第 2 章　卷积神经网络与深度学习

2.1　神经元模型

2.1.1　线性模型

满足下式输入输出关系的模型，称为线性模型：

$$y = \sum_i w_i x_i + b \tag{2-1}$$

式（2-1）中，权重 w_i，偏差 b 为线性模型的参数。当输入只有一个的时候，x 和 y 呈线性关系。如果有 n 个输入，x 和 y 就形成了 $n+1$ 维空间中的一个超平面。在一个线性模型中，通过输入得到输出的函数被称之为一个线性变换。线性模型经过任意次线性模型组合之后仍然是线性模型。

前向传播算法是一种典型的线性模型，常用于深度学习中，如下式所示：

$$a^{(1)} = xW^{(1)}，\ y = a^{(1)}W^{(2)} \tag{2-2}$$

式（2-2）中，x 是输入，W 是参数。通过对下式的推导，可得到 y 和 x 之间的关系：

$$y = (xW^{(1)})W^{(2)} \tag{2-3}$$

$$y = x(W^{(1)}W^{(2)}) = xW' \tag{2-4}$$

$$W' = W^{(1)}W^{(2)} = \begin{bmatrix} W^{(1)}_{1,1} & W^{(1)}_{1,2} & W^{(1)}_{1,3} \\ W^{(1)}_{2,1} & W^{(1)}_{2,2} & W^{(1)}_{2,3} \end{bmatrix} = \begin{bmatrix} W^{(2)}_{1,1} \\ W^{(2)}_{2,1} \\ W^{(2)}_{3,1} \end{bmatrix} =$$

$$\begin{bmatrix} W^{(1)}_{1,1}W^{(2)}_{1,1} + W^{(1)}_{1,2}W^{(2)}_{2,1} + W^{(1)}_{1,3}W^{(2)}_{3,1} \\ W^{(1)}_{2,1}W^{(2)}_{1,1} + W^{(1)}_{2,2}W^{(2)}_{2,1} + W^{(1)}_{2,3}W^{(2)}_{3,1} \end{bmatrix} = \begin{bmatrix} W'_1 \\ W'_2 \end{bmatrix} \tag{2-5}$$

$$y = xW' = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} W'_1 \\ W'_2 \end{bmatrix} = \begin{bmatrix} W'_1 x_1 + W'_2 x_2 \end{bmatrix} \tag{2-6}$$

式（2-6）中，W'是新的参数。式（2-6）所示的前向传播算法就是一个典型的线性模型。即使神经网络有两层（不包括输入层），它还是一个线性模型。进而得知，不管经过多少次线性变换，神经网络依然是线性模型。线性模型的缺点是能够解决的问题是有限的。

2.1.2　激活函数

为了解决线性模型的局限性，人们引入了激活函数的概念。激活函数就是把输入转换成输出。例如输入 a，通过激活函数得到输出 $f(a\times w_1+b)$，其中 $f(\cdot)$ 就是激活函数。

将“n”个输入记为 X_1、…、X_n，相应的权重为 W_{k1}、…、W_{kn}，并设定偏差为 b。用权重与它对应的输入相乘，然后加上偏差得到输出值为 u，如下式所示：

$$u = \sum_n W_n \times X_n + b \tag{2-7}$$

式（2-7）中，u 为线性模型的输出，激活函数就是作用在 u 上，即 $f(u)$，其从神经元接收其输出，如 $yk=f(u)$。常用的激活函数有 Sigmoid，ReLU 和 Softmax 三种。

1. Sigmoid

Sigmoid 函数是最常用的激活函数之一，其函数形式如图 2-1 所示，它被定义为下式：

$$sigmoid(x) = 1/(1 + e^{-x}) \tag{2-8}$$

从图 2-1 中得知，Sigmoid 变换总是会产生一个 0 到 1 之间的数据。

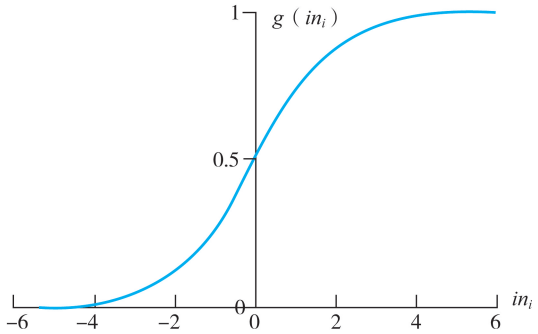


图 2-1　Sigmoid 曲线图

2. ReLU

ReLU 函数又称为整流线性单位。该函数定义为式（2-9）：

$$f(x) = \max(x, 0) \tag{2-9}$$

式（2-9）中，当 $x>0$ 时，函数的输出值为 x；当 $x\leqslant 0$ 时，输出值为 0。函数图如图2-2所示。

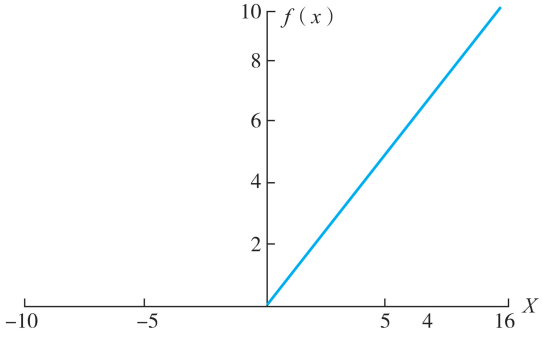


图 2-2　ReLU 函数

使用 ReLU 函数最大的优势是对大于 0 的所有输入来说，都有一个固定的导数值。常数的导数计算简单，加快了网络训练速度。

3. Softmax

Softmax 激活函数通常用于输出层，用于分类问题。它与 Sigmoid 函数是很类似的，唯一的区别就是其输出被归一化，总和为 1。

Softmax 函数形式如下式：

$$f(z)=\frac{e^{z_j}}{\sum_k e^{z_k}} \tag{2-10}$$

Softmax 函数实现的功能如图 2-3 所示。

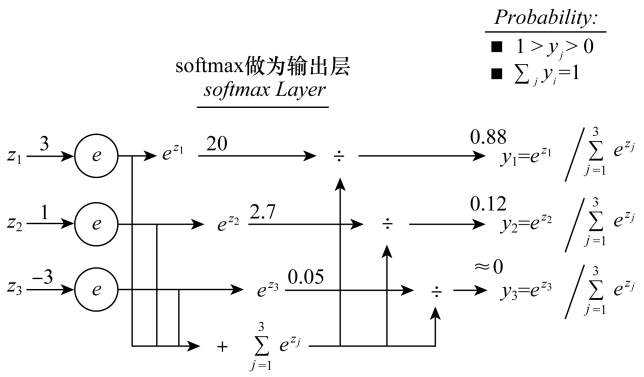


图 2-3　Softmax 功能示意图

Softmax 可以把一个 k 维的实值向量（a_1，…，a_i，…）映射成（b_1，…，b_i，…）。其中，b_i 是一个 0～1 的常数，Softmax 函数可根据 b_i 的大小来进行多分类的任务。

后文中用到的网络模型，在卷积层中使用 ReLU 激活函数，在网络的最后一层使用 Softmax 激活函数。

2.1.3　损失函数

深度学习分为有监督学习和无监督学习。训练数据有特征输入和标签输出就是有监督学习，训练数据不带标签就是无监督学习。

由于训练数据的 data 都是带有标签的数据，所以有监督学习可以根据损失函数来判断预测的准确程度。比如，下式就是一个对数损失函数：

$$L(\hat{y}^{(i)},\ y^{(i)})=-(y^{(i)}\log(\hat{y}^{(i)})+(1-y^{(i)})\log(1-\hat{y}^{(i)})) \tag{2-11}$$

式（2-11）中，$\hat{y}^{(i)}$ 是对于训练样本 $x^{(i)}$ 的预测结果，$y^{(i)}$ 是预期正确的输出值。通过比较 $\hat{y}$ 与 y 之间差值的大小，就可以权衡模型的好坏。当差值比较大时，就说明网络的输出结果不理想。相反，如果差值很小甚至等于 0，说明网络模型很理想，可以输出正确的答案。

式（2-11）是对单个训练样本进行定义的损失函数，而要权衡网络模型则往往要对整个训练集进行预测。有样本的损失函数为

$$J(w,\ b)=\frac{1}{m}\sum_{i=1}^{m}L(\hat{y}^{(i)},\ y^{(i)})=-\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(\hat{y}^{(i)})+(1-y^{(i)})\log(1-\hat{y}^{(i)})] \tag{2-12}$$

式（2-12）在本质上就是对每个训练样本的“损失”进行求和，然后求其平均值，这种整个训练集的损失函数被称为成本函数，损失越大代表预测越不准确。

2.1.4　过拟合

过拟合（overfitting）指在学习过程中经过训练的模型，由于每一个训练样本中存在的随机噪声部分，导致模型没有学习到样本中通用的部分，进而导致模型在测试集上的效果远低于训练集。

标准定义：给定一个假设空间 H、一个假设 h 属于 H，如果存在其他的假设 h' 属于 H，使得在训练样例上 h 的错误率比 h' 小，但在整个实例分布上 h' 比 h 的错误率小，那么就说假设 h 过拟合。

过拟合最直观的表现就是训练精度特别高，但是测试精度特别低，即两者相差特别大。

训练数据少、模型过度复杂都是出现过拟合问题的常见原因。下列四种方法可以有效地避免过拟合问题。

1. Early stopping

在对模型的训练过程中总会用到一些迭代的方法，如本文用到的梯度下降算法。随着训练的进行，当发现迭代至某轮，训练误差逐渐下降，但预测误差却在上升时，应提前结束训练，获得模型。

2. 数据集扩增

训练数据过少，容易出现过拟合。反之，扩增数据集，使训练样本数量庞大并且多样化，可以有效避免过拟合，并且增加模型的鲁棒性。

3. 正则化

正则化方法是指在进行目标函数或代价函数优化时，在目标函数或代价函数后面加上一个正则项，一般有 $L1$ 正则化、$L2$ 正则化等。

$L1$ 正则化：$L1$ 正则是基于 $L1$ 范数，在目标函数后面加上参数的 $L1$ 范数和项，即加上参数绝对值和与参数的积项：

$$C=C_0+\frac{\lambda}{n}\sum_{w}|w| \tag{2-13}$$

式（2-13）中，C_0 代表原始的代价函数，n 是样本的个数，λ 是正则项系数，最后一项即为 $L1$ 正则项。

在计算梯度时，w 的梯度为

$$\frac{\partial C}{\partial w}=\frac{\partial C_0}{\partial w}+\frac{\lambda}{n}\mathrm{sgn}(w) \tag{2-14}$$

式（2-14）中，sgn 是符号函数，使用下式对参数进行更新：

$$w = w + \alpha \frac{\partial C_0}{\partial w} + \beta \frac{\lambda}{n} \operatorname{sgn}(w) \tag{2-15}$$

式（2-15）中，当 w 为正时，更新后的 w 会变小；当 w 为负时，更新后的 w 会变大。因此，$L1$ 正则项是为了使原先处于零（即 $|w| \approx 0$）附近的参数 w 往零移动，使得部分参数为零，从而降低模型的复杂度，防止过拟合，提高模型泛化能力。

$L2$ 正则化：$L2$ 正则是基于 $L2$ 范数，在目标函数后面加上参数的 $L2$ 范数和项，即加上参数的平方和与参数的积项：

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2 \tag{2-16}$$

式（2-16）中，C_0 代表原始的代价函数，n 是样本个数。与 $L1$ 正则项前面的参数不同，$L2$ 项的参数乘了 1/2，这是为了便于计算，因为平方项求导有个 2。λ 是正则项系数，最后一项即为 $L2$ 正则项。

在 $L2$ 正则化中最后使用下式对模型参数进行更新：

$$w = w + \alpha \frac{\partial C_0}{\partial w} + \beta \frac{\lambda}{n}(w) \tag{2-17}$$

式（2-17）中，$L2$ 正则项起到使得参数 w 变小的效果。更小的参数值意味着模型的复杂度更低，对训练数据的拟合刚刚好，不会过分拟合训练数据，从而提高了模型的泛化能力。

4. Dropout

Dropout 是通过修改网络本身结构来避免过拟合的。在训练过程的一次迭代中，Dropout 可按比例（可以设置比例为 1/2、1/3 等）随机删除一些隐藏层神经元，即认为这些神经元不存在，在反向传播对参数进行更新时，不会对删除的神经元进行更新。在下一次迭代中，Dropout 同样随机删除一些神经元，直至训练结束。

2.1.5 Batch

Batch 是批的意思，相当于数据集的子集，代表数据集的一部分内容。在卷积神经网络中，处理的数据是分批进行的，“batch_size”是每批处理的样本的个数。

对 Batch 的选择决定了下降的方向。如果数据集比较小，可以采用全数据集的形式，这是由于全数据集确定的方向能够更好地代表样本总体，因此可以使数据更准确地朝向极值的方向。

对于更大的数据集，随着数据的海量增长和内存限制，一次性载入所有数据变得不可实现。此时选择合适大小的 Batch 用于训练，训练算出来的梯度与用全部数据训练出来的梯度是一样的。

2.2　卷积神经网络

卷积神经网络（Convolutional Neural Network，CNN）的出现，是受到了休伯尔和维厄瑟尔对猫视觉皮层电生理研究的启发。近年来，CNN 在音频处理和图像处理等领域都占有绝对的地位。如图 2-4 所示是一个完整的 CNN 的结构图，此时可以清晰地看到层与层之间的位置关系。

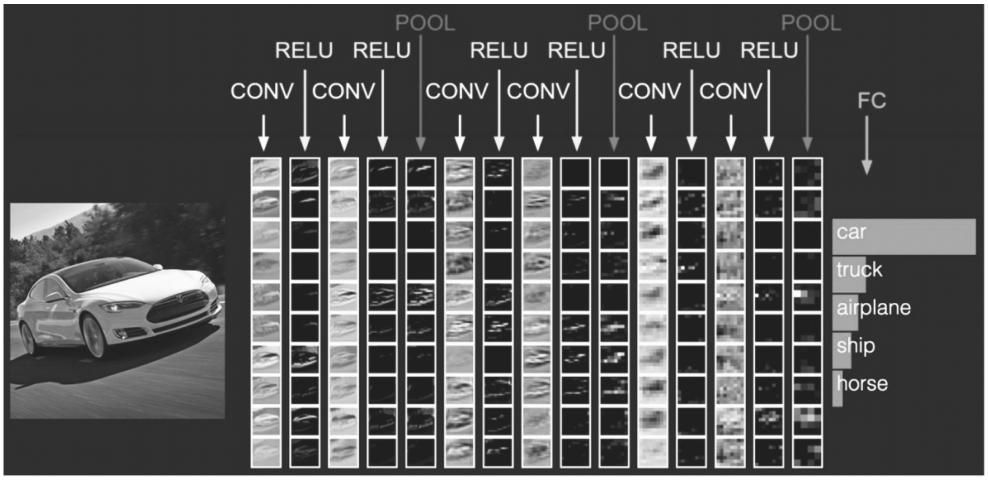


图 2-4　卷积神经网络基本结构图

CNN 包括以下几种类型的结构层。

（1）输入层：用于数据的输入。

（2）卷积层：对矩阵进行卷积运算，达到特征提取的目的。

（3）激励层：使用激励函数对提取到的特征添加非线性映射。

（4）池化层：对下采样，对提取到的特征图像进行稀疏处理，减小数据运算量。

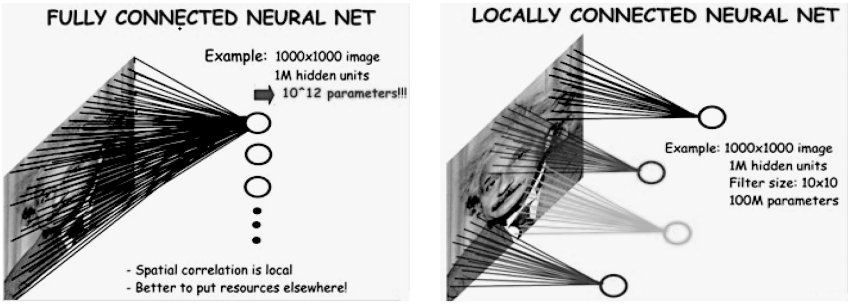
（5）全连接层：是 CNN 隐藏层的最后一层，其作用是重新拟合，整合提取到的特征信息。

（6）输出层：用于输出结果。

2.2.1　局部感知

因为在 CNN 中存在非常庞大的计算量，需要使用 GPU 进行加速计算，而且非常耗时。因此需要使用一种在不影响计算结果的情况下有效减少计算量的方法，其中一种就是局部感知。研究表明，人对外界的感知是从部分到全部的，而图像在空间上的联系也有着类似的关系。相邻的像素之间联系密切，距离远的像素之间联系稀疏。因此，只需要让每个神经元对图像的局部进行连接，在更高的层次将局部的信息拟合起来，就可以达到等同于全连接特征提取的效果。局部连接的思想源于生物学中视觉系统的结构。如图 2-5 所

示，图 a 为全局感知，图 b 为局部感知。



（a）全局连接神经网络　　（b）局部连接神经网络

图 2-5　感知示意图

在图 2-5（a）中，假如输入是 1000×1000 像素的图片，其可以表示为 1000×1000 个二维向量。再假设下一层隐藏层（即输入层和输出层之间的一些用于内部操作计算的层）中的神经元个数与输入层一样，即 1000×1000，则在上图的全连接方式中，一个神经元的权值数量为 1000×1000 个，即有 1000×1000 个神经元，那么隐藏层的权值数量为 1 000 000×1 000 000 个。仅仅是第一层隐藏层的权值数量，就已经非常之大了。

图 2-5（b）是局部连接的方式，每个神经元只与 10×10 个像素值相连，那么隐藏层权值数量为 1 000 000×100 个。而 10×10 个像素值对应的是 10×10 个权值，其实际上就是对图像进行卷积。

2.2.2　权值共享

虽然局部感知减少了一部分参数，但是剩下的参数依然过多，此时就用到了另一种方法：权值共享。在上面的局部感知中，每个神经元都有属于自己的 100 个权值，整个第一层有 1 000 000 个神经元，假设这 1 000 000 个神经元的 100 个权值相等，则权值只剩下 100 个了。因此无论输入图像有多大，经过局部感知和权值共享这两大利器后，两层间连接只需要求解 100 个权值，即 100 个参数。

此时可以把这 100 个参数作为卷积核去对图像进行特征提取，该方法与位置无关，其原理是：图像的一部分统计特性与其他部分是一样的。这表明在这一部分提取的特征可以用在其他部分上，所以在整张图片的不同位置上，都可以使用同样的学习特征。

因为只使用了一个卷积核（即滤波器），只提取到一种特征，所以其不能完全代表图像的特征信息。为了提取图像完整的特征信息，需要添加不同的滤波器，去提取不同的特征。假设用，100 种不同大小、参数不一的滤波器，分别对输入图像进行特征提取，这样不同滤波器在卷积图像时会得到不同特征的放映，其被称为 Fearture Map（特征映射图）。因此，100 种滤波器就有 100 个 Fearture Map，这 100 个 Fearture Map 就构成了一层神经元。此时，这一层就有10 000个参数。

因此在卷积网络的卷积层中，可以通过下式计算网络参数数量：

$$M = (L \times M \times C + 1) \times N \tag{2-18}$$

在式（2-18）中，M 表示参数个数，L 表示卷积核的长，W 表示卷积核的宽，C 表示图像通道数，1 表示偏差部分 b，N 表示滤波器个数。

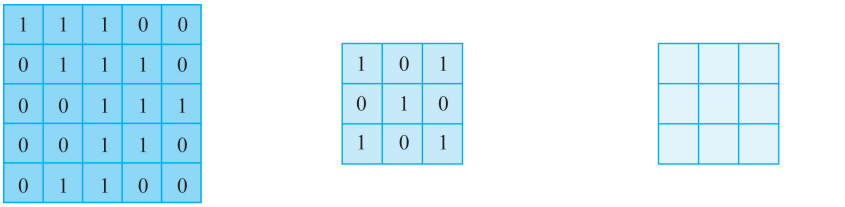
局部感知和权值共享有一个共同的作用，即在不影响结果的情况下，极大地减少了 CNN 模型参数的数量，加快了计算速度。

2.2.3　卷积层

卷积层是 CNN 的核心层。卷积层的功能是对神经网络中的每一小块进行进一步的分析，以达到提取出更高层次特征的目的。滤波器是卷积层中的重中之重。滤波器的功能就是把当前神经网络上的一个子节点矩阵转化为下一层神经网络上的长宽为 1 的单位节点矩阵。

在一个卷积层中，滤波器的长和宽都是手动指定好的，常见的有 3×3 或 5×5。当图像是多通道的时候，滤波器在每个通道上执行二维卷积，最后把每个通道上（i，j）处的卷积结果求和。因此，还需要去指定的是处理图像的通道数，这一设置被称为滤波器的深度。

假定现在有一个 5×5 的图像，使用尺寸为 3×3、偏差为 0 的滤波器对其进行卷积，会得到一个 3×3 的 Feature Map，如图 2-6 所示。



（a）5×5图像　　（b）3×3、偏差为0的滤波器　　（c）生成3×3的特征映射图

图 2-6　卷积操作示意图（1）

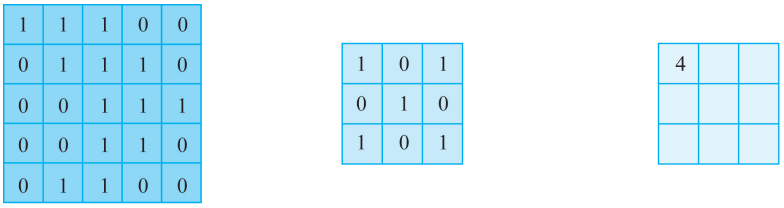
为了详细地叙述卷积操作流程，用 $x_{i,j}$ 表示图像第 i 行、第 j 列的像素点；对滤波器中的权值进行编号，用 $w_{m,n}$ 表示第 m 行、第 n 列的权值，W_b 即为滤波器的偏差值；对 Feature Map 中的元素也进行编号，用 $a_{i,j}$ 表示 Feature Map 第 i 行、第 j 列的元素；用 f 表示 ReLU 激活函数。然后，使用（2-19）所示的公式计算卷积。

$$a_{i,j}=f\left(\sum_{m=0}^{2}\sum_{n=0}^{2}w_{m,n}\times x_{i+m,j+n}+w_b\right) \tag{2-19}$$

例如，对 Feture Map 第 0 行、第 0 列的元素来说，其卷积计算方法为

$$\begin{aligned}a_{0,0}&=f\left(\sum_{m=0}^{2}\sum_{n=0}^{2}w_{m,n}\times x_{m+0,n+0}+w_b\right)\\&=relu(w_{0,0}\times x_{0,0}+w_{0,1}\times x_{0,1}+w_{0,2}\times x_{0,2}+w_{1,0}\times x_{1,0}+w_{1,1}\times x_{1,1}+w_{1,2}\times\\&\quad x_{1,2}+w_{2,0}\times x_{2,0}+w_{2,1}\times x_{2,1}+w_{2,2}\times x_{2,2}+w_b)\\&=relu(4)\\&=4\end{aligned} \tag{2-20}$$

计算过程如图 2-7 所示。

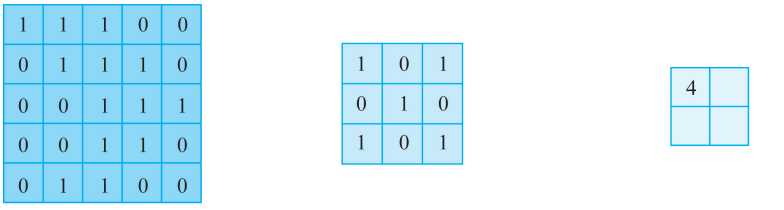


（a）5×5图像　　（b）3×3、偏差为0的滤波器　　（c）生成3×3的特征映射图

图 2-7　卷积操作示意图（2）

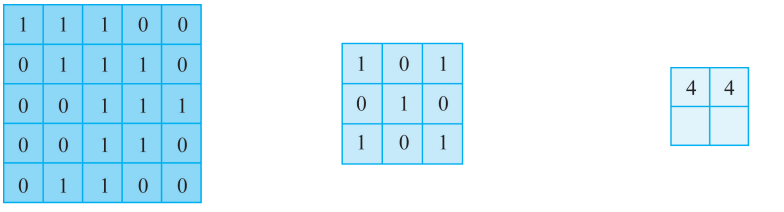
依次计算出 Feature Map 中所有的元素的值（先右移，移到边界处，然后换下一行）。

在上述计算过程中，步长（stride）为 1。当步长为 2 时，Feature Map 的计算如图 2-8、图 2-9 所示。



（a）5×5图像　　（b）3×3、偏差为0的滤波器　　（c）生成2×2的特征映射图

图 2-8　卷积操作示意图（3）



（a）5×5图像　　（b）3×3、偏差为0的滤波器　　（c）生成2×2的特征映射图

图 2-9　卷积操作示意图（4）

当步长设置为 2 的时候，Feature Map 的尺寸就变成了 2×2。步长决定卷积后的 Feature Map 的尺寸，即

$$W_2 = (W_1 - F + 2P)/S + 1 \tag{2-21}$$

$$H_2 = (H_1 - F + 2P)/S + 1 \tag{2-22}$$

式（2-21）中，W_2 是卷积后 Feature Map 的宽度；W_1 是卷积前图像的宽度；F 是滤波器的宽度；P 是零填充（Zero Padding）的数量，是指在原始图像周围补几圈 0，如果 P 的值是 1，那么就补 1 圈 0；S 是步长。式（2-22）中，H_2 是卷积后 Feature Map 的高度，H_1 是卷积前图像的高度。上面两式在本质上是一样的。

比如上面的卷积，其图像宽度 $W_1=5$，滤波器宽度 $F=3$，零填充 $P=0$，步长 $S=2$，则卷积后特征映射图的宽度为

$$\begin{aligned} W_2 &= (W_1 - F + 2P)/S + 1 \\ &= (5 - 3 + 0)/2 + 1 \\ &= 2 \end{aligned} \tag{2-23}$$

从结果可知，Feature Map 宽度是 2。同样，经过计算，Feature Map 的高度也是 2。

如果图像的原始通道数 D 大于 1，此时滤波器的深度也设置为 D，可以得到下式：

$$a_{i,j}=f\left(\sum_{d=0}^{D-1}\sum_{m=0}^{F-1}\sum_{n=0}^{F-1}w_{d,m,n}x_{d,i+m,j+n}+w_b\right) \tag{2-24}$$

式（2-24）中，D 是深度，F 是滤波器的大小，$w_{d,m,n}$表示滤波器的第 d 层、第 m 行、第 n 列，$a_{i,j}$表示 Feature Map 第 i 行、第 j 列的元素。零填充和步长同式（2-23）一样。

为了全面地得到特征信息，需要在每个卷积层中加入多个滤波器。不同的滤波器和图像卷积操作后得到的 Feature Map 也是不同的。因此，卷积后的 Feature Map 的深度（个数）等于卷积层滤波器的数量。

2.2.4　池化层

在经过卷积层后，研究人员可获得图像的特征信息。按理说，此时可以通过提取到的特征去训练分类器，但是，研究人员还需要对庞大的计算量再去做一些处理。例如：在卷积层设置 400 个尺寸为 8×8 的滤波器对一个 96×96 尺寸的图像进行卷积，卷积后得到 400 个 Feature Map。滤波器对每一个特征和图像进行卷积都会得到一个（96-8+1）×（96-8+1）= 7921 维的卷积特征，由于有 400 个特征，所以每个样例都会得到一个 7921×400 = 3168400 维的卷积特征向量。数据量过大，不但会耗费时间，并且很有可能出现过拟合的现象，这是研究人员不希望看到的。

因此，研究人员需要一种操作对经过卷积后的 Feature Map 进行降维操作，这就需要用到 CNN 中的另一个核心部分——池化层。

池化层采取的实际上是一种向下采样的操作。它的操作是把输入区域划分成若干个矩形子区域，对每一个子区域都会输出一个采样值，再用采样值来表示这个子区域。这种机制能够有效的原因在于，在发现一个特征之后，它的精确位置远不及它和其他特征的相对位置关系重要。

在池化层中，有类似于滤波器的“池化核”来对 Feature Map 矩阵进行扫描。“池化核”对矩阵一般有两种操作方法：

（1）Max pooling：取“池化核”矩阵中的最大值。

（2）Average pooling：取“池化核”矩阵中的平均值。

池化层的扫描方式同卷积层一样，采用窗口滑动的方式，先从左到右扫描，结束后则换行继续从左到右扫描。其中，步长决定移动的距离，如图 2-10 所示。

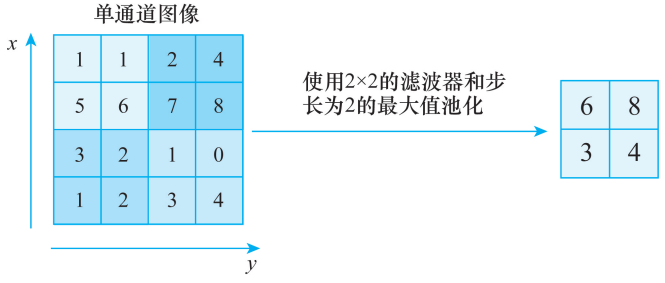


图 2-10　最大值池化

池化操作不同于卷积操作，它会保留 Feature Map 的深度。另外，如果池化层输入的特征尺寸不是 2 的倍数，其会采取边缘补零的方法将特征尺寸补成 2 的整数倍，再进行池化操作。

由于池化层可以有效地减少数据量，所以在 CNN 模型中，卷积层和池化层交替堆叠。

2.2.5 全连接层

全连接层（Fully Connected Layers，FC）的作用是在 CNN 的最后把特征信息进行拟合，获取图像特征的完整信息，之后用于训练分类器。在 CNN 网络中，全连接层将卷积层产生的特征图映射成一个固定长度（一般为输入图像数据集中的图像类别数）的特征向量。这个特征向量包含了输入图像所有特征的组合信息。虽然丢失了图像的位置信息，但是该向量将图像中含有的最具有特点的图像特征保留了下来。

2.3 卷积神经网络学习原理

一个 CNN 预测的是否准确，是由参数 w 和 b 决定的。所以通过大量的训练数据来训练 CNN 模型，其目的就是要找到合适的参数 w 和 b。

反向传播算法是一种有效的训练模型参数的算法，其主要思想为：由于模型的预测结果和训练样本预期结果之间存在误差，因此要把得到的误差通过反向传播传播到输入层；在反向传播的过程中，模型会根据误差的变化去更新网络参数，重复迭代，直到模型收敛为止。

梯度下降算法（gradient descent）是一种常用的反向传播算法。梯度下降算法会根据反向传播的误差变化更新 w 和 b 的值。更新后的 w 和 b 会使损失函数输出减小，即在多次迭代中，逐渐提高模型预测准确率。梯度下降算法如下式所示：

$$\hat{y} = \sigma(w^T x + b) \tag{2-25}$$

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \tag{2-26}$$

式（2-25）是用于预测的算法，即逻辑回归算法，式（2-26）是用于判断预测是否准确的损失函数。当用带标签（即已知输入 x 和输出 y）的训练数据去训练模型的时候，根据损失函数的大小可以判断预测结果是否精确。当损失函数最小的时候，此时的 w 和 b 是最合适的值。

如图 2-11 所示，损失函数的形状是一个漏斗状。训练的目的就是找到在漏斗最底部的一组 w 和 b。这种漏斗状的函数被称为凸函数（向下凸起的函数）。选择 J 为损失函数的原因正是因为 J 是一个凸函数，比起一些不是凸函数的损失函数来说，其更容易找到函数的最小值。

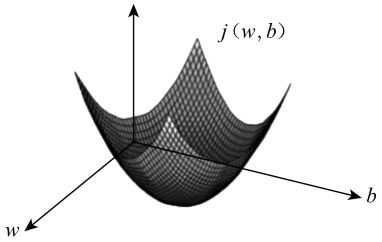


图 2-11　损失函数

如图 2-12 所示，梯度下降算法会改变 w 和 b 的值，以达到减小损失函数的目的，其最终可以找到一组使损失函数输出最小的 w 和 b。

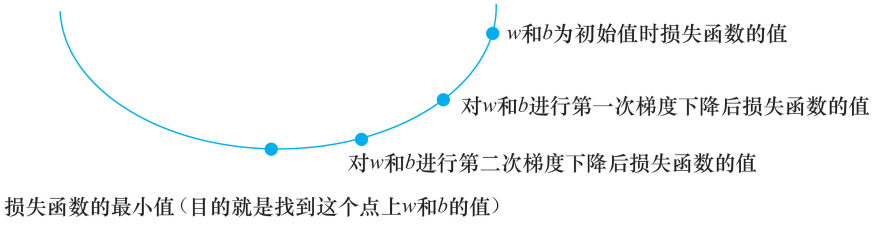


图 2-12　梯度下降原理图

为了理解网络是如何去更新参数的，我们假设损失函数 J 只有一个参数 w（实际上 J 是一个关于 w 和 b 的函数），并且假设 w 只是一个实数（实际上 w 是一个向量或一组实数）。如图 2-13 所示，梯度下降算法一步一步地在改变着 w 的值，在使损失函数的结果越来越小（将 w 的值一步一步地移到最小值处）。梯度下降算法通过式（2-27）来改变 w 的值。

$$w' = w - r \times d_w \tag{2-27}$$

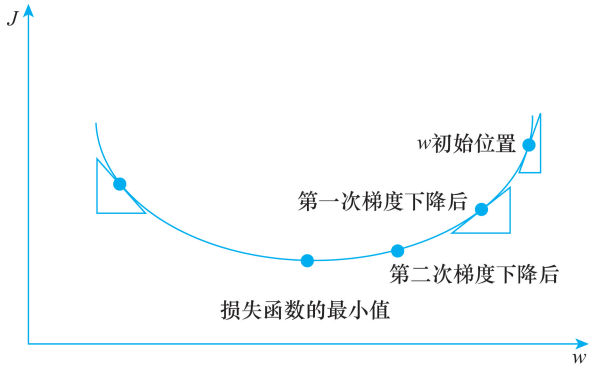


图 2-13　梯度下降过程

梯度下降算法在反向传播的过程中利用上面的公式来修改 w 的值。r 表示学习率（learning rate），是用来控制 w 的变化率的参数。d_w 是参数 w 关于损失函数 J 的偏导数，相当于斜率。新的 w 的值（w'）等于旧的 w 减去学习率 r 与偏导数 d_w 的乘积。

在这里需要注意，选择一个合适的学习率很重要，因为如果每次让 w 改变太多，有可能直接跳过 J 的最小值处。假设学习率设置得过大，在第一次迭代后，w 跳过了 J 的最小值，之后的每次迭代都会使 w 距离最底部越来越远，永远到不了最底部，即模型预测的结果无法收敛。而如果学习率设置的太小，又会影响到学习速度。

为了解决上面如何选择一个正确的学习率的问题，相关学者提出了批量标准化（Batch Normalization，BN）的概念。批量标准化可以在每一个训练中对每一维进行修正，即在每一维减去自身均值，再除以自身标准差。CNN 中的批量标准化操作通常作用于非线性激活函数之前。

批量标准化的好处在于它可以使用一个更高的学习率来训练，即提高了学习速度，也避免了上述问题。

2.4 卷积神经网络模型

在 CNN 的发展历程中，出现过很多经典的网络模型。例如：1986 年的 LeNet-5，2012 年的 AlexNet，2014 年的 GoogLeNet，2014 的年 VGGNet。

2.4.1 LeNet-5

LeNet-5 是一个典型的 CNN 网络模型，该模型是为了识别手写字体和计算机打印字符而设计的，而且该模型确实在手写体识别领域非常成功，曾被广泛应用于美国银行支票手写体识别。LeNet-5 的网络结构如图 2-14 所示。

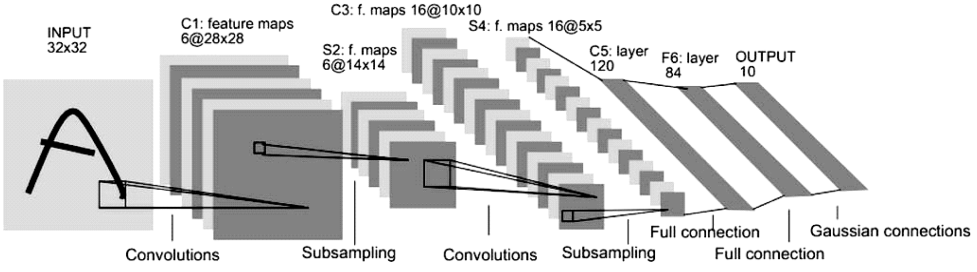


图 2-14 LeNet-5 网络模型结构

LeNet-5 共有 7 层，不包含输入，每层都包含可训练参数；每层有多个 Feature Map，每个 Feature Map 通过一种卷积滤波器提取输入的一种特征，并且每个 Feature Map 有多个神经元。

2.4.2 AlexNet

AlexNet 由 7 层隐藏层构成。其中 1～5 层是卷积层，6～7 是全连接层。图 2-15 即为 AlexNet 的 CNN 结构图。需要注意的是，该模型采用了 2-GPU 并行结构，即第 1、2、4、5 卷积层都是将模型参数分为两部分进行训练的。并行结构分为数据并行与模型并行。数

据并行是指在不同的 GPU 上，模型结构相同，但将训练数据进行切分，对数据分别进行训练得到不同的模型，然后再将模型进行融合。而模型并行则是将若干层的模型参数进行切分，在不同的 GPU 上使用相同的数据进行训练，将得到的结果直接连接作为下一层的输入。AlexNet 网络模型结构如图 2-15 所示。

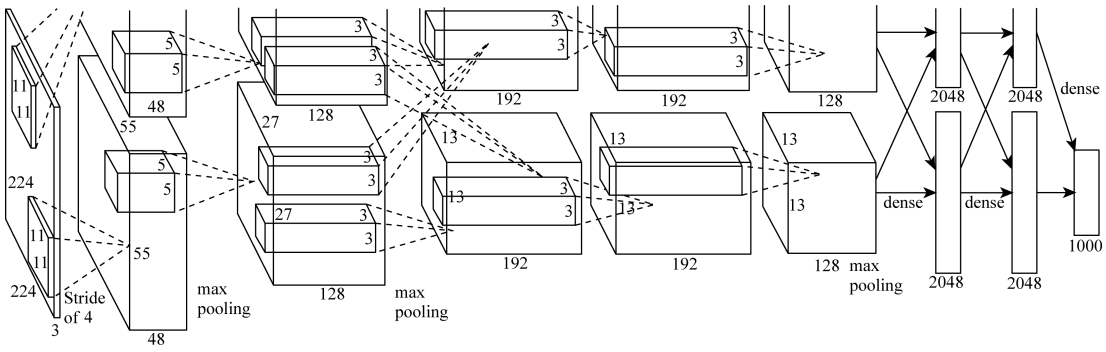


图 2-15　AlexNet 网络模型结构

上图模型的基本参数如下。

输入：224×224 大小的图片，3 通道。

第一层卷积：5×5 的卷积核 96 个，每个 GPU 上 48 个。

第一层 max pooling：2×2 的核。

第二层卷积：3×3 的卷积核 256 个，每个 GPU 上 128 个。

第二层 max pooling：2×2 的核。

第三层卷积：与上一层是全连接；3×3 的卷积核 384 个，每个 GPU 上个 192 个。

第四层卷积：3×3 的卷积核 384 个，每个 GPU 上 192 个；该层与上一层的连接没有经过 pooling 层。

第五层卷积：3×3 的卷积核 256 个，每个 GPU 上 128 个。

第五层 max pooling：2×2 的核。

第一层全连接：4096 维，将第五层 max pooling 的输出连接成为一个一维向量，作为该层的输入。

第二层全连接：4096 维 Softmax 层，输出为 1000，输出的每一维都是图片属于该类别的概率。

2.4.3　VGGNet

VGG 最大的贡献是成功证明了小卷积核+小池化层+深度网络的潜力。通过反复堆叠 3×3 的小型卷积核和 2×2 的池化层，VGG 将卷积神经网络的深度拓展到了 19 层。更重要的是，VGG 的拓展性能非常好，其迁移到其他图片数据上的泛化性非常好。

表 2-1 展示了 VGGNet 的网络结构。作者从 A 方案 11 层卷积神经网络开始，逐渐增加卷积的层数到 E 方案的 19 层。

从表中可以看出，VGGNet 有 5 段卷积，每段中都有 1～3 个 3×3 的小卷积核，每段卷

积核的数量一样，段数越大，卷积核数量越多。后期层数增加后，会有两个小卷积核堆叠在一起的情况，也就是多个小感受野堆叠后实际感受野扩大。例如 2 层的小卷积核叠在一起，相当于 1 个 5×5 的卷积核，但是卷积核的参数比较小。

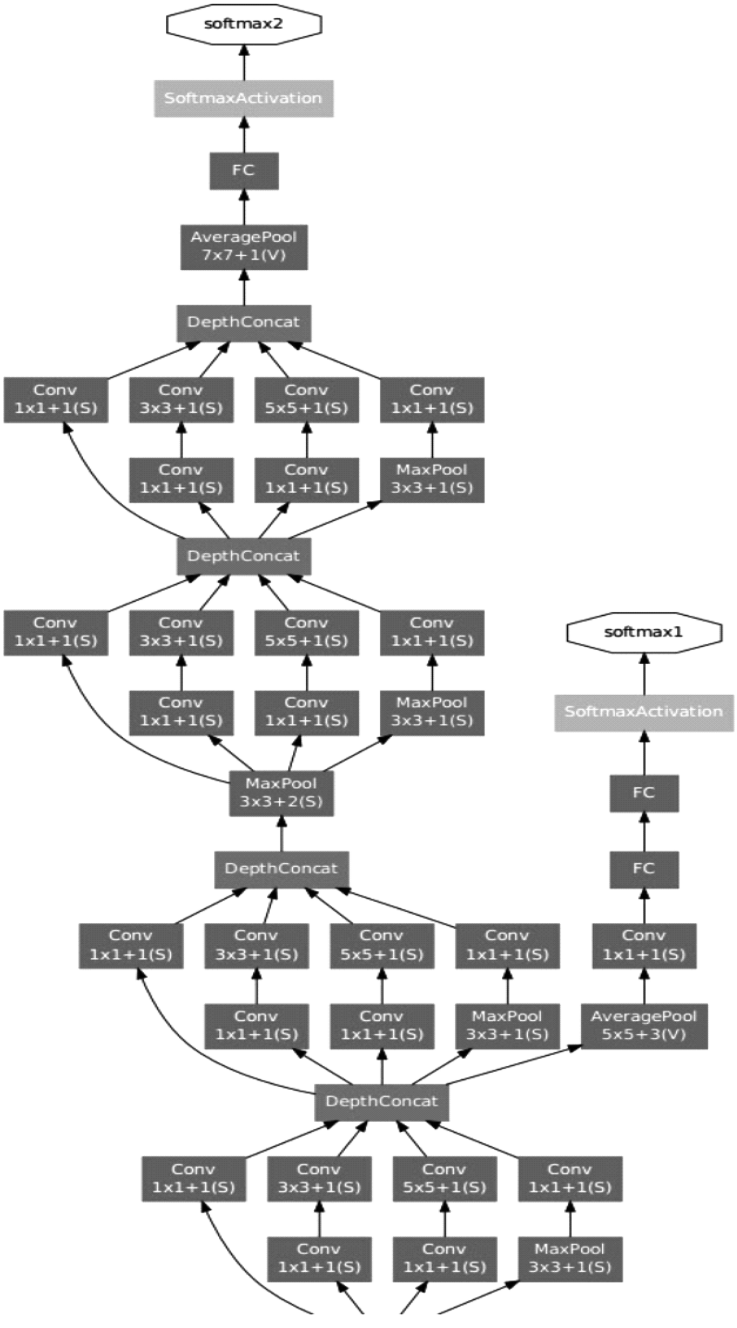
表 2-1　VGG 卷积网络配置表

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input（224×224 RGB image）					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
max pool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
max pool					
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
max pool					
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
max pool					
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
max pool					
FC-4096					
FC-4096					
FC-1000					
Softmax					

2.4.4　GoogLeNet

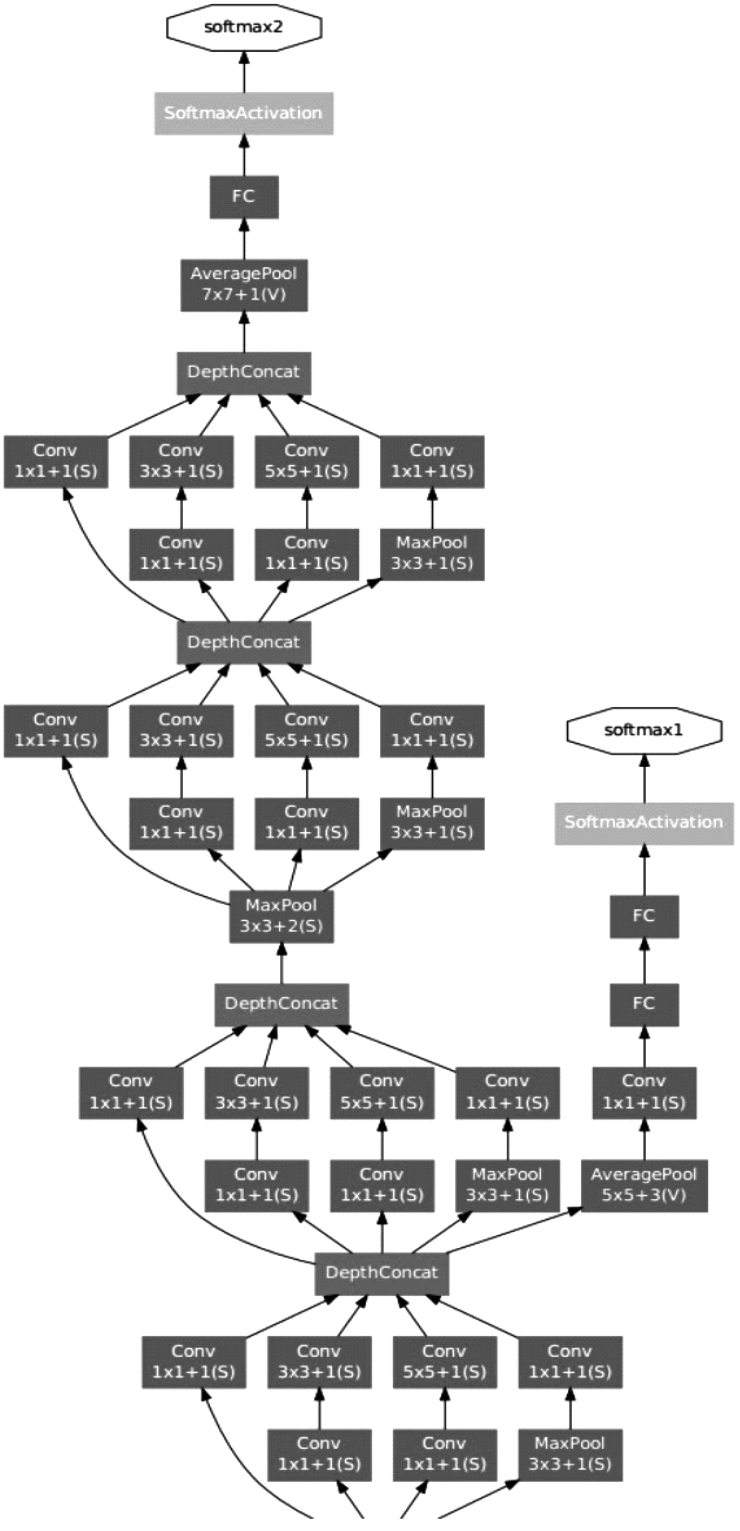
GoogLeNet，是 14 年由 Google 团队提出的模型，这个模型证明了一件事情：用更多的

卷积、更深的层次可以得到更好的结构。图 2-16 为 GoogLeNet 的网络结构图，分为 a、b 两部分。



（a）

图 2-16　GoogLeNet 网络模型结构



（b）

图 2-16　GoogLeNet 网络模型结构（续）

2.5　循环神经网络原理

2.5.1　循环神经网络概述

循环神经网络（Recurrent Neural Network，RNN）也被称作时间递归神经网络，是深度学习的基本算法之一，在语音识别、语音互动、图像分析、视频检测、场景预测等众多领域都有广泛的应用和非常广阔的前景。RNN 有三个基本层：输入层、隐藏层和输出层。不同于其他算法，RNN 最大的特点在于它是一种具有特殊反馈结构的神经网络。简言之，其输出层当前状态的输出值不仅取决于当前状态的输入值和隐藏层的权重，还受它上一状态的输入值的影响，这种反馈结构让它有了类似人脑联系前后文的“思维”能力。

如图 2-17 所示为循环神经网络简图，从下至上依次为输入层（x）、隐藏层（s）和输出层（o）。其中，t 表示时间，U 是由输入层反馈到隐藏层的权重矩阵，V 是由隐藏层反馈到输出层的权重矩阵，W 是隐藏层（隐藏层是 N 个节点相连的循环层）之间由上一层反馈到下一层的权重矩阵。当我们把时间序列展开，不难看出在循环神经网络中不同时刻输入隐藏层的信息是不一样的，而且当前层反馈到下一层的反馈权重信息也会受到当前层输入信息和从上一层收到的反馈信息的影响。人脑在处理信息时会主动提取关键信息，例如：今早下了很大的雨，小华出门时忘了带伞，请确定小华到学校的时候衣服的状态。在读取信息时，人脑会主动把修饰词“大”和“今早”等无用信息剔除，从而组合有用信息“下雨出门忘带伞”，并得出衣服“被淋湿”的结论。循环神经网络就是通过模仿人脑的工作方式来实现对信息的处理与分析。

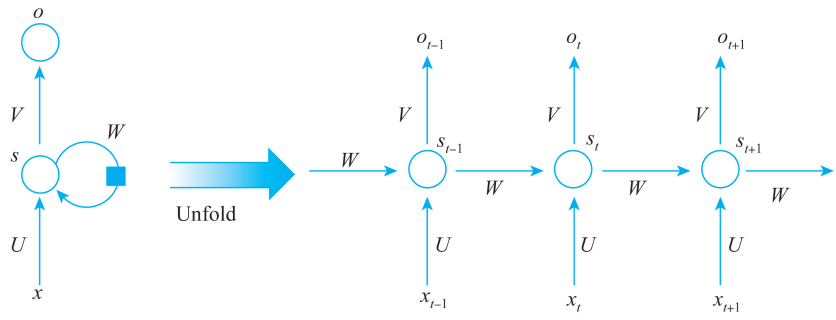


图 2-17　循环神经网络原理

2.5.2　RNN 的激活函数

循环神经网络包含的大量数据会加大计算量，使得整个神经网络运算变得烦琐，并会使得整个系统运行时间变长。隐藏层是一个循环层，大量数据之间的循环运算会直接导致过饱和的情况出现。很多混乱无用的信息会直接使得输出层结果出现很大偏差。上述这些

问题需要激活函数来进行非线性变换进而保证输出结果的非线性，进而得到当前层的输出权重并作用到下一隐藏层。激活函数是神经网络最基本的函数之一。如果不加入激活函数，那么整个神经网络连接即使再复杂，其实际上也只能通过一系列线性运算得到结果。常见的激活函数有 Sigmoid、tanh 、ReLU 等。

Sigmoid 函数也被称为 S 型函数，它可以将整个实数区间映射到（0，1）区间，因此经常被用来计算概率。它也是在传统神经网络中被经常使用的一种激活函数。在使用Sigmoid函数时，当 $x>4$ 和 $x<-4$ 的时候，导数函数的取值趋近于 0，这使得参数 W 即使在迭代更改时也几乎不变，会导致梯度消失。

tanh 函数是双曲正切函数，它将整个实数区间映射到了（-1，1），它的输出是以 0 为中心。tanh 的收敛速度比 Sigmoid 要快，但是由于存在软饱和性，tanh 也存在梯度消失的问题。

ReLU 函数现在是最受欢迎的激活函数，经常被使用在 RNN 神经网络中。ReLU 函数在$x<0$时，输出始终为 0。在 $x>0$ 时，ReLU 函数的导数为 1，所以 ReLU 函数能够在 $x>0$ 时保持梯度不断衰减，从而缓解梯度消失的问题，并且能加快收敛速度，还能使神经网络具有稀疏性表达能力，这也是 ReLU 激活函数能够被使用在深层神经网络中的原因。

在深度学习中，为了防止出现过饱和，损失函数（loss funtion），也称代价函数（cost funtion），需要对输出层的输出值与预想值做反复对比，从而避免输出值与预想值出现过大偏差，减少运算次数。常见的损失函数有：最小二乘法损失函数、smooth L1 损失函数、交叉熵损失函数。为了得到最优结果，研究人员还需要通过优化函数（optimization function）将神经网络的损失值向前传递，从而对网络整体进行优化，使得所得结果更加准确。

2.6　双向 RNN 和深层双向 RNN

循环神经网络在处理当前状态的输入信息时需要结合上一状态输入的权重去计算，但其在处理实际问题时往往会遇到复杂的场景。如果人脑并未接收到完整信息，人有时候需要根据部分已知信息去判断之前所发生的事情。例如下题：今早下了很大的____，小华出门时忘了带伞，到学校的时候衣服已经被淋湿了。显然，人脑在处理这段信息时需要根据后文提取出“衣服被淋湿”和“出门忘了带伞”从而得出结果：下的是“雨”。人脑是通过后面状态所收到的信息向前反馈，但 RNN 简单地从上一状态到下一状态的反馈并不能实现结合上下文预测的功能，因此在行为分析中，如果需要处理这类问题就需要使用双向 RNN。双向 RNN 的原理图如图 2-18 所示。

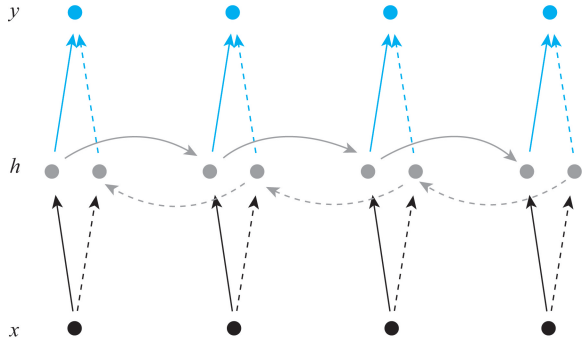


图 2-18　双向 RNN 原理

由图 2-18 可以看出，双向 RNN 是在原有基础上添加了从后向前的权重矩阵，它在进行数据处理时，就需要结合之前和之后两个隐藏层所输出的权重信息。在实际应用中，RNN 不仅像人脑一样拥有了“记忆”功能，还拥有简单的“思维”能力。由于加入了由后向前的反馈势必会导致权重信息的增加，所以双向 RNN 的运行需要更大的内存。也正是由于这种双向反馈的结构，双向 RNN 在语音识别和预测中表现出十分强大的能力。

在实际应用中，双向 RNN 的应用场景往往是非常复杂的。就像学生在背课文时往往并不能看一遍课文就能完全将其记住，而是需要反复诵读才能达到预期效果一样，深层双向 RNN 就是基于这个思路被设计出来的。深层双向 RNN 的原理如图 2-19 所示。

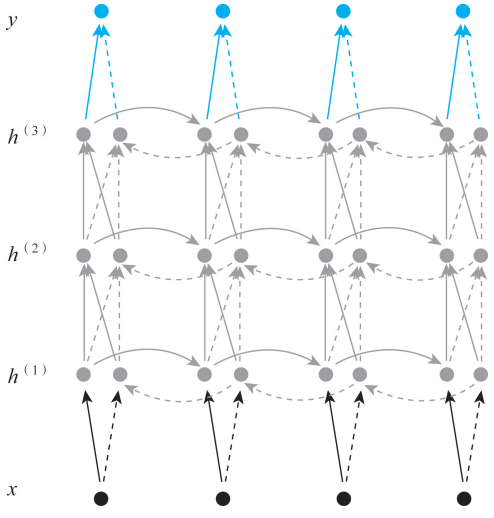


图 2-19　深层双向 RNN 原理

深层双向 RNN 在双向 RNN 的基础上添加了多个隐藏层。深层双向 RNN 在处理数据时，通过多个隐藏层之间的相互影响从而拥有了像人在学习时“逐步熟悉”的能力，简言之就像人脑对复杂信息的“学习”能力，适用于处理场景比较复杂的数据。

练习题

1. 神经元模型的结构包括哪些要素？
2. 深度神经网络由哪些层构成？
3. 反向传播算法的原理是什么？
4. 简述梯度下降法的原理，以及学习率参数的作用是什么？
5. LeNet-5 网络包括哪些结构？
6. AlexNet 网络包括哪些结构？
7. VGGNet 网络包括哪些结构？
8. GoogLeNet 网络有哪些特点？
9. 循环神经网络有哪些特点？
10. 较常规循环神经网络而言，双向循环神经网络有哪些改进之处？