# 摘要

本文主要介绍利用数字图像处理和深度学习技术，搭建一个中文数字识别的系统。在对手写数字的图像预处理中，使用openCV库对每个字符定位，并将其制作成独立字符。识别图像则是将使用经过10000多张照片训练好的神经网络模型进行预测，并且使用了开源的机器学习平台Tensorflow，网络以原始的神经网络为基础，经过增添多个卷积层和全连接层以提高识别能力，并加以防止网络过拟合技术Dropout优化训练。最后利用PyQt5的GUI开发技术将图像预处理以及网络识别整合，用于测试。测试参数包括准确率、查准率、查全率以及混淆矩阵等等。本文训练的神经网络模型可以作为20个以下的分类系统进行参考。

关键词：中文数字识别；神经网络；数字图像处理

# Abstract

This paper mainly introduces building a Chinese number recognition system with the technology of digital image processing and deep learning. In the image preprocessing of handwritten number, openCV library is applied to locate each character and make it into a separate character. The recognition of image is predicted by using neural network model trained by more than 10000 photos, to which the open source machine learning platform — Tensorflow is exploited. The network is based on the original neural network, adding multiple convolution layers and full connection layers to enhance the recognition ability, and using Dropout — the technology of preventing over fitting to optimize the training. Finally, the GUI development technology of pyqt5 is employed to integrate image preprocessing and network recognition for testing. The test parameters include accuracy, precision, recall, confusion matrix, etc. The neural network model trained in this paper provides a reference for less than 20 classification systems.

**Key words:** Chinese digit recognition; neural network; digital image processing

目录

[摘要 1](#_Toc71882639)

[Abstract 2](#_Toc71882640)

[第一章 绪论 4](#_Toc71882641)

[1.1 研究背景及意义 4](#_Toc71882642)

[1.2 国内外研究现状 4](#_Toc71882643)

[1.2.1 光学字符识别（OCR） 4](#_Toc71882644)

[1.2.2 手写字符识别 5](#_Toc71882645)

[1.2.3 神经网络在图像识别的应用 6](#_Toc71882646)

[第二章 理论基础 8](#_Toc71882647)

[2.1 图像预处理 8](#_Toc71882648)

[2.1.1 图像去噪 8](#_Toc71882649)

[2.1.2 倾斜校正 8](#_Toc71882650)

[2.1.2 字符定位 9](#_Toc71882651)

[2.1.3 图像归一化 10](#_Toc71882652)

[2.2 深度卷积神经网络 10](#_Toc71882653)

[2.2.1神经元 10](#_Toc71882654)

[2.2.2全连接层 11](#_Toc71882655)

[2.2.3卷积层 12](#_Toc71882656)

[2.2.4池化层 12](#_Toc71882657)

[2.2.5 BN层 12](#_Toc71882658)

[2.2.6 softmax函数的概率输出 13](#_Toc71882659)

[2.2.7过拟合（overfitting）及应对措施 13](#_Toc71882660)

[2.2.8网络设计与训练 13](#_Toc71882661)

[2.3 PyQt5 –Python GUI 14](#_Toc71882662)

[2.3.1界面设计 14](#_Toc71882663)

[2.3.2图像处理算法封装 15](#_Toc71882664)

[2.3.3业务逻辑 16](#_Toc71882665)

[第三章 测试与分析 17](#_Toc71882666)

[3.1 训练数据评测 18](#_Toc71882667)

[3.1.1 图片规律 18](#_Toc71882668)

[3.1.2 神经网络训练 19](#_Toc71882669)

[3.1.3 查准率、查全率以及混淆矩阵（precision , recall & confusion matrix） 20](#_Toc71882670)

[3.2 收集数据评测 22](#_Toc71882671)

[3.2.1背景分离测试 22](#_Toc71882672)

[3.2.2.角度校正测试 24](#_Toc71882673)

[3.2.3神经网络准确率测试 26](#_Toc71882674)

[第四章 总结 29](#_Toc71882675)

[参考文献 30](#_Toc71882676)

[附件一 32](#_Toc71882677)

[致谢 33](#_Toc71882678)

# 第一章 绪论

## 1.1 研究背景及意义

谷歌公司的Alphago在围棋比赛中击败韩国围棋大师李世石，标志着人工智能（AI）技术的再次兴起，众多计算机科研工作者转向对AI领域的研究。随着研究的深入，在计算机图像识别上的技术也有了重大的突破。运用深度的神经网络，配合大规模的数据集进行训练，便可以得到一个高准确度的图像识别模型。图像的高准确度识别能够带了丰富的应用，如在医疗方面，机器通过识别不同人的心电图规律，来总结出一套判断心跳异常的规律，从而及时识别出心脏出现问题的病人，提高病人的存活率；在农业方面，机器可以识别出畜生食槽的情况，从而根据动物的数量控制食物的投放，确保食物高效利用；在建筑方面，机器可以通过识别桥、路、房屋的裂纹来判断其损坏程度，及时提出警报以保障住行安全；在交通驾驶当中，汽车能够根据前方路面的情况，自动规划驾驶，解放司机的双手；在安全方面，监控摄像头可以通过高精度识别罪犯面孔，从而达到精准抓捕罪犯的效果。而对于手写汉字字符的识别，则可以帮助文印人员减少人工录入的负担，文印员只需要将整理好的手写图像上传到电脑，通过使用制作好的软件便可以将资料识别存盘，办公速度远比仅用眼睛识别以及手动输入要高效。对于手写字符图像的录入，在这个智能手机普及的时代，已经不是问题，将手写纸置于光亮的环境下，摁下快门键，便完成了数据录入。而对于字符的识别，则是一个比较棘手的问题，汉字的结构比较复杂，字种类也比较多，识别的难度是英文字母远远不能及的，所以制作这样的一个汉字识别的稳定系统，需要对汉字的结构进行详细剖析，而若能够实现，对于中国这一个拥有庞大的人口的国家来说，这一技术定能大大提高生产效率。

## 1.2 国内外研究现状

### 1.2.1 光学字符识别（OCR）

光学字符识别是上个世纪提出的关于打印类文本识别的一个概念，最初针对简单的数字以及字母识别，因为分类不多，并且字符基本都是一个连通的状态，所以首先采用降维的方式进行分类识别，将二维的数字、字母作水平以及垂直投影，统计出两个直方图，然后再与标准的直方图作对比，分析相似程度，将相似度最高的字符进行匹配即可，这种方法简单快捷，计算速度快，但因为一维特征在许多字符当中的相似度比较高，所以如果在预处理阶段处理不好，就会使得识别准确率大大下降，因此对于图像的预处理、机器录入设备的要求还是比较高的。

图 1 水平、垂直投影

随着技术的发展，技术人员也注意到这个问题，开始转向对二维图像进行研究，即采用模板匹配的方法。将字符图像二值化之后，根据字符所在的位置，遍历整个图像与标准的模板图像，进行位置对比，选出相似度最高的输出结果，该方法因为位置信息的增加，使得匹配的准确率有了不小的提升，但相应地在计算速度上会相较于前一种方法慢许多。这样的二维匹配方法也被叫做“窥孔法”，即将像素点视为孔，通过比对所有的模板制作的孔板，得到匹配最好的一个孔板，这种方法也与早期的计算机结构单一，速度偏慢有着一定的关系。

### 1.2.2 手写字符识别

手写字符的识别是在印刷体字符能够识别的基础上，进一步地提出的更加智能的识别方案，手写字体没有固定的书写位置，因人而异，因此识别难度也更高。传统的通过模板匹配的识别方案已经不能得到高准确率，需要另辟蹊径，让计算机根据字符的结构特征来判断其类型。局部分析的方法是一个不错的方法，对于阿拉伯数字，因其围绕着上下两个中心点进行书写，因此可以用类似于“丰”字结构的框架进行划分，通过统计数字穿越“丰”字结构的每个边框的次数，进而识别出其类型。“丰”字框架的运用解决了对于位置敏感的模板匹配，并且其算法简单，易于实现。轮廓分析法也是一个不错的方法，该方法首先定义了笔画行走的8个方向，上、下、左、右、左上、左下、右上、右下，分别对应编码0~7，然后将字符轮廓先提取出来，获取字符的轮廓走向编码（游程编码），再与模板对比，这种方法对于字符位置的敏感程度更为低，并且因为轮廓的可辨性比较强，字符的编码区分度就更为明显。

图 2 局部分析法 图 3 游程编码表

中文汉字的分析与简单的字母、数字大不相同，其结构复杂，类型繁多，简单的结构分析方法已经不满足于区分常用字有几千甚至上万的汉字了。因此，国内学者也对汉字的特征提取进行了很长时间的研究。在金连文的博士论文中提到有笔画密度特征提取[1]，文中提到设想有在汉字的不同方向有扫描线条，不同的线条会穿过字符并标记次数，在书写较密汉字结构处，扫描线的标记次数会有所增加，通过不同扫描线的记录，从而分出汉字的差异，此外，扫描线通过非均匀分布，也更加智能化地利用了汉字书写结构的特征。除了有笔画密度特征的提取，还有汉字外围特征的提取，而这不同于轮廓提取，它是通过水平和竖直方向记录扫描线第一、第二次碰到汉字像素点的位置与外围位置的横、纵向距离来描述汉字轮廓，这种方法在一定程度上也能够体现出汉字的包围结构，对于包围结构的汉字特征有了更细致的描述。与之不同的分析方法还有利用汉字的笔画类型（横、竖、撇、捺）来识别，该方法称为汉字的方向分解。方向分解主要利用汉字的像素梯度变化，利用笔画分量算子可以大致从图像中提取出笔画，而这种笔画的提取是建立在将汉字分布在弹性网络上，每个网格再进行细化分解，其效果较好，得益于方向分解能够得到汉字骨架，使得分析难度下降。当然，特征提取完成后对于识别方法，也做了改进，研究人员采用机器学习的方法，如Bayes分类器法，来优化识别方案也使得最后的准确率大大提高[2,3]。

### 1.2.3 神经网络在图像识别的应用

得益于并行计算、图形处理器（GPU）的快速发展，计算速度在新世纪已经不再是理论实现的瓶颈。神经网络在图像识别上的应用效果显著[4,5]，美国一项由5成高中学生书写，5成人口普查局的工作人员书写的阿拉伯数字数据集MNIST，在普通的机器学习的分类方法训练下，识别的准确率不过70%~80%，而在神经网络的训练下能够轻松达到95%以上的准确率，这样高的准确率使得研究人员相信神经网络能拥有更强大的识别功能。于是，更多复杂的神经网络如AlexNet，其在卷积核（kernel）与扫描步长（stride）上做了变化，使得网络的特征提取能力有了进一步的提升[6]；InceptionNet采用多分支的更合理的卷积模块，使得层数加深的情况下，保持较少的参数量，对于计算速度的提升不言而喻[7-9]；ResNet增加了残差模块用于特征融合，加强了更深层次的网络的识别能力[10]。如今，一些模糊得让人类都分不清的图像，都能够在经过大量训练的神经网络面前败下阵来。而从神经网络衍生的分支——生成对抗网络（GAN）甚至可以模拟通过学习了数据样本之后自发生成新的样本，给自动生成人声、人脸、艺术创造等提供了不少帮助[11]。

# 第二章 理论基础

## 2.1 图像预处理

图像预处理无论是在训练模型还是后期预测，都起到很大的作用，假若数据预处理不好，即使再好的模型，也难以得到高准确率，毕竟模型泛化能力有限，前期的处理能大量地减少预测的负担。手写数字的预处理分为几个部分，图像去噪，倾斜校正，字符定位，图像归一化处理。

### 2.1.1 图像去噪

图像去噪是在彩色图片进行灰度转换之后的操作，其灰度转换公式如下：

对于使用手机拍摄的照片，通常会由于光线的不足，而使得图像二值化处理时难以分清背景和前景，所以，需要在图像二值化之前，对灰度图片进行均值滤波以滤除阴影产生的噪声。图像二值化处理则比较简单，选定一个灰度值作为阈值，超过的则置数255，不超过的视为背景，置数0，这样，由此可以得到清晰明了的手写数字图像。

### 2.1.2 倾斜校正

图片数据的准备需要使用摄像头进行拍摄，难免在拍摄的过程当中由于手抖、视觉误差等人为因素导致扫描成数字图像的照片和真实手写照片产生一定角度的倾斜，因此，一定的小角度校正扫描图片有利于在后续字符定位的之时，判断更为准确。比较成熟的直线检测算法是通过霍夫变换进行直线检测[12]。霍夫变换将传统的直角坐标方程转变为极坐标方程，即（x，y）转换成（ρ，θ），其转换公式为：

经过直角坐标下的某个点（x，y）的直线簇可以表示为以下关系式：

由此，图像上每一个目标点（即字符所在的点）都能够有一个ρ-θ变化曲线，而通过所有目标点的变化曲线就能够找到相交最多的点，而这个点所代表的（ρ’，θ’）则是穿过目标点最多的直线，而校正的角度便是θ’。

图 4 ρ-θ 曲线

### 2.1.2 字符定位

字符定位是图像预处理当中最重要的环节，如果字符定位有误，则识别无从谈起。校正后的图片可以使用像素直方图统计的方法来找到每个汉字书写之间的空隙，进而通过空隙切分出每一个汉字。在切分过程中，由于某些汉字结构导致其书写时并非是一个闭合体，如“八”字，因此，需要特别对这些分离的结构体进行识别、合并。除此之外，某些滤除不了的细小噪声，在定位的时候也可能被识别为汉字，这种情况考虑可以通过面积法，将某些比偏旁定位面积还要小的目标滤除即可。其两种算法流程图如下：

图 5 字符过滤算法

### 2.1.3 图像归一化

字符定位得到的单字符图片在存入Python的列表（list）时已经做了尺寸限制，即全部整理为64×64大小的二值化图片。但由于汉字布满整张图片，与训练模型时使用的数据图片不符，考虑到后期识别准确率的问题，需要将单字图片统一处理，即图像归一化。对比训练图片和实际定位图片的区别，训练图片的汉字多集中在图片中央，整体偏小，因此，缩小定位图片的文字即可。但通过实验发现其准确率仍然不高，观察定位图片发现在进行尺寸限制的时候，会把原来的矩形图拉长，从而显得整个字符偏肥大，一般人写字对于字体的高度控制还可以，但是对于宽度的控制则没有那么敏感，肥大的字体则使用宽度微调的方法，将其“方正化”。

图 6 训练字体与真实手写处理字体对比

## 2.2 深度卷积神经网络

深度的卷积神经网络以LeNet5为原型，在Tensorflow框架的支持下，进行了一定的修改，将参数量、隐藏层数提高以提高识别准确度，并添加了Drop-out、BN层来减少训练模型过拟合现象，网络的原理以及设计在如下几个小结进行介绍。

### 2.2.1神经元

图 7 神经元模型

神经元是神经网络的基本单元，由一个或多个输入，图7当中拥有三个输入，每一个输入都对应不同的权重，把输入的三个信号累加之后加上预设偏置，便可以得到神经元的值，这个值送入激活函数，若达到某一个值，便可以触发神经元传递信号到下一个神经元。Glorot X 等人在文章《Deep Sparse Rectifier Neural Networks》中定义了关于生物学激活函数[13]的模型，其公式为：

其中tref是不应期（两个动作电位之间的最短时间），I是输入电流，Vr是静止电位，Vth是阈值电位（Vth> Vr），R，E，τ是膜电阻，电位和时间常数。而要模拟此函数，通常在数学公式上需要构建一些确切的数值，截止目前，激活函数通常使用ReLU，其表达式如下所示，`

由表达式可以清晰地知道，经过此函数得到的数值，负数全部置零，而正数线性变化，ReLU的计算十分简便，同时求导简单，又有着十分出色的梯度特性，ReLU函数在2012年的AlexNet提出来之后，被广泛地作为神经网络的激活函数。它是受人脑的激活模型的启发，所以，将仿生学的理念应用在计算机视觉上的效果不可估量。

图 8人脑激活模型与ReLU函数对比

### 2.2.2全连接层

神经网络中的全连接层是指将一层的所有输入连接到下一层的每个神经元的层。在大多数流行的机器学习模型中，最后几层是完全连接的层，它将前几层提取的数据模拟大脑神经元传递过程，形成最终的输出，并且，因为网络的全连接，每一个神经元都对应有可变的参数，因此计算上的耗时较大，是仅次于卷积层的第二大耗时层。

### 2.2.3卷积层

卷积层为卷积神经网络的核心。卷积层有多个卷积核心（kernel），在信号处理上也可以成为滤波器（filter），在进行卷积操作时，每个核心将会在上层输出图像上进行滑动卷积，并得到一个特征图，如果是彩色图像，则将得到RGB三通道的特征图。在卷积的过程中，也有一些比较值得关注的参数。一个是步长（stride），步长指的是每一次卷积后卷积核在图像上移动的像素数，步长的设置将会影响后期得到的特征图的形状（shape）；另一个是图像填充（padding），因为每一次卷积都会使得特征图形状减小，所以为了多层的设计下都能够持续进行卷积操作，会考虑事先在卷积前对图像边缘补“0”，使得卷积前后形状一致，这种卷积模式成为“same”，反之，不补“0”则成为“valid”。卷积层的参数相较于全连接层要少，得益于设计者的权值共享的理念，权值共享让每一个核心进行卷积操作的时候使用相同的参数，这就巧妙地避免了大量参数的计算，也让深层卷积层设计成为可能。

### 2.2.4池化层

池化层通常在对图像进行卷积操作之后，特征图往往较卷积后的图缩小一半。每一个图的信息都由参数矩阵来构成，倘若不缩小图像的尺寸，则在遇到全连接层时，计算参数激增，所以对特征图进行池化操作，就能够减少计算量；特征图的所有像素点也并非都有作用，进行池化操作之后，能够有效地去除冗余信息，使得网络识别的难度减少；原来较大的图像通过不断地池化操作，使得特征图不断变小，也有助于卷积核逐步获得更大的感知视野。池化操作常见的操作有平均池化和最大值池化，平均池化更能够保留全局信息，而最大值池化则使得网络对某些特定形状更加敏感。

### 2.2.5 BN层

BN是Batch Normalization的缩写，由于神经网络是以批（batch）来进行训练的，假设批大小（batch size）为m，则每一个网络节点都会有m个输出，对其进行BN操作，就是对输出作归一化的调整[14]。其算法流程大致如下：

图 9 Batch Normalization算法流程

BN层对于加快训练的速度、网络收敛的速度、防空梯度爆炸和消失以及过拟合现象都有着一定的作用。

### 2.2.6 softmax函数的概率输出

softmax函数又称归一化指数函数，是二分类函数sigmoid在多分类中的推广，其目的是以概率的形式表示多分类的结果。其近似概率的输出通常被用于网络的最后一层。其表达式如下：

公式运算是分子将W矩阵（权重矩阵）的第y行与第x行相乘，分母则求和所有行，其原理是通过指数函数转换为概率，分母将所有的结果相加，作归一化处理。

### 2.2.7过拟合（overfitting）及应对措施

具有大量参数的模型可以描述惊人的复杂功能和现象。但是过拟合现象伴随而来：大量的参数意味着网络可以自由地描述被训练的数据，准确度虽然提高，但是对于测试数据并不奏效，而网络也没有意识到训练出现了错误。

Dropout是通过在训练过程中随机和临时以p的概率删除隐藏层中的神经元来实现[15]，通过Dropout，如果某一层有n个神经元，则其前向传播和反向传播的过程当中将会有n×p个神经元得以被利用。利用Dropout技术，产生的输出就像大量网络输出的平均结果一样，而且，由于神经元不能依赖于其他神经元的存在，因此被迫学习不依赖于其他神经元已识别存在的特征。 因而网络具有强大的功能，并且不易受到噪声的影响。

图 10Dropout技术

### 2.2.8网络设计与训练

LeNet5的原型输入的卷积层只有较少的6个核，但第二层增加到16个核，网络的卷积核较少，层数较浅，所以其识别能力相对较弱。再加上原来只用sigmoid作为激活函数，很容易就在训练过程中造成梯度消失。因此，在训练中文手写数字集的时候，需要使用tensorflow的keras接口对其核数量、激活函数以及层数做出修改。本文针对LeNet5进行改进，主要在于加深网络的深度和参数的数量，同时利用Dropout技术来减少过拟合，具体的设计在第三章有提到。

## 2.3 PyQt5 –Python GUI

在完成神经网络模型的训练和数据的预处理之后，整合出一个用于测试的图形化用户界面（GUI）必不可少，一体化的GUI对于预处理图像、更改神经网络以及中间图像变化都带来很大的方便，能够加快测试过程，不断完善识别全流程。

由于整个项目的设计都是用到python语言编写，因此要开发GUI，最好的方法是用python系列的库。常见的python开发GUI库有Tkinter，wxPython以及PyQt5。其中Tkinter是Python的标准GUI接口，其特点是轻量化，其出现时间较早，和Python高度融合，通常在安装完Python解释器之后，就能用import关键字调用，但是轻量也意味着其能够实现的功能比较简单，通常只是某些Python开发者用于简单的测试时使用，一般都是在代码中直接创建完成，对于较多的空间和美观的布局描述起来比较费劲，且文档说明较少；wxPython算是比较受欢迎的，但文档依然较少，假若使用起来还是要从零学起；而PyQt5这个GUI库则可以说是心仪之选，它是源自诺基亚公司的QT5可视化开发平台，PyQt5将C++下的GUI库移植到Python之下，由于之前拥有完整的开发流程，成熟的团队，热烈讨论的社区，其在Python系的开发者中受到青睐，很多的设计思路都能从之前的C++开发借鉴过来，因此，无论是社区讨论还是技术博客，开发过程中碰到的问题基本都有解决方案或者有技术人员解答，且QT的开发是跨平台的，有一次开发，多个平台使用的美名。探索过程中也发现其最简单的开发模式下，让程序运行在Windows，Mac以及Linux系统之下都不成问题，基本满足测试需求。因此选择PyQt5作为开发的GUI库。

### 2.3.1界面设计

PyQt5的设计思路是逻辑与界面分离，PyQt5的界面设计可以采用qtdesigner这个工具，可以直接将控件放置在主窗口上，并且为其命名、设计布局。

本文对于手写图像的处理大体分为三个步骤：图像角度校正、字符区域识别和图像预测，因此在GUI中需要设计一个显示多张图片的容器，PyQt5中的List Widget适合这个选择，该组件适合放置各种类型的文件，如图像、视频、音频以及文本等；考虑到输入图像的尺寸大小不一，测试过程中有可能不方便查看，因此在List Widget旁边加入了滑动条，并设置了两者的“伙伴关系”，使得测试过程中可以拉动滑动条自动调整图像的大小。角度校正的阈值对于检测直线比较敏感，因此如果使用此功能也需要人为地输入一个合适的阈值传入到霍夫函数，使其判断更为准确，这里用spin box作为输入容器即可，并修改一次调整的值为10；最后的识别结果也是需要显示的，因此加入了textBrowser作为文本输出显示器即可。界面在局部和整体里面使用垂直和水平布局，因此整体效果为栅格状，只是每个容器所占栅格有所不同。

图 11布局图预览

### 2.3.2图像处理算法封装

因为PyQt5是面向对象的编程思想，强行植入过程式的算法，显得代码臃肿而繁杂，不易读懂和修改，因此在代码有了一定的长度之后，采用面向对象的编程思想较为重要。因此，这里也将图像识别的算法以类的形式做了封装，以便编写GUI的调用逻辑。本文将按照手写图像处理的三个步骤来封装算法成三个类：Image\_Correction、Recognition以及Predict,三个类在实例之后便会调用自身的初始化方法\_\_init\_\_，该函数内部将图像进行一系列处理，并且使用到类内的方法。下面对每个类内方法的介绍：

类：Image\_Correction，用于角度校正；

方法：

DegreeTrans：角度制向弧度制转换；

rotateImage：将图片逆时针旋转输入的角度；

CalcDegree：通过霍夫变换计算角度

imgNormalization：将输入的图像限定在一定尺寸范围，防止尺寸过大导致算法不匹配。

类：Recongnition，用于定位字符位置；

方法：

getHProjection：获取图像水平投影；

getVProjection：获取图像垂直投影；

pojectionMethod\_get\_Image\_characterPosition：获取图像字符位置；

get\_cutting\_Width：获取字符矩形框宽度列表；

widthMethod\_getPosition：使用字符矩形框宽度筛选出偏旁并合并；

areaMethod\_getPosition：识别面积叫小的矩形框，并认定为噪声，去除之；

draw\_rectangle：在图像上绘制字符识别矩形框

imgNormalization：同类Image\_Correction方法。

类：Predict，用于预测单字符图像；

方法：

predict：预测输入的单字符图像；

imgNarrow：将输入的单字符图像归一化；

topK：给出前K中预测可能；

predictProb：给出预测的概率。

### 2.3.3业务逻辑

在页面设计完成之后，qtdesigner会生成.ui后缀名文件，通过工具PyUIC即可将其转换为.py文件，其中有Ui\_MainWindow类，通过再另一个文件夹继承该类，即可写业务逻辑，而启动该GUI，则用main函数来实例化继承的子类即可。

本项目的业务逻辑相对简单，各组件的触发机理如图12所示。先是由两个按钮和进度条各自带有一个槽函数，系统通过监听它们的信号，从而触发相应的槽函数。button1是识别的核心，以点击作为信号，输入相应图像之后处理并在List Widget和textBroser当中显示处理过程和预测结果；spinBox的输入可作为变化参数以调整霍夫变换的直线检测敏感程度，选之则作为传入参数送入button1触发的槽函数；button2为了使得测试能够连续进行，同样以点击作为信号，实现清屏；而进度条则以其所在位置的百分比为信号，一旦更改，则会相应控制图片缩放。

为了方便理解代码的逻辑关系，这里引用图13来表示各类之间的关系，实线、箭头表示类的继承关系，而虚线箭头则表示依赖关系，这样一来，通过五个类便能够很清晰地描述界面、处理逻辑、图像算法之间的信息传递。

图 12 GUI组件关系

图 13 各类之间的关系

# 第三章 测试与分析

本章内容从理论过渡到实际。将具体讲述训练样本的整体情况，也会介绍神经网络模型训练过程中的一些细节，使得训练参数的调整是向好的。对于制作好的Python GUI，在这部分将会用于测试一些新增的数字图片，这些图片数据是作者在其同学当中使用纸笔、电脑收集的，来源真实可靠。并且，本章将会对收集到的部分数据进行特殊处理，以用于测试不同的参数设定，如人为地拍摄倾斜、故意写字偏离整行的水平线、利用室内昏暗环境制造光影、加入某些细点笔墨作为噪声等等。测试将会以先总结，而后结合图片展示的形式在以下各小节展出。

## 3.1 训练数据评测

训练数据集来源自英国纽卡斯尔大学的一个数据收集的项目，该项目由100名中国人参与了数据收集。每个参与者都用标准的黑色墨水笔在一张白色A4纸上画出15个指定区域，将所有15个数字写在一张网格中。每个参与者重复10次这个过程。每张纸以300×300像素的分辨率扫描。而在kaggle网站上，该数据集已经被进一步整理成为15000张独立的64×64照片，并将照片编号排序，方便调用。

### 3.1.1 图片规律

本文首先将该数据集照片使用Python包matplot画图，抽取一些数字进行分析对比（详见附件一）。其中发现，数字“十”和数字“千”之间比较容易混淆，因为有的人对“十”字的“竖”笔画常常不在意是否笔直，所以如果不留意写歪曲的话，将会很容易写成“千”字；另外，“二”和“三”字之间也可能产生误判，原因是有的人在写字的时候，顺带连笔，导致“二”字中间多了一些笔墨，变成了一个“三”；“七”、“九”和“亿”之间也会有混淆的情况，原因是它们的最后一笔都含有“弯钩”；“百”和“四”区分度还算大，但因为都有包围的结构，所以若是模型训练精度不够，也会有混淆的可能；最不容易发生误判的是“零”，因为“零”的笔画最多，结构最复杂。每一个数字在矩形框内位置都是不固定的，这对于位置敏感的算法来说是一个很大的缺陷，但神经网络允许这样的位置偏差，因为本来的设计就是模拟人眼结构，所以，对于这样的位置误差无关紧要，但考虑到识别的准确度问题，在模型训练好之后，还是将测试数据进行归一化调整。

图 14 训练数据样本（局部）

### 3.1.2 神经网络训练

训练神经网络模型部分本文首先将LeNet5作为预先实验，当然，也有改动过LeNet5的激活函数为ReLU，因为ReLU的非线性效果最好，训练过程如图15所示。从准确度曲线来看，中文汉字数据集的训练难度不算很大，能够在20个epoch之时收敛，但是，由于网络比较简单，在后面的训练过程当中表现出了不稳定的现象，但总体都能维持验证集准确度在95%的水准。本文将改进后的LeNet5命名为DC-LeNet，网络包含五个卷积池化层（网络通常将卷积和池化这两个操作算在一个层里，便于描述），并且卷积核的数量有所增加，而后面的全连接层的层数也有所增加，每层的神经元数量以10倍的数量增加，训练过程明显感受到速度变慢，但变慢的代价换来了高准确率，从图中可得知，其训练的收敛速度更快，大概在第10个epoch的位置就能够收敛，而且收敛之后稳定的时间更长，虽然在训练的末期有一个准确率有一个剧烈都抖动，但整体曲线表现要优于LeNet5。有关两个网络的具体每一层的结构、输出图的形状以及可变参数统计可看到表1。结合表中两个网络结构分析可知，DC-LeNet的网络参数量是LeNet5网络参数量的十倍，而参数分布最多是在全连接层的第一层，符合人体神经细胞感知由多变少的过程。

图 15 LeNet5训练过程

图 16 DC-LeNet训练过程

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| LeNet5 | | |  | DC-LeNet | | |
| Layer | Output Shape | Param |  | Layer | Output Shape | Param |
| Conv2D | （None, 64,64,6） | 156 |  | Conv2D | （None, 64,64,64） | 1664 |
| Dropout | （None, 64,64,6） | 0 |  | Dropout | （None, 64,64,64） | 0 |
| MaxPooling2D | （None, 32,32,6） | 0 |  | MaxPooling2D | （None, 32,32,64） | 0 |
| Conv2D | （None, 32,32,16） | 2416 |  | Conv2D | （None, 32,32,64） | 102464 |
| MaxPooling2D | （None, 16,16,16） | 0 |  | MaxPooling2D | （None, 16,16,64） | 0 |
| Flatten | （None, 4096） | 0 |  | Conv2D | （None, 16,16,128） | 204928 |
| Dense | （None, 120） | 491640 |  | MaxPooling2D | （None, 8,8,128） | 0 |
| Dense | （None, 80） | 9680 |  | Conv2D | （None, 8,8,256） | 819456 |
| Dense | （None, 15） | 1215 |  | MaxPooling2D | （None, 4,4,256） | 0 |
| Total params: 505,107 | |  |  | Flatten | （None, 4096） | 0 |
| Trainable params: 505,107 | |  |  | Dense | （None, 1024） | 4195328 |
|  |  |  |  | Dense | （None, 512） | 524800 |
|  |  |  |  | Dense | （None, 128） | 65664 |
|  |  |  |  | Dense | （None, 15） | 1935 |
|  |  |  |  | Total params: 5,916,239 | |  |
|  |  |  |  | Trainable params: 5,916,239 | |  |
| 表1 LeNet5和DC-LeNet的网络结构、输出图形状及所需训练参数 | | | | | | |

### 3.1.3 查准率、查全率以及混淆矩阵（precision , recall & confusion matrix）

查准率和查全率一般针对的二分类问题所设置的指标。查准率表示的是在判断为对的样例当中，到底有多少个是真的对的；查全率表示的是在所有对的样例当中，判断对了多少。对应到本文的数据，查准率和查全率就用单独的一个字看作一类，其他的字看作是另一类来计算。用表格和公式来描述两个指标如下：

|  |  |  |
| --- | --- | --- |
| **真实情况↓模型预测→** | True | False |
| True | True Positive(TP) | True Negative (TN) |
| False | False Positive(FP) | False Negative(FN) |

根据公式对15个中文数字在验证集（validation data）上进行统计，发现神经网络DC-LeNet对“十”和“千”的查准率略低，这也在3.1.1小结的图片规律当中有提到；在查全率中，“一”的分数是100%，说明简单的数字，有利于系统辨别，其他数字得分差距不大。

表2 DC-LeNet的查准率和查全率数据

在混淆矩阵的数据当中，能够看出各数字出现误判的具体问题，比查准率和查全率更为详细。结果发现，所有的数字误判为“零”的案例较多，网络模型虽然能够大部分判断准确，但小部分判断失误且给出比较离谱的可能，这是模型过拟合的一种表现，但因为准确率比较高，稍微的过拟合是在可接受范围内的。

图 17 DC-LeNet的混淆矩阵

## 3.2 收集数据评测

收集数据不同于测试数据，是笔者在现实当中收集的不同人的手写数据，其中背景环境比较复杂的是利用手机在不同光线下进行拍摄收集制作而成，背景比较单一的是使用电脑软件进行书写。

### 3.2.1背景分离测试

背景分离使用的测试图片有纸张后面包含有多种颜色的底版作为背景的、纸张在拍摄过程中用手遮挡造成阴影的以及纸张在光亮的环境下拍照的三张图片。

纸张在光亮的环境下拍摄的照片比较容易实现背景分离，在分离灰度强度阈值大于80的情况下，都能够把手写汉字与纸张空白处分离开来，但也不是说分离阈值越高越好，因为高阈值的分离意味着算法对于噪声不敏感，后期处理噪声同样会很麻烦。所以分离阈值在120~140之间为最佳，既能够分离背景，又不至于把多余的阴影噪声识别进来。

图 18 简单背景阈值分割测试

在用手人为地遮挡产生阴影后，对于背景分离的阈值把控无疑更为困难，阈值过高（160），大部分的阴影都会识别成为物体，导致几乎分离不出手写字体；阈值在中间位置（80~120），也仍然有部分字体被过暗的阴影影响；阈值再小一点，能把阴影分离开了，但有部分字体又开始变得模糊，因为笔画的光线被部分视作阴影而分离开。因此，过多的阴影影响会造成背景分离顾此失彼的现象，在实际识别当中亦要尽量避免。

图 19 阴影下阈值分割测试

复杂环境测试的拍摄底板是偏暗色的木头，一把橙色水果刀和游戏鼠标分居纸张左右侧，光线从右下角摄入，鼠标在光线的投影之下产生一些阴影。测试图片当中最有可能分离出字体的是阈值为160的样张，但是即便是最合适的样张也无法分理处全部的字体；在所有的阈值测试当中，刀刃永远存在于图片的左边，而鼠标与其阴影在高阈值的情况下会识别成背景，随着阈值的降低，鼠标的轮廓会渐渐明显，一直到可以分辨出鼠标上的细节、纹理。复杂环境的测试说明对于简单处理的背景分离算法，最好不要让图片出现无关物体干扰。

图 20 复杂背景阈值分割测试

总结以上的三个分离背景的结果，有多种颜色作为底版的图片无法识别出背景，所以在测试准确率的时候尽量使用单色背景；在有阴影影响下的图片如果没有很好的分割阈值，同样会将部分阴影识别为字符区域，而在光亮环境下拍摄的照片则可以轻松分离出背景，所以对于分割阈值的把握，也是背景分离的难点所在，本文测试得到的背景分离的最佳阈值在120~140之间，具体的数值，还需要视识别图片的阴影环境来调整，阴影过多使用小阈值，阴影较少则使用大阈值。

### 3.2.2.角度校正测试

对水平线不敏感的人通常在拍摄图片时导致手写的数字倾斜一定角度，如图21，另外，有的人在写字的时候习惯于将字往上写，如图21，这在对汉字进行水平投影的时候会造成一定的干扰，所以需要在预处理阶段，将图片的角度进行略微调整，以获得最佳的投影图片。

图21所示的照片是真实手写并拍摄的，在进行霍夫直线检测的过程当中，阈值检测越低，能够检测出来的直线就越多，当然，并不是直线越多越好，过多的直线会造成系统无法识别出最佳旋转角度的范围，过少的直线（如阈值为160时）会导致纠正的角度偏小，检测的直线应该在布满整行或略多一点为佳，在图中阈值为100和120的图片在计算旋转角度的时候最为准确。

图 21 拍摄照片霍夫直线检测

图22是模拟人手写字时，逐渐往上位移的过程，这种情况在要书写的汉字过多时经常出现。霍夫直线检测的结果与图21类似，区别的地方在于其照片的尺寸较之前的图片大约1倍，因此需要进行霍夫检测的直线阈值也需要增加差不多一倍由此可知，霍夫直线检测对于照片以及字体所占像素的大小是十分敏感的，所以在拍摄测试图片的时候尽量要统一图片和字体的尺寸。该角度的校正基本满足后期的水平投影，只是校正的过程通常会导致头几个字的略微倾斜，如果书写者书写往上的位移不大，则对于后期的识别字体没有影响。

图 22 电脑照片霍夫直线检测

霍夫直线检测函数当中还有一个重要的参数：极径ρ，通常设ρ=1，即单位长度，改变极径的长度也会对检测到的直线产生影响，这里使用参数1-8来做测试，发现极径越大，对于图片中的直线的检测就越敏感。极径为1时，函数只能监测到横向的直线；为2时，开始出现竖直方向的直线；继续加大之后，就会出现各个方向的直线。因为本文对于检测的直线只要求简单水平方向，所以将极径设置为1即可。

图 23 极径参数测试

### 3.2.3神经网络准确率测试

在本文提到的两个网络当中，虽然其训练模型时的准确率都比较高，在95%往上，但是由于训练样本的采集方式和测试样本的采集方式有所不同，在真实测试书写数字的识别率上也略有下降，差异主要在图像的预处理上。实际测试两个网络的识别效果发现，LeNet5的测试准确率比较低，通常15个数字能错3-4个，而DC-LeNet则可以稳定在1个甚至没有识别错误。由此可知，训练时模型的验证集（validation data）的准确率越高，就越能够允许模型去适应更陌生的图片。

在神经网络的训练过程当中，topK预测指标表示预测一个图片的前K个可能，该指标通常也会被用于衡量一个模型的预测准确程度。本文对两个网络都进行了topK（K=3）指标的测试。测试结果如表3、4所示。从中发现，DC-LeNet的topK指标呈现一边倒的情况，即预测图片通常一个类的可能概率为100%，其他类的可能概率都为0%，这与训练模型的阶段使用的数据集和训练的循环次数（epoch）有关，是过拟合的表现。当训练模型时，验证模型的准确率超过90%甚至达到95%之后，通常对于后面预测阶段，给出来的预测可能都会出现一边倒的可能。而LeNet的topK指标识别准确率则可以达到100%，即使第一预测可能不准确，也能在第二、第三预测可能中弥补。两种模型没有谁好谁坏，只能说对于具体生产要求不一样的时候，选择的模型也可以不同。这对于开发识别类的软件也给出一个启示：模型预测并不是验证的准确率越高越好，如果希望在预测结果上得到更多的解决方案，则需要在训练的时候通过topK指标来训练出最合适的模型。

|  |  |  |
| --- | --- | --- |
| 图像 | （预测结果1，概率） | （预测结果2，概率） |
|  | （“一”，100%） | （“零”，0%） |
|  | （“二”，100%） | （“零”，0%） |
|  | （“三”，100%） | （“零”，0%） |
|  | （“四”，100%） | （“零”，0%） |
|  | （“五”，100%） | （“零”，0%） |
|  | （“六”，100%） | （“零”，0%） |
|  | （“七”，100%） | （“零”，0%） |
|  | （“八”，100%） | （“零”，0%） |

表3 LeNet-5的top-3预测

|  |
| --- |
| （预测结果3，概率） |
| （“万”，11.6%） |
| （“千”，17.1%） |
| （“五”，0.45%） |
| （“七”，0.03%） |
| （“四”，0.06%） |
| （“七”，0.06%） |
| （“四”，0.03%） |
|  |

表4 DC-LeNet的top-3预测

以下是DC-LeNet的一些测试数据在Python GUI上的示例，大致分为电脑书写和实际用笔书写的预测，准确率都在90%以上。其中也出现了两处“十”误判为“千”的可能，对于手写数字，由于拍照的环境变化较大，而图像预处理的数字也会随之变化，图29当中，粗体的数字也让网络模型出现了比较幼稚的预测错误，即“八”预判为“九。”

图 24 电脑图片一行预测

图 25 电脑图片二行预测

图 26电脑图片三行预测

图 27手写图片一行预测

图 28 电脑图片二行预测

图 29 手写图片三行预测

# 第四章 总结

本文的设计分为图像预处理，神经网络训练以及Python GUI的制作三大部分，在其中遇到的疑难杂症，注意事项，在此一并总结。这不仅是对自己的一个教训，也能给后来者一个参考。

图像预处理运用数字图像处理的知识[16]，相关的处理算法多而杂，初始接触不容易掌握。对此，可以对每一个提到的算法都进行尝试，通过肉眼观察处理出来的图像好坏，再回过头来理解算法的原理就会少了很多枯燥。比如在图像的去噪上，分为空间域和频率域的滤波算法，空间域里又有均值滤波、中值滤波、高斯滤波等，要想直接通过理论去推敲哪种滤波器去噪性能比较好是不太可能的，因为处理的照片情况多变。因此，逐一进行尝试，就能够在最快的时间内选出符合所需求的滤波器。

神经网络的选择要视识别内容的难度来定，并不是网络层数越深越好，使用的网络结构越先进越好，待训练的参数越多越好。对于简单的分类问题，使用LeNet5已经足够，甚至还可以减少卷积核的参数；而对于像识别复杂背景下的某些物体，如动物、人脸、肺部图片的变化等，则需要使用像resnet一类的对加深网络有显著效果的网络。网络的选择需要在搭建的难度、训练时间以及识别效果三方面综合考虑。另外，对于训练过程的loss和accuracy曲线，也要多加观察，及时停止网络的过拟合训练，以达到模型最好的训练状态。训练效果的好坏也不单单只看loss和accuracy曲线，还得结合后期测试过程中的如topK指标来逐步完善预测模型。对于复杂的网络，如果配置较低的电脑运行较慢，可以考虑使用谷歌的机器学习竞赛平台Kaggle或者云平台Colaboratory，其为研究者提供了专用的GPU进行加速，让复杂模型的训练不再是难题。

Python GUI的制作本文采用的是跨平台的Qt5技术，采用Qt5也是出于对参考资料的全面，应用平台广泛而考虑。一个好的开发框架能够为后期的测试带来事半功倍的效果，因为资料的全面使得开发的所有技术细节都得以实现。如果只是想实现一些更为简单的功能，不考虑界面的话，也可以采用Python的Tk GUI工具包——Tkinter，其设计之初就是只为使用代码开发，轻量而高效；如果考虑更复杂的设计，可以考虑web开发技术，web开发技术在如今趋于成熟，H5已经成为了互联网的下一代标准，使用网页端开发应用，可以说是一次开发，多端使用，采纳该方案可以考虑更加丰富的动态效果、更加灵活的设置以及更加长久的维护使用。

本文开发使用的Python IDE为Spyder，虽然功能简单，但是可以实时显示代码执行过程中变量的变化，基本满足科研需求，也建议初学者可以能够很好地利用它来熟悉代码的编写思路，进而修改处自己理想的效果。

# 参考文献

1. 金连文. 手写体汉字识别的研究[D].华南理工大学,1996
2. 周志华. 机器学习[M]. 北京：清华大学出版社，2016.
3. 李航. 统计学习方法[M]. 北京：清华大学出版社，2012.
4. Lecun Y , Bottou L . Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324. 【lenet5】
5. Gonzalez R C. Deep convolutional neural networks [lecture notes][J]. IEEE Signal Processing Magazine, 2018, 35(6): 79-87.【lenet5】
6. Krizhevsky A , Sutskever I , Hinton G . ImageNet Classification with Deep Convolutional Neural Networks[C]// Advances in neural information processing systems. Curran Associates Inc. 2012. 【alexnet】
7. Szegedy C , Wei L , Jia Y , et al. Going Deeper with Convolutions[J]. IEEE Computer Society, 2014. 【inception-net】
8. Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826. 【inception-net】
9. Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-resnet and the impact of residual connections on learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2017, 31(1). 【inception-net】
10. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778 【resnet】
11. Goodfellow I J , Pouget-Abadie J , Mirza M , et al. Generative Adversarial Networks[J]. Advances in Neural Information Processing Systems, 2014, 3:2672-2680.【GAN】
12. Princen J . A formal definition of the Hough transform: Properties and relationships[J]. J.of Math.imaging & Vision, 1992, 1(2):153-168.霍夫直线
13. Glorot X , Bordes A , Bengio Y . Deep Sparse Rectifier Neural Networks[C]// Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS). 2011:315-323.【激活函数】
14. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]//International conference on machine learning. PMLR, 2015: 448-456. 【batch】
15. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The journal of machine learning research, 2014, 15(1): 1929-1958.【dropout】
16. Gonzalez R C, Woods R E. Digital Image Processing. 4th ed. New York: Pearson-Prentice Hall, 2018【数字图像处理技术】

# 附件一

# 致谢

毕业设计以及论文前前后后、断断续续做了有5个月，这篇论文的完成，也标志着我的大学本科生涯到了尾声。

我应该感谢我的家人。我的父母虽然没有稳定的工作、收入，但是，他们知道陪伴对于一个孩子的意义。高中以前，基本上每天放学归家都能见到父母。每天晚上，都能够和我的姐姐坐在饭桌前，或是听爸妈讲“经历”；或是听老姐讲在学校的奇闻奇事；又或是某一回我想做主角，说着一些幼稚的话；又或是对骂开撕。无论好坏，至少，我的童年时时刻刻有着家人的陪伴，我没有感受到孤独，我想我是幸福的。我的爸妈在我的物质需求、教育上从没有缺斤少两，我知道这是伟大而无私的。我的姐姐毕业后签下银行的工作，给家里减轻了负担，也在我上大学的几年给予了我诸多关爱。亲情浓于水，我会好好珍惜。

我也要感谢大学里面每一个尽心尽责教育我的老师。大学上过的课也有超过50门，在我的记忆当中，教大学物理的廖红老师印象最为深刻。每一次大物课的早上，带着昏昏欲睡的状态进教室，却都能听到她将枯燥的知识娓娓道来。教微积分的杨爱芳老师，每一次上课都能将内容十分清晰地板书，让我们轻松地翻越“高数”大山。当然，我的毕设指导老师黄龙庭也在我的最后这一学期给予了我很大的帮助，除了教会了我在整体上布局毕业设计，还在设计的实验内容，实验数据的记录与展示上给予了很多的思路，可谓受益良多。

我比较愚笨，除了在小升初垫底进入市重点中学以外，初升高拿到不到保送的名额，高考也考得比平时很多时候都要糟糕，大学挣扎了三年也没有拿到保研的名额，雅思考了两次也还是达不到出国的要求。不过，到现在，我也坦然接受这个自己，毕竟不可能每个人的学途都是一帆风顺的。所以，如果不能跑，那就慢下来走，如果走不动了，那就爬，只要今日比昨日进步了，那就是没有什么可以抱怨的了。毕竟，我还有很多的愿意帮助我的老师、家人和挚友，珍惜自己所拥有的，就不会患得患失。换个角度去想，也许每天就能过得快乐一些，至少，不会怨天尤人。

论文完成过后，我也会珍惜还在学校的时间，继续备考雅思，不虚度光阴，努力做一个思想上进的青年，给我的本科生涯画上一个不留遗憾的句号。

4.30.2021