让行空板变身为能识别方言的智能音箱

谢作如　浙江省温州中学

胡君豪　上海人工智能实验室

摘要：在当前的中小学AI科创项目中，涉及训练语音识别模型来解决问题的比较少见，训练出AI模型并部署在开源硬件上的更是凤毛麟角。本文先借助短时傅里叶变换（STFT）的方法从语音文件中提取特征，生成二维图像，然后采用卷积神经网络（CNN）训练AI模型，最终将模型转化为ONNX部署在行空板上，实现了方言短词语的识别。

关键词：深度学习；AI模型部署；MMEdu；AI科创活动

中图分类号：G434　**文献标识码：**A　**论文编号：**1674-2117（2023）01-0093-03

问题的提出

通过查询多项AI活动的学生作品列表可以发现，在当前中小学的AI科创项目中通过训练语音识别模型来解决问题的比较少见，训练出AI模型并部署在开源硬件上的更是凤毛麟角。究其原因有二：首先是目前计算机视觉技术比较成熟，又有类似OpenMMLab、MMEdu之类的AI工具包，再加上卷积神经网络（CNN）模型在图像分类方面表现突出，中小学的教材中也内置了类似图像识别的案例。其次是全国统一使用普通话，智能音箱的应用已经遍布千家万户，探究语音识别似乎没有新意。

全国统一使用普通话和智能音箱普及固然是事实，但是这并不等于语音识别不值得探究。我国非北方地区的大部分老人，做不到像年轻人一样能够讲一口流利的普通话，他们和智能音箱的对话是困难的。况且，语音识别应用虽然随处可见，但是一些用户群体较小的方言如温州话，依然找不到相应的AI模型。因此，笔者准备设计一个支持方言识别的智能音箱，让不会说普通话的弱势群体也能享受AI的便利。

可行性分析

按照深度学习的原理和AI科创作品的开发流程，要完成这个智能音箱项目大致需要进行如下工作：

首先，需要一个本地方言的数据集，可以在当地图书馆语音库中寻找，或者自己想办法找各种有代表性的人录音形成数据集。其次，处理原始音频并提取特征，之后再搭建神经网络训练模型。在训练过程中调整各种超参数，最终得到一个性能可行的模型。最后，选择一款支持AI模型推理的开源硬件，生成相应格式的AI模型并部署。如果语音识别的效果不错，那么再增加外围的相关代码，如通过麦克风采音，识别出语音后执行预设的指令等。

从整个流程上，最关键的工作是如何处理原始音频并提取特征和搭建一个怎样的神经网络，以及如何得到能运行在开源硬件上的AI模型，这就需要介绍短时傅里叶变换（short-time Fourier transform，STFT）和ONNX（Open Neural Network Exchange）技术了。

语音的原始音频信号是一维的，如果使用原始信号作为输入数据，信号长度较长，同时，使用该输入对应的网络模型也会很大，更糟糕的是找不到可以参考的直接用一维信号进行语音识别的网络模型。STFT可以将一维信号变为

二维信号，该二维信号叫做时频谱图，横坐标为时间，纵坐标为频率，颜色深度为对应时间和对应频率的大小。如图1所示，变换后的信号就等同于一张图像，那么语音分类问题就等同于图像分类问题了。MMEdu中内置了很多图像分类的轻量型卷积神经网络，如ResNet、MobileNet等，借鉴或者直接使用就能训练模型了。

在本栏目上一期的文章中，笔者已经完成了将ONNX模型部署在行空板上。ONNX是一种通用的AI模型，支持多平台，推理环境搭建非常方便，是部署AI应用的主流选择。MMEdu支持直接导出ONNX模型，新版的行空板也内置了ONNX的推理环境。

● 从数据采集、网络搭建到模型训练

1.语音数据集的采集和特征提取

通过图书馆、温州当地的大数据开放平台，都没有找到温州话的语音库，只好采用最笨也是最踏实的办法——手动采集。笔者找会温州话的学生录制了近300条音频文件，再通过SpecAugment（自动语音识别数据扩充）的方法进行数据集增强，得到1520个训练数据和520个测试数据。

```
x,y = [], []
y.append(file.split('\\')[-2])
wave,fs = librosa.load(file, sr = 16000)
# time 对应的样点数
sample = int(time * fs)
# 若音频短于 time 的样点数，则在音频后面补零，若音频长于 time 的样点数，则对音频
截断
if wave.size <= sample:
        wave = np.concatenate((wave,np.array((sample - wave.size) * [0])))
else:
        wave = wave[0:sample]
spc = librosa.feature.melspectrogram(wave, sr=fs, n_fft=512)
spc = librosa.power_to_db(spc, ref=np.max)
spec_new = (((spc+80)/80)*255).astype(np.uint8)
h,w = spec_new.shape
rgb_matrix = np.array([color_map[i] for i in spec_new.flatten()]).reshape(h, w,
3)
spec_rgb = rgb_matrix/255
# 添加到数据
x.append(spec_rgb)
```

图2

有了音频文件后，再通过librosa、numpy和Pillow库以STFT的方法，将这些音频文件转换为一张张图片，然后按照ImageNet的格式做成数据集。核心代码可参考图2所示的代码。

2.神经网络的搭建

目前，MMEdu已经在图像分类方面内置了5种SOTA（state-of-the-art，指最先进的、最新技术水平的）模型。经过再三比较，笔者选择Resnet18作为最终模型，原因如下：

高精度。ResNet18以其在图像分类任务上的高精度而著称，对于时频谱图的分类，ResNet18的表现超过了其他模型。

模型大小合适。与其他卷积神经网络CNN相比，ResNet18模型适中，推理速度较快，这使得它的训练和部署效率更高。

可迁移学习。由于ResNet18已经在大型图像数据集上进行了训练，因此可以将其用作迁移学习的起点。如果想提高其在特定任务上的性能，仅仅需要在自己的数据集上对其进行微调。

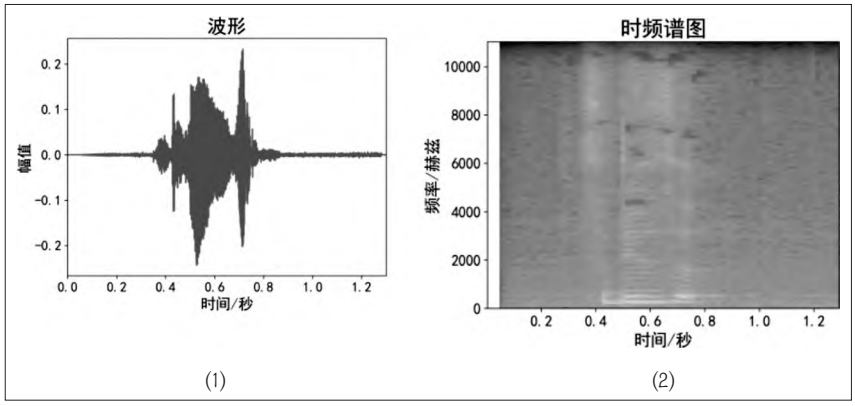

图1 波形信号转化为时频谱图

3.网络模型的训练

完成一个AI模型的训练需要经历多个步骤，收集并整理好数据集是基础，接下来的工作是搭建ResNet18模型，再使用官方的预训练权重文件作为迁移学习的“范本”。至于代码倒很简单，MMEdu的

模型训练代码是公式化的，仅需几行即可完成对预训练ResNet18网络的迁移学习。参考代码如图3所示。

需要说明的是，执行上述的代码需要下载ResNet18模型的预训练权重文件。该文件MMEdu一键安装包中已经内置。另外，因为是基于预训练模型的迁移学习，只要5轮左右就能得到不错的识别效果，使用OpenInnoLab的GPU容器，训练一轮大概需要30多秒，也就是说数分钟内即可训练好这个模型。

在完成训练后，借助MMEdu的“convert”方法就能导出ONNX格式模型，代码（仅仅一行）如下：

```
model.convert(checkpoint = "ResNet18.pth",out_file="ResNet18.onnx")
```

```
Python
# 导入 MMEdu
from MMEdu import MMClassification as cls
# 模型实例化
model = cls(backbone='ResNet18')
# 指定输出类别数量，假设是 6 类
model.num_classes = 6
# 指定数据集的路径 path ='...'改为自己数据集的路径
model.load_dataset(path='...')
# 指定保存模型配置文件和权重文件的路径 ='...' 改为自己想保存的路径
model.save_fold = '...'
# 模型训练，训练完自动保存，checkpoint 后面接上 resnet18-5c106cde.pth 的路径
model.train(epochs=5, validate=True, checkpoint='...')
```

图3

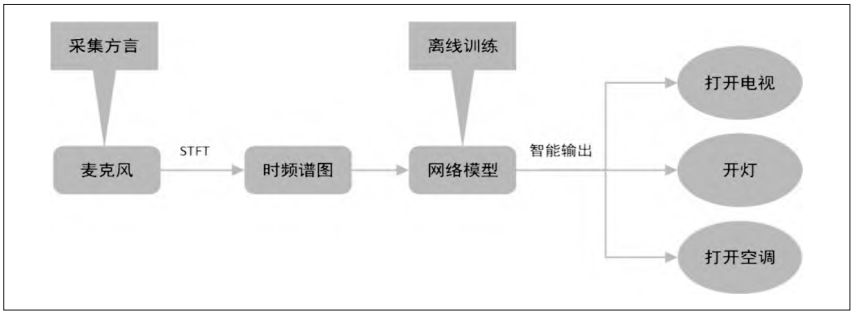

图4 智能方言小助手工作流程

在行空板上部署AI模型

行空板自带麦克风和触控屏，只要加一个小音箱就可以做智能音箱的项目，当然也可以用行空板的蓝牙功能接任意一个蓝牙音箱。笔者希望这个智能音箱不用联网也能识别语音，也就是说从音频采集到模型推理的一系列工作都是在行空板上完成的。因为身边的行空板还不是最新固件，需要安装一些额外的Python库，如librosa、pyaudio和onnxruntime等。前两个针对音频信号处理，最后一个用于运行ONNX模型。

笔者最终的设计是用行空板完成一个用温州话短语控制智能家居的助手项目，其工作流程图如图4所示。

当然，这个项目还需要增加“录音”“界面设计”之类的代码。最终代码在行空板上的运行结果如图5所示。

图5 实物运行

总结

鉴于在中小学很少看到语音识别方面的AI科创研究，笔者设计了这个智能音箱的项目。本项目的最大启示在于，语音分类的问题通过特定的特征处理后，也能够转换为图像分类问题，只要拥有相应的语音数据，就能通过卷积神经网络解决各种模式识别方面的问题。还有AI模型训练和硬件是没有直接关联的，如这个项目训练的模型，除了行空板外，还可以直接部署在冲锋舟、虚谷号、树莓派和香橙派等硬件上。

本项目涉及的相关文件都已经放在OpenInnoLab（浦源-浦育）平台上，通过以下网址可以测试全部代码。

https://www.openinnolab.org.cn/pjlab/project?id=63a02aaf3791ab1c3aa9814a