文章编号:1671-4598(2023)07-0271-07 DOI:10.16526/j.cnki.11—4762/tp.2023.07.039 中图分类号:TP391.41 文献标识码:A

基于字符分割和 LeNet-5 网络的字符验证码识别

张敬勋[1,2]，张俊虎[1]，赵宇波[2]，李 辉[1]
(1. 青岛科技大学 信息科学技术学院，山东 青岛 266061；
2. 山东产业技术研究院，山东 青岛 250102)

摘要：为了解决传统验证码识别方法效率低，精度差的问题，设计了一种先分割后识别的验证码处理方案；该方案在预处理阶段用中值滤波去噪，再利用霍夫变换对图像字符进行矫正；在字符分割阶段，利用垂直投影算法确定验证码字符块个数，以及字符坐标点，再用颜色填充算法对验证码进行初步分割，根据分割后的字符块数量对粘连字符进行二次分割；在识别阶段，我们对 LeNet-5 网络进行了改进，修改了输入层，并用全连接层替换了 LeNet-5 网络中的 C5 层，以此来对验证码字符进行识别；实验表明，对于非粘连验证码和粘连验证码，单张图片分割时间为 0.14 和 0.15 ms，分割准确率为 98.75%和 97.25%，识别准确率为 99.99%和 97.7%；结果表明，该算法对验证码分割和识别都有着很好的效果。

关键词：字符分割；颜色填充分割算法；粘连字符；字符识别；LeNet-5 网络；

Character Verification Code Recognition Based on Character Segmentation and LeNet-5 Network

ZHANG Jingxun[1,2] ZHANG Junhu[1], ZHAO Yubo[2], LI Hui[1]
(1. College of Information Science and Technology, Qingdao University of Science and Technology, Qingdao 266061, China;
2. Shandong institutes of industrial technology, Qingdao 250102, China)

Abstract: To solve the low efficiency and accuracy of traditional captcha recognition methods, a captcha processing scheme of segmentation before recognition is designed. In the preprocessing stage, median filtering is applied for the noise reduction, and Hough transform is used to correct the image characters. In the character segmentation stage, the vertical projection algorithm is used to determine the number of character blocks and their coordinates, and then the color filling algorithm is used for preliminary segmentation. Based on the number of segmented character blocks, a second segmentation for connected characters is performed. In the recognition stage, the LeNet-5 network is improved by modifying the input layer and replacing the C5 layer with a fully connected layer for character recognition. Experimental results showed that for the non-connected and connected captchas, the segmentation time for single image is 0.14ms and 0.15ms, respectively, with the segmentation accuracies of 98.75% and 97.25% and the recognition accuracies of 99.99% and 97.7%. The results show that the algorithm has a good performance for captcha segmentation and recognition.

Keywords: character segmentation; color filling segmentation algorithm; stuck characters; character recognition; LeNet-5 network

0 引言

验证码存在的目的是加强网站系统安全，防止恶意攻击[1]，因此网站系统中的验证码形式也种类繁多，不仅有基于文本的、基于音频的、基于视频动画的验证码，还有基于谜题的验证码等[2]，而对验证码识别技术的研究是是非常关键的一部分，因为验证码的识别技术可以衡量验证码生成算法的质量，并帮助改善验证码生成算法，以达到加强网络系统安全的目的。

在验证码识别领域[3]，字符型验证码[4]依旧是网站使用最多形式的验证码，因此字符型验证码的识别是一个重要的研究方向。通常，研究人员将字符识别算法的过程分为三个步骤，分别为字符定位、字符分割和字符识别[5]。而在识别字符型验证码的时，最主要的是字符分割与字符识别部分，因此单个字符的正确分割是影响字符型验证码识别准确率的关键步骤[6]。有效的分割验证码可以极大的提

收稿日期:2023-02-28； **修回日期**:2023-03-09。
基金项目:国家自然科学基金项目(61702295)。
作者简介:张敬勋(1997-)，男，山东聊城人，硕士研究生，主要从事图像处理、字符分割、字符识别方向的研究。
李 辉(1984-)，男，河南平顶山人，博士，副教授，主要从事计算机视觉方向的研究。
通讯作者:张俊虎(1974-)，男，山东青岛人，博士，副教授，主要从事人工智能方向的研究。
引用格式:张敬勋，张俊虎，赵宇波，等.基于字符分割和 LeNet-5 网络的字符验证码识别[J].计算机测量与控制，2023，31(7)：271-277.

高字符型验证码识别的准确率。在分割方法中，传统的颜色填充分割算法[7]是利用区域填充的思想对字符进行分割的，该方法对于车牌分割[8]和字符间不粘连的验证码有着很好的分割效果，但是对于相互之间粘连的字符无法进行准确分割。垂直投影方法[9]也是一种经典的分割方法，当字符没有扭曲，倾斜等情况时，能够完成很好的分割。

在识别方法中，早期的 OCR 识别技术、形状上下文以及模板匹配等识别技术，仅仅只能局限在识别简单字符。而随着时间的推移，支持向量机以及卷积神经网络的出现，虽然提高了验证码识别的准确率，但是在准确率上还依然有着很大的提升空间。因此，本文提出了一种先分割后识别的验证码处理方法，此方法先将验证码图片进行分割，分割之后再使用 LeNet-5 神经网络对验证码图片识别。结果表明，本文方法能够很好的提升字符验证码的分割以及识别准确率。

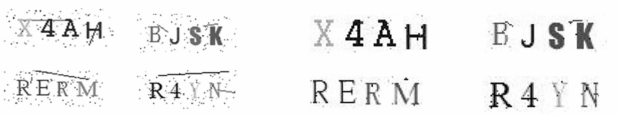
1 字符图像预处理

1.1 图像二值化以及去噪处理

在验证码的实际识别过程中为了准确有效的对验证码进行分割，我们需要对验证码进行预处理[10]，这其中包括灰度化，图片去噪，矫正以及二值化处理。首先，我们先将图片灰度化，而彩色图像转换为灰度图像时，需要计算图像中每个像素有效的亮度值[11]，根据重要性及其它指标，将 R，G，B 三个分量以不同的权值进行加权平均。其计算公式如式（1）所示：

$$Y = 0.299R + 0.578G + 0.114B \tag{1}$$

在图片灰度化之后，把图片进行去噪处理，本文选用中值滤波去噪，把验证码图像中一点的值用该点的一个邻域中各点值的中值代替，让周围的像素值接近真实值，从而消除孤立的噪声点。部分验证码经去噪处理后效果如图 1 所示。



原图像　　去噪后图像

图 1　部分验证码经去噪处理后效果

为了有利于进一步处理图像，使图像变得简单，而且让数据量减小，能突出字符有效信息，即突出所求目标字符的轮廓和特征，我们再对图像进行二值化处理。首先将 RGB 格式的图像转化为灰度图像，再把灰度图像进行二值化。

在图像预处理过程中，首先获得图像灰度继而使用迭代法阈值分割[12]来得到阈值，进而利用此阈值将灰度图像进行二值化，从而较为快速而且有效的来完成灰度图像的二值化。迭代法阈值分割的流程如下。

Step1：选择一个初始估计阈值 $T1$，预设值 ξ。

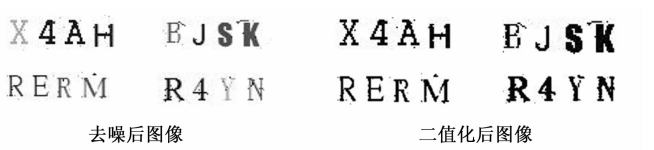
Step2：用 $T1$ 把给定图像分割。这样做会生成两组图像：$G1$ 由所有灰度值大于 T 的像素组成，而 $G2$ 由所有灰度值小于或等于 T 的像素组成。

Step3：对区域 $G1$ 和 $G2$ 中的所有像素计算平均灰度值 $\mu1$ 和 $\mu2$。

Step4：选择一个新阈值 $T1 = T2$，更新 $T2$，$T2 = (\mu1 + \mu2)/2$。

Step5：重复进行 Step2～Step4，直至 $|T2 - T1| > \xi$，输出 $T2$。

部分验证码二值化结果如图 2 所示。



去噪后图像　　二值化后图像

图 2　部分验证码二值化结果

1.2 倾斜矫正

为了网站安全，许多网站验证码中的字符都有或多或少的倾斜字符，为了方便验证码的下一步处理，我们需要对验证码字符进行倾斜矫正。

霍夫变换（hough transform）是图像处理中识别几何形状最常用的方法[13]。用 Hough 变换来检测图像的边缘，确定倾斜图像边缘的直线倾斜角，通过倾斜角的旋转后，可以获得矫正后的图像。倾斜矫正具体流程为：通过 Canny 边缘检测算法获得字符边缘，画出最小矩阵框，然后利用 Hough 变换计算倾斜角度，之后进行字符旋转矫正，最后依次找出所有最小矩阵框并矫正。验证码矫正结果如图 3 所示。

原图像　　标记矩阵框　　矫正后图像

图 3　验证码矫正结果

2 字符分割

目前大多数使用的验证码依旧为四字符的验证码，因此本文研究对象依旧是四字符的验证码图像。本文针对传统颜色填充算法的缺陷和粘连字符分割存在的难点，对传统的颜色填充分割算法进行改进。该算法首先使用垂直投影算法来判断验证码图片包含几个字符块，以及得出每个字符块的坐标点以及字符宽度。如果字符块数量等于四，则说明字符之间没有粘连，则直接使用颜色填充算法对验证码图像进行分割，如果字符块数量不等于四，则使用颜色填充算法分割之后再根据字符块的坐标点以及字符块宽度进行二次分割。本文算法分割流程如图 4 所示。

2.1 垂直投影算法

垂直投影算法就是对二值化后的验证码字符图像的黑色像素个数进行竖直垂直方向上的统计。设每个像素位置 (x,y) 对应的坐标为 $f(x,y)$，当像素点为白色像素点时，

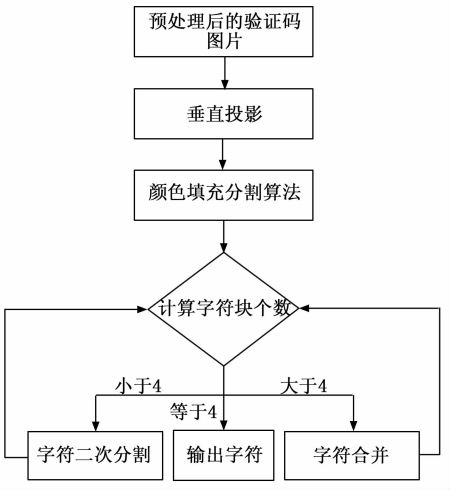


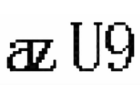
图 4　字符分割流程图

$f(x,y)=0$，为黑色像素点时，$f(x,y)=1$。则每列像素点个数为 $P(x)$，公式如(2) 所示，其中 H 为验证码图片高度。

$$P(x)=\sum_{y=0}^{H} f(x,y) \tag{2}$$

从 $x=0$ 开始，依次查找当 $P(x)=0$ 时 x 的取值，当第一次查找到 $P(x)=0$ 时，则记录此时的 x 坐标，并标记为字符的左边界；然后继续查找，当再次检测到 $P(x)=0$ 时，则记录此时的 x 坐标，并标记为字符的右边界，以此类推，直至将字符型验证码图片检测完毕，之后将字符宽度小于 2 的标记为噪点并去除。部分预处理后的验证码图片的垂直投影结果如表 1 所示。

表 1　部分预处理后的验证码图片垂直投影结果

验证码图片	字符坐标点	字符块数量
	[(10,18),(23,41),(43,52)]	3
	[(12,28),(34,44),(46,55)]	3
	[(7,17),(20,28),(33,44),(46,56)]	4
	[(10,45),(47,57)]	2

对表 1 中的第一张图片进行可视化，可视化结果如图 5 所示。

2.2　颜色填充算法 (CFS)

颜色填充分割算法 (CFS，color filling segmentation algorithm)，颜色填充又称为区域填充，这里的‘区域’是由像素点组成点阵形式的填充图形。一个区域由两部分组成，分别为内点和边界点，内点是指区域内部的所有像素点，在本文中是指验证码字符中的像素点，这些内部的像素点为同一个颜色。边界点是指区域外部的所有像素点，在本文中是指验证码字符之外的所有像素点，这些边界像素点为与内部像素点不同的颜色。具体的表现形式如图 6 所示。

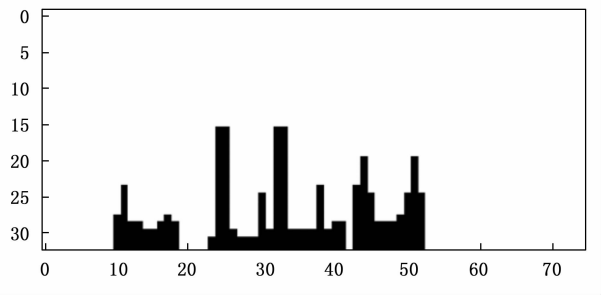


图 5　垂直投影可视化图

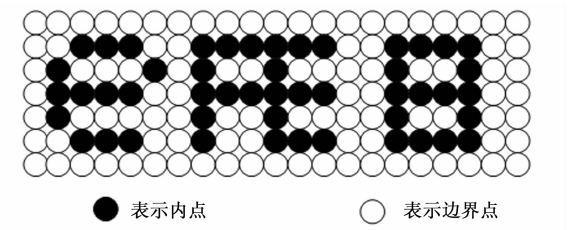


图 6　像素点的表现形式

本文所使用的连通方式为八连通方式，具体表现过程为将区域内的一点标记为种子点，给种子点标记一种颜色，然后通过八向连通的方式，用该种子点的颜色遍历完区域中所有像素点的过程。因本文处理后都是二值化图片，因此该种子点也就是一个黑色像素点（像素值为 0），用其他颜色标记，然后以这个种子点为中心，使用八向连通的方式，继续找相邻的黑色像素点，找到之后就将这个点赋予为和这个种子点相同的颜色，并从找到的黑色像素点开始递归寻找其余黑色像素点直至无相邻的黑色点结束，然后均标记为与此种子点相同的颜色。通过以上的一个递归过程就可以标记出一个颜色均为黑色的区域，然后再用其它几个不同的种子点，逐渐遍历，便可找到许多面积大小不同的区域，而区域面积较小则认为是噪点区域，因此，去除噪点区域外的其他区域则是分割出来的字符。本文基于的颜色填充分割算法采用栈结构来实现。基本流程如下：

Step1：种子元素入栈。

Step2：查看栈是否非空，非空则进行 Step3～Step6。

Step3：栈顶元素出栈。

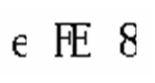
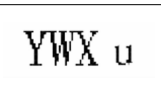
Step4：将出栈元素置成要填充色。

Step5：按八向连通的方式检查与出栈元素相邻的 8 个元素。

Step6：在 Step5 检查的像素中，如果某个像素点是内部点，而且没有填充颜色，则把该像素入栈。

取出部分预处理后的验证码图片使用颜色填充算法分割，部分分割结果如表 2 所示。

表 2 颜色填充分割算法结果

验证码图像	字符分割结果
	
	
	

2.3 字符二次分割

根据使用垂直投影算法计算出的宽度可知，使用颜色填充分割算法分割后粘连的字符要比单一字符的宽度值更大。在讨论粘连字符分割时，本文主要讨论的验证码是四个字符的验证码，因此分割后的可能会有以下情况，1）全部字符都相连，即分割后只有一个字符块，2）字符两两相连，分割后有两个字符块，3）字符部分连接，分割后有多个字符块。在以上三种情况中，我们每对字符完成一次分割，则用 OCR 的识别技术[14]进行一次判断，看分割出的图像是否是单个字符。直至对整个粘连字符图像处理完毕。

针对第一种情况，我们按照公式（3）对字符做出处理，因为全部字符相连，因此我们对字符做处理的结果会分出四个字符块。

$$W_i = \frac{W_{\max}}{4} \pm t, i = 1,2,3,4 \tag{3}$$

其中：$W_{\max}$ 代表最大字符块的宽度，W_i 代表每个字符块的宽度，其中 $W_1+W_2+W_3+W_4=W_{\max}$，t 代表对字符分割时每次移动的像素数，初始值为 1。

针对第二种情况，因为分割后会有两个字符块，因此对两个字符块分别处理，然后对处理后字符进行判断，看是否分割出正确字符。分割公式如（4）所示。

$$\begin{cases} W_i = \dfrac{W_{m1}}{2} + t, i = 1,2 \\ W_j = \dfrac{W_{m2}}{2} + t, j = 3,4 \end{cases} \tag{4}$$

其中：W_{m1} 代表第一个字符块的宽度，W_{m2} 代表第二个字符块的宽度，其中 $W_1+W_2=W_{m1}$，$W_3+W_4=W_{m2}$。t 代表对字符分割时每次移动的像素数，初始值为 1。

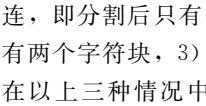
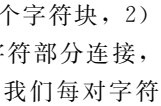
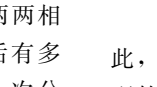
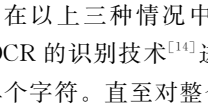
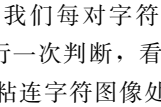
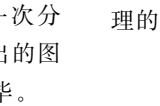
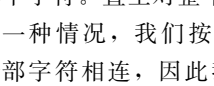
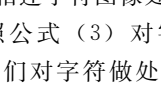
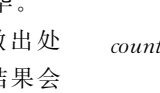
针对第三种情况，字符部分相连，即有可能是两个相连也有可能是三个相连，即最少有一个单个字符，我们令宽度最小的单个字符的宽度为 $W_{\min}$，然后我们对粘连字符按照公式（5）进行处理，然后检查分出的是否是单个字符。

$$\begin{cases} W_i = \dfrac{W_{\max}}{2} + t, i = 1,2, & \text{if} \quad W_{\max} > 2W_{\min} \\ W_i = \dfrac{W_{\max}}{3} + t, i = 1,2,3 & \text{if} \quad W_{\max} > 3W_{\min} \end{cases} \tag{5}$$

其中：$W_{\max}$代表相连字符块的宽度，$W_{\min}$代表宽度最小字符块的宽度，对于式（5）中的第一个式子，$W_1+W_2=W_{\max}$，对于公式（5）中的第二个式子 $W_1+W_2+W_3=W_{\max}$，t 代表对字符分割时每次移动的像素数，初始值为 1。

对粘连部分验证码二次分割处理以后部分验证码结果如表 3 所示。

表 3 部分验证码二次分割处理字符以及分割结果

验证码图像	处理字符	分割结果
		
		
		

2.4 字符合并

在粘连字符分割过程中，也有可能导致字符断裂，因此，我们需要对断裂字符进行合并。以下为对断裂字符处理的流程。

Step1：计算每个字符块的中线坐标为 $x_1, x_2, \cdots, x_n$。

Step2：计算字符块的最小宽度 $W_{\min}$ 和字符块数量 *count* 以及计算相邻字符块中线之间最小距离 $L_{\min}$。

Step3：将 $L_{\min}$ 与 $W_{\min}$ 进行对比，如果 $L_{\min}$ 小于 $W_{\min}$，则将该字符块与其后相邻字符块进行合并，即更改该字符的右边界，使该字符的右边界等于其后相邻的字符右边界，之后将 *count* 减 1。

Step4：重复进行步骤一到步骤三，直至垂直投影后的 $count \leqslant 4$。

3 字符识别

随着深度学习[16]的发展，卷积神经网络（CNN）作为一种深层次的神经网络，被应用在许多领域，如目标检测[17]、图像分割和识别[18]和 CAPTCHA 识别[15]。而在 CAPTCHA 识别领域表现出了优异的性能，其性能远远优于传统的机器学习方法[19]。与传统方法相比，卷积神经网络中的卷积层有着很强的提取特征的能力，降低了人工处理的要求和复杂的预处理流程。在 1998 年，Lecun 等[20]人在神经网络领域有了开创性工作，即提出了 LeNet-5 网络结构，这也正式开启了卷积神经网络时代。他首先应用于手写数字集的识别，并取得了很好的效果，之后在许多学者的改进下，又应用在车牌识别，压印字符识别等领域。

LeNet-5 网络结构包括 7 个层，分别为一个输入层、三个卷积层、两个池化层、一个输出层。LeNet-5 网络的模型结构如图 7 所示。

在图 7 中，C1 是一个具有六个特征图的卷积层。每个特征图都由 5×5 的卷积核采样步长为 1。特征图的大小为 28×28，可以防止输入从边界溢出。C1 包含 156 个可训练参数和 122 304 个连接。S2 是一个具有六个特征图的 14×

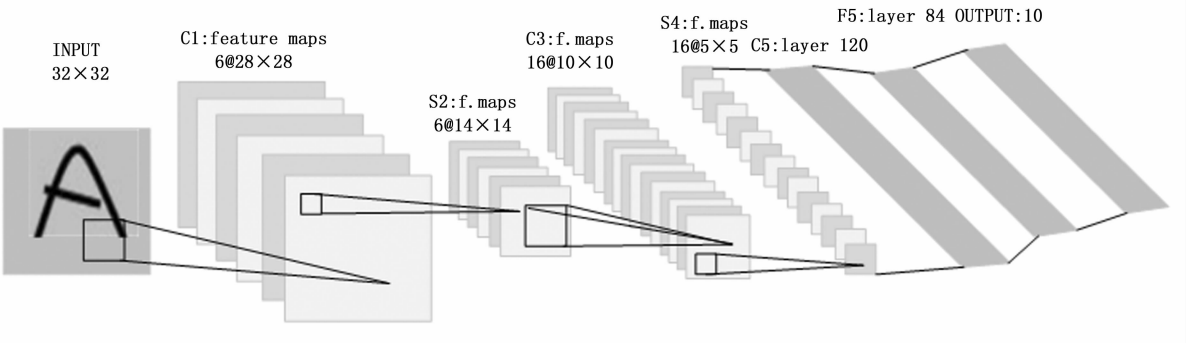


图 7　LeNet-5 模型结构

14 大小的子采样层。每个特征图中的每个单元都与 C1 中相应特征图的 2×2 邻域相连。2×2 的感受野是不重叠的，因此 S2 中的特征图行数和列数比 C1 中的特征图减半。S2 具有 12 个可训练参数和 5 880 个连接。C3 层为卷积层，输入为 S2 中所有 6 个或几个的特征图组合，使用 5×5 的卷积核采样，包含 6 张 10×10 大小的特征图和 1516 个可训练参数以及 156 000 个连接；S4 层为池化层，采样方式为 4 个输入相加，乘以一个可训练参数，再加上一个可训练偏置。结果通过 sigmoid 函数输出。包含 16 张 5×5 大小的特征图和 32 个可训练参数以及 2 000 个连接；C5 为卷积层，输入为 S4 层的全部 16 个单元特征图，卷积核大小 5×5，为包含 120 张 1×1 大小的特征图和 48 120 个可训练参数以及 48 120 个连接；F6 为全连接层，计算输入向量和权重向量之间的点积，再加上一个偏置，结果通过 sigmoid 函数输出。包含 84 个神经元和 10 164 个可训练参数；输出层采用的是径向基函数（RBF）的网络连接方式，输出为 10 个神经元。

RBF 的计算公式如公式（6）所示：

$$y_i = \sum_j (x_j - w_{ij})^2 \tag{6}$$

在公式（6）中，x_j 为全连接层中第 j 个神经元；y_i 为输出层第 i 个神经元；w_{ij} 为全连接层第 j 个神经元与输出层第 i 个神经元之间的权值。

因为 LeNet-5 网络结构最早是针对手写数字识别产生的，其 10 个输出神经元分别对应着 0～9 这 10 类数字，但验证码字符比手写数字更为复杂，识别难度更大，因此我们对传统的 LeNet-5 网络进行改进，使其能够更好的用于识别验证码字符。

由于验证码图片中包含多个字符，所以 LeNet-5 网络中输入层的 32 * 32 像素的大小便不能对验证码字符图片进行有效的识别，因此，我们将输入层的输入大小改为 40 * 30 像素大小，能够更好的符合字符验证码图像的长宽比，也能避免归一化为 32 * 32 大小时，而产生的过多信息丢失。

在 LeNet-5 网络中，C5 层的训练参数数量为 48 120 个，占整体训练数量的 80%。大大增加了训练量以及拖慢训练速度，因此将 LeNet-5 网络模型中的 C5 层去掉，改为 F5 全连接层，并在 F5 全连接层前进行扁平化处理，这样能够将模型总体的训练参数数量减少，提高了算法的运算效率。

本文设计的网络模型结构参数信息如表 4 所示。

表 4　卷积神经网络模型结构

网络结构	输入格式	输出格式
输入	40 * 30	————————————
卷积	40 * 30	40 * 30 * 32
池化	40 * 30 * 32	20 * 15 * 32
卷积	20 * 15 * 32	20 * 15 * 64
池化	10 * 7 * 64	5 * 3 * 64
扁平化	5 * 3 * 64	960
全连接	960	120
全连接	120	84
输出	84	4

本文设计的网络模型包括两个卷积层，两个池化层，和两个全连接层，其中学习率为 0.001，卷积核大小为 3 * 3，填充大小为 1，步长为 1，输入图片大小为 40 * 30，识别过程如图 8 所示。

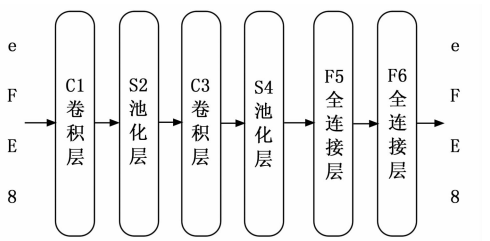


图 8　LeNet-5 网络识别过程

4　实验结果与分析

本文使用 python 语言，运行环境是在 CPU 上运行，是基于 Intel（R）Core（TM）i5-6200U 的处理器，操作系统为 Windows10，编译环境为 python3.7 版本。

本文数据集均来自生成和网络采集，其中由 Captcha 库生成 20 000 张字符型验证码图片，网络采集 2 000 张字符型验证码图片，其中包括粘连字符型和非粘连字符型验证码图片。验证码图片内容包含 0～9 数字及 A-Z 的 26 个大小写英文字符混合。

为了验证算法的优越性，根据验证码字符是否粘连将数据集分为两类，分别为粘连字符验证码和非粘连字符验证码。每类的数据集均为 11 000 张验证码图片，第一类中的验证码字符为较为简单的没有粘连字符验证码；第二类中的验证码字符均为复杂的粘连字符验证码。每类验证码的字符粘连情况如表 5 所示。

表 5 每类验证码的字符粘连情况

类别	验证码字符粘连情况	数据集数量(张)
第一类	不粘连字符	11 000
第二类	粘连字符	11 000

首先为了更好的验证本文分割算法的有效性，我们分别从两类数据集中拿出 1 000 张图片做分割实验，与垂直投影分割算法，颜色填充分割算法进行对比，计算字符分割算法的准确率以及分割所用时间。本文字符分割准确率公式如（6）所示：

$$Acc = \frac{1}{S}\sum_{i=0}^{S} Y_i \tag{6}$$

其中：Acc 代表字符分割准确率，S 为字符总个数，Y_i 代表分割正确的字符。各种算法分割准确率以及分割时间如表 6 所示。

之后，我们再用卷积神经网络对字符验证码进行识别实验，将数据集按照 8：1：1 的比例分为训练集、测试集和验证集，然后放到卷积神经网络中。最终模型第一类验证码训练和测试的准确率与损失曲线如图 9 以及图 10 所示。第二类验证码训练和测试的准确率与损失曲线如图 11 以及图 12 所示。

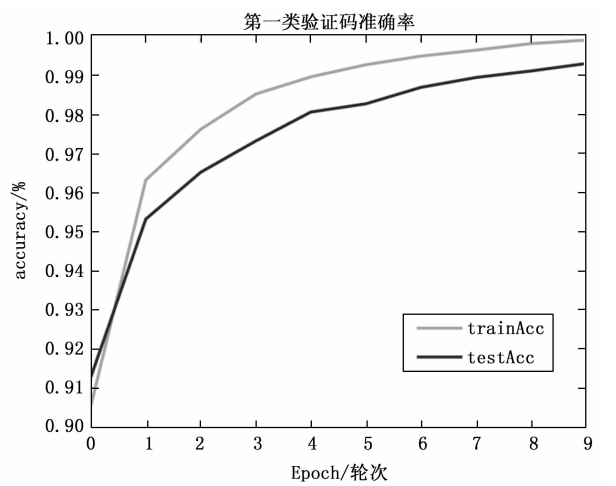


图 9 第一类验证码训练的准确率

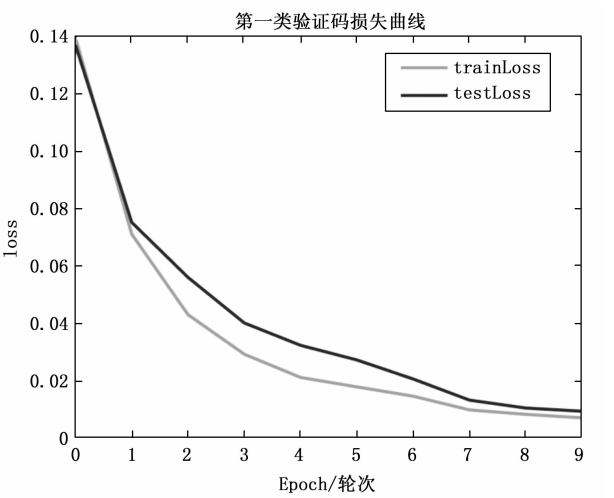


图 10 第一类验证码训练的损失曲线

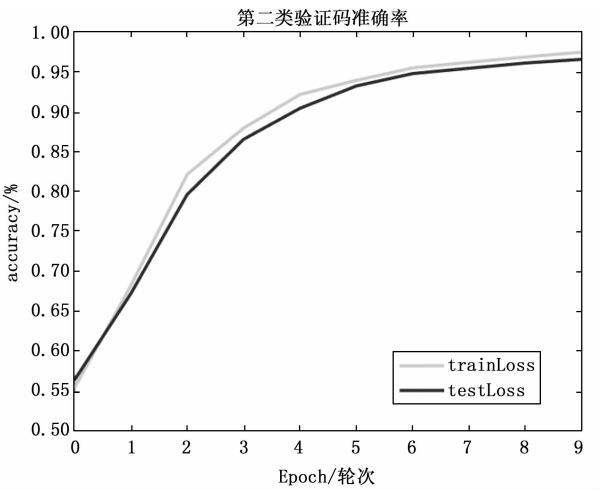


图 11 第二类验证码训练的准确率

从图 9、图 10、图 11 以及图 12 可以看出，无论是对于第一类无粘连字符的验证码，还是对第二类有粘连字符的验证码，本文的方法都具有很好的识别效果。之后，在测试集上对两类验证码进行实验，得到结果为第一类非粘连字符验证码，识别准确率达到 99.9%，对于第二类粘连字符的验证码，识别准确率可以达到 97.7%。除此之外，为了进一步评估本文方法对字符验证码识别的优越性，本文与参考文献中的其它验证码识别方法进行了对比，本文实验结果和其它模型对比结果如表 7 所示。

从表 7 可以看出，无论是对于第一类简单的非粘连验证码，还是对于第二类复杂的粘连验证码，本分对验证码的

表 6 字符分割实验对比

类别	样本数	颜色填充分割算法		垂直投影分割算法		本文方法	
		分割准确率	单张图片分割时间/ms	分割准确率	单张图片分割时间/ms	分割准确率	单张图片分割时间/ms
第一类	1 000	94.37%	0.12	97.41%	0.11	98.75%	0.14
第二类	1 000	57.5%	0.11	79.3%	0.13	97.25%	0.15

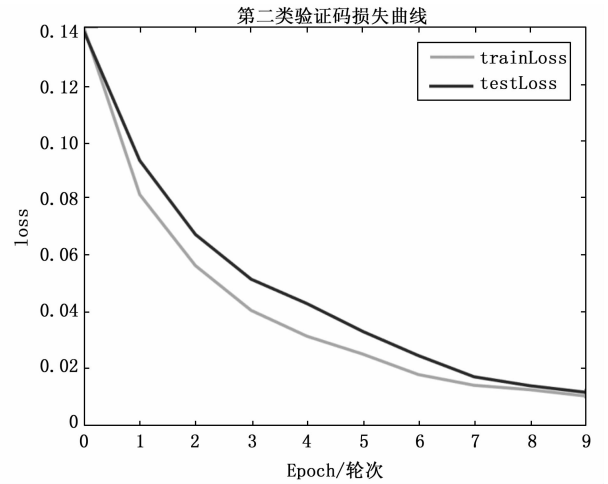


图 12　第二类验证码训练的损失曲线

表 7　各种算法识别正确率对比

类别	识别准确率			
	本文	滑动窗口＋CNN[21]	颜色填充＋CNN[22]	FRN[23]
第一类	99.9%	84.35%	94%	96.77%
第二类	97.7%	83.89%	76%	96.63%

处理方法均有不错的效果。其中对于第一类简单非粘连的验证码，效果有明显提升，对于第二类复杂的验证码，也比识别率最好的 FRN 效果要好。

5　结束语

本文针对传统颜色填充算法的缺陷和粘连字符分割存在的难点，对传统的颜色填充分割算法进行改进，最后再引进 LeNet-5 神经网络对验证码字符进行识别。实验结果表明，本文算法不仅适用于大部分非粘连字符验证码，而且还能够解决传统颜色填充算法无法处理的粘连字符分割问题，但是，对于背景较多干扰线的字符验证码，本文算法可能会出现误分割。

最后，本文在分割算法之后引入改进后的 LeNet-5 神经网络对验证码字符图片识别，这样极大的减少了卷积神经网络的参数，也能起到不错的字符识别效果。实验结果表明，本文提出的方法对验证码的分割和识别起到了良好的作用，提高了字符分割的准确率，也减少了卷积神经网络的参数，提高了运行速率，以及验证码识别准确率。

参考文献：

[1] 周　靖. 探讨验证码 [J]. 办公自动化，2021，26 (4)：22－23，40.

[2] KUMAR M，MUNISH KUMAR. A Systematic Survey on CAPTCHA Recognition：Types，Creation and Breaking Techniques [J]. Archives of Computational Methods in Engineering，2021：1107－1136.

[3] 张　锐，蔡艳林，陈夏裕，等. 验证码的识别与改进 [J]. 电脑编程技巧与维护，2021 (5)：117－119.

[4] 范亚如，徐鹏飞. 字符验证码识别技术的研究 [J]. 安徽电子信息职业技术学院学报，2020，19 (5)：15－18.

[5] 闫晓燊，高　强，朱思萌，等. 亮度不均匀低质量图像中压印字符分割方法 [J]. 计算机工程与应用，2021，57 (8)：185－191.

[6] CHELLAPILLA K，LARSON K，SIMARD P Y，et al. Building segmentation based human-friendly human interaction proofs (HIPs) [J]. HIP，2005，3517：1－26.

[7] 田超雄. 文本类验证码识别方法研究 [D]. 西安：西北大学，2019.

[8] 焦慧华. 基于垂直投影分割法的车牌图像的字符分割研究 [J]. 安徽电子信息职业技术学院学报，2021，20 (6)：17－20.

[9] 冉令峰. 基于垂直投影的车牌字符分割方法 [J]. 通信技术，2012，45 (4)：89－91，98.

[10] 陈志强. 复杂背景下字符图像的预处理及分割算法研究 [D]. 北京：北京邮电大学，2013.

[11] 郭　丽. 基于 RGB 颜色空间的彩色图像灰度化算法研究 [D]. 西安：陕西师范大学，2017.

[12] 杨伟婷，李保育，左文斌. 基于机器视觉的图像处理方法 [J]. 信息技术与信息化，2021 (7)：143－145.

[13] 闵　锋，吴　涛，陈智羽. 接触网支柱号牌定位与字符分割方法 [J]. 计算机工程与设计，2020，41 (3)：789－794.

[14] QI LI，WEIHUA An，ANMI ZHOU，et al. Recognition of Offline Handwritten Chinese Characters Using the Tesseract Open Source OCR Engine [C] // International Conference on Intelligent Human-machine Systems & Cybernetics. IEEE，2016：2－5.

[15] LIN D，LIN F，LV Y，et al. Chinese Character CAPTCHA Recognition and performance estimation via deep neural network [J]. Neurocomputing，2018，288 (may2)：11－19.

[16] HUANG K，HUSSAIN A，WANG Q F，et al. Deep Learning：Fundamentals，Theory and Applications [J]. 2019：111－138.

[17] LI B，FU H. Real Time Eye Detector with Cascaded Convolutional Neural Networks [J]. Applied Computational Intelligence and Soft Computing，2018：1－8.

[18] THOBHANI A，GAO M，HAWBANI A，et al. CAPTCHA Recognition Using Deep Learning with Attached Binary Images [J]. Electronics，2020，9 (9)：1522.

[19] MA Y，ZHONG G，LIU W，et al. Neural CAPTCHA networks [J]. Applied Soft Computing，2020，97 (2)：106769.

[20] LECUN Y，BOTTOU L，BENGIO Y，et al. Gradient-based learning ap-plied to document recognition [J]. Proceedings of the IEEE，1998，86 (11) ：2278－2324.

[21] 杨伟婷，李保育，左文斌. 基于机器视觉的图像处理方法 [J]. 信息技术与信息化，2021 (7)：143－145.

[22] 雷　蕾. 基于卷积神经网络的文本验证码自动识别 [D]. 西安：西安电子科技大学，2017.

[23] DUAN C，ZHANG R，QING K. Feature Refine Network for Text-Based CAPTCHA Recognition [C] // In Proceedings of the 10th International Conference on Image and Graphics. Beijing，China. Springer，2019：64－73