输入层 input layer

白化（预处理）

使得学习算法的输入具有如下性质

1.特征之间相关性较低

2.所有特征具有相同的方差。

卷积计算层conv layer

局部关联：局部数据识别

窗口滑动：滑动预先设定步长，移动位置来得到下一个窗口

深度：转换次数（结果产生的depth）

步长：设定每一移动多少

填充值：可以再矩阵的周边添加一些扩充值（目的是解决图片输入不规整）

激励层 ReLu layer

使用映射函数，来完成非线性的映射

（1）双s和s函数用于全连接层

（2）ReLu用于卷积计算层(迭代较快，只是效果不佳)

（3）普遍使用ELU

（4）Maxout：使用最大值来设置值

池化层 Polling layer

（1）最大池化

（2）平均池化

全连接层 FC

对于数据的汇总计算

Dropout（兼听则明）

1.不要CNN具有太多的泛化能力（不能以来某几个神经元）

2.多次迭代结果的合并可以增加模型的准确率

（相当于删除神经元后形成的不同的模型，多个不同的模型的合并可以提高他的准确率）

LeNet5

ResNet

残差连接：

允许模型存在一些shortcuts，可以让研究者成功训练更深的神经网络，这样也能明显的优化Inception块。

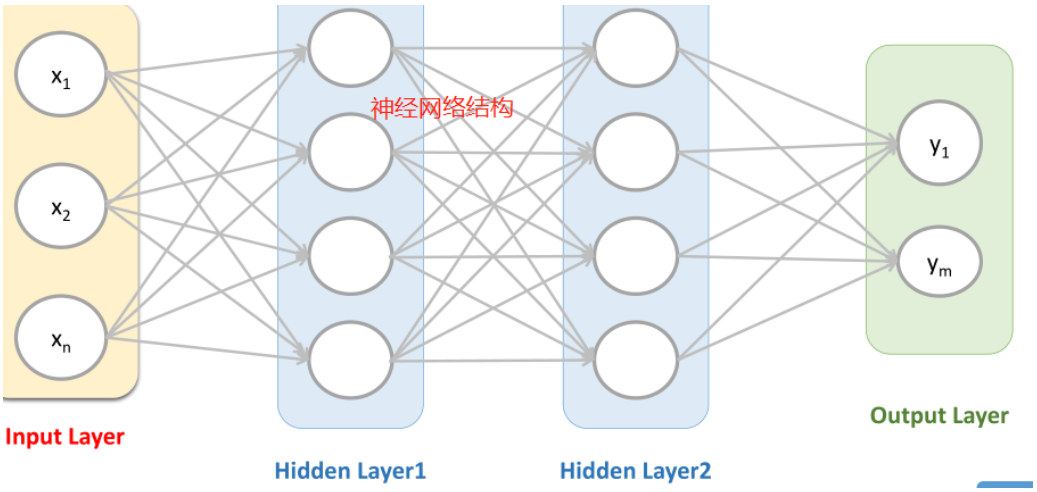
重要的视觉模型发展

AlexNet-》ZFnet->VGGNet->ResNet->MaskRCNN

神经网络和卷积神经网络的区别

正则化与Dropout

典型的结构与训练方式



卷积神经网络(Convolutional Neural Networks，CNN)，CNN可以有效的降低

反馈神经网络(传统神经网络)的复杂性，常见的CNN结构有LeNet-5、AlexNet、

ZFNet、VGGNet、GoogleNet、ResNet等等，其中在LVSVRC2015冠军ResNet是AlexNet的20多倍，是VGGNet的8倍；从这些结构来讲CNN发展的一

个方向就是层次的增加，通过这种方式可以利用增加的非线性得出目标函数的近似结构，同时得出更好的特征表达，但是这种方式导致了网络整体复杂性的增加，

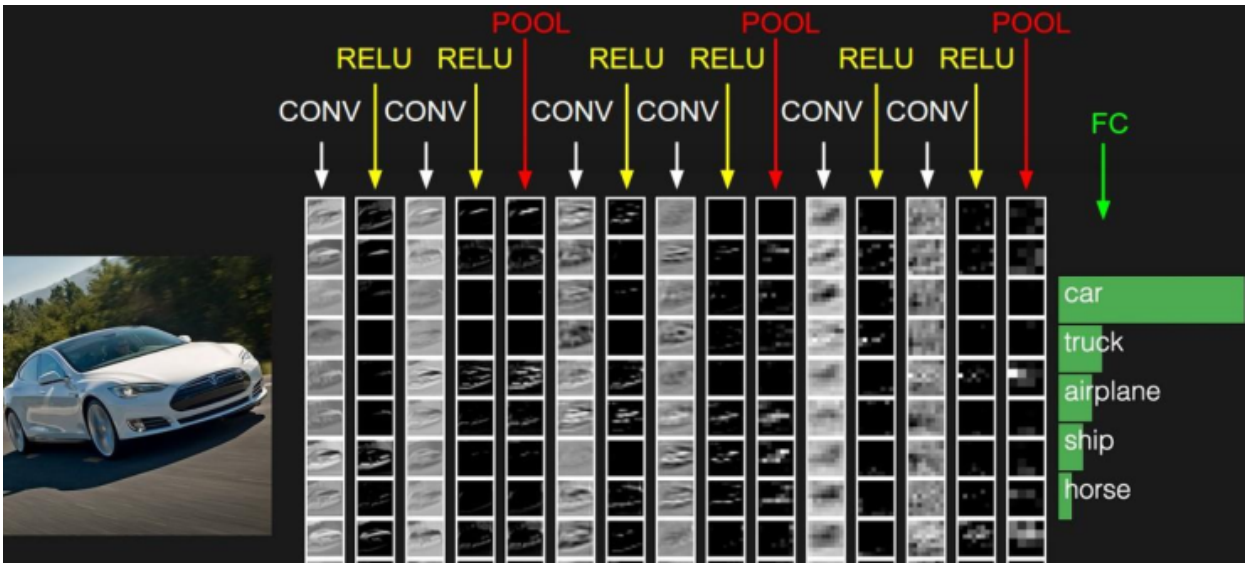
使网络更加难以优化，很容易过拟合。

CNN的应用主要是在图像分类和物品识别等应用场景应用比较多

卷积神经网络

保存了层级的网络结构

不同层次有不同的形式(运算)与功能



卷积神经网络-主要层次 特征提取层 特征映射层

数据输入层：Input Layer

卷积计算层：CONV Layer

ReLU激励层：ReLU Layer

池化层：Pooling Layer

全连接层：FC Layer

卷积神经网络-Input Layer

和神经网络/机器学习一样，需要对输入的数据需要进行预处理操作，原因是：输入数据单位不一样，可能会导致神经网络收敛速度慢，训练时间长

数据范围大的输入在模式分类中的作用可能偏大，而数据范围小的作用就有可能偏小

由于神经网络中存在的激活函数是有值域限制的，因此需要将网络训练的目标数据映射到激活函数的值域

S形激活函数在(0,1)区间以外区域很平缓，区分度太小。例如S形函数f(X)，f(100)与f(5)只相差0.0067

卷积神经网络-Input Layer

常见3种数据预处理方式

去均值

将输入数据的各个维度中心化到0

归一化

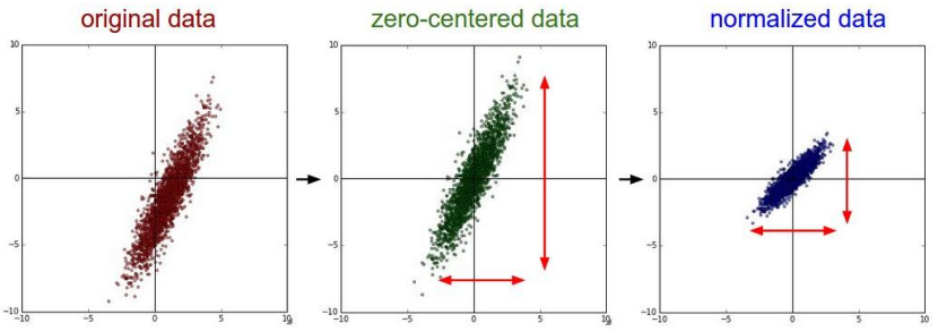
将输入数据的各个维度的幅度归一化到同样的范围

PCA/白化

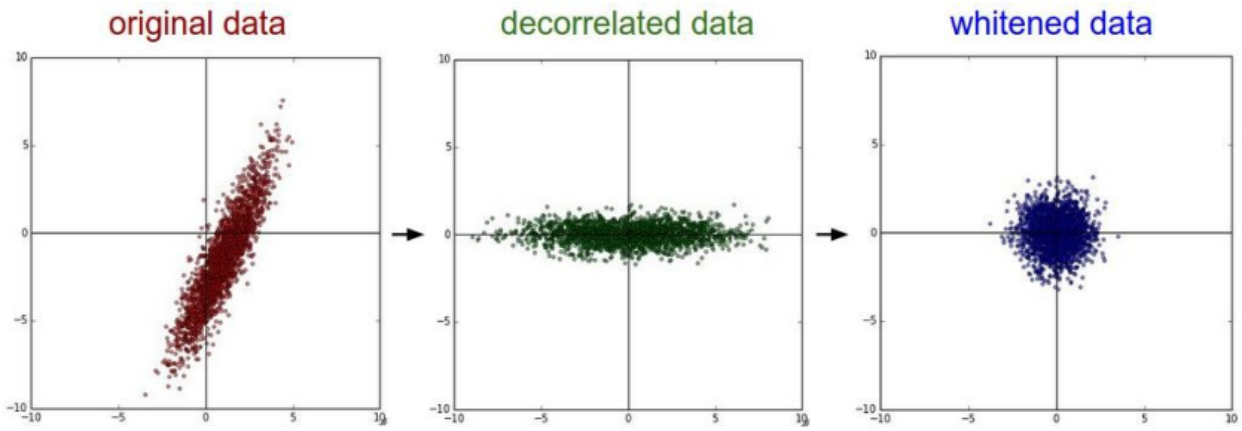
用PCA降维

白化是对数据的每个特征抽上的幅度归一化

去均值&归一化



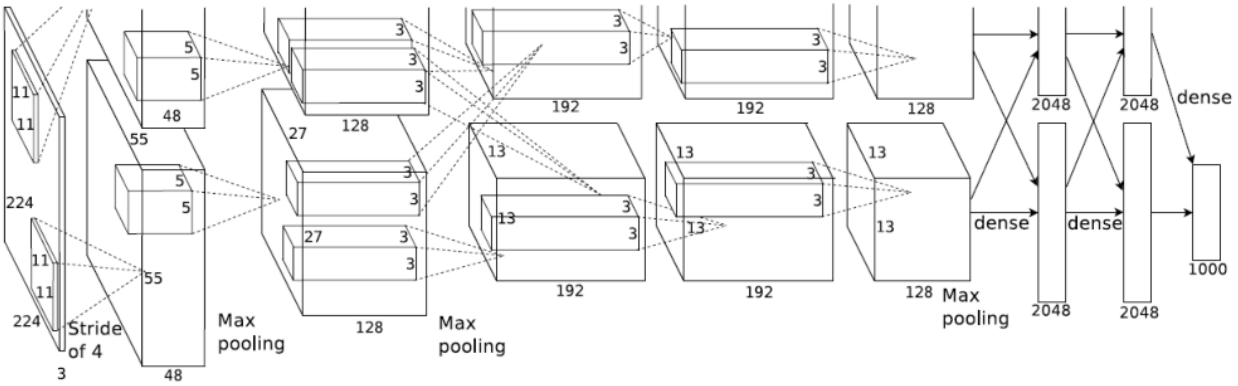
PCA&白化



人大脑识别图片过程

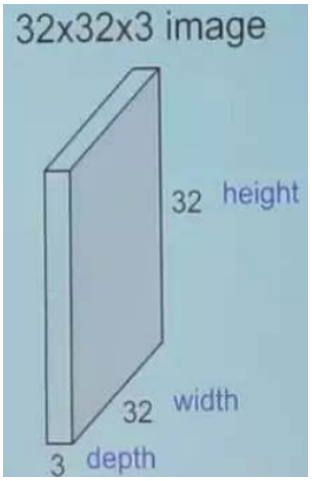
人的大脑在识别图片的过程中，会由不同的皮质层处理不同方面的数据，比如：颜色、形状、光暗等，然后将不同皮质层的处理结果进行合并映射操作，得出最终的结果值，第一部分实质上是一个局部的观察结果，第二部分才是一个整体的结果合并

基于人脑的图片识别过程，我们可以认为图像的空间联系也是局部的像素联系比较紧密，而较远的像素相关性比较弱，所以每个神经元没有必要对全局图像进行感知，只要对局部进行感知，而在更高层次对局部的信息进行综合操作得出全局信息



一个数据输入，假设为一个RGB的图片

在神经网络中，输入是一个向量，但是在卷积神经网络中，输入是一个多通道图像(比如这个例子中有3个通道)



卷积神经网络-CONV Layer

卷积计算层：CONV Layer

局部关联：每个神经元看做一个filter

窗口(receptive field)滑动，filter对局部数据进行计算

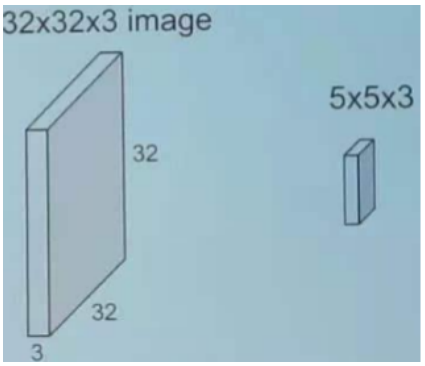
相关概念

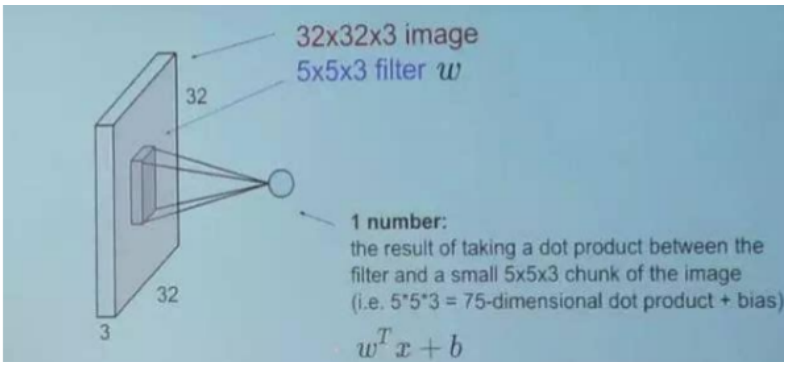
深度：depth

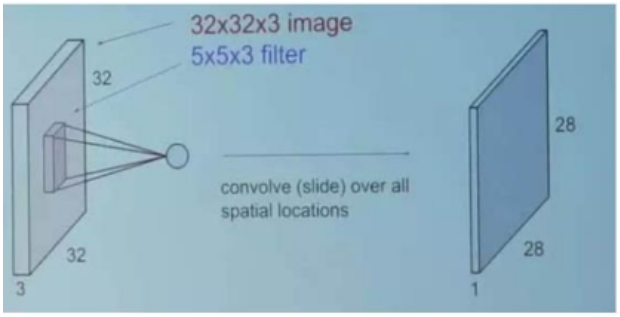
步长：stride

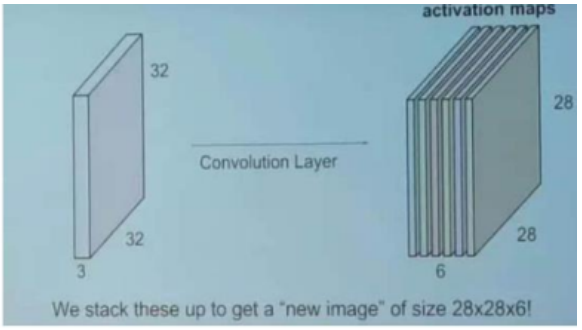
填充值：zero-padding

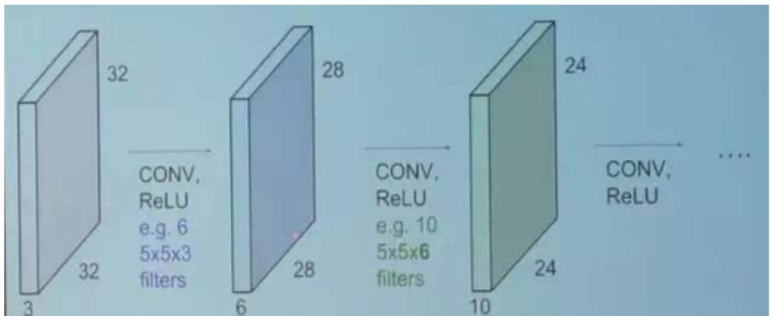
CONV过程参考：http://cs231n.github.io/assets/conv-demo/index.html

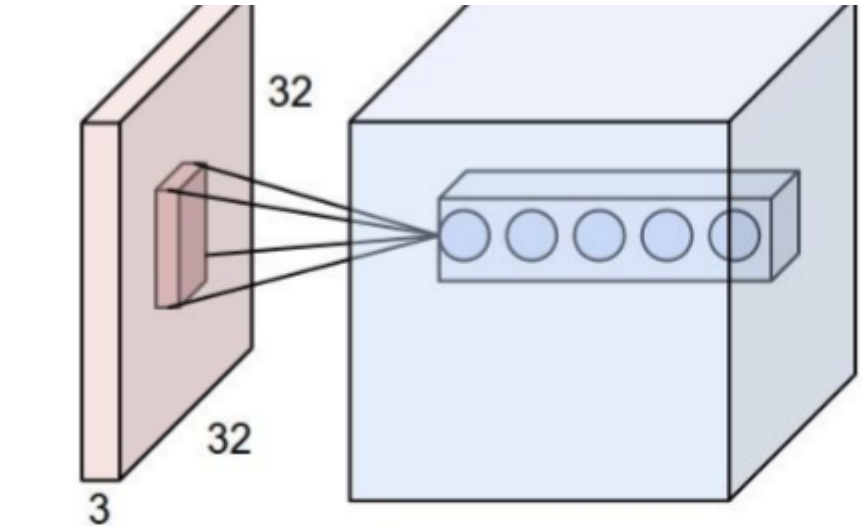


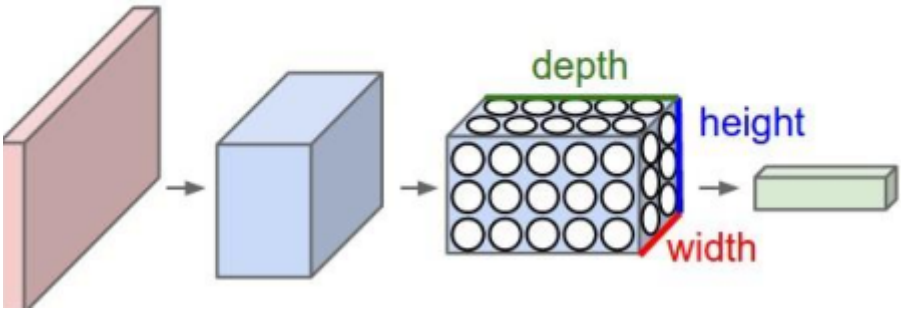


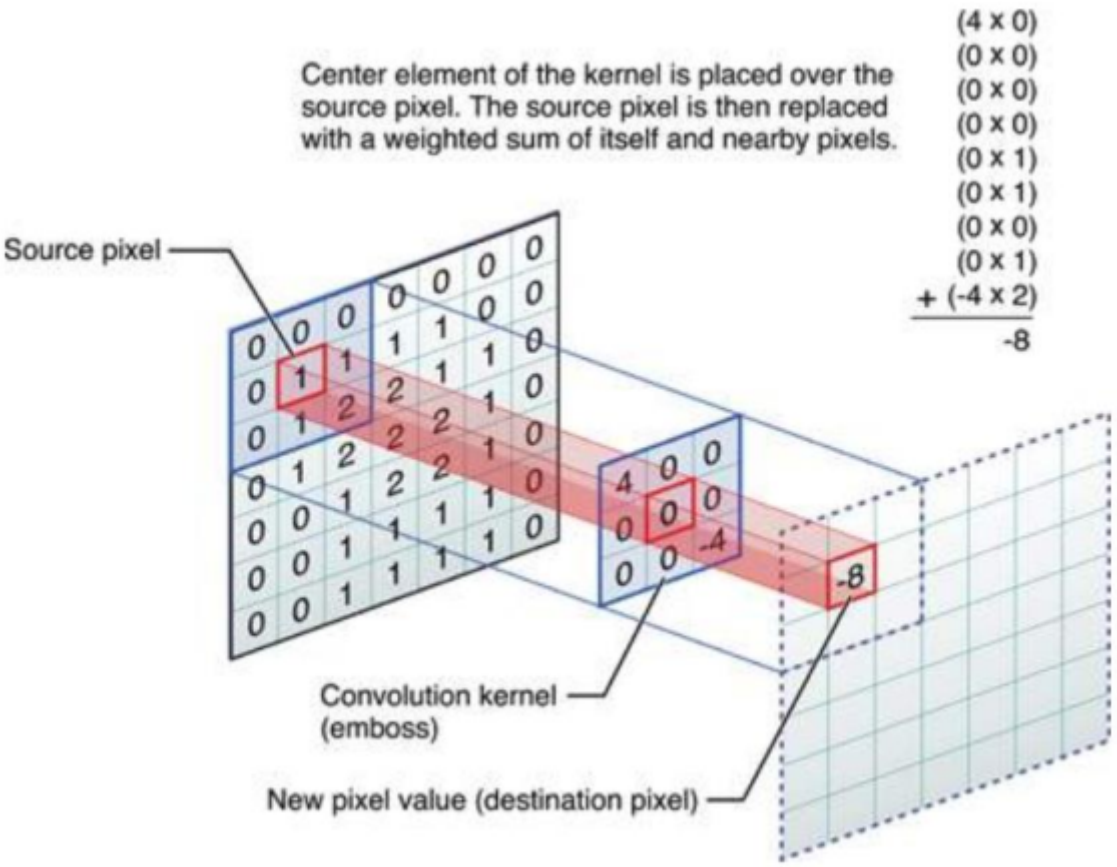












参数共享机制：假设每个神经元连接数据窗的权重是固定的

固定每个神经元的连接权重，可以将神经元看成一个模板；也就是每个神经元只

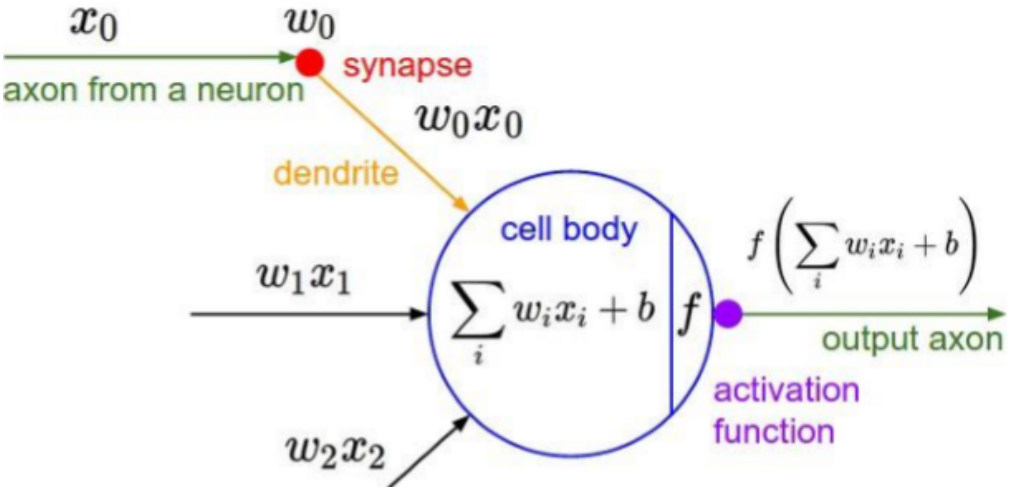
关注一个特性

需要计算的权重个数会大大的减少

一组固定的权重和不同窗口内数据做内积：卷积

卷积神经网络-ReLU Layer

将卷积层的输出结果做一次非线性的映射



常用非线性映射函数

Sigmoid(S形函数)

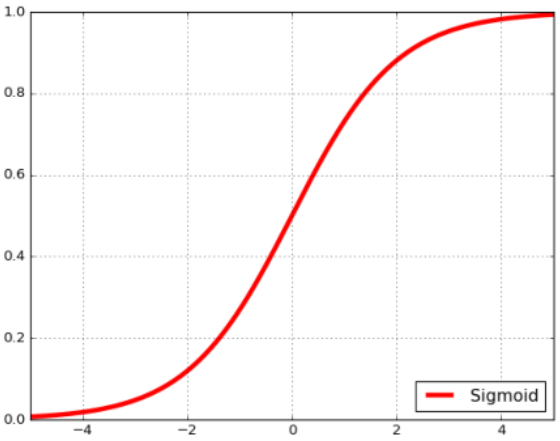
Tanh(双曲正切,双S形函数)

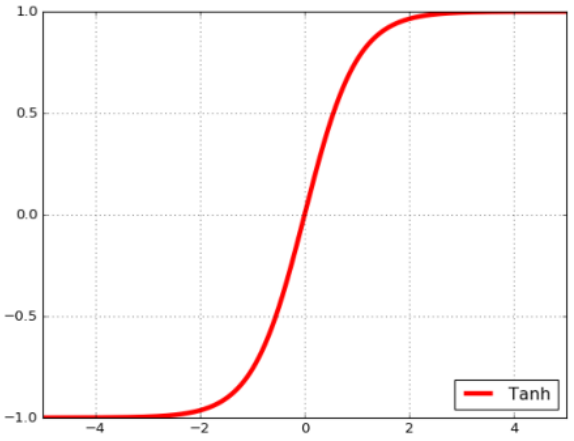
ReLU

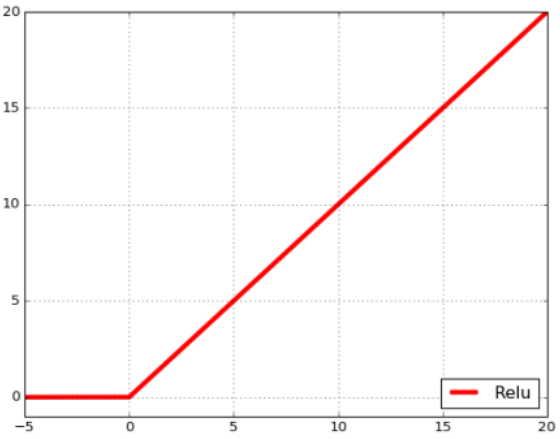
Leaky ReLU

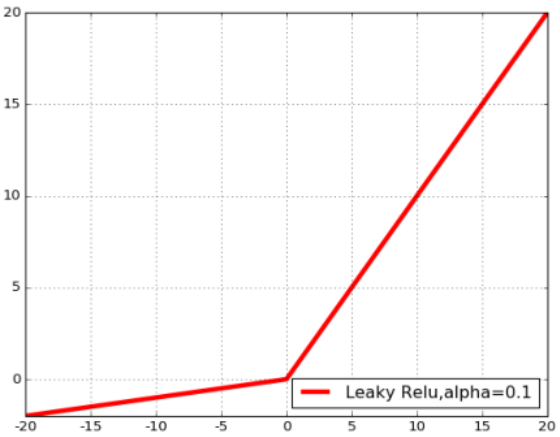
ELU

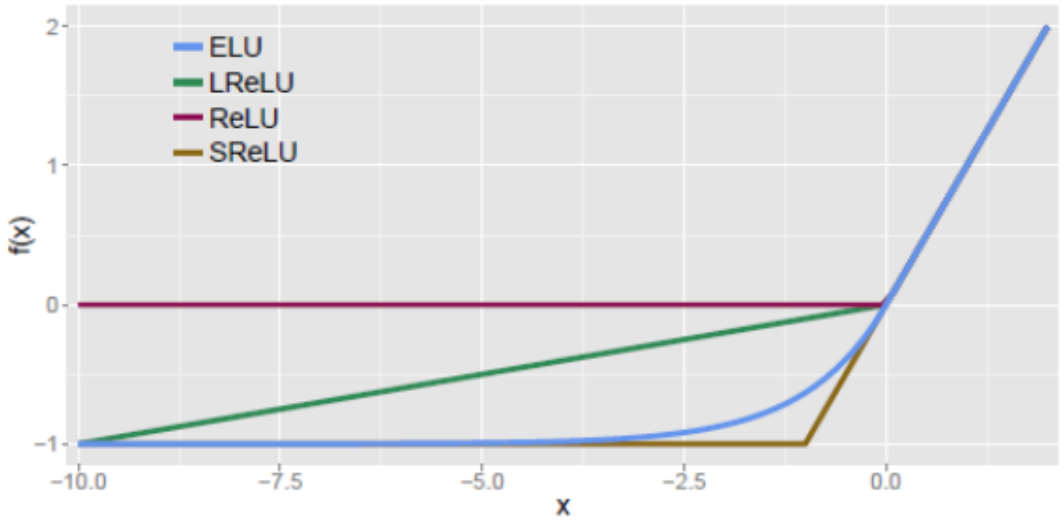
Maxout











The *exponential linear unit* (ELU) with $0 < \alpha$ is

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\left(\exp(x) - 1\right) & \text{if } x \leq 0 \end{cases}, \quad f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ f(x) + \alpha & \text{if } x \leq 0 \end{cases}$$

激励层建议

CNN尽量不要使用sigmoid，如果要使用，建议只在全连接层使用

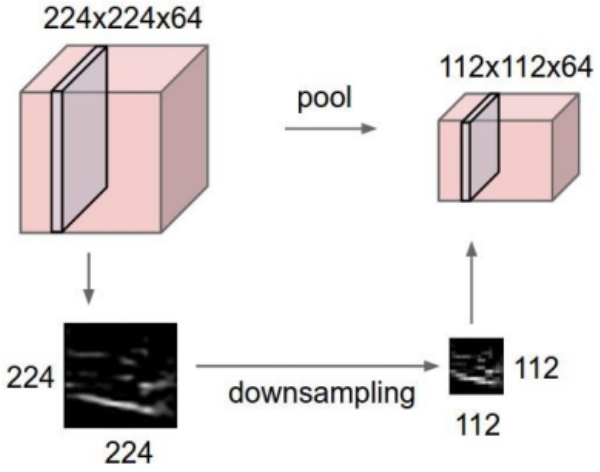
首先使用RELU，因为迭代速度快，但是有可能效果不佳

如果使用RELU失效的情况下，考虑使用Leaky ReLu或者Maxout，此时一般情况都可以解决啦

tanh激活函数在某些情况下有比较好的效果，但是应用场景比较少

卷积神经网络-Pooling Layer

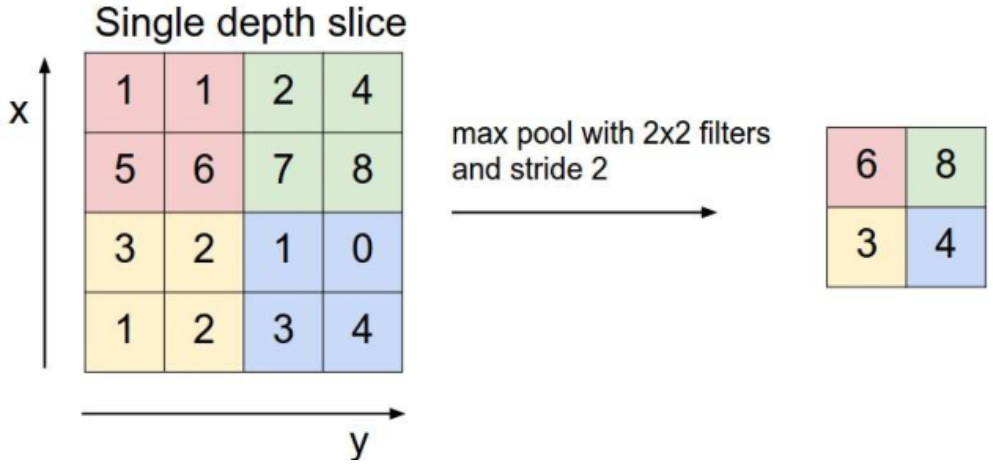
在连续的卷积层中间存在的就是池化层，主要功能是：通过逐步减小表征的空间尺寸来减小参数量和网络中的计算；池化层在每个特征图上独立操作。使用池化层可以压缩数据和参数的量，减小过拟合



在池化层中，进行压缩减少特征数量的时候一般采用两种策略：

Max Pooling：最大池化，一般采用该方式

Average Pooling：平均池化



卷积神经网络-FC

类似传统神经网络中的结构，FC层中的神经元连接着之前层次的所有激活输出；

换一句话来讲的话，就是两层之间所有神经元都有权重连接；通常情况下，在CNN中，FC层只会在尾部出现

一般的CNN结构依次为：

INPUT

[[CONV -> RELU] * N -> POOL?]*M

[FC -> RELU] * K

FC

卷积神经网络训练算法

和一般的机器学习算法一样，需要先定义Loss Function,衡量预测值和实际值之

间的误差，一般使用平方和误差公式

找到最小损失函数的W和b的值，CNN中常使用的是SGD

其实就是一般机器学习中的BP算法；SGD需要计算W和b的偏导，BP算法就是计

算偏导用的，BP算法的核心是求导链式法则。

$$\frac{\partial y}{\partial t}=\frac{\partial y}{\partial x}*\frac{\partial x}{\partial t} \qquad \frac{\partial y}{\partial t_i}=\sum_{l=1}^{m}\frac{\partial y}{\partial x_l}*\frac{\partial x_l}{\partial t_i}$$

使用BP算法逐级求解出ΔW和Δd的值

根据SGD(随机梯度下降)，迭代更新W和b

$$w_i^{+}=w_i-\eta\Delta w_i$$

$$d_i^{+}=d_i-\eta\Delta d_i$$

卷积神经网络优缺点

优点

共享卷积核(共享参数)，对高维数据的处理没有压力

无需选择特征属性，只要训练好权重，即可得到特征值

深层次的网络抽取图像信息比较丰富，表达效果好

缺点

需要调参，需要大量样本，训练迭代次数比较多，最好使用GPU训练

物理含义不明确，从每层输出中很难看出含义来

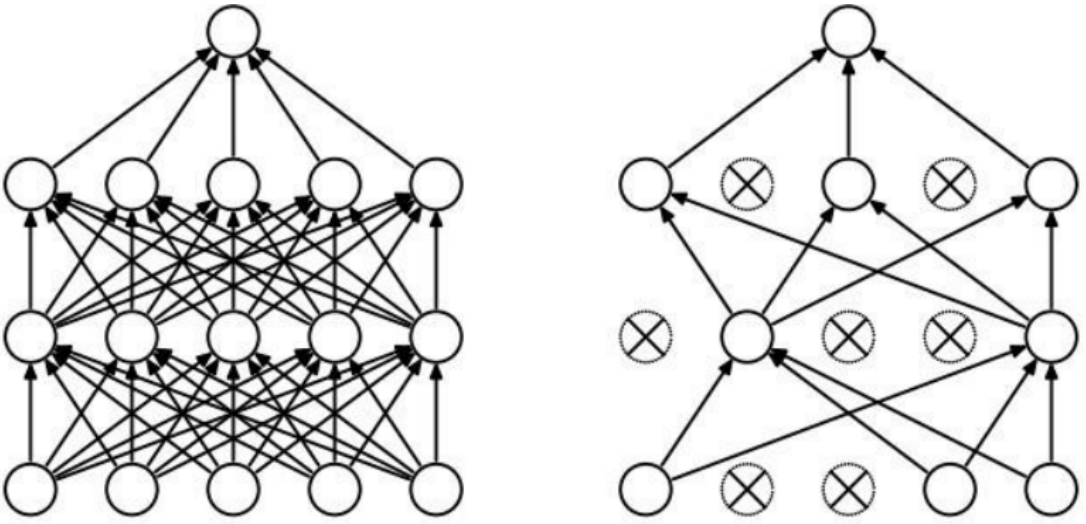
卷积神经网络正则化和Dropout

神经网络的学习能力受神经元数目以及神经网络层次的影响，神经元数目越大，神经网络层次越高，那么神经网络的学习能力越强，那么就有可能出现过拟合的问题

Regularization：正则化，通过降低模型的复杂度，通过在cost函数上添加一个正则项的方式来降低overfitting，主要有L1和L2两种方式

Dropout：通过随机删除神经网络中的神经元来解决overfitting问题，在每次迭代的时候，只使用部分神经元训练模型获取W和d的值，参考：《Dropout: A Simple Way to Prevent Neural Networks from Overfitting》

一般情况下，对于同一组训练数据，利用不同的神经网络训练之后，求其输出的平
均值可以减少overfitting。Dropout就是利用这个原理，每次丢掉一半左右的隐藏
层神经元，相当于在不同的神经网络上进行训练，这样就减少了神经元之间的依赖
性，即每个神经元不能依赖于某几个其它的神经元（指层与层之间相连接的神经元），使神经网络更加能学习到与其它神经元之间的更加健壮robust的特征。
另外
Dropout不仅减少overfitting，还能提高准确率。
正则化是通过给cost函数添加正则项的方式来解决overfitting，Dropout是通过直
接修改神经网络的结构来解决overfitting
卷积神经网络正则化和Dropout



(a) Standard Neural Net　　(b) After applying dropout.