

雷达影像的图像处理技术

谢磊 博士 副研究员

国家水运安全工程技术研究中心

2021-03-25

第四章 人工神经网络简介 (二)

人工神经网络的实现

Matlab程序

```
clc;
```

```
clear;
```

```
% 以下是训练数据
```

```
I = double(rgb2gray(imread('D:/神经网络样本/0.jpg')));
```

```
J0 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/1.jpg')));
```

```
J1 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/2.jpg')));
```

```
J2 = I(:);
```

Matlab程序

```
I = double(rgb2gray(imread('D:/神经网络样本/3.jpg')));
```

```
J3 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/4.jpg')));
```

```
J4 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/5.jpg')));
```

```
J5 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/6.jpg')));
```

```
J6 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/7.jpg')));
```

```
J7 = I(:);
```

Matlab程序

```
I = double(rgb2gray(imread('D:/神经网络样本/8.jpg')));
```

```
J8 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/9.jpg')));
```

```
J9 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/10.jpg')));
```

```
J10 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/11.jpg')));
```

```
J11 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/12.jpg')));
```

```
J12 = I(:);
```

Matlab程序

```
I = double(rgb2gray(imread('D:/神经网络样本/13.jpg')));
```

```
J13 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/14.jpg')));
```

```
J14 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/15.jpg')));
```

```
J15 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/16.jpg')));
```

```
J16 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/17.jpg')));
```

```
J17 = I(:);
```

Matlab程序

```
I = double(rgb2gray(imread('D:/神经网络样本/18.jpg')));  
J18 = I(:);  
I = double(rgb2gray(imread('D:/神经网络样本/19.jpg')));  
J19 = I(:);
```


Matlab程序

% 以下是测试数据。

```
I = double(rgb2gray(imread('D:/神经网络样本/T0.jpg')));
```

```
JT0 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/T1.jpg')));
```

```
JT1 = I(:);
```

```
I = double(rgb2gray(imread('D:/神经网络样本/T10.jpg')));
```

```
JT10 = I(:);
```

Matlab程序

%输入输出数据， 10个样本为训练样本， 2个样本为预测样本

```
input_train =  
[J0,J1,J2,J3,J4,J5,J6,J7,J8,J9,J10,J11,J12,J13,J14,J15,J16,J17,  
J18,J19];
```

```
output_train = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0];
```

```
input_test = [JT0,JT1,JT10];
```

```
output_test = [1,1,0];
```

%输入数据归一化

```
[inputn,inputps] = mapminmax(input_train);
```

Matlab程序

%初始化网络结构

```
net = newff(inputn,output_train,10);
```

```
net.trainParam.epochs=10;%最大训练次数， 1000
```

```
net.trainParam.lr=0.1;%学习速率， 0.1
```

```
net.trainParam.goal=0.001;%网络性能目标， 0.00000004
```

%网络训练

```
net = train(net,inputn,output_train);
```

Matlab程序

% 预测数据归一化

```
inputn_test = mapminmax('apply',input_test,inputps);
```

% 网络预测输出

```
BPoutput = sim(net,inputn_test);
```

% 根据网络输出找出数据属于哪类

```
BPoutput(find(BPoutput<0.5)) = 0;
```

```
BPoutput(find(BPoutput>=0.5)) = 1;
```

Matlab程序

```
%画出预测类别和实际类别的分类图  
figure(1)  
plot(BPoutput,'og')  
hold on  
plot(output_test,'r*');  
legend('预测类别','实际类别')  
title('神经网络预测类别与实际类别比对','fontsize',12)  
ylabel('类别标签','fontsize',12)  
xlabel('样本数目','fontsize',12)  
ylim([-0.5 1.5])
```

Matlab程序

```
% 预测正确率
rightnumber=0;
for i=1:size(output_test,2)
    if BPoutput(i)==output_test(i)
        rightnumber=rightnumber+1;
    end
end
rightratio = rightnumber/size(output_test,2)*100;
sprintf('测试准确率=%0.2f',rightratio)
```

matlab相关函数说明

- `net = newff(P,T,[S1 S2...S(N-1)],{TF1 TF2...TFN1}, BTF,BLF,PF,IPF,OPF,DDF)`

参数:

- **P**: 输入参数矩阵, ($R \times Q1$), 即: 输入数据中一共有 $Q1$ 个 R 元的输入向量。其数据意义是矩阵 P 有 $Q1$ 列, 每一列都是一个样本, 而每个样本有 R 个属性(特征)。一般矩阵 P 需要归一化, 即 P 的每一行都归一化到 $[0 \ 1]$ 或者 $[-1 \ 1]$ 。
- **T**: 目标参数矩阵, ($SN \times Q2$), 即: 输入数据中一共有 $Q2$ 个 SN 元的目标向量。

matlab相关函数说明

- **S**: N-1个隐含层的数目（从S(1)到S(N-1)），默认为空矩阵[]。输出层的单元数目SN取决于矩阵T，返回N层的前馈BP神经网络。
- **TF**: 相关层的传递函数，默认隐含层为tansig函数，输出层为purelin函数。
- **BTF**: BP神经网络学习训练函数，默认值为trainlm函数。
- **BLF**: 权重学习函数，默认值为learngdm。
- **PF**: 性能函数，默认值为mse，可选择的还有sse, sae, mae, crossentropy。
- **IPF, OPF, DDF**均为默认值即可。

matlab相关函数说明

- 传递函数TF

`purelin`: 线性传递函数。

`tansig`: 正切S型传递函数。

`logsig`: 对数S型传递函数。

隐含层和输出层函数的选择对BP神经网络预测精度有较大影响，一般隐含层节点转移函数选用 `tansig` 函数或 `logsig` 函数，输出层节点转移函数选用 `tansig` 函数或 `purelin` 函数。

matlab相关函数说明

- 学习训练函数BTF

traingd: 最速下降BP算法。

traindm: 动量BP算法。

trainda: 学习率可变的最速下降BP算法。

traindx: 学习率可变的动量BP算法。

trainrp: 弹性算法。

变梯度算法: traincgf (Fletcher-Reeves修正算法)

traincgp (Polak_Ribiere修正算法)

traincgb (Powell-Beale复位算法)

trainbfg (BFGS 拟牛顿算法)

trainoss (OSS算法)

matlab相关函数说明

- 通过net.trainParam可以查看参数

Show Training Window Feedback showWindow: true

Show Command Line Feedback showCommandLine: false

Command Line Frequency show: 两次显示之间的训练次数

Maximum Epochs epochs: 训练次数

Maximum Training Time time: 最长训练时间（秒）

Performance Goal goal: 网络性能目标

Minimum Gradient min_grad: 性能函数最小梯度

Maximum Validation Checks max_fail: 最大验证失败次数

Learning Rate lr: 学习速率

matlab相关函数说明

- 通过net.trainParam可以查看参数

Learning Rate Increase lr_inc: 学习速率增长值

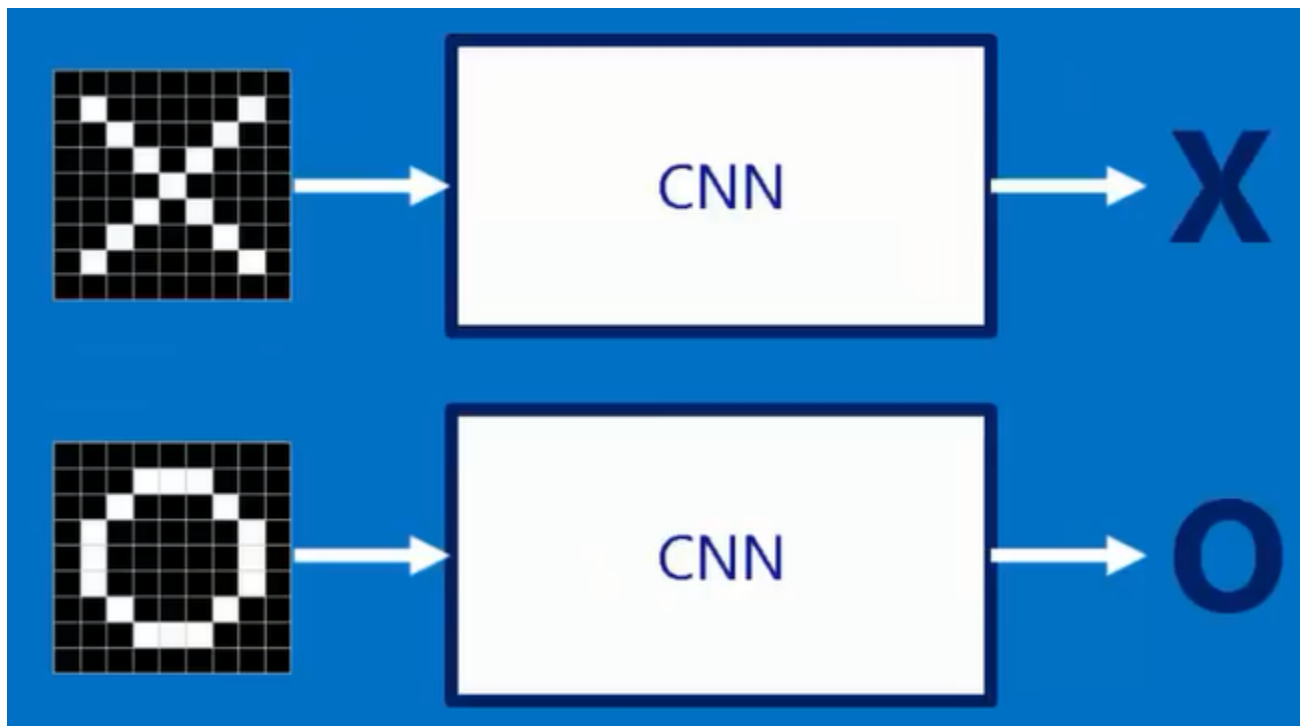
Learning Rate lr_dec: 学习速率下降值

Maximum Performance Increase max_perf_inc:

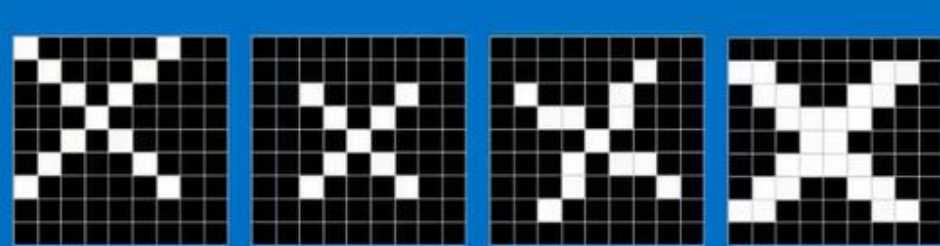
Momentum Constant mc: 动量因子

卷积神经网络简介

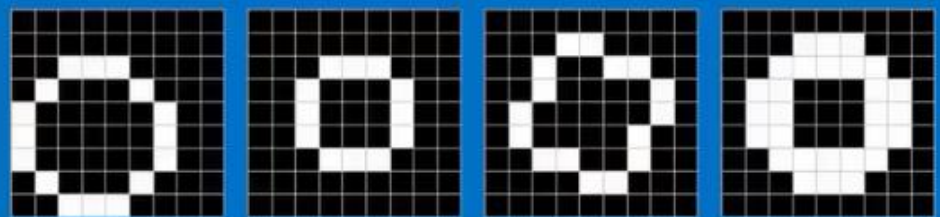
卷积神经网络处理流程



卷积神经网络处理流程

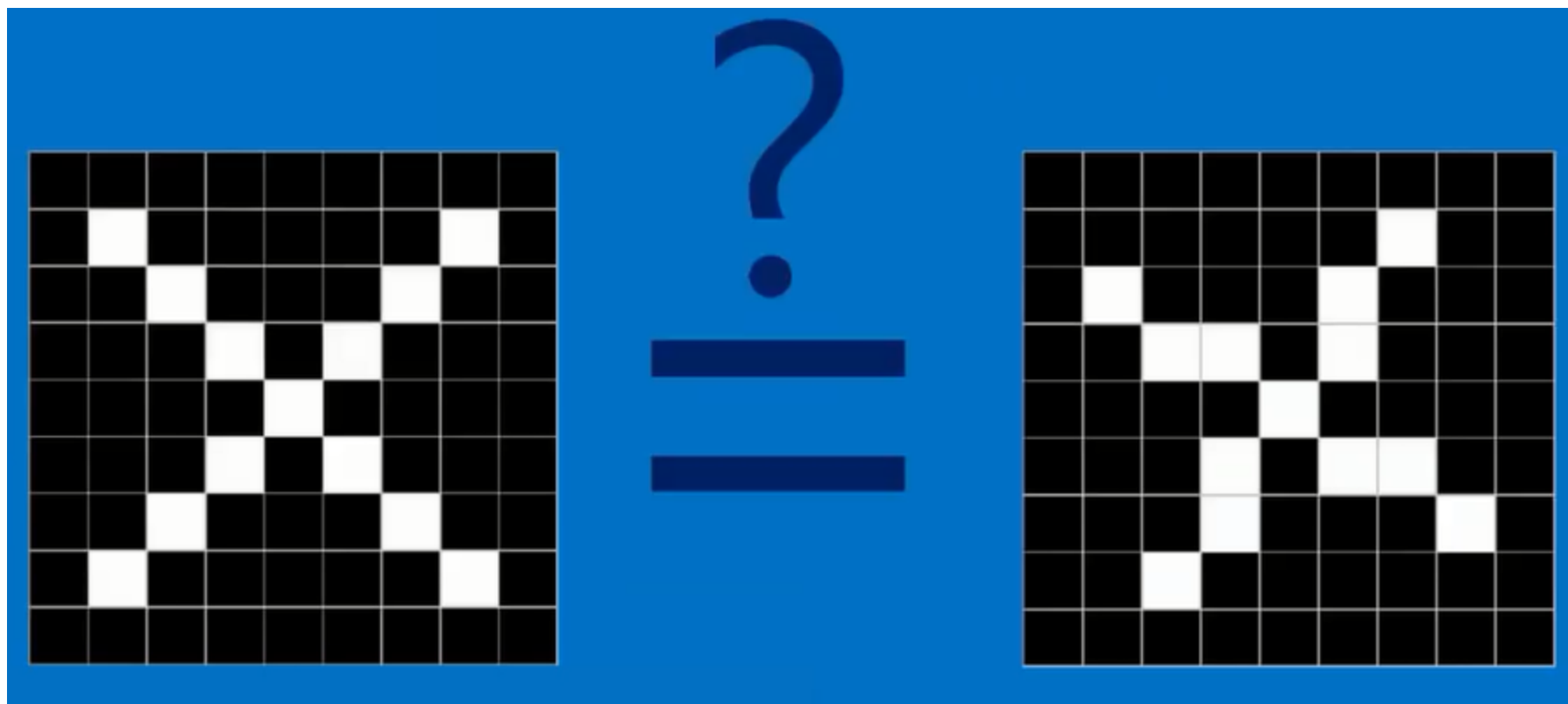


X



O

卷积神经网络处理流程



卷积神经网络处理流程

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

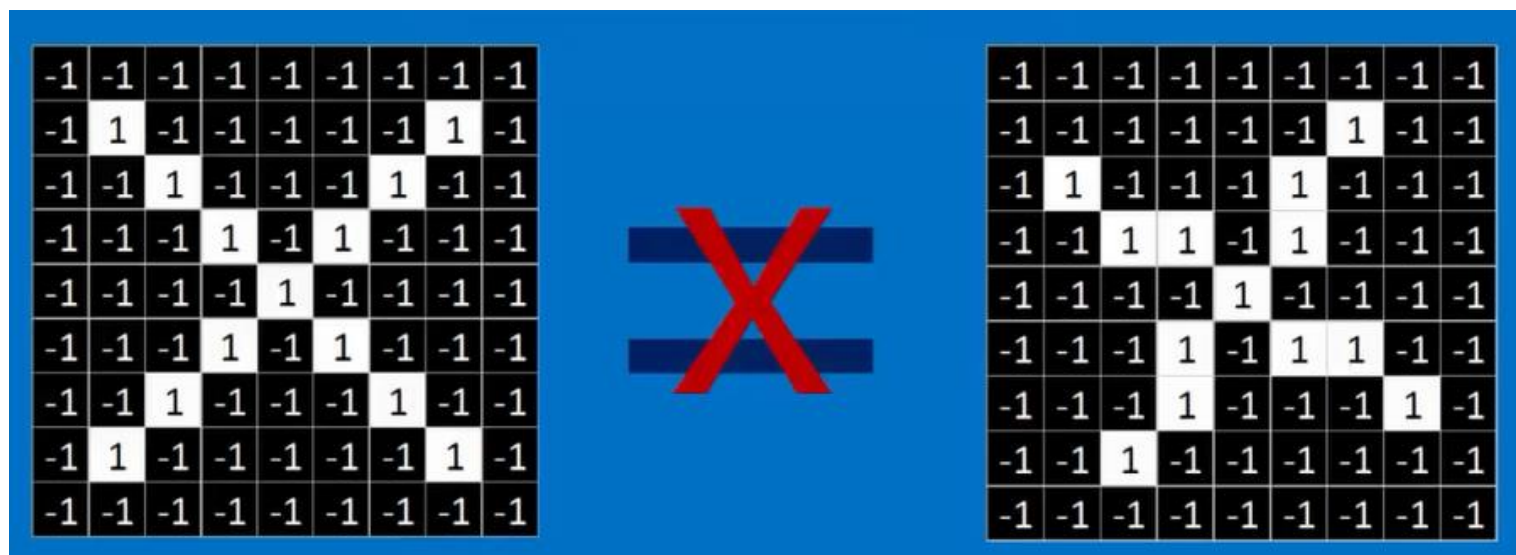
?

=

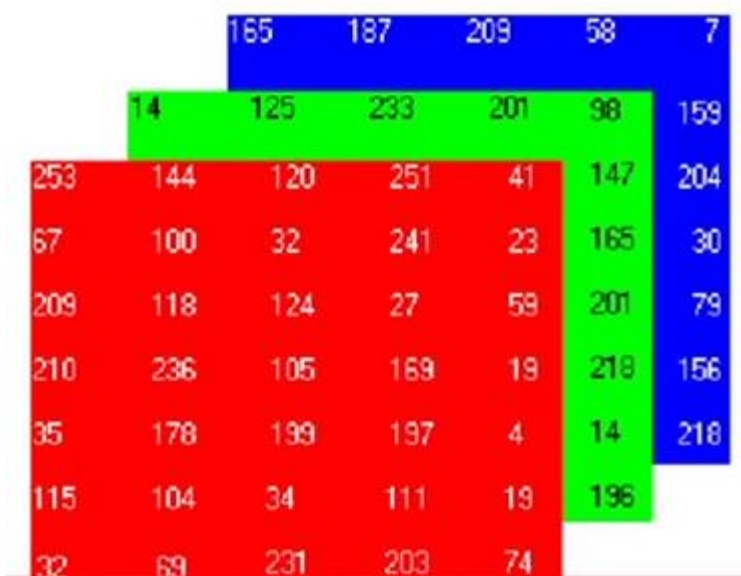
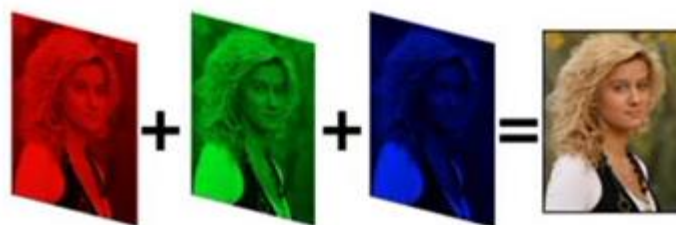
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	1	-1	-1
-1	-1	-1	1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

卷积神经网络处理流程

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	X	-1	-1	-1	-1	X	X	-1
-1	X	X	-1	-1	X	X	-1	-1
-1	-1	X	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	X	-1	-1
-1	-1	X	X	-1	-1	X	X	-1
-1	X	X	-1	-1	-1	-1	X	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



卷积神经网络处理流程



卷积神经网络处理流程

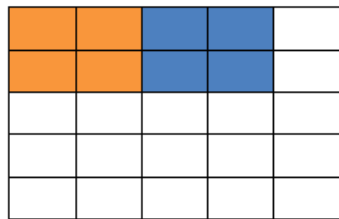
- （2）卷积计算层：这一层就是卷积神经网络最重要的一个层次，也是“卷积神经网络”的名字来源。在这个卷积层，有两个关键操作：

局部关联：每个神经元看做一个滤波器（filter）

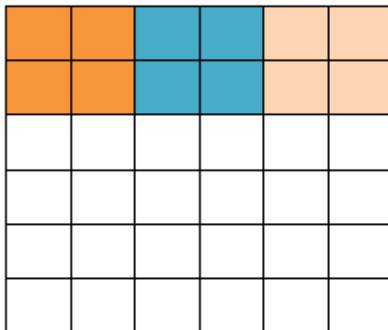
窗口（receptive field）滑动：filter对局部数据计算

卷积神经网络处理流程

- 填充值是什么呢？以下图为例，比如有这么一个 5×5 的图片（一个格子一个像素），我们滑动窗口取 2×2 ，步长取2，那么我们发现还剩下1个像素没法滑完，那怎么办呢？



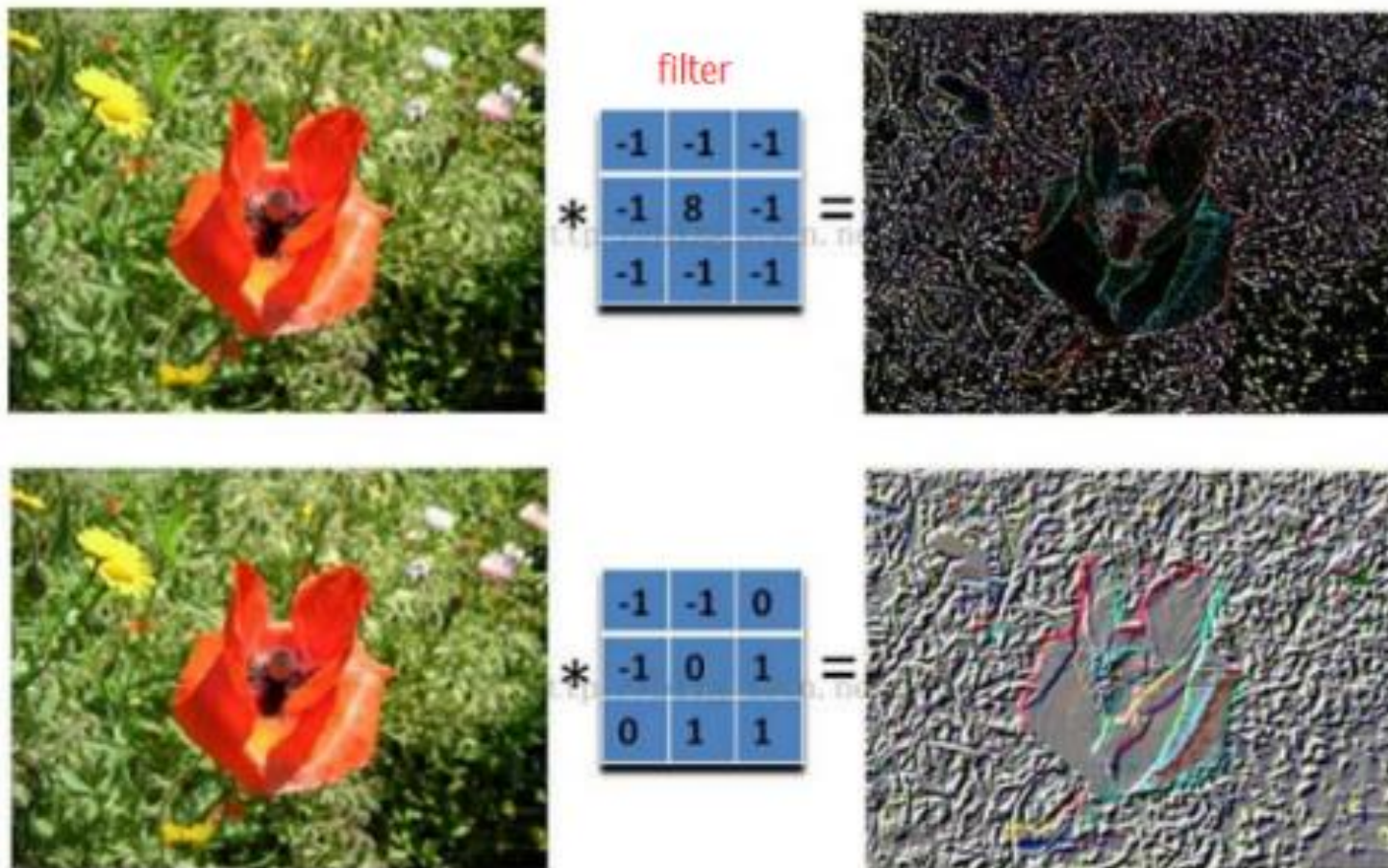
- 那我们在原先的矩阵加了一层填充值，使得变成 6×6 的矩阵，那么窗口就可以刚好把所有像素遍历完。这就是填充值的作用。



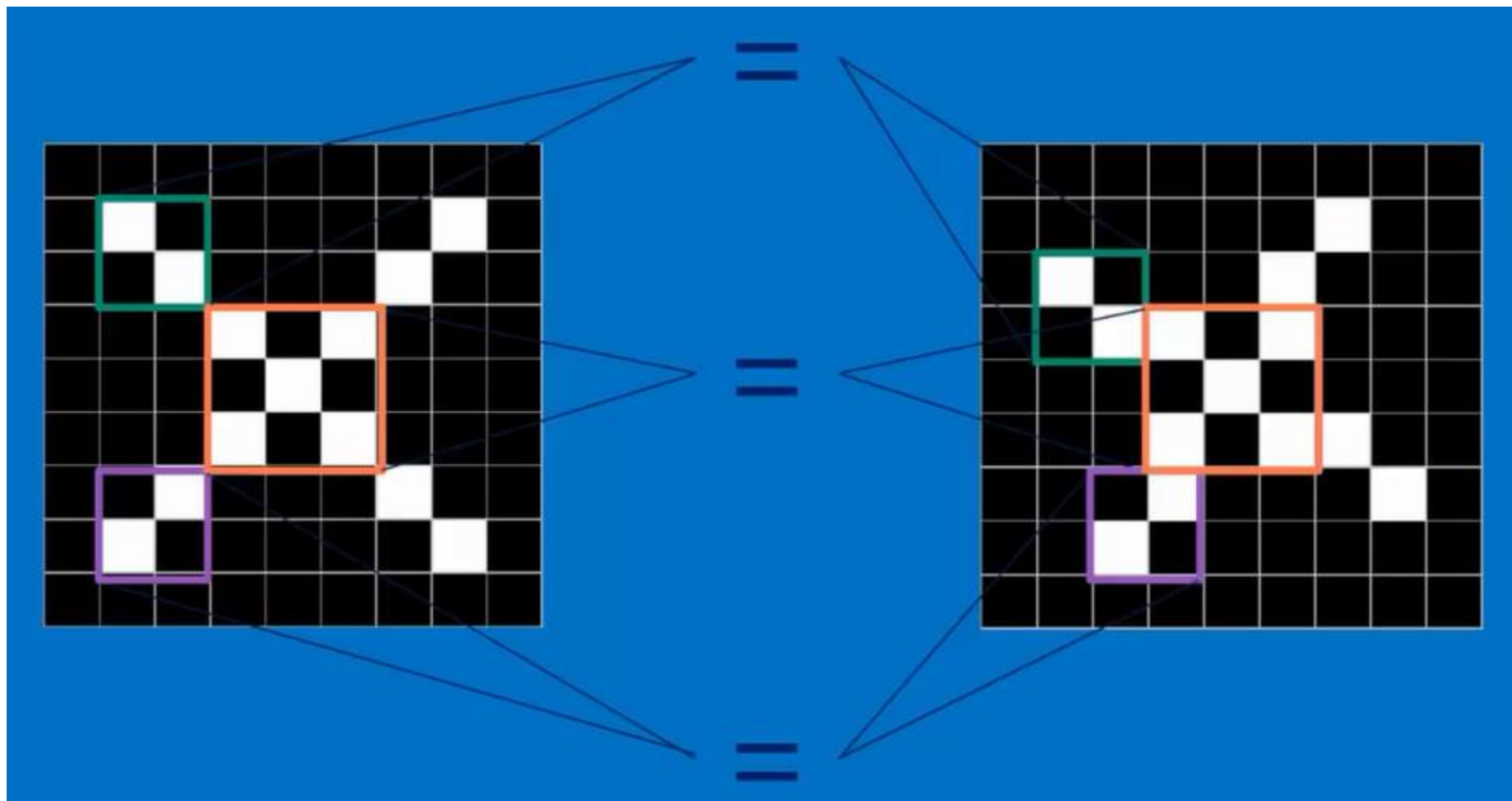
卷积神经网络处理流程

- 参数共享机制：在卷积层中每个神经元连接数据窗的权重是固定的，每个神经元只关注一个特性。神经元就是图像处理中的滤波器，比如边缘检测专用的Sobel滤波器，即卷积层的每个滤波器都会有自己所关注一个图像特征，比如垂直边缘，水平边缘，颜色，纹理等等，这些所有神经元加起来就好比就是整张图像的特征提取器集合。

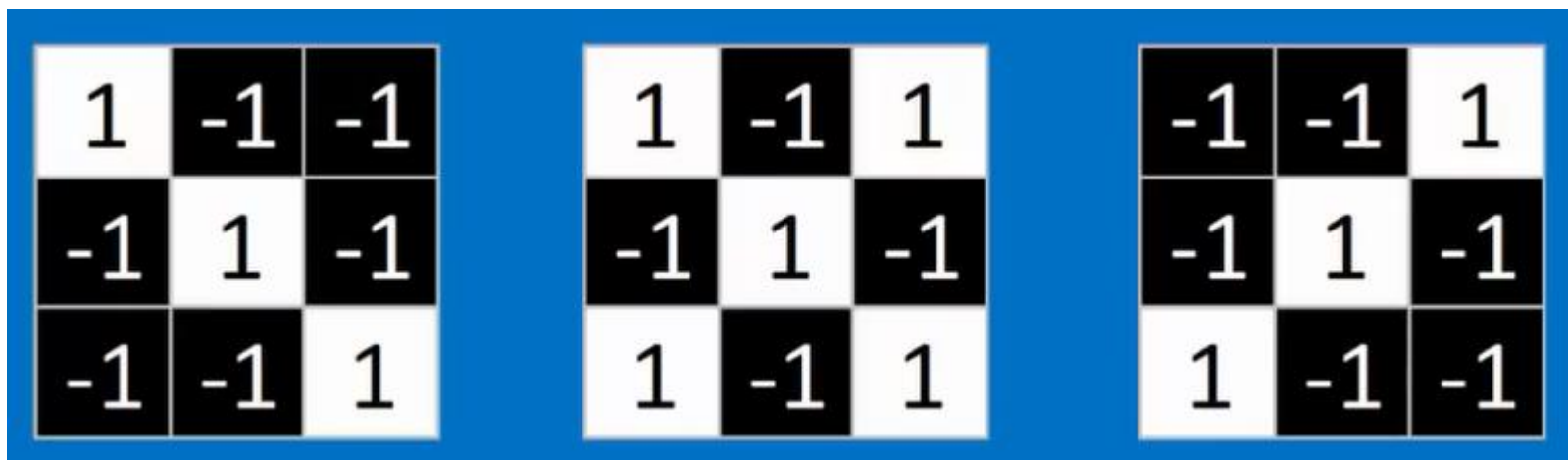
卷积神经网络处理流程



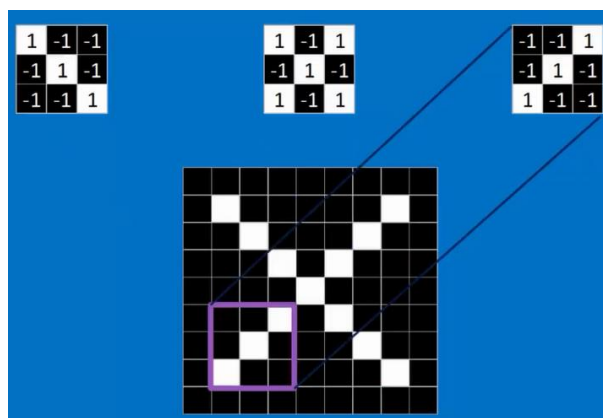
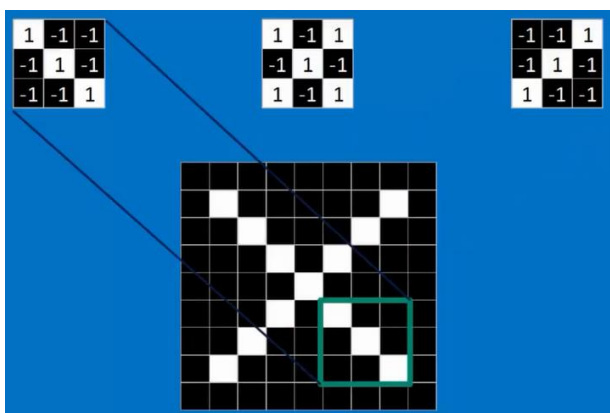
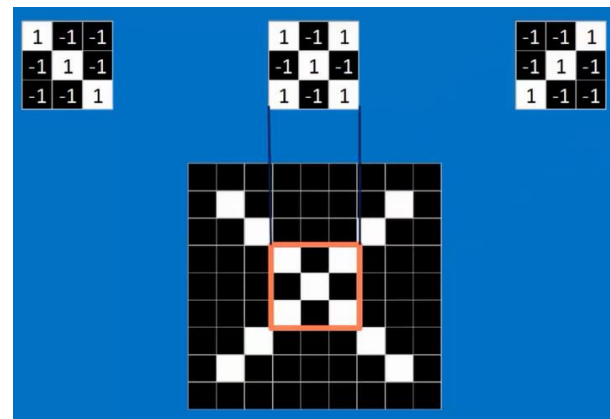
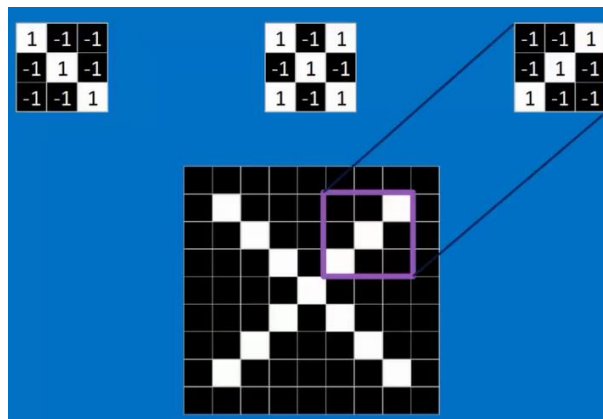
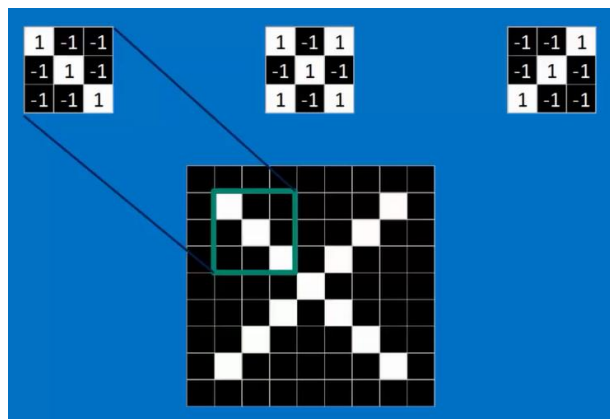
卷积神经网络处理流程



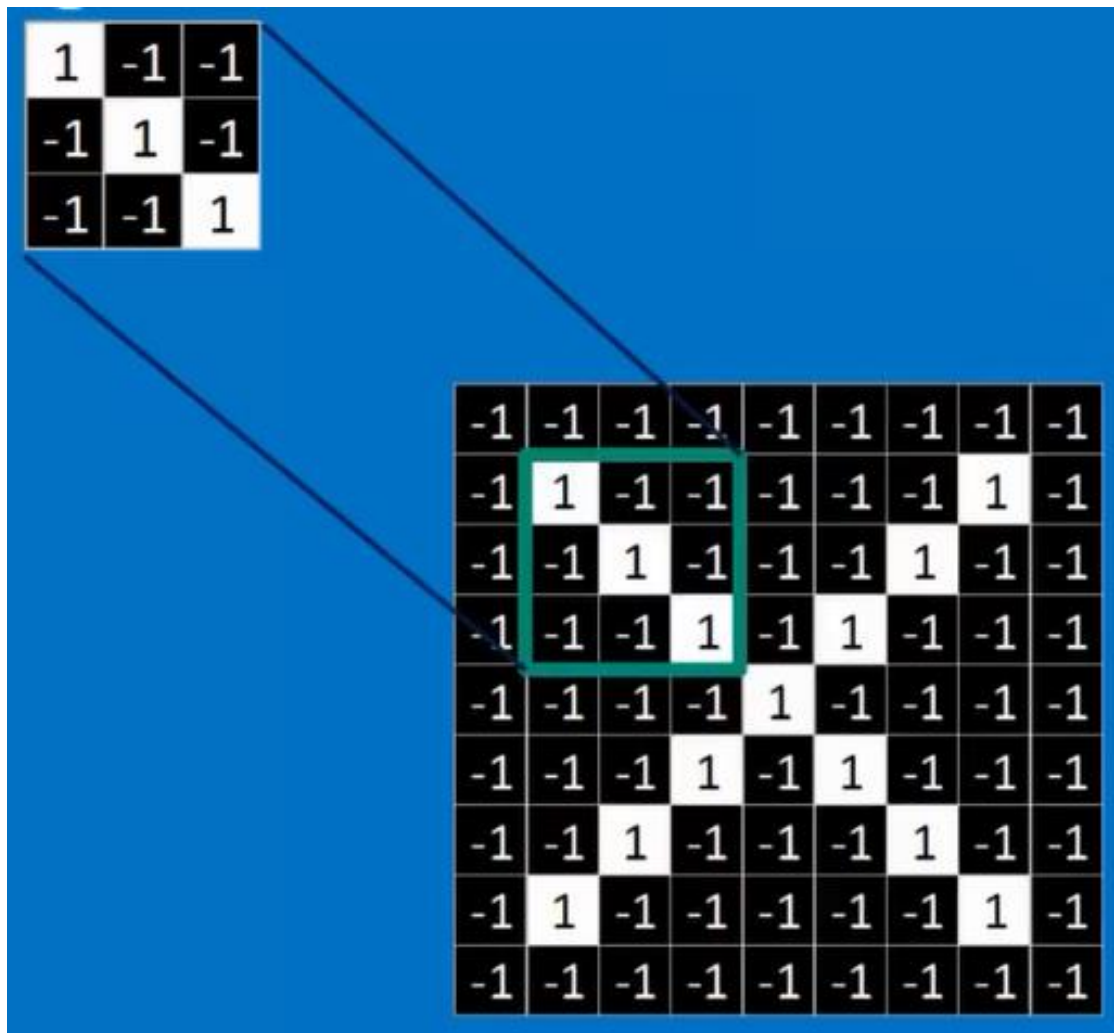
卷积神经网络处理流程



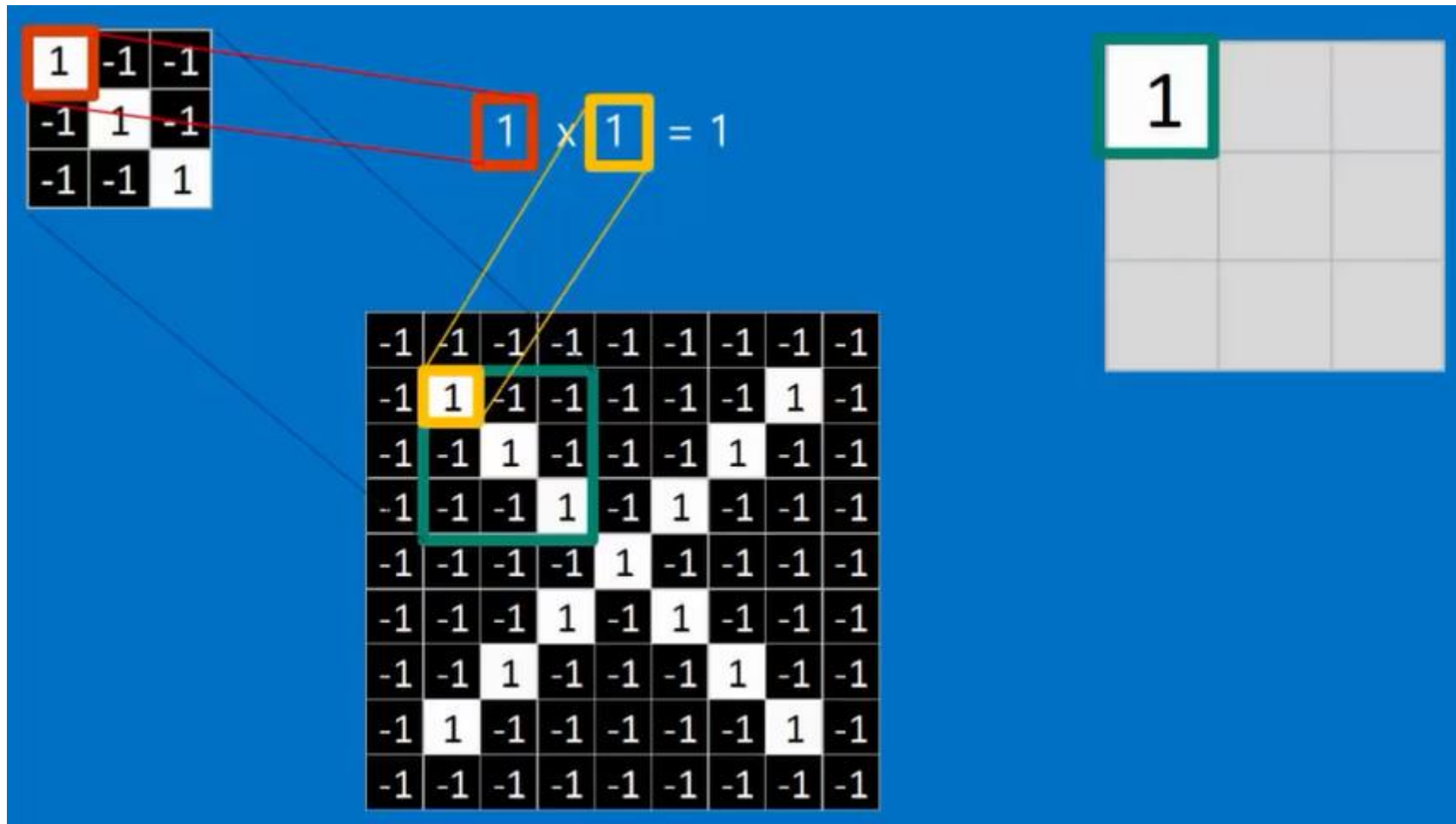
卷积神经网络处理流程



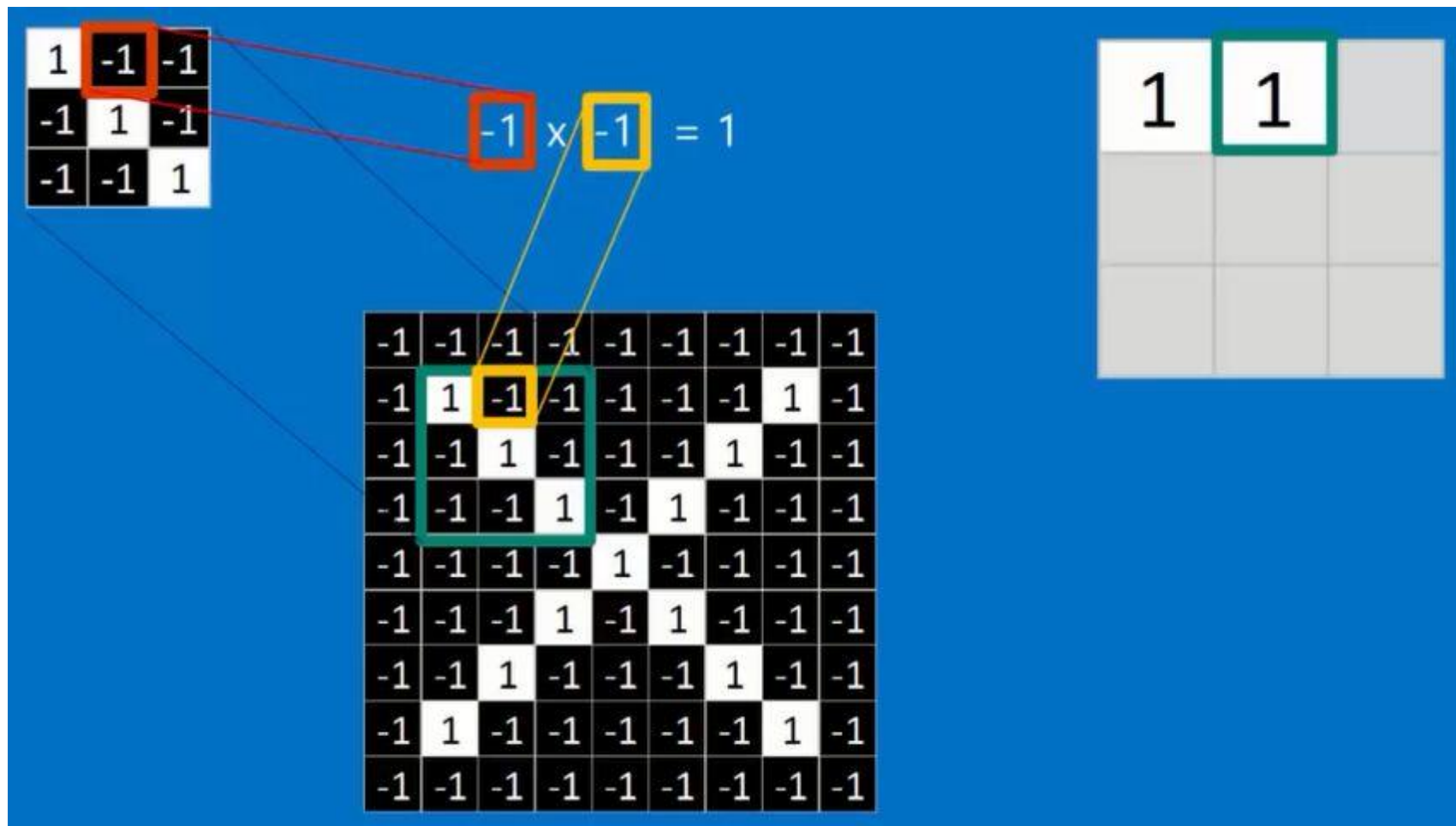
卷积神经网络处理流程



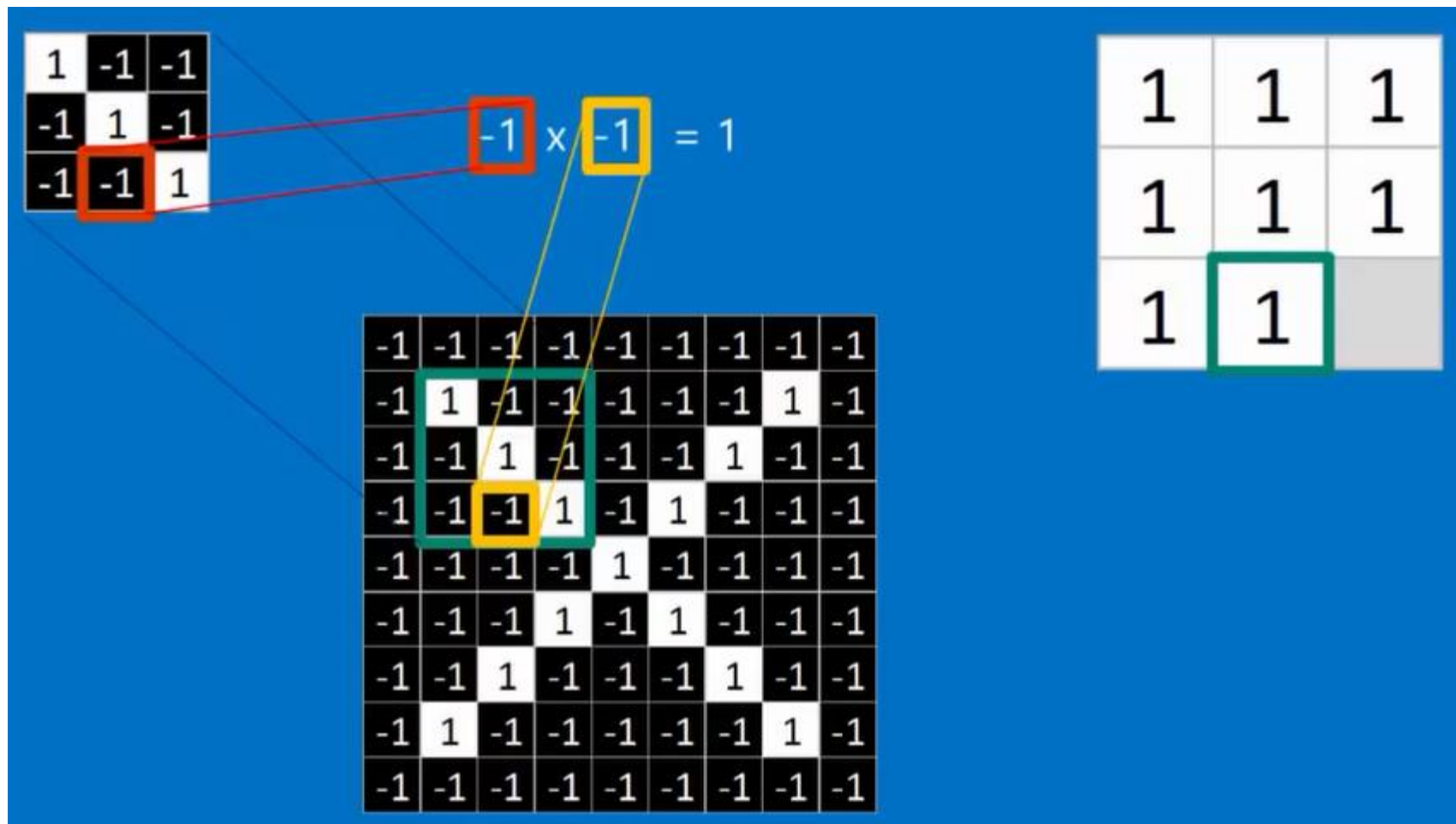
卷积神经网络处理流程



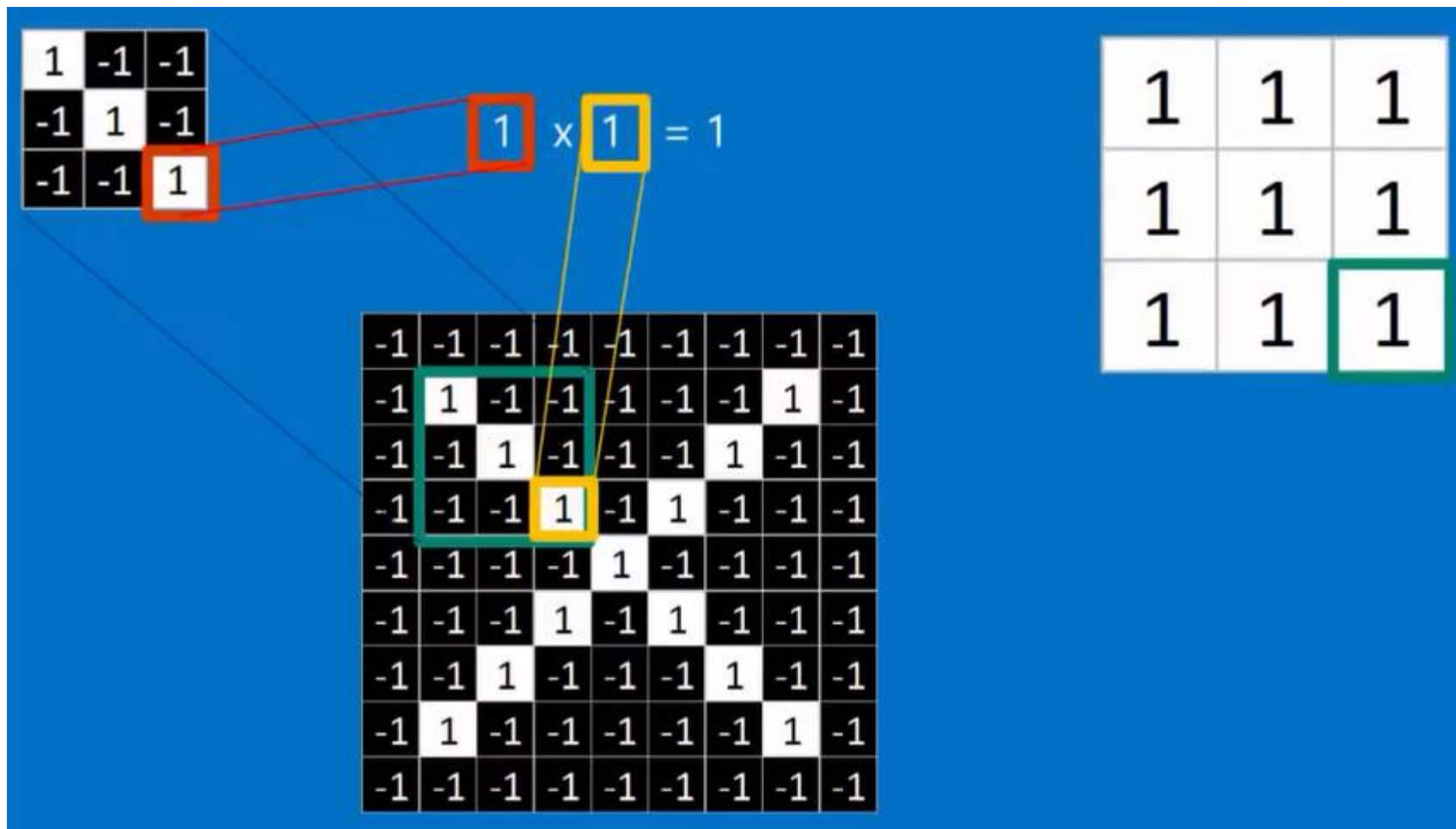
卷积神经网络处理流程



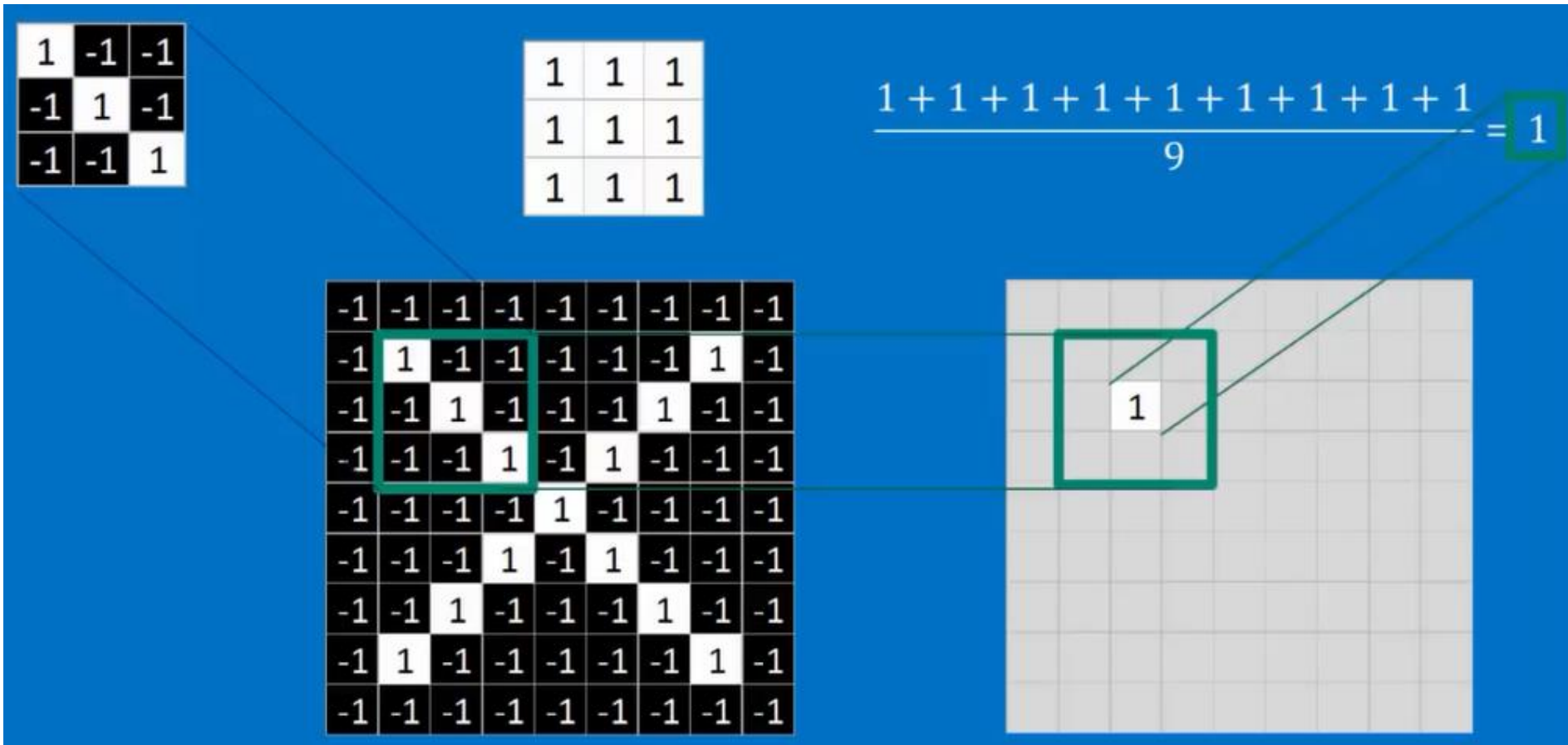
卷积神经网络处理流程



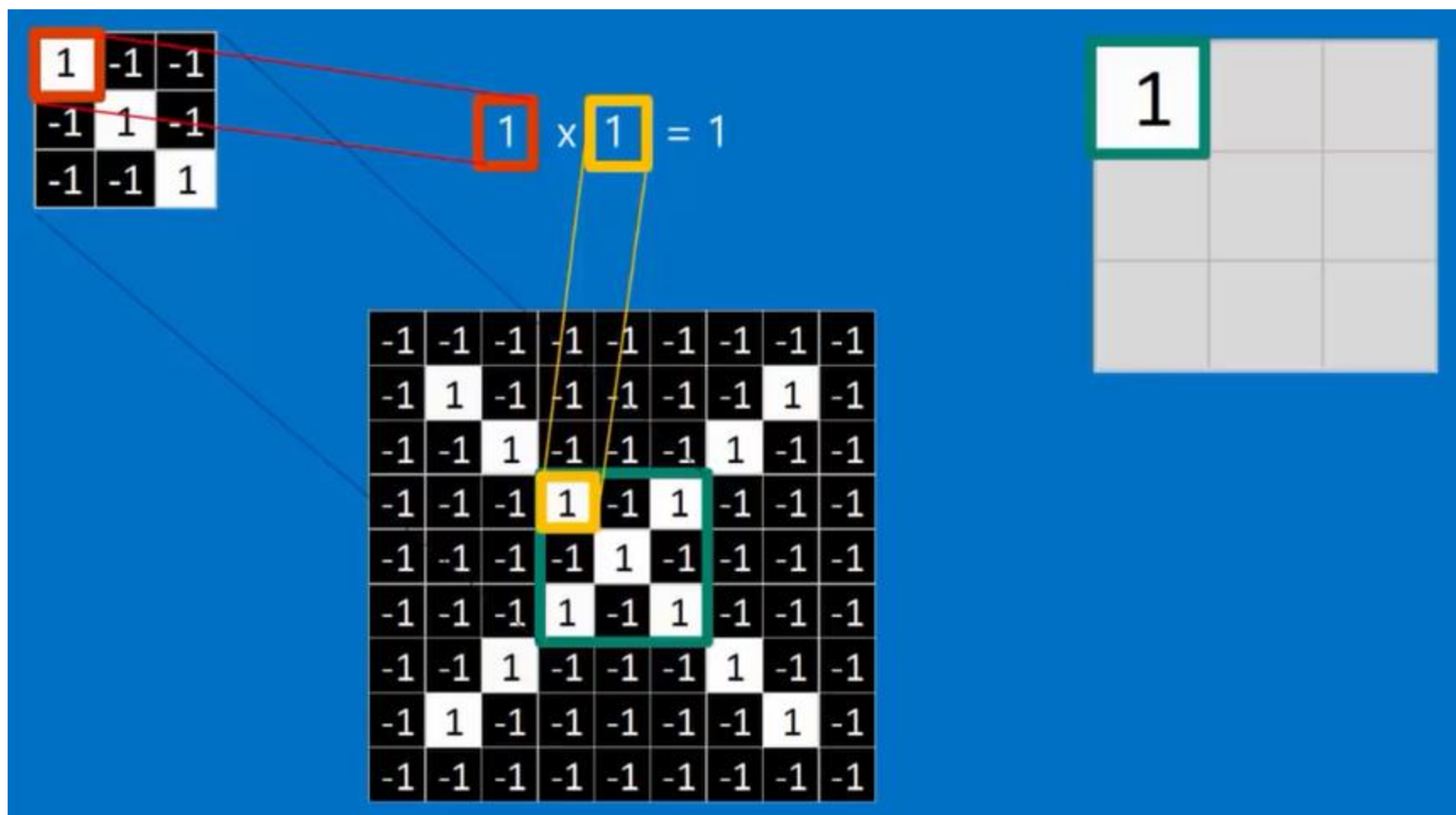
卷积神经网络处理流程



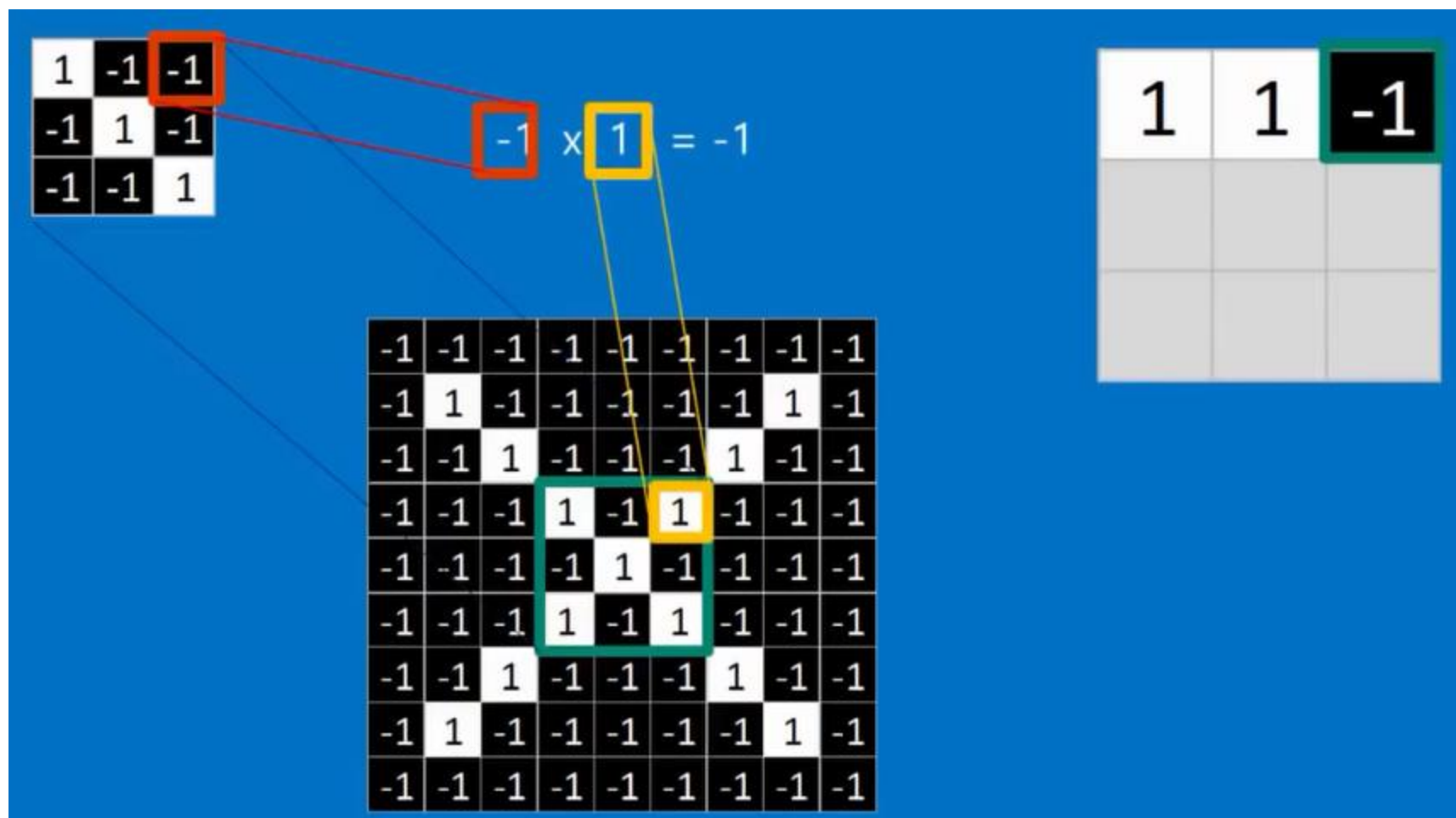
卷积神经网络处理流程



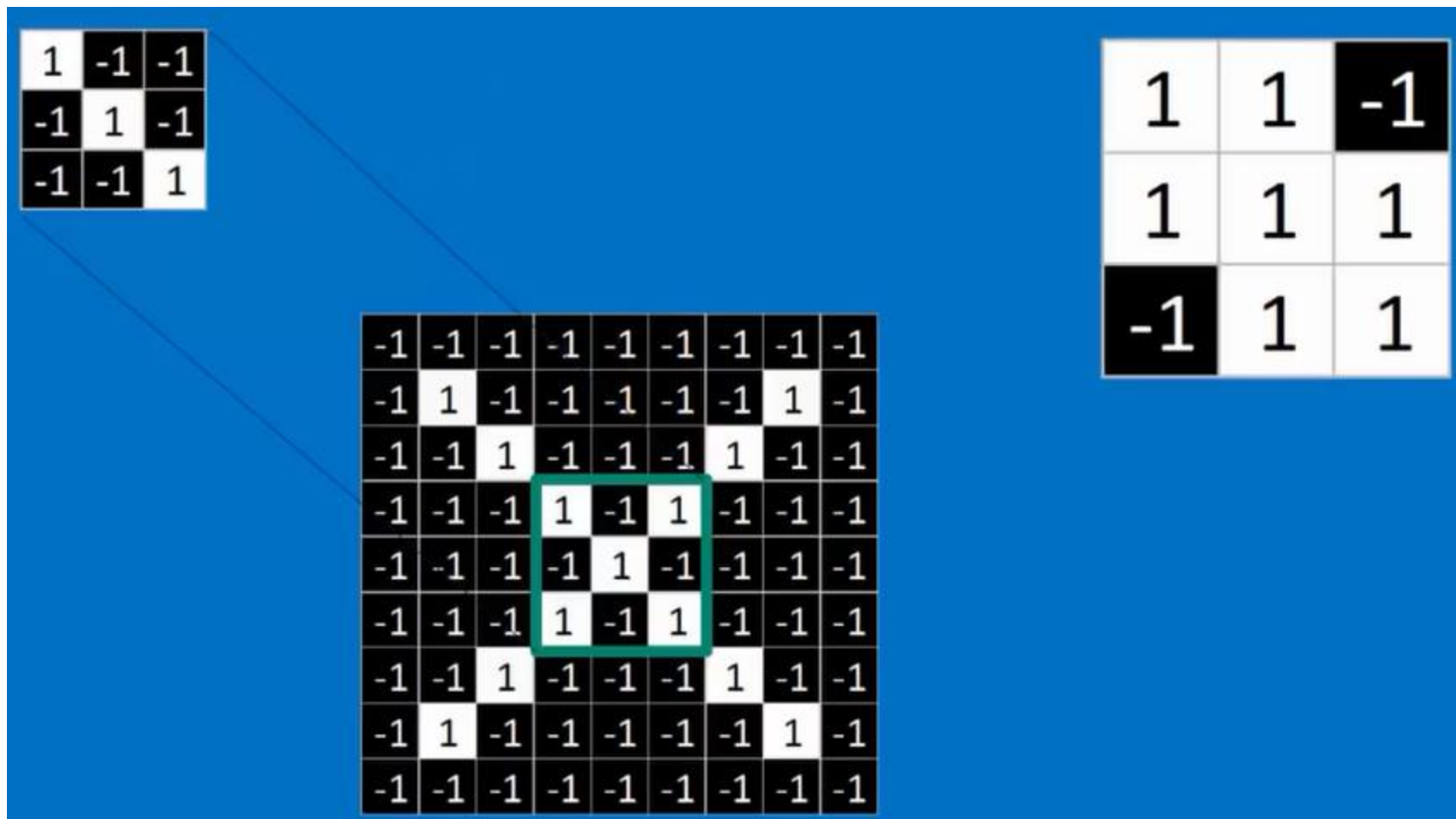
卷积神经网络处理流程



卷积神经网络处理流程



卷积神经网络处理流程



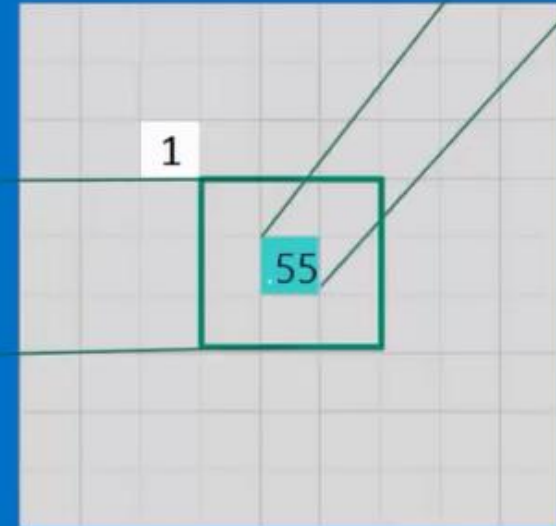
卷积神经网络处理流程

1	-1	-1
-1	1	-1
-1	-1	1

1	1	-1
1	1	1
-1	1	1

$$\frac{1 + 1 - 1 + 1 + 1 + 1 - 1 + 1 + 1}{9} = .55$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



卷积神经网络处理流程

1	-1	-1
-1	1	-1
-1	-1	1

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

卷积神经网络处理流程

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

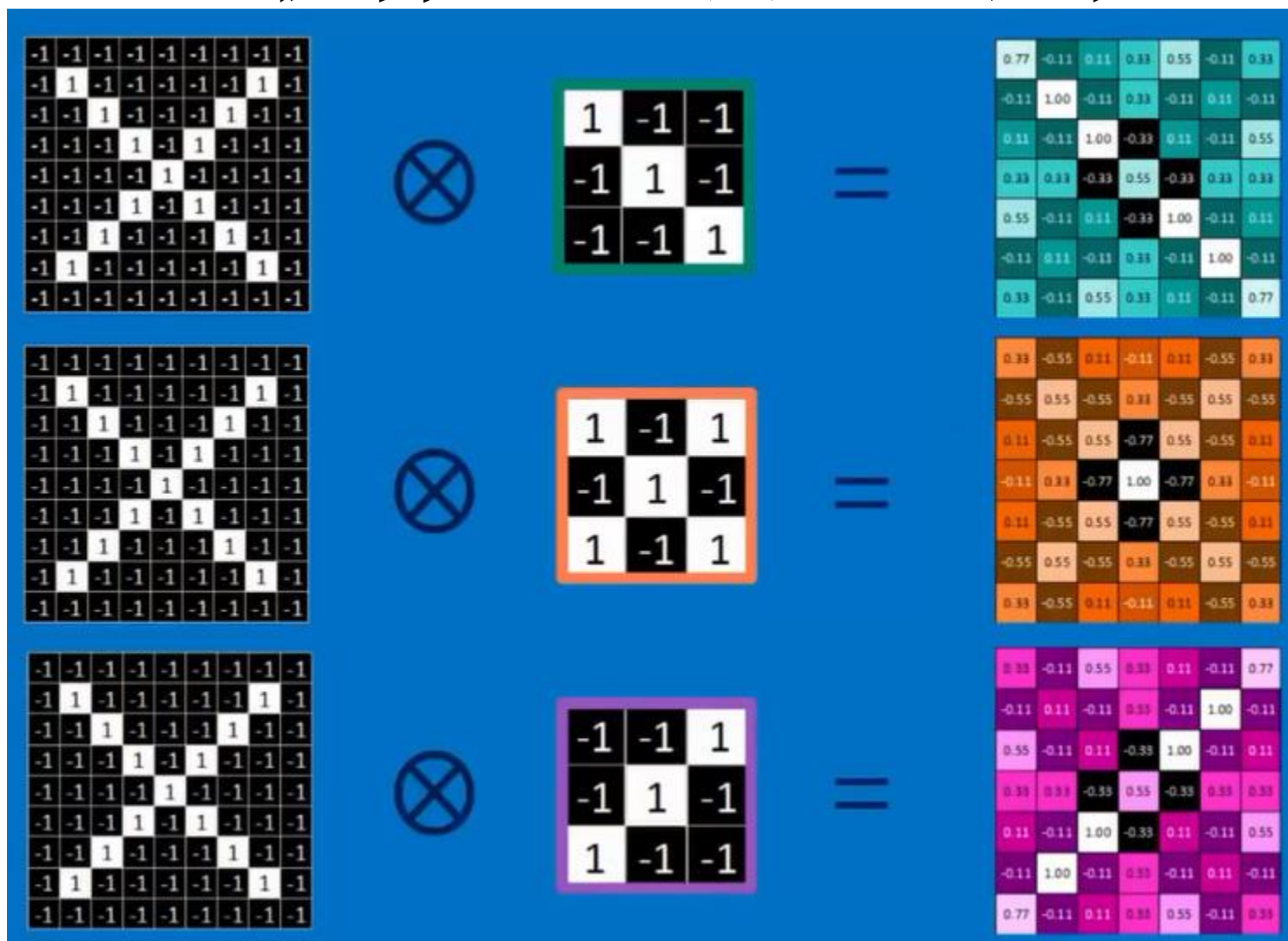


1	-1	-1
-1	1	-1
-1	-1	1

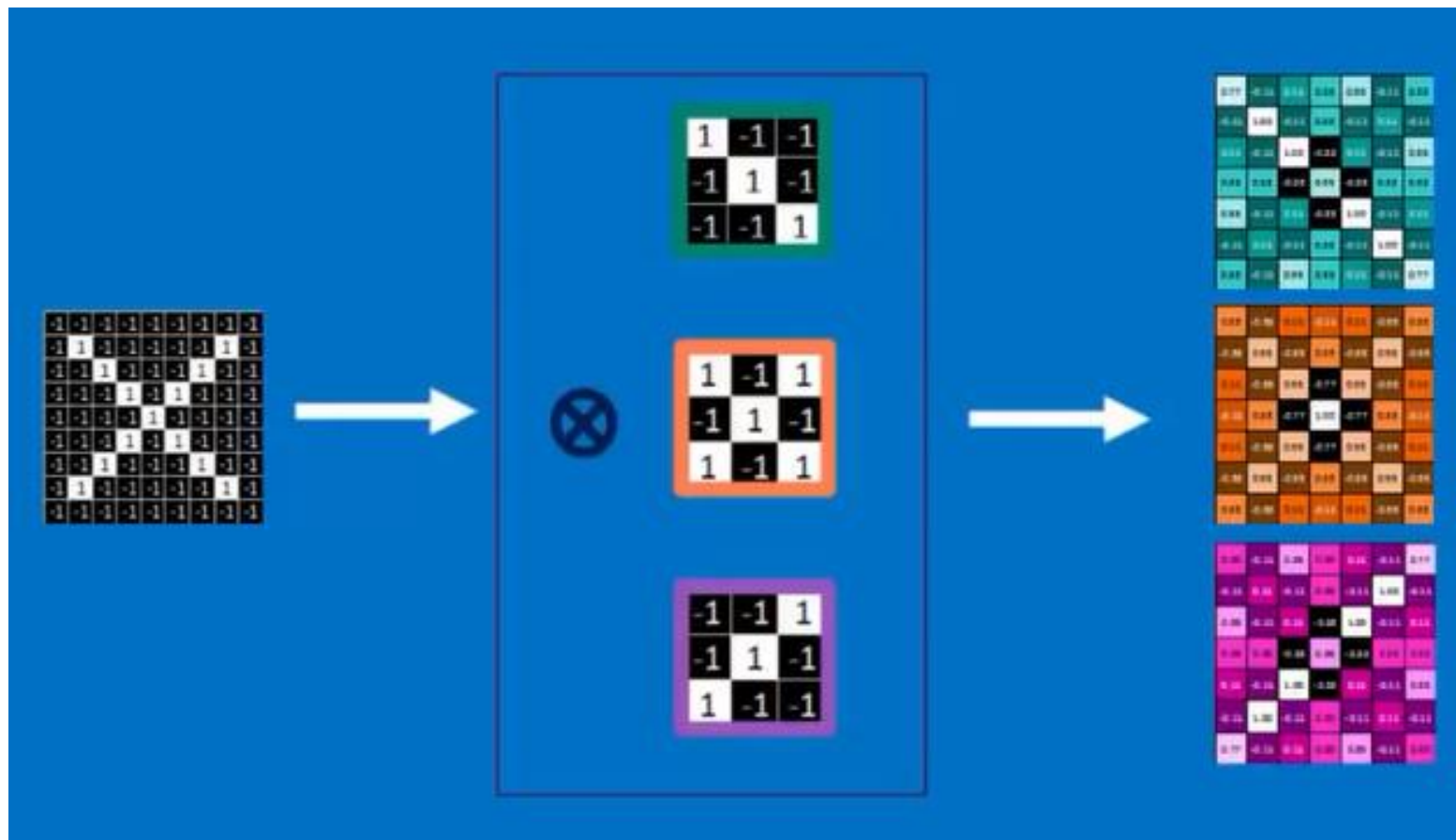
=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

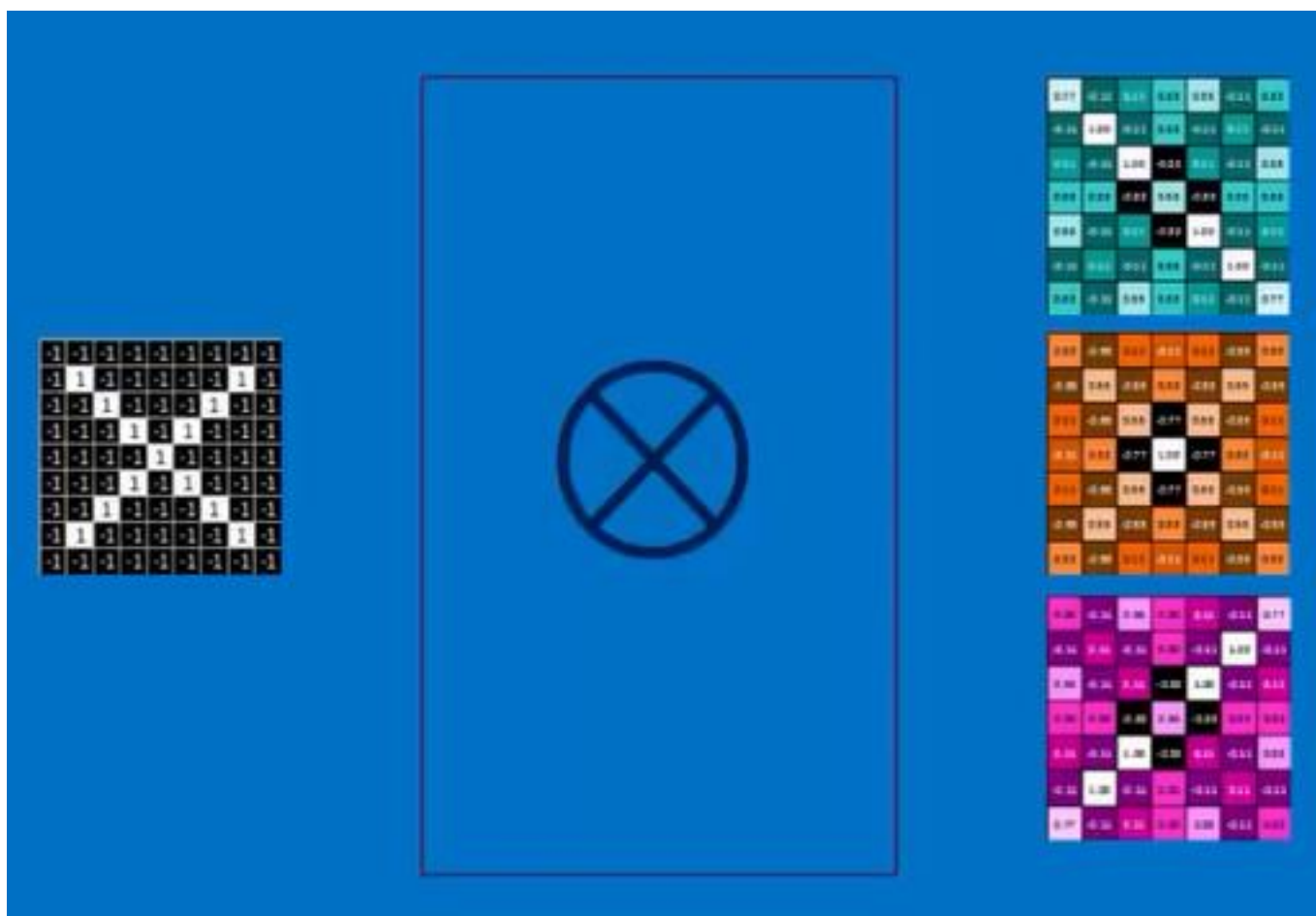
卷积神经网络处理流程



卷积神经网络处理流程

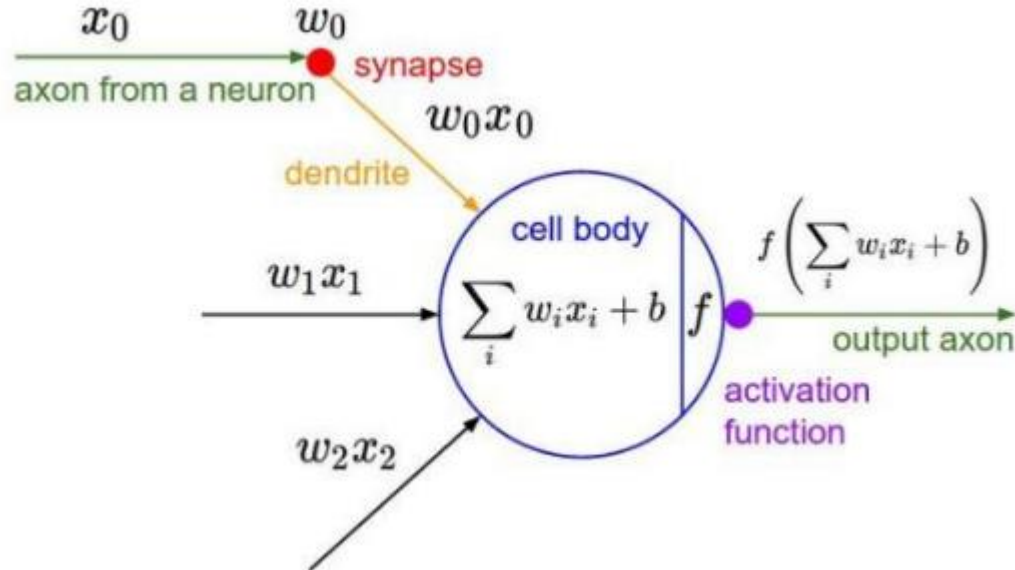


卷积神经网络处理流程



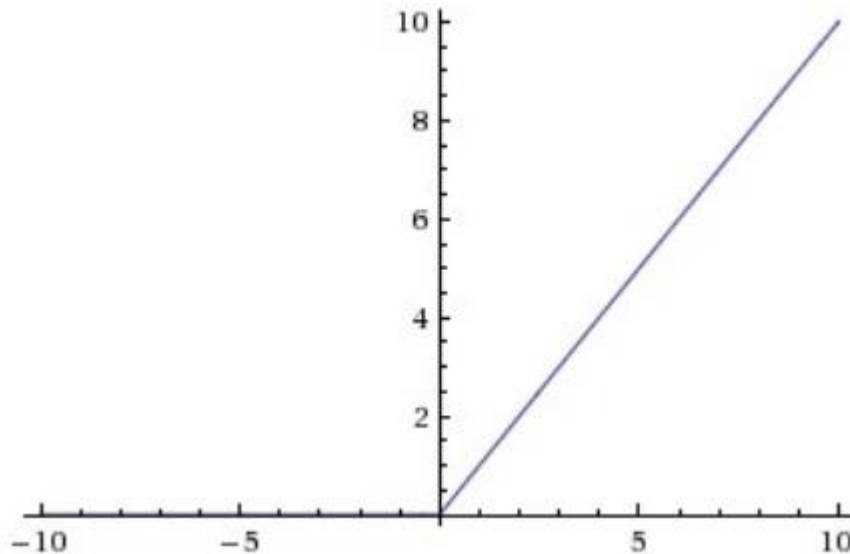
卷积神经网络处理流程

- （3）激励层：把卷积层输出结果做非线性映射。



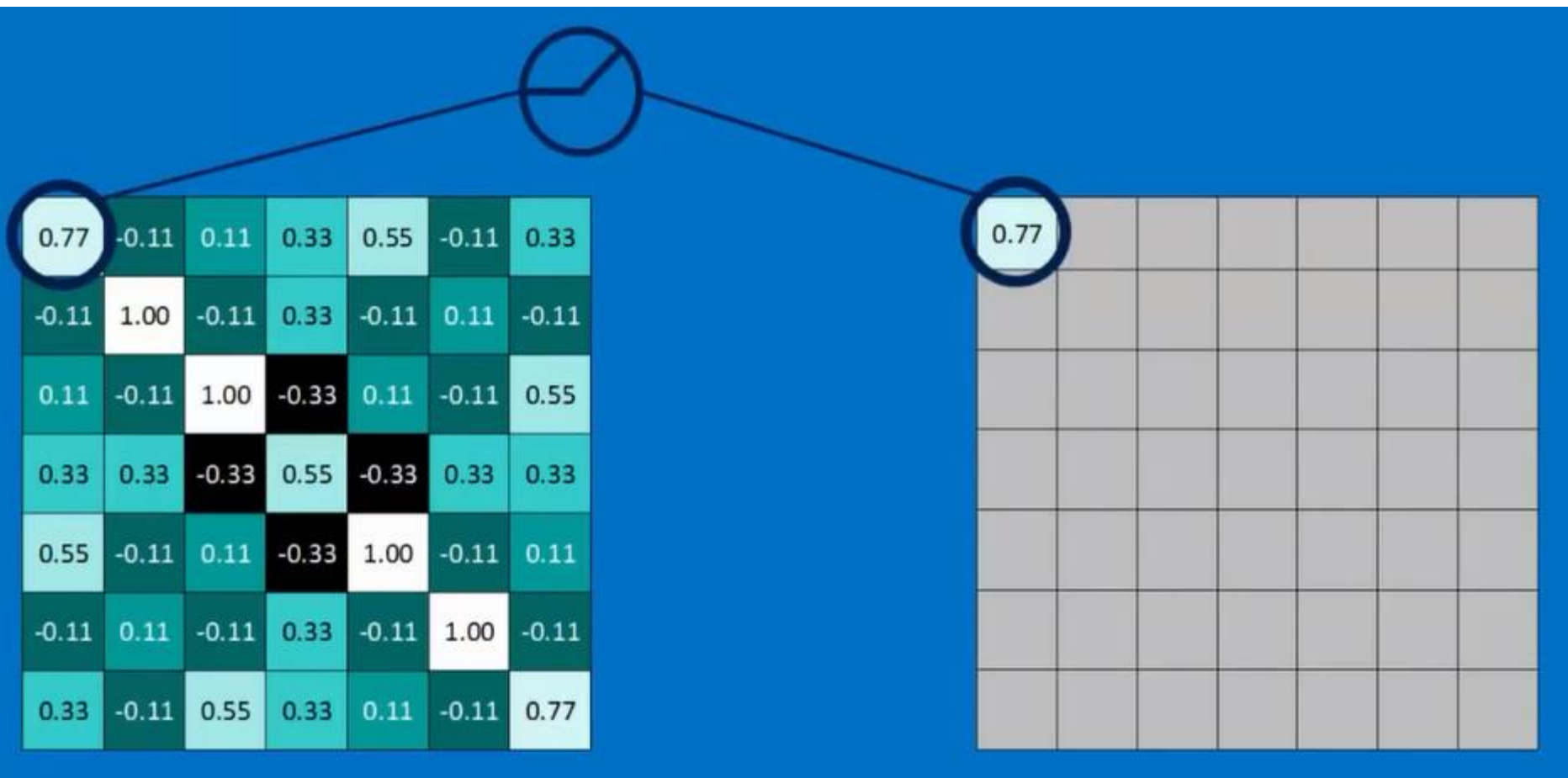
卷积神经网络处理流程

- CNN采用的激励函数一般为ReLU（The Rectified Linear Unit/修正线性单元），它的特点是收敛快，求梯度简单，但较脆弱，图像如下：

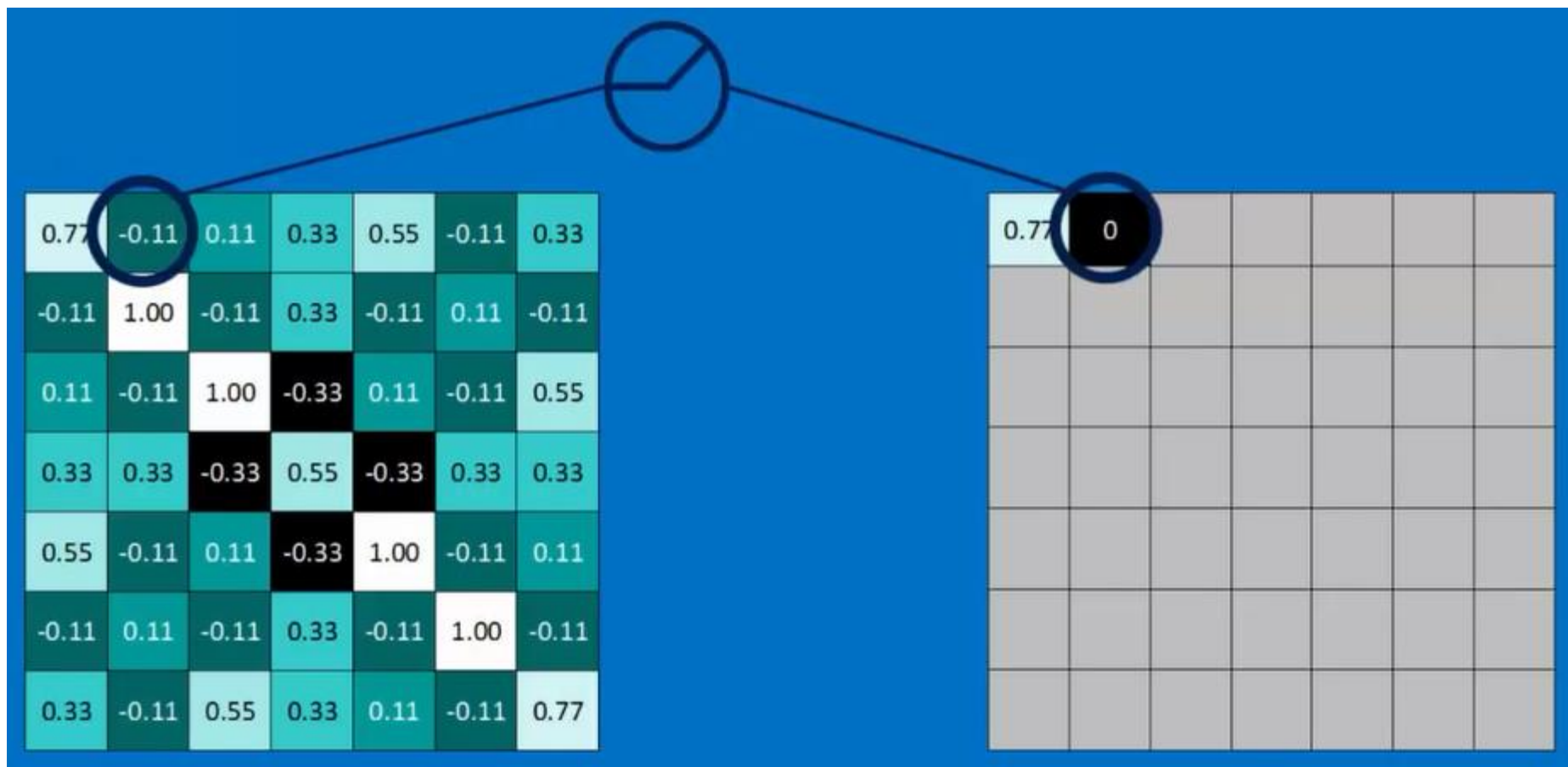


- 注意：激励层没有用sigmoid函数！

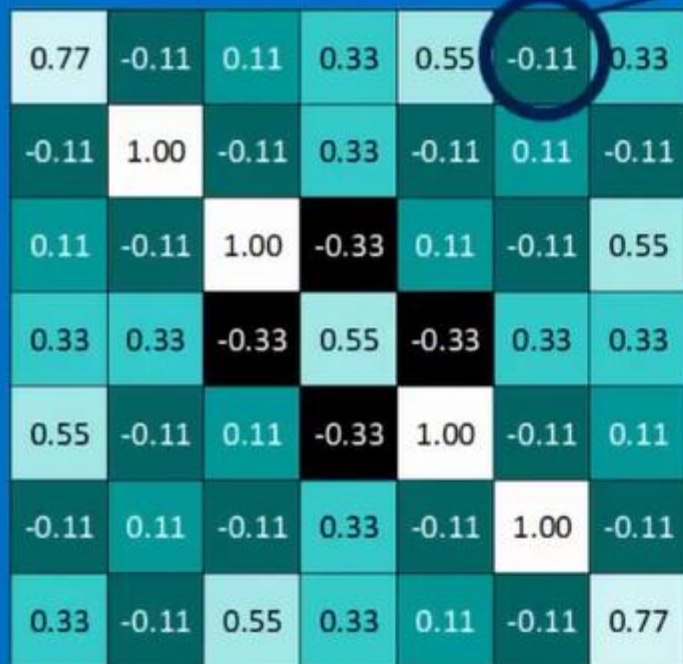
卷积神经网络处理流程



卷积神经网络处理流程

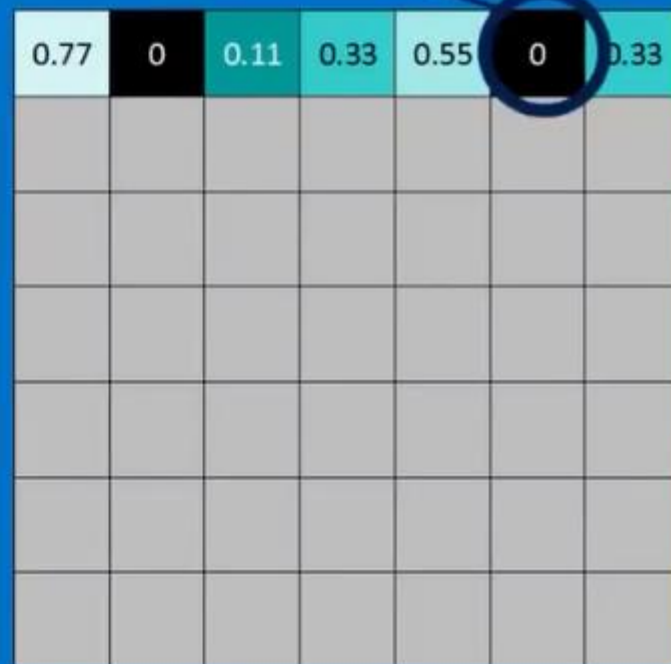


卷积神经网络处理流程



A 7x7 convolution kernel matrix is shown. The top row is highlighted in light blue. A 3x3 region in the top-right corner of this row is circled in dark blue. The values in the circled region are -0.11, 0.33, and 0.55.

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77



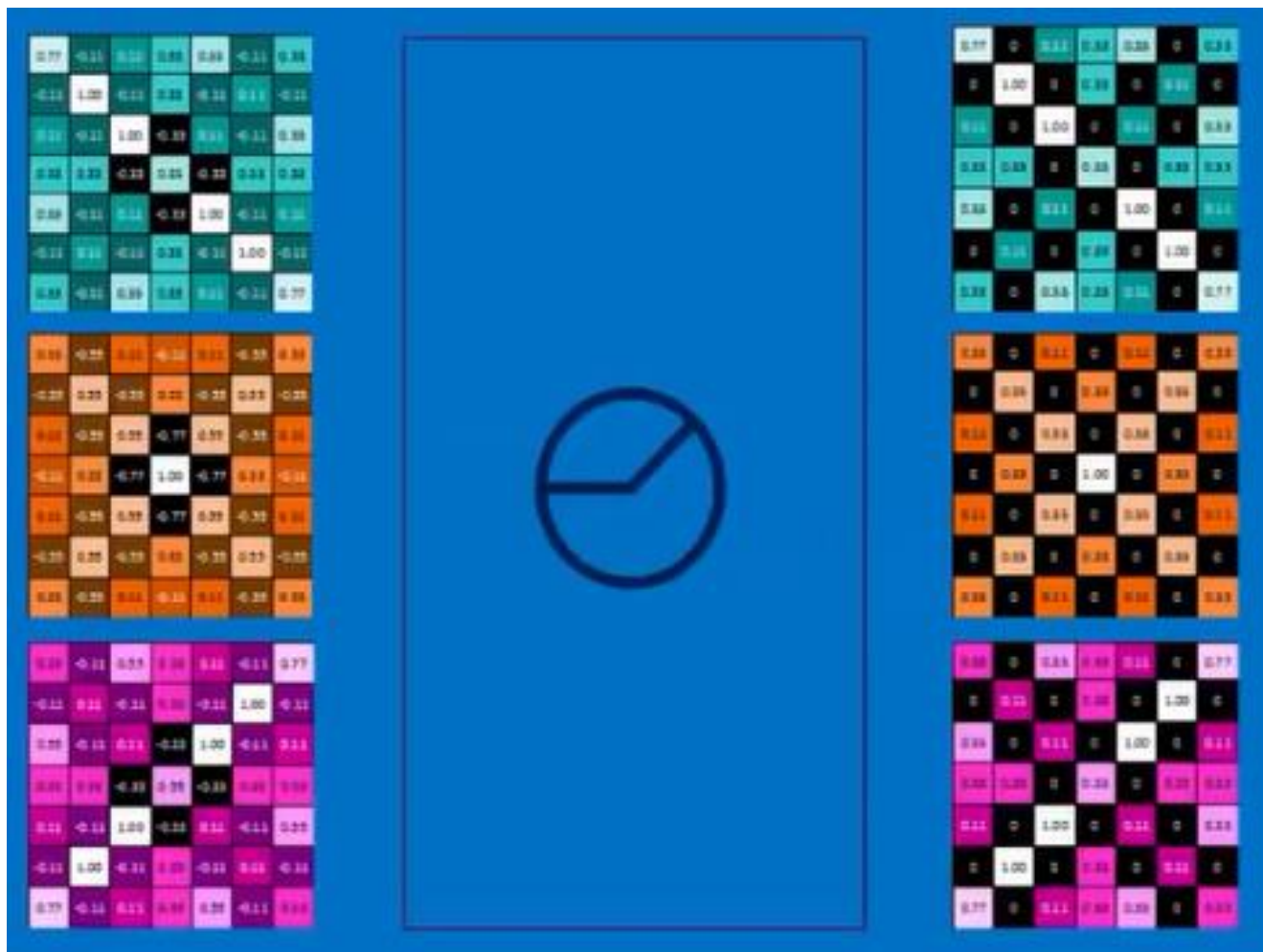
A 7x7 convolution kernel matrix is shown. The top row is highlighted in light blue. A 3x3 region in the top-right corner of this row is circled in dark blue. The values in the circled region are 0, 0.33, and 0.55.

0.77	0	0.11	0.33	0.55	0	0.33

卷积神经网络处理流程



卷积神经网络处理流程



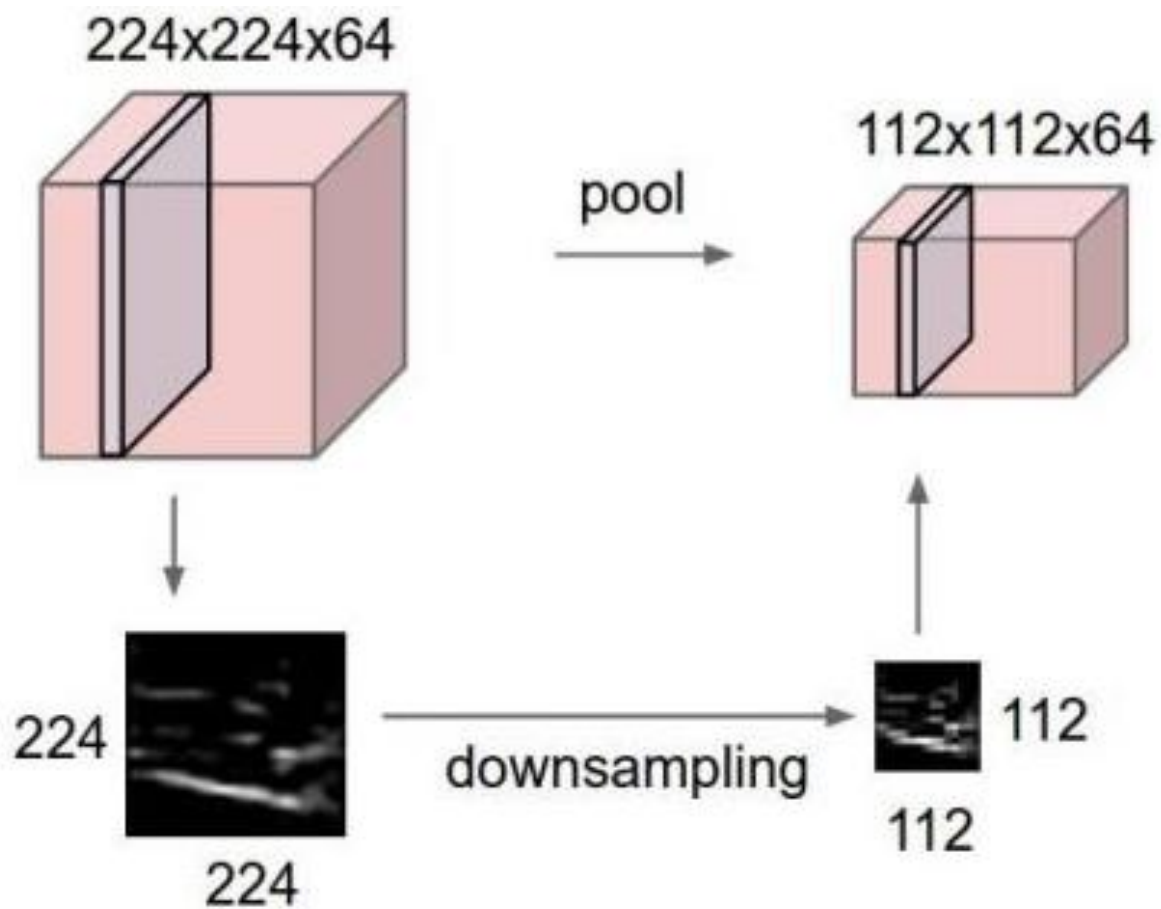
卷积神经网络处理流程

- （4）池化层：池化层夹在连续的卷积层中间，用于压缩数据和参数的量，减小过拟合。简而言之，如果输入是图像的话，那么池化层的最主要作用就是压缩图像。

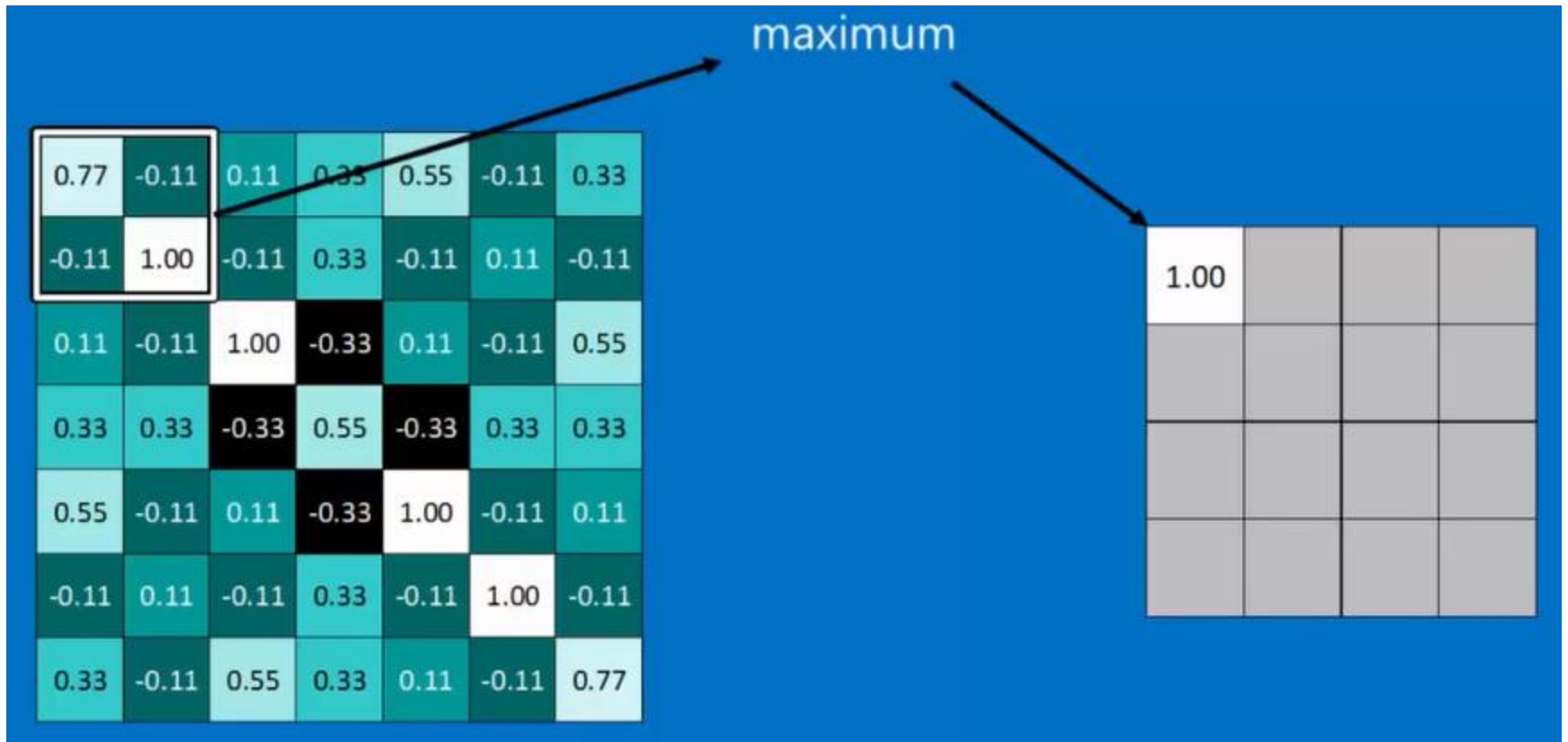
卷积神经网络处理流程

- 池化层的具体作用包括：
 - a. 特征不变性，即：图像特征的尺度不变性。池化操作就是图像尺寸压缩，平时一张狗的图像被缩小了一倍我们还能认出这是狗，说明这张图像中仍保留着狗最重要的特征，我们一看就能判断出来，图像压缩时去掉的信息只是一些无关紧要的信息，而留下的信息则是具有尺度不变性的特征，是最能表达图像的特征。
 - b. 特征降维，我们知道一幅图像含有的信息是很大的，特征也很多，但是有些信息对于我们做图像任务时没有太多用途或者有重复，我们可以把这类冗余信息去除，把最重要的特征抽取出来。
 - c. 在一定程度上防止过拟合，更方便优化。

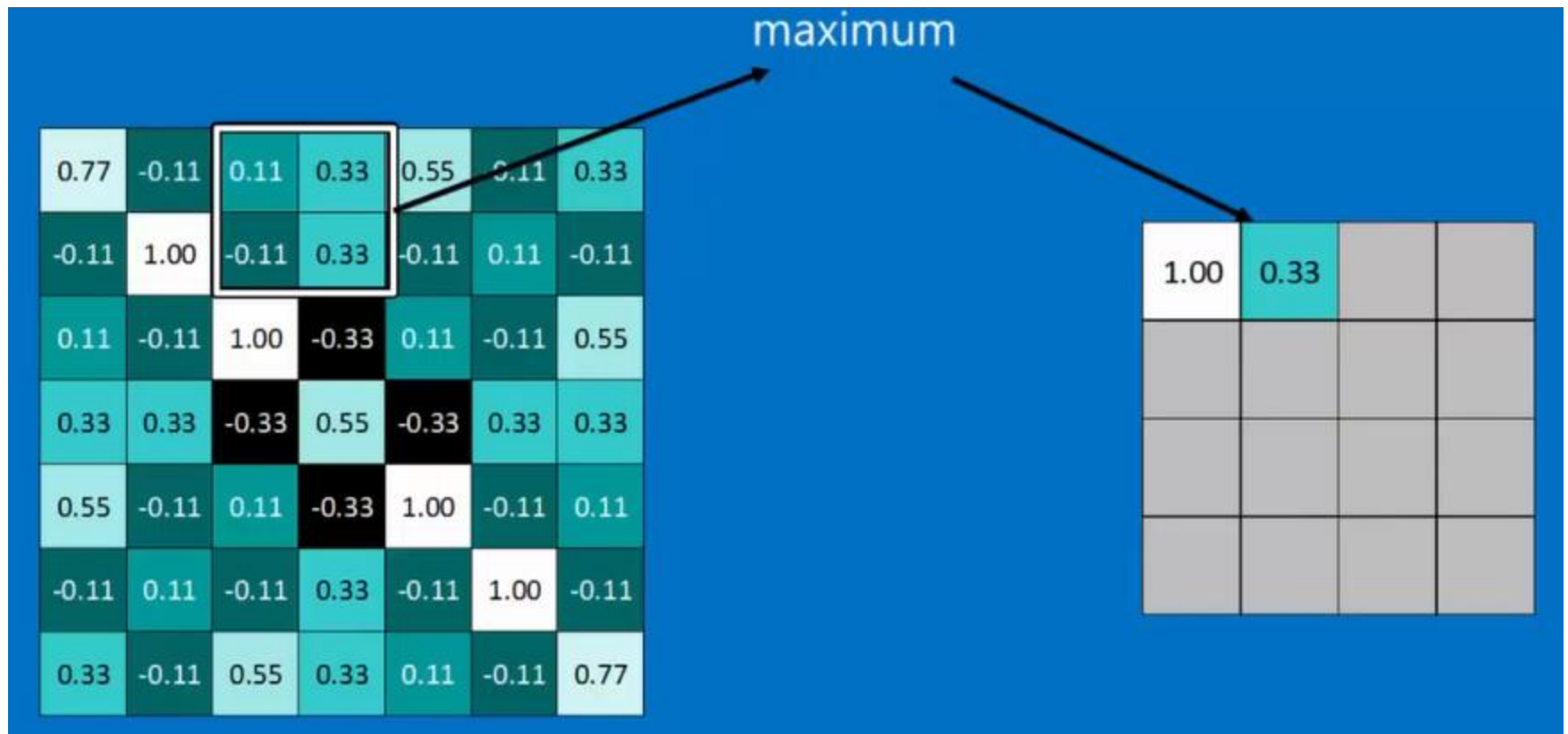
卷积神经网络处理流程



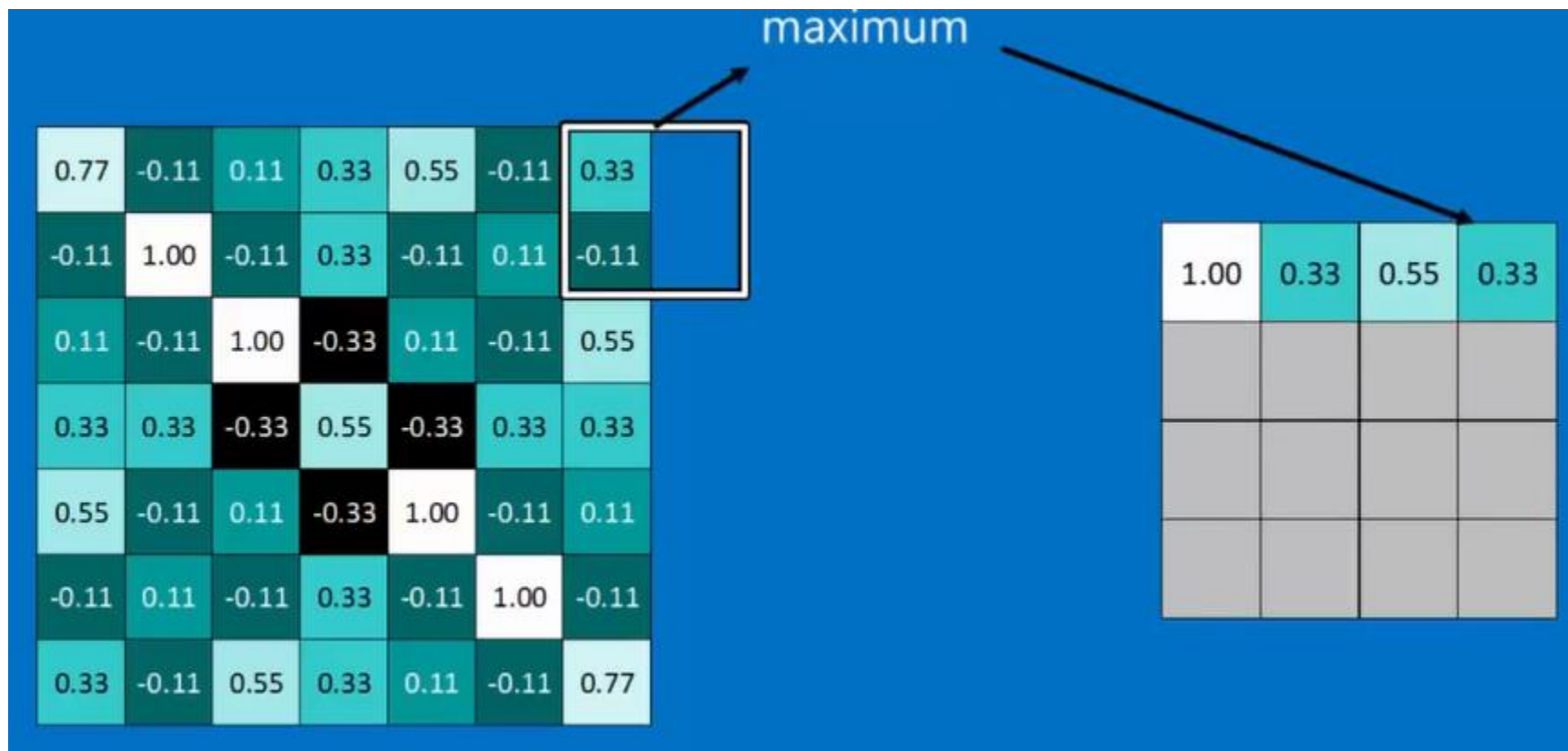
卷积神经网络处理流程



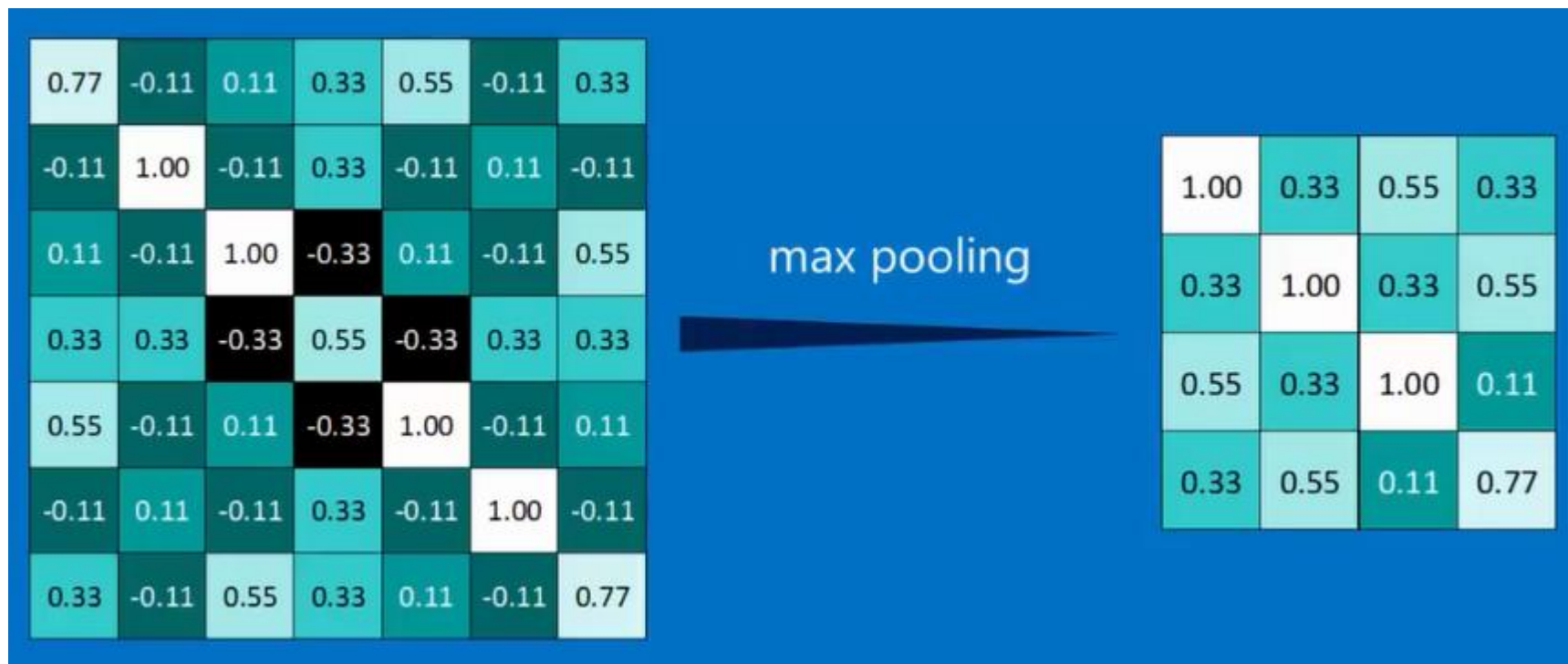
卷积神经网络处理流程



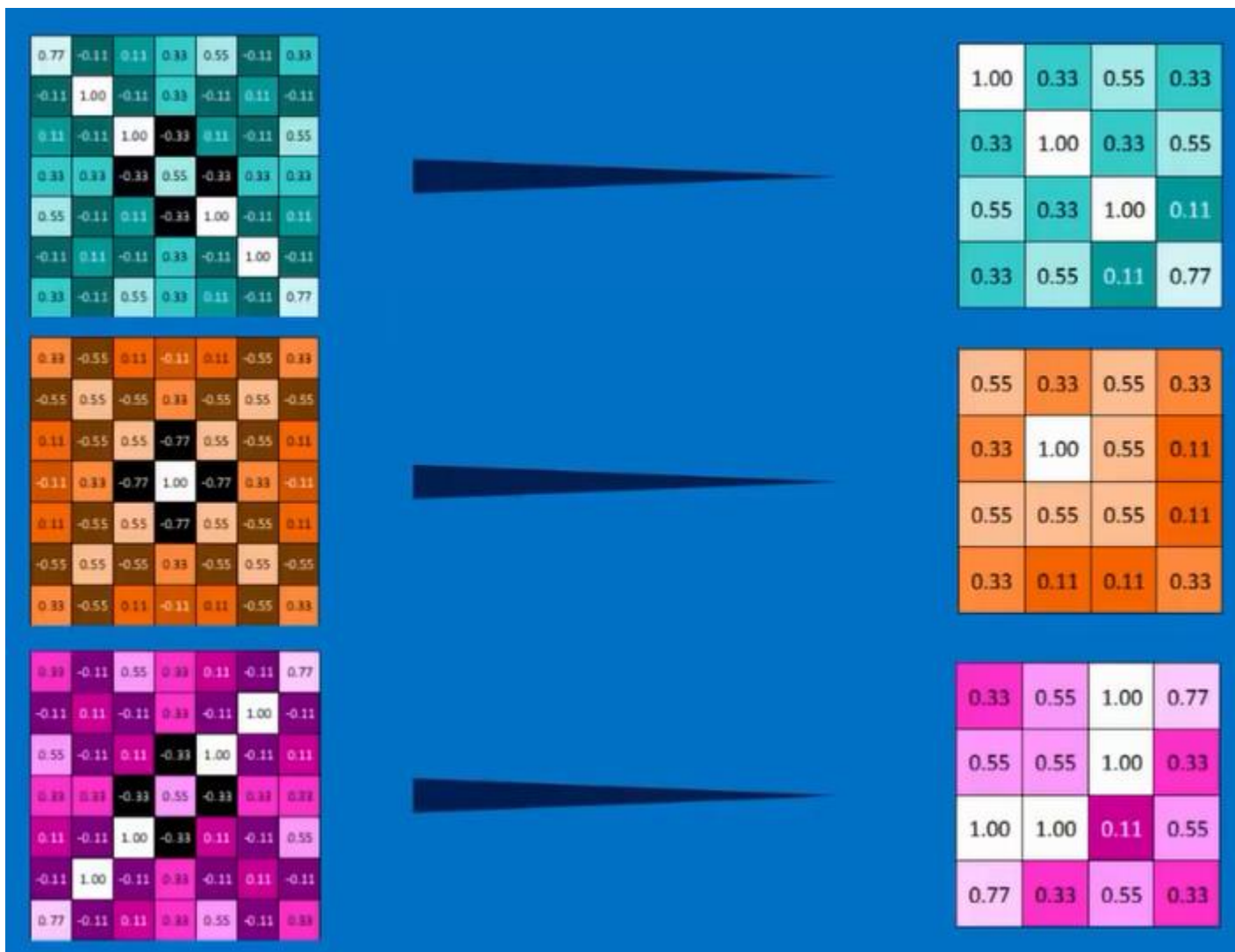
卷积神经网络处理流程



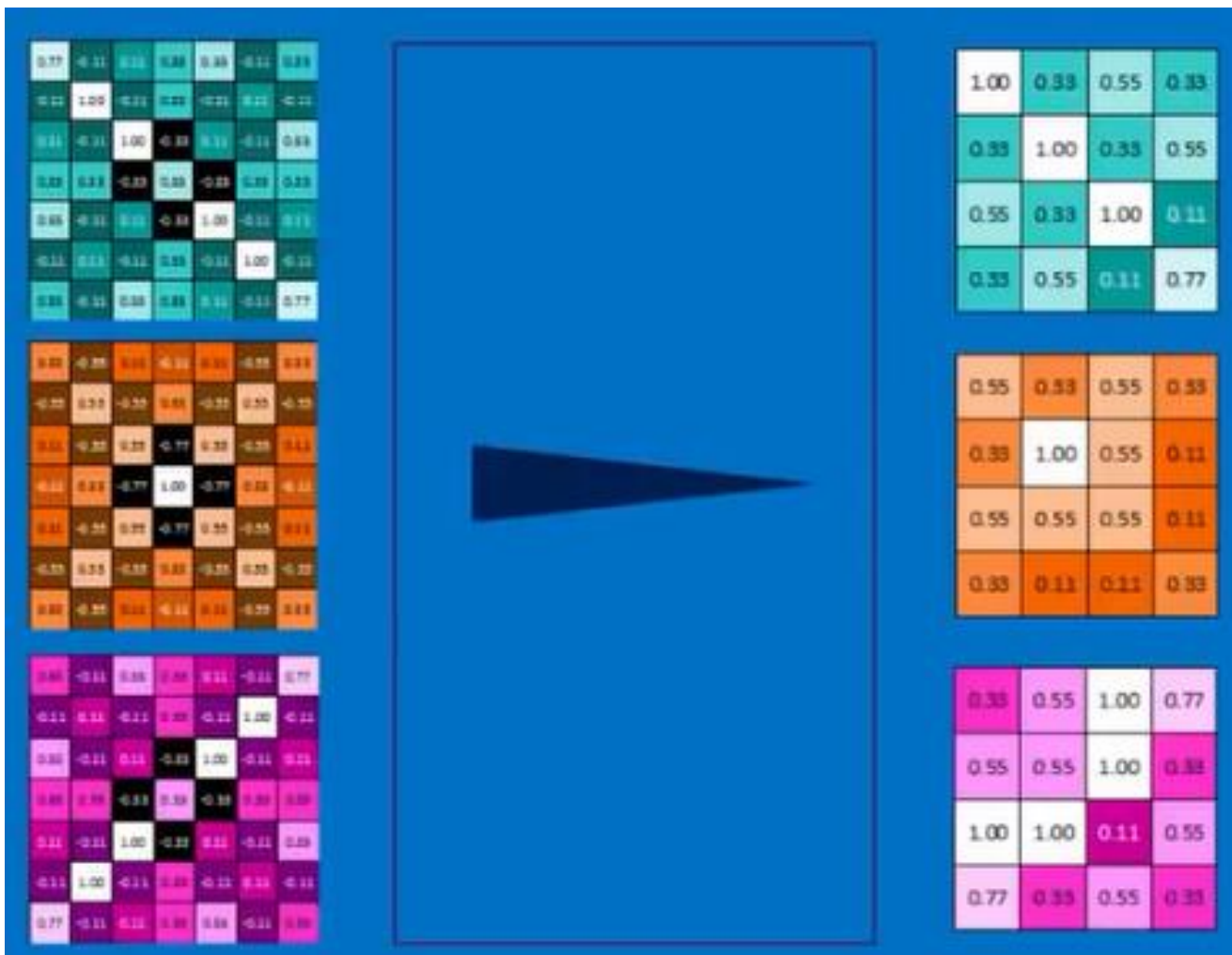
卷积神经网络处理流程



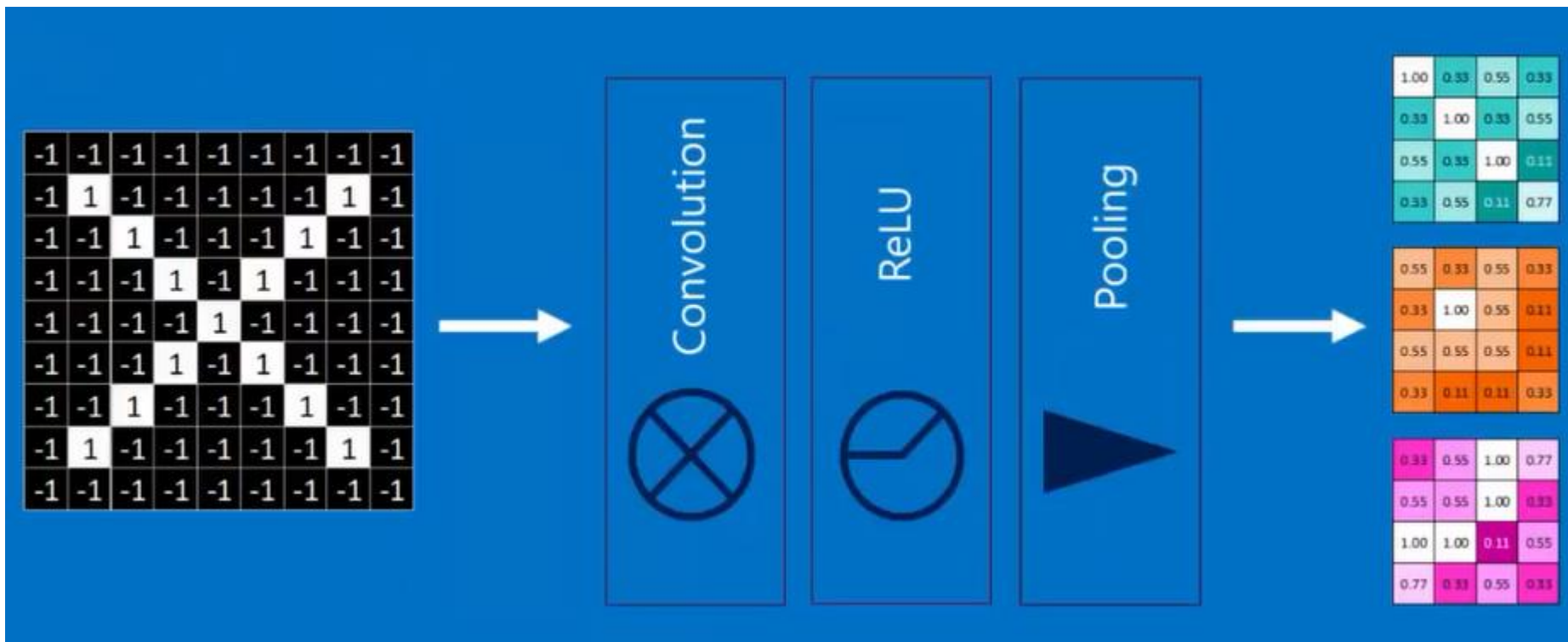
卷积神经网络处理流程



卷积神经网络处理流程



卷积神经网络处理流程

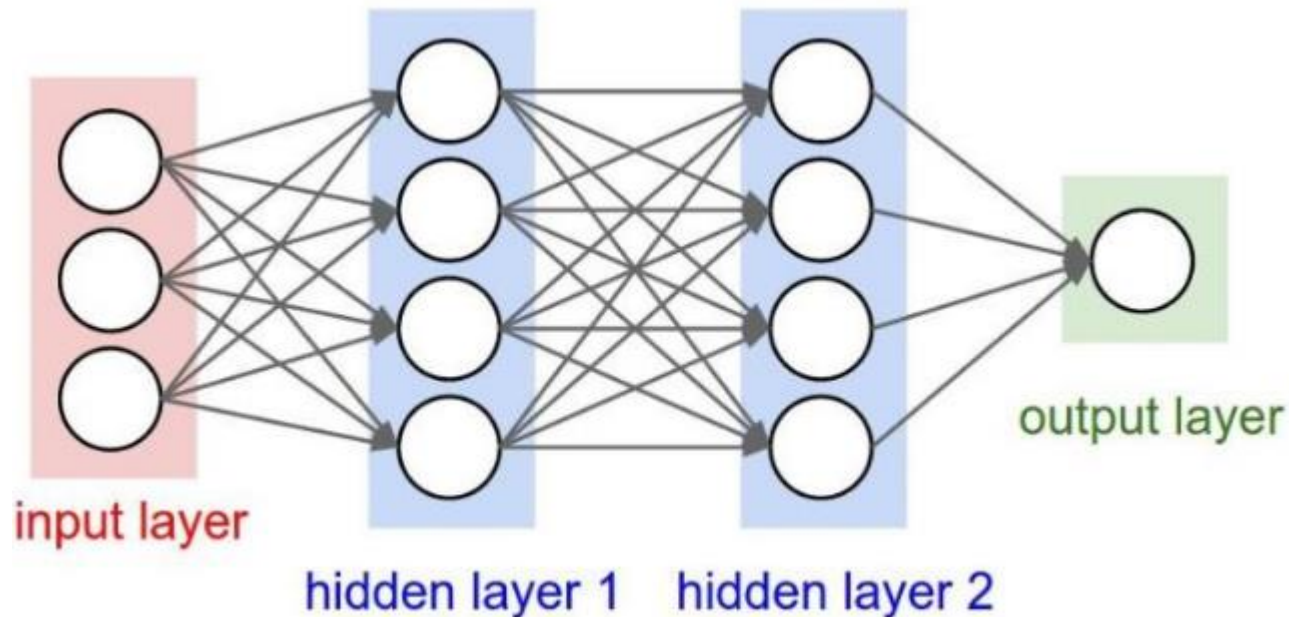


卷积神经网络处理流程

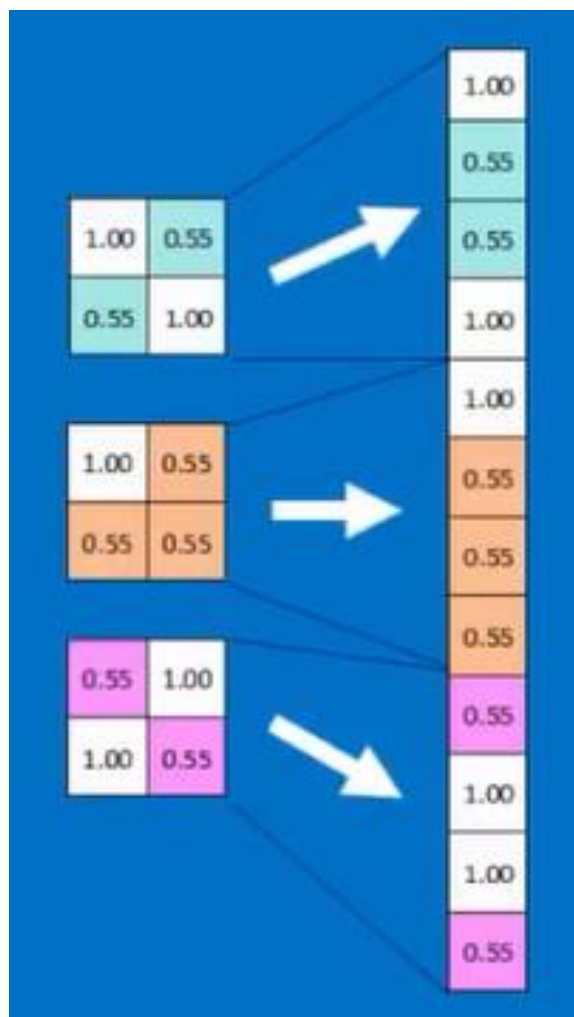


卷积神经网络处理流程

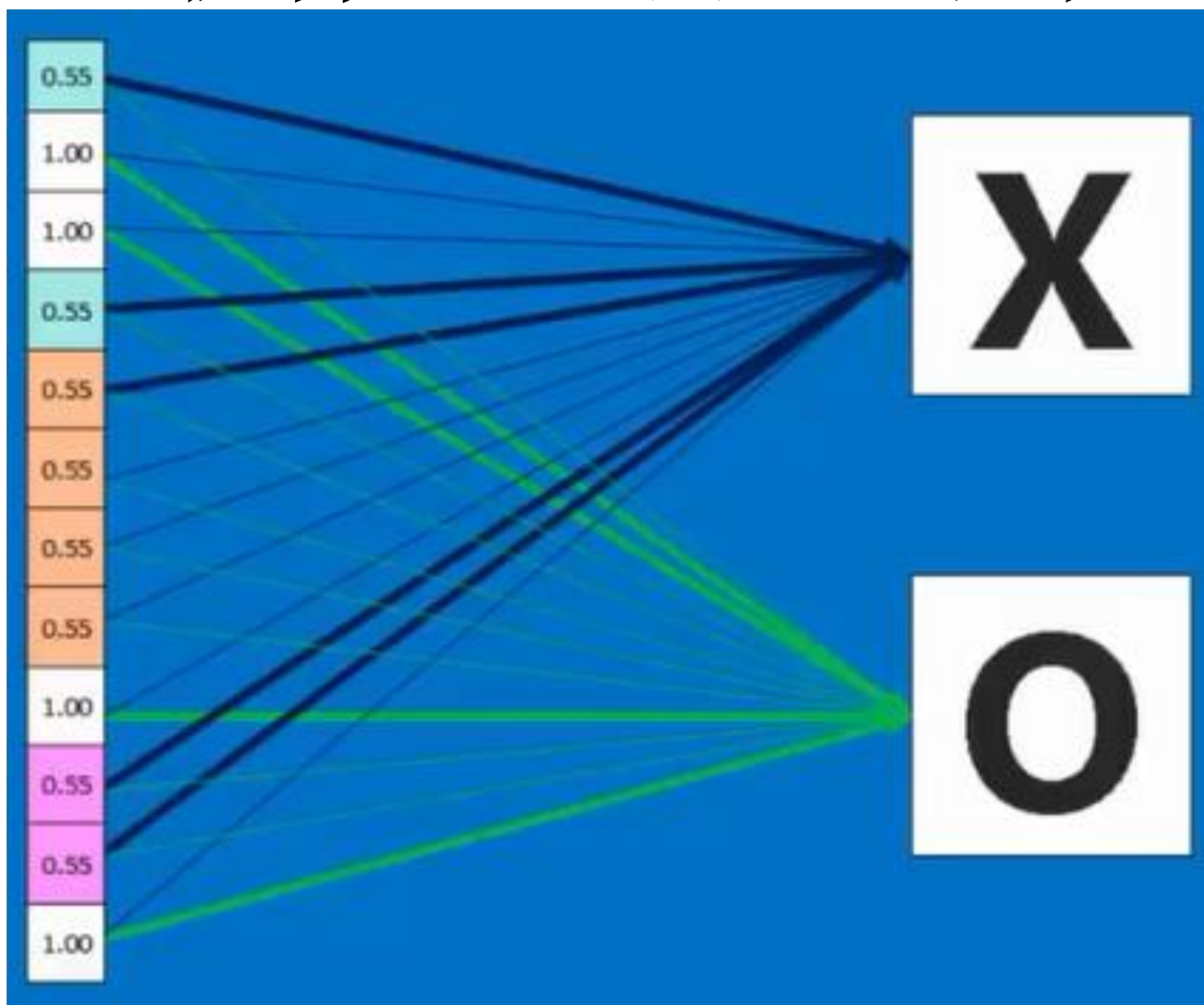
- （5）全连接层：两层之间所有神经元都有权重连接，通常全连接层在卷积神经网络尾部。也就是跟传统的神经网络神经元的连接方式是一样的：



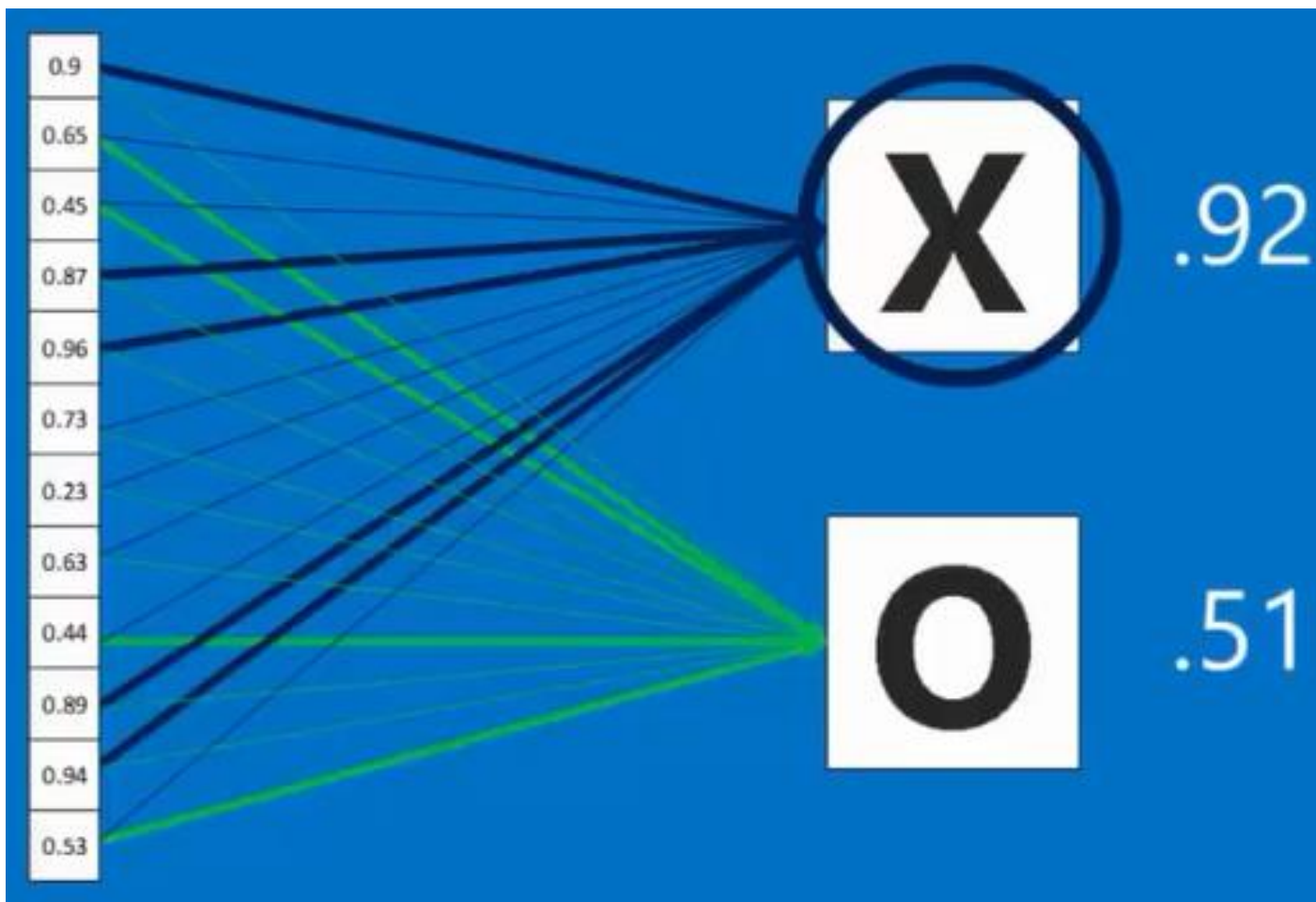
卷积神经网络处理流程



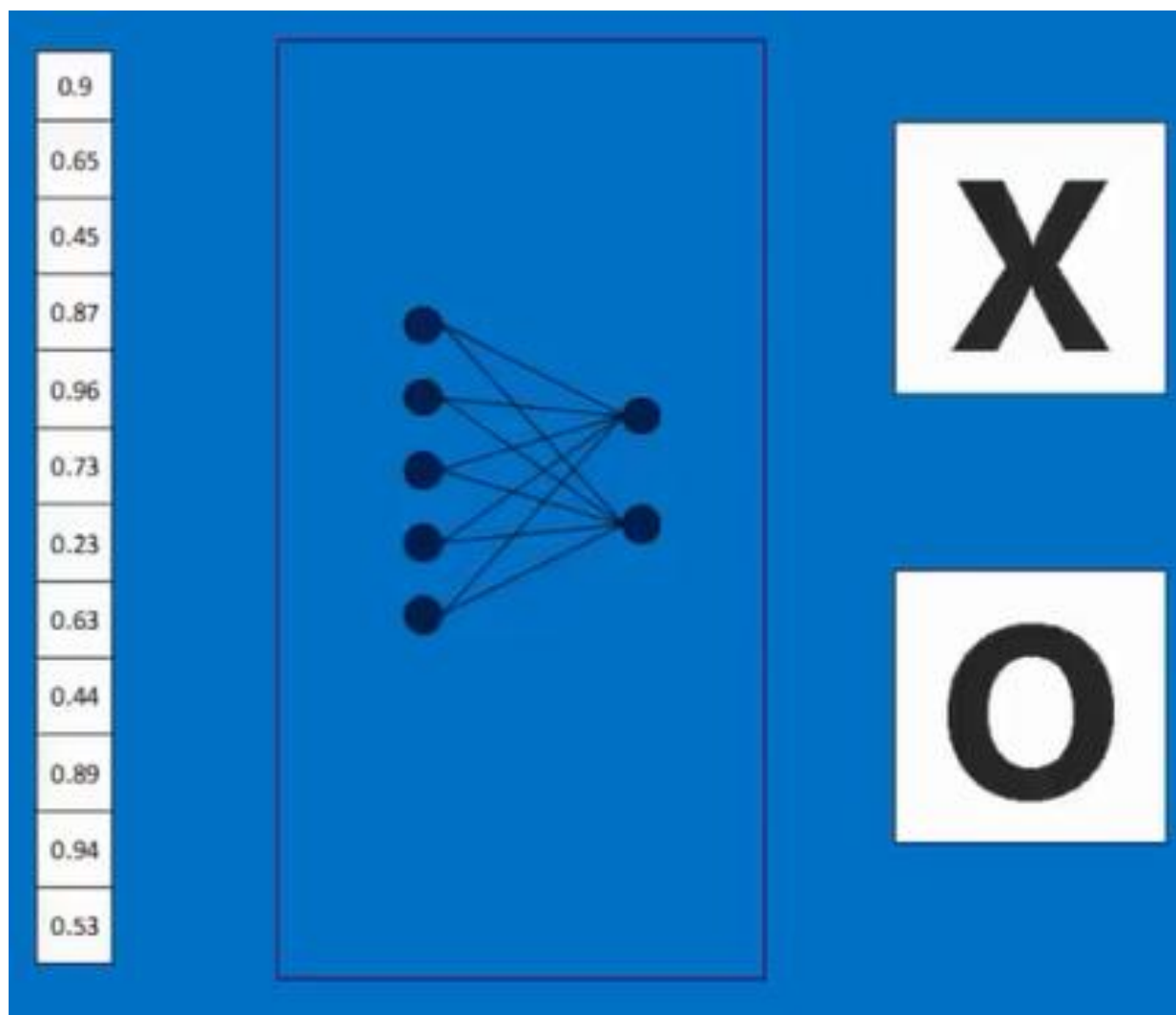
卷积神经网络处理流程



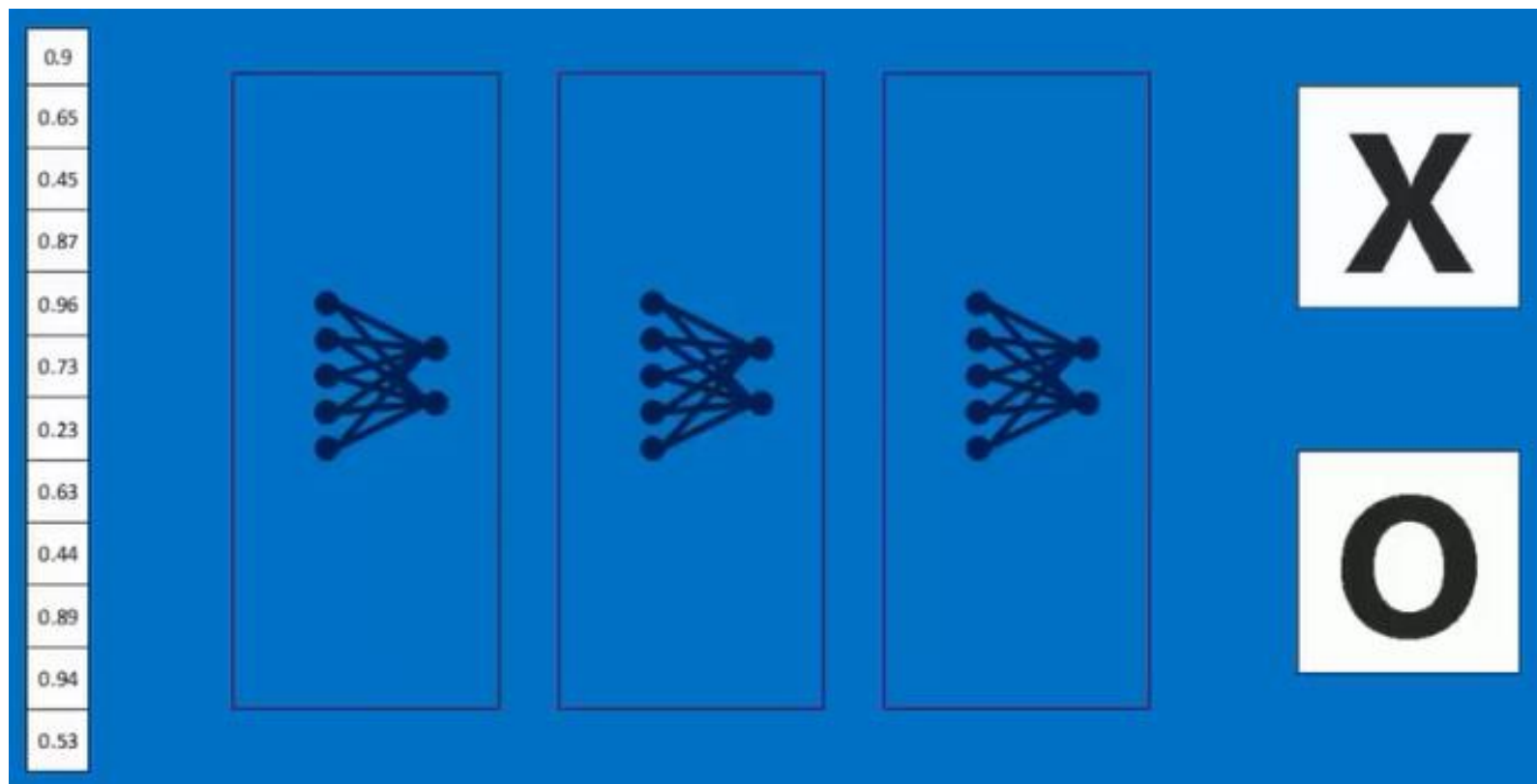
卷积神经网络处理流程



卷积神经网络处理流程



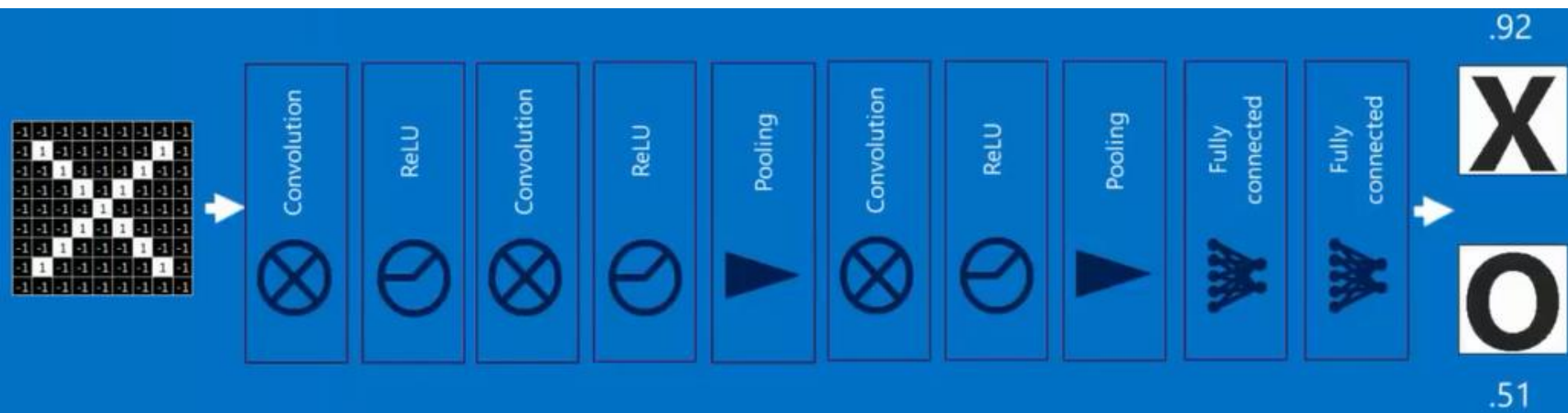
卷积神经网络处理流程



卷积神经网络处理流程



卷积神经网络处理流程

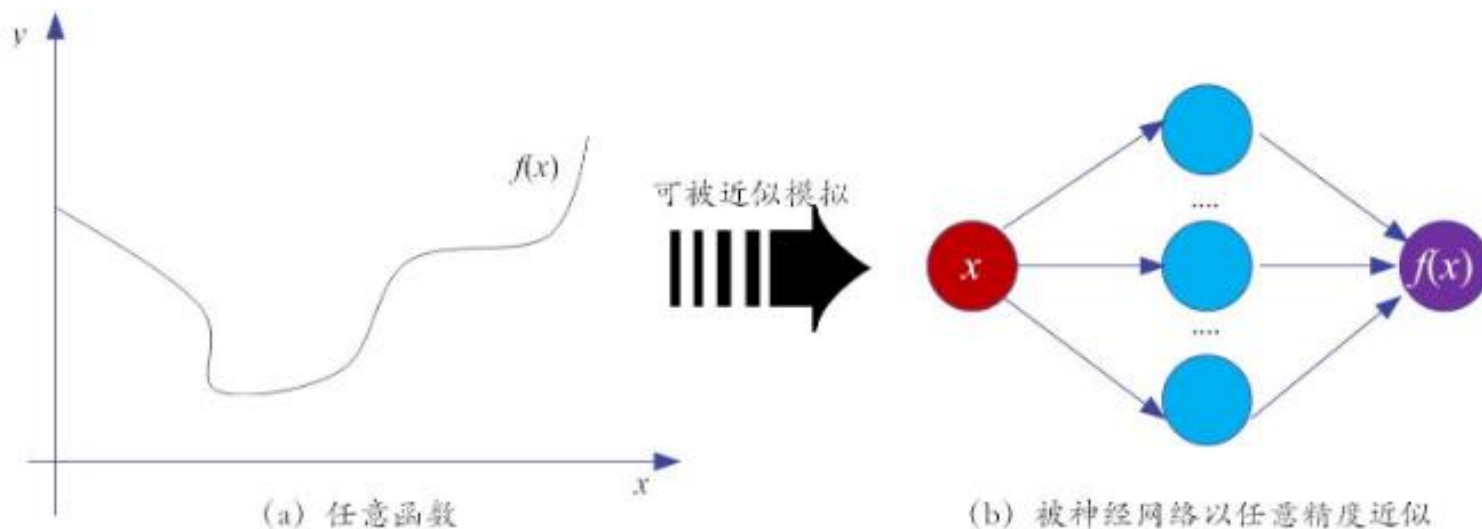


	Right answer	Actual answer	Error
X	1	0.92	0.08
O	0	0.51	0.49
		Total	0.57

关于通用逼近理论

通用逼近理论

- 人工神经网络可以在理论上证明：“一个包含足够多隐层神经元的多层前馈网络，能以任意精度逼近任意预定的连续函数。”这个定理也被称之为通用近似定理（Universal Approximation Theorem）。



通用逼近理论

- “通用近似定理”告诉我们，不管函数 $f(x)$ 在形式上有多复杂，我们总能确保找到一个神经网络，对任何可能的输入 x ，以任意高的精度近似输出值为 $f(x)$ 。
- 即使函数有多个输入和输出，即 $f = f(x_1, x_2, \dots, x_m)$ ，通用近似定理的结论也是成立的。

通用逼近理论

- 使用这个定理时需要注意如下两点：
 - (1) 定理说的是，可以设计一个神经网络尽可能好地去“近似”某个特定函数，而不是说“准确”计算这个函数。我们通过增加隐层神经元的个数来提升近似的精度。
 - (2) 被近似的函数，必须是连续函数。如果函数是非连续的，也就是说有极陡跳跃的函数，神经网络就“爱莫能助”了。

通用逼近理论

- 即使函数是连续的，神经网络能不能解决所有问题，也是有争议的。通用近似定理在理论上是一回事，而在实际操作又是另外一回事。比如，生成对抗网络（GAN）发明者Ian Goodfellow就曾发表意见说：“仅含有一层的前馈网络，的确足以有效地表示任何函数，但是，这样的网络结构可能会格外庞大，进而无法正确地学习和泛化。”

通用逼近理论

- Goodfellow的言外之意是说，“广而浅薄”的神经网络在理论上是万能的，但在实践中却不是那么回事。因此，网络往“深”的方向去做，才是正途。
- 事实上，1989年“通用近似定理”就被提出了，到2006年深度学习开始厚积而薄发，这期间神经网络并没有因为这个理论而得到蓬勃发展。因此，从某种程度上，这也验证了Goodfellow的判断。

通用逼近理论

- 设目标函数为： $f(x) = x^3 + x^2 - x - 1$
- 神经元使用的激活函数是ReLU（Rectified Linear Units，修正线性单元）： $ReLU = \max(0, x)$

通用逼近理论

- 重复尝试ReLU单元的不同组合，直到它拟合地看起来很像目标函数。如：使用6个ReLU神经元进行组合：

$$neruon_1(x) = ReLU(-5x - 7.7)$$

$$neruon_2(x) = ReLU(-1.2x - 1.3)$$

$$neruon_3(x) = ReLU(1.2x + 1)$$

$$neruon_4(x) = ReLU(1.2x - 0.2)$$

$$neruon_5(x) = ReLU(2x - 1.1)$$

$$neruon_6(x) = ReLU(5x - 5)$$

通用逼近理论

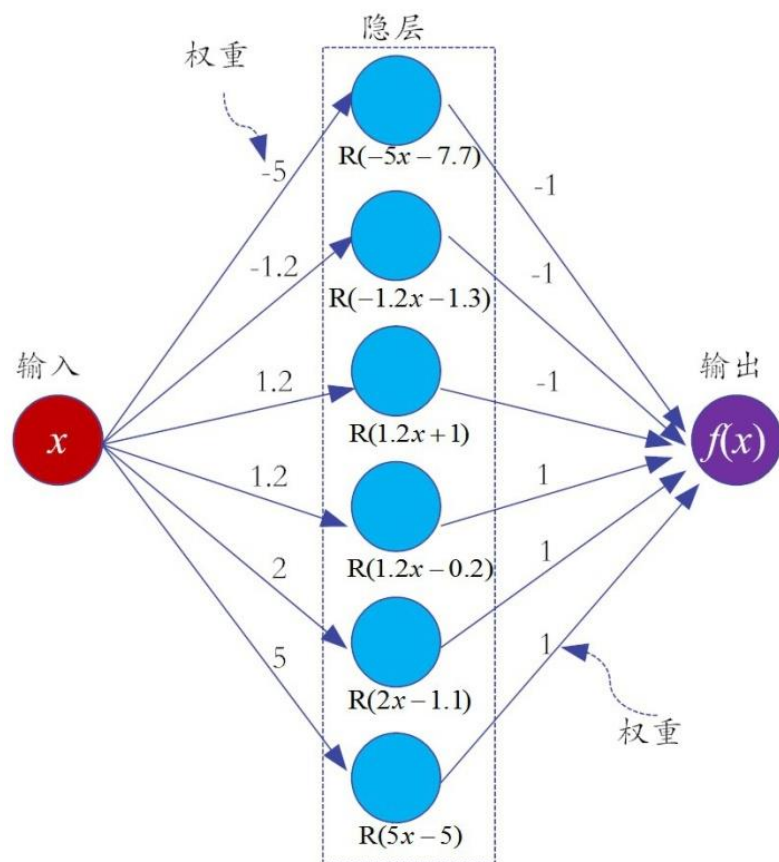
- 在上述公式中，诸如“-5”、“-1.2”等数值，是输入神经元与隐层神经元之间的连接权值（weight），而诸如“-7.7”、“-1.3”等数值，是隐层神经元的偏置（bias）。当然，这些参数都是通过不断地训练学习得到的。
- 然后把这些神经元组合起来，就形成了输出函数（为了简便，我们用n取代了neroun）：

$$f(x) = -n_1(x) - n_2(x) - n_3(x) + n_4(x) + n_5(x) + n_6(x)$$

- 神经元前面的系数“-1”、“-1”、“-1”、“1”、“1”、“1”分别是隐层神经元和输出神经元之间的连接权重值，它们也需要学习才能得到的。

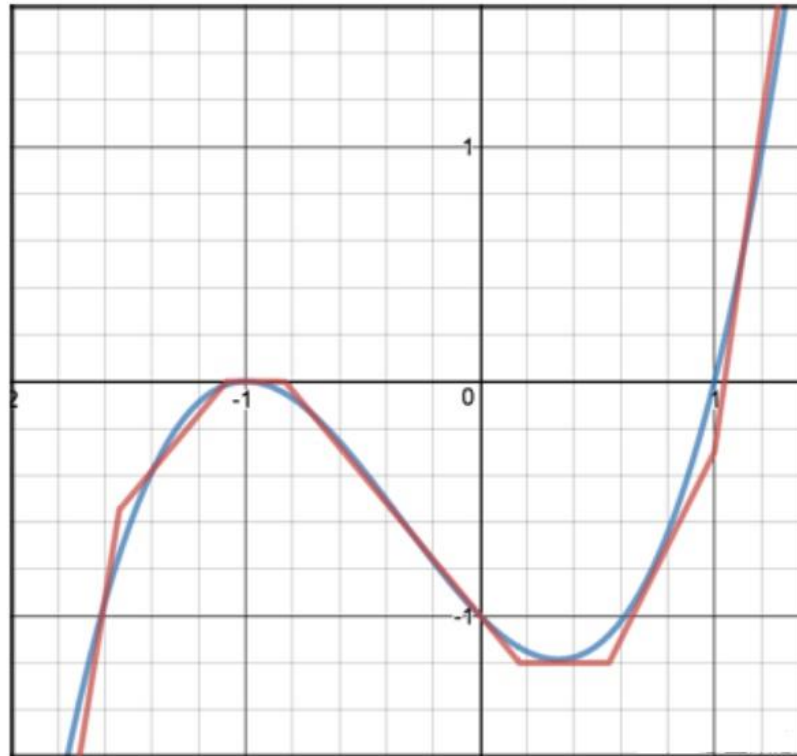
通用逼近理论

- 最终可得到该神经网络如下：



通用逼近理论

- 该神经网络的逼近效果如下：



为什么要“深度”

- 一个机器学习模型的学习能力是与它的容量有关，而容量则于它的复杂程度有关。也就是说模型的学习能力与其复杂度息息相关。
- 为了提升神经网络模型的学习能力，可以增加模型的复杂度。
- 方式一般有两种：增加模型的宽度、增加模型的深度。对于这两种方法而言，增加深度显然会更有效，因为模型变宽，不过是增加了一些计算单元，增加了函数的数量；但是模型变深，不仅仅增加了计算单元，还增加了嵌入的程度，模型也会变得更加复杂。

为什么要“深度”

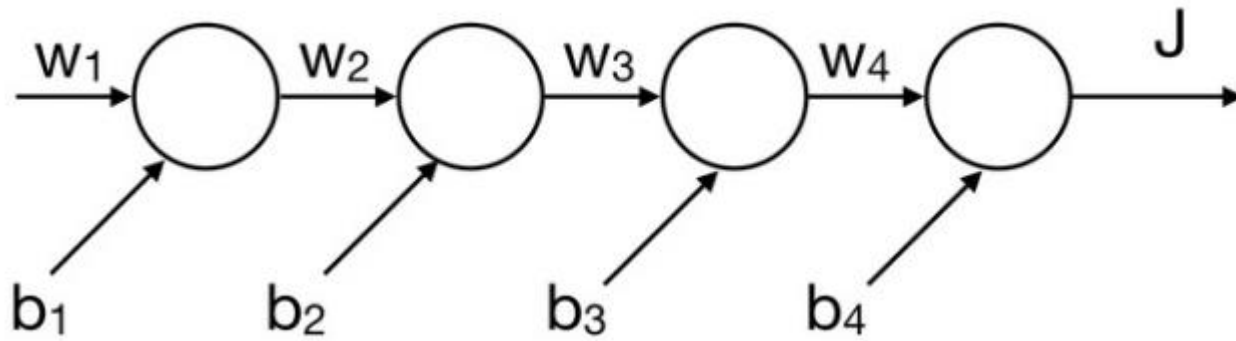
- 神经网络的层数越多，意味着可供调整的参数越多，模型调整的自由度变大，产生的拟合效果也越好。尽管当只有一个隐层的时候，加入无限多的神经元也能增加模型的复杂度，但是经验表明，深度神经网络的表现会比浅层神经网络更高效。
- 简而言之，相对比浅层神经网络，深度神经网络的模型复杂度更高，泛化效果更好。
- 当然了，深度学习有一个前提：如果一个问题是不可微分的、不可计算梯度的，我们就无法为它训练一个神经网络。

“深度”带来的问题

- 梯度消失与爆炸是机器学习中常见的难题，主要是出现在使用梯度下降与方向传播训练人工神经网络的时候。比如使用BP训练网络时，权重的更新值与误差函数的偏导数成比例，由于使用链式法则计算梯度，在某些情况下，梯度值会消失，导致权重无法有效更新，甚至出现网络完全无法继续训练。

“深度”带来的问题

- 假如存在一个4层的网络：



“深度”带来的问题

- 第一个权重的更新值为： $w'_1 = w_1 + \eta \delta_1 x_1$
- 其中：
$$\delta_1 = \frac{\partial J}{\partial b_1} = \frac{\partial J}{\partial b_4} \frac{\partial b_4}{\partial b_3} \frac{\partial b_3}{\partial b_2} \frac{\partial b_2}{\partial b_1}$$
$$= f'_1(b_1)w_2\delta_2 = f'_1(b_1)w_2f'_2(b_2)w_3f'_3(b_3)w_4f'_4(b_4)(y - t)$$
- 如果激活函数都为sigmoid，即：
$$f'(x) = (1 - f(x))f(x) = -(f(x) - \frac{1}{2})^2 + \frac{1}{4}$$
- 则激活函数的导数最大为0.25，因此：
$$|f'(b)w| \leq \frac{1}{4} \quad , \quad \delta_1 \leq f'_1(b_1)(\frac{1}{4})^3(y - t)$$

“深度”带来的问题

- 由此可以发现，神经网络前面的层比后面的层的梯度变化更小，导致变化更慢，从而导致梯度消失。与此类似，假如激活函数的导数大于1，随着层数的递增，梯度变化会以指数形式增加，则可能出现梯度爆炸。

解决方法

- 第一个办法，使用ReLU、Leaky ReLU、ELU等激活函数。

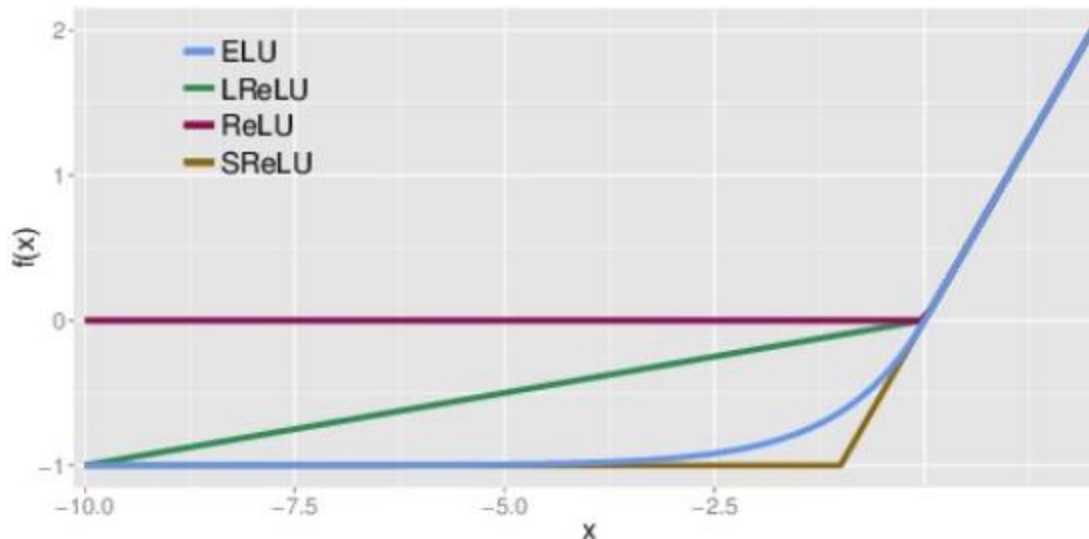
ReLU : $f(x) = \max(0, x)$

- 其思想在于：若激活函数的导数为1，则每层网络的更新速度不会受到影响，不存梯度爆炸与消失的问题。
- 优势：解决梯度消失、爆炸问题；计算简单，速度快
- 缺点：负数部分为0，会影响部分神经元无法激活；输出结果不以0为中心

解决方法

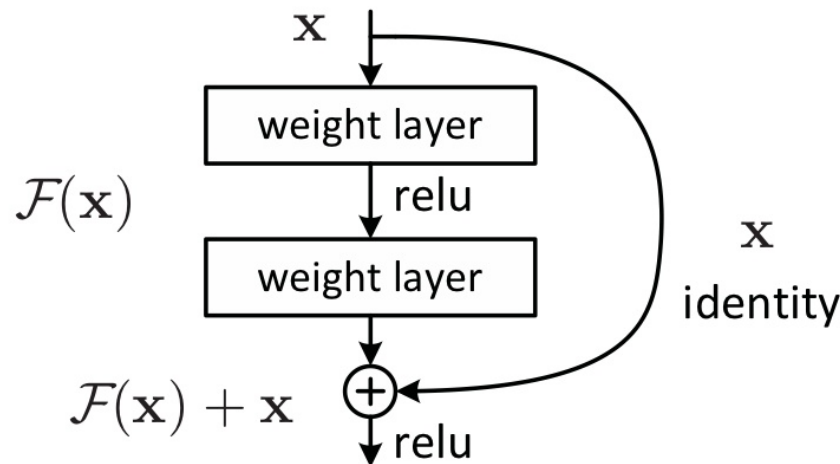
- 尽管ReLU并不完美，但它仍然是使用广泛的一种激活函数。而且，还发展出了ReLU的变种：Leaky ReLU和Elu，它们解决了负数为0的问题：

$$\text{Leaky ReLU: } f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases} \quad \text{ELU: } f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}$$



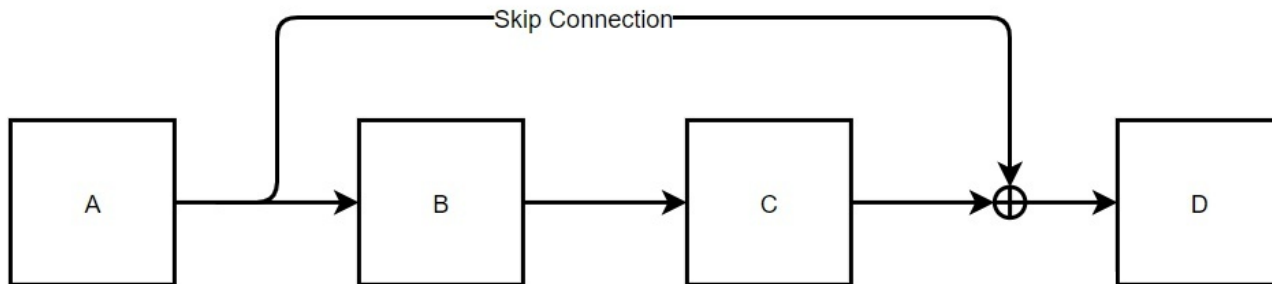
解决方法

- 第二个办法，使用残差结构，构建残差神经网络（ResNets）。
- 这是另外一种有效地解决梯度消失的方式，其核心思想是跃层连接（Skip Connection）。它将输入和输入的一个非线性变化进行线性叠加，如下图所示。



解决方法

- 前面讲到，在反向传播中，当导数小于0时，链式法则会使得梯度的变化越来越小，从而可能造成梯度消失。但是通过跃层连接，神经元的输出变成了 $f(x)+x$ ，在其导数上加上一个1。这样一来，尽管导数可能很小，也能够继续反向传播。例如：



解决方法

- 假设网络输入为 x ，从A到D经历了两次前向传播以及一次跃层连接，根据后向传播的链式法则，有：

$$\frac{\partial L}{\partial X_{Aout}} = \frac{\partial L}{\partial X_{Din}} \frac{\partial X_{Din}}{\partial X_{Aout}}$$

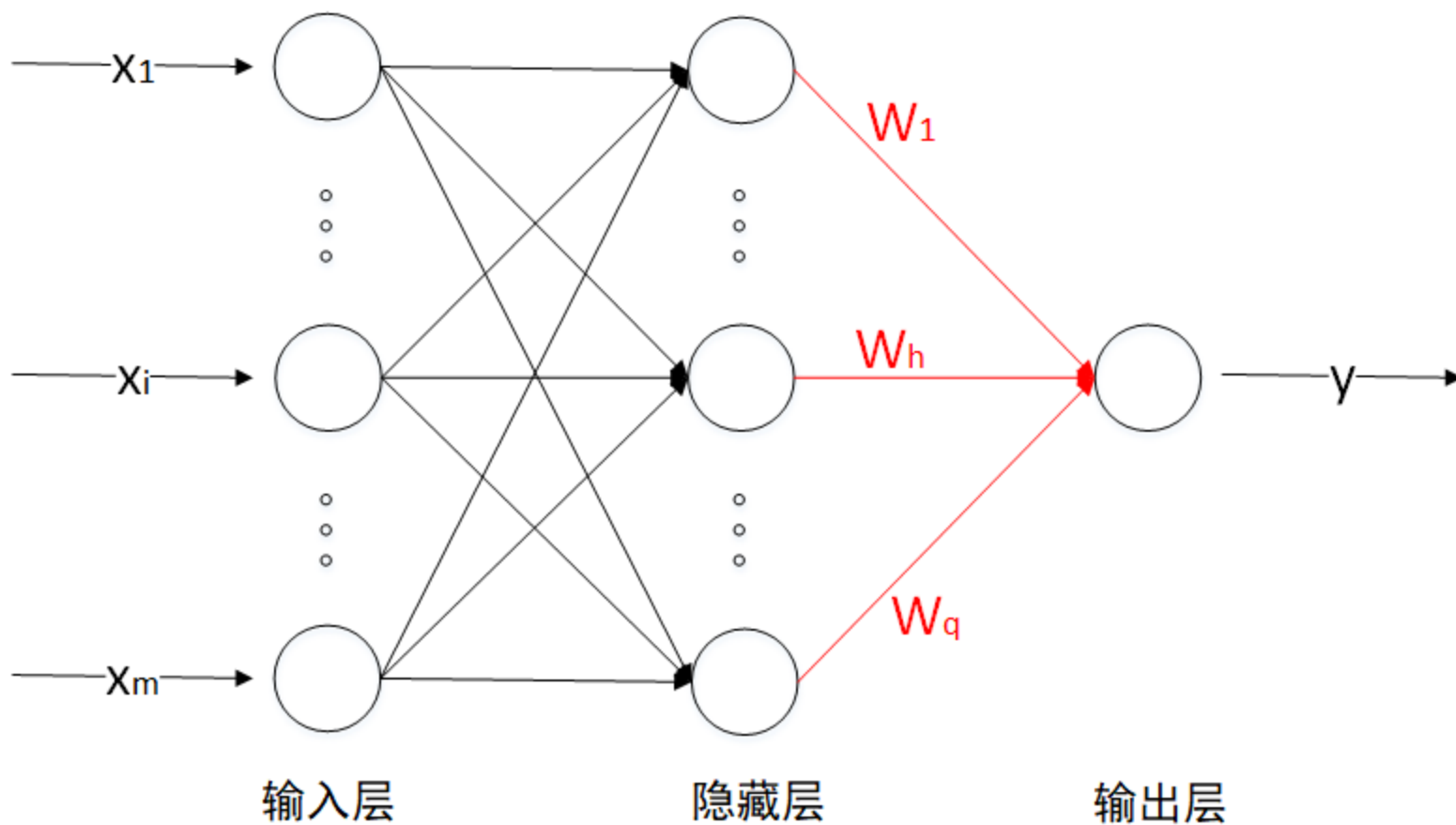
- 由于 $X_{Din} = X_{Aout} + C(B(X_{Aout}))$ ，所以：

$$\frac{\partial L}{\partial X_{Aout}} = \frac{\partial L}{\partial X_{Din}} \left[1 + \frac{\partial X_{Din}}{\partial X_C} \frac{\partial X_C}{\partial X_B} \frac{\partial X_B}{\partial X_{Aout}} \right]$$

- 此时，即使在A-B-C的后向传播中出现了梯度衰减的情况，但是D处的梯度依然能够直接传递到A，也即是实现了梯度的跨层传播。

其他神经网络简介

径向基函数网络（RBF网络）



径向基函数网络（RBF网络）

- RBF网络是一种单隐藏层前馈神经网络，它使用径向基函数作为隐藏层神经元激活函数，而输出层则是对隐藏层神经元输出的线性组合。假定输入为m维向量x，输出为实数值，则RBF网络可表示为：

$$\varphi(x) = \sum_{i=1}^q \varpi_i \rho(x, c_i)$$

- 其中，q为隐藏层神经元个数， c_i 和 ϖ_i 分别是第i个隐藏层对应的中心和权重， $\rho(x, c_i)$ 是径向基函数。

径向基函数网络（RBF网络）

- 径向基函数是某种沿径向对称的标量函数，通常定义为样本 x 到数据中心 c_i 之间的欧氏距离的单调函数。常用的高斯径向基函数形式为：

$$\rho(x, c_i) = e^{-\beta_i \|x - c_i\|^2}$$

- 可以证明，具有足够多隐藏层神经元的RBF网络能够以任意精度逼近任意连续函数。
- 通常，我们采用两步过程来训练RBF网络，第一步先确定神经元中心 c_i ，常用算法包括随机采样、聚类等；第二步则利用BP算法来确定参数 ϖ_i 和 β_i 。

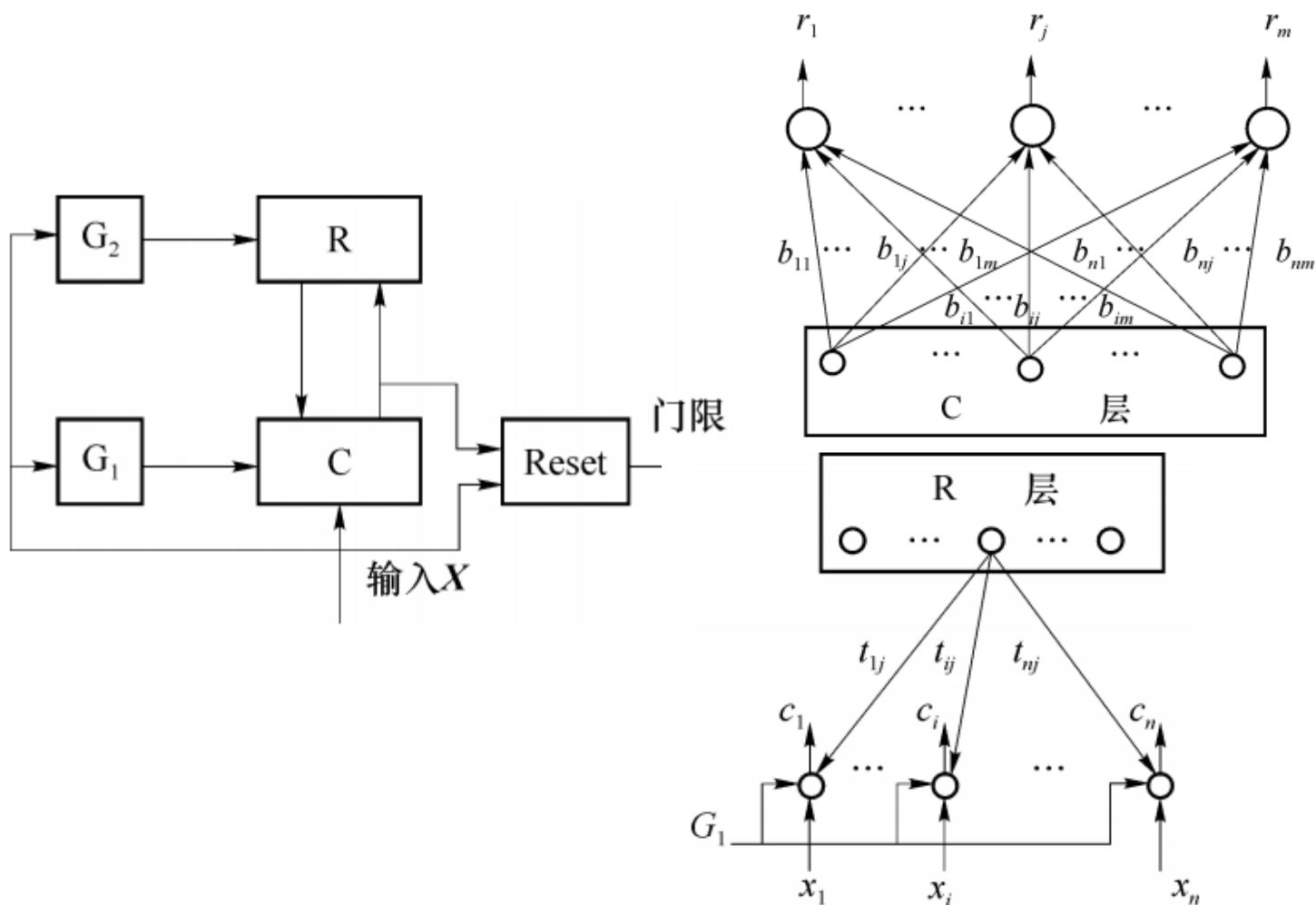
自适应谐振理论网络（ART网络）

- ART网络属于竞争型学习网络。竞争型学习是神经网络中一种常用的无监督学习策略，在使用该策略时，网络的输出神经元相互竞争，每一时刻仅有一个竞争获胜的神经元被激活，其他神经元的状态被抑制，这种机制被称为“胜者通吃”。
- ART网络由比较层C、识别层R、识别阈值和重置模块构成。其中，比较层C和识别层R是由两层神经元构成的两个子系统，比较层C负责接收输入样本，并将其传递给识别层神经元。识别层R每个神经元对应一个模式类，神经元数目可在训练过程中动态增长，以增加新的模式类。此外还包含3种控制信号：复位信号Reset、逻辑控制信号G1和G2。

自适应谐振理论网络（ART网络）

- 在接收到比较层的输入信号之后，识别层神经元之间相互竞争以产生获胜神经元。竞争的最简单方式是：计算输入向量与每个识别层神经元所对应的模式类的代表向量之间的距离，距离最小者胜。获胜神经元将向其他识别层神经元发送信号，抑制其激活。若输入向量与获胜神经元所对应的代表向量之间的相似度大于识别阈值，则当前输入样本将被归为该代表向量所属类别。同时，网络连接权将被更新，使得以后在接收到相似输入样本时该模式类会计算出更大的相似度，从而使该获胜神经元有更大可能获胜；若相似度不大于识别阈值，则重置模块将在识别层增设一个新的神经元，其代表向量就设置为当前输入向量。

自适应谐振理论网络 (ART网络)



自适应谐振理论网络（ART网络）

- C层有 n 个神经元，每个接收来自3个方面的信号：外界输入信号，R层获胜神经元的外接权向量的返回信号和控制信号G1。C层神经元的输出是根据2/3的多数表决原则产生，输出值与三个信号中的多数信号值相同。

自适应谐振理论网络（ART网络）

- 对于C层，网络开始运行时， $G1 = 1$ ，识别层尚未产生竞争获胜神经元，因此反馈信号为0。由2/3规则，C层输出应取决于输入信号，有 $C=X$ 。当网络识别层出现反馈回送信号时， $G1=0$ ，由2/3规则，C层输出取决于输入信号与反馈信号的比较结果，如果 $x_i = t_{ij}$ ，则 $c_i = x_i$ ，否则 $c_i=0$ 。
- 可以看出控制信号G1的作用是使得比较层能够区分网络运行的不同阶段，网络开始运行阶段G1的作用是使得C层对输入信号直接输出，之后G1的作用是使C层行使比较功能，此时 c_i 为 x_i 和 t_{ij} 的比较信号，两者同时为1，则 c_i 为1，否则为0。可以看出R层反馈信号对C层输出有调节作用。

自适应谐振理论网络（ART网络）

- 对于R层，其功能相当于前馈竞争网，R层有 m 个神经元，代表 m 个输入模式类， m 可以动态增长，以设立新的模式类。C层的输出向量C沿着R层神经元的内接权向量到达R层神经元，经过竞争在产生获胜神经元处指示本次输入模式的所属类别。获胜神经元输出为1，其余为0。
- R层每个神经元都对应着两个权向量，一个是将C层前馈信号汇聚到R层的内接权向量，另一个是将R层反馈信号散发到C层的外接权向量。

自适应谐振理论网络（ART网络）

- 信号G2检测输入模式X是否为0，它等于X各分量的逻辑或，如果 x_i 全为0，则 $G2=0$ ，否则 $G2=1$ 。R层输出向量各分量的逻辑或为 $R0$ ，则信号 $G1=G2$ 与（ $R0$ 的非）。当R层输出向量的各分量全为0而输入向量X不是0向量时， $G1$ 为1，否则 $G1$ 为0。 $G1$ 的作用就是使得比较层能够区分网络运行的不同阶段，网络开始运行阶段 $G1$ 的作用是使得C层对输入信号直接输出，之后 $G1$ 的作用是使C层行使比较功能，此时 c_i 为 x_i 和 t_{ij} 的比较信号，两者同时为1，则 c_i 为1，否则为0。Reset信号的作用是使得R层竞争获胜神经元无效，如果根据某种事先设定的测量标准， T_j 与X未达到设定的相似度，表明两者未充分接近，于是系统发出Reset信号，使得竞争获胜神经元无效。

自适应谐振理论网络（ART网络）

- ART网络运行时接受来自环境的输入模式，检查输入模式与R层所有已存储模式类之间的匹配程度。R层所存储的模式类是通过对应R层神经元的外接权向量体现出来的，对于匹配程度最高的获胜神经元，网络要继续考察其存储模式类与当前输入模式的相似程度。

自适应谐振理论网络（ART网络）

- 对比相似程度按照预先设计的参考门限来考察，可能出现如下的情况：
 - A. 如果相似度超过参考门限，将当前输入模式归为该类，全职调整规则是相似度超过参考门限的神经元调整其相应的内外接权向量，以使得以后遇到与当前输入模式接近的样本时能够得到更大的相似度；其他权向量则不做改动。
 - B. 如果相似度不超过门限值，则对R层匹配程度次高的神经元代表的模式类进行相似度的考察，若超过门限，网络的运行回到情况A，否则仍然回到情况B。如果最终存储的所有模式类与当前输入模式的相似度都没有超过门限，此时需在网络输出端设立一个代表新模式类的神经元，用以代表及存储该模式，以便参加以后的匹配过程。网络对所接受的每个新输入样本，都进行上面的运行过程。

自适应谐振理论网络（ART网络）

- 识别阈值对ART网络的性能有重要影响，当识别阈值较高时，输入样本将会被分成比较多、比较精细的模式类；如果识别阈值较低，则会产生比较少、比较粗略的模式类。
- ART网络比较好地缓解了竞争型学习中的“可塑性-稳定性窘境”。可塑性是指神经网络要有学习新知识地能力，而稳定性则是指神经网络在学习新知识时要保持对旧知识的记忆。这使得ART网络具有一个很重要的优点：增量学习或是在线学习。

自组织映射网络（SOM网络）

- SOM网络也是一种竞争学习型的无监督神经网络，它将高维输入数据映射到低维空间（通常为二维），同时保持输入数据在高维空间的拓扑结构，即：将高维空间中相似的样本点映射到网络输出层中的邻近神经元。
- SOM网络中的输出层神经元以矩阵方式排列在二维空间中，每个神经元都拥有一个权向量，网络在接收输入向量后，将会确定输出层获胜神经元，它决定了该输入向量在低维空间的位置。SOM的训练目标就是为每个输出层神经元找到合适的权向量，以达到保持拓扑结构的目的。

自组织映射网络（SOM网络）

- 生物学研究表明，在人脑的感觉通道上，神经元的组织原理是有序排列的。当外界的特定时空信息输入时，大脑皮层的特定区域兴奋，而且类似的外界信息在对应的区域是连续映像的。生物视网膜中有许多特定的细胞对特定的图形比较敏感，当视网膜中有若干个接收单元同时受特定模式刺激时，就使大脑皮层中的特定神经元开始兴奋，输入模式接近，与之对应的兴奋神经元也接近；在听觉通道上，神经元在结构排列上与频率的关系十分密切，对于某个频率，特定的神经元具有最大的响应，位置相邻的神经元具有相近的频率特征，而远离的神经元具有的频率特征差别也较大。大脑皮层中神经元的这种响应特点不是先天安排好的，而是通过后天的学习自组织形成的。

自组织映射网络（SOM网络）

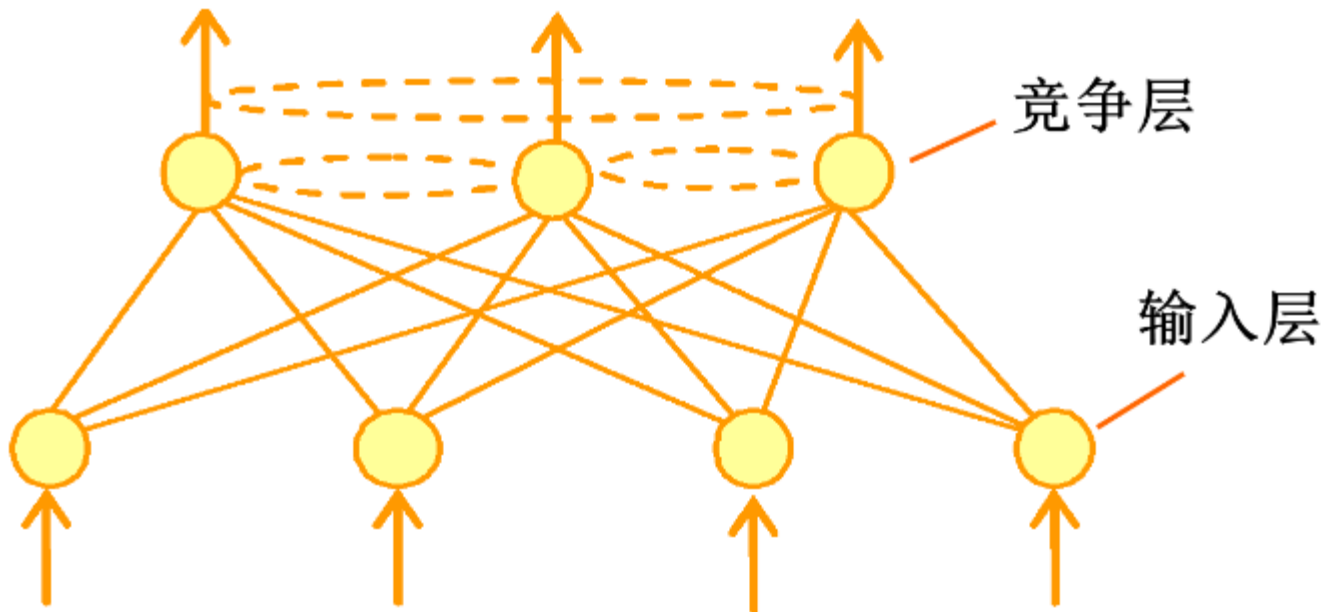
- 在生物神经系统中，存在着一种侧抑制现象，即一个神经细胞兴奋以后，会对周围其他神经细胞产生抑制作用。这种抑制作用会使神经细胞之间出现竞争，其结果是某些获胜，而另一些则失败。表现形式是获胜神经细胞兴奋，失败神经细胞抑制。自组织映射（竞争型）神经网络就是模拟上述生物神经系统功能的人工神经网络。

自组织映射网络（SOM网络）

- 自组织映射（竞争型）神经网络的结构及其学习规则与其他神经网络相比有自己的特点。在网络结构上，它一般是由输入层和竞争层构成的两层网络；两层之间各神经元实现双向连接，而且网络没有隐含层。有时竞争层各神经元之间还存在横向连接。
- 在学习算法上，它模拟生物神经元之间的兴奋、协调与抑制、竞争作用的信息处理的动力学原理来指导网络的学习与工作，而不像多层神经网络（MLP）那样是以网络的误差作为算法的准则。竞争型神经网络构成的基本思想是网络的竞争层各神经元竞争对输入模式响应的机会，最后仅有一个神经元成为竞争的胜者，这一获胜神经元则表示对输入模式的分类。因此，¹¹⁵很容易把这样的结果和聚类联系在一起。

自组织映射网络（SOM网络）

- 下图为一种自组织神经网络的典型结构，由输入层和竞争层组成。主要用于完成的任务基本还是“分类”和“聚类”，前者有监督，后者无监督。



自组织映射网络（SOM网络）

- SOM网络的训练过程为：在接收到一个训练样本后，每个输出层神经元会计算该样本与自身携带的权向量之间的距离，距离最近的神经元成为竞争获胜者，称为最佳匹配单元。然后，最佳匹配单元及其邻近神经元的权向量将被调整，以使这些权向量与当前输入样本的距离缩小。这个过程不断迭代，直至收敛。

玻尔兹曼机（boltzmann机）

- 神经网络中有一类模型是为网络状态定义一个“能量”，当“能量”达到最小时就认为这个网络达到了理想状态，而网络的训练就是在最小化这个能量函数。玻尔兹曼机就是这样一种“基于能量”的模型。其“能量”的定义参照了玻尔兹曼分布（也叫吉布斯分布）。玻尔兹曼分布是一种覆盖系统各种状态的概率分布、概率测量或者频率分布。当有保守外力（如重力场、电场等）作用时，气体分子的空间位置不是均匀分布，不同位置处分子数密度不同。玻尔兹曼分布律则描述了理想气体在受保守外力作用、或保守外力场的作用不可忽略时，处于热平衡态下的气体分子按能量的分布规律。

玻尔兹曼机（boltzmann机）

- boltzmann机的神经元分成两层：显层和隐藏层。显层用于数据的输入和输出，隐藏层则被理解为数据的内在表达。
- boltzmann机中的神经元都是布尔型的，只能取0和1两种状态。状态1表示激活，状态0表示抑制。令向量s表示n个神经元的状态， ϖ_{ij} 表示神经元i和j之间的连接权， θ_i 表示神经元i的阈值，则状态向量s所对应的boltzmann机能量定义为：

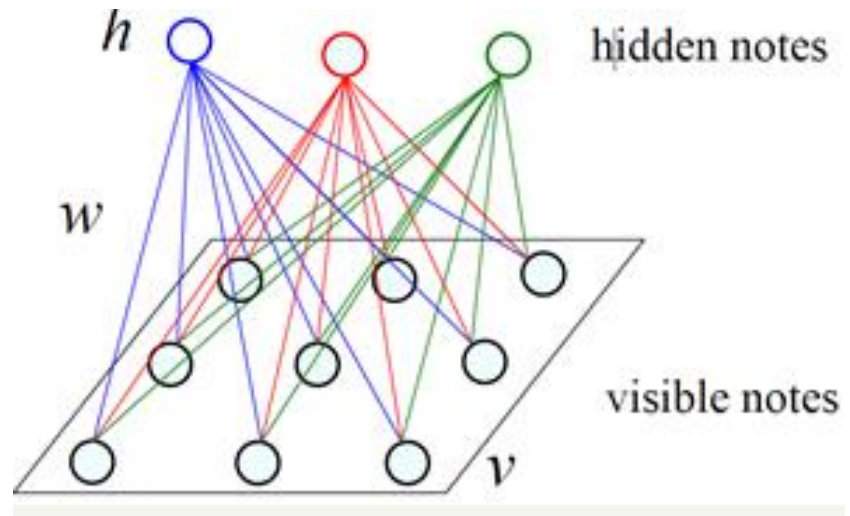
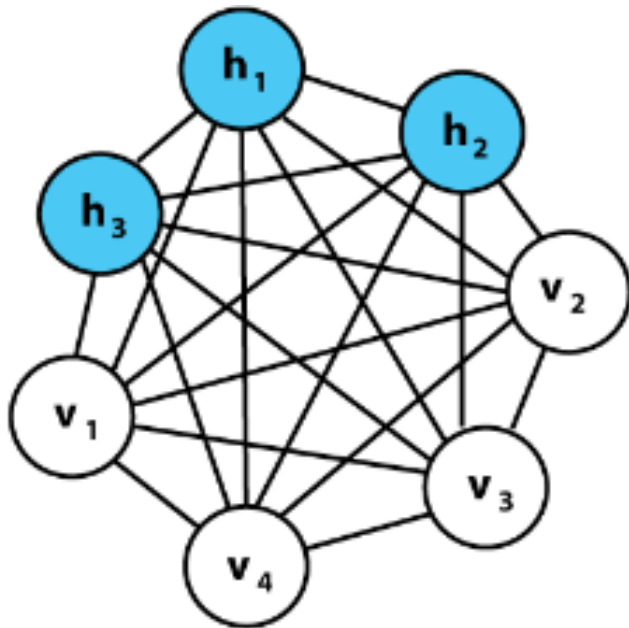
$$E(s) = -\sum_{i=1}^{n-1} \sum_{j=i+1}^n \varpi_{ij} s_i s_j - \sum_{i=1}^n \theta_i s_i$$

玻尔兹曼机（boltzmann机）

- 若boltzmann机中的神经元以任意不依赖于输入值顺序进行更新，则网络最终将达到boltzmann分布，此时状态向量 s 出现的概率将仅由其能量与所有可能状态向量的能量确定：
$$P(s) = \frac{e^{-E(s)}}{\sum_t e^{-E(t)}}$$
- boltzmann机的训练过程就是将每个训练样本视为一个状态向量使其出现的概率尽可能大。标准的boltzmann机是一个全连接图，训练网络的复杂度很高，难以解决实际任务。现实中常采用受限boltzmann机（RBM）。RBM仅保留显层和隐藏层之间的连接，简化了网络结构。

玻尔兹曼机（boltzmann机）

- 下图分别是boltzmann机和受限boltzmann机的网络结构：



玻尔兹曼机（boltzmann机）

- RBM常常采用“对比散度”算法（CD算法）进行训练。假定网络中有d个显层神经元和q个隐藏层神经元。令v和h分别表示显层和隐藏层的状态向量，考虑到同一层内不存在连接，有：

$$P(v|h) = \prod_{i=1}^d P(v_i|h)$$

$$P(h|v) = \prod_{j=1}^q P(h_j|v)$$

玻尔兹曼机（boltzmann机）

- CD算法对每个训练样本 \mathbf{v} ，先计算出隐藏层神经元状态的概率分布，然后根据这个概率分布采样得到 \mathbf{h} ；此外，类似的通过 \mathbf{h} 产生 \mathbf{v}' ，再从 \mathbf{v}' 产生 \mathbf{h}'
- RBM的连接权更新公式为：

$$\Delta \boldsymbol{w} = \eta (\mathbf{v} \mathbf{h}^T - \mathbf{v}' \mathbf{h}'^T)$$

纸质作业提交时间：

4月26日（周一）晚上7点，航海楼515

4月29日（周四）晚上7点，航海楼404

作业提交截止时间：

5月6号（周四），航海楼707