cnn文档

一、代码结构

1、代码包结构

cnn

|--cnn\_v1.py cnn神经网络版本1

|--cnn\_v2.py cnn神经网络版本2

|--cnn\_v3.py cnn神经网络版本3

|--cnn\_v4.py cnn神经网络版本4

|--cnn.py cnn神经网络

|--bmp.py 数据集图片读取接口

|--img\_data.py 数据集接口

|--TRAIN/ 数据集

2、img\_data.py

img\_data内定义数据集类Data

|  |
| --- |
| class Data |
| attribute:  train 训练集数据  test 测试集数据 |
| method:  train\_set(data\_num) 随机选取data\_num个训练集数据  test\_set(data\_num) 随机选取data\_num个测试集数据 |

3、cnn神经网络各版本说明

cnn\_v1网络结构

inputs->conv->relu->pool->fc->relu->fc->softmax

cnn\_v2网络结构

inputs->conv->bn->relu->pool->fc->relu->fc->softmax

cnn\_v3网络结构

inputs->conv->relu->pool->fc(dropout)->relu->fc->softmax

cnn\_v4网络结构

inputs->conv->leaky\_relu->pool->fc->leaky\_relu->fc->softmax

cnn网络结构

inputs->(conv->relu->pool)x2->fc->relu->fc->softmax

二、神经网络结构改进过程及效果

1、最初使用cnn\_v1版本神经网络，结构为inputs -> conv -> relu -> pool -> fc -> relu -> fc -> softmax, 训练5000次得到识别准确率89.2%, 用时424s

二、对卷积神经网络的理解

经过了课程的学习后才知道深度学习，卷积神经网络并没有想象的那么神秘，卷积神经网络仍然还是层级网络结构，只不过是在原来的神经网络结构上进行一定的改进，从而达到更好的训练结果。总体上仍然是之前bp神经网络的思路，首先是正向传播计算，然后根据计算结果进行反向传播修正调整。

但是与之前不同的是，在卷积神经网络当中有很多神奇的操作，从而可以达到大大提升训练效果。首先就是卷积神经网络名号的由来--卷积层，通过通过多个卷积核对图像进行区域卷积操作，得到多个图像特征；经过激励函数输出后再进行池化，即对于不同区域只保留其某些特征值，如最大值或是均值，这一步很关键，通过池化可以大大降低输入数据集的数据量，但是却在最大程度上保留了图像的特称，从而为后来的softmax作准备。卷积池化的操作可以重复多次，然后通过全连接层及激励函数进行输出，最终通过softmax输出结果。经过这样的操作之后，得到的训练结果相比于原来的神经网络有了相当可观的提升。

三、卷积神经网络的优化过程

1、本次实验最初使用最基础网络结构：inputs -> conv -> relu -> pool -> fc -> relu -> fc -> softmax, 训练5000次后得到的识别准确率为89.2%，训练用时424s，虽然相比于原来的bp神经网络的训练结果有了一定的提升，但是提升并不够大，而且还没有达到卷积神经网络应有的性能。

2、进行的第一种改进方式为加入batch normalize操作，在卷积层输出后对输出结果进行bn操作，然后再正常进行relu激励函数操作，网络结构为inputs -> conv -> bn -> relu -> pool -> fc -> relu -> fc -> softmax。加入bn层之后，训练5000次，识别准确率为91%，用时562s，相比于基础版本的神经网络，训练结果有一定的提升。

3、第二种改进方式为使用dropout操作，在全连接层使用dropout操作，网络结构为inputs -> conv -> relu -> pool -> fc(dropout) -> relu -> fc -> softmax, 训练5000次后得到识别准确率91.8%，用时495s，相比于基础版也有进步。

4、第三种改进方式改变了激活函数，将原本的激活函数relu替换为leaky\_relu，网络结构为inputs -> conv -> leaky\_relu -> fc -> leaky\_relu -> fc -> softmax, 修改后训练5000次识别准确率为89.8%， 用时560s

5、第四种改进方式增加了一次卷积池化操作，网络结构为inputs -> (conv -> relu -> pool)x2 -> fc -> relu -> fc -> softmax,

6、综合上述优化方式，进行终极大优化，网络结构为inputs -> (conv -> bn -> leaky\_relu -> pool)x2 -> fc(drop\_out) -> leaky\_relu -> fc -> softmax

|  |  |  |
| --- | --- | --- |
| cnn 神经网络优化 | | |
| structure | accuracy | runtime |
| basal structure | 89.2% | 424s |
| use bn | 91% | 562s |
| use dropout | 91.8% | 495s |
| use leaky\_relu | 89.8% | 560s |
| use double (conv,bn,pool) |  |  |
| use all |  |  |

四、对于几种网络优化方式的理解

1、batch normalization, 了解到这种操作主要是为了解决输入值的分布不均匀问题，难以让模型稳定地学习到规律。为了将输入的分布固定下来，从而通过batch\_normal的操作，将分布固定在一个比较标准的输入，从而避免梯度消失梯度爆炸的出现，也在一定程度上增加了模型的泛化能力。

2、dropout, 即在每次训练时，随机将网络层当中的一部分但愿临时隐藏起来，然后再进行训练，下一次训练时再选择隐藏其他神经元。这样可以弱化各个特征之间过大的相互联系，从而能够在一定程度上避免过拟合问题。

3、leaky\_relu, 是relu的变体，主要是在取值为负时的差别。relu相当于将负值舍弃，只有在超过