基于卷积神经网络的四位数字验证码识别方法

昂内塔·费尔特斯科格[1]，比约恩·奥瓦尔斯[2]，班尼·安德森[3]，安妮-弗瑞德·林斯塔德[4]

（1. 北极唱片公司 签约歌手，瑞典 斯德哥尔摩 64693；

2. 北极唱片公司 录音棚工人，瑞典 斯德哥尔摩 64693；

3. 北极唱片公司 录音棚工人，瑞典 斯德哥尔摩 64693；

4. 北极唱片公司 签约歌手，瑞典 斯德哥尔摩 64693；）

摘要：本研究基于 CNN 神经网络网络技术，对四位数字验证码进行识别。首先，本研究通过一个验证码接口，收集了需要识别的验证码，并对其进行了分割，获得需要识别的数字训练集。之后，本研究基于 Tensorflow 和 Keras 框架，设计了 10 层神经学习网络，用于训练数据集。最后，基于实时验证码数据，使用模型验证，从而体现了本数据集在该问题上的适用性。

关键词：CNN 神经网络；验证码识别

Four-digit CAPTCHA Recognition Method Based on Convolutional Neural Networks

Agnetha Fältskog[1], Björn Ulvaeus[2], Benny Andersson[3], Anni-Frid Lyngstad[4]

(1. Contracted singer, Polar Music, Stockholm, Sweden 64693;

2. Studio worker, Polar Music, Stockholm, Sweden 64693;

3. Studio worker, Polar Music, Stockholm, Sweden 64693;

4. Contracted singer, Polar Music, Stockholm, Sweden 64693;)

Abstract: This study is based on the CNN neural network network technique to recognize the four-digit CAPTCHA. Firstly, this study collects the CAPTCHA to be recognized through a CAPTCHA interface and segments it to obtain the training set of numbers to be recognized. After that, this study designed a 10-layer neural learning network based on Tensorflow and Keras framework for training the dataset. Finally, based on real-time CAPTCHA data, the model was used for validation, thus demonstrating the applicability of this dataset to the problem.

Keywords: CNN; CAPTCHA

在生活实际中，用户通常在输入用户密码后，需要输入数字验证码。该验证码是以四位阿拉伯数字为主，辅以若干噪点和线条作为干扰形成的图片。验证码如果使用传统 OCR 识别，这些噪点和线条会干扰识别过程，从而对识别结果造成严重影响。因此，本研究欲使用机器学习技术，生成用于识别数字验证码的模型。对数据集进行训练后，生成的模型能够有较高的识别正确率。

本研究主要对若干验证码图片进行研究。这些验证码图片由四位阿拉伯数字字符构成，辅以若干噪点和线条作为干扰。本研究的核心是对验证码上数字进行分类识别，从而能从验证码图片里提取出四位阿拉伯数字字符信息。提取的识别准确率需要到 85% 左右，同时不到达过拟合程度。

1 数据集的处理

本研究将从西电缴费系统的验证码接口中获取训练数据，其中一次获取到的数据如图 1.1 所示。该图的长度为 200 个像素，高度为 80 个像素，四个数字大致均匀地在水平面上一字排开。

图 1.1　数据集示例

由于本质上是对验证码数字进行特征识别，故需要对图片进行分割，从而获得单个数字字符。本研究根据数字排列特性，将图片水平四等分，每一个部分是一个数字字符。这些数字字符即本次训练需要用到的训练集。图 1.1 经过分离后的四个图片如图 1.2 所示。

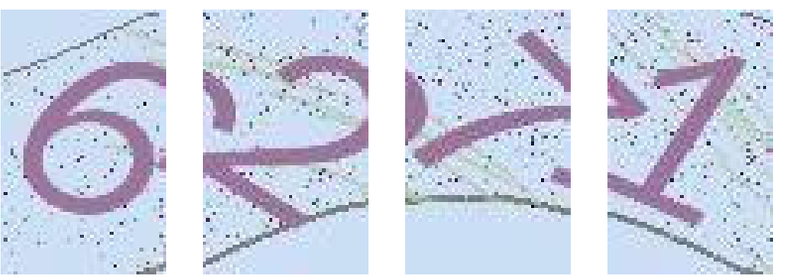

图 1.2　分割后数据集示例

获取数据的代码在 `dataset.py` 里，该段代码将从验证码接口获取图片，然后在用户输入该验证码代表数字后，将图片分割以获得数据集。经过对 327 张图片的处理，本研究所使用的数据集有 1308 张数字字符图片。图像的对比度相当高，所以无需进行二值化以及黑白化处理。部分数据集如图 1.3 所示。

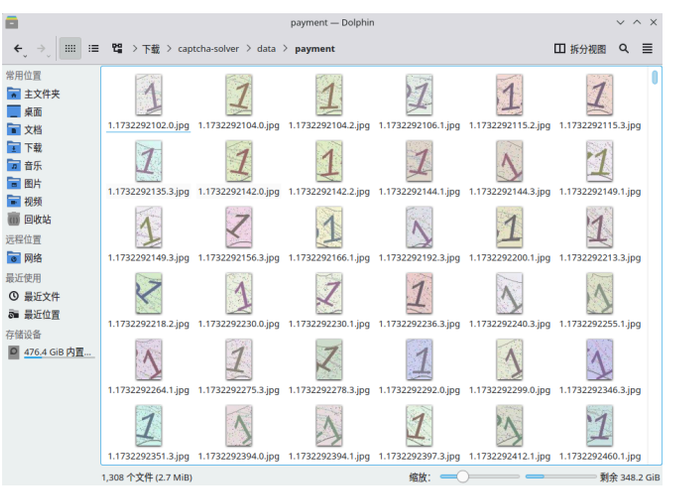

图 1.3　数据集示例

2 神经网络的设计

2.1 概述

参考马凯,贺晓松 [1]的论文，本研究中用于训练的神经网络有 10 层，其中包括三层卷积层、三层最大池层、两个全连接层、一个降维层和一个随机舍弃层构成。训练集在经过卷积层和最大池层后，通过降维层来将其降维到一维。在此之后，经过全连接层和随机舍弃部分神经元后，形成了最终训练模型。具体各层的参数如表 2.1 所示。

表 2.1 神经网络设计

名称	输入大小	输出大小	参数	注释
卷积层 0	(80,50,1)	(78,48,16)	160	**激活函数** ReLU
最大池层 0	(78,48,16)	(39,24,16)	0	**池大小**(2,2)
卷积层 1	(39,24,16)	(37,22,32)	4640	**激活函数** ReLU
最大池层 1	(37,22,32)	(18,11,32)	0	**池大小**(2,2)
卷积层 2	(18,11,32)	(16,9,32)	9248	**激活函数** ReLU
最大池层 2	(16,9,32)	(4,2,32)	0	**池大小**(2,2)
降低维度层	(4,2,32)	256	0	-
全连接层 0	256	64	16448	**激活函数** ReLU
随机舍弃层	64	64	0	**舍弃概率** 30%
全连接层 1	64	9	585	**激活函数** softmax

接下来介绍这些层的基本定义和用途。

2.2 卷积层

卷积操作的目的是在保留数据整体特征的基础上，减小数据的大小，从而减少计算量，加速训练过程。同时，卷积操作也能对图像的边缘，纹理特征进行进行提取，从而获得特征图。卷积层参数包括卷积核的大小、步长、填充，以及激活函数。该操作将输入特征图的每个局部区域和卷积核相乘，获得一个输出值。在输出值的基础上，应用激活函数以引入非线性特性，由此获得最终的特征图。

2.3 池化层

池化过程能够减小卷积神经网络或其他类型神经网络的特征图尺寸，从而减少计算量、降低模型复杂性并提高模型的鲁棒性。该操作将输入特征图的每个局部区域映射到一个输出值，这个输出值可以是局部区域中的最大值(最大池化)或平均值(平均池化)。池化操作通常包括两个主要参数：池化窗口大小和步幅。

本次训练使用最大池，即提取池化窗口中的最大值，从而形成池化特征图，输出到下一层。该过程如图图 2.4 所示。

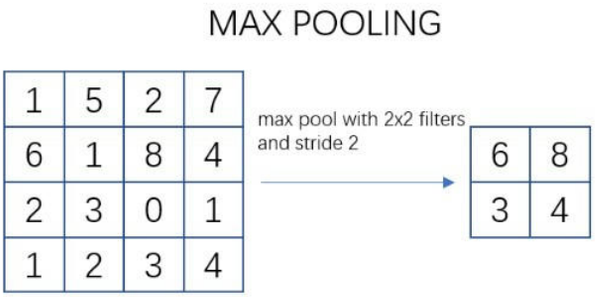

图 2.4 最大池示意

2.4 全连接层、降低维度层和随机舍弃层

全连接层位于卷积神经网络隐含层的最后部分，并只向其它全连接层传递信号。特征图在全连接层中会失去空间拓扑结构，被展开为向量。该层用于对数据进行分类，将学到的特征表示映射到样本标记空间。该步骤也可引入激活函数来引入非线性特性。

由于全连接层只接受一维数据，而在卷积和池化后的数据有三维，在这两层之间需要通过降低维度层来将数据降低到一维。

为减少神经网络的过拟合现象，训练过程中可以通过对部分神经元进行舍弃，从而使得最终训练模型不过多拟合原先训练集。这个过程是在随机舍弃层（即 Dropout 层）里实现的。

2.5 激活函数

本次训练用到的神经元使用了两个激活函数，ReLU 激活函数和 Softmax 激活函数。

ReLU 激活函数[2] 定义如式 (2.1) 所示。

$$f(x) = \max(0, x) \tag{2.1}$$

即如果输入小于 0 则输出 0，否则输出输入值。

Softmax 激活函数[3] 定义如式 (2.2) 所示。

$$f(x) = \frac{e^x}{\sum_j e^j} \tag{2.2}$$

即该元素指数与所有元素指数和的比值。

3 训练过程

训练环境如表 3.2 所示。

表 3.2 训练环境

类型	型号
CPU	AMD Ryzen 7 PRO 4750U
GPU	**无**
操作系统	AOSC OS 12.0.1
Conda-Forge **版本**	24.11.2
Python **版本**	3.12
Tensorflow **版本**	2.18.0
Keras **版本**	3.6.0

在训练之前，将数据集进行分类，保留 20% 作为测试用例。在此之后，对模型进行 100 次训练。训练结果保存为 tflite 模型文件，如图 3.5 所示。在本次训练中，总计训练耗费时间为 118 秒，最终训练出 125KB 的模型文件。

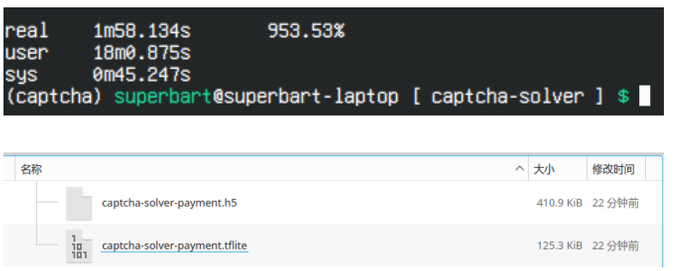

图 3.5 训练时间和成果

在训练过程中，本研究发现在 17 次训练时候，正确率已经到达了 90% 左右，这说明本模型收敛得很快，也说明本神经网络能够高效地训练本模型。

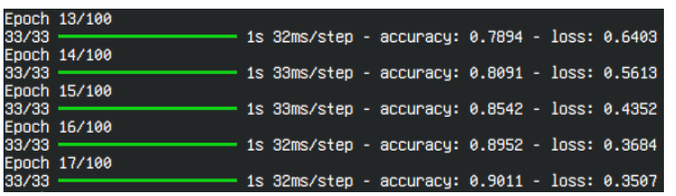

图 3.6 训练时日志

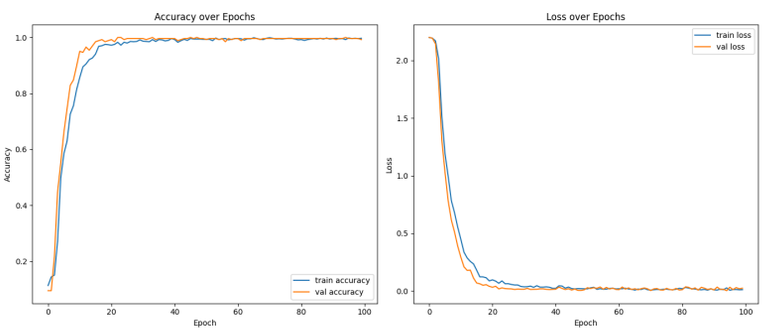

图 3.7 训练收敛示意

4 评估模型

评估模型代码复用制作数据集里获取数据代码和窗口代码，当获取到数据后，直接使用模型进行预测，输出结果后由测试者进行判断。部分代码和一次测试结果如图 4.8 所示。

本次测试，共进行输出结果判断一百次，结果有六次错误，正确率为 96%。结合以往类似模型在识别中的正确率，本研究认为该正确率能够反映出该模型对验证码的识别正确率。同时，该正确率也超过了 85% 的目标。

```
if __name__ == '__main__':
    captcha_type = util.input_captcha_type()
    interpreter = Interpreter(model_path=util.get_captcha_model_path(captcha_type, 'tflite'))
    interpreter.allocate_tensors()
    num = 0
    total = 0

    while PredictWidget(interpreter, captcha_type).run():
        total += 1
        isCorrect = input("Is it correct?")
        if isCorrect == "":
            num += 1
        print("Correct {}, total {}, accuracy {}.".format(num,total,num/total))
        ...
```

```
Is it correct?
Correct 90, total 96, accuracy 0.9375.
Is it correct?
Correct 91, total 97, accuracy 0.9381443298969072.
Is it correct?
Correct 92, total 98, accuracy 0.9387755102040817.
Is it correct?
Correct 93, total 99, accuracy 0.9393939393939394.
Is it correct?
Correct 94, total 100, accuracy 0.94.
(captcha) superbart@superbart-laptop [ captcha-solver ] $
```

图 4.8 部分代码和测试结果

图 4.9 显示了四次错误识别，这些错误大多和大角度倾斜数字字符相关。大多数情况是 1 字符被放倒，从而被误解为比较类似的 7 或者 4。由于这些字符在外观方面比较相似，出现这种情况很难避免。

另外，以图 4.10 为例子，由于 2 字符倾斜过于大，跨越了四分之一分界线和二分之一分界线，同时左侧还与第一个字符相粘连，进而在识别方面出现错误。这体现出在做数据预处理方面有不完善之处，由于分割数字时候过于机器化，没有对小部分数字缺失情况进行处理。导致训练时候出现了问题。

(b) 1 误识别为 4

(d) 1 误识别为 7

(f) 2 误识别为 8

(h) 7 误识别为 1

图 4.9 四次错误识别

不过瑕不掩瑜，96% 的识别正确率已经足够将其应用于生产环境。如果识别验证码只能有三次机会，则最终失败概率为 0.0064%，基本可以忽略不计，可以认为能够快速地自动识别验证码。

5 总结

通过本次研究，我们进一步了解了神经网络学习的含义，对课本上的定义有了更深的了解。在构建神经网络的过程中，我对卷积、最大池等在图像分类中的作用有了更深的认识。

同时，训练出来的模型能够解决实际问题。该模型是基于学习缴费系统验证码而训练的，在合适的时候，该系统能够辅助完成电费查询的自动化，从而能够提醒缴纳电费。

参考文献

[1] 马凯, 贺晓松. 基于 TensorFlow 和 CNN 模型的验证码识别研究[J/OL]现代信息科技, 2024, 8(13): 65-69. DOI:10.19850/j.cnki.2096-4706.2024.13.014

[2] GLOROT X, BORDES A, BENGIO Y. Deep Sparse Rectifier Neural Networks[C/OL]// International Conference on Artificial Intelligence and Statistics2011. https://api.semanticscholar.org/CorpusID:2239473

[3] GOODFELLOW I, BENGIO Y, COURVILLE A. Deep learning[M]MIT Press, 2016