# 長安大學

## 二〇一九届毕业设计

# 基于 TensorFlow 框架的手写数字识别系统


学　　院：信息工程学院

专　　业：计算机科学与技术

姓　　名：宋启迪

学　　号：**201524020401**

指导教师：南春丽


完成时间：**2019** 年 **6** 月 **10** 日

二〇一九年六月

装 订 线

# 摘 要

时下人工智能蔚然成风，作为新型生产要素，在各个领域都极大地推动了经济发展，预示着社会经济将取得巨大增长和和人类潜能将得到深度开发。实现人工智能的手段即为机器学习，而机器学习领域中的一个重要问题就是手写字符的识别。手写数字识别的应用包括邮件分拣，银行支票处理，表格数据输入等。问题的核心在于开发能够识别手写数字的有效算法的能力，并且用户通过该方式提交扫描仪，平板电脑和其他数字设备来完成手写数字的识别。但因个人手写数字特征的千差万别，造成了现有手写数字识别系统的识别准确率普遍较低。

本文旨在通过使用 Google 研发的 Tensorflow 人工智能框架，分别搭建 Softmax 回归模型和卷积神经网络模型，并比较两者的识别率高低。本系统使用的手写数字数据集是 MNIST 数据集，该数据集分为三部分：55000 行训练数据集、10000 行测试数据集、以及 5000 行的验证数据集。每个测试集的图像的长和宽各为 28 个像素点，测试集的标签表示该图像是哪个数字。经过搭建、训练、使用两个模型，Softmax 回归模型和卷积神经网络模型对手写数字的识别率分别为 91.92%和 99.13%，卷积神经网络的识别率已经达到了可以实际应用的程度。本文为人工智能的手写数字识别的发展提供了一定的理论价值和科研依据。

**关键词：**机器学习，手写数字识别，TensorFlow，Softmax，CNN

装 订 线

# ABSTRACT

Nowadays, artificial intelligence has become a common and hottest trend. As a new type of production factor, it has greatly promoted economic development in various fields, indicating that the social economy will achieve tremendous growth and human potential will be deeply developed. The ways to achieve artificial intelligence is machine learning, and an important issue in the field of machine learning is the recognition of handwritten characters. Applications for handwritten digit recognition include mail sorting, bank check processing, form data entry, etc. At the core of the problem is the ability to develop efficient algorithms that recognize handwritten numbers, and in this way users submit it to scanners, tablets and other digital devices to perform handwritten digit recognition. However, due to the wide variety of personal handwritten digital features, the recognition accuracy of existing handwritten digit recognition systems is generally low.

This paper aims to build a Softmax regression model and a convolutional neural network model by using the Tensorflow artificial intelligence framework developed by Google, and compare the recognition rates of both model. The handwritten digital dataset used in this topic is the MNIST dataset. The dataset is divided into three parts: 55,000 rows of training datasets, 10,000 rows of test datasets, and 5000 rows of validation datasets. The length and width of the image for each test set are each 28 pixels, and the label of the test set indicates which number the image is. After constructin、training and applying those models, the recognition rates of handwritten numbers by Softmax regression model and convolutional neural network model are 91.92% and 99.13%, respectively. The recognition rate of convolutional neural network has reached the level of practical application. This paper provides a certain theoretical value and scientific research basis for the development of artificial intelligence handwritten digit recognition.

**KEY WORDS**: Machine Learning,Handwritten digits recognition, TensorFlow, Softmax, CNN,

装 订 线

# 目 录

装 订 线

# 第一章 绪论

## 1.1 课题研究背景及意义

自从 AlphaZero 击败人类顶尖围棋高手后，人们关于人工智能的讨论就喋喋不休，从而出现了人工智能有益论和人工智能有害论两派。有许多人人认为人工智能会使一大批人失去他们所从事的行业，无情的被机器人取代。而更多人则认为，人工智能可以帮助我们创造更多的财富，创造更美好的明天。而目前我所能接触到的人工智能技术，都为人类的发展做出了极大的贡献，人类不能因噎废食。例如在图像识别领域，使用 Google 的 TensorFlow 人工智能框架可以很方便的搭建一个卷积神经网络，并且立即投入使用。

手写数字识别重大研究价值意义在于人机交互方向，提高人机交互的流畅性、方便性以及似人性。在于数字信息自动处理领域，提高处理效率，节省物力财力，加快信息传输，创造出无数的财富。手写体是人类在交流中使用的最常见的标准和常规媒介。即使引入了诸如键盘，声音命令等新技术，它也是一种有效且高效的信息记录方式。通常，手写识别系统是一种用于识别人类手写的任何语言的机制，无论是扫描的手写图像或者在电子设备上使用手写笔进行实时手写，也可以分别称为离线和在线手写[1]。此外，该系统的应用可分为数字，字符和英文字母三种。它广泛用于许多应用，如语言翻译，银行支票和关键字定位。手写数字识别系统的理论价值如下:

1)阿拉伯数字官方符号，被世界各国通用，且应用于生活的方方面面。因此研究手写数字识别具有很高的应用价值。

2)相较于汉字、英文单词等常用字符，数字识别的类型数较小，有助于做深入分析以及验证一些新的理论。

3)虽然人们已经对模式识别的手写数字领域进行了相当长时间的研究，但是手写数字识别系统仍然有很多不足，在某些情况下，还无法与人类的识别能力相提并论，人工智能还有很长的路要走。

4)手写数字识别与许多领域都有关联性，尤其是中英文拼音的识别。事实上，很多研究人员就是同时研究这两者的，由于相似性较强，结合在一起研究更易取得研究成果。

在大规模的数据统计中(如:人口普查等)需要输入极大量的数据，之前则完全要靠手工输入。而近几年来在这类任务中采用手写识别的技术已成为一种趋势。因为数据经常会大规模集中录入系统，所以可以在数字录入时加一些书写限制条件，或者在用户手写数字输入时给出限制提示，从而避免手写体数字的千差万别[2]，得到良好的手写数字源，用于机器学习和自动识别。当前绝大部分操作系统，都会在用户录入数据时增加限制条件，让用户按照规范进行手写数字，另外还采用合适的图形用户界面对识别结果做全面的检查，最终保证正确无误。

银行、表格、证券是手写数字识别可以大放异彩的又一个领域。随着我国经济的迅速发展，

无时无刻不产生大量的账单、支票、税务报表等等手写数据[3]。与上面提到的统计报表[4]处理相比，在这个领域的应用难度更大，因为对识别的精度要求更高，处理的表格种类更多等，这样对识别及预处理的核心算法要求也大大提高了。如果在这个方向上都能做到机器自动识别的话，就能够取得巨大的经济效益。

## 1.2 研究状况

随着科学技术与发展, 计算机视觉已经广泛运用于图像理解、地图导航、医疗制药、无人机和无人驾驶汽车等领域。而这些应用的核心技术就是图像处理[5]、图像定位[6]和图像分类[7]等视觉识别任务。

模式识别是二十世纪五十年代初迅速蓬勃发展起来的一门学科。模式识别研究的是如何用机器来实现高等动物对事物的学习、判断和识别能力，因而受到了国内外众多科研人员的关注，成为人工智能研究的一个重要方面。一个模式识别系统的基本功能是对系统所要处理的模式归属于哪一类做出判断，从该系统的模式输入到系统做出判别之间，主要包括数据检测、图像预处理、特征值的提取和分类四大环节。

字符识别又是模式识别领域中的一个极受关注的问题。问题本身的难度使之成为一个兼具趣味和挑战性的课题;另一方面，因为字符识别不是一项单独的应用技术，其中包含的模式识别领域中其他分支都会遇到的一些基本和共性的问题。

从 19 世纪九十年代末期到目前为止，世界各国的科学家发表了很多关于 MNIST 手写数字识别的文章。从新闻上也可以经常看到这个领域的研究进展。而识别率和识别速度是衡量识别方法优劣的两个关键指标。可以看出，在线性分类器，KNN，SVM，NN 和 CNN 中，目前识别率最高的模型是 CNN。且针对 MNIST 数据集，没有特征提取方面的最优解。直接用 LeNet-5 结构的 CNN，就能得到比较好的结果。

## 1.3 研究内容

本课题主要实现了两种手写数字识别模型：Softmax 回归模型核卷积神经网络模型，并运用 Flask 框架展示在网页当中，Softmax 回归模型和卷积神经网络模型对手写数字的识别率分别为 91.92%和 99.13%。本文的研究内容可总结如下所示：

1） 搭建 TensorFlow 框架，配置 Python 环境，下载 MNIST 手写数字数据集。

2） 学习 Softmax 回归算法，实现 Softmax 回归模型，并进行训练和评估。

3） 学习卷积神经网络的基本原理，搭建共计七层的卷积神经网络模型，不断优化模型参数，训练 CNN 模型，并进行评估。

4） 运用 Flask 框架，通过网页交互，将输入的数据传入后台处理，并返回两个模型的识别

结果，最终显示在网页中。

## 1.4 研究方法

实验研究表明，若手写体数字没有限制，几乎可以肯定没有一劳永逸的方法能同时达到 90%以上的识别率和较快的识别速度。因此，这方面的研究向着更复杂更综合的方向发展。例如人工智能中的专家系统、人工神经网络已经开始应用于手写体数字识别的研究当中。在手写数字识别的发展中，神经网络和多种专家系统的结合是值得探究的方向。模式特征的不同，其决策方式也会不同。可将模式识别的方法大致分为 5 大类[8]。这五类方法各有各自的特点，各有各自的适用条件，最后都能实现手写数字的识别。这五类方法分别为：

1. 句法结构方法
2. 统计模式法
3. 逻辑特征法
4. 模糊模式方法
5. 神经网络方法

下面简单介绍一下这五类方法的适用条件。句式结构法比较简单直观，可以直接反映事物的本质特征，但难点在于不易提取神经元且稳定性较差。统计法用于统计事物的各个特征，优点是比较方便简洁，且鲁棒性较好。但是统计法没有充分利用模式的结构，难以从各个模块之间进行比较。神经网络方法常用人工神经网络方法实现模式识别。一些环境信息可以处理的问题非常复杂，背景情况也不太明了，推理规则没有明确的定义，使得样本存在较大的缺陷和失真。神经网络方法的缺点在于其模型不断丰富和完善。目前，还没有足够的模式可供查明。

神经网络方法允许样本具有大的缺陷和扭曲。它具有运行速度快，自适应性能好，性能高等特点。它还可以快速同时处理大容量的数据，并行的处理数据，也因此具有超高速度的特点。并且，网络的最终输出是由所有神经元共同作用的结果，一个神经元的错误对整体的影响微乎其微，可以忽略不计。所以其容错性也非常的好[9]。基于以上的考虑，本文的手写数字识别采用了卷积神经网络的方法。

## 1.5 论文组织结构

本文共 6 个章节，其结构安排如下：

第 1 章为绪论，介绍了本课题的研究背景及其研究意义、当前的研究状况、研究内容以及研究方法。此外，还简单描述了五种模式识别常用的方法，并介绍这五种方法各自的使用条件及优缺点。

第 2 章为相关技术介绍，首先介绍了 Google 开发的机器学习框架 Tensorflow，并简单论述了

Tensorflow 的工作原理。紧接着介绍了本系统所选择的编程语言 Python 的优缺点以及选择这么语言的原因。然后介绍了 Python 的界面开发工具 Tkinter。最后介绍了 MNIST 手写数字数据集以及该数据及的文件格式。

第 3 章为开发环境配置。本章介绍了本机的硬件开发环境、本系统所选用的集成开发环境 Pycharm、Python3.x 的安装于环境配置、Tensorflow-GPU 的安装及环境配置以及 Tensorflow 的集成配置平台 Anaconda。

第 4 章为系统的设计与实现。本章第一节介绍了 Softmax Regression 算法、模型的训练以及模型的评估。本章第二节介绍了卷积神经网络模型参数的设计和实现、模型的结构和训练过程。之后介绍了本课题设计的图形用户界面以及前台与后端进行数据交换所用的 Flask 框架、模板引用等技术。

第 5 章为系统测试。本章介绍了几个测试案例来测试系统的健壮性鲁棒性。其中既由成功的测试，也有失败案例。

第 6 章为展望与总结。介绍了手写数字识别的当下与未来，并对未来一段时间的机器学习发展进行了展望。

# 第二章 相关技术介绍

本章介绍了本课题所使用的相关技术，并介绍了相关技术的工作原理、优缺点等。相关技术包括 TensorFlow 框架、Python 语言、Tkinter 相关控件及特性以及 MNIST 数据集等。

## 2.1 TensorFlow 框架

### 2.1.1 TensorFlow 框架介绍

TensorFlow 是一个用于机器学习的端到端开源平台。它拥有全面，灵活的工具，库和社区资源生态系统，可让研究人员推动 ML 的最新技术，开发人员可轻松构建和部署 ML(Machine Learning)驱动的应用程序。它还是一个开源软件库，用于语义理解和感知方向的机器学习。TensorFlow 框架是由谷歌人工智能团队开发，用于 Google 相关产品及功能的开发与研制。如语音识别、谷歌邮件、谷歌地图和谷歌搜索引擎。

### 2.1.2 TensorFlow 工作原理

TensorFlow 是一个采用数据流图用于数值计算的开源软件库。节点一般在图中表示数学操作，图中的线则表示在节点间的输入/输出关系，也就是张量。张量从图中流过的直观图像是这个工具取名为“Tensorflow”的原因[10]。

## 2.2 Python 语言

### 2.2.1 Python 介绍

Python 是一种广泛使用的通用高级编程语言。它最初由 Guido van Rossum 于 1991 年设计，由 Python Software Foundation 开发。它主要是为了强调代码可读性而开发的，其语法允许程序员用更少的代码行表述概念。 Python 有两个主要的 Python 版本：Python 2.x 和 Python 3.x。两者差别较大，本文使用的是 Python3.x。

起初，自动化脚本常用 Python 来编写。之后随着 Python 版本的不断升级以及功能的不断完善，它越来越多被用于大型的、独立的项目开发。Python 除了极少数事情不能完成之外，其他基本上可以说全能。多媒体应用、机器学习、人工智能、系统运维、黑客编程、图形处理、爬虫编写、数据库编程、pymo 引擎、文本处理等等都可以用 Python 来实现。Python 常见应用如图 2.1 所示：

### 2.2.2 Python 优缺点介绍

Python 的优点很多，简单的可以总结为以下几点。

1) 简单和明确，做一件事只有一种方法。
2) 学习曲线低，跟其他很多语言相比，Python 更容易上手。
3) 代码有着严格的编写规范，使用 Tab 键来控制结构
4) 解释型语言，天生具有平台可移植性。
5) 学习曲线低，非专业人士也能上手，支持面向对象和函数式编程

Python 的缺点主要集中在以下几点。

1) 执行效率略低于 C 语言，速度略慢
2) 代码无法加密，但是现在的公司很多都不是卖软件而是卖服务
3) 在开发时可以选择的框架太多（如 Web 框架就有 100 多个），有选择的地方就有错误。

## 2.3 Tkinter

### 2.3.1 Tkinter 介绍

Tkinter 模块是 GUI 工具包的标准 Python 接口。Tk 和 Tkinter 都可以在大多数 Unix 平台上以及 Windows 和 MAC 操作系统上使用。从 8.0 版开始，Tk 在所有平台上都提供原生外观。Tkinter 包含着许多模块，Tk 接口由名为_tkinter 的二进制扩展模块提供。它通常是一个共享库（或 DLL），但在某些情况下可能与 Python 解释器静态链接。公共接口通过许多 Python 模块提供。最重要的接口模块是 Tkinter 模块本身。要使用 Tkinter，所需要做的就是导入 Tkinter 模块：

```
import Tkinter
```

或者更常用的是：

```
from Tkinter import *
```

### 2.3.2 Tkinter 模块的 GUI

在 Python 中，常用 tkinter 来开发图形用户界面。Tk 是一个工具包，它提供了跨平台的 GUI 控件，开发图形用户界面十分方便快捷。基本上使用 tkinter 来开发 GUI 应用需要以下 5 个步骤

1) 将需要的 tkinter 模块导入进开发环境中
2) 创建顶层窗口，并在这个顶层窗口开发 GUI
3) 添加 GUI 组件，并将组件放在合适的位置
4) 编写响应函数，将函数与需要响应的按钮绑定
5) 进入 main loop。

### 2.3.3 Tkinter 组件

Python 在 GUI 方面并不强，相比 wxpython 而言，tkinter 内置于 python 库中，无需另外安装，同时基本的控件也能满足基本开发需求，下面介绍 tkinter 的基本用法。

Tkinter 的提供各种各样的控件，例如菜单、按钮、消息、输入控件，便于开发同行用户界面。常用的控件及其描述如表 2.1 所示：

表 2.1 Tkinter 常见组件及介绍

| 控件 | 描述 |
|---|---|
| Button | 按钮控件:在程序中显示按钮。 |
| Canvas | 画布控件:画布本身没有绘图能力，它是图形的容器 |
| Checkbutton | 多选框控件:提供多项选择，选定后再次点击即可取消。 |
| Entry | 输入控件:用于输入数据。 |
| Label | 标签控件:用于在框架中显示标签。 |
| Menu | 菜单控件:显示菜单栏，可添加菜单选项 |
| Message | 消息控件:可以显示一行或多行文本，能自动换行和调整尺寸。 |
| Radiobutton | 单选按钮:为用户提供两个或多个互斥选项，只能选其一。 |

标准属性指的是 tkinter 控件的共同属性。例如控件的颜色、字体、大小、格式等等。Tkinter 标准属性介绍如表 2.2 所示：

表 2.2Tkinter 标准属性及其介绍

| 属性 | 描述 |
|---|---|
| Dimension | 控件大小 |
| Color | 控件颜色 |
| Font | 控件字体 |
| Anchor | 锚点 |
| Reliel | 控件样式 |
| Bitmap | 位图 |
| Cursor | 光标 |

Tkinter 包括三种几个管理类：pack、grid、place。这三种方法都可以管理整块空间区域。最常用的是 pack 和 grid 类。

| 几何方法 | 描述 |
|---|---|
| Pack() | 在二位网格中组织窗口部件，类似于新闻的排版 |
| Grid() | 几何管理器，将窗口部件包装到父部件中 |
| Place() | 可自由指定每个部件的像素位置，也因此容易出现布局混乱 |

## 2.4 MNIST 数据集

### 2.4.1 MNIST 数据集介绍

MNIST（Mixed National Institute of Standards and Technology database）是一个非常庞大的手写数字数据库. MNIST 数据集的官网是 YannLeCun website[11]。该网站提供了一份 Python 源代码用于自动下载和安装这个数据集。可以直接复制粘贴到代码文件里面，用于导入 MNIST 手写数据集。

```
import inputs_data
mnists = inputs_data.read_datas_sets("MNISTS_datas/", one_hots=True)
```

下载下来的数据集可被分为三部分：55000 个训练数据集（mnist.train），10000 个测试数据集 (mnist.test)，以及 5000 个验证数据集（mnist.validation）。MNIST 数据集的划分很重要，因为在机器学习模型设计时，必须提供一个单独的测试数据集，不用于训练而是用来评估这个模型的性能，从而更加容易把设计的模型推广到其他数据集上。这个数据集不需要很大，能体现出训练的成果即可。

MNIST 数据集的每一个单元都由两部分构成：第一部分是手写数字的图片，命名为“images”。第二部分是该单元的手写数字图片对应的标签，命名为“labels”。训练集和测试集都由这两部分组成。例如训练集的图片时 mnist.train.images,而训练数据集的标签则为 mnist.train.labels。图片的长和宽均为 28 像素点，每张图片总共 28*28=784 个像素，可以用长度为 784 的数字数组表示这张图片。标签总共有 10 种可能（0~9），因此可以使用长度为 10 的数组表示标签。例如：MNIST 数据集中某图片矩阵图如图 2.2 所示：

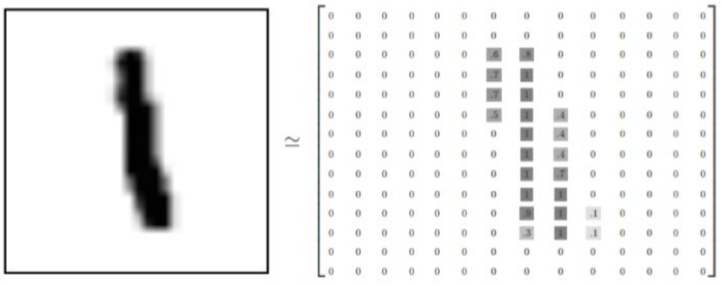


图 2.1 数字“1”矩阵图

将这个数组展开成向量（Vector），在展开时不需要考虑展开的行列顺序，只要保持各个图片采用相同的方式展开即可。从上图可以观察得出：MNIST 数据集的图片就是展开在 784 维向量空间里面的点, 结构并不复杂。

但是展开图片成为一维数组后，会丢失掉图片的二位平面信息，并不是很理想。但好在本文

介绍的 Softmax 回归模型和卷积神经网络模型比较简单，并不会利用这些二维信息。因此，在手写数字训练集中，MNIST.train.images 是一个形状为[55000,784]的二维张量。第一维度表示共有 55000 张图片以备训练，这一维的数字就是图片的序列号，可以用第一维数组来索引图片。第二位度表示每个图片有 784 个像素点。在此张量里的每一个元素，都表示某张图片里的某个像素的强度值，值介于 0 和 1 之间。

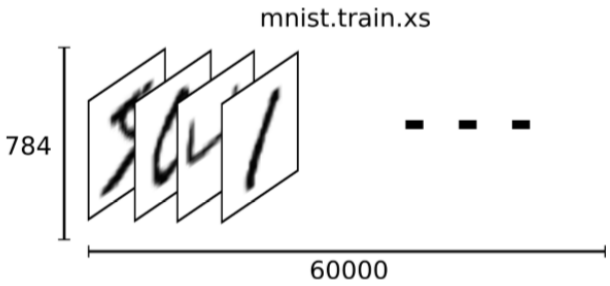



图 2.2 MNIST 数据集的训练图片

相对应的 MNIST 数据集的标签是一个介于 0~9 的数字，用来描述给定图片里表示的数字。除了某一位的数字是 1 以外其余各维度数字都是 0。所以数字 n 将表示成一个只有在第 n+1 维度（因为标签数组是从 0 开始的）数字为 1 的 10 维数组。因此 train.labels 是一个[55000, 10]的二维数组[12]。例如，标签 0 将表示成[1,0,0,0,0,0,0,0,0,0]。如下图所示：

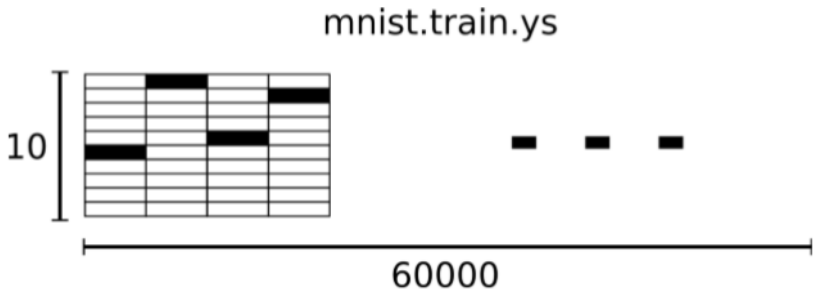



图 2.3 MNIST 数据集的训练标签

### 2.4.2 MNIST 数据集的文件格式

MNIST 数据集用文件格式存储，用于存储二维矩阵和矢量信息。该数据集文件中的所有整数都以大多数非英特尔处理器使用的 MSB 优先（高端）格式存储。英特尔处理器和其他低端机器的用户必须翻转标头的字节。MNIST 数据集包含 4 个文件，如下所示：

train-images-idx03-ubytes: training set images

train-labels-idx01-ubytes: training set labels

t10k-images-idx03-ubytes: test set images

t10k-labels-idx01-ubytes: test set labels

MNIST 数据集包含 60000 个训练集和 10000 个测试集。其中，训练集包含 55000 个训练示例，测试集的最后 5000 个示例取自原始 NIST 训练集，用于测试训练集的效果。

1) 训练集标签文件(train-labels-idx01-ubytes):

| [offset] | [type] | [value] | [description] |
|---|---|---|---|
| 1000 | 32 bit integer | 0x00000800(2048) | magic number |
| 1004 | 32 bit integer | 60000 | number of items |
| 1008 | unsigned byte | ?? | label |
| 1009 | unsigned byte | ?? | label |
| ..... | | | |
| xxxx | unsigned byte | ?? | label |

标签的取值范围是 0 到 9.

2) 训练集图像文件 (train-images-idx03-ubytes):

| [offset] | [type] | [value] | [description] |
|---|---|---|---|
| 1000 | 32 bit integer | 0x00000803 | magic number |
| 1004 | 32 bit integer | 60000 | number of images |
| 1008 | 32 bit integer | 28 | number of rows |
| 1012 | 32 bit integer | 28 | number of columns |
| 1016 | unsigned byte | ?? | pixel |
| 1017 | unsigned byte | ?? | pixel |
| ..... | | | |
| xxxx | unsigned byte | ?? | pixel |

784 个像素点按行排列，每个像素点的值介于 0~255 之间，0 表示白色，255 表示黑色

## 2.5 本章小结

本章第一节介绍了 TensorFlow 框架以及 TensorFlow 的工作原理。第二节 Python 语言及其优缺点。第三节介绍了图形用户界面的开发工具 Tkinter，以及其相关的常用组件、常用标准属性、几何状态管理方法和控件选项。第四节介绍了本次系统选取的数据 MNIST 数据集以及该数据集的两个文件格式。

装 订 线

# 第三章 开发环境配置

## 3.1 硬件设备信息

操作系统：Windows10 专业版 1803

处理器：Intel(R)Core(TM) i7-6700HQ CPU 2.60GHz

显卡：NVIDIA 960M GPU

RAM：8GB

## 3.2 Pycharm IDE

PyCharm 是一个用于计算机编程的集成开发环境，主要用于 Python 语言开发，由捷克公司 JetBrains 开发，提供代码分析、图形化调试器，集成测试器控制系统，并支持使用 Django 进行网页开发。此外，PyCharm 还是一个跨平台的集成开发环境，拥有 Microsoft Windows、macOS 和 Linux 版本的编码协助。下面简单介绍一下 Pycharm 的三个优点：

1) 易于上门

使用 PyCharm JDK 开始学习 Python 不需要以前的编程经验。PyCharm JDK 提供了学习内置 Python 所需的一切。

2) 专业的环境

PyCharm 集成开发环境是基于 IntelliJ 平台的，它有着丰富的编码功能，如智能代码完成，代码检查，可视化调试器等，不仅可以提高您的学习效率，而且可以帮助您轻松无缝地切换到其他工作环境。

3) 智能编译器

充分利用特定于语言的语法和错误突出显示来避免代码错误。了解如何使用代码格式设置代码样式，还可以在代码完成和快速文档的支持下发现 Python 编码错误并及时纠正。

## 3.3 Python3.x 的安装及环境配置

### 3.3.1 Python 安装

1) python 下载：

打开浏览器，输入网址 http://www.python.org/，点击“下载 Python3.7.3”即可下载 python 的安装包。下载 Python 安装包如图 3.1 所示：


图 3.1 Python 安装包下载

2) 解压安装包，双击运行，进入安装向导

3) 选择安装目录。例如：D:\Python36\

4) 选择 Add python.exe to Path>>Entire feature will be installed on local hard drive

5) 点击“Next”，继续下一步安装操作。

6) 检查安装是否成功。 按 Win+R 键，输入 cmd，进入控制台。在控制台下输入 python，若返回 Python 的版本好及安装时间，则证明 Python 环境搭建成功。否则需要进一步配置环境变量。如图 3.2 所示：

```
C:\Users\song>python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

图 3.2 本机 Python 环境配置

### 3.3.2 Python 环境变量配置

方法一：使用 cmd 命令添加 path 环境变量

在控制台下输入：path=%path%;D:\Python36，并输入回车键即可查看 Python 环境变量。 其中: D:\Python36 是 Python 的安装目录。

方法二：在环境变量中添加 Python 目录

1) 右键点击"计算机"，然后点击"属性"

2) 然后点击"高级系统设置"

3) 点击“系统变量”，找到 Path

4) 然后在"Path"行，添加 python 安装路径即可

## 3.4 TensorFlow-GPU 安装

计算机上通常有多个计算设备，CPU 和 GPU。而 TensorFlow 则完美的支持 CPU 和 GPU 这

两种设备。可以用以下字符串表示来指定这些设备，例如:

- "/cpu:0": 本机中的 CPU
- "/gpu:0": 本机中的 GPU, 如果有英伟达的 GPU 的话.
- "/gpu:1": 本机中的第二个 GPU，以此类推。

如果 Tensorflow 代码中既有 CPU 的实现方法，又有 GPU 的实现方法，当这个运算被指派设备时,GPU 有优先权，因为 GPU 的运行速度可以达到 CPU 的 30 倍以上，大大提升计算能力，减小手写数字识别系统的响应时间。如果想使用 TensorFlow-GPU 版本，还需要安装 CUDA 和 CuDNN。

### 3.4.1 下载 CUDA 软件包

首先来到 CUDA 官方网站 https://developer.nvidia.com/cuda-downloads，单击 Windows 按钮后，如下图所示：

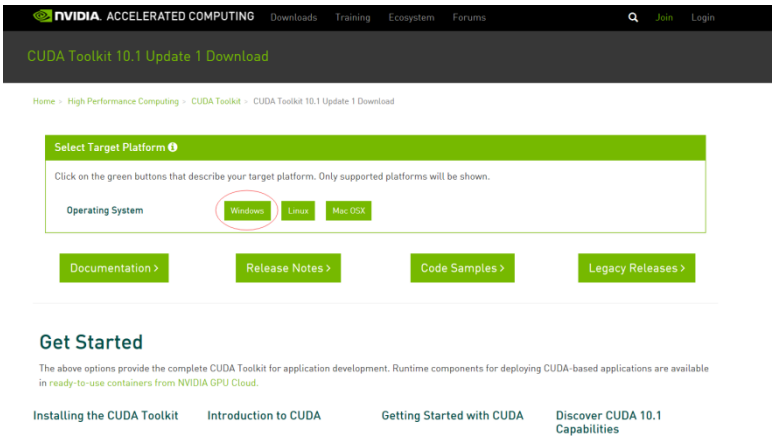



图 3.3 CUDA 安装包下载

注意：CUDA 软件包也有很多个版本，必须与 TensorFlow 的版本对应才行。比如 TensorFlow1.0 以后，直到 TensorFlow 1.5 的版本只支持 CUDA 8.0。可以根据链接 https://developer.nvidia. com/cuda-toolkit-archive 找到更多版本。

### 3.4.2 安装 CuDNN 库

输入网址 https://developer.nvidia.com/cudnn 来到下载页面，注册后下载 CuDNN 安装包。CuDNN 的版本选择也是有规定的。以 Windows 10 操作系统为例，TensorFlow 1.0 到 TensorFlow 1.2 版本使用的是 CuDNN 的 5.1 版本，从 TensorFlow 1.3 版本之后使用的是 cuDNN 的 6.0 版本（cudnn-8.0-windows10- x64v6.0.zip）得到相关包后解压，直接复制到 CUDA 安装路径对应的文件夹下面就行。

并不是所有的显卡都可以安装 TensorFlow-GPU，可使用 nvidia-smi 命令查看显卡信息。在安

装完成 NVIDIA 显卡驱动之后，对于 Windows 用户而言需要注意的是，只有将相关的环境变量添加进去，才能在控制台下识别 nvidia-smi 命令。

## 3.5 Anaconda

### 3.5.1 Anaconda 介绍

Anaconda 是一种 Python 语言的免费增值开源发行版，用于进行大规模数据处理, 预测分析, 和科学计算, 致力于简化包的管理和部署。Anaconda 使用软件包管理系统 Conda 进行包管理。可在 https://www.anaconda.com/download/#macos 网址中下载 Anaconda。

### 3.5.2 Conda 介绍

Conda 是开源包（packages）和虚拟环境（environment）的管理系统。可用 conda 来安装更新卸载工具包。也可在 conda 中建立多个虚拟环境，隔离开发不同项目时，所需要的不同版本的工具包，防止不同安装版本的冲突。例如 Python2.x 和 Python3.x。可以用 conda 建立两个 Python 虚拟环境，在不同的环境中运行不同版本的 Python 代码。

Anaconda 通过管理工具包、开发环境、Python 版本，大大简化了工作流程。不仅可以方便地安装、更新、卸载工具包，而且安装时能自动安装相应的依赖包，同时还能使用不同的虚拟环境隔离不同要求的项目。

Anaconda 安装后，可以从菜单中看到它包含几个应用程序，其中 Anaconda Navigator 是这几个程序的导航入口。Anaconda Navigator 是 Anaconda 发行包中包含的桌面图形界面，可以用来方便地启动应用、方便的管理 conda 包、虚拟环境。Navigator 可以从 Anaconda 云端或本地 Anaconda 仓库中搜索包。提供了 Windwos、maxOS 和 Linux 版本。Anaconda Navigator 主界面如下：

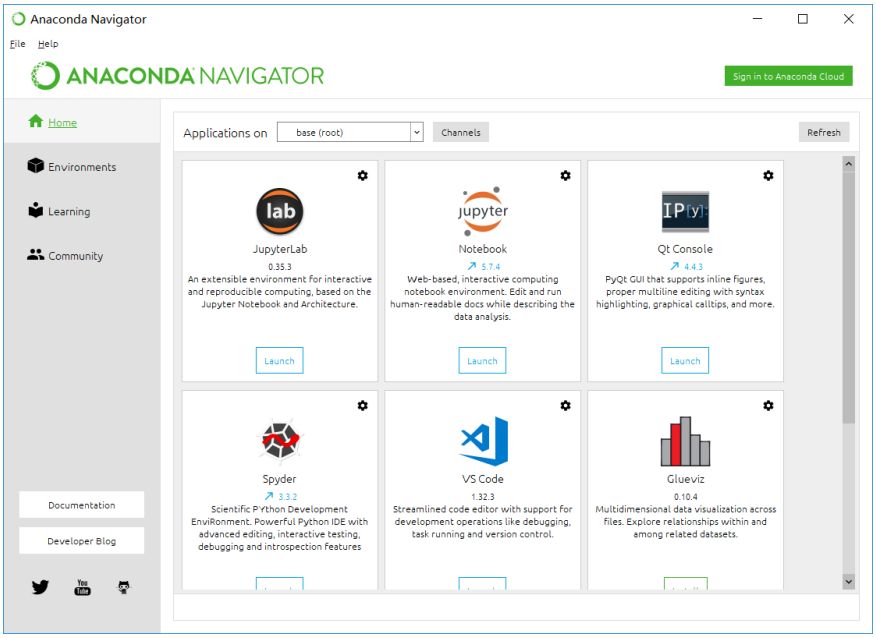



图 3.4 Anaconda Navigator 主界面

在左边菜单栏中可以看到四个选项，一般常用的是 Home 和 Environments。Environments 是你搭建开发环境的地方，你可以在 Environments 中创建一个开发环境，然后下载所需要的包即可。例如：

1) 创建开发环境

点击左下角 create，弹出创建开发环境框，输入所需创建的环境名并选择 python 类型，点击确认即可创建。

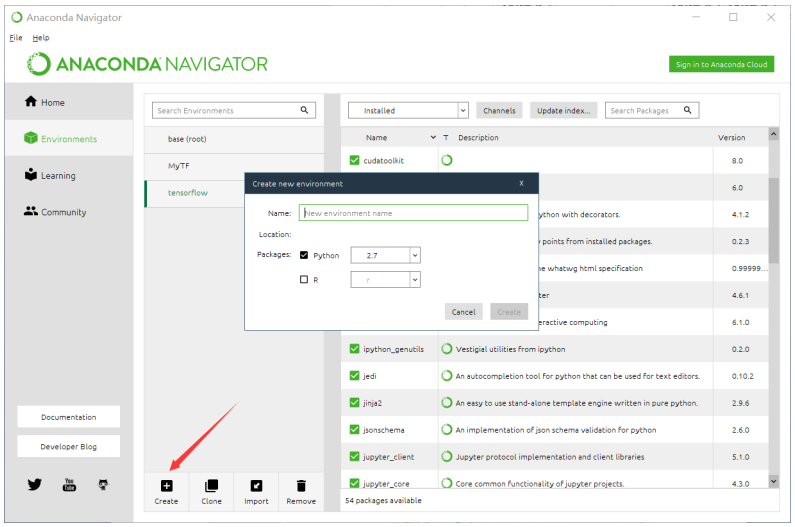



图 3.5 创建开发环境

2) 下载 tensorflow 包

搜索 tensorflow 包，勾选要下载的包，然后点击右下角 Apply 即可。

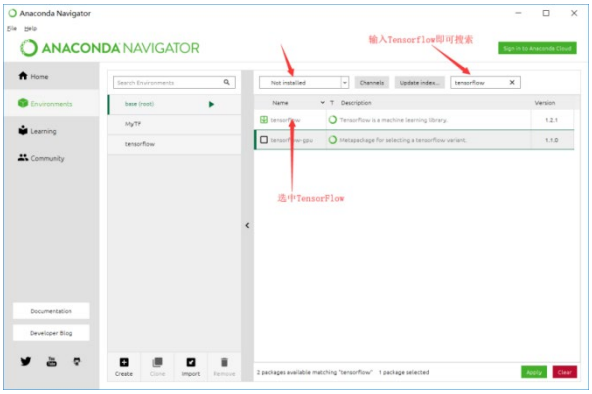



图 3.6 下载 TensorFlow 安装包

Jupyter notebook 常用来编写 TensorFlow 程序。因为 Jupyter notebook 是一种可以在网页上运行的记事本。在写程序时，无需切换到其他开发文档。每写完一段代码，回车即可执行，并保留每一段代码的运行日志，方便查看当前的代码执行状态。而且，调试也极其方便，可以大大的提高开发效率。

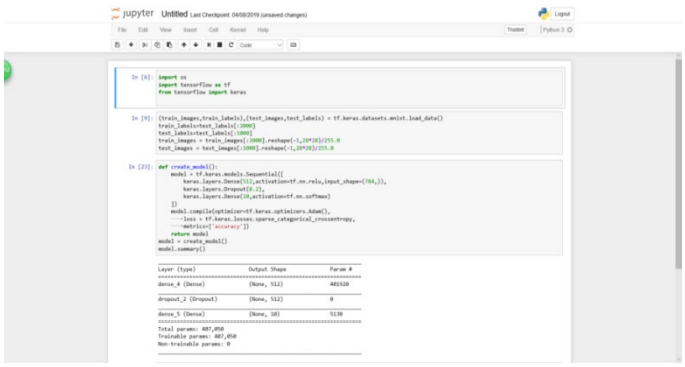



图 3.7 Jupyter 开发

## 3.6 本章小结

本章第一节介绍了本机的硬件设备信息。第二节介绍了本次系统所选的集成开发环境 Pycharm 及其优点。第三节介绍了 Python3.x 的安装及环境配置。第四节介绍了 TensorFlow-GPU 的安装步骤。最后一节介绍了用于安装 Tensorflow 的软件 Anaconda 以及用于开发编写 Tensorflow 代码的插件 Jupyter notebook。

装 订 线

# 第四章 系统的设计与实现

本章将详细的讲述本文所设计的基于 TensorFlow 框架的手写数字识别系统中所设计的关键技术进行阐述。主要包括 SoftMax Regression 模型的设计与实现、CNN 模型的设计与实现、WEB 网页设计、Flask 框架的引用等等。

## 4.1. Softmax Regression

### 4.1.1 回归模型介绍

回归模型是一种预测性的建模技术，它研究的是因变量（目标）和自变量（预测器）之间的关系。这种回归模型通常用于预测分析，时间序列模型以及发现变量之间的因果关系。例如，全国公民的文化程度与全民月读书量之间的关系就很适用于回归模型解决。

回归模型重要的基础或者方法就是回归分析，回归分析是研究一个变量（被解释变量）关于另一些变量（解释型变量）的具体依赖关系的计算方法和理论，是建立模型和数据分析的重要工具。在这里，我们使用曲线或直线来拟合这些数据点。在这种方式下，从曲线或线到数据点的距离差异最小。下面是回归分析的几种常用方法

1) Linear Regression 线性回归
2) Logistic Regression 逻辑回归
3) Polynomial Regression 多项式回归
4) Stepwise Regression 逐步回归
5) SoftMax Regression SoftMax 回归

由于 Logistics Regression 算法复杂度低，较容易实现等特点，因此逻辑回归在工业中得到广泛的使用，但是逻辑回归算法主要用于处理二分类的问题，对于多分类的问题，则是心有余而力不足，需要使用适用于多分类问题的算法。

Softmax Regression 算法是逻辑回归算法在多分类问题上的应用与推广，主要用于处理多分类问题。其中，要求任意两个类之间是线性可划分的。多分类问题，它的类标签 y 的取值个数应大于 2，如手写字识别，即识别{0,1,2,3,4,5,6,7,8,9}是哪一个数字。

MNIST 数据集的每一张图片都表示一个（从 0 到 9） 数字。优良的模型在看到一张图后就能知道它属于各个数字的对应概率。比如，当训练好的模型看到一张数字"9" 的图片，就判断出它是数字"9"的概率为 80%，而有 10%的概率属于数字"8"（因为 8 和 9 比较相似，只是左下方有些区别），同时给予其他数字对应的小概率，因为该图像代表其他的可能性微乎其微。

### 4.1.2 Softmax Regression 算法介绍

Softmax Regression 算法原理[13]简单介绍如下：

对于输入的手写体数字图像对于不同数字的“证据”加权求和，并将加权求和的结果转为对应数字的概率。如果手写体数字图像中像素很像某个数字，则对该数字求和的权值为正数，越像

这个数字，则权值越大。如不像这个数字，则权值为负数，越不像这个数字，则权值的绝对值越大。下图显示了 Softmax Regression 模型学习到的手写体数字图像对于 0~9 共 10 个数字类的权值。蓝色权值为正数，红色权值为负数，颜色越深，权值绝对值越大，如图 4.1 所示：

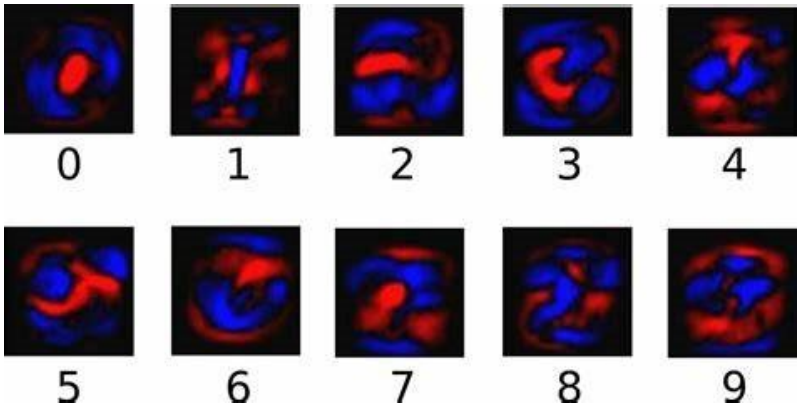



图 4.1 数字类的权值

此外，还需要引入其他“证据”，也就是常说的偏置量。因此对于给定的输入图片 x 代表某数字 i 的总体证据可以表示如公式 3.1 所示:

$$\text{evidence}_i \sum_j W_{i,j} x_j + b_i \tag{3.1}$$

在上述公式中， b(i)代表第 i 类数据的偏置量，W(i)表示的是训练时的权重。j 表示的是对于给定的图片 x 的像素索引，常用于像素求和。求和后调用 Softmax 函数可以把这些证据转换成概率 y，如公式 3.2 所示:

$$y = \text{softmax}(\text{evidence}) \tag{3.2}$$

Softmax 函数可以看成是一个激励（activation）函数，激励函数会将定义好的线性函数的输出，转换理想的格式，也就是关于 0~9 共十个数字类的概率分布。因此，只要给定一张图片，这张图片对于每一个数字的契合程度可以被 Softmax 函数转换成为一个概率值。Softmax 函数的公式定义如公式 3.3 所示：

$$\text{softmax}(x) = \text{normalize}(\exp(x)) \tag{3.3}$$

展开等式右边的子式，可以得到公式 3.4：

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{3.4}$$

Softmax 函数模型常定义为 Normalize()，这样看起来更简洁。Softmax 函数把输入值当成幂指数求值，之后对这些结果值进行正则化处理。Normalize()表示，更大的证据对应更大的假设模型里面的乘数权重值.反之，拥有更少的证据意味着在假设模型里面拥有更小的乘数系数。假设模型里的权值不可以是 0 值或者负值。Softmax 函数则会正则化这些权重值，使它们的总和等于 1，以此来构造一个有效的概率分布。对于 Softmax 回归模型可以用下面的图解释，对于输入的 xs 加权求和，再分别加上一个偏置量，最后再输入到 softmax 函数中，如图 4.2 所示：

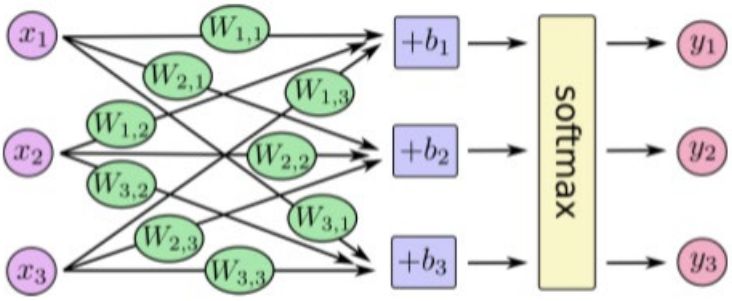



图 4.2 Softmax 函数

将上述方程用矩阵表示，则有以下矩阵，如公式 3.4 所示：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \begin{pmatrix} W_{1,1}x_1 + W_{1,2}x_1 + W_{1,3}x_1 + b_1 \\ W_{2,1}x_2 + W_{2,2}x_2 + W_{2,3}x_2 + b_2 \\ W_{3,1}x_3 + W_{3,2}x_3 + W_{3,3}x_3 + b_3 \end{pmatrix} \tag{3.4}$$

若该过程用向量（Vector）表示，有助于提高计算效率，如公式 3.5 所示：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left( \begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right) \tag{3.5}$$

将上式简化后，即可得到 Softmax 方程，Softmax 方程如公式 3.6 所示：

$$y = \text{softmax}(W_x + b) \tag{3.6}$$

### 4.1.3 Softmax Regression 模型实现

为实现高效快速的数值计算，通常会调用外部函数依赖库（例如 Numpy）， 把类似矩阵乘法这样的复杂运算使用其他外部语言实现。但是，在 Python 和外部计算之间来回切换，会消耗过多的系统资源，尤其是进程之中的资源。若使用 GPU 来计算外部数据[14]，由于 GPU 不能得到连续的执行，会消耗更多的资源。即使是采用分布式计算，也会浪费很多时间去传输外部数据。

TensorFlow 也把复杂的计算放在 python 之外完成，但是为了避免上文所述的那些开销，Tensorflow 做了进一步完善。它不单独地运行单一的复杂计算，而是先用图描述一系列可交互的计算操作，最后全部一起在 Python 之外运行。

使用 TensorFlow 之前，首先导入它：

Import tensorflow as tf

通过操作符号变量来描述这些可交互的操作单元，例如：创建一个 Float 型的占位符如下所示：

```
X = tf.placeholder(“float”,[None,784])
```

变量 X 不是一个特定的值，而是一个占位符，这个占位符 X 会在 Tensorflow 进行数值计算时作为一个 Float 型变量输入进去。而本系统则需要输入一定数量的手写体数字图像，而这种图像需要固定大小，长和宽各位 28 个像素点，因此可以展开为 28*28=784 维的向量。可以采用二位的 Float 型张量来表示手写体数字图像，张量的形状为[None，784]。None 表示张量（Tensor）可以是任意长度的。

Softmax 模型需要偏置值（biases）和权重值（weights），其中一种解决办法是使用占位符来

代替这两个变量，但是 Tensorflow 提供了更为便利的方法，它使用 Variable 函数来提供变量的引用。Variable 变量在描述交互性操作的图中，常被用于计算输入值，也就是说，当需要输入数据时，常用 Variable 来表示。Variable 在计算中可以被修改，因此 Variable 也常用于表示模型的参数值，权重值和偏置值如下所示：

```
W = tf.Variable(tf.zeros[784,10])
B = tf.Variable(tf.zeros[10])
```

通过给 tf.variable 赋予不同的初始值，来创建不同的 Tensor，在 Softmax 回归模型中，需要先对权重值和偏置值赋予全 0 的初始张量。但需要注意的是：权重是表示手写体数字图像，可能的结果为 0~9 共是个数字，因此它的维度必须是[784,10]。由矩阵运算常识可知：若想每一位对应不同的数字类，需要使用 784 维的图片向量乘以 10 维的偏置值向量，才可以得到 10 维的偏置向量。因此，偏置值向量 b 需要初始化维 10 维的向量。有了这些，就可用在几何上实现 Softmax 回归模型了，如下所示：

```
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

tf.matmul 是 tensorflow 的矩阵乘积函数，上述代码表示用输入 X 乘以权重值 W，然后加上偏置值 b，最后用 tf.nn.softmax 函数处理计算的结果。其中 X 是一个拥有多个输入的二维张量，输入的多少取决于训练的手写体数字图像的数目。Tensorflow 框架使 softmax 模型的计算变得十分简单灵活，很方便地描述各种各样的数值计算，正因如此，本系统才选择了 Tensorflow 框架。不论是什么领域方向的模型，只要定义好 tensorflow 模型，就可用运行于各个设备，跨平台移植性极好。

### 4.1.4 Softmax Regression 模型训练

在训练 softmax 回归模型之前，需要先定义一个训练指标，用于评测模型的好坏。在机器学习领域中，通常使用“成本”或者“损失”来评估这个模型的好坏，然后尽量使成本变低，或损失降低。因为训练的损失（loss）越低，证明这个模型越优。常用的成本代价函数是“交叉熵”函数，交叉熵来源于信息论中的参数估计和极大似然估计。英文表示为 cross-entropy。交叉熵的定义如下所示：

$$H_{y'}(y) = -\sum_i y'_i \log(y_i) \tag{3.7}$$

交叉熵可以理解为衡量 softmax 模型描述事实的低效性，交叉熵的值越低，模型预测效果越好，损失函数或者成本函数越低。$y'$ 是手写数字图像的实际分布， y 是 $y'$ 对调用 log 函数后的预测概率分布。为了使用交叉熵来评价函数模型，首先需要引入一个新的 placeholder 占位符来表示输入的值，之后就可用使用$-\sum y'$ log(y)函数去计算交叉熵（也就是损失率），代码如下所示：

```
y = tf.placeholder(“float”,[None,10])
cross_entropys = -tf.reduced_sums(Y_ * tf.logs(y))
```

首先需要用 tf.log 去计算每个手写体数字图像实际分布的对数。将计算结果和 Y_相乘，这样可以对每个元素都计算成本函数。最后，使用 reduce_sums 函数计算所有元素的张量总和。交叉

熵经常会与 softmax 回归一起使用，所以 Tensorflow 提供了 tf.nn.softmax_cross_entropy_with_logits() 函数,相当于对 softmax 函数和 cross_entropy 函数的封装，可以加快计算速度。

交叉熵不仅仅指单一的某一个手写体数字图片的预测和真实值，它指的是全部用于训练的图片的交叉熵总和。对 60000 个手写数字图片的预测性能，肯定要远远由于对某一图片的训练，这样可以更好地描述 softmax 模型的运算性能。

为了更好的降低成本，Tensorflow 框架会自动调用反向传播算法来确定模型中自定义的变量是如何影响 loss 值的。之后通过不断地修改计算单元图，来不断地优化变量，最终达到最小化成本的目的。

```
trains_steps =tf.trains.GradientsDescentOptimizers(0.01).minimize(cro_entros)
```

在上述代码中，Tensorflow 框架使用梯度下降算法来降低交叉熵。本项目中梯度下降率为 0.01。TensorFlow 提供现成的自适应优化器，包括 AdagradOptimizer 和 AdamOptimizer。Tensorflow 每次只需向交叉熵减小的方向移动一点点，微调变量，积少成多，最后训练好机器学习的模型。

模型已经搭建完毕，在运行前需要进行变量的初始化操作，初始化占位符。在进行初始化时，需要为 Tensorflow 的图创建会话（session）操作，会话会分配内存保存当前变量的值。创建完会话后，就可以在会话中训练设计好的模型，下面代码会训练模型 10000 次，每 100 次为一批，一共 100 批次。

```
Init =    tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
for _ in range(10000):
      batch_xs, batch_ys = data.train.next_batch(100)
      sess.run(trains_steps, feed_dict={xs: batch_xs, ys_: batch_ys})
```

上述代码中，tensorflow 会随机选取 100 个批处理数据进行训练。之后，输入一个 784 维的手写体数字图像作为实参，替代已完成初始化的浮点型占位符变量，这就是一次 Train_step 过程。随机训练指的是选取一小部分随机数据进行训练。最理想的情况是对所有的手写体数字图像进行训练，因为训练所有的输入数据会生成最好的训练模型，然而这无疑会需要很大很大的资源消耗，实际项目之中，不能训练全部数据。所以，每次都选取随机的数据，既可以训练出较为优异的模型，又能大大减少训练时间。

### 4.1.5 Softmax Regression 模型评估

在评估回归模型中，经常需要用到 argmax()函数。argmax()函数返回的是最大数的索引。简单地讲，就是返回函数值 y 最大时所对应的 x 值，常用此函数来找出预测结果中哪个数字的概率最大。例如 tf.argmax(ys,1)就表示预测结果中，哪个数字最有可能匹配训练的那个手写体数字图像。

```
corrects_predictions= tf.equals(tf.argmax(ys, 1), tf.argmax(ys_, 1))
accuracys = tf.reduced_means(tf.casts(corrects_predictions, tf.float32))
print(sess.run(accuracys, feed_dicts={xs: data.test.images, ys_: data.test.labels}))
```

上述代码会返回一组布尔值，为了更精确的表示各个预测想的比例，可以将 Bool 值转换为 Float 值。例如：[Flase, True, Flase,False]就会变成[0,1,0,0]，加权求平均值后值为 0.25。最后经过上述代码的训练之后，softmax 模型识别的准确率为 91.92%，已经是一个比较不错的训练模型了。

### 4.1.6 Softmax Regression 代码

算法 1：Softmax 回归算法

```
INPUT：Plenty of Images to Train
OUTPUT: Softmax Regression Model
procedure Softmax Regression
import os
import mnists.model as model
import tensorflow as tf
from tensorflow.examples.tutorials.mnists import input_data
datas = input_data.read_data_sets("/mnists/MNIST_data", one_hot=True)
with tf.variables_scope("Regressions"):
    xs = tf.placeholders(tf.float32, [None, 784])
    # softmax 函数
    ys, variable = models.regression(xs)
# train
ys_ = tf.placeholders("float", [None, 10]) #交叉熵
cross_entropys = -tf.reduced_sum(ys_ * tf.log(ys)) #梯度下降
train_step = tf.train.GradientDescentsOptimizer(0.01).minimize(cross_entropys)
correct_predictions = tf.equal(tf.argmax(ys, 1), tf.argmax(ys_, 1))
accuracys = tf.reduced_mean(tf.cast(correct_predictions, tf.float32))
savers = tf.train.Savers(variable)
with tf.Session() as session:
    sessions.run(tf.global_variables_initializer())
    for i_ in range(1000):
        #每次取出 mnist 数据集中的 100 个数据
        batch_xs, batch_ys = datas.train.next_batchs(100)
        sessions.run(train_steps, feed_dict={xs: batch_xs, ys_: batch_ys})
    print(sessions.run(accuracys,feed_dict={xs:data.test.images, ys_: data.test.labels}))
    #保存模型
    paths = savers.save(
session,os.paths.join(os.paths.dirname(__file__),'data','regressions.ckpt'),
        write_meta_graphs=False, write_states=False)
    print("Saved Path:", paths)          #打印保存模型的路径
end procedure
```

装 订 线

## 4.2 Convolutional Neural Networks 模型

### 4.2.1 CNN 模型介绍

深度学习其目的在于建立一个神经网络，用于模拟人脑进行分析学习。机器学习的许多研究成果，都和对大脑认知原理的研究有着密切的联系，尤其是对视听原理方向的研究。日本一个神经生物学家发现了人类视觉器官的分级信息传输方式，也就是说可视皮层是分级的。人类的视觉视觉从原始的自然信号输入开始，像素点透过视网膜，映射到视觉神经，最后大脑皮层的某些感光细胞会判断出所看到的事物的颜色和形状等，最后抽象出眼前的物体是为何物[15]。

因此根据大脑分层逐级认知的这个特点，法国科学家 Yann LeCun 提出卷积神经网络（Convolutional Neural Networ, 以下简称为 CNN）并应用在手写字体识别上。CNN 是一种多层复合的一种神经网络[16]，此神经网络使用滤波器提取事物的特征值，再使用卷积核作为特征抽取器，并自动训练特征抽取器。也就是说，神经网络层数、阈值大小、卷积核等参数都需要由卷积神经网络去学习，通过大量的训练数据来不断优化提升。此外，有一个便捷之处在于图像可以直接作为卷积神经网络的输入，不再需要传统图像识别那些复杂的特征提取和数据重建过程。

CNN 的基本架构包含三个主要的层——卷积层、池化层和全连接层[17]。卷积层用来学习输入数据的特征，由很多的卷积核组成。池化层用来降低卷积层输出的特征向量，常用的池化方法有最大池化核平均池化，本文选用的是最大池化。而全连接层将前两层叠加起来后，就可以形成 1 层或者 n 层全连接层，拥有更优的推理能力。卷积神经网络很适用于图像处理方向的机器学习问题。CNN 可以通过一步一步的训练、学习、优化，将数据量庞大的图像识别问题不断降低难度，最终训练这庞大的数据量。

在手写数字识别的应用中，卷积神经网络应用极为广泛，且性能也较为优良。常用的卷积神经网络模型就是 LeNet5 模型，LeNet5 特征能够总结为如下几点：

1）卷积神经网络使用三个层作为一个集合：卷积层，池化层，全连接层

2）使用卷积提取空间特征

3）使用映射到空间均值下采样（subsample）

4）双曲线（tang）或 S 型（sigmoid）形式的非线性

5）使用多层神经网络作为最后一个分类容器

6）为减少运算成本，采用层与层之间的稀疏连接矩阵

本文选用的卷积神经网络就由卷积层（Convolutional layer）、池化层（Pooling layer）和全连接层（Fully connected layer）组成。首先由卷积层和池化层相互配合，组成若干个卷积核，并逐层提取特征值，最终通过若干个全连接层完成手写体数字的分类。

总而言之，卷积神经网络通过卷积操作来模拟特征值的区分，最后通过卷积核的权值共用及池化，来降低卷积神经网络参数的数量级，最后通过传统的神经网络模型完成模式匹配等任务。下面就来介绍一下 CNN 的结构以及 CNN 模型如何进行训练。

### 4.2.2 CNN 结构

CNN 不仅可以自动提取功能，还可以完成分类。它是图像识别领域的一种有效识别方法。本文中使用的 CNN 结构，共有 7 层。输入层是尺寸为 28 * 28 的标准化矩阵。C1 层是包含 N1 个特征映射的特征提取层，这些映射的每个神经元都定义了它们的感知域。这里使用 5 * 5 卷积内核来卷积输入图像，然后获得 28 * 28 的大小特征映射。每个 C 层将遵循 S 层，这构成了两个特征提取。S2 层是子采样层，在该层中，特征映射相同 N1 的数量，但是它们的大小减小到 14 * 14。C3 层和 S4 层与 C1 层和 S2 层相同，以获得 N2 特征图。在 C5 层，我们使用 4 * 卷积内核在 S4 层上进行卷积，然后获得 N3 个特征映射。S5 和 C4 的连接是将 C5 层的每个特征值映射连接到 S4 层的 N2 特征映射的完全连接。我们可以通过最后一个完整的连接层获得输出，然后完成分类，LeNet 神经网络模型如图 4.3 所示：

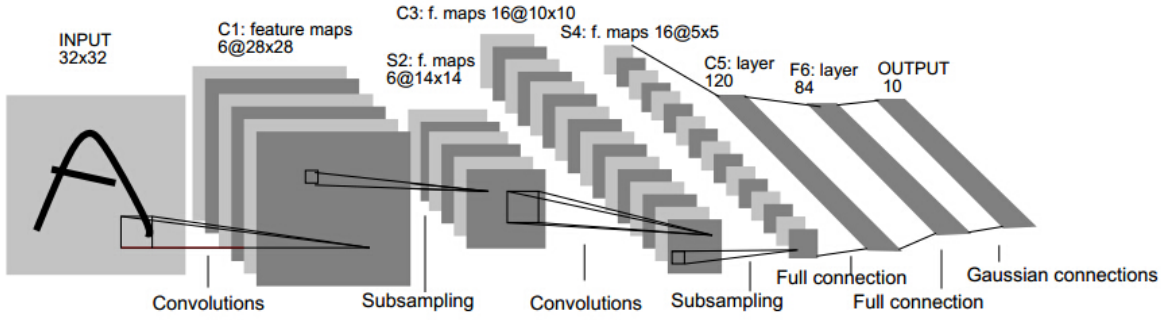



图 4.3 LeNet 神经网络

下面就来详细的介绍本文使用的卷积神经网络的模型参数。X_image 代表模型需要处理的输入数据，输入数据的形状为[-1,28,28,1]。-1 表示暂时不考虑输入数据的维度数，后两个 28 表示输入数据的长宽均为 28 像素，最后的 1 表示的是 Channel 的数量，因为输入的图片是黑白图像，所以通道数为 1，若处理的数据为 RGB 图像，则该参数值应为 3。W_conv1 是卷积层 1 的权重值，该权重是一个[5, 5, 1, 32]的张量，前两个维度是 patch 的大小。卷积核 patch 的大小是 5*5，卷积在每个 5×5 的 patch 中算出 32 个过滤器，因此经过第一层的卷积运算后，输入图像的高度会由 1 层变为 32 层，高度变高。b_conv1 表示卷积层 1 的偏置量，其形状应与权重的形状相同，否则不能进行矩阵加法运算。

```
x_image = tf.reshape(x, [-1, 28, 28, 1])
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])
```

对输入图像进行卷积运算之后，会得到输入图像的有效特征，但随之而来的负面作用就是特征值过多，因此需要对其进行池化运算，池化运算也叫做下采样运算（subsample）。池化分为平均池化核最大池化。本文采用的池化运算为最大池化，也就是在一个卷积核 patch 大小的范围里，选出最大值。对输入的图像 x_image 和权值向量进行卷积相乘，之后加上偏置值，使用 ReLU 激励函数激活特征值，最后进行最大池化，池化步长为 2，长宽均由原来的 28 个像素减少为 14 个像素。长宽各为原来的 1/2，总特征值减为原来的 1/4，大大减少了后面运算的复杂度。

ReLU（Rectified Linear unit）激活函数最近变成了神经网络中隐藏层的默认激活函数。这个

简单的函数包含了返回 max(0,x)，所以对于负值，它会返回 0，其它返回 x。ReLU 函数如图 4.4 所示：

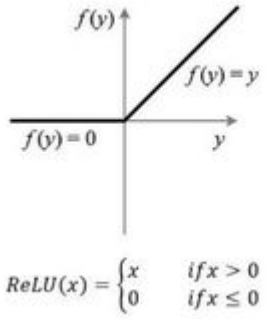



图 4.4 ReLU 函数

```
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)
```

进行一次卷积运算和池化运算后，得出的效果仍不太理想。为了进一步的获取输入图像的有效特征值，决定构建更深一层的卷积神经网络。因此，再进行一次卷积和池化运算。W_conv2 表示池化层 2 的权重。前两维表示池化 patch 的大小维 5*5。32 表示输入的卷积核的数量，64 表示经过第二次卷积运算后，得到的卷积核数量。b_conv2 表示偏置值，其形状应与第二层卷积层形状相同，否则不能进行卷积运算。之后再一次对输入数据 h_pool1 和权重向量进行卷积相乘，并加上偏重值，使用 ReLU 激励函数激活特征值，最后进行最大池化，池化步长为 2，长和宽均由原来的 14 个像素减少为 7 个像素。长宽各为原来的 1/2，总特征值减为原来的 25%。

```
W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])
h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
```

经过两次的卷积和池化运算，特征值提取到一个较好的程度，不需再次进行卷积和池化运算。现在，图片的长宽降到 7*7，共有 64 层过滤器。本全连接层使用包含 1024 个神经元来处理整个图片。W_fc1 表示该全连接层的权重，tensor 的第一个参数表示上一层卷积层的输出，大小为 7*7，并带有 64 个过滤器。第二个参数表示本层中的神经元数量，表示特征经验，其值一般为 2 的 n 次方，可自由设定，本文选择的是 2 的 10 次方，1024 个神经元。接下来，将张量 reshape 成一维形状，重塑形状后，将其乘以权重矩阵 W_fc1，再加上偏置 b_fc1，最后使用 ReLU 激励函数激活。

```
W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])
h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
```

经过全连接之后，模型可以对数据提取出有效的特征值，但是此模型只对这个训练数据有效，不能推广到所有的手写数字。因此需要对部分难以满足该模型的数据果断抛弃，减少过拟合。本文是要用 dropout 函数，来减少有效参数的数量。这包含了移除结点及它们相关的输入、输出连接。决定哪一个神经元丢弃，哪一个神经元保留是随机的。为了选择神经元比较一致，对神经元是否被丢弃赋予一个概率，并以次概率调用 dropout 函数。因为 TensorFlow 的 tf.nn.dropout 操作会自动处理神经元输出值的 scale。所以用 dropout 的时候可以不用考虑 scale。

```
keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
```

最后，添加一个 softmax 层，就像上一节提到的单层 softmax 回归模型一样，将 1024 个神经元运算后的数据转化成 10 个数字的输出概率，这样本文使用的卷积神经网络模型就搭建好了。下面的工作就是给该模型输入数据，进行训练。

```
W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])
y = tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
return y, [W_conv1, b_conv1, W_conv2, b_conv2, W_fc1, b_fc1, W_fc2, b_fc2]
```

### 4.2.3 CNN 的训练过程

1.全连接层

(1)计算输出

我们使用 l 来表示当前层，因此 l 层的输出可以表示为公式 3.8：

$$x^l = f(u^l), u^l = W^l x^{l-1} + b^l \tag{3.8}$$

其中 f(.)表示激活函数，在本系统中，使用 sigmoid 函数将输出值压缩为[0,1]。 其中第 n 号样本的错误可表示为公式 3.9：

$$E^n = \frac{1}{2}\sum_{k=1}^{c}(t_k^n - y_k^n)^2 = \frac{1}{2}\|t^n - y^n\|_2^2 \tag{3.9}$$

(2) 反向传播

反向传播，只有在训练模型时才会用到。神经网络在训练的过程中数据的流向有两个方向，即先通过正向生成一个值，然后观察其与真实值的差距，再通过反向过程将里面的参数进行调整，接着再次正向生成预测值并与真实值进行比对，这样循环下去，直到将参数调整为合适值为止。每个神经元偏差的灵敏度，可以被看作反向传播的误差，定义如公式 3.10：

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial u}\frac{\partial u}{\partial b} = \delta \tag{3.10}$$

L 层在反向传播过程中的灵敏度可以表示如公式 3.11：

$$\delta^{L} = (w^{l+1})^{T}\delta^{l+1} \circ f(u^{l}) \tag{3.11}$$

而输出层的神经元敏感性和上述不同，如公式 3.12 所示：

$$\delta^{L} = f(u^{L}) \circ (y^{n} - t^{n}) \tag{3.12}$$

(3)更新权重

最后，对于每个给定的卷积神经元，可以使用 δ 规则来更新权重。对于 L 层，每个权重的偏导数是输入和灵敏度之间的乘积，如公式 3.13。然后通过将负学习率乘以 l 层中的更新神经元权重来得到偏导数(偏差更新表达式类似于公式 3.14：

$$\frac{\partial E}{\partial W^{l}} = x^{l-1}(\delta^{l})^{T} \tag{3.13}$$

$$\Delta W^{l} = -\eta\frac{\partial E}{\partial W^{l}} \tag{3.14}$$

2.卷积层

据以上所述，为了更新 l 层的每个权重的权重，首先必须获得 l + 1 层的灵敏度和 δl+ 1。然后 δl+ 1 乘以权重，并激活函数在 l 层中输入神经元节点的导数。因此我们得到了 l 层的灵敏度 δl。最后，更新 l 层上每个神经元的重量。但卷积层的难度是计算梯度。下一层卷积层是子采样层，因此为了计算 δl，我们需要对子采样层的灵敏度图进行上采样。在子采样层中，对于一个地图设置相同的权重 β，因此梯度公式如 3.15：

$$\delta_{j}^{l} = \beta_{j}^{l+1}(f(u_{j}^{l}) \circ up(\delta_{j}^{l+1})) \tag{3.15}$$

在这里，计算每个 map 的灵敏度，然后计算 biass 梯度，如公式 3.16 所示：

$$\frac{\partial E}{\partial b_{j}} = \sum_{u,v}(\delta_{j}^{l})_{uv} \tag{3.16}$$

最后，使用 BP 算法计算权重的梯度。

3.子采样层

在子采样层中，输入映射和输出映射的数量相同，但输出映射的大小减小，如公式 3.17 所示：

$$x_{j}^{l}=f'(\beta_{j}^{l}down(x_{j}^{l-1}) + b_{j}^{l}) \tag{3.17}$$

其中 down(.)表示下采样函数。每个输出图对应于权重 β 和偏差 b。

对于子采样层，计算灵敏度是最难的部分。当我们获得灵敏度时，它很容易更新 β 和 b。如果下一层子采样是全连接网络，则可以通过 BP 算法来计算子采样层的灵敏度。首先，计算卷积核的梯度，如公式 3.18 所示：

$$\delta_{j}^{l}=f(u_{j}^{l}) \circ conv2(\delta_{j}^{l+1}, rot180(K_{j}^{l+1}), full) \tag{3.18}$$

然后计算 b 和 β 的梯度。偏差 b 的计算以及方程(9)，但对于权重 β，如在前向传播过程中存在下采样，所以它的公式如 3.19 所示：

$$\frac{\partial E}{\partial b_{j}} = \sum_{u,v}(\delta_{j}^{l} \circ d_{j}^{l})_{uv} \tag{3.19}$$

最后，使用公式 3.13 和公式 3.14 去更新参数。

### 4.2.4 CNN 代码

**算法 2：卷积神经网络代码**

**INPUT：Plenty of Images to Train**
**OUTPUT: Convolutional Neural Networks Model**
**procedure CNN**

```
#import my_modules
import tensorflow as tf
from tensorflow.examples.tutorials.mnist imports input_data
#import sqd_data
mnists = input_data.read_data_sets("MNIST_data/", one_hot=True)
sessions = tf.InteractiveSession()
# Create the model
sx = tf.placeholders(tf.float32, [None, 784])
sy_ = tf.placeholders(tf.float32, [None, 10])
sW = tf.Variable(tf.zeros([784, 10]))
sb = tf.Variable(tf.zeros([10]))
sy = tf.nn.softmax(tf.matmul(x, W) + b)
def weight_variables(shapes):
   sinitial = tf.truncated_normal(shapes, stddev=0.1)
   return tf.Variable(initial)
def bias_variables(shapes):
   sinitial = tf.constant(0.1, shape=shape)
   return tf.Variable(initial)
def convolution2d(x, W):
   return tf.n.covolution2d(xs, Ws, strides=[1, 1, 1, 1], padding='SAME')
def max_pools_2x2(x):
   return tf.nn.max_pools(xs, ksizes=[1, 2, 2, 1],
                                strides=[1, 2, 2, 1], padding='SAME')
W_convs1 = weight_variables([5, 5, 1, 32])
b_convs1 = bias_variable([32])
x_images = tf.reshaped(x, [-1,28,28,1])
h_convs1 = tf.nn.relu(convol2d(x_images, W_convs1) + b_convs1)
h_pools1 = max_pools_2x2(h_convs1)
W_convs2 = weight_variables([5, 5, 32, 64])
b_convs2 = bias_variables([64])
h_convs2 = tf.nn.relu(conv2d(h_pools1, W_convs2) + b_convs2)
h_pools2 = max_pools_2x2(h_convs2)
W_fcs1 = weight_variables([7 * 7 * 64, 1024])
b_fcs1 = bias_variables([1024])
```

```
h_pools2_flat = tf.reshape(h_pools2, [-1, 7*7*64])
h_fcs1 = tf.nn.relu(tf.matmul(h_pools2_flat, W_fc1) + b_fc1)
keep_probs = tf.placeholder(tf.float32)
h_fcs1_drop = tf.nn.dropout(h_fcs1, keep_probs)
W_fcs2 = weight_variable([1024, 10])
b_fcs2 = bias_variable([10])
y_convs=tf.nn.softmax(tf.matmul(h_fcs1_drop, W_fcs2) + b_fcs2)
# Define losses and optimizers
cross_entropys = -tf.reduce_sum(y_*tf.log(y_conv))
strain_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropys)
scorrect_prediction = tf.equals(tf.argmax(y_convs,1), tf.argmax(ys_,1))
saccuracy = tf.reduce_mean(tf.cast(scorrect_prediction, tf.float32))
savers = tf.train.Savers()
sessions.run(tf.initialize_all_variable())
for i_ in range(20000):
   batchs = mnists.train.next_batch(50)
   if i_%200 == 0:
      train_accuracys = accuracy.eval(feed_dict={
            xs:batch[0], ys_: batch[1], keep_probs: 1.0})
      print("steps %d, training accuracy %g"%(i, train_accuracy))
   train_steps.run(feed_dict={xs: batch[0], ys_: batch[1], keep_probs: 0.5})
save_paths = savers.save(session, "saved_model_2_20000/model2.ckpt")
print ("Models saved in file: ", save_paths)
print("tests accuracy %g"%accuracy.eval(feed_dict={
      xs: mnists.test.images, ys_: mnists.test.labels, keep_probs: 1.0}))
end procedure
```

装 订 线

## 4.3 Flask 框架

### 4.3.1 JSON 介绍

JSON 是一种基于 JavaScript 编程语言的数据交换格式。它可以在 Python 和其他语言中使用。主要用于在服务器和 Web 应用程序之间传输数据。

### 4.3.2 JSON in API

JSON 的主要应用之一是在 Web 应用程序中构建 API。这非常有用，因为它允许开发人员使用支持 JSON 的任何语言在我们的 API 之上构建。大多数现代编程语言都支持 JSON。Flask 提供了 jsonify 模块，使我们能够实现在 Python 中构建 Flask 应用程序时返回 JSON 这一目标。
序列化 JSON

计算机处理大量信息后需要进行数据转储。因此，json 库公开了 dump()方法，用于将数据写入文件。还有一个 dumps()方法，用于写入 Python 字符串。 根据相当直观的转换，简单的 Python 对象被转换为 JSON。

### 4.3.3 Request

在前端开发当中，客户端与服务器之间的数据交换极为重要。在 Flask 中由全局的 request 对象来提供这些信息。来自客户端网页的数据作为全局请求对象发送到服务器。为了处理请求数据，应该从 Flask 模块导入。请求对象的重要属性如下所示：

Form：从 post 中请求解析的格式

Args：解析查询字符串的内容，要操作 URL 中提交的参数可以使用该属性

Cookies：持有 Cookie 名称和值的字典对象，类型是 dict

File：与上传文件有关的数据，带有通过 POST 或 PUT 请求上传的文件

Method： 当前请求方法，比如 POST、GET，默认为 GET

将下面程序保存为 hello.py（或是类似的），然后用 Python 解释器来运行。 注意应用文件名不可以为 flask.py，因为 flask.py 与 Flask 框架名相同，容易造成歧义。之后访问 Http:///localhost:5000/或者 http://127.0.0.1:5000/，就会看到 Hello World!显示在屏幕上。

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

上述代码的实现原理如下：

1.首先，本代码导入了 Flask 类。

2.其次，创建一个该类的实例，第一个参数是包的名称或者应用模块的名称。如果使用单一的模块，则应该使用 __name__ ，因为模块的名称将会因其作为单独应用启动还是作为模块导入而有所不同（ 也即是 '__main__' 或实际的导入名）。

3.紧接着使用 Route 函数声明哪类 URL 可以触发，哪类 URL 则不能触发。这个函数的名字也在生成 URL 时被特定的函数采用，这个函数返回希望显示在用户浏览器中的信息。

4.最后调用 Run 函数将应用运行在 localost 服务器中

5.想要关闭服务器，按 Ctrl+C。

要想开发出易于维护的程序，关键在于编写形式简洁且结构良好的代码。到目前为止，你看到的示例都太简单，无法说明这一点，但 Flask 视图函数的两个完全独立的作用却被融合在了一起，这就产生了一个问题。视图函数的作用很明确，即生成请求的响应，对最简单的请求来说，这就足够了。但一般而言，请求会改变程序的状态，而这种变化也会在视图函数中产生。

例如，某用户在网站中注册了一个新账户。该用户在网页的表单中输入 Email 地址和用户密码，然后点击提交按钮。服务器接收到包含用户输入数据的请求，之后有视图函数会收到 Flask 的处理注册请求。这个视图函数需要访问数据库，添加新用户，然后生成响应回送浏览器。这两个过程分别称为业务逻辑和表现逻辑。把业务逻辑和表现逻辑混在一起会导致代码难以理解和维护。假设要为一个大型表格构建 HTML 代码，表格中的数据由数据库中读取的数据以及必要的字符串连接在一起。把表现逻辑移到模板中能够提升程序的可维护性。程序中的视图函数需要修改一下，以便渲染这些模板。

```
from flask import Flask, render_template
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/user/<name>')
def user(name):
    return render_template('user.html', name=name)
```

Flask 提供的 render_template 函数把 Jinja2 模板引擎集成到了程序中。render_template 函数的第一个参数是模板的文件名。随后的参数都是键值对，表示模板中变量对应的真实值。在这段代码中，第二个模板收到一个名为 name 的变量。

前例中的 name=name 是关键字参数，这类关键字参数很常见，但如果你不熟悉它们的话，可能会觉得迷惑且难以理解。左边的“name”表示参数名，就是模板中使用的占位符；右边的“name”是当前作用域中的变量，表示同名参数的值。

Render_template 中使用一种特殊的占位符{{name}}来表示一个变量，name 变量提示模板引擎何时在渲染模板时使用数据。Jinja2 能识别所有类型的变量，甚至是一些复杂的类型。Jinja2 使用过滤器修改变量，在变量名后添加过滤器名，中间使用”|”分隔。示例如下：

Hello,{{name|capitalize}}

Flask 程序运行过程如下：

1.当客户端想要获取资源时，一般会通过浏览器发起 HTTP 请求。

2.此时，Web 服务器会把来自客户端的所有请求都交给 Flask 程序实例

3.程序实例使用 Werkzeug 来做路由分发(URL 请求和视图函数之间的对应关系)

4.根据每个 URL 请求，找到具体的视图函数并进行调用。

ⅰ在 Flask 程序中，路由的实现一般是通过程序实例的装饰器实现

5. Flask 调用视图函数后，可以返回两种内容:

ⅰ字符串内容:将视图函数的返回值作为响应的内容，返回给客户端(浏览器)

ⅱHTML 模版内容:获取到数据后，把数据传入 HTML 模板文件中，模板引擎负责渲染 HTTP 响应数据，然后返回响应数据给客户端(浏览器)，本系统使用的正是模板内容。

## 4.4 WEB 网页设计

打开浏览器输入网址 http:// 127.0.0.1:5000/即可进入手写数字识别程序。网页界面如图 4.5 所

示：

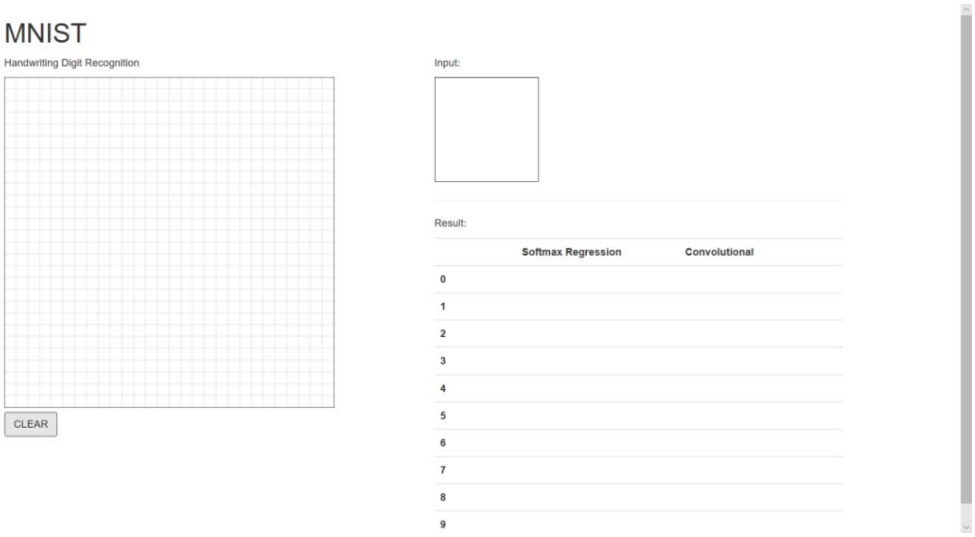



图 4.5 系统初始界面

其中，顶部为 MNIST，全称为 Mixed National Institute of Standards and Technology database，这是一个极其庞大的手写数字数据库。MNIST 下面是 Handwriting Digit Recognition，即为本程序的标题—手写数字识别系统。标题下面是一个 Canvas 画布，可在画布上进行数字的绘制，绘制完毕后，立即在右侧上方显示出所绘数字的缩略图，并在右下方显示出对所绘数字进行两种模型识别的结果，0~9 十个数字的每个数字可能的概率，并对概率最高的一个结果进行高亮显示，即为该模型的最终识别结果，如图 4.6 所示：

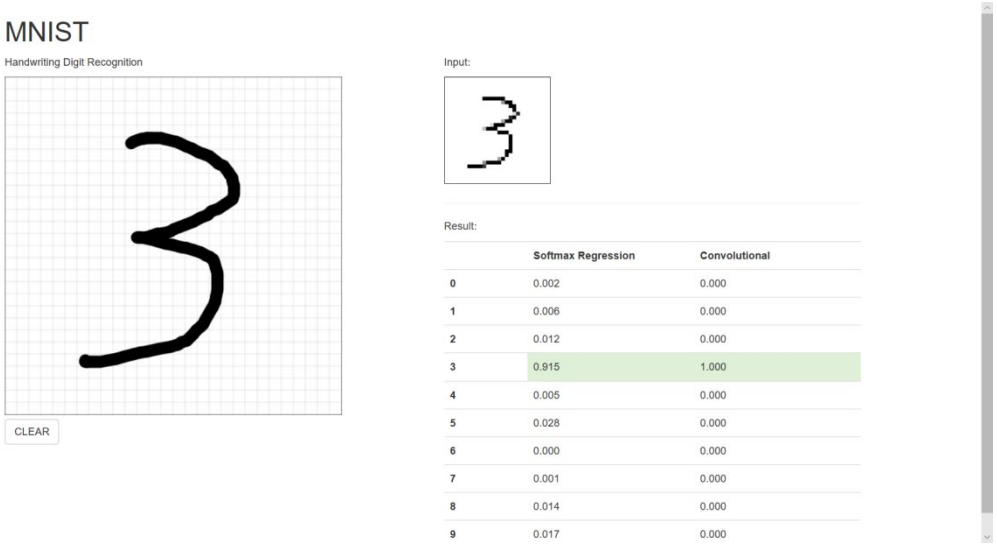



图 4.6 识别结果 1

识别完一次后，可点击左下角的 CLEAR 按钮，点击之后，对画布进行 Initial 初始化操作，画布清零，可在此进行数字的绘制，并识别出结果。

通过对比两种识别模型的识别结果，可清晰的观察出两者的识别率高低。显然，SoftMax Regression 模型对某些数字的识别能力低于 Convolution 模型，对 MNIST 数据集中的 5000 张图片

进行测试后发现，SoftMax Regression 的识别率为 91.92%，而 Convolution 模型的识别率则达到了 99.1%的高识别率，因此可以得出结论：卷积神经网络（CNN）模型对比 Regression 模型而言，更加有效，如图 4.7 所示：

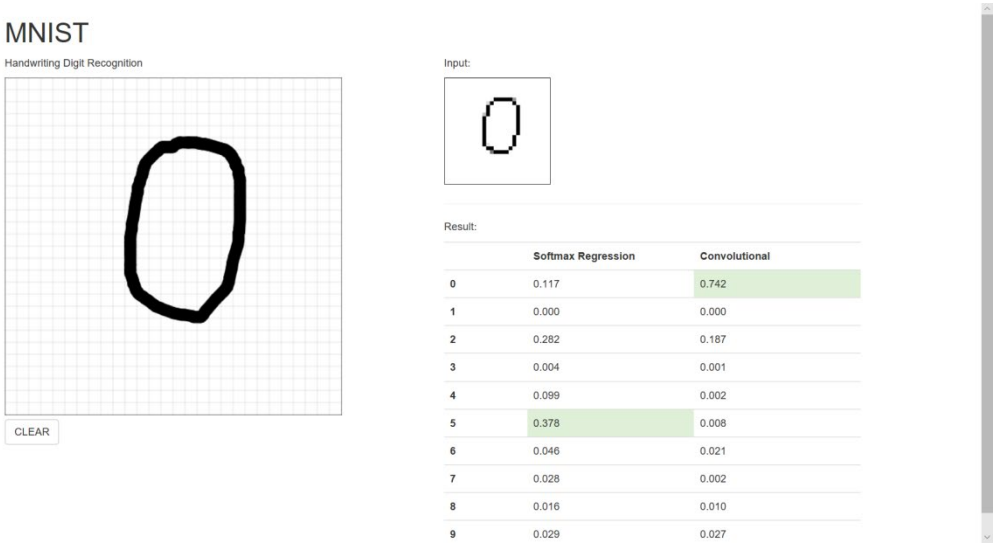



图 4.7 识别结果 2

## 4.5 Ajax

Ajax 即“Asynchronous Javascript And XML”（异步的 JavaScript 和 XML）。Ajax 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。通过在后台与服务器进行少量数据交换，Ajax 可以使网页实现异步更新。

Ajax 是一种异步请求数据的 web 开发技术，对于改善用户的体验和页面性能很有帮助。简单地说，在不需要重新刷新页面的情况下，Ajax 通过异步请求加载后台数据，并在网页上呈现出来。Ajax 占用较少网络数据的传输量，因而达到了提高用户体验感的目的。同时，由于 Ajax 请求获取的是数据而不是 HTML 文档，因此它也节省了网络带宽，让互联网用户的网络冲浪体验变得更加顺畅。

未使用 Ajax 的网页，如果网页有部分被修改，需要更新内容，则必需重新加载整个网页，浪费了大量的带宽，可能造成网络拥堵。而使用 Ajax 技术后，可以只更新那些修改过的数据，无需更新整个页面。这使得 Web 应用程序更为迅捷地回应用户动作，不需要传输那些未曾改变的数据。

本次项目中，每次在画布上写下手写体数字时，都会调用 Ajax 的 post 函数，向服务器提交网页实时修改的那部分数据，而其他部分则不需要改动。在绘制玩手写体数字之后，利用 Ajax 和 Request 技术，将 flatten().tolist()函数返回的每个数字的识别概率传输给服务器，并在页面上显示出来。每次只修改那些有改动的界面，大大减少了数据传输量，减轻了本地服务器的压力，使

每次识别的过程，更加简单、迅速，可以实时的得到服务器返回的数据。Ajax 工作原理如图 4.8 所示：

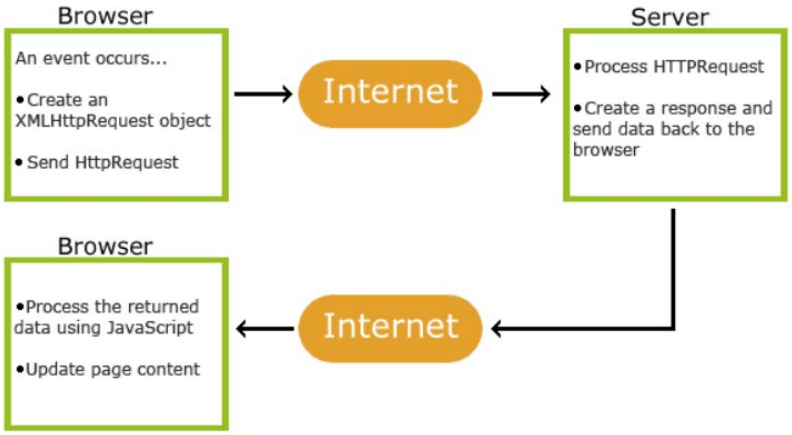



图 4.8 Ajax 工作原理

## 4.6 本章小结

本章介绍了手写数字识别系统的设计与实现，另外还简述了 Flask 框架、WSGI、Ajax、jsonify 等。第一节介绍了 Softmax Regression 模型介绍、模型训练以及模型的应用。第二节介绍了卷积神经网络模型的模型构造、模型训练、前向传播与反向传播、激励函数、损失函数以及卷积层、池化层与全连接层之间的组成。第三节介绍了 FLASK 框架的 Jsonify、Request、Ajax 等技术以及网页界面的概述。

装 订 线

# 第五章 系统测试

为保证手写数字识别系统的正常使用，因此本章节对系统的各个方面进行测试，力求做到运行无误。通过系统测试环节可以直接从用户的角度体验该系统，有助于系统进行更好的完善。

## 5.1 软件测试介绍

软件测试在程序员开发之后确定软件的质量。此过程涉及评估与产品相关的信息。企业在实施软件测试程序时，可以更有效地执行日常活动。软件测试可帮助企业查明软件中的缺陷并进行适当的更正。软件测试还可以帮助企业发现错误和错误，从而提高整体系统容量和准确性

当应用程序质量良好时，即使按下最大容量，它也会持续更长时间并且能够有效地执行。此外，可以对软件进行配置，使其即使在条件不理想的情况下也能正常运行。系统测试还可以提高整体安全性，但系统测试并不是一个简单的过程。每天都会遇到涉及编码和解码的困难挑战。测试过程是软件开发过程中的一个极其重要的阶段，因为必须对每个小模块进行测试以确保其准确性和有效性。

## 5.2 QA 介绍

质量保证或质量保证测试是软件开发过程中的重要一步。通过在开发周期的早期发现缺陷，QA 测试将节省时间和金钱。适当的 QA 测试揭示了新开发的软件中的不一致，错误和冗余。这个过程对于确保正在开发的产品能够在现实世界中生存并且在未来几年内具有长寿命至关重要。QA 测试人员与软件开发生命周期（SDLC）的所有利益相关者和成员进行交互和培养关系，其中包括项目总经理、数据库管理员、软件开发工程师、客户等等。通过这种方式，QA 测试人员可以帮助将所有内容连接在一起，并确保每个阶段都按计划进行。

## 5.3 系统测试

质量测试是系统开发中极其重要的一个环节，不可或缺。现对本手写数字识别系统测试如下：

1) 系统启动测试

打开浏览器，输入网址 http://localhost:5000 或 http://127.0.0.1:5000/并回车，即可看到手写数字识别系统的总界面。本系统使用了 Flask 框架，将 HTML 页面放在本地服务器的 5000 窗口中（Flask 默认为 5000 号窗口，运行端口可修改），系统启动界面如图 5.1 所示：

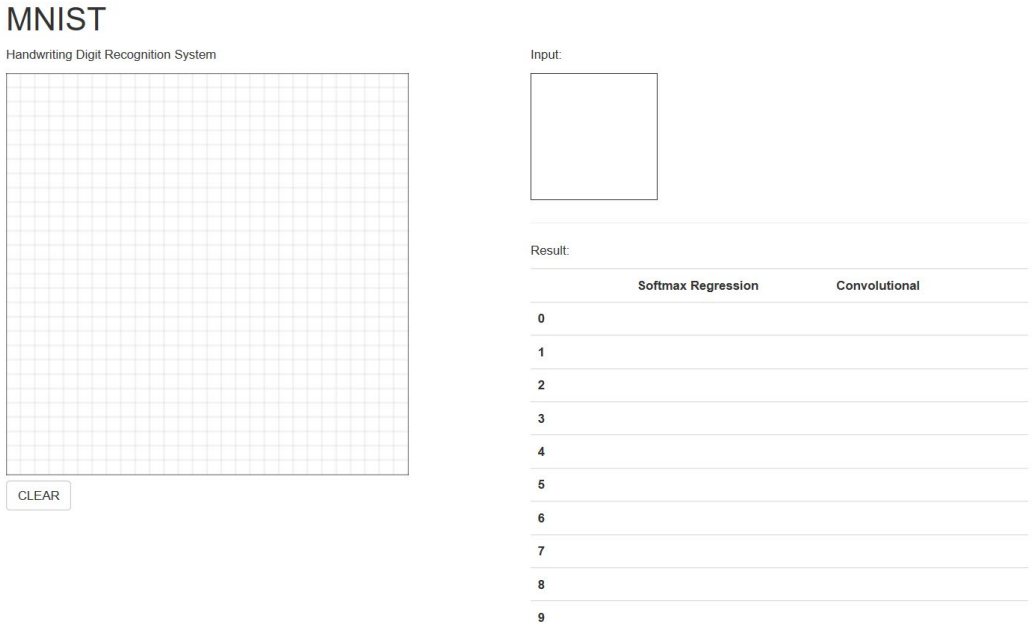



图 5.1 系统启动界面

2) 手写数字测试

在左侧的 Canvas 画布的中央部分进行数字绘制，画笔粗为 16px，画布中每一个小方格均为 16px，而手写数字训练集为 28px，因此将画布的长和宽均设定为 16*28=448px。手写数字绘制后，只需将画布的长和宽分别缩小为之前的 1/16 即可。

在画布上绘制数字后，即可在右上方的输入框中看到刚刚绘制的手写数字缩略图，并在右下方给出两种模型的识别率。例如：绘制数字 3 之后，在起笔时打印输出结果，两种模型的识别率分别如下所示。Softmax Regression 模型的对数字 3 的识别率为 0.56，是 10 个数字中的最高概率，对数字 2 的识别概率为 0.314，因此 Softmax Regression 模型认为该手写数字为 3。反观 CNN 模型，对数字 3 的识别概率为 1.00，而其余数字均为 0.00，由此可见，CNN 模型在本次数字识别中，识别率要远远高于 SoftMax Regression 模型。

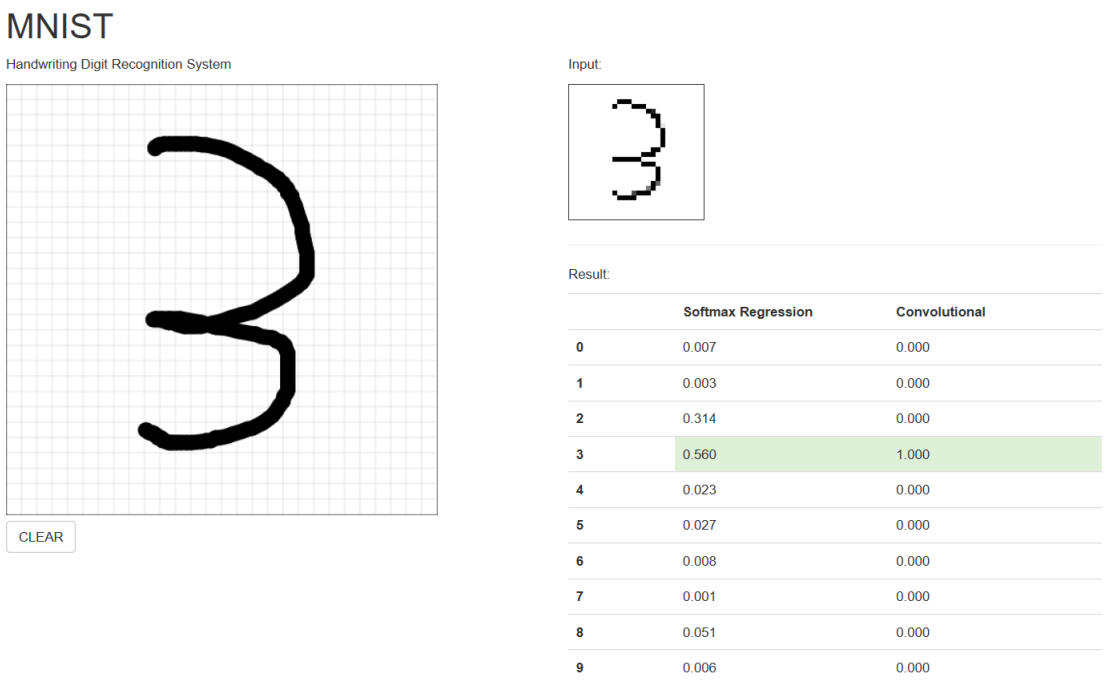



图 5.2 手写数字测试

3) 初始化测试

在识别完一次手写数字之后，点击左下角的 CLEAR 按钮，会调用与该按钮绑定的 main.initialize()函数，对本网页进行初始化操作。初始化操作会将画布、输入框以及结果框全部清零，等待下次手写数字绘制。

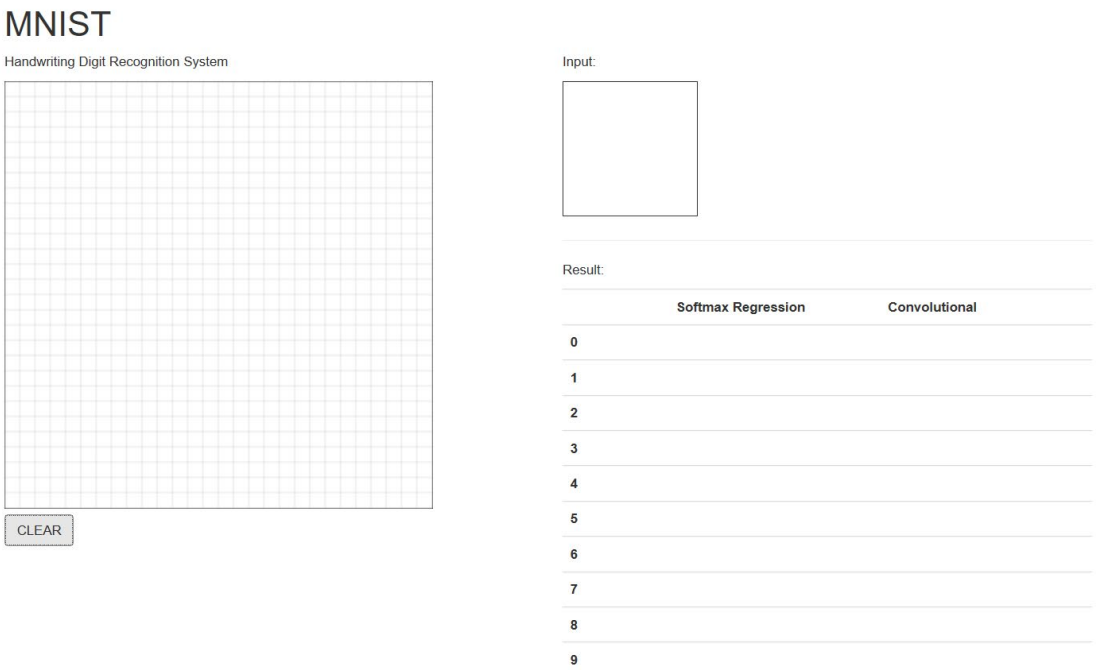



图 5.3 初始化操作结果

4) 识别失败测试

SoftMax Regression 模型的识别率较低，对 1、8、9 这三个数字的识别率尤其更低。因此，在书写不规范的情况下，很可能会识别出错误的结果。此外，运行的 TensorFlow 版本、数据训练的模型量、还有 Canvas 的转换都对识别率有一定的影响，因此一定概率的识别失败是在所难免的。识别失败的测试如下图所示：

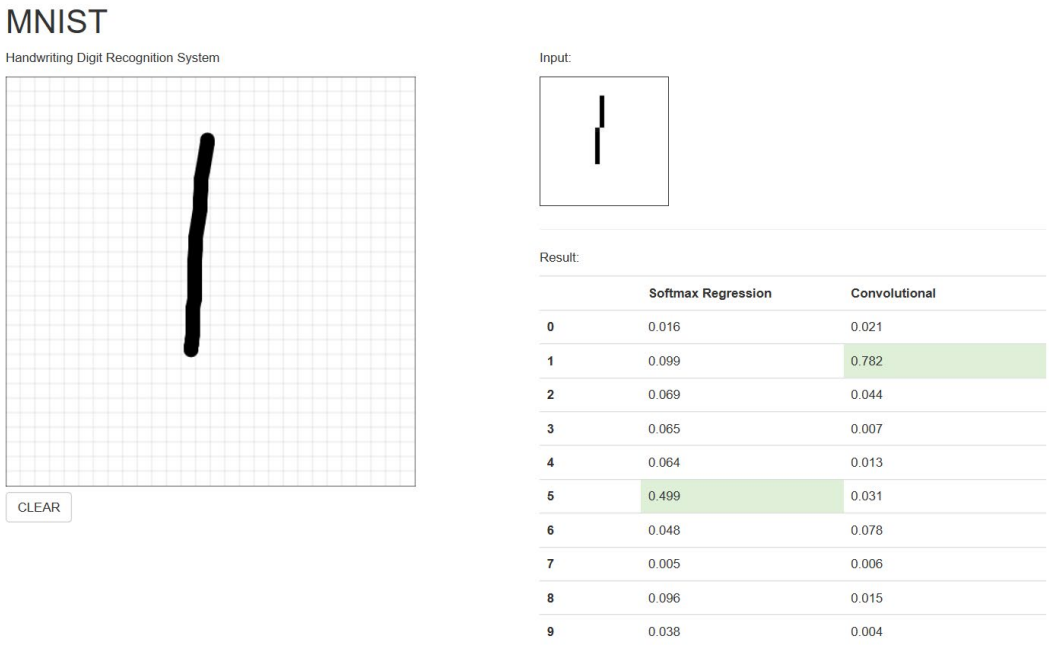



图 5.4 识别失败界面

5) ±45°倾斜手写测试

在某些情况下，由于手写者的自身原因，手写的数字并不一定全是中规中矩的。甚至会出现数字倾斜的情况，本系统需要对该情况考虑在内。因此用户手写数字在±45°倾斜角度之内，都要实现较高的识别率。如下图所示，手写体数字 2 向右倾斜了将近 45°，对于倾斜的情况，两种模型受到了轻微的影响，但仍能进行正确的识别，因此影响可以忽略不计。

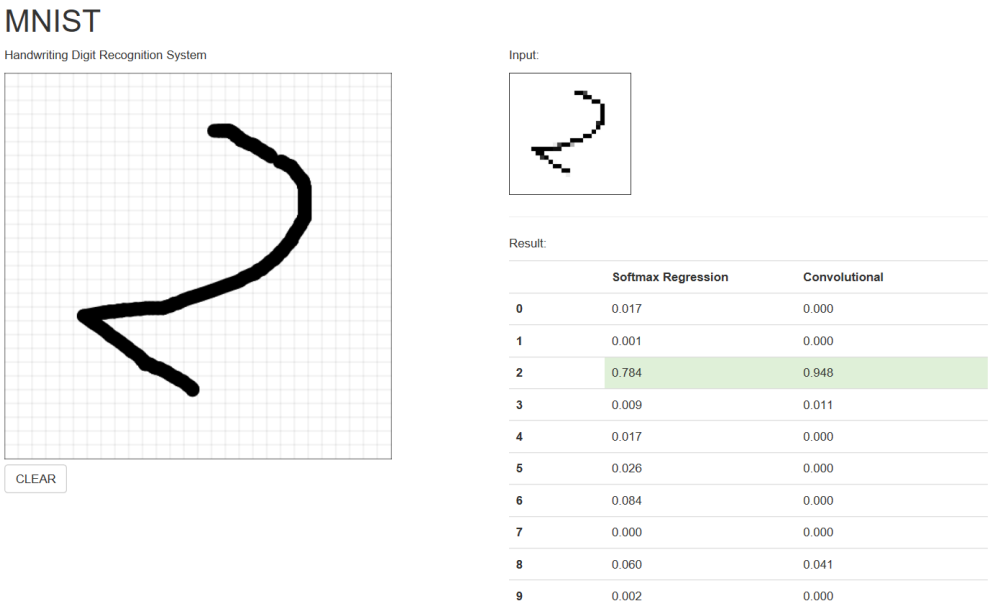



图 5.5 ±45°倾斜手写测试

6) 非正中央绘制测试

由于 MNIST 手写数字测试集中的数字均为在正中央，且对画笔的粗细有一定要求，画笔不能太细。本系统中画布的长宽均为 16*28=448px，而 MNIST 数据集需要转化为 28*28 大小的图像。因此，若没有在画布的正中央书写，则缩小为画布的 1/16 后，只会在某个角中有一个模糊的笔记，人类都很难识别，因此需要在书写时进行一定的限制，尽量书写在画布的中央。缩小前如图 5.6(a)所示，缩小后如图 5.6(b),5.6(c)所示：

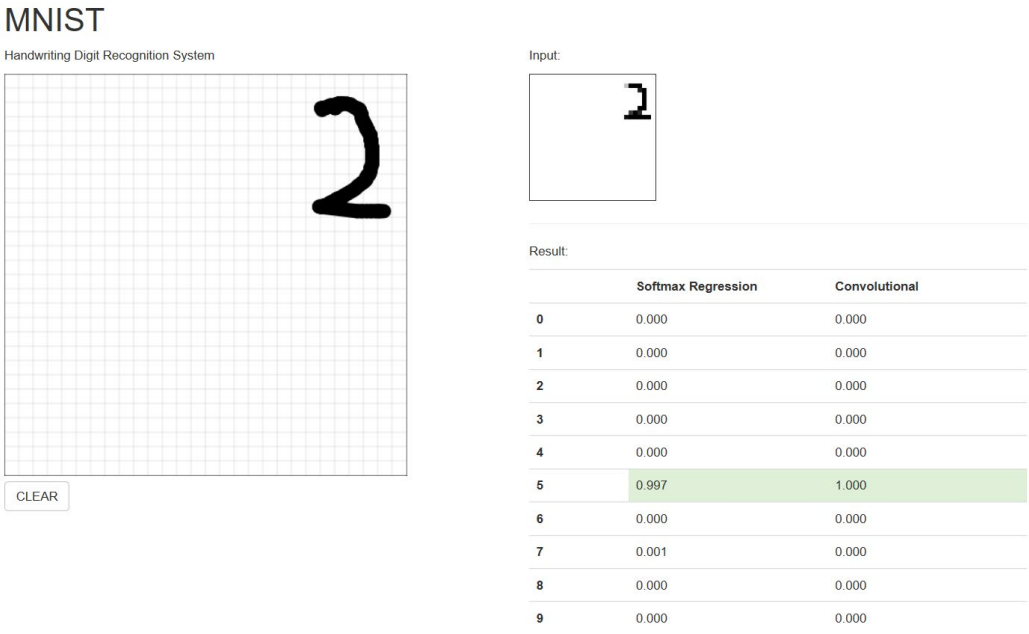



图 5.6(a) 非正中央测试

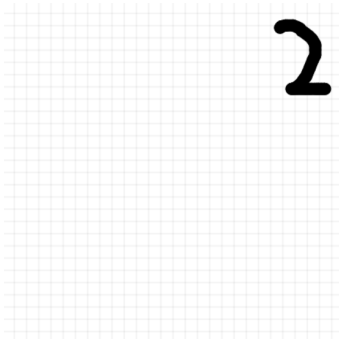


图 5.6(b) 图片缩小前

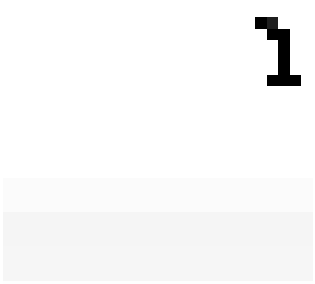


图 5.6(c) 图片缩小后

7) 镜像绘制测试

若书写时，沿中轴镜像翻转 180°，则该手写体数字难以被正确识别，即使是人类，也难以一眼就认出是某个数字的镜像体。因此在绘制时，应限制不能绘制数字的镜像体。

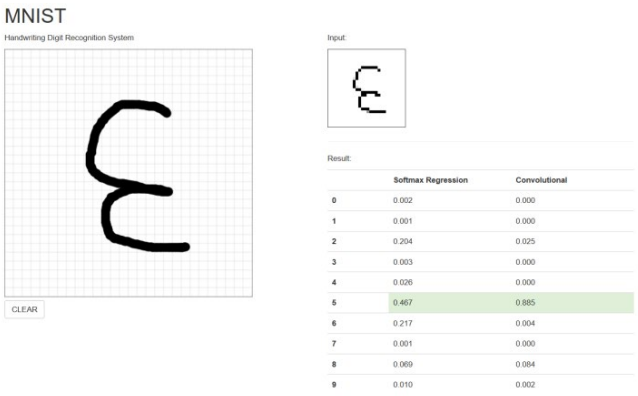



图 5.7 镜像测试

## 5.4 本章小结

本章主要介绍了系统测试的重要性以及 QA 的重要性。第一节简述了软件测试的定义及其重要性。第二节简述了质量保证在软件开发中的重要性。第三节介绍了本系统的几个测试用例。力求使用户在对该系统不了解的情况下，仍能够正常使用，保证系统的鲁棒性。测试用例包含了许多测试失败用例，本系统仍有许多缺陷，有待改进。

装 订 线

# 展望与总结

智能图像分析是人工智能领域的一个有吸引力的研究领域，也是目前各种开放研究难题的关键。手写数字识别是该领域内经过充分研究的子区域，涉及学习模型用于区分预先分割的手写数字。手写数字识别是机器学习，模式识别以及人工智能的许多其他学科中最重要的问题之一。过去十年中机器学习方法的主要应用已经确定了有效的符合竞争的决定性系统对于人类的表现而言，其成就远远超过了手工编写的经典人工智能系统，这些系统在光学字符识别技术的开始使用。但是，并非先前已检查过这些特定型号的所有功能。

研究人员在机器学习和数据挖掘方面的一次次伟大尝试已被设计用于实现从数据中近似识别的有效方法。在二十一世纪，手写数字通信有其自己的标准，并且日常生活中的大多数时间被用作对话手段并记录要与个人共享的信息。手写字符识别的挑战之一完全在于手写字符集的变化和失真，因为不同的社区可以使用不同的手写风格，并且控制绘制其识别的脚本的字符的相似模式。

识别可以提取最佳区分特征的数字是数字识别系统领域的主要任务之一。为了定位这些区域，在模式识别中使用了不同类型的区域采样技术。手写字符识别的挑战主要是由于个体写作风格的大变化。因此，鲁棒特征提取对于改善手写字符识别系统的性能非常重要。如今手写数字识别在模式识别系统领域得到了广泛的关注，并在各个领域得到应用。在接下来的日子里，字符识别系统可以作为通过数字化和处理现有纸质文档来开启无纸化环境[18]。

手写数字数据集本质上是模糊的，因为可能并不总是有尖锐而完美的直线。数字识别的主要目标是特征提取是从数据中去除冗余并通过一组数字属性获得单词图像的更有效的实施例。它涉及从图像原始数据中提取大部分基本信息。此外，曲线不一定像印刷字符一样平滑。此外，目标数据集可以绘制成不同的大小和方向，这些方向总是应该以直接或间接的方式写在使用说明书上。因此，通过考虑这些限制，可以开发出有效的手写识别系统。

装 订 线

# 致 谢

时光如白驹过隙，四年大学时光，匆匆即逝。转眼间，就到了毕业的光景。回首四年岁月，度过了大一时的意气风发，送走了大二时的迷茫困惑，迎来了大三时的幡然悔悟，决定考研，直至今年三月份复试结束，如愿以偿，被中国科学技术大学录取。现在做完了毕设，正在写这篇论文。回头看这四年时光，有幼稚、有迷惘、有奋斗、有期许，总的来说，对大学生活还是比较满意的。四年的学习，使我从一个满脸稚气的小青年，成长为获得一技之长的本科毕业生。还有一个月的时间，就要离开母校了，心中有诸多不舍。不论自己是否是一名合格的毕业生， 毕业之后，都要为祖国，为社会，为家庭竭尽自己所能，奉献自己的一份力。

毕业设计在寒假选题，南春丽老师悉心指导选题问题，最后选择了时下流行的机器学习领域的入门问题—基于 TensorFLow 框架的手写数字识别问题。此问题相当于机器学习界的 HelloWorld 程序，但是由于大学期间对机器学习没有概念，且没有接触过 Python，最难的是 TensorFlow 的安装环境实在是地雷遍布，各种各样的坑都要踩，在这里感谢基地班班长李昀浩同学，在我每次出现问题时，都给我指导解惑，亦师亦友，获益良多。

Python 作为当下最火的语言，市面上也出现了各种各样的 Python 教学课。经同学推荐，我选择了小甲鱼学 Python 系列课程作为我的 Python 入门课。虽说学完 C 语言后，对这些语言应该是触类旁通的，但是为了更好的完成毕业设计，也为了研究生能多掌握一门语言，现在还是扎实的学一下 Python 的基础语法较好。另外，在 Python 学习过程中遇到的问题，可以去鱼 C 工作室提问题，会有一批志同道合的好友（特别感谢一下老白）相互解答，互帮互助。

安装好 Tensor Flow 环境，并了解 Python 的基础语法后，开始在 Bilibili 学习李弘毅的机器学习实战，在这里仿佛开启了全新的世界，机器学习的大门正在向我敞开。李弘毅老师的机器学习课程通俗易懂，诙谐幽默，不懂的问题李老师也在评论区悉心指导。在这里我弄懂了人工智能、机器学习与深度学习的关系，并了解了卷积神经网络如何构建卷积层以及参数如何设定才能达到最优解等等。感谢 Bilibili，感谢李弘毅老师，感谢莫凡老师，没有你们的教导，我可能还在机器学习的大门外游离，没有你们的解惑，我是不可能这么快就完成毕业设计的，再次衷心的感谢你们。

另外，我要感谢我的父母，谢谢你们的关怀与支持，谢谢你们。

最后，感谢所有对我的毕业设计和毕业论文给与指导和建议的每个人。

感谢参与此次论文答辩的专家、教授和老师们，谢谢你们给予的批评和建议，我一定会继续努力，勇往直前的，谢谢你们。

装 订 线

# 参考文献

[1]. 手写数字识别的前景与难点[J]. 张晓. 数码世界. 2016(01)

[2]. 脱机手写数字识别技术研究[J]. 张玉叶,王尚强,王淑娟,王春歆. 电脑知识与技术. 2016(29)

[3]. 楼怡航, 基于数字图像处理的车牌识别技术[J].电子制作:2019

[4]. 郭佳. 基于图像的表格识别算法与自动录入系统[D].北京邮电大学,2018.

[5]. Waris M A, Iosifidis A, Gabbouj M . Object proposals using CNNbased edge filtering [C] / / International Conference on Pattern Recognition. New York: IEEE, 2017:627-632

[6]. Zhang P, Zhuo T, Huang W, et al . Online object tracking based on CNN with spatial-temporal saliency guided sampling [J]. Neurocomputing, 2017, 257:115-127

[7]. Smirnov E A, Timoshenko D M, Andrianov S N . Comparison of regularization methods for image net classification with deep convolutional neural networks [J]. Aasri Procedia, 2014, 6( 1) : 89-94

[8]. 基于深度神经网络的手写数字识别模拟研究[J]. 宋晓茹,吴雪,高嵩,陈超波. 科学技术与工程. 2019(05)

[9]. 王风盼. 基于深度学习的手写数字识别方法研究[D].重庆大学,2018.

[10]. 基于 TensorFlow 实现手写数字识别[J]. 王宇洋. 信息技术与标准化. 2018(11)

[11]. Lecun Y, Boser B, Denker J S, et al. Backpropagation applied to handwritten zip code recognition[J]. Neural Computation, 2014, (4) : 541 – 551.

[12]. 基于 TensorFlow 手写数字识别模型改进[J]. 张哲,张根耀,王珂. 延安大学学报(自然科学版). 2018(04)

[13]. 手写数字识别的原理及应用[J]. 任丹,陈学峰. 计算机时代. 2007(03)

[14]. GPU 加速的神经网络 BP 算法[J]. 田绪红,江敏杰. 计算机应用研究. 2009(05)

[15]. 基于卷积神经网络的计算机视觉关键技术研究[D]. 李彦冬.电子科技大学 2017

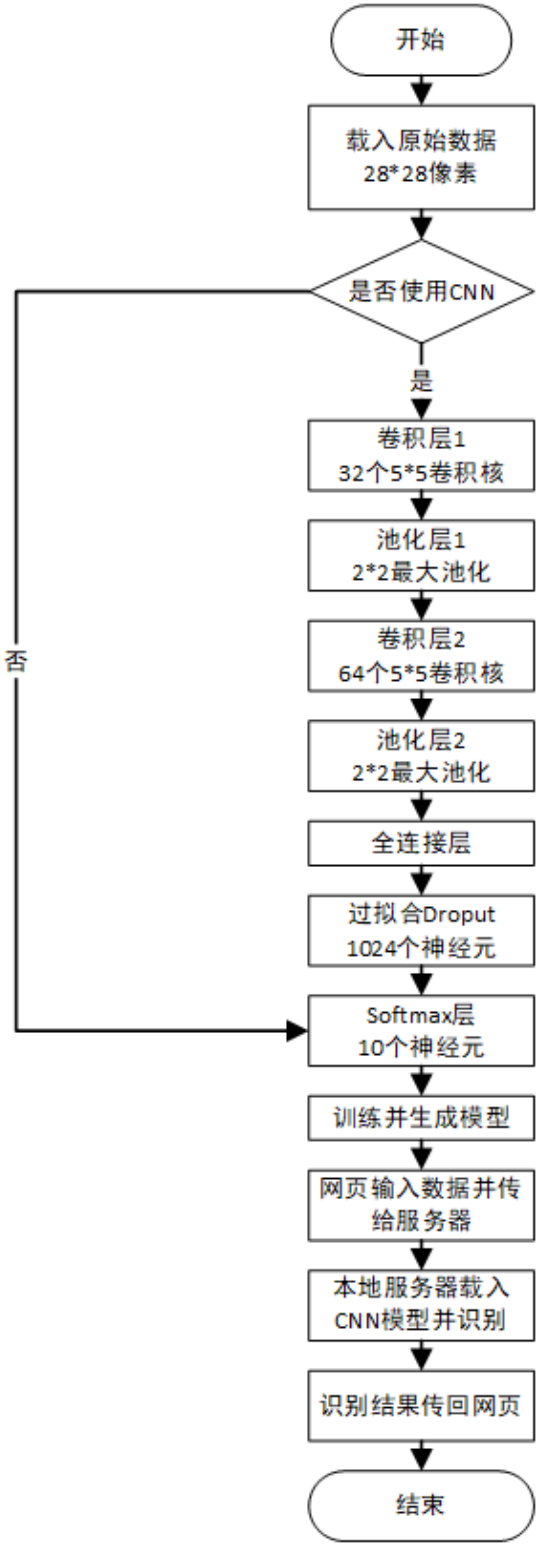
[16]. 王际凯. 基于神经网络的手写数字识别改进算法和系统研究[D].西安电子科技大学,2018.

[17]. 基于卷积神经网络的手写数字识别[J]. 李斯凡,高法钦. 浙江理工大学学报(自然科学版). 2017(03)

[18]. 何西麟.基于深度学习的手写体字符识别研究与实现[D].广州:中山大学, 2015.

装 订 线

# 附 录




系统总流程图