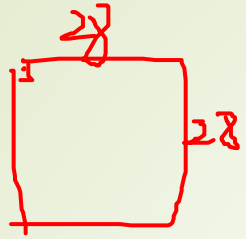
1. 

Total

$L1\,loss = MAE \quad |Y_p - Y|$

X

Y

28X28

↳ x784

(2X2)

Y

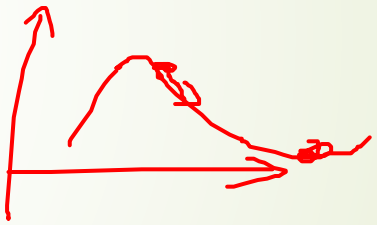
2. ① $\frac{\partial Loss}{\partial Y_p} \cdot \frac{\partial Y_p}{\partial W_i / \partial b} = C$

W (784x1) = $(W_1, \cdots W_{784})$

(2X392) (392X2) random $(2, 2, \cdots 2)$

$\vec{x} \cdot W + b = Y_p$

(2X2)

② 

$W_i = W_i - Lr \times C$

$b = b - Lr \cdot \frac{\partial L}{\partial Y_p} \times \frac{\partial Y_p}{\partial b}$

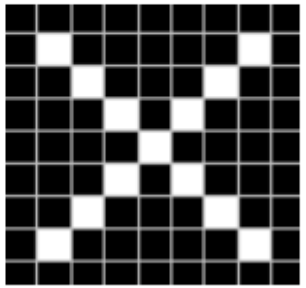
↓ Loss = $(Y_p - Y)^2$

MSE ✓

机器学习概念：卷积层、全连接层
池化层和遗忘层

卷积层：

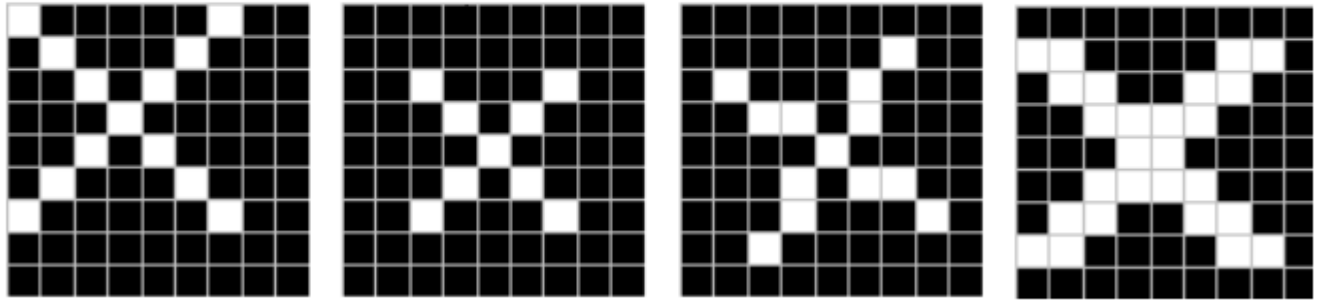
比如，我现在要训练一个最简单的CNN，用来识别一张图片里的字母是X还是O。



我们人眼一看，很简单嘛，明显就是X啊，但是计算机不知道，它不明白什么是X。所以我们给这张图片加一个标签，也就是俗称的Label，Label=X，就告诉了计算机这张图代表的是X。它就记住了X的长相。

卷积层：

但是并不是所有的X都长这样呀。比如说...

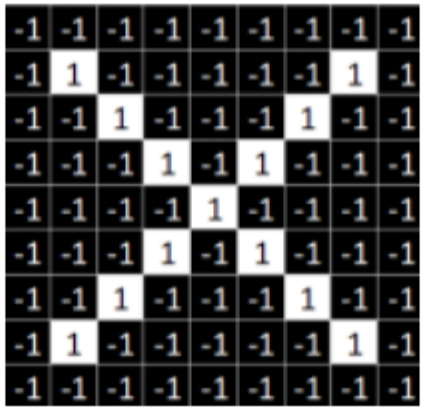


这四个都是X，但它们和之前那张X明显不一样，计算机没见过它们，又都不认识了。

卷积层：

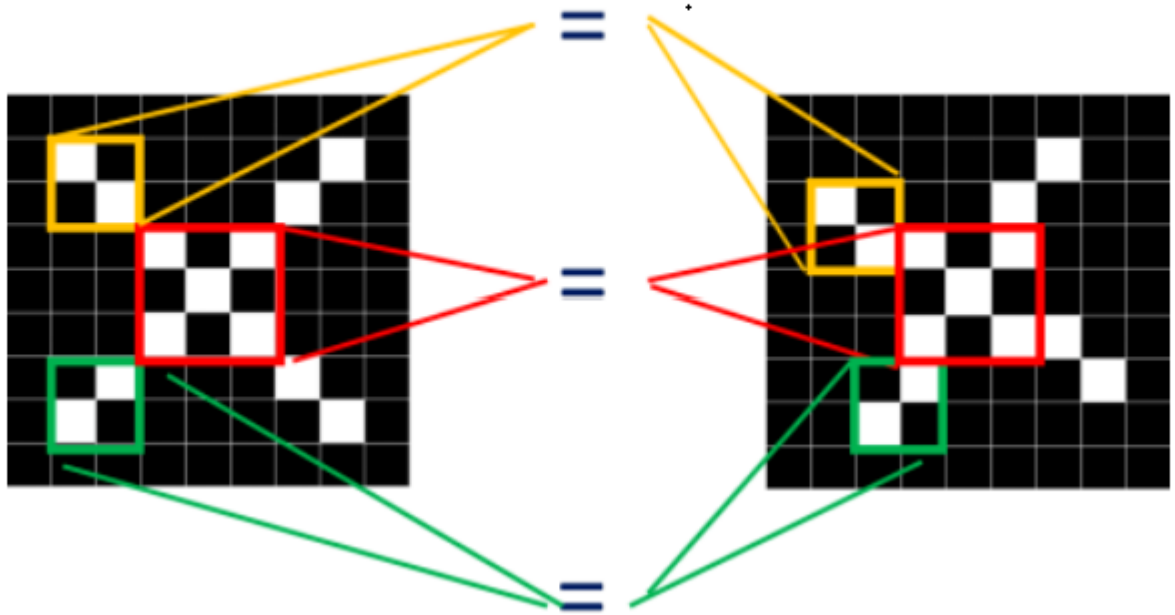
不认识了怎么办，当然是回忆看看是不是见过差不多的呀。这时候CNN要做的，就是如何提取内容为X的图片的特征。

我们都知道，图片在计算机内部以像素值的方式被存储，也就是说两张X在计算机看来，其实是这样子的。



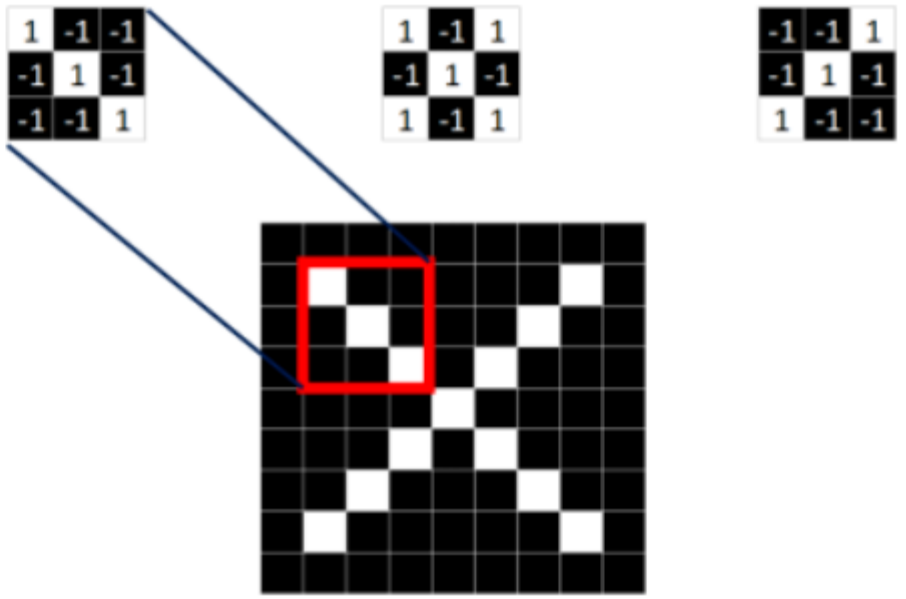
卷积层：

观察这两张X图，可以发现尽管像素值无法一一对应，但也存在着某些共同点。



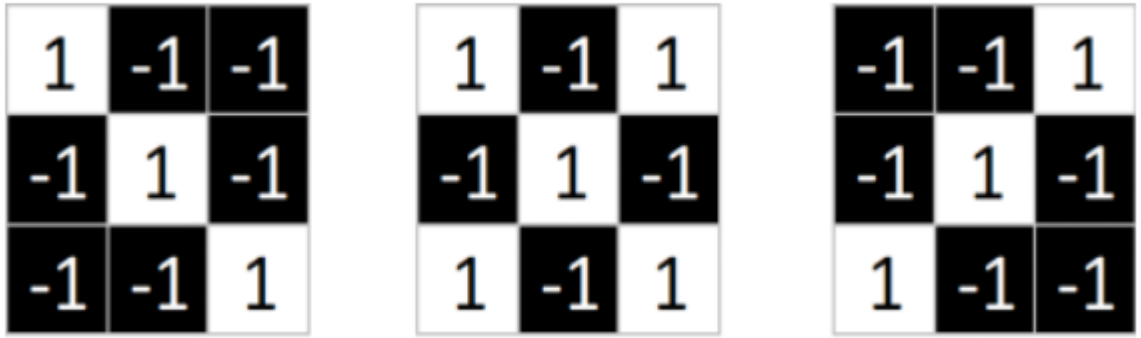
卷积层：

同理，从标准的X图中我们提取出三个**特征**（**feature**）



卷积层：

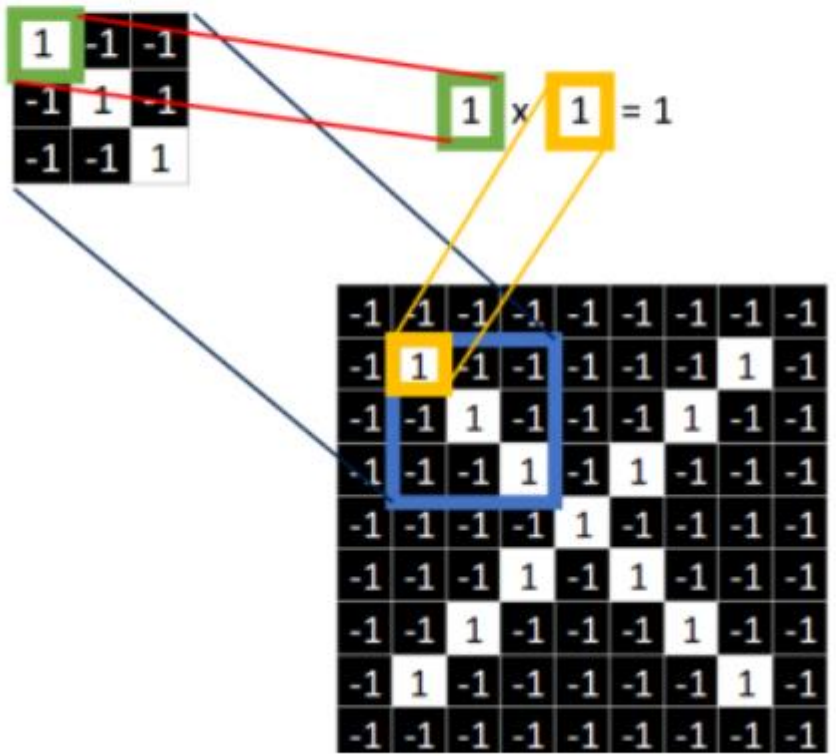
我们发现只要用这三个feature便可定位到X的某个局部。



feature在CNN中也被成为卷积核（filter），一般是3X3，或者5X5的大小。

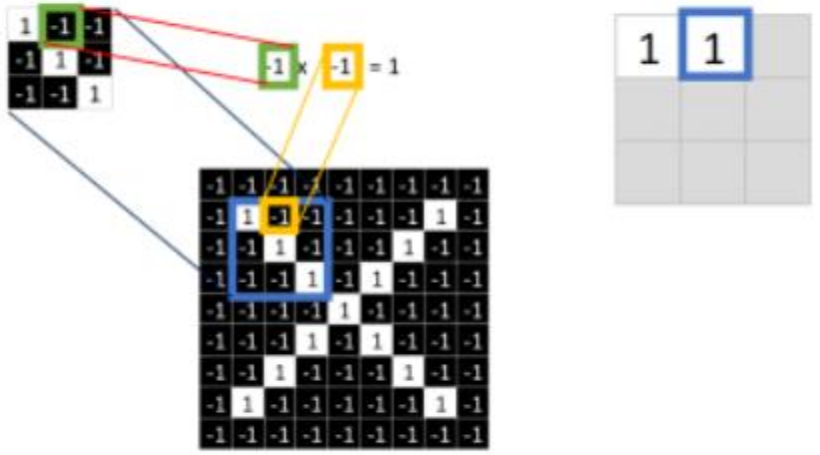
卷积层：

取 feature里的（1，1）元素值，再取图像上蓝色框内的（1，1）元素值，二者相乘等于1。把这个结果1填入新的图中。



卷积层：

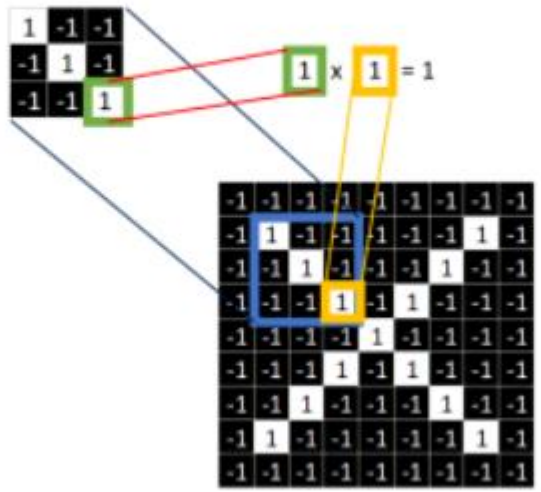
同理再继续计算其他8个坐标处的值



9个都计算完了就会变成这样。

卷积层：

9个都计算完了就会变成这样。

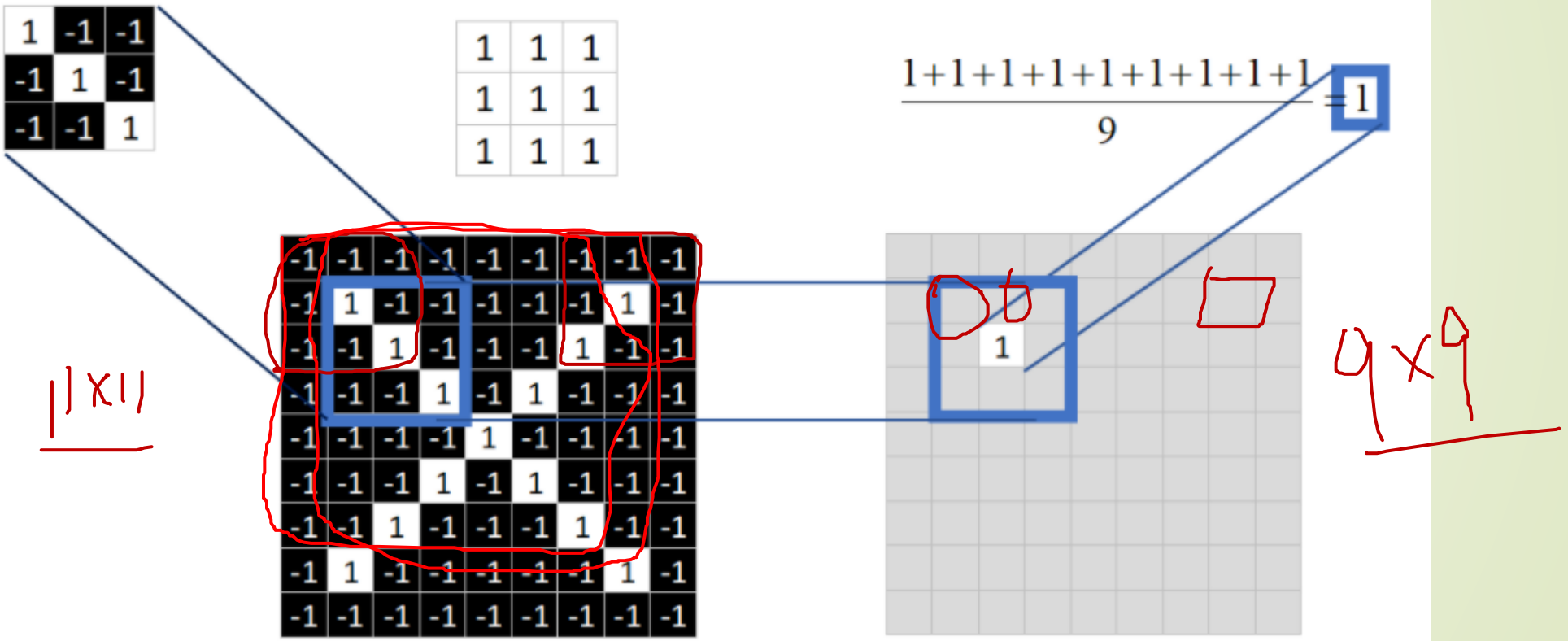


1	1	1
1	1	1
1	1	1

卷积层：

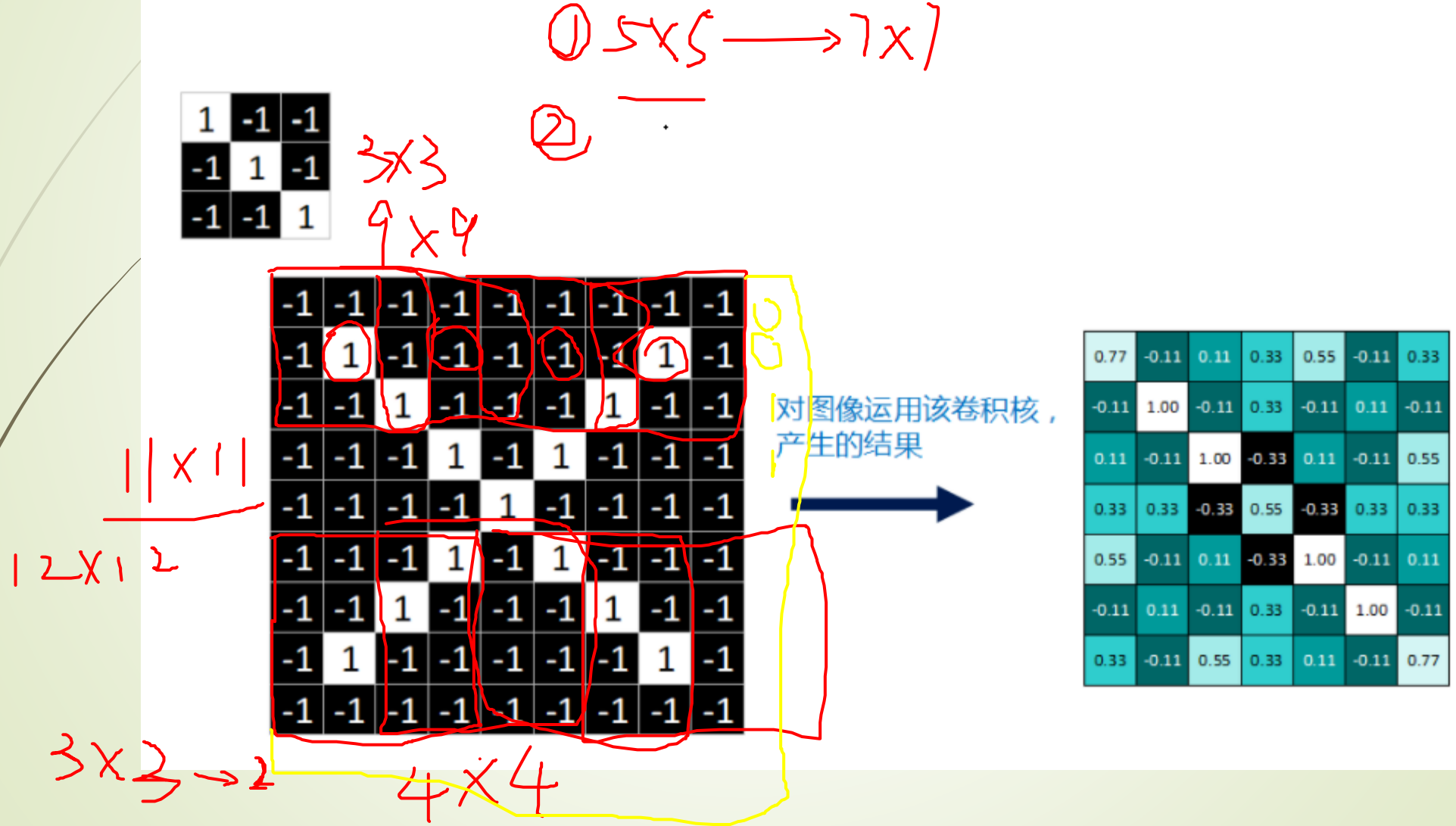
接下来的工作是对右图九个值求平均，得到一个均值，将均值填入一张新的图中。

这张新的图我们称之为 **feature map （特征图）**

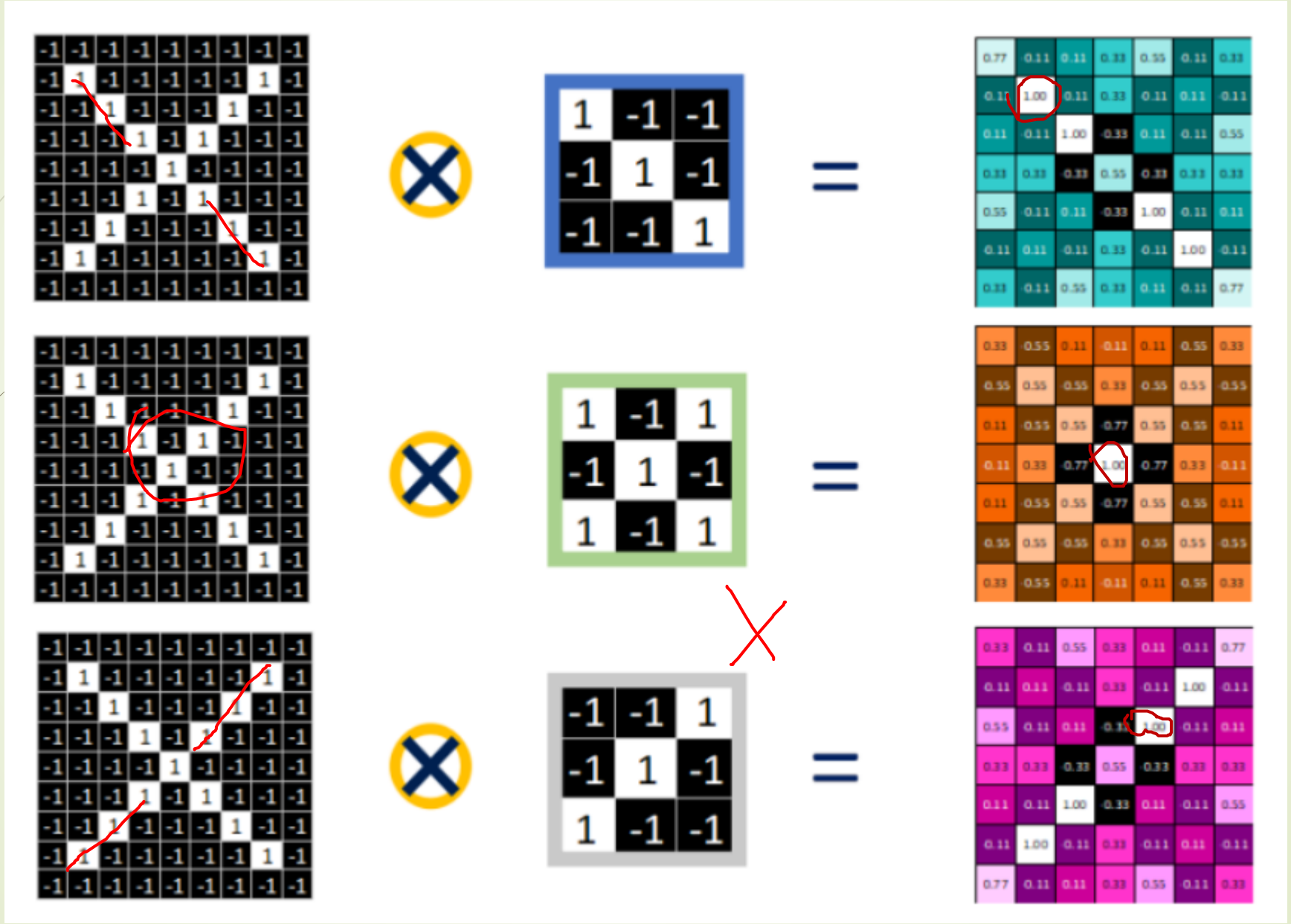


卷积层：

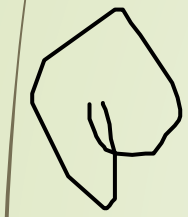
好了,经过一系列卷积对应相乘，求均值运算后，我们终于把一张完整的feature map填满了。



卷积层：



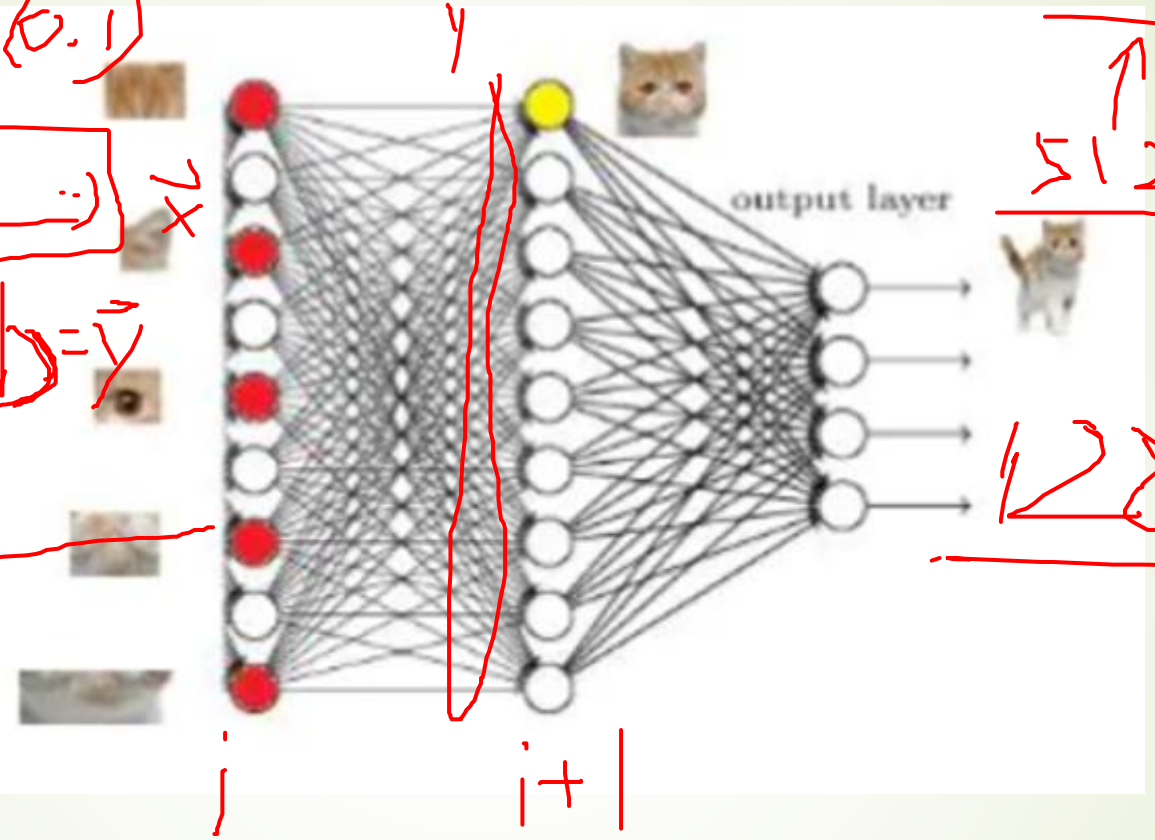
Conv2d层：



- Import torch.model as nn
- nn.Conv2d(in_channels, out_channels, kernel_size, stride=1,padding=0, dilation=1, groups=1,bias=True, padding_mode='zeros')

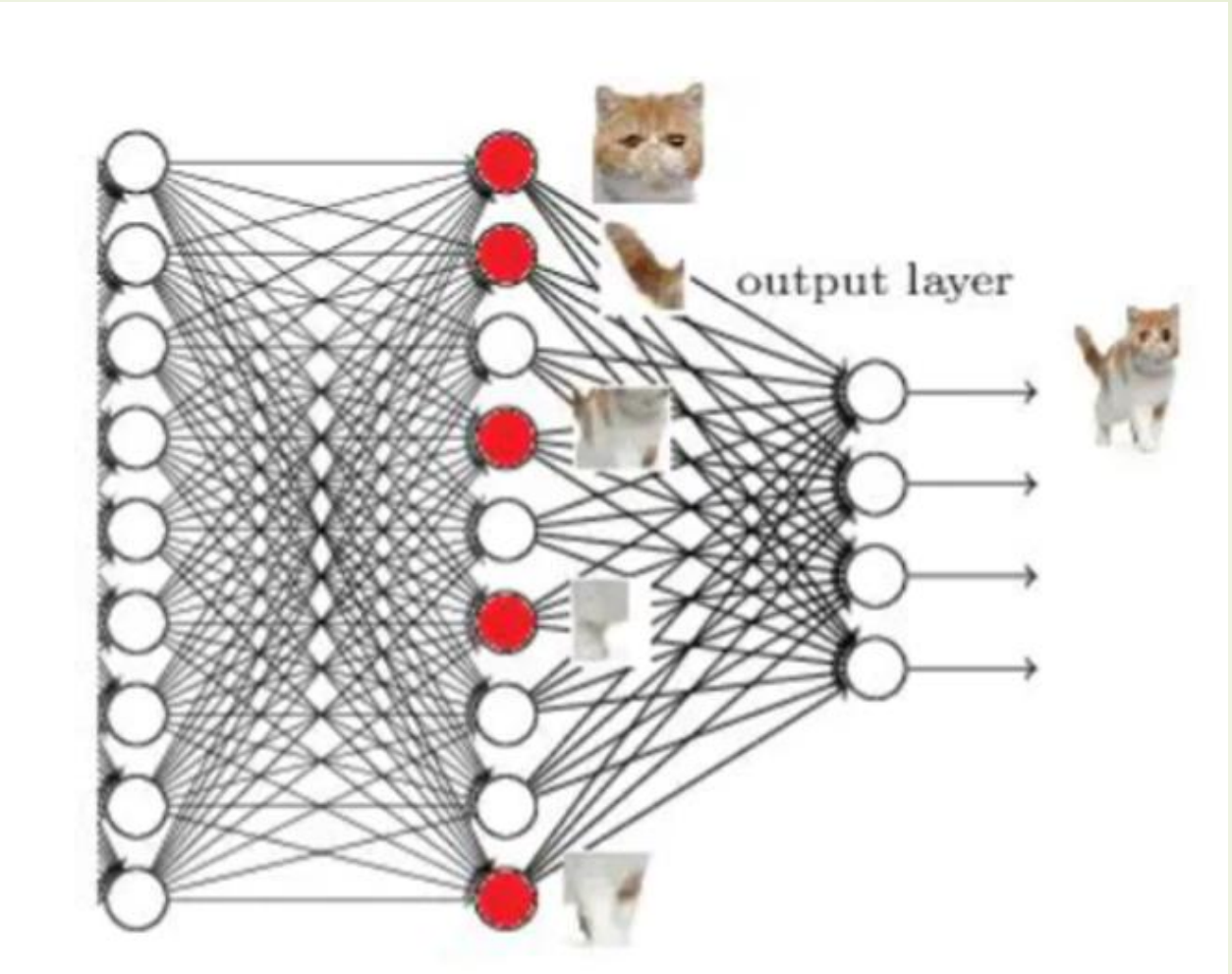
in_channels：输入的通道数目 【必选】
out_channels： 输出的通道数目 【必选】
kernel_size：卷积核的大小，类型为int 或者元组，当卷积是方形的时候，只需要一个整数边长即可，卷积不是方形，要输入一个元组表示高和宽。【必选】
stride： 卷积每次滑动的步长为多少，默认是 1 【可选】
padding： 设置在所有边界增加 值为 0 的边距的大小（也就是在 feature map 外围增加几圈 0 ），例如当 padding =1 的时候，如果原来大小为 3 × 3，那么之后的大小为 5 × 5。即在外围加了一圈 0。【可选】
dilation：控制卷积核之间的间距（什么玩意？请看例子）【可选】

全链接层：



红色的神经元表示这个特征被找到了（激活了）

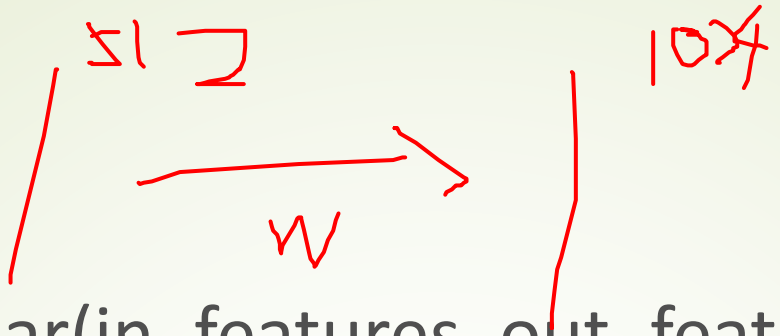
全链接层：



当我们把这些找到的特征组合在一起，发现最符合要求的是猫

ok，我认为这是猫了

FC code :



nn.Linear(in_features, out_features)

```
import torch.nn as nn
import torch
m = nn.Linear(20, 30)
input =
torch.autograd.Variable(torch.randn(1
28, 20))
output = m(input)
print(output.size())#[128, 30]
```

作业：