随笔-141 文章-0 评论-17

Deep Learning模型之：CNN卷积神经网络（一）深度解析CNN

http://m.blog.csdn.net/blog/wu010555688/24487301

本文整理了网上几位大牛的博客，详细地讲解了CNN的基础结构与核心思想，欢迎交流。

[1]Deep learning简介

[2]Deep Learning训练过程

[3]Deep Learning模型之：CNN卷积神经网络推导和实现

[4]Deep Learning模型之：CNN的反向求导及练习

[5]Deep Learning模型之：CNN卷积神经网络（一）深度解析CNN

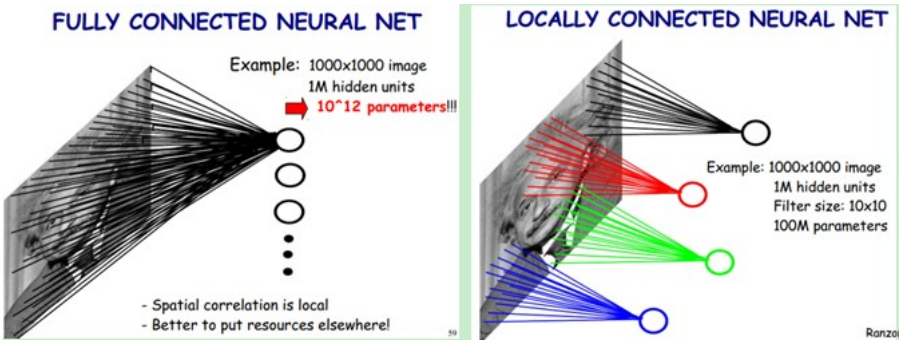
[6]Deep Learning模型之：CNN卷积神经网络（二）文字识别系统LeNet-5

[7]Deep Learning模型之：CNN卷积神经网络（三）CNN常见问题总结

1. 概述

卷积神经网络是一种特殊的深层的神经网络模型，它的特殊性体现在两个方面，一方面它的神经元间的连接是非全连接的， 另一方面同一层中某些神经元之间的连接的权重是共享的（即相同的）。它的非全连接和权值共享的网络结构使之更类似于生物 神经网络，降低了网络模型的复杂度（对于很难学习的深层结构来说，这是非常重要的），减少了权值的数量。

回想一下BP神经网络。BP网络每一层节点是一个线性的一维排列状态，层与层的网络节点之间是全连接的。这样设想一下，如果BP网络中层与层之间的节点连接不再是全连接，而是局部连接的。这样，就是一种最简单的一维卷积网络。如果我们把上述这个思路扩展到二维，这就是我们在大多数参考资料上看到的卷积神经网络。具体参看下图：



上图左：全连接网络。如果我们有1000x1000像素的图像，有1百万个隐层神经元，每个隐层神经元都连接图像的每一个像素点，就有1000x1000x1000000=10^12个连接，也就是10^12个权值参数。

上图右：局部连接网络，每一个节点与上层节点同位置附件10x10的窗口相连接，则1百万个隐层神经元就只有100w乘以100，即10^8个参数。其权值连接个数比原来减少了四个数量级。

根据BP网络信号前向传递过程，我们可以很容易计算网络节点的输出。例如，对于上图中被标注为红色节点的净输入，就等于所有与红线相连接的上一层神经元节点值与红色线表示的权值之积的累加。这样的计算过程，很多书上称其为卷积。

事实上，对于数字滤波而言，其滤波器的系数通常是对称的。否则，卷积的计算需要先反向对折，然后进行

2015年12月 (12)
2015年11月 (4)
2015年8月 (5)
2015年7月 (3)
2015年6月 (7)
2015年5月 (7)
2015年4月 (4)
2015年2月 (3)
2015年1月 (2)
2014年12月 (1)
2014年11月 (1)
2014年7月 (9)
2014年6月 (17)
2014年4月 (1)
2013年4月 (2)
2013年3月 (2)
2012年11月 (1)
2012年9月 (2)
2012年4月 (1)
2011年12月 (2)
2011年7月 (2)
2011年6月 (1)
2011年5月 (4)
2011年4月 (1)
2011年1月 (1)
2010年8月 (4)
2010年7月 (3)
2010年6月 (3)
2010年5月 (6)
2010年4月 (9)
2010年3月 (12)
2009年5月 (3)
2009年4月 (3)
2009年3月 (3)

文章分类(1)
MFC(1)
STL
架构
设计模式

最新评论
1. Re:android 下使用Direct Texture
你好，你有这方面的教程吗，对这方面不是非常懂。
你的意思是，将工程的So编译放到Android源码下编译是吗？那需要哪个Android版本呢？4.2可以吗？
--a483210
2. Re:android 下使用Direct Texture
@a483210放在android源码下去写make，单独生成so文件...
--莫小
3. Re:android 下使用Direct Texture
你好，如何才能引用GraphicBuffer.h呢，如果我从android源代码里拿出它，那么它会有一堆的引用，我感觉应该不可能引用所有的文件吧，难道真的需要完全打包源代码么？如果是这样，那应该用什么.....
--a483210

阅读排行榜
1. Python的线程池实现(12932)
2. Deep Learning模型之：CNN卷积神经网络（一）深度解析CNN(11750)

乘累加的计算。上述神经网络权值满足对称吗？我想答案是否定的！所以，上述称其为卷积运算，显然是有失偏颇的。但这并不重要，仅仅是一个名词称谓而已。只是，搞信号处理的人，在初次接触卷积神经网络的时候，带来了一些理解上的误区。

卷积神经网络另外一个特性是权值共享。例如，就上面右边那幅图来说，权值共享，也就是说所有的红色线标注的连接权值相同。这一点，初学者容易产生误解。

上面描述的只是单层网络结构，前A&T Shannon Lab 的 Yann LeCun等人据此提出了基于卷积神经网络的一个文字识别系统 LeNet-5。该系统90年代就被用于银行手写数字的识别。

2、 CNN的结构

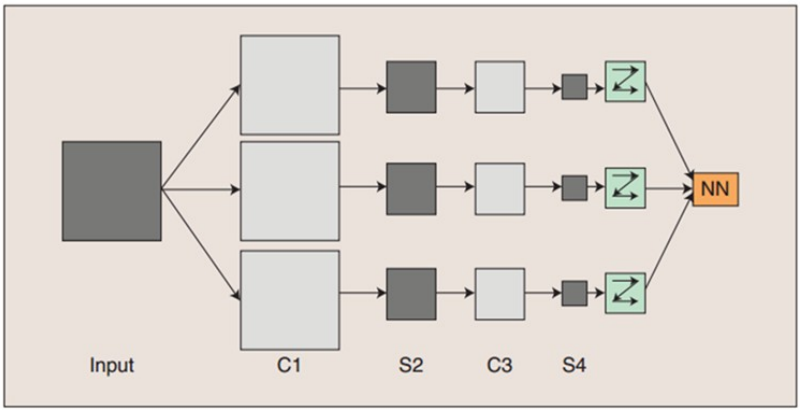
卷积网络是为识别二维形状而特殊设计的一个多层感知器，这种网络结构对平移、比例缩放、倾斜或者共他形式的变形具有高度不变性。 这些良好的性能是网络在有监督方式下学会的，网络的结构主要有稀疏连接和权值共享两个特点，包括如下形式的约束：

1、 特征提取。每一个神经元从上一层的局部接受域得到突触输人，因而迫使它提取局部特征。一旦一个特征被提取出来， 只要它相对于其他特征的位置被近似地保留下来，它的精确位置就变得没有那么重要了。

2 、特征映射。网络的每一个计算层都是由多个特征映射组成的，每个特征映射都是平面形式的。平面中单独的神经元在约束下共享 相同的突触权值集，这种结构形式具有如下的有益效果：a.平移不变性。b.自由参数数量的缩减(通过权值共享实现)。

3、子抽样。每个卷积层后面跟着一个实现局部平均和子抽样的计算层，由此特征映射的分辨率降低。这种操作具有使特征映射的输出对平移和其他 形式的变形的敏感度下降的作用。

卷积神经网络是一个多层的神经网络，每层由多个二维平面组成，而每个平面由多个独立神经元组成。



图：卷积神经网络的概念示范：输入图像通过和三个可训练的滤波器和可加偏置进行卷积，卷积后在C1层产生三个特征映射图，然后特征映射图中每组的四个像素再进行求和，加权值，加偏置，通过一个Sigmoid函数得到三个S2层的特征映射图。这些映射图再进过滤波得到C3层。这个层级结构再和S2一样产生S4。最终，这些像素值被光栅化，并连接成一个向量输入到传统的神经网络，得到输出。

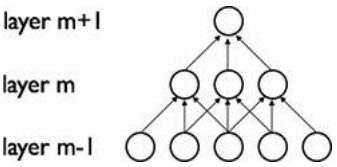
一般地，C层为特征提取层，每个神经元的输入与前一层的局部感受野相连，并提取该局部的特征，一旦该局部特征被提取后，它与其他特征间的位置关系也随之确定下来；S层是特征映射层，网络的每个计算层由多个特征映射组成，每个特征映射为一个平面，平面上所有神经元的权值相等。特征映射结构采用影响函数核小的sigmoid函数作为卷积网络的激活函数，使得特征映射具有位移不变性。

此外，由于一个映射面上的神经元共享权值，因而减少了网络自由参数的个数，降低了网络参数选择的复杂度。卷积神经网络中的每一个特征提取层（C-层）都紧跟着一个用来求局部平均与二次提取的计算层（S-层），这种特有的两次特征提取结构使网络在识别时对输入样本有较高的畸变容忍能力。

2.1 稀疏连接(Sparse Connectivity)

卷积网络通过在相邻两层之间强制使用局部连接模式来利用图像的空间局部特性，在第m层的隐层单元只与第m-1层的输入单元的局部区域有连接，第m-1层的这些局部 区域被称为空间连续的接受域。我们可以将这种结构描述如下：

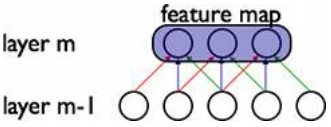
设第m-1层为视网膜输入层，第m层的接受域的宽度为3，也就是说该层的每个单元与且仅与输入层的3个相邻的神经元相连，第m层与第m+1层具有类似的链接规则，如下图所示。



可以看到m+1层的神经元相对于第m层的接受域的宽度也为3，但相对于输入层的接受域为5，这种结构将学习到的过滤器（对应于输入信号中被最大激活的单元）限制在局部空间 模式（因为每个单元对它接受域外的variation不做反应）。从上图也可以看出，多个这样的层堆叠起来后，会使得过滤器（不再是线性的）逐渐成为全局的（也就是覆盖到了更 大的视觉区域）。例如上图中第m+1层的神经元可以对宽度为5的输入进行一个非线性的特征编码。

2.2 权值共享(Shared Weights)

在卷积网络中，每个稀疏过滤器*hi*通过共享权值都会覆盖整个可视域，这些共享权值的单元构成一个特征映射，如下图所示。

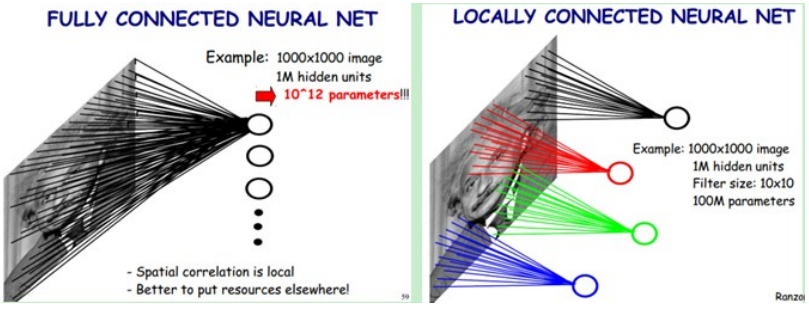


在图中，有3个隐层单元，他们属于同一个特征映射。同种颜色的链接的权值是相同的，我们仍然可以使用梯度下降的方法来学习这些权值，只需要对原始算法做一些小的改动， 这里共享权值的梯度是所有共享参数的梯度的总和。我们不禁会问为什么要权重共享呢？一方面，重复单元能够对特征进行识别，而不考虑它在可视域中的位置。另一方面，权值 共享使得我们能更有效的进行特征抽取，因为它极大的减少了需要学习的自由变量的个数。通过控制模型的规模，卷积网络对视觉问题可以具有很好的泛化能力。

举例讲解：

上面聊到，好像CNN一个牛逼的地方就在于通过感受野和权值共享减少了神经网络需要训练的参数的个数。那究竟是啥的呢？

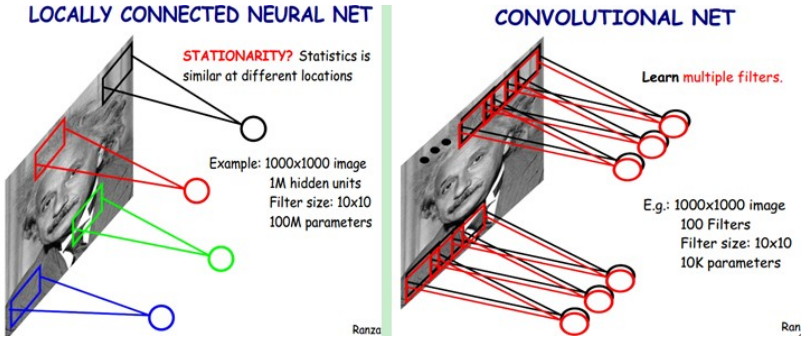
下图左：如果我们有1000x1000像素的图像，有1百万个隐层神经元，那么他们全连接的话（每个隐层神经元都连接图像的每一个像素点），就有1000x1000x1000000=10^12个连接，也就是10^12个权值参数。然而图像的空间联系是局部的，就像人是通过一个局部的感受野去感受外界图像一样，每一个神经元都不需要对全局图像做感受，每个神经元只感受局部的图像区域，然后在更高层，将这些感受不同局部的神经元综合起来就可以得到全局的信息了。这样，我们就可以减少连接的数目，也就是减少神经网络需要训练的权值参数的个数了。如下图右：假如局部感受野是10x10，隐层每个感受野只需要和这10x10的局部图像相连接，所以1百万个隐层神经元就只有一亿个连接，即10^8个参数。比原来减少了四个0（数量级），这样训练起来就没那么费力了，但还是感觉很多的啊，那还有啥办法没？



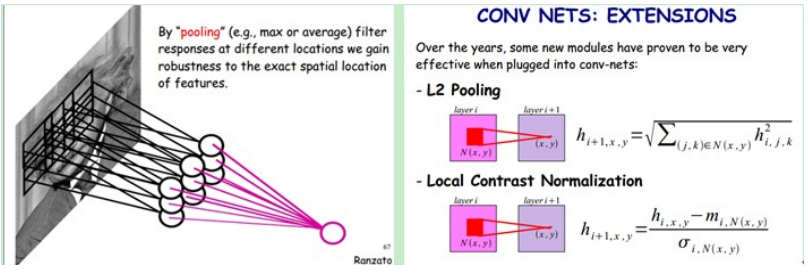
我们知道，隐含层的每一个神经元都连接10x10个图像区域，也就是说每一个神经元存在10x10=100个连接权值参数。那如果我们每个神经元这100个参数是相同的呢？也就是说每个神经元用的是同一个卷积核去卷积图像。这样我们就只有多少个参数？？只有100个参数啊！！！亲！不管你隐层的神经元个数有多少，两层间的连接我只有100个参数啊！亲！这就是权值共享啊！亲！这就是卷积神经网络的主打卖点啊！亲！（有点烦了，呵呵）也许你会问，这样做靠谱吗？为什么可行呢？这个.....共同学习。

好了，你就会想，这样提取特征也忒不靠谱吧，这样你只提取了一种特征啊？对了，真聪明，我们需要提取多种特征对不？假如一种滤波器，也就是一种卷积核就是提出图像的一种特征，例如某个方向的边缘。那么我们需要提取不同的特征，怎么办，加多几种滤波器不就行了吗？对了。所以假设我们加到100种滤波器，每种滤波器的参数不一样，表示它提出输入图像的不同特征，例如不同的边缘。这样每种滤波器去卷积图像就得到对图像的不同特征的放映，我们称之为Feature Map。所以100种卷积核就有100个Feature Map。这100个Feature Map就组成了一层神经元。到这个时候明了了吧。我们这一层有多少个参数了？100种卷积核x每种卷积核共享100个参数=100x100=10K，也就是1万个参数。才1万个参数啊！亲！（又来了，受不了了！）见下图右：不

同的颜色表达不同的滤波器。



嘿哟，遗漏一个问题了。刚才说隐层的参数个数和隐层的神经元个数无关，只和滤波器的大小和滤波器种类的多少有关。那么隐层的神经元个数怎么确定呢？它和原图像，也就是输入的大小（神经元个数）、滤波器的大小和滤波器在图像中的滑动步长都有关！例如，我的图像是1000x1000像素，而滤波器大小是10x10，假设滤波器没有重叠，也就是步长为10，这样隐层的神经元个数就是(1000x1000)/ (10x10)=100x100个神经元了，假设步长是8，也就是卷积核会重叠两个像素，那么.....我就不算了，思想懂了就好。注意了，这只是一种滤波器，也就是一个Feature Map的神经元个数哦，如果100个Feature Map就是100倍了。由此可见，图像越大，神经元个数和需要训练的权值参数个数的贫富差距就越大。



需要注意的一点是，上面的讨论都没有考虑每个神经元的偏置部分。所以权值个数需要加1 。这个也是同一种滤波器共享的。

总之，卷积网络的核心思想是将：局部感受野、权值共享（或者权值复制）以及时间或空间亚采样这三种结构思想结合起来获得了某种程度的位移、尺度、形变不变性。

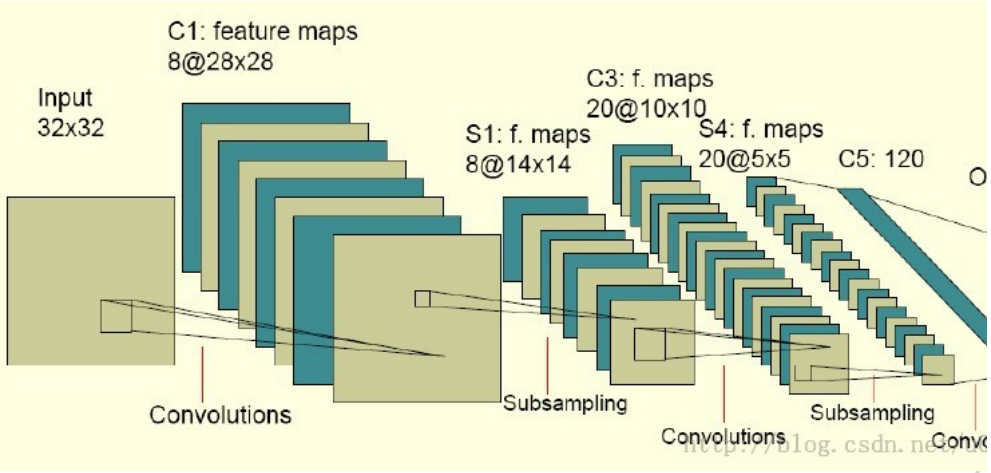
2.3 The Full Model

卷积神经网络是一个多层的神经网络，每层由多个二维平面组成，而每个平面由多个独立神经元组成。网络中包含一些简单元和复杂元，分别记为S-元 和C-元。S-元聚合在一起组成S-面，S-面聚合在一起组成S-层，用Us表示。C-元、C-面和C-层(Us)之间存在类似的关系。网络的任一中间级由S-层与C-层 串接而成，而输入级只含一层，它直接接受二维视觉模式，样本特征提取步骤已嵌入到卷积神经网络模型的互联结构中。

一般地，Us为特征提取层(子采样层)，每个神经元的输入与前一层的局部感受野相连，并提取该局部的特征，一旦该局部特征被提取后，它与其他特征间的位置关系 也随之确定下来；

Uc是特征映射层(卷积层)，网络的每个计算层由多个特征映射组成，每个特征映射为一个平面，平面上所有神经元的权值相等。特征映射结构采用 影响函数核小的sigmoid函数作为卷积网络的激活函数，使得特征映射具有位移不变性。此外，由于 一个映射面上的神经元共享权值，因而减少了网络自由参数的个数，降低了网络参数选择的复杂度。卷积神经网络中的每一个特征提取层(S-层)都紧跟着一个 用来求局部平均与二次提取的计算层(C-层)，这种特有的两次特征提取结构使网络在识别时对输入样本有较高的畸变容忍能力。

下图是一个卷积网络的实例，在博文"Deep Learning模型之：CNN卷积神经网络（二） 文字识别系统LeNet-5 "中有详细讲解：



图中的卷积网络工作流程如下，输入层由32×32个感知节点组成，接收原始图像。然后，计算流程在卷积和子抽样之间交替进行，如下所述：

第一隐藏层进行卷积，它由8个特征映射组成，每个特征映射由28×28个神经元组成，每个神经元指定一个 5×5 的接受域；

第二隐藏层实现子 抽样和局部平均，它同样由 8 个特征映射组成，但其每个特征映射由14×14 个神经元组成。每个神经元具有一个 2×2 的接受域，一个可训练 系数，一个可训练偏置和一个 sigmoid 激活函数。可训练系数和偏置控制神经元的操作点。

第三隐藏层进行第二次卷积，它由 20 个特征映射组 成，每个特征映射由 10×10 个神经元组成。该隐藏层中的每个神经元可能具有和下一个隐藏层几个特征映射相连的突触连接，它以与第一个卷积 层相似的方式操作。

第四个隐藏层进行第二次子抽样和局部平均汁算。它由 20 个特征映射组成，但每个特征映射由 5×5 个神经元组成，它以 与第一次抽样相似的方式操作。

第五个隐藏层实现卷积的最后阶段，它由 120 个神经元组成，每个神经元指定一个 5×5 的接受域。

最后是个全 连接层，得到输出向量。

相继的计算层在卷积和抽样之间的连续交替，我们得到一个"双尖塔"的效果，也就是在每个卷积或抽样层，随着空 间分辨率下降，与相应的前一层相比特征映射的数量增加。卷积之后进行子抽样的思想是受到动物视觉系统中的"简单的"细胞后面跟着"复杂的"细胞的想法的启发而产生的。

图中所示的多层感知器包含近似 100000 个突触连接，但只有大约2600 个自由参数(每个特征映射为一个平面，平面上所有神经元的权值相等)。自由参数在数量上显著地减少是通过权值共享获得 的，学习机器的能力（以 VC 维的形式度量）因而下降，这又提高它的泛化能力。而且它对自由参数的调整通过反向传播学习的随机形式来实 现。另一个显著的特点是使用权值共享使得以并行形式实现卷积网络变得可能。这是卷积网络对全连接的多层感知器而言的另一个优点。

3、 CNN的训练

神经网络用于模式识别的主流是有指导学习网络，无指导学习网络更多的是用于聚类分析。对于有指导的模式识别，由于任一样本的类别是已知的，样本在空间的分布不再是依据其自然分布倾向来划分，而是要根据同类样本在空间的分布及不同类样本之间的分离程度找一种适当的空间划分方法，或者找到一个分类边界，使得不同类样本分别位于不同的区域内。这就需要一个长时间且复杂的学习过程，不断调整用以划分样本空间的分类边界的位置，使尽可能少的样本被划分到非同类区域中。

卷积网络在本质上是一种输入到输出的映射，它能够学习大量的输入与输出之间的映射关系，而不需要任何输入和输出之间的精确的数学表达式，只要用已知的模式对卷积网络加以训练，网络就具有输入输出对之间的映射能力。卷积网络执行的是有导师训练，所以其样本集是由形如：（输入向量，理想输出向量）的向量对构成的。所有这些向量对，都应该是来源于网络即将模拟的系统的实际"运行"结果。它们可以是从实际运行系统中采集来的。在开始训练前，所有的权都应该用一些不同的小随机数进行初始化。"小随机数"用来保证网络不会因权值过大而进入饱和状态，从而导致训练失败；"不同"用来保证网络可以正常地学习。实际上，如果用相同的数去初始化权矩阵，则网络无能力学习。

训练算法与传统的BP算法差不多。主要包括4步，这4步被分为两个阶段：

第一阶段，向前传播阶段：

a）从样本集中取一个样本(X,Y_p)，将X输入网络；

b）计算相应的实际输出O_p。

在此阶段，信息从输入层经过逐级的变换，传送到输出层。这个过程也是网络在完成训练后正常运行时执行的过程。在此过程中，网络执行的是计算（实际上就是输入与每层的权值矩阵相点乘，得到最后的输出结果）：

$$O_p=F_n(\ldots(F_2(F_1(X_pW^{(1)})W^{(2)})\ldots)W^{(n)})$$

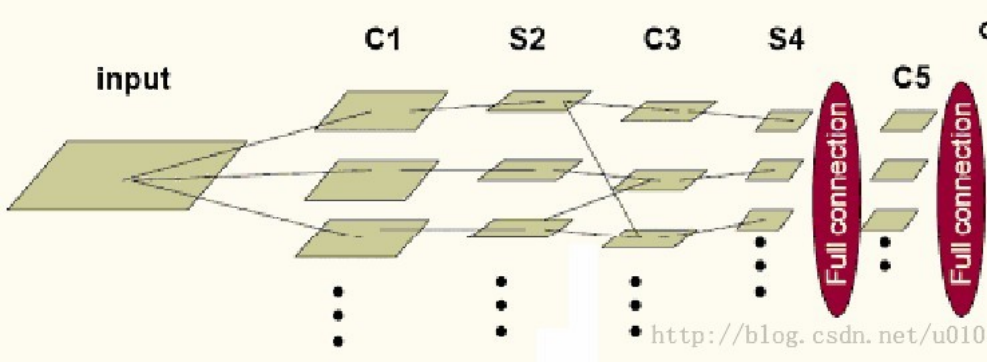
第二阶段，向后传播阶段

a）算实际输出O_p与相应的理想输出Y_p的差；

b）按极小化误差的方法反向传播调整权矩阵。

4、 CNN的学习

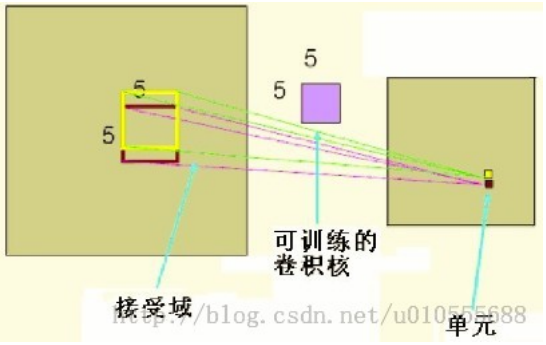
总体而言，卷积网络可以简化为下图所示模型：



其中，input 到C1、S4到C5、C5到output是全连接，C1到S2、C3到S4是一一对应的连接，S2到C3为了消除网络对称性，去掉了一部分连接， 可以让特征映射更具多样性。需要注意的是 C5 卷积核的尺寸要和 S4 的输出相同，只有这样才能保证输出是一维向量。

4.1 卷积层的学习

卷积层的典型结构如下图所示：



卷积层的前馈运算是通过如下算法实现的：

卷积层的输出= Sigmoid(Sum(卷积) +偏移量)

其中卷积核和偏移量都是可训练的。下面是其核心代码：

```
ConvolutionLayer::fprop(input,output) {
  //取得卷积核的个数
  int n=kernel.GetDim(0);
  for (int i=0;i<n;i++) {
      //第i个卷积核对应输入层第a个特征映射，输出层的第b个特征映射
      //这个卷积核可以形象的看作是从输入层第a个特征映射到输出层的第b个特征映射的一个链接
      int a=table[i][0], b=table[i][1];
      //用第i个卷积核和输入层第a个特征映射做卷积
```

```
        convolution = Conv(input[a],kernel[i]);
        //把卷积结果求和
        sum[b] +=convolution;
    }
    for (i=0;i<(int)bias.size();i++) {
        //加上偏移量
        sum[i] += bias[i];
    }
    //调用Sigmoid函数
    output = Sigmoid(sum);
}
```

其中，input是 n1×n2×n3 的矩阵，n1是输入层特征映射的个数，n2是输入层特征映射的宽度，n3是输入层特征映射的高度。output, sum, convolution, bias是n1×(n2-kw+1)×(n3-kh+1)的矩阵，kw,kh是卷积核的宽度高度(图中是5×5)。kernel是卷积核矩阵。table是连接表，即如果第a输入和第b个输出之间 有连接，table里就会有[a,b]这一项，而且每个连接都对应一个卷积核。

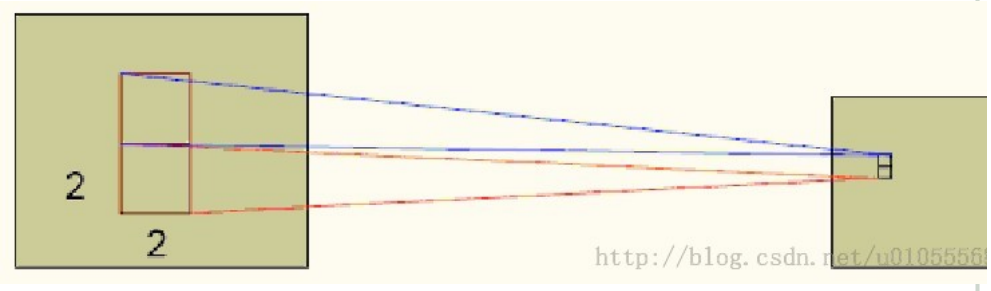
卷积层的反馈运算的核心代码如下：

```
ConvolutionLayer::bprop(input,output,in_dx,out_dx) {
    //梯度通过DSigmoid反传
    sum_dx = DSigmoid(out_dx);
    //计算bias的梯度
    for (i=0;i<bias.size();i++)  {
        bias_dx[i] = sum_dx[i];
    }
    //取得卷积核的个数
    int n=kernel.GetDim(0);
    for (int i=0;i<n;i++)
    {
        int a=table[i][0],b=table[i][1];
        //用第i个卷积核和第b个输出层反向卷积（即输出层的一点乘卷积模板返回给输入层），并把结果累加到第a个输入层
        input_dx[a] += DConv(sum_dx[b],kernel[i]);
        //用同样的方法计算卷积模板的梯度
        kernel_dx[i] += DConv(sum_dx[b],input[a]);
    }
}
```

其中in_dx,out_dx 的结构和 input,output 相同，代表的是相应点的梯度。

4.2 子采样层的学习

子采样层的典型结构如下图所示：



类似的子采样层的输出的计算式为：

输出= **Sigmoid(** 采样*权重 +偏移量**)**

其核心代码如下：

```
SubSamplingLayer::fprop(input,output) {
  int n1= input.GetDim(0);
  int n2= input.GetDim(1);
  int n3= input.GetDim(2);
  for (int i=0;i<n1;i++) {
      for (int j=0;j<n2;j++) {
          for (int k=0;k<n3;k++) {
              //coeff 是可训练的权重，sw 、sh 是采样窗口的尺寸。
              sub[i][j/sw][k/sh] += input[i][j][k]*coeff[i];
          }
      }
  }
  for (i=0;i<n1;i++) {
      //加上偏移量
      sum[i] = sub[i] + bias[i];
  }
  output = Sigmoid(sum);
}
```

子采样层的反馈运算的核心代码如下：

```
SubSamplingLayer::bprop(input,output,in_dx,out_dx) {
  //梯度通过DSigmoid反传
  sum_dx = DSigmoid(out_dx);
  //计算bias和coeff的梯度
  for (i=0;i<n1;i++) {
      coeff_dx[i] = 0;
      bias_dx[i] = 0;
      for (j=0;j<n2/sw;j++)
          for (k=0;k<n3/sh;k++) {
              coeff_dx[i] += sub[j][k]*sum_dx[i][j][k];
              bias_dx[i] += sum_dx[i][j][k]);
          }
  }
  for (i=0;i<n1;i++) {
      for (j=0;j<n2;j++)
          for (k=0;k<n3;k++) {
              in_dx[i][j][k] = coeff[i]*sum_dx[i][j/sw][k/sh];
          }
  }
}
```

5、CNN的优点

卷积神经网络CNN主要用来识别位移、缩放及其他形式扭曲不变性的二维图形。由于CNN的特征检测层通过训练数据进行学习，所以在使用CNN时，避免了显式的特征抽取，而隐式地从训练数据中进行学习；再者由于同一特征映射面上的神经元权值相同，所以网络可以并行学习，这也是卷积网络相对于神经元彼此相连网络的一大优势。卷积神经网络以其局部权值共享的特殊结构在语音识别和图像处理方面有着独特的优越性，其布局更接近于实际的生物神经网络，权值共享降低了网络的复杂性，特别是多维输入向量的图像可以直接输入网络这一特点避免了特征提取和分类过程中数据重建的复杂度。

流的分类方式几乎都是基于统计特征的，这就意味着在进行分辨前必须提取某些特征。然而，显式的特征提取并不容易，在一些应用问题中也并非总是可靠的。卷积神经网络，它避免了显式的特征取样，隐式地从训练数据中进行学习。这使得卷积神经网络明显有别于其他基于神经网络的分类器，通过结构重组和减少权值将特征提取功能融合进多层感知器。它可以直接处理灰度图片，能够直接用于处理基于图像的分类。

卷积网络较一般神经网络在图像处理方面有如下优点： a）输入图像和网络的拓扑结构能很好的吻合；b）特征提取和模式分类同时进行，并同时在训练中产生；c）权重共享可以减少网络的训练参数，使神经网络结构变得更简单，适应性更强。

6、CNN的实现问题

CNNs中这种层间联系和空域信息的紧密关系，使其适于图像处理和理解。而且，其在自动提取图像的显著特征方面还表现出了比较优的性能。在一些例子当中，Gabor滤波器已经被使用在一个初始化预处理的步骤中，以达到模拟人类视觉系统对视觉刺激的响应。在目前大部分的工作中，研究者将CNNs应用到了多种机器学习问题中，包括人脸识别，文档分析和语言检测等。为了达到寻找视频中帧与帧之间的相干性的目的，目前CNNs通过一个时间相干性去训练，但这个不是CNNs特有的。

由于卷积神经网络采用BP网络相同的算法。所以，采用现有BP网络就可以实现。开源的神经网络代码FAAN可以利用。这个开源的实现采用了一些代码优化技术，有双精度，单精度，定点运算三个不同的版本。

由于经典的BP网络是一个一维节点分布排列，而卷积神经网络是二维网络结构。所以，要把卷积神经网络的每一层，按照一定的顺序和规则映射为一维节点分布，然后，按照这个分布创建一个多层反向传播算法的网络结构，就可以按照一般的BP训练算法去学习网络参数。对于实际环境中新样本的预测，也采用BP算法中相同信号前向传递算法进行。具体细节也可以参考网上的一个开源代码，链接如下：
http://www.codeproject.com/Articles/16650/Neural-Network-for-Recognition-of-Handwritten-Digi
注：这个代码在创建CNN的时候有个明显的BUG，如果你看明白了我上面对简化的LeNet-5的结构描述，一眼就会找出问题所在。

7、几点困惑

这篇文章讲解的CNN构建过程，稍微详细点，可是还有几点我没有搞清楚：
<1>局部滑动窗的大小决定什么？
<2>滑动窗的位置怎么决定？
<3>滑动窗的位置决定什么？C层大小？
<4>“根据对前面C1层同样的理解，我们很容易得到C3层的大小为10x10。”不明白怎么得到
<5>“C3层的变成了16个10x10网络!”怎么变成了16个？
<6>第三部分讲 简化的LeNet-5系统，完全不明白节点与权值的计算。

希望能遇到比较精通CNN的朋友稍作讲解，感激不尽！

References:

卷积神经网络（**CNN**）

GitHub卷及神经网络代码（**MATLAB**）

CNN代码理解（**matlab**）

http://blog.csdn.net/nan355655600/article/details/17690029

http://blog.csdn.net/zouxy09/article/details/8782018

分类: 机器视觉

莫小
关注 - 0
粉丝 - 49
+加关注

« 上一篇：技术向：一文读懂卷积神经网络CNN
» 下一篇：GIST特征描述符使用

posted @ 2015-06-09 00:35 莫小 阅读(11752) 评论(1) 编辑 收藏

评论列表

#1楼 2015-12-26 21:01 fengenxiang1123

谢谢分享

支持(0) 反对(0)