Deep Learning Technology and Application

Ge Li

Peking University

Table of contents

卷积函数

卷积函数，是关于两个函数的函数设存在原始函数为：$f(x)$；
设存在对原始函数的输出进行权重修正的函数：$w(t-x)$；则：

$$s(t) = \int_0^t f(x)w(t-x)dx$$

该卷积操作通常表示为：

$$s(t) = (f * w)(t)$$

其中：

- $w(t-x)$ 表示对"$f(x)$ 在 x 点的值"进行加权的权值；
- 它是 t 的函数，t 是与 x 同一维度的自变量，$t-x$ 表示当前 t 点对 x 点的距离；
- 也就是说，$w(t-x)$ 是一个随"t 点对 x 点的距离"而变化的函数；

卷积运算

一个来自知乎的例子：每隔一年存入 100 元，年利率 5%：

本金	第一年	第二年	第三年	第四年	第五年
+100	$100\times(1.05)^1$	$100\times(1.05)^2$	$100\times(1.05)^3$	$100\times(1.05)^4$	$100\times(1.05)^5$
	+100	$100\times(1.05)^1$	$100\times(1.05)^2$	$100\times(1.05)^3$	$100\times(1.05)^4$
		+100	$100\times(1.05)^1$	$100\times(1.05)^2$	$100\times(1.05)^3$
			+100	$100\times(1.05)^1$	$100\times(1.05)^2$
				+100	$100\times(1.05)^1$
					+100

设存钱函数为：$f(\tau)(0 \le \tau \le t)$

设复利计算公式为：$g(t-\tau) = (1+5\%)^{(t-\tau)}$

则最终得到的钱数，即卷积值为：

$$\int_0^t f(\tau)g(t-\tau)d\tau = \int_0^t f(\tau)(1+5\%)^{t-\tau}d\tau$$

卷积函数

在离散的情况下，可以把卷积函数写成：

$$s(t) = (f * w)(t) = \sum_{x=-\infty}^{\infty} f(x)w(t-x)$$

在 x 为多维数据的情况下，如对于二维数据（灰度图像），可以将上式重写为：

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m,j-n)$$

其中：

- 函数 I 表示输入，$I(m,n)$ 为图像在 (m,n) 点的灰度值;
- $K(i-m,j-n)$ 表示对"图像在 (m,n) 点的灰度值 $I(m,n)$"的加权值；
- $i-m$（或 $j-n$）表示 i 点到 m 点（或 j 点到 n 点）的距离；
- 加权值 $K(i-m,j-n)$ 是"i 点到 m 点（或 j 点到 n 点）的距离"的函数；

卷积函数

对公式：

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

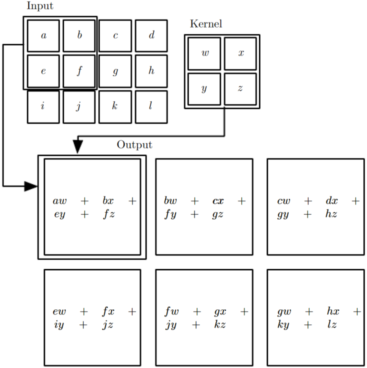
进行近似等价重写，得到 Cross-Correlation（互相关）公式：

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

这就是我们常见的"卷积"公式！其中：

- $I(i+m, j+n)$ 表示以 (i,j) 为起点，以 (m,n) 为宽度和高度的输入区域的灰度值矩阵；
- $K(m,n)$ 表示宽度为 m，高度为 n 的卷积核（$m \times n$ 的矩阵）;
- $S(i,j)$ 表示"以 (i,j) 为起点，宽度为 m，高度为 n 的灰度值矩阵"经过卷积核 K 进行卷积计算的值；
- S 的大小由 (i,j) 的最大值确定，也可以说，由输入区域 I 和卷积核 K 共同决定；

卷积运算



$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

$$S(0,0) = (I * K)(0,0) = \sum_m \sum_n I(0+m, 0+n)K(m,n)$$

$$S(1,0) = (I * K)(1,0) = \sum_m \sum_n I(1+m, 0+n)K(m,n)$$

$$S(2,0) = (I * K)(2,0) = \sum_m \sum_n I(2+m, 0+n)K(m,n)$$

$$S(0,1) = (I * K)(0,1) = \sum_m \sum_n I(0+m, 1+n)K(m,n)$$

$$S(1,1) = (I * K)(1,1) = \sum_m \sum_n I(1+m, 1+n)K(m,n)$$

$$S(2,1) = (I * K)(2,1) = \sum_m \sum_n I(2+m, 1+n)K(m,n)$$

卷积运算

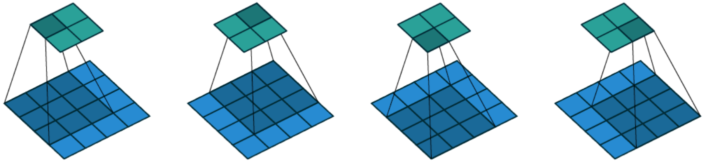


Figure 2.1: (No padding, no strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

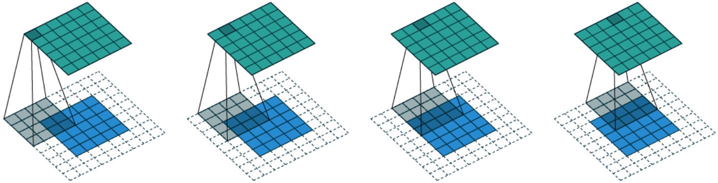


Figure 2.2: (Arbitrary padding, no strides) Convolving a 4×4 kernel over a 5×5 input padded with a 2×2 border of zeros using unit strides (i.e., $i = 5$, $k = 4$, $s = 1$ and $p = 2$).

卷积运算

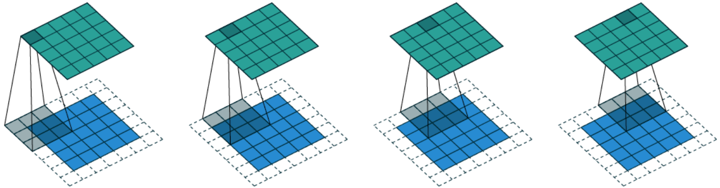


Figure 2.3: (Half padding, no strides) Convolving a 3×3 kernel over a 5×5 input using half padding and unit strides (i.e., $i = 5$, $k = 3$, $s = 1$ and $p = 1$).

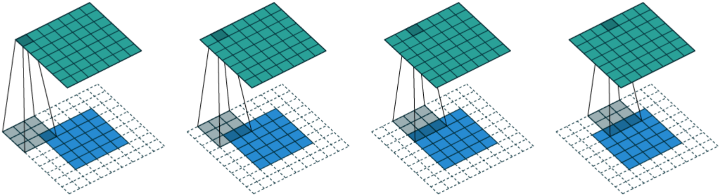


Figure 2.4: (Full padding, no strides) Convolving a 3×3 kernel over a 5×5 input using full padding and unit strides (i.e., $i = 5$, $k = 3$, $s = 1$ and $p = 2$).

卷积运算的物理含义

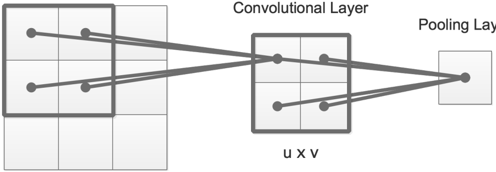
$$K_{horizontal\_high\_magnitude} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(a) Lenna

(b) Horizontal edge

(c) Vertical edge

卷积层的前向计算



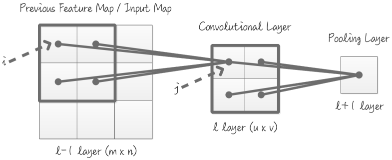
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

可以写成：

$$z^{(l)}_{(u,v)} = \sum_{(m,n)\in M_{(u,v)}} a^{(l-1)}_{(m,n)} * k^{(l)}_{(u,v)(m,n)} + b^{(l)}_{(u,v)}$$

$$a^{(l)}_{(u,v)} = f(z^{(l)}_{(u,v)})$$

卷积层的前向计算



$$z_{(u,v)}^{(l)} = \sum_{(m,n)\in M_{(u,v)}} a_{(m,n)}^{(l-1)} * k_{(u,v)(m,n)}^{(l)} + b_{(u,v)}^{(l)}$$

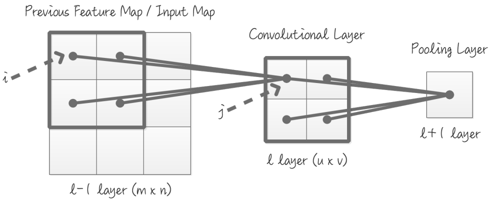
$$a_{(u,v)}^{(l)} = f(z_{(u,v)}^{(l)})$$

对上式进行简化：

$$z_j^{(l)} = \sum_{i\in M_j} a_i^{(l-1)} * k_{ji}^{(l)} + b_j^{(l)} \qquad a_j^{(l)} = f(z_j^{(l)})$$

上式中 $i \in M_j$ 表示：i 在与卷积层节点 j 所对应的输入窗口 M_j 中；

Pooling 层的前向计算



$$z_k^{(l+1)} = \beta_k^{(l+1)} down_{j \in M_k}(a_j^{(l)}) + b_k^{(l+1)} \qquad a_k^{(l+1)} = f_{pooling}(z_k^{(l+1)})$$

其中：

- $down(\cdot)$ 为下采样函数，$j \in M_k$ 表示：j 在与 Pooling 层节点 k 所对应的卷积层的窗口 M_k 中；
- 常见的下采样函数如：取平均（Mean Pooling）或取最大值（Max Pooling）；
- 常数参数 $\beta_k^{(l+1)}$ 可以取 1；偏置参数 $b_k^{(l+1)}$ 可以取 0；

卷积网络的权重计算

$$
\begin{aligned}
w_{ji}^{(l)} &= w_{ji}^{(l)} - \alpha \frac{\partial J(W,b)}{\partial w_{ji}^{(l)}} = w_{ji}^{(l)} - \alpha \delta_j^{(l)} a_i^{(l-1)} \\
&= w_{ji}^{(l)} - \alpha \left(\sum_{k=1}^{n_{l+1}} \delta_k^{(l+1)} w_{kj}^{(l+1)} f'(z_j^{(l)}) \right) a_i^{(l-1)}
\end{aligned}
$$

对照上式，对 $k_{ji}^{(l)}$ 进行求解：首先，在不考虑权值共享的前提下：

$$
\begin{aligned}
k_{ji}^{(l)} &= k_{ji}^{(l)} - \alpha \frac{\partial J(W,b)}{\partial k_{ji}^{(l)}} = k_{ji}^{(l)} - \alpha \frac{\partial J(W,b)}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial k_{ji}^{(l)}} \\
&= k_{ji}^{(l)} - \alpha \frac{\partial J(W,b)}{\partial z_j^{(l)}} a_i^{(l-1)} = k_{ji}^{(l)} - \alpha \delta_j^{(l)} a_i^{(l-1)}
\end{aligned}
$$

接下来，关键看 $\delta_j^{(l)}$ 怎么计算；

情况一：当前层之后为 Pooling 层

对照以前的推导方法：因为 $z_k^{(l+1)} = \sum_{j=1}^{n_{l+1}} w_{kj}^{(l+1)} a_j^{(l)} + b^{(l+1)}$ 所以可以选择从 $z_k^{(l+1)}$ 开始进行对 $z_j^{(l)}$ 进行求导。但此处$z_k^{(l+1)}$为何物呢？

情况一：假设，当前层之后为 Pooling 层；

则 $z_k^{(l+1)}$ 为 Pooling 层的激活函数的输入值：

$$\delta_j^{(l)} = \frac{\partial J(W,b)}{\partial z_j^{(l)}} = \frac{\partial J(W,b)}{\partial z_k^{(l+1)}} \frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} = \delta_k^{(l+1)} \frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}} f'(z_j^{(l)})$$

注意，其中：

$$z_k^{(l+1)} = \beta_k^{(l+1)} down_{j \in M_k}(a_j^{(l)}) + b_k^{(l+1)}$$

情况一：当前层之后为 Pooling 层

对上文的公式进行分析：

- $\frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}}$ 表达了在计算 $z_k^{(l+1)}$ 的过程中 $a_j^{(l)}$（其中 j 为：与 Pooling 层节点 k 所对应的卷积层的窗口 M_k 中的元素）的"贡献程度";
- 因此，算式 $\frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}}$ 的计算中， $\beta_k^{(l+1)}$ 可以保留，即：

$$\delta_j^{(l)} = \left(\beta_k^{(l+1)} \delta_k^{(l+1)} f'(z_j^{(l)})\right) \frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}}$$

- 为保证可计算性和计算效率，可以用一个上采样函数 $up(\cdot)$ 来替代 $\frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}}$ 实现：根据"与 Pooling 层节点 k 所对应的卷积层的窗口 M_k 中的不同神经元输出的贡献程度"，对如上等式中的已知部分 $\left(\delta_k^{(l+1)} \beta_k^{(l+1)} f'(z_j^{(l)})\right)$ 进行分配；

情况一：当前层之后为 Pooling 层

上采样函数 $up(\cdot)$ 的选取：

- 若 Pooling 层的下采样函数采用 Mean Pooling，则该上采样函数可以取：

$$up(x) \equiv \frac{x \otimes 1_{n\times n}}{n \times n}$$

  其中，$\otimes$ 为 Kronecker 乘积。
- 若 Pooling 层的下采样函数采用 Max Pooling，则该上采样函数可以通过记录 Max 值的来源位置，来实现；
- 在计算过程中，要保持"分配后的各个 $\delta_j^{(l)}$ 的和"与"Pooling 层已算得的 $\delta_k^{(l+1)}$"相等。

情况一：当前层之后为 Pooling 层

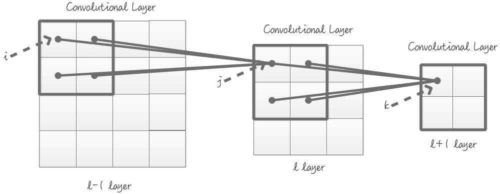
1	3
2	4

1	1	3	3
1	1	3	3
2	2	4	4
2	2	4	4

0.25	0.25	0.75	0.75
0.25	0.25	0.75	0.75
0.5	0.5	1	1
0.5	0.5	1	1

0	0	0	3
0	1	0	0
0	0	0	0
0	2	0	4

情况二：当前层之后为卷积层



情况二：假设当前层之后为卷积层；

则，$z_k^{(l+1)}$ 为下一卷积层激活函数的输入值：

$$\delta_j^{(l)} = \frac{\partial J(W,b)}{\partial z_j^{(l)}} = \sum_K \sum_{k \in C_k \in K} \frac{\partial J(W,b)}{\partial z_k^{(l+1)}} \frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}}$$
$$= \sum_K \sum_{k \in C_k \in K} \delta_k^{(l+1)} k_{kj}^{(l+1)} f'(z_j^{(l)})$$

其中：C_k 为卷积运算中包含 $z_j^{(l)}$ 的 $l+1$ 层中的神经元的集合；K 为卷积核的数量；

卷积网络的权重计算

1	3
2	2

0	0	0	0
0	1	3	0
0	2	2	0
0	0	0	0

0.1	0.2
0.2	0.4

0.1	0.2	0.3	0.6
0.2	0.4	0.6	1.2
0.2	0.4	0.2	0.4
0.4	0.8	0.4	0.8

0.1	0.5	0.6
0.4	1.6	1.6
0.4	1.2	0.8

-0.5	0.4	0.7
0.3	1.9	1.9
0.5	1.5	1.0

2	1
1	1

0	0	0	0
0	2	1	0
0	1	1	0
0	0	0	0

-0.3	0.1
0.1	0.2

-0.6	0.2	-0.3	0.1
0.2	0.4	0.1	0.2
-0.3	0.1	-0.3	0.1
0.1	0.2	0.1	0.2

-0.6	-0.1	0.1
-0.1	0.3	0.3
0.1	0.3	0.2

带有卷积权值的网络

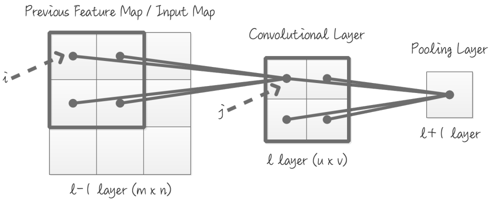
设 α_{ij} 为第 i 个输入特征图与第 j 个卷积核之间的连接强度；

$$a_j^l = f\left(\sum_{i=1}^{N_{in}} \alpha_{ij}(x_i^{l-1} * k_i^l) + b_j^l\right)$$

其中：

- N_{in} 为输入特征图的个数；
- $\sum_i \alpha_{ij} = 1$ 且 $0 \leq \alpha_{ij} \leq 1$；
- 可以设 $\alpha_{ij} = \frac{exp(c_{ij})}{\sum_k exp(c_{kj})}$；
- 接下来的运算，将针对一个确定的输出特征图 j，因此上述公式可以简化为：$\alpha_i = \frac{exp(c_i)}{\sum_k exp(c_k)}$；

带有卷积权值的网络



接下来看一下，推导 α_{ij}，即 α_i 的求解方法：

$$\frac{\partial J}{\partial \alpha_i} = \frac{\partial J}{\partial z^l} \frac{\partial z^l}{\partial \alpha_i} = \sum_{u,v} (\delta^l \circ (a_i^{l-1} * k_i^l))_{uv}$$

继续求解关于 c_i 的导数：

$$\frac{\partial J}{\partial c_i} = \sum_k \frac{\partial J}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial c_i} = \alpha_i \left(\frac{\partial J}{\partial \alpha_i} - \sum_k \frac{\partial J}{\partial \alpha_k} \alpha_k \right)$$

带有卷积权值的网络

为了能够增强 α_i 的效果，我们可以想办法让它变得更稀疏：

$$\hat{J}(W,b) = J(W,b) + \Omega(\alpha) = J(W,b) + \lambda \sum_{i,j} |(\alpha)_{ij}|$$

在这个条件下，α 如何更新呢？

$$\frac{\partial \Omega}{\partial \alpha_i} = \lambda \; sign(\alpha_i)$$

将 α 的求解方式代入：

$$\frac{\partial \Omega}{\partial c_i} = \sum_k \frac{\partial \Omega}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial c_i} = \lambda \left(|\alpha_i| - \alpha_i \sum_k |\alpha_k| \right)$$

因此：

$$\frac{\partial \hat{J}}{\partial c_i} = \frac{\partial J}{\partial c_i} + \frac{\partial \Omega}{\partial c_i}$$

Thanks.