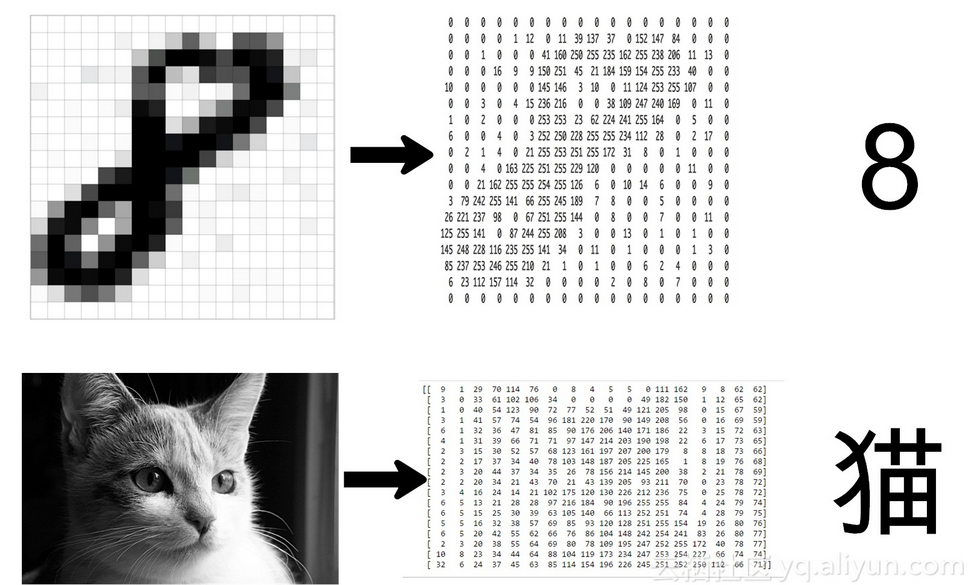
<https://yq.aliyun.com/articles/156269?spm=a2c4e.11153940.0.0.3a5073dds9Wdnu>

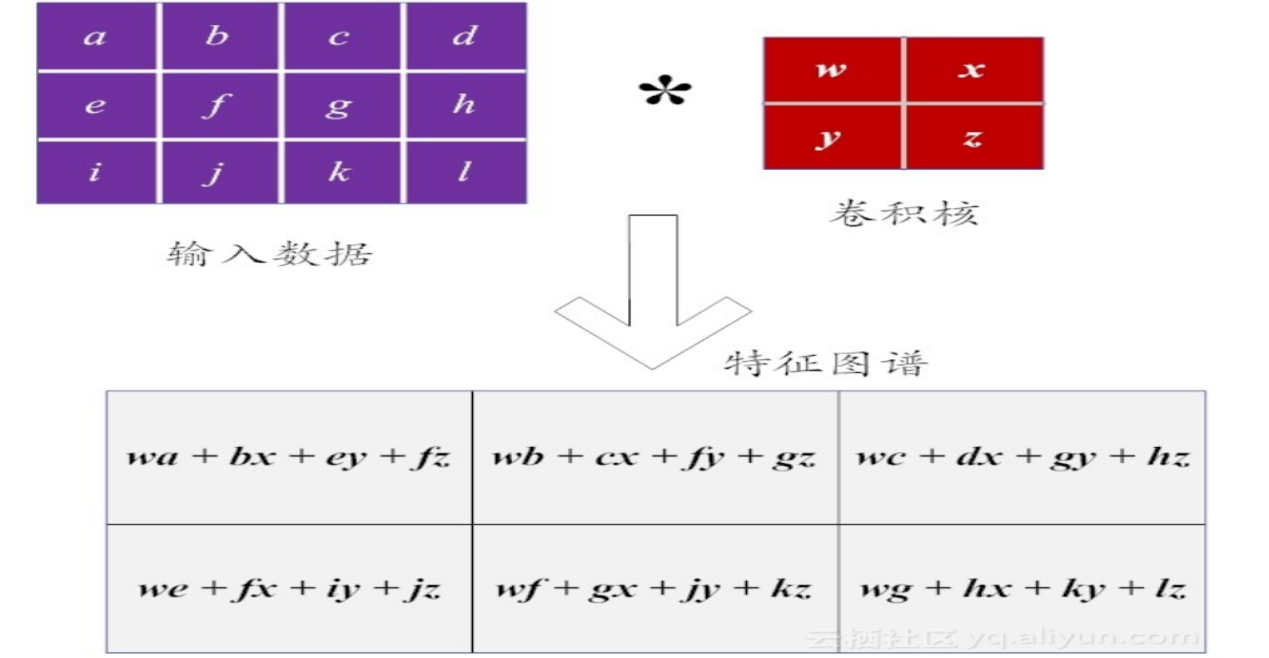
老哥说的非常详尽了

卷积神经网络主要的应用在CV领域，我们先来观察一下图片这种数据的具体数据形态

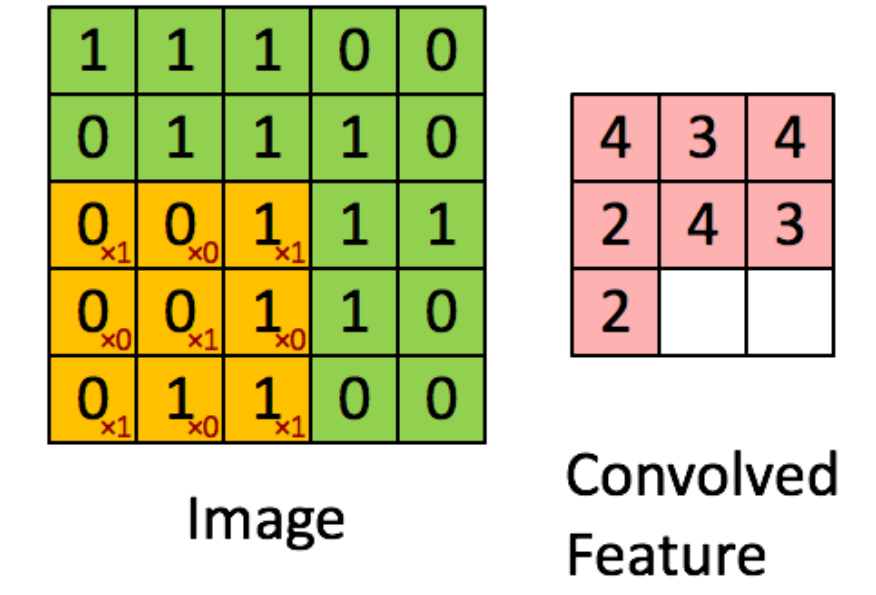
 数字矩阵（每个元素都是0到255之间像素值）

对于黑白图片来说，0代表黑色，255代表白色 像素值越小颜色越深。而对于彩色图片来说，一般有三个通道RGB（红，绿，蓝）。把三个不同通道的像素矩阵堆叠在一起，即可描述彩色图像。在图像处理中应用卷积操作，主要目的就是从图像中提取特征。卷积可以很方便地通过从输入的一小块数据矩阵（也就是一小块图像）中学到图像的特征，并能保留像素间的空间关系。

下面举例说明在二维图像使用卷积过程。

图像数据矩阵的像素值分别用诸如a-b-c-d这样的字母代替，卷积核是一个2×2的小矩阵。需要注意的是，在其他场合，这个小矩阵，也被称为“滤波器（filter）”或“特征检测器（feature detector）”。

在本质上，离散卷积就是一个线性运算。因此，这样的卷积操作也被称为线性滤波。这里的“线性”是指，我们用每个像素的邻域的线性组合来代替这个像素。其实，卷积操作还具有平移不变性（shift-invariant）。这个“平移不变性”是指，在图像的每个位置都执行相同的操作。



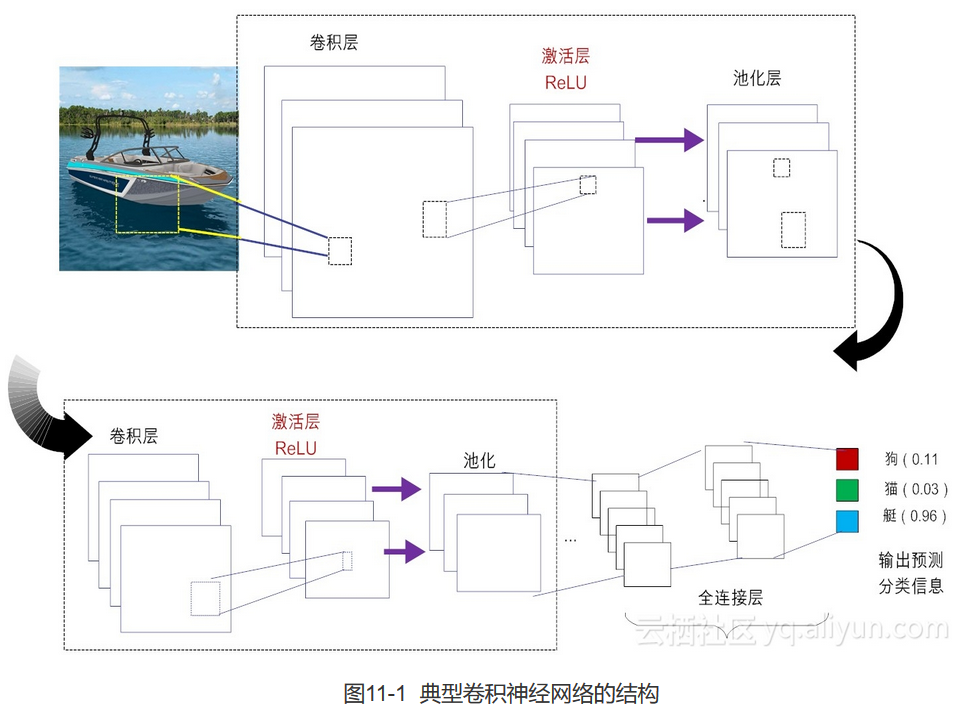
卷积操作在图像上作用之后具体的表现





从上图可以看出，不同的卷积核对图片的特征提取偏好不同，但是可以肯定的是图片数字矩阵元素之间的某些固定的线性组合作用之后，得到的卷积图像不仅仅保留了原图像主要的特征，还凸显了其某些特定方面的特征。

# 一个典型卷积神经网络的结构



**卷积层：**这个是卷积神经网络的核心所在。在卷积层，通过实现“局部感知”和“权值共享”等系列的设计理念，可达到两个重要的目的：

1.对高维输入数据实施降维处理和

2.实现自动提取原始数据的核心特征。

**激活层：**其作用是将前一层的线性输出，通过非线性激活函数处理，从而可模拟任意函数，进而增强网络的表征能力。在深度学习领域，ReLU（Rectified-Linear Unit，修正线性单元）是目前使用较多的激活函数，原因是它收敛更快，且不会产生梯度消失问题。

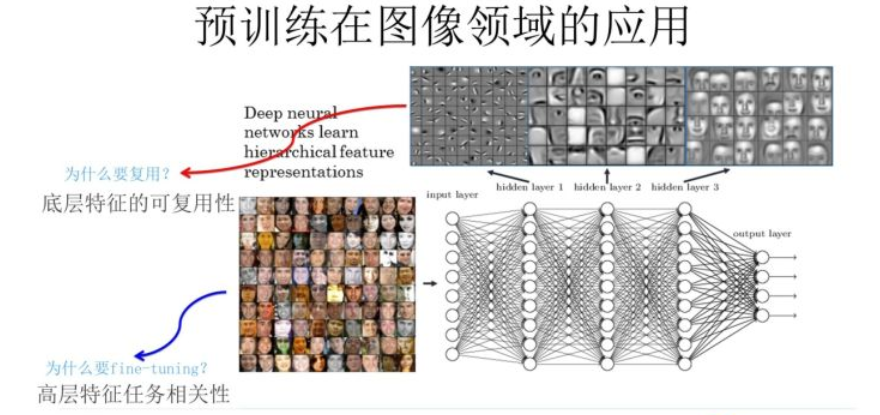
**池化层：**亦称亚采样层（Subsampling Layer）。简单来说，利用局部相关性，“采样”在较少数据规模的同时保留了有用信息。巧妙的采样还具备局部线性转换不变性，从而增强卷积神经网络的泛化处理能力。

**全连接层：**这个网络层相当于传统的多层感知机（Multi-Layer Perceptron，简称MLP）。通常来说，“卷积-激活-池化”是一个基本的处理栈，通过多个前栈处理之后，待处理的数据特性已有了显著变化：

一方面，输入数据的维度已下降到可用“全连接”网络来处理了；

另一方面，此时全连接层的输入数据已不再是“泥沙俱下、鱼龙混杂”，而是经过反复提纯过的结果，因此最后输出的结果要可控得高。

总体来讲，卷积神经网络通过多层处理，逐渐将初始的“低层”特征表示，转化为“高层”特征表示，之后再用“简单模型”即可完成复杂的分类等学习任务。因此在本质上，深度学习就是一个“特征学习（feature learning）”或“表示学习（representation learning）”[3]。



对于层级的CNN结构来说，不同层级的神经元学习到了不同类型的图像特征，由底向上特征形成层级结构，如上图所示，如果我们手头是个人脸识别任务，训练好网络后，把每层神经元学习到的特征可视化肉眼看一看每层学到了啥特征，你会看到最底层的神经元学到的是线段等特征，图示的第二个隐层学到的是人脸五官的轮廓，第三层学到的是人脸的轮廓，通过三步形成了特征的层级结构，越是底层的特征越是所有不论什么领域的图像都会具备的比如边角线弧线等底层基础特征，越往上抽取出的特征越与手头任务相关。正因为此，所以预训练好的网络参数，尤其是底层的网络参数抽取出特征跟具体任务越无关，越具备任务的通用性，所以这是为何一般用底层预训练好的参数初始化新任务网络参数的原因。而高层特征跟任务关联较大，实际可以不用使用，或者采用Fine-tuning用新数据集合清洗掉高层无关的特征抽取器。

**卷积层的3个核心概念**

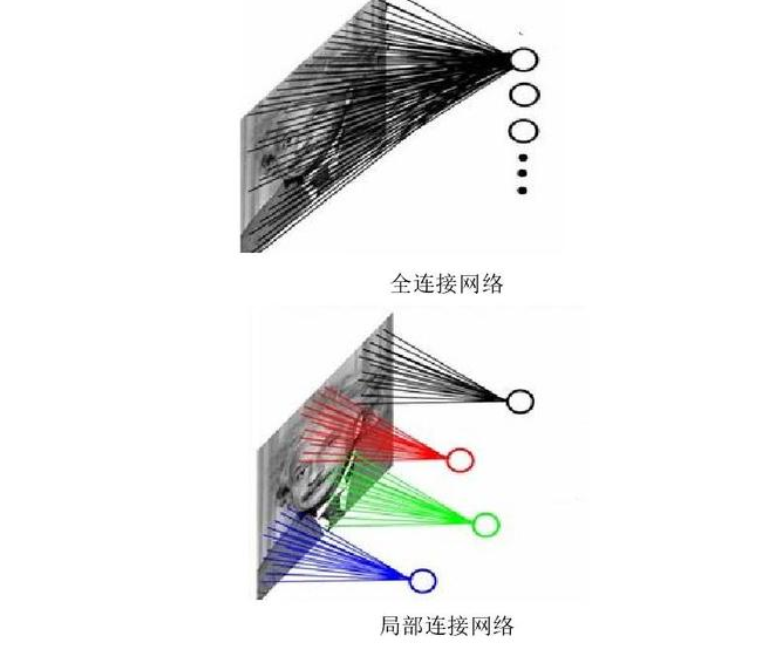
有了上面的工作的铺垫，下面我们来聊聊卷积层的三个核心概念：局部连接、空间位置排列及权值共享。

### 局部连接

全连接的前馈神经网络有个非常致命的缺点，那就是可扩展性（Scalability）非常差。原因非常简单，网络规模一大，需要调参的个数以神经元数的平方倍增，导致它难以承受参数太多之痛。局部连接（Local Connectivity）在能某种程度上缓解这个“参数之痛”。

每一幅CIFAR-10图像都是32×32×3的RGB图像。对于隐藏层的某个神经元，如果是全连接前馈网络中，它不得不和前一层的所有神经元（32×32）都保持连接。对于RGB图像而言，如果我们需要在红色、蓝色和绿色等三个通道提取特征，那么卷积核个数就是3）。所以对于隐藏层的某个神经元，它的前向连接个数是由全连接的32×32×3个。

但现在，对于卷积神经网络而言，隐藏层的这个神经元仅仅需要与前向层的部分区域相连接。这个局部连接区域有个特别的名称叫“感知域（receptive field）”，其大小等同于卷积核的大小（比如说5×5）目前看应该就是卷积层，从卷积过程看，卷积层的每一个像素(神经元)都是原图像某一片区域的线性聚合信息组成的，其实就是那一小块和卷积核同样大小的局部区域。卷积层相当于全连接网络中的隐藏层。所以对于隐藏层的某个神经元，它的前向连接个数是由全连接的32×32×3个，通过卷积操作，减少到局部连接的到5×5×3个，这里的权重矩阵变成了卷积核的矩阵，学习的过程就是不断的调整卷积核的值。但需要注意的是，这里的稀疏连接，仅仅是指卷积核的感知域（5×5）相对于原始图像的高度和宽度（32×32）而言的。卷积核的深度（depth）则需要与原始数据保持一致，不能缩减。在这里，卷积核的深度实际上就是卷积核的个数。对于RGB图像而言，如果我们需要在红色、蓝色和绿色等三个通道提取特征，那么卷积核个数就是3）



对比全连接网络这一层隐藏层表达 W\*X = H

对于某一个隐藏层神经元，连接的权边有W=32×32×3

如果转换为卷积神经网络 W\*C(X(局部)) = H

对于某一个隐藏层神经元，连接的权边有W=5×5×3

这个神经元是原图中局部区域与卷积核线性组合的结果，而不在是和全部像素连接



f1234就是卷积层，就是这一层的神经元，对于f1这个神经元它至于a,b,c,d四个像素连接

而不是全部像素，这就体现了局部连接的道理

### 空间排列

下面我们来谈谈决定卷积层的空间排列（Spatial arrangement）的4个参数，它们分别是：卷积核的大小、深度、步幅及补零。

卷积核的大小（通常多是3×3或5×5的方矩阵）

1. 卷积核的深度（depth）：卷积核的深度对应的是卷积核的个数。每个卷积核只能提取输入数据的部分特征。每一个卷积核与原始输入数据执行卷积操作，会得到一个卷积特征，这样的多个特征汇集在一起，我们称为卷积特征。在图11-1所示的示例中（左上部），我们使用三个不同的滤波器（即卷积核）对原始图像进行卷积操作，这样就可以生成三个不同的卷积特征图。你可以把这三个特征图看作是堆叠在一起的2D（二维）矩阵。每个卷积核提取的特征都有各自的侧重点。因此，通常说来，多个卷积核的叠加效果要比单个卷积核的分类效果要好得多。例如在2012年的ImageNet竞赛中，Hinton教授和他的学生Krizhevsky等人打造了第一个“大型的深度卷积神经网络”，也即现在众所周知的AlexNet。在这个夺得冠军的深度学习算法中，他们使用的卷积核高达96个！可以说，自那时起，深度卷积神经网络一战成名，才逐渐被世人瞩目。
2. 步幅（stride）：即滤波矩阵在输入矩阵上滑动跨越的单元个数。设步幅大小为S，当S为1时，滤波器每次移动一个像素的位置。当S为2时，每次移动滤波器会跳过2个像素。S越大，卷积得到特征图就越小。以一维数据为例，当卷积核为[1，0，-1]，输入矩阵为[0, 1, 2, -1, 1, -3, 0]时，图11-5显示了步幅分别为1和2卷积层的神经元分布情况
3. 补零（zero-padding）：补零操作通常用于边界处理。在有些场景下，卷积核的大小并不一定刚好就被输入数据矩阵的维度大小整除。因此，就会出现卷积核不能完全覆盖边界元素的情况。这时，我们就需要在输入矩阵的边缘使用零值进行填充，使得在输入矩阵的边界处的大小刚好和卷积核大小匹配。这样做的结果，相当于对输入图像矩阵的边缘进行了一次滤波。零填充的好处在于，它可以让我们控制卷积特征图的大小。使用零填充的卷积叫做泛卷积（wide convolution），不适用零填充的叫做严格卷积（narrow convolution）。

综上所述，在构造卷积层时，对于给定的输入数据，如果确定了卷积核的大小，卷积核的深度（个数）、步幅以及补零个数，那么卷积层的空间安排就能确定下来。以一维数据为例，假设数据的大小（数据元素的长度）为W，卷积核的深度为F，步幅大小为S，补零的数目为P，那么对于每个卷积核，在它与输入数据实施卷积操作后得到特征图谱，它包含的神经元个数N可以用（11-1）公式计算得到。

卷积层特征个数就是神经元的个数。

### 权值共享

卷积层设计的第三个核心概念就是权值共享（Shared Weights），由于这些权值实际上就是不同神经元之间的连接参数，所以有时候，也将权值共享称为参数共享（Parameter Sharing）。为什么要设置权值共享呢？其实这也是无奈之举。前文我们提到，通过局部连接处理后，神经元之间的连接个数已经有所减少。可到底减少多少呢？还以CIFAR-10数据集合为例，一个原始的图像大小为32×32×3，假设我们有100个卷积核，每个卷积核的大小为5×5×3，步幅为1，没有补零。先单独考虑一个卷积核，将公式11-1扩展到二维空间，可以很容易计算得到每一个卷积核对应的特征图谱大小是28×28。也就是说，这个特征图谱对应有28×28神经元。而每个神经元以卷积核大小（5×5×3）连接前一层的“感知域（receptive field）”，也就是说，它的连接参数个数为（28×28）×（5×5×3）。如果考虑所有的100个卷积核，（在不考虑偏置参数的情况下）连接的参数个数为（5×5×3）×（28×28）×100 = 5,880,000。

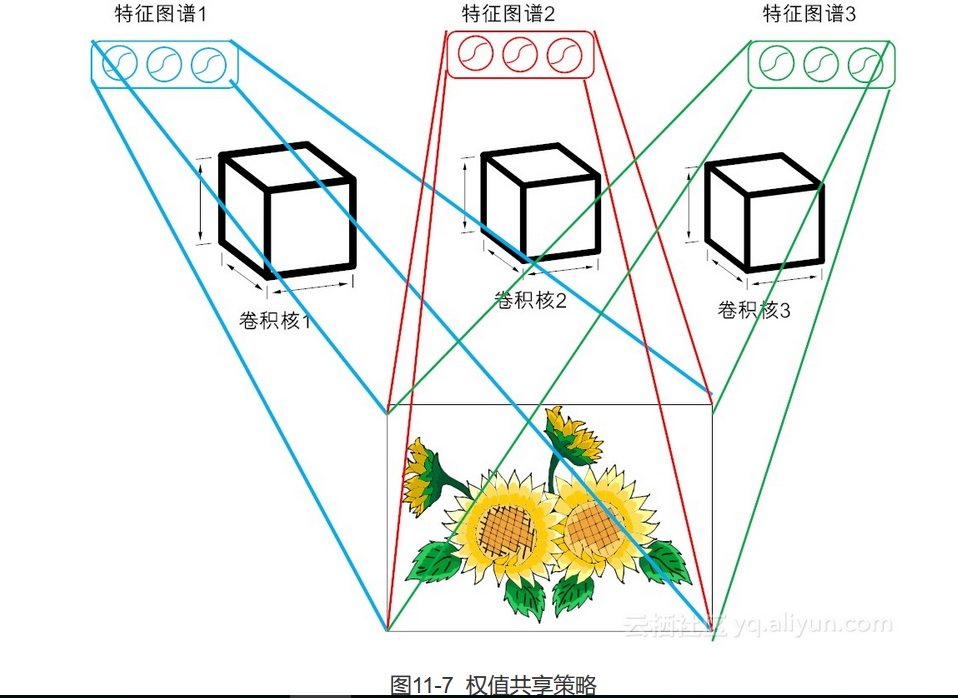
那么全连接的参数个数又是多少呢？仅仅考虑两层网络的情况下，其连接个数为（32×32×3）×（32×32×3）=9,437,184。对比这二者的数字可以发现，局部连接虽然降低了连接的个数，但整体幅度并不大，需要调节的参数个数依然非常庞大，因此还是无法满足高效训练参数的需求。

而权值共享就是来解决这个问题的，它能显著降低参数的数量。该如何理解权值共享呢？首先从生物学意义上来看，相邻神经元的活性相似，从而可以它们共享相同的连接权值。其实就是说，每一个卷积层的神经元的值都是由输入层局部空间（感知域）与卷积核线性作用的结果，每一个值就是卷积核权值的线性组合，所以才有（5×5×3）×（28×28）×100个连接，权值共享的策略就是同属于一个特征图谱的神经元，将共用一个权值参数矩阵，也就是一个卷积核对全部的卷积层神经元只对应有一套权重参数，你们这28\*28个神经元共享。



对于神经元f1234来说，如果对每一个神经元都维护一套卷积核参数w显然成本很大，最终同一卷积层的神经元将共享一套参数

而权值共享就是来解决这个问题的，它能显著降低参数的数量。该如何理解权值共享呢？首先从生物学意义上来看，相邻神经元的活性相似，从而可以它们共享相同的连接权值。其次单从数据特征上来看，我们可以把每个卷积核（即过滤核）当作一种特征提取方式，而这种方式与图像等数据的位置无关。这就意味着，对于同一个卷积核，它在一个区域提取到的特征，也能适用于于其他区域。基于权值共享策略，将卷积层神经元与输入数据相连，同属于一个特征图谱的神经元，将共用一个权值参数矩阵，如图11-7所示。经过权值共享处理后，CIFAR-10的连接参数一下子锐减为5×5×3×1×100 = 7500。这样就不用把每一个和卷积层神经元相连的权重都做更新了，参数空间立马下降了很多

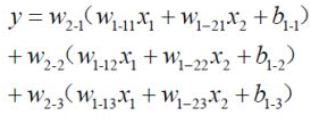


# 激活层

第一个问题，你能用直线画出一张漂亮的笑脸吗？

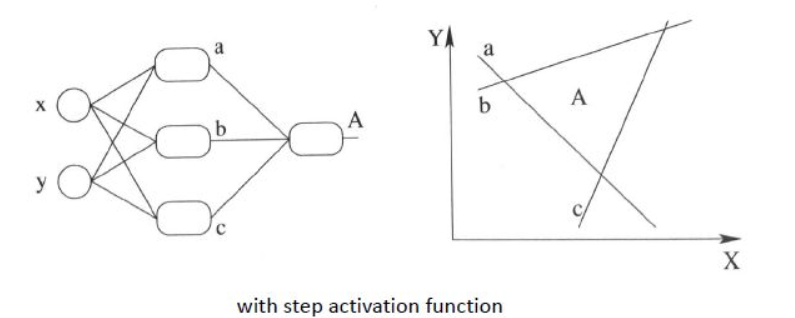
线性的事物，表达能力不强，而非线性则相反。我们知道，从宏观来将，在本质上，人工神经网络就分为两大类层：显层和隐层。“显层”就是我们能感知到的输入层和输出层，而“隐层”则是除了输入输出之外的无法被我们感知的层，它可以理解为数据的内在表达。

线性[linear](http://www.elecfans.com/tags/linear/" \t "_blank)，指量与量之间按比例、成直线的关系，在数学上可以理解为一阶导数为常数的函数；非线性non-linear则指不按比例、不成直线的关系，一阶导数不为常数。两个变量之间的关系是一次关系的这样的两个变量之间的关系就是“线性关系”；



对于神经网络来说，不管在怎么叠加，他的函数始终是一个线性函数，就好比在一维空间他只能表达一条直线，在二维空间他只能是一个平面Ax+By=Z

再盗用一幅经常在课堂上用的图...然而我已经不知道出处是哪了, 好像好多老师都是直接用的, 那我就不客气了嘿嘿嘿~~这幅图就跟前面的图一样, 描述了当我们直接使用step activation function的时候所能获得的分类器, 其实只能还是线性的, 最多不过是复杂的线性组合罢了~~~当然你可以说我们可以用无限条直线去逼近一条曲线啊......额,当然可以, 不过比起用non-linear的activation function来说就太傻了嘛...



三个神经元表达的结果任然是线性组合

[在第二章中](https://yq.aliyun.com/articles/88300)，我们已经提到，如果“隐层”有足够多的神经元，那么神经网络能够以任意精度逼近任意复杂度的连续函数，这就是大名鼎鼎的通用近似定理（Universal Approximation Theorem）[2]。通过[在第八章](https://yq.aliyun.com/articles/110025)BP算法的讲解中，我们可以看到，神经元与神经元的连接都是基于权值的线性组合。我们知道，线性的组合依然是线性的，转换到卷积网络其组合依然是线性的，那网络的表达能力就非常有限了。这样一来，通用近似定理又是如何起作用的呢？这就得请“激活”函数出马了？神经元之间的连接是线性的，但激活函数可不一定是线性的啊，有了非线性的激活层在，多么玄妙的函数，我们都能近似表征出来。所以，在卷积神经网络中，激活层是必须保留的。

在生物意义上的神经元中，只有前面的树突传递的信号的加权和值**大于某一个特定的阈值**的时候，后面的神经元才会被激活。

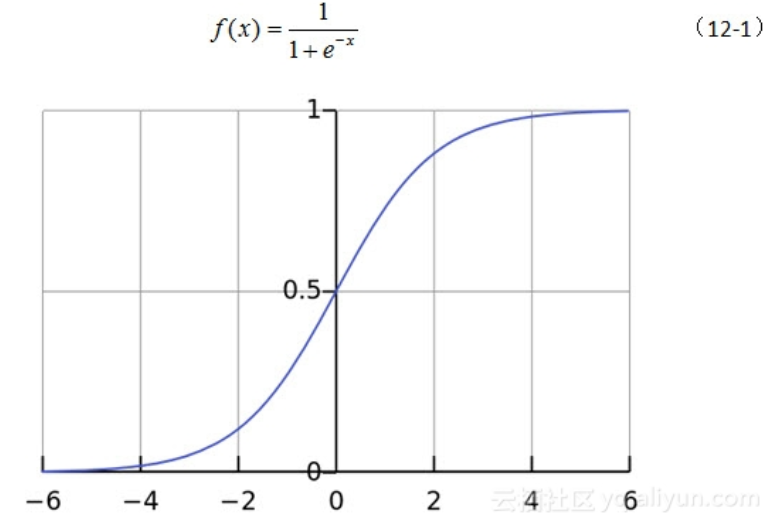
简单的说激活函数的意义在于判定每个神经元的输出

放在人脸识别卷积神经网络中来思考，卷积层的激活函数的意义在于这一块区域的特征强度如果没有达到一定的标准，就输出0，表明这种特征提取方式（卷积核w）不能在该块区域提取到特征，或者说这块区域的这种特征很弱。由于输出0时，激活函数梯度几乎都为0，所以在特征很弱的区域，梯度几乎不下降，也就是说，和该特征无关的区域不会影响到该特征提取方式的训练。

反过来说，如果没有激活函数，就算特征很弱时，仍然可以得到输出，后面的层次继续把这个输出当成一个特征使用，这是不合理的。为了形象起见，我们做个比喻，本来我们可能希望在这个区域看到一个圆形，但是这里却是一个三角形，如果该三角形和我们期望看到的圆形形状相差很大的时候神经元的输出WX+B很小，我们就不希望输出值去衡量这个三角形有多不圆，而是希望输出这里没有一个圆形（对于分类问题而言，有多不圆对于我们的分类没有意义，因为我们是通过了解到形状的组合来判定人脸属于哪一类，验证问题同理）。我们真正期望的是卷积层可以提取那些我们需要的特征，和真实样本接近的特征，也就是说如果某个局部区域通过卷积操作得到的输出非常弱，那么这个神经元就不要往下继续传递信息，也就是说信号应该被抑制，反之某些区域携带了图片的典型特征，那么这个区域的卷积操作输出的神经元应该是激活状态的

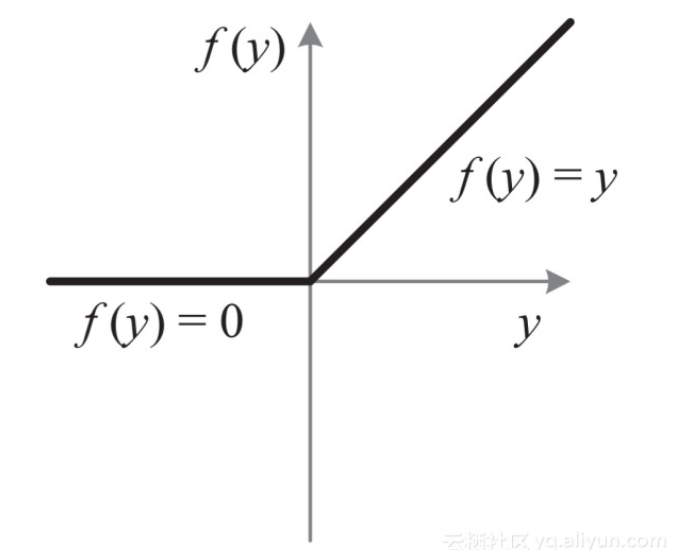
通过前面的铺垫，现在我们应该知道，激活层存在的最大目的，莫过于引入非线性因素，以增加整个网络的表征能力。

这时，选取合适的“激活函数”就显得非常重要了。在前面的章节中，我们提到了常用的激活函数Sigmoid（或tanh函数），也是可用的（如图12-3所示）。



但Sigmoid之类激活函数有个很大的缺点，就是它的导数值很小。从图像就可以看得出来，比如说，Sigmoid的导数取值范围仅为[0, 1/4]。且当输入数据（*x*）很大或者很小的时候，其导数趋都近于0。这就意味着，很容易产生所谓的梯度消失（vanishing gradient）现象。没有了梯度的指导，那么神经网络的参数训练，就如同“无头的苍蝇”，毫无方向感。

因此，如何防止深度神经网络陷入梯度消失，或说如何提升网络的训练效率，一直都是深度学习非常热门的研究课题。目前，在卷积神经网络中，最常用的激活函数久是修正线性单元(Rectified Linear Unit，简称ReLU)。这个激活函数是由Hinton等人2010年提出来的[3]。标准的ReLU函数为*f*(*x*)=max(*x*,0)，即当*x*>0时，输出*x*; 当*x*<=0时，输出0。如图12-4所示，请注意，这是一条曲线啊，只不过它在原点处不够那么圆润而已。

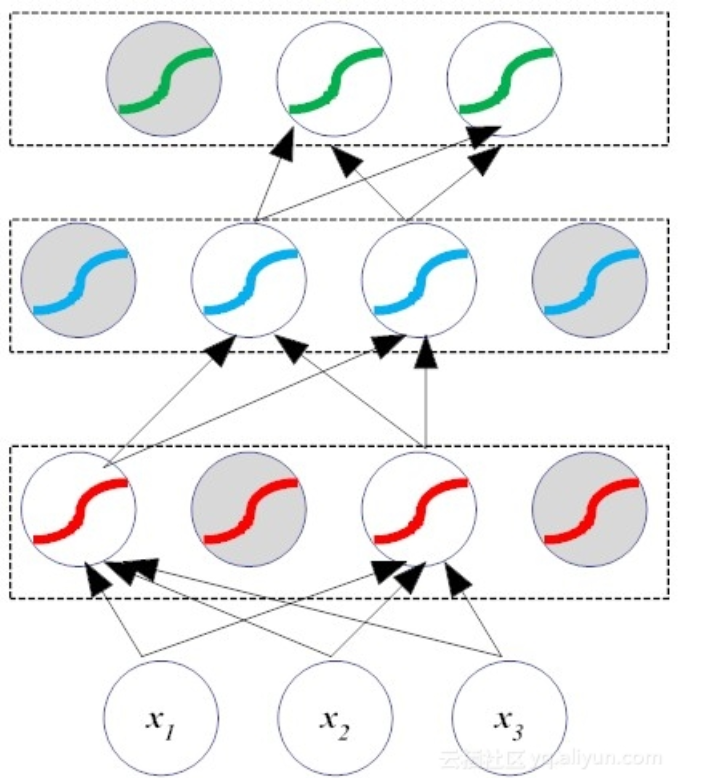


不要小看这个看起来有点简陋的模型，其实它的优点还不少。相比于Sigmoid类激活函数，ReLU激活函数的优点主要体现在如下三点。

**（1）单侧抑制。**观察图12-4可见，当输入小于0时，神经元处于抑制状态。反之，当输入大于0，神经元处于激活状态。

**（2）相对宽阔的兴奋边界。**观察图12-3和图12-4可见，Sigmoid的激活态（即*f*(*x*)的取值）集中在中间的狭小空间，而ReLU这不同，只要输入大于0，神经元一直都处于激活状态。

**（3）稀疏激活性。**相比于Sigmoid之类的激活函数，稀疏性是ReLU的优势所在[4]。Sigmoid把抑制状态的神经元设置一个非常小的值，但即使这个值再小，后续的计算还少不了它们的参与，计算负担很大。但考察图12-4可知，ReLU直接把抑制态的神经元“简单粗暴”地设置为0，这样一来，就使得这些神经元不再参与后续的计算，计算量极度下降。从而造成网络的稀疏性，如图12-5所示。



某些神经元处于抑制状态，不在向后传播信息这个细小的变化，让ReLU在实际应用中大放异彩，除了减少了计算量，还减少了参数的相互依存关系（网络瘦身了不少），使其收敛速度远远快于其他激活函数，最后还在一定程度上缓解了过拟合问题的发生（对Dropout机制比较熟悉的读者可能会发现，图12-5和Dropout的迭代过程何其神似！）。ReLU的卓越表现，让深度学习的三位大咖Yann LeCun、Yoshua Bengio和Geoffery Hinton在2015年表示，ReLU是深度学习领域最受欢迎的激活函数。

说到ReLU激活函数有如此神奇作用，其实还有一个原因，那就是这样的模型正好“暗合”生物神经网络工作机理。2003年纽约大学教授Peter Lennie的研究发现[5]，大脑同时被激活的神经元只有1~4%，即神经元同时只对输入信号的少部分选择性响应，大量信号被刻意地屏蔽了，这进一步表明神经元工作的稀疏性。其实，这是容易理解的，因为生物运算也是需要成本的。进化论告诉我们，作为人体最为耗能的器官，大脑尽要可能节能，才能在恶劣的环境中“适者生存”。

当然，LeRU的这种简单直接的处理方式，也带来一些副作用。最突出的问题就是，会导致网络在训练后期表现得非常脆弱，以至于这时的ReLU也被戏称为“死掉的ReLU（dying ReLU）”

# 池化层

诸葛一生唯谨慎，吕端大事不糊涂

其实是想说明深度学习训练的两大难点：过拟合（overfitting）和欠拟合（underfitting）。那什么是过拟合和欠拟合呢？图12-2可形象地说明这两个概念的差别。

卷积神经网络也追求泛化（即防过拟合）能力，它是如何做到的呢？自然也得学习“吕端”的行为——别管那么多！

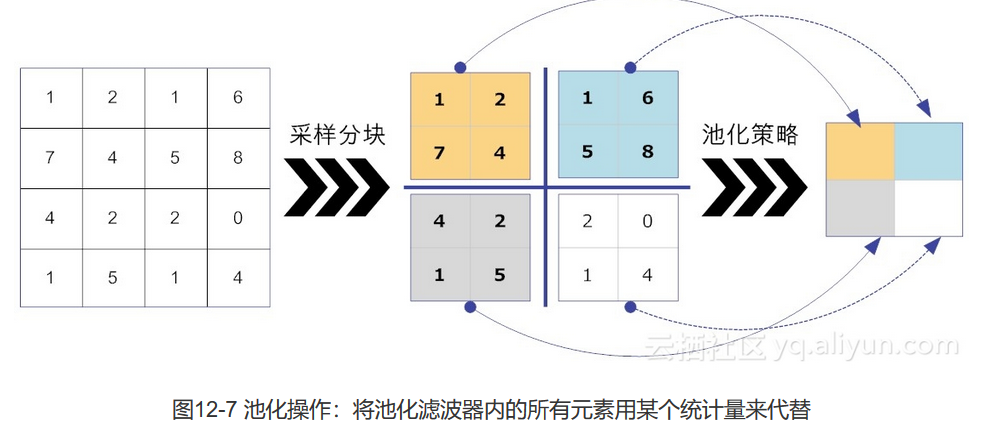
针对神经网络，就是再次降低数据量，让系统少学点。不要认为，训练数据越“全面”越好。想一想人类的学习就知道怎么回事了。当孩子还小正处于学习阶段时，妈妈们的浓浓爱意，总想通过“事无巨细”地照顾孩子表达出来。但在这种环境下“学习”出来的孩子，一旦踏上社会，适应新环境的能力就差很多，并不值得提倡。神经网络也是如此。

那该如何降低数据量呢？最简单的策略自然就是“采样（sampling）”了。其实，**采样的本质就是力图以合理的方式“以偏概全”。**这样一来，数据量自然就降低了。

在卷积神经网络中，采样是针对若干个相邻的神经元而言的，因此也称为“亚采样（Subsampling）”。可能是“亚采样”这个词的逼格不够高吧，于是研究者们又给它取了个更难懂的词：“池化（Pooling）”。“池化”其实仅仅是个字面的翻译，远没达不到“信达雅”的要求，如果非要向“采样”的含义靠拢，中国那句古话，“弱水三千只取一瓢”，似乎更有韵味。南京大学周志华老师就将其的意译为“汇合”，这样的翻译似乎更加传神。但拗不过太多人都把“Pooling”翻译成“池化”，那我们也就“池化”叫下去吧。

池化层亦称子采样层，它也是卷积神经网络的另外一个“神来之笔”。通常来说，当卷积层提取目标的某个特征之后，我们都要在两个相邻的卷积层之间安排一个池化层。

池化层函数实际上是一个统计函数。以如图12-7所示的二维数据为例，如果输入数据的维度大小为W×H，给定一个池化过滤器，其大小为w×h。池化函数考察的是在输入数据中，大小为w×h的子区域之内，所有元素具有的某一种特性。常见的统计特性包括最大值、均值、累加和及L2范数等。池化层函数力图用统计特性反应出来的1个值，来代替原来w×h的整个子区域



因此，可以这么说，池化层设计的目的主要有两个。最直接的目的，就是降低了下一层待处理的数据量。比如说，当卷积层的输出大小是32×32时，如果池化层过滤器的大小为2×2时，那么经过池化层处理后，输出数据的大小为16×16，也就是说现有的数据量一下子减少到池化前的1/4。当池化层最直接的目的达到了，那么它的间接目的也达到了：减少了参数数量，从而可以预防网络过拟合。

池化综合了（过滤核范围内的）全部邻居的反馈，即通过*k*个像素的统计特性而不是单个像素来提取特征，自然这种方法能够大大提高神经网络的性能[9]

**（1）最大池化函数（max pooling）**

**前向传播操作：**取滤波器最大值作为输出结果，因此有forward(1, 2, 3, 2) = 3.

**反向传播操作：**滤波器的最大值不变，其余元素置0。因此有backward(3) = [0, 0, 3, 0]。

**（2）平均池化函数（average pooling）**

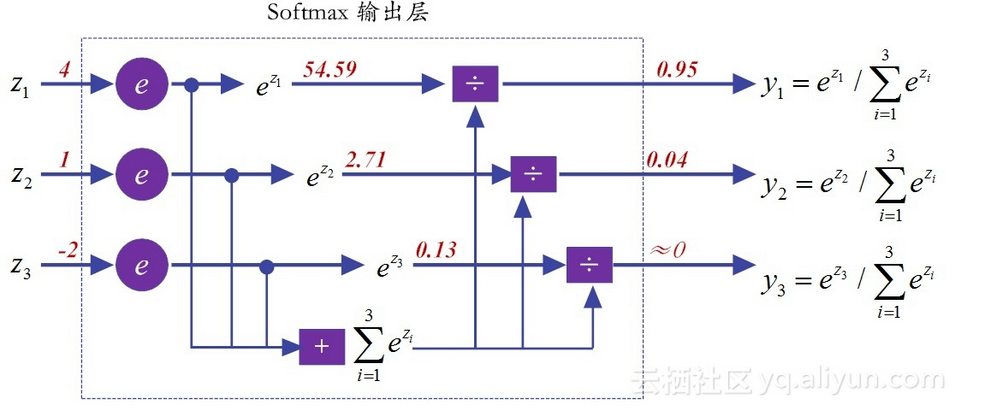
**前向传播操作：**取滤波器范围所有元素的平均值作为数据结果，因此有forward(1, 2, 3, 2) = 2.

**后向传播操作：**滤波器中所有元素的值，都取平均值，因此有backward(2) = [2, 2, 2, 2]。

# 全连接层 FC

前面我们讲解了卷积层、激活层和池化层。但别忘了，在卷积神经网络的最后，还有一个至关重要的“全连接层（Fully Connected Layer，简称FC）”。“全连接”意味着，前层网络中的所有神经元都与下一层的所有神经元连接。全连接层设计目的在于，它将前面各个层学习到的“分布式特征表示”，映射到样本标记空间，然后利用损失函数来调控学习过程，最后给出对象的分类预测。

实际上，全连接层是就是传统的多层感知器（类似于我们在第八章学过的BP网络，不熟悉的读者可以[前往查阅](https://yq.aliyun.com/articles/110025)）。不同于BP全连接网络的是，卷积神经网络在输出层使用的激活函数不同，比如说它可能会使用Softmax函数。



虽然全连接层处于卷积神经网络最末的位置，看起来貌不惊人似的，但由于全连接层的参数冗余，导致该层的参数个数占据整个网络参数的绝大部分。这样一来，稍有不慎，全连接层就容易陷入过拟合的窘境，导致网络的泛化能力难尽人意。

到此为止，我们介绍完毕了卷积神经网络的所有核心层。各个层各司其职，概括起来，

卷积层从数据中提取有用的特征；

激活层为网络中引入非线性，增强网络表征能力；

池化层通过采样减少特征维度，并保持这些特征具有某种程度上的尺度变化不变性。

在全连接层实施对象的分类预测。