Utilizing Multiple CNN Models in One Application Using MAX78000 and MAX78002

Abstract

The MAX78000 [1] and MAX78002 [2] are Artificial Intelligence (AI) microcontrollers with an ultra-low power Convolutional Neural Network (CNN) inference engine to run AI edge applications on a battery-powered IoT device. The device can execute many complex CNN networks to achieve critical performance. This document describes an approach to utilize multiple CNN models in one application.

Introduction

This application note describes a method to execute a facial recognition application which is comprised of three separate models each invoked in a different task:

- The **Face Detection** CNN model detects faces in the captured image and extracts a rectangular sub-image containing only one face.
- The **Face Identification** CNN model identifies a person from their facial images by generating the embedding for a given face image.
- The **Dot Product** model outputs the dot product representing the similarity between the embedding from the given image and embeddings in the database.

Using the dot product similarity as a distance metric, the image is identified as either one of the known subjects or 'Unknown' depending on the embedding distances.

MAX78000 Facial Recognition Application

The Facial Recognition application [4] can only run on MAX78000 Feather board [3] due to the SD card support.

The **Face Detection**, **Face Identification** and **Dot Product** models are executed sequentially.

The challenge of this application is to utilize all models when they are exceeding 432KB of 8-bit weight capacity of MAX78000 CNN engine and MAX78000 internal flash memory storage. In this example, the **Face Detection** and **Dot Product** model's weights are stored in MAX78000 internal flash memory, while the **Face Identification** CNN model weights are stored in the external SD memory card and reloaded as soon as a face is detected.

The SDHC_weights sub-project can be used to store the **Face Identification** CNN weights (weights_2.h) in the SD card in binary format.

Face Detection

The **Face Detection** CNN model has 16 layers and uses 168x224 RGB images as input.

```
  Face Detection CNN:
  SUMMARY OF OPS
  Hardware: 589,595,888 ops (588,006,720 macc; 1,589,168 comp; 0 add; 0 mul; 0 bitwise)
    Layer 0: 4,327,680 ops (4,064,256 macc; 263,424 comp; 0 add; 0 mul; 0 bitwise)
    Layer 1: 11,063,808 ops (10,838,016 macc; 225,792 comp; 0 add; 0 mul; 0 bitwise)
    Layer 2: 43,502,592 ops (43,352,064 macc; 150,528 comp; 0 add; 0 mul; 0 bitwise)
    Layer 3: 86,854,656 ops (86,704,128 macc; 150,528 comp; 0 add; 0 mul; 0 bitwise)
    Layer 4: 86,854,656 ops (86,704,128 macc; 150,528 comp; 0 add; 0 mul; 0 bitwise)
    Layer 5: 86,854,656 ops (86,704,128 macc; 150,528 comp; 0 add; 0 mul; 0 bitwise)
    Layer 6: 173,709,312 ops (173,408,256 macc; 301,056 comp; 0 add; 0 mul; 0 bitwise)
    Layer 7 (backbone_conv8): 86,779,392 ops (86,704,128 macc; 75,264 comp; 0 add; 0
mul; 0 bitwise)
    Layer 8 (backbone_conv9): 5,513,088 ops (5,419,008 macc; 94,080 comp; 0 add; 0 mul;
0 bitwise)
    Layer 9 (backbone_conv10): 1,312,640 ops (1,290,240 macc; 22,400 comp; 0 add; 0 mul;
0 bitwise)
    Layer 10 (conv12_1): 647,360 ops (645,120 macc; 2,240 comp; 0 add; 0 mul; 0 bitwise)
    Layer 11 (conv12_2): 83,440 ops (80,640 macc; 2,800 comp; 0 add; 0 mul; 0 bitwise)
    Layer 12: 1,354,752 ops (1,354,752 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
    Layer 13: 40,320 ops (40,320 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
    Layer 14: 677,376 ops (677,376 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
    Layer 15: 20,160 ops (20,160 macc; 0 comp; 0 add; 0 mul; 0 bitwise)

  RESOURCE USAGE
  Weight memory: 275,184 bytes out of 442,368 bytes total (62%)
  Bias memory:   536 bytes out of 2,048 bytes total (26%)
```

Corresponding CNN weights, bias, and configuration are loaded into the CNN engine every time before execution.

```
        // Power off CNN after unloading result to clear all CNN registers.
        // It's needed to load and run other CNN model
        cnn_disable();

        // Enable CNN peripheral, enable CNN interrupt, turn on CNN clock
        // CNN clock: 50 MHz div 1
        cnn_enable(MXC_S_GCR_PCLKDIV_CNNCLKSEL_PCLK, MXC_S_GCR_PCLKDIV_CNNCLKDIV_DIV1);
        /* Configure CNN 1 to detect a face */
        cnn_1_init(); // Bring CNN state machine into consistent state
        cnn_1_load_weights(); // Load CNN kernels
        cnn_1_load_bias(); // Load CNN bias
        cnn_1_configure(); // Configure CNN state machine
```

The output of the **Face Detection** CNN model is a set of box coordinates and corresponding scores. The Non-Maximum Suppression (NMS) algorithm selects a bounding box with the highest confidence score which is displayed on TFT.

If the **Face Detection** CNN model detects a face, a rectangular sub-image containing only one face is resized to 112x112 RGB image to match the **Face Identification** CNN model input.

Face Identification

The **Face Identification** CNN model consists of 17 layers with 112x112 RGB input.

```
  Face Identification CNN:
  SUMMARY OF OPS
  Hardware: 199,784,640 ops (198,019,072 macc; 1,746,752 comp; 18,816 add; 0 mul; 0 bitwise)
    Layer 0: 11,239,424 ops (10,838,016 macc; 401,408 comp; 0 add; 0 mul; 0 bitwise)
    Layer 1: 29,403,136 ops (28,901,376 macc; 501,760 comp; 0 add; 0 mul; 0 bitwise)
    Layer 2: 58,003,456 ops (57,802,752 macc; 200,704 comp; 0 add; 0 mul; 0 bitwise)
    Layer 3: 21,876,736 ops (21,676,032 macc; 200,704 comp; 0 add; 0 mul; 0 bitwise)
    Layer 4: 7,375,872 ops (7,225,344 macc; 150,528 comp; 0 add; 0 mul; 0 bitwise)
    Layer 5: 21,826,560 ops (21,676,032 macc; 150,528 comp; 0 add; 0 mul; 0 bitwise)
    Layer 6: 1,630,720 ops (1,605,632 macc; 25,088 comp; 0 add; 0 mul; 0 bitwise)
    Layer 7: 14,450,688 ops (14,450,688 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
    Layer 8: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
    Layer 9: 12,544 ops (0 macc; 0 comp; 12,544 add; 0 mul; 0 bitwise)
    Layer 10: 3,261,440 ops (3,211,264 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
    Layer 11: 10,888,192 ops (10,838,016 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
    Layer 12: 912,576 ops (903,168 macc; 9,408 comp; 0 add; 0 mul; 0 bitwise)
    Layer 13: 10,838,016 ops (10,838,016 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
    Layer 14: 809,088 ops (802,816 macc; 6,272 comp; 0 add; 0 mul; 0 bitwise)
    Layer 15: 7,225,344 ops (7,225,344 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
    Layer 16: 22,656 ops (16,384 macc; 0 comp; 6,272 add; 0 mul; 0 bitwise)
    Layer 17: 8,192 ops (8,192 macc; 0 comp; 0 add; 0 mul; 0 bitwise)

  RESOURCE USAGE
  Weight memory: 365,408 bytes out of 442,368 bytes total (82.6%)
  Bias memory:   1,296 bytes out of 2,048 bytes total (63.3%)
```

Before loading the **Face Identification** CNN configuration, weights, and bias, the CNN engine state machine and memory must be cleared from the previous CNN model execution. One way to do that is by powering down the CNN engine by calling the ***cnn_disable()*** function.

```
        // Power off CNN after unloading result to clear all CNN registers.
        // It's needed to load and run other CNN model
        cnn_disable();

        // Enable CNN peripheral, enable CNN interrupt, turn on CNN clock
        // CNN clock: 50 MHz div 1
        cnn_enable(MXC_S_GCR_PCLKDIV_CNNCLKSEL_PCLK, MXC_S_GCR_PCLKDIV_CNNCLKDIV_DIV1);
        /* Configure CNN 2 to recognize a face */
        cnn_2_init(); // Bring state machine into consistent state
        cnn_2_load_weights_from_SD(); // Load CNN kernels from SD card
        cnn_2_load_bias(); // Reload CNN bias
        cnn_2_configure(); // Configure state machine
```

The output of the **Face Identification** CNN model is an embedding vector of length 64 corresponding to the face image. Before feeding the embedding vector to the **Dot Product** model as an input, the vector is L2 normalized.

Dot Product

The **Dot Product** model has one linear layer and uses the embedding vector of length 64 as input.

```
Dot Product CNN:
SUMMARY OF OPS
Hardware: 65,536 ops (65,536 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
  Layer 0: 65,536 ops (65,536 macc; 0 comp; 0 add; 0 mul; 0 bitwise)

RESOURCE USAGE
Weight memory: 65,536 bytes out of 442,368 bytes total (14.8%)
Bias memory:   0 bytes out of 2,048 bytes total (0.0%)
```

Again, before loading **Dot Product** CNN configuration, weights and bias, the CNN engine state machine and memory must be cleared.

```
        // Power off CNN after unloading result to clear all CNN registers.
        // It's needed to load and run other CNN model
        cnn_disable();

        // Enable CNN peripheral, enable CNN interrupt, turn on CNN clock
        // CNN clock: 50 MHz div 1
        cnn_enable(MXC_S_GCR_PCLKDIV_CNNCLKSEL_PCLK, MXC_S_GCR_PCLKDIV_CNNCLKDIV_DIV1);
        /* Configure CNN 3 for dot product */
        cnn_3_init(); // Bring state machine into consistent state
        cnn_3_load_weights(); // Load CNN kernels from
        cnn_3_load_bias(); // Reload CNN bias
        cnn_3_configure(); // Configure state machine
```

The output of the **Dot Product** model is 1024 dot product similarities, where each one represents the similarity between the given face and the faces recorded in the database. If the maximum of dot product similarities is higher than the threshold, the subject with the maximum similarity is declared as the identified face and displayed on TFT. Otherwise, the output will be 'Unknown'.

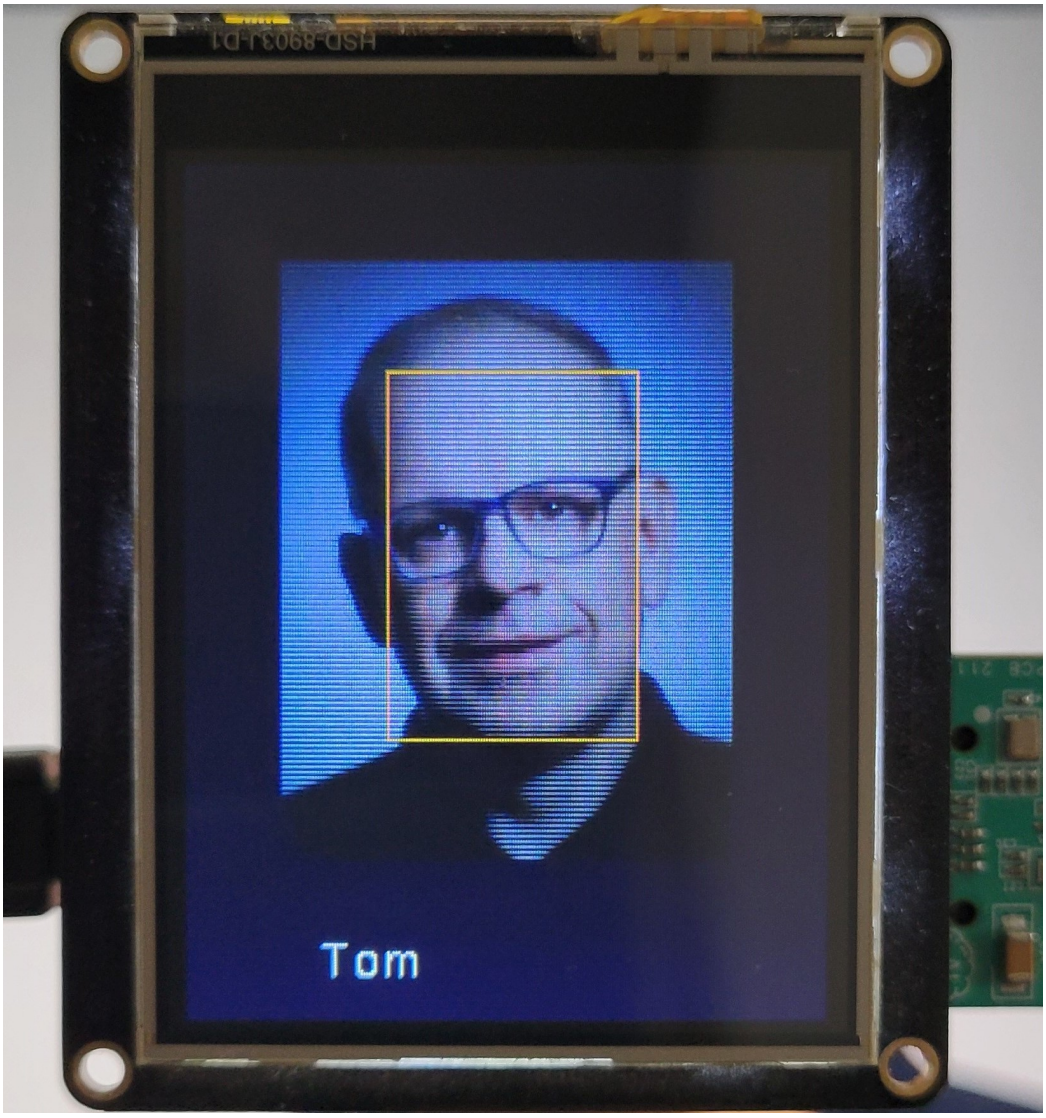

Figure 1. MAX78000 facial recognition result

MAX78002 Facial Recognition Application

The Facial Recognition application [5] runs on MAX78002EVKIT board [6].

Similar to the MAX78000 application, the **Face Detection**, **Face Identification,** and **Dot Product** models are executed sequentially.
However, the MAX78002 features a larger internal flash memory that can store all model's weights.

This application loads all weights and layer configurations only at the initialization, and with every inference, only biases are reloaded. With this approach, the overhead of switching between models is greatly reduced.

To keep all models together in the CNN memory, layer offsets must be arranged in a similar way that models follow each other.
In this example, layer offsets are arranged as shown in Table 1.

Table 1. CNN models organization

CNN Model	Start Layer	End Layer
Face Identification	0	72
Dot Product	73	73
Face Detection	74	89

To utilize this technique, the total layer count should not exceed a maximum 128, as the limitation of MAX78002.

During the synthesis, for the models that are not starting from the layer 0, passthrough layers must be added before the model layers in the network.yaml files. An example of network.yaml files can be found in AI8x-Synthesis Repository [7].

Another consideration for this application is adjusting weight offsets. There are 64 parallel processors in the MAX78002, and each one has 4096 CNN kernels that can keep 9 parameters with 8-bit precision. While adjusting weight offsets, the previous model's kernel memory usage should be considered.

In this example, weight offsets are arranged as shown in Table 2.

Table 2. CNN models weight offset

CNN Model	Weight Offset	Number of Kernels
Face Identification	0	1580
Dot Product	2000	114
Face Detection	2500	

During the synthesis, "**--start-layer**" and "**--weight-start**" arguments can be used to add passthrough layers and weight offsets. Examples of synthesis scripts are provided in AI8x-Synthesis Repository [7].

At the initialization, all model weights and configurations are loaded.

```
    cnn_1_enable(MXC_S_GCR_PCLKDIV_CNNCLKSEL_IPLL, MXC_S_GCR_PCLKDIV_CNNCLKDIV_DIV4);
    cnn_1_init(); // Bring state machine into consistent state
    cnn_1_load_weights(); // Load kernels of CNN_1
    cnn_1_configure(); // Configure CNN_1 layers

    cnn_2_load_weights(); // Load kernels of CNN_2
    cnn_2_configure(); // Configure CNN_2 layers

  cnn_3_load_weights(); // Load kernels of CNN_3
    cnn_3_configure(); // Configure CNN_3 layers
```

Face Detection

The **Face Detection** CNN model has 16 layers and uses 168x224 RGB images as input.

```
  Face Detection CNN:
  SUMMARY OF OPS
    Hardware: 589,595,888 ops (588,006,720 macc; 1,589,168 comp; 0 add; 0 mul; 0 bitwise)
      Layer 74: 4,327,680 ops (4,064,256 macc; 263,424 comp; 0 add; 0 mul; 0 bitwise)
      Layer 75: 11,063,808 ops (10,838,016 macc; 225,792 comp; 0 add; 0 mul; 0 bitwise)
      Layer 76: 43,502,592 ops (43,352,064 macc; 150,528 comp; 0 add; 0 mul; 0 bitwise)
      Layer 77: 86,854,656 ops (86,704,128 macc; 150,528 comp; 0 add; 0 mul; 0 bitwise)
      Layer 78: 86,854,656 ops (86,704,128 macc; 150,528 comp; 0 add; 0 mul; 0 bitwise)
      Layer 79: 86,854,656 ops (86,704,128 macc; 150,528 comp; 0 add; 0 mul; 0 bitwise)
      Layer 80: 173,709,312 ops (173,408,256 macc; 301,056 comp; 0 add; 0 mul; 0 bitwise)
      Layer 81 (backbone_conv8): 86,779,392 ops (86,704,128 macc; 75,264 comp; 0 add; 0 mul; 0 bitwise)
      Layer 82 (backbone_conv9): 5,513,088 ops (5,419,008 macc; 94,080 comp; 0 add; 0 mul; 0 bitwise)
      Layer 83 (backbone_conv10): 1,312,640 ops (1,290,240 macc; 22,400 comp; 0 add; 0 mul; 0 bitwise)
      Layer 84 (conv12_1): 647,360 ops (645,120 macc; 2,240 comp; 0 add; 0 mul; 0 bitwise)
      Layer 85 (conv12_2): 83,440 ops (80,640 macc; 2,800 comp; 0 add; 0 mul; 0 bitwise)
      Layer 86: 1,354,752 ops (1,354,752 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 87: 40,320 ops (40,320 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 88: 677,376 ops (677,376 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 89: 20,160 ops (20,160 macc; 0 comp; 0 add; 0 mul; 0 bitwise)

    RESOURCE USAGE
    Weight memory: 275,184 bytes out of 2,396,160 bytes total (11%)
    Bias memory:   536 bytes out of 8,192 bytes total (7%)
```

As the application starts with the **Face Detection** model, the corresponding bias values are loaded into the CNN engine before every execution.
Then CNN_1 and FIFO configurations are executed for **Face Detection**.

```
    cnn_1_load_bias(); // Load bias data of CNN_1
    // Bring CNN_1 state machine of FaceDetection model into consistent state
    *((volatile uint32_t *) 0x51000000) = 0x00100008; // Stop SM
    *((volatile uint32_t *) 0x51000008) = 0x00004a59; // Layer count
    *((volatile uint32_t *) 0x52000000) = 0x00100008; // Stop SM
    *((volatile uint32_t *) 0x52000008) = 0x00004a59; // Layer count
    *((volatile uint32_t *) 0x53000000) = 0x00100008; // Stop SM
    *((volatile uint32_t *) 0x53000008) = 0x00004a59; // Layer count
    *((volatile uint32_t *) 0x54000000) = 0x00100008; // Stop SM
    *((volatile uint32_t *) 0x54000008) = 0x00004a59; // Layer count
    // Disable FIFO control
    *((volatile uint32_t *) 0x50000000) = 0x00000000;
```

Similar to the MAX78000 application, the output of **Face Detection** CNN model is a set of box coordinates and their corresponding scores. The Non-Maximum Suppression (NMS) algorithm selects a bounding box with the highest confidence score which is displayed on TFT.

If the **Face Detection** CNN model detects a face, a rectangular sub-image that contains only one face is selected and resized to 112x112 RGB image to match **Face Identification** CNN model input.

Face Identification

The **Face Identification** CNN model has 73 layers and uses 112x112 RGB image as an input.

```
  Face Identification CNN:
  SUMMARY OF OPS
    Hardware: 445,470,720 ops (440,252,416 macc; 4,848,256 comp; 370,048 add; 0 mul; 0 bitwise)
      Layer 0: 22,478,848 ops (21,676,032 macc; 802,816 comp; 0 add; 0 mul; 0 bitwise)
      Layer 1: 2,809,856 ops (1,806,336 macc; 1,003,520 comp; 0 add; 0 mul; 0 bitwise)
      Layer 2: 231,612,416 ops (231,211,008 macc; 401,408 comp; 0 add; 0 mul; 0 bitwise)
      Layer 3: 1,404,928 ops (903,168 macc; 501,760 comp; 0 add; 0 mul; 0 bitwise)
      Layer 4: 6,422,528 ops (6,422,528 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 5: 6,522,880 ops (6,422,528 macc; 100,352 comp; 0 add; 0 mul; 0 bitwise)
      Layer 6: 1,003,520 ops (903,168 macc; 100,352 comp; 0 add; 0 mul; 0 bitwise)
      Layer 7: 6,422,528 ops (6,422,528 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 8: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 9: 50,176 ops (0 macc; 0 comp; 50,176 add; 0 mul; 0 bitwise)
      Layer 10: 6,522,880 ops (6,422,528 macc; 100,352 comp; 0 add; 0 mul; 0 bitwise)
      Layer 11: 1,003,520 ops (903,168 macc; 100,352 comp; 0 add; 0 mul; 0 bitwise)
      Layer 12: 6,422,528 ops (6,422,528 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 13: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 14: 50,176 ops (0 macc; 0 comp; 50,176 add; 0 mul; 0 bitwise)
```

```
      Layer 14: 50,176 ops (0 macc; 0 comp; 50,176 add; 0 mul; 0 bitwise)
      Layer 15: 6,522,880 ops (6,422,528 macc; 100,352 comp; 0 add; 0 mul; 0 bitwise)
      Layer 16: 1,003,520 ops (903,168 macc; 100,352 comp; 0 add; 0 mul; 0 bitwise)
      Layer 17: 6,422,528 ops (6,422,528 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 18: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 19: 50,176 ops (0 macc; 0 comp; 50,176 add; 0 mul; 0 bitwise)
      Layer 20: 6,522,880 ops (6,422,528 macc; 100,352 comp; 0 add; 0 mul; 0 bitwise)
      Layer 21: 1,003,520 ops (903,168 macc; 100,352 comp; 0 add; 0 mul; 0 bitwise)
      Layer 22: 6,422,528 ops (6,422,528 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 23: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 24: 50,176 ops (0 macc; 0 comp; 50,176 add; 0 mul; 0 bitwise)
      Layer 25: 13,045,760 ops (12,845,056 macc; 200,704 comp; 0 add; 0 mul; 0 bitwise)
      Layer 26: 702,464 ops (451,584 macc; 250,880 comp; 0 add; 0 mul; 0 bitwise)
      Layer 27: 6,422,528 ops (6,422,528 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 28: 6,472,704 ops (6,422,528 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
      Layer 29: 501,760 ops (451,584 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
      Layer 30: 6,422,528 ops (6,422,528 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 31: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 32: 25,088 ops (0 macc; 0 comp; 25,088 add; 0 mul; 0 bitwise)
      Layer 33: 6,472,704 ops (6,422,528 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
      Layer 34: 501,760 ops (451,584 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
      Layer 35: 6,422,528 ops (6,422,528 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 36: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 37: 25,088 ops (0 macc; 0 comp; 25,088 add; 0 mul; 0 bitwise)
      Layer 38: 6,472,704 ops (6,422,528 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
      Layer 39: 501,760 ops (451,584 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
      Layer 40: 6,422,528 ops (6,422,528 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 41: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 42: 25,088 ops (0 macc; 0 comp; 25,088 add; 0 mul; 0 bitwise)
      Layer 43: 6,472,704 ops (6,422,528 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
      Layer 44: 501,760 ops (451,584 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
      Layer 45: 6,422,528 ops (6,422,528 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 46: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 47: 25,088 ops (0 macc; 0 comp; 25,088 add; 0 mul; 0 bitwise)
      Layer 48: 6,472,704 ops (6,422,528 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
      Layer 49: 501,760 ops (451,584 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
      Layer 50: 6,422,528 ops (6,422,528 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 51: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 52: 25,088 ops (0 macc; 0 comp; 25,088 add; 0 mul; 0 bitwise)
      Layer 53: 6,472,704 ops (6,422,528 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
      Layer 54: 501,760 ops (451,584 macc; 50,176 comp; 0 add; 0 mul; 0 bitwise)
      Layer 55: 6,422,528 ops (6,422,528 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 56: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 57: 25,088 ops (0 macc; 0 comp; 25,088 add; 0 mul; 0 bitwise)
      Layer 58: 12,945,408 ops (12,845,056 macc; 100,352 comp; 0 add; 0 mul; 0 bitwise)
      Layer 59: 351,232 ops (225,792 macc; 125,440 comp; 0 add; 0 mul; 0 bitwise)
      Layer 60: 3,211,264 ops (3,211,264 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 61: 1,618,176 ops (1,605,632 macc; 12,544 comp; 0 add; 0 mul; 0 bitwise)
      Layer 62: 125,440 ops (112,896 macc; 12,544 comp; 0 add; 0 mul; 0 bitwise)
      Layer 63: 1,605,632 ops (1,605,632 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
```

```
      Layer 63: 1,605,632 ops (1,605,632 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 64: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 65: 6,272 ops (0 macc; 0 comp; 6,272 add; 0 mul; 0 bitwise)
      Layer 66: 1,618,176 ops (1,605,632 macc; 12,544 comp; 0 add; 0 mul; 0 bitwise)
      Layer 67: 125,440 ops (112,896 macc; 12,544 comp; 0 add; 0 mul; 0 bitwise)
      Layer 68: 1,605,632 ops (1,605,632 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 69: 0 ops (0 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
      Layer 70: 6,272 ops (0 macc; 0 comp; 6,272 add; 0 mul; 0 bitwise)
      Layer 71: 809,088 ops (802,816 macc; 6,272 comp; 0 add; 0 mul; 0 bitwise)
      Layer 72: 14,464 ops (8,192 macc; 0 comp; 6,272 add; 0 mul; 0 bitwise)

    RESOURCE USAGE
    Weight memory: 909,952 bytes out of 2,396,160 bytes total (38.0%)
    Bias memory:   7,296 bytes out of 8,192 bytes total (89.1%)
```

To run the **Face Identification** model, the corresponding bias values are loaded to CNN before every execution.
Then CNN_2 and FIFO configurations are executed for **Face Identification**.

```
    cnn_2_load_bias(); // Load bias data of CNN_2
    // Bring CNN_2 state machine of FaceID model into consistent state
    *((volatile uint32_t *) 0x51000000) = 0x00108008; // Stop SM
    *((volatile uint32_t *) 0x51000008) = 0x00000048; // Layer count
    *((volatile uint32_t *) 0x52000000) = 0x00108008; // Stop SM
    *((volatile uint32_t *) 0x52000008) = 0x00000048; // Layer count
    *((volatile uint32_t *) 0x53000000) = 0x00108008; // Stop SM
    *((volatile uint32_t *) 0x53000008) = 0x00000048; // Layer count
    *((volatile uint32_t *) 0x54000000) = 0x00108008; // Stop SM
    *((volatile uint32_t *) 0x54000008) = 0x00000048; // Layer count
    // Enable FIFO control
    *((volatile uint32_t *) 0x50000000) = 0x00001108; // FIFO control
```

The output of the **Face Identification** CNN model is 64 length embedding vector that corresponds to the input facial image. Before feeding the embedding vector to the **Dot Product** model as an input, the vector is L2 normalized.

Dot Product

The **Dot Product** model has one linear layer and uses an embedding vector of length 64 as input.

```
Dot Product CNN:
SUMMARY OF OPS
  Hardware: 65,536 ops (65,536 macc; 0 comp; 0 add; 0 mul; 0 bitwise)
    Layer 73: 65,536 ops (65,536 macc; 0 comp; 0 add; 0 mul; 0 bitwise)

  RESOURCE USAGE
  Weight memory: 65,536 bytes out of 2,396,160 bytes total (2.7%)
  Bias memory:   0 bytes out of 8,192 bytes total (0.0%)
```

To run the **Dot Product** model, the corresponding bias values are loaded to the CNN engine before every execution.
Then CNN_3 and FIFO configurations are executed for **Dot Product**.

```
    cnn_3_load_bias(); // Load bias data of CNN_3
    //Dot product cnn state machine configuration
    *((volatile uint32_t *) 0x51000000) = 0x00100008; // Stop SM
    *((volatile uint32_t *) 0x51000008) = 0x00004949; // Layer count
    *((volatile uint32_t *) 0x52000000) = 0x00100008; // Stop SM
    *((volatile uint32_t *) 0x52000008) = 0x00004949; // Layer count
    *((volatile uint32_t *) 0x53000000) = 0x00100008; // Stop SM
    *((volatile uint32_t *) 0x53000008) = 0x00004949; // Layer count
    *((volatile uint32_t *) 0x54000000) = 0x00100008; // Stop SM
    *((volatile uint32_t *) 0x54000008) = 0x00004949; // Layer count
    // Disable FIFO control
    *((volatile uint32_t *) 0x50000000) = 0x00000000;
```

The output of the **Dot Product** model is 1024 dot product similarities, where each one represents the similarity between the given face and the faces recorded in the database. If the maximum of dot product similarities is higher than the threshold, the subject with the maximum similarity is declared as the identified face and displayed on TFT. Otherwise, the output will be 'Unknown'.

Adding the Images of a New Subject

The application allows adding a new subject to the database. By pressing the "**Record**" button on the touchscreen, the subject name can be entered.

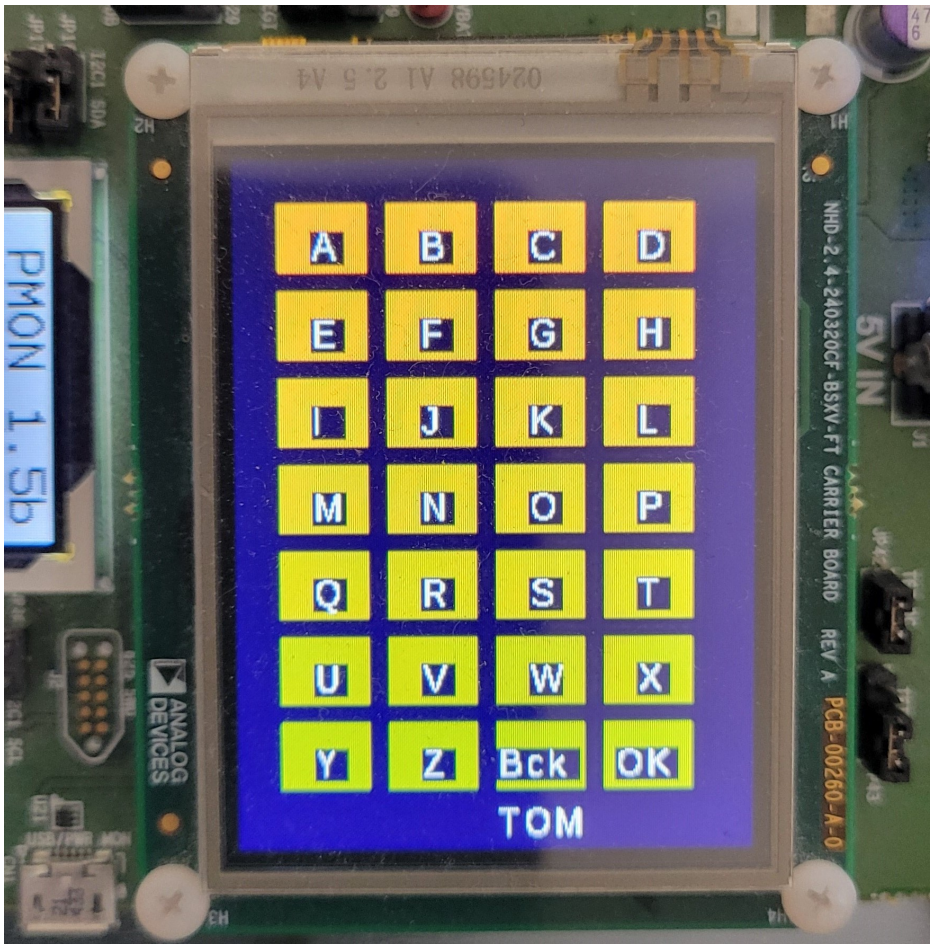

Figure 2. Entering a subject name

The next step is to capture the subject's face with the camera. Pressing "**OK**" button to add the captured image to the database or "**Retry**" to recapture.

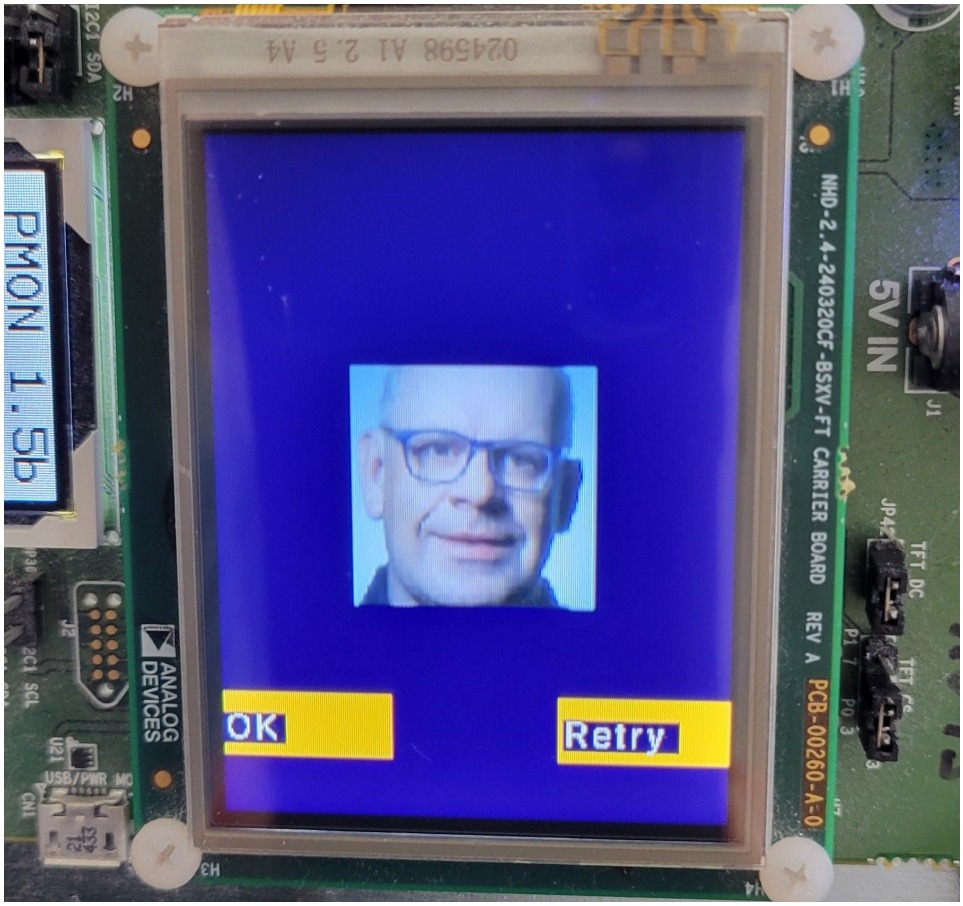

Figure 3. Capturing a subject face

The application calculates an embedding vector based on the new subject's face and stores it in an internal flash database. The **Dot Product** CNN model uses an embedding database to recognize a subject and makes a final decision.

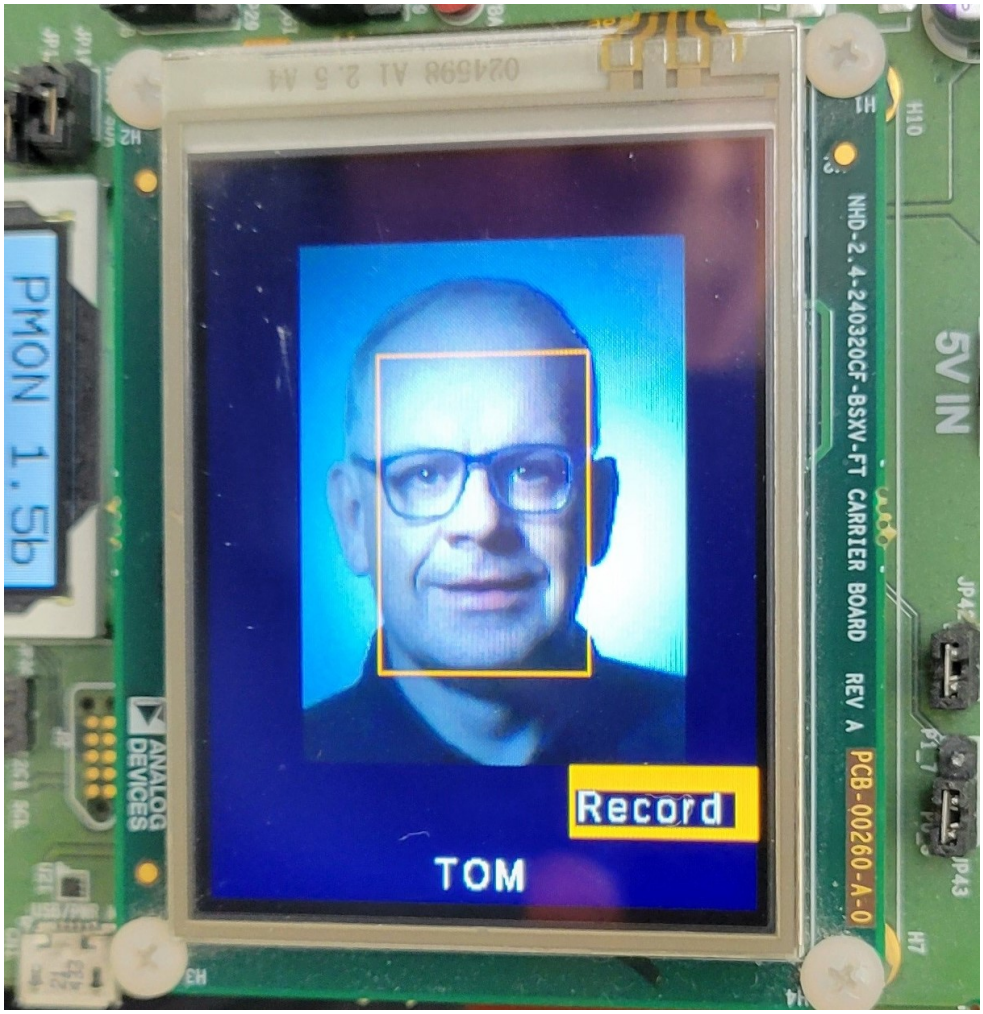

Figure 4. MAX78002 facial recognition result

Conclusion

The ultra-low power Convolutional Neural Network (CNN) inference engine makes MAX7800x Microcontrollers ideal for battery-powered IoT applications. Utilizing multiple models on the MAX78000 [1] and MAX78002 [2] AI microcontrollers enables the implementation of very complex applications in an energy-efficient way.

References

[1] MAX78000 Data Sheet

[2] MAX78002 Data Sheet

[3] MAX78000FTHR Evaluation Kit Data Sheet

[4] MAX78000FTHR Facial Recognition Application

[5] MAX78002EVKIT Facial Recognition Application

[5] MAX78002 AI Facial Recognition Application

[6] MAX78002 Evaluation Kit Data Sheet

[7] AI8X-Synthesis Repository