

实验报告

神经网络构建与地物分类



学号：

日期：2017-06-16

目录

[一、实验目的 2](#_Toc485558447)

[二、实验任务 2](#_Toc485558448)

[三、实验内容 3](#_Toc485558449)

[3．1 分析CNN网络 3](#_Toc485558450)

[3．2 Dropout和Activation function的分析 5](#_Toc485558451)

[3．3 UCML数据集分类 7](#_Toc485558452)

[四、实验结果 9](#_Toc485558453)

[五、实验总结 10](#_Toc485558454)

[附件：文件清单 11](#_Toc485558455)

# 一、实验目的

2013年，《麻省理工学院技术评论》杂志将深度学习列为2012年十大突破性技术之首，其所带来的识别精度的跳跃甚至又重新引燃了人们对人工智能的热情，引发一拨工业应用热潮。卷积神经网络（CNN）作为其中典型代表，不管是计算机视觉（CV），还是自然语言理解（NLP），甚至是自动驾驶、AlphaGo等新兴技术都扮演了非常重要的角色。本作业旨在进行CNN的基础知识、训练技巧、实践应用的考察。

# 二、实验任务

任务1：学习CNN经典网络（如LeNet-5、AlexNet、VGG、ResNet等），尝试回答下面的问题：CNN通常由哪些结构组成，为什么要使用这些结构？CNN能成功应用到多个有明显差异的领域，是CNN中的哪些关键特性在起作用？请列举图表说明。

任务2：CNN从1989年提出伊始，历经十几年发展，有诸多实用性技巧不断完善优化，请从以下关键词中挑选2~3个展开实验：Dropout、Activation function、Batch normalization、Data augmentation、Weight decay等。实验要求在某一数据集上进行对比实验，分析原理，形成实验报告（可采用任务3中的数据集）。

任务3：地物分类作为遥感图像处理的经典应用，是地形信息快速提取、城市规划、环境监测、国土资源调查等应用的关键。请在CNN经典网络的基础上，结合任务2中的训练技巧，优化网络结构，实现样本切片的准确分类。要求：（1）采用UCML数据集，数据下载地址：http://vision.ucmerced.edu/datasets/landuse.html；（2）将训练数据、验证数据、测试数据按7：1：2比例随机划分，请重复划分3次以上；（3）统计在测试数据上的分类精度，一般应在90%以上（性能较佳者可达98%或更高）；（4）形成算法报告，并提交程序代码及相关文件。

# 三、实验内容

实验步骤依次按任务一、二、三展开。

## 3．1 分析CNN网络

LeNet5网络包含有卷积层、池化层、多层感知机（全连接层），AlexNet网络包含有卷积层、池化层、全连接层、激励层等，VGG也同样有这些结构，所以通过对比分析可知，CNN网络主要由卷积层、池化层、全连接层三大部分构成。卷积层用于学会识别输入数据的特性表征，相当于用不同的基地来对上一层的特征进行表达，它的运算过程如图3-1所示。

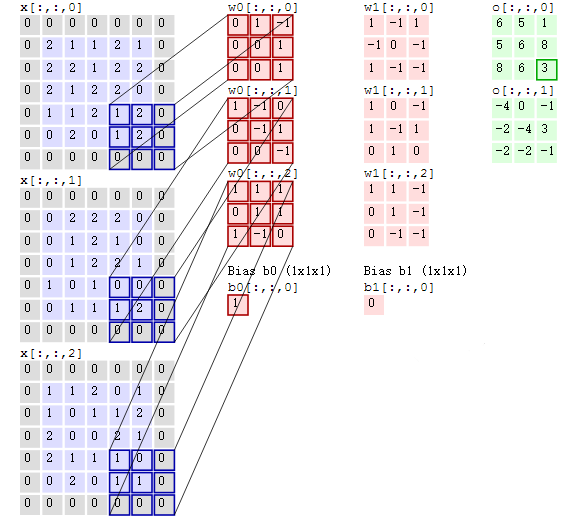


图3-1 CNN 卷积层的计算过程

池化层主要用于减少数据量，降低运算复杂度，典型的有平均池化和最大池化，最大池化的计算过程如图3-2所示。

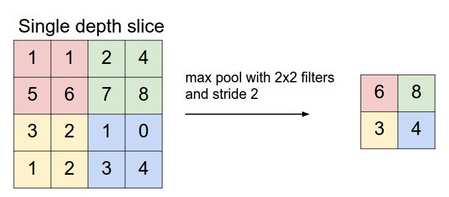


图3-2 最大池化的示意过程

全连接层中的神经元与上一层的神经都有相连，与一般的神经网络一样，可看成是多层感知机，只要用于网络的输出层和最后几层，其链接方式由图3-3所示。

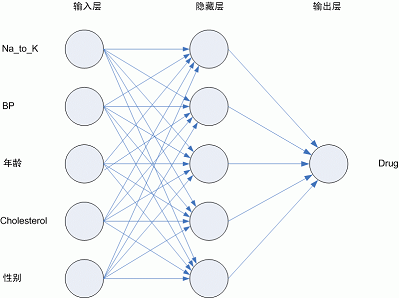


图3-3 全连接层

除去上述的部分，CNN网络中还有正则化层和激活层，其中激活层是对人体神经冲动传递的模拟，有利于加速收敛，并且非线性的激励函数可以更好地构建输入与输出之间的关系。

跟一般的神经网络相比，CNN网络主要加入了卷积层和池化层，其中池化层对于减少数据量和降低复杂度有一定的帮助，但更重要的是卷积层和权值共享的搭配使得需要计算的参数大大减少，这给人们使用CNN网络带来了信心，对于CNN的推广起到了很大的作用。

由上，我们可以将CNN网络的关键技术总结为局部感知，权共享和降采样，同时它也具有一般神经网络的优点，只要给入输入与输出，网络就会自动学习参数，这些特性使得CNN可以用在多个不同的领域，例如图像分类，物体检测，物体追踪，姿态预估，文本检测识别，视觉检测，行动识别，场景标记等等。

## 3．2 Dropout和Activation function的分析

工具：Python 3.5，TensorFlow 1.1.0

（1）大规模的神经网络有两个缺点：费时和容易过拟合。因为神经结构复杂，需要计算的参数繁多，所以特别耗时，同时结构的复杂性会使得训练结果更贴近训练集，而丧失普适性，即在其它数据集上的表现可能不佳。Dropout是缓解上述问题的一个技巧。

Dropout的主要作用为减缓模型过拟合。Dropout是指在模型训练时随机让网络某些隐含层节点的权重不工作，不工作的那些节点可以暂时认为不是网络结构的一部分，在该轮训练中保持其值不变。这就相当于每次的训练的是一个比完整网络瘦一些的网络。

首先，在mnist手写数字识别库上进行有无Dropout的对比实验，发现实验结果差别不大，这主要是因为mnist训练样本有6万，它有足够多的样本来支持网络的训练，而Dropout对于样本较少的训练的影响比较明显。所以在UCML数据集上进行测试，测试结果的对比如表3-1所示。

|  |  |  |
| --- | --- | --- |
|  | 无Dropout | Dropout = 0.5 |
| 测试集准确率 | 90.75% | 93.75% |
| 运行时间 | 107min | 81min |

表3-1 有无Dropout的实验对比

从表中可以看出使用Dropout后，实验时间有所减短，且测试集上的准确率有所提高，这说明Dropout在样本较少时能够减缓过拟合，提高测试精度，并减少训练时间。

（2）激励函数是用于模拟信号在人体神经中的传递机制，而且非线性的激励函数理论上可以逼近任何输入与输出的关系。Tensorflow中提供的有连续非线性的sigmoid, tanh, elu, softplus, softsign，连续但是并不是处处可微分的relu, relu6, crelu, relu\_x，现将sigmoid函数与relu函数进行对比。

从数学上来看，非线性的Sigmoid函数对中央区的信号增益较大，对两侧区的信号增益小，在信号的特征空间映射上，有很好的效果。其图形如图3-4所示。

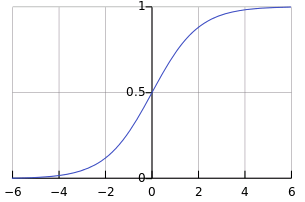


图3-4 Sigmoid函数

而Relu函数跟Sigmoid函数相比具有单侧抑制、相对宽阔的兴奋边界、稀疏激活性的特性，图形如图3-5所示。

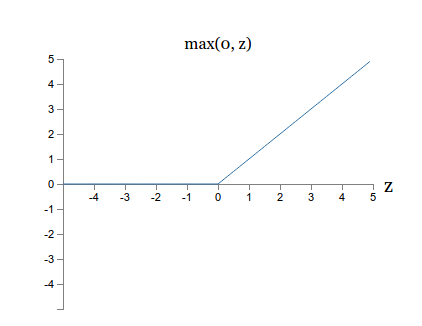


图3-5 Relu函数

在Mnist数据集上分别测试Sigmoid函数和Relu函数，在训练了6000次后测试集上的准确率分别是96.17%和98.65%。同时将两者的收敛情况进行对比，可得到如图3-6所示的结果。

图3-6 Sigmoid函数和Relu函数收敛情况对比

从图中可以看出，Relu函数在训练时准确率上升得更快，也就是说它的收敛速度更快，这和Relu函数本身的性质是相关的。而且Relu会使一些输出为0，造成了网络的稀疏性，缓解过拟合，这就可以解释为什么使用Relu函数的网络准确率要略高一点。

所以通过比较，Relu函数的性质比Sigmoid函数要好。我们也不难发现Relu函数使用得跟更广泛。

## 3．3 UCML数据集分类

工具：Python 3.5，TensorFlow 1.1.0

1. 数据集划分

为了将总的数据集随机划分为7：1：2的训练集、验证集和测试集，在Python里写了个小程序，利用shuffle函数将一个1到100的数组打乱重排，从而将每一个class里的数据随机的分为三份。同时使用PIL库将所有图片统一按比例压缩为227\*227的大小。

1. 生成TXT

为了方便Tensorflow读取数据，利用训练集、验证集和测试集的数据分别生成三个TXT文件，每个TXT文件中包含该数据集中的每张图片的路径和图片对应的分类标签，例如训练集train.txt的文件格式如图3-3所示。

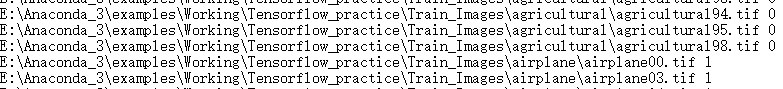


图3-7 train.txt示例

1. 使用AlexNet进行finetune

一开始我使用了三层卷积层和三层全连接层的网络对训练集进行了训练，在训练4000多次后可以在训练集上得到98%左右的准确率，然而在测试集上的准确率却只有50%左右，可见该网络严重过拟合，这跟训练样本只有1470条有很大的关系，所以我决定使用AlexNet网络进行微调，即在以训练好的模型上针对自己的数据进行调整。具体来说，先搭好AlexNet的结构，然后载入它的已经调好的参数作为初始参数，本次实验中对第七、八层重新训练。

AlexNet网络一共有8层，5层卷积层和3层全连接层，其具体的结构如图3-6所示。

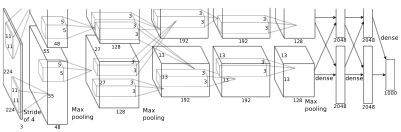


图3-8 AlexNet网络结构

AlexNet网络中的激励函数使用的是Relu函数，最后一层输出层使用softmax函数进行分类。由于电脑内存有限，可分块读入数据，块的大小定义为batch\_size,每一轮训练的次数为数据集除以batch\_size，通过调整训练轮数、学习率等，可以得到不同测试结果，具体结果由下一部分展示。

# 四、实验结果

（1）设置学习率learning\_rate = 0.01，训练轮数num\_epochs = 15，数据块大小batch\_size = 90以及dropout\_rate = 0.6，可得到如下结果：

2017-06-15 15:08:28.170954 Start training...

2017-06-15 15:08:28.170954 Open Tensorboard at --logdir finetune\_alexnet/CNN\_UCMerced\_LandUse

2017-06-15 15:08:28.170954 Epoch number: 1

2017-06-15 15:13:22.935941 Start test

2017-06-15 15:13:39.124273 Test Accuracy = 0.7333

2017-06-15 15:13:39.413904 Saving checkpoint of model...

2017-06-15 15:13:52.297920 Model checkpoint saved at finetune\_alexnet/model\_epoch1.ckpt

……

2017-06-15 16:01:05.856193 Epoch number: 15

2017-06-15 16:03:58.902714 Start test

2017-06-15 16:04:08.287843 Test Accuracy = 0.9333

2017-06-15 16:04:08.288844 Saving checkpoint of model...

2017-06-15 16:04:15.058079 Model checkpoint saved at finetune\_alexnet/model\_epoch15.ckpt

最终测试准确率为93.33%。

(2) 设置学习率learning\_rate = 0.01, 训练轮数num\_epochs = 20, 数据块大小batch\_size = 80以及dropout\_rate = 0.5，结果如下：

2017-06-15 18:41:39.788448 Start training...

2017-06-15 18:41:39.789449 Open Tensorboard at --logdir finetune\_alexnet/CNN\_UCMerced\_LandUse

2017-06-15 18:41:39.789449 Epoch number: 1

2017-06-15 18:45:15.539272 Start validation

2017-06-15 18:45:50.179589 Validation Accuracy = 0.8575

2017-06-15 18:45:50.190596 Saving checkpoint of model...

2017-06-15 18:45:57.477650 Model checkpoint saved at finetune\_alexnet/model\_epoch1.ckpt

……

2017-06-15 19:58:57.327208 Epoch number: 20

2017-06-15 20:02:16.881342 Start validation

2017-06-15 20:02:50.461290 Validation Accuracy = 0.9375

2017-06-15 20:02:50.462291 Saving checkpoint of model...

2017-06-15 20:02:57.443964 Model checkpoint saved at finetune\_alexnet/model\_epoch20.ckpt

最终测试准确率为93.75%。

# 五、实验总结

本次实验让我对CNN网络结构有了进一步的认识，通过在实际数据上的应用我掌握了一些实际操作的技巧，也遇到过一些问题，例如训练样本太少，结果过拟合严重，并通过与同学讨论和网上查阅资料解决了问题，对Tensorflow和Python的使用变得更加熟练了。本次实验中使用的是AlexNet模型，这是2012年就提出的结构，在这之后有更新更强大的网络提出，例如VGG，而因为时间问题本次只对一个网络进行了微调，有时间应该试试其它网络，对图像进行样本增加和增强等处理，这样能得到更高的分类精度。

# 附件：文件清单

AlexNet模型

数据读取与图像处理

微调程序

bvlc\_alexnet.npy Alexnet参数

随机生成训练集、验证集和测试集

生成TXT文件

Mnist数据集测试代码