Baby Cry Detection Guide

Guanyu Li
Wei Xin
Li-Yu Wu
Matt Yushen Ng

Summary:

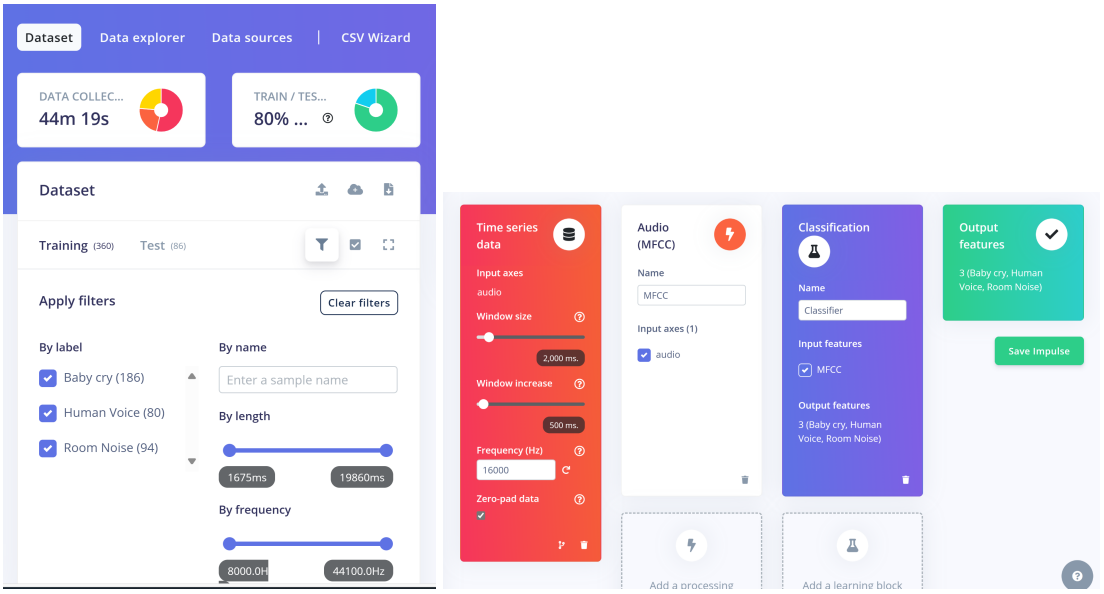
This guide shows how to use the MAX9814 microphone(with BBG) amplifier to continuously detect surrounding noise and determine the type of noise via machine learning. Please notice that other ADC-powered sensors may not operate normally as the detection process is active. If readers desire to work with baby cry detection programs and others concurrently, please devise a solution to avoid displaying gaps since one must be deactivated for another to work. Furthermore, the program is not designed to work in a public place. Please be aware that detection accuracy can be weakened in crowded areas.

Hardware Requirements:

- MAX9813 microphone amplifier
- The BeagleBone Green Board

Software Setup:

- Click the following link to open Edge Impulse and start deploying the detection model: https://www.edgeimpulse.com/
- Enter Baby Cry, Human Voice, and Room noise as the three categories on the page shown below. Provide trading examples of the three categories and set 80% for training and 20% for testing.



- Select the parameters for MFCC training shown in the topic-right picture.
- Go to the “Neural Network Settings” and enter 100 for the Number of training cycles and 0.005 for the Learning Rate.
- After the pre-setting for the detection model is completed, the windows that show the training performance will show up, and the software will generate a corresponding C++ library.
- Follow the provided link to implement the C++ library or code in C: https://isocpp.org/wiki/faq/mixing-c-and-cpp

Microphone Setup:

The guide and running C code for setting up the MAX9814 microphone is provided on the ENSC351 course website. Go to “Resources” and under “Student CMPT351 How-To Guides” to access the setup guide and download the code.

Data Collection Process:

To start detecting noise:

```
void initializebabyCryDetector() {
    pthread_create(&threadBabyCryDetector, NULL, run, NULL);
}
```

To pause the detection process:

```
void pauseBabyCryDetectorThread() {
    pthread_mutex_lock(&mutex);
    pauseFlag = 1;
    pthread_mutex_unlock(&mutex);
}
```

To resume the detection process:

```
void resumeBabyCryDetectorThread() {
    pthread_mutex_lock(&mutex);
    pauseFlag = 0;
    pthread_cond_signal(&cond);
    pthread_mutex_unlock(&mutex);
}
```

To end the detection process:

```
void stopBabyCryDetectorThread() {

    pthread_join(threadBabyCryDetector, NULL);
    pthread_mutex_destroy(&mutex);
    pthread_mutex_destroy(&babyIsCryingmutex);
    pthread_cond_destroy(&cond);
}
```

Code(audio samples / inference):

```
void* run(void* arg) {
    printf("Inside run method.\n");
    int16_t buffer[AUDIO_READ_BUFFER_SIZE];
    int fd = open("/dev/iio:device0", O_RDONLY | O_NONBLOCK);
    if (fd == -1) {
        printf("error");
        exit(1);
    }

    int count = 0;
    printf("init classifier.\n");
    while (1) {
        read(fd, buffer, AUDIO_READ_BUFFER_SIZE * sizeof(uint16_t));
        pthread_mutex_lock(&mutex);
        if (pauseFlag) {
            pthread_cond_wait(&cond, &mutex);
        }
        pthread_mutex_unlock(&mutex);

        for (int i = 0; i < AUDIO_READ_BUFFER_SIZE; i++) {
            if (count >= AUDIO_BUFFER_SIZE) {
                count = 0;
                audioClassifier();
            }
            sound[count++] = buffer[i] * 1.7;
        }
        msleep(25);
    }
    return NULL;
}
```

```c
void audioClassifier() {
    printf("Inside audioClassifier.\n");
    signal_t signal; //wapper for raw data
    static ei_impulse_result_t result; //classifier return

    signal.total_length = EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE;
    signal.get_data = &getSound;

    printf("start run_classifier_continuous.\n");
    EI_IMPULSE_ERROR res = run_classifier(&signal, &result, false);

    printf("%s: %f\n", result.classification[0].label, result.classification[0].value);
    printf("%s: %f\n", result.classification[1].label, result.classification[1].value);
    printf("%s: %f\n", result.classification[2].label, result.classification[2].value);
    float babyCryValue = result.classification[0].value;
    if (babyCryValue > 0.7) {
        printf("Baby cry detected!\n");
        pthread_mutex_lock(&babyIsCryingmutex);
        babyIsCrying = 1;
        pthread_mutex_unlock(&babyIsCryingmutex);
    }
    else {
        printf("No baby cry detected!\n");
        pthread_mutex_lock(&babyIsCryingmutex);
        babyIsCrying = 0;
        pthread_mutex_unlock(&babyIsCryingmutex);
    }
}
```

Debugging:

- If the terminal shows a constant value or value out of (0 - 4000) as the detected value, please try the following:
    - Wiring connection check
    - Double-check the start.sh script commands are executed
    - Double-check the correct GPIO pin(p9 33), Voltage(3.3V) pin, and GND pin are used
- If the program spends too much time on detecting:
    - Change volume threshold
    - Tun parameters of the machine learning model since the example provided is not the best
    - Demo or test the program in a quiet environment.
- “Resource is busy” encounter:
    - Apply single-shot mode and restart the program.
    - “Building Block” / isolated testing is recommended.

Reference

How-To Guide: Electret Microphone Amplifier(MAX9814) - Setup for Audio:
https://opencoursehub.cs.sfu.ca/mba200/grav-cms/ensc351/links/files/2022-student-howtos/MAX9814-SetupForPlayableAudio.pdf

Edge Impulse Audio Inference Guide:
https://opencoursehub.cs.sfu.ca/bfraser/grav-cms/cmpt433/links/files/2023-student-howtos/RunningMLAudioInferenceBBG_MAX9814.pdf