**软件设计说明书**

**项目名称：基于深度学习的银行卡号识别系统**

**小组名称：Leet Code**

**参与人员：黄新昊 陈硕峰**

**2019 年 5 月 20日**

目录

[1引言 3](#_Toc28061)

[1.1研究背景及意义 3](#_Toc7494)

[1.2系统需求 3](#_Toc19229)

[1.3系统概述 4](#_Toc7248)

[2角色及职责 4](#_Toc9471)

[3工具及环境 5](#_Toc12493)

[3.1研发硬件配置 5](#_Toc3008)

[3.2研发语言及编译器 5](#_Toc28669)

[3.3软件支持工具 5](#_Toc21367)

[3.4第三方工具 6](#_Toc27177)

[4概要设计 7](#_Toc19881)

[4.1总体设计 7](#_Toc11158)

[4.1.1整体的处理流程 7](#_Toc15092)

[4.1.2总体结构设计 7](#_Toc2582)

[4.1.3模块功能 8](#_Toc31467)

[4.2接口设计 10](#_Toc19440)

[4.3运行设计 10](#_Toc29640)

[4.4出错处理设计 10](#_Toc23351)

[5 详细设计 10](#_Toc14579)

[5.1软件结构概述 10](#_Toc4309)

[5.2获取银行卡号图像 11](#_Toc28260)

[5.3图像预处理 11](#_Toc15556)

[5.4卡号定位 12](#_Toc6251)

[5.5字符分割 14](#_Toc32086)

[5.6字符识别 14](#_Toc16995)

[6程序描述 14](#_Toc17789)

[6.1程序功能和性能描述 15](#_Toc31838)

[6.1.1图像预处理。 15](#_Toc12898)

[6.1.2银行卡卡号的字符分割 15](#_Toc28363)

[6.1.3字符识别 15](#_Toc4834)

[6.2程序输入输出项目 15](#_Toc31901)

[7程序算法 16](#_Toc22424)

[7.1图像灰度化 16](#_Toc28559)

[7.2图像增强 16](#_Toc6309)

[7.3边缘检测 18](#_Toc18274)

[7.4RGB与YUV颜色模型转换 21](#_Toc1626)

[7.5高斯滤波 23](#_Toc19544)

[7.6大津阈值二值化 24](#_Toc1120)

[7.7支持向量机 26](#_Toc17972)

[7.8基于卷积神经网络识别字符 28](#_Toc27427)

[7.9LeNet-5结构 31](#_Toc1156)

[7.10限制玻尔兹曼机模型 33](#_Toc2436)

[8程序接口 36](#_Toc2570)

[8.1图形化用户界面 36](#_Toc4378)

[8.2图像预处理模块 36](#_Toc31855)

[8.3银行卡号定位模块 37](#_Toc10641)

[8.4银行卡号切割模块 37](#_Toc17075)

[8.5银行卡号字符识别模块 37](#_Toc3047)

[9用户手册 37](#_Toc20466)

# 

# **1引言**

# **1.1研究背景及意义**

## 在继续大资金操作时，人们一般选择直接对银行卡进行操作。在输入银行卡号时，由于银行卡号数字较多，输入时容易输入错误造成不便，因此通过拍照进行银行卡自动识别可以大大提高互联网金融行业以及电子支付中设计银行卡操作的处理效率，实现银行卡的有效管理及相关服务，提升用户体验等，所以，银行卡自动识别具有很大的应用前景。

## **1.2系统需求**

**功能性需求**

1.数据集处理

根据本赛题提供的数据集（共1084张卡号截图及标签）实现数据增强模块，将数据集中的每一张图片使用数据增强方式拓展为80张图片，为之后的图像识别训练提供充足的数据样本，并且该模块程序能够继续处理新加入的数据样本。

2、程序定位银行卡卡号

实现银行卡号文本的检测定位，此模块能够将拍摄的银行卡卡号部分检测出来，并截取相应部分供后续的识别模型使用（银行卡尽量充满图片，横向放置）。

3.程序实现银行卡号字符识别

此模块要求使用数据增强后的数据集训练字符识别模型，最终模型能够识别出赛题提供的测试银行卡卡号以及自拍的银行卡卡号。

4.采用GUI（图形用户界面）

提供良好的用户交互式界面实现。

**非功能性需求**

1. 程序源码要求结构清晰，模块区分较为明确，提供便于读者阅读的源码指南，及项目使用文档。
2. 使用人工智能领域深度学习技术进行实现，提供文本检测模型、文本识别模型。

3.使用GPU(图形处理器)加速计算。

## **1.3系统概述**

目前人们实际使用的银行卡卡号有16或19为数字组成，银行卡大小基本统一，但银行卡背景多种多样、卡号有平面印刷、凹凸压印等不同方式、银行卡在使用过程中存在不同程度的磨损等。

本系统通过卡号定位，获取卡号的精确位置，然后进行字符分割，将分割出的单个数字字符图像送入机器学习模型中进行训练识别，从而达到能够识别各种银行卡卡号的效果。

1. 图像预处理

图像预处理时对所得的银行卡图像进行分割前的相应处理工作，首先进行银行卡图像的灰度化，并通过一系列的处理进行噪声去除和图像增强，最后进行数字字符串图像的精确定位，分割出卡号区域，如图1。



图 1

1. 字符分割

字符分割是将分割出来的卡号区域图像切割成地理且互不影响的单个数字字符图像。字符分割的准确度直接影像到字符识别的准确率，字符分割不准确会引起后期的字符识别的误差。本系统能够消除背景干扰，进行数字的精确分割。

1. 字符识别

识别分割出的数字字符是银行卡号识别系统的最后一步。目前用于字符识别的方法主要有基于模板匹配的字符识别、提取特征运用SVM分类器的字符识别、基于神经网络的字符识别等。由于银行卡号字体不是固定字体，所以模板匹配的方法不能对所有类型的银行卡均达到较高的识别率。提取特征并运用SVM分类器对字符进行识别是最常用的字符识别方法。基于深度学习的字符识别则具有识别率高、抗干扰能力强等特点。本系统基于卷积神经网络的深度学习进行训练来识别字符。

# **2角色及职责**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **姓名** | **角色** | **职责** |
| 1 | 黄新昊 | 项目经理、程序员 | 负责项目过程的监控和管理；  负责编写卡号定位和模型训练代码；  负责软件质量控制；  负责系统的设计； |
| 2 | 陈硕峰 | 程序员、UI设计员 | 负责进行UI界面的设计；  负责可视化界面（web）搭建；  负责整合可视化界面和卡号识别程序； |

表1

# **3工具及环境**

## **3.1研发硬件配置**

本系统卡法所用硬件是实验室电脑，配置如下：

|  |  |
| --- | --- |
| **硬件** | **详细信息** |
| CPU | Intel(R) Core (TM)2 Duo CPU E6550 @ 2.33GHz |
| 主板 | Inter(R)82865G |
| 内存 | 1GB |
| 显卡 | Inter(R) Q33 Express Chipset Family |
| 显示器 | HP L1710 17寸液晶显示器 |
| 硬盘 | 日立 HDS721616PLA380 80G |
| 网卡 | Realtek RTL8139 Family PCI Fast Ethernet NTC#2 |
| 系统 | Microsoft Windows XP Professiona 12002 Service Pack2 |

表2

## **3.2研发语言及编译器**

本系统开发所用语言是Python，开发IDE是JetBrains PyCharm 2018.1.4

## **3.3软件支持工具**

开发使用Rose、PyCharm等软件工具

## **3.4第三方工具**

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **程序包、工具** | **版本** | **用途** |
| 1 | pillow | 5.2.0 | 图像处理库。 |
| 2 | opencv-python | 4.0.0.21 | 实现了[图像处理](https://baike.baidu.com/item/%E5%9B%BE%E5%83%8F%E5%A4%84%E7%90%86/294902" \t "https://baike.baidu.com/item/opencv/_blank)和计算机视觉方面的很多通用算法。 |
| 3 | numpy | 1.16.2 | Python 的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。 |
| 4 | tensorflow-gpu | 1.13.1 | TensorFlow是Google开源的第二代用于数字计算（numerical computation）的软件库。它是基于数据流图的处理框架，用于各类[机器学习](https://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0/217599" \t "https://baike.baidu.com/item/TensorFlow/_blank)（machine learning）算法的编程实现。 |
| 5 | Django | 1.9.13 | 一个python写成的开放源代码的Web应用框架。 |
| 6 | MySQL | 5.5.40 | 关系型数据库管理系统 |
| 7 | PIL | 5.2.0 | Python平台图像处理标准库 |

表3

# **4概要设计**

## **4.1总体设计**

### **4.1.1整体的处理流程**

银行卡号识别是将获取的银行卡卡号图像，先进性卡号定位，获取卡号的精确位置。然后进行字符分割，将分割出的单个数字字符图像送入机器学习模型中进行训练识别，主要过程及流程如图2。

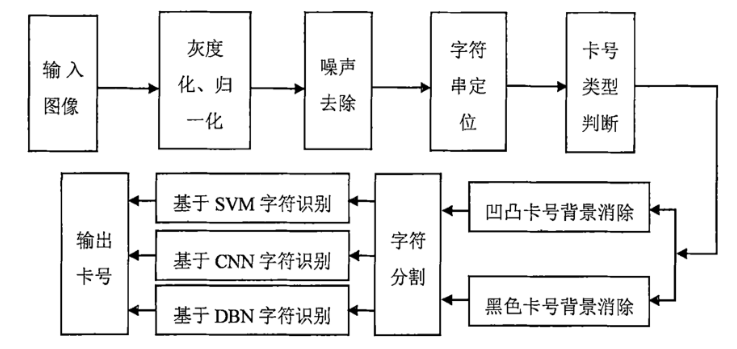


图 2

### **4.1.2总体结构设计**

系统总体结构分为五大模块（如图3所示）

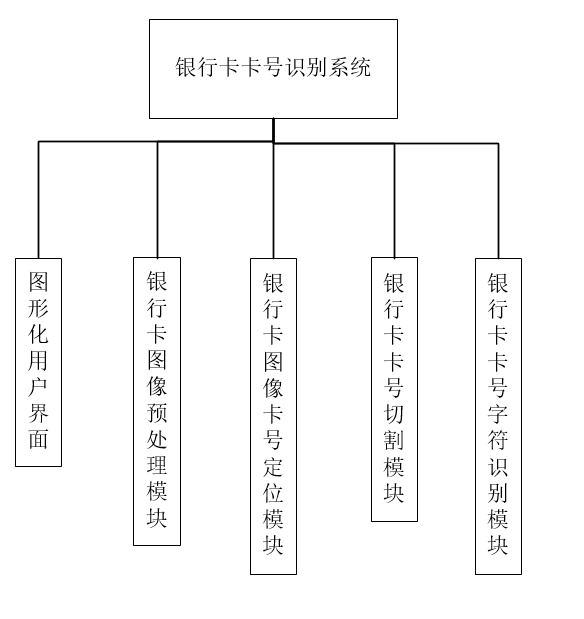


图 3

### **4.1.3模块功能**

1. 图形化用户界面

该模块用于上传用户的银行卡图片，并在处理完成后返回卡号。该模块包含图像预览、上传、卡号显示功能，如图4。

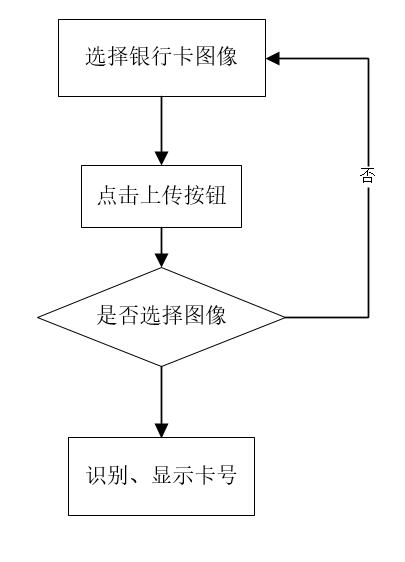


图 4

1. 银行卡图像预处理模块

该模块用于对银行卡图像进行灰度化，并通过一系列的处理进行噪声去除和图像增强，为接下来的卡号定位做准备。

图像灰度化方法一般有如下三种：

* 最大值法：将灰度图像的灰度值去R、G、B三个值中最大的一个。灰度图像的灰度用gray表示，即gray = max(R,G,B)。
* 平均值法：将灰度图像的灰度值取R、G、B三个值的平均值，即gray = (R+G+B)/3。
* 加权法：根据重要性及其他指标给R、G、B赋予不同的权值，将灰度图像的灰度值取它们的加权和，W R、WG、WB分别为R、G、B的权值，即gray = WRR+WGG+WBB。

噪声去除的方法一般有：均值滤波、中值滤波、高斯滤波。

1. 银行卡图像卡号定位模块

精确定位银行卡图像中卡号所在的位置，并将卡号部分截取下来，作为接下来的卡号切割模块的输入。

定位卡号需要进行边缘提取，微分算子是最基本的边缘检测方法，主要有异界微分算子和二阶微分算子，其原理为根据图像边缘处的一阶导数取得极值的像素点或者是根据二阶导数过零点的像素点来判断是否为边缘点，其中一阶微分算子为最常用的微分算子。

1. 银行卡卡号切割模块

将卡号图像切割成一个个的数字字符图像，每个图像中只含有一个数字。

由于常用的银行卡卡号有黑色印刷与凹凸压印两种印刷方式，不同的银行卡类型需要使用不同的图像处理方法才能达到理想的分割效果，该模块流程为：

* 首先统计银行卡图像黑色像素点的占比，设置阈值为15%，判定银行卡图像的类别。
* 对凹凸银行卡，采用YUV颜色空间转换法进行背景干扰消除和字符分割。
* 对于褐色印刷银行卡，采用大津阈值法，对于有深色背景干扰的黑色印刷银行卡，采用黑色像素点颜色转换法进行背景干扰消除和字符分割。

1. 银行卡卡号字符识别模块

将切割后的一系列数字图像输入深度学习模型中，并将识别后的结果返回到图形化界面。该模块分别采用传统的提取特征与支持向量机结合的方法、基于卷积神经网络的方法和基于深信度网络的方法进行字符识别，并对比不同方法的识别效果。

## **4.2接口设计**

1. 外部接口

[图形化用户界面](#_Toc4378)接口，在界面设计上，应该做到简单明了，易于操作。

1. 内部接口

[图像预处理模块：程序从前端界面接收用户的银行卡图像后，将其url传给cut文件夹下的cutting.NumRes()接口，由cutting.NumRes()进行图像预处理。](#_Toc31855) [银行卡号定位模块：图像预处理和卡号定位使用同一接口，通过卡号定位模块可定位卡号在银行卡的大体位置，为下面的卡号切割做好准备。](#_Toc10641)

[银行卡号切割模块：通过滑动窗口的方法将一串银行卡号字符切割为单个字符，切割的单个字符作为银行卡号字符识别模块的输入。](#_Toc17075)

[银行卡号字符识别模块](#_Toc3047) ：其接收行卡号切割模块的单个字符作为输入，通过训练好的循环神经网络自动识别出字符对应的数字。

保证各个模块的高内聚低耦合，并且减少每个模块接口的复杂性，保证每个模块都可以随时替换。

## **4.3运行设计**

1. 各个模块应该能够独立测试运行。
2. 各个模块在软件运行过程中能较好的交换信息、处理数据。
3. 运行控制要严格按照各模块间函数调用关系来实现。
4. 要求系统的运行事件基本可以达到用户需求（小于等于5s）。

## **4.4出错处理设计**

当用户没有选择银行卡图片直接点击识别按钮时，提心用户选择银行卡图片。

# **5 详细设计**

## **5.1软件结构概述**

常见的银行卡号有两种印刷方式，一种为平面印刷字符，另一种为凹凸压印字符。其中平面印刷字符的字体一般为Ｔｉｍｅｓ和Ａｒｉａｌ，凹凸压印字符的字体一般为Ｆａｒｒｉｎｇｔｏｎ。平面印刷字符与背景在同一平面，字体为黑色，一般与背景颜色相差较大，背景的花纹对字符造成的干扰较小，在没有大面积深色背景的情况下，背景对字符分割造成的影响较小。但是，当银行卡背景有较复杂深色花纹或者有大面积深色背景的时候，背景会对字符的处理造成很大干扰。凹凸压印字符有银色，金色，直接压印保留背景色等情况，其中保留背景色的凹凸字符存在背景花纹以及颜色干扰，并且字符边框不显著。凹凸压印字符是通过光影的变化来识别的，光线的强弱和光源方向等对字符阴影有影响的自然条件对字符图像的预处理及后期识别都有较大的影响。对于光线情况较差的凹凸银行卡图像，在处理过程中容易出现灰度图色度不均匀，字符边缘断裂，字符同背景灰度差别不大等情况［１］。

银行卡号识别是将获取的银行卡卡号图像，先进行卡号定位，获取卡号的精确位置。然后进行字符分割，将分割出的单个数字字符图像送入机器学习模型中进行训练识别，主要过程及流程如图1所示：

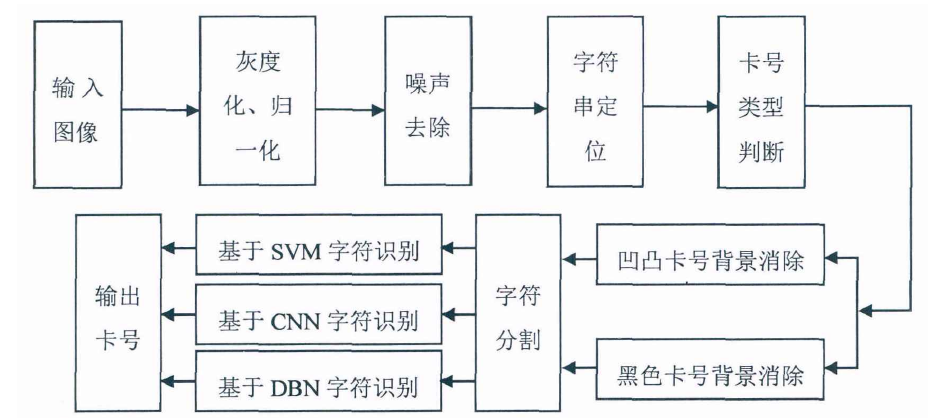


图5

## **5.2获取银行卡号图像**

获取银行卡图像是银行卡号识别的第一步，由于银行卡号涉及到个人隐私和持卡人财产安全问题，获取大量的银行卡号图像具有一定的困难。本文采用模拟银行卡图像进行识别训练。由于近年来定制银行卡的逐渐增多，银行卡背景越来越多样化，本文的模拟银行卡图像采用的背景是从网络多种类型图片里截取而来，共计３７４幅背景图。

## **5.3图像预处理**

图像预处理是对所得的银行卡图像进行分割前的相应处理工作，首先进行银行卡号图像的灰度化，并通过一系列的处理进行噪声去除和图像增强，最后进行数字字符串图像的精确定位，分割出感兴趣区域。在获取银行卡卡号时，由于拍摄环境所处光线以及拍摄角度等都会对后期卡号的识别产生较大的影响。例如光线照射不均匀，某一部分过亮或者过暗都会对银行卡卡号识别的准确率产生影响，所以对于获取到的数字图像必须进行图像预处理，尽量减低各种外界因素对识别结果的影响。本章主要介绍在银行卡图像分割预处理的主要步骤:首先对银行卡图像定位;然后进行背景花纹消除判断:最后得到卡号分割的字符边缘线图像。

在获得银行卡图像后不能直接对整张的银行卡图像进行分割和识别，需要定位出银行卡卡号字符所在的具体区域:然后再根据定位后的银行卡图像进行是否需要进行背景花纹消除的判断;最后得到进行字符分割的银行卡图像。预处理流程如图预处理流程如图所示：

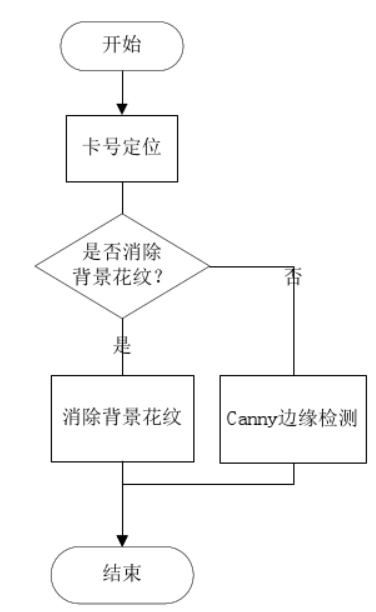
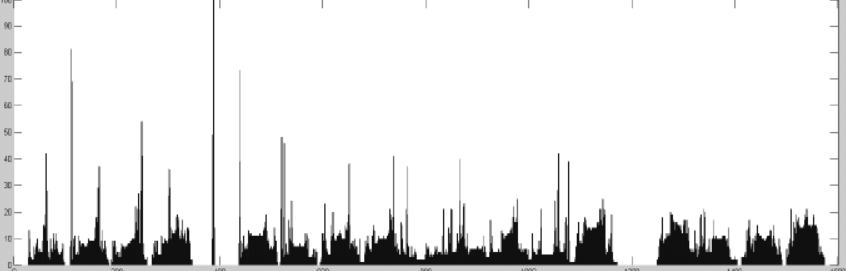


图6

## **5.4卡号定位**

在得到银行卡图像的边缘检测图像后，为了获得图像中像素点分布的情况需要对边缘图像进行投影分析。首先对边缘图像进行水平和垂直方向上的投影，卡号字符所在区域的像素点分布会较为集中，而作为银行卡背景的边框信息处的像素点较少。根据这一特点根据图像中的像素点设定一个阈值，利用这个阈值进行切割得到精确的字符区域。对图2.9进行投影得到的投影图如图所示。



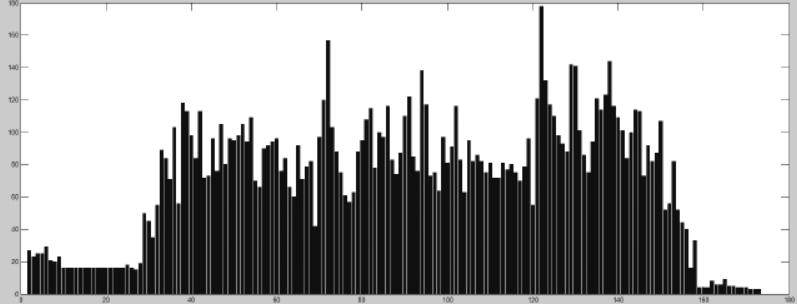


图7

由图可知，在水平和垂直投影方向上的像素统计图中中间大部分的像素点较多，而两边边缘的像素点相对较少有些甚至为0。像素点少的两边即为卡号图像中多于的上下边界和左右边界。因此根据图像中的像素点统计值设定-定的阈值，标志小于阈值的两个边界位置。分别对图像的行列两端进行扫描获得四个标志点。这四个点分别是需要进行切割的像素点的位置。本文采取的水平方向的阈值TH,和垂直方向的阈值TH。

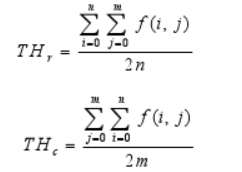


图8

其中f(ij)为在(i.j)处边缘图像的像素值，m表示边缘图像的长度, n表示边缘图像的宽度。对于水平方向的投影图像中从0开始扫描，当得到第一一个大于THr值的点时标志位置然后从最后位置开始扫描得到第-一个大于THr值的位置,得到一个上下界的范围值。同理在垂直方向上进行相同的操作得到左右界的范围值。根据获得的两个范围值对图像进行剪裁得到银行卡卡号字符所在的精确位置。

还可以采用预训练的CTPN模型，自己手动标注银行卡数据数据，如图：



CTPN，全称是“Detecting Text in Natural Image with Connectionist Text Proposal Network”（基于连接预选框网络的文本检测）。该模型主要是对图片中的文本行进行准确定位，其基本做法是直接在卷积获得的feature map（特征图）上生成的一系列适当尺寸的文本proposals（预选框）进行文本行的检测。CTPN模型利用了RNN和CNN的无缝结合来提高检测精度。其中，CNN用来提取深度特征，RNN用于序列的特征识别，二者无缝结合，在检测上性能更好。其中：

（1）CNN（采用VGG16）

CTPN模型通过利用VGG16卷积输出的 feature map（特征图）生成一系列proposals（预选框）进行检测。VGG是卷积神经网络的经典模型，具体原理可详见本公众号之前发布的文章：白话卷积神经网络（VGGNet）

（2）RNN

由于文字信息是由“字符、字符的一部分、多字符”组成的一个序列，所以文字这个检测目标并不是独立、封闭的，而是有前后关联的，因此在CTPN中采用了RNN（Recurrent Neural Networks，循环神经网络）来利用前后文的信息进行文本位置的预测。

整个过程主要分为六个步骤：

第一步：输入3×600(h)×900(w)的图像，使用VGG16进行特征的提取，得到conv5\_3（VGG第5个block的第三个卷积层）的特征作为feature map，大小为512×38×57；

第二步：在这个feature map上做滑窗，窗口大小是3×3，即512×38×57变为4608×38×57（512按3×3卷积展开）；

第三步：将每一行的所有窗口对应的特征输入到RNN（BLSTM，双向LSTM）中，每个LSTM层是128个隐层，即57×38×4608变为57×38×128，Reverse-LSTM同样得到的是57×38×128，合并后最终得到结果为 256×38×57；

第四步：将RNN的结果输入到FC层（全连接层），FC层是一个256×512的矩阵参数，得到512×38×57的结果；

第五步：FC层特征输入到三个分类或者回归层中。第一个2k vertical coordinate和第三个k side-refinement是用来回归k个anchor的位置信息（可以简单理解为是要确定字符位置的小的矩形框，上面示意图中的红色小长框，宽度固定，默认为16），第二个2k scores 表示的是k个anchor的类别信息（是字符或不是字符）；

第六步：使用文本构造的算法，将得到的细长的矩形框，将其合并成文本的序列框。其中文本构造算法的主要的思路为：每两个相近的候选区组成一个pair，合并不同的pair直到无法再合并为止。

## **5.5字符分割**

字符分割是将感兴趣区域里的银行卡号图像切割成独立且互不影响的单个数字字符图像。字符分割的准确度直接影响到字符识别的准确率，字符分割不准确会引起后期的字符识别的误差。本文通过消除感兴趣区域的背景干扰，生成数字字符的像素垂直投影图，根据投影图特点进行字符分割。字符的像素垂直投影图如图9所示：

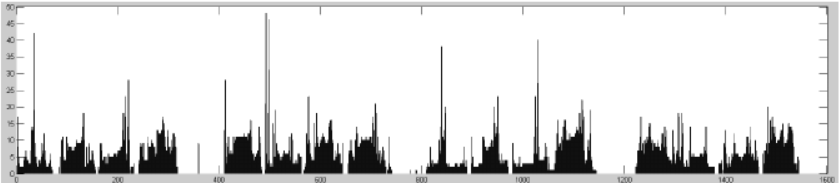


图9

## **5.6字符识别**

识别分割出的数字字符是银行卡号识别系统的最后一步。目前应用于字符识别的方法主要有基于模板匹配的字符识别，提取特征运用ＳＶＭ分类器的字符识别，基于神经网络的字符识别等。由于银行卡号字体不是固定字体，所以模板匹配的方法不能对所有类型的银行卡均达到较高的识别率。提取特征并运用ＳＶＭ分类器对字符进行识别是最常用的字符识别方法。基于深度学习的字符识别是近些年新兴起的识别方法，具有识别率高，抗干扰能力强等特点。本文采用传统的提取特征运用ＳＶＭ分类器的方法，基于卷积祌经网络的方法和基于深信度网络的方法，通过对比三种方法，分析各种方法的识别效果。

CTPN模型最大的亮点是引入RNN来进行检测。先用CNN得到深度特征，然后用固定宽度的anchor（固定宽度的，细长的矩形框）来检测文本区域，将同一行anchor对应的特征串成序列，然后输入到RNN当中，再用全连接层来做分类或回归，最后将小的候选框进行合并，从而得到了文本所在的完整区域。这种把RNN和CNN无缝结合的方法有效地提高了检测精度。

# **6程序描述**

本项目程序参考车牌识别以及其它字符识别等相关识别技术，在图像处理的理论基础上，对银行卡图像进行预处理，字符分割，字符识别等操作。针对银行卡卡号字符的特点，设计了一种根据银行卡类别，自动选择合适预处理及分割方法的流程。对于有大面积背景干扰的黑色印刷体银行卡字符，给出了一种基于字符边缘消除背景花纹干扰的方法。对于有背景干扰的凹凸银行卡，优化了文献［１］中基于ＹＵＶ色彩空间利用色差来消除背景花纹的方法。总体给出了一个适用于各种类型银行卡预处理及分割的处理流程，并达到了理想效果。建立了模拟银行卡卡号样本图像的数据库用于机器学习的识别训练。对于分割完成的数字字符，分别使用传统机器学习方法和深度学习进行识别，并对比分析不同方法的识别效果。

## **6.1程序功能和性能描述**

### **6.1.1图像预处理。**

由于机器学习需要大量的图像进行训练，而银行卡图像由于涉及到银行卡持有人的个人隐私，无法进行大量的图像采集，本文通过计算机模拟银行卡卡号图像，通过在不同的背景图上添加银行卡号数字字符串，实现银行卡号图像的模拟。图像预处理是字符分割前必不可少的准备工作，包括将获得的大小不一的银行卡卡号图片进行归一化、滤波去噪、灰度化、字符串精确定位等操作，此外还通过检测黑色像素点进行了银行卡类别的判定。

### **6.1.2银行卡卡号的字符分割**

设计了一种能对各种类型银行卡都能取得良好分割效果的处理流程。针对凹凸压印的银行卡，在参考文献［１］所使用的方法的基础上，优化字符垂直投影方法，提高分割精确度。针对黑色印刷的银行卡，采用大津阈值法提取字符信息，对于有大面积深色干扰的黑色银行卡，采用本文给出的一种转换图像中黑色像素点，利用转换后图像产生的双层字符边缘保留字符信息，进而减除背景干扰的方法，完成字符的分割。

### **6.1.3字符识别**

本文分别采用了传统的提取特征运用ＳＶＭ分类器，卷积神经网络，深信度网络三种字符识别方法分别对银行卡数字字符进行识别。对本文采用的三种识别方法的识别结果进行对比分析，针对数据库中的无干扰银行卡图像样本和带有磨损的银行卡图像样本分别进行识别效果分析。

## **6.2程序输入输出项目**

用户选择一张银行卡图片作为输入，程序返回该银行卡的卡号，即程序的输入是图片，输出是数字字符串。

# **7程序算法**

## **7.1图像灰度化**

初始获得的银行卡彩色图像是 RGB 模型的，含有 R、G、B 三个图像分量，在这三个分量之前乘上不同的加权系数就可的到相应灰度值的灰度图像。其转换公式 如(2-1) 所示(其中gray表示转换后的灰度图像)：



其中 fr、fg、fb 分别表示 R 分量、G 分量、B 分量的权值。权值不同对于得到的 灰度化图像也不同。如今运用最广泛、效果最好的的权值分别为 0.299、0.587、

0.114，其灰度化公式如(2-2)所示：



如图10为银行卡的彩色图像灰度化后的灰度图像：



图10

## **7.2图像增强**

由于光线照射等环境因素的干扰可能会对彩色图像灰度化后所得到的灰度图像的清晰度产生影响，导致后期字符分割以及识别效率不高。这时对灰度图像进行简单的灰度变换，提高细节的敏锐度，改善图像质量方便下面的字符相关处理。

基本的灰度变换有图像反转、对数变换、幂次变换和分段线性变换函数等。 其中图像反转其表达式 S=L-1-r，其中 r，s 分别为原图和变换后图像的灰度级。图像反转主要适用于增强嵌入于图像暗色区域的白色或灰色细节，尤其当黑色面积处于主导地位。由于银行卡卡号图像的背景花纹存在着多种多样的颜色区域，对于图像反转效果不好，不利于下一步的研究。

1对数变换

对数变换其表达式为：s = c log(1+r)，其中 c 为常量。对数变换使得窄带底灰度 输入图像映射为宽带输出值，可以利用对数变换的这一特性来扩展被压缩的高值图 像中的暗像素。对于对数变换在常量 c 值为 1 时变换后的效果如图 11 所示：



图11

2幂次变换

幂次变换的基本形式为：s=cr，其中 c 和 γ 为正常数。使用幂次变换的过程也叫做伽马校正，其主要用于在计算机屏幕上显示图像时，进行的图像矫正使得显示的图像色彩更加精准。在 c=1，γ=1.5 的情况下对灰度图像所做的灰度变换效果如图 12 所示：

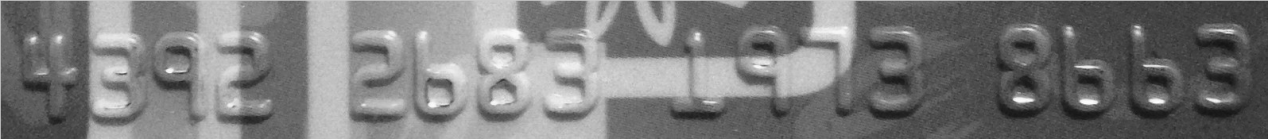


图12

3分段线性变换

分段线性变换函数的基本形式如下所示：

其中(r1,s1)和(r2,s2)的值控制了变换函数的形状，当 r1=s1 且 r2=s2 时，分段线性变换函数变为线性函数，产生的是一个没有变化的灰度图像。对于银行卡灰度图线性拉伸的效果图如图 2.5 所示：



图13

由以上三种灰度变换方法的实验分析可以看出，线性拉伸后的对图像的增强效果最好，凹凸字符的图像边缘信息的对比度增强提高了细节的敏锐度。

## **7.3边缘检测**

由于凹凸银行卡的卡号字符的颜色与卡面的背景花纹差异较小，因此不能通过色差的不同提取卡号的字符信息。由于凹凸银行卡的字符识别主要是通过亮度级的变换识别的，而图像的边缘的亮度值具有不连续性，因此可以通过字符边缘的检测来判断字符的分割点以及字符特征形状并识别。

如今边缘检测算法大部分都是基于图像上的某个像素在其某个邻域内灰度级变化的考察，利用边缘邻近的一阶或二阶方向导数变化规律来检测边缘。通常一阶导数可以用于检测图像中的一个点是否为边缘点，而二阶导数可以判断一个边缘像素点是在边缘暗的一方还是较为明亮的一方。

梯度算子即一阶导数算子，设连续的二维图像用 f(x,y)表示，则 f(x,y)在位置(x,y)处的梯度定义如(2-4)所示：

∇

其中 Gx和 Gy分别表示图像在 x、y 方向的梯度。由上式得知梯度向量指向在坐 标(x,y)的 f 的最大变化率方向。这个向量的大小用▽f 表示，其表达式如下所示：

∇

梯度向量的方向表示为 α(x,y)，由向量分析得到：



典型的边缘检测算子有 Robert 算子、Prewitt 算子 、Sobel 算子 和 Canny算子 等等。JOHN CANNY 在 1986 年提出了 canny 算子，并给出了评价边缘检测 性能优劣的三个指标。

Canny 算子：

Canny 算子性能较好且抗噪能力强，它先对图像进行平滑处理然后再求导，是一种基于多级边缘的检测。Canny 算法包含许多可以调整的参数，它们会影响到算法的计算时间和实效。Canny 算法的第一步是用一个二维高斯函数同原始图像做卷积运算来达到过滤噪声的目的。对于选取的高斯函数曲线 σ 的选取对检测到的边缘有较大的影响，较小的 σ 滤波器适于检测较小、变化明显的细线。较大的σ滤波器使得模糊效果更加明显可以去除较多的噪声信息，这种方法比较适用于检测较大、平滑的边缘。根据检测对象的特点选取合理的 σ 值。

Canny 算法的步骤

(1) 降噪

将原始图像数据与高斯平滑模板作卷积，得到的图像较之原始图像有些轻微的模糊。但是单独的一个像素噪声在经过高斯平滑的图像上几乎没有影响。其运算公式如下：



(2) 微分运算梯度

沿图像的水平 x 与垂直 y 方向利用微分算子分别计算偏导数(Gx, Gy)，运算得到的梯度幅值以及梯度方向如下：



(3) 非极大值抑制

由 1)、2)得到的全局梯度往往存在着许多非边缘的点检测出来，需要进行非极大值抑制，将那些伪边缘点消除。非极大值就是指将该像素点的梯度值与其梯度方向上前后两个像素梯度值进行比较，如果该点的梯度值不是最大，则将该点像素置0，即该点为非极大值。不断重复上述操作，即可得到宽度为单像素点初步的边缘。

(4) 双阈值检测

由上述所得到的图像中可能还存在在噪声。通过设置两个阈值 TH1 和 TH2(TH1<TH2),根据这两个阈值过滤所得到的图像即若当点像素点梯度值小于所

选阈值则像素点置为 0。最后分别得到两个图像 pic1 和 pic2，TH1 检测到的图像 pic1保留了相对完整的图像边缘，但也保留了一定的噪声干扰。而阈值 TH2 检测得到的图像 pic2 过滤掉了大部分噪声，但与此同时也使得到的边缘信息不完整。最后根据pic1 图像的边缘信息来补充连接 pic2 中所缺少的边缘信息。

Canny 算子使用两个阈值比使用一个阈值更加灵活。对于阈值的选取也相当重要，如果设置的阈值过高，可能会漏掉一些重要的信息；如果阈值过低，噪声信息的干扰会对图像产生较大的影响。

对几种典型的边缘检测算子实验得到的效果图分别如下：

对上一步得到的银行卡灰度图进行 Robert 边缘检测，阈值设定为 0.1 时的效果图如图 14所示图：

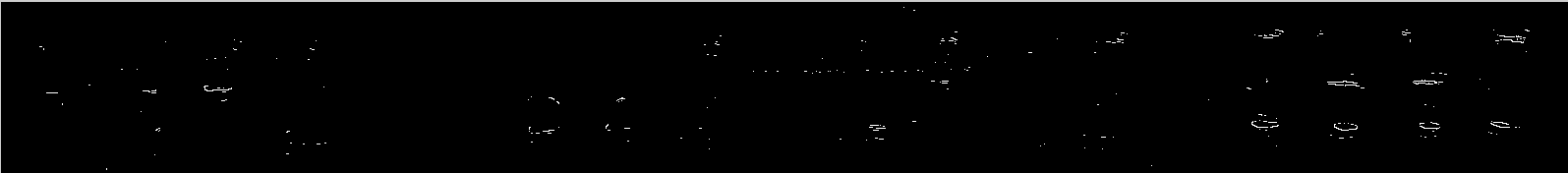


图14

2.对银行卡图像进行 prewitt 边缘检测，阈值设定为 0.1 时的效果如图 2.7 所示：

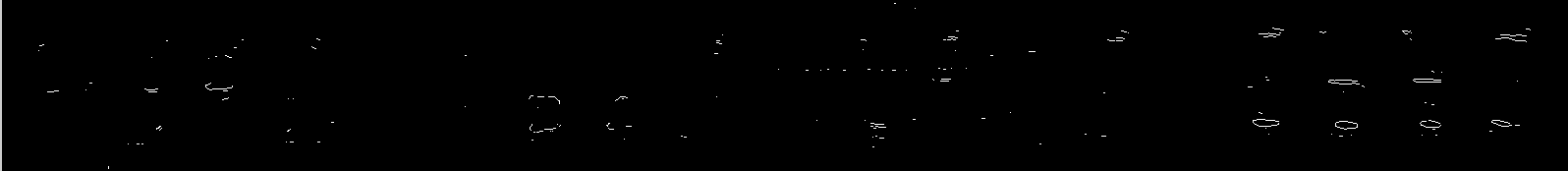


图15

对图像增强后的字符图像进行 sobel 边缘检测，阈值设定为 0.1 时的效果如图2.8 所示：

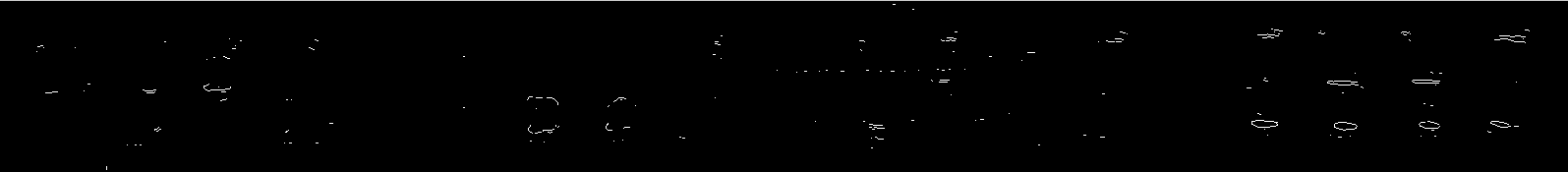


图16

4.对银行卡图像进行 canny 边缘检测，阈值设定为 0.2 时的效果如图 2.9 所示：



图17

边缘检测缺陷及难处

通常对于凹凸字符图像，由于光源照射角度以及字体颜色与背景一致导致图像灰度图中灰度信息跳跃不太明显，并且往往图像中存在着噪声。在边缘检测时必须能准确的检测到凹凸字符的边缘信息并且抑制不相关的细节信息和噪声。

由于边缘检测是根据图像中像素点灰度级的变化来进行检测的，通常是将像素点的灰度值进行微分计算得到梯度值来进行判断。对于图像中的输入信号较为敏感，当图像中的噪声干扰大时对于微分计算的结果变化较大，影响了最终的检测结果。

为了抑制噪声对图像的干扰需对图像进行平滑滤波，但是在平滑时会导致部分信息的丢失，以至于影响到边缘检测的结果。因此对于边缘检测算法没有最为适应的检测算法，要根据所给定图像的具体信息以及检测要求来选择相应的算法。

结合以上效果图以及边缘提取方法原理可得在这几种常用的边缘检测方法中，前面几种算子在阈值设置为 0.1 的情况下所检测到的边缘少之又少，而 canny 算子在阈值为 0.2 的情况下都能得到较好的检测结果，能较好的提取字符边缘信息。

## **7.4RGB与YUV颜色模型转换**

颜色模型间并不是相互独立的，它们可以相互转化。不同的应用场合选择不同的颜色模型。为了获取不同的图像信息可以将 RGB 可向 YUV 模型转换。

1 YUV 彩色模型

YUV 模型应用于 PAL 制系统中，是 PAL 和 SECAM 模拟彩色电视制式采用的颜色空间，是欧洲电视系统所采用的一种颜色编码方法。YUV 模型中”Y”表示的是明亮度，而“U”和“V” 分量表示的则是色度。YUV 彩色模型中它的亮度信号Y 和色度信号 U、V 是分离的。如果只有 Y 信号分量而没有 U、V 分量，那么这样表示的图像就是黑白灰度图像。

2 RGB 彩色模型

RGB 模型是如今最流行以及最常见的彩色模型，该模型主要用于彩色监视器和彩色视频摄像机，生活中用到的彩色扫描仪、手机摄像头、电视摄像机等均是根据 RGB 模型来工作。它主要是通过光的三种颜色来建立的一种模型，分别是 R(红色)、G(绿色)、B(蓝色)。

在 RGB 彩色模型中，任何一张图像都是由 R、G、B 三个图像分量组成，每一个分量都是其原色图像。通过改变这三个图像分量各自的强度比例就可以得到不同的彩色图像。每一个图像分量都是由 8bit 来记录图像，由于每个 RGB 图像都是由这三个分量叠加而成，所以 RGB 每一个像素有 24bit。图 2 是 RGB 的彩色立方示意图，其中三个坐标轴分别表示 R 分量、G 分量、Y 分量，根据不同的 R、G、B值其所显示的颜色也不相同。

3模型转换

在银行卡卡号识别中因为有些卡号字符与背景颜色相同，造成后期识别时会将背景花纹检测出来以至于分辨不出哪些是字符边缘哪些是背景边缘。由于凹凸字符 与背景的花纹颜色一致，而凹凸字主要是靠光线对于凹凸字的不同反射来区分，而凹凸字符在色差上与背景差别较少。利用上述特点可以将获得的 RGB 图像模型转换为 YUV 模型来获取 U、V 分量，再在 U、V 分量的基础上进行边缘检测，这样可得到干扰的背景花纹边缘并消除背景干扰花纹来提高识别率。YUV 模型中白光亮度 Y 分量是由 RGB 模型的各个分量得到，在 RGB 的三个分量前乘上不同的系数即可得到 YUV 的 Y 分量，其中 Y=0.3R+0.59G+0.11B[15]。色差 U、V 是由 R-Y、B-Y 按不同的比例压缩而成。

YUV 与 RGB 的转换公式如下所示：



对于银行卡卡号图像其 RGB 图像如图 18 所示。



图18

将 RGB 转换为 YUV 模型后，其 YUV 图像以及各个分量如图 19-22 所示。

图 19



图 20



图 21

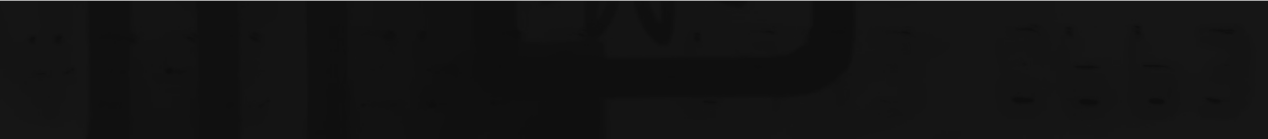


图 22

由图 19-22 可看出对于 U 和 V 分量可明显看出凹凸字符的边缘被弱化，保留了背景花纹。在再 U 和 V 的基础上做边缘检测即可得到干扰的背景花纹边缘。

## **7.5高斯滤波**

在图像采集时，由于光照等环境影响，采集到的图像多伴有各种噪声，在进行进一步处理之前，需要将噪声去除，以免对后续结果造成影响。常用的噪声去除方法有均值滤波，中值滤波，高斯滤波等。

高斯滤波为根据高斯函数的特点来选择权值的线性平滑滤波，高斯滤波对于消除服从正态分布的噪音非常有效。高斯滤波在空间域上是十分有效的低通滤波方法，能很好地保留图像中的低频部分，减少高频部分的作用。常用的高斯滤波模板为:



当用高斯卷积核对图像区域进行卷积时，加权系数与中心像素点的邻域像素点灰度值分别相乘并求和,即可得到邻域中心像素点A(x, y)的新值。如果邻域中所有像素的值都相同，则中心像素点的新值与原值相同。如果邻域中像素点灰度值变化较大，与高斯卷积核卷积之后就会与邻域中其他像素取平均值,这样就达到了消除噪音的目的。相对于均值滤波,高斯滤波的平滑效果更柔和，且能够更好地保留边缘信息。

图2-8中图(1)为未经过去噪处理的边缘图像，图(2)为使用均值滤波后的边缘图像，图(3)为使用中值滤波后的边缘图像，图(4)为使用高斯滤波后的边缘图像。对比图(2)(3)(4)的边缘图像，可见均值滤波不仅损失了比较多的边缘信息，还有少量噪音无法去除，中值滤波和高斯滤波去除噪音效果较好。边缘信息是银行卡分割后续处理的重要信息,较多的边缘信息有助于分割精度的提高，因此该系统选取高斯滤波作为银行卡图像的去噪方法。

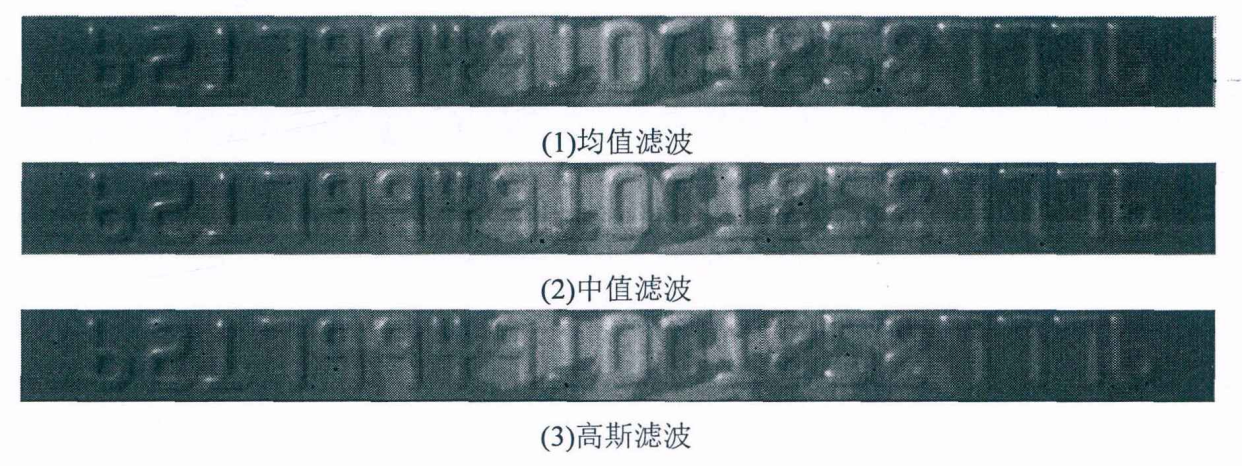


图23

## **7.6大津阈值二值化**

大津阈值(OTSU)又称为最大类间方差阈值，是由日本学者大津(Nobuyuki Otsu)于1979年提出，因此被称为大津算法，是一种自适应的阈值确定方法。大津阈值算法根据输入图像的灰度图,通过灰度特性将图像划分为前景和背景两部分。因为方差可以用来衡量灰度图的灰度分布均匀性，如果前景与背景之间灰度特性的方差越大,说明整幅图像的前景与背景之间的灰度差别越大,意味着能够更好地划分前景与背景，背景错分为前景或者前景错分为背景的概率就越低。大津阈值算法就是通过计算出最佳阈值，从而降低错分的概率。大津阈值的求取原理为:  
 设灰度级为i的像素数为;，该像素点的灰度值为i的概率为;，图像像素总数为N:



则概率，且。

用一个阈值t将图像的灰度值化为两类：。则像素点北划分到和中的概率和期望为：





其中，









则，和两类的方差分别为：



由上面两式可以定义类内方差如下：



定义类间方差如下：



总体方差可表示为：



下列三个公式表示关于t的三个等价判决准则：



因为是常数，是基于一阶的统计特性，是基于二阶的统计特性，是的函数，所以用作为t取到最佳阈值时的最大判决准则：



即为所求的最佳阈值[13，14]。

使用大津阈值处理后的黑色印刷卡号效果如图3-9：



图24

由图可见，大津阈值可以达到相当理想的字符信息提取效果。

## **7.7支持向量机**

提取了字符图像的HOG特征之后，需要通过分类器来判断该图像是哪个数字的图像。本系统选取了支持向量机(Support Vector Machines, SVM)作为分类器。  
  SVM是由美国的贝尔实验室提出的--种基于统计学习理论的模式识别方法，率先应用于美国邮政手写数字库识别研究方面并取得了比较好的效果，随后在文字识别、人脸检测、语音识别等应用领域取得了众多成果。  
  SVM是一种基于结构风险最小原理和统计VC维理论的监督学习方法，最初是针对二分类问题提出的，其原理是寻找一个超平面作为分割界面对两类训练样本进行分类，同时要求分类的错误率最小。对于线性可分问题，必定存在几个超平面可以将训练样本数据完全区分，而在其中寻找一个最优超平面使每--类的训练样本中与超平面距离最近的向量到超平面的距离最大，就是SVM算法的最终目的[22-25]。  
  设有N个n维特征向量(i = 1,2, .,N),特征向量的类别标签为yi，类别标签取1或-1,特征向量中的特征值经过规范化处理，将取值范围设为[0,1]，以降低不同方差特征的不均匀影响。为了最大化间隔，定义两个平行的超平面如公式(4-7)，即为图4-1中两条虚线所示:  
         
式中，w为权向量，b为偏置值。  
      两个超平面之间的距离为2/,若两个超平面之间不存在任何一个特征向量xi，则所有的特征向量必定满足公式(4-8): :  
      

若最小化,即可使两个超平面之间的距离最大化，并且需要同时满足公式(4-8)的条件。使用条件极值求解最优超平面，构造拉格朗日函数如公式所示:

   
式中，ai ≥0是拉格朗日乘子。若要取得|wl|的极值，则需要满足以下偏导公  
式：





由公式（4-10）和（4-11）可以得到新的拉格朗日函数，公式如下：

   
 求得上式的最大值对应的最优解即可求得最优权向量和最优偏置值。每个拉格朗8乘子;均有一个与之对应的特征向量,而当;不为零时其所对应的特征向量即成为支持向量(Support Vector, SV),只有支持向量对最大化两个超平面之间的间隔有作用,其他的特征向量对最终判别函数没有贡献。最终的判别函数见公式:  
          
  对于线性不可分的情况，通过使用核函数( -种非线性映射方法)将低维输入的线性不可分的样本数据映射到高维特征空间使其线性可分。常用的SVM核函数,有: .  
  (1)线性核函数(Linear Kernel) :  
  
       

1. 多项式核函数（Polynomial Kernel）：



1. Sigmoid 核函数（Sigmoid Kernel）:



1. 径向基核函数（Radial Basis Function Kernel）:



1. 高斯核函数（Gaussian Kernel）：



通过核函数将样本数据映射到高维空间之后的拉格朗日函数为:  
   
最终的判别函数也随之变化，如下式:  
 

## **7.8基于卷积神经网络识别字符**

卷积神经网络(Convolutional Neural Network CNN)最早的萌芽起源于1962年神经科学家Hubel和Wiesel等人在研究猫视觉皮层用于局部敏感和方向选择的神经元时提出的Hubcl-Wiesel结构和感受野的概念。1980 年，日本学者Fukushima在感受野的概念基础上提出了可对视觉系统中央凹进行建模的神经认知机模型，它可以看作是卷积神经网络的第-一个实现网络。1998年，加拿大多伦多大学的教授LeCun等人使用误差梯度算法训练卷积神经网络，这-改进将卷积神经网络的研究推向了高潮。现在，随着计算机科学的进步与计算机硬件性能的提高，CNN已经成为众多科学领域的研究热点之一，特别是在语音识别、车牌识别、手写数字识别、人脸特征点检测等模式识别领域。

卷积神经网络结构图如下图所示:

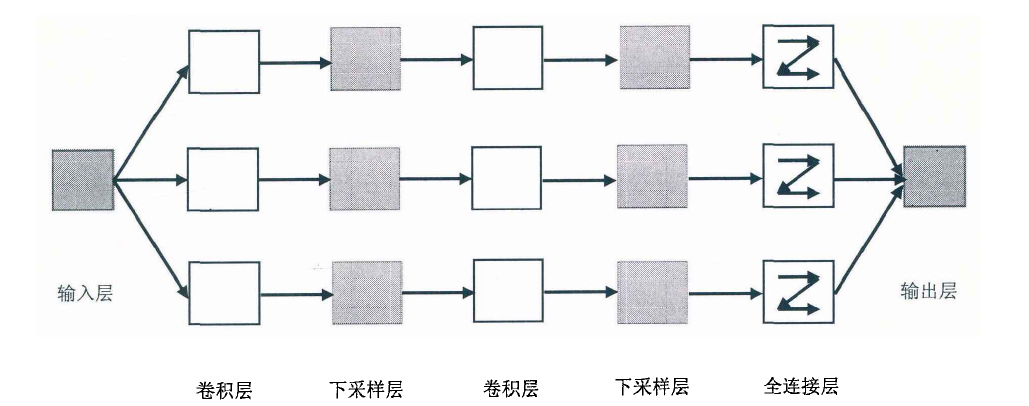


图25

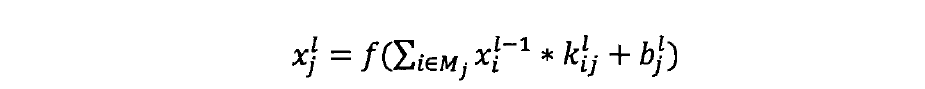
从结构图中可以看到，卷积神经网络的输入层与卷积层相连，输入层为送入网络进行训练和识别的图像数据，网络的中间部分由卷积层和下采样层交替连接而成。每一个结点(也称为神经元)用于提取图像的局部特征，每一个结点都与上一层的局部感受野相连,交替进行卷积和下采样操作,最后将数据送入全连接层得到网络的输出。卷积神经网络是非全连接的网络，卷积层和下采样层交替运算的连接方式可以在出现平移、倾斜、缩放和其它变形时保持高度的不变性。

输出层为数据分类的标签信息,通过有监督的训练方法进行网络收敛计算。卷积神经网络的训练包括前向传播和反向传播两部分，前向传播的卷积层和下采样层的运算过程如下:

(1)卷积层

在卷积层通过卷积运算能够降低信号的噪声以增强图像的特征信息，卷积神经网络的卷积层由多个特征平面构成,而每个特征平面又由多个神经元构成。对于同一个特征平面采用权值共享,全部的神经元具有相同的连接权重，同时特征平面中的神经元都有对应的感受野并且只接受其对应感受野上的信息，同一特征平面

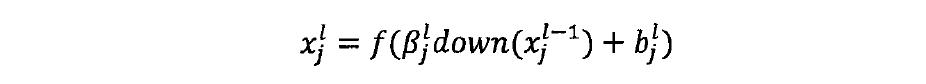
上的感受野大小相同以保证用于共享的权值数量相同。特征图通过卷积运算之后送入激活函数输出结果到下一层，进而构成下一-层的特征图。具体的卷积公式为:



其中，x}表示第l层的第j个特征图，f()表示激活函数，M;表示选择的输入图的集合，b}表示第j个特征图对应的偏置项，ki,表示第l层 的第j个特征图与第i -1层的第i个特征图连接之间的卷积核，“\*” 表示卷积运算。若第i层n1​ ∗ n2特征图与L1 ∗ L2​的卷积核进行卷积运算后，第i+1层即可得到(n1​−L1​+1)∗(n2​−L2​+1)的特征图。

(2)下采样层

下采样层又称为池化层。输入数据经过卷积运算之后数据会映射到高维空间，若只进行卷积操作，不断的向高维空间映射,特征的维数会越来越高最终导致维数灾难，下采样层利用图像局部相关性原理，对图像进行子抽样，降低特征图的分辨率,在减小数据处理量的同时可以保留有效信息，因此卷积神经网络中的卷积层之后一般会紧随着与下采样层相连接。卷积层与下采样层成对出现，这样既完成了特征的多层提取又避免了维数灾难。对于下采样层，输入图像数与输出图像数相同，只是输入图像经过下采样之后尺寸变小，公式表达为:



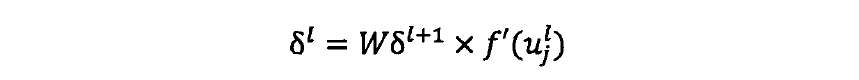
其中，down()表示下采样函数，典型的下采样函数有最大下采样函数和平均下采样函数，分别为对输入图像的不同n x n大小的区域内求最大值和平均值作为下采样函数的输出。β表示第l层的第j个特征图对应的系数, b为其对应的偏置项。

卷积神经网络的训练同经典神经网络的训练相同，前向传播训练之后需要采用后向传播算法的梯度下降原则更新參数，其中最关键的步骤同样也是求取代价函数对于每个参数的偏导。而卷积神经网络与传统神经网络有着不同的连接方式，权值个数与连接数的对应方式也不相同，因此不能按照传统神经网络的反向传播算法来计算每层的残差和偏导，这主要是由于卷积层与下采样层的特殊性引起的，因此卷积神经网络的反向传播的重点是计算卷积层和下采样层的梯度。

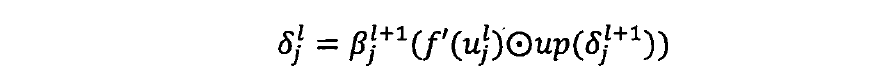
(1)卷积层梯度计算

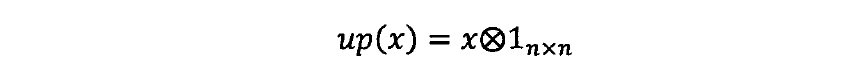
在卷积神经网络中，卷积层l后一般都会与下采样层l + 1相连接，根据传统神经网络的反向传播算法，若要求取第l层的神经元对应权值的梯度，需要先求取第l层的神经节点的误差β。求取误差β的步骤为，先对下一层的节点(与当前第l层的感受野节点相连接的第l+ 1层节点)的误差求和得到βl+1,然后乘以这些连接对应的权值W,再与当前卷积层I神经元节点输入u的激励函数f()的导数相乘，即可求

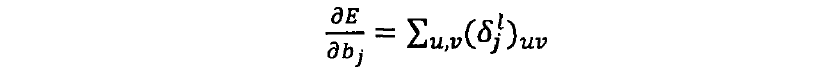
取卷积层l每个神经元节点对应的误差δ。可以用公式表述为:



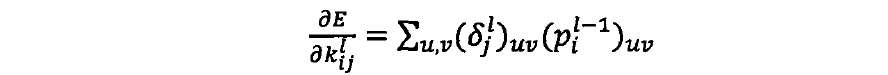
由于采样层的一个神经元节点对应的误差β对应于上一层的输出特征图的一块区域，即下采样窗口的大小，因此卷积层l中的每个特征图的每个节点都只与采.样层l + 1中的一个节点连接。代价函数对每个神经元的偏导即为上述的误差，所以特征图中每个像素都有一个对应的误差,组成特征图对应的误差图。为了有效计算卷积层l的误差，需要上采样和下采样层对应的误差图，将该误差图与卷积层l的特征图的激励值的偏导数逐元相乘。下采样层特征图的权值都取一个相同值，因此将上一步得到的结果与β相乘就完成了第I层的误差计算，用公式表述为:

 其中，⊙为点乘运算，即每个像素点逐个相乘，up()表示- 一个上采样操作,若下采样的采样因子是n的话,该操作即是将其每个像素在水平和垂直方向上复制n次，up()可以用叉乘β运算实现: :

 通过上述公式，可以通过卷积层I中的误差信号进行求和得到该特征图对应偏置项的梯度:



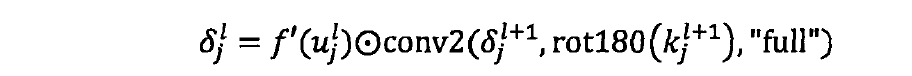
最后对应卷积核的梯度可以用神经网络中的BP反向传播算法进行计算，根据卷积神经网络采用权值共享的特性，任意给定--个权值,则需要对与该权值有关联的所有连接点求梯度，之后对求取的梯度求和:

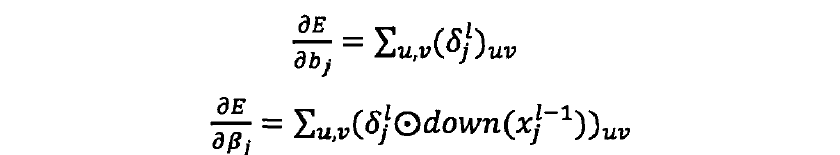
 其中，p为在上述网络进行前向传播计算时，x中在进行卷积运算时与k逐个元素进行相乘的感受野，计算出的卷积特征图在位置(u,v)的值是由上一层位置(u, 0)所在的感受野与卷积核k逐元素相乘运算的结果。

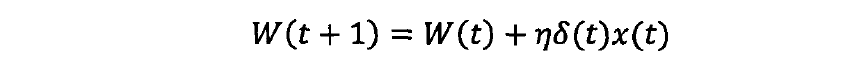
(2)下采样层梯度计算

下采样层的前向传播过程涉及到的参数是每个特征图对应的参数β和一个偏置项b,通过求取此层的误差图即可很容易求出这两个参数的梯度。计算下采样层特征图的误差信号需要先找到当前层的误差图的哪个小块对应于下一层特征图的

哪个像素点，之后才能利用反向传播算法进行计算。通过下-层的误差信号递推得到当前下采样层的误差公式为:

在计算出误差图之后，參数B和偏置项b的梯度计算公式为: :

在计算出前向传播的输入与输出和反向传播的误差项之后，即可在训练是更新权值。在卷积神经网络中由时刻t到时刻t + 1的权值更新可以表示为:

其中n为学习速率，x(t)为神经元的输入，δ(t)为误差项。

## **7.9LeNet-5结构**

LeNet-5模型是具有代表性的卷积神经网络，最初应用于支票上的手写数字识别[38]。本文选用LeNet-5模型进行银行卡卡号字符的识别，其网络结构如图所示:

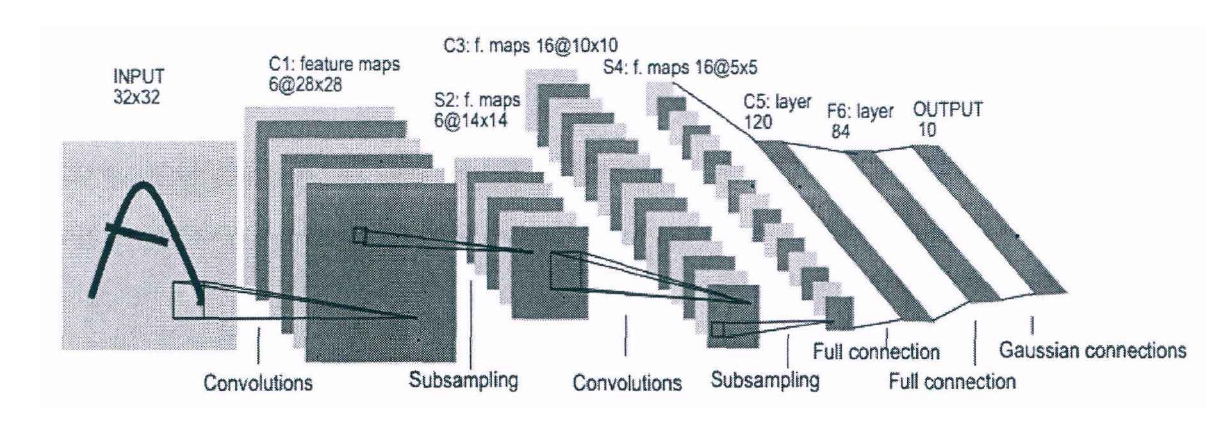


图26

除去输入层，LeNet-5 模型共7层，输入图像为32 x 32的图像。

C1层为第一个卷积层，通过对输入图像进行卷积运算，卷积核大小为5x 5,生成6个特征图，特征图大小为28x28。输入层到C1层的卷积运算共有6个不同的卷积核，每个卷积核包含1个偏置项和5x5 = 25个参数，因此C1层共有(25+1)x6=156个参数。从输入图像提取的特征的不同组合构成本层的特征图，

其组合形式与人眼的视觉系统相似，底层的不同组合构成上层更加抽象化的结构。

S2层为下采样层，基于图像局部相关性的特点，在保留有用信息的同时降低数据量。它对C1层的6个特征图进行尺度为2 x 2的下采样，因此S2层也有6个特征图，每个特征图的尺寸为14X14，共有12个参数需要训练。

C3层是对S2层的6幅特征图进行卷积运算得到的16幅特征图，尺寸为10x10，卷积核大小为5x 5。C3层的每幅特征图均由S2层的某几幅特征图经过卷积得出的，图4-4表示了C3层的特征图与S2层的特征图的连接关系，每个卷积核同时与多个特征图的同-一个位置进行卷积。C3层共有(25+ 1)x16= 1516个参数需要训练。

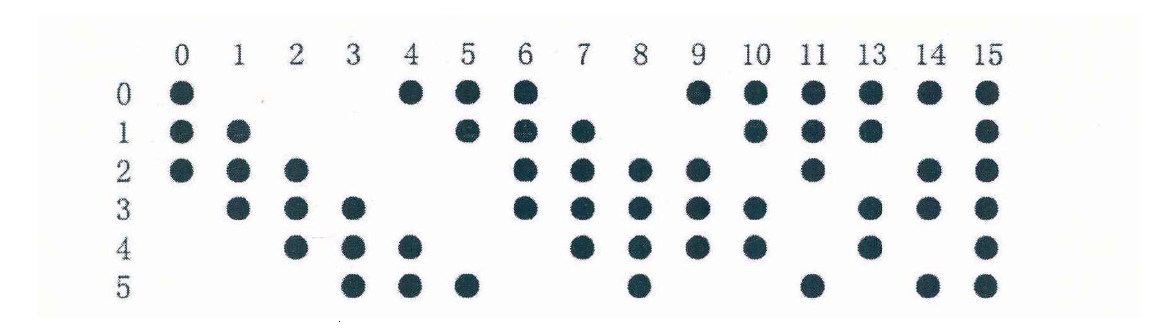


图27

S4层是第二个下采样层，与S2层类似采样尺寸为2x2，有16个特征图，特征图尺寸为5x5，共有32个参数。

C5层为对S4层的进行卷积运算得到的特征图，由于卷积核大小为5x5，与S4层的特征图尺寸大小相同,因此C5层特征图尺寸为1 x 1,特征图即为一个点。C5层的每个特征图都与S4层的所有特征图相连接，这层的参数有(25 + 1) x120 = 3120个。由于输入图像尺寸为32x32,经过上面一系列处理，C5层特征图刚好为1x1，虽然C5层与S4层为全连接，但仍定义C5层为卷积层。

F6层为全连接层，包含84个单元与C5层全连接，一共有(120 + 1)x84=10164个参数。

最后的输出层包含10 个神经元，将输入图像分为10类。由于银行卡卡号字符为0到9，共10类，因此本文选用LeNet-5模型进行卷积神经网络的训练与识别。

## **7.10限制玻尔兹曼机模型**

限制玻尔兹曼机模型是构成深信度网络的基本结构，其可以被视为-个无向图模型，结构如图所示:

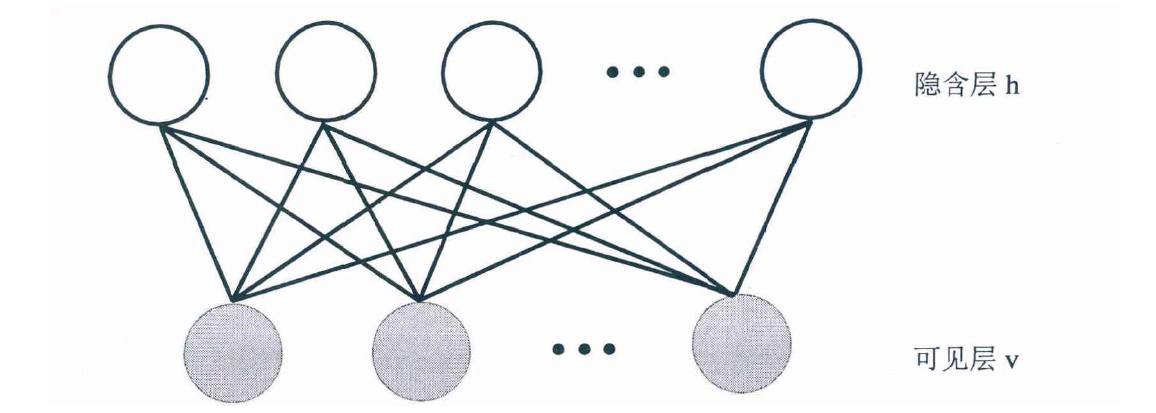
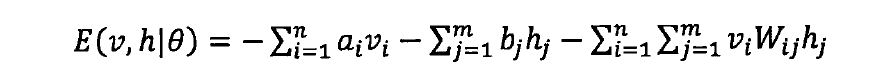


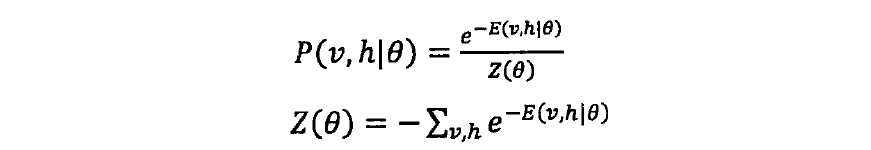
图28

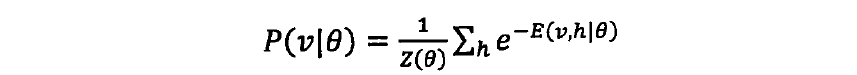
其中h为隐含层，可视为特征提取处理，W为连接矩阵，v为可见层。限制玻尔兹曼机的隐含节点与可见节点可以为任意的指数族节点，如sofmax单元。所有的v和h都设定为二值变量，对任意的i,j, 有0;,h;∈ {0,1}。

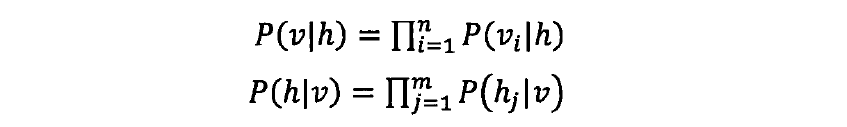
对于一个限制玻尔兹曼机用向量o表示可见节点状态，用向量h表示隐含节点的状态。当在一组给定的状态(o,h)下，限制玻尔兹曼机具备的能量可以被定义为:

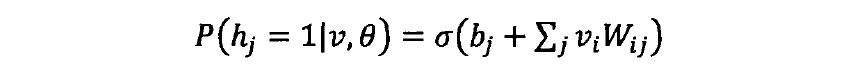


其中n表示可见节点数目，m表示隐含节点数目，θ= {ai, bj, wj}是限制玻尔兹曼机的参数，均为实数。a表示可见节点l的偏置, b;表示隐含节点j的偏置，W:j表示可见层节点与隐含层节点之间的连接矩阵。根据参数和上述能量函数，可以求出(D, h)的联合概率分布，式中Z(θ)为归一化因子:

 接下来需要求取限制玻尔兹曼机的边缘分布P(v|θ)，也称其为似然函数(Likelihood Function):

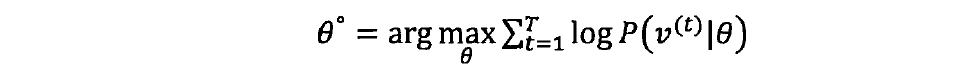
若直接根据上式进行计算，需要非常庞大的计算量才能计算得到Z(0),才能进而得到P(v|0)的分布。由限制玻尔兹曼机层间有连接而层内无连接的特点可知，当某一层的节点状态给定时，另一层节点之间的状态条件分布互相独立，即:

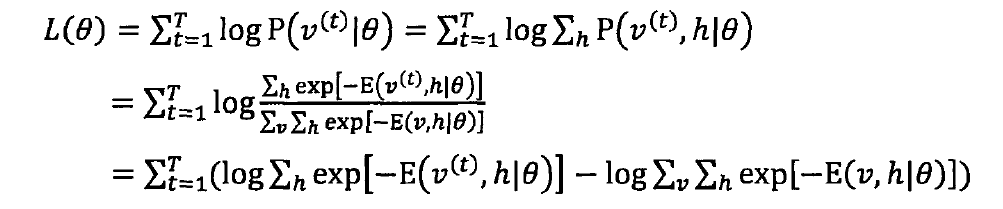
当己知可见节点的状态时，第j个隐含层节点的激活概率为:

式中σ(.)为sigmoid激活函数，σ(x)= 1/(1 + exp(-x))。

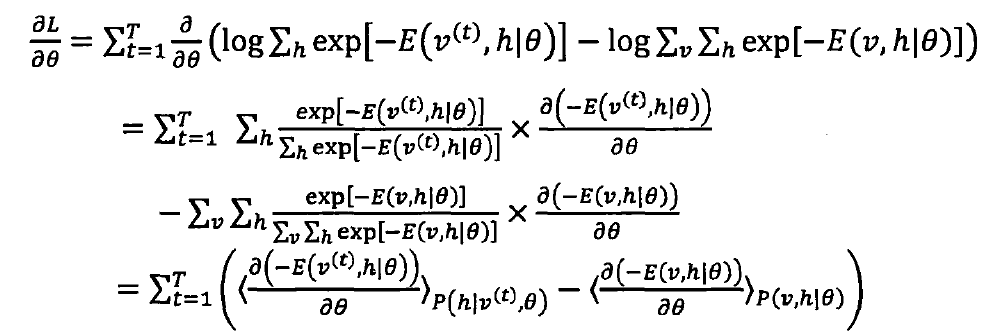
由上式求得所有隐含节点的激活概率之后，由于限制玻尔兹曼机为无方向的全连接对称结构，可见节点的激活概率为:

在训练限制玻尔兹曼机时，其任务就是学习出参数的值，用以拟合数据库中的训练样本。可以利用限制玻尔兹曼机在训练集上的最大化对数似然度来获取参数θ的值，公式如下:

其中T为限制玻尔兹曼机训练集的数据量。为了获取最优参数，一般采用随机梯度上升法，关键步骤为对各个模型参数的偏导数进行计算，由于

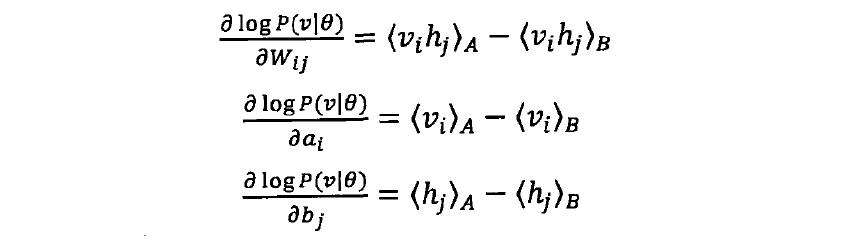


以θ表示ai，bj，W三个参数中的某一个，对数似然函数对于某个参数的梯度为：

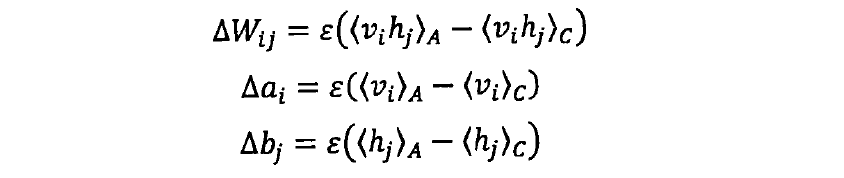


式中P(h|v(t), 8)表示可见层单元的训练样本v^(t)被限定时，隐含层参数的概率分布，<.>p表示求取分布P的均值。P(u, h|0)表示可见层单元与隐含层单元的联合分布，由于归一化因子Z(θ)的计算量非常大，导致公式的后半部分无法计算，只能通过- -些采样方法获取其近似值,一般采用吉布斯采样。在最大化似然函数的过程中，公式(4-42)的偏导数每次进行迭代计算时，只基于部分样本进行迭代以加快计算速度。

假设训练样本数目T= 1时，分别采用A和B来简记概率分布P(h1v(t), )和P(u,h|θ),可以求得对数似然函数1og P(v|θ)关于层间的连接矩阵Wy和两个偏置ai、bj的偏导数如下:

 公式的后半部分由于计算量过大，需要使用吉布斯采样获取近似值。吉布斯采样是-种采样方法，假设X = (X1,X2, .,Xk)是一个K维的随机向量，该随机向量的联合分布P(X)未知,已知其条件概率P(Xp|Xk-)。随机向量X选定随意的一一个状态，可以迭代的使用条件概率P(Xk|Xk-)对Xk-依次进行采样，最终P(X(n))在采样次数足够多的情况下会收敛于X的联合概率分布P(X)，整个采样过程只需;要很短的时间。

虽然可以使用吉布斯采样近似求取对数似然函数，但当观测数据维度较高时，利用吉布斯采样得到学习数据对数似然函数的近似效率仍然不高。为此, Hinton提出了限制玻尔兹曼机的快速学习算法,对比散度算法。对比散度算法通过预训练样本数据，获取1g初始值后只需要进行一到两次吉布斯采样,就能够得到最后的概率近似。给定样本数据D0，根据样本数据计算出所有隐含层节点j的二元状态，之后再根据求出的隐含层节点状态反过来确定可见层节点pq的二值状态，进而生成一个可见层的重构。计算中使用随机梯度上升法最大化对数似然函数在样本数据中的值，其各参数更新的准则如下:



其中，s为学习速率，<.>c表示重构后模型定义的分布。

对于正在训练或者已经训练完成的限制玻尔兹曼机，该限制玻尔兹曼机对于训练数据的对数似然度就是评价其训练结果的标准，可是由于归一化因子Z(0)的存在，无法直接求出对数似然度近似值。本文通过重构误差来评价限制玻尔兹曼机的训练效果。重构误差的计算步骤为:

(1)初始化误差: Error= 0;

(2)进行循环，对于所有的v(),t∈{1,2,....,T}:

计算条件概率分布P(h, v|θ),从条件概率分布中抽取h∈{0,1};

计算条件概率分布P(v',h|θ),从条件概率分布中抽取v'∈{0,1};

Error = Error + |v' – v^(t)|;

(3)返回总误差Error。

重构误差有计算简单,运算开销小等优点,可以对限制玻尔兹曼机的似然度作简单评估。但是使用重构误差来评估其似然度可靠性不够高,在限制玻尔兹曼机模型构建时,无法求得似然度数据或者求取代价比较大时,重构误差在实际运用中依然具有相当的价值。

# **8程序接口**

## **8.1图形化用户界面**

前端界面的HTML代码在CardIdentification/templates/中，其控制代码在CardIdentification/identification/view.py中，通过view.py接收用户上传的银行卡图片，然后调用图像预处理、图像分割、字符识别等模块。

## **8.2图像预处理模块**

程序从前端界面接收用户的银行卡图像后，将其url传给cut文件夹下的cutting.NumRes()接口，由cutting.NumRes()进行图像预处理。图像预处理包括图像灰度化、图像增强、边缘检测、膨胀处理、霍夫直线检测、直线过滤、图像矫正，图像经过预处理后具有较好的识别特性。

## **8.3银行卡号定位模块**

为了简化编程，图像预处理和卡号定位使用同一接口，包括归一化、RBG颜色模型转YUV颜色模型、去除背景图案、异或运算、边缘检测、膨胀处理。通过卡号定位模块可定位卡号在银行卡的大体位置，为下面的卡号切割做好准备。

## **8.4银行卡号切割模块**

银行卡号切割模块在CardIdentification/code/cuts.py中，其接收银行卡号定位模块传递的卡号所在区域图像，通过滑动窗口的方法将一串银行卡号字符切割为单个字符，切割的单个字符作为银行卡号字符识别模块的输入。

## **8.5银行卡号字符识别模块**

银行卡号字符模块在Bank\_Card\_OCR/卡号识别/result.py中，其接收行卡号切割模块的单个字符作为输入，通过训练好的循环神经网络自动识别出字符对应的数字，然后将结果返回到Bank\_Card\_OCR/卡号定位/cutting.py，再由Bank\_Card\_OCR/卡号定位/cutting.py返回到Bank\_Card\_OCR/图形用户界面/view.py，最终由view.py渲染到前端页面。

# **9用户手册**

本系统操作简便，用户只需点击【选择图片】按钮从文件夹中选择一张银行卡图片，然后点击【识别】按钮即可看到该银行卡的卡号信息。

1.进入系统主界面



图29

2.点击【选择图片】按钮选择一张银行卡图片



图30

3.点击【识别】按钮查看该银行卡的卡号信息



图31