Run RNN on K64 for audio noise reduction

Contents

1 Introduction

Ordinary MCUs are limited by resources, and it is difficult to do some complex deep learning. But although it is difficult, it can still be done.

Last time we used CNN for handwritten recognition. This time ,we use RNN for audio noise reduction. This audio noise reduction uses fixed-point noise reduction. MFCC and gain are used as input parameters for training.

MFCC, Mel frequency cepstrum, is often used for audio feature extraction. RNN achieves the effect of noise reduction by adjusting the gain of different frequencies.

2 Experiment

2.1 Required tools: frdm-k64, python 3.7, Pip, IAR

2.2 Download the source code of deep learning framework,
https://github.com/majianjia/nnom This is a pure C framework that does not rely on hardware structure. Transplantation is very convenient

2.3 we select the example ‘bubble’ to add the Inc, port and Src folders in NNoM to the project, as shown in the figure

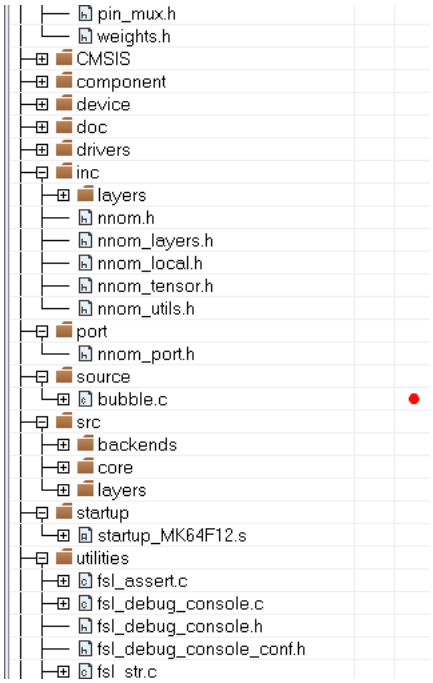

Figure 1

Open the file ‘port.h’ . The definitation of NNOM_LOG is changed to PRINTF (__ VA_ ARGS__) ,

Open the ICF file, and change the heap size to 0x5000, define symbol__ size_ heap__ = 0x5000;

Malloc, which is used in this library, allocates memory from here. If it is small, it can't run the network

2.4 Transplant fatfs file system to bubble project

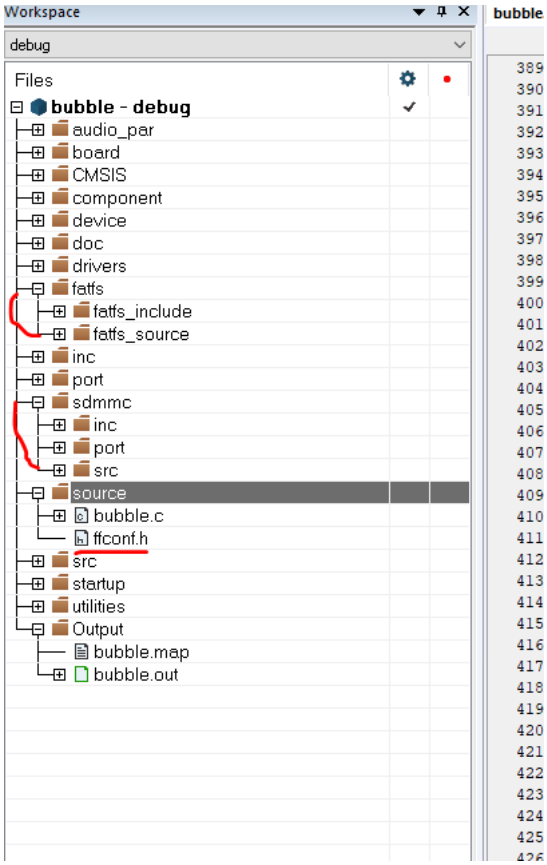

Figure 2

2.5 In the bubble.c file, add the following header file

```
#include "nnom_port.h"
#include "nnom.h"
#include "weights.h"
#include "denoise_weights.h"
#include "mfcc.h"
#include "wav.h"
#include "equalizer_coeff.h"
```

2.6 Find main.c under 'examples\rnn-denoise' in the framework, 'main_arm.c' is used for stm32. Main.c is provided for windows running routines. This code is used to open the audio file, then run the network, and finally generate noise reduction. Use this code to facilitate experiments. The APIs for file operations in this file need to be manually changed to MCU APIs. I added a progress bar display. You can refer the attachment.

2.7 This example also uses DSP, so DSP support needs to be turned on, as shown in the figure

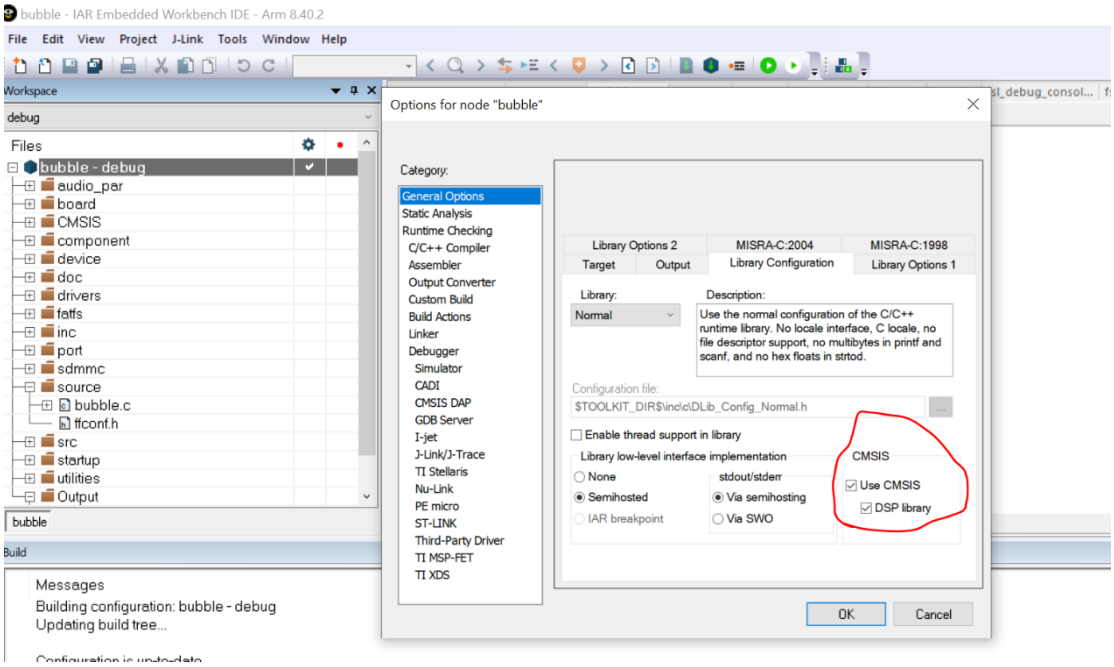

Figure 3

2.8 Add macro

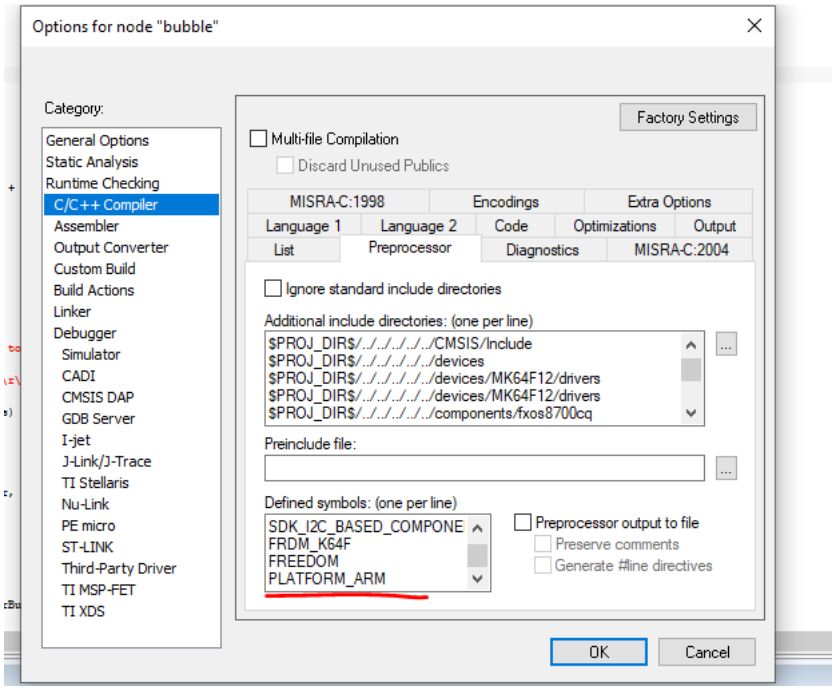

Figure 4

2.9 After the compilation is passed, download the program. The tested wav needs to be named ‘sample.wav’, and the mcu will reduce noise for it. Finnally mcu will generate ‘filtered_sample.wav’. Plug the SD card into the computer to listen to the noise-reduced audio. This is serial message.

Figure 5

This is the audio analyzed by the audio analysis software. The upper part is the noise reduced, and the lower part is the original sound. You can see that a lot of noise has been suppressed.

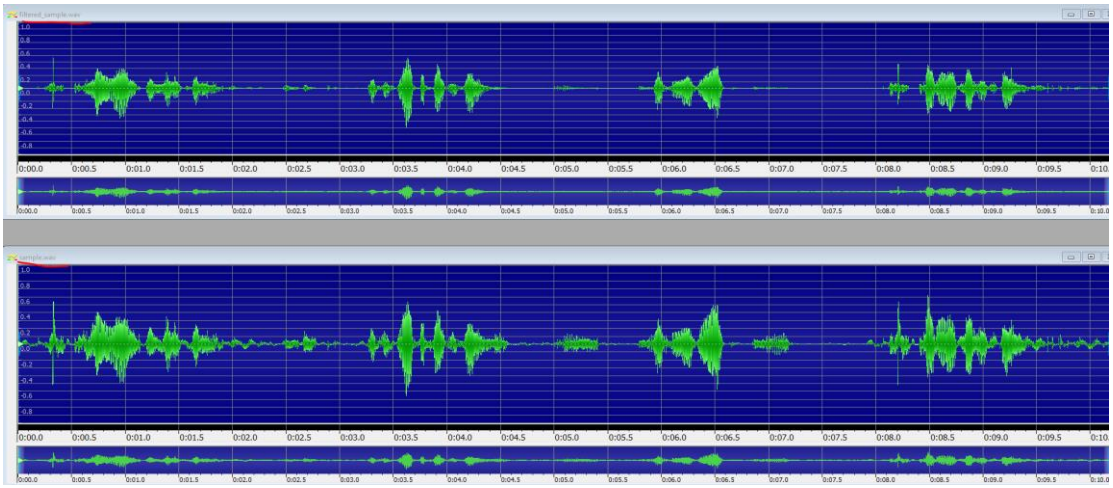

Figure 6

3 training

Through the above steps, we have achieved audio noise reduction, so how is this data trained? You need to install tensorflow, and keras. How to train is written in the README_CN.md file. This tensorflow needs to install the gpu version, and needs to install cuda. Please check the installation yourself.

3.1 Download the voice data https://github.com/microsoft/MS-SNSD and put it under the MS-SNSD folder

3.2 Use pip3 to install the tools in requirements.txt

3.3 Run 'noisyspeech_synthesizer.py' directly, this will report an error. You need to change the

15-line float to int type.

3.4 Run the script gen_dataset.py to generate mfcc and gain

3.5 Run main.py, this will generate the noise file of the sample, which can be used for testing and also generated ‘weights.h,denoise_weights.h,equalizer_coeff.h’