M480 Keyword spotting

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This sample code uses machine learning to implement keyword spotting on the M480.
BSP Version	M480 Series BSP CMSIS V3.04.000
Hardware	NuMaker-PFM-M487 Ver 3.0

1 Function Description

1.1 Introduction

A complete deep learning speech recognition system requires two platforms. As shown in Figure 1-1, one is PC platform. User can program the deep learning code and train the model by Tensorflow and Python. Due to the supervised learning for the training mode, it is necessary to give the system a large amount of training data and labels. Then the user can extract the features of speech data and train the model by deep neural networks (DNN). Until the system reaches the optimization, the user evaluates the accuracy by modifying the training model repeatedly. The other platform is Nuvoton NuMaker-PFM-M487 development board. The speech recognition system can be implemented based on the training parameters from PC platform.

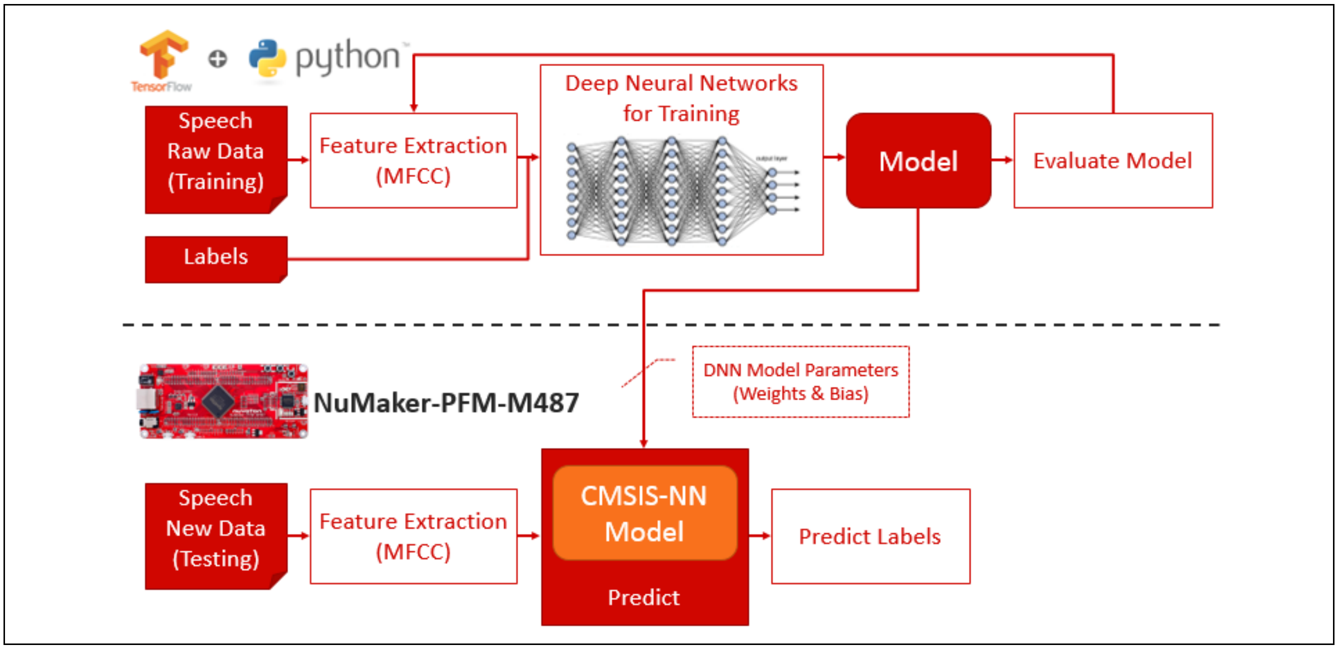

TensorFlow
python
Speech Raw Data (Training)
Feature Extraction (MFCC)
Deep Neural Networks for Training
Model
Evaluate Model
Labels
NuMaker-PFM-M487
DNN Model Parameters (Weights & Bias)
Speech New Data (Testing)
Feature Extraction (MFCC)
CMSIS-NN Model
Predict
Predict Labels

Figure 1-1 The Flow for Speech Recognition System

1.2 Introduction to Deep Learning

Machine learning is a branch of Artificial Intelligence (AI). The operation process is to use an algorithm to train a large amount of data. After the training is completed, a model will be generated. When there is new data in the future, user can predict the new data using the training model. Machine learning applications are quite extensive, such as recommendation engines, targeted advertising, demand forecasting, spam filtering, medical diagnostics, natural languages processing, search engines, fraud detection, securities analysis, visual identification, speech recognition, handwriting recognition, and more.

Deep learning is a branch of machine learning. It is the fastest growing field of artificial intelligence. There are several deep learning frameworks, such as Deep Neural Networks (DNN) and Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), etc.. Actually the answer that the user should use what kind of architectures is not certain. It depends on the size, nature, acceptable calculating time, the urgency of learning, and what you want to do with this data. Deep learning is particularly effective in applications such as visual recognition, speech recognition, natural language processing, and biomedical applications.

Deep learning is to let the machine simulate the working mode of the human brain, and thus has the same learning ability as human beings. However, the human neural network is too complicated to be simulated. The neurons are divided into multiple levels to simulate the neural network. The neural network usually has an input layer, an output layer and a number of hidden layers that can be trained. More than 2 hidden layers can be called Deep Learning.

If the users do not have a basic understanding of deep learning, the course of deep learning can be used to get the most out of this document.

1.2.1 Introduction to Deep Neural Networks (DNN)

Deep Neural Networks (DNN) is the most basic discriminant model for deep learning. It can be trained by the backpropagation algorithm. The weights can be updated iteratively by the gradient descent method:

$$\Delta w_{ij}(t+1) = \Delta w_{ij}(t) + \eta \frac{\partial C}{\partial w_{ij}}$$

η is the Learning Rate, C is the Loss Function and the choice of the function is related to the Active Function. For this document as example, in order to solve the problem of Multi-Class Classification supervised learning, choose Rectified Linear Unit (ReLU) as the activation function, and use Cross Entropy as the loss function. The definition of cross entropy is as follows:

$$C = -\sum_j d_j \log(p_j)$$

d_j represents the target probability of output unit j, p_j represents the probability output unit j after applying the activation function; and the Softmax Function of the output layer is defined as follows:

$$p_j = \frac{exp(x_i)}{\sum_k exp(x_k)}$$

p_j represents the probability of category j. x_i and x_k are inputs to units i and k, respectively. The network architecture of the deep neural networks is shown in Figure 1-2.

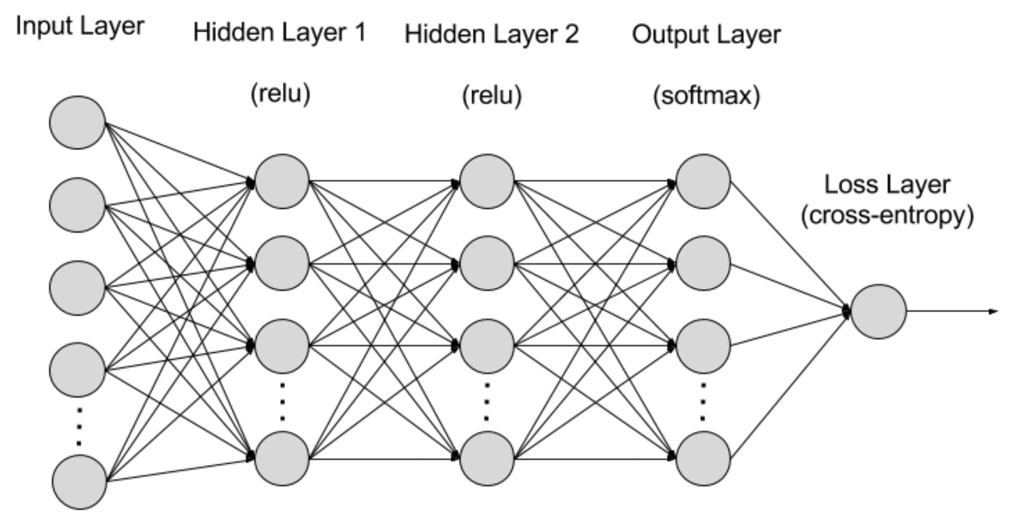


Figure 1-2 The Architecture of Deep Neural Networks (DNN)

1.3 Introduction to TensorFlow

TensorFlow is an open source code library provided by Google. Google has many products that use TensorFlow technology to develop deep learning and machine learning functions, such as Gmail filtering spam, Google voice recognition, Google image recognition, Google Translate, etc. The introduction to deep learning has described the core of deep learning and simulate neural networks with tensor (matrix) operations. Accordingly, the main design of TensorFlow is to maximize the performance of matrix operations and to develop on different platforms.

1.3.1 TensorFlow Architecture

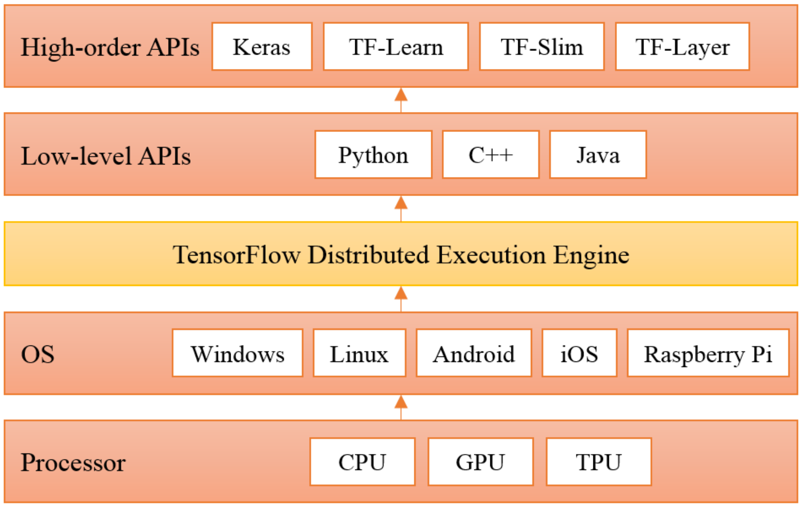


Figure 1-3 TensorFlow Architecture Diagram

The following describes the details of the above architecture diagram, starting from the bottom:

- Processor: TensorFlow can be executed on CPU, GPU, TPU.
  - CPU:Each computer has a central processing unit (CPU) that can execute TensorFlow. It's enough to use CPU for this speech recognition system.
  - GPU:Graphics processor with thousands of small and high-efficiency cores that harness the power of parallel computing.
  - TPU:The Tensor Processing Unit is a proprietary chip developed by Google's Artificial Intelligence and has better execution capabilities than the GPU.

- Platform

  TensorFlow is a cross-platform capability that can be implemented on current mainstream platforms. This document uses the Windows 10 operating system.

- TensorFlow Distributed Execution Engine

  In deep learning, the most time-consuming is the training of the model, especially the large-scale deep learning model, which must be trained with a large amount of data. TensorFlow has the ability of distributed computing, which can perform model training on hundreds of machines at the same time, greatly shorten the time of model training.

- Low-level APIs

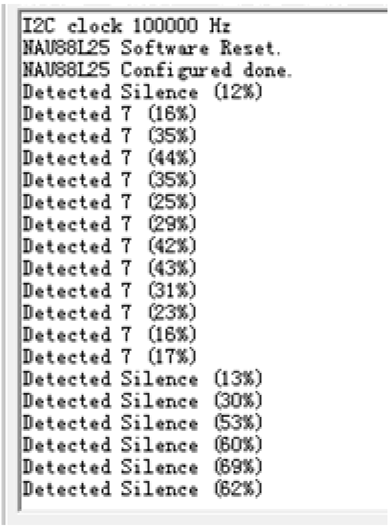
TensorFlow is available in a variety of programming languages and Python has a concise, easy-to-learn, high-productivity, object-oriented, and functional dynamic language that is widely used. The deep learning code of this document is also developed by Python.

- High-order APIs

  TensorFlow is a relatively low-level deep learning APIs. When designing a model, users must design the underlying operations such as tensor product and convolution. Therefore, this document is matched with the high-order API – Keras, which makes developers use more concise and readable codes to construct a variety of complex deep learning models.

1.4 Demo Result

Connect the microphone and speak the English number and print out the result of the identification.



```
I2C clock 100000 Hz
NAU88L25 Software Reset.
NAU88L25 Configured done.
Detected Silence (12%)
Detected 7 (16%)
Detected 7 (35%)
Detected 7 (44%)
Detected 7 (35%)
Detected 7 (25%)
Detected 7 (29%)
Detected 7 (42%)
Detected 7 (43%)
Detected 7 (31%)
Detected 7 (23%)
Detected 7 (16%)
Detected 7 (17%)
Detected Silence (13%)
Detected Silence (30%)
Detected Silence (53%)
Detected Silence (60%)
Detected Silence (69%)
Detected Silence (62%)
```

2 Code Description

Figure 2-1 shows the program flow of the NuMicro® M480 series microcontroller. The block in the figure is the main function in the program. Users can easily understand the architecture of the program.

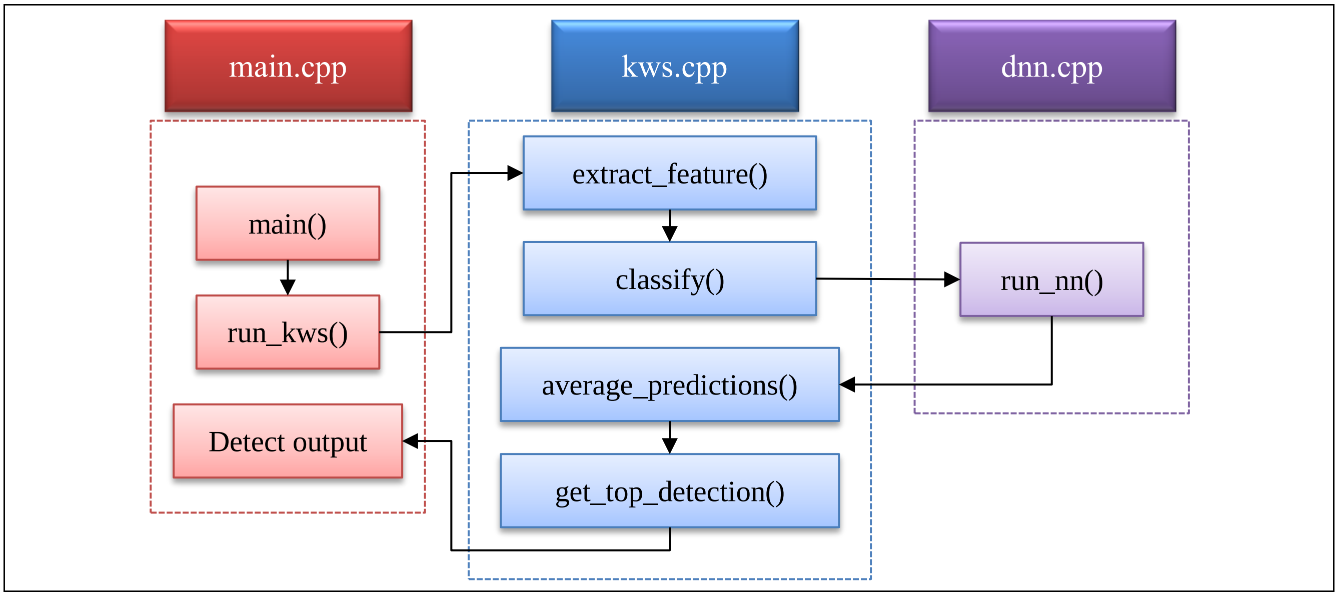


Figure 2-1 The program flow of the NuMicro® M480 series microcontroller

2.1.1 KWS main

```
/* Output Class */
char output_class[12][10] = { "Silence", "Unknown", "One", "Two", "Three", "Four",
"Five" ,"Six", "Seven", "Eight", "Nine", "Zero"};

void run_kws()
{
  //Averaging window for smoothing out the output predictions
  int averaging_window_len = 3;  //i.e. average over 3 inferences or 240ms

  kws->extract_features(2); //extract mfcc features
  kws->classify();          //classify using dnn
  kws->average_predictions(averaging_window_len);
  max_ind = kws->get_top_detection(kws->averaged_output);

  printf("Detected %s\r\n",output_class[max_ind]);
}
```

2.1.2 KWS Function (using MFCC for feature extraction and using DNN model)

```
/* This overloaded function is used in streaming audio case */
void KWS::extract_features(uint16_t num_frames)
{
  //move old features left
  memmove(mfcc_buffer,mfcc_buffer+(num_frames*NUM_MFCC_COEFFS),(NUM_FRAMES-
num_frames)*NUM_MFCC_COEFFS);
  //compute features only for the newly recorded audio
  int32_t mfcc_buffer_head = (NUM_FRAMES-num_frames)*NUM_MFCC_COEFFS;
  for (uint16_t f = 0; f < num_frames; f++) {
    mfcc->mfcc_compute(audio_buffer+(f*FRAME_SHIFT),2,&mfcc_buffer[mfcc_buffer_head]);
    mfcc_buffer_head += NUM_MFCC_COEFFS;
  }
}
void KWS::classify()
{
  nn->run_nn(mfcc_buffer, output);
  // Softmax
  arm_softmax_q7(output,OUT_DIM,output);
  //do any post processing here
}

void KWS::average_predictions(int window_len)
{
  //shift right old predictions
  for(int i=window_len-1;i>0;i--) {
    for(int j=0;j<OUT_DIM;j++)
      predictions[i][j]=predictions[i-1][j];
  }
  //add new predictions
  for(int j=0;j<OUT_DIM;j++)
    predictions[0][j]=output[j];
  //compute averages
  int sum;
  for(int j=0;j<OUT_DIM;j++) {
    sum=0;
    for(int i=0;i<window_len;i++)
      sum += predictions[i][j];
    averaged_output[j] = (q7_t)(sum/window_len);
  }
}
```

```
int KWS::get_top_detection(q7_t* prediction)
{
  int max_ind=0;
  int max_val=-128;
  for(int i=0;i<OUT_DIM;i++) {
    if(max_val<prediction[i]) {
      max_val = prediction[i];
      max_ind = i;
    }
  }
  return max_ind;
}
```

2.1.3 NN Function

```
void DNN::run_nn(q7_t* in_data, q7_t* out_data)
{
// Run all layers


// IP1
arm_fully_connected_q7(in_data, ip1_wt, IN_DIM, IP1_OUT_DIM, 1, 7, ip1_bias, ip1_out,
vec_buffer);
// RELU1
arm_relu_q7(ip1_out, IP1_OUT_DIM);
// IP2
arm_fully_connected_q7(ip1_out, ip2_wt, IP1_OUT_DIM, IP2_OUT_DIM, 2, 8, ip2_bias, ip2_out,
vec_buffer);
// RELU2
arm_relu_q7(ip2_out, IP2_OUT_DIM);
// IP3
arm_fully_connected_q7(ip2_out, ip3_wt, IP2_OUT_DIM, IP3_OUT_DIM, 2, 9, ip3_bias, ip3_out,
vec_buffer);
// RELU3
arm_relu_q7(ip3_out, IP3_OUT_DIM);
// IP4
arm_fully_connected_q7(ip3_out, ip4_wt, IP3_OUT_DIM, OUT_DIM, 0, 6, ip4_bias, out_data,
vec_buffer);


}
```

3 Software and Hardware Environment

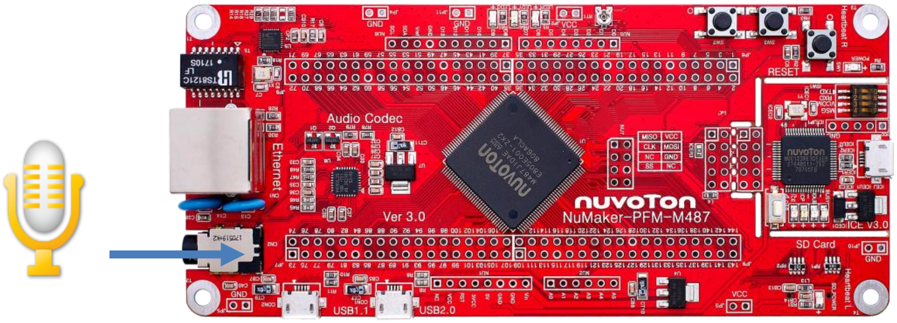
Software environment

- BSP version
  - M480 Series BSP CMSIS V3.04.000
- IDE version
  - Keil uVersion 5.26

Hardware environment

- Circuit components
  - NuMaker-IoT-M487 or other M480 Development Board
  - 3.5 mm microphone
- Diagram

Connect microphone on NuMaker-PFM-M487.



4 Directory Information

🗁 EC_M480_KWS_V1.00			
	🗁 Library		Sample code header and source files
		🗁 CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
		🗁 Device	CMSIS compliant device header file
		🗁 StdDriver	All peripheral driver header and source files
	🗁 ML_PCTool		Machine Learning python source files
	🗁 SampleCode		
		🗁 ExampleCode	Source file of example code

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click M480_KWS.uvproj
2. Enter Keil compile mode
    a. Build
    b. Download
    c. Start/Stop debug session
3. Enter debug mode
    a. Run

6 Revision History

Date	Revision	Description
Oct. 23, 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, “Insecure Usage”.

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer’s risk, and in the event that third parties lay claims to Nuvoton as a result of customer’s Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.