

河南大学 2022 届本科毕业论文

基于 CNN-RNN-CTC 的身份证识别系统
Citizen ID Card Recognition Based On
CNN-RNN-CTC

论文作者姓名： 鲁晨耕

作者学号： 1825030195

所在学院： 软件学院

所学专业： 软件工程

导师姓名： 王强

导师职称： 副教授

2022 年 4 月 25 日

摘 要

各种身份证件在日常生活中承担着不可或缺的功能，例如身份证、学生证等，而在业务办理过程中也基本需要出示相应证件。随着互联网科技的发展，许多公司为打造自己品牌均开发自己的 APP 产品供用户使用，大多需进行身份实名认证，但不少 APP 的识别机制并不完善，经常将证件信息读取错误而使用户有不好的体验。为了使终端用户在实名认证时更加便捷，响应国家实名制政策，本文提出了基于 CNN-RNN-CTC 的身份证识别系统。

系统使用了 SIFT 特征匹配算法来单步地完成文字区域匹配与矫正，所获取的文字区域可直接作为 CNN-RNN-CTC 的输入，具有准确度较高，速度较快的特点。系统通过 CNN-RNN-CTC 技术构建序列到序列文本字符识别方案，使识别过程无需显式地进行文字分区，而是直接将文字识别作为序列学习问题，对于具有不同尺度、文本长度的输入图像，依靠系统中 CNN 与 RNN，可以对文本字符图像进行序列到序列识别，通过深度学习进行隐式文字分区，显著地提高了文字识别过程的准确度。

关键词：身份证；OCR；深度学习；CNN-RNN-CTC；

ABSTRACT

Various documents play an indispensable role in daily life, such as ID cards, student ID cards, etc., and corresponding documents are needed to be shown when handling business. With the development of Internet technology, many companies have developed their own APP products for their users in order to build their own brands, most of which need to carry out real-name authentication. However, the identification mechanism of many APPs is not perfect, and the identification information is often read incorrectly, resulting in unsatisfactory user experience. In order to make real-name authentication more convenient for end users and respond to the national real-name system policy, this thesis proposes an ID card OCR system based on CNN-RNN-CTC.

The system uses the SIFT feature matching algorithm to complete the text area matching and correction in a single step. The acquired text area can be directly used as the input of CNN-RNN-CTC, which has the characteristics of high accuracy and fast speed. Through the end-to-end text recognition based on CNN-RNN-CTC, the recognition process does not need explicit text segmentation, but converts text recognition into a sequence learning problem. For input images with different scales and different text lengths, through the CNN and RNN layers, the entire text image can be recognized, and the text cutting can be directly integrated into deep learning, achieving better accuracy in the recognition process.

Keywords: Citizen ID Card; OCR; Deep Learning; CNN-RNN-CTC

目 录

摘 要	II
ABSTRACT	III
第 1 章 绪 论	3
1.1 课题来源	3
1.2 课题背景	3
1.3 国内外在该方向的研究现状及分析	4
1.3.1 传统 OCR 方法	4
1.3.2 基于深度学习的两步 OCR 方法	5
第 2 章 相关技术简介	6
2.1 OpenCV 工具包	6
2.2 SIFT 特征匹配算法	6
2.3 CNN-RNN-CTC 神经网络	7
2.3.1 CNN 卷积神经网络	8
2.3.2 RNN 循环神经网络	8
2.3.3 CNN-RNN-CTC OCR 模型	9
第 3 章 居民身份证数据准备	11
3.1 数据格式	11
3.2 身份证区域提取	12
3.3 生成文字序列训练数据	16
第 4 章 系统实现	19
4.1 CNN-RNN-CTC 网络构建	19
4.1.1 网络 CNN 构建	20
4.1.2 网络 RNN 构建	23
4.1.3 网络静态计算流图构建	24
4.2 CNN-RNN-CTC 网络训练	25
4.2.1 神经网络 Trainer 类构建	25

4.2.2 神经网络训练与评估..... 29

4.3 公民身份证识别过程构建..... 31

结 论..... 34

参考文献..... 35

致 谢..... 36

第1章 绪 论

本章说明了此课题的课题来源、课题研究背景、国内外当前研究现状以及课题可行性分析。

1.1 课题来源

光学文本字符识别（Optical Character Recognition, OCR）指对于图像中文字区域进行识别的过程，其自动将图像中手写文字或印刷文本转换为相应文字的计算机字符编码，从而使文字能够被计算机直接处理。文字识别是计算机视觉研究领域的一大任务，通过 OCR 能够处理来源于不同场景的图像，如拍摄或扫描得到的各种卡证图像，被广泛运用于工业、商业和金融等领域。随着互联网科技的发展，许多公司为打造自己品牌，均开发自己的 APP 产品供用户使用，基于身份证的实名认证机制更是嵌入了众多的 APP，而目前现有的通用 OCR 技术在此应用场景下仍存在着识别效果不好、识别速度过慢等种种问题，难以应对日益广泛的认证需求，本课题响应国家实名制政策，研究针对公民身份证的专用 OCR 技术，提出一种基于 CNN-RNN-CTC 的新型身份证 OCR 技术，能够有效提高身份证 OCR 识别的准确度与速度，大大改善用户的实名认证体验。为公民身份证识别任务提供了又一种普适且可行的解决方法。

1.2 课题背景

计算机视觉领域中，从文本图像中分析和理解高级语义信息相关的场景文本识别 (STR) 具有一定的复杂性。场景文本识别系统已广泛成功地应用于各种应用，例如图像检索、驾驶员辅助系统、证件文档信息的提取与识别等。此类系统通常由两大阶段所组成：场景文本探测与文本鉴别。文本检测旨在从图像中提取文本的位置。文本识别用于将目标文本转化为文本序列。

先前关于场景文本识别的诸多研究都是针对非结构化文本所提出的，适用于未知布局、复杂背景、各种视角、照明等的无规则文本。传统非结构化文本识别技术往往包含例如 HOG 描述符、笔画宽度变换 (SWT)、最大稳定极值区域 (MSER) 等手工特征提取技术或如条件随机场 (CRF)、支持向量机 (SVM) 等传统分类器。由于传统技术大量使用上述统计机器学习方法，系统在识别过程中会累积各步骤错误。以致传统方法使

检测和识别系统变得复杂且效率低下。

深度学习推动了文本检测和识别的改进，传统场景文本识别系统的缺点进而得以克服。在文本检测领域，基于整体嵌套边缘检测 (HED) 的全卷积网络 (FCN) 被用于生成包含文本区域信息及其关系的全局特征图，文本检测问题进而被视为语义分割问题。EAST 检测器使用基于多通道 FCN 的模型进行文本检测，从而检测到各方向的单词或文本行。而联接文本提议网络 (CTPN) 能够直接定位图像中的文本区域，CTPN 使用 VGG16 模型作为特征提取过程，该网络应用了垂直锚定机制，用于在精细范围内提高文本位置的准确性。对于基于深度学习的端到端的文本识别系统，为实现进行高精度的文本识别，其训练过程需要大量训练数据。由于文本图像数据集通常数据均无法对齐。需要如联接时间分类等额外技术来使标签序列映射至模型最终文本序列输出。除此之外，卷积神经网络 (CNN) 与循环神经网络 (RNN) 已广泛成为文本识别任务的基础工具。由两者所组成的 CNN-RNN-CTC 通过联接时间分类训练的 2D-LSTM 识别文本行。模型使用 CNN 层从文本图像中学习序列特征表示，而后将特征序列馈送到 BiLSTM 以学习特征的上下文信息。最后，由联接时间分类从输入图像中获取文本序列。

1.3 国内外在该方向的研究现状及分析

目前国内外对于证件识别任务的研究，可分为基于传统算法，通过过程化图像处理与统计机器学习方法从图像中提取文本序列的传统 OCR 方法，与基于深度学习进行文本检测与识别的 OCR 方法。

1.3.1 传统 OCR 方法

传统 OCR 方法按算法流程划分为四个阶段，各阶段分别为预处理、文字定位、文字识别和后处理。但其对不同的光照环境，字体间差异等对其影响较大，不适用于复杂的应用环境。传统方法特征单一，算子均由人工指定，能够表征的信息有限，无法提取深层语义信息。针对某些简单的文本识别应用场景，传统 OCR 技术尚能取得相对较高识别精确度。然而传统方法是对于特定场景应用的建模，一旦应用于不同场景，模型即会丧失有效性。随着深度学习技术在文本识别任务上的应用，基于深度学习的 OCR 技术也已日趋主流，且其对于不同应用场景均具有一定的普适性。

1.3.2 基于深度学习的两步 OCR 方法

基于深度学习的 OCR 技术则具有自动优势，其基于深度学习的自动的特征表示学习可以使研究人员摆脱凭经验设计、手工制作的模型设计过程，使其更容易推广至多种应用场景。深度学习 OCR 技术由文本检测与文本识别两步组成，其中文本检测定位目标影像中文本行并交予文本识别系统进行识别，文本检测技术中较有代表性的便是 CTPN，其由经典的目标检测模型 Faster RCNN 改进而来，模型结合了 CNN 与 LSTM 深度网络，从而能够直接在卷积层中定位文本行。

在通过某种文字检测技术对目标影像中文字区域进行定位后，仍需要对区域内文字进行识别。而基于深度学习 OCR 技术能够进行端到端不定长文本序列识别，通过主要用于端到端不定长文本序列识别的 CRNN 识别算法，无需先将字符序列分区为单字符，而是将文本识别转化为时序依赖的序列识别问题，极大的提高了文字识别的实时性。

第2章 相关技术简介

本章主要介绍系统开发时所涉及的算法模型与系统所依赖的外部工具包以及系统用于文字识别的神经网络架构。

2.1 OpenCV 工具包

OpenCV（开源机器视觉库）是一种开放源代码的机器视觉工具包，其中包含着与机器视觉相关的多种机器学习方法。OpenCV 为机器视觉的工程应用提供通用的基础工具，从而加速机器视觉技术的开发与应用。OpenCV 使用 BSD 开源许可证，可以自由用于商业与非商业化用途。

该工具包拥有数千种高速算法，其中有一系列传统机器视觉算法和相应机器学习算法。算法可用于面容检测与面容识别、物体识别、视频人类行为分类、镜头运动跟踪、移动物体追踪、物体 3D 模型提取、立体影像三维模型生成、高分辨率场景图像拼接，图像数据库相似图像查找，去除红眼，眼球运动跟踪等多种应用。OpenCV 拥有完善的开源社区，是应用最为广泛的机器视觉工具包。该工具包被商业机构、研究机构和政府机构广泛使用。

本身份证识别系统通过调用 OpenCV 的 SIFT 特征匹配算法，来完成对身份证图片区域的提取。

2.2 SIFT 特征匹配算法

固定尺度特征变换 (SIFT) 是一种应用于探测和表示目标图像中的物体局部特征的算法。SIFT 定位目标对象图像中某些关键点，提取关键点区域附近的图像特征信息作为描述符，通过匹配目标对象图像中的关键点与输入图像中的关键点，识别输入图像中的目标对象。描述符应该对于各种对目标对象的变换不敏感，从而有效的在不同输入图片中识别目标对象。

SIFT 算法具有以下特性：

- 局部性：特征是局部的；对于光照条件与角度不敏感。
- 独特性：所提取的单个特征可以匹配到大型对象数据集。

- 高特征数量：使用 SIFT，可以从较小对象中提取大量特征。
- 速度：SIFT 可以进行实时的特征提取与匹配。

SIFT 提取目标特征需以下步骤：

- 尺度空间极值检测：识别可以在不同图像中标志同一物体的位置与尺度。
- 关键点定位：拟合模型以确定特征的定位和大小，根据稳定性测量选择关键点。
- 关键点方向分配：计算每个关键点区域的最佳方向。
- 关键点描述：使用选定比例和旋转的局部图像梯度来描述每个关键点区域。

尺度空间极值检测：

现实世界的对象只有在一定的尺度上才有意义。物体的这种多尺度性质在非常普遍。尺度空间极值检测利用物体的这种视觉特性，识别目标对象的视觉特征。通过尺度空间极值检测，识别在同一物体或场景的不同角度下重复出现的定位和尺度。检测时在多个尺度上搜索对于不同视图的稳定特征。

关键点定位：

由于尺度空间极值检测将会产生很多特征，其中一些特征位于边缘或者没有足够的对比度，这类特征无法作为目标对象的关键点，通过关键点定位，SIFT 算法选择目标对象的关键点。关键点定位通过计算前一阶段找到的关键点的拉普拉斯算子实现。

关键点方向分配：

为了实现对于图像旋转无关的特征检测效果，需要针对关键点的方向进行计算，通过关键点的邻域信息，计算邻域梯度的大小和方向，进行关键点的方向分配。

关键点描述：

此时，每个关键点都具有相应的定位、比例、朝向信息。由此得到每个关键点的局部图像区域的描述符，该描述符对于视点和光照的不同等不敏感，具有较高的特殊性。

2.3 CNN-RNN-CTC 神经网络

CNN-RNN-CTC 神经网络又被称为卷积循环神经网络，其由卷积神经网络层与循环神经网络层组成，将文本的探测与识别融合到同一个系统中，是一种序列到序列文本识别模型。

2.3.1 CNN 卷积神经网络

卷积神经网络，简称为 CNN 或 ConvNet，是一类用于处理具有网状维度输入数据的深度神经网络，多用于图像处理。数字图像是视觉数据的二进制表示。数字图像包含网状排列的像素阵列，其中包含像素值，表示矩阵每个像素的亮度和颜色。

人脑在我们看到图像的那一刻就处理了大量的信息。视神经元工作于各自感受区，以覆盖视野的方式与其他神经元建立连接。CNN 中的每个神经元仅在其感受区中处理数据，类似于动物视觉中动物视觉神经对感受区内刺激产生反应。CNN 各层的排列方式使其首先检测单一的视觉元素（线条、曲线等），继而检测复合的视觉信息（面部、物体等）。通过使用 CNN，能够有效的将深度学习应用于机器视觉。

卷积层是 CNN 的组成部分。该层在两个矩阵之间进行点乘，其中参数矩阵含有一组可学习参数，被称为核，另一矩阵是卷积神经网络的视觉感受区。内核在空间上比图像小，但更深入。这意味着，如果网络输入是由三个 (RGB) 通道组成的彩色图像，内核将具有较小的高度和宽度，但其深度会扩展到所有三个通道。在前向传播期间，内核在图像的高度和宽度上滑动，从而产生该感受区域的图像表示。前向传播产生了图像表示张量，称为激活图，它是内核在图像的每个空间位置响应的表示。内核的滑动大小称为步幅。

卷积神经网络能够通过应用内核来捕获图像中的空间信息与时间信息。由于网络内核较少的参数数量以及权重的可复用性，该架构的神经网络可以更好地应用于图像处理任务。

2.3.2 RNN 循环神经网络

循环神经网络 (RNN) 是一种神经网络架构，循环神经网络由若干节点序列组成，节点之间形成时间序列的有向或无向连接。这使其能够提取输入中的时序特征。循环神经网络以传统线性神经网络为基础，能够保存一系列记忆数据，通过其对不定长度的输入数据进行处理。这使其能够被用于例如不存在文本分区的手写文字识别或场景语音识别等应用场景。

RNN 拥有内部“记忆”，记忆单元中存在着所有步骤 t 之前的输入数据。之所以其被称为循环神经网络，因为该网络对序列各个元素执行相应的运算，且网络输出由网

络记忆与当前网络中元素值共同决定。在传统的线性神经网络中，输入被馈送至输入网络层并由多个中间网络层进一步处理并产生网络输出，这类神经网络假设两个连续输入彼此独立或在时间步 t 的输入与时间步 $t-1$ 输入无关。

循环神经网络具有多种应用，例如基于 RNN 的信息序列分类或预测任务，除此之外其还具有以下典型应用：

- 语音分类：-例如，对来自男性和女性声音的声音样本进行区分。此任务具有变长输入，定长输出
- 情感分类：可以利用 RNN，根据文本的情感的不同，将文本（如电影或产品评论）分类为不同的类别
- 序列转换：将上游网络的输出序列进行转换，如 CNN-RNN-CTC 网络中，将 CNN 的输出特征转换为文本序列

2.3.3 CNN-RNN-CTC OCR 模型

本系统用于身份证文本识别的模型为卷积循环神经网络 (CNN-RNN-CTC)，网络结合了卷积神经网络 (CNN) 和循环神经网络 (RNN)，循环神经网络构建了用于序列识别的端到端系统。该模型由三层模型组成，分别为卷积神经网络层、循环神经网络层和转录输出层。卷积层作为输入图像的特征序列提取器。在卷积神经网络之后，构建了多层循环神经网络，用于对卷积层输出张量，即特征张量的每一元素进行变换。模型最后的转录输出层将循环神经网络层的每元一素转化为输出文本。虽然 CNN-RNN-CTC 模型是由不同种类的神经网络 (CNN 和 RNN) 组成的异构网络，但它可以通过同一种损失函数进行同时训练。

- 基于 CNN 进行输入图像特征张量提取：

在 CNN-RNN-CTC 模型中，卷积层的组成部分是通过卷积层和最大池化层来构建的。此类神经网络层用于从输入图像中提取图像特征张量。

- 基于 RNN 进行序列标记：

模型通过在卷积层之后建立双向循环网络作为模型的循环神经网络层，用于基于图像特征的序列标记。

- 转录：

转录是将循环神经网络所输出的每元素预测张量转换为输出文字的过程。

CNN-RNN-CTC 网络在转录过程中使用联接时间分类 (CTC) 来解码 RNN 的输出并将其转换为文本标签。

联接时间分类 (CTC)：

文本字符区域识别任务属于序列标记任务，序列标注任务由输入序列 $X=[x_1, x_2, \dots, x_T]$ 及其对应的输出序列 $Y=[y_1, y_2, \dots, y_U]$ 组成。此任务能够确定从 X 到 Y 的准确映射，由于即 X 和 Y 的长度可变，且 X 序列与 Y 序列缺乏元素间对其关系。文本序列识别任务通常使用联接时间分类进行输出的推断。对于特定的 X ，CTC 能够穷举出所有候选 Y 的输出序列。进而能够使用此序列集来得到可能的文本序列或估算文本序列的概率。

第3章 居民身份证数据准备

对于进行文字序列识别的 CNN-RNN-CTC 网络，需要通过现有的居民身份证数据进行监督训练，由于该类数据敏感性强，来源有限，本系统使用了 BDCI 2019 比赛的 OCR 数据集进行系统训练。

3.1 数据格式

BDCI 2019 身份证 OCR 数据集由训练集数据，训练集表情，以及评估数据组成




Name	Date modified	Type	Size
 Test.zip	8/18/2019 3:52 PM	WinRAR ZIP archive	1,180,793 KB
 Train_DataSet.zip	8/18/2019 3:53 PM	WinRAR ZIP archive	2,363,960 KB
 Train_Labels.csv	12/6/2019 2:53 PM	Microsoft Excel Com...	1,797 KB

图3.1 身份证OCR数据集格式

其中，数据标签为 csv 格式，其中以文本形式存储训练集身份证图片相对应的居民姓名、民族、性别、身份证号码、住址等信息。

A	B	C	D	E	F	G	H	I	J	K	L	M
eb6fb57d17244f388144fbd5a3a3fdf	全香玉	土家	男	1969	8	14	四川省甘	51330119690814900	甘孜藏族	2018.10.02-长期		
96bb9d7a269d4592bcbcd12ba6b57f00	王昌良	塔吉克	男	1975	6	3	广东省江	4.40781E+17	江门市台	2017.09.17-2037.09.17		
c4f708ad1aba4af88a638ad6a6715481	谭芸	汉	男	1987	11	14	广东省河	4.41625E+16	河源市东	2015.05.24-2035.05.24		
faa5a2a64b594ca4980fc46f92c76a8a	莹桂英	佤	男	1983	9	16	新疆维吾	6.52301E+17	昌吉回族	2015.04.18-2035.04.18		
70b98bbdc1a04f0e8d66eaf5432a21a8	党辉	撒拉	男	1971	10	14	海南省海	4.60107E+16	海口市琼	2015.06.02-2035.06.02		
e2f5b51bdc450e89ec3addf05e3d73	周榕榕	白	男	1967	6	2	河北省张	1.30731E+16	张家口市	2018.11.24-长期		
dc940a1ac2a84b9a91510f9c6436fffc	邱大文	高山	女	1986	11	6	湖北省武	4.20114E+16	武汉市蔡	2015.08.27-2035.08.27		
3055c207aa174712bce75b48b635a68f	喻国友	傣	女	1963	1	22	江西省赣	3.60729E+16	赣州市全	2014.01.25-长期		

图3.2 身份证OCR数据训练集标签

训练集及测试集数据由身份证正反面的灰度图片组成。



图3.3 身份证OCR训练集数据

3.2 身份证区域提取

身份证 OCR 识别的第一步应是从图片中分离出身份证图像区域，从而进行后续的身份证 OCR 文字识别及 CNN-RNN-CTC 文字序列识别网络的训练。通过使用 SIFT 特征提取与来自 OpenCV 中 calib3d 模块的同位图匹配功能，能够将身份证图像区域从复杂的背景中分离出来。

SIFT 过程使用了一个 queryImage，并于其中获得了若干特征点，与此同时对于另一个 trainImage，同样也获得了该图像中的特征，并在其中找到了最佳匹配。简而言之，SIFT 在含有身份证区域图像中定位了身份证区域的位置。

为了有效地从 trainImage 中提取身份证区域，需要手动设计 queryImage，其中应包含身份证区域所共有的特征，通过手动提取某些训练数据的身份证区域，得到 queryImage 如下。



图3.4 身份证背面区域queryImage

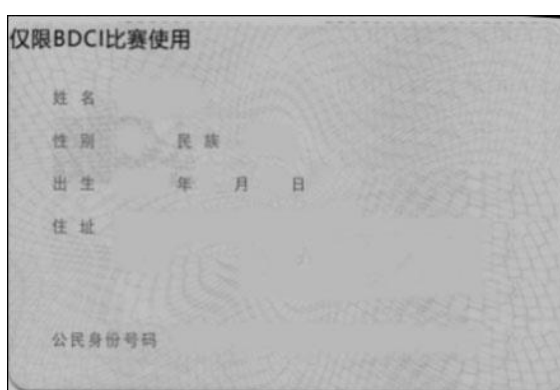


图3.5 身份证正面区域queryImage

为了将身份证区域从 queryImage 中提取出来，该区域提取过程使用了同位图技术，并通过调用 OpenCV 工具包中 calib3d 模块的函数，即 cv.findHomography()来实现此过程。通过 SIFT 过程，能够从 trainImage 与 queryImage 中获取一组相关的点，进而找到对应于身份证区域的透视变换。通过得到的透视变换，系统使用 cv.perspectiveTransform() 来提取身份证区域。

身份证识别系统为身份证区域提取提供了 HomographyPreprocessor 类，由属性默认值为 10 的 MIN_MATCH_COUNT 属性；构造方法；_detect 方法；crop 方法。

```
import re
from os import listdir
from os.path import isfile, join
import numpy as np
import cv2
from matplotlib import pyplot as plt
class HomographyPreprocessor:
    MIN_MATCH_COUNT = 10
    def __init__(self, interactive=False):
```

```
...
def _detect(self, kp, des, tem, train_img):
...
def crop(self, input_img):
...

```

其中 MIN_MATCH_COUNT 属性用于设置 trainImage 与 queryImage 之间相关点的最小数量，小于此数量则匹配失败。构造方法用于初始化 SIFT 匹配器，加载身份证两面 queryImage，从而初始化 HomographyPreprocessor 类。

```
def __init__(self, interactive=False):
    self.interactive = interactive
    self.sift = cv2.xfeatures2d.SIFT_create()
    self.front_tem = cv2.imread('./data/front.jpg', 0)
    self.front_tem_kp, self.front_tem_des = self.sift.detectAndCompute(self.front_tem, None)
    self.back_tem = cv2.imread('./data/back.jpg', 0)
    self.back_tem_kp, self.back_tem_des = self.sift.detectAndCompute(self.back_tem, None)

```

_detect 方法为 HomographyPreprocessor 类主要方法，通过该方法进行身份证区域的检测与提取，该方法首先提取 trainImage 中特征点，并通过 cv2.FlannBasedMatcher 类，调用该类的 knnMatch 方法匹配 queryImage 与 trainImage 中的若干特征点，由 cv2.findHomography 方法定位身份证区域并使用 cv2.warpPerspective 方法进行区域的视角变换，进而取得 trainImage 中的身份证区域。

```
def _detect(self, kp, des, tem, train_img):
    input_kp, input_des = self.sift.detectAndCompute(train_img, None)
    FLANN_INDEX_KDTREE = 0
    ind_para = dict(algo=FLANN_INDEX_KDTREE, tree=10)
    search_para = dict(checks=50)
    flann = cv2.FlannBasedMatcher(index_params, search_params)
    matches = flann.knnMatch(des, input_des, k=2)
    # store all the good matches as per Lowe's ratio test.
    good = []
    for m, n in matches:
        if m.distance < 0.7 * n.distance:
            good.append(m)
    if len(good) > self.MIN_MATCH_COUNT:
        src_pts = np.float32([kp[m.queryIdx].pt for m in good]).reshape(-1, 1, 2)
        dst_pts = np.float32([input_kp[m.trainIdx].pt for m in good]).reshape(-1, 1, 2)
        M, mask = cv2.findHomography(dst_pts, src_pts, cv2.RANSAC, 5.0)
        matches_mask = mask.ravel().tolist()
        h, w = tem.shape
        # pts = np.float32([[0, 0], [0, h - 1], [w - 1, h - 1], [w - 1, 0]]).reshape(-1, 1, 2)
        card = cv2.warpPerspective(train_img, M, (w, h))

```

```

else:
    print("Not enough matches are found - %d/%d" % (len(good),
self.MIN_MATCH_COUNT))
    matches_mask = None
    return None
draw_params = dict(matchColor=(0, 255, 0), # draw matches in green color
                    singlePointColor=None,
                    matchesMask=matches_mask, # draw only inliers
                    flags=2)
img3 = cv2.drawMatches(tem, kp, train_img, input_kp, good, None, **draw_params)
if self.interactive:
    cv2.imshow('img3', img3)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
return card

```

`_detect` 方法可对 SIFT 特征匹配区域进行可视化，体现 SIFT 的特征匹配与视角变换过程。



图3.6 身份证正面区域匹配



图3.6 身份证背面区域匹配

3.3 生成文字序列训练数据

为了对 CNN-RNN-CTC 文字序列识别网络进行训练，需要生成相应的文字序列训练数据，通过调用 HomographyPreprocessor 类并进行文字区域切分，得到神经网络的训练数据集。系统为数据集预处理提供了 data_processor 脚本，脚本由入口与两函数组成，其中 arrange_lines 函数将多行文本转换为单行文本，click_event 函数用于过程可视化的事件响应，脚本首先进行 HomographyPreprocessor 类的实例化，而后对训练集文件进行遍历。

```
import re
from os import listdir
from os.path import isfile, join
import cv2
import numpy as np
from ocr.HomographyPreprocessor import HomographyPreprocessor
```

```
def arrange_lines(img: np.ndarray):
...
def click_event(event, x, y, flags, params):
...
```

```
interactive = False
hcc = HomographyPreprocessor(interactive=interactive)
```

```

path = r'D:\citizenIdData\Train_DataSet'
files = [f for f in listdir(path) if isfile(join(path, f)) and re.match('.*\.\jpg', f)]
#files = ['0e8df807a2a641b2851114bed1cda7a0.jpg']
for file in files:
    img = cv2.imread(path + '\\' + file)
    front, back = hcc.crop(img)
    if front is not None and back is not None:
        if interactive:
            cv2.imshow('back', back)
            cv2.setMouseCallback('back', click_event)
            cv2.imshow('front', front)
            cv2.setMouseCallback('front', click_event)
            cv2.waitKey(0)
            cv2.destroyAllWindows()
        myDict = {
            "pd": back[204:240, 169:388],
            "period": back[245:280, 168:363],
            "name": front[48:84, 71:153],
            "sex": front[86:119, 76:113],
            "ethnic": front[84:117, 176:245],
            "birth": front[112:146, 74:243],
            "address": arrange_lines(front[151:225, 78:300]),
            "id": front[230:272, 123:385]
        }
        for k, v in myDict.items():
            cv2.imwrite(r'D:\citizenIdData\etc_train' + '\\' + file + '-' + k + '.jpg', v)

```

对于每一个训练集数据项，进行其文字区域的获取，并将其作为独立图片保存，图片名称的格式为：来源图片名称-文本区域类型，脚本具有交互选项，能够将文字区域获取过程可视化。

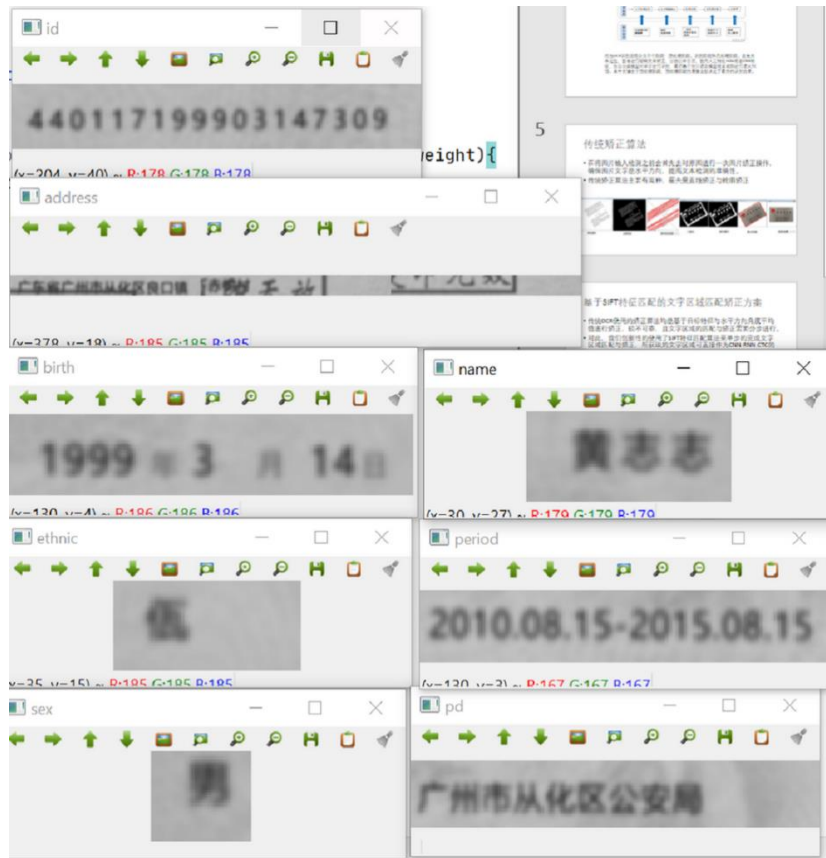


图3.7 身份证文字区域

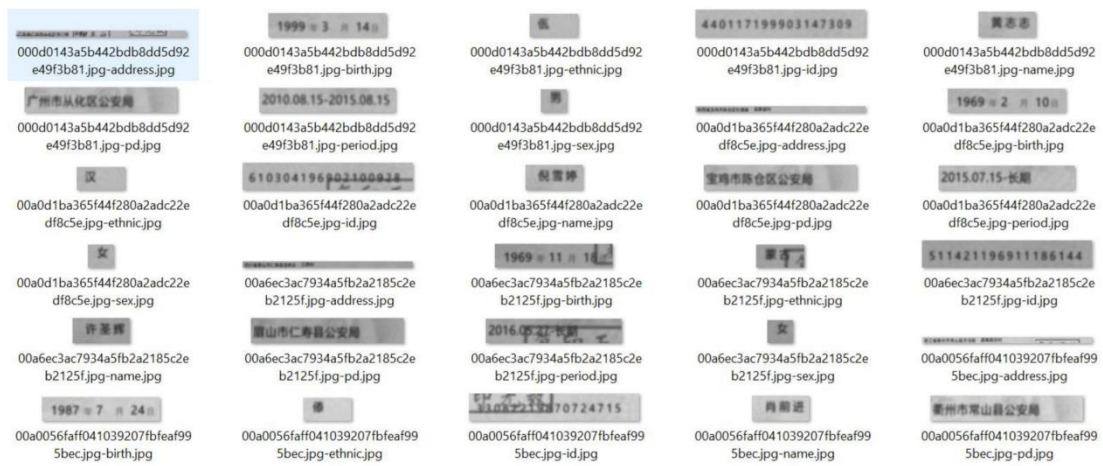


图3.8 神经网络训练集数据

第4章 系统实现

身份证识别系统由 Python 语言实现，通过 OpenCV 实现输入图像的预处理，由 TensorFlow 框架构建文字序列识别网络，并于训练集数据之上训练。以得到用于证件文本的文本识别神经网络，系统通过将证件影像进行预处理，并将其结果输入神经网络进行推断操作，基于该网络的推断结果，得到相应证件信息。

4.1 CNN-RNN-CTC 网络构建

CNN-RNN-CTC 网络由三层网络组成，分别为卷积神经网络层、循环神经网络层和转录输出层。卷积神经网络层作为输入图像的特征序列提取器。卷积神经网络层输出被循环神经网络层作为输入，用于对图像特征张量的每一元素进行变换。最后，转录输出层或 CTC 层用于将循环神经网络层的每元素输出转化为输出文本。

身份证识别系统通过使用 TensorFlow，利用 TensorFlow 静态计算流图，构建 CNN-RNN-CTC 网络，静态计算流图是由一组 `tf.Operation` 对象组成的的数据结构，`tf.Operation` 对象是对神经网络层的抽象。静态计算流图使用 `tf.Tensor` 对象作为网络输入的抽象，表示在操作之间流动的数据单元。

使用静态计算流图，可以使网络在没有 Python 解释器的环境中使用 TensorFlow 图，有利于提高系统的可移植性。在神经网络训练完毕后，将静态计算流图作为保存模型的格式。

身份证识别系统中使用 CRNN 类进行网络结构定义，该类位于 `crnn.py` 脚本中。

```
import numpy as np
import tensorflow as tf
from tensorflow.contrib import rnn
from utils.net_cfg_parser import parser_cfg_file

class CRNN(object):

    def __init__(self, net_params, inputs, seq_len, batch_size, trainable=False, pretrain=False):

        self._input_height = int(net_params['input_height'])
        self._input_width = int(net_params['input_width'])
        self._class_num = int(net_params['classes_num'])
        self._inputs = inputs
```

```

        self._batch_size = batch_size
        self._seq_len = seq_len
        self._trainable = trainable

    def construct_graph(self):
        """
        构建网络
        :return:
        """
        ....

    def _cnn(self, inputs):
        """
        cnn 网络结构
        :param inputs:
        :return:
        """
        ...

    def _rnn(self, inputs, seq_len):
        """
        双向 rnn
        :return:
        """
        ...

```

该类由构造方法，`construct_graph` 方法、`_cnn`、`_rnn` 方法构成，其中构造方法定义了神经网络输入影像的维度，词汇总数，以及序列最大长度。`_cnn`、`_rnn` 方法分别定义了网络的 CNN、RNN。`construct_graph` 方法构造了网络的总体结构。

4.1.1 网络 CNN 构建

网络 CNN 涉及卷积层、最大池化层和批量归一化、校正线性单元 (ReLU) 激活等运算符，用于从文本行图像中提取序列特征。该部分是基于 VGG 架构的思想设计的，其中提取的特征用于创建特征向量，并将特征序列用作 RNN 的输入。

因为 CNN 在相应位置捕获输入数据的局部特征，网络 CNN 非常适合基于图像的序列识别问题，它将文本行图像转换为单词或字符的特征序列，这是文本识别的一项重要步骤。通过 CRNN 类的 `_conv2d` 方法，能够构建神经网络的 2 维 CNN 层。

```

def _conv2d(self, inputs, filters, padding, batch_norm, name):
    if batch_norm:
        activation = None

```



```

else:
    activation = tf.nn.relu
    kernel_initializer = tf.contrib.layers.variance_scaling_initializer()
    bias_initializer = tf.constant_initializer(value=0.0)
    top = tf.layers.conv2d(inputs,
                           filters=filters,
                           kernel_size=3,
                           padding=padding,
                           activation=activation,
                           kernel_initializer=kernel_initializer,
                           bias_initializer=bias_initializer,
                           name=name)

    if batch_norm:
        top = tf.layers.batch_normalization(top, axis=3, training=self._trainable, name=name)
        top = tf.nn.relu(top, name=name + '_relu')
    return top

```

函数指定了校正线性单元 (ReLU)为卷积层的激活函数，指定权重的初始化方式，并调用 TensorFlow 的静态运算图 API 的 tf.layers.conv2d 函数创建该卷积层。

```

def _cnn(self, inputs):
    """
    cnn 网络结构
    :param inputs:
    :return:
    """
    conv1=self._conv2d(inputs=inputs,filters=64,
padding="valid",batch_norm=False,name='conv1')
    conv2=self._conv2d(inputs=conv1,filters=64,
padding="same",batch_norm=True,name='conv2')
    pool1 = tf.layers.max_pooling2d(inputs=conv2, pool_size=2, strides=[2,2], padding='valid')

    conv3=self._conv2d(inputs=pool1,filters=128,padding="same",batch_norm=True,name='conv3')
    conv4=self._conv2d(inputs=conv3,filters=128,padding="same",batch_norm=True,
name='conv4')
    pool2 = tf.layers.max_pooling2d(inputs=conv4, pool_size=2, strides=[2, 1], padding="valid")
    conv5=self._conv2d(inputs=pool2,filters=256,padding="same",batch_norm=True,
name='conv5')
    conv6 = self._conv2d(inputs=conv5, filters=256, padding="same", batch_norm=True,
name='conv6')
    pool3 = tf.layers.max_pooling2d(inputs=conv6, pool_size=2, strides=[2, 1], padding="valid")

    conv7 = self._conv2d(inputs=pool3, filters=512, padding="same", batch_norm=True,
name='conv7')
    conv8 = self._conv2d(inputs=conv7, filters=512, padding="same", batch_norm=True,

```

```
name='conv8')
    pool4 = tf.layers.max_pool2d(input=conv8, pool_size=[3,1], stride=[3,1], pad="valid")
    # 去掉维度为 1 的维度
    features = tf.squeeze(pool4, axis=1, name='features')
    return features
```

CRNN 类通过 `_cnn` 方法构建网络 CNN，该部分由卷积和池化（下采样）层组成，网络卷积部分的前若干层通常提取较基础的特征，例如不同向的边缘等。而后续层则倾向于提取更复杂的特征。池化卷积层用于对特征进行向下样本采集。隐藏层的输出特征维度较大，若特征维度逐渐下降，网络通常会更快拟合。池化层有助于减少所需参数数量，减少了模型推断时所需的计算量。池化有助于避免过拟合。

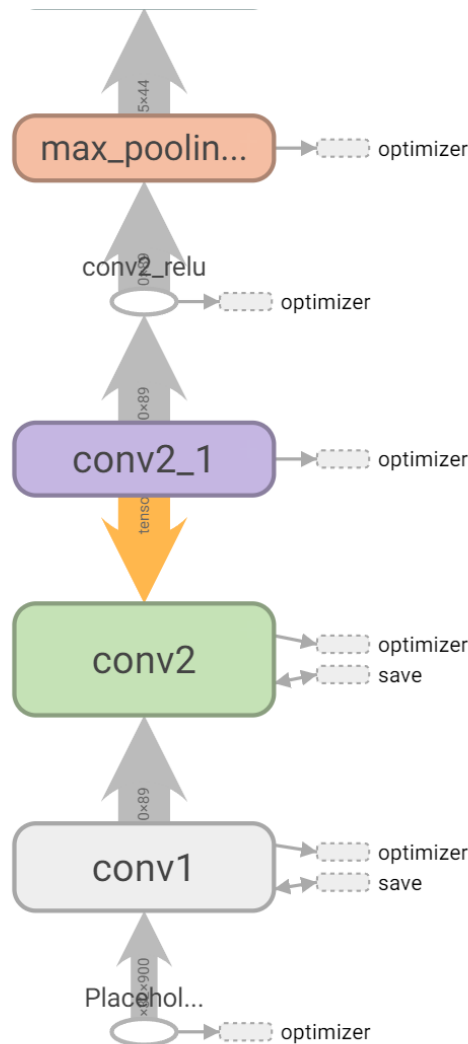


图4.1 网络CNN可视化结构

4.1.2 网络 RNN 构建

神经网络 CNN 产生的特征张量输入网络的 RNN，从而进行之后的序列预测任务，网络 RNN 采用了 LSTM 层根据特征张量预测输出序列词汇。LSTM 非常适合用于基于图像的变长序列识别任务。标准的 LSTM 架构由一个存储过去和未来信息的存储单元组成；三个逻辑门结构：输入逻辑门、输出逻辑门和遗忘逻辑门。输入门发现输入值流入 LSTM 单元的程度。输出门控制存储单元的值，用于决定输出的程度。输入和输出门使得 LSTM 单元长时间存储和检索信息。遗忘门用于发现 LSTM 单元中剩余的值并清除其存储。因此，LSTM 可以充分提取文本序列中经常出现的长距离依赖。通过 CRNN 类的 `_rnn` 方法，能够构建神经网络的 LSTM 层。

```
def _rnn(self, inputs, seq_len):
    """
    双向 rnn
    :return:
    """
    with tf.variable_scope(None, default_name="bidirectional-rnn-1"):
        # 前向 rnn
        lstm_fw_cell_1 = rnn.BasicLSTMCell(256)
        # 反向 rnn
        lstm_bw_cell_1 = rnn.BasicLSTMCell(256)
        inter_output, _ = tf.nn.bidirection_dyn_rnn(lstm_fw_cell_1, lstm_bw_cell_1, inputs, s_len,
dtype=tf.float)
        inter_output = tf.concat(inter_output, 2)
        with tf.variable_scope(None, default_name="bidirectional-rnn-2"):
            # 前向 rnn
            lstm_fw_cell_2 = rnn.BasicLSTMCell(256)
            # 反向 rnn
            lstm_bw_cell_2 = rnn.BasicLSTMCell(256)
            outputs, _ = tf.nn.bidirection_dyn_rnn(lstm_fw_cell_2, lstm_bw_cell_2, inter_output, s_len,
dtype=tf.float)
            outputs = tf.concat(outputs, 2)
            return outputs
```

`_rnn` 方法共创建了 2 层双向循环神经网络层，分别命名为 `bidirectional-rnn-1` 与 `bidirectional-rnn-2`，对于每一循环神经网络层，通过调用 TensorFlow `rnn.BasicLSTMCell` 方法，构建了由宽度为 256 维的 LSTM 网络层组成的正向循环神经网络与反向循环神经网络，并在此基础上，调用 `tf.nn.bidirectional_dynamic_rnn` 方法，创建由前向循环神经网络与反向循环神经网络所构成的双向循环神经网络层。

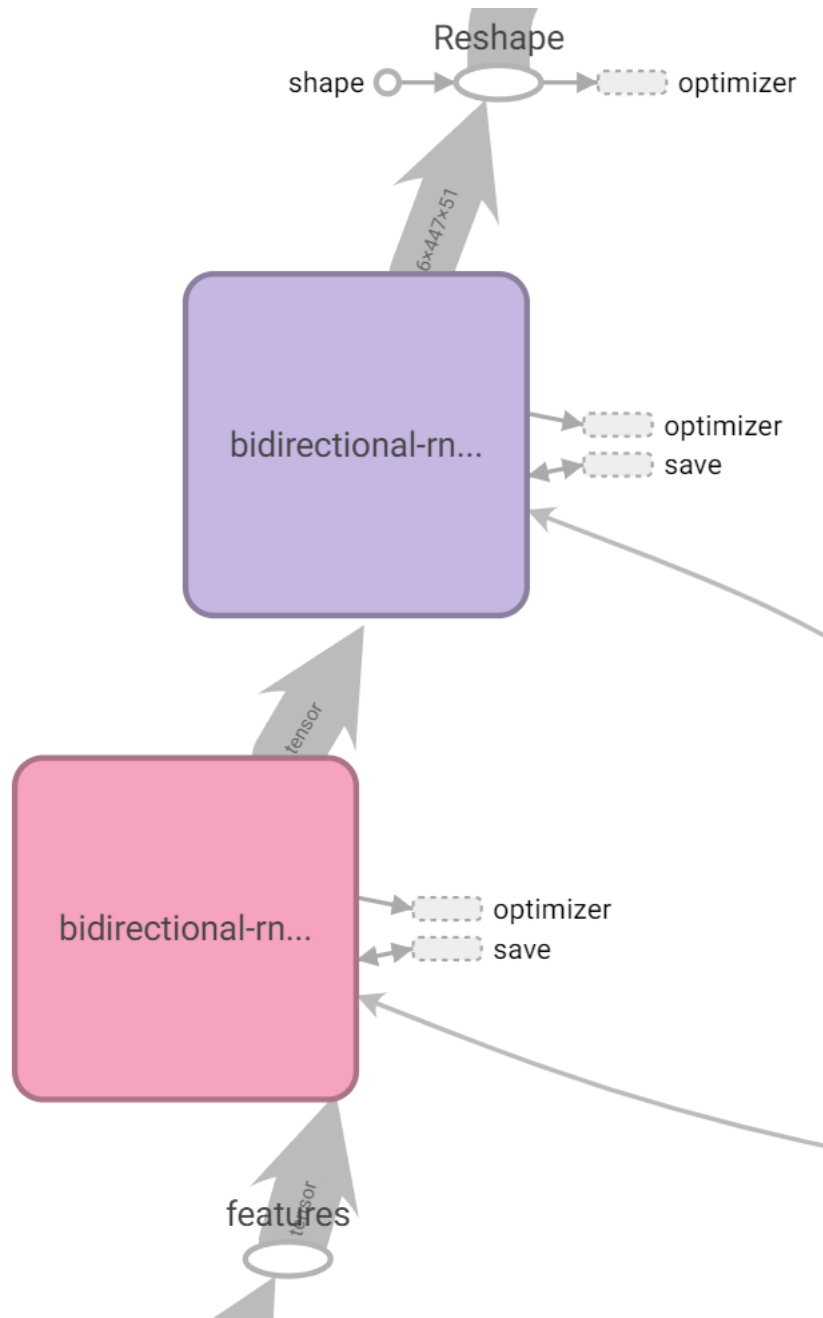


图4.2 网络RNN可视化结构

4.1.3 网络静态计算流图构建

CRNN 类在实例化时调用其 `construct_graph` 方法以初始化神经网络的静态计算流图，`construct_graph` 方法定义了网络的输入输出，并调用 `_cnn` 方法与 `_rnn` 方法对网络进行构建。

```
def construct_graph(self):
    """
    构建网络
```

```

:return:
"""
# 进入 cnn 网络层 shape [batch, length, 32, 1]
cnn_out = self._cnn(self._inputs)
# 送入 rnn 前将 cnn 进行 reshape
max_char_count = cnn_out.get_shape().as_list()[1]
print(max_char_count)
crnn_model = self._rnn(cnn_out, self._seq_len)
logits = tf.reshape(crnn_model, [-1, 512])
W = tf.Variable(tf.truncated_normal([512, self._class_num], stddev=0.1), name="W")
b = tf.Variable(tf.constant(0., shape=[self._class_num]), name="b")
logits = tf.matmul(logits, W) + b
logits = tf.reshape(logits, [self._batch_size, -1, self._class_num])
# 网络层输出
net_output = tf.transpose(logits, (1, 0, 2))
# 解析网络输出
decoded, log_prob = tf.nn.ctc_greedy_decoder(net_output, self._seq_len)
return net_output, decoded, max_char_count

```

方法首先根据输入影像的最大维度调用 `_cnn` 方法构建网络 CNN, 根据 CNN 输出张量的维度, 输出网络支持的最大文本序列长度, 并调用 `_rnn` 方法构建网络 RNN, 至此网络结构构建完成, 方法进而调用 `tf.nn.ctc_greedy_decoder` 为模型附加输出解码器, 并返回模型静态计算流图、解码器及文本序列最大长度。

4.2 CNN-RNN-CTC 网络训练

对于训练数据集, $C = \{d_i, C_i\}$, 其中 d_i 为用于训练的身份证文本行影像而 C_i 为数据标签, 最小化负指数条件可能性的 CTC 损失函数由如下公式所示。

$$C = - \sum_{d_i, C_i \in D} \log p(C_i | y_i)$$

基于该损失函数, 通过训练数据反向传播过程, 使用损失梯度下降 (SGD) 方法对神经网络模型进行训练。由模型的正向数据传播得到模型损失值, 并通过反向数据传播算法向着模型误差减小方向对神经网络参数进行逐层优化。此外, 训练过程通过 ADADELTA 算法针对模型每维的学习速率进行优化。

4.2.1 神经网络 Trainer 类构建

系统构建了用于模型训练、训练集数据加载、模型检查点保存的 `Train_CRNN` 类。

```
import time
import logging
import numpy as np
import tensorflow as tf
from crnn import CRNN
from dataload import Dataload
from utils.net_cfg_parser import parser_cfg_file
```

```
class Train_CRNN(object):
    def __init__(self, pre_train=False):
        ...
    def train(self):
        ...
    def _train_logger_init(self):
        """
        初始化 log 日志
        :return:
        """
        ...
```

该类由构造方法、含有训练循环的 `train` 方法，与负责 TensorBoard 训练日志初始化的 `_train_logger_init`，通过指定该类构造方法的 `pre_train` 参数，能够导入先前所训练的模型参数，构造函数由模型配置文件中读取如输入维度，模型学习速率等参数，并加载可能存在的现有模型参数，之后通过调用模型 `CRNN` 类对模型静态计算流图进行构建，以进行之后的模型训练。

```
def __init__(self, pre_train=False):
    net_params, train_params = parser_cfg_file('./net.cfg')
    self.input_height = int(net_params['input_height'])
    self.input_width = int(net_params['input_width'])
    self.batch_size = int(train_params['batch_size'])
    self._learning_rate = float(train_params['learning_rate'])
    self._max_iterators = int(train_params['max_iterators'])
    self._train_logger_init()
    self._pre_train = pre_train
    self._model_save_path = str(train_params['model_save_path'])
    if self._pre_train:
        ckpt = tf.train.checkpoint_exists(self._model_save_path)
        if ckpt:
            print('Checkpoint is valid...')
            f = open('./model/train_step.txt', 'r')
            step = f.readline()
            self._start_step = int(step)
```

```

        f.close()
    else:
        assert 0, print('Checkpoint is invalid...')
    else:
        self._start_step = 0
        self._inputs = tf.placeholder(tf.float32, [self.batch_size, 32, self.input_width, 1])
        # label
        self._label = tf.sparse_placeholder(tf.int32, name='label')
        # The length of the sequence [32] * 64
        self._seq_len = tf.placeholder(tf.int32, [None], name='seq_len')
        crnn_net = CRNN(net_params, self._inputs, self._seq_len, self.batch_size, True)
        self._net_output, self._decoded, self._max_char_count = crnn_net.construct_graph()
        self.dense_decoded = tf.sparse_tensor_to_dense(self._decoded[0], default_value=-1)

```

Train_CRNN 类的 train 方法包含着模型的训练循环，描述了模型的训练过程，方法于训练循环之前通过调用 TensorFlow 的 tf.nn.ctc_loss 方法定义模型的损失函数为 CTC 损失，通过使用 tf.train.AdamOptimizer 方法为模型训练指定优化器为 Adam，并将模型输出准确度作为模型训练的度量，且对模型的自动保存与日志进行相关配置。

```

def train(self):
    with tf.name_scope('loss'):
        loss = tf.nn.ctc_loss(self._label, self._net_output, self._seq_len)
        loss = tf.reduce_mean(loss)
        tf.summary.scalar("loss", loss)
    with tf.name_scope('optimizer'):
        train_op = tf.train.AdamOptimizer(self._learning_rate).minimize(loss)
    with tf.name_scope('accuracy'):
        accuracy = 1 - tf.reduce_mean(tf.edit_distance(tf.cast(self._decoded[0], tf.int32),
self._label))

        accuracy_broad = tf.summary.scalar("accuracy", accuracy)
    data = Dataload(self.batch_size, './dataset/citizen_id_label.txt', './dataset/ctc_train',
                    img_height=self.input_height, img_width=self.input_width)
    # 保存模型
    saver = tf.train.Saver()
    # tensorboard
    merged = tf.summary.merge_all()
    ...

```

方法接下来进行对训练循环的定义，训练循环将训练数据集条目依次输入到模型中，根据训练数据标签向模型损失减小的方向进行网络参数的调整。外部循环的每一次迭代都完成一次对训练数据集的遍历，将此迭代称为时期(Epoch)，在一个时期内，循环遍历训练数据集中的每个条目，获取其输入 (x) 和标签 (y)。通过数据条目的输入，

得到模型的预测输出并将其与数据条目标签进行比较。通过损失函数测量模型预测的不准确性并计算模型的损失和梯度。使用优化器更新模型的参数。同时，利用 TensorBoard 对训练相关数据进行统计与可视化，以上过程不断重复直到模型拟合。

```

with tf.Session() as sess:
    if self._pre_train:
        saver.restore(sess, self._model_save_path)
        print('load model from:', self._model_save_path)
    else:
        sess.run(tf.global_variables_initializer())
    train_writer = tf.summary.FileWriter("./tensorboard_logs/", sess.graph)
    epoch = data.epoch
    for step in range(self._start_step + 1, self._max_iterators):
        batch_data, batch_label = data.get_train_batch()
        feed_dict = {self._inputs: batch_data,
                      self._label: batch_label,
                      self._seq_len: [self._max_char_count] * self.batch_size}
        summ = sess.run(merged, feed_dict=feed_dict)
        train_writer.add_summary(summ, global_step=step)
        sess.run(train_op, feed_dict=feed_dict)
        if step%20 == 0:
            train_loss = sess.run(loss, feed_dict=feed_dict)
            self.train_logger.info('step:%d, total loss: %6f % (step, train_loss))
            self.train_logger.info('compute accuracy...')
            train_accuracy = sess.run(accuracy, feed_dict=feed_dict)
            val_data, val_label = data.get_val_batch(self.batch_size)
            val_accuracy = sess.run(accuracy, feed_dict={self._inputs: val_data,
                                                         self._label:
val_label,
                                                         self._seq_len:
[self._max_char_count] * self.batch_size})

            self.train_logger.info('epoch:%d, train accuracy: %6f % (epoch,
train_accuracy))

            self.train_logger.info('epoch:%d, val accuracy: %6f % (epoch, val_accuracy))
        if step%100 == 0:
            self.train_logger.info('saving model...')
            f = open('./model/train_step.txt', 'w')
            f.write(str(self._start_step + step))
            f.close()
            save_path = saver.save(sess, self._model_save_path)
            self.train_logger.info('model saved at %s' % save_path)
        if epoch != data.epoch:

```



```
epoch = data.epoch
self.train_logger.info('compute accuracy...')
train_accuracy = sess.run(accuracy, feed_dict=feed_dict)
self.train_logger.info('epoch:%d, accuracy: %6f % (epoch, train_accuracy))
summ = sess.run(accuracy_broad, feed_dict=feed_dict)
train_writer.add_summary(summ, global_step=step)
train_writer.close()
```

4.2.2 神经网络训练与评估

通过 Python 解释器执行系统的 train_crnn.py 脚本，即可开始模型的训练过程。

```
if __name__ == "__main__":
    train = Train_CRNN(pre_train=True)
    train.train()
```

对于模型训练过程中所需要的超参数，如模型训练学习速率、数据批量大小、模型参数衰减与模型训练时期等选项，均可在系统根目录下的 net.cfg 配置文件中进行配置。对于此次训练，过程采用了大小为 16 的数据批量，学习速率为 0.0001。

在模型的训练过程中，主要通过 TensorBoard 对模型的多种度量进行监测，TensorBoard 是用于提供机器学习工作流程中所需的测量和可视化等过程的工具。它可以跟踪实验指标，如损失和准确性、可视化模型图、将嵌入投影到低维空间等等。在执行系统的 train_crnn.py 脚本之后，通过命令提示符调用 tensorboard --logdir tensorboard_logs 命令，即可启动 TensorBoard 服务，使用浏览器载入系统页面，能够实时检测模型训练状态。

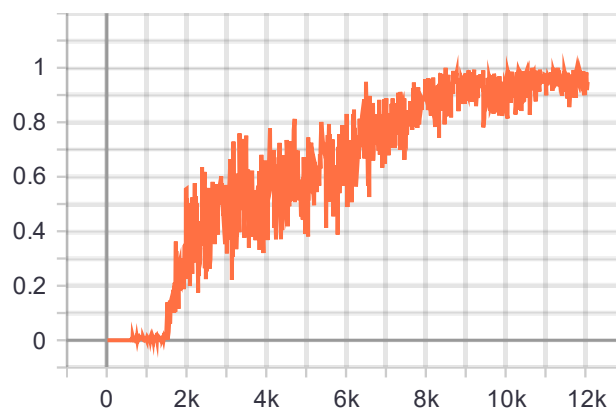


图4.3 模型训练过程Accuracy统计

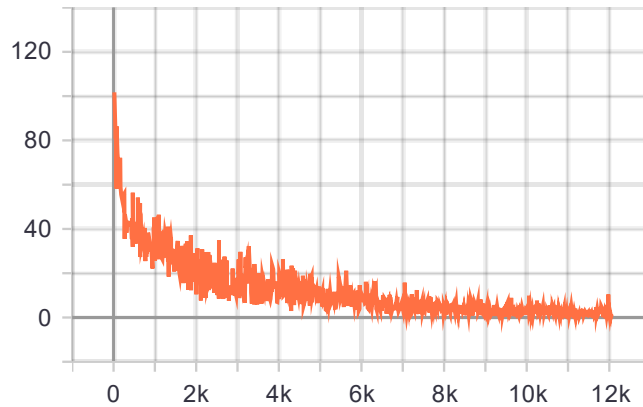


图4.3 模型训练过程Loss统计

对于模型的训练过程，每经过一步（Step），就进行一次模型参数权重更新，即进行一次学习，每一次更新参数都需要一个数据批量(Batch Size)的样本进行运算学习，根据运算结果调整更新模型参数。模型经 12K 步数的权重调整，模型于训练数据集之上达到了 0.95 的准确度，于验证数据集上达到了 0.97 的准确度。

本文所提出的文本识别方法相较于以往文本识别方法体现出了额外的有效性。每个居民身份证图像包括多个文本行区域。通过提取的文本行图像，对该文本序列识别神经网络进行评估，可以得出该方法在准确率、召回率和 F1 分数方面均高于传统的文本识别方案。实验分析表明，本文所提出的文本识别方法相比于传统方法产生了额外的识别精确度。

表4.1 文本识别方法有效性比较

方法	准确度 (%)	召回率 (%)	F1 分数 (%)	用时 (s)
传统识别方法	88.6	89.0	88.8	0.027
CNN-RNN-CTC 方法	97.1	97.0	97.0	0.402

将验证数据集按照信息文本类型进行分类并分别进行模型准确性评估，如表 1 所示。其中各文本行由 SIFT 特征算法提取后通过人工方式验证，而后将其输入 CNN-RNN-CTC 文本识别模型进行相关评估。该表表明，本文所提出的文本识别方法在公民身份证识别任务上产生了优秀的表现。所提出的方法在出生日期、姓名、性别、有效期和身份证号等信息字段中获得了非常高的准确性。

表4.2 模型于不同文本类型准确性

证件文本类型	文本错误率（%）
公民身份证号码	4.66
姓名	0.79
出生年月	0.00
性别	0.22
住址	12.64
签发机关	0.12
证件有效期	0.04

4.3 公民身份证识别过程构建

本身份证识别系统识别过程共分为文本检测与文本识别两步过程，其中文本检测过程系统使用固定尺度特征变换算法（SIFT 定位目标对象图像中证件关键点并提取相关文字区域作为文本识别过程神经网络输入，模型文本识别过程基于 CNN-RNN-CTC 文本识别模型，以证件文本区域影像为输入进行模型推断，从而获得证件文字域文本序列，最终系统将证件信息以结构化数据返回用户。对于公民身份证识别过程，系统构建了 Test_CRNN 类用于文本域提取以及模型推断，Test_CRNN 类由构造方法、get_text_img 等方法构成，其中构造方法首先根据配置文件 net.cfg 中模型权重路径，加载上文所训练模型并从系统词汇列表中加载预训练模型词汇，最终创建用于模型推断的 TensorFlow TrainSaver 对象。

```
self.interactive = interactive
    net_params, train_params = parser_cfg_file('./net.cfg')
    self._model_save_path = str(net_params['model_load_path'])
    self.input_img_height = int(net_params['input_height'])
    self.input_img_width = int(net_params['input_width'])
    if batch_size is None:
        self.test_batch_size = int(net_params['test_batch_size'])
    else:
        self.test_batch_size = batch_size
```

```

# 加载 label onehot
f = open('./dataset/citizen_id_words.txt', 'r', encoding='gbk')
data = f.read()
words_onehot_dict = eval(data)
self.words_list = list(words_onehot_dict.keys())
self.words_onehot_list = [words_onehot_dict[self.words_list[i]] for i in
range(len(self.words_list))]
# 构建网络
self.inputs_tensor=tf.placeholder(tf.float32,[self.test_batch_size,self.input_img_height,
self.input_img_width, 1])
self.seq_len_tensor = tf.placeholder(tf.int32, [None], name='seq_len')
crnn_net = CRNN(net_params, self.inputs_tensor, self.seq_len_tensor, self.test_batch_size,
True)

net_output, decoded, self.max_char_count = crnn_net.construct_graph()
self.dense_decoded = tf.sparse_tensor_to_dense(decoded[0], default_value=-1)
self.sess = tf.Session()
saver = tf.train.Saver()
saver.restore(self.sess, self._model_save_path)

```

模型通过 Test_RCNN 类中 get_text_img 方法实现由公民身份证图像至文本的识别过程，方法首先初始化用于 SIFT 特征匹配过程的 HomographyPreprocessor 类，用于提取身份证影像内文字区域，通过调用其 crop 方法，分别获得图像中所含身份证正反面区域，并通过相对位置裁剪各文本行并获得 id_dict 列表。

```

id_dict = [
    ("pd", back[204:240, 169:388]),
    ("period", back[245:280, 168:363]),
    ("name", front[48:84, 71:153]),
    ("sex", front[86:119, 76:113]),
    ("ethnic", front[84:117, 176:245]),
    ("birth", front[112:146, 74:243]),
    ("address", arrange_lines(front[151:225, 78:300])),
    ("id", front[230:272, 123:385])
]

```

get_text_img 方法通过调用类中 _get_batch_input 方法，将以 OpenCV 矩阵类型存储的文本行数据转化为 TensorFlow 格式张量，并将其组合成同一批次，同时对该张量进行归一化处理，并返回归一化后的张量。

```

resized_img = self._resize_img(img)
reshape_img=resized_img.reshape([1,self.input_img_height,self.input_img_width, 1])
img_norm = reshape_img / 255 * 2 - 1
batch_data[i] = img_norm

```

在数据归一化操作后，方法创建用于模型前向迭代的 `feed_dict`，其中包含归一化后的张量图像数据，通过 `TensorFlow Session.run` 方法，执行模型推断操作，并调用 `Test_RCNN` 类 `_predict_to_words` 方法，根据一位有效编码原则，对模型输出进行解码，以获得模型最终输出。

```
feed_dict = {self.inputs_tensor: batch_data, self.seq_len_tensor: [self.max_char_count] * batch_size}
predict = self.sess.run(self.dense_decoded, feed_dict=feed_dict)
predict_seq = self._predict_to_words(predict)
```

结 论

本文通过研究联接时间分类损失和与 CNN-RNN-CTC 模型在文本序列识别任务的应用，结合固定尺度特征变换算法，提出了公民身份证识别任务的解决方案。利用固定尺度特征变换算法对于光照条件与角度不敏感特点，通过识别身份证件固定的视觉特征，与传统方法相比能够更加可靠的提取出身份证相应文字区域。通过使用端到端文字识别模型 CNN-RNN-CTC，将文字区域识别问题作为序列识别问题，提高了文字识别准确度。通过实验结果证明，本文所提出方法相较于传统方法识别准确性存在显著提高且对于不同的图像噪声不敏感，在多种光照条件与摄像角度均能够保持较高的可靠性。模型在应用于其他语言时也保持了其通用性。该方案相比于传统方法拥有较高的准确性，同时其对硬件算力也有较高要求，系统运算用时均高于传统方案一个数量级，后续计划通过使用模型蒸馏等技术着力降低 CNN-RNN-CTC 模型规模，以降低系统的总体算力需求，并应用多种文本识别错误修正技术，以提高文本识别模型准确度。

参考文献

- [1] Jorge Martinez-Gil. SIFT: An Algorithm for Extracting Structural Information From Taxonomies [DB/OL].
- [2] Kaiming He, Xiangyu Zhang, Shaoqing Ren et al. Deep Residual Learning for Image Recognition [DB/OL].
- [3] Alex Graves, Santiago Fernandez, Faustino Gomez et al. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks [DB/OL].
- [4] Baoguang Shi, Xiang Bai, Cong Yao. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition [DB/OL].
- [5] Zhi Tian, Weilin Huang, Tong He et al. Detecting Text in Natural Image with Connectionist Text Proposal Network [DB/OL].
- [6] Shaoqing Ren, Kaiming He, Ross Girshick et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [DB/OL].
- [7] Yi Hao, Ayush Jain, Alon Orlitsky et al. SURF: A Simple, Universal, Robust, Fast Distribution Learning Algorithm [DB/OL].
- [8] Dat, T. T., Tran, L., Truong, N. N., Cung, P., Ngoc, V. et al. An improved CRNN for Vietnamese Identity Card Information Recognition [DB/OL].
- [9] Sanjana Gunna, Rohit Saluja, C. V. Jawahar. Transfer Learning for Scene Text Recognition in Indian Languages [DB/OL].
- [10] Franz Anders, Ammie K. Kalan, Hjalmar S. Kühl. Compensating class imbalance for acoustic chimpanzee detection with convolutional recurrent neural networks [DB/OL].

致 谢

在学士学位论文即将完成之际，我想向曾经给我帮助和支持的人们表示衷心的感谢。首先要感谢我的导师王强副教授，他在学习和科研方面给了我大量的指导，并为我提供了良好的科研环境，让我学到了知识，掌握了科研的方法，也获得了实践锻炼的机会。他严谨的治学态度、对我的严格要求以及为人处世的坦荡将使我终身受益。除此之外，他对我生活的关心和照顾也使得我得以顺利完成本科生的学业。在此祝愿他身体健康，全家幸福！

最后，衷心感谢在百忙之中评阅论文和参加答辩的各位专家、教授！

G· 格子达论文检测报告【简版】

报告编号:A00F742F7D5640E29AEC7DE39A0F082D

送检文档:题目待定, 后续修改

作者:鲁晨耕

送检单位:河南大学

送检时间:2022-04-21 10:54:01

比对索引库

1989-01-01至2022-04-21

学术期刊库

报纸资源库

本科论文共享库

格子达公示库

学位论文库

互联网资源库

专利库

机构自建库

会议论文库

格子达多元库

检测结果

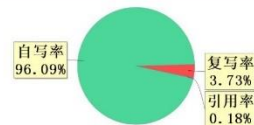
总相似比:3.91%(总相似比=复写率+引用率)

学术检测指标:自写率96.09%复写率3.73%引用率0.18%(含自引率0.0%)

格式检测指标:主体完整 不规范引用句子数8

其他类型检测结果:去除引用后总相似比:3.73%

相似片段:复写片段14引用片段8



指标名称	学校要求	指标检测结果	系统判定
总相似比	不超过20%	3.91%	符合
论文总字数	不少于1000字/单次	33384字	符合

其他检测结果：

指标名称	识别数量	系统判定
代码块检测	20	复写1

复写率索引来源

学术期刊: (0.94%)

学位论文: (1.58%)

本科论文共享库: (0.64%)

格子达源代码: (0.57%)

引用率片段来源

学术规范引用（0.18%） 自我引用（0.0%） 其他引用（0.0%）

免责声明

- 1、本报告为G·格子达系统检测后自动生成，鉴于论文检测技术及论文检测样本库的局限性，G·格子达不保证检测报告的绝对准确，您所选择的检测资源范围内的检验结果及相关结论仅供参考，不得作为其他任何依据；
- 2、G·格子达论文检测服务中使用的论文样本，除特别声明者外，其著作权归各自权利人享有。根据《中华人民共和国著作权法》等相关法律法规，G·格子达网站仅为学习研究、介绍、科研等目的引用论文片段。除非经原作者许可，请勿超出合理使用范围使用本网站提供的检测报告及其他内容。

联系我们



防伪二维码



关注微信公众号

官方网站:co.gochek.cn
客服热线:400-699-3389
客服QQ:800113999