浙江大学

本科实验报告

咳嗽检测

课程名称：	边缘计算开发实践
姓名：	谌梓轩
学院：	信息与电子工程学院
专业：	电子科学与技术
学号：	**3210105209**
指导老师：	皇甫江涛

October 16, 2023

一、 实验目的

1. 通过 impulse 网站掌握语音数据机器学习方法
2. 配置与开发板连接的依赖
3. 通过 arduino nano 33 BLE 进行语音识别

二、 实验原理

从 20 世纪 80 年代开始，现在语音识别采用模式识别的基本框架，分为数据准备、特征提取、模型训练、测试应用这 4 个步骤，在这里我们主要来讲解下模型训练和测试应用。

模型经过训练之后，一段待测的语音需要经过信号处理和特征提取，然后利用训练好的声学模型和语言模型，分别求得声学模型和语言模型得分，然后综合这 2 个得分，进行候选的搜索，最后得出语言识别的结果。

$$\hat{W} = \underset{W \in \Omega}{argmax} p(W|X) = \underset{W \in \Omega}{argmax} P(W)P(X|W)$$

要对声音进行分析，需要对声音分帧，也就是把声音切开成一小段一小段，每小段称为一帧。分帧操作一般不是简单的切开，而是使用移动窗函数来实现。分帧后，语音就变成了很多小段。但波形在时域上几乎没有描述能力，因此必须将波形作变换。常见的一种变换方法是提取 MFCC 特征，根据人耳的生理特性，把每一帧波形变成一个多维向量，可以简单地理解为这个向量包含了这帧语音的内容信息。这个过程叫做声学特征提取。

至此，声音就成了一个 12 行（假设声学特征是 12 维）、N 列的一个矩阵，称之为观察序列，这里 N 为总帧数。

根据原理，我们在本次边缘计算中使用的代码如下所示：

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, InputLayer, Dropout, Flatten,
    Reshape, BatchNormalization, Conv2D, MaxPooling2D, AveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.constraints import MaxNorm

# model architecture
model = Sequential()
model.add(InputLayer(input_shape=(X_train.shape[1], ), name='x_input'))
model.add(Reshape((int(X_train.shape[1] / 13), 13, 1),
    input_shape=(X_train.shape[1], )))
model.add(Conv2D(10, kernel_size=5, activation='relu', padding='same',
    kernel_constraint=MaxNorm(3)))
model.add(AveragePooling2D(pool_size=2, padding='same'))
model.add(Conv2D(5, kernel_size=5, activation='relu', padding='same',
    kernel_constraint=MaxNorm(3)))
model.add(AveragePooling2D(pool_size=2, padding='same'))
model.add(Flatten())
model.add(Dense(classes, activation='softmax', name='y_pred',
    kernel_constraint=MaxNorm(3)))
```

```
17
18      # this controls the learning rate
19      opt = Adam(lr=0.005, beta_1=0.9, beta_2=0.999)
20
21      # train the neural network
22      model.compile(loss='categorical_crossentropy', optimizer=opt,
            metrics=['accuracy'])
23      model.fit(X_train, Y_train, batch_size=32, epochs=9,
            validation_data=(X_test, Y_test), verbose=2)
```

三、 实验步骤

1. 导入数据

将收集得到的咳嗽音频和噪声音频分为训练组和测试组，导入到 impulse 的数据集中，如图 1所示。

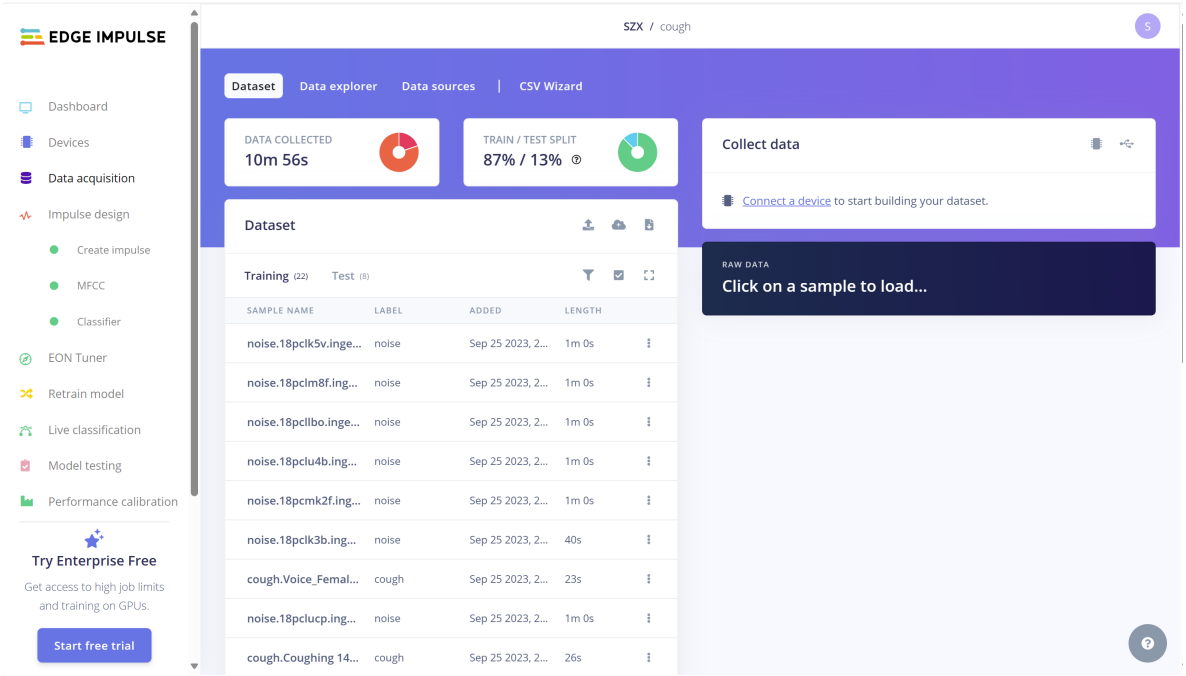

Figure 1: 数据传入与分类

2. create impule

对数据集中的数据处理方式进行选择，并定义类别，如图 2所示。

3. 数据分析

对每一个音频进行分析，转换为能够进行机器学习的数据包，得到 DSP result，如图 3所示。

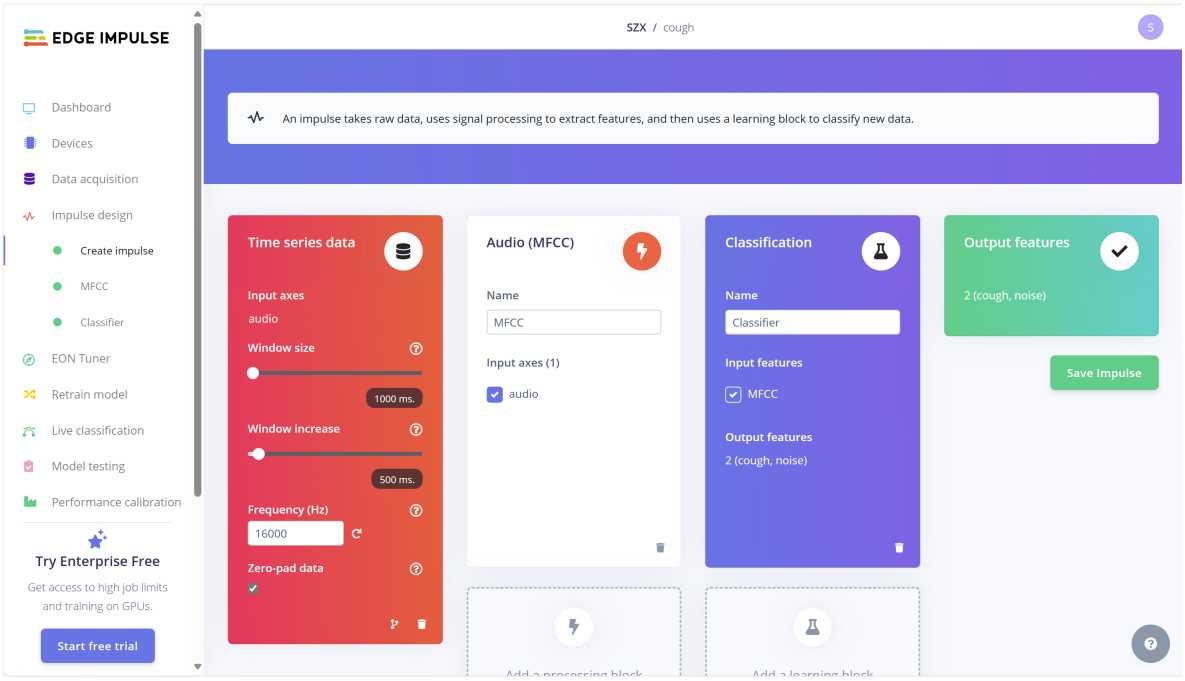

Figure 2: create impulse

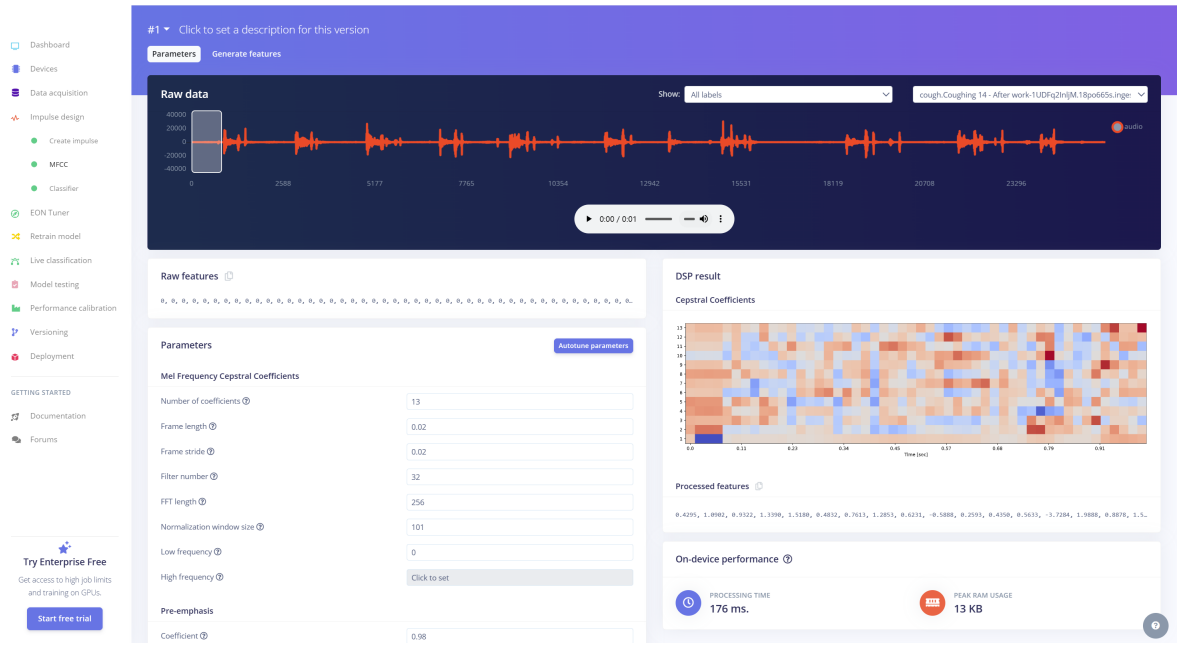

Figure 3: 数据分析

4. 机器学习结果

通过算法框架对 training 部分的音频进行学习之后，可以得到如图 4的机器学习结果, 如图 4所示。可以看到，学习得到的正确率达到 97.8%，被舍弃的部分只有 0.07，得到的结果是非常不错的。

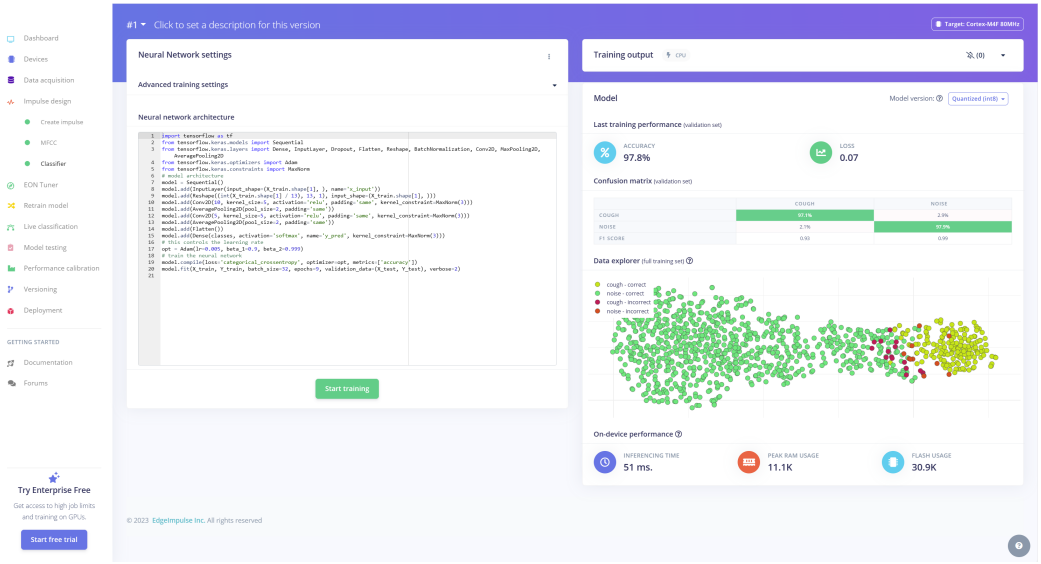

Figure 4: 学习结果

5. 测试集测试结果

根据 training 集得到的机器学习结果，可以对提供的测试集进行初步的测试，通过测试结果能够初步判定其正确率，测试结果如图 5所示。可以看出，测试集测试的结果中，正确率高达 98.09%，正确率是非常理想的，可以预见，在实际情况中，其分辨能力也是较高的。

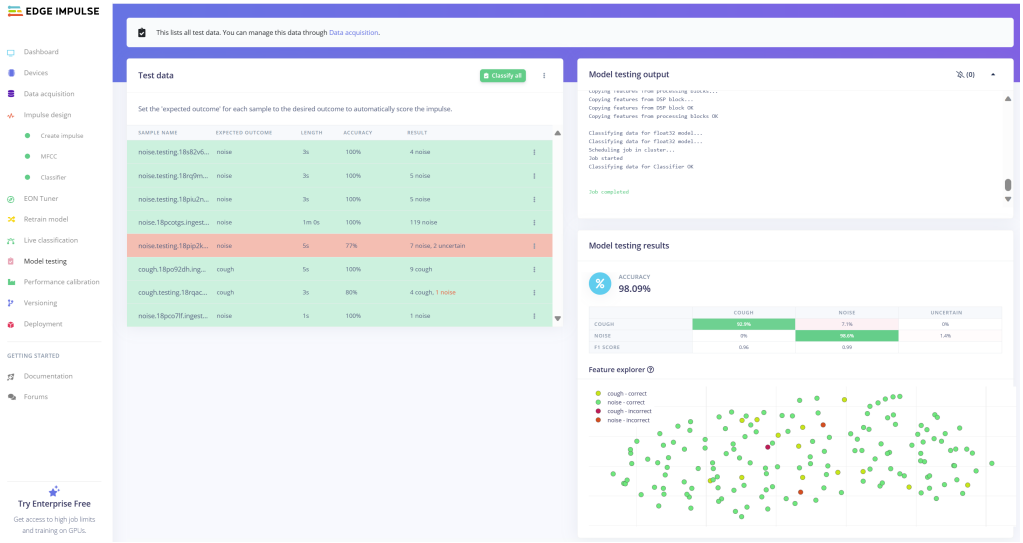

Figure 5: 测试集测试结果

四、 边缘计算结果与分析

将 impulse 机器学习中得到的结果下载下来，安装相应的依赖后，接入 arduino nano 33 BLE 开发板后，运行.bat 文件则在开发板中烧录相应的程序，并对模拟的咳嗽声进行判断，则可得到实际情况下的测试结果。测试结果如图所示。

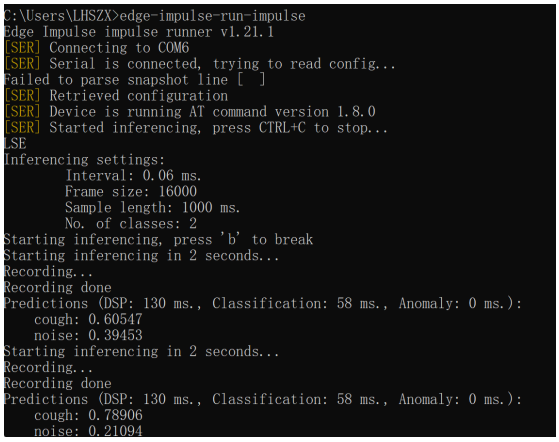

Figure 6: 测试结果 1

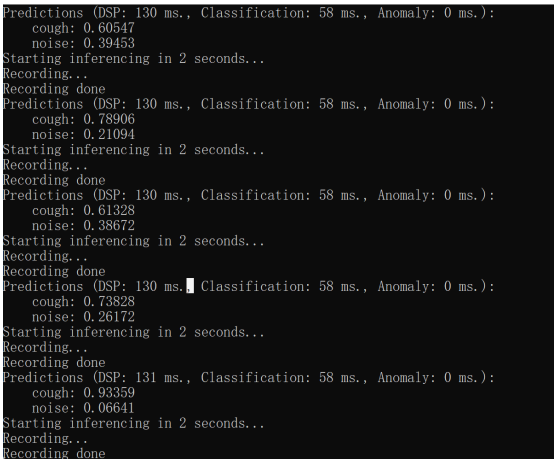

Figure 7: 测试结果 2

从测试结果中，我们可以看出，大部分的测试中，对咳嗽的检测准确度在 70% 以上，最高能够达到 90%，说明对于咳嗽的机器学习结果是较为理想的，在实际情况下也能对咳嗽进行较为准确的判断。但同时也应注意到其正确率远低于测试集的测试结果，这可能说明提供的样本不足，对于实际情况中的复杂情况无法完全覆盖。

五、 小结

在本次 cough 检测的边缘计算实验中，主要问题其实出在了环境配置和依赖的安装上，由于电脑各类语言环境的版本问题，依赖的安装总是失败，最终在更改了环境的版本后才安装成功。此外，通过这次以及其他几次的边缘计算的实验，我进一步了解了边缘计算开发的过程，明白了样本质量的重要性，能偶更好表达实际情况复杂性的样本才能提高检测的准确性；同时，一个好的算法框架也非常重要，它关乎着样本能不能被良好，高效的利用。