一文看尽深度学习中的各种池化方法！

背景

卷积神经网络(Convolution Neural Network, CNN)因其强大的特征提取能力而被广泛地应用到计算机视觉的各个领域，其中卷积层和池化层是组成CNN的两个主要部件。理论上来说，网络可以在不对原始输入图像执行降采样的操作，通过堆叠多个的卷积层来构建深度神经网络，如此一来便可以在保留更多空间细节信息的同时提取到更具有判别力的抽象特征。然而，考虑到计算机的算力瓶颈，通常都会引入池化层，来进一步地降低网络整体的计算代价，这是引入池化层最根本的目的。

作用

池化层大大降低了网络模型参数和计算成本，也在一定程度上降低了网络过拟合的风险。概括来说，池化层主要有以下五点作用：

增大网络感受野
抑制噪声，降低信息冗余
降低模型计算量，降低网络优化难度，防止网络过拟合
使模型对输入图像中的特征位置变化更加鲁棒

对于池化操作，大部分人第一想到的可能就是Max_Pooling和Average_Pooling，但实际上卷积神经网络的池化方法还有很多，本文将对业界目前所出现的一些池化方法进行归纳总结：

池化大盘点

1. Max Pooling(最大池化)

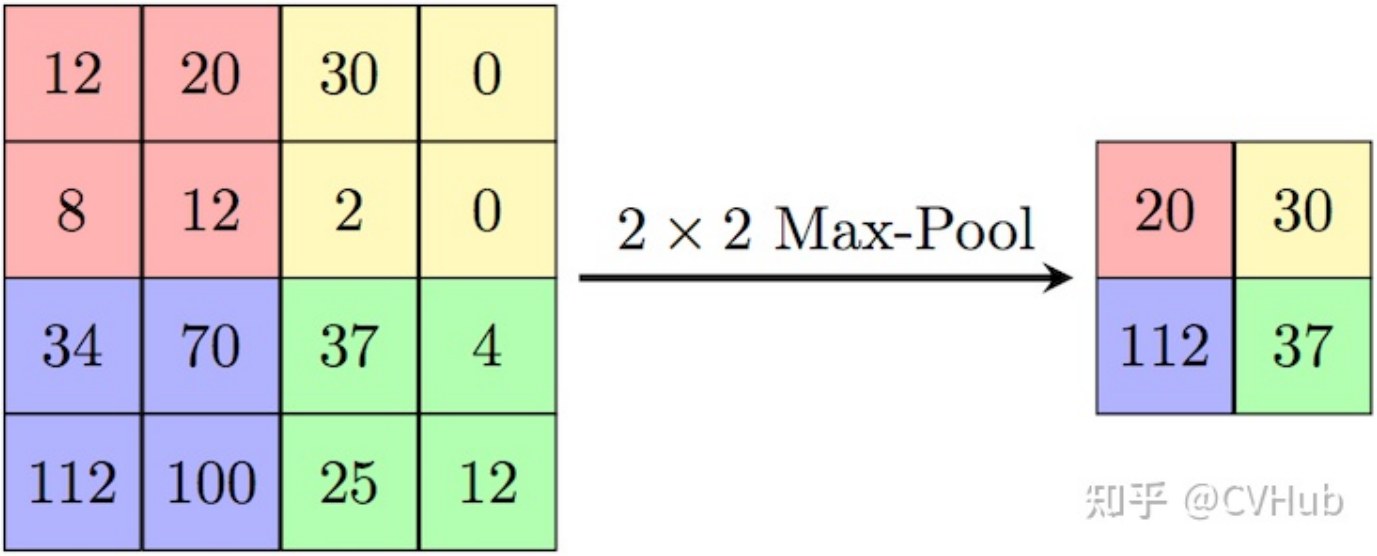
定义

最大池化(Max Pooling)是将输入的图像划分为若干个矩形区域，对每个子区域输出最大值。其定义如下：

$$y_{kij} = \max_{(p,q)\in\mathcal{R}_{ij}} x_{kpq}$$

其中，y_{kij} 表示与第 k 个特征图有关的在矩形区域 $\mathcal{R}_{ij}$ 的最大池化输出值，x_{kpq} 表示矩形区域 $\mathcal{R}_{ij}$ 中位于(p,q)处的元素。

如下图1所示，表示的就是对一个4 × 4的特征图邻域内的值，用一个2 × 2的filter，步长为2进行“扫描”，选择最大值输出到下一层，这叫做最大池化。



解说

对于最大池化操作，只选择每个矩形区域中的最大值进入下一层，而其他元素将不会进入下一层。所以最大池化提取特征图中响应最强烈的部分进入下一层，这种方式摒弃了网络中大量的冗余信息，使得网络更容易被优化。同时这种操作方式也常常丢失了一些特征图中的细节信息，所以最大池化更多保留些图像的纹理信息。

2. Average Pooling(平均池化)

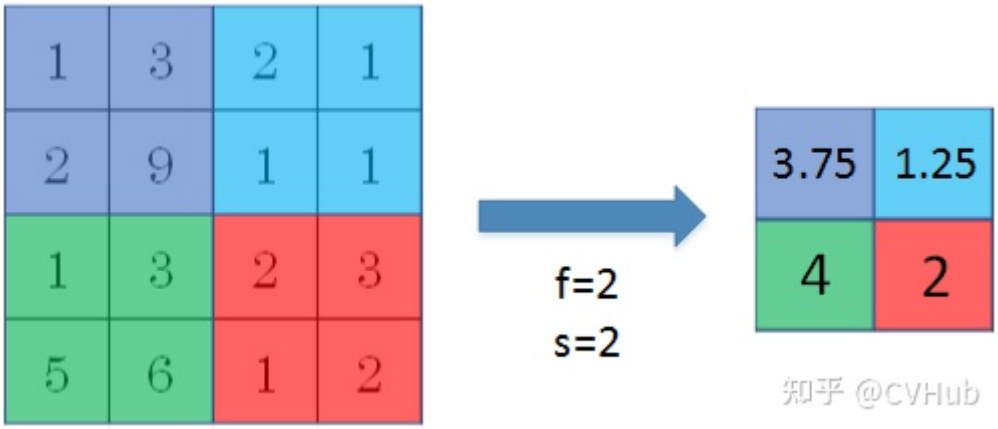
定义

平均池化(Average Pooling)是将输入的图像划分为若干个矩形区域，对每个子区域输出所有元素的平均值。其定义如下：

$$y_{kij} = \frac{1}{|\mathcal{R}_{ij}|} \sum_{(p,q)\in\mathcal{R}_{ij}} x_{kpq}$$

其中，y_{kij} 表示与第 k 个特征图有关的在矩形区域 $\mathcal{R}_{ij}$ 的平均池化输出值，x_{kpq} 表示矩形区域 $\mathcal{R}_{ij}$ 中位于(p,q)处的元素，$|\mathcal{R}_{ij}|$ 表示矩形区域 $\mathcal{R}_{ij}$ 中元素个数。

如下图2所示，表示的就是对一个4 × 4的特征图邻域内的值，用一个2 × 2的filter，步长为2进行“扫描”，对区域内元素取平均，将平均值输出到下一层，这叫做平均池化。



解说

平均池化取每个矩形区域中的平均值，可以提取特征图中所有特征的信息进入下一层，而不像最大池化只保留值最大的特征，所以平均池化可以更多保留些图像的背景信息。

3.Global Average Pooling(全局平均池化)

背景

在卷积神经网络训练初期，卷积层通过池化层后一般要接多个全连接层进行降维，最后再Softmax分类，这种做法使得全连接层参数很多，降低了网络训练速度，且容易出现过拟合的情况。在这种背景下，M Lin等人提出使用全局平均池化**Global Average Pooling[1]**来取代最后的全连接层。用很小的计算代价实现了降维，更重要的是GAP极大减少了网络参数(CNN网络中全连接层占据了很大的参数)。

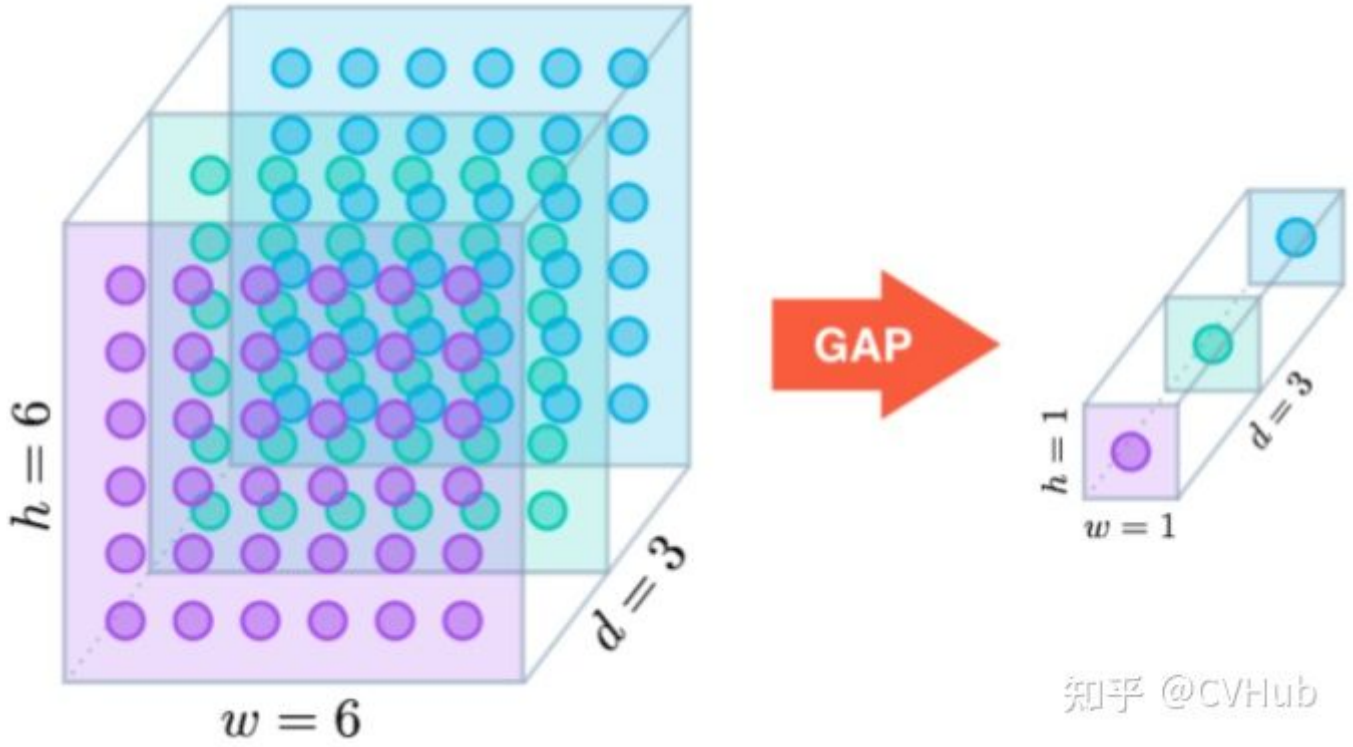
定义

全局平均池化是一种特殊的平均池化，只不过它不划分若干矩形区域，而是将整个特征图中所有的元素取平均输出到下一层。其定义如下：

$$y_k = \frac{1}{|\mathcal{R}|} \sum_{(p,q)\in\mathcal{R}} x_{kpq}$$

其中，y_k 表示与第 k 个特征图的全局平均池化输出值，x_{kpq} 表示第 k 个特征图区域 $\mathcal{R}$ 中位于(p,q)处的元素，$|\mathcal{R}|$ 表示第 k 个特征图全部元素的个数。

如下图3所示，对于一个输入特征图 $\mathbf{X} \in \mathbb{R}^{h\times w\times d}$,经过全局平均池化(GAP)之后生成新的特征图 $\mathbf{O} \in \mathbb{R}^{1\times 1\times d}$ 。



解说

作为全连接层的替代操作，GAP对整个网络在结构上做正则化防止过拟合，直接剔除了全连接层中黑箱的特征，直接赋予了每个channel实际的类别意义。除此之外，使用GAP代替全连接层，可以实现任意图像大小的输入，而GAP对整个特征图求平均值，也可以用来提取全局上下文信息，全局信息作为指导进一步增强网络性能。

4. Mix Pooling(混合池化)

定义

为了提高训练较大CNN模型的正则化性能，受Dropout(将一半激活函数随机设置为0)的启发，Dingjun Yu等人提出了一种随机池化**Mix Pooling[2]**的方法，随机池化用随机过程代替了常规的确定性池化操作，在模型训练期间随机采用了最大池化和平均池化方法，并在一定程度上有助于防止网络过拟合现象。其定义如下：

$$y_{kij} = \lambda \cdot \max_{(p,q)\in\mathcal{R}_{ij}} x_{kpq} + (1-\lambda) \cdot \frac{1}{|\mathcal{R}_{ij}|} \sum_{(p,q)\in\mathcal{R}_{ij}} x_{kpq}$$

其中， λ是0或1的随机值，表示选择使用最大池化或平均池化，换句话说，混合池化以随机方式改变了池调节的规则，这将在一定程度上解决最大池和平均池所遇到的问题。

解说

混合池化优于传统的最大池化和平均池化方法，并可以解决过拟合问题来提高分类精度。此外该方法所需要的计算开销可忽略不计，而无需任何超参数进行调整，可被广泛运用于CNN。

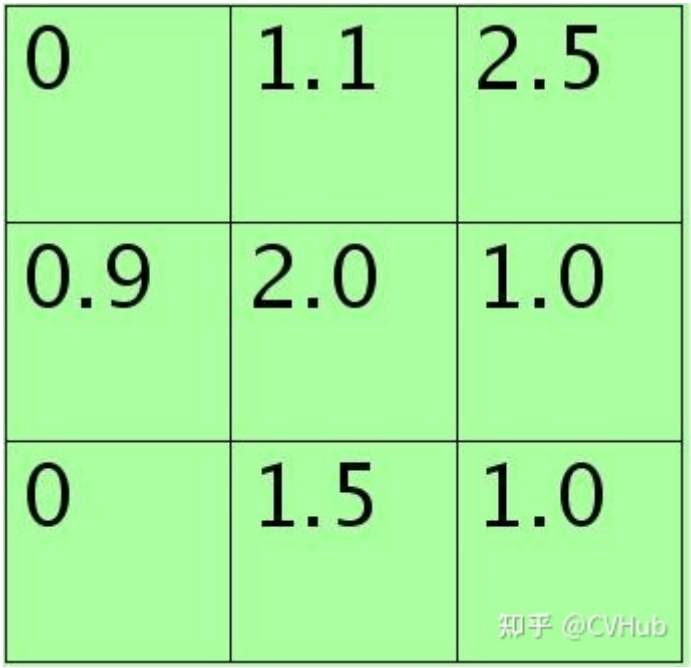
5. Stochastic Pooling(随机池化)

定义

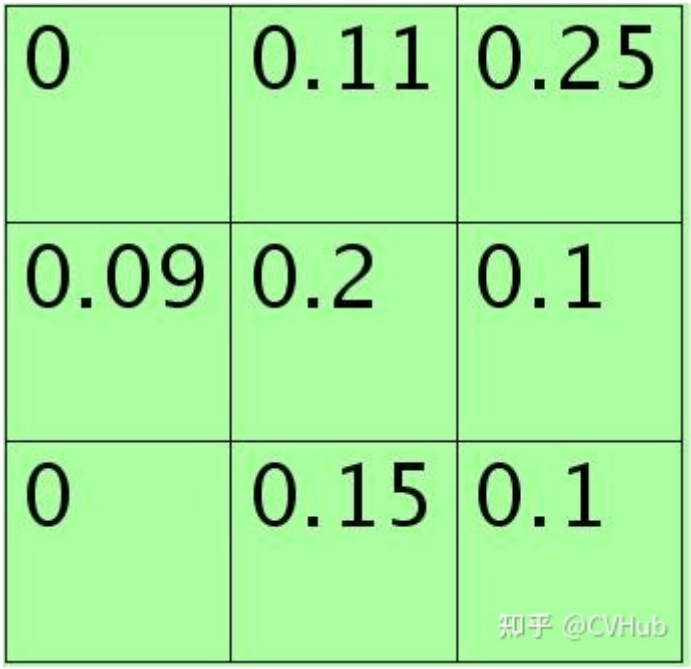
随机池化**Stochastic Pooling[3]**是Zeiler等人于ICLR2013提出的一种池化操作。随机池化的计算过程如下：

先将方格中的元素同时除以它们的和sum，得到概率矩阵。
按照概率随机选中方格。
pooling得到的值就是方格位置的值。

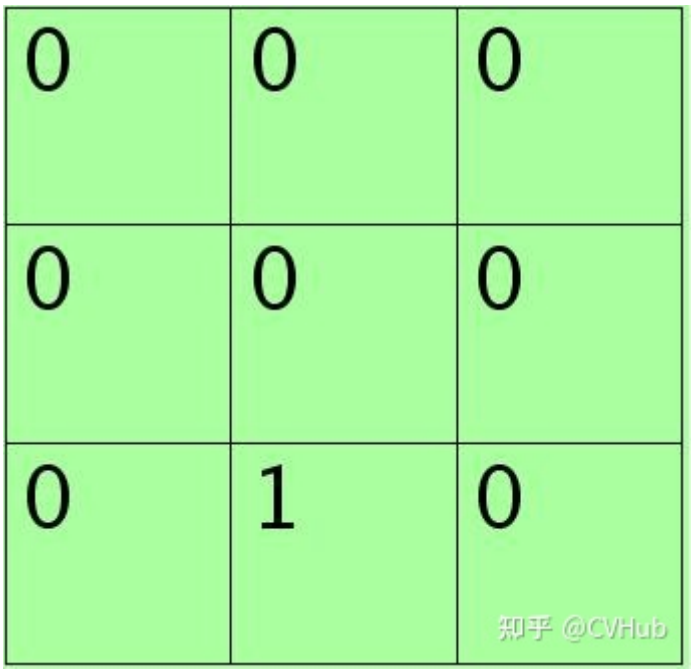
假设特征图中Pooling区域元素值如下(参考Stochastic Pooling简单理解)：



3 × 3大小的，元素值和sum=0+1.1+2.5+0.9+2.0+1.0+0+1.5+1.0=10。方格中的元素同时除以sum后得到的矩阵元素为：



每个元素值表示对应位置处值的概率，现在只需要按照该概率来随机选一个，方法是：将其看作是9个变量的多项式分布，然后对该多项式分布采样即可，theano中有直接的multinomial()来函数完成。当然也可以自己用0-1均匀分布来采样，将单位长度1按照那9个概率值分成9个区间（概率越大，覆盖的区域越长，每个区间对应一个位置），然随机生成一个数后看它落在哪个区间。 比如如果随机采样后的矩阵为：



则这时候的poolng值为1.5。使用stochastic pooling时(即test过程)，其推理过程也很简单，对矩阵区域求加权平均即可。比如对上面的例子求值过程为为:

$$0*0+1.1*0.11+2.5*0.25+0.9*0.09+2.0*0.2+1.0*0.1+0*0+1.5*0.15+1.0*0.1=1.625$$

说明此时对小矩形pooling后的结果为1.625。 在反向传播求导时，只需保留前向传播已经记录被选中节点的位置的值，其它值都为0,这和max-pooling的反向传播非常类似。 本小节参考

Stochastic Pooling简单理解[4]。

解说

随机池化只需对特征图中的元素按照其概率值大小随机选择，即元素值大的被选中的概率也大，而不像max-pooling那样，永远只取那个最大值元素，这使得随机池化具有更强的泛化能力。

6. Power Average Pooling(幂平均池化)

定义

幂平均池化**Power Average Pooling[5]**基于平均池化和最大池化的结合，它利用一个学习参数 p 来确定这两种方法的相对重要性；当 $p=1$ 时，使用局部求和，当 $p \to \infty$ 时，使用最大池化，其定义如下：

$$\tilde{\mathbf{a}} = \sqrt[p]{\sum_{i \in \mathbf{R}} \mathbf{a}_i^p}$$

其中 $\mathbf{R}$ 表示待池化区域中的像素值集。

7. Detail-Preserving Pooling(DPP池化)

为了降低隐藏层的规模或数量，大多数CNN都会采用池化方式来减少参数数量，来改善某些失真的不变性并增加感受野的大小。由于池化本质上是一个有损的过程，所以每个这样的层都必须保留对网络可判别性最重要的部分进行激活。但普通的池化操作只是在特征图区域内进行简单的平均或最大池化来进行下采样过程，这对网络的精度有比较大的影响。基于以上几点，Faraz Saeedan等人提出一种自适应的池化方法-DPP池化**Detail-Preserving Pooling[6]**，该池化可以放大空间变化并保留重要的图像结构细节，且其内部的参数可通过反向传播加以学习。DPP池化主要受**Detail-Preserving Image Downscaling[7]**的启发。

Detail-Preserving Image Downscaling

$$O[p] = \frac{1}{k_p} \sum_{q \in \Omega_p} I[q] \cdot \| I[q] - \tilde{I}[p] \|^{\lambda}$$

其中 I 是原图， O 是output，[ ]表示取对于坐标像素值。

$$\tilde{I} = I_D * \frac{1}{16}[1 \quad 2 \quad 1]^T [1 \quad 2 \quad 1]$$

其中 ID 是施加到输入随后的下采样，其随后由一个近似的二维高斯滤波器平滑化的箱式滤波器的结果。 如下图7展示了DPID的结构图， ID 是用近似高斯分布的filter smooth后的图像：

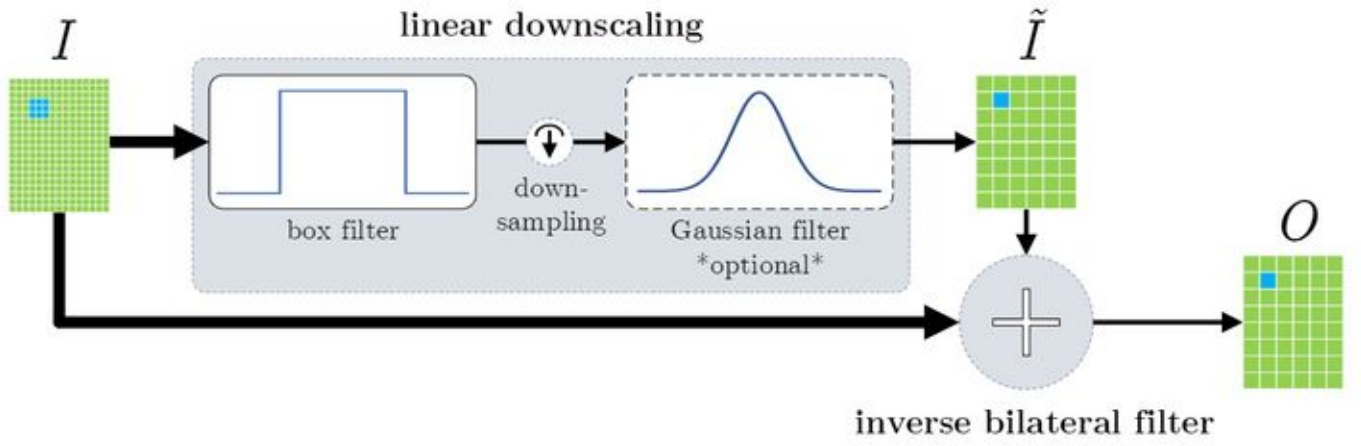


Figure 2. Diagram of detail-preserving downscaling (DPID) [31] and our detail-preserving pooling (DPP). DPP omits the Gaussian filter; Full-DPP replaces the box filter with a learned 2D filter.

知乎 @CVHub

如下图8展示了DPID的滤波图，与普通双边滤波器不同，它奖励输入强度的差异，使得与 I 的差异较大的像素值贡献更大。

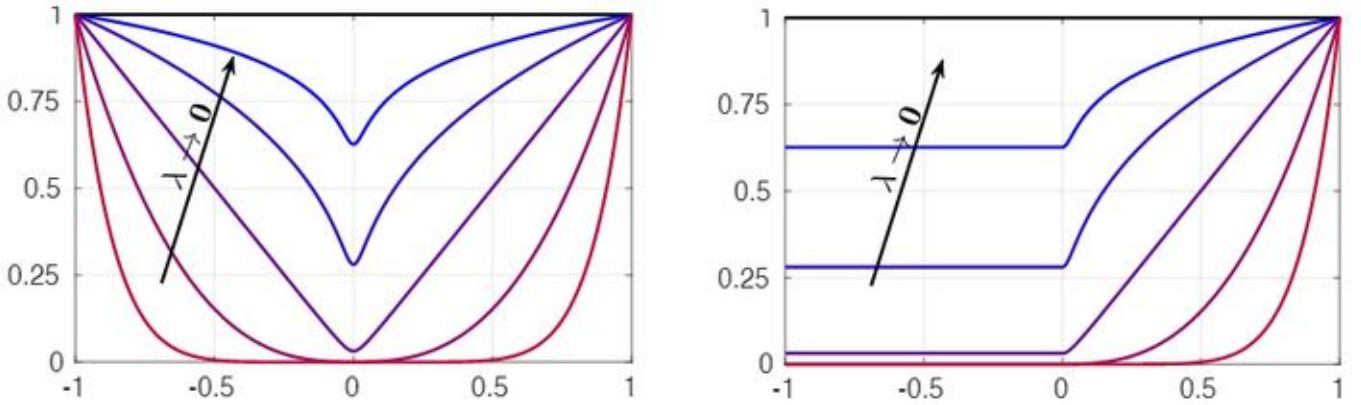


Figure 3. Inverse bilateral functions $\rho_\lambda(\cdot)$ used to calculate the weights of the input nodes in the neighborhood Ω_p: The symmetric function (*left*) rewards arguments with bigger absolute values, the asymmetric function (*right*) rewards arguments with bigger values if they are positive. As smaller parameters λ decrease the dynamic range of the reward function, it tends to being uniform for $\lambda \to 0$, leading to a simple averaging of all neighbors.

知乎 @CVHub

Detail-Preserving Pooling

a. 将上部分中的 $L2Norm$ 替换成一个可学习的generic scalar reward function：

$$D_{\alpha,\lambda}(I)[p] = \frac{1}{\sum_{q' \in \Omega_p} \omega_{\alpha,\lambda}[p,q']} \sum_{q \in \Omega_p} \omega_{\alpha,\lambda}[p,q] I[q]$$

b. 首先给出weight的表示：

$$\omega_{\alpha,\lambda}[p,q] = \alpha + \rho_\lambda(I[q] - \tilde{I}[p])$$

c. 这里给出了两种reward function：

$$\rho_{sym}(x) = \left(\sqrt{x^2 + \varepsilon^2}\right)^\lambda$$
$$\rho_{Asym}(x) = \left(\sqrt{\max(0,x)^2 + \varepsilon^2}\right)^\lambda$$

d. 作者又补充了的生成：

$$\tilde{I}_F[p] = \sum_{q \in \tilde{\Omega}_p} F[q] I[q]$$

解说

DPP池化允许缩减规模以专注于重要的结构细节，可学习的参数控制着细节的保存量，此外，由于细节保存和规范化相互补充，DPP可以与随机合并方法结合使用，以进一步提高准确率。

8. Local Importance Pooling(局部重要性池化)

背景

CNN通常使用空间下采样层来缩小特征图，以实现更大的接受场和更少的内存消耗，但对于某些任务而言，这些层可能由于不合适的池化策略而丢失一些重要细节，最终损失模型精度。为此，作者从局部重要性的角度提出了局部重要性池化**Local Importance Pooling[8]**，通过基于输入学习自适应重要性权重，LIP可以在下采样过程中自动增加特征判别功能。

定义

池化操作可归纳为如下公式：

$$O_{x',y'} = \frac{\sum_{(\Delta x,\Delta y)\in\Omega} F(I)_{x+\Delta x,y+\Delta y} I_{x+\Delta x,y+\Delta y}}{\sum_{(\Delta x,\Delta y)\in\Omega} F(I)_{x+\Delta x,y+\Delta y}}$$

其中 F 的大小和特征 I 一致，代表每个点的重要性。Local Aggregation and Normalization框架如下图所示：

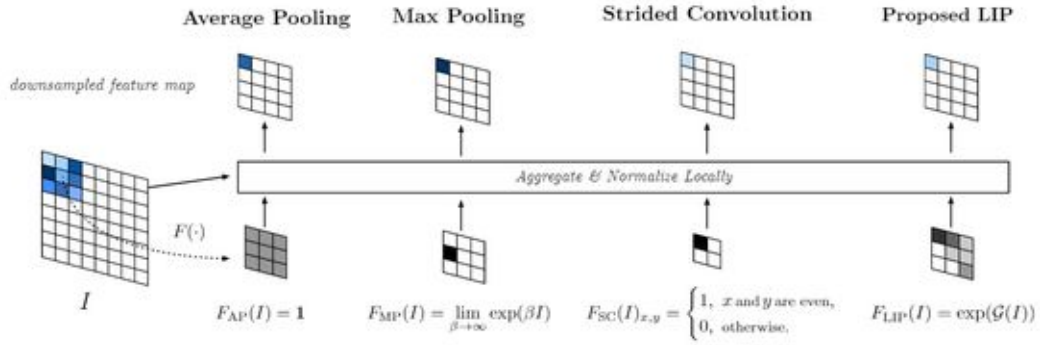
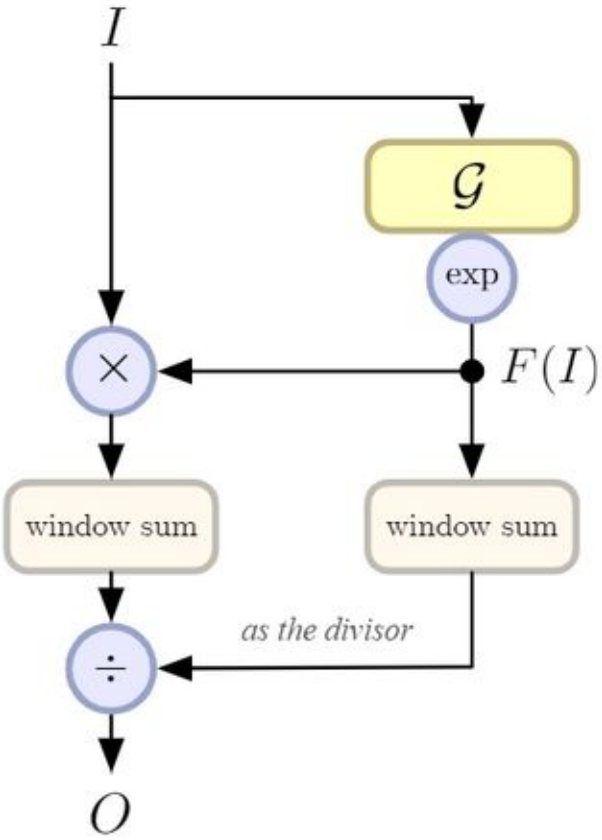


Figure 2: Different downsampling methods viewed in the LAN framework. Activations in the input feature map are blue colored, darker meaning larger. Only activations and corresponding importance within current sliding window are shown. For strided convolution, the window size is equivalent to the stride, which is 2 here.

图中分别对应了平均池化，最大池化和步长为2的卷积。首先最大池化对应的最大值不一定是最具区分力的特征，并且在梯度更新中也难以更新到最具区分力的特征，除非最大值被抑制掉。而步长为2的卷积问题主要在于固定的采样位置。

因此，合适的池化操作应该包含两点：
a. 下采样的位置要尽可能非固定间隔
b. 重要性的函数 F 需通过学习获得

Local Importance-based Pooling




```
import torch
import torch.nn.functional as F

def lip2d(x, logit,
          kernel_size=3,
          stride=2,
          padding=1):
    weight = torch.exp(logit)
    return F.avg_pool2d(x*weight
          , kernel_size, stride,
          padding)/F.avg_pool2d(
          weight, kernel_size,
          stride, padding)
```

(a) (b)

Figure 3: LIP operator and its PyTorch implementation. The logit module $\mathcal{G}$ is not shown in (b). The 'window sum's in (a) mean locally summing within sliding windows.

知乎 @CVHub

局部重要性池化(LIP)结构如上图所示，LIP首先在原特征图上学习一个类似于注意力的特征图，然后再和原特征图进行加权求均值，公式可表述如下：

$$O_{x',y'} = \frac{\sum_{(\Delta x,\Delta y)\in\Omega} I_{x+\Delta x,y+\Delta y}\exp(\mathcal{G}(I))_{x+\Delta x,y+\Delta y}}{\sum_{(\Delta x,\Delta y)\in\Omega}\exp(\mathcal{G}(I))_{x+\Delta x,y+\Delta y}}$$

对于 G 函数，可以通过如下图d和e两种方式实现(分别称之为Projection和Bottleneck-X)。而对应的ResNet-LIP则如下图b所示：

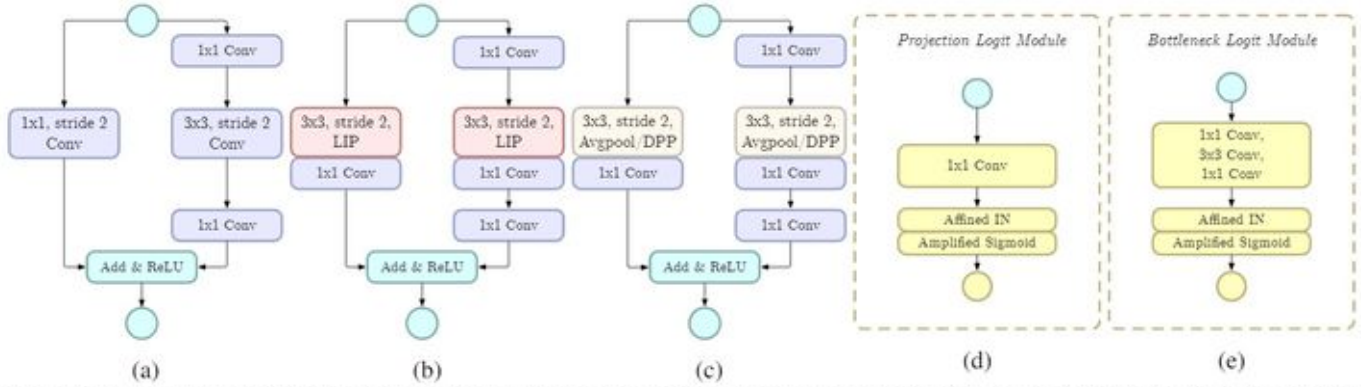


Figure 4: Structures of ResNet building blocks for downsampling and logit modules. There are ResNet building blocks with strided convolutions (a), LIP (b), average pooling or DPP (c). (d) and (e) show the projection and bottleneck logit module. The first two 'Conv's in (e) mean convolutions and following affine Instance Normalization and ReLU function.

解说

Local Importance Pooling可以学习自适应和可判别性的特征图以汇总下采样特征，同时丢弃无信息特征。这种池化机制能极大保留物体大部分细节，对于一些细节信息异常丰富的任务至关重要。

9. Soft Pooling(软池化)

背景

现有的一些池化方法大都基于最大池化和平均池化的不同组合，而软池化**Soft Pooling[9]**是基于softmax加权的方法来保留输入的基本属性，同时放大更大强度的特征激活。与maxpooling不同，softpool是可微的，所以网络在反向传播过程中为每个输入获得一个梯度，这有利于提高训练效果。

定义

SoftPool的计算流程如下：
a. 特征图透过滑动视窗来框选局部数值
b. 框选的局部数值会先经过指数计算，计算出的值为对应的特征数值的权重
c. 将各自的特征数值与其相对应的权重相乘
d. 最后进行加总

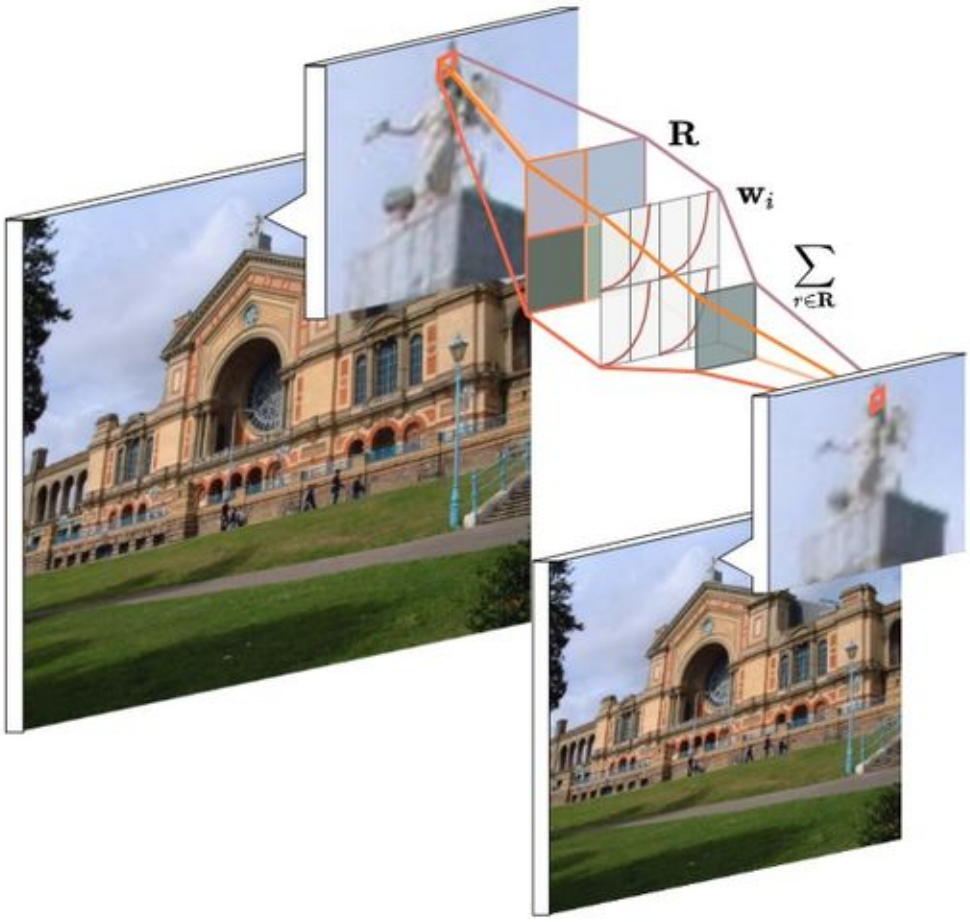


Figure 1: **SoftPool illustration**. The original image is sub-sampled with a 2×2 ($k = 2$) kernel. The output is based on the exponentially weighted sum of the original pixels within the kernel region. This can improve the representation of high-contrast regions, present around object edges or specific feature activations.

知乎 @CVHub

这样的方式让整体的局部数值都有所贡献，重要的特征占有较高的权重。比Max pooling(直接选择最大值)、Average pooling (求平均，降低整个局部的特征强度) 能够保留更多讯息。

SoftPool的数学定义如下：

计算特征数值的权重，其中R为框选的局部区域，a为特征数值

$$\mathbf{w}_i = \frac{e^{\mathbf{a}_i}}{\sum_{j \in \mathbf{R}} e^{\mathbf{a}_j}}$$

将相应的特征数值与权重相乘后做加总操作

$$\tilde{\mathbf{a}} = \sum_{i \in \mathbf{R}} \mathbf{w}_i * \mathbf{a}_i$$

梯度计算: 下图可以很清楚的指导使用SoftPool的Gradient计算流程。与Max Pooling不同，SoftPool是可微的，因此在反向传播至少会分配一个最小梯度值进行更新。

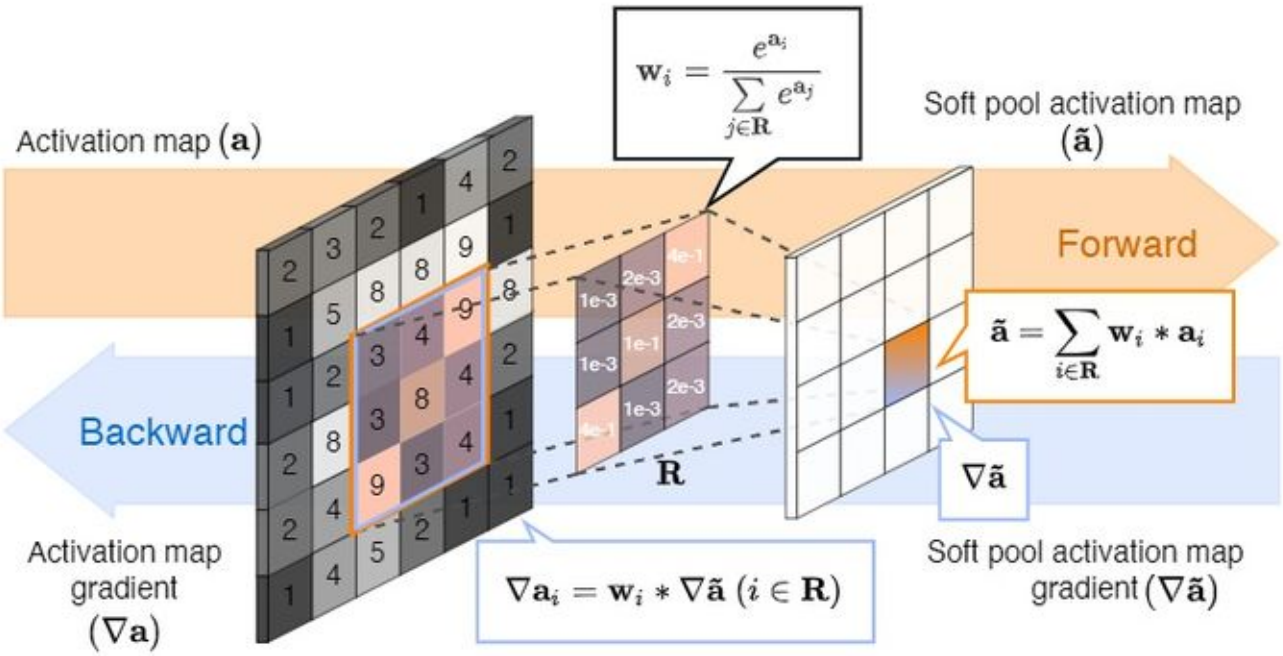


Figure 4: **SoftPool calculation**. In forward operation, in orange, the kernel uses the exponential softmax value of each activation as weight and calculates the weighted sum for region $\mathbf{R}$. These weights are also used for the gradients $(\nabla\tilde{\mathbf{a}}_i)$, in blue. Activation gradients are proportional to the calculated softmax weights.

知乎 @CVHub

解说

作为一种新颖地池化方法，SoftPool可以在保持池化层功能的同时尽可能减少池化过程中带来的信息损失，更好地保留信息特征并因此改善CNN中的分类性能。大量的实验结果表明该算法的性能优于原始的Avg池化与Max池化。随着神经网络的设计变得越来越困难，而通过NAS等方法也几乎不能大幅度提升算法的性能，为了打破这个瓶颈，从基础的网络层优化入手，不失为一种可靠有效的精度提升手段。