2019-01-25

OPEN AI LAB

命令词识别教学案例

变更记录  
(Reversion Record)

|  |  |  |  |
| --- | --- | --- | --- |
| 日期  (Date) | 版本  (Rev) | 说明  (Change Description) | 作者  (Author) |
| 2019-01-25 | V0.1 | 初稿 | 张哲 |
| 2019-01-30 | V0.2 | 增加部署操作方法，部署附件 | 张哲 |
| 2019-02-12 | V0.3 | 删除工具安装方法及部署部分内容，合并教学文档与算法实现文档 | 张哲 |
|  |  |  |  |
|  |  |  |  |

目录(**catalog**)

[1 语音识别介绍 3](#_Toc879672)

[1.1 语音识别概念 3](#_Toc879673)

[1.2 命令词识别概念 4](#_Toc879674)

[2 命令词识别方法 6](#_Toc879675)

[2.1 前处理模块 6](#_Toc879676)

[2.1.1回声消除（AEC） 6](#_Toc879677)

[2.1.2语音活性检测（VAD） 7](#_Toc879678)

[2.1.3噪声抑制（NS） 8](#_Toc879679)

[2.1.4语音去混响（Dereverberation） 10](#_Toc879680)

[2.2 语音识别模型 11](#_Toc879681)

[2.2.1KWS System介绍 11](#_Toc879682)

[2.2.2特征提取器 12](#_Toc879683)

[2.2.3神经网络分类器 14](#_Toc879684)

[2.3 模型量化 20](#_Toc879685)

[3 算法实现流程 21](#_Toc879686)

[3.1 运行环境与使用工具 21](#_Toc879687)

[3.2 下载项目资源 21](#_Toc879688)

[3.3 训练模型 21](#_Toc879689)

[3.4 测试 24](#_Toc879690)

[3.5 模型转pb文件 24](#_Toc879691)

[3.6 检验模型识别效果 25](#_Toc879692)

[4 模型部署 28](#_Toc879693)

[4.1 g++编译 28](#_Toc879694)

[4.2 音频采集及实时识别 28](#_Toc879695)

[附件一 31](#_Toc879696)

[1.1 freeze.py脚本 31](#_Toc879697)

[1.2 部署demo 31](#_Toc879698)

# 语音识别介绍

## 1.1 语音识别概念

语音识别技术，简单来说其实就是利用计算机将语音信号自动转换为文本的一项技术，如下图所示。这项技术同时也是机器理解人类言语的第一个也是很重要的一个过程。与机器进行语音交流，让机器明白你说什么，这是人们长期以来梦寐以求的事情。语音识别技术就是让机器通过识别和理解过程把语音信号转变为相应的文本或命令的高技术。语音识别是一门交叉学科，需要具备生理学、声学、信号处理、计算机科学、模式识别、语言学等相关学科的知识。语音识别是十年来发展最快的技术之一，随着AI的不断发展，深度学习让语音识别技术得到了质的飞跃，开始从实验室走向市场，并逐步走到人们的生活中。人们预计，未来10年内，语音识别技术将进入工业、家电、通信、汽车电子、医疗、家庭服务、[消费电子产品](https://baike.baidu.com/item/%E6%B6%88%E8%B4%B9%E7%94%B5%E5%AD%90%E4%BA%A7%E5%93%81" \t "_blank)等各个领域。随着移动互联网技术的不断发展，尤其是移动端和嵌入端的小型化、多样化变化趋势，语音识别成为区别于键盘、触屏的人机交互手段之一。随着语音识别算法模型、自适应性的加强，相信在未来很长一段时间内，语音识别系统的应用将更加广泛与深入，更多丰富的嵌入式终端语音识别产品将步入人们的日常生活。

其他

应用

语音识别



文本信息

图1-1 语音识别

根据识别的对象不同，语音识别任务大体可分为3类，即孤立词识别（isolated word recognition），连续语音识别和关键词识别（或称关键词检出，keyword spotting）。

孤立词识别的任务是识别事先已知的孤立的词，如“开机”、“关机”等；连续语音识别的任务则是识别任意的连续语音，如一个句子或一段话；关键词识别针对的是连续语音，但它并不识别全部文字，而只是检测已知的若干关键词在何处出现，如在一段话中检测“自然”、“价值观”这两个词。关键词识别主要应用技术有命令词识别、语音监听、语音拨号、对话系统、话题跟踪、数据查询、文档分类等。本文所介绍的主要对象就是命令词识别技术。

一般来说,语音识别的方法有三种：基于声道模型和语音知识的方法、模板匹配的方法以及利用人工神经网络的方法。

基于语音学和声学的方法该方法起步较早，在语音识别技术提出的开始，就有了这方面的研究，但由于其模型及语音知识过于复杂，现阶段没有达到实用的阶段。

模式匹配的方法发展比较成熟，目前已达到了实用阶段。在训练阶段，用户将词汇表中的每一词依次说一遍，并且将其特征矢量作为模板存入模板库。在识别阶段，将输入语音的特征矢量依次与模板库中的每个模板进行相似度比较，将相似度最高者作为识别结果输出。

利用人工神经网络的方法是80年代末期提出的一种新的语音识别方法。人工神经网络本质上是一个自适应非线性动力学系统，模拟了人类神经活动的原理，具有自适应性、并行性、鲁棒性、容错性和学习特性，其强分类能力和输入输出映射能力在语音识别中都很有吸引力。在命令词识别任务中也是选用了人工神经网络。

基于神经网络的语音识别系统的结构主要包括语音信号的采样、前处理、特征提取、语音识别模型以及语音识别后处理部分，下图中给出了基于神经网络的语音识别系统的基本结构。

输入语音

输入特征

语音信号

基本识别结果

语音设备

特征提取

前处理

部署

训练

语音识别模型

图1-2 语音识别系统基本结构

## 1.2 命令词识别概念

语音是人类极为重要的信息传递方式，是社会交往中最为自然便捷的交互手段，通过语音实现更好的人机交互是众多科技工作者为之努力的目标。作为语音相关研究的一部分，语音关键词识别 (Keyword Spotting, KWS) 指从无约束的自然语音流中检测出事先约定的若干关键词并予以反馈的技术。该技术具有广泛的应用前景，它是智能人机对话、语控智能家居、电话安全监控等系统的重要组成部分，该技术的研究与应用不断拉近着科技与生活间的距离。

命令词识别是关键词识别领域的一个子领域，其目的是在语音信号中检测出指定词语，再执行后续功能。作为当前研究的热点，命令词识别任务是将捕捉到的人类话音对其中命令进行响应。用户可以通过命令词检测系统控制智能家电的开启、通过语音菜单控制计算机程序的动作等等。

当下智能设备已经进入千家万户，移动应用服务遍及生活各个角落。同时目前的很多语音助手还缺少可靠的语音唤醒功能。本文介绍的命令词识别方法适用于移动端和嵌入端，改变了语音交互系统的手动唤醒过程，真正解放用户的双手、提升人机交互体验。

命令词识别任务中常把命令词用作唤醒和打断的功能。唤醒就是设备从休眠到唤醒的过程。首先，设备需要被开启，自动加载好资源，这时它处于休眠状态。当用户说出特定的命令词时，设备就会被唤醒，切换到工作状态等待用户接下来的指令。这一过程中用户不需要用手接触，直接可以用语音进行操作。打断也是利用语音唤醒的机制，当用户说出特定的命令词时，设备就会从工作状态被打断，让设备不用实时地处于工作状态，从而节省能耗。如设备在播放音乐时，用户说出命令词，设备就会停止播放音乐。

命令词识别的应用领域比较广泛，例如机器人、手机、可穿戴设备、智能家居、车载等。几乎很多带有语音功能的设备都会需要命令词识别技术作为人和机器互动的一个开始或入口。不同的产品会有不同的命令词，当用户需要唤醒设备时需要说出特定的命令词。大部分的命令词都是三到四个音节，音节覆盖多，音节差异大，相对唤醒效果会比较稳定。

命令词识别可以看成是一种小资源的关键词检索任务，其中小资源是指计算资源比较小和空间存储资源比较小，因此它的系统框架跟关键词检索的系统会有一定的区别，目前常用的系统框架主要有Keyword/Filler Hidden Markov Model System和KWS System两种。本文介绍的命令词识别技术采用的就是第二种系统。

# 命令词识别方法

命令词识别可以分成前处理模块和语音识别模型两部分。前处理模块可以对机器获取的语音进行“提纯”（语音增强），来提升模型识别准确率。前处理模块主要包含自动回声消除（AEC）、语音活性检测（VAD）、噪声抑制（NS）和自动增益控制模块（AGC）。前处理之后的语音信号输入给语音识别模型，经过加窗和梅尔频率倒谱系数（MFCC）等操作之后从时域转化为频率-时间信号，之后使用神经网络进行识别，经过加权平均后输出各个命令词的置信度。

## 前处理模块

机器获取的语音会受到四种类型的噪声干扰，分别是加性噪声、其它语音干扰、混响和回声，它们会消弱语音的质量，进而会影响识别率或者人的主观听感。处理上述四种不利的影响有很多方法，主要集中在传统信号处理邻域，包括时域、频率以及空域（可以获得声源的空间方位信息），以及当前的深度学习邻域。传统的方法都是先对处理过程建模，如VAD使用的高斯混合模型，这些建模是现实物理过程近似的模拟；深度学习的方法用大量标注过的数据训练的方法以获得更为精确的物理过程近似。经过增强后的语音会大大提高模型的识别率。

前处理模块的效果对模型识别率的影响很大，好的前处理模块可以大大提升语音的信噪比，提高模型的识别率，但是如果前处理模块效果很差，可能导致模型完全无法正确识别。这里重点介绍自动回声消除（AEC）、语音活性检测（VAD）、噪声抑制（NS）这3个模块。

## 2.1.1回声消除（AEC）

回声消除是为了消除机器自身发出的声音，不影响外界传递过去的声音，如在通话场景中，来自远端的经过扬声器放出来的声音被消掉，否则经过麦克风采集和近端说话人信号混在一起被传递到远端，则会出现远端的人听到自己说的话，这种体验不好。在远场音箱交互场景中，这自动回声消除的难度比通话场景要复杂一下，主要的原因是近端说话的语音是淹没在大功率喇叭且存在低音单元之中的。

回声消除的基本原理是用一个自适应滤波器对未知的回声信道进行参数辨识，根据扬声器信号与其产生回声信号相关性为基础，建立远端信号模型，模拟回声路径，通过自适应算法调整，使其冲击响应和真实回声路径相逼近。然后将麦克风接收到的信号减去估计值，即可实现回声消除功能。回声消除的基本结构如下图所示。

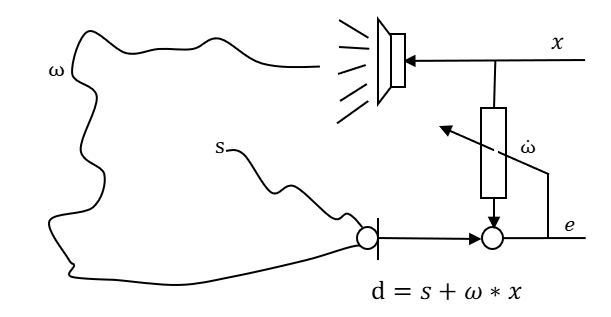


图2-1 回声消除结构

回声消除的相关公式如下：

式中ω是回声通道的时域冲击响应函数，x是远端语音；echo是所得回声；s是近端说话人语音和环境噪声，d为麦克风采集到的信号；ω’是估计的时域冲击相应函数，回声通路的y’是对回声信号的估计值，s’为近端说话人语音和环境噪声的估计。

## 2.1.2语音活性检测（VAD）

VAD(voice activity detection)广泛应用于语音编码和降噪。这里所说的是语音/非语音检测，一个VAD系统通常包括两个部分：特征提取和语音/非语音判决。

常用的特征提取可以分为五类：基于能量、频域、倒谱、谐波和长时信息。基于能量的特征常用硬件实现；倒谱在在低信噪比下可以获得较好的效果；当信噪比达到0dB时，基于语音谐波和长时语音特征更具有鲁棒性。

* 基于能量的特征

基于能量的准则是检测信号的强度，并且假设语音能量大于背景噪声能量，这样当能量大于某一门限时，可以认为有语音存在。然而当噪声大到和语音一样时，能量这个特征无法区分语音还是纯噪声。 早先基于能量的方法，将宽带语音分成各个子带，在子带上求能量；因为语音在2KHz以下频带包含大量的能量，而噪声在2~4KHz或者4KHz以上频带比0~2HKz频带倾向有更高的能量。在信噪比低于10dB时，语音和噪声的区分能力会加速下降。

* 频域特征

通过短时傅里叶变换将时域信号转为频域信号，即使在信噪比到0dB时，一些频带的长时包络还是可以区分语音和噪声的。

* 倒谱特征

对于VAD，能量倒谱峰值确定了语音信号的基频，也有使用MFCC做为特征的。

* 基于谐波的特征

语音的一个明显特征是包含了基频及其多个谐波频率，即使在强噪声场景，谐波这一特征也是存在的。可以使用自相关的方法找到基频。

* 长时特征

语音是非稳态信号。普通语速通常每秒发出10~15个音素，音素下的谱分布是不一样的，这就导致了随着时间变化语音统计特性也是变化的。另一方面，日常的绝大多数噪声是稳态的（变化比较慢的），如白噪声/机器噪声。

常用的判决准则可以分为三类：基于门限、统计模型和机器学习。

* 门限

门限通常是根据训练数据集特征预先得到的，对于噪声变化的场景，需要使用自适应的门限。自适应门限算法能够根据信噪比的差异动态调整门限阈值，更加实时有效地提升检测精度，在低信噪比地环境下任能有效可靠地感知信号。

* 统计模型方法

统计模型地方法最先源于似然比检验（likelihood ratio test LRT），这种方法假设语音和背景噪声是独立高斯分布，这样它们地离散傅里叶系数（DFT）可以用高斯随机变量来描述。基于高斯地统计模型在低信噪比下可以获得更好地VAD检测结果。

* 机器学习

这一思想是语言模型是通过学习得到的，目前训练数据集都是通过传统的VAD方法进行标注的，如果标注的数据里有不准确的，那么如何通过神经网络获得更准确的VAD，这类方法的难点更多的是实际强噪声场景下的训练数据集标注的问题。

## 2.1.3噪声抑制（NS）

噪音抑制技术原理是将实时采样的数字信号进行频谱分析，这样我们就能分析背景噪声响应的强度和频谱分布，然后根据这个模型就能设计一个滤波器，当有人讲话的时候，同时做信号分析，根据分析就能得出讲话者的频谱，那么根据这些背景噪音和讲话者的频谱，这个滤波器根据两个信号的对比实时的改变，让讲话者声音频谱通过，对背景噪声的频谱进行抑制，降低其能量，比如说降低15到20个分贝，就很明显可以感觉体验到噪音抑制的效果。

噪声频谱可以使用如语音/噪声似然函数进行估计，将接收到的每帧信号和频率分量分为噪声或语音。带噪语音模型的表达式如下：

上式中s和v分别表示干净语音和噪声，y表示麦克风采集到的带噪语音。

它们的频谱关系如下式，可以看出语音和噪声是加性且不相关的，对于非加性关系可以用AEC等算法对不同场景进行抑制。根据中心极限定义，一般认为噪声和语音分布服从均值为0，方差为μ的正太分布。

所以这里的中心思想就变成了从Y中估计噪声N，然后抑制N得到语音，即：

所以对噪声估计的准确性是至关重要的，噪声估计的越准确得到的结果就越好。估计噪声的方法有如下几种：

* 基于VAD检测的噪声估计，VAD对Y进行检测，如果检测没有语音，则认为噪声。
* 基于全局幅度最小原理，该估计认为幅度谱最小的情况必然对应没有语音的时候。
* 基于矩阵奇异值分解原理估计噪声的。
* 对似然比函数进行改进，将多个语音/噪声分类特征合并到一个模型中形成一个多特征综合概率密度函数，对输入的每帧频谱进行分析。其可以有效控制风扇/办公室设备等噪声。

对接收到的每一帧带噪语音信号，以对该帧的初始噪声估计为前提，定义语音概率函数，测量每一帧带噪信号的分类特征，使用测量出来的分类特征，计算每一帧基于多特征的语音概率，在对计算出的语音概率进行动态因子（信号分类特征和阈值参数）加权，根据计算出的每帧基于特征的语音概率，修改多帧中每一帧的语音概率函数，以及使用修改后每帧语音概率函数，更新每帧中的初始噪声（连续多帧中每一帧的分位数噪声）估计。

这里噪声抑制算法的核心是采用维纳滤波器来抑制估计出来的噪声。维纳滤波（wiener filtering) 一种基于最小均方误差准则、对平稳过程的最优估计器。这种滤波器的输出与期望输出之间的均方误差为最小，因此，它是一个最佳滤波系统。它可用于提取被平稳噪声所污染的语音信号。

[维纳滤波器](https://baike.baidu.com/item/%E7%BB%B4%E7%BA%B3%E6%BB%A4%E6%B3%A2%E5%99%A8)的输入为含噪声的带噪语音信号。期望输出与实际输出之间的差值为[误差](https://baike.baidu.com/item/%E8%AF%AF%E5%B7%AE)，对该误差求均方，即为[均方误差](https://baike.baidu.com/item/%E5%9D%87%E6%96%B9%E8%AF%AF%E5%B7%AE)。因此[均方误差](https://baike.baidu.com/item/%E5%9D%87%E6%96%B9%E8%AF%AF%E5%B7%AE)越小，噪声滤除效果就越好。为使[均方误差](https://baike.baidu.com/item/%E5%9D%87%E6%96%B9%E8%AF%AF%E5%B7%AE)最小，关键在于求冲激响应。如果能够满足[维纳](https://baike.baidu.com/item/%E7%BB%B4%E7%BA%B3)－霍夫方程，就可使[维纳滤波器](https://baike.baidu.com/item/%E7%BB%B4%E7%BA%B3%E6%BB%A4%E6%B3%A2%E5%99%A8)达到最佳。根据维纳-霍夫方程即可以求解维纳滤波器的冲激响应：

式中， 为语音信号的功率谱密度，为噪声信号的功率谱密度，为维纳滤波系统的增益函数。此时信号s的频谱估计值为：

通过维纳滤波的方法对语音段中噪声出现的概率进行估计，然后更新噪声的功率谱，就可以提取出消除噪声之后的语音信号。

## 2.1.4语音去混响（Dereverberation）

当声源发声时，由于反射和延迟现象，会出现同一个声音多次到达人耳的情形，人耳能明显区别出的是回声(这时反射和原始声音到达时间差可长达50ms,甚至数百上千毫秒)。如果反射的声音和原始声音时间在10ms~30ms之间，由于人耳的时间掩蔽特性会增强人耳的听感，但是对于语音识别是有影响的。混响RT60 是标准的混响时间测量方法，表示从测试信号突然停止到声压级降低 60 dB 所用的时间。 当前去混响算法主要分为三个类别:

* 波束形成

波束形成时空域滤波方法，来自其它方法地混响由于空域上地选择性会被滤除，这就要求目标方向时正确的，在360度的3D场景中，需要先定位出目标方向，而在定位目标方向时，混响是有影响的。

* 语音增强技术方法

当前去混响方法主要分为三个类别：基于统计模型的去混响方法、基于线性预测编码LPC(linear predictive coding)和基于特征值分解法。

* 盲反卷积

声学脉冲响应的未知情况下，根据观察到的信号，设计出反向滤波器来削弱混响的影响。

## 语音识别模型

## 2.2.1KWS System介绍

前面提到命令词识别采用的系统是KWS System，这也是语音识别系统中的核心部分。KWS主要是应用在智能语音设备上实现基于语音的用户交互的关键系统，它要求实时响应和高精度，以获得良好的用户体验。

与传统的语音算法相比，神经网络具有较高的精确度，因此成为KWS系统结构的一个有吸引力的选择。在KWS的论文中训练了各种神经网络架构，并比较了它们的准确性和内存/计算需求。还进一步阐述了神经网络(DS-CNN)，并将其与其他神经网络体系结构进行了比较。DS-CNN的准确率达到95.4%，比参数数量相近的DNN模型高出约10%。

在一些应用程序中，KWS可以用作智能设备（如支持语音的灯泡）的语音命令。由于KWS系统始终处于开机状态，所以它应该具有非常低的功耗来最大化电池寿命。另一方面，KWS系统应该以较高的准确率和较低的延迟检测命令词，以获得最佳的用户体验。这些相互冲突的系统需求是的KWS从50年前就成为一个活跃的研究领域。KWS体系结构的设计必须考虑这些约束条件，在不牺牲精度的前提下，优化内存和计算限制的微控制器架构。

微控制器是一种低成本、节能的处理器，在我们的日常生活中无处不在，它们存在于各种设备中，从家用电器、汽车、消费电子产品到可穿戴设备等等。然而，在微控制器上部署基于KWS的神经网络带来了以下挑战：

* 内存占用

典型的微控制器系统只有几十到几百KB的可用内存。整个神经网络模型，包括输入/输出、权重和激活，都必须符合这个小内存预算。

* 有算的计算资源

由于KWS总是在线的，实时需求限制了每个神经网络推理的操作总数。

这些受微控制器资源约束以及KWS高精确度和低延迟的要求，使得在对资源约束的神经网络体系结构中寻找适合KWS的神经网络结构成为了一个重要的探索方向。

论文中提到在谷歌语音命令数据集上训练了KWS神经网络模型，并在准确性、内存占用和每次推理的操作次数方面进行了比较。此外，还实现了一个新的KWS模型，使用了深度可分离卷积和点可分卷积(DS-CNN)，该模型在精度、模型大小和操作次数等方面都优于以往的其他模型。最后，进行了资源约束神经网络体系结构的探索，并在一组典型微控制器的计算和内存约束中对不同的网络体系结构进行了综合比较。

KWS系统就是由端到端的深度神经网络关键词检测模型进行命令词的识别，即输入是语音，输出直接是命令词。典型的KWS系统由特征提取器和基于神经网络的分类器组成，如下图所示。输入数据为PCM格式的音频数据，经过MFCC特征提取后转化为频率-时间频谱图，之后送入神经网络进行识别，最后输出各命令词的置信度结果。

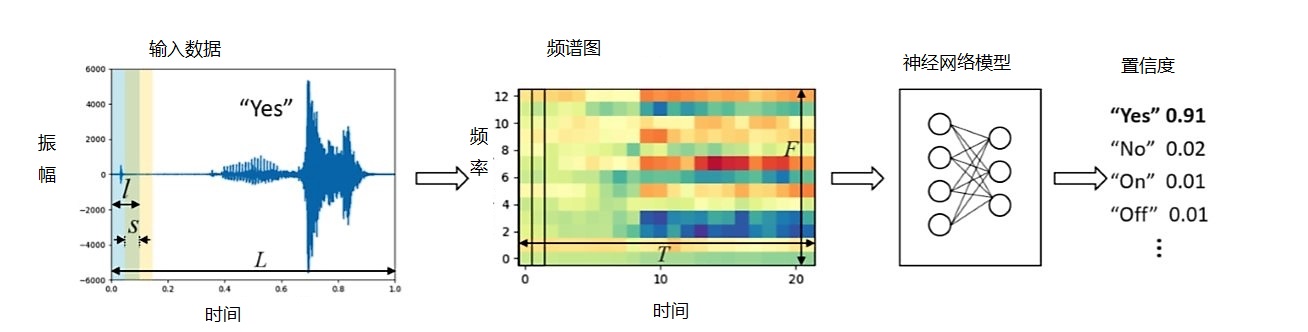


图2-2 KWS系统

该语音识别模型主要特点有：

* 模型结构简单，摒弃了传统语音识别中声学模型-词典-语言模型的复杂结构，使用单个模型即可进行多个关键词的检测。
* 训练方法简单，使用分类模型进行关键词检测，只需要拥有一定的深度学习知识即可完成训练，对声学知识储备的要求较低。
* 计算和内存开销较低，由于CNN的参数共享特性，模型大小相比于传统的全连接网络（FC）模型大幅减小，利用量化技术进一步减小内存和计算开销。
* 识别准确度高，抗噪能力强，相比于FC模型，相同大小的CNN模型能够达到更高的识别率。

## 2.2.2特征提取器

梅尔频率倒谱系数Mel-frequency cepstral coefficients(MFCC) 是基于深度学习的语音识别中常用的人工语音特征，是对传统语音处理技术的改进。使用MFCC进行特征提取涉及到将（time-domain）时域语音信号转换为一组(frequency-domain)频域谱系数，从而实现输入信号的维数压缩。

MFCC特征向量提取的一般计算流程如下图所示，包括预处理、快速傅里叶变换（FFT）、梅尔滤波、对数运算、离散余弦变换（DCT）这几个步骤。

输入语音

DCT(Discrete cosine transform)

取对数

Mel滤波

取绝对值或平方值

FFT(Fast Fourier Transform)

预加重、分帧、加窗

输出特征向量

图2-3 MFCC特征提取计算流程

预处理主要包括预加重、分帧和加窗。

欲加重技术可以增强高频能量，主要用在回声消除以及语音识别中的特征提取中。声道的终端是口和唇，口唇辐射对低频影响比较小，但是对高频段影响比较大，因此可通过欲加重技术来提升语音的高频分辨率。

傅里叶变换要求输入的信号是平稳的。语音信号在宏观上是不平稳的，在微观上是平稳的，具有短时平稳性（10-30ms内可以认为语音信号近似不变），这就可以应用分帧技术，把语音信号分为一些短段来进行处理，每一个短段称为一帧。语音信号的分帧是采用可移动的有限长度的窗口进行加权的方法来实现的。一般每秒的帧数约为33~100帧，视情况而定。一般分帧需要相互重叠一部分，相邻两帧的起始位置的时间差叫做帧移（stride），帧移与帧长的比值一般为0~0.5。

把输入语音做分帧处理后，可计算出输入长度为L的语音中可以分多少帧数。首先将长度L的输入语音信号以步幅s的形式框入每帧长度为 的帧中，得到分帧的帧数T为：

从每一帧中提取出F个语音特征，对长度为L的整个输入语音信号生成一组TxF个特征。

加窗即与一个窗函数相乘，窗函数为最简单的矩形窗。加窗之后是为了进行傅里叶展开。加窗的代价是一帧信号的两端部分被消弱了，所以在分帧的时候，帧与帧之间需要有重叠。加窗可以使全局更加连续，避免出现吉布斯效应（将具有不连续点的周期函数（如[矩形脉冲](https://baike.baidu.com/item/%E7%9F%A9%E5%BD%A2%E8%84%89%E5%86%B2/10697028" \t "_blank))进行[傅立叶](https://baike.baidu.com/item/%E5%82%85%E7%AB%8B%E5%8F%B6" \t "_blank)级数展开后，选取有限项进行合成。当选取的[项数](https://baike.baidu.com/item/%E9%A1%B9%E6%95%B0)越多，在所合成的波形中出现的峰起越靠近原信号的不连续点。当选取的项数很大时，该峰起值趋于一个常数，大约等于总跳变值的9%。这种现象称为[吉布斯](https://baike.baidu.com/item/%E5%90%89%E5%B8%83%E6%96%AF" \t "_blank)效应）；使原本没有周期性的语音信号呈现出周期函数的部分特征。

对每一帧信号进行快速傅里叶变换，从时域变为频域：

根据每一帧FFT后的数据计算谱线的能量：

把求出的每帧谱线能量通过梅尔滤波器，并计算在该梅尔滤波器中的能量，在频域中相当于把每帧的能量谱E(i,k)与梅尔滤波器的频域相应传递函数相乘并相加：

式中，i表示第i帧，k表示频域中的第k条谱线。

最终MFCC的计算公式如下：

式中，S(i,m)是梅尔滤波器中的能量，m是指第m个梅尔滤波器，i是指第i帧，n是DCT后的谱线。

将MFCC提取的语音特征矩阵输入到分类器模块，进行识别，生成输出类的概率。

## 2.2.3神经网络分类器

传统的KWS语音识别技术使用隐马尔可夫模型(Hidden Markov model, HMMs)和Viterbi解码。虽然这些方法实现了合理的精确性，但是它们很难训练，而且在推理过程中计算开销很大。为KWS探索的其他技术包括采用大边界问题公式或递归神经网络(RNN)的判别模型。虽然这些方法在准确率上明显优于基于HMM的KWS，但是它们存在较大的检测延迟。KWS论文中指出在引入了基于全连接层的深度神经网络(DNN)模型和校正线性单元(ReLU)激活函数之后，其性能优于HMM模型，检测延迟非常小。此外，采用低秩近似技术对DNN模型的权值进行压缩，在较少的硬件资源下达到相似的精度。神经网络的主要缺点是在输入语音特征中忽略了局部时间和频谱相关性。为了利用这些相关性，论文中又研究了基于卷积神经网络(CNN)的KWS的不同变体，其准确率高于DNN。CNN建模时变信号(如语音)的缺点是忽略了长期的时间依赖性。结合卷积神经网络和RNN的优点，又研究了基于卷积回归神经网络（CRNN）的KWS，证明了该模型对噪声的鲁棒性。在对已有的KWS神经网络进行交叉熵损失函数训练的基础上，提出了一种基于最大池化损失函数训练长短时记忆网络(LSTM)的KWS模型，其精度优于交叉熵损失训练的DNN。

论文中探讨的所有不同的神经网络架构，包括深度神经网络(DNN)、卷积神经网络(CNN)、回归神经网络(RNN)、卷积回归神经网络(CRNN)和深度可分离卷积神经网络(DS-CNN)。

* Deep Neural Network(DNN)

DNN是一种标准的前馈神经网络，由一组全连接层和非线性激活层组成。DNN的输入是扁平的特征矩阵，它被输入到一堆隐藏的全连接的层中，每个层都有n个神经元。通常，每个全连接层后面都有一个基于激活函数的校正线性单元(ReLU)。在输出端是一个线性层，然后是一个softmax层，生成k个关键字的输出概率。

* Convolutional Neural Network(CNN)

基于DNN的KWS的一个主要缺点是不能有效地对语音特征中的局部时间和频谱相关性建模。CNNs利用这种相关性，将输入的时域和频谱域特征作为图像进行二维卷积运算。卷积层之后通常是批处理归一化，基于ReLU的激活函数和可选的最大/平均池化层，可降低特征的维数。在推理过程中，可以将批处理归一化的参数折成卷积层的权值。

* Recurrent Neural Network(RNN)

RNN在语音识别、语言建模、翻译等多序列建模任务中表现出较好的性能。RNN不仅利用输入信号之间的时间关系，而且利用“门控”机制捕获长期依赖关系。与将输入特征作为二维图像处理的CNN不同，RNN对时间步长T进行操作。下图显示了典型RNN模型的模型架构，其中RNN单元可以是LSTM单元，也可以是门控回归单元(GRU)。由于权重在所有T时间步骤中被重用，因此RNN模型的参数数量往往比CNN少。研究表明，与CNN中的批处理归一化类似，应用分层归一化对训练RNN是有益的，其中隐藏状态在每个时间步中被归一化。

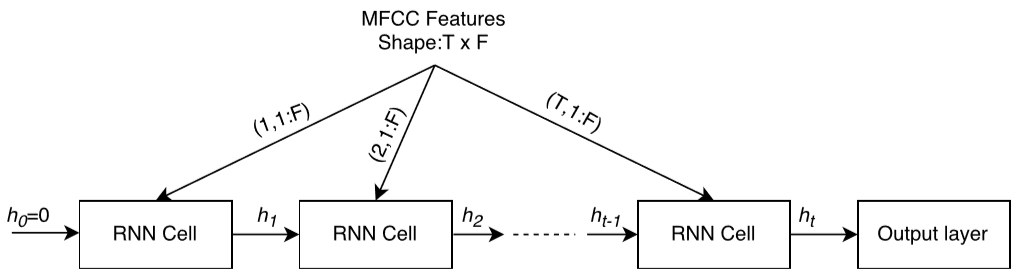


图2-4 RNN模型架构

* Convolutional Recurrent Neural Network(CRNN)

卷积循环神经网络是CNN和RNN的混合体，两者兼而有之。它利用卷积层的局部时间/空间相关性和循环层的语音特征的全局时间依赖性。如下图所示，CRNN模型从卷积层开始，接着是循环层对信号进行编码，然后是全连接层对信息进行映射。其中，循环层为双向，具有多个阶段，提高了网络学习能力。门控回归单元(GRU)作为循环层的基本单元，因为它使用的参数比LSTM少，而且在我们的实验中具有更好的收敛性。

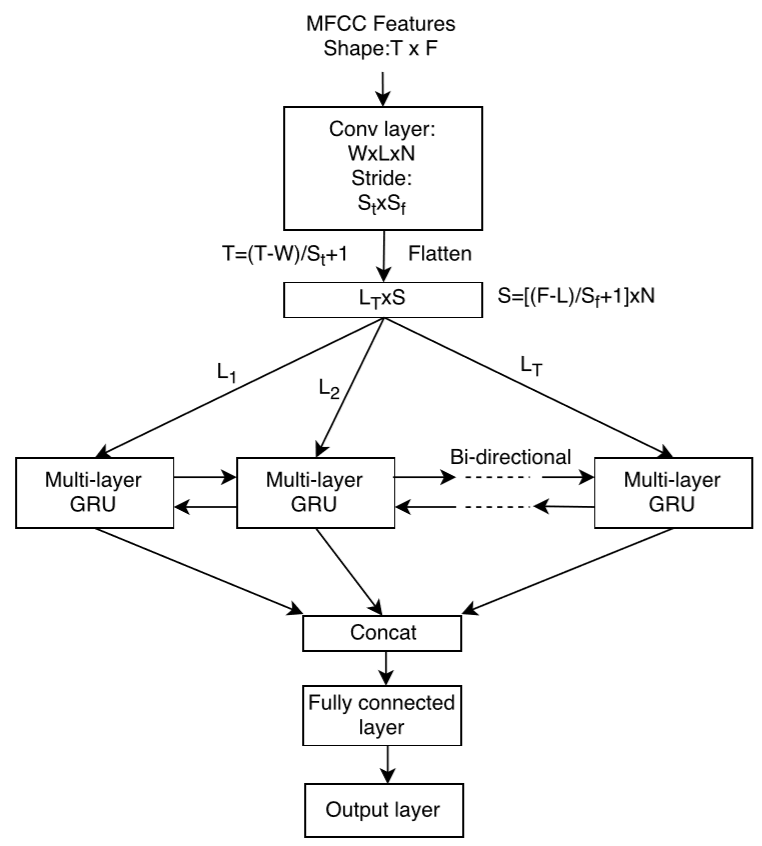


图2-5 CRNN模型架构

* Depthwise Separable Convolutional Neural Network(DS-CNN)

近年来，深度可分离卷积被认为是标准三维卷积运算的有效替代方法，并被用于实现计算机视觉领域的紧凑网络架构。DS-CNN首先用一个单独的二维滤波器对输入特征图中的每个通道进行卷积，然后使用逐点卷积(即1x1)在深度维度上组合输出。通过将标准的三维卷积分解为二维卷积和一维卷积，深度可分离卷积在参数和运算数量上都更加高效，这使得在资源受限的单片机设备中也可以实现更深更广的架构。在论文中采用基于MobileNet实现了深度可分离CNN，如下图所示。最后使用一个平均池和一个全连接层来提供全局交互并减少最后一层中的参数总数。

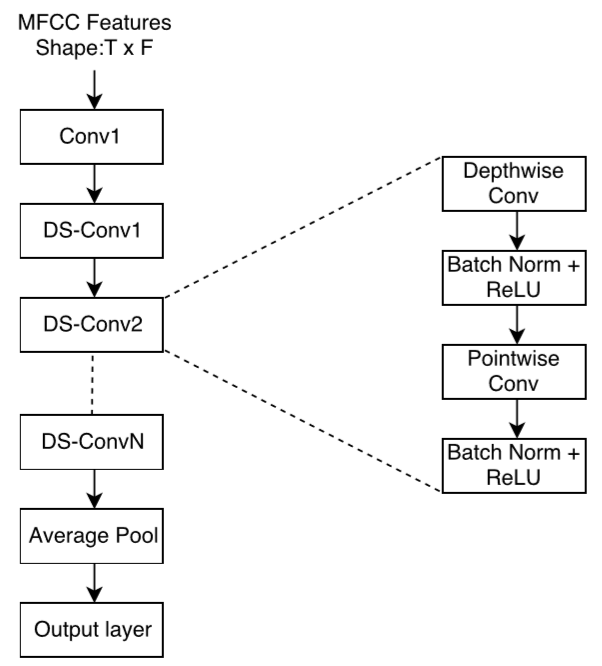


图2-6 DS-CNN模型架构

下表总结了不同神经网络训练的KWS架构的准确性、内存需求和每次推理的操作数。对于所有的模型，使用MFCC特征提取一个长度为40ms，步幅为20ms的语音帧。表中显示的精度为测试集的精度，内存采用8位权值和激活量，足以达到与全精度网络相同的精度。

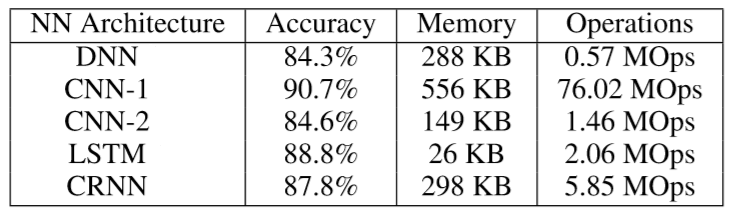


表2-1 不同模型在测试集上的比较

表中的操作数计算了网络中每一层矩阵乘法运算的乘法和加法的总数，代表了整个网络的执行时间。这些结果为KWS的不同神经网络架构提供了有用的见解：

* 虽然神经网络不能达到最佳的准确性，而且往往是内存密集型的，但它们的操作/推 理次数较少，因此非常适合计算能力有限的系统(例如，系统在低工作频率下运行以提高能源效率)
* 另一方面，CNN比DNN具有更高的精度，但需要大量的操作和内存需求。
* LSTM和CRNN在保持良好精度的同时，实现了内存和操作之间的平衡。

内存占用和执行时间是能够在微控制器上运行关键字检测的两个重要考虑因素。在设计和优化运行关键词识别的神经网络时，应考虑这些因素。基于典型的单片机系统配置，论文中提供了神经网络的三组约束条件，针对的是中小型和大型单片机系统，如下表所示。内存和计算限制都是在假定将分配一定数量的资源来运行其他任务(如OS、I/O、网络通信等)的情况下得到的。每个推理限制的操作都假定系统每秒运行10个推理。

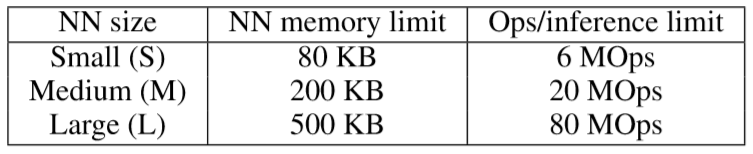


表2-2 神经网络的三组约束条件

下图显示了针对谷歌语音命令数据集训练的的不同神经网络的每次推理的操作次数、记忆需求和测试准确性，以及神经网络在三组约束条件下的计算边界框。理想的模型应该具有较高的精度、较小的内存占用和较低的计算次数，即接近下图中的原点。除了LSTM模型外，其他模型的内存/计算资源太大，不能满足80KB/6MOps内存/计算限制的边界框S。CNN-2、CRNN和DNN模型适合于M和L的包围盒，但精度低于CNN-1模型，后者根本不适合任何一个盒子。

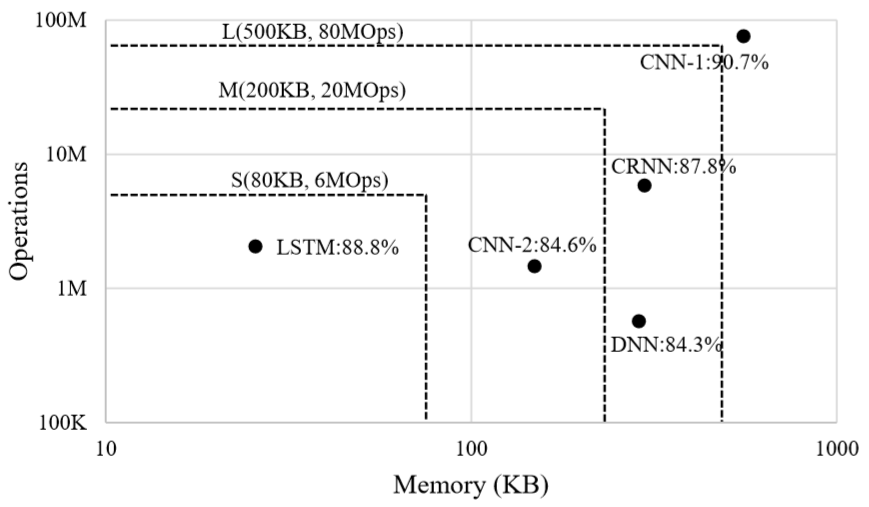


图2-7 不同模型在约束条件下的比较

下表和下图总结了每种神经网络架构的最佳性能模型及其内存需求和操作。从结果可以看出，神经网络是受存储限制的，精度较低，即使模型按比例放大，其饱和度也87%左右。与DNN神经网络相比，CNN神经网络具有更好的准确性，但受最终全连接层的权值的限制。RNN模型(即Basic LSTM、LSTM和GRU)比CNN具有更好的准确率，在某些情况下甚至可以得到更小的模型，而操作次数更少，这表明在相同的资源预算下，利用时间依赖性可以最大限度地提高准确率。CRNN模型结合了CNN和RNN的最佳能，即使在运算量较小的情况下，也比CNN和RNN具有更好的精度。当有更多内存/计算资源可用时，CRNN架构也可以很好地扩展。DS-CNN具有较高的精度和较好的可扩展性，这是由于其更深层次的架构是由计算/内存密集型较低的深度可分离卷积层支持的。

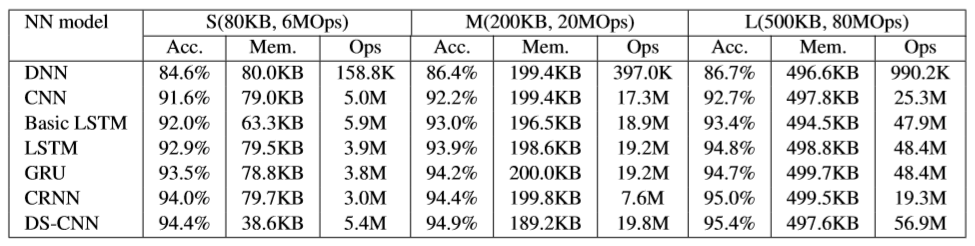


表2-3 约束条件下不同模型的最佳性能比较

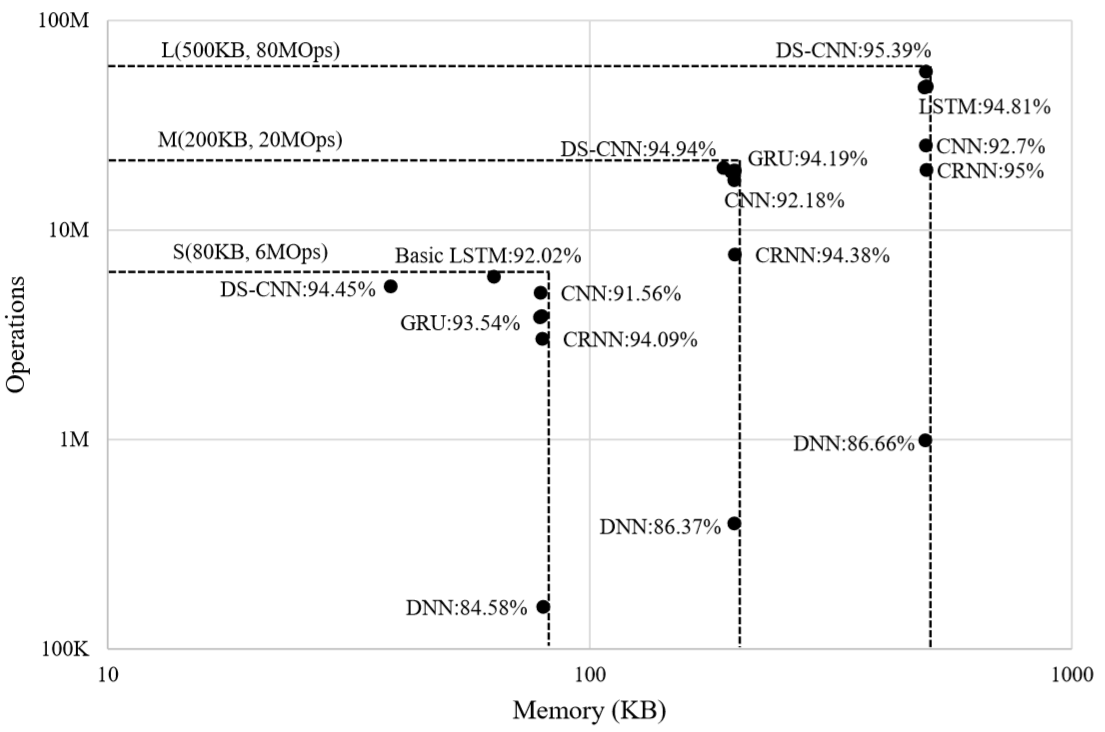


图2-8 约束条件下的最佳性能模型

## 模型量化

神经网络通常使用浮点权重和激活进行训练。以前的研究已经表明，定点权重足以以最小的精度损失运行神经网络。单片机系统内存有限，将32位浮点权重量化为8位定点权重进行部署，从而使模型大大减小。此外，在典型的微控制器中，定点整型操作比浮点操作运行速度快得多，这也是在部署期之前执行量化模型的另一个原因。

每一层权重被量化为8位，通过为每一层找到最优的N，使由于量化造成的精度损失最小化。在所有的权值被量化之后，激活也以类似的方式被量化，从而为每一层找到合适的分数长度N。下表给出了用该方法量化的具有代表性的8位网络的精度，并与原全精度网络进行了比较。经过实测发现，定点量化对识别率影响较小，有时甚至会提升识别率。

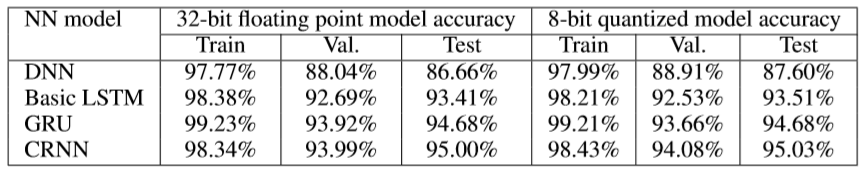


表2-4 8位量化网络与全精度网络的精度比较

由表可知，量化网络的精度要么与全精度网络相同，要么略好于全精度网络，这可能是由于量化带来了更好的正则化。

# 算法实现流程

## 运行环境与使用工具

下表描述了算法实现所用的系统、工具、版本号及下载链接。

|  |  |  |
| --- | --- | --- |
| 系统和工具 | 版本 | 下载链接 |
| Ubuntu | 16.04 | <https://www.Ubuntu.com/download/alternative-downloads> |
| CUDA | 8.0 | https://developer.nvidia.com/cuda-80-download-archive |
| cuDNN | 8.0 | <http://developer.download.nvidia.com/compute/redist/cudnn/v6.0/cudnn-8.0-linux-x64-v6.0.tgz> |
| Tensorflow-gpu | 1.4 | https://tensorflow.google.cn/install/pip?lang=python3 |

表3-1 环境及工具

## 下载项目资源

打开terminal，输入命令：

git clone <https://github.com/ARM-software/ML-KWS-for-MCU>

下载整个项目,得到一个ML\_KWS-for-MCU文件夹。如下图所示。

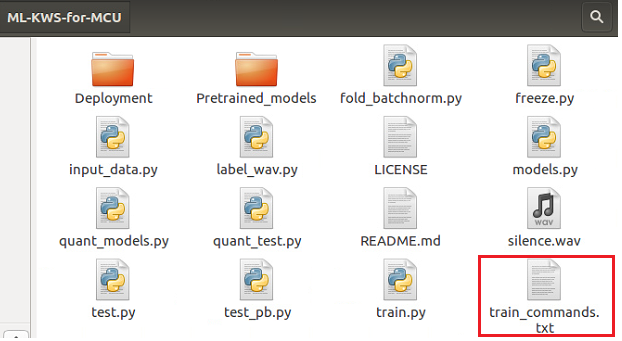


图3-1 ML\_KWS-for-MCU文件夹

这是一个独立的示例，里面的脚本可直接运行。数据集在第一次运行时会自动下载。

## 训练模型

训练模型可直接运行train.py脚本，命令可参考train\_commands.txt里面的命令，部分内容如下图所示。

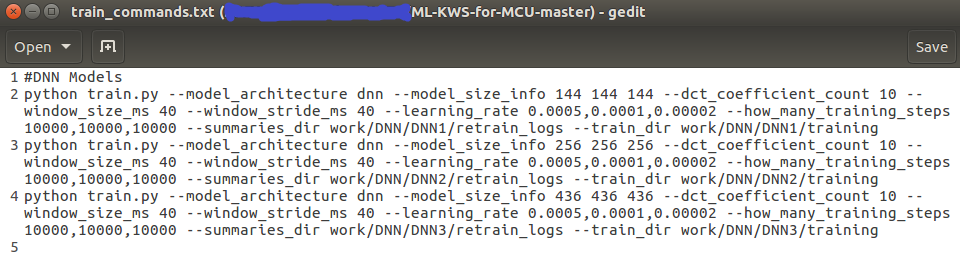


图3-2 部分train\_commands.txt内容

train\_commands.txt文件包含了基于DNN、CNN、LSTM、GRU、CRNN、DS-CNN这几个神经网络的训练命令，并分别设置了不同的超参数。因为这里tensorflow安装在python3环境下，所以训练时需把命令中的python改为python3。

这里选取基于CRNN神经网络的第一条命令进行训练，进入ML-KWS-for-MCU文件夹，打开terminal，命令如下图所示。

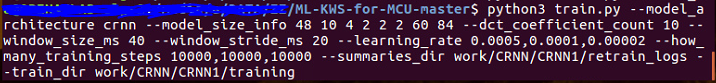


图3-3 训练命令

命令中各超参数含义如下表所示：

|  |  |
| --- | --- |
| model\_architecture crnn | 选用CRNN模型框架 |
| model\_size\_info 48 10 4 2 2 2 60 84 | 模型维度 |
| dct\_coefficient\_count 10 | 每一帧包含的特征数量为10 |
| window\_size\_ms 40 | 每一帧的滑窗长为40ms |
| window\_stride\_ms 20 | 滑窗间隔为20ms |
| learning\_rate 0.0005,0.0001,0.00002 | 指三个阶段的学习率 |
| how\_many\_training\_steps 10000,10000,10000 | 训练时各阶段迭代10000次 |
| summaries\_dir work/CRNN/CRNN1/retrain\_logs | 保存训练日志路径 |
| train\_dir work/CRNN/CRNN1/training | 保存使用的网络模型和检查点 |

表3-2 训练超参数含义

第一次运行时，会自动下载数据集，大小约2.3G，保存在目录/tmp/speech\_dataset/下面，并且会自动解压。

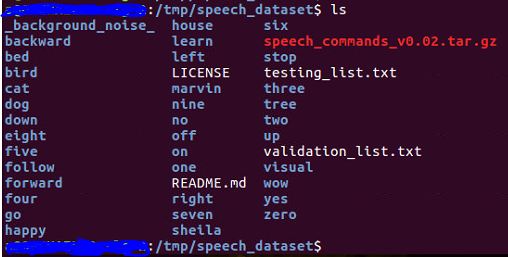


图3-4 下载的数据集

此数据集包含30几个命令词，每个命令词含有约3800个左右的1秒的wav音频文件，每个wav文件包含一个命令词对应的口语单词。虽然数据集中含有几十个命令词，但训练脚本默认只选择10个命令词进行训练，分别是“yes”、“no”、“up”、“down”、“left”、“right”、“on”、“off”、“stop”和“go”，其他命令词做为未知词（unkonwn）训练。可参见train.py脚本中第421行，如下图所示。

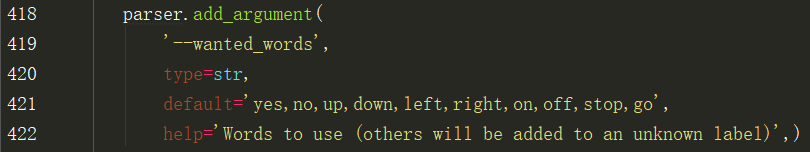


图3-5 默认训练的命令词

数据集下载完成后会自动开始训练， 随着训练的进行，它将打印出accuracy和loss，到最后accuracy应该会超过90%，部分训练过程如下图所示。

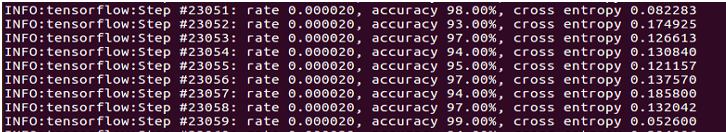


图3-6 部分训练过程

训练时间约2.5小时左右。最终训练准确率在97%附近。

生成的模型保存在如下图所示的路径中。

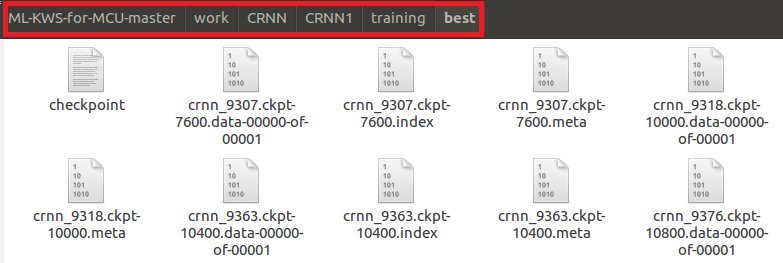


图3-7 模型保存路径

## 测试

打开test.py脚本，在import之后一行加入From six.moves import xrange，如下图所示。这是为了解决可能出现的报错信息“没有xrange模块”。

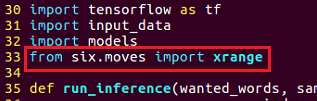


图3-8 导入xrange模块

打开terminal，命令如下图所示。

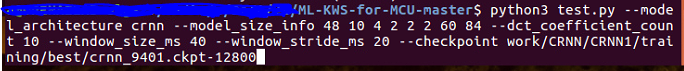


图3-9 测试命令

其中超参数“checkpoint”表示模型的检查点，即加载保存模型的路径。其他超参数与训练命令中的一样。运行此脚本会打印出训练集、验证集和测试集上的准确率。

## 模型转pb文件

脚本文件freeze.py可以把训练生成的模型转为二进制的pb文件，这样才可以部署到Android、iOS或EAIDK板上。命令如下图所示。

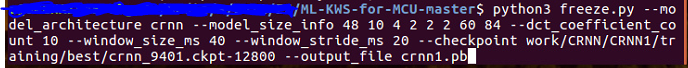


图3-10 模型转pb文件命令

其中超参数“output\_file”为输出文件。如下图所示。

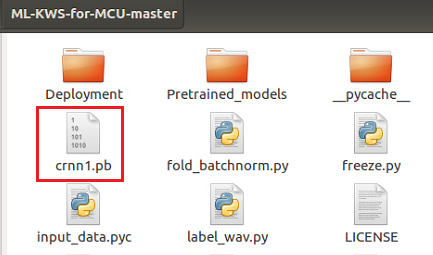


图3-11 生成crnn1.pb文件

注意：这里运行的是原始的freeze.py脚本，只生成了一个模型文件，但在部署时不仅需要模型文件，还需要对应的网络结构pbtxt文件。因此我们对freeze.py文件进行了修改，修改后的freeze.py文件请参见附件二。可用修改后的freeze.py文件替换原始的文件进行转换，并生成对应的pbtxt文件，如下图所示。

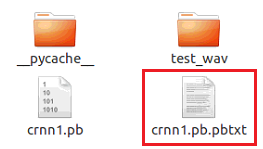


图3-12 网络结构pbtxt文件

## 检验模型识别效果

label\_wav.py脚本可以用实际录制命令词的音频检验模型识别效果。这里有2个不同的音频：0001.wav是一段1秒时长的非命令词的音频；0a2b400e\_nohash\_0.wav是1秒时长命令词为“stop”的音频。如下图所示。

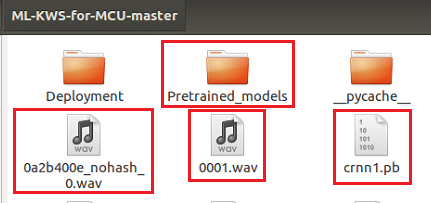


图3-13 检验模型效果音频

这里检验的音频可以自己录制，然后拷到该目录下即可，注意录制的音频时长不能小于1秒，也不要太长（不超过2秒），因为训练音频的时长都为1秒，检测时也只会检测前1秒的音频，不会检测1秒之后的音频。录制时最好在1秒内说出命令词。

打开terminal，命令如下图所示。



图3-14 检验命令

其中超参数的含义如下表所示：

|  |  |
| --- | --- |
| wav 0001.wav | 加载音频文件0001.wav |
| graph crnn1.pb | 加载模型crnn1.pb |
| labels Pretrained\_models/labels.txt | 加载标签labels.txt |
| how\_many\_labels 1 | 打印分类置信度得分最高的1个类别 |

表3-3 超参数含义

加载的wav文件、模型在如图3-13中，标签在图3-15所示的路径中。

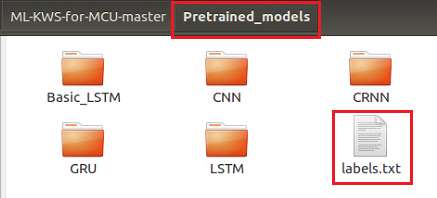


图3-15 标签文件

标签文件内容如下图所示。

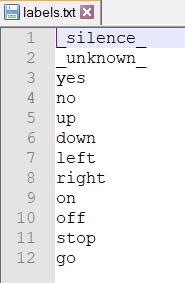


图3-16 labels.txt文件内容

其中silence表示无声，unkonwn表示未知词，其他10个代表10个不同的命令词。

参数中指定的模型、标签和wav文件将被加载，然后根据音频数据输出模型的预测结果，打印到控制台。如下图所示。



图3-17 识别结果

因为0001.wav文件是非命令词的音频，所以属于未知类（unkonwn），识别正确，得分为0.99868。

若用命令词为“stop”的音频文件0a2b400e\_nohash\_0.wav检验，命令和预测结果分别如图3-18和3-19所示。



图3-18 检验命令



图3-19 识别结果

由打印结果可知，模型识别正确，音频命令词为stop，得分为0.99990。

# 模型部署

这里模型部署到我们自己的开发板EAI610上。EAI610是EAIDK中的一款硬件平台。EAIDK（Embedded AI Development Kit）是为AI开发者精心打造，面向边缘计算的人工智能开发套件。为AI应用提供简洁、高效、统一的API接口，加速终端AI产品的场景应用落地。

这里部署的模型选用ds-cnn网络训练的模型，部署所需的工具和脚本全部放在附件一中的KWS-Deployment文件夹中。如下图所示。

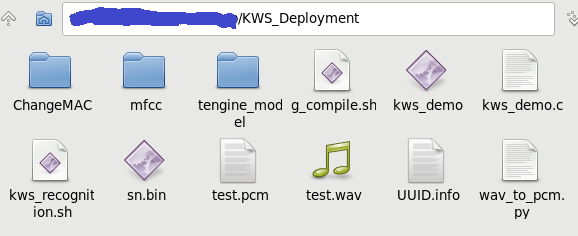


图4-1 部署所需文件

## g++编译

在执行以下命令之前，先删除图4-1中所示的sn.bin文件。

在进行命令词识别之前需进行g++编译。脚本g\_complie.sh里面包含了g++编译的命令，如下图所示。

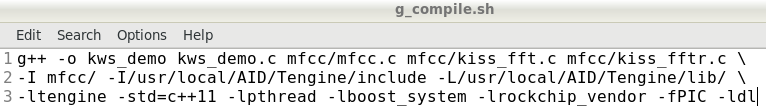


图4-2 g++编译命令

运行这个脚本即可完成编译：

sh g\_complie.sh

## 音频采集及实时识别

这里只需执行kws\_recognition.sh这个脚本即可完成音频采集及识别，脚本中含有几个命令，如下图所示。

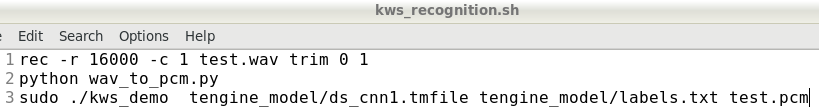


图4-3 音频采集及识别命令

执行这个脚本之前需安装sox音频处理工具进行录音。安装命令如下：

sudo dnf install sox

安装完成即可利用上图中第1行命令进行录音。其中参数含义如下表所示。

|  |  |
| --- | --- |
| rec | 录音 |
| -r 16000 | 采样率为16000 |
| -c 1 | 通道数为单通道 |
| test.wav | 音频名称 |
| trim 0 1 | 从第0秒开始录音，只录制1秒 |

表4-1 录音命令参数含义

生成的音频文件如图4-1中的test.wav文件。

第2行命令是把第一行命令采集的音频的wav格式转为pcm格式。生成的pcm文件如图4-1中的test.pcm文件。

第3行命令即对采集的音频进行识别。其中参数含义如下表所示。

|  |  |
| --- | --- |
| ./kws\_demo | 运行文件 |
| tengine\_model/ds\_cnn1.tmfile | 加载tengine模型ds\_cnn1.tmfile |
| tengine\_model/labels.txt | 加载标签文件labels.txt |
| test.pcm | 加载音频test.pcm |

表4-2 识别命令参数含义

脚本kws\_recognition.sh把录音、音频格式转换及识别这三个功能放在一起，实际应用时只需执行一行命令：

sh kws\_recognition.sh

此命令会立即采集1秒的音频，因此执行这个命令后，需在1秒中内对麦克风说出命令词，EAI610开发板自带有麦克风，也可利用USB连接的麦克风进行录音，录音时尽量使周围环境噪声较小、发音准确。然后就可在显示屏上看到识别结果，而未感觉到较大的延时。识别效果如下图所示。

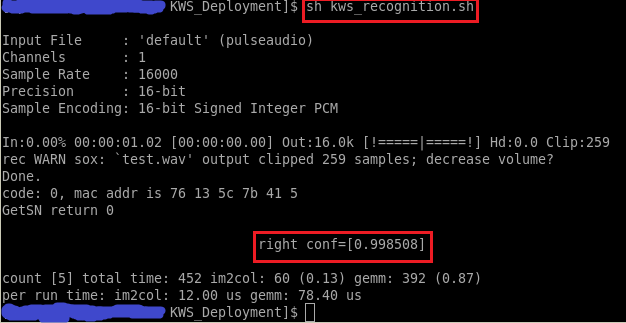


图4-4 识别命令及结果

如上图所示，结果会打印出置信度最高的命令词及得分。

# 附件一

## 1.1 freeze.py脚本

应用于第5节，生成pb模型及pbtxt网络结构。



## 1.2 部署demo

