# Improvements on CNNs

## Convolutional layer

卷积滤波器是一个针对底层图像块的广义线性模型，在潜在概念的实例是线性可分的时候抽象化效果非常好。以下是一些提高其表达能力的工作。

#### Tiled Convolution

CNN中的权值共享可以大大减少参数的数量，并且可以提供一定的平移不变性，但是在其他不变性上，模型容易被约束。Tiled Convolution是一种能够平铺和并联来学习旋转和缩放不变性特征的CNN变种。

在普通的CNN中，对于同一层同一个特征映射，对不同感受野的权值是共享的，而在TCNN中依旧保留了局部感知的特点，但只有间隔为（人为设定）的感受野会共享权值，其他情况均相互独立。如果等于1，那么将退化为传统的CNN。

在NORB和CIFAR-10数据集上，实践表现当时结果最佳，且在小型时间序列数据集上比传统CNN表现更佳。

#### Transposed Convolution

反卷积可以被看作传统卷积的反向传播过程，也被称作或者。大多数文献使用的称呼。和传统卷积层将多个输入与卷积层的一个输出相连接不同，反卷积将一个输入与多个输出相联。相当于输入特征映射，还原原图像。反卷积的步长给了特征映射一个放大因子。特别的，反卷积会先对输入基于步长和边距填充值进行上采样，然后对上采样后的输入进行卷积操作。近年来，反卷积在可视化、识别、定位、语义分割、视觉问答、超分辨率得到了广泛应用。

#### Dilated Convolution

扩张卷积是近期一种将额外的超参数（相对于需要通过学习获得的参数，超参数由人预先设定）引入CNN的进展，通过在滤波器周围嵌入零元素，在参数数目不变的前提下，增大了感受野的大小，让网络可以覆盖更多相关联的信息。这对于哪些需要巨大的感受野以做出预测的任务而言非常重要。

以一维卷积为例，扩张倍数为，表示倍扩张卷积运算，卷积核为，大小为的卷积核对信号做卷积运算，可以表示为因为卷积核扩张的间隙均为0，所以求和的时候只计算有效的位。这种计算也可以直接应用在2D卷积运算中，如下图所示。

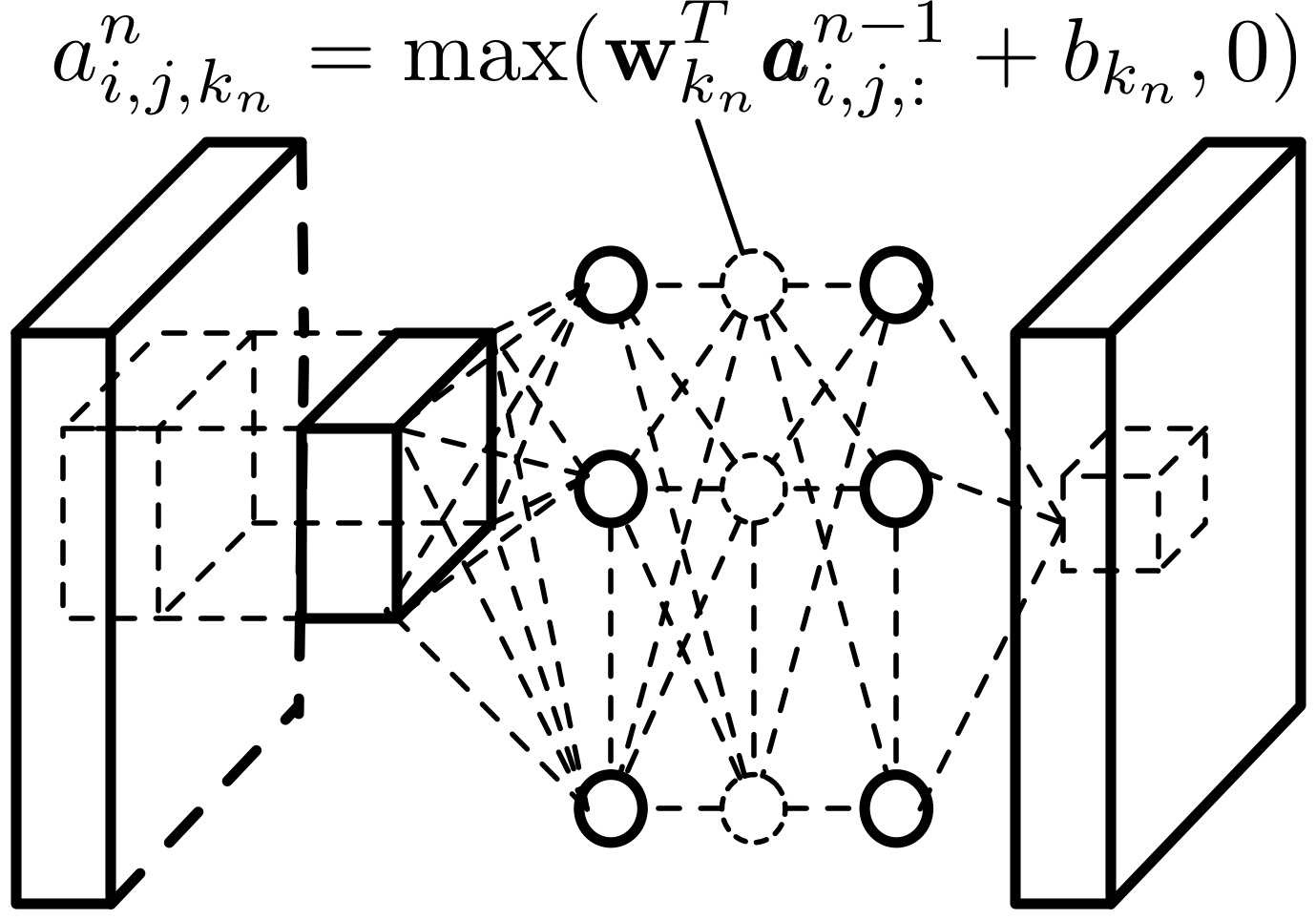
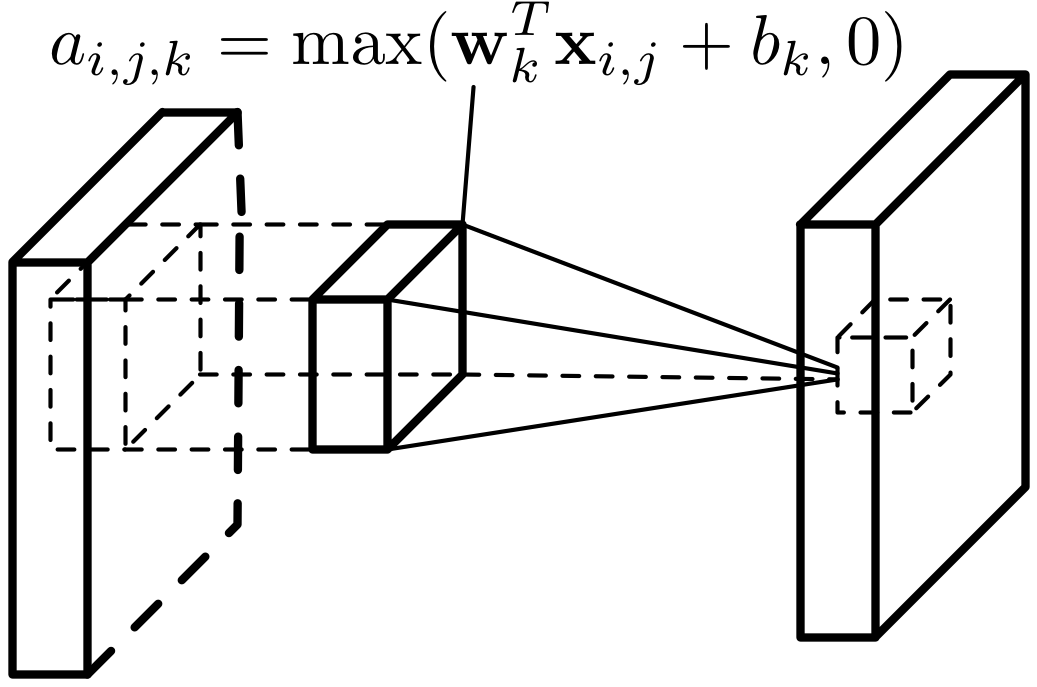


图是一个，图是一个，图是，可见感受野大小为，为放大倍数，为卷积核大小。

扩张卷积广泛应用于场景分隔、机器翻译、语音合成和语音识别。

#### Network in Network

是一种综合网络结构，其将线性的卷积层替换成一个小型网络，本文介绍的层即是其中之一。其有能力进一步接近哪些潜在概念的抽象表示。NIN的整体结构是小型网络的栈式堆叠，如下图所示。



卷积层搭配非线性激活函数(如)的特征映射可以表示为

其中是第个特征映射在位置的值，是在位置的输入块，分别是第个滤波器的权重向量和偏置。

而层表示为

其中，表示层的层数，等于，在层中，的卷积被放置在传统的卷积层之后，相当于跨通道的参数池化操作，这个在中被成功运用。因此层可以被认为是在普通卷积层后的一个级联的跨通道参数池化操作。在最后，设置一个全局均值池化层，用于在最后一层将各特征映射进行空间上的均值，然后直接将输出交给层，相比全连接层，全局均值池化拥有更少的参数并且可以降低过拟合风险和计算量。

#### Inception Module

使用可变的滤波器来捕捉大小不同的可视样式，并且尽可能形成最简洁的结构。包含一个池化操作和三个卷积操作，而在非卷积前放置卷积可以在不增加计算复杂度的情况下增加网络的深度，可以大量减少参数,相比于，可以将参数降低至

为了获取高性能的网络，把计算消耗控制在适中的水平，建议计算中，的大小从输入到输出应该缓慢地下降，且低维度的可以在表示能力没有太大损失的情况下在空间上聚合。最优化的网络性能应该是在每层的滤波器数量和网络的深度中找到平衡。

从中获取灵感，他们最新的组合了体系和 ，他们发现可以显著加速网络的训练，在数据集中，通过组合三个和一个达到了的错误率

## Pooling Layer

池化层是CNN的重要组件，其一方面可以通过减少卷积层之间的链接，从而降低计算成本，另一方面还可以将卷积层的特征映射进一步优化。池化层只有若干个超参数，其没有需要训练的参数。

最基础的池化层有两种，一种是均值池化，一种是最大值池化，前者在区块中计算均值得到输出，后者计算最大值。以下介绍一些近期在使用的池化技术。

#### LP Pooling

由受生物学启发，经理论研究，其比max pooling具备更加的泛化效果。其可以表示为

其中是第各特征映射在对应的池化区域的激活值。当时，就是在对求和，作用类似于均值池化，当，利用极限计算可以发现结果为中的最大值。其作用为最大值池化。

#### Mixed Pooling

由随即丢弃的思想所启发，其结合了最大值池化和均值池化，表示为

其中每次会随机取以此决定是使用最值池化还是均值池化，在反向传播中也会使用相同的进行计算。Mixed Pooling在修正过拟合问题中比最大值池化和均值池化表现得都更好。

#### Stochastic Pooling

同样由随即丢弃思想所启发，不同于直接选择最大值，会根据的多项分布来选择输出值，这样保证了非最大值也有机会被利用。

首先，先计算每个单元的概率

其概率分布

根据其概率分布选择一个位置，使

相比于最大值池化，可以有效抑制过拟合。

#### Spectral Pooling

是通过在频域截去输入的一些表示来降低维度的池化层，可以将的输入降低为的输出。首先其先计算输入特征映射的DFT，然后只截取中间的子矩阵，再通过iDFT还原原图像。相比于，的低通滤波能在相同的维度下保留更多的信息，且此方法也不会像其他方法一样遭遇维数锐减的情况，并且其运算可以用矩阵运算来实现，计算成本也很低。

#### Spatial Pyramid Pooling

可以生成一个定长的表示，与输入的大小无关。输入的特征映射会划分成大小和输入大小成比例的空间容器中，所以划分的容器数是固定的，池化出来的结果也是固定的大小。这与滑动窗口不同，滑动窗口的方法产生的块数与输入大小呈正相关。将最后一层池化层换成，构成的SPP网络可以处理不同大小的图片

#### Multi-scale Orderless Pooling

使用多尺度的弱结构化的Pooling层，在不丢失CNN识别能力的基础上提高不变性。它使用若干尺度从整个图片和局部区块提取深度激活特征，全局特征和传统CNN一致，而区块的用VLAD编码进行聚合，用以捕获更多局部的良好粒度的图片细节，并且提高了不变性。新的图片表示由全局激活值和VLAD的局部特征激活值联系在一起获得。

## Activation Function

适当的激活函数能显著提升CNN性能，一下有一些近期广泛使用的激活函数。

#### ReLU

修正的线性单元（Rectified linear unit），这是一个不饱和的激活函数，其函数定义为

其是一个分段线性函数，分为和，计算速度比要快很多，并且其可以引入稀疏的隐藏单元，从而允许网络更加轻松地产生稀疏表示。在缺少预先训练的时候，也可以高效地训练。尽管在0处地间断可能影响反向传播的性能，许多实际的经验说明ReLU还是表现得更好。

ReLU在正值部分是线性的，导数恒为1，可以规避梯度消失带来的训练困难，但是由于负值部分恒为0，所以一旦有一个很大的负梯度流过，很可能未来的输出都处于负值区域，难以矫正，神经元即死亡，不再有能力更新权值。

#### Leaky ReLU

ReLU的一个劣势是无论一个单元是否激活，都可能存在0梯度，这会导致在初始化未激活的情况下，使用梯度优化算法，不会改变这些单元的权重。并且由于固定在零梯度，训练的速度将变得缓慢。为了缓解这个问题，引入Leaky ReLU，定义如下

其中是之间的一个随机数，相比于ReLU直接把负值映射为0，Leaky ReLU将负值予以压缩，这使得即是单元未激活也可以有一个小的非零的梯度。

#### Parametric ReLU

相比使用一个预先固定的值，Parametric ReLU使每个通道使用不同的可以被学习一提高准确率的，其计算可以被定义为

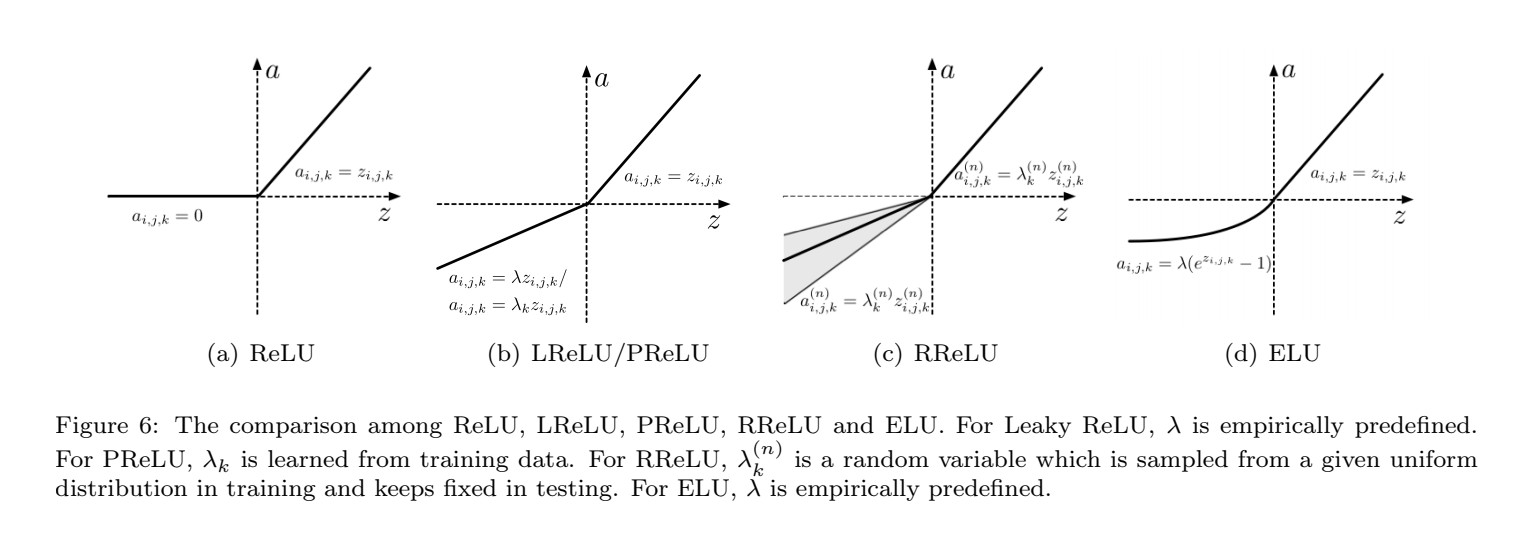
其中是第个通道中可学习的参数，这样对于整个网络来说只引入了少量额外参数（数量和通道数一致），基本没有过拟合的风险和额外的计算消耗（通过一些数据对比，PReLU还是有一定过拟合倾向的），且也能和其他参数一起同时在反向传播中被训练。

#### Randomized ReLU

在RReLU中，负值部分在训练中每计算一组样本都使用均匀分布随机一个采样比例，同样每一个通道使用的参数相互独立。而在测试集中，这个参数是一个固定值。

其中表示的是第个测试样本，表示的是此时对第个通道使用的随机参数。在训练中，指定随机值的上下界，然后每次从中选出随即参数，在测试时不再使用随机值，而是选择上下界的均值作为一个固定参数。

因为其随机化的训练过程，可以避免过拟合。经评估，让负值区域获得一个非0的洒落式的贡献，能提高性能。



#### ELU

ELU可以加速神经网络，还可以提高分类的准确率。像ReLU及变种一样，将正值部分设置为相同的值以规避梯度消失的问题，但和ReLU相反的是，ELU存在负值部分并且可以加速学习。

和LReLU等变种相比，虽然它们都含有一个不饱和的负值部分，但是ELU给负值部分应用了一个饱和函数，这降低了未激活单元的变化幅度，使得ELU对噪音有更强的鲁棒性。ELU计算可以被定义为

其中是一个预定义参数，用来控制负值输入的饱和值。

#### Maxout

Maxout是一个备选的非线性函数，对于空间中的每一个位置，其将会取各通道中的最大值作为输出。可以定义为

其中表示第个通道。值得注意的是，maxout拥有ReLU所有的优点，实际上ReLU是maxout的一个特殊情况，比如并使得为0向量为0。除此之外，maxout非常适合和Dropout配合使用。

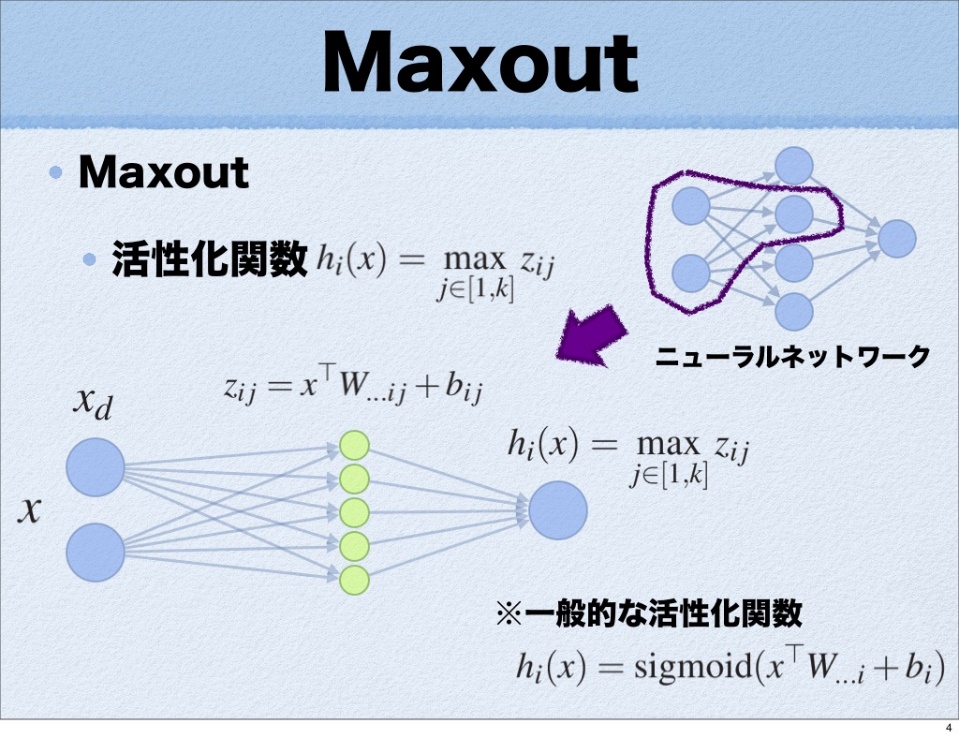
优点：

* 拟合能力强，可以拟合任意凸函数
* 具有ReLU的所有优点
* 没有ReLU的神经元死亡的缺点

缺点：

* Maxout相当于增加了k个神经元作为中介，然后在其中求出最大值，因此参数个数是原先的k倍。

以MLP为例展示Maxout的计算过程。



#### Probout

将maxout转化成基于概率的变种，称为probout。把原先求最大值的操作修改成一个概率采样的过程。首先，为每一个线性单元分配概率值，概率的计算为

其中是一个预定义参数，用来控制概率分布的变化，然后根据概率分布随机选择一个单元的值作为输出。很容易发现，如果，那么其等价于maxout。

为了整合dropout，我们将概率分布定义为

激活函数表示为

其中服从分布

Probout可以平衡maxout获得的令人满意的特性，同时通过让每个通道都产生贡献以提高不变性。然而在实践中，probout因为额外增加的概率计算，其计算成本要高于maxout。