

毕业设计（论文）

基于神经网络的手写数字识别算法的实现

|  |  |
| --- | --- |
| 院 别 | 计算机与通信工程学院 |
| 专业名称 | 计算机科学与技术 |
| 班级学号 | **2133605** |
| 学生姓名 | 张浩天 |
| 指导教师 | 王翠荣 |

**2017**年**6**月**4**日

基于神经网络的手写数字识别算法的实现

摘 要

手写识别是当今模式识别的研究领域中一个极其重要的分支，其不光应用极其广泛，可以推广到的问题又遍布各大领域。作为两种信息的桥梁，手写识别的应用空间广阔，市场需求巨大。所以对手写识别的研究既有很高的理论价值又有广泛的应用价值。

近年来由于多核心计算成本的不断降低，人工神经网络成为了解决手写识别问题的热门研究方向。人工神经网络以其高容错率，高分类能力，高效利用GPU进行加速与自我学习能力等优势，使其在手写识别中鹤立鸡群。人工神经网络可以实时处理大量的手写数据，并达到前所未有的高准确率。

手写识别问题的主要难点在于其无法建立具体的数学模型，本文采用结合卷积神经网络与全连接神经网络的方法对这一问题进行解决。基于神经网络的手写识别其关键的部分在于根据效果的不同，调整神经网络中各部分的参数，从而达到最高的效果。本文采用只利用python最基本的数学计算库的方法，从底层对神经网络进行实现，从而灵活的对神经网络各部分进行调试与分析。

通过测试，本系统对手写数字的识别达到了很高的准确率。

关键词：手写数字识别，python，人工神经网络, 卷积神经网络

Implementation of handwritten digits recognition algorithm based on neural networks

Author: Zhang Haotian

Tutor: Wang Cuirong

Abstract

Handwriting recognition is an important branch of pattern recognition research. It’s not only widely used, but also can be extended to lots of areas. As the connection of real world information and digit information, the market demand is huge. Therefore, both theoretical value and wide application value of handwritten recognition are high.

In recent years, due to the continuous reduction of multi-core computing costs, artificial neural network has become a popular research dirction to solve the problem of handwriting recognition. ANN with its high fault tolerance, high classification ability, efficient use of GPU, it stands out in the field of handwritten recognition. It can handle large amounts of handwritten data in real time and achieve unprecedented high accuracy.

The main difficulty of handwriting recognition is that we can not establish a concrete mathematical model to solve it. This paper deals with this problem by combining convolution neural network and fully connected neural network. To deeply analysis the paramaters. In this paper, we only use the basic mathematical computing library of python to implement the algorithm, to achieve ultimate flexibly when analyzing.

Through the tests, the algorithm achieved a relatively high accuracy.

Keywords：Handwritten recognition, python, ANN, Convolutional Neural Network

目 录

1 绪论 1

1.1 研究背景及现状 1

1.2 研究目标及工作内容 2

1.2.1 研究目标 2

1.2.2 工作内容 2

1.3 本文组织结构 2

2 手写识别与人工神经网络概述 4

2.1 手写识别技术的意义 4

2.2 手写数字识别算法与其理论价值 4

2.3 手写识别算法的技术难点 5

2.4 人工神经网络概述 5

3 基于混合神经网络的手写识别算法的实现 6

3.1 功能介绍 6

3.2 整体架构介绍 6

3.3 详细介绍 8

3.3.1 输入数据处理部分 8

3.3.2 卷积层（C1） 9

3.3.3 池化层 13

3.3.4 全连接层 15

3.3.5 输出层 20

3.3.6 权值更新 20

4 实验分析与优化 25

4.1 实验环境 25

4.1.1 硬件环境 25

4.1.2 软件环境 25

4.2 实验数据 25

4.3 实验结果与优化分析 26

4.3.1 批的大小（batch size） 26

4.3.2 训练次数（epoch） 27

4.3.3 学习率（learning rate） 28

4.3.4 神经元数量 29

4.3.5 卷积层的作用 30

4.3.6 数据标准化 30

4.3.7 偏移量的作用 31

结 论 33

致 谢 34

参考文献 35

附 录 36

# 绪论

## 研究背景及现状

手写识别是模式识别的一种，是由二十世纪六十年代初出现的一门学科。从二十世纪九十年代后，由于GPU等计算核心速度的极大提升，模式识别的应用从以前的单纯的数学计算提升到了对图像甚至视频的范围以至于现在的计算机视觉技术。模式识别的研究方向是用一定的算法，使计算机实现学习的能力。通过大量的学习，使其得到对人类接触到的事物具备的判断与识别的能力。获得这里能力的机器在一定程度上可以取代大量重复的人工劳动如复杂的生产线。也正因为如此，模式识别在近年来成为人工智能领域极其热门的一个领域。模式识别的主要功能是分类，即判断系统所需要处理的数据属于哪一个类别。如判定邮件是否为垃圾邮件，判断音频所对应的英文单词等。模式识别在流程上主要包括信息的输入，信息的预处理，对信息特征的提取，识别与分类等环节。

在模式识别中，手写识别是十分热门也是相对来说比较成熟的一个分支。一方面来说手写识别的应用极为广泛，如今人们在手机与电子阅读设备中的阅读时间远超于纸质产品，电子设备又具有可搜索，环保，方便携带等特点，生活中高准确率的手写识别需求极高；另一方面来说，手写识别有很多推广，许多其他方面的技术如物体识别，路牌等打印文字的识别会与手写识别有技术上的共性。

从手写识别数据的来源来分，手写识别由脱机识别和联机识别两个大类，脱机识别又可以被称作光学字符识别，简称为OCR。联机识别在技术上与OCR完全不同，因为联机识别中数字写下的比划，力度，速度都是有效的信息，这样的信息在OCR中是无法得到的，这样也使联机识别的准确率相对OCR高了很多。在OCR中，由于输入的数据往往是照片，扫描仪等，输入的数据只有图片的像素数据，而且因为环境的不稳定，不同中字符的位置，亮度，纸张噪点等影响，会使OCR相对于联机识别难度更高。

随着计算机运算性能的显著提高，计算机运算成本的不断下降，模式识别已经从模型的理论建立一步步发展到了广泛实际应用阶段。借助现代性能高效的CPU，复杂的模式识别问题如语音识别与图片中的物体识别。但对于生活中一些需要复杂的实时处理的问题时，又由于当今计算机存储容量与其高复杂度运算的矛盾而进入瓶颈，这就促使了得益于大规模并行运算的人神经网络技术应用到模式识别问题中。神经网络技术将模式识别领入了另一个时代，手写识别就是其中一个得益于神经网络的领域。

手写识别是一个古老的课题，在很长时间以来，对于字符识别这样的工作是使用传统算法解决的，而即使是最好的传统识别算法，其对手写印刷体的识别也仅仅停留在了94%到96%之间，对手写体字符的识别更是停留在了研究阶段，其识别率很低以至于无法有效利用。在二十世纪九十年代之后，神经网络通过其自我学习与大规模分布式并行处理的方式，且因为其拥有非线性与强推广能力与高容错性的特点，其集齐了广泛的研究与关注。基于神经网络的手写识别算法在准确率上得到了极大的提升，并有对噪点容错性强等特点，迅速淘汰了传统的手写识别算法。现代高度优化的神经网络算法如LeNet5等，其对处理过的手写字符的准确率达到99.9%以上，对路牌等环境复杂的字符识别准确率也达到了96%以上。

## 研究目标及工作内容

### 研究目标

设计并实现神经网络算法，通过组合不同的神经网络结构并给与优化，通过使用大量手写数据对其进行训练，使神经网络可以对手写数字进行高准确率的识的别。本文仅仅使用python最基本的工具包，从最底层对神经网络进行实现，目的是对算法的实现过程进行深入的分析，同时通过调整算法各部分的细节，更好的对各个参数与部分对神经网络训练的影响进行分析，从而对算法进行彻底的优化。

### 工作内容

1. 详细介绍本神经网络的架构与层次作用。
2. 详细介绍卷积神经网络与全连接神经网络的底层算法与基本原理。
3. 通过理论与实验的分析，分析神经网络各部分对算法性能与准确率的影响，对网络进行优化，并分析此算法的优缺点。对算法中有提升空间的部分进行分析，修改与展望。
4. 对未被识别的开源手写数字数据库部分进行识别，为手写识别研究提供有效数据。

## 本文组织结构

第一章 介绍了手写识别算法的研究背景与研究的意义，介绍了手写识别算法近年来的研究方向与进程，分析了传统手写识别算法的不足之处以及基于神经网络的手写识别算法的优势，在本章的最后，简要说明了本文的研究目标与工作内容，对本文的组织结构进行了简洁的阐述。

第二章 介绍了手写识别技术的意义，对手写识别算法进行了简要的介绍并对其理论价值进行了讨论，着重分析了手写识别算法中的一些技术难点。又对本文解决手写识别问题所基于的人工神经网络进行了概述，描述了人工神经网络基于学习数据不同的分类与的优缺点。

第三章 详细的介绍本算法的实现过程与理论支撑，对算法的各个部分进行了详尽的介绍与解释，并对算法实现中的难点进行了深入的讨论。

第四章 通过使用MINST数据库，对算法进行训练，测试与分析。通过调整算法的参数，深入讨论算法每一部分的作用并对算法进行全面的优化。

第五章 对本论文的内容进行了总结，分析和展望。首先对本论文进行了深入的总结，接下来对本算法的缺点进行分析，并对算法进一步的提升做出规划，最后对机器学习下一步研究的方向进行了展望。

# 手写识别与人工神经网络概述

## 手写识别技术的意义

手写识别在技术上属于模式识别的中最典型的应用之一。手写识别的主要目标是利用计算机，自动的将含有手写识别字符的图片转换为计算机中的字符数据，手写识别在技术上分为光学字符识别（OCR）与联机手写识别（主要是手写输入法），其由于输入信息不同，对应的算法与优化完全不同，本文的算法主要针对光学字符的识别，联机识别算法暂不讨论。

## 手写数字识别算法与其理论价值

手写数字识别是手写识别算法的一个最典型的分支，其研究的理论价值很大：

1. 阿拉伯数字是全球通用的文字，也是世界上被使用次数最多的十个字符。因为这时全球通用的字符，所以对手写数字的研究是超越国界的，其研究与文化背景无关，全球的研究成果都可以进行共享，对手写数字识别算法也研讨也十分方便与统一。
2. 手写数字识别算法的应用范围极广，在很多领域，手写数字识别的自动化可以大大提高处理的效率，降低人工成本。如邮政编码的识别，报税表单与存款单等。这些工作往往包含着大量的重复简单的人工劳动，强度大且效率低。而手写识别的出现可以大大减少人力的成本，提高工作的效率，也使无纸化办公的未来更进一步。
3. 相对于对其他种类字符的识别，手写数字算法所需要的识别的种类只有十个，远少于字符识别（200个左右）甚至汉字识别（10000个以上），其识别率相对较高，所需要的算法工程量也较低，因为这个优点，对于新的模型与理论，往往使用手写数字识别的方法来验证算法的准确性，对其分析以及做深入的研究。因为比较简单，对于基于机器学习与神经网络的手写识别算法来说，其所需要的训练数据与训练时间也相对较少，新开发的机器学习与模式识别算法都是用手写识别数据进行理论有效性检验，然后才推广到更复杂模型如路牌识别与字符识别。典型的例子有Lecun的卷积神经网络LeNet以及[Vladimir N. Vapnik](https://en.wikipedia.org/wiki/Vladimir_N._Vapnik)发明的支持向量机（SVM）。
4. 手写数字识别算法的推广很简单。比如对英文字母的识别与对路牌与车牌号的识别，其算法与手写数字识别大同小异。在现代的研究中，许多新算法就是对手写英文字符与手写数字同时进行识别的。

## 手写识别算法的技术难点

手写数字识别看似简单，只有十种类别且笔画简单，但在实际识别过程中，一些算法的实验表明，即使有几万中类别，印刷体汉字的识别准确率是高于手写数字识别的。联机手写识别的准确率也是一直高于手写数字的OCR的。其主要原因是数字与数字之间比划相差很小，数字的写法相对简单，同一个数字的写法相差很多。虽然数字是世界通用的文字，但是在数字的字体方面，是存在很强的地域性的，不同国家对一些数字的书写习惯不同，如英美数字的“7”要比亚洲人的习惯多一比等。

经过多年的探索，国内与国外针对手写数字识别的算法研究上已经几近成熟，但依然存在下面两大问题

1. 在手写数字识别的精度方面，其精度要更高。手写识别无法通过上下文等方法进行错误验证，而在有些情况，如支票等与经济，财会相关的应用中，任何错误都是不允许发生的。因此，在手写数字识别的研究中，大多都是在向提高识别准确率的方向而努力。
2. 由于数字识别往往涉及到大量的数据，一张数字表格往往就有几千个数字需要识别，所以识别的速度同样是数字识别的要求之一，然而精度与速度往往是相互矛盾的，这就使识别算法需要权衡二者。

## 人工神经网络概述

人工神经网络旨在通过模仿人脑中神经网络的基本结构，将人脑模型数学化并简化而成的数学模型。从生理学看，人脑的神经网络是由大量的，有这复杂连接的神经元组成的，人的学习过程其实是对人脑中神经元间关联强弱的自适应性调整。人脑的神经元在处理不同问题的方面是有不同结构形态的，所以人工神经网络也有这不同的形态，如递归神经网络，卷积神经网络，全连接神经网络等。神经网络的最小组成单元是神经元，虽然种类繁多，但基本结构几乎相同。

和人脑一样，人工神经网络必须经过学习才能准确的工作。通过学习来提高自身性能是神经网络的一个重要特点。一般情况下，其改善自身性能的方法是通过提高内部神经元之间的连接强度（权值）达到的，人工神经网络的学习方式，可以以接收到的信息种类来分成下面三种：

1. 监督学习（Supervised Learning）

监督学习的直观理解其实是模仿，监督学习的学习来源是样本数据，而监督学习就是通过让人工神经网络对样本数据的读取，不断调整自己内部的权值，不断的让自己在输入样本数据时，输出对应的样本结果，由于在学习的过程中，样本数据中含有样本输入与输出，所以这种方法叫做监督学习。监督学习的优点是速度快，结果容易预测。但是监督学习的缺点也有很多，由于监督学习中神经网络其实是单纯的模仿，所以在样本情况比较复杂的情况下，神经网络容易忘掉很久之前的样本信息，造成网络只对训练后期的数据比较敏感。还有在物体识别等算法中，在生活中并没有大量的数据供神经网络去学习，这时就需要大量的人工标注，其高成本也是监督学习的缺点之一。

1. 非监督学习（Unsupervised Learning）

非监督学习不存在处理好的训练样本，其训练数据是单纯的统计数据，非监督学习的程序通过寻找输入数据的规律来调节自身的参数甚至结构，与监督学习不同的是，非监督学习的目标是更好的表示统计数据内部的结构与特征，训练的过程中不断提取新的特征，在应用到模式识别中时，非监督学习通过自身对特征的理解将输入数据分成很多类，训练完成的程序可以讲未出现过的数据进行归类，这是一种完全无人参与的自学习功能。由于无需对数据的人工处理，非监督学习的成本要低于监督学习，也因此在近年来受到更高的关注。

1. 强化学习（Reinforcement Learning）

强化学习介于监督学习与非监督学习的之间，强化学习中对系统的输出结果没有正确与错误的判断，而是给出一个综合评判的奖惩信息，神经网络通过奖惩之间的权衡来改变自己内部的权值与结构，从而得到一个最佳的平衡点。

# 基于混合神经网络的手写识别算法的实现

## 功能介绍

本算法旨在通过用利用神经网络的优势，高效的对手写数字进行识别，即将手写在纸上或设备上的数字的图片信息高正确率地转换为纸上或设备上的数字。

## 整体架构介绍

为了充分发挥各个神经网络的优点，使其既有卷积神经网络高效，速度快，对噪点的容错性高的优点，又有全联接的前导神经网络准确性高的特点，本算法使用将卷积神经网络与前导神经网络抽样后全联接的方法来高效的完成对手写数字图像的识别。



本神经网络的主体分为输入数据处理，神经网络运算和输出数据处理。其中输入数据处理是将原始数据处理成神经网络容易识别的且经过标准化的数据集。神经网络(在运算上分为训练阶段与识别阶段)。在训练阶段神经网络通过所给的训练数据与答案修改与更新自己的参数，从而使自己通过输入数据所计算出来的答案越来越接近真实正确的答案。识别阶段即使用通过大量数据训练过的神经网络，让测试数据通过神经网络进行运算从而得出预测的结果。在这里神经网络部分分为输入层（IN）卷积层（C1），池化层（S1），隐藏层（H1），输出层（OUT）。输出处理则将大量的原始，为优化结果而标准化的数据处理成可以利用，统计的直观数据。在这里输出处理只有一部分。

## 详细介绍

下面将对本算法的各个层次的原理，设计与实现进行详尽的介绍。

### 输入数据处理部分

本部分旨在对原始数据的处理，其中原始数据有训练数据（32000张PNG图片，其中每个图片的分辨率为28\*28，图片为灰度图片）；训练数据的标签，即这32000张图片所显示的具体数字的答案；测试数据（10000张图片，图片格式与训练数据相同）以及测试数据的标签。

对训练与测试数据的处理：

训练数据原始为图片，先用python的Image库将整个图片加载成矩阵，由于输入的图片均为灰阶图片，所以根据Image库的算法，输出的矩阵为输入图片对应像素的灰阶值。

然后对这个矩阵进行标准化，标准化的意思是，通过线性或者非线性转换，将矩阵的最大值-最小值范围控制在0-1之间且让最大值和最小值尽量分别接近0与1。对于本文针对的灰阶矩阵，其最大值为255，最小值为0，所以我们可以通过把每个数字除以255的方式将其标准化。

用此方法，我们成功将输入矩阵进行了标准化。对于标准化的作用，本文将在后面介绍线性神经元与其梯度下降算法的部分详细介绍。

对训练与测试标签数据的处理：

训练数据与测试数据通过神经网络的计算会得到其预测的结果的特征，在这里这个特征的值就决定了本神经网络对输入图像的识别结果。区别于数字本身的连续性，对于神经网络的分类任务来说，其特征应该是互不相干的，例如：1与9之间的相关性应和1与2之间的相关性完全一样。所以需要用编码的方式排除数字本身的相关性，本算法使用的编码方式是独热编码（One-Hot Encoding）

独热编码：

独热编码直观来说就是用总状态数的比特数，其中一个比特为0其他比特为1的方法来表示状态的一种码制。

在本文的算法中，总状态数即为10。下面是编码表：

|  |  |  |
| --- | --- | --- |
| 原始数据 | 二进制 | 独热编码 |
| 0 | 0000 | 1000000000 |
| 1 | 0001 | 0100000000 |
| 2 | 0010 | 0010000000 |
| 3 | 0011 | 0001000000 |
| 4 | 0100 | 0000100000 |
| 5 | 0101 | 0000010000 |
| 6 | 0110 | 0000001000 |
| 7 | 0111 | 0000000100 |
| 8 | 1000 | 0000000010 |
| 9 | 1001 | 0000000001 |

表格 1-a

经过编码后的标签由于其向量积均为0，所以互不相关。且在总状态数较少的情况下独热编码更简单高效，另一方面又增加了状态的空间，使算法更加准确。

### 卷积层（C1）

卷积神经网络(Convolutional Neural Network, CNN)：

卷积神经网络是深度学习与计算机视觉中极其重要的网络结构之一，许多及其成功的模型都是基于CNN的，特别是基于ImageNet等的各种图像物体识别算法。卷积神经网络算法相对于传统的图像处理算法有很多优点，其中一个就是其巧妙的避免了复杂耗时的预处理工作（如人工特征的提取），原始图像数据经过CNN可以直接进入下一步的运算，不需要人工的参与。

如前文所说，在图像识别的过程中，图像往往会被看成一个或多个的二维矩阵，本文用到的MINST图像库中，图像库中每一个图片就可以看成是28 \* 28 的二维矩阵，因为它是黑白图片，只有一个颜色通道。如果是彩色图片就会有RGB三个颜色通道，就会被表示为三个二维矩阵。传统的神经网络算法使用的是全链接的方式，这种连接方式是将每一个颜色通道的每一个像素都与下一层的神经元·所连接，这样会使参数的数量指数增长，导致参数的数量巨大，使整个神经网络的训练及其耗时甚至时间数量级巨大，无法训练。而卷积神经网络通过局部链接，权值共享两个主要方法，在几乎不消耗准确率的情况下大幅减少变量数，从而避免这一困难的发生。

直观的看，卷积层的参数可以看作是一系列的可以训练和学习的过滤器（卷积核）。在前向学习的过程中，每次我们将图片的每一个小部分通过一个过滤器，点乘后再组成新的二维数据，每个过滤器计算成的二维数据再组成三维数据。我可以理解成每个过滤器只关心一部分图片的特征，再综合图片的各个特征进行下一步的训练。

局部连接：

局部连接是卷积神经网络独特的原因之一，上文提到在图片中，使用全连接的神经网络在工程上是不可行的，而局部链接的意思就是卷积神经网络的神经元只会与上层的神经元的局部区域进行连接。可以把上层数据想象成一张画，画的上面有一个滑动的窗口，每次运算时只有窗口内的数据会与下层进行关联。这就引出了卷积神经网络的一个重要的变量：窗口大小，一般窗口的长与宽是相等的，窗口的长\*宽叫做感受野（receptive field），在整个运算中，这个窗口会逐格滑动，一步步近似覆盖图片的每一个小区域。



在本算法中，我们的MINST图片库输入的图片数据大小为[1\*28\*28]，我们设定感受野（窗口）大小为[5\*5]，为了提取不同的特征，我们有6个不同的卷积核，所以经卷积后有6张24\*24的图片数据。

权值共享：

权值共享是卷积神经网络中很巧妙的处理方方式，它可以使卷积神经网络的训练与识别时的计算复杂度和参数的个数下降非常非常多。在本算法中，我们共有6个卷积核，每个卷积核大小为5 \* 5，所以经过特征图（feature map）的长与宽都为（28-5+1），共有6个特征图，所以每个图片的总变量数位6\*24\*24 = 3456个神经元，每个窗口由于数据的连接，有5\*5=25个权重，所以共有25\*3456= 86400个参数，如果每张图片的计算量都是这么大的话，会变得非常非常慢，也就失去了它的意义。

因为每个卷积的神经元可以看成是一个过滤器，那么我们可以大胆的假设，这个神经元用于连接每个数据窗口的权重都是固定的，不会因为窗口的位置而改变。这意味着，对于每一个卷积神经原来说，不管其对应的数据窗口在图片的哪个位置，其权重都是同一组数。那就代表，我们的卷积层中，我们只需要神经元个数\*卷积核的维度= 6\*6\*5=180个权重。

如果每个神经元在不同位置对应的权重是固定的，那么同一个卷积神经元进行计算卷积积分的整个过程就可以看成是一组固定的权重和不同位置的数据窗口做内积的过程，在数学上刚好对应着『卷积』操作，这也是卷积神经网络名字的来源。

另外，因为每个神经元的权重都是固定的，它们都可以看做一个恒定的过滤器，比如我们的6个卷积神经元作为过滤器后可视化之后的样子如图：



需要说明的一点是，参数共享并不是一个在每个场景都适用的一种策略，在一些特定的场合中，如人脸识别中，由于一般照片中人的面部都集中在画面的中央，所以在人脸识别的过程中，我们不能把照片上的每个位置的窗口数据的作用都视作等同的。在这种情况下，我们希望数据窗口滑过中心区域的时候其权重和其他边缘区域是不同的。对于这种情况我们有一种特殊的层对应这种功能，叫做**局部连接层，**在本算法中局部连接层没有被用到，就暂不赘述。

具体实现：

本算法使用的是批梯度下降法（mini-batch learning）进行训练，即将多个输入数据同时通过神经网络进行计算，计算后将更新的数据进行叠加来更新，这样既会利用python的系统优化提高计算效率，又会避免低效重复的更新，具体将在本章后面优化的部分进行详细描述。

首先初始化经卷积后的多个特征图，由于有每次训练是由多个输入数据组成的批，每批有6个特征图，每个特征图是24（图片宽度-窗口宽度+1）\*24的矩阵，我们先初始化特征图组为批大小\*特征图数\*特征图高\*特征图宽的四维矩阵，然后用每个卷积核分别用窗口划过每张图片，每滑动一次求出相应的积并填入特征图相应的位置中。具体代码如下。

conved\_input = np.zeros((train\_data\_batch.shape[0], num\_of\_fmap, image\_size - conv\_core + 1, image\_size - conv\_core + 1), dtype = np.float)  
**for** i **in** range(0, train\_data\_batch.shape[0]):  
 **for** j **in** range(0, num\_of\_fmap):  
 **for** k **in** range(0, image\_size - conv\_core + 1):  
 **for** l **in** range(0, image\_size - conv\_core + 1):  
 conved\_input[i][j][k][l] = sum(sum(two\_d\_input[i][k:k + conv\_core, l:l + conv\_core] \* (in\_to\_conv\_weights[j]. reshape(conv\_core, conv\_core))))

由于本程序是基于CPU，所以每个求积需要单独运算，本算法经过GPU加速后可以根据GPU的核心数（如NVIDIA GeForce GTX1050Ti有640个核心）同时运算，可大幅提高速度。

### 池化层

一般来说，在卷积神经网络中，池化层是在连续的卷积层中间或者卷积到全连接层中间的层。它的作用也非常简单，就是通过抽样的方法逐步地减少和压缩数据和参数的量，在一定程度上也会减小过拟合的现象，即过度的符合训练数据而使容错性降低。池化层做的操作也非常简单，就是将原数据上的每个小区域压缩成一个值(小区域的区域最大值（MAX）或者平均值（AVERAGE）)，最常见的池化层的设定是，将原数据切成由2\*2的小块组成的格式，在每块里面取最大值作为输出，这样我们就自然而然减少了75%的数据量，抽样的方式根据分数据特点的不同而不同。

在最大值和平均值的池化方式外，我们也可以设定其他的抽样方式，如L2范数池化。本算法中对池化层和它的操作的直观理解的示意图为：



如图为本算法中池化层的一个直观的示意，即对上一层（卷积层）处理后得出的6\*24\*24的数据（一般来说，特征图的数量被称为厚度）的每一个切片（1\*24\*24的部分）做了一个下采样。下图为次采样操作的实际操作。



具体实现：

在本算法中，通过上文我们知道，由上层卷积神经网络计算出了维度为批大小\*特征图数\*特征图高\*特征图宽的四维矩阵，我们对每个特征度进行取样单位为2\*2，的最大值取样，所以取样后的特征图的维度为批大小\*特征图数\*（特征图高／2）\*（特征图宽／2），对于每个取样后的特征图的位置，其值为对应的四个特征图的值的最大值。为了方便后面更新权制，需要把最大值所在位置的坐标也同时记录下来，具体代码如下：

conved\_input\_maxloc = np.zeros( (train\_data\_batch.shape[0], num\_of\_fmap, image\_size - conv\_core + 1, image\_size - conv\_core + 1),dtype=np.float)

sampled\_input = np.zeros((train\_data\_batch.shape[0], num\_of\_fmap, int((image\_size - conv\_core + 1) / 2), int((image\_size - conv\_core + 1) / 2)), dtype=np.float)  
**for** i **in** range(0, train\_data\_batch.shape[0]):  
 **for** j **in** range(0, num\_of\_fmap):  
 **for** k **in** range(0, int(conved\_input.shape[2] / 2)):  
 **for** l **in** range(0, int(conved\_input.shape[3] / 2)):  
 sampled\_input[i][j][k][l] = np.amax(conved\_input[i][j][2 \* k:2 \* k + 2, 2 \* l:2 \* l + 2])  
 agmx = np.argmax(conved\_input[i][j][2 \* k:2 \* k + 2, 2 \* l:2 \* l + 2])  
 conved\_input\_maxloc[i][j][2 \* k + int(agmx / 2)][2 \* l + agmx % 2] = 1

### 全连接层

全连接神经网络是神经网络中最基本，最标准的形式。全连接的意思是，神经网络中一层的所有神经元都和上一层的所有神经元之间进行关联，相对于卷积神经网络，全连接神经网络的矩阵运算比较和直接。现代的很多卷积神经网络结构，其末层或末几层都会采用全连接去学习更多的信息。要理解全连接层的工作原理，我们必须了解组成全连接层与所有神经网络的基本单元——神经元的工作原理。

在神经网络中有很多种类的神经元，在全连接神经网络中最常见的神经元叫做现行神经元，又叫做线性过滤器，如图888所示线性神经元的有一系列的输入，它的参数是这些输入的权值，输出是它输入值的加权和。



神经元的偏差方程是用来计算神经元输出值与正确值差别的方程，对于不同的情况，有不同的计算偏差的方式，在数字预测方面经常会使用方差模型，在坐标方面有欧式距离和曼哈顿距离，在文字处理方面经常使用交叉熵作为偏差方程。神经元的目标是通过更新自身的参数（对每个输入的权值）来尽量减少输出的偏差，使自身更佳准确，在这里，偏差是所有残差的平方和。

一个神经元的例子：

假设你每天在食堂吃午餐，每天的午餐由不同数量的鱼，菜，米饭组成，每次点完餐阿姨只会告诉你总价格，几天之后，你就可以推算出每种食物的单价了。在这里假设你就是一个线性神经元：

每个种类的价格组成了线性神经元的权值：

开始的时候我们会随机猜每种菜品的价格，然后每过一天，通过我们点的每种菜的数量和阿姨告诉我们的总价格，一点点更新我们的猜想，使其更加接近正确的值。

我们首先预测每种食物的价格都为50元（这里是初始值，可以为任何非零数，如果是零会导致更新时的变量为0，无法更新，在卷积神经网络中有时候会用到这个方法保持权值的同步）

在这里输入值为:

线性神经元权值

残差

根据学习规则（delta-rule）权值的变化量 其中为学习率，即学习的速度。

假定学习率，则鱼，菜，饭的变化分别为

更新后的权值为 进一步接近正确值，当有新的数据，循环的次数越来越多之后，预测的价格会越来越接近于真实的价值。这个方法的优化后的标准化方法叫做梯度下降，用偏导的方法找到使误差下降最快的变化量。

梯度下降：

梯度下降又被称为最速下降法，是一种迭代的最优化算法，函数在一个位置的梯度可以表示为这个函数在这一点的时候它最“陡峭”的方向，那么如果一个点向它对应方向的梯度增加很小的数量，那么这个更新后的点的函数值一定会下降。而梯度下降法正是利用了这样特点，把每个神经元的参数当作这个上点的“坐标”而这个函数就是神经元权值与整个神经网络的偏差的关系，那么利用梯度下降的方法，经过很多很多次的迭代，不断的更新神经元的权值，整个申请网络的偏差就会不断的下降，最终达到一个最小值（如图888）。



全连接神经网络：

全连接神经网络又多层神经元组成，第n层的神经元的输入与第n-1层的神经元的输出进行相连，同时第n层神经元的输出与第n+1层神经元的输入相连接，这样一层一层传递数据，最终传递出结果，再通过输出结果一层一层的向前更新变量（back-propagation）。通过多层神经网络的数据传递与权值更新，其能通过训练找到数据与其标签的隐藏含义，而相对于单层神经元组成的网络，其能更好的识别更加复杂和隐含的模型。多层神经网络在结构上分为输入层（input layer），隐藏层（hidden layer），和输出层（output layer），其中隐藏层可以有多个。



如图8888所示，在多层神经网络中，由于是全连接的结构，第n+1层与第n层之间的连接数为他们的神经元数的积，在这里表示位于第n层神经元的权值，表示从上一层第i个神经元连向下一层第j个神经元的权值。

在计算的过程中，第n层的第j个神经元的输出值等于第n-1层每个神经元的输出值与其对第n层第i个神经元加权总和

全连接神经网络的计算极其相似于矩阵的乘法，现假设每个批次只有一组输入数据，输入数据为一横行的数据：

设矩阵W[i][j]表示输入数据到第一层隐藏层的权值，纵坐标表示输入神经元i，横坐标表示输出神经元j，W[i][j]表示

将输入矩阵与权制矩阵点乘，通过点乘法则得到矩阵：

正好为隐藏层的输出值，这个数组可以在与下一层的权值进行点乘，如此就可以依次迭代快速得出所有层的值。

在批梯度下降法中，可以将一个批次中每行数据并成一个批次大小\*数据维度的矩阵，这样可以通过矩阵与矩阵的乘法快速高效的点乘出下一层的值：

具体实现：

在本算法中，由上层卷积神经网络经池化后的数据维度为批大小 \* 特征图数 \*（特征图高／2）\*（特征图宽／2），在这里为了更好的利用矩阵的特性进行全连接计算，将池化后的数据进行处理，将维度变化为二维，纵向为批大小，将每批的数据转换为1行，这样输入到全连接神经网络的数据就会变成批大小 \* （特征图数 \*（特征图高／2）2，综合速度与准确性的考虑，本算法中我们设定隐藏层大小为200 个线性神经元，输入数据经过与权值的矩阵点乘后，为了使训练更加高效，又将点乘后的数据加入了可训练的偏差值后得到了隐藏层。由于输出结构为之前独热编码后的0-9的数字，所以有10个输出的神经元。用同样的方法由隐藏层计算出输出层后，由于本程序的宗旨是预测数字，所以比较合适的偏差是计算机对于其认为图像为每个数字概率。在这里使用将输出层映射到sigmoid函数的方法，对于sigmoid函数的作用，将在优化的部分详尽解释。

图

具体代码：

oned\_fnn\_in = sampled\_input.reshape(sampled\_input.shape[0], sampled\_input.shape[1] \* sampled\_input.shape[2] \* sampled\_input.shape[3])

hid\_state = np.dot(train\_batch, in\_to\_hid\_weights)  
hid\_state = hid\_state + hid\_bias  
out\_state = np.dot(hid\_state, hid\_to\_out\_weights)  
out\_state = out\_state + out\_bias  
out\_state = 1 / (1 + np.exp(-out\_state))

### 输出层

输出层的作用很简单，将上层输出的神经网络求出的每个数概率中，选出最大的一项输出结果。

为了测试，也可以输出其他的结果，比如概率最高的N个数或保证一定概率的置信数组。

具体代码：prediction = np.argmax(out\_state, axis=1)

### 权值更新

前文提到，神经元是通过梯度下降的方法来使自身变得准确的，而神经网络是通过其当中每一个神经元的梯度下降来提高整个神经网络的准确性的。相对于一个一个为神经元求其对整个神经网络残差的偏导数，利用偏导数的链式法则（）神经网络利用反向传播的方法来按层批量更新权值。

反向传播：

反向传播的基本思想是梯度下降，对于复杂的神经网络，我们无法通过解方程的方式得到每个权值的最佳值，但是我们可以通过求梯度的方式来得到每个神经元的每个权值在变化量足够小的时候其相对于最佳值的方向，对于复杂的神经网络，由于上层神经元的输出会与多个下层神经元进行连接，所以对于这样的神经元来说，其变化量其实是其所有连接的神经元对结果的影响和的加权值。相对于单独计算，反向传播大大提高了运算效率。

全连接神经网络的反向传播：

之所以叫反向传播，是因为这个方法是按照从输入到输出的顺序一层一层向前推进的，这样巧妙的利用了链式法则的优势。



在这里神经元j是输出层的神经元，由图8888可知，上一层神经元的偏导数实际上是由下一层与其连接的神经元的偏导数叠加而成。

我们假设此网络的偏差方程为方差：

那么输出层神经元j相对于偏差E的偏导数

对于输出层，上文提到，输出层神经元经常会用到像sigmoid或softmax的函数进行优化，所以这里神经元j的输入可能并不与输出成线性相关的关系：

在这里举例使用本算法的sigmoid神经元作为输出神经元：

则下一层的神经元i，因为对于下一层神经元j，神经元相当于对其做了权值Wij的‘贡献’，因此其相对于偏差E的偏导数可以理解为为上一层所有神经元的输入值偏导数的加权和：

对于每层神经元来说，以权值的角度看，可以理解成下层神经元是输入其中的权值的“以输入值为权值的加权和”那么权值的偏导数实际上就是其指向的神经元的偏导数于其对应输入值的积：

按此方法可以依次向前推进，依次求出全连接神经网络中输入层到隐藏层权值的偏导值。如果全连接层前面有其他的神经网络层，也需要求出其输入层的偏导值以进行下一步运算。

卷积神经网络的反向传播：

卷积神经网络的反向传播思想于全连接神经网络大同小异，其宗旨都是找到神经元或权值当把它们当成下一层输入的时候的权值，再用这个权值与其连接下一层的偏导数进行加权乘积。

在本算法卷积神经网络中，由于在卷积之后有最大值池化的优化，所以在求偏导数时需要将池化中筛选掉的输入值同时筛选掉。筛选掉后对于每个卷积核的权值，找到与其进行加权乘积后的输入值，再进行点乘后得出其偏微分。

实现：

在本算法中，输出层为sigmoid神经元，其维度为批大小\*10，结构为：

根据前面推导的sigmoid神经元的偏导公式，输出层的偏导矩阵为：

通过输出层神经元偏导矩阵与隐藏层输出值矩阵的转置进行点乘可以得到隐藏层到输出层权值的偏导，其维度为隐层神经元个数\*输出神经元个数，与隐层神经元到输出神经元的权值矩阵相同：

由于偏移量没有权值，所以起偏导就等于其连接的神经元的偏导和：

通过输出层偏导矩阵与隐藏层到输出层权值转置进行点乘，得到隐藏层神经元的偏导矩阵，矩阵的维度为批大小\*隐层神经元个数，与隐层神经元矩阵相同：

通过同样的方法依次求出输入层到输出层权值偏导矩阵，输入层偏导矩阵，隐藏层偏移量偏导：

再利用全连接神经网络输入层的偏导来求卷积核的权值的偏导数：

本算法中，先利用之前记录下来的最大值的坐标与卷积神经网络的输入数据，合成一个维度与输入数据相同，只保留最大值池化（将池化没有选中的设为零）的数组。再直接用卷积核上的想要求出卷积的那个参数所有“划过”的点乘上其对应的输入向量，最后加和：

再将已经求出的所有的偏微分乘上学习率，用其对应的权值去减去求出的变化量，久完成了权值的更新。

具体代码：

*# bprop:err*err = out\_state - train\_label\_batch  
CE = (err \* err / 2).sum(axis=1)  
d\_Out = out\_state \* (1 - out\_state) \* err  
d\_hid\_to\_out = np.dot(hid\_state.transpose(), d\_Out)  
d\_hid = np.dot(d\_Out, hid\_to\_out\_weights.transpose())  
d\_in\_to\_hid = np.dot(train\_batch.transpose(), d\_hid)  
d\_in = np.dot(d\_hid, in\_to\_hid\_weights.transpose())  
  
*# update weights*del\_hid\_to\_out = -1 \* alpha \* d\_hid\_to\_out / batch\_size  
del\_in\_to\_hid = -1 \* alpha \* d\_in\_to\_hid / batch\_size  
hid\_to\_out\_weights += del\_hid\_to\_out  
in\_to\_hid\_weights += del\_in\_to\_hid  
  
hid\_bias += -1 \* alpha \* d\_hid  
out\_bias += -1 \* alpha \* d\_Out

d\_in = d\_in.reshape(d\_in.shape[0], num\_of\_fmap, int((28 - conv\_core + 1) / 2),  
 int((28 - conv\_core + 1) / 2))  
expd\_d\_in = np.zeros((d\_in.shape[0], d\_in.shape[1], d\_in.shape[2] \* 2, d\_in.shape[3] \* 2), dtype=np.float)  
*# print(d\_in.shape)*kr = [[1, 1], [1, 1]]  
**for** i **in** range(0, train\_data\_batch.shape[0]):  
 **for** j **in** range(0, num\_of\_fmap):  
 expd\_d\_in[i][j] = np.kron(d\_in[i][j], kr)  
*# print(expd\_d\_in[0][0])*expd\_d\_in = expd\_d\_in \* conved\_input\_maxloc  
*# print(expd\_d\_in[0][5])  
# print(train\_data\_batch)*d\_cov\_w = np.zeros((d\_in.shape[1], conv\_core, conv\_core), dtype=np.float)  
**for** i **in** range(0, train\_data\_batch.shape[0]):  
 **for** j **in** range(0, num\_of\_fmap):  
 **for** k **in** range(0, conv\_core):  
 **for** l **in** range(0, conv\_core):  
 d\_cov\_w[j][k][l] += sum(sum(  
 expd\_d\_in[i][j] \* two\_d\_input[i][k:k + expd\_d\_in.shape[2], l:l + expd\_d\_in.shape[3]]))in\_to\_conv\_weights += -alpha \* (d\_cov\_w.reshape(d\_cov\_w.shape[0], d\_cov\_w.shape[1] \* d\_cov\_w.shape[2]))

# 实验分析与优化

## 实验环境

本算法的实验基于python的交互式笔记本Jupyter，Jupyter是一个交互式的开源编程，学习与测试平台，其最大的一个优点是对变量的保存，在神经网络算法中，有些训练后的权值数据是通过几十分钟的训练得到的，如果每次调整都要重新训练的话实验效率将很低，在jupyter的交互式测试环境下，在不损失变量的值的情况下分小段调试代码，直观高效的完成对实验数据的测试。

### 硬件环境

MacBook Pro 2016

Intel Core i7 3.3GHz (2 Cores) CPU

16 GB 2133 MHz LPDDR3 Memory

Intel Iris Graphics 550 1536 MB

### 软件环境

操作系统：macOS Sierra 10.12.6

python版本：python3.6.0

python相关库版本：

ipython 6.0.0

jupyter-console 5.1.0

numpy 1.12.1

pandas 0.20.0

matplotlib 2.0.1

IDE: pyCharm Community Edition 2016.3.2

## 实验数据

本算法的训练与测试数据均来自于MINST 数据库，MINST是一个巨大的手写数字数据库，广泛的在机器学习与手写识别领域被使用，MINST数据库来源是对美国国家标准与技术局的数据的抽样与修正，其手写数字的来源是美国众多高中学生的笔记，扫描后经过切分，标准化于抗锯齿化，并被转化为灰度图像。

MINST数据库由60000张训练数据图与10000张测试数据图组成。其中训练数据图是有经过人工标注的答案的。在机器学习与图形识别领域，很多论文都以其对MINST数据库的识别准确率为算法准确性的标准，其中利用卷积神经网络，有些神经网络将识别错误率降低到了0.23%以下，以支持向量机为核心的算法，其识别错误率也被优化到了0.6%一下，已经达到近乎人类的识别准确率。

为了更好的进行测试，本文的测试数据与训练数据均来自MINST的测试数据，其中训练数据有32000个，测试数据有10000个，为了测试神经网络对新数据的兼容性能，测试数据不出现在训练阶段，训练数据也不会参与测试。

## 实验结果与优化分析

在本算法中，对算法运算时间，准确率，效率，容错性等特性有影响的参数有很多，如卷积核大小，池化层的池化程度，学习率，有无偏移量，隐藏层的神经元数量，隐藏层的层数等。通过调整各个参数的数量，来对算法进行优化，下面通过实验与分析了解各个参数的作用与在本算法中的最佳选择：

### 批的大小（batch size）



如图88888在线上学习（每次只学习一个图像）中，由于对于复杂的神经网络，其残差函数比较负载，在梯度下降时，可能出现每次下降的方向不一样，出现zigzag的现象，通过批学习的方法，用同一组权值同时训练多组输入数据，将每组数据的变化量加和后更新权值，用这种方法可以通过抵消掉其他方向的重合部分，从而提高更新权值的效率。以下是在其它参数固定的情况下，批大小分别为1，10，100，1000时训练准确率与训练进度的图像和训练准确率与训练时间的图像：



由图像我们可以看出，在合理的批的大小范围内，在训练进度相同时，神经网络的准确率几乎是相同的，此时批的大小对训练精度的影响几乎可以忽略不计；但是在训练时间上可以出，在合理的批大小范围内，随着批大小的增大，训练的速度会明显提升。儿训练后的最终精度并不会受到明显的影响。但是当批的大小过大时（batch size=1000），由于每次更新的权值变化量积累过多，会造成过量更新，使权值无法接近最优点，造成训练失败，对于某些情况，可以用减少学习率的方法来解决这个问题。

### 训练次数（epoch）

在机器学习的过程中，由于数据量有限且步长（学习率）较小，特别是当批尺寸比较大的时候，由于权值更新的次数有限，权值在一次训练过并没有达到最优解，这时候可以使用多次训练的方法使其充分学习，批尺寸越大的时候其达到较高精度所需要的训练次数就越多。在训练达到算法精度的极限后，算法的预测准确性不会再因为扩大训练次数而提高：



由图像可以看出，当批大小为1和10时，神经网络的准确率在第3次训练后不再增加，当批大小为200时，神经网络的准确率在第5次训练后不再增加。

### 学习率（learning rate）

因为每次权值的更新变化量都是其偏导与学习率的乘积，所以学习率在一定程度上直接决定着神经网络学习的速度。对于过小的学习率，神经网络对训练数据会不能有效利用，神经网络的学习速度会很慢。而对于过大的学习率，由于每次更新的权值过大，会出现“跳过”最优解的情况，而离最优解更远之后会造成其所在权值偏导增大，求出的权值变化量更大，进一步远离最优解，最后造成学习失败。



在实验中，我们分别使用学习率为0.001，0.01，0.1，0.5四个不同的学习率对神经网络进行训练，下面是学习进度与神经网络识别准确率的关系：

**

由图像可以看出，对于过小的学习率，学习速度将非常慢，算法的效率将很低，当学t率很高时，会使神经网络在接近最优解时出现“跳过”最优解的情况，使神经网络的准确率停留在一个较低的位置，当学习率很高时，神经网络的变化量过大，导致和权值随机变化的效果，造成精度无法提升，学习失败。现代算法中常使用动态学习率的方法，既保留了高学习率学习速度快的优势，又保留了低学习率最终精度高的优势，由于比较复杂，在本算法中暂不考虑。

### 神经元数量

这里的神经元数量指的是全连接层隐藏神经元的数量，隐藏层神经元的数量决定着神经网络的灵活性，隐藏层神经元数量越多，神经网络越复杂，神经网络越能学习出复杂的模型来识别输入的数据。但过多的隐藏层神经元有两个缺点：一个是由于全连接神经网络的权值是输出层和输出层神经元数量与隐藏层神经元数量的乘积，过多的隐藏层神经元会让计算量指数增加，使训练速度变慢；第二个是过多的隐藏层神经元会造成过度拟合（overfitting）即神经网络过于灵活，使其训练的结果过于贴近训练数据而没有容错，其训练的结果仅仅高度拟合训练的数据，而无法对与训练数据稍有不同的测试数据进行准确的识别：

正常与过度拟合图：

在实验中，我们分别对隐藏层神经元数量为1，10，2000的全连接神经网络隐藏层神经元数量的神经网络进行相同训练数据的训练，下面分别是学习进度与神经网络对训练数据识别准确率的关系和学习进度与神经网络对测试数据识别准确率的关系：



对于只有一个隐藏层神经元的网络，由于其灵活性太差，不能完成对10个数的分类识别，造成识别失败，对于有2000个神经元的网络，由于其神经网络隐藏层层过于庞大，造成其对训练数据的过拟合，使其对原数据的识别准确率极高而对未在原数据出现过的训练数据识别率较低。由于本数据集训练数据与测试数据较相似，所以在测试中均出现测试数据正确率与训练数据正确率“同步”上升与下降的现象。值得提到的是，拥有2000个隐藏层神经元的神经网络训练总时间为2166.36秒，为100个隐藏神经元的神经网络（159.93秒）的近20倍。综上，神经网络不能通过盲目增加隐藏神经元的方式来提高识别的精度与效率，反而会适得其反。

### 卷积层的作用

卷积神经网络会在大幅减少神经网络参数个数的同时，由于其对图像进行了预处理，使神经网络对图像中如数字位置，图像噪点等影响的适用性增强，同时减少神经网络的运算量，加快学习的效率。在实验中，我们别用有卷积神经网络的与无卷积神经网络的网络分别对同一组数据进行学习，得出学习进度与准确率的关系：



可以看出，通过卷积神经网络的处理，神经网络对图像的训练效率有所增加，训练达到相同准确率所需要的训练数据有所减小，因此卷积神经网络在图像数据比较匮乏时也可以发挥很大的作用。另外，由于经过卷积和池化，图像无可避免的有信息的损失，所以最终的识别准确性不如全连接神经网络。由于卷积神经网络的运算速度很依赖显卡的加速，由于运行环境的原因无法达成，所以本文暂不在运行时间方面进行二者的比较。

### 数据标准化

在训练过程中，由于训练数据可能有很多数据组成，比如房屋价格的预测算法，可能训练数据中同时有房屋的历史价格（百万单位）与房屋的窗户数（一到几十的单位），这时如果不加处理直接将数据放入神经网络进行训练，神经网络进行梯度下降时的残差函数会在x轴和y轴上的差别很大，假设x轴上的数量级远大于y轴上的数量级，那么每次进行梯度下降时，其x轴方向的梯度必然远小于y轴上的梯度，使梯度下降的方向十分接近y轴的方向，这样会出现在y轴方向来回摆动而在x轴方向移动很慢的现象，使学习的速度急剧下降，效率极低。

为了解决这一问题，可以训练之前，对训练数据与测试数据使用同样的方法，将数据中每种数字都等比例或使用合适的非线性方程转化为(-1,-1)范围的数量级。统一后的数量级的在残差函数上因为数量级相同，会在x与y轴上较接近，避免了沿某一坐标轴来回摆动的现象发生。在本算法中由于数据（像素灰度值）本身在同一数量级中，标准化对算法的提升效果不明显，就暂不进行试验性分析。

### 偏移量的作用

偏移量在神经网络中的作用与标准化相似，如果训练中有相关性较强的两组数据，比如房屋总面积与房屋的可用面积，或图片相邻两个像素的灰度值。当出现这种情况时，残差函数会在y=x或y=-x方向被拉的很长，与上一节遇到的情况相同，这时残差函数的梯度下降的方向会十分接近y=x或y=-x并在这个方向，使权值在这个方向来回摆动而不能接近最低点。通过使用偏移量，将相近的权值同时加上或剪掉一个常数的方法，将其符号调整为不同的，这样会使被拉长的残差函数恢复在各个方向与原点距离数量级相近的状态。



# 结 论

在手写识别领域中，神经网络性能优越，在参数上全面优化的神经网络与正确的图像预处理相结合，其识别准确率可以达到令人满意的程度。

本文在开始时详尽介绍了手写数字识别的难点与原理，在设计的过程中使用了结合卷积神经网络与全连接神经网络的混合网络对开源手写数据库MINST进行训练学习与测试，用python基本库实现了手写数字的识别程序。在实验与测试中，结果表明，利用神经网络对手写数字进行识别方法可行。在优化中，对数据的预处理方面如输入图像的二值化与细化等降噪算法的增加，在本算法中有带进一步研究。

另外，针对神经网络的优化，本文重点对神经网络的学习率，隐藏层神经元个数等进行调整和优化，利用Jupyter进行调试，通过改变各种参数与神经网络结构，对全连接神经网路各部分的作用以及各种神经网络结构进行比较。再测试结果中表明，通过对神经网络各部分的优化，本算法对手写数字拥有较强的识别能力。本文的神经网络对手写数字的识别率达到92%。

对未来的展望：本程序所面对的是一个十分复杂的问题，本算法也有很多可以提高的部分。对于学习率上学习速度与精度的矛盾，可以采用动态学习率进行学习。对于网络结构，可以利用多层卷积神经网络进行进一步的优化，全连接层部分可以使用多层隐藏层进行深度学习，训练出灵活性更高，更复杂的模型。在速度方面，在神经网络中的矩阵运算里，可以利用GPU的多核心明显加快运算速度，实现实时识别。

# 致 谢

时光荏苒，四年的大学本科即将结束。在这四年的生活中，我不断的学习，不断获取新的知识。在这四年中，我感觉很充实和愉快，这离不开老师、亲人、朋友和同学给我的关爱和鼓舞。值此之际，谨向所有关心过我、帮助过我和指导过我的人表示由衷的感谢！

首先，感谢王翠荣导师，在她的悉心指导下毕业论文终于划上了完美的句号。导师渊博的知识，严谨求实的科学态度，踏实进取、勇于开拓的治学精神以及精益求精的工作作风使我终生受益。在本次论文的写作过程中，从论文选题到搜集资料，从写稿到反复修改，王翠荣老师严格把关，循循善诱，给予了我各种指导。在学位论文完成之际，谨向导师表达我最崇高的敬意和深深的谢意。

感谢东北大学秦皇岛分校计算机与通信工程学院各位老师和同学四年来在学习、工作和生活方面给予我的帮助。经过四年的大学生活，我在学习和思想上都受益匪浅，这个除了自身的努力外，与各位老师、同学和朋友的鼓励是分不开的。在此，我表示衷心的感谢。

感谢各位老师百忙之中对本文的审阅和赐教!

参考文献

1. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778
2. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, "Backpropagation applied to handwritten zip code recognition", Neural Computation, vol. 1, no. 4, pp. 541-551, 1989.
3. [Y. Lecun](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Y.%20Lecun.QT.&newsearch=true); [L. Bottou](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.L.%20Bottou.QT.&newsearch=true) ; [Y. Bengio](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Y.%20Bengio.QT.&newsearch=true) ; [P. Haffner](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.P.%20Haffner.QT.&newsearch=true), “Gradient Based Learning Applied to Document Recognition”, Proc. of the IEEE, November 1998
4. Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, pages 253–256. IEEE, 2010.
5. A. Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge 2010. www.image- net.org/challenges. 2010.
6. 赵志宏,杨绍普,马增强. 基于卷积神经网络LeNet-5的车牌字符识别研究[J]. 系统仿真学报,2010,(03):638-641.
7. Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, et al. Hand-written digit recognition with a back-propagation network. In Advances in neural information processing systems, 1990.
8. C.Tappert, C, Suen, and T.Wakahara, “The state of the are in on-line handwriting recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 12, pp. 787-808, 1990.
9. M. Gilloux and M. Leroux, “Recognition of cursive script amounts on postal checks,” in Proc. Europ. Conf. Postal Tech- nol., Nantes, France, June 1993, pp. 705–712.
10. I. Guyon, M. Schenkel, and J. Denker, “Overview and syn- thesis of on-line cursive handwriting recognition techniques,” in Handbook on Optical Character Recognition and Document Image Analysis, P . S. P . Wang and H. Bunke, Eds. New York: World Scientific, 1996.

# 附 录

附录A CONVOLUTIONAL  NEURAL  NETWORKS FOR ISOLATED  CHARACTER  RECOGNITION

 The ability of multilayer networks trained with gradient descent to learn complex, high-dimensional, nonlinear mappings from large collections of examples makes them obvious candidates for image recognition tasks. In the traditional model of pattern recognition, a hand-designed feature extractor gathers relevant information from the input and eliminates irrelevant variabilities. A trainable classifier then categorizes the resulting feature vectors into classes. In this scheme, standard, fully connected multilayer networks can be used as classifiers. A potentially more interesting scheme is to rely as much as possible on learning in the feature extractor itself. In the case of character recognition, a network could be fed with almost raw inputs (e.g., size-normalized images). While this can be done with an ordinary fully connected feedforward network with some success for tasks such as character recognition, there are problems. First, typical images are large, often with several hundred variables (pixels). A fully connected first layer with, e.g., one hundred hidden units in the first layer would already contain several tens of thousands of weights. Such a large number of parameters increases the capacity of the system and therefore requires a larger training set. In addition, the memory requirement to store so many weights may rule out certain hardware implementations. But the main deficiency of unstructured nets for image or speech applications is that they have no built-in invariance with respect to translations or local distortions of the inputs. Before being sent to the fixed-size input layer of an NN, character images, 2282 PROCEEDINGS OF THE IEEE, VOL. 86, NO. 11, NOVEMBER 1998 Fig. 2. Architecture of LeNet-5, a convolutional NN, here used for digits recognition. Each plane is a feature map, i.e., a set of units whose weights are constrained to be identical.  or other 2-D or one-dimensional (1-D) signals, must be approximately size normalized and centered in the input field. Unfortunately, no such preprocessing can be perfect: handwriting is often normalized at the word level, which can cause size, slant, and position variations for individual characters. This, combined with variability in writing style, will cause variations in the position of distinctive features in input objects. In principle, a fully connected network of sufficient size could learn to produce outputs that are invariant with respect to such variations. However, learning such a task would probably result in multiple units with similar weight patterns positioned at various locations in the input so as to detect distinctive features wherever they appear on the input. Learning these weight configurations requires a very large number of training instances to cover the space of possible variations. In convolutional networks, as described below, shift invariance is automatically obtained by forcing the replication of weight configurations across space. Secondly, a deficiency of fully connected architectures is that the topology of the input is entirely ignored. The input variables can be presented in any (fixed) order without affecting the outcome of the training. On the contrary, images (or time-frequency representations of speech) have a strong 2-D local structure: variables (or pixels) that are spatially or temporally nearby are highly correlated. Local correlations are the reasons for the well-known advantages of extracting and combining local features before recognizing spatial or temporal objects, because configurations of neighboring variables can be classified into a small number of categories (e.g., edges, corners, etc.). Convolutional networks force the extraction of local features by restricting the receptive fields of hidden units to be local.

 A. Convolutional Networks

Convolutional networks combine three architectural ideas to ensure some degree of shift, scale, and distortion invariance:

1) local receptive fields; 2) shared weights (or weight replication); and 3) spatial or temporal subsampling. A typical convolutional network for recognizing characters, dubbed LeNet-5, is shown in Fig. 2. The input plane receives images of characters that are approximately size normalized and centered. Each unit in a layer receives inputs from a set of units located in a small neighborhood in the previous layer. The idea of connecting units to local receptive fields on the input goes back to the perceptron in the early 1960’s, and it was almost simultaneous with Hubel and Wiesel’s discovery of locally sensitive, orientationselective neurons in the cat’s visual system [30]. Local connections have been used many times in neural models of visual learning [2], [18], [31]–[34]. With local receptive fields neurons can extract elementary visual features such as oriented edges, endpoints, corners (or similar features in other signals such as speech spectrograms). These features are then combined by the subsequent layers in order to detect higher order features. As stated earlier, distortions or shifts of the input can cause the position of salient features to vary. In addition, elementary feature detectors that are useful on one part of the image are likely to be useful across the entire image. This knowledge can be applied by forcing a set of units, whose receptive fields are located at different places on the image, to have identical weight vectors [15], [32], [34]. Units in a layer are organized in planes within which all the units share the same set of weights. The set of outputs of the units in such a plane is called a feature map. Units in a feature map are all constrained to perform the same operation on different parts of the image. A complete convolutional layer is composed of several feature maps (with different weight vectors), so that multiple features can be extracted at each location. A concrete example of this is the first layer of LeNet-5 shown in Fig. 2. Units in the first hidden layer of LeNet-5 are organized in six planes, each of which is a feature map. A unit in a feature map has 25 inputs connected to a 5 5 area in the input, called the receptive field of the unit. Each unit has 25 inputs and therefore 25 trainable coefficients plus a trainable bias. The receptive fields of contiguous units in a feature map are centered on corresponding contiguous units in the previous layer. Therefore, receptive fields of neighboring units overlap. For example, in the first hidden layer of LeNet-5, the receptive fields of horizontally contiguous units overlap by four columns and five rows. As stated earlier, all the units in a feature map share the same set of 25 weights and the same bias, so they detect the same feature at all possible locations on the input. The other feature maps in the layer use different sets of weights and biases, thereby extracting different types of local features. In the LECUN et al.: GRADIENT-BASED LEARNING APPLIED TO DOCUMENT RECOGNITION 2283 case of LeNet-5, at each input location six different types of features are extracted by six units in identical locations in the six feature maps. A sequential implementation of a feature map would scan the input image with a single unit that has a local receptive field and store the states of this unit at corresponding locations in the feature map. This operation is equivalent to a convolution, followed by an additive bias and squashing function, hence the name convolutional network. The kernel of the convolution is the set of connection weights used by the units in the feature map. An interesting property of convolutional layers is that if the input image is shifted, the feature map output will be shifted by the same amount, but it will be left unchanged otherwise. This property is at the basis of the robustness of convolutional networks to shifts and distortions of the input. Once a feature has been detected, its exact location becomes less important. Only its approximate position relative to other features is relevant. For example, once we know that the input image contains the endpoint of a roughly horizontal segment in the upper left area, a corner in the upper right area, and the endpoint of a roughly vertical segment in the lower portion of the image, we can tell the input image is a seven. Not only is the precise position of each of those features irrelevant for identifying the pattern, it is potentially harmful because the positions are likely to vary for different instances of the character. A simple way to reduce the precision with which the position of distinctive features are encoded in a feature map is to reduce the spatial resolution of the feature map. This can be achieved with a so-called subsampling layer, which performs a local averaging and a subsampling, thereby reducing the resolution of the feature map and reducing the sensitivity of the output to shifts and distortions. The second hidden layer of LeNet-5 is a subsampling layer. This layer comprises six feature maps, one for each feature map in the previous layer. The receptive field of each unit is a 2 2 area in the previous layer’s corresponding feature map. Each unit computes the average of its four inputs, multiplies it by a trainable coefficient, adds a trainable bias, and passes the result through a sigmoid function. Contiguous units have nonoverlapping contiguous receptive fields. Consequently, a subsampling layer feature map has half the number of rows and columns as the feature maps in the previous layer. The trainable coefficient and bias control the effect of the sigmoid nonlinearity. If the coefficient is small, then the unit operates in a quasi-linear mode, and the subsampling layer merely blurs the input. If the coefficient is large, subsampling units can be seen as performing a “noisy OR” or a “noisy AND” function depending on the value of the bias. Successive layers of convolutions and subsampling are typically alternated resulting in a “bipyramid”: at each layer, the number of feature maps is increased as the spatial resolution is decreased. Each unit in the third hidden layer in Fig. 2 may have input connections from several feature maps in the previous layer. The convolution/subsampling combination, inspired by Hubel and Wiesel’s notions of “simple” and “complex” cells, was implemented in Fukushima’s Neocognitron [32], though no globally supervised learning procedure such as back propagation was available then. A large degree of invariance to geometric transformations of the input can be achieved with this progressive reduction of spatial resolution compensated by a progressive increase of the richness of the representation (the number of feature maps). Since all the weights are learned with back propagation, convolutional networks can be seen as synthesizing their own feature extractor. The weight sharing technique has the interesting side effect of reducing the number of free parameters, thereby reducing the “capacity” of the machine and reducing the gap between test error and training error [34]. The network in Fig. 2 contains 345 308 connections, but only 60 000 trainable free parameters because of the weight sharing. Fixed-size convolutional networks have been applied to many applications, among other handwriting recognition [35], [36], machine-printed character recognition [37], online handwriting recognition [38], and face recognition [39]. Fixed-size convolutional networks that share weights along a single temporal dimension are known as time-delay NN’s (TDNN’s). TDNN’s have been used in phoneme recognition (without subsampling) [40], [41], spoken word recognition (with subsampling) [42], [43], online recognition of isolated handwritten characters [44], and signature verification [45].

 B. LeNet-5

This section describes in more detail the architecture of LeNet-5, the Convolutional NN used in the experiments. LeNet-5 comprises seven layers, not counting the input, all of which contain trainable parameters (weights). The input is a 32 32 pixel image. This is significantly larger than the largest character in the database (at most 20 20 pixels centered in a 28 28 field). The reason is that it is desirable that potential distinctive features such as stroke endpoints or corner can appear in the center of the receptive field of the highest level feature detectors. In LeNet-5, the set of centers of the receptive fields of the last convolutional layer (C3, see below) form a 20 20 area in the center of the 32 32 input. The values of the input pixels are normalized so that the background level (white) corresponds to a value of and the foreground (black) corresponds to 1.175. This makes the mean input roughly zero and the variance roughly one, which accelerates learning [46]. In the following, convolutional layers are labeled Cx, subsampling layers are labeled Sx, and fully connected layers are labeled Fx, where x is the layer index. Layer C1 is a convolutional layer with six feature maps. Each unit in each feature map is connected to a 5 5 neighborhood in the input. The size of the feature maps is 28 28 which prevents connection from the input from falling off the boundary. C1 contains 156 trainable parameters and 122 304 connections. Layer S2 is a subsampling layer with six feature maps of size 14 14. Each unit in each feature map is connected to a 2 2 neighborhood in the corresponding feature map in C1. The four inputs to a unit in S2 are added, then multiplied by 2284 PROCEEDINGS OF THE IEEE, VOL. 86, NO. 11, NOVEMBER 1998 Table 1 Each Column Indicates Which Feature Map in S2 Are Combined by the Units in a Particular Feature Map of C3 a trainable coefficient, and then added to a trainable bias. The result is passed through a sigmoidal function. The 2 2 receptive fields are nonoverlapping, therefore feature maps in S2 have half the number of rows and column as feature maps in C1. Layer S2 has 12 trainable parameters and 5880 connections. Layer C3 is a convolutional layer with 16 feature maps. Each unit in each feature map is connected to several 5 5 neighborhoods at identical locations in a subset of S2’s feature maps. Table 1 shows the set of S2 feature maps combined by each C3 feature map. Why not connect every S2 feature map to every C3 feature map? The reason is twofold. First, a noncomplete connection scheme keeps the number of connections within reasonable bounds. More importantly, it forces a break of symmetry in the network. Different feature maps are forced to extract different (hopefully complementary) features because they get different sets of inputs. The rationale behind the connection scheme in Table 1 is the following. The first six C3 feature maps take inputs from every contiguous subsets of three feature maps in S2. The next six take input from every contiguous subset of four. The next three take input from some discontinuous subsets of four. Finally, the last one takes input from all S2 feature maps. Layer C3 has 1516 trainable parameters and 156 000 connections. Layer S4 is a subsampling layer with 16 feature maps of size 5 5. Each unit in each feature map is connected to a 2 2 neighborhood in the corresponding feature map in C3, in a similar way as C1 and S2. Layer S4 has 32 trainable parameters and 2000 connections. Layer C5 is a convolutional layer with 120 feature maps. Each unit is connected to a 5 5 neighborhood on all 16 of S4’s feature maps. Here, because the size of S4 is also 5 5, the size of C5’s feature maps is 1 1; this amounts to a full connection between S4 and C5. C5 is labeled as a convolutional layer, instead of a fully connected layer, because if LeNet-5 input were made bigger with everything else kept constant, the feature map dimension would be larger than 1 1. This process of dynamically increasing the size of a convolutional network is described in Section VII. Layer C5 has 48 120 trainable connections. Layer F6 contains 84 units (the reason for this number comes from the design of the output layer, explained below) and is fully connected to C5. It has 10 164 trainable parameters. As in classical NN’s, units in layers up to F6 compute a dot product between their input vector and their weight vector, to which a bias is added. This weighted sum, denoted for unit is then passed through a sigmoid squashing function to produce the state of unit denoted by (5) The squashing function is a scaled hyperbolic tangent (6) where is the amplitude of the function and determines its slope at the origin. The function is odd, with horizontal asymptotes at and The constant is chosen to be 1.7159. The rationale for this choice of a squashing function is given in Appendix A. Finally, the output layer is composed of Euclidean RBF units, one for each class, with 84 inputs each. The outputs of each RBF unit is computed as follows: (7) In other words, each output RBF unit computes the Euclidean distance between its input vector and its parameter vector. The further away the input is from the parameter vector, the larger the RBF output. The output of a particular RBF can be interpreted as a penalty term measuring the fit between the input pattern and a model of the class associated with the RBF. In probabilistic terms, the RBF output can be interpreted as the unnormalized negative log-likelihood of a Gaussian distribution in the space of configurations of layer F6. Given an input pattern, the loss function should be designed so as to get the configuration of F6 as close as possible to the parameter vector of the RBF that corresponds to the pattern’s desired class. The parameter vectors of these units were chosen by hand and kept fixed (at least initially). The components of those parameters vectors were set to 1 or 1. While they could have been chosen at random with equal probabilities for 1 and 1, or even chosen to form an error correcting code as suggested by [47], they were instead designed to represent a stylized image of the corresponding character class drawn on a 7 12 bitmap (hence the number 84). Such a representation is not particularly useful for recognizing isolated digits, but it is quite useful for recognizing strings of characters taken from the fully printable ASCII set. The rationale is that characters that are similar, and therefore confusable, such as uppercase “O,” lowercase “o,” and zero, lowercase “l” digit one, and square brackets and uppercase “I,” will have similar output codes. This is particularly useful if the system is combined with a linguistic postprocessor that can correct such confusions. Because the codes for confusable classes are similar, the output of the corresponding RBF’s for an ambiguous character will be similar, and the postprocessor will be able to pick the appropriate interpretation. Fig. 3 gives the output codes for the full ASCII set. Another reason for using such distributed codes, rather than the more common “1 of N” code (also called place code or grandmother cell code) for the outputs is that nondistributed codes tend to behave badly when the number LECUN et al.: GRADIENT-BASED LEARNING APPLIED TO DOCUMENT RECOGNITION 2285 Fig. 3. Initial parameters of the output RBF’s for recognizing the full ASCII set. of classes is larger than a few dozen. The reason is that output units in a nondistributed code must be off most of the time. This is quite difficult to achieve with sigmoid units. Yet another reason is that the classifiers are often used not only to recognize characters, but also to reject noncharacters. RBF’s with distributed codes are more appropriate for that purpose because unlike sigmoids, they are activated within a well-circumscribed region of their input space, outside of which nontypical patterns are more likely to fall. The parameter vectors of the RBF’s play the role of target vectors for layer F6. It is worth pointing out that the components of those vectors are 1 or 1, which is well within the range of the sigmoid of F6, and therefore prevents those sigmoids from