eIQ: Transfer Learning Using IMX RT1060 / IMX RT1050 EVK – Without Camera

#### **Contents:**

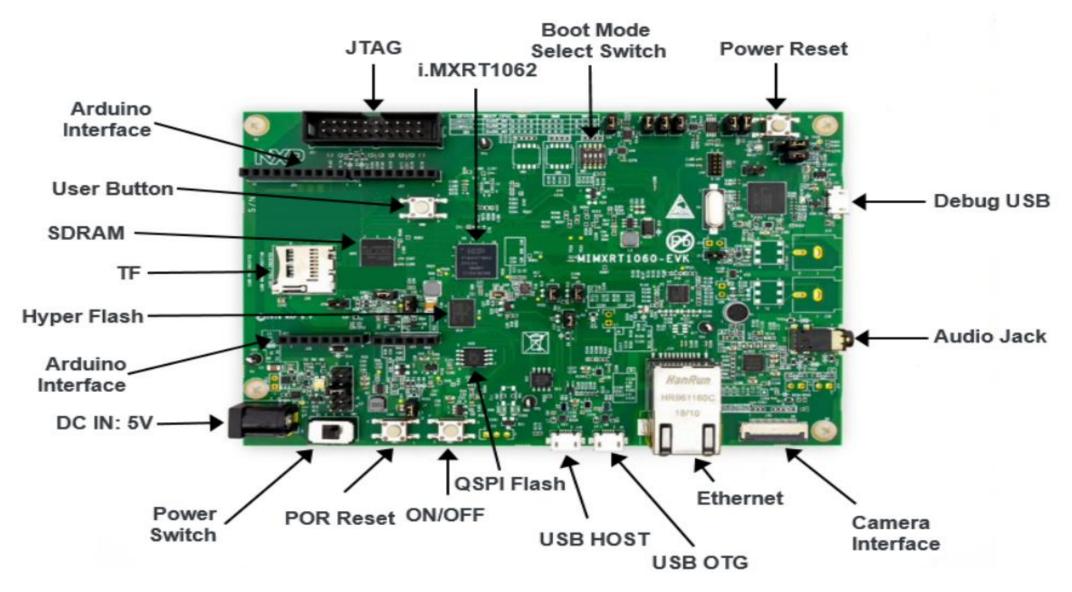
- Lab Overview
- IMX RT1060-EVK
- IMX RT1050-EVK
- Software and Hardware Installation
- Lab Scripts Installation
- Retrain Existing Model
- Convert Model and Data
- Run Demo
- Conclusion

#### Lab Overview:

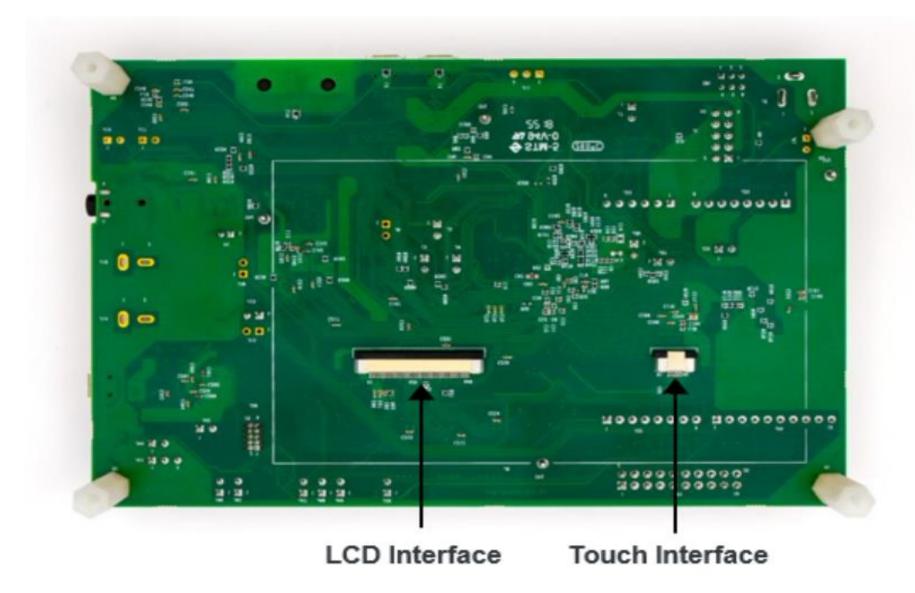
- This lab will cover how to take an existing TensorFlow image classification model, and re-train it to categorize images of flowers. This is known as transfer learning. This updated model will then be converted into a TensorFlow Lite file. By using that file with the TensorFlow Lite inference engine that is part of NXPs eIQ package, the model can be run on an i.MX RT embedded device.
- This lab is used without a camera + LCD, but the flower image will need to be converted to a C array and loaded at compile time.
- This lab is written for the RT1060-EVK. It can also be used with the RT1050-EVKB and RT1064-EVK with minor modifications. Also note that the RT1060-EVK and RT1064-EVK come with a camera sensor. The RT1050-EVKB does not come with a camera sensor. In all cases the LCD must be purchased separately.

# **IMX RT1060-EVK**

#### **Overview of the MIMXRT1060 EVK board (Front side):**

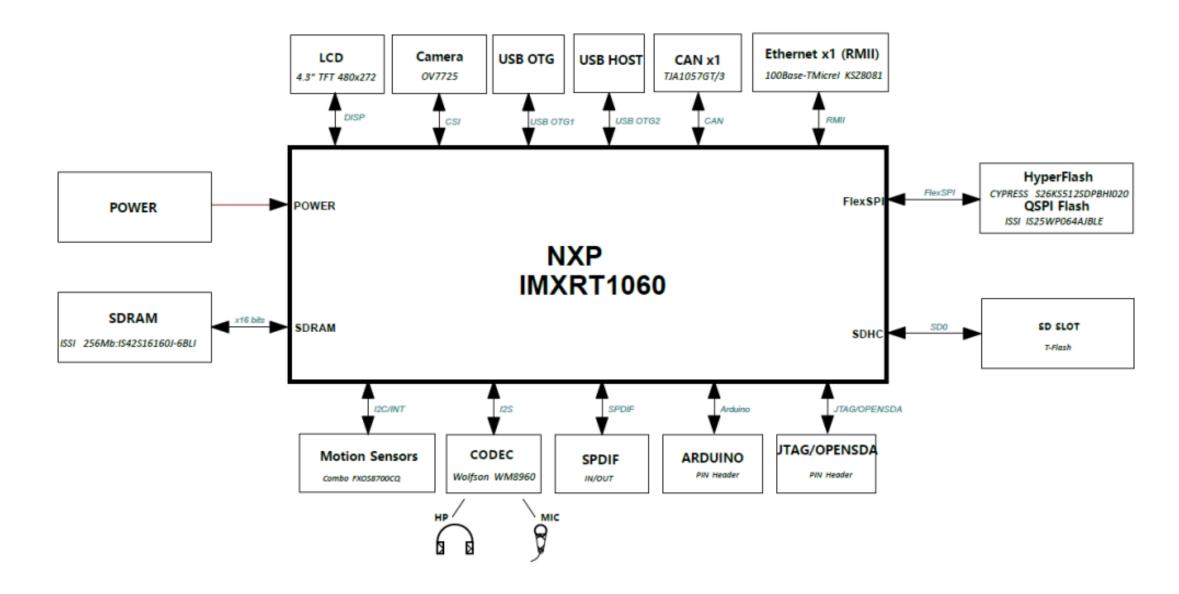


#### **Overview of the MIMXRT1060 EVK board (Back side):**



#### **Block diagram:**

-----



#### **Board Features:**

Processor	NXP Processor	MIMXRT10 6 2DVL6A	
DRAM Memory	SDRAM 256 Mbit, 166 MHz	IS42S16160J-6BLI	
DCDC	MPS MP2144GJ		
LDO	UNION	UM1550S-18 UM1750S-00	
Mass Storage	TF Card Slot		
	64 Mbit Quad SPI flash		
	512 Mbit Hyper flash		
Display Interface	LCD connector		
Ethernet	10/100 Mbit/s Ethernet connector. PHY chip: KSZ8081RNB		
USB	USB 2.0 OTG connector		
	USB 2.0 host connector		

Audio Connector	3.5 mm audio stereo headphone jack		
	Board-mounted microphone		
	Left and right speaker out connectors		
	S/PDIF interface (unpopulated )		
Power Connector	5 V DC-jack		
Debug Connector	JTAG 20-pin connector (SWD by default)		
	OpenSDA with DAP-Link		
Sensor	FXOS8700CQ: 6-Axis Ecompass (3-Axis Mag, 3- Axis Accel)		
	(Some boards are unpopulated)		
Camera	CMOS sensor interface		
CAN	CAN bus connector		
User Interface Button	ON/OFF, POR Reset, Reset, USER button		
LED Indicator	Power Status, Reset, OpenSDA, USER LED		
Expansion Port	Arduino interface		
РСВ	3.937 inch x 5.9055 inch (10 cm x 15 cm), 4-layer board		

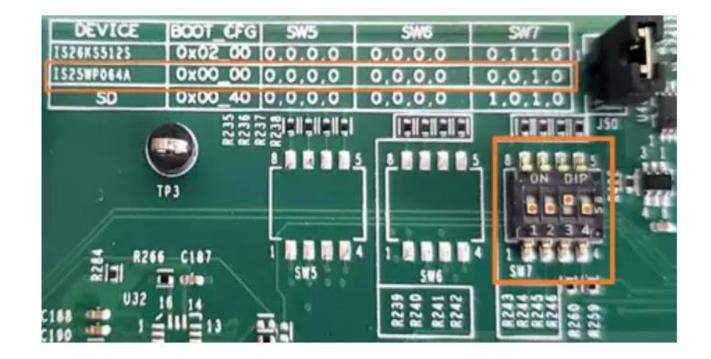
•

.

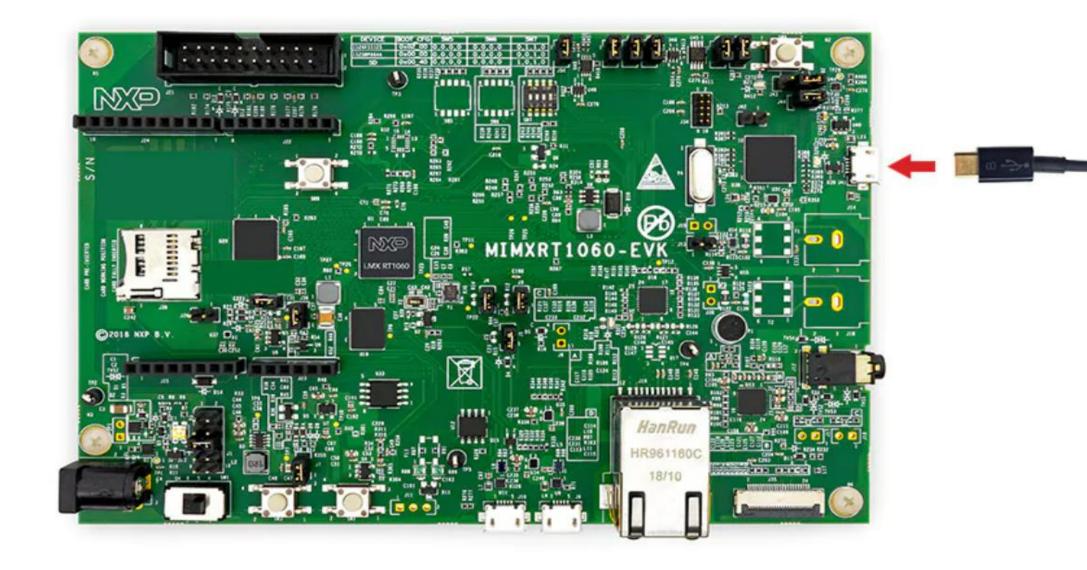
#### **Configure Boot Mode:**

The device has four boot modes (one is reserved for NXP use). The boot mode is selected based on the binary value stored in the internal BOOT\_MODE register. Switch SW7 is used to select the boot mode on the MIMXRT1060-EVK / EVKB / MIMXRT1064-EVK board.

To boot from the QSPI flash, make sure SW7 is set to 0010.

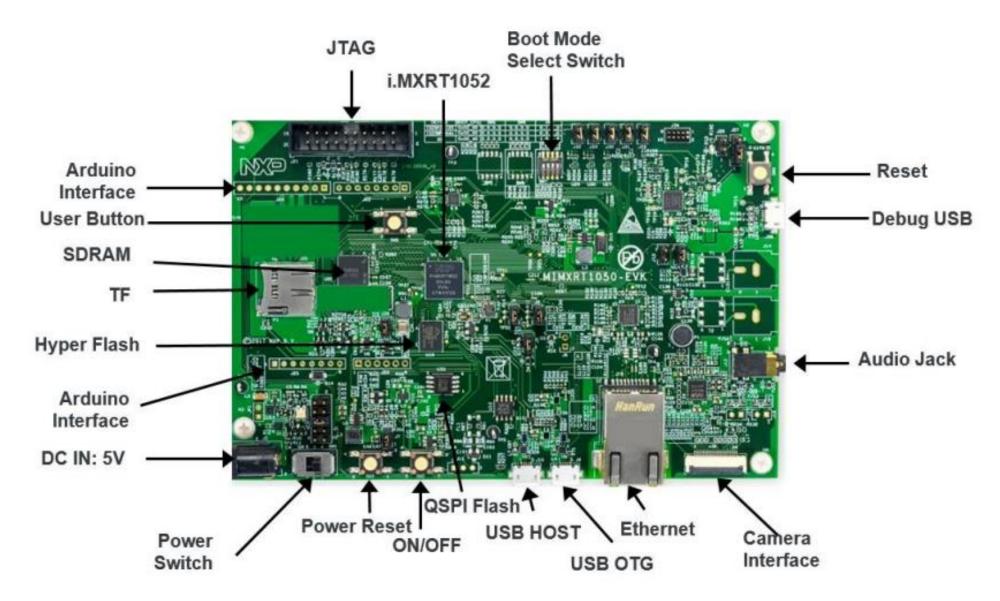


#### **Attach USB Cable:**

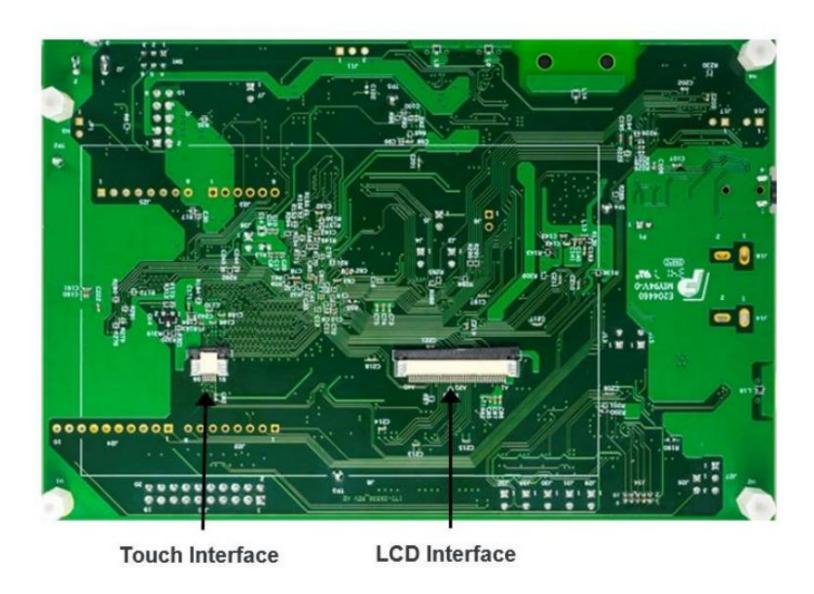


**IMX RT1050-EVK** 

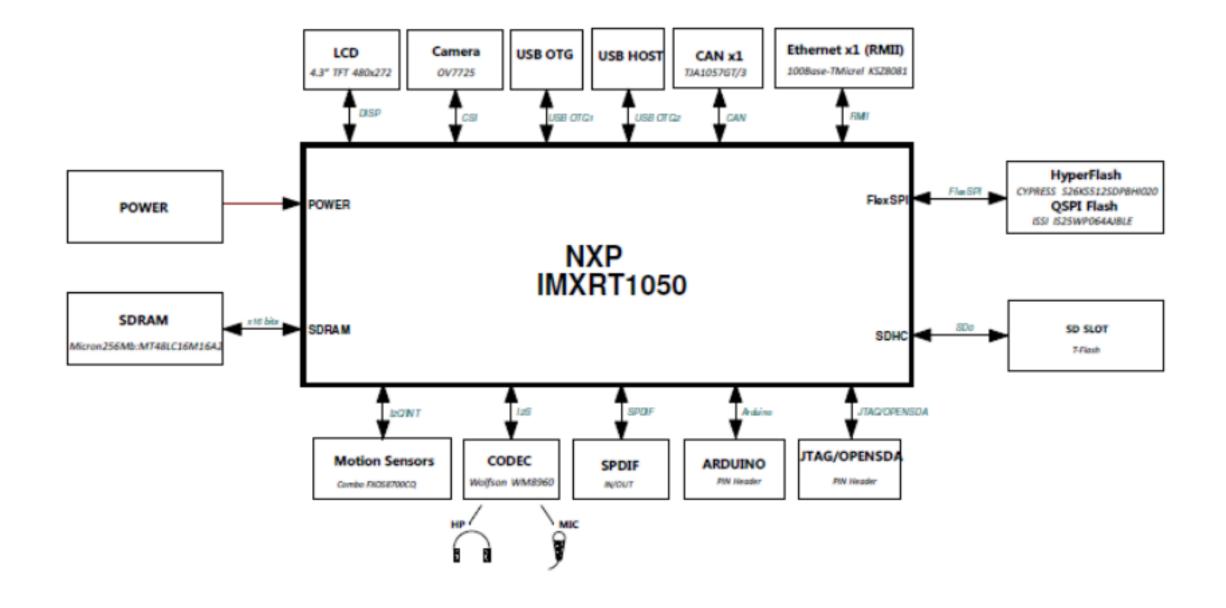
#### **Overview of the MIMXRT1050 EVK board (Front side):**



#### **Overview of the MIMXRT1050 EVK board (Back side):**



#### **Block diagram:**



#### **Board Features:**

			, ⊢		
Processor	NXP Processor	MIMXRT1052DVL6A(A0 silicon)			3.5 mm Audio Stereo Headphone Jack
Processor		MIMXRT1052DVL6B(A1 silicon)			Board-Mounted Microphone
DRAM Memory	SDRAM 256 Mb, 166MHz	MT48LC16M16A2B4-6AIT: G		Audio Connector	Left & Right Speaker Out Connectors
DCDC	MPS	MP2144GJ			SPDIF Interface(unpopulated)
	UNION	ON UM1550S-18		Power Connector	5V DC-Jack
LDO		UM1750S-00		Debug Connector	JTAG 20-pin Connector (SWD by default)
	TF Card Slot Mass Storage 64 Mbit Quad SPI Flash			Debug Connector	OpenSDA with DAP-Link
Mass Storage				Sensor	FXOS8700CQ: 6-Axis Ecompass (3-Axis Mag, 3-Axis Accel)
	512 Mbit Hyper Flash			Camera	CMOS Sensor Interface
	512 Wolt Hyper Flash		$\left  \right $	CAN	CAN Bus Connector
Display Interface	LCD Connector			User Interface Button	ON/OFF, POR Reset, Reset, USER Button
Ethernet	10/100 Mbit/s Ethernet Connector. PHY Chip: KSZ8081RNB			Led Indicator	Power Status, Reset, OpenSDA, USER LED
	USB 2.0 OTG Connector USB 2.0 Host Connector		1	Expansion Port	Arduino Interface
USB			1		
				PCB	3.937-inch x 5.9055-inch (10cm x 15cm), 4-layer board

#### **Configure Boot Mode:**

- The device has four boot modes (one is reserved for NXP use). The boot mode is selected based on the binary value stored in the internal BOOT\_MODE register. Switch (SW7-3 & SW7-4) is used to select the boot mode on the MIMXRT1050 EVK Board.
- Enable Hyper Flash Boot mode to see the output in TeraTerm.

	-			
SW7-1	SW7-2	SW7-3	SW7-4	<b>Boot Device</b>
OFF	ON	ON	OFF	Hyper Flash
OFF	OFF	ON	OFF	QSPI Flash
ON	OFF	ON	OFF	SD Card

### **Software and Hardware Installation**



### **Install the MCUXpresso IDE:**

- Install the latest version of MCUXpresso IDE
- Use the link:

https://www.nxp.com/design/software/development-software/mcuxpresso-softwareand-tools-/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE?tab=Design\_Tools\_Tab

• It will ask you to log in to NXP. If you don't have an account in NXP, create one and proceed.



Q Search

Q

FOLLOW

⊠ ~

Home / Software / Development Software / MCUXpresso Software and Tools / MCUXpresso Integrated Development Environment (IDE)

#### MCUXpresso Integrated Development Environment (IDE)

OVERVIEW	DOCUMENTATION DOWNLOADS	DEVELOPMENT TOOLS	TRAINING & SUPPORT
Jump To	Overview	Features	
Overview & Features		Accomplimentary	unlimited code cize, ecoly to use IDE
Supported Devices	The MCUXpresso IDE brings developers an ea		unlimited code size, easy-to-use IDE
System Requirements	Eclipse-based development environment for N MCUs based on Arm <sup>®</sup> Cortex <sup>®</sup> -M cores, includ general purpose crossover and wireless - enab	ing its coloring, MCU-sp bled and profiling	, compiling and editing with syntax pecific debugging views, code trace
	MCUs. The MCUXpresso IDE offers advanced compiling, and debugging features with the adv MCU-specific debugging views, code trace and	• Use built-in SDK built packages ma	selection tool, or drag and drop pre- ade with SDK Builder
	multicore debugging, and integrated configurat		S / 20.04.2 LTS, Github project

More 🗸

USER GUIDE

DOWNLOAD

**Tera Term** 

#### **Install Tera Term:**

- Tera Term is the terminal emulator for Microsoft Windows, that supports serial port, telnet and SSH connections.
- Among many other features it also has built-in Macro scripting language.
- Tera Term is often used to automate tasks related to remote connections initiated from PC.
- Install the latest version of Tera Term
- Use the link:

https://ttssh2.osdn.jp/index.html.en

# **MCUXpresso SDK**

#### **Install the MCUXpresso SDK:**

- The MCUXpresso SDK is complimentary and includes full source code under a permissive open-source license for all hardware abstraction and peripheral driver software.
- Download MCUXpresso SDK 2.6.2 version for i.MXRT1060 (or) SDK 2.6.0 version for i.MXRT1050. It includes the eIQ software platform and demos:
- Use the link: <u>https://mcuxpresso.nxp.com/en/welcome</u>
- Click on select development board



# • Select boards>i.MX> and your board name(EVK-MIMXRT1060 or EVK-MIMXRT1050)

Q

#### **Select Development Board**

Search for your board or kit to get started.

#### Search for Hardware

U

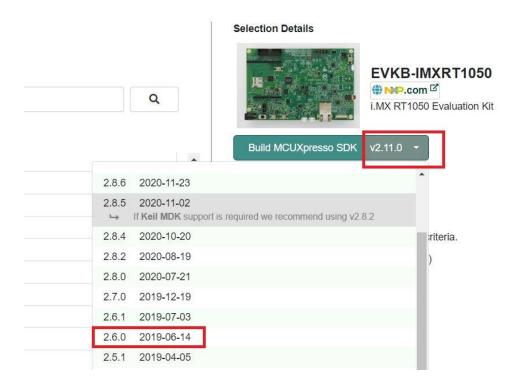
#### Select a Board, Kit, or Processor

B	Boards	
۲	JN	
•	• K32W	
•	Kinetis	
٠	▶ LPC	
•	▶ MW	
•	PN76	
•	QN	
•	• dsc	
	▼ I.MX	

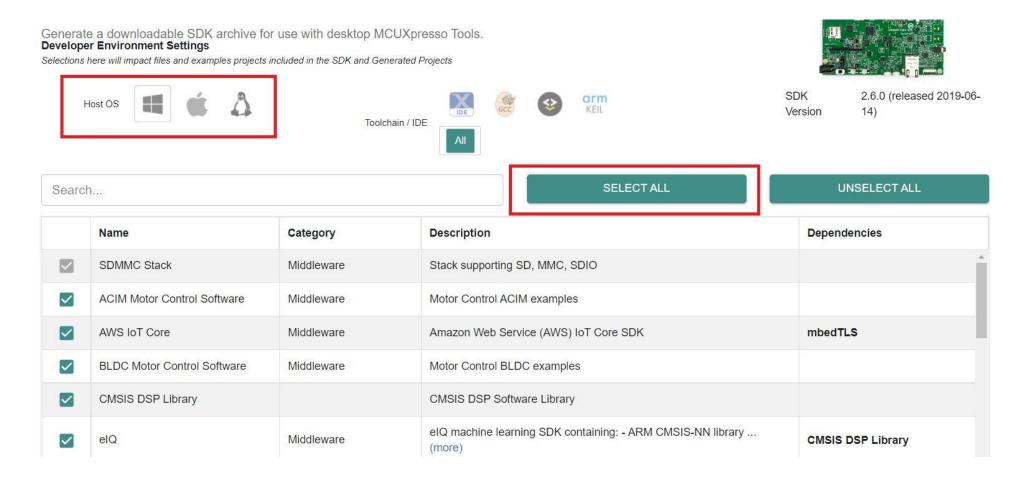
EVIZ MOINVTH D (MOINVTHEWWWW)

<b>▼</b> i.1	MX
	EVK-MCIMX7ULP (MCIMX7U5xxxxx)
	EVK-MIMX8DXL (MIMX8DL1xxxFZ)
	EVK-MIMX8MM (MIMX8MM6xxxLZ)
	EVK-MIMX8MN (MIMX8MN6xxxJZ)
	EVK-MIMX8MNDDR3L (MIMX8MN6xxxJZ)
	EVK-MIMX8MP (MIMX8ML8xxxLZ)
	EVK-MIMX8MQ (MIMX8MQ6xxxJZ)
	EVK-MIMXRT1010 (MIMXRT1011xxxxx)
	EVK-MIMXRT1015 (MIMXRT1015xxxxx)
	EVK-MIMXRT1020 (MIMXRT1021xxxxx)
	EVK-MIMXRT1060 (MIMXRT1062xxxxA)
	EVK-MIMXRT1064 (MIMXRT1064xxxxA)
	EVK-MIMXRT595 (MIMXRT595S)
	EVK-MIMXRT685 (MIMXRT685S)
	EVKB-IMXRT1050 (MIMXRT1052xxxxB)

- It leads to select board details in right most corner of the page. There click on version tab
- For IMXRT1060: Select 2.6.2 version
- For IMXRT1050: Select 2.6.0 version
- After that, click on build MCUXpresso SDK.



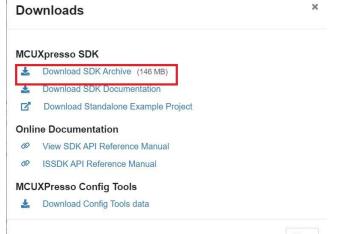
- Then select you Host OS and click on select all icon which will select all the software components related to that board.
- After that go to bottom of the page and click on download SDK



• This will build the required SDK and after building we will be able to download it.

SDK Archive Details	Actions
SDK_2.6.0_EVK-MIMXRT1060 Build Date: 2022-02-17, Board: EVK-MIMXRT1060 OS: Windows, Toolchain: All Toolchains Components: Cypress WICED WIFI Stack, JPEG library, FreeMASTER, PM Motor Control Software, SDMMC Stack, FreeRTOS, CMSIS DSP Library, Li AWS IoT Core, emWin, eIQ, USB Host, Device, OTG Stack, IwIP, Faffs, BL Control Software, QCA WIFI Stack, ACIM Motor Control Software, mbedTLS IEC60730B Safety Library, MCU Boot, Nxp Iot sensing sdk SDK Version: 2.6.0 (2019-06-14)	ittleFS, DC Motor
SDK_2.6.0_EVKB-IMXRT1050 Build Date: 2022-02-17, Board: EVKB-IMXRT1050 OS: Windows, Toolchain: All Toolchains Components: Cypress WICED WIFI Stack, JPEG library, Nghttp2 HTTP/2 C PMSP Motor Control Software, SDMMC Stack, FreeRTOS, CMSIS DSP Lit LittleFS, AWS IoT Core, emWin, eIQ, USB Host, Device, OTG Stack, IwIP, I BLDC Motor Control Software, QCA WIFI Stack, cJSON, ACIM Motor Contr Software, mbedTLS, IEC60730B Safety Library, Secure Element Host Libra Boot, PicoHTTPParser, Nxp iot sensing sdk, Azure IoT SDK Version: 2.6.0 (2019-06-14)	brary, Fatfs, ol

• Then select download SDK archive. Agree the terms and condition. Now SDK archive will be downloaded.

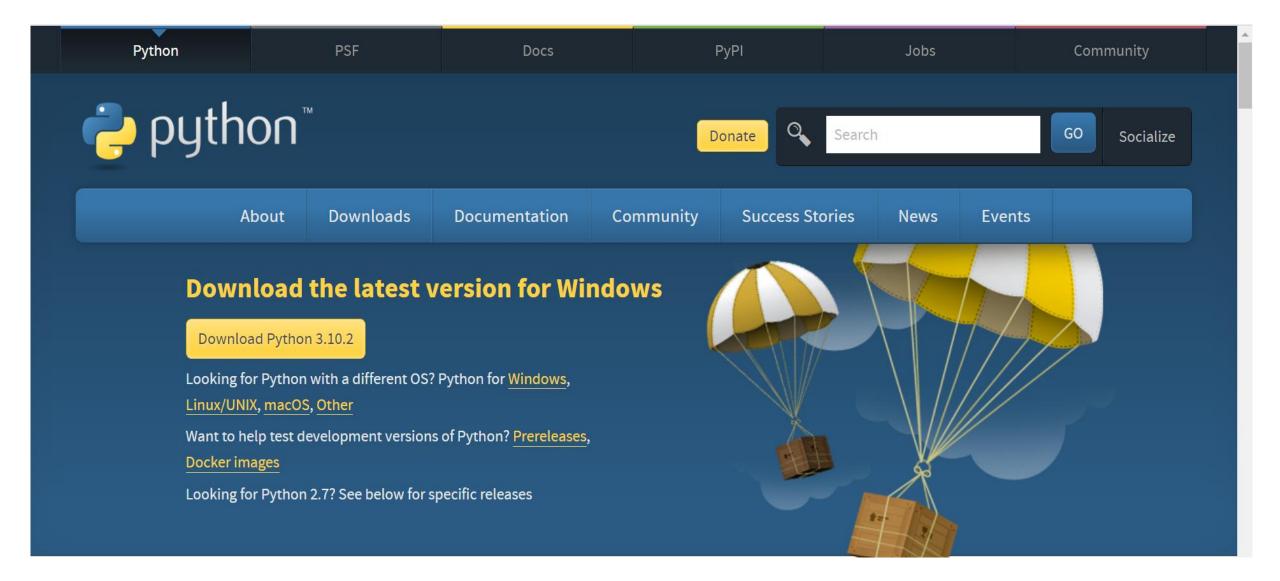


### **TensorFlow**

#### **Install TensorFlow:**

- TensorFlow makes it easy to create machine learning models for desktop, mobile, web, and cloud
- TensorFlow provides a collection of workflows to develop and train models using Python or JavaScript, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use.
- Download and install Python latest version. The 64-bit edition is required.
- Use the link:

https://www.python.org/downloads/



• Verify that the python command corresponds to Python 3.x.Type the Command in command prompt :

- Update the python installer tools. Type the command in command prompt:
   py -m pip install -U pip
   py -m pip install -U setuptools
- Install the latest version Tensorflow libraries and support for python. Type the command in command prompt:
   py -m pip install tensorflow

- Install other useful python packages. Type the command in command prompt: py -m pip install tensorflow-datasets py -m pip install numpy scipy matplotlib ipython jupyter pandas sympy nose py -m pip install opency-python py -m pip install PILLOW py -m pip install netron
- If on Windows, install latest Vim using the link: <u>https://www.vim.org/download.php#pc</u>. There is a binary convertor programmed named xxd.exe located inside that package that will be needed.

SPONSOR Vim development	VOTE for features	the editor	BUY the Vim book	HEL Ugar
not logged in ( <u>login</u> ) ENHANCED BY Google Search	Downloading Vim is availab Most popular:	Vim le for many different systems and there are several versions. This page will help you decide what to	download.	
Home Advanced search About Vim	MS- Windows:	Recent and signed MS-Windows files are available on the <u>vim-win32-installer site</u> The current stable version is <u>gvim_8.2.2825.exe</u> . An alternative is the <u>standard serr-installing executable</u> , currently version 8.2.2824.		
<u>Community</u> <u>News</u> <u>Sponsoring</u>	Unix	See the <u>GitHub</u> page, or <u>Mercurial</u> , if you prefer that. There is also an <u>Appimage</u> which is build date many Linux systems.	ily and runs or	۱
<u>Trivia</u> Documentation Download	Mac	: See the MacVim project for a GUI version and Homebrew for a terminal version		

• If on Windows, add the following directories to your executable PATH if they are not already. Steps to add path is:

Go to system properties>Environment Variables

System Properties X
Computer Name Hardware Advanced System Protection Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings
User Profiles
Desktop settings related to your sign-in
Settings
Startup and Recovery
System startup, system failure, and debugging information
Settings
Environment Variables
OK Cancel Apply

#### Under System Variables, select path and edit.

Add path as: <python\_install\_directory>/scripts <vim\_install\_directory>

#### User variables for Fathima Afreen

Value			
C:\Freescale\KSDK_1.3.0			
C:\Users\Fathima Afreen\OneDrive			
C:\Users\Fathima Afreen\OneDrive			
C:\Users\Fathima Afreen\AppData\Local\Programs\Python\Pyth			
C:\conda_tmp			
C:\conda_tmp			
	••	<b>E</b> 11	Delete
	C:\Users\Fathima Afre C:\Users\Fathima Afre C:\Users\Fathima Afre C:\conda_tmp	C:\Users\Fathima Afreen\OneDrive C:\Users\Fathima Afreen\OneDrive C:\Users\Fathima Afreen\AppData\Loc C:\Users\Fathima Afreen\AppData\Loc C:\conda_tmp	C:\Users\Fathima Afreen\OneDrive C:\Users\Fathima Afreen\OneDrive C:\Users\Fathima Afreen\AppData\Local\Programs\Pyt C:\conda_tmp C:\conda_tmp

System variables	Syster	n varia	bles
------------------	--------	---------	------

Variable	Value
OS	Windows NT
Path	C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 140 Stepping 1, GenuineIntel
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	8c01
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules:C:\WINDOWS\sv
	New Edit Delete
	OK Cancel

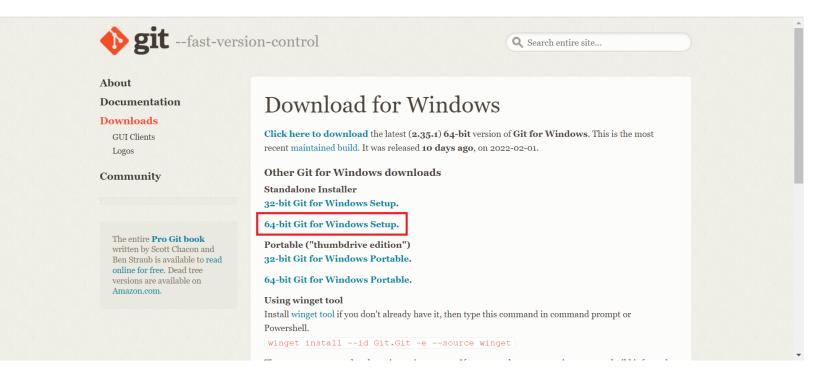
Edit environment variable

C:\WINDOWS\system32	New
E:\WINDOWS	
C:\WINDOWS\System32\Wbem	Edit
C:\WINDOWS\System32\WindowsPowerShell\v1.0\	Edit
C:\WINDOWS\System32\OpenSSH\	Browse
C:\Program Files\MATLAB\R2021a\bin	Browse
C:\Users\Fathima Afreen\anaconda3	
C:\Users\Fathima Afreen\anaconda3\Library\mingw-w64\bin	Delete
C:\Users\Fathima Afreen\anaconda3\Library\usr\bin	
C:\Users\Fathima Afreen\anaconda3\Library\bin	
C:\Users\Fathima Afreen\anaconda3\Scripts	Move Up
C:\Users\Fathima Afreen\AppData\Local\Microsoft\WindowsApps	
C:\nxp\Glow\bin	Move Down
C:\Program Files\Git\cmd	
C:\Program Files (x86)\Vim\vim82	
C:\Users\Fathima Afreen\AppData\Local\Programs\Python\Python310\Scripts	Edit text

ОК

## **Lab Scripts Installation**

- The python scripts that will be used to retrain an already existing model will be downloaded via Git.
- Experiment: We'll be retraining the model to recognize photos of flowers and categorize them into different types(5 category)- daisy, dandelion, roses, sunflowers and tulips
- The new flower data that the model will be retrained on also will be download.
- Steps are:
- 1. Install Git: <u>https://git-scm.com/downloads</u>



2. Open a command prompt, and in a directory of your choosing, download the tutorial repository with git: git clone <u>https://github.com/googlecodelabs/tensorflow-for-poets-2</u>

3. Download a set of Creative Commons licensed flowers images that have already been categorized into 5 different classes: <a href="http://download.tensorflow.org/example\_images/flower\_photos.tgz">http://download.tensorflow.org/example\_images/flower\_photos.tgz</a>

4. Unzip that file which will create a "flower\_photos" directory:

a. If on Windows, you may need to install 7-zip or Winzip to unzip the .tgz file.b. If on Linux, use: tar -xvzf flower\_photos.tgz

5. Place the "flower\_photos" directory inside the "tf\_files" directory that is in the tensorflow-for-poets-2 repo downloaded in the first step. It should look like the following when done:

Name	Date modified	Туре	Size
📜 daisy	2/7/2022 4:45 PM	File folder	
andelion	2/7/2022 4:45 PM	File folder	
roses	2/7/2022 4:45 PM	File folder	
sunflowers	2/7/2022 4:45 PM	File folder	
📜 tulips	2/7/2022 4:45 PM	File folder	
	2/8/2016 9:59 PM	Text Document	409 KB

# **Retrain Existing Model**

• For this lab we will retrain an already existing model with new data. This is called transfer learning. The structure of the model has already been setup for image classification, so the goal is to retrain one layer to classify new images with new custom labels. This greatly shortens the amount of time it will take to train the model. Once retrained, the model can be converted to TensorFlow Lite format and ran on the i.MXRT device. The following steps are based on this Google CodeLabs tutorial: TensorFlow for Poets

https://codelabs.developers.google.com/codelabs/tensorflow-forpoets/index.html/

• 1. Open a command prompt and go to the directory that was created by cloning the git repository in the last section. It should be something like this:

cd C:\eiq\tensorflow-for-poets-2

2. The model being used is called MobileNet.

Open retrain.py file changed 109 line i.e, earlier it was import tensorflow as tf we changed it to import tensorflow.compat.v1 as tf. The image after changing is shown below.

temp.	py × retrain.py ×
91	
92	
93	fromfuture import absolute_import
94	fromfuture import division
95	fromfuture import print_function
96	
97	import argparse
98	import collections
99	from datetime import datetime
100	import hashlib
101	import os.path
102	import random
103	import re
104	import sys
105	import tarfile
106 107	import summer of the
107	import numpy as np from six.moves import urllib
109	import tensorflow.compat.v1 as tf
110	
111	from tensorflow.python.framework import graph util
112	from tensorflow.python.framework import tensor shape
113	from tensorflow.python.platform import gfile
114	from tensorflow.python.util import compat
115	
116	FLAGS = None
117	
118	# These are all parameters that are tied to the particular model architect
119	# we're using for Inception v3. These include things like tensor names and
120	# sizes. If you want to adapt this script to work with another model, you
121	# need to update these to reflect the values in the network you're using.
122	MAX_NUM_IMAGES_PER_CLASS = 2 ** 27 - 1 # ~134M
110	

We will retrain this model for 128x128 pixel images using a python script found inside the tutorial folder. Navigate to the main root directory and run the following command.

This should be one long continuous line like in the image below:

C:\Data\eIQ\lab\tensorflow-for-poets-2>python -m scripts.retrain --bottleneck\_dir=tf\_files/bottlenecks --how\_many\_traini ng\_steps=500 --model\_dir=tf\_files/models --summaries\_dir=tf\_files/training\_summaries/mobilenet\_0.25\_128 --output\_graph=t E\_files/retrained\_graph.pb --output\_labels=tf\_files/retrained\_labels.txt --architecture=mobilenet\_0.25\_128 --image\_dir=t E\_files/flower\_photos

python -m scripts.retrain

- --bottleneck\_dir=tf\_files/bottlenecks
- --how\_many\_training\_steps=500
- --model\_dir=tf\_files/models/
- --summaries\_dir=tf\_files/training\_summaries/mobilenet\_0.25\_128
- --output\_graph=tf\_files/retrained\_graph.pb
- --output\_labels=tf\_files/retrained\_labels.txt
- --architecture=mobilenet\_0.25\_128
- --image\_dir=tf\_files/flower\_photos

3. This will take several minutes to run. While waiting, here's an explanation for the arguments in the command you just ran:

#### --bottleneck\_dir=tf\_files/bottlenecks

Directory to store cached data information

#### --how\_many\_training\_steps=500

# of iterations. The more iterations the longer the training takes, but the more accurate the model will likely be

#### --model\_dir=tf\_files/models/

Directory location to download the pre-existing model

#### --summaries\_dir=tf\_files/training\_summaries/mobilenet\_0.25\_128

Output directory for data files used by an optional analysis program called TensorBoard

#### --output\_graph=tf\_files/retrained\_graph.pb

Name of the retrained model in Protocol Buffer (pb) format.

#### --output\_labels=tf\_files/retrained\_labels.txt

Text file with the labels for the model (determined by the names of the subdirectories of the training data)

#### --architecture=mobilenet\_0.25\_128

The particular type of Mobilenet model to use as a starting point

#### --image\_dir=tf\_files/flower\_photos

Location of the data to train the model on. Each sub-directory is a label classifying those images

4. Once finished, look inside the tf\_files directory. You should have a model file named retrained\_graph.pb. This is the newly trained model.

5. You can test this model file against flower images using the label\_image python script. Open label\_image python script and change import tensorflow as tf to import tensorflow.compat.v1 as tf, tf.compat.v1.disable\_eager\_execution () at line 25 and 26. See below figures for reference.

	temp	p.py × label_image.py ×	≡
	4 5 6	# you may not use this file except in compliance with the License. # You may obtain a copy of the License at #	Î
	7	# http://www.apache.org/licenses/LICENSE-2.0 #	
	9 10 11 12 13 14 15	<pre># Unless required by applicable law or agreed to in writing, software # distributed under the License is distributed on an "AS IS" BASIS, # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. # See the License for the specific language governing permissions and # limitations under the License. # ====================================</pre>	-
	16 17 18 19	<pre>fromfuture import absolute_import fromfuture import division fromfuture import print_function</pre>	
▲	20 21 22 23	import argparse import sys import time	
	24	import numpy as np	
	25 26 27	<pre>import tensorflow.compat.v1 as tf tf.compat.v1.disable_eager_execution()</pre>	
	28 29 30 31	<pre>def load_graph(model_file):     graph = tf.Graph()     graph_def = tf.GraphDef()</pre>	
	32 33 34 35	<pre>with open(model_file, "rb") as f: graph_def.ParseFromString(f.read()) with graph.as_default(): tf.import_graph_def(graph_def)</pre>	

• From the main directory, call a script to use the new graph and have it analyze a daisy image with the following command as one long continuous line:

python -m scripts.label\_image

--graph=tf\_files/retrained\_graph.pb

--input\_height=128

--input\_width=128

--image=tf\_files/flower\_photos/daisy/21652746\_cc379e0eea\_m.jpg

6. It should respond back that that photo is of a daisy. The confidence level may vary slightly as the model training will be slightly different each time. If you see a low confidence level (<.80) for identifying that image as a daisy, try running the retraining script again.

Command Prompt	—	×
C:\Data\eIQ\dfae_lab\tensorflow-for-poets-2>python -m scripts.label_imagegraph=tf_files/retrained_grap ight=128input_width=128image=tf_files/flower_photos/daisy/21652746_cc379e0eea_m.jpg 2019-05-15 13:49:55.342850: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instru s TensorFlow binary was not compiled to use: AVX2		_
Evaluation time (1-image): 0.283s		
daisy (score=0.96397) dandelion (score=0.02553) roses (score=0.00974) sunflowers (score=0.00073) tulips (score=0.00004)		

# **Convert Model and Data**

Now that the retrained model is running on your laptop, the next step is to use a TensorFlow utility named tflite\_convert to convert that model into a file that can be loaded onto an embedded device like the i.MXRT1060/1050.

# **Convert TensorFlow model:**

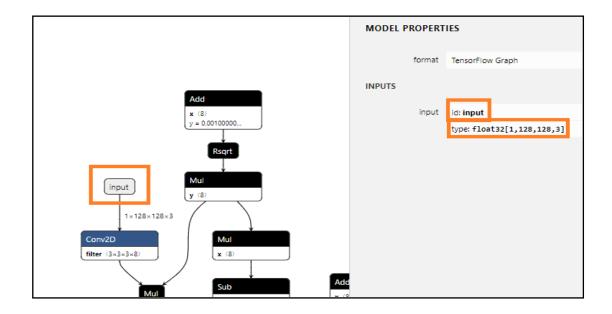
Convert the .pb model file into a format that can be imported into the RT1060 project with **tflite\_convert.** 

1. The first and last layer names need to be read from the TensorFlow .pb model file, which will be used later in the conversion process. This can be done with a neural network visualization tool called Netron.

a. Use Netron by using the following command on a .pb file:

#### netron tf\_files\retrained\_graph.pb

b. While the command is running, open up a web browser and navigate to http://localhost:8080 and click on the first and last nodes to get the layer names and the input shape.





c. After reading the labels, go back to the terminal window and hit Ctrl+c to stop the server and return to the command prompt

2. Use tflite\_convert to transform the model file into a .tflite file. There are options available to quantize and optimize the model during this step, but they seem to significantly decrease the accuracy of the converted model. The following options keep the accuracy about the same as the full model. Type in the command as one long continuous line like below:

#### tflite\_convert

--graph\_def\_file=tf\_files/retrained\_graph.pb
--output\_file=tf\_files/retrained\_graph.tflite
--input\_shape=1,128,128,3
--input\_array=input
--output\_array=final\_result
--inference\_type=FLOAT
--input\_data\_type=FLOAT
--enable\_v1\_converter

Here is what each of those arguments determine:

#### --graph\_def\_file=tf\_files/retrained\_graph.pb

Name of the TensorFlow model to convert

#### --output\_file=tf\_files/retrained\_graph.tflite

Name of the converted tflite file

#### --input\_shape=1,128,128,3

This model takes in a 128 x 128 image that has 3 color channels

#### --input\_array=input

Name of the first layer of the model

#### --output\_array=final\_result

Name of the last layer of the model

#### --inference\_type=FLOAT

Model uses floating (instead of 8-bit quantized) inference

#### --input\_data\_type=FLOAT

Model uses floating (instead of 8-bit quantized) input

3. You may get a warning about AXV2 instructions not being supported. You can ignore this warning.

# 4. Inside the tf\_files directory you should see a new file named **retrained\_graph.tflite**

5. Use the **xxd** utility to convert the .tflite binary file into a C array that can be imported into an embedded project. Do not run this command in Powershell as it will cause compilation issues later. Run it in a standard command prompt:

#### xxd -i tf\_files/retrained\_graph.tflite > tf\_files/retrained\_graph.h

6. The generated header file may need to be modified slightly to define it as a const. Open up the file and, if necessary, change "unsigned char" to "const char". Also make note of the array name as it will be used in the next section.

🔚 retraine	🔚 retrained_graph.h 🔀											
1 [	const cl											
2	0x18,	0x00,	0x00,	0x00,	0x54,	0x46,	0x4c,	0x33,	0x00,	0x00,	0x0e,	0x00,
3	0x18,	0x00,	0x04,	0x00,	0x08,	0x00,	0x0c,	0x00,	0x10,	0x00,	0x14,	0x00,
4	0x0e,	0x00,	0x00,	0x00,	0x03,	0x00,	0x00,	0x00,	0x80,	0x0c,	0x1d,	0x00,
5	0x0c,	0x00,	0x00,	0x00,	0x10,	0x00,	0 <b>x</b> 00,	0x00,	0 <b>x</b> 20,	0x00,	0x00,	0x00,
6	001	000	000	000	00	0	01	000	004	000	000	000

#### 7. The following files should now be in the tf\_files directory:

C:\Data\eIQ\lab\tensorflow-for-poets-2>dir tf\_files Volume in drive C is OSDisk Volume Serial Number is 2648-1211 Directory of C:\Data\eIQ\lab\tensorflow-for-poets-2\tf\_files 08/20/2019 01:54 PM <DIR> <DIR> 8/28/2819 01:54 PM 20/2019 01:32 PM 0 .empty <DIR> 08/20/2019 01:39 PM bottlenecks <DIR> flower\_photos 38/20/2019 01:35 PM 01:38 PM <DIR> models 8/28/2819 retrained graph.h 01:54 PM 11,988,817 8/20/2019 2,828,851 01:48 PM retrained graph.pb 8/20/2019 retrained\_graph.tflite 8/20/2019 01:46 PM 1,904,112 retrained labels.txt 8/28/2819 01:40 PM 40 38/20/2019 01:38 PM <DIR> training summaries 5 File(s) 15,825,020 bytes 6 Dir(s) 429,096,202,240 bytes free :\Data\eIQ\lab\tensorflow-for-poets-2>

# **Image Data Conversion – No Camera Only:**

Now that the model has been converted, an image also needs to be converted into a C array. The Label Image example in eIQ takes in 24-bit BMP image files, so will convert one of the images in the dataset to a BMP file and then convert that into a C array.

8. Convert one of the JPEG flower image files into a 24-bit BMP file. Preferably pick an image that was labeled already using the label\_image python script so that result can be compared to the result on the RT1060. In this case we'll pick the file

#### tf\_files/flower\_photos/daisy/21652746\_cc379e0eea\_m.jpg.

Save the new .bmp file as daisy.bmp

• In Linux use:

convert tf\_files/flower\_photos/daisy/21652746\_cc379e0eea\_m.jpg -type truecolor tf\_files/daisy.bmp

• In Windows use a paint program like MS Paint.

9. After converting the .jpg file to a .BMP file, use the xxd program to convert it to a C array. Do not run this command in Powershell as it will cause compilation issues later. Run it in a standard command prompt:

#### xxd -i daisy.bmp > daisy.h

10. Open the generated header file and, if necessary, change the array type from "unsigned char" to "const char" and make note of the array name, as that name will be used in the next section:

🔚 daisy	h 🔀											
1	Const cl	har	daisy_b	omp[] =	= {							
2	0x42,	0x4d,	0xb6,	0x8c,	0x02,	0x00,	0x00,	0x00,	0x00,	0x00,	0x36,	0x00,
3	0x00,	0x00,	0x28,	0x00,	0x00,	0x00,	0xe7,	0x00,	0x00,	0x00,	0xf0,	0x00,
4	0x00,	0x00,	0x01,	0x00,	0x18,	0x00,	0x00,	0x00,	0x00,	0x00,	0x80,	0x8c,
5	0x02,	0x00,	0xc4,	0x0e,	0x00,	0x00,	0xc4,	0x0e,	0x00,	0x00,	0x00,	0x00,
6	0x00,	0x00,	0x00,	0x00,	0x00,	0x00,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,
7	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,
8	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,	0x01,

# **Run Demo**

• The final step is to take the Label Image example and modify it to use the retrained model with the new image.

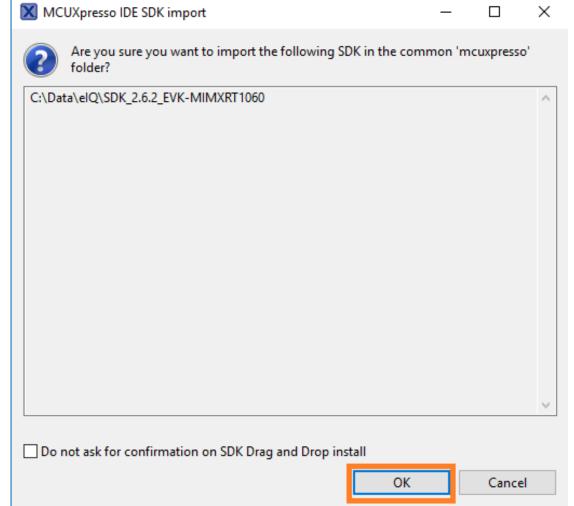
## Copy and Create Files:

- 1. Open MCUXpresso IDE and select a workspace location in an empty directory.
- 2. Because of the large size of the model file, the indexer settings need to be changed by going to Window->Preferences from the menu bar. In the dialog box that comes up, go to the C/C++->Indexer category and uncheck Enable Indexer. Then click on Apply and Close.

See image in next page for reference.

type filter text	Indexer 🗘 🔻
> General	
✓ C/C++	Enable indexer
Appearance	Indexer options
> Build	Index source files not included in the build
Code Analysis	Index unused headers
> Code Style	Index all header variants
Core Build Toolchains	Index all variants of specific headers:
> Debug	
> Editor	Index source and header files opened in editor
File Types	Allow heuristic resolution of includes
Indexer	
Language Mappings	Skip files larger than:
> New C/C++ Project Wizard	Skip included files larger than: MB
> Profiling	Skip all references (Call Hierarchy and Search will not work)
> Property Pages Settings	Skip implicit references (e.g. overloaded operators)
Task Tags	Skip type and macro references (Search for these references will not wor
Template Default Values	
> Help	Indexing strategy
> Install/Update	Automatically update the index
> Library Hover	Update index immediately after every file-save
MCUXpresso Config Tools	Build configuration for the indexer
> MCUXpresso IDE	
> Run/Debug	• Use active build configuration
SWTChart	○ Use the build configuration specified in the project's indexer settings
> Terminal	Cache limits
Validation	Index database cache:
> Version Control (Team)	Limit relative to the maximum heap size: 10 %
> XML	Absolute Limit: 256 MB

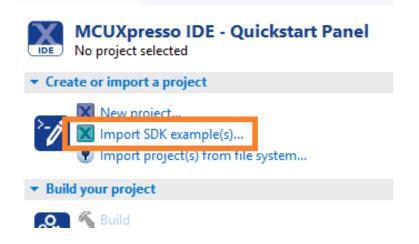
3. Drag-and-drop the unzipped SDK folder into the Installed SDKs window. It should have updated files as described in the first section. You will get the following pop-up, so hit OK.



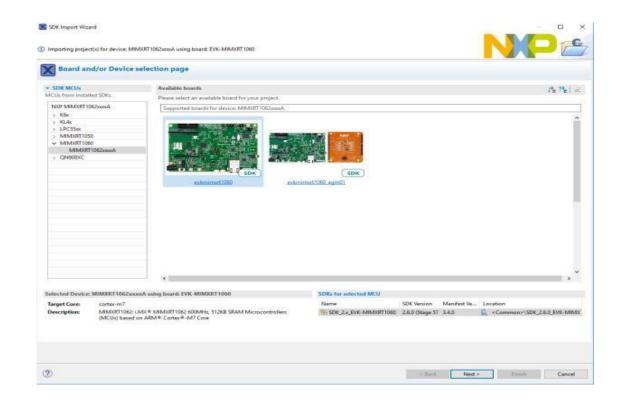
#### 4. Once imported, the Installed SDK window will look something like this

🎁 Installed SDKs 🙁 🔲 Properties   🖹 Pro	oblems 📃 Console 🐰	🛙 Terminal 🛛 📄 Image I	Info 📓 Debugger Console 🛛 🗖 🗖
			🥹 🗞   🗉 🖻   🖽 🦷
🕅 Installed SDKs			
To install an SDK, simply drag and drop an SD	K (zip file/folder) into th	e 'Installed SDKs' view. [	[Common 'mcuxpresso' folder]
Name	SDK Version	Manifest Version	Location
BDK_2.x_EVK-MIMXRT1060	2.6.2 (238 2019-07	3.5.0	Common>\SDK_2.6.2_EVK-MIMXRT1060
SDK_2.x_EVKB-IMXRT1050	2.6.1 (236 2019-07	3.5.0	Common>\SDK_2.6.1_EVKB-IMXRT1050.zip
🗹 🌐 SDK_2.x_FRDM-KL27Z	2.4.1 (19f8bd257a1	3.3.0	Common>\SDK_2.4.1_FRDM-KL27Z.zip
🗹 🌐 SDK_2.x_LPC54018-loT-Module	2.6.0 (228 2019-06	3.5.0	Common>\SDK_2.6.0_LPC54018-IoT-Module.zip
EXTERNO STREET S	2.2.0	3.0.0	Common>\SDK_2.2.0_QN908XCDK.zip

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



#### 6. Select the evkmimxrt1060 board and click on Next



7. Then expand the **eiq\_examples** folder and select **tensorflow\_lite\_label\_image**. Also select **UART** for the SDK Debug Console. Then click on Finish to select that project.

X SDK Import Wizard		X							
You have selected '1' project to import: 'evkmimxrt1060_tensorflow_lite_label_image'									
Import projects									
Project name prefix: evkmimxrt1060	2 Project name suffix:	Ø.							
Use default location									
Location: C:\Users\nxa06332\Documents\MCUXpressolDE_11.0.0_25	16\foo6\evkmimxrt1060	Browse							
Project Type	Project Options								
○ C Project	SDK Debug Console () Semihost () Example default Copy sources Import other files								
Examples		🖮   🖉 🗹 🞇   🕀 🕞							
type to filter									
<ul> <li>FreeMASTER_examples</li> <li>aws_examples</li> <li>bootloader_examples</li> <li>cmsis_driver_examples</li> <li>demo_apps</li> <li>driver_examples</li> <li>eiq_examples</li> <li>cmsis_nn_cifar10</li> <li>cmsis_nn_kws</li> <li>tensorflow_lite_cifar10</li> <li>tensorflow_lite_label_image</li> </ul>	CIFAR-10 example for CMSIS-NN Keyword spotting example for CMSIS-NN CIFAR-10 example for TensorFlow Lite Keyword spotting example for TensorFlow Lite Label image example for TensorFlow Lite TensorFlow Lite library								
?	< Back Next >	Finish Cancel							

8. It will look like the following when imported into the Project Explorer window:

workspace2 - Welcome page - MCUXpresso IDE File Edit Navigate Search Project ConfigTools Run FreeR 📑 🗝 🔚 🐚 | 🛞 🕶 🗞 🕶 🔜 🥔 😓 🕶 ⊑ | 🔌 | 🕪 陷 Project Explorer 🙁 🔀 Peripherals+ 🕮 Registers 🚸 Faults > 😂 evkmimxrt1060\_tensorflow\_lite\_label\_image <Debug>

9. Now we need to import the retrained model file that was generated in the last section into this project.

10. Find the directory location that this example was copied to by right clicking on the project name and select Properties. In the dialog box that comes up, click on the icon to open up that directory inside Windows explorer:

Properties for evkmimxrt1060_tensorflow_lite_label_image - D					
type filter text	Resource	⇔ - ⇔	2 - Contract (1997)		
<ul> <li>Resource</li> <li>Builders</li> <li>C/C++ Build</li> </ul>	Path: Type:	/evkmimxrt1060_tensorflow_lite_label_image Project	1	fsl_elcdif.h	
<ul> <li>C/C++ General</li> <li>MCUXpresso Config Tools</li> </ul>	Location:	C:\Users\nxa06332\Documents\MCUXpressolDE_11.0.0_2516\foo6\evkmimxrt1060_tensorflow_lite_label_image	2	fsl_camera_de	
Project Natures Project References Run/Debug Settings	Text file en		11	ystem Explorer fsl_mt9m114.1 fsl_pxp.h	

11. Go to the "source" directory inside the evkmimxrt1060\_tensorflow\_lite\_label\_image folder that you just opened. It should be something like: C:\Users\nxp\_training\Documents\MCUXpressoIDE\_11.0.0\_2516\workspace\e vkmimxrt10 60\_tensorflow\_lite\_label\_image\source

12. Inside that source directory, copy the retrained\_graph.h file generated in the previous section.

13. If not using the camera, also copy the daisy.h file generated from previous section.

14. In that same directory, create a new header file named flower\_labels.h and put in the following text, which will define the labels used to classify the flower images. This new file will be used to provide the classification labels instead of the labels.h file that was used by the default example. The file should look exactly like the following:

```
std::string labels_txt = R"(daisy
dandelion
roses
sunflowers
tulips
)";
```

15. Directory should look like the beside image when finished:

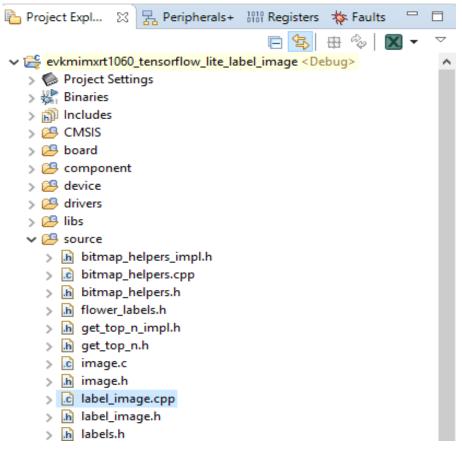
ame	✓ Date modified	Туре	Size
bitmap_helpers.cpp	2/11/2022 3:01 PM	CPP File	4 KB
bitmap_helpers.h	2/11/2022 3:01 PM	H File	2 KB
bitmap_helpers_impl.h	2/11/2022 3:01 PM	H File	4 KB
daisy.h	2/11/2022 3:09 PM	H File	1,020 KB
] flower_labels.h	2/11/2022 3:04 PM	H File	1 KB
] get_top_n.h	2/11/2022 3:01 PM	H File	2 KB
] get_top_n_impl.h	2/11/2022 3:01 PM	H File	3 KB
label_image.cpp	2/11/2022 3:01 PM	CPP File	8 KB
] label_image.h	2/11/2022 3:01 PM	H File	2 KB
labels.h	2/11/2022 3:01 PM	H File	18 KB
] mobilenet_v1_0.25_128_quant_model.h	2/11/2022 3:01 PM	H File	3,034 KB
retrained_graph.h	2/7/2022 7:32 PM	H File	11,700 KB
] semihost_hardfault.c	2/11/2022 3:01 PM	C File	5 KB
] stopwatch_image.h	2/11/2022 3:01 PM	H File	2,684 KB
) timer.c	2/11/2022 3:01 PM	C File	1 KB
] timer.h	2/11/2022 3:01 PM	H File	1 KB

C

# **Modify Source Code:**

Now edit the source files to include these new files

16. Double click on the label\_image.cpp file under the "source" folder in the Project View to open it.



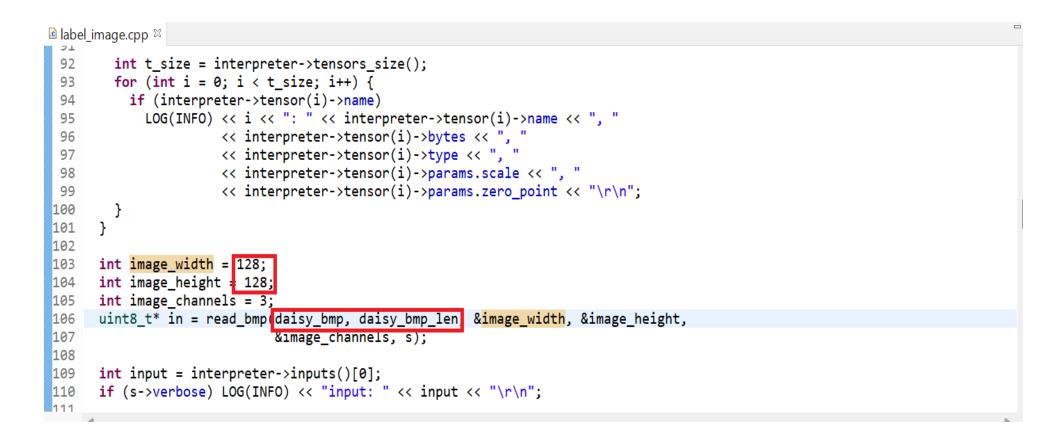
17. Starting on line 34, comment out original #includes for the image, model, and label files. Then add new #includes to bring in the new image, model, and label files that were copied in. It should look like the following when finished:

```
label image.cpp 🖾
 23 #INCLUDE LENSOF LOW/ CONCLED/ LICE/ REFNELS/ TERSTER. IN
 26 #include "tensorflow/contrib/lite/model.h"
 27 #include "tensorflow/contrib/lite/optional_debug_tools.h"
 28 #include "tensorflow/contrib/lite/string_util.h"
 29
 30 #include "label image.h"
 31 #include "bitmap helpers.h"
 32 #include "get top n.h"
 33
 34 //#include "stopwatch_image.h"
 35 //#include "mobilenet v1 0.25 128 quant model.h"
 36 //#include "labels.h"
 37
 38 #include "daisy.h"
 39 #include "retrained_graph.h"
 40 #include "flower labels.h"
 41
 42 #define LOG(x) std::cout
 43
 44<sup>⊕</sup> namespace tflite {
 450 namesnace label image {
```

18. At around line 70, comment out the API call to load the default model, and replace it with the new model name and model length from the header file. It may be a slightly different name than the one listed below:

```
result->emplace_back();
62
63
     return kTfLiteOk;
64
65 }
66
67<sup>©</sup> void RunInference(Settings* s) {
     std::unique_ptr<tflite::FlatBufferModel> model;
68
     std::unique ptr<tflite::Interpreter> interpreter:
     //model = tflite::FlatBufferModel::BuildFromBuffer(mobilenet model, mobilenet model len);
70
     model = tflite::FlatBufferModel::BuildFromBuffer(tf_files_retrained_graph_tflite, tf_files_retrained_graph_tflite_len);
     IT (!model) {
       LOG(FATAL) << "Failed to load model\r\n";
73
       return;
74
75
```

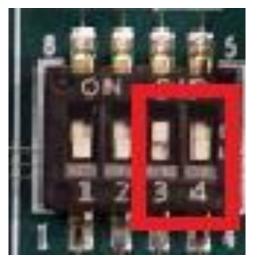
19. At around line 103, change the image height and width to 128 if they are not already, and then update the image buffer name and image length with the new image. The names may be a slightly different name than the one listed below and should match the array name and array length names in the daisy.h file:



## **Run Example:**

20. If using the i.MXRT1064 board, make the changes outlined in the following document: https://community.nxp.com/docs/DOC-344225. If using the i.MXRT1050 or i.MXRT1060 boards, this step is not needed.

21. On the i.MXRT1060 board, change SW7 to configure the board to boot from the flash. SW7 should be OFF-OFF-ON-OFF(as shown below). For i.MXRT1050 board SW7 should be OFF-ON-OFF.

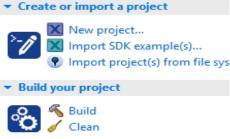


22. Plug the micro-B USB cable into the board at J41.

23. 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.

24. Build the project by clicking on "Build" in the Quickstart Panel.





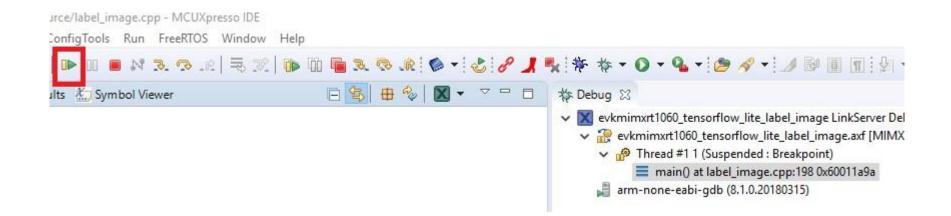


nate, Build and Debug

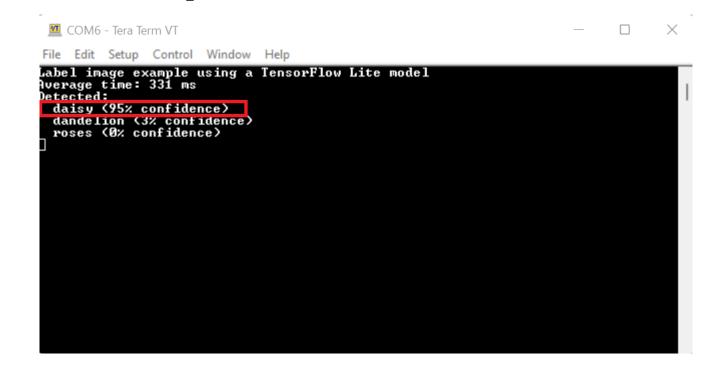
#### 26. It will ask what interface to use. Select CMSIS-DAP.

X	▼ Probes discovered – □ ×											
Con	Connect to target: MIMXRT1052xxxxB											
1 p	1 probe found. Select the probe to use:											
Ava	ailable attached pro	bes										
	Name	Serial number/ID	Туре	Manu	IDE De	bug M	ode					
X	CMSIS-DAP	0227000041114e45	LinkSen	ARM	Non-S	top						
Sup	oported Probes (tick/untick t	o enable/disable)										
$\checkmark$	MCUXpresso IDE LinkServer	r (inc. CMSIS-DAP) pr	obes									
$\sim$	P&E Micro probes											
	SEGGER J-Link probes											
	be search options											
Se	arch again											
	emember my selection (for t	his Launch configuration	tion)									
	emember my selection (for t	nis caunch configura	uonj									
?	)			ОК		Cancel						

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



#### 28. You should see the output on the console:



### **Conclusion:**

- This lab demonstrated how to use the **tflite\_convert** utility convert a TensorFlow model into a format that can be imported and ran on an embedded system using the eIQ software platform.
- The particular model was used to classify flower images. However, the model can also be trained on new types of images by retraining it. 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 with this same process as well. By enabling machine learning in embedded systems, there's a wide world of opportunity for new applications.