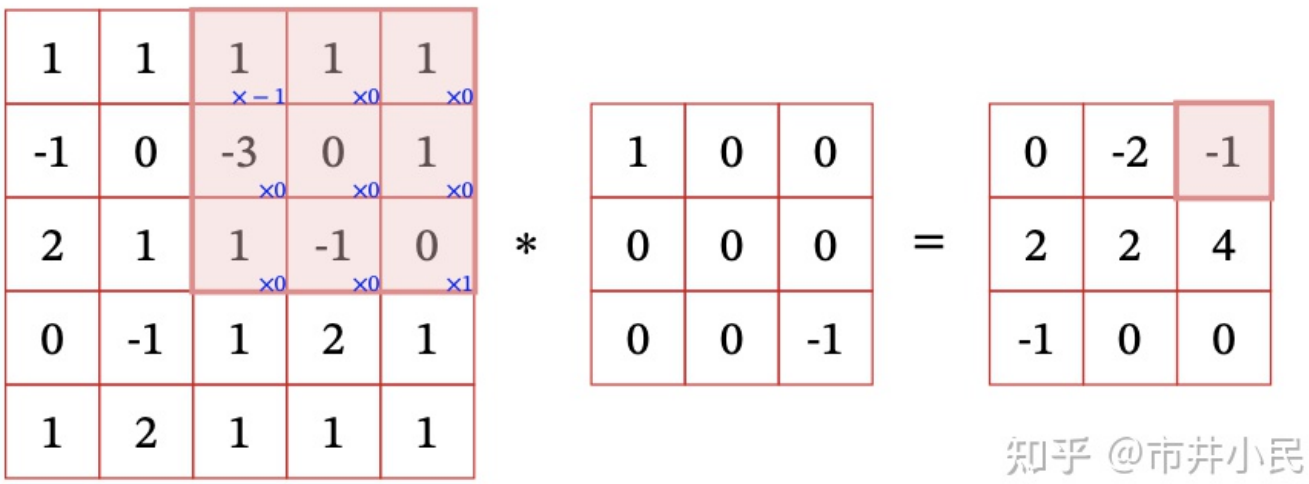
机器学习与深度学习面试系列十三（CNN）

简单介绍一下什么是CNN?

卷积神经网络(Convolutional Neural Networks，CNN)是一种前馈神经网络，其特点是每层的神经元节点只响应前一层局部区域范围内的神经元（全连接网络中每个神经元节点响应前一层的全部节点）。一个深度卷积神经网络模型通常由若干卷积层叠加若干全连接层组成，中间也包含各种非线性操作以及池化操作。卷积运算主要用于处理类网格结构的数据，因此对于时 间序列以及图像数据的分析与识别具有显著优势。

什么是卷积和互相关?

这里不对数学中的卷积运算进行分析，直接来看在图像处理中卷积运算的定义。给定一个图像 $\boldsymbol{X} \in \boldsymbol{R}^{\boldsymbol{M}*\boldsymbol{N}}$ ，一个卷积核（滤波器） $\boldsymbol{W} \in \boldsymbol{R}^{\boldsymbol{U}*\boldsymbol{V}}$ ，一般 $\boldsymbol{U} << \boldsymbol{X}, \boldsymbol{V} << \boldsymbol{Y}$ ，其卷积运算定义为 $\boldsymbol{Y} = \boldsymbol{W} * \boldsymbol{X}$ 。



可以看出，图像被卷积区域的左上角对应卷积核的右下角，而互相关运算，就是将卷积核翻转180度(也可以理解为图像进行翻转)，然后进行"卷积"运算。 $\boldsymbol{Y} = \boldsymbol{W} \otimes \boldsymbol{X} = \boldsymbol{rot}180(\boldsymbol{W}) * \boldsymbol{X}$ 。互相关和卷积的区别仅仅在于卷积核是否进行翻转，因此互相关也可以称为不翻转卷积。在神经网络中使用卷积是为了进行特征抽取，卷积核是否进行翻转和其特征抽取的能力无关。特别是当卷积核是可学习的参数时，卷积和互相关在能力上是等价的。因此，为了实现上(或描述上)的方便起见，我们用互相关来代替卷积。事实上，很多深度学习工具中卷积操作其实都是互相关操作，后面的描述中我们不再区分互运算和卷积运算，只要知道，我们在CNN说的卷积其实都是互运算即可。

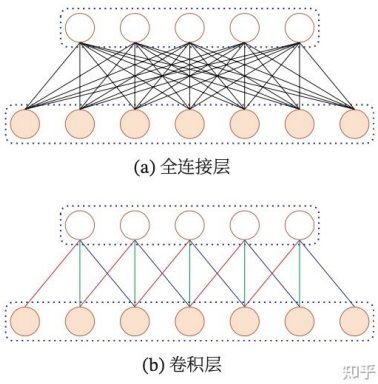
卷积层有哪些特点?

卷积神经网络最早主要是用来处理图像信息，在用全连接前馈网络来处理图像时，会存在以下两个问题：

1. **参数太多**。如果输入图像大小为 100 × 100 × 3(即图像高度为100，宽度为100以及RGB 3个颜色通道)，在全连接前馈网络中，第一个隐藏层的每个神经元到输入层都有 100 × 100 × 3 = 30000个互相独立的连接，每个连接都对应一个权重参数。随着隐藏层神经元数量的增多，参数的规模也会急剧增加. 这会导致整个神经网络的训练效率非常低，也很容易出现过拟合。
2. **很难提取局部不变性特征**。自然图像中的物体都具有局部不变性特征，比如尺度缩放、平移、旋转等操作不影响其语义信息，一般需要进行数据增强来提高性能。

采用卷积神经网络，解决如上两个问题：

1. **局部连接**。在卷积层(假设是第 层)中的每一个神经元都只和前一层(第 −1 层)中某个局部窗口内的神经元相连，构成一个局部连接网络。卷积层和前一层之间的连接数由原来的 $M_l * M_{l-1}$ 个连接变为 $M_l * K$ 个连接， 为滤波器大小。
2. **权重共享**。作为参数的滤波器 $W^{(l)}$ 对于第 层的所有的神经元都是相同的。权重共享可以理解为一个滤波器只捕捉输入数据中的一种特定的局部特征。因此，如果要提取多种特征就需要使用多个不同的滤波器。



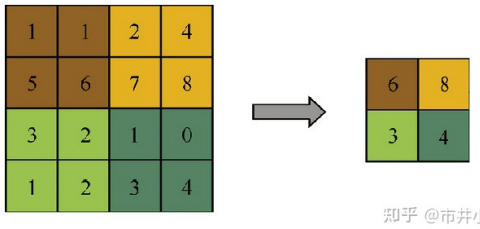
卷积层和全连接层对比

池化层是什么？有什么作用？

常用的池化操作主要针对非重叠区域，又叫子采样（降采样），包括均值池化(mean pooling)、最大池化(max pooling)等。池化操作除了能显著降低参数量外，还能够保持对平移、伸缩、旋转操作的不变性。平移不变性是指输出结果对输入的小量平移基本保持不变。

均值池化通过对邻域内特征数值求平均来实现，能够抑制由于邻域大小受限造成估计值方差增大的现象，特点是对背景的保留效果更好。

最大池化则通过取邻域内特征的最大值来实现，能够抑制网络参数误差造成估计均值偏移的现象，特点是更好地提取纹理信息。



卷积、池化和矩阵运算的关系?

深度学习大部分的操作其实都是矩阵运算，假设一个图像是 $X=\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{pmatrix}$ ，卷积核为

$w=\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ ，卷积步长为2，则：

$$Y=W*X=\begin{pmatrix} ax+by+cz \\ cx+dy+ez \\ ex+fy+gz \end{pmatrix}=\begin{pmatrix} x & y & z & 0 & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 & 0 \\ 0 & 0 & 0 & 0 & x & y & z \end{pmatrix}\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{pmatrix}=AX$$ 。

同样，池化可以看成一种特殊的卷积操作，如平均池化，可以看成卷积核所有参数都为 $\frac{1}{K}$ 的卷积运算。

卷积层、池化层的输出尺寸、参数量计算?

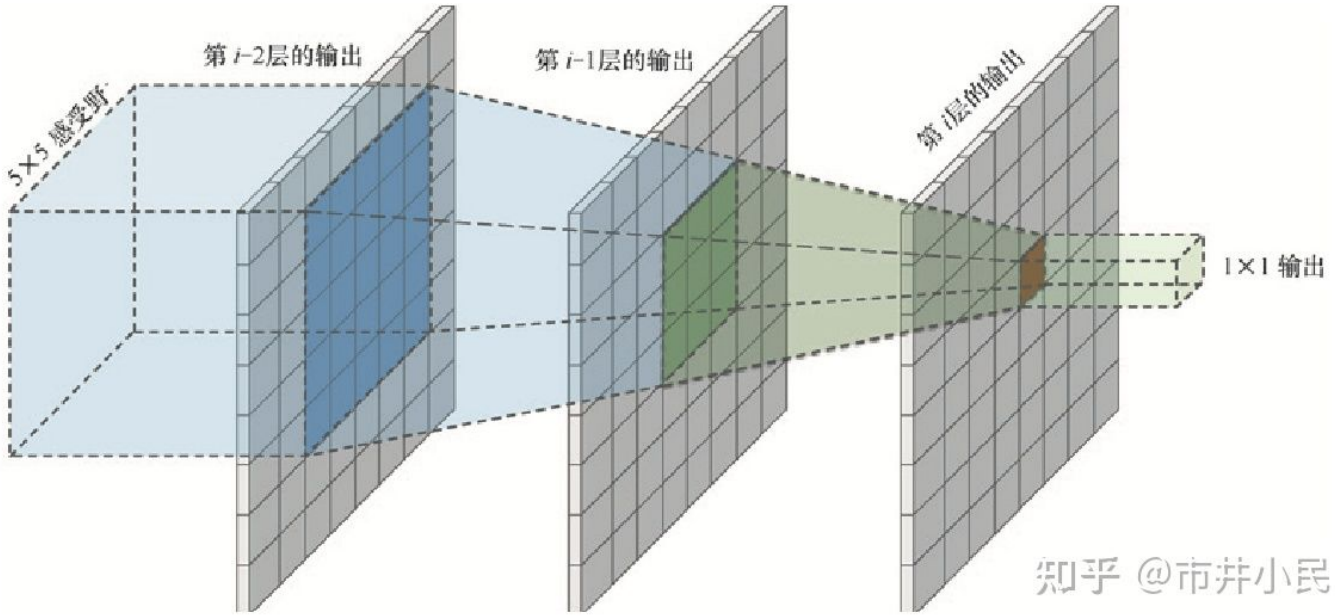
假设在卷积核的滑动过程中，我们对输入特征图($l_w * l_h$)的左右两侧分别进行了 P_w 列填充，上下两侧分别进行了 P_h 行填充，填充后的特征图尺寸为 $(l_w+2P_w)(l_h+2P_h)$ ，卷积核大小为K，卷积步长为S，则输出特征图的尺寸为： $l_e=\lfloor\frac{l_e+2P_e-K}{S}\rfloor+1, \quad e\in\{w,h\}$ 。

卷积层的参数量，主要取决于每个卷积核的参数量以及卷积核的个数。在这里，每个卷积核含有 $c^{(i)}k_wk_h$ 个参数，而卷积核的个数即输出特征图的通道个数 $c^{(0)}$ ，因此参数总量为 $c^{(0)}c^{(i)}k_wk_h$ 。

池化可以看作一种特殊的卷积，$l_e = \lceil \frac{l_e + 2P_e - K}{S} \rceil + 1, \quad e \in \{w, h\}$ ，通常步长就是卷积核的大小（不重叠），如果不考虑填充，则 $l_e = \lceil \frac{l_e - K}{K} \rceil + 1 = \lceil \frac{l_e}{K} \rceil, \quad e \in \{w, h\}$ 。池化层没有需要学习的参数。

什么是感受野？感受野的大小怎么计算？

感受野的定义是，对于某层输出特征图上的某个点，在卷积神经网络的原始输入数据上能影响到这个点的取值的区域。



计算感受野的大小，我们不妨从后向前计算。第i层是由第i-1层卷积运得来的，根据上面卷积输出大小计算公式，我们知道第i层是由第i-1层卷积得到，

$l_e^{(i)} = \lfloor \frac{l_e^{(i-1)} - K^{(i-1)}}{S^{(i-1)}} \rfloor + 1, \quad e \in \{w, h\}$ ，所以：

$l_e^{(i-1)} = (l_e^{(i)} + 1) S^{(i-1)} + K^{(i-1)}, \quad e \in \{w, h\}$ 。

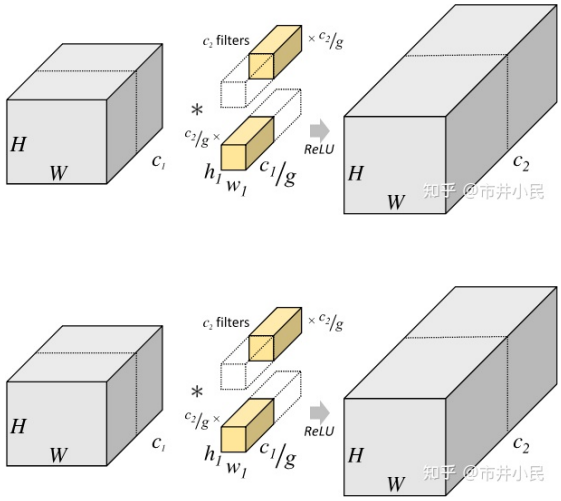
最后一层第i层认为宽和高都是1，依次反向逐层计算，得到最后在输入图像上的感受野大小。

卷积有哪些变种？

分组卷积(Group Convolution)、转置卷积 (Transposed Convolution)、空洞卷积(Dilated/Atrous Convolution)、可变形卷积(Deformable Convolution)。

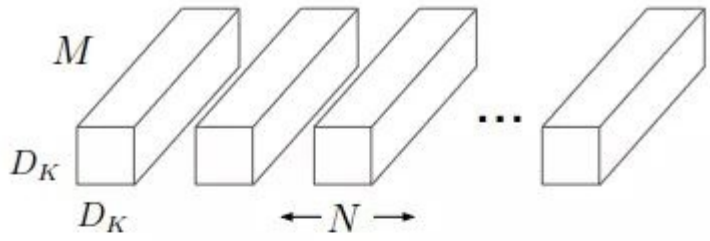
简述分组卷积及其应用场景？

在普通的卷积操作中，一个卷积核对应输出特征图的一个通道，而每个卷积核又会作用在输入特征图的所有通道上(即卷积核的通道数等于输入特征图的通道数)。所谓分组卷积，其实就是将输入通道和输出通道都划分为同样的组数，然后仅让处于相同组号的输入通道和输出通道相互进行“全连接”。

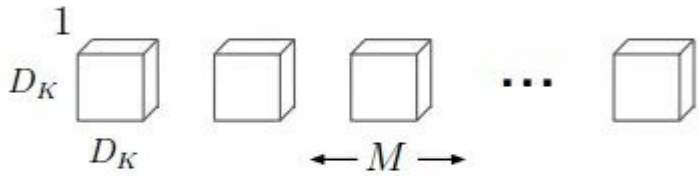


第一张图代表标准卷积操作。若输入特征图尺寸为 $H \times W \times c_1$ ，卷积核尺寸为 $h_1 \times w_1 \times c_1$ ，输出特征图尺寸为 $H \times W \times c_2$ ，标准卷积层的参数量为： $(h_1 \times w_1 \times c_1) \times c_2$ 。第二张图代表分组卷积操作。将输入特征图按照通道数分成 g 组，则每组输入特征图的尺寸为 $H \times W \times (\frac{c_1}{g})$ ，对应的卷积核尺寸为 $h_1 \times w_1 \times (\frac{c_1}{g})$ ，每组输出特征图尺寸为 $H \times W \times (\frac{c_2}{g})$ 。将 g 组结果拼接(concat)，得到最终尺寸为 $H \times W \times c_2$ 的输出特征图。分组卷积层的参数量为 $h_1 \times w_1 \times (\frac{c_1}{g}) \times (\frac{c_2}{g}) \times g = h_1 \times w_1 \times c_1 \times c_2 \times \frac{1}{g}$ 。

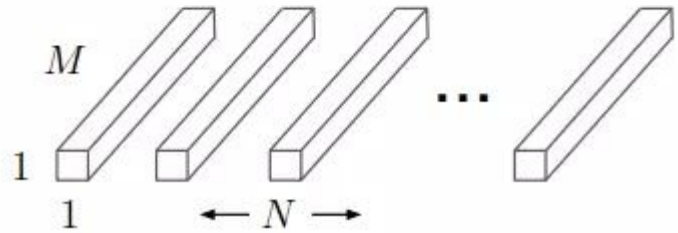
目前，分组卷积更多地被用来构建用于移动设备的小型网络模型。例如，深度可分离卷积(Depthwise Separable Convolution)。深度可分离卷积是MobileNet的精髓，它由deep_wise卷积（深度卷积）和point_wise卷积（逐点卷积）两部分组成。深度卷积其实就是 $g = M = N$ 的分组卷积，kernel_size = K*K*1，第一次卷积总的参数量为K*K*1*M。逐点卷积其实就是把 g 组结果用 1×1 的卷积核拼接起来，kernel_size = 1*1*M，第二次卷积总的参数量为1*1*M*N。深度卷积参数量为 $(D_K \times D_K \times 1) \times M$ ，逐点卷积参数量为 $(1 \times 1 \times M) \times N$ ，所以深度可分离卷积参数量是标准卷积的 $\frac{D_K \times D_K \times M + M \times N}{D_K \times D_K \times M \times N} = \frac{1}{N} + \frac{1}{D_K^2}$ 。



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

简述转置卷积的主要思想以及应用场景?

上文中提到，卷积操作可以看作是一种特殊的矩阵乘法， $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$ ，反过来，记 $\boldsymbol{A}^T$ 为矩阵 $\boldsymbol{A}$ 的转置，定义如下矩阵运算: $\hat{\boldsymbol{Y}} = \boldsymbol{A}^T\hat{\boldsymbol{X}}$ ，其所对应的操作被称为转置卷积， $\hat{\boldsymbol{X}}$ 和 $\hat{\boldsymbol{Y}}$ 分别是转置卷积的输入和输出。其中 $\hat{\boldsymbol{Y}}$ 和 $\boldsymbol{X}$ 的尺寸大小一样， $\hat{\boldsymbol{X}}$ 和 $\boldsymbol{Y}$ 的尺寸大小一样。所以转置卷积也被称为反卷积，不过值得注意的一点是，反卷积的操作得到的 $\hat{\boldsymbol{Y}}$ 只是恢复了矩阵 $\boldsymbol{X}$ 的尺寸大小，并不能恢复 $\boldsymbol{X}$ 的每个元素值，所以它不叫“逆卷积”。

需要注意的是：这里的转置卷积矩阵的参数，不一定从原始的卷积矩阵中简单转置得到的，转置这个操作只是提供了转置卷积矩阵的形状而已。这个转置卷积矩阵的参数是可以学习的，因此我们不需要一些人为预先定义的方法。即使它被称为转置卷积，它并不是意味着我们将一些现存的卷积矩阵简单转置并且使用其转置后的值。

转置卷积主要用于对特征图进行扩张或上采样，代表性的应用场景：

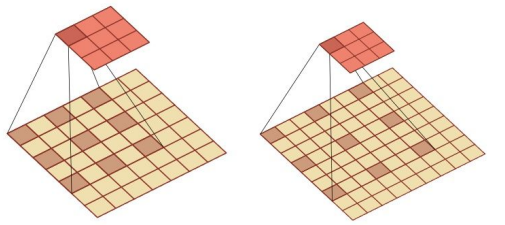
- 语义分割 / 实例分割等任务:由于需要提取输入图像的高层语义信息，网络的特征图尺寸一般会先缩小，进行聚合。此外，这类任务一般需要输出与原始图像大小一致的像素级分割结果，因而需要扩张前面得到的具有较高语义信息的特征图，这就用到了转置卷积。
- 一些物体检测、关键点检测任务，需要输出与源图像大小一致的 热图。

• 图像的自编码器、变分自编码器、生成式对抗网络等。

简述空洞卷积(Atrous Convolution)的设计思路?

对于一个卷积层，如果希望增加输出单元的感受野，一般可以通过三种方式实现：(1)增加卷积核的大小;(2)增加层数，比如两层 3 × 3 的卷积可以近似一层5×5卷积的效果;(3)在卷积之前进行汇聚操作。前两种方式会增加参数数量， 而第三种方式会丢失一些信息。

空洞卷积是一种不增加参数数量，同时增加输出单元感受野的一种方法，也称为膨胀卷积。空洞卷积通过给卷积核插入“空洞”来变相地增加其大小。如果在卷积核的每两个元素之间插入 – 1 个空洞，卷积核的有效大小为 $K' = K + (K-1)(D-1)$ 。其中 称为膨胀率(Dilation Rate). 当 $D = 1$ 时卷积核为普通的卷积核。

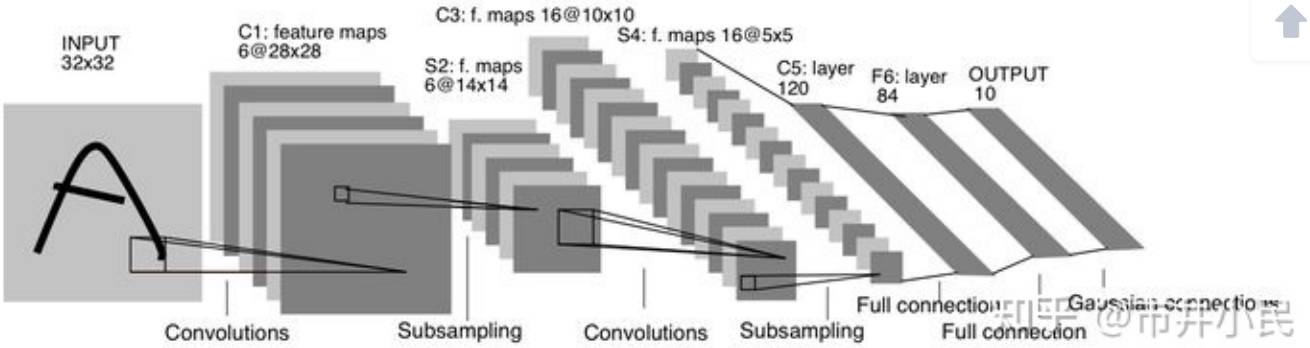


(a) 膨胀率 $D = 2$　　(b) 膨胀率 $D = 3$

简述卷积神经网络近年来在结构设计上的主要发展和变迁(从 AlexNet到ResNet系列)?

LeNet。 1998年提出，它是现代卷积神经网络架构的奠基者，被应用于手写数字识别，取得了很大的成功。以现在的眼光来看，LeNet 绝对是一个小网络，也没什么特点。但是，LeNet 是 CNN 网络结构的开山鼻祖，第一次定义了 CNN 网络结构。

1. 定义了卷积神经网络的基本框架：卷积层 + 池化层（Pooling Layer） + 全连接层
2. 定义了卷积层（Convolution Layer），与全连接层相比，卷积层的不同之处有两点：局部连接（引进“感受野”这一概念）、权值共享（减少参数数量）
3. 利用池化层进行下采样（Downsampooling），从而减少计算量
4. 用 Tanh 作为非线性激活函数（现在看到的都是改进的 LeNet 了，用 ReLU 代替了 Tanh）



AlexNet。以其文章一作姓氏命名的一种神经网络架构，在2012年李飞飞创办的ImageNet图像识别大赛中获得了较先前最高水平大幅提高的成绩，为学者所瞩目。AlexNet的出现也标志着卷积神经网络在图像处理领域的复兴。

1. 采用双 GPU 网络结构，从而可以设计出更“大”、更“深”的网络（相较于当时的算力来说）
2. 采用 ReLU 代替 Tanh，稍微解决梯度消失问题（Gradient Vanishing Problem），加快网络收敛速度。
3. 提出局部相应归一化（LRN, Local Response Normalization），后被认为无必要。
4. 重叠池化方法。令 Pooling 操作中的 stride 小于池化核的大小，从而使相邻的池化区域存在重叠部分，这一操作称为 Overlapping Pooling。据作者所言，这一操作能减少指标 Top-1/Top-5 Error Rate 0.4%/0.3%，并且减少过拟合现象。
5. 利用 Dropout 避免网络过拟合。

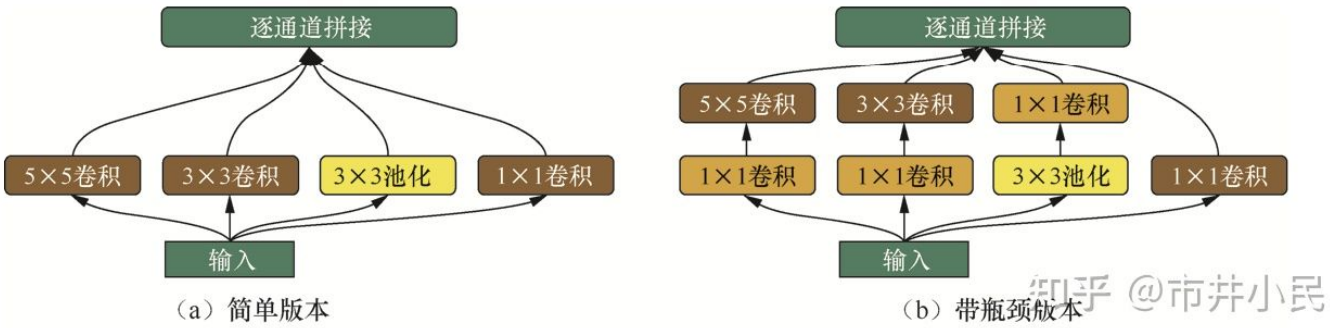
VGGNet。VGGNet之所以闻名于世，是因为其赢得了2014年的ImageNet大赛。VGGNet将网络的深度扩展到了19层，并且在每个卷积层使用了3x3这种小尺寸的卷积核。文章的实验证明，缩小卷积核尺寸，增多层数可以大大提升卷积神经网络的性能。相比于 AlexNet，VGGNet 做了如下改变：

1. 用多个 $3*3$ 小卷积核代替之前的 $5*5, 7*7$ 等大卷积核，这样可 以在更少的参数量、更小的计算量下，获得同样的感受野以及更大的网络深度。
2. 用 $2*2$ 池化核代替之前的3*3池化核。
3. 去掉了局部响应归一化模块。

整体来说，VGGNet 网络结构设计更加简洁，整个网络采用同一种卷积核尺寸($3*3$)和池化核尺寸($2*2$)，并重复堆叠了很多基础模块，最终的网络深度也达到了近 20 层。

Inception-V1 (GoogLeNet)。2014 年，由Google提出，在同年的 ImageNet Challenge 的分类与检测（Detection）任务上夺得第一名。GoogLeNet认为提高模型表达能力的最直接的办法就是增加模型的“大小”，即加宽加深模型。而这又会导致两个问题的产生：模型越大，其网络参数也就越大，就越容易产生过拟合现象，所以就需要更大的数据集，然而大型数据集的构建成本是很高昂的；模型越大，对于计算资源的需求就越大，这在现实任务中是难以接受的。而作者认为解决这两个问题的基本方法是将全连接层，甚至是卷积层改为稀疏的网络结构。（作者还在文中指出，

GoogLeNet 的参数仅有 AlexNet 的 1/12，而 AlexNet 的全连接层的参数量甚至占到了自身参数量的 90% 以上）。Inception系列网络



1. 提出了 Inception模块，它将之前网络中的大通道卷积层替换为由多个小通道卷积层组成的多分支结构，Inception 模块会同时使用 1×1、3×3、5×5 的3种卷积核进行多路特征提取，这样能使网络稀疏化的同时，增强网络对多尺度特征的适应性。如上图(a)所示。
2. 提出了瓶颈(bottleneck)结构，即在计算比较大的卷积层之前， 先使用卷积对其通道进行压缩以减少计算量(在较大卷积层完成计算之后，根据需要有时候会再次使用卷积将其通道数 复原)，如上图(b)所示。
3. 从网络中间层拉出多条支线，连接辅助分类器，用于计算损失并进行误差反向传播，以缓解梯度消失问题。
4. 修改了之前 VGGNet 等网络在网络末端加入多个全连接层进行分类的做法，转而将第一个全连接层换成全局平均池化层(Global Average Pooling)。

Inception-V2。2015年2月由谷歌提出，改进了GoogLeNet：

1. 在 GoogLeNet 基础上增加了 Batch Normalization 操作。
2. 将 Inception Module 中卷积核大小为 5 × 5 的卷积层用两个相连的卷积核大小为 3 × 3 的卷积层进行替换。
3. 将网络第一层的卷积层替换为深度乘子为 8 的可分离卷积。

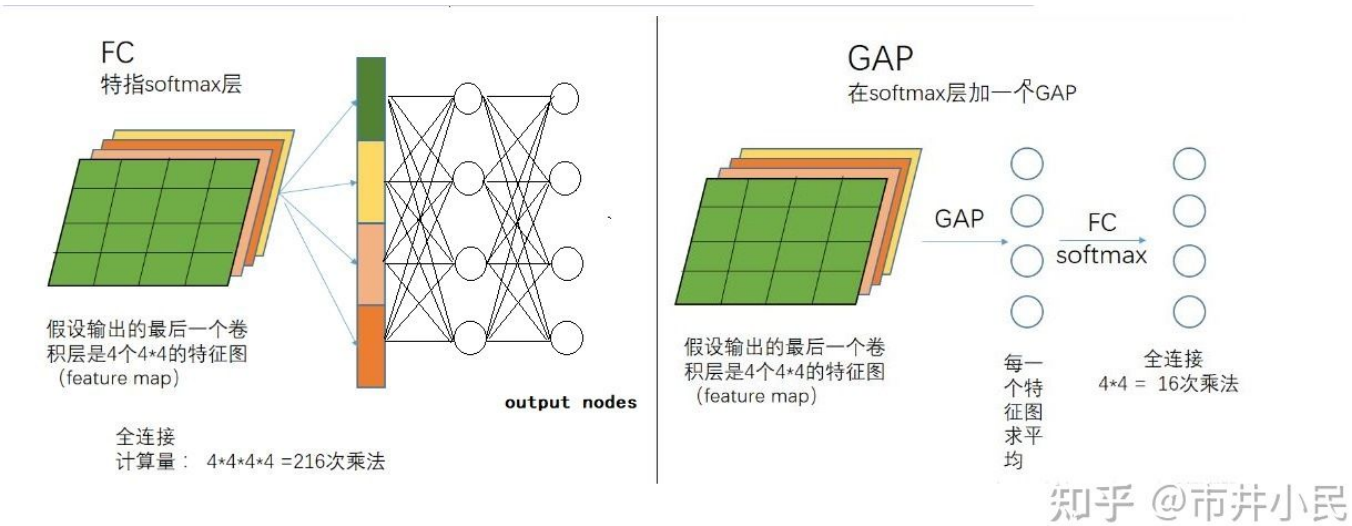
Inception-V3。2015年12月谷歌再次发了篇论文，作者提出了四条设计神经网络的原则：

1. 避免表达瓶颈(representational bottleneck)，尤其是在网络的前几层。具体来说，将整个网络看作由输入到输出的信息流， 我们需要尽量让网络从前到后各个层的信息表征能力逐渐降低，而不能突然剧烈下降或是在中间某些节点出现瓶颈。
2. 特征图通道越多，能表达的解耦信息就越多，从而更容易进行局部处理，最终加速网络的训练过程。
3. 如果要在特征图上做空间域的聚合操作(如卷积)，可以在此之前先对特征图的通道进行压缩，这通常不会导致表达能力的损失。
4. 在限定总计算量的情况下，网络结构在深度和宽度上需要平衡。

ResNet。ResNet的提出源于这样一种现象：随着网络层数的加深，网络的训练误差和测试误差都会上升。这种现象称为网络的退化(degeneration)， 它与过拟合显然是不同的，因为过拟合的标志之一是训练误差降低而 测试误差升高。为解决这个问题，ResNet 采用了跳层连接(shortcut connection)，即在网络中构筑多条“近道”，这有以下两点好处：

1. 能缩短误差反向传播到各层的路径，有效抑制梯度消失的现象， 从而使网络在不断加深时性能不会下降。
2. 由于有“近道”的存在，若网络在层数加深时性能退化，则它可以通过控制网络中“近道”和“非近道”的组合比例来退回到之前浅层时的状态，即“近道”具备自我关闭能力。正如我们在上篇文章中分析的，他可以更好的拟合线性映射，对冲因非线性激活函数带来的信息损失。

为什么现在流行用全局平均池化(GAP)代替全连接网络?



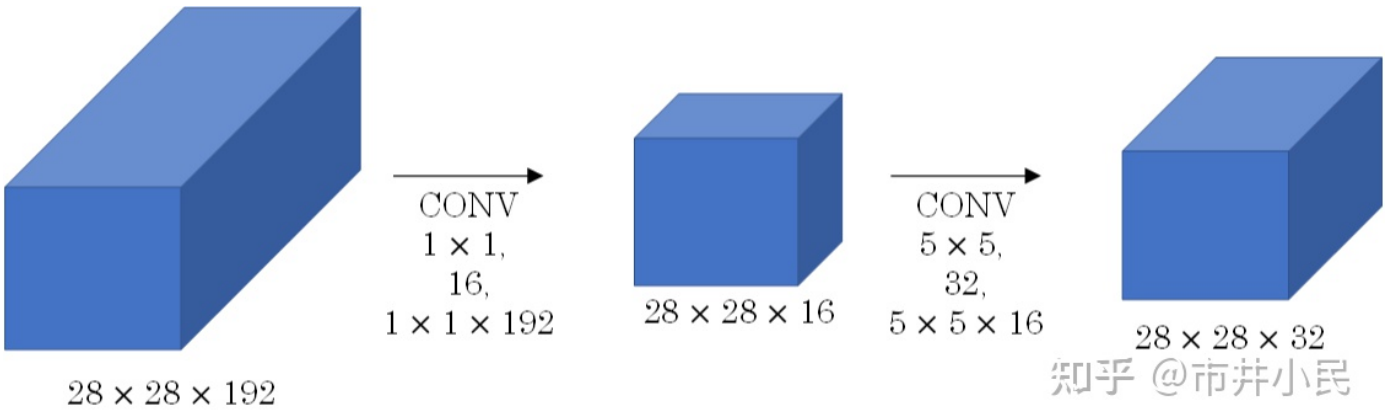
全局平 均池化，它与全连接层有着相似的效果(可以提取全局信息)，并 且具有如下优点：

1. **抑制过拟合**。直接拉平做全连接层的方式依然保留了大量的空间信息，假设feature map是32个通道的10*10图像，那么拉平就得到了32*10*10的向量，如果是最后一层是对应两类标签，那么这一层就需要3200*2的权重矩阵，而GAP不同，将空间上的信息直接用均值代替，32个通道GAP之后得到的向量都是32的向量，那么最后一层只需要32*2的权重矩阵。相比之下GAP网络参数会更少，而全连接更容易在大量保留下来的空间信息上面过拟合。
2. **输入尺寸更加灵活**。在第1点的举例里面可以看到feature map经过GAP后的神经网络参数不再与输入图像尺寸的大小有关，也就是输入图像的长宽可以不固定。
3. **具有较好的可解释性**。比如，我们可以知道特征图上哪些点 对最后的分类贡献最大。

CNN中的瓶颈结构和沙漏结构提出的初衷是什么?可以应用于哪些问题?

瓶颈结构是在 GoogLeNet/Inception-v1 中提出的，而后的ResNet、MobileNet等很多网络也采用并发展了这个结构。瓶颈结构的初衷是为了降低大卷积层的计算量，即在计算比较大的卷积层之

前， 先用一个 1×1 卷积来压缩大卷积层输入特征图的通道数目，以减小计算量，在大卷积层完成计算之后，根据实际需要，有时候会再次使用 一个 1×1 卷积来将大卷积层输出特征图的通道数目复原。



瓶颈结构是卷积神经网络中比较基础的模块，它可以用更小的 计算代价达到与之前相似甚至更好的效果(因为瓶颈结构会增加网 络层数，所以特征提取能力可能也会有相应提升)。瓶颈结构基本上可以用于所有的卷积神经网络中，场景包括物体检测和分割、生成式对抗网络等大方向，以及诸如人脸匹配、再识别、关键点检测等细分领域。

沙漏结构也是卷积神经网络中比较基础的模块，它类似于瓶颈结构，但尺度要更大，涉及的层也更多。沙漏结构一般包括以下两个分支：

1. 自底向上(bottom-up)分支：利用卷积、池化等操作将特征图的尺寸逐层压缩(通道数可能增加)，类似于自编码器中的编码器(encoder)。
2. 自顶向下(top-down)分支：利用反卷积或插值等上采样操作将特征图的尺寸逐层扩大(通道数可能降低)，类似于自编码器中的解码器(decoder)。