**单位代码：11414**

**学 号：2015011499**

****

|  |  |
| --- | --- |
| **题目** | **基于卷积神经网络的MNIST手写体** |
|  | **识别研究** |
| **学院名称** | **理学院** |
| **专业名称** | **数学与应用数学** |
| **学生姓名** | **操锦雯** |
| **指导教师** | **武国宁** |

**起止时间： 2019 年 1 月 1日 至 2019年 6月 1 日**

# 摘 要

卷积神经网络作为神经网络的一种，具有局部连接，权值共享和空间下采样的特点。局部连接减少了模型参数；权值共享减少网络各层之间的连接，同时有降低了过拟合的风险；下采样大大简化了模型的复杂度，减少模型的参数。近年来卷机神经网络成功应用于语音识别、人脸识别、自然语言处理等方面。手写体识别是一个重要研究课题，在金融、税务、数据统计等领域提供广泛的应用。

本文首先对卷积神经网络的理论进行了归纳整理，在此基础上将卷积神经网络应用于MNIST手写体数字识别数值试验。实验结果表明：使用卷积神经网络来识别手写体数字，准确率非常高，最高可达到99.8%，具有很强的可行性。

**关键词：卷积神经网络；手写体数字识别；权值共享**

Research on MNIST Handwriting Recognition Based on Convolutional Neural Network

# **ABSTRACT**

As a kind of neural network, convolutional neural network has the characteristics of local connection, weight sharing and spatial downsampling. Local connections reduce model parameters. Weight sharing reduces the connections between layers of the network, while reducing the risk of overfitting. Downsampling greatly simplifies the complexity of the model and reduces the parameters of the model. In recent years, the convolutional neural network has been successfully applied to speech recognition, face recognition, and natural language processing. Handwriting recognition is an important research topic, providing a wide range of applications in the fields of finance, taxation, statistics.

In this paper, the theory of convolutional neural networks is firstly summarized. On this basis, the convolutional neural network is applied to the MNIST handwritten digit recognition numerical test. The experimental results show that the use of convolutional neural networks to identify handwritten digits has a very high accuracy rate of up to 99.8%, which is highly feasible.

**Key Words：Convolutional neural network；Handwritten digit recognition；Weight sharing**

目 录

[摘 要 I](#_Toc10321121)

[**ABSTRACT** II](#_Toc10321122)

[第1章 绪论 5](#_Toc10321123)

[1.1 手写体数字识别的研究背景和意义 5](#_Toc10321124)

[1.2 手写体数字识别领域难点 5](#_Toc10321125)

[1.3 卷积神经网络在手写体识别领域研究进展 5](#_Toc10321126)

[1.4 本文主要工作和内容安排 6](#_Toc10321127)

[第2章 卷积神经网络 7](#_Toc10321128)

[2.1 卷积神经网络的结构 7](#_Toc10321129)

[2.1.1 卷积层 8](#_Toc10321130)

[2.1.2 池化层 9](#_Toc10321131)

[2.1.3 全连接层 10](#_Toc10321132)

[2.2 卷积神经网络的特点 11](#_Toc10321133)

[2.2.1 局部连接 11](#_Toc10321134)

[2.2.2 权值共享 12](#_Toc10321135)

[2.2.3 空间下采样 13](#_Toc10321136)

[第3章 卷积神经网络的训练 14](#_Toc10321137)

[3.1 算法的流程 14](#_Toc10321138)

[3.2 卷积层的训练 14](#_Toc10321139)

[3.3 池化层的训练 16](#_Toc10321140)

[3.4 激活函数 18](#_Toc10321141)

[第4章 基于卷积神经网络的MNIST手写体识别研究 20](#_Toc10321142)

[4.1 MNIST数据集 20](#_Toc10321143)

[4.2 前向传播 21](#_Toc10321144)

[4.3 反向传播 22](#_Toc10321145)

[4.4 算法的实现 24](#_Toc10321146)

[4.5 实验结果及分析 28](#_Toc10321147)

[第5章 总结 33](#_Toc10321148)

[参 考 文 献 34](#_Toc10321149)

[附 录A 35](#_Toc10321150)

[致 谢 42](#_Toc10321151)

# 第1章 绪论

## 1.1 手写体数字识别的研究背景和意义

数字是我们生活中非常常见的符号，随着社会信息化的飞速发展，人们不得不处理大量的数字信息。识别手写数字有两种方法，第一种方法是基于手写字符识别算法，且这种算法基于结构特征；手写数字自动识别的实现，是通过模板匹配的方法去识别原始的元素的。虽然这种类型的方法可以较为直接地描述字符的结构，但是它具有字符丢失和缺乏对噪声的鲁棒性的缺点；第二种方法则是手写字符识别算法，这种算法基于统计特征。这种算法基于大量样本的表征，然后进行变换和学习；相应分类器的训练，是基于估计不同样本类别的特征空间分布来完成的，并且分类器用于对未知模式进行分类。如果想用这种方法达到高识别准确率，必须选择足够多的样本基数。根据手写数字识别的特点和要求，分析现有算法存在的问题，利用卷积神经网络进行手写数字识别，卷积神经网络的算法采用多个简单神经元网络的输入和输出，并且不是线性的关系，近似庞杂的函数，拟合观测样本，并提取输入样本的基本特征，实现手写数字的自动识别。[1]为了节省劳动力，物质资源，财力和提高数字信息处理的效率，使用神经网络进行自动数字识别已经成为了一个非常重要的课题。

## 1.2 手写体数字识别领域难点

数字只有10个，即0,1,2,3,4,5,6,7,8,9，它们的笔画很简单，然而，在实践中，一些测试结果表明正确的数字识别率不如印刷的汉字那么高。首先，数字笔画简单流畅，字形不会改变太多，数字很难准确区分；其次，一个数字的写法有很多种，世界各地的人们都在使用数字，并且书写方法都是不同的。所以，其难度很高。[2]

## 1.3 卷积神经网络在手写体识别领域研究进展

目前，手写数字的识别方法主要包括最近邻算法，支持向量机，模型匹配方法，统计决策方法，语法结构方法，模糊识别方法，逻辑推理方法等等。但是，这些方法对复杂分类问题的数学函数的表达能力有限，并且网络的泛化能力有限，存在提取特征难的缺点，无法达到高精度的识别要求，所以识别效果不太好。[3]卷积神经网络的引入为这个问题提供了一个新的解决方案。Hinton和他的同事在2006年的时候提出了深度学习的概念，这是一个机器学习研究的新领域，目的是建立一个模拟人类大脑进行分析学习的神经网络。Lecun和他的合作者们提出的卷积神经网络是第一个真正的多层结构学习算法，它减少了参数的数量，以提高网络驱动器的性能。卷积神经网络通过一系列方法永久地降低了图像识别问题的维数，并最终允许其学习。[4]亚·莱卡提出了卷积神经网络应用到以后几年的手写体识别中，对神经网络模型的研究一直在持续，由于模型的分散性和收敛速度慢的渐变反向传播算法，替换为矢量机和贝叶斯网络等方法。一旦提出了深度学习，卷积神经网络就会逐步改进并重新提出和应用。强大的学习能力和自适应性，以及并行处理能力，容错能力和自动特征提取的功能；这是卷积神经网络的显著优势。

## 1.4 本文主要工作和内容安排

本篇论文的主要研究内容为基于卷积神经网络的MNIST手写体识别研究，把卷积神经网络的方法应用到识别手写体数字中，对模型的结构及其特点进行了详细的分析和推导，将其应用到手写体识别中，对识别准确率进行了详细的研究，分析了样本基数和迭代次数对它的影响。本文内容安排如下：

第一章主要讲述了手写体数字识别的研究背景和意义，手写体数字识别的研究难点，以及卷积神经网络在手写体数字识别领域的研究进展，最后写了本文的主要工作和内容安排。

第二章详细阐述了卷积神经网络的理论，第一节阐述了卷积神经网络的结构框架，第二节讲述了卷积神经网络的特点。

第三章对卷积神经网络模型进行了深入分析，进行了卷积神经网络的公式推导，阐述了算法的流程，然后训练了卷积层和池化层，最后对激活函数进行了详细的研究。

第四章结合上一章的研究结果进一步进行了基于卷积神经网络的MNIST手写体识别研究，介绍MNIST数据集，进行了前向传播和后向传播的推导，运用python实现了算法，并对结果进行了详细的分析。

第五章对总结了本文所研究的内容。

# 第2章 卷积神经网络

## 2.1 卷积神经网络的结构

若干个卷积层，池化层，全连接层是卷积神经网络的基本组成结构。它的常用架构模式是，它对应的首先是A个卷积层的叠加；接着叠加一个池化层，然后重复这个结构B次；最后叠加C个全连接层。也就是说，卷积神经网络是由多个2D平面组成的多层神经网络，并且它的每一层，是由多个独立的神经元来组成的。[5]

卷积神经网络的基本结构如下：一，卷积层，通常被称作特征提取层；每个神经元的输入连接到前一层的局部感受野，并且在提取实体之后提取局部特征，用来确定与其他实体的位置关系。二，池化层，通常被称作特征映射层；每个计算层组成几个特征映射，并且每个特征映射都是一个平面。平面中所有神经元的权重是相等的。一般而言，卷积网络的核心思想是将三种类型的局部连接，权重共享和时间或空间子采样相结合，以获得一定程度的位移，规模和尺度不变性。卷积神经网络主要用于二维数据处理，通过大量的学习来解决输入和输出之间的关系。在卷积神经网络中，交替地配置卷积层和采样层，并且每个层包括多个特征提取器。[6]通过卷积层的卷积运算，可以提取二维数据的特征信息，并且可以减少特征之间的噪声干扰。 采样层采样二维数据，降低分辨率，尽可能地改善图像特征信息，同时减小数据处理的维度，从而提高信息处理的速度。卷积神经网络的直观图如下：

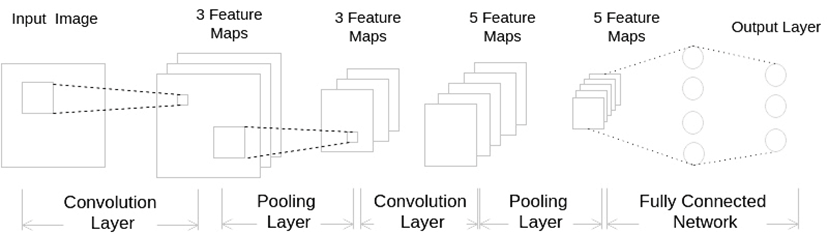


图2.1 卷积神经网络的结构

**Fig.2.1 Convolutional neural network structure**

### 2.1.1 卷积层

卷积神经网络输出值的计算，首先来编号图像的每个像素，字符的解释如下，：图像的第i行第j列的元素；接下来对滤波器的每个权重进行编号，：第m行第n列的权重，：滤波器的偏置项；接着对特征图的每个元素进行编号，：特征图的第i行第j列的元素；用f表示激活函数。然后，运用如下公式计算卷积：

 (2.1)

比如说，对于特征图左上方的元素来说，它的卷积计算方法如下所示：

 (2.2)

接着，特征图的元素的卷积的计算方法是：

 (2.3)

同理，我们可以用上述的方法来计算出要素图中所有元素的值。在以上的计算中，步幅为1，其实步幅可以设置为比1大的数字。当步幅设置成2，特征图变为2\*2。[7]步幅和卷积满足以下关系：

 (2.4)

 (2.5)

：卷积后特征的宽度；：卷积前图像的宽度;F：滤波器的宽度；*P*：0填充的数量（0填充的数量是指在原始图像的周围补几圈0，如果*P*的值为1，那么就补1圈0）；S：步幅；：卷积后特征图的高度；：卷积前的图像宽度。展开一下上公式，就可以获得深度大于1的卷积公式：

 (2.6)

在公式（2.6）中，D：深度；滤波器的大小（宽度或者高度，二者都是相同的）；：滤波器的第d层第m行第n列的权重；：图像的第d层第i行第j列的像素；其他符号的含义和前述相同。在卷积层计算方法中；每个神经元层仅仅连接到上一层部分神经元，并且滤波器的权值涉及所有神经元上层，这一切都是相同的，参数数量与前一层神经元的数量并无关系，这体现出局部连接和权值共享的特点。[8]

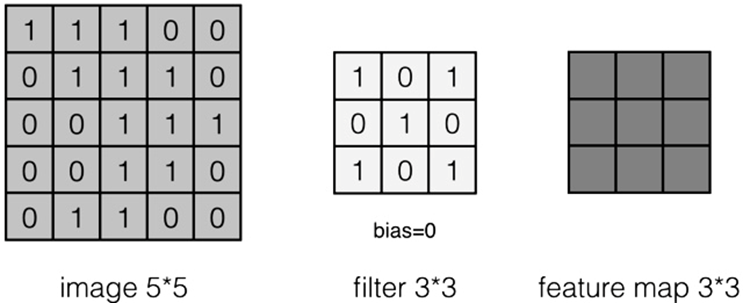


图2.2 卷积层图例

**Fig.2.2 Convolution layer legend**

### 池化层

下采样是池化层的主要作用，可以通过去除特征图中的不重要样本，减少了参数的数量。有很多池化的方法，比如说，最常见的是最大池化法。最大池化实际上也就是将n\*n的样本的最大值作为采样后的样本值。最大池化是一种非线性下采样。最大池化将输入图像分离成一系列非重叠矩形，并且对于每个这样的子区域，显示最大值。由于它为位置提供了额外的稳健性，所以最大池化是一种降低中间表示维度的“智能”方法。池化的效果是即使数据量大大的减少，特征的统计属性依旧可以描述图像，并且由于降低了数据的维度，高效地避免了过度拟合的现象。为了得到更好的结果，根据下采样的方法，池化分为最大值下采样与平均值下采样。[9]下图表示的是我们常用的最大池化方法的图例：

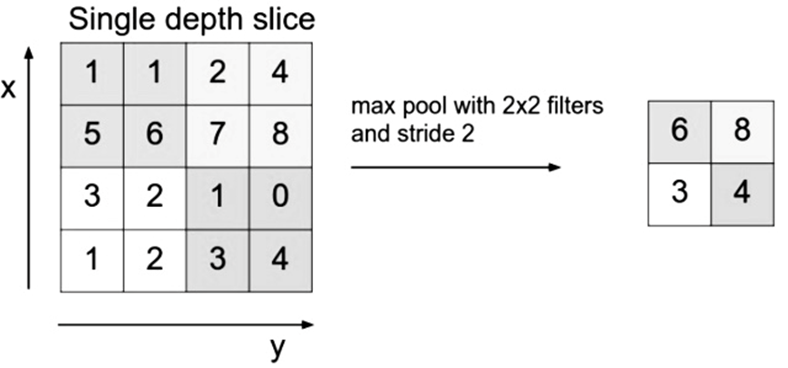


图2.3 池化层图例

**Fig.2.3 Pooled layer legend**

### 2.1.3 全连接层

全连接神经网络的特性为：第一，在层中放入神经元，我们称最左边的层是输入层，将接收输入数据的最右层称作输出层，我们可以从这个层来获取神经网络的数据。输入层和输出层之间的层被称为隐藏层。第二，同一层的神经元之间没有连接。第三，每个第N层神经元连接到第N-1层的所有神经元（即，完全连接的方向），第N-1层神经元的输出是第N层神经元的输入，这是一个输入。最后，每个连接都有一个权值。[10]

输出层前面的最后一个层是全连接层，其功能是连接所有的要素，并且将输出值发送到分类器以获得最终结果。卷积层、池化层等操作的目的是将原始数据映射到隐藏层特征空间；而全连接层的作用，是在样品标记空间上映射“分布式特征表示”。在实际使用的过程中，通过卷积运算来实现全连接层。

之所以说全连接神经网络不太适合图像识别任务，是因为以下几个原因：第一，参数数量太多；输入图像1000\*1000像素(即100万像素，现在已经不能算是一个大图像)，输入层为一百万个节点。如果第一个隐藏层中只有一百个节点，那么这一层就有(1000\*1000+1)\*100=1亿个参数，参数非常多；图像只是略微膨胀，参数数量就会变得多很多，这就导致图像的扩展性非常差。第二，在图像识别任务中使用像素之间的位置信息，每一像素与其周围像素之间的关系相对近，与远程像素的接触可能非常少。当我们学完每个连接权重之后，我们发现，有大量权重的值很小(也就是说，这些连接并不重要)。学习海量不重要的权重，这件事非常无效。第三，网络层数的限制，显而易见，网络层数越多，它们的表达能力就越强，但是很难通过梯度下降法使完全连接的神经网络具有深度。[11]因此，我们不能获得很深的全连接神经网络，从而限制了它的有效性。

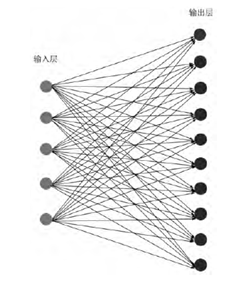


图2.4 全连接层图例

**Fig.2.4 Full connection layer legend**

## 2.2 卷积神经网络的特点

### 2.2.1 局部连接

在传统的多层反馈神经网络中，各层神经节点都是一维线性排列，并且这些层与层的中间节点是完全连接的。在卷积神经网络中，基于层间的神经节点没有完全连接,不过，但是使用层之间的局部空间相关性，并且各隐藏层被使用。正中间的神经元是上层输入神经元的一部分，也就是局部连接，这样可以减少很多的参数，由于全连接中网络参数过多，无法有效地学习神经网络生成的神经网络存在的问题。通过局部感受野，神经元可以提取边缘和角落的基本视觉特征。这些特征被发送到更高层次的神经元中，这些神经元经过处理后可以结合更复杂的视觉特征。层m中的隐藏单元的输入来自层m-1中的单元的子集，具有空间连续的接收场的单元。[12]我们可以用图形说明如下：

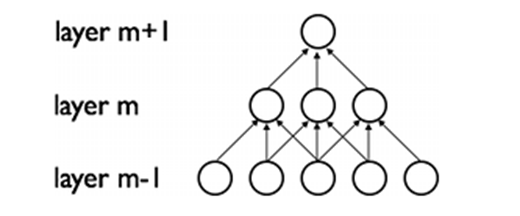


图2.5 局部连接图例

**Fig.2.5 Local connection legend**

想象一下，m-1层是输入视网膜。在上图中，层m中的单元在输入视网膜中具有宽度为3的感受区域，因此仅连接到视网膜层中的3个相邻神经元。层 m + 1中的单元与下面的层具有类似的连通性。我们说他们关于下面层的感知场也是3，但他们对输入的感知场更大。每个单元对其视网膜的感受区域之外的变化没有反应。

### 2.2.2 权值共享

因为自然图像具有统一的统计分布，这使得基本特征提取器能够应用于特定的部分，也可以应用于整个图像。因此，在卷积神经网络中，卷积层中的每个滤波器重复的作用于整个感受野，卷积输入的图像，提取图像的局部特征，并形成特征图。

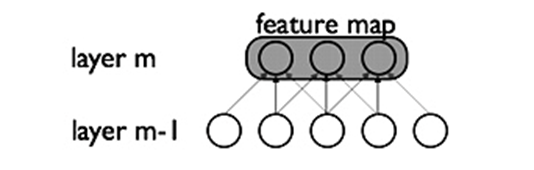


图2.6 权值共享图例

**Fig.2.6 Weight sharing legend**

在上图中，我们显示了属于同一特征映射的3个隐藏单元。相同颜色的权重被共享约束为相同。以这种方式复制单元允许检测特征，不管它们在视野中的位置。并且，通过显著减少所学习的自由参数的数量，权值共享提高了它的学习效率。约束模型使得卷积神经网络能够更好地推广视觉问题。[13]

### 2.2.3 空间下采样

空间下采样是一种非线性采样法，通常汇集输入特征量图内的各n\*n区域的要素，目前最常见的是：最大值汇总、平均值汇总和随机汇总。运用卷积得到图像特征，然后分类这些特征。提取的所有特征数据都被用于分类器训练，通常，如果产生大量的计算，那么必须使用下采样法来减少图像的卷积特征，以便获得图像的卷积特征。同时，如果特征图中的连续范围被选择为下采样区域，并且只操作由相同隐藏单元生成的特征的话，则这些下采样特征会具有一定的平移不变性。当输入的图像受到了微小的平移，进行卷积操作和下采样操作，提取出一样的特征图像，这对于特征分类来说是有好处的。[14]

# 第3章 卷积神经网络的训练

## 3.1 算法的流程

卷积神经网络，是从输入到输出的一种映射；通过使用已知的模式去训练卷积神经网络，网络就可以映射输入和输出对。这种训练算法和传统的BP算法有些类似，它主要由四个步骤组成，这四个步骤可以分为两个阶段：即前向传播阶段和反向传播阶段。通过卷积神经网络，不断的重复两个阶段，不断的更新参数，训练集训练的越来越深，误差也就逐渐减小。

通过链式求导，计算出每个权重的损失函数的偏导数，并根据梯度下降方程来更新权重。步骤如下所示：第一，前向计算；算出每个神经元的输出值（j的意思是卷积神经网络的第j个神经元）。第二，反向计算；算出每个神经元的误差项，它是卷积神经网络的损失函数对神经元加权输入的偏导数，公式为。第三，计算出每个神经元的连接权重的梯度，它表示从神经元i到神经元j的权重，公式为，：神经元的输出。最后，用梯度下降法来更新每个权重[15]。

## 3.2 卷积层的训练

卷积层误差项的传递，以步长是1，输入深度是1，滤波器个数是1为例子。：第l-1层第j列的误差项，：滤波器第m行第n列权重，：滤波器的偏置项，：第l-1行第j列的神经元的输出， ：第l-1行神经元的加权输出， ：第l-1层的激活函数。皆为数组，由组成的数组用来表示，卷积操作用来表示。公式如下所示：

 (3.1)

 (3.2)

由链式求导法则可知，

 (3.3)

 (3.4)

第l层的滤波器的权重数组用来表示。把上式的卷积展开成求和的形式：

 (3.5)

 (3.6)

所以

将上式表达成卷积的形式：（表示将矩阵中每个对应元素相乘）

 (3.7)

卷积层权重梯度的计算

 (3.8)

(3.9)

第l-1层的输出为，第l层filter的权重为，第l层的敏感图为。由于权值共享的原因，权值影响了所有的。依照上述计算可以得出



我们可以推出： ；

偏置项的梯度为：



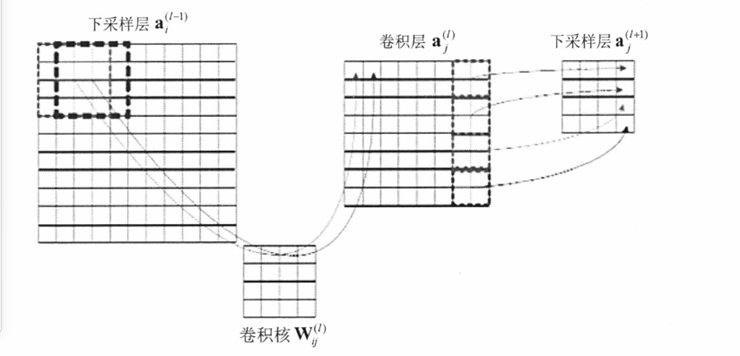


图3.1 卷积层训练图例

**Fig.3.1 Convolutional layer training legend**

## 3.3 池化层的训练

1，最大池化误差项的传递： ，假设最大值为，那么上式等同于，可以求得，，，，所以，同理，其余三项为0。所以可以得出，在最大池化中，将下一层的误差项的值传递给与前一层的神经元，并且和前一层的对应块中的最大值相对应，但是其他神经元的误差项的值为0。[16]

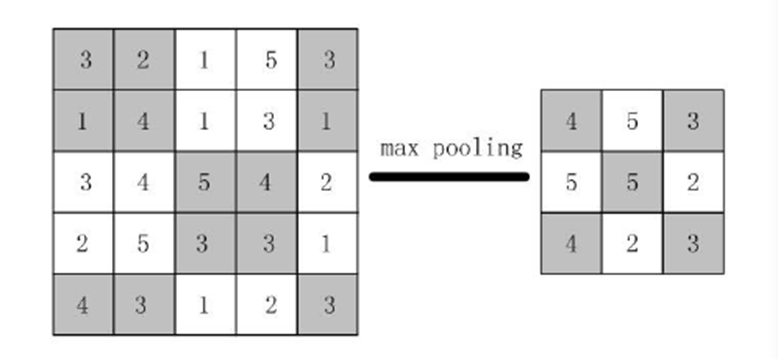


图3.2 最大池化图例

**Fig.3.2 Max-pooling legend**

2，平均池化误差项的传递：



，，，

由链式求导法则得到

同理，得到

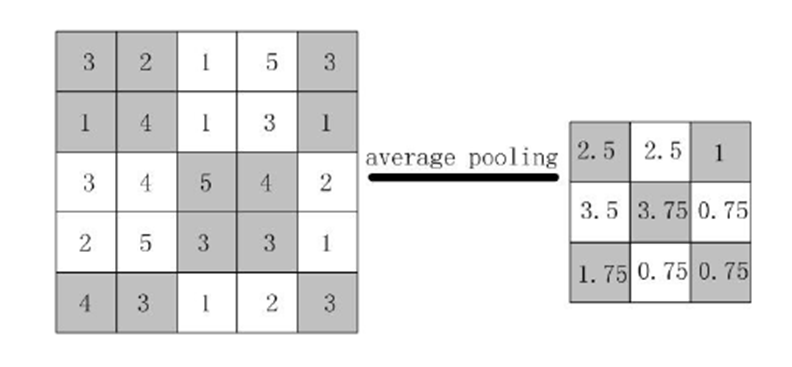


图3.2 平均池化图例

**Fig.3.2 average pooling legend**

可以得出，平均池化中，下一层的误差项的值被均等地分配给前一层的相应块中的所有神经元。

## 3.4 激活函数

卷积神经网络的一个重要部分是激活函数，它的目的是解决训练集数据中可能存在的非线性问题。实验表明，当仅使用一个激活函数来处理数据时，准确度不令人满意。所以，结合传统的激活函数选择方案和改进来选择适当的激活函数是非常有必要的。在卷积神经网络中，激活函数通常分成两种，一种是饱和非线性激活函数，另一种是不饱和非线性激活函数。只有在添加激活函数之后，深度神经网络财具有分层非线性映射学习能力[17]。

Sigmoid函数的表达式为，这是一种比较传统的激活函数，这种激活函数和生物神经元相似度很高，能在较浅的神经网络层中得到比较好的分类效果。但是它的缺点也较为明显，比如，具有软饱和性，容易出现梯度消散的问题；输出不以0为中心，容易导致数据分散，分类效果不佳的问题。为了解决上述的问题，我们引入另一种激活函数，双曲正切函数tanh函数：，该激活函数的分类效果优于sigmoid函数。relu函数：，该激活函数有速度快的优点， relu激活函数计算成本要小得多。该激活函数还有减轻梯度消失问题的优点，是计算梯度的公式，是sigmoid函数的导数，当使用反向传播算法计算梯度的时候，对于每个sigmoid神经元层，将梯度乘以。该函数的最大值为1/4。因此，乘以使得斜率变得越来越小。对于深度网络的训练来说，这是一个大问题。 relu函数的导数为1，梯度不会减小。激活函数只是减小斜率的一个因素，但是relu激活函数的效果要好于sigmoid，可以运用relu函数是训练更深层次的网络。Relu函数的另一个优点是稀疏性，对大脑的研究表明，当大脑运作时，只有大约5％的神经元是活跃的，并且使用sigmoid激活功能的人工神经网络的激活率约为50％。 一些论文上说人工神经网络是很理想的，激活率为15％-30％。 如果输入小于0，则relu功能完全无效，导致激活率降低。在以往的卷积神经网络中，往往会用sigmoid函数和tanh函数来做激活函数，但是它们有收敛速度慢和梯度弥散产生的问题，例如产生网络不收敛的问题。所以现在往往会采用relu函数作为激活函数。Softplus函数表达式为。[18]比较上述四种激活函数，根据它们的特性和优点，如果输入特性更复杂或相位差得不是特别大，当需要进行细微的分类决策时，Sigmoid函数效果会更好一些。当使用Sigmoid函数作为激活函数时，输入要素数据需要归一化，否则激活值会落入平坦区域，导致所有隐藏层输出都会收敛。Relu函数的激活功能是稀疏的，并且Softplus激活功能比Relu功能更平滑，且激活功能在时变得更有利。

# 第4章 基于卷积神经网络的MNIST手写体识别研究

## 4.1 MNIST数据集

训练数据集是我们首先要做的事。我们使用的数据集是MNIST手写体数据训练库，该数据库由Lecun构建。使用MNIST数据集，这包括数万个手写数字的扫描图像和正确的分类。名称MNIST来自国家标准和技术研究所MNIST收集的两个数据集的修改子集。

MNIST数据分为两部分。第一部分包括六万张用于训练数据的图像。图像为灰度，大小是28\*28像素。它的第二部分是用作测试数据使用的10,000张图像。同样，这些是28\*28的灰度图像。使用测试数据，评估新网络为了识别数字而学习的程度。为了进行性能恰当的测试，测试数据将从原来的训练数据中获得的不同组250人的测试数据。这帮助我们相信，我们的系统能够识别出训练期间没见过的人写的数字。作为训练集和测试集的MNIST数据库，训练样本集是60,000，测试样本集是10,000。MNIST库的原始图像尺寸是28×28，缩放是32×32，数据值范围是[-1,1]，并且扩展值是作为输入层输入数据，它被认为是-1。在Label的情况下，输出层有10个节点，相应位置的节点值设置为0.8，其他节点设置为-0.8作为输出层数据[19]。



图4.1 数据集图例

**Fig.4.1 Data set legend**

## 4.2 前向传播

卷积神经网络的前向传播可分为：卷积层的前向传播，下采样层的前向传播，全连接层的前向传播和softmax层的前向传播。

一，卷积层的前向传播的重点是特征图的提取过程。假设一个从第l-1层到第l层是卷积层的前向传播过程，第l-1层有个特征图，并且它的第i个特征图是，第l层有个特征图，它的第j个特征图是。其之间的卷积核是，偏置项是。矩阵-向量方式对这一过程的表示如下：

 (4.1)

 (4.2)

二，通过下采样层的前向传播，得到输入特征图通过降采样后的结果。假设一个从第l层到第l+1层的下采样传播过程，第l层有个特征图，第l+2层有个特征图，表达如下：

 (4.3)

是降采样函数，由于这个功能对第1+1层输入特征映射的不同n\*n区域中的每一个执行聚合操作，因此l+1层的输出特征映射比每个维度中的输入特征映射小n倍。[20]

三，全连接层与传统的多层前馈神经网络的前向传播过程是一样的。假设第l层的激活响应，第l+1层的激活响应,设两层之间的权重是，设偏置项为，前项算法的计算步骤如下：

 (4.4)

 (4.5)

四，softmax回归，是为了解决多分类问题。针对多分类问题，softmax层属于卷积神经网络的输出层。假定一个网络有L层，softmax层的前向传播如下：

 (4.6)

 (4.7)

## 4.3 反向传播

卷积神经网络中神经元节点的连接数远多于参数数量，是因为权重共享和下采样的影响。本节主要介绍的是反向传播算法和卷积神经网络的形式化推导，推导过程着重于基于二维图像数据多分类问题计算“误差敏感项”的方法和相关网络层的梯度。接着用求得的梯度校正网络的参数。

一，对于多分类的问题，将单个样本的目标函数设置为如下交叉熵的形式：

 (4.8)

通过不断地最小化目标函数和修改网络参数，我们可以渐渐地减小样本的实际标签分布与模型的输出信号分布之间的差距，最后卷积神经网络是可以很好地拟合数据，做出正确的分类判断。所以，神经网络参数的解决方案，实际上是一个非线性优化问题。也就是在解空间中找到一组最佳参数和，使得：

 (4.9)

该解决方案可以使用批量随机梯度下降以及传统的多层前馈神经网络训练，但是该过程需要使用反向传播算法。我们为网络的每一层的每一个节点定义一个误差敏感项，这种“误差敏感”可用于测量每个神经元节点对网络最终输出层的误差的贡献。softmax层通常属于网络的最后一层。当给定网络层数L时，对于softmax层的每一个神经元节点，它的误差敏感项计算如下：

 (4.10)

 (4.11)

 (4.12)

 (4.13)

进一步来求出目标函数对于softmax层的参数W和b的偏导数：



 (4.14)

一，全连通层反向传播算法如下，l+1层中每个节点j的误差灵敏度是,对于第l层中的每个节点i来说，它的“误差敏感项”计算如下所示：

 (4.15)

 (4.16)

 (4.17)

在获得网络中每个节点的误差敏感项之后，这可以用于计算损失函数，对于参数W和b的偏导数，用于计算偏导数的公式与上式是相同的。

三，下采样层的反向传播；卷积层l之后是下采样层l+1。下采样层的特征图中的每个元素对应于卷积层的特征图中的区域。所以，我们说，卷积层l的每个单元，仅连接到下采样层的l+1层的一个单元。如果已知l+1层的每个节点的误差信号，则计算l+1层（对应于层的每个神经元节点）的误差信号图，用来有效地计算层l的相应位置的误差信号。对误差信号的二维映射进行上采样的操作，然后对上采样的误差信号映射和层l“激活函数部分映射”，执行基于元素的乘法。（矩阵符号）表示图层第j个要素图中所有节点的误差敏感项。图层反向传播的向下传播如下：

 (4.18)

在这里，表示上采样操作，其可以根据在下采样期间执行的聚合操作而改变，但是上采样之后的误差信号映射匹配卷积层的特征映射大小。

当计算卷积层中每个元素的“误差敏感项”的时候，可以将所有，求和相加以计算偏差项的目标函数的偏导数。



并且对于权重共享，卷积核的目标函数的偏导数包含所有梯度的求和。



在上式中，第i个特征图在卷积的过程中，当卷积核移动到坐标时，和它相乘的那一块区域，我们用来表示。

四，卷积层的反向传播；假设网络的第l层是卷积层，那么当l-1层是下采样层的时候，卷积层l-2的输出到下采样层l-1的输入的前向传播仅是下采样操作，并且也不没有参数。所以，这个层的反向传播不必相对于网络参数计算目标函数的偏导数，并且不包括梯度的计算。下采样层l-1的输入到输出的过程没有经过非线性激活函数。也就是说，，故。下采样层l-1中第i个特征图的位置的元素所对应的误差敏感项的公式如下所示：

 (4.19)

## 4.4 算法的实现

卷积神经网络是由各层由多个独立的神经元构成的多个二维平面构成的多层神经网络。典型的卷积神经网络模型是LeNet-5，包含输入在内，一共有8层。本文所设计的卷积神经网络结构为“卷积层-卷积层-池化层-卷积层-卷积层-池化层-全连接层-全连接层”；输入层的神经元是32\*32，C1层的神经元数量是6\*28\*28，S2层的神经元数量是6\*14\*14，C3层的神经元数量是16\*10\*10，S4层的神经元数量是16\*5\*5，C5层的神经元数量是120\*1\*1，输出层的神经元数量是10\*1\*1。执行过程可以表述如下：第一，在MNIST数据库，得到训练样本，加测试样本的数据；[-1,1]是其取值的范围。第二，初始化权值和偏置。第三，前向传播，依照权值和偏置来计算神经元的值，输入的是数据。第四，反向传播，为了更新权值和偏置，需要计算每个层神经元，权值和偏置的误差。第五，更新各层权值和偏置。第六，测试准确率。最后，对输入的图像数据进行识别。

运用卷积神经网络识别手写体数字的流程可以概括如下：首先，在初始化网络中的参数之后，当输入训练集数据时，网络重复多次前向传播和反向传播，最后达到最大迭代次数，或输出实际训练集与理论输出值落在允许范围内。 在该过程中，连续更新和保存由每次迭代产生的诸如权重和偏移的特征参数，并且最终获得数字手写字符的卷积神经网络识别网络。 然后，将测试集数据输入到训练的卷积神经网络，来识别网络，用来获得相应的识别结果。采用Dropout层来避免过拟合。如果以前的网络公式为：

 (4.20)

 (4.21)

采用Dropout层以后，公式变成：

 (4.22)

 (4.23)

 (4.24)

 (4.25)

Dropout层可缓解过拟合有如下原因：第一，取平均的作用；先返回标准模型，使用相同的训练数据训练5个不同的神经网络，通常会得到5个不同的结果。结果是“取平均值”或“大多数获胜投票策略”，以确定最终结果。例如，如果三个网络的结果9，则实际真正结果是9，而其他两个网络很可能给出错误的结果。这种“整体平均”的策略通常可以有效防止过度拟合问题。不同的网络可能导致不同的匹配，因此平均可以导致“相反”匹配相互抵消。丢弃掉不同的隐藏神经元，这类似于训练不同的网络。随机移除掉一半隐藏的神经元会导致不同的网络结构。不同的网络具有不同的过度拟合，因此一些“反向”配件相互抵消并且整体过度拟合减少。第二，减少神经元之间复杂的协同适应关系；Dropout程序是因为每次两个神经元不一定出现在一个Dropout网络中。此类权重更新不依赖于隐式节点与固定关系的交互，并且防止仅在某些其他特征下启用某些特征的情况。网络被迫学习更强大的功能，这些功能也存在于其他神经元的随机子集中。如果我们的神经网络正在进行某种预测，它对某些线索不应该过于敏感，如果我们失去一些线索，我们将从许多其他线索中分享一些共同的线索。即使某些神经元连接丢失，减轻重量也会使网络更加强大。第三，Dropout类似于性别在生物进化中的作用：物种往往适应这种环境生存，环境突变可能使物种难以及时作出反应。它有效地防止过度拟合。通过先前计算的每层的权重，每层的权值误差，每个层的偏置，每个层的偏置误差和学习速率用于更新层权重和偏置。当前层用l表示，当前层的输出如下表示：

 (4.26)

上式中，表示当前层的输入，表示l层特征图的权值，表示上一层的输出，表示当前层的额外偏置，f表示激活函数。使用卷积核对上层特征映射进行卷积，然后使用激活函数获得卷积层的输出函数映射。卷积层的计算如下：

 (4.27)

l：网络层数；：采样层的第j个神经元的输出；：上一层的第i个神经元的输出；：前一层第i个神经元和前一层第j个神经元之间的权值（卷积核）每个特征图都有不一样的卷积核，模板为5\*5；：当前层，第j个神经元额外偏置；f表示激活函数。

在采样层之中，对先前卷积层的特征映射下采样，采样后的输入和输出特征映射的数量不变，计算格式如下：



n：从卷积层到抽样层的窗口的大小;：选择的输入特征图的集合;

对于单个的样本，用代价函数表示它经过网络产生的误差，公式如下：

 (4.28)

用平方误差代价函数来表示m个训练样本来表示，如下：

 (4.29)

在实验期间,将惩罚项添加到平方误差成本函数,以防止网络过度拟合：

 (4.30)

：第i个样本的理想输出；：第i个样本对应网络的实际输出；：权重衰减参数；：均方误差，表示代价函数；：权重的衰减项，减小权重幅度，防止过拟合。

在反向传播的过程之中，更新对应于每个第l层神经元的每个权值的权重，首先确定第l层的每个神经节点的灵敏度，然后确定第n层（输出层）的每个节点的灵敏度。灵敏度计算如下所示：

 (4.31)

的各个层中，如下是当前层l，每个神经节点i对应的灵敏度的计算公式如下：

 (4.32)

 (4.33)

 (4.34)

把上式中的n-1和n替换成l和l+1，得到：

 (4.35)

的各个层的灵敏度是：

 (4.36)

各层中误差对W和b的偏导数，表示如下所示：

 (4.37)

 (4.38)

对层l中的参数W和b，进行调整和更新，表示学习率，公式如下：

 (4.39)

 (4.40)

这个过程可以总结如下：首先，我们需要初始化卷积神经网络的权重和偏置，以便模型可以稳定地执行后续迭代训练。接下来，执行前向传播和后向传播的多次迭代，每次迭代更新由前一次迭代生成的权重和偏移。在完成设定的迭代次数之后，最后生成的层权重和偏移量存储在数据文件中。目的是在实时的数字识别中，节省训练过程所花费的较长时间，达到识别目标的快速性，稳定性和准确性。

## 4.5 实验结果及分析

由于训练样本数量众多，不可能同时将所有样本输入网络，因此批量输入用于全面训练网络。 因此输入网络的样本图像的数量被设置为一批输入30个数据，共60000个数据。并且所获得的实验结果在图中示出。

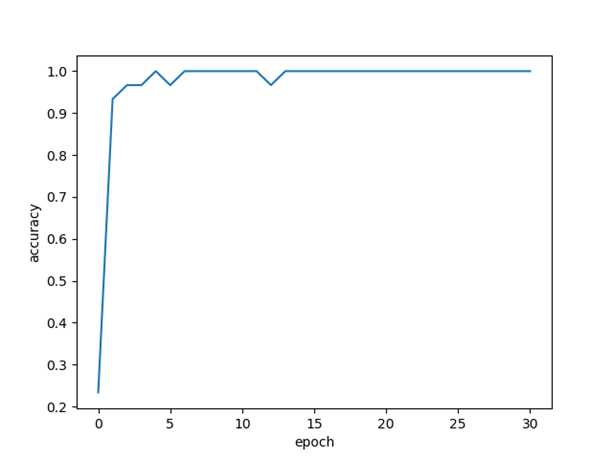


图4.2 迭代次数与准确率关系图

**Fig.4.2 Iteration number and accuracy relationship diagram**

图4.2中，横坐标表示迭代次数，纵坐标表示准确率。上图是训练样本数为60000的时候不同的迭代次数下，卷积神经网络的识别率。迭代次数，也就是遍历训练数据集的次数，如果设置的太小，就会导致训练的深度达不到要求，如果太大，训练需要较长时间，训练效率就会较低，因此我们选择迭代次数设为30。从图中可以看出，随着迭代次数的增加，识别率基本呈现增加趋势，网络逐渐收敛，当每批中30个样本输入网络时，迭代到达约15次时，识别效果已经达到99.8%。因为训练样本设置的数量比较大，所以整体的识别正确率也非常高。

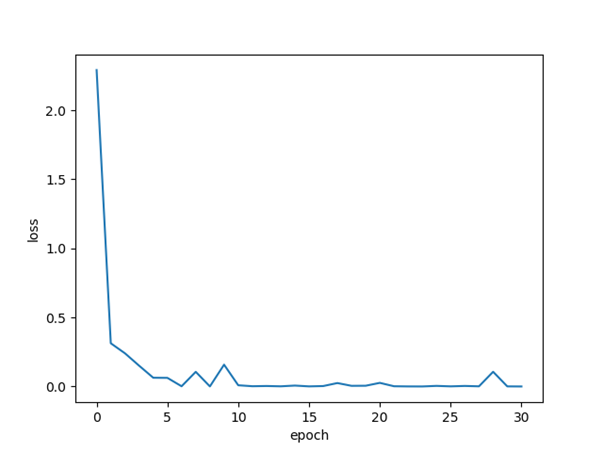


图4.3 迭代次数与误差关系图

**Fig.4.3 Iteration number and error relationship diagram**

图4.3中，横坐标表示迭代次数，纵坐标表示误差；我们可以看出，随着迭代次数越大，误差逐渐减小，最终趋向于0。当每批输入30个数据，迭代15次左右时，误差就已非常低，达到非常高的识别效率。

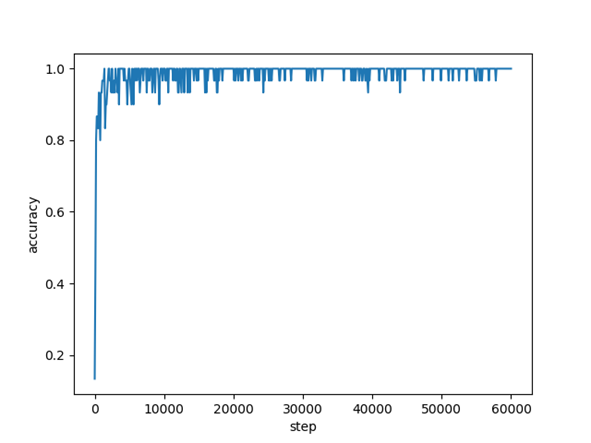


图4.4 样本基数与准确率关系图

**Fig.4.4 Sample base and accuracy relationship diagram**

图4.4中，横坐标表示样本数量，纵坐标表示识别准确率，我们可以看出，在迭代次数不变的情况下，随着样本基数的增加，识别准确率也越高，最终趋于稳定，识别率最高达99.7%。

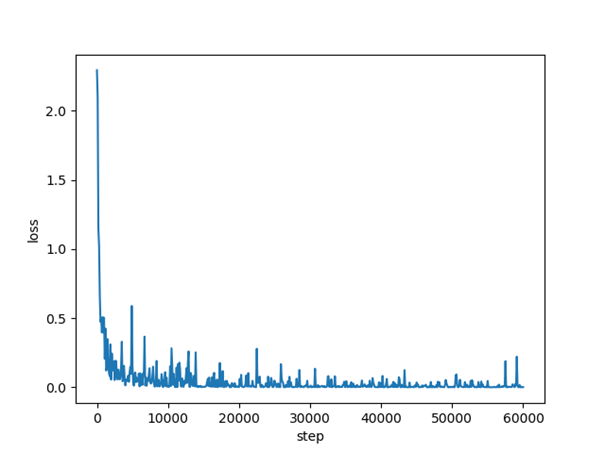


图4.5 样本基数与误差关系图

**Fig.4.5 Sample base and error relationship diagram**

图4.5中，横坐标表示样本数量，纵坐标表示误差，我们可以看出，在迭代次数不变的情况下，随着样本基数的增大，误差整体呈现减小的趋势。

在对卷积神经网络进行训练和验证后，我们获得了一种具有更高识别精度的学习模型。为了进一步测试训练模型是否具有实际适用性，以及事实准确性，使用该模型对实际数字图像进行识别测试。结果如图4.6所示（下图显示4个数字的识别结果），对10个数字进行识别，均能得到正确的识别结果。

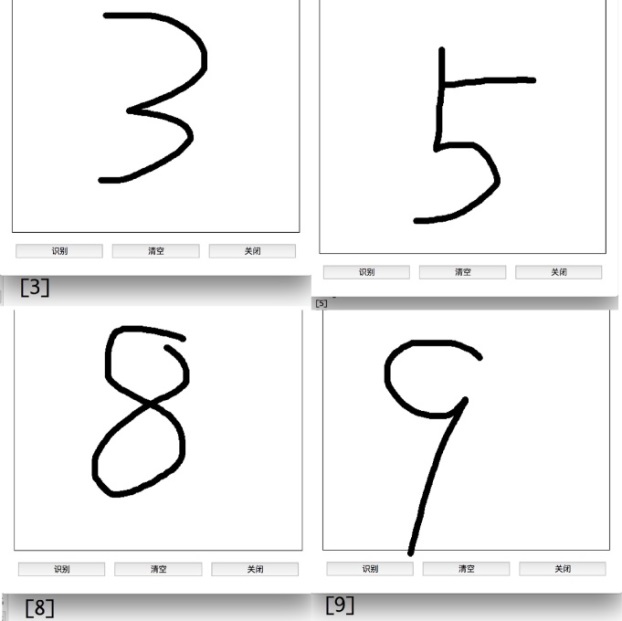


图4.6 测试结果

**Fig.4.6 Test Results**

# 第5章 总结

实验表明，使用卷积神经网络可以减少预处理工作量并具有较高的网络识别率。当训练样本数量足够大，并且迭代次数合适的时候，模型可以达到很高的识别率，本文的实验中最高识别率已经趋近于1（最高达到99.8%），误差趋近于0，可以对绝大多数的数字进行正确识别。卷积神经网络适合于应用在手写数字识别上，因为手写数字算法需要手动预处理和特征提取，卷积神经网络本身具有局部连接，权值共享，子采样等特征，可以直接从原始数字图像中提取样本的特征信息，有效地减少了作为神经网络输入的特征，缩小神经网络的规模。这种方法比较简单，并且比较容易实现，对提高数字手写体识别准确率具有非常实际的意义。本文的研究可以为以后卷积神经网络模型的设计提供思路。要想取得较高的准确率，需要非常大的样本基数，在现实生活的分类问题之中，很少有如此大的样本基数，所以在样本基数有限时，如何取得较高的识别率，是我们今后所需要解决的问题。

# 参 考 文 献

[1]吕红.基于卷积神经网络的手写数字识别系统的设计[J].智能计算机与应用用,2019,9(02):54-56+62.

[2]朱书眉.基于卷积神经网络的手写体数字识别[J].数字技术与应用,2018,36(11):44+46.

[3]高诺,胡文烨,杨玉娜.采用卷积神经网络的数字手写体识别的研究[J].齐鲁工业大学学报,2018,32(05):45-49.

[4]陈岩,李洋洋,余乐,王瑶,吴超,李阳光.基于卷积神经网络的手写体数字识别系统[J].微电子学与计算机,2018,35(02):71-74.

[5]马媛媛,史加荣.卷积神经网络及其在手写体数字识别中的应用[J].湖北工程学院学报，2017,37(06):66-72.

[6]张红,马静.基于卷积神经网络的手写数字识别算法[J].电子技术与软件工程,2017(22):176.

[7]张庆辉,万晨霞,陈众威.基于改进卷积神经网络的手写体识别研究[J].中原工学院学报,2017,28(04):79-85.

[8]王光新. 脱机手写体汉字智能识别模型与相似样本识别研究[D].合肥工业大学,2017.

[9]李斯凡,高法钦.基于卷积神经网络的手写数字识别[J].浙江理工大学学报(自然科学版), ,2017,37(03):438-443.

[10]关鑫.卷积神经网络在手写体识别的应用[J].电子技术与软件工程,2016(13):110.

[11]郭鹏. 深度卷积神经网络及其在手写体汉字识别中的应用研究[D].四川师范大学,2016.

[12]刘欣. 基于卷积神经网络的联机手写汉字识别系统[D].哈尔滨工业大学,2015.

[13]常欢. 基于卷积神经网络的孤立手写体汉字识别研究[D].安徽大学,2015.

[14] Using neural nets to recognize handwritten digits: Michael A. Nielsen,Oct 2018.

[15] 代贺，陈洪密，李志申.基于卷积神经网络的数字识别.北京:中国农业大学,2017.

[16] Neural Networks and Deep Learning: Michael A. Nielsen,2015

[17] CS231n Convolutional Neural Networks for Visual Recognition.

[18]宋晓茹,吴雪,高嵩, 等.基于深度神经网络的手写数字识别模拟研究[J].科学技术与工程,2019,19(5):193-196.

[19]高灿.基于卷积神经网络的脱机手写汉字识别系统研究[D].安徽:安徽理工大学,2017.

[20]姜新猛.基于TensorFlow的卷积神经网络的应用研究[D].湖北:华中师范大学,2017.

# 附 录A

训练模型代码：

def build\_graph():  
keep\_prob\_1=tf.placeholder(dtype=tf.float32,shape=[],

name='keep\_prob\_1')  
keep\_prob\_2=tf.placeholder(dtype=tf.float32,

shape=[], name='keep\_prob\_2')  
keep\_prob\_3=tf.placeholder(dtype=tf.float32,

shape=[], name='keep\_prob\_3')  
h\_conv1 = slim.conv2d(x\_image,32,[3,3],1,

padding='SAME', activation\_fn=tf.nn.relu)  
h\_conv2=slim.conv2d(h\_conv1,32,[3,3],1,

padding='SAME', activation\_fn=tf.nn.relu)  
h\_pool1 = slim.max\_pool2d(h\_conv2, [2, 2], [2, 2], padding='SAME')  
h\_conv3=slim.conv2d(slim.dropout(h\_pool1,keep\_prob\_1),64,[3,3],1, padding='SAME', activation\_fn=tf.nn.relu)  
h\_conv4=slim.conv2d(h\_conv3,64,[3,3],1,

padding='SAME', activation\_fn=tf.nn.relu)  
 h\_pool2 = slim.max\_pool2d(h\_conv4, [2, 2], [2, 2], padding='SAME')  
flatten = slim.flatten(h\_pool2)  
h\_fc1=slim.fully\_connected(slim.dropout(flatten,keep\_prob\_2),256, activation\_fn=tf.nn.relu)  
y\_conv=slim.fully\_connected(slim.dropout(h\_fc1,keep\_prob\_3),10,activation\_fn=None)  
predict = tf.argmax(y\_conv, 1)  
return {  
'x': x,  
'y\_': y\_,  
'keep\_prob\_1': keep\_prob\_1,  
'keep\_prob\_2': keep\_prob\_2,  
'keep\_prob\_3': keep\_prob\_3,  
'loss': cross\_entropy,  
'accuracy': accuracy,  
'train\_step': train\_step,  
'y\_conv': y\_conv,  
'predict': predict,  
}

STEP = []  
ACCURACY = []  
LOSS = []  
graph['y\_']:batch[1],graph['keep\_prob\_1']:0.25,graph['keep\_prob\_2']:0.5, graph['keep\_prob\_3']: 0.5})  
if i % 50 == 0:  
train\_accuracy=sess.run(graph['accuracy'], feed\_dict={graph['x']: batch[0], graph['y\_']: batch[1],graph['keep\_prob\_1']: 1.0, graph['keep\_prob\_2']: 1.0, graph['keep\_prob\_3']: 1.0})  
loss = sess.run(graph['loss'], feed\_dict={graph['x']: batch[0], graph['y\_']: batch[1], graph['keep\_prob\_1']: 1.0, graph['keep\_prob\_2']: 1.0, graph['keep\_prob\_3']: 1.0})  
STEP.append(i)  
ACCURACY.append(train\_accuracy)  
LOSS.append(loss)  
if i % 2000 == 0:  
saver.save(sess, './model.ckpt')  
plt.plot(STEP, ACCURACY)  
plt.plot(STEP, LOSS)  
plt.show()train()

识别代码：

from PyQt5.QtWidgets import (QWidget, QApplication, QPushButton, QLabel, QHBoxLayout, QVBoxLayout)  
from PyQt5.QtGui import (QPainter, QPen, QPixmap)  
from PyQt5.QtCore import Qt, QSize, QPoint  
from PIL import ImageGrab  
import sys  
import cv2  
import mnist  
class MyMnistWindow(QWidget):  
def \_\_init\_\_(self):   
super(MyMnistWindow, self).\_\_init\_\_()   
self.initUI()  
def initUI(self):  
self.painter = QPainter()  
self.size = QSize(500, 500)   
self.board = QPixmap(self.size)   
self.board.fill(Qt.white)   
self.lastPos = QPoint(0, 0)   
self.currentPos = QPoint(0, 0)   
self.label\_draw = QLabel('', self)   
self.label\_draw.setStyleSheet("QLabel{border:1px solid black;}")  
self.label\_draw.setAlignment(Qt.AlignCenter)

self.btn\_recognize = QPushButton("识别", self)

self.btn\_recognize.clicked.connect(self.btn\_recognize\_on\_clicked)   
self.btn\_clear = QPushButton("清空", self)

self.btn\_clear.clicked.connect(self.btn\_clear\_on\_clicked)   
self.btn\_close = QPushButton("关闭", self)   
self.btn\_close.clicked.connect(self.btn\_close\_on\_clicked)

hbox\_1 = QHBoxLayout()  
hbox\_1.addWidget(self.btn\_recognize)   
hbox\_1.addWidget(self.btn\_clear)   
hbox\_1.addWidget(self.btn\_close)   
vbox = QVBoxLayout()  
vbox.addWidget(self.label\_draw)   
vbox.addLayout(hbox\_1)   
self.setLayout(vbox)   
self.setGeometry(0, 0, 500, 500)   
self.setWindowTitle("手写数字识别")   
self.show()  
def paintEvent(self, event):   
painter = QPainter(self)   
self.painter.begin(self)   
self.painter.drawPixmap(0, 0, self.board)   
self.painter.end()  
def mousePressEvent(self, event):   
self.lastPos = event.pos()  
def mouseMoveEvent(self, event):   
self.currentPos = event.pos()  
self.painter.begin(self.board)   
self.painter.setPen(QPen(Qt.black, 10, Qt.SolidLine, Qt.MPenCapStyle))   
self.painter.drawLine(self.lastPos, self.currentPos)

self.painter.end()  
self.lastPos = self.currentPos

self.update()

def btn\_recognize\_on\_clicked(self):  
bbox = (30, 30, 460, 460)  
im = ImageGrab.grab(bbox)   
im.save('./data/test.png', 'png')   
image = cv2.imread("./data/test.png")

image = cv2.bitwise\_not(image)  
process.measure(image)  
mnist.recognition("./data/test.png")  
def btn\_clear\_on\_clicked(self):   
self.board.fill(Qt.white)   
self.update()  
self.initUI()  
self.setGeometry(0, 0, 501, 501)  
def btn\_close\_on\_clicked(self):   
self.close()

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

example = MyMnistWindow()  
example.show()  
app.exec\_()

数据集代码：  
from PIL import Image  
import tensorflow as tf  
import tensorflow.contrib.slim as slim  
import matplotlib.pyplot as plt  
global\_times = 0  
image\_prepare   
Image.open(file\_path).convert('L')

im.resize((28, 28), Image.ANTIALIAS)

list(im.getdata())# [(255 - x) \* 1.0 / 255.0 for x in tv] --- normalize pixels to 0 and 1. 0 is pure white, 1 is pure black  
im.save('./data/sample.png')

plt.imshow(im)，plt.show()

def image\_prepare(file\_path):  
im = Image.open(file\_path).convert('L')  
im = im.resize((28, 28), Image.ANTIALIAS)  
tv = list(im.getdata())  
tva = [(255 - x) \* 1.0 / 255.0 for x in tv]  
im.save('./data/sample.png')  
plt.imshow(im)  
plt.show()  
return tva  
def build\_graph():  
keep\_prob\_1=tf.placeholder(dtype=tf.float32,shape=[], name='keep\_prob\_1')  
keep\_prob\_2=tf.placeholder(dtype=tf.float32,shape=[], name='keep\_prob\_2')  
keep\_prob\_3=tf.placeholder(dtype=tf.float32,shape=[], name='keep\_prob\_3')  
x\_image = tf.reshape(x, [-1, 28, 28, 1])  
h\_conv1=slim.conv2d(x\_image,32,[3,3],1,padding='SAME', activation\_fn=tf.nn.relu)  
h\_conv2=slim.conv2d(h\_conv1,32,[3,3],1,padding='SAME', activation\_fn=tf.nn.relu)  
h\_pool1=slim.max\_pool2d(h\_conv2,[2,2],[2,2],padding='SAME')  
h\_conv3=slim.conv2d(slim.dropout(h\_pool1,keep\_prob\_1),64,[3,3],1, padding='SAME', activation\_fn=tf.nn.relu)  
h\_conv4=slim.conv2d(h\_conv3,64,[3,3],1,padding='SAME', activation\_fn=tf.nn.relu)  
h\_pool2 = slim.max\_pool2d(h\_conv4,[2,2],[2,2],padding='SAME')  
flatten = slim.flatten(h\_pool2)  
h\_fc1=slim.fully\_connected(slim.dropout(flatten,keep\_prob\_2),256, activation\_fn=tf.nn.relu)  
y\_conv=slim.fully\_connected(slim.dropout(h\_fc1,keep\_prob\_3),10, activation\_fn=None)  
predict = tf.argmax(y\_conv,1)  
{  
'x': x,

'y\_': y\_,  
'keep\_prob\_1': keep\_prob\_1,  
'keep\_prob\_2': keep\_prob\_2,  
'keep\_prob\_3': keep\_prob\_3,  
'accuracy': accuracy,  
'train\_step': train\_step,  
'y\_conv': y\_conv,  
'predict': predict,  
}  
def predict\_prepare():  
sess = tf.Session()  
graph = build\_graph()  
saver = tf.train.Saver()  
ckpt = tf.train.latest\_checkpoint('./model/')  
if ckpt:

saver.restore(sess, ckpt)  
return graph, sess  
def recognition(file\_path):  
global global\_times  
if global\_times == 0:  
global graph, sess  
graph, sess = predict\_prepare()  
image = image\_prepare(file\_path)  
predict=sess.run(graph['predict'],feed\_dict={graph['x']:[image], graph['keep\_prob\_1']:1.0,graph['keep\_prob\_2']:1.0, graph['keep\_prob\_3']: 1.0})  
global\_times = 1  
print('recognize result:')  
print(predict)  
return predict  
else:  
image = image\_prepare(file\_path)  
predict = sess.run(graph['predict'], feed\_dict={graph['x']: [image], graph['keep\_prob\_1']: 1.0, graph['keep\_prob\_2']: 1.0, graph['keep\_prob\_3']: 1.0})  
print('recognize result:')  
print(predict)  
return predict

# 致 谢

在这临近毕业之际，首先我想感谢的是我的本科毕业论文指导老师武国宁老师，本论文是在武国宁老师的严格指导和监督下完成的。老师对我论文的研究方向提出了非常具有指导性的意见；在数学理论推导和程序编写的过程中，都对我提供了非常重要的帮助；在论文的撰写过程中，给予了悉心的指导，提出了许多建设性的意见，投入了很多的心血和精力。武国宁老师不仅在学术上给予了我莫大的帮助，同时也教会了许多为人处世的道理，在此，对武老师对我的帮助和关怀表示诚挚的感谢！

时光匆匆，四年的大学生涯即将过去；人生是不断学习的过程，回望大学四年，往事一幕幕都浮现在眼前。首先，感谢老师传授给我们丰富的知识，让我们更好的应对在学习中遇到的困难，其次，感谢在大学里认识的每一位同学和朋友，你们让我收获了珍贵的友谊，尤其是与我朝夕相处的室友们，感谢你们在论文撰写过程中对我提供的帮助，也感谢你们对我的包容与爱护！