

分 类 号： TP311

学校代码： 11460

学 号： 14131613

# 南京晓庄学院本科生毕业设计

## 基于卷积神经网络的图像识别研究

Realization of image recognition based on Convolution Neural  
Network.

所 属 学 院： 信息工程学院

学 生 姓 名： 刘邦龙

指 导 教 师： 王寅同

职 称： 讲 师

研究起止日期： 二〇一八年一月至二〇一八年六月

二〇一八年六月

## 学位论文独创性声明

本人郑重声明：

- 1.坚持以“求实、创新”的科学精神从事研究工作。
- 2.本论文是我个人在导师指导下进行的研究工作和取得的研究成果。
- 3.本论文中除引文外，所有实验、数据和有关材料均是真实的。
- 4.本论文中除引文和致谢的内容外，不包含其他人或其它机构已经发表或撰写过的研究成果。
- 5.其他同志对本研究所做的贡献均已在论文中作了声明并表示了谢意。

作者签名：

年 月

## 【摘要】

随着人工智能技术的高速发展，手写体字符识别技术在邮政编码、财务报表、表格录入、银行票据处理、办公自动化等方面都有着广泛的应用。手写体字符识别是属于图像识别领域下的一个分支，作为一种信息处理手段，手写字符识别有着广阔的应用背景和巨大的市场需求。

卷积神经网络是近年来提出的一种基于多个隐层的深层次神经网络模型，用来研究并处理该领域的一种新方法，该方法具有一些传统技术所没有的优点：良好的容错能力、分类能力强、并行处理和自学习能力，并且是离线训练与在线识别。这些优点使它在手写字符识别中能对大量数据进行快速实时处理，达到良好的识别效果。

本文基于经典的卷积神经网络模型，使用 Tensorflow、Keras、Flask 等工具将其用于手写字符识别任务中。主要任务如下：

- (1) 构建卷积神经网络模型并应用于手写数字字符图像识别问题。
- (2) 构建卷积神经网络模型并应用于手写英文字符图像识别问题。
- (3) 搭建 Flask 网站并使用卷积神经网络模型对手写字符图像进行预测。

本文的实验结果验证了卷积神经网络应用于手写数字字符图像和手写英文字符图像识别问题上的可行性，并且无需对神经网络模型的局部参数进行调整和修改，能取得比较好的识别效果。

**【关键词】：**卷积神经网络；图像识别；手写数字识别；网站建设

## 【Abstract】

With the rapid development of artificial intelligence technology, handwritten character recognition technology has been widely used in zip code, financial statements, form entry, bank note processing, office automation and so on. Handwritten character recognition belongs to the branch of image recognition. As an information processing method, handwritten character recognition has a wide application background and a huge market demand.

Convolutional neural network is a deep neural network model based on multiple hidden layers proposed in recent years. It is used to study and deal with a new method in this field. This method has some advantages that traditional techniques do not have: good fault tolerance Strong ability, classification ability, parallel processing and self-learning ability, and offline training and online recognition. These advantages make it possible to handle a large amount of data in real-time in handwritten character recognition to achieve a good recognition effect.

This paper is based on the classic convolutional neural network model, Using Tensorflow, Keras, Flask and other tools for handwritten character recognition tasks. The main tasks are as follows:

- (1) Construct a convolutional neural network model and apply it to handwritten digital character image recognition problems.
- (2) Construct a convolutional neural network model and apply it to handwritten English character image recognition problems.
- (3) Build a Flask website and use a convolutional neural network model to predict handwritten characters.

The experimental results in this paper verify the feasibility of Convolution Neural Network applied to the recognition of handwritten digit characters and handwritten English characters, and it is unnecessary to adjust and modify the local parameters of the neural network model to obtain better recognition results.

**【Keywords】** : CNN; Image recognition; Handwritten digital recognition; Website construction

# 目录

|                           |    |
|---------------------------|----|
| 【摘要】 .....                | I  |
| 【Abstract】 .....          | II |
| 第 1 章 绪论 .....            | 1  |
| 1.1 开发背景和意义 .....         | 1  |
| 1.2 图像识别技术现状 .....        | 1  |
| 1.3 论文的研究内容 .....         | 2  |
| 第 2 章 相关研究技术 .....        | 3  |
| 2.1 Python 语言 .....       | 3  |
| 2.2 卷积神经网络 .....          | 4  |
| 2.2.1 简介 .....            | 4  |
| 2.2.2 基本结构 .....          | 4  |
| 2.2.3 第三方集成库 .....        | 7  |
| 第 3 章 实验准备 .....          | 14 |
| 3.1 Python3.6 安装及测试 ..... | 14 |
| 3.2 虚拟环境搭建 .....          | 14 |
| 3.3 Flask 网站框架安装 .....    | 15 |
| 3.4 手写字符的数据准备 .....       | 16 |
| 3.4.1 手写数字数据集 .....       | 16 |
| 3.4.2 手写英文数据集 .....       | 17 |
| 3.5 实验工作流程 .....          | 18 |
| 第 4 章 手写字符图像识别实现 .....    | 19 |
| 4.1 加载数据 .....            | 20 |
| 4.2 卷积神经网络的构建 .....       | 21 |
| 4.3 卷积神经网络的训练 .....       | 22 |
| 4.4 模型评估 .....            | 23 |
| 4.5 网页端识别手写字符 .....       | 23 |
| 4.5.1 加载训练过的模型 .....      | 23 |
| 4.5.2 浏览器网页端访问 .....      | 23 |
| 4.5.3 识别应用预测结果 .....      | 24 |
| 第 5 章 总结 .....            | 29 |
| 参考文献 .....                | 30 |
| 致谢 .....                  | 31 |

## 第 1 章 绪论

### 1.1 开发背景和意义

近些年，伴随着人工神经网络理论和图像识别技术的发展，基于卷积神经网络的图像识别技术逐渐开始新兴起来。图像识别，简单来说，就是利用现代计算机技术采集需要的对象，以图像的数据为基础，对图像进行对象识别，让计算机或其他机器来模仿人的视觉感知，能够自动完成对某一信息的处理能力，以识别不同模式的目标和对象的技术，代替人完成图像分类及辨别的任务。就当前而言，图像识别的主要应用：光学字符的识别（如手写数字、信封上邮政编码、汉字、英文、条形码识别、二维码、汽车牌照等），以及生物特征识别（如光学指纹、人脸面部特征、虹膜等）。

在日常生活中，离不开数字和英文字符的使用，我们每天都要处理大量的数字和英文字符工作，比如邮政编码、快递单号、公司统计报表、银行汇款转账、公司账务报表等。与其花大量时间来处理这些繁琐的任务，不如让计算机代替人来实现字符图像识别这样繁重的手工劳动。

字符识别主要任务是利用计算机对输入的手写字符图像翻译成计算机文字的过程，手写体识别根据书写的时间和识别时间的关系，分为联机手写体识别和脱机手写体识别两种，即实时手写体识别和非实时手写体识别。其中脱机手写体识别由于书写者的关系，其书写的字符的随意性较大，如手写字符的随意性很大，都是由人为书写的，如字体的笔画粗细，字体大小，字的倾斜角度等等因素都有可能直接影响到字符的识别准确率，所以手写体字符识别是一个有挑战性的课题。而传统的识别方法如贝叶斯算法、支持向量机算法、Kmeans 聚类算法等对复杂分类问题数学函数表示能力以及网络的泛化能力有限，经常达不到较高的识别准确率。随着科学技术的发展与研究的不断深入，卷积神经网络的出现为解决这一问题提供了可能，卷积神经网络的优势在于其可以通过训练样本数据自动获取学习样本的特征，能够避免字符特征提取的难题。它已经成为当前图像识别领域研究的热点。

### 1.2 图像识别技术现状

随着计算机信息技术，移动互联网、智能手机和社交网络的不断发展，图像识别这一技术在社会各个领域的应用越来越广泛和普及，并且已经逐渐开始渗入到了我们的日常生活当中。

与文本相比，图片数据可以为用户提供更生动、更容易理解、更有趣、更艺术化的信息。其次，从图片来源的角度来看，智能手机为我们提供了方便的拍摄和截图，帮助我们更快地收集和记录信息。然而，随着图像成为互联网的主要信息载体，问题出现了。当信息通过书面记录或存储成文字时，我们可以通过关键词搜索，容易找到和编辑内容，但当信息被图像记录时，我们不能检索图像的内容，这时就需要计算机来完成图像识别的任务了。就目前而言，图像识别技术是排在世

界第一我国的图像识别其本身具有一定的优势，加之我国的人口众多，上网的用户总体基数大，对于处理图像存储和识别等问题，都有较好的方法和经验。现在图像存储的像素点高，处理精度高、特别是数字图像，在图像分析的基础上研究图像中目标的性质。但是，在实际发展过程中，该技术还是存在着一定的问题。

通过查阅资料，发现手写字符图像识别是属于图像识别学科下的分支，是图像处理与模式识别研究领域的重要应用之一。在过去的数十年中，许多研究者和前辈提出了许多的传统识别方法，并且取得了一定的成效。而且字符识别在大规模数据统计如全国的身份证上的姓名、性别、民族、出生日期、全国人口普查、公司的财务与税务、邮件和信封上的分拣等应用领域都有着广阔的应用前景。

本文所探讨的是基于卷积神经网络的手写字符图像识别研究，是在传统计算机图像识别方法的基础上融合了当今卷积神经网络算法的一种图像识别方法，通过构建卷积神经网络模型并将该模型用于训练标准的样本数据，并且成功的将该卷积神经网络模型运用到了网页端的手写字符图像识别应用程序上。

### 1.3 论文的研究内容

本论文研究的内容是：基于卷积神经网络，利用公用数据集 MNIST 和 EMNIST，对样本数据集进行训练和学习迭代，以达到高效率，高准确度识别目标。本文的基本框架如下：

第一章 绪论，本章分析了论文研究的背景和意义，现今国内外图像识别的技术现状及论文所要研究的内容。

第二章 相关研究技术，本章针对论文中所实现的基于卷积神经手写字符图像识别中所用到的开发工具 Python，背景技术卷积神经网络与 Flask 网站进行简单的介绍。

第三章 实验准备，通过在 CentOS 平台上搭建并测试和运行 Python3 环境，安装虚拟环境与 Flask 网站，并对其进行测试和运行。手写字符数据集的准备，包括公共手写数字字符和公共手写英文字符的数据集下载和预处理准备工作。

第四章 手写字符图像识别实现，通过基于经典卷积神经网络算法，构建神经网络模型，使用公共数据集对神经网络模型进行样本的训练，并且成功的将该卷积神经网络模型运用到了网页端的手写字符图像识别应用程序上。

第五章 总结，对论文的完成情况进行总结和说明，表述其开发过程中所遇到的问题及解决方法，并对今后工作的展望。

## 第 2 章 相关研究技术

### 2.1 Python 语言

Python 是面向对象的解释型高级计算机程序设计语言，具有非常丰富和强大的库以及第三方的库文件。一种高级的、多用途的、解释的、交互式的和面向对象的编程语言。Python 易于学习，通用脚本语言 Python 是一种编程语言，它支持结构化和功能性方法，并且具有数据结构的构建、可移植性和可扩展性。它也是一种可扩展的语言。解释意味着它在运行时通过解释器进行处理。在执行之前不需要编译程序。它是用来给数据库交互的能力，构建的应用程序可以运行在 Windows 环境中，建立的 CGI 脚本可以运行从 Web 浏览器等。到处都是用于从商业网站在线游戏，从简单的转换脚本到复杂网络更新银行和其他金融机构的例程。

Python 是一种编程语言，可以在许多不同的平台上应用。它附带了标准库，包括几乎所有的 Internet 协议、操作系统接口、测试和字符串处理。字符串处理中的正则表达式，计算文件之间的差异，Unicode。Internet 协议—HTTP、SMTP、FTP、XML-RPC、IMAP、POP、CGI 编程。软件工程中的单元测试、日志记录、剖析和解析 Python 代码。操作系统接口中的系统调用、文件系统和 TCP/IP 套接字。Python 程序被送入 Python 的解释器后，Python 解释器将直接运行它们，所以没有编译（其他语言就是这种情况）的过程。这将有助于快速和更容易的获取有关 Python 代码的反馈更快，（例如，查找错误）。Python 意味着你可以更快地完成并执行程序，这使编程变得有趣！

与其他编程语言不同，没有几行代码能胜任这份工作。不需要花括号，但需要适当的缩进。不需要 Python 的变量数据类型声明将处理它。Python 代码不需要有类，可以通过在函数或方法中声明它们来获得功能。创建模块并导入它以重用代码。它被用来支持有限的多重继承形式，非常容易学习。它支持其他编程语言所具有的面向对象的程序设计系统概念。Python 是跨平台的，几乎任何人都可以使用它，可以在 Windows、Mac 和 Linux 操作系统上运行 Python 程序，从大型服务器到像树莓派这样的小型计算机。甚至可以在 Android 和 iOS 平板电脑上运行 Python 程序。

Python 是免费开源软件。可以在遵守 Python 协议条款的规定下，重新发布或修改。可以在官网上自由下载并运行 Python，而无需支付任何费用，并且使用它编写的任何程序都可以使用并以任何想要的方式共享。这也意味着 Python 源代码（计算机运行的可读形式）是可用的，Python 的源代码公开的，任何人都可以下载查看 Python 源代码。

Python 的特点：面向对象的、免费的、可移植的、强大的、可伸缩的、可溶混的、易于使用的、容易学习、稳定的、支持 GUI 编程。



## 2.2 卷积神经网络

### 2.2.1 简介

在机器学习中，卷积神经网络(Constitutional Neural Networks, CNN)是一种类神经网络，是基于多层深度神经网络而建立起来的一套学习算法。卷积神经网络，因为其具有卷积运算操作，所以称为卷积神经网络。卷积神经网络主要的领域应用于图像识别的相关任务上。诸如，图像分类、图像语义分割、图像检索、物体检测等计算机视觉问题。还有卷积神经网络去自然语言的处理中的文本进行分类等工作，软件工程数据挖掘中的软件 BUG 问题预测等问题都可以使用卷积神经网络进行处理，并取得了较传统的算法甚至其他神经网络算法更优的预测效果。

卷积神经网络与一般的神经网络有着不同的结构，常规的神经网络通过一系列隐藏的层来转换输入。每一层都由一组神经元组成，每一层都与前一层的所有神经元完全相连。最后，还有一个完全连接的层连接输出层输出预测结果。卷积神经网络有点不同。首先，这些层是在三维空间中组织的：宽度，高度和深度。此外，一层的神经元不连接到下一层的所有神经元，但只连接到它的一小部分。最后，最终的输出将被简化为一个概率分数的向量，沿着深度维度组织。

卷积神经网络受到大脑的启发，神经元之间的连通性模式类似于动物视觉皮层的组织。个体皮层神经元对刺激的反应只在被称为接收场的视觉区域的受限区域。不同神经元的接受域部分重叠，从而覆盖整个视野。卷积神经网络允许计算机被看到，换句话说，通过将原始图像通过图层转换成一个类的分数来识别图像。CNN 受到视觉皮层的启发。每次我们看到一些东西，一系列的神经元层就会被激活，每一层都能检测出一系列的特征。比如线条、边缘。高层次的层将检测更复杂的特征，以便识别我们所看到的。

### 2.2.2 基本结构

卷积神经网络是由一个输入层，一个输出层以及多个隐藏层组成。如图 2.1 LeNet-5 结构。

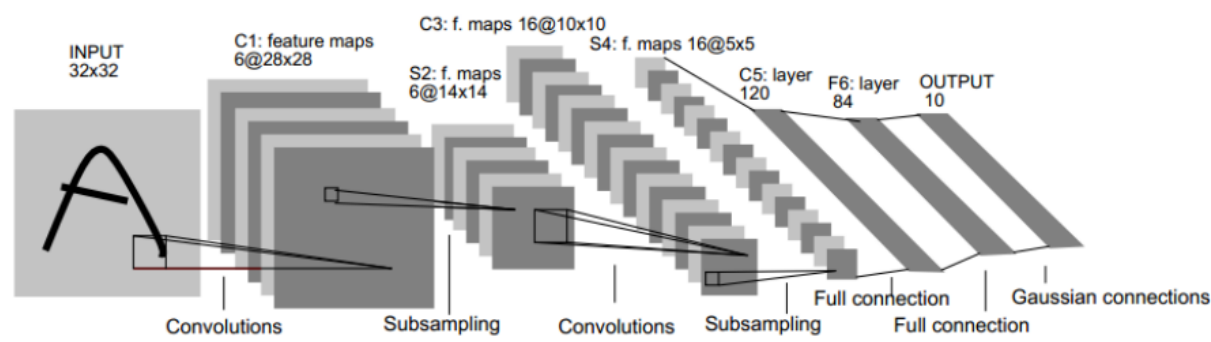


图 2.1 LeNet-5 结构

LeNet-5 结构是典型的卷积神经网络模型，其隐藏层中有以下 3 层：

1. 卷积层 (Convolutional layer)，是卷积神经网络中的基础操作，卷积层是 CNN 的核心组成部分。卷积层将卷积运算应用于输入，将结果传递给下一层。卷积模拟了单个神经元对视觉刺激的反应，每个卷积神经只处理其接收域中的数据。该层的参数由一组可学习的过滤器(或内核)组成，它们有一个小的接受域，但可以通过输入卷的全部深度进行扩展。在转发过程中，每个过滤器都在输入卷的宽度和高度上进行卷积，计算过滤器和输入之间的点积，并生成该过滤器的二维激活映射。因此，当网络在输入的某个空间位置检测到某种特定类型的特征时，它就会学习过滤器。在深度维度上叠加所有过滤器的激活映射，形成卷积层的全部输出量。因此，输出卷中的每一个条目都可以被解释为一个神经元的输出，该神经元在输入中观察一个小区域，并在相同的激活图中与神经元共享参数。卷积层对输入应用卷积操作，将结果传递给下一层。卷积模拟单个神经元对视觉刺激的反应，每个卷积神经只处理它的接收域的数据。

虽然完全连接的前馈神经网络可以用来学习特征和分类数据，但是将这种结构应用于图像是不现实的。由于与图像相关的输入尺寸非常大，因此即使在较浅(与较深)的体系结构中，也需要大量的神经元，因为每个像素都是相关的变量。例如，对于大小为  $100 \times 100$  的(小)图像，一个完全连接的层对第二个层中的每个神经元都有 10000 权值。卷积运算减少了自由参数的数量，使得网络在参数较少的情况下更加深入，从而为这个问题提供了一个解决方案。无论图像大小如何，大小为  $5 \times 5$  的平铺区域(每个区域具有相同的共享权重)只需要 25 个可学习参数。该方法解决了传统的多层神经网络在训练过程中由于梯度的消失问题

2. 池化层 (Pooling layer)，池化操作，这是一种非线性下采样。用几个非线性函数来实现池化操作，通常使用的池化操作为平均值池化和最大值池化，本文手写体识别应用采用的是最大值池化 (max-pooling)。通常使用的池化操作的为均值池化 (average-pooling) 和最大值池化 (max-pooling)，本文的卷积神经网络便使用的是最大值池化。池化层的作用是可以使输入的图像的特征值不变，还能达到图像的特征降维。卷积网络可能包括局部或全局汇聚层，它将神经元簇的输出与下一层的单个神经元相结合。例如，最大池化使用的最大值是来自前一层的每一个神经元簇。另一个例子是平均值池化，它使用前一层的每个神经元的平均值。如果使用的卷积层相同，但操作不相同。池化层就不需要包含其需要的学习参数。使用时仅需指定池化类型(平均值或最大值等)、池化操作的核大小和池化操作的步长等超参数即可。池化层在输入在每个深度切片上是独立运行的，并在存储空间上动态调整其大小。最常见的形式是一个大小为  $2 \times 2$  的滤波器，如下图 2.2 使用  $2 \times 2$  过滤器最大池化 (步幅为 2)。本文手写体识别应用采用的是  $3 \times 3$  的滤波器。在输入在每个矩阵上沿着高度与宽度，每个滤波器的步幅为 2。在这种情况下，如果每个矩阵的最大操作超过 4 个数字。深度和维度将保持

不变。池化函数除了最大值池化之外，还有平均值池化和 L2 范数池化。平均值池化通常已经不怎么使用了，一般分类问题中使用最多的是就是最大值池化，在实践中效果更好。

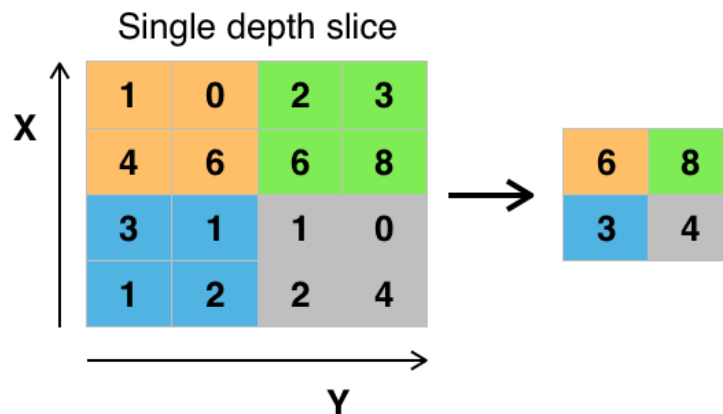


图 2.2 使用 2x2 过滤器最大池化（步幅为 2）

3. 全连接层（Fully-Connected layer），把每一层神经元与另一层的每个神经元连接起来，在整个卷积神经网络中起到了“分类器”的作用。最后，在经过卷积层和池化层化之后，神经网络中高层部分的分类操作则要通过全连接层来完成。一个完全连接层将前一层中所有神经元(无论是完全连接的、共享的还是卷积的)连接到它所拥有的每一个神经元。完全连接层不再是空间位置的了(可视化为成一维的)，因此在完全连接的层之后就不会有卷积层。本文所中最后两层采用的两个全连接层，第一个全连接层用来连接前面卷积层过来的输入数据，第二个全连接层是用来分类输出。

卷积神经网络核心的思想就是局部感知、权值共享以及池化，简化神经网络参数并使网络具有一定程度的位移、尺度、形状不变性。下面介绍下局部感知和权值共享两个概念。

**局部感知（Local connectivity）：**在处理像图像这种维度的数据作为输入时，将每个神经元连接到前一层中的所有神经元是不现实的，因为这样的神经网络结构是没有考虑数据存储的空间结构。卷积网络利用相邻层神经元之间的局部连接模式，利用空间局部相关性:每个神经元只连接到输入量的一小部分。这种连通性的程度是一个被称为神经元接收的超参数。连接在空间中是局部的(沿宽度和高度)，但始终沿着输入卷的整个深度进行扩展。这样的体系结构确保了所学习的过滤器对空间局部输入模式产生最强烈的响应。

**权值共享（Spatial arrangement）：**Spatial arrangement 在卷积层中使用权值共享方案来控制参数的数量。它依赖于一个合理假设:如果切片的特征对于在某个空间位置计算，则在其它位置计算也是有用的。换句话说，将一个二维的深度切片表示为深度切片，我们在每个深度切片中限制神经元使用相同的权重和偏差。由于单个深度切片中的所有神经元都具有相同的参数，所以在 CONV 层的每个深度切片上的前向传递可以被计算为神经元的权重与输入量的卷积(因此名称为卷积层)。因此，通

常将权重集称为卷积核(或内核),它与输入进行卷积。这个卷积的结果是一个激活映射,每个不同的过滤器的激活映射集合一起叠加在深度维度上,以产生输出量。参数共享有助于 CNN 架构的翻译不变性。有时,参数共享假设可能没有意义。在 CNN 的输入图像有一些特定的中心结构的情况下尤其如此,我们期望在不同的空间位置上可以学习到完全不同的特性。一个实际的例子是,当输入是以图像为中心的人脸时:我们可能会期望在图像的不同部分中学习不同的眼睛特定或特定毛发的特征。在这种情况下,放大参数共享方案是很常见的,而只是将其称为局部连接的层。

### 2.2.3 第三方集成库

Anaconda 是一个 python 发行版,用于数据科学和机器学习相关应用(大规模数据处理,预测分析,科学计算)。具有安装和包管理工具,提供了大量的软件包和商业支持。同时也是一个环境管理器,提供了创建不同 python 环境的工具,每个环境都拥有自己独立的个性化设置,在数据科学平台上提供了更大的优势。Anaconda 发行版附带超过 1,000 个数据包,以及 Conda 包和虚拟环境管理器(称为 Anaconda Navigator),因此无需学习如何独立安装每个库。可以使用 conda 安装命令或使用与 Anaconda 一起安装的 pip 安装命令从 Anaconda 存储库单独安装开放源码数据包。Pip 包提供了 conda 包的许多特性,在大多数情况下它们可以协同工作。可以使用 conda 构建命令来创建自己的定制包,可以通过将它们上传到 Anaconda Cloud、PyPi 或其他存储库来与他人共享它们。

Keras 是一个用 Python 编写的开源神经网络库,用于深度学习。它能够在 TensorFlow, Microsoft Cognitive Toolkit, Theano 或 MXNet 上运行。它的开发是为了使深度学习模型的实现尽可能快速和容易的进行研究和开发。在支持深度神经网络的快速实验,它侧重于用户友好、模块化和可扩展性。它是作为 ONEIROS(开放式神经电子智能机器人操作系统)项目研究工作的一部分而开发的, Keras 的主要作者和维护者是谷歌工程师 Francois Chollet。它在 Python 2.7 或 3.5 上运行,并且可以在给定底层框架的 gpu 或 cpu 上无缝地执行。它是在麻省理工学院许可下发布的。Keras 是由谷歌工程师 Francois Chollet 开发和维护的,使用了四个指导原则:

1. 模块化: 一个模型可以被理解为一个序列或一个图。深度学习模型的所有关注点都是可以任意组合的离散组件。
2. 极简主义: 该库提供的内容仅够实现一个结果,没有不必要的装饰和可读性最大化。
3. 可扩展性: 新组件在框架中有意添加和使用,目的是让研究人员尝试并探索新的想法。
4. 使用 Python: 没有使用自定义文件格式的独立模型文件。一切都是原生 Python 文件。

2017 年,谷歌的 TensorFlow 团队决定在 TensorFlow 的核心库中支持 Keras。Chollet 解释说, Keras 被认为是一个接口,而不是一个独立的机器学习框架。它提供了一组更高级、更直观的抽象,

使开发深度学习模型更容易,而不管使用的计算后端是什么。Microsoft 也向 Keras 添加了一个 CNTK 后端,可从 CNTK v2.0 获得。Keras 包含了许多常用的神经网络构建块的实现,如层、目标、激活函数、优化器和许多工具,以使处理图像和文本数据更容易。代码托管在 GitHub 上,社区支持论坛包括 GitHub 问题页面和 Slack 通道。Keras 允许用户在智能手机(iOS 和 Android)、网络或 Java 虚拟机上生产深度模型。它还允许在图形处理单元(GPU)集群上使用深度学习模型的分布式训练。

NumPy 是一个 Python 包,表示的是“Numerical Python”。它是科学计算的核心库,包含一个强大的  $n$  维数组对象,为集成 C、C++等提供了工具。它在线性代数、随机数能力等方面也很有用。

NumPy 目标是 Python 的 CPython 引用实现,这是一个非优化的字节码解释器。为这个版本的 Python 编写的数学算法通常比编译后的同类算法运行得慢得多。NumPy 解决了缓慢的问题,部分原因是它提供了多维数组、函数和运算符,它们在数组上高效地运行,需要重写一些代码,大部分是使用 NumPy 的内部循环。在 Python 中使用 NumPy 可以提供与 MATLAB 类似的功能,因为它们都被解释为,并且它们都允许用户编写快速的程序,只要大多数操作都是在数组或矩阵上进行,而不是在标量上。相比之下, MATLAB 拥有大量的附加工具箱,特别是 Simulink,而 NumPy 则是与 Python(一种更现代、更完整的编程语言)进行了内在的集成。此外,还提供了互补的 Python 包;SciPy 是一个增加更多 matlablike 功能的库,Matplotlib 是一个提供 matlablike 绘图功能的绘图包。在内部, MATLAB 和 NumPy 都依赖 BLAS 和 LAPACK 进行有效的线性代数计算。Numpy 广泛使用的计算机视觉库 OpenCV 的 Python 绑定使用 NumPy 数组来存储和操作数据。由于具有多个通道的图像被简单地表示为三维数组,因此对其他数组进行索引、切片或屏蔽是访问图像特定像素的非常有效的方法。NumPy 数组作为 OpenCV 中的通用数据结构,用于图像、提取特征点、过滤内核等,大大简化了编程工作流程和调试。

NumPy 的核心功能是它的“ndarray”,用于  $n$  维数组和数据结构。这些数组是内存上的跨接视图。与 Python 内置的列表数据结构(尽管名称是动态数组)不同,这些数组是同质类型的:单个数组的所有元素必须是相同的类型。这些数组也可以被看作是 C/C++、Cython 和 Fortran 扩展分配给 CPython 解释器的内存缓冲区,而不需要到处复制数据,从而与现有的数字库具有一定的兼容性。SciPy 包利用了这个功能,它封装了许多这样的库(特别是 BLAS 和 LAPACK)。NumPy 支持内存映射的 ndarray。

Scikit-learn 是 Python 编程语言的免费软件机器学习库。它提供了大量常用算法的有效版本,具有支持向量机、随机森林、梯度增强、k-means、DBSCAN 等多种分类、回归和聚类算法,设计用于与 Python 的数字和科学库 NumPy 和 SciPy 互操作。它是在许可的简化 BSD 许可下授权的,并且在许多 Linux 发行版下发布,鼓励学术和商业使用。该库构建在 SciPy(科学 Python)之上,在使用 scikit-learn 之前,必须安装该 SciPy(科学 Python)。这个栈,包括:NumPy: 基  $n$  维数组包、科学计算的基

本库、Matplotlib:全面的 2 d / 3 d 绘图、IPython:增强交互控制台、SymPy:数学符号、Pandas:数据结构和分析。SciPy 护理的扩展或模块,通常称为 SciKits。因此,该模块提供学习算法并命名为 scikit-learn。要深入关注诸如易于使用、代码质量、协作、文档和性能等问题。虽然这个接口是 Python 的,但是 c 库可以提高性能,比如用于数组和矩阵操作的 numpy、LAPACK、LibSVM 和使用 cython。

Scikit-learn 的特点是一个干净的、统一的、流线型的 API,以及非常有用的、完整的在线文档。这种一致的好处是,理解 Scikit-learn 的基本用法和语法,切换到新的模型或算法就非常简单了。Scikit-learn 项目以 scikits 的形式开始,David Cournapeau 设计的谷歌夏季代码项目《学习》。它的名称源于它是一个“SciKit”(SciPy 工具包),一个独立开发的分布式第三方扩展到 SciPy。Scikit-learn 通过 Python 的一致接口提供了一系列监督和非监督学习算法。Scikit-learn 大部分是用 Python 编写的,一些核心算法用 Cython 编写以实现性能。支持向量机由一个围绕 LIBSVM 的 Cython 包装器实现;逻辑回归和线性支持向量机通过类似的包装在 LIBLINEAR。

Tensorflow 是一个开源免费软件库,用于跨一系列任务的数据流编程,专注于谷歌创建的机器学习、深度学习和其他统计和预测分析工作负载,并于 2015 年由谷歌作为开源库发布。它的设计目的是简化为数据科学家、统计学家和预测建模者等用户开发和执行高级分析应用程序的过程。TensorFlow 最初作为 Apache 2.0 开源许可的一部分发布,最初由谷歌 Brain Team 的工程师和研究人员开发,主要用于内部使用。TensorFlow 被认为是闭源应用程序 DistBelief 的继任者,

TensorFlow 是一个将复杂计算表示为图形的框架,这使得分析模型变得更容易,使用称为张量的多维数组来做同样的事情。TensorFlow 软件处理以图形形式排列为计算节点的数据集。连接图中节点的边可以表示多维向量或矩阵,从而产生所谓的张量。因为 TensorFlow 程序使用的数据流架构与计算的广义中间结果一起工作,所以它们对非常大规模的并行处理应用程序尤其开放,而神经网络就是一个常见的例子。

在很大程度上,TensorFlow 应用是在机器学习和深度学习领域中先进的大规模人工智能项目。在为谷歌公司的 RankBrain 机器学习系统提供动力的过程中,TensorFlow 被用来提高公司旗舰搜索引擎的信息检索能力。TensorFlow 紧跟在一个名为 DistBelief 的封闭的谷歌框架的基础上,公司内部使用这个框架来进行无监督的特性学习和基于非常大的神经网络和反向传播算法的深度学习应用程序。TensorFlow 是深度学习实现的试验台,包括高级图像和语音识别、自然语言处理、推荐引擎和预测分析。由于 TensorFlow 的设计目的是能够与谷歌自己的计算基础设施分开工作,因此它的代码更容易移植到外部使用。它也是一种更通用的机器学习框架,不像 DistBelief 那样专注于神经网络。此外,它的设计是为了支持更快的配置和运行在高级 API 上。

TensorFlow 使得开发人员更容易设计、构建和训练深度学习模型。开发人员还可以使用其他一

些框架，但张量流是其中使用最广泛的框架，主要是因为其高度灵活的系统架构。一旦使用 TensorFlow 编程，你可以在任何地方运行它。它使它特别容易转移到谷歌的云平台。这就是谷歌计划如何通过帮助公司利用机器学习来解决云计算市场的问题。

谷歌还将 TensorFlow 框架用于应用程序，包括自动电子邮件响应生成、图像分类和光学字符识别，以及该公司与斯坦福大学(Stanford University)研究人员共同开发的药物发现应用程序。技术主管和高级研究员在谷歌讨论开发并使用 TensorFlow。其他上市公司 TensorFlow 网站框架的用户包括 Airbnb、可口可乐、eBay、英特尔、高通、SAP、Twitter、乳房和 Snapchat 开发者快速 STATS LLC 公司另一个用户,一个体育咨询公司运行 TensorFlow-based 深度学习模型来分析诸如职业体育奥运会期间运动员的运动。基于 tensorflow 的深度学习也已经成为了一些实验和测试的一部分，这些实验和测试涉及到如今规模更大的一项创新，即自动驾驶汽车。

2015 年，谷歌在 Apache 2.0 许可下将 TensorFlow 作为开源技术发布。从那时起，该框架在谷歌之外获得了各种追随者。例如，TensorFlow 工具作为附加模块支持，用于 IBM、Microsoft 和其他公司的机器学习和人工智能开发套件。

### 2.3 网站搭建技术

本实验所采用的服务器是阿里云的 CentOS7 x64 位,网站域名为: [http:// banglong.site:5000](http://banglong.site:5000), IP 地址是: 101.132.172.41。下面介绍有关网站搭建的相关技术。

HTML 超文本标记语言 (HyperText Markup Language)，是一种用普通文本描述网页的语言，HTML 在一个文件中插入的一组标记符号或代码，以便在万维网浏览器页面上显示。标记告诉 Web 浏览器如何为用户显示 Web 页面的文字和图像。每个标记代码都被称为元素(也称为标记)。有些元素成对出现，表示内容何时开始， Web 浏览器从 Web 服务器或本地存储接收 HTML 文档，并将文档呈现到多媒体 Web 页面中。HTML 从语义上描述 web 页面的结构，最初包含文档外观的提示。HTML 元素是 HTML 页面的构建块。使用 HTML 结构，图像和其他对象(如交互式表单)可以嵌入到呈现的页面中。HTML 为文本(如标题、段落、列表、链接、引号和其他项目)提供了一种创建结构化文档的方法。HTML 元素由标记描述，使用尖括号编写。标签如和直接将内容引入页面。其他标签如包围和提供有关文档文本的信息，可能包括其他标签作为子元素。浏览器不会显示 HTML 标记，而是使用它们来解释页面的内容。HTML 可以嵌入用脚本语言编写的程序，如 JavaScript，这会影响 web 页面的行为和内容。

第一个公开的 HTML 描述是一个名为“HTML 标签”的文档，这是 Tim Berners-Lee 在 1991 年年底在互联网上首次提到的。它描述了包含 HTML 最初的、相对简单的设计的 18 个元素。除了超链接标记之外，它们还受到了 SGMLguid 的强烈影响。SGMLguid 是 CERN 内部基于标准的通用

标记语言(SGML)的文档格式。

这些元素中的 11 个仍然存在于 HTML 4 中 HTML 是一种标记语言, web 浏览器用来将文本、图像和其他材料解释和组合成可视的或可听的 web 页面。每个 HTML 标记项的默认特性都在浏览器中定义, 这些特性可以通过 web 页面设计器对 CSS 的额外使用进行修改或增强。许多文本元素被发现在 1988 年 ISO 技术报告 TR 9537 技术使用 SGML, 而覆盖早期文本格式的功能所使用的语言如流命令在 1960 年代早期开发 CTSS(兼容的分时系统)操作系统这些:格式化命令来自排字工人所使用的命令手动格式文件。然而, 广义标记的 SGML 概念是基于元素(带有属性的嵌套注释范围), 而不仅仅是打印效果, 还有结构和标记的分离;使用 CSS, HTML 已经逐步向这个方向移动。

Berners-Lee 认为 HTML 是 SGML 的应用。它是由互联网工程特别工作组(IETF)正式定义的, 1993 年中期发布了 HTML 规范的第一个提案, 即“超文本标记语言”(HTML)。Berners-Lee 和 Dan Connolly 起草的 Internet 草案, 其中包括一个 SGML 文档类型定义来定义语法。该草案在六个月后到期, 但值得注意的是, 它承认 NCSA Mosaic 浏览器的自定义标记用于嵌入在线图像, 反映了 IETF 基于成功原型的标准理念。

CSS 层叠样式表(Cascading Style Sheets), 是一种样式表语言, 用于描述用标记语言(如 HTML)编写的文档的表示。除了 HTML 和 JavaScript 之外, CSS 是万维网的基础技术之一。CSS 设计的目的是使表示和内容分离, 包括布局、颜色和字体。这种分离可以提高内容的可访问性, 在表示特征的规范中提供更大的灵活性和控制, 通过在单独的 CSS 文件中指定相关的 CSS, 允许多个 web 页面共享格式, 并减少结构内容的复杂性和重复。格式和内容的分离也使得在不同的渲染方法(如屏幕、打印、语音(通过基于语音的浏览器或屏幕阅读器)和基于盲文的触觉设备上以不同的样式显示相同的标记页面成为可能。如果内容在移动设备上被访问, CSS 也有备用格式的规则, 名称级联来自指定的优先级方案, 以确定如果多个规则与特定元素匹配, 将应用哪种样式规则。这种级联优先级方案是可预测的。CSS 规范由万维网联盟(W3C)维护。网络媒体类型(MIME 类型)的文本是注册为使用 css 的 RFC 2318。W3C 为 CSS 文档运行一个免费的 CSS 验证服务

JavaScript 通常缩写为 JS, 是一种高级的、解释的编程语言。它是一种语言, 也被描述为动态的、弱类型的、基于原型的和多范式的。JavaScript 支持交互式 web 页面, 因此是 web 应用程序的重要组成部分。绝大多数的网站使用它, 所有主要的 web 浏览器都有一个专用的 JavaScript 引擎来执行它。作为一种多范式语言, JavaScript 支持事件驱动、函数式和命令式(包括面向对象和基于原型的)编程风格。它有一个用于处理文本、数组、日期、正则表达式和 DOM 的基本操作的 API, 但是语言本身不包含任何 I/O, 比如网络、存储或图形工具, 这些都依赖于它所嵌入的主机环境。最初只在 web 浏览器客户端实现, JavaScript 引擎现在嵌入在许多其他类型的主机软件, 包括在 web 服务器和数据



库服务器端和非 web 项目。如文字处理软件和 PDF 格式的软件，可以在 JavaScript 编写移动和桌面应用程序，包括桌面小部件。尽管 JavaScript 和 Java 之间有很强的外部相似性，包括语言名称、语法和各自的标准库，但这两种语言在设计上是截然不同的。1980 年，欧洲核子研究中心的承包商、物理学家蒂姆·伯纳斯-李(Tim Berners-Lee)提出并制作了一套“询盘”系统，供欧洲核子研究中心的研究人员使用和共享文件。1989 年，伯纳斯-李写了一份备忘录，提议建立一个基于互联网的超文本系统。Berners-Lee 在 1990 年末指定了 HTML 并编写了浏览器和服务端软件。那年，伯纳斯-李(Berners-Lee)和欧洲核子研究中心(CERN)数据系统工程师罗伯特·卡里奥，他 1990 年的个人笔记中，他列出了“使用超文本的许多领域中的一些领域”，并将百科全书放在首位。

Werkzeug 是 Python 语言编写的应用程序库，也就是 Web 服务器网关接口 (Web Server Gateway Interface) 应用程序的工具包，并且已获得 BSD 许可。Werkzeug 可以实现请求，响应和实用功能的软件对象。它可以用来构建一个定制的软件框架，并支持 Python 2.6, 2.7 和 3.3。

Jinja 是 Python 编写的 HTML 模板引擎，由 Armin Ronacher 创建的 BSD 许可证进行许可。Jinja 提供类似 Python 的表达式，是一种基于 HTML 的模板语言，因此可用于生成任何标记以及源代码。

requests 是一个 Python HTTP 库，在 Apache2 许可证下发布。该项目的目标是使 HTTP 请求更简单，更人性化。它的口号是：“Requests: HTTP for Humans”。

Flask 是用 Python 编写的基于 Werkzeug 工具包和 Jinja2 模板引擎的微型网站框架。Flask 的独特之处在于它是一个自定义的“微框架”，这意味着它背后的理念是为应用程序提供一个功能强大、简洁的核心，剩下的就交给开发人员了。它被归类为微框架，因为它不需要特定的工具或库。它没有数据库抽象层、表单验证或任何其他组件，在这些组件中，已有的第三方库提供公共功能。然而，Flask 支持扩展，可以添加应用程序特性，就好像它们是在 Flask 本身中实现的一样。对象关系映射器、表单验证、上载处理、各种开放身份验证技术和一些通用框架相关工具都有扩展。扩展比核心 Flask 程序更新得更频繁。Flask 被认为是更轻量级的，更注重体验。Flask 应用程序可以是单个 Python 文件，本文的手写体识别应用就是一个独立的 Python 文件。

在 2018 年，Pocoo 成为了一个由 Python 爱好者组成的国际组织。Flask 是由 Pocoo 的 Armin Ronacher 创建的：“这是一个愚人节玩笑，但事实证明，这个笑话很受欢迎，它本身就能应用到严肃的场合。当罗尼切和乔治·勃兰登堡门的时候使用一个用 Python 编写的公告板系统，Pocoo 项目 Werkzeug 和 Jinja2 也被开发出来。尽管没有一个主要的版本，Flask 仍然在 Python 爱好者中非常流行。截至 2016 年年中，它是 GitHub 上最流行的 Python web 开发框架。

WSGI 是 Web 服务器网关接口 (Web Server Gateway Interface)。它是一个规范，描述了 Web 服务器如何与 Web 应用程序进行通信，以及 Web 应用程序如何链接在一起以处理一个请求。WSGI 有

两个方面的应用：1. 服务器/网关。这通常是一个完整的 web 服务器(如 Apache 或 Nginx)，或者是一个可以与 web 服务器(如 flup)进行通信的轻量级应用服务器。2. 应用程序/框架。这是一个可调用的 Python，由 Python 程序或框架提供。在服务器和应用程序之间，可能有一个或多个 WSGI 中间件组件，它们实现 API 的两面，通常是在 Python 代码中实现的。

WSGI 没有指定应该如何启动 Python 解释器，也没有指定应该如何加载或配置应用程序对象，不同的框架和 web 服务器以不同的方式实现这一点。WSGI 中间件组件是一个可调用的 Python，它本身就是一个 WSGI 应用程序，但是可以通过委托给其他 WSGI 应用程序来处理请求。这些应用程序本身可以是 WSGI 中间件组件根据目标 URL 将请求路由到不同的应用程序对象，然后相应地更改环境变量。允许多个应用程序或框架在同一进程中并行运行，通过在网络上转发请求和响应来实现负载均衡和远程处理，执行内容后处理。

Ajax 代表了一组广泛的 Web 技术，可用于实现与后台服务器通信的 Web 应用程序，而不会干扰页面的当前状态。在创造“Ajax”一词的文章中，的杰西·詹姆斯·加勒特(Jesse James Garrett)解释道：用于表示的 HTML(或 XHTML)和 CSS。用于动态显示和与数据交互的文档对象模型(Document Object Model, DOM)。用于交换数据的 JSON 或 XML，以及用于操作的 XSLT。用于异步通信的 XMLHttpRequest 对象和 JavaScript 将这些技术结合在一起然而，从那时起，在 Ajax 应用程序中使用的技术和术语 Ajax 本身的定义中出现了许多开发。数据交换不再需要 XML，因此，处理数据不再需要 XSLT。JavaScript 对象表示法(JSON)经常被用作数据交换的替代格式，尽管也可以使用其他格式，如预格式化的 HTML 或纯文本。各种流行的 JavaScript 库(包括 JQuery)都包含抽象，以帮助执行 Ajax 请求。异步 HTML 和 HTTP (AJAX)涉及到使用 XMLHttpRequest 检索(X)HTML 片段，然后直接插入到 Web 页面中。

Ajax 使用 XHTML 处理内容，CSS 处理表示，文档对象模型和 JavaScript 处理动态内容显示。传统的 web 应用程序使用同步请求向服务器发送和接收信息。这意味着需要填写一个表单，点击 submit，然后使用来自服务器的新信息指向一个新页面。而使用 AJAX 时，单击 submit，JavaScript 将向服务器发出请求，解释结果，并更新当前屏幕。在最纯粹的意义，用户永远不会知道任何东西甚至被传输到服务器。XML 通常用作接收服务器数据的格式，尽管可以使用任何格式，包括纯文本。AJAX 是一种独立于 web 服务器软件的 web 浏览器技术。当客户端程序从后台服务器请求信息时，用户可以继续使用应用程序。直观自然的用户交互。不需要单击，鼠标移动是一个足够的事件触发器。AJAX 是由数据驱动的，而不是页面驱动的。

## 第 3 章 实验准备

### 3.1 Python3.6 安装及测试

在 CentOS7 服务器中 Python 的默认版本是 Python2.7 的，所以需要安装 Python3.6。在 CentOS 上使用源码安装的方法，以下是安装的步骤：

(1) 从官网下载 Python3.6 的安装包

```
[root@CentOS7 ~]# wget https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tar.xz
--2018-06-04 22:19:51-- https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tar.xz
Resolving www.python.org (www.python.org)... 151.101.72.223, 2a04:4e42:11::223
Connecting to www.python.org (www.python.org)|151.101.72.223|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17049912 (16M) [application/octet-stream]
Saving to: 'Python-3.6.5.tar.xz'
```

(2) 解压并安装 Python3.6

解压：

```
[root@CentOS7 ~]# xz -d Python-3.6.5.tar.xz
[root@CentOS7 ~]# tar xvf Python-3.6.5.tar
配置(configure)、编译(make)、安装(make install):
[root@CentOS7 Python-3.6.5]# ./configure --prefix=/usr/local/python3
[root@CentOS7 Python-3.6.5]# make && make install
```

其中配置时，`--prefix=/usr/local/python3` 选项是设置 Python3 的安装路径为 `/usr/local/python3`。

(3) 测试 Python3 是否安装成功

```
[root@CentOS7 Python-3.6.5]# python
Python 2.7.5 (default, Apr 11 2018, 07:36:10)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
```

### 3.2 虚拟环境搭建

安装完 Python3 后，还需要通过 Python3 为本文的手写字符图像识别应用设置虚拟环境。

```
[root@CentOS ~]# source ENV/bin/activate
(ENV) [root@CentOS7 ~]# python
```

```
Python 3.6.5 (default, May 26 2018, 15:22:34)
```

```
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
```

```
Type "help", "copyright", "redits" or "license" for more information.
```

上面的代码是使用 Python3.6 原生的 venv 模块创建 Python3 的虚拟环境。venv 模块提供了与 CentOS 系统中的 Python2 的隔离操作，每一个所创建的虚拟环境都拥有自己的 Python 版本和独立的 Python 包。手写体识别程序单独使用一个虚拟环境，不会影响到其他项目的环境，也不会影响到 CentOS 服务器中的 Python2 与 Python3，实现了 Python2 与 Python3 共存。

### 3.3 Flask 网站框架安装

通过使用 3.2 小节中 Python3 创建的 venv 虚拟环境。

(1) 激活虚拟环境并查看 Python 版本

```
[root@CentOS ~]# source ENV/bin/activate
```

```
(ENV) [root@CentOS7 ~]# python
```

```
Python 3.6.5 (default, May 26 2018, 15:22:34)
```

```
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
```

```
Type "help", "copyright", "redits" or "license" for more information.
```

(2) 安装 Flask 0.12.2

```
(ENV) [root@CentOS7 ~]# pip install Flask==0.12.2
```

(3) 测试 Flask 是否安装成功

首先创建 hello.py 并写入以下代码：

```
(ENV) [root@CentOS7 ~]# cat hello.py
```

```
from flask import Flask # 导入 Flask
```

```
app = Flask(__name__)
```

```
@app.route('/') # app.route 是 flask 中的路由
```

```
def hello_world():
```

```
    return 'Hello Flask!'
```

```
if __name__ == '__main__':
```

```
    app.run()      #函数 run（）启动本地服务器
```

然后运行程序：

```
(ENV) [root@CentOS7 ~]# python hello.py
```

\* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

测试结果:

```
[root@CentOS7 ~]# curl 127.0.0.1:5000
```

Hello Flask!

屏幕上成功输出 Hello Flask!则 Flask 安装完成。

## 3.4 手写字符的数据准备

### 3.4.1 手写数字数据集

手写数字 MNIST 的数据集来自于 <http://yann.lecun.com/exdb/mnist/>，由 Google 实验室的 Corinna Cortes 和纽约大学柯朗研究所的 YannLeCun 共同建立的一个手写数字数据库，训练库中共有 60,000 张手写数字图像，测试库中有 10,000 张。MNIST 是 NIST 一个更大的集合的子集。这些数字经过了尺寸标准化，并以固定大小的图像为中心。对于那些想在真实数据上学习技术和模式识别方法，同时在预处理和格式化上花费很少精力的人来说，这是一个很好的数据库。NIST 的原始黑白(两层)图像被归一化以适合于一个 20x20 像素的盒子，同时保留它们的纵横比。由于归一化算法所使用的抗锯齿技术，产生的图像包含灰色层次。通过计算像素的质心，将图像居中于 28x28 的图像中，并对图像进行平移，使这一点位于 28x28 字段的中心。

网站提供四份标准数据集文件如下：

```
.
├── mnist
│   ├── t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
│   ├── t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)
│   ├── train-images-idx3-ubyte.gz: training set images (9912422 bytes)
│   └── train-labels-idx1-ubyte.gz: training set labels (28881 bytes)
```

从上面网址下载公用数据集文件 MNIST.mat，见图 3.1。其大小为 22.178M，格式为 mat。为后续章节的实验提前做准备。


| Name  | Date modified  | Type                 | Size      |
|---|----------------|----------------------|-----------|
|  mnist.mat | 12/20/16 13:38 | Microsoft Access ... | 22,178 KB |

图 3.1 手写数字 MNIST 数据集文件

查看 emnist 中的数据，打印里面图像的维度。代码：`print(training_images.shape)`，输出结果为：(697932, 28, 28, 1)，共 97932 张训练图像，其维度为一个黑白通道的 28\*28 的像素的图

像，以矩阵方式存储在内存中。查看训练图中的一张图像的例子，先打印第 23 号图像和对应的编号：代码：`print(display(training_images[23]),training_labels[23])`，其输出结果见下图 4.2：



图 3.2 数字 3 储存方式和对应的编号

该数据的标签是数字 3，图像的数据维度为 28\*28 的矩阵，矩阵上的每个数值代表的是每个像素点的灰度值。

### 3.4.2 手写英文数据集

EMNIST 数据集来自 [https://www.westernsydney.edu.au/bens/home/reproducible\\_research/emnist](https://www.westernsydney.edu.au/bens/home/reproducible_research/emnist)，由悉尼大学 Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik 共同建立。MNIST 数据集已经成为学习、分类和计算机视觉系统的标准基准。促进其广泛采用的是任务的可理解和直观的性质，其相对较小的大小和存储要求以及数据库本身的可访问性和易用性。

EMNIST 数据库来自一个更大的数据集，称为 NIST 特殊数据库 19，它包含数字、大写和小写的手写字母。EMNIST 是 NIST 特殊数据库 19 派生出来的一组手工字符数字，并转换成与 MNIST 数据集直接匹配的 28x28 像素的图像格式和数据集结构。上述文章介绍了一个完整的 NIST 数据集的变体，称之为 Extended MNIST (EMNIST)，它遵循与创建 MNIST 数据集相同的转换范式。结果是一组数据集，这些数据集构成了一个更具有挑战性的分类任务，涉及字母和数字，并且与原始 MNIST 任务共享相同的图像结构和参数，允许与所有现有的分类器和系统直接兼容。通过对转换后的 NIST 数字和 MNIST 数字的分类结果的比较，给出了基准测试结果，并对转换过程进行了验证。数据集以两种文件格式提供。

这两个版本的数据集都包含相同的信息，提供这些信息完全是为了方便。第一个数据集以 Matlab 格式提供，可以通过 Matlab 和 Python(使用 `scipy.io`)访问。`loadmat` 功能)。第二个版本的数

数据集以与原始 MNIST 数据集相同的二进制格式提供。

从上面网址下载公用数据集文件 EMNIST.mat，见图 3.3。其大小为 263.138M，格式为 mat。为后续章节的实验提前作准备。


| Name   | Date modified  | Type                 | Size       |
|--|----------------|----------------------|------------|
|  emnist.mat | 12/18/16 13:28 | Microsoft Access ... | 263,138 KB |

图 3.3 手写英文 MNIST 数据集文件

### 3.5 实验工作流程

本文实验中的 Flask 网站基本运行流程如下图 3.1 所示：

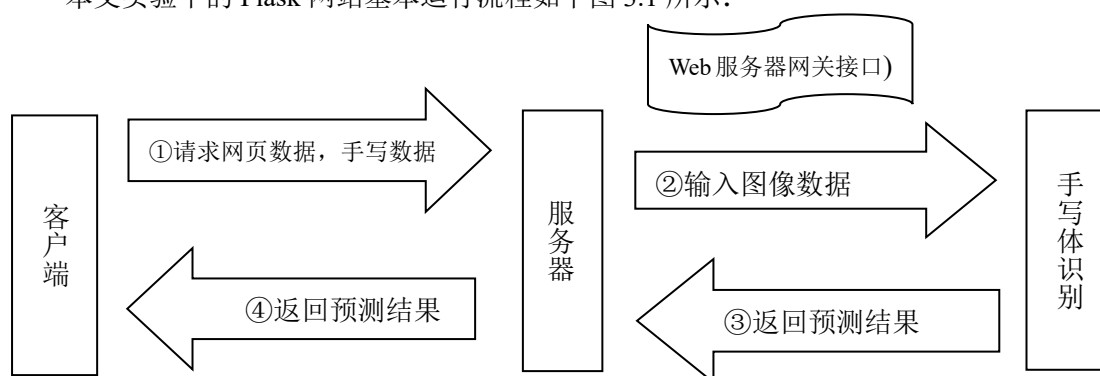


图 3.4 Flask 网站运行流程

如上图 3.4 所示，Flask 网站是由服务器，客户端和应用程序，客户端与服务器之间通过是通过 TCP/IP 协议中的 HTTP 互相访问的。服务器端上运行 Flask 程序，手写体识别应用运行在 Flask 服务端。其执行的顺序为：①用户浏览器打开网址：banglong.site:5000，请求网页数据，并输入手写字符。②Flask 服务器端将用户画布里输入的图像转成 base64 格式的 png，经过转置处理操作后再传给手写体识别应用。③手写体识别应用拿 Flask 传过来的图像后，跟训练过的模型进行比较，得到预测结果传回 Flask。④Flask 接收到返回预测结果再传回到网页端。

应用程序是要搭建在服务器之上的，因此，几乎所有的 Python Web 框架都是采用服务器网关接口 WSGI(Web Server Gateway Interface)来实现应用程序与服务器之间通信。本文的手写体识别程序就是由 Flask 创建的 app。应用程序要跑起来，就必须通过服务器。这里的服务器可以是 Apache、nginx、gunicorn 等。

WSGI 规定了应用程序与服务器之间通过所用到的接口。服务器调用应用程序时，会传两个参数至应用程序：environ 这是要请求的信息，start\_response 是应用程序在处理运算完后调用的函数，参数是状态码，响应头部与错误信息。本文服务器端所采用的是在服务器端 Flask + gunicorn 搭建起来的。

## 第 4 章 手写字符图像识别实现

两次实验分别实现了手写数字和手写英文字母识别，并对手写字符识别的方法进行了简要介绍和分析。本文所实现的手写字符识别程序，基于经典的 CNN（卷积神经网络）算法，并使用了 9 层神经网络模型。使之运行在 Flask 网站构架平台下，具有手写字符图像读取、特征提取及识别功能。

两次实验采用的 9 层卷积神经网络模型分别是 1 输入层、2 层卷积层、1 池化层、1 Dropout 层，1 Flatten 层，1 全连接层，1 Dropout 层，1 全连接层。各层连接和简单描述如下图 4.1 卷积神经网络构建所示。

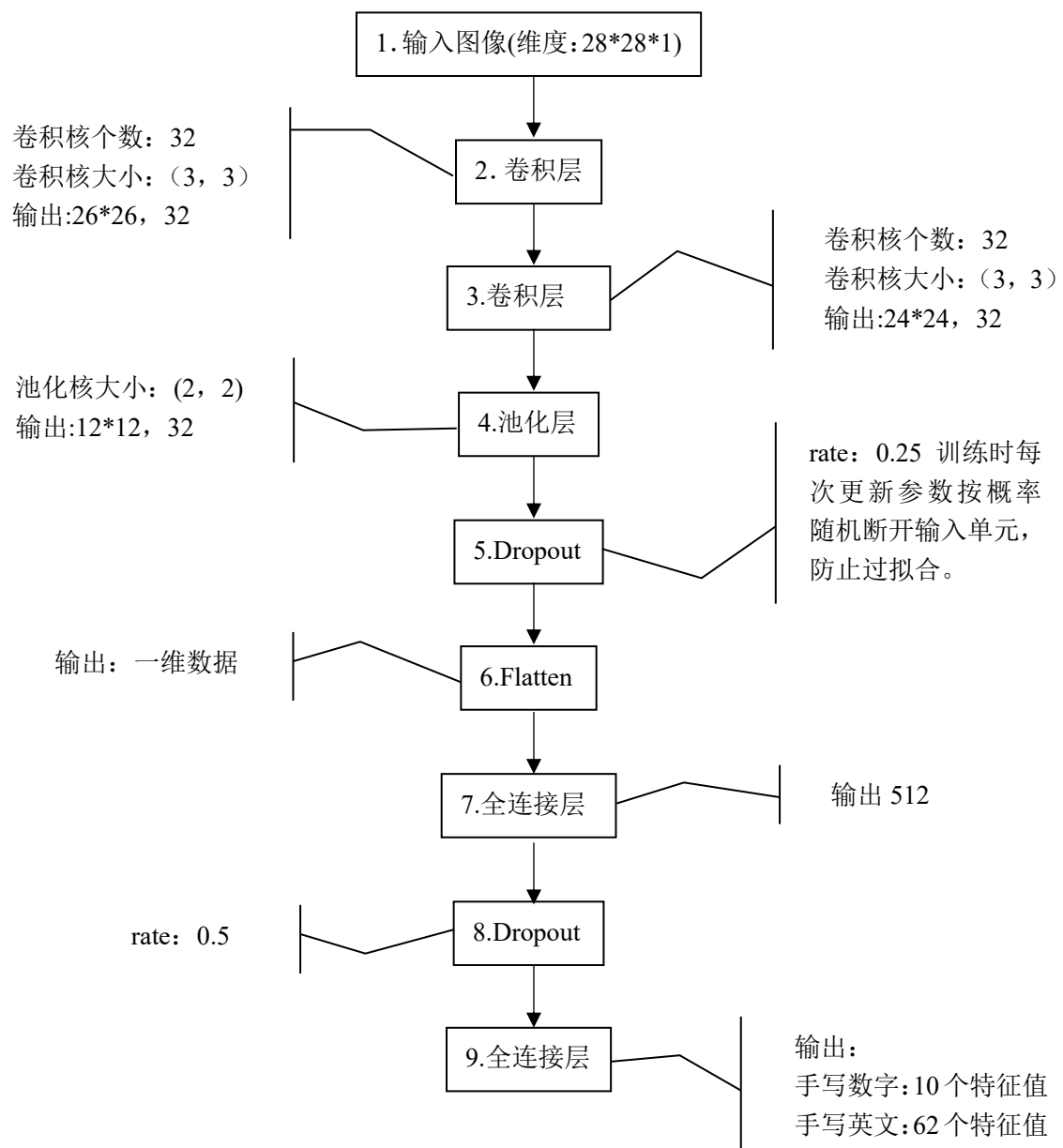


图 4.1 卷积神经网络构建



对图 4.1 卷积神经网络构建中的各层说明：

(1) 输入层：预处理后的 28\*28 灰度图像数据，无权值和偏置；

(2) 卷积层：卷积核大小 3\*3，输出特征图数量 32，卷积核种类 1\*32=32，输出特征图大小 26\*26。Tensor("conv2d\_1/Relu:0", shape=(?, 26, 26, 32), dtype=float32)

(3) 卷积层：卷积核大小 3\*3，输出下特征图数量 32，卷积核种类 32，输出特征图大小 24\*24  
Tensor("conv2d\_2/Relu:0", shape=(?, 24, 24, 32), dtype=float32)

(4) 池化层：池化核大小 2\*2，将使图片在两个维度上（竖直，水平）均变为原长的一半，输出下采样数量 12\*12。

Tensor("max\_pooling2d\_1/MaxPool:0", shape=(?, 12, 12, 32), dtype=float32)

(5) Dropout 层：设置神经元随机失活概率为 0.25，输出 12\*12

Tensor("dropout\_1/cond/Merge:0", shape=(?, 12, 12, 32), dtype=float32)

(6) Flatten 层：将输入的图像数据压成一维数据

Tensor("flatten\_1/Reshape:0", shape=(?, ?), dtype=float32)

(7) 全连接层：卷积核大小 1\*1，输出 512（自己设置的）

Tensor("dense\_1/Relu:0", shape=(?, 512), dtype=float32)

(8) Dropout 层：设置神经元随机失活概率为 0.25，输出 512

Tensor("dropout\_2/cond/Merge:0", shape=(?, 512), dtype=float32)

(9) 全连接层：输出特征图数量 62（手写英文字母，手写数字为：10）

Tensor("dense\_2/Softmax:0", shape=(?, 62), dtype=float32)

## 4.1 加载数据

关键代码：

# 加载 mat 格式数据集

mat = loadmat(mat\_file\_path)

# 加载训练数据样本

if max\_ == None:

max\_ = len(mat['dataset'][0][0][0][0][0][0]) # 求出数据的数据集中图像个数

training\_images = mat['dataset'][0][0][0][0][0][0][:max\_].reshape(max\_, height, width, 1) # 原始图像

training\_labels = mat['dataset'][0][0][0][0][0][1][:max\_] # 数据的标签

# 加载测试数据样本

if max\_ == None:

max\_ = len(mat['dataset'][0][0][1][0][0][0]) # 求出数据的数据集中图像个数

testing\_images = mat['dataset'][0][0][1][0][0][0][:max\_].reshape(max\_, height, width, 1) # 原始图像

testing\_labels = mat['dataset'][0][0][1][0][0][1][:max\_] # 数据的标签

其中数字和英文字母所对应的特征值数量是不同的，数字的特征值数量为 10 个。映射的格式为：字典：{key:value}，其中 key 代表的序号，value 是对应的 ASCII 值，如序号为 3 的对应的 ASCII 为 51，代表的是数字 3。{0: 48, 1: 49, 2: 50, 3: 51, 4: 52, 5: 53, 6: 54, 7: 55, 8: 56, 9: 57}

英文字母有 62 个为大写 A-Z 的序号 10 至 35(26 个)，小写 a-z 序号 36 至 61(26 个)，数字的序号 0 至 9（10 个）。映射的方式和上面的数字是一样的。例如序号为 5 对应的 ASCII 值为 53，代表的是数字 5。序号为 28 的对应的 ASCII 值为 82，代表的是大写字母：R。

{0: 48, 1: 49, 2: 50, 3: 51, 4: 52, 5: 53, 6: 54, 7: 55, 8: 56, 9: 57, 10: 65, 11: 66, 12: 67, 13: 68, 14: 69, 15: 70, 16: 71, 17: 72, 18: 73, 19: 74, 20: 75, 21: 76, 22: 77, 23: 78, 24: 79, 25: 80, 26: 81, 27: 82, 28: 83, 29: 84, 30: 85, 31: 86, 32: 87, 33: 88, 34: 89, 35: 90, 36: 97, 37: 98, 38: 99, 39: 100, 40: 101, 41: 102, 42: 103, 43: 104, 44: 105, 45: 106, 46: 107, 47: 108, 48: 109, 49: 110, 50: 111, 51: 112, 52: 113, 53: 114, 54: 115, 55: 116, 56: 117, 57: 118, 58: 119, 59: 120, 60: 121, 61: 122}

## 4.2 卷积神经网络的构建

# 构建顺序模型

model = Sequential()

'''

输入图像数据的 shape,默认数据为(28,28,1)第一个卷积层需要接受一个输入图像数据的参数，而后面各层可以自动推导出中间图像数据的 shape

'''

# 首先，第一个卷积层，构造 32 个滤波器，每个滤波器覆盖范围是 3\*3\*1

# 滤波器的挪动步长为 1，其图像四周补一圈 0，并用 relu 进行非线性变换

model.add(Convolution2D(nb\_filters, kernel\_size, strides=(1, 1), padding='valid',

input\_shape=input\_shape, activation='relu'))

model.add(Convolution2D(nb\_filters, kernel\_size, activation='relu'))

# 添加一层 MaxPooling，在 2\*2 的格子中取最大值

model.add(MaxPooling2D(pool\_size=pool\_size))

```
# 设立 Dropout 层，将 Dropout 的概率设为 0.25
```

```
model.add(Dropout(0.25)) # 神经元随机失活
```

```
# 把当前层节点展平，拉成一维数据，才能全连接
```

```
model.add(Flatten())
```

```
# 全连接层 1
```

```
model.add(Dense(512, activation='relu'))
```

```
model.add(Dropout(0.5)) # 随机失活
```

```
# 全连接层 2, Softmax 评分
```

```
model.add(Dense(nb_classes, activation='softmax'))
```

```
'''
```

最后编译模型，并设置模型的学习过程

1.loss 损失函数：目标函数，可为预定义的损失函数

2.optimizer 优化器：参数可指定为预定义的优化器名字，如 rmsprop、adagrad 或一个

Optimizer 类对象

3.指标列表：如果是分类问题，则一般该列表参数为 metrics=['accuracy']

```
'''
```

```
model.compile(loss='categorical_crossentropy',
```

```
              optimizer='adadelta',
```

```
              metrics=['accuracy'])
```

### 4.3 卷积神经网络的训练

```
'''
```

训练模型，放入批量样本，进行训练

batch\_size: 指定梯度下降时每个 batch 包含的样本数

nb\_epoch: 训练的轮数，nb 指 number of

verbose: 日志显示，0 为不在标准输出流输出日志信息，1 为输出进度条记录，2 为 epoch 输出一行记录

validation\_data: 指定验证集

fit 函数返回一个 History 的对象，其 History.history 属性记录了损失函数和其他指标的数值随 epoch 变化的情况，如果有验证集的话，也包含了验证集的这些指标变化情况

```
'''
```

```
model.fit(x_train, y_train,
```

```
        batch_size=batch_size,
```

```
        epochs=epochs,
```

```
        verbose=1,
```

```
validation_data=(x_test, y_test))
```

## 4.4 模型评估

手写数字模型评估:

```
score = model.evaluate(x_test, y_test) # 模型评估, 基于测试样本评价模型的准确度
D:\Program Files\Anaconda3\python.exe" I:/毕设/EMNIST/training.py -f ./matlab/emnist-mnist.mat
Using TensorFlow backend.
Train on 60000 samples, validate on 10000 samples
60000/60000[=====] - 139s 2ms/step - loss: 0.0184 -acc: 0.994 -
val_loss:0.0.250 - valacc: 0.9919
Test score: 0.025195021106
Test accuracy:0.9919
60000 个样本数据, 10000 个测试数据。迭代 10 次, 准确率达到 99.19%。
```

手写英文字母模型评估:

```
D:\Program Files\Anaconda3\python.exe" I:/毕设/EMNIST/training.py -f ./matlab/emnist-byclass.mat
Using TensorFlow backend.
Train on 697932 samples, validate on 116322 samples
697932 /697932 [=====] - 139s 2ms/step - loss: 0.0184 -acc: 0.994 -
val_loss:0.0.450 - valacc: 0.9825
Test score: 0.025195021106
Test accuracy:0.9825
697932 个样本数据, 116322 个测试数据。迭代 10 次, 准确率达到 98.19%。
```

## 4.5 网页端识别手写字符

### 4.5.1 加载训练过的模型

```
# 载入 YAML 和创建模型
yaml_file = open('%s/model.yaml' % bin_dir, 'r')
loaded_model_yaml = yaml_file.read()
yaml_file.close()
model = model_from_yaml(loaded_model_yaml)
# 将已训练过的模型加载
model.load_weights('%s/model.h5' % bin_dir)
```

### 4.5.2 浏览器网页端访问

运行网站应用程序:

```
(ENV) [root@CentOS7 EMNIST-master]# python server.py --host 0.0.0.0
```

\* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)

浏览器输入网址：<http://banglong.site:5000/>，访问手写字符识别应用程序。如下图 4.2 手写字符图像识别所示。

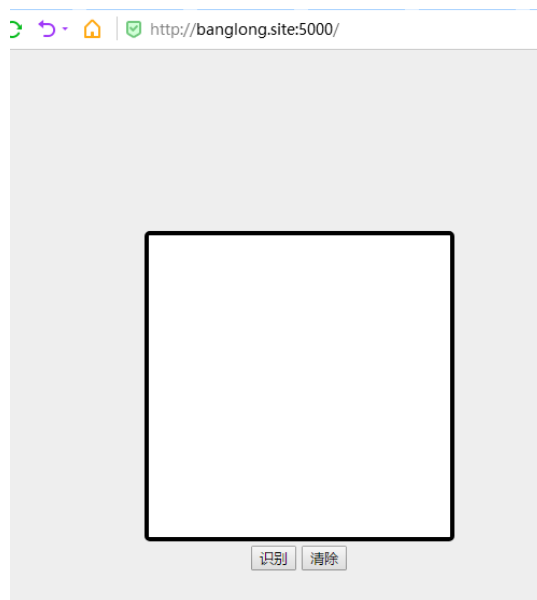


图 4.2 手写字符图像识别

### 4.5.3 识别应用预测结果

(1) 从网页输入图像数据到识别应用中

# 将网页端用户输入的画布图像使用 Ajax 传到 Flask 中（JS 代码），见图 4.3 网页端绘制手写图像。

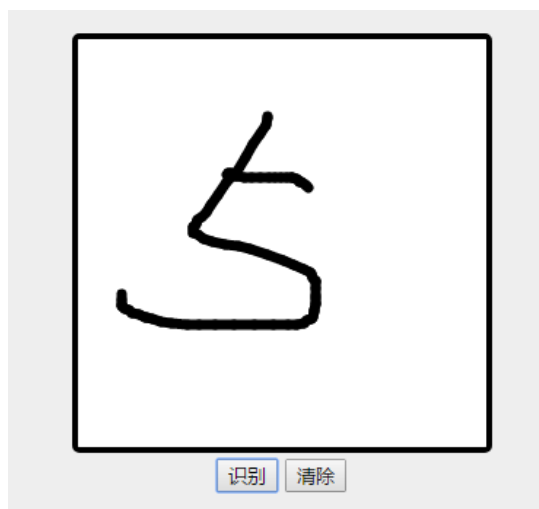


图 4.3 网页端绘制手写图像

# 将图像数据传入 Flask 的手写字符识别应用程序中:

文件 index.html 中的 JavaScript 主要代码如下:

```
$("#predict_button").click(function(){ //单击识别按钮

    var canvasObj = document.getElementById("canvas");

    var img = canvasObj.toDataURL('image/png');

    //canvas 提供了一个重要的方法 toDataURL(), 这个方法能把画布里的图案转变成 base64
    编码格式的 png, 然后返回 Data URL 数据。
```

```
$.ajax({
    type: "POST",
    url: "/predict/",
    data: img,
    success: function(data){
        $('#result').text('预测: ' + data.prediction);
        $('#confidence').text('相似度: ' + data.confidence + '%');}
});
```

(2) 解析通过 Ajax 使用 POST 方法传过来的图像数据

# training.py 主要代码:

```
"""
```

定义 parseImage()方法, 用来预处理网页端传过来的手写图像。

参数 imgData: 格式是 base64 编码的 png 图像。

```
"""
```

```
def parseImage(imgData):
```

```
    # 将画布 base64 位格式的图像保存成 output.png
```

```
    imgstr = re.search(b'base64,(.*)', imgData).group(1) # 使用正则表达式进行匹配
```

```
    with open('output.png','wb') as output: # 打开并写入 output.png 图像
```

```
        output.write(base64.decodebytes(imgstr))
```

```
    parseImage(request.get_data())
```

# 从网页记录下绘图的数据并保存成 output.png 图像, 存在服务器当前目录下。图像数据保存结果见下图 4.4。output.png 中图像的高度和宽度为 280\*280 像素, 其中一个像素点的取值范围是 0~255。

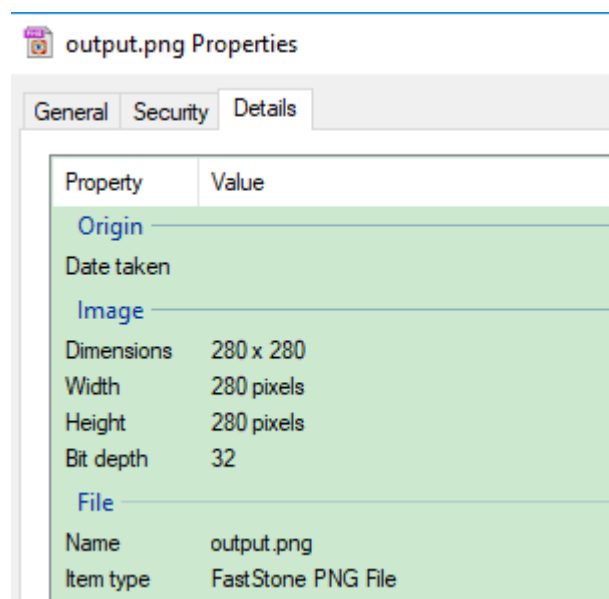


图 4.4 网页端输入的图像保存成 output.png

```
# 以每 8 位黑白模式（L）读取分析图像
```

```
x = imread('output.png', mode='L')
```

```
x = np.invert(x) #将图像数据转置，原图像是白底黑色，转为黑底白色
```

```
# 将已转置过的图像保存成 resized.png 见图 4.5
```

```
imsave('resized.png', x)
```

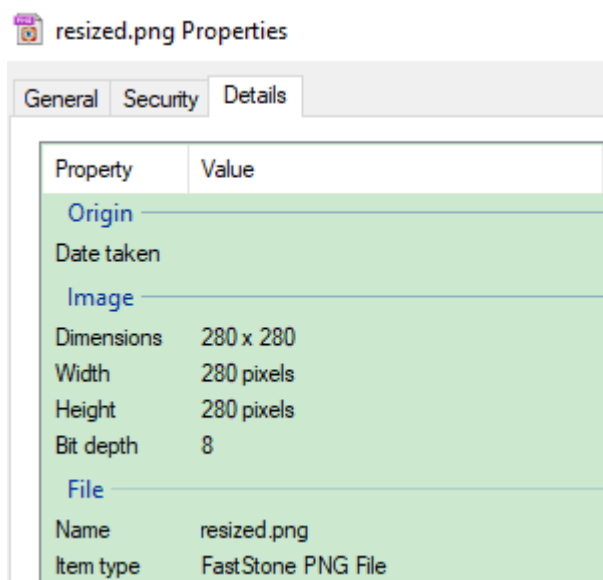


图 4.5 output.png 转置成黑底白字的图像

```
x = imresize(x,(28,28))
```

```
# 原始图像为 280*280，8bits 重构图像数据维度成一维的 28*28，1 通道，用于神经网络
```

```

x = x.reshape(1,28,28,1)
x = x.astype('float32')
# 统一格式
x /= 255

(3) 从已训练过的模型中预测结果

out = model.predict(x) # 输出 out

[[ 8.99958536e-07  3.87457334e-07  1.23956556e-06  9.16228890e-04
  1.73078922e-06  9.634920e-01  2.03575291e-05  4.30803411e-08
  1.17835616e-05  3.39387945e-04]]

# 生成 Json 格式的相似度最大的结果并返回给网页端，见图 4.6 手写数字识别结果

response = {'prediction': chr(mapping[(int(np.argmax(out, axis=1)[0]))]),
            'confidence': str(max(out[0]) * 100)[:6]}

return jsonify(response)

```

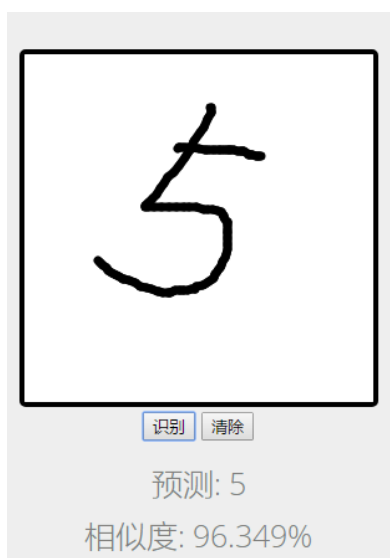


图 4.6 手写数字识别结果

以上便是整个实验的流程，下面展示一些手写字符的识别结果图。从下图 4.7 手写数字字符识别的结果可以看到对于一些特定的数字，例如 9，1，4，6，2 等这些数字的特征值较其它数字比较容易提取，实验的识别率通常能达到 90%。而像 0，8，3，5，7 这些数字的特征值比较容易相似，所以计算机有时会识别出错，识别率低且没能达到正确的识别结果。

而对于手写字母字符识别实验中，在原手写数字字符识别应用中增加了对字母字符的识别，识别过程既要数字进行识别，同时也要对英文字母识别。从下图 4.8 手写英文字符识别结果中可以



看到，与手写数字识别应用一样，识别相似特征值的字符时，准确率通常都不会太高。手写数字和手写英文字符有许多字符有太多相似的地方，例如数字 0 和字母 o、O（大写的字母 o）。有时书写者本意是想输入其中的一个，但由于太过相似，识别应用程序无法给出书写者想要的正确字符。这时就不能仅仅通过对单个字符的识别，而需要通过对于由字符组成的句子进行主义的识别，从而判断字符相对应的识别结果。

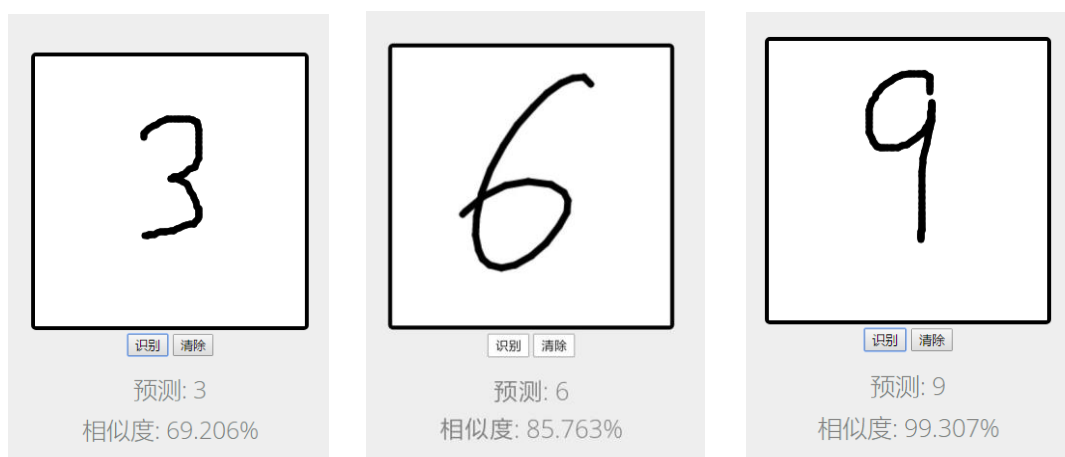


图 4.7 手写数字字符识别

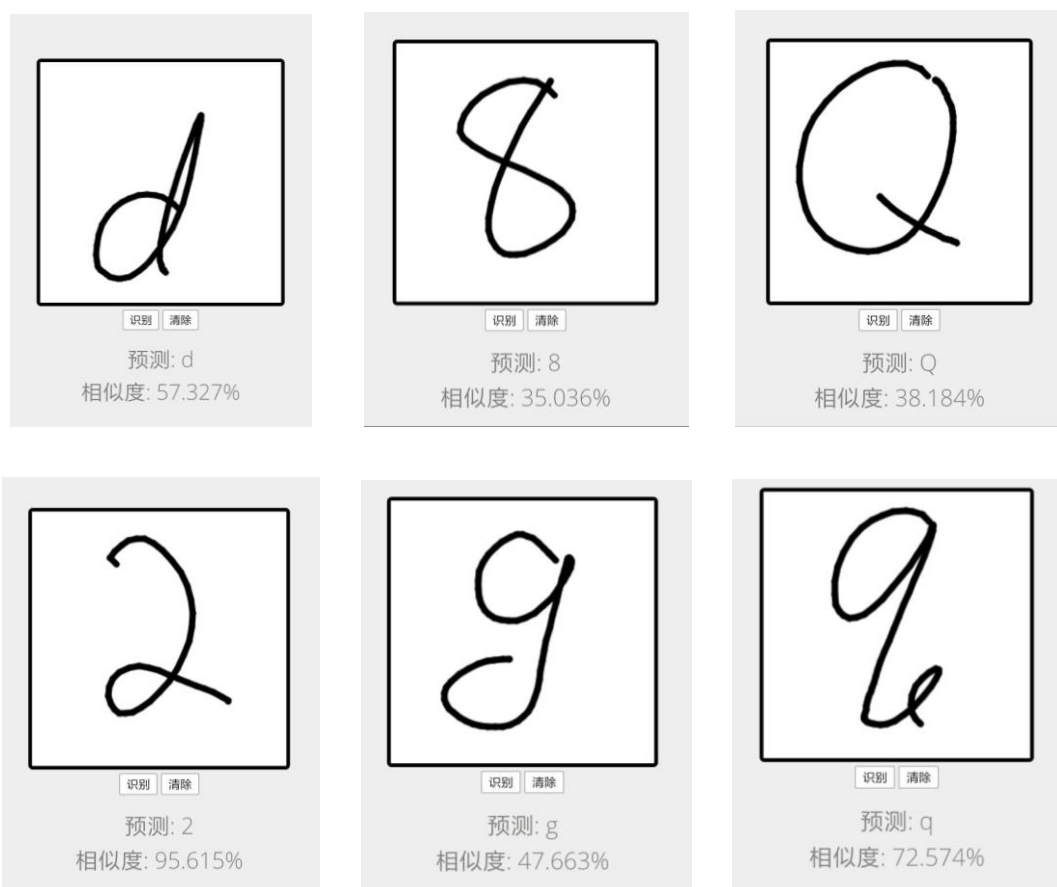


图 4.8 手写英文字符识别

## 第 5 章 总结

目前，人工智能已潜移默化的渗入到我们生活的各个方面，图像识别技术已被广泛运用于指纹，人脸，车辆牌照，手写体数字、邮政编码、汽车牌照、汉字、条形码识别等领域。本文使用 Keras 作为卷积神经网络框架，后端采用 Tensorflow 进行神经网络卷积运算。公共数据集文件采用了 MNIST 和 EMNIST。手写字符识别应用程序采用 Flask 网站框架和构建经典的卷积神经网络对手写数据进行预测。

作为刚接触到图像识别和神经网络的初学者，从实验前期的准备至后期实验的完成，其中有很多知识理解的不深入，自知本身还需多多学习。对于卷积神经网络方面的知识，很多都是通过网上查阅资料，发现了许多研究者和前辈在很久之前便提出了比较好的数学模型与其验证方法。本文的实验所用到的思想很大一部分受自前辈们研究成果的影响。本文的实验还有提高的空间，其卷积神经网络算法还要进一步完善，而且数字和英文字母有时不易区分。例如写了一个数字 0，计算机并不知道这是数字 0 还是字母大写的 o，所以对于字符识别不能仅仅通过对单个字符的识别，还要结合由字符组成的词语，句子和语意进行预测才能达到较好的期望值。作为使用 Flask 网站做平台实现神经网络，后期可以采用多并发的，高性能的 nginx 服务端。

在撰写本文时，我收获到了有关论文写作的规则和框架技巧。通过本次论文的设计，学到了不仅仅是对卷积神经网络的单一方面的认识与了解，更多的是思路，对问题处理的方法。在大学期间所学的专业知识仍然有限，像计算机这种专业，每天都会有新的技术更新，旧的就会被新知识所淘汰，社会是需要自学能力强的，今后，对于今后的神经网络和图像识别方面的工作，自己会继续保持着努力学习的心态，去努力不断的充实自己。

## 参考文献

- [1] 许可.卷积神经网络在图像识别上的应用的研究[D].浙江大学,2012.
- [2] 严军.空间手写识别特征提取研究[D].浙江大学,2012.
- [3] 应攀.3D 手写识别特征集取技术研究[D].浙江大学,2013.
- [4] 户保田.基于深度神经网络的文本表示及其应用[D].哈尔滨工业大学,2016.
- [5] 冯子勇.基于深度学习的图像特征学习和分类方法的研究及应用[D].华南理工大学,2016.
- [6] 王强.基于 CNN 的字符识别方法研究[D].天津师范大学,2014
- [7] 李卫.深度学习在图像识别中的研究及应用[D].武汉理工大学,2014
- [8] 刘荣荣.基于卷积神经网络的手写数字识别软件的设计与实现[D].内蒙古大学,2015
- [9] 王瑞.基于卷积神经网络的图像识别[D].河南大学,2015
- [10] 李明威.图像分类中的卷积神经网络方法研究[D].南京邮电大学,2016
- [11] 王斐.卷积神经网络在手绘草图识别中的应用研究[D].安徽大学,2016.
- [12] 苏哲文.手写汉字图像动态信息恢复方法研究[D].华中科技大学,2009..
- [13] 段萌.基于卷积神经网络的图像识别方法研究[D].郑州大学,2017
- [14] 吴正文.卷积神经网络在图像分类中的应用研究[D].电子科技大学,2015.
- [15] 何西麟.基于深度学习的手写体字符识别研究与实现[D].中山大学,2015.
- [16] 陈先昌.基于卷积神经网络的深度;学习算法与应用研究[D].浙江工商大学,2014.
- [17] 柳振东.卷积神经网络在图像分类中的研究与应用[D].中国民航大学,2017.
- [18] 刘鹏宇.基于内容的图像特征提取算法的研究[D].吉林大学,2004.

## 致谢

本次毕业设计的基于卷积神经网络的图像识别程序和论文都已经完成，首先要向我的指导老师王寅同和专业课老师们表示衷心的感谢。论文的顺利完成，要归功于指导老师与专业老师们的悉心指导，正是他们的不断的督促、指导以及关怀，我才能按时完成。我从尊敬的导师身上，我学习了的不仅仅是扎实与深广的专业知识与技能，更学会的是做人与处理的道理，在此我要向我的导师致以最衷心的感谢和深深的敬意。

其次，我要感谢与我相处一起学习的同学们和朋友们，遇到问题时我都会与大家一起讨论，分析问题的原因，找出解决的办法，正是因为由于你们的存在，我才能顺利的完成基于卷积神经网络的图像识别研究的论文。我还要感谢论文中涉及的学者和前辈们，从他们的研究成果中我得到了帮助和启发，否则我将难以完成本文。

最后，我要感谢学校的所有的老师们，感谢四年来你们对我的关怀和教育，如果不是你们这四年来不断的授业传道解惑，我不会成长的像今天这样的优秀，是你们让我获取了人生最宝贵的财富，领悟了知识的力量，感悟到了人生的真谛，真的很感谢你们。

大学的时光即将结束，感谢这四年来在我生活中出现的所有人，是你们让我的大学生活充满了色彩，是你们让我体会了人生的酸甜苦辣，感谢大家。