第5章 卷积神经网络

卷积神经网络(Convolutional Neural Network,CNN)是一种包含卷积计算的前馈型神经网络,其在大型图像处理方面有出色的表现,目前已经广泛用于图像分类、定位等领域。相比于其他神经网络结构,卷积神经网络需要的参数相对较少,使其能够广泛应用。虽然卷积网络也存在浅层结构,但准确度和表现力等原因很少使用。卷积神经网络学术界和工业界不再进行特意区分,一般指深层结构的卷积神经网络,几层、几十层甚至上百层。

本章首先介绍卷积神经网络的基本概念,包括卷积神经网络的发展历史,以及卷积神经网络的基本构成、组件;其次介绍卷积神经网络和传统前向全连接网络的关系;然后介绍卷积神经网络的可解释性,神经网络的可解释性是力图打开"黑盒"效应的重要方法;最后介绍几种典型的神经网络,对每个典型结构进行详细分析。

5.1 卷积神经网络概述

卷积神经网络是一种处理已知类似格型拓扑数据的神经网络。这种格型可以是一维的,也可以是二维的,例如以时间间隔采样构成的一维时序数据,或二维像素格型构成的图像数据,卷积神经网络在处理这类数据时具有独特的优势。之所以称为卷积网络,是因为该网络采用了卷积算子,而卷积运算是一种特殊的线性运算。从数学角度看,卷积网络就是在网络各层利用卷积运算替代通用的矩阵乘法而构成的神经网络。

卷积神经网络是一种特殊的深层神经网络模型,特殊性体现在两个方面:一方面它的神经元之间是非全连接的;另一方面同一层中某些神经元之间的连接的权重是共享的(相同的)。它的非全连接和权值共享的网络结构使之更类似于生物神经网络,降低了网络模型的复杂度(对于很难学习的深层结构来说这是非常重要的),减少了权值的数量。

5.1.1 卷积神经网络的历史

卷积网络最初是受视觉神经机制的启发而设计,是为识别二维形状而设计的一个多层感知器,这种网络结构对平移、比例缩放、倾斜或者其他形式的变形具有高度不变性。1962 年,大卫·胡贝尔(David Hubel)和托斯滕·维塞尔(Torsten Wiesel)通过对猫视觉皮层细胞的研究,提出了感受野的概念[1]。在此概念基础上,日本学者福岛邦彦(Kunihiko Fukushima)仿造生物的视觉皮层提出神经认知机模型[2]。神经认知机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。神经认知机的 S 层-C 层组合能够进行特征提取和筛选,部分实现了卷积神经网络中卷积层和池化层的功能,因此被认为是卷积神经网络的启发性开创性研究。

1987 年,著名学者亚历山大·韦贝尔(Alexander Waibel)等提出第一个卷积神经网络——时间延迟网络(Time Delay Neural Network,TDNN)[3]。TDNN 是一个应用于

解决语音识别问题的卷积神经网络，使用快速傅里叶变换（Fast Fourier Transform，FFT）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，用来提取频率域上的平移不变特征[4]。在TDNN出现之前，BP算法取得了突破性进展，用于进行人工神经网络的训练，因此TDNN也通过BP算法进行学习。TDNN提出后，在同等条件下其性能超过了当时最主流的模型——隐马尔可夫模型（Hidden Markov Model，HMM）。

1988年，张伟（Wei Zhang）等提出了第一个二维卷积神经网络——平移不变人工神经网络（Shift-Invariant Artificial Neural Network，SIANN），并将其应用于检测医学影像[5]。与此同时，杨立昆（Yann LeCun）等在1989年也同样构建了应用于计算机视觉问题的卷积神经网络，即LeNet的最初版本[6]。LeNet包含2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。杨立昆对权重进行随机初始化后使用了随机梯度下降进行学习，而后期的深度学习研究一直保持该训练策略。此外，杨立昆在论述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。

1993年，杨立昆在贝尔实验室完成代码开发并部署于美国国家收音机公司（National Cash Register Coporation）的支票读取系统。但总体而言，由于学习样本不足、数值计算能力有限等问题，且同一时期以支持向量机（Support Vector Machine，SVM）为代表的核学习方法的兴起，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，应用推广非常受限。

在LeNet的基础上，1998年，杨立昆等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[7]。LeNet-5沿用了杨立昆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，微软公司在2003年使用卷积神经网络开发了光学字符读取（Optical Character Recognition，OCR）系统[8]。后期基于卷积神经网络的应用研究也不断拓展，进入人像识别、手势识别等应用领域。

在2006年深度学习理论被提出后，卷积神经网络的表征学习能力得到了关注，并随着计算设备的更新得到发展。自2012年AlexNet网络开始[9]，得到GPU计算集群支持的复杂卷积神经网络多次成为ImageNet大规模视觉识别竞赛（ImageNet Large Scale Visual Recognition Challenge，ILSVRC）的优胜算法，如2013年的ZFNet[10]、2014年的VGGNet[11]、GoogLeNet[12]和2015年的残差网络等[13]。

5.1.2 卷积神经网络的结构

1. 卷积神经网络的一般结构

卷积神经网络作为深度神经网络的一种，也符合其一般的结构特点，即由输入层、隐含层和输出层构成，如图5-1所示。

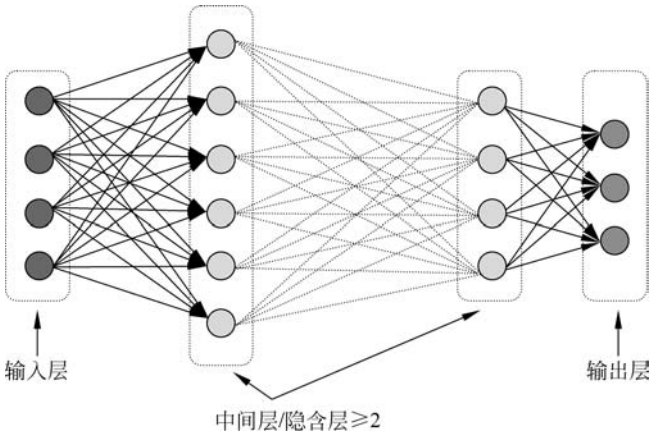


图 5-1　深度神经网络的基本结构

1）输入层

卷积神经网络的输入层可以处理多维数据。常见的，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 通道。与其他神经网络算法类似，由于使用梯度下降算法进行学习，卷积神经网络的输入特征需要进行标准化处理。具体地，在将学习数据输入卷积神经网络前，需在通道或时间/频率维对输入数据进行归一化，若输入数据为像素，也可将分布于[0，255]的原始像素值归一化至[0，1]区间。输入特征的标准化有利于提升卷积神经网络的学习效率和表现。

2）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 Softmax 函数输出分类标签。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。

3）隐含层

卷积神经网络的隐含层包含卷积层、池化层和全连接层三类常见类型神经元结构，在一些更为复杂的模型结构中可能有 Inception 模块、残差块等由卷积层和池化层按一定形式组合构成的网络基本单元。在常见神经网络类型中，卷积层和池化层为卷积神经网络所特有的。

2. 卷积层

1）卷积核

卷积层的功能是对输入数据进行特征提取，其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，加上一个偏差量，类似于一个前馈神经网络的神经元。

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，称为感受野，其含义可类比视觉皮层细胞的感受野。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：

$$Z^{l+1}(i,j)=[Z\otimes w^{l+1}](i,j)+b$$
$$=\sum_{k=1}^{K_l}\sum_{x=1}^{f}\sum_{y=1}^{f}[Z_k^l(s_0i+x,s_0j+y)w_k^{l+1}(x,y)]+b$$
$$(i,j)\in\{0,1,\cdots,L_{l+1}\},\quad L_{l+1}=\frac{L_l+2p-f}{s_0}+1$$

式中：求和部分等价于求解一次交叉相关，b 为偏差量；Z^l 和 Z^{l+1} 表示第 $l+1$ 层的卷积输入与输出，也称为特征图；L_{l+1} 为 Z^{l+1} 的尺寸，这里假设特征图长宽相同；$Z(i,j)$ 对应特征图的像素；K 为特征图的通道数；f、s_0 和 p 是卷积层参数，对应卷积核大小、卷积步长和填充点数。

上式以二维卷积核作为例子，一维或三维卷积核的工作方式与之类似，如图 5-2 所示。理论上卷积核也可以先翻转 180°，再求解交叉相关，其结果等价于满足交换律的线性卷积，但这样做在增加求解步骤的同时并不能为求解参数取得便利，因此线性卷积核使用交叉相关代替了卷积。

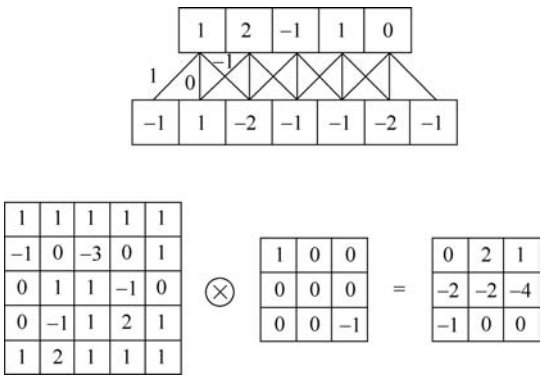


图 5-2　一维和二维卷积运算示例

特殊地，当卷积核是 $f=1,s_0=1$ 且不包含填充的单位卷积核时，卷积层内的交叉相关计算等价于矩阵乘法，并由此在卷积层间构建了全连接网络：

$$Z^{l+1}=\sum_{k=1}^{K_l}\sum_{x=1}^{L}\sum_{y=1}^{L}(Z_{i,j,k}^l w_k^{l+1}+b)=\boldsymbol{w}_{l+1}^{\mathrm{T}}\mathbf{Z}_{l+1}+b,\quad L^{l+1}=L$$

由单位卷积核组成的卷积层也称为网中网（Network-In-Network，NIN）或多层感知器卷积层（multilayer perceptron convolution layer，mlpconv）。单位卷积核可以在保持特征图尺寸的同时减少图的通道数，从而降低卷积层的计算量。完全由单位卷积核构建的卷积神经网络是一个包含参数共享的多层感知器（Multi-Layer Perceptron，MLP）。

在线性卷积的基础上，一些卷积神经网络使用了更为复杂的卷积，包括平铺卷积、反卷积和扩张卷积。平铺卷积的卷积核只扫过特征图的一部分，剩余部分由同层的其他卷

积核处理，因此卷积层间的参数仅被部分共享，有利于神经网络捕捉输入图像的旋转不变特征。反卷积或转置卷积将单个的输入激励与多个输出激励相连接，对输入图像进行放大。由反卷积和向上池化层构成的卷积神经网络在图像语义分割领域有应用，也用于构建卷积自编码器(Convolutional AutoEncoder，CAE)。扩张卷积在线性卷积的基础上引入扩张率以提高卷积核的感受野，从而获得特征图的更多信息，在面向序列数据使用时有利于捕捉学习目标的长距离依赖。

2) 卷积层参数

卷积层参数包括卷积核大小、步长和填充，三者共同决定了卷积层输出特征图的尺寸，是卷积神经网络的超参数。其中卷积核大小可以指定为小于输入图像尺寸的任意值，卷积核越大，可提取的输入特征越复杂。

卷积步长定义了卷积核相邻两次扫过特征图时位置的距离，卷积步长为 1 时，卷积核会逐个扫过特征图的元素，步长为 n 时，会在下一次扫描跳过 $n-1$ 像素。

由卷积核的交叉相关计算可知，随着卷积层的堆叠，特征图的尺寸会逐步减小，例如 16×16 的输入图像在经过单位步长、无填充的 5×5 的卷积核后，会输出 12×12 的特征图。为此，填充是在特征图通过卷积核之前人为增大其尺寸以抵消计算中尺寸收缩影响的方法。常见的填充方法为按 0 填充(图 5-3)和重复边界值填充。

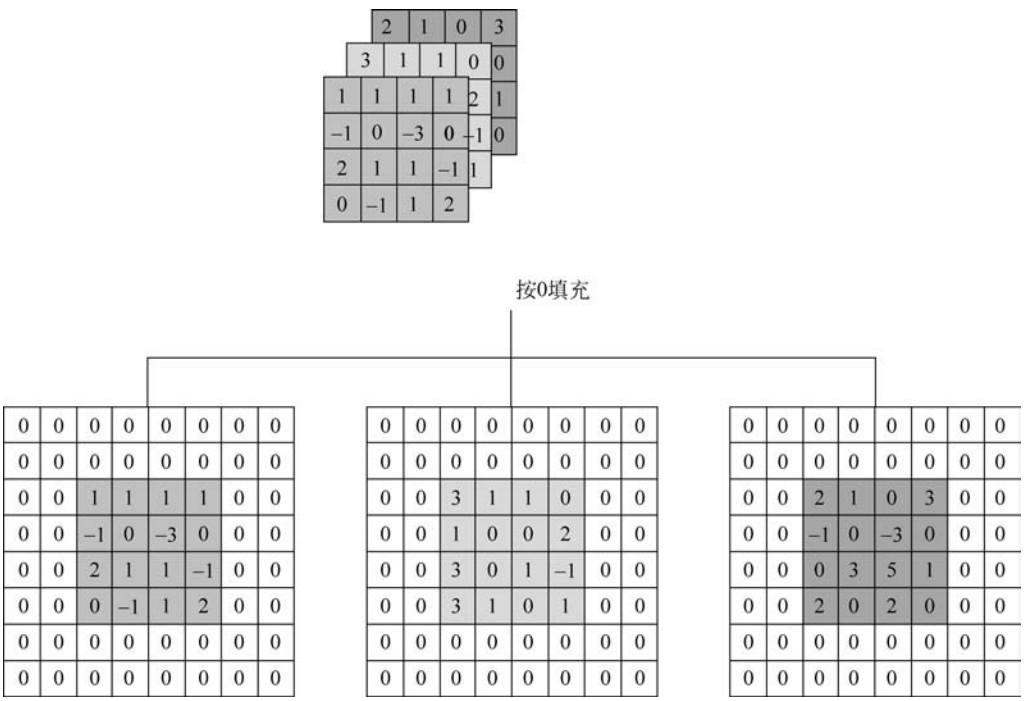


图 5-3　卷积核中 RGB 图像的按 0 填充

填充依据其层数和目的可分为四类：

(1) 有效填充：完全不使用填充，卷积核只允许访问特征图中包含完整感受野的位

置。输出的所有像素都是输入中相同数量像素的函数。使用有效填充的卷积称为窄卷积，窄卷积输出的特征图尺寸为$(L-f)/s+1$。

(2) 相同填充/半填充：只进行足够的填充来保持输出和输入的特征图尺寸相同。相同填充下特征图的尺寸不会缩减，但输入像素中靠近边界的部分相比于中间部分对特征图的影响更小，即存在边界像素的欠表达。使用相同填充的卷积称为等长卷积。

(3) 全填充：进行足够多的填充使得每个像素在每个方向上被访问的次数相同。步长为1时，全填充输出的特征图尺寸为$L+f-1$，大于输入值。使用全填充的卷积称为宽卷积。

(4) 任意填充：介于有效填充和全填充之间，人为设定的填充，较少使用。

再看先前的例子，若16×16的输入图像在经过单位步长的5×5的卷积核之前先进行相同填充，则会在水平和垂直方向填充两层，即两侧各增加2个像素($p=2$)变为20×20大小的图像，通过卷积核后，输出的特征图尺寸为16×16，保持了原本的尺寸。

3. 激励函数

卷积层中包含激励函数以协助表达复杂特征，其表达式为

$$A_{i,j,k}^{l}=g(Z_{i,j,k}^{l})$$

类似于其他深度学习算法，卷积神经网络通常使用ReLU，其他类似ReLU的变体包括有Leaky ReLU、参数化ReLU、随机化ReLU、指数线性单元等，具体细节见第4章。在ReLU出现以前，Sigmoid函数和双曲正切函数也有使用。

激励函数操作通常在卷积核之后，一些使用预激活技术的算法将激励函数置于卷积核之前。在一些早期的卷积神经网络研究，例如LeNet-5中，激励函数在池化层之后。

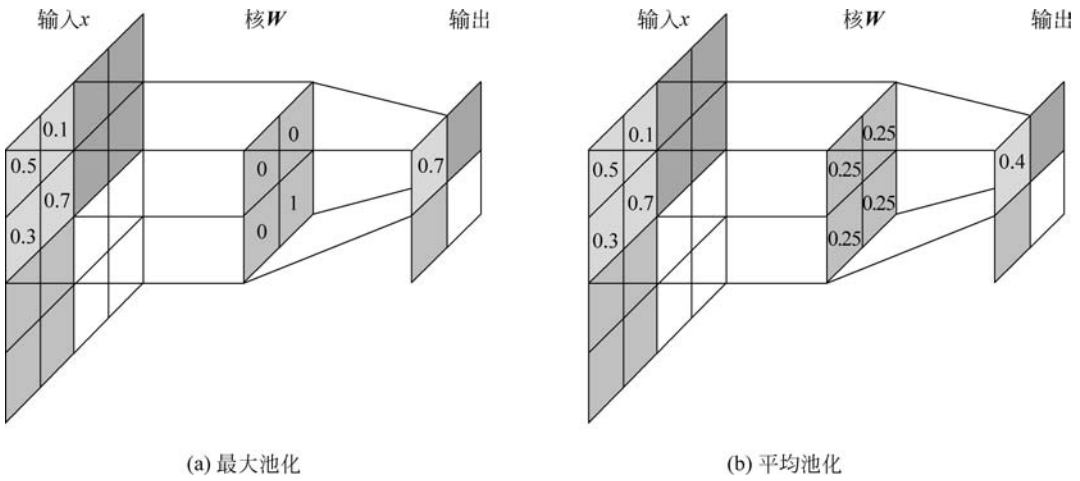
4. 池化层

在卷积层进行特征提取后，输出的特征图会被传递至池化层进行特征选择和信息过滤。池化是卷积神经网络中另一个重要的概念，它实际上是一种形式的降采样。有多种不同形式的非线性池化函数，而其中最大池化和平均池化最为常见，如图5-4所示。最大池化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；而平均池化是对每个子区域输出平均值。最大池化是经常采用的方法，在直觉上是有效的，原因在于：在发现一个特征之后，它的精确位置远不及它与其他特征的相对位置重要。池化层会不断地减小数据的空间大小，因此参数的数量和计算量会下降，这在一定程度上也控制了过拟合。

池化层通常会分别作用于每个输入的特征并减小其大小。当前最常用的池化层形式是每隔2个元素从图像划分出2×2的区块，然后对每个区块中的4个数取平均值或池化值，这将会减少75%的数据量。

池化层的引入是仿照人的视觉系统对视觉输入对象进行降维和抽象。在卷积神经网络过去的工作中，研究者普遍认为池化层有如下三个功效：

(1) 特征不变性：池化操作使模型更加关注是否存在某些特征而不是特征具体的



(a) 最大池化　　(b) 平均池化

图 5-4　池化示意

位置。

(2) 特征降维：池化相当于在空间范围内做了维度约减，从而使模型可以抽取更广范围的特征。同时减小了下一层的输入大小，进而减少计算量和参数个数。

(3) 在一定程度上防止过拟合，更方便优化。

5. 扁平化层

卷积层之后是无法直接连接密集全连接层的，需要把卷积层的数据做扁平化处理，然后直接加上全连接层。也就是把三维(height，width，channel)的数据压缩成长度为 height×width×channel 的一维数组，然后再与全连接层连接。其中 height 和 width 分别代表图像的平面尺寸的高度和宽度，而 channel 表示卷积通道数，也就是上一层采用的卷积核个数。

从图 5-5 中可以看到，随着网络的深入，图像块平面尺寸越来越小，但是通道数往往会增大。图中表示长方体的平面尺寸越来越小，这主要是卷积和池化的结果，而通道数由每次卷积的卷积核个数来决定。

6. 全连接层

卷积神经网络中的全连接层等价于传统前馈神经网络中的隐含层。全连接层位于卷积神经网络隐含层的最后部分，并只向其他全连接层和输出层传递信号。特征图在全连接层中会失去空间拓扑结构，被展开为向量并通过激励函数。

按表征学习观点，卷积神经网络中的卷积层和池化层能够对输入数据进行特征提取，全连接层的作用是对提取的特征进行非线性组合以得到输出，即全连接层本身不被期望具有特征提取能力，而是试图利用现有的高阶特征完成学习目标。

在一些卷积神经网络中，全连接层的功能也由全局均值池化(Global Average Pooling，GAP)取代，全局均值池化会将特征图每个通道的所有值取平均。例如，若有

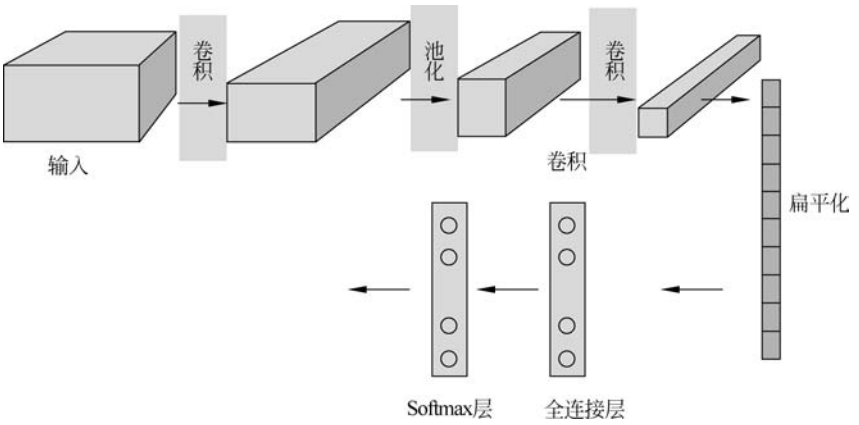


图 5-5　扁平层示意图

7×7×256 维的特征图，全局均值池化将返回一个 256 维的向量，其中每个元素都是 7×7、步长为 7、无填充的均值池化值。

5.2　卷积神经网络与全连接网络的关系

卷积神经网络与前向神经网络没有很大差别，传统神经网络本质上就是多个全连接层的叠加，而卷积神经网络无非是把全连接层改成卷积层和池化层，即把传统的由一个个神经元组成的层变成了由滤波器组成的层。但卷积神经网络有两种独特机制来减少参数量：一是连接稀疏性；二是参数共享。

5.2.1　连接稀疏性

例如，图像尺寸为 8×8，即 64 像素，假设有 36 个单元的全连接层，如图 5-6 所示。这一层需要 64×36＋36＝2340 个参数，其中 64×36 是全连接的权重，每一个连接都需要一个权重 w；而另外 36 个参数对应的偏置项 b，每个输出节点对应一个偏置项。

卷积神经网络的第一个特性是网络部分连通特性，即模拟局部感受野，以图 5-7 所示的 9 个单元滤波器为例来说明。由卷积的操作可知，输出图像中的任何一个单元只与输入图像的一部分有关系。如图 5-7 所示，左侧图像的阴影区域通过滤波器与右侧对应输出单元相连接。而该输出单元与左侧区域的其他单元没有连接，因此连接是稀疏的。而传统前向神经网络中由于都是全连接，所以输出的任何一个单元都要受输入的所有单元影响。这样无形中会降低图像识别效果。从前面的描述可知，滤波器是用来检测特征的，每个滤波器都侧重某一方面特征的描述和发现，因此不同的滤波器能够描述图像的不同模式或特征。因此，期望每一个区域都有自己的专属特征，不希望其受到其他区域的影响。

这种局部稀疏性或者说局部感受野使得每个输出单元只有 9 个连接，因此对应的连

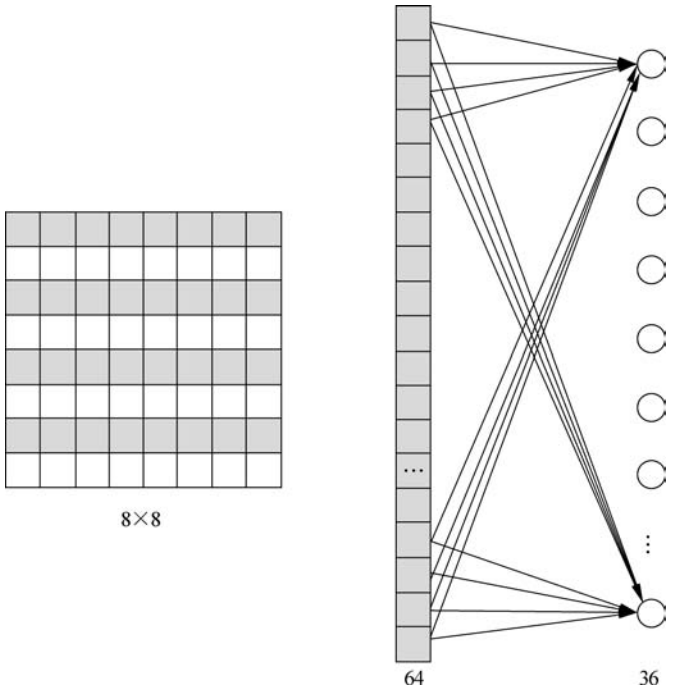


图 5-6　全连接网络示意图

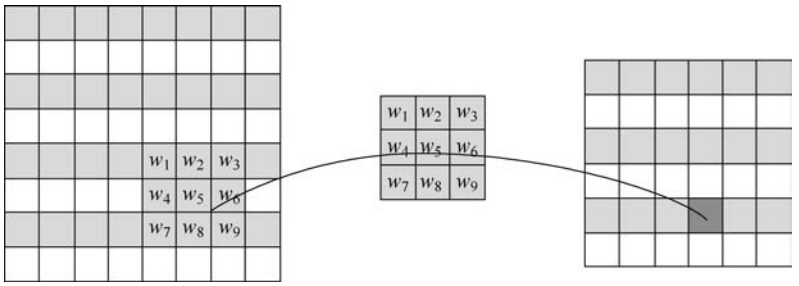


图 5-7　连接的稀疏性示意图

接数就是 9×36，那么参数是不是 9×36＋36 呢？答案当然不是，此时参数的个数为 9＋1 个，其中 9 个为卷积核参数，1 个为卷积核所对应的偏置量。为什么会出现这种结果，下面进一步剖析其产生原因，即参数共享机制。

5.2.2　参数共享机制

仍然以图 5-7 所示的 9 个单元滤波器为例来说明。滤波器有几个单元就几个参数，所以图中所示总共只有 9 个参数。暂时不考虑偏置项，为什么是 9 个参数，而不是 16×9 个参数呢，这是因为对于不同的区域都共享同一个滤波器，因此共享这同一组参数，如图 5-8 所示。

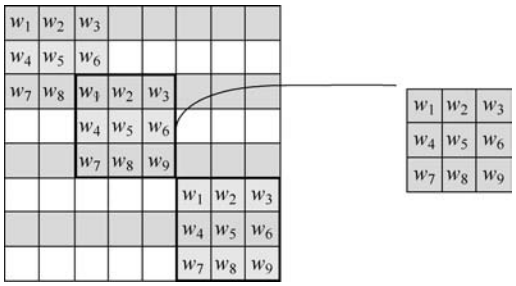


图 5-8　卷积神经网络示意图

如前所述，每个滤波器都侧重某一方面特征的描述和发现，因此不同的滤波器能够描述图像的不同模式或特征。通常情况下，某一个特征很可能在不止一个地方出现，比如“竖直边界”就可能在一幅图中多处出现，鸟的羽毛就可能出现在图像的不同位置，因此共享同一个滤波器有客观存在的合理性作支撑。由于采用这种参数共享机制，使得卷积神经网络的参数数量大大减少。这样，就可以用较少的参数训练出更加好的模型，而且可以有效地避免过拟合。

同样，由于滤波器的参数共享，即使图片进行了一定的平移操作，同样可以识别出特征，这种特性称为平移不变性。因此，模型也会更加稳健。

连接稀疏性使得卷积神经网络的参数减少，而参数共享机制使得网络参数进一步减少，正是由于上面这两大优势，使得卷积神经网络超越了传统神经网络，开启了神经网络的新时代。

5.3　典型的卷积神经网络

CNN 的开山之作是杨立昆提出的 LeNet-5[7]，而其真正的爆发期是在 2012 年 AlexNet[9]取得 ImageNet 比赛分类任务的冠军之后，当年 AlexNet 的分类准确率远远超过传统方法。该模型能够取得成功主要有三个原因：

（1）海量有标记的训练数据支撑，也就是李飞飞团队提供的大规模有标记的数据集 ImageNet；

（2）计算机硬件的支持，尤其是 GPU 的出现，为复杂的计算提供了强大的支持；

（3）算法的改进，包括网络结构加深、数据增强（数据扩充）、ReLU 激活函数和 Dropout 等。

AlexNet 之后，深度学习发展迅速，分类准确率每年都有很大提高，表 5-1 给出了四种不同模型的结构对比。图 5-9 展示了模型的变化情况，随着模型的变深，Top-5 错误率也越来越低，在 2017 年已经降低到 2.3%左右；同样，对于 ImageNet 数据集，人眼的辨识错误率大概为 5.1%，这说明深度学习在图像识别领域的能力已经超过了人类。

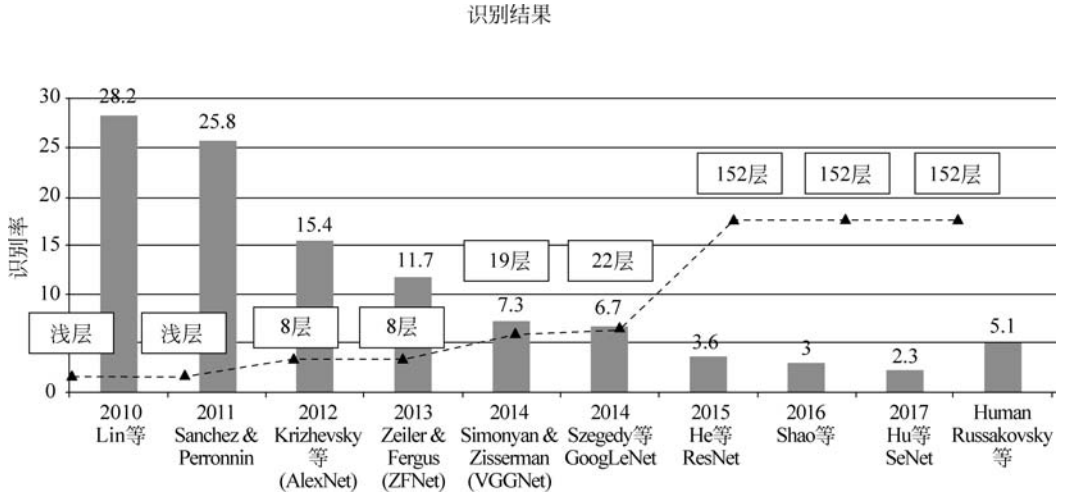


图 5-9　ILSVRC 历年的 Top-5 错误率

表 5-1　四种模型对比[14]

模　型　名	AlexNet	VGGNet	GoogLeNet	ResNet
提出时间	2012 年	2014 年	2014 年	2015 年
层数	8	19	22	152
Top-5 错误率/%	15.4	7.3	6.7	3.57
数据增强	+	+	+	+
Inception	−	−	+	−
卷积层数	5	16	21	151
卷积核大小	11,5,3	3	7,1,3,5	7,1,3,5
全连接层数	3	3	1	1
全连接层大小	4096,4096,1000	4096,4096,1000	1000	1000
Dropout	+	+	+	+
局部相应归一化	+	−	+	−
批归一化	−	−	−	+

5.3.1　LeNet-5 网络

LeNet-5 网络可以认为是目前深度经典卷积神经网络的开创者，该网络共包含 7 层（不包括输入层），如图 5-10 所示。

网络各个层的设置具体如下：

1. C_1 卷积层

C_1 卷积层由 6 个卷积核构成。输入图像大小是 32×32，输入图像首先经过 6 个尺寸为 5×5 的卷积核进行滤波后得到特征图，由于每个卷积核（滤波器）产生一个特征映射图，所以 6 个卷积核共产生 6 个特征图，特征图的大小为 28×28。由于每个卷积核滤波操作对整幅图像处理，以像素步长 1 进行移动，那么水平和垂直方向各有(32−5)/1+

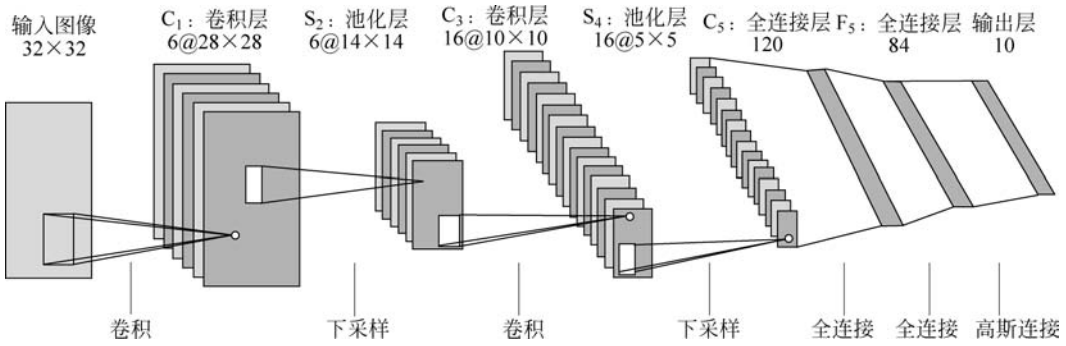


图 5-10　LeNet-5 结构[7]

1＝28 次卷积操作，所以得到输出特征图尺寸为 28×28。C_1 有 156 个可训练参数（每个滤波器 5×5＝25 个滤波器参数和 1 个偏差参数，共 6 个滤波器，共(5×5＋1)×6＝156 个参数）；C_1 输出特征图大小为 28×28，因此共产生 156×(28×28)＝122304 个连接。

2. S_2 池化层

S_2 池化层生成 6 个 14×14 的输出特征图。S_2 输出特征图中的每个单元与 C_1 中相对应特征图的 2×2 邻域相连接。S_2 层每个单元的 4 个输入相加，乘以一个可训练参数，再加上一个可训练偏置，然后通过 Sigmoid 函数（或其他函数）计算。可训练系数和偏置控制着激活函数的非线性程度。每个单元的 2×2 感受野并不重叠，因此 S_2 中每个特征图的大小是 C_1 中特征图大小的 1/4（行和列各 1/2）。由于每个特征图只需要一个权值和一个偏置项，S_2 层有 6×(1＋1)＝12 个可训练参数，共形成 14×14×(2×2＋1)×6＝5880 个连接。

3. C_3 卷积层

C_3 卷积层同样通过 5×5 的卷积核去处理 S_2 层的输出特征图，得到的特征图尺寸为 10×10。（同样每个方向移动步长为 1，那么(14－5)/1＋1＝10）。C_3 层输出 16 个特征图，但需要注意的是，C_3 中不是由单独 16 个卷积核构成，而是每个特征图由 S_2 中所有 6 个或者其中的 3 个、4 个特征图组合而成。不把 S_2 中的每个特征图连接到每个 C_3 的特征图的原因：第一，不完全的连接机制将连接的数量保持在合理的范围内；第二，也是最重要的一点，通过不同特征图的不同输入来抽取上一层的不同特征。相应地，16 个特征图的组合方式如表 5-2 所示。这是一种稀疏连接的方式，可以减少连接的数量，同时打破了网络的对称性。

表 5-2　S_2 层到 C_3 层的连接方式

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X

续表

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

虽然原始特征图只有 6 种，但是由于不同的选择方式，构造了 16 个新特征图。由于每种组合里原始特征图需要选择不同的系数，因此也可以将新生成的 16 个特征图看作由 60 个滤波器构成(6 组 3 个特征图进行组合，9 组 4 个特征图进行组合，还有一组是全部 6 个特征图进行组合，则 $3\times6+4\times9+6=60$)，而每个滤波器的参数都是 $5\times5=25$ 个，而最后生成 16 个特征图，每个特征图需要一个偏置项，所以参数为 $60\times5\times5+16=1516$ 个，总连接数为 $1516\times10\times10$ 个。

4. S_4 池化层

S_4 池化层生成输出 16 个 5×5 大小的特征图。特征图中的每个单元与 C_3 中相应特征图的 2×2 邻域相连接，与 C_1 和 S_2 之间的连接一样。S_4 层有 32 个可训练参数(每个特征图需要一个可训练参数和一个偏置，因此有 $16\times(1+1)=32$ 和 $16\times(2\times2+1)\times5\times5=2000$ 个连接。

5. C_5 卷积层

C_5 卷积层输出 120 个特征图。每个单元与 S_4 层的全部 16 个特征图的 5×5 邻域相连。由于 S_4 层特征图的大小也为 5×5(同滤波器一样)，故 C_5 特征图的大小为 1×1(5－5＋1＝1)，这构成了 S_4 和 C_5 之间的全连接。之所以仍将 C_5 标示为卷积层而非全连接层，是因为本例中输入图像尺寸为 32×32，导致此时输出尺寸为 1；如果 LeNet-5 的输入变大，而其他的保持不变，此时特征图的维数就会比 1×1 大，因此卷积层更符合本身特性。C_5 层有 $120\times(16\times5\times5+1)=48120$ 个可训练连接，由于与全部 16 个单元相连，故只加一个偏置。

6. F_6 全连接层

F_6 全连接层输出有 84 个单元，与 C_5 层全相连。输出层选择 84 个单元的原因是来自输出层的设计。因为在计算机中字符的编码是 ASCII 编码，这些图是用 $7\times12=84$ 大小的位图表示的。F_6 层输出有 $84\times(120\times(1\times1)+1)=10164$ 个可训练参数。如同经典神经网络，F_6 层计算输入向量和权重向量之间的点积，再加上一个偏置，然后将其传递给激活函数函数产生每个单元的一个状态。

7. 输出层

输出层由欧几里得径向基函数单元组成，每类一个单元，每个有 84 个输入。

5.3.2 AlexNet 网络

2012 年著名人工智能学者杰夫·辛顿和他的学生亚历克斯·克里热夫斯基(Alex krizhevsky)等创造了一个“大型的深度卷积神经网络”,赢得了该年度 ImageNet 大规模视觉识别挑战赛(iLsvrc)的冠军,其论文 *ImageNet classification with deep convolutional networks*[9] 被引用约 2.7 万次,被业内普遍视为行业最重要的论文之一。ILSVRC 比赛被誉为计算机视觉的年度奥林匹克竞赛,全世界的团队相聚一堂进行技术竞技。2012 年采用 CNN 获得 Top-5 误差率 15.4%(Top-5 误差率是指给定一张测试图像,其标签不在模型认为最有可能的 5 个结果中的概率),远优于当时排名第二模型 26.2%的错误率,震惊了整个计算机视觉界。自 2012 年起,CNN 网络风靡多年。

图 5-11 是 AlexNet 结构,由于采用两台 GPU 服务器进行运算,所以分成上、下两个部分。图 5-12 为 AlexNet 结构分析图。由图中可知,共有 8 个隐含层,包括 5 个卷积层和 3 个全连接层,图中每个卷积层标注下方给出了每层的输入特征图尺寸,其中前两个数字为平面尺寸,后一个数字为通道数。以卷积层 1 为例,输入图像尺寸为 227×227×3,说明原始输入图像是尺寸 227×227 的三通道彩色图像。需要说明的是,图 5-11 与图 5-12 的输入图像尺寸分别是 224 和 227,这是由于作者在 2012 年和 2014 年的文章中就分别给出这 2 种配置。实际上 224 尺寸图像可以通过补 1 个零获得与 227 相同的处理结果。该层在两个 GPU 上各采用了 48 个尺寸为 11×11 的卷积核,步长为 4,采用的非线性激活函数为 ReLU 函数,池化尺寸为 3×3、步长为 2,并且在该层进行了层标准化。首先输入图像经过卷积核作用后,平面尺寸变为 55×55,即(227−11)/4+1=55。因此,经过 96 个卷积核作用后,输出特征图为 55×55×96。然后再进入池化,则根据上述设置,池化后的特征图尺寸变为 27×27,即(55−3)/2+1=27,而通道数 96 保持不变。因此,经过卷积层 1 的整个作用后,将 227×227×3 的特征图转换为 27×27×96 的特征图。依次类推,图中给出了每层作用的参数分析结果,包括后续的其他 4 个卷积层和 3 个全连接层。

图 5-13 为 AlexNet 的网络结构简图[9]。AlexNet 的结构很简单,只是 LeNet 的放大版,输入是一个 227×227 的图像,经过 5 个卷积层、3 个全连接层(包含一个分类层),输出到最后的标签空间。图中也给出了相应的图像尺寸以及通道数变化的情况。

除了上述结构上的创新外,AlexNet 网络还有很多独到之处:

(1) 非线性激活函数 ReLU。在 AlexNet 之前,一般神经元的激活函数会选择 Sigmoid 函数或者 tanh 函数,然而在研究 AlexNet 网络时,发现在训练时间的梯度衰减方面,这些非线性饱和函数要比非线性非饱和函数慢很多。在 AlexNet 中用的非线性非饱和函数 ReLU。实验结果表明,采用 ReLU 函数后相比原来的 tanh 函数,其训练速度大大加快。

(2) 双 GPU 并行处理。为提高运行速度和提高网络运行规模,AlexNet 网络中采用双 GPU 的设计模式,即每个 GPU 负责 1/2 的运算处理,且规定 GPU 只能在特定的层进行通信交流。如图 5-11 所示,卷积层 1、卷积层 2、卷积层 4 和卷积层 5 仅连接相同 GPU 上的特征图,而卷积层 3 和后面的三个全连接层,连接前面层所有 GPU 的特征图。

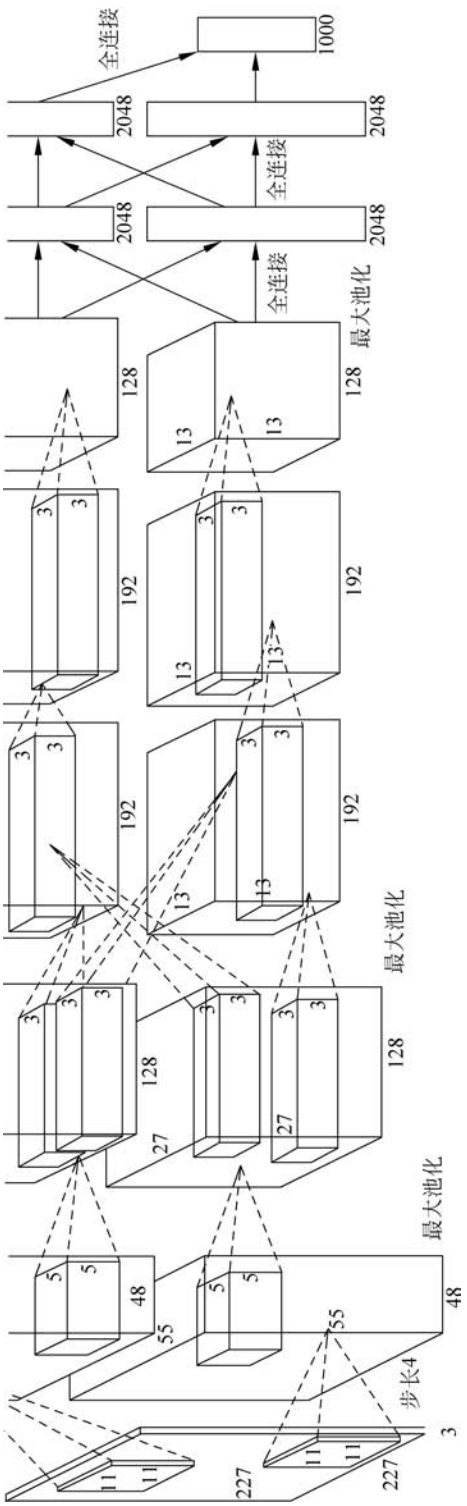


图 5-11　AlexNet 结构[9]

图 5-12　AlexNet 的网络结构分析图

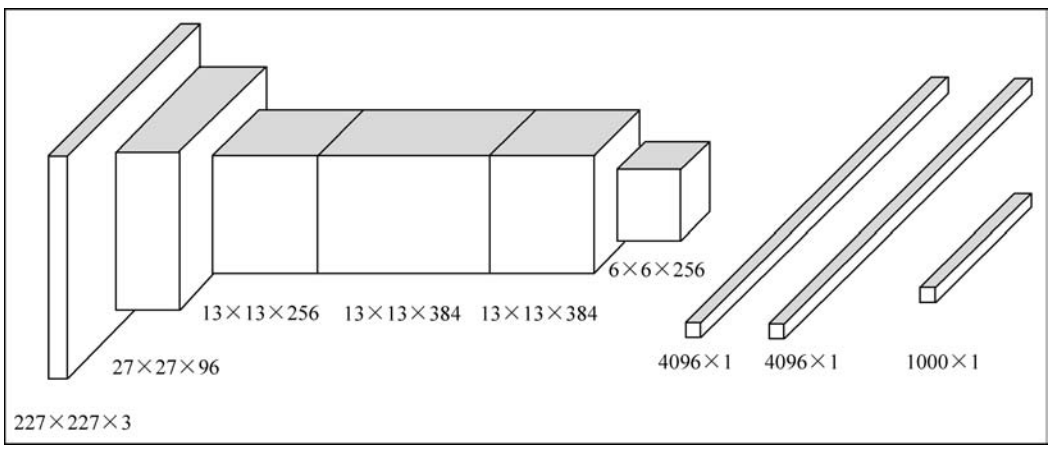


图 5-13　AlexNet 的网络结构简图

（3）局部响应归一化技术。ReLU 本来是不需要对输入进行标准化，但进行局部标准化能提高性能。局部响应归一化（Local Response Normalization，LRN）实际就是利用临近的数据做归一化，即

$$b_{x,y}^{i}=a_{x,y}^{i}/\left(k+\alpha\sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)}(a_{x,y}^{j})^{2}\right)^{\beta} \tag{5.1}$$

式中：$a_{x,y}^{i}$ 为第 i 个卷积核的输出特征图(x,y)坐标经过了 ReLU 激活函数的输出；n 为相邻的几个卷积核；N 为这一层总卷积核数量；k、n、α 和 β 为超参数，这些超参数值是在验证集上通过实验获取，最终选择 $k=2$，$n=5$，$\alpha=10^{-4}$，$\beta=0.75$。

这种归一化操作受真实神经元的某种行为启发，卷积核矩阵的排序是随机任意的，并且在训练之前就已经决定好顺序，因此这种归一化机制实现了某种形式的横向抑制。该策略贡献了约 1.2%的误差率下降。LRN 作为深度学习训练时的一种提高准确度的技术方法，一般在激活、池化后进行。需要说明的是，这种归一化也可在通道内进行，即 i 代表通道序号。

(4) 交叠池化。池化层是对相同卷积核特征图中周围神经元输出进行总结，也可以理解成按一定步长移动对前面卷积层的结果进行分块，对块内的卷积映射结果做总结。在 AlexNet 网络中，使用了重叠的最大池化，而此前 CNN 中普遍使用平均池化。AlexNet 全部使用最大池化，避免平均池化的模糊化效果；而重叠体现了其输入会受到相邻池化单元的输入影响。并且 AlexNet 中提出步长比池化核的尺寸小，池化层的输出之间会有重叠和覆盖，提升了特征的丰富性。实验结果表明，使用带交叠的池化相比于传统方法，其 Top-1 和 Top-5 上分别提高了 0.4%和 0.3%，在训练阶段有避免过拟合的作用。

(5) Dropout 技术。在 AlexNet 网络中，最后的三个全连接层均使用了 Dropout 技术。训练时使用 Dropout 随机忽略一部分神经元，以避免模型过拟合。AlexNet 将 Dropout 实用化，通过实践证实了它的效果。

5.3.3 ZFNet 网络

2013 年，ImageNet 大规模视觉识别挑战赛(ILSVRC 2013)的冠军获胜模型是纽约大学 Matthew Zeiler 和 Rob Fergus 设计的卷积神经网络[10]。该网络以两个作者的名字来命名，即 Zeiler&Fergus Net，简称 ZFNet。ZFNet 模型网络结构没有太大突破，其本质是 AlexNet 架构的微调优化版，只是在卷积核和步幅上面做了一些优化调整。ZFNet 的两个主要贡献：一是该网络使用一个多层的反卷积网络来可视化训练过程中特征的演化情况，并及时发现潜在的问题；二是根据遮挡图像局部对分类结果的影响，来探讨到底哪部分输入信息对分类任务更为重要。

1. 利用反卷积实现特征可视化

为了清楚了解卷积神经网络的机理和性能优越的原因，需要对 CNN 学习到的特征进行分析与解释，为此 ZFNet 网络提出采用反卷积网络技术进行特征可视化[15]。反卷积网络可以看成是卷积网络的逆过程，最早提出用于无监督学习，而在 ZFNet 中，反卷积技术仅用于特征可视化。反卷积可视化是以各层得到的特征图作为输入，通过反卷积网络得到最终结果，用以验证显示各层提取到的特征图。例如，想了解 AlexNet 网络的每层都学习到什么信息，可以对特定层的特征图进行反卷可视化。将 AlexNet 网络的第 3 个卷积层输出特征图后面接一个反卷积网络，通过反池化、反激活、反卷积这样的一系列

过程，把本来 13×13 大小的特征图反卷放大(AlexNet 第三个卷积层输出特征图 13×13×384)，得到一张与原始输入图片一样大小的图片(227×227)，这样就能够了解第 3 个卷积层学习到的信息。图 5-14 给出了利用反卷积实现特征可视化过程。从图中可以看出，右侧部分是图像卷积网络前向识别处理过程，即每层处理后的特征图，先经过卷积层进行卷积和非线性激活，然后进行池化得到池化后的特征图。其中采用 ReLU 函数进行激活，采用最大池化方法进行池化。左侧给出了反卷积实现可视化过程，首先得到上面所有层重建后的信息，然后进行正向处理相反的过程，即反池化、反激活和反卷积三步，得到重建后的图像。需要注意的是，左侧图中的反激活就是激活、而反卷积采用转置卷积滤波器进行反卷。这样处理的原因将在后续反池化、反激活、反卷积的基本思想中进行解释[10]。

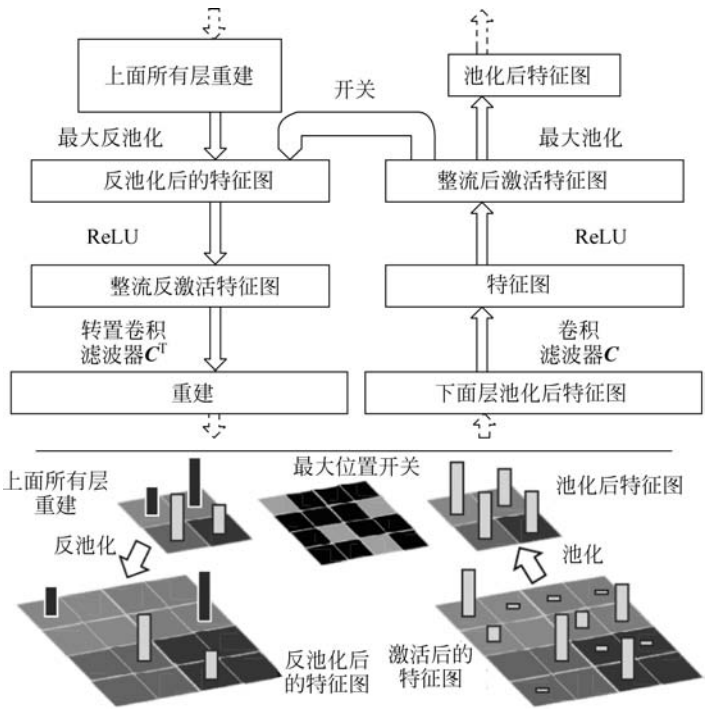


图 5-14　利用反卷积实现特征可视化过程[10]

1) 反池化

从前面定义可知，池化过程不可逆。以最大池化为例，为了进行反池化，需要记录池化过程中最大激活值的坐标位置。然后在反池化时，把池化过程中最大激活值所在位置坐标值激活，其他位置的值置为 0。需要注意的是，这种反池化过程只是一种近似过程(因为在正向池化时，其他位置的值并不为 0)。

图 5-15 给出反池化两个示例，每个示例中左侧都是池化过程，右侧是反池化过程。假设池化块的大小是 3×3，采用最大池化后，左侧和右侧示例分别得到输出神经元激活值为 9。池化是下采样过程，因此经过池化后 3×3 大小的像素块就变成 1 个值。而反池化是池化的逆过程，是一个上采样过程，需要将 1 个值恢复成 3×3 大小的像素块。为了使池化可逆，需要记录池化过程中最大值位置坐标(−1,1)和(0,0)。因此，在进行反池

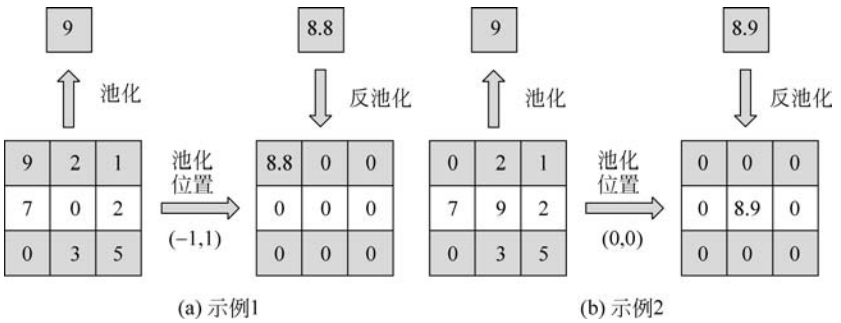


图 5-15　反池化示意图

化时，根据记录的位置坐标，把对应的像素值填充回该位置，而其他位置的神经元激活值全部为 0。需要注意的是，本例中给出的坐标位置，是以像素块的中心点(0,0)来记录其他的位置，因此图 5-15(a)池化位置为(−1,1)，而图 5-15(b)池化位置为(0,0)。

2) 反激活

在 AlexNet 中，ReLU 函数是用于保证每层输出的激活值都是正数，因此对于反向过程，同样需要保证每层的特征图为正值，因此反激活过程和激活过程都是直接采用 ReLU 函数。

3) 反卷积

为了说明反卷积原理，下面举例来对比卷积和反卷积的过程。如图 5-16 所示，假设

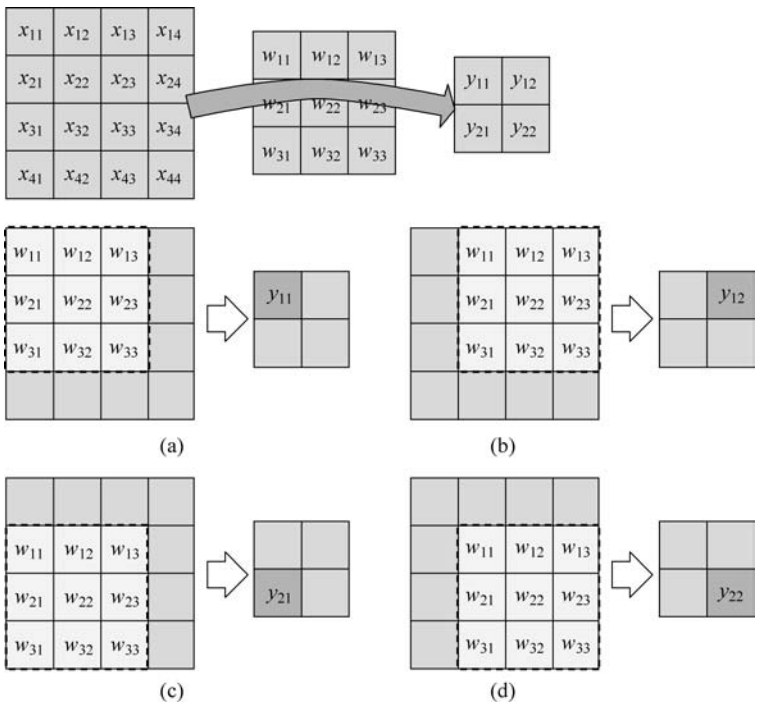


图 5-16　卷积示意图

输入图片尺寸为4×4，卷积核尺寸为3×3，步长为1，则卷积后的特征图尺寸为2×2。为了便于对应表示，可以将卷积写成矩阵运算形式。由于输入尺寸元素个数为4×4，将其展开成16×1的向量表示为$\boldsymbol{A}\in\mathbb{R}^{16\times1}$，输出图像尺寸为2×2，将其展开成4×1的向量表示为$\boldsymbol{B}\in\mathbb{R}^{4\times1}$，则将卷积核表示为矩阵$\boldsymbol{C}\in\mathbb{R}^{4\times16}$。需要注意的是，矩阵$\boldsymbol{C}$是为了后期矩阵运算而扩展的卷积核矩阵，则卷积运算可以表示为

$$\boldsymbol{B}=\boldsymbol{C}\boldsymbol{A} \tag{5.2}$$

式中

$$\boldsymbol{A}=\begin{bmatrix}x_{11}\\x_{12}\\x_{13}\\x_{14}\\x_{21}\\x_{22}\\x_{23}\\x_{24}\\x_{31}\\x_{32}\\x_{33}\\x_{34}\\x_{41}\\x_{42}\\x_{43}\\x_{44}\end{bmatrix},\quad \boldsymbol{C}=\begin{bmatrix}w_{11}&0&0&0\\w_{12}&w_{11}&0&0\\w_{13}&w_{12}&0&0\\0&w_{13}&0&0\\w_{21}&0&w_{11}&0\\w_{22}&w_{21}&w_{12}&w_{11}\\w_{23}&w_{22}&w_{13}&w_{12}\\0&w_{23}&0&w_{13}\\w_{31}&0&w_{21}&0\\w_{32}&w_{31}&w_{22}&w_{21}\\w_{33}&w_{32}&w_{23}&w_{22}\\0&w_{33}&0&w_{23}\\0&0&w_{31}&0\\0&0&w_{32}&w_{31}\\0&0&w_{33}&w_{32}\\0&0&0&w_{33}\end{bmatrix}^{\mathrm{T}},\quad \boldsymbol{B}=\begin{bmatrix}y_{11}\\y_{12}\\y_{21}\\y_{22}\end{bmatrix} \tag{5.3}$$

可以看出，卷积核矩阵$\boldsymbol{C}$的4行的每一行就对应图5-16(a)、(b)、(c)和(d)四种情况。图5-16(a)中，原卷积核为

$$\boldsymbol{C}_{1,\mathrm{origin}}=\begin{bmatrix}w_{11}&w_{12}&w_{13}\\w_{21}&w_{22}&w_{23}\\w_{31}&w_{32}&w_{33}\end{bmatrix} \tag{5.4}$$

为了后期矩阵计算，将卷积核对应图像尺寸进行拓展成4×4，则拓展后的卷积核为

$$\boldsymbol{C}_1=\begin{bmatrix}w_{11}&w_{12}&w_{13}&0\\w_{21}&w_{22}&w_{23}&0\\w_{31}&w_{32}&w_{33}&0\\0&0&0&0\end{bmatrix} \tag{5.5}$$

将$\boldsymbol{C}_1$矩阵矢量化，则得到式(5.2)和式(5.3)中的卷积矩阵$\boldsymbol{C}$的第一行。同理，图5-16(b)、(c)和(d)所示卷积核分别对应卷积矩阵$\boldsymbol{C}$的第二至第四行。

有了前面卷积的矩阵分析，则很容易理解反卷积的矩阵形式。对于反卷积，若输入尺寸为 2×2 的图像，矢量化后表示为 $\boldsymbol{B}' \in \mathbb{R}^{4\times 1}$，则将反卷积等同于如下矩阵运算，即

$$\boldsymbol{A}' = \boldsymbol{C}^{\mathrm{T}}\boldsymbol{B}' \tag{5.6}$$

其中反卷积后的特征图尺寸 4×4，矢量化后表示为 $\boldsymbol{A}' \in \mathbb{R}^{16\times 1}$，反卷积如图 5-17 所示。

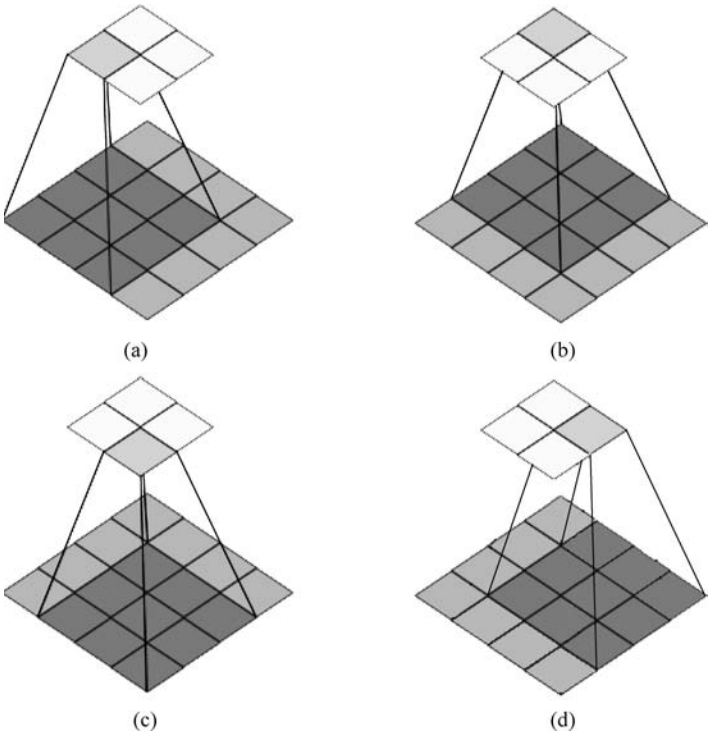


图 5-17　反卷积示意图

经过上面的分析可以看出，反卷积过程非常简单，就是采用卷积过程中转置后的滤波器作为卷积核，直接按照卷积过程运算。因此，反卷积本质上就是转置卷积。

2. ZFNet 网络结构分析

图 5-18 给出了 ZFNet 网络结构简图，网络模型参数配置如表 5-3 所示。表 5-3 给出了每个卷积核和池化层的具体参数配置，以及处理前和处理后的图像尺寸的变化。从图 5-18 和表 5-3 中可以看出，ZFNet 模型包括 5 个卷积层和 3 个全连接层。从图中可以看出，ZFNet 模型网络结构与 AlexNet 网络结构类似，只是在卷积核和步幅上面做了一些优化调整，其本质是 AlexNet 架构的微调优化版。

ZFNet 与 AlexNet 的区别：

(1) ZFNet 网络的第一个卷积层的卷积核尺寸由 11×11 改为 7×7，步长由 4 改为 2。之所以对这些参数进行调整，是由于对特征图可视化而得出的一些结论。通过可视化 AlexNet 第一层和第二层的特征，发现比较大的步长和卷积核提取的特征不理想，所以缩小了第一层的卷积核的尺寸，实验结果也证明了这种尺寸变化提升了图像分类性能。

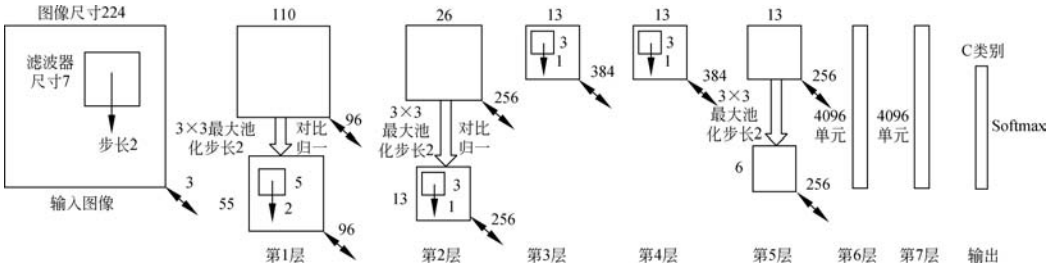


图 5-18　ZFNet 网络结构简图

表 5-3　ZFNet 模型参数配置

类　　型	核尺寸(卷积核个数)步长、补零和非线性激活函数	输入特征图尺寸	输出特征图尺寸
卷积层 1 (Conv1)	7×7 卷积核(96 个) stride=2 padding=0 ReLU	224×224×3	110×110×96
池化	3×3 stride=2 padding=0	110×110×96	55×55×96
归一化	对比归一化		
卷积层 2 (Conv2)	5×5 卷积核(256 个) stride=2 padding=0 ReLU	55×55×96	26×26×256
池化	3×3 stride=2 padding=0	26×26×256	13×13×256
归一化	对比归一化		
卷积层 3 (Conv3)	3×3 卷积核(384 个) stride=1 padding=1 ReLU	13×13×256	13×13×384
卷积层 4 (Conv4)	3×3 卷积核(384 个) stride=1 padding=1 ReLU	13×13×384	13×13×384
卷积层 5 (Conv5)	3×3 卷积核(256 个) stride=1 padding=1 ReLU	13×13×384	13×13×256
池化	3×3 stride=2 padding=0	13×13×256	6×6×256
全连接层 1 (FC6)	ReLU	9216	4096
全连接层 2 (FC7)		4096	4096
全连接层 3 (FC8)		4096	分类数

另外，通过特征可视化可以知道，克里热夫斯基(Krizhevsky)的 CNN 结构学习到的第一层特征只对于高频和低频信息收敛，对于中层信息还没有收敛；同时，第二层特征出现了混叠失真，主要是第一个卷积层的步长设置为 4 引起的，为了解决这个问题，不仅将第一层的卷积核的大小设置为 7×7，同时也将步长设置为 2。

(2) ZFNet 网络将第三、四和五层卷积层中卷积核的数量后期进行调整，分别由 384、384、256 调整为 512、1024、512。

卷积神经网络的结构要素，如卷积核尺寸大小、池化步长、卷积核数量选择等方面都是事先人为设定的。这种设定虽然有一定的先验信息或知识作参考，但仍然带有一定的盲目性，因此需要通过多次试验寻求最佳的设计。ZFNet 网络通过可视化网络过程，为卷积神经网络结构要素的选择提供了一个分析方法或依据，虽然还不能确定具体的参数设置，但给出了大致的优化方向。这种方法对于结构要素选择是非常有意义的一种探讨思路。

5.3.4 VGGNet 网络

VGGNet 模型是牛津大学计算机视觉组和谷歌 DeepMind 公司的研究员一起研发的深度卷积神经网络，随后该网络以牛津大学视觉几何组(Visual Geometry Group)的缩写来命名[11]。该模型参加 2014 年的 ImageNet 图像分类与定位挑战赛(ILSVRC 2014)，获得了分类任务上排名第二、定位任务排名第一的优异成绩。

1. VGGNet 网络介绍

该模型的突出贡献在于证明使用尺寸较小的卷积核(如 3×3)，同时增加网络深度有助于提升模型的效果。VGGNet 模型对其他数据集具有很好的泛化能力，到目前为止，VGGNet 依然经常用来提取图像特征。表 5-4 给出了 6 种 VGGNet 模型参数配置。这 6 种参数配置中，比较著名的有 VGG16 和 VGG19，分别指的是 D 和 E 两种配置。表中的黑体部分代表当前配置相对于前面一种配置增加的部分。例如，A-LRN 相对于 A 网络增加了 LRN；而 B 相对于 A 而言增加了 2 个卷积层，均表示为 Conv3-64，Conv3-64 是指采用了含有 64 个 3×3 卷积核的卷积层。为了便于表示，VGGNet 模型中构造了卷积层堆叠块，例如由若干个卷积层堆叠后进行最大池化后构成的结构，可以采用 2 层卷积层、3 个卷积层甚至是 4 个卷积层堆叠后池化。

2. VGG16 典型结构分析

VGGNet 成功探讨了卷积神经网络的深度与其性能之间的关系，通过反复堆叠 3×3 的小型卷积核和 2×2 的最大池化层，VGGNet 成功地构筑了 16～19 层深的卷积神经网络。VGGNet 相比之前主流网络结构，错误率大幅下降。下面以经典的 VGG16 网络结构进行分析。图 5-19 给出了配置 D 情况下 VGG16 的网络结构简图，共包括 13 个卷积层和 3 个全连接层，其中卷积层、池化层和全连接层分别如图中实线、虚线和实线阴影所示，每个卷积堆叠块最后都是池化层。表 5-5 给出了配置 D 情况下 VGG16 模型的主要模型参数。从表中可以看出，采用了 5 段卷积层堆叠第一和第二卷积层段各包括了 2 个卷积层，而第三至五卷积层段各包括了 3 个卷积层，所有的卷积层都采用了 3×3 尺寸的卷积核，卷积核步长为 1；每个卷积层段后跟一个最大池化层，池化尺寸为 2×2，步长为 2，所以池化后图像平面尺寸减半，而通道数保持不变。

表 5-4　VGGNet 模型参数配置

卷积网络配置						
A	A-LRN	B	C	D	E	
11 层	11 层	13 层	16 层	16 层	19 层	
输入图像(224×224 RGB 图像)						
Conv3-64	Conv3-64 **LRN**	Conv3-64 **Conv3-64**	Conv3-64 Conv3-64	Conv3-64 Conv3-64	Conv3-64 Conv3-64	Block 1
最大池化层(Max-pooling)						
Conv3-128	Conv3-128	Conv3-128 **Conv3-128**	Conv3-128 Conv3-128	Conv3-128 Conv3-128	Conv3-128 Conv3-128	Block 2
最大池化层(Max-pooling)						
Conv3-256 Conv3-256	Conv3-256 Conv3-256	Conv3-256 Conv3-256	Conv3-256 Conv3-256 **Conv1-256**	Conv3-256 Conv3-256 **Conv3-256**	Conv3-256 Conv3-256 Conv3-256 **Conv3-256**	Block 3
最大池化层(Max-pooling)						
Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 **Conv1-512**	Conv3-512 Conv3-512 **Conv3-512**	Conv3-512 Conv3-512 Conv3-512 **Conv3-512**	Block 4
最大池化层(Max-pooling)						
Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 **Conv3-512**	Conv3-512 Conv3-512 **Conv3-512**	Conv3-512 Conv3-512 Conv3-512 **Conv3-512**	Block 5
最大池化层(Max-pooling)						
全连接层(FC-4096)						
全连接层(FC-4096)						
全连接层(FC-1000)						
Softmax 层						

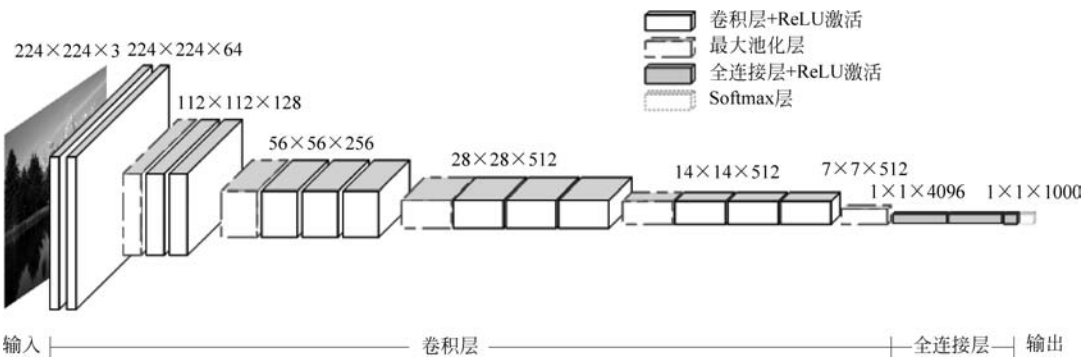


图 5-19　VGG16 网络结构简图

表 5-5　VGG16 模型参数配置

类　　型	核尺寸/步长	输入特征图尺寸	输出特征图尺寸
第一段卷积层	3×3/1	224×224×3	224×224×64
	3×3/1	224×224×64	224×224×64
Max-Pooling	2×2/2	224×224×64	112×112×64
第二段卷积层	3×3/1	112×112×64	112×112×128
	3×3/1	112×112×128	112×112×128
Max-Pooling	2×2/2	112×112×128	56×56×128
第三段卷积层	3×3/1	56×56×128	56×56×256
	3×3/1	56×56×256	56×56×256
	3×3/1	56×56×256	56×56×256
Max-Pooling	2×2/2	56×56×256	28×28×256
第四段卷积层	3×3/1	28×28×256	28×28×512
	3×3/1	28×28×512	28×28×512
	3×3/1	28×28×512	28×28×512
Max-Pooling	2×2/2	14×14×512	14×14×512
第五段卷积层	3×3/1	14×14×512	14×14×512
	3×3/1	14×14×512	14×14×512
	3×3/1	14×14×512	14×14×512
Max-Pooling	2×2/2	14×14×512	7×7×512
全连接层 1(FC1)	ReLU	25088	4096
全连接层 2(FC2)		4096	4096
全连接层 3(FC3)		4096	1000

另外，从配置表可以看出，特征图经过卷积层后，尺寸没有发生变化(这是因为采用了补零策略，由于卷积核尺寸为 3×3，卷积后尺寸不发生变换，因而补零个数为 1)，每个卷积层输出特征图的通道数就是每层卷积核的个数。

此外，VGGNet 的另外一种经典配置就是配置 E 情况下的 VGG19。VGG19 包括 16 个卷积层和 3 个全连接层。由于结构非常类似，不再做相应分析，可参考 VGG16。

3. VGGNet 特点分析

从前面的卷积神经网络分析很容易看清 VGG16 的网络结构，下面针对 VGG16 特点进行详细分析。

1) 卷积层堆叠式结构

为了了解卷积层堆叠式结构的产生原因，下面通过一个简单例子进行分析。假设输入图像平面尺寸为 $m\times n$，卷积核的步长为 1，那么采用 5×5 的卷积核做一轮卷积运算，则输出特征图的平面尺寸为$(m-5+1)\times(n-5+1)$，即$(m-4)\times(n-4)$。同理，利用 3×3 的卷积核对上述相同的图像进行卷积，可以得到输出特征图尺寸为$(m-3+1)\times(n-3+1)$，即$(m-2)\times(n-2)$。如果采用两层堆叠式结构，则对平面尺寸为$(m-2)\times(n-2)$图像再采用 3×3 的卷积核进行卷积，则此时输出特征图尺寸为$(m-4)\times(n-$

4)。由上述例子可知,利用步长为1、尺寸为3×3的卷积核构成的卷积堆叠层,可以实现与5×5的卷积核的单卷积层同样的效果,即实现了5×5的感受野。同理,三层3×3卷积层堆叠可以实现7×7的感受野。

因此,VGGNet网络相比于ImageNet的冠军网络AlexNet和ZFNet而言,有两个优势:

(1) 包含三个ReLU卷积层而不是一个,使得决策函数更有判别性;

(2) 卷积层堆叠式结构相比单层卷积层而言,同样感受野的条件下但堆叠式模型参数更少,比如输入与输出都是L个通道,使用平面尺寸为3×3的3个卷积层需要$3\times(3\times3\times L)\times L=27\times L^2$个参数,使用7×7的1个卷积层需要$(7\times7\times L)\times L=49\times L^2$个参数,这可以看作是为7×7的卷积施加一种正则化,使它分解为3个3×3的卷积。

2) 1×1卷积核的运用

从表5-4可以看出,在配置C的网络中,出现了1×1卷积核。使用1×1的卷积层,有两个作用:一是改变通道数;二是若输入通道与输出通道数相同,为了实现线性变换,线性变换后也可增加决策函数的非线性,而不影响卷积层的感受野,虽然1×1的卷积操作是线性的,但是ReLU增加了非线性。

5.3.5 GoogLeNet网络

GoogLeNet是2014年谷歌公司的克里斯蒂安·塞格迪(Christian Szegedy)提出的一种全新的深度学习网络结构[12],该网络在2014年ImageNet的大规模视觉识别挑战赛(ILSVRC14)的分类和检测上取得了最好结果,将错误率降到6.7%。GoogLeNet和VGGNet被誉为2014年ILSVRC14的"双雄",在分类任务和检测任务上,GoogLeNet获得了第一名,VGGNet获得了第二名;而在定位任务上,VGGNet获得第一名。

该网络名字为GoogLeNet而不是GoogleNet,主要原因:该名称命名由两部分组成,一是该网络的提出者克里斯蒂安·塞格迪在谷歌工作,二是研究者为了表达对1989年杨立昆所提出的LeNet网络的致敬,将两部分组合起来命名为GoogLeNet,既体现了谷歌公司的贡献,也表达了对已有重要学者和重要贡献的崇敬。从技术角度而言,该网络模型的核心内容是发明了深度感知模块(Inception Module,IM),因此GoogLeNet还有另外一个名字叫InceptionNet,"Inception"一词蕴含着"深度感知"。

GoogLeNet和VGGNet模型结构的共同特点是层次更深。VGGNet继承了LeNet以及AlexNet的一些框架结构,AlexNet、VGGNet等结构都是通过增加网络的深度(层数)来获得更好的任务精度;但层数的增加会出现过拟合、梯度消失、梯度爆炸等问题。而GoogLeNet则做了更加大胆的网络结构尝试,虽然深度只有22层,但模型参数规模比AlexNet和VGGNet小很多,GoogLeNet参数为500万个,AlexNet参数个数是GoogLeNet的12倍,VGGNet参数是AlexNet的3倍。因此,在内存或计算资源有限时,GoogLeNet是比较好的选择;从模型结果来看,GoogLeNet在较少的参数规模下能获得更加优越的性能,其根本原因在于深度感知模块(Inception Module,IM)的提出。深

度感知模块也称为深度感知结构(Inception Architecture,IA),需要说明的是,深度感知模块是本书编者从自身理解角度给出的名称,并不具有通用性,为了阐述方便,后续直接以“Inception 模块”进行表述。

1. Inception 模块提出缘由

通常而言,提高深度神经网络性能最直接的方法是增加网络的尺寸,包括增加网络的深度(网络层数)和网络宽度(每一层的单元数目)两种方式。特别是在可获得大量标记的训练数据的情况下,这种方式是训练高质量模型的一种简单而安全的方法。但是,这个简单的解决方案有三个主要的缺点:一是更大的尺寸意味着更多的参数,参数规模的增加使得网络容易陷入过拟合,特别是在训练集标注样本有限的情况下,而通常情况下标注数据又难以迅速获取。二是当层数增加时,使得网络优化更为困难,更容易出现梯度消失和梯度爆炸问题。三是增加网络尺寸需要更多的计算资源。例如,在一个深度卷积神经网络中,若两个卷积层相连,它们的滤波器数目的任何均匀增加都会导致计算的平方式增加。如果增加容量,则使用率低下(如果大多数权重结束时接近于 0),会浪费大量的计算能力。由于在实际中的计算预算总是有限的,因此计算资源的有效分布更偏向于尺寸无差别的增加。

解决上述不足的方法是引入稀疏特性和将全连接层转换成稀疏连接。这个思路来源于两方面:一是生物的神经系统连接是稀疏的;二是有文献指出,如果数据集的概率分布能够被大型且非常稀疏的 DNN 所描述,那么通过分析前面层的激活值的相关统计特性,以及将输出高度相关的神经元进行聚类,便可逐层构建出最优的网络拓扑结构。这种方式说明,复杂冗余的网络可以在不降低性能情况下进行精简。但是,现在的计算框架对非均匀稀疏数据的计算非常低效,主要是因为查找和缓存的开销。因此,GoogLeNet 作者提出了一个思想,既能保持滤波器级别的稀疏特性,又能充分利用密集矩阵的高计算性能。有大量文献指出,将稀疏矩阵聚类成相对密集的子矩阵,能提高计算性能。上述观点便是 Inception 模块的想法来源。

2. Inception 模块结构分析

传统单纯依靠扩大网络规模或者增大训练数据集是迫不得已的解决方法,从本质上提高网络性能,需要引入新型稀疏连接结构,即采用“小”且“分散”的可重复性堆叠模块,构成深度感知网络来完成复杂任务学习。Inception 模块是 GoogLeNet 的核心组成单元,其结构如图 5-20 所示[12]。

如图 5-20(a)所示,原始 Inception 模块基本组成结构有 1×1 卷积、3×3 卷积、5×5 卷积和 3×3 最大池化 4 个部分,最后对 4 个成分运算结果进行通道上组合。Inception 模块的核心思想是通过多个卷积核提取图像不同尺度的信息,最后进行融合,这样可以得到图像更好的表征。Inception 模块的特点如下:

(1) 多感受野感知:采用不同大小的卷积核意味着不同大小的感受野。

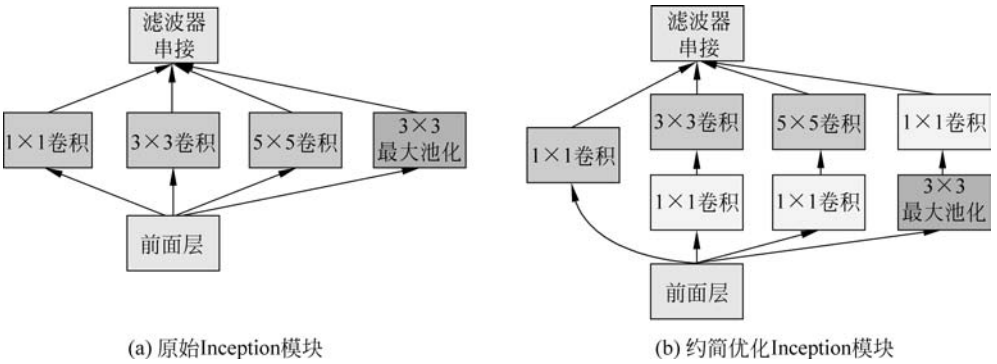


图 5-20　Inception 模块

(2) 池化模块：池化作用比较明显，在 Inception 模块专门嵌入了最大池化。

(3) 特征拼接融合：不同大小的感受野需要进行融合，才能得到更好表示。为了方便对齐，采用尺寸分别为 1×1、3×3 和 5×5 的卷积核，得到相同维度的特征进行拼接。设定步长 stride=1 后，只需要分别设定补零值为 0、1、2，就能得到相同的输出特征图平面尺寸。例如，假设输入特征图平面尺寸为 $m\times n$，利用 1×1 滤波器卷积后，其平面尺寸分别为 $(m-1+0+1)\times(n-1+0+1)$，即 $m\times n$；利用 3×3 滤波器卷积后，其平面尺寸分别为 $(m-3+2\times1+1)\times(n-3+2\times1+1)$，即 $m\times n$；利用 5×5 滤波器卷积后，其平面尺寸分别为 $(m-5+2\times2+1)\times(n-5+2\times2+1)$，即 $m\times n$。因此，得到的输出特征图平面尺寸都是 $m\times n$，只是通道数不同，可以方便拼接。Inception 模块的特征融合如图 5-21 所示，经过设置，前面层的输出特征图经过多尺度卷积后，得出融合后的输出特征图。

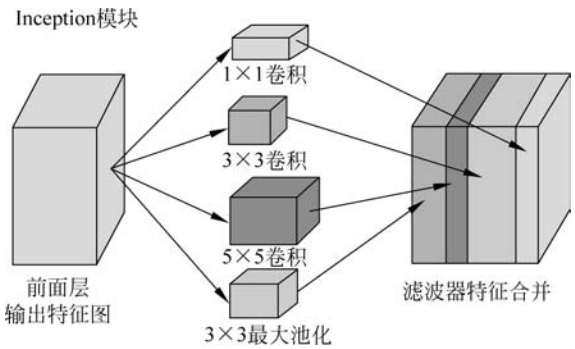


图 5-21　Inception 模块特征融合

可以看出，Inception 模块的提出，使得深层卷积神经网络设定时，不需要人工确定卷积层中的过滤器类型或是否需要创建卷积层和池化层，而是由网络自行决定这些参数，可以给网络添加所有可能值，将输出连接起来，网络自己决定需要学习什么样的参数。

由原始 Inception 模块堆叠起来的网络也存在缺点。由于 Inception 模块是逐层栈式

堆叠的，故输出的关联性统计会产生变化，即更高层抽象的特征会由更高层次捕获，而它们的空间聚集度会随之降低(因为随着层次的升高，3×3 和 5×5 的卷积的比例也会随之升高)。由于所有的卷积核都紧接着上一层的输出，而前面特征图是合并得到的，又有 5×5 卷积核的通道数与前一层特征图通道数相同，因此计算量很大。

(4) 1×1 卷积的特殊运用：图 5-20(b)展示了针对上述缺点改进的通道约简后的优化 Inception 模块，它在实际中被广泛使用。对比图 5-20(a)和(b)可以看出，相对于原始模块，约简优化 Inception 模块在每个操作后都增加了一个 1×1 的卷积操作。主要有两点作用：一是对数据进行降维，减少整体参数量；二是由于 1×1 卷积线性变换后要经过 ReLU 函数，因此引入更多的非线性，提高模型的泛化能力。下面通过一个简单例子来分析 1×1 卷积操作数据维度约简和减少参数的作用。例如，上一层的输出为 100×100×128，经过具有 256 个输出的 5×5 卷积层之后(stride＝1，pad＝2)，输出数据的大小为 100×100×256。其中，卷积层的参数量为 5×5×128×256。假如上一层输出先经过具有 32 个输出的 1×1 卷积层，再经过一个 256 个 5×5 核的卷积层，那么最终的输出数据的大小仍为 100×100×256，但采用 1×1 的卷积的参数量为 1×1×128×32＋5×5×32×256。对比而言，使用 1×1 卷积后参数量大致为原来的 1/4。

3. GoogLeNet 网络结构分析

图 5-22 给出了 GoogLeNet 网络结构。GoogLeNet 网络有 22 层，包括 9 个 Inception 模块的堆叠，最后一层使用了 2014 年新加坡国立大学林敏等提出的网中网模型(Network In Network，NIN)中的全局平均池化层，加上全连接层后再输入到 Softmax 函数中。其中，卷积和池化模块下面的数字(如 3×3＋1(S))分别代表卷积核尺寸或池化尺寸、步长，“S”代表特征图大小不变，而“V”代表有效卷积或有效池化。

表 5-6 给出了 GoogLeNet 网络的具体参数配置。从表中可以看出，对于每个 Inception 模块构成的变换层，其输出通道个数是 1×1、3×3 和 5×5 滤波器个数总和加上池化投影个数，即

$$\text{输出通道个数} = C_{1\times1} + C_{3\times3} + C_{5\times5} + \text{池化投影个数}$$

如表 5-6 中黑体所示，其中 $C_{1\times1}$、$C_{3\times3}$ 和 $C_{5\times5}$ 分别为不同尺寸的滤波器个数。

实验证明，引入平均池化层提高了准确率，而加入全连接层更便于后期的微调。此外，GoogLeNet 依然使用 Dropout 技术来防止过拟合。

GoogLeNet 增加了两个辅助的 Softmax 分支，如图 5-22 中最右侧输出部分。这两个 Softmax 分支有两个作用：一是由于 GoogLeNet 层数很多，为了避免梯度消失导致模型难以训练，在中间位置增加分支，用于向前传导梯度(反向传播时如果有一层求导为 0，链式求导结果则为 0)；二是将中间某一层输出用作分类，起到模型融合作用，最后的总损失函数是三个损失函数的加权和，但这只用在模型训练过程中。在实际测试时，这两个辅助 Softmax 分支会被去掉。

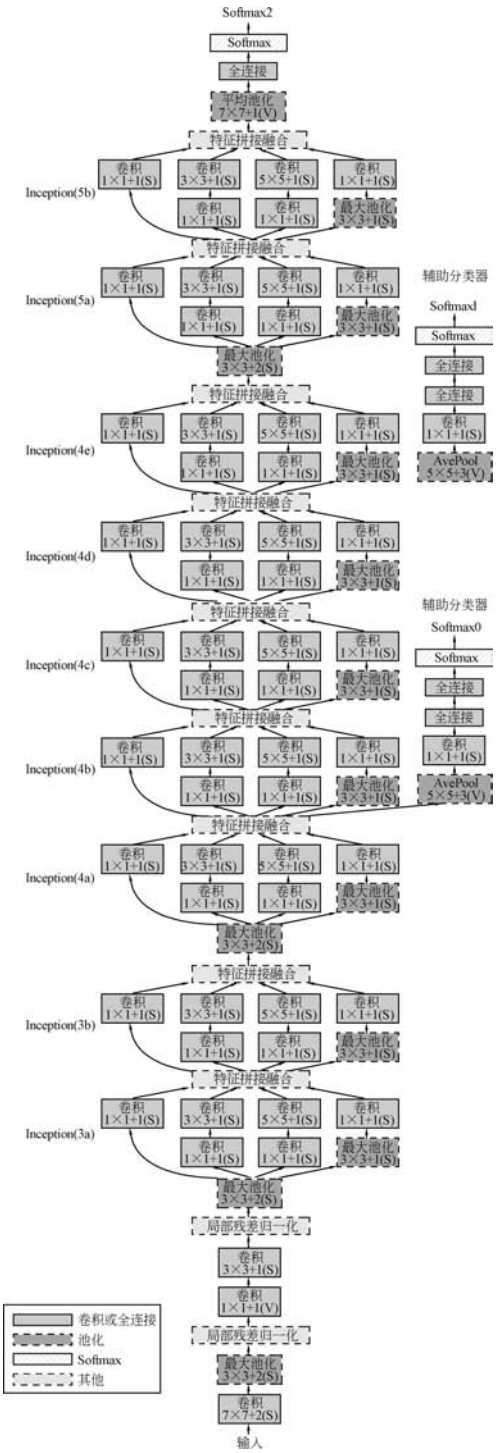


图 5-22　GoogLeNet 网络结构

表 5-6　GoogLeNet 网络参数配置

类型	尺寸/步长	输出特征图	深度	1×1	3×3投影	3×3	5×5投影	5×5	池化投影	参数	运算
卷积	7×7/2	112×112×64	1							2.7K	34M
最大池化	3×3/2	56×56×64	0								
卷积	3×3/1	56×56×192	2		64	**192**				112K	360M
最大池化	3×3/2	28×28×192	0								
Inception(3a)		28×28×**256**	2	**64**	96	**128**	16	**32**	**32**	159K	128M
Inception(3b)		28×28×**480**	2	**128**	128	**192**	32	**96**	**64**	380	304M
最大池化	3×3/2	14×14×480	0								
Inception(4a)		14×14×**512**	2	**192**	96	**208**	16	**48**	**64**	364K	73M
Inception(4b)		14×14×**512**	2	**160**	112	**224**	24	**64**	**64**	437K	88M
Inception(4c)		14×14×**512**	2	**128**	128	**256**	24	**64**	**64**	463K	100M
Inception(4d)		14×14×**528**	2	**112**	144	**288**	32	**64**	**64**	580K	119M
Inception(4e)		14×14×**832**	2	**256**	160	**320**	32	**128**	**128**	840K	170M
最大池化	3×3/2	7×7×832	0								
Inception(5a)		7×7×**832**	2	**256**	160	**320**	32	**128**	**128**	1072K	54M
Inception(5b)		7×7×**1024**	2	**384**	192	**384**	48	**128**	**128**	1388K	71M
平均池化	7×7/1	1×1×1024	0								
Dropout(40%)		1×1×1024	0								
线性		1×1×1000	1							1000K	1M
Softmax		1×1×1000	0								

4. Inception 模块的改进版本 Inception-v2 和 Inception-v3

GoogLeNet 网络中提出的 Inception 模块后来被称为 Inception version1，简称 Inception-v1，后期很多学者也不断提出 Inception 模块的改进版本，包括 Inception-v2、Inception-v3 和 Inception-v4。

1）设计原则

Inception-v2 和 Inception-v3 出现在同一篇论文中[16]，在了解 Inception-v2 的结构之前，首先来深入了解 Inception-v2 设计的思想和初衷，总结起来是 4 个原则。

（1）原则一：慎用信息压缩，避免信息表征性瓶颈。

直观上来说，当卷积不会大幅度改变输入维度时，神经网络可能会执行得更好。过多地减少维度可能会造成信息的损失，这也称为“表征性瓶颈”。为了提高模型分类精度，从输入层到输出层特征维度应缓慢下降，尽量避免使用信息压缩的瓶颈模块，尤其是不应当在模型的浅层使用。从图结构角度来看，CNN 模型本质上是一个有向无环图（Directed Acyclic Graph，DAG），其信息自底向上流动，而每一个瓶颈模块的使用都会损失一部分信息，因此当出于计算与存储节省而使用瓶颈模块时，一定要慎重。前期的信息到后面不可恢复，因此尽量不要过度使用信息压缩模块，如不要用 1×1 的卷积核骤降输出特征图的通道数目，尽量在模型靠后的几层使用。

(2) 原则二：网络局部处理中的高维度特征表达。

高维度特征表达在网络的局部中处理起来更加容易。暂不考虑计算与内存开销增加的负面因素，增加每层卷积核数量可以增强其表达能力。简单理解，可以认为每个卷积核都具有发现不同模式的能力，因此增加卷积核个数，能够从不同角度去描述输入信号，增强其表达能力。另外，增加卷积神经网络每个神经元的激活值会更多地解耦合特征，使得网络训练更快。网络训练快是指整体所需要的迭代次数少，而不是整体训练所需的时间。

(3) 原则三：低维嵌入上的空间聚合。

可以在较低维的嵌入上进行空间聚合，而不会损失很多表示能力。例如，在进行更分散(如 3×3)的卷积之前，可以减小输入表示的尺寸，而不会出现严重的不利影响。之所以这样做，原因在于低维空间降维期间相邻单元之间相关性很强，降维丢失的信息较少。鉴于这些信号应易于压缩，因此降维可以促进更快地学习。

(4) 原则四：平衡网络的深度与宽度。

谷歌公司的研究人员将深度学习网络的设计问题视为一个在计算/内存资源限定情况下的结构优化问题，即通过有效组合、堆加各种层/模块，最终使得模型分类精度最高。后期大量实验表明，CNN 设计一定要将深度与宽度相匹配，换句话说“瘦高”或“矮胖”的 CNN 不如深宽匹配度高的网络。

2) Inception-v2 和 Inception-v3 核心技术

(1) 批归一化技术。

批归一化[17](Batch Normalization，BN)技术在用于神经网络某层时，会对每一个 minibatch 数据的内部进行标准化处理，使输出规范化到 N(0，1)的正态分布，减少了内部神经元分布的改变。BN 技术提出后得到广泛应用。相关文献指出，传统的深度神经网络在训练时，每一层输入的分布都在变化，导致训练变得困难，因此在使用 BN 之前，只能使用一个很小的学习速率解决这个问题。而对每一层使用 BN 后，就可以有效地解决这个问题，学习速率可以增大很多倍，达到之前的准确率所需要的迭代次数减少为原来的 1/10 甚至更少，训练时间大大缩短。因为 BN 某种意义上还起到了正则化的作用，所以可以减少或者取消 Dropout，简化网络结构。

(2) 基于大滤波器尺寸分解卷积。

可以将大尺度的卷积分解成多个小尺度的卷积来减少计算量。比如，将 1 个 5×5 的卷积分解成两个 3×3 的卷积串联，如图 5-23 所示。假设 5×5 和两级 3×3 卷积输出的特征数相同，那两级 3×3 卷积的计算量就是前者的(3×3＋3×3)/5×5＝18/25。为了便于表示，将图 5-23(b)所示的 Inception 模块表示为 Inception A。

(3) 不对称卷积分解。

上面的卷积分解方案减小了参数数量，同时也减小了计算量。但随后也出现了两个困惑：第一，这种分解方案是否影响特征表达，是否会降低模型的特征表达能力？第二，如果这种方案的目的是因式分解计算中的线性部分，那么是否在第一个 3×3 层使用线性激活？为此，文献[16]进行了对照实验，一组采用两层 ReLU，另一组采用线性＋

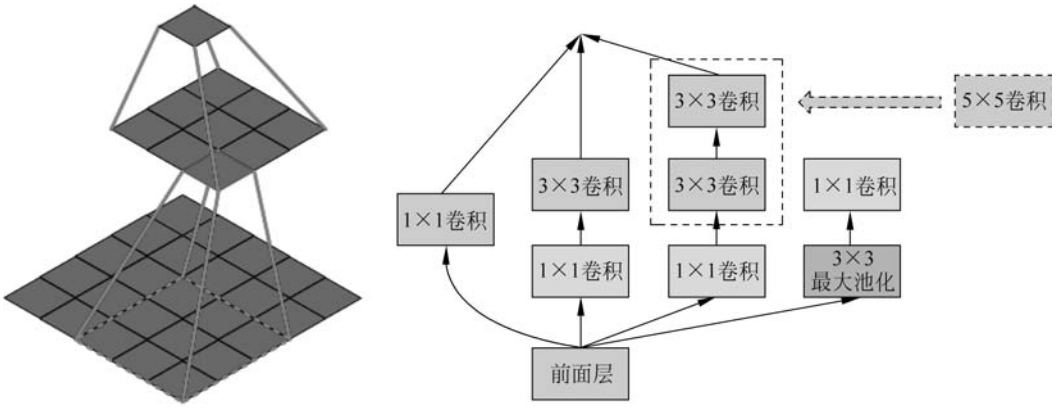


图 5-23　5×5 的卷积分解

ReLU。结果发现，线性＋ReLU 的效果总是低于两层 ReLU。产生这种差距的原因是多一层的非线性激活可以使网络学习特征映射到更复杂的空间，尤其是当对激活输出使用了批归一化技术。

既然大于 3×3 的卷积可以分解成 3×3 的卷积，是否继续考虑将其分解成更小的卷积核？当然很容易想到将 3×3 分解成两个 2×2 的卷积核，实际中没有采用这种对称分解，而是将 3×3 的卷积核分解成 1×3 和 3×1 卷积的串联，这种分解被称为非对称分解。原因在于，采用非对称分解后，能节省 33%的计算量，而将 3×3 卷积分解为两个 2×2 卷积仅节省了 11%的计算量。3×3 卷积分解如图 5-24 所示[18]。为了便于表示，将图 5-24(b)所示的 Inception 模块表示为 Inception B。

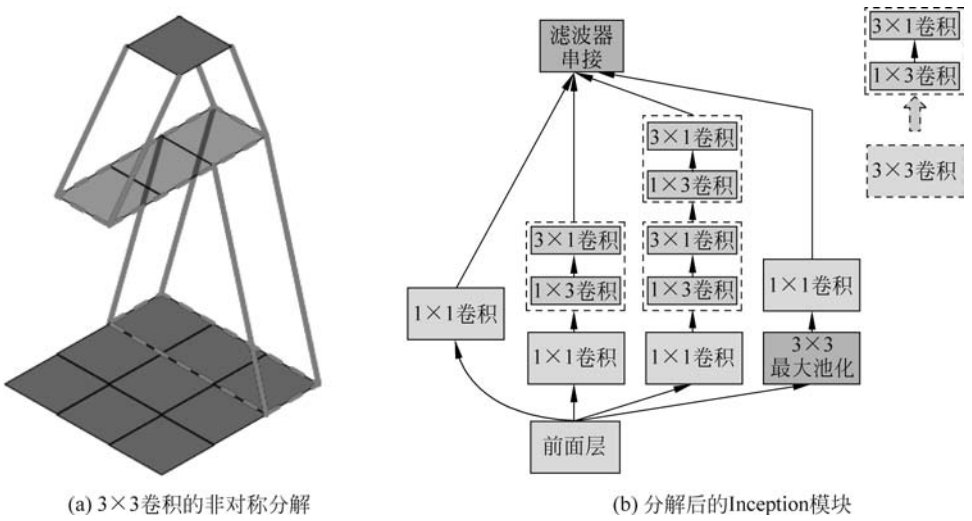


图 5-24　3×3 卷积分解

当然，在理论上可以更进一步，任意的 $n \times n$ 卷积都可以被 $1 \times n$ 加上 $n \times 1$ 卷积核来代替，而且随着 n 的增大，节省的参数和计算量将激增。研究表明，这种因式分解在网络的浅层似乎效果不佳，但是对于中等大小的特征图有着非常好的效果(12～20 的特征图)。

(4) 宽度扩展卷积分解。

为了便于表示，将图 5-25 所示的 Inception 模块表示为 Inception C。根据前面所述，按照前面三个原则用来构建三种不同类型的 Inception 模块，分别如图 5-23(b)、图 5-24(b)、图 5-25 所示，按引入顺序称为 Inception A、Inception B 和 Inception C，使用“A、B、C”作为名称只是为了叙述方便。

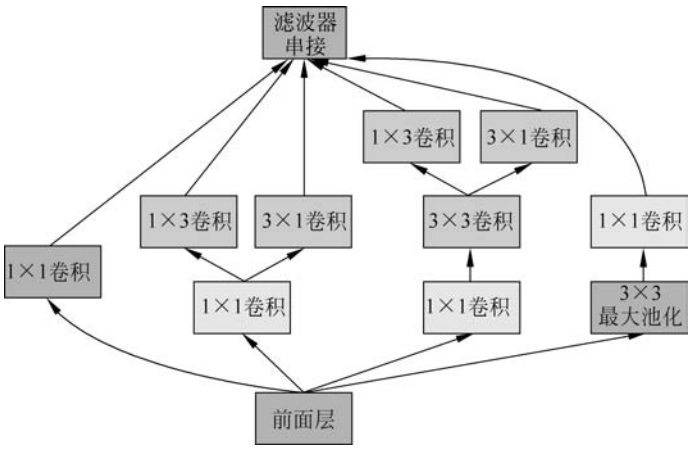


图 5-25　宽度扩展卷积分解

(5) 辅助分类技术。

在 Inception-v1 构建 GoogLeNet 网络中引入了辅助分类器的概念，以改善非常深层网络的收敛性。最初的想法是将有用的梯度推到较低的层，以使它们立即可用，并通过在非常深的网络中解决消失的梯度问题来提高训练期间的收敛性。也有其他学者进一步证实辅助分类器可以促进更稳定的学习和更好的收敛。然而，后期 Szegedy 等发现辅助分类器并未在训练初期改善收敛性：在两个模型都达到高精度之前，有无辅助分类器的网络训练进程似乎相同。在训练快要结束时，带有辅助分支的网络开始超越没有任何辅助分支的网络的精度，并达到略高的平稳期。

研究表明，在网络训练的不同阶段使用两个辅助分类项，但删除下部辅助分支不会对网络的最终质量产生任何不利影响。因此，Szegedy 改变了他们先前的看法，认为可以把辅助分类器充当正则化器。为此，他们在辅助分类损失项对应的全连接层中添加了批归一化和 Dropout，结果获得了一定的提升，这也为批处理归一化能充当正则化作用的想法提供了少量的证据支持。

(6) 网络约简技术。

传统的卷积网络会使用池化操作来减少特征图的大小。为了避免特征表达瓶颈，在进行池化之前都会扩大网络特征图的数量。例如，有 K 个 $d \times d$ 的特征图，如果变成 $2K$

个 $d/2\times d/2$ 的特征图，那么可以采用两种方式：一是先卷积再池化。先构建一个步长为 1、$2K$ 个卷积核的卷积层，再进行池化。这意味着总计算开销将由大特征图上进行的卷积运算所主导，约 $2d^2K^2$ 次操作。二是先池化后卷积。计算量就变为 $2(d/2)^2K^2$，计算量变为第一种方法的 1/4。但第二种方法会引入表达瓶颈问题，因为特征的整体维度由 d^2K 变成了 $(d/2)^2\times 2K$，这导致网络的表达能力减弱。图 5-26 给出了两种经典网络约简技术，其中 $d=35$，$K=320$。

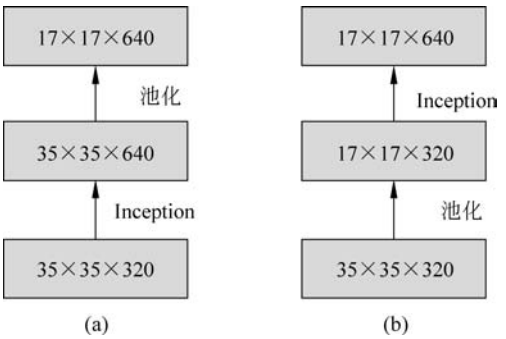


图 5-26　两种经典网络约简技术

为了克服表达瓶颈问题，Inception-v2 采用了一种更高效的数据压缩方式——网格约简技术。为了将特征图的大小压缩为 1/2 大小，同时通道数量变为 2 倍，采用了一种类似 Inception 的约简结构，同时做池化和卷积，步长为 2，再将两者结果堆叠起来，实现了特征图的压缩和通道的扩增。这种方法既能够减少计算开销，也能避免表达瓶颈。新型网络约简技术如图 5-27 所示。

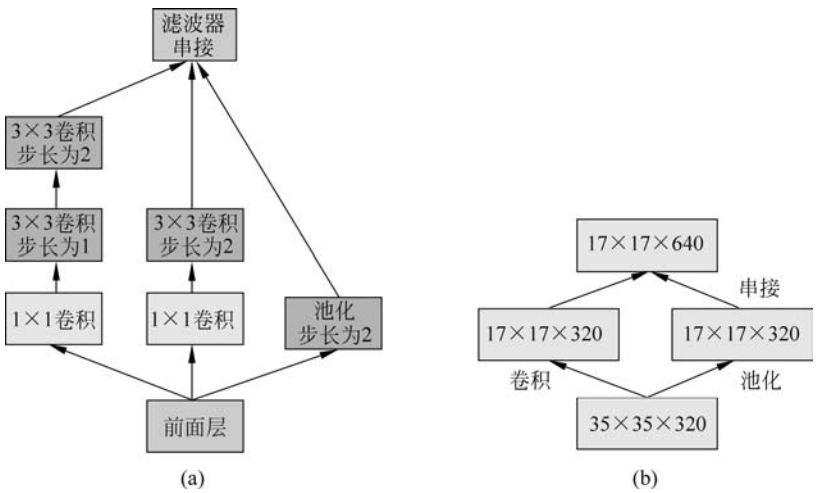


图 5-27　新型网络约简技术

(7) 标签平滑正则。

标签平滑正则(Label-smoothing Regularization，LSR)是一种通过估计标签丢弃的

边缘化效应来正则分类器层的机制。

假设 z_i 为未被归一化的对数概率，p 为样本的预测概率，q 为样本的真实类别标签概率。当分类采用 Top-1 进行表示时，样本的真实概率为狄拉克(Dirac)函数，即 $q(k)=\delta_{k,y}$，其中 y 为真实的类别标签。对于每个训练样本 x，网络模型计算其关于每个类别标签 $k\in\{1,2,\cdots,K\}$ 的概率值，则 Softmax 层输出的预测概率为

$$p(k\mid x)=\frac{\exp(z_k)}{\sum_{i=1}^{K}\exp(z_i)} \tag{5.7}$$

式中：z_i 为未归一化的对数概率值。

假设对该样本关于类别标签 $p(k|x)$ 的真实分布进行归一化，则有

$$\sum_{k}p(k\mid x)=1 \tag{5.8}$$

为简单起见，忽略关于样本 x 的 p 和 q 之间的依赖性。

定义样本的损失函数为交叉熵为

$$l=-\sum_{k=1}^{K}q(k)\log(p(k)) \tag{5.9}$$

最小化交叉熵函数等价于最大化特定类别标签的对数似然值，该特定类别标签是根据其真实分布 $q(k)$ 选定。由于交叉熵损失函数关于 z_i 是可微的，因此可以用于深度模型的梯度训练，其梯度的相对简洁形式为

$$\frac{\partial l}{\partial z_k}=p(k)-q(k),\quad \text{其值区间为}[-1,1] \tag{5.10}$$

假设只有单个真实类别标签 y 的情况，则 $q(y)=1$ 且 $q(k)=0(k\neq y)$ 时，最小化交叉熵损失函数等价于最大化正确类别标签的对数概率。

对于某个样本 x，其类别标签为 y，对 $q(k)$ 计算最大化对数概率，$q(k)=\delta_{k,y}$，即 $k=y$ 时，$\delta_{k,y}=1$，$k\neq y$ 时，$\delta_{k,y}=0$。

在采用预测的概率来拟合真实的概率时，只有当对应于真实类别标签的对数值远远大于其他类别标签的对数值时才可行。但其面临两个问题：

① 可能导致过拟合。如果模型学习的结果是对于每个训练样本都将全部概率值分配给真实类别标签，则不能保证其泛化能力。

② 其鼓励最大逻辑回归值和其他逻辑回归值间的差异尽可能大，但结合梯度 $\partial l/\partial z_k$ 的有界性，其削弱了模型的适应能力。

也就是说，只有模型对预测结果足够有信心时才可能发生的情况。

Inception-v3 提出了一种机制，鼓励模型少一点自信。虽然最大化训练标签对数似然度，但不是真正期望的目标，这样做能够正则化模型，并提升模型的适应能力。假设有独立于训练样本 x 的类别标签的分布 $u(k)$，对于真实类别标签为 y 的训练样本，将其类别标签的分布 $q(k|x)=\delta_{k,y}$ 替换为

$$q'(k\mid x)=(1-\varepsilon)\delta_{k,y}+\varepsilon u(k) \tag{5.11}$$

式中：ε 为平滑参数。可以看出 $q'(k|x)$ 是原始真实分布 $q(k|x)$、固定分布 $u(k)$ 和权重

ε 的组合。

将类别标签 k 的分布计算可以看作为两点：一是将类别标签设为真实类别标签，$k=y$；二是采用平滑参数 ε，将从分布 $u(k)$ 中的采样值来取代 k。

Inception-v3 采用类别标签的先验分布作为 $u(k)$，如均匀分布，$u(k)=1/K$，则有

$$q'(k \mid x)=(1-\varepsilon)\delta_{k,y}+\varepsilon\frac{1}{K} \tag{5.12}$$

故称为类别标签平滑正则化(Label-Smoothing Regularization，LSR)。LSR 交叉熵变为

$$H(q',p)=-\sum_{k=1}^{K}q'(k)\log(p(k))=(1-\varepsilon)H(q,p)+\varepsilon H(u,p) \tag{5.13}$$

等价于将单个交叉熵损失函数 $H(q,p)$ 替换为损失函数 $H(q,p)$ 和 $H(u,p)$ 的加权和。损失函数 $H(u,p)$ 惩罚了预测的类别标签分布 p 相对于先验分布 u 的偏差，根据相对权重 $\frac{\varepsilon}{1-\varepsilon}$。该偏差也可以从 KL 分歧的角度计算，因为 $H(u,p)=D_{\mathrm{KL}}(u \parallel p)+H(u)$，当 u 为均匀分布时，$H(u,p)$ 是评价预测的概率分布 p 与均匀分布 u 间的偏离程度。

(8) Inception-v2 网络组成。

表 5-7 给出了 Inception-v2 网络结构。由前面知识可知，普通卷积后，输出特征图平面尺寸变为(原平面尺寸-卷积核尺寸)/步长+1；而补零卷积不改变平面尺寸，只是将输入通道数变化变成卷积核个数，因此卷积核个数也就是新通道数。前面的 7 行是经典卷积层、池化层。前三行将传统的 7×7 卷积分解为三个 3×3 卷积串接，并且在第一个卷积处就通过步长为 2 来降低分辨率。而后面的 5～7 行卷积也一样，但分辨率降低时通过本组的第二个卷积来实现。对于网络的 Inception 部分，在 35×35 处有 3 个传统的 Inception 模块(图 5-23(b))，每个模块有 288 个过滤器。采用网格约简技术，将其缩减为具有 768 个滤波器的 17×17 网格。之后是如图 5-24(b)所示的 5 个因式分解 Inception 模块的实例，利用图 5-27 所示的网络约简技术将其缩减为 8×8×1280。在最粗糙的 8×8 级别，有两个图 5-25 所示 Inception 模块，每个图块的串联输出滤波器组大小为 2048。

表 5-7　Inception-v2 网络结构

行号	类　　型	输入图像	尺度/步长、滤波器数	输出图像
1	卷积	299×299×3	3×3/2、32 个	149×149×32
2	卷积	149×149×32	3×3/1、32 个	147×147×32
3	补零卷积	147×147×32	3×3/1、64 个	147×147×64
4	池化	147×147×64	3×3/2	73×73×64
5	卷积	73×73×64	3×3/1、80 个	71×71×80
6	卷积	71×71×80	3×3/2、192 个	35×35×192
7	补零卷积	35×35×192	3×3/1、288 个	35×35×288
8	3×Inception	35×35×288	如图 5-23(b)所示，288 个	17×17×768
9	5×Inception	17×17×768	如图 5-24(b)所示	8×8×1280
10	2×Inception	8×8×1280	如图 5-25 所示	8×8×2048
11	池化	8×8×2048	8×8	1×1×2048

续表

行号	类　　型	输入图像	尺度/步长、滤波器数	输出图像
12	线性	1×1×2048		1×1×1000
13	Softmax	1×1×1000	分类器	1×1×1000

模型结果与旧的 GoogLeNet 相比有较大提升，如表 5-8 所示。

表 5-8　Inception-v2 识别结果

网　　络	Top-1 误差/%	Top-5 误差/%	Bnops 代价
GoogLeNet	29	9.2%	1.5
BN-GoogLeNet	26.8	—	1.5
BN-Inception	25.2	7.8	2.0
Inception-v2	23.4	—	3.8
Inception-v2 RMSProp	23.1	6.3	3.8
Inception-v2 Label-Smoothing	22.8	6.1	3.8
Inception-v2 Factorized-7×7	21.6	5.8	4.8
Inception-v2 BN-auxiliary	21.2	5.6	4.8

(9) Inception-v3 网络组成。

Inception-v3 的结构是 Inception v2 版本的升级，除了上面的优化操作，还使用了四种技术：一是采用了 RMSProp 优化器，这是常用的一种神经网络优化算法，在第二章中有具体细节阐述；二是采用了非对称因式分解的 7×7 卷积，对于将 $n\times n$ 的卷积分解成 $n\times 1$ 和 $1\times n$ 卷积时，发现在网络的前层这样进行卷积分解起不到多大作用，不过在网络网格为 $m\times m$（m 在[12,20]之间）时结果较好，因此这里将 7×7 卷积直接分解成 1×7 和 7×1 的串联；三是辅助分类器使用了批归一化技术；四是将标签平滑技术用于辅助函数的正则化项，如前面技术细节所示，防止网络对某一类别过分自信，出现过拟合现象。

5.3.6　残差网络

1. 残差网络的提出缘由

残差网络(Residual Network，ResNet)于 2015 年提出[13]，在 ImageNet 比赛分类任务上获得第一名，因为它“简单与实用”并存，之后很多方法都建立在 ResNet50 或者 ResNet101 的基础上完成，检测、分割和识别等领域都纷纷使用 ResNet。而且由于其优越的性能，在 AlphaGo-Zero 的版本中也采用残差网络替代经典卷积神经网络。

实际研究过程中发现，随着网络深度的增加，会出现一种模型退化现象。模型退化就是当网络越来越深时，模型的训练准确率会趋于平缓，但是测试误差会变大，或者训练误差和测试误差都变大(这种情况不是过拟合)。换句话说，模型层数的增加并没有导致性能的提升，而是导致模型性能的下降，这种现象称为模型退化。例如，假设一个最优化的网络结构有 10 层。当设计网络结构时，由于预先不知道最优网络结构的层数，假设设计了一个 20 层网络，那么就会有 10 个冗余层，这对建模显然没有好处。很容易想到，能否让中间多余的层数变成恒等映射，也就是经过恒等映射层时输入与输出完全一样。但是，往往模型很难将这 10 层恒等映射的参数学习正确，因此出现退化现象，其核心原因在于冗余的网络层学习了非恒等映射的参数。

为此提出了残差网络。与传统深度学习利用多个堆叠层直接拟合期望特征映射的过程不同，残差网络显式地用多个堆叠层拟合一个残差映射。

2. 残差网络基本原理

残差网络的基本结构如图 5-28 所示，是带有跳跃的结构。可见图中有一个捷径连接或直连，这个直连实现恒等映射。假设期望的特征映射为 $H(\boldsymbol{x})$，而堆叠的非线性层拟合的是另一个量 $F(\boldsymbol{x})=H(\boldsymbol{x})-\boldsymbol{x}$，那么一般情况下最优化残差映射比最优化期望的映射更容易，也就是 $F(\boldsymbol{x})=H(\boldsymbol{x})-\boldsymbol{x}$ 比 $F(\boldsymbol{x})=H(\boldsymbol{x})$更容易优化。比如，极端情况下期望的映射要拟合的是恒等映射，此时残差网络的任务是拟合 $F(\boldsymbol{x})=0$，普通网络拟合 $F(\boldsymbol{x})=\boldsymbol{x}$，明显前者更容易优化。通过这种残差网络结构构造的网络层数可以很深，且最终的分类效果也非常好。

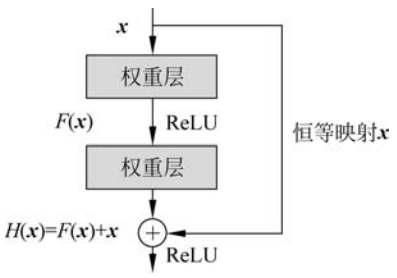


图 5-28　残差网络结构

图 5-29 给出了一个 34 层的残差网络结构图。经过直连后，$H(\boldsymbol{x})=F(\boldsymbol{x})+\boldsymbol{x}$，但图中出现了实线连接和虚线连接的情况。

(1) 实线连接表示通道相同，如图 5-29 中第一个方格底纹矩形和第三个方格底纹矩形，都是 3×3×64 的特征图，由于通道相同，所以采用计算方式为 $H(\boldsymbol{x})=F(\boldsymbol{x})+\boldsymbol{x}$。

(2) 虚线连接表示通道不同，如图 5-29 的第一个斜线底纹矩形和第三个斜线底纹矩形，分别是 3×3×64 和 3×3×128 的特征图，通道不同，采用的计算方式为 $H(\boldsymbol{x})=F(\boldsymbol{x})+\boldsymbol{W}\boldsymbol{x}$，其中 $\boldsymbol{W}$ 是卷积操作，用来调整 $\boldsymbol{x}$ 维度。

除了上面提到的两层残差学习单元，还有三层的残差学习单元，如图 5-30 所示。图中所示两种结构分别针对 ResNet34(图 5-30(a))和 ResNet50/101/152(图 5-30(b))，一般把图 5-30(a)和(b)所示的结构称为一个残差模块，其中图 5-30(a)是普通模块，图 5-30(b)是“瓶颈设计模块”，采用这种结构的目的就是降低参数数量。从图中可以看出，第一个 1×1 的卷积把 256 维的通道数降到 64 维，然后在 64 维通道数上进行卷积，最后再通过 1×1 卷积将通道数进行恢复，图 5-30(b)中用到参数为 1×1×256×64＋3×3×64×

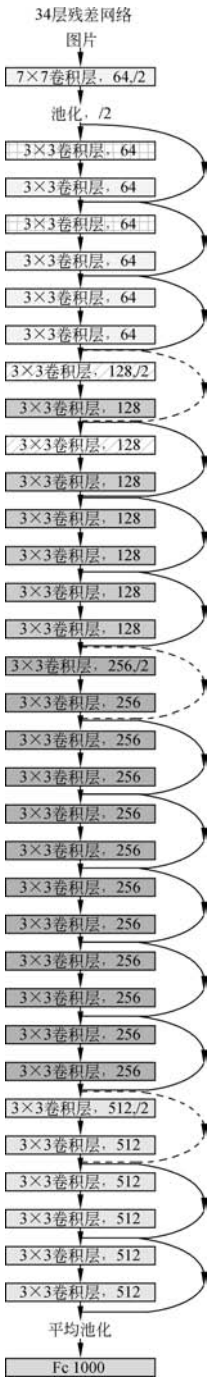


图 5-29　34 层残差网络结构

64＋1×1×64×256＝69632 个，而不使用“瓶颈”设计模块则是两个 3×3×256 的卷积，参数为 3×3×256×256×2＝1179648 个，是降维优化后参数的 16.94 倍。

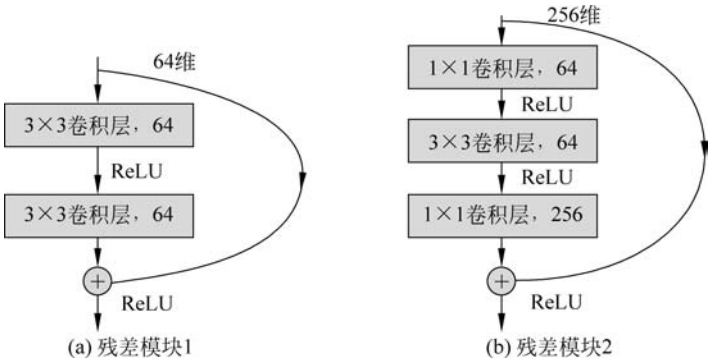


图 5-30　两种 Res 层设计

同样，在由 1000 个类组成的 ImageNet 2012 分类数据集上评估，模型在 128 万训练图像上进行训练，并对 50k 验证图像进行评估，其中分别采用了 18 层和 34 层普通网。图 5-31 给出了性能对比，图 5-31(a)是普通网络性能图，图 5-31(b)是残差网络性能图。从图中可以看出：采用普通网络，网络出现退化现象；采用残差网络，深层网络性能得到优化提升。

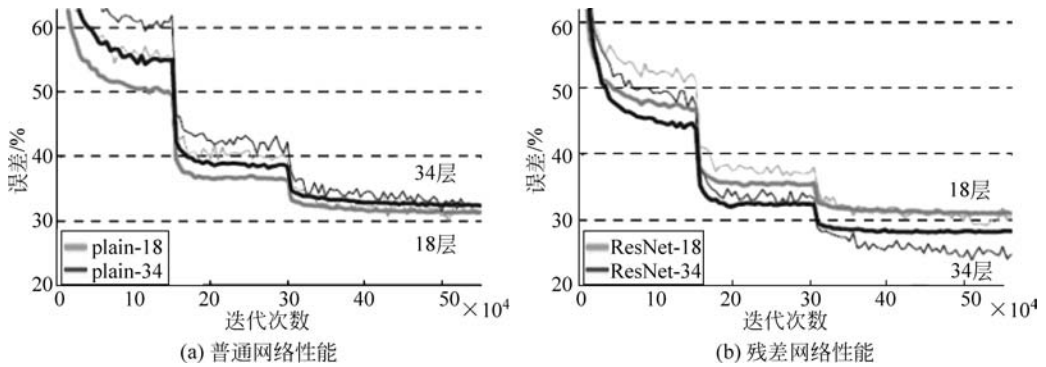


图 5-31　性能对比

表 5-9 列出了不同层数时模型参数配置，给出了不同 5 种残差网络的结构，深度分别是 18、34、50、101 和 152。首先都通过一个 7×7 的卷积层，接着是一个最大池化，之后就是堆叠残差块，其中 50、101、152 层的残差网络使用的残差块是“瓶颈”结构，各网络中残差块个数从左到右依次是 8、16、16、33、50。此外，在网络最后层通常连接一个全局平均池化，好处是使参数不需要最优化防止过拟合，对输入与输出的空间变换更具有鲁棒性，加强了特征映射与类别的一致性。

表 5-9　不同层数时模型参数配置

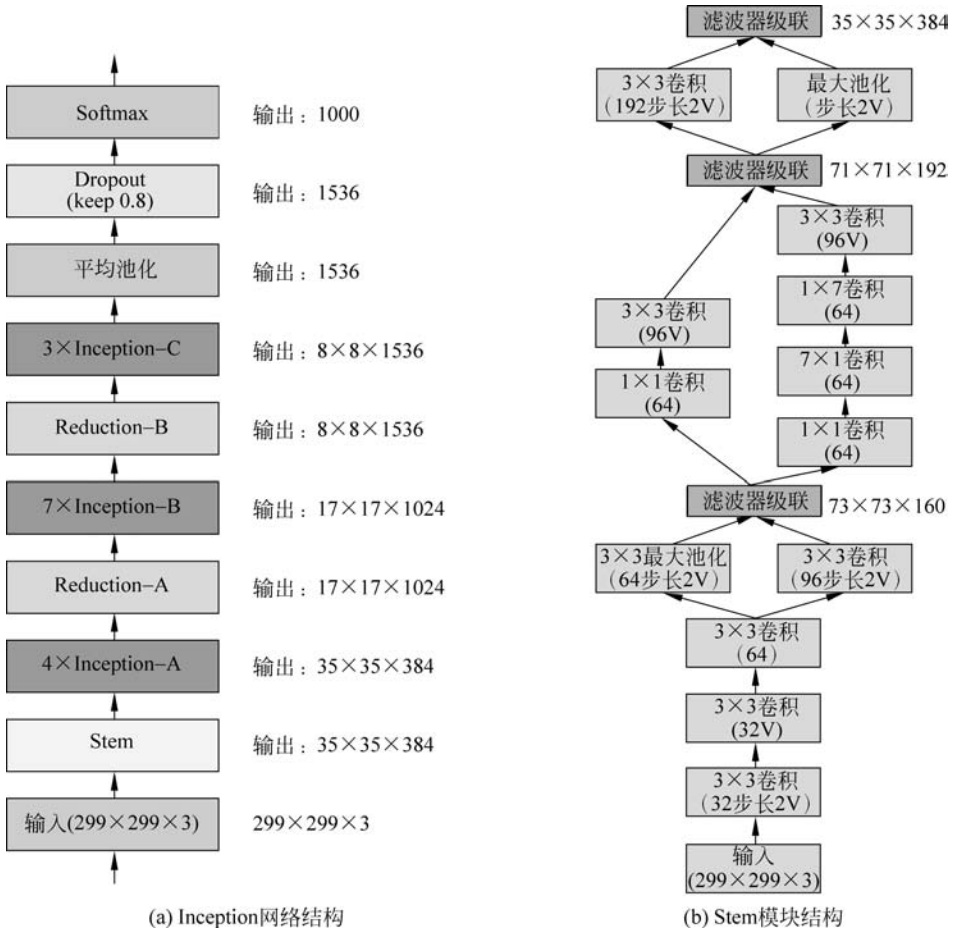
层　名	输出尺寸	18 层	34 层	50 层	101 层	152 层
卷积层 1	112×112	7×7,64,步长 2				
卷积层 2_x	56×56	3×3 最大池化,步长 2				
		$\begin{bmatrix}3\times3,64\\3\times3,64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,64\\3\times3,64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$
卷积层 3_x	28×28	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times8$
卷积层 4_x	14×14	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times36$
卷积层 5_x	7×7	$\begin{bmatrix}3\times3,512\\3\times3,512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,512\\3\times3,512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times3$
	1×1	平均池化,1000 维的全连接层,Softmax				
FLOP		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

3. Inception-v4

Inception-v4 网络中[19],对 Inception 块的每个网格大小进行了统一,结构如图 5-32 所示。图(a)为总结构图,图(b)为 Stem 模块的结构。其中:所有图中没有标记“V”的卷积使用相同尺寸填充原则,即其输出网格与输入的尺寸正好匹配;使用“V”标记的卷积使用有效卷积填充原则,即每个单元输入块全部包含在前几层中,同时输出激活图的网格尺寸也相应会减少。从图 5-32(b)分析可知,输入经过 32 个步长为 2 的有效卷积,则输出特征图尺寸变成 149×149×32,其中(299−3)/2+1=149;因此依次进行卷积运算,可以得到级联后的特征图为 73×73×160,其中通道数是 64+96=160;在第一个滤波器级联和第二个滤波器级联之间可以看到,左侧先后经过 1×1 和 3×3 有效卷积,尺寸变为 71×71×96,其中平面尺寸 71×71 由 73−3+1=71 得来,右侧经过 1×1、1×7、7×1 和 3×3 有效卷积,前三个卷积不改变平面尺寸,而 3×3 有效卷积同样使得平面尺寸变为 71×71,而通道数为 96,因此滤波器级联后的特征图为 71×71×192,经过卷积和池化后尺寸变为 35×35×384。

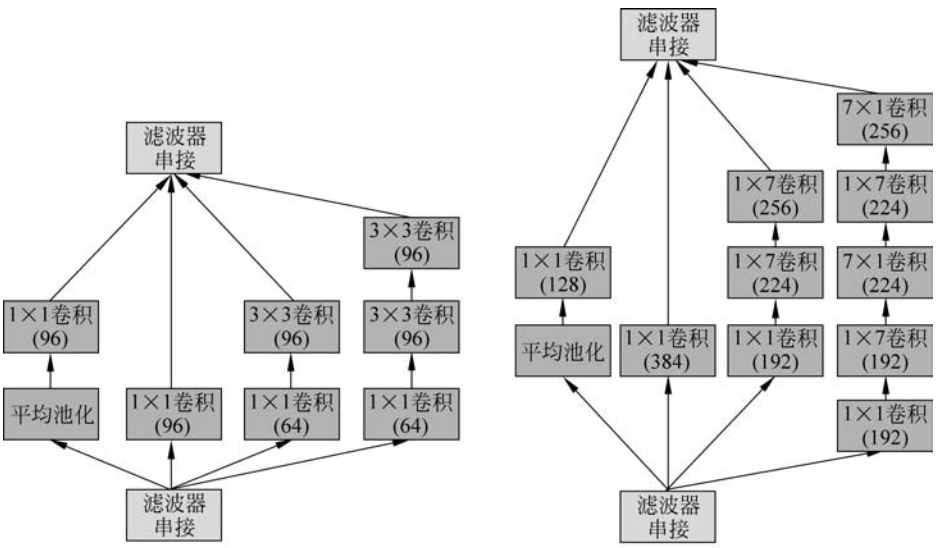
图 5-33 给出了 Inception-v4 网络中的模块结构,其中图(a)是 35×35 网格块框架(对应图中 Inception-A 块),图(b)是 Inception-v4 网络 17×17 网格块框架(对应图中 Inception-B 块);图(c)是 Inception-v4 网络 8×8 网格块框架(对应图中 Inception-C 块)。

图 5-34 给出了 Inception-v4 网络中的约简(Reduction)模块结构,其中图(a)将 35×35 网格约简到 17×17 的网格块,图(b)将 17×17 网格约简到 8×8 的网格块。需要注意的是,Reduction-A 模块是一种通用结构。由图中可以看出,k、l、m、n 分别为 1×1 卷积、



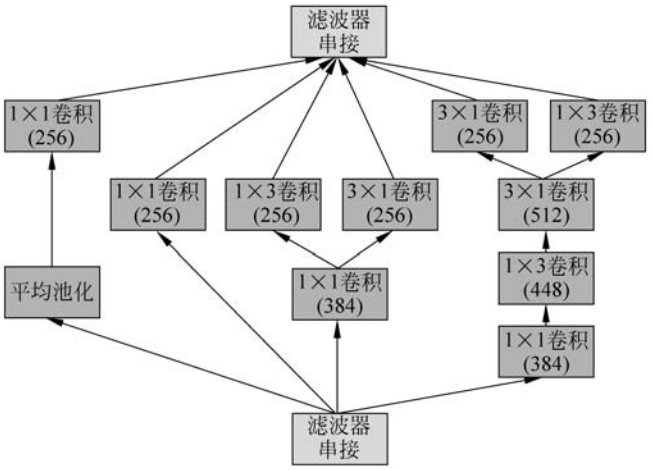
(a) Inception网络结构　　(b) Stem模块结构

图 5-32　Inception-v4 的结构



(a) Inception-A结构　　(b) Inception-B结构

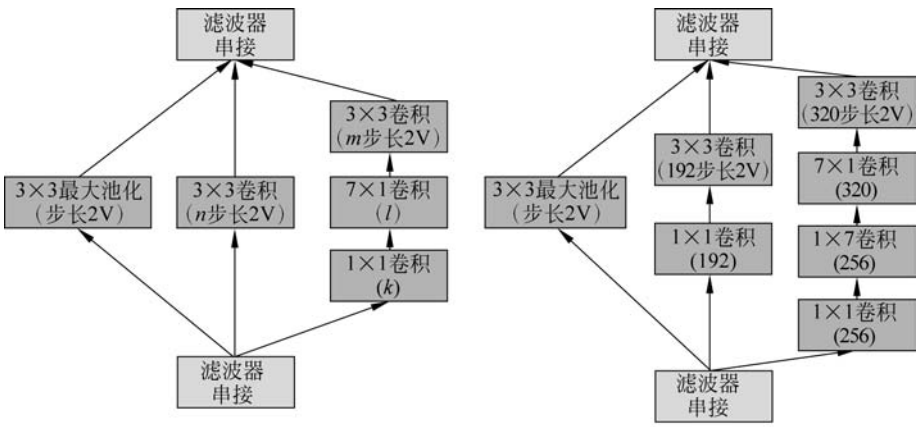
图 5-33　Inception-v4 网络中的模块结构



(c) Inception-C结构

图 5-33 （续）

7×1 卷积、3×3 卷积、3×3 卷积的卷积核个数。不同的网络结构中，其参数不同。Inception-v4 网络、后续残差网络 Inception-ResNet 的两种版本都采用了这个 Reduction-A 模块，只是 k、l、m、n 的个数不同。Inception-v4 中 k、l、m、n 分别为 192、224、256 和 384。



(a) Reduction-A结构　　(b) Reduction-B结构

图 5-34　Inception-v4 中的 Reduction 模块结构

4. Inception-ResNet

相较最初的 Inception 模块，其残差版本采用了更精简的 Inception 模块。每个 Inception 模块后紧连接着滤波层（没有激活函数的 1×1 卷积）以进行维度变换，以实现输入的匹配，这样补偿了在 Inception 块中的维度降低。残差网络在提出后，为了更好验证性能，在残差网络中尝试了不同的 Inception 版本，一个是“Inception-ResNet-v1”，计算代价跟 Inception-v3 大致相同，另一个“Inception-ResNet-v2”的计算代价跟 Inception-v4

网络基本相同。Inception-ResNet 的两个版本结构基本相同，只是细节不同。图 5-35 给出了 Inception-ResNet 的结构图[19]。图 5-35(a)为 Inception-ResNet 结构，其中包括了 Inception-ResNet-A、Inception-ResNet-B 和 Inception-ResNet-C 三种类型的模块，但两个版本的上述三种类型的 Inception 模块细节有差异；图 5-35(b)为 Inception-ResNet-v1 的 Stem 模块，而 Inception-ResNet-v2 的 Stem 模块与 Inception-4 的 Stem 模块相同。这里仍补充说明，图 5-35(b)中的 Stem 模块中的给出了步长和卷积的类型，“V”仍然代表有效卷积，因此尺寸会缩小。

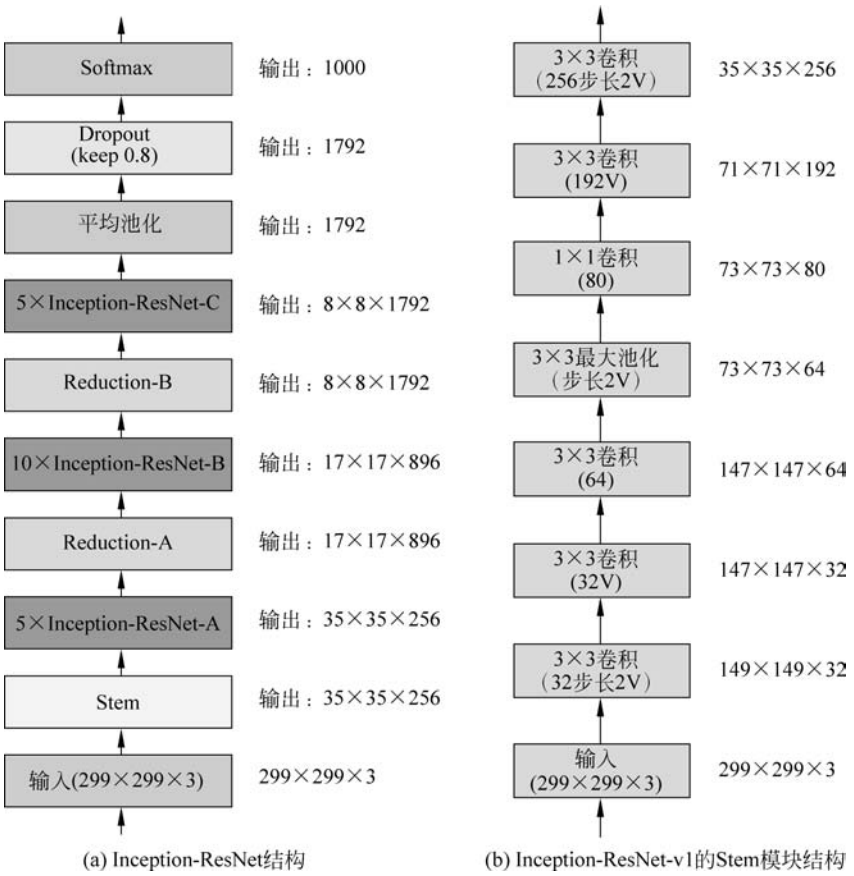


图 5-35　Inception-ResNet 网络结构

Inception-ResNet-v1 和 Inception-ResNet-v2 两个版本的网络中对应的 Inception-ResNet-A 模块如图 5-36 所示。其平面尺寸分别为 35×35，经过不同尺寸的卷积后，获得了感知度的特征图，进行级联后再通过 1×1 的无激活的线性卷积变换通道数，分别将通道数变换为 256 和 384。

Inception-ResNet-v1 和 Inception-ResNet-v2 两个版本的网络中对应的 Inception-ResNet-B 模块如图 5-37 所示。其平面尺寸分别为 17×17，经过不同尺寸的卷积后，获得了感知度的特征图，进行级联后再通过 1×1 的无激活的线性卷积变换通道数，分别将通道数变换为 896 和 1154。

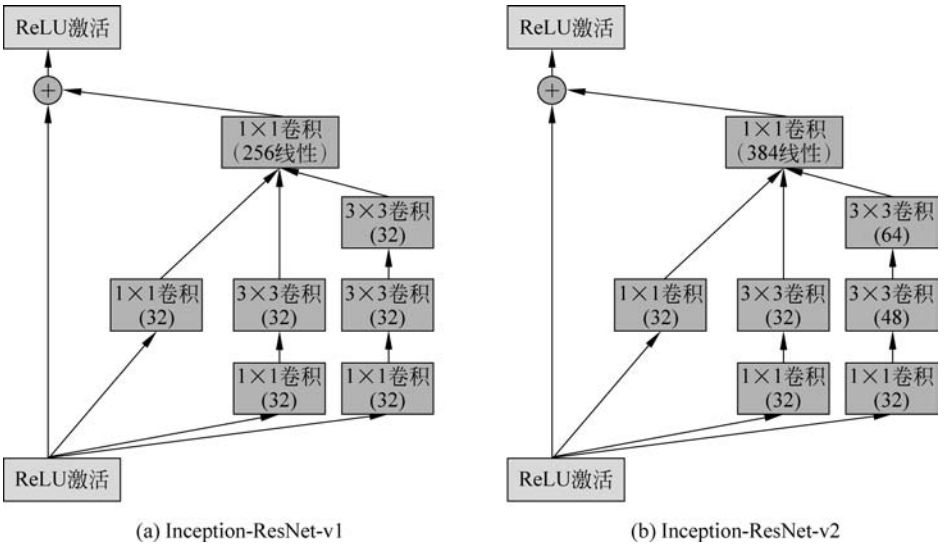


图 5-36　Inception-ResNet 中的 Inception-ResNet-A 模块结构(35×35)

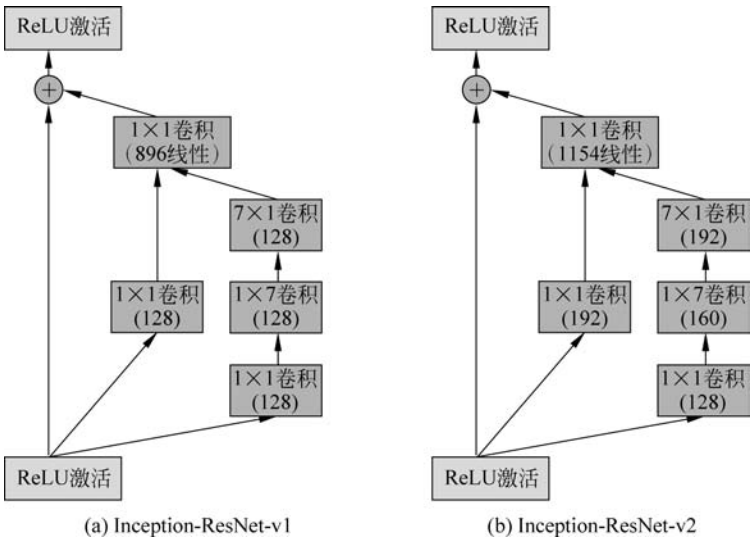


图 5-37　Inception-ResNet 中的 Inception-ResNet-B 模块结构

Inception-ResNet-v1 和 Inception-ResNet-v2 两个版本的网络中对应的 Inception-ResNet-C 模块如图 5-38 所示。其平面尺寸分别为 8×8，经过不同尺寸的卷积后，获得了感知度的特征图，进行级联后再通过 1×1 的无激活的线性卷积变换通道数，分别将通道数变换为 1792 和 2048。

Inception-ResNet-v1 和 Inception-ResNet-v2 对应的 35×35 到 17×17 的 reduction-A 模块与 Inception-v4 中的一样，如图 5-33(a)所示；Inception-ResNet-v1 和 Inception-ResNet-v2 对应的 17×17 变为 8×8 模块，即 Reduction-B 模块，如图 5-39 所示。

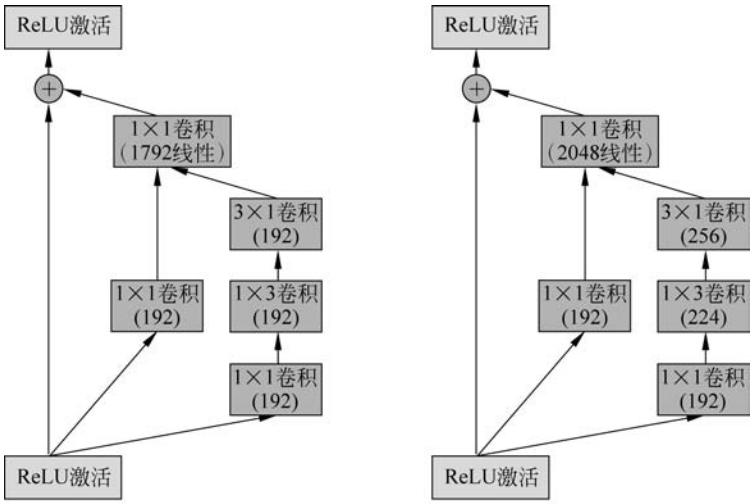


图 5-38　Inception-ResNet 中的 Inception-ResNet-C 模块结构

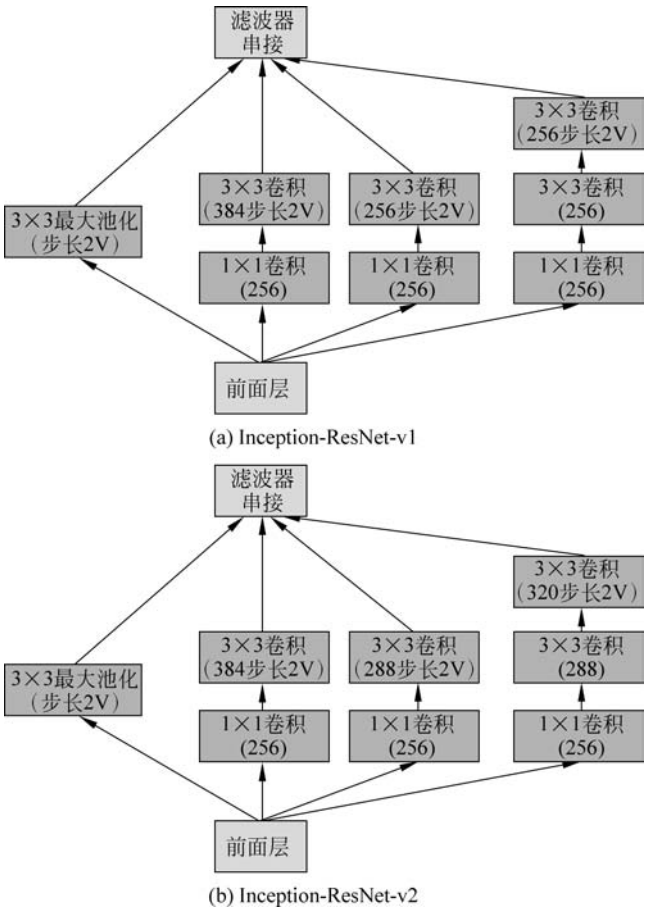


图 5-39　Inception-ResNet 中的 Reduction-B 模块结构

5.3.7 挤压激励网络

2017年，中国科学院软件研究所的胡杰等提出挤压激励网络（Squeeze-and-Excitation Networks，SENet）[20]，该网络获得了2017年ILSVR挑战赛的冠军，使得在ImageNet上的TOP-5错误率下降至2.25%。SENet网络的创新点在于关注通道之间的关系，希望模型可以自动学习到不同通道特征的重要程度。

1. SE模块

SE模块首先对卷积得到的特征图进行挤压得到通道级的全局特征，然后对全局特征进行激活学习各个通道间的关系，也得到不同通道的权重，最后乘以原来的特征图得到最终特征[21]。本质上，SE模块是在通道维度上做注意力或者门选操作，这种注意力机制让模型可以更加关注信息量最大的通道特征，而抑制那些不重要的通道特征。另外，SE模块是通用的，这意味着其可以嵌入现有的网络架构中。

如图5-40所示，SE模块主要包括挤压、激活两个操作，可以适用于任何映射 F_{tr}：$\boldsymbol{X}\rightarrow\boldsymbol{U}$，$\boldsymbol{X}\in\mathbb{R}^{H'\times W'\times C'}$，$\boldsymbol{U}\in\mathbb{R}^{H\times W\times C}$，卷积核为 $\boldsymbol{V}=[\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_C]$，其中 $\boldsymbol{v}_c$ 表示第 c 个卷积核。那么输出 $\boldsymbol{U}=[\boldsymbol{u}_1,\boldsymbol{u}_2,\cdots,\boldsymbol{u}_C]$ 为

$$\boldsymbol{u}_c=\boldsymbol{v}_c*\boldsymbol{X}=\sum_{s=1}^{C'}\boldsymbol{v}_c^s*\boldsymbol{x}^s \tag{5.14}$$

式中：$*$ 代表卷积操作；而 $\boldsymbol{v}_c^s$ 代表一个 s 通道的2维卷积核，其输入一个通道上的空间特征，它学习特征空间关系，但是由于对各个通道的卷积结果进行求和，所以通道特征关系与卷积核学习到的空间关系混合在一起。而SE模块就是为了抽离这种混杂，使得模型直接学习到通道特征关系。

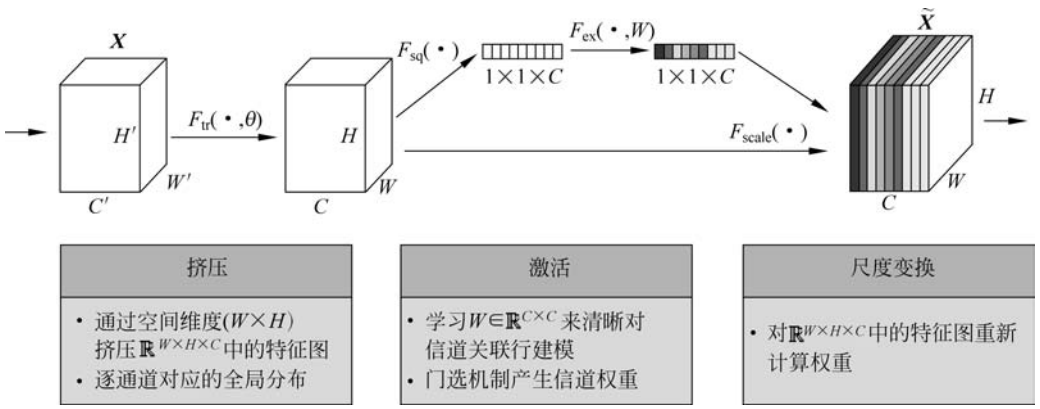


图5-40 SE模块

1）挤压

由于卷积只是在一个局部空间内进行操作，$\boldsymbol{U}$ 很难获得足够的信息来提取通道之间的关系，对于网络中前面的层，这一问题更严重，因为感受野比较小。为了缓解该问题，

在 SENet 中提出挤压运算，将一个通道上整个空间特征编码为一个全局特征，采用全局平均池化来实现（原则上也可以采用更复杂的聚合策略）：

$$z_c = \boldsymbol{F}_{\text{sq}}(\boldsymbol{u}_c) = \frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W} u_c(i,j) \tag{5.15}$$

其中：$\boldsymbol{z} = [z_1, z_2, \cdots, z_C] \in \mathbb{R}^C$。

2）激活

挤压运算得到全局描述特征，接下来采用另外一种运算来获取通道之间的关系。该运算需要满足两个准则：一是要灵活，可以学习到各个通道之间的非线性关系；二是学习关系不是互斥的，即允许多通道特征，非 one-hot 形式。基于此采用 Sigmoid 形式的门选机制：

$$\boldsymbol{s} = \boldsymbol{F}_{\text{ex}}(\boldsymbol{z}, \boldsymbol{W}) = \sigma(g(\boldsymbol{z}, \boldsymbol{W})) = \sigma(\boldsymbol{W}_2 \delta(\boldsymbol{W}_1 \boldsymbol{z})) \tag{5.16}$$

式中：δ 和 σ 分别为 ReLU 和 Sigmoid 函数，$\boldsymbol{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ 且 $\boldsymbol{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}$。为了降低模型复杂度以及提升泛化能力，采用包含两个全连接层的瓶颈结构，其中第一个全连接层起到降维的作用，降维系数为 r（是一个超参数），然后采用 ReLU 激活。第二个全连接层将其恢复为原始维度。最后将学习到的各个通道的激活值 $\boldsymbol{s}$（Sigmoid 激活，介于 0～1 之间）乘以原始特征 $\boldsymbol{U}$ 得到最终输出，即

$$\tilde{\boldsymbol{x}}_c = \boldsymbol{F}_{\text{scale}}(\boldsymbol{u}_c, s_c) = \boldsymbol{s}\boldsymbol{u}_c \tag{5.17}$$

其实整个操作可以看成学习到了各个通道的权重系数 s_c，从而使得模型对各个通道的特征更有辨别能力，这应该也算一种注意力机制。

2. SE 模块在 Inception 和 ResNet 上的应用

SE 模块的灵活性在于它可以直接应用现有的网络结构中，这里以 Inception 和 ResNet 为例进行阐释。对于 Inception 网络直接应用 SE 模块；对于 ResNet，SE 模块嵌入残差结构中的残差学习分支中，具体如图 5-41 所示。

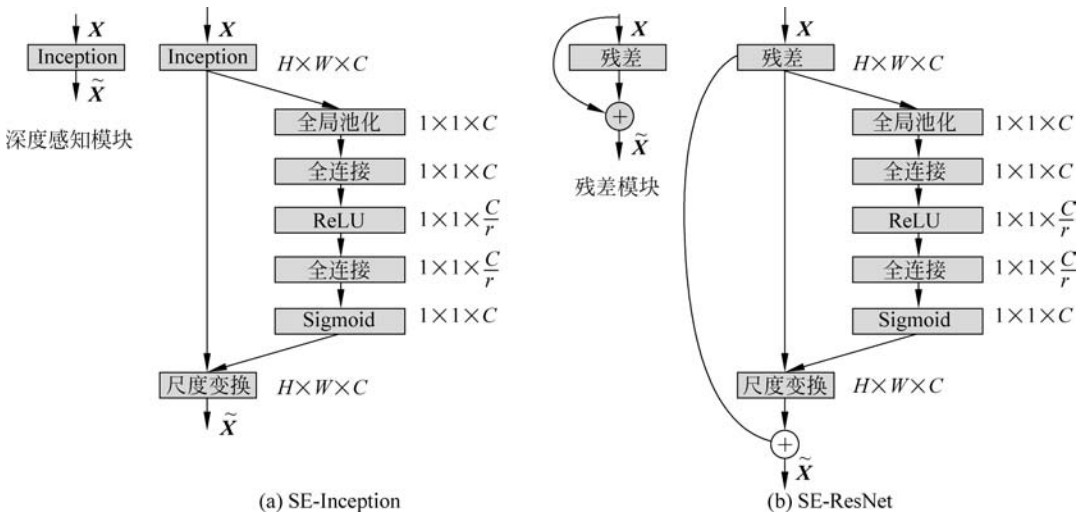


图 5-41　加入 SE 模块的 Inception 和 ResNet 网络

同样地，SE 模块也可应用于其他网络结构，如 ResNetXt、Inception-ResNet、MobileNet 和 ShuffleNet 中。表 5-10 为 SE 模块应用在其他网络的参数配置，这里给出 SE-ResNet-50 和 SE-ResNetXt-50 的具体结构。

表 5-10　SE 模块应用于其他网络的参数配置

输出尺寸	ResNet-50	SE-ResNet-50	SE-ResNeXt-50(32×4d)
112×112	卷积 7×7,64,步长为 2		
56×56	最大池化 3×3,步长为 2		
	[卷积,1×1,64 卷积,3×3,64 卷积,1×1,256] ×3	[卷积,1×1,64 卷积,3×3,64 卷积,1×1,256 全连接,[16,256]] ×3	[卷积,1×1,128 卷积,3×3,128 卷积,1×1,256 全连接,[16,256] C=32] ×3
28×28	[卷积,1×1,128 卷积,3×3,128 卷积,1×1,512] ×4	[卷积,1×1,128 卷积,3×3,128 卷积,1×1,512 全连接,[32,512]] ×4	[卷积,1×1,256 卷积,3×3,256 卷积,1×1,512 全连接,[32,512] C=32] ×4
14×14	[卷积,1×1,256 卷积,3×3,256 卷积,1×1,1026] ×6	[卷积,1×1,256 卷积,3×3,256 卷积,1×1,1024 全连接,[64,1024]] ×6	[卷积,1×1,512 卷积,3×3,512 卷积,1×1,1024 全连接,[64,1024] C=32] ×6
7×7	[卷积,1×1,512 卷积,3×3,512 卷积,1×1,2048] ×3	[卷积,1×1,512 卷积,3×3,512 卷积,1×1,2048 全连接,[128,2048]] ×3	[卷积,1×1,1024 卷积,3×3,1024 卷积,1×1,2048 全连接,[128,2048] C=32] ×3
1×1	全局平均池化,1000 维全连接,Softmax		

增加了 SE 模块后，模型参数以及计算量都会增加，以 SE-ResNet-50 为例，对于模型参数增加量为

$$\frac{2}{r}\sum_{s=1}^{S}N_sC_s^2$$

式中：r 为降维系数，S 为阶段数量，C_s 为第 s 个阶段的通道数，N_s 为第 s 个阶段的重复块数。当 r=16 时，SE-ResNet-50 只增加了约 10%的参数量，但计算量却增加不到 1%。

3. SE 模块的实验效果

SE 模块很容易嵌入其他网络中，为了验证 SE 模块的作用，文献在流行网络如 ResNet 和 VGG 中引入 SE 模块，测试其在 ImageNet 上的效果，如表 5-11 所示。可以看到所有网络在加入 SE 模块后分类准确度均有一定的提升，且计算量基本持平。

表 5-11　ImageNet 实验结果和复杂度

	原始		再现			SENet		
	Top-1 错误率/%	Top-5 错误率/%	Top-1 错误率/%	Top-5 错误率/%	GFLOPs /%	Top-1 错误率/%	Top-5 错误率/%	GFLOPs /%
ResNet-50[13]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101[13]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152[13]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50[22]	22.2	—	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101[22]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
VGG-16[11]	—	—	27.02	8.81	15.47	$25.22_{(1.80)}$	$7.70_{(1.11)}$	15.48
BN-Inception[17]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2[19]	19.9+	4.9+	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

此外，文献[21]还测试了 SE 模块在轻量级网络 MobileNet[23] 和 ShuffleNet[24] 上的效果，效果也有提升。最终采用了一系列的 SENet 进行集成，在 ImageNet 测试集上的 Top-5 错误率为 2.251%，赢得了 2017 年 ImageNet 竞赛的冠军。其中最关键的模型是 SENet-154，其建立在 ResNeXt 模型基础上。

从前面讨论可知，对于残差网络而言，通过构建不同的版本可以得到两种常用的残差网络结构，即 Inception-ResNet-v1 和 Inception-ResNet-v2。具体而言，残差网络中主要包括 Inception-ResNet-A、Inception-ResNet-B 和 Inception-ResNet-C 三种类型的 Inception 模块，以及 Reduction-A 和 Reduction-B 两类维度约简模块。首先在 Stem 模块，Inception-ResNet-v1 采用了图 5-35(b)中的结构，而 Inception-ResNet-v2 采用了与 Inception-v4 相同的结构；其次 Inception-ResNet-A、Inception-ResNet-B 和 Inception-ResNet-C 在两个版本中都有差异；再次约简模块，Reduction-B 在两种版本中略有差异。总体看来，两个版本的主要差异是 Reduction-A 结构的不同。表 5-12 给出了 Inception-v4 以及 Inception-ResNet-v1、Inception-ResNet-v2 的滤波器参数设置。

表 5-12　Reduction-A 参数配置

网　　络	k	l	m	n
Inception-v4	192	224	256	384
Inception-ResNet-v1	192	192	256	384
Inception-ResNet-v2	256	256	384	384

残差和非残差 Inception 的另外一个技术性区别是，Inception-ResNet 网络中，在传统层的顶部而非所有层的顶部中使用批归一化。在全部层使用批归一化是合理的，保持每个模型副本在单个 GPU 上就可以训练。结果证明，使用更大激活尺寸在 GPU 内存上更加耗时。在部分层的顶部忽略批归一化能够充分增加 Inception 块的数量。为了可以更好地利用计算资源，这种折中没有必要。

5.3.8 区域卷积神经网络及其拓展版本

RossGirshick 等于 2014 年提出区域卷积神经网络(Region Convolutional Neural Network，R-CNN)，是利用深度学习算法进行目标检测的开创者[25]。其本质就是来做目标检测和语义分割的神经网络。随后几年，提出了多种 R-CNN 的改进版本，如 Fast RCNN[26]、Faster RCNN[27]等，代表当时目标检测的前沿水平。

1. 图像分类、目标检测与图像分割

从图像中解析出可供计算机理解的信息，是机器视觉的中心问题。图像分类、目标检测、分割是计算机视觉领域的三大任务。下面具体介绍三大任务的内涵和区别。

图像分类即是将图像结构化为某一类别的信息，用事先确定好的先验类别或实例 ID 来描述图片。该任务是最简单、最基础的图像理解任务，也是深度学习算法最先取得突破和实现大规模应用的任务。其中，ImageNet 是最权威的评测集，每年的 ILSVRC 催生了大量的优秀深度网络结构，为其他任务提供了基础。根据不同的应用领域，可以进行人脸识别、场景识别等。由于图像分类通常聚焦于图像中最突出的物体，且将图像划分为单个类别，而现实世界的很多图像通常包含的不只是一个物体，此时如果使用图像分类模型为图像分配一个单一标签相对而言比较粗糙并不准确，因此目标检测应运而生。

由于图像分类任务关心图像的整体信息，给出整张图片的内容描述，难以描述图片中的更精确信息；而目标检测能识别出一张图片的多个物体，关注特定的物体目标，同时获得所关注目标的类别和位置信息。相比分类，目标检测给出的是对图片前景和背景的理解，需要从背景中分离出感兴趣的目标，并确定这一目标的描述(类别和位置)，因此检测模型的输出是一个列表，列表的每一项使用一个数组给出检出目标的类别和位置(常用矩形检测框的坐标表示)。目标检测在很多场景有用，如无人驾驶和安防系统。

图像分割包括语义分割和实例分割[28]，前者是对前背景分离的拓展，要求分离开具有不同语义的图像部分，后者是检测任务的拓展，要求描述出目标的轮廓(相比检测框更为精细)。分割是对图像的像素级描述，它赋予每个像素类别(实例)意义，适用于理解要求较高的场景，如无人驾驶中对道路和非道路的分割。

图 5-42 和图 5-43 给出了三大任务的示例。图 5-42(a)为图像分类，给出了图像中的主题信息表达；图 5-42(b)为图像分类与定位，不仅给出了图像中目标的类别，还将目标的位置信息以矩形框的形式给出；图 5-42(c)为多目标的目标检测，同时给出了所有目标的分类信息和位置信息；图 5-42(d)为多目标的实例分割，将实例的边界标记出来。

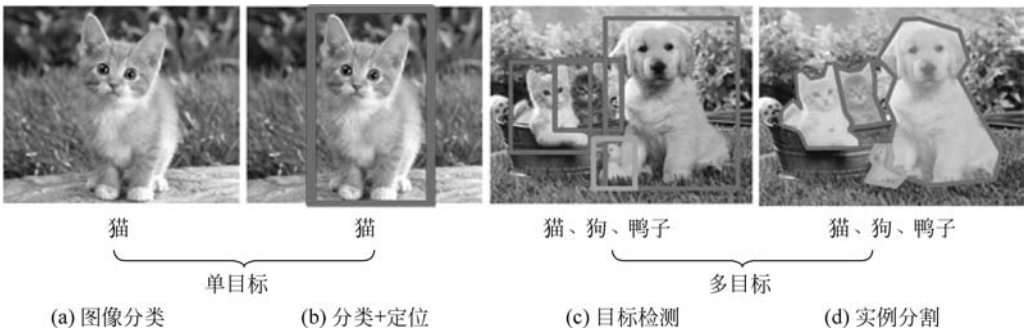


图 5-42　图像分类、目标检测与语义分割

图 5-43 是多目标任务的示意图，图 5-43(a)和(b)分别为图像分类和目标检测，图 5-43(c)和(d)分别为图像分割中的语义分割和实例分割。从图中可以看出，语义分割中对相同的目标类型，例如 cube 采用了相同的颜色表示，对于重叠部分没有给出边界，实例分割中对于相同种类的个体也进行了分别表示。

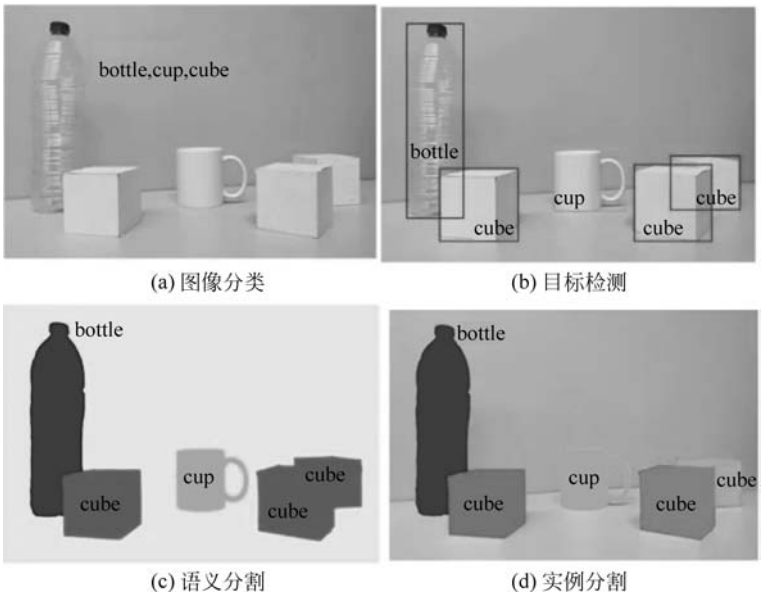


图 5-43　图像分类、目标检测与语义分割

图 5-44 给出了一个更为详细的示例。从图中可以看出，置信度也说明了当前的检测框属于需要查找的目标的一个概率值。为了减少错误检测，通常会给出一个阈值，通过该阈值来过滤掉一些检测错误的目标。图 5-44 检测出非常多的目标区域，不同的目标区域也通过不同颜色的矩形框表示出来。

2. 区域卷积神经网络

区域卷积神经网络是针对图像目标检测任务而提出的，它的目标是解决物体识别的

图 5-44　目标检测的详细示例

难题。在获得特定图像后，希望能够绘制图像中所有物体的边缘。这一过程可以分为区域建议和分类两个部分。

基于 R-CNN 网络的目标检测系统由三个模块组成(图 5-45)[25]：第一个模块生成类别独立候选区域，这些候选区域定义了检测器可用的候选边界框集合，其主要方法就是选择性搜索方法；第二个模块是从各个区域提取固定长度特征向量的大型卷积神经网络；第三个模块是一组类别特定的线性 SVM 分类器。下面将介绍每个模块的设计思路，描述其测试时间使用情况，详细了解其参数的学习方式，并展示在 PASCAL VOC 2010-12 上的结果。

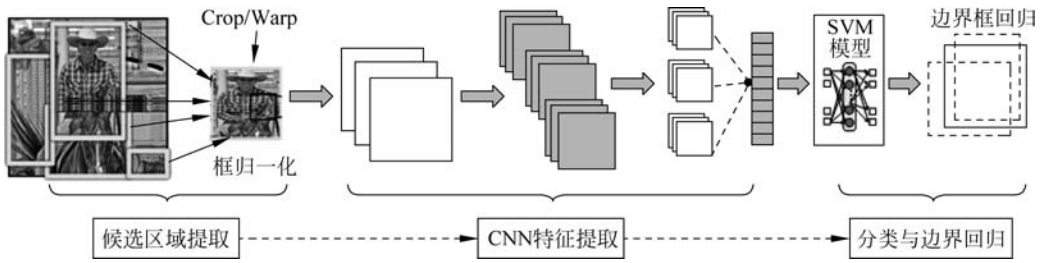


图 5-45　目标检测的 R-CNN 网络模型

训练过程从前到后可分成五个部分：一是构造微调 CNN 网络的训练数据集；二是对 CNN 网络进行微调后，获得候选区域的深度特征表示；三是区域分类算法的数据集构造；四是区域分类算法 SVM 算法的训练；五是进行边界框回归算法提高精度。

1) 独立候选区域生成算法

图像包含的信息非常丰富，其中的物体有不同的形状、尺寸、颜色、纹理，因此很难从图像中识别出一个物体，而找到物体后确定物体在图像中的位置更加困难。图 5-46 给出了四个例子来说明物体识别的复杂性以及难度。图 5-46(a)中的场景是一张桌子，桌子上面放了碗、瓶子和其他餐具等，此时图像中不同物体之间有一定的层次关系，比如要识别“桌子”，可能只是指桌子本身，也可能包含其上面的其他物体。图 5-46(b)中给出了两只猫，可以通过纹理来找到这两只猫，却又需要通过颜色来区分它们。图 5-46(c)中变色

龙和周边颜色接近，可以通过纹理来区分。图 5-46(d)中的车辆，很容易把车身和车轮看作一个整体，但它们两者之间在纹理和颜色方面差别都非常大。

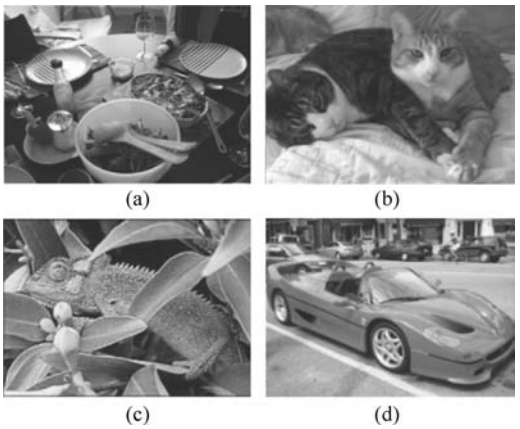


图 5-46　复杂场景的图像示例

由于实际情况的复杂性，因此进行目标识别时不能通过单一的策略来区分不同的物体，需要充分考虑图像物体的多样性。此外，在图像中物体布局有一定的层次关系，考虑这种关系才能够更好地对物体的类别进行区分。在详细阐述选择性搜索算法前，先给出选择性搜索算法的设计考虑：一是要适应不同尺度。目标可以在图像以任何比例来出现，且某些目标的边界不如其他目标清晰。穷举搜索通过改变窗口大小来适应物体的不同尺度，选择搜索必须考虑所有对象比例，通过分层算法来实现是很自然的一种选择，因此选择性搜索采用了图像分割以及层次算法来解决这个问题。二是要采用多样化策略。单一的策略无法应对多种类别的图像。正如图 5-46 中所观察到的，虽然可能仅通过颜色、纹理或封闭部件就能将区域形成目标，但是阴影和光色等照明条件很可能影响区域判定。因此，希望采用一套多样化的策略来应对所有情况。三是要能够快速计算。选择性搜索的目的是产生一组可能的对象位置，以用于实际的对象识别框架。该集合的创建不应成为计算瓶颈，因此算法的计算速度也成为考虑的一个重要因素。

针对上述算法的设计考虑，下面介绍选择性搜索算法的具体步骤。

(1) 基于图表示的图像分割方法。

图像分割的主要目的是将图像分割成若干个特定的、具有独特性质的区域，然后从中提取出感兴趣的目标。其中，图像区域之间的边界定义是图像分割算法的关键，研究人员基于图表示给出图像区域之间的边界判断[29]，分割算法就是利用这个判断标准使用贪婪选择来产生分割。该算法在时间效率上基本与图像表示的边数量呈线性关系，而图像的图表示边与像素点成正比，也就说图像分割的时间效率与图像的像素点个数呈线性关系。这个算法有一个非常重要的特性，它能保持低变化区域的细节，同时能够忽略高变化区域的细节。这个性质特别重要，对图像有一个很好的分割效果，即能够找出视觉上一致的区域，换而言之就是高变化区域有一个很好聚合。

(2) 层次合并算法。

根据上述方法获得原始区域,然后进行区域合并,区域的合并方式是层次合并算法(Hierarchical Grouping Algorithm,HGA),类似于哈夫曼树的构造过程。区域包含的信息比像素丰富,更能够有效地代表物体的特征。层次合并算法如算法 5-1 所示。

算法 5-1 层次合并算法

输入:彩色图片(三通道)
输出:目标定位位置的候选集合 L
采用基于图表示的图像分割方法获得原始分割区域 $R=\{r_1,r_2,\cdots,r_n\}$
初始化相似度集合 $S=\varnothing$
for each 相邻区域对(r_i,r_j)do
    计算两两相邻区域之间的相似度 $s(r_i,r_j)$
    将其添加到相似度集合 S 中,即 $S=S\cup s(r_i,r_j)$
End
while $S=\varnothing$ do
    从相似度集合 S 中找出最大相似度 $s(r_i,r_j)=\max(S)$
    将最大相似度的两个区域 r_i 和 r_j 合并成为一个区域 r_t,即 $r_t=r_i\cup r_j$
    从相似度集合中除去所有关于 r_i 区域的相似度,即 $S=S\backslash s(r_i,r_*)$
    从相似度集合中除去所有关于 r_j 区域的相似度,即 $S=S\backslash s(r_*,r_j)$
    计算 r_t 与其相邻区域(原先与 r_i 或 r_j 相邻的区域)的相似度 S_t
    更新相似度集合 S,把区域 S_t 添加到区域集合 S 中,即 $S=S\cup S_t$
    同时更新区域集合 R,把区域 r_t 添加到区域集合 R 中,即 $R=R\cup r_t$
获取每个区域的边界框,该结果就是物体位置的可能结果 L

从算法 5-1 可知,层次合并算法中的一个关键点是计算两个区域之间的相似度 $s(r_i,r_j)$,计算公式为

$$s(r_i,r_j)=a_1 s_{\text{colour}}(r_i,r_j)+a_2 s_{\text{texture}}(r_i,r_j)+a_3 s_{\text{size}}(r_i,r_j)+a_4 s_{\text{fill}}(r_i,r_j) \tag{5.18}$$

从式(5.18)中可以看出,其相似度采用了多样化的形式,包括颜色相似度 $s_{\text{colour}}(r_i,r_j)$、纹理相似度 $s_{\text{texture}}(r_i,r_j)$、尺寸相似度 $s_{\text{size}}(r_i,r_j)$和吻合相似度 $s_{\text{fill}}(r_i,r_j)$四种形式。

(3) 颜色相似度。

将某个区域 r_i 内的像素值分成 25bins,那么 3 个通道就组成了长度为 75 的特征向量$\boldsymbol{C}_i=\{c_i^1,\cdots,c_i^n\}$,然后对$\boldsymbol{C}_i$ 进行 L_1 归一化(也就是$\boldsymbol{C}_i$ 的每一个值除以区域 r_i 内像素点的个数),特征向量$\boldsymbol{C}_i$ 实际上表示了像素值的分布,于是区域 r_i 和区域 r_j 的颜色相似度为

$$s_{\text{colour}}(r_i,r_j)=\sum_{k=1}^{n}\min(c_i^k,c_j^k) \tag{5.19}$$

颜色相似度度量了两个区域像素值分布的重叠程度。当两个区域合并之后,新的区域的特征向量$\boldsymbol{C}_t$ 就是合并前各个特征向量之和,但是由于特征向量$\boldsymbol{C}_i$ 进行了 L_1 归一

化，所以没有归一化的特征向量为 $\text{size}(r_i)\times \boldsymbol{C}_i$。因此，合并后的特征向量为

$$\boldsymbol{C}_t=\frac{\text{size}(r_i)\times \boldsymbol{C}_i+\text{size}(r_j)\times \boldsymbol{C}_j}{\text{size}(r_i)+\text{size}(r_j)} \tag{5.20}$$

(4) 纹理相似度。

这里的纹理采用 SIFT-Like 特征，具体做法是对每个颜色通道的 8 个不同方向计算方差 $\sigma=1$ 的高斯微分，每个通道每个颜色获取 10bins 的直方图(L_1-norm 归一化)，这样就可以获取到一个 240 维的向量 $\boldsymbol{T}_i=\{t_i^1,\cdots,t_i^n\}$。区域之间纹理相似度计算方式和颜色相似度计算方式类似，合并之后新区域的纹理特征计算方式和颜色特征计算相同，即

$$s_{\text{texture}}(r_i,r_j)=\sum_{k=1}^{n}\min(t_i^k,t_j^k) \tag{5.21}$$

(5) 尺寸相似度。

尺寸是指区域中包含像素点的个数。使用尺寸相似度计算，主要是为了尽量让小的区域先合并，即

$$S_{\text{size}}(r_i,r_j)=1-\frac{\text{size}(r_i)+\text{size}(r_j)}{\text{size}(\text{im})} \tag{5.22}$$

式中：im 是指整幅图像。

(6) 吻合相似度。

吻合相似度主要是衡量两个区域是否更加“吻合”，其指标是合并后的区域的边界框(能够框住区域的最小矩形(没有旋转))越小，其吻合度越高。如果某一个区域被另一个区域完全包裹，这两个区域就应该融合，或者两个区域几乎不接触，这两个区域应该就属于不同的区域。其计算方式为

$$s_{\text{fill}}(r_i,r_j)=1-\frac{\text{size}(\text{BB}_{ij})-\text{size}(r_i)-\text{size}(r_j)}{\text{size}(\text{im})} \tag{5.23}$$

式中：BB_{ij} 为区域 r_i 和区域 r_j 的外接矩，im 指整幅图像。

2) CNN 微调训练集构造

给定一系列的输入图像，CNN 微调训练集如何构造是一个核心问题。首先用选择性搜索方法在每个图像上生成很多的候选区域，然后在每张图上依次计算每个候选区域与图中目标的真实标注框之间的重叠程度，重叠程度通常用交并比(IoU)来表示。如果重叠程度大于 0.5，则标记这个区域为此目标的正样本；否则，为负样本。对所有的训练图像执行这样的操作，把得到的所有的候选区域保存下来。假如有 20 个目标，对每个目标都有一些属于其类别的样本称为正样本，其他的不属于任何类的区域图像称为负样本。

3) 训练卷积神经网络来抽取候选区域深度特征

提取候选区域的深度特征采用了 AlexNet 结构。AlexNet 结构如图 5-11 所示，中间层配置与 AlexNet 的默认配置相同，这里主要关注其输入和输出。输入是 227×227 大小的图像，提取到的是一个 4096 维度的特征向量，由于是 1000 类的分类任务，因此最后一层的节点数为 1000。需要注意的是，由于提取到的候选区域大小都是不相同的，因此

需要把每个区域调整尺寸到规定大小 227×227，每个区域提取的深度特征的维度依旧是 4096 维。输出层要根据任务的不同做出相应的改变，原本 AlexNet 中输出层数目 1000 是指 1000 类的分类任务，如果是对 VOC 数据集进行处理，则要把输出节点数目设置为 20+1(其中，20 表示有 20 个待识别目标，1 表示背景类)；对 ILSVRC2013 来说，要设置为 200+1。原文[25]采用 AlexNet 网络受限于当时 CNN 的发展，后续的多种 CNN 都可以用于候选区域深度特征的提取。

下面以 VOC 数据集为例进行说明。假设按照第一部分已经构造好训练数据集，其类别共有 21 类，则每个类都有自己对应的样本区域，且这些区域是类别独立的。然后将训练样本输入到改进版的 AlexNet 进行训练，AlexNet 的最后一层依旧是 Softmax 层，即此时有监督地训练一个 21 类的图像分类网络。卷积神经网络的具体训练方法：首先用 ILSCRC2012 数据集对网络进行有监督预训练，然后使用第一部分里面提取的训练集进行微调。微调的时候采用 SGD 的方法，学习率设置为预训练时候的 1/10(具体为 0.001)，这样使得网络慢慢地收敛。mini-batch 的大小为 128，mini-batch 分成两部分，即从第一部分保存下来的候选区域里每次随机抽取 32 个正样本(在所有类别上)、96 个负样本(背景类)。

4) 训练集构造(用于训练多个 SVM 分类器)

前面阐述了如何构造训练集来对预训练后的 CNN 进行微调，与真实标注框之间的 IoU 大于 0.5 的区域都被标定为正样本。此时，CNN 已经训练完成，现在要为每个类别(如猫、狗等)单独训练一个二分类 SVM 分类器，如“SVM_猫”专门检测猫这个目标，“SVM_狗”专门检测狗这个目标，因此需要对每个二分类的 SVM 分类器构造其训练集合。以猫目标为例，其具体做法：首先以每张图像上猫目标的真实标注框作为正样本。然后在图像上生成很多候选区域，考察每个区域与猫目标的真实标注框之间的 IoU，如果 IoU 小于 0.3，就认定这个区域为负样本，IoU 在 0.3～1 之间的不用做训练。可以想象，对于训练“SVM_猫”来说，正样本都是包含猫的区域，负样本有可能是背景，也有可能包含了其他目标，如狗。但无论如何，只要该区域不包含猫，或者只包含了一部分猫，其 IoU 小于 0.3，都会被标记为负样本。

5) 为每个类训练一个二分类 SVM 分类器

假如有 20 个类，那么就要训练 20 个二分类器。上一部分讲述了如何为每一个两分类 SVM 构造相应的训练集，训练集里面的正样本和负样本都要使用上面已经训练好的 CNN 来提取各自的 4096 维度的特征向量，然后再对分类器进行训练。需要说明的是，SVM 非常适合用于高维度小样本的分类任务，这也是在 SVM 之前要用 CNN 将每个图片表示为 4096 维矢量的原因。利用 CNN 将图片映射到高维准线性空间，便于利用 SVM 进行处理。

6) 边界框回归算法

类似于为每个类别训练一个二分类 SVM，还需要为每个类别构造一个边界框回归器来提升检测的准确率。

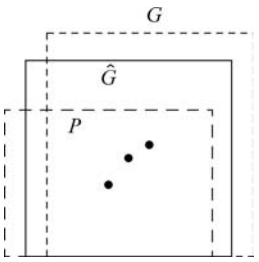


图 5-47　候选目标框 P、预测目标框 $\hat{G}$ 和真实目标框 G

训练算法的输入是 N 个样本对 $\{(P^i, G^i)\}_{i=1,\cdots,N}$，其中 $P^i=(P^i_x, P^i_y, P^i_w, P^i_h)$ 是指候选边界框 P^i 的中心像素坐标以及框的宽度和高度，为了后面方便表示，将上标号 i 省略。同样，将每个边界框的真实标注表示为 $G=(G_x, G_y, G_w, G_h)$，学习算法的目标是学习候选框 P 和真实标注框 G 的变换。

图 5-47 候选目标框 P、预测目标框 $\hat{G}$ 和真实目标框 G 的示意图，分别用长虚线框、实线框和短虚线框来表示，现在的目的就是想学习到一种变换，这个变换可以将 P 映射到 G。变换的过程如下：

$$\begin{aligned}
\hat{G}_x &= P_w d_x(P) + P_x, \quad d_x(P) = \boldsymbol{w}_x^{\mathrm{T}} \boldsymbol{\phi}_5(P) \\
\hat{G}_y &= P_h d_y(P) + P_y, \quad d_y(P) = \boldsymbol{w}_y^{\mathrm{T}} \boldsymbol{\phi}_5(P) \\
\hat{G}_w &= P_w \exp(d_w(P)), \quad d_w(P) = \boldsymbol{w}_w^{\mathrm{T}} \boldsymbol{\phi}_5(P) \\
\hat{G}_h &= P_h \exp(d_n(P)), \quad d_h(P) = \boldsymbol{w}_h^{\mathrm{T}} \boldsymbol{\phi}_5(P)
\end{aligned} \tag{5.24}$$

式中：$d_*(P)$ 是候选样本框 P 用已训练 CNN 计算 pool5 层特征 $\boldsymbol{\phi}_5(P)$ 的线性函数，即 $d_*(P)=\boldsymbol{w}_*^{\mathrm{T}}\boldsymbol{\phi}_5(P)$，其中 $\boldsymbol{w}_*=\{\boldsymbol{w}_x, \boldsymbol{w}_y, \boldsymbol{w}_h, \boldsymbol{w}_w\}$ 是待学习的回归器参数。因此，问题变得很直接：就是通过一定的方法求出变换参数 $\boldsymbol{w}_*=\{\boldsymbol{w}_x, \boldsymbol{w}_y, \boldsymbol{w}_h, \boldsymbol{w}_w\}$，即为回归器，$\boldsymbol{w}_*=\{\boldsymbol{w}_x, \boldsymbol{w}_y, \boldsymbol{w}_h, \boldsymbol{w}_w\}$ 通过正则化的最小均方误差函数（脊回归）来实现，即

$$\hat{\boldsymbol{w}}_* = \underset{\boldsymbol{w}_*}{\operatorname{argmin}} \sum_{i=1}^{N} (t^i_* - \boldsymbol{w}_*^{\mathrm{T}} \boldsymbol{\phi}_5(P^i))^2 + \lambda \| \boldsymbol{w}_* \|^2 \tag{5.25}$$

其训练对 (P, G) 的回归目标 t^i_* 表示为

$$\begin{cases}
t_x = (G_x - P_x)/P_w \\
t_y = (G_y - P_y)/P_h \\
t_w = \log(G_w/P_w) \\
t_h = \log(G_h/P_h)
\end{cases} \tag{5.26}$$

作为典型的正则化最小均方误差问题，上述优化可以有效地通过闭式解形式获得。

在执行边界框回归时，有两个细微的问题需要注意：一是正则化很重要，根据验证集将 λ 设置为 1000。二是在选择要使用的训练对 (P, G) 时必须小心。直观地讲，如果 P 远离所有真实标注框，那么将 P 转换为真实标注框 G 的任务就没有意义，因为距离较远时，问题演变成复杂的非线性问题，线性回归假设前提不合理，这种学习问题变成无效学习问题。因此，候选框 P 至少在一个真实标注框附近时才进行学习。当且仅当 IoU 大于阈值（利用验证集将其设置为 0.6）时，通过将 P 分配给最大 IoU（如果 IoU 大于 1）的真实标注框 G 来实现“接近”，所有未分配的投标都将被丢弃。为了学习特定类别的回归问题，应为每个对象类执行一次此操作。

在测试时，会对每个候选框进行评分，并仅预测一次其新的检测窗口。原则上，可以

重复此过程(给新预测的边界框重新打分,然后根据该边界框预测新边界框,依此类推)。但是,发现迭代并不能改善结果。

7) R-CNN 的缺点

R-CNN 利用神经网络进行自底向上的候选区域判定,实现对目标的分类和定位,在 Pascal VOC 2012 的数据集上,将目标检测的验证指标 mAP 提升到 53.3%,相对于之前最好的结果提升了 30%。在 R-CNN 提出的同时,也给出了一个实用的方法策略,当缺乏大量的标注数据时,通过迁移学习可以减少对数据的依赖性,即采用在其他大型数据集训练过后的神经网络,然后在小规模特定的数据集中进行微调。

R-CNN 存在以下缺点:

(1) 重复计算。重复为每个候选框提取特征非常耗时。选择性搜索为每张图像产生大约 2000 个候选框,那么每张图像需要经过 2000 次完整的 CNN 前向传播得到特征,而且这 2000 个候选框有很多重叠的部分,造成很多计算都是重复的,这将导致计算量大幅上升,相当耗时。

(2) 性能瓶颈。由于所有的候选框会被放缩到固定的尺寸,这将导致图像的畸变,不符合物体的常见比例;而且由于重复计算的问题,时间上难以容忍多尺度多比例的数据增强方法去训练模型,使得模型的性能很难有进一步的提升。

(3) 步骤烦琐。整个训练过程分为多个阶段,首先对卷积神经网络微调训练;然后提取全连接层特征作为 SVM 的输入,训练得到目标检测器;最后训练边框回归器。整个过程步骤烦琐不易操作,而且每个阶段分开训练,不利于取得最优的解。此外,每个阶段得到的结果都需要保存,消耗大量的磁盘空间。

(4) 训练占用内存大。每一类分类器和回归器都需要大量的特征作为训练样本。

(5) 目标检测速度慢。由于每次都需要利用获取候选框,并且利用 CNN 进行特征表示,因此目标检测速度较慢,难以适应实时性要求高的场合。

3. 空间金字塔池化网络

前面分析可以看出,R-CNN 候选区域缩放后的畸变问题和提取特征时的重复计算导致了模型性能和速度的瓶颈。何凯明等提出的空间金字塔池化网络(Spatial Pyramid Pooling Network,SPPNet)[30]从两个方面进行改进,有效地解决了传统卷积神经网络对输入图像尺寸的限制,在保证性能的同时,检测速度也有了较大的提升。在 2014 年的 ImageNet 大规模视觉识别挑战赛(ILSVRC)中,该方法在 38 个参赛团队中取得目标检测第二名、图像分类第三名的成绩。

1) SPP-Net 的主要改进

SPP-Net 的改进主要体现在两个方面:一是解决传统 R-CNN 在提取特征前需要变换到特定尺寸的问题;二是解决图片的多个候选框重复计算的问题。

CNN 一般分为两个部分,即卷积层和全连接层。卷积层不要求固定大小的输入,但全连接层在设计时固定了神经元的个数,故需要固定长度的输入。这也就是 CNN 需要固定输入大小的问题所在。SPPNet 的解决办法是使用“空间金字塔变换层”将接收任意

大小的图像输入，输出固定长度的向量，这样就能让 SPPNet 可接收任意大小的输入图片，不需要对图像做截断/拉伸操作。

在 R-CNN 中，每个候选区域都要送入 CNN 内提取特征向量，假设一张图片有 2000 个候选区域，那么需要经过 2000 次 CNN 的前向传播，这 2000 次重复计算过程会有大量计算冗余，耗费大量时间。SPPNet 提出了一种从候选区域到全图的特征映射之间的对应关系，通过映射关系可以直接获取候选区域的特征向量，不需要重复使用 CNN 提取特征，从而大幅度缩短训练时间。

2）金字塔池化层

空间金字塔池化层接收任意大小的输入，输出固定的向量。空间金字塔池化（Spatial Pyramid Pooling，SPP）层可以对图片提取特征，图 5-48 给出其原理示意，即将图片分成 1×1、2×2、4×4 大小的 bin 块，然后将图片以三种方式切割并提取特征，这样可以得到一共 1+4+16=21 种特征，即以不同的大小的 bin 块来提取特征的过程就是空间金字塔池化。

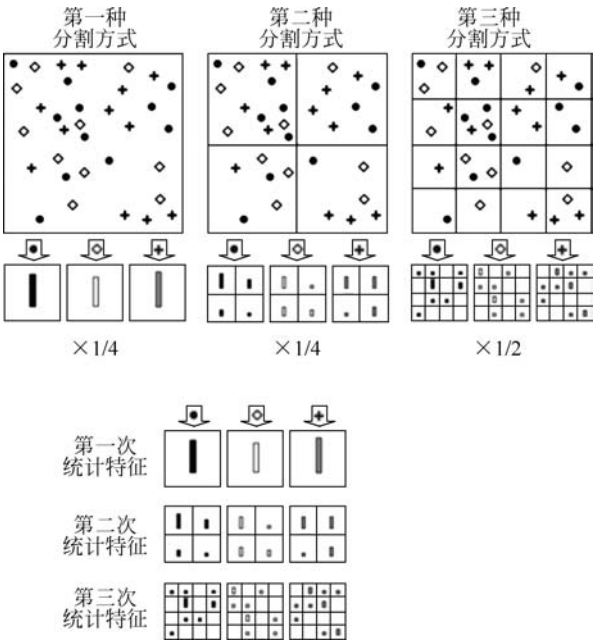


图 5-48　空间金字塔池化层原理示意

空间金字塔池化层与传统的池化层的不同在于空间金字塔池化层由不同尺度的池化层构成，并且池化层的大小与输入的特征图矩阵的通道数成正比。设输入特征图矩阵的尺寸为 $a\times a$，数目为 k，池化层的分割尺度为 $n\times n$，则该池化层的窗口大小为 $[a/n]$，移动步长为 $[a/n]$，输出的池化后的特征向量长度为 $n\times n\times k$ 维。n 可以取多个值，构成多个尺度上的池化层，这样空间金字塔池化层的输出特征向量的维度只与输入特征图矩阵的数目和池化层的分割尺度有关，而不再对网络模型的输入图像有尺寸限制。

下面以具体的例子来说明，根据前面步骤，不同大小候选区域在特征图上映射送入

SPP层，如图5-49所示。图中卷积层特征图通道数为256，SPP层尺度分成三种，即分成1×1(塔底)、2×2(塔中)、4×4(塔顶)三张子图，对每个子图的每个区域做最大池化，再将得到的特征连接到一起，得到(16+4+1)×256维特征向量。因此，无论输入图像大小如何，空间金字塔池化层将该图像变换为固定维度(16+4+1)×256的特征。

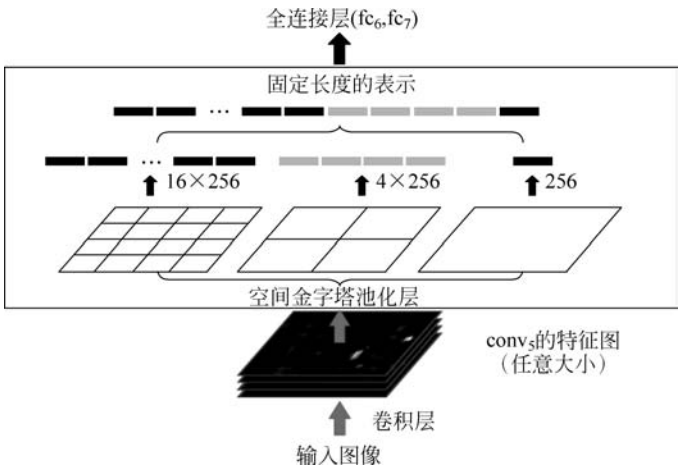


图5-49　空间金字塔池化层的网络结构

3) 候选区域到特征图映射变换

在讲解算法之前，先给出感受野前向变换的具体过程，感受野前向变换是指某一层输出结果中一个元素所对应的上一层的区域大小。

(1) 感受野变换。

由卷积神经网络知识可知，在卷积参数计算过程中，卷积层处理后的输出尺寸与输入尺寸、滤波器尺寸、步长和填充大小的关系如表5-13所示。

表5-13　卷积层输出尺寸与其他参数的关系

类　　型	大　　小
输入尺寸	$W_1 \times H_1$
卷积核	$F \times F$
输出尺寸	$W_2 \times H_2$
步长	S
填充大小	P

则有：

$$\begin{cases} W_2 = \dfrac{W_1 - F + 2P}{S} + 1 \\ H_2 = \dfrac{H_1 - F + 2P}{S} + 1 \end{cases} \tag{5.27}$$

式(5.27)是当前层到下一层的推导,如果已知后层感受野大小,则从式(5.23)可以推导出前一层的感受野计算式为

$$\begin{cases} W_1 = (W_2 - 1)S - 2P + F \\ H_1 = (H_2 - 1)S - 2P + F \end{cases} \tag{5.28}$$

相邻两层的感受野计算公式为

$$r_i = (r_{i+1} - 1)S_i + F_i - 2P_i \tag{5.29}$$

$$r_i = r_{i+1} \tag{5.30}$$

式中:r_i 为感受野。式(5.29)对卷积层和池化层成立,式(5.30)对神经元层(如 ReLU 函数、Sigmoid 等)成立。

图 5-50 给出了从后向前计算感受野的过程。从图中可以看出,特征图 Map3 中的 1×1 区域对应特征图 Map2 中的 7×7 区域,而 Map2 中的 7×7 区域对应于 Map1 中的 11×11 区域。其中 Map1 和 Map2 之间的参数可以为 $F=5, P=0, S=1$ 或 $F=5, P=3, S=2$,而 Map2 和 Map3 之间的参数可以为 $F=7, P=0, S=1$,其相互关系只要满足式(5.27)即可,但不同的参数性能有差异。

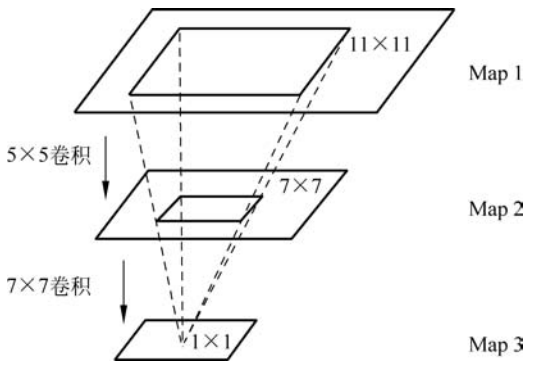


图 5-50 感受野变化

感受野计算时有一些情况需要说明:一是反向第一层卷积层的输出特征图像素的感受野尺寸等于滤波器尺寸;二是深层卷积层的感受野尺寸和它之前所有层的滤波器尺寸和步长有关系;三是计算感受野尺寸时,忽略了图像边缘的影响,即不考虑填充的尺寸。

通常需要知道网络中任意两个特征图之间的坐标映射关系(一般是中心点之间的映射),其计算公式为

$$p_i = s_i p_{i+1} + (k_i - 1)/2 - \text{padding} \tag{5.31}$$

$$p_i = p_{i+1} \tag{5.32}$$

式(5.31)对卷积层和池化层成立,式(5.32)对神经元层(如 ReLU 函数、Sigmoid 等)成立,其中 k_i 表示卷积核尺寸,padding 表示补零。式(5.31)可以计算任意一层输入与输出的坐标映射关系,如果是计算任意特征图之间的关系,只需要用简单的组合就可以得到。图 5-51 是一个简单的例子,其中步长和补零个数已经给出。

则有

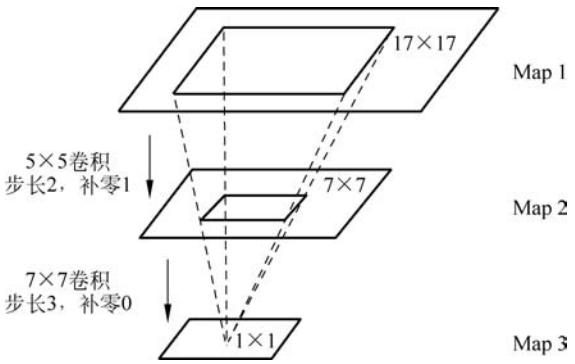


图 5-51　感受野变化举例

$$\left.\begin{aligned} p_1 &= 2p_2 + \left(\frac{5-1}{2} - 1\right) = 2p_2 + 1 \\ p_2 &= 3p_3 + \left(\frac{7-1}{2} - 0\right) = 3p_3 + 3 \end{aligned}\right\} \Rightarrow p_1 = 6p_3 + 7$$

因此获得了网络中任意两个特征图之间的坐标映射关系。何凯明在 SPP-Net 中采用一定的方法化简了公式，假设每一层的补零为

$$\text{padding} = \lfloor k_i/2 \rfloor \tag{5.33}$$

式中：$\lfloor \cdot \rfloor$ 表示向下取整。

则有

$$p_i = s_i \cdot p_{i+1} + ((k_i - 1)/2 - \lfloor k_i/2 \rfloor) \tag{5.34}$$

当 k_i 为奇数时，$(k_i-1)/2-\lfloor k_i/2 \rfloor = 0$，则有 $p_i = s_i \cdot p_{i+1}$；

当 k_i 为偶数时，$(k_i-1)/2-\lfloor k_i/2 \rfloor = -0.5$，则有 $p_i = s_i \cdot p_{i+1} - 0.5$。

由于 p_i 是坐标值，因此其值为整数，基本上可以认为 $p_i = s_i p_{i+1}$，那么感受野中心点坐标 p_i 只跟前一层 p_{i+1} 有关。

将式(5.31)进行串接，得到通用的解决方案，即

$$i_0 = g_L(i_L) = \alpha_L(i_L - 1) + \beta_L \tag{5.35}$$

式中：α_L 为所有 L 层的总步长，是所有步长的乘积；β_L 为式(5.31)的第二项串接以后的结果，则可以得到感受野坐标点的变换。并且有

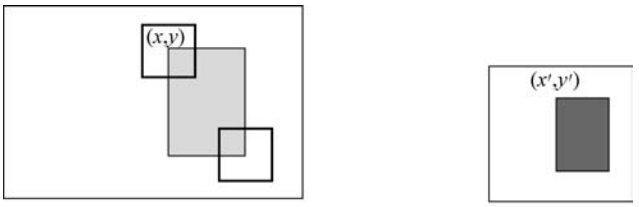
$$\alpha_L = \prod_{p=1}^{L} S_p$$

$$\beta_L = 1 + \sum_{p=1}^{L} \left(\prod_{q=1}^{p-1} S_q\right) \left(\frac{F_p - 1}{2} - P_p\right)$$

(2) 候选区域到特征图变换。

SPP-Net 是把原始感兴趣的区域(RoI)的左上角和右下角映射到特征图上的两个对应点，如图 5-52 所示，特征图上的两对角点就确定了对应的特征图区域(图 5-52(b))。

其映射方法为将左上角的点(x, y)映射到特征图上的(x', y')，使得(x', y')在原始图上感受野((a)图阴影框)的中心点与(x, y)尽可能接近。下面针对对应点之间的映射



(a) 感兴趣区域　　(b) 特征图

图 5-52　映射示意图

作具体说明。将前面每层都填充 $P/2$ 得到的简化公式 $p_i = s_i p_{i+1}$，然后把式进行级联得到 $p_0 = S p_{i+1}$，$S = \prod_0^i S_i$；对于特征图上的(x', y')，其在原始图的对应点为$(x, y) = (S_{x'}, S_{y'})$；最后把原始图片中的 RoI 映射为特征图中的映射区域((b)图阴影框)，则

左上角取：$x' = \lfloor x/S \rfloor + 1$；$y' = \lfloor y/S \rfloor + 1$

右下角取：$x' = \lfloor x/S \rfloor - 1$；$y' = \lfloor y/S \rfloor - 1$

综上，SPP-Net 通过上述两个方面的改进，即解决尺寸变换和 R-CNN 中重复卷积的问题，使得其网络速度提高到 R-CNN 的 20 多倍，并且没有牺牲检测精度(VOC07 mAP=59.2%)。SPP-Net 虽然有效地提高了检测速度，但仍然存在一些不足：一是多阶段训练过程步骤烦琐，仍然需要精调整预训练网络，然后对每个类别都训练 SVM 分类器，最后还要进行边界框回归，且区域候选也是要通过选择搜索来获得；二是 SPP-Net 只对其全连接层进行微调，而忽略了之前的所有层；三是仍然存在 R-CNN 时间和内存消耗比较大的问题。在训练 SVM 和回归的时候需要用网络训练的特征作为输入，特征保存和读取时间消耗较大。

4. Fast R-CNN

2015 年，R. Girshick 提出了 Fast R-CNN 检测器[26]，这是对 R-CNN 和 SPP-Net 的进一步改进。Fast R-CNN 使人们能够在相同的网络配置下同时训练检测器和边界框回归器。在 VOC07 数据集上，Fast R-CNN 将 mAP 从 58.5%(R-CNN)提高到 70.0%，在 VOC2012 上的 mAP 约为 66%。基于 VGG16 的 Fast R-CNN 算法在训练速度上比 R-CNN 快了将近 9 倍，比 SPP-Net 快大概 3 倍；检测速度比 R-CNN 快了 213 倍，比 SPP-Net 快了 10 倍。

图 5-53 给出了 Fast R-CNN 系统流程图，首先进行候选框提取，采用选择性搜索在图片中获得大约 2000 个候选框；然后进行 CNN 特征图提取与候选框 RoI 映射，并进行 RoI 池化。Fast R-CNN 在数据输入尺寸无限制，而实现这个无限制要求的关键所在正是 RoI 池化层，该层的作用是可以在任何大小的特征映射上为每个输入 RoI 提取固定的维度特征表示，然后确保每个区域的后续分类可以正常执行；最后将池化后的 RoI 进行特征提取，进行分类与回归。图 5-53 的简化流程图如图 5-54 所示。

需要说明的是，使用卷积网络提取图片特征。类似于 R-CNN，在获取特征映射之

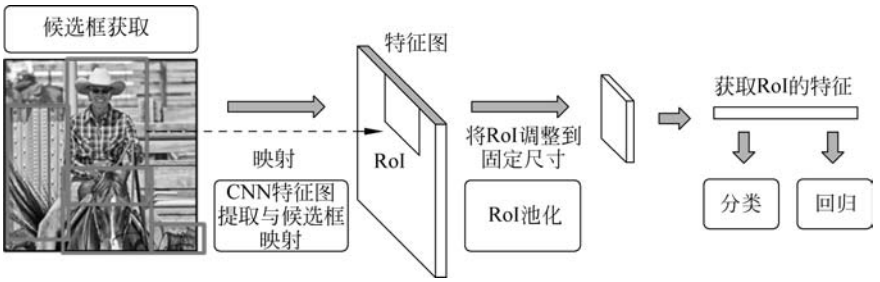


图 5-53　Fast R-CNN 系统流程图

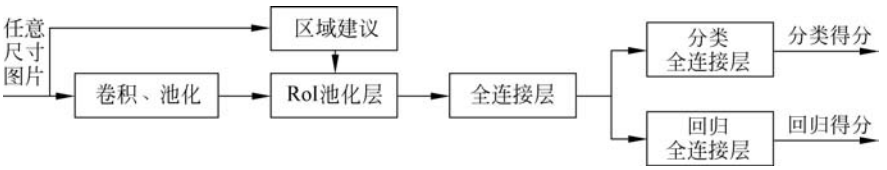


图 5-54　Fast R-CNN 简化流程图

后，需要卷积神经网络来进行卷积操作。在此处 Fast R-CNN 使用的卷积神经网络为普通的全链接层 fc_7，但是有所改动，也有使用 VGG16 的神经网络。

1）RoI 池化层

RoI 是指在选择性搜索完成后得到的“候选框”在特征图上的映射。RoI 池化层将每个候选区域分为 $m \times n$ 个块。针对每个块执行最大池化操作，使得特征映射上不同大小的候选区域被变换为均匀大小的特征向量，然后送入下一层。举例来说，某个 RoI 区域坐标为(x_1, y_1, x_2, y_2)，那么输入尺寸大小为$(y_2 - y_1) \times (x_2 - x_1)$；如果合并输出的大小为池高乘以池宽，即 Pooledheight * Pooledwidth，则每个网格的大小都为

$$\frac{y_2 - y_1}{\text{Pooledheight}} \times \frac{x_2 - x_1}{\text{Pooledwidth}} \tag{5.36}$$

虽然 RoI 池化可以看作是针对 RoI 的特征图像的池化操作，但存在非固定大小输入的问题，故每次池化网格的大小都必须手动计算。因此，首先用选择性搜索等候选框提取算法得到候选框坐标，然后，输入到卷积神经网络中来预测每个候选框中包含的对象。上述步骤中神经网络对 RoI 进行分类，并没有对整个图片进行分类。

在前面介绍的 R-CNN 中，进行卷积操作之前一般是先将图片分割与形变到固定尺寸，这也正是 R-CNN 的缺点之一。因为这会让图像产生形变，或者图像变得过小，使一些特征产生损失，继而对之后的特征选择产生巨大影响。Fast R-CNN 与 R-CNN 不同，Fast R-CNN 在数据的输入上并不对其有什么限制，而实现的关键正是 RoI 池化层。RoI 池化层如图 5-55 所示。该层的作用是可以在任何大小的特征映射上为每个输入 RoI 提取固定维度的特征表示，确保每个区域的后续分类可以正常执行。

对比图 5-49 所示的 SPP-Net 可以看出，RoI 的尺度可以是 4×4、2×2、3×3 等，可以动态调整。不像 SPP 是固定的 4×4、2×2 和 1×1。而 RoI 为单尺度的原因是进行了准

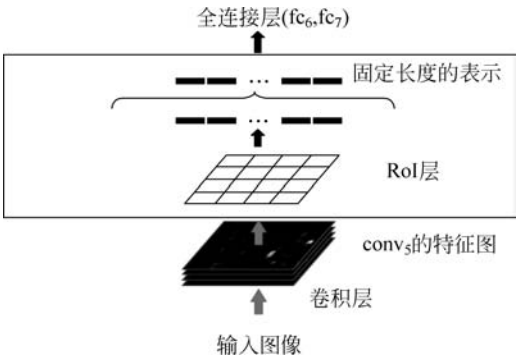


图 5-55　RoI 池化层

确度和时间的折中。通常,单尺度是直接将图像定为某种尺寸,直接输入网络来训练即可;多尺度则要生成一个金字塔,然后在金字塔上找到一个大小与目标比较接近 227×227 的投影版本。与 RoI 池化比,SPP-Net 更准确一些,但是 RoI 时间要省很多,所以实际中大都采用 RoI。Fast R-CNN 比 SPP-Net 快很多原因也在此。

2) 联合多任务建模——联合候选框回归与目标分类

在进行 RoI 池化后就可以直接进行深度图像检测,此时神经网络仍然只是对图像进行分类,只不过不是整幅图像分类,而是 RoI 分类。在 Fast R-CNN 中,输出层不仅仅完成简单分类,而是将候选框目标分类与候选框回归并列放入全连接层,设置两个输出层,形成一个多任务训练模型。

第一个是针对每个 RoI 的分类概率预测,即获得 $\boldsymbol{P}=(p_0,p_1,\cdots,p_K)$,表示属于 K 类和背景的概率(其中 K 为总类别数),用 $L_{\text{cls}}(\boldsymbol{P},u)$表示分类误差,则有

$$L_{\text{cls}}(\boldsymbol{P},u)=-\log p_u \tag{5.37}$$

式中:u 为该 RoI 的真实类别标签。

第二个是候选框回归损失函数,即针对每个 RoI 坐标偏移优化。假设 RoI 中 $t^k=(t_x^k,t_y^k,t_w^k,t_h^k)(0\leqslant k\leqslant K,k$ 是多类检测的类别序号),对于类别 u,在图片中标注了一个真实坐标 $t^u=(t_x^u,t_y^u,t_w^u,t_h^u)$,而预测值 $t=(t_x,t_y,t_w,t_h)$,二者理论上越接近越好,则将损失函数定义为

$$L_{\text{loc}}(t,t^u)=\sum_{i\in\{x,y,w,h\}}\text{smooth}_{L_1}(t_i,t_i^u) \tag{5.38}$$

式中

$$\text{smooth}_{L_1}(x,x')=\begin{cases}0.5(x-x')^2, & |x-x'|\leqslant 1\\ |x-x'|-0.5, & \text{其他}\end{cases} \tag{5.39}$$

式中:smooth 函数在(−1,1)之间为二次函数,而其他区域为线性函数,该函数可以增强模型对异常数据的鲁棒性,其具体形式如图 5-56 所示。

总损失为上述两个任务单的损失函数和,即

$$L(P,u,t^u,t)=L_{\text{cls}}(\boldsymbol{P},u)+\lambda\lceil u\geqslant 1\rceil L_{\text{loc}}(t^u,t) \tag{5.40}$$

式中：λ 为常数；$\lceil u \geqslant 1 \rceil$ 为指示函数，当 $u \geqslant 1$ 时值为 1，即表示当为前景时，考虑两种损失，当 $u=0$ 时，即图像为背景时，只考虑分类损失函数。

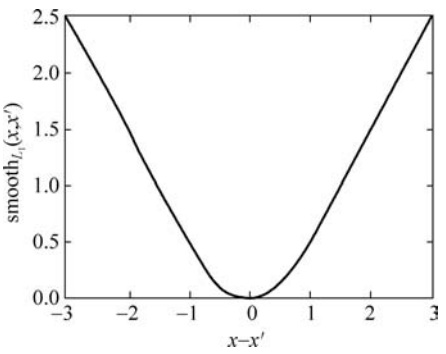


图 5-56　Smooth 函数

3）训练和测试

首先用 ILSVRC 20XX 数据集进行预训练，预训练是进行有监督分类的训练。然后在 PASCAL VOC 样本上进行特定调优，调优的数据集包括 25% 的正样本和 75% 的负样本。其中正样本是指真实框 IoU 在 0.5～1 的候选框，而负样本是指与真实框 IoU 在 0.1～0.5 的候选框。PASCAL VOC 数据集中既有物体类别标签，也有物体位置标签，有 20 种物体；正样本仅表示前景，负样本仅表示背景；回归操作仅针对正样本进行。在调优训练时，每一个 mini-batch 中首先加入 N 张完整图片，而后加入从 N 张图片中选取 R 个候选框。R 个候选框可以复用 N 张图片前 5 个阶段的网络特征，例如 $N=2$，$R=128$。微调前，需要对有监督预训练后的模型进行三步转化：一是 RoI 池化层取代有监督预训练后的 VGG-16 网络最后一层池化层。二是两个并行层取代上述 VGG-16 网络的最后一层全连接层和 Softmax 层。并行层之一是新全连接层 1，即将原 Softmax 层 1000 个分类输出修改为 21 个分类输出，表示 20 种类＋背景类，并行层之二是新全连接层 2＋候选区域窗口回归层。三是上述网络由原来单输入（一系列图像）修改为双输入（一系列图像和这些图像中的一系列候选区域）。

测试流程图如图 5-57 所示，即输入图片经过 5 个卷积池化块操作后，经过 2 个全连接层获得 4096 维度的向量，然后计算分类损失和回归损失，最后按照总损失进行输出。

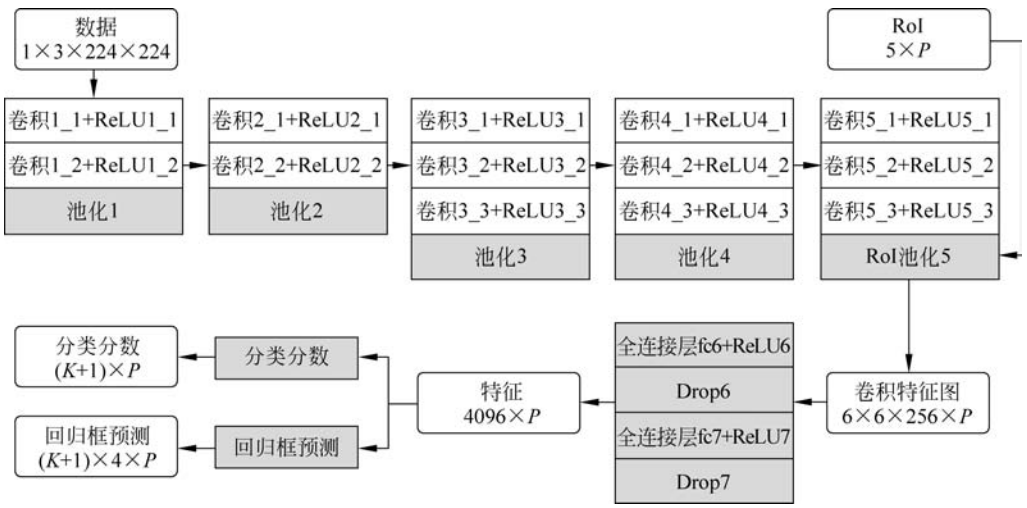


图 5-57　测试流程图

4) SVD 全连接层加速网络

图像分类任务中，卷积层比全连接层计算时间长；而在目标检测任务中，选择性搜索算法提取的建议框数目巨大(约 2000 个)，几乎有一半的前向计算时间花费在全连接层。就 Fast R-CNN 而言，由于每个建议框都需要计算分类结果，因此 RoI 池化层后的全连接层需要进行约 2000 次，由此考虑在 Fast R-CNN 中采用 SVD 分解加速全连接层计算。具体实现如下：

(1) 物体分类和窗口回归都是通过全连接层实现，假设全连接层输入数据为 $\boldsymbol{x}$，输出数据为 $\boldsymbol{y}$，全连接层参数为 $\boldsymbol{W}$，尺寸为 $u\times v$，那么该层全连接计算为

$$\boldsymbol{y}=\boldsymbol{W}\boldsymbol{x} \tag{5.41}$$

计算复杂度为 $u\times v$。

(2) 若将 W 进行 SVD 分解，并用前 t 个特征值近似代替，即

$$\boldsymbol{W}=\boldsymbol{U}\Sigma\boldsymbol{V}^{\mathrm{T}}\approx\boldsymbol{U}(u,1:t)\cdot\Sigma(1:t,1:t)\cdot\boldsymbol{V}(v,1:t)^{\mathrm{T}} \tag{5.42}$$

那么原来的前向传播分解成两步，即

$$\boldsymbol{y}=\boldsymbol{W}\boldsymbol{x}=\boldsymbol{U}(\Sigma\boldsymbol{V}^{\mathrm{T}})\boldsymbol{x}=\boldsymbol{U}\cdot\boldsymbol{z} \tag{5.43}$$

其具体实现过程如图 5-58 所示。计算复杂度为 $u\times t+v\times t$，若 $t<\min(u,v)$，则这种分解会大大减少计算量。

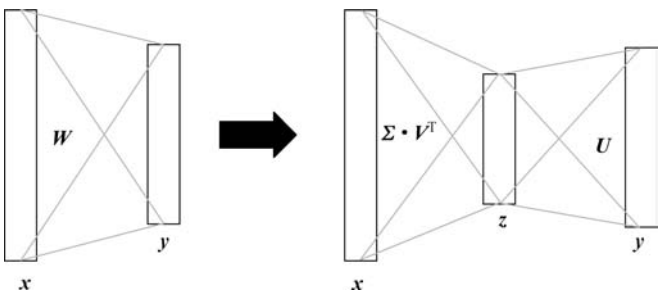


图 5-58　SVD 全连接层分解网络

实现时，相当于把一个全连接层拆分为两个全连接层，第一个全连接层不含偏置，第二个全连接层含偏置。实验表明，SVD 分解全连接层能使 mAP 只下降 0.3%的情况下提升 30%的速度，同时该方法也不必再执行额外的微调操作。

5) 图片中心化采样

R-CNN 和 SPP-Net 中采用 RoI 中心采样，即从全部图片的所有候选区域中均匀取样，这样每个 SGD 的 mini-batch 中包含了不同图像的样本，不同图像之间不能共享卷积计算和内存，运算开销大。Fast R-CNN 中采用图片中心化采样：mini-batch 采用层次采样，即先对图像采样(N 个)，再在采样到的图像中对候选区域采样(每个图像中采样 R/N 个，一个 mini-batch 共计 R 个候选区域样本)，同一图像的候选区域卷积共享计算和内存，降低了运算开销。

图片中心化方式采样的候选区域来自同一图像，相互之间存在相关性，可能会减慢训练收敛的速度，但是在实验中并没有明显的表现，反而使用 $N=2$，$R=128$ 的图片中心

化方式比 R-CNN 收敛更快。

这里解释为什么 SPP-Net 不能更新空间金字塔池化层前面的卷积层，而只能更新后面的全连接层：一种解释卷积特征是线下计算的，无法在微调阶段反向传播误差；另一种解释反向传播需要计算每一个 RoI 感受野的卷积层梯度，通常所有 RoI 会覆盖整个图像，如果用 RoI-中心采样方式，会由于计算太多整幅图像梯度而变得又慢又耗内存。

5. Faster R-CNN 网络

Faster R-CNN 网络[27]是 Ross B. Girshick 在 R-CNN 和 Fast R-CNN 基础上于 2016 年提出的网络。从结构上看，Faster R-CNN 用神经网络生成待检测框，替代了其他 R-CNN 算法通过规则等产生候选框的方法，且将特征抽取、候选框提取、候选框回归和分类都整合在一个网络中实现端到端训练，使得网络的综合性更强，尤其是检测速度得以大幅改善。该网络主要包括骨干网络、区域候选网络（Region Proposal Networks，RPN）、RoI 池化与分类三个部分。Faster R-CNN 的示意图和具体流程分别如图 5-59 和图 5-60 所示。其中，骨干网络由经典网络共享基础卷积层而构成，用于提取整张图片的特征，例如 ZFNet、VGG16 或 ResNet101 等网络去除后面全连接层构成主干网络。主干网络输出下采样后的特征图，送入后续区域候选网络和全连接层进行处理。RPN 用于生成区域候选框，该网络通

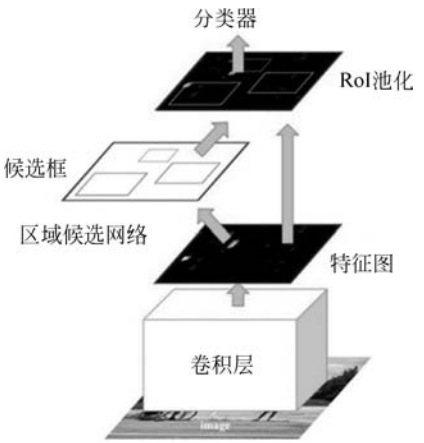


图 5-59　Faster R-CNN 基本结构

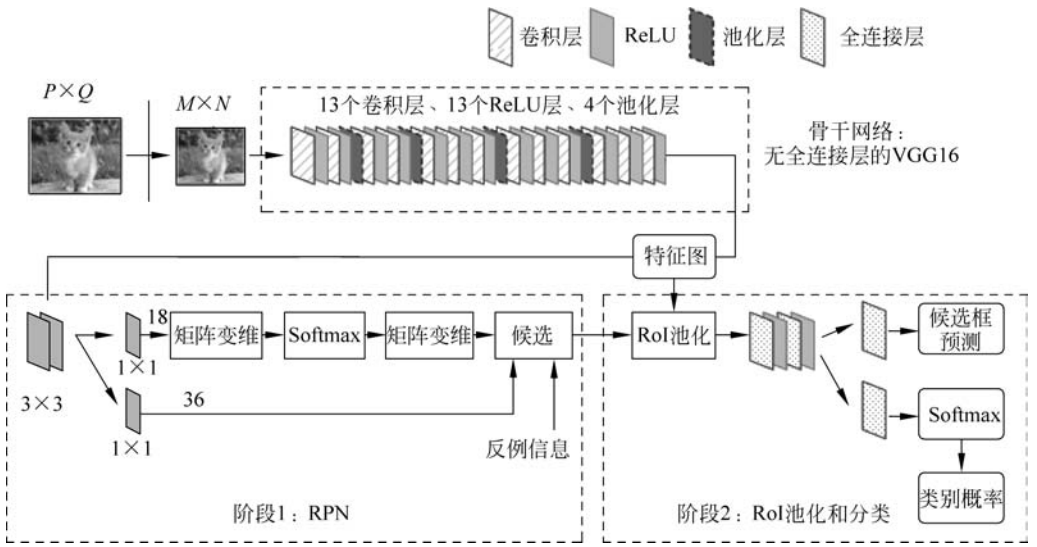


图 5-60　Faster R-CNN 具体流程图

过 Softmax 函数判断锚框属于正类或者负类，再利用候选框回归算法来修正锚框获得精确的候选区域。RoI 池化与分类网络对候选检测框进行分类，并且再次微调候选框坐标。RoI 池化收集输入的特征图和候选区域进行特征提取，分类层利用候选框特征图计算候选框的类别，同时利用候选框回归算法获得检测框最终的精确位置。

1）骨干网络

骨干网络可采用去掉全连接层后 ZFNet、VGG16 或 ResNet101 网络等不同类型，实验中采用了 ZFNet 和 VGG16 两种模型。骨干网络 VGG16 共有 13 个卷积层和 4 个池化层，其中卷积层后过 ReLU 激活函数。骨干网络中所有设置相同，即卷积层尺寸为 3×3、补零尺寸为 1，步长为 1。这意味着在 Faster R-CNN 骨干网络对所有的卷积都做了扩边处理（pad=1，即填充一圈 0），导致原图尺寸变为 $(M+2)\times(N+2)$，再做 3×3 卷积后输出 $M\times N$，如图 5-61 所示。正是这种设置，使得骨干网络中的卷积层不改变输入和输出矩阵大小。类似地，池化层尺寸为 2×2、没有补零，步长为 2。这样每个经过池化层的 $M\times N$ 矩阵，都会变为 $(M/2)\times(N/2)$ 大小。综上所述，骨干网络的卷积层不改变输入与输出大小，池化层使输出长和宽都变为输入的 1/2。因此，对于 VGG16 骨干网络而言，由于有 4 个池化层，一个 $M\times N$ 大小的矩阵经过骨干网络后固定变为 $(M/16)\times(N/16)$。而对于 ZFNet 骨干网络，由于有三个池化层，则一个 $M\times N$ 大小的矩阵经过骨干网络后固定变为 $(M/8)\times(N/8)$。

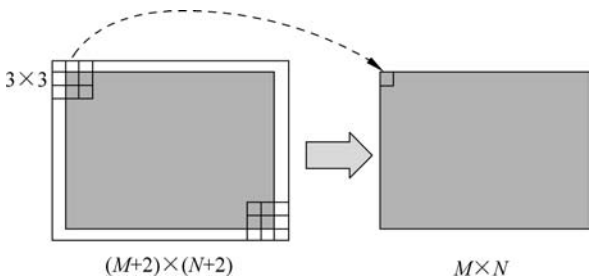


图 5-61　卷积示意图

2）区域候选网络

经典的检测方法生成检测框非常耗时，如 R-CNN 使用选择性搜索方法生成检测框，还有采用滑动窗口方法。而 Faster R-CNN 则抛弃了前面方法，直接使用区域候选网络生成检测框，这也是 Faster R-CNN 的巨大优势，能极大提升检测框的生成速度。从图 5-62 所示可以看出，RPN 实际分为两个路径：一是通过 Softmax 分类函数将锚框分成正类和负类；二是通过计算对于锚框的边界框回归偏移量，获得精确的候选区域。而最后的候选层则负责综合正类锚框和对应边界框回归偏移量获取候选区域，同时剔除太小和超出边界的区域框。其具体处理过程如图 5-62 所示。

经过 RPN 后完成目标的定位。图 5-63 给出了具体的示意图。由图中可以看出，假设锚框个数 $k=9$，假设骨干网络输出的特征图尺寸是 $W\times H\times 512$，经过一个尺寸为 3×3 的滑动窗卷积层，再通过两个参数个数完全相同但不共享参数的两个 1×1 的卷积层（可

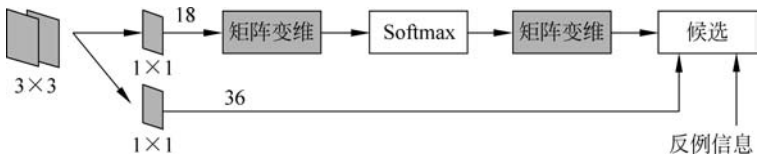


图 5-62　阶段 1：RPN

称为姐妹卷积层)，输出两个特征图，一个为分数特征图 $W\times H\times 2k$，另一个为坐标特征图 $W\times H\times 4k$。实验中主干网络采用了 ZFNet 和 VGGNet 两种配置，而在 ZFNet 和 VGGNet 中最后的卷积层 5 的输出维度分别为 256 和 512，因此经过两种不同主干网络后的特征维度分别为 256 或 512。在卷积层 5 之后，做了 RPN 的 3×3 卷积又融合了周围 3×3 的空间信息，同时保持维度 256 或 512 不变。假设在卷积层 5 的每个点上都有 k 个锚框(默认 $k=9$)，而每个锚框要分正例和负例，所以每个点由 256 维矢量转化为 $2k$ 分类结果；而每个锚框都有(x,y,w,h)对应 4 个偏移量，所以回归参数有 $4k$ 个坐标。全部锚框进行训练导致数据规模巨大，因此随机选取 128 个正类锚框和 128 个负类锚框进行训练。

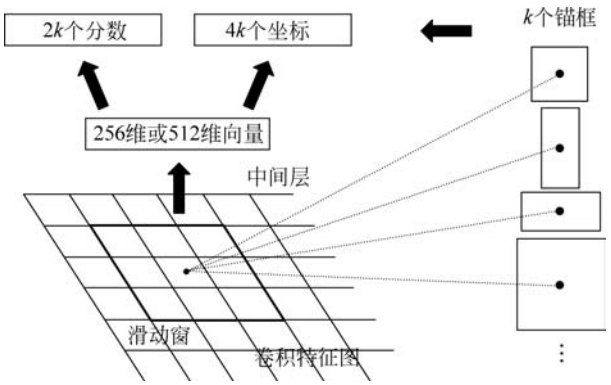


图 5-63　Faster R-CNN 细节信息展示

(1) 锚框定义。

锚框实际上就是一组由向量生成的矩形，如其中每行的 4 个值(x_1,y_1,x_2,y_2)表示矩形左上和右下角点坐标。锚框尺寸根据检测图像设置。例如，可以将任意大小的输入图像矩阵变形为 800×600(即图 5-60 中的 $M=800,N=600$)。下面计算下锚框的个数，由于原图 800×600，VGG 下采样 16 倍后，特征图中每个点设置 9 个锚框，所以有

$$\text{ceil}\lceil 800/16\rceil\times\text{ceil}\lceil 600/16\rceil\times 9=50\times 38\times 9$$

式中：ceil$\lceil\cdot\rceil$表示向上取整。因此，VGG 输出的特征图尺寸为 50×38。输出特征图的每个特征点映射回原图感受野的中心点当成一个基准点，然后围绕基准点选取 k 个不同尺度、不同比例的锚框。图 5-64 中采用了 3 个尺度 3 种面积的锚框，面积分别为$\{128^2、256^2、512^2\}$，比例尺寸分别为{1∶1,1∶2,2∶1}。9 个矩形共有 3 种形状，长宽比约为{1∶1,1∶2,2∶1}三种。实际上，通过锚框引入了检测中常用的多尺度方法。图 5-65 给

出一组锚框坐标示例。

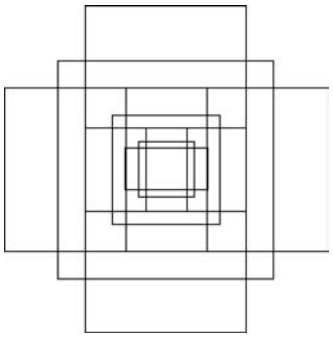


图 5-64　锚框示意图

```
[[-84. -40. 99. 55.]
[-176. -88. 191. 103.]
[-360. -184. 375. 199.]
[-56. -56. 71. 71.]
[-120. -120. 135. 135.]
[-248. -248. 263. 263.]
[-36. -80. 51. 95.]
[-80. -168. 95. 183.]
[-168. -344. 183. 359.]]
```

图 5-65　锚框坐标

(2) 矩阵变形。

从图 5-60 可以看出,将 1×1 卷积后的张量先矩阵变形,便于使用 Softmax 层计算,之后再矩阵变形回来。一幅 $M\times N$ 大小的矩阵送入 VGGNet 主干网络后,到 RPN 前变为$(M/16)\times(N/16)$,不妨设宽度 $W=M/16$ 和高度 $H=N/16$。在进入矩阵变形与 Softmax 函数之前,先做了 1×1 卷积,而 1×1 卷积的输出通道数为 18,因此经过该卷积后,该卷积的输出图像尺寸为 $W\times H\times 18$,通道数 18 为 9 个锚框可能出现正类和负类的所有情况相对应。后面按 Softmax 分类获得正类锚框,初步提取了检测目标候选区域边框信息。

在进行 Softmax 前后都要经过矩阵变形主要是因为便于进行二分类。其主要原因是:在一些深度学习框架中,数据存储的格式是 $W\times H\times \text{Channel}=W\times H\times(2\times k)$,这里的通道数 $\text{Channel}=2\times k$,即 2 倍的锚框个数。而在 Softmax 部分要进行正、负二分类,因此会将利用矩阵变形(Reshape 层)将其变成 $W\times H\times \text{Channel}=W\times(H\times k)\times(2)$,就是将某个维度变成二维进行 Softmax 分类,而分类后还要恢复成原来的存储形式。

(3) 锚框正类与负类划分。

不是所有锚框都被选择来进行训练,若对每幅图的所有锚框都去优化损失函数,那么会因为负样本过多导致最终得到的模型对正样本预测准确率很低,因此需要对锚框的正类和负类进行选择。假设训练集中的每张图像(含有人工标定的真实框)的所有锚框($N\times M\times k$),则正、负样本的划分规则如下:

① 对每个标定的真实框区域,与其 IoU 最大的锚框记为正样本,保证每个真实框至少对应一个正样本锚框。

② 剩余的锚框,若其与某个标定真实框区域 IoU 大于 0.7,则记为正样本;若其与任意一个标定真实框的 IoU 都小于 0.3,则记为负样本。因此,每个真实标定框可能会对应多个正样本锚框,但每个正样本锚框只能对应一个真实标定框。

③ 前两步还剩余的锚框舍弃不用。

④ 跨越图像边界的锚框舍弃不用。

锚框正类和负类选择如图 5-66 所示。

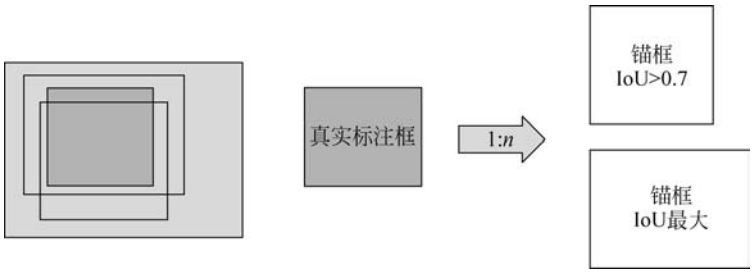


图 5-66　锚框正类和负类选择示意图(一个真实标注框可对应多个锚框)

除了进行上述选择外,锚框的数目也通过下面方式进行压缩:一是对每一个预测为正类锚框的分数进行排序,然后采用非极大值抑制方法,即首先找到分数最高的锚框,然后以此锚框作为基准,计算其他锚框与该锚框的 IoU,如果 IoU 大于阈值(如 0.7),就认为是同一个目标,直接舍弃这个锚框,每个图像剩下的锚框数为 A;二是再次选择,由于 A 个锚框中依然有很多副样本,通过 IoU(大于或等于 0.5 为正样本,否则为负样本),最后选择 B 个正、负样本。

下面通过一个具体例子来说明选择的大致基准。假设原始图片尺寸为 1000×600,那么经过骨干 VGG16 网络下采样后,则特征图尺寸变为(1000/16)×(600/16)×512,即下采样 16 倍,且通道数为 512;经过 RPN 后则变成 63×38×9=21546 个锚框。经过去除跨越图像边界的部分,则剩余 6000 个左右的锚框;然后利用非极大值抑制方法,每幅图像筛选剩下 2000 个左右的锚框;最后再次选择剩余 256 左右的锚框。

3) 边界框回归算法

假设有预测框、锚框和真实标签框三种类型的框,且 x、x_a 和 x^* 分别是 x 的预测值、锚框和真实标签对应值,变量 y、w 和 h 同理。x、y、w、h 分别是预测矩形框的中心点坐标及宽和高,x_a、y_a、w_a、h_a 分别是锚框的中心坐标及宽和高,x^*、y^*、w^*、h^* 分别是真实标签对应框的中心坐标及宽和高。则有预测框到锚框的变换表示为

$$\begin{cases} x = x_a + \nabla x \\ y = y_a + \nabla y \\ w = w_a \cdot \nabla w \\ h = h_a \cdot \nabla h \end{cases} \tag{5.44}$$

而∇部分都是基于 w、h 计算出来的,则有

$$\begin{cases} \nabla x = w_a \cdot t_x \\ \nabla y = h_a \cdot t_y \\ \nabla w = w_a \cdot e^{t_w} \\ \nabla h = h_a \cdot e^{t_h} \end{cases} \tag{5.45}$$

式中：$\{t_x, t_y, t_w, t_h\}$为预测框(x, y, w, h)对于锚框(x_a, y_a, w_a, h_a)的偏移量。

根据式(5.44)和式(5.45)，可得

$$\begin{cases} t_x = (x - x_a)/w_a \\ t_y = (y - y_a)/h_a \\ t_w = \log(w/w_a) \\ t_h = \log(h/h_a) \end{cases} \tag{5.46}$$

而$\{t_x^*, t_y^*, t_w^*, t_h^*\}$表示真实标记框$(x^*, y^*, w^*, h^*)$相对于锚框$(x_a, y_a, w_a, h_a)$的偏移量，即

$$\begin{cases} t_x^* = (x^* - x_a)/w_a \\ t_y^* = (y^* - y_a)/h_a \\ t_w^* = \log(w^*/w_a) \\ t_h^* = \log(h^*/h_a) \end{cases} \tag{5.47}$$

学习目标自然就是让$\{t_x, t_y, t_w, t_h\}$接近$\{t_x^*, t_y^*, t_w^*, t_h^*\}$的值。

上述算法中用 exp 函数和 log 函数，这是因为尺度系数大于零，因此 exp 函数来表示尺度系数可以满足。而利用 log 函数的主要原因是对数函数具有一个很重要的特性——非线性压缩，即压缩较大值，因此采用这种方式计算的损失函数可以减小大的目标的作用，有助于改善模型只能学到大目标的缺点。从式(5.46)和式(5.47)可以看出，对于中心点坐标采用线性变换，而对于宽度和高度采用非线性变换，这种设置也是合理的，对于 log 函数，当自变量较小时可近似为线性变换。

4) 损失函数

Faster R-CNN 的损失主要分为 RPN 的损失和 Fast R-CNN 的损失，并且两部分损失都包括分类损失和回归损失，其总损失函数如下：

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{\text{cls}}} \sum_i L_{\text{cls}}(p_i, p_i^*) + \lambda \frac{1}{N_{\text{reg}}} \sum_i p_i^* L_{\text{reg}}(t_i, t_i^*) \tag{5.48}$$

式中：$\frac{1}{N_{\text{cls}}} \sum_i L_{\text{cls}}(p_i, p_i^*)$为分类损失，包括 RPN 分类损失和 Fast R-CNN 分类损失；$\frac{1}{N_{\text{reg}}} \sum_i p_i^* L_{\text{reg}}(t_i, t_i^*)$为回归损失，也包括 RPN 回归和 Fast R-CNN 回归损失。

(1) 分类损失。

① RPN 分类损失：RPN 的产生的锚框只分为前景和背景，前景的标签为 1，背景的标签为 0。在训练 RPN 的过程中，会选择 256 个锚框，即 $N_{\text{cls}} = 256$。真实标注框标签为

$$p_i^* = \begin{cases} 0, & \text{负样本} \\ 1, & \text{正样本} \end{cases} \tag{5.49}$$

$L_{\text{cls}}(p_i, p_i^*)$是两个类别的对数损失，即

$$L_{\text{cls}}(p_i, p_i^*) = -\log[p_i^* p_i + (1 - p_i^*)(1 - p_i)] \tag{5.50}$$

式中：p_i 为锚框预测为目标的概率。

式(5.50)是经典二分类交叉熵损失，对于每一个锚框计算对数损失，然后求和除以总的锚框数量 N_{cls}。

② Fast R-CNN 分类损失：RPN 的分类损失时二分类的交叉熵损失，而 Fast R-CNN 是多分类的交叉熵损失。在 Fast R-CNN 的训练过程中会选出 N_{cls} 个进行计算。

(2) 回归损失。

这里将RPN和Fast R-CNN的回归损失一起来描述，即 $\lambda \frac{1}{N_{\text{reg}}} \sum_i p_i^* L_{\text{reg}}(t_i, t_i^*)$ 部分。从前面可以知道，$\{t_x, t_y, t_w, t_h\}$ 是预测框 (x, y, w, h) 对于锚框 (x_a, y_a, w_a, h_a) 的偏移量；$\{t_x^*, t_y^*, t_w^*, t_h^*\}$ 表示真实标记框 (x^*, y^*, w^*, h^*) 相对于锚框 (x_a, y_a, w_a, h_a) 的偏移量，则有

$$L_{\text{reg}}(t_i, t_i^*) = \tilde{\text{s}}\text{mooth}_{L_1}(t_i, t_i^*) \tag{5.51}$$

式中：$\tilde{\text{s}}\text{mooth}_{L_1}(t_i, t_i^*)$ 是式(5.39)所定义的平滑 L_1 函数的拓展，即

$$\tilde{\text{s}}\text{mooth}_{L_1}(x, x') = \begin{cases} 0.5(x - x')^2 \times 1/\sigma^2, & |x - x'| \leqslant 1/\sigma^2 \\ |x - x'| - 0.5, & \text{其他} \end{cases} \tag{5.52}$$

不同之处在于：RPN 训练中 $\sigma = 3$，而原始 Fast RCNN 中 $\sigma = 1$。对于每一个锚框计算完 $L_{\text{reg}}(t_i, t_i^*)$ 后还要乘以 p_i^*，如前所述，p_i^* 为正类时值为 1，说明里面有物体时，p_i^* 为负类时值为 0，说明没有物体。这意味着，只有前景才计算损失，背景不计算损失。

对于参数 λ 和 N_{reg} 在 RPN 训练过程中的设置进行解释说明。N_{reg} 为输出特征图尺寸，即经过主干网络下采样后的 $W \times H = 63 \times 38 \approx 2400$，而 $\lambda = 10$，这样分类和回归两个损失函数的权重基本相同，即 $N_{\text{cls}} = 256$，而 $N_{\text{reg}}/\lambda \approx 240$，二者接近。而代码实现中，往往直接将 N_{reg} 取值为正、负样本的总数 256(批处理量)，然后将 $\lambda = 1$，这样直接使得分类和回归的权重相同，均为 1/batch_size，因此不需要额外设置 λ。

5) 候选层

候选层负责综合预测框和前景锚框，计算出精准的候选，送入后续 RoI 池化层。候选层有 3 个输入，分别是前景/背景分类器结果、预测的边界框以及 im_info 信息；另外，还有参数 feat_stride = 16。首先解释 im_info，对于一幅任意大小 $P \times Q$ 图像，传入 Faster R-CNN 前首先矩阵变形到固定 $M \times N$，im_info = [M, N, scale_factor]则保存了此次缩放的所有信息；然后经过卷积层，经过 4 次池化尺寸变为 $W \times H = (M/16) \times (N/16)$大小，其中 feature_stride = 16 则保存了该信息，用于计算锚框偏移量。

候选层按照以下顺序依次处理：

(1) 生成锚框，利用预测框对所有的锚框做边界框回归；

(2) 按照输入的前景框分数由大到小对锚框进行排序，提取前面若干锚框，即提取修正位置后的前景锚框；

(3) 限定超出图像边界的前景锚框为图像边界；

(4) 剔除非常小(宽度和高度小于阈值)的前景锚框；

(5) 进行非极大值抑制后处理，再次按照前景锚框分数由大到小排序，提取前若干结

果作为候选输出。之后输出候选框坐标$[x_1, y_1, x_2, y_2]$。注意，由于在第三步中将锚框映射回原图判断是否超出边界，所以这里输出的锚框是对应 $M \times N$ 输入图像尺度。

6）Faster R-CNN 训练

Faster R-CNN 训练是在已经训练好的模型（如 ZFNet、VGG16 和 Res101）的基础上继续进行训练。实际中训练步骤如下：

步骤 1：在已经训练好的 ImageNet 模型上训练 RPN，利用训练好的 RPN 收集候选框；

步骤 2：利用已有的 ImageNet 模型，第一次训练 Faster R-CNN；

步骤 3：利用 Faster R-CNN 参数，第二次训练 RPN；

步骤 4：再次利用步骤 3 中二次训练好的 RPN 收集候选框，第二次训练 Faster R-CNN。

训练过程类似于一种“迭代”的过程，循环了 2 次。通过实验发现，即使迭代循环更多次也不会进一步提升。图 5-67 给出了 Faster R-CNN 训练过程。

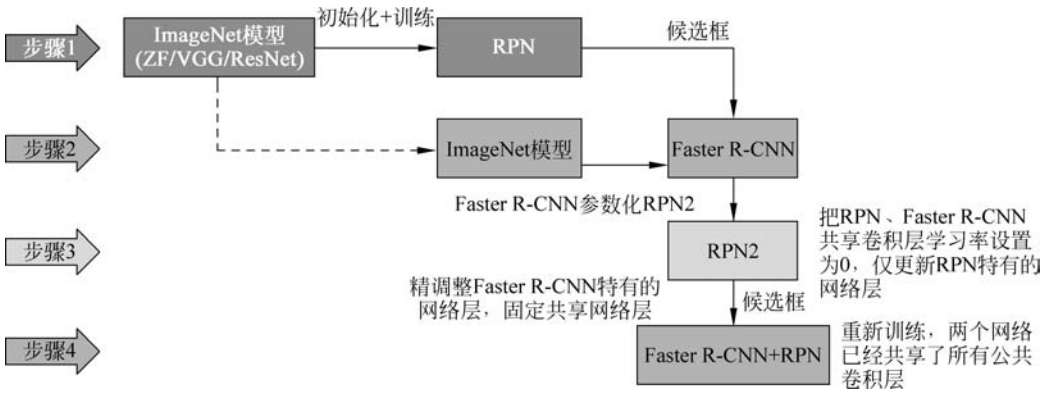


图 5-67　Faster R-CNN 训练过程

5.4　本章小结

卷积神经网络是一类非常重要的神经网络，在深度学习发展历史上具有重要地位。本章系统介绍了卷积神经网络的基本概念、基本原理和关键技术，并选取典型的 CNN 网络 LeNet5、AlexNet、ZFNet、VGGNet、GoogLeNet、ResNet、区域卷积神经网络进行了详细分析与解读。

参考文献

[1] Kandel E R. An introduction to the work of David Hubel and Torsten Wiesel[J]. The Journal of Physiology, 2010, 587(12): 2733-2741.

[2] Fukushima K, Miyake S. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position[J]. Pattern Recognition, 1982, 15(6): 455-469.

[3] Waibel A. Phoneme recognition using time-delay neural networks[C]. Meeting of the Institute of Electrical, Information and Communication Engineers(IEICE), 1987.

[4] Waibel A, Hanazawa T, Hinton G, et al. Phoneme recognition using time-delay neural networks [J]. IEEE transactions on acoustics, speech, and signal processing, 1989, 37(3): 328-339.

[5] Zhang W. Shift-invariant pattern recognition neural network and its optical architecture[C]. In Proceedings of annual conference of the Japan Society of Applied Physics(JSAP), 1988.

[6] LeCun Y, Boser B, Denker J S, et al. Backpropagation applied to handwritten zip code recognition [J]. Neural computation, 1989, 1(4): 541-551.

[7] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[8] Simard P Y, Steinkrau D, Platt J C. Best practices for convolutional neural networks applied to visual document analysis[C]. In: Proceedings of IAPR International Conference on Document Analysis and Recognition(ICDAR), 2003, 3: 958-962.

[9] Krizhevsky A, Sutskever I, Hinton G E. ImageNet Classification with Deep Convolutional Neural Networks [J]. Communications of the ACM, 2012, 60: 84-90.

[10] Zeiler M D, Fergus R. Visualizing and Understanding Convolutional Neural Networks[C]. In: Proceedings of European Conference on Computer Vision(ECCV), 2014: 818-833.

[11] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition[C]. In: Proceedings of International Conference on Learning Representations(ICLR), 2015: 1-14.

[12] Szegedy C, Liu W, Jia Y, et al. Going Deeper with Convolutions[C]. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2015: 1-9.

[13] He K M, Zhang X Y, Ren S Q, et al. Deep Residual Learning for Image Recognition[C]. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2016: 770-778.

[14] 我爱机器学习. Deep Learning 回顾之 LeNet、AlexNet、GoogLeNet、VGG、ResNet [EB/OL]. https://www.cnblogs.com/52machinelearning/p/5821591.html.

[15] Zeiler M D, Taylor G W, Fergus R. Adaptive deconvolutional networks for mid and high level feature learning[C]. In: Proceedings of IEEE International Conference on Computer Vision (ICCV), 2011: 2018-2025.

[16] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception Architecture for Computer Vision[C]. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016: 2818-2826.

[17] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[C]. In: Proceedings of International Conference on Machine Learning (ICML), 2015: 448-456.

[18] Feature Extractor [inception v2 v3][EB/OL]. https://www.cnblogs.com/shouhuxianjian/p/7756192.html.

[19] Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning[C]. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2016: 4278-4284.

[20] Hu J, Shen L, Albanie S, et al. Squeeze-and-Excitation Networks[C]. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2018: 7132-7141.

[21] Hu J, Shen L, Albanie S, et al. Squeeze-and-Excitation Networks[J]. IEEE Transactions on

Pattern Analysis and Machine Intelligence,2020,42(8): 2011-2023.

[22] Xie S,Girshick R,Dollar P,et al. Aggregated residual transformations for deep neural networks [C]. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2017: 5987-5995.

[23] Howard A G,Zhu M,Chen B,et al. MobileNets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv: 1704.04861,2017.

[24] Zhang X,Zhou X,Lin M,et al. ShuffleNet: An extremely efficient convolutional neural network for mobile devices[C]. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition(CVPR),2018: 6848-6856.

[25] Girshick R,Donahue J,Darrell T,et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition(CVPR),2014. 580-587.

[26] Girshick R. Fast r-cnn[C]. In: Proceedings of the IEEE International Conference on Computer Vision(ICCV). 2015: 1440-1448.

[27] Ren S Q,He K M,Girshick R,et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2015,39: 1137-1149.

[28] Garcia-Garcia A,Orts-Escolano S,Oprea S,et al. A Review on Deep Learning Techniques Applied to Semantic Segmentation[J]. arXiv preprint arXiv: 1704.06857,2017.

[29] Felzenszwalb P F,Huttenlocher D P. Efficient graph-based image segmentation[J]. Internation Jounal of Computer vision,2004,59(2): 167-181.

[30] He K,Zhang X,Ren S,et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence,2014,37(9): 1904-1916.

[31] Simonyan K,Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition [J]. arXiv preprint arXiv: 1409.1556,2014.

本章知识点补充

知识点 1　Hebbian 原理

神经反射活动的持续与重复会导致神经元连接稳定性的持久提升，当两个神经元细胞 A 和 B 距离很近，并且 A 参与了对 B 重复、持续的兴奋时，某些代谢变化会导致 A 将作为能使 B 兴奋的细胞。即“一起发射的神经元会连在一起”，学习过程中的刺激会使神经元间的突触强度增加。

受 Hebbian 原理启发，文章 *Provable Bounds for Learning Some Deep Representations* 提出，如果数据集的概率分布可以被一个很大、很稀疏的神经网络所表达，那么构筑这个网络的最佳方法是逐层构筑网络。具体思路是将上一层高度相关的节点聚类，并将聚类出来的每一个小簇连接到一起，如图 5-68 所示。这个相关性高的节点应该被连接在一起的结论，即是从神经网络的角度对 Hebbian 原理有效性的证明。

因此，一个“好”的稀疏结构应该符合 Hebbian 原理，应该把相关性高的一簇神经元

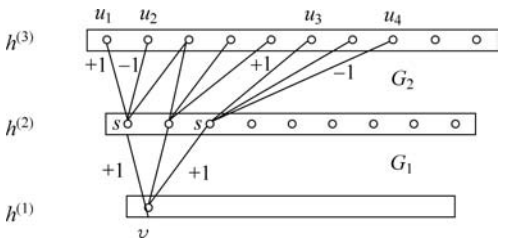


图 5-68　将高度相关的节点连接在一起形成稀疏网络

节点连接在一起。在普通的数据集中，这可能需要对神经元节点聚类，但是在图片数据中，邻近区域的数据相关性高，因此相邻的像素点被卷积操作连接在一起。而实际中可能有多个卷积核，在同一空间位置但在不同通道的卷积核的输出结果相关性极高。因此，1×1 的卷积就可以很自然地把这些相关性很高的、在同一个空间位置但是不同通道的特征连接在一起。这就是 1×1 卷积频繁应用于 GoogLeNet 中的原因。1×1 卷积所连接的节点的相关性最高，而稍大尺寸的卷积，如 3×3、5×5 卷积所连接的节点相关性也很高，因此也可以适当地使用一些大尺寸的卷积，增加多样性。深度感知模块通过 4 个分支中不同尺寸的 1×1、3×3、5×5 等小型卷积将相关性很高的节点连接在一起，就完成了其设计初衷，构建出了很高效的符合 Hebbian 原理的稀疏结构。

知识点 2　交并比

在目标检测当中，有一个重要的概念就是交并比，一般是指代模型预测的候选框和真实标注框之间的交并比。交并比的定义为

$$\mathrm{IoU}=\frac{A\cap B}{A\cup B}$$

集合 A 和集合 B 的并集包括了图 5-69 所示的三个区域，集合 C 是集合 A 与集合 B 的交集。在目标检测当中，IoU 就是上面两种集合的比值。$A\cup B$ 其实就是 $A+B-C$。因此，交并比公式可以写为

$$\mathrm{IoU}=\frac{A\cap B}{A+B-(A\cap B)}$$

因此可以看出，IoU 衡量两个集合的重叠程度。IoU 为 0 时，两个框不重叠，没有交集；IoU 为 1 时，两个框完全重叠。IoU 为 0～1 时，代表两个框的重叠程度，数值越高，重叠程度越高。

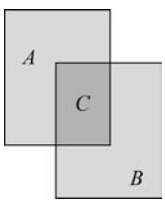


图 5-69　三个集合区域示意