第3章　卷积神经网络

卷积神经网络(Convolutional Neural Networks, CNN)是经典的深度学习神经网络模型,本章将以CNN的发展历程为切入点,介绍CNN的基本结构,以及前馈运算与反馈运算,在此基础上详细叙述CNN中的各层网络及操作,并分析CNN的经典结构。

3.1　卷积神经网络简介

3.1.1　卷积神经网络的发展历程

CNN是一类独特的人工神经网络,也是神经网络中最受关注的类别之一,专门用于处理具有网格结构的数据,特别是高维数据(如图像和视频)[1]。CNN的结构与一般的神经网络非常相似,它与其他神经网络最关键的区别在于是否有网络层使用卷积运算。使用卷积运算的网络层被称为卷积层,卷积层中的每个卷积核都是一个滤波器,它与该层的输入进行卷积运算,这有助于提取图像或视频等输入信息的特征,并且由于卷积核权值共享,因此可以大大降低网络层中的参数的数量。CNN是一种前馈神经网络,在多个应用领域发挥出了重要的作用。

1959年,神经科学家David H. Hubel和Torsten Nils Wiese通过对猫的大脑研究,得出在大脑的初级视觉皮层上,神经元是以层级结构的形式组织起来的,这些神经元先提取视觉信息的局部特征,然后将局部特征组合起来,从而学习识别视觉信息模式,这是对大脑初级视皮层的开创性研究,“感受野”(Receptive Field)这一概念也是在此时首次被提出的[2]。受此启发,日本学者福岛邦彦(Kunihiko Fukushima)提出一种最早的CNN形式,它由多层网络结构组成,包含了卷积层和池化层,可自动对模式识别任务进行分层特征提取[3]。在前人研究的基础上,1998年,Yann LeCun等提出LeNet-5网络模型,将BP算法首次应用到此神经网络模型的训练中,并将此模型成功应用于手写数字识别中,这就形成了当代CNN的雏形[4]。但此时的CNN效果并不太好,并且训练也非常困难,虽然在阅读支票、识别数字等任务上很有效果,但由于在一般的实际任务中表现不如支持向量机(Support Vector Machine, SVM)、Boosting等算法好,因此,一直处于学术界边缘的地位。

直到2012年,在著名的ILSVRC竞赛中,Hinton团队的论文提出AlexNet模型,引入了全新的深层网络结构和dropout方法,一次性将错误率从25%以上降低至15%,力挫对手一举取得该次竞赛冠军,颠覆了图像识别领域的传统认知[5]。虽然现在看来AlexNet模型是一项非常简陋的模型,但是当时这让很多研究学者意识到原来福岛邦彦提出的CNN形式

和 Yann LeCun 等提出的 LeNet-5 网络模型，都有很大改进空间。受到 AlexNet 思想的启发，2013 年，LeCun 团队提出 Dropconnect 操作，成功将错误率降低至 11%。而新加坡国立大学的颜水成团队则提出 Network in Network(NIN)，NIN 的思想是改变卷积神经网络原来的结构，加入一层 MLPConv(Multilayer Perceptron Convolution)。2014 年，NIN 模型也在 ILSVRC 竞赛中获得冠军[6]。在此基础上，2014 年又出现的 Inception 和 VGG 网络模型将层数加深到 20 层左右，图像识别的错误率也大幅降低到 6.7%，而人类的错误率为 5.1%，二者十分接近。

CNN 的基本结构与十几年前的差异并不大，但是在这数十年间，数据形式和硬件设备，尤其是图形处理器(GPU)的巨大发展，成为进一步推动 CNN 领域技术革新的主要动力，因此，深度神经网络不再是纸上谈兵和象牙塔里的研究，它真正成为能够切实应用到实际的生产和生活中的可行且好用的模型。

3.1.2 卷积神经网络的基本结构

CNN 是一种层次模型(hierarchical model)，输入的是原始数据(raw data)，如 RGB 图像、音频数据等。CNN 的基本网络层结构如图 3.1 所示，CNN 经过卷积(convolution)操作、池化(pooling)操作和非线性激活函数(non-linear activation function)映射等一系列操作的层层复合，将高层语义信息逐层由原始数据输入层中提取出来，这一过程是“前馈运算”(feed-forward)。在网络中每单独的一次操作被称为“层”，卷积操作对应“卷积层”，池化操作对应“池化层”，等等。最后，目标任务(分类、回归等)化为目标函数(objective function)形式作为 CNN 的最后一个输出层[7]。

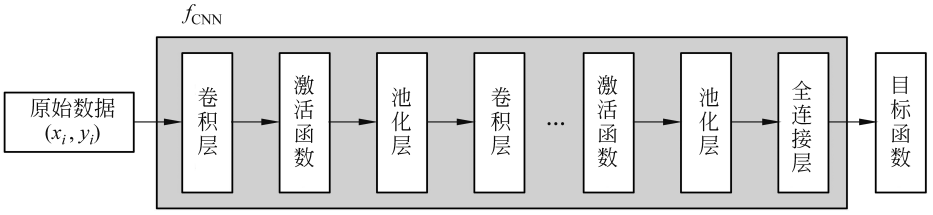


图 3.1 CNN 的基本网络层结构

通过计算预测值与真实值之间的误差(或者称为损失)，使用 BP 算法将误差由最后一层逐层向前反馈，以此更新每一层的参数，进行多轮的误差反馈和参数更新操作，直至网络参数和误差收敛或达到预定训练轮数[8]。

本质上，CNN 就是不断进行函数复合的过程，将卷积操作等作为一个个函数依次作用在原始数据上，逐层复合，最后计算损失函数将其作为最后一个函数复合，一般情况下，每层数据都可以表示为 $H \times W \times D$ 的张量(tensor)。例如，在目标检测等图像识别领域的应用中，CNN 的数据输入通常为 RGB 图像，有 H 行、W 列和 R、G、B 3 个通道，将输入数据记为 x^1，x^1 经过一次操作可得 x^2，对应第一个操作层中的参数为 w^1；第二个操作层中的参数为 w^2，以 x^2 作为第二层的输入，可得 x^3，这样直到第 $L-1$ 层，此时网络输出为 x^L，在整个过程中，每层操作一般是单独一个卷积操作、池化操作、激活函数或其他操作，也可以是不同形式操作的组合。最后，整个网络以损失值的计算结束。若 y 是输入 x^1 对应的真实标记，则

可将损失函数表示为

$$z = f(x^L, y) \tag{3-1}$$

式中，函数 $f()$ 中的参数为 w^L。

对于层中的某些特定操作，其参数 w^L 可以为空，如池化操作、无参数的激活函数以及无参数的损失函数等。在实际应用中，对于不同的目标需求，损失函数的形式也不相同。以回归问题为例，常用的 ℓ_2 损失函数即可作为卷积网络的目标函数，此时有

$$z = f(x^L, y) = \frac{1}{2} \| x^L - y \|^2 \tag{3-2}$$

在分类问题中，网络的目标函数常采用交叉熵(cross entropy)损失函数，于是有

$$z = f(x^L, y) = -\sum_i y_i \log(p_i) \tag{3-3}$$

式(3-3)中，有

$$p_i = \frac{\exp(x_i^L)}{\sum_{j=1}^{c} \exp(x_j^L)} (i = 1, 2, \cdots, C) \tag{3-4}$$

式中，C 为分类任务类别数。

无论是回归问题还是分类问题，在计算损失函数 z 之前，均需要通过合适的操作得到与 y 同维度的 x^L，方可正确计算样本预测的损失值。

3.1.3　前馈运算与反馈运算

与第 2 章介绍的 BP 神经网络模型类似，CNN 包括前馈运算与反馈运算[9]。实际上，无论是训练模型时计算误差还是模型训练完毕后获得样本预测，CNN 中的前馈运算都较直观。图 3.2 是前馈神经网络示意图，如以图像分类任务为例，假设网络已训练完毕，参数已收敛到某最优解，此时可用此网络进行图像类别预测，预测过程实际就是一次网络的前馈运算，即将测试集图像作为网络的数据输入，记为 x^1，再将输出传入隐含层，各隐含层输出记为 x^i，依此下去，直至输出 $x^L \in R^C$，利用交叉熵损失函数训练后，x^L 的每一维可表示样本分别属于 C 个类别的后验概率，如此可得 x^L 数值最大的那一维维数为输入图像对应的预测标记。

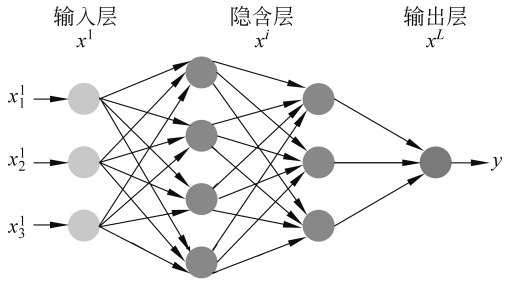


图 3.2　前馈神经网络示意图

神经网络具有的高容错性、鲁棒性及自组织性，是传统信息处理技术无法比拟的。因此，神经网络技术在语音识别、指纹识别、人脸识别、遥感图像识别和工业故障检测等方面都

得到了广泛的应用。在半个多世纪的研究和发展过程中，神经网络已经与众多学科和技术紧密结合，并且在众多领域都得到了广泛的应用和推广。下面简单阐述模式识别、信号处理、拟合逼近、优化选择、工程技术、博弈游戏等领域的应用。

和其他的许多机器学习模型一样，CNN(包括其他所有深度学习模型)也是将最小化损失函数值作为模型参数的训练目标，即最小化式(3-1)中的 z。从凸优化理论看，神经网络模型不仅是非凸(non-convex)函数，而且非常复杂，这使得优化求解变得十分困难，鉴于此原因，CNN 采用反馈运算中的随机梯度下降法进行模型参数更新迭代[10]。

在 CNN 求解时，特别是针对大规模应用问题，往往采用批处理的随机梯度下降法。批处理的随机梯度下降法在训练模型时随机选取 n 个样本作为一批样本，先通过前馈运算得到预测值并计算其对应的损失值，再使用梯度下降法进行参数更新，梯度从后往前逐层反馈，直至将网络的第一层参数更新完毕，这样的一次参数更新过程称为一次批处理过程。在处理不同批次时，按照不放回抽样原则遍历所有训练集样本，遍历一次训练样本称为一轮(epoch)。其中，处理样本的大小(batch size)不宜过小，过小时(如 batch size 为 1 或者 2 等)，由于样本采样的随机性，按照该样本上的误差更新模型参数不一定在全局上最优(此时仅为局部最优更新)，会使得训练过程产生振荡，也会导致训练批次大大增加，从而降低训练效率。而批处理大小的上限则主要取决于硬件资源的性能，最主要的就是 GPU 显存大小，如果批处理大小过大，很容易产生显存溢出的问题，通常批处理大小根据训练时显存占用率设为 2 的幂次方(如 32，64 或 256)。

如果某批处理前馈运算后得到的一批共 n 个样本上的误差和为 z，为便于理解，假设最后一层的损失函数取为 ℓ_2 损失函数，则可以得出

$$\frac{\partial z}{\partial w^L}=0 \tag{3-5}$$

$$\frac{\partial z}{\partial x^L}=x^L-y \tag{3-6}$$

通过式(3-5)和式(3-6)可以看出，对每一层的操作都可以求出两种偏导数：一种是损失函数值对第 i 层网络的参数 w^i 的偏导数；另一种是损失函数值对该层输入 x^i 的偏导数，则第 i 层参数的更新公式为

$$w_i \leftarrow w_i-\eta\frac{\partial z}{\partial w_i} \tag{3-7}$$

式(3-7)中，η 是每次随机梯度下降的步长，可令其随训练轮数的增加而减小。而损失值对该层输入的偏导数则是梯度下降算法的关键，其实质就是最终的损失值最后一层 L 传递到第 i 层的误差信号。

不失一般性，当误差反向传播至第 i 层时，使用第 $i+1$ 层的误差信号即可求得第 i 层参数更新时所需的参数导数和传至 $i-1$ 层的误差，根据多元复合函数求导的链式法则，可以得到

$$\frac{\partial z}{\partial(\mathrm{vec}\,(w^i)^{\mathrm{T}})}=\frac{\partial z}{\partial(\mathrm{vec}\,(x^{i+1})^{\mathrm{T}})}\cdot\frac{\partial \mathrm{vec}(x^{i+1})}{\partial(\mathrm{vec}\,(w^i)^{\mathrm{T}})} \tag{3-8}$$

$$\frac{\partial z}{\partial(\mathrm{vec}\,(x^i)^{\mathrm{T}})}=\frac{\partial z}{\partial(\mathrm{vec}\,(x^{i+1})^{\mathrm{T}})}\cdot\frac{\partial \mathrm{vec}(x^{i+1})}{\partial(\mathrm{vec}\,(x^i)^{\mathrm{T}})} \tag{3-9}$$

式中，vec 标记的作用是将张量转化为向量，从而便于计算和表示。向量求导的知识可以从矩阵论相关书籍中获得。通过式(3-7)、式(3-8)和式(3-9)可以实现 i 层参数的更新，并且将误差信号反向传递到前层，不断进行每一层导数的计算，直到更新到第一层就完成了一批次数据的参数更新训练。

3.2　卷积神经网络中的各层网络及操作

本节将对 CNN 中各个不同的网络层及操作进行详细介绍，通过这些网络的层层堆叠使得 CNN 可以直接从原始数据中学习其特征表示并完成最终任务。

3.2.1　卷积层

卷积运算在不同领域的定义有很多不同之处，本节只对 CNN 中的卷积运算进行讲解，卷积操作中的一个重要概念就是卷积核，其相当于一个滤波器[11]。卷积层中的每个卷积核都是一个由单独数字组成的网格。图 3.3 所示是 2×2 的卷积核示例，每个卷积核的权重(网格中的数字)是在 CNN 的训练过程中学习到的。而一般情况下，初始的权值都是在训练最初进行随机初始化的，然后每进行一次批处理，就使用 3.1.3 节中提到的随机梯度下降法对权值进行调整，不断调整之后能得到最终的卷积核。

2	0
1	3

图 3.3　2×2 的卷积核示例

在卷积层中，卷积操作是在卷积核和该层的输入之间进行的，图 3.4 是二维输入和二维卷积核的卷积操作示例，即给定一个 3×3 矩阵大小的二维输入特征图和一个 2×2 矩阵大小的卷积核，每次用此 2×2 大小的卷积核与输入特征图中的灰色区域(大小也为 2×2)进行对应数字的相乘得到积后再将积相加，从而生成输出特征图中的一个值。卷积核沿着输入特征图的水平位置或垂直位置移动，直到不能再进一步移动为止，卷积核每次移动的距离被称为卷积操作的步长，视需要可以将步长设置为不同的值，这里选取的移动步长是 1。

从图 3.4 可以看出，输出特征图的尺寸比输入特征图小，如果输入特征图的大小为 $h\times w$，卷积核的大小为 $f\times f$，卷积操作的步长为 s，则输出特征图的长 h' 和宽 w' 为

$$h'=\left\lfloor\frac{h-f+s}{s}\right\rfloor,w'=\left\lfloor\frac{w-f+s}{s}\right\rfloor \tag{3-10}$$

式中，$\lfloor\ \rfloor$ 代表向下取整操作。

在实际应用中，如在图像去噪和图像分割等应用中，我们希望在卷积操作后能够保持特征图的大小不变(甚至更大)，从而在网络设计中提供更大的灵活性，这可以通过在输入特征图周围使用零填充来实现。如果 p 表示沿每个维度的填充尺寸，那么可以将输出特征图的尺寸表示为

$$h'=\left\lfloor\frac{h-f+s+p}{s}\right\rfloor,w'=\left\lfloor\frac{w-f+s+p}{s}\right\rfloor \tag{3-11}$$

卷积操作是一种作用于输入特征图局部的操作，其有选择性地获取输入的各处局部特征，再将特征拼接到一起，因此卷积核也可以称为滤波器，为了体现卷积操作的效果，图 3.5

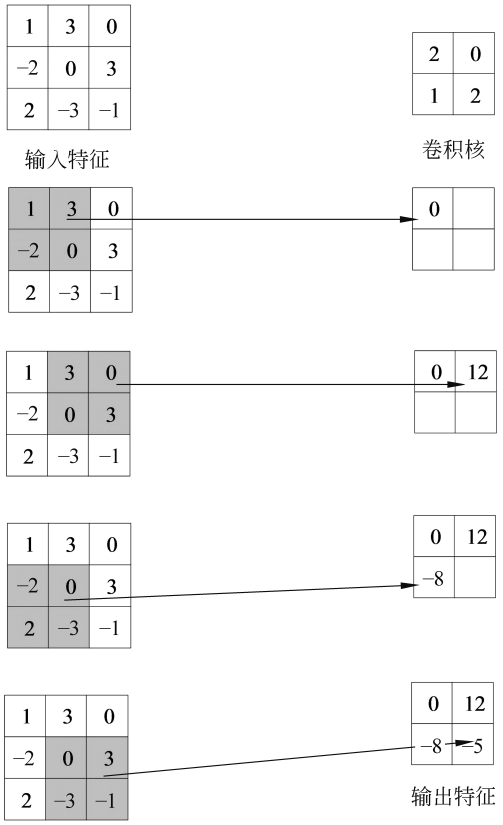


图 3.4　卷积操作示例

先给出大小都为 3×3 的 3 种卷积核，分别是整体边缘滤波器 K_e、横向边缘滤波器 K_h 和纵向边缘滤波器 K_v。图 3.6 是 3 种滤波器分别作用于原图后得到的输出效果对比图。

$$K_e=\begin{bmatrix}0&-4&0\\-4&16&-4\\0&-4&0\end{bmatrix}\quad K_h=\begin{bmatrix}1&2&0\\0&0&0\\-1&-2&-1\end{bmatrix}\quad K_v=\begin{bmatrix}1&0&-1\\2&0&-2\\1&0&-1\end{bmatrix}$$

图 3.5　大小为 3×3 的 3 种卷积核

从图 3.5 和图 3.6 可以看出，这 3 种滤波器可分别保留原图像整体、横向和纵向的边缘信息，这也是卷积操作的实际意义和功能。

为进一步体现卷积核的作用，图 3.7 是一个由两层卷积层组成的 CNN，其中第一层的输入是 RGB 图像，有 6 个尺寸为 7×7×3 的卷积核，第二层有一个尺寸为 5×5×6 的卷积核。由于输入图像有 3 个通道，因此，第一层卷积核的通道数都应为 3。对第一层的输入图像，每用一个卷积核进行一次卷积操作，都将产生一个单通道图像，因此第一层卷积层的输出是一个 6 通道的图像，从而第二层卷积核的通道数为 6，由于在第二层卷积层中只有一个卷积核，因此该层的输出将为一个单通道图像。在本实例中，第一卷积层和第二卷积层的卷积核是随机生成的，在第一层卷积层的输出，有两个卷积核起到了低通滤波器（即平滑滤波器）的作用，而另外 4 个卷积核起到了高通滤波器（即边缘滤波器）的作用，将第二层的输出图像与第

(a) 原图　　(b) 整体边缘滤波器

(c) 横向边缘滤波器　　(d) 纵向边缘滤波器

图 3.6　3 种滤波器作用结果对比图

一层的输出图像进行比较，可以看到第二层强化了猫的眼睛和鼻子四周轮廓明显的边缘，但是猫毛轮廓不够明显的边缘被减弱了，尽管猫毛在第一层网络输出时被加强。

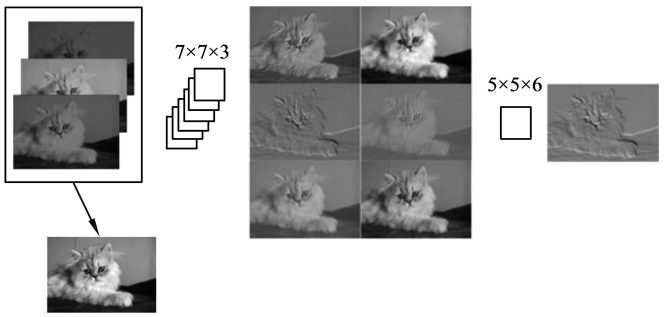


图 3.7　随机卷积核作用示例图

通过以上分析可以得出，由于卷积操作是权值共享的局部操作，每层卷积层的参数数量只取决于卷积核的大小，相比全连接网络，网络中的参数大大减少，因此，网络的复杂度大大降低，从而降低了训练网络所需要的硬件资源和时间，如使用随机梯度下降法调整参数时，所需要计算的导数大大减少。另外，卷积操作的作用体现在卷积核的滤波器效果，从图 3.7 可以看出，即使是随机选取的卷积核，仍然能够起到低通滤波器或者高通滤波器的作用，因此，经过在训练过程中不断调整参数，卷积核能成为适应需求的滤波器，能尽量显现所需要的信息，排除不重要的信息。

3.2.2 池化层

在 CNN 中，通常会将池化层夹杂在卷积层中间，池化层对输入特征图的某区域进行操作，此操作由一个池化函数定义。常用的池化操作有平均池化和最大池化，平均池化操作取区域中数值的平均数作为此区域的对应输出，最大池化则选取此区域中最大的数作为输出。与卷积操作类似，池化操作也需要选择池化核的大小和操作步长，并且和卷积核有着相同的移动方法，图 3.8 是最大池化操作示例图，其中池化核的尺寸为 2×2，池化步长为 1。

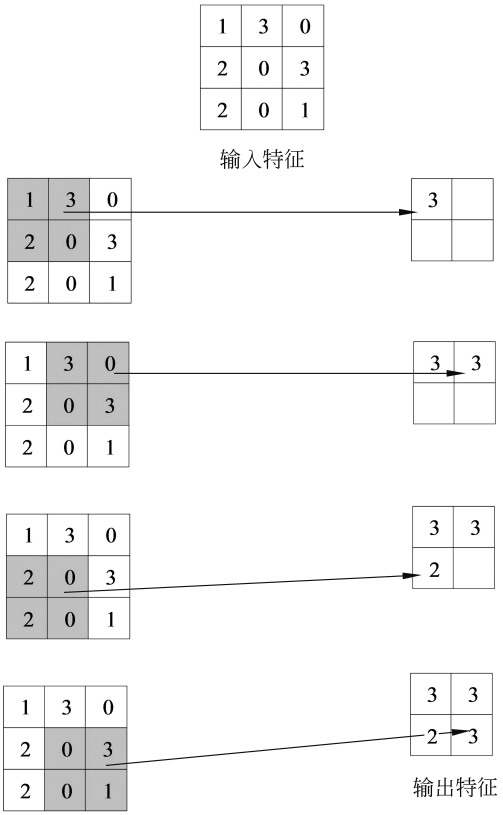


图 3.8　最大池化操作示例图

如果输入特征图的尺寸为 $h \times w$，池化核的尺寸为 $f \times f$，池化操作步长为 s，则池化层输出的特征图尺寸可表示为

$$h' = \left\lfloor \frac{h - f + s}{s} \right\rfloor, w' = \left\lfloor \frac{w - f + s}{s} \right\rfloor \tag{3-12}$$

假设将一幅 190×190 的图像输入到一层包含 50 个尺寸为 7×7 卷积核的卷积层中，则此卷积层将输出 50 张尺寸为 184×184 的特征图，将它们组合起来就是一张有 50 个通道的图像。从另一个角度看，就可以发现输出图像实质是一个 184×184×50＝1692800 维的向量。显然，这个向量的维数特别高，而池化层的主要目的是降低特征图的维数，因此，池化操作也被称为降采样[12]。假设有一个 12 维的向量 $\boldsymbol{X}$＝[1,10,8,2,3,6,7,0,5,4,9,2]作为特

征输入，使用步长 $s=2$ 的简单降采样，即每次选取区域中的第一个元素作为降采样输出，则经过此操作之后能得到特征输出向量[1,8,3,7,5,9]，显然，此向量的维数为 6，确实达到了降采样的效果，但是使用这种降采样方法降低 $\boldsymbol{X}$ 的维数时，会忽略区域中不同像素之间的数值关系，若有 $\boldsymbol{X}_1=[1,10,8,2,3,6,7,0,5,4,9,2]$ 和 $\boldsymbol{X}_2=[1,100,8,20,3,60,7,0,5,40,9,20]$，降采样步长为 2，则输入其中任意一个都是向量[1,8,3,7,5,9]作为输出。然而，$\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 代表两种完全不同的输入，因此，使用此降采样方法可能丢失重要的信息。

为了尽量减少信息的丢失，池化操作会考虑每个区域中数值之间的相互关系，假设对 $\boldsymbol{X}_1=[1,10,8,2,3,6,7,0,5,4,9,2]$ 和 $\boldsymbol{X}_2=[1,100,8,20,3,60,7,0,5,40,9,20]$ 进行最大池化操作，输入 $\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 将分别产生输出向量[10,8,6,7,5,9]和[100,20,60,7,40,20]。可以看出，与简单降采样不同的是，最大池化操作不会忽略任何元素，相反，它在降低特征图维数的同时还能保证重要信息不丢失。池化操作的步长通常设置为 2，池化核的尺寸通常为 2×2 或者 3×3，因此操作作用的区域可能会相互重叠。在实际应用中，平均池化操作很少用于中间的网络层，因此通常 CNN 的池化层都使用最大池化层进行保持特征不变性的下采样操作。

3.2.3　激活函数层

在 CNN 的权重层（如卷积层和全连接层）之后通常会复合一个非线性激活函数（或分段线性函数），大多数激活函数是将大范围的输入值压缩到一个很小的范围内，通常是区间[0,1]或者[−1,1]。在权重层后复合一个非线性激活函数是非常重要的，因为非线性因素会使得 CNN 在训练过程中能够很好地拟合非线性映射，在没有非线性激活函数的情况下，多个权重层叠加形成的网络只能等价于一个从输入空间到输出空间的线性映射。一个非线性激活函数也可以被认为是一种开关或选择机制，它决定神经元是否减小或者舍弃得到的输入值。本节主要讲述两种重要的激活函数，即 Sigmoid 激活函数和 ReLU 激活函数[13]。

Sigmoid 激活函数能够模拟生物神经元的兴奋或抑制状态，因此其在神经网络发展历史中曾经有着举足轻重的地位，其表达式为

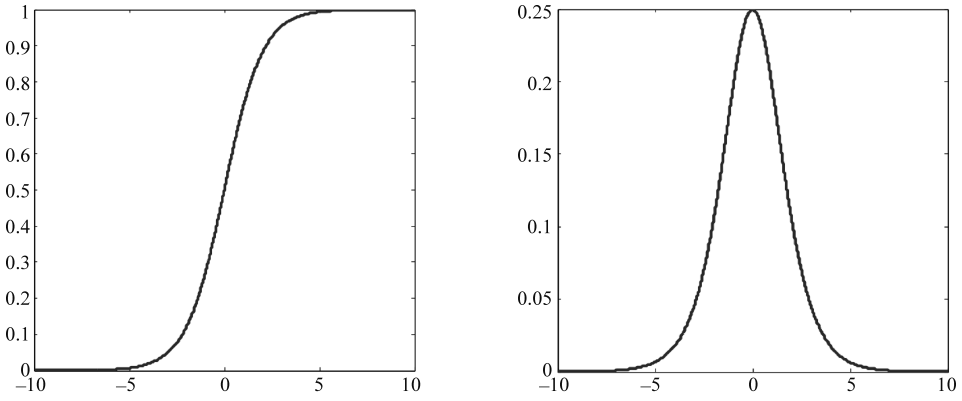
$$f(x)=\frac{1}{1+\exp(-x)} \tag{3-13}$$

图 3.9 是 Sigmoid 激活函数及梯度图，在经过 Sigmoid 激活函数的作用之后，输出的值域被压缩到[0,1]区间，而 0 对应生物神经元的“抑制状态”，1 则对应“兴奋状态”。还可以看到，在 Sigmoid 激活函数两端，对于大于 5 或者小于−5 的值，无论多大或者多小，都会压缩到 1 或者 0，这便带来一个非常严重的问题，即梯度“饱和效应”，在大于 5 或者小于−5 的区间内梯度接近 0，这会导致在误差反向传播过程中导数处于该区域的误差将很难甚至根本无法传递至前层，进而导致整个网络无法继续训练（导数为 0 将无法更新网络参数）。因此，在参数初始化的时候还须特别注意避免初始化参数直接将前层输出的值域带入这一区域，即当初始化参数过大时，将直接引发梯度饱和效应而无法训练。

为避免梯度饱和效应发生，引入了 ReLU 激活函数。ReLU 激活函数是目前 CNN 中最为常用的激活函数之一[14]，它实际是一个分段函数，其定义为

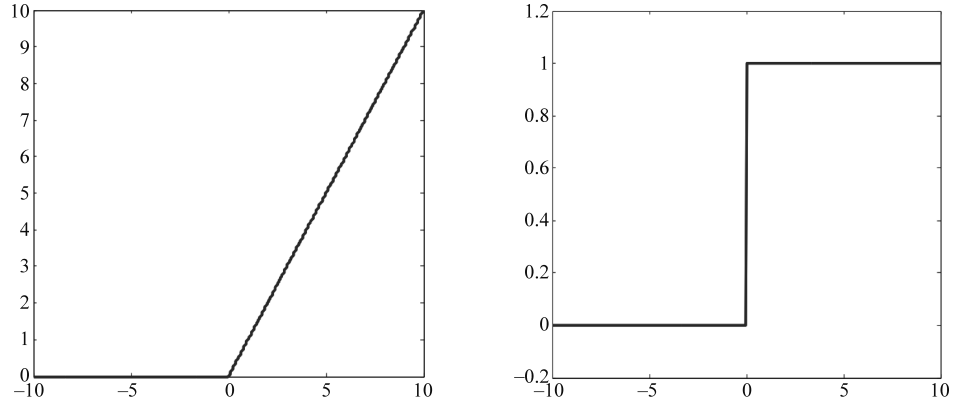
$$f(x)=\begin{cases}x & x\geqslant 0\\ 0 & x<0\end{cases} \tag{3-14}$$

图 3.10 是 ReLU 激活函数及梯度图，ReLU 激活函数的梯度在 $x \geqslant 0$ 时为 1，反之为 0。在 $x \geqslant 0$ 区间内完全消除了梯度饱和效应。同时，相比 Sigmoid 激活函数，ReLU 函数有助于加快随机梯度下降方法收敛，收敛速度约快 6 倍左右。正是由于 ReLU 激活函数的这些优越性质，目前其已成为 CNN 以及其他深度学习模型（如 RNN 等）激活函数的首选。



(a) Sigmoid 函数图像　　(b) Sigmoid 函数梯度图像

图 3.9　Sigmoid 激活函数及梯度图



(a) ReLU 函数图像　　(b) ReLU 函数梯度图像

图 3.10　ReLU 激活函数及梯度图

3.2.4　全连接层

在 CNN 结构中，多个卷积层和池化层之后，通常都会连接一个或者多个全连接层（Fully Connected Layer，FC），全连接层基本上可以看成一个卷积核大小为输入特征图尺寸，卷积核数量为输出维数的卷积层。全连接层中的每个输出单元都与输入的所有单元完全相连。在一般的 CNN 中，全连接层放置在网络的末端，但是有的研究学者也发现在 CNN 的中间层使用全连接层也能构造出效果很好的网络。全连接层可以表示为矩阵的乘法，如果在全连接层之后添加一个激活函数 f，则全连接层可以表示为

$$\boldsymbol{Y} = f(\boldsymbol{WX} + \boldsymbol{b}) \tag{3-15}$$

式中，$\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 分别为输入激活和输出激活的向量，$\boldsymbol{W}$ 为神经元之间连接的权值矩阵，$\boldsymbol{b}$ 为偏移向量。

全连接层与多层感知器中的权重层基本相同，其存在是为了将卷积层或者池化层中具有类别区分性的局部信息（特征信息）进行整合。因此，全连接层在整个 CNN 中起到“分类器”的作用[15]。

3.2.5　损失函数

CNN 中的最后一层使用一个损失函数（Loss Function），也被称为目标函数，用以评估网络对训练数据的预测质量，其中真实的标签是已知的。损失函数量化了模型的估计输出（预测）和正确输出（真实标签）之间的差距，网络训练的目的就是找到使损失函数最小化的网络参数。在 CNN 中使用的损失函数类型取决于对预测目的需求。常用的损失函数有 ℓ_2 损失函数和交叉熵损失函数，它们二者的详细叙述已经在 3.1.2 节中给出，这里不再赘述[16]。

3.3　卷积神经网络经典结构

CNN 的经典模型包括 LeNet 模型、AlexNet 模型、Network-In-Network 模型、VGG-Net 模型以及 GoogLeNet 模型。通过分析这些经典 CNN 模型，读者可以充分体会每一种 CNN 模型在不同领域的应用，实际应用中，需要读者根据具体的情景需求选择恰当的模型。

3.3.1　LeNet 模型

1989 年，LeCun 等提出权值共享策略，并由此设计出卷积层和池化层，接着在 1998 年，他们又设计了一个被命名为 LeNet-5 的卷积神经网络，其中数字 5 表示它拥有 5 层权重层，该卷积神经网络的网络结构如图 3.11 所示。

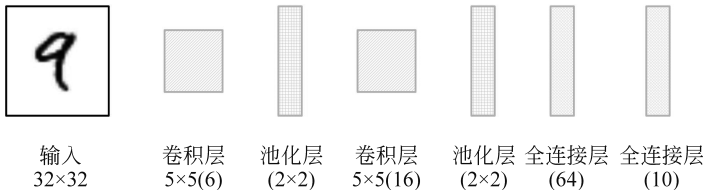


图 3.11　LeNet-5 模型结构图（见彩插）

设计 LeNet-5 的初衷是用于识别手写数字，网络的输入是一幅单通道的 32×32 图像，第一层卷积层包含 6 个尺寸为 5×5 的卷积核，用这些卷积核对一幅尺寸为 32×32 的图像进行卷积操作，将生成 6 幅尺寸为 28×28 的特征图。第一层卷积层之后是池化层，其操作的步长为 2，因此第一层池化层的输出是 6 幅尺寸为 14×14 的特征图，它们共同组成一幅 6 通道的输入特征图。接着在第二层的卷积层中对此 6 通道图像应用 16 个尺寸为 5×5 的 6 通道卷积核，因此其实质尺寸为 5×5×6，此卷积层的输出为 16 幅尺寸为 10×10 的特征图，

合在一起是一幅 16 通道的特征图。接下来，第二层池化层应用步长 2 的池化操作，生成 16 幅尺寸为 5×5 的特征图，这些图像输入一层有 84 个神经元的全连接层，经过处理后再输入最后一层的全连接层中，全连接层实质上起到一个分类器的作用，由于 LeNet-5 模型的目的是识别手写数字，因此有 10 个输出神经元。需要特别提出的是，LeNet-5 模型中的池化操作并不是常用的最大池化操作或平均池化操作，它将池化核对应的 4 个数字相加，将它们除以 a 再加上偏差 b，并将结果作为输出，其中 a 和 b 都是训练得到的。此外，LeNet-5 模型使用 Sigmoid 激活函数，并且只在池化层之后应用激活函数，而在卷积层之后则不存在激活函数。由于 LeNet-5 模型是早期的 CNN 模型，因此结构比较简单，只有 60000 个需要训练的参数，难以处理较复杂的问题。

3.3.2 AlexNet 模型

AlexNet 模型是第一个引起计算机视觉中深度神经网络复兴的大型 CNN 模型，该模型在 2012 年以巨大的优势赢得 ILSVRC 竞赛的冠军。AlexNet 模型结构图如图 3.12 所示。

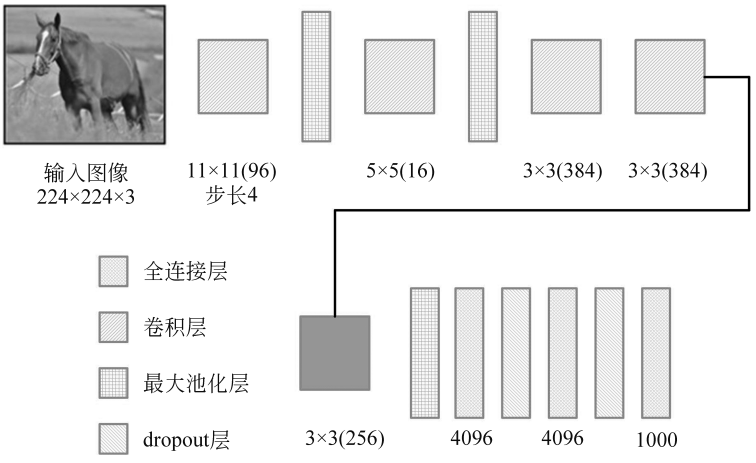


图 3.12 AlexNet 模型结构图（见彩插）

AlexNet 模型架构与其前身之间的主要区别是网络深度的增加，这导致网络参数显著增加，并且使用正则化方法（如 dropout 和数据增强）。它总共包含 8 个权重层，其中前 5 个是卷积层，后 3 个是全连接层。最后的全连接层（即输出层）是将输入图像分类为 ImageNet 数据集的 1000 类数据中的一种，因此有 1000 个神经元。dropout 应用于 AlexNet 模型中的前两个全连接层，这缓解了过拟合现象，使模型拥有更好的泛化性能。AlexNet 模型的另一个创新是在每层卷积层和全连接层后使用 ReLU 激活函数，这样可以大大提高训练效率。

AlexNet 模型在首次训练网络时，利用两块 GPU 进行训练，因为 3GB 显存的单个 NVIDIA GTX580 显卡不能容纳包含约 6200 万个参数的完整网络，ImageNet 训练数据集包含属于 1000 种不同对象的 1200 万幅图像，因此在完整的 ImageNet 数据集上训练该网络大约花了 6 天的时间。

3.3.3　Network-In-Network 模型

Network-In-Network 模型是一个简单而轻量级的 CNN 模型，它在小规模数据集上通常表现得非常好。图 3.13 是 Network-In-Network 模型结构图。

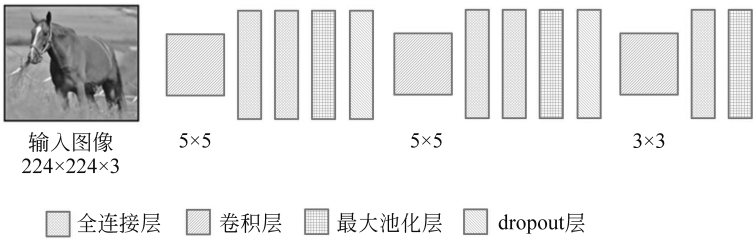


图 3.13　Network-In-Network 模型结构图（见彩插）

在此网络中提出两个 CNN 设计中的新想法。首先，在卷积层之间加入全连接层有助于网络训练。因此，该示例模型的 3 个卷积层分别位于权重层中的第 1 层、第 4 层和第 7 层，其卷积核的尺寸分别为 5×5、5×5 和 3×3。每层卷积层后面都紧跟着一层全连接层和一层最大池化层。其次，该模型在网络的末尾使用一层平均池化层进行正则化操作。值得注意的是，在前两个最大池化层之后使用 dropout 正则化方法，也有助于在给定的数据集上实现更低的测试错误率。

3.3.4　VGG-Net 模型

VGG-Net 模型是自 2014 年以来最受欢迎的 CNN 模型之一，其在 2014 年的 ILSVRC 竞赛分类项目中取得第二名，并且在定位项目中荣获第一名。它受欢迎的原因是其模型的简单性和其使用了小型的卷积核，但这也导致网络的层次非常深。常见的 VGG-Net 模型有两种，分别是 VGG16 与 VGG19[17]。

VGG-Net 模型中的卷积核尺寸都为 3×3，多层卷积层和夹杂其中的池化层用于特征提取，最后 3 层全连接层用于分类。在 VGG-Net 模型中，每层卷积层之后都有一个 ReLU 激活函数，并且使用尺寸较小的卷积核，使网络参数数量相对减少，从而可以进行更有效的训练和测试。此外，通过使用一系列尺寸为 3×3 的卷积核，增大了感受野的范围。最重要的是，使用更小的卷积核，就可以堆叠更多的层，从而产生更深层的网络，这使得 VGG-Net 模型在计算机视觉领域上有更好的表现。因此，VGG-Net 模型设计的中心思想就是使用更深层次的网络改进特征学习。图 3.14 是 VGG-Net 中性能最好的模型（VGG16 模型）结构图。

VGG16 模型有 1.38 亿个参数，与 AlexNet 模型类似，它也在前两个全连接层后应用 dropout 操作避免过拟合。

3.3.5　GoogLeNet 模型

CNN 经典模型中的 LeNet 模型、AlexNet 模型、Network-In-Network 模型和 VGG-Net 模型都是只有一条路径的顺序结构。在这条路径上，不同类型的层，如卷积层、池化层、

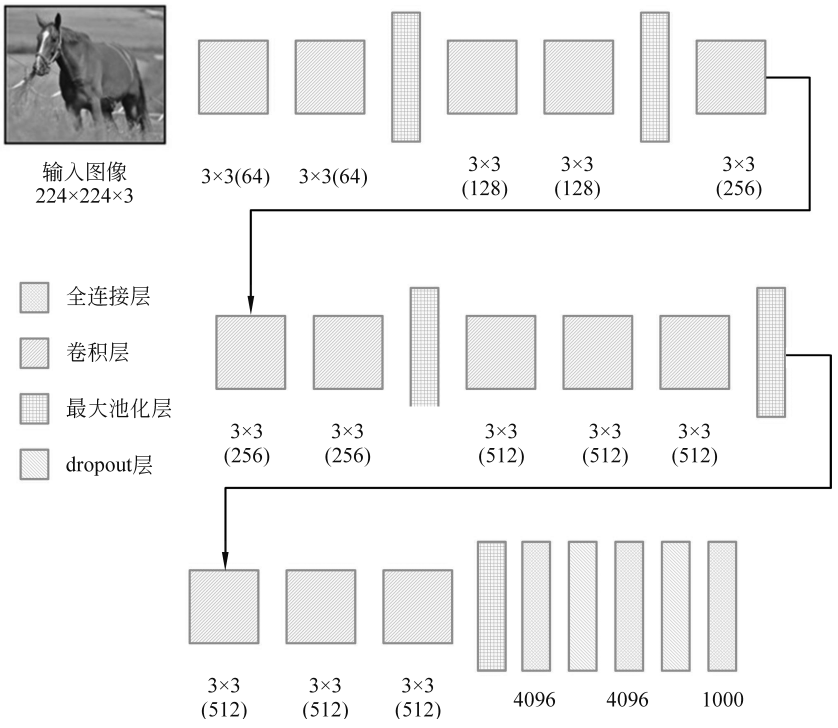


图 3.14　VGG16 模型结构图(见彩插)

ReLU 激活函数、dropout 方法和全连接层相互堆叠,以创建所需的模型。GoogLeNet 模型是首个使用具有几个网络分支的复杂架构的流行模型,2014 年它在 ILSVRC 竞赛中以 6.7%的分类错误率获得前五名。

GoogLeNet 模型总共有 22 层权重层,其设计的基础是 Inception 模块,因此 GoogLeNet 模型也被称为 Inception 网络。Inception 模块中网络层的运算是并行的,这与模型的顺序运算相反[18]。Inception 模块示意图如图 3.15 所示。

其核心思想是将所有基本网络层并行放置,并将输出的特征进行结合后再输出。这种设计的好处是多个 Inception 模块可以堆叠在一起,从而组成一个巨型网络,而不需要考虑每个单独网络层的设计。然而,如果通过增加通道数的方式将各个网络层的输出特征进行结合,将产生一个维数非常高的特征输出结果。为了解决这个问题,完整的 Inception 模块在将特征输入卷积层之前进行降维操作,这种降维方法是通过添加一层卷积核尺寸为 1×1 的卷积层实现的。假设此卷积层有 d_0 个卷积核,输入图像尺寸为 $h\times w\times d$ 且 $d_0<d$,该层的输出将降低维数至 $h\times w\times d_0$,通过使用这样的一层卷积层可以在降维的同时组合来自多个特征通道的信息,从而提高 Inception 模块的性能。

尽管 GoogLeNet 模型架构看起来比较复杂,但它包含的参数数量明显少得多(600 万),因此,GoogLeNet 模型对显存的需求更小,相比于其他网络模型拥有更高的训练效率和精度,也是最直观的 CNN 模型之一,这充分说明了良好的网络模型设计的重要性。

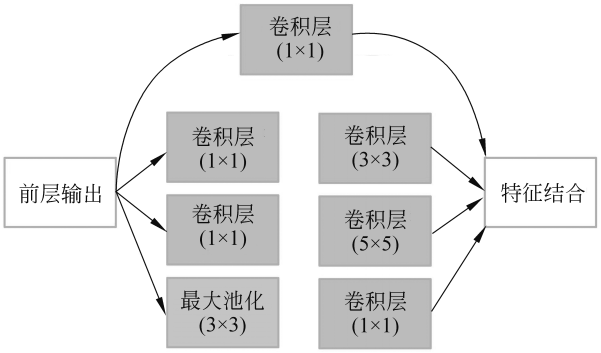


图 3.15　Inception 模块示意图(见彩插)

3.4　本 章 小 结

本章首先对 CNN 进行了叙述,内容包括 CNN 的发展历程、基本结构、前馈预算及反馈运算,其次对 CNN 中的各层网络及操作进行详细叙述,内容包括卷积层、池化层、激活函数层、全连接层和损失函数,最后对 CNN 经典结构中的 LeNet 模型、AlexNet 模型、Network-In-Network 模型、VGG-Net 模型以及 GoogleNet 模型进行叙述。通过本章内容的学习,读者可以全面了解 CNN 模型。

3.5　章 节 习 题

1. 查阅相关资料,并结合本书总结卷积层有哪些基本参数。

2. 卷积核是否越大越好?

3. 编程实现一个卷积神经网络模型和一个全连接神经网络模型,在 MINIST 手写数字数据集(数据集见 http://yann.lecun.com/exdb/mnist/)上进行实验测试,并对比二者的结果。

参 考 文 献

[1] 郭丽丽, 丁世飞. 深度学习研究进展[J]. 计算机科学, 2015,42(5): 28-33.

[2] HUBEL D H, WIESEL T N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex [J]. The Journal of Physiology, 1962,160(1): 106-154.

[3] FUKUSHIMA F K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position [J]. Biological Cybernetics, 1980,36(4): 193-202.

[4] 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.

[5] KRIZHEVSKY A, SUTSKEVER I, GEOFFREY E. Hinton. Imagenet classification with deep

convolutional neural networks [J]. Advances in neural information processing systems, 2012, 25: 1097-1105.

[6] LIN M, CHEN Q, YAN S. Network in network[J]. arXiv preprint arXiv, 2013: 1312-4400.

[7] 邱锡鹏. 神经网络与深度学习[M]. 北京：机械工业出版社，2020.

[8] RUMELHART D E, HINTON G E, WILLIAMS R J. Learning representations by back-propagating errors[J]. Nature, 1986, 323(6088): 533-536.

[9] 孙志军，薛磊，许阳明，等. 深度学习研究综述[J]. 计算机应用研究，2012，29(8)：2806-2810.

[10] 袁梅宇. 机器学习基础：原理、算法与实践[M]. 北京：清华大学出版社，2018.

[11] WU J. Introduction to convolutional neural networks[J]. National Key Lab for Novel Software Technology, 2017, 5(23): 495.

[12] GOODFELLOW I J, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial nets[C]// International Conference on Neural Information Processing Systems, 2014: 2672-2680.

[13] 刘玉良. 深度学习[M]. 西安：西安电子科技大学出版社，2020.

[14] NAIR V, HINTON G E. Rectified linear units improve restricted boltzmann machines [C]// International Conference on Machine Learning (ICML), 2010: 807-814.

[15] 王汉生. 深度学习：从入门到精通[M]. 北京：人民邮电出版社，2021.

[16] ZHANG C L, ZHANG H, WEI X S, et al. Deep bimodal regression for apparent personality analysis [C]//European Conference on Computer Vision, 2016: 311-324.

[17] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition [J]. arXiv preprint arXiv, 2014: 1409-1556.

[18] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2015: 1-9.