密级 _____

中国科学院大学
University of Chinese Academy of Sciences

博士学位论文

中文手写文本行识别

作者姓名 吴一超

指导教师 刘成林 研究员

中国科学院自动化研究所

联合指导教师 殷飞 副研究员

中国科学院自动化研究所

学位类别 工学博士

学科专业 模式识别与智能系统

培养单位 中国科学院自动化研究所

2017年12月

A Study on Handwritten Chinese Text Recognition

By

Yi-Chao Wu

Supervisor:

Prof. Cheng-Lin Liu

Prof. Fei Yin

A Dissertation Submitted to
University of Chinese Academy of Sciences
In partial fulfillment of the requirement
For the degree of
Doctor of Engineering

Institute of Automation
University of Chinese Academy of Sciences

December, 2017

独创性声明

本人声明所递交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确地说明并表示了谢意。

签名：吴一超 日期：2017年12月12日

关于论文使用授权的说明

本人完全了解中国科学院自动化研究所有关保留、使用学位论文的规定，即：中国科学院自动化研究所有权保留送交论文的复印件，允许论文被查阅和借阅；可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后应遵守此规定）

签名：吴一超 导师签名：[illegible] 日期：2017.12.12

摘　要

中文手写文本识别方法主要分为基于显式切分的方法和基于隐式切分的方法。虽然目前基于显式切分的方法占据主要地位，但是其中的语言模型和形状模型等各个模块有待改进。另一方面，基于隐式切分的递归神经网络识别框架可以克服显式切分识别系统过于依赖切分的缺陷，在拉丁语系文本识别中已经占据主导地位，但是在中文识别领域尚未显现优势。本文对两种不同的识别框架分别展开了深入的研究，贡献主要有以下几点：

- 将神经网络语言模型首次引入到基于过切分的中文手写字符串识别系统中。神经网络语言模型可以将词（字）从离散空间投影到一个连续空间中，并在该空间中对语言模型进行隐式的平滑以及序列概率的预测，从而可以建模高阶文法。实验中对神经网络语言模型在过切分识别系统中的作用进行了全面的评价，主要比较了前馈神经网络语言模型、递归神经网络语言模型和混合语言模型。结果表明，神经网络语言模型可以提升系统识别性能，混合递归神经网络语言模型可以得到最好的结果。
- 提出一种包含单字分类器、过切分以及几何模型在内的卷积神经网络形状模型用于过切分识别系统。其中，搭建了一个融入领域知识的15层卷积神经网络作为字符分类器；提出了一种基于学习的两步过切分方法，将传统的基于前景点可见性分析的方法与滑动窗卷积神经网络分类器相结合，使得召回率有了进一步的提升；并且将几何上下文模型从传统的分类器转换为基于卷积神经网络的模型。在中文手写文本行识别实验中，基于本模型的识别系统在标准数据集上得到了最高识别性能。
- 在基于递归神经网络的识别框架中，提出了一种可分离二维递归神经网络模块。与传统二维模块相比，该模块可以有效地提取多方向的信息，同时可以消耗更少的计算资源。基于这种二维模块，设计了更深的网络结构，并且改良了解码算法。实验结果表明，本方法的精度比之前的同类型方法有了显著的提升。

关键词： 中文手写文本行识别，神经网络语言模型，卷积神经网络形状模型，

递归神经网络识别框架

Abstract

Handwritten Chinese text recognition (HCTR) has been addressed by two groups of methods, explicit segmentation based ones and implicit segmentation based ones. Although over-segmentation based methods have been demonstrated success in handwritten character string recognition, the involved linguistic context model and shape models need enhancement. On the other hand, recognition methods using recurrent neural network (RNN) can overcome the defects of over-segmentation based methods, thus have dominated in text recognition of Latin scripts. However, such methods have shown less success in HCTR. In this thesis, we study into both two frameworks to attack the existing difficulties of HCTR. The contributions of this dissertation are summarized as follows:

- For modeling higher order dependency in linguistic context, we introduce two types of neural network language models (NNLMs) into the over-segmentation based recognition system, namely, feedforward neural network LMs (FNNLMs) and recurrent neural network LMs (RNNLMs). In NNLMs, history characters are projected into a continuous space to perform an implicit smoothing and estimate the probability of a sequence. We perform a comprehensive evaluation of NNLMs in HCTR and further propose hybrid NNLMs to improve the performance. Experimental results show that the NNLMs improve the recognition performance, and hybrid RNNLMs outperform the other LMs.
- We propose an over-segmentation based HCTR method using convolutional neural network (CNN) shape modules, namely, character classifier, over segmentation, and geometric context models. We build a 15-layer CNN incorporating the domain-specific knowledge as the character classifier. For improving over-segmentation, we adopt a two-stage CNN based over-segmentation method, which combines visibility-based foreground analysis and CNN-based sliding window classification. We also utilize CNN for modeling the geometric context models. Based on the proposed CNN shape models, we achieve new

benchmarks on two HCTR standard datasets.

- In the RNN-based framework, we propose a HCTR method using Separable Multi-Dimensional LSTM-RNN (SMDLSTM-RNN) modules. Compared with the traditional MDLSTM-RNN, SMDLSTM-RNN not only extracts contextual information in various directions for better modeling the context, but also consumes much less computation efforts and resources so that we can explore much deeper structures. Based on this effective module, we design much deeper network structures and modify the decoding algorithm. Experimental results show that the proposed method performs significantly better than the previous LSTM-based methods.

Keywords: Handwritten Chinese Text Recognition, Neural Network Language Model, Convolutional Neural Network Shape Models, RNN-based Recognition System

目　录

表　格

插　图

第一章　绪论

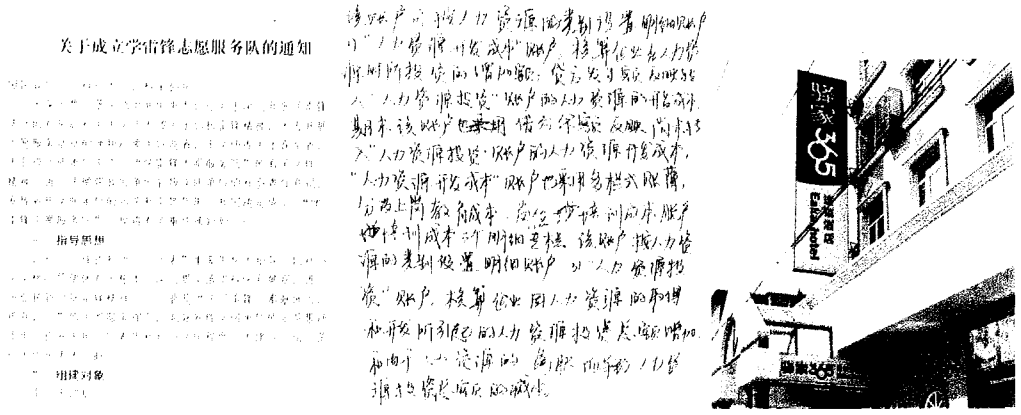
1.1　研究背景和意义

得益于深度学习算法、硬件计算能力以及互联网技术的迅速发展，人工智能正在对社会的发展和变革产生深刻的影响。智能金融、智能医疗、智能家居以及智能教育等概念的提出和落地，也在迅速改变传统行业的布局和规划，影响着国民经济生活的方方面面。2017年7月，国务院印发《新一代人工智能发展规划》，提出面向2030年我国新一代人工智能发展的指导思想、战略目标、重点任务和保障措施——举全国之力，在2030年一定要抢占人工智能全球制高点。可想而知，包含计算机视觉、语音识别和自然语言处理等各种技术手段在内的人工智能方法将扮演极其重要的角色，并极大地推动人类社会的发展。其中，光学字符识别（Optical Character Recognition，OCR）作为智能系统领域的一个重要分支和组成模块，有着极高的理论研究价值和应用前景。

文字作为一种普遍使用的交流工具，随着人们日常交往的日益增多，大量的文字信息也随之产生。在金融领域，保险、银行等公司单位每天都要将数以万计的保单和票据录入系统；在教育领域，学生提交的作业和试卷产生了大量的文档数据需要批改；在医疗领域，也有海量的病例亟待归档以便医生进行查阅。由于数据量的爆炸式增长，因此纸质文档数据的电子化已经是大势所趋，电子化之后的信息也更利于进一步的自动处理和分析。麦肯锡全球研究院在2017年1月推出的一份报告中称，金融和保险领域的工作，有43%的可能性会被自动化替代。其中，文字识别技术的发展将会起到关键作用。曾有公司做过统计，对于10000张保单，按最熟练的工作人员两分钟录入一张计算，如果该人每天满负荷工作八小时，需要42天时间完成；如果使用高识别率的识别引擎进行录入，在极少量的人工校对的参与下，只需要一到两天即可完成。这对于人力成本的节省程度可想而知。同样的，对试卷或者病例进行准确地识别，就成为自动阅卷和诊断的前提，进而促进智能教育和智能医疗产业的发展。

另一方面，在带来巨大的经济效益的同时，文字识别技术的发展也有助于保护我们的灿烂文化。中国具有源远流长的传统文化，是人类文明的重要发祥地之一。据统计，目前收藏在中国各图书馆之中的古籍（编纂出版于1912年

前）达2717.5万册，加之高等院校图书馆、文物保护部门、寺庙等单位的收藏，古籍总数超过3000万册。如何对先哲留下来的这些宝贵文献进行整理和研究一直是一个倍受关注的问题。由于数字化的文件具有易于保存、存储空间小和便于交流的特点，数字化成为了古籍整理和保护的重要手段和趋势。此外，在现今移动互联网时代，手机和平板的广泛使用使得图像和视频数据飞速增长。对于这些海量多媒体信息的审核，排除敏感和不良文字信息，也有助于社会的和谐稳定和健康发展。所以从文化角度上看，对于高性能文档识别（特别是中文文档）技术同样存在迫切的需求。



(a) (b) (c)

图 1.1: 中文文本图像示例。(a) 印刷体文本；(b) 手写文档；(c) 场景文本。

汉字识别是OCR技术的一个分支，中文文本识别按照字符的书写类型可以分为印刷体文本识别和手写体文本识别。印刷体文本（如图 1.1(a)）因为字形规范，所以识别精度很高。经过大量学者的多年研究，对于简单背景的印刷体文本识别技术已经相当成熟，比如汉王、文通以及FineReader等公司对印刷体文档分析都能提供令人满意的解决方案。然而，由于手写体中文文本识别存在：(1) 中文字符分类比较困难；(2) 文本行中字符粘连导致单个字符较难分割；(3) 书写内容复杂等三个问题，虽然经过多年的研究，还是不能达到令人满意的性能，一直被认为是一个非常具有挑战性的难题。中文手写文本识别一般又分为单字识别和字符串识别。单字识别尽管存在类别数大、字形复杂以及书写风格差别大等特点，但经过众多研究工作者的艰苦努力，识别精度已经超越了人类 [3, 4]。目前单字识别的研究问题主要集中在小样本学习和模型小型化上。与单字识别相比较，中文手写字符串识别的研究成果还不尽人意。除了需要更

好的上下文信息建模方式之外，对于涂抹、修改的拒识尚没有完整成熟的解决方法（如图 1.1(b)）。近年来，场景文字的识别（如图 1.1(c)）也渐渐成为研究和应用热点。与传统手写字符串识别相比，场景文字图片存在着背景复杂、视角多变以及形变更严重等难点。事实上，在接下来章节的阐述中，我们会发现在端到端框架下，两者在方法上愈来愈趋近一致，所以本文的研究重点还是集中在中文手写字符串识别上。

综上所述，中文手写字符串识别技术有着广阔的应用前景，对于国民经济的促进以及社会的稳定健康发展意义重大，同时又是一个极具挑战性的技术难题，具有重要的研究价值。

1.2 研究历史及现状

自1956年在达特茅斯学院举行的一次会议上正式确立人工智能研究领域以来，如何教计算机像人一样进行“识字”一直吸引着广大的研究者。此外，文字识别作为模式识别学科的一个传统研究问题，研究人员经常将此作为理论方法的实践检验 [3]。早期的文字识别研究主要集中在小类别集孤立字符的识别，比如数字和单个拉丁语系字母的识别，一般使用基于模板匹配和统计的方法。从20世纪80年代到90年代期间基于结构表示的方法 [5]开始出现，同时基于统计机器学习的方法也得到了大力发展。90年代以后，随着计算机计算能力的快速发展，基于统计的方法开始占据主导地位，基于特征提取和分类器相结合的框架被大量使用。作为字符识别技术的重要组成部分，在过去的50多年时间里，汉字识别技术有了长足的进步。在传统统计学习框架中，研究人员在字符归一化 [6, 7]、特征提取 [8, 9]以及分类器设计 [10–12]等方面做了大量的努力，取得了不错的成果，在自由书写的数据集上识别精度最高可以达到近95.0%的结果 [13]。近年来，随着深度学习技术的提出和不断进步，基于卷积神经网络 [14]的方法在性能上有了质的飞跃，随着网络结构的优化和训练方法的改良 [15–18]，目前可以达到近98.0%的识别精度，已经超越了人类在该数据集上识别的平均水平（96.13%） [4]。

然而，在实际的文档分析系统中，需要被识别的对象都是以连续手写的形式存在的。20世纪90年代开始，研究人员逐步转向文本行识别。这方面，对于拉丁语系文字，特别英文文本识别的研究开展的较早。早期的研究主要关注连续手写词识别 [19]，这期间的识别系统主要是面向特定领域的字符串识别，比如邮政地址识别 [20]，邮政编码识别 [21, 22]，以及银行支票和表单识别 [23]。

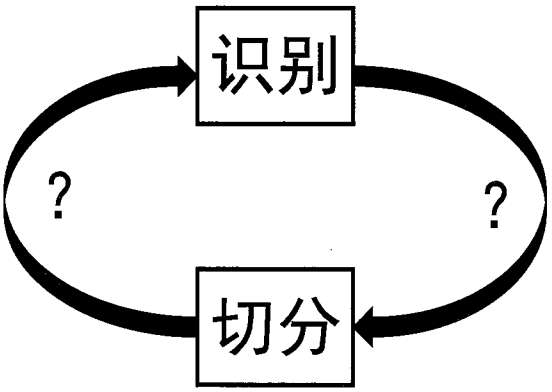


图 1.2: 文本行识别中的Sayre悖论。

这些系统的应用范围比较局限，一般只能识别限定字符集或特定的语法结构，但是为之后的无约束字符串识别奠定了一定的算法基础。大规模无约束字符串识别的主要难点在于切分和识别相互依赖，如图 1.2所示，成功的切分需要准确的识别，而准确的识别同时也需要精确的切分，这就构成了所谓的Sayre悖论（Sayre's paradox） [24]。为了解决该问题，研究人员将在语音识别中获得成功使用的隐马尔科夫模型（Hidden Markov Models，HMM） [25]应用到了脱机无约束手写字符串识别中 [26]。作为一种隐式切分的框架，HMM一般先将文本行图片以很窄的滑动窗从左到右遍历，从而得到特征序列，然后基于语句的HMM获得整个文本行的字符识别结果。其后的数十年，研究人员分别从状态建模 [27, 28]以及语言模型 [29]等方面对HMM进行了改良和优化，使其在著名的英文手写数据库IAM [30]上的词识别率从最初的不到50%提升至现今的90%以上。2009年，研究者Alex Graves等人提出一种新的基于连接时序分类（Connectionist Temporal Classification，CTC）的端到端训练识别框架 [31]，该框架在拉丁语系手写文本识别领域产生了巨大的影响。相比传统的HMM模型，该识别框架对于状态的建模更加灵活，可以捕获更多的上下文信息，而不再只依赖上一个时间状态，并且通过引入判别训练，使得性能进一步提升。该模型提出当年，将HMM能达到的最高64.5%的精度提升至74.1%。该方法在阿拉伯文 [32]、希腊文 [33]等各种语言上都获得了成功应用，并且随着工作的推进，通过引入二维信息和更深的网络结构 [34, 35]，系统的识别性能得到了进一步地提升。目前，使用基于CTC的框架，在IAM数据集上最高的性能可以达到91.0%左右的词识别正确率。近年来基于注意力机制的序列模型开始在语音

识别和机器翻译领域受到重视，也有学者将该机制用于文本识别 [36]，也能获得不错的识别结果，A2iA公司甚至使用该框架进行篇章级的文本识别任务 [37, 38]。

虽然对于中文手写文档的研究相对起步较晚，但是21世纪初也涌现出不少针对特定领域文本识别的工作 [1, 39, 40]，均取得了不错的结果。国内参与中文手写文本研究的单位主要包括清华大学、哈尔滨工业大学、华南理工大学以及中科院自动化研究所等。自2006年哈尔滨工业大学发布国际上首个自由书写的中文文本数据库HIT-MW [41]以后，中文手写文本识别研究者开始了真实书写文本识别的研究。在第十一届国际文档分析与识别大会（ICDAR 2011）上，中科院自动化研究所公布了更大规模的手写中文数据集CASIA-HWDB [42]，并且随后连续举办了了两届中文文本识别竞赛：ICDAR-2011 [43]以及ICDAR-2013 [4]，极大地促进了手写中文识别的研究发展。在短短的数年内，HIT-MW数据集上的识别性能从一开始的39.97% [44]提升到了91.86%，而在ICDAR-2013数据集上最高可以有89.28%的识别准确率 [45]。与拉丁语系文字不同，由于汉字结构复杂多变且类别数较大，HMM的建模十分困难，因而虽然不少研究者提出了一些基于HMM的中文字符串识别模型 [44, 46–48]，但是均没有能达到同时代的最好性能。在中文手写字符串识别中，基于显式切分识别的方法一直占据着主导地位。

在显式集成切分识别框架中，首先使用过切分算法将文本行图片切分成一系列的基元片段，每一片段对应一个汉字或者其中的一部分，因为每一个候选字符可以由连续的几个基元片段组合而成，组合不同的连续基元片段便构成了一个以候选字符模式为单位的切分候选网格。对于每一个候选字符模式，通过字符识别器能得到多个候选字符类别，从而进一步得到字符候选网格。该网格中每一条路径不仅代表了一种切分方式，同时也代表了一种识别结果，因此称为切分识别路径。最后，综合字符分类的置信度、语言上下文、几何上下文等各方面的信息给每一条切分识别路径进行打分，并且通过搜索算法选择出一条最好的路径作为识别结果。虽然文献 [45]通过融合多个上下文信息获得了当前最好性能，但是其整体性能还不足以满足文档电子化的需要，还有很大提升的空间。一般来说，影响一个过切分识别系统性能的因素主要包括过切分算法和路径评价方式。目前广泛使用的过切分算法都是利用汉字的几何形状、粘连、覆盖等特性使用启发式规则进行字符切分 [1]，并不能应对复杂粘连的情况，只有一小部分工作是着眼于基于自动学习得到切分模型。其中被成功应用于中

文手写字符串的方法也只能处理单粘连的情况 [49]，这极大地限制了系统的识别上限。另一方面，虽然基于深度学习方法的手写单字识别率已经超过人类，但是只有极少量的工作 [50]对基于卷积神经网络的分类器在串识别中的表现进行综合深入的研究。同时，尽管几何模型在许多工作中被证明可以提高识别精度 [45, 51, 52]，理论上也能利用深度模型对其性能进行提升，但是目前学术界并没有出现过这样的工作。此外，[45]中仅仅使用简单的回退语言模型，并不能很好地对高阶上下文信息进行有效的建模，这也是进一步提升过切分识别性能的关键因素。

2015年以来，基于递归神经网络的方法在手写中文文本识别上也已经有相关的工作。作为一种端到端的隐式切分识别框架，这种方法除了可以克服传统过切分框架中过切分的制约以及模型优化的一致性这两个问题以外，还具有弱标记学习的功能。相比于传统方法需要大量标记到字符级别的强标记样本，弱标记学习算法能极大地降低标注难度和节约成本，具有很大的应用潜力。虽然在联机中文手写文本识别中，这种新的识别方式已经超过传统方法 [53, 54]，但是该方法在脱机中文手写字符串中的研究应用还不是非常充分，只有一个工作可以得到和当前最好性能相媲美的结果 [55]，并且该工作需要经过其他语言文字的预训练等较为复杂的流程。此外，对于手写文本的识别，为了处理多方向的字符形变，一般都是用二维递归神经网络模块 [32]，但是这种结构在训练时存在着梯度易爆炸、不易并行优化等缺点。

回顾中文手写文本识别的发展史，随着研究问题不断深入，我们对于系统实用性能的要求也在不断增高。虽然已有上述前人的工作发表，但是仍存在较多不足，对于传统的过切分方法：

1. 由于手写内容千变万化，前人使用的语言模型捕获上下文信息的能力较弱，不能进行高阶语言模型的建模。
2. 中文手写字符串切分的手段比较单一，缺乏基于统计学习的算法，只有一小部分工作是着眼于基于自动学习得到切分模型，其中被成功应用于中文手写字符串的方法也只能处理单粘连的情况。
3. 尽管基于卷积神经网络的分类模型在单个字符分类上得到成功应用，但是只有极少数的工作证明其在串识别中的提升程度。同样的，尽管许多研究表明几何模型可以提高识别精度，但是迄今为止缺乏关于如何使用深度学习模型对几何模型进行改良的系统深入的研究工作。

对于新兴的基于递归神经网络的方法：

1. 现有的工作需要大量的其他语言样本进行预训练，而且使用的二维递归神经网络模块在训练时存在着容易梯度爆炸、不易并行优化等缺点，这极大地增加了训练的复杂度，如何简化训练流程并且加快训练速度是一个值得讨论的问题。
2. 作为一个深度模型，一般都需要大规模样本进行训练，而事实上样本标注工作总是需要耗费大量的人力物力，如何在样本较少的情况下达到甚至超越现今最好性能的系统是一个非常有价值的问题。

1.3 本文主要工作及贡献

针对前面提到的中文手写字符串识别研究工作的不足，本文对两种不同的识别框架分别展开了深入的研究。主要包含以下几点工作：

（一）神经网络语言模型

在文本识别系统中，语言模型对于性能的提升发挥了重大的作用，我们也总是希望能构建更高阶的文法以便更好地进行上下文建模。可是在实际情况中，传统回退语言模型由于估计的参数非常多，非常容易出现数据稀疏问题（也就是维数灾难问题），导致无法得到很好的高阶模型。为了克服上述问题，近来有学者提出了神经网络语言模型，从而有效缓解了数据稀疏问题。神经网络语言模型不但能够大幅降低模型的困惑度，而且已经被成功地应用在了语音识别、机器翻译和英文手写识别中。本文将神经网络语言模型首次引入到基于过切分的中文手写字符串识别系统中，通过神经网络语言模型，词（字）可以从离散空间投影到一个连续空间中，并在该空间中可以对语言模型进行隐式的平滑以及序列概率的预测，从而使得高阶文法建模成为可能。本文对神经网络语言模型在过切分识别系统中的作用进行了全面的评价，主要比较了前馈神经网络语言模型和递归神经网络语言模型。这两种语言模型还将与回退语言模型结合形成混合语言模型。实验表明，这种新型的语言模型可以提升系统识别性能，其中混合递归神经网络语言模型可以得到最好的结果。

（二）卷积神经网络形状模型

在本文中，单字分类器、过切分以及几何模型合称为形状模型，它们主要是从图像形状层面对文本行进行建模，发挥着十分重要的作用。虽然卷积

神经网络在汉字字符识别领域获得了许多令人鼓舞的成果，但是只有极少数的工作对其在串识别系统中的性能进行深入和系统的分析。本文搭建了一个15层CNN作为字符分类器，在设计的过程中，还考虑融入领域知识，所以除了原始图像之外，还生成了八方向非线性归一化图像作为额外的输入通道。由于卷积神经网络是一个判别分类器，在单字分类器的训练过程中，还需要加入非字符类以便网络显式地拒绝非字样本。对于过切分，学术界提出了很多方法处理粘连字符，但是大多数方法都是基于启发式规则，这就导致这些方法的泛化性难以令人满意。为此，本文提出了一种基于学习的两步过切分方法：首先使用基于前景点可见性分析的方法进行预切分，然后使用两类分类的卷积神经网络在预切分的文本联通部件上进行滑动，卷积网络用于判断窗口的中心点是否是一个合法的切分点。这种算法使得召回率有了进一步的提升，同时精度仅有微弱的下降。对于几何模型，尽管许多研究表明它可以提高识别精度，但是目前为止还没有工作使用基于深度学习的几何模型。本文将几何模型从传统的分类器转换为基于卷积神经网络的模型。传统几何模型中，我们一般人工提取一些书写特征（比如投影、轮廓等），再使用支持向量机或者二次判别函数进行分类。若使用卷积网络，如果只是简单地将归一化图像送入分类器将会损失书写信息。为了解决这一问题，本文使用多项式回归首先得到文本行的大致走向，然后动态调整候选模式图像的上下空白。

（三）基于新的二维递归神经网络结构的识别方法

基于递归神经网络的框架可以一定程度上克服过切分识别框架的不足，它已经在英文等拉丁语系文字中得到了成功的应用，但是在中文上还缺乏相应的研究，仅有的较为完善的工作还需要在其他语种的大规模数据集上进行预训练。在手写字符串识别中，学术界一般使用二维递归神经网络模块以便捕捉多方向上的字符形变，但是传统的二维结构在训练过程存在着容易梯度爆炸、难以并行计算的问题。为此，本文提出了一种可分离二维递归神经网络模块，与传统二维模块相比，该模块既可以有效地提取多方向的信息，同时可以减少计算资源的消耗并且也容易使用图像处理器进行加速。基于这种全新的二维模块，本文设计了更深的网络结构，并且改良了解码算法。为了缓解训练数据量不足的问题，本文还提出一种结合书写人模态分布的串级别样本合成算法。最终的实验结果表明，该方法的精度比之前的同类型方法有了显著的提升。

综上所述，本文的主要贡献在于以下几个方面：

1. 针对高阶语言模型建模过程中出现的数据稀疏和维度灾难等问题，本文首次将神经网络语言模型引入到基于过切分的中文手写字符串识别系统中。神经网络语言模型可以在连续空间进行隐式的平滑以及序列概率的预测，从而可以很大程度上克服传统高阶语言模型的不足。本文对神经网络语言模型在过切分识别系统中的作用进行了全面的评价，主要比较了前馈神经网络语言模型、递归神经网络语言模型和混合语言模型。实验表明，神经网络语言模型可以提升系统识别性能，混合递归神经网络语言模型可以得到最好的结果。
2. 考虑到包含单字分类器、过切分以及几何模型在内的形状模型在文本行图像建模中的巨大作用，本文提出一种卷积神经网络形状模型，将其融入过切分识别系统之后发现可以大幅提升系统性能。文本搭建了一个融入领域知识的15 层卷积神经网络作为字符分类器。对于过切分，本文提出了一种基于学习的两步过切分方法，将传统的基于前景点可见性分析的方法与滑动窗卷积神经网络分类器相结合，使得召回率有了进一步的提升。另外，本文将几何上下文模型从传统的分类器转换为基于卷积神经网络的模型。在中文手写文本行识别实验中，基于本模型的识别系统在标准数据集上得到了最高识别性能。相比之前的基准系统，可以将字符错误率相对降低近50%。
3. 基于递归神经网络的识别框架作为一种新兴的识别方法，在中文字符串识别中研究尚不充分。在脱机字符串识别中，一般使用基于二维递归神经网络模块的方法，可以相比一维序列建模方法有更好的效果。本文提出了一种可分离二维递归神经网络模块。与传统二维模块相比，该模块同样能够有效地提取多方向的信息，并且可以更有效地利用计算资源。基于这种二维模块，本文设计了更深的网络结构，并且改良了解码算法，还提出一种串级别样本合成算法。实验结果表明，本方法的精度比之前的同类型方法有了显著的提升。

1.4 本文组织结构

本文的组织结构如下：

第一章介绍中文手写字符串识别的研究背景和意义、研究历史和研究现状，并简要阐述本文的主要工作及贡献。

第二章回顾中文手写字符串中的两类主要的框架：基于显式切分的方法和基于隐式切分的方法，针对性地重点介绍近几年在深度学习浪潮的带动下，这两类方法的最新进展。

第三章介绍在过切分框架中引入神经网络语言模型以便进行高阶上下文建模的方法，对前馈神经网络语言模型和递归神经网络语言模型进行详细的评测研究，通过在公开数据集上的实验证明神经网络语言模型的有效性。

第四章介绍如何在过切分框架中引入卷积神经网络形状模型，具体阐述了基于深度模型的字符分类器、过切分算法以及几何模型的设计、训练方法和识别效果。

第五章介绍专门为中文手写字符串识别设计的一种可分离二维递归网络模块，以及基于该模块的新的网络结构。此外，还介绍了基于连接时序分类的改良解码算法以及串级别样本合成算法。最后使用新的框架在公开数据集上验证了算法的有效性，识别率可以和目前最好的系统相当。

最后一章对本文的主要工作进行了总结，并进一步展望未来的研究方向。

第二章　中文手写字符串识别方法概述

因为汉字的类别数巨大并且语法组合十分复杂，所以中文手写字符串识别一般使用基于切分的方法。根据切分方式的不同，识别框架可以分成基于显式切分和基于隐式切分两种。本章首先介绍基于显式切分的方法（又称过切分方法），主要包括基本流程、语言上下文模型以及形状模型（包含单字分类模型、几何上下文模型和过切分算法）等方面；然后介绍基于隐式切分的方法，主要包含基于隐马尔科夫模型的方法和基于递归神经网络的方法；最后对本章内容进行小结。

2.1　基于显式切分的方法

2.1.1　系统流程

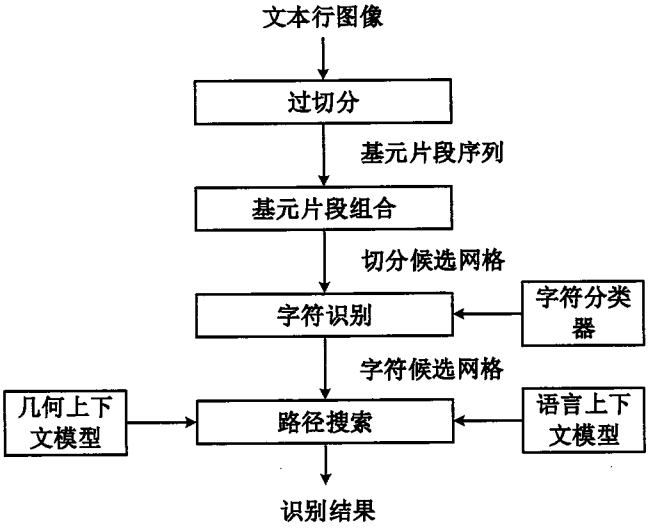


图 2.1: 显式切分识别框架。

文本的研究重点是文本行识别，因此假设文档页面已经完成图像预处理和文本行分割步骤。一个基于过切分的系统主要由文本行过切分、切分识别路径的构建以及融合上下文信息的路径搜索这三个模块组成，系统框图如图 2.1所示。具体地，一般分为以下四个步骤:

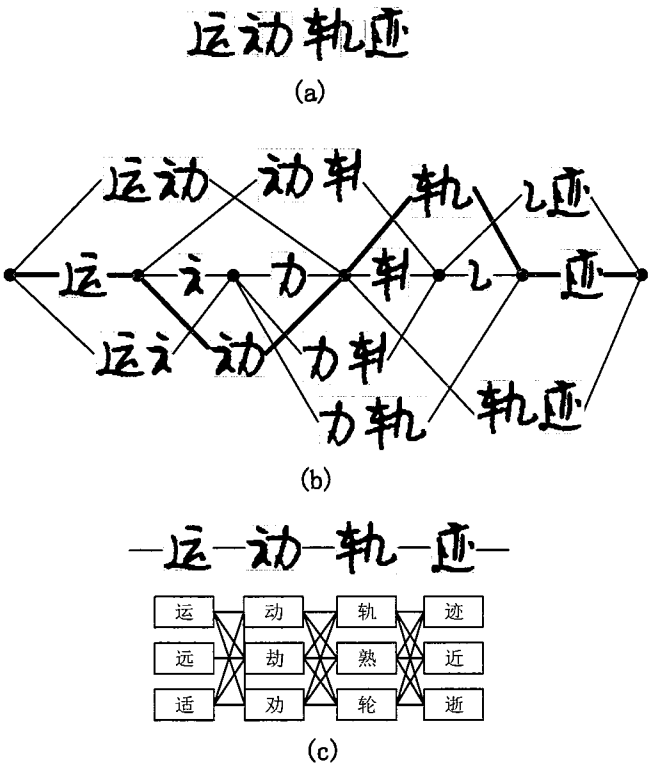


图 2.2: 手写中文文本行识别具体示例。(a) 文本行过切分；(b) 与(a)相对应的切分候选网格；(c) 与(b)中粗蓝线路径相对应的字符候选网格。

1. **文本行切分。**通过连通部件分析，文本行图像被切分为一系列的基元片段（如图 2.2(a)），这些基元片段理应对应一个字或者是字的一部分，但是在切分失败的情况下也有可能对应多个字，这会严重影响后续的路径搜索，所以这一步应尽可能保证召回率。

2. **基元片段组合。**将一个或者连续几个相邻片段进行组合，构成了切分候选网格。如图 2.2(b)所示，网格的每一条路径都对应其中一条候选切分路径，图中粗蓝线标记的路径为正确的切分候选路径。

3. **字符识别。**使用字符分类器，为每一条候选切分路径中的候选模式分配一定数量的候选类别，这样在一个候选切分路径中的所有候选模式就构成了候选字符网格。如图 2.2(c)所示，该候选字符网格对应图 2.2(b)中的粗蓝线切分候选路径。候选字符网格中的每一条路径不仅代表一条切分路径，同时也代表一种识别结果。

4. **路径搜索**。合并所有的候选字符网格便可以构成输入图像完整的切分识别网格，每一个候选单字序列结合其对应的候选字符模式序列（即字符切分候选路径）构成了一条候选切分识别路径。路径搜索就是利用合适的路径评价函数（融合语言上下文信息、字符识别器信息以及几何上下文信息）对每一条这样的路径进行评分，通过高效的搜索算法找到得分最高（或者损失最小）的路径作为最终的识别结果。

由上所述，一个基于过切分的中文手写字符串识别系统是由多个复杂的模块组成，涉及到许多相关的工作，比如文本行切分、字符分类器、语言模型、几何模型以及搜索算法等。文献 [45]首次将该框架成功应用于中文手写字符串识别，重点阐述了路径评价方式、路径搜索以及参数估计这三个方面。对于路径评价函数，从贝叶斯决策的角度融入了单字分类器、几何模型和语言模型；对于路径搜索，作者提出了一种精简的集束搜索方式，可以更高效地得到识别结果；对于参数学习，作者提出一种称为最大字符准确率的准则优化各个上下文信息的融合参数。最终，整个系统在公开数据集上可以获得最高的90.75%的准确率。关于路径搜索和参数学习，已经有不少的工作在这两个方面做出了贡献 [45, 56]，由于这两个方面不是本文的重点，所以不做介绍。

本文主要讨论语言上下文模型、过切分算法、单字分类器以及几何模型这几个方面。进一步，可以将它们分为语言模型和形状模型。形状模型是对文本行图像进行建模，包含过切分、单字分类器和几何上下文模型；语言模型刻画的是识别结果语言信息方面的上下文依赖性。下面本文将分别对它们的研究现状进行介绍，主要侧重于近几年使用深度学习方法获得成功的工作。

2.1.2　语言上下文模型

语言上下文的概念来源于自然语言处理，包括句法、语义、语用等内容 [57]。统计语言模型的作用是给出一个字符（或者词）序列在语言学范畴内合理程度的先验概率，已经在许多领域诸如语言识别、机器翻译以及信息检索都有了广泛的应用 [58]。目前广泛使用的N-gram模型最早在20世纪80年代提出，并成功应用在语音识别系统中 [59]。

一般来说，高阶语言模型可以捕获更多的上下文信息从而能够更准确地估计序列概率。文献 [60]发现如果给定充足的训练样本，N-gram语言模型的性能可以随着文法数的增大而持续提升，直至八元文法。然而，在实际运用过程

中，N-gram语言模型本身存在的缺陷导致其很难扩展到高阶模型。首先，因为语料的稀疏性，往往有较多语言组合在训练文本中没有出现，这就会导致很多N-gram的概率估计为零。为了解决该问题，研究人员提出了各种不同的平滑算法，将非零概率的N-gram平滑到零概率的组合上。常见的平滑算法主要包含Good-Turing、Katz以及Kneser-Ney等，文献 [61]详细介绍了目前常见的几种平滑算法，在文字识别中，各种方法差别不大 [62]。其次，对于传统基于词频的N-gram语言模型，随着阶数的增高，其参数数目会呈指数上升[1]（即维数灾难），这就导致高阶语言模型很难稳定地估计上下文，因而传统语言模型都存在短距离的限制。为此，研究者提出了不少解决方案。文献 [63]提出使用基于缓存的语言模型，其核心思路是认为在同一个文档中，前面出现的词在后面出现的可能性会相应增大。虽然这种基于缓存的方法可以大大降低测试语料库上的困惑度（Perplexity，PPL），但是由于在实际的识别过程中，经常会出现错误的组合存在缓存中，所以该语言模型的表现并没有达到预期。文献 [64]提出一种基于随机森林的语言模型，可以通过查询决策树中某一个长序列和某个特定的词的搭配情况来获得语言信息。Rosenfeld提出最大熵（Maximum Entropy）模型将各种语言上下文知识进行整合统一建模 [65]，有效地克服了N-gram 模型信息单一的缺点，显著降低了困惑度和识别错误率，但是也带来了很高的模型复杂度。另外，利用潜在语义分析（Latent Semantic Analysis，LSA） [66]或者话题模型（Topic Model） [67]等方法都在一定程度上克服了N-gram 模型的短距离限制，加入了全局的语言上下文信息。同时，另外一种用来克服数据稀疏的方式是采用基于词类的语言模型。一般方法是将词划分成为不同的类别，再使用传统的N-gram方法进行训练。这种方式可以使得系统对于没有见过的语言模式有更好的鲁棒性。文献 [58]对几种常见的词类语言模型进行了比较。在基于词类的语言模型中，如何进行词类的划分是一个关键问题同时也是一个理论难点。同时，文献 [58]也表示，随着数据量的增长，这些新技术对于语言模型的提升十分有限。

2003年，Bengio等人提出一种新的语言模型，称为神经网络语言模型 [68]。这种新的语言模型可以将字（或词语）从离散空间变换到连续空间中，从而有效地解决了数据稀疏问题，同时在连续空间中可以进行隐式的平滑。相比传统N-gram语言模型，该语言模型可以有效地降低困惑度。神经网络语言模型已经在语音识别 [69, 70]、机器翻译 [71, 72]以及手写文本识别领域 [29, 73]都

[1]可以认为对每一种语言文法组合都分配一个参数。

获得了成功的应用。研究人员在进一步提高性能 [70, 74–77]和减少时间复杂度 [78–81]方面做了大量的工作。Mnih等人 [74]提出了一种基于连续空间词向量表示的双线性语言模型，使用预测词向量和各个已知词的词向量的相似度估计概率，可以在训练和预测速度有较大提升的前提下进一步提升效果。Mikolov [70]使用递归神经网络构建语言模型，通过隐层的递归连接，在进一步减少参数的同时，理论上还可以摆脱语言模型的阶数限制。神经网络语言模型的复杂度比传统N-gram语言模型要高很多，因此文献 [78]使用输出层分解的方法替代传统的Softmax层，可以进一步加速训练过程。

神经网络语言模型根据网络结构的不同一般可以分为前馈模型 [29, 68, 69, 71, 72]和递归模型 [70, 80, 82, 83]。网络结构对于神经网络的性能影响很大，许多研究人员对此进行了综合深入的研究 [80, 84–88]。Mikolov等人 [80]在两个语料库上给出了前馈网络和递归网络语言模型的初步比较结果，最终发现递归网络的表现均要优于标准前馈神经网络。此外，文献 [84]使用多种语言模型技术在公开数据集上对困惑度进行了评测，进一步证明了递归神经网络语言模型的优越性。然而，上述两个工作仅仅是在语料库上进行了语言模型的评测，并没有在实际系统中比较不同的语言模型结构（在实际系统中可能存在带有噪声的输入）。Sundermeyer等人 [85]在法语语音识别的任务中比较了长短时记忆神经网络语言模型（递归神经网络语言模型的一种变形）和前馈网络。他们的实验结果表明基于长短时记忆结构的网络不论在困惑度和词错误率上都要表现得更好。这个工作随后得到了扩展 [86]，他们在两个连续语音识别大数据上进一步比较了传统N-gram语言模型、递归网络、长短时记忆网络以及前馈网络语言模型，结果再一次证明了递归神经网络语言模型的优越性。文献 [87]比较了深层前馈神经网络和递归神经网络语言模型，结果表明深层前馈网络对性能并不会有显著的提升。而文献 [88]则发现，十元文法前馈网络语言模型要优于递归网络语言模型，但是在大规模英语法语翻译任务上两者的表现基本一致，除此之外，几乎所有的文献都表明递归网络的性能要好于前馈神经网络语言模型。

另一方面，虽然统计语言模型在实际的应用中取得了巨大的成功，但是它一直受到以乔姆斯基（Chomsky）为首的语言学家诟病。乔姆斯基认为一个句子的概率是一个完全无用的概念，并不能对一个语句的意义进行理解和解释；后来，随着N-gram 模型的广泛应用，他们认为N-gram 模型没有能力进行长距离建模，而语言学中的句法分析恰恰就能对一个句子进行完整的剖析，得到全局的结构信息。为此，研究者通过句法分析得到结构语言模型（Structured

Language Model），并且融合到N-gram模型中。近年来这种结构语言模型也陆续被应用到语音识别中 [89, 90]，并且性能与存储空间也有了逐步的改进 [91]。

尽管语言模型的研究发展迅猛，但是目前大量的研究基本都在自然语言处理和语音识别中进行。而在文字识别中，尽管很多系统都使用传统语言模型提升了系统识别性能 [26, 92–94]，可是相关的应用研究还相对较少 [95]。而对于最新的神经网络语言模型，更是仅有少量的工作验证其效果 [29, 73]。文献 [29] 将前馈神经网络语言模型与三个当前最好的英文手写识别系统的解码模块结合，实验证明与传统N-gram语言模型相比，新的语言模型可以显著的降低其错误率。文献 [73] 将递归神经网络语言模型融入一个手写文本识别系统的N-best列表重打分步骤，同样在性能上取得了显著的提升。上述两个工作缺乏对不同结构的神经网络语言模型深入系统的比较。在中文识别中，常规的方法都只是用到了简单的传统N-gram语言模型 [45, 56, 96–98]，对于长距离上下文的建模能力还十分有限，有待进一步的扩展。

2.1.3 形状模型

除了语言上下文，字符分类器、过切分算法以及几何上下文模型在字符串识别系统中同样发挥着重要的作用。由于这三者都是从图像层面对文本行进行建模，所以本文统称它们为形状模型。下面分别对它们的基本方法进行介绍。

2.1.3.1 过切分算法

对于连续书写的文档图片，由于书写人的书写随意性，字符之间往往存在粘连。过切分算法的目标就是在切分开粘连字符的前提下，使产生的冗余切分点数量越少越好。早期的过切分算法主要依据竖直投影 [99]、轮廓分析 [1]和骨架分析 [100]等策略进行字符切分，也取得了一定的成功。Liu等人 [1]提出了一种基于局部轮廓分析的过切分算法。但笔划区域表示文字图像中那些具有竖直笔划穿越数为一，并且游程长度有限的水平位置。为了检测粘连点，他们对局部轮廓进行了形状分析，并与其中经验性规定的粘连结构类型进行匹配。在基于粘连类型匹配的切分之后，再执行基于图像投影轮廓分析的后处理步骤，从而在较宽的连通区域上产生额外的竖直切分线段。该切分方法分别在日文邮政地址 [1]和手写中文文本识别 [45]中得到了成功应用。欠切分错误的产生主要是由于基于单笔划区域检测粘连点的不足以及经验性定义的粘连类型的局限性。Xu等人 [2]提出一种基于前景点可见性分析的粘连字符切分方法，该方法

结合了前景骨架分析和字符轮廓分析。相比较轮廓分析，前景骨架分析有利于更准确地找到正确切分点，同时基于轮廓分析的切分点可见性度量能有效地过滤冗余切分点。相比 [1]，该方法能进一步提升切分点召回率，而且保持冗余切分点比例适中。

大多数中文手写字符串切分算法都是基于启发式规则，手段比较单一，使得它们在处理复杂粘连或者复杂任务的时候泛化性不是很好。目前仅有少量[49, 94, 101, 102]的方法基于统计学习，也在切分字符串上获得了很高的召回率。[94]提出一种在场景文字识别中使用的切分方法，通过滑动窗方法使得在公开数据集上的识别性能有了很大的提升。对于手写中文文本，文献 [102]利用训练好的模糊决策树来过滤冗余切分点，但是其提取的部分几何特征都假设了字符两两粘连，不适用于多字符粘连的情形。文献 [49]提出一种基于学习的切分线段过滤算法，通过训练一个线性分类器判定切分线段是否冗余。但是，这种方法同样也只能处理单粘连情况。

2.1.3.2　字符分类器

经过文本行切分以后，相邻的连续几个部件组合而成一个候选字符模式。字符分类器的作用就是对每一个候选字符模式进行分类，得到一些候选字符类别及其对应的相似度得分。

传统的字符分类器涉及三个方面：字符形状归一化、特征提取以及分类器设计。文献 [103, 104]对这三个方面进行了详细的介绍。一般在手写字符识别中，目前效果较好的是线密度插值归一化（Line Density Interpolation，LDI） [7]和归一化自合梯度特征提取（Normalization-Cooperated Gradient Feature，NCGF） [8]的方式。在汉字识别中，流行的传统分类器有两种，分别为修正的二次判别函数（Modified Quadratic Discriminant Function，MQDF） [10]和最近邻原型分类器（Nearest Prototype Classifier，NPC） [12]。相比较而言，MQDF分类器能够比NPC分类器获得更高的分类精度，但是也要付出更多的存储和计算代价。文献 [13]提出利用特征间的相关信息进行二次升维的方法可以在公开数据集上获得目前传统方法最高能达到的94.92%的识别率。

神经网络方法使用判别学习训练网络参数，训练复杂度很高，长久以来很少在汉字识别这样的大类别集分类任务中直接使用。随着图形处理器被广泛应用于科学计算，基于神经网络方法的训练变为可能。其中，基于卷积神经网络的方法在很多视觉任务中都取得了突破性进展 [105]。该网络由多个卷积层、

池化层和全连接层构成。卷积层使用权重共享和局部连接的网络连接方式，一方面大大减少了权重参数，另一方面利用了图像的局部特征。池化层对卷积层提取的特征在各区域内进行降采样，使得提取的特征对字符形变具有一定的不变性，而全连接层相当于分类器。深层卷积神经网络拥有多个卷积层和池化层，因而能够提取更具有表示能力的高层图像特征。相比传统方法，基于卷积神经网络的字符分类器在分类精度上已经有了压倒性的优势，占据了统治地位 [4]，近几年涌现了许多有意义的工作 [15–18]。文献 [15]集成多个10层的深度卷积神经网络，在手写汉字脱机和联机数据集上均取得了超过以往方法的效果。该方法在每轮训练中使用仿射变换和弹性畸变合成训练样本，增强网络的抗形变能力。文献 [16] 对CNN的部分卷积层去掉参数共享的限制，从而增强了网络的拟合能力，同时在训练中对各网络层使用轮替训练避免过拟合。实验中，使用四个网络集成的系统获得了接近于人类识别结果的性能。Zhang等人[18]将传统的NCGF特征图与卷积神经网络进行了结合，在没有扰动数据的帮助下，使用较为简单的网络在联机和脱机手写数据库上再次刷新了性能，超过了人类的识别精度。文献 [17]则是将Gabor特征图作为网络的额外输入，同样取得了不错的结果。尽管单字分类精度已经很高，但是除了文献 [50] 之外，还没有工作表明其能对字符串识别表现有多大的提升程度，这一工作有待进一步深入。

2.1.3.3 几何上下文模型

几何上下文指的是候选字符模式相对整个字符串的高度、宽度、位置和相邻候选模式之间的距离、相对位置等信息。由于在单字分类时，经过归一化等操作，字符的几何上下文信息会全部丢失。在字符串识别中，还要考虑众多的上下文依赖关系，因此如果能将这类信息补充进去，可以进一步增强识别性能。在中文手写字符串中，几何模型的融入对于提升英文字母、标点符号等字符的分类精度尤为明显 [45]。

传统的几何上下文模型一般要先提取几何特征，然后采用一定的分类器模型进行建模。几何模型根据表示对象数量的不同可以分为一元模型和二元模型。一元模型根据单个字符的几何特征进行建模，二元模型提取的是相邻两个字符之间的几何特征。文献 [106]将字符在单词中的相对位置和相对大小等一元特征进行建模，文献 [107]将26 个英文字母按照外形聚类为三个超类，然后在识别过程中计算基于超类的相邻候选之间的相容程度。另一方面，根据

表示对象类别的不同，几何模型又可以分为类别相关的模型和类别无关的模型。类别相关模型可以被视作字符分类器的一种补充，通过几何特征将字符划分到某个特定的超类中；类别无关模型用来表示一个对象是否属于一个字符模式或者一个字符间的切分点。将上述模型组合起来，实际上可以组成一元类别相关、二元类别相关、一元类别无关以及二元类别无关几何模型 [51]。文献 [51, 52]对类别有关的一元和二元几何特征分别用两个二次判别函数（Quadratic Discriminant Function，QDF）进行建模，而对与类别无关的一元和二元几何特征则采用两个支持向量机（Support Vector Machine，SVM）进行建模。

对于中文字符串的识别，早期有研究者将几何上下文特征应用于中文邮政地址识别中，文献 [108, 109]使用了字符的几何位置以及长宽来提高识别精度，这些工作采用的几何特征相对比较简单。文献 [45]首次将几何上下文模型和字符识别模型有效地集成到统一的路径评价准则中，并将其用自由书写的中文字符串识别中，显著提升了系统识别精度。此外，文献 [52]成功将上述四种几何模型应用到文本行对齐中，获得了不错的效果。对几何模型来说，如何选取特征以便更好地建模是其核心问题。因而，传统的基于特征提取结合分类器的方法需要仔细设计特征才能使几何模型发挥重要的作用。文献 [52]为一元类别相关模型设计了多达42种特征。尽管几何模型对字符串识别系统性能的提升有重要的作用，但是目前还没有工作进行基于深度学习的几何上下文模型的研究。此类研究的好处在于可以自动学习特征，在告别繁杂的特征设计环节的同时能够进一步提升字符串识别的性能。

2.2　基于隐式切分的方法

尽管基于显式切分的方法在中文手写字符串识别中表现优异，但是它存在两个不足的方面：首先，在复杂粘连的情况下，过切分算法的表现并不稳定，影响了系统识别性能的上限；其次，识别框架由多个模块组成，各个模块的优化相对独立，很难达到全局最优。基于隐式切分的方法可以一定程度上避免上述两个不足，而且一般都能进行弱标记训练（训练样本不必标注到字符级别，只需要标记到行级别），其方法主要分为两类：基于隐马尔科夫模型的方法以及基于递归神经网络的方法。下面分别对这两种方法进行介绍。

2.2.1　基于隐马尔科夫模型的方法

基于隐马尔科夫模型（Hidden Markov Model，HMM）识别框架的一般流

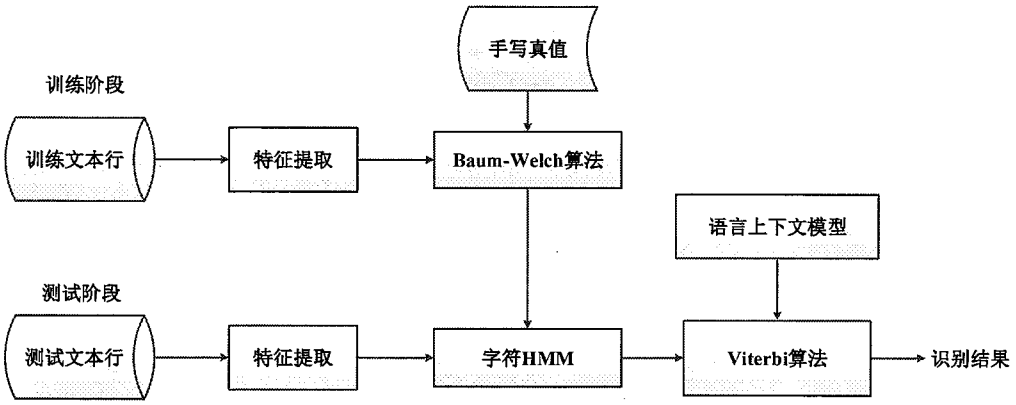


图 2.3: 基于隐马尔科夫模型的识别基本框架。

程为，首先将手写文本行几乎等间隔地划分为一系列的小窗口进行特征提取，然后在训练阶段使用Baum-Welch算法实现对初始字符模型的循环重估，在测试阶段，通过Viterbi算法在语言模型的约束下得到输出字符串序列。其系统框图如图 2.3所示。HMM在拉丁语系字符串识别中研究的较为充分 [26–29]。借用语音识别中成功使用HMM的经验，文献 [26]首次将隐马模型应用于大规模无约束手写英文字符串识别中，并且详细研究了不同规模的语言信息对于识别率的影响。文献 [27]提出一种混合HMM和神经网络的识别框架。传统的HMM主要使用混合高斯模型（Gaussian Mixture Model，GMM）进行字符状态建模，作为一种生成模型其判别能力不强，而替换成为神经网络模型之后可以使得系统的识别性能有较大幅度的提升。

在中文文档识别中，也有不少研究者提出了一些基于隐马尔科夫模型的识别框架 [44, 46–48]。文献 [46]提出使用HMM进行孤立手写中文字符的识别，字符图像首先被切分成一系列的局部区域并提取特征，特征向量可以作为HMM的观测值，而HMM状态的变化则反映了字符特征空间的结构，因而可以用来进行单字识别，实验结果表明这种方法有不错的效果。文献 [44]提出了首个基于HMM的无约束中文手写字符串识别系统。该系统使用嵌入Baum-Welch算法（Embedded Baum - Welch）训练得到字符隐马模型，然后使用Viterbi算法将后验概率最大的路径作为识别结果，同时也揭示了HMM和基于切分的系统之间的互补性。文献 [47]提出使用基于深度神经网络的隐马尔科夫模型进行中文字符串识别的框架。该方法将从滑动窗中提取到的梯度特征送入深度神经网络，网络的输出作为HMM状态的后验概率，最后使用贝叶斯决

策理论融入N-gram统计语言模型。实验结果表明，使用上述策略可以明显提升识别精度。文献 [48]研究了基于HMM的书写人自适应问题，在 [47]的基础上，提出一种基于书写人编码的自适应方法，可以进一步提升识别系统对于特定书写人的识别精度。书写人编码可以利用书写人信息将通用的深度网络模型变换到特定的书写人空间。在训练阶段，使用轮替优化的方式，分别优化书写人信息层与卷积深度网络；而在识别阶段，使用两遍识别的方式来产生无监督的书写人编码，进而指导识别器更好地识别文本行。总的来说，虽然使用HMM的方法在中文手写字符串领域有了一些成果，但是由于中文字形复杂、类别数较大，使用HMM很难对字符进行很好的建模，也较难利用高阶上下文信息，所以还有巨大的提升空间。

2.2.2　基于递归神经网络的方法

基于端到端递归神经网络（Recurrent Neural Network，RNN）的方法是近几年提出的一种新的识别框架，已经在拉丁语系文本识别问题上得到了广泛的应用 [31–35]，可以取得比基于隐马尔科夫模型的方法更好的效果。相比传统的HMM，基于RNN的框架有两个优势 [31]：首先，状态的建模可以不再只依赖上一个状态，可以更好更灵活地对上下文信息进行建模；其次，作为一种判别模型，一般可以获得比作为生成模型的HMM方法更好的性能。作为一种隐式切分的框架，基于RNN 的框架同样可以进行弱标记学习，即标记真值不需要包含每个字符的位置信息。与前馈神经网络相比，递归神经网络通过隐层中的递归连接，可以捕获更多的上下文信息。一般的，还可以使用双向RNN对序列进行更全面的建模。另一方面，连接时序分类（Connectionist Temporal Classification，CTC）的方法发挥了很大的作用。CTC损失最早发表在 [110]，作为一种基于序列的训练准则，它可以在不对文本行图像进行预切分的情况下，使用前向后向算法计算一个输出序列可能对应的多条路径的概率之和，进而可以最大化真实序列的输出概率。

基于RNN的识别框架一般又可以细分为两种不同的流程。一种是基于滑动窗的方法，核心思路是将文本行通过滑动窗的方式转变为一维序列。如图2.4(a)所示，在手写文本行图像中，使用滑动窗对文本行进行切分，然后分别在窗口内提取特征，接着使用递归神经网络得到一组输出序列，最后使用CTC准则对网络进行训练。这种方式和HMM框架有些类似，但是使用RNN进行状态建模可以获得更好的性能。文献 [31]首次提出这种框架，最终，在公开IAM数

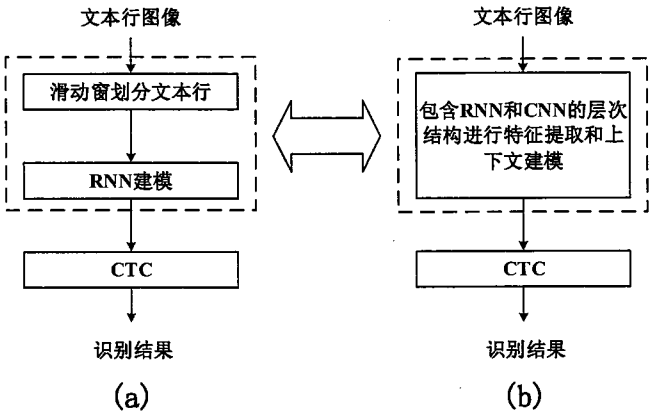


图 2.4: 基于端到端递归神经网络的基本框架，虚线标出的是两个方法之间不同的部分。(a) 基于滑动窗的方法；(b) 基于全局图像特征提取建模的方法。

据集上，提取与HMM框架相同的图像特征，在IAM数据集上的识别正确率提高近10%。另一种方法是基于全局图像特征提取建模的方法。如图 2.4(b)所示，文本行图像经过层次化的深度神经网络进行特征提取，深度神经网络包含卷积、二维RNN等模块，最后通过池化（Pooling）操作自然地转化为高度为一的一组序列，该框架也使用CTC准则进行网络参数的更新。鉴于手写字符串是一个二维序列，一般形变较大，如果使用第一种框架，只能对水平方向上的变形进行建模，也不能自动学习特征，阻碍精度的进一步提升；而使用第二种方法可以真正对文本图像数据进行完整的建模，在实际使用中有更好的表现。这种二维结构由文献 [34]首次提出，使用单纯的图像作为网络输入，在几乎没有预处理和后处理的帮助下，在阿拉伯文识别中超过了所有之前发表的结果。文献 [35]进一步改良了二维递归神经网络的扫描方式，并使用图形处理器上进行了重新实现，最终在IAM数据集上获得了最高性能。实际上，这种基于递归神经网络的方法也已经成功应用于场景文字识别中。该方法可以在没有图像二值化、字符前景提取模块的情况下，直接使用文本行原图作为输入，已经在公开数据集获得了最好结果 [111, 112]。

在手写中文字符串识别中，使用基于RNN框架的工作并不多。文献 [54]提出使用混合的双向RNN和前馈下采样层次结构进行联机中文手写字符串的识别，最终性能超越了传统方法。文献 [53]提出首先使用path signature特征图将联机文本转换为脱机文本，然后利用全卷机网络提取图像特征，再结合多层双向RNN结构以及隐式语言模型的新框架，最终识别表现超越了之前的所有方

法。以上两个都是基于联机手写字符串的工作。文献 [55]是唯一一个在脱机中文手写字符串识别上获得成功的工作。该工作使用层次结构，结合二维RNN以及卷积网络，最终能达到和现有最好系统可以比较的精度。该系统的缺陷是需要在其他语言样本上进行预训练，否则网络很难收敛到满意的结果。此外，基于注意力机制的框架已经在语音识别和机器翻译中得到成功的应用，近来有学者将该机制用于文本识别 [36]，也能获得不错的识别结果，A2iA公司甚至使用该框架进行篇章级的手写文本识别任务 [37, 38]。但是在中文手写文本识别领域，这方面的研究工作还没有得到开展。

2.3　本章小结

本章主要介绍了中文手写字符串中的两种常用的框架：基于显式切分和基于隐式切分的框架。尽管目前基于显式切分的框架可以取得最高的识别性能，但是也存在依赖过切分以及很难得到全局最优模型的缺陷，而基于隐式切分的方法一定程度上可以克服上述缺陷，也能进行弱监督学习，但是目前的研究还不够充分，在识别精度上还有一定的提升空间。

在基于显式切分的框架中，语言上下文模型以及形状模型有着十分重要的作用。对于语言模型，尽管基于词频统计的传统N-gram模型在字符串识别系统中已经有了广泛的应用，但是模型本身很难扩展到高阶以便获得更好的性能。有很多工作尝试从解决数据稀疏和维度灾难这两方面改善N-gram语言模型，但是随着数据量的增长改善效果十分有限。而近期提出的神经网络语言模型可以将字（或词）从离散空间变换到连续空间中，可以在空间中进行隐式的平滑，一般根据网络结构的不同一般分为前馈神经网络语言模型和递归神经网络语言模型。神经网络语言模型可以取得比传统方法都要好的性能，也有许多新的改良算法出现，在语音识别和机器翻译中已经得到了成功的应用。但是在中文手写字符串识别领域中，尚且缺乏深入系统的研究。

形状模型包含过切分算法、单字分类器和几何上下文模型。其中，过切分算法的好坏将直接影响切分识别系统识别性能的上限。虽然基于竖直投影分析、轮廓分析、骨架分析等启发式规则的方法可以获得不错的切分效果，但是在处理复杂粘连的情况下并不鲁棒。而另一方面，基于统计学习的方法的研究还并不深入，目前还只能处理简单的单粘连情况。对于单字分类器，有较多的工作对传统方法进行了研究，包含字符归一化、特征提取以及分类器设计等方面，都获得了较大的成功。随着基于深度卷积神经网络的模型的出现，深度学

习方法已经在单字识别方面占据了主导地位，甚至在识别精度上已经超过了人类。虽然孤立字符识别器的性能已经有了明显的提升，但是目前只有极少量的工作对于其在串识别系统中的表现有相关的研究。对于几何模型，尽管大量文献表明其对于提高系统识别的性能有很大的帮助，但是其特征设计较为复杂，目前还缺乏基于深度学习的方法，这一点也有待进一步深入研究。

基于隐式切分的方法包含基于隐马尔科夫模型和基于端到端递归神经网络的方法。基于HMM的方法在拉丁语系文本识别领域获得了不错的应用效果。在中文识别领域，尽管HMM也能获得不错的精度，特别是与深度网络结合之后在精度上获得了大幅提升，但是由于汉字字形复杂且类别很大，所以这些方法一般都较难达到当前最好性能。

对于基于递归神经网络的方法，可以主要分为基于滑动窗和基于全图两种。基于滑动窗的方法一般只能捕获图像水平方向上的信息，所以在手写字符串识别中的应用受到较大限制。而基于全图的方法在拉丁语系手写字符串识别中能得到最高的识别精度。但是，在中文手写字符串识别领域，基于RNN的研究相对较少，且主要集中在联机领域。在脱机领域，仅有一个工作能得到可以和传统方法相比较的精度，但是需要在其他语言样本上进行预训练，否则网络很难收敛到满意的结果，因此该方向有待进一步深入研究。

第三章　基于神经网络语言模型的显式切分识别方法

3.1　引言

如前所述，基于显式切分的系统融合了语言上下文模型和形状模型，已经在中文手写字符串识别上获得了成功 [4, 45]。本章主要介绍使用神经网络语言模型进一步提升系统的识别率，基于形状模型的改进将在下一章进行详细介绍。

统计语言模型，主要用于给出一个字（或词）[1]合理程度的先验概率，已经在语音识别、文本识别和机器翻译等领域有了广泛的应用。它可以让最终的输出序列更符合一般语法规律。尽管传统的基于词频的N-gram语言模型已经在字符串识别领域应用了十余年，但是其仍旧是众多识别系统的第一选择。一般高阶语言模型可以对更大范围内的上下文进行建模，所以会取得更好的效果，但传统语言模型受制于数据稀疏和维度灾难等问题，很难扩展到高阶模型。后续出现的诸如缓存语言模型等方法虽然有一定效果，但是在实际系统中对于识别精度的帮助并不明显。而近年提出的一种神经网络语言模型可以将字从离散空间映射到连续空间，进而在连续空间中进行隐式的平滑，可以极大地提升语言模型的性能。

神经网络语言模型根据结构可以分为前馈神经网络语言模型和递归神经网络语言模型，不同的网络结构对于语言模型的性能有较大的影响。此外，神经网络语言模型虽然运算量较大，但存在大量的策略在不降低性能的前提下减少其计算复杂度。目前在过切分识别系统中，对于神经网络语言模型还未进行过深入综合的研究。针对以上问题，本文提出将神经网络语言模型融入过切分识别系统。具体贡献包括以下几点：

1. 首次提出将神经网络语言模型融入中文手写字符串识别系统，综合比较了不同结构的神经网络语言模型以及混合语言模型在识别系统中的作用。
2. 比较不同的神经网络语言模型加速策略对于减小计算复杂度和精度的影响。

[1]本文不严格区分字序列和词序列。

3. 最终通过融入混合递归神经网络语言模型，识别精度超越当前最好结果。

本章余下的内容安排如下：在3.2节中介绍了基于贝叶斯决策的路径评价准则；3.3节详细地介绍了神经网络语言模型的原理和结构；3.3节阐述了两种常见的神经网络语言模型加速方法；最后给出实验结果和本章小结。

3.2 系统建模

基于显式切分的识别系统的基本流程可以参见章节 2.1.1。这里假设已经得到输入文本行完整的切分识别网格，那么接下来的工作就是设计融合多种信息的路径评价函数，然后通过高效的搜索算法得到得分最高或者损失最小的切分识别路径。

本文从贝叶斯决策（Bayesian Decision）的观点出发来对路径评价函数进行建模。令$S = s_1...s_m$表示字符串的一种切分方式，$X = x_1...x_m$是该切分方式下对应的候选字符模式序列，每个候选字符模式x_i用分类器识别后得到候选类别c_i，那么识别结果可以表示为字符序列$C = c_1...c_m$，(X, C)表示一个候选切分-识别结果（对应一条候选切分-识别路径）。这样，字符串类别的后验概率可以定义为：

$$P(C|X) = \sum_{S} P(C, S|X). \tag{3.1}$$

理论上应该选取最大$P(C|X)$对应的字符串作为识别结果，但是由于候选切分路径数量庞大，为了避免对多条候选切分路径进行求和的巨大运算代价，最优字符串识别结果常用下式近似求解：

$$C^* = \arg\max_{C,S} P(C, S|X). \tag{3.2}$$

其中，$P(C, S|X)$可以理解为对一条切分-识别路径的评分，可以分解为：

$$\begin{aligned} P(C, S|X) &= \frac{P(X|C, S)P(C, S)}{P(X)} \\ &= \frac{P(X|C, S)P(S|C)P(C)}{P(X)}. \end{aligned} \tag{3.3}$$

假设字符形状互相独立，上式可以进一步近似为：

$$\begin{aligned} P(C,S|X) &\approx P(C)\prod_{i=1}^{n}\frac{P(x_i|c_i,s_i)P(s_i|c_i)}{P(x_i)} \\ &= P(C)\prod_{i=1}^{n}\frac{P(c_i,s_i|x_i)}{P(c_i)} \\ &= P(C)\prod_{i=1}^{n}\frac{P(c_i|s_i,x_i)P(s_i|x_i)}{P(c_i)}, \end{aligned} \tag{3.4}$$

其中，$P(s_i|x_i)$表示几何上下文信息，先验概率$P(C)$表示语言上下文的概率，$P(c_i|s_i,x_i)$表示字符识别模型，因为一般与几何上下文无关，可以用$P(c_i|x_i)$近似。

这样，通过式 3.4进一步化简便可以得到路径评价函数。为了方便介绍，本文约定：第i个字符模式被字符分类器分为候选概率c_i的概率表示为$P(c_i|x_i)$；$P(c_i|h_i)$表示为语言上下文信息，其中h_i表示c_i的历史字符；对于几何模型，一元类别相关（unary class-dependent geometric，**ucg**）、一元类别无关（unary class-independent geometric，**uig**）、二元类别相关（binary class-dependent geometric，**bcg**）以及二元类别无关（binary class-independent geometric，**big**）模型分别表示为$P(c_i|g_i^{ucg})$，$P(z_i^p=1|g_i^{uig})$，$P(c_{i-1},c_i|g_i^{bcg})$和$P(z_i^g=1|g_i^{big})$，其中g_i表示相应提取到的几何特征。进一步的，本文做如下标记：

$$\begin{aligned} lp_0 &= \log P(c_i|x_i), \\ lp_1 &= \log P(c_i|g_i^{ucg}), \\ lp_2 &= \log P(z_i^p=1|g_i^{uig}), \\ lp_3 &= \log P(c_{i-1},c_i|g_i^{bcg}), \\ lp_4 &= \log P(z_i^g=1|g_i^{big}), \\ lp_5 &= \log P(c_i|h_i), \\ lp_6 &= \log\frac{1}{P}. \end{aligned}$$

其中lp_6表示一个常数。

通过上述模型，可以得到对数形式的路径评价准则$f(X,C)$。根据路径惩

罚机制的不同，路径评价准则一般来说有以下四种形式（本文不详细介绍文献[45]的推导过程），它们在一定程度上都可以避免路径长度对识别结果的影响：

1. 归一化形式（NPL）：

$$f(X,C)=\frac{1}{m}\sum_{i=1}^{m}(lp_0+\lambda_1 lp_1+\lambda_2 lp_2+\lambda_3 lp_3+\lambda_4 lp_4+\lambda_5 lp_5). \quad (3.5)$$

2. 路径惩罚项形式（WIP）：

$$f(X,C)=\sum_{i=1}^{m}(lp_0+\lambda_1 lp_1+\lambda_2 lp_2+\lambda_3 lp_3+ \lambda_4 lp_4+\lambda_5 lp_5)+\lambda_6[m\cdot lp_6]. \quad (3.6)$$

3. 片段数加权形式（WSN）：

$$f(X,C)=\sum_{i=1}^{m}(k_i lp_0+\lambda_1 lp_1+\lambda_2 lp_2+\lambda_3 lp_3+\lambda_4 lp_4+\lambda_5 lp_5). \quad (3.7)$$

4. 候选字符宽度加权形式（WCW）：

$$f(X,C)=\sum_{i=1}^{m}(w_i lp_0+\lambda_1 lp_1+\lambda_2 lp_2+\lambda_3 lp_3+\lambda_4 lp_4+\lambda_5 lp_5). \quad (3.8)$$

其中，m表示路径中候选模式个数，k_i表示候选字符模式所包含的片段数，w_i为相应候选字符的宽度。文献 [45]测试了四种路径评价准则函数的性能，实验表明四种路径评价准则均能克服路径长度影响，大幅度提高识别率。四种方法里，WSN和WCW性能相当，比WIP和NPL稍好，在这里我们选取WCW，即用候选字符的宽度对字符分类器的输出进行加权，字符越宽，候选字符加权值越大，但对固定字符串，整条路径上的片段的宽度和是相等的。λ_1至λ_5 表示的是各个上下文的线性组合系数，用来平衡各个模型的作用。在这里，本文采用的是文献 [45]提出的基于最大化字符准确率（Maximum Character Accuracy，MCA）的训练方法获得各个融合参数的大小。按照上述的方法，首先每个模型都经过置信度转换将其映射成为0到1之间的一个概率值，然后通过包含字符

分类器、语言模型、几何模型等模型在内的加权和获得整个切分识别路径的评分，最后通过集束搜索算法（Beam Search）得到最终识别结果。本文采用[45]提出的一种动态规划算法和集束搜索算法结合的两级路径消减算法进行路径寻优，它可以在保证字符串识别率的前提下，大幅提高搜索效率。第一级消减是在每个候选字符的候选类别中，用动态规划算法进行消减，即终止于某一候选字符的候选类别的路径中，保留一条最优路径；第二级消减，是在每一切分点进行消减，即在终止于某一切分点的多个最优候选路径中（前面的动态规划步骤里保存的最优路径），最多保留BW条（一般取10）最优路径，而将其他路径消减，即集束搜索过程。

3.3　神经网络语言模型

如章节 2.1.2所述，传统语言模型推广到高阶时会遇到数据稀疏和维度灾难的问题。为此，本文引入神经网络语言模型以便在克服语言模型平滑问题的基础上，融入高阶语言模型使得中文手写字符串识别系统的精度有进一步的提升。本文主要介绍两种神经网络语言模型，分别是前馈神经网络语言模型和递归神经网络语言模型。为了叙述简洁，两种神经网络语言模型分别简写为FNNLM和RNNLM。此外，由于传统N-gram语言模型一般使用回退方式（Back-off）进行平滑，所以本文简写为BLM。

这里首先给出语言模型的概率表示。假设一个序列C包含m个字符，那么序列概率$P(C)$可以分解为：

$$p(C)=\prod_{i=1}^{m}p(c_i|c_1^{i-1}), \tag{3.9}$$

其中$c_1^{i-1}=<c_1,...,c_{i-1}>$，表示字符$c_i$的历史字符序列。对于一个N-gram语言模型，只考虑前$N-1$的历史词，所以$P(C)$可以进一步表示为：

$$p(C)=\prod_{i=1}^{m}p(c_i|c_{i-N+1}^{i-1})=\prod_{i=1}^{m}p(c_i|h_i), \tag{3.10}$$

其中，$h_i=c_{i-N+1}^{i-1}=<c_{i-N+1},...,c_{i-1}>$（$h_1$为空）。尽管FNNLM相对BLM可以对更大范围内的上下文信息进行建模，但是它本质上仍然是一种N-gram语言模型。另一方面，理论上RNNLM通过递归连接可以对任意范围的上下文

进行建模，从而摆脱N-gram模型对于文法数的限制。在这种情况下，可以有$h_i = c_1^{i-1}$。

3.3.1 前馈神经网络语言模型

前馈神经网络语言模型的核心思想就是将历史词从离散空间投影到连续空间中，从而完成对语言模型的隐式平滑以便更好地给出字符串的语言概率。在FNNLM中，投影和概率估计可以使用一个简单的多层神经网络进行联合优化。FNNLM的概念最早由Bengio等人提出 [68]，可以很好地克服维数灾难。如图 3.1所示是单隐层神经网络语言模型的基本结构。

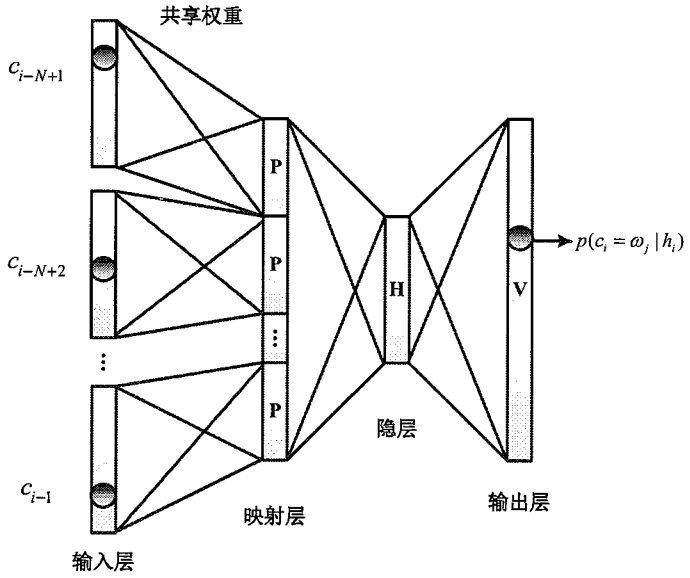


图 3.1: 单隐层前馈神经网络语言模型基本结构。P表示映射层的大小，H和V分别表示隐层和输出层大小。

FNNLM仍然是一个基于统计的N-gram语言模型，它的输入是前面$N-1$个历史词构成的序列h_i，一般将这些历史词串联起来以便保持相对位置关系。网络的输出是词表中所有字符在当前位置的后验概率，如下所示：

$$p(c_i = \omega_j | h_i), \qquad j = 1, \ldots, V. \tag{3.11}$$

上式中，V表示词表的规模，ω_j表示词表中的某个字符类别。一般的，使

用FNNLM估计词语概率可以分为以下几个步骤[2]：

1. 将$N-1$个历史词中的每一个词通过独热编码（One-Hot Encoding）变换成为长度等于词表规模的V维向量。
2. 将每一个V维字符向量经过映射层变换为连续空间中的一个低维向量$\boldsymbol{r}$。事实上，$P \times V$维投影矩阵中的每一列对应一个词向量，并且映射层的所有参数是被所有历史词所共享的。
3. 如果将映射层和隐层之间的连接权重表示为$\boldsymbol{W}_{P,H}$（维度为$H \times ((N-1) \times P)$），$N-1$历史词向量表示为$\boldsymbol{R} = [\boldsymbol{r}_{i-N+1}^T, ..., \boldsymbol{r}_{i-1}^T]^T$，那么隐层的输出$\boldsymbol{S}$可以用下式进行计算：

$$\boldsymbol{S} = \tanh(\boldsymbol{W}_{P,H} * \boldsymbol{R}), \tag{3.12}$$

   其中$\tanh(\cdot)$表示正切激活函数，按照逐个元素的方式对每一个单元进行激活操作（Element Wise）。如果要将模型推广到多层，只需要将前一隐层的输出作为当前层的输入，然后对当前层重复进行式 3.12的运算即可。

4. 最后，词表中所有词的概率可以使用下式计算：

$$\boldsymbol{M} = \boldsymbol{W}_{H,O} * \boldsymbol{S}, \tag{3.13}$$

$$\boldsymbol{O} = \exp(\boldsymbol{M}) / \sum_{i=1}^{V} \exp(m_i), \tag{3.14}$$

   其中，$\boldsymbol{W}_{H,O}$是维度大小为$V \times H$的输出层权值矩阵，$\boldsymbol{M}$表示Softmax归一化前的激活向量，m_i是$\boldsymbol{M}$中的第i个元素。指数函数$\exp(\cdot)$同样也是使用按照逐个元素的方式对每一个单元进行除法操作（Element Wise）。

需要说明的是，为了叙述简洁，上面的公式中的偏置项都归并为权值参数中的一部分 [113]，可以视作一种增广形式。在完成以上所有步骤以后，$\boldsymbol{O}$中的第j个元素（表示为o_j）便对应语言模型的估计概率$p(c_i = \omega_j | h_i)$。在前馈神经网络语言模型中，一般都使用标准的BP算法（Back-Propagation）进行网络参

[2]本文采用列优先的方式表示矩阵。

数的优化，使用的损失函数为带正则的交叉熵准则：

$$E = -\sum_{j=1}^{V} t_j \log o_j + \beta(\|\boldsymbol{W}_{P,H}\|_2^2 + \|\boldsymbol{W}_{H,O}\|_2^2), \tag{3.15}$$

其中t_j是期望的网络输出，当它为1时表示对应训练语料库中的下一个词，如果不是，则为0。

3.3.2 递归神经网络语言模型

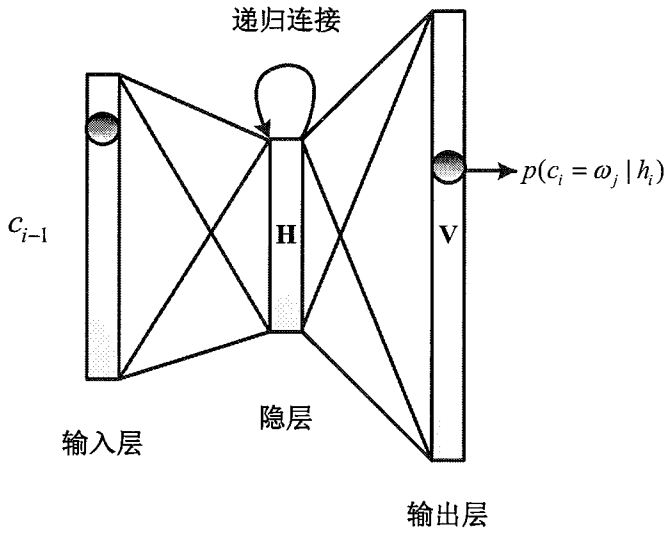


图 3.2: 递归神经网络语言模型基本结构。H和V分别表示隐层和输出层的大小。

递归神经网络语言模型在2010年由Mikolov等人首次提出 [70]，它的结构与前馈神经网络语言模型非常相似，如图 3.2所示，两者最大的不同在于RNNLM通过隐层的递归连接理论上可以对任意范围内的上下文进行建模。RNNLM同样也将词投影到连续空间中，估计词语概率一般要分为以下三个步骤：

1. 首先，组合生成t时刻网络的输入$\boldsymbol{R}(t)$。它由两部分经过连接组成：前一个历史词的独热编码向量$\boldsymbol{x}(t-1)$，以及前一时刻的隐层输出$\boldsymbol{S}(t-1)$。可以将$\boldsymbol{R}(t)$表示为：

$$\boldsymbol{R}(t) = [\boldsymbol{x}(t-1)^T \boldsymbol{S}(t-1)^T]^T. \tag{3.16}$$

2. 然后，将输入向量$\boldsymbol{R}(t)$中的元素做投影，并聚合成为一个连续向量$\boldsymbol{S}(t)$，公式如下所示，同时该向量也将作为送入下一个时刻的隐层输出：

$$\boldsymbol{S}(t) = \text{sigm}(\boldsymbol{W}_{I,H} * \boldsymbol{x}(t-1) + \boldsymbol{W}_{H,H} * \boldsymbol{S}(t-1)), \tag{3.17}$$

其中，sigm(·)表示sigmoid激活函数，同样按照逐个元素的方式对每一个单元进行激活操作，$\boldsymbol{W}_{I,H}$和$\boldsymbol{W}_{H,H}$分别为$H \times V$维投影矩阵以及$H \times H$维递归权值矩阵。

3. 最后，使用与3.3.1节中前馈神经网语言模型步骤4相同的方式对词表中所有的词给出基于历史词序列的概率估计。

与前馈神经网络语言模型的训练准则3.15相似，递归神经网络语言模型的优化同样需要最小化带正则的交叉熵损失。不同的是，RNNLM在优化递归权重的时候需要使用BPTT（Back-Propagation Through Time）算法[80]。BPTT算法的核心思路是将递归连接按照时间顺序展开，然后进行梯度的反向传播，如图3.3所示。由于BPTT不是本文的主要研究内容，所以具体计算公式可以参见[80]。另一方面，为了防止梯度消失或者梯度爆炸，常见的RNNLM都只展开预定的有限步长，即都使用截断的BPTT算法进行优化。

比较FNNLM与RNNLM，两者最大的不同在于对于历史信息的表示方式。对于FNNLM，历史信息被限制在一定数量的前几个词内，本质上还是一种N-gram语言模型；但是对于RNNLM，由于递归连接的存在，理论上隐层可以表示任意范围的上下文。所以，RNNLM相比FNNLM能更有效地利用上下文信息。

3.3.2.1　递归神经网络最大熵语言模型

随着语料库规模的增大，递归神经网络语言模型的隐层节点数也需要增多，否则其性能甚至要比BLM更差[82]。然而，隐层节点的增加也意味着计算复杂度的增加。为了克服这个问题，Mikolov等人将RNNLM与最大熵语言模型进行结合[82]，并称为递归神经网络最大熵模型（RNNME）。该模型同样可以使用BPTT进行训练，而且可以在隐层规模较小的情况下得到不错的性能。

RNNME模型如图3.4所示，递归神经网络语言模型部分保持不变，最大熵语言模型可以视作从网络输入到输出的直连接权值矩阵，权值矩阵中存储的

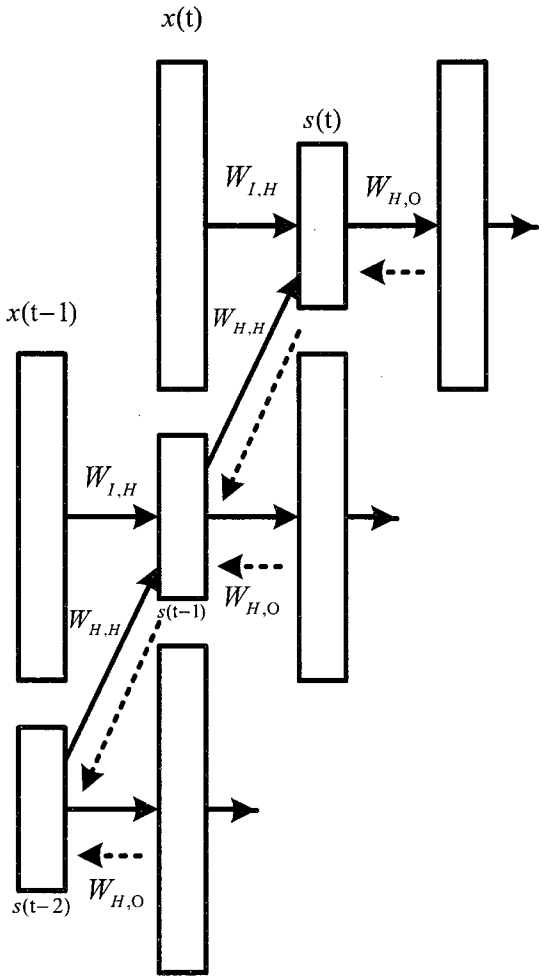


图 3.3: BPTT算法示意图。红色虚线箭头表示将递归神经网络展开后递归连接梯度的传播方向。

是N-gram特征 [114]，N-gram特征表示语言模型中的各种词语组合映射成的连续向量（类似于词向量）。直连接部分可以给递归神经网络语言模型提供很好的互补性，因而可以使得RNNLM用较少的参数获得不错的性能。此外，因为对于最大熵模型部分来说，因为N-gram特征太多，一般采用哈希表的方式进行存取。由于哈希表有冲突的可能，很自然的，对于经常出现的组合就训练的更为充分，所以哈希表的大小可以控制最大熵语言模型的规模。

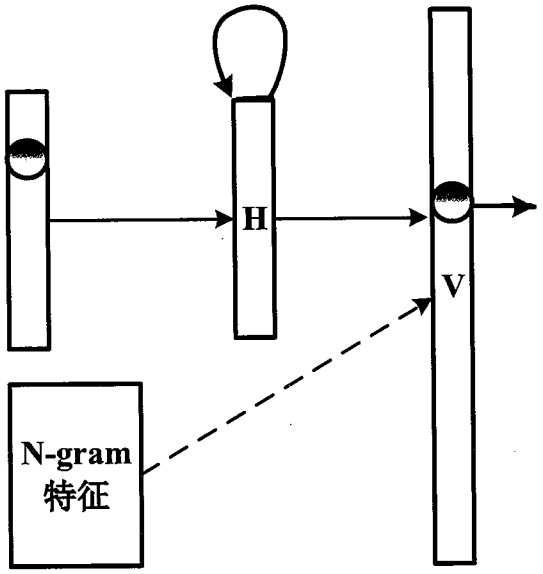


图 3.4: 递归神经网络最大熵语言模型。红色虚线箭头表示最大熵语言模型部分。

3.3.3　混合语言模型

对于识别类别较大的任务，通常的做法是将神经网络语言模型与传统的BLM进行先行插值，以便进一步提升性能 [72]。插值得到的语言模型一般称为混合语言模型（Hybrid Language Model，HLM），如公式 3.18所示。

$$\log P_{hlm} = \lambda \log P_{nnlm} + (1-\lambda) \log P_{blm}. \tag{3.18}$$

其中，P_{hlm}、P_{nnlm}和P_{blm}分别表示混合语言模型、神经网络语言模型以及传统语言模型的概率分布，λ为线性插值系数，一般通过最小化在验证集上的困惑度得到。

为了克服神经网络语言模型较高的计算复杂度这一缺陷，通常会选择一些简单的结构或者一些近似的策略（这在下一节会有介绍），但这会使得神经网络语言模型的性能有一定的降低。有很多文献表明神经网络语言模型与BLM有较强的互补性 [58, 68, 69, 84]，甚至在后面的介绍中可以发现，性能并不强的神经网络语言模型与BLM结合之后也能显著提升HLM的表现。因此，经过简化的神经网络语言模型需要与传统BLM相结合。文献 [81]说明这种现象是由于神经网络语言模型与传统的语言模型分别学到了不同的语言分布。本文将在实

验部分验证这一现象。

3.4 加速策略

传统的BLM可以通过直接查询并结合非常简单的回退插值计算得到相应词语组合的概率值，但是对于神经网络语言模型，由于包含有较多网络层之间的矩阵计算，所以它在训练和测试阶段都有较高的计算复杂度。下面对其计算复杂度做一个详细分析。

以单隐层前馈神经网络语言模型为例。假设h表示隐层节点数，n表示语言模型阶数，m表示词向量，那么一次前馈的时间复杂度可以近似为：

$$|V|(1+h)+h(1+nm)+mn. \tag{3.19}$$

上式中的三项分别表示输出层、隐层以及映射层的计算复杂度。由于在实际应用中，词表的规模一般都很大（中文大概在1万左右，英文更是有数万），所以可以认为复杂度的量级大致为$O(|V|)$ [68]。也就是说，输出层Softmax层占了绝大多数的计算量。因此大多数的加速算法主要着眼于如何减少输出层的运算量。下面主要介绍两种加速算法，分别是短列表（Short-List）和输出层分解（Output Factorization）技术。

3.4.1 短列表

受Bengio等人的工作启发 [68]，Schwenk等人提出使用短列表技术对神经网络语言模型进行加速 [72]，并且成功将其应用到语音识别系统中的网格重打分步骤。该方法的核心思路是选取出现词频概率最高的$s(s \ll V)$个词作为短列表，这些属于短列表的词将使用神经网络语言模型进行概率估计，以便减少输出层的单元数。通过这种方式，词语的概率估计可以用下式进行计算：

$$\hat{P}(c_i|h_i)=\begin{cases}\hat{P}_N(c_i|h_i,L)\cdot P_B(h_i|L), & \text{如果} \quad c_i \in \text{short-list} \\ \hat{P}_B(c_i|h_i), & \text{其他}\end{cases} \tag{3.20}$$

其中，$\hat{P}_N$表示神经网络语言模型估计的词语概率，这些词都存在于短列表之中，$\hat{P}_B$是通过BLM得到的估计概率，随机变量L定义将要估计的词存在于短列

表中，概率$P_B(h_i|L)$通过式 3.21计算得到：

$$P_B(h_i|L) = \sum_{c_i \in \text{short-list}} \hat{P}_B(c_i|h_i). \tag{3.21}$$

式 3.20中，对于$P_B(h_i|L)$的计算仍旧十分耗时，所以可以进一步简化 [72]。具体方法是，在神经网络语言模型的输出部分加入一个额外的类别，表示所有不属于短列表的词集合，这个概率分布可以通过神经网络优化得到。通过这种方式，我们可以认为网络对不属于短列表的词概率给出了足够可信的估计，因而 3.20可以简化为：

$$\hat{P}(c_i|h_i) = \begin{cases} \hat{P}_N(c_i|h_i), & \text{如果} \quad c_i \in \text{short-list} \\ \hat{P}_B(c_i|h_i), & \text{其他} \end{cases} \tag{3.22}$$

这种方式下，语言模型的概率并不严格保证和为1，但是文献 [72]发现使用这种简化形式对于语言模型的性能几乎没有任何影响。通过这种短列表的方式，语言模型的复杂度近似降低为$O(|s|)$。

3.4.2　输出层分解

输出层分解的想法最初源自于文献 [78]（基于最大熵模型的改进形式 [115]），该文献提出在先验知识的约束下，使用二值层次聚类对输出层进行分解，这样计算复杂度可以降到$\log_2|V|$。由于这种层次结构的构建并不是一个简单的问题，在实际使用中通常使用基于类别的模型。在文本中，所有的词被划分成数量较小的词类，这样，输出层的词语概率可以分解为：

$$p(c_i|h_i) = p(\text{class}(c_i)|h_i) * p(c_i|\text{class}(c_i), h_i). \tag{3.23}$$

这种方式下，可以对词类和类内的词分别进行归一化，因而可以降低计算复杂度。有文献表明 [80]，相比使用长度为1万的全词表进行输出层归一化，基于词的分解输出层分解方式可以提升15倍的速度，相比于短词表也更有潜力（短词表方法对于低频词失效）。

在划分词类的过程中，一般的做法是根据某一统计量（比如词频、似然等）得到一个直方图，然后采用类似于图像中“直方图均衡化”的做法，将各

个词划分到相应类别中，使得每一类的统计量之和基本一致。文献 [116]对于统计量的各种选取方法进行了深入比较。尽管基于词频（Frequency-Based）的划分方式比基于概率似然（Likelihood-Based）的方式在困惑度指标上要稍逊色，但是它在分类速度上有明显的优势。因此，为了兼顾精度和速度，本文使用基于词频的划分方式。特别地，本文使用改良的方式进行词类划分，基于词频的平方根而不是词频本身进行词的聚类，理论上可以获得更好的性能[117]。

3.5 实验

为了验证上面介绍的神经网络语言模型的效果，本文将在两个数据集上进行一系列的实验，分别是文献 [42]中公布的一个大规模手写中文数据库，以及文献 [4]公布的一个手写中文竞赛数据集。这两个数据集上都有充足的其他相应的工作可以进行对比。整个中文手写字符串识别系统是在Intel Core i7-4790 3.60 GHz CPU的个人电脑上使用Microsoft Visual Studio 2008 C++实现的。在训练神经网络语言模型的时候，使用了NVIDIA Titan X图形处理器进行加速。下面，本文首先介绍其数据库和实验中设置的相关信息，然后给出所有实验结果。

3.5.1 数据库与实验设置

中文文本识别中最常用的数据集是由中国科学院自动化研究所模式识别国家重点实验室收集的一个大规模自由书写中文数据库，称为CAISA-HWDB [42]。它采集了1020个书写者的样本，包含孤立单字样本和连续自由书写文本，并且按照书写者不同分为训练集和测试集。训练集由816个书写者完成，另外204个书写者的样本归为测试集。训练集中包括3118477个孤立单字样本，类别数为7356类，另外还有4076个手写文本页面（包含1080017个字符）。对于串识别系统，一般是将文本页面中的字符提取出来，与前面提到的孤立字符一起用来训练字符分类器，也就是有4198494个字符训练样本。本文使用的测试集之一就是上述提到的由204个书写者构成的测试集中的文本页面样本，一共有1015个页面，包括了10449个文本行，对应268629个字符。此外，本文还使用2013年ICDAR中文手写识别竞赛中公布的数据集作为测试集（简称为ICDAR-2013） [4]。该数据集规模稍小，包含60个书写人书写的300个页面，并且这些书写人没有参与CASIA-HWDB数据集的书写。

在本章中，主要与相同设置下文献 [45, 97]提出的系统进行对比实验，因而使用与这两个工作完全一致的字符分类器和几何上下文模型，这两类模型同样是在CASIA-HWDB训练集上训练得到的，并且也保证使用相同过切分算法和相同的训练语料库。其中过切分算法使用的是文献 [1]提出的基于轮廓分析的算法，语料库采用的是中文语言资源联盟（Chinese Linguistic Data Consortium，CLDC）提供提供的包含5000多万字符的通用文本库 [45]。

本章使用的单字分类器是在上述提到的孤立字符与文本页面中提取得到的4198494个字符图像上训练得到的，类别数为7356类。对于每一个字符灰度图，首先提取512维归一化组合梯度特征（Normalization-Cooperated Gradient Feature，NCGF） [8]；然后，利用Fisher线性判别分析（Fisher Linear Discriminant Analysis，FLDA）降维到160维；最后对这些向量利用改进的二次判别函数分类器（Modified Quadratic Discriminant Function，MQDF） [10]进行建模。本文将前面提到的4198494个训练字符样本分为两部分，其中五分之四用来训练MQDF分类器的参数，剩下的五分之一用来训练MQDF分类器输出置信度转换的参数（将分类器的输出转化为0到1之间的后验概率）。为了构建几何上下文模型 [52]，本文首先从来自训练集文本页面的41781个文本行中提取几何特征，接着根据CASIA-HWDB的字符边界标记训练相应的几何模型（**ucg**，**uig**，**bcg**，以及**big**）。

前面提到，本文使用与文献 [45]一样的通用语料库。除此之外，本文还从人民日报语料库 [118]和ToRCH2009语料库 [119]收集了总共380万字的语料库作为验证集。验证集有两个作用，首先是用来评测训练得到的语言模型，其次可以用来训练得到混合语言模型的融合参数。文章采用的搜索参数与文献 [45]一致，最大部件连接数、字符分类器候选数以及集束宽度分别设置4、20和10。

在中文字符串识别中，一致沿用两个字符层次的性能指标 [44]，分别是识别正确率（Correct Rate，CR）和识别精确率（Accurate Rate，AR）：

$$\begin{aligned} CR &= (N_t - D_e - S_e)/N_t, \\ AR &= (N_t - D_e - S_e - I_e)/N_t, \end{aligned} \tag{3.24}$$

其中，N_t表示真实文本中的字符数目，（S_e）、（D_e）和（I_e）分别表示替换错误（Substitution Error）、删除错误（Deletion Error）和插入错误（Insertion

Error），这三种错误可以通过将识别结果与真实文本字符串采用动态规划对齐的方式计算得到。除了上述评测识别率的两个指标之外，在评测语言模型好坏的时候还经常使用困惑度这一个指标（Perplexity，PPL），计算方式为:

$$PPL(C) = P(c_1...c_m)^{-\frac{1}{m}} = \sqrt[m]{\frac{1}{P(c_1...c_m)}} = \sqrt[m]{\prod_{i=1}^{m}\frac{1}{P(c_i|c_1...c_{i-1})}} \tag{3.25}$$

由于过切分识别系统中包含很多上下文模型，为了叙述方便，这里给出一个汇总表 3.1:

表 3.1: 中文手写字符串识别系统上下文模型列表。

简称	指代的模型
cls	字符模型
g	几何模型
cbi	字符二元文法
cti	字符三元文法
cfour	字符四元文法
cfive	字符五元文法
rnn	字符递归神经网络语言模型
iwc	词类插值二元文法

3.5.2 实验结果

在本节中，所有语言模型都在相同的通用语料库上训练得到，以便可以公平比较各种模型对于字符串识别系统精度提升的程度大小。为了方便比较，本节的大多数实验都在CASIA-HWDB上进行。首先，本文给出使用不同阶数传统BLM时基准系统的识别表现；然后，本文将比较各种不同结构下FNNLM和RNNLM的表现；最后，本文将给出使用神经网络语言模型的过切分识别系统在ICDAR-2013数据集上的表现。

3.5.2.1 基准系统表现

在基准系统中，使用的语言模型为传统的回退语言模型。为了与前人的工作保持一致 [45, 97]，基准系统的候选部件最大连接数设置为4，字符分类器候选设置为20，集束搜索宽度设置为10。本文使用SRILM（SRI Language Model

toolkit (1.7.1)）工具 [120]训练二元到五元文法，和 [45]一样，采用SRILM默认的Katz平滑算法对于没有见到过的词语组合进行平滑，并且采用基于熵的剪枝（Entropy-Based Pruning）对语言模型进行适当的裁剪。对于二元到五元文法，裁剪阈值分别设置为5×10^{-8}、10^{-7}、10^{-7}以及10^{-7}。其中，对于二元和三元文法的剪枝参数来说，与文献 [45]是一样的.

表 3.2: 使用BLM的基准系统识别结果。时间栏表示识别所有测试页面所花费的小时数。

模型组合	AR（%）	CR（%）	时间（h）	PPL
cls+cbi+g [45]	89.56	90.27	11.33	-
cls+cti+g [45]	90.20	90.80	11.54	-
cls+cbi+g	89.57	90.28	6.60	144.81
cls+cti+g	90.21	90.81	6.68	82.97
cls+cfour+g	90.23	90.82	6.72	73.72
cls+cfive+g	90.23	90.82	6.84	73.09

表 3.2给出了在CASIA-HWDB测试数据集上使用不同组合上下文模型的识别结果。从表中可以看到，在二元文法**cbi**和三元文法**cti**下，本文的结果与 [45]的结果几乎一样，这说明本文对于参数和实现细节的控制是正确的。另一方面，可以发现，文本的识别速度相比 [45]快了50%，这是由于使用了更好的处理器的原因。通过对表中不同文法数的结果进行相互比较可以看到，尽管从二元文法到三元文法提升非常明显，但是从三元文法到更高阶的四元、五元文法对于识别系统性能的提升非常微弱。这可以印证本文在章节 3.1中提到的观点：高阶文法下，由于传统BLM受制于数据稀疏问题，因而难以很好地对高阶上下信息进行建模。

3.5.2.2　前馈神经网络语言模型

本文使用CSLM（v3）工具包 [121]进行前馈神经网络语言模型的训练，该工具提供对于短列表和GPU完整的支持。此外，在测试阶段，本文借助英特尔数学运算库MKL（Math Kernel Library）加速神经网络的前馈矩阵运算。对于每种神经网路结构，本文都从多个不同的初始值进行多次训练，最后选取能在验证集上得到最低困惑度的模型作为最终的模型。

本文对三种不同结构的前馈神经网络语言模型进行了评测，这些模型的词向量维度均为320，具体结构如表 3.3所示。这三种网络均采用随机梯度下降

表 3.3: 三种不同结构的前馈神经网络语言模型。

结构	隐层大小	短列表长度	初始学习率
FNNLM-1	1024×512	8330	0.06
FNNLM-2	1024×512	1023	0.06
FNNLM-3	512	1023	0.10

法进行优化，每一个批量大小（Batch Size）为128，权重衰减（Weight Decay）系数为10^{-7}，训练20轮即可收敛。比较这三个模型的结构可以看到，FNNLM-1和FNNLM-2都包含两个隐层，而FNNLM-3只有一个隐层。FNNLM-1模型输出层包含了词表中所有的词，而FNNLM-2和FNNLM-3使用短列表进行加速。模型学习率$lrate$的初始的设置方式如表 3.3所示，在训练过程中它遵循下面的方式减小：

$$lrate = lrate_0/(1 + \lambda n) \tag{3.26}$$

其中$lrate_0$表示初始学习率，n为已经经历的样本数，λ指的是衰减超参数，在本文中设置为5×10^{-8}。如章节 3.3.3所述，本文也将比较研究混合语言模型（HLM）的具体表现，混合语言模型通过神经网络语言模型与传统的回退语言模型进行线性插值得到。在本文中，插值系数通过SRILM工具包中的compute-best-mix-tool脚本计算得到，该脚本通过最小化验证集上的困惑度得到最优融合参数值。对应前面三种不同结构的前馈神经网络语言模型，有三种不同的混合语言模型，分别称为HFLM-1、HFLM-2和HFLM-3。

表 3.4给出了使用不同语言模型组合时过切分识别系统的表现。由于训练和测试FNNLM-1和FNNLM-2的各阶文法过于耗时，本文只给出这两种结构下五元文法的结果，一般来说五元文法都能比四元或者三元文法表现更好。将FNNLM与表 3.2中的传统回退语言模型进行比较可以发现，拥有复杂结构的FNNLM往往可以直接提升识别精度。在困惑度和识别精度上，FNNLM-1相比同阶回退语言模型有更好的效果。另一方面，尽管FNNLM一般可以得到比同阶回退语言模型更低的困惑度，但是这种优势对于识别性能的提升并不完全等价。此外，将FNNLM与BLM通过线性插值得到的混合语言模型可以在这两个模型的基础上更进一步降低困惑度和提升识别精度。特别地，五元HFLM-1模型可以将识别系统的AR和CR分别提升至90.69%和91.24%，然而，五元回退语言模型只能分别达到90.23%和90.82%。这说明神经网络语言模型对高阶信息的建模能力有了本质的提升。对于短列表长度为1023的FNNLM-2模型来说，

其计算复杂度相比FNNLM-1要小了不少。尽管识别表现上稍微降低了一些，但是它仍旧可以获得比回退语言模型更好的性能。同样，相应的混合语言模型HFLM-2在识别性能上相比HFLM-1也有所降低，但是其耗时相比HFLM-1降低了87.09%，同时其表现要优于单纯的五元回退语言模型和FNNLM-2。

对于FNNLM-3模型来说，它只包含一个隐层，也采用短列表技术进行加速，其时间复杂度要远远小于FNNLM-1和FNNLM-2。因而，就困惑度和识别精度上说，其表现也较这两者有明显的降低。使用FNNLM-3的识别系统得到的AR和CR甚至要低于同阶传统回退语言模型。然而，将其与BLM结合之后所得到的HFLM-3却可以在性能上有极大的提升。其识别率甚至可以和同阶的HFLM-2模型几乎一致，且计算时间可以减少很多。这些结果说明，一般情况下，为了在精度和时间复杂度上达到平衡，可以通过将结构简单的神经网络语言模型与传统回退语言模型进行插值从而得到混合语言模型来实现。

表 3.4: 前馈神经网络和混合神经网路语言模型的识别结果。

语言模型类型	模型组合	AR（%）	CR（%）	时间（h）	PPL
FNNLM-1	cls+cfive+g	90.29	90.88	129.30	68.60
HFLM-1	cls+cfive+g	90.69	91.24	140.32	59.44
FNNLM-2	cls+cfive+g	90.21	90.82	17.78	71.64
HFLM-2	cls+cfive+g	90.51	91.09	18.11	63.03
FNNLM-3	cls+cbi+g	89.59	90.30	9.06	141.39
	cls+cti+g	90.00	90.64	9.45	87.18
	cls+cfour+g	90.04	90.66	10.04	79.63
	cls+cfive+g	90.12	90.75	10.52	76.75
HFLM-3	cls+cbi+g	89.62	90.32	8.92	140.38
	cls+cti+g	90.33	90.92	9.73	76.35
	cls+cfour+g	90.39	90.97	10.21	66.83
	cls+cfive+g	90.49	91.07	10.81	64.66

3.5.2.3　递归神经网络语言模型

本文使用RNNLM（0.4b）[117]工具包进行递归神经网络语言模型的训练，该工具提供对于输出层分解、RNNME模型训练的完整支持，程序实现也极为高效，尽管没有使用GPU 进行加速。同时，为了与前馈神经网络语言模型进行公平的比较，本工作还修改了该工具包使其支持短列表的训练。由于RNNLM工具包不支持GPU加速，所以本文训练只使用短列表或者输出层加

速的递归神经网络语言模型，对于大词表来说，无法在可接受时间范围内训练得到全词表输出的递归神经网络语言模型。

为了深入研究隐层大小对于RNNLM的影响，本文首先训练了两种短词表长度均为1023的递归神经网络语言模型，分别为SRNNLM-1和SRNNLM-2，这两个模型的隐层大小分别为300和600。其余的参数如表 3.5所示。训练过程中，如果在验证集上的似然提升率小于一定的阈值，那么就将学习率减半。

表 3.5: 递归神经网络语言模型训练参数。

BPTT 步长	6
初始学习率	0.1
权值衰减	10^{-7}
最小提升率	1.003

本文将对同时都使用短列表加速的RNNLM和FNNLM进行深入的比较，此外，也将对这两种不同结构的模型所对应的混合语言模型进行比较。由于递归神经网络语言模型不再受制于N-gram模型，所以将其与低阶回退模型进行插值并没有太大的意义。因此本文只考虑将递归神经网络语言模型与传统五元回退语言模型进行插值。与上述SRNNLM-1和SRNNLM-2对应的混合语言模型分别称为HSRLM-1和HSRLM-2。识别结果如表 3.6所示。

表 3.6: 短列表递归神经网络语言模型与相应的混合语言模型的作用。

语言模型类型	模型组合	AR（%）	CR（%）	时间（h）	PPL
SRNNLM-1	cls+rnn+g	89.88	90.57	14.30	81.49
HSRLM-1	cls+rnn+g	90.43	91.02	14.42	61.44
SRNNLM-2	cls+rnn+g	90.30	90.94	28.74	65.19
HSRLM-2	cls+rnn+g	90.61	91.20	28.99	55.86

本文主要将SRNNLM-1和SRNNLM-2与五元FNNLM-2和FNNLM-3进行比较，这四个模型在复杂度上差别不大。根据网络的特性，可以计算这四个不同网络的参数大小，分别是SRNNLM-2(6M)>FNNLM-2(5M)>FNNLM-3(4M)>SRNNLM-1(3M)。将表 3.6中的SRNNLM模型与表 3.4中的FNNLM进行比较，可以看到SRNNLM-1在这四个模型中表现最差。但是，当它与五元BLM进行插值之后，得到的HSRLM-1比相应的五元HFLM-2和HFLM-3有更低的困惑度，同时其在字符串识别系统的表现也与这两者几乎一致。这表明SRNNLM模型可以相比FNNLM对于BLM有更好的互补性，因此可以使

用结构非常简单的SRNNLM组合成为性能比较强大的混合语言模型。尽管SRNNLM-2模型比FNNLM-2更复杂一些，但是它在性能上明显比FNNLM-2更好。当它们与传统回退语言模型进行插值之后，对应的HSRLM-2同样可以在困惑度和识别精度两个指标上好于HFLM-2。事实上，HSRLM-2模型甚至可以和表 3.4中的HFLM-1性能相当，但是五元FNNLM-1的参数规模大概有9M，而且其在计算复杂度上远远高于HSRLM-2。总的来说，这些实验结果表明在相同参数的情况下，递归神经网络语言模型要比前馈神经网络语言模型有更好的表现。而就计算复杂度来说，由于在识别解码阶段有关于隐层变量的频繁内存交换，递归神经网络语言模型比前馈神经网络模型较慢一些。

本文还将比较使用短列表加速的RNNLM与使用输出层分解加速的RNNLM（简写为FRNNLM）和RNNME模型。因为语料库中词表V的大小为8330，所以本文设置词类数为100。这与文献 [117]推荐的$\sqrt{|V|}$十分接近。在本节实验中，RNNME模型的隐层节点数设置为400，并且使用四元文法特征[3]，哈希表大小设为100M。使用输出层分解技术得到的RNNLM和RNNME模型以及它们所对应的混合语言模型的识别结果如表Table 3.7所示，其中FRNNLM-1和FRNNLM-2分别表示隐层为300和600的输出层分解RNNLM，HFRLM-1和HFRLM-2 是它们对应的混合语言模型，同样是与传统回退五元文法插值得到。HRMELM表示RNNME模型与五元BLM线性插值得到的混合语言模型。

表 3.7: 使用输出层分解技术的RNNLM和RNNME模型的识别结果。

语言模型类型	模型组合	AR(%)	CR(%)	时间(h)	PPL
FRNNLM-1	cls+rnn+g	89.57	90.26	15.95	87.67
HFRLM-1	cls+rnn+g	90.47	91.02	15.99	60.22
FRNNLM-2	cls+rnn+g	90.03	90.70	31.36	70.95
HFRLM-2	cls+rnn+g	90.61	91.16	31.61	55.30
RNNME	cls+rnn+g	90.89	91.38	16.44	56.50
HRMELM	cls+rnn+g	**91.04**	**91.52**	16.48	52.92
BLM	cls+iwc+g+cca* [45]	90.75	91.39	18.78	-

cca: candidate character augmentation

首先，与表 3.6中的短列表递归神经网络语言模型的结果相比，可以看到FRNNLM-1相比隐层节点数同样为300的SRNNLM-1性能要弱一些，这是由于其网络太小以致其不能在全词表范围内捕捉上下文信息。但是，混合模

[3]根据文献 [117]所示，对于最大熵模型来说四元文法特征已经足够用来建模。

型HFRLM-1则要优于HSRLM-1，这是因为通过输出层分解的带有完整词表的输出层可以给正确的识别提供更好的互补性。当将隐层大小增加到600时，FRNNLM-2再次表现得比SRNNLM-2更差，可是混合模型HFRLM-2和HSRLM-2却表现相当。在计算效率方面，短列表（Short-List）的方法显得比输出分解更有效一些。

接下来，可以比较表 3.7中的基于输出层分解的RNNLM和RNNME模型的性能。由于隐层大小为600的RNNLM仍然不足以对通用语料库中的所有字符进行建模，所以本文将RNNME模型引入识别系统。不难发现，单独的RNNME就可以极大地提高识别性能，甚至可以说优于混合模型HFRLM-2，尽管它的困惑度值略高于HFLM-2。事实上，在单独的RNNME模型的帮助下，识别系统的准确率已经高于文献 [45]报告的最优结果，该最优结果通过使用候选字符扩增技术（Candidate Character Augmentation，CCA）来提高切分识别网格中包含正确的候选字符类别的概率，而本文并没有使用CCA方法。通过与五元BLM进行插值，混合语言模型HRMELM可以得到AR为91.04%、CR为91.52%的最佳结果。

3.5.2.4 ICDAR-2013数据集识别结果

由于2013年ICDAR中文手写字符串数据集得到越来越多的关注，较多文献也将该数据集作为基准，所以本文也在ICDAR-2013上进行了相应的评测。这里主要考虑三种具有代表性的典型语言模型，分别是五元回退语言模型、RNNME模型以及HRMELM模型。语言模型和识别系统的设置与在CASIA-HWDB数据集上进行测试时是完全一致的。识别结果如表 3.8所示.

表 3.8: ICDAR-2013数据集识别结果。最好的结果已经用粗体标明。

语言模型类型	模型组合	AR(%)	CR(%)	时间(h)	PPL
BLM	cls+iwc+g+cca [45]	89.28	90.22	-	-
BLM	cls+cfive+g	89.03	89.91	2.44	73.09
RNNME	cls+rnn+g	89.69	90.41	5.86	56.50
HRMELM	cls+rnn+g	**89.86**	**90.58**	5.84	52.92

表 3.8同时也给出了二元词类插值语言模型结合候选字符扩增技术的结果。由于候选字符扩增的影响，其表现要比单纯使用五元文法更好。通过比较五元回退语言模型、RNNME模型以及HRMELM模型可以看到，这些模型

的精度优劣与表 3.7基本一致：RNNME模型比五元回退语言模型表现更好，而HRMELM模型是其中性能最好的模型。与同样配置下的当前最好结果 [45]进行比较发现，本文系统仅仅只用简单的基于字符的语言模型便可以将错误率相对降低了5.41%，这证明了神经网络语言模型的巨大作用。

3.6 小结

综上所述，本章对基于过切分识别系统中十分重要的语言模型模块进行了深入地研究。统计语言模型主要用于给出一个字序列合理程度的先验概率，可以让输出的最终识别序列更符合一般语法规律。一般情况下，由于高阶语言模型可以对更大范围内的上下文进行建模，因而会取得更好的效果。然而，传统回退语言模型受制于数据稀疏和维度灾难等问题，很难扩展到高阶模型。所以本章通过引入神经网络语言模型对高阶上下文进行建模，以便能进一步提升系统识别性能。

本章首先介绍了不同结构的神经网络语言模型，包括前馈神经网络语言模型和递归神经网络语言模型。神经网络语言模型的核心思想是将词语从离散空间中映射到连续空间中，从而可以进行隐式的平滑进而估计词语的出现概率。前馈神经网络语言模型本质上仍旧是一个基于N-gram的语言模型，但是递归神经网络语言模型通过递归连接理论上可以对任意范围内的上下文进行建模。其次，介绍了神经网络语言模型的两种加速算法，包括基于短词表和基于输出层分解的方法。虽然神经网络语言模型有很好的性能，但是其计算复杂度远大于传统语言模型。经过分析，其计算耗时主要集中在输出层。因而两种方法都是针对输出层的改进。短词表方法将输出层限定为词表中的高频词，而低频词通过传统语言模型进行估计；输出层分解的方式则将词表中的词聚成不同的词类进而减少输出层节点的数量级。最后，本章进行了大量充分的实验验证神经网络语言模型的有效性，同时也比较了前馈网络和递归网络的优劣，以及不同加速方法对于性能和速度的影响。实验表明，递归神经网络语言模型相比前馈模型能获得更好的性能，而混合语言模型可以进一步提升语言模型的性能，同时，基于输出层分解的技术可以有更大潜力。最终，通过融入混合递归神经网络语言模型，使得过切分识别系统的精度超越了当前最好结果。

在本章中，自始至终使用的都是最简单的基于字符的语言模型，但是对于中文文本来说，词语是更有意义的语言单元，所以将来可以考虑融入基于词的神经网络语言模型，以便能更好地提升系统识别精度，当然这样也会进一步增

加系统的计算复杂度。另一方面，语言模型在集束搜索中的主要作用是对于切分识别网格的路径进行打分，但是如果形状模型本身不能给出包含正确路径的网格，那么语言模型的作用就无从谈起，所以对于形状模型的改进也是势在必行。这就是第四章工作的一个基本出发点。

第四章　基于卷积神经网络形状模型的显式切分识别方法

4.1　引言

本章继续讨论如何对基于显式切分的中文手写字符串识别系统进行改良。除了本文第三章提到的语言模型之外，过切分算法、字符分类器以及几何模型同样在识别系统中扮演着十分重要的角色。由于这三种模型都是从图像层面对文本行进行建模，所以本文将它们合称为形状模型。

对于连续书写的文档图片，由于书写人的书写随意性，字符之间往往存在粘连。过切分算法的目标就是在切分开粘连字符的同时，产生较少数量的冗余切分点。现有的切分算法虽然能获得不错的切分精度，但是基本都是基于启发式规则的方法，其鲁棒性较差。有少数的基于统计学习的方法，但是它们只能处理中文文本行中单粘连的情况，对于复杂情况的研究还不够充分。单字分类器的设计是文字识别领域一个非常热门的研究内容。早期的识别流程都是基于特征提取结合统计学习分类器的方式，也可以取得较高的识别率。近几年，基于卷积神经网络（Convolutional Neural Network，CNN）的图像分类方法已经开始占据主导地位，使用CNN进行字符分类的精度也已经超过了人类识别精度。但是目前只有极少数的工作对于其在字符串识别系统中的作用有过研究。因为在单字分类时，字符经过归一化等操作后，几何上下文信息会全部丢失。在字符串识别中，还要考虑众多的上下文依赖关系，所以如果能将这类信息补充进去，可以进一步提升识别性能。在中文手写字符串中，几何模型的融入可以明显地提升英文字母、标点符号等字符的分类精度。但是设计几何特征的过程相当繁杂，目前也缺乏基于深度网络的几何模型的研究工作。

针对以上问题，本章提出使用卷积神经网络形状模型对现有过切分识别系统进行改进。具体贡献包括以下几点：

1. 在过切分模块中，提出将传统算法与CNN分类器滑动窗相结合的方法，可以在精度仅有微弱下降的情况下大幅提升切分召回率。
2. 对于字符分类器，提出将领域知识融入CNN模型中，并且设计了可以应对大类别集的新结构，使得精度有了大幅提升。

3. 对于几何模型，提出通过曲线拟合的方式估计候选字符在文本行中的位置，并进而使用CNN进行建模。
4. 提出使用网格正确率估计系统识别上限，并且对基于卷积神经网络形状模型的过切分识别系统进行了评测。

本章内容安排如下：在4.2节中介绍包含过切分模块、字符分类器以及几何模型在内的卷积神经网络形状模型；4.3节详细描述了实验流程和结果并验证卷积神经网络形状模型的有效性，此外还给出网格正确率的计算方法并给出评测结果和分析；4.4节对本章内容做出小结。

4.2 卷积神经网络形状模型

随着深度学习技术在各个领域获得成功 [105, 122]，本文同样考虑使用深度学习方法改良包括字符分类器、过切分模块以及几何模型在内的形状模型，下面将分别对它们进行介绍。

4.2.1 字符模型

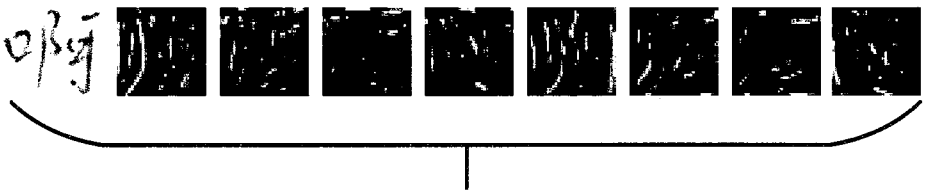


图 4.1: CNN字符模型输入示意图。

本文构建了一个15层的CNN作为字符分类器，其结构如表 4.1所示，该结构与文献 [18]提出的结构类似。与文献 [123]将领域知识融入CNN网络类似，本文提取的是$9 \times 32 \times 32$扩展方向特征图，包含八方向非线性归一化图像和保持宽高比归一化成大小为32×32的原始字符图像，如图 4.1所示。提取梯度图的具体做法是：首先对原始字幅图像进行线密度插值归一化 [7]，然后提取八方向梯度特征，特征平面的大小设置为32×32。对于梯度方向分解，先使用Sobel算子在原图中进行边缘检测，接着使用平行四边形法则将边缘梯度分解到八个方向。实验表明，使用特征图作为额外的输入平面，可以极大地增强

表 4.1: CNN字符分类器结构. 第一行对应网络的底层，maps、k、s和p分别表示卷积平面特征数、卷积核（Kernel）大小、步长（Stride）以及补齐（Padding）大小，Window表示池化层（Pooling）的窗口大小。下同。

网络层类型	配置
输入层	$9 \times 32 \times 32$ 扩展方向特征图
卷积层	#maps: 50, k: 3×3, s:1, p:1, dropout: 0.0
卷积层	#maps: 100, k: 3×3, s:1, p:1, dropout: 0.1
卷积层	#maps: 100, k: 3×3, s:1, p:1, dropout: 0.1
最大池化层	Window: 2×2, s: 2
卷积层	#maps: 150, k: 3×3, s:1, p:1, dropout: 0.2
卷积层	#maps: 200, k: 3×3, s:1, p:1, dropout: 0.2
卷积层	#maps: 200, k: 3×3, s:1, p:1, dropout: 0.2
最大池化层	Window: 2×2, s: 2
卷积层	#maps: 250, k: 3×3, s:1, p:1, dropout: 0.3
卷积层	#maps: 300, k: 3×3, s:1, p:1, dropout: 0.3
卷积层	#maps: 300, k: 3×3, s:1, p:1, dropout: 0.3
最大池化层	Window: 2×2, s: 2
卷积层	#maps: 350, k: 3×3, s:1, p:1, dropout: 0.4
卷积层	#maps: 400, k: 3×3, s:1, p:1, dropout: 0.4
卷积层	#maps: 400, k: 3×3, s:1, p:1, dropout: 0.4
最大池化层	Window: 2×2, s: 2
全连接层	#hidden units: 900 , dropout: 0.5
全连接层	#hidden units: 200, dropout: 0.0
Softmax层	#units: 7357

分类器的鲁棒性。在上述网络中，一般都是用3×3 的小卷积核，并且所有卷积的步长都设置为1，卷积平面数从50缓慢递增到400。为了使网络层数尽可能深一些，本文的网络中每三个卷积层之间才插入一个空间池化层。本文的网络使用最大池化层，每次该层都将特征平面的大小减半。在经过12个卷积层和4个池化层之后，将特征平面展平成1600 维向量送入后面的两个全连接层，全连接层的大小分别为900和200。最后，Softmax归一化层对将7357个输出节点的值变换到0到1之间的概率值。7357类中，7356个类别为字符类，还有一个是非字符类，用于显式地对非字符模式进行建模。非字符类在切分候选网格中十分常见 [124]，作为一个判别模型，CNN模型对于异常类的建模性能并不好，需要显式地加入样本才能使CNN对非字类有拒识的效果。此外，文献 [125]发现，这种增加一个非字类的方式比使用级联CNN的方法效果更好。

4.2.2 过切分算法

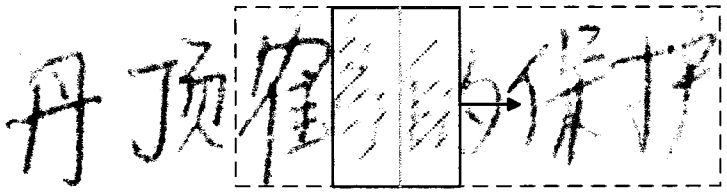


图 4.2: 滑动窗过切分算法。

过切分算法的作用是将文本行图像切分为一系列的基元片段，这些基元片段对应一个字或者是字的一部分，因而连续一个或几个部件组合在一起就可以构成一个完整的字符。尽管中文识别中基于启发式的连通部件分析方法已经得到广泛应用，特别是基于轮廓形状分析的方法 [1]已经在许多工作中得到了成功应用 [45, 97]，但是汉字字符的切分仍旧还有较大的改进空间，特别是对于复杂粘连的情况这些方法并不鲁棒，如图 4.3(a)所示。

为了进一步提升切分点召回率，本文提出一种两步过切分算法，算法流程为:

1. 首先，使用基于前景点可见性分析的粘连字符切分算法 [2]对文本行图像进行初始的切分，将两个连续切分段之间的位置视作一个候选切分点。

2. 接着，使用滑动窗在上一步骤中得到的候选切分片段上生成一系列相同大小的窗口，并且使用一个二值CNN分类器对这些窗口进行分类，判断窗口的中心是否属于合法切分点。由于这一步会产生一些冗余的切分点，所以需要一个简单的规则来合并相近的切分点。

使用滑动窗进行过切分的方法比较简单，为了进一步提升其性能，本文主要有两方面的创新：第一，本工作使用基于前景点可见性分析的切分算法对文本行进行预切分，可以将粘连程度较轻的文字先切开，在减少后面滑动窗分类器的分类代价的同时也加速了后续操作；第二，为了获得更高的切分点检测率，本文采用基于CNN的网络进行二值分类。下面主要介绍第二步的详细做法。

对于文本行图像中的一个连通部件，用一个步长为0.1倍连通部件高度的窗口从左往右滑动，如图 4.2所示。通过CNN分类器来判断窗口的中心是否是一

个合法切分点。滑动窗的高度与连通部件高度一致，其宽度取0.8倍的连通部件高度。在实验中，本文发现滑动窗步进系数取0.04到0.1，以及窗口宽度系数取0.6到1.0之间的值都可以使切分识别效果有稳定的提升。

当使用CNN训练切分点分类器时，如果使用类似于表 4.1中的复杂网络则很容易造成过拟合。因此，本文构建了一个简单的4层CNN网络用于切分点二值分类，如表 4.2所示。该网络同样使用章节 4.2.1中介绍的扩展方向特征图作为输入。在训练过程中，如果切分点位于窗口中心，那么该窗口就被认为是正样本，否则为负样本。在滑动窗过切分之后，需要对距离较近的候选切分点进行合并。本文的合并策略是，对一定水平距离内的切分点进行分析，取其中前景像素点竖直投影值最小的点作为合并的最终结果。一般来说，待合并的切分点的距离范围为1倍的笔画宽度，而笔画宽度则通过文本行图像的前景轮廓分析得到。图 4.3给出了不同方法下的过切分效果。可以看到，文献 [2]方法的召回率比文献 [1]的方法更好，而本文提出的方法可以取得相比 [1]和 [2]更好的切分召回率。

表 4.2: 过切分网络配置。

网络层类型	配置
输入层	$9 \times 32 \times 32$ 扩展方向特征图
卷积层	#maps: 50, k: 3×3, s:1, p:1, dropout: 0.0
最大池化层	Window: 2×2, s: 2
卷积层	#maps: 100, k: 3×3, s:1, p:1, dropout: 0.1
最大池化层	Window: 2×2, s: 2
全连接层	#units: 200
Softmax层	#units: 2

4.2.3　几何上下文模型

几何上下文模型已经成功地用于字符串识别 [45, 51]以及文本对齐 [52]领域，它在排除非字符进一步提高系统性能方面发挥重要作用。在本文中，我们采用文献 [52]提出的几何上下文模型框架，它将几何模型分为四个统计模型，分别为一元类别相关、二元类别相关、一元类别无关、二元类别无关几何模型，相应的简写为**ucg**、**bcg**、**uig** 和**big**。

由于针对单字分类器设计的归一化操作可能会失去与风格相关的上下文信息，而几何模型是针对没有归一化的原始文本行图像进行建模，所以它可以看

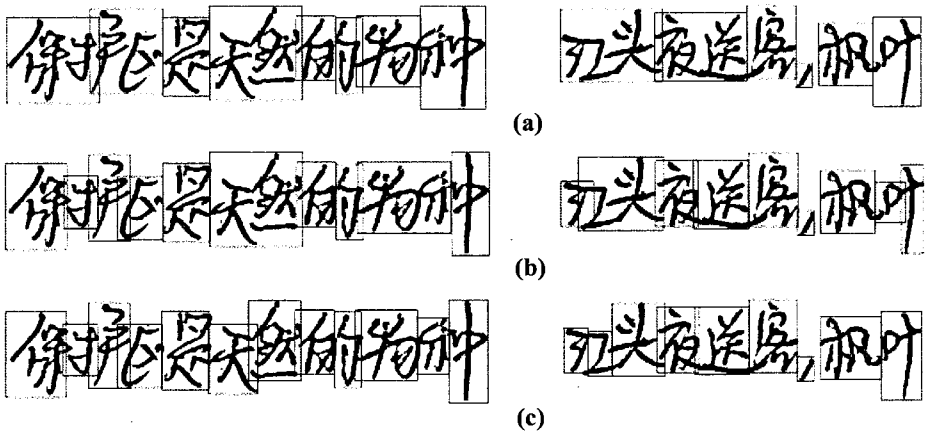


图 4.3: 不同方法的过切分效果图。(a) 文献 [1]方法结果；(b) 文献 [2]方法结果；(c) 本文结果。

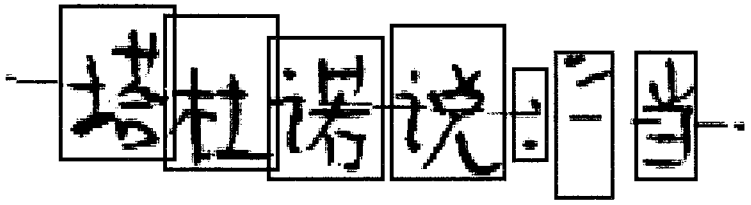


图 4.4: 文本行多项式拟合示意图。

做是对字符分类器的有效补充。根据文献 [52]，本文将字符根据几何信息聚为六个超类。另一方面，一元类别无关几何模型主要用来度量候选模式是否是有效字符，而二元类别无关模型用于度量过切分的间隔是否是有效的字符间隔。因而，这两种模型都是二分类模型。

为了对四个几何模型进行建模，传统的做法都是首先提取几何特征，再使用二次判别函数（用于**ucg**和**bcg**）或者支持向量机（用于**uig**和**big**）进行分类，最后通过置信度转换将分类器的输出转化为概率。由于几何模型的特征设计比较繁复，本文利用CNN对特征提取和分类进行联合训练，然后直接使用特定单元的输出作为最终得分。由于几何上下文模型需要保持文本行的书写风格，因而不能采用将候选模式进行尺寸归一化之后直接送入CNN的方式。本文提出首先通过多项式拟合获得文本行的中心曲线，如图 4.4所示。多项式的阶数设置为连通部件数目的0.075倍。之后，根据中心曲线和字符高度对每个连通部件的

顶部和底部边界进行调整。在这里，除了输出层单元数不同，其余部分本文使用了与 [18]相同的CNN结构。为了保持书写风格，这里只使用原始的连通部件图像作为输入而不使用归一化方向特征图。

4.3 实验

为了验证本章提出的卷积神经网络形状模型的作用，本节将给出替换传统模型之后的识别结果。本实验使用开源工具Caffe [126]训练包含字符分类器、过切分分类器以及几何模型在内的卷积神经网络形状模型。本章首先介绍三种模型的训练细节，然后将形状模型融入字符串识别系统与传统模型进行对比。此外，本章实验的最后还将进行两个综合实验，一个是将通用语料库扩展到大语料库上之后，融合现阶段各个最好的模型观察系统能达到的最高精度，另一个是通过引入网格正确率来探究系统识别精度的上限。

4.3.1 卷积神经网络形状模型的评测

本小节首先介绍三种模型的训练细节，然后将形状模型融入字符串识别系统与传统模型进行对比。需要说明的是，对于卷积字符分类器和几何模型，本文直接将它们相应的Softmax层的输出作为最终模型的打分，而不再进行置信度转换。此外，本节使用与第三章相同体系的语言模型。

4.3.1.1 字符分类器

CNN字符分类器使用Xavier方法 [127]进行初始化。训练准则为多类负似然对数损失（Multi-Class Negative Log-Likelihood Loss），训练过程使用带动量（Momentum）随机梯度下降法进行优化，每批样本数量为1024，动量参数大小为0.9。网络的初始学习率设置为0.01，当训练集上损失不再下降时，就将学习率减半。整个训练大概在90轮左右结束。

在多次尝试中，本文发现尽管数据增强（增加扰动样本）不能明显提升单字识别率，但是却可以提升字符分类器在文本行识别中的表现，这可能是由于连续自由书写的文字形变更大。本文使用文献 [13]介绍的字符扩充技术，主要包含几何变换、局部缩放以及弹性形变等几种方式对字符进行混合变形处理，如图 4.5所示。

通过上述方式，本文加入了两倍扰动样本，即训练集一共包含了12595460个字符样本。另一方面，本文还加入了非字样本使得字符分类器对于异常样本有

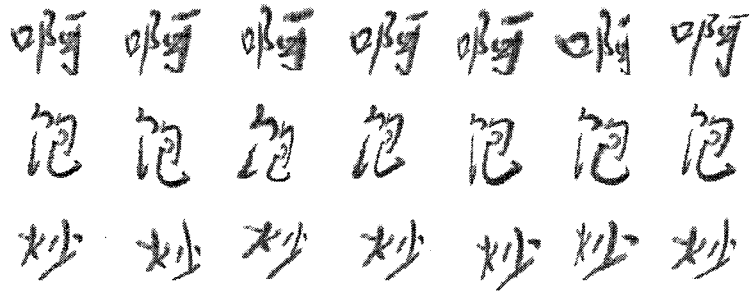


图 4.5: 扩充样本示例。第一列为真实样本，其余为扰动样本。

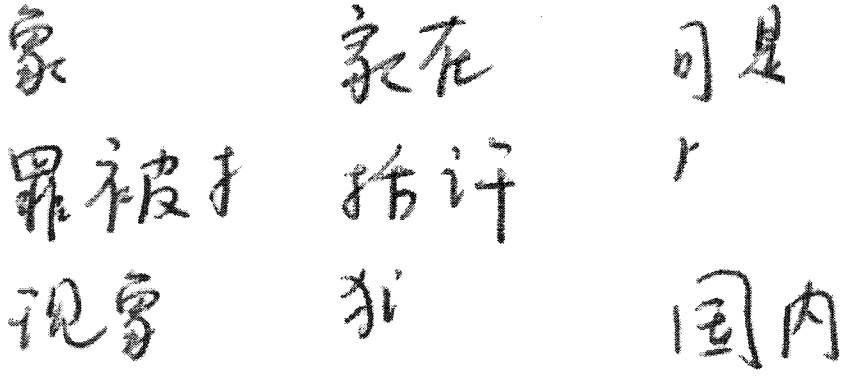


图 4.6: 非字样本示例。共有三列样本。

拒识能力。由于CASIA-HWDB数据集把每个字符的位置都标记出来，所以可以首先使用过切分算法在训练集文本行上切分，然后根据字符边界生成非字样本，如图 4.6所示。本文从训练文本行中生成了5160425非字样本。尽管就每一类的样本而言，加入非字类样本以后存在着严重的类别不均衡问题，但是在实际的实验过程中，并没有发现这对于系统的表现有明显的负面影响。随着训练轮数的增加，网络还是可以对各种类别正常建模。因此，本文没有使用任何方式去处理类别不均衡问题。最终训练出来的分类器在CASIA-HWDB数据集文本测试集上的字符分类正确率为92.17%，与文献 [45]使用MQDF分类器得到的83.78%相比，性能有了明显的提升。

4.3.1.2 过切分算法

本文提出的过切分算法涉及到滑动窗分类器，我们使用CASIA-HWDB数据库中的训练集训练卷积滑动窗分类器。由于数据集中的所有文本行已经做了字符级别的切分和标注，所以能很方便地获取分割点的真值。为了生成训练

样本，首先，我们在文本行中的连通部件上进行划窗。当某个窗口的中心位置与字符的边界之间的距离小于当前连通部件高度的0.1倍，该窗口被认为是正样本；当该距离大于连通部件高度的0.12倍时，窗口则被认为是负样本。否则，窗口被认为是不明确的样本，会增加分类边界的模糊性，因而不能用于训练。

CNN滑动窗分类器的初始化与训练策略与训练卷积字符分类器一致。一般来说，文本行中切分得到的负样本数（1870534）比正样本数要多很多。为了克服类别不平衡问题（事实上，甚至可以接受负样本被分为正样本）并且尽可能提升切分召回率，本文将负样本损失调整为原来的两百分之一（0.5%）。训练一般可以在100轮左右完成。

表 4.3: CASIA-HWDB文本行测试集上的过切分结果。

模型类型	精度（%）	召回（%）
[1]	74.32	98.23
[2]	68.07	99.22
只有滑动窗过切分	63.75	99.39
本文的方法	64.23	99.58

在评测文本行过切分性能时，本文采用切分点召回率和精度来衡量。文本使用切分点检测召回率和精度来衡量文本行过切分算法的性能。当滑动窗分类器判断某一窗口为正样本时，如果该窗口中心与实际字符边界距离小于两倍笔画宽度（在该文本行中估计得到），那么其中心点就被认为是一个正确的切分点，否则就认为是被误识的样本。表 4.3给出了CASIA-HWDB文本行测试集上各种切分算法的精度和召回率。可以看到，文献 [2]的方法可以取得比文献 [1]方法更高的召回率，同时精度只有微弱的下降。当使用滑动窗过切分时，召回率可以与文献 [1, 2]的方法相比都有进一步地提升。本文使用将文献 [2]方法和滑动窗分类方法相结合的框架，可以取得最高的召回率，进而可以使得过切分框架有更高的识别上限。由于本文的算法生成了相对较多的连通部件，所以在解码阶段，我们将最大部件连接数设置为7（之前是4）。

4.3.1.3　几何上下文模型

由于CASIA-HWDB2.0-2.2的数据集是标记到字符级别的强标记数据，因此很容易就可以从数据集中生成四种几何模型（**ucg**、**uig**、**bcg**和**big**）的样本。本文对这四种模型分别生成了1081153（**ucg**），7498977（**uig**），1221326

（**bcg**）以及1331428（**big**）个样本。网络模型的初始化和训练流程与CNN字符模型相似，这里便不再赘述。

4.3.1.4 卷积字符模型的实验结果

上面我们分别介绍了卷积神经网络形状模型中各个模型的训练方法以及单独的精度评测。本节，我们将它们融入字符串识别系统以便验证模型的表现。本节使用与第三章相同的语言模型体系。为了使得结果更可信，我们也讨论了形状模型与不同语言模型相结合的效果。基于第三章的论述，这里我们只给出最好的神经网络语言模型，即混合递归神经网络最大熵语言模型（HREMLM）的融合识别结果。表 4.4给出了使用不同形状模型在CASIA-HWDB和ICDAR-2013的识别结果。

表 4.4: 使用不同形状模型在两个数据集上的识别精度。

数据集	形状模型	语言模型类型	模型组合	AR(%)	CR(%)	时间(h)
CASIA-HWDB	传统	BLM	cls+iwc+g+cca [45]	90.75	91.39	18.78
		BLM	cls+cti+g+lma* [97]	91.73	92.37	10.17
		BLM	cls+cfive+g	90.23	90.82	6.84
		HRMELM	cls+rnn+g	91.04	91.52	16.48
	CNN	BLM	cls+cfive+g	95.05	95.15	8.67
		HRMELM	cls+rnn+g	**95.55**	**95.63**	16.83
ICDAR-2013	传统	BLM	cls+iwc+g+cca [45]	89.28	90.22	-
		BLM	cls+cfive+g	89.03	89.91	2.44
		HRMELM	cls+rnn+g	89.86	90.58	5.84
	CNN	BLM	cls+cfive+g	94.51	94.64	2.96
		HRMELM	cls+rnn+g	**95.04**	**95.15**	6.68

在表 4.4中，比较传统模型和本文提出的卷积模型，可以发现卷积神经网络形状模型在性能上有极大的优势。相对于传统形状模型在两个数据集上的表现，卷积模型可将字符错误率（等价于$1-AR$）相对降低近50%。我们还发现，由于本文的方法都可以使用GPU加速，卷积字符模型的识别系统在时间复杂度上仅比传统方法高一点。此外，不同语言模型之间的比较也再次证明了神经网络语言模型的优势。将卷积字符模型与HRMELM结合之后，在CASIA-

HWDB数据集上AR和CR分别可以达到95.55%和95.63%；在ICDAR-2013数据集上AR和CR 分别为95.04%和95.15%。与之前的方法相比，本文的方法并没有使用复杂度较高的动态字符扩增技术，或者是需要进行两边识别的语言自适应的方法（文献 [97] 的方法还使用了更大的语料库）。

为了更进一步说明本文方法的优势，我们将其与其他几个典型的工作进行了对比。在ICDAR-2013竞赛中 [4]，文献 [45]可以得到AR为89.28%、CR为90.22% 的最好结果。文献 [98]使用基于递归神经网络的识别框架（原始框架由 [31]提出）可以在中文手写字符串上取得令人鼓舞的结果，在ICDAR-2013数据集上的可以达到89.40%的AR，可是该结果相比本文的结果还有一定的差距。近期，文献 [125]使用与本文类似的框架，也可以达到很高的识别精度，在CASIA-HWDB数据集上AR和CR 分别为95.21%和96.28%，在ICDAR-2013数据集上AR和CR分别为94.02%和95.53%。尽管本文的CR比该方法略低一些，但是可以获得明显更高的字符准确率，而这一般认为是一个更合适的度量。另外，需要说明的是在ICDAR-2013数据集上进行测试的时候，文献 [98, 125] 在实际系统实现中都删除了部分特殊字符，而本文没有进行任何的删除操作。

4.3.2　综合实验

综合第三章和第四章内容，本文通过神经网络语言模型和卷积神经网络形状模型极大地提升了系统识别性能。本节主要介绍两个扩展的综合实验。首先，将通用语料库扩展为大语料库，融合现阶段各个最好的模型观察系统能达到的最高精度；其次，引入网格正确率探究系统识别精度的上限，并对目前的系统作错误分析。

4.3.2.1　大语料库

为了更好地对语言上下文进行建模，本文使用更大规模的包含16亿个字符的语料库（搜狗实验室提供）训练语言模型（第三章的语料库仅包含五千万字符），文献 [97]曾使用该语料库进行过语言模型自适应的研究工作。对于传统语言模型，这里我们同样使用Katz算法进行平滑，并将剪枝阈值设置为10^{-7}。由于在大型语料库上训练神经网络语言模型耗时过长，所以我们只使用在包含五千万个字符的通用语料库上训练得到的NNLM，然后将其与在大语料库上训练的BLM 相结合。特别地，我们使用在第三章中通用语料库上训练得到

的RNNME 模型，并将其与在大语料库上训练的五元传统回退语言模型进行组合来得到混合语言模型模型HRMELM。在两个数据集上的识别结果见表 4.5。

表 4.5: 使用大语料库语言模型的识别结果。

数据集	语言模型类型	形状模型	模型组合	AR(%)	CR(%)	时间(h)	PPL
CASIA-HWDB	BLM	传统	cls+cti+g+lma [97]	91.73	92.37	10.17	-
			cls+cti+g [97]	90.66	91.28	9.83	-
			cls+cfive+g	90.79	91.41	6.88	65.21
		CNN	cls+cfive+g	95.36	95.46	8.91	65.21
	HRMELM	传统	cls+rnn+g	91.64	92.11	16.57	46.25
		CNN	cls+rnn+g	**95.88**	**95.95**	16.83	46.25
ICDAR-2013	BLM	传统	cls+cfive+g	91.48	92.17	2.44	65.21
		CNN	cls+cfive+g	96.18	96.31	2.93	65.21
	HRMELM	传统	cls+rnn+g	91.59	92.23	5.93	46.25
		CNN	cls+rnn+g	**96.20**	**96.32**	5.93	46.25

从表 4.5中，我们可以主要观察到三点。首先，与在较小的通用语料库上训练的高阶语言模型的表现不同，由于大语料库减轻了数据稀疏问题，五元BLM在大语料库上的性能显著优于三元文法。其次，虽然大型语料库提高了五元BLM的性能，但是RNMME模型仍然显著提升了混合语言模型HRMELM的性能：与五元BLM相比，HRMELM降低了9.23%的错误率。最后，卷积神经网络形状模型与大型语料库语言模型结合可以进一步提高了系统性能，因为它不仅有更大的包含正确候选字符模式的潜力，还有更强的分类能力。与使用语言模型自适应的先前最好的结果相比，本文使用HRMELM和卷积神经网络形状模型的方法将AR降低了足足有4.15 %。

值得注意的是，在表 4.5中，当使用大型语料库来训练语言模型时，在ICDAR-2013数据集上，HRMELM相比BLM并没有明显的优势，是因为我们发现ICDAR-2013数据集的文本真值都被包括在大语料库中。因此，基于词频统计的BLM可以很好地适应测试数据并极大地提升识别精度。为了进一步弄清这个问题，我们从大型语料库中删除了ICDAR-2013数据集的文本真值中出现的句子。然而，由于该语料库的主题非常典型和集中，我们在处理后语料库上训练五元BLM仍然可以得到96.16%的AR和96.29%的CR。这提醒我们在实际实验过程中，还需要注意测试文本图像的文本真值与语言模型之间的过拟合问

题，而另一方面，神经网络语言模型可以很好地推广到之前没有见过的文本内容中。

4.3.2.2　过切分识别系统深入分析

经过第三章和第四章的叙述，尽管实验结果表明神经网络语言模型和卷积神经网络形状模型对系统性能的提升有很大的帮助，但是与人类的识别能力相比仍然存在一定的差距。因此，本节我们将首先分析识别系统的性能上限，然后给出一些文本行识别的具体示例。

为了给系统的识别上限提供一个定量指标，我们考虑采用网格错误率（Lattice Error Rate，LER）来评估切分识别中候选网格的优劣。网格错误率是字符错误率的下界，它被定义为网格错误总数除以文本真值中的字符总数。计算LER的规则可以通过下面的算法来实现。

对于一个文本行（字符串序列）S，其对应的文本真值为$\mathbf{y}_{1:N} = y_1 \cdots y_N$，我们用$\mathbf{G}$表示$S$生成的网格。网格错误率可以通过计算$\mathbf{y}_{1:N}$和$\mathbf{G}$之间的距离得到，即定义为$\mathbf{y}_{1:N}$和$\mathbf{G}$中任意符号序列$Y$之间的最小编辑距离：

$$Dist(\mathbf{y}_{1:N}, \mathrm{G}) = \min_{Y \in \mathbf{G}} Dist(\mathbf{y}_{1:N}, Y). \tag{4.1}$$

令$\mathbf{y}_{1:i}, i \leq N$为部分符号序列，$j \in \{0, ..., T\}$为文本行候选切分点，其中0是起始点，$T$是结束点，$\mathbf{G}_{0:j}$作为切分点0和$j$之间的部分网格路径。那么$\mathbf{y}_{1:i}$和$\mathbf{G}_{0:j}$之间的距离$D(i,j)$就可以通过下面的动态规划算法递归计算得到：

1. 初始化

$$\begin{aligned} &D(0,0) = 0, \\ &D(i,0) = i, \text{ for } 1 \leq i \leq N, \\ &D(0,j) = \min_{k:(k,j) \in \mathbf{G}} D(0,k) + 1, \text{ for } 1 \leq i \leq T \end{aligned} \tag{4.2}$$

2. 递归计算. For $1 \leq i \leq N, 1 \leq j \leq T$,

$$D(i,j) = \min \begin{cases} \min\limits_{k,Y_{(k,j)}:(k,j)\in \mathbf{G}} D(i-1,k) + 1 - \delta(y_i, Y_{(k,j)}), \\ D(i-1,j) + 1, \\ \min\limits_{k:(k,j)\in \mathbf{G}} D(i,k) + 1, \end{cases} \tag{4.3}$$

3. 算法终止

$$Dist(\mathbf{y}_{1:N}, \mathrm{G}) = D(N,T), \tag{4.4}$$

其中，$(k,j)(k < j)$表示在切分点k和j之间的一个候选字符模式，$Y_{(k,j)}$表示(k,j)的一个候选类别。

从上面的计算过程中我们可以发现网格错误率受到过切分算法和候选字符分类模块的影响，这两个模块分别有生成字符切分和分配字符种类的作用。我们将传统形状模型和卷积神经网络形状模型分别融入识别系统比较这两种不同模型下网格错误率的情况。为了便于直观理解，我们在表 4.6中给出了网格精度（Lattice Accuracy Rate，LAR）的度量，表示为1 - LER。从表中我们可以看到，卷积神经网络形状模型，特别是字符分类器和过切分算法，显著提高了两个数据集上的LAR（降低了LER）。根据文献 [45]所述，在CASIA-HWDB的文本行测试数据集中，MQDF分类器在字符分类中前20候选（Top-20）累积精度为98.24%，而我们的CNN字符分类器的top-20累积精度则达到了99.75%。总的来说，实际识别准确率（在我们的实验中通常小于96%）和网格正确率之间的差距意味着在提升系统对切分识别候选网格的上下文建模方面还有很大的改进空间。

表 4.6: 在两个数据集上的网格正确率。

形状模型	CASIA-HWDB LAR（%）	ICDAR-2013 LAR（%）
传统	96.65	96.16
CNN	99.20	99.27

下面，我们展示一些文本行识别的例子，如图 4.7所示。这里主要考虑两个典型的情况，分别是五元BLM结合传统模型，以及最好的语言模型HRMELM与卷积神经网络形状模型相结合。这四个例子证明了语言模型与形状模型的有效性。图 4.7(a)中的识别错误也在文献 [97]中有出现，该错误不能通过语言模

型自适应的方式进行纠正，但是它可以通过使用HRMELM 进行高阶上下文建模的方式得到纠正。在(b)中，基于CNN 的字符分类器和几何模型可以更好地对图像上下文进行建模，因而可以纠正该错误。在(c)中，基于CNN的过切分方法可以处理复杂粘连的情况，因而本文的方法可以纠正该错误，而传统方法在此时并不具有很好的鲁棒性。本文的模型还不能纠正(d)中的错误，这说明候选切分识别的路径评价函数还有提升的空间。

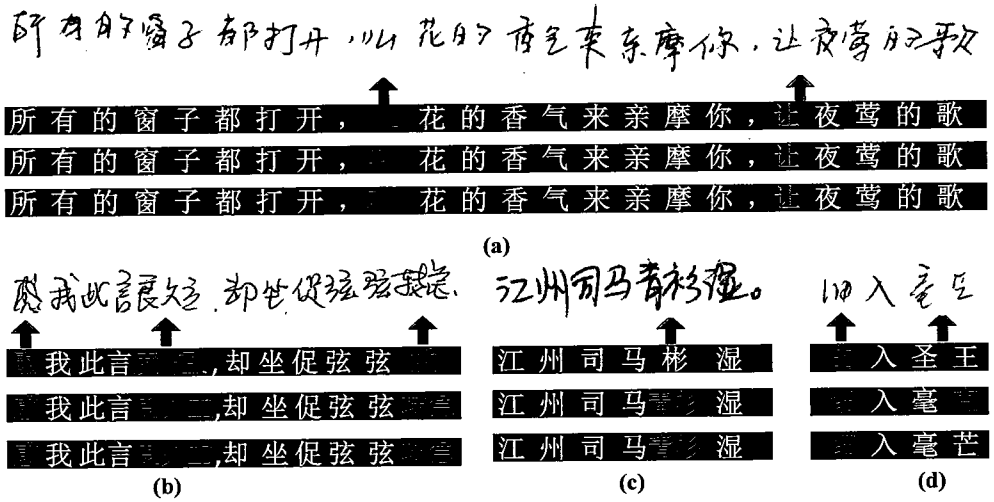


图 4.7: 过切分系统识别示例。对于每一个样本，第一行表示文本行图像，第二行表示使用传统五元BLM与传统形状模型的结果，第三行表示使用HRMELM和卷积神经网络形状模型的结果；第四行表示文本行图像对应的文本真值。

4.4 小结

本章主要从图像层面提出改进过切分识别系统性能的算法。过切分算法、字符分类器以及几何模型在识别系统中扮演着十分重要的作用，本文将它们合成为形状模型。

虽然传统的切分算法能获得不错的效果，但是在处理复杂粘连情况时鲁棒性较差。而对于单字分类器，虽然基于卷积神经网络的分类器在孤立字符识别上已经超过了传统分类器，但是目前只有极少数的工作将其应用于字符串识别系统中。同时，因为在单字分类时，字符经过归一化等操作，几何上下文信息会全部丢失。因此在字符串识别中，几何上下文模型可以将这类信息补充进

去，进一步增强识别性能。但是传统几何模型的特征设计过程相当繁杂，目前也缺乏基于深度网络的几何模型的工作。

针对以上问题，本章首先介绍了包含以上三种模型的卷积神经网络形状模型。对于字符分类器，本文构建了一个15层的CNN作为字符分类器，并且提取$9 \times 32 \times 32$扩展方向特征图作为网络输入，这种融入领域知识的方式可以进一步提升分类器的分类精度。此外，通过在输出节点增加一个非字类别的做法，可以让CNN具有对非字的拒识能力。对于过切分算法，本文提出两步过切分算法，将基于前景点可见性分析的粘连字符切分算法与滑动窗算法进行结合。CNN滑动窗过切分算法通过在文本行图像上进行滑窗并进行CNN分类的方式可以进一步提升切分点的召回率，在处理复杂粘连时有很好的效果。对于几何模型，为了保持书写风格，本文通过曲线拟合的方式估计候选字符在文本行中的位置，并进而使用CNN进行建模，对CNN字符分类器有很好的互补性。其次，在综合实验部分，本文给出了度量过切分识别系统的网格正确率的计算方式，并通过该算法得到目前系统的识别上限。实验表明，虽然目前改良的系统可以有很高的识别精度，但是离人类的识别水平还有较大的差距。

综合第三章和第四章的内容，本文从语言模型和形状模型两个方面对过切分识别系统进行了大幅改进，字符错误率相对降低了近50%，将来的改进可以考虑设计更好的路径评价函数对识别系统性能进行提升。另一方面，尽管基于显式切分的方法在中文手写字符串识别中表现优异，但是它还有两个不足：第一，系统性能非常依赖过切分算法的表现，这影响了系统识别的上限；第二，识别框架由多个模块组成，各个模块的优化相对独立，很难达到全局最优。而基于隐式切分的方法可以一定程度上避免上述两个不足，而且一般都能进行弱标记训练，这也是第五章提出使用基于递归神经网络识别方法的基本出发点。

第五章　基于递归神经网络的识别方法

5.1　引言

如前文所述，尽管基于显式切分的方法在中文手写字符串识别中表现优异，但是它还有两个不足：第一，系统性能非常依赖过切分算法的表现，这影响了系统识别的上限；第二，识别框架由多个模块组成，各个模块的优化相对独立，很难达到全局最优。此外，显式切分往往需要标记到字符级别的强标记数据，但是现实生活中的字符串数据往往是行级别的弱标记数据，这无形之中增加了标注成本和对于海量数据的利用难度。作为一种隐式切分识别系统，基于递归神经网络的识别系统能很大程度上克服上述的不足，因而理论上有更大的提升识别精度的潜力。本章将主要系统性地介绍该识别框架。

基于递归神经网络（Recurrent Neural Network，RNN）的方法已经在拉丁语系文本识别问题上得到了广泛的应用。但是在中文手写字符串识别领域，使用基于RNN框架的工作并不多。主要原因有两个：其一，中文类别数多且形状复杂，网络训练收敛较为困难；其二，RNN本身的计算复杂度要比CNN大很多，且并行程度较低，再加上中文对于数据量的需求很大，训练耗时极长。如2.2.2章节所述，基于RNN的识别框架一般又可以细分为两种不同的流程，分别是基于滑动窗的方法和基于全局图像特征提取建模的方法。其中，第二种方法可以在更大的感受野范围内更灵活地提取特征，因而可以取得更好的效果。本章将主要讨论使用基于全局图像特征提取建模的方法进行中文手写字符串识别。

此外，手写文本行图像是一个二维序列，各个方向的形变较大，而普通的递归神经网络只能对一维序列建模，这大大限制了模型的建模能力。尽管有文献提出使用二维递归神经网络对图像进行建模，但是这种网络结构的并行程度更低，很难对中文字符串进行大规模训练从而获得较好的识别效果。针对以上问题，本文提出一种基于端到端递归神经网络的识别系统。具体贡献包括以下几点：

1. 提出一种可分离二维递归神经网络模块，这种结构既可以有效利用图像二维信息，又可以进行高效地并行优化。

2. 基于可分离二维递归神经网络模块，提出两种不同的网络结构，并且与传统双向递归神经网络结构进行对比，实验结果说明了二维结构的优越性，其中残差结构可以获得最好的性能。
3. 设计并比较了基于CTC集束搜索以及基于加权有限状态转换器的解码方式的优劣。

本章余下的内容安排如下：在5.2节中对基于递归神经网络的识别框架进行了概述；5.3节详细地介绍本章提出的中文手写字符串识别系统，包含网络结构和解码模块等几个方面；5.4节阐述了串级别合成样本生成算法；最后给出实验结果和本章小结。

5.2 系统概述

本节主要简述基于端到端递归神经网络的识别框架。字符串识别本质上是一个序列识别问题，而序列识别问题中一个核心的问题就是如何更好地利用上下文信息进行建模。对于基于全局图像特征提取建模的方法来说（如图2.4(b)所示），网络的输入是一整张文本行图像，所以需要设计合适的网络模块以便能提取更具有判别能力的上下文特征。

事实上，卷积神经网络已经具有较强的捕获上下文信息的能力，随着层数的加深，CNN的感受野（Receptive Field）逐渐增大，所以如果能设计好合适的卷积核大小和足够的网络层数，CNN同样可以进行长距离上下文信息的建模。文献 [128]使用单纯的深度卷积网络在语音识别任务上取得了成功。尽管卷积神经网络被证明可以对序列进行建模，但是一般需要设计特定的卷积核和足够的深度才能在合适的上下文范围内提取有效信息，这影响了该模型在序列模式识别中的进一步应用。因而对于序列建模，一般更常见的是使用递归神经网络。RNN的核心思路是使用上一时刻的隐层信息作为下一时刻输入的一部分，通过这种递归连接，理论上只使用一层RNN便可以对任意长度的上下文进行建模。基本的RNN模型其实已经在 3.3.2章节中有所介绍。这里，本文再进一步对其进行介绍。

在传统的人工神经网络模型中，网络由若干人工神经元结点互联而成，从输入层到隐含层再到输出层，层与层之间都是前馈连接，每层之间的节点是没有连接的。这种普通的神经网络只适合于处理静态数据分析，如回归、分类等任务，对于很多序列问题却无能无力。而RNN比较适合进行动态数据的分析，

它之所以称为递归神经网络，是因为一个序列当前的输出与前面的输出也有关联。具体的表现形式为网络会对前面的信息进行记忆并应用于当前输出的计算中，即隐藏层之间的节点不再是无连接，而是有连接的，并且隐藏层的输入不仅包括输入层的输出还包括上一时刻隐藏层的输出。

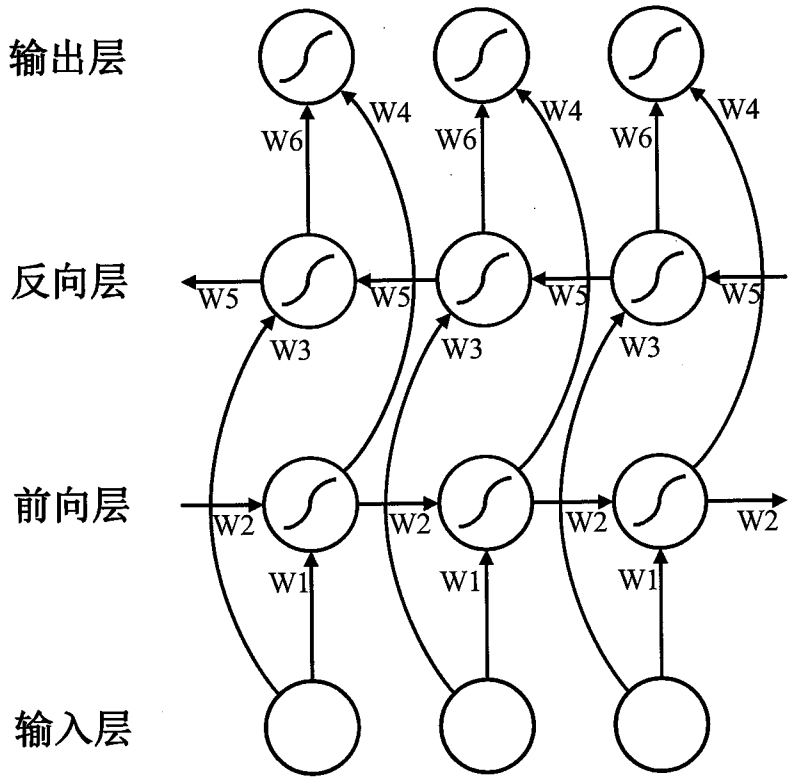


图 5.1: BRNN基本结构图。

在传统RNN的基础上，Schuster等人 [129]提出了双向递归神经网络（Bi-directional RNN，BRNN）。BRNN的基本结构如图 5.1所示，每一个网络分别由前向和后向RNN组成，这两个子网络层使用加和或并联的方式连接同一个输出层。这种结构使输出层能够获得输入序列中每一个点的完整的过去和未来的上下文信息。递归神经网络的训练与传统神经网络的训练相似，同样使用反向传播算法。因为RNN在所有时刻的参数是共享的，但是每个输出的梯度不仅依赖当前时刻的计算，还依赖之前时刻的计算，需要用BPTT算法进行梯度更新。然而利用BPTT算法训练普通RNN时很容易发生梯度消失问题。为了解决这个问题，有学者提出使用门控制的方式改进RNN。在 5.2.1中将会进行详细地介

绍。

在使用神经网络对图像进行特征提取之后，文本行图像转化成为了一组不定长的一维特征序列，这时候利用连接时序分类（Connectionist Temporal Classification，CTC）准则便可以进行文本真值的对齐进而进行网络的训练。CTC可以在不对文本行图像进行预切分强制对齐的情况下，使用前向后向算法计算一个输出序列可能对应的多条路径的概率之和，进而可以最大化真实序列的输出概率。在5.2.2中将会进行详细地介绍。

5.2.1 长短时记忆单元

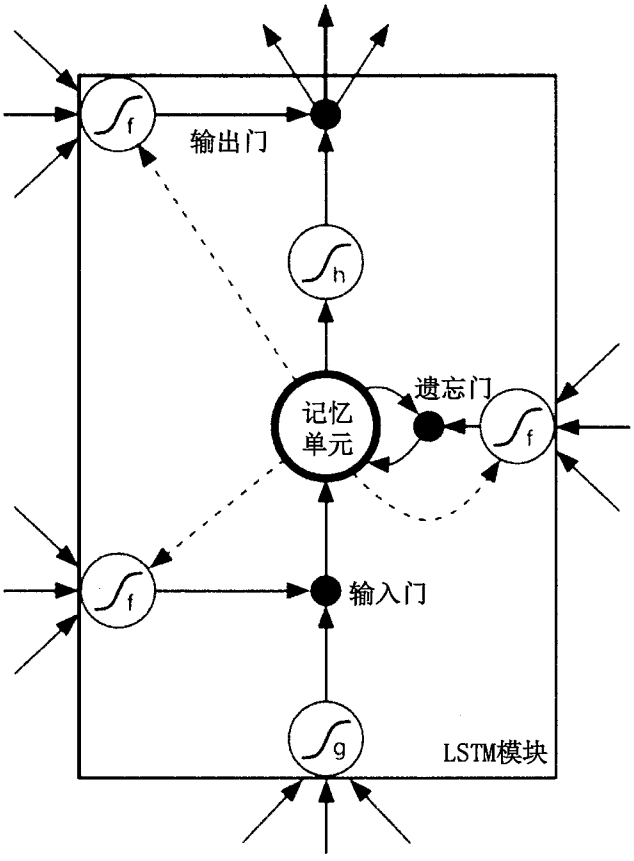


图 5.2: LSTM基本结构图。

为了解决递归神经网络在训练过程中的梯度消失问题，需要在传统RNN结构基础上进行改进。1997 年由Hochreiter和Schmidhuber [130]提出了一种称作长

短时间记忆单元（Long Short-Term Memory，LSTM）的特殊递归神经网络结构。该网络结构能够通过几个特殊的门结构有效避免梯度消失问题。

LSTM 网络由一个一个的单元模块组成，每个单元模块一般包含一个或者多个反馈连接的神经元及三个门结构，分别为输入门、输出门和遗忘门。我们可以用这些门结构控制数据是否输入、输出及遗忘。常用取值为0或1的输入门，输出门和遗忘门与对应数据相乘实现如图 5.2所示。输入门控制是否允许输入信息输入到当前网络的隐层节点中，如果门取值为1则输入门打开允许输入数据，相反，若门取值为0，则输入门关闭不允许数据输入；输出门控制经过当前节点的数据是否传递给下一个节点，如果门取值为1，则当前节点的数据会传递给下一个节点，相反若门取值为0，则不传递该节点信息；遗忘门控制神经元是否摒弃当前节点所保存的历史时刻信息，若门取值为1则保留以往的历史信息，相反若门取值为0，则清除当前节点所保存的历史信息。当然，实际参与计算的时候这些门的取值都是连续值。

与BRNN对应的，也可以对一维序列问题设计双向LSTM结构（BLSTM）。但是，在图像数据中，一维序列很难对上下文进行完整的建模，如果能设计二维结构便可以更好地捕捉上下文信息。这一点将在 5.3.1.2进行具体介绍。

5.2.2　连接时序分类

RNN模型广泛用于各类序列学习问题中，然而该类模型要求预先分割训练数据，通过后处理将模型输出转换为标签序列，因此它的应用受到较大的限制。为了解决这一问题，Graves 等人 [110]提出了连接时序分类准则（Connectionist Temporal Classication，CTC），可以在不对文本行图像进行预切分强制对齐的情况下直接进行概率估计。对于序列学习任务，上下文信息非常丰富，冗余信息量很大，每个时刻未必都具有实际意义。对于无意义的时刻，CTC准则引入空白（blank）标签作为其类别。在CTC序列中，通过首先删除重复标签再删除空白标签的方式可以得到最终的结果。例如与两个序列（a,-,a,b,-）和（-,a,a,-,-,a,b,b）（-表示空白标签），都对应识别结果aab。下面简单介绍CTC的算法细节。

CTC准则的优化目标是最大化一个输出序列对应的所有可能路径的概率之和。在CTC中，一般在网络的Softmax输出层包含所有的字符类别再加上一个空白标签，我们用$P(k|t)$表示每一个时刻t的输出层概率分布，CTC路径π就是

由字符标签和空白标签组成的长度为T的序列。这样，概率$P(\pi|X)$就可以表示为每一个时间步发射概率连续相乘的形式：

$$P(\pi|X)=\prod_{t=1}^{T}P(\pi_t|t,X). \tag{5.1}$$

由于一个识别结果可以对应许多可能的路径，所以定义CTC映射函数B采用上述的先去除重复再去除空白的操作将路径映射为识别结果。这样，识别结果y的条件概率可以通过将所有可能的路径进行加和而得到：

$$P(y|X)=\sum_{\pi\in B^{-1}(y)}P(\pi|X). \tag{5.2}$$

为了避免对上式进行直接加和，一般的做法是使用前向后向算法来对目标序列对应的所有的路径进行递归求和。

5.3 中文手写字符串识别方法

本节将详细介绍端到端中文手写字符串识别系统。本文提出采用多层堆叠的网络结构进行底层特征提取，考虑到中文类别数多且字形复杂的问题，本文使用二维LSTM网络结构对字符图像进行建模。在解码部分，本文提出两种解码算法，都可以结合统计语言模型从而进一步提升识别精度。

5.3.1 网络结构

本文提出的结构都是类似于卷积网络的层次结构。网络的输入是整个文本行图像，底层的网络提取的是图像局部特征，随着层数的增加，高层的网络将提取到更具有判别能力的全局特征。本节将主要介绍一种二维LSTM网络结构，为了增加二维LSTM的并行程度，还提出一种可分离二维递归网络模块作为二维网络结构的基本模块。此外，还提出一种基于双向LSTM的网络结构，以便进行后续的对比。

5.3.1.1 基于二维LSTM的网络结构

本文构建了一个七层的层次结构进行文本行图像的建模，该结构包含卷积层和可分离二维递归网络层（Separable Multi-Dimensional LSTM-RNN，SMDLSTM），如图 5.3，具体的网络配置可以见表 5.1。

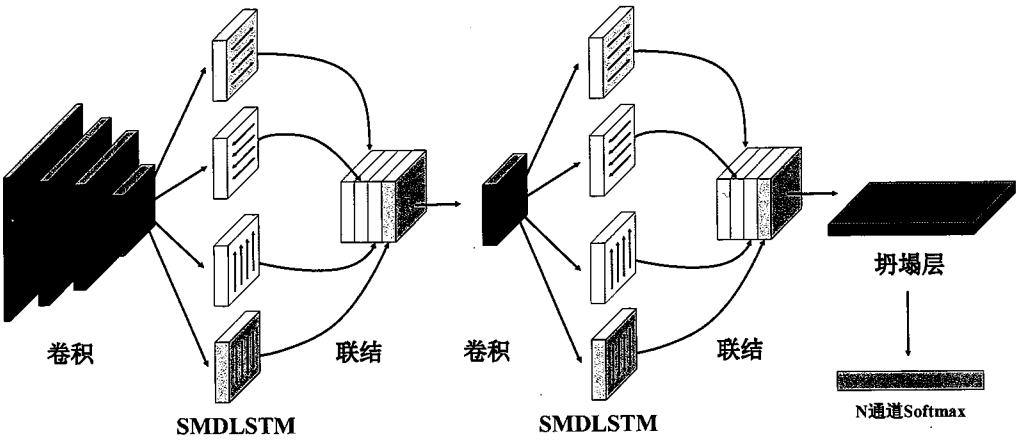


图 5.3: 二维LSTM网络结构系统框图。其中SMDLSTM表示可分离二维递归网络层。

首先，本文使用一系列的卷积层自动提取底层文本行图像特征，卷积层的滤波器均采用3 × 3的小卷积核，并且所有卷积的步长都设置为1。卷积特征平面数从64缓慢递增到256，而另一方面，空间池化层对图像进行下采样，使其尺寸迅速变小，这样可以使网络变得更深。然后使用可分离二维递归网络层提取不同方向上的局部上下文信息。在第一个可分离二维递归网络层（SMDLSTM）后，我们再使用一个卷积层生成局部上下文信息的高层表示，接着再使用最大池化层对输入特征图进行降采样。在最后一个SMDLSTM层之后，坍塌层（Collapse）用来将输入特征平面的每一列上的元素相加，这样就生成了一组一维向量。最后Softmax归一化层用来对包括“blank”类在内的所有类别进行分类。在上述的网络结构中，所有的卷积层都是用截断的ReLU激活函数：

$$\sigma(x) = \min\{\max\{x, 0\}, 20\}. \tag{5.3}$$

这种方式可以使得训练更加稳定。在某些不同的网络层之间，本文还加入了Batch Normalization层 [131]，使得训练能更快收敛，同时防止过拟合。通过上面的结构，一方面可以很好地应对中文类别数大且字形复杂的情况，另一方面通过引入SMDLSTM可以增加二维LSTM的并行程度，使得训练速度得到极大提升。下面将重点介绍可分离二维递归网络层的技术细节。

表 5.1: 二维LSTM网络结构配置。第一行对应网络的底层，maps、k、s和p分别表示特征平面数、卷积核（Kernel）大小、步长（Stride）以及补齐（Padding）大小，Window表示池化层（Pooling）的窗口大小，class表示类别数。下同。

网络层类型	配置
输入层	128 (高度) $\times W$ 灰度图
卷积层	#maps: 64, k: 3×3, s:1×1, p:1×1
最大池化层	Window: 2×2, s: 2×2
卷积层	#maps: 128, k: 3×3, s:1×1, p:1×1
BatchNormalization	-
最大池化层	Window: 2×2, s: 2×2
卷积层	#maps: 256, k: 3×3, s:1×1, p:1×1
最大池化层	Window: 2×2, s: 2×2
卷积层	#maps: 256, k: 3×3, s:1×1, p:1×1
BatchNormalization	-
最大池化层	Window: 2×2, s: 1×2
SMDLSTM层	#maps: 1024
BatchNormalization	-
卷积层	#maps: 512, k: 3×3, s:1×1, p:1×1
BatchNormalization	-
最大池化层	Window: 2×2, s: 2×1
SMDLSTM层	#maps: 1024
BatchNormalization	-
坍塌层	-
Softmax层	#class

5.3.1.2 可分离多维递归网络层

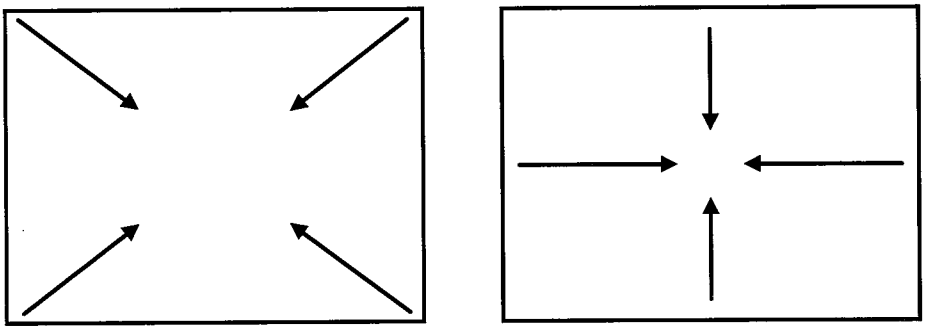
如 5.2.1章节所述，传统的双向递归神经网络只能处理一维序列问题。然而，对于脱机手写字符串识别来说，如果简单地将文本行图像展开成一维序列（按行或者按列），那么整个系统将无法很好地对不同方向的形变进行建模。因此，Graves等人提出使用多维递归神经网络（Multi-Dimensional Recurrent Neural Network，MDRNN）对多维数据进行建模，这种类型的网络可以沿着多个不同的时空维度进行递归连接。本文以二维数据为例[1]。如果用$h(u,v)$表示MDRNN层中位置(u,v)的状态，那么沿着不同维度的前一时刻的状态可以分别表示为$h(u-1,v)$和$h(u,v-1)$，这样便可以得到MDRNN层前馈的一个简化

[1]本文主要讨论二维灰度图像数据，不严格区分多维递归网络层和二维递归网络层。

算式:

$$h(u,v) = f(Wx(u,v) + Uh(u-1,v) + Vh(u,v-1) + b), \tag{5.4}$$

其中，$x(u,v)$表示当前输入，W、U和V表示网络权重，b是一个偏置向量，$f(\cdot)$表示一个广义的非线性激活函数（既可以表示一个传统的激活函数，也可以表示为类似于LSTM一样的运算单元）。尽管这样单独的一层已经可以沿着扫描方向进行完整的上下文信息的建模，但是更好的做法是使用这样的四个MDRNN层并行联合建模，这四个并行层分别处理左上到右下、右下到左上、右上到左下以及左下到右上这四个可能的方向，如图 5.4(a)所示。最后，将四个方向的MDRNN层联合起来，这样便可以对整个上下文进行完整的建模。



(a) 标准　　(b) 可分离

图 5.4: 两种多维递归神经网络的扫描方式。

尽管上述的MDRNN层得到了成功的应用，但是其并行程度很低，很难应对大规模中文手写字符串识别的挑战，因而应用范围有限。在这里，本文同样认为RNN需要沿着每个输入数据的维度来捕获上下文信息，并对原始MDRNN层进行了改进，提出一种可分离多维递归网络层（Separable MDLSTM，SMDLSTM），可以兼顾效果和效率。事实上，类似的结构已经在图像检测和分类领域 [132, 133]、英文字母识别领域 [134]得到了成功应用，但是迄今为止还没有工作证明其在中文手写字符串识别中的效果。

对于SMDLSTM层，LSTM单元对输入序列的每一个方向都进行扫描，即将输入文本行图片按照从左往右、从右往左、从上往下和从下往上的顺序分别进行扫描，如图 5.4(b)所示。通过这种方式，多维递归网络层实际上退化为四

个传统的LSTM层：

$$h(u,v)=\begin{cases} f(Wx(u,v)+Uh(u-1,v)+b), & \text{纵向} \\ f(Wx(u,v)+Uh(u,v-1)+b), & \text{横向} \end{cases} \tag{5.5}$$

如上式，每一个RNN都只依赖沿着横向或纵向的前一时刻的信息，这样所有的四个RNN都能并行计算。在SMDLSTM中，图像中所有的行或列之间也是相互独立的，可以同时计算，这样进一步加大了并行程度。通过这样的方式，SMDLSTM可以更有效地利用计算资源，进而减少计算复杂度。在本文中，对于每一个方向的RNN的输出，通过联接的方式结合起来形成特征的高层表达。与传统MDLSTM不同的是，在单一SMDLSTM层中，我们不直接对不同行或者列之间的信息进行建模，但很显然的是，如果叠加多层SMDLSTM或者甚至是卷积层，整个网络便可以沿对角线方向得到更全局的信息。在网络中，SMDLSTM可以按照图 5.3方式融入层次结构中。实际上在这种方式下，RNN和CNN之间的界限已经基本消除，在多数情况下可以根据需求进行相互替换。此外，SMDLSTM层还可以很方便地推广到诸如视频、遥感图像等更高维度的数据上。

5.3.1.3 基于双向LSTM的网络结构

除了上面提到的二维LSTM网络结构，本文还提出一种基于双向LSTM的识别网络结构，参数配置见表 5.2。在这里，网络结构主要由两部分组成，分别是卷积特征提取层和双向LSTM层。在网络层的开始，使用一系列的卷积层生成特征序列。紧接着，叠加三个BLSTM层对特征序列中的每一帧的概率分布进行建模。其中，BLSTM层的两个方向的信息采用相加的方式进行融合。尽管这种类似的网络结构已经在中文联机手写字符串 [53]以及场景文字字符串识别 [112]中得到成功的应用，但是对于脱机中文手写字符串来说，由于多方向的形变十分严重，仅仅使用双向LSTM难以在二维空间内对文本行图像进行很好的建模。本文将在实验部分对以上两种结构进行深入比较。

5.3.2 解码

解码指的是对于给定输入序列X需要找到最可能的识别结果y。解码一个CTC网络相比过切分的网格的最大不同是还需要考虑连续重复字符和

表 5.2: 基于双向LSTM的网络结构。hidden units表示BLSTM层节点数。

网络层类型	配置
输入层	128（高度）×W 灰度图
卷积层	#maps: 64, k: 3 × 3, s:1 × 1, p:1 × 1
最大池化层	Window: 2 × 2, s: 2 × 2
卷积层	#maps: 128, k: 3 × 3, s:1 × 1, p:1 × 1
BatchNormalization	-
最大池化层	Window: 2 × 2, s: 2 × 2
卷积层	#maps: 256, k: 3 × 3, s:1 × 1, p:1 × 1
最大池化层	Window: 2 × 2, s: 2 × 2
卷积层	#maps: 256, k: 3 × 3, s:1 × 1, p:1 × 1
BatchNormalization	-
最大池化层	Window: 2 × 2, s: 2 × 2
卷积层	#maps: 512, k: 1 × 1, s:1 × 1, p:0 × 0
卷积层	#maps: 512, k: 4 × 1, s:4 × 1, p:0 × 0
BatchNormalization	-
卷积层	#maps: 1024, k: 2 × 1, s:2 × 1, p:0 × 0
BLSTM	#hidden units: 512
BatchNormalization	-
BLSTM	#hidden units: 512
BatchNormalization	-
BLSTM	#hidden units: 512
BatchNormalization	-
Softmax层	#class

“blank”类的影响。另一方面，在识别系统中引入语言模型可以使得系统识别性能有大幅提升，所以我们需要设计特定的解码算法方便统计语言模型的融入。本节将主要给出两种解码算法，分别是精简的CTC集束搜索和基于加权有限状态自动机的解码算法。

5.3.2.1 精简的CTC集束搜索

CTC集束搜索最初由文献 [135]提出，在语音识别中得到成功应用。本文用$Pr^{-}(y,t)$、$Pr^{+}(y,t)$和$Pr(y,t)$分别表示t时刻部分识别转录结果y的“blank”类、非“blank”类以及总路径的概率，那么t时刻类别k的扩展概率$Pr(k,y,t)$可

以表示为：

$$Pr(k,y,t) = Pr(k,t|X)P^{\alpha}(k|y)\begin{cases} P^{-}(y,t-1), & \text{如果} \quad y^{e}=k \\ P(y,t-1), & \text{其他} \end{cases} \tag{5.6}$$

其中$Pr(k,t|X)$表示t时刻类别k的CTC层发射概率，y^{e}是y序列的最后一个字符，$P(k|y)$是从y到$y+k$的语言转移概率（也就是语言模型），可以用参数α对其进行加权。Algorithm 1描述了具体的搜索算法。与原始的CTC集束搜索相比，本文的搜索算法有两方面的不同。首先，我们对扩展概率的表达式引入了超参数α，以便对语言模型进行加权；其次，为了进一步减小搜索空间我们在t时刻对发射概率进行了进一步的双策略剪枝，首先只保留概率最高的CN个候选类别，其次这些候选类别的概率还需要大于平均概率。除此之外，CTC集束搜索不仅可以融入传统的回退语言模型，还可以融入第三章中介绍的神经网络语言模型，神经网络语言模型可以将词从离散空间投影到连续空间进行隐式的平滑，以便对更高阶的上下文进行建模从而帮助进一步提升系统识别精度。在实际系统中我们一般取集束搜索宽度为300，因为候选字符采用的是双策略剪枝，所以CN可以设置为总类别数的三分之一。

5.3.2.2 基于加权有限状态转换器的解码

基于加权有限状态转换器（Weighted Finite-State Transducers，WFST）的解码方法在语音识别领域已经得到广泛使用。本节将介绍其在中文手写字符串识别中的使用方法。WFST 是以半环代数（Semiring）作为数学基础，是有限状态接收器（Finite-State Acceptor，FSA）的一种推广，每一个状态转换都有一个输入符号、输出符号和一个权重。在WFST中的一条路径对应一个输入符号序列和一个输出符号序列。有关WFST的更多知识可以参见 [136]，这里我们仅就如何使用WFST进行基于CTC的解码做详细的介绍。

本文的解码算法主要基于Eesen工具包 [137]实现，我们将CTC符号序列、词典以及语言模型分别表示成令牌（Token）WFST、词典WFST以及语法WFST，下面将分别对它们进行介绍。

令牌WFST（表示为T）将帧级别的CTC符号序列映射成为词典单元序列（本文指的是中文字符）。对于一个词典单元，T可以在帧级别上包含所有可能的符号序列。因此，令牌WFST允许“blank”类的出现，也允许任何非

Algorithm 1 精简的CTC集束搜索

Initialize: $B \leftarrow \{\phi\}; Pr^{-}(\phi, 0) = 1$
for $i = 1 \rightarrow T$ **do**
  $\hat{B} \leftarrow$ **the N-best sequences in** B
  $B \leftarrow \{\}$
  for $y \in \hat{B}$ **do**
    if $y \neq \phi$ **then**
      $Pr^{+}(y,t) \leftarrow Pr^{+}(y,t-1)Pr(y^{e},t|X)$
      if $\hat{y} \in \hat{B}$ **then**
        $Pr^{+}(y,t) \leftarrow Pr^{+}(y,t) + Pr(y^{e},\hat{y},t)$
      end if
    end if
    $Pr^{-}(y,t) \leftarrow Pr^{-}(y,t-1) + Pr(-,t|X)$
    add y **to** B
    sort emission probabilities at time t **and retain the top** CN **classes with each** $P > \frac{1}{\#output\_nodes}$
    for $k = 1 \rightarrow CN$ **do**
      $Pr^{-}(y+k,t) \leftarrow 0$
      $Pr^{+}(y+k,t) \leftarrow Pr(k,y,t)$
      add $y+k$ **to** B
    end for
  end for
end for
Return: $\max_{y \in B} Pr^{\frac{1}{\|y\|}}(y,T)$

“blank”类的连续重复出现。图 5.5给出了一个令牌WFST的例子，其中所有的路径都会被映射成一个词典单元“天”。语法WFST（表示为G）用于编码被允许出现的字符序列。组成它的符号是字或词，其边上的权重代表语言模型的概率。一个简单的语言模型的例子如图 5.6所示，该WFST主要对“天安门”和“天通苑”进行建模。词典WFST（表示为T）主要用于将词典单元映射为词语。图 5.7给出了一个简单的示例。因为本文没有使用词语言模型，所以词典WFST并没有什么作用。

在获得了上述三个WFST以后，我们要将它们组合成一个搜索图。为了尽可能减小搜索空间以便提升解码速度，最终的WFST可以通过下面的操作得到：

$$TLG = T \circ min(det(L \circ G)), \tag{5.7}$$

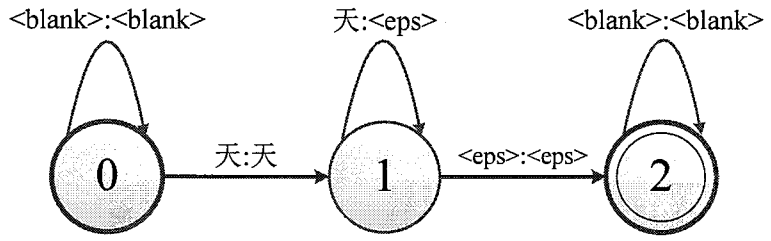


图 5.5: 表示字符“天”的令牌WFST示例。⟨blank⟩表示blank类，⟨eps⟩表示空输入或者空输出。下同。

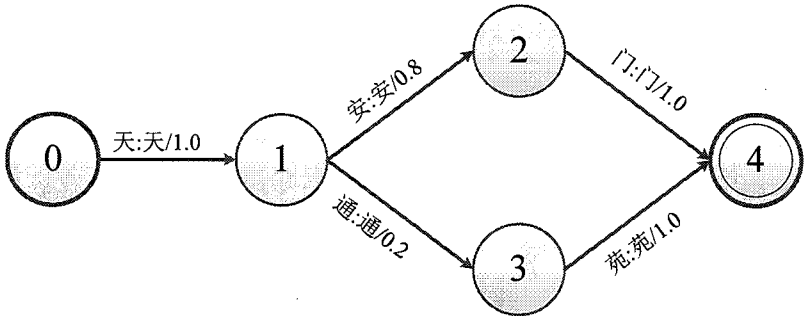


图 5.6: 一个简单的语法WFST示例。图上的数字表示给定历史字符时语言模型的转移概率。

其中$\circ$、det和min分别表示组合、确定化以及最小化操作。最终的搜索图TLG以字符网络的输出（即CTC符号序列）作为输入，通过集束搜索得到识别字符序列。

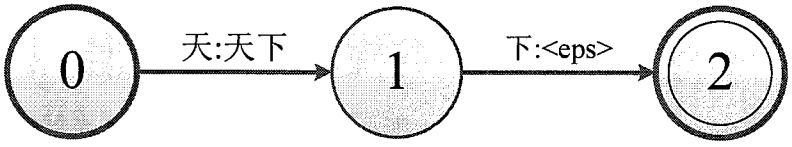


图 5.7: 表示词条“天下”的词典WFST。

使用WFST进行解码时，一般需要从HMM模型的角度进行建模。CTC层的输出值理解为后验概率$p(y_k|x_t)$，表示给定输入特征x_t时字符y_k的概率分布，然而在HMM模型中观测概率一般建模为：

$$p(x_t|y_k) = \frac{p(y_k|x_t)p(x_t)}{p(y_k)}, \tag{5.8}$$

其中$p(y_k)$是每一个类别出现的先验概率，$p(x_t)$是一个可以被忽略的独立变量，这样上式等价为：

$$p(x_t|y_k) = \frac{p(y_k|x_t)}{p(y_k)^{\alpha}}, \tag{5.9}$$

其中，α是一个超参数，一般在验证集上优化得到。在实际的实验中，本文发现对于除“blank”之外的类别取等先验可以得到最好的效果，而“blank”类的概率一般取0.25~0.5 之间的值，在实际实验中设置为1/3。此外，我们取α为0.5。

5.4　串级别合成样本生成算法

本章的目标是使用递归神经网络结构进行大规模手写中文字符串系统的搭建，但是现有的文本行数据十分有限，很难对模型进行充分的训练。如果我们简单地补充单字样本进行训练则会面临两个问题：首先，单字样本所包含的上下文信息有限，不足以“教会”网络如何对上下文进行正确地建模；其次，过多的单字样本会增加系统数据交换的负担，使得训练速度变慢。因此，本节将详述串级别合成样本生成算法，这种串级别合成样本生成算法的优点在于可以结合不同写者的书写风格，按照字符位置的模态分布来生成串级别的合成样本，减小了样本合成过程中带来的训练集与测试集的分布差异。

5.4.1　串级别合成样本基本生成算法

深度学习算法在提升OCR系统表现的同时，也带来了训练样本需求量巨大和现有数据集样本数不足的问题。一种做法是使用巨大的人力物力来收集和制作所需要的数据，比如斯坦福模拟人类识别系统建立的ImageNet图像数据库[138]，其层次结构的每个节点都由数百和数千个图像描绘，而且每个节点平均拥有超过500个图像。但是这种做法成本消耗过大，普适性较差。另一种做法则是结合有限的资源根据具体问题合成实验所需要的的数据集。在OCR领域，字符串识别的对象包括印刷体样本和手写体样本。印刷体样本书写工整，字符间距固定，合成难度较小。可以根据确定的字体选择所需的文本再按照简单的排列规则进行合成。对于手写体的样本，如果我们已有不同书写者的单字数据集，在合成串级别的合成样本时，基本的做法可以类似于印刷体样本。首先选择不同书写者，再结合语料库选择该书写者对应的单字图像，在单字之间加入随机的字符间距，按照自左向右，自上到下的书写方向进行串级别的文本合

成。这种基本的生成算法生成的数据集虽然包含了相应的手写字符以及上下文信息，但是由于不同书写者书写时字符的间隔分布不均，以及单字在字符串不同位置的大小不一，会造成合成数据集分布与测试集分布之间存在差异，从而导致实验的识别精度受到影响。所以，我们有必要对不同书写风格的分布差异在合成时进行一些处理。

5.4.2 结合书写人模态分布的样本合成算法

在改进的串级别合成样本生成算法中，我们将书写风格建模为符合高斯分布的水平和竖直方向上的位移。在本文中，我们已有不同书写者书写的少量的文本行文件（在CASIA-HWDB数据集中，我们可以按书写人找到其书写的单字和文本行样本），我们对这些文本文件进行了统计模型的分析，计算这些文本的字符位置分布的统计量，进而对不同的书写风格的模态分布进行建模。由于文本图像并不是单纯的一维的结构，因此我们计算了不同书写者书写文本的字符中心在竖直方向分布的均值$Mean_y$和标准差Std_y，以及文本在水平方向的字符间距模型的均值$Mean_{gap}$和标准差Std_{gap}。对于某一位具体的书写者，现有其书写的文本数据$\mathbf{X}$，文本中的单个字符$\mathbf{x}_i \in \mathbf{X}$，$i \in |\mathbf{X}|$，其中$|\mathbf{X}|$表示字符数目，设文本行图像中心的纵坐标为$MidLine$，则有单个字符$\mathbf{x}_i$中心位置在文本行图像中的竖直像素坐标：

$$midy_i = 0.5 * (top_i + bottom_i). \tag{5.10}$$

其中top_i和$bottom_i$分别为候选字符的顶部和底部坐标，我们可以进一步得到字符中心偏离文本行竖直中心位置的像素均值：

$$Mean_y = \frac{1}{|\mathbf{X}|} \sum_i (midy_i - MidLine). \tag{5.11}$$

字符中心偏离文本行竖直中心位置的像素的标准差：

$$Std_y = \sqrt{\frac{1}{|\mathbf{X}|} \sum_i (wid_i - Mean_y)^2}. \tag{5.12}$$

值得注意的是，在文本行上部和下部的标点符号不参与竖直方向统计量的计算。类似的，我们也可以得到水平方向的字符间距模型的均值$Mean_{gap}$和标准

差Std_{gap}。在得到了真实样本字符在竖直和水平方向的统计量之后，结合基本的串级别合成样本算法，再采用高斯分布$\boldsymbol{N}(Mean, Std^2)$来拟合该书写者书写风格的字符位置的分布，就得到了本文所使用的串级别合成样本生成算法。图5.8给出了两个合成实例，其中第一行是合成样本，第二行是书写人样本。可以看到，我们的合成算法明显捕捉到了书写风格信息，合成图片和原始书写图片几乎难以分辨，图5.8(a)书写得稀疏一些，图5.8(b)相对紧密一些，这些在合成样本中都有所反应，甚至还自然地模拟出了粘连的情况。

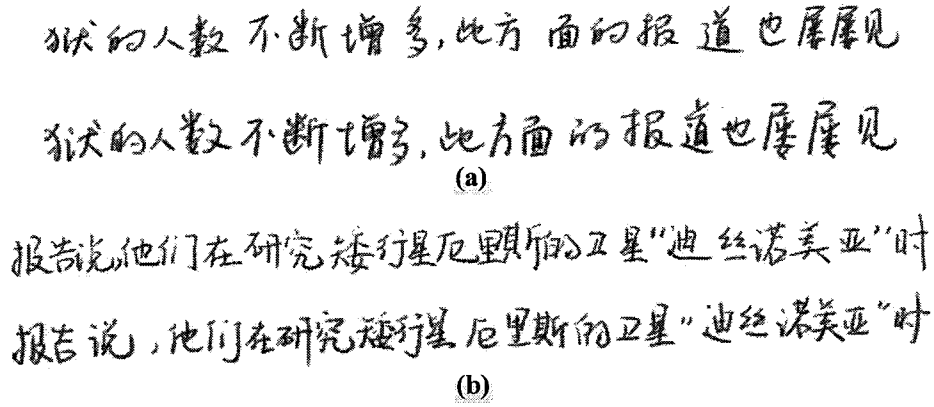


图5.8: 合成样本示例。对于每一组样本，第一行表示合成图像，第二行表示真实书写人书写图像。

5.5 实验

为了验证本章提出的基于递归神经网络识别系统的效果，文本将主要在ICDAR-2013竞赛数据集上进行一系列的实验。整个中文手写字符串识别系统是Torch 7 [139] 平台上实现的，我们使用cuDNN v5对网络实现进一步加速。以下所有的实验都是在Intel(R) Xeon(R) E5-2680v3 2.50GHz CPU的工作站上进行的。该工作站配备256GB内存，以及四块NVIDIA Titan X图形处理器。本节首先对实验数据和实现细节进行简介，包含训练数据和测试数据的介绍以及网络训练的细节；其次，我们将给出基础的实验结果；接着，将对两种解码算法进行详细的比较；最后将初步探讨两种进一步提升精度的技巧。

5.5.1 数据及实现细节

本文使用CASIA-HWDB数据集作为训练集，包含其中的文本行数据部分（HWDB 2.0-2.2）和单字部分（HWDB 1.0-1.2）。这些数据对于训练大规模手

写中文字符串识别系统来说还并不充足，所以我们通过三种方式对训练集进行扩展：（1）对文本行图像进行尺度变换、随机裁剪以及旋转等数据增强操作；（2）将来自单字数据集的元素和从文本行数据集中提取的字符进行随机打乱，然后按照5.4.2介绍的方式合成新的文本行；（3）仍旧采用孤立字符合成文本行，但是文本内容取自3.5.1章节使用的的训练语料库。为了研究不同类别对于识别系统性能的影响，我们使用用两个具有不同类别数的数据集来训练我们的模型。其中一个类别与测试集类别相同，为2672类，简称为Train-2672；另一个具有完整类别数（7356类），简称为Train-7356。之所以要合成乱序的样本是因为基于LSTM的字符模型在训练过程中会记住大量语义信息，为了使得模型能更集中于图像层面的建模，所以我们需要一定程度上破坏这种机制。最终在合成样本中，有语义信息的样本数量与乱序样本数量基本相当。

本节我们使用ICDAR-2013竞赛数据集评估手写中文识别系统的性能。数据集详细介绍可以参见章节3.5.1。需要说明的是，这里使用的测试集比ICDAR-2013略小，因为我们去除了不包含在训练集中的字符。其中，我们将ICDAR-2013中不包含在Train-2672中的字符去除掉后的测试集称为Test-2672，将不包含在Train-7356中的字符去除掉后的测试集称为Test-7356。测试集的主要信息见表5.3。从表中可以看到，本节中使用的测试集不管是单字集还是文本行的数量上都非常接近标准的测试集，因此可以认为将我们的结果和此前的工作进行对比是公平的。

表5.3: 数据集概述。

数据集类型	#行数	#字符数
Train-2672	297106	5950411
Train-7356	1069678	20268321
Test-2672	3432	89750
Test-7356	3432	91473
标准测试集	3432	91563

下面介绍一下实现细节。对于所有的文本行图像，如果其高度小于128像素，我们将其填补到128像素；如果高度大于128像素的话，我们将其保持宽高比缩放到128像素。我们对系统进行了精心设计，以使其能够在多个图形处理器上同步训练。此外，我们还是用近似的变长训练策略，即同一批次训练图像的宽度填补到相同值（一般取该批次下的最大值），但是在不同批次之间可以具有不同的宽度。

我们的网络使用RMSProp算法进行优化 [140]，基础学习率设置为$5e-4$，最小批样本数为8。对于每轮训练，我们随机选取1/10的样本用于训练。在训练过程中，为了使得网络更好地收敛，在第一轮训练中我们采取课程学习的方法（Curriculum Learning） [141]进行训练，即按照样本长度递增的方式进行初始训练，然后对于剩余的训练轮数，我们转换到随机顺序的方式进行训练。对于Train-2672和Train-7356数据集，训练一轮的时间分别为12分钟和45分钟，通常经过160轮之后即可终止训练。

5.5.2　基础实验结果

本节我们将评测基于SMDLSTM的模型和基于BLSTM的模型，这两个模型分别使用2672类和7356类的数据集进行训练，我们称相应的模型为SMDLSTM-2672，SMDLSTM-7256，BLSTM-2672和BLSTM-7356。和第三章第四章一样，这里同样使用字符级别的正确率CR和准确率AR评价系统的识别率。具体的计算公式可以参见 3.24。需要说明的是，本节我们的重点评测对象是网络模型，所以解码算法统一都使用基于WFST的算法。

5.5.2.1　不同网络结构的影响

表 5.4: 两种模型的识别结果。

模型类型	AR（%）	CR（%）
SMDLSTM-2672	90.02	90.72
BLSTM-2672	86.77	87.16
SMDLSTM-7356	86.64	87.43
BLSTM-7356	82.97	83.37
[98]	83.5	-

不同类别下，两种不同结构的模型的识别结果如表 5.4所示。可以看到，无论是对于2672类还是7356类，基于SMDLSTM的模型都要比基于BLSTM的模型具有更好的性能。同时很显然的，当类别数较少时模型性能可以更好。当类别数增大时，两种不同结构的模型之间的精度差异变得更加明显，这证实了SMDLSTM结构的有效性。与只能捕获一维上下文信息的BLSTM结构相比，基于SMDLSTM的模型可以有效地处理完整的上下文信息，因此可以更好地对文本行图像进行建模。特别地，就CER（字符错误率，等价于1-AR）

而言，SMDLSTM-7356相对BLSTM-7356来说降低21.6%。另外，SMDLSTM-7356的模型大小（18.5M）要比BLSTM-7356的（21.7M）要更小一些。和同样使用超过7000类样本进行训练的基于标准MDLSTM的框架 [98]相比，基于SMDLSTM的模型结果明显比它更好，而且我们并没有使用任何其他语言的样本对模型进行预训练，这极大简化了模型训练的流程。

5.5.2.2 语言模型的作用

我们使用与第三章相同的通用语料库训练语言模型[2]，该语料库包含大约五千万字符。表 5.5 库给出了不同文法数下回退语言模型对于识别精度的影响。因为我们使用WFST 进行解码，所以可以很方便地融入更高阶的语言模型。为了方便叙述，这里我们只汇报AR这一评价指标。

表 5.5: 使用不同文法数语言模型得到的AR（%）。

模型类型	语言模型文法数				
	2	3	4	5	8
SMDLSTM-2672	91.87	92.51	92.59	92.59	**92.61**
BLSTM-2672	89.02	89.99	90.07	90.08	90.08
SMDLSTM-7356	89.40	90.26	90.37	90.37	90.38
BLSTM-7356	85.13	86.38	86.49	86.51	86.51
[98]	88.0	89.3	89.4	89.1	-
[45]	-	89.28	-	-	-
第四章方法	-	-	-	**95.04**	-

和表 5.4相比，很明显，所有结构的识别精度都能通过BLM得到提升。通过对比不同文法数的语言模型，可以发现从二元文法到三元文法语言模型的提升是非常明显的，但是从三元文法语言模型到更高阶语言模型的性能提升十分有限。这与 3.5.2.1章节中，过切分识别系统语言模型的表现是一致的。该现象是由于数据稀疏问题造成的，因为数据稀疏问题对高阶语言模型的影响更加明显，从而抵消了高阶语言模型的优势。使用类似的BLM，SMDLSTM-7356模型的性能要比 [98]的更好，但是两者的精度差距变小了，这可能是由于我们补充的基于语料库的样本造成的。这个现象意味着基于LSTM的模型可能比较容易造成语言上下文的过拟合。结合八元BLM的SMDLSTM-2672模型可以得到92.61%的AR，相对于此前的ICDAR-2013参考基准 [45]而言有了很大提升。

[2]本节不讨论使用章节4.3.2.1介绍的大语料库。因为如该章节所述，ICDAR2013竞赛集的文本真值与该语料库高度重合，会出现严重的过拟合现象。

SMDLSTM-7356 也比之前的最好结果要好，但是两者差距较小。然而，上面的精度和第四章中过切分框架下使用神经网络语言模型和卷积神经网络形状模型得到的精度相比还有一定的差距。这意味着对于基于LSTM 的模型而言仍然存在着很大的提升空间。

5.5.2.3　单字数据集识别结果

上文说到基于LSTM的模型在对图像建模的同时也学习到了很多语言信息，为了进一步研究基于LSTM模型的内在本质，我们在两个不含有语言上下文信息的单字数据集上进行了文本行识别实验。其中一个单字测试集是在文本行集合Test-7356中抽取的1381类，简称为Test-1381，另一个数据集取自ICDAR-2013竞赛的单字识别数据集，包含224419个字符，共有3755类，简称为Test-3755。在识别过程中，每个单字被看作一个文本行，因而可以同样给出AR和CR的评价结果，如表 5.6所示。实际上，Test-1381和Test-7356测试集具有完全相同的字符，但是无论是SMDLSTM-7356还是BLSTM-7356都要比表 5.4中的识别表现更差。另一方面，SMDLSTM-7356和BLSTM-7356在测试集-7356上的结果差距很明显。SMDLSTM-7356的识别率下降到了一个可以接受的范围之内，但是BLSTM-7356的AR急剧下降到了62.63%。上面的实验结果表明SMDLSTM在字符信息建模上更具优势，但同时也表明两个模型都可能会受到语言上下文过拟合问题的影响。

表 5.6: 单字数据集上的识别结果。

模型类型	Test-1381		Test-3755	
	AR（%）	CR（%）	AR（%）	CR（%）
SMDLSTM-7356	84.15	86.19	80.09	81.52
BLSTM-7356	81.97	83.26	62.63	64.04

5.5.2.4　错误分析

图 5.9给出了本章文本行识别系统的几个识别示例，这些图揭示了引起识别错误的几个因素。尽管我们的训练集补充了不少形变样本，但是图片中存在比较扭曲的文本行 5.9(a)或者是字形相当不规范 5.9(b)的时候比较容易出现识别错误。除此之外，标点符号 5.9(c)和英文字母 5.9(d)也比较容易引发识别错误，因为这些字符样本在训练样本中出现次数较少。

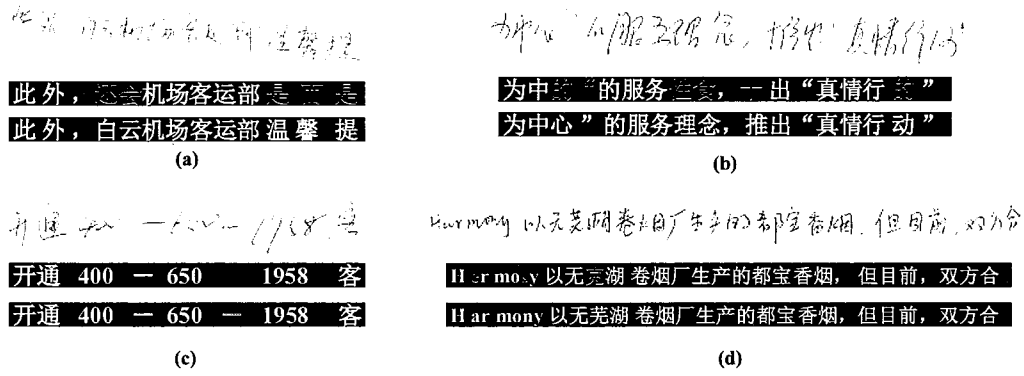


图 5.9: 错误识别样本示例。对于每一个样本，第一行表示文本行图像，第二行表示SMDLSTM-7356模型结合八元BLM的识别结果，第三行表示文本行图像对应的文本真值。

5.5.3 两种解码算法的比较

上面表 5.5的识别结果是基于WFST的解码方式得到的。在章节 5.3.2.1中，本文还介绍了基于CTC集束搜索的算法，下面将对这两种解码算法进行比较。为了叙述方便，这里只使用SMDLSTM-7356作为基础的字符模型，表 5.7给出了分别结合五元文法和混合神经网络语言模型HRMELM（与章节 3.5.2.3相同的模型）时CTC集束搜索的表现。

表 5.7: CTC集束搜索解码结果。

语言模型类型	AR（%）	CR（%）
五元BLM	89.98	90.36
HRMELM	90.35	90.69

对比表 5.7和表 5.5可以发现，在同样使用五元文法的情况下，基于WFST的方式相比基于CTC集束搜索的方式在精度上略高一些。经过我们统计，两种方式在时间消耗上比较类似，平均每个字符串耗时0.3秒。对于BLM而言，基于WFST的方式可以更方便地扩展到高阶文法，而且将来融入词语言模型也会更方便一些。另一方面，CTC 集束搜索可以更方便地融合神经网络语言模型，NNLM也可以在同样条件下获得相对更好的识别精度。但是融合之后解码速度较慢，每个文本行需要10秒的时间进行解码。相对地，由于基于WFST的算法属于静态解码，带有类似缓存机制的递归神经网络语言模型很难编译成为静态图（前馈神经网络语言模型则可以通过提前前馈的方式预先得到大量语言组合

概率）。文献 [142]提出可以通过语言模型采样的方式，获得类似于传统BLM的语言模型形式，这可以作为下一步的工作。

5.5.4 两种提升精度的策略初探

在实验的过程中，我们还发现两种可以继续提升识别精度的策略，这两种策略可以显著提升系统识别性能。下面分别对这两种策略进行介绍。需要说明的是，下面的实验都是在Train-7356数据集上进行的。

5.5.4.1 学习率调整策略

表 5.8: 不同学习率调整策略对SMDLSTM模型结果的影响。

学习率调整策略	AR（%）	CR（%）
RMSprop	86.64	87.43
混合方式	88.67	89.37

学习率是神经网络模型优化过程中的一个极其重要的超参数，在本文中我们采用了学习速率衰减的策略结合RMSprop自适应优化算法来设置学习率，实验证明这种混合的学习率调整策略对我们的模型优化有很大的帮助。通常在采用非自适应的优化算法（比如随机梯度下降法）时，我们会使用学习率衰减这种策略，按照人为预先设置的学习率衰减阶段值或者衰减函数随着迭代次数来设置学习率大小。但是这种做法有着主观的随机性，而且人为学习率设定值之间的跳变会导致目标损失函数值的抖动和原始优化步长的改变，从而给优化过程带来不确定性。为了减少这种随机性和不确定性，本文使用可以自适应调整学习率的RMSprop算法，但是在实验过程中我们发现，当网络第一次收敛时，再将学习率减少为原来的十分之一（即$5e-5$）可以使得网络的识别精度有明显的提升，如表 5.8所示，对于SMDLSTM模型，调整前后CER相对降低了15.19%，这说明这种混合策略对精度提升有明显的效果。

5.5.4.2 残差网络结构

为了进一步提升系统性能，我们考虑设计判别能力更强的网络结构。这里，我们主要借鉴了在2015年ImageNet竞赛中大放异彩的MSRA Kaiming He团队设计的残差网络结构ResNet [143]。本文设计了基于SMDLSTM的bottleneck基本模块，如图 5.10所示。与原始ResNet的基本模块不同的是，我们设计的bottleneck并

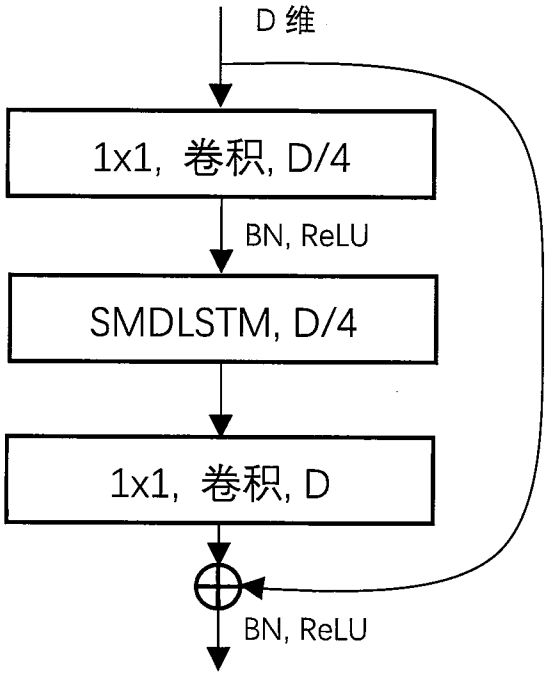


图 5.10: 基于SMDLSTM的bottleneck结构。其中BN表示Batch Normalization层。

不是采用卷积、池化相互堆叠的方式，而是使用卷积层、SMDLSTM层、卷积层的结构来构建。在bottleneck结构中，第一个1×1 kernel大小的卷积层将输入特征平面数降为原来的四分之一，随后经过一个SMDLSTM层进行序列建模，最后再用1×1的kernel恢复原始特征平面数，以便和主路的通道数匹配。这种做法可以通过降低参数个数而大大增加网络层数。网络结构的具体配置如表 5.9所示。网络的开始使用传统的卷积加上池化的方式在主路上将图像尺寸迅速变小，随后加入bottleneck结构，bottleneck结构一共分成三组，每组重复五次，按照特征平面数从256递增为1024，最后在网络的主路上还有一个SMDLSTM层进行全局特征提取。

本文将上述网络称为ResSMDLSTM。该网络的训练方式与SMDLSTM结构相同，在训练过程中，我们同样发现在网络第一次收敛的时候降低学习率为$5e-5$可以将AR从89.20%提升至90.86%，这进一步验证上述学习率混合调整策略的有效性。具体的识别结果如表 5.10所示。和SMDLSTM比较，显然ResSMDLSTM精度有了极大的提升，与神经网络语言模型结合之后，可以最高达到93.20%的AR，在大类别集上又进一步逼近了过切分识别系统所能达

表 5.9: 残差二维LSTM网络结构配置。

网络层类型	配置
输入层	128（高度）×W 灰度图
卷积层	#maps: 16, k: 3 × 3, s:1 × 1, p:1 × 1
最大池化层	Window: 2 × 2, s: 2 × 2
卷积层	#maps: 64, k: 3 × 3, s:1 × 1, p:1 × 1
BatchNormalization	-
最大池化层	Window: 2 × 2, s: 2 × 2
卷积层	#maps: 128, k: 3 × 3, s:1 × 1, p:1 × 1
最大池化层	Window: 2 × 2, s: 2 × 2
bottleneck	{#maps: 256} × 5
bottleneck	{#maps: 512} × 5
bottleneck	{#maps: 1024} × 5
SMDLSTM层	#maps: 1024
BatchNormalization	-
坍塌层	-
Softmax层	#class

到的最高结果（本文第三、四章的工作）。此外，ResSMDLSTM的参数也仅有18.5M，与18M的SMDLSTM几乎一致，这说明增加网络深度可以进一步提升识别精度。将来的工作可以进一步加深ResSMDLSTM以便更系统地观测深度对于识别率的影响。

表 5.10: ResSMDLSTM识别结果。

语言模型类型	AR（%）	CR（%）
-	90.86	91.24
五元BLM	92.59	93.71
HRMELM	93.20	93.37
[45]	89.28	90.22
第四章方法	95.04	95.15

5.6 小结

本章主要介绍使用递归神经网络进行手写中文字符串识别的方法，包含网络设计、解码算法以及训练策略的改良。使用基于递归神经网络的框架进行手写字符串识别可以克服过切分识别框架中对于切分过度依赖以及不能全局优化模型这两个不足，这种方法已经在拉丁语系文本识别问题上得到了广泛的应

用。但是在手写中文字符串识别领域，使用基于RNN框架的工作并不多，主要是因为中文类别数众多且字形复杂导致现有的网络结构很难对其上下文进行完整高效的建模。

针对上述问题，本章首先简述基于递归神经网络的识别框架，对基于全局图像特征提取建模的方法做了基本的介绍。其次，详细介绍了中文手写字符串识别系统，考虑到中文类别数多且字形复杂的问题，本文提出采用多层堆叠的网络结构进行底层特征提取。由于文本行图像属于二维数据，本文提出采用二维LSTM对字符串进行建模，以便能对多方向的形变进行建模。为了使网络能够进行大规模训练，本文提出一种可分离多维递归网络层，在保持对多维数据的建模能力之外，可以大大提高网络并行程度，使得大规模训练成为可能。本文还提出使用两种不同的解码方法。基于WFST的解码方法可以很方便地推广到高阶语言模型，而基于CTC集束搜索的方法可以更灵活地融入神经网络语言模型。接着，为了解决中文大规模训练缺乏足够数据量的问题，本文还提出一种结合书写人模态分布的串级别样本合成方法，可以更真实地合成文本行数据。最后，在实验部分验证了本文算法的有效性，并且还初步提出两种进一步提升识别精度的策略。

尽管本章提出的算法识别精度与过切分算法最高精度相当，但是系统中还是有很多可以改良之处，主要在以下方面：

1. 鉴于LSTM对语言上下文有过拟合倾向，样本合成的策略可以进一步改良。
2. 基于WFST的解码方式可以融入神经网络语言模型以便进一步提升性能，还可以考虑融入词语言模型。
3. 基于残差网络的模型的探索还比较初步，可以设计更深的网络。

总而言之，基于端到端递归神经网络的识别框架还有巨大的潜力有待发掘，这将是文字识别技术发展的一个契机。

第六章　总结与展望

6.1　本文工作总结

人工智能正在对社会的发展和变革产生深刻的影响，光学字符识别作为智能系统领域的一个重要分支和组成模块，有着极高的理论研究和应用价值。其中，中文字符串识别是一个成熟的文档分析系统所不可或缺的一部分，对于文档电子化有着极大的促进作用，可以在国民经济生活、保护国家灿烂文化等方面有积极的影响。

中文手写文本识别主要分为基于显式切分的方法和基于隐式切分的方法。虽然目前显式切分的方法可以取得较高的识别精度，但是其包含的语言模型和形状模型等各个模块还存在不少的缺陷。其中，传统语言模型利用的上下文信息还比较有限，还无法进行高阶建模，而形状模型对于候选字符的建模还并不充分，这些模型的性能有待进一步提升。另一方面，基于隐式切分的端到端递归神经网络识别框架也日益受到重视，该方法可以克服显式切分识别系统过于依赖切分的缺陷，在拉丁语系文本识别中已经占据主导地位，但是在中文识别领域还鲜有成果。鉴于上述两个框架的不足，本文提出了若干个有效的改进策略，主要贡献主要有以下几点：

1. 在文本识别系统中，语言模型对于性能的提升发挥了重大的作用，但是高阶语言模型的建模也面临着数据稀疏和维度灾难等挑战。为了克服这些问题，本文首次将神经网络语言模型引入到基于过切分的中文手写字符串识别系统中。神经网络语言模型首先将词（字）从离散空间投影到一个连续空间中，然后在该空间中对语言模型进行隐式的平滑以及序列概率的预测。本文对神经网络语言模型在过切分识别系统中的作用进行了全面的评价，主要比较了前馈神经网络语言模型、递归神经网络语言模型和混合语言模型。实验表明，这种新型的语言模型确实可以提升系统识别性能，混合递归神经网络语言模型可以得到最好的结果。
2. 在本文中，单字分类器、过切分以及几何模型合称为形状模型，它们主要是从图像形状层面对文本行进行建模，发挥着十分重要的作用。但是这三类模型目前都缺乏基于深度模型的工作，其效果有待进一步提升。

为此，本文提出一种卷积神经网络形状模型，将其融入过切分识别系统之后发现可以大幅提升系统性能。文本搭建了一个15 层CNN 作为字符分类器，在设计的过程中，还考虑融入领域知识，所以除了原始图像之外，还生成了八方向非线性归一化图像作为额外的输入通道。在单字分类器的训练过程中，本文还加入非字符类以便网络能显式地拒绝非字样本。对于过切分，本文提出了一种基于学习的两步过切分方法，将传统的基于前景分析的方法与滑动窗卷积神经网络分类器相结合，使得召回率有了进一步的提升。对于几何模型，本文将几何模型从传统的分类器转换为基于卷积神经网络的模型。为了使送入分类器的候选模式仍旧保留书写风格信息，本文使用多项式回归首先得到文本行的大致走向，然后动态调整候选模式图像的上下空白。在中文手写文本行识别实验中，基于本模型的识别系统在标准数据集上得到了最高识别性能。相比之前最好结果的字符错误率相对降低了近50%，成为新的基准系统。

3. 作为一种新兴的识别方法，基于递归神经网络的框架可以一定程度上克服过切分识别框架的不足，它已经在英文等拉丁语系文字中得到了成功的应用，但在中文字符串识别中研究尚不充分。在手写字符串识别中，学术界一般使用基于二维递归神经网络模块的方法，相比一维序列建模方法可以有更好的效果。但是传统的二维结构在训练过程存在着容易梯度爆炸、难以并行计算的问题。为此，本文提出了一种可分离二维递归神经网络模块，该模块既可以有效地提取多方向的信息，同时可以减少计算资源的消耗并且也容易使用图像处理器进行加速。基于这种全新的二维模块，本文设计了更深的网络结构，并且改良了解码算法。为了缓解训练数据量不足的问题，本文还提出一种结合书写人模态分布的串级别样本合成算法。最终的实验结果表明，该方法的精度比之前的同类型方法有了显著的提升，同时也可以和本文最好结果相比较。

6.2 未来工作展望

尽管文本提出的方法使得中文手写字符串识别系统性能有了进一步提升，但是仍有不少改进的空间。对于过切分识别系统，首先，本文自始至终使用的是最简单的基于字符的语言模型，但是对于中文文本来说，词语是更有意义的语言单元，所以将来的可以考虑融入基于词的神经网络语言，以便能更好地提

升系统识别精度；其次，目前的系统各个模块都是单独训练的，并不能保证全局最优，如果可以进行串级别的联合训练，相信系统的性能会得到提升。

对于基于递归神经网络的识别系统，首先，最重要的是需要克服LSTM-RNN对于语言上下文的过拟合问题，这种过拟合的情况一定程度上阻碍了递归神经网络对于文本行数据的正常建模，可能会由于过拟合问题导致更多的数据也不会提升识别性能；其次，从第五章的残差网络结构出发，可以尝试设计更深层次的结构以便进一步提升系统表现。

实际上，基于递归神经网络的文本识别方法可以进一步推广，和语音识别、机器翻译等序列模式识别结合在一起，成为更一般意义上的序列模式识别问题。在这个层面上，我有如下的思考：

- 模型的可解释性问题。在过切分框架或者是HMM框架中，一般都可以清晰地分析出状态转移的实际物理意义。但是在本文的端到端框架中，我们只能得到一个识别结果，无法知道为什么能得到这样的结果，各个时间步具体对应识别结果的什么部分，RNN就像是一个黑箱一样不可捉摸。这种困惑会导致两个问题：首先，很难进行序列分析的后处理；其次，我们不知道应该采用什么手段去进一步大幅提升性能。近来出现的基于注意力机制（Attention）的模型可以有效地缓解这一问题，但是其在识别性能上和本文的算法还有一定的差距，数据需求量也更大。结合注意力机制的序列建模可能是下一步字符识别技术的一个发展趋势。

- 训练样本规模问题。深度学习时代，大量的训练数据、超量的计算资源似乎已经成为一种标配。这仿佛给人一种错觉，没有大量的数据、没有充足的计算资源，似乎就不能进行深度学习，甚至可以说不能进行科学研究了。这是一个非常危险的信号。本文基于递归神经网络的端到端识别框架中，我们也不能免俗，同样利用了不少计算资源。尽管用合成大量数据的方式缓解了数据量的不足，但是不能从根本上解决对于数据量的依赖。这种趋势实际上违背了很多算法的初衷。比如CNN的设计，就是通过局部连接减小参数量使得网络能训得动进而得到不错的结果，RNN同样也是通过递归连接减小了参数量。所以，使网络变得更小、在更少的数据量上利用更少的资源能够训练出不错的结果应该也是将来的趋势之一。

最后，本文对将来的文档分析系统做一定的展望。一个完整的文本分析系统一般都包含图像预处理、版面分析以及文本行识别等模块。这里实际上也蕴含了一定意义上的Sayre 悖论。比如理论上，我们需要提取文本行就得认得文本行中的内容以便对其进行建模，但是文本行识别却需要提前分割出行图片，这显然又是一个相互交融的问题。现有的系统都是将这些步骤分开，分别优化。如果能够对这些步骤进行联合训练，那么将使得计算机对于文档建模的理解更为全面深入。也许这时就能和真正的“智能”产生一些关联。相信经过研究人员的辛勤努力终能实现这样的畅想。