cnn 实验文档

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_v5.py
|--cnn_v5.5.py
|--cnn_v6.py
|--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 神经网络各版本结构说明（训练采用随机选取样本进行训练）

	神经网络结构
cnn_v1	inputs->conv->relu->pool->fc->relu->fc->softmax
cnn_v2	inputs->conv->bn->relu->pool->fc->relu->fc->softmax
cnn_v3	inputs->conv->relu->pool->fc(dropout)->relu->fc->softmax
cnn_v4	inputs->conv->leaky_relu->pool->fc->leaky_relu->fc->softmax
cnn_v5	inputs->(conv->relu->pool)x2->fc->relu->fc->softmax
cnn_v5.5	inputs->(conv->relu->pool)x3->fc(dropout)->relu->fc->softmax
cnn_v6	inputs->(conv->bn->leaky_relu->pool)x2->fc(dropout)->relu->fc->softmax

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 91.3%，训练用时 514s，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 93.4%，用时 606s，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 94.6%，用时 569s，相比于基础版也有进步。

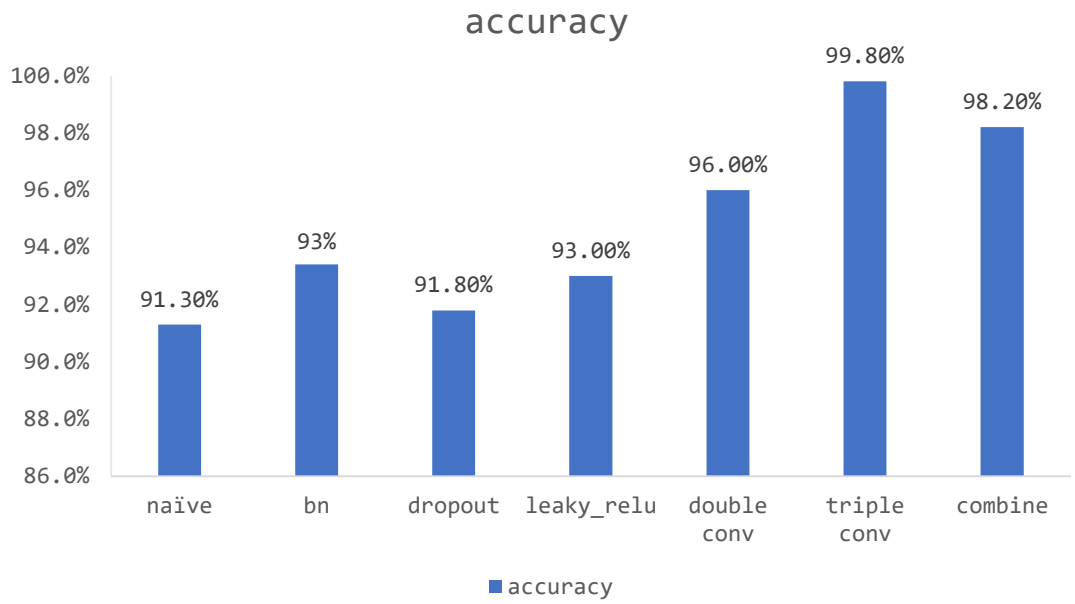
4、第三种改进方式改变了激活函数，将原本的激活函数 relu 替换为 leaky_relu，网络结构为 inputs -> conv -> leaky_relu -> fc -> leaky_relu -> fc -> softmax，修改后训练 5000 次识别准确率为 93%， 用时 575s

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，训练 5000 次得到准确率 96%，用时 806s；增加两次卷积池化操作，即三层卷积，训练 5000 次后得到最高准确率 99.8%，用时 695s，但训练 5000 次最终准确率仅为 6%，推断为神经网络结构复杂在迭代多次后导致梯度爆炸，模型坍塌，所以将该模型巡礼次数修改为 2000 次

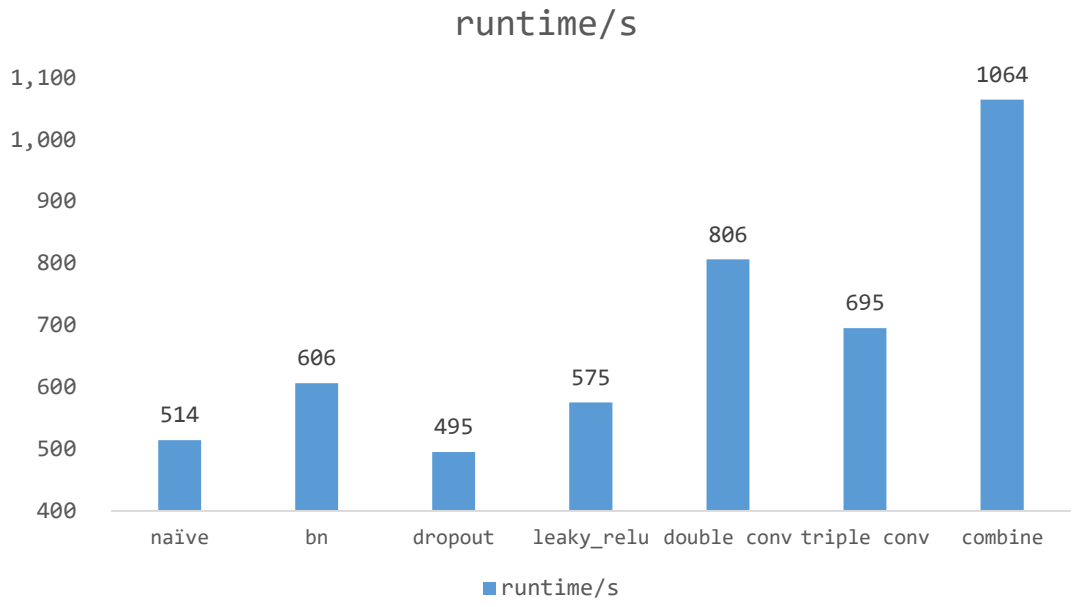
6、综合上述优化方式，进行终极大优化，网络结构为 inputs -> (conv -> bn -> leaky_relu -> pool)x2 -> fc(drop_out) -> leaky_relu -> fc -> softmax，经过该优化后，训练 5000 次准确率为 98.2%，用时 1064s

以下为实验数据（均以训练 5000 次为准）

cnn 神经网络优化		
structure	**accuracy**	**runtime**
naive (v1)	91.3%	514s
use bn (v2)	93.4%	606s
use dropout (v3)	91.8%	495s
use leaky_relu (v4)	93%	575s
use double (conv,bn,pool) (v5)	96%	806s
use triple conv (v5.5)	99.8%	695s
use all (v6)	98.2%	1064s



分析：通过准确率对比可以看出，几项措施均有优化效果，其中增加卷积层数的优化措施最为有效，得到了最高的性能提升，其中 dropout 的优化效果不明显，认为原因在于 dropout 更大层面是在于防止在复杂网络结构下出现过拟合现象，所以在网络结构较简单时优化效果并不明显。另外值得注意的是，虽然使用三层卷积网络得到了极大的优化，但是在训练中发现该神经网络结构并不稳定，有可能出现梯度爆炸导致模型坍塌的情况，而且随着训练次数的增加，模型不稳定性增加，所以对于该神经网络结构，将其训练次数定为 2000 次。



分析：总体可以看出，随着神经网络结构的复杂程度增加，训练相同次数的用时也在增加，而使用了 dropout 后，因为在训练过程中隐藏了部分神经元，减少了计算开销，所以反而用时减少是可以理解的。

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，只有在超过一定阈值之后，输出值才会作用于后面的网络，但在实际情况中，那些低于阈值的输入并不一定没有作用，所以 leaky_relu 对于负输入值采用一个较小的斜率，从而能够更好的保留输入数据的特征。

4、**多层卷积**，这个比较好理解，多层的神经网络也就使得模型的性能增强，但是也并不是越多越好，在使用的时候要注意过拟合的问题，可以使用 dropout 在一定程度上避免过拟合。

五、实验当中遇到的问题

1、框架的使用，本次实验使用了 tensorflow 框架，因为 tensorflow 的框架机制与大部分传统的框架存在一些差异，他将所有的任务迁移至一个高效的 c++后端，这样导致在实验过程中 debug 存在一些困难，不过随着框架的熟悉，以及对 tensorflow 报错信息的逐渐熟悉，这个问题也逐渐解决。

2、卷积神经网络的性能优化，自己最初编写的 cnn 版本性能并不理想，准确率还没有达到 90%，所以在优化上也下了一定的功夫，通过加入 bn 层，dropout 等操作，也成功地对神经网络的性能有了一定的优化。