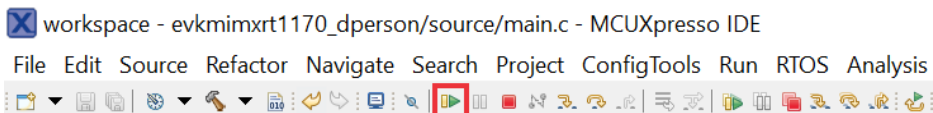
5. Debug the project by clicking on **Debug** in the Quickstart Panel.



6. It will ask what interface to use. Select the detected J-Link debug probe.



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



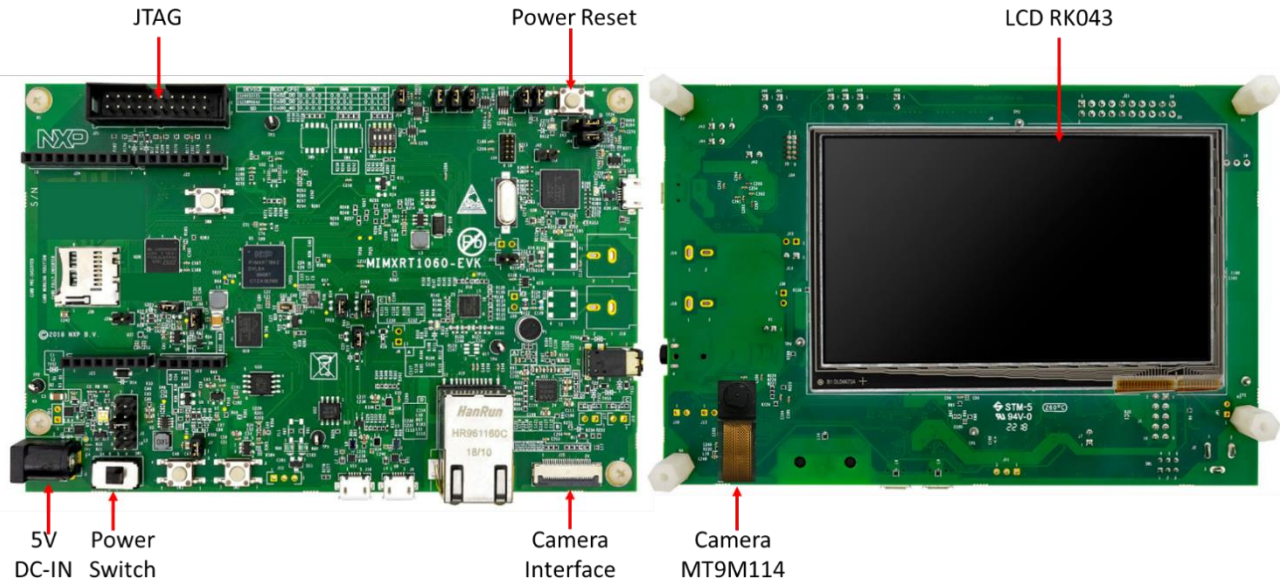
8. The following text should appear in the terminal program if there is one person in front of the camera.

```
[Bare-metal-multi-person-detection]: latency 230 ms
[multi-person-detection]: counting 0, latency 280 ms
[multi-person-detection]: counting 1, latency 280 ms
[multi-person-detection]: counting 1, latency 280 ms
[multi-person-detection]: counting 1, latency 280 ms
[multi-person-detection]: counting 1, latency 280 ms
[multi-person-detection]: counting 1, latency 280 ms
[multi-person-detection]: counting 1, latency 280 ms
[multi-person-detection]: counting 1, latency 280 ms
[multi-person-detection]: counting 1, latency 280 ms
[multi-person-detection]: counting 1, latency 281 ms
[multi-person-detection]: counting 1, latency 280 ms
[multi-person-detection]: counting 1, latency 280 ms
```

9. Meanwhile, the frame and the detection results will be shown on the LCD panel in real time.

5.2 Building, Flashing, and Running for i.MX RT1060 EVK

Users can also follow the user guide of [Quick Start Guide-MIMXRT060-EVK](#) to get familiar with the i.MX RT1060 EVK board. Note that the camera module mt9m114 and LCD panel [RK043FN66HS-CTG](#) should be installed on the corresponding connector.



1. Build the project **evkbmimxrt1060_appswpacks_ml_person_detector** by clicking **Build** in the Quickstart Panel.

Quickstart Panel × (x)= Variables Breakpoints

Build your project



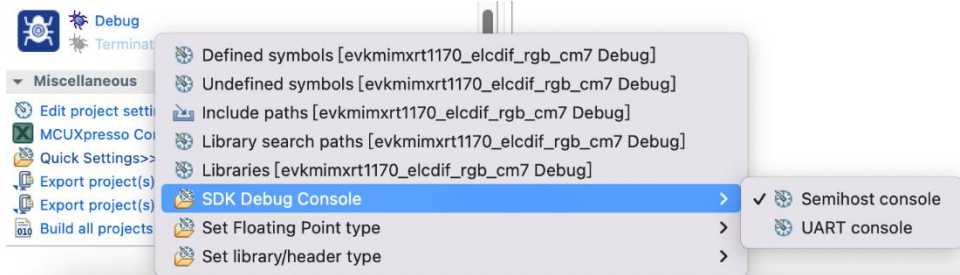
2. Once the build finishes, the output console should look like below

```
Building target: evkbmimxrt1060_appswpacks_ml_person_detector.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -Xlinker -Map="evkbmimxrt1060_appswpacks_ml_person_detector.map" -o
Memory region      Used Size  Region Size  %age Used
BOARD_FLASH:        382972 B      8 MB         4.57%
BOARD_SDRAM:        1598848 B     30 MB         5.08%
NCACHE_REGION:       1020 KB      2 MB         49.80%
SRAM_DTC:            0 GB        128 KB         0.00%
SRAM_ITC:            0 GB        128 KB         0.00%
SRAM_OC:            0 GB        768 KB         0.00%
Finished building target: evkbmimxrt1060_appswpacks_ml_person_detector.axf
```

Performing post-build steps

```
arm-none-eabi-size "evkbmimxrt1060_appswpacks_ml_person_detector.axf"; # arm-none-eabi-objcopy
text  data  bss  dec  hex filename
382856  116 2643200 3026172 2e2cfc evkbmimxrt1060_appswpacks_ml_person_detector.axf
```

3. For the RT1060EVK, the Serial Port was not enabled in the ml-person-detector project. However, users can use Semihost console by clicking **Quick Settings->SDK Debug Console->Semihost console**.



4. Debug the project with the same steps as described in steps 5 to 7 in section 5.2.
5. The logs should appear in the **Console** if there is one person in front of the camera. Meanwhile, the frame and the detection results will be shown in the LCD panel in real time.

```

Installed SDKs Properties Problems Console x Terminal Image Info Debugger Console
evkbmimxrt1060_appswpacks_ml_person_detector JLink Debug [GDB SEGGER Interface Debugging]
[Bare-metal-multi-person-detection]: latency 349 ms
[multi-person-detection]: counting 0, latency 364 ms
[multi-person-detection]: counting 1, latency 365 ms
[multi-person-detection]: counting 1, latency 364 ms
[multi-person-detection]: counting 1, latency 365 ms
[multi-person-detection]: counting 1, latency 364 ms
[multi-person-detection]: counting 1, latency 365 ms
[multi-person-detection]: counting 1, latency 364 ms
[multi-person-detection]: counting 1, latency 365 ms
[multi-person-detection]: counting 1, latency 364 ms
[multi-person-detection]: counting 1, latency 365 ms
[multi-person-detection]: counting 1, latency 364 ms

```

6 Profiling the Application

6.1 Accuracy Verification

The scripts written in Python are provided to help users verify the proposed ML model of the person detector on PC side. To verify the performance of the given model, the Python test scripts are provided in the *Scripts* folder. To use the verification tool, go to the *examples/ml_person_detector/scripts* folder and run the below commands.

```
python image_test.py
```

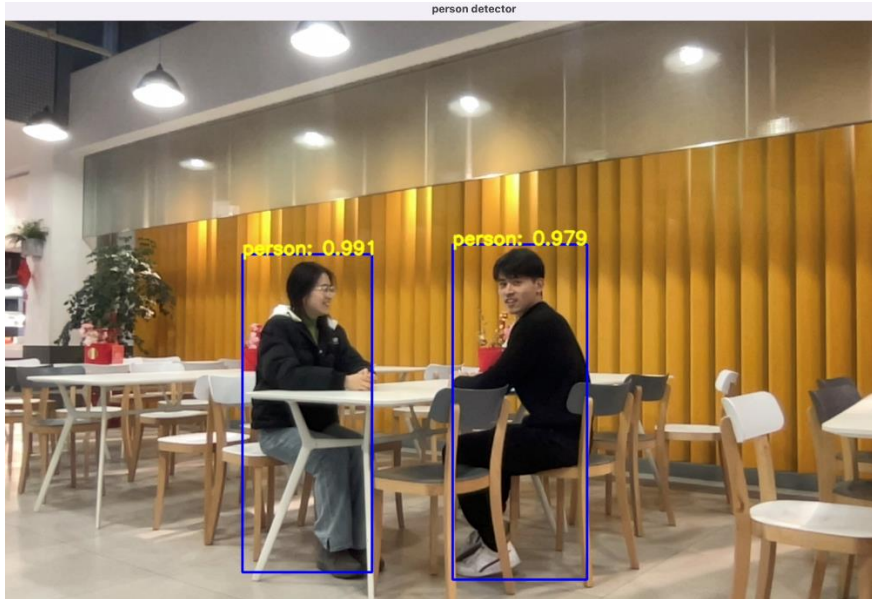
The test results of the original float model will come into being with four images samples.



Users can also have real-time experience through the video testing tools with below commands under the *examples/ml_person_detector/scripts* folder

```
python video_test.py
```

Then the living video window will come out with detection results as below



6.2 Footprint Profiling

The memory usage of the person detection model can be found in the header file **dperson_shufflenetv2.h**.

```
// Memory sizes (bytes).
#define DPERSON_SHUFFLENETV2_CONSTANT_MEM_SIZE 246848
#define DPERSON_SHUFFLENETV2_MUTABLE_MEM_SIZE 743040
#define DPERSON_SHUFFLENETV2_ACTIVATIONS_MEM_SIZE 645120
```

The three parts of the required memory are settled in different regions, which are realized in the **uvita_dperson.c**. Note that the constant weights can be read directly from RAM during inferencing, or from non-volatile Flash by adding a “const” in front of the weight. Adding a “const” will help decrease the amount of RAM required compared to having the weights copied to RAM. The trade-off is that the latency of the model inference will often be slower, but in a less obvious way for the proposed model.

```
GLOW_MEM_ALIGN(DPERSON_SHUFFLENETV2_MEM_ALIGN)
const uint8_t dperson_constantWeight[DPERSON_SHUFFLENETV2_CONSTANT_MEM_SIZE] = {
    #include "dperson_shufflenetv2.weights.txt"
};
```

The memory required for both the input and output data buffers are labelled as **dperson_mutableWeight**. The memory must be allocated in RAM.

```
// Statically allocate memory for mutable weights (model input/output data).
GLOW_MEM_ALIGN(DPERSON_SHUFFLENETV2_MEM_ALIGN)
uint8_t dperson_mutableWeight[DPERSON_SHUFFLENETV2_MUTABLE_MEM_SIZE];
```

The scratch memory, required for intermediate computations needed by the model, are named as **dperson_activations**. This buffer must be located in RAM. The default configuration is to locate the buffer in SDRAM, but it can be changed to OCRAM by adding “__attribute__((section(".data.\$SRAM_OC1"), aligned(64)))”. Note that the impact of locating the activation buffer from SDRAM vs OCRAM is very important and will be discussed in the section of the

latency profiling.

```
// Statically allocate memory for activations (model intermediate results).
GLOW_MEM_ALIGN(DPERSON_SHUFFLENETV2_MEM_ALIGN)
// __attribute__((section(".data.$SRAM_OC1"), aligned(64)))
uint8_t dperson_activations[DPERSON_SHUFFLENETV2_ACTIVATIONS_MEM_SIZE];
```

6.3 Latency Profiling

There are some ways to customize the application to help you best integrate the ML person detector into your own application. The most important one is to change allocation of the memory for activations from SDRAM to OCRAM. The following defines are found starting at line 29 in **uvita_dperson.c** on the i.MX RT1170 EVK. Note that if users try to give the latency profiling on the i.MX RT1060 EVK, the OCRAM section name should be ".data.\$SRAM_OC" instead of ".data.\$SRAM_OC1".

```
// Statically allocate memory for activations (model intermediate results).
GLOW_MEM_ALIGN(DPERSON_SHUFFLENETV2_MEM_ALIGN)
__attribute__((section(".data.$SRAM_OC1"), aligned(64))) // For i.MX RT1170
// __attribute__((section(".data.$SRAM_OC"), aligned(64))) // For i.MX RT1060
uint8_t dperson_activations[DPERSON_SHUFFLENETV2_ACTIVATIONS_MEM_SIZE];
```

After the memory allocation in OCRAM, you should re-build the project and run the project again. Then, you will find that the latency of the person detector significantly decreases from 280ms to 165ms on the i.MX RT1170 EVK, while decreases from 364ms to 255ms on the i.MX RT1060 EVK.

```
[Bare-metal-multi-person-detection]: latency 162 ms
[multi-person-detection]: counting 0, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
[multi-person-detection]: counting 1, latency 165 ms
```

7 Enabling the ML Person Detector with eIQ

To learn how eIQ was used for the provided compiled object files of the neural networks in this project, you need to download the Glow installer from [Glow Installer for Windows](#) and install it into the *examples/ml_person_detector/converter* folder. Then, there should be the *bin* folder under the *converter* folder.

appswpacks-ml-person-detector > examples > ml_person_detector > converter

<input type="checkbox"/> Name	Date modified	Type	Size
bin	2023/4/11 16:22	File folder	
doc	2023/4/11 16:22	File folder	
LA_OPT_NXP_Software_License.txt	2022/7/12 9:18	Text Document	43 KB
README.md	2023/4/11 15:25	MD File	1 KB
Software-Content-Register.txt	2022/5/24 0:32	Text Document	3 KB

Then, copy the *examples/ml_person_detector/models* folder and *examples/ml_person_detector/data* folder, and paste them under the *examples/ml_person_detector/converter/bin* folder.

appswpacks-ml-person-detector > examples > ml_person_detector

<input type="checkbox"/> Name	Date modified	Type	Size
app	2023/4/11 15:25	File folder	
converter	2023/4/11 16:22	File folder	
<input checked="" type="checkbox"/> data	2023/4/11 15:25	File folder	
images	2023/4/11 15:25	File folder	
<input checked="" type="checkbox"/> models	2023/4/11 15:25	File folder	
scripts	2023/4/11 15:25	File folder	

7.1 Model profiling

Glow uses profile guided quantization and running inference to extract statistics regarding possible numeric values of each tensor within the neural network. Images in PNG format with the same resolution as the input should be prepared in advance. You can go into the *bin* folder located where you selected the Glow installer path, and use command below to generate *yaml* profile:

```
image-classifier.exe -input-image-dir=data/Calibration -image-mode=0to1 -image-layout=NCHW -
image-channel-order=BGR -model=models/Onnx/dperson_shufflenetv2.onnx -model-input-
name=input.1 -dump-profile=models/Glow/dperson_shufflenetv2.yml
```

Then you will get a *dperson_shufflenetv2.yml* under *bin/models/Glow* folder. The *yaml* profile contains quantization information and will be used to generate the binary object file.

7.2 Generate Glow bundle

Bundle generation represents the model compilation to a binary object file (bundle). Bundle generation is performed using the model-compiler tool. There are usually two options. One is that the models, like the ones generated by Glow in the prior step, are converted into object files with the underlying codes from the original Glow. Another is that the object files are further optimized with underlying codes from CMSIS-NN. Have the command window open in the *bin* folder and then use the commands below to generate the bundle files.

- Compile a float32 model to an int8 bundle:

```
model-compiler.exe -model=models/Onnx/dperson_shufflenetv2.onnx -model-
input=input.1,float,[1,3,192,320] -emit-bundle=models/Glow/int8_bundle -backend=CPU -
target=arm -mcpu=cortex-m7 -float-abi=hard -load-profile=models/Glow/dperson_shufflenetv2.yml
-quantization-schema=symmetric_with_power2_scale -quantization-precision-bias=Int8
```

- Compile a float32 model to an int8 bundle with CMSIS-NN:





```
model-compiler.exe -model=models/Onnx/dperson_shufflenetv2.onnx -model-
input=input.1,float,[1,3,192,320] -emit-bundle=models/Glow/int8_cmsis_bundle -backend=CPU -
```



```
target=arm -mcpu=cortex-m7 -float-abi=hard -load-profile=models/Glow/dperson_shufflenetv2.yml
-quantization-schema=symmetric_with_power2_scale -quantization-precision-bias=Int8 -use-cmsis
```

Then, the glow bundle is derived from the output of the Glow compiler. There are 4 files that are generated into the directory specified by the `-emit-bundle`. These files are provided before but users can reproduce them with Glow bundle generation commands.

appswpacks-ml-person-detector > examples > ml_person_detector > converter > bin > models > Glow > int8_cmsis_bundle

<input type="checkbox"/> Name	Date modified	Type	Size
 dperson_shufflenetv2.h	2023/4/11 16:44	C Header Source F...	3 KB
 dperson_shufflenetv2.o	2023/4/11 16:44	O File	494 KB
 dperson_shufflenetv2.weights.bin	2023/4/11 16:44	BIN File	242 KB
 dperson_shufflenetv2.weights.txt	2023/4/11 16:44	Text Document	1,471 KB

The **dperson_shufflenetv2.h** file contains the memory usage and the inference function API. The **dperson_shufflenetv2.o** file is the object file that contains the compiled model code in the form of library. Generally, the size of the object file is larger than the flash size required by itself.

7.3 Modify the model bundle

Then, users can add the newly generated model bundle files to the project under the *uvita/model* folder. The files included from the default project are based on the int8 bundle with CMSIS-NN. Users can also drop in newly generated int8 bundle with or without CMSIS-NN and observe the different performance. For instance, if the int8 bundle without CMSIS-NN is loaded on the i.MX RT1170 and RT1060, their expected latency should be 537ms and 789ms respectively.

```

v > uvita
  > algorithm
  > camera
  > display
  > include
  > message
  v model
    > dperson_shufflenetv2.h
    > dperson_shufflenetv2.o
    > dperson_shufflenetv2.weights.bin
    > dperson_shufflenetv2.weights.txt
  > presentation

```

8 Conclusion

This lab demonstrated how to enable a ML person detector on i.MX RT1170 EVK and i.MX RT1060 EVK using eIQ tools.