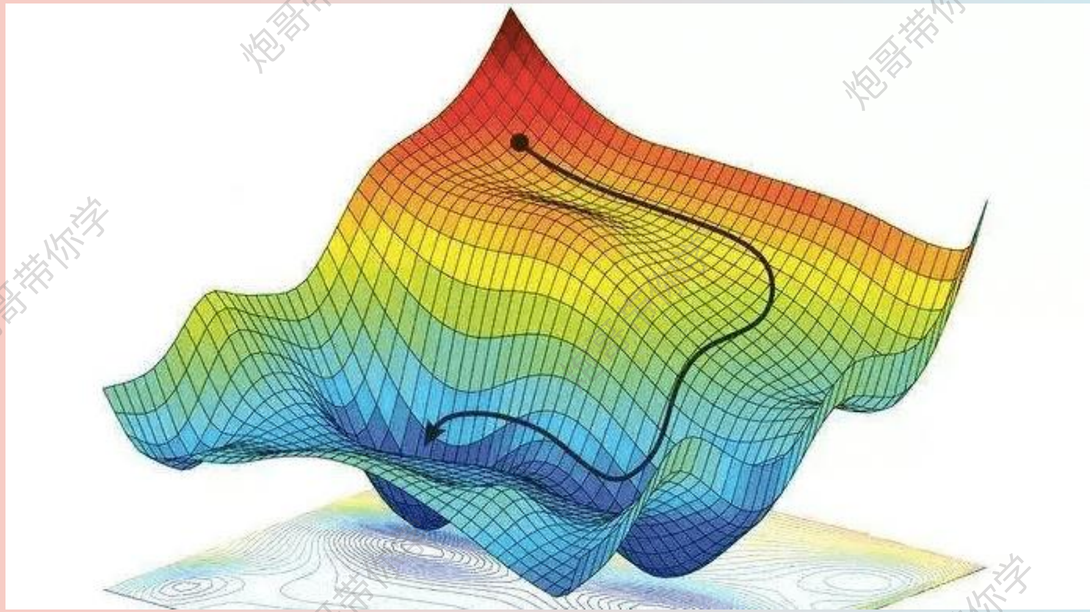
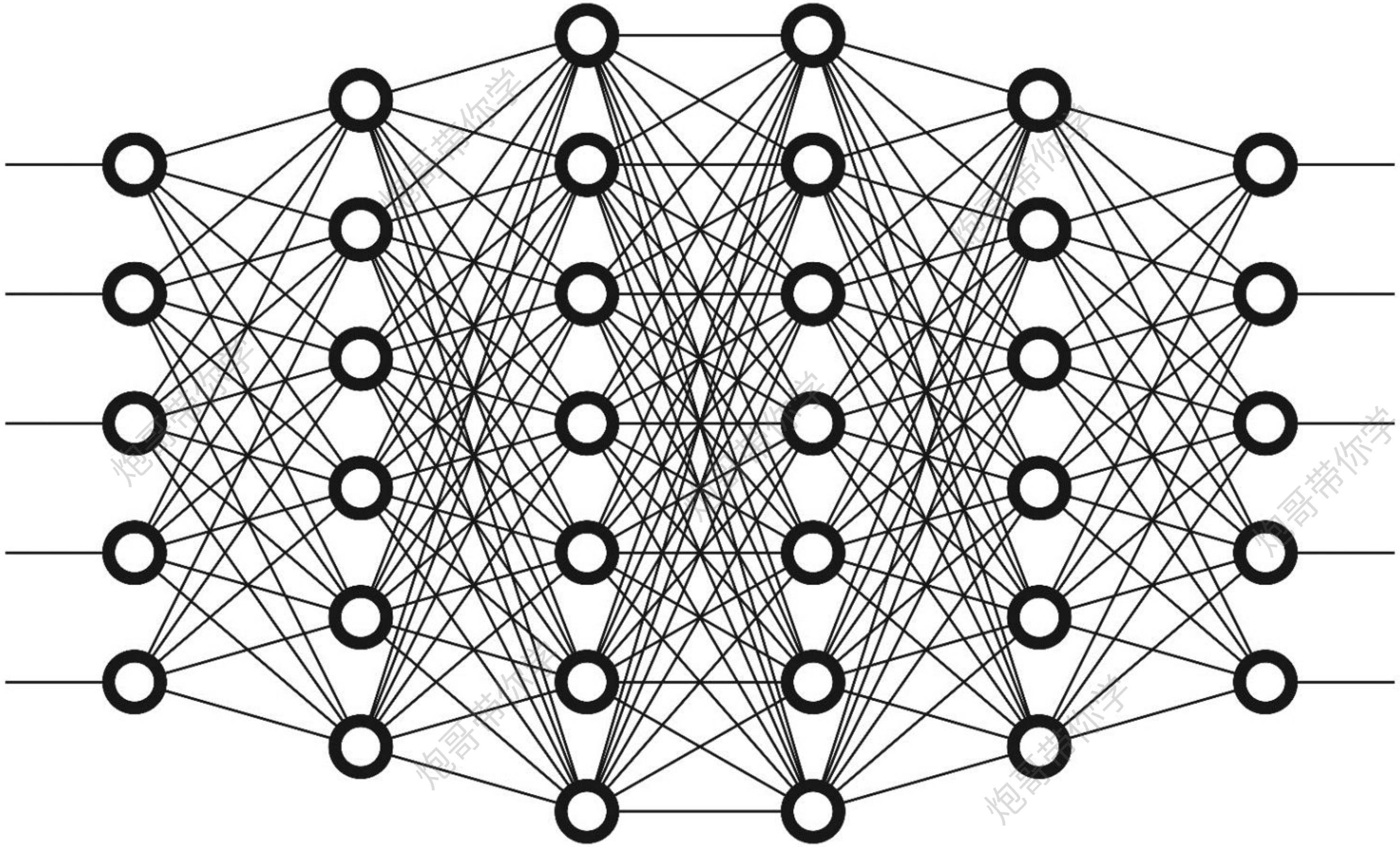
PyTorch

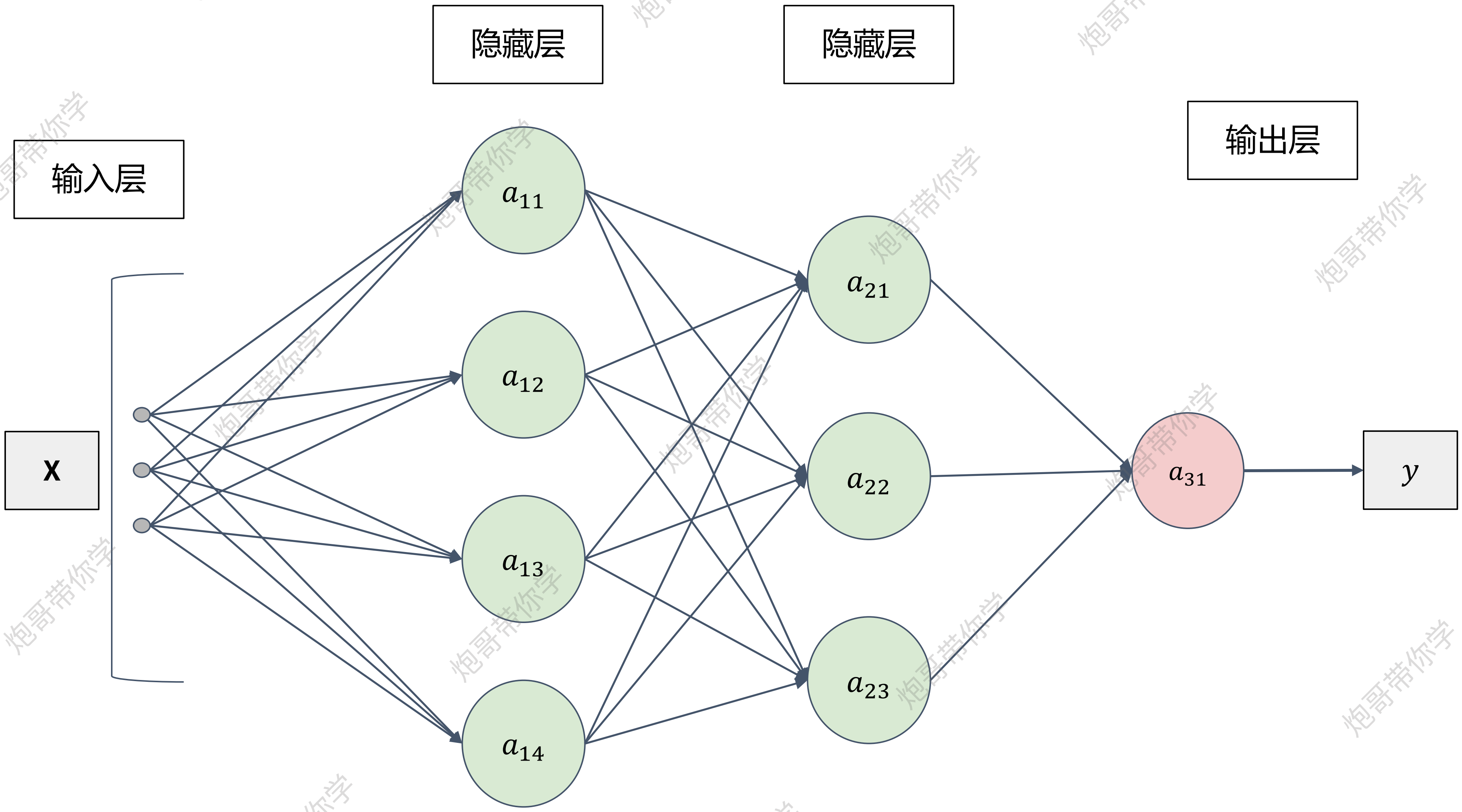


第2章 CNN卷积神经网络算法原理

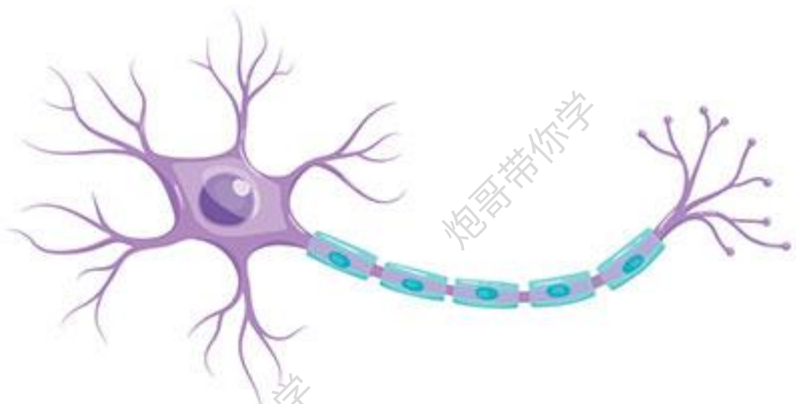
全连接神经网络的整体结构

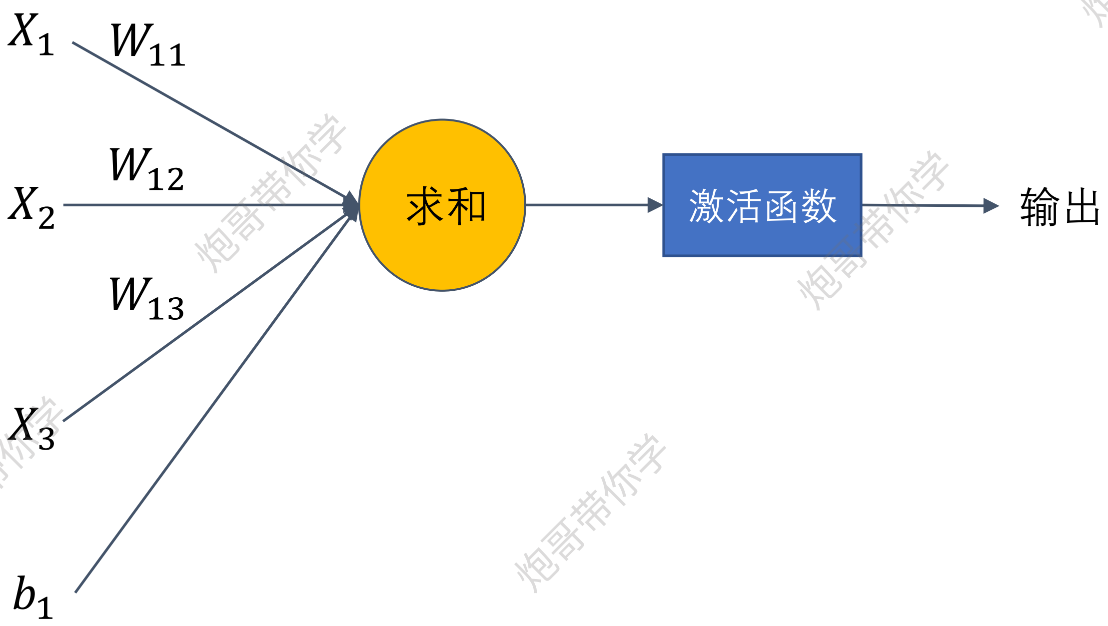


全连接神经网络的整体结构



全连接神经网络的结构单元

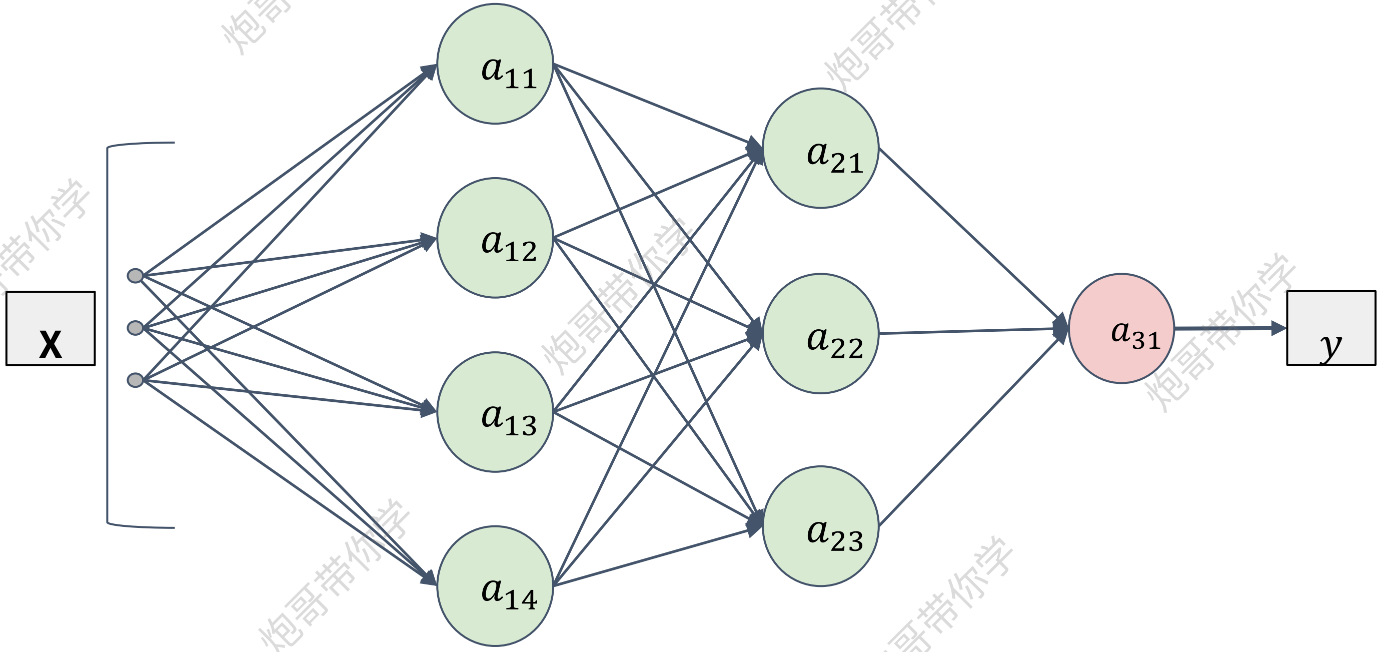




神经元的数学表达式如式所示：

$$a = h(w * x + b)$$

为什么要加入激活函数

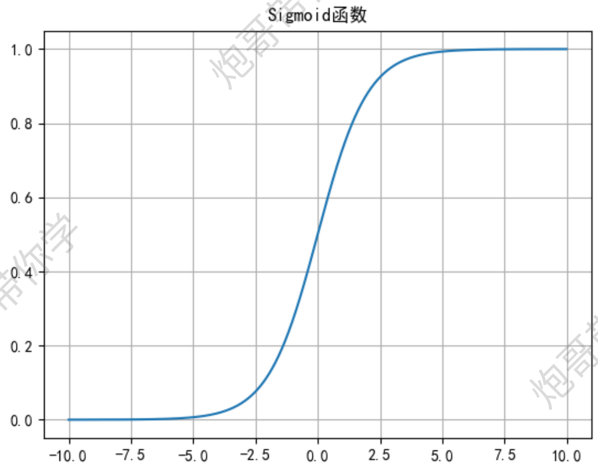


这里我们考虑把线性函数$h(x) = cx$作为激活函数，把$y(x) = h(h(h(x)))$的运算对应3层神经网络，这个运算会进行$y(x) = c * c * c * x$的乘法运算，但是同样的处理可以由$y(x) = ax$（注意，这里$a = c ** 3$）一次乘法运算（既没有隐藏层的神经网络）来表示。

激活函数---Sigmoid函数

Sigmoid函数的公式和导数如式所示：

$$y = \frac{1}{1 + e^{-z}} \Rightarrow y' = y(1 - y)$$



Sigmoid函数优点：

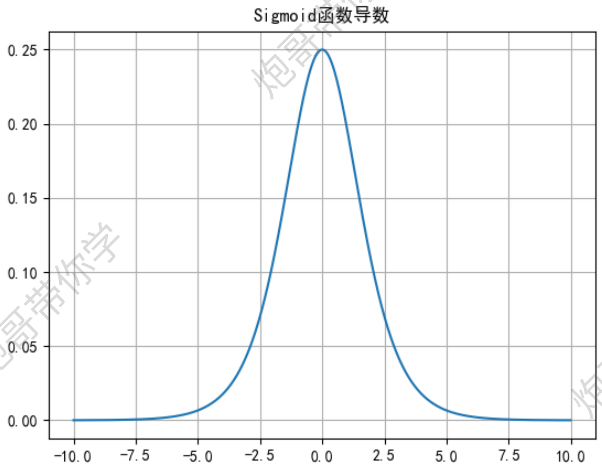
1、简单、非常适用分类任务；

Sigmoid函数缺点：

1、反向传播训练时有梯度消失的问题；

2、输出值区间为(0,1)，关于0不对称；

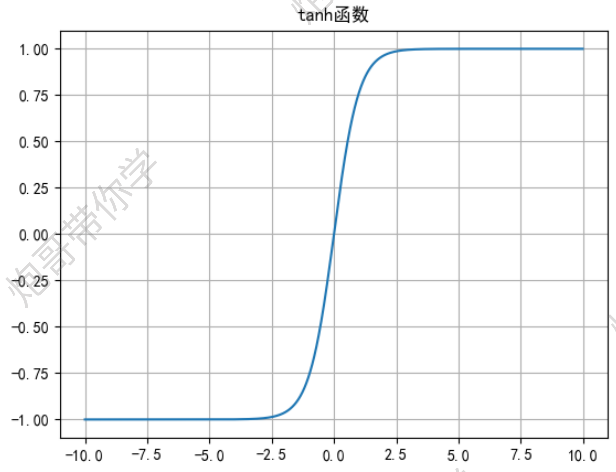
3、梯度更新在不同方向走得太远，使得优化难度增大，训练耗时



激活函数---Tanh函数

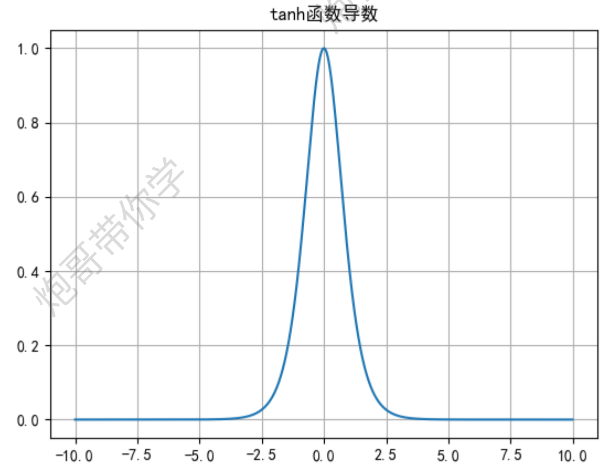
Tanh函数的公式和导数如式所示：

$$y=\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \Rightarrow y'=1-y^{2}$$



Tanh函数优点：

1、解决了Sigmoid函数输出值非0对称的问题
2、训练比Sigmoid函数快，更容易收敛；



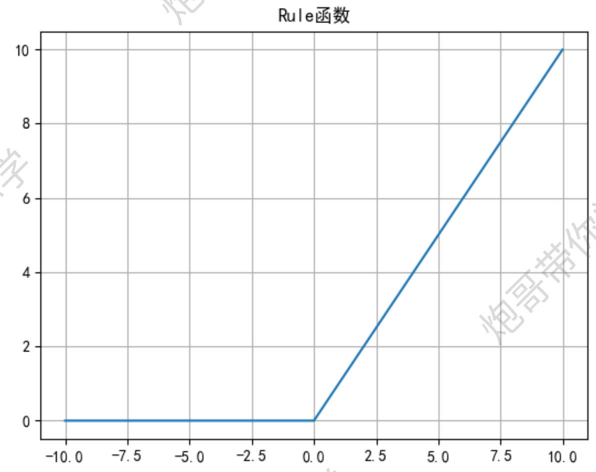
Tanh函数缺点：

1、反向传播训练时有梯度消失的问题；
2、Tanh函数和Sigmoid函数非常相似。

激活函数---ReLU函数

ReLU函数的公式和导数如式所示：

$$y=\begin{cases} z & \text{if } z>0 \\ 0 & \text{if } z<=0 \end{cases} \Longrightarrow y'=\begin{cases} 1 & \text{if } z>0 \\ 0 & \text{if } z<=0 \end{cases}$$



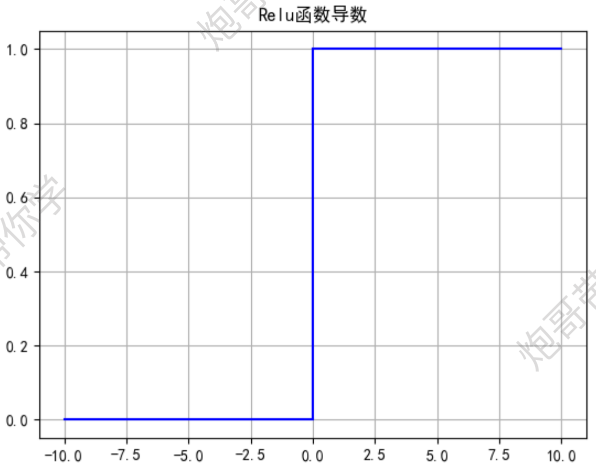
ReLU函数优点：

1、解决了梯度消失的问题；

2、计算更为简单，没有Sigmoid函数和Tanh函数的指数运算；

ReLU函数缺点：

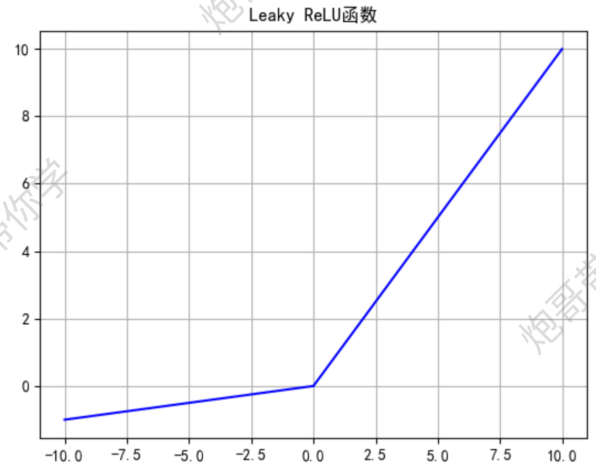
1、训练时可能出现神经元死亡；



激活函数---Leaky ReLU函数

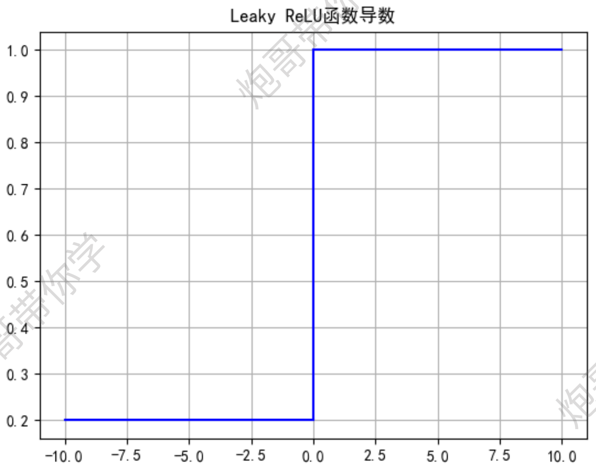
Leaky ReLU函数的公式和导数如式所示：

$$y = \begin{cases} z & \text{if } z > 0 \\ az & \text{if } z <= 0 \end{cases} \Longrightarrow y' = \begin{cases} 1 & \text{if } z > 0 \\ a & \text{if } z <= 0 \end{cases}$$



Leaky ReLU函数优点：

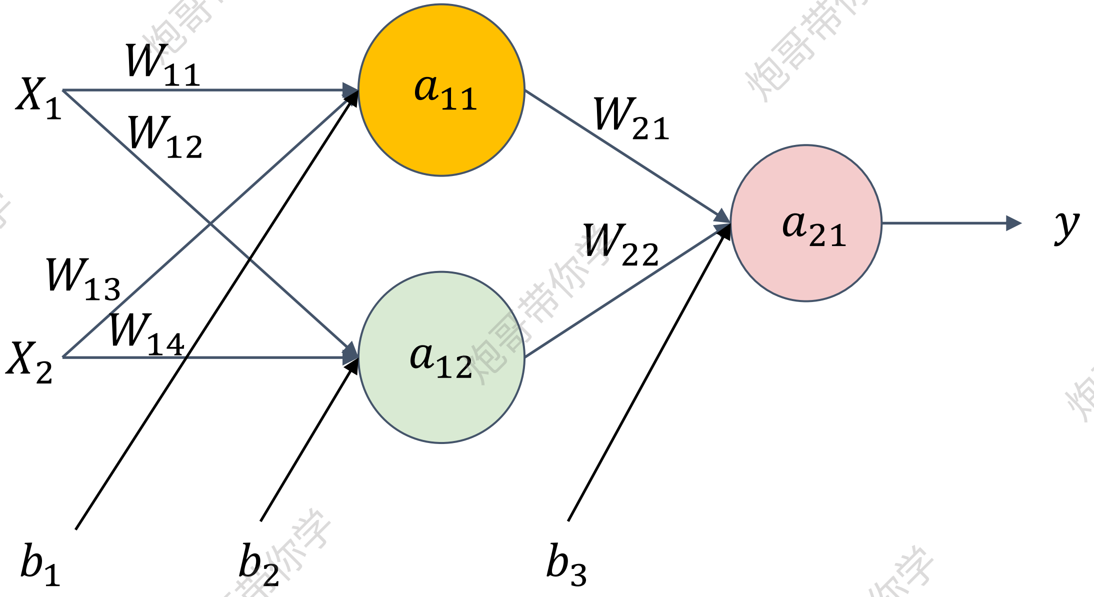
1、解决了ReLU的神经元死亡问题;



Leaky ReLU函数缺点：

1、无法为正负输入值提供一致的关系预测（不同区间函数不同）

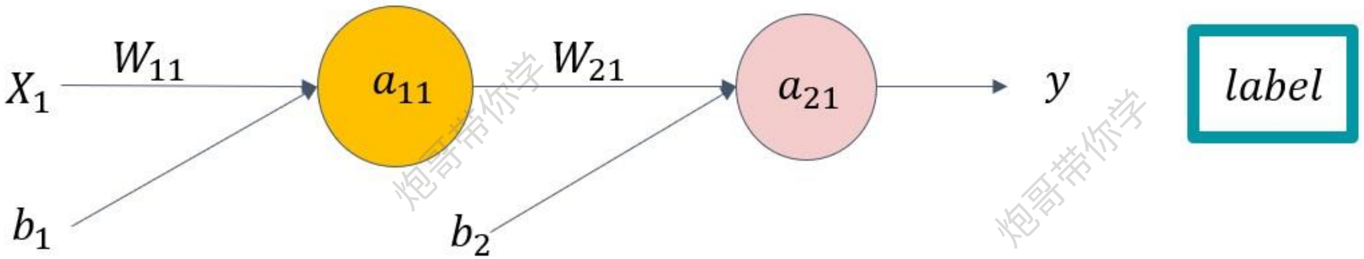
全连接神经网络前向传播



前向传播计算过程：

$$a_{11} = \text{sigmoid}(x_1 w_{11} + x_2 w_{13} + b_1)$$
$$a_{12} = \text{sigmoid}(x_1 w_{12} + x_2 w_{14} + b_2)$$
$$a_{21} = \text{relu}(a_{11} w_{21} + a_{12} w_{22} + b_3)$$
$$y = a_{21}$$

前向传播具体计算过程



前向传播计算公式：

$$a_{11} = \text{sigmoid}(x_1 w_{11} + b_1)$$
$$a_{21} = \text{relu}(a_{11} w_{21} + b_2)$$
$$y = a_{21}$$

具体计算过程：

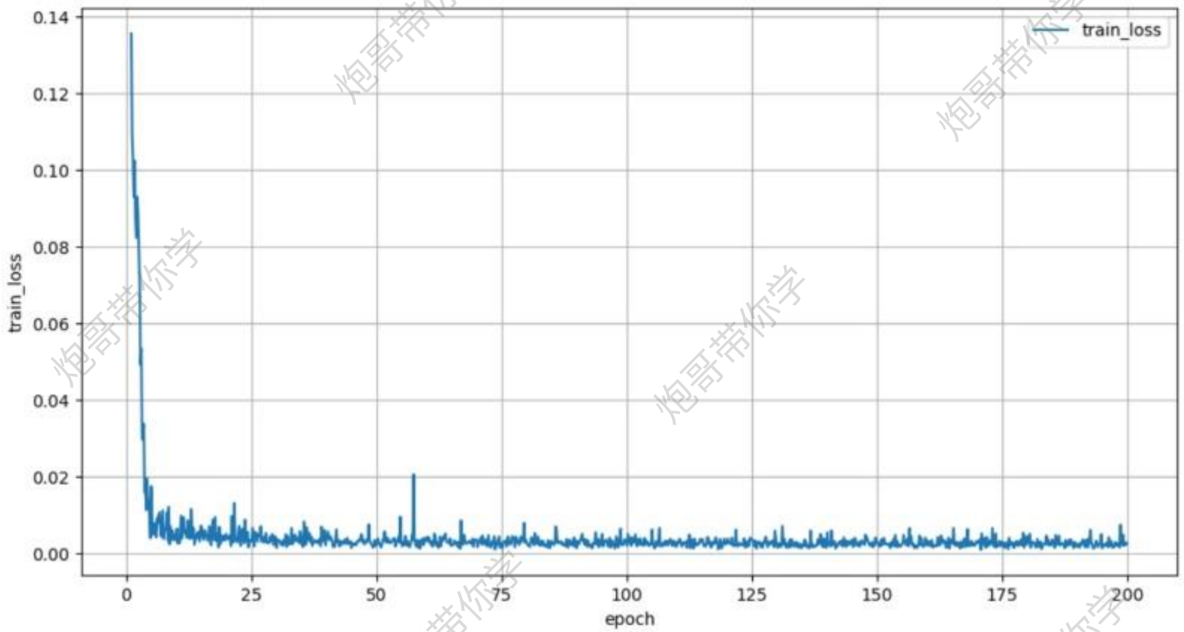
初始化的值为$w_{11} = 0.5$，$w_{21} = 1$，$b_1 = 0.5$，$b_2 = 1$。初始化的值为$w_{11} = 0.5$，$w_{21} = 1$，$b_1 = 0.5$，$b_2 = 1$，$x_1 = 1$，label=2。

$$a_{11} = \text{sigmoid}(1 \times 0.5 + 0.5) = 0.731$$
$$a_{21} = \text{relu}(0.731 \times 1 + 1) = 1.731$$
$$y = 1.731$$

神经网络的损失函数

均方误差的损失函数：

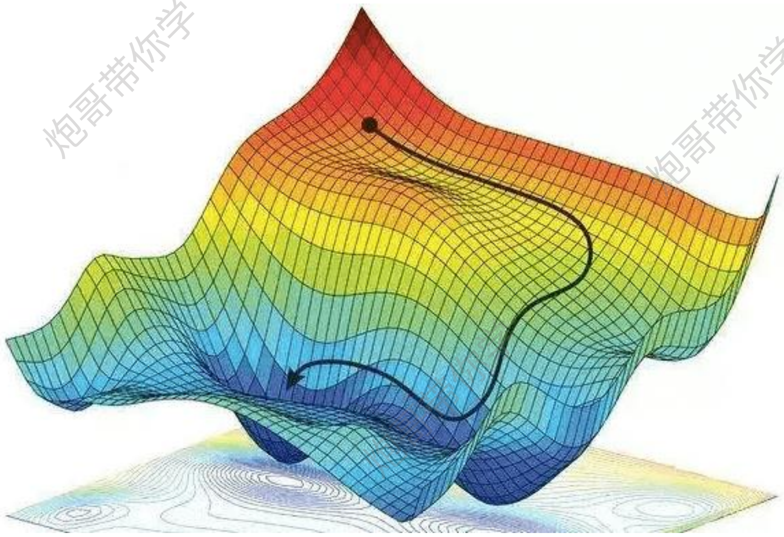
$$J(x) = \frac{1}{2m}\sum_{i=1}^{m}(f(x_i) - y_i)^2$$



梯度下降法

场景：

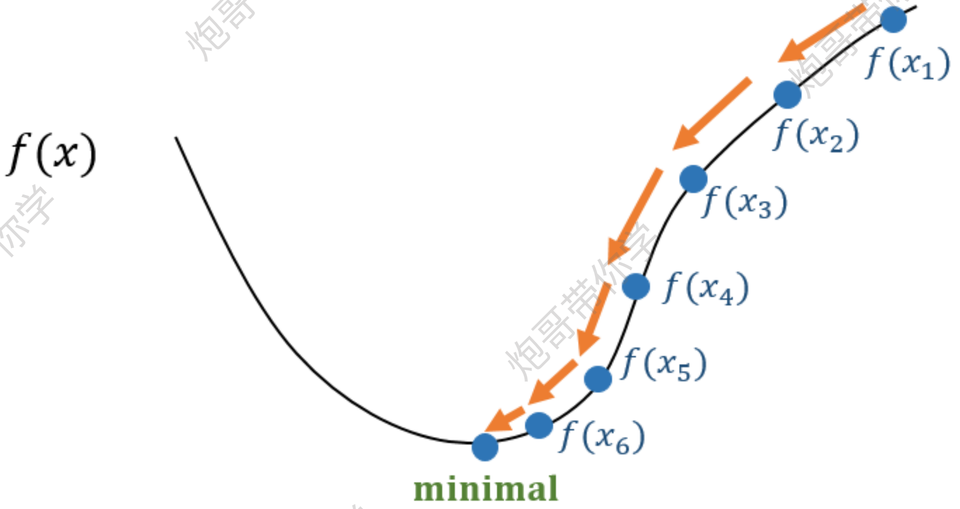
在一个漆黑的夜晚，一个人要下山，但是他完全看不到周围的环境，只能通过手去感知。因此这个人就想到一个办法，朝着自己的四周去摸山体的坡度，如果摸到一个方法的坡度是向下的并且也是最陡峭的，那么就走到这个手摸到的位置，就是通过这样的方法不断一步一步的走，这个人终于走到了山底。具体可以想象成右图，那个黑点就是人。



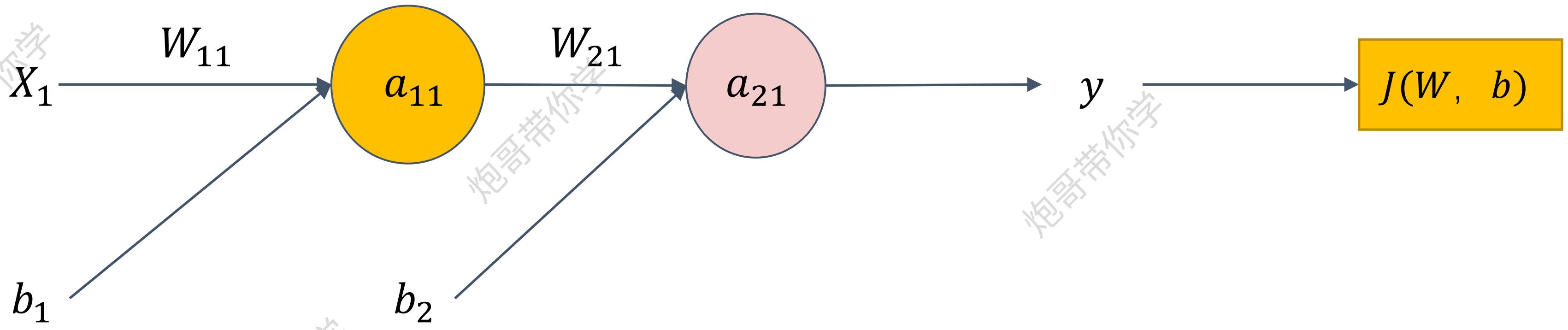
梯度下降法

梯度下降法参数更新的计算公式就如下所示：

$$w = w - a\frac{\partial J(w)}{\partial w}$$



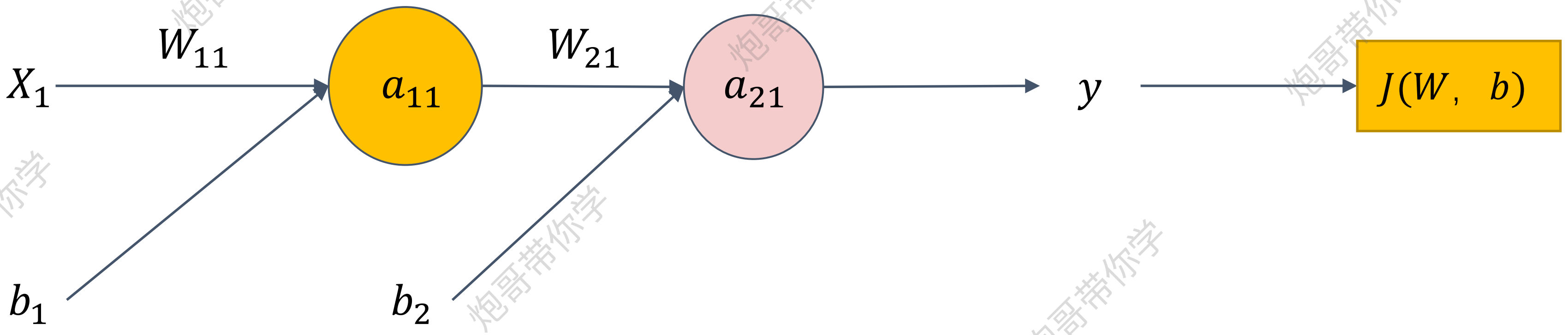
反向传播---案例



案例

从图中可以看到该神经网络链一共有4个参数，分别为w_{11}，w_{21}，b_1，b_2，同时4个参数和输入值x_1分别有对应的初始值，分别为$w_{11} = 0.5$，$w_{21} = 1$，$b_1 = 0.5$，$b_2 = 1$，$x_1 = 1$。同时通过前向传播计算出来的$y = 1.731$，label值为2。现在利用神经网络的前向传播计算出来的输出值和真实值label之间的误差进行反传播进行梯度更新。

反向传播---求解参数梯度



参数w_{21}、b_2的求解：

$$\frac{\partial J(w_{21})}{\partial w_{21}} = \frac{\partial J(w_{21})}{\partial y} \cdot \frac{\partial y}{\partial a_{21}} \cdot \frac{\partial a_{21}}{\partial w_{21}}$$

$$= \frac{\partial \frac{1}{2}(y-2)^2}{\partial y} \cdot \frac{\partial a_{21}}{\partial a_{21}} \cdot \frac{\partial relu(a_{11}w_{21}+b_1)}{\partial w_{21}}$$

$$= (y-2) \times 1 \cdot \frac{\partial relu(a_{11}w_{21}+b_1)}{\partial (a_{11}w_{21}+b_1)} \cdot \frac{\partial (a_{11}w_{21}+b_1)}{\partial w_{21}}$$

$$= (y-2) \times a_{11} \times \frac{\partial relu(a_{11}w_{21}+b_1)}{\partial (a_{11}w_{21}+b_1)}$$

因为：

$$a_{11}w_{21} + b_1 = 1.731 > 0$$

所以：

$$\frac{\partial relu(a_{11}w_{21}+b_1)}{\partial (a_{11}w_{21}+b_1)} = 1$$

最终：

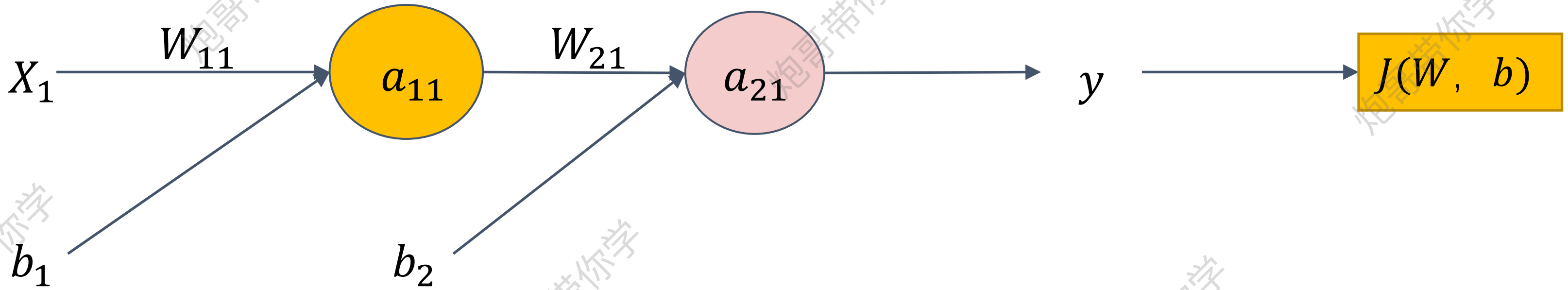
$$\frac{\partial J(w_{21})}{\partial w_{21}} = (y-2) \times a_{11}$$

$$= (1.731-2) \times 0.731 = -0.196639$$

同理可以得出：

$$\frac{\partial J(b_2)}{\partial b_2} = -0.269$$

反向传播---求解参数梯度



参数$\boldsymbol{w_{11}}$、$\boldsymbol{b_1}$的求解：反向传播---求解参数梯度

$$\frac{\partial J(w_{11})}{\partial w_{11}} = \frac{\partial J(w_{11})}{\partial y}\cdot\frac{\partial y}{\partial a_{21}}\cdot\frac{\partial a_{21}}{\partial a_{11}}\cdot\frac{\partial a_{11}}{\partial w_{11}}$$

$$= \frac{\partial \frac{1}{2}(y-2)^2}{\partial y}\cdot\frac{\partial a_{21}}{\partial a_{21}}\cdot\frac{\partial\, relu(a_{11}w_{21}+b_1)}{\partial a_{11}}\cdot\frac{\partial \mathrm{sigmoid}(x_1w_{11}+b_1)}{\partial w_{11}}$$

$$= (y-2)\times 1\times\frac{\partial relu(a_{11}w_{21}+b_1)}{\partial(a_{11}w_{21}+b_1)}\cdot\frac{\partial(a_{11}w_{21}+b_1)}{\partial a_{11}}\cdot\frac{\partial \mathrm{sigmoid}(x_1w_{11}+b_1)}{\partial w_{11}}$$

$$= (y-2)\times w_{21}\cdot\frac{\partial \mathrm{sigmoid}(x_1w_{11}+b_1)}{\partial w_{11}}$$

$$= (y-2)\times w_{21}\times \mathrm{sigmoid}(x_1w_{11}+b_1)\big(1-\mathrm{sigmoid}(x_1w_{11}+b_1)x_1$$

$$= (1.731-2)\times 1\times 0.731\times(1-0.731)\times 1$$

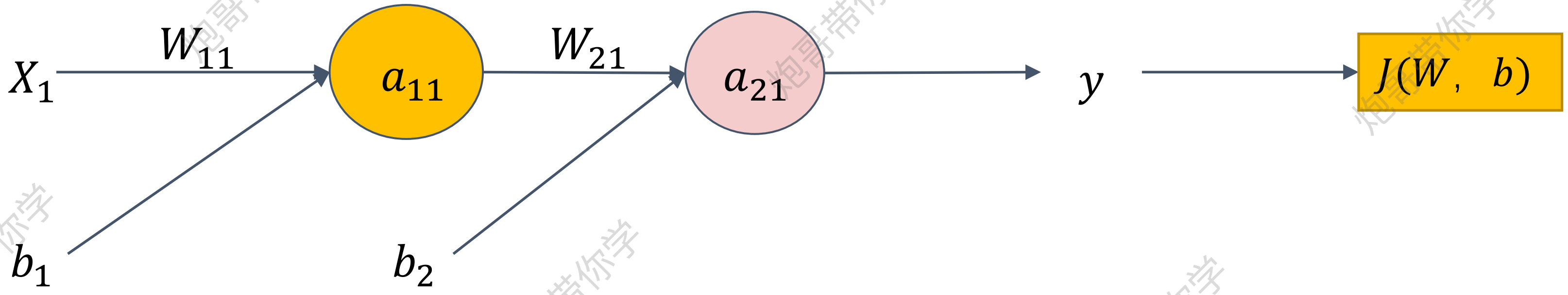
$$= -0.269\times 0.731\times 0.269\times 1$$

$$= -0.052895$$

同理可以得出：

$$\frac{\partial J(b_1)}{\partial b_1} = -0.052895$$

反向传播---利用梯度参数更新



对应参数的梯度：

$$\frac{\partial J(w_{21})}{\partial w_{21}} = -0.196639, \quad \frac{\partial J(b_2)}{\partial b_2} = -0.269,$$
$$\frac{\partial J(w_{11})}{\partial w_{11}} = -0.052895, \quad \frac{\partial J(b_1)}{\partial b_1} = -0.052895$$

设置学习率大小为0.1：

$$w_{11} = 0.5 - 0.1 \times (-0.053) = 0.505$$
$$w_{21} = 1 - 0.1 \times (-0.197) = 1.020$$
$$b_1 = 0.5 - 0.1 \times (-0.053) = 0.505$$
$$b_2 = 1 - 0.1 \times (-0.269) = 1.027$$

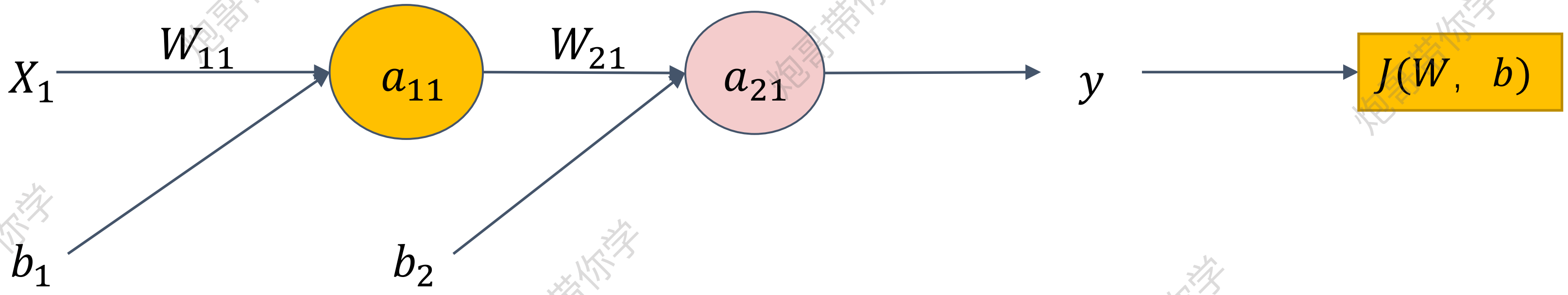
保留3位小数方便计算：

$$\frac{\partial J(w_{21})}{\partial w_{21}} = -0.197, \quad \frac{\partial J(b_2)}{\partial b_2} = -0.269,$$
$$\frac{\partial J(w_{11})}{\partial w_{11}} = -0.053, \quad \frac{\partial J(b_1)}{\partial b_1} = -0.053$$

整理一下：

$$w_{11} = 0.505$$
$$w_{21} = 1.020$$
$$b_1 = 0.505$$
$$b_2 = 1.027$$

反向传播---新参数前向传播



新参数：

$$w_{11} = 0.505$$
$$w_{21} = 1.020$$
$$b_1 = 0.505$$
$$b_2 = 1.027$$

前向传播：

$$a_{11} = \text{sigmoid}(1 \times 0.505 + 0.505) = 0.733$$
$$a_{21} = \text{relu}(0.733 \times 1.02 + 1.027) = 1.775$$
$$y = 1.775$$

结论：

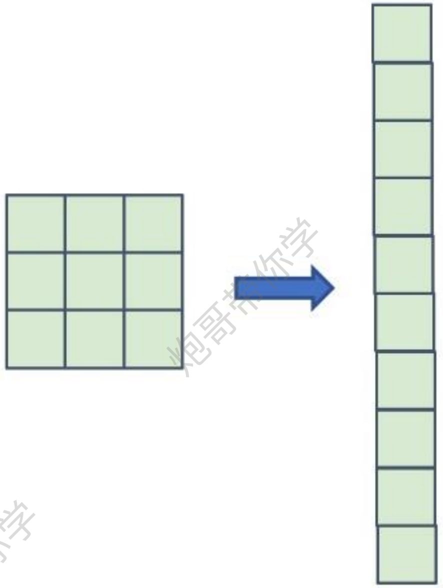
此时最终的输出结果是 $y = 1.775$，可以看出这个数值比之前的前向传播输出值1.731要更加接近真实值2了

图像在计算机中的的本质是什么?

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

彩色图像在计算机中是什么样子的？

全连接神经网络存在的问题



卷积层---卷积运算

输入 / 核 / 输出

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 19 & 25 \\ 37 & 43 \end{bmatrix}$$

卷积层---卷积运算过程

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 19 & \\ & \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 19 & 25 \\ & \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 19 & 25 \\ 37 & \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 19 & 25 \\ 37 & 43 \end{bmatrix}$$

卷积层---加入偏置的卷积运算

输入		核		偏置		输出
$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$	*	$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$	+	$\begin{bmatrix} 2 \end{bmatrix}$	=	$\begin{bmatrix} 21 & 27 \\ 39 & 45 \end{bmatrix}$

卷积层---填充

$$
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 3 & 4 & 5 & 0 \\ 0 & 6 & 7 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 8 & 4 \\ 9 & 19 & 25 & 10 \\ 21 & 37 & 43 & 16 \\ 6 & 7 & 8 & 0 \end{bmatrix}
$$

卷积层---步幅

$$\begin{bmatrix}1&2&3&4\\5&9&6&7\\9&0&3&2\\1&2&3&4\end{bmatrix} * \begin{bmatrix}1&2\\3&4\end{bmatrix} = \begin{bmatrix}56&\\&\end{bmatrix}$$

$$\begin{bmatrix}1&2&3&4\\5&9&6&7\\9&0&3&2\\1&2&3&4\end{bmatrix} * \begin{bmatrix}1&2\\3&4\end{bmatrix} = \begin{bmatrix}56&57\\&\end{bmatrix}$$

$$\begin{bmatrix}1&2&3&4\\5&9&6&7\\9&0&3&2\\1&2&3&4\end{bmatrix} * \begin{bmatrix}1&2\\3&4\end{bmatrix} = \begin{bmatrix}56&57\\20&\end{bmatrix}$$

$$\begin{bmatrix}1&2&3&4\\5&9&6&7\\9&0&3&2\\1&2&3&4\end{bmatrix} * \begin{bmatrix}1&2\\3&4\end{bmatrix} = \begin{bmatrix}56&57\\20&32\end{bmatrix}$$

卷积层---经过卷积运算后的特征图大小

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 3 & 4 & 5 & 0 \\ 0 & 6 & 7 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 8 & 4 \\ 9 & 19 & 25 & 10 \\ 21 & 37 & 43 & 16 \\ 6 & 7 & 8 & 0 \end{bmatrix}$$

计算公式：

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

$$OH = \frac{3 + 2 * 1 - 2}{1} + 1 = 4$$

$$OW = \frac{3 + 2 * 1 - 2}{1} + 1 = 4$$

卷积层---经过卷积运算后的特征图大小

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 9 & 6 & 7 \\ 9 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 56 & 57 \\ 20 & 32 \end{bmatrix}$$

计算公式：

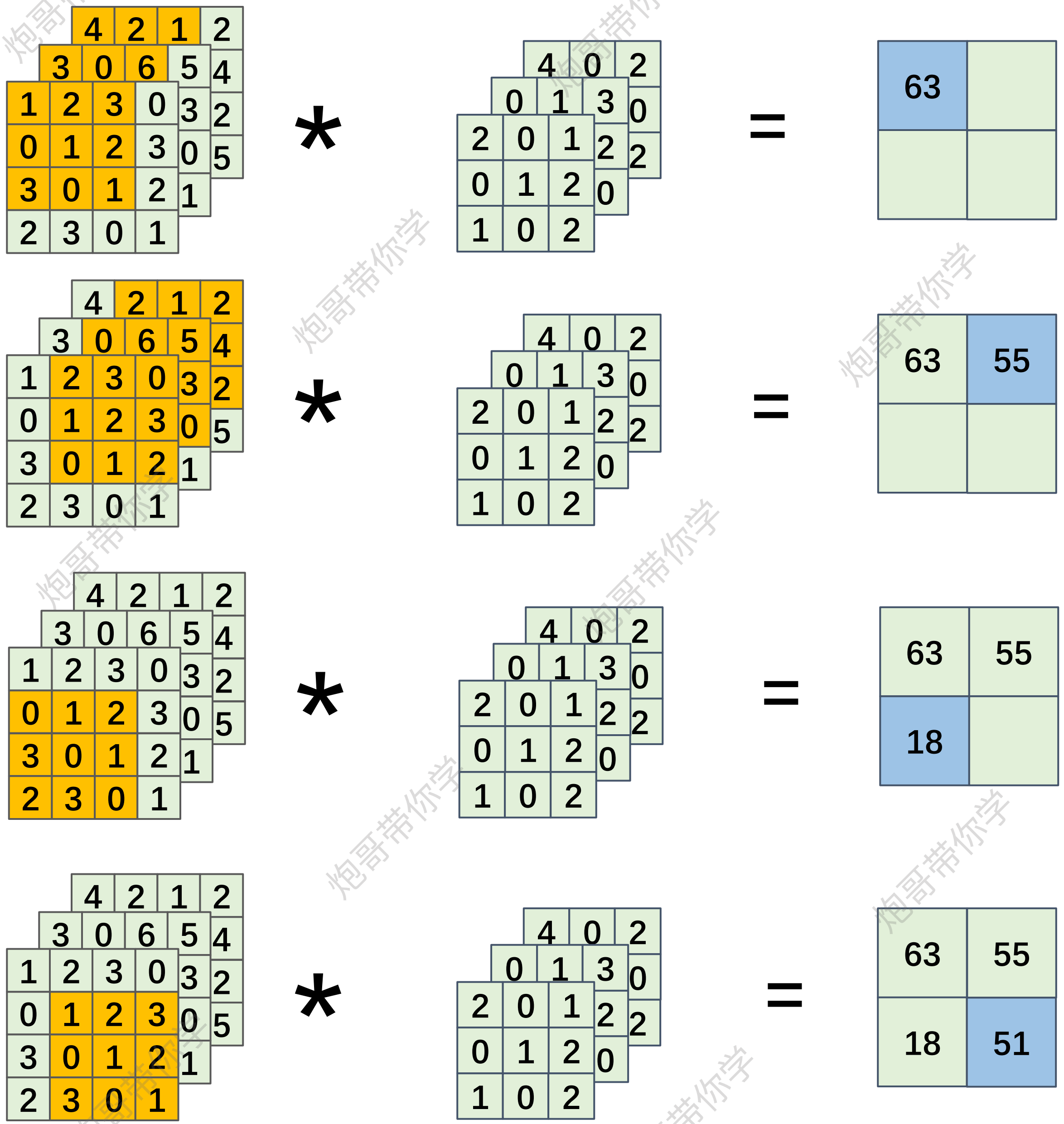
$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

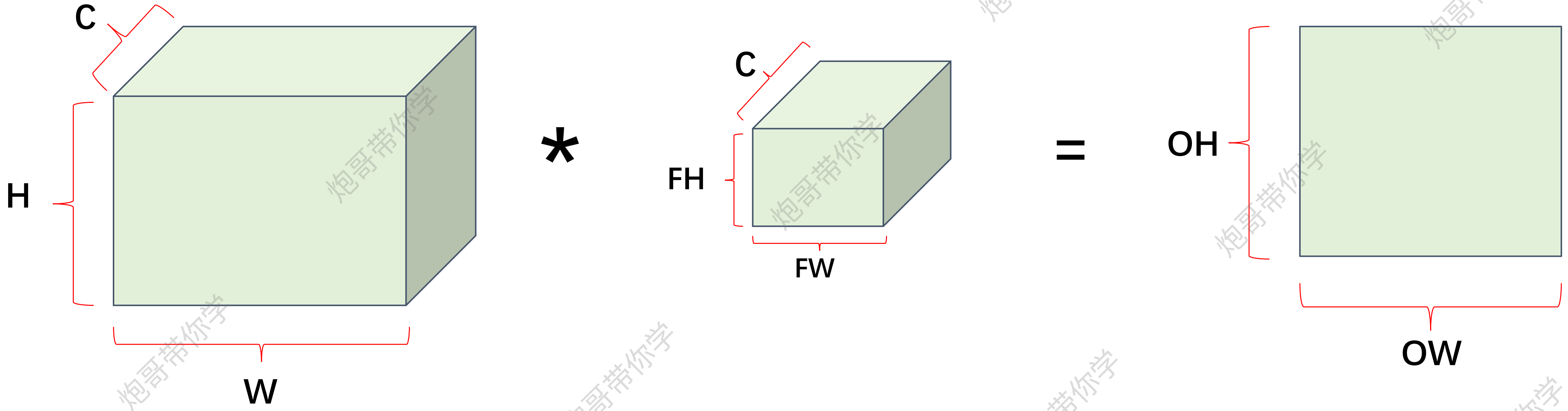
$$OH = \frac{4 + 2 * 0 - 2}{2} + 1 = 2$$

$$OW = \frac{4 + 2 * 0 - 2}{2} + 1 = 2$$

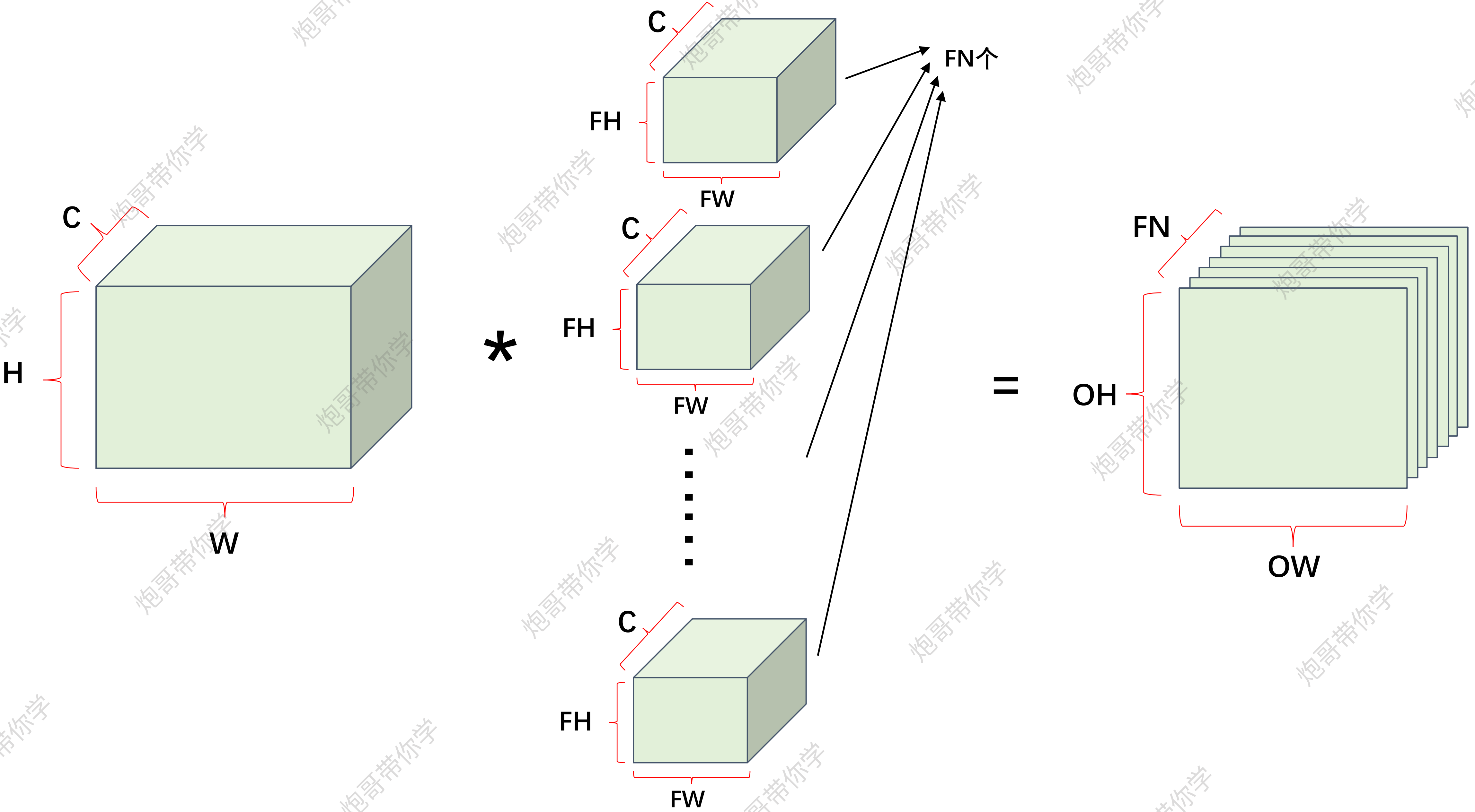
卷积层---多通道数据卷积运算



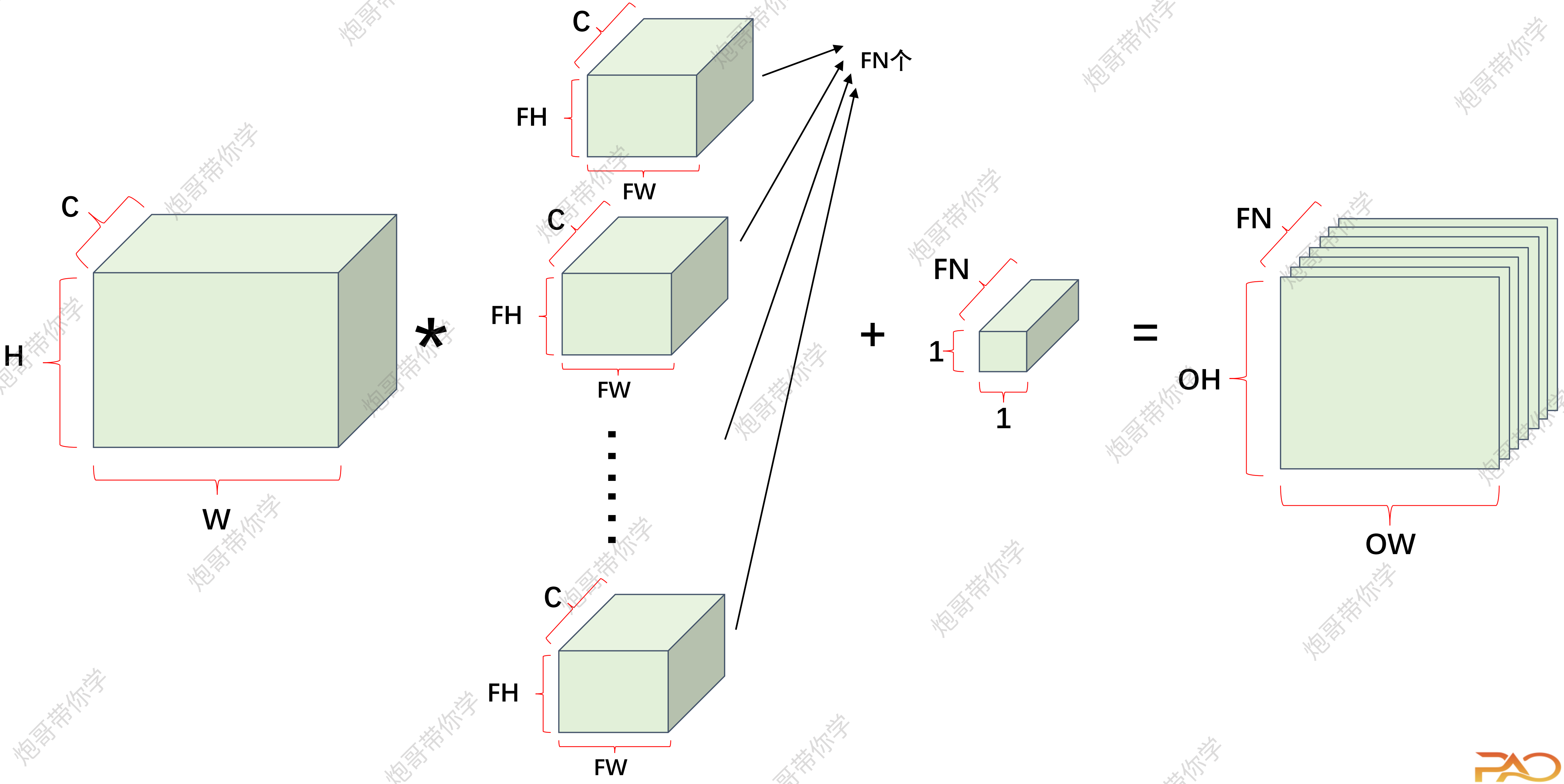
卷积层---利用立体图来表示卷积运算



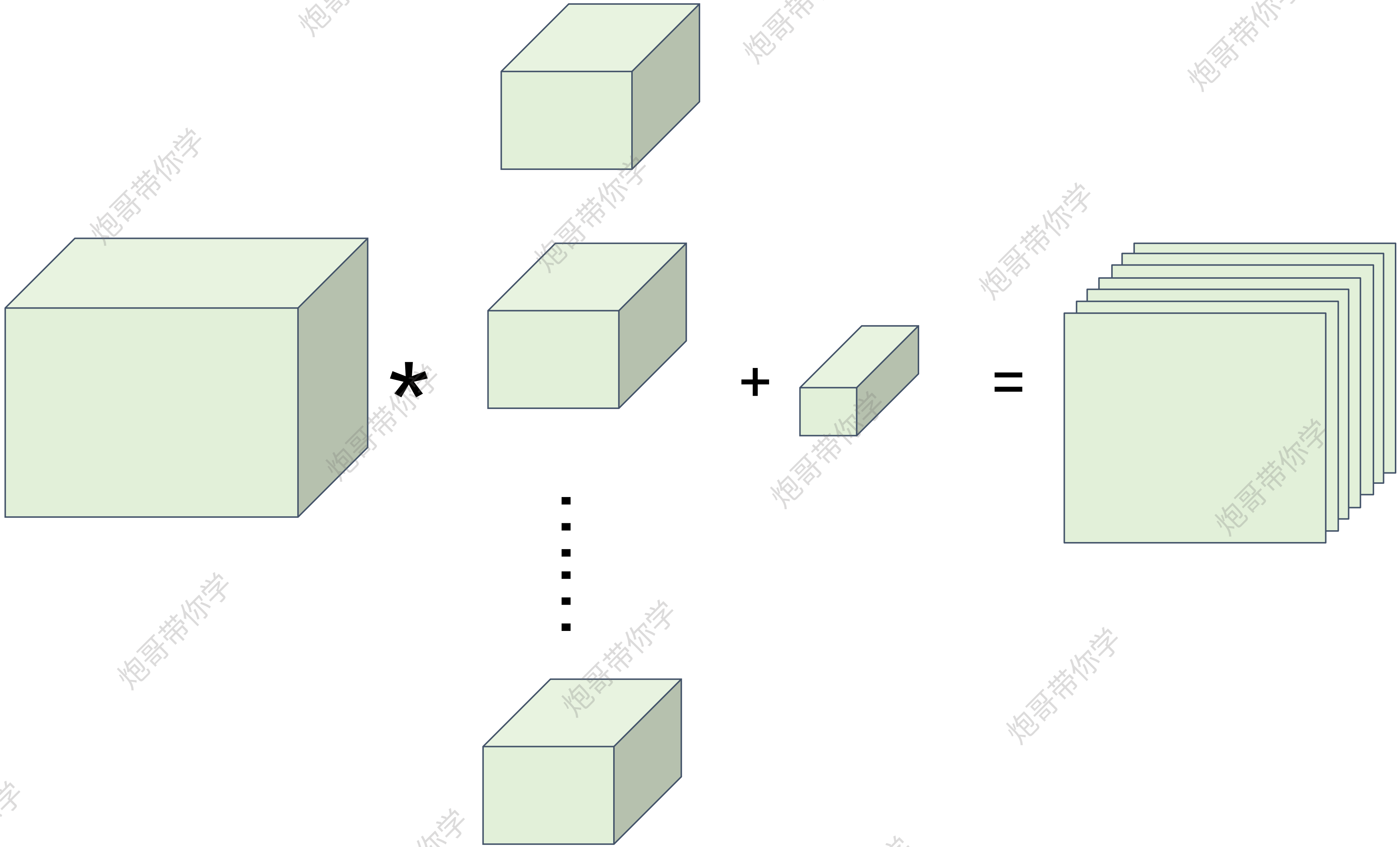
卷积层---利用立体图来表示卷积运算



卷积层---利用立体图来表示卷积运算



卷积层---利用立体图来表示卷积运算



池化层---最大池化运算

1	2	3	3
0	1	1	5
3	0	1	0
1	4	1	2

→

2	

1	2	3	3
0	1	1	5
3	0	1	0
1	4	1	2

→

2	5

1	2	3	3
0	1	1	5
3	0	1	0
1	4	1	2

→

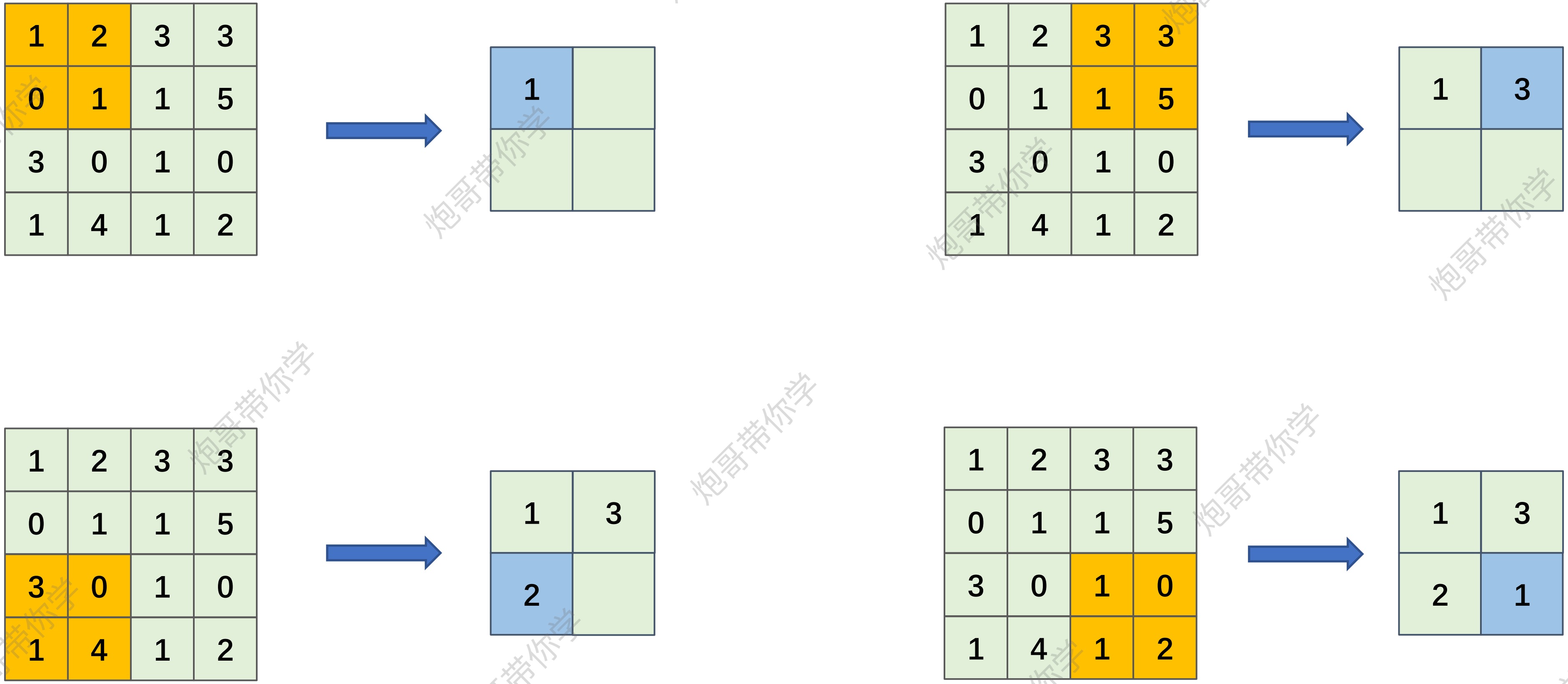
2	5
4	

1	2	3	3
0	1	1	5
3	0	1	0
1	4	1	2

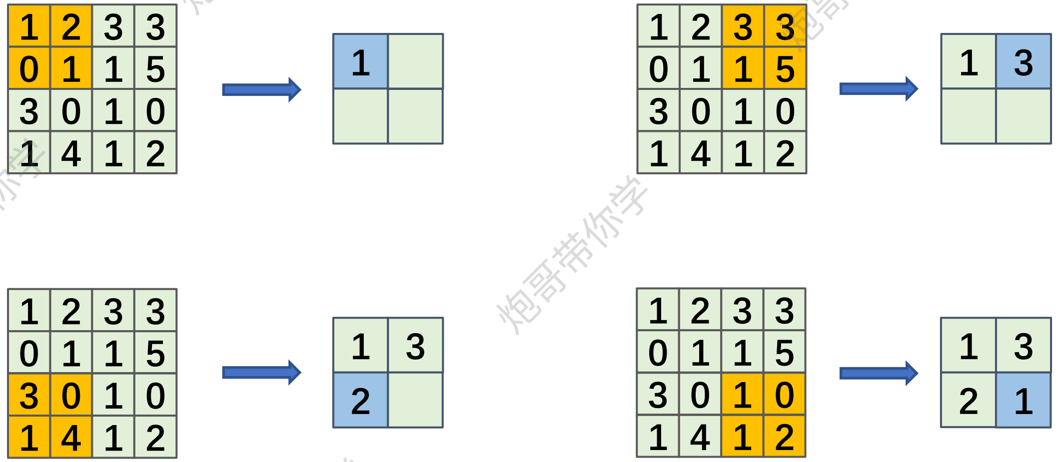
→

2	5
4	2

池化层---平均池化运算


1 2 3 3
0 1 1 5
3 0 1 0
1 4 1 2
1
1 2 3 3
0 1 1 5
3 0 1 0
1 4 1 2
1 3
1 2 3 3
0 1 1 5
3 0 1 0
1 4 1 2
1 3
2
1 2 3 3
0 1 1 5
3 0 1 0
1 4 1 2
1 3
2 1

池化层---经过池化层后的特征图大小



计算公式：

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

$$OH = \frac{4 + 2 * 0 - 2}{2} + 1 = 2$$

$$OW = \frac{4 + 2 * 0 - 2}{2} + 1 = 2$$

卷积神经网络整体结构