

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 5 月 30 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，对神经网络进行调整，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。

在实验与测试中，本文对搭建好的神经网络进行分析与测试，通过调整神经网络中学习率等参数进行试验与比较。并结合数学推导，详细分析了各个参数对系统运算速度，稳定性，最终准确率等的影响，并逐步优化，最终使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，人工神经网络, 卷积神经网络，python

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 direction 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 parameters. 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, we designed and analyzed the neural network we built, tested and compared different algorithms by adjusting parameters such as learning rate. Combined with mathematical deduction, we analyzed different parameters and their influence to system's benchmarks and optimized them, resulting in decent recognize accuracy.

Keywords: handwriting recognition, artificial neural network, convolutional neural network, python

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

1.4 论文的组织结构 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

2.5 本章小节 6

3 基于混合神经网络的手写识别算法的实现 7

3.1 功能介绍 7

3.2 整体架构介绍 7

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 9

3.3.3 池化层 12

3.3.4 全连接层 14

3.3.5 输出层 18

3.3.6 权值更新 18

3.4 本章小结 23

4 实验分析与优化 24

4.1.1 硬件环境 24

4.1.2 软件环境 24

4.2 实验数据 24

4.3 实验结果与优化分析 25

4.3.1 批的大小 25

4.3.2 训练次数 26

4.3.3 学习率 27

4.3.4 神经元数量 28

4.3.5 卷积层的作用 29

4.3.6数据标准化 29

4.3.7 偏移量的作用 30

4.4 本章小结 31

结 论 32

致 谢 33

参考文献 34

附 录 35

1 绪论

1.1 研究背景及现状

手写识别是模式识别的一种，出现于二十世纪六十年代初。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计算机中一个热门的领域。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组成。脱机识别又可以被称为光学字符识别，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。

在计算机速度的高速提升，计算机的广泛普及下，其运算能力已经可以将曾经模式识别的理论带到现实中。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，这就促使了并行且高效利用计算性能的神经网络的推广。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。

1.2 研究目标及工作内容

1.2.1 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。

1.2.2 工作内容

本课题主要包含了以下几点工作内容。

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。

1.3 论文的组织结构

本文主要对基于神经网络的手写数字识别算法进行分析，设计，实现，测试与优化。全文文共分为五章，每章的主要内容如下：

第一章 介绍了手写识别算法的研究背景与研究的意义，介绍了其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。

2 手写识别与人工神经网络概述

2.1 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。

2.2 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。首先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。

其次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。

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" \o "Vladimir N. Vapnik)发明的支持向量机（SVM）。

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。

2.3 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。

2.4 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。

2）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将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。

3）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。

2.5 本章小节

本章详细对本文相关的技术进行了介绍。分别介绍了手写识别与人工神经网络的意义与原理，又着重讨论了手写识别算法的技术难点，了解了手写识别算法研究的重要意义了展望。

3 基于混合神经网络的手写识别算法的实现

3.1 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。

3.2 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。



图3.1 本神经网络的整体结构

如图3.1，本神经网络在结构上主要分为输入数据处理部分，神经网络运算和输出数据处理部分。其中输入数据处理的目标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。识别阶段中，测试数据通过大量数据训练过的神运算从而得出预测的结果。在这里神经网络部分由五个部分组成，分为输入层（IN）卷积层（C1），池化层（S1），隐藏层（H1），输出层（OUT）。输出处理部分的作用是将则将大量的原始输出数据进行处理，将在程序中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。

3.3 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。

3.3.1 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。

1、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，见式（3.1）。

（3.1）

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。

（3.2）

利用这个方法，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。

2、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，在程序中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。数字本身是连续的，但是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：在图像中1与9的差别并不比1与2要大很多，所以1与9之间的相关性应和1与2之间的相关性不应该被数字本身的关系所影响。在程序中就需要使用编码的方式来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，见式3.3与3.4。

（3.3）

（3.4）

在本文的算法中，总状态数即为10。编码表如表3.1：

表 3.1 独热编码表

|  |  |  |
| --- | --- | --- |
| 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 |

经过编码后的标签由于其向量积均为0，所以每两个数之间都互不相关。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的大小，使算法更加准确。

3.3.2 卷积层

卷积神经网络(Convolutional Neural Network, CNN)是模式识别与计算机视觉中很典型并影响深远的网络结构之一。现在几乎所有识别准确率领先的模型都是基于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，这种连接方式是将三个颜色的每一个像素都与下一层的神经元进行连接，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。

从另一方面来看，卷积层的参数可以看作是一组权值的窗口， 但是和全连接神经网络相同，这个窗口是可以训练与自我更新的（卷积核）。在学习的过程中，我们将图片的每个一小部分都与权值相乘，得出二维的结果，由于每个窗口都有一个二维的结果，其结果会被拼接成三维的结果。我可以理解成每个权值窗口只关心一部分图片的特征，就像我们可能只关注图片的边缘而不是图片内部的亮度，再综合图片的各个特征进行下一步的训练。

1、局部连接：

局部连接是卷积神经网络性能超群的原因之一，上文提到在分辨率很高的图片中，使用全连接的神经网络在工程上就是无法实现的，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），如图3.2，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。



图3.2 局部连接示意图

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。

2、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。如果用这种方法计算，那么在本网络的卷积层中，总权值个数为卷积核权值个数与神经元个数的积，只有6\*6\*5=180个。

如果卷机核在不同位置的权值是固定的，那么实际上可以把卷积核看成一个多色的窗口，卷积预算可以看成这个窗口一点点划过图片并求点乘积的过程（见式3.5）在数学中这个过程和几何中的卷积机器的相似，虽然没有直接用到卷积的性质，但是由于操作相同，就恰好将这种神经网络叫做卷积神经网络了。

（3.5）

但是参数共享的性质并不适用于所有的图像识别问题，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。相反，我们想要对人脸出现频率高的区域采用和边缘不同的权值。对于这种情况在卷积神经网络中常常使用一个额外的神经网络层来管理，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，每个特征图是边长为24（图片宽度-窗口宽度+1）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。

3.3.3 池化层

一般来说，在卷积神经网络中，池化层通常在两个卷积层中间或者在卷积层后面。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。池化层的算法很简单，其根本是将每个小像素块提取成一个值，根据需求与特点的不同，这个值可能是小区域的最大值也可能是平均值。一种很常见的方法是，首先将原数据切成由2\*2的小块，再将每个小块的最大值提取出来，组成新的数据矩阵（见式3.5），这样我们不光增大了容错性，还减少了四分之三的数据量。

（3.5）

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可以直观的理解为对上一层（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。图3.3为池化的实际操作示意：



图3.3 池化的操作示意图

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，核心代码如下：

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

3.3.4 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。在现代的神经网络系统中，经过卷积将数据减少到一定程度后，就开始使用全连接神经网络处理。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。



图3.4 神经元的结构

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线性神经元，又叫做线性过滤器，如图3.4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（见式3.6）。

(3.6)

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 本算法中，偏差是所有残差的平方和。

用一个例子来解释神经元权值的更新，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，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 。权值进一步接近正确值，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，用函数偏微分的方法找到使误差下降最快的下降方向，一步一步减小误差。

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3.5）。

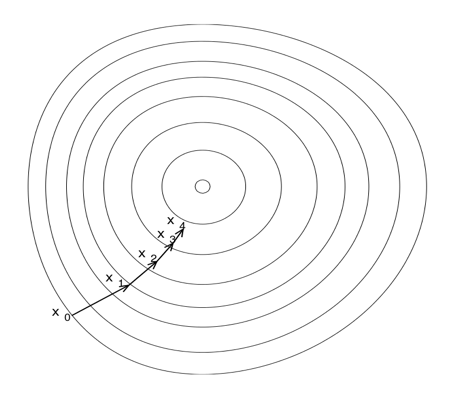


图3.5 梯度下降示意图

全连接神经网络：

由很多层神经元进行相连就会组成全连接神经网络，在全连接神经网络中，每层中每个神经元的输入都会与其上一层的输出相连接，其输出也会与下层每个神经元的输入进行连接，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。



图3.6 全连接神经网络的结构

如图3.6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（见式3.7）。

(3.7)

全连接神经网络的计算极其相似于矩阵的乘法，如式3.8，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：

(3.8)

式3.9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，则W[i][j]表示神经元i到神经元j的权值。

(3.9)

如式3.10，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。

(3.10)

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（见式3.11与3.12）。

(3.11)

(3.12)

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。以下是核心代码：

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))

3.3.5 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（见式3.13）。

(3.13)

如果以测试为目的，也可以使用其他的处理方式处理输出数据，比如概率最高的N个数或保证一定概率的置信数组。下面是本算法中输出部分的核心代码：

prediction = np.argmax(out\_state, axis=1)

3.3.6 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微分，利用偏微分的链式法则（）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。

1、全连接神经网络的反向传播：

之所以叫做反向传播，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，如图3.7，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。



图3.7 反向传播的链式法则



图3.8 反向传播的示意

假设式此网络的偏差方程为方差（见式3.14）。

(3.14)

那么输出层神经元j相对于偏差E的偏导数如式3.15

(3.15)

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，神经元j的输入的偏微分方程如式3.16：

(3.16)

如果使用本算法的sigmoid神经元作为输出神经元，那么j的输入值的偏导如式3.17：

(3.17)

那么如图3.18，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：

(3.18)

如式3.19，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。

(3.19)

按照这样的方法，各权值可以按照自后向前的方式依次求出，这样就可以求出全连接神经网络中每层权值的偏导值。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。

2、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（见式3.20）。

(3.20)

反向传播算法的实现：

由于本算法输出层为sigmoid神经元，且其维度为【批大小\*10】，所以结构如式3.21：

(3.21)

根据前面推导的sigmoid神经元的偏微分公式，输出层的每个神经元的都要进行偏微分方程运算，经过运算的输出神经元偏微分矩阵如式3.22：

(3.22)

如式3.23，隐藏层到输出层权值的偏微分可以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得到，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。

(3.23)

如式3.24，对于偏移量，因为其没有权值（可以看为权值永远为1），所以起偏导就等于其连接的神经元的偏导和。

(3.24)

如式3.25，隐藏层神经元的偏导矩阵可以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，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。

(3.25)

如式3.26-3.28，可以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，隐藏层偏移量偏微分，在这里不进行详细的推导。

(3.26)

(3.27)

(3.28)

如式3.29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微分。

(3.29)

如式3.30，在本算法中，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。

(3.30)

在求出所有变量的偏微分后，将偏微分值与学习率成绩得到所有变量在更新时的变化量，再用其值去减去求出的变化量，就完成了变量值的更新。下面是核心代码：

*# 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]))

3.4 本章小结

本章首先介绍了本手写识别算法的整体架构，然后对本神经网络算法中每一个层次进行了详细的说明与推导，并对每一个层次对整个算法的作用进行了深入的讨论。成功的对算法进行了底层的实现并对实现中遇到的困难进行了分析和解决

4 实验分析与优化

4.1 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。

4.1.1 硬件环境

计算机：MacBook Pro 2016

处理器：Intel Core i7 3.3GHz (2 Cores) CPU

内存：16 GB 2133 MHz LPDDR3 Memory

GPU：Intel Iris Graphics 550 1536 MB

4.1.2 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

4.2 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。

4.3 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：

4.3.1 批的大小



图4.1 线上学习与批学习的权值更新比较

如图4.1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。图4.2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：



图4.2 不同的批大小与速度和时间的关系

由图4.2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。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。

4.3.2 训练次数

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。



图4.3 批大小与训练次数后准确率的关系

由图4.3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。

4.3.3 学习率

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。如图4.4，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。



图4.4 合适，过高，过低的学习率下误差函数与权值更新的关系

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：



图4.5 不同学习率与学习进度的关系

由图4.5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。

4.3.4 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。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：



图4.6 不同隐藏层神经元数量其对测试与训练数据准确率与学习进度的关系

如图4.6 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。

4.3.5 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：



图4.7 卷积神经网络与全连接神经网络识别准确率与学习进度的关系

由图4.7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。

4.3.6 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（见图4.8）。



图4.8 标准化前的与标准化后数据的梯度下降示意

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。

4.3.7 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。(如图4.9)



图4.9 有偏移量与无偏移量的神经网络梯度下降比较

4.4 本章小结

本章着重对本神经网络算法进行分析与优化，首先使用实验的方法在一定范围内在一定范围内对参数进行调整，结合数学推导分析参数的作用，然后结合每种参数的作用，调整参数以得出最佳组合，完全的优化本算法以得到最好的识别率。

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 ; L. Bottou ; Y. Bengio ; P. Haffner, “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.

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, 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.

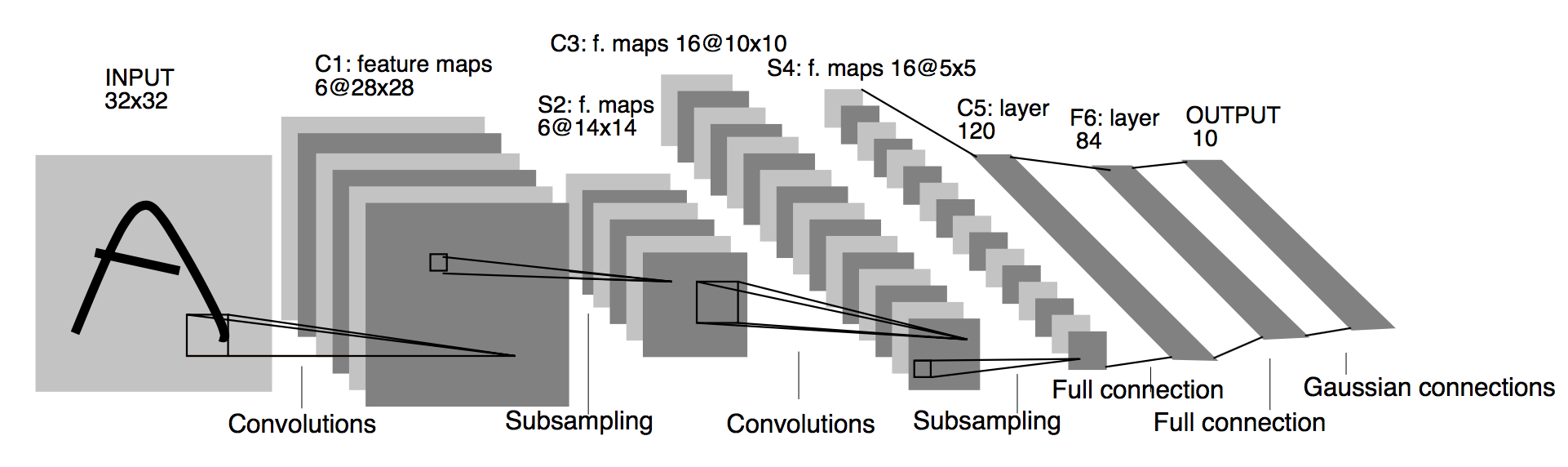


Fig. 2. Architecture of LeNet-5, a convolutional NN, here used for digits recognition. Each plane is a feature map, i.e., a set of units whose weights are constrained to be identical.

 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. Local connections have been used many times in neural models of visual learning. 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. 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 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, 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. 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, machine-printed character recognition, online handwriting recognition, and face recognition. 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), spoken word recognition (with subsampling), online recognition of isolated handwritten characters, and signature verification.

中文译文A

用于单个字符识别的卷积神经网络

通过梯度下降训练的多层网络以其从大量示例中学习复杂，高维非线性映射的能力，成为图像识别任务的佼佼者。在传统的模式识别模型中，特征信息首先由人工设计的特征提取器中提出，然后再消除不相关的变量。然后在用一个可训练分类器将所得到的特征向量分类。在这种算法中可以使用，完全连接的多层网络可以用作分类器。一个潜在的更有趣的方案是尽可能地依赖于特征提取器本身的学习。在字符识别的情况下，网络的输入可以使用原始数据（例如，尺寸标准化图像）。虽然这可以通过普通的完全连接的前馈网络来完成，但是对于诸如字符识别等任务而言，想用这种方法完成是有一定困难的。

首先，典型的图像很大，通常有几百个变量（像素）。例如在第一层中有100个隐藏单元的话，全连接神经网络的第一层将已经包含数万个权值。这样大量的参数增加了系统的压力，因此需要很大的训练集。另外，存储这么多权重的存储器要求在硬件上是无法实现的。但是，图像或语音应用程序的非结构化网络的主要缺点是它们对于输入的翻译或局部扭曲没有容错性。在发送到神经网络的固定大小输入层之前，字符图像或其他2-D或一维（1-D）信号必须在输入字段中被近似大小归一化并居中。但是没有这样的预处理可以是完美的：例如笔迹通常在单词级别被标准化，这可能导致个别字符的大小，倾斜和位置变化。再加上写入风格的变化，会导致输入对象中特征的不同。原则上，足够大小的完全连接的网络可以学习产生相对于这种变化不变的输出。然而，对这样的任务的学习可能会导致具有相似权重模式的多个单元位于输入中的不同位置，以便在输入上利用不同地方来检测不同的特征。学习这些权值配置需要大量的训练实例来覆盖可能变化的空间。在卷积网络中，下文将提到，可以通过强制重复配置跨越空间来自动获得移位不变性。

其次，全连接架构的缺陷是输入的数据拓扑信息会被完全忽略掉。在程序中输入变量可以以任何（固定）顺序呈现，而不影响训练结果。相反，图像（或语音的时间 - 频率表示）具有很强的二维局部结构：他们在空间或时间上附近的变量（或像素）是高度相关的。局部相关性是在识别空间或时间对象之前提取和组合局部特征的众所周知的优点，因为相邻变量的配置可以分为不同的类别（例如，边缘，角落等）。卷积网络通过将隐藏单元的接收域限制为局部强制来强制提取这些局部特征。

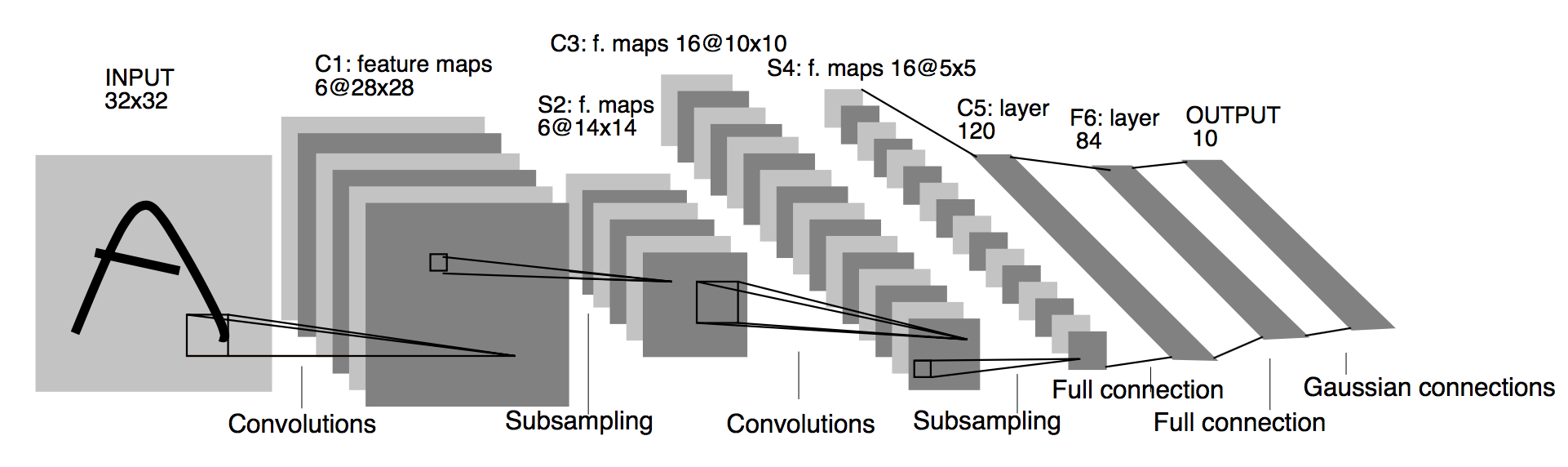


图2 LeNet-5的体系结构，用于数字识别的卷积神经网络。 每个平面是一个特征图，即其权重被约束为相同的一组数。

卷积网络：

卷积网络结合了三种结构上的思想，以确保图片在一定程度的平移，放缩和失真时系统的容错性：

1）局部感受野; 2）共享权值; 和3）空间或时间的池化。用于识别被称为LeNet-5的字符的典型卷积网络如图2所示，输入平面接收大小归一化和居中的字符图像。层中的每个单元接收来自位于上一层中的小窗口中的一组单元的输入。在1960年代早期，将单位连接到局部感受野的想法可以追溯到感知器，与Hubel和Wiesel在猫的视觉系统中发现了局部敏感的，定向选择性神经元几乎同时发生。局部连接在视觉学习的神经模型中被多次使用。使用局部感受野，神经元可以提取基本的视觉特征，例如边缘，端点，角（或其他信号如语音频谱图中的类似特征）。然后将这些特征由后续层组合，以便检测更高阶的特征。如前所述，输入的扭曲或变化可能导致显着特征的位置变化。此外，在图像的一部分上有用的基本特征检测器可能在整个图像上是有用的。该知识可以通过强迫一组变量（其接受领域位于图像上的不同位置）来应用，使其具有相同的权重向量。层中的单位被组织在所有单元共享相同权重集的平面内。这种平面中的单位的输出集称为特征图。特征图中的单位都被限制使其对图像的不同部分执行相同的操作。完整的卷积层由几个特征图（具有不同的权重向量）组成，从而可以在每个位置提取多个特征。一个具体的例子是图2所示的LeNet-5的第一层。 LeNet-5的第一个隐藏层中的单元分布在六个平面中，每个平面都是一个特征图。特征图中的一个单位有25个输入连接到输入中的5 \* 5区域，称为单元的感受野。每个单元有25个输入，因此25个可训练系数加上可训练的偏差。特征图中的连续单位的感受野集中在上一层中的相应连续单元。因此，相邻单元的感受野重叠。例如，在LeNet-5的第一个隐藏层中，水平连续单元的感受野重叠四列五行。如前所述，特征图中的所有单位共享相同的25个权重集和相同的偏差，因此它们在输入上的所有可能位置检测到相同的特征。层中的其他特征图使用不同的权重和偏差集，从而提取不同类型的局部特征。在LeNet-5的情况下，在六个特征图中，在每个输入位置，六个不同类型的特征由相同位置的六个单元提取。特征图将顺序使用具有局部感受野的单个单元扫描输入图像，并将该单元的状态存储在特征图中的相应位置处。该操作相当于卷积，然后是加性偏差和压缩函数，因此名称卷积网络。卷积核是特征图中单位使用的一组连接权重。卷积层的一个有趣的特性是，如果输入图像被移位，则特征图输出将被移位相同的量，将保持不变。该属性是卷积网络对输入的转换和扭曲的容错性的基础。一旦检测到特征，其确切的位置变得不重要。只有其相对于其他特征的近似位置是重要的。例如，一旦我们知道输入图像包含左上区域中的大致水平段的端点，右上区域中的拐角以及图像下部的大致垂直段的端点，则可以告诉输入图像是七。这些功能中的每个功能的精确位置不仅与识别图案无关，但是由于位置对于角色的不同实例而言可能会有所不同，因此有潜在的危险。降低特征位置在特征图中被影响精度的简单方法是减小特征图的空间分辨率。这里可以通过所谓的池化层来实现，所述子采样层执行局部平均和子采样，从而降低特征图的分辨率并降低输出对偏移和失真的灵敏度。

LeNet-5的第二个隐藏层是一个池化层。该层包括六个特征图，每个对应上一层中的每个特征图。每个单元的感受野是上一层相应特征图中的2 \* 2区域。每个单元计算其四个输入的平均值，乘以可训练的系数，增加可训练偏差，并将结果传递通过S形函数。相邻单元不具有重叠的感受野。因此，池化层的特征图具有上一层中的特征图一半的行和列的维度。可训练的系数和偏差控制了乙状结构非线性的影响。如果系数小，则该单元以准线性模式工作，并且子采样层仅仅采用模糊输入。如果系数大，则可以将子采样单元视为根据偏置值执行“OR”或“AND”功能。相继的卷积和二次采样层的交替通常会产生“双锥体”现象：在每一层，随着空间分辨率的降低，特征图的数量会增加。图2中第三个隐藏层中的每个单元。具有来自上一层中的几个特征图的输入连接。其灵感来自于Huelel和Wiesel在福岛的新知识体系中的“简单”和“复杂”单元格的卷积/二次采样组合的实现，尽管没有如反向传播的全程的监督学习过程。通过逐渐增加表示的复杂度（特征图的数量）来补偿空间分辨率的逐渐降低，可以实现对输入的几何变换的很大程度的容错性。由于所有权重都通过反向传播学习，所以卷积网络可以被看作是自己学习的特征提取器。权值共享技术具有减少自由参数数量的有趣的作用，从而降低系统的“总大小”，减少测试误差与训练误差之间的差距。图2中的网络包含345 \* 308个连接，但由于权值共享只有6万个可训练的自由参数。固定尺寸的卷积网络已经应用于许多应用，包括手写识别，机器打印字符识别，在线手写识别和面部识别。称为时间延迟的神经网络（TDNN）的固定尺寸卷积网络在单个时间维度上共享权重。 TDNN已被用于声音识别（无池化层），口语识别（带池化），独立手写字符的在线识别和签名验证。