# 第3章

视频讲解

# 卷积神经网络的基本构建

本章将首先深入探讨卷积神经网络的核心构建模块，基于卷积运算的基本原理介绍卷积核的作用、不同类型卷积操作的特点，以及如何进行图像特征的提取，然后将详细阐述可变形卷积技术，以及反卷积和目标分割的概念。最后，本章将讨论卷积神经网络中池化层的多重特性、全连接层的作用与影响，以及数据的标准化与正则化。通过本章的学习，读者将获得构建和优化 CNN 模型的实战技巧，为进一步的深度学习项目开发奠定基础。

## 3.1 卷积层的多种操作

### 3.1.1 卷积运算的基本原理

卷积神经网络(Convolutional Neural Networks，CNN)，又称为卷积网络，是一种专门设计用于处理具有类似网格结构的数据的神经网络。这类数据包括时间序列微观数据(可视为在时间轴上有规律地采样形成的一维网格)和图像数据(可视为二维的像素网格)。卷积网络在多个应用领域都表现出色。术语“卷积神经网络”指的是该网络采用卷积(Convolution)这种数学运算。卷积是一种特殊的线性运算，最初应用于信号处理。在信号处理中，卷积的实现方式是通过对输入信号进行卷积计算，输入信号通过信号处理系统并输出信号。卷积的核心概念是通过一个函数在另一个函数上进行滑动重叠计算，这个函数可以通过对另一个函数的翻转得到。

实际上卷积操作并不涉及“积”的运算，而是一个滑动叠加的值。以信号卷积为例，卷积与时间和当前时刻的信号值大小相关。卷积过程中的“卷”是指函数之间的滑动操作。卷积操作的输出是当前时刻信号与之前输入信号的叠加值，这反映了卷积操作的物理意义。卷积操作可用于连续点的实际信号，也可通过对实际点进行离散化后进行信号卷积。在连续或离散情况下，卷积操作的实际过程相同。

通常情况下卷积是两个实变函数的数学运算，通过局部感知和参数共享，卷积实现了对输入数据的特征提取和表示学习。图像由各个像素点组成，每个像素点包含 R、G、B

三个通道值。这些通道值的范围是0～255，数值越大表示颜色越深。RGB通道的不同值大小形成不同颜色，因此图像上的各种颜色点可通过RGB像素值直接展现。典型的卷积过程，如图3-1所示。

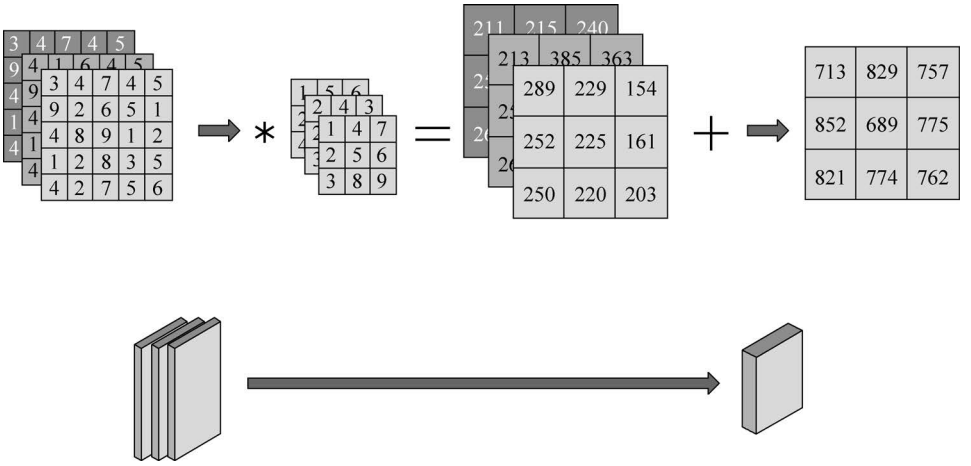



图3-1　典型的卷积过程

在卷积运算中，有三个重要概念需要理解：卷积核（或滤波器）、填充（Padding）和步长（Stride）。以下是对这些概念及卷积运算基本原理的详细介绍。

**1. 卷积核（Kernel）**

卷积核是卷积运算的关键组成部分，通常为一个小矩阵，包含可学习的权重参数。常见大小如3×3、5×5等，每个元素代表卷积核在输入数据中的权重。卷积核的作用类似图像中的滤波器，通过不同参数配置从像素和频率两个角度分析图像信息。在反向传播的过程中，卷积核参数不断更新，以逐渐提取和识别目标的特征。

**2. 填充（Padding）**

在卷积运算中，填充是一项重要考虑因素，其目的是在输入数据周围添加额外值，以防止卷积核越过边缘而导致信息损失。填充的存在能够影响输出特征图的大小，从而调整网络的感受野。通过调整填充参数，可以在卷积运算中更好地控制信息的流动。

**3. 步长（Stride）**

步长是卷积核在输入数据上滑动的距离，也是影响输出特征图大小的关键因素。调整步长可以控制输出特征图的尺寸，影响信息的采样密度。不同步长值的选择会在卷积运算中产生不同的感受野，进而影响网络对输入数据的理解和学习。

**4. 卷积的基本公式**

在卷积网络中，卷积的基本公式如下，卷积运算通常用星号表示：

$$s(t)=(x*w)(t) \tag{3-1}$$

其中，$s(t)$是输出，$x$是输入，$w$是核函数。输出有时被称为特征映射（Feature Map）。随着卷积层的增加，卷积核的数量也在增加。通过滑动方式提取图像特征的原因是图像通常很大，有助于减少参数数量。增加网络层数和卷积核数量，或者使用多个卷积核进行

权值共享，都是提高特征提取效果的方式。

图像中的像素值以矩阵的方式进行处理，如图 3-1 所示，其中每个像素的位置用 R、G、B 像素值表示为三元组。这种矩阵形式使得图像能够参与数学上的卷积操作，类似图像离散信号中的卷积操作。卷积操作对图像进行处理，提供了一种专门的卷积技术，其卷积核的功能可类比信号中的系统响应函数，用于处理图像并提取特征。在反向传播的过程中，卷积核中的值会不断更新，从而使其能够更好地提取和识别目标的特征。

值得注意的是，图像中卷积核的值在反向传播过程中不断地进行修改和更新，以逐渐提取其特征。因此，卷积核是否进行与信号的卷积模式相似的翻转操作，对图像来说并没有太大的作用和影响。图像的卷积过程可以概括为以下 3 部分。

**1. 卷积核的初始化**

对图像的卷积核的大小和其中的值进行初始化，为后续的卷积操作做准备。

**2. 局部卷积操作**

卷积核对图像中的每一部分进行卷积操作，乘积和得到卷积过程，即图像中对应的像素点与卷积核中的点的乘积和。这一步是图像特征提取的关键。

**3. 滑动卷积核**

卷积核按照一定的规则进行滑动，并同样进行卷积操作，导致卷积后的图像特征图相比原特征图的尺寸有所缩小。这样的滑动方式有助于有效地捕捉图像中的特征。

在神经网络中，图像在第一层的卷积过程，实际上是在图像的同一位置上同时进行着三层的卷积操作，而每个图像矩阵中的点的值都可以视为离散信号中的离散点。卷积操作提取图像特征的原理在于卷积核实际上也可以看成是图像的一种滤波器。图像除了可以从像素点的角度分析信息之外，还可以从频率的角度来分析，如高频信号、低频信号等。通过不同卷积核中参数的配置，可以实现对图像信息的多方面提取。反向传播算法通过对提取的特征和实际的图像进行损失计算，可以逐渐更新卷积核中的参数，实现图像的特征提取。

### 3.1.2　常规卷积操作

标准卷积是卷积方式中最简单且最常用的，它通过多个卷积核（通道数）对图像的 R、G、B 三个通道进行分别卷积，从输入数据中提取特征信息。卷积操作基于滑动窗口和加权求和原理，卷积核与输入数据逐元素相乘并求和，生成输出特征图。在卷积操作中，涉及四个主要参数：卷积核尺寸、步长、填充尺寸以及通道数。

**1. 卷积核尺寸**

卷积核尺寸代表网络中感受野的大小，不同尺寸的卷积核导致生成特征图的大小不同。常用的卷积核尺寸为 3×3，卷积核的尺寸可以根据任务需求进行调整。

**2. 步长**

步长定义为卷积核在图像上滑动时的跨越长度。每移动一次步长，卷积核进行一次卷积操作。在常规卷积操作中，步长默认为 1，即卷积核每次移动 1 像素；若步长为 2，则卷积核每次移动 2 像素。

**3. 填充尺寸**

填充的目的是使运算结果与输入图像的大小保持一致。受卷积特性的影响，输入图像与卷积核进行卷积后的结果中损失了部分值，输入图像的边缘被“修剪”，为了保证输入和输出的大小保持一致，就需要对输入图像进行填充，填充值通常为0。填充尺寸大小与图像的尺寸、卷积核尺寸和步长有关。

**4. 通道数**

输出图像的通道数与卷积核数量相同，一个卷积核只能提取图像中某一个特征。卷积核的数量决定了提取的特征数量，实际项目中的输出通道数可以根据需求设定。

下面通过一个示例来演示常规卷积的计算。假设有一张尺寸为256像素×256像素的彩色图像，有三个通道（红色、绿色和蓝色）。请使用卷积操作来检测图像中的边缘特征。

首先，定义一个大小为3×3的边缘检测卷积核，权重参数如下：

$$\boldsymbol{K}=\begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix} \tag{3-2}$$

接下来，将输入图像和卷积核进行卷积操作。选择不进行填充操作，步长为1。因此，输出特征图的尺寸将缩小为254像素×254像素。

卷积核的左上角与输入图像的左上角对齐，逐步滑动卷积核，在每个位置上计算卷积运算。例如，在输出特征图位置(1,1)上，执行以下计算：

$$O(1,1)=\sum_{m=0}^{2}\sum_{n=0}^{2}I(1+m,1+n)\cdot K(m,n) \tag{3-3}$$

其中，$I$表示输入图像，$O$表示输出特征图。

通过对整个输出特征图进行计算，得到一个新的尺寸为254像素×254像素的特征图，其中每个元素表示对应位置的边缘强度。最后，应用非线性激活函数（如ReLU）增加网络的表达能力，产生一个高亮显示图像中边缘特征的最终特征图。结果表明，卷积操作能较好地提取图像中的边缘特征，卷积的代码示例如代码3-1所示。

**【代码3-1】** PyTorch中卷积的实现。

```
#定义卷积核尺寸
kernel_size = (3, 3)
#定义步长
stride = (1, 1)
#定义填充尺寸
padding = (1, 1)
#输入通道数
in_channels = 3
#输出通道数
out_channels = 10
#定义卷积
conv = torch.nn.Conv2d(
    in_channels = in_channels,
    out_channels = out_channels,
```

```
        kernel_size = kernel_size,
        stride = stride,
        padding = padding,
    )
```

卷积操作的优点之一是权重共享性质。在这个例子中，卷积核的权重在整个图像上共享，只需学习一个小的卷积核，而不是为每个位置设计一个滤波器。这降低了模型的参数数量，提高了计算效率。使用不同卷积核，可以检测图像中的不同特征，如边缘、纹理、角点等，使得卷积操作成为计算机视觉任务中不可或缺的工具，如目标检测、图像分割和人脸识别等。

### 3.1.3　深度可分离卷积

深度可分离卷积(Depthwise Separable Convolution)，包括逐通道卷积(Depthwise Convolution)和逐点卷积(Pointwise Convolution)。深度可分离卷积的目的是减少网络中的参数，实现轻量化，从而提高算法的推理速度。

深度卷积的实现过程可以分为逐通道卷积和逐点卷积两部分。

**1. 逐通道卷积**

逐通道卷积用于实现对单个通道的特征提取。标准卷积通常通过经验确定需要输出的通道数目，而深度卷积中，第一层卷积核的数目等于图像的通道数。每个通道由一个卷积核负责，并且只由该通道进行卷积。这意味着输出的特征图的个数也与通道数相同。卷积的操作公式如下所示：

$$O_i = I_i * K_i \tag{3-4}$$

式中：$O_i$ 表示输出特征图的第 $i$ 个通道；$I_i$ 表示输入图像的第 $i$ 个通道；$K_i$ 表示第 $i$ 个卷积核。逐通道卷积示意图如图 3-2 所示。

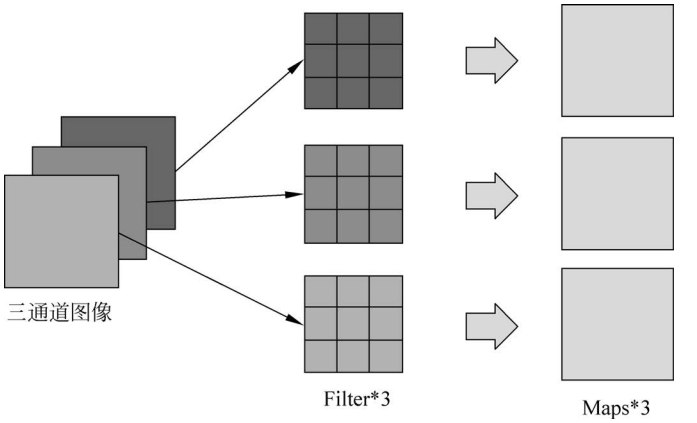



图 3-2　逐通道卷积示意图

**2. 逐点卷积**

逐点卷积的作用是在多个通道之间实现信息交互。像素之间的卷积图如图 3-3 所示。

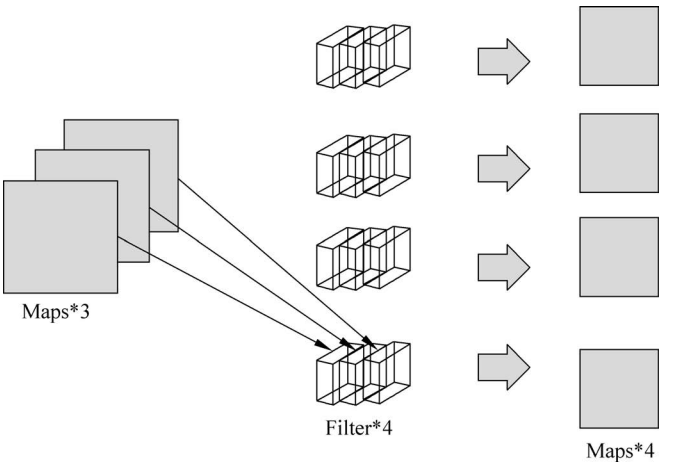



图 3-3　像素之间的卷积图

在逐通道卷积完成后，每个通道已经提取到了相应的特征图。但输入通道与输出通道是一一对应关系，每个输出通道仅与一个输入通道相关，无法参考其他通道中的特征信息。

1×1 卷积核的作用是在通道维度上进行卷积，通过对各通道之间的信息进行融合，形成最终的输出。这个过程可以被视为对特征图进行综合和调整，以更好地捕捉图像中不同通道之间的关系。因此，深度卷积的结构更注重在通道级别上的信息整合，使得网络能够更好地理解和表达图像中的内容。深度可分离卷积通过分离逐通道卷积和逐点卷积的方式，实现了对图像更全面、灵活的卷积操作。这种结构的设计旨在提高网络的参数效率和推理速度，使其更适用于各种计算机视觉任务，包括目标检测、图像分割和人脸识别等，深度可分离卷积的代码实现如代码 3-2 所示。

**【代码 3-2】** 深度可分离卷积。

```
def depthwise_separable_conv(kernel_size = (3, 3), stride = (1, 1), padding = (1, 1), in_
channels = 3, out_channels = 10):
    #逐通道卷积
    channel_conv = torch.nn.Conv2d(in_channels, in_channels, kernel_size, stride,
padding, groups = in_channels)
    #逐点卷积
    point_conv = torch.nn.Conv2d(in_channels, out_channels, (1, 1), stride, padding)
    #深度可分离卷积
    conv = torch.nn.Sequential(
        channel_conv,
        point_conv
    )
    return conv
```

深度卷积中参数的减少不仅会提高算法推理的速度，也关系到神经网络的推理精度和速度，接下来对深度卷积神经网络与标准神经网络之间的参数进行对比说明。

举例来说，考虑某一神经网络层的输入通道数为 8，卷积核尺寸为 3×3。通过标准

卷积操作，该层的输出通道数为 12。这意味着标准卷积操作所需的参数量为 $3\times3\times8\times12=864$。

深度可分离卷积与传统的卷积操作不同，它不是直接用一个大滤波器处理所有通道的输入数据，而是先对每个通道独立进行空间滤波（逐通道卷积），然后再通过 $1\times1$ 卷积（逐点卷积）来整合通道信息。假设输入通道数为 8，输出通道数为 12，则逐通道卷积中的参数量为 $8\times3=24$，逐点卷积中的参数量为 $1\times1\times8\times12=96$，深度可分离卷积中的参数量为 $8\times3+1\times1\times8\times12=120$，由此可以清晰地看到深度可分离卷积相对于标准卷积在参数上的显著降低。

这种差异主要来自深度可分离卷积的分离设计，使得每个通道都有独立的卷积核，而不是像标准卷积那样需要一个卷积核处理所有通道。这种设计不仅提高了参数的效率，还有助于提升网络的计算速度。在实际应用中，这样的优势使得深度卷积成为模型轻量化设计的首选，尤其在计算资源有限的场景下更为明显。

### 3.1.4　分组卷积与扩展卷积

分组卷积最开始被使用在经典入门卷积神经网络 AlexNet 上，用于减少运算量和参数量，深度可分离卷积中的逐通道卷积是分组卷积的一种特殊情况。在分组卷积中，卷积核被分为两组或多组，假设将卷积核平均分为 $G$ 组，每组包含 $C$ 个卷积核。同时，为保持一致性，图像中的通道也需要进行相应的分组，使得分组后的组数与卷积核分组数相等。

分组卷积的优势在于不仅能够缩短图像的训练时长，还能够充分调用硬件资源，将任务分配给多个 CPU 或 GPU 进行训练。这种并行性的设计有助于提高训练效率，特别是在处理大规模图像数据时，分组卷积的应用更加显著。

在分组卷积的具体操作中，卷积核和图像的通道被划分为不同的组。如果假设通道分组数为 2，输入图像尺寸为 $H\times W\times C$，输出图像尺寸为 $H\times W\times C'$，其中 $H$ 为图像高度，$W$ 为图像宽度，$C$ 为图像通道数。在分组卷积中，每个通道组负责对应通道的一部分卷积核进行滤波操作。例如，每个通道组内的数据与对应组的卷积核进行卷积，卷积后每个通道的数目为 $C/2$。通过将两个通道组的卷积结果叠加，最终得到与原卷积操作相同的输出通道数 $C'$，既不增加也不减少。分组卷积操作过程如图 3-4 所示。

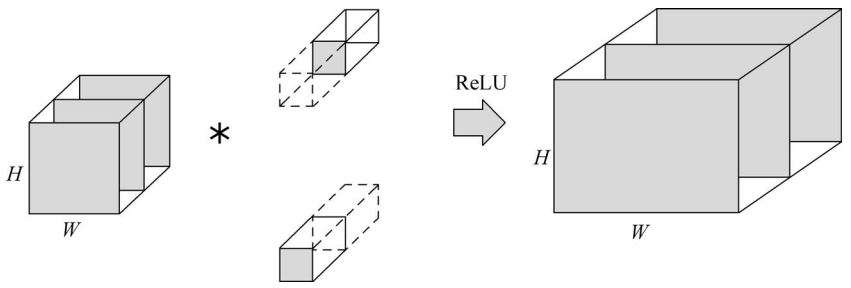



图 3-4　分组卷积操作过程

使用分组卷积有许多优势。首先，分组卷积可以有效减少网络训练所需的参数个数。其次，这种卷积方式可以充分调用硬件资源，实现高速的网络训练。对于更深层次

的网络模型，分组卷积可以充当系数矩阵的角色，减少卷积核参数之间的相关性，从而提高模型的训练效率，分组卷积的代码示例如代码 3-3 所示。

**【代码 3-3】** 分组卷积。

```
#定义卷积核尺寸
kernel_size = 3
#定义步长
stride = 1
#定义填充尺寸
padding = 1
#输入通道数
in_channels = 4
#输出通道数
out_channels = 10
#分组数
groups = 2
#判断输入通道是否能被分组数整除
assert in_channels % groups == 0, "in_channels must be divisible by groups"
#定义卷积
conv = torch.nn.Conv2d(
    in_channels,
    out_channels,
    kernel_size,
    stride,
    padding,
    groups = groups
)
```

另一种卷积方式是扩展卷积，也称为扩张卷积或膨胀卷积。这种方法通过在标准卷积核中引入空洞，以增加模型的感受野。与分组卷积和深度可分离卷积不同，扩展卷积的主要目标是减少图像中像素之间的相关性。扩展卷积的实现方式类似标准卷积，但其卷积核在提取感受野特征时有所不同。扩展卷积是为了解决卷积神经网络中下采样过程导致特征信息丢失的问题而提出的创新思路。虽然扩展卷积的感受野相对较大，但并非全部使用感受野中的像素点，因为相邻像素点之间存在强烈的相关性。因此，扩展卷积只提取感受野中的部分像素点。这一做法的直接好处是，相同大小的卷积核可以拥有更大的感受野，无须使用采样操作即可获取更多特征信息。扩展卷积操作示意图如图 3-5 所示。

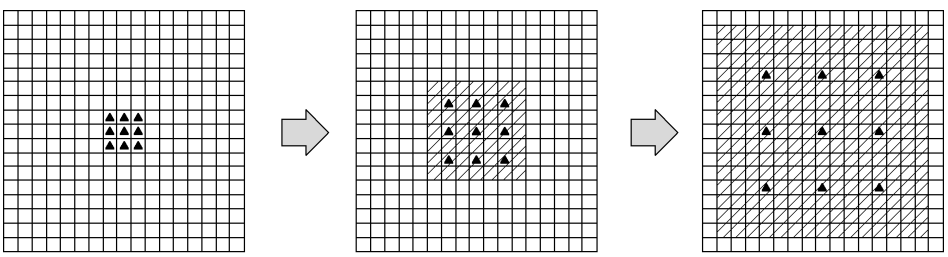



图 3-5 扩展卷积操作示意图

在扩展卷积中，扩张率是一个关键概念，它指的是卷积核中间隔点的数量。在图 3-5 所示中，网格中的点表示需要学习的值，而其他网格则需要填充为 0。举例来说，图 3-5(a) 中的扩张率为 0，此时卷积过程与标准卷积相同；而图 3-5(b)中的扩张率为 1，此时除了图中的点之外的区域需要填充为 0。扩展卷积示例如图 3-6 所示。

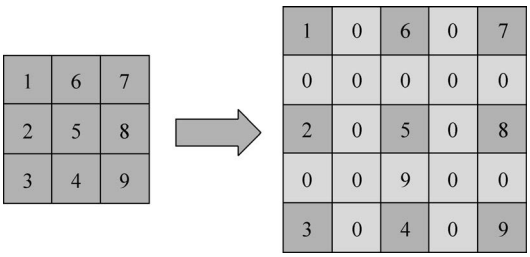



图 3-6　扩展卷积示例

扩展卷积代码实现如代码 3-4 所示。

**【代码 3-4】** 扩展卷积实现。

```
#定义扩展卷积
conv = torch.nn.Conv2d(
    in_channels,
    out_channels,
    kernel_size,
    stride,
    padding,
    #扩张因子默认为 1,可以通过控制扩张因子调整卷积核之间的间隙
    dilation = 2
)
```

虽然扩展卷积能够扩大神经网络的感受野，捕获更多上下文信息，但扩展卷积只提取图像中的部分像素点，导致局部信息的丢失，进而影响卷积结果的相关性。同样地，扩展卷积的独特取样方式也直接影响卷积后分类的效果。在使用扩展卷积时，需要权衡感受野的扩大和计算效率之间的关系，以确保算法在速度和准确性之间取得平衡。

## 3.2　可变形卷积技术

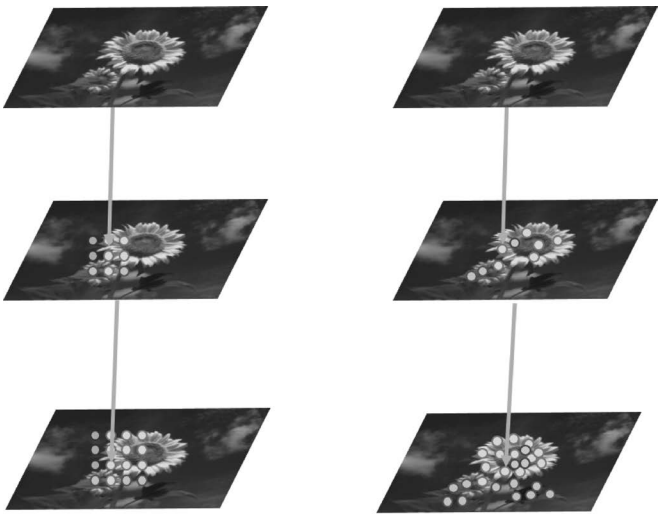
可变形卷积神经网络(Deformable Convolutional Networks，DCN)与标准卷积神经网络操作基本相似，但其学习的特征点有所不同。在标准卷积神经网络中，通过卷积核直接学习图像中的固定感受野。相比之下，可变形卷积神经网络专注于学习图像中特征点的变化趋势。考虑到图像是二维平面，描述特征点的变化趋势需要使用两个参数，因此可变形卷积神经网络相对于标准卷积神经网络多出一个层的参数。

本书选择单独介绍可变形卷积神经网络的原因是可变形卷积与深度卷积、分组卷积以及空洞卷积的实现方式有所不同。可变形卷积的核心创新在于其能够动态调整卷积核的采样位置，以更好地适应输入数据中的几何变化和复杂结构，而传统卷积及其变种则主要通过改变卷积核的结构或组合方式来提升模型的表现力。接下来将详细介绍可

变形卷积神经网络的实现原理及方法。在深入了解之前，先回顾一下卷积神经网络的基本概念，为后续内容的理解打下基础。

### 3.2.1 可变形卷积的数学基础

为了深入对比可变形卷积神经网络与标准卷积神经网络之间的差异，将对实际图像中的特征提取过程进行详细对比。卷积过程在图像处理中起着关键作用，下面将具体进行说明。标准卷积和可变形卷积示例如图 3-7 所示。



(a) 标准卷积过程　　(b) 可变形卷积过程

图 3-7　标准卷积和可变形卷积示例

在标准卷积中，卷积核通过固定的感受野对图像进行扫描。这个感受野是卷积核在图像上移动时覆盖的区域，通过卷积操作提取特征。在图 3-7(a)所示的标准卷积示例中，卷积核在图像上滑动，对每个感受野进行卷积操作，得到输出特征图。可变形卷积神经网络则引入了更加灵活的特征点学习方式，在可变形卷积中不再局限于固定的感受野，而是学习图像中特征点的变化趋势。这就意味着卷积核能够根据图像中的实际情况进行适应性调整，更好地捕捉局部特征。

具体到图 3-7(b)所示的示例中，可以清晰地看到可变形卷积的操作方式。卷积核不再只是简单地滑动，而是通过考虑特征点的变化，更加智能地提取图像的特征。这种灵活性使得可变形卷积在处理复杂图像结构时，具有更好的性能。通过对标准卷积和可变形卷积的对比，既可以更好地理解它们在特征提取过程中的异同，也为后续深入研究可变形卷积神经网络的实现原理提供了更清晰的背景。

可变形卷积神经网络主要的计算过程如下：

$$y(p_0)=\sum_{p_n\in R} w(p_n)\cdot x(p_0+p_n) \tag{3-5}$$

$$y(p_0)=\sum_{p_n\in R} w(p_n)\cdot x(p_0+p_n+\Delta p_n) \tag{3-6}$$

其中，$R$ 对应的是位置的集合；$p_n$ 是 $R$ 集合中位置的枚举，在可变形卷积神经网络中常规的网格 $R$ 可以通过一个偏移矩阵进行位置的扩展。

方程(3-5)和方程(3-6)描述了可变形卷积神经网络对输入特征图的处理方式。在方程(3-5)中，权重 $w(p_n)$ 与输入特征图的对应位置 $x(p_0+p_n)$ 相乘并求和，得到输出特征图的相应位置 $y(p_0)$。而在方程(3-6)中，引入了偏移量 $\Delta p_n$，使得卷积核在考虑了位置偏移的情况下进行特征提取。这样的处理方式使得可变形卷积网络能够更灵活地适应输入图像中的局部特征变化。

需要注意的是，$R$ 对应的位置集合的定义在可变形卷积神经网络中具有重要意义，而偏移矩阵的引入则增强了卷积核的感受野，使得网络能够更好地捕捉图像中的细节信息。这种位置的扩张和偏移的机制，使得可变形卷积神经网络在处理复杂图像结构时表现出更强大的特征提取能力。

可变形卷积代码实现如代码 3-5 所示。

**【代码 3-5】** 可变形卷积。

```
class DeformConv2D(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size, stride = 1, padding = 0,
dilation = 1):
        super(DeformConv2D, self).__init__()
        self.in_channels = in_channels
        self.out_channels = out_channels
        self.kernel_size = kernel_size
        self.stride = stride
        self.padding = padding
        self.dilation = dilation

        #可变形卷积需要额外的偏移量参数
        self.offset_conv = nn.Conv2d(in_channels, 2 * kernel_size[0] * kernel_size[1],
kernel_size, stride, padding, dilation)

        #初始化权重
        nn.init.constant_(self.offset_conv.weight, 0)
        nn.init.constant_(self.offset_conv.bias, 0)

    def forward(self, x):
        offset = self.offset_conv(x)          #获取偏移量
         out = F.deform_conv2d(x, self.weight, self.bias, offset, self.stride, self.
padding, self.dilation)
        return out
```

## 3.2.2　可变形卷积的网络结构

标准卷积操作虽然在很多任务中表现良好，但由于其对空间变形缺乏鲁棒性，导致其在处理具有变形特性的物体时表现不佳。标准卷积核的位置是固定的，这意味着它们无法适应输入数据中的形状变化和姿态变化。而在自然图像中，物体的形状和姿态通常是多变的，因此需要一种能够动态调整卷积核位置的方法来更好地捕捉这些变化。

可变形卷积的实现过程首先是通过添加一个专门用于生成偏移量的卷积层。这一偏移量卷积层的输入是原始特征图，而输出则是一个二维的偏移量场，形状与输入特征图相同。每个输出位置的偏移量(d$x$，dy)表示相对于标准卷积核位置的偏移。通过这个偏移量卷积层，网络可以学习到不同位置的最优偏移，从而自适应地调整卷积核的位置以捕捉输入数据的几何变形。偏移量卷积层的输出通道数通常是 2 乘以卷积核的大小，因为每个卷积核位置需要一个二维偏移量。

之后使用生成的偏移量来计算实际的采样位置。对于每个卷积核位置，使用偏移量调整标准卷积核的中心位置，从而得到一个浮点数形式的采样位置。这与标准卷积不同，后者的采样位置总是整点位置。浮点数形式的采样位置允许卷积核更精细地调整其位置，以更好地适应输入数据的形变。这一步的计算需要考虑偏移量的准确应用，以确保采样位置能够正确反映偏移后的卷积核位置。

由于实际的采样位置是浮点数，而不是标准卷积中的整数位置，需要使用双线性插值来计算该位置的特征值。双线性插值通过周围的四个整点位置的特征值加权求和得到实际采样位置的特征值。这种插值方法确保了在浮点位置上能够获得平滑且连续的特征值，有助于保持卷积操作的稳定性和有效性。双线性插值的权重由采样位置相对于周围整点位置的距离决定，因此能够精确地计算出浮点位置的特征值。

最后，使用计算出的采样位置上的特征值进行标准的卷积计算。具体来说，将卷积核的权重与经过双线性插值得到的特征值进行点积，得到输出特征图。这一步与标准卷积的主要区别在于特征值的来源，标准卷积直接使用整点位置的特征值，而可变形卷积则使用经过偏移和插值处理后的特征值。这使得卷积核能够根据输入数据的形状和姿态动态调整，从而更好地捕捉到多变的几何特征，提高网络的表示能力和性能。

可变形卷积神经网络通过引入特殊的卷积操作，使网络能够更灵活地捕捉图像中的局部特征变化。这种机制在处理需要考虑特征点偏移的任务时表现出更强大的特征提取能力。

可变形卷积神经网络结构单元示例如图 3-8 所示。

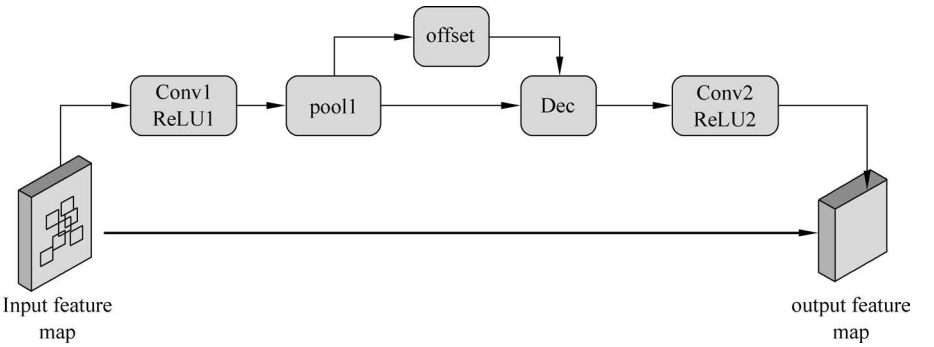



图 3-8　可变形卷积神经网络结构单元示例

### 3.2.3　可变形卷积的应用

可变形卷积神经网络应用示例如图 3-9 所示。

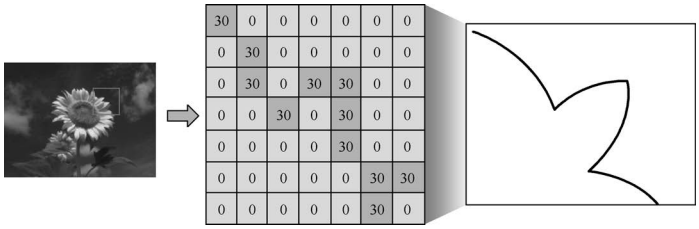



图 3-9　可变形卷积神经网络应用示例

在图 3-7 中，标准卷积中的感受野和采样位置在顶部特征图（图 3-7(a)）上都是固定的，而可变形卷积可以根据对象的规模和形状进行自适应调整（图 3-7(b)）。标准卷积过程中，主要对需要重点关注的目标区域利用池化方法获得目标图像关键特征。而在可变形卷积的过程中感兴趣区域（Region of Interest，RoI）池中网格结构的规则性不再成立，相反，部分偏离 RoI 池并移动到附近的对象前景区域上。可变形卷积增强了定位能力，特别是对于非刚性对象。

可变形卷积神经网络的核心创新在于引入可变形卷积和相对应的降维技术，使卷积核和最大特征输出能够根据输入数据的形变情况进行动态调整，从而提高网络的适应性。其核心组件为可变形卷积层（Deformable Convolution Layer）和可变形池化层（Deformable Pooling Layer）。其中可变形卷积层的卷积核形状是可学习的，通过学习从输入特征图中提取的偏移量进行调整。这使得网络能够在学习的过程中适应输入数据的空间变化。而可变形池化层引入了可学习的偏移量，用于调整池化窗口的位置，以更好地捕捉输入特征图的局部结构。DCN 可以嵌入卷积神经网络中，取代传统的卷积和池化操作。在处理形状和姿态复杂的目标时，DCN 能够提高检测性能。对于语义分割任务，DCN 可以更准确地分割图像中的不同物体，尤其是在处理非刚性形变时。同时，在人体姿态估计中，DCN 有助于更好地捕捉人体部位的形变信息，提高关键点检测的性能。

可变形卷积的卷积核作用过程示例如图 3-10 所示。

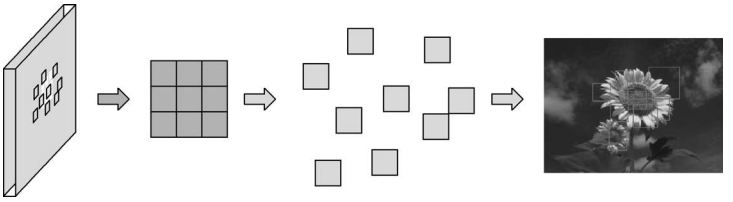


图 3-10　可变形卷积的卷积核作用过程示例

如图 3-10 所示，在处理复杂形状和非刚性变化的物体时，DCN 表现十分出色。

## 3.3　反卷积与目标分割

### 3.3.1　反卷积的数学原理

在讨论反卷积的概念时，大家往往会联想到标准卷积中的图像上采样实现过程。上采样是一种直接对图像特征图中的像素点进行扩充的方法，以提高图像分辨率。实际

上，反卷积可以看作是图像上采样的一种具体实现方式，也可称为标准卷积实现过程的逆运算。标准卷积和反卷积的具体实现过程可分为两个过程。

**1. 标准卷积的实现过程**

在标准卷积中，图像的像素点通过一个自定义的矩阵进行表示，以 $4\times4$ 的矩阵为例，表示为

$$\text{input}=\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{bmatrix} \tag{3-7}$$

卷积核的尺寸选择为 $3\times3$，卷积核矩阵如下：

$$\text{input}=\begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} \\ w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix} \tag{3-8}$$

卷积后图像特征图的尺寸计算公式为

$$C_1=\frac{n-k+2p}{s}+1 \tag{3-9}$$

式中：$C_1$ 为卷积后特征图的宽或高；$n$ 为图像的宽或高；$k$ 为卷积核的尺寸；$p$ 为填充的大小；$s$ 为移动的步长。

卷积完成后，通常会进行降维特征提取操作，该操作同样影响图像尺寸，计算公式为

$$C_2=\frac{n-f}{s}+1 \tag{3-10}$$

式中：$C_2$ 为池化后特征图像尺寸；$n$ 为图像的宽或高；$f$ 为池化核的尺寸；$s$ 为移动的步长。

经过计算可以得到输出的特征图尺寸，从实际输出图像的尺寸来看，图像经过卷积之后的特征图尺寸相比原来的图像缩小了。这是从标准卷积的计算过程来看的，实际过程是将卷积核对应的感受野进行了卷积的计算，因此提取特征后输出的特征图直接导致了图像分辨率的缩小。

**2. 反卷积实现过程**

反卷积是标准卷积的逆向操作，通过对卷积获取的特征图进行反卷积，从而实现图像卷积处理的逆向操作，尝试还原原始图像。需要注意的是，反卷积的结果并非原始图像的真实值，而是通过算法计算得到的像素点，通过添加缺失的像素点形成的。反卷积实现过程如图 3-11 所示。

反卷积的计算过程可以通过线性代数中的矩阵计算表示。

卷积操作公式表达如下：

$$Y(i,j)=\sum_m\sum_n X(i-m,j-n)\cdot K(m,n) \tag{3-11}$$

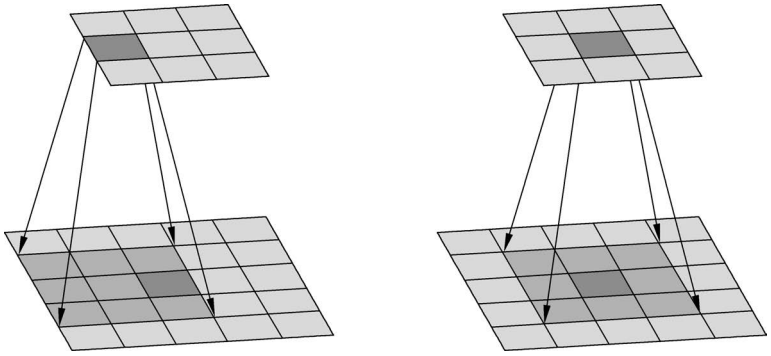


图 3-11　反卷积实现过程

其中，$i$ 和 $j$ 是输出的空间位置；$m$ 和 $n$ 是卷积核的索引。

反卷积操作公式表达如下：

$$\boldsymbol{X}'(i,j)=\sum_{m}\sum_{n}\boldsymbol{Y}(i+m,j+n)\cdot\boldsymbol{K}'(m,n) \tag{3-12}$$

其中，$\boldsymbol{X}'$是反卷积的输出；$\boldsymbol{K}'$是反卷积核。

将卷积和反卷积表达式写成矩阵形式：

$$\mathrm{vec}(\boldsymbol{Y})=\boldsymbol{K}_{\mathrm{mat}}\cdot\mathrm{vec}(\boldsymbol{X}) \tag{3-13}$$

$$\mathrm{vec}(\boldsymbol{X}')=(\boldsymbol{K}')_{\mathrm{mat}}\cdot\mathrm{vec}(\boldsymbol{Y}) \tag{3-14}$$

其中，vec(·)表示将矩阵展平成向量；$\boldsymbol{K}_{\mathrm{mat}}$ 和$(\boldsymbol{K}')_{\mathrm{mat}}$ 是对应的矩阵形式的卷积核和反卷积核。

通过对矩阵的逆运算，可以将输出图像与稀疏矩阵的转置进行矩阵运算，从而得到原始图像。

反卷积的代码实现如代码 3-6 所示。

**【代码 3-6】**　反卷积的代码实现。

```
deconv = nn.ConvTranspose2d(
            in_channels,
            out_channels,
            kernel_size = kernel_size,
            stride = stride,
            padding = padding
        )
```

### 3.3.2　全卷积网络

全卷积神经网络(Fully Convolutional Networks，FCN)是一种主要由卷积层构成的网络结构，其主要应用领域之一是图像分割。在全卷积神经网络中，网络的构成可以分为两个主要部分。首先，通过卷积神经网络实现对图像特征的提取；其次，通过反卷积操作实现图像的反向恢复，这两部分的结合呈现了全卷积神经网络的实现过程。全卷积神经网络实现过程如图 3-12 所示。

需要注意的是，全卷积神经网络与标准卷积神经网络在结构的后半部分存在明显差

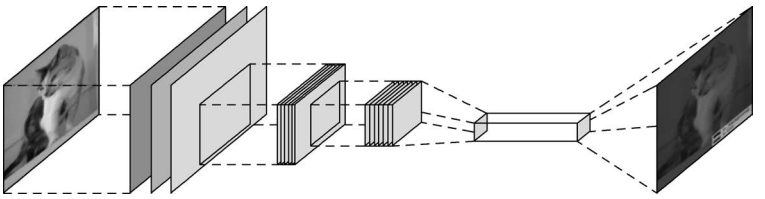


图 3-12　全卷积神经网络实现过程

异。标准卷积神经网络的后半部分结构通常包括全连接层和 Softmax 层，用于计算与原结果的交叉熵，而全卷积神经网络的后半部分则完全由卷积层构成。

### 3.3.3　反卷积在全卷积网络中的应用

标准卷积神经网络的单元结构执行顺序一般为输入图像、卷积操作、激活函数激活、池化操作，最终得到第 1 层卷积后的特征图。不同的网络结构执行的结构部分也不相同。在全卷积神经网络中，反卷积部分是标准卷积的逆向操作。其执行顺序包括获取已经卷积的特征图、进行反池化(反卷积中的一种操作)、通过激活函数激活，最终得到图像的反卷积恢复后的原图。这种反卷积操作有助于在图像分割任务中实现更精准的像素级别的分类。全卷积神经网络的这一特性使其成为图像语义分割等领域的重要工具。

全卷积神经网络中的卷积层可以被视为对全连接层的一种转换，通过这种转换实现了图像的反卷积结构。与标准卷积神经网络中的全连接层相比，卷积层在处理图像特征时更加注重空间信息的保留。这意味着卷积层能够有效地捕捉图像中的局部特征，并保持输入图像的空间结构。在标准卷积神经网络中，全连接层扮演着特征分类器的角色，用于标记一类目标的特征点。其标记过程实际上是将各类图像的特征点进行映射。在全连接层之后，通过 Softmax 函数进行图像的分类，将图像进行一维化处理。以一个具体的例子来说明，如果输入图像的尺寸为 224×224×3，经过多层卷积后到达全连接层的维度为 1×4096 的尺寸向量。

标准卷积过程如图 3-13 所示，全卷积过程如图 3-14 所示。

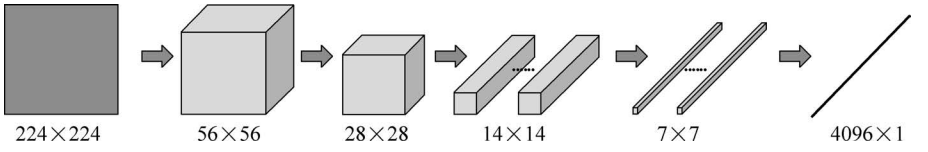



图 3-13　标准卷积过程

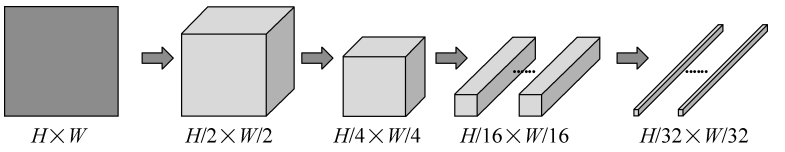



图 3-14　全卷积过程

这两种神经网络在处理图像时的不同阶段展示了它们的结构差异。全卷积神经网络通过卷积层实现了对输入图像的逐像素处理，从而更好地适应图像的空间特征。

这种结构使得全卷积网络在像素级别的任务中表现出色，特别适用于图像分割等应用场景。

在图 3-14 中，全连接层的输入部分被替换为二维特征图，经过多次卷积操作后，特征图的分辨率相对于原图降低。为了将特征图还原到原图的尺寸，采用了上采样操作，通过不断放大特征图，最终达到与原图相同的尺寸。图像分割示例如图 3-15 所示。

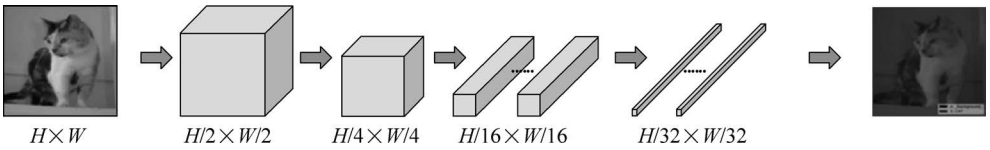



图 3-15　图像分割示例

虽然反卷积操作可以实现原图的部分恢复，但在数学推导中已经提到，反卷积并不是百分之百地还原原图，而是对缺失像素点位置进行恢复。

值得注意的是，反卷积的操作会导致结果的一定精度损失，即使输出的图像相对平滑。这是因为反卷积是通过插值等方式进行像素点的估计，而这个过程不是完全精确的。此外，反卷积在提取图像特征时是间接进行的，没有考虑像素点之间的空间关系，因此即使采用反卷积的恢复方式，仍然无法包含图像中各个像素点之间的关系。

这种恢复方式的缺点主要体现在精度和空间性方面，而在实际应用中，需要权衡这些缺点并根据具体任务的需求来选择适当的网络结构和恢复方式。

## 3.4　池化层的多重特性

池化操作通常在卷积神经网络中与卷积操作配合使用，可以起到调节数据维数的作用，并且具有抑制噪声、降低信息冗余、降低模型计算量、防止过拟合等特点。池化没有可以学习的参数，所以某种程度上与激活函数相似，池化在一维或多维张量上的操作与卷积层也有很多相似之处。

池化层是卷积神经网络中必不可少的组件，在网络训练中发挥着至关重要的作用。通过引入平移不变性、旋转不变性和尺度不变性等特性，池化层提高了网络的鲁棒性，使得网络对于图像轻微地平移、旋转或尺度变化都能保持稳定的识别结果。这种抗变性有助于网络更好地适应不同应用场景中的不同表现形式。

池化操作类似图像下采样，通过减小特征图的尺寸，有效降低了网络层中特征的维度。这不仅有助于防止过拟合，减轻了网络的负担，同时还能在降维的同时保持对图像特征的有效提取。此外，池化层还实现了非线性操作，为网络引入了非线性因素，增加了网络的表达能力。

实验研究表明，在图像分类中，池化层的平移不变性对于提高性能尤为关键；而在目标检测任务中，平移相等性则显得更为重要，即使目标发生平移，模型对应的预测结果也会相应地发生平移。然而，当输出存在较大的平移或旋转时，池化操作的效果可能会减弱，需要谨慎权衡这一特性在实际应用中的影响。

### 3.4.1 下采样与池化操作

下采样的目标是通过降低输入数据的尺寸，减少计算负担，提高计算效率，并在一定程度上提取输入数据的主要特征。通过对输入数据的局部区域进行采样，池化操作有助于保留关键信息，使得模型更专注于学习重要特征，提高泛化性能。这种操作不仅减小了内存需求，而且增强了模型对输入数据的平移不变性，使其更适应不同的数据变化，同时也扩大了感受野。在深度学习任务中，特别是在卷积神经网络中，下采样通过形成编码器-解码器结构，成为处理复杂任务如图像分类、语义分割等的重要组成部分。

常用的下采样方法主要包括平均池化与最大池化。最大池化通过在输入数据的局部区域中选择最大值，将该最大值作为池化后的输出。这个过程有助于保留输入区域中最显著的特征，提高模型对关键信息的敏感性。在数学上，最大池化的输出是池化区域中所有元素的最大值：

$$\text{Maxpooling}(x)=\text{Max}(x_{i,j}) \tag{3-15}$$

最大池化方法如图 3-16 所示。

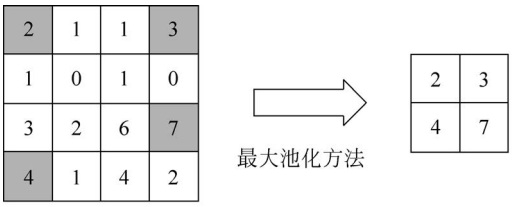



图 3-16 最大池化方法

最大池化的代码实现如代码 3-7 所示。

**【代码 3-7】** 最大池化。

```
maxpool = nn.MaxPool2d(
    kernel_size,
    stride = stride,
    padding = padding
)
```

平均池化通过在输入数据的局部区域中取平均值，将该平均值作为池化后的输出。这个过程有助于获取输入区域的整体信息，平滑图像特征，减轻噪声的影响。在数学上，平均池化的输出是池化区域中所有元素的平均值：

$$\text{Averagepooling}(x)=\frac{1}{m\times n}\sum x_{i,j} \tag{3-16}$$

平均池化方法如图 3-17 所示。

平均池化的代码实现如代码 3-8 所示。

**【代码 3-8】** 平均池化。

```
avgpool = nn. AvgPool2d(
    kernel_size,
```

```
        stride = stride,
        padding = padding
    )
```

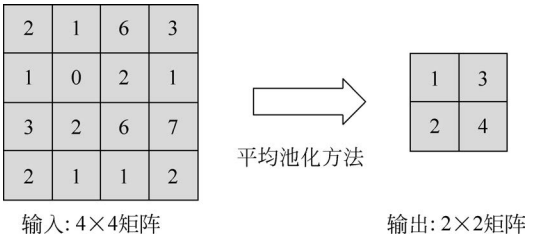



图 3-17　平均池化方法

这些池化操作通常应用于深度学习架构的编码器部分，用于降低输入数据的空间分辨率，提取关键特征。选取合适的池化方式，减少计算负担，在提升学习效率的同时在一定程度上防止过拟合。

## 3.4.2　上采样与特征扩充

上采样是一种图像处理或深度学习中的操作，其目的是将输入图像或特征映射的尺寸增加。这里的图像尺寸增加，并不是简单地放大，也不是下采样的逆操作，而是通过在图像中插入新的像素或在特征映射中插入新的特征值，以增加图像或特征映射的分辨率，这也是图像扩充的一种核心方法。

常见的上采样方法包括双线性插值、最近邻插值，以及 3.2 节提到的反卷积方法。其中，双线性插值和最近邻插值是最简单和最常用的上采样方法，反卷积则是一种专门用于卷积神经网络中的上采样方法。上采样的常用方法如以下两部分所示。

**1. 最近邻方法**

最近邻方法(Nearest Neighbor Method)是一种常用的插值方法，用于将低分辨率的图像或特征图放大到高分辨率的过程。该方法的原理是对于每个需要放大的像素，选择最近的像素作为其放大后的值。

最近邻插值方法通过放大低分辨率图像或特征图中的每个像素，使其达到所需的尺寸。在放大的像素矩阵中，采用最近的像素值作为其插值后的值。例如，将一个分辨率为 2×2 的像素矩阵放大到 4×4 的像素矩阵时，最近邻插值方法会将原低分辨率像素矩阵中的每个像素扩展为一个区域，并在放大的像素矩阵中，使用最近的像素值来填充扩展区域的值。最近邻方法如图 3-18 所示。

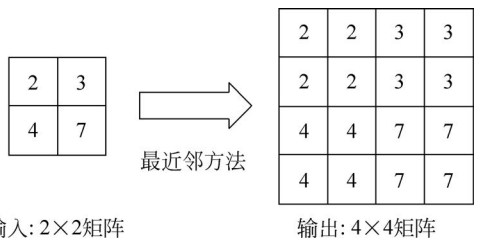



图 3-18　最近邻方法

最近邻方法具有简单、快速的优点，可以在不消耗太多计算资源的情况下实现图像或特征图的上采样。但最近邻方法存在着失真和锯齿等问题，因为它只考虑了离放大像素最近的像素的值，而没有考虑周围像素的信息。在一些对图像质量要求较高的任务中，如图像生成和图像超分辨率等任务，通常不推荐使用最近邻方法，而是使用更加精确的插值方法。

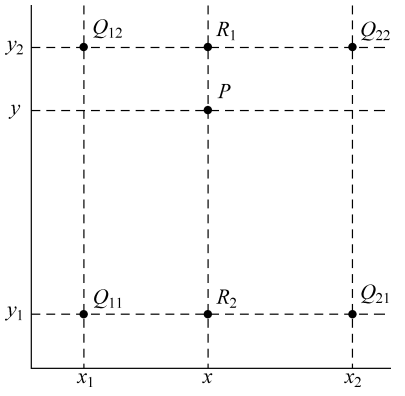



图 3-19　双线性插值法

**2. 双线性插值法**

双线性插值(Bilinear Interpolation)是一种常用的图像处理技术，用于在已知离散数据点的情况下，计算在任意位置的插值。它通常用于图像缩放、旋转等操作中，以提高图像的质量和精度。

双线性插值法的基本思想是，对于给定的插值位置，先在数据点坐标系中找到其所在的四个数据点，然后按照位置关系进行加权平均，得到插值结果。双线性插值法如图 3-19 所示。

在已知 4 个点 $Q_{11}$、$Q_{12}$、$Q_{21}$、$Q_{22}$ 后，要求得出其中间点 $P$ 的位置，则先在 $X$ 轴方向进行线性插值得到 $R_1=(x,y_1)$，$R_2=(x,y_2)$，方法如下：

$$f(R_1)\approx\frac{x_2-x}{x_2-x_1}f(Q_{11})+\frac{x-x_1}{x_2-x_1}f(Q_{21}) \tag{3-17}$$

$$f(R_2)\approx\frac{x_2-x}{x_2-x_1}f(Q_{12})+\frac{x-x_1}{x_2-x_1}f(Q_{22}) \tag{3-18}$$

再对 $R_1$、$R_2$ 在 $Y$ 轴方向进行线性插值得到 $P$：

$$f(P)\approx\frac{y_2-y}{y_2-y_1}f(R_1)+\frac{y-y_1}{y_2-y_1}f(R_2) \tag{3-19}$$

双线性插值法的优点在于它具有较高的计算效率和插值精度，同时可以避免出现锯齿状的插值结果。但需要注意的是，双线性插值法仅适用于在较小范围内的插值问题，如果插值范围过大，可能会导致插值结果的误差较大。此外，双线性插值法还需要保证数据点之间的间距足够小，以确保插值结果的精度。

## 3.5　全连接层的作用与影响

连接层是深度学习神经网络中的核心组件之一，在整个深度学习网络中主要起到分类器的作用。如果说卷积层、池化层和激活函数等操作是将原始数据映射到隐层特征空间的话，全连接层则起到将学到的分布式特征表示映射到样本标记空间的作用。

### 3.5.1　全连接层的基本原理

全连接层顾名思义，是指每一个节点都与上一层的所有节点相连，用来把前边提取

到的特征综合起来。由于其全相连的特性，一般全连接层的参数也是最多的。通过将前一层的所有节点与当前层的每个节点相连接，实现了对输入数据的全局信息整合。全连接层的连接方式如图 3-20 所示。

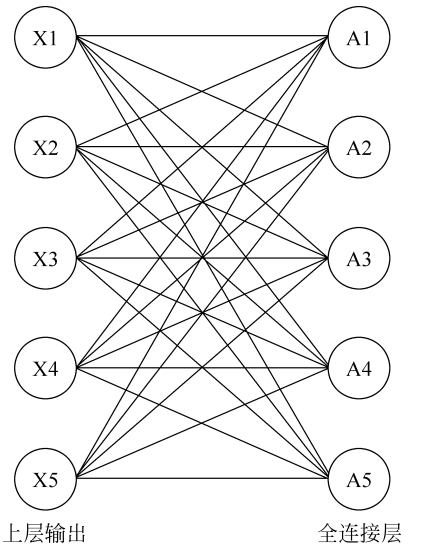


图 3-20　全连接层的连接方式

全连接层的全局信息整合使得模型能够捕捉输入数据中的全局关联性，有助于提高对整体特征的理解和学习。全连接层的激活函数引入了非线性变换，使得神经网络能够学习更为复杂的非线性映射，从而提高了模型的表达能力，适应各种复杂任务。

尽管全连接层具有强大的表示能力，但其参数量庞大、计算复杂度高的特点也带来了一些挑战。大量参数容易导致模型过拟合，尤其是在数据量较小的情况下。为了克服这些缺点，研究者们常常采用正则化技术、Dropout 等手段来提高模型的泛化能力。此外，随着深度学习的发展，人们也逐渐认识到全连接层并非在所有情况下都是必要的。在一些大规模图像任务中，卷积神经网络等结构更为常见，以降低计算复杂度，提高模型的效率。

全连接层在神经网络中的应用较为简单，在需要加入全连接层的位置确定上层网络或整体网络的输入维度，然后确定网络的输出维度，调用相应的函数接口即可在神经网络中添加一层全连接层。全连接层的层数和输出维度则需要根据相应任务的特性进行灵活设计。

综合而言，全连接层在深度学习中有其独特的优势，但在实际应用中需要根据任务的特点和数据规模进行灵活的选择和组合，以取得更好的性能和效率。

### 3.5.2　全连接层之间的关联性

多个全连接层的叠加使得神经网络能够学习更为复杂的输入与输出之间的映射关系。每一层都可以看作是对数据进行一次抽象和转换，层层堆叠可以帮助网络学习多层次、高级别的特征表示。全连接层连在一起的结构使得神经网络具备强大的表达能力，多个多项式的叠加能够逼近各种复杂函数。这对于处理高维输入数据、解决复杂问题非常重要。同时，多个全连接层通过不同的权重和非线性激活函数对输入进行组合，有助于提取输入数据中的各种抽象特征。这有利于网络学习到更有判别性的特征表示。

然而，与单层的全连接层类似，当网络规模变大时，全连接层的计算量会呈指数增长，而多个全连接层连接使用会加剧其参数量庞大，计算复杂度高的特点。应当根据实际要解决问题的整个网络结构的特点，合理设计网络。

## 3.6 数据标准化和正则化

在深度学习中，标准化和正则化是两个关键的概念，它们在模型训练过程中发挥着重要作用。标准化旨在调整输入数据的尺度和分布，提高模型的稳定性和收敛速度，而正则化则通过控制模型参数的大小，防止过拟合，促使模型更好地泛化到新数据。这两者的结合为深度学习模型的性能和鲁棒性提供了强大支持。在深入探讨这些概念的实际应用之前先思考一下，如果没有标准化和正则化，深度学习模型在面对多样化、大规模的数据时，可能会面临怎样的挑战。数据标准化有利于网络加速收敛、防止梯度爆炸并适应不同的分布，而防止过拟合则是正则化的成功，本节对数据的标准化和正则化进行描述。

### 3.6.1 数据标准化的重要性

在深度学习中，数据的标准化是一种特征缩放的关键策略，属于数据预处理的必要步骤。由于不同评价指标通常具有不同的量纲和单位，这种差异可能对数据分析产生影响。不进行标准化，则模型可能受到极端数据的影响，且如果模型需要计算样本间的距离，如欧氏距离或曼哈顿距离，一个特征域过大则所有距离计算都要取决于这个特征，与实际需要的理念相悖。

为了消除上述影响，可采用数据标准化处理，确保各指标处于相同的数量级。这样一来，经过处理后的数据更适合进行全面综合的对比评价，为模型训练和性能评估提供更可靠的基础。通过调整数据的尺度和分布，可以避免模型对不同特征的敏感性差异，解决梯度问题，加速收敛速度。

针对数据的不同分布情况，常用的数据标准化方法有线性归一化、裁剪归一化与标准差标准化，该三个方法如下所示。

**1. 线性归一化**

线性归一化也被称为最小-最大规范化或者离散标准化，是对原始数据的线性变换，将数据值映射到[0,1]区间内，用公式表示为

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3-20}$$

线性归一化保留了原来数据中存在的关系，是消除量纲和数据取值范围影响的最简单的方法。代码实现如代码 3-9 所示。

**【代码 3-9】** 线性归一化。

```
1  def MaxMinNormalization(x,Max,Min)
2      x = (x - Min) / (Max - Min)
3      return x
```

从理论上讲，可以将数据通过线性变换映射到任意范围中，比如有时希望将数据映射到[−1,1]区间，则可采用公式

$$x' = \frac{2 \times (x - \min(x))}{\max(x) - \min(x)} - 1 \tag{3-21}$$

进行线性归一化。

线性归一化虽然十分方便易用，但对于极端数据非常大的情况，会使较多数据趋于0，这对原数据的分布进行了较大的改变，不是想要的结果，为此引入裁剪归一化。

**2. 裁剪归一化**

裁剪归一化在严格意义上讲并不属于一种归一化，其本质是在归一化前或归一化后的一个操作，旨在将极端数据进行裁剪处理，使数据分布更加合理，不影响其他操作。但裁剪归一化会在一定程度上影响数据集的表现能力，因此要根据数据集选择合理的裁剪范围，达到较好的效果。

**3. 标准差标准化**

标准差标准化也叫 Z-score 标准化或零-均值归一化，是将数据变换为均值为 0、标准差为 1 的分布，变换后依然保留原数据分布。用公式表示为

$$x' = \frac{x - \mu}{\delta} \tag{3-22}$$

$$\delta = \sqrt{\frac{\sum_{i=1}^{N} (x - \mu)^2}{N}} \tag{3-23}$$

其中，$\delta$ 为数据的标准差；$\mu$ 为数据的均值。

标准差标准化主要对数据赋予了原始数据均值和标准差，经过处理的数据符合标准正态分布，能够符合大部分场景的需求。与其他标准化方法相同，这种方法也是一种线性变换。

## 3.6.2　批标准化与层标准化

对于整个深度神经网络而言，仅对输入数据进行标准化不足以让每一层的输入都保持在一个良好的分布，在经过一层网络后数据的分布又会随之改变，因此在层之间与批之间，也要引入标准化的概念，保持隐藏层中数值的均值、方差不变，让数值更稳定，为后面网络提供坚实的基础。

层标准化和批标准化的基本方法与标准差标准化的方法类似：

$$x' = \frac{x - \mu}{\sqrt{\delta^2 + \varepsilon}} \tag{3-24}$$

$$\delta^2 = \frac{\sum_{i=1}^{N} (x - \mu)^2}{N} \tag{3-25}$$

其中，$\delta$ 为数据的标准差；$\mu$ 为数据的均值。与标准差标准化不同的是，一般会在分母部分的方差使用时加上一个极小的正值 $\varepsilon$，如 $\varepsilon = 10^{-8}$，以人为地强制性避免分母为 0 时产生的各种问题，如除以零所产生的问题以及梯度未定义的问题。

层标准化与批标准化的主要区别在于标准化的对象和范围。层标准化对每个隐藏

层的输出进行标准化，使得每个样本在单个特征维度上都具有零均值和单位方差，而批标准化则对整个批次中的所有样本在每个特征维度上进行标准化。此外，层标准化通常应用在循环神经网络等结构中，而批标准化在卷积神经网络等结构中更为常见。层标准化与批标准化如图 3-21 所示。

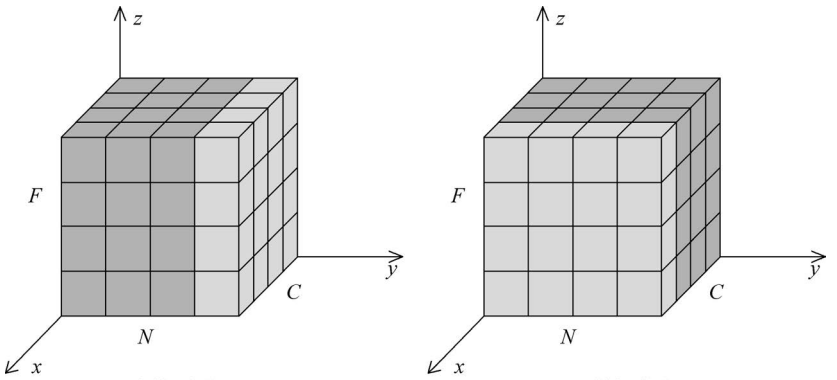



图 3-21　层标准化与批标准化

图 3-21 中，$N$ 代表样本轴，$C$ 代表通道轴，$F$ 是每个通道的特征数量。图 3-21(a)所示的层标准化，进行的是同一样本不同通道的标准化；图 3-21(b)所示的批标准化，进行的是不同样本同一通道进行的标准化。

选择使用层标准化还是批标准化通常取决于具体的任务和网络结构。在一般深度学习任务中，批标准化是较为常用的选择，因为它可以提高训练的稳定性和加速收敛性。然而，对于循环神经网络等结构，层标准化可能更适用，有助于缓解梯度消失问题。在标准化后，还需要进行缩放与平移，得到最终的标准化结果：

$$y_{ij} = \gamma x'_{ij} + \beta \tag{3-26}$$

其中，$y_{ij}$ 是经过平移与缩放后最终得到的标准化结果。平移与缩放的目的是采取线性变换保证数据的表达能力，而式中的参数 $\gamma$ 与 $\beta$ 则是需要进行训练学习的参数。

在实践中，研究者和工程师通常根据任务的特性和网络的结构灵活选择使用这两种正则化方法，甚至在某些情况下同时使用它们以取得更好的效果。层标准化的方法还通过规范数据分布，使网络在一定程度上防止过拟合。

### 3.6.3　正则化方法

深度学习的正则化方法旨在应对过拟合问题，即模型过度适应训练数据而在未见过的数据上表现不佳。正则化通过一系列技术手段，寻求在训练过程中防止模型对训练数据的过度拟合，从而提高模型在未知数据上的泛化性能。过拟合的风险在于模型可能会学到训练数据中的噪声和细节，而正则化方法的关键目标是使模型更具有泛化能力，适应更广泛的数据分布。

在正则化的框架下，可以理解为进行结构风险最小化，即通过在经验风险项后引入表示模型复杂度的正则化项或惩罚项。经验风险是指模型在训练数据上的误差，而结构

风险将这一概念进一步扩展为同时考虑模型的复杂度。这种综合的风险最小化方法旨在保持模型在训练数据上的性能同时，通过降低模型复杂度来提高泛化能力。简而言之，正则化方法不仅关注训练误差最小化（经验风险最小化），还考虑了模型的复杂度，以更全面地指导模型的学习过程。

正则化方法的主要类别包括权重正则化、Dropout、数据增强、标准化等，4 种方法具体如下所示。

（1）权重正则化。

权重正则化通过在损失函数中添加正则项进行正则化，L1 正则化和 L2 正则化可以看作是损失函数的惩罚项。所谓“惩罚”是指对损失函数中的某些参数做一些限制。L1 正则化是权值的绝对值之和：

$$\| x \|_1 = \sum_{i=1}^{n} | x_i | \tag{3-27}$$

L1 正则化可以产生稀疏权值矩阵，即产生一个稀疏模型，可以用于特征选择。

L2 正则化是向量元素绝对值的平方和的平方根，它会使优化求解稳定快速，使权重平滑：

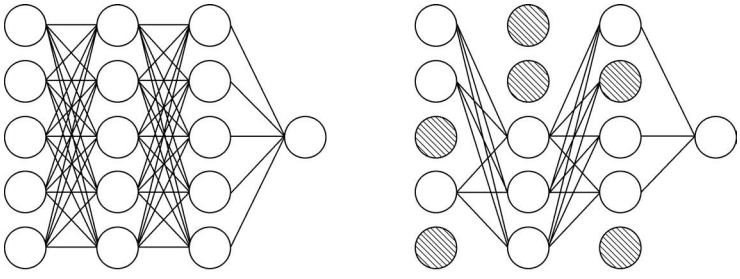
$$\| x \|_2 = \sqrt{\sum_{i=1}^{n} | x_i |^2} \tag{3-28}$$

L1 正则化与 L2 正则化是最简便快捷的正则化方法，均作用于损失函数，都能够在一定程度上防止数据过拟合。

（2）Dropout 正则化。

Dropout 是一种深度学习中常用的正则化技术，旨在减轻神经网络的过拟合问题。它的原理是在训练过程中，随机关闭神经网络中的一些神经元，使得每个神经元都有可能在一个小批次的训练中被暂时忽略。这样的随机性迫使网络不依赖特定的神经元，有效地增加了模型的鲁棒性，减少了对某些特定特征的过度依赖。

Dropout 的作用不仅在于减少神经元间的共适应性，还有助于训练更加稳健和泛化能力强的模型。通过防止网络对特定输入模式的过度适应，Dropout 可以被看作是一种“集成学习”的形式，让不同子网络学到不同的特征表示。Dropout 示意图如图 3-22 所示。



(a) 正常网络结构　　(b) Dropout后网络结构

图 3-22　Dropout 示意图

在实践中，Dropout 通常应用在隐藏层，其参数表示每个神经元被关闭的概率，一般设置在 0.2～0.5。这种简单而有效的正则化手段已被广泛应用于各种深度学习任务，为模型的性能和泛化提供了重要的改善。

(3) 数据增强。

数据增强是一种用于改进深度学习模型性能和泛化能力的技术，通过对训练数据集中的样本引入随机变换来扩充数据。这些变换可以包括平移、旋转、缩放、翻转等，从而生成更多样化的训练样本，使模型更好地适应各种输入变化。数据增强的核心思想是通过引入变异性，使模型在训练过程中更好地捕捉和学习数据的不同方面。

数据增强有助于避免过拟合的原因在于它减轻了模型对训练数据的过度依赖。当模型在训练过程中接触到更多变化和噪声的数据时，它更有可能学到更鲁棒的特征表示，而不是仅仅记住训练集中的特定样本。通过引入更多样的数据情况，数据增强使得模型更具泛化能力，更有可能在未见过的数据上表现良好。这使得模型能够更好地适应实际应用中的各种变化和噪声，提高模型的稳健性。在图像分类、目标检测等任务中，数据增强常常是训练深度学习模型的标准实践之一，对模型的性能提升至关重要。

(4) 标准化。

批标准化和层标准化通过规范化隐藏层的输出，有助于训练过程的稳定性和泛化性能，详见 3.6.2 节内容。

选择何种正则化方法取决于问题的性质和数据的特点。权重正则化适用于降低模型复杂度，而数据增强则适用于增加训练数据的多样性，层标准化与批标准化则较多地应用于各种网络。在实践中，通常需要综合考虑并尝试不同的正则化技术，以找到最适合特定任务的组合。

## 3.7 本章小结

第 3 章深入探讨了卷积神经网络的核心组成部分及其运作原理。本章从卷积层的基础概念出发，详细介绍了卷积操作的多种形式。这些内容不仅涵盖了卷积核、填充、步长等基本概念，还展示了不同类型的卷积操作如何有效地应用于图像处理和特征提取。同时，本章通过介绍反卷积和目标分割，展示了卷积神经网络在图像重建和分辨率提升中的应用。反卷积特别在上采样过程中发挥着重要作用，用于处理图像分辨率的改变与恢复。

池化层作为卷积神经网络的另一核心组件，其多重特性也在本章中被详细讨论。通过阐明池化操作如何降低网络的复杂度并增强模型的泛化能力，读者可以更好地理解卷积神经网络在减少过拟合和提升效率方面的机制。

本章旨在为读者提供一个关于卷积神经网络的基本构架和操作原理的全面而深入的理解框架。通过综合探讨各部分，本章不仅加深了对卷积神经网络结构和功能的理解，也为实际应用中的问题解决提供了理论支撑。这将有助于读者在后续的学习和研究中，更好地运用卷积神经网络解决复杂的图像处理任务。