CNN(convolutional neural networks)是Deep Learning在计算机视觉领域的一个成功模型结构，当然不止计算机视觉，语音识别或者文本识别等领域也可以应用，但在计算机视觉领域应用最为广泛，也更为成熟。本文尝试从简单的图像处理方法入手，进而引出CNN的结构和处理方法，最后探讨下谷歌的Alpha-Go是如何采用CNN结构进行训练的。

1. **为什么要用CNN做图像处理？**

用Deep Learning来做图片识别最简单的办法就是用一个全连接的神经网络做模型，如图1所示，然后训练和优化。



图1 全连接神经网络做图像识别示意图

假设图1中的输入图片猫大小为100x100，转化为一个输入向量则有30000维（彩色图片RGB表示3个channel），假设第一层Hidden Layer有1000个neural，则第一层Hidden Layer的总参数为3千多万个，而实际上稍微高清点的图片远大于100x100，同时Hidden Layer的层数也比较多，所以用全连接的神经网络做模型最大的问题就参数数量太多，对参数训练来说太困难。CNN做的处理就是根据人们对图像处理的理解或者说对图像处理所具有的一些特点来简化神经网络的结构，减少参数。

图像处理所具有的一些特点，如图2和图3所示：

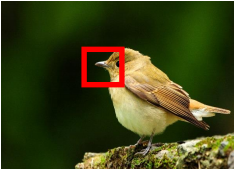


图2 鸟嘴识别

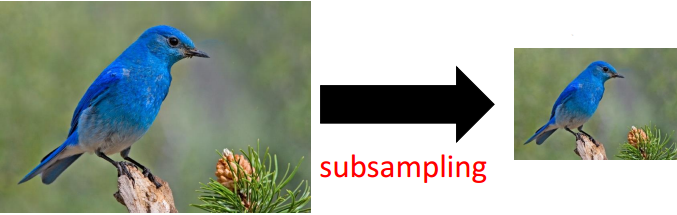


图3 改变图片大小

1. **某些图案要远小于整张图片的大小**；
2. **相同的图案可能出现在图片的不同位置**；

3）**对图片做sub-sampling后，不会改变图片中的目标。**

如要识别图2中的鸟嘴，只需要识别红框中的图案就好了；相同的识别目标“鸟嘴”可能位于相同大小图片中的不同位置（如图2中的2幅图）；如图3做下采样后“鸟”这个目标还是显而易见的。

1. **如何用CNN做图像处理？**

**（1）传统图像处理方式**

为方便理解和计算，我们假设输入图片为6x6的灰度图片，如图4所示：



图4 6x6灰度值图片 图5(a) Filter1 图5(b) Filter2

1）当我们想要识别图4中红框图案时，可以用图5（a）这个Filter，与原图片做卷积就可以识别出是否有目标图案存在；想要识别图4中绿框图案时，用5（b）这个Filter与原图片做卷积就可识别出是否有目标图案存在。

2）同时对于图片中不同位置的相同图案（如红框中的对角1，在图片左下角也存在），用同一个Filter可以识别。



图6 用Filter对图片做卷积，提取特征

1. 如图6，从左上角开始用Filter做卷积得到值3，然后右移一个单位（stride=1，移位大小可以设定）再卷积，直到移位到图片右下角得到卷积值-1。很明显，图6中的Filter作用是检测看图片中有没有斜对角为1的特征，结果可以看出当卷积为3时，就代表该特征出现，即图片中左上角和左下角有该特征，这就是特点2中所说的：相同的图案可能出现在图片的不同位置。

每个Filter对图片卷积操作后得到4x4 image就叫做Feature Map，如图7所示。



图7 卷积操作示意图

上面描述了灰度图片的处理，对于彩色图片的处理基本是一样的，不同点在于彩色图片有3个channel，RGB表示其颜色，大小为6x6x3，所以我们的Filter高度也变成了3，再做卷积（内积），如图8所示。



图8 彩色图片的卷积操作

**（2）神经网络做卷积**

实际上，对一张图片做卷积操作可以用一个全连接的前馈神经网络来实现，不同的是很多参数（weight）是0，而且很多weight是可以共享的；这样就大大简化了参数的数量。利用NN处理卷积操作如图9所示。明显的：1）参数大大减小，数量与Filter大小相关；2）Filter在图片上的移动就相当于相同的weight在输入向量不同的元素上的移动，这个就是weight共享（颜色相同的箭头代表权重相同），这样参数就更少了。



图9 神经网络的卷积操作

**（3）Max-Pooling**

原始image经过卷积之后得到的是size小一些的image，也称为Feature Map；对Feature Map做Max-Pooling本质上就是利用了图像处理中的第3个特点“**对图片做sub-sampling后，不会改变图片中的目标**”。



图10 Map-Pooling操作

1. **CNN网络结构**

经过卷积和Pooling（本文示例中用的是Max-Pooling，实际上Pooling还可以是别的类型，比如求均值等），形成了新的image，最后将新的image打平（Flatten），在将打平的数据丢进一个全连接的前向神经网络做训练，最后得到结果。所以CNN的结构就是：

1. Convolution & Pooling；这两个模块可以有多个，得到的结果是一个新的图片；
2. 将经过多次Convolution & Pooling后的new image打平（Flatten）；
3. 再将打平的数据丢进Network做训练。

这个过程中需要学习的weight主要有Filter的值和Network的参数，其中Filter的大小和个数是认为指定的。



图11 CNN网络结构

1. **CNN在学习什么？**

从前面示例中给出的2个filter可以很容易的知道：

1. 第一层的filter是在学习图像中很基础的特征（斜线、竖线、横线等）；
2. 第二层的filter是在第一层学习到的特征基础上再提取出更抽象的特征，同时这个更抽象的特征也比第一层的filter看到的图像范围更大，因为第二次的Convolution的image是经过sub-sampling的；
3. 经过多次特征抽象，最后得到的New image是一个高度抽象的特征；将其Flatten后放入前馈网络做分类训练。

上述就是CNN在做的事情，特性学习如图12所示：



图12 CNN特征学习

1. **CNN在Alpha-Go中的应用**

能否应用CNN结构来做Deep Learning主要取决于该应用是否有满足类似图像处理中的三个特点，如果把一个围棋棋盘看做一副大小为19x19的图像，1表示黑子，-1表示白子，0表示无子。



图13 围棋图像

如图13所示，该图案可以出现在围棋棋盘中的不同位置，明显对前2个特点，围棋图像是满足的，但是对与第三个sub-sampling，围棋图像就不能满足了，因为19x19的大小是不能变的，否则就不是围棋了。那Alpha-Go中CNN架构是如何处理的呢？图14是谷歌发表的关于Alpha-Go的论文《Mastering the game of Go with deep neural networks and tree search》中关于Alpha-Go中CNN架构的描述：



图14 Alpha-Go中的CNN架构

从论文中可知：

1. 输入是19x19x48的image，其中19x19是棋盘大小，而对棋盘中的每个位置Alpha-Go用一个48维的向量来描述该位置的状态；
2. 第一个Layer做了zero-padding，即将19x19的图像大小补零为23x23的图像，然后用5x5的filter做卷积，filter个数为k，k=192，stride=1，激活函数用的ReLU；
3. 后续的Hidden Layer的输入都是将前一层的输出（卷积后的图像）补零为大小21x21的image，再输入；同时Hidden Layer中使用的都是3x3的filter，stride=1。

所以Alpha-Go中是没有Pooling这个架构的，这说明CNN的架构并不是唯一的，需要根据不同的应用特点去设计不同的架构，当然Alpha-Go除了CNN架构，还用到了其他技术，比如增强学习等。

到此，CNN的经典架构以及在Alpha-Go中的应用就介绍完了。



2017.09.09.