一、Cnn简介

卷积神经网络的基本原理是通过卷积层对输入信号进行特征提取，然后通过池化层对输入进行缩小操作，最后通过线性层和简单的bp网络一样将信号进行传输。而训练过程则是使用反向传播算法，目的是最小化损失函数。通过不断调整卷积核的权重和偏置，优化目标识别的效果。

卷积层

卷积核的原理就是使用一个矩阵对矩阵范围内框选到的数据进行加权求和，然后得到一个新的数据，将一系列的数据拼接成一个新的数据作为卷积层的输出，其目的是为了提取区域的特征。卷积层的反向传播原理其实就是利用反向传播算法将输出结果与实际值比较的损失函数反向传播。

池化层

池化层核的原理就是选取某一区块的特征值进行提取，其目的就是为了缩减输入数据的维度，并且保留原始特征。一般在实际操作过程中，都常用max\_pooling的方法，即选取某一区块中的最大值作为该区块的特征值进行压缩。而由于池化层操作没有参数，所以不需要进行反向传播的更新操作。

线性层

线性层一般放在最后，进行对向量的降维和分类。之前卷积层和池化层输入的数据为高维数据，首先要对数据进行展平操作，将数据变成一维的数据，以方便后续线性层的处理。然后线性层的前向和后向传播则是与bp神经网络相同，网络中的每个节点包含不同的权值与偏置，通过反向传播算法对权值进行优化以使得最后输出的误差函数尽可能小。

二、代码基本结构

代码主要分为model、main、test。其中model就是卷积神经网络模型。Main中包含了数据处理、训练、预测过程。Test则是利用保存的模型进行预测并输出准确率。

Model.py：

在model模块，主要包含两种模型，一种是LeNet-5，复现了其每一层的结构，但是原版的输入维度为32\*32，本实验的手写汉字的输入维度为28\*28，所以对线性层的节点个数有所修改，除此以外，包含通道数、卷积核大小、池化层核大小、步长、padding均保留为原始结构。另一个结构是自己尝试的cnn结构，相比LeNet-5，MyCNN只使用了两个卷积层和两个池化层，通道数有所增加，1->16->32。除此以外，卷积层的核大小也从5改成3，并且将padding改成1，以保持特征向量的大小不变。线性层只使用了一层，一方面增加了网络的运行效率，另一方面也可以有效防止线性层过多导致的梯度爆炸问题。

在激活函数的选择方面，我比较了relu函数和tanh在cnn中的效果，最终选择了准确率更高和选择的人最多的relu作为激活函数（是在已经做好优化之后才进行的激活函数的尝试）。

在后续实验过程中，还加入了批标准化（batch\_normalization）和dropout的优化方法，所以增加了bn层和dropout层，但是在实验过程中发现增加了dropout层之后的准确率反而下降了，所以具体结构并没有使用dropout，而是只使用了bn。

Main.py：

在主函数中主要就是进行数据的预处理和进行神经网络的训练和预测。

首先进行的是数据的预处理，第一步先使用cv2库将数据从.bmp文件转换为array，并用文件夹名对汉字的类型进行独热编码，第二步是调用transform包进行数据增强（由于增强的方式不能够提高学习率，后续对这一操作进行了改进，将原始的随机裁剪和水平反转改成随机旋转15°以内的角度，结果与不进行数据增强的结果差别不大，所以后续操作中不使用数据增强）。第三步是划分训练集与测试集。

主函数中进行的操作主要就是调用模型，然后用训练集进行训练和使用测试集进行预测和计算准确率，并且将训练好的模型存入对应的文件中。

Test.py：

在测试文件中主要就是load已经训练好的模型，然后使用模型测试准确率。

三、改进方法及算法原理

LeNet-5

最开始使用的模型就是经典的LeNet-5模型，由Yann LeCun发明。

网络结构如下图：

按照 网络的构思，其输入为32\*32的图片，但是手写汉字只有28\*28，所以为了保证所有的特征笔画都处于卷积核的中心，可以使用卷积核中的padding操作对图像边缘进行填充。

Adam

首先加的第一个优化就是最出名的adam优化。其目的是使得梯度优化的过程能够尽可能的快，并且等到数据误差小的的时候会缩小梯度变化的尺度使得尽可能的逼近局部最优，而不至于在局部最优处反复震荡。

其具体实现为：

批标准化

在之后我还加上了一个批标准化的优化，一方面可以使得每一层输入的值都是有效的值，不会出现巨大的偏差，另一方面可以有效防止梯度消失、过拟合等现象的发生。

其实现原理为在每一层之后都加入一个批标准化层，计算上一层输出的均值和标准差，对输出的数据进行以batch为单位的标准化。

公式

集成

集成算法的思想和投票类似，就是通过训练多个弱的训练模型，然后通过一定的策略将这些模型进行组合，得到一个更强的模型。

具体来说的算法分为bagging、boosting、stacking三大类：

Bagging的想法就是利用一个数据集中的不同子数据集对不同的子模型进行训练，这些数据集可能存在交叉，然后分别训练这些子模型，最终集成得到一个更好的强训练模型。

Boosting的想法就是通过反复学习得到一系列弱分类器，然后对这些弱分类器进行组合，最后得到一个强分类器。主要思路是，模型的一方面来自于上一个模型，另一方面来自于之前所有模型的加权总和。

Stacking的想法是首先用一部分数据训练出几个基模型，然后用另一部分数据放入模型当中进行预测，得到预测结果之后又将结果作为训练集对原来总的模型进行训练，这样得到的模型将会是一个整体。

在实现过程中，我主要使用的是bagging方法，在训练集的选择上，一开始使用的是全部的训练集作为输入，这样方便实现，并且在输出结果的时候就是简单的输入为多个模型输出的均值。在后续改进过程中使用随机的2/3的训练集作为输入，准确率有所提升。

数据增强：

数据增强的原理就是对输入数据进行平移、压缩、旋转等方法使得数据更加多样，以增加模型的泛化能力，防止过拟合。

Dropout：

Dropout方法的思想类似于集成方法，在训练过程中，随机裁剪掉层内的一些节点，然后每次训练时裁剪掉的节点都不相同（可能会有重合），然后最后在测试时又将所有的节点释放，对数据进行预测。这样做一方面可以防止过拟合，增加模型的泛化能力；另一方面可以通过类似投票的方式提高模型的准确度。

自适应学习率：

自适应学习率的方法有点类似于adam优化器，即在训练过程中，当模型的准确度不再上升时，说明可能已经达到局部最优，也可能到达欠拟合和过拟合的边界，这时候每训练一次就将学习率降低为原来的1/10，以逐步逼近局部最优，避免在最有附近震荡。

防止过拟合的方法（bonus）

以上方法为实验中尝试了提高准确率的方法。其中能够防止过拟合现象发生的方法包括批标准化、模型集成、数据增强、dropout方法、都可以通过泛化模型来达到防止过拟合的目的。而自适应学习率则是通过调整训练次数过多以后的学习率，避免训练次数过多而导致梯度下降之后的权重和偏置过于拟合训练数据，从而避免过拟合现象降低准确率。从这个角度出发，adam优化器也可以通过调整梯度变化的大小一定量的避免过拟合现象的发生。

四、网络的改进实验

了解了以上方法之后，便是设计实验论证以上方法的有效性。

**第一轮优化**

最开始使用的模型就是一个简单的LeNet-5模型，在不使用任何优化的情况下，结果表现非常差，只有个位数的准确率。

然后尝试使用了adam优化器，效果十分好，在只训练10次的情况下直接将准确率提升到79，然后在100次的情况下，准确率达到96.6，在训练1000次的情况下，准确率达到98.3。

在此之下改进了一下网络结构，在之前的基础之上，增加了卷积层的通道数使得特征值更多，并且增加了padding使得所有的特征值都聚在卷积核的中间，减小了卷积核的大小，使得经过卷积层之后仍然能保持更多的特征值，最后减小了线性层的数量，避免发生梯度爆炸的现象。改进了网络结构之后，准确率也有所提升，训练10次，准确率就达到了90.4，训练100次就达到了98.1，训练1000次就达到了98.7。

为了提高测试的效率，之后改进方法的训练次数都只使用10次，并且网络结构确定使用自定义的网络结构。

接下来测试的是批梯度下降的方法，效果明显，在只训练10次的情况下就能够达到95.6的准确率，说明每一层之后进行标准化数据确实能够有效提高准确率。

然后使用的就是集成方法。我主要使用的是bagging方法，在训练集的选择上，一开始使用的是全部的训练集作为输入，这样方便实现，并且在输出结果的时候就是简单的输入为多个模型输出的均值。这样做使得准确率提高到了97.3。

然后尝试使用了数据增强技术。在实现过程中，一开始使用了裁剪和反转的方法，但是在助教的提醒之下，发现以上的方法并不适用于手写汉字识别，因为会破坏汉字原本的结构，准确率没有太大的变化，只有97.2。所以在第一次实验中并没有增加数据增强操作。

然后在网络中尝试dropout方法，由于只有一层线性层，所以将dropout层放在第二个卷积层之后，结果反而降低了，只有96.1。可能是因为dropout方法在使用时不能用于卷积层之后，卷积层之间的相邻的像素包含相关的信息，而删除这些节点则可能导致传播过程收到影响。所以在一开始并没有使用dropout方法。

最后一个优化是对学习率的优化，使用了ReduceLROnPlateau函数能够自适应的调整学习率，设置3轮之内准确率没有上升则将学习率降低为原来的1/10，以防止过拟合的发生，最后的准确率也成功提高到98.2。这个也是优化当中最高的准确率。

后续所有工作将在之前的优化当中又进行优化。

**第二轮优化**

首先对网络结构进行了尝试优化，由于之前在bp网络中使用的是tanh函数，所以cnn也尝试使用，结果明显不如relu的表现，为95.4，可能是因为对于卷积核，使用更大的梯度可以使其更好的收敛。

接着对数据增强进行了改进，将原来的方法改成了使用旋转的方法。随机旋转15度以内的角度以增加数据的多样性，结果的效果和原始比较差别不大，甚至有所下降，为97。

然后对集成方法进行了改进，首先使用的方法是将数据分割成很多块，然后分别对不同块的数据进行学习，得到若干个模型，最后将训练的结果加和，这样的准确率可以达到98.3。从原来初始的使用整个训练集训练改成了使用随机的2/3的训练集作为输入，准确率提升到了98.8。

为了使用dropout操作，并且同时为了避免单层网络降维过快导致的参数爆炸，将线性层进行了扩展，使得原来的单个线性层变成了两个线性层，从原来的32\*7\*7->12中间加了一层含120个节点的线性层。结果可以达到92，较之前有所提升，在使用dropout操作后，准确率可以达到98.9。

五、对网络设计的理解

相比于简单的bp全连接神经网络，卷积神经网络更加适合于图像识别，因为其算法中的卷积核天然自带多维向量的特征提取的功能，所以很快就能够达到比简单的全连接神经网络更高的准确率。但是如何进一步提高cnn的准确率则需要诸多的优化方法。

在优化过程中，我深刻体会到优化方法对网络设计的重要性。一个好的优化方法可以极大提高网络的准确性和泛化性，同时也能够提升训练和学习的效率。在设计过程中，我尝试了很多方法，其中有的方法能够使准确率有很大提升，有的方法则比较不适用于当前的模型，同时又有的方法之间能够相互作用共同提高识别的准确度，这些都需要通过实验进行验证。网络设计不仅是对方法的掌握，还是一门实验科学，需要不断的尝试，结合实际的模型进行验证才能够得到一个最优的算法模型。