eIQ Transfer Learning Lab with i.MX 8

Table of Contents

1. Introduction

This lab will 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 ran on an i.MX 8QM-MEK board. This lab can also be used with different i.MX 8 devices.

Please check also: eIQ Transfer Learning Lab with i.MX RT.

2. Requirements

- i.MX 8QM-MEK board
- Image + eIQ packages (follow *Yocto installation guide* from eIQ)
- Ubuntu Host PC (this tutorial can also run on Windows OS but might need some minor changes)

3. Software Installation

1. Install the needed Python packages:

```
$ apt install python-pip
$ pip install --upgrade  "tensorflow==1.12.*"
$ pip install PILLOW
```

2. Download the needed repository and dataset for the retraining process:

```
$ git clone https://github.com/googlecodelabs/tensorflow-for-poets-2.git
$ wget http://download.tensorflow.org/example_images/flower_photos.tgz
$ tar -xvzf flower_photos.tgz -C tensorflow-for-poets-2/tf_files/
```

4. 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.MX 8* devices. The following steps are based on this Google CodeLabs tutorial: TensorFlow for Poets.

We will retrain a MobileNet 128px 0.25 model in this lab, but different resolutions/sizes can also be used for this step. These will impact on performance and accuracy of the model.

To begin the retraining process, run:

```
$ cd tensorflow-for-poets-2
$ 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
```

Where:

- *--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 files used by an optional analysis program called TensorBoard.
- *--output_graph=tf_files/retrained_graph.pb*: Name of the retrained model in .pb format.
- *--output_labels=tf_files/retrained_labels.txt*: Text file with the labels for the model (determined by the sub-directories 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.

This might take a few minutes to complete.

Once finished, you should have the newly trained model named **retrained_graph.pb** in the **tf_files** directory. To test this model file, run the **label_image** script against any image from the dataset:

```
$ python -m scripts.label_image \
   --graph=tf_files/retrained_graph.pb \
   --input_height=128 \
   --input_width=128 \
   --image=tf_files/flower_photos/tulips/12517756805_56b74be742.jpg

Evaluation time (1-image): 0.071s

tulips (score=0.98604)
roses (score=0.01383)
dandelion (score=0.00012)
daisy (score=0.00001)
sunflowers (score=0.00000)
```

It should return the correct label for the chosen image. 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 correctly, try running the retraining script again.

5. Convert TensorFlow Model

To run the inference on the i.MX 8* devices, the retrained model needs to be converted into a TensorFlow Lite file that can be loaded onto an embedded device.

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:

```
$ 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
```

Where:

- *--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.

Once the conversion if finished, you will find the new file named **retrained_graph.tflite** inside the **tf_files** directory. Copy this file and the **retrained_labels.txt** to your SD card image in the following path:

```
$ cp tf_files/retrained_graph.tflite /path/to/your/rootfs/usr/bin/tensorflow-lite-
1.12.0/examples
$ cp tf_files/retrained_labels.txt /path/to/your/rootfs/usr/bin/tensorflow-lite-
1.12.0/examples
#or if image is already running on the board:
$ scp tf_files/retrained_graph.tflite root@<board_ip>:/usr/bin/tensorflow-lite-
1.12.0/examples
$ scp tf_files/retrained_labels.txt root@<board_ip>:/usr/bin/tensorflow-lite-
1.12.0/examples
```

6. Convert Image Files

The **label_image** example in eIQ takes in 24-bit BMP image files, so we will convert one of the images in the dataset to a BMP file. You can convert as many images as you want to test and then deploy them to the board.

To convert a image, run:

```
$ convert tf_files/flower_photos/tulips/12517756805_56b74be742.jpg -type truecolor
tf_files/tulip.bmp
```

Copy the wanted files to the SD card image in the following path:

```
$ cp tf_files/tulip.bmp /path/to/your/rootfs/usr/bin/tensorflow-lite-1.12.0/examples
#or if image is already running on the board:
$ scp tf_files/tulip.bmp root@<board_ip>:/usr/bin/tensorflow-lite-1.12.0/examples
```

7. Run Demo

After booting the board, go to the TensorFlow Lite examples folder and run the **label_image** example with the newly trained model, labels and the wanted image:

```
# cd /usr/bin/tensorflow-lite-1.12.0/examples
# ./label_image --tflite_model retrained_graph.tflite --threads 1 --image tulip.bmp --
labels retrained_labels.txt
Loaded model retrained_graph.tflite
resolved reporter
invoked
average time: 18.42 ms
0.995619: 4 tulips
0.00244371: 2 roses
0.00192906: 1 dandelion
```

You can run the command above with different images by changing the **--image** parameter. As you can see above, the **label_image** TensorFlow Lite application correctly predicts the image category on an i.MX 8* device.

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.