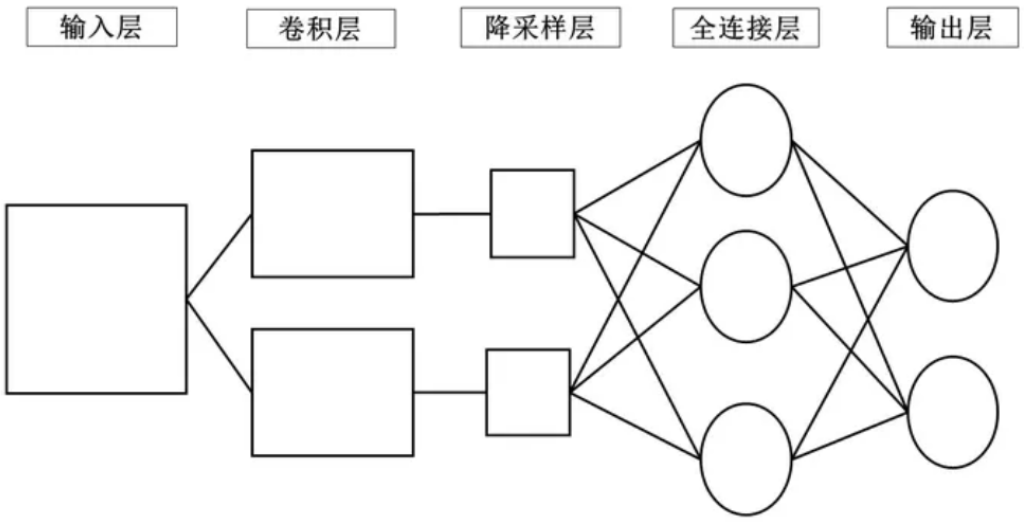
卷积神经网络（CNN）

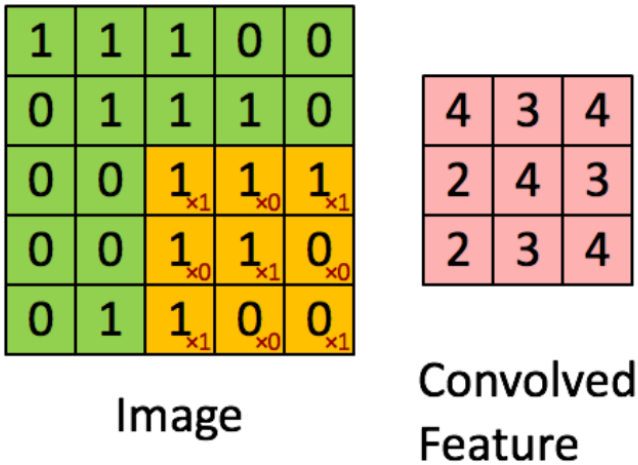
> **引入：全连接神经网络在图像处理方面所遇到的困难**
>
> 作为一幅图像，其中的信息维度通常是很高的,在参数如此巨大的情况下，难以获取足够数据防止神经网络的欠拟合。因此我们需要一种运算方式来减小输入全连接网络的特征向量的维度。这种运算就是卷积运算。

一个完整的卷积神经网络通常由三部分构成，分别是卷积层（Conv）、池化层（Pool）和全连接层（FC）。



1.什么是卷积?

- 卷积（convolution），指的是数字图像处理的卷积操作，简单概括为**对应相乘再相加**，如图
- **二维卷积**



- image代表原始的图像，卷积一般有卷积模板（或者叫卷积核），图中黄色方块右下角组成的3×3的卷积核，该卷积核为：

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

- **Valid convolution**：可以看出在经过3×3 的卷积核卷积操作之后，生成的feature map尺寸和原图像不一样，卷积操作隐含一个参数stride（步长），步长就是卷积核滑动的长度，这里默认stride=1。假设原始图像尺寸为$M \times N$,卷积核尺寸为$W_1 \times W_2$,步长stride=S,则feature map的尺寸为

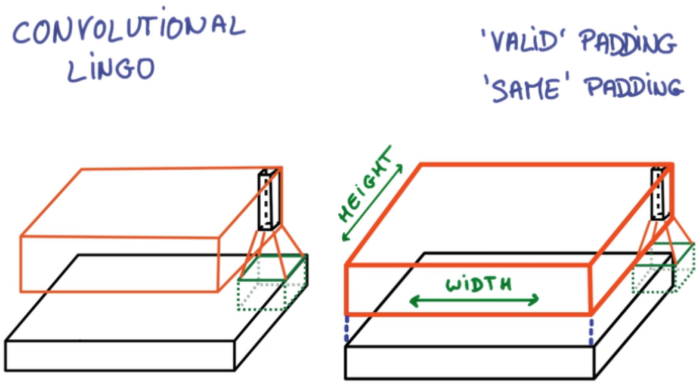
  $$((M-W_1)/S+1)\times((N-W_2)/S+1)$$

  **(但是我们发现这样的运算方式对图像边缘的信息利用较少，甚至经过多次卷积运算之后，原图像边角的像素信息基本无法保留下来，那么该怎么去改善这样的情况呢？）

- **Same convolution**：我们需要引入padding的概念，也就是扩展像素，这里设置padding=1，（即原始图像上下左右扩展1个像素，一般用0填充），则经过padding后的图像尺寸为7×7，再通过卷积则会得到5×5的feature map。则padding=P,则feature map输出尺寸为

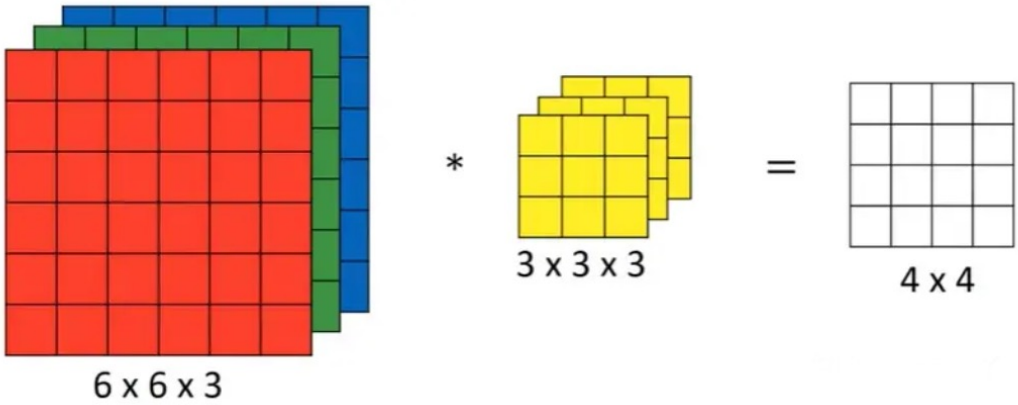
  $$((M-W_1+2P)/S+1)\times((N-W_2+2P)/S+1)$$

  （括号内不能整除则需向下取整）



- **三维卷积**

  图像处理来讲，图像信息通常是三维的（每个像素点包含RGB三通道值）。卷积核也需要相应地拓展维度，此时的卷积可以想象为下面的样子。



三个维度的名称分别为height×width×channels，卷积核与原图像的通道数必须保持一致。运算出的结果是一个4×4×1的矩阵。

- **具体的运算规则：**

将卷积核视为一个3×3×3的立方体，原图像可以看作一个6×6×3的槽。卷积核先放入原图像的左上角中，对应的27个数相乘后加和，得到输出图像矩阵(1,1)像素值，之后卷积核以一定步幅滑动，与二维卷积大同小异。

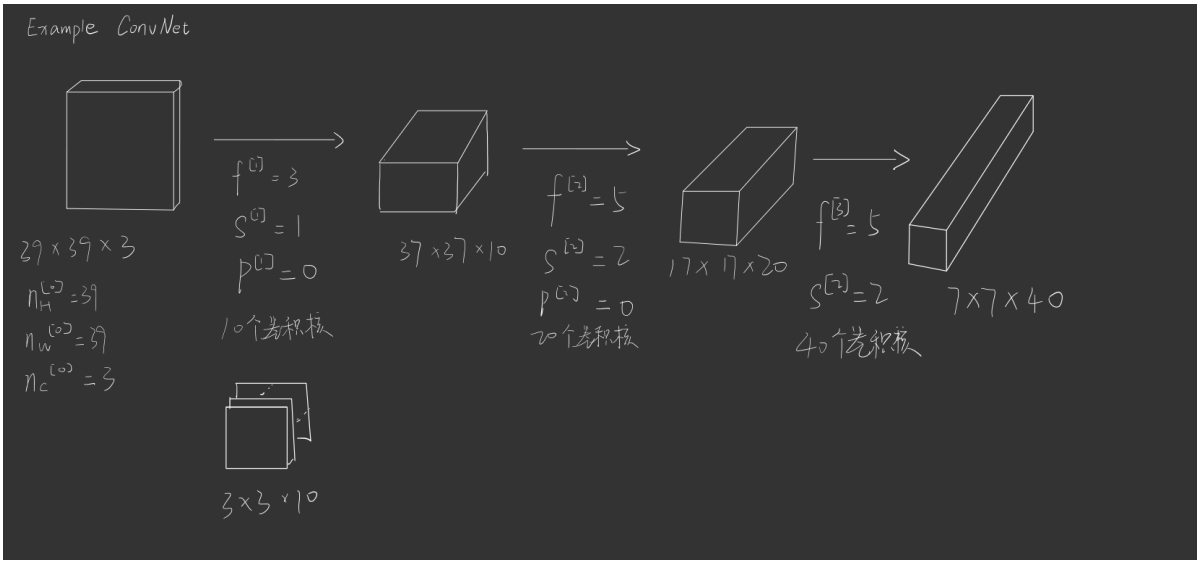
稍加思考可以发现，上述运算步骤可以分解为3个互不相关的二维卷积，得到3个4×4的输出图像。三维卷积的结果只是将这3个二维卷积输出结果叠加起来而已。

针对这种特性，并且由于不同通道间的互不相关，我们可以赋予卷积核上各通道以不同的功能。具体应用**对RGB不同通道的不同处理。**

**当然我们也可以使用多个卷积核来提取不同的特征，如用两个卷积核，我们就可以得到两个4×4的矩阵，将它们叠放在一起就可以得到维度$4 \times 4 \times 2$，也就是也就是说输出图像的通道数取决于使用的卷积核的个数。这样我们就可以检测原图像的多种特征，而这些特征都将体现在输出图像的各个通道中。*

卷积层（ConvNet）

事实上，用多个卷积核对一个多通道图像做卷积运算，最后得到一个新的多通道图像的计算过程，就构成了卷积层的一个单元。而像下图所示这样将若干个单元串在一起，就得到了一个简单但完整的卷积层。

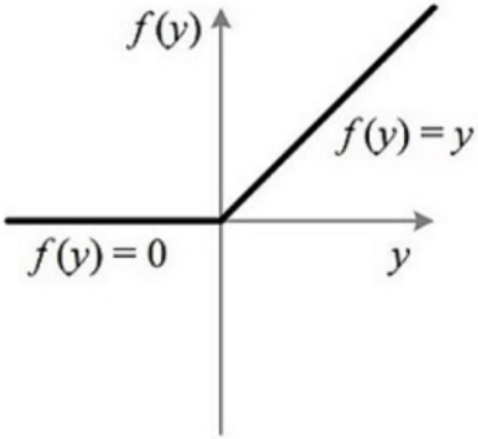


其中$n_H^{[i]},n_W^{[i]},n_C^{[i]}$分别为每层图像的高、宽、通道数（原图像i=0），$f^{[i+1]}$,$s^{[i+1]}$,$p^{[i+1]}$分别是作用于第i层图像的卷积核的边长、卷积步幅、Padding值，同时卷积核的个数决定了第i+1层图像的通道数$n_C^{[i]}$。

从上述例子里我们可以看出，通常情况下，随着卷积层深度的增加，图像的长宽逐渐减小，而通道数逐渐增大。经过上面的卷积层，图像的特征量39×39×3=4563减小为7×7×40=1960，之后就可以将这1960个特征量展开为一个长度为1960的向量，作为输入向量X传入全连接神经网络。或者会在这二者间加上一个池化层，这就涉及卷积神经网络的结构了。

非线性激活ReLu：

- ReLu激活函数是在CNN中经常用到的，$ReLu(x) = max(0, x)$，卷积层对原图运算多个卷积产生一组线性激活响应，而非线性激活是对之前的结果进行非线性的响应。
- 是对卷积后的特征图进行有目的的提取，小于0的不是识别特征，故记为0，而大于0的就是与卷积模板匹配的特征保留下来。



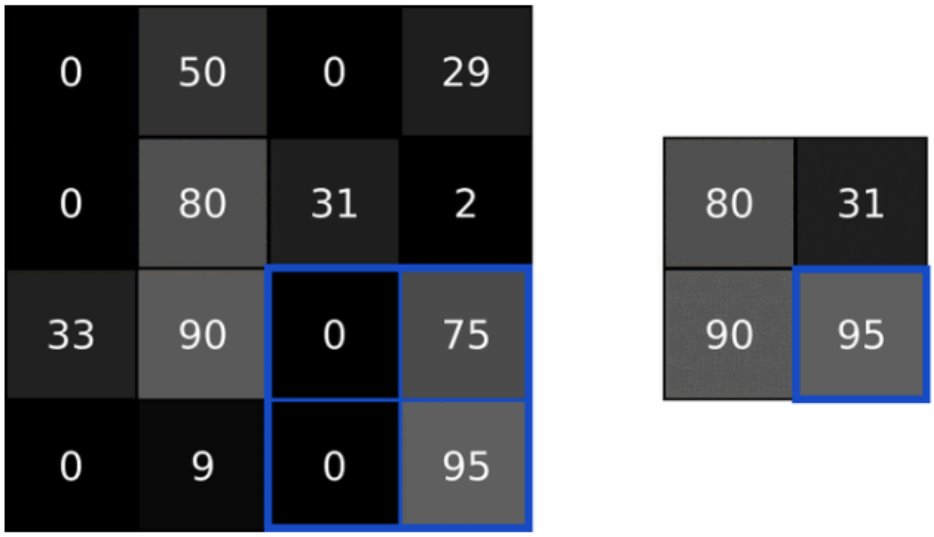
$$ReLU(x) = \begin{cases} x & if x > 0 \\ 0 & if x \leq 0 \end{cases}$$

池化层（Pooling Layers）

首先需了解池化层是如何工作的。

1.最大池化（Max pooling）

以最大池化为例，我们将一个4×4的原图像分割为4个2×2的子区域，每个子区域中提取最大值即可。



上面就是**Max Pooling** 的过程，其中蓝色的是池化模板的尺寸为2×2，池化也有stride步长，一般stride=2，最大池化就是取2×2特征值中的最大值，然后以stride=2进行移动池化模板，遍历完整个feature map。设原来feature map尺寸为$M\times N$，池化模板尺寸为$W\times W$,步长为S(一般取2)，padding 为P（池化也可以先padding,不过要注意的是，在池化中通常不设置padding值，即p=0。），可以得到池化层的输出尺寸公式：

$$((M-W+2P)/S+1)\times((N-W+2P)/S+1)$$

多通道图像的池化也很容易想象，各通道分别做池化，再将结果叠放在一起。所以**池化的输入和输出的通道数是一致的。**

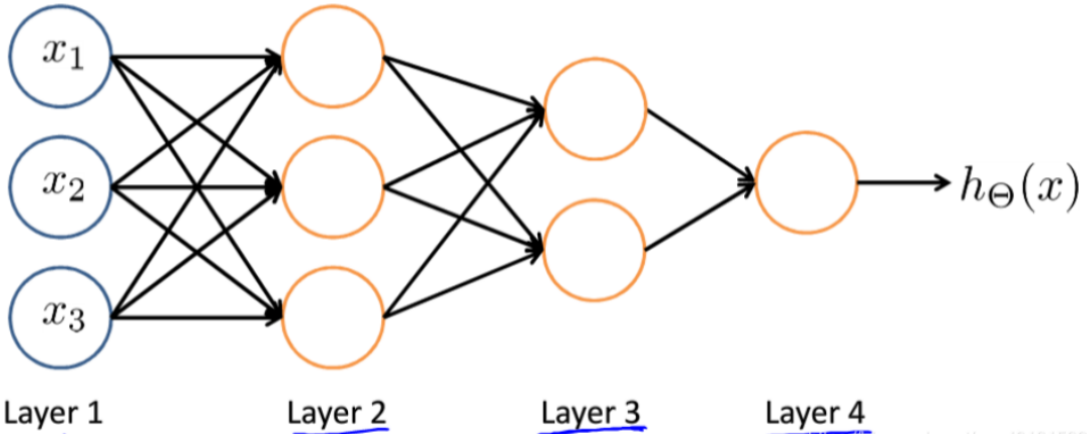
2.平均池化（Average Pooling）

与最大池化相比，我认为除了输出取子区域内平均值而非最大值以外，没有什么不同，就不再赘述了。

与卷积过程中可以选取不同的卷积核参数相比，池化是一种**固定**的运算，在确定了超参数后它就不存在继续调整或是训练的需要了。

全连接层（FC）

- 上面讨论过卷积和池化之后，深度学习CNN的神秘面纱也就被揭开，Deep主要体现在了网络的深度上，也就是许多的卷积层和池化层堆叠的结果，每一层都依据上述原理进行。
- 我们的目的是什么，回头看我们要提取图像的高层次特征不就是为了识别出该图像是什么吗？也就是**图像分类**，那么经过这么多卷积和池化之后已经提取了高维特征，下面就是对其进行分类了，这里的分类就用的是**全连接层**，全连接层的作用就是对特征进行维度上的改变来得到每个分类类别对应的概率值。
- 全连接层就是一个基本的神经网络



这边需要提一下全连接层最后一层输出使用的是Softmax分类函数,它可以将网络输出变成一个概率分布，且n类的概率相加等于1，这就很方便的可以判断网络最终分类的结果。

$$Softmax(y_i) = \frac{e^{y_i}}{\sum_{i=1}^{n} e^{y_i}}$$

也可以知道全连接层的输入是将池化后的`feature map`先进行了`flatten`打平操作，然后再输出到最后一层，之后通过Softmax函数输出概率。

卷积层和全连接层的区别

- **全连接**：它的每一个神经元都与上一层的神经元进行连接，故称全连接，相邻两层神经元与神经元之间都是通过一个参数相连。
- **卷积层"局部连接"、"参数共享"的思想：**

  前面的卷积层就肯定不是全连接了，他是**"局部连接"的思想，每一次只有3×3的卷积核与对应的图像进行相连，只不过这个卷积核通过步长在滑动。也就是说每次只有卷积核大小的图像与卷积核进行参数连接，而未连接的部分是通过将窗口滑动起来的方法进行后续的连接，这个思想就是"参数共享"**，这里的参数就是每个卷积核的值，对于同一个卷积核，在对图像进行卷积时，卷积核是不变的，所以参数可以共用。

CNN的反向传播

- 在CNN中，我们需要训练的就是那些卷积核的参数，训练的方法仍然是之前神经网络的正向传播和反向传播算法，只不过计算每一层的梯度和误差时不一样而已。
- 接下来我们只需知道卷积层、Pooling层的误差以及对每个参数梯度该怎么求。就可以进行训练了。而CNN反向传播算法的详细计算方法可以从这篇文章中借鉴<Convolutional Neural Networks backpropagation: from intuition to derivation – Grzegorz Gwardys (wordpress.com)>
- 首先对**卷积层**来说，δ误差是损失函数对于当前层未激活输出z^l的导数$\delta^l(x,y)=\frac{\partial C}{\partial z^l}=\sigma^{'}(z^l)$

  这边我们用另一篇笔记——卷积神经网络(CNN)反向传播算法推导中的结论：

  $$\delta^l=\delta^{l+1}*\mathrm{rot180}(w^{l+1})\odot\sigma^{'}(z^l)$$

  其中$\mathrm{rot180}()$表示将矩阵旋转180度，符号⊙为元素相乘。

  而对每一层的误差项，有

  $$\frac{\partial C}{\partial w^l}=\delta^l*\sigma(z^{l-1})$$

  $$\frac{\partial C}{\partial b^l}=\sum_x\sum_y\delta^l$$

- **池化层**的反向传播

最大池化：记录下最大值所在的位置。

平均池化：权重平均。

用公式来表示为

$$\delta^l=\mathrm{upsample}(\delta^{l+1})\odot\sigma^{'}(z^l)$$

CNN的训练

训练的基础算法是梯度下降算法

梯度下降常用的三种方法：**随机梯度SGD**，**Mini-batch梯度下降**，**BGD**

区别：SGD每次需要1个样本，BGD一次需要全部的样本，而Mini-batch位于中间。

参数的更新：

$$W^l=W^l-\eta*\frac{1}{\mathrm{batchsize}}\sum\frac{\partial L}{\partial W^l}$$

$$b^l=b^l-\eta*\frac{1}{\mathrm{batchsize}}\sum\frac{\partial L}{\partial b^l}$$

CNN整个训练过程

- 1.对神经网络进行初始化，定义好网络结构，设定好激活函数，对卷积层的卷积核w、偏置b进行随机初始化，对全连接层的权重矩阵W和偏置b进行随机初始化。
  设置好训练的最大迭代次数，每个训练batch的大小以及学习率η.
- 2.从训练数据中取出一个batch的数据
- 3.从该batch数据中取出一个数据，包括输入x以及对应的正确标注y.

- 4.将输入x送入神经网络的输入端，得到神经网络各层输出参数z^l和a^l
- 5.根据神经网络的输出和标注值y计算神经网络的损失函数Loss
- 6.计算损失函数Loss对输出层的误差δ^l
- 7.利用相邻层之间delta误差的递推公式求得每一层的delta误差

  全连接层:

  $$\delta^l = (W^{l+1})^T\delta^{l+1} \odot \sigma^{'}(z^l)$$

  卷积层:

  $$\delta^l = \delta^{l+1} * \mathrm{rot180}(w^{l+1}) \odot \sigma^{'}(z^l)$$

  池化层:

  $$\delta^l = \mathrm{upsample}(\delta^{\mathrm{l}+1}) \odot \sigma^{'}(z^l)$$

-
- 8.利用每一层的delta误差求出损失函数对该层参数的导数

  全连接层:

  $$\frac{\partial L}{\partial w^l} = \delta^l(a^{l-1})^T$$

  $$\frac{\partial L}{\partial b^l} = \delta^l$$

  卷积层:

  $$\frac{\partial L}{\partial w^l} = \delta^l * \sigma(z^{l-1})$$

  $$\frac{\partial L}{\partial b^l} = \sum_x\sum_y \delta^l$$

- 9.将求得的导数加到该batch数据求得的导数之和上(初始化为0)，跳转到步骤3，直到该batch数据都训练完毕
- 10.利用一个batch数据求得的导数之和，根据梯度下降法对参数进行更新

  $$W^l = W^l - \eta * \frac{1}{\mathrm{batchsize}}\sum\frac{\partial L}{\partial W^l}$$

  $$b^l = b^l - \eta * \frac{1}{\mathrm{batchsize}}\sum\frac{\partial L}{\partial b^l}$$

- 11.重复步骤2，直到到达指定的迭代次数。