

# 中国科学技术大学

# 学士学位论文



题 目 基于机器学习的验证码识别研究

CAPTCHA Recognition using Machine Learning

院 系 信息学院电子工程与信息科学系

姓 名 刘邦 学 号 PB09210365

导 师 张荣 副教授

日 期 2013年6月8日



## 致谢

感谢我的导师张荣副教授。她是我科研的启蒙导师，引导我走好了科研道路的第一步：建立信心。感谢她百忙之中仍不辞辛苦指导我完成毕业论文。从选题到研究算法，再到结题与论文撰写，张老师多次为我指点迷津，开阔研究思路，悉心指导。她那一丝不苟，求真务实的严谨作风感染着我。同时，她不仅在学术上对我进行悉心地指导，也教育我为人处事的道理。

感谢尹龙师兄。在研究过程中，每当我有困难向他请教，他都充满耐心地一一解答。从基础知识的学习到研究思路的形成，再到编码实现算法，尹龙师兄对我的帮助我都铭记于心。同时，他那刻苦钻研、锐意进取、寓教于学的精神也深深感染着我。他对我的鼓励与称赞也让我充满信心。

感谢中国科技大学，感谢我的任课老师们！科大是我成长的地方，是我梦想起航的地方。在这里，我打下了坚实的基础知识，见识了许多充满个性的老师，结交了很多朋友。科大对学生的关怀和其开放自由的政策让我可以自由地成长。

感谢我的家人，你们的爱与期望是我前进的动力。

# 目 录

摘要 .....	3
ABSTRACT .....	4
第 1 章 绪论.....	6
1.1 研究背景.....	6
1.1.1 验证码概念.....	6
1.1.2 验证码的作用.....	6
1.1.3 常见验证码.....	8
1.1.4 验证码识别技术的意义.....	11
1.2 研究现状.....	12
1.3 本文工作.....	12
第 2 章 字符识别理论.....	14
2.1 最小均方误差(LMS)算法 .....	14
2.2 BP 神经网络 .....	15
2.2.1 BP 神经网络的拓扑结构.....	15
2.2.2 前向传播计算.....	16
2.2.3 反向误差传播与权值调整.....	17
2.2.4 基于 BP 神经网络的字符识别.....	20
2.2.5 BP 神经网络小结 .....	20
2.3 卷积神经网络.....	20
2.3.1 卷积神经网络的特性.....	20
2.3.2 卷积计算.....	22
2.3.3 卷积神经网络的拓扑结构.....	23
2.3.4 前向过程.....	24
2.3.5 反向过程.....	25
2.3.6 卷积神经网络小结.....	27
第 3 章 搜狐验证码的识别.....	28
3.1 预处理.....	28
3.1.1 彩色图像灰度化.....	28
3.1.2 灰度图像二值化.....	29
3.1.3 图像去噪.....	30
3.2 字符分割.....	32
3.3 字符识别.....	34
3.3.1 不同算法的识别过程.....	34
3.3.2 识别结果和分析.....	35
第 4 章 验证码识别软件 CAPTCHA Recognizer .....	38
4.1 设计目标.....	38
4.2 系统设计.....	38

4.2.1 需求分析.....	38
4.2.2 结构建模.....	39
4.3 系统实现.....	40
4.3.1 界面设计.....	40
4.3.2 关键技术.....	42
4.4 测试运行.....	44
第5章 结论.....	48
参考文献.....	49

## 摘要

验证码 (CAPTCHA) 是一种区分用户是计算机和人的公共全自动程序。在 CAPTCHA 测试中, 作为服务器的计算机会自动生成一个问题由用户来解答。这个问题可以由计算机生成并评判, 但是必须只有人类才能解答。由于计算机无法解答 CAPTCHA 的问题, 所以回答出问题的用户就可以被认为是人类。对验证码识别技术的研究可以及时发现和改善验证码的漏洞, 增强网络安全性, 并可促进人工智能技术的发展。

本文运用机器学习、模式识别等相关理论, 以搜狐网验证码为研究对象进行识别研究, 提出了针对性的破解方法, 揭示了其不安全的可能性。并通过不同的识别算法的对比, 使研究具有一定的理论和实际价值。主要工作和成果如下:

- (1) 针对已有的搜狐验证码, 提出了一种识别方案。该方案利用最大类间方差 (OSTU) 法二值化, 采用腐蚀、膨胀和连通域去噪的方法对二值化的图像进行去噪, 有效地解决了搜狐验证码模糊粘连和干扰线的问题。并分别采用了最小均方算法 (LMS)、BP 神经网络和卷积神经网络进行字符训练和识别, 识别率分别达到了 56.5%、52% 和 39.5%。
- (2) 设计和开发了一个验证码识别软件 “CAPTCHA Recognizer”, 该软件具有识别单张验证码、批处理识别验证码和图像处理工具箱功能。可以利用普适的图像处理的算法对验证码进行预处理, 以及利用模式识别和机器学习算法对验证码进行识别。同时该软件各模块耦合性低, 易于扩展。

**关键字:** 验证码, 识别, LMS, BP 神经网络, 卷积神经网络, CAPTCHA Recognizer

## ABSTRACT

CAPTCHA is a type of challenge-response test used in computing as an attempt to ensure that the response is generated by a human being. The process usually involves a computer asking a user to complete a simple test which the computer is able to grade. These tests are designed to be easy for a computer to generate but difficult for a computer to solve, but again easy for a human. If a correct solution is received, it can be presumed to have been entered by a human. Research about CAPTCHA recognition technology is useful to find out the shortage of CAPTCHAs being used and enhance the secure of Internet. It can also benefit the development of Artificial Intelligence.

This paper recognizes the CAPTCHA of Sohu website by machine learning and pattern recognition. We developed an algorithm to break Sohu CAPTCHA, revealed that it is not safe using this kind of CAPTCHAs. The research is valuable for theory and practice with the comparison of different algorithms. The mainly work and results are as follows:

- (1) For the existing Sohu CAPTCHA, we propose a recognition scheme. It use OSTU binary threshold to get binary image. Clean images by methods such as erode, dilate and connected domain denoising algorithm. Effectively solved the merged characters and interfering curve in Sohu CAPTCHA. Then BP neural network and simplified convolutional neural network are used to train and recognize single character, and the LMS algorithm is also used to recognize characters. They achieve recognition rate at 56.5%(LMS), 52%(BP) and 39.5% respectively.
- (2) Designed and developed CAPTCHA recognition software “CAPTCHA Recognizer”, which contains functions of single CAPTCHA recognition, batch CAPTCHAs recognition and image processing toolbox. It can preprocess an image using normal image processing algorithms, and then recognize the CAPTCHA by algorithms of pattern recognition or machine learning. What is more, the software modules coupled low and easy to extend.

**Key words:** CAPTCHA, recognition, LMS, BP neural network, convolutional neural network, CAPTCHA Recognizer





# 第 1 章 绪论

## 1.1 研究背景

### 1.1.1 验证码概念

全自动区分计算机和人类的图灵测试（英语：Completely Automated Public Turing test to tell Computers and Humans Apart，简称 CAPTCHA），俗称验证码，是一种区分用户是计算机和人的公共全自动程序。在 CAPTCHA 测试中，作为服务器的计算机会自动生成一个问题由用户来解答。这个问题可以由计算机生成并评判，但是必须只有人类才能解答。由于计算机无法解答 CAPTCHA 的问题，所以回答出问题的用户就可以被认为是人类<sup>[1]</sup>。

验证码来源于卡内基梅隆大学的一个科研项目，Yahoo! 是 CAPTCHA 的第一个用户。CAPTCHA 在网络上的大规模使用起源于 1999-2000 年 Yahoo 网站的账号注册。目前大部分网站都引入了验证码机制来加强网络的安全验证。

### 1.1.2 验证码的作用

验证码作为一种反图灵测试，用于区分用户是人类还是机器人（程序）。它有以下作用：

#### (1) 用于登陆验证

每次发送登陆的请求后，服务器会随机给一个验证码要求用户输入，这样可以防止有人用软件不断的试密码。例如 QQ 用户在登陆时就需要输入验证码。

#### (2) 防止批量注册

很多网站，如 Google、Sohu、QQ 等，提供免费的邮箱服务。而类似的服务普遍受到机器人程序攻击的困扰。这些机器人程序可以每分钟注册上千个账户。使用验证码可以保证只有人才能获得免费账户。另外，随着游戏外挂的泛滥，一些网络游戏中也采用了验证码来保护游戏。一些网络上的交易系统（如订票系统、银行等）为避免被恶意计算机程序以暴力大量尝试交易也会使用验证码机制。

#### (3) 防止垃圾评论与邮件

许多论坛或博客为防止有人利用计算机程序在论坛上张贴广告或发布垃圾信息，在留言提交的位置处都会设置验证码，要求留言者必须输入所给图片的中字符或完成所给算术题等才可以提交留言。通过使用验证码技术，只有人可以在博客里输入评论。

Spammer 通常先在网络上搜寻到以文本形式存储的邮件地址，下一步进行垃圾邮件轰炸。验证码可以提供隐藏人们邮件地址的有效机制。设想一下，如果 spammer 在获得邮件地址之前被要求输入验证码，他们将耗费大量的时间。

验证码为我们创造了一个干净的互联网浏览环境。

#### (4) 保证在线投票的真实性

随着互联网的发展和个人电脑的普及，在线投票变得越来越方便。同时由于网络实行匿名制，人们在网络上也更加愿意表达自己的真实想法。我们需要做的是保证只有人类可以投票，而不是机器。验证码毫无疑问是我们最好的选择。

除了用于网络安全，验证码还有其他新用途：

#### (1) 广告

宇初网络清晰云验证码基于一条与传统的验证码完全不同的安全认证路线，特有的验证码标记技术，使其防破解的核心完全做到了与验证码图像本身无关，因此可以实现要求输入的验证码内容清晰但却更安全。这个新技术的神奇之处不仅在于解决了验证码的用户体验问题和防止广告机的安全问题，而是与此同时更创造了一种全新的网络广告模式。利用验证码做广告实现的传播效果极其惊人，其广告点击率可以比普通的条幅广告高出两个数量级！这足以让中国最大的广告代理公司群邑的互动营销总监惊呼：“这是整个广告业的一次革命！”



图 1.1 具有广告作用的验证码

## (2) 书籍数字化

验证码甚至用于将计算机革命前出版的书籍和报刊数字化,使它们能够被索引、搜索并存储为在线文本。这些老旧的文本通常模糊不清,多达20%的字无法被光学字母识别(OCR)软件读取。人工辨认这些文本将耗资巨大,除非将它们用于安全字谜——卡内基梅隆将之称为“再验证码”(reCaptchas)——互联网用户将免费进行辨认。

卡内基梅隆大学教授路易斯·范·安(Luis von Ahn)表示,在第一年的运作中,被解答的再验证码超过12亿,被辨认出的字超过4.4亿。他表示:“我们证明,我们能够利用人类可能会白白浪费掉的处理能力,并引导其去完成电脑尚无法处理的任务。”

### 1.1.3 常见验证码

验证码生成方式多种多样,验证码的种类也有很多。常见的有以下几种:

#### (1) 字符验证码

这是最为广泛的验证码形式。用户需要输入给出的验证码图片中的字符,字符可能为英文,数字或汉字等。一般这种验证码图片都有很多干扰噪声,字符扭曲、粘连、变形。这种验证码识别难度大,是我们的研究对象。



图 1.2 各大网站邮箱的验证码

## (2) 图片验证码

这种验证码要求用户基于图片来作出正确的操作，而非识别字符。相比于字符型验证码，应用较少。



图 1.3 左边是传统的验证系统，右边的要求用户将图形旋转到正确方向

## (3) 音频验证码

这种验证码的内容不是依靠视觉来判断，而是依靠听力听到内容并输入。这是为了让有视力障碍无法阅读辨认屏幕上验证码的上网用户而开发的更为安全的音频验证程序。



图 1.4 音频验证码

## (4) 数学验证码

这种验证码给出一个算式或含有数学问题的图片，让用户作答。有些难度很大，用户体验极差。而太简单的则很容易被破解。

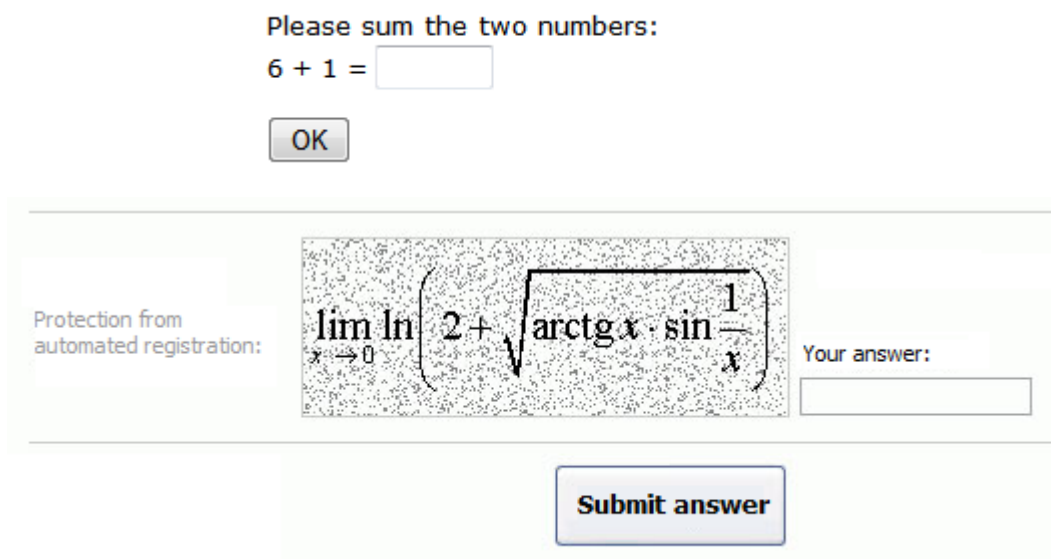


图 1.5 数学验证码

#### (5) 其他验证码

随着验证码技术的发展,各种新型验证码层出不穷。图 1.6 的验证码靠“情感”鉴定人类用户,要完成这个验证过程,用户必须要从中选出一个正确的词汇,而这个词是对上面一段文字描述让一个正常人类产生的情感的对应词汇。图 1.7 的验证码是一种新型验证码输入方式——滑动验证码,用户只需将鼠标滑动到正确的位置即能轻松完成验证过程。这种验证方式的灵感源自苹果手机的滑动解锁功能,它由上方的一张扭曲图片和下方一个横向滑动的滑动块组成,滑动块的移动会造成图片的变化,当用户将滑动块移动到能使图片正常显示的位置时,验证即为成功,否则,用户需要再次验证。这商业运用上也有一定的前景:网站可以将扭曲的图片替换成广告图,用户在验证时因为需要仔细观察图案,停留的时间比较长,所以会大大加深人们对广告内容的印象。可见验证码的作用已不仅局限于验证是否人类了。

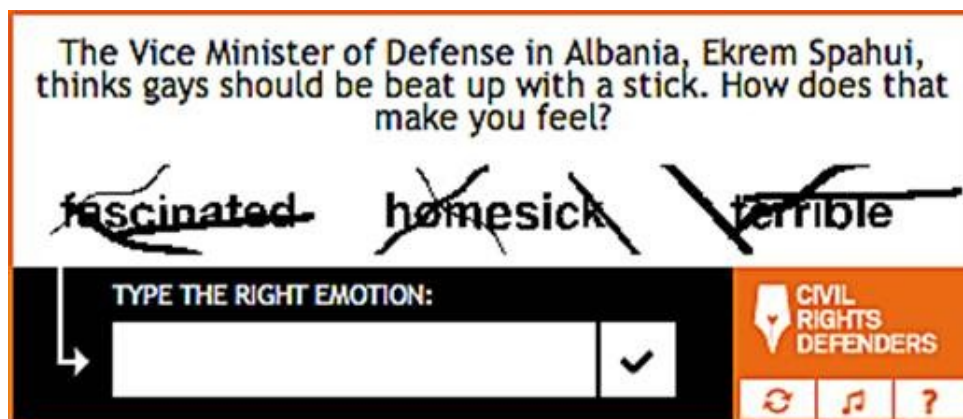


图 1.6 情感验证码



图 1.7 滑动验证码

#### 1.1.4 验证码识别技术的意义

验证码的出现大大加强了网络安全的保障。然而，现有的验证码仍存在很多的不足。对验证码识别技术的研究，有利于发现验证码的不足，以便改善和创造更具效果的验证码。

同时，验证码识别的研究，涉及到图像处理、机器视觉、模式识别、人工智能等多个领域。其研究成果不仅对于 CAPTCHA 识别这项图灵测试本身是一种挑战，更对这些领域中的其他研究有帮助，很多研究成果可以推广到其他的应用领域。例如手写文字识别技术(即 OCR 技术)等。

总之，验证码识别的研究永远是一个双赢的结果：如果一种验证码不能被破解，那么网络安全依然有保障；如果验证码被识别，则可以针对其弱点加以改进，并且人工智能和机器学习水平得到进一步的提高。

## 1.2 研究现状

验证码识别本质上是图像处理与字符识别的问题。现有的成熟的字符识别技术有 OCR 系统、车牌识别等。然而，由于验证码的特殊用途，它的识别难度难于一般的光学字符识别。由于验证码设计的目的本就是让机器难以破解，因此通常的验证码图片干扰性强，字符间有扭曲、变形、粘连，干扰信号变化相对单调，识别较难，图像较小。目前对无扭曲旋转、有少量噪声和无粘连字符的验证码识别上已取得了较好的识别结果。而对于有彩色线噪声，字符粘连、重叠、扭曲以及缩放等干扰的验证码的识别效果尚不理想。对于此种验证码的破解，属于极端情况下的字符识别。

验证码识别的研究是一个较为热门的领域，关于验证码识别的理论体系也日渐趋于完善。2003 年，Mori 和 Malik[2]提出了用形状上下文匹配的方法识别 EZ-Gimpy 和 Gimpy 的验证码，识别率分别为 92%和 33%。2004 年 Hocevar.S [3]在发布了他的验证码识别程序 PWNtcha(Pretend We're Not a Turing Computer but a Human Antagonist 的缩写,即“假设我们不是图灵机，而是人类的对手)，以告知人们验证码不是一种安全的交互验证手段。这个项目现已开源，PWNtcha 针对的主要是背景易分离和非粘连的验证码。同年，Gabriel Moy[4]等提出了一种扭曲估计的算法破解 EZ-Gimpy 的验证码，有很高的识别率。2005 年，Kumar Chellapilla 和 Patrice Y.Simard[5]的研究表明，在验证码识别中，分割是比识别更为困难的问题。一旦将字符单独分开，运用机器学习算法可以轻易的解决识别问题，但现今还没有一种通用有效的方法解决字符分割的问题。2007 年，Per-Ola Kristiansson[6]对验证码提取特征并用神经网络训练的方法对一种简单的验证码进行了识别。2008 年，Jeff Yan[4]等人以高达 90%以上的识别率破解了微软先前的验证码。2009 年，A.A. Chandavale[7]等针对 EZ-Gimpy 的四类验证码分别进行了识别。2011 年，王璐[8]利用神经网络与形状上下文识别了猫扑、西祠胡同和天涯的验证码。

目前看来，所有的验证码识别技术都具有针对性，因为每个网站的验证码生成技术，图像背景、噪声，字符粘连、扭曲方式等都不同，因而无法用统一的方式进行识别。另外，对于背景可分离，字符分开的验证码，其识别比较简单，而背景与字符难以分离，字符严重扭曲和粘连的验证码，识别难度很大。

## 1.3 本文工作

本文以搜狐验证码为研究对象，研究了不同的验证码识别技术。

验证码的识别一般包括图像预处理、字符分割、字符识别这三个步骤。本文对验证码的识别思路是：首先，利用图像处理技术如灰度化、OSTU 二值化、腐蚀膨胀等进行预处理，再利用投影法等进行字符的分割，最后分别利用最小均方误差(LMS)算法，BP 神经网络，卷积神经网络对分割后的字符进行训练与识别。最后，构建了验证码识别软件 CAPTCHA Recognizer，用于演示算法的效果和整合不同的算法。

本文按五章展开。

第一章，阐述了课题的背景及研究意义，介绍了验证码识别在国内外的发展状况，最后大体介绍了本文的主要工作。

第二章，介绍了用于字符识别的最小均方误差(LMS)算法、BP 神经网络和卷积神经网络，给出了它们的推导过程。

第三章，具体针对已有的搜狐验证码进行破解，提出具体的分割和识别方案，给出识别结果。

第四章，介绍验证码识别软件 CAPTCHA Recognizer 的结构与应用。

第五章，总结工作。



## 第 2 章 字符识别理论

验证码字符识别的问题本质上是图像处理和模式识别的问题。本文用到的识别算法包括：最小均方误差(LMS)算法、BP 神经网络和卷积神经网络。LMS 算法是基于模版匹配的方法，它相对简单，但所需模版库较大，算法的鲁棒性好；BP 神经网络对字符提取特征，然后输入网络进行训练，若训练时间足够、样本合适，则在模版数量不变的情况下，可以取得更好的效果，对字形的扭曲、倾斜等形变具有更强的鲁棒性；卷积神经网络更为复杂，相对于 BP 神经网络，它具有可以自适应提取合适的特征的特点，而不需要人工选择特征，相应的代价则是算法复杂度更高，需要的样本数量很多，训练速度更慢，难度更大。

### 2.1 最小均方误差 (LMS) 算法

LMS 算法全称为 Least mean square 算法，中文是最小均方算法，由美国斯坦福大学的 Widrow 和 Hoff[9]在研究自适应理论时提出，由于其容易实现而很快得到了广泛应用。LMS 算法通过计算两个向量间的均方误差来衡量二者的相似性：

$$D(\mathbf{X}, \mathbf{T}) = \sum_i (x_i - t_i)^2 \quad (2.1)$$

将所有的模版向量  $\mathbf{T}$  与提取的向量  $\mathbf{X}$  比较，计算  $D(\mathbf{X}, \mathbf{T})$ ，取  $D(\mathbf{X}, \mathbf{T})$  最小的  $\mathbf{T}$  的类别作为  $\mathbf{X}$  的分类结果。

试验中我们提取的是网格特征：



图 2.1 提取字符网格特征

通过字符的 25 维网格特征向量与模版的网格特征向量对比，计算均方误差，取均方误差最小的类别作为识别结果。

## 2.2 BP 神经网络

神经网络的长期发展目标是自动化的机器智能，然而人工神经网络的现代应用主要是在模式识别领域。对于数据分类这个子领域，神经网络方法的效果可以媲美统计学方法（例如回归分析、概率密度估计等）。

对于图像分类，应用最广泛的神经网络算法是前向-反馈神经网络算法。

### 2.2.1 BP 神经网络的拓扑结构

神经网络是衍生于脑科学的多种计算结构中的一员[10][11]。单层神经网络有严重的缺陷：能够实现的分类任务十分有限。Minsky 和 Papert[12]在 1969 年表明，一个双层结构的前向网络能够克服很多限制，但是没有解决如何调节输入单元到隐藏单元间的权值的问题。Rumelhart, Hinton 和 Williams 在 1986 年给出了这个问题的答案[13]，并且相似的解答在更久以前就已经被发表[14][15][16]。这个解答的核心思想是：隐含层节点的误差由输出层节点误差反向传播来决定，因而这个算法也被称为反向传播学习规则。而应用这种学习规则的前向神经网络就是 BP 神经网络。

BP (Back Propagation)神经网络是使用最为广泛的一种神经网络，全称为基于误差反向传播算法的人工神经网络。它具有分层结构，每层包含多个节点，节点输入来自前一层的输出，而节点的输出则作为后一层的输入，同层节点之间不相连。对于输入层，其输入数据就是输出数据；对于隐含层与输出层，输入数据之和经过激活函数，得到的结果就是该节点的输出。

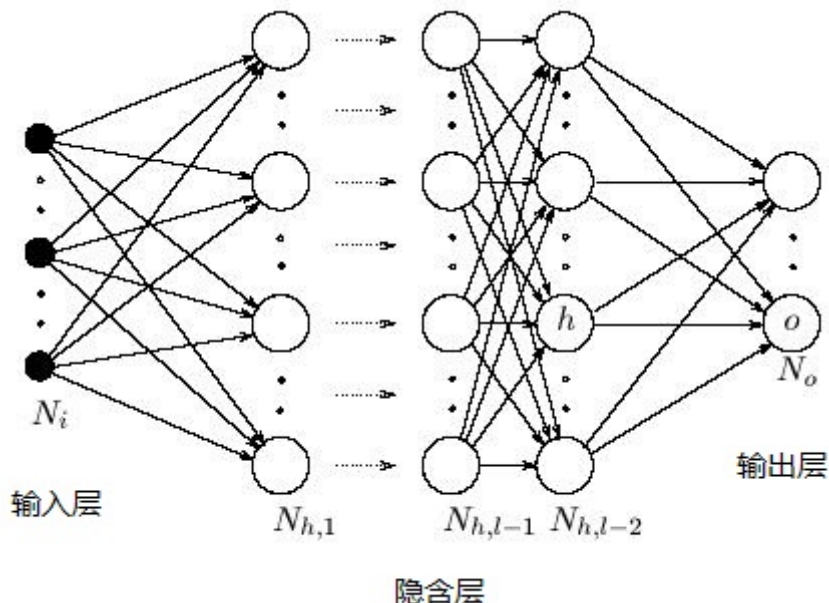


图 2.2 BP 神经网络的拓扑结构

BP 神经网络的学习过程可以分为两步：数据的前向计算和误差的反向传播。通过数据的前向计算，将输出层结果与训练样本标签数据对比，计算误差以衡量现有网络对数据的分类能力；通过误差的反向传播，利用梯度下降算法对网络权值进行调整，使得网络的输出结果与样本相匹配。

### 2.2.2 前向传播计算

BP 神经网络的前向计算过程得到输出层向量，通过与目标向量对比，计算误差，从而判断网络的分类效果。若误差大小小于误差容限，则停止训练；否则进入误差反向传播过程，调节网络权值。

以三层网络结构来推导 BP 神经网络正向传播的计算公式。设  $i$  代表输入层结点， $j$  代表隐含层结点， $k$  则表示输出层神经元。每输入一个样本  $x$ ，有输入层向量  $\bar{X} = (x_1, \dots, x_i, \dots, x_n)^T$ ，隐含层输出向量  $\bar{Y} = (y_1, \dots, y_j, \dots, y_m)^T$ ，输出层输出向量为  $\bar{O} = (o_1, \dots, o_k, \dots, o_r)^T$ ， $T$  代表向量转置。输入层到隐含层之间的权值矩阵为  $V$ ，定义  $V = (\bar{v}_1, \dots, \bar{v}_j, \dots, \bar{v}_m)$ ，其中  $n$  维列向量  $\bar{v}_j$  代表隐含层第  $j$  个单元的权值向量， $\bar{v}_j = (v_{1j}, \dots, v_{ij}, \dots, v_{nj})^T$ ， $v_{ij}$  是输入层第  $i$  个单元与隐含层第  $j$  个单元的权值。隐含层到输出层之间的权值矩阵为  $W$ ，

$W = (\overrightarrow{w_1}, \cdots, \overrightarrow{w_k}, \cdots, \overrightarrow{w_r})$ ,  $m$  维列向量  $\overrightarrow{w_k}$  是输出层第  $k$  个单元对应的权值向量,  $\overrightarrow{w_k} = (w_{1k}, \cdots, w_{jk}, \cdots, w_{mk})^T$ 。  $w_{jk}$  是隐层第  $j$  个神经元到输出层第  $k$  个神经元的权值。节点激励函数采用 *Sigmoid* 函数, 定义为:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.2)$$

它的导数为:

$$f'(x) = f(x)(1 - f(x)) \quad (2.3)$$

则正向传播的各层节点值计算如下:

隐含层结点  $j$  的输出值  $y_j$  有:

$$y_j = f(\text{net}_j), \text{net}_j = \sum_{i=0}^n v_{ij} x_i, j = 1, 2, \cdots, m \quad (2.4)$$

输出层节点  $k$  的节点值  $o_k$  有:

$$\begin{aligned} o_k &= f(\text{net}_k) \\ \text{net}_k &= \sum_{j=0}^r v_{jk} y_j, k = 1, 2, \cdots, r \end{aligned} \quad (2.5)$$

通过上述转换, 完成了向前学习过程, 得到了输出信号。

### 2.2.3 反向误差传播与权值调整

反向传播就是误差信号从输出端传向输入端。在该过程中, 网络的权值由误差反馈进行调节, 在权值向量空间执行梯度下降策略, 通过迭代不断的修改权值向量, 从而使得网络的输出逼近期望输出, 使得网络误差达到最小。权值的修改采用 *Delta* 规则, *Delta* 规则简单讲来就是: 若神经元实际输出比期望输出大, 则减小所有输入为正的连接的权重, 增大所有输入为负的连接权重。反之, 若神经元实际输出比期望输出小, 则增大所有输入为正的连接的权重, 减小所有输入为负的连接权重。这个增大或减小的幅度就根据上面的式子来计算。

网络的误差采用平方型误差函数。设共输入  $P$  个样本, 用  $z_1, \cdots, z_t, \cdots, z_p$  表示。第  $t$  个样本输入网络中网络输出为  $o_{tk}$ , 期望输出为  $d_{tk}$ , 则第  $t$  个样本误差为:

$$E_t = \frac{1}{2} \sum_{k=1}^r (o_{tk} - d_{tk})^2 \quad (2.6)$$

对于  $P$  个样本，全局误差  $E$  为：

$$E = \sum_{t=1}^P E_t \quad (2.7)$$

反向传播根据梯度下降法调整各层权值使得全局误差  $E$  最小。

(1) 对输出层与隐含层的权值矩阵  $W$  的调整。

根据梯度下降法，隐层权值增量  $\Delta w_{jk}$  有：

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} \quad (2.8)$$

$\eta$  是学习率，表示训练速率。综合式(2.7)、(2.6)、(2.5)、(2.3)、(2.2)可推导：

$$\begin{aligned} \Delta w_{jk} &= -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{jk}} \\ &= -\eta \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial w_{jk}} \\ &= -\eta(o_k - d_k) f'(net_k) y_j \\ &= -\eta(o_k - d_k) f(net_k)(1 - f(net_k)) y_j \\ &= -\eta(o_k - d_k) o_k(1 - o_k) y_j \end{aligned} \quad (2.9)$$

记输出层神经元的局部误差信号

$$\delta_k^o = -\frac{\partial E}{\partial net_k} = -(o_k - d_k) o_k(1 - o_k) \quad (2.10)$$

则有：

$$\Delta w_{kj} = \eta \delta_k^o y_j \quad (2.11)$$

调整后的权值矩阵  $W$  中的元素  $w_{jk}$  有：

$$w_{jk} = w_{jk} + \Delta w_{jk} \quad (2.12)$$

(2) 对输入层和隐含层之间的权值矩阵  $V$  的调整

与推导  $\Delta w_{jk}$  类似，对权值增量  $\Delta v_{ij}$  有：

$$\begin{aligned}
\Delta v_{ij} &= -\eta \frac{\partial E}{\partial v_{ij}} = -\eta \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial y_j} \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial v_{ij}} \\
&= -\eta \sum_{k=1}^r \delta_k^o w_{jk} f'(net_j) x_i \\
&= -\eta \sum_{k=1}^r \delta_k^o w_{jk} f(net_j) (1 - f(net_j)) y_j \\
&= -\eta \sum_{k=1}^r \delta_k^o w_{jk} y_j (1 - y_j) x_i
\end{aligned} \tag{2.13}$$

记隐含层神经元的局部误差误差信号

$$\delta_j^y = -\frac{\partial E}{\partial net_j} = -\sum_{k=1}^r \delta_k^o w_{jk} y_j (1 - y_j) \tag{2.14}$$

则有：

$$\Delta v_{ij} = \eta \delta_j^y x_i \tag{2.15}$$

调整后的权值矩阵  $W$  中的元素  $v_{ij}$  有：

$$v_{ij} = v_{ij} + \Delta v_{ij} \tag{2.16}$$

至此完成一次权值矩阵的更新。下一个输入样本仍从输入层开始正向传播，利用已更新后的权值矩阵，达到输出层得到输出信号，计算输出样本的误差。若样本总体误差大于给定的最小误差，继续向后学习，更新权值矩阵，直到样本误差小于给定的最小误差或达到最大迭代次数。

综上所述，BP 网络的学习过程可以归纳如下：

- (1) 设置变量和参数，包括各层权值矩阵，训练样本数目、学习速率  $\eta$  和样本最小给定误差。
- (2) 初始化，给各个权值矩阵一个较小的随机非零向量。
- (3) 输入随机样本  $x$ 。
- (4) 对输入样本  $x$ ，通过前向学习过程计算 BP 网络每层神经元的输入和输出，在输出端得到网络最终的输出信号。
- (5) 根据实际输出和期望输出求得误差。若该误差小于给定的最小误差则转第 (8)，否则转入 (6)。
- (6) 判断是否已经达到最大迭代次数，若已经达到，转 (8)，否则进入网络反向学习过程，反向计算每层神经元的局部误差。
- (7) 根据局部误差修正各个权值矩阵。
- (8) 判断是否学习完所有的样本，是则结束，否则转 (3)。

## 2.2.4 基于 BP 神经网络的字符识别

BP 神经网络的输入输出都在区间 $[0,1]$ 之间。输出层节点数代表了字符分类的类别数。每个节点都赋予了不同的含义,表征了所要识别的字符集。BP 网络的识别过程其实就是前向传播过程。将待测样本输入网络,经隐含层到达输出层,得到输出信号。网络的输出符合最大值检出原则,选择输出节点值最大者输出。例如,将 BP 神经网络应用于数字识别,则输出层 10 个节点代表了 10 个数字。若待测样本输入网络后,输出层第 3 个节点的节点值最大,则输出第 3 个节点代表的数字 3,即是该样本的识别结果。

## 2.2.5 BP 神经网络小结

BP 神经网络是已经很成熟的算法,理论推导严谨。它能学习和存贮大量的输入-输出模式的映射关系,不需要事先知道这种映射关系的数学方程,只需要提供足够多的样本模式;有很强的容错能力,输入样本中带有较大的误差甚至个别错误对网络的输入输出规律影响很小。

然而,将 BP 神经网络应用于字符识别时,需要先对待识别的字符提取特征,然后使用得到的特征向量来训练神经网络的分类器。提取到的特征的好坏直接影响了最终的分类性能。这在一定程度上限制了最后的识别率。

## 2.3 卷积神经网络

在传统的模式识别中,特征提取这一步,一般是手工进行的,在提取了大量特征后,还要对这些特征进行相关性分析,把对分类无关的特征和自相关的特征去掉。但这些特征的提取相当依赖于人的主观意识,提取出的不同的特征对分类性能影响很大;同时,图像预处理对特征提取的影响很大。这样,把特征提取这一过程作为一个自适应、自学习的机器,甚至把它与分类识别装置集成就是很自然的了。卷积神经网络就是具有这样一种特性的神经网络结构。

### 2.3.1 卷积神经网络的特性

1998 年,前 At&T Shannon Lab 的 Yann LeCun 等人提出了基于卷积神经网络 (Convolutional Neural Network) 的一个文字识别系统 LeNet-5[17]。

这个方法不但能够把特征提取和识别结合在一起，综合评价和学习，而且能够自动提取局部特征（线段，边缘，端点，拐角等等），在不断的反向传播学习过程选择、优化这些特征。

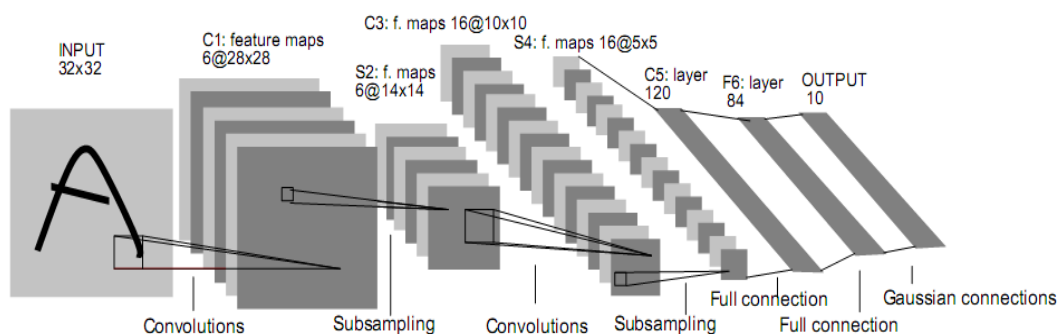


图 2.3 LeNet-5 文字识别系统的结构

由于卷积神经网络结构复杂，参数众多，Patrice Y. Simard, Dave Steinkraus 于 2003 年提出了改进后的卷积神经网络[18]。

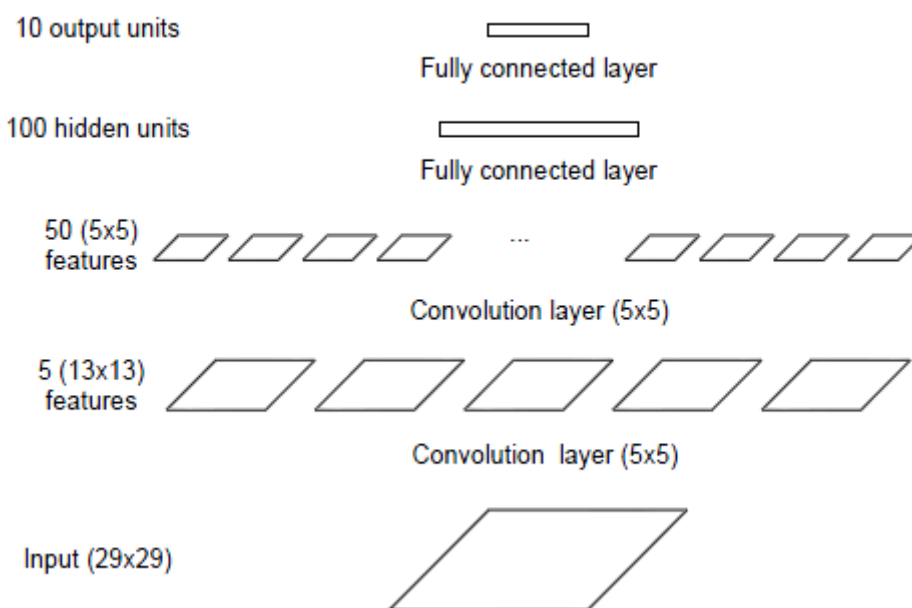


图 2.4 简化后的卷积神经网络平面拓扑结构图

卷积神经网络通过 3 大特征来保证对位移 (shift)，缩放 (scale)、扭曲 (Distortion) 的鲁棒性：局部感受区域，共享权值，时空亚取样。卷积神经网络是一个多层的神经网络，每层由多个平面组成，每个平面有多个独立的神经元。



神经元提取图像局部特征，将其映射成一个平面，特征映射函数采用 sigmoid 函数作为卷积网络的激活函数，使得特征映射具有位移不变性。每个神经元与前一层的局部感受野相连。同一层的神经元权值相同，有相同程度的位移、旋转不变性。每个特征提取后都紧跟着一个用来求局部平均与二次提取的亚取样层。这种特有的两次特征提取结构使得网络对输入样本有较高的畸变容忍能力[19]。

卷积神经网络的处理包括向前学习和向后学习。向前学习是图像矩阵进入输入层与卷积层的核函数卷积，最终得到输出。而向后学习则是根据实际输出与期望输出的误差来修正权值，直到最终误差达到要求的最小误差或达到规定的最大迭代次数，至此网络训练截止。网络训练结束后将每层的权值向量保存到相应的文件中。测试时只需将同样归一化到  $29 \times 29$  大小的图像矩阵经输入层进入网络，网络的输出就是最终的识别结果。当然也可以对每层参数进行调节。

### 2.3.2 卷积计算

一个  $N \times N$  的图像与一个  $K \times K$  的核卷积，就是将这个核在图像上重迭地依次滑过。每次核处在图像的某一位置，就产生一个输出像素值，该值是将核与图像重叠区域对应像素点的值乘积求和得到（点乘）。图 2.5 表现了输出图像 Y 的前两个像素值的计算过程：通过一个  $6 \times 6$  的输入图像 X 与一个  $2 \times 2$  的核 W 卷积。

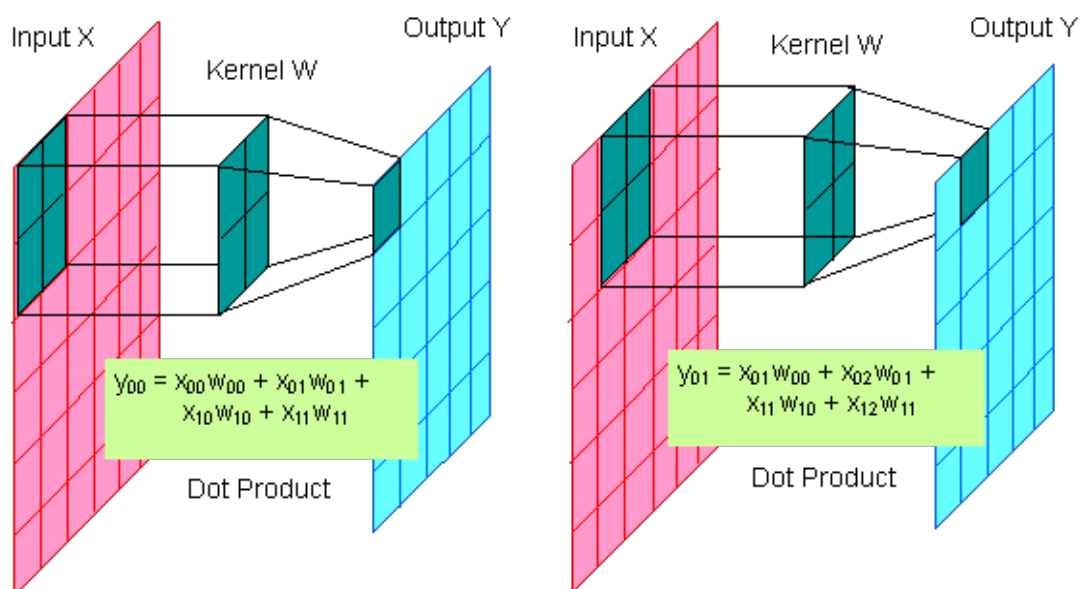


图 2.5 2D 卷积的图示

一般地，一个  $N \times N$  的图像与一个  $K \times K$  的核卷积，得到的输出图像大小为  $(N-K+1) \times (N-K+1)$ 。

### 2.3.3 卷积神经网络的拓扑结构

一个卷积神经网络由多层组成，每层包含多个特征图 FM(feature maps)。一个特征图与一般的图像一样，它的像素点在神经网络的拓扑结构中称为神经元(neurons)。

第一层只有一幅特征图：输入图像本身。在后续的层中，每个特征图拥有一定数目的特有的核（2D 权值序列），核的数目等于前一层包含的特征图的数目。每个特征图的核，其大小尺寸相同，是可以设置的参量。特征图中的像素值是通过将它的核与对应的上层特征图卷积得到。最后一层的特征图数目等于分类数目。例如，对 0~9 数字图像进行分类，则输出层将有 10 个特征图，而像素值最大的特征图就是分类结果。

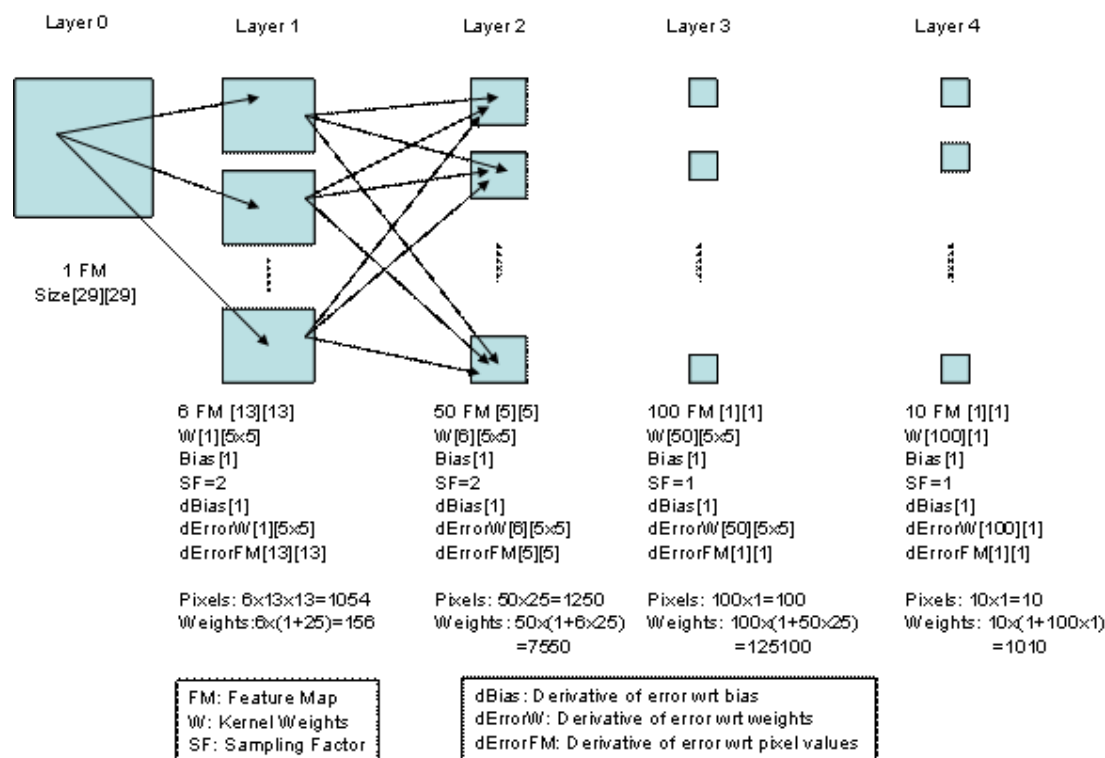


图 2.6 卷积神经网络拓扑结构示意图

图 2.6 为一种卷积神经网络的结构。每层的特征图数目以及其尺寸都在该层下面标注了。如第一层有 6 个特征图，每个特征图的尺寸都是  $13 \times 13$ 。每层每

个特征图包含的核也在图中标注。如第二层下面标注的  $W[6][25]$  表示该层每个特征图拥有 6 个核（与前一层特征图数目相同），每个核有 25 个权值（ $5 \times 5$  的权值矩阵）。除了这些权值，每个特征图还拥有一个乖离权值（bias weight）。每个像素还有一个采样因子参量 SF(sampling factor)，在前向过程中将介绍。另外 3 个参量 dBias, dErrorW 和 dErrorFM 将在反向过程中介绍。图中每层下面的最后两行给出了该层的总像素（神经元）数目和权值数目。

### 2.3.4 前向过程

如图 2.7 所示，每个特征图的像素值通过它的核与上层特征图卷积得到。假设第 J 层有 3 个特征图，那么第 J+1 层的每个特征图将有 3 个核。每个核都将与第 J 层相应的特征图卷积，以求得第 J+1 层特征图的像素值，如下图所示：

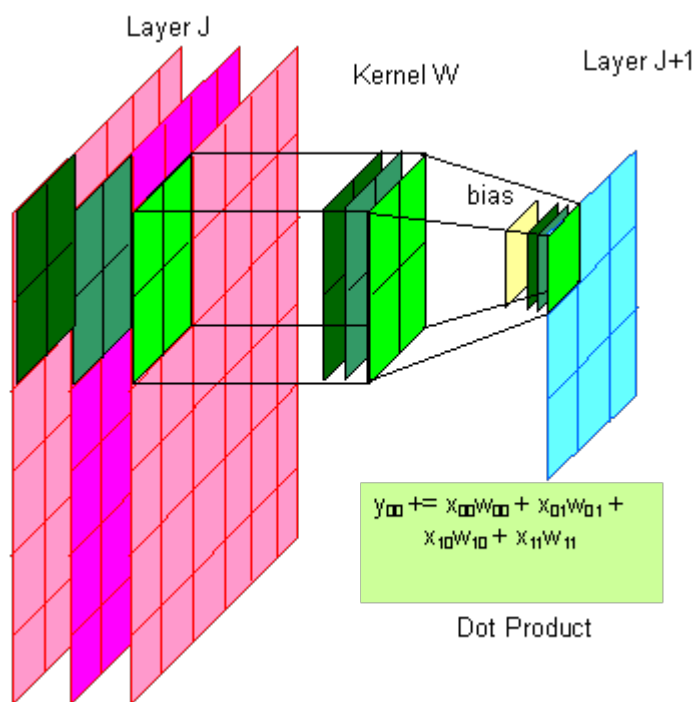


图 2.7 特征图与核卷积以及采样因子为 2 的亚取样得到下一层特征图

图中第四个被加上的值是乖离值(bias)，特征图中的每个像素点都要加上该值。 $Y_{00}$  是第 J+1 层第一个节点输入值， $X_{00}$  等是上层的输出，W 则是卷积核的权值。如果第 J+1 层有更多的特征图，那么每个特征图都将拥有 3 个自己的核，并通过同样的卷积方式来计算像素值。

按照上述方式计算出像素值之后，我们需要运用激活函数（常选择 Squashing 函数或 Sigmoid 函数）来将这些值限制在-1 到 1 的范围内。在这里我们选用下面的激活函数：

$$X[n] = 1.7159 \tanh(0.66666667Y[n]) \quad (2.17)$$

$Y[n]$ 和  $X[n]$ 分别是通过激活函数之前和之后的像素值。

这里有一个问题：按照这种方式得到的特征图尺寸与理论上应得的特征图尺寸不符。例如， $13 \times 13$ 的特征图与  $5 \times 5$ 的核卷积，下一层应得到  $9 \times 9$ 的特征图，而图 2.6 中得到的却是  $5 \times 5$ 的特征图。实际上，这是亚取样的结果。由 Dr. LeCun 提出的网络中，在第一层与第二层之间还有一层亚取样层，它采取隔行、隔列取值的方法，保留一个像素值，而丢弃同行下一列与同列下一行的像素值。Dr. Simrad 提出了一种新的方式：与其先计算后丢弃，不如直接计算将会保留的像素值。这样卷积核移动的步长就是取样因子大小。这种方式减少了卷积计算量。

Dr. LeCun 和 Dr. Simrad 都将网络的最后两层设计为全连接层。在编程实现中，权值属于特征图，全连接层相当于核为  $1 \times 1$ ，而前一层的特征图尺寸也为  $1 \times 1$ ，这样全连接层与卷积层本质上是一样的。在反向过程中，我们需要计算每一层输出的像素值和核的权值的误差。因此，对于每个特征图，我们还需要两个数组，大小分别等于特征图的大小和核的大小。这些数组和大小在图 2.6 中都表示出来了。

### 2.3.5 反向过程

反向过程用于训练卷积神经网络，调节权值大小。输出的误差，如，最后一层像素值的误差，被反向传播经过各层，进而调节每层的权值。这通过以下步骤实现：

- (1) 误差对最后一层（输出层）神经元输出值的导数

首先求出误差对最后一层（输出层）神经元输出值的导数。这只需要将输出值与理想输出值相减即可：

$$dErrorX = X - X_{ideal} \quad (2.18)$$

注意对于最后一层的神经元，对应于正确类别的神经元理想输出为 1，其他神经元理想输出为 0。

- (2) 误差对输出层输入值的导数

上一步求出了误差对输出层输出值的导数。现在计算误差对输出层输入值的导数（即还没有经过激活函数作用的像素值的导数）。只需要将上述导数乘以激活函数的反函数即可：

$$dErrorY = \text{InverseSIGMOID}(X) * dErrorX \quad (2.19)$$

$$\text{InverseSIGMOID}(X) = \frac{0.66666667}{1.7159} * (1.7159 + X) * (1.7159 - X) \quad (2.20)$$

### (3) 误差对权值的导数

接下来计算误差对权值的导数。在上一层中，所有在前向计算过程中共享某一权值的神经元都将被用于计算误差对该权值的导数。

考虑图 2.7 表示的前向计算过程：第 J 层的一个特征图中的一个  $2 \times 2$  的方块通过一个  $2 \times 2$  的核与第 J+1 层的一个像素点相连。这些连接对于误差对核中的权值的导数的贡献按如下方式计算：

$$\begin{aligned} dErrorW_{00} &+= dErrorY_{00} * X_{00} \\ dErrorW_{01} &+= dErrorY_{00} * X_{01} \\ dErrorW_{10} &+= dErrorY_{00} * X_{10} \\ dErrorW_{11} &+= dErrorY_{00} * X_{11} \end{aligned} \quad (2.21)$$

同样的核也将第 J+1 层所有其他像素点与第 J 层同一特征图的其他像素点连接起来，这些连接对权值的共享也将以同样的方式对总  $dErrorW$ （对这个核）的计算作出贡献。同样的步骤在所有其他属于第 J+1 层每个特征图的核上重复进行。乖离值（bias）是在前向过程中加到特征图的每个像素值上的，与前一层没有连接。因此， $dErrorb$ （误差对乖离值的导数）可以简单地通过将特征图中的所有像素点的  $dErrorX$  相加得到（相当于  $X=1$ ）。

以图 2.6 第 4 层第一个特征图为例。它有一个 bias 和 100 个权值，每一个权值都与第 3 层的一个不同的神经元相连。该特征图只有一个像素点，它的  $dErrorY$  与所有第 3 层中和它相连的 100 个神经元的值相乘，这样就得到了所有的权值的  $dErrorW$ 。

对于以上的解释，可以理解为反向计算与前向计算的卷积运算十分相似。如果前向计算的代码已经正确实现了，反向计算的代码就很容易写出。

### (4) 更新权值

计算了  $dErrorW$  之后，我们就可以用下面的公式更新权值：

$$(weight)_{new} = (weight)_{old} - eta * dErrorW \quad (2.22)$$

$eta$  是学习率，是很重要的参数之一。

### (5) 前移到上层

现在我们要移到前一层。对于最后一层，在第 1 步中很容易得到  $dErrorX$ ，但是对于其他层则比较困难，因为我们并不知道其他层的理想输出值。我们需要使用类似于步骤 3 的一个过程。所有的神经元，如果从前一层某一神经元得到了贡献（即有连接），它们的  $dErrorY$  将与相对应的权值相乘，乘积相加。这给出了前一层的  $dErrorX$ ，然后就可以进行步骤 2~4 来更新该层权值。

考虑图 2.7，第  $J+1$  层中的一个像素点与第  $J$  层中的一个特征图的 4 个像素点连接。在反向过程中，第  $J+1$  层的这个像素点对第  $J$  层的那 4 个像素点的  $dErrorX$  有贡献，计算方式如下：

$$\begin{aligned}(dErrorX_{00})_j &+= (dErrorY_{00})_{j+1}(W_{00})_j \\(dErrorX_{01})_j &+= (dErrorY_{00})_{j+1}(W_{01})_j \\(dErrorX_{10})_j &+= (dErrorY_{00})_{j+1}(W_{10})_j \\(dErrorX_{11})_j &+= (dErrorY_{00})_{j+1}(W_{11})_j\end{aligned}\quad (2.23)$$

注意到第  $J+1$  层的同一个像素点通过不同的权值与第  $J$  层其他特征图的相同位置的 4 个像素点连接，并且将以相同的方式对它们的  $dErrorX$  贡献。类似地，如果第  $J+1$  层有更多的特征图，它们也将以相同的方式对第  $J$  层的每个特征图的所有像素贡献。这一步的代码与第 3 步很相似。

为了进一步理解这一步骤，考虑图 2.6 第一层第一个特征图的第一个像素点。它与第 2 层的 50 个特征图的第一个像素点通过 50 个不同的权值（第 2 层每一个特征图的第一个核的第一个权值）相连接。这 50 个权值与第 2 层这 50 个神经元的  $dErrorY$  的点乘将给出第一层第一个神经元的  $dErrorX$ 。

#### (6) 循环步骤

在第 5 步之后，回到第 2 步，然后对其他所有层重复循环，除了第一层（输入层），因为该层没有权值需要更新。

### 2.3.6 卷积神经网络小结

卷积神经网络是近几年发展迅速的机器学习理论：深度学习(Deep Learning)的模型中的一种。深度学习的核心思想是多层网络对事物的多层抽象，可以学习到深层次的特征。因此，相对于 BP 神经网络这种层数较少的神经网络，卷积神经网络具有更为强大的学习能力。并且其对特征的提取是学习的一部分，不需要人为地提取特征。当然，这种强大的学习能力有一定的代价：

### (1) 复杂的训练过程

由于网络的结构复杂,参数众多,因此对于卷积神经网络这种深网络(Deep Neural Network)的权值训练过程比 BP 神经网络困难。

### (2) 大量的训练样本

权值增加导致的一个自然而然的结果是对训练集数目的需求大大增加。若训练样本不足,可能导致网络的过拟合,从而影响对新数据的分类能力。

深学习的原理还有很多部分未被人们完全理解,越来越多的研究人员对深度学习理论研究的产生兴趣。相信未来的一段时间里,深度学习理论将得到更大的进展。

## 第 3 章 搜狐验证码的识别

本文的工作是针对搜狐验证码进行识别。尽管验证码的识别具有针对性,不同类型的验证码有不同的分割和识别算法,但它们总的破解方案都是基于以下流程,不同的只是每个模块的算法:

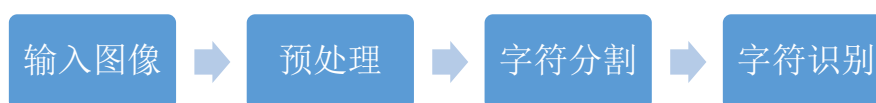


图 3.1 验证码识别流程

预处理一般包括彩色图像灰度化、二值化和去噪等,目的是从背景中提取出干净的字符。之后进行字符分割,将图形中的字符分割成单个字符。最后利用分类器对字符进行训练与识别。

### 3.1 预处理

#### 3.1.1 彩色图像灰度化

大部分验证码是真彩色图像,预处理的第一个步骤是将图像从 RGB 空间转化为灰度空间。灰度化根据以下公式进行操作:

$$Y=0.299*R+0.587*G+0.114*B \quad (3.1)$$

当颜色信息可以用来分割和分类时不需灰度化。

搜狐验证码有以下几种典型样式:

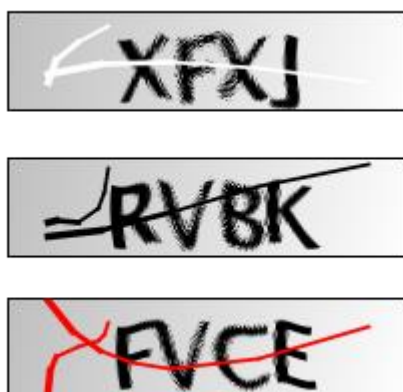


图 3.2 典型搜狐验证码

由此可见搜狐验证码的特征：

- (1) 字符为非印刷体，需要提取多种模版；
- (2) 干扰线贯穿整个验证码，且颜色不一，使字符难以分割；
- (3) 具有水波状模糊效果，使字符噪声增加，加大了去噪与分割的难度。

经过灰度化之后图像如下：



图 3.3 灰度化的搜狐验证码

### 3.1.2 灰度图像二值化

一般的验证码背景和前景在灰度上区分很明显，在直方图上体现出双峰的特征。我们拟使用 OSTU（最大类间方差）全局阈值分割来进行二值化操作。

OSTU 方法由日本大津于 1978 年提出[21]，从模式识别的角度，最佳阈值应产生最佳的背景类和目标类的分离性能。此性能用类别方差来表示，为此我们引



入类内方差 $\sigma_w^2$ 、类间方差 $\sigma_B^2$  和总体方差 $\sigma_T^2$  ,人们一般通过最大化 $\sigma_B^2/\sigma_T^2$  获取最有阈值。

在实际运用中, 往往使用以下简化计算公式。

$$\sigma^2(T) = W_A(\mu_a - \mu)^2 + W_B(\mu_b - \mu)^2 \quad (3.2)$$

其中,  $\sigma^2$ 为两类间最大方差,  $W_A$  为A类概率,  $\mu_a$ 为A类平均灰度,  $W_B$ 为B类概率,  $\mu_b$ 为B类平均灰度,  $\mu$ 为图像总体平均灰度。

阈值T将图像分为A,B两个部分,使得两类间总方差 $\sigma^2(T)$  取最大值的T, 即为最佳分割阈值。

当图像字符和背景灰度值区别很明显时, 可用固定阈值法二值化。

利用 OSTU 算法二值化后的搜狐验证码效果如下:

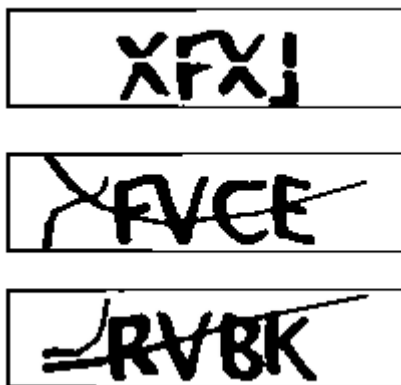


图 3.4 搜狐验证码二值化图

### 3.1.3 图像去噪

一般有点噪声、线噪声和块噪声。去除图像中的噪声的运算在图像处理中称为图像平滑, 它主要利用噪声高频, 孤立大偏差的特点进行。图像平滑常用的方法有邻域平均和中值滤波等方法。针对不同的噪声应采用不同的去噪方法。

#### (1) 点噪声的去除

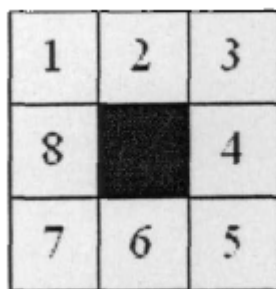


图 3.5 点噪声去除方法

主要针对杂点噪声。在已经二值化且黑色代表字符的图像中，遍历图像的每个黑像素点。

(2) 细线噪声的去除

当直线噪声宽度明显小于字符时，即对于细线噪声，可以通过形态学上的腐蚀算法，腐蚀可以进行多次操作。

(3) 粗线和块噪声的去除

对于粗线噪声和块噪声，采取连通域去噪的方法。由于噪声连通域的面积小于字符域。

(4) 其他去噪方法

若噪声形状比较规则一致，可以利用模板匹配的方法找到噪声位置处将其去除。

对于椒盐噪声等可采用中值滤波。

中值滤波是一种非线性信号处理方法，在一定条件下可以克服线性滤波器所带来的

通常噪声处理时这些方法可以混合使用。

经过试验，发现通过以下步骤对搜狐验证码的二值化图像进行去噪处理效果较好：

(1) 连通域去噪

通过计算像素点 8 邻域内的黑点个数，将黑点个数小于或等于 2 的像素点标记为噪点。全部标记完后，将黑色噪点转化成白色背景。

(2) 腐蚀

利用 OpenCv 自带的函数 `cvErode` 对图像进行一次腐蚀。由于该函数将白色作为前景，黑色作为背景，因此在腐蚀之前需要将图像灰度翻转，腐蚀后再次翻转。

### (3) 二次连通域去噪

经过腐蚀后，验证码中的干扰线几乎被清除。但又产生了新的黑色噪点，因此进行第二次连通域去噪，与第一步操作相同。

通过以上三个步骤的处理，得到的去噪验证码如下：

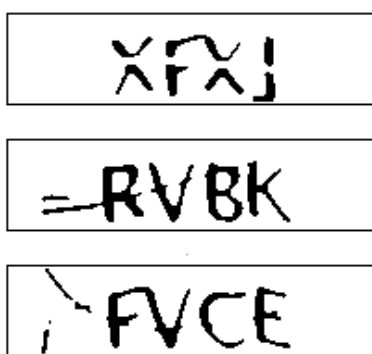


图 3.6 搜狐验证码去噪图

## 3.2 字符分割

关于字符分割，我们拟采用以下方法水平垂直投影法。

水平垂直投影字符分割算法是分别向 X 轴和 Y 轴投影，投影值是该行或该列的

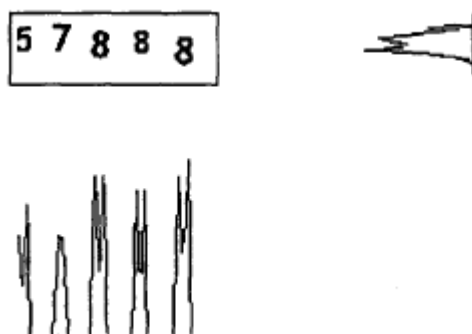


图 3.7 图像的水平垂直投影图

若字符有粘连

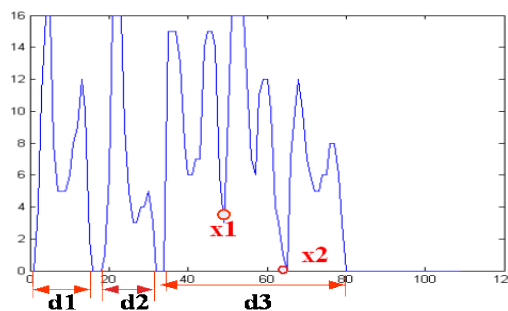


图 3.8 黏连字符的 X 轴投影

字符的分割点处是投影波形的极小值点。而极小值点是由投影值的差分得到的。设投影波形 $d(x)$ ，其差分  $d'(x) = d(x+1) - d(x)$ 。极小值点须满足

$$\begin{cases} d'(x-1) < 0 \\ d'(x+1) < 0 \end{cases} \quad (3.3)$$

或

$$\begin{cases} d'(x) \leq 0 \\ d'(x-1) \leq 0 \\ d'(x+1) \leq 0 \end{cases} \quad (3.4)$$

在求解得到的所有极小值点中，认为满足下式且投影值最小的极小值点就是正确的分割点。

$$D_{min} < D(x_i, x_j) < D_{max} \quad (3.5)$$

其中， $D(x_i, x_j)$ 是两个极小值点 $x_i, x_j$ （包括端点）间的距离， $D_{min}, D_{max}$ 分别为最小字母宽度和最大字母宽度。

同样的方法在 Y 轴投影计算，即可找到单个字符在 Y 轴的位置范围。这样即可将字符单独分离开来。

根据搜狐验证码去噪后的特征，对于投影效果明显的区域采用投影分割，如果投影效果不明显，即发现分割字符数大于 4，则采用等分法分割以增加分割正确的概率。得到的分割字符并大小归一化后如下：



图 3.9 搜狐验证码分割子图（归一化）

### 3.3 字符识别

针对搜狐验证码，我们试验了三种识别算法：LMS 算法、BP 神经网络、卷积神经网络。

#### 3.3.1 不同算法的识别过程

##### (1) LMS 算法识别

对于分割后的字符，将其提取  $5 \times 5$  的网格特征：即算出每个网格中黑色像素点占网格面积的比例，再与模版库中的所有模版的网格特征向量对比，计算均方误差，取误差值最小的模版的字符作为字符识别的结果。

利用 LMS 算法，得到的搜狐验证码的识别率为 56.5%。

##### (2) BP 神经网络识别

采用 BP 网络的关键在于提取特征。特征的选择和特征的个数直接影响着分类器的分类效果。利用 BP 神经网络识别搜狐验证码的流程如下图：

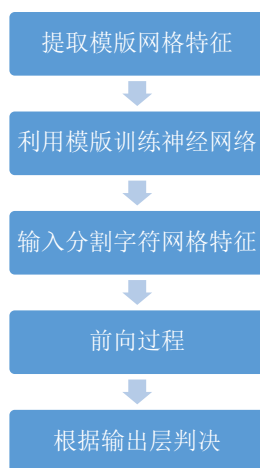


图 3.10 BP 神经网络识别搜狐验证码流程

实验中使用的向量仍然是 25 维网格特征向量。参数设置为：隐含层节点数 20，学习率 0.05，最小误差限 0.02。

利用 BP 神经网络算法，得到的搜狐验证码的识别率为 52%。

### (3) 卷积神经网络识别

卷积神经网络识别过程与 BP 神经网络类似。不同的是网络结构的差别和训练参数的设置。

经过多次试验，我们选取的参数为：网络结构与图 2.7 类似，不同的是输出层为 19 个神经元，因为搜狐验证码有 19 中可能的字符；输入向量也为  $29 \times 29$  的字符图片，因此模版的大小也归一化为这种尺寸以进行训练；学习率为 0.0005；训练至样本数据全部分类正确为止。



实验中使用的训练样本为字符模版库，与 BP 神经网络的样本库相同。不同的是，BP 神经网络的输入是模版库提取的 25 维字符特征，而卷积神经网络的输入是模版库归一化为  $29 \times 29$  的尺寸图片。

利用卷积神经网络，得到的搜狐验证码的识别率为 39.5%。

## 3.3.2 识别结果和分析

搜狐验证码的部分识别结果如下：

表 3.1 部分识别结果

验证码	识别结果	验证码	识别结果
	XFXJ		NXAE

<del>KMDD</del>	KMDD	<del>HWAB</del>	HWAT
<del>XFXD</del>	XFXD	<del>JUTP</del>	JUJP
<del>KCUB</del>	KCUB	<del>NBHH</del>	NRHH
<del>XAXU</del>	UAXU	<del>UXFN</del>	UXFN
<del>WMHJ</del>	WMHJ	<del>DMTN</del>	DMTN
<del>YHTX</del>	FXTX	<del>HDHK</del>	HDUL
<del>UCVB</del>	UCVB	<del>FBRT</del>	FBRT
<del>BJHM</del>	RRJB	<del>ACPH</del>	ACPH
<del>WTLK</del>	WTLK	<del>UXAL</del>	LAAL

搜狐验证码的识别率如下表(样本总数为 200):

表 3.2 不同分类算法的识别率

分类算法	LMS 算法	BP 神经网络	卷积神经网络
识别率	56.5%	52%	39.5%

由上表可见 LMS 算法虽然思想简单,但是取得的效果很好;BP 神经网络经过有效的训练之后可以取得接近 LMS 算法的效果;卷积神经网络效果则不如前两个算法,但是 39.5%的正确率证明了该算法的潜力。

LMS 算法执行简单,效果稳定。但是无法抓住字符的结构特征,在字符十分扭曲的情况下效果不好。

BP 神经网络在字符扭曲的情况下,通过大量样本的训练,可以取得很好的效果。但 BP 神经网络十分依赖训练参数的设置以及特征向量的选取。特征选取的好坏直接影响取得的正确率。

卷积神经网络可以自适应地选择恰当的特征而不需要人工提取特征,这是相对于 BP 神经网络的一大进步。但是相应地,其结构更加复杂,训练难度更大,

需要的样本也更多。在搜狐验证码的识别中，在样本相同的情况下，效果不如BP神经网络。



## 第 4 章 验证码识别软件 CAPTCHA Recognizer

为了整合验证码识别算法，便于演示以及今后的扩展，利用 Qt5.0.1 与 OpenCv1.0，在 Win7 环境下构建了验证码识别软件 CAPTCHA Recognizer。

### 4.1 设计目标

CAPTCHA Recognizer 的设计旨在实现以下几个功能：

- (1) 选择一幅验证码图片，可以利用已有的算法对其进行预处理与识别；
- (2) 选择多幅验证码图片，可以批量进行识别；
- (3) 可以对图片进行一般的图像处理，如灰度化、二值化、锐化等等

### 4.2 系统设计

#### 4.2.1 需求分析

根据软件的设计目标，设想利用该软件可以有如下的场景模型：

##### (1) 识别单张验证码

用户点击“打开”菜单或者工具打开文件夹，选择计算机中某验证码图片。软件将图片显示，并且判断该图片属于哪种类型的验证码。用户点击“识别”工具，可以得到字符识别结果；用户点击“清理”工具，可以显示该图片经预处理之后的结果。

##### (2) 批量识别验证码

用户点击“选择图像”按钮，选择了多个验证码。软件读入所有图片并存储于列表。用户点击“识别图像”按钮，软件依次逐个识别并显示结果。如果有正确的标签可以对比，用户可以选择记录着正确标签的文本文件，点击“识别报告”按钮，软件可以将识别结果与记录文本对比，报告识别的正确率，以显示算法的有效性。

##### (3) 预处理工具箱

对一幅验证码图像，用户可以利用常用的数字图像处理算法对其进行预处理，并且可以应用于验证码识别的预处理阶段。选择一幅验证码图片后，

在预处理工具箱中有各种处理步骤，如灰度化、二值化、滤波、锐化等等。用户可以选择多个处理步骤并调节参数，以达到好的预处理效果。而这一系列步骤可以固定并替代已有算法的预处理函数。该功能主要用于开发新类型的验证码的识别算法，适合具有专业知识的用户使用。

以上操作可以用如下的 UML 活动图表示：

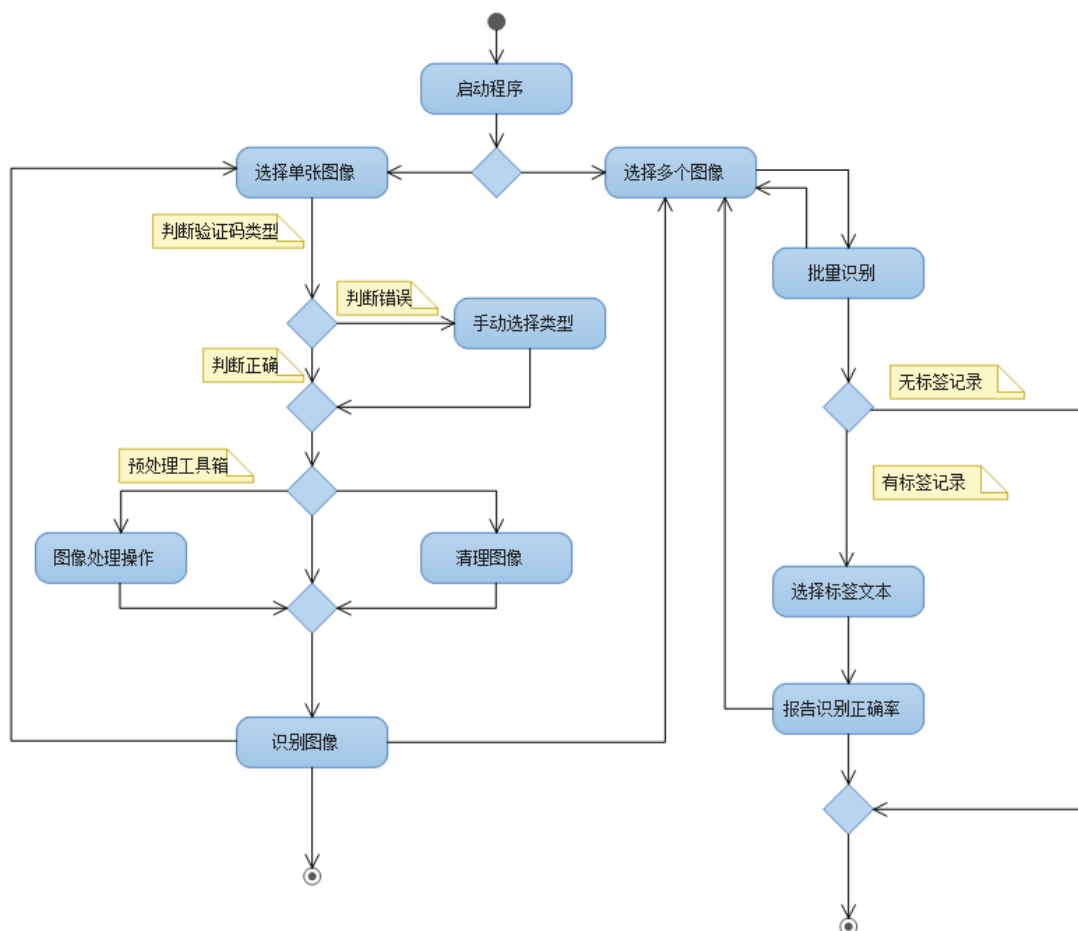


图 4.1 CAPTCHA Recognizer UML 活动图

## 4.2.2 结构建模

根据软件的设计目的与活动流程，构思的软件结构模块模型如下：

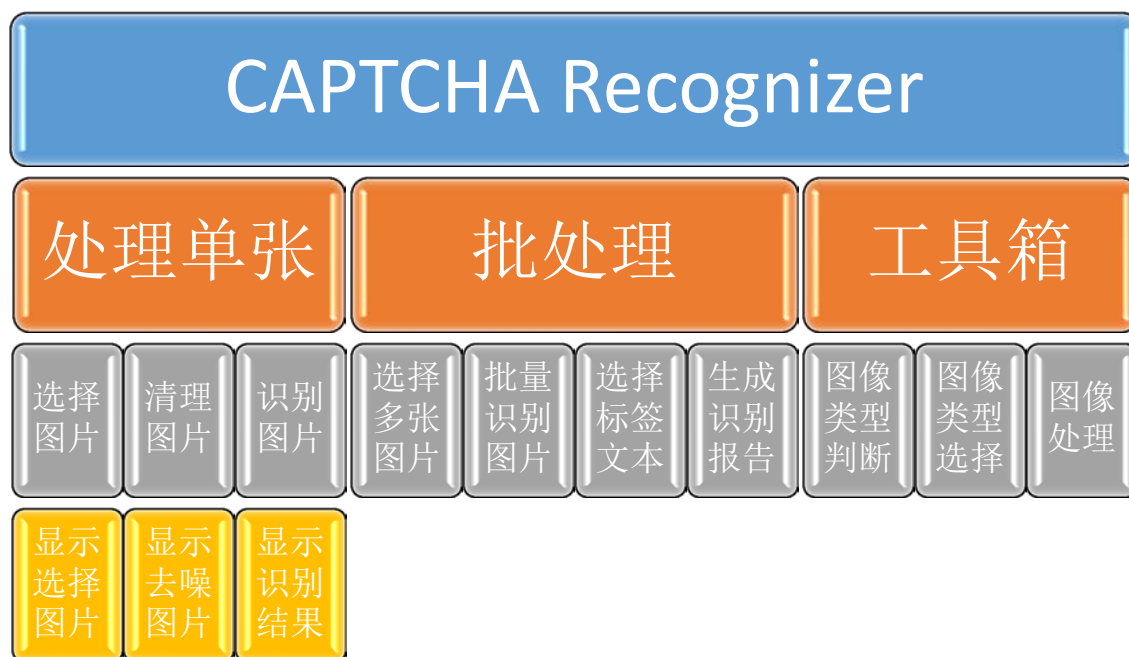


图 4.2 CAPTCHA Recognizer 软件架构图

如图 4.2 所示。CAPTCHA Recognizer 主要包含 3 大模块，分别实现 3 个主要设计目标：单张验证码识别、批量验证码识别、图像处理工具。软件各个模块的耦合性低，相互间独立性强，这样便于模块复用和功能的独立实现。对于各个模块的子模块，之间的耦合性也尽可能降至最低。这种模块化结构设计，有利于今后增加其他功能模块和加入新的算法。

## 4.3 系统实现

### 4.3.1 界面设计

设计的软件界面如下：



图 4.3 CAPTCHA Recognizer 界面 1

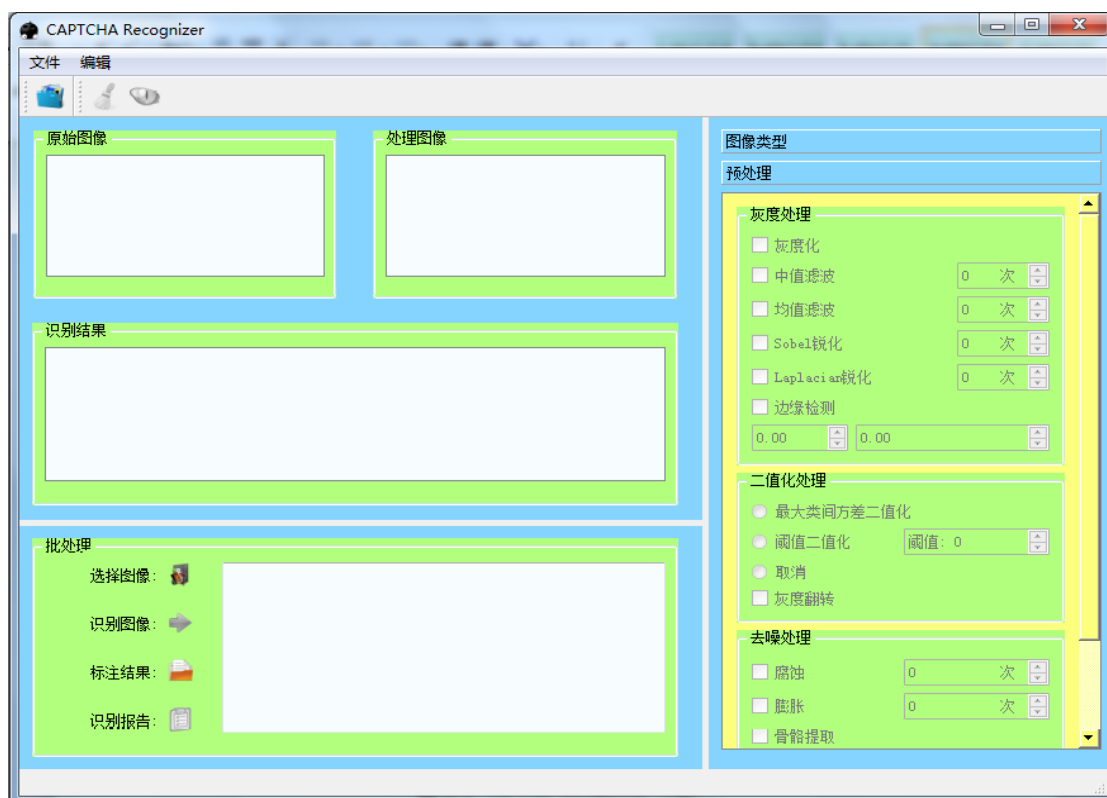


图 4.4 CAPTCHA Recognizer 界面 2

软件主要分为 3 部分：

#### (1) 识别单张图片

左上方区域为单张图片识别模块。在工具栏上的三个工具分别为打开、预处理、识别。上方两个矩形框分别用于显示打开的图片和预处理后的图片；下方大矩形框用于显示识别结果。

#### (2) 批处理

左下方为批处理模块。左边的四个按钮功能分别对应图 4.2 中的批处理的四个子模块功能；右边矩形区域用于显示批量识别的结果。

#### (3) 工具箱

工具箱包含两部分：类型判断与预处理。图 4.3 右方为类型模块。最上方的矩形区域用于显示当前选择的图片为何种类型，下方单选框用于人工设定图像类型。图 4.4 右方为预处理工具箱，主要包括灰度化处理、二值化处理、去噪处理。

各个组件拥有激活条件：如当没有读入图像时，预处理和识别工具无法使用，此时为非激活状态，因此图标呈现灰白色。当读入了图片之后，可以使用的工具将自动转换为激活状态，可以使用。其他工具与按钮也是如此。

### 4.3.2 关键技术

下面针对各个模块，简要介绍实现的关键步骤。

#### (1) 工具栏

工具栏包含“打开”、“清理”、“识别”这三个工具。“打开”动作的槽函数产生一个文件选择对话框。若用户选择了文件，记录文件地址和文件名，并激活可用的组件功能。同时，调用图像类型判断模块判断类型。“清理”和“识别”工具的槽函数为预处理函数与识别函数。在类型判断之后，设定了这两个工具应该调用哪种验证码算法的函数。

#### (2) 单张验证码识别

CAPTCHA Recognizer 用于集成多种验证码的识别算法。对于每种验证码，有相应的一个类与之对应。这个类提供一个预处理函数和识别函数供软件调用。预处理函数的输入为一个验证码图片，输出为去噪后的二值图片；识别函数的输入为一个验证码图片，输出为字符识别结果。我们设定两个函数指针，分别指向预处理函数和识别函数。类型判断模块可以设定这两个函数指针分别指向哪种验证码识别类的子函数。

### (3) 批处理

批处理模块中，我们可以选择多个文件。选中的文件，其完整路径名被存储在一个文件列表中。识别函数依次从列表中读入一幅图像的地址，对其进行识别，输出的识别结果显示在文本框中。如果有预先记录好的结果标签，则选择该标签文本文件。点击“识别报告”按钮，其槽函数将在标签文本文件中逐行读入（每行对应一个验证码图片的字符），与识别结果列表中相应行的字符比较。最后统计识别正确的数目，得到识别率。

### (4) 类型判断

CAPTCHA Recognizer 的一个关键技术是：图像类型判断。

由于可以识别的验证码种类不唯一，而每类验证码对应的识别算法不一样，调用的函数相应的也不一样。因此，读入验证码图像之后，对其类型进行准确地判断，从而设置相应的预处理和识别函数指针，是很关键的一步。

由于同类的验证码字符不用，噪声等的分布也不一致，因此无法采用模版匹配的方法。而同种验证码之所以给人的感觉相似，是因为其灰度分布相似。因此，采用灰度直方图比较的方法，对验证码类型进行判别。图像类型判断的流程如下：

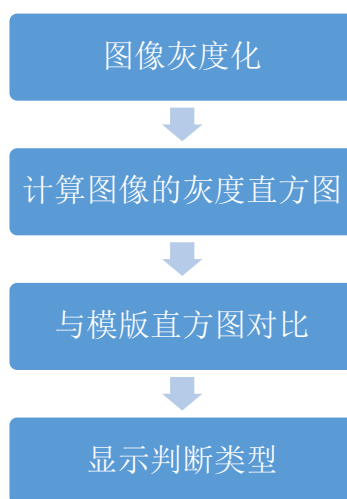


图 4.5 图像类型判断流程

每类验证码挑选几个典型类型的图片作为比较模版，计算出它们的直方图，存储在直方图模版库中。对于读入的图片，将其灰度化之后，计算

得到直方图数组  $H_1$ ；模版的直方图数组为  $H_2$ 。则利用下面的公式计算两直方图数组的相关系数：

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \bar{H}_1)(H_2(I) - \bar{H}_2)}{\sqrt{\sum_I (H_1(I) - \bar{H}_1)^2 \sum_I (H_2(I) - \bar{H}_2)^2}} \quad (4.1)$$

最后，选择相关系数最大的直方图对应的验证码类型作为判别类型。下图显示了图像类型判断的结果。经检验，准确率达 100%。



图 4.6 不同图像类型判断结果

#### (5) 预处理工具箱

预处理工具箱中包含各种图像处理算法。选中相应的复选框，则在图像的预处理过程中加入该处理步骤。

## 4.4 测试运行

下面演示 CAPTCHA Recognizer 的主要功能。

#### (1) 已经破解的验证码

以搜狐验证码为例，在选择一幅验证码图片之后，点击识别工具可以进行识别，点击清理工具可以观察算法对其二值化并去噪后的结果，并且图像的种类可以利用直方图对比自动判别，如下图：

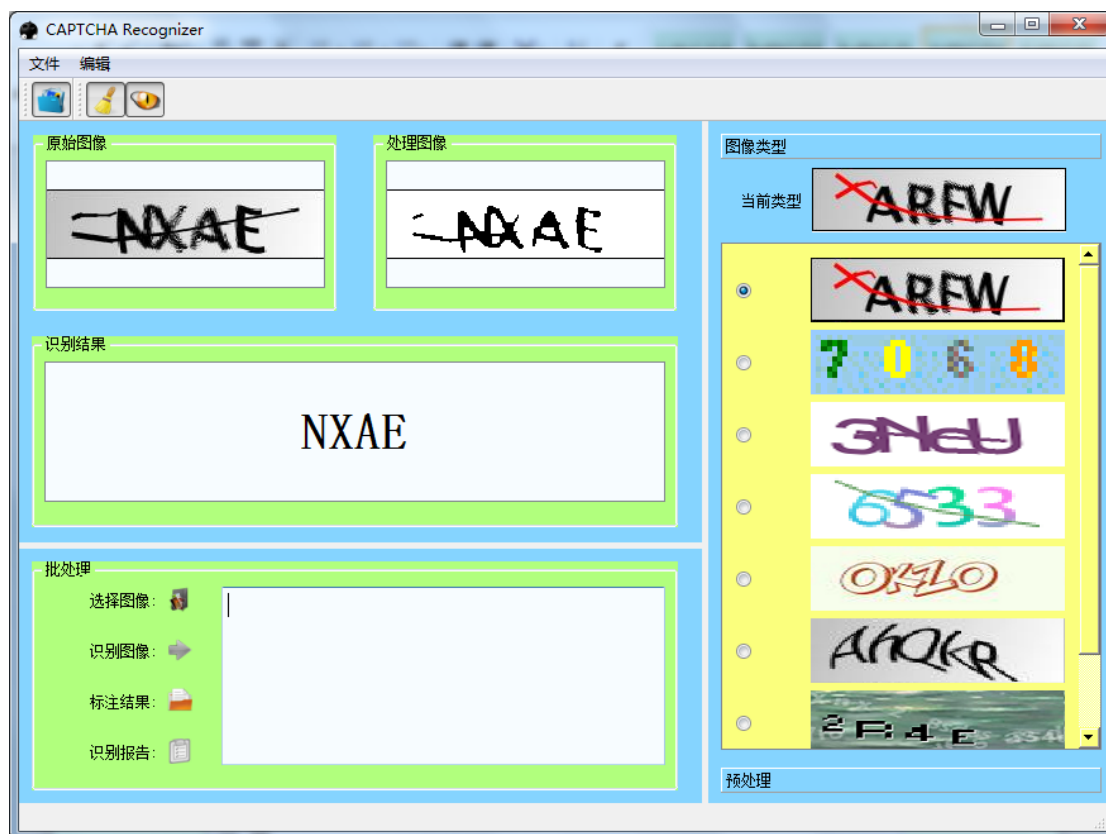


图 4.7 识别单张验证码

## (2) 工具箱

图像处理工具箱模块可以对验证码进行灰度化，二值化，腐蚀，膨胀等图像处理步骤，并且可以用于新验证码的预处理，后期交由识别算法进行识别：





图 4.8 预处理工具箱

### (3) 批量识别

批处理模块可以对验证码进行批量识别，如果有正确的结果标注，还可以对比识别结果与标注的结果，得到算法的识别率，以验证算法的有效性：

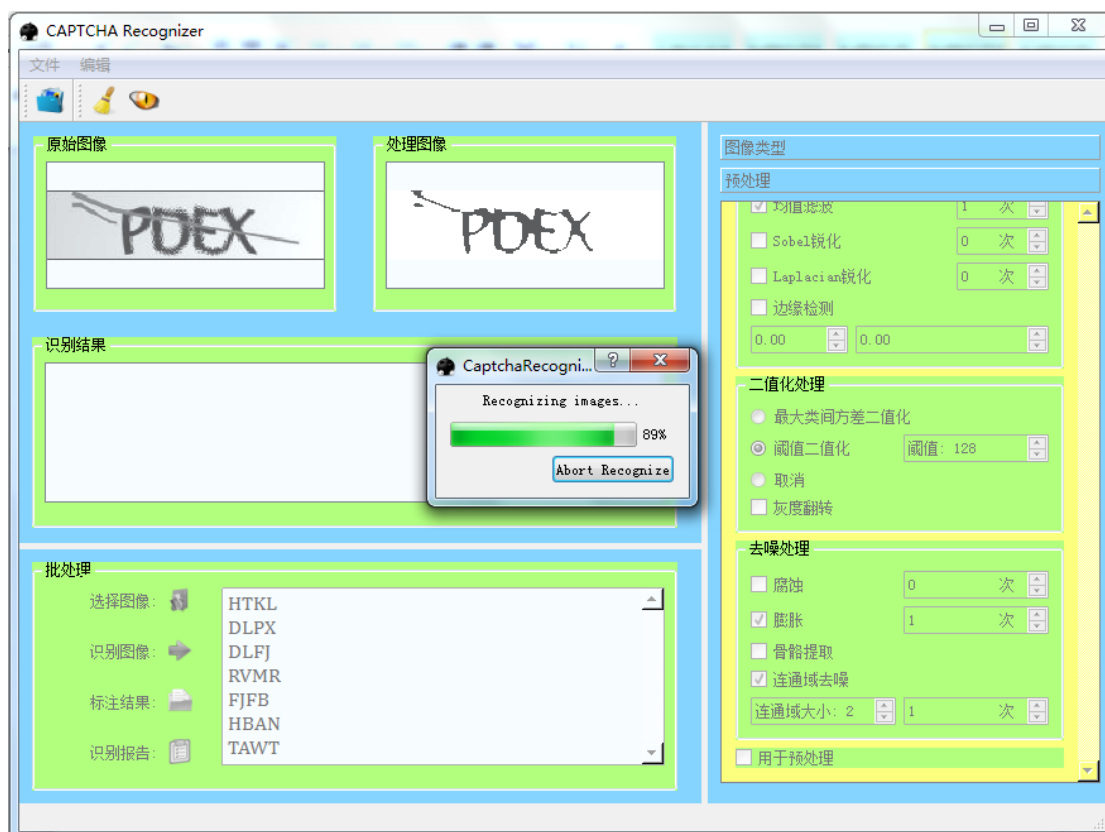


图 4.9 批量识别验证码

利用此平台，可以很方便地实现对已经破解的验证码的识别。对于未破解的验证码，可以利用预处理工具箱实验预处理的方法。另外，软件的设计方式为模块化，各组件耦合度较低。对于新的验证码识别算法，可以很方便地加入。因此，CAPTCHA Recognizer 既可以作为验证码识别工具，也可以作为验证码识别算法实验开发平台。

## 第5章 结论

本文对搜狐验证码的识别进行了研究,通过不同的算法实现了验证码的识别,说明了该验证码的设计依然存在不足。。

对搜狐验证码,我们应用了三种不同算法:LMS 算法, BP 神经网络, 卷积神经网络。达到的识别率说明了几个问题:

- (1) 验证码识别的主要难点在于前期的预处理和分割。如果分割结果很好,那么识别难度很小;
- (2) LMS 算法、BP 神经网络算法具有很高的鲁棒性,在验证码的识别中作为后期分类器效果不错,但前提是字符分割准确,特征选取恰当;
- (3) 卷积神经网络是比较复杂的神经网络,属于深度学习算法的一种。其多层结构使得它可以学习很复杂的内在特征,从而达到传统算法难以达到的效果。并且它不需人工提取特征。但是由于网络结构复杂,权值数目多,训练量大,样本需求量也很大。训练参数的选择也很关键。

利用字符分割技术和一些机器学习和模式识别的算法,我们可以破解大部分的验证码。这些验证码并不能足够可靠安全的应用在网络安全方面,存在安全隐患,需要引起使用者的注意。应该设计机器更难以破解但人类又容易识别的验证码。

验证码的识别具有针对性,不同类型的验证码有不同的分割识别方法。没有统一的方法可以识别所有的验证码。在验证码识别中,分割是比识别更为困难的问题。一旦将字符单独分开,运用机器学习算法可以轻易的解决识别问题,但现今还没有一种通用有效的算法解决所有字符分割的问题。字符分割是验证码识别中的一个关键技术,尤其是粘连、扭曲字符的分割是一个瓶颈,对它的研究至今仍然很少。

对于含噪声、扭曲、粘连的验证码,能否利用神经网络和深度学习的方法提取深层特征,从而不进行分割和预处理(或少量预处理),而是直接识别图像中的字符,这是未来研究的方向之一。

## 参考文献

- [1] <http://zh.wikipedia.org/wiki/%E9%AA%8C%E8%AF%81%E7%A0%81>
- [2] Greg Mori, Jitendra Malik, 2003, Recognizing Objects in Adversarial Clutter: Break a Visual CAPTCHA[C], IEEE Conference on Computer Vision & Pattern Recognition, IEEE Computer Society, 1(124-141)
- [3] Hoevar.S, PWNtcha[CP/OL], <http://caca.zoy.org/wiki/PWNtcha>
- [4] Gabriel Moy, Nathan Jones, Curt Harkless, Randall Potter, 2004, Distortion Estimation Techniques in Solving Visual CAPTCHAs[C], IEEE Conference on Computer Vision and Pattern Recognition(CVPR'04), 2 (23-28)
- [5] Kumar Chellapilla, Patrice Y. Simard, 2005, Using Machine Learning to Break Visual Human Interaction Proofs(HIPs)[C], in L.K.Saul, Y.Weiss, and L. Bottou, editors, Advances in Neural Information Processing Systems 17, pp.265-272. MIT Press, Cambridge, MA
- [6] Kristiansson P O. Defeating a simple CAPTCHA using Optical Character Recognition[J]. 2007.
- [7] Jeff Yan , A.EI Ahmad, 2008, A Low-cost Attack on a Microsoft CAPTCHA[C], Proceedings of the 15<sup>th</sup> ACM Conference on Computer and Communications Security, New York, USA: ACM Press:543-554
- [8] 王璐, 2011, 验证码识别技术研究, [硕士], 合肥, 中国科学技术大学
- [9] Widrow B, Hoff M E. Adaptive switching circuits[J]. 1960.
- [10] McClelland J L, Elman J L. The TRACE model of speech perception[J]. Cognitive psychology, 1986, 18(1): 1-86.
- [11] Luger G F, Stubblefield W A. Artificial Intelligence. 1993[J].
- [12] Minsky M, Papert S. Perceptron: an introduction to computational geometry[J]. The MIT Press, Cambridge, expanded edition, 1969, 19: 88.
- [13] Rumelhart D E, Hintont G E, Williams R J. Learning representations by back-propagating errors[J]. Nature, 1986, 323(6088): 533-536.
- [14] Werbos P. Beyond regression: New tools for prediction and analysis in the behavioral sciences[J]. 1974.
- [15] Parker D B. Learning logic[J]. 1985.

- [16] Le Cun Y. A learning scheme for asymmetric threshold networks[J]. Proceedings of Cognitiva, 1985, 85: 599-604.
- [17] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.
- [18] Simard P Y, Steinkraus D, Platt J C. Best practices for convolutional neural networks applied to visual document analysis[C]//Proceedings of the seventh international conference on document analysis and recognition. 2003, 2: 958-962.
- [19] 陆璐, 卷积神经网络的研究及其在车牌识别中的应用[D]:[硕士], 合肥, 合肥工业大学, 15-17, 53-57
- [20] <http://www1.i2r.a-star.edu.sg/~irkhan/conn1.html>
- [21] Ostu N. A threshold selection method from gray-level histogram[J]. IEEE Transactions on Systems, Man and Cybernetics, 1979, 9(1): 62-66.