文章编号：2096-1472(2019)-06-01-04
DOI:10.19644/j.cnki.issn2096-1472.2019.06.001

# 基于Python和CNN的验证码识别

晋大鹏，张天心，刘 涛
(上海理工大学，上海 200093)

**摘 要**：针对目前互联网上关于页面自动登录环节出现的难点，由于部分登录界面有验证码的存在，自动登录的时长被增加，并且有的验证码难以识别，这就提出了基于Python和卷积神经网络(CNN)相结合的验证码识别。首先本文对三千多张验证码的样本集进行图片预处理，分别有灰度化处理、二值化处理和去噪点处理三步操作。然后利用三个池化层和一个全连接层的结构设计卷积神经网络，随后训练样本集，并对随机的十个样本进行预测。

**关键词**：验证码；Python；二值化；卷积神经网络
**中图分类号**：TP315 **文献标识码**：A

# Verification Code Recognition Based on Python and CNN

JIN Dapeng,ZHANG Tianxin,LIU Tao
( *University of Shanghai for Science and Technology*,*Shanghai* 200093,*China*)

**Abstract:**The paper focuses on the current difficulties in the automatic login of pages on the Internet.Due to the existence of verifications code in some login interfaces,the duration of automatic login is increased,and some verification codes are difficult to identify.Accordingly,based on Python and Convolutional Neural Network (CNN),a combined identification of verification codes is proposed in this study.Firstly,this paper preprocesses the sample set of more than 3,000 verification codes,including the three processing steps of graying,binarization and denoising.The Convolutional Neural Networ is then designed using three pooled layers and a fully connected layer structure,followed by training the sample set and predicting ten random samples.

**Keywords:**verification code;Python;binarization;Convolutional Neural Network

## 1 引言(Introduction)

伴随着互联网行业的兴起，人们的生活工作也越来越便利，很多需要大量工作量、计算量的工作渐渐被计算机所取代，在减轻人类压力的同时，也给了研究人员继续研究机器学习的动力。

验证码本身是用来区分服务对象是人还是计算机脚本的一种公共全自动程序，当前大多数浏览器都具有记住账号密码的功能，然而为了防止频繁的自动登录，减轻服务器端的压力，并在一定程度上防止机器人、外挂等非法程序的攻击，验证码就被研发了出来。

卷积神经网络(CNN)是受对猫视觉皮层电生理研究启发所提出的，Yann Lecun最早将CNN用于手写数字识别这一方向。时至今日，CNN在多个方向都有应用，例如，语音识别、人脸识别、通用物体识别、运动分析、自然语言处理甚至脑电波分析。

## 2 图片预处理(Image preprocessing)

### 2.1 灰度化

一般来说，我们获取到的图片都是彩色的图像，它的类型主要有RGB和CMYK两种，其中RGB彩色图像是由红色、绿色和蓝色组成；CMYK则是由青C、品M、黄Y和黑组成。识别图片中最重要的信息是梯度，梯度就代表着边缘，这是图片最本质的部分，而计算梯度就需要进行灰度化处理。灰度化就是将彩色的，含有很多噪点的图像转化为灰度图像的过程，也叫作灰度化处理[1]。在一般RGB模型中，如果R=G=B=某个定值时，其彩色就表示一个等于特定值的颜色，那个定值叫灰度值。因此，灰度化处理之后的图像中任意单个像素仅需要一个字节用来存取灰度值，0—255就是其中灰度值的取值范围。R、G、B三个分量决定了彩色图像

中的每个像素的颜色，每个分量又有255个值可以取，因此这样一个像素点可以有255*255*255=16581375的颜色的变化范围。与之对应的是，灰度图像的一个像素点仅有255种变化范围，因此在进行图像处理时，一般会先将各种格式的图像转变为灰度图像，以便之后对图像的计算量减轻[2]。灰度图像的描述和彩色图像一样仍然反映了整幅图像的整体和局部色度和亮度等级的分布和特征。对于灰度化的方式有分量法、最大值法、平均值法和加权平均值法等，以平均值法为例，求出R、G、B这三个分量的在彩色图像中的亮度，并根据三个分量的亮度求简单的平均值，将计算出来的值输出进而得到所需要的灰度图。其实现的表达式如下：$Gray(i,j)=(r(i,j)+g(i,j)+b(i,j))/3$。

### 2.2 二值化处理

任意图像的像素一般在二维空间中都会对应一个特定存在的位置，并且包含一个或者多个与该像素相关的采样值组成数值。通过对图片进行了灰度化处理后，还需要再把获取的灰度图像进行再处理，也就是二值化操作[3]。对于二值化，其目的是在不改变图片主题内容和轮廓的基础上使图片变得简单，并使需要操作的数据量减小，这样便于图像的再处理。对灰度图像进行二值化再处理一般最常用的方法是阈值法[4]，就是利用图片中目标和背景的差异，从而将图像分别定义为两个不相同的级别，选定一个恰当的阈值，从而确定某个像素到底是目标还是背景，进而获得二值化处理之后的图像。

二值化处理的基本过程：

(1)首先对初始的图像进行中低通滤波操作，也就是对图像的预处理，从而起到降低或者去除噪声的作用。

(2)用算法确定最佳阈值$T$。

(3)每当像素的灰度值大于或等于该阈值的都设置成255，小于这个阈值的全部设置成0。这样处理后的图像就只有黑白两种颜色，进而就将灰度范围分割成了目标和背景两个大类，最终就实现了对图像的二值化操作。

其中，公式为：$g(x,y)=\begin{cases}255(白)\ f(x,y)\geq T\\ 0(黑)\ f(x,y)<T\end{cases}$。

其中，最佳阈值的选取，一般有五种方法：(1)双峰法；(2)P参数法；(3)最大类间房差法(Otsu、大津法)；(4)最大熵阈值法；(5)迭代法(最佳阈值法)。

在这里本文选取迭代法。迭代法是利用类似于数学归纳法的无限逼近的思想实现的，可以总结为五个步骤：

(1)初始化。选取一个最初的阈值$T(j)$，一般来说我们选择整体图像的平均灰度值用作初始阈值。其中$j$为迭代的次数，开始是$j$的值为0。

(2)分割。以初始阈值$T(j)$为分割函数，分割图像得到$C_1^{(j)}$和$C_2^{(j)}$两个大区域。

(3)计算平均灰度值。利用以下公式计算平均灰度值，其中$N_1^{(j)}$、$N_2^{(j)}$代表当进行到第$j$次迭代时，$C_1$和$C_2$两个区域内的像素点数目，$f(x,y)$代表图像中点$(x,y)$处的灰度值。

$$u_1^{(j)}=\frac{1}{N_1^{(j)}}\sum_{f(x,y)\in C_1^{(j)}}f(x,y)$$

$$u_2^{(j)}=\frac{1}{N_2^{(j)}}\sum_{f(x,y)\in C_2^{(j)}}f(x,y)$$

(4)计算门限值。利用以下公式计算，公式中变量意义同上。

$$T(j+1)=\frac{u_1^{(j)}+u_2^{(j)}}{2}$$

(5)重复迭代。另其中的$j=j+1$，重复迭代步骤(1)—(3)，知道其中的$T(j+1)$和$(j)$的差值小于其中的规定值。

### 2.3 求连通域面积去噪点处理

当图片经过了灰度化和二值化后，原始图片数据上一般只存在噪点和干扰线了，这时候就需要进行降噪处理。一般来说，遇到较大的早点，会先去求它的面积，因为字符像素(pixel)大部分时相互连通的，因此求出每个相互连通的黑色点的个数，如果个数超过一定值，那么证明这一片像素集很可能是字符的部分。反之，若一个连通域内像素集个数很少，那么基本就可以确定这一片像素集是噪点，应当去除。对于求连通域的面积，有很多API可以用，其主体思路就是先求出连通域的外部轮廓，随后用指定的形状进行拟合，求出那个连通域的面积。但这种方法的误差较大，在这里我使用了泛水填充法[5]，API为floodFill(Mat,cvPoint($i$,$j$),cvScalar(color))；其中Mat为图片的矩阵对象，cvPoint($i$,$j$)为图片中位置为$(i,j)$的一个点，cvScalar为颜色对象，该函数意义是将与坐标为cvPoint($i$,$j$)连通的所有点的颜色都改为cvScalar(color)，这个过程就像往一张纸上滴一滴水，水泛染的样子，因此叫做泛水填充法。其具体步骤如下：

(1)计算时，每扫描到一个灰度值为0的黑点时，就将与该点连通的所有点灰度值改为1，因此该连通域的点都不会再做二次计算了。下一个灰度值为0的点，其所有连通点改为2，这样一次迭代，直到扫描完所有点。

(2)再次扫描所有点，统计各个灰度值所对应的个数，各个灰度值的点的个数对应该连通域的大小，并且由于不同连通域的灰度值不同，因此每个点仅不重复的计算一次。这样就统计到了每个连通域的大小，再根据预设的阈值，判断该连通域的大小是否低于阈值，如果低于，则判定其为噪点，该算法适合检查较大的噪点。

基于以上处理之后的图片，可以在保留原始图片信息的前提下，将数据预处理，使样本集体积缩小，便于后面的卷积神经网络的训练和预测。

## 3 利用CNN进行样本训练(Sample training using CNN )

### 3.1 卷积神经网络概述

卷积神经网络是近几年发展起来，被不断重视的一种用于高效识别的网络引擎[6]。20世纪60年代，Hubel和Wiesel研究猫脑皮层中用于局部敏感，以及方向选择的神经元时发现了其独特的网络结构可以有效降低反馈神经网络的复杂性，继而提出卷积神经网络(Convolutional Neural Networks，CNN)[7]。

CNN由三种不同的层组成，分别是：卷积层、池化层、密集层或全连接层。

卷积层：假设某张图片有5*5个像素，其中1代表白，0代表黑，这幅图就被视为是一张5*5的单色图像。现在用一个由随机的0和1组成的3*3矩阵去和图像中的字区域做乘法，每次迭代移动一个像素，这样该乘法就会得到一个全新的3*3矩阵，这个矩阵被叫作滤波器[8]，它的任务是提取图像特征，它使用一种优化算法来决定这个3*3矩阵中具体的0和1。在神经网络的卷句层中需要使用许多这种滤波器来提取多个特征，这个3*3矩阵的每一个单个步骤被称作步幅[9]。

池化层：池化层主要使用不同的函数为其中的输入矩阵降维。一般情况下，最大池化层出现在卷积层之后。池化层使用2*2矩阵，利用与卷积层相同的方式处理图像，不过它的目的是为了给图像本身进行降维操作。

全连接层：全连接层位于之前一层和激活函数之间[10]。

### 3.2 模型训练

#### 3.2.1 图片信息代码化

首先，先输入图片的信息作为备用，通过图片属性可以得知是114*450像素的，最多包含6个字母，每个字母我们利用0和1来进行数字化表示，比如a可以数字化为100000000000000000000000000，b可以数字化为010000000000000000000000000，以此类推。基于以上的信息，我们在代码中将其表示为：

IMAGE_HEIGHT=114

IMAGE_WIDTH =450

MAX_CAPTCHA=6

CHAR_SET_LEN=26

#### 3.2.2 定义函数

定义一个从本文训练集中提取图片的函数，此处有一步预处理被省略，即将训练集图片重命名，以简化到只需获取验证码的名称和图片即可，其中以矩阵的形式返回图片[11]。再定义两个函数，一个将名字转换为向量，另一个将向量转换为名字。

代码如图1所示。

```
def get_name_and_image():
    all_image = os.listdir('J:/CNN/captcha4/')
    random_file = random.randint(0, 3429)
    base = os.path.basename('J:/CNN/captcha4/' + all_image[random_file])
    name = os.path.splitext(base)[0]
    image = Image.open('J:/CNN/captcha4/' + all_image[random_file])
    image = np.array(image)
    return name, image


def name2vec(name):
    vector = np.zeros(MAX_CAPTCHA*CHAR_SET_LEN)
    for i, c in enumerate(name):
        idx = i * 26 + ord(c) - 97
        vector[idx] = 1
    return vector


def vec2name(vec):
    name = []
    for i in vec:
        a = chr(i + 97)
        name.append(a)
    return "".join(name)
```

图1 定义函数的代码

Fig.1 Code for defining functions

#### 3.2.3 生成采样集

首先本文通过之前定义的get_name_and_image()函数得到的图片已经被以含布尔值的矩阵形式返回了，接下来通过语句1*(image.flatten())将其转换成1行114*450列的只含有0和1的矩阵，从而生成采样集。代码如图2所示。

```
def get_next_batch(batch_size=64):
    batch_x = np.zeros([batch_size, IMAGE_HEIGHT*IMAGE_WIDTH])
    batch_y = np.zeros([batch_size, MAX_CAPTCHA*CHAR_SET_LEN])

    for i in range(batch_size):
        name, image = get_name_and_image()
        batch_x[i, :] = 1*(image.flatten())
        batch_y[i, :] = name2vec(name)
    return batch_x, batch_y
```

图2 生成采样集的代码

Fig.2 Code for generating the sample set

#### 3.2.4 定义卷积神经网络结构

通过方案选择，本文采用3+1的结构，即三个卷积层加上一个全连接层来定义卷积神经网络结构。在每个卷积层中，都选择使用2*2的最大池化层和一个随机失活(dropout)层，其中卷积核尺寸选择的是5*5。在全连接层中，原始图片114*450的结果经过了三层池化层，其长和宽都被压缩了八倍，也就是15*57左右的大小。代码如图3所示。

```
def crack_captcha_cnn(w_alpha=0.01, b_alpha=0.1):
    x = tf.reshape(X, shape=[-1, IMAGE_HEIGHT, IMAGE_WIDTH, 1])
    # 3 conv layer
    w_c1 = tf.Variable(w_alpha * tf.random_normal([5, 5, 1, 32]))
    b_c1 = tf.Variable(b_alpha * tf.random_normal([32]))
    conv1 = tf.nn.relu(tf.nn.bias_add(tf.nn.conv2d(x, w_c1, strides=[1, 1, 1, 1], padding='SAME'), b_c1))
    conv1 = tf.nn.max_pool(conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
    conv1 = tf.nn.dropout(conv1, keep_prob)

    w_c2 = tf.Variable(w_alpha * tf.random_normal([5, 5, 32, 64]))
    b_c2 = tf.Variable(b_alpha * tf.random_normal([64]))
    conv2 = tf.nn.relu(tf.nn.bias_add(tf.nn.conv2d(conv1, w_c2, strides=[1, 1, 1, 1], padding='SAME'), b_c2))
    conv2 = tf.nn.max_pool(conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
    conv2 = tf.nn.dropout(conv2, keep_prob)

    w_c3 = tf.Variable(w_alpha * tf.random_normal([5, 5, 64, 64]))
    b_c3 = tf.Variable(b_alpha * tf.random_normal([64]))
    conv3 = tf.nn.relu(tf.nn.bias_add(tf.nn.conv2d(conv2, w_c3, strides=[1, 1, 1, 1], padding='SAME'), b_c3))
    conv3 = tf.nn.max_pool(conv3, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
    conv3 = tf.nn.dropout(conv3, keep_prob)

    # Fully connected layer
    w_d = tf.Variable(w_alpha * tf.random_normal([15 * 57 * 64, 1024]))
    b_d = tf.Variable(b_alpha * tf.random_normal([1024]))
    dense = tf.reshape(conv3, [-1, w_d.get_shape().as_list()[0]])
    dense = tf.nn.relu(tf.add(tf.matmul(dense, w_d), b_d))
    dense = tf.nn.dropout(dense, keep_prob)

    w_out = tf.Variable(w_alpha * tf.random_normal([1024, MAX_CAPTCHA * CHAR_SET_LEN]))
    b_out = tf.Variable(b_alpha * tf.random_normal([MAX_CAPTCHA * CHAR_SET_LEN]))
    out = tf.add(tf.matmul(dense, w_out), b_out)
    return out
```

图3 定义CNN结构的代码

Fig.3 Code that defines the CNN structure

#### 3.2.5 训练样本集

当把结构设计完成后，就可以开始训练样本集了，通过交叉熵(Cross Entropy)函数sigmoid_cross_entropy_with_logits()来比较其中的loss值[12]，并采用adam优化器来进行优

化。训练中，每一步的loss值都要输出，每100步的准确率也要输出。通过设置，本文这里将目标准确率设置为99%，即只有当样本训练准确率达到了99%后，才会结束训练。代码如图4所示。

```
def train_crack_captcha_cnn():
    output = crack_captcha_cnn()
    loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits=output, labels=Y))
    optimizer = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)

    predict = tf.reshape(output, [-1, MAX_CAPTCHA, CHAR_SET_LEN])
    max_idx_p = tf.argmax(predict, 2)
    max_idx_l = tf.argmax(tf.reshape(Y, [-1, MAX_CAPTCHA, CHAR_SET_LEN]), 2)
    correct_pred = tf.equal(max_idx_p, max_idx_l)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

    saver = tf.train.Saver()
    with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())

        step = 0
        while True:
            batch_x, batch_y = get_next_batch(64)
            _, loss_ = sess.run([optimizer, loss], feed_dict={X: batch_x, Y: batch_y, keep_prob: 0.5})
            print(step, loss_)

            # 每100 step计算一次准确率
            if step % 100 == 0:
                batch_x_test, batch_y_test = get_next_batch(100)
                acc = sess.run(accuracy, feed_dict={X: batch_x_test, Y: batch_y_test, keep_prob: 1.})
                print(step, acc)
                # 如果准确率大于50%,保存模型,完成训练
                if acc > 0.99:
                    saver.save(sess, "./crack_capcha.model", global_step=step)
                    break

            step += 1

train_crack_captcha_cnn()
```

图4 训练样本集的代码

Fig.4 Code for training sample set

训练结束后，得到模型输出文件，训练部分到这里就结束了。

### 3.3 模型预测

在模型训练成功的基础上，需要检测此模型的预测水平，这时本文随机找十张类似的图片作为预测图片进行预测，定义一个预测模型的函数crack_captcha()，代码如图5所示。

```
def crack_captcha():
    output = crack_captcha_cnn()

    saver = tf.train.Saver()
    with tf.Session() as sess:
        saver.restore(sess, tf.train.latest_checkpoint('.'))
        n = 1
        while n <= 10:
            text, image = get_name_and_image()
            image = 1 * (image.flatten())
            predict = tf.argmax(tf.reshape(output, [-1, MAX_CAPTCHA, CHAR_SET_LEN]), 2)
            text_list = sess.run(predict, feed_dict={X: [image], keep_prob: 1})
            vec = text_list[0].tolist()
            predict_text = vec2name(vec)
            print("正确: {}  预测: {}".format(text, predict_text))
            n += 1

crack_captcha()
```

图5 模型预测的代码

Fig.5 Model prediction code

预测结束后发现准确率在40%以上，但是细化到单个验证码，其准确率就可以达到，主要是因为训练集的准确率有99%，在这其中，如果调低keep_prob的值(这个值在代码中规定的是0.5，这个参数是用来控制着机器的拟合的)[13]，增加样本集的数量并增加卷积层，最后的准确率会更高。预测结果如图6所示。

```
正确: cochin  预测: cochin
正确: picaro  预测: picara
正确: hackee  预测: hacbee
正确: cattie  预测: cottie
正确: bathed  预测: barhed
正确: picaro  预测: picara
正确: cochin  预测: cochin
正确: baleen  预测: baleen
正确: baleen  预测: baleen
正确: hackee  预测: hacbee
```

图6 预测结果

Fig.6 Forecast result

## 4 结论(Conclusion)

通过对该模型的设计和预测，使用python和CNN可以有效地满足验证码识别的部分需求。考虑到以往利用支持向量机来识别验证码，卷积神经网络的代码更加轻量化。在对图片做了灰度化、二值化和去噪点操作后，通过对系统训练集准确率的设置，CNN可以保证该模型的高性能、高可用性和高复用性。当样本集数量越大，该模型可以更加准确地识别预测集中的验证码，对纯数字和数字加字母的验证码机制也同样适用。当然，它也存在一些不足之处，当样本集越大，虽然对识别准确率可以起到显著增加的效果，但是由于样本集过于庞大，其在训练模型的阶段所耗的时间也就会越久，其模型输出文件的体积也就会越大。当该模型对验证码的识别预测达到一定的准确率，那么页面自动登录的时间也就可以大大缩短了。

## 参考文献(References)


[1] 罗婷婷.两种改进的彩色图像灰度化算法研究[D].浙江工商大学,2014.

[2] 贺姣.基于色差模型的彩色图像灰度化算法研究[D].西安电子科技大学,2014.

[3] 王鹏.一种局部二值化方法及其应用[D].吉林大学,2007.

[4] 陈业慧,黄凯.基于OV7725二值化图像的无线实时传输[J].长沙大学学报,2018(02):39-41.

[5] 曹青媚,王雪莲.快速位移图像高阶累积量噪点检测技术[J].计算机仿真,2017(07):281-284.

[6] 张庆辉,万晨霞.卷积神经网络综述[J].中原工学院报,2017(03):82-86.

[7] 王嘉鑫,邹科文,陈义明.基于卷积神经网络的人脸识别[J].电脑知识与技术,2016(29):187-190.

[8] 范望,韩俊刚,苟凡,等.卷积神经网络识别汉字验证码[J].计算机工程与应用,2018(03):160-165.

[9] 贺桂娇.几种经典的图像边缘检测算子分析比较[J].计算机光盘软件与应用,2014(09):182-183.

[10] 欧先锋,向灿群,湛西羊,等.基于CNN的车牌数字字符识别算法[J].成都工业学院学报,2016(04):26-30.

[11] T.Lossau,H.Nickisch,T.Wissel,et al.Motion artifact recognition and quantification in coronary CT angiography using convolutional neural networks[J].Medical Image Analysis,2019(02):68-79.

[12] Huaqing Wang,Shi Li,Liuyang Song,et al.A novel convolutional neural network based fault recognition method via image fusion of multi-vibration-signals[J].Computers in Industry,2019(02):182-190.

[13] Saya Fujino,Taichi Hatanaka,Naoki Mori,et al.Evolutionary deep learning based on deep convolutional neural network for anime storyboard recognition[J].Neurocomputing,2019(03):393-398.


## 作者简介：


晋大鹏(1995-)，男，硕士生.研究领域：智能制造.

张天心(1994-)，男，硕士生.研究领域：金融管理和会计.

刘　涛(1995-)，男，硕士生.研究领域：智能制造.