

申请上海交通大学博士学位论文

语音识别深度序列建模及解码

论文作者 陈哲怀

学 号 0140339029

导 师 俞凯教授

副 导 师 钱彦旻教授

专 业 计算机科学与技术专业

答 辩 日期 2018 年 12 月 1 日



Submitted in total fulfillment of the requirements for the degree of Doctor  
in A Very TODO Major

# Deep Sequential Modeling and Decoding in Speech Recognition

ZHEHUAI CHEN

Advisor

Prof. KAI YU

Co-advisor

Prof. YANMIN QIAN

DEPART OF XXX, SCHOOL OF XXX

SHANGHAI JIAO TONG UNIVERSITY

SHANGHAI, P.R.CHINA

Dec. 1st, 2018



## 上海交通大学 学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名: 某某

日 期: 某 年 某 月 某 日

6

aaaaa

上海交通大学  
学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保 密  在 \_\_\_\_\_ 年解密后适用本授权书。

本学位论文属于

不保密

(请在以上方框内打√)

学位论文作者签名: 某某

指导教师签名: 某某

日 期: 某 年 某 月 某 日

日 期: 某 年 某 月 某 日



## 语音识别深度序列建模及解码

### 摘要

语音识别中一个独有并且有趣的自然现象是声学序列和语言学序列的长度可变性，这使得语音识别需要同时建模两种序列之间的状态对齐与模式识别。在训练阶段，一组带有已知标签的输入特征被提供给系统进行模型构建，序列建模和模式识别是前一问题的两大核心，这决定了语音识别系统可达到的识别精度的上限；而测试阶段，则基于特征序列和其它知识源如语言模型和字典进行模型搜索解码，这决定了识别速度和实际可达的识别精度。近年来，深度学习模型被引入到语音识别的声学和语言建模当中替代传统分类器，显著改善了模式识别问题的精度。基于深度学习的语音识别由于只是替换了分类器，语音识别的序列建模和搜索解码未有本质改变。本论文围绕深度学习背景下的序列建模和搜索解码技术展开了一系列探索和研究。

本论文针对关键词检测和多说话人重叠语音信号识别这两类非传统语音识别任务提出了序列建模方案。序列建模方法通常只在训练标准语音识别模型时进行使用，但针对其它非传统识别任务的序列建模研究并不充分，这包括：关键词检测任务和多说话人重叠语音信号识别任务等。这类任务仍然是序列预测问题，但是却没有合适的训练准则和相应的设计，来充分优化分类器的序列建模能力。为了将序列鉴别性训练引入关键词检测任务，核心难题是设计相应的竞争可能性建模方法。本论文提出采用无词图鉴别性训练框架来解决这一问题：隐性使用音素或半词单元的语言模型来建模。另一方面，单通道多说话人混叠语音识别也属于序列级问题，因此序列鉴别性准则将有助于这样的序列分类问题。我们提出了一种传统鉴别性训练技术变种，它在进行鉴别性训练的同时，也抑制输出通道上说话人跟踪错误。通过联合优化，迁移学习，序列鉴别性训练等方式，我们改善了原来语音分离、信号增强和语音识别的联合训练系统。

本论文从解码搜索并行化和降低搜索复杂度两个层面进行搜索解码算法加速。一方面，本论文提出并行的解码搜索算法并在 GPU 上实现开源该套算法，该框架可以显著加速现有推理搜索算法。语音识别解码网络中多数 WFST 边之间并不直接相关，具有并行处理的可能性；基于 GPU 的并行加速也已成功应用于声学分数的计算。但是基于 GPU 并行计算的 WFST 解码不容易实现，且现有研究存在诸多缺陷。针对设计中的几大难点，本论文提出了如下解决方案：本论文将维特比算法中的令牌合并操作实现为一个 GPU 并行计算中的原子操作，以便减少维特比束剪枝算法中同步消耗；提出了动

态负载均衡的方式以更高效地进行并行计算，提高其多线程之间的利用率；重新设计了基于 GPU 并行计算的精确的词图生成和剪枝算法，以便充分利用 GPU 的性能特点。在 Switchboard 上实验表明，本论文所提出的方法在取得完全一致的 1-best 和词图质量情况下，可以得到 3-15 倍的加速。除此之外，如果再进行多句子的并行处理，最终的加速比将达到 46 倍。所提出的离线解码器对语言模型和声学模型没有特别的限制，并且可以工作在各种架构的 GPU 上。

另一方面，本论文基于端到端建模，系统地提出了标签同步算法，其通过一系列方法使得搜索解码过程从逐帧同步变为标签同步，从而加速解码。当前主流的推理搜索方法是帧层面的维特比束搜索算法，该算法复杂度很高，限制了语音识别的广泛应用。本论文提出将特征层面的搜索过程改变为标签层面，使得解码速率等于标注速率，从而小于特征速率。具体来说，在标签搜索阶段，对帧层面声学模型的输出增加一步后处理过程，得到对每个输出标签概率计算的近似值，再进行标签搜索。与传统方法相比，该方法的优势是搜索空间更小，且搜索过程被大大加速。本论文提出的一系列通用方法在隐马尔科夫模型和连接时序分类模型上得到了验证，取得大幅度语音识别解码速度改善。

标签同步算法同时还能产生高质量的音素词图，称为 CTC 音素词图。基于 CTC 音素词图，本论文进一步探讨标签同步解码算法的一些扩展应用。这包括关键词检测，多识别任务统一置信度框架，以及端到端语音识别。

我们在关键词检测中，基于前述 LSD 算法得到的音素词图，提出了一套基于编辑距离的后处理算法以引入音素混淆性，使系统更加鲁棒。在前面提出的标签同步解码算法的基础上，本论文进一步提出了两种置信度生成算法。更细致的研究显示这种基于 CTC 的音素词图是得到更好性能的关键所在。本论文还提出了辅助归一化搜索空间的概念，并尝试使用这样的搜索空间来建模所有 ASR 应用领域的置信度。而针对这样做在低功耗设备上带来的挑战，本论文采用基于 CTC 的标签同步解码来进行处理，由此带来了很大的效率改善。同时本论文还研究了将标签同步算法应用于直接建模输出序列形态学组合的端到端模型的方案。本论文使用模块化训练的思想来改善端到端模型建模，使其更易于使用外在知识源来训练每一个端到端模型的子模块。值得注意的是，模型最后需要进行联合优化，因此最终在推理搜索阶段，模型仍然工作在端到端模式下。在实验部分，本论文提出系统一方面取得大幅度语音识别解码速度改善，另一方面在端到端建模上取得了更快和更好的模型收敛和模型准确度。

总而言之，本论文围绕深度学习背景下的序列建模和解码搜索技术展开了一系列探索和研究。本论文提出的并行解码搜索技术与标签同步解码算法相结合，可以得到一个极大的速度提升；再结合所提出的通用推理搜索和置信度方案，将得到一个低功耗、高性能的通用语音识别系统。本论文不仅在主要数据集上验证了相关算法，并且开源了

部分算法系统。

**关键词：**语音识别，序列建模，搜索解码，平行计算，标签同步解码，置信度，序列鉴别性训练



# **Deep Sequential Modeling and Decoding in Speech Recognition**

## **ABSTRACT**

TODO

**KEY WORDS:** TODO, TODO



# 目 录

插图索引	xii
表格索引	xiv
主要符号对照表	xv
<b>第一章 绪论</b>	<b>1</b>
1.1 自动语音识别 . . . . .	1
1.1.1 语音识别简史 . . . . .	1
1.1.2 语音识别架构 . . . . .	4
1.2 语音识别中的序列建模 . . . . .	6
1.3 语音识别中的解码搜索 . . . . .	7
1.4 论文主要内容、创新点及组织结构 . . . . .	8
<b>第二章 自动语音识别及深度序列模型</b>	<b>11</b>
2.1 自动语音识别框架 . . . . .	11
2.1.1 特征提取 . . . . .	11
2.1.2 声学模型 . . . . .	13
2.1.3 语言模型 . . . . .	20
2.1.4 解码及搜索 . . . . .	22
2.2 深度学习在语音识别中的应用 . . . . .	22
2.2.1 深度学习与神经网络 . . . . .	22
2.2.2 神经网络训练 . . . . .	28
2.2.3 深度神经网络-隐马尔科夫模型混合系统 . . . . .	32
2.3 基于深度学习的序列建模 . . . . .	34
2.3.1 生成式序列模型和鉴别式序列模型 . . . . .	34
2.3.2 基于 HMM 的序列鉴别性训练 . . . . .	36
2.3.3 基于 CTC 的序列鉴别性训练 . . . . .	37
2.4 语言识别中的端到端模型 . . . . .	38
2.4.1 无词图鉴别性训练的模型 . . . . .	39

2.4.2	连接时序分类模型 . . . . .	41
2.4.3	基于注意力机制的序列到序列建模 . . . . .	41
2.5	本章小结 . . . . .	42
<b>第三章</b>	<b>语音识别中的解码搜索问题</b>	<b>43</b>
3.1	维特比解码搜索算法 . . . . .	43
3.2	语音识别中的解码搜索问题 . . . . .	44
3.2.1	关键词检测 . . . . .	46
3.2.2	孤立词识别、语法网络识别和大词汇连续语音识别 . . . . .	48
3.2.3	解码搜索问题复杂度分析 . . . . .	49
3.3	解码器技术综述 . . . . .	51
3.3.1	解码器的技术流派 . . . . .	51
3.3.2	主要模块和核心算法 . . . . .	54
3.4	解码器研究中的待解决问题 . . . . .	62
3.4.1	研究现状总结 . . . . .	62
3.4.2	发展机遇和亟待解决问题 . . . . .	63
3.5	本章小结 . . . . .	65
<b>第四章</b>	<b>非传统识别任务的序列建模</b>	<b>67</b>
4.1	关键词检测的序列建模 . . . . .	67
4.1.1	关键词检测中传统序列建模方法的缺陷 . . . . .	68
4.1.2	关键词检测的序列鉴别性训练 . . . . .	68
4.1.3	实验结果 . . . . .	72
4.2	鲁棒语音识别中的深度序列建模 . . . . .	76
4.2.1	排列不变性训练及其在鸡尾酒会问题中应用 . . . . .	77
4.2.2	基于迁移学习的排列不变性训练 . . . . .	77
4.2.3	基于序列鉴别性训练的排列不变性训练 . . . . .	79
4.2.4	实验结果 . . . . .	82
4.3	本章小结 . . . . .	85
<b>第五章</b>	<b>基于 GPU 并行计算的解码速度优化</b>	<b>87</b>
5.1	维特比算法并行化框架 . . . . .	88
5.2	并行的令牌传递算法 . . . . .	89
5.3	图搜索的动态负载均衡算法 . . . . .	90

5.4 并行的词图处理算法 . . . . .	90
5.4.1 精确的词图生成算法 . . . . .	90
5.4.2 词图剪枝 . . . . .	92
5.5 实验结果 . . . . .	93
5.5.1 实验配置 . . . . .	93
5.5.2 性能和速度 . . . . .	94
5.5.3 分析 . . . . .	95
5.6 本章小结 . . . . .	97
<b>第六章 基于标签同步解码的搜索空间优化</b>	<b>99</b>
6.1 基于连接时序分类模型的标签同步解码 . . . . .	100
6.1.1 标签同步解码 . . . . .	100
6.1.2 标签同步解码算法实现 . . . . .	102
6.2 基于无词图鉴别性训练的模型的标签同步解码 . . . . .	102
6.2.1 标签同步解码 . . . . .	102
6.2.2 标签同步解码算法实现 . . . . .	104
6.3 逐帧同步解码与标签同步解码的对比 . . . . .	105
6.4 实验结果 . . . . .	105
6.4.1 DSM 实验 . . . . .	107
6.4.2 GSM 实验 . . . . .	110
6.5 本章小结 . . . . .	113
<b>第七章 标签同步解码的扩展应用</b>	<b>115</b>
7.1 基于标签同步解码的关键词检测 . . . . .	115
7.1.1 模型训练 . . . . .	116
7.1.2 后处理 . . . . .	116
7.1.3 实验结果 . . . . .	117
7.2 多识别任务统一置信度框架 . . . . .	120
7.2.1 置信度与搜索空间 . . . . .	121
7.2.2 基于标签同步解码的置信度框架 . . . . .	122
7.2.3 基于附属归一化搜索空间的置信度方法 . . . . .	126
7.2.4 实验结果 . . . . .	127
7.3 基于标签同步解码的声学特征到词的端到端语音识别 . . . . .	135
7.3.1 训练和解码框架 . . . . .	136

---

7.3.2 模块化 . . . . .	137
7.3.3 标签同步解码 . . . . .	137
7.3.4 联合训练 . . . . .	138
7.3.5 实验结果 . . . . .	138
7.4 本章小结 . . . . .	140
<b>第八章 全文总结</b>	<b>141</b>
8.1 非传统识别任务的序列建模 . . . . .	141
8.2 基于 GPU 并行计算的搜索速度优化 . . . . .	142
8.3 基于标签同步解码的搜索空间优化 . . . . .	142
8.4 标签同步解码的扩展应用 . . . . .	143
8.5 后续工作展望 . . . . .	144
<b>参考文献</b>	<b>145</b>
<b>攻读学位期间发表的学术论文</b>	<b>165</b>
<b>攻读学位期间申请的专利</b>	<b>169</b>
<b>致 谢</b>	<b>171</b>

## 插图索引

1-1 语音识别词错误率变迁图(截止 2009 年) . . . . .	3
1-2 语音识别框架 . . . . .	5
2-1 单高斯三音素的状态聚类 . . . . .	14
2-2 隐马尔科夫模型 . . . . .	16
2-3 深度神经网络 . . . . .	23
2-4 激活函数 . . . . .	24
2-5 卷积操作和池化操作 . . . . .	25
2-6 受限玻尔兹曼机 . . . . .	31
2-7 DNN-HMM 混合系统 . . . . .	33
2-8 HMM, CTC 和本文提出的方法中隐藏状态拓扑结构示意图 . . . . .	36
2-9 端到端系统举例 . . . . .	39
3-1 有限状态语法网络网络以及 unigram 和 monophone . . . . .	49
3-2 Bigram 和 monophone 的语法网络 . . . . .	50
3-3 Bigram 和 cross-word triphone 的语法网络 . . . . .	50
3-4 Trigram 和 within-word triphone 的语法网络 . . . . .	52
3-5 通用解码器架构 . . . . .	55
3-6 N-Best 候选序列和词图 . . . . .	61
4-1 基于 <code>filler</code> 解码的搜索空间结构图 . . . . .	71
4-2 PIT-ASR 联合训练与模块化初始化和逐步联合训练比较 . . . . .	78
4-3 逐步联合训练模型系统框架 . . . . .	79
4-4 基于迁移学习逐步联合训练方法 . . . . .	80
4-5 原始联合训练和所提出的方法在验证集上学习曲线比较 . . . . .	81
4-6 50 小时数据集的解码例子 . . . . .	84
5-1 并行维特比束剪枝算法以及精确词图处理系统的框架 . . . . .	88
5-2 一个针对动态负载均衡的例子 . . . . .	91
5-3 语言模型大小, 帧率, GPU 架构的比较 . . . . .	96
5-4 针对解码器的时间占比分析 . . . . .	97

---

6-1 HMM, CTC 的模型输出分布示例 . . . . .	101
6-2 LSD 和 FSD 框架中平均活跃令牌数随 LM 变大的变化趋势 . . . . .	108
6-3 swb 子集, CTC 中, 使用不同剪枝技术时 WER 随平均活跃令牌数的变化趋势 . . . . .	109
6-4 LF-MMI 中, 使用不同剪枝技术时 WER 随平均活跃令牌数的变化趋势 . . . . .	111
7-1 MED 方法框架 . . . . .	117
7-2 非固定关键词的关键词检测的 ROC 曲线比较 . . . . .	119
7-3 LSD CTC Lattice 的例子 . . . . .	124
7-4 FSD HMM 和 LSD CTC 所产生的词图比较 . . . . .	125
7-5 不同语音识别任务的架构比较 . . . . .	128
7-6 OPER v.s. 词图密度在 FSD 和 LSD 中的比较 . . . . .	129
7-7 词图密度 v.s. 相对 OWER 下降比例 . . . . .	130
7-8 LSD 和 FSD 的词边界稳定性 . . . . .	131
7-9 词图密度 v.s. 混淆网络深度 . . . . .	131
7-10 模块化训练策略的框架 . . . . .	136

## 表格索引

3-1 知名开源解码器 . . . . .	54
4-1 基于 <i>HMM</i> 的序列鉴别性训练的模型框架 . . . . .	73
4-2 基于 <i>HMM</i> 的鉴别性训练的准则比较 . . . . .	73
4-3 序列鉴别性训练系统的后处理 . . . . .	74
4-4 非固定关键词的关键词检测中的性能和速度比较 . . . . .	74
4-5 固定关键词的 <i>KWS</i> 系统的性能和速度比较 . . . . .	76
4-6 50 小时数据集上的性能改善总结 . . . . .	83
4-7 150 小时数据集上的性能比较 . . . . .	84
5-1 <i>l-best</i> 和词图性能比较 ( <i>beam=14</i> ) . . . . .	94
5-2 所提出系统的速度比较 ( <i>beam=14</i> ) . . . . .	95
6-1 <i>LSD</i> versus <i>FSD</i> in <i>DSM</i> . . . . .	107
6-2 <i>LSD</i> 与跳帧方法的对比 . . . . .	108
6-3 <i>LSD</i> 与 <i>FSD</i> 在不同的 <i>GSM</i> 模型上的性能和速度比较 . . . . .	111
6-4 生成式序列模型中的 <i>blank</i> 粒度 . . . . .	112
6-5 生成式序列模型中的 <i>blank</i> 拓扑结构 . . . . .	112
7-1 音素 <i>CTC</i> 在是否包含 <i>wb</i> 时的模型性能 . . . . .	118
7-2 <i>CTC</i> 关键词检测系统的后处理 . . . . .	118
7-3 非固定关键词的关键词检测中的性能和速度比较 . . . . .	118
7-4 固定关键词的 <i>CTC</i> 的 <i>KWS</i> 系统的性能和速度比较 . . . . .	120
7-5 <i>WER</i> 比较 . . . . .	128
7-6 针对音素声学置信度的比较 . . . . .	132
7-7 基于混淆网络的置信度比较 . . . . .	133
7-8 <i>KWS</i> 任务 . . . . .	133
7-9 基于上下文的语音识别 . . . . .	134
7-10 <i>LVCSR</i> 任务 . . . . .	134
7-11 各个模块的性能比较 . . . . .	138
7-12 针对是否包含模块化训练的性能比较 . . . . .	139



# 主要符号对照表

## 通用标记

$s$	标量使用普通小写字母
$\mathbf{v}$	列向量使用加粗的小写字母
$\mathbf{X}$	矩阵使用加粗的大写字母
$\mathcal{F}$	训练准则
$\mathcal{M}$	模型参数
$\mathbf{O}$	观测向量序列, $\mathbf{O} = [\mathbf{o}_1, \dots, \mathbf{o}_T]^\top$
$\mathbf{o}_t$	第 $t$ 帧的观测特征
$\mathbf{w}$	词(标注或者假设)序列
$a_{i,j}$	离散的状态转移概率
$b_j(\mathbf{o})$	状态 $j$ 的状态输出分布
$m$	高斯成分的索引
$\boldsymbol{\mu}^m$	第 $m$ 个高斯成分的均值向量
$\boldsymbol{\Sigma}^m$	第 $m$ 个高斯成分的协方差矩阵
$\mathbf{W}^l$	神经网络第 $l$ 层的权重矩阵
$\mathbf{b}^l$	神经网络第 $l$ 层的偏置向量
$\mathbf{x}^l$	神经网络第 $l$ 层的激励向量
$\mathbf{y}^l$	神经网络第 $l$ 层的激活向量

## 数学标记

$p(\cdot)$	概率密度函数
$p(\cdot \cdot)$	条件概率密度函数
$P(\cdot)$	概率质量函数
$\{\cdot\}^\top$	向量或矩阵的转置
$\{\cdot\}^{-1}$	方阵的逆

## 英文缩写

<b>AM</b>	声学模型
<b>ASR</b>	自动语音识别
<b>CI</b>	上下文无关
<b>CD</b>	上下文相关

---

<b>CE</b>	交叉熵
<b>CMN</b>	倒谱均值归一化
<b>CTC</b>	连接时序分类模型
<b>CVN</b>	倒谱方差归一化
<b>CN</b>	混淆网络
<b>DNN</b>	深度神经网络
<b>DSM</b>	鉴别式序列模型
<b>GMM</b>	混合高斯模型
<b>GPU</b>	图形处理芯片
<b>GSM</b>	生成式序列模型
<b>HMM</b>	隐马尔科夫模型
<b>KWS</b>	关键词检测
<b>LM</b>	语言模型
<b>LSD</b>	标签同步解码
<b>LSTM</b>	长短时记忆网络
<b>LVCSR</b>	大词汇连续语音识别
<b>MAP</b>	最大后验估计
<b>MSE</b>	均方差
<b>SGD</b>	随机梯度下降
<b>WER</b>	词错误率

# 第一章 绪论

## 1.1 自动语音识别

在日常生活中，语音是人与人之间交流最主要也是最有效方式。目前人机交互方式主要由键盘、鼠标和触摸屏来完成，随着移动设备的不断发展，过去的人机交互方式已经不再适用。用语音来进行人机交互能极大提高移动设备的易用性。使用语音来进行人机交互包含语音识别、语义理解、对话管理和语音合成等关键技术，而其中自动语音识别 (Automatic Speech Recognition, ASR) 作为整个闭环的入口无疑是最重要的环节。大词汇连续语音识别 (Large Vocabulary Continuous Speech Recognition, LVCSR) 是语音识别的经典任务，它的功能就是将人的语音转换为相应文本或者指令以用于后续的处理，这也是各种技术发展的核心测试任务；语音识别的另一大类应用是关键词检测 (Keyword Spotting, KWS)，它的目标是得到一个高准确度和高效率的识别器，用于检测特定的一些关键词在语音中的出现。

语音识别中一个独有并且有趣的自然现象是声学序列和语言学序列的长度可变性，这使得语音识别需要同时建模两种序列之间的状态对齐与模式识别。模式识别，序列建模，搜索解码是语音识别中的三部分主要问题，下面将进行介绍。

### 1.1.1 语音识别简史

最早的语音识别系统出现在 1952 年贝尔实验室 [1]，这是一个只能进行孤立数字识别的系统。它没有使用通用计算机和任何统计机器学习的方法，只是一个精心设计端到端电路。现代的语音识别的基础开始于 70 年代，代表是隐马尔科夫 (Hidden Markov Model, HMM) 模型提出 [2, 3]。在这个模型中，观测特征的生成过程可以被两个条件概率所描述，发声的过程被描述为一个随机生成过程：状态转移概率和状态输出概率。HMM 被用来对发声单元进行建模，发声单元包括音素，词或者句子。在接下来的几十年中，随着高斯混合模型 (Gaussian Mixture Model, GMM) 被用于模拟状态输出概率和 GMM-HMM 理论的不断发展以及计算资源的不断改进，语音识别得到了显著的发展。语音识别任务也从简单的孤立词识别任务发展到大词汇量连续语音识别任务。从 1988 年开始，美国国家标准与技术研究所 (National Institute of Standards and Technology, NIST) 以及美国国防部高级研究计划局 (Defence Advanced Research Project Agency, DARPA) 联合组织几场对连续词汇语音识别评估。这些评估极大地推动了语音识别研究的发展，并

为语音识别设定了几个里程碑。语音识别词汇从 1988 年的资源管理任务中的 900 个单词改进到 1993 年华尔街日报中的 20000 个单词，实现了真正的词汇连续语音识别。不仅词汇量增加，而且语音识别的任务也朝着更现实的识别任务发展。例如，记录环境从干净的环境变为嘈杂的环境，并且记录工具从专用记录设备变为普通电话语音。随着记录环境变得更加复杂，优化的目标也从孤立的单词变为连续的单词序列预测。在 20 世纪 90 年代后期，在 HMM 的基础上，研究人员进一步提出了自适应和自适应训练技术 [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16] 来应对不断复杂的语音环境，以及序列鉴别性训练技术 [17, 18, 19, 20, 21, 22, 23] 来使用序列级准则进行模型优化。这两项技术是在深度神经网络 (Deep Neural Network, DNN) 提出之前对 GMM-HMM 系统在复杂环境下提升性能最核心的技术。苹果手机第一代 Siri 中使用的就是这些技术。

另一方面，语言模型在自动语音识别领域中，作为一个声学模型重要的“帮手”而存在。语言模型的作用是对一句话中有多大可能在自然语言语料之中出现进行建模。统计语言模型是在词语序列（即句子）层面的一个概率分布，比如说在给定  $m$  个词的句子中，语言模型的任务是获取整个句子的概率表示。N-gram 语言模型 [24] 由其简单和有效而被广泛地使用，但是由于模型特性，其受到了“维度诅咒”，即在现实语料中历史 gram 呈几何性增长，不可能被一一充分地建模。不同的“平滑”手段被提出来解决这一个问题，使得语言模型质量得到了显著改善。

最早在 20 世纪 90 年代 [25, 26, 27]，研究人员提出了一种使用神经网络 (Neural Network, NN) 对隐马尔科夫模型中的状态输出概率进行建模的方法。但是，由于当时缺乏计算资源和数据，该框架没有达到比 GMM-HMM 系统更好的性能。随着摩尔定律继续有效，今天的计算机计算能力已经从二十年前实现了巨大的飞跃。图形处理单元 (Graphics Processing Units, GPU) 使计算机能够更快地执行并行计算，从而可以训练更强大的模型。随着越来越先进的移动互联网和云计算，现在可以更轻松地收集足够的训练数据。这些因素都使得深度神经网络 (Deep Neural Network, DNN) 在今天可以被成功地训练。DNN-HMM 提出是对 GMM-HMM 框架的一次变革，令语音识别性能再次获得了巨大的提高，真正走出了实验室研究层次 [28, 29, 30, 31, 32, 33, 34, 35]。谷歌、微软、苹果等国际 IT 巨头近年都推出了以语音识别为核心技术的商业级产品。

字错误率 (Word Error Rate, WER) 是衡量语音识别系统质量的重要指标。给定正确注释和语音识别系统的解码结果，将单词错误率定义为两个单词序列之间的编辑距离，值越小越好。图 1-1 反映的是截止到 2009 年深度神经网络出来之前 NIST 赞助的各个语音识别任务词错误率变迁。其中横坐标是时间，纵坐标是词错误率，每一条折线代表着一项语音识别任务。

最早的任务是在干净的环境中的阅读语音的识别。随着 GMM-HMM 模型的改进与

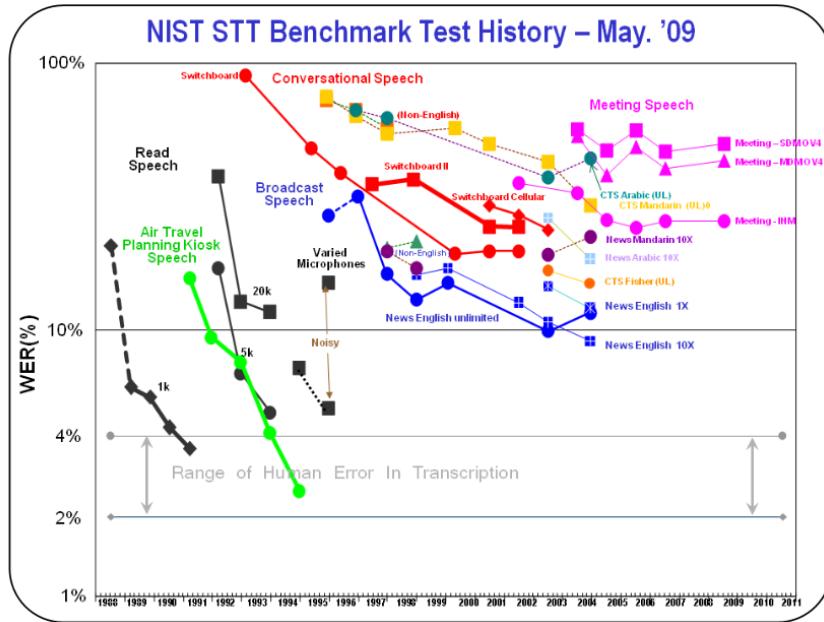


图 1-1 语音识别词错误率变迁图 (截止 2009 年)

Fig 1-1 History of WER on several tasks (until 2009)

人类的识别率相同，到了 20 世纪 90 年代，任务逐渐接近日益复杂的现实环境。仅使用 GMM-HMM 模型，单词错误仍然很高。其中一个标志性的任务是 Switchboard，它是一种电话语音识别。语音识别任务中最大的两个困难是：第一，语音信号是高度非线性信号；第二，声学环境（扬声器，噪声等）对语音信号有很大影响。从图中可以看出，任务场景越向右开，就越难。在 21 世纪初，学者们分别研究了序列鉴别性训练（序列建模）和自适应方法来解决这两个问题。可以看出，Switchboard 的字板错误率已大大降低。2010 年，微软学者提出使用深度神经网络对语音信号进行建模。神经网络的高非线性与语音信号非常吻合，并且 Switchboard 字的错误率进一步降低。由于 DNN-HMM 框架是 GMM-HMM 框架的一次革命，序列鉴别性训练和自适应技术是独立于框架的两个技术方向。因此，他们在新框架下有了新的发展空间。近年来，基于深度神经网络序列的鉴别性训练和自适应技术已成为新时代语音识别领域的核心研究内容。本文主要研究基于深度神经网络的序列建模技术。

另一方面，语音识别既是模式识别问题又是相应的推理搜索问题。前一个问题确定了语音识别系统可以实现的识别准确度的上限；在给定模型的情况下，后一问题研究如何使输入语音与模型匹配并推断最佳识别结果，其确定识别速度和实际可达到的识别准确度。近年来，DNN-HMM 框架为代表的深度学习模型被引入到语音识别声学和语言建模中以取代传统的分类器，但分类器之外的推理搜索问题和前文提到的序列建模没有

根本改变。基于加权有限状态机 (Weighted Finite State Transducer, WFST) 的推理搜索问题静态搜索空间构造技术 [36] 和帧同步维特比 (Viterbi) 网络搜索算法 [37] 是目前性能最好的解决方案。

### 1.1.2 语音识别架构

在迄今为止最为成功的基于统计的语音识别框架中，语音识别过程可以被抽象为如下数学公式：

$$\mathbf{w}^* = \arg \max_{\mathbf{w} \in \mathcal{H}} P(\mathbf{w} | \mathbf{O}) \quad (1-1)$$

即在所有可能的候选序列  $\mathcal{H}$  中寻找最大后验概率  $P(\mathbf{w} | \mathbf{O})$  对应的词序列  $\mathbf{w}^*$ 。其中  $\mathbf{w} = [w_1, \dots, w_n]^\top$  是词序列， $\mathbf{O} = [\mathbf{o}_1, \dots, \mathbf{o}_T]^\top$  是特征向量序列。

在 LVCSR 中，直接对后验概率  $P(\mathbf{w} | \mathbf{O})$  建模是比较困难的，这个问题可以通过贝叶斯公式转换成条件似然  $p(\mathbf{O} | \mathbf{w})$ ，先验  $P(\mathbf{w})$  和  $p(\mathbf{O})$ 。因为边缘分布  $p(\mathbf{O})$  在解码过程中与假设词无关，所以可以忽略掉。在剩下部分中， $p(\mathbf{O} | \mathbf{w})$  被称为声学模型， $P(\mathbf{w})$  被称为语言模型。声学模型用做建模子词单元生成特征序列的概率的，语言模型描述的是局部语法和整句的语言语义信息。

$$\begin{aligned} \mathbf{w}^* &= \arg \max_{\mathbf{w} \in \mathcal{H}} P(\mathbf{w} | \mathbf{O}) \\ &= \arg \max_{\mathbf{w} \in \mathcal{H}} \frac{p(\mathbf{O} | \mathbf{w}) P(\mathbf{w})}{p(\mathbf{O})} \\ &\propto \arg \max_{\mathbf{w} \in \mathcal{H}} p(\mathbf{O} | \mathbf{w}) P(\mathbf{w}) \end{aligned}$$

在关键词检测中，既可以采用上述的方式进行贝叶斯公式转换，此时将  $P(\mathbf{w})$  定义为各个关键词序列出现的先验概率；也可以不采用贝叶斯公式，直接对后验概率  $P(\mathbf{w} | \mathbf{O})$  进行建模。

图 1-2 是对当前流行语音识别系统的框架描述，它主要由四个部分组成，包括前端信号处理、声学模型、语言模型和解码器。

- 前端信号处理：原始模拟信号首先由输入设备转换为数字信号。前端信号处理部分负责从数字化语音中提取鲁棒声学特征信息，主要包括多麦克风阵列噪声降低和人耳听觉声学特性的提取。详细内容将在章节 2.1.1 中介绍。

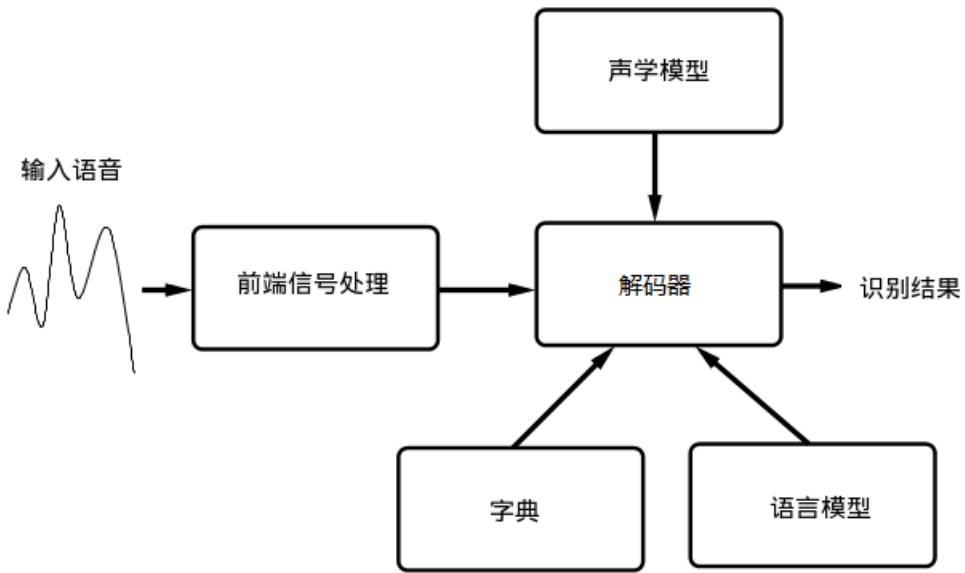


图 1-2 语音识别框架

Fig 1-2 Framework of an automatic speech recognition system

- 声学模型 (Acoustic Model, AM): 声学模型是语音识别系统中最重要的模型之一。声学模型的好坏可以直接决定了语音识别系统的性能，也是本论文研究重点之一。声学模型建模的是给定词序列生成出来的所观测到的特征向量序列条件概率  $p(\mathbf{O}|\mathbf{w})$ ，目前主流的语音识别系统通常是使用隐马尔科夫模型 (Hidden Markov Model, HMM) 来做为声学模型的。在 HMM 中，存在一个概率分布熵被称为状态输出概率，这个概率可以通过去使用混合高斯模型来得到建模，也可以通过深度神经网络来建模。使用前者语音识别系统可以被称为 GMM-HMM 系统，使用后者被称为混合系统 (DNN-HMM) 系统。具体内容将在章节 2.1.2.2 和章节 2.2.3 中详细介绍。
- 语言模型 (Language Model, LM): 过去了数十年，N 元组模型 (n-gram) 是最主要的使用语言模型 [38, 39, 40]。近几年，基于深度神经网络语言模型也逐渐开始得到发展并取得了巨大性能的提升 [41, 42]。
- 解码器及搜索 (Decoder): 解码器功能是对声学模型中计算出的声学特征上的概率和语言模型计算出的语言概率去进行组合来得到最大概率的词序列。目前主流的解码算法是去使用基于动态规划思想的维特比算法 (Viterbi Algorithm)，将在第3.2章节中详细介绍。

## 1.2 语音识别中的序列建模

语音识别是一种序列预测问题，在语音识别中一个独有并且有趣的自然现象是声学序列和语言学序列的长度可变性。序列模型正是被提出用来解决两个序列之间的关联性。依据序列的不同建模方式，目前有两种不同的序列鉴别性训练方法，一种针对生成式序列模型 (Generative Sequence Model, GSM) 比如上文介绍的 HMM 以及其相应的深度学习系统；另一种则针对鉴别式序列模型 (Discriminative Sequence Model, DSM)，比如连接时序分类模型 (Connectionist Temporal Classification, CTC) 或者 encoder-decoder 模型。在这些序列模型中，除了需要研究如何将各种基于深度学习的模型方法应用到逐帧分类当中，使用序列级的鉴别性训练准则来强化模型的序列分类能力，也被证明是取得业界最好的大词汇连续语音识别系统的关键。

针对 GSM，使用隐马尔科夫模型，在进行序列鉴别性训练之前，需要首先使用贝叶斯公式计算序列后验概率。根据贝叶斯公式，可以将序列后验概率展开为标注序列的声学模型概率和语言模型概率两项，以及分母项针对所有可能的序列的概率进行求和的边缘概率项，共三部分。由于语音识别的搜索空间巨大，一般研究中通常采用词图对边缘概率项的可能序列进行限制。而该部分词图的质量也是取得良好性能的关键之一。DSM 则是直接对序列后验概率进行估计的一类模型，如 CTC 是一个典型的例子 [43, 44]。序列鉴别性准则定义为所有可能的 CTC 标签序列的概率求和，作为最终的优化方向。一个额外的 blank 单元被引入到逐帧展开公式中以建模语音信号的混淆区段。

上述基于词图的序列建模方法存在几方面缺陷：词图构建引入了一个额外的步骤，使得词图更新频度和词图质量需要进行额外的权衡；逐句生成的词图导致序列建模过程中较难进行多句并行训练，而并行训练是目前最主流的加速模型训练的方式，训练速度的问题导致目前难以进行大规模的序列建模训练；该框架基于一个预训练的种子模型，因此种子模型的质量一方面关系到目前流行的深度学习模型的最终优化结果，另一方面也影响标签序列逐帧强制对齐的精度，从而影响最终序列建模训练的结果。

另一方面，序列建模方法通常只在训练 LVCSR 模型时进行使用，但针对其它非传统识别任务的序列建模研究并不充分，这包括：关键词检测任务和多说话人重叠语音信号识别任务等。这类任务仍然是序列预测问题，但是却没有合适的训练准则和相应的设计，来充分优化分类器的序列建模能力。因此这些领域近几年的进展主要仍然来自于深度学习本身所带来的更好的逐帧分类能力。

因此，尽管基于深度学习模型的序列建模研究已经取得了一定进展，但是更好、更通用的序列建模框架，以及针对更多语音识别任务的拓展，仍是亟待研究的问题。

### 1.3 语音识别中的解码搜索

语音识别技术虽然相比多年以前已经有了长足的进步，但是在实际应用中还有很多困难需要处理。其中一个最主要难题就是语音识别推理搜索问题。

语音识别既是模式识别问题又是相应的推理搜索问题。前一个问题在数学上表示和描述了各种语音和语言现象，并且基于统计学习模式识别框架执行建模，该框架确定了语音识别系统可以实现的识别准确度的上限。在给定模型的情况下，后一问题研究如何使输入语音与模型匹配并推断最佳识别结果，其确定识别速度和实际可达到的识别准确度。在语音识别的推理搜索阶段，解码器功能组合声学模型以计算声学特征概率和语言模型计算的语言概率以获得最大概率词序列。在语音识别推理搜索阶段，解码器是语音识别系统的核心和灵魂，所有信息都收集在这里。它将来自不同来源，不同层次和不同性质的知识和信息联系起来，以便它们相互补充并获得正确的语音识别结果。因此，如何有机地整合各种不同的信息是解码网络和解码算法设计中必须仔细研究和解决的问题。从解码器功能的角度来说，它不仅是语音识别研究中各种理论，模型和算法的验证。正确性和准确性的基本实验平台也是构建实际系统的基础所在。因此，在解码器的设计中必须平衡研究的便利性和工程的实际应用。

近年来，深度学习模型被引入到语音识别声学和语言建模中以取代传统的分类器，显着提高了模式识别问题的准确性。在基于深度学习语音识别中，推理搜索问题没有根本改变，因为深度学习只是取代了分类器。基于加权有限状态机的推理搜索问题静态搜索空间构造技术 [36] 和帧同步维特比（Viterbi）网络搜索算法 [37] 是目前性能最好解决方案。**WFST** 是一组加权有限状态机表示的状态图跳转信息，语音信号的识别各模型组件可以分别构建成一组 WFST。当各个知识源 WFST 组件被构建完毕以后，可以利用 WFST 合成算法和优化算法将各组件最后进行合并和最终优化。最终生成的 WFST 包含了所有的知识源，维特比搜索就在它上面逐帧进行。该方法虽然比传统动态解码技术更高效，但仍存在一系列显著缺陷：

1. 该种方案基于传统的混合高斯-隐马尔科夫模型（GMM-HMM）的声学模型和 N 元文法（N-gram）的语言模型而提出，针对目前性能最好的基于深度学习的声学模型和语言模型的研究并不充足，如何将新型的声学和语言模型引入该框架；如何充分发挥模型性能的同时改善推理速度；如何基于多知识源给出可靠的推理置信度算法，都是有待解决的问题。
2. 该种方案基于对语音识别中各知识源（声学、语言、语义等）进行搜索空间的预先构建及整体优化，导致最终得到的搜索空间巨大，包括离线构建、在线使用、动态修改等各环节算法的计算量和内存消耗都非常大，是阻碍语音识别应用场景扩

展的一个重要原因。旨在解决该问题，针对新型声学和语音模型的搜索空间整体优化研究尚不充分，而基于推理中间状态对搜索空间进行动态优化的研究几乎处于空白。

3. 目前，语音识别系统基于多知识源建模结果，并对输入音频进行推理搜索。建模和推理搜索过程非常复杂，知识来源的划分依赖于强大的先验知识。大规模或非标记语音数据收集以及基于并行的深度学习技术使得构建直接模拟语音数据和文本序列的端到端模型及其相应的识别和推理搜索算法成为可能。当前的解码框架没有这种设计。

因此，尽管基于深度学习模型，加权有限状态机和基于帧的维特比网络搜索算法的深度学习已经发展到基本可用水平，但是精度仍然不能满足人类之间的正常交流要求。速度限制也使得语音识别无法在低成本和低功耗的解决方案上工作，这些解决方案一起阻碍了语音识别技术的大规模商业应用。

## 1.4 论文主要内容、创新点及组织结构

本论文围绕深度学习模型的序列建模和解码，针对上面探讨的诸多研究缺陷，展开了一系列探索和研究，主要涉及了非传统识别任务的序列建模，基于 GPU 并行计算的搜索速度优化，基于标签同步解码搜索空间优化，标签同步解码的扩展应用等内容。

1. 本论文针对关键词检测和多说话人重叠语音信号识别这两类非传统语音识别任务提出了序列建模方案。

序列建模方法通常只在训练 LVCSR 模型时进行使用，但针对其它非传统识别任务的序列建模研究并不充分，这包括：关键词检测任务和多说话人重叠语音信号识别任务等。这类任务仍然是序列预测问题，但是却没有合适的训练准则和相应的设计，来充分优化分类器的序列建模能力。为了将序列鉴别性训练引入关键词检测任务，核心难题是设计相应的竞争可能性建模方法。本论文提出采用无词图鉴别性训练框架来解决这一问题：隐性使用音素或半词单元的语言模型来建模。另一方面，单通道多说话人混叠语音识别也属于序列级问题，因此序列鉴别性准则将有助于这样序列分类问题。我们提出了一种传统鉴别性训练技术变种，它在进行鉴别性训练的同时，也抑制输出通道上说话人跟踪错误。通过联合优化，迁移学习，序列鉴别性训练等方式，我们改善了原来语音分离、信号增强和语音识别的联合训练系统。

2. 本论文提出并行的解码搜索算法并在 GPU 上实现开源该套算法, 该框架可以显著加速现有推理搜索算法。

语音识别解码网络中多数 WFST 边之间并不直接相关, 具有并行处理的可能性; 基于 GPU 的并行加速也已成功应用于声学分数的计算。但是基于 GPU 并行计算的 WFST 解码不容易实现, 且现有研究存在诸多缺陷。针对设计中的几大难点, 本论文提出了如下解决方案: 本论文将维特比算法中的令牌合并操作实现为一个 GPU 并行计算中的原子操作, 以便减少维特比束剪枝算法中同步消耗; 提出了动态负载均衡的方式以更高效地进行并行计算, 提高其多线程之间的利用率; 重新设计了基于 GPU 并行计算的精确的词图生成和剪枝算法, 以便充分利用 GPU 的性能特点。在 Switchboard 上实验表明, 本论文所提出的方法在取得完全一致的 1-best 和词图质量情况下, 可以得到 3-15 倍的加速。除此之外, 如果再进行多句子的并行处理, 最终的加速比将达到 46 倍。所提出的离线解码器对语言模型和声学模型没有特别的限制, 并且可以工作在各种架构的 GPU 上。

3. 本论文基于端到端建模, 系统地提出了标签同步算法, 其通过一系列方法使得搜索解码过程从逐帧同步变为标签同步, 从而加速解码。

当前主流的推理搜索方法是帧层面的维特比束搜索算法, 该算法复杂度很高, 限制了语音识别的广泛应用。本论文提出将特征层面的搜索过程改变为标签层面, 使得解码速率等于标注速率, 从而小于特征速率。具体来说, 在标签搜索阶段, 对帧层面声学模型的输出增加一步后处理过程, 得到对每个输出标签概率计算的近似值, 再进行标签搜索。与传统方法相比, 该方法的优势是搜索空间更小, 且搜索过程被大大加速。本论文提出的一系列通用方法在隐马尔科夫模型和连接时序分类模型上得到了验证, 取得大幅度语音识别解码速度改善。

4. 标签同步算法同时还能产生高质量的音素词图, 称为 CTC 音素词图。基于 CTC 音素词图, 本论文进一步探讨标签同步解码算法的一些扩展应用。这包括关键词检测, 多识别任务统一置信度框架, 以及端到端语音识别。

我们在关键词检测中, 基于前述 LSD 算法得到的音素词图, 提出了一套基于编辑距离的后处理算法以引入音素混淆性, 使系统更加鲁棒。在前面提出的标签同步解码算法的基础上, 本论文进一步提出了两种置信度生成算法。更细致的研究显示这种基于 CTC 的音素词图是得到更好性能的关键所在。本论文还提出了辅助归一化搜索空间的概念, 并尝试使用这样的搜索空间来建模所有 ASR 应用领域的置信度。而针对这样做在低功耗设备上带来的挑战, 本论文采用基于 CTC 的标签同步解码来进行处理, 由此带来了很大的效率改善。同时本论文还研究了将标签同

步算法应用于直接建模输出序列形态学组合的端到端模型的方案。本论文使用模块化训练的思想来改善端到端模型建模，使其更易于使用外在知识源来训练每一个端到端模型的子模块。值得注意的是，模型最后需要进行联合优化，因此最终在推理搜索阶段，模型仍然工作在端到端模式下。在实验部分，本论文提出系统一方面取得大幅度语音识别解码速度改善，另一方面在端到端建模上取得了更快和更好的模型收敛和模型准确度。

本文剩余章节安排如下：首先，第二章中将介绍基于深度神经网络的自动语音识别，特别是深度序列建模，端到端建模；随后，第三章中会介绍语音识别的解码搜索问题；继而，第四章系统地提出一些针对非传统语音识别任务的序列建模方法，使之更加通用；第五章将提出并行解码搜索算法并在 GPU 上实现开源该套算法；第六章基于端到端建模，系统地提出了标签同步算法；第七章将标签同步算法进一步应用到更多语音识别应用中；最后，第八章给出本文总结以及对今后可能有用研究思路和展望。

## 第二章 自动语音识别及深度序列模型

自动语音识别技术 (Automatic Speech Recognition) 是一种将人的语音转换为文本的技术。本章中我们会最先介绍自动语音识别的框架，并概述每个子模块的基本设计。近年来，深度学习模型被引入到语音识别的声学和语言建模当中替代传统分类器 (NN-HMM)，显著改善了模式识别问题的精度，从而改善了语音识别的准确度 [29, 30]。另一方面，语音识别本质上是一个序列标注问题。NN-HMM 模型的序列关联性依赖于隐马尔科夫模型进行建模，而隐马尔科夫模型的状态转移建模能力有限。除此之外，NN-HMM 模型对深度学习模型和隐马尔科夫模型进行分别训练，缺少了整体优化也将影响最终的模式识别性能。因此我们着重介绍了深度学习背景下的序列建模研究。在最后一部分，得益于分类能力有所提升，深度学习模型时序建模能力被用于搭建端到端语音识别系统。其得益于更好的序列分类能力和更简单推理搜索框架，已经成为当前研究的热点。

### 2.1 自动语音识别框架

这一章我们将主要介绍在大词汇连续语音识别中一些基本内容，图 1-2 中所描绘的几个部分都会在本章描述。这主要包括：特征提取前端、子词单元的挑选、隐马尔科夫、解码搜索、语言模型等内容。

#### 2.1.1 特征提取

语音信号原始形态是一种连续的波形语音，为了能进行更有效的识别，通常我们会先将连续的波形转换为一个离散的实数序列向量  $\mathbf{O} = [\mathbf{o}_1, \dots, \mathbf{o}_T]^T$ 。每个向量是压缩语音变化的表示。这些矢量也称为特征矢量或观察特征矢量。语音信号是准静态信号，因此我们首先需要将其切割成多个重叠的离散段，通常是以 10 毫秒的间隔向后滑动 25 毫秒长的窗口。通过此方法提取的段称为帧。汉明窗口或汉宁窗口通常用于平滑，以减少边界效应，然后使用快速傅立叶变换将其从时域特征变换到频域特征。在得到频域上的复数特征后，通过利用不同的后处理方法可以得到不同特征：典型的有感知线性预测系数 (Perceptual Linear Prediction, PLP) [45] 和梅尔倒谱系数 (Mel-Frequency Cepstral Coefficients, MFCC) [46]。近来，由于深度神经网络拥有更强大的建模能力，研究者发现保留梅尔滤波器输出中维度之间的相关性的滤波器组特征 (Filter Bank Feature, FBANK) [47] 更适合于深度神经网络利用，接下来将对这三种特征进行详细介绍：

- 滤波器组特征

1. 在获得频谱的复杂特征之后，通常丢弃相位信息，并且仅留下复杂特征的幅度部分。然后通过梅尔频率缩放公式调整频率轴。最后，我们可以得到一个缩放的幅度频域特征：

$$\text{Mel}(f) = 2595 \log_{10}\left(1 + \frac{f}{500}\right) \quad (2-1)$$

2. 接下来，不同的滤波器包含不同的滤波器增益，并且将使用一组三角滤波器来对该频域特征进行下采样。最后，一个滤波器的输出是幅度特性之和乘以滤波器中相应频率增益的自然对数。通常，36 组滤波器用于 8K 采样率，40 组用于 16K 语音。。

- 梅尔倒谱系数

梅尔倒谱系数特征在 FBANK 特征基础上去进一步利用离散余弦变换来计算倒谱系数以便减少滤波器的组之间相关性。通常是利用 12 个的倒谱系数加上归一化的功率自然对数组成一个 13 维的向量特征。

- 感知线性预测系数

1. 感知线性预测系数是另外一个倒谱的特征，它利用 Bark 公式来去缩放频率的轴：

$$\text{Bark}(f) = 6 \log \left( \left( \frac{f}{600} + 1 \right)^{0.5} + \frac{f}{600} \right) \quad (2-2)$$

2. 接着它利用功率谱（幅度的平方）来提取感知线性的识别特征，之后该功率谱会与一个临界的频带滤波器进行卷积操作并且通过等响度的曲线进行预加重操作。  
3. 最后通过利用线性预测分析来获得一种倒谱的系数。

在提取了原始声学的特征之后，通常会利用一些后处理方法：

- 动态特征 [48]: 一阶动态特征的计算公式如下

$$\Delta_{\mathbf{o}_t} = \frac{\sum_{k=1}^K k(\mathbf{o}_{t+k} - \mathbf{o}_{t-k})}{2 \sum_{k=1}^K k^2} \quad (2-3)$$

其中  $K$  是动态特征计算窗的大小，通常设置为 2。二阶动态特征是最常用的，其计算方法和一阶一致，只不过将  $\mathbf{o}_t$  替换为  $\Delta_{\mathbf{o}_t}$ 。在利用了动态特征后，特征的不同维度之间产生了相关性，这与后面一些声学模型建模方法中作出特征各维度之间独立性的假设产生了冲突。因此，为消除特征各维度之间不同的相关性，通常会利用线性投影的方法，如异方差线性判别分析 (Heteroscedastic Linear Discriminant Analysis, HLDA) [49] 等。

- 特征正则化: 特征正则化目标是消除声学特征中的非语音信号的变化, 同时它也能将特征的值域范围进行归一化, 这一操作对于深度神经网络来说特别重要。传统的正则化方法包括倒谱均值归一化 (Cepstral mean normalisation, CMN) [50], 倒谱方差归一化 (Cepstral variance normalisation, CVN)[51] 以及声道长度归一化 (Vocal tract length normalisation, VTLN) [13]。其中倒谱均值归一化 (CMN) 将输入特征向量的每个维度的均值归一化为 0, 倒谱方差归一化 (CVN) 将输入特征向量的每个维度的方差归一化为 1。归一化可以运用在不同层面-包括说话人层面及句子层面。声道长度归一化 (VTLN) 被用来减少声学特征中的说话人变化, 它的原理是将来自于同一个说话人特征频率轴进行同样大小缩放。

## 2.1.2 声学模型

声学模型的作用是计算一个候选词序列  $\mathbf{w}$  生成出观测到的特征向量序列  $\mathbf{O}$  概率, 即  $p(\mathbf{O}|\mathbf{w})$ 。它从概率论角度提供了给定词标注之后语音信号的信号生成的过程。在传统 GMM-HMM 声学模型中, HMM 建模了语音信号的序列性, GMM 建模了特征向量生成的概率。在最新的基于深度神经网络的声学模型中, 深度神经网络被用来计算特征向量生成概率。GMM-HMM 将在章节2.1.2.2中详细介绍, DNN-HMM 将在章节2.2.3中详细介绍。声学模型是语音信号的识别系统中最核心部件之一, 也是本论文研究重点。

### 2.1.2.1 声学单元和参数绑定

当进行小词汇语音信号的识别时, 如识别数字时通常可以用隐马尔科夫模型直接建模独立的单词。然而当词汇数量从中等词汇上涨到大词汇 ( $>10000$ ) 时, 利用隐马尔科夫模型来建模每一个词汇变成了不可能事情。一个广泛利用的解决该问题方法为建模子词单元。

音素是一个被广泛利用的子词单元, 它是语音信号的中最小声学元素。利用音素的好处是存在将每个词转换为音素标准, 这样每个词能很容易的被分解成各个音素。在音素上建模的模型被称作音素模型, 音素数目一般远远小于待识别词的个数。在当前最先进的大词汇连续语音信号的识别系统里, 通常利用 46 个音素。

利用音素能更容易的获得足够数据以训练出更具有鲁棒性的模型参数。需要注意是当利用音素来建模时, 需要提供一个字典用于将单词序列映射成子词单元序列, 然后才能在子词单元的层面进行识别和运算。在识别最后, 还需要将子词单元序列转换回单词序列。

目前有两种被广泛利用音素集合。一种是单音素, 也称为上下文无关音素; 另一种则是上下文相关音素。由于协同发音现象的影响, 当前音素发音和前一个、后一个音素

之间具有很强的相关性，所以在许多识别任务里，仅仅利用单音素效果不是很好。为了建模协同发音现象，目前最先进语音信号的识别系统利用的是上下文相关音素，其中三音速利用最为广泛，它将当前音素的前一个和后一个音素同时建模。

比如，以 one 来说，它对应三音素序列即扩展为  $\text{one} = \{\text{sil-w-ah}, \text{w-ah-n}, \text{ah-n-sil}\}$ 。虽然利用更多的上下文信息可以建立更复杂的音素模型，如五音素 [52](quin-phones)，但三音素依然是目前利用最广泛音素模型。根据是否考虑单词间的边界，三音素模型可以继续细分为跨单词三音素和单词内三音素。跨单词三音素允许三音素的扩展跨越单词：在两个单词边界，当前音素的前一个和后一个音素分别是上一个单词的最后一个音素和下一个单词第一个音素。词内三音素则相反，音素只能在单词的内部进行扩展。因此，在每个单词开头和结尾，是一个双音素。跨单词三音素在大规模词汇连续语音信号的识别系统里效果更好 [53]。

利用三音素后，声学单元个数将变得巨大。当利用 46 个单音素时，所有可能的三音素个数达到了  $46^3 = 97336$ ，这一数字甚至超过了词表大小，不可能有充足数据能训练如此大的模型。目前一个通用解决该问题的方法是利用参数共享 [54, 55]。它基本思想是将一些参数绑定在一起。参数共享可以存在于各个层面，包括音素，状态，高斯等。由于利用最广泛的是在状态共享层面，因而也被称为状态聚类。在利用状态聚类时，来自同一组的状态将共享状态输出分布，每一个实际状态输出分布被称为一个语素(senone)，如图2-1所示。

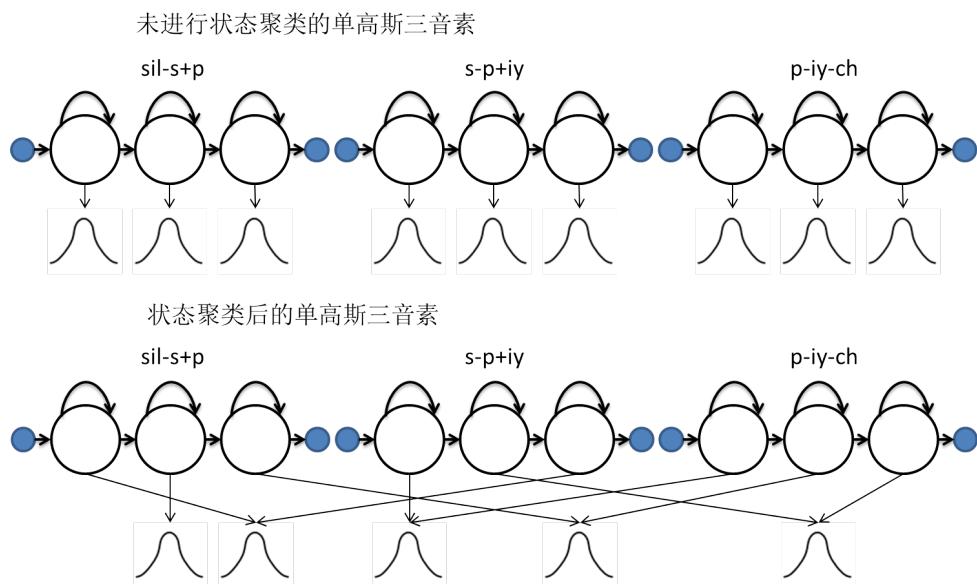


图 2-1 单高斯三音素的状态聚类

Fig 2-1 State clustering for single gaussian triphones

目前利用最广泛的聚类方法是基于数据驱动自底向上聚类。对训练集中存在的每一对语素之间计算一个距离，距离在某个阈值以内的语素将会被聚集在一起。数据驱动的主要问题在于当没有足够多训练数据时，这一方法将不再可靠，更严重的是，训练数据中不存在信息将无法被捕捉到。

另外一个更好的方法是利用音标决策树的方法来进行聚类。音标决策树是一棵包含了一系列关于每个音素左右上下文问题的二叉树，因为问题答案只有对和错。聚类自上而下开始，所有的状态都从根节点出发，通过回答上下文问题来分割左右儿子。直至处于当前节点的训练数据状态数目小于一个阈值，分割过程停止，此时叶子节点即为一个语素。决策树中的关键难题在于提问的顺序，目前最常用方法是在每次分裂时挑选能在分裂后似然增加的最大的那个问题。虽然这样决策树可能陷入局部最优，但它确实有效处理了对于不可见的三音素分类问题。

### 2.1.2.2 隐马尔科夫模型 (HMM)

隐马尔科夫模型是一个统计学中生成模型，在语音信号的识别领域获得了重大成功。在隐马尔科夫模型中，一个特定声学的单元，如一个单词或一个音素将被建模为一个有限状态机，而我们所观测到特征序列则是由该音频所对应的词序列连接成的有限状态机生成的。在每个时间单元，状态将以一个给定的概率分布发生转变：跳转到下一个状态或保持在当前状态。变换完成后，将由另一个概率函数生成一个特征向量。一个拥有三个输出状态的从左到右隐马尔科夫模型如图2-2所示，其中状态 1 和 5 是进入状态和退出状态，它们并不输出观测特征向量。这一结构是语音信号的识别中最常用的序列模型，我们将在第2.3.1章节详细介绍这种结构，并将其与其他序列模型进行比较。

令  $\mathbf{O} = [\mathbf{o}_1, \dots, \mathbf{o}_T]^\top$  为由一个声学单元生成的特征向量序列，其中  $\mathbf{o}_t$  是一个第  $t$  时刻的  $D$  维的语音特征向量， $T$  是语音信号的序列的总帧数。声学特征序列的生成过程如下面所示：

1. 在第 0 时刻从状态 1 开始
2. 在时刻  $t (0 \leq t \leq T - 1)$ 。假设当前处于状态  $i$ ，以概率  $a_{i,i+1}$  跳转到状态  $i+1$  或者以概率  $a_{i,i}$  停留在当前状态。
3. 假设跳转完后处于状态  $j$ ，若此时处在输出状态，则以  $b_j(\mathbf{o}_t)$  的概率输出声学向量  $\mathbf{o}_t$ 。
4. 重复 2 直到达状态 5

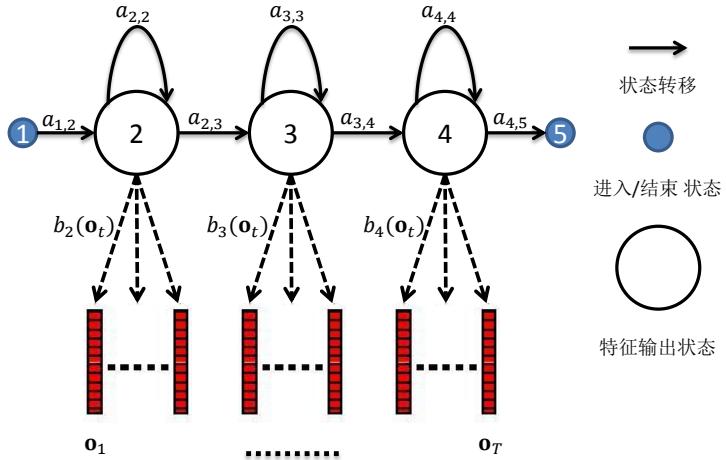


图 2-2 隐马尔科夫模型  
Fig 2-2 Hidden Markov Model

这样我们可以利用一个状态序列来描述语音信号的特征的输出过程  $\mathbf{s} = [s_1, \dots, s_T]^\top$ 。而现实中我们只能观察到由状态输出语音信号的特征序列，状态序列  $\mathbf{s}$  是隐藏的，这也是该模型被称为隐马尔科夫模型原因。一个隐马尔科夫模型通常包含如下参数：

- $\pi$  初始状态分布：

令  $s_t$  表示在时刻  $t$  时所处状态，那么  $\pi_i = P(s_0 = i)$ ,  $\sum_{i=1}^N \pi_i = 1$ ,  $\pi_i \geq 0$ ，其中  $N$  是总状态数。在拥有进入状态隐马尔科夫模型中，通常  $\pi_1 = 1$ 。

- 状态转移概率矩阵  $\mathbf{A}$ ：

$a_{i,j} = P(s_{t+1} = j | s_t = i)$ ，在最常用 5 状态隐马尔科夫模型中，通常只有  $a_{i,i}$   $a_{i,i+1}$  不为 0

- 状态输出概率分布  $\mathbf{B}$ ：

每个特征输出状态  $i$  都有一个概率分布来输出一帧声学特征， $b_i(\mathbf{o}_t) = p(\mathbf{o}_t | s_t = i)$

接下来我们将讨论如何计算隐马尔科夫模型的似然度，状态发射概率密度函数的定义将留到下一章节的 GMM 中或者第2.2.3章节中的 DNN 中进行具体讨论。似然计算目标是给定一个语音信号的特征向量序列和一个隐马尔科夫模型，计算该模型生成给定的语音信号的特征向量序列概率，即计算  $p(\mathbf{O}|\mathbf{w}, \mathcal{M})$ ，其中  $\mathbf{O}$  为观测到的语音信号的特征向量序列， $\mathbf{w}$  为对应文本标注， $\mathcal{M} = \{\pi, \mathbf{A}, \mathbf{B}\}$  是所有模型参数。由于状态序列  $\mathbf{s}$  是隐

藏，因而需要枚举所有可能的状态序列并求其期望，即：

$$p(\mathbf{O}|\mathbf{w}, \mathcal{M}) = \sum_s p(\mathbf{O}, \mathbf{s}|\mathbf{w}, \mathcal{M}) \quad (2-4)$$

$$= \sum_{\mathbf{s}} P(\mathbf{s}|\mathbf{w}, \mathcal{M}) p(\mathbf{O}|\mathbf{s}, \mathcal{M}) \quad (2-5)$$

$$= \sum_{\mathbf{s}} a_{s_0, s_1} \prod_{t=1}^T a_{s_{t-1}, s_t} b_{s_t}(\mathbf{o}_t) \quad (2-6)$$

以上只考虑了单个隐马尔科夫模型似然计算。对于连续语音信号的识别或利用子单词的声学单元，语音信号的序列将对应一个模型序列。各个单词或者子单词单元之间准确时间边界是未知的，而解决方法则是扩展单隐马尔科夫模型，将一部分单独隐马尔科夫模型连接起来组成组合隐马尔科夫模型。

似然计算是利用和训练隐马尔科夫模型时需要考虑的最核心问题，直接枚举所有状态序列复杂度无疑是很高的。这一概率可以利用前向-后向算法快速计算，该算法也被称作鲍姆威尔士算法 [56](Baum-Welsh algorithm)。前向-后向算法通过利用动态规划思想，只需要  $O(N^2T)$  时间复杂度即可以计算出该概率，其中  $N$  是总状态数， $T$  是总帧数。

### 2.1.2.3 混合高斯模型 (GMM)

在传统的 GMM-HMM 中，状态输出概率  $b_i(\mathbf{o}_t)$  通常由一个混合高斯模型来建模。概率计算公式如下：

$$b_i(\mathbf{o}_t) = \sum_{m=1}^{M_i} c_i^m \mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_i^m, \boldsymbol{\Sigma}_i^m) \quad (2-7)$$

其中  $M_i$  是属于第  $i$  个状态混合高斯模型含有的高斯成分个数， $c_i^m$  是混合权重，满足  $c_i^m \geq 0, \sum_{m=1}^{M_i} c_i^m = 1$ 。 $\mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_i^m, \boldsymbol{\Sigma}_i^m)$  是均值为  $\boldsymbol{\mu}_i^m$ ，协方差矩阵为  $\boldsymbol{\Sigma}_i^m$  多变量高斯分布。

$$\mathcal{N}(\mathbf{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{o}-\boldsymbol{\mu})} \quad (2-8)$$

### 2.1.2.4 最大似然估计

训练 GMM-HMM 通常采用最大似然估计 (Maximal Likelihood Estimation, MLE) 准则。令  $\mathcal{M} = \{\{a_{i,j}, 1 \leq i, j \leq N\}, \{c^m, \boldsymbol{\mu}^m, \boldsymbol{\Sigma}^m, 1 \leq m \leq M\}\}$  为 GMM-HMM 中所有参数。其中  $N, M$  分别为总状态数和总高斯成分数。优化准则即为：

$$\hat{\mathcal{M}}_{MLE} = \arg \max_{\mathcal{M}} \log p(\mathbf{O}|\mathbf{w}, \mathcal{M}) \quad (2-9)$$

由于存在隐藏变量，直接优化上述公式是很困难的，利用最大期望算法 (Expectation-Maximization, EM) [57] 可对此进行优化。EM 算法被广泛运用于含有隐变量统计学模型中，它基本思想是引入一个辅助函数作为 log 似然下界，通过不断迭代优化辅助函数来优化 log 似然函数。对于隐马尔科夫模型而言，辅助函数定义为：

$$\mathcal{Q}_{\text{MLE}}(\mathcal{M}_{k+1}; \hat{\mathcal{M}}_k) = \sum_{\mathbf{s}} P(\mathbf{s}|\mathbf{O}, \mathbf{w}, \hat{\mathcal{M}}_k) \log p(\mathbf{O}, \mathbf{s}|\mathbf{w}, \mathcal{M}_{k+1}) \quad (2-10)$$

$$= \sum_{t,i} \gamma_i(t) \log b_i(\mathbf{o}_t) + \sum_{t,i,j} \xi_{ij} \log a_{i,j} \quad (2-11)$$

其中  $\hat{\mathcal{M}}_k$  是第  $k$  轮迭代计算出最优参数，

$$\gamma_i(t) = P(s_t = i | \mathbf{O}, \mathbf{w}, \hat{\mathcal{M}}_k) \quad (2-12)$$

$$\xi_{ij}(t) = P(s_{t-1} = i, s_t = j | \mathbf{O}, \mathbf{w}, \hat{\mathcal{M}}_k) \quad (2-13)$$

EM 算法是一个迭代的优化过程，其优化步骤如下：

1. 初始化模型  $\hat{\mathcal{M}}_0$
2. 设当前迭代到第  $k$  轮，已经训练好模型参数  $\hat{\mathcal{M}}_k$
3. 利用  $\hat{\mathcal{M}}_k$  估计后验概率  $\gamma_i(t), \xi_{ij}(t)$ 。这两个概率可以由上一节提到的前向后向算法计算  $\alpha, \beta$  快速获得：

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{p(\mathbf{O}|\mathbf{w}, \hat{\mathcal{M}}_k)} \quad (2-14)$$

$$\xi_{ij}(t) = \frac{\alpha_i(t-1)a_{i,j}b_j(\mathbf{o}_t)\beta_j(t)}{p(\mathbf{O}|\mathbf{w}, \hat{\mathcal{M}}_k)} \quad (2-15)$$

4. 利用最大似然估计  $\hat{\mathcal{M}}_{k+1}$

$$\hat{\mathcal{M}}_{k+1} = \arg \max_{\mathcal{M}} \sum_{t,i} \gamma_i(t) \log b_i(\mathbf{o}_t) + \sum_{t,i,j} \xi_{ij} \log a_{i,j} \quad (2-16)$$

5. 转移概率更新为：

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^T \xi_{i,j}(t)}{\sum_{t=0}^{T+1} \gamma_i(t)} \quad (2-17)$$

6. 当利用 GMM 作为状态输出概率时，高斯成分的索引可以被视为一个特殊隐子状态，转移概率是各成分的权重乘以状态转移概率。因此可以求得各个高斯成分后验占用率：

$$\gamma_j^m(t) = \frac{\sum_{i=2}^{N-1} \alpha_i(t-1) a_{i,j} c_j^m \mathcal{N}(\mathbf{o}; \boldsymbol{\mu}_j^m, \boldsymbol{\Sigma}_j^m) \beta_j(t)}{p(\mathbf{O}|\mathbf{w}, \hat{\mathcal{M}}_k)} \quad (2-18)$$

这里  $\gamma_j^m(t)$  表示状态  $j$  的第  $m$  个高斯成分在第  $t$  包含后验占有量。

### 7. GMM 的参数更新为:

$$\hat{c}_j^m = \frac{\sum_{t=1}^T \gamma_j^m(t)}{\sum_{m,t} \gamma_j^m(t)} \quad (2-19)$$

$$\hat{\mu}_j^m = \frac{\sum_{t=1}^T \gamma_j^m(t) \mathbf{o}_t}{\sum_{t=1}^T \gamma_j^m(t)} \quad (2-20)$$

$$\hat{\Sigma}_j^m = \text{diag} \left( \frac{\sum_{t=1}^T \gamma_j^m(t) (\mathbf{o}_t - \hat{\mu}_j^m)(\mathbf{o}_t - \hat{\mu}_j^m)^\top}{\sum_{t=1}^T \gamma_j^m(t)} \right) \quad (2-21)$$

在公式中，我们只估计了协方差矩阵对角元素。由于在大词汇连续语音信号的识别任务中通常需要利用大量高斯成分，在此基础上若利用满秩矩阵将对计算和存储资源需求巨大，因而对于每个高斯成分，通常只利用对角矩阵。

### 8. 重复 2 直到收敛

虽然利用最大似然估计 GMM-HMM 已获得了巨大成功，然而只有在拥有充足数据量和正确的模型假设前提下它才是合适优化准则。现实中，由于在 HMM 模型中存在马尔科夫和条件独立性两个假设，并不符合真正语音信号的生成过程，因此利用最大似然估计来最优化 HMM 将无法估计出最合适的参数。其中一个解决方案就是利用序列鉴别性训练准则 [17, 18, 19, 20, 21, 22, 23]

#### 2.1.2.5 序列鉴别性训练

在基于最大似然估计中，优化目标是给定标注生成语音信号的特征向量序列的似然。鉴别性训练与之不同处在于：鉴别性训练优化目标为最大化给定语音信号的特征向量序列所对应文本标注的后验概率，即最大化  $P(\mathbf{w}_{\text{ref}}|\mathbf{O})$ 。这样相当于直接将语音信号的识别评判准则引入优化目标之中。目前最先进的语音信号的识别系统中都利用了序列鉴别性准则。这章将简单地介绍其中两种：最大互信息 (Maximum Mutual Information, MMI) 和最小贝叶斯风险 (Minimum Bayes' Risk, MBR)

- 最大互信息

最大互信息准则在后验概率  $P(\mathbf{w}_{\text{ref}}|\mathbf{O})$  基础上增加一个经验缩放  $\kappa^1$ ，它的优化目标如下：

$$\mathcal{F}_{\text{MMI}}(\mathcal{M}) = \frac{p^\kappa(\mathbf{O}|\mathbf{w}_{\text{ref}}, \mathcal{M})P(\mathbf{w}_{\text{ref}})}{\sum_{\mathbf{w}} p^\kappa(\mathbf{O}|\mathbf{w}, \mathcal{M})P(\mathbf{w})} \quad (2-22)$$

---

<sup>1</sup>它的作用是为了使不太可能假设对准则有所贡献，并使准则更加平滑可区分，通常等于识别中利用的语言模型缩放系数倒数

这里  $\mathbf{w}_{\text{ref}}$  是语音信号的特征向量序列对应标注， $\mathbf{w}$  是所有可能的标注序列，包括正确标注和错误标注。虽然理论上  $\mathbf{w}$  应该包括所有可能词序列，实际上通常只考虑最具有混淆性标注。它通常由在训练数据上解码生成 N-Best 候选列表或者词图构成。我们将在第3.3.2.7章节详细介绍词图，它是一种对搜索空间的紧致图表示方法。

- 最小贝叶斯风险

最小贝叶斯风险准则目标在最小化期望损失，即

$$\mathcal{F}_{\text{MBR}}(\mathcal{M}) = \sum_{\mathbf{w}} P(\mathbf{w}|\mathbf{O}; \mathcal{M}) L(\mathbf{w}, \mathbf{w}_{\text{ref}}) \quad (2-23)$$

这里  $L(\mathbf{w}, \mathbf{w}_{\text{ref}})$  是标注与候选的标注之间损失函数，通常包含句子层，单词层和音素层。

- 句子层：这一准则希望最小化句子层的错误

$$L(\mathbf{w}, \mathbf{w}_{\text{ref}}) = \begin{cases} 1 & \mathbf{w} \neq \mathbf{w}_{\text{ref}} \\ 0 & \mathbf{w} = \mathbf{w}_{\text{ref}} \end{cases} \quad (2-24)$$

- 损失函数也可以定义在词层面(最小词错误)和音素层面(最小音素错误)。比如在最小词错误中， $L(\mathbf{w}, \mathbf{w}_{\text{ref}})$  为两个词序列之间编辑距离，即词错误数。最小音素错误在最先进语音信号的识别系统中利用非常广泛 [22]

在定义了各种鉴别性训练准则后，如何根据这些准则对模型参数进行优化就成为了另一个重要的问题。一个好的模型参数优化方法必须能够有效的优化上述准则、并可以高效率的实现，还要拥有良好的收敛性能。遗憾的是，到目前为止，对区分性训练准则而言还没有一种算法能够保证其收敛性。这样的局面与最大似然准则下的情形不同。因此，研究者通常致力于寻找一些在经验上有比较良好优化能力、又能较快收敛性的算法，以便对鉴别性训练准则进行模型参数优化。常用的参数优化方法主要分为两大类，一类是基于梯度下降 (Gradient Descent, GD) 的方法，二类是基于扩展 Baum-Welch(EB) 方法。在后文将讨论的深度学习框架下，基于梯度下降的算法被广泛使用，将在第 2.2.2.3 章进行详细介绍。

### 2.1.3 语言模型

语音信号的模型建模的是候选标注的先验概率，假设候选标注含有  $K$  个单词， $\mathbf{w} = \{w_1, \dots, w_K\}$ 。通过条件概率公式，语音信号的模型概率可以扩展为一部分条件概率连

乘：

$$P(\mathbf{w}) = \prod_{k=1}^K P(w_k | w_{k-1}, \dots, w_1) \quad (2-25)$$

这里  $w_k$  是序列中的第  $k$  个单词。利用公式 2-25 来计算先验概率需要记录整个句子的历史，然而在大词汇语音信号的识别任务中，由于词表大小往往能达到 10000 词以上，所有可能的历史数据太过庞大，因此很难对每个可能的词序列都进行鲁棒估计。一种解决方案是限制历史的长度， $N$  元组 ( $n$ -gram) 语音信号的模型即利用了这一策略，它也是目前语音信号的识别中最广泛利用的统计语言模型。它所作出假设是只需要最多利用  $N$  个单词作为历史就足够计算概率：

$$P(w_k | w_{k-1}, \dots, w_1) \approx P(w_k | w_{k-1}, \dots, w_{k-N+1}) \quad (2-26)$$

这里  $N$  是预先定义的历史窗口大小，语音信号的识别中通常利用  $N = 3$ ，也被称为三元语言模型。语言模型通常利用最大似然的准则进行训练，它的更新公式如下：

$$P(w_k | w_{k-1}, \dots, w_{k-N+1}) = \frac{f(w_k, w_{k-1}, \dots, w_{k-N+1})}{\sum_w f(w, w_{k-1}, \dots, w_{k-N+1})} \quad (2-27)$$

这里  $f(w, w_{k-1}, \dots, w_{k-N+1})$  表示这  $N$  个词按顺序出现在训练数据中的个数。因为采用的是最大似然估计，所以每个  $N$ -元序列得拥有充足的样本才能得到最鲁棒的估计。然而，即使  $N$  非常小，这一条件在大词汇连续语音信号的识别任务中依然很难达成。所以，过去研究提出了一部分平滑的方法来获得鲁棒的估计。

- 降权 (Discounting)

降权方法用于解决训练集中未观测到  $N$  元组概率无定义的问题，它的主要思想是将被观测到的  $N$  元组概率乘上一个降权系数，将剩下的部分平均分配给未观测到  $N$  元组。最常用的降权方法有 Good-Turing 法 [38, 39]，Witten-Bell 法 [58] 和绝对降权法 [59]

- 回退法 (Back off)

回退法利用训练数据中观测到的具有较短历史信息的词序列的组合概率来近似那些没有出现过、由较长历史信息组成词序列组合。

- 多模型插值 (Interpolation)

当一个  $N$  元语言模型不是很鲁棒时，可以通过和更低阶的语言模型进行插值以获得更平滑模型，比如在语音信号的识别中通常会将单元，二元和三元语言模型进行插值来构建一个更鲁棒语言模型。插值的权重通常在一个校验集上调节得到。同样，我们也可以用同样的方法对由不同语料训练出  $N$  元语言模型进行插值。

语言模型训练和评价的指标通常是使用困惑度 (Perplexity, PPL)，它定义是词序列生成概率的几何平均倒数：

$$PPL = 2^{-\frac{1}{K} \log(P(\mathbf{w}))} \quad (2-28)$$

其中  $K$  是词序列包含的总词数，拥有更低 PPL 语言模型具有更低的不确定度和困惑度，通常也能潜在的降低语音信号的识别词错误率。当然，这一关系并非一直成立，因而在语音信号的识别中最终还是以词错误率来判断语言模型的好坏。

#### 2.1.4 解码及搜索

语音信号的识别既是一个模式识别问题，也包含相应的推理搜索问题。前一个问题对各种语音信号的、语言现象进行数学表示和描述，在基于统计学习的模式识别框架下进行建模，这决定了语音信号的识别系统可达到识别准确度的上限。而后一个问题在给定模型情况下，研究如何高效地将输入语音信号的和模型相匹配，推理搜索得到最优识别结果，这决定了识别速度和实际可达的识别准确度。在语音信号的识别的推理搜索阶段，解码器功能是对声学模型计算出的声学特征概率和语言模型计算出的语言概率进行组合来得到最大概率的词序列。

语音识别系统可以分为关键词检测，孤立词识别，语法网络识别，大词汇连续语音识别和基于神经网络语音模型的大词汇连续语音识别。我们将在第三章进一步探讨解码器的具体设计。

## 2.2 深度学习在语音识别中的应用

近年来，深度学习模型被引入到语音识别声学和语言建模中以取代传统的分类器，显着提高了模式识别问题的准确性。在这一框架中，HMM 被用于描述语音信号的信号动态变化，而 DNN 则用于估计语音信号的特征向量的输出概率，通常是用来估计给定观测语音信号的特征后，该特征由 HMM 中某个状态释放后验概率。下面的章节我们将具体介绍深度学习模型与训练方法，以及深度学习在语音识别中的具体应用方法。

### 2.2.1 深度学习与神经网络

目前语音识别领域最常使用的神经网络结构包括三种：前馈网络、卷积神经网络以及循环神经网络，本章将对这三种网络结构进行逐一介绍。在语音识别领域，深度前馈网通常简称为深度神经网络。在本文中，深度神经网络默认表示为深度前馈网络。

### 2.2.1.1 深度神经网络

神经网络 [60] 也被称为多层感知器，它是一种最基本神经网络结构，也是目前使用最广泛结构。深度神经网络（Deep Neural Network, DNN）是一个含有很多（超过两个）隐层的多层感知器。图 2-3 绘制了一个总共五层深度神经网络，包括一个输入层、三个隐层以及一个输出层。这里我们将 DNN 的输入层设为层 0，将输出层设为层 L。神经网

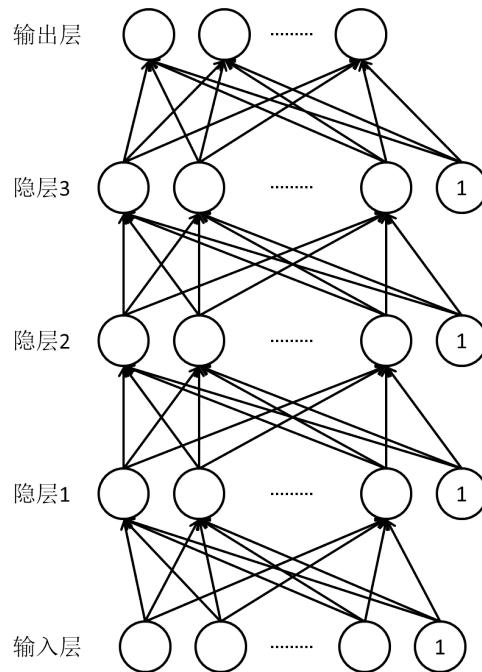


图 2-3 深度神经网络

Fig 2-3 Deep neural network

络运算可以被定义为：

$$\mathbf{y}^l = f(\mathbf{x}^l) = f(\mathbf{W}^l \mathbf{y}^{l-1} + \mathbf{b}^l), 0 < l < L \quad (2-29)$$

这里  $\mathbf{x}^l, \mathbf{y}^l, \mathbf{W}^l, \mathbf{b}^l$  分别是第  $l$  个隐层的激励向量、激活向量、权重矩阵和偏置向量。 $\mathbf{y}^0 = \mathbf{o}$  是网络输入特征向量。 $f$  是一种对激励向量进行元素级计算的激活函数。常用激活函数有：

- sigmoid 函数

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2-30)$$

- 双曲正切函数

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2-31)$$

双曲正切函数是 sigmoid 函数的调整版本，它们具有相同建模能力。区别是 sigmoid 函数的值域是 (0,1)，这有助于得到更稀疏表示。而 tanh 的值域是 (-1,1) 是对称，更容易训练。

- 整流线性单元 (ReLU)

$$\text{ReLU}(x) = \max(0, x) \quad (2-32)$$

它的导数更加简洁且不会随着层数增多而出现梯度消失现象。

如图 2-4 所示，其中黑色的曲线为 sigmoid；红色曲线为 tanh；蓝色的曲线为 ReLU。因为 sigmoid 函数被应用得更加广泛，在本文中如果没有特别标明，默认都是使用 sigmoid 函数。

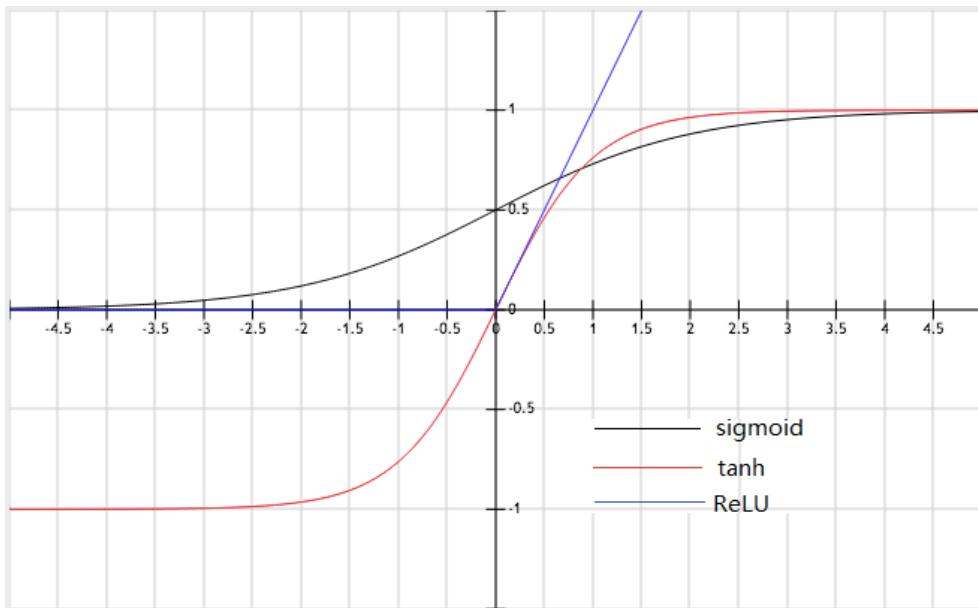


图 2-4 激活函数

Fig 2-4 Activation functions

深度神经网络输出层选定通常根据任务而变化，如果是回归任务，通常使用一个线性层：

$$\mathbf{y}^L = \mathbf{x}^L = \mathbf{W}^L \mathbf{y}^{L-1} + \mathbf{b}^L \quad (2-33)$$

如果是多分类的任务，通常会使用输出层每个神经元代表一类，其中第  $i$  个神经元的值即代表了当前特征属于类别  $i$  概率，因此它必须满足多项式分布，即  $y_i^L \geq 0$  且  $\sum_{i=1}^C y_i^L = 1$ ，

这里  $C$  为总类别数。为了满足这一限制，我们通常会使用 softmax 函数进行归一化：

$$y_i^L = P_{\text{dnn}}(i|\mathbf{o}) = \text{softmax}_i(\mathbf{x}^L) \quad (2-34)$$

$$= \frac{e^{x_i^L}}{\sum_{j=1}^C e^{x_j^L}} \quad (2-35)$$

给定了特征向量的输入后，DNN 输出由模型参数  $\mathcal{M} = \{\mathbf{W}^l, \mathbf{b}^l, 0 < l \leq L\}$  确定，通过使用前面提到的公式计算激活向量以及最后输出，这一过程也被称为前向计算。

### 2.2.1.2 卷积神经网络

传统的深度卷积神经网络 (convolutional neural network, CNN) [61] 通常由若干卷积层和池化层组成，最后接上一些全连接层。和深度神经网络相比，它主要区别在于卷积层和池化层。本章将具体介绍卷积层和池化层。

特征图谱是卷积层和池化层中最基本的单元，每个卷积层输入和输出都是若干张特征图谱。在语音识别中，包含静态特征、一阶和二阶动态特征的经典语音信号的特征可以表示为三张特征图谱。每张特征图谱是一张大小为  $N_t \times N_f$  的图片，通常为  $11 \times 40$ 。这里  $N_t$  是上下文窗口大小， $N_f$  是提取特征的维度。相比于 MFCC 或者 PLP 特征，FBANK 特征更适合 CNN。首先 FBANK 特征中各个维度之间是相关的，这种相关性可以被卷积进行捕捉。其次，池化操作中降采样需在一个有序的频率特征图谱中进行才有意义。MFCC 特征中的离散余弦变换将破坏这些性质。

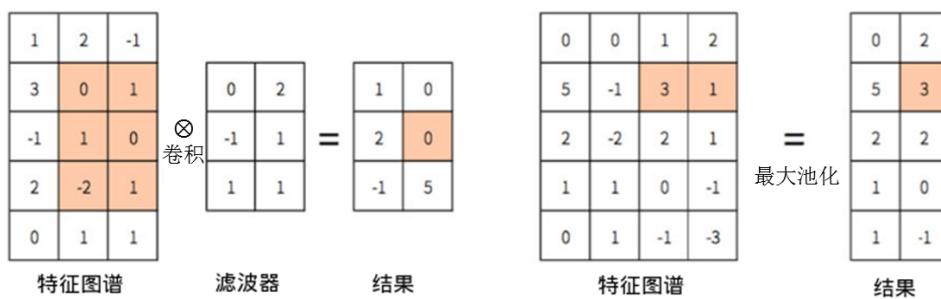


图 2-5 卷积操作和池化操作，左边为卷积，右边为最大池化

Fig 2-5 Convolution and pooling, (left: convolution, right: max pooling)

卷积操作可以视为在特征图谱上使用一个滤波器。特征图谱和滤波器都可以通过矩阵表示。卷积过程就是将滤波器从特征图谱的左上角逐行扫描到右下角。在这一过程中被滤波器覆盖到的区域被称为感受野。在每一步中，感受野会将滤波器和它所覆盖到的

特征图谱区域进行点乘得到一个输出值，将整张特征图谱扫描完后即可得到输出特征图谱，卷积数学定义为：

$$\mathbf{y} = \mathbf{W} \otimes \mathbf{x} \quad (2-36)$$

$$y_{i,j} = \sum_{a=1}^A \sum_{b=1}^B W_{ab} x_{i+a-1, j+b-1} \quad (2-37)$$

$$0 < i \leq W - A + 1 \quad (2-38)$$

$$0 < j \leq H - B + 1 \quad (2-39)$$

$$(2-40)$$

其中  $W$  是大小为  $A \times B$  的二维滤波器， $\mathbf{x}$  是大小为  $W \times H$  输入特征图谱， $\mathbf{y}$  是大小为  $W - A + 1 \times H - B + 1$  的输出特征图谱。如图 2-5 中左边部分所示为卷积操作的图例，图中输入为一个  $5 \times 3$  特征图谱，滤波器大小为  $3 \times 2$ ，最后得到一个  $3 \times 2$  输出特征图谱。

整个卷积层可以由如下公式表示：

$$\mathbf{y}_i^l = \sigma \left( \sum_{j=1}^N (\mathbf{W}_{i,j}^l \otimes \mathbf{y}_j^{l-1}) \oplus b_i^l \right) \quad (2-41)$$

这里  $\mathbf{y}_i^l, \mathbf{y}_j^{l-1}$  分别是第  $l$  个卷积层的第  $j$  个输入特征图谱和第  $i$  个输出特征图谱。 $\mathbf{W}_{i,j}^l$  是这两个特征图谱间滤波器。 $b_i^l$  是一个应用于整张特征图谱偏置值， $\otimes$  为卷积操作， $\oplus$  表示特征图谱中每个元素都加上相同的值  $b_i^l$ 。 $\sigma$  是激活函数，通常使用 sigmoid 或者 ReLU。 $N$  是输入特征图谱个数。由于各个感受野之间的参数是共享的，因而 CNN 中参数个数相对 DNN 是很少的。

池化层通常扮演降采样功能，它输入若干特征图谱并输出降低了分辨率后特征图谱。本文使用了最大值池化方法，一个  $1 \times 2$  的最大值池化效果如图 2-5 中的右边部分所示。

### 2.2.1.3 循环神经网络

循环神经网络 (recurrent neural network, RNN) 是深度神经网络一个变种，在建模时序数据中特别有效。与深度神经网络不同点在于它包含有一条回环，在循环神经网络中，隐层在时刻  $t$  输出不仅仅作为下一层的输入，同时也会在时刻  $t+1$  输回给自身。因此它相当于存在内部状态，内部状态将过去历史信息进行了编码。最简单的循环神经网络可以被描述为：

$$\mathbf{y}_t^l = \sigma(\mathbf{W}_y^l \mathbf{y}_{t-1}^{l-1} + \mathbf{W}_h^l \mathbf{y}_t^{l-1} + \mathbf{b}^l) \quad (2-42)$$

这里  $\mathbf{y}_t^l$  是网络第  $l$  层第  $t$  帧的隐层输出,  $\mathbf{W}_y^l, \mathbf{W}_h^l$  分别是用于对上层输出和对历史输出权重矩阵。

通过这一设计, 神经网络可以使用所有的过去帧的信息, 上下文信息在声学模型建模中扮演了很重要的作用。在 [62, 63, 64, 65] 中, 作者使用了深度循环神经网络来建模声学模型。双向循环神经网络也被用于语音识别, 它不仅能捕捉过去历史信息, 也能捕捉未来的整个输入序列信息。

在早期的工作中, RNN 层仅仅使用了一个线性变换接上一个元素级激活函数, 这种结构没法保留长时信息, 因为通用的激活函数会对动态范围进行压缩, 因此在  $n$  帧之前历史信息在到达当前帧时已经被压缩了  $n$  次, 很难对当前决策产生影响。在使用沿时误差反向传播算法中, 这一问题也被称为梯度消失问题 [66], 即误差无法沿着时间维传播太多帧。为了解决这一问题, 在 [67] 中, 长短时记忆循环神经网络 (long short-term memory, LSTM) 被提出用来取代传统的 RNN 结构。它通过引入记忆单元和门操作来避免梯度消失问题, 一个典型的 LSTM 可由如下公式描述:

$$\mathbf{i}_t^l = \sigma(\mathbf{W}_{yi}^l \mathbf{y}_t^{l-1} + \mathbf{W}_{hi}^l \mathbf{y}_{t-1}^l + \mathbf{W}_{ci}^l \mathbf{c}_{t-1}^l + \mathbf{b}_i^l) \quad (2-43)$$

$$\mathbf{f}_t^l = \sigma(\mathbf{W}_{yf}^l \mathbf{y}_t^{l-1} + \mathbf{W}_{hf}^l \mathbf{y}_{t-1}^l + \mathbf{W}_{cf}^l \mathbf{c}_{t-1}^l + \mathbf{b}_f^l) \quad (2-44)$$

$$\mathbf{c}_t^l = \mathbf{f}_t^l \odot \mathbf{c}_{t-1}^l + \mathbf{i}_t^l \odot \tanh(\mathbf{W}_{yc}^l \mathbf{y}_t^{l-1} + \mathbf{W}_{hc}^l \mathbf{c}_{t-1}^l + \mathbf{b}_c^l) \quad (2-45)$$

$$\mathbf{o}_t^l = \sigma(\mathbf{W}_{yo}^l \mathbf{y}_t^{l-1} + \mathbf{W}_{ho}^l \mathbf{y}_{t-1}^l + \mathbf{W}_{co}^l \mathbf{c}_{t-1}^l + \mathbf{b}_o^l) \quad (2-46)$$

$$\mathbf{y}_t^l = \mathbf{o}_t^l \odot \tanh(\mathbf{c}_t^l) \quad (2-47)$$

和循环神经网络一样, 它需要从第一帧开始迭代进行前向传播, 这里  $\sigma$  是非线性 sigmoid 函数,  $\mathbf{i}_t^l, \mathbf{f}_t^l, \mathbf{o}_t^l, \mathbf{c}_t^l, \mathbf{y}_t^l$  分别是第  $t$  帧的输入门、遗忘门、输出门、记忆单元和隐层输出。 $\odot$  表示元素级乘法。 $\mathbf{W}_*, \mathbf{b}_*$  分别是权重矩阵和偏置, 其中  $\mathbf{W}_{ci}, \mathbf{W}_{cf}, \mathbf{W}_{co}$  是对角矩阵。与最初的 LSTM 结构不同之处在于, 在这一结构中我们增加了窥孔连接 (peephole connection)。即: 将记忆单元的状态同时输入给各个门操作。

在大词汇连续语音识别任务中, 按照 [64] 中建议, 通常会使用一个投影层来降低计算量 (Long Short-Term Memory with Projection, LSTMP)。它和公式 2-43 的主要区别是在输出  $\mathbf{y}_t^l$  上增加一个线性变换来降低维度, 即:

$$\mathbf{y}_t^l = \mathbf{W}_r(\mathbf{o}_t^l \odot \tanh(\mathbf{c}_t^l)) \quad (2-48)$$

## 2.2.2 神经网络训练

### 2.2.2.1 训练准则

在 DNN 训练中通常有两个常用的准则，对于回归任务常使用均方误差 (mean square error, MSE) 准则：

$$\mathcal{F}_{\text{MSE}}(\mathcal{M}) = \frac{1}{T} \sum_{t=1}^T \|\mathbf{y}_t^L - \bar{\mathbf{y}}_t\|_2^2 \quad (2-49)$$

这里  $\mathbf{y}_t^L$  是由神经网络估计出输出， $\bar{\mathbf{y}}_t$  是正确标注。

对于分类任务而言，正确的标注为一个概率分布，通常使用交叉熵 (cross entropy, CE) 来进行优化：

$$\mathcal{F}_{\text{CE}}(\mathcal{M}) = -\frac{1}{T} \sum_{t=1}^T \sum_{c=1}^C \bar{y}_t(c) \log y_t^L(c) \quad (2-50)$$

这里  $\bar{y}_t(c)$  是标注中类  $c$  概率， $y_t^L(c)$  是神经网络估计出的类  $c$  概率。最小化交叉熵准则等价于最小化标注分布与 DNN 估计分布之间的 KL 距离 (Kullback-Leibler divergence, KLD)。一般，研究者通常使用硬标注作为标注分布，即：

$$\bar{y}_t(c) = \begin{cases} 1 & c = l_t \\ 0 & c \neq l_t \end{cases} \quad (2-51)$$

这里  $l_t$  是第  $t$  帧标注。此时，CE 准则退化为负对数似然准则 (negative log-likelihood, NLL)

$$\mathcal{F}_{\text{NLL}}(\mathcal{M}) = -\log y_t^L(l_t) \quad (2-52)$$

### 2.2.2.2 反向传播算法

给定了训练准则后，可通过著名的误差反向传播 (error back-propagation, BP) [60] 算法来进行参数优化。由于 BP 算法基于梯度下降方法来更新模型参数，因而其中的关键步骤是使用链式法则进行梯度计算。

最顶层权重矩阵的梯度取决于所使用训练准则。对于回归问题，当使用 MSE 的训练准则时，输出层权重矩阵的梯度是

$$\frac{\partial \mathcal{F}_{\text{MSE}}(\mathcal{M})}{\partial W_{ij}^L} = \frac{\partial \mathcal{F}_{\text{MSE}}(\mathcal{M})}{\partial x_j^L} \frac{\partial x_j^L}{\partial W_{ij}^L} \quad (2-53)$$

$$= e_j^L \frac{\partial x_j^L}{\partial W_{ij}^L} \quad (2-54)$$

$$= (x_j^L - \bar{y}_j) y_i^{L-1} \quad (2-55)$$

这里  $e_j^L$  为误差信号:

$$\mathbf{e}^L \triangleq \left[ \frac{\partial \mathcal{F}_{\text{MSE}}(\mathcal{M})}{\partial x_1^L}, \dots, \frac{\partial \mathcal{F}_{\text{MSE}}(\mathcal{M})}{\partial x_N^L} \right]^\top \quad (2-56)$$

$N$  是最后一层输出维度。也可以将对  $\mathbf{W}^L$  梯度写为矩阵形式:

$$\nabla_{\mathbf{W}^L} \mathcal{F}_{\text{MSE}}(\mathcal{M}) = \mathbf{e}^L \mathbf{y}^{L-1} \top = (\mathbf{x}^L - \bar{\mathbf{y}}) \mathbf{y}^{L-1} \top \quad (2-57)$$

当使用分类任务时, CE 准则通常和 softmax 输出层一起使用, 同样先计算出误差信号:

$$e_j^L = \frac{\partial \mathcal{F}_{\text{CE}}(\mathcal{M})}{\partial x_j^L} = -\frac{\partial \sum_{c=1}^C \bar{y}_c \log \text{softmax}_c(\mathbf{x}^L)}{\partial x_j^L} \quad (2-58)$$

$$= \frac{\partial \sum_{c=1}^C \bar{y}_c \log \sum_{i=1}^C e^{x_i^L}}{\partial x_j^L} - \frac{\partial \sum_{c=1}^C \bar{y}_c \log e^{x_c^L}}{\partial x_j^L} \quad (2-59)$$

$$= \frac{e^{x_j^L}}{\sum_{i=1}^C e^{x_i^L}} - \bar{y}_j \quad (2-60)$$

$$= y_j^L - \bar{y}_j \quad (2-61)$$

接着可以求出对  $\mathbf{W}^L$  的梯度:

$$\nabla_{\mathbf{W}^L} \mathcal{F}_{\text{CE}}(\mathcal{M}) = (\mathbf{y}^L - \bar{\mathbf{y}}) \mathbf{y}^{L-1} \top \quad (2-62)$$

值得注意一点是, 虽然 CE 准则和 MSE 准则梯度公式看上去有相同的形式, 但实际上它们是不同的, 因为在做回归时  $\mathbf{y}^L = \mathbf{x}^L$ , 而在做分类时  $\mathbf{y}^L = \text{softmax}(\mathbf{x}^L)$ 。

对于非顶层权重矩阵的梯度  $0 < l < L$ , 则有:

$$\frac{\partial \mathcal{F}(\mathcal{M})}{\partial W_{ij}^l} = \frac{\partial \mathcal{F}(\mathcal{M})}{\partial x_j^l} \frac{\partial x_j^l}{\partial W_{ij}^l} \quad (2-63)$$

$$= e_j^l \frac{\partial x_j^l}{\partial W_{ij}^l} \quad (2-64)$$

$$= e_j^l y_i^{l-1} \quad (2-65)$$

这里  $\mathbf{e}^l \triangleq \nabla_{\mathbf{x}^l} \mathcal{F}(\mathcal{M})$  是对第  $l$  层的激励误差信号:

$$e_i^l = \frac{\partial \mathcal{F}(\mathcal{M})}{\partial x_i^l} \quad (2-66)$$

$$= \sum_{j=1}^{N^{l+1}} \frac{\partial \mathcal{F}(\mathcal{M})}{\partial x_j^{l+1}} \frac{\partial x_j^{l+1}}{\partial y_i^l} \frac{\partial y_i^l}{\partial x_i^l} \quad (2-67)$$

$$= \sigma'(x_i^l) \sum_{j=1}^{N^{l+1}} e_j^{l+1} W_{ji}^{l+1} \quad (2-68)$$

这里  $N^{l+1}$  是第  $l + 1$  层隐层节点数，写出矩阵形式即：

$$\mathbf{e}^l = (\mathbf{W}^{l+1}^\top \mathbf{e}^{l+1}) \odot \sigma'(\mathbf{x}^l) \quad (2-69)$$

这里  $\odot$  为元素级相乘， $\sigma'(x_j^l)$  为激活函数的元素级导数，对于 sigmoid 函数，它等于：

$$\sigma'(x_j^l) = (1 - y_j^l)y_j^l \quad (2-70)$$

令  $\sigma'(\mathbf{x}^l) = [\sigma'(x_1^l), \dots, \sigma'(x_{N^l}^l)]^\top$ 。

因此对权重矩阵梯度矩阵形式为：

$$\nabla_{\mathbf{W}^l} \mathcal{F}(\mathcal{M}) = \mathbf{e}^l \mathbf{y}^{l-1} \top \quad (2-71)$$

从公式2-69中可以观察到，误差信号需要自顶层向下进行反向传播，因此该方法被称为误差反向传播算法。

### 2.2.2.3 神经网络的优化

深度神经网络采用梯度下降的方法进行优化，因此每一轮迭代都需要对一个批量 (batch) 训练样本计算梯度，称为小批量随机梯度下降。批量大小的选择上通常会同时影响收敛速度与模型的性能。

最简单批量选择方法是去使用整个训练集，这种方法被称为批量训练 (batch training)。它有如下优势：首先，批量训练的收敛性是众所周知的，其次，批量训练可以在多计算机之间并行。但由于每次参数更新都要遍历全部数据集 [68]，所以对大词汇语音识别任务而言，这是十分低效的。

另外一种选择是随机梯度下降法 (Stochastic gradient decent, SGD) [69]，在机器学习领域也被称作在线学习。**SGD** 根据从单个训练样本估计出的梯度来更新模型参数 [68]。如果样本点是独立同分布的 (在训练数据中随机采样很容易达成)，可以证明这种方法是对梯度的无偏估计，只不过存在噪声。这点虽然看似不利，实则是 **SGD** 相比批量算法优势所在：由于 DNN 是高度非线性的且优化准则也是非凸的，因而目标函数包含很多局部最优，而 **SGD** 估计中存在噪声，从而使得其能从局部最优中跳出来进入一个更好全局最优。

**SGD** 通常比批量训练快很多，特别是在大词汇连续语音识别中。这是因为在大数据集中，通常有很多样本是相似甚至重复的，所以用全部数据集来计算梯度是浪费的。然而，由于 **SGD** 只计算一个样本梯度，无法高效的利用 GPU 的并行能力，且因为噪声的存在无法完全收敛至局部最优点，所以通常会使用 **SGD** 和批量训练折中方案，小批

量 (mini-batch) 训练 [68]。小批量数据通过从训练样本中抽取一小组数据来计算梯度使得其可以高效的使用 GPU 的并行能力，并且也拥有 SGD 跳出局部最优的性质。

惯性系数 [70] 是优化中一种加速收敛的方法，它主要思想是使用过往梯度的一个累积量和当前梯度的插值作为最终更新使用的梯度，即：

$$\Delta \mathbf{W}_p^l = \rho \Delta \mathbf{W}_{p-1}^l + (1 - \rho) \frac{1}{M} \sum_{m=1}^M \nabla_{\mathbf{W}_p^l} \mathcal{F}(\mathcal{M}, \mathbf{o}_m, \bar{\mathbf{y}}_m) \quad (2-72)$$

这里， $\nabla_{\mathbf{W}_p^l} \mathcal{F}(\mathcal{M}, \mathbf{o}_m, \bar{\mathbf{y}}_m)$  是使用样本  $\mathbf{o}_m, \bar{\mathbf{y}}_m$  计算出的梯度。 $\Delta \mathbf{W}_p^l$  是第  $p$  轮迭代累积的梯度。 $\rho$  即为惯性系数。使用惯性系数会令参数更新变得更加平滑，同时减少梯度估计时方差，因此可以加速训练。

#### 2.2.2.4 参数初始化与预训练

通常有很多启发式的方法来初始化 DNN 模型，它们的主要思想都是希望初始化后的激活范围处于 sigmoid 函数的线性段。即激活值为 0.5 左右。如果权重过大将导致激活值趋近于 0 或者 1，这样计算出梯度就会非常小。随机初始化的另外一个作用是打破深度神经网络的对称性。在 [71] 中，Lecun 建议从一个均值为 0，方差为  $\frac{1}{\sqrt{N}}$  的高斯分布中采样。由于在语音识别中隐层节点的个数通常从 1000 至 2000 不等，所以一般选取 [-0.05, 0.05] 的均匀分布作为权重矩阵的初始化，而偏置初始化为 0。

由于 DNN 模型高度非线性和非凸性，研究者通常会使用一些预训练的方法使其能从一个更优的初始值开始优化。最著名的两个预训练方法是深度置信网络和鉴别性预训练。

深度置信网络 (Deep belief network, DBN) 是由多层受限玻尔兹曼机 (restricted Boltzmann machine, RBM) 组成生成性模型。

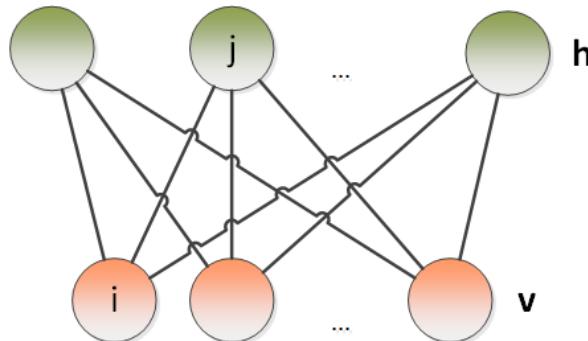


图 2-6 受限玻尔兹曼机

Fig 2-6 Restricted Boltzmann machine

受限玻尔兹曼机是一种生成性模型，它是玻尔兹曼机的一个变种。它结构如 2-6 所示<sup>1</sup>，它取消了玻尔兹曼机中可见层神经之间和隐藏层神经之间的连接。因此，它形成了一张二分图。

在 **RBM** 中，由于隐层神经元之间没有连接，因而可见层神经元之间也没有连接 [68]。所以，可以很方便计算后验概率。将多个 **RBM** 网络自底向上逐层堆叠就得到了深度信念网络，在深度信念网络中每一层的输出都作为后一层的输入，然后用新特征继续训练一个 **RBM**。最终 **DBN** 训练完成后将 **DBN** 中的权重直接作为 **DNN** 网络初始化。

使用 **DBN** 进行预训练有如下潜在优势：

- **DNN** 是高度非线性且非凸的，初始化点会很大程度影响最终模型。
- 预训练阶段使用的是生成性准则与反向传播算法中使用的鉴别性准则不同 [68]，因此潜在对模型进行了正则化。
- 在预训练时可以利用大量的无标注数据。

**DNN** 的参数也可以使用鉴别性预训练 (discriminative pretrain, DPT) 来鉴别性地初始化，即逐层进行 **BP**。鉴别性预训练流程如下：首先使用标注鉴别性训练一个单隐藏层的 **DNN** 若干轮 (并不需要训练到收敛)，接着在最后一个隐层和输出层之间插入一个新的随机初始化的隐藏层，并且鉴别性训练整个网络若干轮，直至达到期望的层数。预训练目标是希望网络能从一个更优秀的初试点开始进行训练以潜在地改善神经网络鲁棒性。然而，当训练数据足够大 (大于 15 小时) 或者使用 **ReLU** 作为激活函数之后，预训练带来性能提升就不那么明显了。

### 2.2.3 深度神经网络-隐马尔科夫模型混合系统

在前文介绍的深度神经网络并不能直接为语音信号的信号建模，这是由于语音信号处理的信号是一种时序连续信号，而深度神经网络的输入是固定大小的向量。早在 20 世纪 80 年代末就已经有研究提出了结合人工神经网络 (**ANN**) 和 **HMM** 方法用于语音识别 [72]。目前有效的方法是如图 2-7 中所描绘的 **DNN-HMM** 混合系统。在这一框架中，**HMM** 被用于描述语音信号的信号动态变化，而 **DNN** 则用于估计语音信号的特征向量的输出概率，通常是用来估计给定观测语音信号的特征后，该特征由 **HMM** 中某个状态释放后验概率。换句话说，**DNN** 只是替代了 **GMM** 在公式 (2-4) 中对  $p(\mathbf{O}|\mathbf{s}, \mathcal{M})$  的建模。

---

<sup>1</sup> 图片引用自 [28]

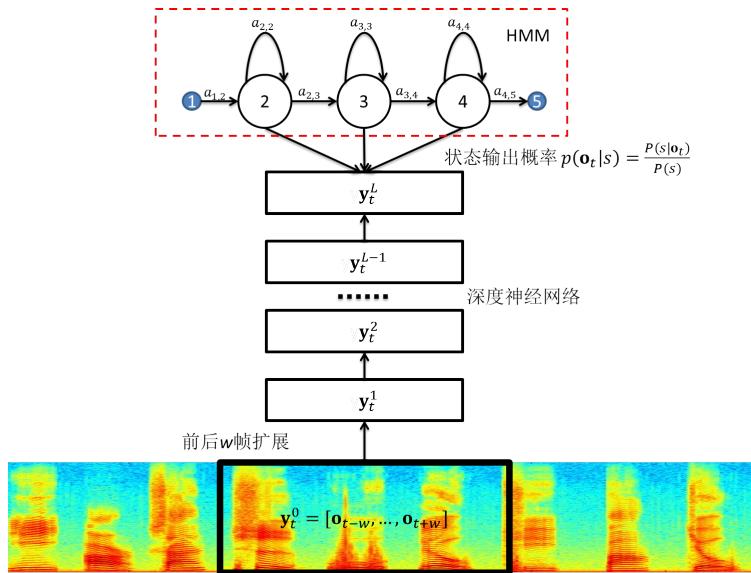


图 2-7 DNN-HMM 混合系统

Fig 2-7 Structure of DNN-HMM hybrid system

DNN-HMM 的优点在于它能方便的利用 DNN 内在的鉴别性能力，另外训练过程可以直接使用维特比算法，解码过程不需要有特别改变。

在 [25, 73, 74] 中，该方法被称为 ANN-HMM 混合模型，它只使用了上下文无关音素状态之间且仅用于小词汇语音识别。随后在 [26] 中被扩展为上下文相关的音素建模，以及被用于 LVCSR 任务 [75]。但是，因为过去计算能力有限，通常只使用了两层隐层的神经网络，所在并没有比 GMM-HMM 框架取得更好性能。

最近的技术发展 [29, 30, 76, 68] 则说明，如下几个改变可以使其获得更重大识别性能提升。

- 首先，使用更深的深度神经网络，比如拥有 6 个隐层 DNN。
- 其次，使用语素（即状态聚类后的三音素状态）来代替单音素状态作为深度神经网络的输出单元。这一模型被称为上下文相关的深度神经网络隐马尔科夫模型（Context-dependent pre-trained deep-neural-network hidden markov model, CD-DNN-HMM）。直接使用语素还有一个额外好处，即它对 CD-GMM-HMM 系统修改最小。

在 CD-DNN-HMM 中。对于所有的状态  $s \in [1, S]$  中只有一个完整的 DNN 来估计状态后验概率  $P(s_t = s | \mathbf{o}_t)$ 。这和传统的 GMM 是不同的，在 GMM 中，我们使用不同 GMM 来对不同的状态建模。除此之外，典型的 DNN 输入不是单独一帧，而是一个  $2w + 1$  帧

大小窗口特征  $\mathbf{y}_t^0 = [\mathbf{o}_{t-w}^\top, \dots, \mathbf{o}_t^\top, \dots, \mathbf{o}_{t+w}^\top]^\top$

在解码过程中，因为 HMM 需要使用似然度  $p(\mathbf{o}_t|s_t)$  而非后验概率，所以我们需要将后验概率转为似然度：

$$p(\mathbf{o}_t|s_t = s) = \frac{P(s_t = s|\mathbf{o}_t)p(\mathbf{o}_t)}{P(s)} \quad (2-73)$$

其中， $p(s) = \frac{T_s}{T}$  是从训练集中统计出的每个状态先验概率。 $T_s$  是状态  $s$  的帧数， $T$  是总帧数。因为  $p(\mathbf{o}_t)$  是与词序列无关的，计算时可忽略。所以最终似然可以通过  $p(\mathbf{o}_t|s_t = s) = \frac{P(s_t = s|\mathbf{o}_t)}{P(s)}$  计算。

CD-DNN-HMM 训练可以使用维特比算法来进行，CD-DNN-HMM 包含了三个组成部分，深度神经网络 DNN，隐马尔科夫模型 HMM 和 GMM-HMM 系统中的状态绑定结构的状态。所以 CD-DNN-HMM 第一步是去使用训练数据训练一个 GMM-HMM，然后在该系统上使用维特比算法获得状态级的强制对齐，同时保留该系统中的状态转移概率和状态映射关系，接着使用由 GMM-HMM 系统计算得到状态对齐 (state alignment) 作为标注使用 CE 准则训练一个 DNN。最后，解码时使用公式 2-73 来得到似然进行解码。

## 2.3 基于深度学习的序列建模

由于语音识别本身是一个序列预测问题，所以序列级的准则往往能够带来性能的提升。依据序列的不同建模方式，目前有两种不同的序列鉴别性训练方法，一种针对生成式序列模型 (GSM) 比如上文介绍的 HMM 以及其相应的深度学习系统；另一种则针对鉴别式序列模型 (DSM)，比如 CTC。Encoder-decoder 模型也是一类鉴别式序列模型，我们将在第2.4章节讨论。

### 2.3.1 生成式序列模型和鉴别式序列模型

在语音识别中一个独有并且有趣的自然现象是声学序列和语言学序列的长度可变性。该任务中，在训练阶段，一组带有已知标签的输入特征被提供给系统进行模型构建；而测试阶段，则基于特征序列和其它知识源如语言模型和字典进行模型推理搜索。序列标注问题与传统模式识别框架的区别在于以下两个方面。

- 序列内数据的相关性。无论是特征序列，还是标签序列，序列中各数据点均不符合独立同分布 (i.i.d.) 假设。特征序列是由声道的连续运动而产生的。而标签序列则受到句法和语法规则、字典以及语言模型的约束。因此，特征和标签均为强相关序列。

- 标签和特征序列之间的相关性。ASR 中，特征和标签之间的对齐方式是未知的，标签序列总是短于特征序列，即其主要问题在于由语速变化等导致的特征序列的可变长性。这就要求序列模型能够同时确定输出标签的位置和内容。

如果我们定义语音识别所要建模的标签输出序列为  $\mathbf{L}$ ，特征输入序列为  $\mathbf{O}$ ，那么为了对上述序列  $\mathbf{L}$  与  $\mathbf{O}$  的相关特征进行建模，人们提出了序列模型。下文中我们将首先给出序列模型的定义方式，而后在第2.3.2章节和第2.3.3章节讨论这些模型在语音识别任务中如何进行序列建模。

大部分序列模型都包含一个用来对时序特性建模的部分，比如 GMM [53] 和深度学习模型 [77] 用来对局部的声学特征进行建模，再加上一个对序列长度和转移进行建模的部分，比如 HMM。近期提出的鉴别式序列模型，encoder-decoder [78] 则尝试直接对序列建模而不是进行序列到逐帧的拆分。但在大部分语音识别研究中，性能和延迟在上述模型中都有一定限制。因此不参与本章节讨论。本章节所有的序列鉴别性训练方法均基于逐帧的分解方式。基于模型是对条件似然度或者后验概率进行建模的不同，概率模型通常可以分为生成式和鉴别式两种类型。值得注意的是在序列上或者在帧上，都可以使用生成式或者鉴别式进行分别建模，二者工作在不同层面上。比如目前通常使用的混合 NN-HMM 模型就是采用生成式序列模型和鉴别式帧分类器。在本章节中，我们主要关注序列模型。下文我们将比较模型结构和优化准则，在生成式和鉴别式序列模型中的异同。

生成式序列模型 (Generative Sequence Model, GSM) 通常被定义为条件似然度  $p(\mathbf{O}|\mathbf{L})$ ，公式中  $\mathbf{O}$  表示特征输入序列， $\mathbf{L}$  表示标签输出序列。隐马尔科夫模型 (HMM) 是这类生成式序列模型的一个经典例子<sup>1</sup>。在 NN-HMM 混合系统中，语音信号的动态性使 HMMs 和来自深度学习模型的后验概率进行建模。

$$\begin{aligned} p(\mathbf{O}|\mathbf{L}) &= \sum_{\mathbf{q} \in \mathcal{A}(\mathbf{L})} p(\mathbf{O}, \mathbf{q}|\mathbf{L}) = \sum_{\mathbf{q}} \prod_{t=1}^T p(\mathbf{o}_t|q_t) P(q_t|q_{t-1}) \\ &\propto \sum_{\mathbf{q}} \prod_{t=1}^T \frac{P(q_t|\mathbf{o}_t)}{P(q_t)} P(q_t|q_{t-1}) \end{aligned} \quad (2-74)$$

公式中  $\mathbf{L}$  表示标签序列，比如 LVCSR 中的词序列。 $\mathbf{q}$  表示 HMM 状态序列， $q_t$  表示在帧  $t$  上的 HMM 状态。 $P(q_t|q_{t-1})$  表示 HMM 状态转移概率， $P(q_t)$  表示状态的先验概率。 $\mathcal{A}$  表示由标签序列  $\mathbf{L}$  到它的相应 HMM 状态序列  $\mathbf{q}$  之间的映射，

$$\mathcal{A} : \mathbb{L} \mapsto \{q_0^{(1)}, \dots, q_4^{(1)}, \dots, q_4^{(|\mathbb{L}|)}\} \quad (2-75)$$

<sup>1</sup>另一类 GSM 例子是 Kalman 滤波器 [79, 80].

$\mathbf{L}$  表示  $\mathbf{L}$  中所有的标签。 $q_s^{(l)}$  表示第  $l$  个 HMM 的第  $s$  个 HMM 状态。具体来说，每个标签单元，比如 tri-phone，对应于一个 HMM 模型。每个 HMM 模型则包含 5 个独立的状态，如图 2-8(a) 所示。状态后验概率  $P(q_t|\mathbf{o}_{ut})$  是由神经网络来估计得到的。当一个序列级鉴别性训练准则，比如序列后验概率，被优化时，我们一般采用贝叶斯公式分解来使得生成式模型可以使用鉴别性准则进行优化。

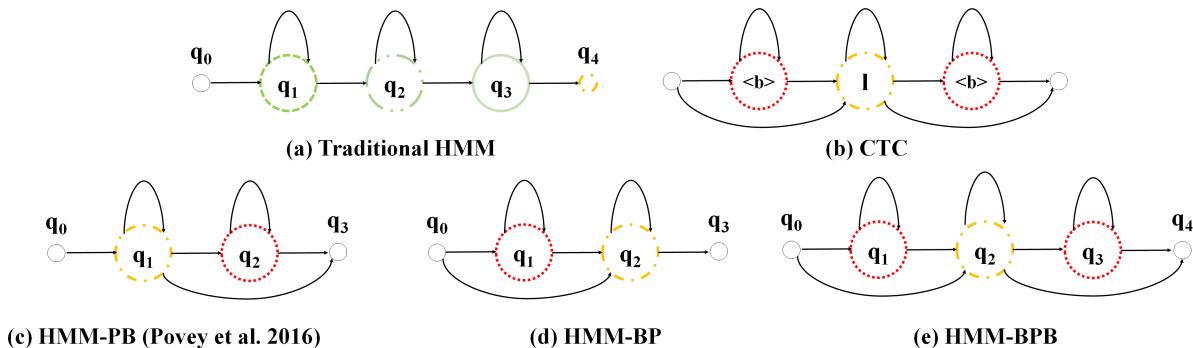


图 2-8 HMM, CTC 和本文提出的方法中隐藏状态拓扑结构示意图。在后面三种拓扑结构中，B 指 blank HMM 状态，P 指标签输出 HMM 状态。每个圆圈代表一个由神经网络建模发射概率的 HMM 状态。其中，点划线圆圈表示输出标签建模，如 (b) CTC 中的 1，都各自分配一个特定的模型单元。虚线圆圈表示 blank 建模，但并不完全相同，如 (b) CTC 中的  $<\text{b}>$  是使用公共的 blank 建模；但 (c) 中的  $q_2$ ，每个输出标签有独立的 blank 建模，本文将详细比较不同 blank 的粒度和拓扑结构所带来的区别。其它实线小圆圈，如 (c) 中  $q_0, q_3$ , (d) 中  $q_0, q_3$ , (e) 中  $q_0, q_4$ ，代表非发射状态。自循环状态转移表示该状态接受当前状态的重复输出。本文将对这些拓扑结构进行详细比较。

Fig 2-8 The hidden state topologies of HMM, CTC, and the proposed method.

鉴别式序列模型 (Discriminative Sequence Model, DSM) 由  $P(\mathbf{L}|\mathbf{O})$  进行定义，表示给定特征序列  $\mathbf{O}$  建模序列  $\mathbf{L}$  的后验概率。当使用了序列鉴别性模型之后，序列鉴别性准则的优化形式就非常直接了。连接时序分类模型 (CTC) [43] 是其中一种鉴别式序列模型。它引入了 blank 标签来建模输出标签之间的混淆部分。我们将在第 2.4.2 章节详细介绍 CTC 的定义方式。

### 2.3.2 基于 HMM 的序列鉴别性训练

隐马尔科夫模型 (HMM) 是一个序列生成模型，根据前文的介绍，其建模  $p(\mathbf{O}|\mathbf{L})$ ，所以使用 HMM 来对序列后验概率  $p(\mathbf{L}|\mathbf{O})$  建模时，需要利用贝叶斯公式将后验计算转

化为用条件概率（即 HMM 的定义）表达的形式。

$$P(\mathbf{W}_u | \mathbf{O}_u) = \frac{p(\mathbf{O}_u | \mathbf{W}_u) P(\mathbf{W}_u)}{p(\mathbf{O}_u)} \quad (2-76)$$

公式中  $\mathbf{W}_u$  是句子  $u$  对应的词序列。在语音识别中， $P(\mathbf{W}_u)$  是语言模型概率。在关键词检测任务中， $P(\mathbf{W}_u)$  则可以被定义为关键词和非关键词序列的先验概率。 $p(\mathbf{O}|\mathbf{W})$  是相应的声学部分，由下式得到。

$$p(\mathbf{O}|\mathbf{W}) = \sum_{\mathbf{L} \in \mathcal{L}(\mathbf{W})} p(\mathbf{O}|\mathbf{L}) P(\mathbf{L}|\mathbf{W}) \quad (2-77)$$

公式中  $p(\mathbf{O}|\mathbf{L})$  是由 HMM 中的定义公式 (2-74) 给定的。 $\mathcal{L}$  定义了从词序列  $\mathbf{W}$  到它的标签序列  $\mathbf{L}$  的映射，比如 LVCSR 中的 tri-phone 序列。 $P(\mathbf{L}|\mathbf{W})$  是发音概率 [81] 通常由词典和语言模型来决定。

特征序列  $\mathbf{O}_u$  的边缘概率  $p(\mathbf{O})$  是由所有可能的序列的概率进行求和得到的。

$$p(\mathbf{O}_u) = \sum_{\mathbf{W}} p(\mathbf{O}_u, \mathbf{W}) = \sum_{\mathbf{W}} P(\mathbf{W}) p(\mathbf{O}_u | \mathbf{W}) \quad (2-78)$$

公式中  $\mathbf{W}$  表示其中一条竞争的可能路径，具体来说是词图中的一条解码路径。作为序列鉴别性训练的一个例子，最大互信息准则 maximum mutual information (MMI) [17] 如下定义，

$$\mathcal{F}_{\text{MMI}} = \sum_u \log \frac{p(\mathbf{O}_u | \mathbf{W}_u)^\kappa P(\mathbf{W}_u)}{\sum_{\mathbf{W}} p(\mathbf{O}_u | \mathbf{W})^\kappa P(\mathbf{W})} \quad (2-79)$$

公式中的分布  $P(\mathbf{W}_u | \mathbf{O}_u)$ ，在 (2-76) 中定义，并被添加了一个常量  $\kappa$  进行语言模型的权重调节。在公式 (2-79) 中给定了序列后验概率表达式，而其它一些序列鉴别性训练准则则是由贝叶斯风险框架推导得到的，将在接下来的章节中进行讨论 4.1.2.1。

上面提到的序列竞争可能路径  $\mathbf{W}$  在 (2-78) 中可以得到，与语言模型一起进行联合解码。解码得到的词图作为一种搜索空间的紧致表示，可以用于计算公式 (2-78) 中的边缘概率  $p(\mathbf{O}_u)$ 。

### 2.3.3 基于 CTC 的序列鉴别性训练

鉴别式序列模型是直接对序列后验概率进行估计的一类模型，如公式 (2-82) 所示，符合语音识别任务对序列后验概率  $p(\mathbf{L}|\mathbf{O})$  建模的需求。连接时序分类模型 (CTC) 是一个典型的例子 [43, 44]，其目标函数为，

$$P(\mathbf{W}_u | \mathbf{O}_u) = \sum_{\mathbf{L} \in \mathcal{L}(\mathbf{W}_u)} P(\mathbf{L} | \mathbf{O}_u) P(\mathbf{W}_u | \mathbf{L}) \quad (2-80)$$

公式中  $\mathbf{L}$  是 CTC 模型的标签序列，比如半词或者音素序列等， $\mathcal{L}$  定义了从词序列  $\mathbf{W}$  到它的标签序列  $\mathbf{L}$  的映射。语音识别中，CTC 通常被用来建模半词或者音素序列等。在 [82] 中，CTC 被应用到 KWS 中，其输出也可以被定义为所有可能的关键词序列  $\mathbf{W}$  的后验概率。在语音识别中， $P(\mathbf{W}_u|\mathbf{L})$  由语言模型和词典给出。在声学 KWS 中，没有引入语言模型， $P(\mathbf{W}_u|\mathbf{L})$  是一个确定性的由词典到关键词序列的映射。 $P(\mathbf{L}|\mathbf{O}_u)$  可以进一步映射到 CTC 模型的标签序列，并分解到每一帧，如公式 (2-82) 所示。注意到一个额外的 blank 单元被引入到逐帧的公式 (2-82) 中以建模语音信号的混淆区段。序列鉴别性准则定义为所有可能的 CTC 标签序列的概率求和，作为最终的优化方向。

## 2.4 语言识别中的端到端模型

第2.2章节介绍的传统 DNN-HMM 模型核心在于将整体语音识别系统依据人类语言的先验知识进行模块化 (modularization)，而后分部分建模，再使用第2.3章节中的序列鉴别性训练技术进行联合调优。这样的系统普遍存在两部分潜在缺点：

- 模块化限制了序列建模能力。模块化依据人类语言学知识来进行建模步骤的划分，但往往这些先验知识并不一定反映真实的自然语言现象。而模块化通常假设各模块之间是条件独立的 (比如语言模型只建模词语之间的概率，而与声学特征无关)，那么错误的条件独立假设将会限制最终系统的性能。这些条件独立假设通常出现在人类语言的不同粒度上 (比如音素、字词)，因此上述性能的限制就具体体现在序列建模能力上。
- 模块化使得系统构建步骤复杂。如第2.1章节所述，传统系统搭建步骤复杂，还需要大量人类数据科学家进行模型超参数调优。语音识别的广泛推广，使得对大量语言和大量工作场景的支持成为了迫切的需求，这就对系统搭建难度和速度提出了新的要求。同时语言学研究尚未深入涉及人类的大量语言，而 DNN-HMM 这类模块化系统较难在语言学知识缺失 (比如音素集合缺失) 的情况下被搭建。

得益于标注数据的增多和算力改善，深度学习模型取得了更强的分类能力。由此人们开始考虑直接使用深度序列模型对序列进行直接建模，以取得更好序列建模能力。我们称这种“直接建模”的发展趋势为“端到端”。如图 2-9所示，原来传统系统中的 tri-phone HMM 状态建模可以直接被深度模型所取代，由此直接对音素序列进行建模。值得注意的是，根据上一章节中的分类，这类模型绝大部分为鉴别式序列模型，由  $P(\mathbf{L}|\mathbf{O})$  进行定义，表示给定特征序列  $\mathbf{O}$  建模输出序列  $\mathbf{L}$  的后验概率。这里的  $\mathbf{L}$  不仅可以是上

面提到的音素，也可以是字（grapheme or character），字母（letter），音节（syllable），或词语（word）。



图 2-9 端到端系统举例

Fig 2-9 Motivation and Example of End-to-end Modeling

下文中将逐一讨论各种端到端序列建模模型。

#### 2.4.1 无词图鉴别性训练的模型

第2.3章节所讨论的传统序列鉴别性训练系统是一种比较经典的序列建模方法，但存在几方面缺陷：

- 词图构建引入了一个额外的步骤，使得词图更新频度和词图质量需要进行额外的权衡。词图用于对搜索空间进行限制，因此较准确的方法是使用本次模块更新迭代之前的原始模型来生成词图，再依托该词图在进行模型更新迭代。但是每次生成词图需要联合语言模型进行语音识别解码，计算复杂度较大。因此传统序列建模方法通常会降低词图更新的频度以减少计算量。
- 逐句生成的词图导致序列建模过程中较难进行多句并行训练，而并行训练是目前最主流的加速模型训练的方式，训练速度的问题导致目前难以进行大规模的序列建模训练。因此传统 DNN-HMM 模型无法直接进行序列建模训练，而只能依托交叉熵训练作为初始化，然后使用少量数据或少量迭代来进行序列建模训练。
- 基于词图的方法基于一个初始化过的种子模型，因此种子模型的质量一方面关系

到目前流行的深度学习模型的最终优化结果，另一方面也影响标签序列逐帧强制对齐的精度，从而影响最终序列建模训练的结果。

除此之外，该方法依托传统 DNN-HMM 训练系统搭建，因此还存在前文讨论的模块化所带来的潜在缺点。

无词图鉴别性训练 (*lattice-free maximum mutual information*, LF-MMI [83, 84]) 是指使用一个预先剪枝过的音素语言模型来代替传统鉴别性训练中词图。该语言模型用于建模鉴别性训练中的搜索空间。

训练阶段依据公式2-81计算声学模型序列后验概率，表示为 LF-MMI。

$$\mathcal{F}_{\text{LF-MMI}} = \sum_u \log \frac{\sum_{\mathbf{L}_u} p(\mathbf{O}_u | \mathbf{L}_u)^\kappa P(\mathbf{L}_u)}{\sum_{\mathbf{L}} p(\mathbf{O}_u | \mathbf{L})^\kappa P(\mathbf{L})} \quad (2-81)$$

在公式中， $\mathbf{L}$  是由音素组成的发音序列， $\mathbf{L}_u$  是标注序列，也由音素组成。 $p(\mathbf{O}_u | \mathbf{L})$  和  $p(\mathbf{O}_u | \mathbf{L}_u)$  由 HMM 模型公式定义得到。

该方法与传统鉴别性训练还有以下的不同之处：

- 在分母式子中所使用的标注序列  $\mathbf{L}_u$  存在多种候选路径，这些候选路径来自于标注软对齐中对标签在一定窗宽内左右帧移。这里将所有可能的对齐路径都存储在分子词图中。
- 模拟搜索空间的分母语言模型  $P(\mathbf{L})$ ,  $P(\mathbf{L}_u)$  使用的是一个在训练标注文本中训练得到的音素语言模型。以这个模型来替代词图的使用。
- 一个专用 HMM 拓扑结构被专门提出，它包含有两个 HMM 状态。其中状态  $\mathbf{q}_2$  用于模拟 CTC 中的 blank 建模单元，同时其它状态来模拟输出标签单元。这里不同之处在于每个 tri-phone 都维护一个它专有的 blank 建模。LF-MMI 的隐状态可以表示为图 2-8(c)。
- 输出帧率被降低了 3 倍。

[83] 中的研究发现，经过上述的诸多修改之后，LF-MMI 模型可以在无初始化情况下直接训练得到。而更多的研究显示，LF-MMI 模型的搭建步骤还可以进一步简化 [85]。我们近期的一些初始研究表明，LF-MMI 显示出了端到端系统的一些特性，比如可以直接对音素或字母进行建模。文献 [85] 将 LF-MMI 模型也归类为一种端到端模型。

### 2.4.2 连接时序分类模型

对于 DNN-HMM 混合模型而言，我们通常选取帧层面的交叉熵函数作为代价函数对 DNN 模型进行更新，这个过程中预先需要由 GMM 模型提供训练语料的帧层面的对齐。连接时序分类模型（Connectionist Temporal Classification, CTC）提出自动学习输入帧特征序列与其对应标签序列之间的对齐关系。

我们可以预先定义训练集的标签序列包含  $K$  个不同标签，这其中还包含意味着无发射标签的空符号标签 `blank` 建模单元。我们定义输入帧特征序列以及相对应的标签序列。这样我们就可以通过在给定输入序列的条件下最大化标签序列的对数似然比，对预测模型进行更新。

然而由于标签并不是对齐到帧层面的，因此我们无法直接计算该似然比。为了建立前端预测模型输出与标签序列之间联系，CTC 模型被定义为标注序列在给定特征序列后的后验概率  $P(\mathbf{L}|\mathbf{O})$ 。CTC 引入上述 `blank` 状态来对标签之间的混淆区段进行建模。具体来说，模型在标注  $l_{i-1}$  和  $l_i$  之间，通常预测出 `blank` 标注。

$$P(\mathbf{L}|\mathbf{O}) = \sum_{\mathbf{q} \in \mathcal{B}(\mathbf{L})} P(\mathbf{q}|\mathbf{O}) = \sum_{\mathbf{q}} \prod_{t=1}^T p(q_t|\mathbf{O}) \quad (2-82)$$

公式中  $\mathcal{B}$  是一个一对多的映射函数<sup>1</sup>：

$$\mathcal{B} : \mathbb{L} \mapsto \mathbb{L} \cup \{\text{blank}\} \quad (2-83)$$

$\mathcal{B}$  决定了标签序列  $\mathbf{L}$  和它相应由建模单元所组成的模型状态序列  $\mathbf{q}$ 。这一映射做法是在每个组成  $\mathbf{L}$  的  $l$  的标签中，插入一个可选并且可重复的 `blank` 标签。 $P(q_t|\mathbf{O})$  是由深度学习模型直接根据给定的特征序列  $\mathbf{O}$  来估计得出的，比如使用 LSTM [67]。

### 2.4.3 基于注意力机制的序列到序列建模

另一类端到端模型称为序列到序列模型。下文讨论将以基于注意力机制的序列到序列模型（encoder-decoder 模型）[78] 为主进行介绍，该类模型也是当前主要研究热点。该模型预测了在直接给定特征序列  $\mathbf{O}$  下的标签序列和之前已经预测出的标签历史序列  $\mathbf{L}_{1:i-1}$  后验概率。

$$P(\mathbf{L}|\mathbf{O}) = \prod_i P(l_i|\mathbf{O}, \mathbf{L}_{1:i-1}) \quad (2-84)$$

<sup>1</sup> 在最早文献 [43] 中使用的公式定义为一个多对一映射函数，同时使用它的反函数来定义  $\mathcal{B}^{-1}(\cdot)$  表示 CTC 模型中的映射。这里我们修改为一个一对多映射函数，使公式与 HMM 中的定义更加一致。

$$\mathbf{H} = \text{Encoder}(\mathbf{O}) \quad (2-85)$$

$$P(l_i | \mathbf{O}, \mathbf{L}_{1:i-1}) = \text{AttentionDecoder}(\mathbf{H}, s_{i-1}) \quad (2-86)$$

在公式中， $\mathbf{H}$  表示编码器（encoder）网络的隐层状态。 $s_{i-1}$  表示解码器（decoder）的上一输出相应隐层状态。Encoder( $\cdot$ ) 通常是一个单向或者多向的 LSTM 网络，同时 AttentionDecoder( $\cdot$ ) 通常是一个单向 LSTM 网络。

与传统混合系统 [77] 相比，该模型所使用的 AttentionDecoder( $\cdot$ ) 隐含地建模了语言模型信息，并将其与 Encoder 所组成的声学模型进行联合训练。这种建模方式的优势包括：基于深度学习的历史建模，更强的序列建模能力，以及更好的系统联合训练。除此之外，得益于该模型融合了语音识别的所有知识源，由此模型训练和推理搜索的系统设计都将非常简洁。但是由于该模型中的声学模型和语言模型进行了紧密的联合训练，这种端到端系统不容易被自适应到新的领域或者上下文环境中。与之相反，传统系统则可以通过直接修改语言模型而进行领域自适应 [86]。这是该模型的一个重要劣势，也是目前的研究热点之一 [87]。

## 2.5 本章小结

在这章中我们简要介绍了语音信号的识别的基本内容，首先讨论了特征提取，包括 MFCC, PLP, FBANK 等三种语音信号的识别中常用特征；接着介绍了语音信号的识别中最成功的声学模型——HMM 以及利用最大似然估计和序列鉴别性准则来优化 HMM 模型；继而讨论了现实语音信号的识别系统中的声学单元和参数绑定；最后讨论了 N 元语言模型和 Viterbi 解码等内容。

紧接着我们回顾了语音识别中最常用神经网络结构，主要包括深度神经网络、卷积神经网络、循环神经网络。接着探讨了神经网络的训练方法，包括训练准则、反向传播算法以及一些实际优化细节。随后回顾了深度神经网络-隐马尔科夫模型混合系统，这是目前语音识别领域最有效的将 DNN 与 HMM 结合的办法。接着引出了目前研究的热点，即端到端语音识别系统。我们系统介绍了端到端建模各个变种，它们的特点及优势，为之后几章改进和推理搜索框架研究进行了铺垫。

## 第三章 语音识别中的解码搜索问题

本章中，我们首先介绍基本的维特比解码搜索算法，而后引入其在语音识别中的应用。我们对语音识别中的解码搜索问题及其当前解决方案进行了详细介绍。这些不同的技术流派包括同步和非同步之分，以及动态和静态之分。我们对近十年主流的加权有限状态机技术进行了重点介绍。基于上面的介绍，我们总结了当前解码器研究中的待解决问题，以及上一章中深度序列学习模型所带来的研究机遇。后续本论文的一系列工作将基于本章针对解码器的分析，讨论和总结。

### 3.1 维特比解码搜索算法

维特比算法是一种动态规划算法，用于寻找维特比路径，其对应于最有可能产生观测事件的序列，即隐含状态序列。这一算法最早由安德鲁·维特比 (Andrew Viterbi) 在 1967 年提出，用于在数字通信链路中解卷积以消除噪音。此算法被广泛应用于 GSM 和 CDMA 数字蜂窝网络、拨号调制解调器、深空通信、卫星和 802.11 无线网络中的解卷积码。另一大类应用为计算语言学和生物信息学，以及语音识别和关键字检测。例如在语音识别中，声音信号作为观察到的事件序列，而文本字符串，被看作是隐含的产生声音信号的给定原因，因此可对声音信号应用维特比解码搜索算法寻找最有可能的文本字符串序列。

假设给定隐式马尔科夫模型 (HMM) 状态空间  $S$ ，初始处于状态  $i$  的概率  $\pi_i$  以及转移概率  $a_{i,j}$ ，其对应于从状态  $i$  转移到状态  $j$  的概率。同时我们观测到输出序列  $\mathbf{o}_1, \dots, \mathbf{o}_T$ 。其对应的产生该观察序列的相应隐状态  $s_1, \dots, s_T$  可以由递推关系得到。

$$V_{1,j} = P(\mathbf{o}_1 | j) \cdot \pi_j \quad (3-1)$$

$$V_{t,j} = \max_{i \in S} (P(\mathbf{o}_t | j) \cdot a_{i,j} \cdot V_{t-1,i}) \quad (3-2)$$

这里的  $P(\mathbf{o}_t | j)$  表示状态  $j$  的输出概率， $V_{t,j}$  表示前  $t$  步最终状态为  $j$  的观测结果最有可能对应的状态序列的概率。通过保存向后指针记住在第二个等式中用到的状态  $i$  可以获得维特比路径。声明一个函数  $\text{Ptr}(x_t, t)$ ，它返回若  $t > 1$  时计算  $V_{t,j}$  用到的上一个隐状态  $i$ ，这样：

$$s_T = \arg \max_{i \in S} (V_{T,i}) \quad (3-3)$$

$$s_{t-1} = \text{Ptr}(s_t, t) \quad (3-4)$$

该算法复杂度为  $O(T \times |S|^2)$ 。一个更好的估计存在于迭代只针对那些可以连接到当前状态的下一个状态当中，那么算法复杂度为，

$$O(T \times (|S| + |E|)) \quad (3-5)$$

其中的  $E$  表示图中所存在的边， $S$  表示图中所有的状态。

## 3.2 语音识别中的解码搜索问题

语音信号的识别既是一个模式识别问题，也包含相应的推理搜索问题。前一个问题对各种语音信号的、语言现象进行数学表示和描述，在基于统计学习的模式识别框架下进行建模，这决定了语音信号的识别系统可达到识别准确度的上限。而后一个问题在给定模型情况下，研究如何高效地将输入语音信号的和模型相匹配，推理搜索得到最优识别结果，这决定了识别速度和实际可达的识别准确度。在语音信号的识别的推理搜索阶段，解码器功能是对声学模型计算出的声学特征概率和语言模型计算出的语言概率进行组合来得到最大概率的词序列。

在语音识别中，声音信号作为观察到的事件序列，而文本字符串，被看作是隐含的产生声音信号的给定原因，因此可对声音信号应用维特比解码搜索算法寻找最有可能的文本字符串序列，称为语音识别的推理搜索阶段。如公式1-1所示，解码的过程是在给定声学模型和语言模型之后寻找拥有最大概率路径。

$$\mathbf{w}^* = \arg \max_{\mathbf{w} \in \mathcal{H}} p(\mathbf{O}|\mathbf{w})P(\mathbf{w}) \quad (3-6)$$

$$= \arg \max_{\mathbf{w} \in \mathcal{H}} \sum_{\mathbf{s}} P(\mathbf{s}|\mathbf{w})p(\mathbf{O}|\mathbf{s})P(\mathbf{w}) \quad (3-7)$$

这里  $\mathbf{s}$  是所有可能的状态序列， $P(\mathbf{w})$  通过语言模型计算， $P(\mathbf{s}|\mathbf{w})p(\mathbf{O}|\mathbf{s})$  通过声学模型计算。正如我们在章节??中讨论过的，这一似然可以通过前向后向算法进行计算。然而，

因为候选词序列可能性过于庞大，无法通过对每个候选进行前向后向算法来计算似然。因此，实际中通常利用最大概率的状态路径来近似整个概率，即：

$$\mathbf{w}^* = \arg \max_{\mathbf{w} \in \mathcal{H}} \max_{\mathbf{s}} P(\mathbf{s}|\mathbf{w}) p(\mathbf{O}|\mathbf{s}) P(\mathbf{w}) \quad (3-8)$$

这样我们可以通过利用动态规划 (Dynamic Programming, DP) 的方法计算出拥有最大概率状态序列，然后以此推理搜索出词序列，这一算法被称为 Viterbi 算法 [88]。另  $\phi_i(t)$  表示在时刻  $t$  且输出了部分从  $\mathbf{o}_1$  至  $\mathbf{o}_t$  声学特征后停留在状态  $i$  的最大概率。它可以利用递归来计算：

$$\phi_i(t) = \max_j \{\phi_j(t-1) a_{j,i}\} b_i(\mathbf{o}_t) \quad (3-9)$$

这里  $a_{i,j}$  是状态转移概率， $b_i(\mathbf{o}_t)$  是状态输出概率。最终：

$$p(\mathbf{O}|\mathbf{w}) \approx \phi_N(T+1) = \max_i \{\phi_i(T) a_{i,N}\} \quad (3-10)$$

Viterbi 算法可以很容易扩展到连续语音信号的识别，一种扩展后的 Viterbi 算法也被称为令牌传递 (Token Passing) 算法 [89]。在这一算法中，每一个状态将不仅仅保留一个最优路径，而是保存一部分个令牌。每一个令牌记录着某个候选词序列在第  $t$  时刻到达状态  $i$  的最优路径。在到达一个词边界时，语言模型的分数会直接加上。在整个观测序列结束时，拥有最大概率的令牌将被提取出来回溯其整个 HMM 序列。

即使利用了令牌传递算法，在大词汇连续语音信号的识别中，传播所有令牌搜索代价依然十分庞大。为了减少计算代价，通常会利用基于集束搜索 (beam search) 剪枝方法。在这一方法中，所有  $\phi_i(t)$  低于  $i$  中最大概率减去一个阈值的令牌都会被移除，这一阈值也被称为集束 (beam) 宽度，剪枝也可以在加完语言模型之后进行。虽然 beam 搜索方法可以显著减少计算量，但是有可能在早期阶段利用了过窄的 beam 宽度而剪去了真正最优路径导致识别错误。所以，beam 宽度设置需要在减少计算量和提升识别准确率之间进行均衡。

语言分数和声学分数之间动态范围的区别也是需要考虑问题，通常会分别对声学分数和语言分数进行缩放。另外需对插入错误添加额外惩罚，从而均衡词错误率中的插入和删除错误比例，因而最终语音信号的识别中利用的准则是：

$$\mathbf{w}^* = \arg \max_{\mathbf{w} \in \mathcal{H}} \{\log p(\mathbf{O}|\mathbf{w}) + \alpha \log P(\mathbf{w}) + \beta L_{\mathbf{w}}\} \quad (3-11)$$

这里  $\alpha$  是语言模型分数缩放系数， $\beta$  是插入错误惩罚系数， $L_{\mathbf{w}}$  是词序列长度。通常会选择最大概率路径作为识别结果，也可以输出一部分候选结果然后用更强的语言模型来重新评价（比如利用神经网络语言模型）。这一候选的路径一般利用 N-best 列表 [90] 或者利用能包含更多信息的词级词图 (Word Graph/Word Lattice) [91]。

在语音信号的识别推理搜索阶段，解码器是语音信号的识别系统的核心和灵魂，所有信息都汇集于此。它将不同来源、不同层次、不同性质的知识和信息关联在一起，使它们互相之间取长补短，从而得到正确的语音信号的识别结果。因此，如何将各种性质不同的信息有机融合是解码网络和解码算法设计中必须认真研究和解决的重要问题之一。从解码器的作用来看，它不仅是验证语音信号识别中各种理论，模型和算法正确性的基本实验平台，也是构建实际系统的基础。因此，在解码器设计中，还必须考虑到研究的便利性和工程的实际应用。近些年来，基于加权有限状态机 (Weighted Finite-State Transducer, WFST) [36] 的搜索解码器被愈加多的人采用，它的优点是可以将语言模型提前地融合成一个统一搜索网络，从而在解码时加快解码速度。

语音识别系统可以分为关键词检测，孤立词识别，语法网络识别，大词汇连续语音识别和基于神经网络语音模型的大词汇连续语音识别，下面我们将对它们中的解码搜索问题进行逐一介绍。

### 3.2.1 关键词检测

关键词检测是语言识别最主要的应用之一，它的目标是得到一个高准确度和高效率的识别器，用于检测特定的一些关键词在语音中的出现。KWS 可以被用于声学数据挖掘 [92]，低资源的音频检索 [93]，语料库检索 [94] 和唤醒词识别 [95]。

#### 3.2.1.1 关键词检测模型的分类

KWS 技术可以被分为两大类：i) 非监督的 *query-by-example* (QbyE) [96, 97, 98]，它使用了关键词的声学样例来产生一系列模板，之后尝试将模板与测试音频样例进行匹配来决定是否被唤醒。ii) 监督的基于文本的方法，这部分可以被进一步分类为基于大词汇连续语音识别 (LVCSR) 方法 [94, 99] 和基于声学的关键词检测 (声学 KWS) [100]。对于前者，在训练阶段需要构建一个词，半词或音素的语音识别系统。声学和语言模型在测试阶段一起对语音进行转录并得到词图。之后在词图上进行关键词搜索得到最终的结果。声学 KWS 不需要语言模型，通过直接建模目标关键词，半词或音素，构建一个声学模型来进行关键词的判别。一些方法中也会考虑引入非关键词元素到建模当中 [101]。QbyE 主要用于低资源音频检索，本文不针对这个方向。在语料库检索中，基于 LVCSR 方法通常能得到更好的性能，相比声学 KWS。但是基于 LVCSR 方法也有如下一些缺点：需要训练一个非常通用的大词汇连续语音识别系统，因此需要大量语料；同时在测试阶段也需要非常大的词表进行覆盖。这样的缺点限制了它在一些应用，比如唤醒词识别中的应用。除此之外，基于 LVCSR 的方法忽略了 KWS 任务本身的一些特性，该类方法

的推理搜索阶段解码搜索算法主与大词汇连续语音识别中类似，详细讨论参加第3.2.2章节。

### 3.2.1.2 基于声学的关键词检测

基于声学的关键词检测模型通常是要最小化每一帧的分类错误。在深度学习的 HMM 混合系统 (NN-HMM) 中，使用的是 tri-phone state 作为建模单元，而深度学习模型则用来对每个 HMM 状态的后验概率进行建模。特别是，对特征输入  $\mathbf{o}_{ut}$  关于时间  $t$  在句子  $u$  中，定义  $y_{ut}(s) = P(s|\mathbf{o}_{ut})$ ，表示神经网络在 HMM 状态  $s$  上的相应输出。这里的公式类似于 GMM-HMM 系统 [102]，除了需要使用伪对数似然度  $\log p(\mathbf{o}_{ut}|s)$  来针对 HMM 状态  $s$ 。

$$p(\mathbf{o}_{ut}|s) \propto \frac{y_{ut}(s)}{P(s)} \quad (3-12)$$

公式中  $P(s)$  是状态  $s$  的先验概率。由此在训练阶段，这里要最小化每一帧的 HMM 状态分类错误；在测试阶段则根据等式 (3-8) 利用这些逐帧的 HMM 状态概率进行搜索，得到最终的关键词序列。值得注意的是等式 (3-8) 中的  $P(\mathbf{w})$  在该问题中可以使用各个关键词序列出现的先验概率进行定义。

在深度学习系统中，也可以直接对关键词进行建模 [95]，那么关键词  $w$  的后验概率是要被直接训练得到的：

$$P(w|\mathbf{o}_{ut}) = y_{ut}(w) \quad (3-13)$$

这里通常可以使用负对数后验概率来作为交叉熵的目标函数。

$$\mathcal{F}_{\text{CE}} = - \sum_u \sum_t \log y_{ut}(s_{ut}^{(r)}) \quad (3-14)$$

公式中  $s_{ut}^{(r)}$  是在时间  $t$  句子  $u$  上的标注，通过状态级强制对齐算法来得到 [53]。

在声学关键词检测中，模型通常是进行逐帧分类的。解码方法用于解决模型所造成的误唤醒问题。解码的搜索空间通常包含两个部分：领域内搜索空间和领域外搜索空间。前者由关键词序列得到，而后者通常可以由特定模型进行建模（比如后文将提到的 *filler* 模型）领域外的搜索空间。另一方面如果 KWS 建模不是直接针对关键词，而是针对关键词的子序列，则搜索空间还需要建模子序列与目标关键词之间的映射关系（比如词典和发音序列）。

### 3.2.2 孤立词识别、语法网络识别和大词汇连续语音识别

孤立词识别是指对用户单个词语孤立发音的语音识别。在特定人孤立词语识别中，最为简单有效的方法是采用动态时间规整 (Dynamic Time Warping, DTW) 算法，该算法基于动态规划的思想，解决了语音长短不一的模板匹配方面的问题，是语音识别中出现最早且较为经典的一类算法。这类算法依然基于动态规划和维特比搜索解码。DTW 通过把时间序列进行延伸和缩短，来计算两个时间序列性之间的相似性。其计算遵循一定的搜索规则，根据语音信号在时间上的连贯性，认为所有路径均是从第一帧出发，并在最后一帧和最后一个输出上结束。由于两序列之间不等长，所以需要在匹配过程中限定弯折率来实现。最优路径为在满足约束条件下，计算路径累计距离的最小值。

孤立词的语音并不是人类沟通的自然方式。为了建模人类的自然语言，一种方法是使用语法网络来进行语音识别。语法网络可以规定词语按照什么规则来合成句子，即概率式句法结构。在该网络中，每个句子由若干词条组成，每一个词条都选自词汇表。句子中一个要选择的词条以一定的概率出现，而选择第二个词条的概率与前一个词条出现的概率有关，以此类推，直到句子的结束。在此框架里，每一个词语由若干个音素串接而成。而后每一个音素用一个 HMM 模型以及一套参数来表示。每一个 HMM 模型中最基本的构成单位是状态与状态之间的转移弧以及每个状态的概率建模。这样，从 HMM 状态出发逐层扩大到音素、词语、句子。每一个句子是包含许多状态的复杂的状态图，该句子就是用所有状态形成的结构、状态之间的转移概率，以及每个转移弧产生某个特征输出的概率来描述。对于特定的词表和句法，所有可能出现的句子构成一个更大的状态图。在完成识别任务的时候，要根据一个输入语音特征矢量序列来确定一个最可能的句子。这就需要在这个大的状态图中搜索一条路径，该路径上产生上述特征矢量的概率最大，有路径可以进一步确定句子中的每一个词。这种路径搜索就是采用维特比搜索算法。

语音识别中更具有挑战性的研究课题则是大词汇量、非特定人的连续语音识别，称为大词汇连续语音识别。大词连续语音识别中，语音信号先经过分析后形成特征矢量，并按隐马尔科夫模型，上下文音素，字典和词模型，语言模型结合而成的搜索空间进行识别，最后得到相应的句子。

与关键词检测和孤立词识别相比：传统语音识别模型要针对整个短语序列进行识别显然是不可能的，因为语言中短语的数量太多，所以必须把输入的语流切分为更小的组分来进行，相比较人类感知语音的过程也是类似的。由于连续语音中间没有间歇，所以在识别前必须先把各字拆分开，这就需要系统必须能够识别单词之间的边界关系。但这是非常困难的，因为确定单词间的边界位置还没有现成的方法来进行。尽管有时候可以采用能量最低点作为边界的判断准则，但是通常要根据发音信息加以辅助验证。另一方

面，连续语音的发音比孤立词的发音跟随便，受协同发音的影响也就更为严重。除此之外，连续语音识别系统中的很多问题都与相应的语言学知识有关，特别是大词汇连续语音识别系统要更多地强调运用语言学知识来进行。

### 3.2.3 解码搜索问题复杂度分析

大词汇连续语音识别的解码过程复杂度非常高。解码搜索的复杂度与上述由隐马尔科夫模型一直到语言模型的各个层面相关。下面我们将举例子进行说明。首先在有限状态语法网络网络以及 unigram 和 monophone 情况下，其语法网络如图 3-1 所示。该语法网络在每帧上的近似的复杂度包括两部分：

1. 内部传递:  $W_{ug} \times P \times N$ , 其中  $P$  是每个词的平均音素数量,  $N$  是每个模型的状态个数
2. 外部传递:  $W_{ug} \times P + W_{ug}$

而扩展到 unigram 的情况，在每一个词的开始，将 unigram 的概率加到令牌的对数似然度上，使得语言模型的信息尽可能早地合并进来。

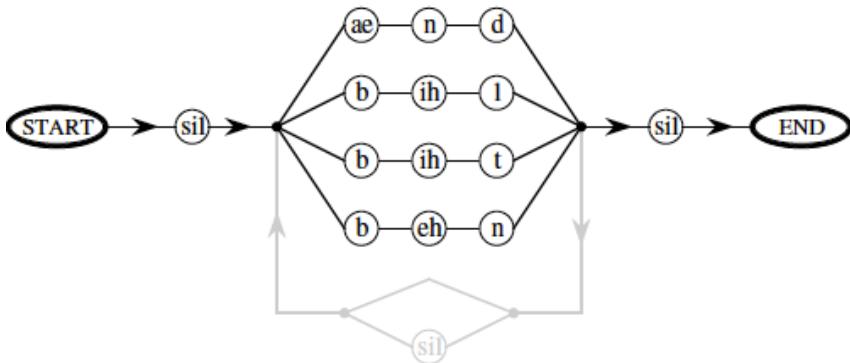


图 3-1 有限状态语法网络网络以及 unigram 和 monophone  
Fig 3-1 Finite State Grammar Newwork with unigram and monophone

而如果扩展到 Bigram 和 monophone 情况下，其语法网络如图 3-2 所示。在每帧上的近似的复杂度则包括内部传递:  $W_{ug} \times (P + 1) \times N$  和外部传递:  $W_{ug} \times P + W_{ug}^2$ 。而如果使用回退 back-off 语言模型，词传递大致是  $W_{bg} + W_{bo} \times W_{ug}$ 。

再扩展到 Bigram 和 cross-word triphone 情况下，其语法网络如图 3-3 所示。在每帧上的近似的复杂度则包括内部传递:  $W_{ug} \times (P - 2 + P_{prefix} + P_{suffix}) \times N$  和外部传递:  $W_{ug}^2 + W_{ug} \times (P - 2 + P_{prefix} + P_{suffix})$ 。

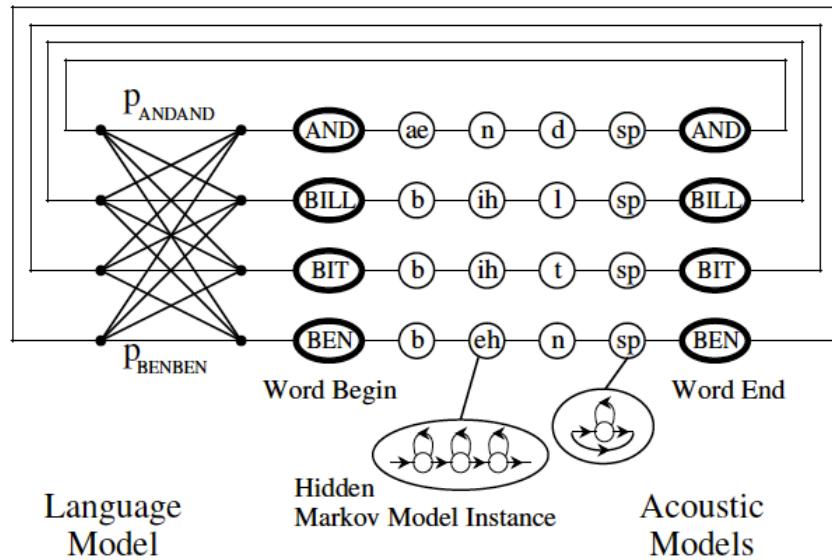


图 3-2 Bigram 和 monophone 的语法网络

Fig 3-2 Finite State Grammar Newwork with Bigram and monophone

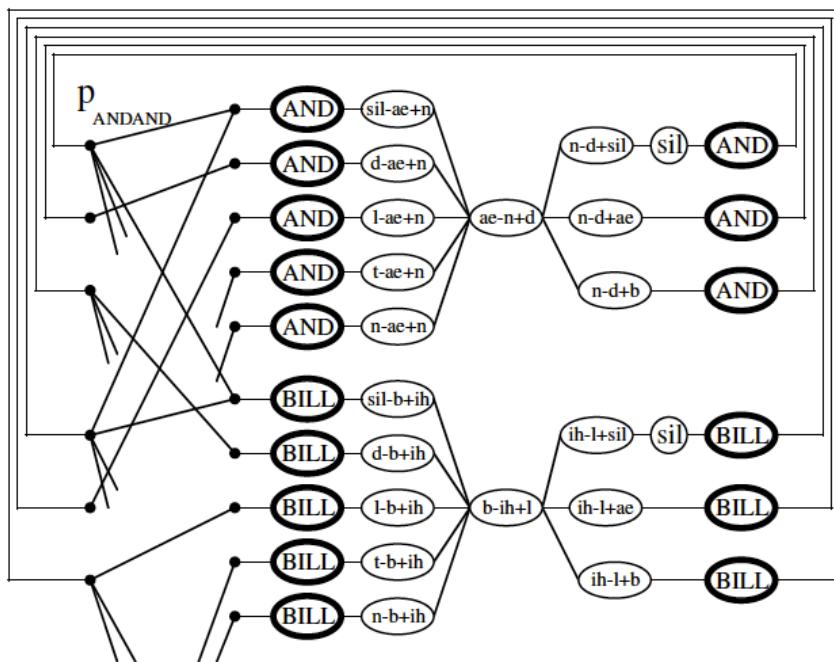


图 3-3 Bigram 和 cross-word triphone 的语法网络

Fig 3-3 Finite State Grammar Newwork with Bigram and cross-word triphone

如果使用 Trigram 和 within-word triphone，其语法网络如图 3-4 所示。在每帧上的近似的复杂度则包括内部传递:  $W_{ug}^2 \times (P + 1) \times N$  和外部传递:  $W_{ug}^3 + W_{ug}^2 \times (P + 1)$ 。如果是回退 back-off LM，词传递大致是  $W_{tg} + W_{bo}^{bg}(W_{bg} + W_{bo}W_{ug})$ 。

目前业界最主流的系统使用 trigram 语言模型和跨词三音子 triphone 声学模型，该情况下，解码复杂度会变得甚至更大。

总的来说，解码器的搜索空间复杂度与每个词的平均音素数量  $P$ ，每个 HMM 模型的状态个数  $N$ ，词表大小  $W_{ug}$ ，ngram 历史个数为  $W_{ng}$ ，其复杂度  $C$  大致为

$$C_{space} = O(P \cdot N \cdot W_{ug}^2) + O(W_{ng}) \quad (3-15)$$

另一方面，在传统解码算法中使用第 3.1 章介绍的维特比搜索，其需要在每一帧进行一次完整的搜索空间搜索过程，并由于是动态规划算法，只需要保留上一帧搜索空间内各点的最优结果。因此如果定义搜索空间内每个状态的平均入度为  $I$ ，结果前文分析和公式 (3-5)，则最终的复杂度如下，

$$C = O(T) \cdot C_{frame} = O(T) \cdot (O(I) \cdot C_{space}) \quad (3-16)$$

商用系统中  $W_{ug}$  和  $W_{ng}$  的数量都极其庞大，分别达到百万级别和百亿级别；而  $T$  往往可以达到数千帧。根据公式 (3-16)，这里的复杂度仍然非常大，因此在下一章节中我们将介绍如何在复杂度如此之高的搜索网络上进行解码搜索。

### 3.3 解码器技术综述

#### 3.3.1 解码器的技术流派

##### 3.3.1.1 静态和动态的搜索空间构建方法

根据前面章节讨论，完整的语音信号识别系统由自下而上的五层映射关系组成：语音信号  $\mathbf{O}$  到 HMM 状态的观察  $\mathbf{s}$ ，HMM 状态到依赖于上下文的音素  $\mathbf{c}$ ，依赖于上下文的音素到单一音素  $\mathbf{l}$ ，音素到单词  $\mathbf{w}$ ，单词到句子。具体来说公式 3-8 可以进一步展开如下：

$$\begin{aligned} \mathbf{w}^* &= \arg \max_{\mathbf{w} \in \mathcal{H}} \max_{\mathbf{s}} P(\mathbf{s}|\mathbf{w})p(\mathbf{O}|\mathbf{s})P(\mathbf{w}) \\ &= \arg \max_{\mathbf{w}} \sum_{\mathbf{l}} \sum_{\mathbf{c}} \sum_{\mathbf{s}} p(\mathbf{O}|\mathbf{s}) \cdot P(\mathbf{s}|\mathbf{c}) \cdot P(\mathbf{c}|\mathbf{l}) \cdot P(\mathbf{l}|\mathbf{w}) \cdot P(\mathbf{w}) \end{aligned} \quad (3-17)$$

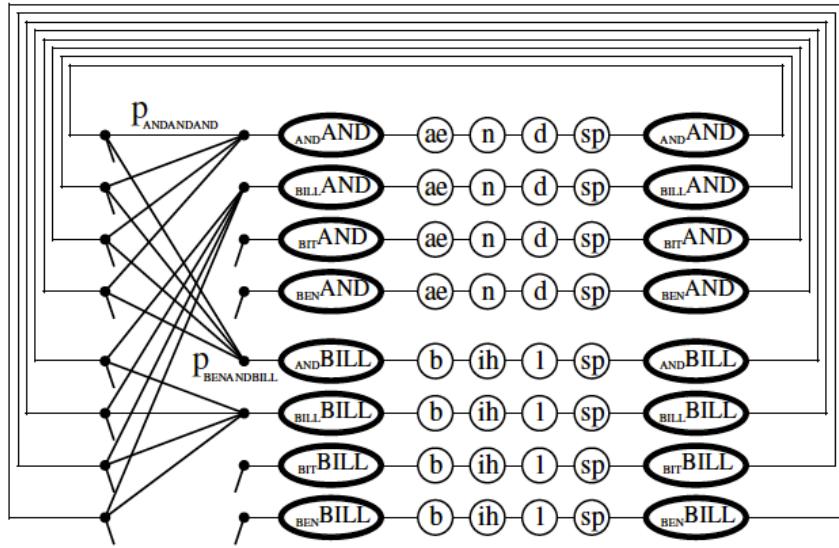


图 3-4 Trigram 和 within-word triphone 的语法网络

Fig 3-4 Finite State Grammar Newwork with Trigram and within-word triphone

其中展开后式子每一项分别对应上述的五层映射关系。在识别之前不能知道语音信号的观察，因此需要在解码过程中动态地建立观察到的语音信号的 HMM 状态的映射，并且还需要实时计算相应的声学分数。对于从 HMM 状态到依赖于上下文的音素，依赖于上下文的音素到音素，音素到单词的三级映射关系，一旦确定了发音字典  $\mathcal{H}$ ，语言模型  $P(\mathbf{w})$ ，和声学模型，它们就不会改变。为了追求解码效率，人们通常将它们静态编译成解码网络。对于没有任何约束的大词汇量连续语音信号的识别任务，可以以任何方式将单词组织成句子，因此原则上，单词 - 句子映射关系只能在解码过程中动态建立。然而，由于表征单词和单词之间的关联度（概率）的 N-gram 语法模型在解码之前已经存在并且是有限集，因此在实践中，语言模型得分计算可以以不同方式实现。在语音信号识别领域，根据解码器中语言模型的表示，语言模型状态获取和语言模型得分计算方法，解码器通常分为两大流派：

- 动态网络解码器：在动态网络解码器中，解码网络仅包含发音字典和声学模型，不会含有语言模型任何信息。语言模型状态在解码过程中还要随着词与词相连成句而动态地生成、语言模型分数通过去查表的方式动态地获取。这类解码器的典型代表是基于发音前缀树（Pronunciation Prefix Tree，PPT）[53] 网络解码器。
- 静态网络解码器：在静态网络解码器中，解码网络不仅包含发音字典和声学模型，也包含完整的语言模型分数。语言模型状态以及状态转移以 WFST 的形式合成到解码网络中去，语言模型分数则作为状态转移概率存储于边上。解码时，仅需逐

边积累整条路径状态转移概率便可获得语言模型分数。这类解码器的典型代表是基于加权有限状态转换器 [36] 的解码器。

作为一种时空变换策略，静态网络解码器通常可以通过仔细优化实现更快的识别速度，但缺点是构建的解码网络太大，特别是在高阶语言中。随着硬件设备的发展，存储器对解码网络大小的限制逐渐减少，本论文工作主要基于静态网络解码器，将在第3.3.2.2章节对其理论做进一步介绍。

### 3.3.1.2 同步和异步的搜索算法

除了根据解码网络的结构进行分类之外，还可以根据搜索最优路径的不同方式将解码器划分为时间异步搜索解码器和时间同步搜索解码器：

- **时间异步搜索解码器：**在较老的解码器，例如：IBM ViaVoice 中，采用深度优先方式在解码网络中搜索最优词序列，是一种时间异步搜索。由于这类解码器在解码过程中需要用到一些后进先出的缓冲区（即：堆栈）来保存扩展得到词识别假设，所以也常常被称为堆栈解码器（Stack decoder）[103]。与时间同步搜索相比，优点在于通过选择适当的启发函数，搜索可以控制于最佳路径附近，并且提高了搜索效率。但也造成了它的主要缺陷：1) 启发函数综合声学和语言分数，有时需要“未来”路径信息，因此很难获得；2) 因为它是时间异步搜索，所以搜索过程产生的路径长度各不相同，使得修剪很难实现。因此，当前的堆栈解码很少用于直接解码语音信号的观察，而更多地用作用于从字图生成 N-Best 结果的后处理中。
- **时间同步搜索解码器：**时间同步搜索是目前解码器设计和实现的主流方法，也称为帧同步搜索。它使用广度优先方法找到与来自解码网络的输入特征序列最佳匹配的状态序列，以便获得相应的最佳音素序列和字序列。这种类型的解码器通常使用维特比算法或令牌传递算法 [53] 实现。

时间异步的搜索算法还包括多遍解码算法。多遍解码算法通常在每次解码过程中对之前的搜索空间进行剪枝，缩小空间大小，而后将压缩后的搜索空间交给下一次解码算法，进行进一步搜索，直到最终得到最佳结果。为了在搜索中利用各种各样的知识源，通常要进行多遍搜索，第一遍要使用代价较低的知识源，产生一个候选序列列表或词候选网格，在此基础上再进一步进行使用代价高的知识源的第二遍搜索来得到最佳路径。这里的知识源包括声学模型、语言模型和音标词典，这些可以用于第一遍搜索。为实现更准确的语音识别，往往要利用一些代价更高的知识源，如高阶的上下文相关的 N-Gram

语音模型、词间混淆相关模型、分段模型或语法分析算法，进行重新打分。目前主流的实时大词汇连续语音识别系统许多都使用这种多遍搜索策略。与单遍搜索（1-pass）相比，多遍搜索具有优点和缺点，并且它也在实践中使用。多遍搜索的支持者认为：第一遍粗搜索可以大大减少解码空间，使得在后续解码过程中使用更精细的语言模型和声学模型成为可能，避免了单遍解码器中的时间和空间限制。。单遍搜索的支持者认为在多遍解码中存在三个主要问题：1) 由于第二遍解码必须等待第一遍解码，因此多遍搜索难以应用于实时解码。2) 每次通过解码引入了不可恢复的修剪误差，其不能被后续解码中使用的任何模型补偿。要解决这个问题，我们仍然需要努力研究单遍搜索算法。最好直接进行单遍解码器。3) 多次解码器的每次解码中使用的特征，声学模型，语言模型和搜索算法是不同的。

解码器不仅算法复杂，而且需要高工程能力和技能。在开发完成后，通常需要大量的人力和时间进行调整。因此，各种研究机构和商业公司都有自己的解码器实现细节。它就像它一样深。尽管如此，世界上仍有一些用于学术研究的开源解码器可以作为研究的基础。该表列出了比较知名的那些。这些解码器的网络结构和解码算法不同，但由于学术研究的目的，大多数速度优化都比较差。

表 3-1 知名开源解码器

解码器	研发机构	网络结构	搜索算法
HDecode	英国剑桥大学	动态，发音前缀树	单遍，令牌传递
Sphinx	美国卡内基梅隆大学	动态，发音前缀树	单遍，令牌传递
RASR	德国亚琛工业大学	动态，发音前缀树	单遍，Viterbi
Juilus	日本京都大学	动态，发音前缀树	两遍，前后向搜索。
Juicer	瑞士 IDIAP	静态，WFST	单遍，令牌传递
Kaldi	美国约翰霍普金斯大学	静态，WFST	两遍，令牌传递

### 3.3.2 主要模块和核心算法

#### 3.3.2.1 解码器框架

图 3-5给出了典型解码器结构。不论是哪种类型解码器通常都包含网络生成、分数计算、搜索、剪枝与路径管理等部分，它们的功能逐一介绍如下。

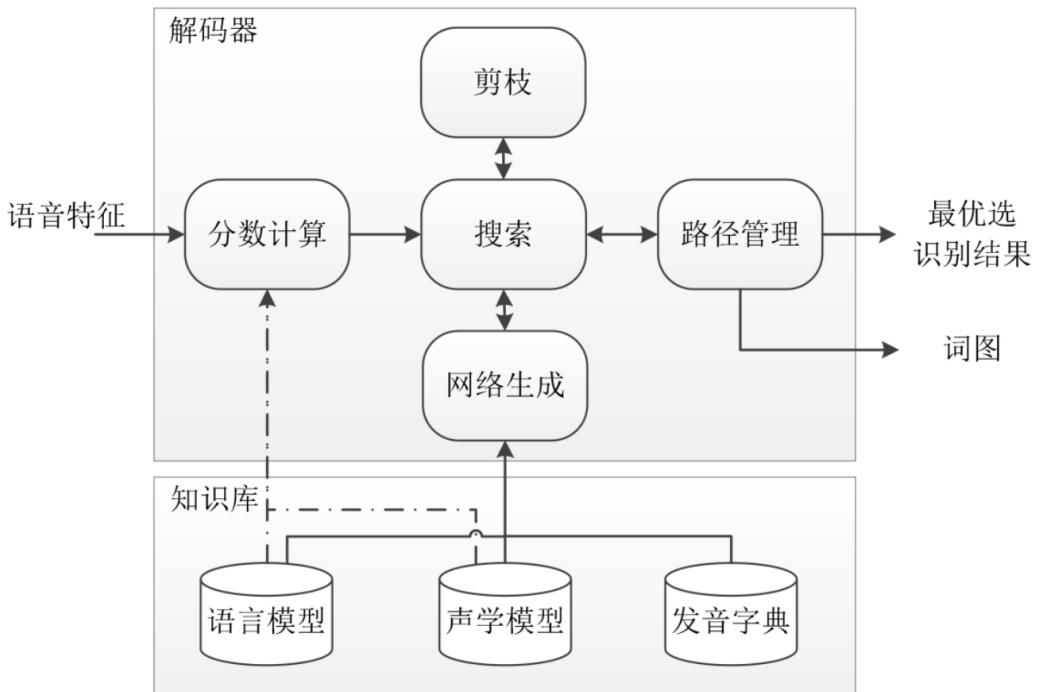


图 3-5 通用解码器架构  
Fig 3-5 General Architecture of ASR Decoder

### 3.3.2.2 基于加权有限状态机的网络生成算法

网络生成模块主要负责构建解码搜索空间。搜索空间（也称为解码网络）通常与音素 HMM 或 HMM 状态连接，从语言模型，发音词典，声学模型和其它相关知识源编译。大词汇量连续语音信号解码网络的识别系统由各种知识源组成，形成搜索空间，一般可分为动态构造的解码网络和静态网络。基于动态网络解码器，前缀树发音字典用作搜索网络。语言模型通过动态查询将得分引入解码过程，然后通过重新输入字典树或字典树副本来自搜索整个解码网络 [89]。动态网络解码器主要优势在于，由于字典和语言模型是分离，其占用内存极少，这个特点在以往移动网络技术和硬件技术不发达的时代，尤其是在嵌入式设备上内置解码器的时代，占有绝对优势。但是动态网络的缺点是它解码速度较慢，时间复杂度较高，这也使其愈加难以满足当前海量语音信号的识别请求。随着移动互联网和嵌入式设备的普及以及云技术的发展，语音信号的识别应用已发展为仅在嵌入式设备上保持简单的前端，而识别系统仍保留在服务器云中。此时，反映了基于 WFST 的静态网络解码器的快速优势。较短的识别时间允许服务器每单位时间接受更多的识别任务，因此静态编译的解码网络更适合于大量语音信号识别任务。

在 LVCSR 任务中，解码网络通常非常大。因此，网络结构需要采用各种手段来优

化网络结构，并在不改变网络功能的情况下尽可能地减少网络中的节点和边数量。解码网络是解码器的基础，并确定在解码器的其它部分中应该使用哪种方法。在静态网络解码器中，解码网络不仅包含发音字典和声学模型，还包含完整的语言模型。语言模型状态和状态转换以有限状态机的形式合成到解码网络中，并且语言模型得分被存储为边上的状态转移概率。当解码时，仅通过累积整个路径状态转移概率可以获得语言模型得分。这类解码器的典型代表是基于加权有限状态转换器 [36] 的解码器。

WFST 最开始由 AT&T 实验室 Mohri 和 Riley 等人在 1997 年引入语音信号的识别领域。并且在理论上发展了确定化、最小化等一系列网络优化算法，为语音信号的识别的应用打好了理论基础。具体来说，上文讨论的解码搜索问题基于公式 (3-17)，其中  $p(\mathbf{O}|\mathbf{s})$  由声学模型建模得到，语音识别的其它各模型组件分别用如下方式进行 WFST 构建：

- N-gram 语言模型在 WFST 中表示为  $\mathbf{G}$ ，在公式 (3-17) 中表示为  $P(\mathbf{w})$ 。根据 N-gram 语言模型的含义，它提供了单词历史的当前单词概率。语言模型需要具有记录最多  $N$  个单词的历史的信息。但是，当识别语音信号时（即，输入和输出是其符号集的单个元素），WFST 应用程序不允许使用字符串输入和输出。因此，不可能在  $\mathbf{G}$  构造过程的输入和输出上表达其历史，而是记录该状态的历史。另一个问题是 N-gram 模型的回退（Back-off），当语言模型训练语料未出现某一个  $N$  元词组的时候，就会以一个回退的 ( $N-1$ ) 元模型概率乘以一个系数来替换。这部分表示为一个带有着权重但是输入为空的状态跳转边。
- 字典在 WFST 中被表示为  $\mathbf{L}$ ，在公式 (3-17) 中表示为  $P(\mathbf{l}|\mathbf{w})$ 。字典实际上是发音规则的表示。最简单的方法是列出开始状态和结束状态之间每个单词的发音。同时，由于在与语言模型合成时连接了单词和单词，因此无法达到终止状态，并且需要将闭合边从终止状态连接到开始状态以形成闭环。针对字典的同发音问题，Mohri 在文献 [36] 中提出对于不同词的同发音发音序列加入一组辅助符号，这样对于同发词语，其环路的输入符号序列就不再相同，可以确保字典的可确定化。
- 上下文相关声学模在 WFST 中被表示为  $\mathbf{C}$ ，在公式 (3-17) 中表示为  $P(\mathbf{c}|\mathbf{l})$ 。一个上下文相关 triphone 模型一般表示为  $a-b+c$ ，其中  $b$  是中心音素， $a$  和  $c$  为其前后上下文音素 [76]。用 WFST 表示 triphone 模型实际上是在把三音素和单个音素之间进行对应，其输出就和发音字典的输入做相互对应，由此可以进行进一步合成。
- 隐马尔科夫模型在 WFST 中被表示为  $\mathbf{H}$ ，在公式 (3-17) 中表示为  $P(\mathbf{s}|\mathbf{c})$ 。HMM 模型天然具有多个的状态跳转特性，所以可以直接将其构建为多个 WFST 状态。

隐马尔科夫模型的转移概率于是被转化为 WFST 的权重。而 WFST 输入是隐马尔科夫模型的状态索引，输出是该隐马尔科夫模型所表示的 triphone 建模单元。

当各个知识源 WFST 组件被构建完毕以后，可以利用 WFST 合成算法和优化算法将各组件最后进行合并和最终优化。如下：

$$HCLG = \min(\det(H \circ \min(\det(C \circ \min(\det(L \circ G)))))) \quad (3-18)$$

最终生成的 WFST 包含了所有的知识源，它是一个输入为 HMM 状态  $\mathbf{s}$ ，输出为词语  $\mathbf{w}$  的 WFST 网络，用于建模  $p(\mathbf{s}|\mathbf{w}) \cdot p(\mathbf{w})$ 。结合声学模型建模的  $p(\mathbf{O}|\mathbf{s})$  概率，后文所讨论维特比搜索就在该 WFST 网络上面进行。

### 3.3.2.3 分数计算

分数计算模块计算输入特征序列声学和语言分数，并将它们提供给搜索模块以供使用。应该计算哪个分数与解码网络结构和搜索算法有关。例如，在静态网络解码器中，语言模型得分已经被编译到解码网络中，因此只需要计算声学得分；对于动态网络解码器，除声学分数外，还需要计算语言模型分数。分数计算的研究主要集中在如何实现分数的快速计算。对于声学分数，通常执行以下三个方面：

- 硬件加速。利用 CPU 矢量计算器 [104] 或者图形处理器（Graphics Processing Unit, GPU）[105] 加速分数计算。这类方法通常来说不会带来计算误差（与硬件实现相关），所以对识别准确度没有影响。
- 模型简化。为了减少声学模型推理的计算开销，研究人员们提出了一系列效率更高的声学模型，包括一些比较新颖的神经网络结构 [106, 107]，跳帧 [108, 109, 110] 和端到端系统 [111, 112]。这还包括采用复杂度更小、参数更少的分类器模型，或通过聚类和参数共享减小模型复杂度。例如采用半连续 HMM [113] 替代连续 HMM，以及传统的参数共享方式 [89] 亦属此类。这类算法一般都会带来识别准确度上面的损失，所以需要在识别准确度、模型复杂度和计算速度之间做好一定的权衡。
- 算法优化。对声学分数的计算过程进行简化。这类方法一般是对声学分数进行近似计算，所以必然会带来一定识别准确度损失。一般而言，衡量一个近似算法是否可用的标准是看其带来相对识别准确度损失是否可控制在 5% 以内 [114]。

对于语言分数计算，当采用 N 元文法模型时，通常从两个方面进行加速：1) 减小每次查表操作上面的耗时，典型方法是采用最小完美哈希（Minimal Perfect Hash, MPH）表实现语言模型的存储与查取 [115, 116]；2) 引入分数缓存 [117] 减少查表熵的次数。

### 3.3.2.4 帧同步的维特比解码算法

在大词汇连续语音识别中，解码算法的目标是找到最佳的词序列。通过应用字典和语言模型将词序列映射到标签序列，解码公式可推导如下：

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} \{P(\mathbf{w})p(\mathbf{x}|\mathbf{w})\} = \operatorname{argmax}_{\mathbf{w}} \{P(\mathbf{w})p(\mathbf{x}|\mathbf{l}_w)\} \quad (3-19)$$

其中， $\mathbf{w}$  是词序列， $\mathbf{w}^*$  是最佳词序列。 $\mathbf{l}_w$  表示  $\mathbf{w}$  通过映射得到的标签序列，如 NN-HMM 系统中的上下文相关音素。

以 CTC 为例：

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} \left\{ \frac{P(\mathbf{l}_w|\mathbf{x})P(\mathbf{w})}{P(\mathbf{l}_w)} \right\} \quad (3-20)$$

$$= \operatorname{argmax}_{\mathbf{w}} \left\{ P(\mathbf{w}) \max_{\mathbf{l}_w} \frac{P(\mathbf{l}_w|\mathbf{x})}{P(\mathbf{l}_w)} \right\} \quad (3-21)$$

这里以单音素的 CTC 为例 (CTC 标签符号集合包括音素标签和 blank 符号)。 $P(\mathbf{l}_w)$  是音素序列的先验概率。对于某个特定的 CTC 标签序列，其前向概率可定义 [43] 并近似为：

$$P(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}(\mathbf{l})} \prod_{t=1}^T y_{\pi_t}^t \cong \max_{\pi \in \mathcal{B}(\mathbf{l})} \prod_{t=1}^T y_{\pi_t}^t \quad (3-22)$$

其中， $\mathcal{B}$  的定义见公式2-83。因此，公式3-21可进一步被推导为如下的帧同步维特比束搜索 (Frame Synchronous Viterbi Beam Search, 简称为 Frame Synchronous Decoding, FSD)。这里，整体优化搜索空间-WFST，在每一帧都需要被遍历。

$$\mathbf{w}^* \cong \operatorname{argmax}_{\mathbf{w}} \left\{ P(\mathbf{w}) \max_{\pi \in \mathcal{B}(\mathbf{l})} \frac{1}{P(\mathbf{l}_w)} \prod_{t=1}^T y_{\pi_t}^t \right\} \quad (3-23)$$

在 FSD 框架中，将特征帧的数量除以语句的长度定义为特征速率，标签输出数量除以语句的长度定义为标注速率，而 WFST 解码的帧数除以语句的长度定义为解码速率。该框架中，三个速率均相等。也就是说， $\prod_{t=1}^T y_{\pi_t}^t$  与帧  $t$  有关，而最大迭代次数则与序列可能的对齐方式和词汇量的大小有关。因此，解码复杂度  $C_{FSD}$  可表示为：

$$C_{FSD} = O(T) \cdot C_{frame} \quad (3-24)$$

其中  $T$  是语句中帧的数量， $I$  和  $C_{frame}$  分别表示搜索空间内每个状态的平均入度和每一帧搜索解码的总复杂度，定义参见公式3-16。

### 3.3.2.5 令牌传递算法

解码搜索模块负责在解码网络上搜索得到最优路径。不论是静态网络解码器还是动态网络解码器，目前都以令牌传递算法 [118] 为主流的搜索算法。

令牌传递是上文讨论的帧同步的维特比算法的另一个更通用和简单的实现。在算法中，每个 HMM 状态可以与令牌（Token）相关联，令牌存储令牌所经历的历史路径和路径得分直到当前帧。解码是根据状态转换（即，沿着解码网络的边）将令牌从解码网络的初始状态传递到终止状态的过程。每个帧令牌向前传递一次，同时传递令牌得分和路径信息，并在多个令牌同时传递到状态时进行令牌合并，仅保留具有最高得分的令牌。在处理完所有帧之后，获得最佳路径和最佳路径得分。简化的算法流程如算法3-1所示。

---

#### 算法 3-1 令牌传递算法 (Inputs: 初始 Token; 声学分数矩阵; 总帧数)

---

```

1: procedure TOKEN PASSING(startTok, amScores, T)
2:   • 算法初始化
3:   Queue = [startTok]
4:   stateMap={};
5:   • 令牌传递
6:   for t in T do                                ▷ 每一帧
7:     for curTok in Queue do                  ▷ 每一个上一帧的 Token
8:       for arc in curTok.arcs do            ▷ 每个 Token 的出边
9:         amScore=amScores[t][arc.inId];
10:        nextScore=curTok.score+amScore-arc.weight;
11:        nextState=arc.nextState;
12:        • 令牌合并处理
13:        if nextState  $\notin$  stateMap then          ▷ 如果该 WFST State 无 Token
14:          newTok=Token(nextScore, curTok);
15:          stateMap[nextState]=newTok;
16:          if stateMap[nextState].score < nextScore then    ▷ 存在但分数小
17:            stateMap[nextState].score=nextScore;
18:            stateMap[nextState].prevTok=curTok;
19:        • 下一帧的预处理
20:        Queue=stateMap.toList();
21:        stateMap.clear();

```

---

令牌传递算法的优点是可以方便地在令牌上保存其它信息以便在状态节点之间进行传递，从而实现更复杂解码过程，例如：为了生成词图，在令牌合并后，不是将次优令牌直接丢弃，而是作为其它候选路径保存于合并后的令牌之上。

### 3.3.2.6 剪枝

观察公式 (3-16)，在 LVCSR 中，搜索空间太大而无法对整个解码网络执行完全搜索。剪枝用于消除解码过程中的低分数路径，从而减少计算量并确保不降低识别性能。根据解码器设计，剪枝可以应用于搜索的各个阶段（例如，开始或结束），或者可以应用于各种级别（例如，单词级别，音素级别，状态级别，令牌级别），但是可以归纳为三个基本类别：

- 束剪枝 (Beam pruning)：束剪枝保留与最优路径分数较近的次优路径。令  $v(t, j)$  表示第  $t$  帧位于状态  $j$  最优路径分数，则第  $t$  帧的最优路径分数为：

$$v_{max}(t) = \max_s v(t, s) \quad (3-25)$$

令  $f$  为剪枝阈值（又称为束宽），则束剪枝只保留所有满足下式的路径： $v(t, j) > v_{max}(t) \cdot f$

- 直方图剪枝 (Histogram pruning)：与束剪枝不同，直方图剪枝仅保留分数最高  $K$  条路径， $K$  为设定剪枝阈值。之所以称为直方图剪枝是因为该方法可以采用直方图统计高效地实现 [119]。
- 向前预测声学和语言分数 [120, 121]：通过算法的改进，合理地预测未来的声学和语言分数，并据此提前对可能性进行剪枝，也是一种主要的优化方式。

以直方图剪枝为例，由于每帧只需要考虑  $K$  条路径，并且上一帧只保留  $K$  条路径，因此如果定义搜索空间内每个状态的平均出度也为  $I$ ，复杂度公式 (3-16) 中的  $C_{frame}$  将显著减少如下，

$$C_{frame} = O(K \cdot I) \quad (3-26)$$

比如在语音识别中通常取  $K = 10000$ ，那么相比第3.2.3章节中在商用系统情况下，对公式 (3-16) 的分析，搜索解码的复杂度将下降百万倍。

### 3.3.2.7 路径管理和词图

路径管理模型主要作用是用于对搜索过程中得到的路径链表进行回溯，获取最优路径、N-Best 结果和词图。同时，它也负责对剪枝过程中所剪掉的较差路径进行内存回收等操作。

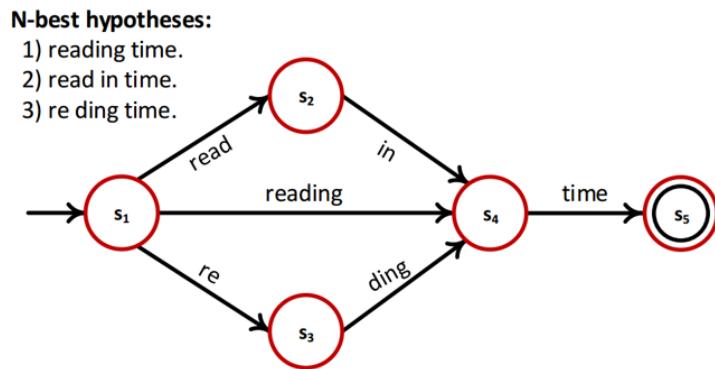


图 3-6 N-Best 候选序列和词图

Fig 3-6 N-Best Hypothesis and Lattices

由于 N-Best 候选序列中往往存在大量的公共部分（比如图 3-6 中的“time”这个词），因此将公共部分优化在一起进行表示，也就是将多个候选序列转变成一张图来进行存储，将会更加紧凑。在语音信号的识别中，这样一张图就称为词图。值得注意的是，在语音识别中，并没有对词图的统一定义 [122]<sup>1</sup>。本文主要强调词图在概念上是一种对搜索空间的紧凑表示。而词图的具体格式在本文中默认使用 Kaldi 格式。词图的生成所带来的额外消耗与生成 N-Best 候选序列类似。具体来说，其往往要求在解码过程中做令牌合并时，保留多个历史令牌记录。在记录完词图后，还需要词图剪枝步骤，其将冗余和无法连通边及其相应结点删去，得到最终比较紧凑的词图。

词图具有较多用途，比如：词图语言模型重打分，鉴别性训练，置信度计算，混淆网络生成等。因此相较于 N-Best 候选序列，词图的处理往往是语音信号的识别系统中的一个标准模块。

### 3.3.2.8 语音识别中的置信度

在最近十年时间，自动语音识别(ASR)取得了非常显著的成功。但是当语音识别系统由实验室产品转向真正的应用时，即使是最好的语音识别系统，仍然会不可避免地产

<sup>1</sup> 比如在 HTK 中，音素级别的时间边界对齐结果，语言模型分数，声学模型分数，和发音词典分数都被存储到词图当中。而在 Kaldi 中，则存储了 HMM 状态级别的时间边界对齐结果。

生一些识别中的错误 [123]，也就是说 ASR 系统的输出总会包含或多或少的一些各种各样的错误。因此，在真实的应用场景中，人们一般需要 ASR 系统能够自动评估自身的可靠性和发生错误的概率，而这些都由系统自身得出。

在语音识别中，置信度 (CM) 就是提出来进行这种类型的可靠度评估的 [124]。这类模型可以被分为如下几种类别：

- 基于用于建模的特征的置信度. 基于 ASR 识别过程的特征被称为用于建模的特征，而它的概率分布在识别正确和识别错误时会产生显著的不同。CM 可以由它们之中的一个或两个来组成，比如对归一化后的声学分数 [125]，时长 [126]，局部交叉熵 [127]。但是，这些用于建模的特征并不完美地表示解码的过程 [124]。所以一些分类器模型也可以考虑被加入到这些用于建模的特征上，比如 CRF [128]，深度学习 [129]，等等。但是这些模型仍然不够完美。首先各个用于建模的特征之间并不完全独立，其次它需要额外的训练阶段，并假设训练数据与测试数据匹配。
- 基于后验概率的置信度. 另一种方法将 ASR 过程建模为最大后验概率 (MAP) 的决策过程。给定整个句子后的 ASR 输出的后验概率可以用来表示 CM。许多方法研究了如何对归一化项进行建模，比如 filler 模型 [130]，词图 [131] 和混淆网络 (CN) [132]。然而，ASR 解码器通常设计用于寻找最优的首选路径，这使得它所得到的词图并不完美，同时会导致 CM 的建模并未较好地归一化 [133]。

## 3.4 解码器研究中的待解决问题

### 3.4.1 研究现状总结

语音识别技术虽然相比多年以前已经有了长足的进步，但是在实际应用中还有很多困难需要处理。其中一个最主要难题就是语音识别推理搜索问题。语音识别既是模式识别问题又是相应的推理搜索问题。前一个问题在数学上表示和描述了各种语音和语言现象，并且基于统计学习模式识别框架执行建模，该框架确定了语音识别系统可以实现的识别准确度的上限。在给定模型的情况下，后一问题研究如何使输入语音与模型匹配并推断最佳识别结果，其确定识别速度和实际可达到的识别准确度。在语音识别推理搜索阶段，解码器是语音识别系统的核心和灵魂，组合声学模型以计算声学特征概率和语言模型计算的语言概率以获得最大概率词序列。它将来自不同来源，不同层次和不同性质的知识和信息联系起来，以便它们相互补充并获得正确的语音识别结果。因此，如何有机地整合各种不同的信息是解码网络和解码算法设计中必须仔细研究和解决的问题。从解码器功能的角度来说，它不仅是语音识别研究中各种理论，模型和算法的验证。正

确性和准确性的基本实验平台也是构建实际系统的基础所在。因此，在解码器的设计中必须平衡研究的便利性和工程的实际应用。

近年来，深度学习模型被引入到语音识别声学和语言建模中以取代传统的分类器，显着提高了模式识别问题的准确性。在基于深度学习语音识别中，语音识别的推理搜索问题并没有得到根本改变。因为在该框架中，只有分类器得到了改进。基于加权有限状态机的推理搜索问题静态搜索空间构造技术 [36] 和帧同步维特比 (Viterbi) 网络搜索算法 [37] 仍是目前性能最好的解决方案。该类系统存在的一系列显著缺陷包括：

1. 目前，语音识别系统基于多知识源建模结果，并对输入音频进行推理搜索。建模和推理搜索过程非常复杂，知识来源的划分依赖于强大的先验知识。大规模或非标记语音数据收集以及基于并行的深度学习技术使得构建直接模拟语音数据和文本序列的端到端模型及其相应的识别和推理搜索算法成为可能。当前的解码框架没有这种设计。
2. 该种方案基于传统的混合高斯-隐马尔科夫模型 (GMM-HMM) 的声学模型和 N 元文法 (N-gram) 的语言模型而提出，针对目前性能最好的基于深度学习的声学模型和语言模型的研究并不充足，如何将新型的声学和语言模型引入该框架；如何充分发挥模型性能的同时改善推理速度；如何基于多知识源给出可靠的推理置信度算法，都是有待解决的问题。
3. 该种方案基于对语音识别中各知识源 (声学、语言、语义等) 进行搜索空间的预先构建及整体优化，导致最终得到的搜索空间巨大，包括离线构建、在线使用、动态修改等各环节算法的计算量和内存消耗都非常大，是阻碍语音识别应用场景扩展的一个重要原因。旨在解决该问题，针对新型声学和语音模型的搜索空间整体优化研究尚不充分，而基于推理中间状态对搜索空间进行动态优化的研究几乎处于空白。

因此，尽管基于深度学习模型，加权有限状态机和基于帧的维特比网络搜索算法的深度学习已经发展到基本可用水平，但是精度仍然不能满足人类之间的正常交流要求。速度限制也使得语音识别无法在低成本和低功耗的解决方案上工作，这些解决方案一起阻碍了语音识别技术的大规模商业应用。

### 3.4.2 发展机遇和亟待解决问题

1. 基于 GPU 并行计算提高解码搜索速度。

随着深度学习的发展和计算设备的更新，基于 GPU 的并行计算是一种潜在的针对语音识别计算的加速方式。针对声学模型推理，由于大部分计算量集中在矩阵形式的运算，因此它易于通过将类似于训练过程中 [134] 的序列批处理 [135] 引入模型推理阶段，以加速运算速度。但是，基于 GPU 并行计算的解码搜索不容易实现。本质上它不再是一个易于并行的矩阵运算问题，而是一个图上最优路径的搜索问题。尽管一些研究在小型语言模型上取得了一定成功 [136]，但这些系统仍然受制于两部分缺陷：i) 由于 GPU 显存有限，这些方法并不能有效利用较大的商用语言模型。ii) 这些方法并不通用，常常受制于特定的声学或语言模型，并且没有词图生成功能。我们将在第五章节对该方向进行研究。

## 2. 基于端到端建模降低解码搜索复杂度。

近来神经网络的发展使得更强的上下文和历史建模效果成为可能 [64, 31]。同时，更多的标注数据也进一步缓解了模型的稀疏性和泛化问题。这些进展使得研究人员们有可能在更大的模型粒度上-从帧到整个序列层面 [137, 138, 139, 140, 78] 进行序列分解。在这些研究中，标签速率小于特征速率，但解码速率仍然等于特征速率。针对端到端模型的特性，特征层面的搜索过程可以考虑被修改为标签层面，即搜索空间是由不同历史的标签组成的，使得解码速率等于标注速率，从而小于特征速率。由于标签速率显著小于特征速率，这样做的好处是将带来搜索复杂度的大幅下降，从而得到搜索速度的大幅提升。我们将在第六章节对该方向进行研究。而这种方法还能产生一种高效的音素词图，并以此为中介影响其他一系列 ASR 应用。

## 3. 基于端到端建模简化和统一解码推理框架。

由于不同 ASR 应用之间不同的搜索空间大小和效率要求，当前业界最优的置信度及其相应的解码算法在不同应用上具有不同架构，这些不同应用包括：关键词检测，基于上下文的语音识别和大词汇连续语音识别。要为所有的 ASR 应用设计一套统一的推理搜索框架，并取得最好的性能和速度，是一项非常有挑战的研究。这里面的关键点在于：i) 如何对最佳识别结果进行规范化，以得到对该最佳结果的置信度估计，也就是置信度中的归一化项的建模 ii) 如何保证在低功耗设备中的计算量控制在一个较小范围内，这对于置信度中的归一化项建模尤其具有挑战。借助上面提到的端到端建模，一方面可以对建模中的归一化项进行统一的较复杂的计算，另一方面可以有效降低上述计算中的复杂度。我们将在第七章节对该方向进行研究，将端到端建模所带来的解码搜索优化应用到更多领域中。

### 3.5 本章小结

本章回顾了语音信号识别和推理搜索阶段的搜索和解码部分的基本内容和方法。解码器将由声学模型计算的声学特征概率与由语言模型计算的语言概率组合，以获得最大概率词序列。在语音信号的识别和推理搜索阶段，解码器是语音信号识别系统的核心和灵魂，并且在此收集所有信息。它将来自不同来源，不同层次和不同性质的知识和信息联系起来，使它们相互补充，得到正确语音信号的识别结果。因此，如何有机地整合各种不同的信息是在解码网络和解码算法的设计中必须仔细研究和解决的问题。从解码器的作用来看，它不仅是验证语音信号识别中各种理论，模型和算法正确性的基本实验平台，也是构建实际系统的基础。因此，在解码器的设计中，还需要考虑研究的便利性和工程的实际应用。

本章中重点总结了目前解码器的研究现状及存在的缺点和实现难点。后续几章我们的改进工作将具体针对所有这些目前存在的问题进行展开。



## 第四章 非传统识别任务的序列建模

语音识别是一种序列预测问题，在语音识别中一个独有并且有趣的自然现象是声学序列和语言学序列的长度可变性。序列模型正是被提出用来解决两个序列之间的关联性。在这些序列模型中，除了需要研究如何将各种基于深度学习的模型方法应用到逐帧分类当中，使用序列级的鉴别性训练准则来强化模型的序列分类能力，也被证明是取得业界最好的大词汇连续语音识别系统的关键。

序列建模方法通常只在训练 LVCSR 模型时进行使用，但针对其它非传统识别任务的序列建模研究并不充分，这包括：关键词检测任务和多说话人重叠语音信号识别任务等。这类任务仍然是序列预测问题，但是却没有合适的训练准则和相应的设计，来充分优化分类器的序列建模能力。因此这些领域近几年的进展主要仍然来自于深度学习本身所带来的更好的逐帧分类能力。

本章节将针对上面所说的两类任务探讨序列建模的方案，使之更加通用。我们提出了针对固定或者非固定关键词的关键词检测系统的序列鉴别性训练框架。实验表明我们提出的方法取得了一致并且显著的改善。这些测试包括固定关键词或非固定关键词的关键词检测任务，比较的基线是逐帧的深度学习声学 KWS 模型方法。另一方面，在多说话人重叠语音信号识别任务中，本章节通过联合优化，迁移学习，序列鉴别性训练等方式，改善了原来语音分离、信号增强和语音识别的联合训练系统。

### 4.1 关键词检测的序列建模

关键词检测 (KWS) 是语言识别最主要的应用之一，它的目标是得到一个高准确度和高效率的识别器，用于检测特定的一些关键词在语音中的出现。KWS 可以被用于声学数据挖掘 [92]，低资源的音频检索 [93]，语料库检索 [94] 和唤醒词识别 [95]。本文主要考虑后两种应用。

我们在第3.2.1章节对关键词检测技术的分类和相应的声学建模方法进行了总结。在声学关键词检测中，模型通常是进行逐帧分类的。目前研究趋势包括两方面：应用一个较强的逐帧分类器，比如深度学习模型，使最终系统带来性能提升 [95, 141]。其次，语音识别本身是一个序列分类问题，传统的 GMM-HMM 系统在使用序列级准则，序列鉴别性训练时，往往能带来显著性能提升，比如 discriminatively trained sub-word verification function [142], minimum classification error (MCE) [143] 和 performance-related

discriminative training [144]。在上述的鉴别性训练中，对识别可能路径的建模，即完整的搜索空间的建模是成功的关键。但是，在 KWS 中，由关键词所定义的领域内搜索空间要显著小于领域外的搜索空间。因此领域外的搜索空间通常还需要特定的建模单元进行建模。对领域外的搜索空间建模并不容易直接得到的问题限制了序列鉴别性训练在关键词检测中的广泛应用。特别是在非固定关键词的关键词检测，所以可能的竞争可能路径通常无法被穷举，同时这种可能路径的产生过程在使用 LVCSR 框架进行的时候 [22] 非常耗时。

在这篇文章中，我们为深度学习的声学非固定关键词的关键词检测设计了相应的序列鉴别性训练方法。对这两种框架，竞争可能路径的建模都是核心难题。本论文提出采用无词图鉴别性训练框架来解决这一问题：隐性使用音素或半词单元的语言模型来建模。在 HMM 中，受到近期一些研究中使用特点音素语言模型来对 LVCSR 的搜索空间进行建模的启发 [83, 84]，我们提出使用这种音素语言模型来对关键词的完整搜索空间进行建模。为了加强关键词的鉴别能力，在关键词出现时候它们的梯度可以被增强。除此之外，许多神经网络结构和鉴别性训练准则也进行了详细比较。

#### 4.1.1 关键词检测中传统序列建模方法的缺陷

序列建模方法在语音识别中的应用如第 2.3 章所述。而在声学 KWS 方法中，只有关键词序列被定义，而非关键词的竞争序列还未知。针对生成式序列模型 (GSM)，基于似然度比值的可能路径测试框架 [101] 被提出以进行鉴别性训练，这里对其他竞争序列的似然度进行了惩罚。而这部分被惩罚的似然度使用两种单元进行建模：`filler` 模型，建模非关键词的  $p(\mathbf{O}_u|\Phi)$ ，建模错误识别词的  $p(\mathbf{O}_u|\Psi)$ 。与公式 (2-79) 相比，关键词序列的  $p(\mathbf{O}_u|\mathbf{W}_u)$  可以被忽略，而对数边缘概率则被定义如下，

$$\log p(\mathbf{O}_u) = \left\{ \frac{1}{2} [\log p(\mathbf{O}_u|\Psi)^\lambda + \log p(\mathbf{O}_u|\Phi)^\lambda] \right\}^{1/\lambda} \quad (4-1)$$

这里的序列竞争可能序列  $\mathbf{W}$  是由 N-best 序列得到的。在这些方法中，人工加入的非关键词单元并不能进行很好的建模，原因是发音和声学的不同并没有办法完全囊括在一个建模单元当中。除此之外，该方法也没有对关键词和非关键词之间的上下文进行建模。另外由 N-best 得到竞争序列也并不重复。最后，该类方法也无法应用到非固定关键词的关键词检测中。

#### 4.1.2 关键词检测的序列鉴别性训练

如前面的章节 2.3.2 所述，将基于 HMM 的 KWS 使用序列鉴别性训练进行改善的主要难度在于如何得到词级别的可靠竞争序列估计。受到之前研究中使用一个裁剪过

的音素语言模型来代替词图在鉴别性训练中的应用的启发(称为无词图鉴别性训练(LF-MMI) [83, 84]), 我们提出了一种通用的序列鉴别性训练框架, 并针对非固定关键词的关键词检测任务。这里的关键词序列由音素声学模型进行建模。相应我们就使用音素的语言模型来构建一个完整的搜索空间, 使之包含关键词和非关键词的竞争序列。

#### 4.1.2.1 模型训练

在所提出的音素声学模型中, (2-79) 被转换为 (2-81) 并表示为 LF-MMI。公式中  $\mathbf{L}$  是音素序列。 $\mathbf{L}_u$  是标注标签序列。 $p(\mathbf{O}_u|\mathbf{L})$  和  $p(\mathbf{O}_u|\mathbf{L}_u)$  由公式 (2-74) 得到。

我们在第2.4.1章节总结了该方法与传统鉴别性训练的不同之处。

为了更好地将 LF-MMI 应用到关键词检测当中, 我们进行了如下改进: i) 我们使用音素而不是 tri-phone 作为建模单元。首先是在测试阶段, 效率将会得到改善; 其次是这样会使得后文将讨论的 `filler` 构建显著简化。ii) HMM 的拓扑结构, 在第2.3.1章节介绍的基础上, 如图 2-8(c-e) 所示进行了修改。具体来讲, 在图2-8 (c) 中, 每个 CD 音素都有独立的 blank 状态, 称为 CD 音素 blank (CD phone blank)。为减少模型单元的数量并进一步加快算法速度, 将中心音素相同的 blank 状态绑定在一起, 称为音素级 blank (phone blank); 最后如果绑定所有的 blank 状态则称作全局 blank (global blank)。此外, 鉴于标签延迟带来的性能改进 [137], 图2-8 (d) 中提出 HMM-PB 模型的延迟标签变种, 即 HMM-BP。也就是说, 模型在确定性标签输出之前输出混淆输出 blank。另外, 作为对 CTC 的完整模拟, 图2-8 (e) 中提出了 HMM-BPB, 允许在标签输出前后都存在 blank。我们的初步实验结果表明, 这两种类型的 blank 展现出了不同的功能。因此没有将它们绑定在一起。而输出标签单元后面的所有 blank 则都被绑定在了一起, 以减少所需的模型单元数量。在实验部分我们将详细比较这些拓扑结构。

为了对模型的鉴别性做进一步改善, 一系列其它的序列鉴别性训练准则也在 KWS 的框架下进行了研究。首先, 我们对序列中出现错误的部分进行了增强 [145], 称为增强的 LF-MMI, LF-bMMI。

$$\mathcal{F}_{\text{LF-bMMI}} = \sum_u \log \frac{\sum_{\mathbf{L}_u} p(\mathbf{O}_u|\mathbf{L}_u)^\kappa P(\mathbf{L}_u)}{\sum_{\mathbf{L}} p(\mathbf{O}_u|\mathbf{L})^\kappa P(\mathbf{L}) e^{-b \max_{\mathbf{L}_u} A(\mathbf{L}, \mathbf{L}_u)}} \quad (4-2)$$

公式中  $A(\mathbf{L}, \mathbf{L}_u)$  表示逐个状态的准确度, 通过比较序列和标注序列而得到。 $b$  是增强参数。由于  $\mathbf{L}_u$  在分母中应用时是一个软对齐结果, 因此增强量被选择为使用最佳准确度来获得。另一种鉴别性训练框架尝试最小化预期错误 [146]。无词图的 minimum Bayes risk (LF-sMBR) 也在本章节中进行了研究。

$$\mathcal{F}_{\text{LF-sMBR}} = \sum_u \frac{\sum_{\mathbf{L}} p(\mathbf{O}_u | \mathbf{L})^\kappa P(\mathbf{L}) \max_{\mathbf{L}_u} A(\mathbf{L}, \mathbf{L}_u)}{\sum_{\mathbf{L}} p(\mathbf{O}_u | \mathbf{L})^\kappa P(\mathbf{L})} \quad (4-3)$$

这里所提出的序列鉴别性训练方法也可以被拓展到固定关键词的关键词检测任务中。为了增强针对特定关键词的区分度，我们可以把关键词相关的梯度进行加权。我们采用将逐帧的非一致性权重加入到损失函数中的方法，类似于 [147] 在 MCE 中的做法。这里的重点是要增强针对关键词的误唤醒或者未唤醒的情况。比如在 LF-MMI 中可以进行如下修改，得到非一致性 LF-MMI (NU-LF-MMI):

$$\frac{\partial \mathcal{F}_{\text{NU-LF-MMI}}}{\partial \log p(\mathbf{o}_{ut} | s)} = \frac{\partial \mathcal{F}_{\text{LF-MMI}}}{\partial \log p(\mathbf{o}_{ut} | s)} \cdot \ell(t, u) \quad (4-4)$$

公式中  $s$  表示建模单元， $p(\mathbf{o}_{ut} | s)$  是真经网络在状态  $s$  时间  $t$  句子  $u$  上的输出。 $\ell(t, u)$  是求导后针对导数的权重方程，它给定了时间  $t$  和句子  $u$ 。 $\ell(t, u)$  可以定义如下：

$$\ell(t, u) = \begin{cases} \min(\alpha, \beta) & r_{ut} \in \mathbf{K} \wedge i_{ut} \in \mathbf{K} \\ \alpha & r_{ut} \in \mathbf{K} \wedge i_{ut} \notin \mathbf{K} \\ \beta & i_{ut} \in \mathbf{K} \wedge r_{ut} \notin \mathbf{K} \\ 1 & others \end{cases} \quad (4-5)$$

公式中  $\mathbf{K}$  是所有的关键词序列。 $r_{ut}$  是音素级的标注序列， $i_{ut}$  是相应的推理搜索序列。 $\alpha$  和  $\beta$  为相应的增强参数（它们都大于 1），它们依次针对未唤醒和误唤醒。对于公式 (4-5) 的第一种例子， $\min(\alpha, \beta)$  被使用，原因是模型已经能够预测出相应的关键词。 $\ell(t, u)$  可以由 LF-MMI 的种子模型得到。

#### 4.1.2.2 后处理

在这项工作中，我们采用后验平滑和基于 `filler` 的后处理方式。

- 后验平滑

后验平滑的目的是将噪声后验值简单地滤除掉。该方法第一次由 [95] 提出，下面我们仅罗列主要公式，具体讨论详见 [148]。

$$P'(s' | \mathbf{o}_{ut'}) = \mathcal{N}_1 \left( \mathbf{P}(s | \mathbf{o}_{ut})^{(s=s', t \in [t' - \frac{1}{2} w_s, t' + \frac{1}{2} w_s])} \right) \quad (4-6)$$

$$P''(s'|\mathbf{o}_{ut'}) = \mathcal{N}_2 \left( \mathbf{P}'(s|\mathbf{o}_{ut})^{(s=s', t \in [t'-w_m+1, t'])} \right) \quad (4-7)$$

$$\mathcal{C}(\mathbf{k})^{(t')} = \mathcal{N}_3 \left( \mathbf{P}''(s|\mathbf{o}_{ut})^{(s \in \mathbf{k}, t=t')} \right) \quad (4-8)$$

公式中  $P(s'|\mathbf{o}_{ut'})$ ,  $P'(s'|\mathbf{o}_{ut'})$  和  $P''(s'|\mathbf{o}_{ut'})$  是三种不同的归一化函数。 $\mathbf{P}(s|\mathbf{o}_{ut})^{(s \in \mathbf{s}, t \in \mathbf{t})}$ ,  $\mathbf{P}'(s|\mathbf{o}_{ut})^{(s \in \mathbf{s}, t \in \mathbf{t})}$  和  $\mathbf{P}''(s|\mathbf{o}_{ut})^{(s \in \mathbf{s}, t \in \mathbf{t})}$  是三种平滑后的状态序列。 $\mathcal{N}_1(\cdot)$ ,  $\mathcal{N}_2(\cdot)$  和  $\mathcal{N}_3(\cdot)$  是三种不同的平滑函数。在工作 [95] 中, 算数平均, 最大值, 集合平均, 分别被采用。

- **filler** 解码

**filler** 解码方法用于建模前面我们所讨论的领域外的搜索空间。在这项工作中的搜索空间如图 4-1 所示。

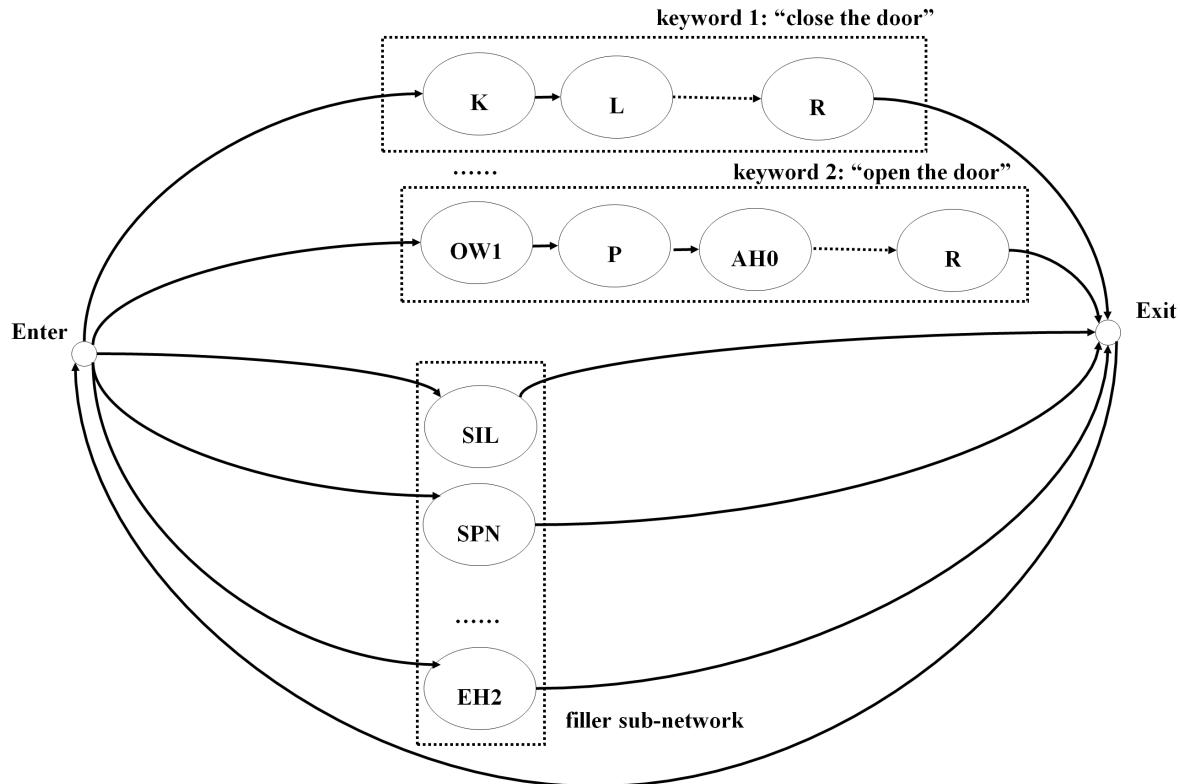


图 4-1 基于 **filler** 解码的搜索空间结构图

Fig 4-1 **filler** based decoding search space

它包含了两个部分: 领域内搜索空间和领域外搜索空间。前者由关键词的子序列建模得到, 如图 4-1 中的音素序列。**filler** 子网络用于对领域外的搜索空间进行建模。**filler** 子网络是由所有的音素自环组成的。

### 4.1.3 实验结果

本部分的关键词检测实验将分别在非固定关键词的关键词检测(英文语料库检索任务),和固定关键词的唤醒词识别(中文唤醒词识别任务)上进行。所有实验都使用声学KWS。

#### 4.1.3.1 非固定关键词的关键词检测

**Wall Street Journal (WSJ0)** 数据集的一个说话人独立的 5k 词表数据集 [149] 被用于评估。至少出现过 5 词的词或短语被随机挑选出来作为关键词。总计有 50 个关键词。

由于 WSJ0 数据集比较小, 同时关键词出现次数较少, 因此我们在建模中使用音素作为建模单元, 以提高泛化能力, 而词作为建模单元将在下章中进行。输出的音素由 CMU 发音词典得到。24 维对数 filter-bank 系数及其第一和第二阶导数组成了 10ms 固定帧率的特征序列。HMM 模型的配置与 [83] 中相似, 但使用了更好的参数如表 4-1 和表 4-4 所示。我们估计了三元的音素语言模型用于 LF-MMI 的训练。我们使用  $\alpha = 2.5$  和  $\beta = 2.5$  作为 NU-LF-bMMI 的参数。所有声学模型都使用 Kaldi 进行训练 [150]。等错误率 (EER) 用来度量上面两种测试下的错误率, 该指标反映了无唤醒和未唤醒错误的均值。越低的 EER 表示越好的模型性能。在基于 **filler** 的解码中, 表示为 *kw-filler*, 而 EER 是由扫描修改 **filler** 权重而得到的。后验概率平滑, 表示为 *smooth*。

$$\mathcal{T}_{EER}(\mathbf{k}) = \mathcal{T}_0 + \mathcal{T}(\mathbf{k}) \quad (4-9)$$

公式中  $\mathcal{T}(\mathbf{k})$  是针对关键词  $\mathbf{k}$  的阈值估计。 $\mathcal{T}_0$  在所有关键词中共享, 并通过调节它得到 EER 结果。

实时率 (RTF), 解码时间与音频时长的百分比, 被作为对整体速度的衡量指标。越低的 RTF 表示速度越好。在测试阶段, 我们使用的 CPU 型号为 *Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00 GHz*。

本章节比较了不同的声学模型配置。基于 HMM 的序列鉴别性训练结果见表 4-1。这里的所有模型都是通过公式 (2-81) 中的 LF-MMI 模型进行训练得到的, 同时这里使用了 KW-Filler 的后处理方法。

我们首先对深度学习模型的架构进行了研究。第一二行比较了双向 LSTM (BLSTM) 和时延神经网络 (TDNN) 在同等参数下的性能。BLSTM 包含两层前后向各 80 个节点的 LSTM 网络。映射层包含 30 个节点。CD TDNN 包含 7 层 100 个节点, CI TDNN 包含 7 层 150 个节点。结果显示了相似的 EER 性能。我们相信这是由于: i) 在 KWS 中, 作为模型推理搜索所需要的上下文并不需要很长, TDNN 已经足够来针对这样的应用。ii) 由

表 4-1 基于 *HMM* 的序列鉴别性训练的模型框架

NN Model	Context	# Param.	CEW	HMM	EER
BLSTM	CD	0.60M	0.1	PB	3.3
TDNN	CI	0.51M	0.1	PB	3.3
			0.4		3.2
			<b>0.7</b>		3.1
			1.0		3.2
			0.7	<b>BP</b>	3.0
				BPB	3.0

于 KWS 模型的参数较少，因此限制了 BLSTM 模型的性能。在接下来的实验中，我们只讨论 TDNN，原因是同等性能情况下它的速度更快<sup>1</sup>。其次，模型的建模单元在第二和三行中进行了比较。tri-phone 状态模型，表示为上下文相关系统 (CD)，与单音素状态模型，表示为上下文无关系统 (CI) 进行了比较。CD 模型基于三状态的从左到右的 triphone 模型，包括 1536 个绑定状态 (senones)。它们的性能接近于 CI 模型。第三，对于交叉熵规范化权重，表示为 CEW，在第三到第六行中进行了调整。结果显示 0.7 可以得到最优结果，该值被使用在了后续的实验中。该值比 LVCSR 中偏大，原因是音素语言模型在测试阶段并不存在，这与 LVCSR 不同，因此训练过程仍在鉴别性训练和交叉熵准则之间权衡。最终我们使用的拓扑结构如图 2-8 所示在第五，七和八行进行了比较。我们提出的 BP，BPB 比 PB[83] 轻微改善。但是在 50 个关键词上的显著性检验显示结果并不充分 ( $\alpha = 0.05, p = 0.18$ )。性能改善的原因可能为标签延迟所带来的改善 [137]，但需要后续更多的研究。由于 BP 的搜索空间比 BPB 小，而性能相似，因此后续实验使用 BP。

表 4-2 基于 *HMM* 的鉴别性训练的准则比较

鉴别性训练准则	EER
LF-MMI	3.0
<b>LF-bMMI</b>	2.9
LF-sMBR	2.9
NU-LF-bMMI	2.7

<sup>1</sup> 我们暂未比较 LSTM。近期的一些研究发现 TDNN 和 LSTM 可以被结合并取得更好的性能 [151]

第 4.1.2.1 章给出了不同准则，在表 4-2 中进行了比较。这里使用 kw-filler 进行后处理。LF-bMMI 和 LF-sMBR 都显示好于 LF-MMI 的性能，这与 LVCSR 中的结论一致 [152]。由于 LF-bMMI 训练快于 LF-sMBR 而性能相似，因此它被用在了后续实验中。除此之外，NU-LF-bMMI 也参与了比较。这是一个固定关键词的关键词检测算法。在这种情况下，50 个预先定义的关键词在训练阶段进行了使用，以加强关键词相关的梯度权重。经过训练之后，声学模型与关键词相关。NU-LF-bMMI 带来额外的性能提升，而另一方面传统的固定关键词的关键词检测方法 [95] 并不能泛化这样的关键词无关训练集。后续 NU-LF-bMMI 并不继续参与该数据集的比较，因为它与关键词无关算法并不可比。

不同的针对 HMM 的后处理算法在本节中进行了检查。值得注意的是，本部分是非固定关键词的关键词检测任务，因此音素被用来构建关键词序列。

表 4-3 序列鉴别性训练系统的后处理

Model (Crit.)	Post	EER	RTF
HMM (LF-bMMI)	smooth	9.8	0.008
	<b>kw-filler</b>	2.9	0.028

如表 4-3 所示，在 LF-bMMI 中，后验概率平滑方法会得到比 kw-filler 显著差的结果。但是由于它的速度优势，该方法可以作为实际的前处理，滤除一些容易判断的样本，以减少语料库检索同的计算量。

最后，我们将性能和速度的比较总结在表 4-4 中。其中所有的系统都使用 kw-filler 作为后处理方法。

表 4-4 非固定关键词的关键词检测中的性能和速度比较

Model	Context	# Param.	Criterion	EER	RTF
TDNN HMM	CD	0.6M	CE	4.0	0.051
	CD	0.6M	CE+sMBR	3.5	0.050
	CI	0.5M	LF-bMMI	<b>2.9</b>	<b>0.028</b>

这里我们使用交叉熵训练得到的系统作为第一行的基线。它是一个基于 NN-HMM 的传统系统，包含了聚类的 tri-phone 状态建模。这里的 HMM 拓扑结构见图 2-8(a)。传统鉴别性训练在交叉熵系统基础上进行。这里的词图由模型与一元语言模型解码得

到 [153]。前两行显示传统的词图鉴别性训练改善了 12% 性能，这一结论与 LVCSR 中类似 [22]。在 RTF 上的轻微改善则是由于更好的声学模型提供了更少的搜索混淆性。

第三行对比第二行显示了所提出的最好的基于 HMM 的序列鉴别性训练方法与传统鉴别性训练方法的比较。这里显示出 17% 相对改善 ( $\alpha = 0.05, p = 0.06$ , 在显著性检验测试中) 相比传统鉴别性训练和 28% 相对改善 ( $\alpha = 0.05, p = 10^{-5}$ ) 相比交叉熵训练系统。这来源于更好的在所提出方法中的建模效果：

- 如前文所述，修改后的 HMM 结构和更低的帧率都能带来性能的改善。
- KWS 模型的 LVCSR 词图并不包含较好的竞争路径。这些词图来自声学模型与语言模型的共同解码，而 KWS 的声学模型性能较弱，并不是针对 ASR 识别进行设计的。因此这些质量更差的词图限制了鉴别性训练所带来的提升。
- 传统鉴别性训练在生成词图时候使用的词级别语言模型在 KWS 的测试阶段并不存在。而 LF-MMI 中使用的词级别语言模型则是对于测试阶段使用的由词典构成的关键词序列的一个很好的近似。

除此之外，与 CE 基线相比，LF-bMMI 系统取得了一倍的加速，原因在于模型推理搜索和解码搜索都得到了加速。对于前者，主要是由于帧率减少。而后者主要是由于所使用的 CI 单元比传统 CD 单元少，使得搜索空间变少，这也包括对 filler 模型的相应简化。

#### 4.1.3.2 固定关键词的唤醒词识别

本章节进一步测试了在固定关键词的 KWS 系统中的性能。我们准备了一个类似于文献 [95] 和 [154] 中介绍的数据集。它包含两部分：通用语音数据和关键词相关数据集。第一部分包含 100 小时，第二部分包含 30K 关键词正例，和 180K 关键词负例。详细的数据集配置可以参见 [148]。

测试数据包含两部分：关键词相关集合和环境噪声集合。第一部分用于测试模型在区分关键词和非关键词时候的能力 [95]。它们包含 2K 正例和 10K 负例，代表了 20% 左右的比值，这符合该应用的实际应用场景。前文我们讨论的 EER 被作为准则。环境噪声集合则用于测试模型对误唤醒的鲁棒性 [154]。它包含 300 小时的环境噪声。每小时的误唤醒次数记为误唤醒频率 (FAF)，也被作为度量模型性能的指标。后处理中获取 EER 的超参数方法如前面章节所述。每个关键词都单独包含它自己专门的平滑方法的阈值。总体的 EER 是每个关键词 EER 的平均值。之后我们会固定所有的超参数并测试系统在噪声集合中的 FAF。

TODO 在该任务中，我们测试了词级系统和音素级系统。HMM 的鉴别性训练系统与传统交叉熵系统进行了比较。词级系统的输出为关键词序列中的每个词，而音素级系统的输出则为中文中的不带调音节。特征和声学模型配置与前面章节一致。NU-LF-bMMI 采用  $\alpha = 10.0$  和  $\beta = 10.0$ 。

表 4-5 显示了相应的结果。一个交叉熵训练的词级别的 TDNN 系统被作为基线系统 [95]。该系统采用了后验平滑作为后处理方式。

表 4-5 固定关键词的 KWS 系统的性能和速度比较

Model	Unit	Criterion	Post	EER	FAF	RTF
[95]	Word	CE	smooth	6.2	0.64	0.014
HMM	Syllable	CE	kw-filler	10.2	1.40	0.041
		LF-bMMI		8.3	1.06	0.033
		NU-LF-bMMI		<b>5.2</b>	<b>0.51</b>	<b>0.029</b>

在第二行中，模型也是通过 CE 准则进行训练的。模型的建模单元是音节，后处理方法为 kw-filler。该系统性能在 EER 和 FAF 中都有所下降，原因是音素级系统具有更多建模单元。而这些建模单元都被等同地进行训练。与之相反，词级系统则可以对特定关键词具有更强的建模能力。[95] 中显示了类似结论，这说明后验概率平滑方法的 CE 模型系统是一个较强的基线系统。

在第三行中，我们所提出的 LF-bMMI 系统显著改善了 CE 系统，取得了一致更优的 EER 和 FAF。但是，所得到的系统仍然差于基线系统。为了解决针对关键词区段识别性能差的问题，我们使用了 NU-LF-bMMI 方法，在第四行中。结果显示 NU-LF-bMMI 系统得到了相对第一行基线中显著更好的结果。我们认为这包括两方面原因：第一，非一致地训练关键词相关的音素序列，理论上等同于对关键词的每个词单元专门进行训练，所以其建模能力类似；第二，由于音素系统通过所提出的 NU-LF-bMMI 增强了关键词之间与关键词和非关键词之间的鉴别能力，因此它能够取得更优的性能。关于效率，虽然该系统的模型推理搜索速度较快，但由于引入 kw-filler 的搜索部分，因此导致两倍的速度减慢。使用标签同步解码算法对系统进行加速是未来一个值得研究的问题。

## 4.2 鲁棒语音识别中的深度序列建模

近年来随着深度学习发展，将语音信号的增强和语音识别进行联合训练成为一种新的趋势。这样做好处包括：更好地利用序列信息进行训练；对模型前后模块进行联

合调优以解决模型失配问题。在多说话人重叠语音信号识别任务中，本章节通过联合优化，迁移学习，序列鉴别性训练等方式，改善了原来语音分离、信号增强和语音识别的联合训练系统。

#### 4.2.1 排列不变性训练及其在鸡尾酒会问题中应用

鸡尾酒会问题 [155, 156] 是指多说话人重叠语音信号的识别问题，该问题在会议转录系统，自动音视频字幕生成等任务中非常重要，因为语音信号的重叠是常见的一种信号现象。同时该问题也是语音识别中最困难的问题之一 [157, 158, 159, 160]。在鸡尾酒会问题中，最困难的是无监督鸡尾酒会问题。这种情况下，多人同时说话，而只有一个麦克风系统进行收音，同时系统不提前具有说话人信息，需要泛化到没有见过说话人情况。

排列不变性训练是解决该问题的一种有效框架 [161]。该框架首先通过选取最小的当前排列下的推理搜索误差来决定当前最优排列，而后依据该排列来使用其相应的误差对模型进行反向误差传递的梯度更新。在鸡尾酒会的语音识别问题中，原先的信号分离误差被替换为针对语音识别结果的交叉熵误差。

$$\mathcal{J}_{CE-PIT} = \sum_u \min_{s' \in \mathbf{S}} \sum_t \frac{1}{N} \sum_{n \in [1, N]} CE(l_{utn}^{(s')}, l_{utn}^{(r)}) \quad (4-10)$$

公式中  $\mathbf{S}$  表示所有针对标注和推理搜索序列的可能排列组合。 $l_{utn}^{(s')}$  为在排列  $s'$ ，第  $t$  帧，第  $u$  个句子上的第  $n$  个推理搜索结果。 $l_{utn}^{(r)}$  是相应的从干净语音信号进行对齐算法得到的标注序列 [53]。PIT-ASR 准则 [161] 优雅地将语音信号的分离，说话人跟踪，语音识别都融合在一个系统里，如图 4-2(a) 所示。

在接下来章节中，我们提出将无监督的鸡尾酒会问题拆分为三个子问题，并逐一进行初始化：逐帧的语音信号的分离，说话人跟踪，语音识别，如图 4-3 所示。每个模块通过在上一个模块基础上添加若干层神经网络而得到，由此逐渐解决更加困难问题。在初始化之后，各模块组合起来进行联合调优。我们提出了两种联合训练思路：针对自身的迁移学习，多输出的序列鉴别性训练；除了这些工作之外，我们还提出了一系列对建模过程的改善，参见 [162]。

#### 4.2.2 基于迁移学习的排列不变性训练

迁移学习（教师-学生训练）被引入到这个问题中，以便使用平行干净语料来改善目标领域的混合语音识别系统。在这一框架中，学生是指该多说话人语音识别系统。它工作在目标领域，即混合语音信号的数据上，并尝试得到每个说话人相应的话语内容。

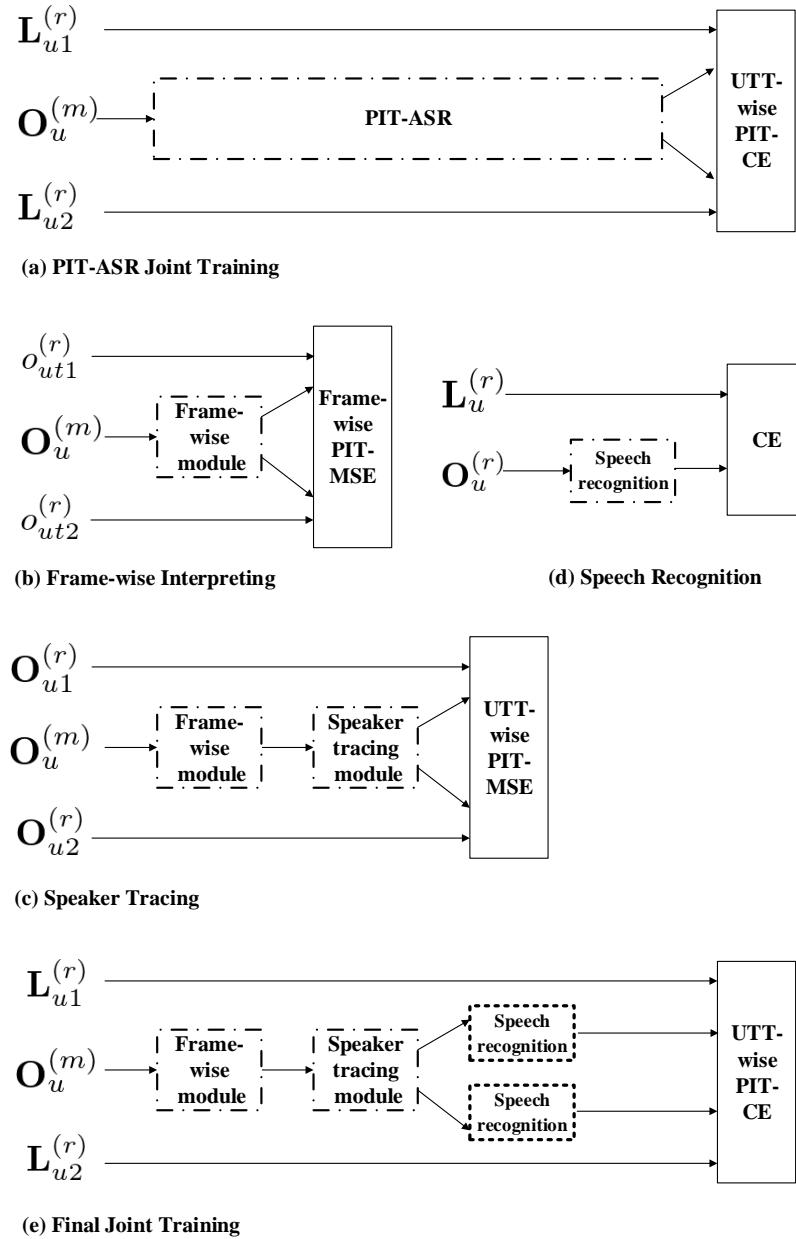


图 4-2 PIT-ASR 联合训练与模块化初始化和逐步联合训练比较。点划线框表示可以学习的模型参数。  
点线框表示可以学习的并且是共享的模型参数。

Fig 4-2 The comparison between joint training, modular initialization, and progressive joint training.

这里教师模型输入源领域，即干净语音信号，并尝试分别对每个说话人得到相应推理搜索序列。

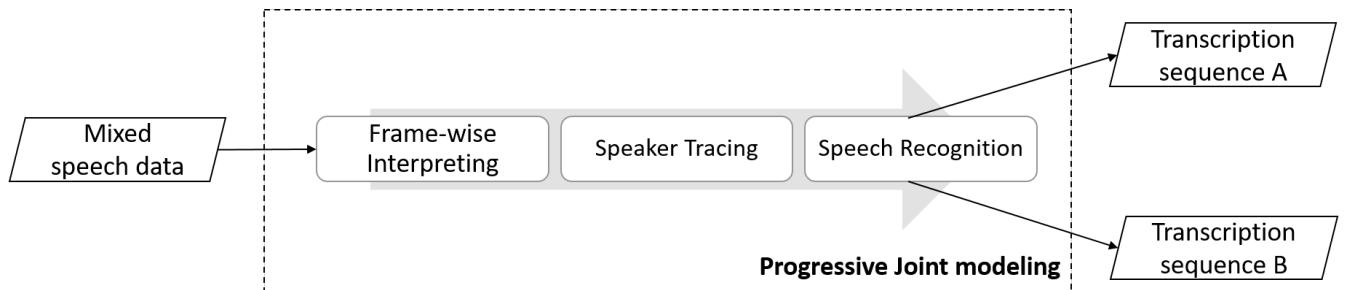


图 4-3 迁移学习模型系统训练框架。[161] 中提出的结构由虚线框部分表示，其用于推理搜索出每个说话人的语音信号的信息。该结构被模块化（三个实线框）并作逐一增量预训练。针对自身的迁移学习和多输出序列鉴别性训练在模块初始化后神经网络上进行。

Fig 4-3 The progressive joint training model framework.

这一针对自身的迁移学习将最小化混合语音信号的模型和干净语音信号的模型输出分布之间的 KL 散度 (KLD)。这一 KLD 散度被定义为句子层面干净语音信号的和混合语音信号的推理搜索分布之间的 PIT 准则，如下所示，

$$\begin{aligned} \mathcal{J}_{\text{KLD-PIT}} &= \sum_u \min_{s' \in S} \sum_t \frac{1}{N} \sum_{n \in [1, N]} \\ &KLD(P(l_{utn}^{(c)} | \mathbf{O}_{un}^{(r)}), P(l_{utn}^{(s')} | \mathbf{O}_u^{(m)})) \end{aligned} \quad (4-11)$$

该式中，每个  $KLD(\cdot)$  对的计算过程和经典领域自适应方式迁移学习公式相似。值得注意的一点是，当该方法被应用到如图 4-4 所示的结构时，语音识别模块可以直接使用教师模型进行初始化。

图 4-5 给出了多种训练方法在验证集上的学习曲线比较。从图中可以看出，本章节所提出的方法具有更好初始点和最终的收敛点。这些都得益于本章节方法使得神经网络的学习任务更简单，最终收敛效果相应得到了提升。

#### 4.2.3 基于序列鉴别性训练的排列不变性训练

本章节我们提出了一种传统鉴别性训练技术变种，它在进行鉴别性训练的同时，也抑制输出通道上说话人跟踪错误。

语音识别是一种序列分类和预测问题。因此在单输出语音识别中，序列级的准则将有助于改善系统性能。而在鸡尾酒会问题中，对于单通道多说话人混叠语音识别系统，

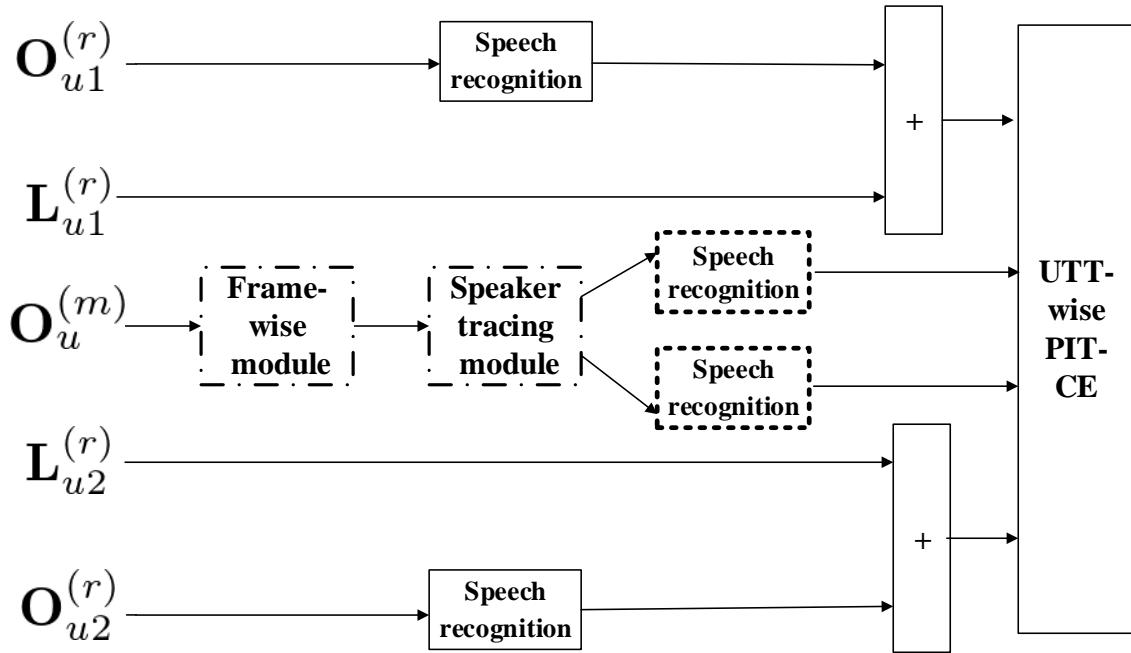


图 4-4 基于迁移学习逐步联合训练方法。点划线框表示可以学习的参数，点线框表示可以学习并且是共享的模型参数。

Fig 4-4 The self-transfer learning based model framework.

则又包含了说话人跟踪问题，也属于序列级问题。因此序列鉴别性准则将有助于这样的序列分类问题。在单输出语音识别中，等效于公式 (2-76)，我们使用如下公式对序列后验概率进行建模，

$$P(\mathbf{L}_u | \mathbf{O}_u) = \frac{p(\mathbf{O}_u | \mathbf{L}_u) P(\mathbf{L}_u)}{p(\mathbf{O}_u)} \quad (4-12)$$

公式中  $\mathbf{L}_u$  是句子  $u$  的词序列。 $P(\mathbf{L}_u)$  是语言模型概率。 $p(\mathbf{O}_u | \mathbf{L}_u)$  是相应声学模型概率。 $p(\mathbf{O}_u)$  是针对特征序列  $\mathbf{O}_u$  观察概率进行建模的边缘概率，它等于所有可能的识别序列的概率求和。

通常来说，语言模型搜索空间是通过训练集语料来估计得到。之后在该搜索空间上计算相应的序列准则。由于之前训练搜索空间的语料未出现词语交换问题，因此搜索空间也不包含这些建模。在这样的搜索空间上训练将导致对搜索空间估计不准确，而最终的推测序列中将会增加较多词语交换错误。我们提出了一系列方法以解决该问题，这里

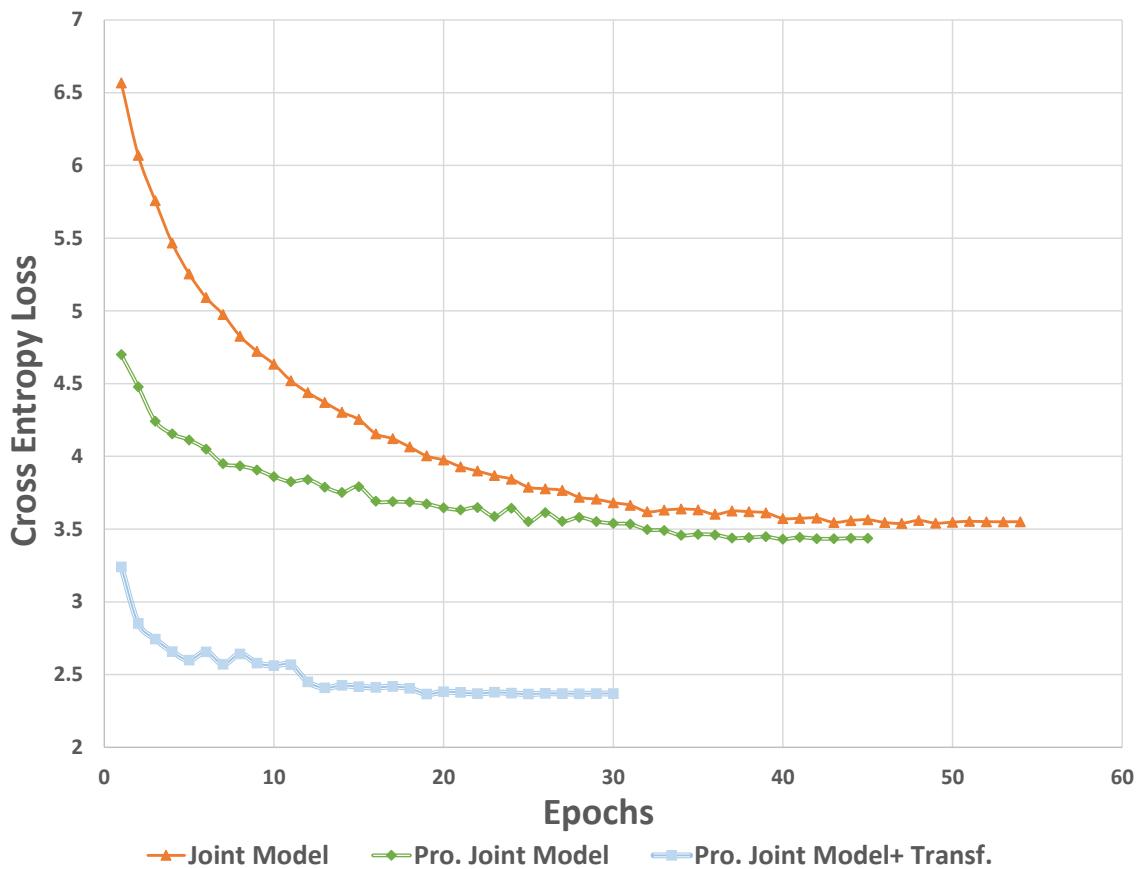


图 4-5 原始联合训练和所提出的方法在验证集上学习曲线比较。在图中，联合训练，逐步联合训练，基于迁移学习逐步联合训练被分别表示为 Joint Model, Pro. Joint Model 和 Pro. Joint Model + Transf. 图中每个 epoch 包含 24 小时训练数据。

Fig 4-5 The learning curve comparison of naive joint training and the proposed methods.

的想法是将词语交换错误添加到搜索空间中，由此减轻问题。

$$\mathcal{J}_{\text{LF-DC-MMF}} = \sum_u \log \left[ \frac{\sum_{\mathbf{L}_u} p(\mathbf{O}_u | \mathbf{L}_u)^\kappa P(\mathbf{L}_u)}{\left( \sum_{\mathbf{L}} p(\mathbf{O}_u | \mathbf{L})^\kappa P(\mathbf{L}) \right)^{1-\lambda}} \cdot \frac{1}{\left( \sum_{\mathbf{L}_{\hat{u}}} p(\mathbf{O}_u | \mathbf{L}_{\hat{u}})^\kappa P(\mathbf{L}_{\hat{u}}) \right)^\lambda} \right]. \quad (4-13)$$

在公式 4-13 中，除当前优化的输出序列之外其它输出序列表示为  $\mathbf{L}_{\hat{u}}$ 。公式中添加了一项插值系数  $\lambda$ 。该部分额外式子最后被加入到分母优化当中。通过这样的方法，在优化目标序列的同时，也同时抑制了跨信道的词语交换错误。

#### 4.2.4 实验结果

多说话人重叠语音信号识别任务在 Switchboard 数据集的人工混叠集合上进行，排列不变性训练是当前主流的建模方法之一，我们以排列不变性训练 [161] 作为该任务下的基线系统，并分别验证了前文所提出的联合优化，迁移学习，序列鉴别性训练等方式。

##### 4.2.4.1 实验配置

本章节使用 300 小时 Switchboard 数据集 [163]。测试使用 NIST 2000 CTS (hub5e-swbd) 测试集的 Switchboard (SWB) 子集。两个说话人的重叠语音是通过人工混叠而后合并得到的 (0dB)。文献 [162] 中描述了混叠的具体过程。经过混叠之后，我们拥有 150 小时作为训练数据，称为 150 小时数据集，以及 915 句测试集。通过解码之后，由于每句话有 2 个说话人，因此我们将得到 1830 个句子用于最终评估准确率。我们同时取了三分之一的训练数据组成一个更小的集合，称为 50 小时数据集。

在模型训练阶段使用 80 维 log-filterbank 特征。CNN 模型使用 41 帧作为窗长。模型使用 Microsoft Cognitive Toolkit (CNTK) [164] 进行训练。声学模型使用 9000 个 triphone 绑定状态。所有数据的状态信息是通过在人工混叠之前的干净数据上进行强制对齐而得到的。本章节使用的基线系统类似于文献 [161] 中的联合训练 PIT-ASR 系统。这个 PIT-ASR 模型包含 10 层双向 LSTM 层，包含 768 个模型单元在每个方向中。本章节系统中的语音识别模型包含 4 层双向 LSTM，每层包含 768 个单元，在干净语料上预训练得到。

在测试阶段，本章节使用 30k 词表大小的语言模型进行解码，详细配置参见文献 [162]。在测试中，PIT 模型的两个输出分别进行解码，以得到两个说话人分别的转录序列。对于 WER 的计算，我们分别计算各种两两排列组合底下的 WER，并选取更好的那一种组合作为最终的 WER [161]。我们这里给出了两个输出的平均 WER 作为最终的 WER。

#### 4.2.4.2 实验结果和比较

表 4-6 总结了前文所提出的各种改进对性能的影响。

表 4-6 50 小时数据集上的性能改善总结

深度学习	模型	WER	相对改善比 (%)
10·0 BLSTM	PIT-CE	57.5	0
6·4 BLSTM	progressive joint training	50.2	-13
	+ clean teacher	38.9	-32.4
	+ MMI clean teacher	35.8	-37.7
	+ LF-DC-bMMI	35.2	-38.8
1 LACE + 5·4 BLSTM	progressive joint training	47.4	-17.5
	+ clean teacher	36.0	-37.4
	+ MMI clean teacher	34.6	-39.8
	+ LF-DC-bMMI	34.0	-40.9

PIT-ASR 模型 [161] 在第一行中, 表示为 PIT-CE, 被作为联合训练方式的基线系统。本文提出的逐步联合训练方法在第二行中, 是一个更好的通过分部分微调结合起来得到的联合训练系统。本文提出的基于自迁移学习的联合训练模型显示出了额外的非常显著的性能改善, 显示在第三和第四行中。最后, 多输出的序列鉴别性训练被进一步应用到上述系统中, 并取得了额外的改善, 即使教师模型已经是一个基于 MMI 序列鉴别性准则训练的模型, 这与 [165] 中观察到的现象类似。图 4-6 中给出了一些本文提出的系统所得到的解码例子, 以及其相对应的 PIT 基线系统的例子。基线系统包含了很多模型泛化能力差所带来的错误。通过所提出的方法, 错误得到了显著的减少。注意到在这个例子中, 本章提出的基于自迁移学习的联合训练方法主要减少了相似发音之间的错误识别, 而序列鉴别性训练则主要减少一些明显的语法和语言学错误, 这些现象符合我们对两种训练方法的预期, 也显示了它们可以被结合在一起。

另一方面, 在相似参数量情况下但使用其它的深度学习模型, 即 1 LACE + 5·4 BLSTM, 这样的系统可以带来一致性的性能改善, 显示在第六到第九行中。我们相信这些额外的改善来自对于问题模块化的合理假设, 由此对深度学习模型进行了合理的使用。这些相似比较分析可以参见文献 [162]。

表 4-7 进一步将数据集扩展到 150 小时数据集, 由此显示了再更多训练数据时候的性能。

该表中传统的联合训练基线在第一行中, 它相比之前 50 小时系统显著改善了性能,

图 4-6 50 小时数据集的解码例子，这里比较了所提出的方法与 PIT 基线的不同性能。上半部分来自 A 输出，下半部分来自 B 输出。C, S, D, I 分别表示：正确，替换错误，删除错误，插入错误。

Fig 4-6 The decoding examples of 50 hours corpus.

表 4-7 150 小时数据集上的性能比较

深度学习	模型	WER	相对改善比 (%)
10·0 BLSTM	PIT-CE	42.2	0
6·4 BLSTM	progressive joint training	41.0	-2.9
	+ clean teacher	32.8	-22.3
	+ LF-DC-bMMI	30.8	-27.0
1 LACE + 5·4 BLSTM	progressive joint training	39.4	-6.6
	+ clean teacher	30.4	-27.9
	+ LF-DC-bMMI	28.0	-33.6

由此缩短了与逐步联合训练模型在第二行之间的性能差距。但是它仍然显著差于前面表中本文所提出的基于自迁移学习和鉴别性训练的 50 小时系统。这说明 PIT 这类模型确实比较受制于模型的复杂度和模型泛化的不充分性。换句话说，为了提升模型的泛化能力，使用更多的数据，不如使用本文所提出的更好的模型训练方法。另一方面，使用更多数据时模型的收敛速度也四倍慢于本文所提出的 50 小时系统。

比较表 4-7 与表 4-6，本文所提出的基于自迁移学习的联合训练方法以及鉴别性训练方法都带来了显著的相对改善，相比于上文中的基线系统。相比于表 4-6，序列鉴别性训练方法带来了更大的相对性能改善比，相比于传统基于 CE 准则训练的教师模型的迁移学习系统。

在两个数据集上，本文所提出的的方法都比当前最后的系统提升了相对 30%。这里的性能改善来自于更好的模型泛化能力，更高的训练效率，和更好地融合序列级的语言学信息。

### 4.3 本章小结

本章节提出了基于序列鉴别性训练的深度学习关键词检测模型的训练框架。通过使用音素语言模型的方式，对竞争可能路径进行建模，由此得到更好和更快的序列鉴别性训练。我们的实验在语料库检索任务和唤醒词识别任务上进行。对于前者，词表大小通常是无限制的，而对于后者，则要求有更强的噪声鲁棒性。相比于传统的逐帧 CE 深度学习模型在固定关键词和非固定关键词的关键词检测任务上的最好的系统，尽管不同应用具有不同的特征，但是实验显示我们所提出的鉴别性训练方案可以取得一致和显著的改善。另一方面，在多说话人重叠语音信号识别任务中，本章节通过联合优化，迁移学习，序列鉴别性训练等方式，改善了原来语音分离、信号增强和语音识别的联合训练系统，在多个不同大小数据集上显示了一致和非常显著的性能改善。



## 第五章 基于 GPU 并行计算的解码速度优化

本章节中我们将着重介绍我们针对 Kaldi 开源工具包的一个扩展，以便使它能够支持图形处理芯片 (GPU) 上的 WFST 解码推理搜索。该框架可以显著加速现有推理搜索算法，特别是在基于深度序列学习的一系列模型上进行了验证。

近来深度学习语音识别的发展唤起了大量语音识别转录的需求。在这一系统中，计算量较大的部分主要包括：声学模型推理和语言模型 WFST 解码。本章节主要针对后者。除了第三章节的总结，基于 GPU 的并行计算是一种潜在的针对语音识别计算的加速方式。针对声学模型推理，由于大部分计算量集中在矩阵形式的运算，因此它易于通过将类似于训练过程中 [134] 的序列批处理 [135] 引入推理搜索阶段，以加速运算。

如第 3.4.2 章节中对于解码搜索的研究机遇的讨论，语音识别解码网络中多数 WFST 边之间并不直接相关，具有并行处理的可能性。但是，基于 GPU 并行计算的 WFST 解码不容易实现。尽管一些研究在小型语言模型上取得了一定成功 [136]，但这些系统仍然受制于两部分缺陷：i) 由于 GPU 显存有限，这些方法并不能有效利用较大的商用语言模型。比如依据第 3.3.2.2 章节的讨论，商用语言模型的 WFST 大小将达到百 GB 以上，已经大幅超出了 GPU 的显存大小。ii) 这些方法并不通用，常常受制于特定的声学或语言模型，并且没有词图生成功能。

这项工作作为 Kaldi 工具包的一个扩展 [150]，实现了基于 GPU 并行计算的 WFST 解码。这是一个通用的离线解码器<sup>1</sup>，该解码器对语言模型和声学模型没有特别的限制，并且可以工作在各种架构的 GPU 上。

我们将维特比算法中的令牌合并操作实现为一个 GPU 并行计算中的原子操作，以便减少维特比束剪枝算法中的同步开销；我们提出了动态负载均衡的方式以更高效地进行并行计算，提高其多线程之间的利用率。为了支持第二遍重打分和更丰富的后处理，我们的设计基于 WFST 解码和词图生成的架构 [122]。我们重新设计了基于 GPU 并行计算的精确的词图生成和剪枝算法，以便充分利用 GPU 的性能特点。我们对这项工作进行了开源<sup>2</sup>，它将会与大多数 Kaldi 脚本相兼容。

在 Switchboard 上实验表明，我们所提出的方法在取得完全一致的最优路径 (1-best) 和词图质量情况下，可以得到 3-15 倍的加速，并在绝大部分 GPU 架构上进行了验证。除此之外，如果再进行多句子的并行处理，最终的加速比将达到 46 倍。

<sup>1</sup>近期的研究显示，使用 CPU 比较容易得到一个实时的解码器来针对在线解码应用 [107, 110]。因此我们的目标集中在如何解码海量的离线语音，降低其计算损耗。

<sup>2</sup><https://github.com/chenzhehuai/kaldi/tree/gpu-decoder>

## 5.1 维特比算法并行化框架

我们所提出的系统工作在二遍解码框架下 [166]，以便能够支持更大的语言模型，并支持词图上丰富的后处理。

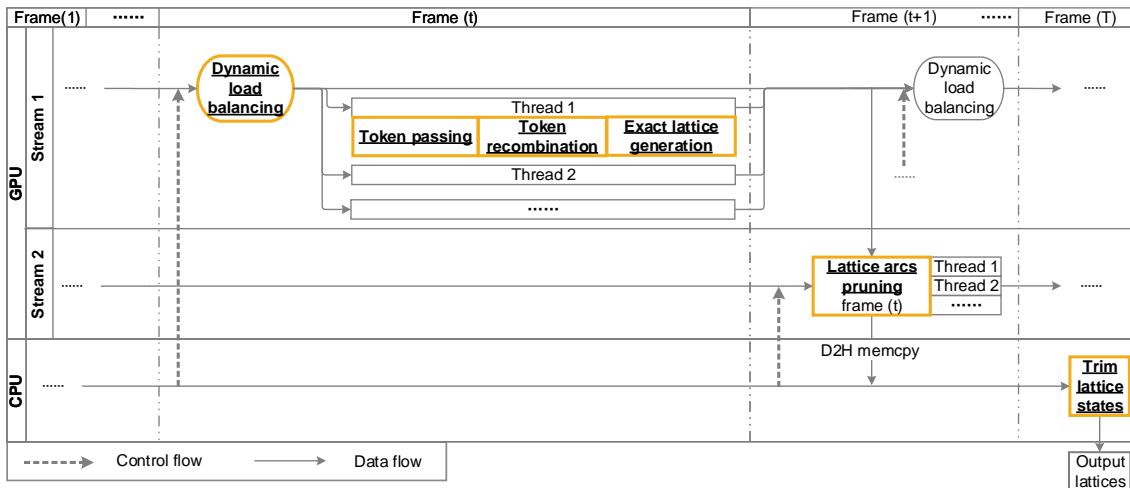


图 5-1 并行维特比束剪枝算法以及精确词图处理系统的框架

Fig 5-1 The framework of the parallel Viterbi beam search algorithm with exact lattice processing.

图 5-1 显示了系统的框架，这包括两个 GPU 同时处理流，分别进行解码和词图剪枝，它们完全并行同时受不同 CPU 线程所控制。具体来说，对第  $(t + 1)$  帧，第二个处理流将处理第一个流产生的第  $(t)$  帧的词图。

解码的流程类似于 CPU 版本 [150]，如算法3-1所示。本章节的并行算法将使得算法3-1中的第 7 和第 8 行的串行循环处理并行化。结合公式 (3-16) 和公式 (3-26)，串行版本的解码器复杂度大致为，

$$\mathbb{C}_{\text{serial}} = O(T \cdot I \cdot K) \quad (5-1)$$

定义上述算法3-1中第 7 行的并行度为  $P_1$ ，第 8 行的并行度为  $P_2$ ，并行处理所带来的额外复杂度为  $\mathbb{C}_{\text{extra}}$ ，则并行版本的解码器复杂度大致为，

$$\mathbb{C}_{\text{parallel}} = O(T \cdot \frac{I}{P_2} \cdot \frac{K}{P_1}) \cdot \mathbb{C}_{\text{extra}} \quad (5-2)$$

因此并行算法理论上的最大加速比为  $(P_1 \cdot P_2)$ 。下面的章节将具体展开一些针对并行处理所需要的特殊设计，以提升并行效率（提高  $(P_1 \cdot P_2)$ ），降低同步损耗（减少  $C_{extra}$ ）。其中负载均衡算法处理了线程并行之间的任务分配，它们具体体现在如何分配 WFST 状态和边。

## 5.2 并行的令牌传递算法

下面我们首先介绍维特比搜索中将会引入的同步时间开销，它具体出现在令牌合并过程中。在 ASR 解码中，维特比搜索被实现为 令牌传递算法 [166]，其令牌表示的是截止  $t$  帧情况下的一个局部识别序列，同时对于每一个 WFST 状态在  $t$  帧时都可以用一个可移动的令牌进行表示。在每一帧，维特比路径是通过 令牌合并过程得到的，其它一个  $min$  操作被加到了每个状态的所有输入边上（比如对状态 7，在图 5-2 中，那么它的输入边包括 2 至 7,5 至 7,7 至 7 三条），这里我们需要计算得到它们之中的最佳分数，以及这个分数来自哪一个状态。

---

**算法 5-2** 线程级别的令牌合并算法 (Inputs: accumulated cost, an out-going WFST arc and a current token)

---

```

1: procedure RECOMBINE(cost, arc, curTok)
2:   oldTokPack = state2tokPack[arc.next_state]
3:   curTokPack = packFunc(cost, arc.id)                                ▷ pack into uint64
4:   ret = atomicMin1(oldTokPack, curTokPack)
5:   if ret > curTokPack then                                         ▷ recombine
6:     perArcTokBuf[arc.id] = *curTok                                     ▷ store token

```

---

原始的 CPU 算法在进行令牌合并时（算法 3-1 的 12-18 行）是串行的。我们即将讨论如何在 GPU 中实现令牌合并，而将如何高效并行这些指令放在下面的章节中。一种最简单的实现是通过添加临界区（critical sections）[168] 以便使令牌合并这一步仍然恢复串行。这种设计不仅低效，而且会在早期 GPU 中引入死锁。[136] 提出一种在每个状态下做针对所有令牌传递结果的规约操作，但这样将引入额外的规约过程中的写冲突和同步损耗，而且这些损耗总是发生在计算的最后阶段。[169] 提出将所有数据编码到 32 比特上，然后使用 GPU 原子操作来实现令牌合并。这种方法损失了计算精度，并且使解码算法受制于特定的模型。

---

<sup>1</sup>  $atomicMin(*address, val)$  [167]. Computes  $min(*address, val)$ , writes the result to  $address$ , and returns the original  $*address$ .

我们提出算法 5-2, 它是一个通用的计算方法, 适应于任何模型的令牌合并, 没有精度损失。这个算法执行于每一个 GPU 线程上, 并且针对每一条 WFST 边进行并行, 比如在图5-2 中从状态 2 到状态 7 的边可以由这一算法进行处理。我们首先将分数和边的索引合并为一个 64 比特整形数, 以表示合并之前的令牌, 同时我们将分数放在高位上, 使得它可以控制后续的合并结果。在每一帧, 我们将所有令牌的信息保存在一个长度为所有可行边的数组上。这保重合并过程没有写入的冲突, 也就不会带来同步损耗。所有的令牌都被处理后, 我们收集所有仍然活跃的合并令牌, 将 64 比特数字返回原先的分数和边的索引, 由此将真正的令牌信息存入相应的令牌中, 而这一步也同样是并行进行的。

### 5.3 图搜索的动态负载均衡算法

另一个并行算法的问题是负载均衡。对每一个状态, 我们并行地遍历所有它的输出边, 直到到达了一个终止节点。由于 WFST 状态可能具有完全不同的输出边数, 如果不能合理地将线程分配给这些边, 将导致负载不均问题。文献 [136, 170] 中重新设计了 WFST 结构以减少这种不均。不同于这些工作, 我们不希望解码算法依赖于特定的声学, 词典, 语言模型。因此我们依然基于原始的 WFST 结构。我们首先提出静态负载均衡算法, 它首先对每个线程计算如何分配 WFST 边才能得到大致相等的数量, 经过那之后才开始进行处理。这样的设计引入了额外的并行求取累积和的开销<sup>1</sup>, 这种方法将引入计算和 GPU 内核启动的额外时间开销。

受 [171] 的启发, 我们进一步引入了动态负载均衡算法, 其呈现在算法 5-3 中。我们使用一个调度中心来分配令牌, 同时我们令  $N$  个线程为一个组 ( $N = 32$ ) 来共同处理从一个状态 (令牌) 出发的所有 WFST 边。当一个令牌的所有边都被处理完毕了, 它就会找调度中心取到下一个令牌。我们同时将调度中心实现为一个 GPU 原子操作。图 5-2 显示了其中的一个例子。在实验中我们比较了两种不同的负载均衡算法。

### 5.4 并行的词图处理算法

#### 5.4.1 精确的词图生成算法

WFST 精确词图 [122] 是指在词图上保存了准确的分数和状态级对齐关系, 这些信息对于语言模型重打分, 丰富的语音识别后处理等都有重要作用。<sup>2</sup>。但是在 GPU 中实

<sup>1</sup>具体来说是一个 GPU 上的 DeviceScan 操作, 其工作在大约 10K 个整形数上。

<sup>2</sup>除了本文中研究的置信度应用 [172], 它同样可以用于加速最小贝叶斯风险解码 [20], 系统融合 [173], 鉴别性训练 [22], 等等。

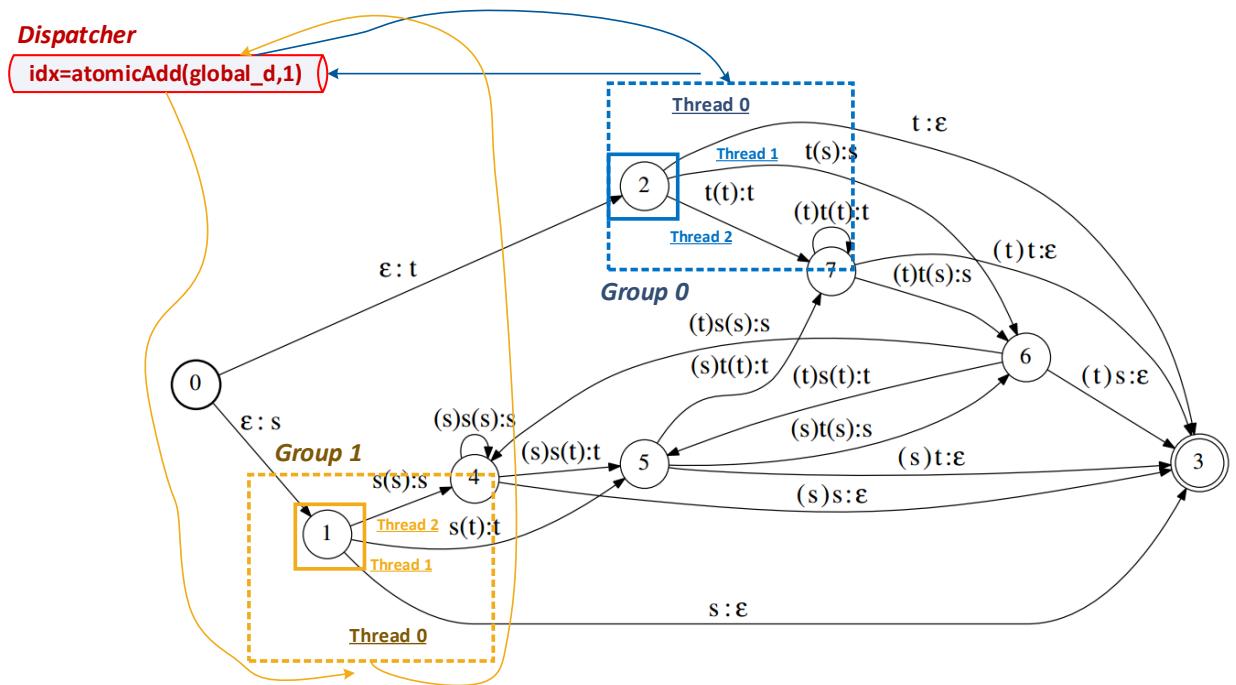


图 5-2 一个针对动态负载均衡的例子。这里的虚线框表示一个 CUDA cooperative group，不同的组分表示为不同的颜色。每一个组由线程 0 进行控制。当某个组分处理完了从一个状态出发的所有边，线程 0 将会向调度中心索要下一个令牌，并通知组分里的其它线程。而调度中心使用原子操作来保证每个令牌只分配给一个组分。组 0 和组 1 完全工作在并行方式中。

Fig 5-2 An example of dynamic load balancing.

---

**算法 5–3 Grid 级别的令牌传递算法 (N=32; Inputs: the current active token vector)**

---

```

1: procedure DYNAMIC LOAD BALANCING (toks)
2:   group = cooperative_groups::tiled_partition<32>
3:   if group.thread_rank() == 0 then                                ▷ rank 0 in each group
4:     i = atomicAdd(global_d, 1)                                     ▷ allocate new tokens
5:     i = group.shfl(i, 0)                                           ▷ rank 0 broadcasts i to whole group
6:     if i >= sizeof(toks) then return                                ▷ all tokens processed
7:     for arc in tok2arcs(toks[i]) do                                ▷ thread parallelism
8:       call Recombine(toks[i].cost+arc.cost, arc, toks[i])

```

---

现词图处理算法并不简单。[174] 提出在 GPU 中解码，而后在 CPU 中生成词图，这不仅引入新的同步损耗，使解码器速度下降，而且引入了大量的 device-to-host (D2H) 内存拷贝问题。我们通过重新设计了一个并行版本的词图处理算法 [122] 更彻底地解决了这个问题。

在词图生成中，对每一个令牌传递操作，会有一条词图边被记录到 GPU 显存中。我们设计了一种 GPU 向量结构来保存这些边。这种向量实现了 *push\_back* 算法，其通过原子操作来分配特定内存给相应词图边，而整体内存则被预先分配完毕，增量扩展。

为了减少原子操作所带来的损耗，我们提前分配好  $K$  个向量，并随机选取一个向量执行 *push\_back* 操作。经过令牌传递之后，这些来自  $K$  个向量的边被收集起来。对于词图节点，我们可以使用类似的方案<sup>1</sup>。

### 5.4.2 词图剪枝

并行词图剪枝算法基于引文 [175, 122] 中方法，并针对 GPU 并行处理进行了必要的修改。

原始的 CPU 版本词图剪枝算法会迭代地更新节点和边的额外权重，直到它们停止改变；而额外权重定义为当前边的最佳权重与最佳路径上的权重的差值。对于一些权重过高的边，将会被剪枝，直到一个节点上的所有边都被剪枝。在 GPU 版本实现中：i) 我们将迭代更新节点和边的过程进行了 GPU 线程的并行化；ii) 我们使用一个全局的边向量数组来代替原始实现中的链表结构，原因是链表结构并不能被随意访问 (random access)，而是总要从头进行访问；iii) 我们将额外权重迭代更新实现为一个原子操作，以减小同步时间开销。

---

<sup>1</sup>在我们早前的实验中，在  $K = 32$  时，这一优化可以 10 倍加速词图边处理的速度。这项加速针对词图节点并不明显。

当一条词图边被剪枝后，我们不会物理上删除这条边，原因是内存分配时间开销较大。相反，我们在剪枝的最后进行并行收集操作，将所有剩余边一次集中到 GPU 向量结构中，类似于第 5.4 章中的做法。这种实现方法的内存开销将会在实验中进行讨论。同时，我们没有裁剪词图的节点，原因是：i) 我们需要词图边上能够寻迹到词图节点上，并且在 D2H 拷贝前后都要有唯一对应，因此我们没有改变节点的相对位置。ii) 在 CPU 中会重新建立起节点和边的链表关系，因此没有边连接的节点将被隐含删除。iii) 节点的 D2H 拷贝是与解码和词图处理异步的，因此不会对最终速度造成影响。

本文提出的词图处理算法总结如下：

---

**算法 5-4 Grid 级别的词图处理算法 (Input: processing frame, lattice token vector and lattice arc vector are taken as inputs)**

---

Function  $ArcExtraCost(arc)$  returns the cost difference between the best path including the arc, and the best overall path.

```

1: procedure PRUNE LATTICE FOR FRAME (f, toks, arcs)
2:   for tok in toks(f-1) do                                ▷ extraCost initialization
3:     tok.extraCost = FLT_MAX
4:   while modified == 1 do
5:     modified = 0
6:     for arc in arcs(f) do                                ▷ thread parallelism
7:       cost = ArcExtraCost(arc)
8:       if cost < latticeBeam then
9:         atomicMin(tok.extraCost,cost)
10:        atomicAdd(modified,1)

```

---

## 5.5 实验结果

### 5.5.1 实验配置

在 Switchboard 300 小时数据集上，我们测试了两个声学模型基线系统：一个是文献 [107] 中的“TDNN-LSTM-C”模型，其使用无词图鉴别性训练准则 (LF-MMI) [83]，包含了降帧率技术。我们同时测试了未降帧率模型，它是一个多层 BLSTM 模型 [64]，使用交叉熵进行训练。

测试是在 NIST 2000 CTS 集合上进行的。一个 30K 词表大小，从 Switchboard 数据

集标注训练的 3 元语言模型被用于第一遍解码，与 Fisher 数据集插值的 4 元语言模型被用于词图重估打分。Kaldi 的 1-best 解码器和词图解码器被用作 CPU 解码器的基线系统<sup>1</sup>。

我们对 1-best 结果和词图质量都进行了比较。为了公平起见，我们保持两种解码器得出相同的词图密度 (*lat.den.*, 其使用 arcs/frame 进行度量) [166]。在语音识别任务中，我们使用词错误率 (WER), 词图重估错误率 (lattice rescored WER) (+rescored), 词图最佳错误率 (lattice oracle WER, OWER) [176]。归一化交叉熵 (NCE) [177] 用来作为词图所得到的词级别置信度质量的评估，在大词汇连续语音识别中。越大的 NCE 表示模型性能越好。这里我们训练了决策树以将这些词图后验概率转换为我们最终需要的置信度 [178, 179]。实时率 (real-time factor, RTF) 用于度量解码器速度和相应针对 CPU 基线系统的加速比 ( $\Delta$ )。由于本文专注在 WFST 解码的加速上，因此我们所报道的 RTF 排除了声学模型的推理搜索所占用的时间。下文中我们会对总体的 RTF 稍作讨论。由于本文主要考虑快速的离线转录系统，因此时延并未加入比较。所有的实验都是在 E5-2686 v4 @ 2.30GHz 的 CPU 上进行，其包含 1 socket (8 threads)。一块 Tesla V100 默认地用于 GPU 解码，后续我们还会比较其它 GPU 架构下的性能。

### 5.5.2 性能和速度

表 5-1 中比较了我们所提出的 GPU 词图解码器在使用相同 beam 大小情况下的准确度 (性能) 比较。结果中所有的指标都非常接近。在词图结果中的轻微差别来自于词图生成过程中的不同访问次序。

表 5-1 1-best 和词图性能比较 (beam=14).

system	<i>lat. den.</i>	WER	+rescored	OWER	NCE
CPU	30.3	15.5	14.3	11.2	0.322
GPU	30.2	15.5	14.3	11.2 <sup>2</sup>	0.328

解码的实时率和加速比的比较参见表 5-2。在 1-best 和词图解码中，我们分别取得了 15 倍和 9.7 倍的加速。这里的大部分加速来自于平行处理 WFST 的节点和边的过程。当我们使用 GPU 的并行技术 MPS 时，我们可以得到 46 倍的序列并行加速比。MPS 是一种减小 GPU context 切换损耗的技术 [167]。作为对比，我们在 CPU 上使用了类似的序列并行技术，但仅取得了 1.8 倍的加速比。

<sup>1</sup>Kaldi 中的 *decode-faster-mapped* 和 *latgen-faster-mapped* 工具

虽然原子操作优化的加速比并不明显，它有效去除了 critical section，而这部分使得使用传统 GPU 框架进行解码成为可能，这部分我们将在后文中讨论。同时动态负载均衡取得了最好的实验效果。

为了更公证地比较总体的 RTF，我们使用一个 GPU 独立进行单序列声学模型推理。单序列的总体 RTF 在 1-best 解码器和词图解码器上分别为 0.18 和 0.03<sup>1</sup>。更深层次地将声学模型推理和语言模型解码相结合，是一个有意义的未来研究课题。

表 5-2 所提出系统的速度比较 ( $beam=14$ )。

system	search		+ lattice	
	RTF	$\Delta$	RTF	$\Delta$
CPU	<b>0.16</b>	<b>1.0X</b>	<b>0.27</b>	<b>1.0X</b>
+ 8-sequence (1 socket) <sup>2</sup>	-	-	0.15	1.8X
GPU	0.016	10X	0.080	3.3X
+ atomic operation	0.015	11X	0.077	3.5X
+ dyn. load balancing	<b>0.011</b>	<b>15X</b>	0.075	3.6X
+ lattice prune	-	-	<b>0.028</b>	<b>9.7X</b>
+ 8-sequence (MPS)	0.0035	46X	0.0080	34X

### 5.5.3 分析

图 5-3 显示在各种不同的语音识别系统上，该解码器能够取得一致的显著加速。

- GPU 架构比较。

该系统可以工作在 Kepler 之后的 GPU 架构中。在非常早期的 K20 GPU 上它仍然取得了 3 倍相比 CPU 的加速比。

- 声学模型帧率

我们测试了原始帧率而非降帧率后的声学模型，用图中的虚线表示。它仍然显著比 CPU 的降帧率系统好。值得注意的是这说明 CPU 系统比 GPU 系统慢了 3 倍以上，因为降帧率技术同样非常重要，可以显著加速解码系统 [108, 110]。而这项技术在 GPU 解码器中仍然很重要。

<sup>1</sup>这里的声学模型推理还可以使用序列批处理进行额外加速 [135]，我们以往的经验是有 10 倍左右的加速比。

- 语言模型大小和内存开销。

这里我们测试了 Fisher 数据集插值后的 4 元语言模型，并将它剪枝到不同大小后编译成 HCLG [36] 进行测试 (13MB, 62MB, 196MB, 258MB)。这里的 CPU 解码器一致性地慢于 GPU 系统。

另一方面，我们使用一个 11GB WFST 在 12GB 显存的 TITAN GPU 上测试了该 GPU 解码器。这种大小的 WFST 基本可以满足商用需要。更多关于内存优化的细节可以参考我们最新发布的开源代码。

除此之外，我们发现当 CPU 解码器使用的语言模型大小变化之后，其速度变化相对 GPU 略微小一些，原因是 CPU 解码器会进行逐边剪枝；而 GPU 解码器则是进行统一的剪枝。因此针对 GPU 解码器的剪枝策略的研究也是未来一个很有意义的课题。

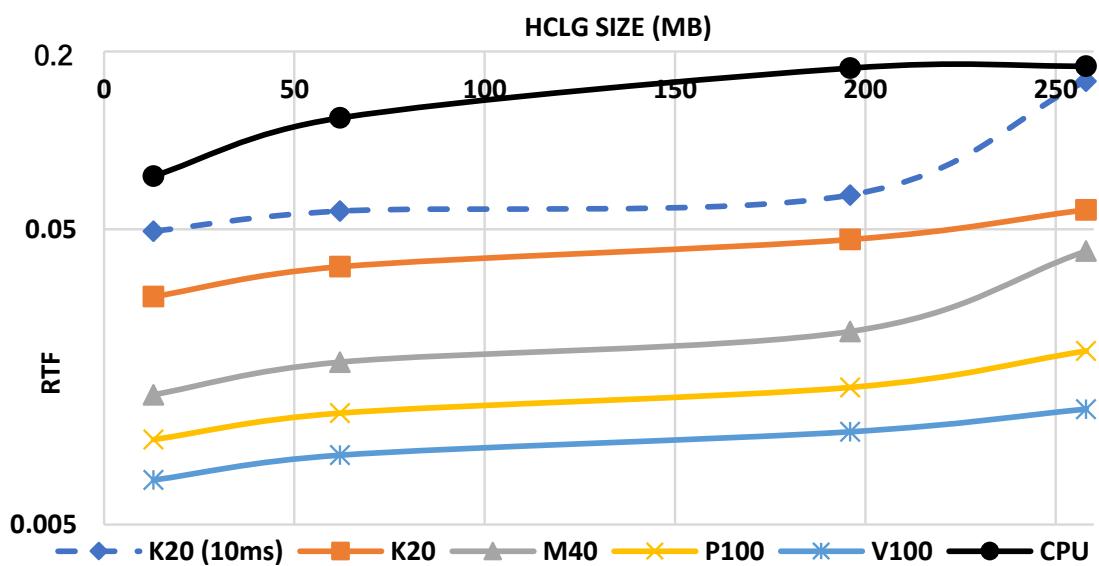


图 5-3 语言模型大小，帧率，GPU 架构的比较

Fig 5-3 The comparison between different LM sizes, AM frame rates, GPU architectures.

- 计算时间占比分析。

图 5–4 比较了 CPU 和 GPU 的各个子模块的时间占用<sup>1</sup>。GPU 在显著快于 CPU 版本的同时，每部分分布比较均衡，显示出系统并没有显著的瓶颈。GPU 解码器的令牌传递和词图处理都得到了显著加速。同时“other”部分包含了 GPU 的内核启动时间和同步等待时间。对于这些部分的优化也是未来研究的一个有意义话题。

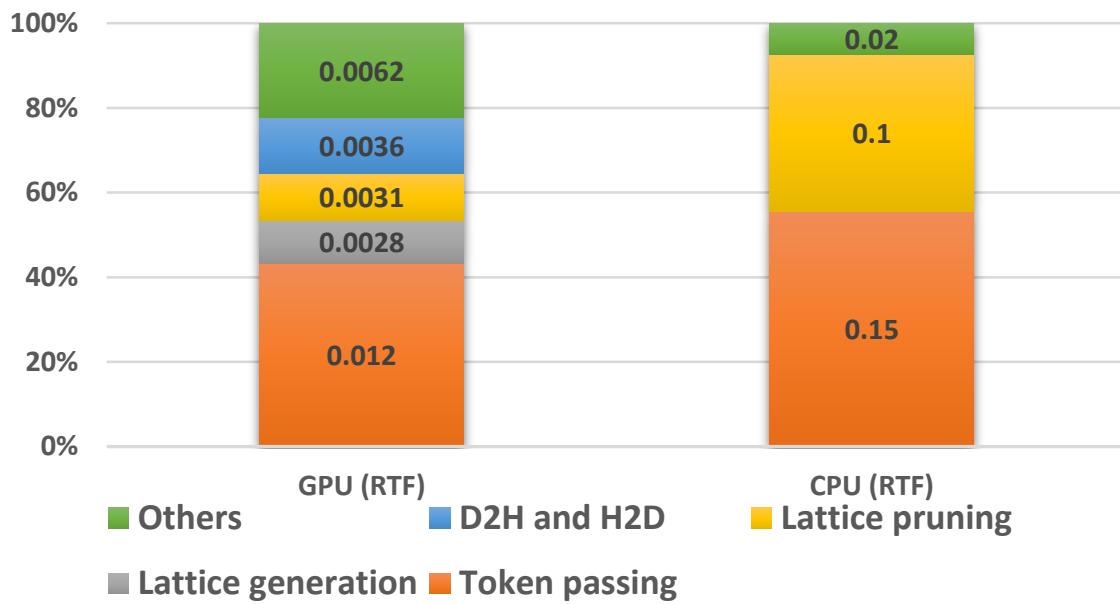


图 5–4 针对解码器的时间占比分析

Fig 5–4 The computational profiling analysis of the proposed decoder.

## 5.6 本章小结

在这项工作中，我们描述了所提出的基于 GPU 并行计算的 WFST 解码器，我们将该实现融合在 Kaldi 开源工具包中。我们设计了并行版本的解码和词图处理算法。该系统相比 CPU 版本取得了显著的性能提升，并且该实现可以推广到大部分 GPU 架构中。最后，我们还将该实现完全开源。

<sup>1</sup>词图生成在 CPU 中被包含到令牌传递部分中



## 第六章 基于标签同步解码的搜索空间优化

本论文基于端到端建模，系统地提出了标签同步算法，其通过一系列方法使得搜索解码过程从逐帧同步变为标签同步，从而加速解码。

为了对语音识别中的时序特征进行建模，人们提出了序列模型。根据其建模过程，序列模型包含 **GSM** 和 **DSM**，在第2.3章节进行了介绍。通常来说，出于以下原因，**GSM** 和 **DSM** 被分解为帧层面的训练准则：1) 为了更加高效地发挥帧层面分类器的建模效果，如混合高斯模型 (*Gaussian mixture model, GMM* [53]) 和深度神经网络 (*deep neural network, DNN* [77])；2) 为了减轻模型的稀疏性，以及通过将简单模型分解为多个组分来增强模型的泛化能力，例如 **ASR** 中将模型分解为声学模型、字典和语言模型等；3) 未经序列分解的模型在推理搜索前需要得到整个序列信息进而进行后续处理，这将给解码过程造成严重的运行延时。本文提出的序列标注方法即是基于帧层面分解的模型 [37, 36]。

在推理搜索阶段，为了找到与输入特征最为匹配的标签序列，搜索过程需要将声学模型，语言模型和字典等模型结合起来。该过程是通过在每帧使用基于束剪枝的维特比算法来实现的 [37]，称为帧同步解码 (*frame synchronous decoding, FSD*)。在该框架中，我们将特征帧的数量和语句长度的比值定义为特征速率，将标签输出数量与语句长度的比值定义为标注速率，将解码的帧数与语句长度的比值定义为解码速率。那么，在帧同步解码中，这三个速率均相等。

虽然已经被广泛使用，但帧同步解码仍存在一些缺点：1) 这是一个等间隔搜索算法，在处理可变长序列时较为低效。2) 由于序列被分解为帧来作为特征序列，模型的粒度变小，导致搜索空间很大。比如 **ASR** 中词语历史，音素序列，以及 **HMM** 状态之间的关联性通常以加权有限状态机 (*weighted finite-state transducer, WFST*) 进行表示（通常称为 **HCLG** [36] 搜索空间）。由于由多个庞大知识源共同组成，因此组成该搜索空间的状态机最终达到百亿条边。3) 在每帧进行贪心束剪枝通常很难兼顾搜索效率和搜索误差。

如第 3.4.2 章节中对于解码搜索的研究机遇的讨论，近来神经网络的发展使得更强的上下文和历史建模效果成为可能 [64, 31]。同时，更多的标注数据也进一步缓解了模型的稀疏性和泛化问题。这些进展使得研究人员们有可能在更大的模型粒度上-从帧到整个序列层面 [137, 138, 139, 140, 78] 进行序列分解，如 Soltau 等人报道的一个基于单词粒度深度学习的声学模型 [138]，在 125K 小时标注数据上的表现优于较小粒度的模

型。在这些研究中，标注速率小于特征速率，但解码速率仍然等于特征速率。

本文提出将特征层面的搜索过程改变为标签层面，即搜索空间是由不同历史的标签组成的，使得解码速率等于标注速率，从而小于特征速率。具体来说，在标签推理搜索阶段，对帧层面声学模型的输出增加一步后处理过程：i) 判断当前帧是否存在标签输出；ii) 若有，执行搜索过程；若无，则丢弃标签输出。因此该后处理过程可被看作是每个输出标签概率计算的近似。与传统方法相比，该方法的优势是搜索空间更小，且搜索过程被大大加速。该文提出的一系列通用方法在隐马尔科夫模型和连接时序分类模型上得到了验证。

最终在实验部分，该章节系统取得大幅度语音识别解码速度改善。

## 6.1 基于连接时序分类模型的标签同步解码

我们在第2.3.1章节总结了语音识别中的序列建模方法。在模型推理搜索阶段，为了找到与输入特征最为匹配的标签序列，搜索过程需要将前述序列模型与其它知识源，即字典、语言模型等融合起来。即解码标签序列是由前述各分解序列所共同决定的。该搜索过程是通过在每帧上使用基于束剪枝的维特比算法进行的 [37]，即第3.3.2.4章节介绍的传统帧同步解码算法。该框架中，解码速率等于标注速率，标注速率等于特征速率。

尽管被广泛使用，FSD 方法仍有一些缺点：1) 它是一个等间隔搜索算法，处理变长特征序列较为低效。即使在采用第3.3.2.3章节总结的跳帧等分数计算优化算法时仍存在该问题。2) 当序列被分解为帧层面作为特征序列时，模型粒度较小，导致搜索空间很大。3) 在每帧均进行贪心束剪枝，很难平衡搜索效率和搜索误差。因此，本文通过将特征层面的搜索过程改变为标签层面，提出了基于端到端建模的标签同步推理搜索，接下来作者将对该框架及其应用进行详细介绍。

该部分中，本文提出将搜索过程从特征层面改为标签层面，称为标签同步解码 (Label Synchronous Decoding, LSD)。本部分将对 DSM 中的 LSD 进行公式推导，具体实现方案及一些解码加速的经验方案也将进行讨论。

### 6.1.1 标签同步解码

在测试阶段，在基于音素的 CTC 模型中，从公式 3-19 可以推导出公式 3-21。而根据 CTC 中输出标签之间的条件独立性假设， $P(\mathbf{l}|\mathbf{x})$  可以下获得：

$$P(\mathbf{l}|\mathbf{x}) = \prod_{l \in \mathbf{l}} P(l|\mathbf{x}) \quad (6-1)$$

因此在标签级别上，维特比搜索如下所示：

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} \left\{ P(\mathbf{w}) \max_{\mathbf{l}_w} \frac{\prod_{l \in \mathbf{l}_w} P(l|\mathbf{x})}{P(\mathbf{l}_w)} \right\} \quad (6-2)$$

观察图 6-1 中的 CTC 模型输出分布  $P(l|\mathbf{x})$ ，我们可以发现，传统 HMM 模型的输出分布非常平滑。CTC 相比于 HMM 模型，能够给出更突出的概率分布。具体来说在大多数帧上 CTC 模型输出较大的 blank 的概率，只在少数帧上输出较大的非 blank 的音素概率。这一现象符合端到端建模的初衷，即建模只关心最终的音素输出，而不关心每一帧的状态级强制对齐结果。而解码过程中，由于 blank 输出并不会影响后续的词典和语言模型概率，因此解码搜索算法也只关心非 blank 的音素输出概率，即图中的尖峰部分。

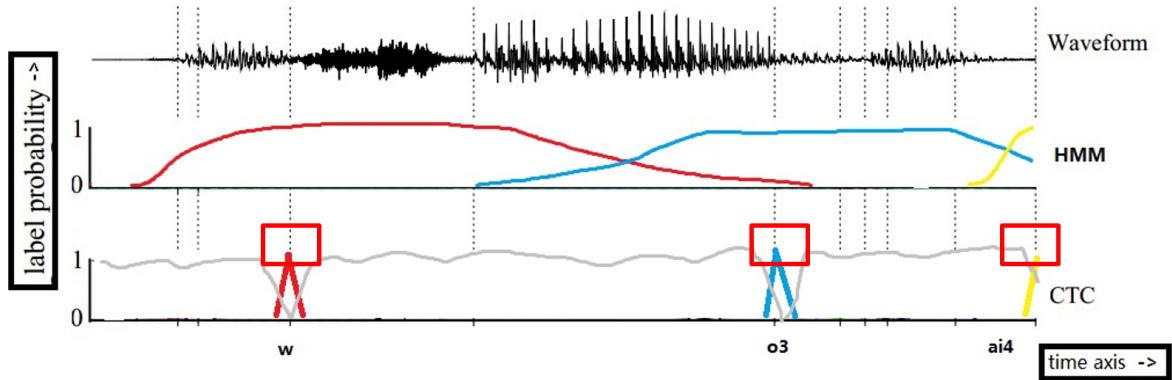


图 6-1 HMM, CTC 的模型输出分布示例。纵轴表示模型输出概率，横轴表示时间。灰色分布表示 CTC 模型的 blank 概率输出

Fig 6-1 The inference distributions of HMM and CTC models.

在音素级的  $P(l|\mathbf{x})$  的计算过程中，本文提出在帧级神经网络的输出上进行一步后处理，以便滤除 blank 输出，仅保留非 blank 的音素输出概率来进行公式 (6-2) 的解码搜索。本文定义公共 blank 帧的集合如下：

$$U = \{u : y_{\text{blank}}^u > \mathcal{T}\} \quad (6-3)$$

其中  $y_{\text{blank}}^u$  是神经网络在第  $u$  帧输出 blank 单元的概率。在 CTC 模型中的 softmax 层，如果 blank 单元的声学得分足够大且接近常数 1，则可以认为所有竞争路径共享相

同跨度的 blank 帧。因此，忽略这些帧的分数并不会影响解码中的声学得分排序。

$$\begin{aligned} P(l|\mathbf{x}) &= \sum_{\pi \in \mathcal{B}(\mathbf{l})} \prod_{\pi} P(\pi|\mathbf{x}) \\ &\simeq \sum_{\pi \in \mathcal{B}(\mathbf{l})} \prod_{\pi \in U} y_{b_l}^u \prod_{\pi \notin U} y_{p_l}^u \end{aligned} \quad (6-4)$$

由于  $\prod_{\pi \in U} y_{b_l}^u \simeq 1$ ，公式6-4可被推导为6-5：

$$P(l|\mathbf{x}) \simeq \sum_{\pi \in \mathcal{B}(\mathbf{l})} \prod_{\pi \notin U} y_{p_l}^u \quad (6-5)$$

### 6.1.2 标签同步解码算法实现

DSM 的标签同步解码算法如算法 6-5 所示。S 和 E 是预编译的 WFST 网络的起始和结束节点。Q 指有效令牌，B' 指解码路径，T 是总帧数。NNPropagate(t) 是每帧的声学模型推理过程。isBlankFrame(F) 用于检测每帧是否为 blank。ViterbiBeamSearch(F, Q) 是 FSD 中的标准维特比搜索算法，但在 LSD 中仅在标签级别执行。finalTransition(E, S, Q) 用于搜索 WFST 的终止节点 [180]。

---

**算法 6-5 DSM 的标签同步维特比束搜索算法** (**Inputs:** 起始节点，结束节点，令牌队列，时间帧)

---

```

1: procedure LSD FOR DSM (S, E, Q, T)
2:    $Q \leftarrow S$                                       $\triangleright$  起始节点初始化
3:   for each  $t \in [1, T]$  do                    $\triangleright$  逐帧神经网络前向传播
4:      $F \leftarrow NNPropagate(t)$ 
5:     if !isBlankFrame( $F$ ) then            $\triangleright$  逐音素解码
6:        $Q \leftarrow ViterbiBeamSearch(F, Q)$ 
7:      $\hat{B} \leftarrow finalTransition(E, S, Q)$            $\triangleright$  到达结束节点
8:     backtrace( $\hat{B}$ )

```

---

## 6.2 基于无词图鉴别性训练的模型的标签同步解码

### 6.2.1 标签同步解码

在 GSM 中，相邻 HMM 之间的输出标签也是条件独立的：

$$P(\mathbf{x}|\mathbf{l}) = \prod_l P(\mathbf{x}|l) \quad (6-6)$$

类似地，在标签级别上进行的维特比搜索如下所示。

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} \left\{ P(\mathbf{w}) \max_{\mathbf{l}_w} \prod_{l \in \mathbf{l}_w} P(\mathbf{x}|l) \right\} \quad (6-7)$$

在标签中， $P(\mathbf{x}|l)$  的计算如下所示：

$$\begin{aligned} P(\mathbf{x}|l) &= \sum_{\pi: \pi \in L', \mathcal{A}(\pi_{1:T})=l} \prod_{t=1}^T P(\mathbf{x}|\pi_t) P(\pi_t|\pi_{t-1}) \\ &= \sum_{\pi: \pi \in L', \mathcal{A}(\pi_{1:T})=l} \prod_{t=1}^T P(\pi_t|\mathbf{x}) \frac{P(\pi_t|\pi_{t-1})}{P(\pi_t)} \end{aligned} \quad (6-8)$$

我们在第4.1.2.1章节介绍了几种本文所扩展的 HMM 拓扑结构，如图 2-8(c-e)。虽然在我们的实验中，这些模型的输出分布比 CTC 中更平滑，但 DSM 中提出的公式 (6-3) 和 (6-4) 可以被扩展到 GSM。

这里提出对神经网络的输出  $P(\pi_t|\mathbf{x})$  进行后处理，其中  $\pi_t$  是帧  $t$  的推理搜索模型单元。由于这些模型中的模拟 blank 状态，公式 6-7 中的维特比束搜索不必包括标签输出候选序列的所有帧。因此，给出某一帧的模型推理搜索分布时，是否从维特比搜索中排除某帧的判决如下：

$$U = \{u : \sum_{l \in L} (y_{p_l}^u - y_{b_l}^u) > T\} \quad (6-9)$$

其中  $y_{p_l}^u$  是帧  $u$  处标签输出状态 1 的神经网络输出， $y_{b_l}^u$  是对应的 blank 状态的输出。在第  $u$  帧是否有标签输出，是由所有 blank 状态与标签输出状态的概率差异的总和决定的。 $T$  是在开发集中得到的阈值。因此， $P(\mathbf{x}|l)$  的计算可以根据  $\pi \in U$  与否分为如下两部分：

$$\begin{aligned} P(\mathbf{x}|l) &\simeq \sum_{\pi: \pi \in L', \mathcal{A}(\pi_{1:T})=l} \{ \\ &\quad \prod_{\pi \notin U} \frac{y_{b_l}^u P(b_l|\mathbf{x})}{P(b_l)} \prod_{\pi \in U} \frac{y_{p_l}^u P(p_l|\mathbf{x})}{P(p_l)} \} \end{aligned} \quad (6-10)$$

其中第一部分是标签输出状态。在该情况下，每个标签输出均在 WFST 中进行维特比搜索。而另外一组的 blank 部分，则假设没有标签输出。不同于 CTC，不同标签输出维护自己的 blank 状态版本。即使是 blank 帧，也可能包含不同的输出标签信息。因此， $\prod_{\pi \notin U} \frac{y_{b_l}^u P(b_l|\mathbf{x})}{P(b_l)}$  的分数不能被丢弃。下文提出一种高效的算法对这一项进行计算。

这里所提出的后处理可以被视为输出标签概率  $P(\pi|\mathbf{x})$  的近似，从而使得维特比束搜索得以在标签级别上进行。

## 6.2.2 标签同步解码算法实现

用于 **GSM** 的标签同步解码算法如算法6-6所示。与算法6-5相比，在每个 **blank** 帧中，输出序列可以包含不同的 **blank** 单元。因此对相邻的 **blank** 帧计算  $\prod_{\pi \notin U} \frac{y_{b_l}^u P(b_l|\mathbf{x})}{P(b_l)}$ 。在非 **blank** 帧中，首先将各个 **blank** 单元各自累积得到的概率得分分别添加到当前帧的所有候选序列分数中，之后再进行维特比搜索算法。这些不同之处在算法中使用红色字体来表示。

---

**算法 6-6 GSM 的标签同步维特比束搜索算法**(**Inputs:** 起始节点，结束节点，令牌队列，时间帧)

---

```

1: procedure LSD FOR DSM (S, E, Q, T)
2:    $Q \leftarrow S$                                       $\triangleright$  起始节点初始化
3:   for each  $t \in [1, T]$  do                    $\triangleright$  逐帧神经网络前向传播
4:      $F \leftarrow NNPropagate(t)$ 
5:     if !isBlankFrame( $F$ ) then            $\triangleright$  逐音素解码
6:        $F \leftarrow addAccumulatedBlankScore(V, F)$ 
7:       reset( $V$ )
8:      $Q \leftarrow ViterbiBeamSearch(F, Q)$ 
9:   else                                 $\triangleright$  accumulate blank scores
10:     $V \leftarrow accumulateBlankScore(V, F)$ 
11:     $\hat{B} \leftarrow finalTransition(E, S, Q)$            $\triangleright$  到达结束节点
12:    backtrace( $\hat{B}$ )

```

---

除此之外，在 **HMM** 拓扑结构方面，本文将第4.1.2.1章节提出的图2-8 (d-e) 所示的几种改进结构都应用在了 **GSM** 中。

本文在维特比搜索中除了使用传统的束剪枝算法 [37] 和直方图剪枝算法 [181] (自适应束剪枝 [182]) 之外，提出了另外两种剪枝方法。在 **LSD** 中，**blank** 帧占总帧数的百分比与加速比成正比，而 **blank** 帧通过公式进行判定。作为束剪枝算法的变体，这里提出了基于 **blank** 帧阈值  $T$  的剪枝算法，称为 **blank** 剪枝。当阈值  $T$  固定时，推理搜索分布的尖峰属性决定了加速比，而尖峰属性显示了神经网络输出分布的置信度。在神经网络的模型训练阶段，本文又提出了基于假设剪枝的熵剪枝算法。在文献 [183] 中，作者通过惩罚确定的输出分布来防止过拟合和提高神经网络的泛化能力。受这项工作的启发，我们在 **LSD** 框架中对输出分布的熵进行了控制，作为候选序列的剪枝方法。具体来说，在模型训练中将输出分布的熵添加到负对数似然  $\mathcal{L}(\theta)$  中，公式如下所示：

$$\mathcal{L}(\theta) = -(p_\theta(\pi|\mathbf{x}) - \beta H(p_\theta(\pi|\mathbf{x})) \quad (6-11)$$

其中  $H(\cdot)$  是输出分布  $p_\theta(\pi|\mathbf{x})$  的熵， $\beta$  是正比例因子。与文献 [183] 不同的是，基于熵剪枝算法的训练目的是最小化模型的原有训练准则以及输出分布的熵。而通常情况下，基于熵剪枝算法是基于一个已经训练好的模型对参数进行微调。在使用新的准则训练之后，LSD 框架可在少量性能损失的情况下得以加速。在接下来的实验部分，本文将详细比较这四种剪枝方法。

### 6.3 逐帧同步解码与标签同步解码的对比

本文提出将特征层面的搜索过程改变为标签层面，即搜索空间是由不同历史的标签组成的，使得解码速率等于标注速率，从而小于特征速率。具体来说，所提出的 LSD 的解码复杂度如下：

$$\mathbb{C}_{LSD} = O(T - |U|) \cdot \mathbb{C}_{frame} \quad (6-12)$$

其中空白帧的数量  $|U|$ ，总是接近于  $T$ 。对比公式 (3-24) 和公式 (6-12)，FSD 得到了很大的加速。这里将 FSD 和 LSD 之间的主要区别总结如下：

- 不同的信息率。在 FSD 中，声学和语言信息均在每帧进行处理，使得二者的处理速率均和声学特征的帧率相同。而在 LSD 中，声学信息是以声学特征的帧率进行处理的，而语言信息则按声学模型推理的标注速率进行处理。声学和语言信息处理的不同速率去除了大量的搜索冗余。
- 可调整的搜索间隔。在 FSD 框架下，WFST 网络是以等间隔遍历的（即使带有跳帧的深度神经网络在解码 [184] 时是以更长的间隔遍历语言搜索空间，但其间隔仍然是相等的）。而在 LSD 中，搜索间隔可通过灵活的自我调整（在不造成性能下降的前提下）来去除 blank 帧带来的语言搜索空间搜索冗余，这给解码带来了很大的效率提升。

### 6.4 实验结果

本文实验使用 300 小时的英语 Switchboard 数据集作为训练数据 [163]，使用 NIST 2000 CTS 作为测试集，对 NIST 2000 CTS 测试集所包含的 switchboard（称为 swb）和

call-home (称为 callhm) 两个子集分别进行了评估。在所有实验中使用的是经过工程优化的标准 WFST 解码器；实验过程中没有生成词图，也没有使用语言模型重打分 [122] 技术。解码过程中使用在 Switchboard 和 Fisher 转录文本上训练的插值的 4 阶语言模型；在 DSM 算法验证中，默认使用了经过剪枝的 3 阶语言模型；在 GSM 算法验证中，默认使用 4 阶语言模型，使得结果与文献 [83] 具有可比性。解码使用的机器配置为 Intel (R) Xeon (R) CPU E5-2690 v2 @ 3.00GHz。

本文的 DSM 实验中，使用具有 1.2 M 参数的小型 CTC 模型，使得其适用于嵌入式设备，与文献 [86] 可比；使用 40 维的对数滤波器组特征，特征提取窗宽为 25 ms，帧移为 10 ms；使用 46 个单音素作为声学建模单元；声学模型使用 3 层带有投影层的长短时记忆网络，每层包括 400 个节点并通过投影层压缩为 128 个节点 [64]；使用 EESEN [185] 作为训练工具，训练过程与文献 [186] 相似。

本文的 GSM 实验是在一系列由 KALDI 流程 [150] 训练的基于 HMM 的大型模型上进行的，这些模型均适用于服务器应用。声学模型建模单元是上下文相关音素。为了提升解码性能 [108, 83]，相对于输入层特征 10 ms 每帧的帧率，将输出层的输出帧率下降到 30 ms 每帧。声学模型分别使用每层 625 个节点的 7 层时延神经网络 (TDNN)；以及 3 层带有投影层的双向长短时记忆网络 (BLSTM)，其中每层的前向后向层均具有 1024 个节点，并通过投影层压缩为 256 个节点。

本文的模块化训练策略的基线系统是一个无模块化训练的端到端系统，也即直接的声学到词语 CTC 模型，类似于 [111]，它使用与音素 CTC 相同的架构，除了在输出层上使用了 30K 大小的词表作为推理搜索单元。它使用了音素 CTC 进行初始化。

本文实验过程中，使用词错误率 (word error rate, WER) 来评估不同解码框架下的模型性能，使用搜索过程中的实时率 (real time factor (RTF) of the search process, SRTF) 和每帧中的活动令牌的平均数量 (active tokens, #AT) 来评价搜索速度。在降低帧率的声学模型中，#AT 使用降帧率之前的帧数进行计算；SRTF 指解码时间与音频时间的百分比。值得注意的是，这里的解码时间不包括神经网络传播的时间 [136, 187]。本文所提出的框架主要加速搜索过程而非神经网络传播，因此不针对不同声学模型的计算速度进行比较，同时这里采用 SRTF 而不是 RTF 来评价搜索速度。由于在维特比搜索的搜索迭代过程-即令牌传递算法 [188] 中，搜索迭代速度与有效令牌的数量相关，因此搜索速度评价指标 AT 与 SRTF 总是正相关。为了更加清晰地对比结果，我们还提供了上述指标的相对变化率 ( $\Delta$ ) 作为参考。

### 6.4.1 DSM 实验

(1) 加速: 表 1 给出了 CTC 模型下, LSD 系统相对 FSD 系统的加速对比。基于 FSD 的 CTC 模型是基线系统。在我们的这项工作中 [189] 曾进一步对比了 CTC 模型的性能和基于 HMM 的系统的性能。

表 6-1 *LSD versus FSD in DSM*

测试子集	解码性能		搜索加速			
	FSD $\rightarrow$ LSD		FSD $\rightarrow$ LSD		FSD $\rightarrow$ LSD	
	WER	$\Delta(\%)$	SRTF	$\Delta(\%)$	#AT	$\Delta(\%)$
swb	18.7	+0.5	0.075	-71	2221	-77
callhm	33.3	+0.0	0.073	-70	2211	-77

swb 测试子集中, 在词错误率相对损失不到 0.5% 的情况下, 与 FSD 框架相比, LSD 框架实现了相对 70% 以上的 SRTF 下降 (也就是 3.4 倍的解码加速)。解码加速主要来自于解码过程中减少了搜索迭代次数, 解码过程中的活动令牌的数量也体现了这一结果。同时在 callhm 子集的实验中也能观察到一致的加速效果。

(2) 速度鲁棒性: 上面的实验解码过程中都使用一个中等大小的语言模型 (3 阶, 3.1 M 语言模型), 为了测试 LSD 框架相对于 FSD 加速效果的鲁棒性 (也即对复杂的语言搜索空间下的可扩展性), 如图 2 所示, 这里将解码语言模型从 2 阶变大到 4 阶 [ ], 大小从 0.2 M 变大到 4.7 M, 使用每帧中的平均活动令牌数 (#AT) 来评价解码速度。从图 6-2 中可以看出, 随着语言模型的增大, LSD 的 #AT 值几乎没有变化, 而与此同时, FSD 的 #AT 值则明显加速增长。此外, FSD 的 #AT 值总是远远超过 LSD 的 #AT 值。也就是说, LSD 实现的加速对 LM 搜索空间的增加是鲁棒的, GSM 的实验也得出了类似的结论, 因此 LSD 适合应用于更复杂的 LM。

(3) 结合跳帧方法: 该部分对比了跳帧方法下的 LSD 与 FSD 框架, 实现结果表明可以将跳帧方法与 FSD 框架进行结合使用。值得注意的是, 在后面的实验中, LSD 也可以应用于跳帧或降低帧率的 GSM 声学模型中。

跳帧的实现方法类似于文献 [190], 这里使用 LSTM-CTC 的 2 倍跳帧 (FS), 并且在神经网络后验概率输出层上没有根据原始特征帧率补全后验概率, 因此 FS 也可以加速解码过程。该方法在没有性能损失的情况下, 应用于 CTC 模型的 FS 可以获得近 2 倍的解码性能加速。这与文献 [108] 中的观察结果一致, 并且在 LSTM-HMM[190] 和 DNN-HMM[184] 也有类似的结果。LSD 可以进一步与 FS 组合, 并且获得更高的效率, 即在搜索过程中进一步减少 57% (累计为 78%) 的时间。

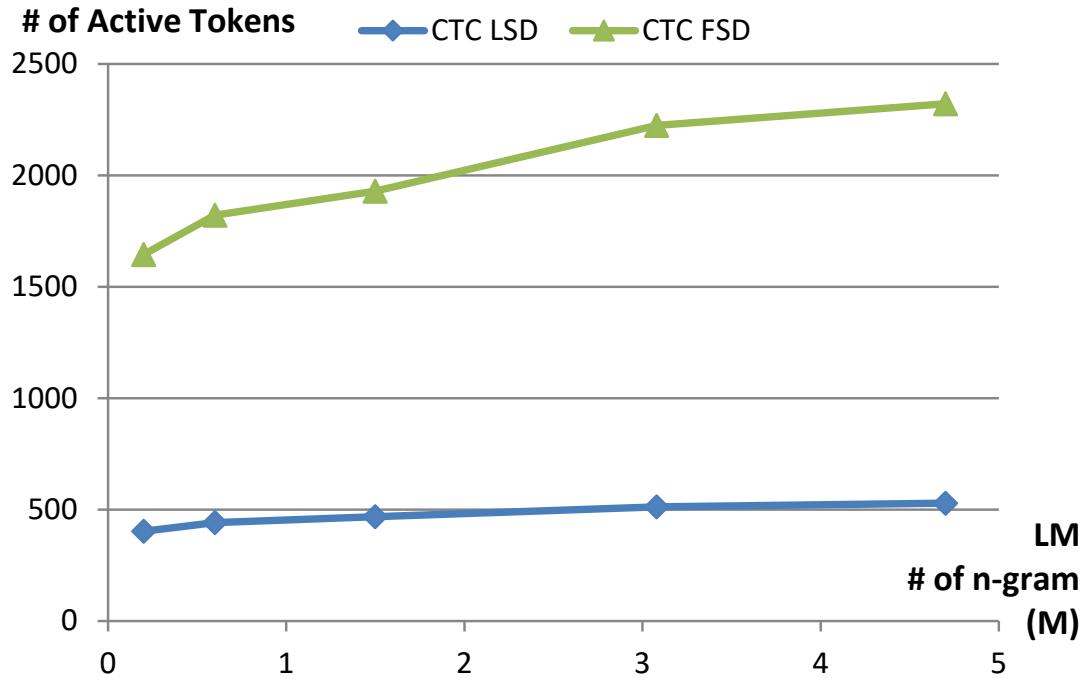


图 6-2 LSD 和 FSD 框架中平均活跃令牌数随 LM 变大的变化趋势。为清晰起见，这里仅绘制 swb 子集，callhm 子集具有类似变化趋势。

Fig 6-2 The trend of average active token numbers and the sizes of LMs in LSD and FSD based frameworks.

表 6-2 LSD 与跳帧方法的对比

测试子集	解码性能		搜索加速		
	FSD $\rightarrow$ FS+LSD		FSD $\rightarrow$ FS $\rightarrow$ FS+LSD		
	WER	$\Delta(\%)$	SRTF	$\Delta_{FS}$	$\Delta_{+LSD} (\sum)$
swb	18.7	-1.6	0.075	-48	-57 (-78)
callhm	33.3	-0.6	0.073	-47	-57 (-77)

(4) 候选序列剪枝：图6-3比较了本文提出的 LSD 框架与传统的剪枝技术，即束剪枝（表示为 beam）和直方图剪枝（表示为 histogram）。在 LSD 中，通过调整公式6-3和6-9中定义的 T 来调整加速比和性能，这也可以被视为另一种候选序列剪枝方法（表示为 blank）。上文中提出的基于熵剪枝算法表示为 entropy。

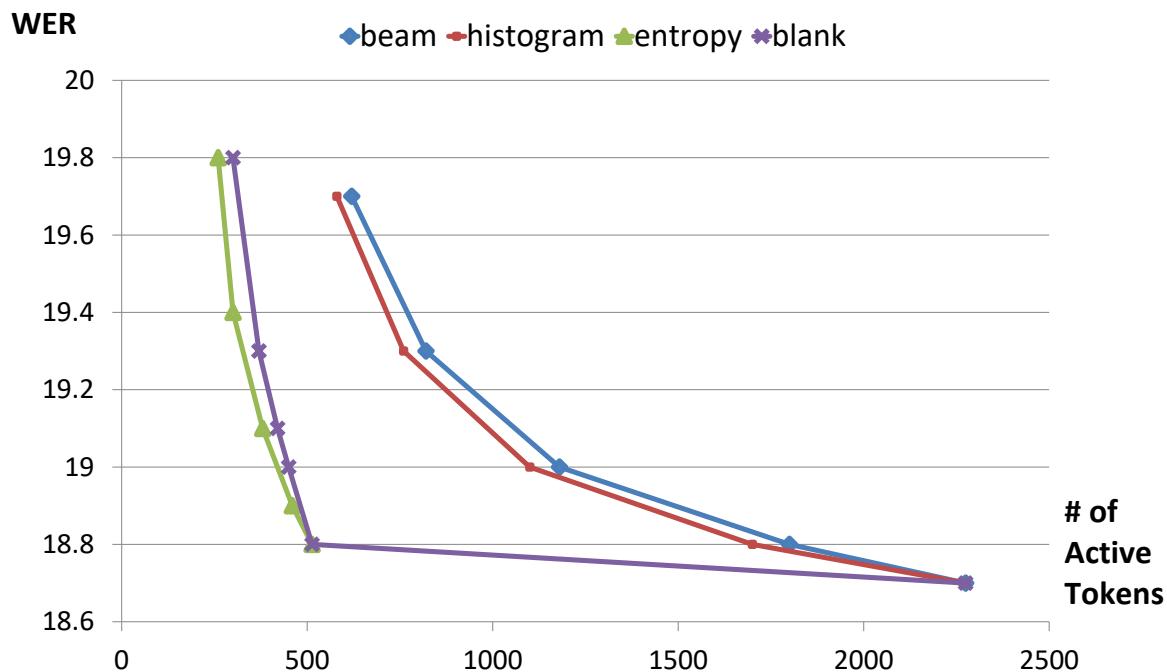


图 6-3 swb 子集，CTC 中，使用不同剪枝技术时 WER 随平均活跃令牌数的变化趋势。calhm 子集结果类似

Fig 6-3 The trend of WER v.s. the number of active tokens using different pruning methods in CTC and swb-subset.

从图中可以看出，基于 LSD 的方法，entropy 和 blank，与基于 FSD 的方法 beam 和 histogram 相比保持显著优势。其原因在于，基于 FSD 的候选序列剪枝对所有候选序列进行统一处理，而相反，LSD 框架可以看作是将候选序列划分为特征级别和标签级别。因此，如公式6-2及公式6-3所示，LSD 框架中，可以在特征层和标签层进行剪枝。也即，基于 LSD 的候选序列剪枝方法受益于将搜索过程从特征层转移到标签层。

另外结果显示，为了加速解码，两种框架中的方法都会降低解码性能，而且基于 LSD 的方法解码性能下降更严重。如前所述，基于 FSD 和 LSD 的方法之间的关键区别在于后者的阈值 T 仅与特征层上的候选序列剪枝有关。特征层候选序列剪枝的加速

比几乎是固定的，在不损害性能的情况下，70-80% 的候选序列可以被剪枝掉。图6-3中 blank 曲线具有明显的拐点 (#AT = 513, WER = 18.8)，也源于同样的原因。在图的最右侧，未进行特征层剪枝时，blank 曲线最终达到了与基于 FSD 的方法的同一点。由于上面讨论的明显拐点及固定加速比，可以很容易得到等式6-2中的阈值 T。此外，特征层的 entropy 和 blank 剪枝可以进一步与标签层的 beam 和 histogram 方法相结合，以得到最佳的解码加速效果。为了使对比更加清晰，图中没有给出融合系统的曲线。

最后，entropy 的效率略高于 blank (相对约 10%)。我们认为原因在于神经网络中剪枝能更好地利用神经网络输出分布中的信息，并且产生更好的精度和效率。而 blank 剪枝则仅利用了输出分布中的最佳分数而未使用整个分布的信息。

#### 6.4.2 GSM 实验

(1) 在各种模型和准则中的应用：LSD 应用于生成式序列模型 (GSM) 时，本文使用了多种不同的神经网络模型结构和模型训练准则进行对比；默认使用上下文相关的音素作为模型建模单元。表6-3给出了在 NIST 2000 CTS 测试集合上的结果。总体而言，LSD 框架也可以取得比较显著的解码加速，但是与表6-1中的结果相比，解码加速性能变差。这是因为 FSD 基线的帧率已经降低到原来的 1/3[108] (类似前文中帧率改变技术可以与提出的 LSD 框架结合)。而且与表6-2相比，加速比也略小，原因是这些模型的推理搜索分布概率不像 CTC 那样尖锐。如何在 GSM 中获取更尖锐的推理搜索分布概率将在后续章节中进行讨论。

具体地，如表 6-3 所示，第一行列出了文献 [108] 中提出的低帧率模型 (LFR) 的结果；第二行是使用 LF-MMI 准则 [83] 训练出来的结果，显示出比 LFR 更快的搜索速度；此外，还可以看到从 FSD 到 LSD，基于 LF-MMI 准则训练的模型可以取得更快的加速。与文献 [191] 中观察到的类似，这都源于序列区分性训练准则得到的模型相对于交叉熵准则训练的模型有更尖锐的输出概率分布。第三行表示为 + sMBR，是在 LF-MMI 模型的基础上，使用基于 LM 的 sMBR 准则微调模型参数得到的结果；第四、五行列出了基于增强的 MMI[145] 及 sMBR 准则变体的无需词图的区分性训练准则取得的结果，分别表示为 LF-bMMI 和 LF-sMBR。可以观察到，本文提出的 LSD 框架在以上模型和准则中可以取得一致的解码加速。另外我们还使用了 BLSTM 模型，也都取得了相似结果。

(2) 候选序列剪枝：如图6-4所示，我们在生成式序列模型下进行了一系列候选序列剪枝相关的实验，其变化趋势类似于 DSM 中的结果，读者可以参考那里的讨论。与图6-3中一个区别在于，beam, histogram, blank, entropy 在图6-4中的最左边位于相同点，这表明在降低帧率的情况下，特征层候选序列剪枝的最大比例较小。然而，在解码性能的 WER 达到最佳时，仍然有接近两倍的解码加速。

表 6-3 LSD 与 FSD 在不同的 GSM 模型上的性能和速度比较

模型	准则	性能		速度			
		FSD→LSD	WER △(%)	FSD→LSD	SRTF △(%)	FSD→LSD	#AT △(%)
TDNN	CE	17.8	+1.0	0.16	-38	3705	-41
	LF-MMI	15.6	+1.0	0.13	-43	3386	-45
	+sMBR	15.4	+1.0	0.12	-41	3295	-43
	LF-bMMI	15.0	+1.0	0.11	-42	3198	-44
	LF-sMBR	15.3	+1.0	0.12	-41	3288	-44
BLSTM	LF-MMI	15.2	+1.0	0.12	-44	3290	-47
	LF-bMMI	14.3	+1.0	0.11	-43	3205	-45

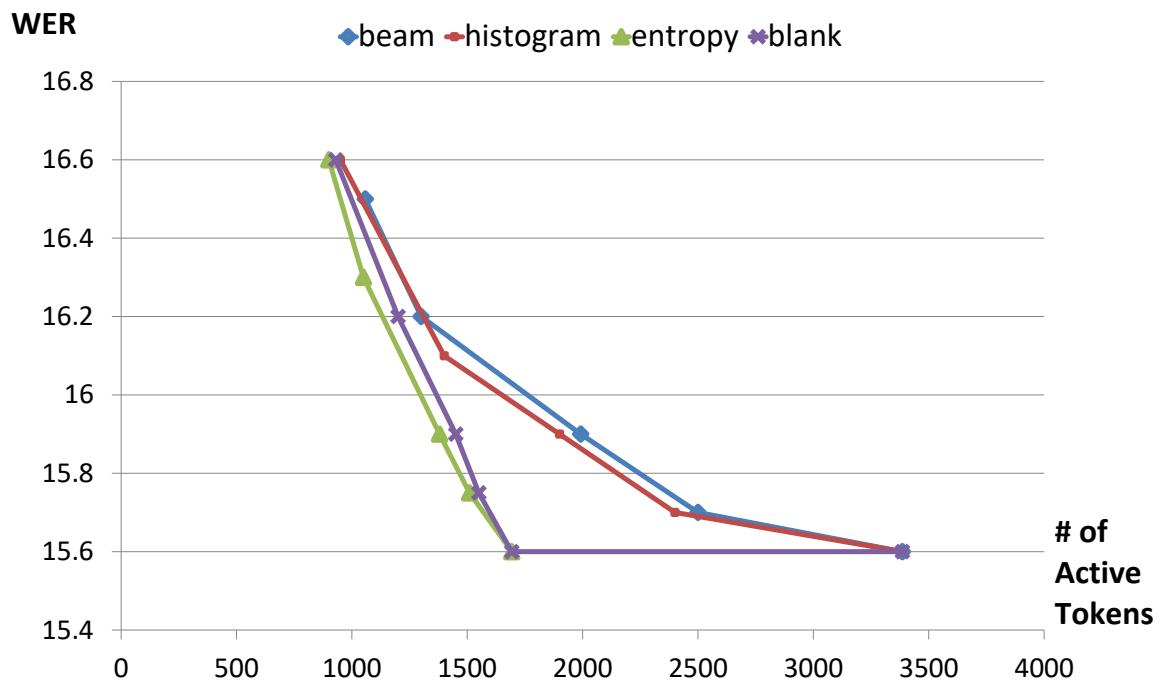


图 6-4 LF-MMI 中，使用不同剪枝技术时 WER 随平均活跃令牌数的变化趋势  
Fig 6-4 The trend of WER v.s. the number of active tokens using different pruning methods in LF-MMI.

(3) 进一步设计：本节将对比前文中讨论的各种转移模型，以及由此获得的效率的进一步提高。该部分所有实验均在 LF-MMI 准则上进行，但实验结论可以扩展到其它神经网络和训练准则上。

表 6-4 生成式序列模型中的 blank 粒度

系统	blank	性能		速度			
		FSD $\rightarrow$ LSD	WER Δ(%)	FSD $\rightarrow$ LSD	SRTF Δ(%)	FSD $\rightarrow$ LSD	#AT Δ(%)
TDNN LF-MMI	CD phone	15.6	+1.0	0.13	-43	3386	-45
	phone	15.7	+0.9	0.09	-47	2785	-50
	global	16.8	+0.8	0.09	-49	2512	-54

表 6-5 中列出了不同 blank 粒度的对比结果，即上下文相关的音素 blank (CD phone blank)，音素 blank (phone blank) 和全局 blank (global blank)。与 CD phone blank 基线相比，phone blank 在取得近似的解码性能的同时，实现了显著的搜索过程加速；这里搜索加速主要源于较少的模型建模单元，即模型状态数从 6K 减少到 3K。此外，从表中可以看出，global blank 会带来明显的性能下降；global blank 需要足够的数据来覆盖不同相邻音素之间的上下文环境（我们认为这也是 CTC 准则在这个语料库中表现更差的原因之一）；CD phone blank 可以缓解 blank 训练数据不足的问题，但会导致搜索速度变慢；因此，绑定中心音素相同的 CD phone blank 在加速搜索过程的同时，也可以更好地建模 blank 模型；因此，phone blank 是解码性能和搜索速度之间的最佳平衡。此外，从表 6-5 中可以看出，在 LSD 框架下，较少的模型单元可以持续带来明显的搜索过程时间缩短：从 43% 到 47%，再到 49%。最后，phone blank 是基于 GSM 的 LSD 框架的最佳选择。

表 6-5 生成式序列模型中的 blank 拓扑结构

测试集	拓扑	性能		速度			
		FSD $\rightarrow$ LSD	WER Δ(%)	FSD $\rightarrow$ LSD	SRTF Δ(%)	FSD $\rightarrow$ LSD	#AT Δ(%)
TDNN LF-MMI	PB	15.6	+1.0	0.13	-43	3386	-45
	BP	15.6	+1.0	0.13	-46	3392	-49
	BPB	15.6	+1.0	0.13	-47	3388	-51

表 6-5 对比了前文提出的不同 HMM 拓扑结构。在 FSD 框架下，所有拓扑结构都有

相似的解码结果和相同的搜索速度。对比前两行可以看出，与基线 PB 拓扑结构相比，在 LSD 框架下，BP 可以获得更大的搜索加速。我们认为这个更优的搜索加速源于标签延迟现象，类似于文献 [137] 中观察到的现象，这使得模型能更可靠地推断标签输出状态并减少混淆。因此，这能带来更尖锐的输出分布。从表中还可以看出，BPB 拓扑结构可以进一步改善搜索速度；一些解码路径的例子也表明这种拓扑结构可以使每个上下文相关的隐马尔科夫模型输出更多的 blank 状态。最后，与前面表中 CTC 的结果相比，GSM 中 LSD 框架能减少 49% 的搜索时间。

## 6.5 本章小结

在本章中，我们针对语音识别和端到端建模的推理搜索阶段，提出了标签同步解码算法，其通过一系列方法使得搜索解码过程从逐帧同步变为标签同步，这包括使用高效的 blank 结构和后处理方法。该文提出的一系列通用方法在隐马尔科夫模型和连接时序分类模型上得到了验证。同时该章节还介绍了将标签同步算法应用于序列到序列的端到端模型的方案。在实验部分，该章节系统取得了大幅度语音识别解码速度改善。



## 第七章 标签同步解码的扩展应用

在上一章中，我们导出了标签同步算法，使得搜索解码复杂度得到了显著下降。这一算法同时还能产生高质量的音素词图，称为 CTC 音素词图，我们将在第 7.2.4.1 章节通过实验验证其质量。基于 CTC 音素词图，本章节我们将进一步探讨标签同步解码算法的一些扩展应用。这包括关键词检测，多识别任务统一置信度框架，以及端到端语音识别。

在关键词检测中，我们基于前面章节介绍的 LSD 算法提出了一种高效的后处理算法，以解决音素混淆建模的问题。在基于 CTC 的关键词检测中，虽然 CTC 是一个序列级准则，但是并没有对非关键词部分进行建模。因此该准则改善了关键词之间的鉴别性，但是没有改善关键词和非关键词之间的鉴别性。本文将非关键词建模单元直接引入到建模当中。同时我们基于前述 LSD 算法得到的音素词图，提出了一套基于编辑距离的后处理算法以引入音素混淆性，使系统更加鲁棒。

由于不同 ASR 应用之间不同的搜索空间大小和效率要求，当前业界最优的置信度及其相应的解码算法在不同应用上具有不同架构，这些不同应用包括：关键词检测，基于上下文的语音识别和大词汇连续语音识别。针对基于词图后验概率的置信度，计算量主要集中在词图部分的边缘概率计算过程。如第 3.4.2 章节中对于解码搜索的研究机遇的讨论，本章节中，我们将结合前面章节对解码的大幅加速和搜索空间的优化等工作，提出一系列针对不同应用的通用置信度，并尝试将不同应用中的语音识别推理搜索过程统一到同一框架中。本章节所提出的置信度在 Switchboard 数据集上得到了显著改善，同时这类统一框架的通用置信度在不同应用中取得了一致的非常有竞争力的性能，使得各种语音识别应用可以工作在同一框架中。

另一方面，该章节研究将标签同步算法应用于序列到序列的端到端模型的方案。在实验部分，该章节系统一方面取得大幅度语音识别解码速度改善，另一方面在端到端建模上取得了更快和更好的模型收敛和模型准确度。

### 7.1 基于标签同步解码的关键词检测

在基于 CTC 的关键词检测中，我们基于前面章节介绍的 LSD 算法提出了一种高效的后处理算法，以解决音素混淆建模的问题，使系统更加鲁棒。除此之外，虽然 CTC 是一个序列级准则，但是并没有对非关键词部分进行建模。因此该准则改善了关键词之间

的鉴别性，但是没有改善关键词和非关键词之间的鉴别性。本文将非关键词建模单元直接引入到建模当中。

### 7.1.1 模型训练

如前面章节2.3.3所述，序列鉴别性训练的关键之处是竞争序列的建模。在本文中，我们研究了两种方向：词语建模和音素建模的 CTC 关键词检测模型。

为了更好地建模词级 CTC 模型的竞争序列，这里针对非关键词引入了新的建模单元。类似于传统的声学 KWS 方法，代表非关键词的 `filler` 被引入。在训练阶段，`filler` 则取代所有标注中的非关键词。

另一个方向是针对音素级模型，我们进行修改的动机是：i) 任何音素序列，也包括非关键部分，都可以由音素模型来进行建模。ii) 使得声学模型容易做到与关键词无关，应用于非固定关键词的关键词检测任务。除此之外，我们在词语边界处引入了 `wb` 单元 [192]。在音素 CTC 中 `wb` 和 `blank` 被分别作为词和音素的边界进行建模。区分这样的边界区段不仅可以改善泛化能力，而且也对后面将介绍的后处理有帮助。

这里针对引入的建模单元，公式修改如下：

$$P(\mathbf{L}_u | \mathbf{O}_u) = P(\mathbf{L}'_u | \mathbf{O}_u)_{\mathbf{L}'_u = \mathcal{D}(\mathbf{L}_u)} \quad (7-1)$$

$\mathcal{D}$  是标签的映射函数。 $\mathcal{D}$  针对词级或者音素级 CTC 具有不同的定义。

$$\begin{aligned} \mathcal{D}_{\text{word}} : \mathbb{L} &\mapsto \mathbb{L} \cup \{\text{filler}\} \\ \mathcal{D}_{\text{sub-word}} : \mathbb{L} &\mapsto \mathbb{L} \cup \{\text{wb}\} \end{aligned} \quad (7-2)$$

经过修改之后，在词级或者音素级建模中，搜索空间都将包含关键词，非关键词，音素边界和词边界。

### 7.1.2 后处理

我们进一步基于最小编辑距离（MED）提出了一个针对 CTC 推理搜索分布的后处理方法，以便引入音素的混淆性先验知识，增强模型性能。这项工作受到 [193] 启发，这里基于 CTC 词图和 MED 算法 [189] 来针对音素引入混淆建模。

图 7-1 给出了在音素 CTC 中使用 MED 算法的框架。在一句音频中，关键词出现于 CTC 词图的概率是用三种编辑距离的概率相乘得到的，它们包括插入，删除，替换。每一种操作的概率由 MED 算法决定。现实中，每个音素在 CTC 的建模中性能可能各不相同，因此需要进一步采用音素的先验来设计每个音素的阈值，这部分可以通过预先统计得到 [192]。

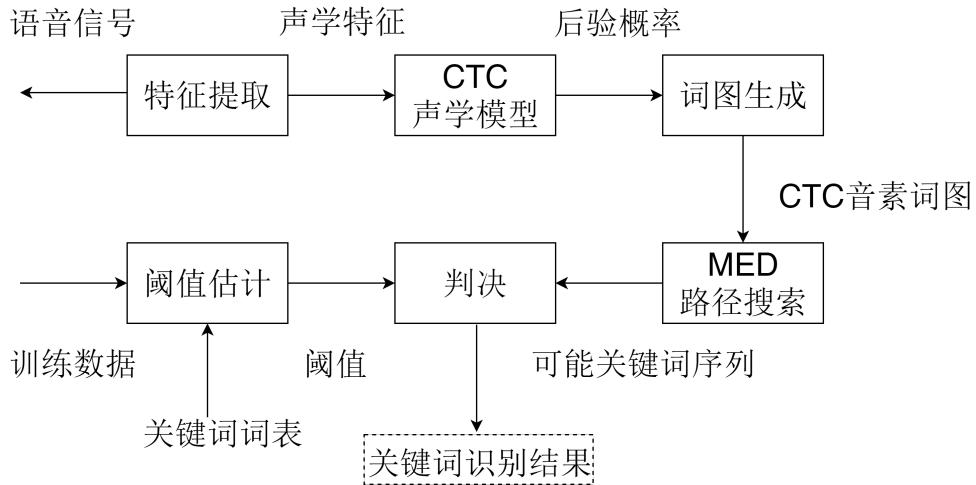


图 7-1 MED 方法框架。这里使用音素 CTC 作为例子

Fig 7-1 The framework of MED based method. Phone based CTC is taken as an example.

### 7.1.3 实验结果

本章节在非固定关键词的关键词检测和固定关键词的唤醒词识别任务上都进行了验证，采用与第4.1章节相同的数据集，实验配置，和评价指标。我们也绘出了 receiver operating characteristic (ROC) 曲线，以总结实验结果。CTC 模型的配置详见 [189]。我们使用单向每层 384 节点的两层 LSTM 进行实验，其带有 128 个节点的映射层。CTC MED 后处理方法，表示为 *MED*，并使用 [189] 中提出的固定阈值方法。我们在第7.2.4.1章节针对 CTC 音素词图质量进行更细致的分析。

#### 7.1.3.1 非固定关键词的关键词检测

在音素 CTC 中，引入的 *wb* 建模单元在表 7-1 中进行了比较。对于少于 6 个音素的关键词被认为是短关键词，其余的为长关键词。因此可以将关键词集分成两部分分别检查他们的模型性能。在短和长关键词中 *wb* 都可以带来性能提升。长关键词的性能一致地优于短关键词，原因是短关键词的音素序列更可能是其它关键词的子集，造成了混淆和误唤醒率；*wb* 的引入缓解了这个问题。

不同的针对 CTC 的后处理算法在本节中进行了检查。值得注意的是，本部分是非固定关键词的关键词检测任务，因此音素被用来构建关键词序列。

如表 7-2 所示，在 CTC 中，后验概率平滑方法会得到比 kw-filler 显著差的结果。但是由于它的速度优势，该方法可以作为实际的前处理，滤除一些容易判断的样本，以减少语料库检索的计算量。在 CTC 中我们比较了所提出的 MED 算法。它显示了明显比

表 7-1 音素 CTC 在是否包含 wb 时的模型性能

Keyword Length	wb	EER
short	×	9.0
	✓	4.5
long	×	3.1
	✓	1.8

表 7-2 CTC 关键词检测系统的后处理

Model (Crit.)	Post	EER	RTF
CTC	smooth	11.4	0.026
	<b>kw-filler</b>	3.2	0.038
	MED	3.6	0.031

kw-filler 更差一些的性能，但效率较高。该方法将会在后续章节中进一步检验。kw-filler 系统在未来的工作中可以使用前述的标签同步解码算法进行进一步速度优化。

最后，我们将性能和速度的比较总结在表 7-3 中。我们同时列出了第 4.1 章节中最好的系统 LF-bMMI 作为参照。我们绘制出了 ROC 曲线如图 7-2，其中越低的曲线结果越好。其中所有的系统都使用 kw-filler 作为后处理方法。

表 7-3 非固定关键词的关键词检测中的性能和速度比较

Model	Context	# Param.	Criterion	EER	RTF
TDNN HMM	CD	0.6M	CE	4.0	0.051
TDNN HMM	CI	0.5M	LF-bMMI	<b>2.9</b>	<b>0.028</b>
LSTM CTC	CI	0.8M	CTC	<b>3.2</b>	<b>0.038</b>

这里我们使用交叉熵训练得到的系统作为第一行的基线。它是一个基于 NN-HMM 的传统系统，包含了聚类的 tri-phone 状态建模。这里的 HMM 拓扑结构见图 2-8(a)。

使用 LSTM 的 CTC 系统在最后一行中，显示了比传统 CE 模型更好的性能 ( $\alpha = 0.05, p = 0.01$ )。从图 7-2 中看，CTC 系统相比第 4.1 章节中最好的系统 LF-bMMI 存在相对较多误唤醒。

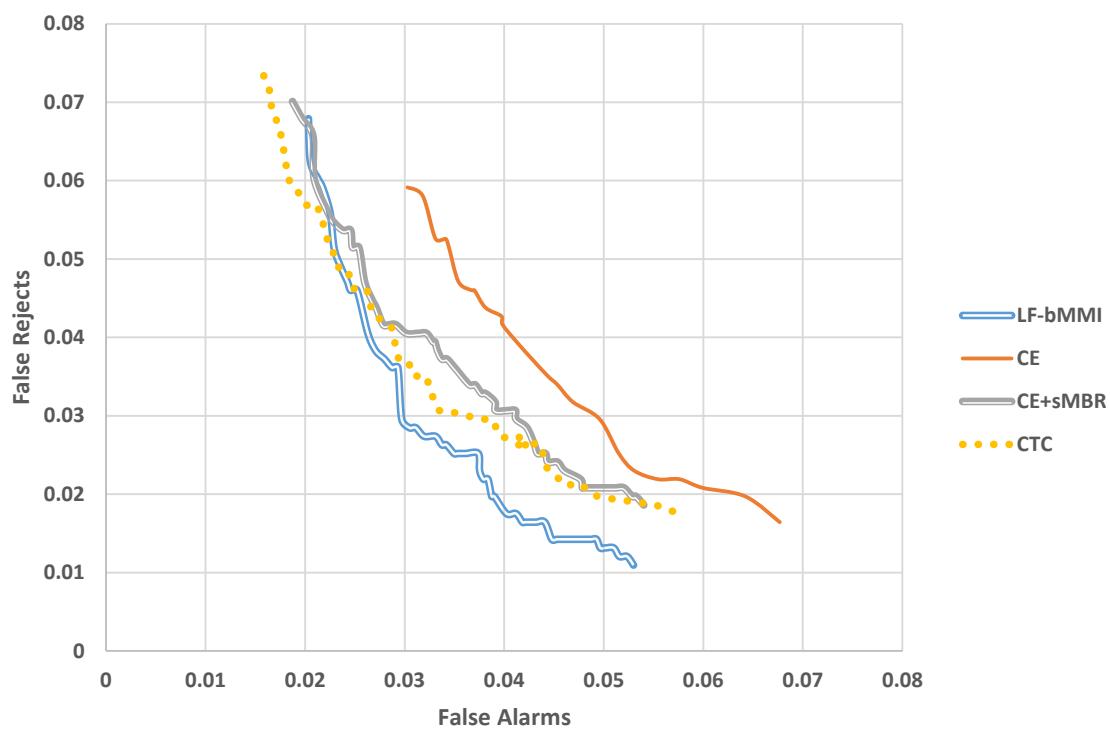


图 7-2 非固定关键词的关键词检测的 ROC 曲线比较  
Fig 7-2 The ROC curve comparison of unrestricted KWS.

我们在一些初始的尝试中使用基于 HMM 的序列鉴别性训练作用在 CTC 初始化的变种模型上 [140, 194]。但结果并不成功，没有带来改善：在词级别 CTC 中 [195], LVCSR 解码词图不包含有效的竞争路径。而在音素级别模型上，如前所述，词图的质量限制了传统框架下鉴别性训练所能得到的改善。

### 7.1.3.2 固定关键词的唤醒词识别

在该任务中，我们测试了词级系统和音素级系统。CTC 的关键词检测系统与传统交叉熵系统进行了比较。词级系统的输出为关键词序列中的每个词。我们同时列出了第4.1章节中最好的系统 NU-LF-bMMI 作为参照。

表 7-4 显示了相应的结果。一个交叉熵训练的词级别的 TDNN 系统被作为基线系统 [95]。该系统采用了后验平滑作为后处理方式。

表 7-4 固定关键词的 CTC 的 KWS 系统的性能和速度比较

Model	Unit	Criterion	Post	EER	FAF	RTF
[95]	Word	CE	smooth	6.2	0.64	0.014
HMM	Syllable	NU-LF-bMMI		<b>5.2</b>	<b>0.51</b>	<b>0.029</b>
CTC	Word +filler	CTC	smooth MED	9.1 <b>7.0</b>	1.13 <b>0.90</b>	0.024 <b>0.029</b>

在第二行中的系统类似于 [82]。虽然它在文献 [82] 中取得了较好的性能，但在本部分测试中仍然差于词级别的 CE 系统。我们相信这是因为我们使用的训练和测试集相比该文献具有更大的挑战。我们针对这一系统如前文所述，添加了 `filler` 建模单元，以改善关键词与非关键词之间的鉴别性；同时我们增加了基于 MED 的后处理算法，以便引入音素混淆性的先验知识。我们所提出的系统在第三行中取得了显著更好的结果。但是这个系统仍然没有超过传统的 CE 系统。我们认为进一步的改善应该包括两方面：对 CTC 模型的噪声鲁棒性的研究；使用更好的深度学习模型来改善模型性能。

## 7.2 多识别任务统一置信度框架

要为所有的 ASR 应用设计一套统一的推理搜索框架，并取得最好的性能和速度，是一项非常有挑战的研究。这里面的关键点在于：i) 如何对最佳识别结果进行规范化，以得到对该最佳结果的置信度估计，也就是置信度中的归一化项的建模 ii) 如何保证在

低功耗设备中的计算量控制在一个较小范围内，比如关键词检测技术就经常被用于个人助理情况，需要对功耗进行优化。*Keyword-filler* 方法 [130] 和 *utterance verification* [196] 方法都可以视为这方面的尝试。比如，在关键词检测中，一些研究提出使用上下文无关的 (CI) 语言学单元，称为 *filler* 来对所有非关键词部分进行建模，而这种建模往往是不完美的。在基于上下文的语音识别中，这种框架更加受制于较弱的上下文建模能力，而这将影响 *filler* 的识别效果。在 LVCSR 中，理论上更好的办法是基于识别结果的后验概率方法的置信度 [131]，也就是前文讨论的第二种置信度。在这种置信度里，ASR 被建模为 *maximum a posterior (MAP)* 过程：给定句子，求取当前识别结果的后验概率，并将其作为置信度。而这里的观察概率（归一化项）使用针对搜索空间的所有可能候选概率求和来得到。由于 ASR 搜索空间很大，词图建模很耗时。在 LVCSR 中，这种框架往往取得最好的效果，但在其它任务中结论不尽相同。

最近在鉴别性训练领域，有些研究采用一个特别设计的音素语言模型作为搜索空间来代替词图，该方法显示出了较好的性能 [84][83]。受此启发，我们尝试将这样的音素语言模型应用到置信度的归一化项的建模当中，由此我们提出了辅助归一化搜索空间的概念。我们尝试使用这样的搜索空间来建模所有 ASR 应用领域的置信度。

针对这样做在低功耗设备上带来的挑战，我们采用前面章节讨论的基于 CTC 的标签同步解码 [197] 来进行处理，由此带来了很大的效率改善。本章节我们提出了一个统一并且高效的置信度框架，并且将其应用于目前主流的上述三种 ASR 应用。

## 7.2.1 置信度与搜索空间

关于搜索空间和解码框架的不同，依据第3.2章节的总结，通常可以根据不同 ASR 应用将其分为三类，而三种应用上目前最好的置信度算法方式并不统一。

### 7.2.1.1 关键词检测

关键词检测任务目标是准确和快速地检测语音中是否包含所关心的词或短语。所以，关键词检测的搜索空间是所有的关键词序列<sup>1</sup>。误接收表示错误地将某些语音段识别为关键词，而这并不是希望得到的。一系列研究 [130, 95] 尝试解决这样的问题，包括使用一定阈值进行后处理估计，或将 Filler 加入声学建模中。

<sup>1</sup> 基于大词汇连续语音识别的关键词检测目前并不包含在讨论中，因为这类方法主要研究在于如何提高声学模型性能以及关键词索引技术。同时计算量过大也不适合端侧设备。

### 7.2.1.2 基于上下文的语音识别

针对语音助手等应用，目前对于基于上下文的语音识别的需求越来越强。在这些场景中，语法 [53] 或基于类的语言模型 [198] 都是比较主流方法 [199]。这种任务的错误识别包括：i) 错误将领域外的句子识别为领域内的结果 ii) 正确识别出了领域，但是上下文短语的部分识别错误。为了给出合理的结果，置信度用于对识别结果进行判别。对于领域内识别，语音模式和上下文信息识别可以被看做是一个完整的搜索空间，这时使用 LVCSR 的后验概率的置信度是合理的。但是对于领域外的句子，上面提到的置信度并没有对其搜索空间进行建模。因此这样的置信度目前还没有比较合理的方案进行解决。

### 7.2.1.3 大词汇连续语音识别

在大词汇连续语音识别 (LVCSR) [53] 中，搜索空间由  $N$  元语言模型进行建模。为了支持语言学后处理 [200]，CM 被用来提供对语音识别结果的可靠度分析。基于识别结果后验概率的 CM [131] 是 LVCSR 中最通用的置信度方法。在这一框架中，ASR 使用 *maximum a posterior (MAP)* 决策过程进行建模。给定整个特征序列的 ASR 输出的后验概率被作为句子的置信度。对于 MAP 的归一化项，即观察概率建模，使用的是所有识别结果组成的搜索空间的后验概率求和。由于语音识别的搜索空间很大，通常这一过程由解码得到的词图来进行限制。

## 7.2.2 基于标签同步解码的置信度框架

CTC[43] 近期被提出来并作为目前主流的新模型 [201][202][137][140]。同时，在相比传统上下文相关的聚类混合深度学习 HMM 模型 [140][186][185][86] 的情况下，上下文无关的单音素 CTC 也显示出非常有竞争力的性能。

我们在第3.3.2.8章节总结目前主流的置信度技术。然而，由于引入了 `blank`，置信度在 CTC 框架下的计算亟待研究，特别是最主流的针对传统基于音素似然度归一化或者基于词图后验概率的混淆网络等方法。如本文的实验部分所示，如果只是将 `blank` 标签当做一个特殊的音素并使用传统的 CM 方法，将会引入较大的性能下降。在前文中我们提出了标签同步解码 (LSD) 的推理搜索框架。它的主要作用是针对 CTC 模型进行高效的解码搜索。该算法提出了一些方式来自动地将 `blank` 帧进行忽略，由此不仅得到了搜索上的加速，还得到了一种非常紧致高效的 CTC 音素词图。我们将会在本章节的实验部分验证了 CTC 音素词图的质量。在这项工作中，两种置信度生成算法在标签同步解码算法的基础上被提出。更细致的研究显示这种基于 CTC 的音素词图是得到更好性能的关键所在。在英文 Switchboard 上的大词汇连续语音识别任务显示这里提出的

LSD CTC 词图置信度算法可以显著改善原先传统的基于逐帧解码算法的 CTC 置信度或者 HMM 模型的置信度。

### 7.2.2.1 标签同步的音素声学置信度

在前文中，我们已讨论了标签同步解码算法，它的主要作用是针对 CTC 模型进行高效的解码搜索。该算法提出了一些方式来自动地将 `blank` 帧进行忽略，由此不仅得到了搜索上的加速，还得到了一种非常紧致高效的 CTC 音素词图。下面我们将集中讨论基于这些 CTC 音素词图如何得到较好的置信度。

这里 CTC 音素后验概率可以依据词语进行区分，属于同一个词的后验概率可以用于一起对一个特定词在词图中出现的后验概率进行建模。在 LSD 框架中，词级别的 CM  $\mathcal{C}(w)$  可以被定义为对数域的后验概率，针对其相应的最优路径。

$$\mathcal{C}(w) = \log \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l}_w)} P(\pi | \mathbf{x}) \triangleq \max_{\pi': \pi' \in L, \mathcal{B}(\pi'_{\mathbf{j}_w}) = \mathbf{l}_w} \sum_{j:j \in \mathbf{j}_w} \log(y_{\pi'_j}^{t_j}) \quad (7-3)$$

$\mathbf{l}_w$  表示词语  $w$  对应的音素序列。 $j$  是音素序列的索引 (i.e. [110] 中定义的非 `blank` CTC 标签序列)。由于 LSD CTC 中的词边间比较确定， $\mathbf{j}_w$  可以被定义为  $w$  词所最佳对应的音素序列索引。我们提出的基于标签同步解码的声学分数可以作为单独的置信度，也可以与其它用于建模的特征结合，作为置信度模型的输入 [129]。具体来说， $\mathcal{B}$  函数的建模在 CTC 中并不完美，使得这里会出现同一个音素有多帧的音素后验概率输出。所以，我们需要在它上面进一步进行归一化。其中一种方法是进行算数平均，被称为 *peak-mean*。但是，由于有多个概率输出尖峰，更好的一种方法是忽略不完美的部分并保留最大的结果，所以从中挑选最大的后验概率作为其置信度，被称为 *peak-max*。除此之外，不同词语具有不同的长度，因此我们需要针对长度作一次平均。为了使性能更好，我们这里对音素序列的长度也进行了归一化称为 *phone-mean*。

另一类比较现实的问题是，`blank` 的区段和音素区段有所重合。因此这里需要对非 `blank` 的概率同样进行一些建模。音素置信度 (称为 *phone-conf*) 可以被定义为某帧上输出音素标签的概率。以上这些针对不同模块的设计总结如下 (7-4)，

$$\mathcal{C}(w) \triangleq \max_{\pi': \pi' \in L, \mathcal{B}(\pi'_{\mathbf{j}_w}) = \mathbf{l}_w} \frac{1}{|\mathbf{j}_w|} \sum_{j:j \in \mathbf{j}_w} \max_{t:t \in \mathbf{t}_j} \log(y_{\pi'_j}^t (1 - y_{\text{blank}}^t)^\alpha) \quad (7-4)$$

这里的 *peak-max* 作为一个例子出现在公式中。 $\mathbf{t}_j$  是音素  $j$  在最优路径下所相应的帧。 $\alpha$  是置信度融合的权重概率。

### 7.2.2.2 基于混淆网络和 CTC 标签同步解码词图的置信度

类似于 [132] 中的做法，这里生成第3.3.2.8章节介绍的混淆网络（CN）主要有两步：a) 从音素级别的标签同步 CTC 音素词图（图 7-3）中产生词级词图；b) 将词级词图转换为混淆网络，利用其包含的时间边界信息。在 [200] 中提出的 pivot clustering 算法使得混淆网络生成复杂度为  $O(n)$ ， $n$  为词图的边数。在这项工作中，我们将最优路径视为 pivot，由于 CTC 词图非常紧致，因此 CN 产生的过程非常高效。在 CN 的构建过程中，我们需要计算词后验概率，而其自然地成为了词级置信度。

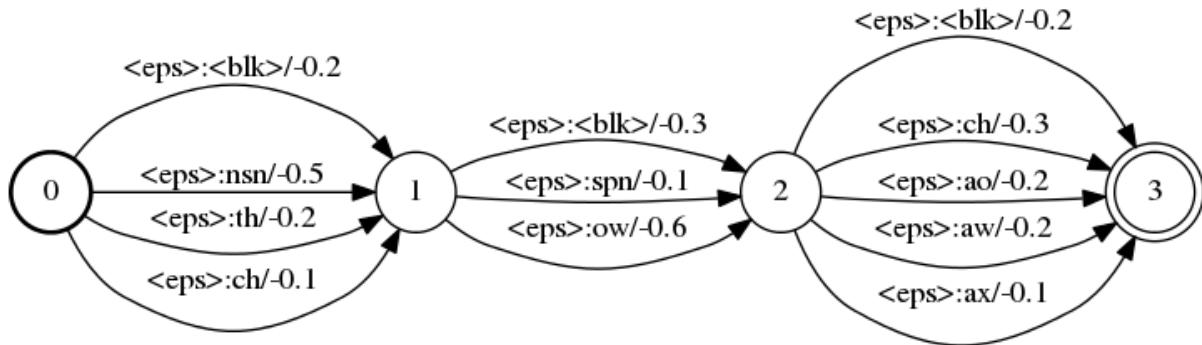


图 7-3 LSD CTC Lattice 的例子  
Fig 7-3 The example of LSD CTC Lattice.

图 7-4 中是一个真实的例子（句子为”OH YEAH”）表示了 LSD CTC 产生的 CN 的高效性，以及与 HMM-DNN 产生的词图之间的比较。由 HMM 和 CTC 得到的推理搜索结果在图 7-4(a) 中进行显示，其得到的音素词图，词级词图，分别在 HMM 和 CTC 系统中展示于图 7-4(b~e) 中。我们可以观察到，CTC 基于 LSD 得到的音素词图和词级词图比相应的 HMM 词图更加紧致，这是源于  $\beta$  函数的建模结果。另一方面，HMM 的词图需要额外的启发式方法来进行多对一的映射，以便去除词图冗余性。这里的做法是进行词图裁剪 [203]，而它不如  $\beta$  函数的效果那么强。

另一方面，当 CTC 模型被使用于传统的逐帧同步解码 (FSD) 框架时，音素和词级词图可以由同样方法进行产生 [122]，类似于处理 HMM-DNN 词图的方式对 blank 标签进行等同于音素的处理。但是由于逐帧搜索和词图此案件所带来的搜索误差，以及词图边界的混淆性，所得到的 CN 将会具有更差的质量。我们将会在后续的实验章节中进行详细比较。

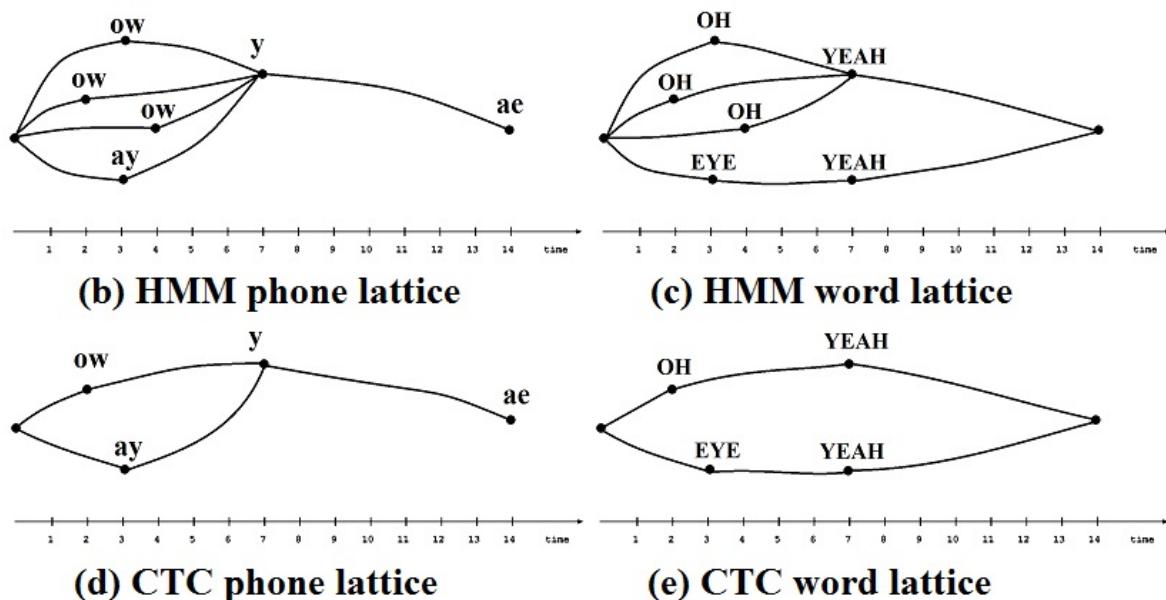
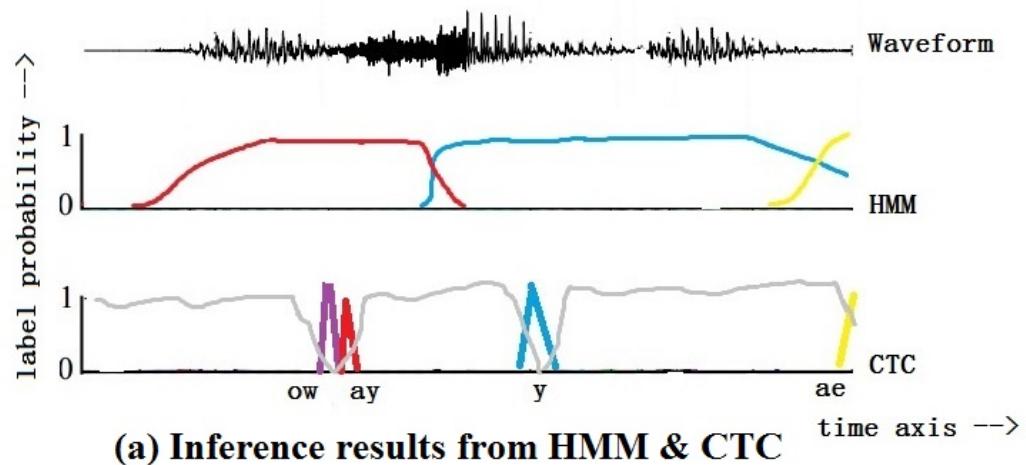


图 7-4 FSD HMM 和 LSD CTC 所产生的词图比较

Fig 7-4 The comparison of FSD HMM and LSD CTC generated lattices.

### 7.2.3 基于附属归一化搜索空间的置信度方法

这篇文章里，我们尝试提出一个能够适用于各种 ASR 应用的统一框架。我们的方案基于所提出的附属归一化搜索空间和基于 CTC 的标签同步解码方法。

#### 7.2.3.1 统一的置信度框架

对于 ASR 输出的后验概率可以在 MAP 框架中作为一个句子级别的置信度。

$$CM = P(\mathbf{w}|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{w}) \cdot P(\mathbf{w})}{P(\mathbf{x})} \quad (7-5)$$

公式中  $P(\mathbf{w})$  表示语言模型概率  $P(\mathbf{x}|\mathbf{w})$  是声学模型的部分。 $P(\mathbf{x})$  是观察概率  $\mathbf{x}$ ，由下式建模，

$$P(\mathbf{x}) = \sum_H P(\mathbf{x}, H) = \sum_H P(H) \cdot P(\mathbf{x}|H) \quad (7-6)$$

这里  $H$  表示所有可能的识别结果的路径。 $H$  根据不同的 ASR 应用而有所不同。因此  $H$  的建模通常都是性能的瓶颈。

构建这样的通用框架的主要挑战在于：i) 如何对任务相关的集合  $H$  使用一个统一框架进行建模 ii) 如何在计算效率较高的情况下对无限的  $H$  进行建模。

#### 7.2.3.2 附属归一化搜索空间

为了解决这个问题，我们提出使用附属归一化搜索空间，将其作为归一化项的搜索空间进行建模。该方法的框架图 7-5 所示。

在基于词图的方法中， $P(\mathbf{x})$  是从解码网络的一个子区域，词图中进行计算的。在基于 **filler** 的方法中， $P(\mathbf{x})$  是由自环的音素组成的。在我们所提出的方法中  $P(\mathbf{x})$  由下述的搜索空间得到

$$P(\mathbf{x}) \approx \max_H P(H) \cdot P(\mathbf{x}|H) \quad (7-7)$$

这里，我们尝试了三种不同的附属归一化搜索空间。

- 自环音素搜索空间 (**AX1**)。类似于传统的关键词-filler 置信度 [130]，一个由所有的音素组成的网络可以用来建模归一化项的边缘概率值。
- 无词典搜索空间 (**AX2**)。受到近期无词图鉴别性训练方式的启发 [83]，附属归一化搜索空间可以由一个音素级语言模型而得到，用于近似搜索空间。

- 基于词典的搜索空间 (AX3). 在一些语言当中, 比如汉语, 音频与字符的关系是多对一映射的<sup>1</sup>. 所以, 在给定相对固定数量的字符后, 我们可以预期的发音数量是有限的。所有可能的发音可以作为一个附属归一化搜索空间。

除此之外, 我们所提出的方法可以作为词级别的置信度。这种情况下公式 (7-5) 和 (7-7) 可以被转化为公式 (7-8) 和 (7-9),

$$CM = P(w|\mathbf{x}) = \frac{P(\mathbf{x}|w) \cdot P(w)}{P(\mathbf{x}^w)} \quad (7-8)$$

$$P(\mathbf{x}^w) \approx \max_{H^w} P(H^w) \cdot P(\mathbf{x}^w|H^w) \quad (7-9)$$

附属归一化搜索空间的解码结果是对观察概率的一个很好估计。由于声学模型的单元通常是音素, 这样的置信度框架在大多数语音识别应用中都是合适的。与传统词图方法相比较, 这里提出的方法相比其它的 ASR 搜索空间更加稳定鲁棒, 我们将在实验中证实这一点。除此之外, 这套方法不需要再额外引入一系列非关键词的模型建模单元, 比如基于 **filler** 或者语句验证这类估计框架。所以, 置信度归一化建模可以是相对原始搜索空间和声学模型而独立的部分。

### 7.2.3.3 基于 CTC 的标签同步解码

上面讨论的方法具有一定的计算昂贵性, 特别是针对低功耗应用比如关键词检测。基于 CTC 的标签同步解码 [197] 可以考虑被采用, 以加速这样的应用。由于这类方法可以跳过 blank 的区段, 使得结果中具有更少的混淆, 这体现在公式 (7-6) 中。这类方法也被前面的章节证明, 解码只占全部计算量的一小部分 [197][110]。在实验中我们将验证相关结论。

### 7.2.4 实验结果

我们的置信度实验主要在 300 小时 Switchboard 上进行。而针对多种任务的融合框架则在后面介绍。我们训练得到了上下文相关的状态级别 HMM (CD-state-HMM) 和上下文无关的音素级别 CTC (CI-phone-CTC)。训练的配置与解码配置和上文相似 [110]。所有的模型都使用大约 2-2.5M 参数, 以便进行公平比较。在测试中, 我们使用 NIST Hub5e00 测试集的 switchboard 子集 (1831 句子)。对 CTC 模型, 我们测试 FSD 和 LSD 两种方法。表 7-5 给出了不同模型和解码框架下的基础性能, 其与前文保持一致。

---

<sup>1</sup>而像英语, 这个映射是多对多

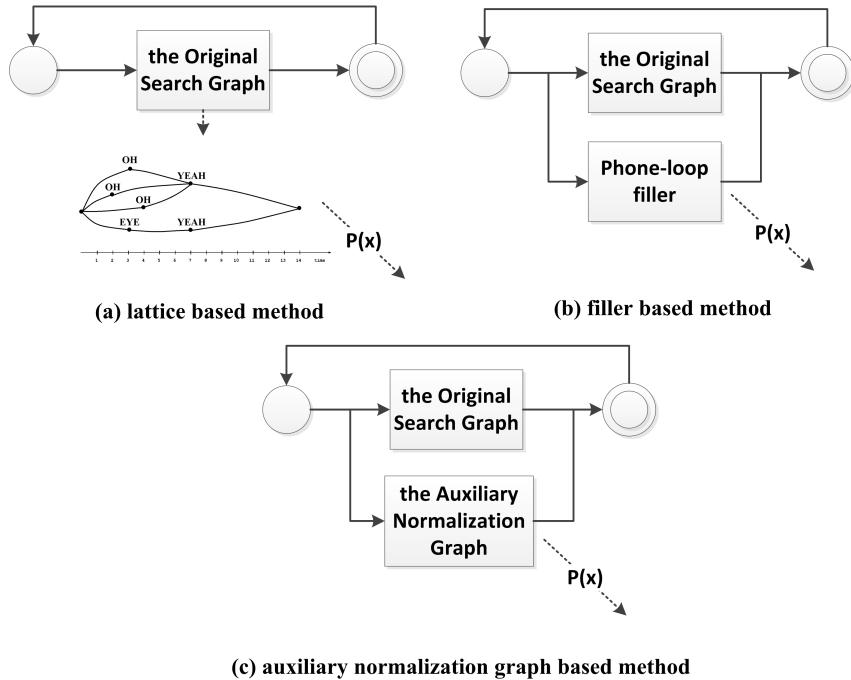


图 7-5 不同语音识别任务的架构比较。这里原始的搜索空间包括，关键词检测，基于上下文的语音识别，和大词汇连续语音识别。

Fig 7-5 The architecture comparison of different ASR tasks.

表 7-5 WER 比较

Model Unit	AM	Decoding	WER
CD-state	DNN-HMM	FSD	16.7
CI-phone	LSTM-CTC	FSD LSD	18.7 18.8

在测试针对各种 ASR 应用的通用融合框架时，我们在三种应用上分别进行试验：关键词检测，基于上下文的语音识别和大词汇连续语音识别。一个 5000 小时中文数据训练的 CTC 模型被用于测试，其配置与 [197] 中相同。

在关键词检测中，词级别的置信度质量使用误唤醒和未唤醒错误来衡量。在基于上下文的语音识别中，使用句子级别的置信度来区分上文中介绍的领域内错误和领域外错误。等错误率 (EER) 用来度量上面两张测试下的错误率，该指标反映了误唤醒和未唤醒错误的均值。越低的 EER 表示越好的模型性能。归一化交叉熵 (NCE) [177] 用来作为词级别置信度质量的评估，在大词汇连续语音识别中。越大的 NCE 表示模型性能越好。为了保证 ASR 准确率没有受到该框架的影响，我们还度量了前两个测试中的句子的召回率，以及在大词汇连续语音识别中的字错误率 (CER)。为了度量框架的效率，我们给

出了解码搜索时间占总体计算时间的比例，表示为 *portion of time except acoustic model* (PEA)。因为实验总是在相同的声学模型上进行，越低的 PEA 表示计算过程越少消耗在解码框架上，这包括搜索空间解码，词图生成，后处理等。因此越低的 PEA 效果越好。

在基线置信度方法中包含针对用于建模的特征的置信度，表示为 AC 和 CN。我们提出的方法分别在实验中进行了比较，他们是 AX1, AX2 和 AX3。附属归一化搜索空间 AX2 是由一个三元音素级语言模型而得到的，它包含了 145K 组词历史。AX3 是由词级别搜索网络  $L$  与一个由所有发音自环组成的网络  $G$  合成得到的,  $L \circ G$ . *filler* 方法没有被包括在评估中，因为它理论上和 AX1 相同。

本章研究了 LSD 生成的混淆网络的质量比 FSD 好的原因。我们首先比较了音素词图质量，而后讨论了词级词图和混淆网络的生成方案。

#### 7.2.4.1 音素级词图质量分析

如前面章节所讨论，CTC 模型尝试对多对一函数  $\mathcal{B}$  进行建模，使得最终结果能得到非常突出的音素推理搜索结果。LSD 音素词图是从丢弃了一定数量的 blank 帧的推理搜索分布中生成得到的，之后可以将剩余帧的一定阈值以内的音素后验概率收集起来组成时间上不连续的词语串。这样的方式避免了一些搜索错误和混淆的音素边界问题，使得最终得到的词图更紧致。

全局最优音素错误率 (OPER) 被用来对音素词图的质量进行评估。它使用词图中最优的一条路径计算得到的错误率来作为全局错误率 [176]。词图密度 (Arcs/Sec) 则被用来衡量词图的紧致程度 [53]。

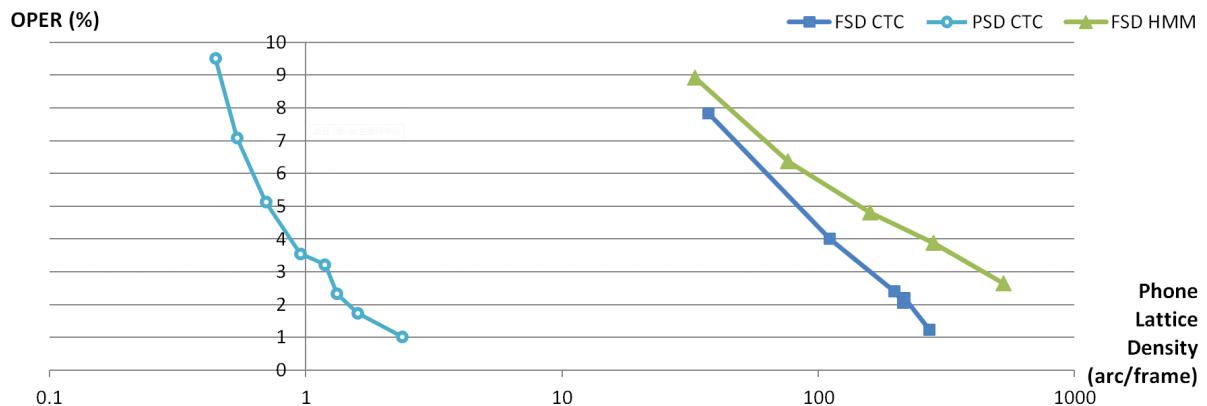


图 7-6 OPER v.s. 词图密度在 FSD 和 LSD 中的比较

Fig 7-6 OPER v.s. lattice density in FSD and LSD

图 7-6 表示了 OPER 和词图密度在不同解码配置结果中的对比，这包括由 FSD 和 LSD 生成的音素词图。在相似 OPER 情况下，对同一个 CTC 模型，LSD 词图的大小比 FSD 词图小超过 10 倍。而 LSD 词图比 HMM-DNN 缩小的数量更大。产生这样现象的原因包括两方面：CTC 更突出的后验概率特性而得到的紧致词图；LSD 避免了词图生成过程的大量 blank 帧，而 FSD 则需要进行一些近似 [175] 或进行词图裁剪 [122]，这将导致更大的搜索误差。总结起来，在相同大小下，LSD CTC 包含更多的音素声学信息。由于接下来的重点是比较 FSD 和 LSD，因此我们仅罗列该部分结果。

为了构建混淆网络，需要先生成词级词图。图 7-7 表示了词级词图质量在 FSD 和 LSD 中的对比。我们使用 OWER 进行比较。在图中纵轴，我们画出了  $1 - \frac{OWER}{WER}$  的值作为 OWER 相对下降值，以体现词图对最终质量改善的上限。从图中可以看出，在相同词图密度情况下，LSD 系统总是得到更好的 OWER。

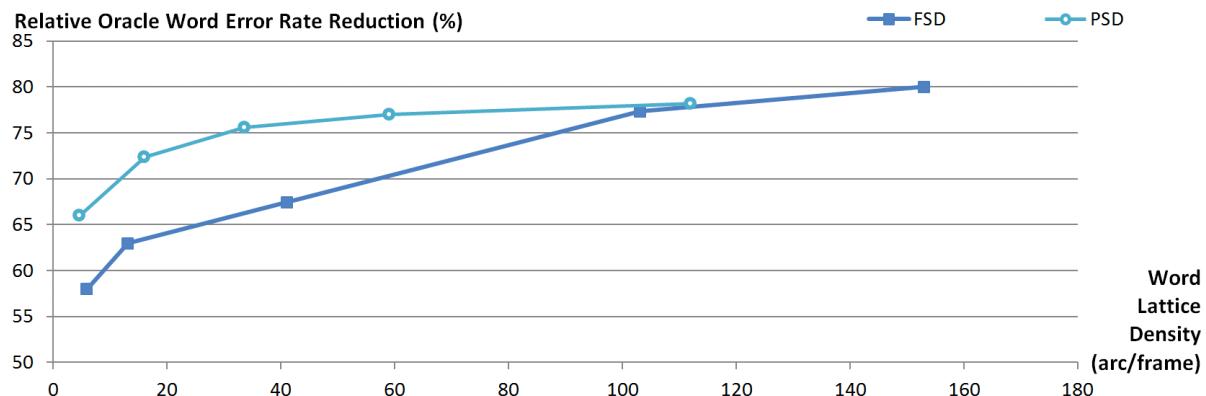


图 7-7 词图密度 v.s. 相对 OWER 下降比例

Fig 7-7 Lattice density v.s. rel. OWER reduction

一旦词级词图被构建好，通过混淆网络的生成中的基于 pivot 的词聚类方法，相似时间边界的词边将会被合并在一起。因此，词级词图的时间边界的准确度对最终的混淆网络影响很大。为了分析词图的时间边界质量，我们提出最近 pivot 边界距离 (NPBD)，它定义为  $|b_{arc} - b_{cluster}| + |e_{arc} - e_{cluster}|$ ，其中  $b_*$  和  $e_*$  为词的开始和结束时间边界， $arc$  是被对齐到最佳重叠 pivot 词 cluster 上的相应边。

图 7-8 显示了 LSD 和 FSD 的 NPBD。从图中可以看出 LSD 的词图的 NPBD 要明显小。换句话说，它的词边界更加稳定，由此可以得到更紧致的混淆网络。

最后我们讨论了从词图到混淆网络的转换。图 7-9 显示了词图密度与混淆网络的深度之间的关系 [200]。结果显示，即使在相似词图密度下，LSD 的混淆网络往往包含更高的混淆网络深度，因此将得到更好的归一化项建模以及置信度建模方案。

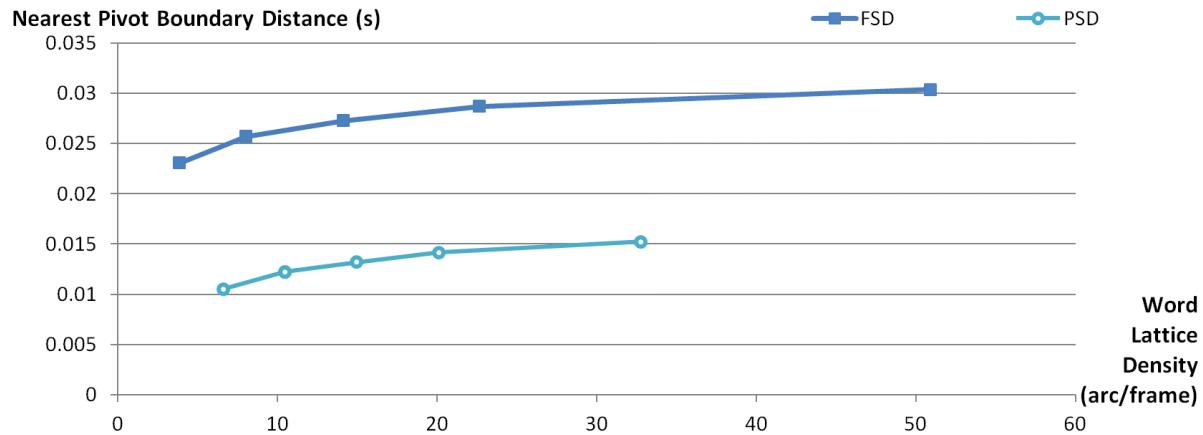


图 7-8 LSD 和 FSD 的词边界稳定性

Fig 7-8 The word boundary stability of LSD and FSD.

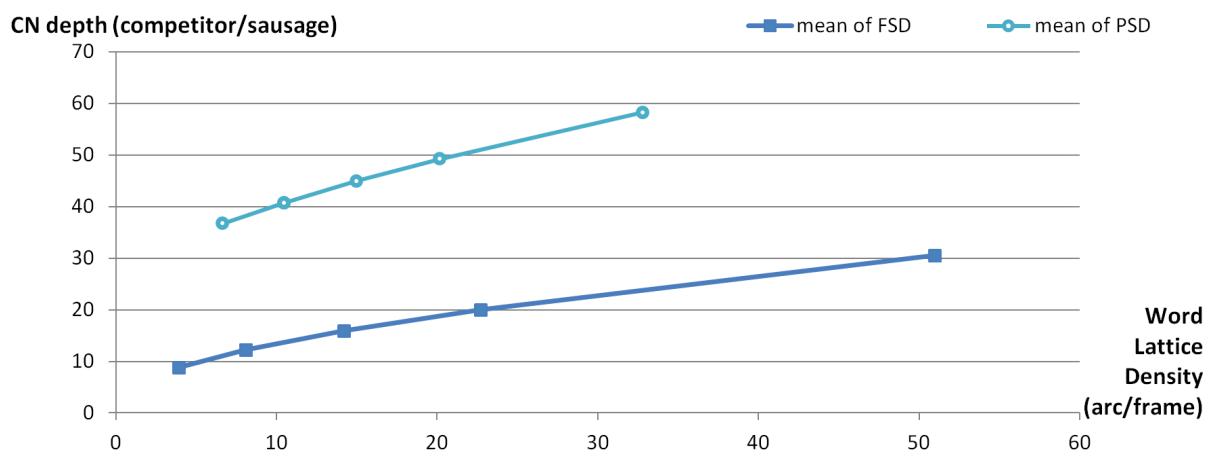


图 7-9 词图密度 v.s. 混淆网络深度

Fig 7-9 Lattice density v.s. the depth of confusion network.

### 7.2.4.2 置信度评估

本章中我们比较上面讨论的混淆网络进一步得到置信度后的性能。为了补偿词图大小和对后验概率的过度估计，这里训练了一个决策树，来将所有概率映射到置信度分数上 [132]。注意到，句子级置信度也可以由类似方法来得到。我们使用上文提到的 NCE 来衡量词级置信度质量优劣。

$$NCE = \frac{H(\mathbf{C}) - H(\mathbf{C}|\mathbf{x})}{H(\mathbf{C})} \quad (7-10)$$

公式中  $H(\mathbf{C})$  表示的是标注的序列的熵， $H(\mathbf{C}|\mathbf{x})$  是置信度序列的熵。这是一个针对相比于总是提供平均分数的系统，其信息量增益的数量的指标。更高的 NCE 代表更好的系统。

表 7-6 比较了所提出的 LSD 音素声学置信度与传统音素后验平均置信度。该传统方法 [125] 最初应用于 HMM，并在本工作中拓展到 FSD 的 CTC 系统中 (表示为 *baseline*)。7.2.2.1 提出的 *peak-mean*, *peak-max*, *phone-mean*, *phone-conf* 等方案被表示为 PN1, PN2, PN3 and PC。结果显示，[125] 提出的音素声学置信度并不能被推广到词级。这是由于词

表 7-6 针对音素声学置信度的比较

AM	Decoding	CM	NCE
DNN-HMM	FSD	baseline	0.024
	FSD	baseline	0.058
LSTM-CTC	LSD	PN1 $\oplus$ PN3	0.105
		PN2 $\oplus$ PN3	0.135
		PN2 $\oplus$ PN3 $\oplus$ PC	0.141

边界的不确定性使得声学分数计算的时候存在重叠，因此表现出了更差的结果。当该方法被使用在 CTC 模型时，词边界重叠问题得到了缓解，但是对于 `blank` 概率的分配仍然存在混淆，比如对任意一个 `blank` 是应该分配给前一个音素还是后一个音素存在不确定性。

在 LSD 框架里，词边界和 `blank` 分配问题都能够被解决，使得我们可以得到质量更好的置信度。同时当声学置信度和基于 *phone-conf* 的置信度进行融合时，往往能取得最好的结果。因此在后续实验中，我们使用  $PN2 \oplus PN3 \oplus PC$  作为所提出的最优置信度。

表 7-7 比较从 LSD 和 FSD 产生的混淆网络的置信度的质量。

该结果显示，虽然这种置信度在 CD-state-HMM 框架下效果较好 (与文献 [132][131] 一致)，但却不能够被直接应用到 CI-phone-CTC 模型上。这也是由于 `blank` 的分配问题。

表 7-7 基于混淆网络的置信度比较

AM	Decoding	CM	NCE
DNN-HMM	FSD	CN	0.172
LSTM-CTC	FSD	CN	0.019
	LSD	CN AC+CN	0.224 0.230

相反，基于 CN 的置信度可以被直接应用到 LSD CTC 音素词图框架并取得显著的性能改善。而且 NCE 还显著好于 CD-state-HMM 系统。我们相信这是因为 CTC 词图包含了更多的竞争信息，使得归一化项建模得更好，如图 7-9 所示。

当 LSD 音素声学置信度 (PN2  $\oplus$  PN3  $\oplus$  PC 在表 7-6) 和基于 CN 的置信度相结合时 (表示为 AC+CN)，性能得到了进一步改善。结果显示这两种置信度具有互补作用。我们认为第一种关注于局部而第二种则注重整个句子的评估。

#### 7.2.4.3 统一解码框架

在第一部分关键词检测任务中，我们使用了 398 个家居领域的关键词和相应的 17332 句测试集，这包括 11789 个正例和 5543 个负例。表 7-8 显示了不同方法之间的差别。

表 7-8 KWS 任务

CM	setup	EER(%)	snt recall(%)	PEA(%)
Phonemic	AC	11.65	88.4	10
	CN	12.55	88.4	11
	AX1	11.60	88.0	10
	AX2	10.16	88.2	16
	AX3	10.10	88.2	15

结果显示 AC 效果比 CN 好。原因是 KWS 中对搜索空间归一化项的建模非常薄弱，使得该部分估计不准。本文所提出的附属归一化搜索空间可以减轻这样的问题，表中显示出了很大的改善。

除此之外，召回率显示本文提出的方法轻微影响了模型的准确度。考虑到在误唤醒方面显著的改善，这样的损失是可以接受的。

在效率方面，CN 计算大约比 AC 占据时间增加 10%。这方面来源于词图和混淆网络生成。虽然 PEA 比 AC 和 CN 方法大，但是总的时间仍然很小。这里的原因是由于使用了

上文的 LSD 技术使得解码整体速度大幅加快 [197]。

第二部分实验基于上下文的语音识别包含 13186 句语音助手语音，其中有 7923 正例和 5263 反例。这些句子包括打电话，命令等。这里的语法包括了所支持的说话方式和联系人信息。这里的负例包含领域内和领域外的错误。

表 7-9 基于上下文的语音识别

CM	setup	EER(%)	snt recall(%)	PEA(%)
Phonemic	AC	19.86	87.4	38
Hypothesis	CN	15.78	87.4	43
	AX1	19.80	86.0	38
	AX2	16.23	87.2	41
	AX3	16.12	87.2	40

表 7-9 给出了结果。在相比之前更大的搜索空间下，CN 显著超越了 AC，因为 AC 并没有使用其它的竞争信息，由此容易造成误唤醒。AX2 和 AX3 类似于 CN。原因是提出我们所提出的附属归一化搜索空间是针对 ASR 搜索空间的一个很好近似。AX3 依然是性能最好的一个系统，我们将其用于下面的实验。

关于效率，我们测试的框架对其基本没有影响。原因是与原来的搜索空间相比，我们提出的附属归一化搜索空间由于非常小，只会带来轻微影响。

第三部分实验将在一个 25 小时左右的对话测试集中进行。解码使用一个 118K 词表的三元语言模型，其包含 1.9M 个词对历史。

表 7-10 LVCSR 任务

CM	setup	NCE	CER(%)	PEA(%)
Phonemic	AC	0.182	10.2	45
Hypothesis	CN	0.302	10.2	50
	AX3	0.260	10.1	46

与前面结论一致，CN 性能好于 AC。所提出的 AX3 的结果比 CN 差但差距不大。原因是对于 LVCSR 的搜索空间建模以及比较完整，附属归一化搜索空间并没有带来什么新的提升。另一方面基于效率考虑，AX3 使用最优路径来近似搜索空间，因此不如基于词图的建模更加完整。

附属归一化搜索空间的系统的 CER 有轻微改善。我们认为是由于该搜索空间的存在将一些错误的搜索路径进行了剪枝，所以减少了插入和替换错误。关于效率，我们所提出的方法与之前一样影响不大。

### 7.3 基于标签同步解码的声学特征到词的端到端语音识别

当前的端到端语音识别系统显示出诸多缺点，在开始使用标签同步解码算法进行优化之前，我们先对这些系统的缺点进行总结。

首先，声学和文本数据并不能够被系统地共同利用，这将导致声学数据的需求量非常大。目前大多数对文本数据的应用停留于初始化阶段，但却不能在一个系统框架里，系统地使用声学和文本数据。与之相对，传统基于深度学习的隐马尔科夫训练框架则通过贝叶斯公式，将建模拆分为声学模型，转移模型，词表和语言模型，由此进行模块化的知识源融合。

其次，声学和语言模型总是使用相同的建模单元，而这样的建模单元往往很难兼顾泛化能力和模型性能。音素感知是人类语言感知的基础单元 [204]。如果直接使用字符或者词作为建模单元，则并不能建立起这样的建模单元与发音之间的直接关系。这样一些研究忽略了音素这一人类语言中的先验知识。另一方面，使用字符而不是单词作为语言模型的建模单元，同样有损性能 [205]. 总的来说，端到端系统建立起了与 NN-HMM 混合系统完全不同的一个模型框架。因此先验知识和之前在语言和声学模型方法的研究很难迁移到这一新的框架中。

该章节将标签同步算法应用于序列到序列的端到端模型的方案。我们使用模块化训练的思想来改善端到端模型建模，使其更易于使用外在知识源来训练每一个端到端模型的子模块。值得注意的是，模型最后需要进行联合优化，因此最终在推理搜索阶段，模型仍然工作在端到端模式下。

我们为端到端模型添加的优化包括：i) 利用声学数据和文本数据来逐一训练端到端系统，并使用模块化的策略将训练结果进行融合。具体来说先使用声学数据训练一个基于 CTC 的音频到音素的声学模型，而后使用文本数据训练一个基于 CTC 或序列到序列模型的音素到词语的语言模型。ii) 在训练的后期，将不同模型融合为音频到词语模型，使用上文提到的标签同步解码算法对声学模型进行降采样，而后进行联合训练优化。最终，音频到音素的声学模型，标签同步解码模块，音素到词语的语言模型被堆叠起来，在联合优化后，可以直接进行端到端推理搜索，保留了端到端系统解码简便的特性。这样做的优点包括：i) 得益于模块化和初始化所带来的更简单的建模难度和更快的模型收敛速度 ii) 更易于与传统的声学建模和语言建模技术相结合，并分别使用声学和文本数据进行训练。

### 7.3.1 训练和解码框架

之前针对端到端语音识别的研究工作集中在将所有的模型单元融合在一起并进行联合训练和端到端解码。在这项工作中，我们提出了一种模块化的训练策略，以便优化外在知识源分别训练各个子模块的潜力。同时，我们在训练结束后仍然保留端到端解码的优点。图 7-10 显示了这一框架。

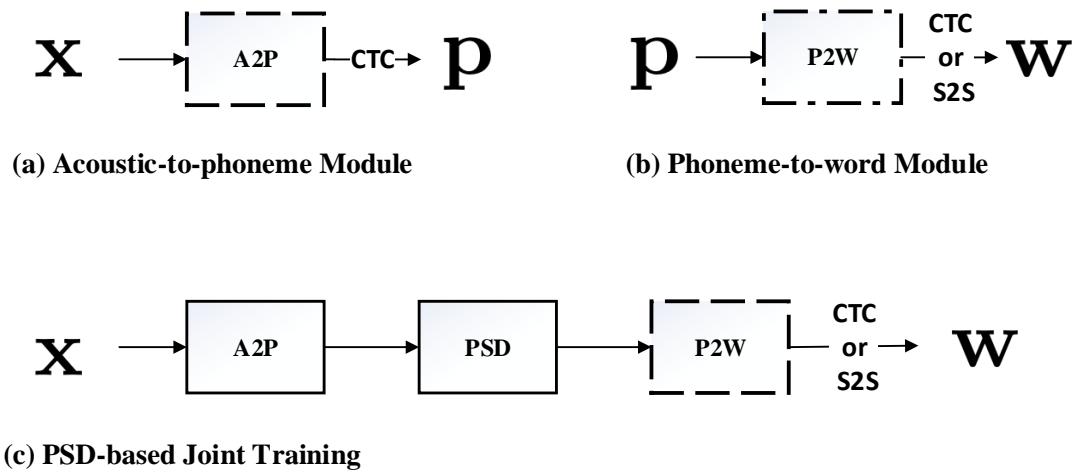


图 7-10 模块化训练策略的框架。实线框表示模型参数固定不变的部分。虚线部分和点划线部分分别表示模型参数使用声学或者文本数据进行训练。

Fig 7-10 The framework of modular training strategy.

这一端到端词序列识别系统被模块化如下：

$$P(\mathbf{w}|\mathbf{x}) \approx \max_{\mathbf{p}} [ P(\mathbf{w}|\mathbf{p}) \cdot P(\mathbf{p}|\mathbf{x}) ] \quad (7-11)$$

公式中  $\mathbf{w}$ ,  $\mathbf{p}$  和  $\mathbf{x}$  为词序列，音素序列和音频特征序列。CTC 准则被用于训练一个使用声学数据的声学到音素模型 (A2P)。同时，我们使用文本数据和 CTC 或 S2S 准则训练一个音素到词语模型。

最终各模块将被融合到一起，成为声学到词语模型 (A2W)，在联合训练过程中我们使用了音素同步解码算法 (PSD) [110]。

$$P(\mathbf{w}|\mathbf{x}) \approx \max_{\mathbf{p}} [ P(\mathbf{w}|\mathbf{p}) \cdot PSD(P(\mathbf{p}|\mathbf{x})) ] \quad (7-12)$$

在解码阶段，经过联合训练后的模型将进行端到端推理搜索，因此其复杂度与普通端到端模型相同 [111]。

对于 CTC，我们将最佳的推理搜索序列连接起来，得到最终解码序列。对于 S2S 模型，我们使用维特比束剪枝算法来得到相应结果。这里的端到端模型还可以进一步与外在语言模型相结合以改善性能。在这种情况下，N 元语言模型将被编译为词 WFST。这种情况下 LSD 搜索算法 [110] 还可以进一步用于加速系统。

### 7.3.2 模块化

由于音素是人类语言重要的先验知识，它建立起了声学和语言之间的关系，因此我们采用它作为理想的建模单元。声学到音素模块使用声学数据进行训练，并预测  $P(\mathbf{p}|\mathbf{x})$  如图 7-10(a) 所示，其与传统音素 CTC 建模相同 [186]。注意的是，虽然 CTC 被用于该项工作，其它传统声学模型同样可以被使用，比如 RNA [206] 或者 LF-MMI [83]。

不同于传统模型，我们这里所提出的语言模型使用音素作为输入，推理搜索词序列，即  $P(\mathbf{w}|\mathbf{p})$ ，也就是一个音素到词的模型，如图 7-10(b) 所示。另一个主要不同是，P2W 模块使用文本和词典，不再需要使用声学数据。总的来说，P2W 模块类似于传统语言模型，除了以下几方面：

- i) P2W 输入的是音素序列，并隐含进行文本切分。
- ii) P2W 给定音素序列情况下预测词序列。因此不同于传统的语言模型专注于预测下一个词，在给定前词情况下。P2W 实际上得到了更多关于下一个词的信息。因此实验结果显示，P2W 的预测准确度明显高于普通的语言模型。
- iii) P2W 通过序列级准则进行训练，比如 CTC 和 S2S，这样就能够自动学习到音素和词语之间的对齐关系。

我们这里同时引入了一个词边界单元  $\text{wb}$  到音素列表中，以改善文本切分效果。 $\text{wb}$  出现在词语的边界处，比如对词语 “okay ow k ey”，其被修改为 “okay ow k ey wb”。这里的想法是使用  $\text{wb}$  作为文本切分的线索，比如它可以区分一些短词可能是长词前缀的情况。

总结这里的模块化的优点包括：i) 声学和语言模型都使用了合理的建模单元 ii) 系统地结合了声学和文本数据的使用来强化模型训练。

### 7.3.3 标签同步解码

标签同步解码在前文已进行了充分介绍，下面我们将使用它， $PSD(\cdot)$ ，作为融合系统中的一个降采样模块，应用在 A2P 模块上，以减少输入序列的长度。我们这样做

的优点包括：i) 更短的序列减轻了 LSTM 进行时序建模的难度 ii) 加速了联合优化过程 iii) 该方法同时也加速了解码速度 [110].

### 7.3.4 联合训练

最终，所有模块将会被堆叠起来，如图 7-10(c) 所示。声学数据将用来调优该模型。词语级别的 CTC 准则使用方法类似于 [138]。同时 S2S 被用于词级别的预测。在优化过程中，只有 P2W 进行联合训练，原因是：i) A2P 工作在音素上，通常已经能取得较好性能 [186, 140]. ii) 固定住 A2P 使得 PSD 可以显著减少音频帧数，加快训练速度。经过联合训练后，该模型仍工作在端到端推理搜索上，保留了优点。

### 7.3.5 实验结果

表7-11针对各个模块的验证集 (CV) 上的性能进行了比较。加粗的系统将在后续中进行使用。

表 7-11 各个模块的性能比较

模块	模型	推理搜索单元	词边界	PER/WER CV (%)
A2P	CTC	phoneme	×	13.0
			√	<b>12.0</b>
P2W	CTC	word	×	16.0
			√	<b>4.3</b>
	S2S	word	×	13.9
			√	<b>2.8</b>

在 A2P 中，音素 CTC 的识别性能与 [43] 中一致。引入的 wb 并不影响 ASR 性能。该系统中轻微的性能改善来自统计 PER 时包含了 wb 这一单元。而进一步统计 wb 的预测准则显示其低于 4%。

在 P2W 模块中，CTC 和 S2S 都进行了比较。在不包含 wb 情况下，二者的性能都不好。正如前文所讨论的，wb 可以提供音素序列切分的线索，因此 CTC 和 S2S 在包含 wb 之后都得到了提升。S2S 一致地由于 CTC，这是由于 CTC 中的 CIA 被去除，使建模能力得到增强 [78]。不同于传统的语言模型，这里不适用 PPL 作为准则，原因是本系统自动推理搜索出音素与词之间的对齐关系，并使用序列准则进行训练。

在第二部分中，我们比较了联合训练之后的结果，显示在表 7-12 中。为了更好地支撑这些结果，我们与近期在同一数据集上的公开结果进行比较 [111]。这里的不同实验配置包括：i) 添加 i-vector 自适应 ii) 使用 BLSTM iii) 语言模型使用了 Fisher 数据集。因此本实验系统与 [111] 中的系统包含相对 20-30% 的差距。

表 7-12 针对是否包含模块化训练的性能比较

Name	E2E Opt.	Modularization		WER (%)	
		A2P	P2W	swbd	callhm
CD-phone CE	×	HMM	WFST	14.9	27.6
CI-phone CTC	×	CTC	WFST	19.4	33.5
Word CTC	✓	n/a	n/a	29.6	41.7
Mod. CTC	✓	CTC	CTC	24.9	36.5
	✓	CTC	+WFST	23.0	35.1
Mod. S2S	✓	CTC	S2S	31.2	40.5

基线的混合系统 (CD-phone CE) 和音素 CTC (CI-phone CTC) 系统罗列于第一和二行。他们都是通过 WFST 联合解码得到的。CI-phone CTC 的性能比 CD-phone CE 差<sup>1</sup>，这里的性能差距类似于之前在 [111] 中的发现。直接进行 A2W 建模的 CTC (Word CTC) 模型在第三行。它包含了音素模型初始化，但不包含 GloVe 初始化 [111]。这里的性能明显差于 CI-phone CTC。该系统为不带有模块预训练的基线系统。

我们所提出的模块化训练系统 (Mod. CTC) 在第四行。我们使用了基于标签同步解码的联合训练，该工作的具体效果将在表 7-13 中继续讨论。Mod. CTC 显著改善了基线系统的性能。更好的性能来自：i) 得益于模块化和初始化所带来的更快和更好的模型收敛 ii) 更易于分别使用声学和文本数据来融合传统的声学和语音模型训练技术。

表 7-13 显示了引入 PSD 算法的重要性。所有的结果都是基于一张 Titan GPU 所得到。“fr./s.” 表示每秒钟所处理的声学帧数。这里的训练加速来自两方面：i) PSD 减少了后续 P2W 需要处理的序列长度 ii) 由于序列长度有所减小，更多的序列可以被载入 GPU 显存，由此可以加速并行计算。同时，性能也有所改善。我们认为改善的结果来自于序列长度的缩短。虽然我们使用 LSTM，但模型仍然很难记忆很长的序列。但是对于 A2W 建模，需要记忆的历史显著长于传统的 CI-phone CTC 或者混合系统。在其它一些研究中 [78] 也有类似的现象，而这些研究通过金字塔式的降帧率来缓解该问题。PSD 是解决该问题的另一种思路。

<sup>1</sup>在更多数据情况下，CTC 性能比传统系统好 [140]。我们之前的研究也显示同样的结论 [110]

表 7-13 是否包含 PSD 情况下的模型性能和训练速度

Name	PSD	训练速度		WER (%)	
		Seq./GPU	fr./s.	swbd	callhm
Mod. CTC	✗	5	1027	32.0	42.5
	✓	<b>30</b>	<b>5851</b>	<b>24.9</b>	<b>36.5</b>

为了减轻 CTC 的 CIA 假设对模型性能所带来影响，我们实验了以下一些方案。首先，一个从 N 元语言模型得到的 WFST 被用于与模型一起进行联合解码，其结果显示在表 7-12 的第五行。该系统得到了一定性能提升。由此第 2 行与第 5 行的性能差距被缩减到少于 15%。另一种方法是使用 S2S 来代替 CTC。我们提出的基于模块化训练的 S2S 系统 (Mod. S2S) 在表 7-12 中最后一行。不同于在表 7-11 中的现象，S2S 系统并未得到性能改善。通过进一步分析解码结果，我们发现 S2S 很易于受到音素识别错误的影响。经过联合优化后，S2S 并不能恢复这些错误，这项观察同样见于其它研究 [207]。在我们的研究中尚未包含字符级别的建模系统，原因是字符对于声学和语言建模的难度都比较大，如前文所述，音素和词语对本文的建模更加直接。

## 7.4 本章小结

本章节我们针对标签同步解码算法的三个扩展应用：关键词检测，多识别任务统一置信度框架，以及端到端语音识别。我们基于前面章节介绍的 LSD 算法提出了一种高效的后处理算法，以解决关键词检测中的音素混淆建模的问题。我们将结合前面章节对解码的大幅加速和搜索空间的优化等工作，提出一系列针对不同应用的通用置信度，并尝试将不同应用中的语音识别推理搜索过程统一到同一框架中。另一方面，我们将标签同步算法应用于序列到序列的端到端模型。在实验部分，该章节系统一方面取得大幅度语音识别解码速度改善，另一方面在端到端建模上取得了更快和更好的模型收敛和模型准确度。

## 第八章 全文总结

本论文围绕深度序列模型的建模和解码展开了一系列探索和研究。本论文的主要贡献在第四章、第五章、第六章、第七章中介绍。第一个贡献是系统地研究序列建模方法，并应用到一系列非传统语音识别任务中，具体将在第8.1章总结；第二个贡献是提出并行的解码搜索算法并在 GPU 上实现开源该套算法，具体将在第8.2章总结；第三个贡献是基于端到端建模，系统地提出了标签同步算法，具体将在第8.3章总结。上述两项贡献从不同层面使得解码搜索技术得到了极大幅度加速。第四个贡献是将标签同步算法进一步应用到更多语音识别应用中，具体将在第8.4章总结。除上述贡献外，作者在端到端建模和鲁棒语音识别建模中均有贡献，概述于第2.4章和第4.2章。可能的后续工作展望在第8.5章中总结。

### 8.1 非传统识别任务的序列建模

本论文针对关键词检测和多说话人重叠语音信号识别这两类任务提出了序列建模方案。

在本文中，我们为深度学习的声学非固定关键词的关键词检测设计了相应的序列鉴别性训练方法，同时该方法也可以应用到固定关键词的关键词检测中。对这两种框架，竞争可能性的建模都是核心难题。本论文提出采用无词图鉴别性训练框架来解决这一问题：隐性使用音素或半词单元的语言模型来建模。

近年来随着深度学习发展，将语音信号的增强和语音识别进行联合训练成为另一种新的趋势。这样做的好处包括：更好地利用序列信息进行训练；对模型前后模块进行联合调优以解决模型失配问题。在单输出语音识别中，由于语音识别本身是一种序列分类和预测问题，序列级的准则将有助于改善系统性能。而在鸡尾酒会问题中，对于单通道多说话人混叠语音识别系统，则又包含了说话人跟踪问题，也属于序列级问题。因此序列鉴别性准则将有助于这样的序列分类问题。我们提出了一种传统鉴别性训练技术变种，它在进行鉴别性训练的同时，也抑制输出通道上说话人跟踪错误。在多说话人重叠语音信号识别任务中，本文通过联合优化，迁移学习，序列鉴别性训练等方式，改善了原来语音分离、信号增强和语音识别的联合训练系统。

## 8.2 基于 GPU 并行计算的搜索速度优化

在本论文中，我们针对 Kaldi 开源工具包的推理搜索部分进行了一项重要扩展，以便使它能够支持图形处理芯片（GPU）上的 WFST 解码推理搜索。该框架可以显著加速现有推理搜索算法，特别是在基于深度序列学习的一系列模型上进行了验证。

这是一个通用的离线解码器，该解码器对语言模型和声学模型没有特别的限制，并且可以工作在各种架构的 GPU 上。为了支持第二遍重打分和更丰富的后处理，我们的设计基于 WFST 解码和词图生成的架构 [122]。针对设计中的几大难点，我们提出了如下解决方案：我们将维特比算法中的令牌合并操作实现为一个 GPU 并行计算中的原子操作，以便减少维特比束剪枝算法中同步消耗；我们提出了动态负载均衡的方式以更高效地进行并行计算，提高其多线程之间的利用率；我们重新设计了基于 GPU 并行计算的精确的词图生成和剪枝算法，以便充分利用 GPU 的性能特点。

在 Switchboard 上实验表明，我们所提出的方法在取得完全一致的 1-best 和词图质量情况下，可以得到 3-15 倍的加速，并在绝大部分 GPU 架构上进行了验证。除此之外，如果再进行多句子的并行处理，最终的加速比将达到 46 倍。同时我们对这项工作进行了开源<sup>1</sup>，它将与大多数 Kaldi 脚本相兼容。这项工作作为 Kaldi 工具包的一个扩展 [150]，完整实现了基于 GPU 并行计算的 WFST 解码。

## 8.3 基于标签同步解码的搜索空间优化

基于端到端建模，我们系统地提出了标签同步算法，其通过一系列方法使得搜索解码过程从逐帧同步变为标签同步，这包括使用高效的 blank 结构和后处理方法。该文提出的一系列通用方法在隐马尔科夫模型和连接时序分类模型上得到了验证。同时我们还介绍了将标签同步算法应用于序列到序列的端到端模型的方案，使之取得了更快和更好的模型收敛和模型准确度。

自动语音识别等序列标注任务的一个独特点是其对相邻帧的时序序列关联性建模。用于对相邻帧进行时序建模的主流序列模型包括隐马尔科夫模型 (Hidden Markov Model, HMM) 和连接时序分类模型 (Connectionist Temporal Classification, CTC)。针对这些模型，当前主流的推理搜索方法是帧层面的维特比束搜索算法，该算法复杂度很高，限制了语音识别的广泛应用。深度学习的发展使得更强的上下文和历史建模成为可能。通过引入 blanks (“空”) 单元，端到端建模系统能直接预测标签在给定特征下的后验概率。

我们提出将特征层面的搜索过程改变为标签层面，即搜索空间是由不同历史的标签

<sup>1</sup><https://github.com/chenzhehuai/kaldi/tree/gpu-decoder>

组成的，使得解码速率等于标注速率，从而小于特征速率。具体来说，在标签推理搜索阶段，对帧层面声学模型的输出增加一步后处理过程：i) 判断当前帧是否存在标签输出；ii) 若有，执行搜索过程；若无，则丢弃标签输出。因此该后处理过程可被看作是每个输出标签概率计算的近似。与传统方法相比，该方法的优势是搜索空间更小，且搜索过程被大大加速。我们提出的一系列通用方法在隐马尔科夫模型和连接时序分类模型上得到了验证。

在实验部分，我们提出的系统取得了大幅度语音识别解码速度改善。

## 8.4 标签同步解码的扩展应用

标签同步算法同时还能产生高质量的音素词图，称为 CTC 音素词图。基于 CTC 音素词图，本文进一步探讨标签同步解码算法的一些扩展应用。这包括关键词检测，多识别任务统一置信度框架，以及端到端语音识别。

在关键词检测中，我们基于前面章节介绍的 LSD 算法提出了一种高效的后处理算法，以解决音素混淆建模的问题。在基于 CTC 的关键词检测中，虽然 CTC 是一个序列级准则，但是并没有对非关键词部分进行建模。因此该准则改善了关键词之间的鉴别性，但是没有改善关键词和非关键词之间的鉴别性。本文将非关键词建模单元直接引入到建模当中。同时我们基于前述 LSD 算法得到的音素词图，提出了一套基于编辑距离的后处理算法以引入音素混淆性，使系统更加鲁棒。

连接时序分类模型 (CTC) 是一种目前比较主流的 LVCSR 模型。但是由于 blank 的引入，使得基于 CTC 的词语级别的置信度 (CM) 并不能够被直接得到，特别是基于最主流的针对传统基于音素似然度归一化或者基于词图后验概率的混淆网络等方法。在前面提出的标签同步解码算法的基础上，我们进一步提出了两种置信度生成算法。更细致的研究显示这种基于 CTC 的音素词图是得到更好性能的关键所在。在英文 Switchboard 上的大词汇连续语音识别任务显示这里提出的 LSD CTC 词图置信度算法可以显著改善原先传统的基于逐帧解码算法的 CTC 置信度或者 HMM 模型的置信度。另一方面由于不同 ASR 应用之间不同的搜索空间大小和效率要求，当前业界最优的置信度及其相应的解码算法在不同应用上具有不同架构，这些不同应用包括：关键词检测，基于上下文的语音识别和大词汇连续语音识别。针对基于词图后验概率的置信度，计算量主要集中在词图部分的边缘概率计算过程。本论文中，我们提出一系列针对不同应用的通用置信度，并尝试将不同应用中的语音识别推理搜索过程统一到同一框架中。具体来说，我们提出了辅助归一化搜索空间的概念。我们尝试使用这样的搜索空间来建模所有 ASR 应用领域的置信度。而针对这样做在低功耗设备上带来的挑战，我们采用基于 CTC 的标

签同步解码 [197] 来进行处理，由此带来了很大的效率改善。最终这一统一高效的置信度框架被应用于目前主流的上述三种 ASR 应用。

同时本论文还研究了将标签同步算法应用于直接建模输出序列形态学组合的端到端模型的方案。我们使用模块化训练的思想来改善端到端模型建模，使其更易于使用外在知识源来训练每一个端到端模型的子模块。值得注意的是，模型最后需要进行联合优化，因此最终在推理搜索阶段，模型仍然工作在端到端模式下。在实验部分，该章节系统一方面取得大幅度语音识别解码速度改善，另一方面在端到端建模上取得了更快和更好的模型收敛和模型准确度。

## 8.5 后续工作展望

本论文围绕深度学习背景下的序列建模和解码搜索技术展开了一系列探索和研究。进一步的后续研究方向包括：

- 在本论文中，我们将并行计算的思想第一次应用到图搜索算法中。而自然语言处理中包含大量图搜索算法，将这种思路进一步推广和应用到其它任务中将会是非常有意义的研究。
- 在本论文中，我们基于端到端建模来简化传统语音识别中的搜索解码计算量。未来的研究包括两方面：一是进行更加端到端的建模，以进一步逐渐简化搜索解码；二是如何从深度学习的角度解释传统图搜索算法，由此提出更高效的深度学习模型来替代图搜索算法。
- 在本论文中，我们研究了关键词检测系统的序列鉴别性训练。采用其它的序列训练或序列准则以进一步改善关键词检测系统是一个有意义的研究话题，特别是尝试与本论文中提到的一些端到端建模方法相结合。
- 最后，将本论文提出的并行解码搜索技术与标签同步解码算法相结合，并推广到所提出的通用推理搜索和置信度框架中，得到一个速度大幅提升，同时性能和置信度得到改善的通用系统，将会非常有意义。同时得益于速度的大幅提升和框架的简化，该方案具有进一步推广到端侧边缘计算的极大想象空间。

## 参考文献

- [1] KH Davis, R Biddulph and Stephen Balashek. “*Automatic recognition of spoken digits*”. *The Journal of the Acoustical Society of America*, **1952**, 24(6): 637–642.
- [2] James Baker. “*The DRAGON system—an overview*”. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **1975**, 23(1): 24–29.
- [3] Frederick Jelinek. “*Continuous speech recognition by statistical-methods*”. *Proceedings of the IEEE*, **1976**, 64(4): 532–556.
- [4] Tasos Anastasakos, John McDonough, Richard Schwartz *et al*. “*A compact model for speaker-adaptive training*”. In: *Proc. International Conference on Spoken Language Processing (ICSLP)*, **1996**: 1137–1140.
- [5] Vassilios V Digalakis, Dimitry Rtischev and Leonardo G Neumeyer. “*Speaker adaptation using constrained estimation of Gaussian mixtures*”. *IEEE Transactions on Speech and Audio Processing*, **1995**, 3(5): 357–366.
- [6] Saduoki Furui. “*Unsupervised speaker adaptation based on hierarchical spectral clustering*”. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **1989**, 37(12): 1923–1930.
- [7] Mark JF Gales. “*Cluster adaptive training for speech recognition*.” In: *Proc. International Conference on Spoken Language Processing (ICSLP)*, **1998**: 1783–1786.
- [8] Mark JF Gales. “*Maximum likelihood linear transformations for HMM-based speech recognition*”. *Computer Speech & Language*, **1998**, 12(2): 75–98.
- [9] Mark JF Gales. “*Adaptive training for robust ASR*”. In: *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, **2001**: 15–20.
- [10] Mark JF Gales. “*Multiple-cluster adaptive training schemes*”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **2001**: 361–364.
- [11] J-L Gauvain and Chin-Hui Lee. “*Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains*”. *IEEE Transactions on Speech and Audio Processing*, **1994**, 2(2): 291–298.

- [12] Roland Kuhn, Patrick Nguyen, Jean-Claude Junqua *et al.* “*Eigenvoices for speaker adaptation.*” In: *Proc. International Conference on Spoken Language Processing (ICSLP)*, **1998**: 1774–1777.
- [13] Li Lee and Richard C Rose. “*Speaker normalization using efficient frequency warping procedures*”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **1996**: 353–356.
- [14] Christopher J Leggetter and Philip C Woodland. “*Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models*”. *Computer Speech & Language*, **1995**, 9(2): 171–185.
- [15] Leonardo Neumeyer, Ananth Sankar and Vassilios Digalakis. “*A comparative study of speaker adaptation techniques*”. In: *Proc. European Conference on Speech Communication and Technology (EUROSPEECH)*, **1995**.
- [16] David Pye and Philip C Woodland. “*Experiments in speaker normalisation and adaptation for large vocabulary speech recognition*”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **1997**: 1047–1050.
- [17] LR Bahl, Peter F Brown, Peter V De Souza *et al.* “*Maximum mutual information estimation of hidden Markov model parameters for speech recognition*”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **1986**: 49–52.
- [18] Ralf Schlüter, Wolfgang Macherey, Boris Müller *et al.* “*Comparison of discriminative training criteria and optimization methods for speech recognition*”. *Speech Communication*, **2001**, 34(3): 287–310.
- [19] Wu Chou, Chin-Hui Lee and Biing-Hwang Juang. “*Minimum error rate training based on N-best string models*”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **1993**: 652–655.
- [20] Vaibhava Goel and William J Byrne. “*Minimum Bayes-risk automatic speech recognition*”. *Computer Speech & Language*, **2000**, 14(2): 115–135.
- [21] Biing-Hwang Juang, Wu Hou and Chin-Hui Lee. “*Minimum classification error rate methods for speech recognition*”. *IEEE Transactions on Speech and Audio Processing*, **1997**, 5(3): 257–265.

- [22] Daniel Povey. *Discriminative training for large vocabulary speech recognition* [phdthesis], **2005**.
- [23] Daniel Povey and Philip C Woodland. “*Improved discriminative training techniques for large vocabulary continuous speech recognition*”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **2001**: 45–48.
- [24] Stanley F Chen and Joshua Goodman. “*An empirical study of smoothing techniques for language modeling*”. *Computer Speech & Language*, **1999**, 13(4): 359–394.
- [25] Hervé Bourlard and Nelson Morgan. “*A continuous speech recognition system embedding MLP into HMM*.” In: *Proc. Neural Information Processing Systems (NIPS)*, **1989**: 186–193.
- [26] Hervé Bourlard, Nelson Morgan, Chuck Wooters *et al.* “*CDNN: a context dependent neural network for continuous speech recognition*”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **1992**: 349–352.
- [27] Hervé Bourlard and Nelson Morgan. *Connectionist speech recognition: a hybrid approach*. Springer Science & Business Media, **2012**.
- [28] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer London, **2014**. <https://books.google.com/books?id=rUBTBQAAQBAJ>.
- [29] George E Dahl, Dong Yu, Li Deng *et al.* “*Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition*”. *IEEE Transactions on Audio, Speech, and Language Processing*, **2012**, 20: 30–42.
- [30] Geoffrey E Hinton, Li Deng, Dong Yu *et al.* “*Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups*”. *IEEE Signal Processing Magazine*, **2012**, 29: 82–97.
- [31] Yanmin Qian, Mengxiao Bi, Tian Tan *et al.* “*Very deep convolutional neural networks for noise robust speech recognition*”. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **2016**, 24(12): 2263–2276.
- [32] Vijayaditya Peddinti, Daniel Povey and Sanjeev Khudanpur. “*A time delay neural network architecture for efficient modeling of long temporal contexts*”. In: *Proc. Annual Conference of International Speech Communication Association (INTERSPEECH)*, **2015**: 3214–3218.

- [33] Dario Amodei, Rishita Anubhai, Eric Battenberg *et al.* “Deep Speech 2: End-to-End Speech Recognition in English and Mandarin”. In: *Proc. International Conference on Machine Learning (ICML)*, **2016**.
- [34] Dong Yu, Wayne Xiong, Jasha Droppo *et al.* “Deep convolutional neural networks with layer-wise context expansion and attention.” In: *Proc. Annual Conference of International Speech Communication Association (INTERSPEECH)*, **2016**: 17–21.
- [35] Wayne Xiong, Jasha Droppo, Xuedong Huang *et al.* “The microsoft 2016 conversational speech recognition system”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **2017**: 5255–5259.
- [36] Mehryar Mohri, Fernando Pereira and Michael Riley. “Weighted finite-state transducers in speech recognition”. *Computer Speech & Language*, **2002**, 16(1): 69–88.
- [37] G David Forney. “The viterbi algorithm”. *Proceedings of the IEEE*, **1973**, 61(3): 268–278.
- [38] Irving J Good. “The population frequencies of species and the estimation of population parameters”. *Biometrika*, **1953**, 40(3-4): 237–264.
- [39] Slava Katz. “Estimation of probabilities from sparse data for the language model component of a speech recognizer”. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **1987**, 35(3): 400–401.
- [40] Peter F Brown, Peter V Desouza, Robert L Mercer *et al.* “Class-based n-gram models of natural language”. *Computational linguistics*, **1992**, 18(4): 467–479.
- [41] Tomáš Mikolov, Martin Karafiat, Lukas Burget *et al.* “Recurrent neural network based language model.” In: *Proc. Annual Conference of International Speech Communication Association (INTERSPEECH)*, **2010**: 3.
- [42] Tomáš Mikolov. “Statistical language models based on neural networks”. *Presentation at Google, Mountain View, 2nd April, 2012*.
- [43] Alex Graves, Santiago Fernández, Faustino Gomez *et al.* “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks”. In: *Proceedings of the 23rd international conference on Machine learning*, **2006**: 369–376.
- [44] Mingkun Huang, Yongbin You, Zhehuai Chen *et al.* “Knowledge Distillation for Sequence Model.” In: *Interspeech 2018*.

- [45] Hynek Hermansky. “*Perceptual linear predictive (PLP) analysis of speech*”. *the Journal of the Acoustical Society of America*, **1990**, 87(4): 1738–1752.
- [46] Steven B Davis and Paul Mermelstein. “*Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences*”. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **1980**, 28: 357–366.
- [47] Frank Seide, Gang Li, Xie Chen *et al.* “*Feature engineering in context-dependent deep neural networks for conversational speech transcription*”. In: *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, **2011**: 24–29.
- [48] Sadaoki Furui. “*Speaker-independent isolated word recognition using dynamic features of speech spectrum*”. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **1986**, 34(1): 52–59.
- [49] Nagendra Kumar and Andreas G Andreou. “*Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition*”. *Speech Communication*, **1998**, 26(4): 283–297.
- [50] Bishnu S Atal. “*Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification*”. *the Journal of the Acoustical Society of America*, **1974**, 55(6): 1304–1312.
- [51] Philip C Woodland, Christopher J Leggetter, JJ Odell *et al.* “*The development of the 1994 HTK large vocabulary speech recognition system*”. In: *Proceedings ARPA workshop on spoken language systems technology*, **1995**: 104–109.
- [52] Thomas Hain, Philip C Woodland, Gunnar Evermann *et al.* “*Automatic transcription of conversational telephone speech*”. *IEEE Transactions on Speech and Audio Processing*, **2005**, 13(6): 1173–1185.
- [53] Philip C Woodland, Julian J Odell, Valtcho Valtchev *et al.* “*Large vocabulary continuous speech recognition using HTK*”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **1994**: II–125.
- [54] Steve J Young and Philip C Woodland. “*The use of state tying in continuous speech recognition*”. In: *Proc. European Conference on Speech Communication and Technology (EUROSPEECH)*, **1993**: 2203–2206.

- [55] Steve J Young, Julian J Odell and Philip C Woodland. “*Tree-based state tying for high accuracy acoustic modelling*”. In: *Proceedings of the workshop on Human Language Technology*, **1994**: 307–312.
- [56] Leonard E Baum, John Alonzo Eagon *et al.* “*An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology*”. *Bull. Amer. Math. Soc.*, **1967**, 73(3): 360–363.
- [57] Arthur P Dempster, Nan M Laird and Donald B Rubin. “*Maximum likelihood from incomplete data via the EM algorithm*”. *Journal of the royal statistical society. Series B (methodological)*, **1977**: 1–38.
- [58] Ian H Witten and Timothy C Bell. “*The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression*”. *IEEE Transactions on Information Theory*, **1991**, 37(4): 1085–1094.
- [59] Hermann Ney, Ute Essen and Reinhard Kneser. “*On the estimation of ‘small’ probabilities by leaving-one-out*”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **1995**, 17(12): 1202–1212.
- [60] David E Rumelhart, Geoffrey E Hinton and Ronald J Williams. “*Learning representations by back-propagating errors*”. *NATURE*, **1986**, 323(6088): 533.
- [61] Yann LeCun, Léon Bottou, Yoshua Bengio *et al.* “*Gradient-based learning applied to document recognition*”. *Proceedings of the IEEE*, **1998**, 86(11): 2278–2324.
- [62] Alex Graves, Navdeep Jaitly and Abdel-rahman Mohamed. “*Hybrid speech recognition with deep bidirectional LSTM*”. In: *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, **2013**: 273–278.
- [63] Alex Graves, Abdel-rahman Mohamed and Geoffrey E Hinton. “*Speech recognition with deep recurrent neural networks*”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **2013**: 6645–6649.
- [64] Haşim Sak, Andrew Senior and Françoise Beaufays. “*Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition*”. *arXiv preprint arXiv:1402.1128*, **2014**.

- [65] Hasim Sak, Oriol Vinyals, Georg Heigold *et al.* “Sequence discriminative distributed training of long short-term memory recurrent neural networks”. In: *Proc. Annual Conference of International Speech Communication Association (INTERSPEECH)*, **2014**.
- [66] Yoshua Bengio, Patrice Simard and Paolo Frasconi. “Learning long-term dependencies with gradient descent is difficult”. *IEEE Transactions on Neural Networks*, **1994**, 5(2): 157–166.
- [67] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. *Neural Computation*, **1997**, 9(8): 1735–1780.
- [68] Dong Yu, Li Deng and George E Dahl. “Roles of pre-training and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition”. In: *Proc. Neural Information Processing Systems (NIPS)*, **2010**.
- [69] Léon Bottou. “Online learning and stochastic approximations”. *On-line learning in neural networks*, **1998**, 17(9): 142.
- [70] Boris T Polyak. “Some methods of speeding up the convergence of iteration methods”. *USSR Computational Mathematics and Mathematical Physics*, **1964**, 4(5): 1–17.
- [71] Yann LeCun, Léon Bottou, Genevieve B Orr *et al.* “Efficient backprop”. In: *Neural networks: Tricks of the trade*. Springer, **1998**: 9–50.
- [72] Edmondo Trentin and Marco Gori. “A survey of hybrid ANN/HMM models for automatic speech recognition”. *Neurocomputing*, **2001**, 37(1-4): 91–126.
- [73] Hervé Bourlard and Christian J Wellekens. “Links between Markov models and multilayer perceptrons”. In: *Proc. Neural Information Processing Systems (NIPS)*, **1989**: 502–510.
- [74] Nelson Morgan and Hervé Bourlard. “Continuous speech recognition using multilayer perceptrons with hidden Markov models”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **1990**: 413–416.
- [75] Anthony J Robinson, Gary D Cook, Daniel PW Ellis *et al.* “Connectionist speech recognition of broadcast news”. *Speech Communication*, **2002**, 37(1-2): 27–45.
- [76] Frank Seide, Gang Li and Dong Yu. “Conversational speech transcription using context-dependent deep neural networks.” In: *Proc. Annual Conference of International Speech Communication Association (INTERSPEECH)*, **2011**: 437–440.

- [77] G. Hinton, L. Deng, D. Yu *et al.* “*Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups*”. *IEEE Signal Processing Magazine*, **2012**, 29(6): 82–97.
- [78] William Chan. *End-to-End Speech Recognition Models* [phdthesis], **2016**.
- [79] Vassilios Digalakis, Jan Robin Rohlicek and Mari Ostendorf. “*A dynamical system approach to continuous speech recognition*”. In: *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, **1991**: 289–292.
- [80] Pieter Abbeel, Adam Coates, Michael Montemerlo *et al.* “*Discriminative Training of Kalman Filters*.” In: *Robotics: Science and systems*, **2005**: 1.
- [81] Guoguo Chen, Hainan Xu, Minhua Wu *et al.* “*Pronunciation and silence probability modeling for ASR*.” In: *Interspeech 2015*, **2015**: 533–537.
- [82] Santiago Fernández, Alex Graves and Jürgen Schmidhuber. “*An application of recurrent neural networks to discriminative keyword spotting*”. In: *Artificial Neural Networks–ICANN 2007*. Springer, **2007**: 220–229.
- [83] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez *et al.* “*Purely sequence-trained neural networks for ASR based on lattice-free MMI*”. *Interspeech 2016*, **2016**.
- [84] Stanley F Chen, Brian Kingsbury, Lidia Mangu *et al.* “*Advances in speech transcription at IBM under the DARPA EARS program*”. *IEEE Transactions on Audio, Speech, and Language Processing*, **2006**, 14(5): 1596–1608.
- [85] Hossein Hadian, Hossein Sameti, Daniel Povey *et al.* “*End-to-end speech recognition using lattice-free MMI*”. In: *Proc. Interspeech*, **2018**: 12–16.
- [86] Ian McGraw, Rohit Prabhavalkar, Raziel Alvarez *et al.* “*Personalized speech recognition on mobile devices*”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, **2016**: 5955–5959.
- [87] Zhehuai Chen, Mahaveer Jain, Yongqiang Wang *et al.* “*End-to-end Contextual Speech Recognition using Class Language Models and a Token Passing Decoder*”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, UK, **2019**.
- [88] Andrew Viterbi. “*Error bounds for convolutional codes and an asymptotically optimum decoding algorithm*”. *IEEE Transactions on Information Theory*, **1967**, 13(2): 260–269.

- [89] Steve J Young, Gunnar Evermann, Mark JF Gales *et al.* “*The HTK book*”. Cambridge university engineering department, **2002**, 3: 175.
- [90] Richard Schwartz and Y-L Chow. “*The N-best algorithms: an efficient and exact procedure for finding the N most likely sentence hypotheses*”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **1990**: 81–84.
- [91] Stefan Ortmanns, Hermann Ney and Xavier Aubert. “*A word graph algorithm for large vocabulary continuous speech recognition*”. *Computer Speech & Language*, **1997**, 11(1): 43–72.
- [92] Lina Zhou, Yongmei Shi, Jinjuan Feng *et al.* “*Data mining for detecting errors in dictation speech recognition*”. *IEEE transactions on speech and audio processing*, **2005**, 13(5): 681–688.
- [93] Wade Shen, Christopher M White and Timothy J Hazen. *A comparison of query-by-example methods for spoken term detection* [techreport], **2009**.
- [94] John S Garofolo, Cedric GP Auzanne and Ellen M Voorhees. “*The TREC spoken document retrieval track: A success story*”. In: *Content-Based Multimedia Information Access-Volume 1*, **2000**: 1–20.
- [95] Guoguo Chen, Carolina Parada and Georg Heigold. “*Small-footprint keyword spotting using deep neural networks*”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, **2014**: 4087–4091.
- [96] Yaodong Zhang and James R Glass. “*Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriograms*”. In: *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, **2009**: 398–403.
- [97] MS Barakat, Christian H Ritz and David A Stirling. “*An improved template-based approach to keyword spotting applied to the spoken content of user generated video blogs*”. In: *Multimedia and Expo (ICME), 2012 IEEE International Conference on*, **2012**: 723–728.
- [98] Guoguo Chen, Carolina Parada and Tara N Sainath. “*Query-by-example keyword spotting using long short-term memory networks*”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **2015**: 5236–5240.
- [99] Kenney Ng and Victor W Zue. “*Subword-based approaches for spoken document retrieval*”. *Speech Communication*, **2000**, 32(3): 157–186.

- [100] Anupam Mandal, KR Prasanna Kumar and Pabitra Mitra. “Recent developments in spoken term detection: a survey”. *International Journal of Speech Technology*, **2014**, 17(2): 183–198.
- [101] Rafid A Sukkar, Anand R Setlur, Mazin G Rahim *et al.* “Utterance verification of keyword strings using word-based minimum verification error (WB-MVE) training”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, **1996**: 518–521.
- [102] Steve J Young and Philip C Woodland. “State clustering in hidden Markov model-based continuous speech recognition”. *Computer Speech & Language*, **1994**, 8(4): 369–383.
- [103] Douglas B Paul. “An efficient A\* stack decoder algorithm for continuous speech recognition with a stochastic language model”. In: *Proceedings of the workshop on Speech and Natural Language*, **1992**: 405–409.
- [104] Stephan Kanthak, Kai Schutz and Hermann Ney. “Using SIMD instructions for fast likelihood calculation in LVCSR”. In: *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, **2000**: 1531–1534.
- [105] Jike Chong, Ekaterina Gonina, Youngmin Yi *et al.* “A fully data parallel WFST-based large vocabulary continuous speech recognition on a graphics processing unit”. In: *Tenth Annual Conference of the International Speech Communication Association*, **2009**.
- [106] Jian Xue, Jinyu Li, Dong Yu *et al.* “Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network”. In: *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **2014**: 6359–6363.
- [107] Vijayaditya Peddinti, Yiming Wang, Daniel Povey *et al.* “Low latency acoustic modeling using temporal convolution and LSTMs”. *IEEE Signal Processing Letters*, **2018**, 25(3): 373–377.
- [108] Golan Pundak and Tara N Sainath. “Lower Frame Rate Neural Network Acoustic Models”. *Interspeech 2016*, **2016**: 22–26.
- [109] Zhehuai Chen, Wei Deng, Tao Xu *et al.* “Phone Synchronous Decoding with CTC Lattice”. In: *Interspeech 2016*, **2016**: 1923–1927.

- [110] Z. Chen, Y. Zhuang, Y. Qian *et al.* “*Phone Synchronous Speech Recognition With CTC Lattices*”. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **2017**, 25(1): 86–97.
- [111] Kartik Audhkhasi, Bhuvana Ramabhadran, George Saon *et al.* “*Direct Acoustics-to-Word Models for English Conversational Speech Recognition*”. *arXiv preprint arXiv:1703.07754*, **2017**.
- [112] Zhehuai Chen, Qi Liu, Hao Li *et al.* “*On Modular Training of Neural Acoustics-to-word Model for LVCSR*”. In: *ICASSP*, **2018**.
- [113] Xuedong Huang, K-F Lee and H-W Hon. “*On semi-continuous hidden Markov modeling*”. In: *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, **1990**: 689–692.
- [114] Jun Cai, Ghazi Bouselmi, Yves Laprie *et al.* “*Efficient likelihood evaluation and dynamic Gaussian selection for HMM-based speech recognition*”. *Computer Speech & Language*, **2009**, 23(2): 147–164.
- [115] Xiaolong Li and Yunxin Zhao. “*A fast and memory-efficient N-gram language model lookup method for large vocabulary continuous speech recognition*”. *Computer Speech & Language*, **2007**, 21(1): 1–25.
- [116] Antonio Cardenal-López, F Javier Diéguez-Tirado and Carmen García-Mateo. “*Fast LM look-ahead for large vocabulary continuous speech recognition using perfect hashing*”. In: *Proceedings of International Conference on Acoustics, Speech and Signal Processing (CASSP, 2002)*: I–705.
- [117] Marijn Huijbregts, Roeland Ordelman and Franciska de Jong. “*Fast N-Gram language model look-ahead for decoders with static pronunciation prefix trees*”. In: *Ninth Annual Conference of the International Speech Communication Association*, **2008**.
- [118] Stephen John Young, NH Russell and JHS Thornton. *Token passing: a simple conceptual model for connected speech recognition systems*. Cambridge University Engineering Department Cambridge, UK, **1989**.
- [119] Janne Pylkkönen. “*New pruning criteria for efficient decoding*”. In: *Ninth European Conference on Speech Communication and Technology*, **2005**.

- [120] Hagen Soltau and George Saon. “*Dynamic network decoding revisited*”. In: *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, **2009**: 276–281.
- [121] David Nolden, Ralf Schlüter and Hermann Ney. “*Search space pruning based on anticipated path recombination in lvcsr*”. In: *Interspeech*, **2012**.
- [122] Daniel Povey, Mirko Hannemann, Gilles Boulian et al. “*Generating exact lattices in the WFST framework*”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, **2012**: 4213–4216.
- [123] Sherry Ruan, Jacob O Wobbrock, Kenny Liou et al. “*Speech Is 3x Faster than Typing for English and Mandarin Text Entry on Mobile Devices*”. *arXiv preprint arXiv:1608.07323*, **2016**.
- [124] Hui Jiang. “*Confidence measures for speech recognition: A survey*”. *Speech communication*, **2005**, 45(4): 455–470.
- [125] Wenping Hu, Yao Qian and Frank K Soong. “*A new DNN-based high quality pronunciation evaluation for computer-aided language learning (CALL)*.” In: *INTERSPEECH*, **2013**: 1886–1890.
- [126] Zejun Ma, Xiaorui Wang and Bo Xu. “*Fusing Multiple Confidence Measures for Chinese Spoken Term Detection*”. In: *Twelfth Annual Conference of the International Speech Communication Association*, **2011**.
- [127] Daniele Falavigna, Roberto Gretter and Giuseppe Riccardi. “*Acoustic and word lattice based algorithms for confidence scores*.” In: *INTERSPEECH*, **2002**.
- [128] Mathew Stephen Seigel. *Confidence Estimation for Automatic Speech Recognition Hypotheses* [phdthesis], **2013**.
- [129] Dong Yu, Jinyu Li and Li Deng. “*Calibration of confidence measures in speech recognition*”. *IEEE Transactions on Audio, Speech, and Language Processing*, **2011**, 19(8): 2461–2473.
- [130] Sheryl R Young. “*Detecting misrecognitions and out-of-vocabulary words*”. In: *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, **1994**: II–21.

- [131] Frank Wessel, Ralf Schluter, Klaus Macherey *et al.* “*Confidence measures for large vocabulary continuous speech recognition*”. *IEEE Transactions on speech and audio processing*, **2001**, 9(3): 288–298.
- [132] Gunnar Evermann and Philip C Woodland. “*Large vocabulary decoding and confidence estimation using word posterior probabilities*”. In: *Acoustics, Speech, and Signal Processing, 2000. ICASSP’00. Proceedings. 2000 IEEE International Conference on*, **2000**: 1655–1658.
- [133] Peng Yu, Duo Zhang and Frank Seide. “*Maximum entropy based normalization of word posteriors for phonetic and lvcsr lattice search*”. In: *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, **2006**: I–I.
- [134] Karel Vesely, Lukáš Burget and František Grézl. “*Parallel training of neural networks for speech recognition*”. In: *International Conference on Text, Speech and Dialogue*, **2010**: 439–446.
- [135] Paul R Dixon, Tasuku Oonishi and Sadaoki Furui. “*Harnessing graphics processors for the fast computation of acoustic likelihoods in speech recognition*”. *Computer Speech & Language*, **2009**, 23(4): 510–526.
- [136] Kisun You, Jike Chong, Youngmin Yi *et al.* “*Parallel scalability in speech recognition*”. *IEEE Signal Processing Magazine*, **2009**, 26(6).
- [137] Dario Amodei *et al.* “*Deep Speech 2: End-to-End Speech Recognition in English and Mandarin*”. *arXiv preprint arXiv:1512.02595*, **2015**.
- [138] Hagen Soltau, Hank Liao and Hasim Sak. “*Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition*”. *arXiv preprint arXiv:1610.09975*, **2016**.
- [139] Ronan Collobert, Christian Puhrsch and Gabriel Synnaeve. “*Wav2letter: an end-to-end convnet-based speech recognition system*”. *arXiv preprint arXiv:1609.03193*, **2016**.
- [140] Haşim Sak, Andrew Senior, Kanishka Rao *et al.* “*Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition*”. *arXiv preprint arXiv:1507.06947*, **2015**.
- [141] Tara N Sainath and Carolina Parada. “*Convolutional neural networks for small-footprint keyword spotting*”. In: *Proc. Annual Conference of International Speech Communication Association (INTERSPEECH)*, **2015**: 1478–1482.

- [142] Rafid A Sukkar and Chin-Hui Lee. “*Vocabulary independent discriminative utterance verification for nonkeyword rejection in subword based speech recognition*”. *IEEE Transactions on Speech and Audio Processing*, **1996**, 4(6): 420–429.
- [143] Eric D Sandness and I Lee Hetherington. “*Keyword-based discriminative training of acoustic models*.” In: *Interspeech 2000*, **2000**: 135–138.
- [144] Joseph Keshet, David Grangier and Samy Bengio. “*Discriminative keyword spotting*”. *Speech Communication*, **2009**, 51(4): 317–329.
- [145] Daniel Povey, Dimitri Kanevsky, Brian Kingsbury *et al.* “*Boosted MMI for model and feature-space discriminative training*”. In: *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, **2008**: 4057–4060.
- [146] Matthew Gibson and Thomas Hain. “*Hypothesis spaces for minimum Bayes risk training in large vocabulary speech recognition*.” In: *Interspeech 2006*, **2006**: 2406–2409.
- [147] Zhong Meng and Biing-Hwang Juang. “*Non-Uniform Boosted MCE Training of Deep Neural Networks for Keyword Spotting*”. *Interspeech 2016*, **2016**: 770–774.
- [148] Zhehuai Chen, Yanmin Qian and Kai Yu. “*Sequence Discriminative Training for Deep Learning based Acoustic Keyword Spotting*”. *Speech Communication*, **2018**.
- [149] John Garofalo, David Graff, Doug Paul *et al.* “*Continous Speech Recognition (CSR-I) Wall Street Journal (WSJ0) news, complete*”. *Linguistic data consortium, philadelphia*, **1993**.
- [150] Daniel Povey, Arnab Ghoshal, Gilles Boulianne *et al.* “*The kaldi speech recognition toolkit*”. In: *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, **2011**.
- [151] Vijayaditya Peddinti, Yiming Wang, Daniel Povey *et al.* *Low latency modeling of temporal contexts*.
- [152] Karel Veselý, Arnab Ghoshal, Lukáš Burget *et al.* “*Sequence-discriminative training of deep neural networks*.” In: *Proc. Annual Conference of International Speech Communication Association (INTERSPEECH)*, **2013**: 2345–2349.
- [153] Daniel Povey and Brian Kingsbury. “*Evaluation of proposed modifications to MPE for large scale discriminative training*”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, **2007**: IV–321.

- [154] Fengpei Ge and Yonghong Yan. “Deep Neural Network Based Wake-Up-Word Speech Recognition with Two-stage Detection”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*. New Orleans, USA, **2017**: 2761–2765.
- [155] E Colin Cherry. “Some experiments on the recognition of speech, with one and with two ears”. *The Journal of the acoustical society of America*, **1953**, 25(5): 975–979.
- [156] Albert S Bregman. *Auditory scene analysis: The perceptual organization of sound*. MIT press, **1994**.
- [157] DeLiang Wang and Guy J Brown. *Computational auditory scene analysis: Principles, algorithms, and applications*. Wiley-IEEE press, **2006**.
- [158] Martin Cooke, John R Hershey and Steven J Rennie. “Monaural speech separation and recognition challenge”. *Computer Speech & Language*, **2010**, 24(1): 1–15.
- [159] Jun Du, Yanhui Tu, Yong Xu *et al*. “Speech separation of a target speaker based on deep neural networks”. In: *Signal Processing (ICSP), 2014 12th International Conference on*, **2014**: 473–477.
- [160] Chao Weng, Dong Yu, Michael L Seltzer *et al*. “Deep neural networks for single-channel multi-talker speech recognition”. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, **2015**, 23(10): 1670–1679.
- [161] Dong Yu, Xuankai Chang and Yanmin Qian. “Recognizing Multi-talker Speech with Permutation Invariant Training”. *CoRR*, **2017**, *abs/1704.01985*. <http://arxiv.org/abs/1704.01985v3>.
- [162] Zhehuai Chen, Jasha Droppo, Jinyu Li *et al*. “Progressive joint modeling in unsupervised single-channel overlapped speech recognition”. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, **2018**, 26(1): 184–196.
- [163] John J Godfrey, Edward C Holliman and Jane McDaniel. “SWITCHBOARD: Telephone speech corpus for research and development”. In: *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, **1992**: 517–520.
- [164] Frank Seide and Amit Agarwal. “CNTK: Microsoft’s Open-Source Deep-Learning Toolkit”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, **2016**: 2135–2135.

- [165] L. Lu and S. Renals. “*Small-Footprint Highway Deep Neural Networks for Speech Recognition*”. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **2017**, 25(7): 1502–1511.
- [166] Steve Young. “*The HTK book version 3.4. 1*”. In: **2009**.
- [167] *CUDA Toolkit Documentation*.
- [168] Leslie Lamport. “*How to make a multiprocessor computer that correctly executes multiprocess programm*”. *IEEE transactions on computers*, **1979**, (9): 690–691.
- [169] Jungsuk Kim, Kisun You and Wonyong Sung. “*H-and C-level WFST-based large vocabulary continuous speech recognition on graphics processing units*”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, **2011**: 1733–1736.
- [170] Charith Mendis, Jasha Droppo, Saeed Maleki *et al.* “*Parallelizing WFST speech decoders*”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, **2016**: 5325–5329.
- [171] Ali M Alakeel. “*A Guide to dynamic Load balancing in Distributed Computer Systems*”. In: *International Journal of Computer Science and Network Security (IJCSNS)*, **2010**.
- [172] Lidia Mangu, Eric Brill and Andreas Stolcke. “*Finding consensus among words: Lattice-based word error minimization*”. In: *Sixth European Conference on Speech Communication and Technology*, **1999**.
- [173] Jonathan G Fiscus. “*A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER)*”. In: *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, **1997**: 347–354.
- [174] Jungsuk Kim and Ian Lane. “*Accelerating large vocabulary continuous speech recognition on heterogeneous cpu-gpu platforms*”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, **2014**: 3291–3295.
- [175] Andrej Ljolje, Fernando Pereira and Michael Riley. “*Efficient general lattice generation and rescoring*”. In: *Sixth European Conference on Speech Communication and Technology*, **1999**.
- [176] Björn Hoffmeister, Tobias Klein, Ralf Schlüter *et al.* “*Frame based system combination and a comparison with weighted ROVER and CNC*.” In: *INTERSPEECH*, **2006**.

- [177] Zhehuai Chen, Yimeng Zhuang and Kai Yu. “*Confidence Measures for CTC-based Phone Synchronous Decoding*”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, **2017**.
- [178] Zhehuai Chen, Yimeng Zhuang and Kai Yu. “*Confidence measures for CTC-based phone synchronous decoding*”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, **2017**: 4850–4854.
- [179] Zhehuai Chen, Yanmin Qian and Kai Yu. “*A Unified Confidence Measure Framework Using Auxiliary Normalization Graph*”. In: *International Conference on Intelligent Science and Big Data Engineering*, **2017**: 123–133.
- [180] Takaaki Hori, Chiori Hori, Yasuhiro Minami *et al.* “*Efficient WFST-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition*”. *Audio, Speech, and Language Processing, IEEE Transactions on*, **2007**, 15(4): 1352–1365.
- [181] Volker Steinbiss, Bach-Hiep Tran and Hermann Ney. “*Improvements in beam search.*” In: *ICSLP*, **1994**: 2143–2146.
- [182] Hugo Van Hamme and Filip Van Aelten. “*An adaptive-beam pruning technique for continuous speech recognition*”. In: *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, **1996**: 2083–2086.
- [183] Gabriel Pereyra, George Tucker, Jan Chorowski *et al.* “*Regularizing neural networks by penalizing confident output distributions*”. *arXiv preprint arXiv:1701.06548*, **2017**.
- [184] Vincent Vanhoucke, Matthieu Devin and Georg Heigold. “*Multiframe deep neural networks for acoustic modeling*”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, **2013**: 7582–7585.
- [185] Yajie Miao, Mohammad Gowayyed, Xingyu Na *et al.* “*An empirical exploration of CTC acoustic models*”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, **2016**: 2623–2627.
- [186] Yajie Miao, Mohammad Gowayyed and Florian Metze. “*EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding*”. *arXiv preprint arXiv:1507.08240*, **2015**.

- [187] Johann Hauswald, Michael A Laurenzano, Yunqi Zhang *et al.* “*Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers*”. In: *ACM SIGPLAN Notices*, **2015**: 223–238.
- [188] Takaaki Hori and Atsushi Nakamura. “*Speech recognition algorithms using weighted finite-state transducers*”. *Synthesis Lectures on Speech and Audio Processing*, **2013**, 9(1): 1–162.
- [189] Z. Chen, Y. Zhuang, Y. Qian *et al.* “*Phone Synchronous Speech Recognition With CTC Lattices*”. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **2017**, 25(1): 86–97.
- [190] Yajie Miao, Jinyu Li, Yongqiang Wang *et al.* “*Simplifying long short-term memory acoustic models for fast training and decoding*”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, **2016**: 2284–2288.
- [191] Matthias Paulik. “*Improvements to the Pruning Behavior of DNN Acoustic Models*”. In: *Sixteenth Annual Conference of the International Speech Communication Association*, **2015**.
- [192] Yimeng Zhuang, Xuankai Chang, Yanmin Qian *et al.* “*Unrestricted Vocabulary Keyword Spotting Using LSTM-CTC*”. *Interspeech 2016*, **2016**: 938–942.
- [193] Upendra V Chaudhari and Michael Picheny. “*Improvements in phone based audio search via constrained match with high order confusion estimates*”. In: *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, **2007**: 665–670.
- [194] Naoyuki Kanda, Xugang Lu and Hisashi Kawai. “*Minimum Bayes Risk Training of CTC Acoustic Models in Maximum a Posteriori Based Decoding Framework*”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*. New Orleans, USA, **2017**: 4855–4859.
- [195] Jinyu Li, Rui Zhao, Zhuo Chen *et al.* “*Developing far-field speaker system via teacher-student learning*”. *arXiv preprint arXiv:1804.05166*, **2018**.
- [196] Richard C Rose, Biing-Hwang Juang and Chin-Hui Lee. “*A training procedure for verifying string hypotheses in continuous speech recognition*”. In: *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, **1995**: 281–284.

- [197] Zhehuai Chen, Wei Deng, Tao Xu *et al.* “*Phone Synchronous Decoding with CTC Lattice*”. In: *Interspeech 2016*, **2016**: 1923–1927. <http://dx.doi.org/10.21437/Interspeech.2016-831>.
- [198] Wayne Ward and Sunil Issar. “*A class based language model for speech recognition*”. In: *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, **1996**: 416–418.
- [199] Lucy Vasserman, Ben Haynor and Petar Aleksic. “*Contextual language model adaptation using dynamic classes*”. In: *Spoken Language Technology Workshop (SLT), 2016 IEEE*, **2016**: 441–446.
- [200] Dilek Hakkani-Tür, Frédéric Béchet, Giuseppe Riccardi *et al.* “*Beyond ASR 1-best: Using word confusion networks in spoken language understanding*”. *Computer Speech & Language*, **2006**, 20(4): 495–514.
- [201] Santiago Fernández, Alex Graves and Jürgen Schmidhuber. “*Phoneme recognition in TIMIT with BLSTM-CTC*”. *arXiv preprint arXiv:0804.3269*, **2008**.
- [202] Tara Sainath, Kanishka Rao *et al.* “*Acoustic modelling with CD-CTC-SMBR LSTM RNNS*”. In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, **2015**: 604–609.
- [203] Sabato Marco Siniscalchi, Torbjorn Svendsen and Chin-Hui Lee. “*A bottom-up modular search approach to large vocabulary continuous speech recognition*”. *Audio, Speech, and Language Processing, IEEE Transactions on*, **2013**, 21(4): 786–797.
- [204] David B Pisoni, Howard C Nusbaum, Paul A Luce *et al.* “*Speech perception, word recognition and the structure of the lexicon*”. *Speech communication*, **1985**, 4(1-3): 75–95.
- [205] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster *et al.* “*Exploring the limits of language modeling*”. *arXiv preprint arXiv:1602.02410*, **2016**.
- [206] Haşim Sak, Matt Shannon, Kanishka Rao *et al.* “*Recurrent Neural Aligner: An Encoder-Decoder Neural Network Model for Sequence to Sequence Mapping*”. *Proc. Interspeech 2017*, **2017**: 1298–1302.
- [207] Rohit Prabhavalkar, Kanishka Rao, Tara N Sainath *et al.* “*A Comparison of Sequence-to-Sequence Models for Speech Recognition*”. *Proc. Interspeech 2017*, **2017**: 939–943.



## 攻读学位期间发表的学术论文

- [1] **Zhehuai Chen**, Jasha Droppo, Jinyu Li, Wayne Xiong, Progressive Joint Modeling in Unsupervised Single-channel Overlapped Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 1, pp. 184-196, Jan. 2018. doi: 10.1109/TASLP.2017.2765834.
- [2] **Zhehuai Chen**, Yimeng Zhuang, Yanmin Qian, Kai Yu. Phone Synchronous Speech Recognition with CTC Lattices. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 25, no. 1, pp. 86-97, Jan. 2017. doi: 10.1109/TASLP.2016.2625459.
- [3] **Zhehuai Chen**, Yanmin Qian, Kai Yu, Sequence Discriminative Training for Deep Learning based Acoustic Keyword Spotting. *Speech Communication*, vol. 102, 100-111, 2018.
- [4] **Zhehuai Chen**, Mahaveer Jain, Yongqiang Wang, Michael Seltzer, Christian Fuegen, End-to-end Contextual Spebech Recognition using Class Language Models and a Token Passing Decoder, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, 2019.
- [5] **Zhehuai Chen**, Justin Luitjens, Hainan Xu, Yiming Wang, Daniel Povey, Sanjeev Khudanpur, A GPU-based WFST Decoder with Exact Lattice Generation, *19th Annual Conference of International Speech Communication Association (InterSpeech)*, 2018 [**Best Paper Nomination**].
- [6] **Zhehuai Chen**, Linguistic Search Optimization for Deep Learning Based LVCSR, in Doctoral Consortium, *19th Annual Conference of International Speech Communication Association (InterSpeech)*, 2018.
- [7] **Zhehuai Chen**, Jasha Droppo, Sequence Modeling in Unsupervised Single-channel Overlapped Speech Recognition, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018.

- [8] **Zhehuai Chen**, Qi Liu, Hao Li, Kai Yu, On Modular Training of Neural Acoustics-to-word Model for LVCSR, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, Canada, 2018.
- [9] **Zhehuai Chen**, Yanmin Qian, and Kai Yu. A unified confidence measure framework using auxiliary normalization graph, IScIDE, 2017.
- [10] **Zhehuai Chen**, Yimeng Zhuang, Kai Yu. Confidence Measures for CTC-based Phone Synchronous Decoding. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, USA, 2017.
- [11] **Zhehuai Chen**, Wei Deng, Tao Xu, Kai Yu. Phone Synchronous Decoding with CTC Lattice. 17th Annual Conference of the International Speech Communication Association (InterSpeech), San Francisco, America, 2016.
- [12] **Zhehuai Chen**, Kai Yu, An Investigation of Implementation and Performance Analysis of DNN Based Speech Synthesis System. 12th IEEE International Conference on Signal Processing (ICSP), Hangzhou, 2014.
- [13] 陈哲怀, 郑文露, 游永彬, 钱彦旻, 俞凯, 标签同步解码算法及其在语音识别中的应用. 计算机学报(投出), 2019.
- [14] Mingkun Huang, Yongbin You, **Zhehuai Chen**, Yanmin Qian, Kai Yu. Knowledge Distillation for Sequence Model, 19th Annual Conference of International Speech Communication Association (InterSpeech), 2018.
- [15] Yue Wu, Tianxing He, **Zhehuai Chen**, Yanmin Qian and Kai Yu. Multi-view LSTM Language Model with Word-synchronized Auxiliary Feature for LVCSR, CCL, 2017.
- [16] Da Zheng, **Zhehuai Chen**, Yue Wu, Kai Yu, Directed Automatic Speech Transcription Error Correction Using Bidirectional LSTM. International Symposium on Chinese Spoken Language Processing (ISCSLP), Tianjin, China, 2016.
- [17] Bo Chen, **Zhehuai Chen**, Jiachen Xu, Kai Yu. An Investigation of Context Clustering for Statistical Speech Synthesis with Deep Neural Network. 16th Annual Conference of

the International Speech Communication Association (InterSpeech), Dresden, Germany,  
2015.



## 攻读学位期间申请的专利

- [1] voice recognition based decoding method and device PCT/CN2016/081334
- [2] Sequence discriminative training based wakeup word device CN201710343427.5
- [3] lattice rescoring system of deep learning based language model CN201810054749.2
- [4] self-tagging, single direction and multi-view language modeling CN201710561261.4
- [5] a decoding method and device of speech recognition CN201610221182.4
- [6] adaptative wakeup word method and device CN201610462976.X
- [7] an audio recognition method and device CN201810025834.6
- [8] a speech recognition method and device CN201810054315.2
- [9] confidence measure based speech recognition method and device CN201710060942.2
- [10] model compression and optimization of speech recognition device CN201810021903.6



## 致 谢

TODO