卷积神经算子开发

# CNN卷积神经网络（42页补充）

卷积神经网络（Convolutional Neural Networks）是一类包含[卷积](https://baike.baidu.com/item/%E5%8D%B7%E7%A7%AF/9411006)计算且具有深度结构的[前馈神经网络](https://baike.baidu.com/item/%E5%89%8D%E9%A6%88%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C/7580523)，是[深度学习](https://baike.baidu.com/item/%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0/3729729)的代表算法之一。卷积神经网络具有[表征学习](https://baike.baidu.com/item/%E8%A1%A8%E5%BE%81%E5%AD%A6%E4%B9%A0/2140515)能力，能够按其阶层结构对输入信息进行平移不变分类，因此也被称为“平移不变人工神经网络。

## 历史

对卷积神经网络的研究可追溯至日本学者福岛邦彦提出的neocognitron模型。在其1979和1980年发表的论文中，福岛仿造生物的[视觉皮层](https://baike.baidu.com/item/%E8%A7%86%E8%A7%89%E7%9A%AE%E5%B1%82/10986729)设计了以“neocognitron”命名的神经网络。neocognitron是一个具有深度结构的神经网络，并且是最早被提出的深度学习算法之一，其隐含层由S层（Simple-layer）和C层（Complex-layer）交替构成。其中S层单元在[感受野](https://baike.baidu.com/item/%E6%84%9F%E5%8F%97%E9%87%8E/8989338)（receptive field）内对图像特征进行提取，C层单元接收和响应不同感受野返回的相同特征。通过neocognitron内部S/C层组合能够实现图像特征的提取与筛选，部分实现了卷积神经网络中[卷积层](https://baike.baidu.com/item/%E5%8D%B7%E7%A7%AF%E5%B1%82/22701737)（convolution layer）和池化层（pooling layer）的功能，被认为是启发卷积神经网络的开创性研究。

第一个卷积神经网络是1987年由Alexander Waibel等提出的TDNN时间延迟网络（Time Delay Neural Network）。TDNN是一个应用于语音识别问题的卷积神经网络，使用[FFT](https://baike.baidu.com/item/FFT/4766072)预处理的语音信号作为输入，其隐含层由2个一维卷积核组成，以提取频率域上的平移不变特征。由于在TDNN出现之前，人工智能领域在[反向传播算法](https://baike.baidu.com/item/%E5%8F%8D%E5%90%91%E4%BC%A0%E6%92%AD%E7%AE%97%E6%B3%95/522948)（Back-Propagation,BP）的研究中取得了突破性进展，因此TDNN得以使用BP框架内进行学习。在原作者的比较试验中，TDNN的表现超过了同等条件下的HMM [隐马尔可夫模型](https://baike.baidu.com/item/%E9%9A%90%E9%A9%AC%E5%B0%94%E5%8F%AF%E5%A4%AB%E6%A8%A1%E5%9E%8B/7932524)（HiddenMarkovModel,），而后者是二十世纪80年代语音识别的主流算法。

1988年，Wei Zhang提出了第一个二维卷积神经网络：平移不变人工神经网络，并将其应用于检测[医学影像](https://baike.baidu.com/item/%E5%8C%BB%E5%AD%A6%E5%BD%B1%E5%83%8F/4954291)。YannLeCun在1989年同样构建了应用于计算机视觉问题的卷积神经网络LeNet。LeNet包含两个卷积层，2个全连接层，共计6万个学习参数，规模远超TDNN和SIANN，且在结构上与现代的卷积神经网络十分接近。LeCun(1989)对权重进行随机初始化后使用了SGD随机梯度下降（Stochastic Gradient Descent）进行学习，这一策略被其后的深度学习研究所保留。此外，LeCun(1989)在论述其网络结构时首次使用了“卷积”一词，“卷积神经网络”也因此得名。LeCun的工作在1993年由[贝尔实验室](https://baike.baidu.com/item/%E8%B4%9D%E5%B0%94%E5%AE%9E%E9%AA%8C%E5%AE%A4/686816)（AT&T Bell Laboratories）完成代码开发并被部署于NCR的支票读取系统。但总体而言，由于数值计算能力有限、学习样本不足，加上同一时期以SVM[支持向量机](https://baike.baidu.com/item/%E6%94%AF%E6%8C%81%E5%90%91%E9%87%8F%E6%9C%BA/9683835)（Support Vector Machine）为代表的核学习（kernel learning）方法的兴起，这一时期为各类图像处理问题设计的卷积神经网络停留在了研究阶段，应用端的推广较少。

在LeNet的基础上，1998年YannLeCun及其合作者构建了更加完备的卷积神经网络LeNet-5并在手写数字的识别问题中取得成功。LeNet-5沿用了LeCun(1989)的学习策略并在原有设计中加入了池化层对输入特征进行筛选。LeNet-5及其后产生的变体定义了现代卷积神经网络的基本结构，其构筑中交替出现的卷积层-池化层被认为能够提取输入图像的平移不变特征。LeNet-5的成功使卷积神经网络的应用得到关注，微软在2003年使用卷积神经网络开发了OCR光学字符读取（Optical Character Recognition）系统。其它基于卷积神经网络的应用研究也得到展开，包括[人像识别](https://baike.baidu.com/item/%E4%BA%BA%E5%83%8F%E8%AF%86%E5%88%AB18/1705333)、[手势识别](https://baike.baidu.com/item/%E6%89%8B%E5%8A%BF%E8%AF%86%E5%88%AB19/12629655)等。

在2006年[深度学习](https://baike.baidu.com/item/%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0/3729729)理论被提出后，卷积神经网络的表征学习能力得到了关注，并随着数值计算设备的更新得到发展。自2012年的AlexNet开始，得到GPU计算集群支持的复杂卷积神经网络多次成为[ImageNet](https://baike.baidu.com/item/ImageNet/17752829)数据集上大规模视觉识别竞赛ILSVRC的优胜算法，包括2013年的ZFNet、2014年的VGGNet、GoogLeNet和2015年的ResNet。

## 结构

卷积神经网络中层顺序：输入层—卷积层—批归一化层—激活层—池化层—全连接层—输出层。

### 输入层

卷积神经网络的输入层可以处理多维数据，一维卷积神经网络的输入层接收一维或二维数组，其中一维数组通常为时间或频谱采样；二维数组可能包含多个通道；二维卷积神经网络的输入层接收二维或三维数组；三维卷积神经网络的输入层接收四维数组。由于卷积神经网络在[计算机视觉](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E8%A7%86%E8%A7%89/2803351)领域应用较广，因此许多研究在介绍其结构时预先假设了三维输入数据，即平面上的二维像素点和[RGB](https://baike.baidu.com/item/RGB/342517)通道。训练过程大多采用SGD随机[梯度下降](https://baike.baidu.com/item/%E6%A2%AF%E5%BA%A6%E4%B8%8B%E9%99%8D%E7%AE%97%E6%B3%95/1095627)的方式进行学习，一般而言，卷积神经网络的输入数据均需要进行预处理操作，比如去均值、尺寸缩放、将数值范围在[0~255]的原始像素归一化至[0~1]区间的数值缩放等等。有利于提升卷积神经网络的学习效率。

### 隐含层

常见卷积神经网络的隐含层包含批量归一化层、[卷积层](https://baike.baidu.com/item/%E5%8D%B7%E7%A7%AF%E5%B1%82/22701737)、激活层、池化层、全连接层、5种常见结构。传统的机器学习结构一般是没有隐层的或者只有单隐层，而CNN卷积网络中一般多个隐层通过非线性的方式进行连接，多隐层的网络结构具有更大数据的表达能力，可以从原始数据中学习到模式（特征），并通过逐层特征提取将数据表示的更加抽象，从而提高了分类的准确率。但这种提高并非没有上限，当隐层的数量过高时，由于反向传播算法出现的梯度消失现象，准确率可能会下降，随着深度卷积网络的发展，出现了resnet残差网络和densenet网络有效的解决了这个问题，其不再使用多个隐层的简单堆叠，而是通过short connection跳连的方式连接输入与输出。这样网络可以设置的更深，可达到几百甚至上千层而不会发生梯度消失的现象，alpha-GO和alpha-Zero就应用到了Resnet网络完成围棋的博弈方法。隐含层中的卷积层、批归一化层、全连接层包含需要参与训练的权重系数，激活层大部分没有需要进行训练的权重参数，但有些例如Prelu激活层需要。池化层不包含需要进行训练的权重参数，权重参数在根据损失函数与方法在模型训练过程中不断的更新。

### 输出层

卷积神经网络中输出层的上游通常是全连接层，因此其结构和工作原理与传统前馈神经网络中的输出层相同。对于图像分类问题，输出层使用逻辑函数或归一化指数函数（softmax）输出分类标签。在物体识别问题中，输出层可设计为输出物体的中心坐标、大小和分类。在图像语义分割中，输出层直接输出每个像素的分类结果。

# 算子（167页）

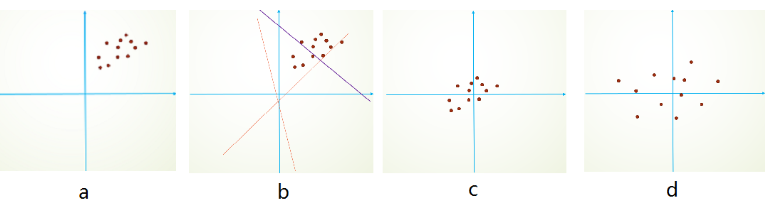
|  |  |  |  |
| --- | --- | --- | --- |
| **CNN层** | **具体模式** | **计算类型** | **操作个数** |
| **卷积层** | Conv | 矩阵内积 | 乘加： |
| Dilation-Conv | 矩阵内积 |  |
| Group-Conv | 矩阵内积 | 乘加： |
| depthwise-Conv | 矩阵内积 | 乘加： |
| **激活层** | LogSigmoid | 向量元素操作 | 指数：  加法：  除法： |
| Tanh | 向量元素操作 | 指数：  加法：  除法： |
| RELU | 向量元素操作 | 比较： |
| RELU6 | 向量元素操作 | 比较： |
| Leaky-ReLU | 向量元素操作 | 比较：  乘法： |
| **池化层** | 平均池化 | 向量元素操作 | 加法：  除法： |
| 最大池化 | 向量元素操作 | 比较： |
| **全连接层** |  | 矩阵向量乘积 | 乘加： |

算子是一个函数空间到函数空间上的映射O：X→X。广义上的算子可以推广到任何空间，对任何函数进行某一项操作都可以认为是一个算子。

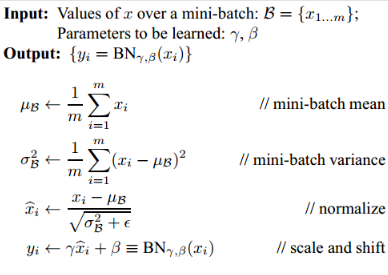
## 批归一化层（待完善概念）

网络训练过程中各层输入数据分布随着网络模型（主要是前若干层）中权重参数的变化而变化，使得训练深度神经网络难度变得越来越复杂。这通过要求较低的学习速率和仔细的参数初始化来减慢训练，并且使得训练具有饱和非线性的模型变得非常困难。我们将这种现象称为 internal covariate shift。早起解决方案就是使用较小的学习率和特殊初始化参数方式，并对对输入数据做白化处理，但是显然治标不治本。

介绍internal covariate shift需要说明下covariate shift的概念，covariate shift主要描述由于训练数据和测试数据存在分布的差异性，给网络的泛化性和训练速度带来了影响，经常使用的方法对输入数据进行归一化或者白化。比如简单线性分类，假设样本数据分布如a所示，网络模型中权重参数的初始化一般为零均值和较小的方差，训练初期完成y=wx+b拟合的如b图中橘色线分界线所示，经过多次迭代后将达到紫色线分界线，此时将存在很好的分类效果，但是将样本其归一化到0点附近如c图所示将会加快训练速度，如此更进一步的通过尺度变换拉大数据之间的相对差异性，训练过程将更快更好。



为提高每层网络的训练效率和效果需要针对每层网络进行Internal Covariate Shift，假设将每一层输出数据其归一化参数为0均值，1方差，那每层数据分布都是标准正太分布，导致其完全学习不到输入数据的特征，训练过程中提取的特征分布被归一化，需要稍作修改，加入可训练的参数做归一化BatchNorm，2015年论文中对其进行了详细的说明，公式如下[《Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift》](https://arxiv.org/pdf/1502.03167v3.pdf)。



### 输入

批归一化层的输入为四维张量 ，其中为分别表征张量的batch数量、通道长度、行长度与列长度。

### batchNorm处理

每个批归一化层批归一化卷积层包含scale比例和bias偏置两种类型参数，比例参数，与参数偏置参数，其中表征张量通道数，批量归一化层参数如下：

num\_features：输入张量通道数;

eps ： 常数确保除法计算结果稳定性，避免除0操作；

momentum ：滑动平均参数；

通过池化处理得到的输出张量大小为, 其中为分别表征张量的batch数量，通道长度、行长度、列长度，与张量输入维度一致，batch-norm运算可以形式化的表示为

## 卷积层（convolutional layer）

卷积层的功能是对输入数据进行特征提取，比如图像中可能有很多种边缘特征，包括对角线特征、三角形特征、圈特征，甚至麻花形的特征等。为了提取出图像中不同的特征，神经网络需要有多个不同的卷积核。提取出不同的特征之后，每一个神经网络层会输出多个特征图（或者说有多个输出通道），每一个特征图代表一种特征。

卷积层内部包含多个卷积核，组成卷积核的每个元素都对应一个权重系数和一个偏差量（bias vector），类似于一个前馈神经网络的神经元（neuron）。卷积层内每个神经元都与前一层中位置接近的区域的多个神经元相连，区域的大小取决于卷积核的大小，在文献中被称为“[感受野](https://baike.baidu.com/item/%E6%84%9F%E5%8F%97%E9%87%8E/8989338)（receptive field）”，其含义可类比[视觉皮层](https://baike.baidu.com/item/%E8%A7%86%E8%A7%89%E7%9A%AE%E5%B1%82/10986729)细胞的感受野。卷积核在工作时，会有规律地扫过输入特征，在感受野内对输入特征做矩阵元素乘法求和并叠加偏差量

### 数学定义

定义任意两个信号的卷积为这里的代表卷积的运算符号，是中间变量，两个信号的卷积仍是以t为变量的信号。离散的信号的卷积和：（补充CNN中的卷积处理与常规定义的区别）

### 输入

卷积层的输入一般为三维张量 ，其中为分别表征输入张量的通道长度、行长度与列长度。

### 普通Conv处理

普通卷积层包含weights权重和biases偏置两种类型参数，权重参数，与参数偏置参数，其中分别表征卷积核个数、卷积核内部通道长度、行长度与列长度。卷积层参数如下：

kernel\_size ：分别为卷积操作的滑动窗行与列方向大小；

stride ：分别为卷积滑动窗行与列方向步近长度，通常为卷积核大小；

padding ：分别为输入张量各维度各边界要补齐0的层数；

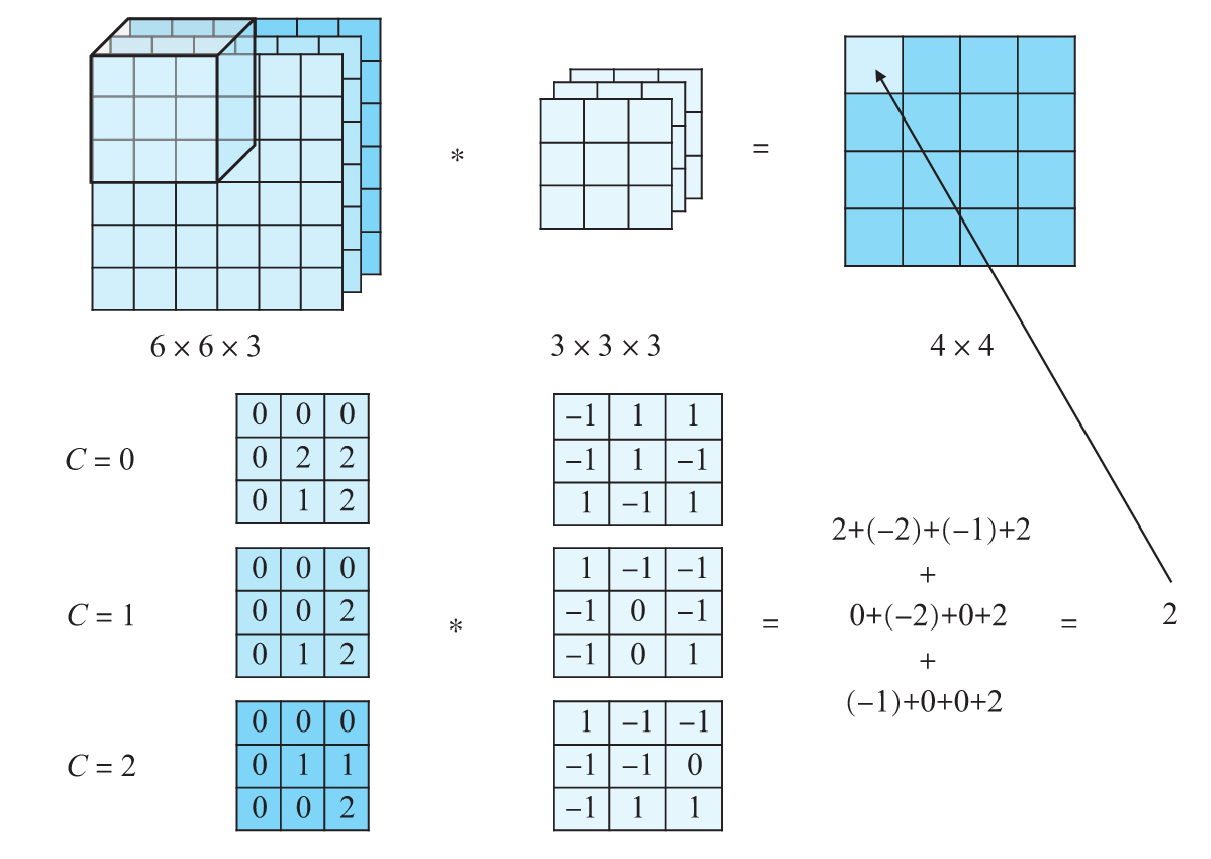
通过卷积处理得到的输出张量大小为, 其中为分别表征输出张量的通道长度、行长度、列长度，数值可通过输入张量尺寸与卷积层参数计算获取具体公式如下：

分别表征行方向与列方向padding补零个数，分别表征行方向与列方向的步进长度。

普通卷积计算运算可以形式化的表示为：

注：公式中的为张量 padding扩充后的张量，考虑到和计算公式采用加1向下取整的方式，经过处理后的与需进行判断,当中索引值超过其数组范围时，返回数值为0。

卷积示例：该神经网络卷积层的输入是3个6×6大小的特征图（即6×6×3的三维矩阵），分别表示对角线特征、圈特征、三角形特征；卷积核是3个不同的3×3卷积核（即3×3×3的三维矩阵），对应输入的3个不同的特征图；二者卷积输出一个4×4大小的特征图。在卷积运算中，输入特征图的通道数和卷积核的通道数必须一致。如果输入有3个特征图（也称为3个特征通道），就需要有3个卷积核来共同计算出最终的输出特征图。



### Dilation Conv处理

dilation卷积层包含weights权重和biases偏置两种类型参数，权重参数，与参数偏置参数，其中分别表征卷积核个数、卷积核内部通道长度、行长度与列长度。卷积层参数如下：

kernel\_size ：分别为卷积操作的滑动窗行与列方向大小；

dilation ：分别为卷积滑动窗中各有效元素间的距离；

stride ：分别为卷积滑动窗行与列方向步近长度，通常为卷积核大小；

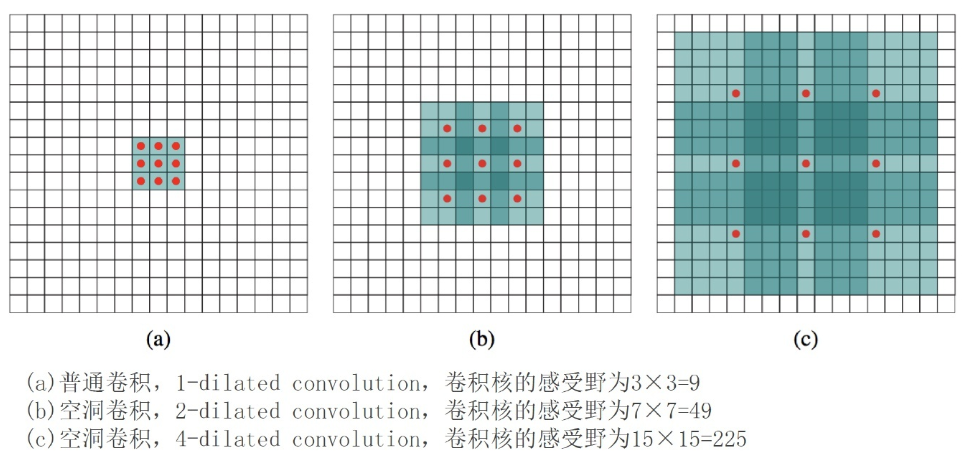
padding ：分别为输入张量各维度各边界要补齐0的层数；

通过卷积处理得到的输出张量大小为, 其中为分别表征输出张量的通道长度、行长度、列长度，数值可通过输入张量尺寸与卷积层参数计算获取具体公式如下：

分别表征行方向与列方向padding补零个数，分别表征行方向与列方向的步进长度。分别变成行方向与列方向的卷积核映射到输入特征图的张量在行列平面的元素间隔长度。

dilation卷积计算运算可以形式化的表示为：

注：公式中的为张量 padding扩充后的张量，考虑到和计算公式采用加1向下取整的方式，经过处理后的与需进行判断,当中索引值超过其数组范围时，返回数值为0。



### Group Conv处理

分组卷积Group Conv层包含weights权重和biases偏置两种类型参数，权重参数，与参数偏置参数，其中分别表征卷积核个数、卷积核内部通道长度、行长度与列长度。卷积层参数如下：

kernel\_size ：分别为卷积操作的滑动窗行与列方向大小；

dilation ：分别为卷积滑动窗中各有效元素间的距离；

stride ：分别为卷积滑动窗行与列方向步近长度，通常为卷积核大小；

padding ：分别为输入张量各维度各边界要补齐0的层数；

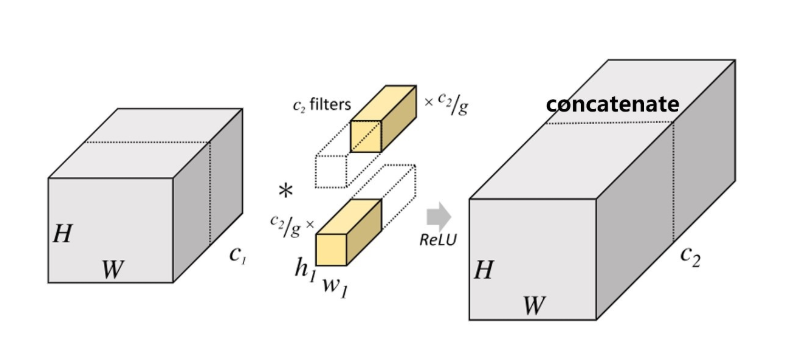
group ：分别为输入通道的分组个数；

通过卷积处理得到的输出张量大小为, 其中为分别表征输出张量的通道长度、行长度、列长度，数值可通过输入张量尺寸与卷积层参数计算获取具体公式如下：

分别表征行方向与列方向padding补零个数，分别表征行方向与列方向的步进长度。分别变成行方向与列方向的卷积核映射到输入特征图的张量在行列平面的元素间隔长度。

分组卷积计算运算可以形式化的表示为：

注：公式中的为张量 padding扩充后的张量，考虑到和计算公式采用加1向下取整的方式，经过处理后的与需进行判断,当中索引值超过其数组范围时，返回数值为0。



### Depth-wise Conv处理

分组卷积等于输入通道的个数Group Conv分组卷积就变成了Depth-Wise-Conv深度可分离卷积，Depth-Wise-Conv层包含weights权重和biases偏置两种类型参数，权重参数，与参数偏置参数，其中分别表征卷积核个数、卷积核内部通道长度、行长度与列长度。卷积层参数如下：

kernel\_size ：分别为卷积操作的滑动窗行与列方向大小；

dilation ：分别为卷积滑动窗中各有效元素间的距离；

stride ：分别为卷积滑动窗行与列方向步近长度，通常为卷积核大小；

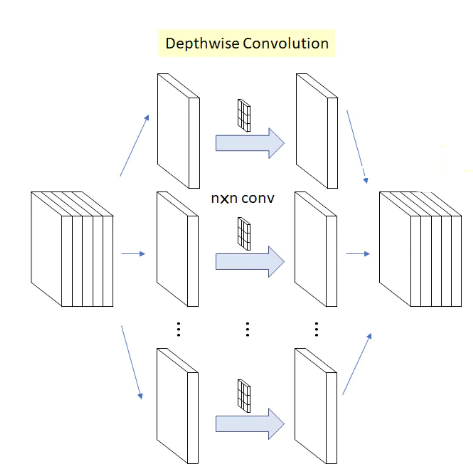
padding ：分别为输入张量各维度各边界要补齐0的层数；

通过卷积处理得到的输出张量大小为, 其中为分别表征输出张量的通道长度、行长度、列长度，数值可通过输入张量尺寸与卷积层参数计算获取具体公式如下：

分别表征行方向与列方向padding补零个数，分别表征行方向与列方向的步进长度。分别变成行方向与列方向的卷积核映射到输入特征图的张量在行列平面的元素间隔长度。

分组卷积计算运算可以形式化的表示为：

注：公式中的为张量 padding扩充后的张量，考虑到和计算公式采用加1向下取整的方式，经过处理后的与需进行判断,当中索引值超过其数组范围时，返回数值为0。为求余处理算子。



## 激活层（activation）

神经网络中的每个神经元节点接收上一层神经元的输出值作为本神经元的输入值，并将输入值传递给下一层，在多层神经网络中上层神经节点的输出和下层节点的输入之间具有一个函数关系，这个函数称为激活函数或激励函数，如果不用激励函数，每层节点输入与上层节点输出为线性关系，网络的逼近能力有限。无法实现更加强大的网络表达能力。（不再是输入的线性组合，而是几乎可以逼近任意函数）。

早期研究神经网络主要采用sigmoid函数或者tanh函数，输出有界，很容易充当下一层的输入。近些年Relu函数及其改进型（如Leaky-ReLU、Tanh、ReLU、RELU6、PReLU、等）在多层神经网络中应用比较多。下面我们来总结下这些激活函数：

### 输入

卷积层的输入一般为三维张量 ，其中为分别表征输入张量的通道长度、行长度与列长度。

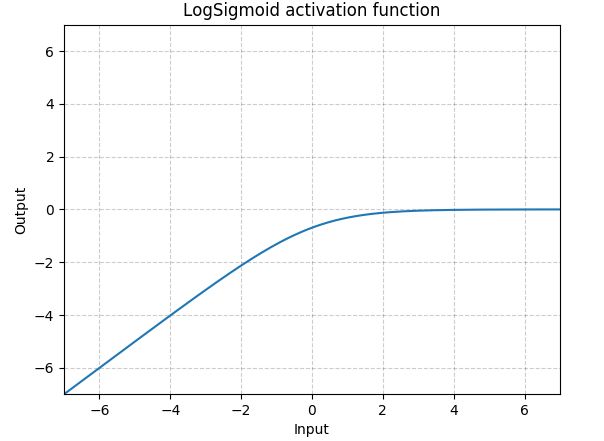
### LogSigmoid函数

LogSigmoid是Sigmoid中参数为1的情况，Sigmoid函数解析表达与求导公式如下所示：

LogSigmoid函数解析表达、求导公式和对应的曲线如下所示：

通过激活处理得到的输出张量大小为, 其中为分别表征输出张量的通道长度、行长度、列长度，数值与输入张量尺寸一直具体公式如下，后面讨论的其他激活函数与Sigmoid处理一致，将即可，后续不在反复列出。

Sigmoid函数计算运算可以形式化的表示为：

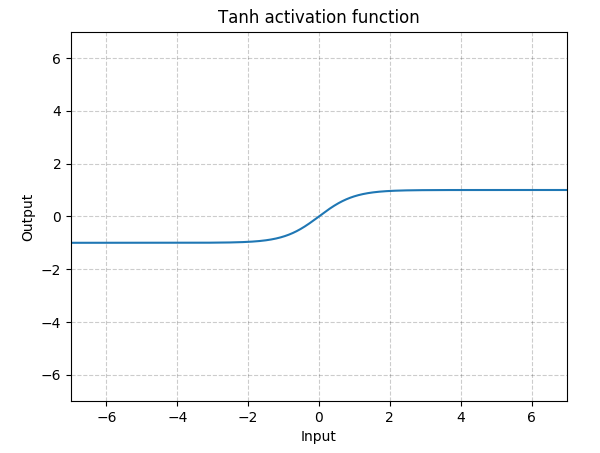


Sigmoid是[BP神经网络](https://baike.baidu.com/item/BP%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C)中常用的非线性作用函数，能够把输入的连续实值变换为0和1之间的输出，即对于输入非常大的负数，输出为0；对于输入非常大的正数，输出为1。函数都是连续的、单调递增的数值函数，常被应用于基于BP（误差反向传播）算法的神经网络中。

sigmoid函数曾经被经常使用，不过近年来，越来越少了。主要是因为它固有的一些缺点。1）在深度神经网络中梯度反向传递容易存在梯度消失现象，神经网络的权重参数数值大多在-1和1之间，由反向传播算法的数学推导和Sigmoid函数的导数可以获取，梯度从后向前传播时每传递一层梯度值都会减小为原来的0.25倍，针对神经网络模型较深的情况，梯度在经过多层后将变得很小接近于零值，无法更新权重参数；2）：Sigmoid输出数值不是0均值。导致对求局部梯度则都为正，这样在反向传播的过程中要么都往正方向更新，要么都往负方向更新。可以通过设置batch-size可以缓解这个问题；3）Sigmoid函数中存在幂运算，计算求解耗时加长，对于规模比较大的深度网络，将会较大地增加训练时间。

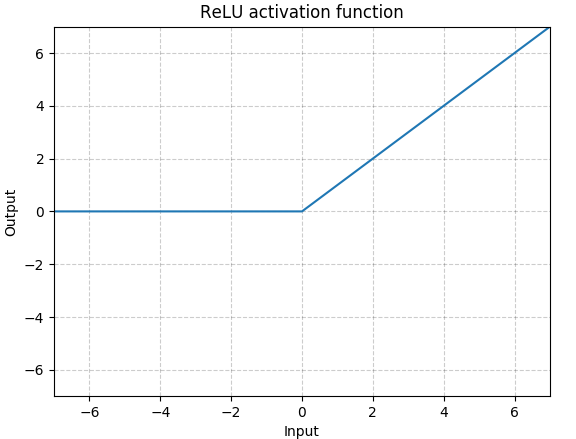
### Tanh函数

Tanh为双曲正切函数，其解析表达、求导公式和对应的曲线如下所示：



### ReLU函数

RELU线性整流函数（Rectified Linear Unit）又称修正线性单元，其解析表达、求导公式和对应的曲线如下所示：



RELU是一种[人工神经网络](https://baike.baidu.com/item/%E4%BA%BA%E5%B7%A5%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C)中常用的激活函数，相比于传统的神经网络激活函数，诸如[逻辑函数](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E5%87%BD%E6%95%B0)（Logistic sigmoid）和tanh等[双曲函数](https://baike.baidu.com/item/%E5%8F%8C%E6%9B%B2%E5%87%BD%E6%95%B0)，线性整流函数有着以下几方面的优势：1）参考仿生物学原理，相关大脑方面的研究表明生物神经元的信息编码通常是比较分散及稀疏的，通常情况下，大脑中在同一时间大概只有1%-4%的神经元处于活跃状态。使用线性修正以及正则化可以对机器神经网络中神经元的活跃度（即输出为正值）进行调试；而逻辑函数Sigmoid在输入为0时达到0.5，即已经是半饱和的稳定状态，不够符合实际生物学对模拟神经网络的期望。2）更加有效率的[梯度下降](https://baike.baidu.com/item/%E6%A2%AF%E5%BA%A6%E4%B8%8B%E9%99%8D)以及反向传播：避免了梯度爆炸和梯度消失问题，3）简化计算过程：没有了其他复杂激活函数中诸如指数函数的影响；同时活跃度的分散性使得神经网络整体计算成本下降。

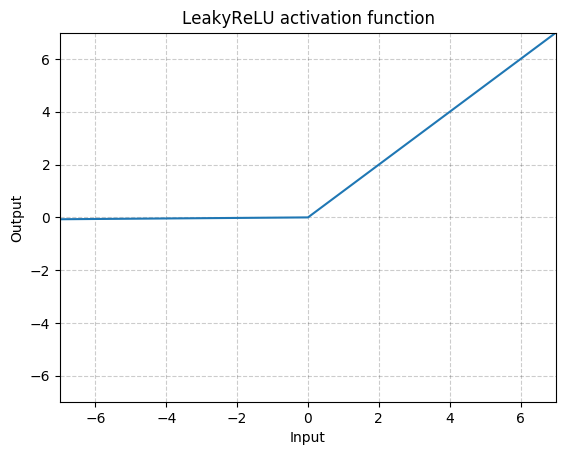
### Relu6函数

,

### Leaky-ReLU函数

Leaky ReLU带泄露线性整流函数（Rectified Linear Unit）又称修正线性单元，其解析表达、求导公式和对应的曲线如下所示：

,



## 池化层

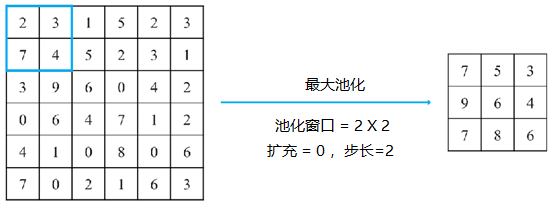
CNN卷积神经网络中池化层可以降低图片的尺寸，例如，输入图片或特征图的大小为100×100，经过池化，可能变成50×50。通过池化层可以减少后续卷积等操作所需的权重参数数量和计算量，达到缩减模型规模提高计算效率的效果，同时权重参数的降低可以抑制训练过程中的过拟合显现，提高所提取特征的鲁棒性，池化层一般没有参数，训练时很简单。存在多种类型的池化方式，例如最大池化（max pooling）、平均池化（avg pooling）、L2池化等。这里我们重点介绍平均池化和最大值池化。

### 输入

池化层的输入一般为三维张量 ，其中为分别表征张量的通道长度、行长度、列长度。

### 最大池化处理

最大池化是一种常用的池化方法，在池化窗口内找最大值作为输出。以下图为例，假设池化窗口为2×2，步长为2，不做边界扩充，从输入特征图的左上角的2×2子矩阵找到最大值7作为第1个输出，池化窗口在输入特征图上右移2格后找到最大值5作为第2个输出，池化窗口继续右移2格找到最大值3作为第3个输出，继续向下滑动池化窗口可以得到所有的输出值。最大池化法仅保留池化窗口内特征的最大值，可以提高特征的鲁棒性。



每个池化层通过自身的数据进行计算无需额外数据参与，平均池化层计算所需的参数如下：

kernel\_size ：分别为池化操作的滑动窗行与列方向大小；

stride ：分别为池化滑动窗行与列方向步近长度，通常为池化核大小；

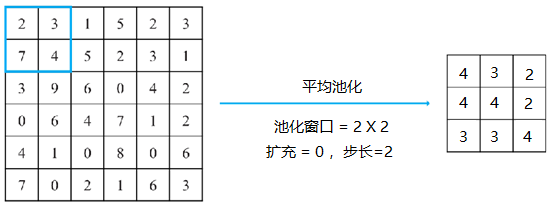
padding ：分别为输入张量各维度各边界要补齐0的层数；

通过池化处理得到的输出张量大小为, 其中为分别表征张量的通道长度、行长度、列长度，根据最大池化输入的张量与池化层参数，的具体计算公式如下，表示向下取整处理。

平均池化运算可以形式化的表示为：

### 平均池化处理

平均池化法也是一种常见的池化方法。该方法在池化窗口内对所有的数取平均值，以下图为例，假设池化窗口为2×2，步长为2，不做边界扩充，从输入特征图的左上角的2×2子矩阵找到平均值4作为第1个输出，池化窗口在输入特征图上右移2格后找到平均值3作为第2个输出，池化窗口继续右移2格找到平均值2作为第3个输出，继续向下滑动池化窗口可以得到所有的输出值。平均池化法会把图像的一些特征平均化，也就是模糊化。



平均池化层通过自身的数据进行计算无需额外数据参与，平均池化层计算所需的参数如下：

kernel\_size ：分别为池化操作的滑动窗行与列方向大小；

dilation ：分别为池化滑动窗中各有效元素间的距离；

stride ：分别为池化滑动窗行与列方向步近长度，通常为池化核大小；

padding ：分别为输入张量各维度各边界要补齐0的层数；

通过池化处理得到的输出张量大小为, 其中为分别表征张量的通道长度、行长度、列长度，根据平均池化输入的张量与池化层参数，的具体计算公式如下，表示向下取整处理。

最大池化运算可以形式化的表示为：

## 全连接层

卷积层和池化层构成特征提取器，而全连接层（fully-connected layer）是分类器，主要对前一层提取到的特征进行综合。全连接层将特征提取得到的高维特征图映射成一维特征向量，该特征向量包含所有特征信息，可以转化为最终分类为各个类别的概率。例如，一个224×224大小的输入图片经过多层卷积和池化，可能变成4096个1×1大小的特征图，根据这4096个特征可以做一个全连接层，来判定最后是猪狗猫牛羊中的哪一个。全连接层具有全连接的特性，全连接层每一个输出节点均与前一层所有节点相连，由于其运算处理所需的权重参数量规模较大。

### 输入

全连接层输入一维张量 ，其中表征张量输入的节点数量。

### 处理

全连接层包含weights权重和biases偏置两种类型参数，权重参数，偏置参数其中分别表征输出节点个数、列长度卷积层计算所需的参数如下：

## 输出层

### 输入

全连接层输入一维张量 ，其中表征张量输入的节点数量即分类的种类总量。

### Softmax逻辑回归（50页补充说明）

softmax逻辑回归模型是logistic回归模型在多分类问题上的推广，softmax对输出进行归一化，输出分类概率。例如：在多分类问题中，类标签y可以取两个以上的值。Softmax回归模型对于诸如MNIST手写数字分类等问题是很有用的，Softmax归一化指数函数其解析表达、求导公式和对应的曲线如下所示：

，

表征样本的总分类数。

通过softmax处理得到的输出张量大小为，softmax运算可以形式化的表示为：