

申请上海交通大学博士学位论文

语音识别中深度序列模型的解码搜索技术研究

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 2019 年 X 月 X 日

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

Inference Framework and Search Optimization for Deep Sequential Models 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, 2019

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

语音识别中深度序列模型的解码搜索技术研究

摘要

TODO

关键词： TODO TODO

Inference Framework and Search Optimization for Deep Sequential Models in Speech Recognition

ABSTRACT

TODO

KEY WORDS: TODO, TODO

目 录

插图索引	x
表格索引	xi
主要符号对照表	xiii
第一章 绪论	1
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	3
1.2 自动语音识别的推理问题	5
1.3 论文主要内容、创新点及组织结构	7
第二章 自动语音识别	9
2.1 自动语音识别框架	9
2.1.1 特征提取	9
2.1.2 声学模型	11
2.1.3 语言模型	19
2.1.4 解码及搜索	20
2.2 大词汇连续语音识别中的解码搜索技术	21
2.2.1 解码器的技术流派	22
2.2.2 结构和主要模块	24
2.3 本章小结	29
第三章 语音识别中的深度序列模型	31
3.1 深度学习与神经网络	31
3.1.1 深度神经网络	31
3.1.2 卷积神经网络	33
3.1.3 循环神经网络	35
3.2 神经网络训练	36

3.2.1	训练准则	36
3.2.2	反向传播算法	37
3.2.3	小批量随机梯度下降	39
3.2.4	惯性系数	39
3.2.5	参数初始化与预训练	40
3.3	深度神经网络-隐马尔可夫模型混合系统	42
3.4	端到端语音识别中的深度序列模型	44
3.4.1	无词图鉴别性训练模型	44
3.4.2	连接时序模型	45
3.4.3	基于注意力机制的序列到序列模型	47
3.5	鲁棒语音识别中的深度序列模型	47
3.5.1	排列不变性训练及其在鸡尾酒会问题中的应用	47
3.5.2	基于迁移学习的排列不变性训练	48
3.5.3	基于序列鉴别性训练的排列不变性训练	52
3.6	本章小结	52
第四章	基于 GPU 并行计算的搜索速度优化	55
4.1	引言	55
4.2	维特比算法并行化框架	56
4.2.1	系统框架	56
4.3	并行的令牌传递算法	57
4.4	图搜索的动态负载均衡算法	58
4.5	并行的词图处理算法	59
4.5.1	精确的词图生成算法	59
4.5.2	词图剪枝	60
4.6	实验结果	60
4.6.1	实验配置	60
4.6.2	性能和速度	62
4.6.3	分析	62
4.7	本章小结	64
第五章	基于标签同步解码的搜索空间优化	67
5.1	引言	67
5.2	基于连接时序模型的标签同步解码	69

5.2.1	语音识别解码算法研究现状分析	69
5.2.2	基于端到端建模的标签同步推理	72
5.3	基于无词图鉴别性训练模型的标签同步解码	74
5.4	基于标签同步解码的端到端语音识别	77
5.4.1	训练和解码框架	78
5.4.2	模块化	79
5.4.3	标签同步解码	80
5.4.4	联合训练	80
5.5	实验结果	80
5.5.1	DSM 实验	81
5.5.2	GSM 实验	84
5.5.3	标签同步解码在模块化训练中的应用	87
5.6	本章小结	89
第六章	基于标签同步解码的统一解码框架	91
6.1	基于标签同步解码的置信度框架	91
6.2	基于标签同步解码的多识别任务统一框架	91
6.3	实验结果	91
6.4	本章小结	91
第七章	关键词检测的序列建模和标签同步解码	93
7.1	基于深度学习的关键词检测及其置信度	93
7.2	关键词检测的序列鉴别性训练	93
7.3	关键词检测的标签同步解码	93
7.4	实验结果	93
7.5	本章小结	93
第八章	全文总结	95
8.1	基于 GPU 并行计算的搜索速度优化	95
8.2	基于标签同步解码的搜索空间优化	95
8.3	基于标签同步解码的统一解码框架	95
8.4	关键词检测的序列建模和标签同步解码	95
8.5	后续工作展望	95
参考文献		97

攻读学位期间发表的学术论文 **111**

致 谢 **113**

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	4
2-1 隐马尔可夫模型	11
2-2 单高斯三音素的状态聚类	18
2-3 通用解码器架构	25
2-4 N-Best 候选序列和词图	29
2-5 词图的用途举例	29
3-1 深度神经网络	32
3-2 激活函数	33
3-3 卷积操作和池化操作	34
3-4 受限玻尔兹曼机	40
3-5 DNN-HMM 混合系统	42
3-6 端到端系统举例	44
3-7 传统系统与端到端系统比较	46
3-8 PIT-ASR 联合训练与模块化初始化和逐步联合训练的比较。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。	49
3-9 模型系统训练框架。[102] 中提出的结构由虚线框部分表示，其用于推理出每个说话人的语音信息。该结构被模块化 (三个实线框) 并作逐一增量的预训练。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的神经网络上进行。	50
3-10 基于迁移学习的逐步联合训练方法。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。	50
3-11 原始联合训练和所提出的方法在验证集上的学习曲线比较。在图中，联合训练，逐步联合训练，基于迁移学习的逐步联合训练被分别表示为 <i>Joint Model</i> , <i>Pro. Joint Model</i> 和 <i>Pro. Joint Model + Transf.</i> 图中每个 epoch 包含 24 小时训练数据。	51
4-1 并行维特比束剪枝算法以及精确词图处理系统的框架	56

4-2 一个针对动态负载均衡的例子。这里的虚线框表示一个 <i>CUDA cooperative group</i> ，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，并通知组分里的其他线程。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。	59
4-3 语言模型大小，帧率， <i>GPU</i> 架构的比较	64
4-4 针对解码器的时间占比分析	65
5-1 <i>HMM</i> , <i>CTC</i> 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， <i>B</i> 指 <i>blank HMM</i> 状态， <i>P</i> 指标签输出 <i>HMM</i> 状态。每个圆圈代表一个由神经网络建模发射概率的 <i>HMM</i> 状态。其中，点划线圆圈表示输出标签建模，如 (b) <i>CTC</i> 中的 <i>l</i> , 都各自分配一个特定的模型单元。虚线圆圈表示 <i>blank</i> 建模，但并不完全相同，如 (b) <i>CTC</i> 中的 <i></i> 是使用公共的 <i>blank</i> 建模；但 (c) 中的 <i>q2</i> , 每个输出标签有独立的 <i>blank</i> 建模，本文将详细比较不同 <i>blank</i> 的粒度和拓扑结构所带来的区别。其它实线小圆圈，如 (c) 中 <i>q0</i> 、 <i>q3</i> , (d) 中 <i>q0</i> 、 <i>q3</i> , (e) 中 <i>q0</i> 、 <i>q4</i> , 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。	74
5-2 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。	78
5-3 <i>LSD</i> 和 <i>FSD</i> 框架中平均活跃令牌数随 <i>LM</i> 变大的变化趋势。（为清晰起见，这里仅绘制 <i>swb</i> 子集， <i>calhm</i> 子集具有类似变化趋势。）	82
5-4 对于 <i>swb</i> 子集， <i>CTC</i> 中，使用不同剪枝技术时 <i>WER</i> 随平均活跃令牌数的变化趋势。 <i>calhm</i> 子集结果类似	84
5-5 <i>LF-MMI</i> 中，使用不同剪枝技术时 <i>WER</i> 随平均活跃令牌数的变化趋势	86

表格索引

2-1 知名的开源解码器	24
4-1 <i>1-best</i> 和词图性能比较 (<i>beam=14</i>).	62
4-2 所提出系统的速度比较 (<i>beam=14</i>).	63
5-1 <i>LSD versus FSD in DSM</i>	81
5-2 <i>LSD</i> 与跳帧方法的对比	83
5-3 <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	85
5-4 生成式序列模型中的 <i>blank</i> 粒度	85
5-5 生成式序列模型中的 <i>blank</i> 拓扑结构	87
5-6 各个模块的性能比较	87
5-7 针对是否包含模块化训练的性能比较	88
5-8 是否包含 <i>PSD</i> 情况下的模型性能和训练速度	89

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, $\mathbf{O} = [\mathbf{o}_1, \dots, \mathbf{o}_T]$
\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
Σ^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

TODO

英文缩写

AM	声学模型
ASR	自动语音识别
CAT	基类自适应训练

CE	交叉熵
CMN	倒谱均值归一化
CVN	倒谱方差归一化
DNN	深度神经网络
FMLLR	特征最大似然线性回归
GMM	混合高斯模型
HMM	隐马尔可夫模型
LM	语言模型
LSTM	长短时记忆网络
MAP	最大后验估计
MSE	均方差
SD	说话人相关模型
SGD	随机剃度下降
SI	说话人无关模型
WER	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的方式。目前的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的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一个环节。它的功能就是将人的语音转换为相应的文本或者指令以用于后续的处理。

1.1.1 语音识别简史

最早的语音识别系统出现在 1952 年的贝尔实验室 [1]，这是一个只能进行孤立的数字识别的系统。它没有使用通用计算机和任何统计机器学习的方法，只是一个精心设计的端到端的电路。现代的语音识别的基础开始于 70 年代，代表是隐马尔可夫 (Hidden Markov Model, HMM) 模型的提出 [2, 3]。在这个模型中，发声的过程被描述为一个随机生成过程，观测特征的生成过程可以被两个条件概率所描述：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。在接下来的几十年里，随着混合高斯模型 (Gaussian mixture model, GMM) 被用于建模状态输出概率以及 GMM-HMM 理论的不断发展和计算资源的不断提高，语音识别得到了显著的发展。语音识别的任务也从简单的孤立词识别任务进展到大词汇连续语音识别任务。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的评估。这些评估大大推进了语音识别研究的发展并为语音识别设立了若干里程碑。语音识别的词汇量从 1988 年资源管理任务的 900 个词提高到 1993 年华尔街日报任务的 20000 词，达到了真正意义上的大词汇连续语音识别。不仅仅是词汇量的增加，语音识别的任务也向着更现实的识别任务发展，比如录音环境从干净环境变成噪声环境，录音工具从专门的录音设备变成一般的电话语音。随着录音环境越来越复杂，优化的目标也从孤立的单词变为连续的词序列预测。在 90 年代末期，在 HMM 的基础上，研究者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] 来使用序列级的准则进行模型优化。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。苹果手机的第一代 Siri 中使用的就是这些技术。

最早在 20 世纪 90 年代 [24, 25, 26]，已经有研究者提出了使用神经网络 (Neural Network, NN) 来对隐马尔可夫模型中的状态输出概率进行建模的方法。然而，因为那个年代计算资源与数据的匮乏，该框架并没有取得比 GMM-HMM 系统更好的性能。随着摩尔定律持续有效，如今的计算机的运算能力比二十年前有了巨大的飞跃。通用计算图形处理器 (General Purpose Graphical Processing Units, GPGPUs) 使得计算机可以更快速的进行并行计算，这使得训练更强大的模型变得可能。借助越来越先进的移动互联网和云计算，现在可以更容易的收集到足够多的训练数据。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 的提出是对 GMM-HMM 框架的一次变革，令语音识别的性能再次获得了巨大的提高，真正走出了实验室的研究层次 [27, 28, 29, 30, 31, 32, 33, 34]。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。

词错误率 (Word Error Rate, WER) 是衡量语音识别系统好坏的重要指标。给定正确标注和语音识别系统的解码结果，词错误率定义为两个词序列之间的编辑距离。它的数值越小越好。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的词错误率变迁。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。最早的任务是干净环境下的阅读语音识别，随着 GMM-HMM 模型的完善达到了和人类一样的识别率，到 90 年代，任务逐渐向越来越复杂的真实环境靠近。仅仅使用 GMM-HMM 模型，词错误依然很高。其中一个标志性的任务是 Switchboard，该任务为电话语音识别。语音识别任务的两个最大难点在于，第一，语言信号是一个高度非线性的信号；第二，声学环境（说话人，噪音等）对语音信号会有很大影响。从图中可以看到，越往右边的任务场景越开放，难度也越大。在 21 世纪初，学者们研究了鉴别性训练和自适应方法分别用于解决这两个难题，可以看到 Switchboard 的词错率有了很大下降。在 2010 年，微软学者提出了使用深度神经网络来对语音信号进行建模，神经网络的高度非线性非常契合语音信号，Switchboard 的词错误率得到了进一步下降。因为 DNN-HMM 框架是对 GMM-HMM 框架的一个变革，而序列鉴别性训练与自适应技术是与框架独立的两个技术方向。因此，它们在新框架下有了发展的新空间。最近，基于深度神经网络的序列鉴别性训练与自适应技术也成为了新时代语音识别领域最核心的研究内容。本论文主要关注于基于深度神经网络的自适应技术研究。

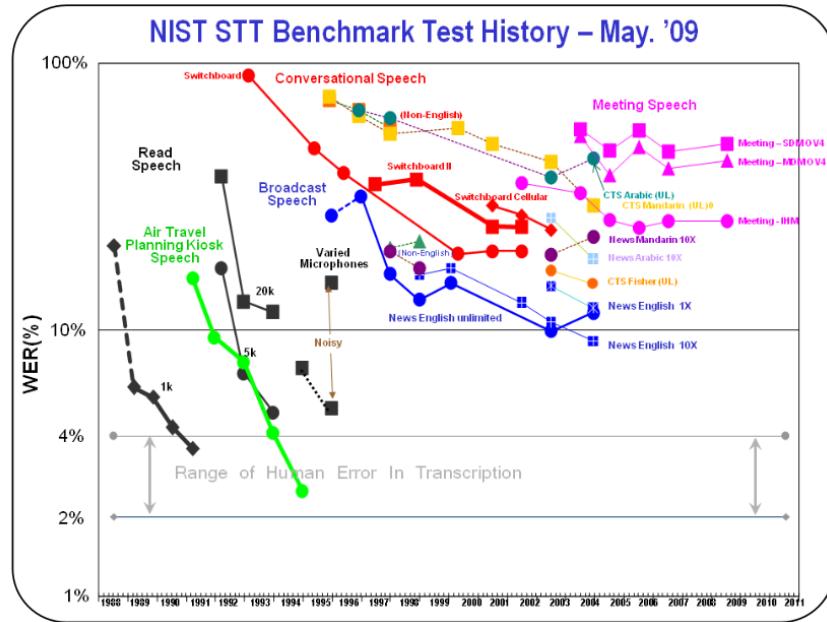


图 1-1 语音识别词错误率变迁图 (截止 2009 年)

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

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的框架中，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假设 \mathcal{H} 中寻找拥有最大后验概率 $P(\mathbf{w} | \mathbf{O})$ 的词序列 \mathbf{w}^* 。其中 $\mathbf{w} = [w_1, \dots, w_n]$ 是词序列， $\mathbf{O} = [\mathbf{o}_1, \dots, \mathbf{o}_T]$ 是特征向量序列。

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

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的词无

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。声学模型用来建模子词单元生成特征序列的概率，语言模型描述的是局部的语法和整句话的语义信息。

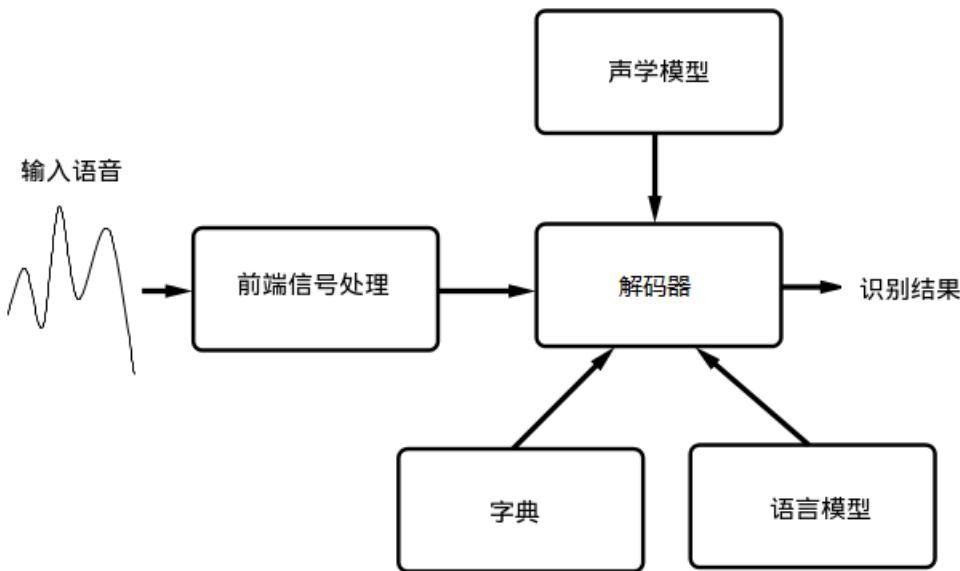


图 1-2 语音识别框架

Fig 1-2 Framework of an automatic speech recognition system

图 1-2 是对当前流行的语音识别系统的框架的描述，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：原始模拟信号首先经录入器件转化为数字信号。前端信号处理部分负责从数字化后的语音中提取鲁棒的声学特征信息，主要包括多麦克风阵列降噪和提取符合人耳听觉感知的声学特征等。详细内容将在章节 2.1.1 中介绍。
- 声学模型 (Acoustic Model, AM)：声学模型是语音识别系统中最核心的模型之一。声学模型的好坏直接决定了语音识别系统的性能，也是本论文的研究重点之一。声学模型建模的是给定的词序列生成出所观测到的特征向量序列的条件概率 $p(\mathbf{O}|\mathbf{w})$ ，目前主流的语音识别系统通常使用隐马尔可夫模型 (Hidden Markov Model, HMM) 来做为声学模型。在 HMM 中，存在一个概率分布被称为状态输出概率，这个概率可以通过使用混合高斯模型来建模，也可以通过深度神经网络来建模。使用前者的语音识别系统被称为 GMM-HMM 系统，使用后者的被称为 DNN-HMM 系统。具体内容将在章节 2.1.2.1 和章节 3.3 中详细介绍。

- 语言模型 (Language Model, LM): 在过去的数十年, N 元组模型 (n-gram) 是最主要使用的语言模型 [35, 36, 37]。近几年, 基于深度神经网络的语言模型也开始得到发展并取得了巨大的性能提升 [38, 39]。
- 解码器及搜索 (Decoder): 解码器的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的语言概率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使用基于动态规划思想的维特比算法 (Viterbi Algorithm), 将在 2.1.4 和 2.2 中详细介绍。

1.2 自动语音识别的推理问题

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。其中一个最主要的难题就是语音识别的推理问题。

语音识别既是一个模式识别问题,也包含相应的推理问题。前一个问题对各种语音、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,这决定了语音识别系统可达到的识别精度的上限。而后一个问题在给定模型的情况下,研究如何将输入语音和模型相匹配,推理得到最优识别结果,这决定了识别速度和实际可达的识别精度。在语音识别的推理阶段,解码器的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。在语音识别推理阶段,解码器是语音识别系统的核心和灵魂,所有的信息都汇集于此。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识别结果。因此,如何将各种性质相异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问题。从解码器的作用来看,它既是语音识别研究中验证各种理论、模型、算法的正确性的基本实验平台,也是构建实用系统的基础。因此,在解码器的设计中也需要兼顾研究的方便与工程实际应用。

根据上一章节的讨论,一个完整的语音识别系统包含了自底向上的五层映射关系:语音观测到 HMM 状态、HMM 状态到上下文相关音素、上下文相关音素到音素、音素到词、词到句子。语音观测无法在识别之前得知,因此语音观测到 HMM 状态的映射需要在解码过程中动态建立,对应的声学分数也需要实时计算。对于 HMM 状态到上下文相关音素、上下文相关音素到音素、音素到词这三层映射关系,一旦发音字典、声学模型确定便不再改变。出于对解码效率的追求,人们通常把它们静态地编译到解码网络中去。对于没有任何约束的大词汇连续语音识别任务而言,词可以以任意方式组织成句,故而从原理上讲,词、句间的映射关系只能在解码过程中动态建立。但是由于表征词与词之间关联度(概率)的 N 元文法模型在解码前便已存在且是一个有限集合,所以在实

际中，语言模型分数的计算却可以用不同的方式实现。在语音识别领域，通常根据解码器中语言模型的表示、语言模型状态的获取以及语言模型分数计算方法的不同把解码器分为两大流派：

- 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，不含有语言模型的任何信息。语言模型状态在解码过程中随着词与词相连成句而动态地生成、语言模型分数通过查表的方式动态获取。这类解码器的典型代表是基于发音前缀树（Pronunciation Prefix Tree，PPT）[40] 网络的解码器。
- 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。语言模型状态以及状态转移以有限状态机的形式合成进解码网络中去，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的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（WFST）[41] 的解码器。

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，语音识别的推理问题未有本质改变。针对推理问题，基于加权有限状态机（WFST）的静态搜索空间构建技术[41] 和帧同步的维特比（Viterbi）网络搜索算法[42] 是目前性能最好的解决方案，但其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，针对新型声学和语言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. 目前的语音识别系统基于多知识源建模结果，对输入音频进行推理，其建模和推理过程都很复杂，且针对知识源的划分依赖很强的先验知识。海量标注或非标注

语音数据的收集，以及基于并行计算的深度学习技术，使得构建直接对语音数据和文本序列进行建模的端到端模型，及其相应的识别推理算法，成为另一种可能。当前的解码框架未有这方面设计。

因此虽然基于深度学习模型，加权有限状态机，和帧同步的维特比网络搜索算法的基于深度学习的语音识别方案已经使其发展到基本可用的程度，但其准确度依然无法满足人类之间正常交流的要求，而速度上的限制也使其无法工作在廉价和低功耗解决方案上，这些原因共同阻碍了语音识别技术的大规模商用。

1.3 论文主要内容、创新点及组织结构

本论文围绕基于深度序列模型的解码搜索技术展开了一系列探索和研究，主要涉及了基于 GPU 并行计算的搜索速度优化，基于标签同步解码的搜索空间优化，基于标签同步解码的统一解码框架，关键词检测的序列建模和标签同步解码等内容。

TODO 每一点做概述和总结创新点，可参考摘要和开题报告表

本文剩余章节安排如下：首先，TODO

第二章 自动语音识别

2.1 自动语音识别框架

这一章将主要介绍大词汇连续语音识别中的一些基本内容，图 1-2 中所描绘的几个基本组件都会在本章描述。主要包括：前端特征提取、隐马尔可夫模型、子词单元的挑选、语言模型、搜索解码等内容。

2.1.1 特征提取

语音信号最原始的形态是一个连续的语音波形，为了能更有效的进行识别，通常我们会先将连续的波形转换为一个离散的实数向量序列 $\mathbf{O} = [\mathbf{o}_1, \dots, \mathbf{o}_T]$ 。每一个向量都是压缩后的语音变化的一种表示。这些向量也被称为特征向量或者观测特征向量。语音信号是一个准平稳信号，所以我们首先需将其切分成若干重叠的离散片段，通常是将一个 25 毫秒长的窗口以 10 毫秒的间隔向后滑动，通过此方法提取出的一个片段被称为一帧。为减小边界效应，通常会使用汉明 (Hamming) 窗或者汉宁 (Hanning) 窗来进行平滑，继而使用快速傅立叶变换将其从时域特征转变为频域特征。在得到频域上的复数特征后，通过使用不同的后处理方法可以得到不同特征：典型的有梅尔倒谱系数 (Mel-Frequency Cepstral Coefficients, MFCC) [43] 和感知线性预测系数 (Perceptual Linear Prediction, PLP) [44]。最近，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) [45] 更适合于深度神经网络使用，接下来将对这三种特征进行详细介绍：

- 滤波器组特征

1. 在得到了频率谱的复数特征后，仅仅只有复数特征的幅度部分会留下，而相位信息通常会被丢弃，接着频率轴会通过梅尔频率缩放公式进行调整，最终我们可以得到一个缩放后的幅频域特征：

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

2. 接着，一组三角滤波器将被用于降采样这一幅频域特征，不同的滤波器含有不同的滤波增益。最终一个滤波器的输出为幅度特征乘上该滤波器中对应频率增益的求和自然对数。通常而言，对于 8K 采样率的语音会使用 36 组滤波器，对于 16K 的语音会使用 40 组。

- 梅尔倒谱系数

梅尔倒谱系数特征在 FBANK 特征的基础上进一步使用了离散余弦变换来计算倒谱系数以减少滤波器组之间的相关性。通常使用 12 个倒谱系数加上归一化后功率的自然对数组成一个 13 维的特征向量。

- 感知线性预测系数

1. 感知线性预测系数是另外一种倒谱特征，它使用 Bark 公式来缩放频率轴：

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

2. 接着它使用功率谱（幅度的平方）来提取感知线性特征，之后该功率谱会与一个临界频带滤波器进行卷积并且通过等响度曲线进行预加重。
3. 最后通过使用线性预测分析来获得倒谱系数。

在提取了原始的声学特征之后，通常会使用一些后处理方法：

- 动态特征 [46]: 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 $\Delta_{\mathbf{o}_t}$ 。在使用了动态特征后，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做出的特征各维度之间的独立性假设产生了冲突。因此，为消除特征各维度之间的相关性，通常会使用线性投影的方法，如异方差线性判别分析 (Heteroscedastic Linear Discriminant Analysis, HLDA) [47] 等。

- 特征正则化：特征正则化的目标是消除声学特征中的非语音变化，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) [48]，倒谱方差归一化 (Cepstral variance normalisation, CVN)[49] 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的特征频率轴进行同样大小的缩放。

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} 的概率。它从概率论角度提供了给定词标注之后语音信号的生成过程。在传统的 GMM-HMM 声学模型中，HMM 建模了语音的序列性，GMM 建模了特征向量的生成概率。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的生成概率。GMM-HMM 将在章节2.1.2.1中详细介绍，DNN-HMM 将在章节3.3中详细介绍。声学模型是语音识别系统中最核心的部件之一，也是本论文的研究重点。

2.1.2.1 隐马尔可夫模型 (HMM)

隐马尔可夫模型是一个统计学中的生成模型，在语音识别领域获得了重大成功。在隐马尔可夫模型中，一个特定的声学单元，如一个单词或一个音素将被建模为一个有限状态机，而我们所观测到的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。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的隐马尔可夫模型如图2-1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。这一结构是语音识别中最常用的结构。

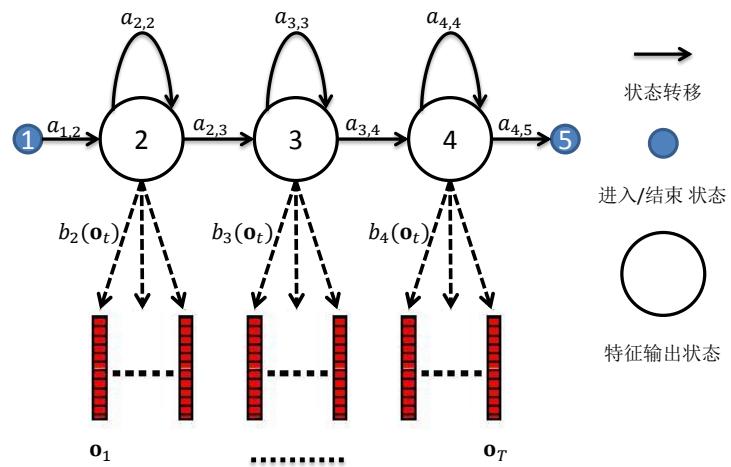


图 2-1 隐马尔可夫模型

Fig 2-1 Hidden Markov Model

令 $\mathbf{O} = [\mathbf{o}_1, \dots, \mathbf{o}_T]$ 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 是语音序列的总帧数。声学特征序列的生成过程如下所示：

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

这样我们可以使用一个状态序列来描述语音特征的输出过程 $\mathbf{s} = [s_1, \dots, s_T]$ 。而现实中我们只能观察到由状态输出的语音特征序列, 状态序列 \mathbf{s} 是隐藏的, 这也是该模型被称为隐马尔可夫模型的原因。一个隐马尔可夫模型通常包含如下的参数:

- π 初始状态分布:
令 s_t 表示在时刻 t 时所处的状态, 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。在拥有进入状态的隐马尔可夫模型中, 通常 $\pi_1 = 1$ 。
- 状态转移概率矩阵 \mathbf{A} :
 $a_{i,j} = P(s_{t+1} = j | s_t = i)$, 在最常用的 5 状态隐马尔可夫模型中, 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, $b_i(\mathbf{o}_t) = p(\mathbf{o}_t | s_t = i)$

2.1.2.2 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:

$$b_i(\mathbf{o}_t) = \sum_{m=1}^{M_i} c_i^m \mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_i^m, \boldsymbol{\Sigma}_i^m) \quad (2-4)$$

其中 M_i 是属于第 i 个状态的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 $\boldsymbol{\Sigma}_i^m$ 的多变量高斯分布。

$$\mathcal{N}(\mathbf{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{o}-\boldsymbol{\mu})} \quad (2-5)$$

2.1.2.3 HMM 的似然计算

在定义好了 HMM 中的分布之后，我们就可以计算隐马尔可夫模型的似然。似然计算的目标是给定一个语音特征向量序列和一个隐马尔可夫模型，计算该模型生成给定的语音特征向量序列的概率，即计算 $p(\mathbf{O}|\mathbf{w}, \mathcal{M})$ ，其中 \mathbf{O} 为观测到的语音特征向量序列， \mathbf{w} 为对应的文本标注， $\mathcal{M} = \{\pi, \mathbf{A}, \mathbf{B}\}$ 是所有的模型参数。由于状态序列 \mathbf{s} 是隐藏的，因而需要枚举所有可能的状态序列并求其期望，即：

$$p(\mathbf{O}|\mathbf{w}, \mathcal{M}) = \sum_{\mathbf{s}} p(\mathbf{O}, \mathbf{s}|\mathbf{w}, \mathcal{M}) \quad (2-6)$$

$$= \sum_{\mathbf{s}} P(\mathbf{s}|\mathbf{w}, \mathcal{M}) p(\mathbf{O}|\mathbf{s}, \mathcal{M}) \quad (2-7)$$

$$= \sum_{\mathbf{s}} a_{s_0, s_1} \prod_{t=1}^T a_{s_{t-1}, s_t} b_{s_t}(\mathbf{o}_t) \quad (2-8)$$

以上只考虑了单个隐马尔可夫模型的似然计算。对于连续的语音识别或使用子单词的声学单元，语音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的时间边界是未知的，而解决方法则是扩展单隐马尔可夫模型，将若干个单独的隐马尔可夫模型连接起来组成组合隐马尔可夫模型。

似然计算是使用和训练隐马尔可夫模型时需要考虑的最核心问题，直接枚举所有状态序列的复杂度无疑是很高的。这一概率可以使用前向-后向算法快速计算，该算法也被称作鲍姆威尔士算法 [50](Baum-Welsh algorithm)。前向-后向算法通过使用动态规划的思想，只需要 $O(N^2T)$ 的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。

定义前向概率 $\alpha_i(t)$ 为到 t 时刻为止，且第 t 时刻所处状态为 i ，观测到特征序列 $(\mathbf{o}_1, \dots, \mathbf{o}_t)$ 的概率。

$$\alpha_i(t) = p(\mathbf{o}_1, \dots, \mathbf{o}_t, s_t = i | \mathbf{w}, \mathcal{M}) \quad (2-9)$$

前向概率可以通过递归快速计算，对于 $1 < i < N, 0 < t \leq T$

$$\alpha_i(t) = \left(\sum_{j=1}^{N-1} \alpha_j(t-1) a_{j,i} \right) b_i(\mathbf{o}_t) \quad (2-10)$$

边界条件为：

$$\alpha_i(t) = \begin{cases} 1 & i = 1, t = 0 \\ 0 & i \neq 1, t = 0 \\ \sum_{j=2}^{N-1} \alpha_j(T) a_{j,N} & i = N, t = T + 1 \end{cases} \quad (2-11)$$

对于常用于语音识别的五状态 HMM 而言，因为转移只存在于相邻的两个状态之间，所以时间复杂度减少为 $O(NT)$ 。

同样我们可以定义后向概率 $\beta_i(t)$ 为从第 t 时刻开始，且第 t 时刻所处状态为 i 的概率，观测到特征序列 $(\mathbf{o}_{t+1}, \dots, \mathbf{o}_T)$ 的概率。

$$\beta_i(t) = p(\mathbf{o}_{t+1}, \dots, \mathbf{o}_T | s_t = i, \mathbf{w}, \mathcal{M}) \quad (2-12)$$

后向概率也可以通过递归快速计算，对于 $1 < i < N, 0 < t < T$

$$\beta_i(t) = \sum_{j=1}^N a_{i,j} b_j(\mathbf{o}_{t+1}) \beta_j(t+1) \quad (2-13)$$

边界条件为：

$$\beta_i(t) = \begin{cases} a_{i,N} & t = T \\ \sum_{j=2}^{N-1} a_{i,j} b_j(\mathbf{o}_1) \beta_j(1) & i = 1, t = 0 \end{cases} \quad (2-14)$$

计算完前向概率和后向概率之后，可以很容易的得到似然的公式为：

$$p(\mathbf{O} | \mathbf{w}, \mathcal{M}) = \alpha_N(T+1) = \beta_1(0) \quad (2-15)$$

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) 的准则。令 $\mathcal{M} = \{\{a_{i,j}, 1 \leq i, j \leq N\}, \{c^m, \mu^m, \Sigma^m, 1 \leq m \leq M\}\}$ 为 GMM-HMM 中的所有参数。其中 N, M 分别为总状态数和总高斯成分数。优化的准则即为：

$$\hat{\mathcal{M}}_{\text{MLE}} = \arg \max_{\mathcal{M}} \log p(\mathbf{O} | \mathbf{w}, \mathcal{M}) \quad (2-16)$$

由于存在隐藏变量，直接优化上述公式是很困难的，使用最大期望算法 (Expectation-Maximization, EM) [51] 可对此进行优化。EM 算法被广泛运用于含有隐变量的统计学模型中，它的基本思想是引入一个辅助函数作为 \log 似然的下界，通过不断迭代优化辅助函数来优化 \log 似然函数。对于隐马尔可夫模型而言，辅助函数定义为：

$$\mathcal{Q}_{\text{MLE}}(\mathcal{M}_{k+1}; \hat{\mathcal{M}}_k) = \sum_{\mathbf{s}} P(\mathbf{s} | \mathbf{O}, \mathbf{w}, \hat{\mathcal{M}}_k) \log p(\mathbf{O}, \mathbf{s} | \mathbf{w}, \mathcal{M}_{k+1}) \quad (2-17)$$

$$= \sum_{t,i} \gamma_i(t) \log b_i(\mathbf{o}_t) + \sum_{t,i,j} \xi_{ij} \log a_{i,j} \quad (2-18)$$

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

$$\gamma_i(t) = P(s_t = i | \mathbf{O}, \mathbf{w}, \hat{\mathcal{M}}_k) \quad (2-19)$$

$$\xi_{ij}(t) = P(s_{t-1} = i, s_t = j | \mathbf{O}, \mathbf{w}, \hat{\mathcal{M}}_k) \quad (2-20)$$

EM 算法是一个迭代的优化过程，其优化的步骤如下：

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. 使用 $\hat{\mathcal{M}}_k$ 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的 α, β 快速获得：

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{p(\mathbf{O}|\mathbf{w}, \hat{\mathcal{M}}_k)} \quad (2-21)$$

$$\xi_{ij}(t) = \frac{\alpha_i(t-1)a_{i,j}b_j(\mathbf{o}_t)\beta_j(t)}{p(\mathbf{O}|\mathbf{w}, \hat{\mathcal{M}}_k)} \quad (2-22)$$

4. 使用最大似然估计 $\hat{\mathcal{M}}_{k+1}$

$$\hat{\mathcal{M}}_{k+1} = \arg \max_{\mathcal{M}} \sum_{t,i} \gamma_i(t) \log b_i(\mathbf{o}_t) + \sum_{t,i,j} \xi_{ij} \log a_{i,j} \quad (2-23)$$

5. 转移概率的更新为：

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^T \xi_{i,j}(t)}{\sum_{t=0}^{T+1} \gamma_i(t)} \quad (2-24)$$

6. 当使用 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的隐子状态，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的后验占有量：

$$\gamma_j^m(t) = \frac{\sum_{i=2}^{N-1} \alpha_i(t-1) a_{i,j} c_j^m \mathcal{N}(\mathbf{o}; \boldsymbol{\mu}_j^m, \boldsymbol{\Sigma}_j^m) \beta_j(t)}{p(\mathbf{O}|\mathbf{w}, \hat{\mathcal{M}}_k)} \quad (2-25)$$

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

7. GMM 的参数更新为：

$$\hat{c}_j^m = \frac{\sum_{t=1}^T \gamma_j^m(t)}{\sum_{m,t} \gamma_j^m(t)} \quad (2-26)$$

$$\hat{\boldsymbol{\mu}}_j^m = \frac{\sum_{t=1}^T \gamma_j^m(t) \mathbf{o}_t}{\sum_{t=1}^T \gamma_j^m(t)} \quad (2-27)$$

$$\hat{\boldsymbol{\Sigma}}_j^m = \text{diag} \left(\frac{\sum_{t=1}^T \gamma_j^m(t) (\mathbf{o}_t - \hat{\boldsymbol{\mu}}_j^m)(\mathbf{o}_t - \hat{\boldsymbol{\mu}}_j^m)^\top}{\sum_{t=1}^T \gamma_j^m(t)} \right) \quad (2-28)$$

在公式中，我们只估计了协方差矩阵的对角元素。由于在大词汇连续语音识别任务中通常需要使用大量的高斯成分，在此基础上若使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，通常只使用对角矩阵。

8. 重复 2 直到收敛

虽然使用最大似然估计 **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的前提下它才是合适的优化准则。现实中，由于在 **HMM** 模型中存在马尔可夫和条件独立性两个假设，并不符合真正的语音生成过程，因此使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使用序列鉴别性的训练准则 [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，优化的目标是给定标注生成语音特征向量序列的似然。鉴别性训练与之不同处在于：鉴别性训练优化的目标为最大化给定语音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识别的评判准则引入优化目标之中。目前最先进的语音识别系统中都使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 $P(\mathbf{w}_{\text{ref}}|\mathbf{O})$ 的基础上增加一个经验缩放 κ^1 ，它的优化目标如下：

$$\mathcal{F}_{\text{MMI}}(\mathcal{M}) = \frac{p^\kappa(\mathbf{O}|\mathbf{w}_{\text{ref}}, \mathcal{M})P(\mathbf{w}_{\text{ref}})}{\sum_{\mathbf{w}} p^\kappa(\mathbf{O}|\mathbf{w}, \mathcal{M})P(\mathbf{w})} \quad (2-29)$$

这里 \mathbf{w}_{ref} 是语音特征向量序列对应的标注， \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} 应该包括所有可能的词序列，实际上通常只考虑最具有混淆性的标注。它通常由在训练数据上解码生成的 **N-Best** 列表或者词图构成。

- 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

$$\mathcal{F}_{\text{MBR}}(\mathcal{M}) = \sum_{\mathbf{w}} P(\mathbf{w}|\mathbf{O}; \mathcal{M})L(\mathbf{w}, \mathbf{w}_{\text{ref}}) \quad (2-30)$$

¹它的作用是为了使不太可能的假设对准则有所贡献，并使准则更加平滑可区分，通常等于识别中使用的语言模型缩放系数的倒数

这里 $L(\mathbf{w}, \mathbf{w}_{\text{ref}})$ 是标注与候选的标注之间的损失函数，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

$$L(\mathbf{w}, \mathbf{w}_{\text{ref}}) = \begin{cases} 1 & \mathbf{w} \neq \mathbf{w}_{\text{ref}} \\ 0 & \mathbf{w} = \mathbf{w}_{\text{ref}} \end{cases} \quad (2-31)$$

- 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， $L(\mathbf{w}, \mathbf{w}_{\text{ref}})$ 为两个词序列之间的编辑距离，即词错误数。最小音素错误在最先进的语音识别系统中使用非常广泛 [22]

2.1.2.6 声学单元和参数绑定

当进行小词汇语音识别时，如识别数字时通常可以用隐马尔可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，使用隐马尔可夫模型来建模每一个词汇变成了不可能的事情。一个广泛使用的解决该问题的方法为建模子词单元。

音素是一个被广泛使用的子词单元，它是语音中最小的声学元素。使用音素的好处是存在将每个词转换为音素的标准，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，音素的数目一般远远小于待识别的词的个数。在当前最先进的大词汇连续语音识别系统里，通常使用 46 个音素。

使用音素能更容易的获得足够的数据以训练出更具有鲁棒性的模型参数。需要注意的是当使用音素来建模时，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。在识别的最后，还需要将子词单元序列转换回单词序列。

目前有两种被广泛使用的音素集合。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，当前音素的发音和前一个、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，仅仅使用单音素效果不是很好。为了建模协同发音现象，目前最先进的语音识别系统使用的是上下文相关音素，其中三音速使用最为广泛，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，它对应的三音素序列即扩展为 $\text{one}=\{\text{sil-w-ah, w-ah-n, ah-n-sil}\}$ 。虽然使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使用最广泛的音素模型。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：在两个单词的边界，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的第一个音素。词内三音素则相反，音素只能在单词的内部进行扩展。因此，在每个单词的开头和结尾，是一个双音素。跨单词三音素在大规模词汇连续语音识别系统里效果更好 [40]。

使用三音素后，声学单元个数将变得巨大。当使用 46 个单音素时，所有可能的三音素的个数达到了 $46^3 = 97336$ ，这一数字甚至超过了词表的大小，不可能有充足的数据能训练如此大的模型。目前一个通用的解决该问题的方法是使用参数共享 [53, 54]。它的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。由于使用最广泛的是在状态共享层面，因而也被称为状态聚类。在使用状态聚类时，来自同一组的状态将共享状态输出分布，每一个实际的状态输出分布被称为一个语素 (senone)，如图 2-2 所示。

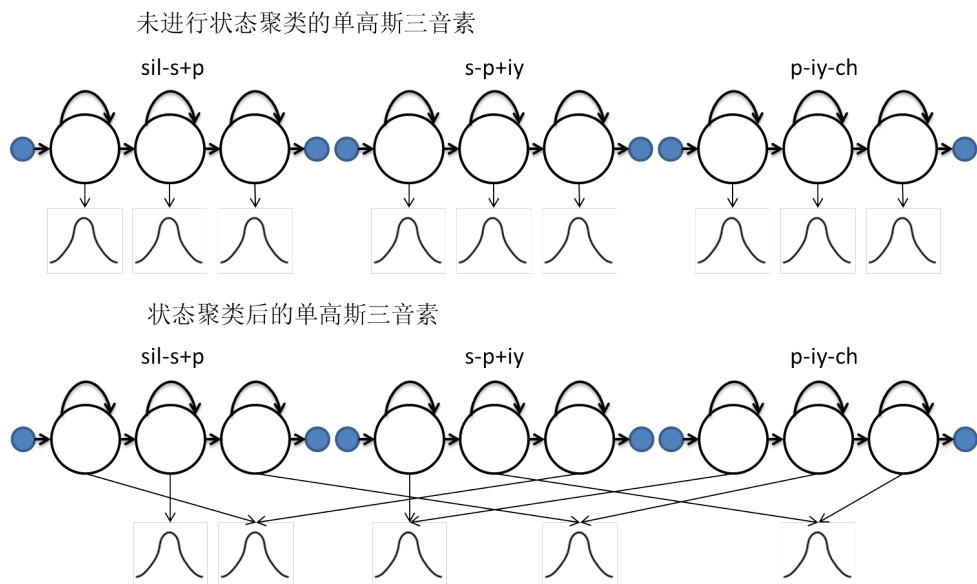


图 2-2 单高斯三音素的状态聚类
Fig 2-2 State clustering for single gaussian triphones

目前使用最广泛的聚类方法是基于数据驱动的自底向上的聚类。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的训练数据时，这一方法将不再可靠，更严重的是，训练数据中不存在的信息将无法被捕捉到。

另外一个更好的方法是使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的左右上下文问题的二叉树，因为问题的答案只有对和错。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的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，目前最常用的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，但它确实有效的处理了对于不可见的三音素的分类问题。

2.1.3 语言模型

语音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，语音模型概率可以扩展为若干条件概率的连乘：

$$P(\mathbf{w}) = \prod_{k=1}^K P(w_k | w_{k-1}, \dots, w_1) \quad (2-32)$$

这里 w_k 是序列中的第 k 个单词。使用公式2-32来计算先验概率需要记录整个句子的历史，然而在大词汇语音识别任务中，由于词表的大小往往能达到 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的估计。一种解决方案是限制历史的长度，N 元组 (n-gram) 语音模型即使用了这一策略，它也是目前语音识别中最广泛使用的统计语言模型。它所作出的假设是只需要最多使用 N 个单词作为历史就足够计算概率：

$$P(w_k | w_{k-1}, \dots, w_1) \approx P(w_k | w_{k-N+1}) \quad (2-33)$$

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

$$P(w_k | w_{k-1}, \dots, w_{k-N+1}) = \frac{f(w_k, w_{k-1}, \dots, w_{k-N+1})}{\sum_w f(w, w_{k-1}, \dots, w_{k-N+1})} \quad (2-34)$$

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识别任务中依然很难达成。所以，过去的研究提出了若干平滑的方法来获得鲁棒的估计。

- 降权 (Discounting)

降权的方法用于解决训练集中未观测到的 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 元组的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的 N 元组。最常用的降权方法有 Good-Turing 法 [35, 36]，Witten-Bell 法 [55] 和绝对降权法 [56]

- 回溯法 (Backing-off)

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

- 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的模型，比如在语音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的语言模型。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的 N 元语言模型进行插值。

语言模型的训练和评价指标通常是困惑度 (Perplexity, PPL)，它的定义是词序列生成概率的几何平均的倒数：

$$PPL = 2^{-\frac{1}{K} \log(P(\mathbf{w}))} \quad (2-35)$$

其中 K 是词序列包含的总词数，拥有更低 PPL 的语言模型具有更低的不确定度和混淆度，通常也能潜在的降低语音识别的词错误率。当然，这一关系并非一直成立，因而在语音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

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的路径。

$$\mathbf{w}^* = \arg \max_{\mathbf{w} \in \mathcal{H}} p(\mathbf{O}|\mathbf{w})P(\mathbf{w}) \quad (2-36)$$

$$= \arg \max_{\mathbf{w} \in \mathcal{H}} \sum_{\mathbf{s}} P(\mathbf{s}|\mathbf{w})p(\mathbf{O}|\mathbf{s})P(\mathbf{w}) \quad (2-37)$$

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2.1.2.3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，因为候选的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，实际中通常使用最大概率的状态路径来近似整个概率，即：

$$\mathbf{w}^* = \arg \max_{\mathbf{w} \in \mathcal{H}} \max_{\mathbf{s}} P(\mathbf{s}|\mathbf{w})p(\mathbf{O}|\mathbf{s})P(\mathbf{w}) \quad (2-38)$$

这样我们可以通过使用动态规划 (Dynamic Programming, DP) 的方法计算出拥有最大概率的状态序列，然后以此推理出词序列，这一算法被称为 Viterbi 算法 [57]。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。它可以使用递归来计算：

$$\phi_i(t) = \max_j \{\phi_j(t-1)a_{j,i}\} b_i(\mathbf{o}_t) \quad (2-39)$$

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：

$$p(\mathbf{O}|\mathbf{w}) \approx \phi_N(T+1) = \max_i \{\phi_i(T)a_{i,N}\} \quad (2-40)$$

Viterbi 算法可以很容易的扩展到连续语音识别，一种扩展后的 Viterbi 算法也被称为令牌传递 (Token Passing) 算法 [58]。在这一算法中，每一个状态将不仅仅保留一个最优路径，而是保存若干个令牌。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。在到达一个词的边界时，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。

即使使用了令牌传递算法，在大词汇连续语音识别中，传播所有令牌的搜索代价依然十分庞大。为了减少计算代价，通常会使用基于集束搜索 (beam search) 的剪枝方法。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 搜索的方法可以显著的减少计算量，但是有可能在早期阶段使用了过窄的 beam 宽度而剪去了真正的最优路径导致识别错误。所以，beam 宽度的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的问题，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的比例，因而最终语音识别中使用的准则是：

$$\mathbf{w}^* = \arg \max_{\mathbf{w} \in \mathcal{H}} \{\log p(\mathbf{O}|\mathbf{w}) + \alpha \log P(\mathbf{w}) + \beta L_{\mathbf{w}}\} \quad (2-41)$$

这里 α 是语言模型分数缩放系数， β 是插入错误惩罚系数， $L_{\mathbf{w}}$ 是词序列的长度。通常会选择最大概率的路径作为识别结果，也可以输出若干候选结果然后用更强的语言模型来重新评价（比如使用神经网络语言模型）。这一候选结果一般使用 N-best 列表 [59] 或者使用能包含更多信息的词图 (Word Graph/Word Lattice) [60]。近些年来，基于加权有限状态机 (Weighted Finite-State Transducer, WFST) [41] 的解码器被越来越多的人采用，它的优点是可以将语言模型提前融合成一个网络，从而在解码时加快解码速度。

2.2 大词汇连续语音识别中的解码搜索技术

语音识别既是一个模式识别问题，也包含相应的推理问题。前一个问题对各种语音、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

定了语音识别系统可达到的识别精度的上限。而后一个问题在给定模型的情况下，研究如何将输入语音和模型相匹配，推理得到最优识别结果，这决定了识别速度和实际可达的识别精度。在语音识别的推理阶段，解码器的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。在语音识别推理阶段，解码器是语音识别系统的核心和灵魂，所有的信息都汇集于此。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识别结果。因此，如何将各种性质相异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问题。从解码器的作用来看，它既是语音识别研究中验证各种理论、模型、算法的正确性的基本实验平台，也是构建实用系统的基础。因此，在解码器的设计中也需要兼顾研究的方便与工程实际应用。

2.2.1 解码器的技术流派

根据前面章节的讨论，一个完整的语音识别系统包含了自底向上的五层映射关系：语音观测到 HMM 状态、HMM 状态到上下文相关音素、上下文相关音素到音素、音素到词、词到句子。具体来说公式 2-38 可以进一步展开如下：

$$\begin{aligned} \mathbf{w}^* &= \arg \max_{\mathbf{w} \in \mathcal{H}} \max_{\mathbf{s}} P(\mathbf{s}|\mathbf{w}) p(\mathbf{O}|\mathbf{s}) P(\mathbf{w}) \\ &= \arg \max_{\mathbf{w}} \sum_{\mathbf{l}} \sum_{\mathbf{c}} \sum_{\mathbf{s}} p(\mathbf{O}|\mathbf{s}) \cdot P(\mathbf{s}|\mathbf{c}) \cdot P(\mathbf{c}|\mathbf{l}) \cdot P(\mathbf{l}|\mathbf{w}) \cdot P(\mathbf{w}) \end{aligned} \quad (2-42)$$

其中展开后式子的每一项分别对应上述的五层映射关系。语音观测无法在识别之前得知，因此语音观测到 HMM 状态的映射需要在解码过程中动态建立，对应的声学分数也需要实时计算。对于 HMM 状态到上下文相关音素、上下文相关音素到音素、音素到词这三层映射关系，一旦发音字典、声学模型确定便不再改变。出于对解码效率的追求，人们通常把它们静态地编译到解码网络中去。对于没有任何约束的大词汇连续语音识别任务而言，词可以以任意方式组织成句，故而从原理上讲，词、句间的映射关系只能在解码过程中动态建立。但是由于表征词与词之间关联度（概率）的 N 元文法模型在解码前便已存在且是一个有限集合，所以在实际中，语言模型分数的计算却可以用不同的方式实现。在语音识别领域，通常根据解码器中语言模型的表示、语言模型状态的获取以及语言模型分数计算方法的不同把解码器分为两大流派：

- 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，不含有语言模型的任何信息。语言模型状态在解码过程中随着词与词相连成句而动态地生成、语言模型分数通过查表的方式动态获取。这类解码器的典型代表是基于发音前缀树（Pronunciation Prefix Tree, PPT）[40] 网络的解码器。

- 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。语言模型状态以及状态转移以有限状态机的形式合成进解码网络中去，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的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（WFST）[41]的解码器。

静态网络解码器作为一种以空间换时间的策略，通过精心的优化，一般可以获得更快的识别速度，但是其缺点也是显而易见的，即：构建的解码网络规模过大，尤其是在采用高阶语言模型和声学模型的情况下，由于内存大小的限制，要将所有的知识源都编译进解码网络中往往是不可行的。因此，近年来基于动态解码网络的传统语音识别方法又逐渐受研究人员青睐[61, 62]，能够几乎不受任何约束地使用高阶语言模型和声学模型以便获得更优的识别精度，应用面更广，这是动态网络解码器最为诱人之处。然而动态网络解码器的设计和束搜索剪枝方法更为复杂，需要调整的参数和门限更多，挑战性更大。

除了从解码网络的结构上进行分类，还可以根据搜索最优路径的不同方式，将解码器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若干后进先出的缓冲区（即：堆栈）来保存扩展得到的词识别假设，所以也常常被称为堆栈解码器（Stack decoder）[63]。与时间同步搜索相比，其好处在于：通过选取适当的启发式函数（Heuristic function），可以将搜索局限于最优路径附近，提高搜索效率。但这也造成了它的主要缺陷：1) 启发式函数要综合声学和语言两方面的分数，有时还需要“将来”路径的信息，因此非常难以获得；2) 由于是时间异步搜索，所以搜索过程中产生的路径长度各不相同，使得剪枝难于实现。因此，目前堆栈解码已很少用于直接对语音观测进行解码，而是更多地作为一种后处理手段，用于从词图中生成 N-Best 结果。
- 时间同步搜索解码器：时间同步搜索是目前解码器设计与实现中的主流方法，又称帧同步搜索。它采用广度优先方式从解码网络中找出与输入特征序列最匹配的状态序列，从而得到相应的最优音素序列和词序列。这一类解码器通常采用 Viterbi 算法或是令牌传递（Token passing）算法[40]实现。

解码器不仅算法复杂，并且实现起来需要较高的工程化能力和技巧，在开发完成后也往往需要不小的人力和时间才能够调优，因此各个研究机构与商业公司都对自己解码

器的实现细节讳莫如深。尽管如此，目前世界上还是存在一些以学术研究为目的的开源解码器可以作为研究的基础，表列出了其中较为知名的。这些解码器的网络结构和解码算法各异，但由于是以学术研究为目的，因此速度优化方面大部分较差。

表 2-1 知名的开源解码器

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

多遍搜索与单遍搜索（1-pass）相比各有优劣，在实际中也都有应用。多遍搜索的拥护者认为：通过第一遍粗搜索可以大大缩小解码空间，使得在后续解码过程中使用更加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、空间约束问题。单遍搜索的拥护者认为多遍解码有三个主要问题：1) 由于第二遍解码必须等待第一遍解码完成，所以多遍搜索很难应用于实时解码。2) 每一遍解码都会引入不可恢复的剪枝错误，这些错误不论在后续解码中采用何种模型都无法弥补。要解决这一问题还是得在单遍搜索算法上下功夫，与其这样，还不如直接做好单遍解码器。3) 多遍解码器的每一遍解码采用的特征、声学模型、语言模型、搜索算法都不尽相同。

2.2.2 结构和主要模块

2.2.2.1 解码器框架

图 2-3给出了典型的解码器结构。不论是哪种类型的解码器通常都包含网络生成、分数计算、搜索、剪枝与路径管理这五部分，它们的功能逐一介绍如下。

2.2.2.2 网络生成

网络生成模块主要负责将构建解码搜索空间。搜索空间，又称解码网络，一般以音素 HMM 或 HMM 状态连缀而成，由语言模型、发音字典、声学模型和其他相关知识源编译而来。大词汇连续语音识别系统的解码网络是由各个知识源构成的一个搜索空间，一般来讲可以分为动态构建的解码网络和静态网络。基于动态网络的解码器，以前缀树

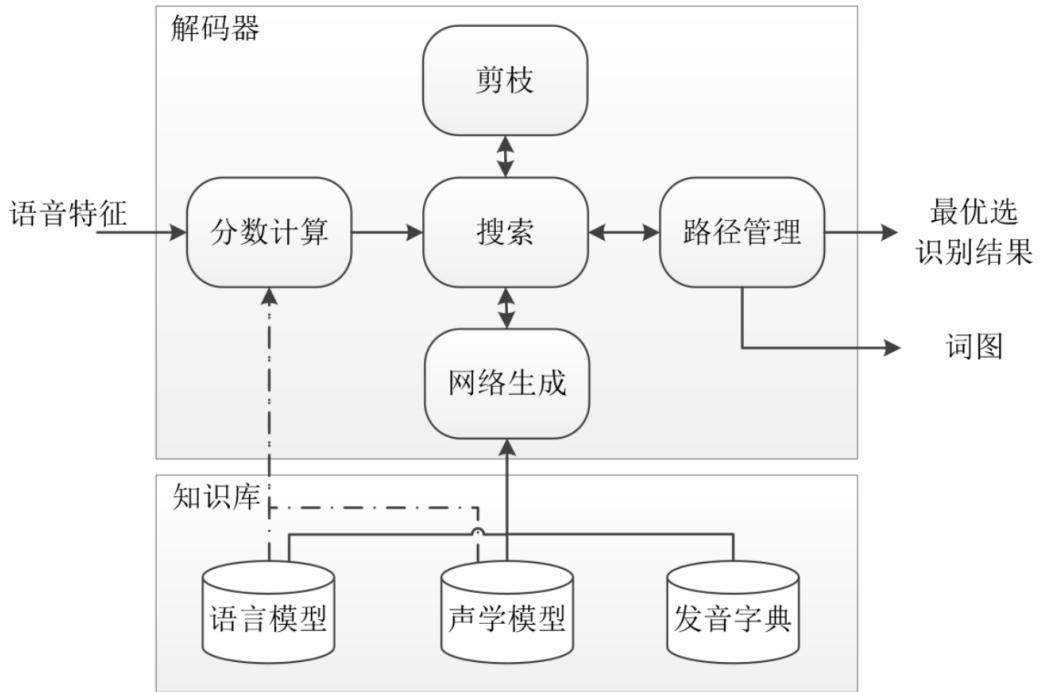


图 2-3 通用解码器架构
Fig 2-3 General Architecture of ASR Decoder

的发音词典作为搜索网络，语言模型则通过动态查询的方式把得分引入解码过程之中，然后利用重入字典树或者字典树拷贝的方式对整个解码网络进行搜索 [58]。动态网络解码器的优势在于，由于字典和语言模型是分离的，其占用内存较少，这个特点在以往移动网络技术和硬件技术不发达的时期，尤其是在嵌入式设备上内置解码器的时代占有绝对优势，这是由嵌入设备的硬件条件决定的。但是动态网络的缺点是它的时间复杂度较高、解码速度较慢，这也使其越来越难以满足当前海量语音识别的需求。随着移动互联网和嵌入式设备的普及以及云技术的发展，语音识别的应用发展为在嵌入式设备上仅仅保留简单的前端，而识别系统保留在服务器云端。这时基于加权有限状态机的静态网络解码器的快速优势就体现出来了，更短的识别时间能让服务器在单位时间内接受更多的识别任务，因此静态编译的解码网络更适合用于海量的语音识别任务。

在 LVCSR 任务中，解码网络通常非常庞大，因此网络生成过程中需采用多种手段对网络结构进行优化，在不改变网络功能的前提下，尽可能地减少网络中节点和边的数目。解码网络是解码器的基础，决定了解码器的其他部分应当采用何种方法实现。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。语言模型状态以及状态转移以有限状态机的形式合成进解码网络中去，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的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（WFST [41]）的解码器。

加权有限状态机最开始是由 AT&T 实验室的 Mohri 和 Riley 等人在 1997 年引入语音识别领域的。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，为语音识别的应用打好了理论基础。语音识别的各模型组件分别用如下方式进行 WFST 的构建：

- N 元语言模型在 WFST 中被表示为 G 。从 N 元语言模型的含义可知，它提供的是当前词对于词历史的概率。语言模型需要有信息记录其至多 N 个词的历史。但是，WFST 应用在语音识别时是不允许输入输出带有串的（即输入输出都是其符号集的单个元素）。因此不能用 G 构造过程的输入输出上表示其历史，而应选用在状态上记录历史。另外一个问题是在 N 元语言模型的回退（Back-off），当语言模型的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权重的空输入的状态跳转边。
- 字典在 WFST 中被表示为 L 。字典其实是对发音规则的一种表示，最简单的方法就是在开始状态和结尾状态之间列出每个词的发音。同时，由于在和语言模型合成的时候，词与词是相连的，不能够到达终止状态就完结了，需要用一个空边从终止状态连接到开始状态形成一个闭环。针对字典的同发音问题，Mohri 在文献 [41] 中提出对于不同词的同发音的发音序列加入一组辅助符号，这样对于同发音的词，其环路的输入符号就不再相同，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 C 。一个上下文相关的 triphone 模型一般表示为 $a-b+c$ ，其中 b 是中心音素， a 和 c 为其前后的上下文音素 [64]。用 WFST 表示 triphone 模型实际上是把三音素和单个的音素进行对应，其输出就和发音字典的输入相互对应，由此可以进行进一步合成。
- 隐马尔可夫模型在 WFST 中被表示为 H 。隐马尔可夫模型天然具有多个状态跳转的特性，因此可以直接将其构建为多个 WFST 的状态。隐马尔科夫模型的转移概率被转化为 WFST 的权重。而 WFST 的输入是隐马尔科夫模型的状态编号，输出是该隐马尔科夫模型所表示的 triphone 建模单元。

当各个知识源的 WFST 组件被构建完毕以后，可以使用 WFST 的合成和优化算法将各组件最后进行合并。如下：

$$HCLG = \min(\det(H \circ \min(\det(C \circ \min(\det(L \circ G)))))) \quad (2-43)$$

最终生成的 WFST 包含了所有的知识源，后文所讨论的维特比搜索就在它上面进行。

2.2.2.3 分数计算

分数计算模块计算输入特征序列的声学和语言分数，提供给搜索模块使用。应当计算何种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，语言模型分数已编译于解码网络之中，所以只需要计算声学分数即可；而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。在分数计算方面的研究主要集中于如何实现分数的快速计算，对于声学分数通常从如下三个方面进行：

- 硬件加速。利用 CPU 矢量计算器 [65] 或通用图形处理器 (Graphics Processing Unit, GPU) [66] 加速分数计算。这类方法通常不会带来计算误差（与硬件实现相关），所以对识别精度没有影响。
- 模型简化。采用复杂度更小、参数更少的模型，或通过聚类和参数共享减小模型的复杂度。例如采用半连续 HMM [67] 替代连续 HMM，以及传统参数共享方法 [58] 亦属此类。这类算法一般都会带来识别精度的损失，所以需要在识别精度、模型复杂度和计算速度之间做好权衡。
- 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的识别精度损失。一般而言，衡量一个近似算法是否可用的标准是看其带来的相对识别精度损失是否可控制在 5% 以内 [68]。

对于语言分数计算，当采用 N 元文法模型时，通常从两个途径进行加速：1) 减小每次查表操作的耗时，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检索 [69, 70]；2) 引入分数缓存 [71] 减少查表次数。

2.2.2.4 维特比搜索

搜索模块负责在解码网络上搜索得到最优路径。不论是静态还是动态网络解码器，目前都以令牌传递算法 [72] 为主流的搜索算法。

令牌传递是 Viterbi 算法的另一种更为普遍和简便的实现形式。在该算法中，每一个 HMM 状态都可以关联一个令牌 (Token)，令牌中存储着到当前帧为止该令牌所经历的历史路径和路径分数。解码是把令牌从解码网络的初始状态按照状态转移的约束（即：沿解码网络的边）向终止状态传递的过程。每一帧令牌向前传递一次，传递的同时更新令牌分数与路径信息，并在多个令牌同时传递到一个状态上时进行令牌合并——只保留分数最高的令牌。在所有帧都处理完成后，即可得到最优路径和最优路径分数。

令牌传递算法的优点是可以方便地在令牌上附着其他信息以便在状态节点之间传递，从而实现更复杂的解码过程，例如：为了生成词图，在令牌合并时，不是将次优令牌丢弃，而是作为其他候选路径附于合并后的令牌之上。

2.2.2.5 剪枝

LVCSR 中，搜索空间的过于庞大使得对整个解码网络进行全搜索不可能实现，剪枝用于在解码过程中把分数过低的路径剔除，从而达到降低计算量，同时保证识别性能基本不降的目的。根据解码器的不同，剪枝可应用于搜索的各个阶段（如：词首或词尾），或是各个层次（如：词级、音素级、状态级、令牌级），但都可归纳为两种基本方式：

- 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 的最优路径分数，则第 t 帧的最优路径分数为：

$$v_{max}(t) = \max_s v(t, s) \quad (2-44)$$

令 f 为剪枝门限（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的 N 条路径， N 为设定的剪枝门限。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 [73]。

2.2.2.6 路径管理和词图

路径管理模型主要作用是对搜索过程中得到的路径链表进行回溯，生成 1-Best、N-Best 结果和词图。同时，它也负责对剪枝过程中剪掉的“垃圾”路径进行内存回收等操作。

由于 N-Best 候选序列中往往存在大量的公共部分（比如图 2-4 中的“time”这个词），因此将公共部分压缩在一起进行表示，也就是将多个候选序列转换成一张图来进行存储，将会更加高效。在语音识别中，这样的一张图就称为词图。词图的生成所带来的额外操作与生成 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的边和结点删去，得到最终比较紧致的词图。

词图具有较多用途（一些例子如图 2-5 所示），因此相较于 N-Best 候选序列，词图的处理往往是语音识别系统中的一个标准模块。

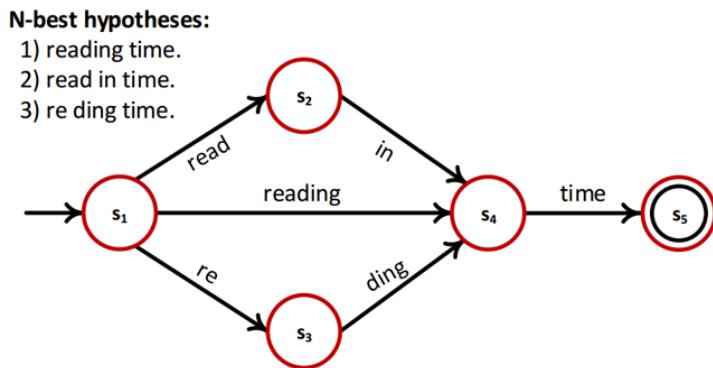


图 2-4 N-Best 候选序列和词图

Fig 2-4 N-Best Hypothesis and Lattices



图 2-5 词图的用途举例

Fig 2-5 Applications of Lattices in ASR

2.3 本章小结

在这章中我们简要介绍了语音识别的基本内容，首先讨论了特征提取，包括 MFCC, PLP, FBANK 等三种语音识别中常用的特征；接着介绍了语音识别中最成功的声学模型——HMM 以及使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。

同时，本章回顾了语音识别推理阶段的搜索解码部分的基本内容和方法。解码器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。在语音识别推理阶段，解码器是语音识别系统的灵魂，所有的信息都汇集于此。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识别结果。因此，如何将各种性质相异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问题。从解码器的作用

来看，它既是语音识别研究中验证各种理论、模型、算法的正确性的基本实验平台，也是构建实用系统的基础。因此，在解码器的设计中也需要兼顾研究的方便与工程实际应用。

第三章 语音识别中的深度序列模型

最早在 1990 年 [24, 25, 26]，已经有研究者提出使用神经网络 (Neural Network, NN) 来对状态输出概率进行建模的方法。近几年，随着计算资源的不断提高，在 NN-HMM 混合系统的基础上，微软的研究者提出了上下文相关-深度神经网络-隐马尔可夫模型混合系统 (Context-Dependent Deep-Neural-Network Hidden Markov Model, CD-DNN-HMM) [28, 29]，即 DNN-HMM 混合系统，将语音识别系统的性能进一步提高了相对 30%，这一系统是对传统的 GMM-HMM 系统的重要改进，并使得语音识别可以进行真正的商用。另一方面，得益于分类能力的提升，深度学习模型的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的推理框架，已经成为当前研究的热点。这类序列模型还被应用于鲁棒语音识别，通过联合优化，迁移学习，序列鉴别性训练等方式改善原来的信号增强和语音识别联合训练系统。

3.1 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。

3.1.1 深度神经网络

神经网络 [74] 也被称为多层感知器，它是一种最基本的神经网络结构，也是目前使用最广泛的结构。深度神经网络 (DNN) 是一个含有很多 (超过两个) 隐层的多层感知器。图 3-1 绘制了一个总共五层的深度神经网络，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。神经网络的运算可以被定义为：

$$\mathbf{y}^l = f(\mathbf{x}^l) = f(\mathbf{W}^l \mathbf{y}^{l-1} + \mathbf{b}^l), 0 < l < L \quad (3-1)$$

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。 $\mathbf{y}^0 = \mathbf{o}$ 是网络的输入特征向量。 f 是一种对激励向量进行元素级计算的激活函数。常用的激活函数有：

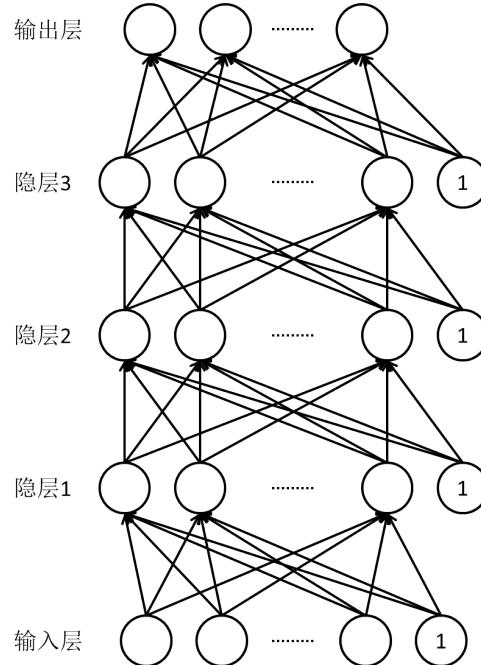


图 3-1 深度神经网络

Fig 3-1 Deep neural network

- sigmoid 函数

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3-2)$$

- 双曲正切函数

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3-3)$$

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

- 整流线性单元 (ReLU)

$$\text{ReLU}(x) = \max(0, x) \quad (3-4)$$

它的导数更加简洁且不会随着层数的增多而出现梯度消失现象。

如图 3-2 所示，其中黑色的曲线为 sigmoid；红色的曲线为 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。

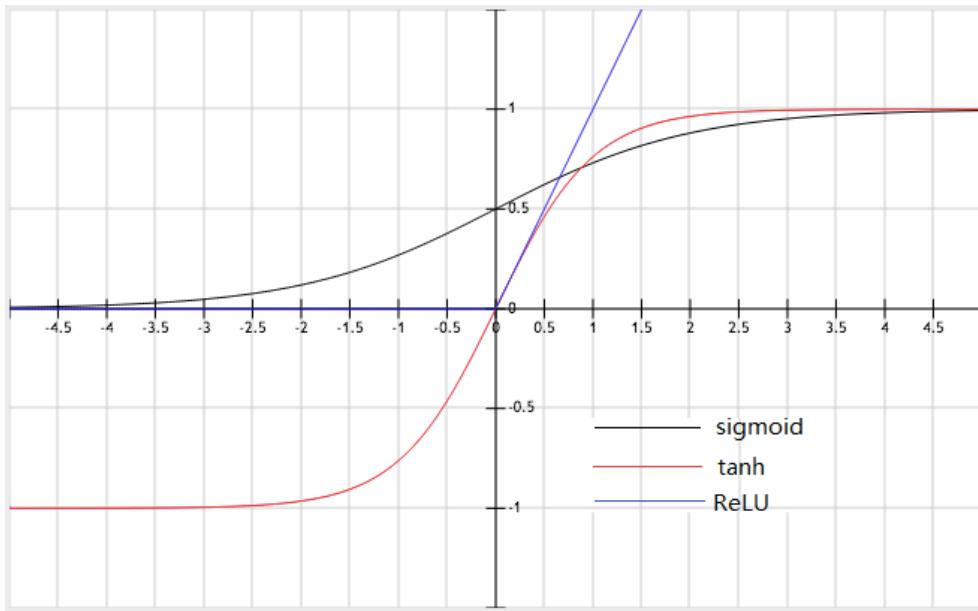


图 3-2 激活函数

Fig 3-2 Activation functions

深度神经网络输出层的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：

$$\mathbf{y}^L = \mathbf{x}^L = \mathbf{W}^L \mathbf{y}^{L-1} + \mathbf{b}^L \quad (3-5)$$

如果是多分类的任务，通常会使用输出层的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 的概率，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：

$$y_i^L = P_{\text{dnn}}(i|\mathbf{o}) = \text{softmax}_i(\mathbf{x}^L) \quad (3-6)$$

$$= \frac{e^{x_i^L}}{\sum_{j=1}^C e^{x_j^L}} \quad (3-7)$$

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

3.1.2 卷积神经网络

传统的深度卷积神经网络 (convolutional neural network, CNN) [75] 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，它的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，每个卷积层的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 11×40 。这里 N_t 是上下文窗口的大小， N_f 是提取的特征的维度。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，池化操作中的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。

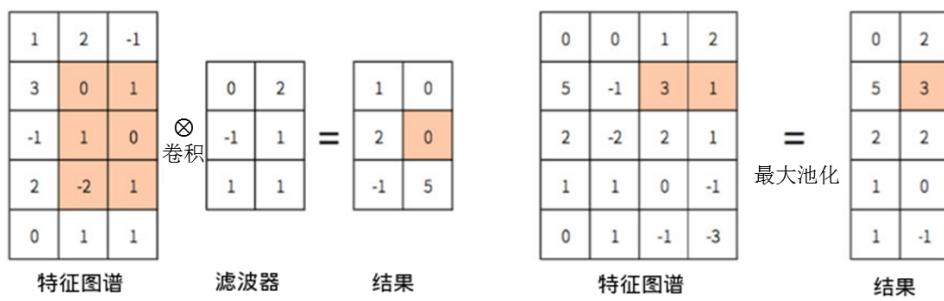


图 3-3 卷积操作和池化操作，左边为卷积，右边为最大池化

Fig 3-3 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。卷积的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的特征图谱，卷积的数学定义为：

$$\mathbf{y} = \mathbf{W} \otimes \mathbf{x} \quad (3-8)$$

$$y_{i,j} = \sum_{a=1}^A \sum_{b=1}^B W_{ab} x_{i+a-1, j+b-1} \quad (3-9)$$

$$0 < i \leq W - A + 1 \quad (3-10)$$

$$0 < j \leq H - B + 1 \quad (3-11)$$

$$(3-12)$$

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

整个卷积层可以由如下公式表示：

$$\mathbf{y}_i^l = \sigma \left(\sum_{j=1}^N (\mathbf{W}_{i,j}^l \otimes \mathbf{y}_j^{l-1}) \oplus b_i^l \right) \quad (3-13)$$

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的滤波器。 b_i^l 是一个应用于整张特征图谱的偏置值， \otimes 为卷积操作， \oplus 表示特征图谱中的每个元素都加上相同的值 b_i^l 。 σ 是激活函数，通常使用 sigmoid 或者 ReLU。 N 是输入特征图谱的个数。由于各个感受野之间的参数是共享的，因而 CNN 中的参数个数相对 DNN 是很少的。

池化层通常扮演降采样的功能，它输入若干特征图谱并输出降低了分辨率后的特征图谱。本文使用了最大值池化方法，一个 1×2 的最大值池化的效果如图 3-3 中的右边部分所示。

3.1.3 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 的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，内部状态将过去的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：

$$\mathbf{y}_t^l = \sigma(\mathbf{W}_y^l \mathbf{y}_t^{l-1} + \mathbf{W}_h^l \mathbf{y}_{t-1}^l + \mathbf{b}^l) \quad (3-14)$$

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的权重矩阵。

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的建模中扮演了很重要的作用。在 [76, 77, 78, 79] 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，它不仅能捕捉过去的历史信息，也能捕捉未来的整个输入序列的信息。

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的激活函数，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 帧之前的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 [80]，即误差无法沿着时间维传播太多帧。为了解决这一问题，在 [81] 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

免梯度消失的问题，一个典型的 LSTM 可由如下公式描述：

$$\mathbf{i}_t^l = \sigma(\mathbf{W}_{yi}^l \mathbf{y}_t^{l-1} + \mathbf{W}_{hi}^l \mathbf{y}_{t-1}^l + \mathbf{W}_{ci} \mathbf{c}_{t-1}^l + \mathbf{b}_i^l) \quad (3-15)$$

$$\mathbf{f}_t^l = \sigma(\mathbf{W}_{yf}^l \mathbf{y}_t^{l-1} + \mathbf{W}_{hf}^l \mathbf{y}_{t-1}^l + \mathbf{W}_{cf} \mathbf{c}_{t-1}^l + \mathbf{b}_f^l) \quad (3-16)$$

$$\mathbf{c}_t^l = \mathbf{f}_t^l \odot \mathbf{c}_{t-1}^l + \mathbf{i}_t^l \odot \tanh(\mathbf{W}_{yc}^l \mathbf{y}_t^{l-1} + \mathbf{W}_{hc} \mathbf{c}_{t-1}^l + \mathbf{b}_c^l) \quad (3-17)$$

$$\mathbf{o}_t^l = \sigma(\mathbf{W}_{yo}^l \mathbf{y}_t^{l-1} + \mathbf{W}_{ho}^l \mathbf{y}_{t-1}^l + \mathbf{W}_{co} \mathbf{c}_{t-1}^l + \mathbf{b}_o^l) \quad (3-18)$$

$$\mathbf{y}_t^l = \mathbf{o}_t^l \odot \tanh(\mathbf{c}_t^l) \quad (3-19)$$

和循环神经网络一样，它需要从第一帧开始迭代的进行前向传播，这里 σ 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 表示元素级的乘法。 $\mathbf{W}_*, \mathbf{b}_*$ 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 结构的不同之处在于，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 [78] 中的建议，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 3-15 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：

$$\mathbf{y}_t^l = \mathbf{W}_r(\mathbf{o}_t^l \odot \tanh(\mathbf{c}_t^l)) \quad (3-20)$$

3.2 神经网络训练

3.2.1 训练准则

在 DNN 的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：

$$\mathcal{F}_{\text{MSE}}(\mathcal{M}) = \frac{1}{T} \sum_{t=1}^T \|\mathbf{y}_t^L - \bar{\mathbf{y}}_t\|_2^2 \quad (3-21)$$

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

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：

$$\mathcal{F}_{\text{CE}}(\mathcal{M}) = -\frac{1}{T} \sum_{t=1}^T \sum_{c=1}^C \bar{y}_t(c) \log y_t^L(c) \quad (3-22)$$

这里 $\bar{y}_t(c)$ 是标注中类 c 的概率， $y_t^L(c)$ 是神经网络估计出的类 c 的概率。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：

$$\bar{y}_t(c) = \begin{cases} 1 & c = l_t \\ 0 & c \neq l_t \end{cases} \quad (3-23)$$

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

$$\mathcal{F}_{\text{NLL}}(\mathcal{M}) = -\log y_t^L(l_t) \quad (3-24)$$

3.2.2 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) [74] 算法来进行参数优化。由于 BP 算法基于梯度下降的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。

最顶层的权重矩阵的梯度取决于所使用的训练准则。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

$$\frac{\partial \mathcal{F}_{\text{MSE}}(\mathcal{M})}{\partial W_{ij}^L} = \frac{\partial \mathcal{F}_{\text{MSE}}(\mathcal{M})}{\partial x_j^L} \frac{\partial x_j^L}{\partial W_{ij}^L} \quad (3-25)$$

$$= e_j^L \frac{\partial x_j^L}{\partial W_{ij}^L} \quad (3-26)$$

$$= (x_j^L - \bar{y}_j) y_i^{L-1} \quad (3-27)$$

这里 e_j^L 为误差信号：

$$\mathbf{e}^L \triangleq \left[\frac{\partial \mathcal{F}_{\text{MSE}}(\mathcal{M})}{\partial x_1^L}, \dots, \frac{\partial \mathcal{F}_{\text{MSE}}(\mathcal{M})}{\partial x_N^L} \right]^\top \quad (3-28)$$

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

$$\nabla_{\mathbf{W}^L} \mathcal{F}_{\text{MSE}}(\mathcal{M}) = \mathbf{e}^L \mathbf{y}^{L-1\top} = (\mathbf{x}^L - \bar{\mathbf{y}}) \mathbf{y}^{L-1\top} \quad (3-29)$$

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：

$$e_j^L = \frac{\partial \mathcal{F}_{\text{CE}}(\mathcal{M})}{\partial x_j^L} = -\frac{\partial \sum_{c=1}^C \bar{y}_c \log \text{softmax}_c(\mathbf{x}^L)}{\partial x_j^L} \quad (3-30)$$

$$= \frac{\partial \sum_{c=1}^C \bar{y}_c \log \sum_{i=1}^C e^{x_i^L}}{\partial x_j^L} - \frac{\partial \sum_{c=1}^C \bar{y}_c \log e^{x_c^L}}{\partial x_j^L} \quad (3-31)$$

$$= \frac{e^{x_j^L}}{\sum_{i=1}^C e^{x_i^L}} - \bar{y}_j \quad (3-32)$$

$$= y_j^L - \bar{y}_j \quad (3-33)$$

接着可以求出对 \mathbf{W}^L 的梯度:

$$\nabla_{\mathbf{W}^L} \mathcal{F}_{\text{CE}}(\mathcal{M}) = (\mathbf{y}^L - \bar{\mathbf{y}}) \mathbf{y}^{L-1}^\top \quad (3-34)$$

值得注意的一点是，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 $\mathbf{y}^L = \text{softmax}(\mathbf{x}^L)$ 。

对于非顶层的权重矩阵的梯度 $0 < l < L$ ，则有：

$$\frac{\partial \mathcal{F}(\mathcal{M})}{\partial W_{ij}^l} = \frac{\partial \mathcal{F}(\mathcal{M})}{\partial x_j^l} \frac{\partial x_j^l}{\partial W_{ij}^l} \quad (3-35)$$

$$= e_j^l \frac{\partial x_j^l}{\partial W_{ij}^l} \quad (3-36)$$

$$= e_j^l y_i^{l-1} \quad (3-37)$$

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

$$e_i^l = \frac{\partial \mathcal{F}(\mathcal{M})}{\partial x_i^l} \quad (3-38)$$

$$= \sum_{j=1}^{N^{l+1}} \frac{\partial \mathcal{F}(\mathcal{M})}{\partial x_j^{l+1}} \frac{\partial x_j^{l+1}}{\partial y_i^l} \frac{\partial y_i^l}{\partial x_i^l} \quad (3-39)$$

$$= \sigma'(x_i^l) \sum_{j=1}^{N^{l+1}} e_j^{l+1} W_{ji}^{l+1} \quad (3-40)$$

这里 N^{l+1} 是第 $l+1$ 层隐层节点数，写出矩阵形式即：

$$\mathbf{e}^l = (\mathbf{W}^{l+1}^\top \mathbf{e}^{l+1}) \odot \sigma'(\mathbf{x}^l) \quad (3-41)$$

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：

$$\sigma'(x_j^l) = (1 - y_j^l) y_j^l \quad (3-42)$$

对于 ReLU，它等于

$$\sigma'(x_j^l) = \begin{cases} 1 & x_j^l > 0 \\ 0 & x_j^l \leq 0 \end{cases} \quad (3-43)$$

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的矩阵形式为：

$$\nabla_{\mathbf{W}^l} \mathcal{F}(\mathcal{M}) = \mathbf{e}^l \mathbf{y}^{l-1}^\top \quad (3-44)$$

从公式3-41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。

3.2.3 小批量随机梯度下降

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) 训练的样本计算梯度。批量大小的选择通常会同时影响收敛速度与模型性能。

最简单的批量选择方法是使用整个训练集，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，因而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) [82]，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。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** 相比批量算法的优势所在：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的全局最优。

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** 只计算一个样本的梯度，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** 和批量训练的折中方案，小批量 (mini-batch) 训练。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** 的跳出局部最优的性质。

3.2.4 惯性系数

惯性系数 [83] 是优化中一种加速收敛的方法，它的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：

$$\Delta \mathbf{W}_p^l = \rho \Delta \mathbf{W}_{p-1}^l + (1 - \rho) \frac{1}{M} \sum_{m=1}^M \nabla_{\mathbf{w}_p^l} \mathcal{F}(\mathcal{M}, \mathbf{o}_m, \bar{\mathbf{y}}_m) \quad (3-45)$$

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。 ρ 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的方差，因此可以加速训练。

3.2.5 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，这样计算出的梯度就会非常小。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 [84] 中，Lecun 建议从一个均值为 0，方差为 $\frac{1}{\sqrt{N^l}}$ 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 模型的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。

3.2.5.1 深度置信网络

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) 组成的生成性模型。

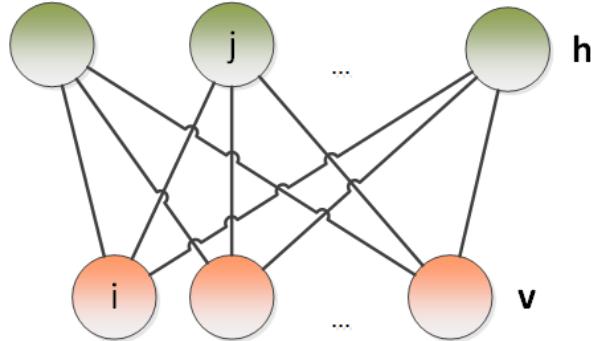


图 3-4 受限玻尔兹曼机

Fig 3-4 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。它的结构如 3-4 所示¹，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。

RBM 对于每一对给定的隐藏层向量 \mathbf{h} 和可见层向量 \mathbf{v} 定义了一个能量函数：

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{a}^\top \mathbf{v} - \mathbf{b}^\top \mathbf{h} - \mathbf{h}^\top \mathbf{W} \mathbf{v} \quad (3-46)$$

¹图片引用自 [27]

这里 \mathbf{W} 是连接可见层和隐藏层之间的权重矩阵， \mathbf{a} 和 \mathbf{b} 分别是可见层和隐藏层的偏置向量。如果可见层的值域是实数域，那么能量函数定义为：

$$E(\mathbf{v}, \mathbf{h}) = \frac{1}{2}(\mathbf{v} - \mathbf{a})^\top(\mathbf{v} - \mathbf{a}) - \mathbf{b}^\top \mathbf{h} - \mathbf{h}^\top \mathbf{W} \mathbf{v} \quad (3-47)$$

定义完能量函数后可以定义可见向量和隐藏向量的联合概率分布：

$$P(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v}, \mathbf{h})}}{Z} \quad (3-48)$$

这里 $Z = \sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}$ 被称为配分函数 (partition function)。

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。所以，可以很方便的计算后验概率 $P(\mathbf{v}|\mathbf{h})$ 和 $P(\mathbf{h}|\mathbf{v})$ ：

$$P(\mathbf{h}|\mathbf{v}) = \frac{e^{-E(\mathbf{v}, \mathbf{h})}}{\sum_{\bar{\mathbf{h}}} e^{-E(\mathbf{v}, \bar{\mathbf{h}})}} \quad (3-49)$$

$$= \frac{\prod_i e^{b_i h_i + h_i \mathbf{w}_i^\top \mathbf{v}}}{\prod_i \sum_{\bar{h}_i} e^{b_i \bar{h}_i + \bar{h}_i \mathbf{w}_i^\top \mathbf{v}}} \quad (3-50)$$

$$= \prod_i P(h_i|\mathbf{v}) \quad (3-51)$$

这里 \mathbf{w}_i^\top 是矩阵 \mathbf{W} 的第 i 行，该公式表明在给定可见层向量后，隐藏层神经元之间是条件独立的，且：

$$P(h_i = 1|\mathbf{v}) = \frac{e^{b_i + \mathbf{w}_i^\top \mathbf{v}}}{e^{b_i + \mathbf{w}_i^\top \mathbf{v}} + 1} = \sigma(b_i + \mathbf{w}_i^\top \mathbf{v}) \quad (3-52)$$

这里 σ 是 sigmoid 函数。同样，我们可以推导出二进制可见层神经元的后验概率：

$$P(\mathbf{v}|\mathbf{h}) = \sigma(\mathbf{W}^\top \mathbf{h} + \mathbf{a}) \quad (3-53)$$

对于实数可见层神经元，有：

$$P(\mathbf{v}|\mathbf{h}) = \mathcal{N}(\mathbf{v}; \mathbf{W}^\top \mathbf{h} + \mathbf{a}, I) \quad (3-54)$$

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，然后用新的特征继续训练一个 RBM。最终 DBN 训练完成后将 DBN 中的权重直接作为 DNN 网络的初始化。

使用 DBN 进行预训练有如下潜在的优势：

- DNN 是高度非线性且非凸的，初始化点会很大程度的影响最终模型。
- 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，因此潜在的对模型进行了正则化。
- 在预训练时可以利用大量的无标注的数据。

3.2.5.2 鉴别性预训练

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。鉴别性预训练的流程如下：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，并且鉴别性的训练整个网络若干轮，直至达到期望的层数。预训练的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提高神经网络的鲁棒性。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，预训练带来的性能提升就不那么明显了。

3.3 深度神经网络-隐马尔可夫模型混合系统

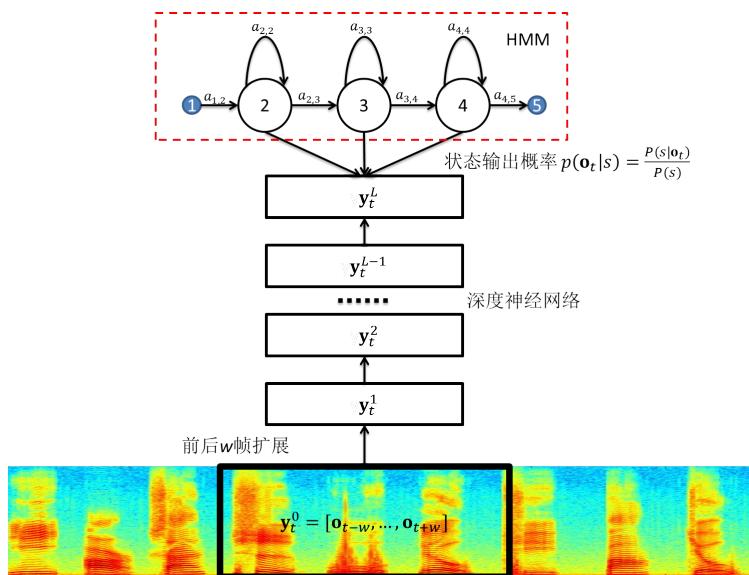


图 3-5 DNN-HMM 混合系统
Fig 3-5 Structure of DNN-HMM hybrid system

在前文介绍的深度神经网络并不能直接为语音信号建模，这是由于语音信号是一个时序连续信号，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 的方法用于语音识别 [85]。目前有效的方法是如图3-5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动态变化，而 DNN 则用于估计语音特征向量的输出概率，通常是用来估计给定观测语音特征后，该特征由 HMM 中某个状态释放的后验概率。DNN-HMM 的

优点在于它能方便的利用 DNN 内在的鉴别性能力，另外训练过程可直接使用维特比算法，解码过程也无需有特别改变。

在 [24, 86, 87] 中，该方法被称为 ANN-HMM 混合模型，它只使用了上下文无关的音素状态且仅用于小词汇语音识别。随后在 [25] 中被扩展为上下文相关的音素建模，以及被用于大词汇自动语音识别任务 [88]。但是，因为过去的计算能力有限，通常只使用了两层隐层的神经网络，所在并没有比 GMM-HMM 框架取得更好的性能。

最近的技术发展 [28, 29, 64, 89] 则说明，如下几个改变可以使其获得更重大的识别性能提升。

- 首先，使用更深的深度神经网络，比如拥有 6 个隐层的 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可夫模型 (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的好处，即它对 CD-GMM-HMM 系统的修改最小。

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 来估计状态的后验概率 $P(s_t = s | \mathbf{o}_t)$ 。这和传统的 GMM 是不同的，在 GMM 中，我们使用不同的 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 $2w + 1$ 帧大小的窗口特征 $\mathbf{y}_t^0 = [\mathbf{o}_{t-w}^\top, \dots, \mathbf{o}_t^\top, \dots, \mathbf{o}_{t+w}^\top]^\top$

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：

$$p(\mathbf{o}_t | s_t = s) = \frac{P(s_t = s | \mathbf{o}_t)p(\mathbf{o}_t)}{P(s)} \quad (3-55)$$

其中， $p(s) = \frac{T_s}{T}$ 是从训练集中统计出的每个状态的先验概率。 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 的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，隐马尔可夫模型 HMM 和 GMM-HMM 系统中的状态绑定结构。所以 CD-DNN-HMM 的第一步是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 系统计算得到的状态对齐 (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 3-55 来得到似然进行解码。

3.4 端到端语音识别中的深度序列模型

得益于标注数据的增多和算力的改善，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，以取得更好的序列建模能力。如图 3-6 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。

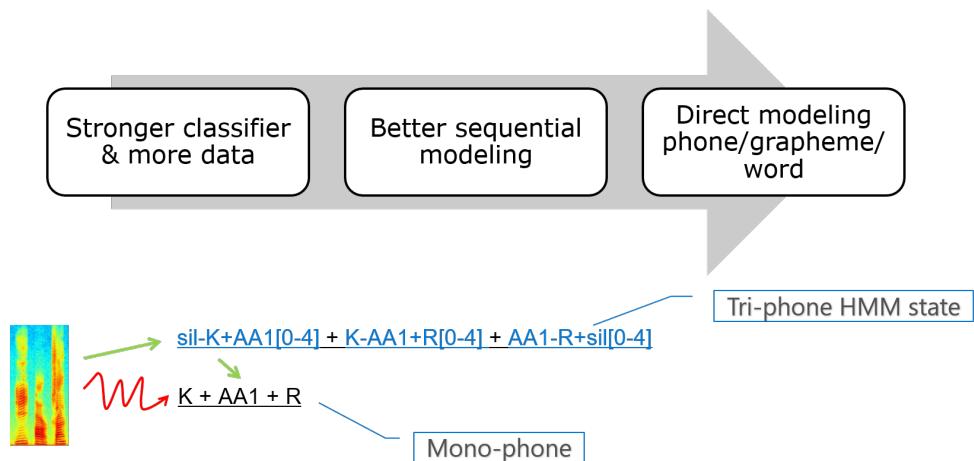


图 3-6 端到端系统举例

Fig 3-6 Motivation and Example of End-to-end Modeling

图 3-7 给出了一系列端到端序列建模模型，下文中将逐一进行讨论。

3.4.1 无词图鉴别性训练模型

无词图鉴别性训练 (*lattice-free maximum mutual information*, LF-MMI [90, 91]) 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的词图。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 3-56 计算声学模型的序列后验概率，表示为 LF-MMI。

$$\mathcal{F}_{\text{LF-MMI}} = \sum_u \log \frac{\sum_{\mathbf{L}_u} p(\mathbf{O}_u | \mathbf{L}_u)^\kappa P(\mathbf{L}_u)}{\sum_{\mathbf{L}} p(\mathbf{O}_u | \mathbf{L})^\kappa P(\mathbf{L})} \quad (3-56)$$

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 模型的公式定义得到。

该方法与传统鉴别性训练不同之处在于：

- 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的左右帧移。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 $P(\mathbf{L})$, $P(\mathbf{L}_u)$ 是使用一个在训练标注文本中训练得到的音素语言模型。
- 一个专用的 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，同时其他状态来模拟输出标签单元。这里的不同之处在于每个 tri-phone 都维护一个它专有的 blank 建模。
- 输出帧率被降低了 3 倍。

3.4.2 连接时序模型

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。连接时序模型（Connectionist Temporal Classification, CTC）提出自动学习输入帧特征序列与其对应的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 个不同的标签，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前端预测模型进行更新。

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的联系，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 引入上述的 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。

$$P(\mathbf{L}|\mathbf{O}) = \sum_{\mathbf{q} \in \mathcal{B}(\mathbf{L})} P(\mathbf{q}|\mathbf{O}) = \sum_{\mathbf{q}} \prod_{t=1}^T p(q_t|\mathbf{O}) \quad (3-57)$$

公式中的 \mathcal{B} 是一个一对多的映射函数¹:

$$\mathcal{B} : \mathbb{L} \mapsto \mathbb{L} \cup \{\text{blank}\} \quad (3-58)$$

¹在最早文献 [92] 中使用的公式定义为一个多对一的映射函数，同时使用它的反函数来定义 $\mathcal{B}^{-1}(\cdot)$ CTC 模型中的映射。这里我们修改为一个一对多的映射函数，使公式与 HMM 中的定义更加一致。

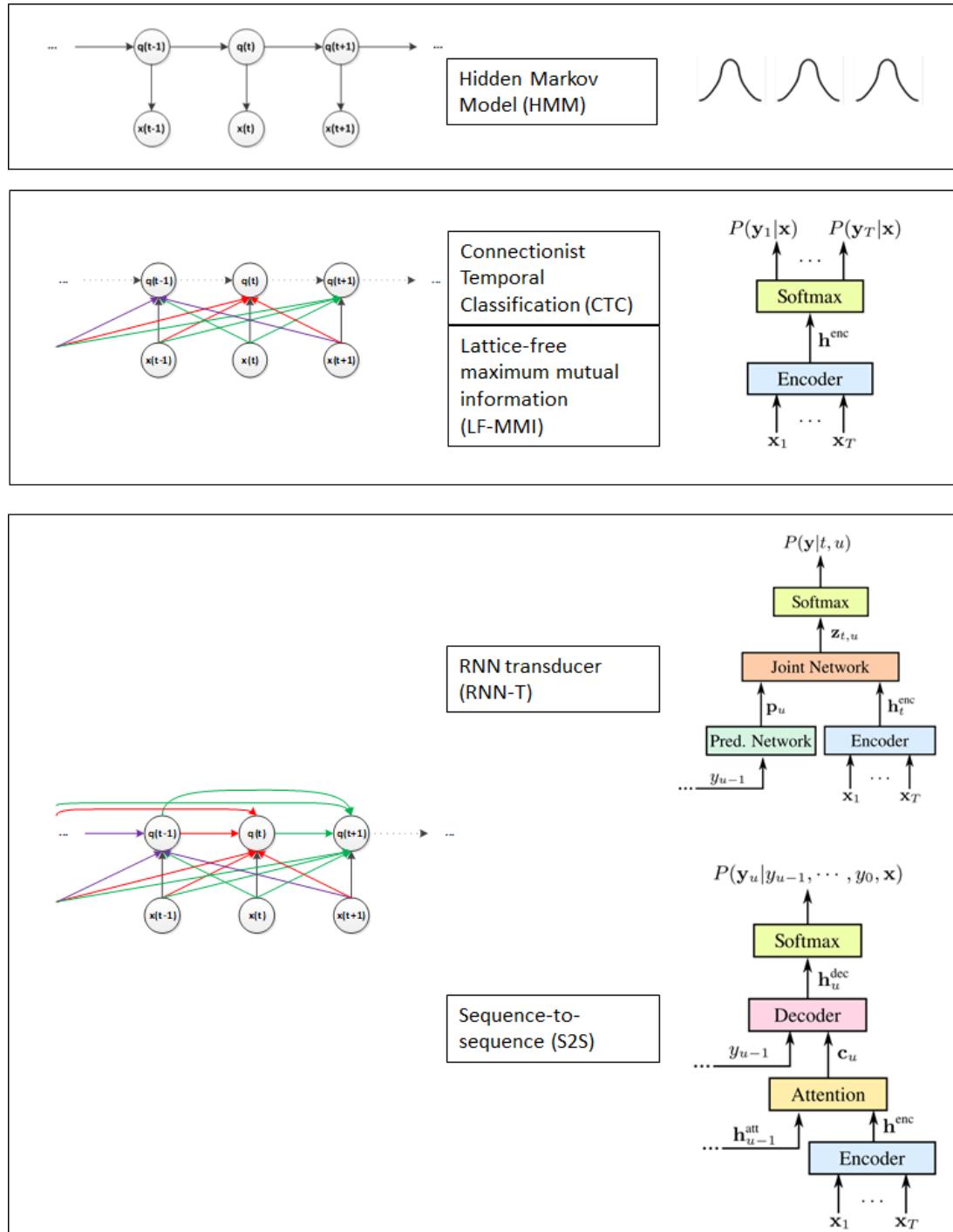


图 3-7 传统系统与端到端系统比较

Fig 3-7 Comparison of Various End-to-end Models

\mathcal{B} 决定了标签序列 \mathbf{L} 和它相应的由建模单元所组成的模型状态序列 \mathbf{q} 。这一映射的做法是在每个组成 \mathbf{L} 的 l 的标签中，插入一个可选的并且可重复的 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 [81]。

3.4.3 基于注意力机制的序列到序列模型

另一类端到端模型称为序列到序列模型。下文的讨论将以基于注意力机制的序列到序列模型 [93] 为主进行介绍，该类模型也是当前主要的研究热点。该模型预测了在直接给定特征序列 \mathbf{x} 下的标签序列和之前已经预测出的标签历史序列 $\mathbf{l}_{1:i-1}$ 的后验概率。

$$P(\mathbf{l}|\mathbf{x}) = \prod_i P(l_i|\mathbf{x}, \mathbf{l}_{1:i-1}) \quad (3-59)$$

$$\mathbf{h} = \text{Encoder}(\mathbf{x}) \quad (3-60)$$

$$P(l_i|\mathbf{x}, \mathbf{l}_{1:i-1}) = \text{AttentionDecoder}(\mathbf{h}, s_{i-1}) \quad (3-61)$$

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）的上一输出相应的隐层状态。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 [94] 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。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，这种端到端系统并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 [95]。

3.5 鲁棒语音识别中的深度序列模型

近年来随着深度学习的发展，将语音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。下文将以笔者在鸡尾酒会问题中的一些研究为例展开讨论。

3.5.1 排列不变性训练及其在鸡尾酒会问题中的应用

鸡尾酒会问题 [96, 97] 是指多说话人重叠语音的识别问题，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 [98, 99, 100, 101]。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的说话人情况。

排列不变性训练是解决该问题的一种有效框架 [102]。该框架首先通过选取最小的当前排列下的推理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。

$$\mathcal{J}_{\text{CE-PIT}} = \sum_u \min_{s' \in \mathbf{S}} \sum_t \frac{1}{N} \sum_{n \in [1, N]} CE(l_{utn}^{(s')}, l_{utn}^{(r)}) \quad (3-62)$$

公式中 \mathbf{S} 表示所有针对标注和推理序列的可能的排列组合。 $l_{utn}^{(s')}$ 为在排列 s' ，第 t 帧，第 u 个句子上的第 n 个推理结果。 $l_{utn}^{(r)}$ 是相应的从干净语音进行对齐算法得到的标注序列 [40]。PIT-ASR 准则 [102] 优雅地将语音分离，说话人跟踪，语音识别都融合在一个系统里，如图 3-8(a) 所示。

在接下来的章节中，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：逐帧的语音分离，说话人跟踪，语音识别，如图 3-9 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的问题。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。

3.5.2 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，以便使用平行的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，即混合语音数据上，并尝试得到每个说话人相应的话语内容。这里的教师模型输入源领域，即干净语音，并尝试分别对每个说话人得到相应的推理序列。

这一针对自身的迁移学习将最小化混合语音模型和干净语音模型的输出分布之间的 KL 散度（KLD）。这一 KLD 散度被定义为句子层面的干净语音和混合语音推理分布之间的 PIT 准则，如下所示，

$$\begin{aligned} \mathcal{J}_{\text{KLD-PIT}} &= \sum_u \min_{s' \in \mathbf{S}} \sum_t \frac{1}{N} \sum_{n \in [1, N]} \\ &KLD(P(l_{utn}^{(c)} | \mathbf{O}_{un}^{(r)}), P(l_{utn}^{(s')} | \mathbf{O}_u^{(m)})) \end{aligned} \quad (3-63)$$

该式中，每个 $KLD(\cdot)$ 对的计算过程和经典领域自适应方式的迁移学习公司相似。值得注意的一点是，当该方法被应用到如图 3-10 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。

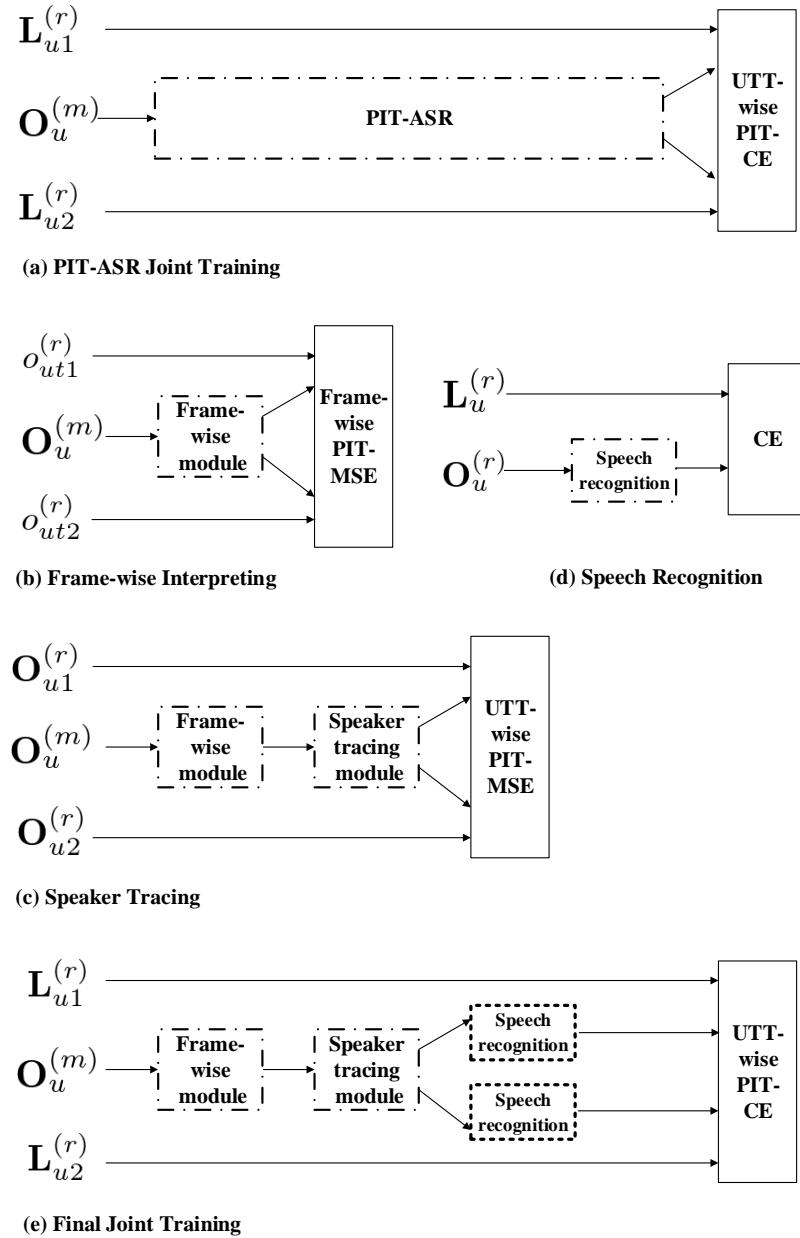


图 3-8 PIT-ASR 联合训练与模块化初始化和逐步联合训练的比较。点划线框表示可以学习的模型参数。点线框表示可以学习的并且是共享的模型参数。



图 3-9 模型系统训练框架。[102] 中提出的结构由虚线框部分表示，其用于推理出每个说话人的语音信息。该结构被模块化（三个实线框）并作逐一增量的预训练。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的神经网络上进行。

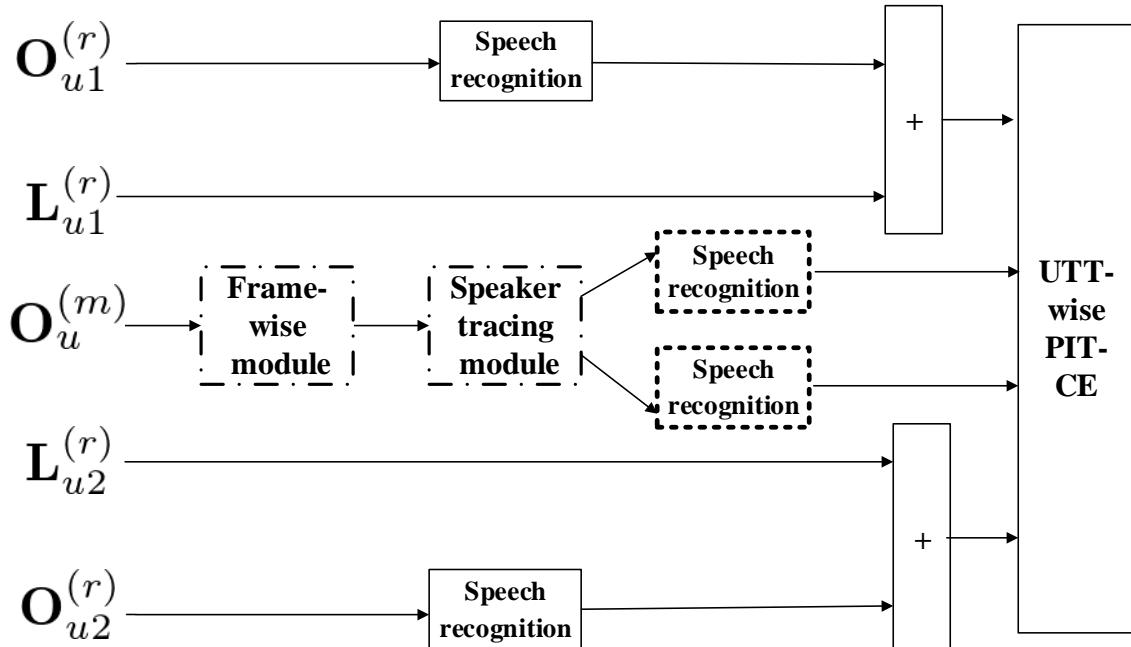


图 3-10 基于迁移学习的逐步联合训练方法。点划线框表示可以学习的参数，点线框表示可以学习并且是共享的模型参数。

图 3-11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的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。

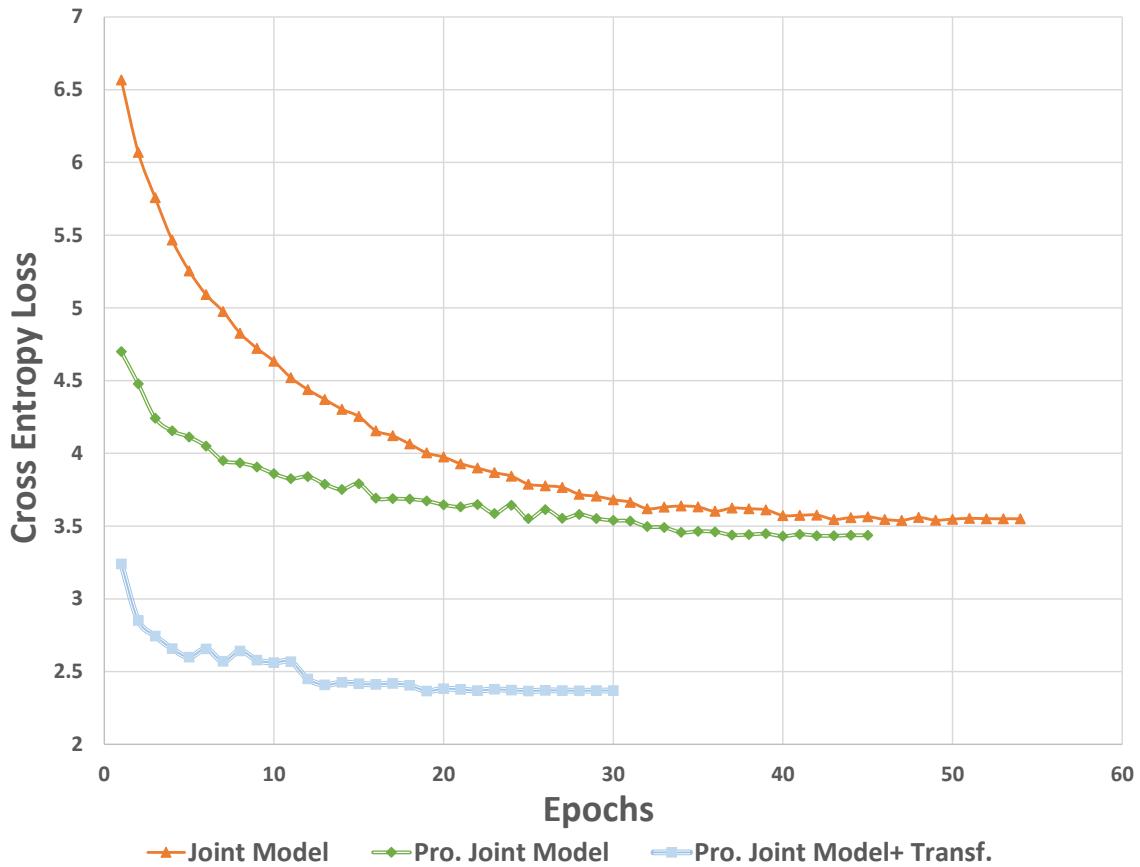


图 3-11 原始联合训练和所提出的方法在验证集上的学习曲线比较。在图中，联合训练，逐步联合训练，基于迁移学习的逐步联合训练被分别表示为 *Joint Model*, *Pro. Joint Model* 和 *Pro. Joint Model + Transf.*。图中每个 epoch 包含 24 小时训练数据。

3.5.3 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的变种，它在进行鉴别性训练的同时，也抑制输出通道上的说话人跟踪错误。

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，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，我们使用如下公式对序列后验概率进行建模，

$$P(\mathbf{L}_u | \mathbf{O}_u) = \frac{p(\mathbf{O}_u | \mathbf{L}_u) P(\mathbf{L}_u)}{p(\mathbf{O}_u)} \quad (3-64)$$

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。 $p(\mathbf{O}_u | \mathbf{L}_u)$ 是相应的声学模型概率。 $p(\mathbf{O}_u)$ 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的。之后在该搜索空间上将计算相应的序列准则。由于之前训练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。

$$\begin{aligned} \mathcal{J}_{\text{LF-DC-MMI}} = & \sum_u \log \left[\frac{\sum_{\mathbf{L}_u} p(\mathbf{O}_u | \mathbf{L}_u)^\kappa P(\mathbf{L}_u)}{\left(\sum_{\mathbf{L}} p(\mathbf{O}_u | \mathbf{L})^\kappa P(\mathbf{L}) \right)^{1-\lambda}} \cdot \right. \\ & \left. \frac{1}{\left(\sum_{\mathbf{L}_{\hat{u}}} p(\mathbf{O}_u | \mathbf{L}_{\hat{u}})^\kappa P(\mathbf{L}_{\hat{u}}) \right)^\lambda} \right] \end{aligned} \quad (3-65)$$

在公式 3-65 中，除当前优化的输出序列之外的其他输出序列表示为 $\mathbf{L}_{\hat{u}}$ 。公式中添加了一项插值系数 λ 。该部分额外式子最后被加入到分母的优化当中。

3.6 本章小结

在本章中，首先回顾了语音识别中最常用的神经网络结构，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的优化细节。随后回顾了深度神经网络-隐马尔可夫模型混合系统，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的各个变种，它们

的特点及优势，为之后几章的改进和推理框架研究进行了铺垫。最后我们介绍了基于深度序列模型的鲁棒语音识别及其在鸡尾酒会问题中的几项研究。

第四章 基于 GPU 并行计算的搜索速度优化

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。该框架可以显著加速现有推理算法，特别是在基于深度序列学习的一系列模型上进行了验证。

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。

4.1 引言

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。

为了减少声学模型推理的计算消耗，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 [103, 104]，定点化 [95]，跳帧 [105, 106, 107] 和端到端系统 [108, 109]。同时算法的改进，比如剪枝 [41, 110] 和向前预测 [61, 111] 是针对解码部分主要的加速方式。

基于 GPU 的并行计算是另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 [112] 的序列批处理 [113] 引入推理阶段，以加速运算。但是，基于 GPU 并行计算的 WFST 解码并不容易实现。尽管一些研究在小型语言模型上取得了一定成功 [114]，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。

这项工作作为 Kaldi 工具包的一个扩展 [115]，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¹，该解码器对语言模型和声学模型没有特别的限制，

¹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 [104, 107]。因此我们的目标

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 [116]。我们对这项工作进行了开源¹，它将会与大多数 Kaldi 脚本相兼容。

4.2 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 [117]，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。

4.2.1 系统框架

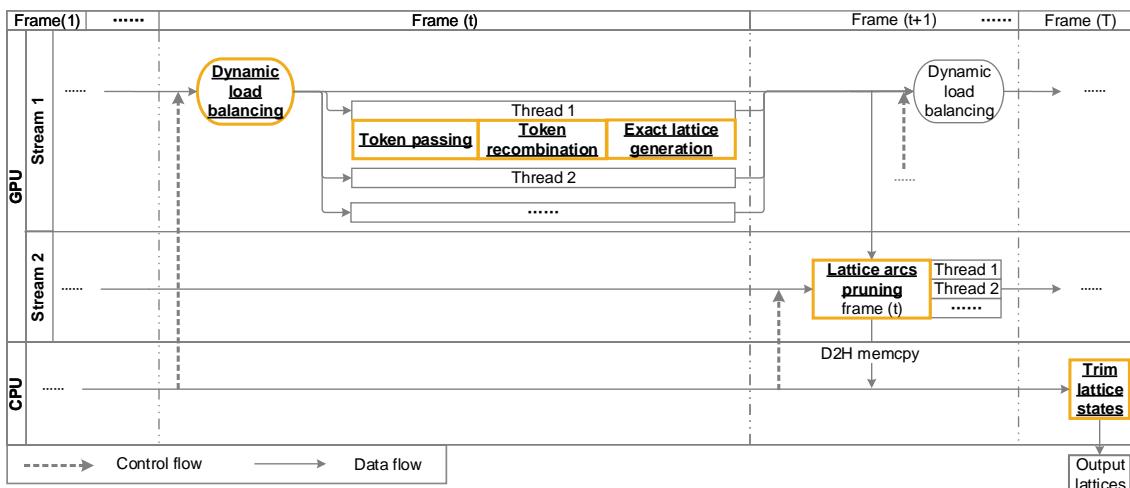


图 4-1 并行维特比束剪枝算法以及精确词图处理系统的框架

图 4-1 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 [115]，但工作在不同的一些设计上，以适应并行计算需求，其罗列于下面的章节。负载均衡算法则处理了线程之间并行的分配，它们具体体现在如何分配 WFST 状态和边。

集中在如何解码海量的离线语音，降低其计算损耗。

¹<https://github.com/chenzhehuai/kaldi/tree/gpu-decoder>

4.3 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消耗，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 [117]，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，其他一个 \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 4-2 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 4-1 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候是串行的。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 critical sections [119] 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。[114] 提出一种在每个状态上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。[120] 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。

我们提出算法 4-1，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 4-2 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。领过

¹ $\text{atomicMin}(*\text{address}, \text{val})$ [118]. Computes $\min(*\text{address}, \text{val})$, writes the result to address , and returns the original $*\text{address}$.

所有的令牌处理后，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。

4.4 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。不同于 [114, 121]，这里作者重新设计了 WFST 结构以减少这种不均；在这项工作中，我们不希望解码算法依赖于特定的声学，词典，语言模型。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消耗¹，这种方法将引入计算和 GPU 内核启动的时间消耗。

受 [122] 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 4-2 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。我们同事将调度中心实现为一个 GPU 原子操作。图 4-2 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。

算法 4-2 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])

```

¹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。

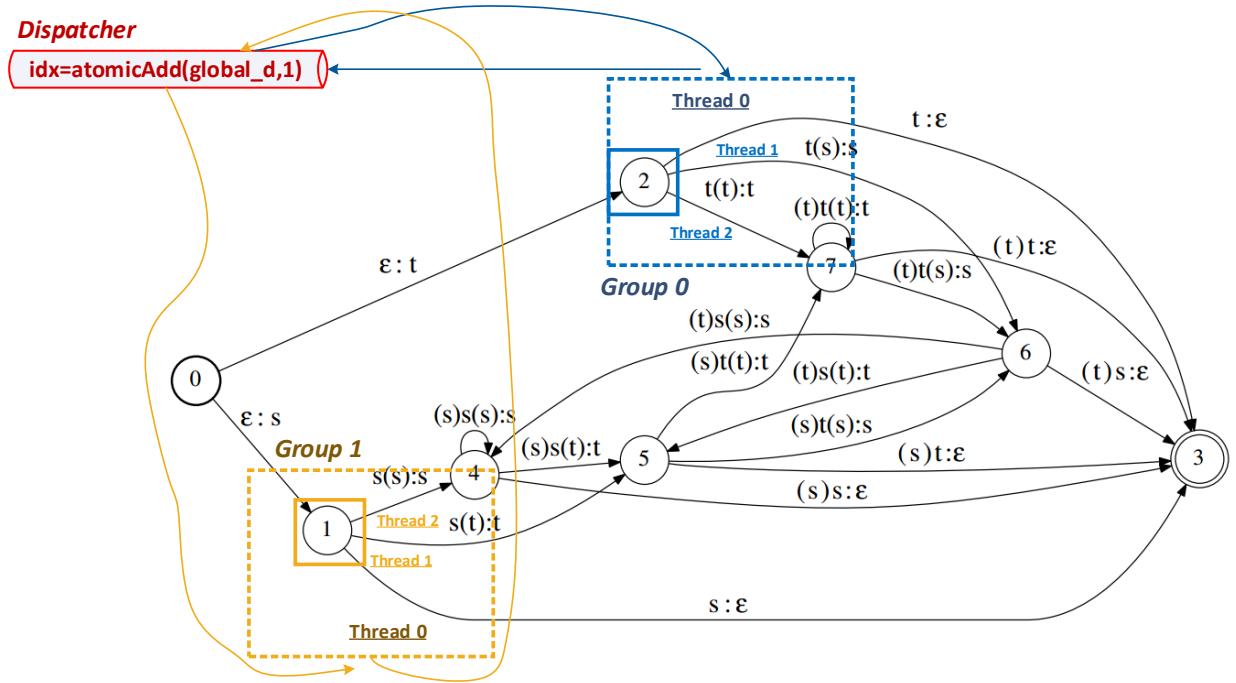


图 4-2 一个针对动态负载均衡的例子。这里的虚线框表示一个 *CUDA cooperative group*，不同的组分表示为不同的颜色。每一个组由线程 0 进行控制。当某个组分处理完了从一个状态出发的所有边，线程 0 将会向调度中心索要下一个令牌，并通知组分里的其他线程。而调度中心使用原子操作来保证每个令牌只分配给一个组分。组 0 和组 1 完全工作在并行方式中。

4.5 并行的词图处理算法

4.5.1 精确的词图生成算法

WFST 精确词图 [116] 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。¹ 但是在 GPU 中实现词图处理算法并不简单。[125] 提出在 GPU 中解码，而后在 CPU 中生成词图，这不仅使解码器速度下降，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 [116] 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

¹除了本文中研究的置信度应用 [123]，它同样可以用于加速鉴别性解码 [20]，系统融合 [124]，鉴别性训练 [22]，等等。

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。

为了较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法¹。

4.5.2 词图剪枝

并行词图剪枝算法基于引文 [126, 116] 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，以减小同步时间消耗。

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消耗较大。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 4.5 章中的做法。这种实现方法的内存消耗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：

4.6 实验结果

4.6.1 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 [104] 中的“TDNN-LSTM-C”模型，其使用无词图鉴别性训练准则 (LF-MMI) [90]，包含

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

算法 4-3 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)

```

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 [78]，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¹。

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) [117]。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) [127]。Normalized cross entropy (NCE) [128] 被用于评估词图得出的置信度的质量。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 [129, 130]。实时率 (RTF) 用于度量解码器速度和相应针对 CPU 基线系统的加速比 (Δ)。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

¹Kaldi 中的 *decode-faster-mapped* 和 *latgen-faster-mapped* 工具

GPU 解码，后续我们还会比较其他 GPU 架构下的性能。

4.6.2 性能和速度

表 4-1 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 4-1 1-best 和词图性能比较 (beam=14).

system	lat. den.	WER	+rescored	OWER	NCE
CPU	30.3	15.5	14.3	11.2	0.322
GPU	30.2	15.5	14.3	11.2 ¹	0.328

解码的实时率和加速比的比较参见表 4-2。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 [118]。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。

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 0.03²。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。

4.6.3 分析

图 4-3 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

²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 [113]，我们以往的经验是有 10 倍左右的加速比。

表 4-2 所提出系统的速度比较 ($beam=14$)。

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

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 [105, 107]。而这项技术在 GPU 解码器中仍然很重要。

- 语言模型大小和内存消耗。

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 [41] 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。

- Profiling.

图 4-4 比较了 CPU 和 GPU 的各个子模块的时间占用²。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。

²词图生成在 CPU 中被包含到令牌传递部分中

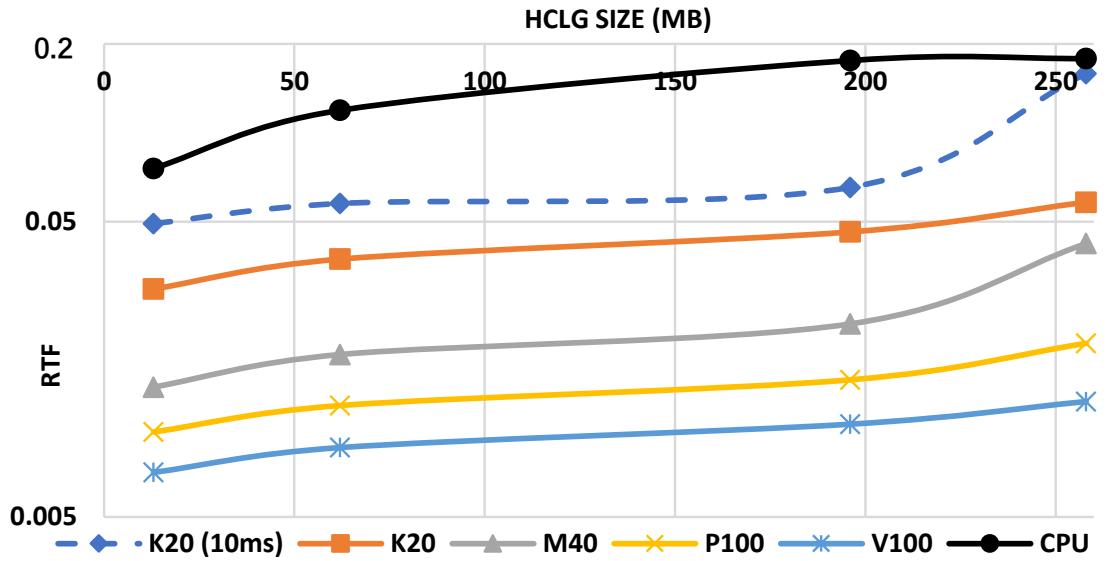


图 4-3 语言模型大小，帧率，GPU 架构的比较

4.7 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。

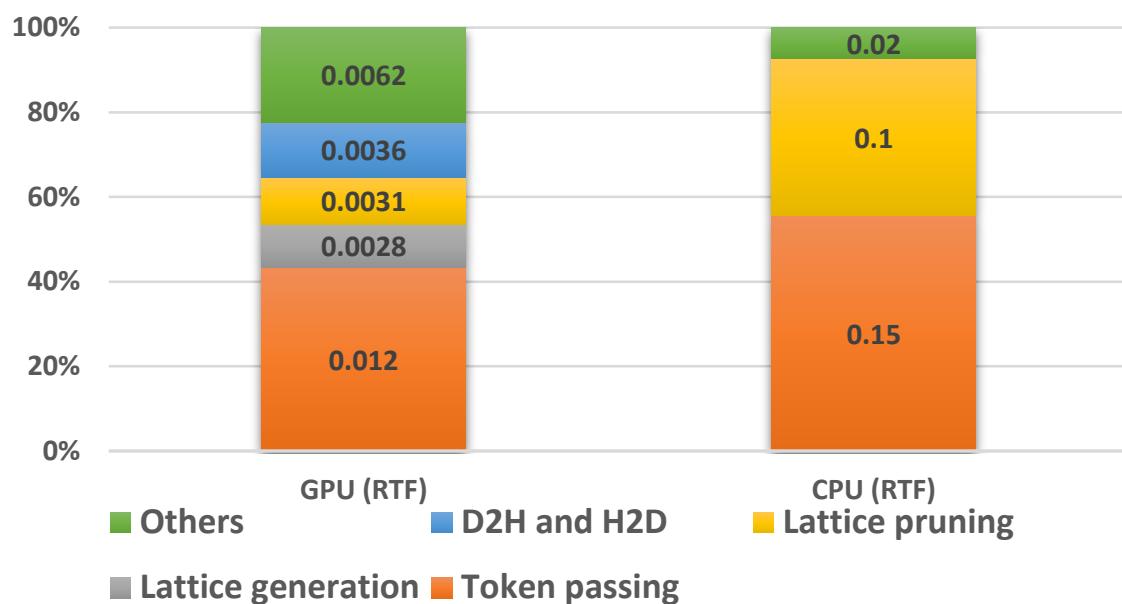


图 4-4 针对解码器的时间占比分析

第五章 基于标签同步解码的搜索空间优化

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) 和连接时序模型 (Connectionist Temporal Classification, CTC)。针对这些模型，当前主流的推理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。该章节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。

5.1 引言

序列标注问题是指一类将给定的数据序列转化为标签序列的任务 [131]，如自动语音识别 (automatic speech recognition, ASR) 和手写体识别 (optical character recognition, OCR) 等。这类问题与传统模式识别框架不同的是，给定的样本中各数据点不符合独立同分布 (independent and identically distributed, i.i.d) 假设。其主要问题在于特征向量序列的可变长性，如 ASR 中由语速变化导致的语音信号时长的不同。

为了对上述时序特征进行建模，人们提出了序列模型。根据其建模过程，序列模型可以分为以下两类：1) 生成式序列模型 (generative sequence models, GSM)，如隐马尔科夫模型 (hidden Markov model, HMM)；2) 判别式序列模型 (discriminative sequence models, DSM)，如连接时序模型 (connectionist temporal classification, CTC) 等。对于 GSM，在序列鉴别性训练时，需要在序列层面使用贝叶斯定理来由条件似然度推导出序列后验概率；而 DSM 则可以直接推导和优化序列后验概率。

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 [40]) 和深度神经网络 (deep neural network, DNN [94])；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这样的模型 [42, 41]。

在推理阶段，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 [42]，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 [41] 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。

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 [78, 30]。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 [132, 133, 134, 135, 93] 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 [133]，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。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，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。

最终在实验部分，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。

5.2 基于连接时序模型的标签同步解码

5.2.1 语音识别解码算法研究现状分析

5.2.1.1 序列标注与序列模型

序列标注包括所有将数据特征序列转化为标签序列的任务 [131]，本节以 ASR 为例进行简要介绍。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。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。

(1) 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。

(2) 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。

为了对上述序列相关性这一特征进行建模，人们提出了序列模型。根据其建模过程，序列模型可被分为生成式序列模型 (GSM) 和判别式序列模型 (DSM)。

生成式序列模型是通过计算给定标签序列时特征序列的概率 $p(\mathbf{x}|\mathbf{l})$ 来定义的。该模型通过使用贝叶斯方法引入人类语音产生物理过程的先验知识，来提供时序和长度约束。HMM 因其作为生成式序列模型来表征人类语音声学特征的能力，而成为 ASR 的

流行建模方法。在神经网络-隐马尔科夫模型 (neural network-hidden Markov model, NN-HMM) 混合系统中, HMMs 用来对语音信号的动态变化进行建模, 而观测概率则通过神经网络来进行估计。

$$\begin{aligned} p(\mathbf{x}|\mathbf{l}) &= \sum_{\pi \in \mathcal{A}(\mathbf{l})} p(\mathbf{x}|\pi) = \sum_{\pi} \prod_{t=1}^T p(\mathbf{x}|\pi_t) P(\pi_t|\pi_{t-1}) \\ &= \sum_{\pi} \prod_{t=1}^T p(\pi_t|\mathbf{x}) \frac{P(\pi_t|\pi_{t-1})}{P(\pi_t)} \end{aligned} \quad (5-1)$$

其中 \mathbf{l} 是生成式序列模型的标签序列, 如上下文相关 (context dependent, CD) 的音素序列。 π 是 HMM 状态序列, π_t 是第 t 帧对应的 HMM 状态, $\pi_s^{(l)}$ 是指第 l 个 HMM 模型的第 s 个 HMM 状态。 $P(\pi_t|\pi_{t-1})$ 是 HMM 状态转移概率, $P(\pi_t)$ 是 π_t 的状态先验概率。 \mathcal{A} 是指从标签序列 \mathbf{l} 到其相应 HMM 状态序列 π 的映射函数, 如下所示。

$$\mathcal{A} : L \mapsto \{\pi_1^{(1)}, \dots, \pi_5^{(1)}, \dots, \pi_5^{(|L|)}\} \quad (5-2)$$

L 是标签序列 \mathbf{l} 的各个单元的集合。其中每个标签序列单元对应一个 HMM 模型, 而每个 HMM 模型对应五个 HMM 状态, 如图 5-1(a) 中所示。状态后验概率 $p(\pi_t|\mathbf{x})$ 可通过神经网络进行估计得出。

而判别式序列模型则是直接计算给定特征序列 \mathbf{x} 时产生标签序列 \mathbf{l} 的后验概率 $p(\mathbf{l}|\mathbf{x})$ 。其中, 连接时序模型 (CTC) 用于解决未分割序列数据的标注问题, 它通过引入 `blank` 标单元, 实现对输入序列的任意一点的一对一输出标签预测。

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}(\mathbf{l})} p(\pi|\mathbf{x}) = \sum_{\pi} \prod_{t=1}^T p(\pi_t|\mathbf{x}) \quad (5-3)$$

其中 \mathcal{B} 为如下所定义的一对多映射:

$$\mathcal{B} : L \mapsto L \cup \{\text{blank}\} \quad (5-4)$$

\mathcal{B} 决定了标签序列 \mathbf{l} 以及 \mathbf{l} 对应的模型单元序列 π 的集合。如图 5-1(b) 所示, 通过在序列 \mathbf{l} 的每个标签单元 \mathbf{l} 之间插入一个可选的自循环 `blank` 单元进行映射。 $p(\pi_t|\mathbf{x})$ 则可使

用以特征序列 \mathbf{x} 为输入的循环递归神经网络 (recurrent neural network, RNN) 或长短时记忆神经网络 (long short term memory, LSTM) [81] 等估计得到。

通常, 如引言中所述, 为了有效利用帧级分类器如 GMM[40] 和神经网络 [94] 的建模效果, 减轻建模的稀疏性和增强泛化能力, 避免未经分解的模型因处理整个序列而导致的运行延时等问题, GSM 和 DSM 都被分解为帧层面上的训练, 本文接下来便对传统的帧同步解码进行介绍。

5.2.1.2 帧同步解码算法

在模型推理阶段, 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 [42], 即帧同步解码 (FSD)。FSD 框架中, 解码速率等于标注速率, 标注速率等于特征速率。

在大词汇量连续语音识别 (large vocabulary conversational speech recognition, LVCSR) 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:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} \{P(\mathbf{w})p(\mathbf{x}|\mathbf{w})\} = \underset{\mathbf{w}}{\operatorname{argmax}} \{P(\mathbf{w})p(\mathbf{x}|\mathbf{l}_w)\} \quad (5-5)$$

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:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} \left\{ \frac{P(\mathbf{l}_w|\mathbf{x})P(\mathbf{w})}{P(\mathbf{l}_w)} \right\} \quad (5-6)$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}} \left\{ P(\mathbf{w}) \max_{\mathbf{l}_w} \frac{P(\mathbf{l}_w|\mathbf{x})}{P(\mathbf{l}_w)} \right\} \quad (5-7)$$

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 [92] 并近似为:

$$P(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}(\mathbf{l})} \prod_{t=1}^T y_{\pi_t}^t \cong \max_{\pi \in \mathcal{B}(\mathbf{l})} \prod_{t=1}^T y_{\pi_t}^t \quad (5-8)$$

其中, \mathcal{B} 的定义见公式 5-4。因此, 公式 5-7 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)。这里, 整体优化搜索空间-WFST, 在每一帧都需要被遍历。

$$\mathbf{w}^* \cong \operatorname{argmax}_{\mathbf{w}} \left\{ P(\mathbf{w}) \max_{\pi \in \mathcal{B}(\mathbf{l})} \frac{1}{P(\mathbf{l}_w)} \prod_{t=1}^T y_{\pi_t}^t \right\} \quad (5-9)$$

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 可表示为：

$$C \propto T \cdot |L'| \cdot |W| \quad (5-10)$$

其中 T 是语句中帧的数量， L' 是模型单元的集合，W 为词汇量。

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。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，接下来作者将对该框架及其应用进行详细介绍。

5.2.2 基于端到端建模的标签同步推理

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。

5.2.2.1 标签同步解码

在测试阶段，在基于音素的 CTC 模型中，从公式 5-5 可以推导出公式 5-7。而根据 CTC 中输出标签之间的条件独立性假设， $P(\mathbf{l}|\mathbf{x})$ 可以如下获得：

$$P(\mathbf{l}|\mathbf{x}) = \prod_{l \in \mathbf{l}} P(l|\mathbf{x}) \quad (5-11)$$

因此在标签级别上，维特比搜索如下所示：

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} \left\{ P(\mathbf{w}) \max_{\mathbf{l}_w} \frac{\prod_{l \in \mathbf{l}_w} P(l|\mathbf{x})}{P(\mathbf{l}_w)} \right\} \quad (5-12)$$

在 $P(l|\mathbf{x})$ 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。其中公共 blank 帧的集合定义如下：

$$U = \{u : y_{\text{blank}}^u > \mathcal{T}\} \quad (5-13)$$

其中 y_{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。

$$\begin{aligned} P(l|\mathbf{x}) &= \sum_{\pi \in \mathcal{B}(l)} \prod_{\pi} P(\pi|\mathbf{x}) \\ &\simeq \sum_{\pi \in \mathcal{B}(l)} \prod_{\pi \in U} y_{b_l}^u \prod_{\pi \notin U} y_{p_l}^u \end{aligned} \quad (5-14)$$

由于 $\prod_{\pi \in U} y_{b_l}^u \simeq 1$ ，公式 5-14 可被推导为 5-15：

$$P(l|\mathbf{x}) \simeq \sum_{\pi \in \mathcal{B}(l)} \prod_{\pi \notin U} y_{p_l}^u \quad (5-15)$$

5.2.2.2 标签同步解码算法实现

DSM 的标签同步解码算法如算法 5-4 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 [136]。

算法 5-4 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  $\neg isBlankFrame(F)$  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( $\hat{B}$ )

```

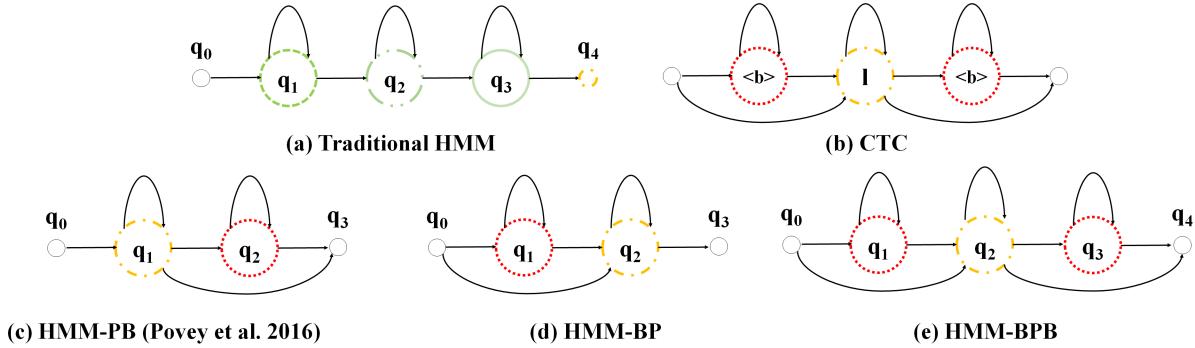


图 5-1 HMM, CTC 和本文提出的方法中隐藏状态拓扑结构示意图。在后面三种拓扑结构中, B 指 blank HMM 状态, P 指标签输出 HMM 状态。每个圆圈代表一个由神经网络建模发射概率的 HMM 状态。其中, 点划线圆圈表示输出标签建模, 如 (b) CTC 中的 l , 都各自分配一个特定的模型单元。虚线圆圈表示 blank 建模, 但并不完全相同, 如 (b) CTC 中的 $< b >$ 是使用公共的 blank 建模; 但 (c) 中的 q_2 , 每个输出标签有独立的 blank 建模, 本文将详细比较不同 blank 的粒度和拓扑结构所带来的区别。其它实线小圆圈, 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。

5.3 基于无词图鉴别性训练模型的标签同步解码

5.3.0.1 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:

$$P(\mathbf{x}|l) = \prod_l P(\mathbf{x}|l) \quad (5-16)$$

类似地, 在标签级别上进行的维特比搜索如下所示。

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} \left\{ P(\mathbf{w}) \max_{\mathbf{l}_w} \prod_{l \in \mathbf{l}_w} P(\mathbf{x}|l) \right\} \quad (5-17)$$

在标签中, $P(\mathbf{x}|l)$ 的计算如下所示:

$$\begin{aligned} P(\mathbf{x}|l) &= \sum_{\pi: \pi \in L', \mathcal{A}(\pi_{1:T})=l} \prod_{t=1}^T P(\mathbf{x}|\pi_t) P(\pi_t|\pi_{t-1}) \\ &= \sum_{\pi: \pi \in L', \mathcal{A}(\pi_{1:T})=l} \prod_{t=1}^T P(\pi_t|\mathbf{x}) \frac{P(\pi_t|\pi_{t-1})}{P(\pi_t)} \end{aligned} \quad (5-18)$$

最近, 研究人员们提出了一些新的 HMM 拓扑结构 [90, 105], 它们具有与式 5-4 中 CTC 的 \mathcal{B} 函数类似的一对多映射。以文献 [90] 为例, 每个 CD 音素由两个状态建模, 如

图 5-1(c) 所示, 且转移概率设置为常数值 0.5, 因此在式 5-18 中可被省略。具体来说, 其中一个状态模拟 **blank** 建模, 如图 5-1 (b) 中的 $\langle b \rangle$, 另外一个状态则模拟输出标签单元, 如图中的 **l**。不同之处在于文献 [90] 中的每个 CD 音素都保留了自己的 **blank** 版本。因此 HMM 中的状态由标签输出状态或者与 CTC 类似的 **blank** 状态组成。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 (5-13) 和 (5-14) 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 π_t 是帧 **t** 的推理模型单元。由于这些模型中的模拟 **blank** 状态, 公式 5-17 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:

$$U = \{u : \sum_{l \in L} (y_{p_l}^u - y_{b_l}^u) > T\} \quad (5-19)$$

其中 $y_{p_l}^u$ 是帧 **u** 处标签输出状态 **l** 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:

$$\begin{aligned} P(\mathbf{x}|l) \simeq & \sum_{\pi: \pi \in L', A(\pi_{1:T})=l} \{ \\ & \prod_{\pi \notin U} \frac{y_{b_l}^u P(b_l|\mathbf{x})}{P(b_l)} \prod_{\pi \in U} \frac{y_{p_l}^u P(p_l|\mathbf{x})}{P(p_l)} \} \end{aligned} \quad (5-20)$$

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。上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。我们尝试对 FSD 和 LSD 进行对比。

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:

$$\mathbb{C} \propto (T - |U|) \cdot |L'| \cdot |W| \quad (5-21)$$

其中空白帧的数量 $|U|$, 总是接近于 **T**。对比式 5-10 和式 (5-21), 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 [137] 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。

5.3.0.2 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 5-5 所示。与算法 1 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。

算法 5-5 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( $\hat{B}$ )

```

本文将图 1 (d-e) 所示的几种改进的 HMM 拓扑结构应用在了 GSM 中。具体来讲，在图 5-1 (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 bank (global blank)。此外，鉴于标签延迟带来的性能改进 [132]，图 1 (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 5-1 (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。

本文在维特比搜索中除了使用传统的束剪枝算法 [42] 和直方图剪枝算法 [138] (自适应束剪枝 [139]) 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 [140] 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：

$$\mathcal{L}(\theta) = -(p_\theta(\pi|\mathbf{x}) - \beta H(p_\theta(\pi|\mathbf{x})) \quad (5-22)$$

其中 $H(\cdot)$ 是输出分布 $p_\theta(\pi|\mathbf{x})$ 的熵， β 是正比例因子。与文献 [140] 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。

5.4 基于标签同步解码的端到端语音识别

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，但却不能系统地使用声学和文本数据，在一个系统框架里。与之相对，传统基于深度学习的隐马尔可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。

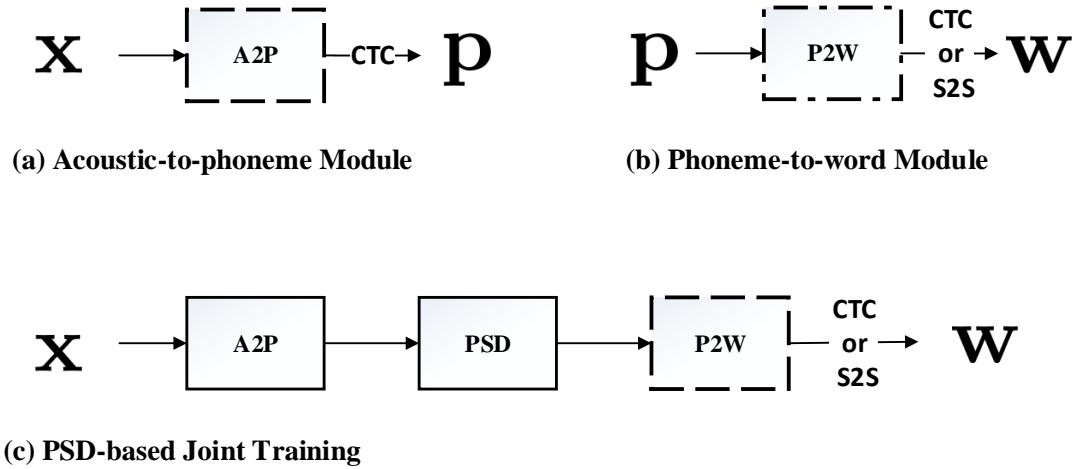


图 5-2 模块化训练策略的框架。实线框表示模型参数固定不变的部分。虚线部分和点划线部分分别表示模型参数使用声学或者文本数据进行训练。

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。音素感知是人类感知的基础单元 [141]。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 [142]。

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。

5.4.1 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 5-2 显示了这一框架。

这一端到端词序列识别系统被模块化如下：

$$P(\mathbf{w}|\mathbf{x}) \approx \max_{\mathbf{p}} [P(\mathbf{w}|\mathbf{p}) \cdot P(\mathbf{p}|\mathbf{x})] \quad (5-23)$$

公式中 w , p 和 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 准则训练了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) [107]。

$$P(\mathbf{w}|\mathbf{x}) \approx \max_{\mathbf{p}} [P(\mathbf{w}|\mathbf{p}) \cdot PSD(P(\mathbf{p}|\mathbf{x}))] \quad (5-24)$$

在解码阶段，经过联合训练后的模型将进行端到端推理，因此其复杂度与普通端到端模型相同 [108]。

对于 CTC，我们将最佳的推理序列连接起来，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 [107] 还可以进一步用于加速系统。

5.4.2 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 5-2(a) 所示，其与传统音素 CTC 建模相同 [143]。注意的是，虽然 CTC 被用于该项工作，其他传统声学模型同样可以被使用，比如 RNA [144] 或者 LF-MMI [90]。

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，推理词序列，即 $P(\mathbf{w}|\mathbf{p})$ ，也就是一个音素到词的模型，如图 5-2(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，并隐含进行文本切块。
- 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 wb 到音素列表中，以改善文本切块效果。 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 wb 作为文本快切分的线索，比如它可以区分一些短词可能是长词前缀的情况。

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) 系统地结果了声学和文本数据的使用来强化模型训练。

5.4.3 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 [107].

5.4.4 联合训练

最终，所有模块将会被堆叠起来，如图 5-2(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 [133]。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 [143, 135]. 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上，保留了优点。

5.5 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 [145]，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 [116] 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 [90] 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 [95] 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 [78]；使用 EESEN [146] 作为训练工具，训练过程与文献 [143] 相似。

本文的 GSM 实验是在一系列由 KALDI 流程 [115] 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 [105, 90]，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 [108]，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 [114, 147]。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 [148] 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 (Δ) 作为参考。

5.5.1 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 [149] 曾进一步对比了 CTC 模型的性能和基于 HMM 的系统的性能。

表 5-1 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 5-3 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。

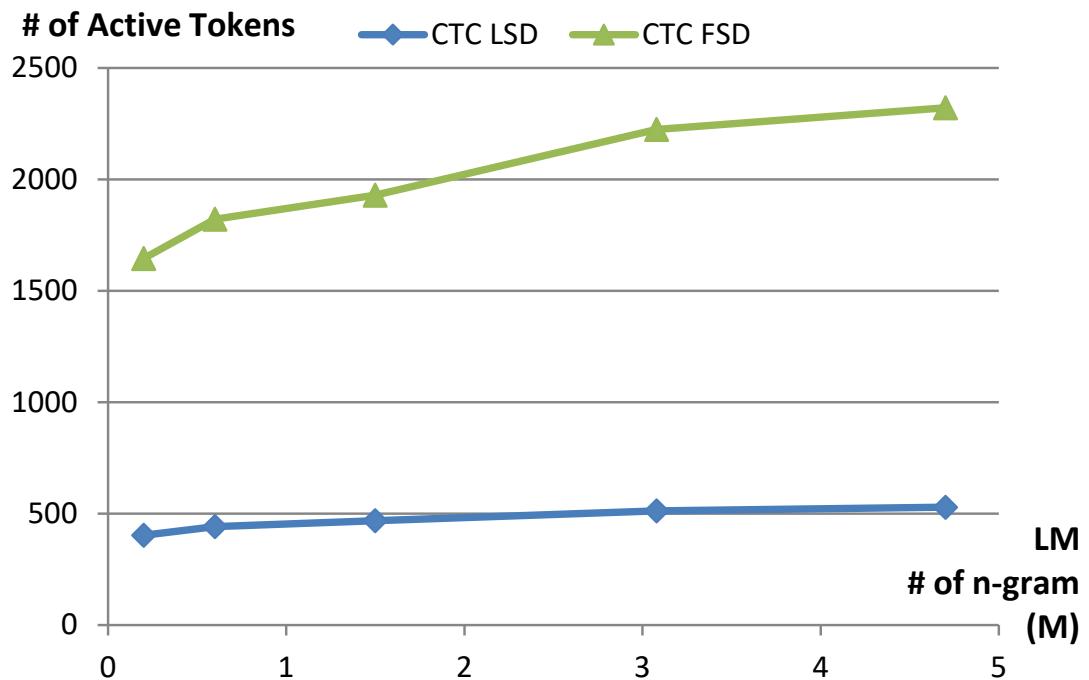


图 5-3 LSD 和 FSD 框架中平均活跃令牌数随 LM 变大的变化趋势。（为清晰起见，这里仅绘制 swb 子集，callhm 子集具有类似变化趋势。）

(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。

实现方法类似于文献 [150]，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；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 [105] 中的观察结果一致，并且在 LSTM-HMM[150] 和 DNN-HMM[137] 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。

表 5-2 LSD 与跳帧方法的对比

测试子集	解码性能		搜索加速		
	FSD→FS+LSD		FSD→FS→FS+LSD		
	WER	Δ(%)	SRTF	Δ _{FS}	Δ _{+LSD} (Σ)
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5-4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5-13和5-19中定义的 T 来调整加速比和性能，这也可以说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。

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5-12及公式5-13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5-4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5-12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。

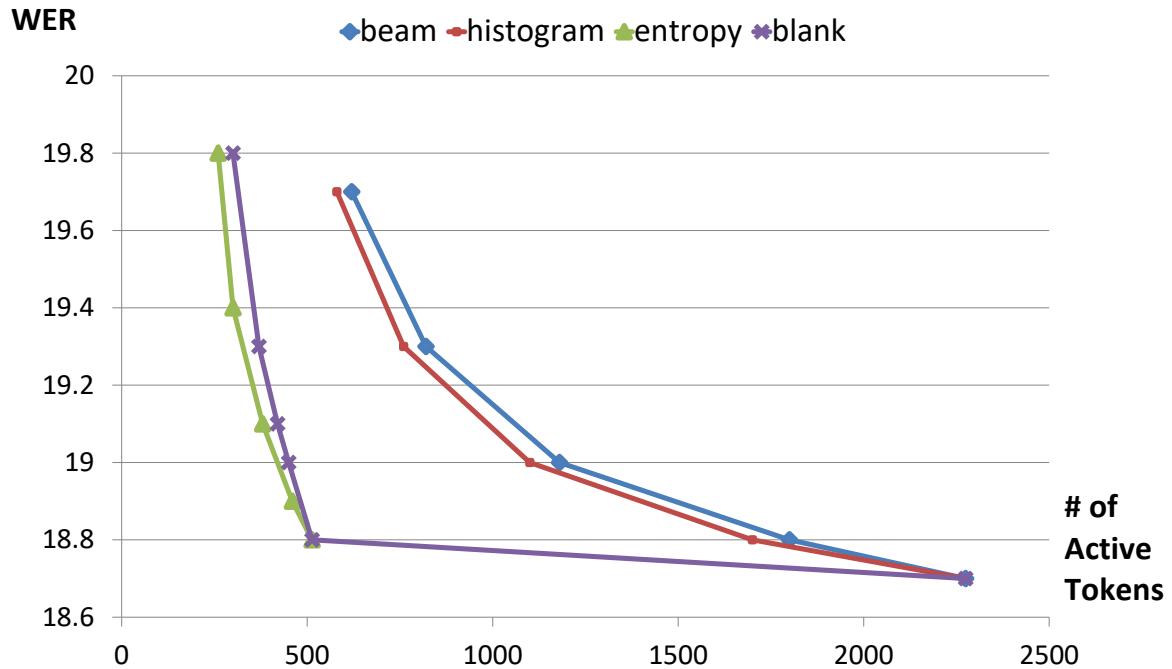


图 5-4 对于 swb 子集，CTC 中，使用不同剪枝技术时 WER 随平均活跃令牌数的变化趋势。calhm 子集结果类似

5.5.2 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5-3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5-1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[105]（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5-2相比，加速比也略小，原因是这些模型的推理分布概率不像 CTC 那样尖锐。如何在 GSM 中获取更尖锐的推理分布概率将在该部分的 (2) 和 (3) 中进行讨论。

具体地，如表 5-3 所示，第一行列出了文献 [105] 中提出的低帧率模型（LFR）的结果；第二行是使用 LF-MMI 准则 [90] 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 [151] 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[152] 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。

表 5-3 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 Δ(%)	FSD→LSD	SRTF Δ(%)	FSD→LSD	#AT Δ(%)
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

(2) 候选序列剪枝：如图5-5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5-4中一个区别在于，beam, histogram, blank, entropy 在图5-5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。

(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 5-4 生成式序列模型中的 blank 粒度

系统	blank	性能		速度			
		FSD→LSD	WER Δ(%)	FSD→LSD	SRTF Δ(%)	FSD→LSD	#AT Δ(%)
TDNN	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

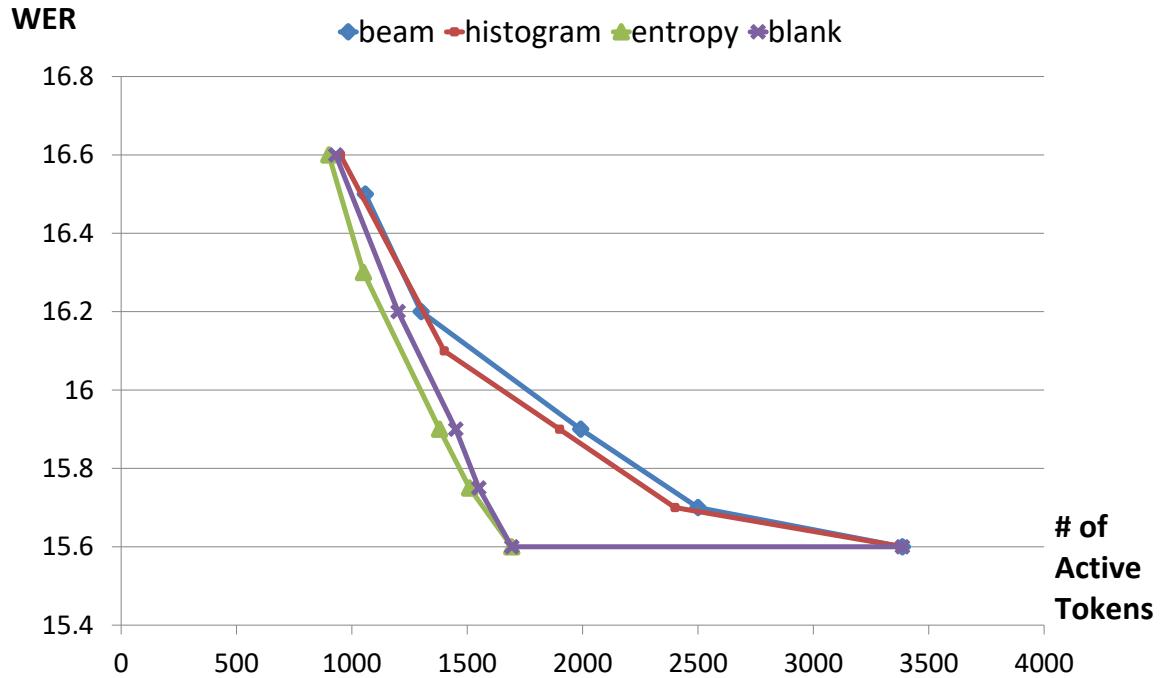


图 5-5 LF-MMI 中，使用不同剪枝技术时 WER 随平均活跃令牌数的变化趋势

表 5-5 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 5-5 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：43% → 47% → 49%。最后，phone blank 是基于 GSM 的 LSD 框架的最佳选择。

表 5-5 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 [132] 中观察到的现象，这使得模型能更可靠地推断标签输出状

表 5-5 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	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

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可夫模型输出更多的 blank 状态。最后，与表 2 中 CTC 的结果相比，GSM 中 LSD 框架能减少 49% 的搜索时间。

5.5.3 标签同步解码在模块化训练中的应用

表 5-6 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 5-6 各个模块的性能比较

模块	模型	推理单元	词边界	PER/WER CV (%)
A2P	CTC	phoneme	×	13.0
			√	12.0
P2W	CTC	word	×	16.0
			√	4.3
	S2S	word	×	13.9
			√	2.8

在 A2P 中，音素 CTC 的识别性能与 [92] 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 [93]。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 5-7 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 [108]。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 [108] 中的系统包含相对 20-30% 的差距。

表 5-7 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¹，这里的性能差距类似于之前在 [108] 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 [108]。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 5-8 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 5-8 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。在其他一些研究

¹在更多数据情况下，CTC 性能比传统系统好 [135]。我们之前的研究也显示同样的结论 [107]

中 [93] 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。

表 5-8 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
	✓	30	5851	24.9	36.5

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 5-7 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 5-7 中最后一行。不同于在表 5-6 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他研究 [153]。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。

5.6 本章小结

在本章中，我们针对语音识别和端到端建模的忒单，提出了一系列标签同步解码算法，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。

第六章 基于标签同步解码的统一解码框架

6.1 基于标签同步解码的置信度框架

6.2 基于标签同步解码的多识别任务统一框架

6.3 实验结果

6.4 本章小结

第七章 关键词检测的序列建模和标签同步解码

7.1 基于深度学习的关键词检测及其置信度

7.2 关键词检测的序列鉴别性训练

7.3 关键词检测的标签同步解码

7.4 实验结果

7.5 本章小结

第八章 全文总结

本论文的主要贡献在第章、第章中介绍。第一个贡献是。。。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，概述于章节和章节。

8.1 基于 GPU 并行计算的搜索速度优化

8.2 基于标签同步解码的搜索空间优化

8.3 基于标签同步解码的统一解码框架

8.4 关键词检测的序列建模和标签同步解码

8.5 后续工作展望

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] Hervé Bourlard and Nelson Morgan. “*A continuous speech recognition system embedding MLP into HMM*.” In: *Proc. Neural Information Processing Systems (NIPS)*, **1989**: 186–193.
- [25] 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.
- [26] Hervé Bourlard and Nelson Morgan. *Connectionist speech recognition: a hybrid approach*. Springer Science & Business Media, **2012**.
- [27] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer London, **2014**. <https://books.google.com/books?id=rUBTBQAAQBAJ>.
- [28] 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.
- [29] 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.
- [30] 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.
- [31] 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.
- [32] 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**.

- [33] 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.
- [34] 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.
- [35] Irving J Good. “The population frequencies of species and the estimation of population parameters”. *Biometrika*, **1953**, 40(3-4): 237–264.
- [36] 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.
- [37] 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.
- [38] Tomáš Mikolov, Martin Karafiát, Lukas Burget *et al.* “Recurrent neural network based language model.” In: *Proc. Annual Conference of International Speech Communication Association (INTERSPEECH)*, **2010**: 3.
- [39] Tomáš Mikolov. “Statistical language models based on neural networks”. *Presentation at Google, Mountain View, 2nd April, 2012*.
- [40] 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.
- [41] Mehryar Mohri, Fernando Pereira and Michael Riley. “Weighted finite-state transducers in speech recognition”. *Computer Speech & Language*, **2002**, 16(1): 69–88.
- [42] G David Forney. “The viterbi algorithm”. *Proceedings of the IEEE*, **1973**, 61(3): 268–278.
- [43] 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.
- [44] Hynek Hermansky. “Perceptual linear predictive (PLP) analysis of speech”. *the Journal of the Acoustical Society of America*, **1990**, 87(4): 1738–1752.

- [45] 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.
- [46] 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.
- [47] Nagendra Kumar and Andreas G Andreou. “*Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition*”. *Speech Communication*, **1998**, 26(4): 283–297.
- [48] 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.
- [49] 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.
- [50] 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.
- [51] 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.
- [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] 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.
- [54] 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.

- [55] 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.
- [56] 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.
- [57] Andrew Viterbi. “*Error bounds for convolutional codes and an asymptotically optimum decoding algorithm*”. *IEEE Transactions on Information Theory*, **1967**, 13(2): 260–269.
- [58] Steve J Young, Gunnar Evermann, Mark JF Gales *et al.* “*The HTK book*”. Cambridge university engineering department, **2002**, 3: 175.
- [59] 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.
- [60] 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.
- [61] Hagen Soltau and George Saon. “*Dynamic network decoding revisited*”. In: *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, **2009**: 276–281.
- [62] David Rybach, Ralf Schütter and Hermann Ney. “*A comparative analysis of dynamic network decoding*”. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing*, **2011**: 5184–5187.
- [63] 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.
- [64] 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.
- [65] Stephan Kanthak, Kai Schütter 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.

- [66] 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**.
- [67] 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.
- [68] 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.
- [69] 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.
- [70] Antonio Cardenal-López, F Javier Diéguez-Tirado and Carmen Garcia-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.
- [71] 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**.
- [72] 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**.
- [73] Janne Pylkkönen. “New pruning criteria for efficient decoding”. In: *Ninth European Conference on Speech Communication and Technology*, **2005**.
- [74] David E Rumelhart, Geoffrey E Hinton and Ronald J Williams. “Learning representations by back-propagating errors”. *NATURE*, **1986**, 323(6088): 533.
- [75] Yann LeCun, Léon Bottou, Yoshua Bengio *et al.* “Gradient-based learning applied to document recognition”. *Proceedings of the IEEE*, **1998**, 86(11): 2278–2324.

- [76] 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.
- [77] 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.
- [78] 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**.
- [79] 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**.
- [80] 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.
- [81] Sepp Hochreiter and Jürgen Schmidhuber. “*Long short-term memory*”. *Neural Computation*, **1997**, 9(8): 1735–1780.
- [82] Léon Bottou. “*Online learning and stochastic approximations*”. *On-line learning in neural networks*, **1998**, 17(9): 142.
- [83] Boris T Polyak. “*Some methods of speeding up the convergence of iteration methods*”. *USSR Computational Mathematics and Mathematical Physics*, **1964**, 4(5): 1–17.
- [84] Yann LeCun, Léon Bottou, Genevieve B Orr *et al.* “*Efficient backprop*”. In: *Neural networks: Tricks of the trade*. Springer, **1998**: 9–50.
- [85] Edmondo Trentin and Marco Gori. “*A survey of hybrid ANN/HMM models for automatic speech recognition*”. *Neurocomputing*, **2001**, 37(1-4): 91–126.
- [86] Hervé Bourlard and Christian J Wellekens. “*Links between Markov models and multilayer perceptrons*”. In: *Proc. Neural Information Processing Systems (NIPS)*, **1989**: 502–510.

- [87] 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.
- [88] 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.
- [89] 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**.
- [90] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez *et al.* “*Purely sequence-trained neural networks for ASR based on lattice-free MMI*”. *Interspeech 2016*, **2016**.
- [91] 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.
- [92] 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.
- [93] William Chan. *End-to-End Speech Recognition Models* [phdthesis], **2016**.
- [94] 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.
- [95] 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.
- [96] 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.
- [97] Albert S Bregman. *Auditory scene analysis: The perceptual organization of sound*. MIT press, **1994**.
- [98] DeLiang Wang and Guy J Brown. *Computational auditory scene analysis: Principles, algorithms, and applications*. Wiley-IEEE press, **2006**.

- [99] Martin Cooke, John R Hershey and Steven J Rennie. “*Monaural speech separation and recognition challenge*”. *Computer Speech & Language*, **2010**, 24(1): 1–15.
- [100] 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.
- [101] 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.
- [102] 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>.
- [103] 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.
- [104] 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.
- [105] Golan Pundak and Tara N Sainath. “*Lower Frame Rate Neural Network Acoustic Models*”. *Interspeech 2016*, **2016**: 22–26.
- [106] Zhehuai Chen, Wei Deng, Tao Xu *et al.* “*Phone Synchronous Decoding with CTC Lattice*”. In: *Interspeech 2016*, **2016**: 1923–1927.
- [107] 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.
- [108] Kartik Audhkhasi, Bhuvana Ramabhadran, George Saon *et al.* “*Direct Acoustics-to-Word Models for English Conversational Speech Recognition*”. *arXiv preprint arXiv:1703.07754*, **2017**.
- [109] Zhehuai Chen, Qi Liu, Hao Li *et al.* “*On Modular Training of Neural Acoustics-to-word Model for LVCSR*”. In: *ICASSP*, **2018**.

- [110] Takaaki Hori, Chiori Hori and Yasuhiro Minami. “*Fast on-the-fly composition for weighted finite-state transducers in 1.8 million-word vocabulary continuous speech recognition*”. In: *Eighth International Conference on Spoken Language Processing*, 2004.
- [111] David Nolden, Ralf Schlüter and Hermann Ney. “*Search space pruning based on anticipated path recombination in lvcsr*”. In: *Interspeech*, 2012.
- [112] Karel Veselý, Lukáš Burget and František Grézl. “*Parallel training of neural networks for speech recognition*”. In: *International Conference on Text, Speech and Dialogue*, 2010: 439–446.
- [113] 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.
- [114] Kisun You, Jike Chong, Youngmin Yi *et al.* “*Parallel scalability in speech recognition*”. *IEEE Signal Processing Magazine*, 2009, 26(6).
- [115] Daniel Povey, Arnab Ghoshal, Gilles Boulian *et al.* “*The kaldi speech recognition toolkit*”. In: *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.
- [116] 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.
- [117] Steve Young. “*The HTK book version 3.4. 1*”. In: 2009.
- [118] *CUDA Toolkit Documentation*.
- [119] Leslie Lamport. “*How to make a multiprocessor computer that correctly executes multiprocess programs*”. *IEEE transactions on computers*, 1979, (9): 690–691.
- [120] 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.
- [121] 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.

- [122] Ali M Alakeel. “*A Guide to dynamic Load balancing in Distributed Computer Systems*”. In: *International Journal of Computer Science and Network Security (IJCSNS, 2010)*.
- [123] 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*.
- [124] 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.
- [125] 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.
- [126] Andrej Ljolje, Fernando Pereira and Michael Riley. “*Efficient general lattice generation and rescoring*”. In: *Sixth European Conference on Speech Communication and Technology, 1999*.
- [127] Björn Hoffmeister, Tobias Klein, Ralf Schlüter *et al.* “*Frame based system combination and a comparison with weighted ROVER and CNC.*” In: *INTERSPEECH, 2006*.
- [128] Manhung Siu and Herbert Gish. “*Evaluation of word confidence for speech recognition systems*”. *Computer Speech & Language, 1999, 13(4)*: 299–319.
- [129] 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.
- [130] 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.
- [131] Alex Graves. “*Supervised sequence labelling*”. In: *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, **2012**: 5–13.
- [132] Dario Amodei *et al.* “*Deep Speech 2: End-to-End Speech Recognition in English and Mandarin*”. *arXiv preprint arXiv:1512.02595, 2015*.
- [133] 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*.

- [134] Ronan Collobert, Christian Puhrsch and Gabriel Synnaeve. “*Wav2letter: an end-to-end convnet-based speech recognition system*”. *arXiv preprint arXiv:1609.03193*, **2016**.
- [135] 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**.
- [136] 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.
- [137] 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.
- [138] Volker Steinbiss, Bach-Hiep Tran and Hermann Ney. “*Improvements in beam search.*” In: *ICSLP*, **1994**: 2143–2146.
- [139] 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.
- [140] Gabriel Pereyra, George Tucker, Jan Chorowski *et al.* “*Regularizing neural networks by penalizing confident output distributions*”. *arXiv preprint arXiv:1701.06548*, **2017**.
- [141] 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.
- [142] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster *et al.* “*Exploring the limits of language modeling*”. *arXiv preprint arXiv:1602.02410*, **2016**.
- [143] 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**.
- [144] 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.

- [145] 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.
- [146] 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.
- [147] 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.
- [148] 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.
- [149] 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.
- [150] 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.
- [151] Matthias Paulik. “Improvements to the Pruning Behavior of DNN Acoustic Models”. In: *Sixteenth Annual Conference of the International Speech Communication Association*, **2015**.
- [152] 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.
- [153] 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] CHEN H, CHAN C T. Acoustic cloaking in three dimensions using acoustic metamaterials[J]. Applied Physics Letters, 2007, 91:183518.
- [2] CHEN H, WU B I, ZHANG B, et al. Electromagnetic Wave Interactions with a Metamaterial Cloak[J]. Physical Review Letters, 2007, 99(6):63903.

致 谢

感谢所有测试和使用交大学位论文 **LATEX** 模板的同学！

感谢那位最先制作出博士学位论文 **LATEX** 模板的交大物理系同学！

感谢 William Wang 同学对模板移植做出的巨大贡献！